GaussianImage: 1000 FPS Image Representation and Compression by 2D Gaussian Splatting – Supplementary Material –

Xinjie Zhang[1,3⋆†], Xingtong Ge[2,3⋆†], Tongda Xu[4], Dailan He[5], Yan Wang[4], Hongwei Qin[3], Guo Lu[6], Jing Geng[2†], and Jun Zhang[1†]

[1] The Hong Kong University of Science and Technology
[2] Beijing Institute of Technology [3] SenseTime Research
[4] Institute for AI Industry Research (AIR), Tsinghua University
[5] The Chinese University of Hong Kong [6] Shanghai Jiaotong University
xzhangga@connect.ust.hk, xingtong.ge@gmail.com, x.tongda@nyu.edu
hedailan@link.cuhk.edu.hk, wangyan202199@163.com, qinhongwei@sensetime.com
luguo2014@sjtu.edu.cn, janegeng@bit.edu.cn, eejzhang@ust.hk

1 Details of Gradient Computation

In this section, we delineate the process of computing the gradients of a scalar loss function with respect to the input Gaussian parameters. Beginning with the gradient of the scalar loss $\mathcal{L}$ with respect to each pixel of the output image, we employ the standard chain rule to propagate the gradients backward toward the original input parameters.

1.1 Gradients of Accumulated Rasterization

The initial step involves back-propagating the loss gradients from a given pixel i to the Gaussian that contributed to the pixel. For a Gaussian n impacting pixel i, we aim to calculate the gradients with respect to its color $\boldsymbol{c}' \in \mathbb{R}^3$, the 2D means $\boldsymbol{\mu} \in \mathbb{R}^2$ and 2D covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times 2}$.

For the color, we have

$$\frac{\partial C_i^k}{\partial c_n'^k} = \exp(-\sigma_n). \tag{1}$$

where k indicates the color channel.

For the σ_n, we have

$$\frac{\partial C_i^k}{\partial \sigma_n} = -\exp(-\sigma_n). \tag{2}$$

For the 2D mean, we have

$$\frac{\partial \sigma_n}{\partial \boldsymbol{\mu}_n} = \frac{\partial \sigma_n}{\partial \boldsymbol{d}_n} = \boldsymbol{\Sigma}_n^{-1}\boldsymbol{d}_n \in \mathbb{R}^2. \tag{3}$$

where $\boldsymbol{d} \in \mathbb{R}^2$ is the displacement between the pixel center and the 2D Gaussian center.

For the 2D covariance, we have

$$\frac{\partial \sigma_n}{\partial \boldsymbol{\Sigma}_n} = -\frac{1}{2}\boldsymbol{\Sigma}_n^{-1}\boldsymbol{d}_n\boldsymbol{d}_n^\top\boldsymbol{\Sigma}_n^{-1} \in \mathbb{R}^{2\times 2}. \tag{4}$$

For detailed derivation of the gradient with respect to the 2D covariance, please refer to [17].

1.2 Gradients of 2D Gaussian Formation

Firstly, for the Cholesky decomposition, we have

$$\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top, \boldsymbol{L} = \begin{bmatrix} l_1 & 0 \\ l_2 & l_3 \end{bmatrix}. \tag{5}$$

Let $G = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = \begin{bmatrix} g_1 & g_2 \\ g_2 & g_3 \end{bmatrix}$, we then derive the gradiants of l_1, l_2 and l_3.

For l_1, we have

$$\frac{\partial \mathcal{L}}{\partial l_1} = \langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}, \frac{\partial \boldsymbol{\Sigma}}{\partial l_1} \rangle = \begin{bmatrix} g_1 & g_2 \\ g_2 & g_3 \end{bmatrix} \begin{bmatrix} 2l_1 & l_2 \\ l_2 & 0 \end{bmatrix} = 2g_1 l_1 + 2g_2 l_2. \tag{6}$$

For l_2, we have

$$\frac{\partial \mathcal{L}}{\partial l_2} = \langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}, \frac{\partial \boldsymbol{\Sigma}}{\partial l_2} \rangle = \begin{bmatrix} g_1 & g_2 \\ g_2 & g_3 \end{bmatrix} \begin{bmatrix} 0 & l_1 \\ l_1 & 2l_2 \end{bmatrix} = 2g_2 l_1 + g_2 l_2. \tag{7}$$

For l_3, we have

$$\frac{\partial \mathcal{L}}{\partial l_3} = \langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}, \frac{\partial \boldsymbol{\Sigma}}{\partial l_3} \rangle = \begin{bmatrix} g_1 & g_2 \\ g_2 & g_3 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 2l_3 \end{bmatrix} = 2g_3 l_3. \tag{8}$$

Secondly, for the rotation-scaling (RS) decomposition, we have

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top, \boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \boldsymbol{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}. \tag{9}$$

For θ, we have

$$\frac{\partial \mathcal{L}}{\partial \theta} = \langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}, \frac{\partial \boldsymbol{\Sigma}}{\partial \theta} \rangle = \begin{bmatrix} g_1 & g_2 \\ g_2 & g_3 \end{bmatrix} (\frac{\partial \boldsymbol{R}}{\partial \theta}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top + \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\frac{\partial \boldsymbol{R}^\top}{\partial \theta}). \tag{10}$$

where

$$\frac{\partial \boldsymbol{R}}{\partial \theta} = \begin{bmatrix} -\sin\theta & -\cos\theta \\ \cos\theta & -\sin\theta \end{bmatrix}, \frac{\partial \boldsymbol{R}^\top}{\partial \theta} = \begin{bmatrix} -\sin\theta & \cos\theta \\ -\cos\theta & -\sin\theta \end{bmatrix}. \tag{11}$$

For s_1, we have

$$\frac{\partial \mathcal{L}}{\partial s_1} = \frac{\partial \mathcal{L}}{\partial s_1^2}\frac{\partial s_1^2}{\partial s_1} = \frac{\partial \mathcal{L}}{\partial s_1^2} 2s_1 = \boldsymbol{R}\begin{bmatrix} 2s_1 & 0 \\ 0 & 0 \end{bmatrix}\boldsymbol{R}^\top. \tag{12}$$

For s_2, we have

$$\frac{\partial \mathcal{L}}{\partial s_2} = \frac{\partial \mathcal{L}}{\partial s_2^2}\frac{\partial s_2^2}{\partial s_2} = \frac{\partial \mathcal{L}}{\partial s_1^2} 2s_2 = \boldsymbol{R}\begin{bmatrix} 0 & 0 \\ 0 & 2s_2 \end{bmatrix}\boldsymbol{R}^\top. \tag{13}$$

2 Details of Partial Bits-Back Coding

In Fig. 4 of Section 4.2, we show the results of our codec ("Ours"), alone with two variants using bits-back coding ("Ours+BB", "Ours-Bound"). The "Ours" is the original GaussianImage codec without any bits-back coding. It is the practical codec that achieves 2000 FPS. The "Ours+BB" is the partial bits-back coding codec described in Section 3.3. It reduces a bitrate of

$$\log(N-K)! - \log(N-K) \tag{14}$$

from the original GaussianImage codec. And K is selected as the lowerbound of previous K 2D Gaussian whose cumulative bitrate is at least $\log(N-K)!$:

$$K^* = \inf K, \text{s.t.} \sum_{k=1}^{K} R_k \geq \log(N-K)!. \tag{15}$$

This rate reduction is introduced in [11], and can be implemented using a first in last out entropy coder named Asymmetric Numeral Systems (ANS) [6]. The encoding procedure has a sub-procedure of decoding, and the decoding procedure has a sub-procedure of encoding [16]. The general process of partial bits-back coding is described in Algorithm 1 and 2, where $\mathcal{U}_d$ is uniform distribution with d elements.

When applied to a whole dataset, the partial bits-back coding becomes unnecessary. More specifically, we no longer require encoding previous K Gaussian for initial bits. Instead, we can just use previous image as initial bits. In that case, we can just follow the vanilla bits-back coding in [16]. For a dataset with infinite images, the average rate reduction is

$$\log N! - \log N. \tag{16}$$

However, this rate reduction is never achieved as dataset is never infinite. While it is indeed the greatest lowerbound as any rate greater is achievable. Or to say, the greatest lowerbound of rate for bits-back coding is not achievable. Therefore, we name it "Ours-Bound".

Algorithm 1 Partial Bits-Back Coding Encode

input the 2D Gaussian parameters $G[1:N]$.
procedure Partial-Bits-Back-Encode($G[1:N]$)
$m \leftarrow \emptyset$
for $K = 1$ **to** N **do**
  $m \leftarrow$ ans-encode$(m, G[K])$ {Rate $+R_K$ }
  if len$(m) \geq \log(N-K)!$ **then**
    break
  end if
end for
$m, G[K+1:N] \leftarrow$ ans-decode$(m, \mathcal{U}_{(N-K)!}, G[K+1:N])$ {Rate $-\log(N-K)!$}
$m \leftarrow$ ans-encode$(m, G[K+1:N])$ {Rate $+\sum_{i=K+1}^{N} R_i$}
$m \leftarrow$ ans-encode$(m, N-K)$ {Rate $+\log(N-K)$}
return m

Algorithm 2 Partial Bits-Back Coding Decode

input the bitstream m.
procedure Partial-Bits-Back-Decode(m)
$m, N-K \leftarrow$ ans-decode(m)
$m, G[K+1:N] \leftarrow$ ans-decode(m)
$m, G[K+1:N] \leftarrow$ ans-encode$(m, \mathcal{U}_{(N-K)!}, G[K+1:N])$
$m, G[1:K] \leftarrow$ ans-decode(m)
return G[1:K]

3 Experiments

3.1 Implementation Details

GaussianImage. During the formation of 2D Gaussian, we apply the tanh function to limit the range of position parameters to (-1,1). For covariance parameters, we add 0.5 to the diagonal elements l_1, l_3 of the lower triangular matrix $\boldsymbol{L}$ in the Cholesky factorization or the scaling elements s_1, s_2 in the rotational-scaling factorization. This adjustment prevents the scaling of the covariance from becoming excessively small. In addition, the covariance parameters and weighted color coefficients are initialized using a uniform distribution. The position parameters are initialized as follows:

$$\boldsymbol{\mu} = \text{atanh}(\text{rand}(2) * 2 - 1). \tag{17}$$

where rand(n) generates n random numbers from a uniform distribution.
Baselines. For SIREN [15] and WIRE [14], we implement them by using the open-source project[1] from WIRE. For I-NGP [13] and NeuRBF [5], we adopt the project[2] from NeuRBF. As for compression baselines, the implementation

[1] https://github.com/vishwa91/wire
[2] https://github.com/oppo-us-research/NeuRBF

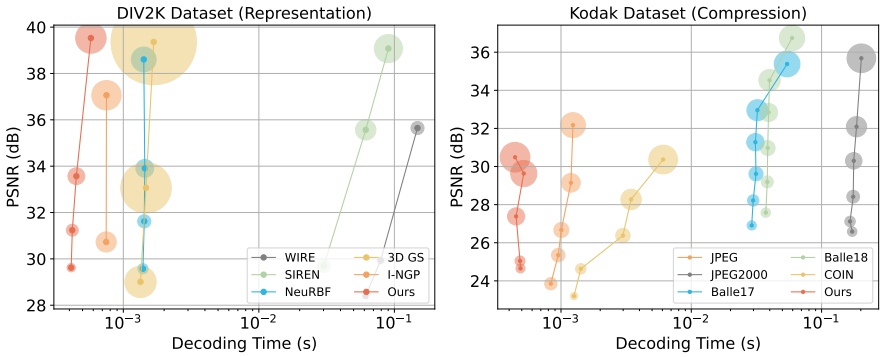

Fig. 1: Image representation (left) and compression (right) results with different decoding time on the DIV2K and Kodak dataset, respectively. The radius of each point indicates the parameter size (left) or bits per pixel (right). Our method enjoys the fastest decoding speed regardless of parameter size or bpp.

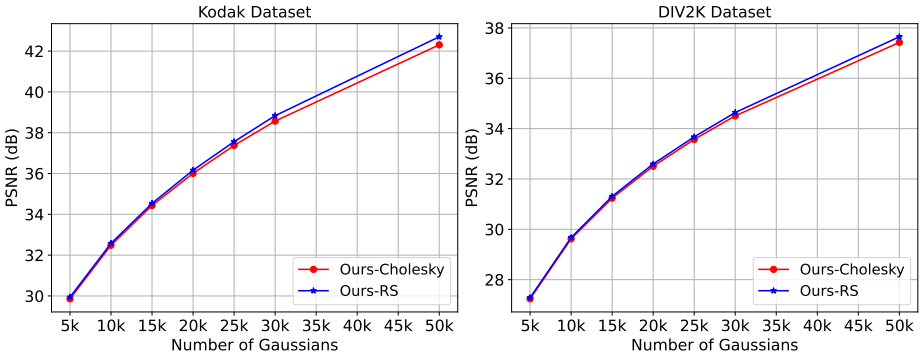

Fig. 2: Image representation with different number of Gaussians.

of COIN [7] uses their library[3]. We evaluate the VAE-based codecs (Ballé17 [1], Ballé18 [2]) using the MSE-optimized models provided by CompressAI [3]. It is worth noting that during the inference of the INR methods, we sample all image coordinates at once to output the corresponding RGB values. This is the maximum inference speed that the INR methods can achieve when the GPU memory resources are sufficient.

3.2 Image Representation and Compression

As shown in Fig. 1, we provide performance comparisons of image representation and compression on DIV2K and Kodak datasets, respectively.

[3] https://github.com/EmilienDupont/coin

Table 1: Ablation study of additive operation on Kodak and DIV2K datasets with 30000 Gaussian points over 50000 training steps.

Variants	Kodak		DIV2K	
	PSNR	MS-SSIM	PSNR	MS-SSIM
Ours-Cholesky + w/ add 0.5	38.57	0.9961	34.51	0.9924
Ours-Cholesky + w/o add 0.5	35.05	0.9906	31.63	0.9830
Ours-RS + w/ add 0.5	38.83	0.9964	34.64	0.9927
Ours-RS + w/o add 0.5	36.34	0.9930	32.51	0.9859

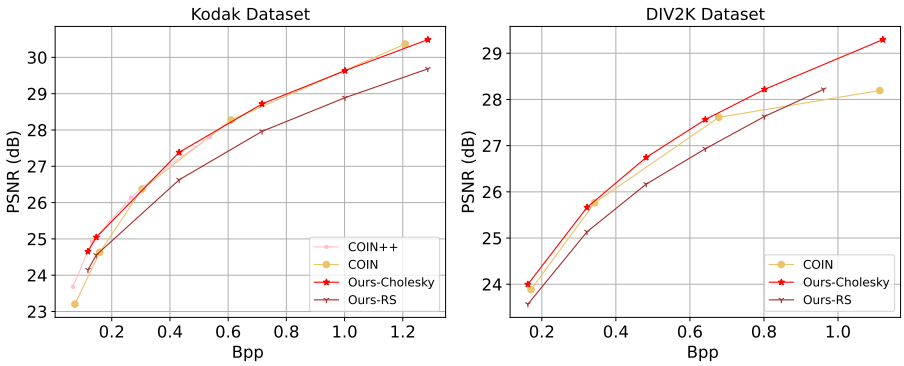

Fig. 3: Image compression results with different factorized forms on the Kodak and DIV2K dataset, respectively.

3.3 Ablation Study

Number of Gaussians. As shown in Fig. 2, our proposed methods improve the quality of the fitted image as the number of Gaussians increases.

Effect of additive operation. The additive operation can be seen as convolving the covariance matrix with a continuous Gaussian, which helps anti-aliasing, thereby effectively improving the fitting performance, as illustrated in Table 1.

Robustness in different factorized forms. Fig. 3 highlights the application of identical quantization approaches to various factorized forms. Notably, the RS codec variant underperforms the Cholesky codec variant, suggesting that rotation and scaling parameters are particularly susceptible to compression distortions, which requires carefully tailored specialized quantization strategies to achieve efficient compression.

4 Discussion

In this paper, we simply apply existing compression techniques to build our image codec. As depicted in Fig. 4 in the main paper, there remains a considerable

discrepancy between our codec and existing traditional/VAE-based codecs in the compression performance. This gap indicates an imperative need for the development of specialized compression algorithms tailored for Gaussian-based codecs to elevate the performance. Moreover, as shown in Table 2 in the main paper, although our encoding speed has been improved by three orders of magnitude compared with COIN [7], there is still a gap of four orders of magnitude compared with VAE-based codecs [1,2]. Therefore, exploring avenues for more rapid image fitting and Gaussian compression emerges as a critical research direction.

Considering that our GaussianImage provides an explicit representation and coding of images, we will further investigate this line from various aspects in the future. First, recent literature successfully performs segmentation-based text-guided editing on 3D scene represented by Gaussians [4, 8], since this discrete representation naturally provides a semantics layout. Intuitively, a similar property also exists in 2D Gaussian images, and it has the potential to develop few-shot text-guided editing on them. Second, image coding for machine [10] is a popular topic in the learned image coding community. This explicit representation is also likely to benefit downstream vision tasks like classification and detection. Finally, high-fidelity image representation [9, 12] is also an essential task to delve into.

References

1. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. In: International Conference on Learning Representations (2017) 5, 7
2. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations (2018) 5, 7
3. Bégaint, J., Racapé, F., Feltman, S., Pushparaja, A.: Compressai: a pytorch library and evaluation platform for end-to-end compression research. arXiv preprint arXiv:2011.03029 (2020) 5
4. Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H., Lin, G.: Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2024) 7
5. Chen, Z., Li, Z., Song, L., Chen, L., Yu, J., Yuan, J., Xu, Y.: Neurbf: A neural fields representation with adaptive radial basis functions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4182–4194 (2023) 4
6. Duda, J.: Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. arXiv: Information Theory (2013), https://api.semanticscholar.org/CorpusID:13409455 3
7. Dupont, E., Golinski, A., Alizadeh, M., Teh, Y.W., Doucet, A.: Coin: Compression with implicit neural representations. In: Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021 (2021) 5, 7
8. Fang, J., Wang, J., Zhang, X., Xie, L., Tian, Q.: Gaussianeditor: Editing 3d gaussians delicately with text instructions. arXiv preprint arXiv:2311.16037 (2023) 7

9. He, D., Yang, Z., Yu, H., Xu, T., Luo, J., Chen, Y., Gao, C., Shi, X., Qin, H., Wang, Y.: Po-elic: Perception-oriented efficient learned image coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1764–1769 (2022) 7
10. Hu, Y., Yang, S., Yang, W., Duan, L.Y., Liu, J.: Towards coding for human and machine vision: A scalable image coding approach. In: 2020 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2020) 7
11. Kunze, J., Severo, D., Zani, G., van de Meent, J.W., Townsend, J.: Entropy coding of unordered data structures. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=afQuNt3Ruh 3
12. Mentzer, F., Toderici, G.D., Tschannen, M., Agustsson, E.: High-fidelity generative image compression. Advances in Neural Information Processing Systems **33**, 11913–11924 (2020) 7
13. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022) 4
14. Saragadam, V., LeJeune, D., Tan, J., Balakrishnan, G., Veeraraghavan, A., Baraniuk, R.G.: Wire: Wavelet implicit neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18507–18516 (2023) 4
15. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in neural information processing systems **33**, 7462–7473 (2020) 4
16. Townsend, J., Bird, T., Barber, D.: Practical lossless compression with latent variables using bits back coding. arXiv preprint arXiv:1901.04866 (2019) 3
17. Ye, V., Kanazawa, A.: Mathematical supplement for the gsplat library. arXiv preprint arXiv:2312.02121 (2023) 2