PIC: Revisiting INR for Image Coding with Fast Encoding and Sub-Millisecond Decoding

Xiang Liu[1,3⋆], Jinxiang Wang[1⋆], Bin Chen[2⋆⋆], Zimo Liu[3], Mingyao Hong[3], Jiawei Li[4], Yaowei Wang[2,3], and Shu-tao Xia[1,3]

[1] Tsinghua University, China
[2] Harbin Institute of Technology, Shenzhen, China
[3] Peng Cheng Laboratory, China
[4] Joy Future Academy, JD Group, China

Abstract. Implicit neural representation (INR) has achieved remarkable progress in novel view synthesis and image/video coding in recent years. Compared to conventional end-to-end image codecs, INR-based compressors demonstrate significant advantages in decoding complexity. However, their practical application has been hindered by the inferior encoding speed and underutilized decoding efficiency. In this work, we propose a feedforward INR image coding architecture, **P**ractical **I**NR Image **C**odec (PIC), that computes all the necessary information for INR network in a single forward pass, achieving an encoding speed of 20 FPS. Additionally, we implement a highly optimized decoder that reaches 2000 FPS decoding speed, significantly surpassing JPEG's performance at comparable rate-distortion (RD) performance. To the best of our knowledge, this work presents the first learning-based image codec that simultaneously outperforms or is comparable with JPEG in both RD performance and decoding speed while maintaining practical encoding speed. Code is available at `https://github.com/actcwlf/PIC`.

Keywords: Image Compression · Implicit Neural Representation · Feed Forward Network

1 Introduction

Image compression has long been a fundamental topic in signal processing and remains a critical technology underpinning more complex systems such as video coding. With the rapid expansion of Internet data, industrial data, AIGC data, and other forms of data, compression technology remains a critically important research area. Image compression technology has undergone significant evolution, encompassing traditional encoders such as JPEG [47], end-to-end models [1, 2], and recently emerged compression methods utilizing implicit neural representation (INR) [11, 12, 24] or Gaussian Splatting (GS) [20, 26, 53]. Each

⋆ Equal contribution.
⋆⋆ Corresponding author.

of these approaches demonstrates distinct strengths and limitations in different aspects of image compression tasks. Traditional image encoders typically leverage human understanding of signals and visual perception, employing predefined rules to discard visually insignificant information for compression. For instance, JPEG exploits the human eye's reduced sensitivity to certain high-frequency components by decomposing the image into different frequencies using Discrete Cosine Transform (DCT) and selectively retaining the most perceptually critical parts, thereby achieving highly efficient compression. Additionally, its codec design strikes a balance between rate-distortion (RD) performance and encoding/decoding speed. Subsequent traditional encoders further improved RD performance by adopting more sophisticated transformation techniques, such as wavelet transforms [41] and predictive coding [4], although at the cost of increased computational complexity and slower processing speeds.

Classic end-to-end image compression algorithms [1,2] conceptually follow the transform coding paradigm used in traditional image codecs. The key difference is that traditional compressors incorporate numerous manually designed transforming rules, whereas the end-to-end approach employs data-driven method to learn the transformation. Taking advantage of the powerful learning capability of neural networks, this data-driven nonlinear transformation can effectively model the image distribution prior to the data, thereby achieving RD performance that surpasses traditional image codecs [15, 19]. Furthermore, significant progress has been made based on generative models [33], particularly in low bit-rate compression scenarios oriented to perceptual metrics [50].

More recently, representation-based methods—such as INRs [11] and 3D Gaussian Splatting [20]—have emerged as promising alternatives. Their key advantage lies in extremely low decoding complexity [24, 30], often achieving orders-of-magnitude speedups over neural codecs [53]. Some variants also rival traditional codecs in rate-distortion performance [21]. However, these gains come at the cost of extremely slow encoding, often requiring minutes or hours to train a model per image, severely limiting real-world deployment.

In this paper, we propose a new image compression paradigm, Practical INR Image Codec (PIC), which bridges the gap between these paradigms. PIC directly produces a low-bit-rate neural representation in a single forward pass through an end-to-end trained network, combining fast encoding, ultra-fast decoding, and competitive rate-distortion performance. Tab. 1 demonstrates the main differences among three paradigms.

The primary contributions of this work are summarized below:

- We propose a novel image coding paradigm PIC that directly generates low-bit-rate neural representation models through neural networks, simultaneously achieving fast encoding, ultra-fast decoding, and comparative RD performance.
- We have implemented an optimized decoder that fully translates the low-complexity characteristics of neural representation models into practical high decoding speeds.

Table 1: Comparision of different paradigm. E2E and RM are abbreviation for end-to-end and representation model respectively. Feedforward means the method is able to encode an image in one forward pass, instead of a full training process. RD represents rate distortion performance. We selected three representative methods, Factorized [1], Hyperprior [2], COIN [11], Cool-Chic v4.2 (fast) [21,24,25] and GaussianImage [53], for comparison. BD-Rate are calcualted relative to JPEG on Kodak dataset. Other results are representative value from the same experiment. Due to the lack of a fair FLOPs calculation method, the complexity of GaussianImage is left blank. Detailed numerical results of all metrics are shown in Fig. 3 and Fig. 4.

Model	Paradigm	Feedforward	BD-Rate↓	Enc.[ms]↓	Dec.[ms]↓	FLOPs/pixels↓
Factorized	E2E	√	-47.34%	**24.3**	35.7	42.175K
Hyperprior	E2E	√	<u>-57.58%</u>	141	169	45.546K
COIN	RM		25.53%	1.40×10^6	3.18	29.057K
Cool-Chic v4.2	RM		**-64.86%**	1.77×10^5	216	1.303K
GaussianImage	RM		36.55%	6.99×10^5	<u>0.439</u>	-
PIC (Our)	E2E RM	√	-12.78%	<u>28.6</u>	**0.418**	**1.074K**

– Through comprehensive experiments, we validate the performance of our proposed method, demonstrating significant improvements in both RD performance and encoding/decoding speed compared to prior representation-based image coding approaches. Notably, our method surpasses nvJPEG in terms of decoding speed.

2 Related Work

2.1 End-to-end Image Compression

Classic end-to-end image compression extends transform coding paradigm, which use neural network as both analysis transform and synthesis transform [1]. An important feature that distinguishes compression models from other models is the integer symbol constraints in entropy coding, which introduces non-differentiable quantization in training [13]. Ballé *et al.* [1] pioneered a method to jointly optimize reconstruction loss and bit-rate constraints by maximizing the Evidence Lower Bound (ELBO) and formalizes the compression model optimization task as

$$\mathcal{L}_{\phi_g,\theta_g} = \lambda R + D(g_s(Q(g_a(\boldsymbol{x};\phi_g));\theta_g),\boldsymbol{x}), \tag{1}$$

where $g_a(\cdot;\phi_g)$ and $g_s(\cdot;\theta_g)$ are analysis transformation and synthesis transformation respectively. $Q(\cdot)$ is quantization operation. To achieve better task performance, several previous works have also explored many differentiable approximation of quantization [1, 13]. R is estimated bit rate. λ balances the reconstruction quality and bit-rate.

This fundamental architecture has been extensively developed in subsequent research. One important approach is to explore more efficient network architec-

tures. Ballé *et al*. [2] introduced scale hyperpriors to better model latent distribution. More work investigate auto-regressive structure [16, 35, 36] or Transformer model [28,54]. Generative models represent another important category. Mentzer *et al*. [33] leveraged GAN architectures for high-fidelity reconstruction. With the rise of diffusion models, more research has begun exploring compression in extremely low bit-rate scenarios [50].

Although end-to-end methods are limited in practical applications due to computational constraints, their single forward encoding capability offers distinct advantages over INR training-based encoding, and provides valuable inspiration for exploring similar INR encoding schemes.

2.2 Representation-based Image Compression

These representation-based methods can be further categorized into several types, with the most prominent paradigm being the direct mapping of positional coordinates to target spaces. For instance, NeRF [34] maps ray angles to density and color along the ray, which has significantly impacted the fields of volume rendering and novel view synthesis, bringing widespread attention to INR technology [3]. Subsequently, this paradigm has expanded to other signal representation domains, including neural rendering [43], image representation, video representation, and further into image and video compression [7, 11].

Another approach involves using learnable parameters or grid as inputs instead of directly utilizing coordinates. Within the NeRF series, Instant-NGP [37] is a representative work that employs a multi-resolution spatial hash grid to store these learnable parameters. This same concept can be extended to other neural representation tasks. There are many workss have made remarkable progress in representing and compressing textures [44], BRDF [10], etc. Similarly, the COOL-CHIC [24] and its successors [21] have demonstrated impressive performance in image compression. This methodology also finds broad applications in video compression [23].

The 3D Gaussian Splatting (3DGS) [20] provides a different perspective of representation models. By directly organizing learnable parameters through specific structures, without relying on per-instance overfitted neural networks, 3DGS has demonstrated unique advantages in novel view synthesis tasks. Compared with INR, this representation method often exhibit more interpretable structure, providing unique advantages beyond quantity performance. Similarly, such methods have been successfully applied to tasks including 3D scene representation [18, 31], image compression [53], and video compression [29].

This class of methods has demonstrated unique advantages in compression, such as low decoding complexity, fast decoding speed, and high reconstruction quality. However, these strengths are difficult to achieve simultaneously in a single model. Moreover, the reliance on model training for encoding hinders their practical deployment. In this work, our proposed method explores how to balance multiple metrics to construct a practical encoder.

2.3 Hypernetwork for INR

Previous work has explored the concept of hypernetworks [14,22], which dynamically modulate neural network weights during inference. This idea can be traced back to even earlier mechanisms such as Fast Weight Programmers (FWP) [39], and several studies have also investigated its theoretical connections to linear Transformers [38]. Existing research on hypernetworks has predominantly focused on tasks such as continual learning [46], few-shot learning [40] and domain adaptation [45]. In the field of INR related research, the implementation of generalizable INR generation itself can be formulated as a hypernetwork model. This differs to some extent from the task of generalizable 3DGS generation [9]: 3DGS is, to a certain degree, an explicit representation model, whose distribution characteristics are closely coupled with the final target information. In contrast, the network parameter space of an INR lacks such a correlation with the signal it encodes, which poses a critical challenge for the design of generalizable INR frameworks. In terms of specific algorithm design, hypernetwork-based INR approaches have been explored in various tasks including content generation [42,51], yet their investigation in compression tasks remains insufficient [6].

This paper addresses this gap by further exploring a generalizable/feedforward INR image encoder based on the hypernetwork mechanism. In particular, targeting the lack of research that simultaneously considers RD performance and coding efficiency, we attempt to design a codec that comprehensively considers performance across all dimensions through this approach, offering new insights into the field of learned image codecs.

3 Method

The primary reason for the slow encoding of representation models lies in the fact that the encoding process itself is the training process. Consequently, methods based on implicit neural representations—as well as similar approaches like Gaussian splatting—can only be applied in scenarios where encoding time is highly insensitive. A straightforward idea is to directly generate this representation model itself through a neural network. The proposed PIC follows the idea. Fig. 1 demonstrates the overall architecture of our method. Sec. 3.1 shows the details of transforming a representation to compression models. Sec. 3.2 introduces entropy estimation module. Sec. 3.3 describes the full pipeline and implementation details.

3.1 Repurposing Representation Models for Compression

To better introduce the proposed method, we begin with the image representation model g_r in Fig. 1. For a $H \times W$ image, $\hat{\boldsymbol{y}}$ is a set of pyramid-like multi-resolution latents

$$\hat{\boldsymbol{y}} = \{\hat{y}_i \in \mathbb{R}^{H_i \times W_i}, i = 1, 2, \ldots, L\}, \tag{2}$$

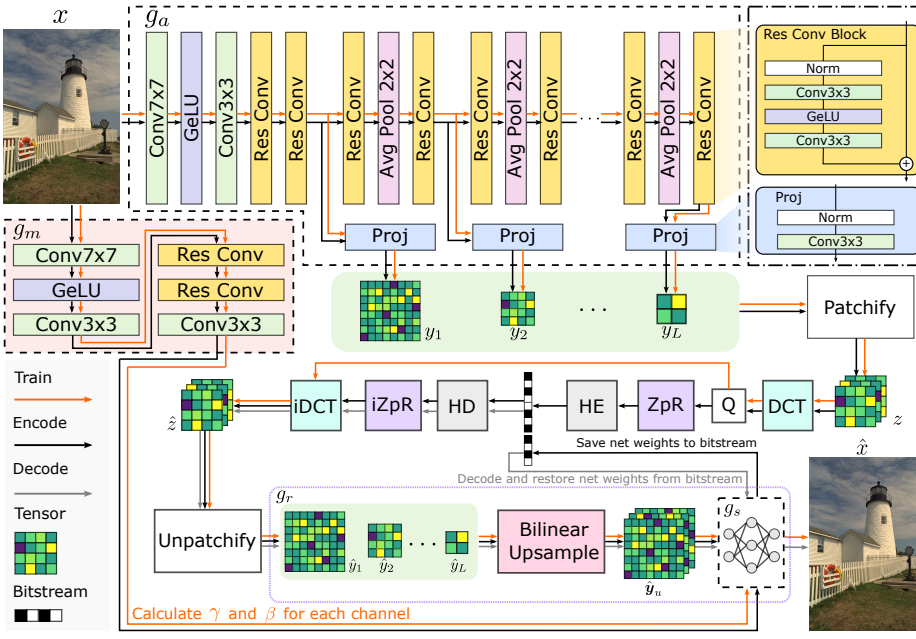


Fig. 1: The framework of PIC. The overall data flow is presented in an S-shaped in the diagram, following $g_a \rightarrow \hat{\boldsymbol{y}} \rightarrow z \rightarrow \hat{z} \rightarrow g_r \rightarrow g_s$. g_a and g_m are latent encoder and modulation net respectively. $\hat{\boldsymbol{y}} = \{y_1, y_2, \dots y_L\}$ are latents generated by latent encoder g_a. All latents are divided into patches of size 8×8 and then concatenated together as z. Similar to other compression methods, z is quantized after DCT transformation. The ZpR module converts the quantized symbols into a more compact form, and we will discuss this module in detail in the Sec. 3.3. HE and HD are Huffman encoder and decoder. iZpR and iDCT are inverse transformation of DCT and ZpR respectively. g_r is part of the decoder and also acts as an image representation model. g_s is synthesis network. γ and β are mean and standard deviation of each channel of g_m output respectively.

where $H_i = \frac{H}{2^{L-i}}, W_i = \frac{W}{2^{L-i}}$. L is the number of latents. To match the resolution of the output image, $\hat{\boldsymbol{y}}$ is upsamlped to $\hat{\boldsymbol{y}}_u \in \mathbb{R}^{C \times H \times W}$ before being fed into the reconstruction network g_s. If we apply g_r in image representation task, image information will be stored in the weights of the network. All parameters, including $\hat{\boldsymbol{y}}$ and weights in g_s, are trainable and optimized jointly via gradient descent.

Obviously, achieving a fast encoding process through training is highly challenging. Therefore, our approach generates all weights of the representation model in a single step. Since the latents inherently lies in a space similar to the image domain, designing a simple yet functional encoder g_a is relatively straightforward, as shown in Fig. 1. The primary challenge is generating the network weights of g_s. While previous works have explored methods for network

weight generation, their performance in compression tasks still leaves room for improvement [8]. In this paper, we adopt a simple strategy called channel-wise normalization and modulation.

We first divide g_s into shared part MLP^s and instance-dependent part MLP^i. Suppose N is batch size and $\boldsymbol{f} \in \mathbb{R}^{N\times C\times H\times W}$ is the intermediate feature between MLP^s and MLP^i, the instance-normalized feature is

$$\bar{\boldsymbol{f}} = \frac{\boldsymbol{f} - \text{mean}(\boldsymbol{f})}{\text{std}(\boldsymbol{f})}. \tag{3}$$

Then modulate the normalized feature $\bar{\boldsymbol{f}}$

$$\tilde{\boldsymbol{f}} = \gamma \odot \bar{\boldsymbol{f}} + \beta, \tag{4}$$

where $\gamma \in \mathbb{R}^{N\times C}$ and $\beta \in \mathbb{R}^{N\times C}$ are generated by modulation net g_m. Fraction and $\odot$ represent element-wise division and multiplication at C dimension respectively. Note g_m will generate corresponding γ and β for each input x. One notable advantage of this transformation is that both the normalization and modulation parameters can ultimately be fused into the network weights of MLP^i, thereby achieving the goal of generating corresponding network weights for each input. For detailed derivation, please refer to the Supplementary.

3.2 Entropy Estimation

In neural representation models, constrained by the overall model size, it is challenging to incorporate a large entropy estimation network. One solution is to employ a compact neural network to enhance RD performance via auto-regressive methods [24]. However, the decoding speed of such approaches remains limited by their auto-regressive design. Inspired by Luo *et al.* [32], we estimate the distribution of DCT parameters to achieve rate estimation accordingly.

For latents generated by g_a

$$\boldsymbol{y} = \{y_i \in \mathbb{R}^{H_i\times W_i}, i = 1, 2, \ldots, L\}, \tag{5}$$

we divide them into patches of size 8×8 , concatenate together as $z \in \mathbb{R}^{n\times 8\times 8}$ and transform to symbols through DCT

$$z = \text{Patchify}(y_1, \ldots, y_L), \tag{6}$$

$$s = \text{DCT}(z). \tag{7}$$

Let $s_{i,k}$ is the k-th DCT component ($k \in \{0, \ldots, 63\}$) of i-th block, the estimated entropy is

$$\mathcal{L}_{\text{entropy}} = -\sum_{i=1}^{n}\sum_{k=0}^{63} \log_2 p_{\theta_k}(s_{i,k} + u), u \sim \text{Uniform}(-0.5, 0.5). \tag{8}$$

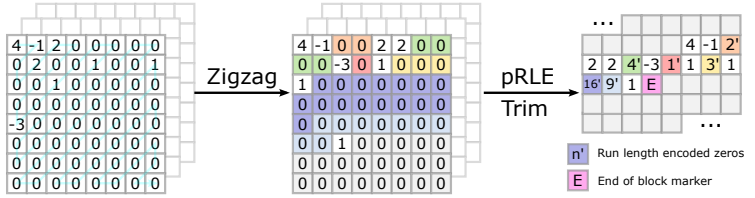


Fig. 2: The pipeline of ZpR module. ZpR transform quantized DCT coefficients to compact symbols through three steps: zigzag reorder, trim trailing zeros and encode zeros using run length encode (partial RLE).

We use 64 channels entropy model, which means we model the distribution of DCT components independently. θ_k is the corresponding parameters of the piecewise linear function for the k-th channel in entropy model [1, 5]. Similar to other compression framework , the entire pipeline is non-differentiable after quantization of s , so we use a simple uniform noise relaxation [1] to build an end-to-end trainable pipeline.

3.3 Full Pipeline and Hardware-affinity Implementation

JPEG reduces bit-rate through lossless compression techniques like the zigzag transform and run-length encoding (RLE). In our PIC, we similarly employ **Z**igzag reordering and **p**artial **R**un-length encoding (ZpR) module to improve RD performance. As shown in Fig. 2, ZpR reduce zeros in zigzag reordered symbols. The main difference between ZpR and RLE in JPEG is we only reduce zeros in symbols. This strategy is more straightforward to implement while effectively reducing the bit-rate.

Unlike JPEG, our method employs image-specific Huffman coding tables, meaning the storage overhead of the Huffman tables also impacts the final bit-rate performance. To ensure $O(1)$ complexity for decoding one symbol, the coding table includs 2^p entries (p is precision) which covers all possible bitstream pattern for decoding one symbol. For small pictures like those in Kodak dataset, the overhead of storing the code tables is non-negligible. Fortunately, the actual number of symbols used in encoding is typically fewer than 128, allowing for further compression of the code tables to reduce this overhead. Benefiting from the prefix code property of Huffman coding, we can apply RLE to compress the original code table, reducing the number of table entries to match the symbol count and significantly decreasing the storage overhead of the original code table.

As illustrated in Fig. 1, we integrated all of the above modules to achieve an trainable codec. The final loss function is

$$\mathcal{L} = \lambda \mathcal{L}_{\text{entropy}} + D(x, \hat{x}), \tag{9}$$

where D is distortion metric, λ balances the trade-off between reconstruction quality and bit-rate.

Another important topic is how to implement an optimized decoder. JPEG's concise design and long-term optimizations have made it one of the fastest encoders. However, due to its early introduction, JPEG has not fully leveraged the hardware advancements in recent years. Benefiting from recent research on parallel Huffman decoding [49] and the emergence of hardware features like Tensor Cores, it has become possible to develop a decoder that surpasses JPEG in performance.

In architecture design, we avoided using relatively large synthesis network like COOL-CHIC-family [21, 24] to comply with Tensor Core's matrix structure requirements. Each layer in our reconstruction network g_s maintains a width of 16, precisely matching the warp matrix multiplication of TF32 operations in Tensor Cores. This alignment maximizes hardware utilization and accelerates reconstruction speed. It should be noted that, although our current implementation relies on NVIDIA's Tensor Cores, other hardware also supports similar types of functionalities, such as AMD's rocWMMA. We believe such acceleration hardware will eventually be supported by more manufacturers' devices in the future as well.

Furthermore, since the precision of TF32 is lower than standard FP32, a gap would arise if training were conducted with FP32 while inference/decoding relied on Tensor Core-accelerated operations. To address this, we draw inspiration from neural rendering techniques and implement the training-phase code using the differentiable shading language Slang [17], which significantly reduces the complexity of integrating inline CUDA code in differentiable framework. This approach ensures consistency between training and inference while maintaining computational efficiency during training.

Qualitatively speaking, our decoder's complexity is actually higher than JPEG's. However, by fully leveraging hardware capabilities, our decoder has surpassed commercial decoders and provided new insights for the design of image codecs.

4 Experiments

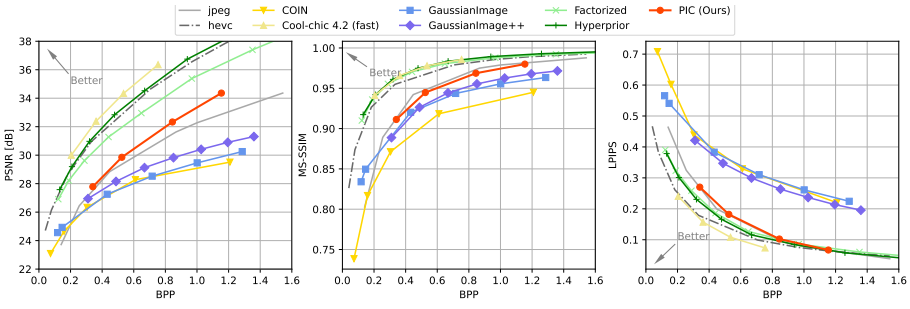
4.1 Experiment Setup

Dataset. Since the proposed PIC is a generalizable method, it is necessary to train our model on a large dataset like end-to-end methods. We use LSDIR dataset [27] as training set, which includes 84,991 natural images. During training, these images were randomly cropped to a resolution of 512×512. Additionally, Our evaluation is conducted on two popular datasets, Kodak[5] and CLIC[6]. Kodak dataset includes 24 images of size 768×512. The CLIC dataset contains 41 high-resolution natural images.

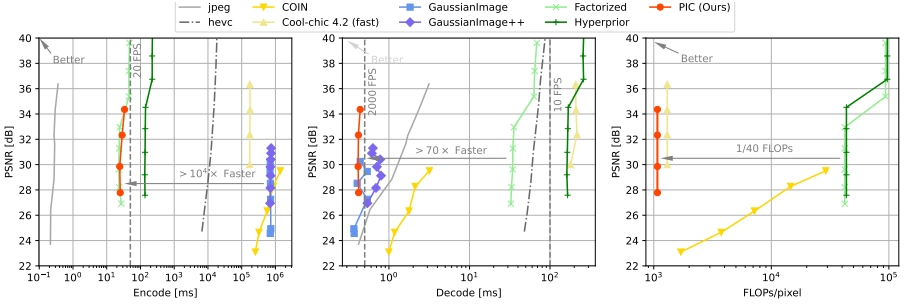
Metrics. We use the peak signal-to-noise ratio (PSNR) in RGB 4:4:4 as distortion metric, which are the most widely used metric in compression. We also report and MS-SSIM [48] LPIPS [52] in main experiment, which is a popular

[5] http://r0k.us/graphics/kodak

[6] https://clic.compression.cc/2021/tasks/index.html



(a) Rate-distortion performance on Kodak dataset.



(b) Speed and complexity comparison of all methods on Kodak dataset.

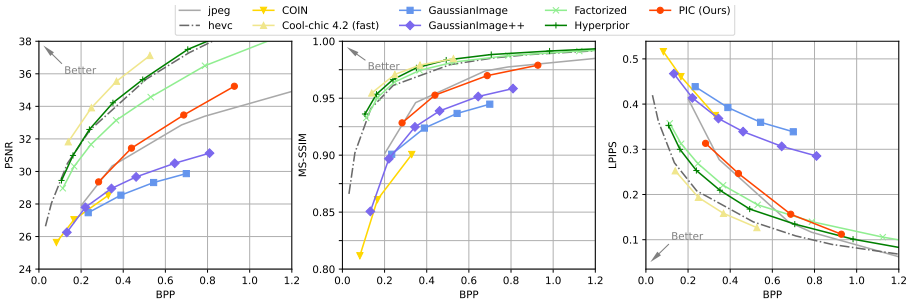
Fig. 3: Results on Kodak dataset.

perceptual metric to measure realism. Bit-per-pixel (BPP) is used as coding efficiency metric. To demonstrates the efficiency of PIC, we report encoding, decoding time and complexity as well.

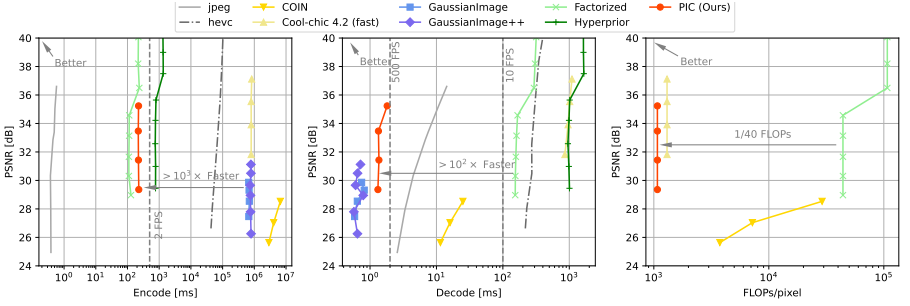
Implementation Details. Our complete framework is implemented in PyTorch, with the key distinction that the synthesis network used in training is built upon Slang-torch [7] to simplify Tensor Core programming within the PyTorch environment. The ZpR module and decoder are CUDA-accelerated, while the Huffman codec is adapted and modified from GPUHD [49]. Other auxiliary components such as bitstream I/O operations are implemented in C++. All experiments are performed on a single NVidia RTX 4090D. We jointly optimize the full set of parameters in g_a, g_m and $g_s.g_s$ is a 4 layers MLP with ReLU activation. The final bitstream includes header information, Huffman code table, encoded latents, and parameters of MLP^i in FP32 format.

Hyper-parameters settings. For the proposed PIC, we use total $L = 8$ latents. This means that the minimum size of images we can encode is 256×256. This also suggests that encoding smaller images requires reducing the value of L. During the training, we set batch size to 16 and randomly cropped the input image to a resolution of 512×512. This optimization is performed separately for each $\lambda = \{0.01, 0.005, 0.002, 0.001\}$ using Adam optimizer and learning rate of 1e-4. All convolution layers have 16 channels. We trained our model for 10

[7] https://github.com/shader-slang/slang-torch



(a) Rate-distortion performance on CLIC dataset.



(b) Speed and complexity comparison of all methods on CLIC dataset.

Fig. 4: Results on CLIC dataset.

epochs in all experiments. Mean squard error (MSE) is used as distortion metric in training.

Benchmarks. Our method is benchmarked against competitive representation based methods like COIN [11] GaussianImage [53] and GaussianImage++ [26]. We also compares with Cool-Chic v4.2 (fast) [21, 24, 25] which is the state-of-the-art method in representation based image codec. We reproduce all the results using the original source code. Another baselines include pretrained Factorized model [1] and Hyperprior [2] in CompressAI [5] and nvJPEG[8], which is a CUDA-accelerated JPEG codec. We also include HEVC in comparison. Note we use MSE as distortion metrics in loss function for PIC in all experiments. For other methods, we follow their original loss function settings, even if they used a loss function different from MSE.

4.2 Quantity and Quality Results

Fig. 3 and Fig. 4 shows the main results of different methods on Kodak dataset and CLIC dataset repectively. Fig. 3a and Fig. 4a demonstrate the RD performance of all methods. All results are averaged on same corresponding hyperparameters setting for each method. PIC achieves better PSNR at high bit-rate region and comparable MS-SSIM and LPIPS performance with JPEG. PIC also

[8] https://developer.nvidia.com/nvjpeg

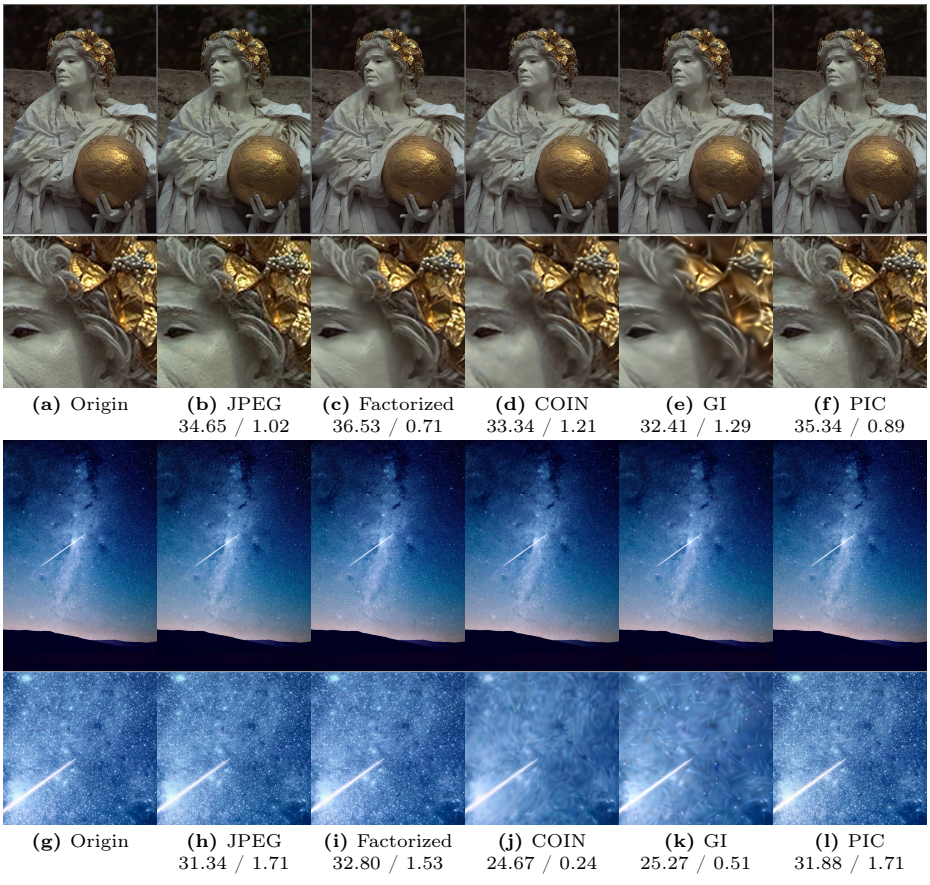
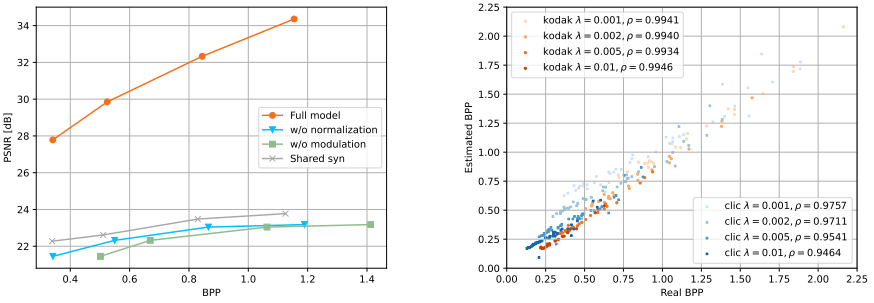


Fig. 5: Visualizations on kodim17.png from Kodak dataset and juskteez-vu-1041.png from CLIC dataset.

outperforms other representation-based methods with similar decoding speed in all quality metrics at high bit rate region.

Fig. 3b and and Fig. 4b present comprehensive comparisons of encoding/decoding speeds across all methods. PIC achieves comparable encoding speed to Factorized model [1] and Hyperprior model [2] but significantly faster decoding speed and lower decoding complexity. Compared with other representation-based approaches [11, 24, 53], PIC has an acceleration of exceeding three orders of magnitude. In terms of decoding speed, our approach ranks second only to GaussianImage [53] while outperforming all other alternatives.

This not only demonstrates the superiority of our proposed method, but also highlights the unique advantages of grid-based approaches in reconstruction quality. Our experiments further validate the performance advantages of COIN [11] and GaussianImage [53] in the low bit-rate region, while these meth-



(a) Ablation of synthesis net g_s architecture. All components are necessary for performance.

(b) Ablation of entropy model. ρ is Pearson correlation coefficient.

Fig. 6: Ablation results.

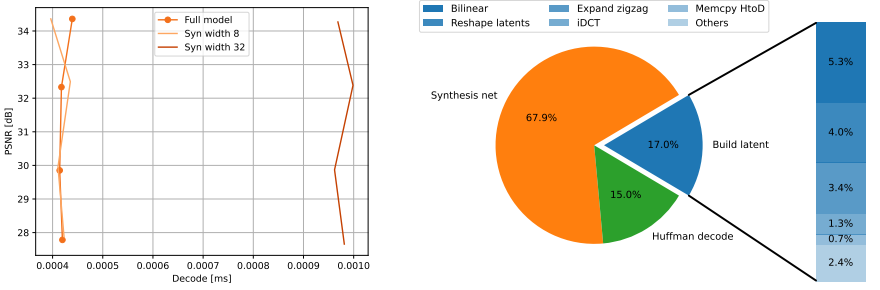
ods exhibit weaker quality improvement compared to other approaches as the bit-rate increases.

Fig. 5 is the visualization of all methods including JPEG [47], Factorized model [1], COIN [11], GaussianImage (abbreviated as GI) [53] and proposed PIC. Below each image are the corresponding PSNR/bpp results. On juskteez-vu-1041.png, we select the model setting with best quality in original paper for COIN and GaussianImage. Our method demonstrates marginally superior performance compared to JPEG while significantly outperforming COIN and GaussianImage. Our approach also achieves comparable results to JPEG in terms of texture details and artifacts, which can be largely attributed to similar entropy modeling. In contrast, the Factorized model tends to produce overly smoothed outputs. COIN exhibits swirling artifacts, and the GaussiaImage displays Gaussian-like speckled artifacts.

Overall, although there is still a certain gap between our method and state-of-the-art approaches in terms of RD performance, PIC has surpassed the widely used JPEG and is a practically deployable approach. For other learning-based methods, the reality is that current architectures, whether based on end-to-end autoencoders or the overfitted Cool-Chic architecture, are constrained by their inherent limitations, making them difficult to deploy in general scenarios. Even the most lightweight Factorized model [1] is hampered by its high computational overhead. For Cool-chic-like models, their excessively slow encoding and decoding speeds render them impractical for most application. Meanwhile, the proposed PIC exhibits no significant weaknesses across key practical metrics. This offers a new perspective for future research on learning-based codec in real-world scenarios.

4.3 Ablation Study

Fig. 6a presents the architectural ablation results of the synthesis network g_s. "Shared syn" denotes using the same synthesis net g_s for all images, in which case our method degenerates into an end-to-end model. Evidently, the model



(a) Decoding speed for different width of synthesis network

(b) Detailed breakdown of decoding time

Fig. 7: Ablation results of decoding speed.

performance under this configuration proves significantly inferior due to the limited capacity of decoder network. "W/o normalization" and "w/o modulation" respectively indicate the exclusion of normalization and modulation in g_s. Unsurprisingly, such configurations significantly degrade model performance. Only when incorporating all modules does our method achieve the desired performance.

Due to the inherent approximation nature of entropy networks, discrepancies with actual performance are unavoidable. Furthermore, post-processing in the ZpR module may potentially amplify these deviations. Fig. 6b illustrates the differences between the actual bit-rate and the estimated bit-rate. Although some discrepancies exist, overall consistency has been maintained. This finding aligns with prior studies [32]. Developing more accurate entropy estimation models remains an important direction for future improvements.

Fig. 7a is the ablation results of decoding speed for different width of synthesis network. Since the minimum matrix size supported by Tensor Cores for the TF32 data type is currently 16, a 16×16 matrix padded with zeros is used when the width is 8. Under this configuration, there is no significant difference in decoding speed between a width of 8 and a width of 16. Overall, the configuration with a width of 16 is the optimal choice, taking into account decoding speed, RD performance, and hardware compatibility. Fig. 7b is detailed breakdown of decoding time including host-to-device I/O time.

We further analyze the latent representation structure to better understand the model's performance. The detailed design of the pRLE module and the structural design of the synthesis network are also investigated. The visulization of latents and more results are included in the Supplementary.

5 Conclusion

In this work, we introduced PIC, an end-to-end INR image coding framework that generates all network parameters in a single forward pass, leading to an

encoder substantially faster than prior representation-based approaches. We further designed a highly optimized decoder that outperforms JPEG in speed while delivering comparable rate-distortion performance. Together, these advances establish a new balance between efficiency and quality, setting a milestone for INR-based compression. While there is still headroom for improving RD performance, our method opens a novel paradigm for practical learning-based image codecs and provides a foundation for extending INR-based compression to other modalities.

Acknowledgements

This work is supported in part by the National Science and Technology Major Project under grant 2025ZD1601000, National Natural Science Foundation of China under grant 62301189, 62576122,62571298, Guangdong Basic and Applied Basic Research Foundation under grant 2026A1515011139.

References

1. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. In: International Conference on Learning Representations (2017)
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)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5470–5479 (2022)
4. Bellard, F.: Bpg image format (2018), `https://bellard.org/bpg/`
5. Bégaint, J., Racapé, F., Feltman, S., Pushparaja, A.: Compressai: a pytorch library and evaluation platform for end-to-end compression research (2020)
6. Catania, L., Allegra, D.: Nif: A fast implicit image compression with bottleneck layers and modulated sinusoidal activations. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 9022–9031 (2023)
7. Chen, H., He, B., Wang, H., Ren, Y., Lim, S.N., Shrivastava, A.: Nerv: Neural representations for videos. Advances in Neural Information Processing Systems **34**, 21557–21568 (2021)
8. Chen, H., Xie, S., Lim, S.N., Shrivastava, A.: Fast encoding and decoding for implicit video representation. In: European Conference on Computer Vision. pp. 402–418. Springer (2024)
9. Chen, Y., Xu, H., Zheng, C., Zhuang, B., Pollefeys, M., Geiger, A., Cham, T.J., Cai, J.: Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In: European conference on computer vision. pp. 370–386. Springer (2024)
10. Dou, Y., Zheng, Z., Jin, Q., Ni, B., Chen, Y., Ke, J.: Real-time neural brdf with spherically distributed primitives. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4337–4346 (2024)
11. Dupont, E., Goliński, A., Alizadeh, M., Teh, Y.W., Doucet, A.: Coin: Compression with implicit neural representations. arXiv preprint arXiv:2103.03123 (2021)

12. Dupont, E., Loya, H., Alizadeh, M., Golinski, A., Teh, Y.W., Doucet, A.: Coin++: Data agnostic neural compression. arXiv preprint arXiv:2201.12904 **1**(2), 4 (2022)
13. Guo, Z., Zhang, Z., Feng, R., Chen, Z.: Soft then hard: Rethinking the quantization in neural image compression. In: International Conference on Machine Learning. pp. 3920–3929. PMLR (2021)
14. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. arXiv e-prints pp. arXiv–1609 (2016)
15. He, D., Yang, Z., Peng, W., Ma, R., Qin, H., Wang, Y.: Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5718–5727 (2022)
16. He, D., Yang, Z., Peng, W., Ma, R., Qin, H., Wang, Y.: Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. arXiv e-prints (2022)
17. He, Y., Fatahalian, K., Foley, T.: Slang: language mechanisms for extensible real-time shading systems. ACM Transactions on Graphics (TOG) **37**(4), 1–13 (2018)
18. Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2d gaussian splatting for geometrically accurate radiance fields. In: ACM SIGGRAPH 2024 conference papers. pp. 1–11 (2024)
19. Jiang, W., Yang, J., Zhai, Y., Ning, P., Gao, F., Wang, R.: Mlic: Multi-reference entropy model for learned image compression. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 7618–7627 (2023)
20. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
21. Kim, H., Bauer, M., Theis, L., Schwarz, J.R., Dupont, E.: C3: High-performance and low-complexity neural compression from a single image or video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9347–9358 (2024)
22. Klocek, S., Maziarka, Ł., Wołczyk, M., Tabor, J., Nowak, J., Śmieja, M.: Hypernetwork functional image representation. In: International Conference on Artificial Neural Networks. pp. 496–510. Springer (2019)
23. Kwan, H.M., Gao, G., Zhang, F., Gower, A., Bull, D.: Hinerv: Video compression with hierarchical encoding-based neural representation. Advances in Neural Information Processing Systems **36**, 72692–72704 (2023)
24. Ladune, T., Philippe, P., Henry, F., Clare, G., Leguay, T.: Cool-chic: Coordinate-based low complexity hierarchical image codec. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13515–13522 (2023)
25. Leguay, T., Ladune, T., Philippe, P., Clare, G., Henry, F.: Low-complexity overfitted neural image codec. arXiv preprint arXiv:2307.12706 (2023)
26. Li, T., Zhang, X., Ge, X., Xu, T., He, D., Zhang, J., Wang, Y.: Gaussianimage++: Boosted image representation and compression with 2d gaussian splatting. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 40, pp. 6442–6449 (2026)
27. Li, Y., Zhang, K., Liang, J., Cao, J., Liu, C., Gong, R., Zhang, Y., Tang, H., Liu, Y., Demandolx, D., et al.: Lsdir: A large scale dataset for image restoration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1775–1787 (2023)
28. Liu, J., Sun, H., Katto, J.: Learned image compression with mixed transformer-cnn architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1–10 (2023)

29. Liu, X., Chen, B., Liu, Z., Wang, Y., Xia, S.T.: An exploration with entropy constrained 3d gaussians for 2d video compression. In: The Thirteenth International Conference on Learning Representations (2025)
30. Liu, X., Chen, J., Chen, B., Liu, Z., An, B., Xia, S.T., Wang, Z.: An efficient implicit neural representation image codec based on mixed autoregressive model for low-complexity decoding. arXiv preprint arXiv:2401.12587 (2024)
31. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20654–20664 (2024)
32. Luo, X., Talebi, H., Yang, F., Elad, M., Milanfar, P.: The rate-distortion-accuracy tradeoff: Jpeg case study. arXiv preprint arXiv:2008.00605 (2020)
33. Mentzer, F., Toderici, G.D., Tschannen, M., Agustsson, E.: High-fidelity generative image compression. Advances in neural information processing systems **33**, 11913–11924 (2020)
34. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis (2020)
35. Minnen, D., Ballé, J., Toderici, G.: Joint autoregressive and hierarchical priors for learned image compression (2018)
36. Minnen, D., Singh, S.: Channel-wise autoregressive entropy models for learned image compression (2020)
37. 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)
38. Schlag, I., Irie, K., Schmidhuber, J.: Linear transformers are secretly fast weight programmers. In: International conference on machine learning. pp. 9355–9366. PMLR (2021)
39. Schlag, I., Schmidhuber, J.: Gated fast weights for on-the-fly neural program generation. In: NIPS Metalearning Workshop (2017)
40. Sendera, M., Przewięźlikowski, M., Karanowski, K., Zięba, M., Tabor, J., Spurek, P.: Hypershot: Few-shot learning by kernel hypernetworks. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 2469–2478 (2023)
41. Skodras, A., Christopoulos, C., Ebrahimi, T.: The jpeg 2000 still image compression standard. IEEE Signal processing magazine **18**(5), 36–58 (2001)
42. Skorokhodov, I., Ignatyev, S., Elhoseiny, M.: Adversarial generation of continuous images. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10753–10764 (2021)
43. Sztrajman, A., Rainer, G., Ritschel, T., Weyrich, T.: Neural brdf representation and importance sampling. In: Computer Graphics Forum. vol. 40, pp. 332–346. Wiley Online Library (2021)
44. Vaidyanathan, K., Salvi, M., Wronski, B., Akenine-Möller, T., Ebelin, P., Lefohn, A.: Random-access neural compression of material textures. ACM Transactions on Graphics (2023)
45. Volk, T., Ben-David, E., Amosy, O., Chechik, G., Reichart, R.: Example-based hypernetworks for out-of-distribution generalization. arXiv preprint arXiv:2203.14276 (2022)
46. Von Oswald, J., Henning, C., Grewe, B.F., Sacramento, J.: Continual learning with hypernetworks. arXiv preprint arXiv:1906.00695 (2019)
47. Wallace, G.K.: The jpeg still picture compression standard. IEEE transactions on consumer electronics **38**(1), xviii–xxxiv (1992)

48. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. vol. 2, pp. 1398–1402. Ieee (2003)
49. Weißenberger, A., Schmidt, B.: Massively parallel huffman decoding on gpus. In: Proceedings of the 47th International Conference on Parallel Processing. pp. 1–10 (2018)
50. Xia, Y., Zhou, Y., Wang, J., An, B., Wang, H., Wang, Y., Chen, B.: Diffpc: Diffusion-based high perceptual fidelity image compression with semantic refinement. In: The Thirteenth International Conference on Learning Representations (2025)
51. You, T., Kim, M., Kim, J., Han, B.: Generative neural fields by mixtures of neural implicit functions. Advances in Neural Information Processing Systems **36**, 20352–20370 (2023)
52. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
53. Zhang, X., Ge, X., Xu, T., He, D., Wang, Y., Qin, H., Lu, G., Geng, J., Zhang, J.: Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. arXiv preprint arXiv:2403.08551 (2024)
54. Zou, R., Song, C., Zhang, Z.: The devil is in the details: Window-based attention for image compression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 17492–17501 (2022)