REDUCING THE AMORTIZATION GAP IN END-TO-END IMAGE COMPRESSION

Anonymous

Anonymous, Anonymous

ABSTRACT

End-to-end deep neural compression is about to exceed the performance of the traditional handcrafted compression techniques on videos and images. The core idea is to learn a non-linear transformation into latent space, jointly with an entropy model of the latent distribution. These methods enforce the latent to follow some prior distributions. Since the priors are learned by amortizing its parameters over the entire training set, it cannot fit exactly on every single instance. As a consequence, it damages the compression performance by enlarging the bitstream. In this paper, we proposed simple yet efficient instance-based parameterization methods to reduce this amortization gap at a minor cost. The proposed method is applicable to any end-to-end compressing methods and improves the state-of-the-art (sota) end-to-end compression bitrate more than 1% without any impact on the reconstruction quality.

***Index Terms*—** Neural Image Compression, Entropy Model, Amortization Gap.

1. INTRODUCTION

End-to-end deep compression methods can be seen as a special case of Variational Autoencoder (VAE) model as in [1], where the approximate posterior distribution is uniform distribution centered to the encoder's outputs (latents) at train time, has fixed variance output distribution and has trainable priors [2, 3, 4, 5, 6, 7]. It was shown that minimizing the evidence lower bound (ELBO) of this special VAE is equivalent to minimizing jointly the mean square error (MSE) of reconstruction and the entropy of latents w.r.t the priors [4]. At test time, these codecs encode quantized latents in lossless manner *w.r.t* priors into a bit-stream. It is typically performed by any entropy encoder such as range or arithmetic coding [8]. Since the decoder and the priors known at receiver side (or priors can be reconstructed by side information [4, 5, 6, 7]), quantized latents can be read back from bitstream by any entropy decoder (range decoder), then reconstruct the input data. The proposed models differ mainly by the modelling of priors: using either fully-factorized [3], zero-mean gaussian [4], gaussian [5, 6] or mixture of gaussian [7], where some predict priors using an autoregressive schema [5, 6, 7].

This paper is under review for ICIP2022.

All prior models are learned by amortizing its parameters over the entire training set. Consequently, this prior is sub-optimal for a given specific data instance. This problem is refereed to as the amortization gap [9]. Methods proposed to solve this problem are two folds: firstly, enforce that the latent of the given instance obeys the priors [10, 11, 12, 13, 14]; secondly, modify the priors to better fit the given instance's latents [15, 16, 17, 18]. The first class of methods does not need any update on the receiver side, but has limited gain. The second class of methods can update the encoder/decoder in addition to entropy model as well resulting more gain, at the additional cost of transmitting these updates to the receiver. However, all methods require post-training to overfit on a given data instance, which increases the encoding time significantly.

In this paper, we propose two main contributions: first, we define the amortization gap of entropy model in compressing perspective and report the amortization gap of some recent neural image compression model over benchmark datasets. Second, we propose simple yet efficient methods for factorized and hyperprior entropy models to adjust the priors to fit on any new instance to be compressed. Our solution does not need post-training and does not add computational complexity. We show that a gain of at least 1% can be expected from sota end-to-end compression method without any impact on the reconstruction quality.

2. NEURAL IMAGE COMPRESSION

In end-to-end image compression, the encoder $\mathbf{y} = g_a(\mathbf{x}; \phi)$ transforms each input $\mathbf{x} \in \mathbf{R}^{n \times n \times 3}$ into a lower dimensional latent $\mathbf{y} \in \mathbf{R}^{m \times m \times o}$ and quantize it to obtain $\hat{\mathbf{y}} = \mathbf{Q}(\mathbf{y})$. Later, the quantized latents are compressed losslessly by the entropy coder using factorized entropy model $p_f(\hat{\mathbf{y}}|\Psi)$ in [3]. However, if the entropy model of the latent $\hat{\mathbf{y}}$ is conditioned with side information to account for the spatial dependencies, the side information $\mathbf{z} = h_a(\mathbf{y}; \Phi)$ where $\mathbf{z} \in \mathbf{R}^{k \times k \times f}$ (and its quantization $\hat{\mathbf{z}} = \mathbf{Q}(\mathbf{z})$) is also learnt. In this setting, $\hat{\mathbf{y}}$ is encoded with the hyperprior entropy model $p_h(\hat{\mathbf{y}}|\hat{\mathbf{z}})$, and $\hat{\mathbf{z}}$ is encoded with factorized entropy model $p_f(\hat{\mathbf{z}}|\Psi)$. The decoder $\hat{\mathbf{x}} = g_s(\hat{\mathbf{y}}; \theta)$ converts the latents backs to the reconstructed image $\hat{\mathbf{x}}$. The parameters of g_a, g_s, p_f, p_h are obtained by

minimizing the following rate-distortion loss.

$$\mathcal{L} = \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p_x \\ \epsilon\sim U}} [-log(p_h(\hat{\mathbf{y}}|\hat{\mathbf{z}},\Theta)) - log(p_f(\hat{\mathbf{z}}|\Psi)) + \lambda.d(\mathbf{x},\hat{\mathbf{x}})] \quad (1)$$

Here, $d(.,.)$ is any distortion loss such as MSE, λ is the trade-off parameter to control compression ratio and quality, $\mathbf{Q}(.)$ is continuous relaxation at train time as $\mathbf{Q}(x) = x + \epsilon$, $\epsilon \sim U(-0.5, 0.5)$. In factorized entropy model (second term in (1)), each $k \times k$ slice of side latent has a trainable cumulative distribution function (cdf) in entropy model shown by $\bar{p}^{(c)}_{\Psi}(.),\ c = 1 \dots f$, and probability mass function (pmf) for a given value of x is derived as $\hat{p}^{(c)}_{\Psi}(x) = \bar{p}^{(c)}_{\Psi}(x + 0.5) - \bar{p}^{(c)}_{\Psi}(x - 0.5)$. Thus, the entropy model applies as follows;

$$p_f(\hat{\mathbf{z}}|\Psi) = \prod_{c=1}^{f} \prod_{i,j=1}^{k,k} \hat{p}^{(c)}_{\Psi}(\hat{\mathbf{z}}_{i,j,c}) \quad (2)$$

In hyperprior entropy (first term in (1)), each latent point is modeled as $1d$ Gaussian distribution and its pmf is $\hat{N}(x;\mu,\sigma) = \bar{N}(x + 0.5;\mu,\sigma) - \bar{N}(x - 0.5;\mu,\sigma)$. The hyperprior entropy model is written $p_h(\hat{\mathbf{y}}|\hat{\mathbf{z}},\Theta) = \prod_i \hat{N}(\hat{\mathbf{y}}_i;\boldsymbol{\mu}_i,\boldsymbol{\sigma}_i)$ at train time where $\boldsymbol{\mu},\boldsymbol{\sigma} = h_s(\hat{\mathbf{z}};\Theta)$ as in [4] or $\boldsymbol{\mu_i},\boldsymbol{\sigma_i} = h_s(\hat{\mathbf{z}},\hat{\mathbf{y}}_{<\mathbf{i}};\Theta)$ in autoregressive prediction in [5, 7]. h_s is a trainable model implemented as a neural network with parameter Θ. However, this implementation is not effective at test time due to the necessity of recalculating the pmf table at receiver side for each latent points i. Thus, it is common practice to use s number of predefined integer resolution pmf tables under zero means but different scale parameters (logarithmic distributed scale values between σ_{min} to σ_{max}) [4, 5, 6, 7]. As long as $\tilde{\mathbf{y}}_i = Q(\mathbf{y}_i - \boldsymbol{\mu}_i)$, $\hat{\mathbf{y}}_i = \tilde{\mathbf{y}}_i + \boldsymbol{\mu}_i$, σ_c is c-th predefined scale and $\mathcal{N}(\sigma_c)$ is a set of latent indices whose winning scale is σ_c. Thus, the hyperprior entropy model is implemented as follows at test time:

$$p_h(\hat{\mathbf{y}}|\hat{\mathbf{z}},\boldsymbol{\Theta}) = \prod_{c=1}^{s} \prod_{i\in\mathcal{N}(\sigma_c)} \hat{N}(\tilde{\mathbf{y}}_i;0,\sigma_c) \quad (3)$$

3. PROPOSED METHOD

In this section, we define the amortization gap of the entropy models and propose solutions to reduce it.

3.1. Amortization Gap of the Entropy Model

At training time, parameters $\phi, \theta, \Phi, \Theta$ and Ψ are estimated by optimizing $\mathcal{L}$ over the data distribution $\mathbf{x} \sim p_x$. The parameters may be optimal *in average* for entire dataset, but not any specific instance $\mathbf{x}$, which is known as an amortization gap [9]. Actually, the amortization gap in compression schema may occur for each trainable blocks in the model. However, we are here only interested in the gap for the entropy models.

Definition 1. *The amortization gap of the entropy model is the expected gain in bit length, if the entropy models' pmfs are optimal on every input instance. The gap can be calculated for the factorized entropy model as follows:*

$$\mathcal{G}_f = -log(p_f(\hat{\mathbf{z}}|\Psi)) + \log(p^*_f(\hat{\mathbf{z}})) \quad (4)$$

$p^*_f(\hat{\mathbf{z}})$ refers to the optimal entropy model within the same entropy family $\mathcal{P}_f$, i.e., $p^*_f(\hat{\mathbf{z}}) = argmax_{p_f\in\mathcal{P}_f} log\,(p_f(\hat{\mathbf{z}}))$. For the hyperprior entropy model, it reads:

$$\mathcal{G}_h = -log(p_h(\hat{\mathbf{y}}|\hat{\mathbf{z}},\Theta)) + log(p^*_h(\hat{\mathbf{y}})) \quad (5)$$

$p^*_h(\hat{\mathbf{y}})$ refers to the optimal entropy model within the same entropy family $\mathcal{P}_h$, i.e., $p^*_h(\hat{\mathbf{y}}) = arg\max_{p_h\in\mathcal{P}_h} \log\,(p_h(\hat{\mathbf{y}}))$.

In order to find the instance-specific optimal entropy model within the same family, one does not need to optimize the log-likelihood, since the normalized histogram is the optimal pmf [19]. The two following lemmas state that the actual latent histogram can be used.

Lemma 1. *The optimal factorized entropy model has the following closed form expression:*

$$p^*_f(\hat{\mathbf{z}}) = \prod_{c=1}^{f} \prod_{i,j=1}^{k,k} h^{(c)}_f(\hat{\mathbf{z}}_{i,j,c}) \quad (6)$$

where $h^{(c)}_f(x) = \sum_{i,j} \delta(x,\hat{\mathbf{z}}_{i,j,c})/k^2$ is the normalized frequency of symbol x on $k \times k$ slice of $\hat{\mathbf{z}}$ and $\delta(.,.)$ is the Kronecker symbol.

Lemma 2. *The optimal hyperprior entropy model has the following closed form expression:*

$$p^*_h(\hat{\mathbf{y}}) = \prod_{c=1}^{s} \prod_{i\in\mathcal{N}(\sigma_c)} h^{(c)}_h(\tilde{\mathbf{y}}_i) \quad (7)$$

where $h^{(c)}_h(x) = \sum_{i\in\mathcal{N}(\sigma_c)} \delta(x,\tilde{\mathbf{y}}_i)/|\mathcal{N}(\sigma_c)|$ is the normalized frequency of symbol x on $\tilde{\mathbf{y}}_i$, where $i \in \mathcal{N}(\sigma_c)$ and $|\mathcal{N}(\sigma_c)|$ is the number of elements in the given set.

3.2. Explicit Parameterization

Using lemmas 1 and 2, one could completely fill the amortization gap of the entropy model simply by replacing learned pmf tables with histogram of actual latents. However, the additional cost of transmitting these histograms should also be taken into account. In this paper, when an input image has to be encoded, we propose an explicit parameterization $\tilde{p}_{f|h}(.,\beta)$ of the pmfs for factorized and hyperprior entropy models. This approximation is illustrated in Fig. 1. This low-level approximation can be hopefully transmitted at a negligible cost, and aims at improving the encoding using $\tilde{p}_{f|h}(.,\beta)$. We propose two variants for approximating $\tilde{p}_{f|h}(.,\beta)$. The

first one is generic, uses Gaussian mixture model and is described in section 3.2.1. It is dedicated to the factorized entropy model, since this modeling is flexible. The second one is a simplified version, where the central bin is spread on neighbouring bins, and is described in section 3.2.2. This approach is dedicated to the hyperprior entropy model, since the discrepancy between learned and actual pmf is smaller.

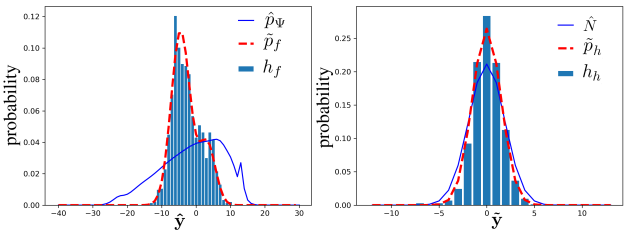

Fig. 1. Learned pmfs $(\hat{p}_\Psi, \hat{N})$, reparameterized pmfs $(\tilde{p}_f, \tilde{p}_h)$ and normalized frequencies (h_f, h_h) for a certain image's selected latent under factorized (left) and hyperprior (right) entropy models. Our reparameterization fits better on the normalized frequencies, leading to improved compression.

3.2.1. Truncated Gaussian Mixture on Discrete Support

Since the factorized entropy model is a non-parametric distribution model [3, 4], the pmf is flexible enough to have any shape. One of the most successful parametric distribution model is Gaussian Mixture Model (GMM) which can approximate any smooth density function [20] with a cost of three parameters per component. However, in our case, the function to be approximated is defined on integer center and support domain is truncated such that $[x_{min} \dots x_{max}]$. Thus, we can write our mixture model's pmf for factorized entropy model $\tilde{p}_f(x; \beta^{(c)}))$ as:

$$\tilde{p}_f(x; \beta^{(c)}) = \frac{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k)}{\sum_{z=x_{min}}^{x_{max}} \sum_{k=1}^{K} \pi_k N(z; \mu_k, \sigma_k)} \quad (8)$$

Here $\beta^{(c)} = \{\pi_k, \mu_k, \sigma_k\}_{k=1\dots K}$ denotes the set of parameters to be inferred in (8) for c-th latent band. Since there is no closed form solution of $\beta^{(c)*}$ that maximize (8), optimization is used for estimation. In practice, given the small number of parameters, convergence is fast. Fig 1 (left) displays the result of pmf approximation using a mixture of two Gaussians (K=2). In practice, the tuning of parameter K leads to a trade-off between approximation accuracy and transmission cost.

In some specific cases (e.g., the hyperprior entropy model), where the latent distribution has zero-mean and is unimodal, it is possible to set $K = 1$ and $\mu = 0$, also called zero-mean truncated Gaussian approximation. This is illustrated in Fig. 1(right) where $\tilde{p}_h(x; \beta^{(c)})$ obtained by equation (8) under $K = 1$ and $\mu = 0$.

3.2.2. Difference of the Center Bins Probability

We propose in this section a simpler alternative for the hyperprior entropy, since the latter is parametric and by construction has a zero-mean Gaussian shape. Hence, in that special case, we propose an alternative at minimal transmission cost. We use the heuristic of computing the error of center bins probability and spread this error to the other bins proportionally seems arguable strong closed form alternative. In this approach, reparameterized c-th pmf of hyperprior entropy model can be written in (9) where the parameter β is the error of the center bins probability between learned one and the actual one in the histogram such as $\beta^{(c)} = \hat{N}(0; 0, \sigma_c) - h_h^{(c)}(0)$.

$$\tilde{p}_h(x; \beta^{(c)}) = \begin{cases} \hat{N}(x; 0, \sigma_c) - \beta^{(c)} & \text{if } x = 0 \\ \hat{N}(x; 0, \sigma_c)(1 + \frac{\beta^{(c)}}{1 - \hat{N}(0; 0, \sigma_c)}) & \text{if } x \neq 0 \end{cases} \quad (9)$$

4. EXPERIMENTAL RESULTS

We used CompressAI library [21] to test our contributions on already implemented 6 SOTA deep compression methods, as well as a very recent method [22]. First, we measured the amortization gaps of pre-trained methods and our gains on Kodak test set [23]. Results are given in Table 1. Regardless of the baseline method, the factorized entropy model's amortization gap is quite large (%7.6-%11.8), compared to the hyperpriors one (%1.9-%4.5). This observation is also predictable by Fig 1 where it clearly shows that mismatch between h_f and $\hat{p}_\Psi$ is much more bigger than h_h and $\hat{N}$. This can be explained easily: the hyperprior entropy model uses instance specific information, and fully factorized model does not. The hyper-prior methods encode very small amount of the data (%0.6-%5.9) with less effective entropy model (factorized entropy), but vast majority of them (%94.1-%99.4) is encoded by effective entropy model (hyperprior entropy), in average their amortization gap (%1.9-%4.7) is smaller compared to the fully-factorized method (%9.5). From the different version of same method, it can be seen that when the amount of side information decreases, correspondingly the hyperprior gap increases. For instance, the hyperprior gap is %3.4 where there is %5.9 side information but %4.5 where there is %3.5 side information. It is worth to mention that due to the fixed extra parameter cost in our proposal, the efficiency of our method is proportionally less when the amount of encoded information decreases. For instance, we can reduce the factorized entropy gap from %9.5 to %6.79 when all information is encoded by this entropy model in **bmshj2018-factorized** model, where it can reduce the same gap to %2.77 when %2.3 of the information encoded by this entropy model in **cheng2020-attn**. Our proposed methods manage to save significant amount of bits from all studied methods in total.

In order to measure the performance of proposed method beyond the 24 images of the Kodak dataset, we also used

Table 1. Ratio of the encoded information, the amortization gap of the entropy models and our gain for each entropy model relative to the original bit-length for the methods trained lowest bpp objective. The gap and the gain is much more higher in factorized entropy. Our solution reduces the gap significantly both factorized and hyperprior entropy model. In total, proposed method saves more than %1 of file size from sota model. Results are averaged over Kodak Test set.

Model	Factorized Entropy			Hyperprior Entropy			Total	
	Ratio	Gap	Gain	Ratio	Gap	Gain	Gap	Gain
bmshj2018-factorized [3]	%100	%9.5	%6.79	-	-	-	%9.5	%6.79
bmshj2018-hyperprior [4]	%3.5	%10.0	%4.65	%96.5	%4.5	%1.84	%4.7	%1.98
mbt2018-mean [5]	%5.9	%11.4	%4.27	%94.1	%3.4	%1.24	%3.8	%1.43
mbt2018 [5]	%2.3	%9.9	%4.09	%97.7	%2.5	%0.99	%2.7	%1.06
cheng2020-anchor [7]	%1.2	%11.8	%3.68	%98.8	%2.3	%1.06	%2.5	%1.09
cheng2020-attn [7]	%2.3	%9.5	%2.77	%97.7	%1.9	%0.88	%2.1	%0.92
InvCompress [22]	%0.6	%7.6	%2.18	%99.4	%3.0	%1.32	%3.0	%1.33

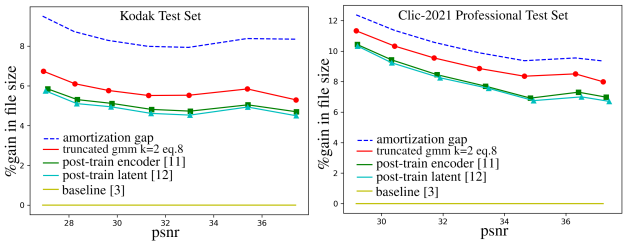

Fig. 2. Experimental results on Kodak and Clic Test Set for **bhshj2018-factorized** model in [3] trained on 7 different psnr objectives. Proposed method based on truncated GMM saves %6.8 original bit length for Kodak, %11.5 for Clic test test on lower psnr and outperformed post-train based high computational demanding alternatives.

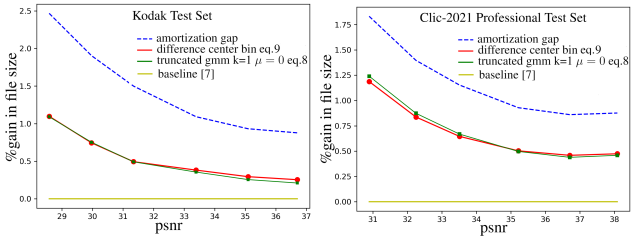

Fig. 3. Experimental results on Kodak and Clic Test Set for **cheng2020-anchor** model in [7] trained on 6 different psnr objectives. The gap and our gain decrease on higher psnr target. Proposed two methods save more than %1 of file size for both test sets at lower psnr. Our simple center bin difference solution gives comparable results even better at higher psnr.

60 images of the Clic-2021 Challenge's Professional test set [24]. First, we measured the amortization gap and show how our proposed explicit parameterization by truncated GMM reduces this gap for fully factorized entropy model [3] in Fig 2. We use $k = 2$ and write 5 parameters (2 mean, 2 scale and 1 relative weight quantized into 1024 bin), transmitted alongside the quantized latent, using 10-bit per parameter. In practice, a channel-wise selection mechanism is used to determine if the learned pmf should be replaced by the approximated one. We plugged our method on already trained model proposed in [3] provided by [21] for 7 different psnr target. According to results in Fig 2, the amortization gap varies from %8.5 to %9.5 in Kodak dataset where the proposed method gains from %5.3 to %6.8 in file size. In Click-2021 dataset, the gap (%9.5-%12.5) and our gain (%8-%11.5) are even bigger. To compare our method with the existing method, we also implemented two instance based post-training methods: post-train encoder (trains the encoder for given test image) [11], post-train latent (learns more effective instance's latent directly without training encoder)[12]. According to our test, we have found that [12] is faster (but still needs significant time to train) than [11] but with less performance as can be seen in Fig 2. Our proposal reaches better results and outperforms significantly compared to these two approaches even without giving any significant computational complexity.

To test our proposals for hyperprior entropy model described in section 3.2.2, we have selected the best neural compressing model **cheng2020-anchor** [7] provided in [21]. Models are trained for 6 different objectives on RD curve. According to results shown in Fig.3, our method saves more than %1 of original file size in lower bit-rate and save around %0.5 in highest bit-rate. The simplest approach that parameterizes the new probability by the difference between center bin's probability in Eq.9 gives competitive result even better in higher psnr with zero-mean Gaussian distribution.

5. CONCLUSION

We have proposed here to improve the efficiency of entropy models in deep neural image compression algorithm. First, we have defined the amortization gap for entropy models, and we measured experimentally the gap for sota methods. Then, we have proposed an effective and computationally-friendly method to fill the gap, which differs from previously published methods. We have shown experimentally that a gain above 1% was obtained, on different dataset and on 7 various SOTA compression methods. Our method can actually be applied to any end-to-end deep compression technique, without any impact on the reconstruction quality. One known lim-

itation of deep methods is reconstruction failure, due to architecture discrepancy (the architecture used at encoding and decoding differ). This is explained by the need of recomputing the pmf table at decoding, and small errors occur because of floating-value approximations [25]. In this regards, our method based on center-bin difference will be robust since it needs only integer divide operation. Future work may explore the link between explicit parameterization of the entropy model, and fine-tuning of the encoder. We may expect to increase the reconstruction quality, at constant bit-rate.

6. REFERENCES

[1] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," in *ICLR*, 2013.

[2] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár, "Lossy image compression with compressive autoencoders," in *ICLR*, 2017.

[3] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, "End-to-end optimized image compression," in *ICLR*, 2017.

[4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, "Variational image compression with a scale hyperprior," in *ICLR*, 2018.

[5] David Minnen, Johannes Ballé, and George D Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, vol. 31.

[6] David Minnen and Saurabh Singh, "Channel-wise autoregressive entropy models for learned image compression," in *ICIP*, 2020, pp. 3339–3343.

[7] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *CVPR*, 2020.

[8] J. Rissanen and G. Langdon, "Universal modeling and coding," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 12–23, 1981.

[9] Chris Cremer, Xuechen Li, and David Duvenaud, "Inference suboptimality in variational autoencoders," *ICML*, 2018.

[10] Caglar Aytekin, Xingyang Ni, Francesco Cricri, Jani Lainema, Emre Aksu, and Miska Hannuksela, "Block-optimized variable bit rate neural image compression," in *CVPR workshop*, June 2018.

[11] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao, "Content adaptive and error propagation aware deep video compression," in *ECCV*, 2020, vol. 12347, pp. 456–472.

[12] Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers, "Content adaptive optimization for neural image compression," in *CVPR Workshops*, June 2019.

[13] Yibo Yang, Robert Bamler, and Stephan Mandt, "Improving inference for neural image compression," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., 2020, vol. 33, pp. 573–584.

[14] Tiansheng Guo, Jing Wang, Ze Cui, Yihui Feng, Yunying Ge, and Bo Bai, "Variable rate image compression with content adaptive optimization," in *CVPR Workshops*, June 2020.

[15] Jan P. Klopp, Liang-Gee Chen, and Shao-Yi Chien, "Utilising low complexity cnns to lift non-local redundancies in video coding," *IEEE Transactions on Image Processing*, vol. 29, pp. 6372–6385, 2020.

[16] Yat-Hong Lam, Alireza Zare, Francesco Cricri, Jani Lainema, and Miska M. Hannuksela, "Efficient adaptation of neural network filter for video compression," in *ACM International Conference on Multimedia*, 2020, MM '20, p. 358–366.

[17] Nannan Zou, Honglei Zhang, Francesco Cricri, Hamed R. Tavakoli, Jani Lainema, Miska Hannuksela, Emre Aksu, and Esa Rahtu, "L2c – learning to learn to compress," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 2020, pp. 1–6.

[18] Ties van Rozendaal, Iris AM Huijben, and Taco S Cohen, "Overfitting for fun and profit: Instance-adaptive data compression," in *ICLR*, 2021.

[19] Marc Mezard and Andrea Montanari, *Information, physics, and computation*, Oxford University Press, 2009.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, 2016.

[21] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, 2020.

[22] Yueqi Xie, Ka Leong Cheng, and Qifeng Chen, "Enhanced invertible encoding for learned image compression," in *Proceedings of the ACM International Conference on Multimedia*, 2021.

[23] Eastman Kodak, "Kodak Lossless True Color Image Suite (PhotoCD PCD0992)," .

[24] "CLIC: Challenge on learned image compression," http://compression.cc

[25] Johannes Ballé, Nick Johnston, and David Minnen, "Integer networks for data compression with latent-variable models," in *ICLR*, 2019.