Low-Complexity Overfitted Neural Image Codec

Thomas Leguay, Théo Ladune, Pierrick Philippe, Gordon Clare, Félix Henry, Olivier Déforges

Low-complexity Overfitted Neural Image Codec

Thomas Leguay, Théo Ladune, Pierrick Philippe, Gordon Clare, Félix Henry
Orange Innovation, France
firstname.lastname@orange.com

Olivier Déforges
IETR, France
olivier.deforges@insa-rennes.fr

***Abstract*—We propose a neural image codec at reduced complexity which overfits the decoder parameters to each input image. While autoencoders perform up to a million multiplications per decoded pixel, the proposed approach only requires 2 300 multiplications per pixel. Albeit low-complexity, the method rivals autoencoder performance and surpasses HEVC performance under various coding conditions. Additional lightweight modules and an improved training process provide a 14% rate reduction with respect to previous overfitted codecs, while offering a similar complexity. This work is made open-source at https://orange-opensource.github.io/Cool-Chic/.**

***Index Terms*—Image coding, Overfitting, Low-complexity**

I. Introduction

In many use cases (TV, video on demand), videos are encoded once using a dedicated device, while they are decoded many times on a variety of low-power devices such as smartphones. Consequently, the complexity and energy consumption are substantially constrained for the decoder. For decades, conventional codecs (H.264/AVC [1], H.265/HEVC [2] and H.266/VVC [3]) have been designed with this constraint in mind. Each successive codec provides enhanced compression performance while still offering a low decoder complexity.

Conventional codecs have many different possibilities for compressing a signal. Each time a signal is to be compressed, the encoder assesses the different options available and only selects the few most suited ones. Encoding is framed as a discrete optimization problem, *i.e.*, a competition between all parameters of all tools, selecting the best ones through a rate-distortion (RD) cost. These tools and the compressed signal are sent to the decoder which simply applies the selected tools on the compressed signal. As such, the decoder complexity remains low at the cost of an expensive encoding process which optimizes an RD cost for each signal. Successive conventional codecs have provided an ever-increasing number of hand-crafted coding tools while maintaining the individual tool complexity sufficiently small. This leads to better signal adaptation and improved compression performance.

Learned codecs based on autoencoders (AE) [4]–[7] offer an alternative coding paradigm. Here, the codec no longer looks for the optimal decoding tools each time a signal has to be compressed. Instead, it is designed during an offline training stage and computes in a single shot a compressed representation optimizing an *average* RD cost. Thus, the instance-wise online RD optimization performed by conventional codecs is replaced by an average offline optimization.

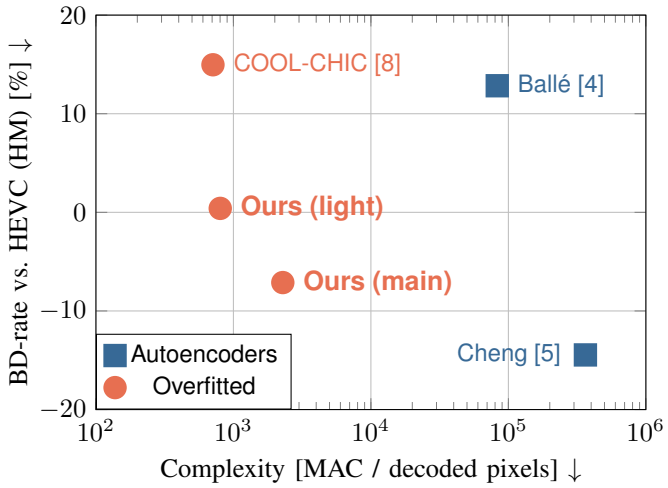


Fig. 1: Rate savings versus HEVC (HM) on the CLIC 2020 professional validation set [9]. Negative results mean that less rate is required to achieve the same quality as HEVC.

The JPEG-AI standardization effort shows that AE-based image codecs are able to outperform conventional codecs [10]. However, AE decoders are often orders of magnitude more complex as they must offer good performance on a wide variety of images. This results in prohibitive decoding complexity *e.g.* 800 kMAC / pixel (kilo multiplication-accumulation) for the JPEG-AI development model [11]. This might hinder the practical usage of AE-based codecs.

Authors in [12], [13] propose to improve AE-based codecs by reintroducing an instance-wise RD optimization. An initial solution from an autoencoder is refined through gradient descent, overfitting the autoencoder on the signal to compress. This suggests the possibility of using overfitting to improve compression efficiency.

COOL-CHIC by Ladune *et al.*, [8] goes one step further, leveraging Implicit Neural Representations [14] to design an overfitted image codec without relying on an autoencoder. In Ladune *et al.*, compressing an image consists in overfitting a lightweight MLP (multilayer perceptron) decoder and a latent representation. The overfitted MLP parameters and latent variables are then conveyed to the receiver and used to reconstruct the image. This yields compression performance on par with Ballé's autoencoder [4] for a decoder complexity a hundred times smaller *i.e.*, of less than 1 kMAC / pixel.

By reintroducing an instance-wise RD optimization,

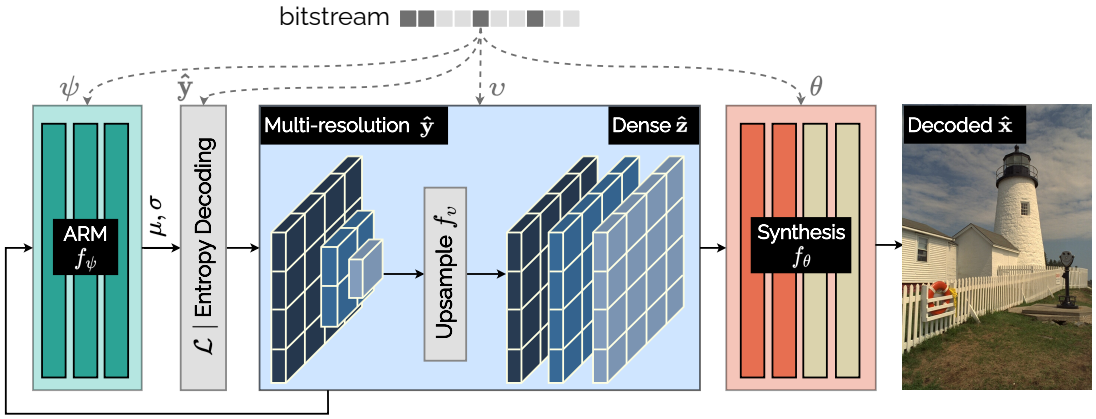


Fig. 2: Decoding process of the proposed system. ARM stands for AutoRegressive Module and $\mathcal{L}$ is a Laplace distribution.

this overfitting-based approach provides compelling coding performance with a low decoder complexity. Overfitting allows automatic learning of the adapted decoder for each image and rate constraint. This avoids the hand-crafted design of the coding modes.

This paper enhances COOL-CHIC while maintaining a low decoding complexity. Our contributions are as follows:

1) A low-complexity adapted upsampling of the latent variables is proposed;
2) Lightweight convolution layers are introduced into the original MLP-based decoder, improving its performance;
3) Improved training. We show that considering the actual quantization instead of its relaxed version during the overfitting results in better performance.

These contributions lead to a rate reduction of 14%, outperforming HEVC for a decoder complexity of 2.3 kMAC / pixel. In order to promote the design of low-complexity codecs, this work is made open-source [15].

II. System overview

This section presents the proposed decoding scheme based on [8] and illustrated in Fig. 2. Let $\mathbf{x} \in \mathbb{N}^{C \times H \times W}$ be an $H \times W$ image to compress, with C color channels. The proposed system is composed of three neural networks (NN): an auto-regressive module (ARM) $f_{\boldsymbol{\psi}}$, an upsampler $f_{\boldsymbol{\upsilon}}$ and a synthesis $f_{\boldsymbol{\theta}}$. These NN are complemented with $\hat{\mathbf{y}}$, a set of L two-dimensional multi-resolution discrete latent variables:

$$\hat{\mathbf{y}} = \left\{ \hat{\mathbf{y}}_l \in \mathbb{Z}^{\frac{H}{2^l} \times \frac{W}{2^l}}, l \in 0, \dots, L-1 \right\}. \tag{1}$$

The compressed representation of $\mathbf{x}$ comprises the NN weights $\{\boldsymbol{\psi}, \boldsymbol{\theta}, \boldsymbol{\upsilon}\}$ and the latent variable $\hat{\mathbf{y}}$. The encoding stage optimizes these parameters while decoding consists simply in applying the obtained parameters.

A. Decoding

The first decoding step retrieves the NN parameters $\{\boldsymbol{\psi}, \boldsymbol{\theta}, \boldsymbol{\upsilon}\}$ from the bitstream following the method proposed in [8]. Then, each latent variable $\hat{\mathbf{y}}_l$ is entropy decoded using a range coder driven by the ARM $f_{\boldsymbol{\psi}}$. The ARM models the distribution of the i-th value from the l-th latent variable $y_{l,i}$ conditionaly to $\mathbf{y}_{l,<i}$, a set of already decoded values from the same latent variable:

$$y_{l,i} \sim \mathcal{L}(\mu_{l,i}, \sigma_{l,i}), \text{ where } \mu_{l,i}, \sigma_{l,i} = f_{\boldsymbol{\psi}}(\mathbf{y}_{l,<i}). \tag{2}$$

Then, the L latents $\hat{\mathbf{y}}_l$ are upsampled to obtain a dense latent representation $\hat{\mathbf{z}}$, with the spatial dimensions of the image:

$$\hat{\mathbf{z}} = f_{\upsilon}(\hat{\mathbf{y}}), \text{ with } \hat{\mathbf{z}} \in \mathbb{R}^{L \times H \times W}. \tag{3}$$

The synthesis transform is finally applied on the dense latent representation to compute the decoded image $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = f_{\boldsymbol{\theta}}(\hat{\mathbf{z}}). \tag{4}$$

B. Encoding

Encoding an image is achieved by overfitting the decoder parameters to determine the optimal ones according to a rate-distortion (RD) objective:

$$\{\hat{\mathbf{y}}, \boldsymbol{\psi}, \boldsymbol{\theta}, \boldsymbol{\upsilon}\} = \arg\min D(\mathbf{x}, \hat{\mathbf{x}}) + \lambda R(\hat{\mathbf{y}}), \tag{5}$$

with D the mean-squared error, R the rate (approximated by the entropy) and λ the Lagrange multiplier balancing rate and distortion. All the model parameters are learned for a single image and are therefore adapted for its content.

Gradient descent is used to optimize the RD objective, requiring continuously valued parameters. As such, the quantization of the latent representation $\hat{\mathbf{y}} = Q(\mathbf{y})$ is modeled first as independent noise addition [16] and then using a modified Straight-Through Estimator [17] (see Section III-C). Once the encoding is finished, the NN parameters and the latent representation are quantized and entropy coded to be sent efficiently to the decoder.

III. Proposed improvements

This section details the contributions of this paper: designing an adapted upsampling module, adding convolutional elements to the synthesis and better considering the quantization during the training stage.

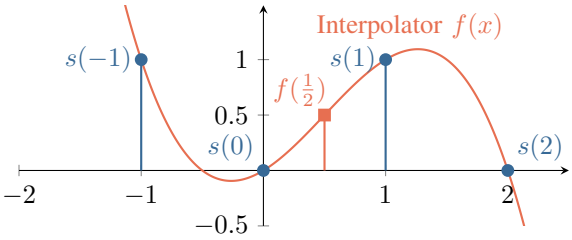


Fig. 3: Cubic interpolation of a discrete signal $s(n)$.

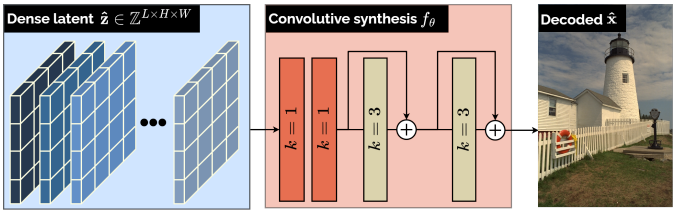


Fig. 4: Convolution-based synthesis function $f_{\boldsymbol{\theta}}$. The kernel size of each layer is denoted by k. More details in Table I.

A. Adapted upsampling

The coding scheme relies on an upsampling step to obtain a dense representation $\hat{\boldsymbol{z}}$ from the set of multi-resolution latent variables $\hat{\mathbf{y}}$. In [8], a chained bicubic upsampling [18] by a factor of 2 is implemented. It is applied multiple times for each dimension to obtain the desired resolution. In this section we recall that bicubic upsampling is equivalent to a convolution operation. Here it serves as a proper initialization to learn a more adapted upsampling.

For the sake of clarity, let us consider a one-dimensional discrete signal $s\colon \mathbb{Z} \to \mathbb{R}$ whose value $s(n)$ is known for all integers $n \in \mathbb{Z}$ (Fig. 3). In the system, the upsampling step estimates the value $s(n + \frac{1}{2})$. Without loss of generality, the case $n = 0$ is considered. Cubic upsampling defines an interpolation function $f\colon [0,1] \to \mathbb{R}$ as a third degree polynomial:

$$f(x) = \sum_{i=0}^{3} a_i x^i = \mathbf{a}^\mathsf{T}\mathbf{x} \text{ with } \mathbf{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_3 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x^0 \\ \vdots \\ x^3 \end{bmatrix}. \quad (6)$$

To obtain the polynomial coefficients $\mathbf{a}$, the interpolator is constrained to be equal to the actual signal for neighboring integers *i.e.*, $f(k) = s(k), \forall k \in \{-1, 0, 1, 2\}$. Writing these conditions using matrix notation yields:

$$\mathbf{s} = \begin{bmatrix} s(-1) \\ s(0) \\ s(1) \\ s(2) \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \mathbf{B}\mathbf{a}. \quad (7)$$

Combining equations (6) and (7) allows to obtain the expression of the interpolation function:

$$f(x) = \left(\mathbf{B}^{-1}\mathbf{s}\right)^\mathsf{T}\mathbf{x} = \mathbf{s}^\mathsf{T}\left(\mathbf{B}^{-1}\right)^\mathsf{T}\mathbf{x}. \quad (8)$$

Since the upsampling always has a factor of 2, we are only concerned with $f(\frac{1}{2})$ and $\mathbf{x} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} \end{bmatrix}$. As such, eq. (8) is written as the application of a kernel $\boldsymbol{v}$ on the signal $\mathbf{s}$:

$$f(\frac{1}{2}) = \boldsymbol{v}^\mathsf{T}\mathbf{s}, \text{ with } \boldsymbol{v} = \frac{1}{16}\begin{bmatrix} -1 \\ 9 \\ 9 \\ -1 \end{bmatrix}. \quad (9)$$

The same reasoning holds for all integers n and for two-dimensional signals at the expense of a two-dimensional (separable) kernel $\boldsymbol{v}$. Hence, computing the upsampled value $f(n + \frac{1}{2})$ consists in convolving the original signal $\mathbf{s}$ with $\boldsymbol{v}$.

Contribution. It is proposed to learn an adapted upsampling $f_{\boldsymbol{v}}$. The encoder sets the desired filter properties (*e.g.* , cut-off frequencies or non-separability) to cope with the directional patterns of the image. This gives the encoder more possibilities, allowing to further optimize the rate-distortion trade-off. The upsampling kernel is quantized and transmitted to the decoder similarly to the ARM and synthesis parameters [8].

B. Convolution-based synthesis

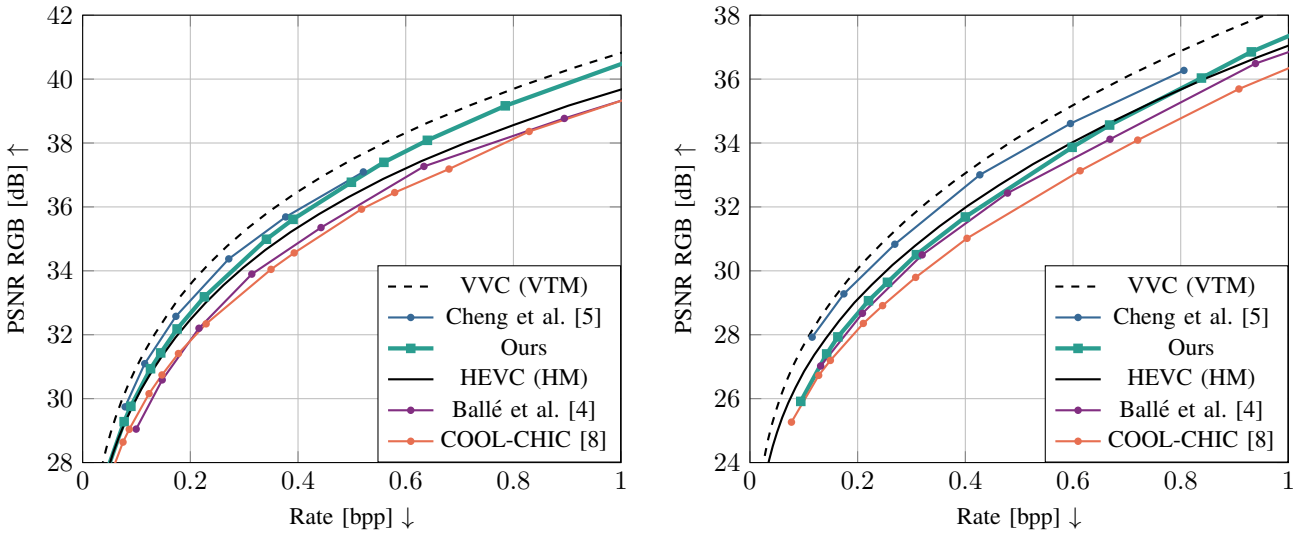
The synthesis $f_{\boldsymbol{\theta}}$ in [8] is implemented as an MLP (*i.e.*, convolution layers with kernel of size 1). As such, it synthesizes independently each pixel of the decoded image.

Contribution. We introduce convolution layers with kernels of size 3 to take advantage of neighboring latent values when synthesizing one pixel of the decoded image. This new architecture is presented in Fig. 4. As the synthesis operates under a strict complexity constraint, these convolution layers are located at the end of the synthesis where they act as residual post-filters on a 3-feature signal. Using kernel of size 3, such convolution layer represents 81 MAC / decoded pixel. These layers are overfitted alongside the whole system to obtain better rate-distortion performance.

C. Quantization during training

Quantization $Q(x) = \lfloor x \rceil$ is a key element of a lossy coding scheme as it reduces the entropy of a signal. However, $\frac{\partial Q}{\partial x}(x)$ is null almost everywhere (see Fig. 6) preventing the usage of gradient-based optimization methods. In the literature, quantization is often modeled as a uniform noise addition during the training stage [16]. In order to reduce the discrepancy between training and inference some authors propose to switch to the actual quantization at the end of the training [19]. In this case, a straight-through estimator (STE) [17] is used, setting $\frac{\partial Q}{\partial x}(x) = 1$ manually in the backward pass. The same mechanism is implemented in [8].

Contribution. We argue that the STE is not the most suited gradient estimator. Since $\frac{\partial Q}{\partial x}(x)$ is null almost everywhere, setting a gradient close to zero is more consistent with the behavior of the quantization function. We propose the ϵ-STE which sets the derivative to a small value *i.e.*, $\frac{\partial Q}{\partial x}(x) = \varepsilon$.



(a) CLIC 2020 professional validation set.

(b) Kodak dataset.

Fig. 5: Rate-distortion results on two datasets.

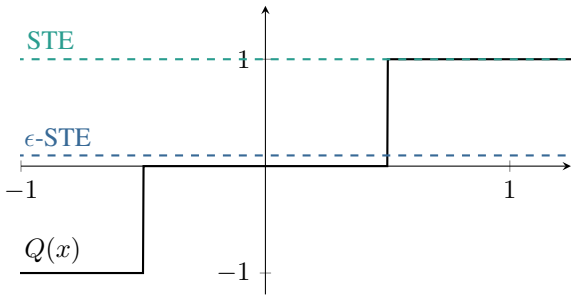


Fig. 6: Different approximations of the quantization derivative.

Config	ARM f_ψ	Upsampling f_υ	Synthesis f_θ
Main	24 / 24 Linear	TConv $k=8$ $s=2$	7 / 40 Conv $k=1$
	24 / 24 Linear		40 / 3 Conv $k=1$
	24 / 2 Linear		3 / 3 Conv $k=3$
			3 / 3 Conv $k=3$
Light	12 / 12 Linear	TConv $k=8$ $s=2$	7 / 18 Conv $k=1$
	12 / 12 Linear		18 / 3 Conv $k=1$
	12 / 2 Linear		3 / 3 Conv $k=3$

TABLE I: Architecture of the proposed systems. I/O indicates the number of input and output features. Each layer is followed by a ReLU, except the last one of each module. Kernel size and stride are denoted by k and s respectively. TConv is a transpose convolution.

Empirically, $\varepsilon = 10^{-2}$ gives the best results when used alongside the Adam optimization algorithm. This allows for a better optimization of the RD cost during the encoding.

IV. Experiments

A. Rate-distortion results

Experimental framework. The proposed system is evaluated in two configurations: light (worst case complexity of 0.8 kMAC / decoded pixel, similar to [8]) and main (worst case complexity of 2.3 kMAC / decoded pixel). The details of both configurations are given in Table I. The ARM uses respectively 12 or 24 spatial neighbours as input. Both configurations have $L = 7$ latents and are optimized using the loss function presented in eq. (5). These models are compared to HEVC and VVC through their reference implementations HM 16.20 and VTM 11.1 following the conditions of [10]. It is also compared to learned overfitted codecs [8] and autoencoder-based systems: Ballé *et al.*, [4], Cheng *et al.*, [5]. Performance of the codecs are expressed using the BD-rate [20] *i.e.*, the relative rate required to achieve the same quality (here the PSNR) than a reference codec (here HEVC).

RGB performance. Figure 1 shows a complexity-performance graph of both configurations on the CLIC 2020 professional validation set [9]. The light system outperforms [8] while maintaining a similarly low complexity, proving the relevance of the proposed improvements. Moreover, the light configuration is competitive with HEVC (BD-rate: 0.4%) and is more performant than Ballé *et al.*, at a significantly lower decoder complexity (0.8 vs. 83 kMAC / decoded pixel). Finally, our main configuration outperforms HEVC (BD-rate: -7.1%) with as few as 2.3 kMAC / decoded pixel.

Figure 5 presents the rate-distortion curves of the main configuration on the CLIC 2020 and Kodak [21] datasets. On CLIC 2020, the proposed model outperforms COOL-CHIC, Ballé *et al.*, and HEVC on a wide range of quality. At higher rates, it even comes close to Cheng *et al.*, while having a decoder-side complexity 100 times smaller. Similar albeit less favorable results are observed on the Kodak dataset. Section V discusses the causes of these less favorable results.

Sequence-wise results. Figure 7 presents the sequence-wise BD-rate of the main system against HEVC. Out of the

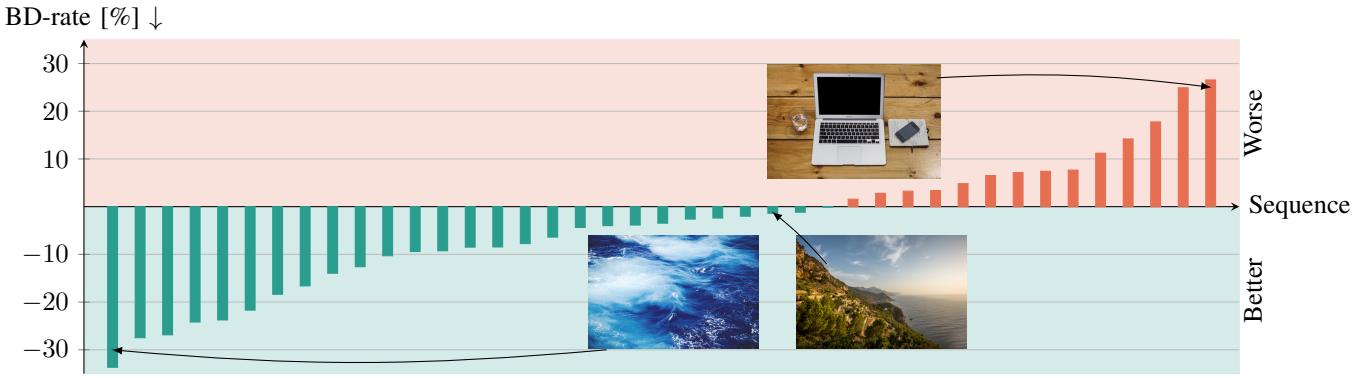


Fig. 7: Sequence-wise BD-rate of the proposed system versus HEVC (HM) on the CLIC 2020 professional validation dataset. PSNR is measured in the RGB domain.

41 images of the CLIC dataset, 26 are better compressed with the main configuration, highlighting its compelling results. The system offers interesting performance on contents exhibiting fewer directional patterns (*e.g.* water) which are notoriously difficult for conventional codecs. This hints that the system can be a interesting complement to conventional codecs.

YUV420 results. As a first step towards video coding, the proposed system evaluated on the first frame of the MPEG Common Test Conditions videos [22]. Due to the YUV420 format, a nearest neighbour downscaling is added after the synthesis. The performance against HEVC is presented in Table II. It is often empirically noticed that PSNR YUV420 is less favorable to learned codecs since conventional ones are designed with YUV420 in mind. Here, this leads to worse compression performance of our system versus HEVC. Yet, it still outperforms HEVC on specific contents such as the class F corresponding to screen content.

Class	B	C	D	E	F	Average
Average BD-rate [%] ↓	19.4	8.0	25.0	34.4	**-5.6**	16.2

TABLE II: Average BD-rate of the proposed main configuration versus HEVC. PSNR computed in the YUV420 domain.

Visual results. Figure 8 presents the same image compressed at two rate targets by HEVC and the proposed system. At low rate, HEVC exhibits blocking and ringing artifacts detrimental to the quality. Since our system is not based on blocks, there is no such artifact. At higher rates, both codecs are able to deliver high visual quality.

B. Ablation study

Table III presents the BD-rate loss when disabling particular modules. Removing all contributions corresponds to [8]. The ablation shows that each contribution increases performance while keeping the overall complexity constant. Note that the best performance gain is due to the ϵ-STE during training *i.e.*, to a better optimization of the decoder. This hints that better performance could be obtained without increasing the decoder complexity by further improving the training stage.

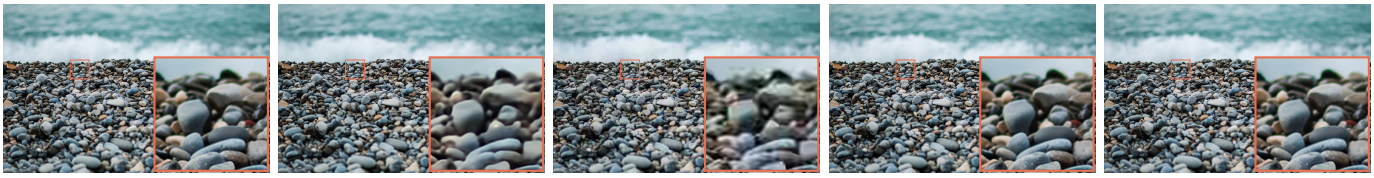
Conv. synthesis	Learned upsample	ϵ-STE	Complexity [kMAC / decoded pix] Synthesis	Upsampling	Total	BD-rate vs. full model [%] ↓
			0.8	0.03	2.5	14.0
✓			0.6	0.03	2.2	9.6
✓	✓		0.6	0.1	2.3	6.0
✓	✓	✓	0.6	0.1	2.3	0.0

TABLE III: Ablation study of the main configuration on CLIC 2020. ARM f_ψ remains identical for all tests and has a complexity of 1.6 kMAC / decoded pixel. When disabled, ϵ-STE is replaced by STE.

V. Limitations and Future Work

NN parameters rate. The proposed coding scheme assumes that the latent variables represent most of the rate while the cost of sending the NN parameters is neglectable. As such, the loss function does not take the network rate into account, see eq. (5). Yet, this assumption does not hold for all types of images and rate targets. Figure 9 presents the share of rate allocated to the NN parameters. At lower rates or for low-resolution images (Kodak), more than 10% of the rate is dedicated to the NN parameters, explaining the relatively worse results obtained for the Kodak dataset. Future work should focus on reducing the rate of NN parameters by considering techniques from the literature *e.g.* pruning, tensor decomposition, distillation [23].

Encoding. The experimental results show that a better encoding *i.e.*, a better optimization (using ϵ-STE) of the RD cost leads to significant compression gains. Yet, this could be further refined by using more advanced optimization techniques [24] and more suited weights initialization. Beside improving the compression efficiency this would also reduce the encoding duration as it currently requires 10 to 60 minutes per image depending on the resolution. Note that a better



(a) Original (b) Ours 0.077 bpp (c) HEVC 0.074 bpp (d) Ours 0.801 bpp (e) HEVC 0.813 bpp

Fig. 8: Comparison of HEVC and our system on CLIC 2020 image *jeremy-cai-1174* at two different rates.

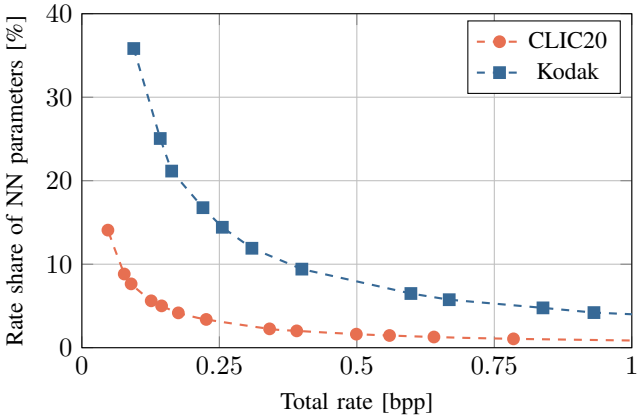


Fig. 9: Rate allocated to the NN parameters at different bitrates

implementation would also drastically reduce the encoding time as demonstrated by Instant-NGP [25].

VI. Conclusion

This paper proposes a learned image coding scheme with a decoding complexity of 2.3 kMAC / pixel *i.e.*, a hundred times smaller than autoencoder-based codecs. This lightweight codec offers up to 7% rate reduction compared to modern conventional codecs such as HEVC. It also significantly outperforms classical autoencoders from Ballé *et al.* This performance is achieved by overfitting and conveying a neural decoder for each image. This paper refines both the components of the decoder (upsampling, synthesis) and its overfitting (ϵ-STE) leading to a rate reduction of 14% compared to COOL-CHIC while maintaining a low decoder-side complexity.

Future work should focus on improving the encoding process and better compression of the neural network parameters in order to improve compression efficiency without increasing the decoder-side complexity.

References

[1] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[3] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC)," *IEEE*, 2021.

[4] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations*, 2018.

[5] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*.

[6] T. Ladune and P. Philippe, "AIVC: Artificial intelligence based video codec," in *2022 IEEE International Conference on Image Processing*.

[7] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang, "Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5708–5717, 2022.

[8] T. Ladune, P. Philippe, F. Henry, G. Clare, and T. Leguay, "COOL-CHIC: Coordinate-based low complexity hierarchical image codec," 2023. [Online]. Available: https://arxiv.org/abs/2212.05458

[9] CLIC20, "Challenge on learned image coding 2020," *http://clic.compression.cc/2021/tasks/index.html*, 2020.

[10] "ISO/IEC JTC 1/SC29/WG1 N100250, REQ "report on the JPEG AI call for proposals results"," 2022.

[11] "ISO/IEC JTC 1/SC29/WG1 M99069, CPM "on the computational complexity of the operators."," 2023.

[12] Y. Yang and S. Mandt, "Asymmetrically-powered neural image compression with shallow decoders," *CoRR*, vol. abs/2304.06244, 2023.

[13] J. Campos, S. Meierhans, A. Djelouah, and C. Schroers, "Content adaptive optimization for neural image compression," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*.

[14] E. Dupont, A. Golinski, M. Alizadeh, Y. W. Teh, and A. Doucet, "COIN: compression with implicit neural representations," 2021. [Online]. Available: https://arxiv.org/abs/2103.03123

[15] "COOL-CHIC repository," *https://orange-opensource.github.io/Cool-Chic/*.

[16] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *5th International Conference on Learning Representations, ICLR 2017*.

[17] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," 2017. [Online]. Available: https://arxiv.org/abs/1703.00395

[18] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[19] Z. Guo, R. Feng, Z. Zhang, X. Jin, and Z. Chen, "Learning cross-scale weighted prediction for efficient neural video compression," 2023.

[20] G. Bjøntegaard, "Calculation of average psnr differences between rd-curves," 2001.

[21] "Kodak image dataset," *http://r0k.us/graphics/kodak/*.

[22] "Common test conditions and software reference configurations," in *JCTVC-T2010*.

[23] J. O. Neill, "An overview of neural network compression," 2020. [Online]. Available: https://arxiv.org/abs/2006.03669

[24] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. W. Mahoney, "ADAHESSIAN: an adaptive second order optimizer for machine learning," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, 2021.

[25] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, Jul. 2022.