Training Heuristics for Rate-Constrained Learning-based Image Compression

Nilson D. Guerin Jr.[⋆], Renam Castro da Silva[†], Matheus C. de Oliveira[⋆], Henrique C. Jung[⋆], Luiz G. Martins[⋆], Pedro Sanches[⋆], Eduardo Peixoto[⋆], Bruno Macchiavello[⋆], Edson M. Hung[⋆], Vanessa Testoni[†], Pedro Garcia Freitas[†]
[⋆]Universidade de Brasília, [†]Samsung R&D Institute Brazil
nguerinjr@gmail.com, renam.cs@samsung.com

***Abstract*—In this work, we present a modification to the loss function in a variational image autoencoder. The modification aims to solve the problem of encoding with a rate constraint. Rate control is an open problem in end-to-end learning-based image codecs. In a real application, encoding to a specific rate can be very common and sometimes a requirement. Our proposal creates a network trained for encoding at a specific bitrate. Results have shown that our modification can achieve rate constrained encoding with minimal losses in MS-SSIM.**

***Index Terms*—rate control, target rate, learning-based image compression, convolutional neural network**

I. Introduction

Image compression is an essential task in digital communication, broadcasting, and storage. Traditionally, it has been based on techniques such as transform coding, prediction techniques, and scalar quantization. These techniques were so successfully adopted in the design of image compression methods that they were employed in several standards. For instance, JPEG-1 [1], a DCT-based codec, has prevailed in the field of practical lossy image compression since its introduction in 1992. Its successor, JPEG 2000 [2], has been used in digital cinema, medical applications, and other professional markets. Other image coding standards include BPG [3] and JPEG XR [4], targeting better compression than JPEG-1 with limited complexity overhead. These codecs, although quite efficient at the time they were released, are somewhat obsolete nowadays. These standards have performance limitations for High Dynamic Range (HDR) images. To overcome this limitation, the JPEG committee released the JPEG XT [5], a backward-compatible standard supporting compression of HDR images. Recently, the JPEG committee also launched the JPEG XL [6] standardization initiative, targeting a next-generation image coding system with outstanding rate-distortion performance for High Definition (HD) and Ultra HD (UHD) images with useful features for web distribution and a wide spectrum of applications.

Recently, image codecs based on Deep Neural Networks (DNN) have attracted great attention: they have steadily improved, from barely reaching JPEG-1 performance to methods that can surpass BPG [7]–[10]. Not only that, but it is also hoped that they will be adapted more easily to HDR and HD images, as they don't rely heavily on hand-crafted techniques like more traditional codecs.

Toderici et al. [11] proposed one of the first architectures for encoding images using Recurrent Neural Networks (RNN) based on a Long Short-Term Memory (LSTM) architecture in combination with fully-connected and convolutional residual autoencoders. The residuals at each iteration can be further encoded to achieve higher bitrates and better quality, which presents a special characteristic of being scalable and not requiring multiple training to reach the desired bitrate. Minnen et al. [12] improved Toderici's model by including an intra-prediction stage before encoding. Similarly, Jung et al. [13] enhanced Toderici's work by adding a block-based multi-mode intra prediction. Although these RNN-based codecs are flexible in terms of rate, they have significant computational complexity and limited compression performance.

GAN-based codecs were first proposed by Rippel & Bourdev [14], achieving outstanding results in terms of rate-distortion with quality measured by MS-SSIM. Agustsson et al. [10] proposed a GAN-based framework for image compression and presented a thorough study of such a framework for full-resolution image compression.

Convolutional Neural Networks (CNN)-based codecs are particularly modeled using Convolutional AutoEncoders (CAEs) [15], [16]. Among CNN-based codecs in literature, Ballé et al. [17], [18] proposed one of the most pertinent methods in terms of rate-distortion performance with an end-to-end autoencoder model. This model was advanced in [8] by introducing a hyperprior that captures spatial dependencies and uses it in the entropy model for further reduction in rate. Several works [19]–[21] have been done to improve upon Ballé's model.

Despite the interest and advancements in the field of learning-based image compression, these methods have not been widely deployed in practical applications. Some reasons include diffidence in metrics, lead time required to establish and concretize a novel standard, and run-time requirements. In order to overcome these issues, the Joint Photographics Experts Group (JPEG) committee announced the JPEG AI initiative to explore studies on the use of learning-based solutions for its standards [22]. Among the various challenges to be overcome in defining a robust learning-based image compression standard, the complexity reduction of neural compression

Part of the results presented in this work was obtained through the *Deep Codec* project funded by Samsung Eletrônica da Amazônia Ltda under the Brazillian Informatics Law 8.248/91. BM thanks CNPq PQ 308548/2018-3.

networks is crucial to enable real-time applications. Another problem to be surpassed is the definition of bit allocation and rate control for DNN-based codecs. Toderici's proposal [11] enables the user to define a target rate, however, this is performed in increasing fixed steps, being a complex solution to be adopted in real applications. Alternatively, Ballé's model requires the network to be trained for each operation point. But, each trained model may output a significantly different quality and rate for different images. In this paper, we propose a modification of the loss function to introduce rate control. Our method allows constraining the bitrate by defining a target bitrate and controlling the variation around that target rate. This modification can be adopted in several CNN-based architectures.

II. The JPEG AI Call for Evidence

The objective of the Learning-based Image Coding Challenge (and JPEG Call for Evidence) is to objectively and subjectively evaluate learning-based image coding solutions to demonstrate the potential of this coding approach. Certain requirements were outlined for the challenge. A training and validation data set were made available on March 2020, nevertheless, other image sets can be used for training as well. The test image set was hidden and only made available after the submission of the decoder. Each image on the test set is required to be encoded at certain target bitrates. Such bitrates are 0.06, 0.12, 0.25, 0.50, 0.75, 1.00, 1.50, and 2.00 bits per pixel (bpp). The maximum deviation from those rates should not exceed 15%. These requirements help to stress the importance of rate control for learning-based image coding.

III. Proposed Method

Typically, the network is optimized using a traditional Lagrangian rate-distortion function:

$$J = D + \lambda \cdot R \tag{1}$$

where D is a distortion measure, R is the rate and λ is the Lagrangian multiplier controlling the exchange between rate and distortion.

The optimization algorithm attempts to find parameters for the model that minimizes the cost function over a large training dataset, thus both distortion and rate need to be evaluated in the course of training. Evaluating the distortion comes down to computing the chosen distortion measure between the raw input image and its reconstruction. As for the rate, one needs an entropy model for the latent representation.

Gradient descent optimizers rely on the differentiability of operations in the model, so quantization poses an obstacle that needs to be handled. To enable training using backpropagation, a differentiable function is used to approximate the actual quantization that takes place at the inference stage. The entropy model for the latent representation is modeled in several ways in the literature with different degrees of sophistication and computational cost. Usually, different rate-distortion operation points are obtained by training the network with different values of the λ parameter. Thus, each operation point has its own model, with its own set of weights for the network.

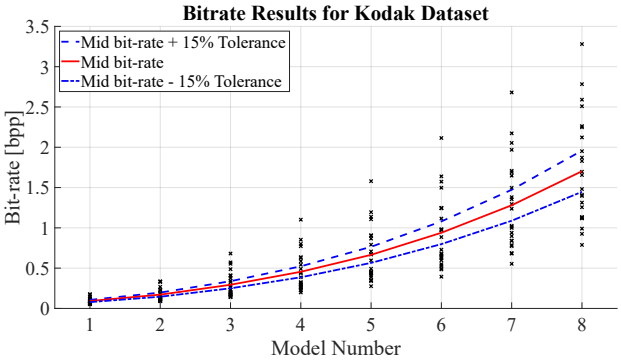


Fig. 1. Results of the Models provided by Ball et al. in the Kodak Dataset.

Fig. 1 shows the results for Kodak dataset [23] from Ball trained models [24]. It shows the variation in spent bitrate using each of the 8 pre-trained models. Although these models were not trained with a specific target bitrate in mind, it can be seen that the same model produces very different bitrates depending on the image being encoded. While it is true that the models are ordered for all images, i.e., model n yields a larger bitrate, and higher PSNR, than model k, provided $n > k$. Each model shows a large variance in terms of bitrate and quality. For instance, model 1 ranges from 0.053 to 0.177 bpp (3.3 times), whereas model 5 ranges from 0.276 to 1.579 (5.7 times). This is a typical behavior of image and video codecs, and it is observed by several conventional codecs.

If we need to achieve a certain target bitrate, such as those required by the JPEG-AI challenge, a straightforward solution would be to encode each image with multiple models and choose the model that yields the target bitrate for that particular image. Since the models are usually ordered, a binary search could be employed to avoid testing all models. Still, for a given target bitrate, a decoder would need to have all possible models. In the rest of this section, we will show the modifications we are proposing on this loss function in order to overcome this issue.

A. Modifying the Loss

In this work, we proposed the use of a constrained cost function to drive the rate operation point R resulted from training to attain a given target rate R_t. To this end, we devised a simple cost function that penalizes rate deviations from the target rate.

$$J = D + \beta \cdot f(R, R_t) \tag{2}$$

where,

$$f(R, R_t) = \left(\frac{R_t - R}{R_t} \right)^2 \tag{3}$$

Referring to eq. 3, the greater the deviation of R from R_t, the greater the penalty to the cost function. In the extreme case where R equals R_t, the rate does not contribute to the

cost function and the model attempts to minimize only the distortion. In addition to the mentioned analytical properties, experimentally we found this function yields well-behaved gradients. As will be shown experimentally, the parameter β varies according to the target rate and acts as a knob to control the variation of achieved rate around the target rate.

B. Training procedure to achieve the target rate

The proposed loss is defined by two hyper-parameters: β and R_t. R_t defines the target rate during training, and β controls the deviation from the target rate. Both parameters need to be set taking into consideration the architecture and convergence of the network. During training, the network may not reach the specified rate, and then R_t needs to be adjusted accordingly, and β needs to be adjusted depending on the value set for R_t and the allowed deviation from the rate. A procedure to reach the desired rate and variance is depicted in Figure 2:

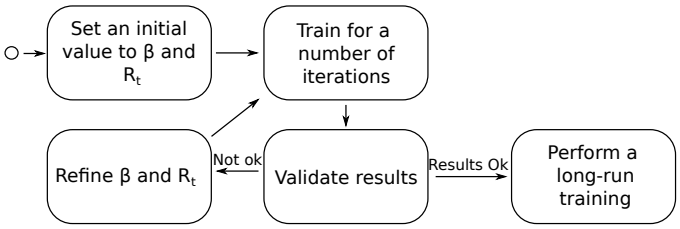


Fig. 2. Flowchart of the heuristics to reach a target rate and variance.

The first step in the heuristic is to define an initial value to β and set the target R_t to the desired rate value. This initial value of β can be any positive number, but a good start is to specify a low value (e.g. $\beta = 1$). Once the parameters are defined, the next step is the training of the model for a desired number of iterations.

Empirical results suggest that the networks can converge quite fast to the defined mean rate. In the initial iterations, a good convergence can take more time, but in the later adjustments of parameters, 50K iterations are enough to see the effects of the change of hyper-parameters. Nevertheless, this value may change for different architectures.

Once the network is trained, it is necessary to use a validation dataset to evaluate the mean rate obtained and the variance of the rate. If the results are not satisfactory, it may be necessary to adjust R_t and β. Changes in R_t will shift the obtained mean. For example, if R_t was set to 0.25 bpp and the results reached a mean of 0.2, it may be needed to set $R_t = 0.3$ to reach a real mean of 0.25. This shift is discussed in Subsection IV-B for one of the baseline architectures we used to show the effectiveness of the proposed solution. When adjusting β, the rule is simpler: β is increased if less variance is required or decreased otherwise. It may be useful to increase (or decrease) β in smaller steps if a fine control of variance is desired. As mentioned earlier, β is dependent both on the rate and on the network architecture. Once the new parameters are established, another round of fine-tuning is performed and the results are evaluated again, as depicted in Fig. 2. This process finishes when the rate is confined within the allowed rate fluctuation around the target rate. After that, a final training, with enough number of iterations, is performed to make the network more generalized.

IV. Baseline architectures

To show the effectiveness of the proposed loss function, we use two model architectures, which are distinguished by the devised underlying entropy model.

A. Nonparametric distribution model

The nonparametric architecture used in this work was introduced in [18]. It is a variational autoencoder that models directly the rate-distortion trade-off in the loss. Quantization is modeled in training time as additive uniform noise. The latent is modeled by P_q, a nonparametric factorized distribution defined as:

$$p_{\tilde{\mathbf{y}}|\psi}(\tilde{\mathbf{y}}|\psi) = \prod_i \left(p_{y_i|\psi^{(i)}}(\psi^{(i)}) * \mathcal{U}(-1/2, 1/2) \right)(\tilde{y}_i) \qquad (4)$$

where ψ are the splines used to model the distribution of the latent $\mathbf{y}$ and $\tilde{\mathbf{y}}$ is the latent after addition of the uniform noise.

B. Parametric distribution model with hyperprior

The architecture is an extension of the work [18] and was firstly proposed in [8]. The model includes a variational hyperprior, whose latent $\mathbf{z}$ is considered as side information and is modeled using the nonparametric distribution. The output of this hyperprior is the standard deviation of each element of the main latent $\mathbf{y}$, modeled by a Gaussian:

$$p_{\tilde{\mathbf{y}}|\tilde{\mathbf{z}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{z}}) = \prod_i \left(\mathcal{N}(0, \tilde{\sigma}_i^2) * \mathcal{U}(-1/2, 1/2) \right)(\tilde{y}_i) \qquad (5)$$

where $\tilde{\sigma}_i$ is the standard deviation given by the hyperprior for each latent element.

An extension of [8] was proposed in [9], where the main contribution is an autoregressive model designed to help the hyperprior estimate the mean and standard deviations. However, they also suggest a version where the hyperprior alone produces both parameters of the Gaussian, without an autoregressive component, which we use in this work as one of our baselines (i.e., the parametric model). Although using an autoregressive model greatly improves model performance, encoding and decoding become "unfeasibly" slow, and we, therefore, have chosen not to use it.

Remarks on rate shift between training and inference

Recall that during training the quantization step needs to be replaced by a differentiable approximation to allow training using backpropagation. In the baseline architecture with hyperprior, the raw latent representation $\mathbf{y}$ is scalar quantized at the inference stage:

$$\hat{\mathbf{y}} = \left\lfloor \mathbf{y} - \boldsymbol{\mu} + \frac{1}{2} \right\rfloor + \boldsymbol{\mu} \qquad (6)$$

$\lfloor \cdot \rfloor$ is the floor function and $\boldsymbol{\mu}$ is mean vector.

During training, quantization is modeled as random noise. To this end, an uniformly distributed random noise is added to the raw latent representation $\mathbf{y}$ giving rise to the noisy

latent $\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{u}(-1/2, 1/2)$. As the target bitrate is lowered, the estimated rate of the latent $\tilde{\mathbf{y}}$ shifts from the rate of the quantized latent $\hat{\mathbf{y}}$. This is because as the training target rate moves towards lower bitrates, the latent representations become less and less uncertain (low entropy). The Gaussian conditional distributions modeling the latent elements get more and more peaked around their mean, with a narrower scale of variation. In this scenario, the difference between the estimated rate computed using the noisy latent $\tilde{\mathbf{y}}$ and the estimated rate using the quantized latent $\hat{\mathbf{y}}$ becomes noticeable. Consider, for instance, a latent element y_i with scale parameter $\sigma_i < 1/2$ and mean $\mu_i = 0$. After quantization, the latent element $\hat{y}$ will be zero, and the likelihood will be close to 1. On the other hand, the noisy latent element $\tilde{y}_i$ will lie in $[-1, 1]$. The likelihood will be given by $cdf(\tilde{y}_i + 1/2) - cdf(\tilde{y}_i - 1/2)$ [8]. For $\tilde{y}_i \neq 0$, the likelihood will deviate from 1, resulting in the rate being overestimated during training. The same line of argumentation may be applied to latent elements with nonzero mean. This becomes more and more evident at lower bitrates as the useful (nonzero entropy) portion of latent is shrank.

V. Results

All of our simulations (training and evaluation) were performed using the following setup:

- CPU setup: Intel i7 8700 with 3.2 GHz, 64 GB RAM
- GPU setup: NVIDIA 2080Ti GPU from GIGABYTE

In order to train the models we have used 256×256 non-overlapping patches from the following public available image databases: (i) CLIC Professional dataset [25] (628 images); (ii) CLIC Mobile dataset [25] (1111 images); (iii) DIV2K dataset [26] (902 images); (iv) Ultra-Eye Ultra HD dataset [27] (41 images); (v) MCL-JCI [28], [29] (51 images); and (vi) FLICKR2K dataset [30] (2650 images). It is important to note that the rate-distortion (RD) performance of the proposed method is limited to the performance of the unconstrained baseline architecture.

We trained 8 networks (as described in sec. III-B) to achieve each of the 8 target bitrates with a maximum of 15% deviation, as established by the challenge. We trained our networks with only 500K iterations to show that our proposal can achieve desired results without the need for exhaustive training.

A. Target rate and deviation analysis

First, it is important to show that the proposed methodology yields the expected results. Hence, using the Kodak dataset, we run similar experiments like the ones provided in Fig. 1. We introduce the proposed modified loss function in the parametric architecture. The results are presented in Fig. 3.

It can be seen that the submitted model produced images within specified target bitrates, even for the more challenging low bitrates (where the 15% margin gets very small).

B. β Variation

The parameter β controls both the capacity of the loss of achieving the target bitrate and the spread around this bitrate. We have trained models using both the parametric and the non-parametric models, varying the parameter β attempting to achieve a target bitrate of 0.25 bpp and 1.0 bpp. These results are shown in Table I for the Kodak dataset. It can be seen that, as expected, allowing a larger spread around the target bitrate results in better RD results.

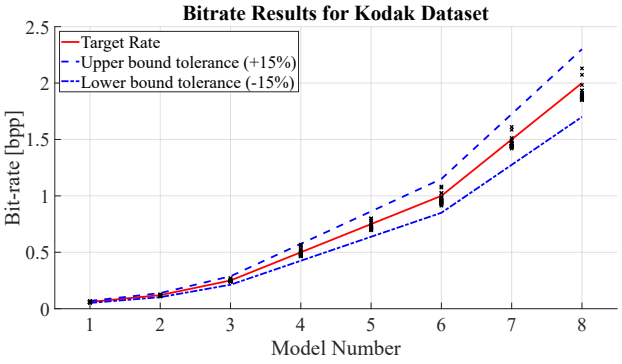


Fig. 3. Results of our submitted models in the Kodak Dataset.

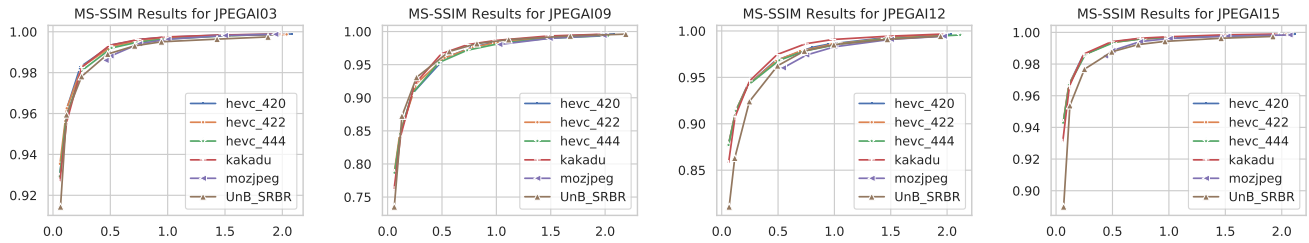
TABLE I
Results of β Experiments.

Model	Target	β	Average BPP	Average MS-SSIM	Range BPP	Range MS-SSIM
Non Parametric	0.25	10^4	0.22	0.951	[0.21, 0.25]	[0.900, 0.974]
		10^3	0.24	0.953	[0.20, 0.30]	[0.915, 0.976]
		10^2	0.27	0.954	[0.19, 0.44]	[0.930, 0.974]
	1.00	10^4	1.00	0.990	[0.97, 1.07]	[0.983, 0.995]
		10^3	1.00	0.989	[0.84, 1.33]	[0.986, 0.994]
		10^2	1.12	0.992	[0.76, 1.88]	[0.989, 0.996]
Parametric	0.25	10^4	0.23	0.951	[0.21, 0.25]	[0.891, 0.970]
		10^3	0.26	0.955	[0.17, 0.43]	[0.925, 0.978]
		10^2	0.36	0.974	[0.19, 0.74]	[0.961, 0.987]
	1.00	10^4	0.97	0.991	[0.95, 1.01]	[0.982, 0.995]
		10^3	0.93	0.990	[0.69, 1.33]	[0.986, 0.994]
		10^2	1.14	0.994	[0.61, 2.00]	[0.992, 0.997]

C. Performance using the Provided Test data set

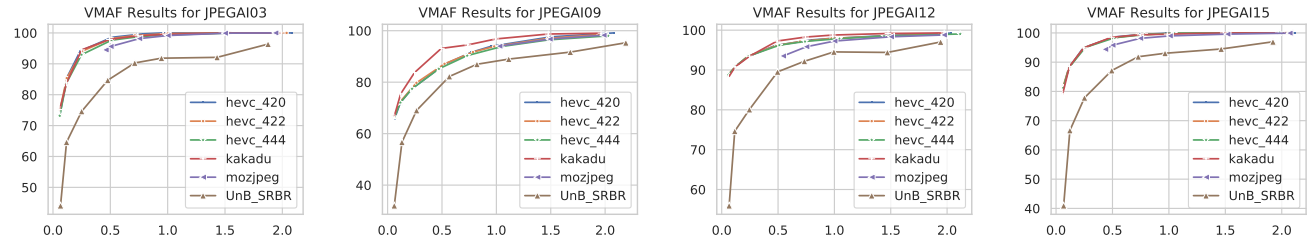
The trained models were evaluated using the 16 test images provided in the challenge. These images were encoded with the following conventional image encoders: HEVC intra using YUV format 4:4:4, 4:2:2, and 4:2:0; JPEG2K (Kakadu implementation), JPEG (Mozjpeg implementation). MS-SSIM metrics were calculated using only the luminance (Y) channel as defined by the challenge, and VMAF using YUV 4:4:4. The QP parameters were varied until the bpp for each image was within the desired range. No rate control was used for these codecs since such a tool is not standardized. The test images were also encoded using the parametric model with the proposed modification (UnB_SRBR). The trained models achieved correct rates for 127 of the 128 reconstructed images.

In Fig. 4 and Fig. 5 the RD results using MS-SSIM and VMAF metrics are presented, respectively for 4 images. As it can be observed, for MS-SSIM, the proposal is very competitive. Meanwhile, for the VMAF metrics, the losses were more significant.



(a) JPEGAI03 (b) JPEGAI09 (c) JPEGAI12 (d) JPEGAI15

Fig. 4. MS-SSIM results for 4 images of the Test Set.



(a) JPEGAI03 (b) JPEGAI09 (c) JPEGAI12 (d) JPEGAI15

Fig. 5. VMAF results for 4 images of the Test Set.

In Table II, we provide the average metrics for the conventional standards and our proposal. Note that our bitrate variance (Var. BPP) is similar to all standards. The average MS-SSIM from the proposed model is also comparable to the standards. For the VMAF metric, there are significant losses at low rates. This is mainly due to the fact that the distortion considered in our loss function was the MSE. The distortion function will need to be modified to achieve better results in terms of VMAF.

Regarding encoding and decoding time, we noticed the time variation to be small for the different target bitrates. On CPU, the average encoding time was between 8.82s and 8.4s, whereas decoding time was between 9.39s and 9.57s. On GPU, the average encoding time was between 10.42s and 10.46s, whereas decoding time was between 12.53s and 12.78s. It is worth mentioning that these figures include time for loading the model and for graph allocation. Our GPU's memory did not fully support the model, so the potential parallelism was hampered by the need to reallocate operations.

VI. Conclusion

In this paper, we focus on an open problem for learning-based image codecs: rate constrained encoding. We propose a modification to the loss function that takes into account a target rate. This modification can be applied to various codecs based on CNNs. We introduce a parameter β that can be used to set the maximum deviation from a target rate. As with any image encoder, rate-distortion performance will be affected when a rate-control mechanism is introduced. However, in a real scenario application rate control is desirable and sometimes even required. The results show that our proposal achieves the desired target rates with competitive MS-SSIM results.

References

[1] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[2] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal processing magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[3] F. Bellard, "The BPG image format," Online, April 21 2018, available from http://bellard.org/bpg/.

[4] F. Dufaux, G. J. Sullivan, and T. Ebrahimi, "The jpeg xr image coding standard [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 195–204, 2009.

[5] A. Artusi, R. K. Mantiuk, T. Richter, P. Hanhart, P. Korshunov, M. Agostinelli, A. Ten, and T. Ebrahimi, "Overview and evaluation of the jpeg xt hdr image compression standard," *Journal of Real-Time Image Processing*, vol. 16, no. 2, pp. 413–428, 2019.

[6] J. Alakuijala, R. van Asseldonk, S. Boukortt, M. Bruse, Z. Szabadka, I.-M. Coma, M. Firsching, T. Fischbacher, E. Kliuchnikov, S. Gomez *et al.*, "Jpeg xl next-generation image compression architecture and coding tools," in *Applications of Digital Image Processing XLII*, vol. 11137. International Society for Optics and Photonics, 2019, p. 111370K.

[7] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[8] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=rkcQFMZRb

[9] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 771–10 780.

[10] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool, "Generative adversarial networks for extreme learned image compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 221–231.

[11] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, Apr. 2016.

TABLE II

AVERAGE OBJECTIVE METRICS FOR THE ENTIRE TEST SET. BITRATE AVERAGE ($\bar{R}$) AND VARIANCE (σ_R^2) ARE PRESENTED. MS-SSIM AND VMAF REPORTED ARE THE AVERAGE AMONG THE DATASET. NOTE THAT JPEG WAS UNABLE TO ACHIEVE SOME BITRATE TARGETS . *IMAGES JPEGAI9 AND JPEGAI11 NOT INCLUDED. **IMAGE JPEGAI9 NOT INCLUDED.

	HEVC 4:4:4				**HEVC 4:2:2**				**HEVC 4:2:0**			
Target Rate	$\bar{R}$	σ_R^2	**MS-SSIM**	**VMAF**	$\bar{R}$	σ_R^2	**MS-SSIM**	**VMAF**	$\bar{R}$	σ_R^2	**MS-SSIM**	**VMAF**
0.06	0.06	6e−6	0.909	72.02	0.06	4e−6	0.896	72.50	0.06	6e−6	0.909	71.89
0.12	0.12	2e−5	0.937	80.20	0.12	3e−5	0.924	80.70	0.12	2e−5	0.934	80.43
0.25	0.24	1e−4	0.960	88.07	0.25	1e−4	0.951	88.74	0.26	1e−4	0.953	88.74
0.50	0.50	3e−4	0.977	93.55	0.50	4e−4	0.971	94.01	0.51	3e−4	0.969	94.41
0.75	0.76	1e−3	0.985	95.87	0.74	1e−3	0.980	96.26	0.76	8e−4	0.978	96.47
1.00	1.02	1e−3	0.990	97.07	1.00	9e−4	0.986	97.30	0.99	1e−3	0.982	97.49
1.50	1.50	3e−3	0.994	98.14	1.51	2e−3	0.992	98.36	1.47	2e−3	0.988	98.55
2.00	2.00	4e−3	0.997	98.68	2.01	4e−3	0.995	98.86	2.03	3e−3	0.993	99.05
	JPEG 2000				**JPEG**				**UnB_SBBR**			
Target Rate	$\bar{R}$	σ_R^2	**MS-SSIM**	**VMAF**	$\bar{R}$	σ_R^2	**MS-SSIM**	**VMAF**	$\bar{R}$	σ_R^2	**MS-SSIM**	**VMAF**
0.06	0.06	5e−7	0.896	70.77	–	–	–	–	0.06	1e−5	0.856	34.61
0.12	0.12	5e−7	0.926	80.55	–	–	–	–	0.12	3e−5	0.927	58.60
0.25	0.25	1e−6	0.952	89.65	–	–	–	–	0.25	3e−5	0.960	70.44
0.50	0.49	5e−6	0.973	95.34	0.55*	6e−3	0.960	92.72	0.50	6e−4	0.979	82.20
0.75	0.74	1e−5	0.981	97.06	0.75**	1e−4	0.968	94.91	0.73	9e−4	0.987	87.61
1.00	0.99	2e−5	0.986	97.79	1.00	1e−4	0.973	96.41	0.98	2e−3	0.991	89.41
1.50	1.48	5e−5	0.992	98.56	1.51	2e−4	0.982	97.89	1.48	3e−3	0.994	91.19
2.00	1.98	9e−5	0.995	98.87	2.00	1e−3	0.987	98.49	1.94	6e−3	0.996	94.91

[12] D. Minnen, G. Toderici, M. Covell, T. Chinen, N. Johnston, J. Shor, S. J. Hwang, D. Vincent, and S. Singh, "Spatially adaptive image compression using a tiled deep network," in *International Conference on Image Processing (ICIP)*, Feb. 2017. [Online]. Available: http://arxiv.org/abs/1802.02629

[13] H. C. Jung, N. D. Guerin Jr, R. S. Ramos, B. Macchiavello, E. Peixoto, E. M. Hung, T. Campos, R. C. Silva, V. Testoni, and P. G. Freitas, "Multi-mode intra prediction for Learning-based Image Compression," in *International Conference on Image Processing (ICIP)*, Oct. 2020.

[14] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2922–2930.

[15] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=rJiNwv9gg

[16] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium, PCS 2018, San Francisco, CA, USA, June 24-27, 2018*. IEEE, 2018, pp. 253–257. [Online]. Available: https://doi.org/10.1109/PCS.2018.8456308

[17] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.06281

[18] ——, "End-to-end optimized image compression," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=rJxdQ3jeg

[19] P. Akyazi and T. Ebrahimi, "Learning-based image compression using convolutional autoencoder and wavelet decomposition," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, p. 0.

[20] H. Akutsu and T. Naruko, "End-to-end learned roi image compression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[21] C. Aytekin, F. Cricri, A. Hallapuro, J. Lainema, E. Aksu, and M. Hannuksela, "A compression objective and a cycle loss for neural image compression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[22] W. . via SC 29 Secretariat, "Sc29-liaison statement from itu-t sg 16 to sc 29_wg 1 on jpeg artificial intelligence," in *ISO/IEC JTC1/SC29/WG1 M85041. JPEG*. JPEG, 2019, pp. 1–1.

[23] R. W. Franzen, "Kodak image dataset," Online, Updated on January 27 2013, available from http://r0k.us/graphics/kodak/.

[24] Johannes Ballé and Sung Jin Hwang and Nick Johnston and David Minnen, "Tensorflow compression," Online, available from https://tensorflow.github.io/compression/.

[25] "Dataset of the CVPR workshop and challenge on learned image compression (CLIC)," online, http://www.compression.cc.

[26] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017, DIV2K dataset: DIVerse 2K resolution high quality images as used for the challenges at NTIRE (CVPR 2017 and CVPR 2018) and at PIRM (ECCV 2018), available from https://data.vision.ee.ethz.ch/cvl/DIV2K/.

[27] H. Nemoto, P. Hanhart, P. Korshunov, and T. Ebrahimi, "Ultra-Eye: UHD and HD images eye tracking dataset," in *Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, Singapore, September 2014, available from http://mmspg.epfl.ch/ultra-eye. [Online]. Available: http://infoscience.epfl.ch/record/200190

[28] L. Jin, J. Y. Lin, S. Hu, H. Wang, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, "Mcl-jci dataset," online, http://mcl.usc.edu/mcl-jci-dataset/.

[29] ——, "Statistical Study on Perceived JPEG Image Quality via MCL-JCI Dataset Construction and Analysis," in *Electronic Imaging (2016), the Society for Imaging Science and Technology (IS&T)*, 2016.

[30] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.