Neural Block Compression: Variable Bitrates Feature Blocks for Texture Representation

Rui Shi[1], Yishun Dou[2], Zhong Zheng[2], Xiangzhong Fang[1*], Wenjun Zhang[1], Bingbing Ni[1*]

[1]Shanghai Jiao Tong University, Shanghai 200240, China
[2]Huawei, Shanghai, China
{shi-rui,xzfang,nibingbing}@sjtu.edu.cn

Abstract

The imperative for compression of material textures emerges from the critical demand for high-quality rendering, which necessitates sophisticated textures that, in turn, require substantial storage and memory resources. Thus, low-bitrate compression is crucial, especially in modern games demanding higher texture resolutions. Concurrent methodologies in texture compression predominantly employ a block-based paradigm based on color space, which inevitably leads to representational redundancies and a limited compression scope, particularly at lower bitrates. In the context of mobile devices, bandwidth during texture loading and runtime memory are major bottlenecks, making existing compression algorithms inadequate for high-resolution textures. To mitigate these limitations, we propose a novel multi-resolution texture compression scheme, *Neural Block Compression* (NBC), developed within the neural feature domain. Our encoding scheme is constructed on a hierarchy of multi-resolution neural feature blocks, and the key ingredient is the variable bitrates quantization scheme. It allocates higher bitrates to higher feature mip-levels and lower bitrates to lower feature mip-levels, thereby extending the concept of block compression from color domain into neural feature domain. Extensive experiments demonstrate the superior texture compression quality achieved by the proposed scheme, especially at low bitrates.

1 Introduction

Textures constitute a fundamental component in modern 3D graphics, serving as a foundational element across many applications. They encapsulate critical surface characteristics, lighting effects, and physical attributes, significantly enhancing the visual authenticity of graphical representations. Nevertheless, textures pose a twofold challenge in real-time rendering: *i*) their extensive utilization strains graphics accelerator memory bandwidth, and *ii*) their storage requirements can impose burdens on disk space and download bandwidth. For instance, in the case of an 8K texture with 24-bit depth, the required storage space amounts to 192 megabytes (MB). To alleviate these issues, texture compression techniques (Knittel et al. 1995; Torborg and Kajiya 1996; Beers et al. 1996) have been developed.

*Corresponding authors: Xiangzhong Fang and Bingbing Ni.

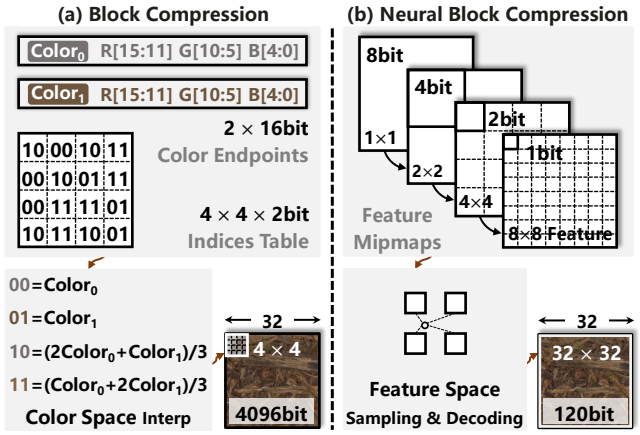

Figure 1: Motivation. (a) Block Compression (BC1) (Microsoft Corporation 2020) stores two 16-bit color endpoints for every block of 4×4 texels, along with a 2-bit index value assigned to each texel in the block. It compresses 16 texels with 64 bits, resulting in 1.33 bits per pixel per channel (BPPC). (b) We extend the concept of block compression from color domain into neural feature domain. NBC compresses textures into a hierarchy of quantized feature mipmaps, each with a distinct bitrate, and employs a residual fusion scheme, to sum up low-bit feature displacements to high-bit base features. It can compress 1024 texels with 120 bits, yielding a compression ratio of 0.04 BPPC.

Most traditional texture compression formats (Ström et al. 2004, 2005, 2007; Microsoft Corporation 2020; Nystad et al. 2012) follow the block-based paradigm originated in block truncation coding (Delp and Mitchell 1979; Fränti et al. 1994), as shown in Fig. 1 (a). In these approaches, textures are divided into blocks, storing high-bitrate base colors and texel-wise low-bitrate displacements. These block-based texture compression techniques, characterized by their capacity to deliver high-quality compression performance with efficient hardware implementations, are widely used in contemporary GPUs. However, these methods are ill-suited for low-bitrate regimes and suffer from poor performance at lower compression bitrates. They operate only in the original color space, completely ignoring the large redundancy existing in the transformed feature domain.

As a representative of neural texture compression methods, NTC (Vaidyanathan et al. 2023) introduces a neural compression technique specifically designed for texture materials, supporting real-time decompression with random access. Following the paradigm of implicit neural representations (INR) (Müller et al. 2022; Takikawa et al. 2021), NTC employs quantized neural latent grids for the storage of spatially-varying appearance and a low-cost multi-layer perceptron (MLP) for decoding. Compared to traditional texture compression formats, NTC mitigates redundancy within mipmap levels and among distinct texture materials. However, NTC utilizes a fixed quantization bitrate, falling short of performing texture compression at low bitrates.

In this work, we address the limitations of traditional texture compression methods and NTC, with a novel variable bitrates quantization scheme built on a hierarchy of multi-resolution feature blocks for texture compression, named *Neural Block Compression* (NBC). Drawing inspiration from conventional block-based texture compression paradigms (Ström et al. 2004, 2005, 2007; Microsoft Corporation 2020; Nystad et al. 2012), as illustrated in Fig. 1, we extend the concept of block compression (BC) from color domain into neural feature domain, representing compressed textures with multi-resolution feature blocks, where each block contains a high-bit base feature and each grid within the block stores a low-bit feature displacement.

More concretely, we allocate higher bitrates to higher feature mip-levels in the multi-resolution blocks hierarchy and lower bitrates to lower feature mip-levels. Based on the multi-resolution blocks hierarchy, we design a residual fusion scheme, namely, we can sum up the high-bitrate features from the coarsest mipmap to any of the preceding mipmaps, yielding feature representations with gradually increased texture details, *i.e*., from low-frequency to high-frequency contents. At the compression stage, feature vectors within the multi-resolution blocks are optimized through quantization-aware training and backpropagation, together with the MLP decoder. While at the decompression stage, features are sampled and subsequently fed into the decoder to carry out texture decompression.

To summarize, we make the following key contributions:

- We propose a novel ***variable bitrates quantization*** scheme NBC for ***low bitrate*** texture compression, extending the conventional block compression paradigm from color space into neural feature domain.
- NBC delivers remarkably low bitrate texture compression, maintaining an acceptable or comparable quality level. While at higher bitrates, it attains a performance that is either on par with, or superior to, leading image and texture compression techniques.

2 Related Works

2.1 Texture Compression

Block-based paradigm is extensively utilized in the realm of classical texture compression, where textures are represented with high-bit base colors at the block level and low-bit displacements at the texel level. This approach was initially introduced in block truncation coding (Delp and Mitchell 1979) for compression of grey-scale images, where each 4×4 pixel block stores two 8-bit colors, alongside an additional 1-bit per pixel for selecting one of the two colors.

S3 Texture Compression (S3TC) (Iourcha et al. 1997) extends the idea and has yielded seven variant schemes, denoted as BC1-BC7 (Microsoft Corporation 2020). The BC1 compression format represents a texture on two levels, *i.e*., block level and texel level. Each block of 4×4 texels stores two 16-bit colors, which are the endpoints of a line segment in the RGB space. While for each texel in the block, a 2-bit index is assigned to interpolate the intermediate color on the line segment. The remaining S3TC variants are purposefully designed for the compression of alpha channels, normal maps, and high-dynamic range (HDR) textures.

PACKMAN (Ström et al. 2004) employs a storage strategy that encompasses a 12-bit base color and 2-bit per-texel luminance modulation index within 2×4 blocks. iPACKMAN (Ström et al. 2005) groups two adjacent blocks into a 4×4 block. It has been standardized as Ericsson Texture Compression (ETC1) and later evolved into ETC2 (Ström et al. 2007), which introduces new modes and enhances the quality of compression further. Adaptive Scalable Texture Compression (ASTC) (Nystad et al. 2012) adheres to the standard block-based paradigm, supporting a broad spectrum of textures, *e.g*., LDR, HDR, 3D textures, *etc*.

2.2 Neural Image Compression

Neural image compression (Ballé et al. 2017; Li et al. 2018; Rippel et al. 2017; Theis et al. 2017) offers an alternative to traditional compression methods, demonstrating the ability to achieve high-quality compression, even at low bitrates. Typically, these methods employ an autoencoder framework, comprising an encoder, quantizer, entropy model, and decoder, which is optimized in an end-to-end manner by minimizing a rate-distortion trade-off loss function. Recent advancements in neural image compression have witnessed various improvements, including coarse-to-fine hierarchical hyperprior (Hu, Yang, and Liu 2020), discretized Gaussian mixture likelihoods (Cheng et al. 2020), attention modules (Chen et al. 2021), and INR-based compression (Strümpler et al. 2022; Girish et al. 2023).

NTC (Vaidyanathan et al. 2023) integrates neural image compression techniques into the field of texture compression, built upon the framework of INR. It compresses textures into a feature pyramid containing quantized latent vectors, and utilizes an MLP decoder for decompression.

2.3 Variable Bitrates Compression

Variable bitrates compression methods typically target progressive streaming, where increased bitrates lead to higher-quality decompression. It can be applied to various signals, *e.g*., neural fields (Davies et al. 2021; Dupont et al. 2021; Lu et al. 2021). VQ-AD (Takikawa et al. 2022) proposes a dictionary-based approach for compressing feature grids, where feature vectors are replaced with indices referencing a learned codebook. While in the field of image compression, variable bitrates compression methods mainly include RNN-based autoencoders (Johnston et al. 2018; Toderici et al. 2017, 2016), conditional autoencoders (Choi,

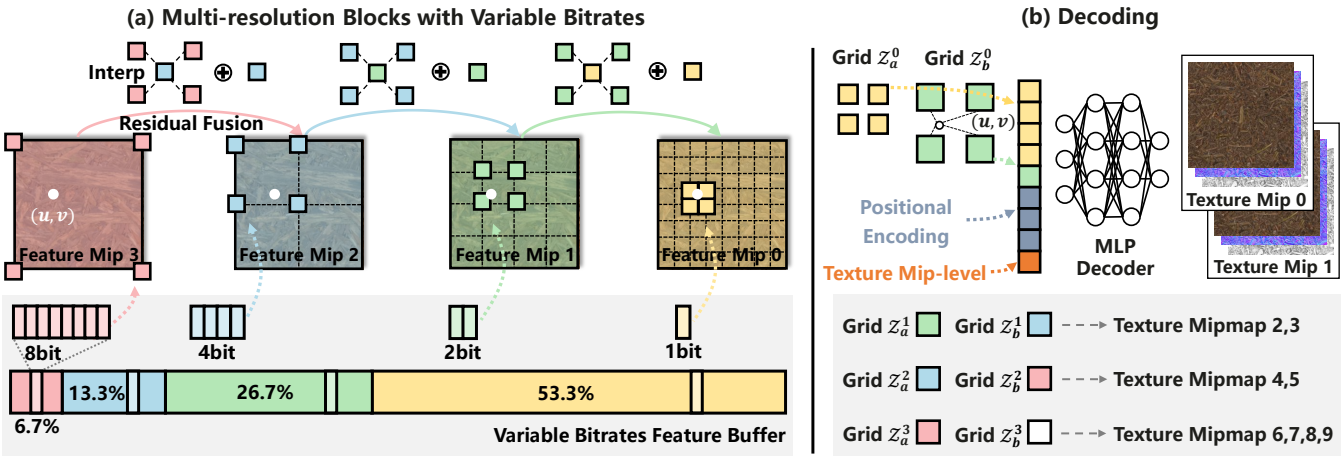

Figure 2: Pipeline. (a) Our compressed representation is a hierarchy of multi-resolution blocks quantized with variable bitrates, wherein a residual fusion scheme implemented by interpolation and addition is introduced to add low-bit displacements to high-bit base features progressively. (b) We concatenate four adjacent feature vectors from $\mathcal{Z}_a$, bilinearly interpolate $\mathcal{Z}_b$, and concatenate them with positional encoding as well as the target texture mipmap level for feature sampling. A lightweight MLP is shared across all mipmap levels, carrying out texture decompression of the target texture mipmap level.

El-Khamy, and Lee 2019; Yang et al. 2020), and channel-wise attention-based autoencoders (Zhong et al. 2020). In (Zhong et al. 2020), distinct bitrates are allocated to different channels according to their channel significance determined by a channel importance module. AG-VAE (Cui et al. 2021) introduces a continuous rate-adjustable learnable image compression framework, leveraging a pair of gain units to achieve discrete rate adaptation within a singular model. These rate adaptation methods usually adapt their rates based on the input data, but rather, our "*variable bitrates*" refers to distinct quantization bitrates in different feature mipmaps of the multi-resolution feature blocks.

3 Methodology

Inspired by block-based texture compression paradigm (Ström et al. 2004, 2005, 2007; Nystad et al. 2012; Microsoft Corporation 2020), we design a hierarchy of multi-resolution feature blocks with variable bitrates to perform neural texture compression, where higher bitrates are allocated to higher feature mipmaps (mip-level 3) and lower bitrates are assigned to lower mipmaps (mip-level 0). This extends the conventional block-based texture compression routine, which primarily operates at the block level (high-bit base colors) and the texel level (low-bit displacements), into *multiple neural feature levels*. With this design, texture compression achieves unprecedented levels of low bitrates while upholding acceptable or comparable quality standards.

As depicted in Fig. 2, the compressed representation of textures is a hierarchy of quantized multi-resolution feature blocks, each feature mipmap with a distinct bitrate. Features are sampled from the multi-resolution blocks and fed into a decoder to reconstruct the decompressed texture. Our compressed representation undergoes optimization directly through simulated quantization and backpropagation via the decoder, intending to minimize the reconstruction loss.

Feature Mip-level ℓ	$\mathcal{Z}_a^\ell$ Res.	$\mathcal{Z}_b^\ell$ Res.	Texture Mip-level
0	1024	512	$0, 1$
1	512	256	$2, 3$
2	256	128	$4, 5$
3	128	64	$6, 7, 8, 9$

Table 1: Multi-resolution Blocks Representation. We show feature mip-level, texture mip-level, and feature resolutions for a 4096×4096 texture set. (Res. short for resolution.)

3.1 Preliminary: Block-based Compression

Despite the diversity of methods employed in the block-based texture compression paradigm, their core conceptual framework is unified: textures are represented with high-bit *base* colors at the block level and low-bit *displacements* at the texel level. As a representative of classical block-based texture compression methods, BC1 allocates two 16-bit base colors to each 4×4 texel block, essentially forming two endpoints of a linear spectrum in the RGB color space. Within these blocks, each individual texel is endowed with a 2-bit displacement value. This facilitates the generation of intermediate colors through interpolation along the segment defined by the base colors, thereby achieving a nuanced and efficient color representation within the block.

3.2 From Color Blocks to Neural Feature Blocks

NBC extends the block representation in block-based texture compression from *color* (RGB) blocks to neural *feature* blocks, as illustrated in Fig. 1. It is built upon a hierarchy of multi-resolution feature blocks, containing quantized learnable latent feature vectors organized in grids of different resolutions. The multi-resolution blocks hierarchy, denoted as $\mathcal{Z}$, exhibits a progressive reduction in resolution by a factor of k at each successive mip-level. Within each grid cell, a c-dimensional quantized feature vector is stored.

The representation follows the block-based paradigm, in that, the feature vector in a grid cell in the first feature mipmap $\mathcal{Z}^0$ corresponds to a texel block in the texture, while that in feature mipmap $\mathcal{Z}^\ell$ (which can be considered as *base* feature) maps to a $k \times k$ feature block in the preceding feature mipmap $\mathcal{Z}^{\ell-1}$ (feature *displacements*).

In this work, we utilize a hierarchy of multi-resolution blocks with 4 levels, with the resolution reduction factor k set as 2. Following Vaidyanathan et al., we design two grids with different resolutions for each feature mipmap $\mathcal{Z}^\ell$, namely $\mathcal{Z}_a^\ell$ in high resolution and $\mathcal{Z}_b^\ell$ in low resolution. Each feature mipmap $\mathcal{Z}^\ell$ represents two or more texture mip-levels. Tab. 1 showcases the feature mip-levels, texture mip-levels, and grid resolutions for a 4096×4096 texture set.

3.3 Variable Bitrates

Based on multi-resolution neural feature blocks, we propose *variable bitrates quantization*, following the block-based texture compression paradigm. We allocate higher quantization bitrates for coarser mip-levels within the multi-resolution blocks, acting as *base* features for blocks in the preceding mip-level. Conversely, a lower quantization bitrate is assigned for the preceding mip-level, functioning as low-bitrate *displacements* for feature grids in the block.

As a typical configuration, we quantize all latent values in feature grids in $\mathcal{Z}^3$ with 8 bits, $\mathcal{Z}^2$ with 4 bits, $\mathcal{Z}^1$ with 2 bits, and $\mathcal{Z}^0$ with 1 bit (abbreviated as 8-4-2-1 configuration). With varying configurations of variable bitrates and multi-resolution feature blocks (*i.e.*, feature vector dimension, quantization bitrate, and resolution), NBC enjoys enhanced flexibility in terms of compression ratios.

Residual Fusion. We design a residual fusion scheme for features in the hierarchy of multi-resolution feature blocks, progressively adding low-bitrate feature displacements to high-bitrate base features. Concretely, the feature $\hat{\mathcal{Z}}^\ell$ (ready for decoding) at mip-level ℓ is obtained by adding the feature stored in $\mathcal{Z}^\ell$ to the interpolation of feature $\hat{\mathcal{Z}}^{\ell+1}$ from the subsequent mip-level $\ell+1$, which is similarly acquired by residual fusion. Therefore, we can recursively derive the features at mip-level ℓ using this residual fusion process:

$$\begin{aligned} \hat{\mathcal{Z}}^\ell &= \mathcal{Z}^\ell \oplus \mathsf{interp}(\hat{\mathcal{Z}}^{\ell+1}), \quad \ell = 0, \cdots, L-1, \\ \hat{\mathcal{Z}}^L &= \mathcal{Z}^L, \end{aligned} \tag{1}$$

where $\oplus$ denotes element-wise addition, $\mathsf{interp}(\cdot)$ denotes bilinear interpolation, and L is the maximum feature mip-level in the hierarchy, set as 3 in this work.

Quantization. Quantization converts full-precision real numbers ($r \in \mathbb{R}$) to quantized values ($q \in \mathbb{R}$). We follow the quantization strategy used in NTC (Vaidyanathan et al. 2023). We define the quantization range as $[R_{\text{min}}, R_{\text{max}}]$, typically adopting $R_{\text{min}} = -R_{\text{max}}$. For n-bit quantization, the quantization range is uniformly split into 2^n bins, yielding bin size of $B_n = \frac{1}{2^n} \times 2R_{\text{max}}$. All real values in a quantization bin are quantized to the bin's central value. To ensure the real value $r = 0$ is quantized to $q = 0$ with no quantization error, we follow (Jacob et al. 2018) to shift the quantization range to the right by half of the bin size, while maintaining R_{max} as the upper threshold. Consequently, we get an asymmetric quantization range of $\left[R_{\text{max}}\left(-1+\frac{1}{2^n}\right), R_{\text{max}}\right]$. The quantization scheme is represented as:

$$q = \mathsf{round}\left(\frac{r}{B_n}\right) \times B_n, \quad \text{for } n \in \{1, 2, 4, 8\}, \tag{2}$$

where n is the number of bits in the variable bitrates configuration, and $\mathsf{round}(\cdot)$ denotes rounding to the nearest integer. Tab. 2 provides a detailed illustration of the variable bitrates configuration, allowing for the adjustment of the number of bits to accommodate different compression rates.

Feature Mip-level ℓ	Bits	Bins	R_{min}	R_{max}	Bin Size
0	1	2	-0.256	0.512	0.512
1	2	4	-0.384	0.512	0.256
2	4	16	-0.480	0.512	0.064
3	8	256	-0.510	0.512	0.004

Table 2: Variable Bitrates Configuration. We showcase the detailed quantization configuration for each feature mip-level. The fundamental principle underlying variable bitrates quantization is the allocation of higher bitrates to higher feature mip-levels and lower bitrates to lower ones.

Variable Bitrates Simulated Quantization. To circumvent gradient issues induced by the rounding function at training time, we follow (Ballé et al. 2017) to employ a simulated quantization strategy, where the quantization is simulated by an additive uniform noise in the range $\left(-\frac{B_n}{2}, \frac{B_n}{2}\right)$. Under the context of variable bitrates and residual fusion, we extend the simulated quantization to accommodate the variable bitrates design. For feature vectors stored at each mip-level within the multi-resolution blocks, we simulate quantization based on the specific quantization configuration designated for that level, which is performed on $\mathcal{Z}^\ell$ prior to the residual fusion operation. Subsequently, the residual fusion scheme sums up the simulated quantized features level by level, as depicted in Eq. 1, yielding feature representations of the target feature mip-level.

3.4 Decoding

Feature Sampling. The quantized feature vectors in multi-resolution blocks are sampled before being fed into the decoder. Given a query point (u, v) together with its mipmap level, the first step involves selecting the corresponding feature mip-level ℓ (as specified in Tab. 1). At the selected feature mip-level ℓ, to sample features from the high-resolution feature grid $\hat{\mathcal{Z}}_a^\ell$, we concatenate four adjacent feature vectors around the query point. In contrast, for low-resolution $\hat{\mathcal{Z}}_b^\ell$, we utilize bilinear interpolation. It is worth noting that the features referred to here are not those stored within multi-resolution blocks, but rather, they represent features that have undergone processing via residual fusion as Eq. 1.

To facilitate the network's capacity to faithfully retain high-frequency intricacies, we adopt positional encoding techniques (Mildenhall et al. 2021; Müller et al. 2021), leveraging periodic $\sin(2^h \pi x)$ and $\cos(2^h \pi x)$ functions to project input coordinates into a higher dimensional space.

MLP Decoder. The decoder is shared across all mipmap levels, taking as input the concatenation of sampled features, positional encoding and normalized texture mip-level, ultimately carrying out texture decompression. It is architectured as a lightweight MLP comprising two hidden layers, each with 64 channels. Each layer within the MLP employs GELU (Hendrycks and Gimpel 2016) activation function.

4 Implementation

4.1 Feature Reuse

As discussed in Sec. 3.2, to enable each feature mipmap to capture not only high-frequency details but also low-frequency contents, we design two grids with different resolutions for each feature mip-level ℓ. For memory efficiency consideration, we propose a *feature reuse strategy* across feature mipmaps. Specifically, the high-resolution $\mathcal{Z}_a^\ell$ is exactly the grid stored in the current feature mipmap $\mathcal{Z}^\ell$, while the low-resolution $\mathcal{Z}_b^\ell$ leverages the reuse of grid stored in the subsequent feature mipmap $\mathcal{Z}^{\ell+1}$. Notably, there is no feature grid reuse for $\mathcal{Z}_b^L$ in the highest feature mip-level L, instead, an extra feature grid of lower resolution is introduced (see $\mathcal{Z}_b^3$ in Fig. 2 (b)). The reuse of features enforces representation consistency across feature mipmaps and contributes to an enhanced compression ratio within the representation of multi-resolution feature blocks.

4.2 Compression & Decompression

We implement NBC in PyTorch. The compressed representation of textures in NBC is a hierarchy of multi-resolution feature blocks, together with a lightweight MLP decoder.

Compression. The model is trained using an auto-decoder framework (Park et al. 2019), where the multi-resolution feature blocks and the decoder are jointly optimized. We initialize each feature mipmap $\mathcal{Z}^\ell$ of the multi-resolution blocks using uniform distribution within the same quantization range as this mip-level. The training loss is measured by the L2 distance between the decompressed texture and the ground truth texture. We employ Adam optimizer (Kingma and Ba 2014) with an initial learning rate of 0.01 for multi-resolution feature blocks and 0.005 for the MLP decoder. We train the model for 20k iterations and apply cosine annealing (Loshchilov and Hutter 2016) to facilitate learning rate reduction, gradually decreasing the learning rate to 0 at the end of the training process.

At training time, we simulate quantization by introducing additive uniform noise to feature vectors. While within the final 10% of training iterations, explicit quantization of the feature vectors is performed to further optimize the MLP decoder to adapt to quantized features. At this stage, feature blocks are frozen, and optimization focuses on the decoder.

Decompression. In the decompression phase, quantized feature vectors are sampled from the multi-resolution blocks and decoded via the MLP decoder. A notable attribute of NBC is its *random access* capability. Specifically, each texel correlates to a feature vector of a fixed bit size. This architecture permits direct indexing of the target features during the extraction of individual texels, thereby enhancing the efficiency of the decompression process.

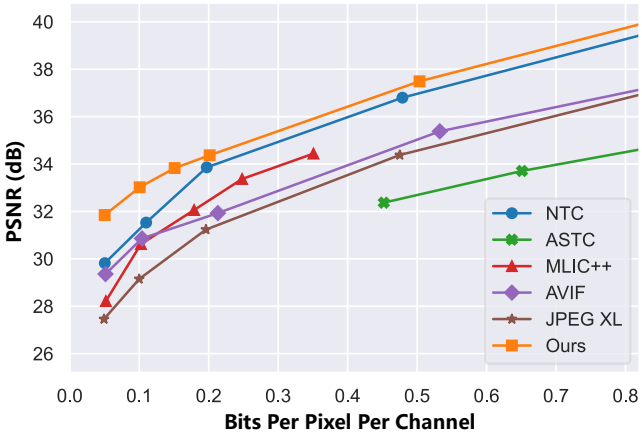

Figure 3: Quantitative Results. The plot illustrates the mean PSNR scores of different methods over the entire dataset (y-axis) across different BPPCs (x-axis). NBC outperforms others under different BPPCs, especially at low bitrates.

5 Experiments

NBC exhibits flexibility by allowing the adjustment of resolutions of multi-resolution feature blocks and the configuration of variable bitrates. This design allows it to achieve an unprecedentedly low bitrate at compression while maintaining an acceptable or comparable performance, or alternatively, to outperform other texture compression methods with an equivalent high bitrate.

To validate the superior performance of NBC, we experiment extensively on diverse textures under different compression ratios. We employ bits per pixel per channel (BPPC) as a metric to quantify the compression ratio, and design multiple configurations tailored to various BPPCs.

Datasets. We collect a dataset from open-source texture assets databases, ambientCG, EISKO, KaiMoisch, and PolyHaven. It contains 16 texture sets, varying from 2048^2 to 8192^2. Typically, a texture set consists of multiple material properties, *e.g.*, diffuse color, normal map, displacement map, ambient occlusion, roughness, and metalness. We concatenate these materials along the channel dimension for each texture set.

Baselines. We compare our method qualitatively and quantitatively against NTC (Vaidyanathan et al. 2023), as well as conventional block-based texture compression techniques BC (Microsoft Corporation 2020) and ASTC (Nystad et al. 2012). To provide a fair assessment, we also compare it with high-quality image compression methods that utilize entropy encoding (*i.e.*, AVIF (Chen et al. 2018) and JPEG XL (Alakuijala et al. 2019)), and state-of-the-art neural image compression method MLIC++ (Jiang and Wang 2023).

Given the unavailability of the source code for NTC, we reimplement it with PyTorch, adhering to the techniques described in (Vaidyanathan et al. 2023). For MLIC++, we use the officially released code. For other methods, we utilize the publicly available command-line tools. For BC, we apply BC4 for single-channel textures, BC7 for normals and BC1 for the other three-channel textures. For ASTC, we choose

	∼ 0.05 BPPC		∼ 0.1 BPPC				∼ 0.2 BPPC			∼ 0.5 BPPC				∼ 1.0 BPPC			
	Ours	**MLIC++**	**NTC**	**JPEG XL**	**Ours**	**MLIC++**	**NTC**	**Ours**	**AVIF**	**ASTC**	**NTC**	**Ours**	**AVIF**	**JPEG XL**	**Ours**	**NTC**	**ASTC**
BPPC	0.050	0.052	0.099	0.100	0.101	0.103	0.197	0.201	0.213	0.452	0.479	0.504	0.533	0.896	0.921	0.958	1.017
PSNR ↑	**31.85**	28.22	31.53	29.15	**33.02**	30.62	33.89	**34.38**	31.93	32.37	36.80	**37.49**	35.38	37.48	**40.66**	40.48	35.66
SSIM ↑	**0.832**	0.733	0.835	0.771	**0.891**	0.821	0.908	**0.913**	0.883	0.884	0.939	**0.956**	0.945	0.946	**0.976**	0.969	0.939
LPIPS ↓	**0.195**	0.409	0.183	0.289	**0.123**	0.258	0.090	**0.085**	0.123	0.085	0.054	0.040	**0.034**	**0.018**	0.023	0.031	0.033
ꟻLIP ↓	**0.102**	0.138	0.092	0.097	**0.081**	0.102	0.071	**0.065**	0.067	0.065	0.050	0.050	**0.042**	**0.028**	0.037	0.042	0.044

Table 3: Performance Comparison under Different BPPC Levels.

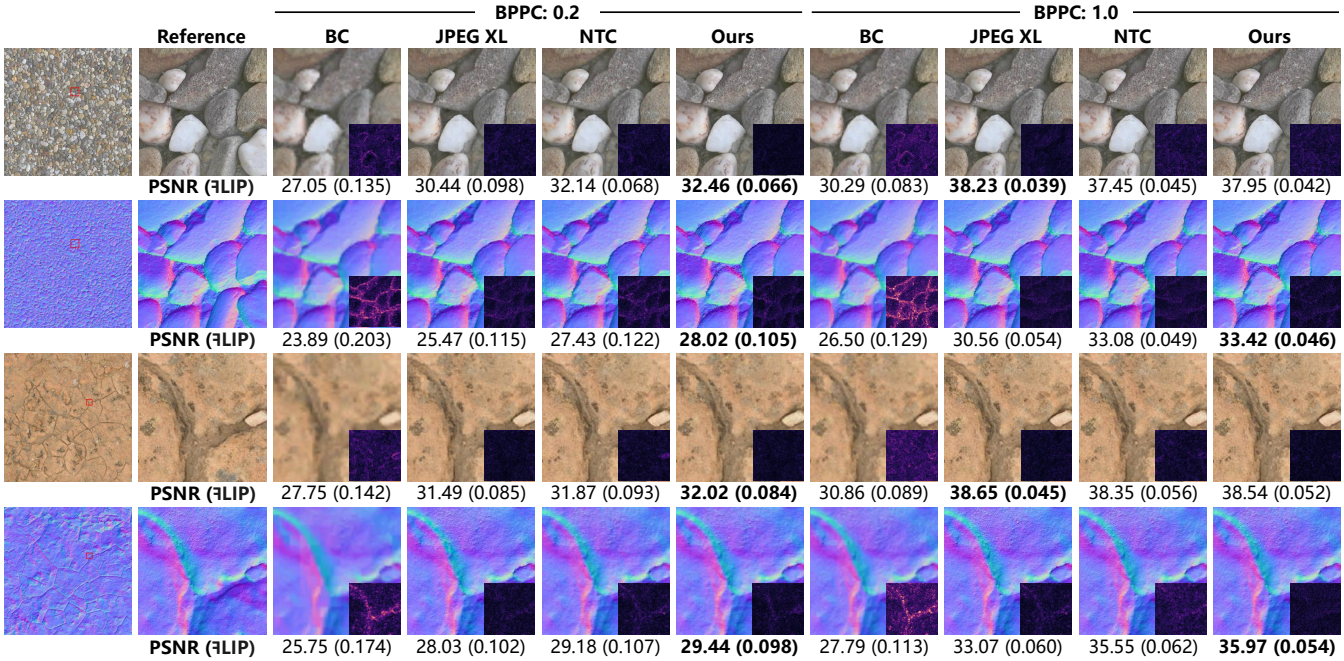

Figure 4: Iso-storage Comparison. We compare BC, JPEG XL, NTC and NBC at two distinct compression ratios, 0.2 and 1.0 BPPC. Additionally, we provide a quantitative comparison in terms of PSNR and ꟻLIP. The ꟻLIP error map is displayed, with brightness proportional to the ꟻLIP error. We multiply ꟻLIP by 2 in the error map under 1.0 BPPC for better visualization.

3 variants in terms of the block size, specifically, 12×12, 8×8 and 5×5, targeting different compression ratios. Since it is infeasible to directly specify the quality configuration of AVIF and JPEG XL based on the desired compression ratio, we employ a binary search technique to determine the optimal quality parameter for these two methods, until the desired BPPC is achieved.

Metrics. For a comprehensive quantitative evaluation, we compute PSNR, together with perceptual quality metrics LPIPS (Zhang et al. 2018), SSIM (Wang et al. 2004), and ꟻLIP (Andersson et al. 2020). These metrics are computed across all texture sets in the dataset and all mipmap levels.

5.1 Quantitative Results

Experiments are conducted in 5 levels of compression ratios, namely around 0.05, 0.1, 0.2, 0.5 and 1.0 BPPC. Though we cannot guarantee identical compression ratios for all methods, we aim to align them approximately with the target BPPC. Tab. 3 showcases the comparison of different methods in terms of mean PSNR, SSIM, LPIPS and ꟻLIP over the entire dataset. Besides, we visualize the compression performance in Fig. 3, presenting the PSNR value vs. BPPC of different methods. A curve near the left-top corner indicates superior performance across different compression ratios.

Remarkably, utilizing the variable bitrates quantization scheme, NBC manages to further reduce the redundancy in compression representations, achieving an unprecedentedly low bitrate of 0.05 BPPC. In contrast, both conventional block-based texture compression methods BC and ASTC suffer from limited compression ratios and fail to achieve a low bitrate of 0.2 BPPC. Moreover, NBC enjoys flexibility in variable bitrates configuration and feature resolution adjustment, allowing for compression rates ranging from 0.05 to 1.0 BPPC, or even higher. At high bitrates, NBC significantly surpasses block compression methods under the same compression ratio. Meanwhile, at low bitrates, NBC outperforms the neural texture compression method NTC, state-of-the-art neural image compression method MLIC++ and high-quality image compression standards AVIF and JPEG XL. Particularly, at the aggressive compression ratio of 0.1 BPPC, the PSNR of NBC exceeds that of JPEG XL by 13%, and even outweighs AVIF and JPEG XL at 0.2 BPPC,

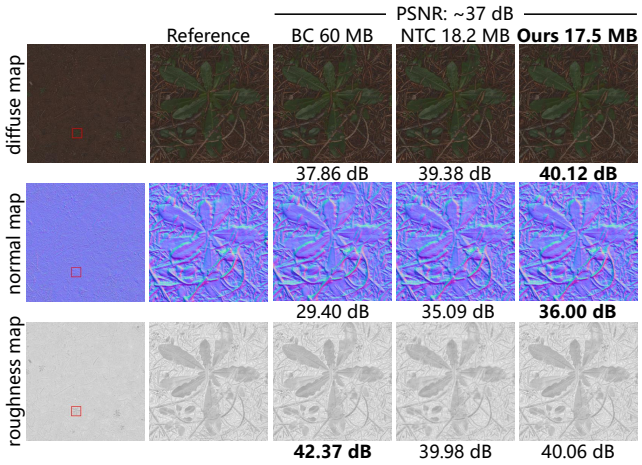

Figure 5: Iso-quality Comparison. We compare BC, NTC and NBC under comparable PSNR value conditions of around 37 dB. This comparison illustrates that NBC can achieve comparable performance with much lower bitrates.

indicating its satisfying performance at low bitrates.

5.2 Qualitative Results

The key to texture compression is the trade-off between storage and quality. Therefore, we conduct iso-storage and iso-quality comparisons, for qualitative evaluation.

We illustrate the iso-storage comparison in Fig. 4, where the compression ratio is restricted to 0.2 and 1.0 BPPC. Since BC could not achieve such a low BPPC, to conduct an iso-storage comparison, we bilinearly upsample its third mipmap to create mipmap level 0. NBC exhibits a pronounced superiority over BC in terms of PSNR and ꟻLIP, while maintaining a roughly equivalent storage footprint. Compared to the image compression technique JPEG XL, NBC enjoys significantly enhanced compression quality for normal maps, albeit with a marginally less optimal performance on diffuse textures.

In the iso-quality comparison shown in Fig. 5, where the PSNR score over the texture set is approximately 37 dB, NBC excels over BC with less than one-third of its storage requirement. Moreover, NBC demands 0.7 MB less storage than NTC, whilst delivering better compression quality.

To evaluate rendered quality, we showcase a rendered image of a crown using uncompressed textures, textures compressed with BC, NTC, and NBC in Fig. 6. Though we exclude the two highest mipmaps of BC for a fair comparison under matched storage conditions, it still demands 38% more storage. Requiring the least storage, NBC produces higher-quality rendered images than other methods.

5.3 Ablation Studies

Variable Bitrates Configuration. The variable bitrates configuration enjoys flexibility in the adjustment of quantization bitrates. Under a fixed feature dimension of 8, we conduct experiments over different bitrates configurations, as well as a full-precision configuration, to compare their compression

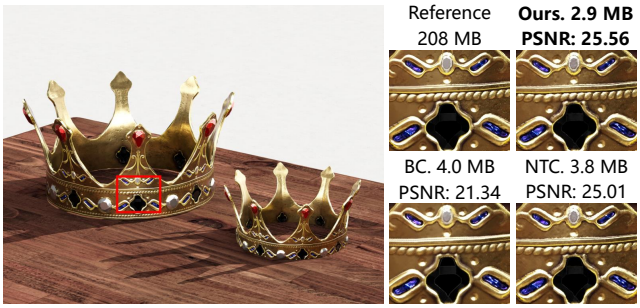

Figure 6: Rendered Image of a Crown. We present rendering quality using NBC, NTC and BC. NBC surpasses others in rendering quality while demanding the least storage.

Configuration	BPPC	PSNR ↑	SSIM ↑	LPIPS ↓	ꟻLIP ↓
(a) variable bitrates					
full-precision	2.252	35.09	0.910	0.082	0.062
8-4-4-4	0.286	34.78	0.916	0.084	0.061
8-4-4-2	0.180	33.86	0.899	0.099	0.070
8-4-2-2	0.154	33.19	0.809	0.110	0.077
8-4-2-1	0.101	33.02	0.891	0.123	0.081
2-2-2-2	0.141	32.36	0.880	0.130	0.086
(b) feature reuse					
w/ feature reuse	0.141	32.36	0.880	0.130	0.086
w/o feature reuse	0.176	32.78	0.889	0.124	0.083

Table 4: Ablation Studies. Ablation of (a) different variable bitrates configurations, and (b) feature reuse strategy under a fixed quantization rate of 2-bit.

qualities. As showcased in Tab. 4 (a), full-precision configuration outperforms our default 8-4-2-1 bitrates configuration by 2.07 dB but demands 22× more bits. More bitrates lead to better quality but more storage requirements.

Feature Reuse. As described in Sec. 4.1, for memory efficiency consideration, the lower resolution feature grid $\mathcal{Z}_b^\ell$ in feature mip-level ℓ leverages the feature reuse from the subsequent mip-level $\ell + 1$. Alternatively, the multi-resolution blocks hierarchy can be designed without the feature reuse strategy, meaning the lower resolution $\mathcal{Z}_b^\ell$ itself is stored within the multi-resolution blocks and is not shared with any other feature grids. Tab. 4 (b) performs a comparison under 2-bit quantization, and the design without feature reuse slightly boosts the PSNR score by 0.42 dB, however, it comes at the cost of requiring 25% more bits.

6 Conclusion

In this work, we take steps towards low-bitrate quantization in the realm of texture compression, extending the conventional block-based texture compression paradigm from color space to neural feature domain. Utilizing variable bitrates quantization on multi-resolution feature blocks, along with a residual fusion scheme, our NBC is capable of achieving texture compression at an unprecedentedly low bitrate, while maintaining an acceptable or comparable quality. Comprehensive experiments demonstrate the effectiveness of NBC. In the future, we will further broaden the scope of application for variable bitrates quantization.

Acknowledgements

This work was supported by National Science Foundation of China (U20B2072,61976137). This work was also partially supported by Grant YG2021ZD18 from Shanghai Jiao Tong University Medical Engineering Cross Research. This work was partially supported by STCSM22DZ2229005.

References

Alakuijala, J.; Van Asseldonk, R.; Boukortt, S.; Bruse, M.; Comșa, I.-M.; Firsching, M.; Fischbacher, T.; Kliuchnikov, E.; Gomez, S.; Obryk, R.; et al. 2019. JPEG XL next-generation image compression architecture and coding tools. In *Applications of Digital Image Processing XLII*, volume 11137, 112–124.

Andersson, P.; Nilsson, J.; Akenine-Möller, T.; Oskarsson, M.; Åström, K.; and Fairchild, M. D. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2).

Ballé, J.; et al. 2017. End-to-end Optimized Image Compression. In *International Conference on Learning Representations*.

Beers, A. C.; et al. 1996. Rendering from Compressed Textures. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 373–378.

Chen, T.; Liu, H.; Ma, Z.; Shen, Q.; Cao, X.; and Wang, Y. 2021. End-to-End Learnt Image Compression via Non-Local Attention Optimization and Improved Context Modeling. *IEEE Transactions on Image Processing*, 30: 3179–3191.

Chen, Y.; Murherjee, D.; Han, J.; Grange, A.; Xu, Y.; Liu, Z.; Parker, S.; Chen, C.; Su, H.; Joshi, U.; et al. 2018. An overview of core coding tools in the AV1 video codec. In *2018 picture coding symposium (PCS)*, 41–45. IEEE.

Cheng, Z.; Sun, H.; Takeuchi, M.; and Katto, J. 2020. Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules. In *Conference on Computer Vision and Pattern Recognition*.

Choi, Y.; El-Khamy, M.; and Lee, J. 2019. Variable Rate Deep Image Compression With a Conditional Autoencoder. In *International Conference on Computer Vision*, 3146–3154.

Cui, Z.; Wang, J.; Gao, S.; Guo, T.; Feng, Y.; and Bai, B. 2021. Asymmetric gained deep image compression with continuous rate adaptation. In *Conference on Computer Vision and Pattern Recognition*, 10532–10541.

Davies, T.; et al. 2021. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes. In *International Conference on Machine Learning*.

Delp, E.; and Mitchell, O. 1979. Image Compression Using Block Truncation Coding. *IEEE Transactions on Communications*, 27(9): 1335–1342.

Dupont, E.; Goliński, A.; Alizadeh, M.; Teh, Y. W.; and Doucet, A. 2021. COIN: COmpression with Implicit Neural representations. *arXiv preprint arXiv:2103.03123*.

Fränti, P.; et al. 1994. Compression of Digital Images by Block Truncation Coding: A Survey. *The Computer Journal*, 37(4): 308–332.

Girish, S.; et al. 2023. SHACIRA: Scalable HAsh-grid Compression for Implicit Neural Representations. In *International Conference on Computer Vision*, 17513–17524.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*.

Hu, Y.; Yang, W.; and Liu, J. 2020. Coarse-to-Fine Hyper-Prior Modeling for Learned Image Compression. *AAAI Conference on Artificial Intelligence*, 34(07): 11013–11020.

Iourcha, K.; et al. 1997. System and method for fixed-rate block-based image compression with inferred pixel values.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A. G.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Conference on Computer Vision and Pattern Recognition*.

Jiang, W.; and Wang, R. 2023. MLIC++: Linear Complexity Multi-Reference Entropy Modeling for Learned Image Compression. *arXiv preprint arXiv:2307.15421*.

Johnston, N.; Vincent, D.; Minnen, D.; Covell, M.; Singh, S.; Chinen, T.; Jin Hwang, S.; Shor, J.; and Toderici, G. 2018. Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks. In *Conference on Computer Vision and Pattern Recognition*, 4385–4393.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knittel, G.; Schilling, A.; Kugler, A.; and Straßer, W. 1995. Hardware for Superior Texture Performance. In *Proceedings of the Tenth Eurographics Conference on Graphics Hardware*, 33–40.

Li, M.; Zuo, W.; Gu, S.; Zhao, D.; and Zhang, D. 2018. Learning Convolutional Networks for Content-Weighted Image Compression. In *Conference on Computer Vision and Pattern Recognition*, 3214–3223.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*.

Lu, Y.; Jiang, K.; Levine, J. A.; and Berger, M. 2021. Compressive Neural Representations of Volumetric Scalar Fields. *Computer Graphics Forum*, 40(3): 135–146.

Microsoft Corporation. 2020. Texture Block Compression in Direct3D 11.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Communications of the ACM*, 65(1): 99–106.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics*, 41(4).

Müller, T.; Rousselle, F.; Novák, J.; and Keller, A. 2021. Real-Time Neural Radiance Caching for Path Tracing. *ACM Transactions on Graphics*, 40(4).

Nystad, J.; Lassen, A.; Pomianowski, A.; Ellis, S.; and Olson, T. 2012. Adaptive Scalable Texture Compression. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, 105–114.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; and Lovegrove, S. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*.

Rippel, O.; et al. 2017. Real-Time Adaptive Image Compression. In *International Conference on Machine Learning*, volume 70, 2922–2930. PMLR.

Ström, J.; et al. 2004. PACKMAN: Texture Compression for Mobile Phones. In *ACM SIGGRAPH 2004 Sketches*, 66.

Ström, J.; et al. 2005. IPACKMAN: High-Quality, Low-Complexity Texture Compression for Mobile Phones. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 63–70.

Ström, J.; et al. 2007. ETC2: Texture Compression Using Invalid Combinations. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, 49–54.

Strümpler, Y.; Postels, J.; Yang, R.; Gool, L. V.; and Tombari, F. 2022. Implicit Neural Representations for Image Compression. In *European Conference on Computer Vision*.

Takikawa, T.; Evans, A.; Tremblay, J.; Müller, T.; McGuire, M.; Jacobson, A.; and Fidler, S. 2022. Variable Bitrate Neural Fields. In *ACM SIGGRAPH 2022 Conference Proceedings*.

Takikawa, T.; et al. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes.

Theis, L.; Shi, W.; Cunningham, A.; and Huszár, F. 2017. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations*.

Toderici, G.; O'Malley, S. M.; Hwang, S. J.; Vincent, D.; Minnen, D.; Baluja, S.; Covell, M.; and Sukthankar, R. 2016. Variable Rate Image Compression with Recurrent Neural Networks. In *International Conference on Learning Representations*.

Toderici, G.; Vincent, D.; Johnston, N.; Hwang, S. J.; Minnen, D.; Shor, J.; and Covell, M. 2017. Full Resolution Image Compression with Recurrent Neural Networks. In *Conference on Computer Vision and Pattern Recognition*.

Torborg, J.; and Kajiya, J. T. 1996. Talisman: Commodity Realtime 3D Graphics for the PC. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 353–363.

Vaidyanathan, K.; Salvi, M.; Wronski, B.; Akenine-Moller, T.; Ebelin, P.; and Lefohn, A. 2023. Random-Access Neural Compression of Material Textures. *ACM Transactions on Graphics*, 42(4).

Wang, Z.; Bovik, A.; Sheikh, H.; and Simoncelli, E. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612.

Yang, F.; Herranz, L.; Weijer, J. v. d.; Guitián, J. A. I.; López, A. M.; and Mozerov, M. G. 2020. Variable Rate Deep Image Compression With Modulated Autoencoder. *IEEE Signal Processing Letters*, 27: 331–335.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Conference on Computer Vision and Pattern Recognition*.

Zhong, Z.; et al. 2020. Channel-Level Variable Quantization Network for Deep Image Compression. In *International Joint Conference on Artificial Intelligence*.