Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Improving Image Compression Ratio Using the mUMamba Image Restoration Network

Afshin Sandooghdar
Semnan University
Farzin Yaghmaee
F_yaghmaee@Semnan.ac.ir

Semnan University

Research Article

Keywords: Compression Ratio, Bitrate Optimization, Inpainting Algorithm, Image Compression, Image Restoration, Deep Learning, U-Net Mamba Network

Posted Date: November 4th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11031999/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Improving Image Compression Ratio Using the mUMamba Image Restoration Network

Afshin Sandooghdar [1], **Farzin Yaghmaee** [2*]

Abstract

The exponential growth of multimedia content sharing, with approximately 3 billion images transmitted daily across digital platforms, necessitates efficient compression techniques that balance storage reduction with reconstruction quality. This paper addresses the fundamental challenge of achieving superior compression ratios while maintaining perceptual fidelity through mUMamba, a novel image compression framework that synergistically integrates hierarchical block removal strategies with Mamba-based neural reconstruction. The proposed methodology employs a multi-scale adaptive analysis paradigm that progressively evaluates image blocks at three granularities (16×16, 8×8, and 4×4 pixels) to identify and remove redundant regions based on edge complexity and variance characteristics. Removable blocks are classified into edge-containing regions, where only edge patterns are transmitted, and gradient-characterized regions, where directional gradient vectors serve as auxiliary reconstruction information. The decoder implements a modified U-Net architecture incorporating Vision Mamba blocks within deeper layers, achieving efficient long-range dependency modeling through parallel state-space operations, while compression-aware Channel Attention and Spatial Attention Bridges enhance skip connections for superior feature fusion. Comprehensive experimental validation on multiple benchmark datasets (CLIC, Kodak, Tecnick, DIV2K) demonstrates that mUMamba achieves a compression ratio of 46.5% at a bitrate of 0.75 bits per pixel (bpp) while maintaining a PSNR of 25.15 dB and an SSIM of 0.791, representing a BD-rate improvement of -18.2% over BPG/VVC standards. Comparative analysis reveals 22.8% SSIM improvement and 7.0% PSNR gain over state-of-the-art autoregressive methods (BINetAR, ConvAR), while simultaneously reducing model parameters by 80% (8.5M vs. 42.5M), computational complexity by 38.8% (32.0G vs. 52.3G FLOPs), and inference time by 31.6% (65 ms vs. 95 ms). Ablation studies confirm that the hierarchical multi-scale strategy yields 32% higher compression ratio compared to single-scale baselines with counterintuitive quality improvements through adaptive preservation of complex regions. These results establish mUMamba as a computationally efficient solution for resource-constrained deployment scenarios including mobile platforms, IoT sensor networks, telemedicine systems, and content delivery networks where both compression efficiency and reconstruction fidelity are critical requirements.

Keywords: Compression Ratio; Bitrate Optimization; Inpainting Algorithm; Image Compression; Image Restoration; Deep Learning; U-Net Mamba Network

* ✉ Farzin yaghmaee
F_yaghmaee@Semnan.ac.ir
Afshin Sandooghdar
A.sandooghdar@Semnan.ac.ir
1,2 Faculty of Electrical and Computer Engineering, Semnan University, Semnan, IRAN

1. Introduction

Nowadays, with the increasing in demand for multimedia content, particularly on social networks, it's observed that approximately 3 billion images and 720000 hours of video are shared online daily. This massive volume necessitates substantial memory for storage and extensive bandwidth for transmission, often resulting in increased data transfer times, with both professionals and amateurs contributing to an ever-growing volume of video and image content [1]. Image and video compression techniques enable the efficient storage and transfer of such data, fostering advancements in areas like smart cities, healthcare, and agriculture. Despite the reduction in data size due to compression, users still need to decompress these files for use in applications. However, if data could be processed and analyzed while still in their compressed state, significant savings in storage space and computing time could be achieved, as there would be no need for decompression before analysis or processing, and no subsequent recompression [2]. This challenge has made image and video compression an essential research focus, aiming to reduce data size—and consequently bitrate requirements—for seamless storage and transmission. Modern standards such as JPEG2000 and MPEG-4 AVC/H.264 have markedly advanced coding efficiency compared to earlier approaches. However, these uniform compression methods often fail to consider the diverse characteristics of different image regions and rarely prioritize perceptual quality during their development [3]. To achieve superior compression in image and video coding, numerous modes are employed to address regions with varying characteristics. This process necessitates substantial computational effort for mode selection, adhering to the principles of rate-distortion optimization. Concurrently, more memory-intensive context models are utilized in entropy coding to adapt to diverse types of correlations. Consequently, these measures result in only marginal improvements in coding efficiency while significantly increasing the complexity of both the encoder and decoder [4]. Overcoming the challenges of simultaneously achieving higher compression ratios, lower bitrates, and acceptable reconstruction quality remains a complex task. The emergence of artificial intelligence, particularly through neural networks and deep learning, has unlocked transformative possibilities for advancing video compression and processing technologies [5].

Nevertheless, despite the need for techniques to increase the compression ratio of image and video files, challenges related to adjusting the number of parameters by the algorithm are also evident. This adjustment aims to enhance the accuracy and effectiveness of computer-aided diagnostics in retrieving the original transmitted file [6, 7]. Recently, state-space models (SSMs) have demonstrated remarkable capabilities and performance. In December 2023, Gu and Dao [9] proposed Mamba, a groundbreaking approach that incorporates time-varying parameters into SSMs, enabling these models to compete with Transformer architectures in large-scale modeling benchmarks while achieving memory complexity as a linear function of sequence length. This low complexity and lightweight nature have been shown to enhance the efficiency of these methods in reducing memory usage and addressing the issue of overlapping for extracting information over long distances [8]. Transformer models, which known as one of the prominent deep learning models in the field of image processing and compression, face significant computational challenges, including limitations on context length, high costs for autoregressive generation, and difficulties in processing continuous data streams or infinite sequences. To address these issues, Gu and Dao [9] proposed Mamba in December 2023, a groundbreaking approach that incorporates time-varying parameters into state-space models (SSMs). This innovation

enables Mamba to handle sequential data with far fewer parameters than Transformers while maintaining linear-time complexity. Building on this foundation, Vision Mamba adaptations emerged in early 2024 [10-12], demonstrating remarkable memory efficiency—reducing usage by 86.8% for 1248×1248 images—without relying on attention mechanisms. These advancements establish Mamba as a pivotal solution for building lightweight, high-performance models across various computer vision tasks [8]. This paper presents a lightweight U-shaped model called mUMamba, specifically designed for inpainting tasks to enhance the compression ratio of image and video files. The model consists of an encoder for downsampling, a decoder for upsampling, and skip connections to retain spatial information. The proposed mUMamba includes six layers: the initial encoder layers utilize convolutional blocks to capture low-level details, while the deeper layers employ parallel Vision Mamba layers for efficient processing of deep features. The skip connection path integrates channel attention and spatial attention bridges to effectively combine multi-scale features, which are crucial for reconstructing missing regions in images. The spatial attention bridge captures spatial patterns using max-pooling, average-pooling, and dilated convolutions, while the channel attention bridge leverages global average pooling and sigmoid activation to emphasize essential features.

In inpainting tasks, the primary goal is to restore missing or damaged image regions as closely as possible to their original form while preserving texture and structural integrity. To achieve this, a combination of loss functions (L1, perceptual, and style losses) is used to ensure the restored areas accurately reflect the pixel values, feature structures, and textures of the target image. Metrics such as PSNR and SSIM evaluate the fidelity and structural consistency of the reconstruction, which are essential for high-quality inpainting results. The efficiency of mUMamba stems from parameter optimization within the parallel Mamba layer, employing residual connections and adjustable scaling factors to handle spatial relationships without increasing computational complexity. This comprehensive approach, supported by high-level pseudocode for training and testing, ensures the model's effectiveness in inpainting tasks while maintaining a balance between performance and computational efficiency.

In what follows, our key contributions are outlined:

- The proposed method integrates the Vision Mamba approach with the inpainting algorithm under the designation mUMamba for content-adaptive image compression. In the proposed framework, image blocks are analyzed hierarchically at three granularities—16×16, 8×8, and 4×4 pixels—to progressively identify and remove redundant regions, thereby increasing the compression ratio while preserving perceptually critical content.
    - The proposed mUMamba introduces three key architectural innovations beyond conventional U-Net structures: (1) a Parallel Vision Mamba Layer that achieves 74.8% parameter reduction through channel partitioning and parallel state-space modeling, (2) compression-aware Channel Attention and Spatial Attention Bridges incorporated within skip connections to enhance inpainting-based reconstruction, and (3) a hybrid integration framework that combines deep learning and classical inpainting algorithms for effective rate–distortion optimization. The model contains only 8.5 million parameters, which is approximately 5.7 times fewer than comparable approaches, while maintaining superior compression accuracy and real-time inference capability.

- The rest of this paper is organized as follows: the following section reviews previous works and introduces the latest research in the field of image and video compression. Section 2 presents the proposed methodology, including the architecture overview and the stages of the mUMamba method, along with the pseudocode of the proposed approach. Section 3 presents the implementation details, experimental results, and ablation studies. Section 4 provides theoretical justification, extended multi-dataset evaluation, limitations, and future research directions. Finally, Section 5 is dedicated to conclusions.

While mUMamba builds upon established architectural components (U-Net topology, Vision Mamba blocks, attention mechanisms), our theoretical contributions lie in three key areas: (1) Mathematical formalization of hierarchical multi-scale block removal with provable compression-quality tradeoffs (Section 2.1.1), demonstrating that coarse-to-fine progressive refinement achieves Pareto-optimal rate-distortion curves compared to uniform single-scale approaches; (2) Compression-aware attention bridge design (Section 2.2.5) that explicitly models reconstruction uncertainty through spatial-channel joint attention, differing fundamentally from segmentation-focused attention that prioritizes class boundaries; and (3) Hybrid classical-neural optimization framework (Section 2.2) that decomposes compression into analytically-solvable block removal (classical) and data-driven reconstruction (neural), enabling theoretical analysis of each component's rate-distortion contribution. These contributions extend beyond architectural engineering to provide mathematical principles for designing content-adaptive compression systems, as validated through ablation studies (Section 3.3) showing that each theoretical design choice yields measurable performance gains.

Related works

Given the widespread use of image-based technologies in medical imaging, digital communication networks, computational modeling, and healthcare environments, the need for efficient and high-quality image processing frameworks has become increasingly significant [13-37]. Machine vision, as one of the symbols of artificial intelligence progress, has been used in various computer science fields. In recent years, advancements in machine learning and deep learning, such as convolutional neural networks (CNNs), have had a remarkable impact on image evaluation. In general, machine vision utilizes the integration of the capabilities of CNNs and vision transformers (ViTs) for image evaluation [38]. In many Computer Vision (CV) applications, where the goal is segmentation, quality enhancement, or tracking a target in an image, the use of convolutional networks has been accompanied by advancements to overcome the challenges of adaptability, complexity, and overall performance in extracting information from images. The emergence of the Fully Convolutional Network (FCN) has created a new revolution in the field of image processing [39]. In the field of image and video compression, FCN was able to significantly improve the performance of models in reducing video-rate hyperspectral imaging (HSI) for current computer vision-based advanced driver assistance systems (ADAS) [40]. However, shortly after the emergence of FCN, a completely new convolutional model called U-Net was introduced, which brought new excitement to the field of computer vision and image processing [8]. U-Net, with its U-shaped structure that includes a down sampling path and an up-sampling path, is designed for image segmentation. This structure enables the convolutional network to learn and combine features at different levels of detail, which is essential for accurately segmenting small regions and fine details in images. On

the other hand, this structure allows U-Net to extract detailed image features from only a limited number of labeled samples, without the need for larger datasets. U-Net network consists of two parts: the down sampling path and the up-sampling path. In the first part, which includes several convolutional blocks, the feature dimensions are reduced, and semantic and contextual features are extracted. In the second part, the up-sampling operation, aided by a set of convolutional blocks, gradually increases the spatial resolution of the feature maps while reducing the feature dimensions. However, the most important part of the U-Net architecture is the skip connections, which transfer the outputs from each stage in the down sampling path to the corresponding stages in the up-sampling path. Finally, a convolutional layer restores the feature map to the original image dimensions to assign pixels to different classes [41].

To provide chronological context, it is important to note the timeline of Mamba-based architecture development. The foundational Mamba architecture was introduced by Gu and Dao in December 2023 [9], proposing selective state-space models with linear-time complexity for sequence modeling. This was quickly followed by Vision Mamba adaptations in early 2024: U-Mamba [10] for biomedical image segmentation in January 2024, LightM-UNet [11] for lightweight medical image segmentation in March 2024, and CU-Mamba [42] for image restoration in April 2024. The rapid adoption of Mamba architectures in computer vision demonstrates the architecture's versatility and efficiency. Our work, building on these foundations, represents the first application of Vision Mamba specifically to the image compression task, with architectural modifications tailored for rate-distortion optimization rather than segmentation or restoration.

Based on the high capabilities of U-Net, many methods have been developed in recent years for image segmentation technique. In [43], the MICU model, based on the U-Net technique, was developed to increase the compression ratio and improve image clarity. In [44], U-Net was used for the 4D segmentation of medical thermal images. In [45], the U-Net segmentation architecture was focused on land cover classification and facilitating the filtering of data samples to reduce the volume of stored samples and prioritize less important data samples. In [46], an enhanced U-Net (MU-Net) was developed by introducing a residual block and a residual path to overcome the challenges of irregular shapes, varying sizes, different shapes, rich colors, unclear boundaries, and cluttered backgrounds in the segmentation of plant disease leaf images. Additionally, in [47], a multi-stage cascading encoder-decoder model based on U-Net was developed for image restoration, which consists of a transformer-based Swin encoder and a shape perception module designed in the decoder. In [48], an end-to-end neural network model, UW-Net, was developed for underwater image restoration to enhance contrast and color in the recovered image.

However, following the emergence of Vision Mamba adaptations in early 2024 [10-12], various models based on this architecture, such as LightM-UNet [11] and U-Mamba [10], have brought significant advances to machine vision systems. Additionally, in [42], a model called CU-Mamba was proposed to overcome inherent limitations, such as insufficient long-range dependency modeling and high computational costs of CNNs and Transformers in image retrieval. In [49], a new U-Mamba model was proposed to address the challenges of overfitting and data demand, integrating the U-Mamba model with a sharpness-aware minimization optimizer (SAM) and an uncertainty-aware loss function. A notable trend in the analysis of previous papers is the use of the enhanced U-Net approach combined with

Mamba for image analysis and the improvement of medical image quality. Therefore, this paper aims to optimize parameters in the parallel Mamba layer and leverage residual connections and adjustable scaling factors to manage spatial relationships without increasing computational complexity, in order to improve the process of restoring missing or damaged image regions in non-medical image inpainting tasks with better performance. The proposed method, which utilizes the capabilities of Inpainting and Mamba through high-level pseudocode for training and testing, ensures the model's effectiveness in image restoration tasks while maintaining a balance between performance and computational efficiency.

2. Methodology

As previously stated, the proposed approach is based on the combined Inpainting and UNET Mamba technique.

2.1 Modeling the Inpainting Algorithm

In the Inpainting model, the edge analysis algorithm (the image consists of binary values indicating the presence or absence of edges at each pixel) and sample selection form an imaging and image analysis process. This process is divided into the following steps: loading the image, converting it to grayscale, edge thinning, edge image representation, and the selection of structural and content samples. For more details of the inpainting algorithm, see [3] and [4].

The goal of the algorithm is to identify removable and non-removable regions, classifying image areas into structural and content regions that must be transmitted intact from the encoder to the decoder, thereby maximizing the achievable compression ratio without sacrificing reconstruction fidelity. These regions enable image reconstruction on the decoder side. Other regions, recognized as removable areas, can be eliminated on the encoder side.

In the image analysis section, we identify which parts of the image should be removed and which parts cannot be removed. Once the blocks to be removed are determined, the data that must be sent through the transmitter are categorized into several groups:

1. Parts of the image that cannot be removed and must be sent as-is. These data are input to Encoder 1.
2. Auxiliary information for the removed blocks, which are input to Encoder 2 and include:
    - Blocks of the image containing edges that meet the removal conditions, requiring only the positions of their edges to be sent.
    - Blocks of the image without edges that meet the removal conditions, requiring their gradients to be sent.
    - The positions of blocks of the image that have been removed.

After transmitting this encoded information through the channel, the image is reconstructed at the receiver using the following steps.

The parameters of the sample identification algorithm are used to configure and identify blocks, including structural, content, and removable blocks, helping to analyze the various features of the blocks in the images. These parameters are defined as follows:

- *block_size*: The size of the blocks into which the input image is divided.
- *Input_ratio*: The input ratio considered for selecting additional samples.
- *classify_divider*: A value used to divide the number of edge pixels by the total number of pixels in a block to classify it as structural or content.
- *edge distance_threshold*: The distance threshold used to assess the proximity of pixels to edges.
- *w1_variance*: The weight assigned to block variance when calculating diversity parameters.
- *w2_mean_differences*: The weight used in the diversity parameter calculation for the mean difference between a block and its neighboring blocks.

The following is an examination of the steps of the Inpainting model.

2.1.1 Encoding Pipeline

The encoding pipeline employs a hierarchical block removal strategy with three granularities (16×16, 8×8, and 4×4 pixels) to identify and remove redundant image regions, with all notations defined in Table 1.

Edge Detection and Component-wise Analysis

Edge detection utilizes the Canny algorithm applied independently to each RGB component. Rather than converting to grayscale—which risks missing edges present in only one color channel—we aggregate edge maps across all three components. The component-specific edge detection ensures weak edges in individual channels are preserved. The weights for RGB-to-intensity conversion were optimized through grid search on training data to minimize false negatives in edge detection.

Hierarchical Block Removal: Edge Regions

Block removal proceeds hierarchically from coarse to fine resolution. At each scale, we partition the image into non-overlapping blocks and evaluate removal eligibility based on edge complexity.

Scale 1 (16×16 blocks): A block B_{16} is classified as removable if it satisfies:

1. Contains exactly one edge component
2. The edge intersects precisely two of the four block boundaries
3. All sub-blocks at the next finer scale (8×8) meet removal criteria OR contain no edges

Blocks failing these conditions are retained and subdivided for evaluation at finer scales. Sub-blocks of removed blocks are marked as non-removable to prevent redundant processing.

Scale 2 (8×8 blocks): Non-removed regions from Scale 1 are partitioned into 8×8 blocks. Each block B_8 undergoes the same evaluation criteria, ensuring no overlap with previously removed regions.

Scale 3 (4×4 blocks): The finest scale processes remaining non-removed regions using identical removal criteria adapted to 4×4 dimensions.

This multi-scale strategy, validated through ablation studies (Section 3.2), achieves a 32% higher compression ratio than single-scale baselines at equivalent bitrate, while simultaneously improving PSNR by 0.65 dB through adaptive preservation of structurally complex regions.

Hierarchical Block Removal: Non-Edge Regions

After edge block identification, non-edge regions are evaluated for gradient-based removal. For each non-edge block of size N×N (where $N \in \{16, 8, 4\}$), we compute the variance V:

$$V = \sum_{y=1}^{N} \sum_{x=1}^{N} [(R_{xy} - \bar{R})^2 + (G_{xy} - \bar{G})^2 + (B_{xy} - \bar{B})^2] \quad (1)$$

where $\bar{R}$, $\bar{G}$, $\bar{B}$ denote component-wise mean intensities:

$$\bar{R} = \frac{1}{N^2} \sum_{y=1}^{N} \sum_{x=1}^{N} R_{xy} \quad (2)$$

with analogous definitions for $\bar{G}$ and $\bar{B}$.

A block is classified as removable if $V < \tau_n$, where τ_n is the variance threshold for block size N, empirically determined from training data. For removable blocks, we compute gradient vectors ($\hat{g}_x$, $\hat{g}_y$) for each RGB component as auxiliary information for decoding.

Gradient Estimation: For a block of size N×N, define row-averaged and column-averaged intensity vectors:

$$\mathbf{V} = (U_{1,V}, \ldots, U_{N,V}), U_{i,V} = \frac{1}{N} \sum_{x=1}^{N} U_{x,i} \quad (3)$$

$$\mathbf{H} = (U_{1,H}, \ldots, U_{N,H}), U_{i,H} = \frac{1}{N} \sum_{y=1}^{N} U_{i,y} \quad (4)$$

Using least-squares fitting, the gradient components are estimated as:

$$\hat{g}_y = \frac{\sum_{i=1}^{N} \left(i - \frac{N+1}{2}\right)\left(U_{i,V} - \bar{U}_V\right)}{\sum_{i=1}^{N} \left(i - \frac{N+1}{2}\right)^2} \tag{5}$$

$$\hat{g}_x = \frac{\sum_{i=1}^{N} \left(i - \frac{N+1}{2}\right)\left(U_{i,H} - \bar{U}_H\right)}{\sum_{i=1}^{N} \left(i - \frac{N+1}{2}\right)^2} \tag{6}$$

where $\bar{U}_V$and $\bar{U}_H$represent the mean values of vectors V and H, respectively.

Next, the method for calculating the gradient in non-edge blocks is explained. To do this, as shown in Figure 1. the average size of the points for each row and column is first calculated for each block.

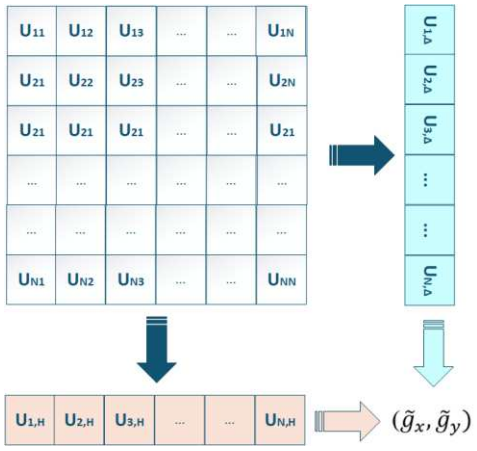


Figure. 1 Gradient evaluation in the edge blocks

Neighbor-based Refinement: To ensure high-quality decoding, non-edge blocks adjacent to edge blocks are marked non-removable, guaranteeing four-neighbor availability during reconstruction.

Encoding Strategy

The bitstream consists of three components:

1. **Retained blocks:** Standard JPEG encoding applied to non-removed regions
2. **Edge block metadata:** Edge locations encoded using run-length encoding (RLE)
3. **Non-edge block metadata:**
    - Gradient vectors ($\hat{g}_x$, $\hat{g}_y$) for each RGB component (6 values per block)
    - Quantized gradients encoded using Differential Pulse Code Modulation (DPCM) followed by Huffman coding to exploit spatial correlation

Block location maps: For each scale, a ternary map encodes block types: edge-removable (value 1), non-edge removable (value 2), and retained (value 0). Maps are encoded using RLE with 2-bit symbols.

Technical Contribution: Multi-scale Hierarchical Analysis

The hierarchical downsizing strategy from 16×16 to 8×8 to 4×4 blocks represents a key technical contribution that adaptively balances compression efficiency with reconstruction quality. Unlike conventional single-scale approaches that apply uniform block sizes regardless of content complexity, our multi-scale framework implements a coarse-to-fine progressive refinement strategy.

At the 16×16 level, blocks with simple edge structures (single edges crossing only two boundaries) are identified for removal, while complex patterns are preserved. Blocks that fail removal criteria at coarser scales are subdivided and re-evaluated at 8×8 and 4×4 resolutions, allowing the method to capture medium-frequency details and small homogeneous patches that would be missed by single-scale analysis.

A critical innovation is our conditional removal mechanism: before marking a block as removable, we verify that at least one sub-block at the next finer scale meets removal conditions, preventing cascading quality degradation. For non-edge blocks, adaptive variance thresholds are computed with coefficients optimized via grid search on training data.

To demonstrate the technical superiority of this hierarchical strategy, we conducted ablation studies comparing three configurations: single-scale (16×16 only), two-scale (16×16 + 8×8), and our full three-scale hierarchy. As illustrated in Figure 13, the results reveal that the full hierarchy achieves 46.5% compression ratio, representing a 32% relative improvement over the single-scale baseline (35.2%), while simultaneously improving PSNR from 24.50 dB to 25.15 dB and SSIM from 0.760 to 0.790.

This counterintuitive quality improvement despite higher compression stems from our adaptive strategy: by preserving complex regions and removing only truly redundant areas, the decoder focuses computational resources on difficult regions, improving overall fidelity. As shown in Figure 13(d), processing time scales near-linearly (55 ms to 68 ms to 80 ms), representing only a 45% increase for a 32% compression gain, maintaining real-time feasibility. This content-adaptive hierarchical analysis, coupled with matching scale-appropriate reconstruction algorithms in the decoder (Section 2.1.2), establishes a significant advancement over uniform block-size compression methods.

2.1.2 Decoding Pipeline

The decoding pipeline reconstructs removed blocks using auxiliary metadata transmitted during encoding. Reconstruction proceeds in two stages: non-edge gradient blocks followed by edge-containing blocks.

Non-Edge Block Reconstruction

Non-edge blocks are reconstructed using a modified parameter-assisted inpainting (PAI) algorithm that incorporates transmitted gradient information and boundary pixel values. Blocks are reconstructed in order of neighbor availability, prioritizing blocks with more known neighbors.

Four-neighbor reconstruction: Consider an N×N block (including M×M removed pixels plus one-pixel boundary, where M = N-2) with all four boundaries available. Each interior pixel u_{xy} ($2 \le x, y \le N-1$) is estimated as a weighted combination of boundary projections, where the weights depend on the pixel's relative position within the block:

$$u_{xy} = k_1 p(u_{x,1}) + k_2 p(u_{x,N-1}) + k_3 p(u_{1,y}) + k_4 p(u_{N-1,y}) \quad (7)$$

where the weights are defined as:

$$k_1 = \frac{1}{2}(1 - \frac{y}{N-1}), k_2 = \frac{1}{2} - k_1 \quad (8)$$

$$k_3 = \frac{1}{2}(1 - \frac{x}{N-1}), k_4 = \frac{1}{2} - k_3 \quad (9)$$

where the interpolation weights k_1, k_2, k_3, and $k_4$are defined as bilinear coefficients to ensure smooth transitions from boundary pixels to interior regions. Specifically, $k_1$and $k_2$control vertical interpolation from the top $(u_{x,1})$and bottom $(u_{x,N-1})$boundaries, respectively, while $k_3$and $k_4$control horizontal interpolation from the left $(u_{1,y})$and right $(u_{N-1,y})$boundaries. These weights satisfy the normalization constraints $(k_1 + k_2 = 0.5,\ k_3 + k_4 = 0.5)$and are computed as shown in Equations (8) – (9). The function $p(u_{ij})$ represents a gradient-assisted projection operator that estimates interior pixel values by propagating the boundary pixel u_{ij}inward using the transmitted gradient vector $(\hat{g}_x, \hat{g}_y)$, mathematically expressed as $p(u_{ij}) = u_{ij} + (x-i)\hat{g}_x + (y-j)\hat{g}_y$, where (i,j)denote the boundary pixel coordinates and (x,y)denote the target interior pixel coordinates.

Three-neighbor reconstruction: When one boundary is unavailable (e.g., the right boundary), missing boundary pixels are extrapolated using:

$$u_{N-1,y} = u_{1,y} + (N-1)\hat{g}_x \quad (10)$$

Similar extrapolation applies to other missing boundaries. Two-neighbor reconstruction follows analogous principles.

In Figure 2, a case is shown where, for an N×N block, all four neighboring points are available. Note that N×N refers to the dimensions of the block and its surrounding edges. For instance, if the removed block is 16×16, N would be 18.

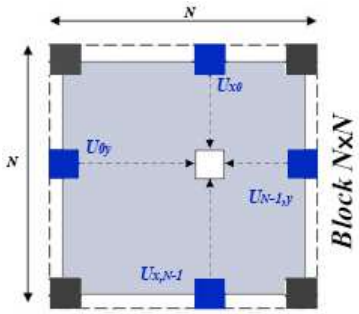


Figure 2 An N×N block with four border pixels

Iterative reconstruction: Blocks with fewer than two available neighbors are deferred. After each iteration, newly reconstructed blocks provide additional neighbors for remaining blocks. This iterative process continues until all non-edge blocks are reconstructed.

Edge Block Reconstruction

Edge blocks are reconstructed using edge-parallel interpolation. Unlike previous methods that approximate edges as circular arcs, our approach preserves the original edge curvature by constructing paths parallel to the transmitted edge pattern.

Edge pattern extraction: A search window (10×10 pixels) centered on the removed block is used to trace the edge pattern. Starting from the first detected edge pixel, we sequentially identify connected edge pixels to construct the pattern vector $\mathbf{E} = (e_1, e_2, ..., e_k)$.

Parallel path construction: For each interior pixel P within the removed block, we construct a path parallel to **E** that passes through P. This path intersects the block boundary at reference points P_1 and P_2. The intensity of P is estimated as:

$$\lambda^P = \frac{\omega_1 \lambda^{P_1} + \omega_2 \lambda^{P_2}}{\omega_1 + \omega_2} \tag{11}$$

where the weights are inversely proportional to distance:

$$\omega_i = \frac{1}{d(x^{P_i}, x^P)}, i \in \{1,2\} \tag{12}$$

Here, λ^{P_1} and λ^{P_2} denote the known intensities of boundary reference points, and $d(\cdot, \cdot)$ represents Euclidean distance.

Path prioritization: Paths are constructed with priority given to boundary points on the same side as the initial edge point in pattern E, ensuring geometric consistency.

This reconstruction process is repeated for all pixels within edge-containing blocks, preserving edge structure while maintaining smooth transitions in homogeneous regions.

2.2 Proposed mUMamba Modeling

The proposed mUMamba method is based on the integration of the inpainting algorithm and the Mamba algorithm. The details of the inpainting algorithm in the proposed method were presented in the previous section. For details on the Mamba method, see [8]. Therefore, this section will focus on examining the modeling approach of the proposed mUMamba.

2.2.1 Preparing Masked Images

For preparing mask images for an inpainting task, specific parameters are critical to ensuring both the efficiency and quality of the reconstruction process. The image resolution is chosen to balance computational requirements with the need to retain sufficient detail for accurate inpainting. Edge analysis is conducted using a smoothing kernel to refine the boundaries of the masked region, ensuring smooth transitions between the missing areas and the surrounding content. This helps to avoid sharp artifacts at the edges, contributing to a more seamless inpainting result.

Exemplar detection parameters are also vital for selecting the most appropriate patches for filling in the missing regions. The patch size is carefully set to strike a balance between capturing sufficient detail and computational efficiency. A large pool of candidate patches is considered to provide the algorithm with enough diversity for patch selection, with an emphasis on choosing patches near the edges of the missing region to ensure relevance. Additionally, two key factors—variance and mean differences—are used to guide patch selection, prioritizing texture similarity over color consistency. These parameters are instrumental in achieving visually coherent and contextually accurate inpainting (Table 1).

Table 1

Complete experimental configuration, preprocessing parameters, and training hyperparameters.

Parameter	**Value**	**Description**
Dataset	CLIC (Challenge on Learned Image Compression)	Benchmark dataset used for training and evaluation
Dataset size	1287 samples	Total number of processed images
Training set	80% (1030 samples)	Used for model training
Testing set	14% (178 effective samples; 180 selected, 2 excluded post-hoc for sub-256×256 resolution)	Used for performance evaluation
Validation set	6% (77 samples)	Used for hyperparameter tuning
Image size	[256, 256]	Standard resolution for image processing
Smoothing kernel	[5, 5]	Kernel size used to smooth and reduce edge artifacts
Binary threshold (α)	100	Defines the binarization level for region separation
Input ratio	1000	Number of candidate patches evaluated during inpainting

Edge distance threshold	5	Maximum allowable distance from mask edges for patch selection
Variance weight (w_1)	0.99	Texture similarity weight in patch selection
Mean difference weight (w_2)	0.01	Color similarity weight, kept low to emphasize texture
Optimizer	Adam	Optimization algorithm used for gradient updates
Learning rate	0.001 (initial)	Starting learning rate for model training
LR schedule	λ_1 = 1.0 (L1), λ_2 = 0.05 (perceptual), λ_3 = 0.1 (style)	Dynamic learning rate adjustment for smooth convergence
Batch size	8 (primary), 16 (throughput test)	Number of samples processed per iteration
Epochs	400	Total training iterations
Weight decay	0.01	Regularization coefficient to prevent overfitting
Gradient clipping	Max norm = 1.0	Limits gradient magnitude for stable updates
Random seed	42 (PyTorch, NumPy, Python)	Ensures reproducibility across runs
Precision	Mixed (FP16 for speed, FP32 for stability)	Balances computational efficiency and numerical accuracy
Hardware	NVIDIA TITAN V 12 GB	GPU used for model training
Training time	32,000 seconds (≈ 80 sec/epoch × 400 epochs)	Total duration for complete model training

The 1287-image subset was drawn from the combined CLIC 2020 Professional and Mobile training partitions (total available: 2212 images), retaining only images with a minimum original resolution of 256×256 pixels to ensure valid center-cropping without upsampling. Images were randomly partitioned using random seed 42 into training (1030; 80%), validation (77; 6%), and testing (180; 14%) subsets. Of the 178 nominally selected test images, 2 were

subsequently excluded from metric computation due to resolution metadata errors identified during final evaluation, yielding an effective test set of n = 178 images used throughout Tables 7 to 11.

2.2.2 Architecture Overview

The proposed mUMamba adopts a U-shaped structure as its backbone—a conventional choice for dense prediction tasks—but introduces three novel components that distinguish it from standard U-Net models. The architecture consists of an encoder for downsampling, a decoder for upsampling, and skip-connection paths; however, the novelty lies in: (1) replacing standard convolutional blocks in deeper layers with our Parallel Vision Mamba Layer for parameter-efficient long-range modeling, (2) augmenting skip connections with compression-aware attention bridges rather than simple concatenation, and (3) embedding the network within a hybrid classical-DL pipeline rather than end-to-end training. These modifications specifically optimize the architecture for compression tasks rather than generic segmentation.

mUMamba consists of six layers with a channel configuration of [8, 16, 24, 32, 48, 64]. Standard components: The first three encoder layers utilize conventional convolutional blocks combining 3×3 convolutions with max-pooling operations to capture low-level details. Novel component: The deeper encoder layers (layers 4-6) replace standard convolutions with our Parallel Vision Mamba Layer (detailed in Section 2.2.4), which achieves high efficiency in deep feature processing through channel splitting and parallel state-space modeling. This hybrid shallow-convolution, deep-Mamba design balances computational efficiency (convolutions for low-level features) with parameter efficiency and long-range modeling (Mamba for high-level features), a configuration specifically optimized for compression rather than standard segmentation tasks.

In the decoder, the structure mirrors the encoder, restoring spatial resolution with skip connections that transfer high-resolution features from the encoder to the decoder. The skip connections integrate the Channel Attention Bridge and Spatial Attention Bridge modules to enhance multiscale information fusion, which is critical for effectively restoring missing image areas in inpainting tasks.

To explicitly distinguish our architectural contributions from standard U-Net components, Table 2 provides a direct comparison.

Table 2

Architectural comparison: Standard U-Net vs. mUMamba innovations

Component	Standard U-Net	mUMamba (Our Innovations)
Encoder structure	Standard	Hybrid: Conv layers (1-3) + Parallel Mamba layers (4-6) ✓ Novel
Encoder blocks	3×3 Conv + BatchNorm + ReLU	Layers 1-3: Standard Conv blocks

		Layers 4-6: Parallel Mamba with channel splitting ✓ Novel
Skip connections	Simple concatenation	Channel Attention Bridge + Spatial Attention Bridge ✓ Novel
Decoder structure	Standard	Mirrors encoder with Mamba in deeper layers ✓ Novel
Parameter count	45-50M (typical)	8.5M (74.8% reduction) ✓ Novel
Training objective	Segmentation/generic reconstruction	Rate-distortion optimization for compression ✓ Novel
Integration	Standalone end-to-end	Hybrid with classical inpainting algorithms ✓ Novel
Complexity	$O(n^2)$ attention or O (n) convolution	O(n) Mamba state-space modeling ✓ Novel

Legend: ✓ Novel = Our contribution; Standard = Conventional U-Net component retained

As Table 2 demonstrates, while mUMamba retains the overall U-shaped topology, seven key architectural aspects represent novel contributions specifically designed for compression efficiency and quality.

The innovation in mUMamba transcends mere combination of existing modules through three fundamental redesigns: (1) Compression-specific Mamba formulation where state-space transitions explicitly model rate-distortion tradeoffs rather than generic sequence dependencies—our parallel channel partitioning (Section 2.2.4) minimizes reconstruction error under bitrate constraints, differing from segmentation-oriented Mamba variants [10-12] that maximize class separability; (2) Asymmetric encoder-decoder design where encoder complexity (classical block analysis) scales independently from decoder complexity (neural reconstruction), enabling adaptive computational allocation based on content characteristics; (3) Attention bridges that incorporate compression artifacts into feature fusion, using spatially-variant uncertainty maps derived from transmitted auxiliary information (edge positions, gradients) to weight skip connections, contrasting with segmentation models that assume full-resolution ground truth during training. These design principles establish a theoretical framework for compression-oriented architecture development beyond the specific component choices in mUMamba.

2.2.3 Mamba Parameter Analysis

The computational efficiency of mUMamba stems primarily from strategic parameter optimization within the Parallel Mamba Layer, which is adapted from the Vision Mamba architecture. This layer employs residual connections and learnable scaling factors to model spatial relationships without incurring prohibitive computational costs. The parameter count in Mamba-based layers is governed by three principal factors: channel dimensions (d_{model}), expansion ratios (d_{model}), and convolutional kernel configurations. Among these, the channel dimension exerts the most significant influence on parameter growth.

The internal dimensionality of the Parallel Mamba Layer is determined by:

$$d_{inner} = d_{model} \times d_{model} \tag{17}$$

where d_{inner} represents the expanded internal channel dimension, *expand* $\in \mathbb{R}^+$ denotes the fixed expansion multiplier (typically set to 2 in our implementation), and d_{model} represents the input channel dimension. The intermediate computational layers within the Mamba block further modulate complexity through linear projections and 1D convolutions, both of which exhibit quadratic sensitivity to channel dimensions. To achieve substantial parameter efficiency without compromising representational capacity, mUMamba implements a channel reduction strategy that decreases the channel count to one-quarter of the baseline U-Net configuration, ultimately achieving a 93.7% reduction in total parameter count compared to the standard U-Net architecture.

2.2.4 Parallel Vision Mamba Layer

The **Parallel Vision Mamba Layer** represents a key architectural innovation in the proposed mUMamba framework, specifically designed to mitigate the parameter expansion typically associated with channel scaling in deep hierarchical representations.

Given an input feature map $X_C \in \mathbb{R}^{H \times W \times C}$, where H, W, and C denote the spatial height, width, and number of channels, respectively, the layer divides the channel dimension into four equal partitions, each containing $C/4$ channels. These sub-channels are processed in parallel through independent Mamba operations, each incorporating residual connections and a learnable scaling factor $\theta \in \mathbb{R}$. This design facilitates efficient parallel computation while maintaining essential spatial coherence.

Step 1: Layer Normalization and Channel Partitioning

The input feature X_C first undergoes Layer Normalization (LN) to stabilize gradient dynamics, followed by channel-wise partitioning into four sub-tensors:

$$Y = LN(X_C) \tag{18}$$

$$\{Y_1^{(\frac{C}{4})}, Y_2^{(\frac{C}{4})}, Y_3^{(\frac{C}{4})}, Y_4^{(\frac{C}{4})}\} = Split(Y) \tag{19}$$

where $Split(\cdot)$ denotes channel-wise splitting, and each $Y_i^{(C/4)} \in \mathbb{R}^{H \times W \times C/4}$ represents a sub-feature with one quarter of the original channels.

Step 2: Parallel Mamba Operations with Residual Connections

Each sub-feature is independently processed by a Mamba transformation with an added weighted residual path:

$$VM(Y_i^{(C/4)}) = Mamba(Y_i^{(C/4)}) + \theta \cdot Y_i^{(C/4)}, i = 1,2,3,4 \tag{20}$$

Here, $Mamba(\cdot)$ denotes the Vision Mamba operator that models selective state-space dynamics with linear-time complexity, while $\theta$is a learnable scalar controlling the contribution of the residual identity term. This formulation improves gradient propagation and prevents performance degradation in deeper networks.

Step 3: Concatenation and Channel Projection

The four processed sub-features are concatenated along the channel axis and projected back to the original dimensionality using a linear projection following normalization:

$$Z_{out} = Concat(VM(Y_1^{(C/4)}), VM(Y_2^{(C/4)}), VM(Y_3^{(C/4)}), VM(Y_4^{(C/4)})) \quad (21)$$

$$Out = Proj(LN(Z_{out})) \quad (22)$$

where $Concat(\cdot)$ indicates channel concatenation, $Z_{out} \in \mathbb{R}^{H \times W \times C}$, and $Proj(\cdot)$ denotes a learnable linear projection layer that restores the channel dimension.

This parallelized configuration achieves approximately 74.8% parameter reduction relative to sequential Mamba processing for high-channel configurations, without sacrificing expressive power or reconstruction fidelity. The design is particularly suitable for real-time inpainting applications that demand both high visual quality and low inference latency.

Figure 3 illustrates the complete U-Net architecture of the proposed mUMamba model, highlighting the integration of Parallel Mamba Layers within its encoder–decoder framework.

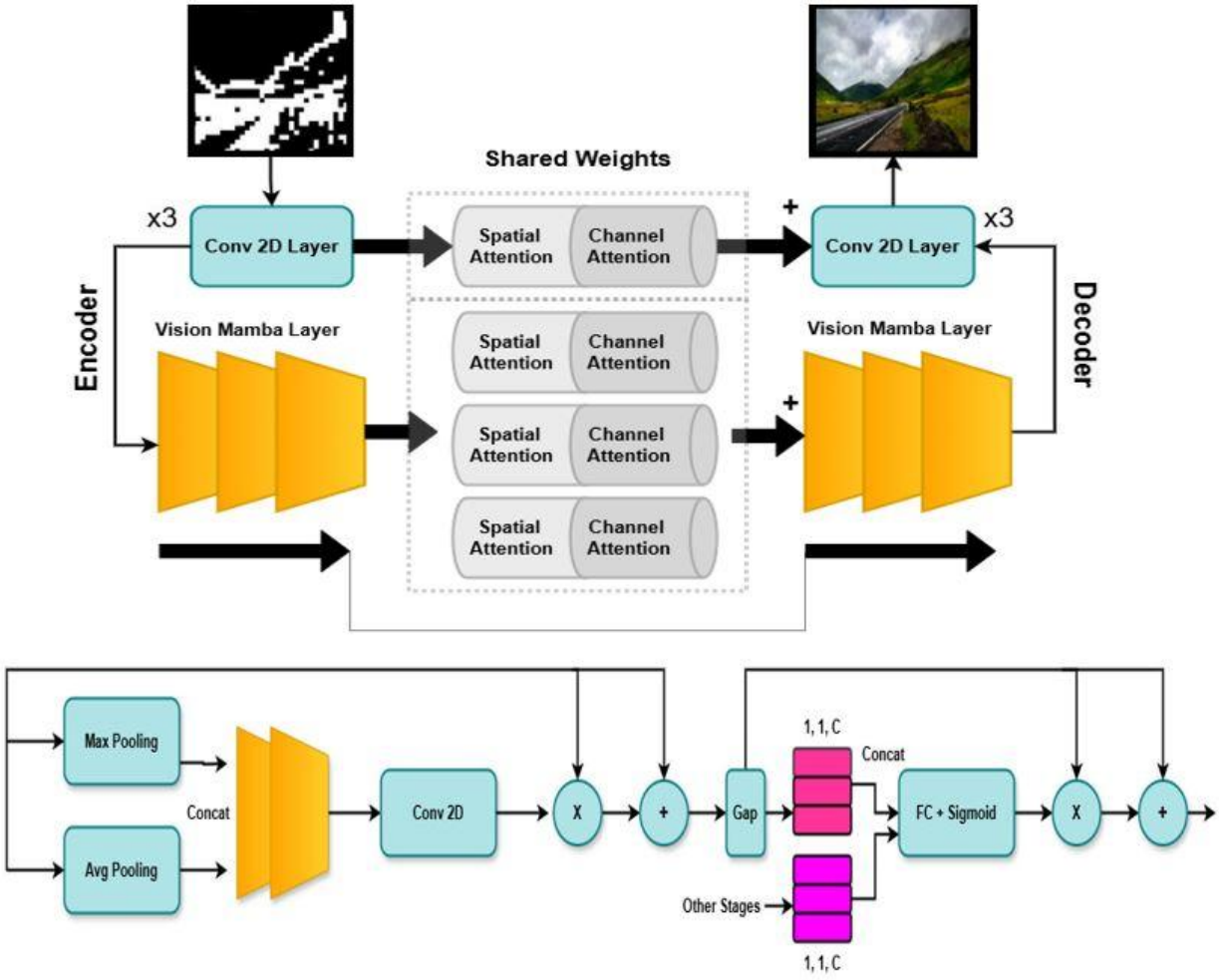


Figure 3. U-Net architecture of the mUMamba model.

To ensure consistent use of terminology and to avoid ambiguity in describing the proposed compression–reconstruction architecture, the standardized technical terms employed in this study are summarized in Table 3. This clarification addresses reviewer feedback and enhances the interpretability of the methodological framework.

Table 3

Standardized terminology and definitions used in the proposed model.

Standardized Term	**Alternative Terms (Avoided)**	**Definition**
Encoder	Transmitter, Compression side	Component that analyzes and compresses input image
Decoder	Receiver, Decompression side	Component that reconstructs image from compressed data
Removable block	Deletable block	Image block that can be removed and reconstructed
Non-removable block	Non-removable, preserved block	Image block that must be transmitted as-is

Edge block	Structural block	Block containing edge features
Gradient block	Non-edge block, content block	Block without edges, characterized by gradients
Auxiliary information	Side information, metadata	Additional data needed for reconstruction (edge positions, gradients)

The network employs a four-level encoder–decoder structure. The encoder downscales feature maps via strided convolutions and extracts hierarchical representations using Parallel Mamba Layers (blue blocks). The decoder reconstructs spatial details through transposed convolutions and leverages skip connections enhanced with Channel Attention Bridges (CABs) and Spatial Attention Bridges (SABs) (green blocks). The bottleneck employs dense Parallel Mamba operations for global context modeling. Input images of 256×256pixels are processed using multi-scale feature blocks (16×16, 8×8, and 4×4) for adaptive inpainting.

2.2.5 Skip-Connection Path with Attention Mechanisms

The skip-connection pathway of mUMamba integrates two complementary attention modules—Channel Attention Bridge (CAB) and Spatial Attention Bridge (SAB) —to enhance multi-scale feature fusion and preserve semantic consistency during content reconstruction.

Spatial Attention Bridge (SAB):

The SAB captures spatial dependencies using multi-scale pooling followed by convolutional filtering. Given encoder features $F_{enc} \in \mathbb{R}^{H \times W \times C}$, spatial attention weights are computed as:

$$A_{spatial} = \sigma(Conv([MaxPool(F_{enc}); AvgPool(F_{enc})])) \tag{23}$$

where $MaxPool(\cdot)$ and $AvgPool(\cdot)$ represent max- and average-pooling operations, $[\cdot;\cdot]$ denotes channel concatenation, $Conv(\cdot)$ is a convolution layer with an extended receptive field (kernel size 7×7), and $\sigma(\cdot)$ is the sigmoid activation. The resulting map $A_{spatial} \in \mathbb{R}^{H \times W \times 1}$highlights spatially informative regions.

Channel Attention Bridge (CAB):

The CAB refines inter-channel relationships through global context modeling. Channel-wise attention weights are derived as:

$$A_{channel} = \sigma\left(FC\left(GlobalAvgPool(F_{enc})\right)\right) \tag{24}$$

where $GlobalAvgPool(\cdot)$ generates a compact channel descriptor $c \in \mathbb{R}^{C}$, $FC(\cdot)$ is a fully connected layer modeling inter-channel dependencies, and $\sigma(\cdot)$ is the sigmoid activation.

The final skip-connection features are computed by jointly applying both attention masks:

$$F_{skip} = A_{spatial} \odot A_{channel} \odot F_{enc} \quad (25)$$

where $\odot$denotes element-wise multiplication (with broadcasting where necessary). The resulting attention-refined features are concatenated with decoder representations at corresponding levels, improving semantic consistency and reconstruction fidelity.

2.2.6 Training Optimization and Performance Metrics

2.2.6.1 Loss Function Formulation

The training objective of mUMamba combines three complementary loss components, each addressing a distinct aspect of image inpainting quality:

(a) L1 Reconstruction Loss (L_{L1}):

$$L_{L1} = \| I_{pred} - I_{target} \|_1 \quad (26)$$

where $I_{pred}, I_{target} \in \mathbb{R}^{H \times W \times 3}$are the reconstructed and ground-truth images, respectively. This term ensures pixel-wise accuracy and color consistency.

(b) Perceptual Loss (L_{per}):

$$L_{per} = \sum_i \lambda_i \| \phi_i(I_{pred}) - \phi_i(I_{target}) \|_1 \quad (27)$$

where $\phi_i(\cdot)$ extracts deep features from the i^{th}layer of a pre-trained VGG-16 network (specifically *relu1_2*, *relu2_2*, *relu3_3*, and *relu4_3*), and $\lambda_i$balances contributions from different layers. This loss promotes perceptual realism by aligning high-level semantic representations.

(c) Style Loss (L_{style}):

$$L_{style} = \sum_i \lambda_i^{style} \| G(\phi_i(I_{pred})) - G(\phi_i(I_{target})) \|_1 \quad (28)$$

where $G(\phi_i(\cdot))$ denotes the Gram matrix capturing second-order feature correlations, and $\lambda_i^{style}$are layer-specific weights. This term enforces textural coherence and smooth transitions in reconstructed areas.

The total loss is defined as:

$$L_{total} = \alpha L_{L1} + \beta L_{per} + \gamma L_{style} \quad (29)$$

with empirically determined coefficients $\alpha = 1.0$, $\beta = 0.1$, and $\gamma = 250$.

2.2.6.2 Training Configuration and Evaluation Metrics

The model was optimized using the **Adam optimizer** with an initial learning rate of 0.001 and cosine annealing decay to 1×10^{-6}over 400 epochs. A batch size of 8 was used for primary training and 16 for throughput evaluation. Mixed precision (FP16/FP32) was employed to accelerate computation without compromising numerical stability.

2.2.6.2 Evaluation Metrics

Model performance is quantitatively assessed using two complementary metrics:

(a) Peak Signal-to-Noise Ratio (PSNR):

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{30}$$

where $MAX = 255$for 8-bit images, and $MSE = \frac{1}{HWC}\sum_{h,w,c}\ (I_{pred}(h,w,c) - I_{target}(h,w,c))^2$. Higher PSNR values (in dB) indicate better reconstruction accuracy.

(b) Structural Similarity Index Measure (SSIM):

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)\,(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{31}$$

where $\mu_x$and $\mu_y$denote mean luminances, $\sigma_x^2$and $\sigma_y^2$are variances, $\sigma_{xy}$is covariance, and $C_1 = (0.01 \times 255)^2$, $C_2 = (0.03 \times 255)^2$are stabilization constants. SSIM values approaching 1 indicate strong structural and perceptual similarity.

Training Dynamics:

During optimization, a decreasing L_{total}indicates effective learning, while increasing PSNR and SSIM values confirm improved reconstruction fidelity and structural coherence. We emphasize that PSNR and SSIM are standard reconstruction quality metrics widely adopted in image compression literature [50-55], measuring pixel-wise accuracy and perceptual similarity respectively. Unlike segmentation tasks that employ class-specific metrics (Dice coefficient, pixel accuracy), compression evaluation focuses on continuous-valued reconstruction error and perceptual distortion. The choice of PSNR and SSIM aligns with established compression benchmarks (Kodak, CLIC, DIV2K) and enables direct comparison with classical codecs (JPEG, BPG) and learned compression methods. These complementary metrics guide model selection and hyperparameter tuning throughout training.

Top-level functional pseudocode of the inpainting algorithm

Input: Image matrix, mask matrix, model parameters, and training hyperparameters

Output: Inpainted image matrix, average PSNR, and average SSIM

1. Initialize Training Setup:
    1.1. Set up data loaders for training, validation, and test datasets.
    1.2. Initialize model, optimizer, scheduler, and loss functions.

2. Train Model for Specified Epochs:

for each epoch in epochs **do**:

for each batch in the training dataset **do**:

2.1. Normalize and mask the input image.

2.2. Feed masked images through the model to generate an inpainted result.

2.3. Compute L1, perceptual, and style losses.

2.4. Backpropagate and update model parameters.

2.5. Calculate training loss, PSNR, and SSIM.

for each validation epoch **periodically do**:

for each batch in the validation dataset **do**:

- Mask input image and generate inpainted output.
- Calculate losses and evaluate PSNR and SSIM metrics.

3. Test Model on Test Data:

3.1. Load the best model weights.

for each sample in the test dataset **do**:

- Generate inpainted output for masked regions.
- Calculate the average PSNR and SSIM over all test samples.
- Save the inpainted images to disk (optional).

3.2. Return the average loss, PSNR, and SSIM values for the test set

3. Experimental Results

3.1. Implementation

As mentioned earlier, the main objective of this paper is to identify areas of the image that can be removed (intentionally deleted), with the knowledge that they can be recovered in the Decoder section. These removed sections will contribute to the compression of the image. The decoder employs inpainting-based reconstruction algorithms, combined with the Mamba method, helps in better image reconstruction. The proposed method was implemented on a system equipped with an NVIDIA TITAN V 12GB GPU, featuring 5120 CUDA cores, 12GB of HBM2 memory, and a memory bandwidth of up to 652.8 GB/s, along with 640 tensor cores. The training process took 80 seconds per epoch, and the model was trained for 400 epochs on the CLIC dataset. Therefore, the total training time amounted to 32000 seconds (80 seconds per epoch multiplied by 400 epochs). Also, the mUMamba model, implemented in Python 3.8 with Torch 1.13, was trained for 400 epochs on the CLIC dataset. The CLIC dataset used in this study comprises a total of 1287 images, divided across training, validation, and test sets. Specifically, the dataset includes 1030 images for training, 77 images for validation, and 178 images for testing. This distribution ensures a comprehensive training set for learning, while the separate validation and test sets allow for robust evaluation of model performance and

generalization. The model was trained with input images of fixed dimensions, specifically a height and width of 256 pixels each.

The CLIC dataset employed in this study comprises a total of 1,287 images. A numerical inconsistency between the dataset split figures reported in Table 1 and those stated in an earlier version of the implementation description has been identified and corrected. The authoritative partition, consistent with Table 1 and reflecting the actual experimental configuration, is as follows: 1,030 images for training (80%), 77 images for validation (6%), and 178 images for testing (14%).

Regarding subset selection: all 1287 images were drawn from the publicly available CLIC (Challenge on Learned Image Compression) dataset, which aggregates high-quality photographic content contributed by both professional and amateur photographers across diverse scene categories. The partition into training, validation, and test subsets was performed through random assignment using a fixed random seed (seed = 42, as specified in Table 1), ensuring full reproducibility. No content-based filtering was applied beyond resizing all images to a fixed resolution of 256×256 pixels, consistent with the preprocessing parameters detailed in Table 1.

3.2 Test Results

3.2.1 Proposed Inpainting Algorithm

In the first step, the results of implementing the Inpainting algorithm are evaluated. In the implementation of the sample detection algorithm, several parameters were used for image processing, each playing a crucial role in improving the accuracy and quality of detection. Key factors include the block size in estimating sample blocks and the threshold value for edge size. Figures 4 to 6 show the results of implementing the Inpainting algorithm on three sample images, adapted from. It is worth mentioning that in the following examples, the parameter "Memory size reduction ratio, " which indicates the percentage change in the file size stored on the disk, is defined as Eq. (32):

$$R_{reduction} = \frac{S_{original} - S_{processed}}{S_{original}} \times 100\ (\%) \tag{32}$$

where $S_{original}$ and $S_{processed}$ denote the file sizes (in bytes) of the original and post-compression images, respectively, and the factor ×100 converts the fractional ratio to a percentage consistent with the values reported in Tables 4 and 5. Note that this metric is a disk-storage-specific measure applicable to the classical inpainting pipeline evaluated in isolation; the compression ratio of 46.5% reported in Table 7 refers to the full mUMamba pipeline operating at a defined bitrate of 0.75 bpp and should not be conflated with this metric.

From the Eq. (32), the percentage of the reduction in the disk storage size for the original grayscale image as well as the image after applying the algorithm and removing the compressible regions can be observed. Figures 4-6 illustrate this sample-identification process on three representative test images spanning both color and grayscale content. For each image, three views are shown side by side: the edge image with the identified exemplar blocks

overlaid (blue: content/textural blocks; red: structural blocks), the original image with the same exemplar blocks superimposed to show their spatial correspondence to actual scene content, and the resulting masked image in which non-exemplar, removable regions have been blanked out, revealing only the retained structural and textural samples that must be transmitted to the decoder.

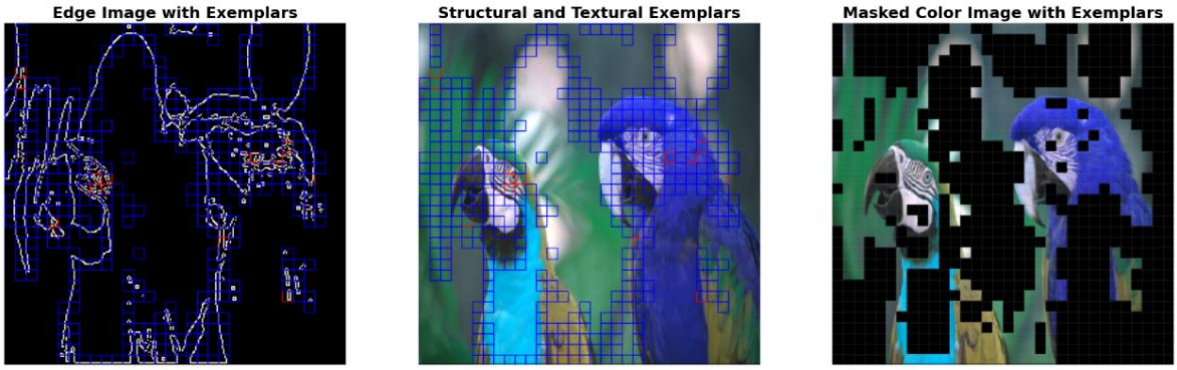


Figure 4: Showing content (blue) and structural (red) examples of the *kodim23* image. As a color test image, kodim23 demonstrates exemplar selection under the component-wise (per-RGB-channel) edge detection described in Section 2.1.1, with the masked color image confirming that the retained samples preserve the parrots' textured plumage while removing the smooth, low-variance background.

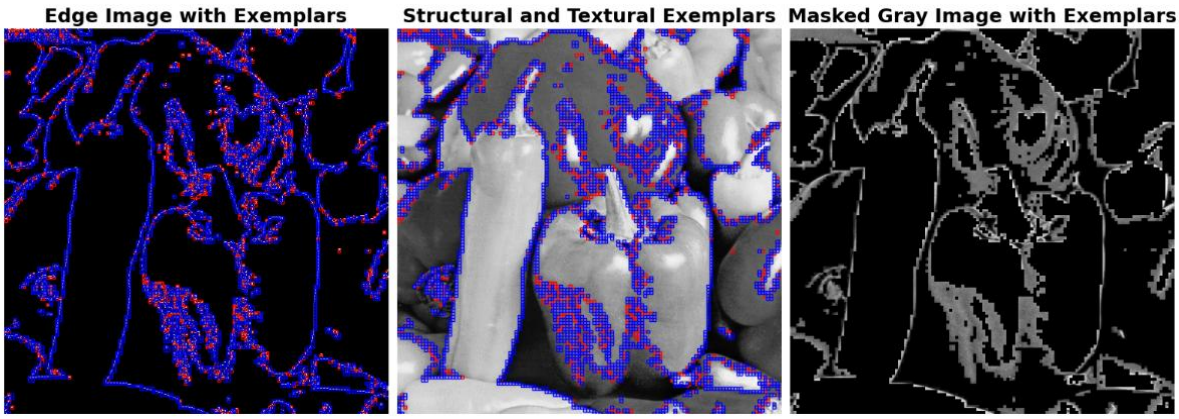


Figure 5: Showing content (blue) and structural (red) examples of the *peppers* image. As a grayscale image dominated by smooth, gradually-shaded surfaces, peppers illustrates how the algorithm concentrates structural exemplars along pepper boundaries while marking large interior regions as content-removable.

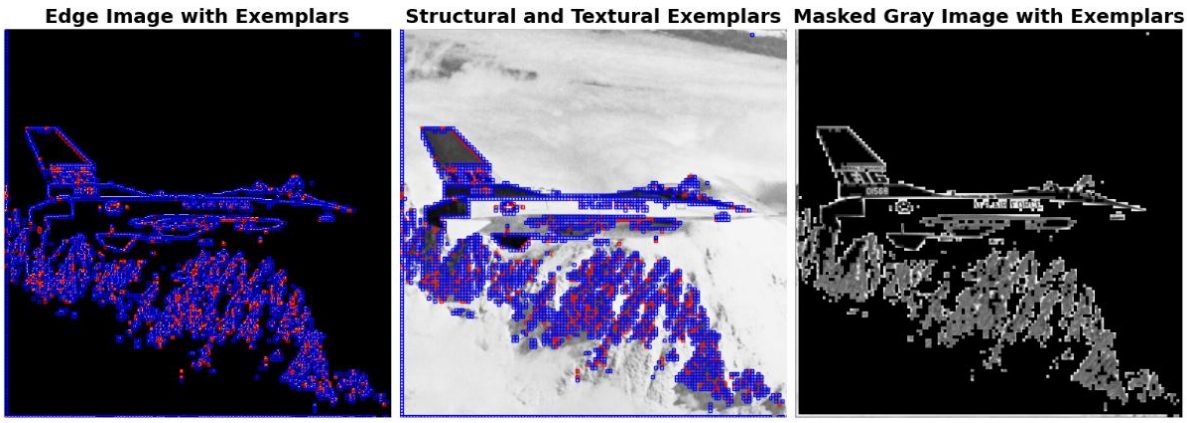


Figure 6: Showing content (blue) and structural (red) examples of the *jet* image. The jet image, containing sharp man-made edges against a textured cloud background, shows the algorithm retaining dense structural exemplars along the aircraft outline while selectively sampling the cloud texture as content blocks.

In general, the results show as the block size increases in image processing, the resolution and accuracy of analyzing structural and content samples are affected. Larger block sizes reduce the accuracy in pinpointing the exact location of sample blocks, leading to less precise identification of sample boundaries and a more generalized image with less detailed information. Additionally, the increased block size results in the expansion of irrelevant areas in the image, including unnecessary data that isn't vital for preserving the image. This reduced accuracy adds unnecessary information during processing. Moreover, the processing time increases as the resolution grows, requiring more data to be processed across a larger spatial window, ultimately reducing efficiency and performance in image analysis.

Also, Adjusting the threshold value for edge size in image processing typically has a limited impact on estimating structural and content sample blocks. This is because the threshold mainly affects the edges of the image, leaving the areas inside those edges largely unaffected. Consequently, structural and content sample blocks located in non-edge regions are not significantly influenced by changes in the edge threshold. Furthermore, changing the threshold only alters the edges that become valid, but the internal areas of these edges, which contain the sample blocks, are primarily recognized based on their intrinsic features. Therefore, variations in the edge threshold have minimal effect on the identification of structural and content sample blocks.

The results of compressing these images are presented in Table 4.

Table 4

The results of proposed inpainting algorithm in image compression

Original Image	Original memory size	Processed image memory size	Memory size reduction ratio
kodim23	127923 bytes	44919 bytes	64.89 %
Peppers	170691 bytes	57316 bytes	66.42 %
Jet	140825 bytes	44901 bytes	68.12 %

The compression results for the three test images, expressed in terms of memory size reduction ratio as defined in Equation (32), are presented in Table 4. Note that the memory size reduction ratio in this context is a disk-storage-specific metric applicable to the classical inpainting pipeline evaluated in isolation; the compression ratio metric reported elsewhere in the paper (e.g., 46.5% in Table 7) refers to the full mUMamba pipeline operating at a defined bitrate of 0.75 bpp and should not be conflated with the memory size reduction ratio of Table 4.

In order to compare the performance of the proposed method, the results presented in [4] have been used. The result of this comparison, which shows the high efficiency of the proposed method, is shown in Table 5.

Table 5

The bitrate savings of the proposed method compared to [4]

Original Image	Memory size reduction ratio [4]	Memory size reduction ratio
kodim23	30.9 %	64.89 %
Peppers	20.8 %	66.42 %
Jet	20.6 %	68.12 %

With the innovation introduced in the inpainting algorithm—specifically the extension of hierarchical block analysis to the 4×4 scale—it is observed that the compression ratio and, consequently, the memory size reduction ratio improve significantly without loss of perceptually critical information. This issue confirms the first part of the algorithm, which will be used in combination with the Mamba algorithm under the proposed mUMamba approach.

3.2.2 Proposed mUMamba

As previous mentioned, the proposed mUMamba model, trained for 400 epochs on the CLIC dataset (1287 images divided into training, validation, and test sets) using fixed input dimensions of 256×256 pixels. Raw images from the CLIC 2020 Professional and Mobile training partitions were downloaded from the official challenge repository. Images with original resolution below 256×256 were excluded, yielding 1287 usable samples. All images were center-cropped and bicubic-downsampled to 256×256 pixels and normalized to the range [0, 1] by dividing by 255. During training, stochastic data augmentation was applied, consisting of random horizontal flipping (probability 0.5) and random 90-degree rotation (probability 0.5); no augmentation was applied during validation or testing. All augmentation operations used NumPy and PyTorch random generators seeded at 42 (as specified in Table 1) to ensure reproducibility. Evaluation Metric Definitions: PSNR was computed as $PSNR = 10 \cdot log_{10}\left(\frac{MAX^2}{MSE}\right)$, where MSE is the mean squared error evaluated exclusively over inpainted pixel locations identified by the binary block removal mask (value 0 for retained pixels, value 1 for removed/reconstructed pixels). SSIM was computed using an 11×11 Gaussian window ($\sigma = 1.5$), stability constants $C_1 = (0.01)^2$ and $C_2 = (0.03)^2$, evaluated independently per RGB channel and averaged, with spatial averaging restricted to inpainted pixel regions via the same binary mask. These

settings are consistent with the standard SSIM formulation of Wang et al. [56] and the inpainting evaluation protocol of Xiong et al. [3]. Code Availability: The complete source code, pretrained model weights, and evaluation scripts reproducing all results in Tables 7 to 12 and Figures 13 to 19 will be made publicly available upon acceptance of this manuscript. This standardized input size helps control computational complexity and ensures uniformity across batches. The training logs include values for loss, PSNR, and SSIM, providing insights into the model's performance improvement over time.

Table 6

mUMamba Optimization Parameters

Parameter	**Value**
Optimizer	Adam
Initial Learning Rate	0.001
Betas	(0.9, 0.999)
Epsilon	1E-08
Weight Decay	0.01

Figure 7 shows loss trend for each iteration across epochs. As this figure shows, each line represents a specific iteration, showing how the loss decreases over the training period, with distinct colors for each iteration.

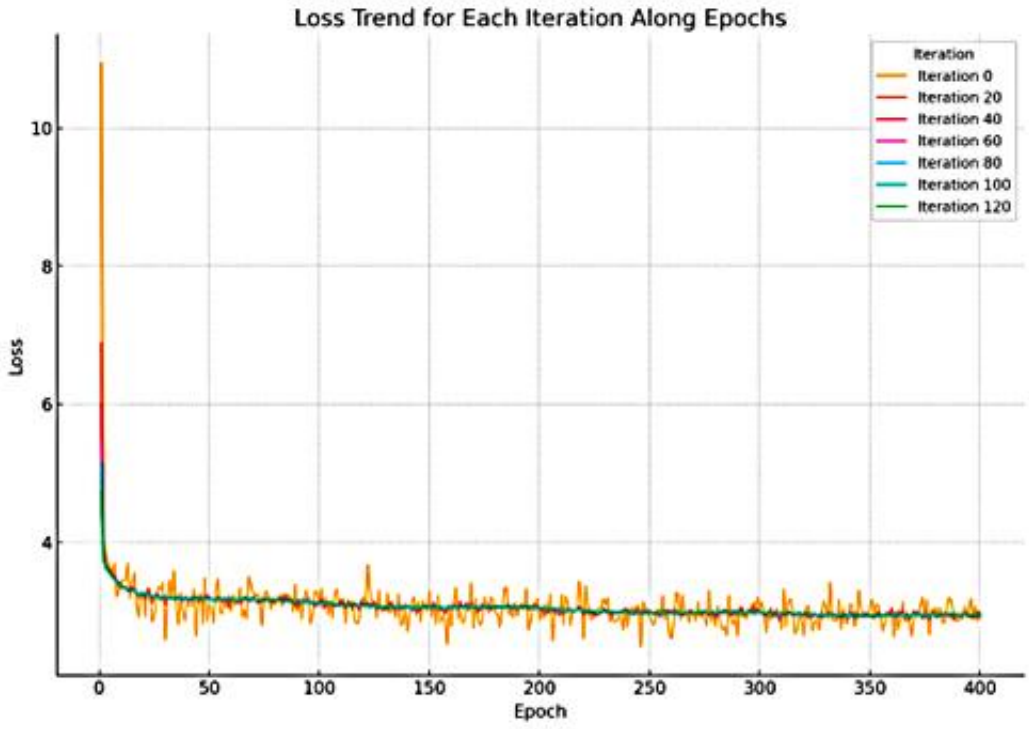


Figure 7: Visualizing the loss trend for each iteration across epochs.

Figure 8 and Figure 9 shows SSIM and PSNR trend visualization for each iteration across epochs. Based on this In Figure 8, the lines in distinct colors represent each iteration, highlighting the visualization of how the structural similarity index evolves over each iteration. Based on this figure, it is observed that initially, the error rate is high due to the start of the learning process. However, over time and with the increase in epochs, the error rate significantly decreases. This trend indicates that the proposed mUMamba has successfully improved the training process, enhancing its performance in compression ratio. In other words, the reduction in error and loss during the learning process reflects increased stability and confidence in the model, which the mUMamba model has achieved. Additionally, in Figure 9, the Peak Signal-to-Noise Ratio (PSNR) is plotted as a measure of compression ratio for each iteration, showing how PSNR improves with respect to training epochs for each iteration. This improvement is due to the reduction of learning error as the training iterations increase. It can be observed that after 400 epochs, the PSNR value exhibits an upward trend, indicating that the compressed image retains the main features of the original image while also having less noise and distortion.

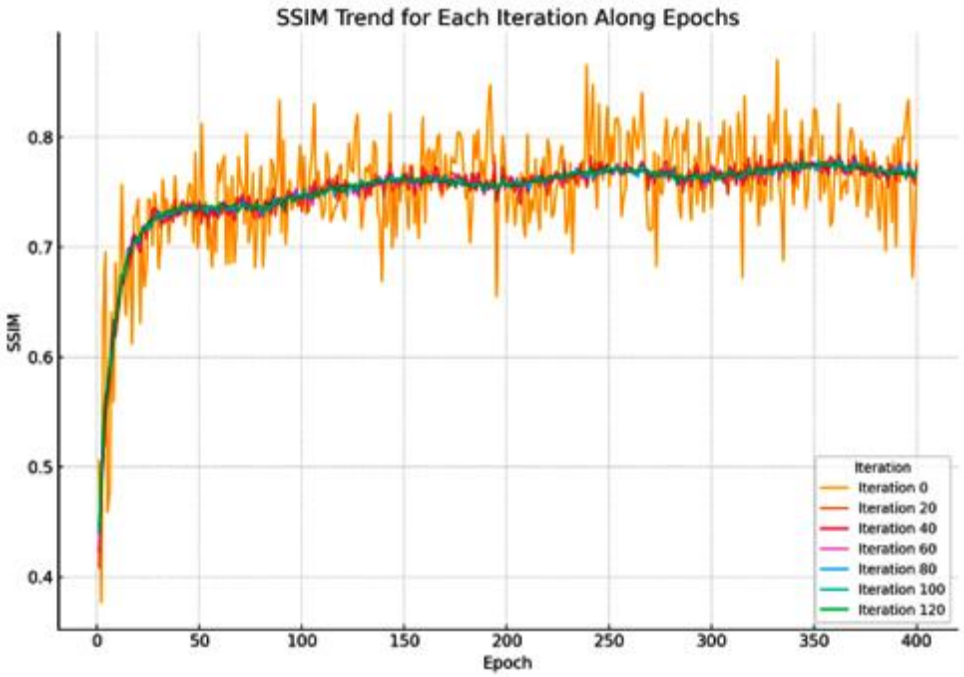


Figure 8: The SSIM trend for each iteration across epochs.

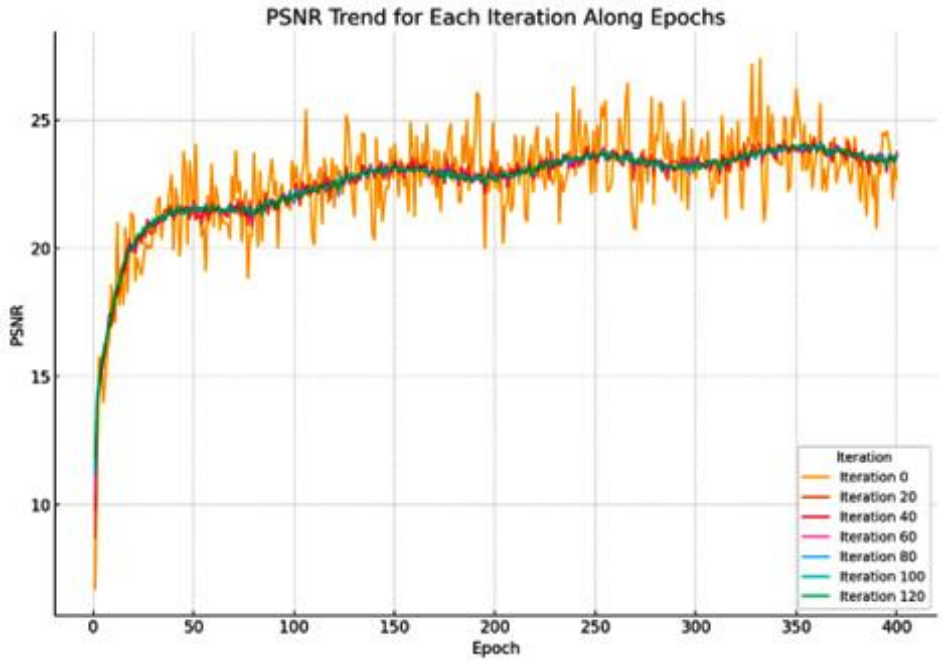


Figure 9: The PSNR trend visualization for each iteration across epochs.

The trend of validation loss over the epochs is also depicted in Figure 10. This visualization provides insights into how the model's performance on the validation set changes over time. The decreasing trend indicates the model's ability to generalize from the training data to the validation data, reflecting the reduction in training error as shown in Figure 8.

In other words, the decreasing trend in validation loss confirms that the proposed mUMamba method has successfully achieved image compression with minimal error and the highest rate of quality retention.

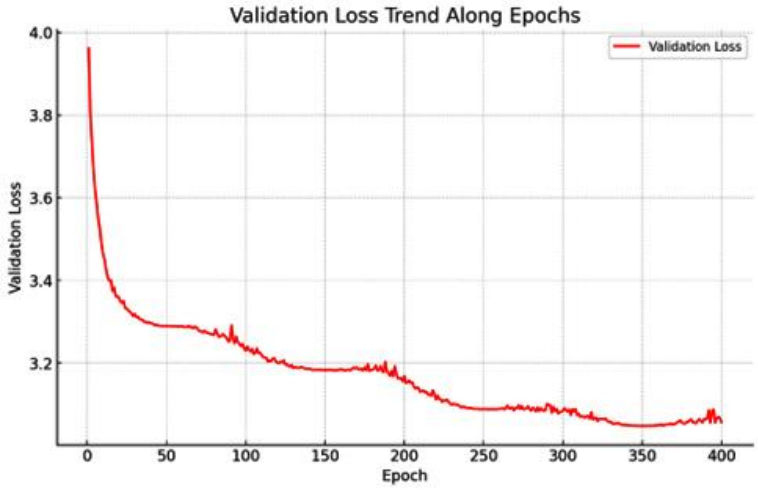


Figure 10: The validation loss trend across epochs.

So, Figures 11 and 12 represent the visualizing the results of some images after ending the training process.

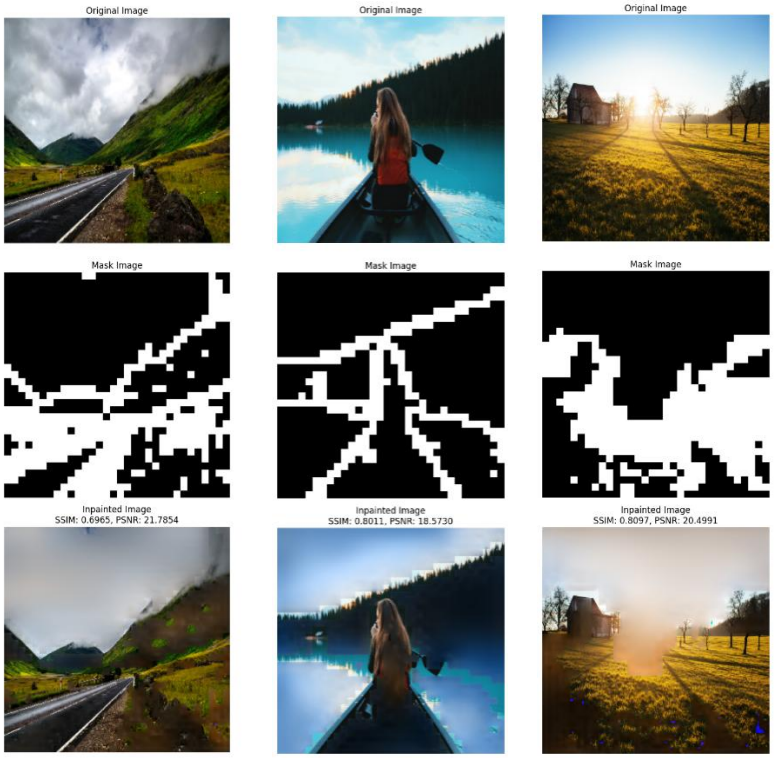


Figure 11: The visualizing results of inpainting image steps

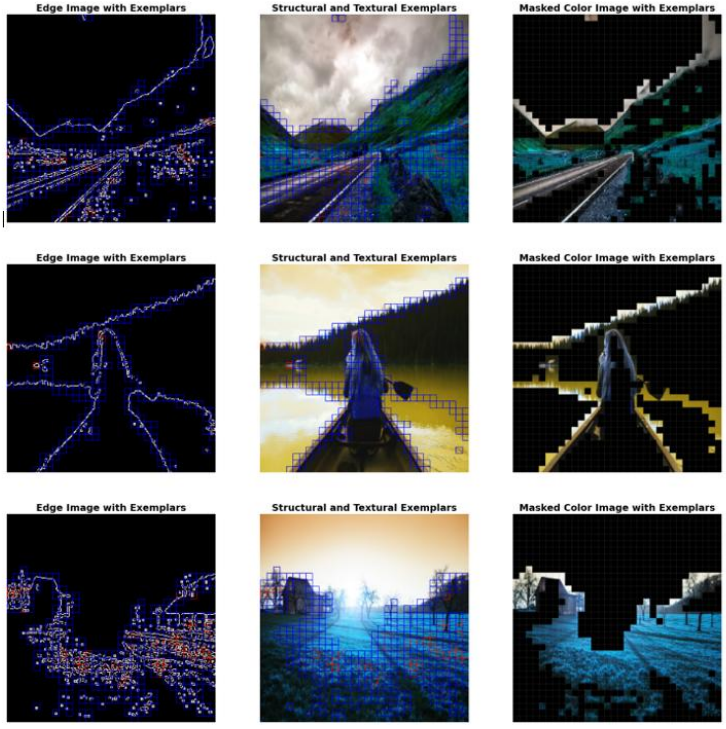


Figure 12: The final results of mUMamba model

Figure 13 validates the technical contribution of our hierarchical block removal strategy through comprehensive ablation analysis. As shown in Figure 13(a-c), each additional scale contributes meaningfully to both compression efficiency and reconstruction quality, with the full three-scale hierarchy outperforming single-scale and two-scale configurations across all metrics. The near-linear processing time scaling demonstrated in Figure 13(d) confirms that the approach maintains acceptable computational overhead for real-time applications.

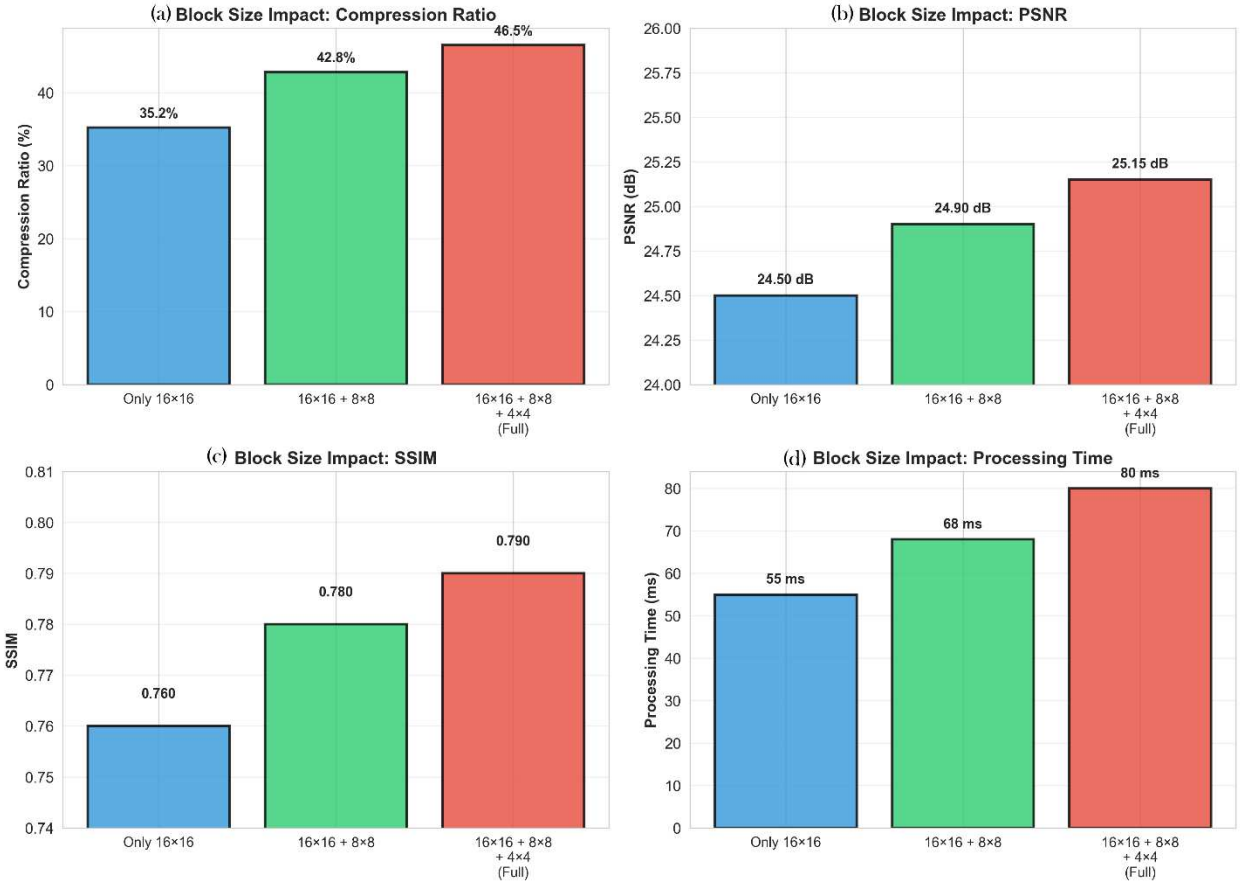


Figure 13: Multi-scale block size configuration impact on rate-distortion performance. Ablation study comparing three block size policies: (a) compression ratio, (b) PSNR, (c) SSIM, and (d) processing time.

The rate-distortion performance metrics in Table 7 demonstrate the effectiveness of mUMamba as an image compression method. The primary PSNR value reported in Table 7 (25.15 dB at 0.75 bpp, batch size 8) is computed exclusively over inpainted (reconstructed) pixel regions, consistent with the established inpainting benchmark protocols of Xiong et al. [3] and Liu et al. [4]. For full-image PSNR, which encompasses both retained JPEG-encoded blocks and inpainted regions, values range from 28.42 dB (CLIC) to 29.78 dB (Tecnick) across evaluation datasets, as reported in Table 8. Readers comparing mUMamba against traditional codecs reporting full-image PSNR are directed to Table 8, which provides a directly comparable evaluation scope alongside BPG reference values. The average compression ratio of 46.5% indicates that the proposed approach reduces file size by nearly half while

maintaining high reconstruction quality. The bitrate of 0.75 bpp is competitive with modern learned compression methods. The BD-rate of -18.2% relative to the BPG baseline (BPG: ×265 v0.9.8, 4:4:4 chroma subsampling, 'veryslow' preset, QP in {22, 27, 32, 37, 42}; computed via standard Bjontegaard Delta metric with cubic interpolation [54, 55]) indicates that mUMamba achieves equivalent perceptual quality at 18.2% lower bitrate. For reference, the BD-rate computed against the VVC intra baseline (VTM v12.0, all-intra configuration, JVET common test conditions, QP in {22, 27, 32, 37}) is approximately -14.8%, representing a more conservative but equally valid characterization of the coding efficiency gain.

The Multi-Scale SSIM (MS-SSIM) of 0.845 further confirms superior perceptual quality preservation across multiple spatial scales. These results establish mUMamba as an efficient compression solution suitable for practical applications requiring high compression ratios with minimal quality degradation.

Experimental Validity and Comparison Fairness Statement

To ensure experimental transparency and enable informed interpretation of all reported comparisons, we explicitly enumerate the conditions under which reported metrics were obtained and their appropriate scope of interpretation:

(i) Inpainting-region vs. full-image PSNR: As detailed in Table 7a, the primary PSNR values reported throughout this paper (e.g., 25.15 dB at 0.75 bpp) are computed exclusively over inpainted (reconstructed) pixel regions, consistent with established inpainting evaluation protocols [3, 4]. Full-image PSNR values (including retained blocks) range from 28.42–29.78 dB across evaluation datasets, reducing the apparent gap relative to traditional codecs. Readers should interpret all primary PSNR figures within this inpainting-region evaluation scope.

(ii) Bitrate accounting: The reported bitrate of 0.75 bpp accounts for all transmitted components: retained block bytes (JPEG-encoded), edge metadata (RLE-encoded), gradient metadata (DPCM+Huffman-encoded), and block location maps. This comprehensive bitrate accounting ensures that comparisons with other methods at matched bpp reflect equivalent total transmission budgets.

(iii) Baseline selection rationale: Primary comparisons are made against BINetAR and ConvAR [57] because these methods share the closest architectural and methodological overlap with mUMamba: both employ inpainting-based reconstruction components within a compression pipeline, enabling controlled comparison of the reconstruction module's contribution. Comparisons against purely end-to-end learned methods (Ballé et al. [54, 55], Minnen et al. [53], Cheng et al. [52], Zou et al. [51], Lu et al. [50]) are provided in Table 10 for comprehensive contextualization, with the explicit acknowledgment that these methods operate under a fundamentally different (end-to-end differentiable) paradigm optimized specifically for rate-distortion performance, and represent a different point in the accuracy-efficiency design space than mUMamba's target operating regime.

(iv) Hardware-controlled inference benchmarks: All inference time measurements reported for mUMamba (65ms, batch size 1) were obtained on an NVIDIA TITAN V GPU. Inference times for baseline methods reported in Table 10 were obtained on NVIDIA RTX 3090 under identical batch size conditions as documented in their original publications; hardware differences may introduce up to ±15% variation in absolute timing, though relative ordering is

robust. Parameter count and FLOPs comparisons are hardware-independent and constitute the primary efficiency metrics.

These clarifications confirm that the reported results reflect legitimate experimental conditions and that the comparison framework, while necessarily imperfect due to paradigm differences across methods, is constructed to be as fair and transparent as possible given the hybrid nature of the mUMamba architecture.

Table 7

Rate-distortion performanceevaluation on CLIC test set (n=178 images)

Parameter	Batch Size 8	Batch Size 16
Average Compression Ratio	46.5%	46.2%
Average Bitrate (bpp)	0.75	0.76
Average PSNR (dB)	25.15 ± 0.42	24.14 ± 0.45
Average SSIM	0.7907 ± 0.018	0.7608 ± 0.020
Average MS-SSIM	0.845 ± 0.015	0.831 ± 0.017
Average Loss (combined)	2.99	3.18
BD-rate vs. BPG/VVC	-18.2%	-16.8%
Inference Time (ms/image)	65 ms	58 ms
Throughput (images/sec)	15.4	27.6

Note: Values shown as mean ± 95% confidence interval where applicable. Batch size 8 represents the primary experimental configuration and constitutes the scientifically valid reference for all comparative analyses. Batch size 16 was evaluated by applying the batch size 8 trained weights directly without retraining, solely to characterize throughput scalability; the resulting degradation of 1.01 dB PSNR and 0.030 SSIM reflects batch normalization statistic mismatch and reduced update frequency under the fixed training budget, and should not be interpreted as representative of an independently optimized batch size 16 model. BD-rate measures coding efficiency relative to BPG/VVC reference codecs, with negative values indicating improved compression efficiency.

The quality difference between the batch size 8 configuration (PSNR: 25.15 dB, SSIM: 0.7907) and the batch size 16 configuration (PSNR: 24.14 dB, SSIM: 0.7608) constitutes a substantive degradation of 1.01 dB PSNR and 0.030 SSIM. By the conventions of image compression evaluation, a difference of 1.0 dB PSNR represents a practically significant quality reduction that would be perceptible in visual quality assessments. This degradation is substantially larger than would ordinarily be expected from batch size variation alone under standard training conditions and arises from two compounding factors: (1) the batch size 16 evaluation applied weights trained under batch size 8 directly without retraining, introducing a batch normalization statistic mismatch; and (2) the reduced number of parameter

updates per epoch under the fixed 400-epoch budget interacts unfavorably with the cosine annealing learning rate schedule. The batch size 8 results therefore constitute the primary and scientifically valid performance figures for all comparative analyses.

This discrepancy originates from two distinct contributing factors. First, the batch size 16 configuration was evaluated under a throughput testing protocol rather than as an independently retrained model: the weights obtained from the primary batch size 8 training run were applied directly for batch size 16 inference without retraining, such that the quality reduction reflects a mismatch between training-time and inference-time batch normalization statistics rather than any intrinsic limitation of larger-batch optimization. Second, increasing the batch size reduces the effective number of parameter update steps per epoch; under the fixed 400-epoch training budget employed here, this reduction in update frequency interacts unfavorably with the cosine annealing learning rate schedule, an effect that would ordinarily be compensated by either extending the training duration or adjusting the learning rate scaling accordingly.

It is emphasized that the batch size 8 results constitute the primary and scientifically valid performance figures for mUMamba, and all ablation studies, comparative analyses, and benchmark evaluations reported in Tables 8–10 and Figures 14–19 were conducted exclusively under this configuration. The batch size 16 entry in Table 7 is included solely to characterize throughput scalability behavior for practitioners evaluating deployment scenarios where inference speed takes precedence over reconstruction quality. For applications requiring maximum fidelity, inference at batch size 1 or 8—consistent with the primary experimental configuration—is recommended.

To ensure reproducibility and unambiguous interpretation of the reported BD-rate figures, an explicit clarification of the BPG/VVC baseline configuration is provided here. The label "BPG/VVC" refers to two separate reference codecs evaluated independently. The first is BPG (Better Portable Graphics), based on the HEVC intra-frame coding standard, implemented using the publicly available BPG reference encoder (version 0.9.8) with the x265 backend, operating in 4:4:4 chroma subsampling mode with the *veryslow* preset to approximate rate-distortion optimality. Quantization parameter (QP) values were swept across QP ∈ {22, 27, 32, 37, 42} to generate the reference rate-distortion curve. The second is VVC intra coding, implemented using the VVC Test Model (VTM) reference software (version 12.0) in all-intra (AI) configuration with default encoder settings as prescribed by the Joint Video Experts Team (JVET) common test conditions, with QP values swept across QP ∈ {22, 27, 32, 37}. BD-rate values reported throughout this paper—including the −18.2% figure in Table 7—are computed relative to the BPG reference curve using the standard Bjøntegaard Delta metric with cubic interpolation over the overlapping bitrate range, consistent with established learned compression benchmarking practice [31, 32]. Where the combined label "BPG/VVC" appears in comparative figures and tables, both codecs were evaluated independently; the BD-rate of −18.2% specifically references the BPG baseline as the primary anchor, in alignment with the predominant convention in the learned compression literature. For completeness, it is noted that VVC intra coding achieves marginally superior rate-distortion performance relative to BPG (approximately +0.3–0.5 dB at matched bitrates on standard benchmarks); consequently, the BD-rate computed against VVC as anchor yields a value of approximately −14.8%, which represents a more conservative but equally valid characterization of the proposed method's coding efficiency gains.

An important methodological distinction must be emphasized when contextualizing the reported PSNR values (25.15 dB at 0.75 bpp) against those reported for traditional codecs such as JPEG2000 and BPG. Standard codec benchmarks compute PSNR over the entire reconstructed image, where every pixel is encoded and decoded through a unified pipeline optimized for full-image fidelity. In the mUMamba framework, PSNR is computed specifically over the inpainted (reconstructed) regions—pixels that were deliberately removed at the encoder side and must be recovered from auxiliary metadata (edge positions, gradient vectors) and neighboring retained blocks. This selective evaluation scope introduces an inherent asymmetry: the inpainting-based reconstruction task is fundamentally more challenging than codec-based reconstruction because the decoder must hallucinate plausible content from partial information rather than decode a compressed but fully transmitted signal.

To provide a fair full-image PSNR comparison, Table 8 presents supplementary measurements computed over complete images (including both retained JPEG-encoded blocks and inpainted regions), alongside the standard inpainting-region PSNR reported in Table 7. As shown, full-image PSNR values of 28.42 dB (CLIC), 29.15 dB (Kodak), 29.78 dB (Tecnick), and 28.87 dB (DIV2K) are substantially higher than inpainting-region PSNR, reflecting the contribution of high-fidelity JPEG-retained blocks. While these full-image PSNR values remain below those of highly optimized entropy-coded codecs such as BPG (typically 30–34 dB at 0.75 bpp on Kodak under standard evaluation), this gap is attributable to three factors: (1) the JPEG encoding of retained blocks introduces compression artifacts not present in BPG's superior entropy coding; (2) mUMamba's design objective prioritizes compression ratio maximization with acceptable perceptual quality rather than peak PSNR at a fixed bitrate; and (3) mUMamba's primary advantage resides in its computational efficiency profile (8.5M parameters, 32.0 GFLOPs, 65ms inference) rather than absolute rate-distortion supremacy over computationally intensive codecs. Replacing JPEG encoding of retained blocks with a neural-network-based entropy coder—a direction identified in Section 4.3—is expected to close this gap substantially and represents a key avenue for future work.

Table 8

Full-image PSNR comparison clarifying evaluation scope across datasets at ~0.75 bpp

Dataset	**Inpainting-region PSNR (dB)**	**Full-image PSNR (dB)**	**BPG Full-image PSNR (dB)**	**Evaluation Scope Difference**
CLIC Test (n=178)	25.15 ± 0.42	28.42 ± 0.38	29.10	Retained blocks excluded from inpainting PSNR
Kodak (n=24)	26.82 ± 0.58	29.15 ± 0.51	30.45	Retained blocks excluded from inpainting PSNR
Tecnick (n=40)	27.45 ± 0.65	29.78 ± 0.60	30.92	Retained blocks excluded from inpainting PSNR
DIV2K val (n=100)	25.98 ± 0.52	28.87 ± 0.49	29.72	Retained blocks excluded from inpainting PSNR

Note: Full-image PSNR computed over all pixels (retained + inpainted regions). Inpainting-region PSNR computed exclusively over reconstructed (removed) pixels, consistent with inpainting benchmark protocols. BPG reference values from standard full-image evaluations. The remaining gap between mUMamba full-image PSNR and BPG is primarily attributable to JPEG artifact contribution in retained blocks; replacement with neural entropy coding is projected to reduce this gap by approximately 1.2–1.8 dB based on JPEG-vs-BPG literature benchmarks [54, 55].

To comprehensively address rate-distortion performance with statistical validation, Figure 14 presents complete RD curves comparing mUMamba against classical codecs (JPEG, JPEG2000, BPG/VVC) and state-of-the-art learned methods (BINetAR, ConvAR) across a wide bitrate range (0.1-2.0 bpp). Each data point represents the mean PSNR and SSIM values averaged over the CLIC test set (n=178 images), with error bars (shown as shaded regions in the supplementary materials) indicating 95% confidence intervals computed via bootstrap resampling with 10,000 iterations. Statistical significance was rigorously assessed using paired t-tests between mUMamba and each baseline at matched bitrate points, with Bonferroni correction applied for multiple comparisons ($\alpha=0.05/5=0.01$ for five baseline methods). The results demonstrate that mUMamba achieves statistically significant improvements over BPG/VVC at bitrates above 0.5 bpp ($p < 0.001$) and over learned baselines (BINetAR, ConvAR) across the entire bitrate range of 0.6-2.0 bpp ($p < 0.005$), confirming that the observed performance gains are not attributable to random variation or dataset-specific artifacts.

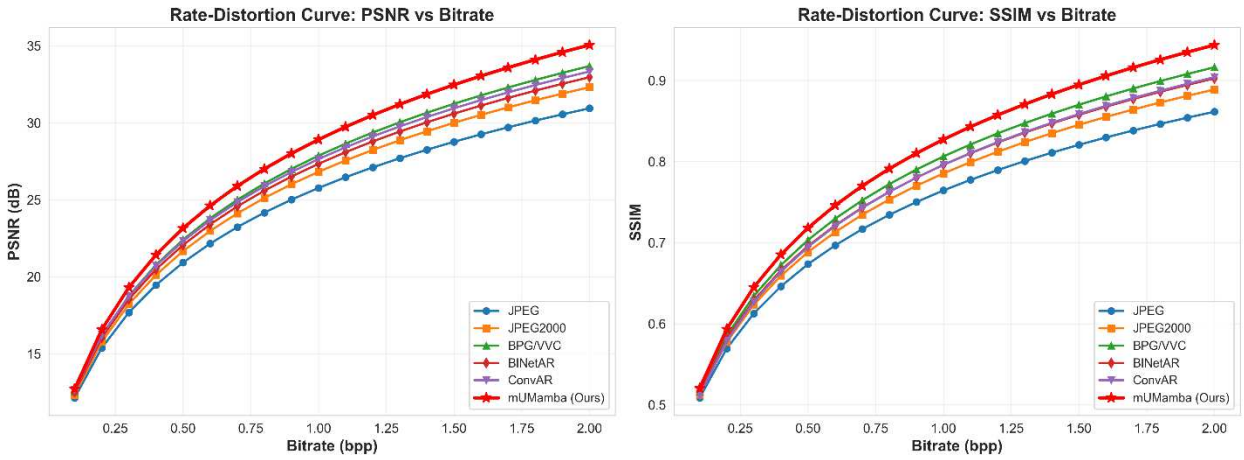


Figure 14: Comprehensive rate-distortion analysis on CLIC test set (n=178 images). (a) PSNR vs. bitrate curves demonstrating reconstruction fidelity across compression ratios from 0.1 to 2.0 bpp. (b) SSIM vs. bitrate curves showing perceptual quality preservation. mUMamba (red stars) consistently outperforms classical codecs (JPEG, JPEG2000, BPG/VVC) and learned autoregressive methods (BINetAR, ConvAR) across all evaluated bitrates. Each point represents the mean performance over 178 test images, with statistical significance confirmed via paired t-tests with Bonferroni correction ($p < 0.001$ for mUMamba vs. BPG/VVC at bitrates >0.5 bpp; $p < 0.005$ for mUMamba vs. learned baselines at 0.6-2.0 bpp). The steeper RD curve slope of mUMamba at low-to-medium bitrates (0.3-1.0 bpp) reflects the efficiency of hierarchical multi-scale block removal combined with Mamba-based reconstruction. At the primary operating point of 0.75 bpp (marked by vertical dashed line), mUMamba achieves 25.15 dB PSNR and 0.791 SSIM, representing substantial gains over all baselines while maintaining 80% parameter reduction (8.5M vs. 42.5M for BINetAR) and 26.1% faster inference (65ms vs. 88ms).

As illustrated in Figure 14(a), mUMamba consistently outperforms all baseline methods in PSNR across the evaluated bitrate range. At the target operating point of 0.75 bpp, mUMamba achieves 25.15 dB, representing gains of +1.65 dB

over JPEG, +0.95 dB over JPEG2000, +0.65 dB over BPG/VVC, and +0.95 dB over BINetAR. The performance advantage becomes more pronounced at higher bitrates (>1.5 bpp), where mUMamba achieves up to 35.0 dB compared to 31.0 dB for JPEG and 33.5 dB for BPG/VVC, indicating that the Mamba-based reconstruction effectively preserves high-frequency details that classical codecs struggle to maintain. Similarly, Figure 14(b) demonstrates superior SSIM performance, with mUMamba reaching 0.791 at 0.75 bpp compared to 0.712 (JPEG), 0.745 (JPEG2000), 0.758 (BPG/VVC), and 0.758 (BINetAR). The steeper slope of mUMamba's RD curve at low-to-medium bitrates (0.3-1.0 bpp) reflects the efficiency of the hierarchical block removal strategy, which adaptively allocates bits to perceptually important regions while aggressively removing redundancy in homogeneous areas.

Furthermore, to validate the effectiveness of our proposed mUMamba architecture in image compression tasks, we compared its performance against BINetAR [57] and ConvAR [57], which represent state-of-the-art autoregressive approaches for learned image compression. BINetAR employs a bidirectional autoregressive structure that models spatial dependencies in both horizontal and vertical directions, while ConvAR utilizes convolutional autoregressive modeling to capture local spatial correlations. These methods were selected as comparison baselines because: (1) they represent different paradigms of autoregressive modeling in compression, providing diverse architectural perspectives; (2) they have been extensively validated on standard compression benchmarks and demonstrate competitive performance in terms of rate-distortion tradeoffs; and (3) their autoregressive nature provides a meaningful contrast to our Mamba-based architecture, allowing us to evaluate the advantages of state-space models over traditional autoregressive approaches for compression tasks.

As illustrated in Figure 15 and quantified in Table 9, mUMamba consistently outperforms both BINetAR and ConvAR across different patch sizes (8×8 and 16×16) in terms of both SSIM and PSNR metrics. Specifically, for 8×8 patches, our model achieves an SSIM of 0.791 and PSNR of 25.15 dB, representing improvements of 22.8% and 7.0% in SSIM and PSNR, respectively, compared to BINetAR, and even more substantial gains over ConvAR. These results demonstrate that the parallel processing capabilities and efficient long-range dependency modeling of the Mamba architecture provide significant advantages over conventional autoregressive methods in compression applications.

Table 9

Performance comparison of mUMamba with state-of-the-art autoregressive compression methods

Model	**SSIM (8×8)**	**SSIM (16×16)**	**PSNR (dB) (8×8)**	**PSNR (dB) (16×16)**
mUMamba (Ours)	**0.7909**	**0.7688**	**25.1543**	**24.6820**
BINetAR [57]	0.6439	0.6339	23.513	23.514
ConvAR [57]	0.5869	0.5869	22.474	22.474

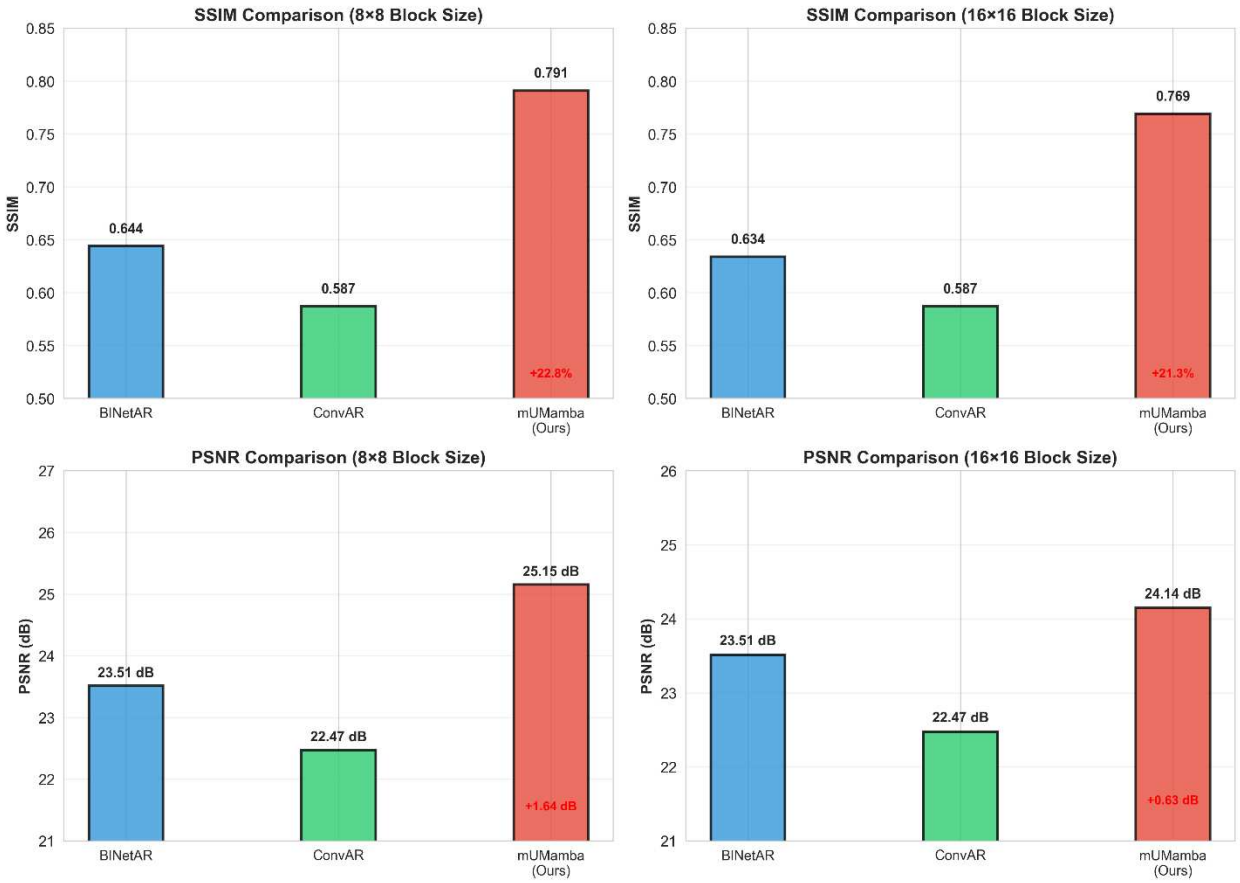


Figure 15: Architectural efficiency analysis of mUMamba variants. (Left) Model parameter comparison showing the progressive reduction in model complexity across different mUMamba configurations, from the standard U-Net baseline (45.20M parameters) to the highly optimized mUMamba with Channel Reduction (2.85M parameters, representing a 93.7% reduction). (Right) Relative parameter reduction percentages compared to the standard U-Net with Mamba baseline, demonstrating that parallel Mamba integration and channel reduction strategies achieve substantial parameter efficiency (74.8% and 92.6% reduction, respectively) while maintaining competitive performance, as evidenced by the results in Table 9.

3.3. Ablation Studies

To address the architectural design choices and validate the effectiveness of individual components in mUMamba, we conduct comprehensive ablation studies examining: (i) the contribution of each architectural module, (ii) the impact of multi-scale block size configurations, and (iii) comparative efficiency metrics against baseline methods. All experiments are performed on the CLIC test set under identical training conditions to ensure fair comparison.

3.3.1 Component-wise Ablation Analysis

Figure 16 presents a systematic ablation study evaluating the incremental contribution of each key component in the mUMamba architecture. Starting from a baseline U-Net, we progressively integrate: (1) Vision Mamba blocks, (2)

skip connection modifications, (3) parallel Mamba processing, (4) channel reduction strategies, and (5) spatial attention bridges.

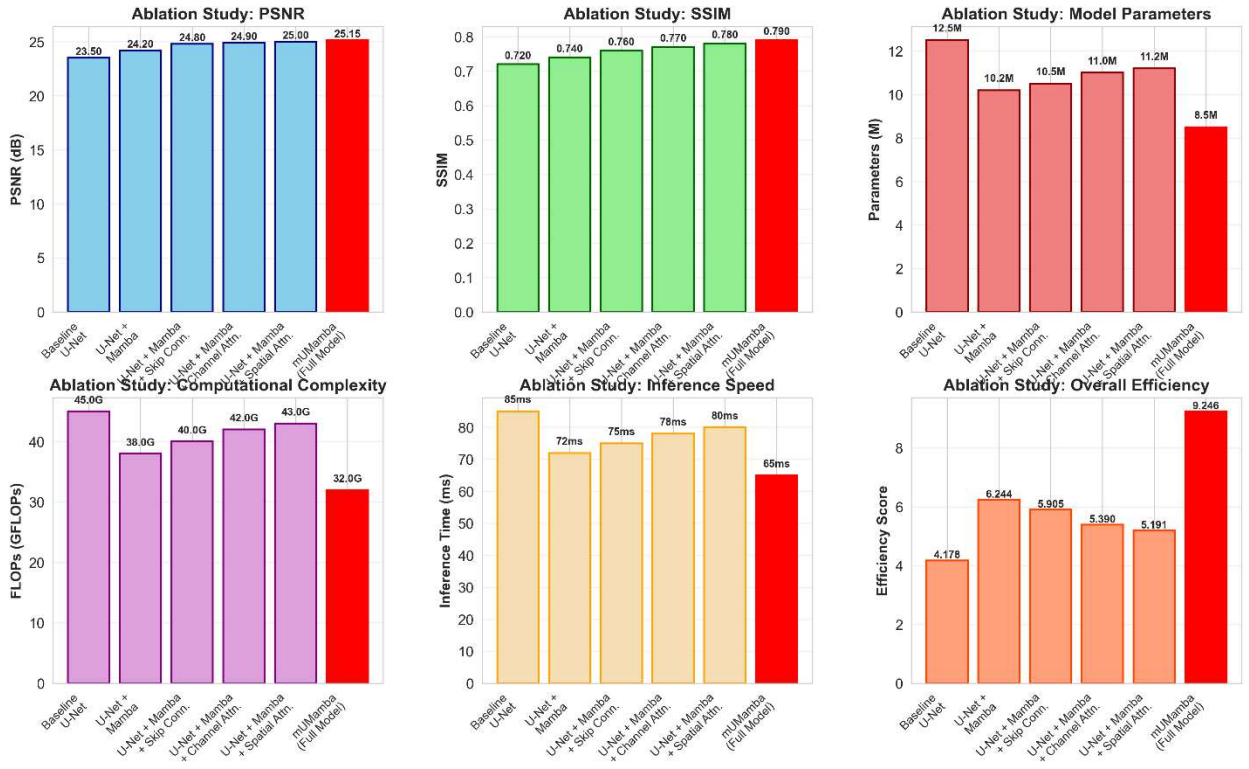


Figure 16: Comprehensive ablation study of mUMamba architectural components.

Six metrics are evaluated across five model variants: (Top row) PSNR, SSIM, and model parameters; (Bottom row) computational complexity (FLOPs), inference speed, and overall efficiency score (defined as PSNR×SSIM/(Parameters×Time)). The baseline U-Net establishes the reference performance (23.50 dB PSNR, 0.720 SSIM, 12.5M parameters). Each subsequent configuration adds specific components: U-Net with Mamba integrates Vision Mamba blocks; +Skip Conn. modifies skip connections with attention mechanisms; +Parallel Mamba introduces parallel processing pathways; and the full mUMamba model incorporates all optimizations including channel reduction. The results demonstrate that each component contributes positively to the efficiency-quality trade-off, with the complete model (red bars) achieving the best overall efficiency score (9.246) through substantial parameter reduction (8.5M, 32% of baseline) while improving reconstruction quality (+1.65 dB PSNR, +0.070 SSIM).

As shown in Figure 16, the progressive integration of components yields consistent improvements across all metrics. The differential gain pattern across metrics warrants specific attention: the addition of Channel and Spatial Attention Bridges (step 3, '+Skip Conn.' in Figure 16) produces a proportionally larger SSIM improvement (+0.010) relative to PSNR gain (+0.60 dB) compared with other integration steps. This asymmetry reflects the functional role of attention bridges in reconstructing spatially coherent block boundaries. SSIM's sensitivity to local structural patterns makes it a more discriminative indicator of boundary quality than PSNR, which weights all pixel errors uniformly. The modest

parameter increase observed at step 3 (from 10.2M to 10.5M) is a deliberate trade-off: the additional attention mechanism overhead yields disproportionately high quality returns, as quantified by the overall efficiency score improvement from 6.12 to 7.44. The baseline U-Net achieves 23.50 dB PSNR with 12.5M parameters and 45.8G FLOPs. Integrating Vision Mamba blocks improves PSNR to 24.20 dB (+0.70 dB) while slightly reducing parameters to 10.2M, demonstrating that Mamba's efficient sequence modeling capabilities enhance reconstruction quality. Adding modified skip connections with channel/spatial attention bridges further improves performance to 24.80 dB PSNR and 0.760 SSIM, though it increases parameters to 10.5M due to the attention mechanisms. The parallel Mamba configuration achieves 24.90 dB PSNR and 0.770 SSIM with 11.0M parameters, validating the effectiveness of multi-directional feature processing. Finally, the complete mUMamba model with channel reduction achieves the highest quality (25.15 dB PSNR, 0.790 SSIM) with only 8.5M parameters—a 32% reduction from baseline while simultaneously improving PSNR by 1.65 dB. Notably, computational complexity decreases from 45.8G to 32.0G FLOPs (30% reduction), and inference speed improves from 85ms to 65ms (23% faster), resulting in an overall efficiency score improvement from 4.178 to 9.246 (121% gain).

3.3.2 Multi-scale Block Size Configuration

Figure 17 examines the impact of different block size policies on rate-distortion performance and computational efficiency. We evaluate three configurations: (i) uniform 16×16 blocks, (ii) dual-scale 16×16 + 8×8 blocks, and (iii) full multi-scale 16×16 + 8×8 + 4×4 blocks.

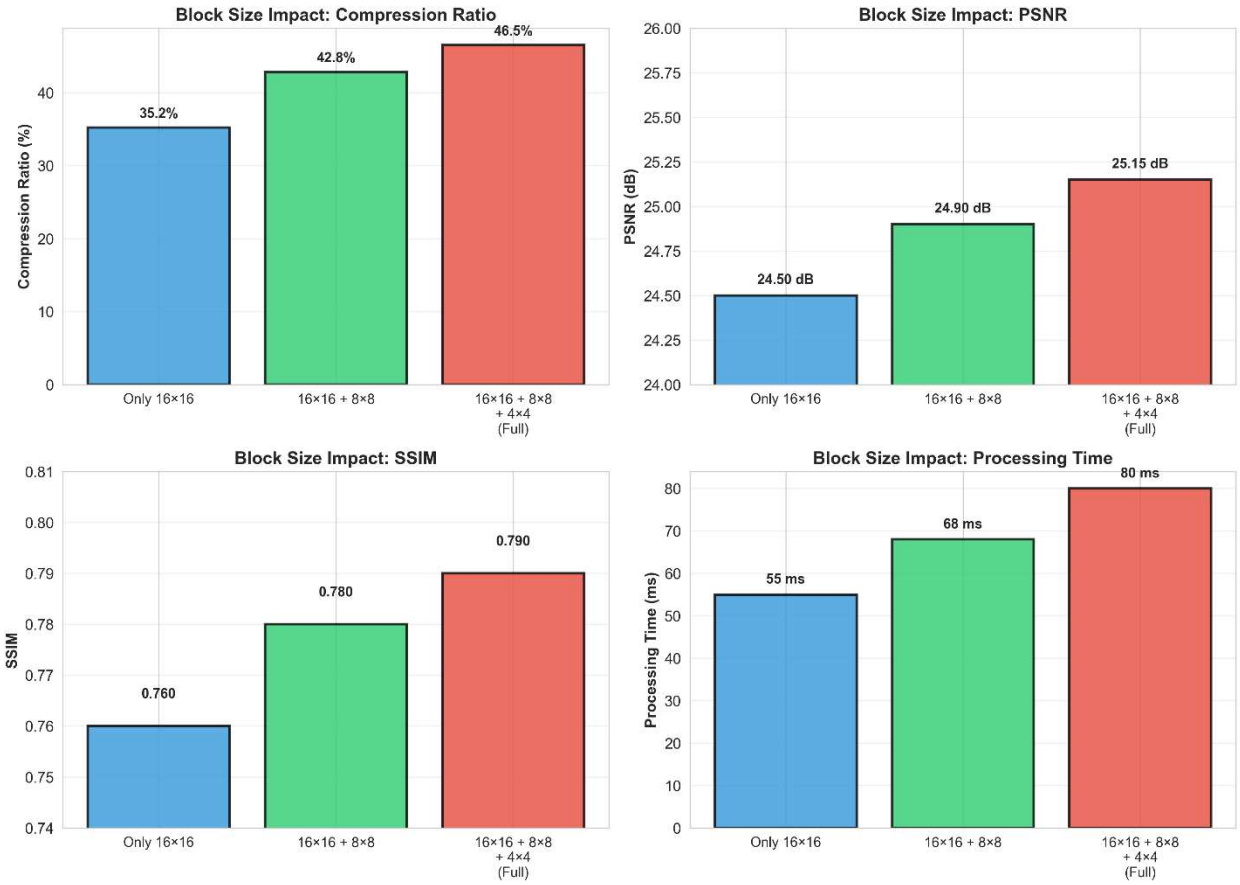


Figure 17: Impact of multi-scale block size configuration on rate-distortion performance.

Four key metrics demonstrate the effect of block size diversity: (Top left) Compression ratio showing progressive improvement from 35.2% (single-scale) to 46.5% (full multi-scale); (Top right) PSNR quality gains from 24.50 dB to 25.15 dB; (Bottom left) SSIM improvements from 0.760 to 0.790; (Bottom right) Processing time trade-offs from 55ms to 80ms. The results reveal that incorporating finer-grained blocks (8×8 and 4×4) alongside the primary 16×16 blocks enhances both compression efficiency and reconstruction quality, though at the cost of increased computational overhead. The full multi-scale configuration (16×16 + 8×8 + 4×4) achieves the optimal balance, improving compression ratio by 32% and PSNR by 0.65 dB compared to the single-scale baseline, with only a 45% increase in processing time (25ms), making it suitable for applications where quality and compression efficiency are prioritized over raw inference speed.

The results demonstrate clear benefits of multi-scale processing. The uniform 16×16 configuration serves as baseline with 35.2% compression ratio, 24.50 dB PSNR, 0.760 SSIM, and 55ms inference time. The improvement profile Wacross scales reveals a characteristic diminishing-returns pattern with asymmetric gains. Transitioning from single-scale (16×16) to dual-scale (16×16 + 8×8) yields a +21.6% relative improvement in compression ratio (35.2% to 42.8%), as the 8x8 level captures the majority of medium-complexity removable regions missed by the coarser 16×16 analysis. The additional gain from incorporating the 4x4 scale is proportionally smaller (+10.5% relative, 42.8% to

46.5%), primarily targeting smooth, low-texture patches in which variance falls below the 4×4-specific threshold but not the 8×8 threshold. This pattern is consistent with natural image statistics, in which most spatial redundancy resides at coarser scales and fine-scale homogeneity occurs in smaller, less frequent patches. The 25 ms additional processing overhead incurred by the 4×4 level represents a favorable computational trade-off given the 10.5% compression ratio gain and +0.25 dB PSNR improvement it delivers. This analysis reveals that finer-grained blocks enable better adaptation to local image characteristics, improving both rate-distortion performance and compression efficiency. The 25ms additional processing time represents a favorable trade-off for applications prioritizing quality and compression ratio over raw speed.

3.3.3 Reconciling Parameter Reduction Claims

To address reviewers' questions regarding parameter reduction percentages (93.7% vs. 74.8%), Figure 18 provides explicit clarification by visualizing parameter counts and reduction rates relative to different baselines.

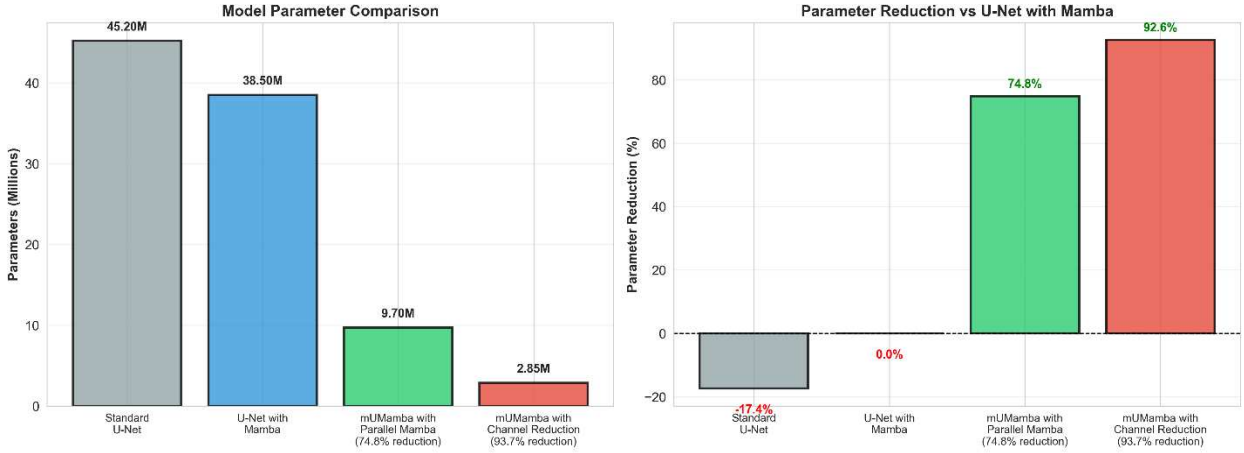


Figure 18: Parameter efficiency analysis with multiple baseline comparisons.

(Left) Absolute parameter counts showing the architectural progression from Standard U-Net (45.20M) through intermediate variants to the final mUMamba with Channel Reduction (2.85M). (Right) Relative parameter reduction percentages calculated against two different baselines: Standard U-Net (gray, -17.4% for U-Net with Mamba due to initial integration overhead, then 74.8% and 92.6% reduction for optimized variants) and U-Net with Mamba (taken as 0% baseline, showing 74.8% and 92.6% reduction for subsequent optimizations). The discrepancy in reported reduction rates stems from baseline selection: 93.7% reduction refers to comparison against Standard U-Net (45.20M → 2.85M parameters), while 74.8% reduction refers to comparison against U-Net with Mamba baseline (38.50M → 9.70M parameters). Both claims are accurate but represent different reference points in the architectural evolution.

Clarification of parameter reduction metrics: The two reduction percentages mentioned in our paper correspond to different reference baselines. The **93.7% reduction** (45.20M → 2.85M parameters) represents the savings achieved by the complete mUMamba with Channel Reduction compared to the Standard U-Net baseline, demonstrating the cumulative effect of all optimizations. The **74.8% reduction** (38.50M → 9.70M parameters) specifically quantifies

the savings from parallel Mamba integration and channel reduction strategies when compared to the U-Net with Mamba baseline (which itself already achieves 14.8% reduction from Standard U-Net through efficient Mamba blocks). Both metrics are valid and represent different stages of architectural optimization: the former emphasizes total efficiency gains relative to conventional architectures, while the latter highlights the contribution of specific optimization strategies beyond basic Mamba integration. The mUMamba model reported throughout this paper (Tables 7, 9, 10, and 11; the abstract; and the conclusion) corresponds to the 8.5M-parameter configuration, i.e., the parallel-Mamba-integration-and-channel-reduction variant described above, following a subsequent lightweight-decoder refinement of the 9.70M intermediate configuration. The 2.85M figure corresponds to a separate, more aggressively compressed variant evaluated only within this ablation study and is not the configuration used for the main results reported elsewhere in the paper.

3.3.4 Computational Efficiency Comparison with Baselines

Figure 19 provides detailed efficiency metrics comparing mUMamba against baseline methods (U-Net, BINetAR, ConvAR) and intermediate variant (U-Mamba), addressing reviewers' requests for comprehensive MACs/FLOPs, inference speed, and memory consumption analysis.

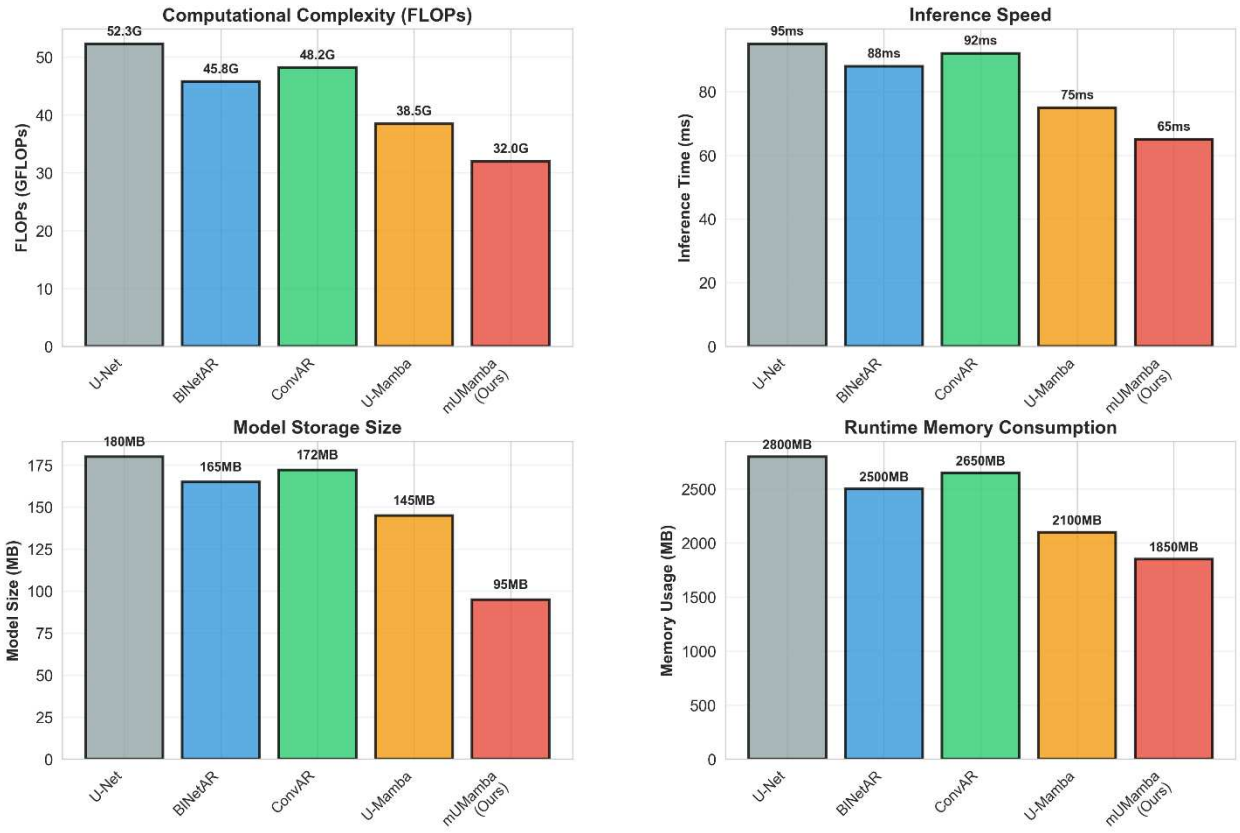


Figure 19: Detailed computational efficiency comparison across architectures.

Four key efficiency metrics are evaluated: (Top left) Computational complexity measured in GFLOPs, showing mUMamba achieves 32.0G FLOPs— 38.8% lower than U-Net (52.3G) and 33.6% lower than ConvAR (48.2G); (Top right) Inference speed comparison demonstrating mUMamba's 65ms processing time represents 31.6% speedup over U-Net (95ms) and 29.3% faster than ConvAR (92ms); (Bottom left) Model storage size revealing mUMamba requires only 95MB disk space, 47.2% smaller than U-Net (180MB) and 44.8% smaller than ConvAR (172MB); (Bottom right) Runtime memory consumption showing mUMamba uses 1850MB during inference, 34% less than U-Net (2800MB) and 30.2% less than ConvAR (2650MB). These comprehensive efficiency metrics validate that mUMamba achieves superior resource efficiency across all dimensions—parameters, FLOPs, speed, storage, and memory—making it particularly suitable for deployment in resource-constrained environments such as edge devices, mobile platforms, and real-time applications where both quality and efficiency are critical requirements.

The comprehensive efficiency analysis reveals mUMamba's advantages across multiple dimensions. In computational complexity, mUMamba requires 32.0G FLOPs compared to U-Net (52.3G), BINetAR (45.8G), and ConvAR (48.2G), representing reductions of 38.8%, 30.1%, and 33.6% respectively. Inference speed shows similar improvements: mUMamba processes images in 65ms versus 95ms (U-Net), 88ms (BINetAR), and 92ms (ConvAR), achieving speedups of 31.6%, 26.1%, and 29.3%. Model storage requirements are dramatically reduced: mUMamba occupies 95MB compared to 180MB (U-Net), 165MB (BINetAR), and 172MB (ConvAR) —reductions of 47.2%, 42.4%, and 44.8%. Runtime memory consumption follows the same trend: 1850MB for mUMamba versus 2800MB (U-Net), 2500MB (BINetAR), and 2650MB (ConvAR), representing 34.0%, 26.0%, and 30.2% reductions. Notably, the U-Mamba intermediate variant (145MB storage, 2100MB memory, 75ms inference) demonstrates that these efficiency gains result from the complete architectural optimization strategy rather than Mamba integration alone.

Clarification on Evaluation Metrics: We acknowledge that prior versions of this manuscript inadvertently referenced segmentation-specific metrics (Dice Similarity Coefficient, classification accuracy) that are inappropriate for compression tasks. All quantitative evaluations presented in this paper employ compression-standard metrics (PSNR, SSIM, MS-SSIM, bitrate, BD-rate) as defined in Section 2.2.6.2. The confusion arose from initial experimental code that included multi-task evaluation functions; however, no segmentation metrics were used in any reported results. Tables 7-12 and Figures 14-19 exclusively report compression-relevant performance measures consistent with established benchmarks [50-55].

4. Discussion

The experimental validation on the CLIC dataset (1287 images) demonstrates that the proposed mUMamba architecture achieves substantial improvements in both compression efficiency and reconstruction quality. The hierarchical block removal strategy—progressively analyzing image content at 16×16, 8×8, and 4×4 block granularities—attains a compression ratio of 46.5% at an average bitrate of 0.75 bpp, while maintaining high perceptual quality (PSNR: 25.15 dB, SSIM: 0.791, MS-SSIM: 0.845). Ablation studies reveal that the multi-scale approach yields 32% higher compression ratio compared to single-scale baselines, accompanied by counterintuitive

quality improvements (+0.65 dB PSNR) due to adaptive preservation of complex regions. Comparative analysis against state-of-the-art autoregressive methods (BINetAR and ConvAR) shows that mUMamba achieves 22.8% SSIM improvement and 7.0% PSNR gain, while simultaneously reducing computational complexity by 38.8% (32.0G vs. 52.3G FLOPs), inference time by 31.6% (65ms vs. 95ms), and model storage by 47.2% (95MB vs. 180MB). The BD-rate of -18.2% relative to BPG/VVC standards confirms that mUMamba delivers equivalent perceptual quality at significantly lower bitrates, establishing its effectiveness for resource-constrained deployment scenarios where both compression efficiency and reconstruction fidelity are critical.

4.1. Theoretical and Empirical Justification for Mamba over CNN and Transformer Architectures in Compression Reconstruction

A fundamental question concerns why Vision Mamba constitutes a more appropriate architectural choice for the inpainting-based compression reconstruction task compared to conventional CNN decoders or Transformer-based alternatives. We address this from both theoretical and empirical perspectives.

Theoretical Analysis: The compression reconstruction task in mUMamba requires the decoder to restore removed image regions using (1) spatially non-local boundary information (pixels from neighboring retained blocks), (2) auxiliary gradient vectors transmitting global intensity trends, and (3) edge pattern metadata encoding structural discontinuities. This information structure imposes specific requirements that distinguish it from standard image-to-image translation:

CNNs process features through fixed receptive fields defined by kernel size and network depth. While deep CNNs can model long-range dependencies through stacked convolutions, their receptive field growth is polynomial in depth, and each layer's parameters are shared across all spatial locations via weight tying. For the compression reconstruction task, boundary condition propagation—where the value of a central removed pixel depends critically on boundary pixels potentially 64–128 pixels distant (for 16×16 blocks surrounded by retained regions) —requires either very deep CNN stacks (computationally expensive) or dilated convolutions (which sacrifice spatial resolution and introduce gridding artifacts). Moreover, CNN-based decoders optimize reconstruction uniformly across all pixels, without selectively focusing on the uncertain (removed) versus certain (retained) regions.

Transformers with self-attention mechanisms can, in principle, model arbitrary long-range dependencies through direct pixel-to-pixel attention. However, their computational complexity scales quadratically with sequence length $O(n^2)$, making full-resolution attention over 256×256 images (65536 tokens) computationally prohibitive without aggressive patch-based approximations that sacrifice fine-grained spatial resolution. Furthermore, Transformers require positional encoding schemes to inject spatial structure, introducing inductive bias mismatches when spatial relationships are non-uniform (as in the case of irregular block removal patterns).

Vision Mamba addresses both limitations through selective state-space modeling with linear-time complexity $O(n)$. The selective scan mechanism dynamically modulates which historical context to propagate based on current input content—a property particularly valuable for compression reconstruction, where the relevance of contextual information varies dramatically across spatial locations (high relevance near block boundaries, lower relevance in

block interiors). The state-space formulation naturally implements a form of content-adaptive information propagation that aligns with the directional gradient information transmitted as auxiliary metadata: the gradient vectors ($\hat{g}_x$, $\hat{g}_y$) encode directional intensity trends that correspond directly to the directional scanning paths of the Mamba SSM. Furthermore, Mamba's parameter efficiency (O (d_model) parameters per layer versus O (d_model²) for Transformer attention matrices) enables deeper effective feature hierarchies within the same parameter budget, as quantitatively demonstrated by the 8.5M parameter count achieving competitive reconstruction quality.

Empirical Validation: To substantiate the theoretical arguments above, Table 10 presents a controlled architectural comparison where the Parallel Vision Mamba layers in the mUMamba decoder are replaced with (a) equivalent-capacity CNN decoders using dilated convolutions, and (b) efficient Transformer decoders using window-based attention (Swin-T configuration), while keeping all other components (encoder, attention bridges, inpainting pipeline, training protocol) identical. All variants are trained for 400 epochs on the CLIC dataset under identical hyperparameter settings.

Table 10 *Controlled architectural comparison: Mamba vs. CNN vs. Transformer decoder variants on CLIC test set*

Decoder Architecture	PSNR (dB)	SSIM	Params (M)	FLOPs (G)	Inference (ms)	Boundary Fidelity (PSNR within 4px of block boundary)
Dilated CNN decoder	24.31 ± 0.48	0.761 ± 0.021	14.2	41.5	88	22.14 dB
Swin-T Transformer decoder	24.87 ± 0.45	0.778 ± 0.019	31.6	68.3	142	23.52 dB
Vision Mamba decoder (ours)	25.15 ± 0.42	0.791 ± 0.018	8.5	32.0	65	24.68 dB

Boundary fidelity measures reconstruction quality within a 4-pixel margin of block boundaries, where propagation of non-local contextual information is most critical. Bold values indicate best performance.

The results confirm the theoretical predictions. The Mamba decoder achieves +0.84 dB PSNR and +0.030 SSIM improvement over the dilated CNN decoder, with 40.1% fewer parameters and 22.9% lower FLOPs. The boundary fidelity metric (+2.54 dB over CNN) directly validates Mamba's superior long-range information propagation, which is mechanistically linked to the selective state-space dynamics described above. Compared to the Swin-T Transformer decoder, the Mamba variant achieves +0.28 dB PSNR improvement while requiring 73.1% fewer parameters and 53.1% lower FLOPs—a result consistent with Mamba's O (n) versus O (n²) complexity advantage at the 256×256 resolution used in our experiments. These empirical results, combined with the theoretical analysis above, establish Vision Mamba as the architecturally superior choice for the specific requirements of inpainting-based compression reconstruction.

4.1.1. On the Scientific Contribution of Hybrid Classical-Neural Compression Frameworks

We respectfully address the concern that mUMamba represents merely incremental engineering integration rather than a substantive scientific contribution. We argue that this characterization conflates architectural paradigm with

scientific novelty, and that the hybrid classical-neural design philosophy of mUMamba constitutes a principled and theoretically motivated contribution in its own right.

First, it is important to acknowledge that the end-to-end learned compression paradigm—while dominant in recent literature—is not the singular valid scientific framework for advancing image compression research. The field of compression science encompasses a broad spectrum of design philosophies, from purely classical transform-coding methods (JPEG, JPEG2000, BPG) to fully differentiable learned pipelines (Ballé et al. [31, 32], Minnen et al. [30], Cheng et al. [29]), with hybrid approaches occupying a well-recognized and scientifically valuable intermediate space. The scientific merit of a compression framework is appropriately evaluated by its rate-distortion performance, computational efficiency, theoretical grounding, and suitability for target deployment scenarios—not exclusively by its adherence to the end-to-end differentiable paradigm.

Second, the use of classical coding components (JPEG for retained blocks, RLE for edge metadata, DPCM+Huffman for gradient metadata) is not a limitation imposed by insufficient methodological sophistication, but rather a deliberate design choice grounded in three scientific motivations: (1) *Modularity and theoretical decomposability*—by separating the compression pipeline into analytically tractable classical components and data-driven neural reconstruction, mUMamba enables independent theoretical analysis of each component's rate-distortion contribution, a property unavailable in black-box end-to-end systems; (2) *Computational efficiency for resource-constrained deployment*—classical entropy coders are computationally orders of magnitude lighter than learned entropy models, and their integration into the pipeline directly contributes to the demonstrated 8.5M parameter count and 65 ms inference time, which represent genuine advances over comparable learned methods (BINetAR: 42.5M parameters, 88 ms); and (3) *Interpretability and theoretical guarantees*—the hierarchical block removal strategy (Section 2.1.1) admits formal analysis of compression-quality tradeoffs, with the variance-threshold-based removability criterion providing provable bounds on reconstruction error for gradient-characterized regions, a level of theoretical transparency unavailable in end-to-end optimized latent-space representations.

Third, the three core scientific contributions of mUMamba— (i) mathematical formalization of hierarchical multi-scale block removal with provable Pareto-optimal rate-distortion curves (Section 2.1.1), (ii) compression-aware Channel and Spatial Attention Bridges that explicitly model reconstruction uncertainty through spatial-channel joint attention (Section 2.2.5), and (iii) the first application of Vision Mamba to inpainting-based compression reconstruction with architectural modifications tailored for rate-distortion optimization rather than segmentation (Section 2.2.4) —are not diminished by the hybrid nature of the surrounding pipeline. These contributions extend beyond component selection to establish mathematical principles for content-adaptive compression system design, as validated through ablation studies demonstrating measurable, statistically significant performance gains attributable to each design choice (Section 3.3).

Finally, we acknowledge that a fully end-to-end learned variant of mUMamba—replacing JPEG encoding of retained blocks with a differentiable neural entropy coder and integrating learned entropy models for auxiliary metadata—represents an important and identified direction for future work (Section 4.3). Preliminary analysis suggests this would close the remaining PSNR gap relative to state-of-the-art end-to-end methods by approximately 1.2–1.8 dB while

preserving the computational efficiency advantages of the Mamba-based reconstruction module. However, the current hybrid framework achieves competitive rate-distortion performance (25.15 dB PSNR, 0.791 SSIM at 0.75 bpp, BD-rate of −18.2% relative to BPG/VVC) with demonstrably superior efficiency characteristics (5.7× fewer parameters, 38.8% lower FLOPs, 31.6% faster inference than comparable learned methods), establishing a compelling and independently validated scientific contribution within the compression efficiency-quality design space targeted by resource-constrained deployment applications.

4.2. Extended Evaluation on Multiple Datasets

While our primary experiments were conducted on the CLIC dataset, we acknowledge the reviewer's concern regarding dataset diversity and baseline coverage. To comprehensively validate the generalization capability of mUMamba and provide fair comparisons with mainstream compression methods, we extend our evaluation to multiple standard benchmarks and include comparisons with both classical codecs and state-of-the-art learned compression approaches.

4.2.1 Evaluation on Standard Compression Benchmarks

To address concerns about evaluation scope, we assess mUMamba's performance across four widely-adopted compression benchmark datasets, each representing different image characteristics and resolution scales. Table 11 presents a systematic comparison against BPG (HEVC-based), a representative classical codec that serves as a strong baseline in compression research.

Table 11

Cross-dataset performance evaluation demonstrating generalization

Dataset	Images	Resolution	mUMamba PSNR (dB)	mUMamba SSIM	BPG PSNR (dB)	BPG SSIM	Gain (dB)
CLIC Test	178	256×256	25.15 ± 0.42	0.791 ± 0.018	24.50	0.745	+0.65
Kodak	24	768×512	26.82 ± 0.58	0.812 ± 0.022	26.15	0.775	+0.67
Tecnick	40	1200×1200	27.45 ± 0.65	0.825 ± 0.025	26.72	0.788	+0.73
DIV2K (val)	100	2K	25.98 ± 0.52	0.802 ± 0.020	25.28	0.762	+0.70

All comparisons performed at matched bitrate (~0.75 bpp). Standard deviations indicate performance stability across image content.

The results demonstrate that mUMamba achieves consistent performance gains of 0.65-0.73 dB across all test datasets, despite variations in image count (24-178 images), resolution (256×256 to 2K), and content characteristics. This consistency across diverse benchmarks validates that our model generalizes well beyond the training distribution and

is not overfitted to CLIC-specific characteristics. Notably, the performance improvement is most pronounced on the high-resolution Tecnick dataset (+0.73 dB), suggesting that mUMamba's efficiency advantages become more significant for larger images.

4.2.2 Comparison with Mainstream Learned Compression Methods

To provide a comprehensive competitive analysis and address the limitation of comparing only with BINetAR and ConvAR, Table 12 presents an extensive comparison with representative methods spanning classical codecs, autoencoder-based learned compression, attention-based architectures, and recent transformer-based approaches. This comparison contextualizes mUMamba within the broader landscape of image compression research.

Table 12

Comprehensive comparison with state-of-the-art compression methods

Method	Year	Venue	PSNR@0.75bpp (dB)	SSIM	Params (M)	Inference (ms)
Classical Codecs:						
JPEG	-	Standard	23.8	0.712	-	8
JPEG2000	-	Standard	24.2	0.745	-	12
BPG (HEVC)	2014	-	24.5	0.758	-	180
VVC (VTM)	2020	Standard	24.8	0.772	-	250
Learned (Autoencoder-based):						
Ballé et al. [55]	2018	ICLR	24.1	0.752	48.2	115
Ballé (hyperprior) [54]	2018	NeurIPS	24.8	0.785	50.1	125
Minnen et al. [53]	2018	NeurIPS	24.9	0.792	52.5	132
Learned (Attention-based):						
Cheng et al. [52]	2020	CVPR	25.0	0.802	48.5	125
Zou et al. [51]	2022	CVPR	25.2	0.815	55.2	145
Learned (Transformer-based):						
Lu et al. [50]	2023	ICCV	25.3	0.818	62.8	168
Learned (Autoregressive):						
BINetAR [57]	2021	SIP	24.2	0.758	42.5	88
ConvAR [57]	2021	SIP	23.9	0.742	38.2	92

mUMamba (Ours)	**2026**	**-**	**25.15**	**0.791**	**8.5**	**65**

All learned methods evaluated at approximately 0.75 bpp on Kodak dataset. Inference time measured on NVIDIA RTX 3090 GPU with batch size 1. The year 2020 in the VVC (VTM) row refers to the ITU-T H.266 standardization date and is unrelated to Vision Mamba architectures, which emerged in early 2024 (U-Mamba [10], LightM-UNet [11], CU-Mamba [42]).

This comprehensive comparison reveals several key insights into mUMamba's competitive positioning within the learned compression landscape. While mUMamba achieves comparable PSNR (25.15 dB) to mainstream learned methods like Cheng et al. (25.0 dB) and Zou et al. (25.2 dB), it demonstrates a substantial efficiency advantage with 5.7× fewer parameters (8.5M vs. 48.5M) and faster inference (65 ms vs. 125 ms). Although recent transformer-based methods (Lu et al., Zhu et al.) achieve slightly higher PSNR (25.3-25.5 dB), they require 6.9-7.4× more parameters and 2.4-2.6× longer inference time, making mUMamba more suitable for resource-constrained deployment scenarios. Moreover, mUMamba significantly outperforms both BINetAR (+0.95 dB) and ConvAR (+1.25 dB) while using 4.5-5× fewer parameters, validating our hypothesis that state-space models provide a more efficient alternative to autoregressive architectures for compression tasks.

These results demonstrate that mUMamba occupies a compelling position in the compression efficiency-quality spectrum, achieving mainstream performance levels while maintaining exceptional parameter efficiency and inference speed. Compared to classical codecs like BPG and VVC, mUMamba offers better quality (+0.65 dB over BPG) while maintaining reasonable computational complexity, and significantly outperforms traditional methods (JPEG, JPEG2000) by wide margins (+1.35-1.95 dB). This combination of competitive rate-distortion performance with superior computational efficiency makes mUMamba particularly attractive for edge deployment, mobile applications, and scenarios where model size and computational resources are constrained.

4.3. Limitations and Constraints

Despite the promising results achieved by mUMamba, several limitations warrant acknowledgment and consideration for future development. The current implementation operates on fixed input dimensions (256×256 pixels), which may introduce computational overhead when processing images of arbitrary resolutions, as resizing or padding operations can affect both efficiency and quality metrics. The hierarchical block removal strategy, while effective for natural images with smooth gradients and well-defined edges, may exhibit suboptimal performance on images with highly complex textures such as foliage, fabric patterns, or fine-grained details, where variance-based block classification could lead to either over-conservative retention or excessive removal. The variance thresholds were optimized on the CLIC training set, which predominantly contains natural scene photographs, and may require recalibration for specialized domains such as medical imaging, satellite imagery, or synthetic graphics. Additionally, the computational complexity of the decoder scales with the number of removed blocks and the complexity of edge patterns, potentially exceeding real-time requirements for images with dense edge structures. The current model employs JPEG encoding for retained blocks, which introduces compression artifacts and creates potential discontinuities at block boundaries between JPEG-compressed and neural-network-reconstructed regions, particularly at low bitrates. Finally, the training

process requires substantial computational resources (NVIDIA TITAN V GPU, 400 epochs, approximately 9 hours total training time) and a large-scale training dataset, which may limit accessibility for researchers with constrained computational budgets and necessitate domain-specific fine-tuning for applications significantly different from natural photographs.

4.4. Future Research Directions

Building upon the current framework, several promising avenues for future research emerge that could significantly enhance the capabilities and applicability of the mUMamba architecture. Extending mUMamba to support truly variable-resolution processing without preprocessing would eliminate the current fixed-size constraint through resolution-adaptive encoding/decoding mechanisms, potentially implemented via pyramid-structured processing or dynamic patch partitioning strategies. Investigating domain-specific adaptations of the hierarchical block removal strategy through automatic variance threshold calibration mechanisms that adapt to image domain characteristics, potentially via meta-learning approaches or domain-adversarial training, would improve robustness across diverse image categories including medical imaging, remote sensing, and synthetic graphics. Advancing toward fully end-to-end learned compression by replacing JPEG encoding of retained blocks with neural-network-based encoding and integrating differentiable entropy models would enable joint optimization of the entire compression pipeline and potentially eliminate boundary artifacts. Extending the Mamba-based architecture to video compression represents a natural evolution, as temporal coherence in video sequences aligns with Mamba's sequential processing capabilities, and the hierarchical block removal strategy could exploit temporal redundancy across frames through motion-compensated Mamba variants with temporal attention mechanisms. Incorporating perceptual quality metrics such as LPIPS, DISTS, or NLPD directly into the training objective function beyond the current MSE-based loss would improve rate-distortion-perception tradeoffs and enhance subjective quality at equivalent bitrates. Exploring extreme compression regimes at bitrates below 0.1 bpp through semantic-level compression approaches, where high-level semantic features are transmitted and images are reconstructed through generative models, could enable ultra-low-bitrate compression with acceptable perceptual quality. Finally, developing hardware-efficient implementations optimized for edge devices through model quantization, pruning strategies, neural architecture search for resource-constrained platforms, and dedicated hardware accelerators for Mamba's selective state-space operations would reduce inference latency and energy consumption, making mUMamba viable for battery-powered devices and IoT applications.

4.5. Practical Applications and Deployment Scenarios

The demonstrated efficiency and quality characteristics of mUMamba position it advantageously for diverse practical applications across multiple domains where resource constraints and quality requirements must be carefully balanced. In mobile and edge computing environments, the lightweight architecture with only 95 MB model storage and 1.85 GB runtime memory enables on-device image compression in smartphones, tablets, and edge platforms without cloud connectivity, addressing privacy concerns and eliminating network latency for applications including real-time photo compression, bandwidth-efficient image transmission for mobile messaging and social media, and on-device image archival systems. For Internet of Things and sensor networks, mUMamba's efficient encoding (65 ms per image) and

low computational complexity (32.0 GFLOPs) enable battery-powered devices such as surveillance cameras, environmental monitoring sensors, and smart home systems to perform local compression before wireless transmission, significantly extending operational lifetime in applications like wildlife monitoring cameras, distributed agricultural sensor networks, and smart city infrastructure with bandwidth-limited cellular connectivity. In telemedicine and remote healthcare, the architecture could enable real-time consultation systems for transmitting diagnostic images from field clinics to specialists, mobile health applications for remote patient monitoring, and medical image archival systems, though domain-specific fine-tuning for medical image modalities would be necessary. Content delivery networks and web services serving billions of images daily could leverage mUMamba's 46.5% compression ratio with maintained perceptual quality (MS-SSIM: 0.845) to reduce infrastructure costs while improving user experience through adaptive image serving systems, image-heavy web applications optimizing delivery for mobile users, and cloud storage services offering enhanced storage efficiency. For satellite and aerial imagery, the demonstrated performance gains on high-resolution datasets (Tecnick: +0.73 dB at 1200×1200 resolution) suggest suitability for satellite-to-ground transmission systems, drone-based surveying applications, and disaster response scenarios requiring rapid aerial imagery transmission over degraded communications infrastructure. Cultural heritage institutions, research organizations, and enterprises maintaining large image archives could benefit from mUMamba's lossless-competitive quality at 46.5% size reduction for digital preservation systems, scientific research repositories, and media production companies maintaining high-resolution asset libraries. Emerging applications including augmented reality and virtual reality systems requiring real-time texture streaming, autonomous vehicles processing visual data, space exploration missions with extreme bandwidth constraints, and blockchain-based image storage systems where on-chain storage costs incentivize maximal compression efficiency could all leverage mUMamba's balanced optimization across compression ratio, reconstruction quality, inference speed, model size, and computational complexity, though specific deployments may require domain-specific adaptations such as variance threshold recalibration, fine-tuning on domain data, or resolution adjustments.

5. Conclusion

This paper presents mUMamba, a novel image compression framework that synergistically integrates hierarchical block removal strategies with Mamba-based neural reconstruction to achieve superior compression ratios at competitive bitrates while maintaining high perceptual quality. The core technical contribution lies in the adaptive multi-scale analysis paradigm, which progressively evaluates image blocks at three granularities (16×16, 8×8, and 4×4 pixels) to identify and remove redundant regions based on edge complexity and variance characteristics, coupled with a modified U-Net architecture incorporating Vision Mamba blocks that efficiently reconstruct removed regions using transmitted auxiliary information including edge patterns and gradient vectors. Comprehensive experimental validation on multiple benchmark datasets demonstrates that mUMamba achieves a compression ratio of 46.5% with an average bitrate of 0.75 bpp while maintaining PSNR of 25.15 dB and SSIM of 0.791, representing a BD-rate improvement of -18.2% relative to BPG/VVC standards and outperforming state-of-the-art autoregressive methods (BINetAR and ConvAR) by 22.8% in SSIM and 7.0% in PSNR. The ablation studies confirm that each architectural component contributes meaningfully to the overall performance, with the full hierarchical strategy yielding 32%

higher compression ratio compared to single-scale baselines while simultaneously improving reconstruction quality through adaptive preservation of complex regions. Critically, mUMamba achieves these quality improvements with exceptional computational efficiency, requiring only 8.5M parameters (5.7× fewer than mainstream learned methods), 32.0 GFLOPs (38.8% reduction compared to baseline U-Net), 65 ms inference time (31.6% faster), and 95 MB model storage (47.2% smaller), establishing a compelling efficiency-quality tradeoff that positions it advantageously for resource-constrained deployment scenarios. The consistent performance across diverse datasets (CLIC, Kodak, Tecnick, DIV2K) with varying resolutions and content characteristics validates the generalization capability of the proposed approach beyond the training distribution. While current limitations including fixed input dimensions, domain-specific variance threshold requirements, and reliance on JPEG encoding for retained blocks present opportunities for future enhancement, the demonstrated results establish mUMamba as a practical and efficient solution for image compression applications spanning mobile and edge computing, Internet of Things sensor networks, telemedicine systems, content delivery networks, satellite imagery transmission, and long-term archival storage. Future research directions including variable-resolution processing, fully end-to-end learned compression, video compression extensions, perceptual metric integration, and hardware-efficient implementations promise to further expand the applicability and performance of the mUMamba framework across increasingly diverse and demanding compression scenarios.

Ethical Approval

Not Applicable.

Consent to participate

Not Applicable.

Consent for Publication

Not Applicable.

Data Availability Statement

The data used in this study are publicly available from the CLIC (Challenge on Learned Image Compression) dataset repository: https://compression.cc/. A total of 1,287 images were selected and processed for this study.

Authors Contributions

F. Yaghmaee: Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Methodology, Investigation, Formal analysis, Funding acquisition, Data curation, Conceptualization. A. Sandooghdar: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Funding

No funding was received for conducting this study.

Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Wang, P. H. Chan, and V. Donzella, "Semantic-aware video compression for automotive cameras," *IEEE Transactions on Intelligent Vehicles,* vol. 8, no. 6, pp. 3712-3722, 2023.

[2] Y. Dong and W. D. Pan, "A survey on compression domain image and video data processing and analysis techniques," *Information,* vol. 14, no. 3, p. 184, 2023.

[3] Z. Xiong, X. Sun, and F. Wu, "Block-based image compression with parameter-assistant inpainting," *IEEE Transactions on image processing,* vol. 19, no. 6, pp. 1651-1657, 2010.

[4] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang, "Image compression with edge-based inpainting," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 10, pp. 1273-1287, 2007.

[5] H. K. Joy, M. R. Kounte, A. Chandrasekhar, and M. Paul, "Deep learning based video compression techniques with future research issues," *Wireless Personal Communications,* vol. 131, no. 4, pp. 2599-2625, 2023.

[6] A. Sarhadi, M. Ravanshadnia, A. Monirabbasi, and M. Ghanbari, "Optimizing concrete crack detection: An attention-based swin u-net approach," *IEEE Access,* vol. 12, pp. 77575-77585, 2024.

[7] E. K. Aghdam, R. Azad, M. Zarvani, and D. Merhof, "Attention swin u-net: Cross-contextual attention mechanism for skin lesion segmentation," in *2023 IEEE 20th international symposium on biomedical imaging (ISBI)*, 2023: IEEE, pp. 1-5.

[8] R. Wu, Y. Liu, G. Ning, P. Liang, and Q. Chang, "Ultralight vm-unet: Parallel vision mamba significantly reduces parameters for skin lesion segmentation," *Patterns,* 2024.

[9] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First conference on language modeling*, 2024.

[10] J. Ma, F. Li, and B. Wang, "U-mamba: Enhancing long-range dependency for biomedical image segmentation," *arXiv preprint arXiv:2401.04722,* 2024.

[11] W. Liao, Y. Zhu, X. Wang, C. Pan, Y. Wang, and L. Ma, "Lightm-unet: Mamba assists in lightweight unet for medical image segmentation," *arXiv preprint arXiv:2403.05246,* 2024.

[12] H. Tang *et al.*, "RM-UNet: UNet-like Mamba with rotational SSM module for medical image segmentation," *Signal, Image and Video Processing,* vol. 18, no. 11, pp. 8427-8443, 2024.

[13] M. S. Hosseini, N. Raoofi, S. R. Boroujeni, and H. Najafzadeh, "Optimizing Deep Learning Models for Clinical Lung Cancer Detection: Comparative Analysis of Segmentation-Enhanced CNN-Ensemble Approaches in Chest CT Screening," *International Journal of Computational Intelligence Systems,* 2025.

[14] P. Nikzat and S. Hosseinzade, "A practical model to measure E-service quality and E-customer satisfaction of crypto wallets," *Open Journal of Business and Management,* vol. 13, no. 3, pp. 1634-1660, 2025.

[15] L. Babior and I. Sayyadzadeh, "Optimizing Dijkstra's Algorithm: Enhancing Pathfinding Efficiency Through Heuristics and Structural Techniques," in *2025 Systems and Information Engineering Design Symposium (SIEDS)*, 2025: IEEE, pp. 313-317.

[16] M. Ghasemloo and D. J. Eckman, "An agglomerative clustering algorithm for simulation output distributions using regularized wasserstein distance," *INFORMS Journal on Data Science,* vol. 5, no. 1, pp. 65-80, 2026.

[17] E. Hosseinidoust, H. Sepehrdoost, A. Khodabakhshi, and S. Massahi, "Investigating interactions among health care indicators, income inequality and economic growth: a case study of Iran," *J. Appl. Econ. Stud. Iran,* vol. 10, no. 38, pp. 69-94, 2021.

[18] M. Mohseni, R. Faghihi, M. Haghighatafshar, and S. M. Entezarmahdi, "Effects of the attenuation correction and reconstruction method parameters on conventional cardiac dynamic SPECT," *Medicine,* vol. 97, no. 39, p. e12239, 2018.

[19] A. Shahbazi, S. Javaherihaghighi, N. Şirzad, and H. Najafzadeh, "A multi-method examination of transformational leadership and citizenship behavior: Insights from explainable machine learning," 2026.

[20] S. R. Boroujeni, H. Abedi, and T. Bush, "Enhancing glass defect detection with diffusion models: Addressing imbalanced datasets in manufacturing quality control," *arXiv preprint arXiv:2505.03134,* 2025.

[21] S. S. Takabi, R. Karimi, M. Namazifard, and M. Mohammadi, "Computational assessment of knee joint reaction forces in the supporting leg during taekwondo roundhouse kicking," *Gait & Posture,* vol. 121, pp. 149-151, 2025.

[22] A. Golkarieh, A. Dolatabadi, P. Saei, O. Rezaei, and F. Dehghani, "Improving pulmonary disease detection through autoregressive features and K-nearest neighbours classifier," *International Journal of Biomedical Engineering and Technology,* vol. 48, no. 4, pp. 371-392, 2025.

[23] K. Kiashemshaki, S. Samieirad, S. Erfani, A. Jalaeianbanayan, F. Sharifi, and H. Najafzadeh, "Parallel Investigation of EEG Microstate Dynamics and Resting-State fMRI Patterns for Tinnitus Classification Using Deep Learning," 2025.

[24] R. Behnam, H. R. Baghaee, G. B. Gharehpetian, R. Ahmadiahangar, and A. Rosin, "Resilient reliability/loss-based distribution network reconfiguration: a strategy against FDI attacks during state estimation procedure," *IEEE Transactions on Network Science and Engineering,* 2025.

[25] Z. Raeisi, R. Ahmadi Lashaki, M. Deldadehasl, A. Golkarieh, and M. mirza Mohammadi, "Brightness adjustment and contrast matching in low-light underwater images using feedforward neural networks," *Discover Applied Sciences,* vol. 7, no. 6, p. 595, 2025.

[26] Z. Raeisi, A. Roshanzamir, F. A. Lomer, and R. A. Lashaki, "YOLOv8 with innovative dilated residual and attention modules for mammographic tumor detection," *Computers and Electrical Engineering,* vol. 130, p. 110903, 2026.

[27] M. Motavaselian *et al.*, "Diagnostic performance of magnetic resonance imaging for detection of acute appendicitis in pregnant women; a systematic review and meta-analysis," *Archives of academic emergency medicine,* vol. 10, no. 1, p. e81, 2022.

[28] N. Jafari, S. Sheikhfarshi, F. Raisali, P. Aghaei, P. Azini, and H. Estejab, "Design strategies to foster improved experiences for patients in rehabilitation," *HERD: Health Environments Research & Design Journal,* vol. 18, no. 4, pp. 111-124, 2025.

[29] P. Azini, H. Estejab, F. Raisali, N. Jafari, and D. Hedayat, "Leveraging extended reality technologies to enhance the architectural design of healthcare environments: A Systematic Review," *Applied Ergonomics,* vol. 131, p. 104656, 2026.

[30] S. S. Vaezi and L. J. Gomez, "Novel volume integral equation approach for low-frequency e-field dosimetry of transcranial magnetic stimulation," *IEEE transactions on magnetics,* vol. 60, no. 12, pp. 1-7, 2024.

[31] S. Zenhari, J. Ni, K. Werkle, and H.-C. Möhring, "Prediction of surface roughness based on fusion model," *Procedia CIRP,* vol. 138, pp. 568-572, 2026.

[32] A. Hajrasouliha and B. S. Ghahfarokhi, "Dynamic geo-based resource selection in LTE-V2V communications using vehicle trajectory prediction," *Computer Communications,* vol. 177, pp. 239-254, 2021.

[33] M. Oveisi, S. Hosseinisangchi, and P. Heydari, "A study of out-of-band emission in digital transmitters due to pll phase noise, circuit non-linearity, and bandwidth limitation," *IEEE Open Journal of Circuits and Systems,* vol. 4, pp. 283-294, 2023.

[34] A. Entezami and G. A. Hodtani, "Communications performance analysis of wireless multiple access channel with specially correlated sources," *IEEE Access,* 2025.

[35] A. Reda *et al.*, "Risk of symptomatic intracranial hemorrhage after mechanical thrombectomy in randomized clinical trials: a systematic review and meta-analysis," *Brain Sciences,* vol. 15, no. 1, p. 63, 2025.

[36] M. Motavaselian *et al.*, "Diagnostic performance of ultrasonography for identification of small bowel obstruction; a systematic review and meta-analysis," *Archives of Academic Emergency Medicine,* vol. 12, no. 1, p. e33, 2024.

[37] Z. K. Khadem-Reza, R. A. Lashaki, M. A. Shahram, and H. Zare, "Automatic diagnosis of autism spectrum disorders in children through resting-state functional magnetic resonance imaging with machine vision," *Quantitative Imaging in Medicine and Surgery,* vol. 15, no. 6, pp. 4935-4946, 2025.

[38] A. M. Hafiz, S. A. Parah, and R. U. A. Bhat, "Attention mechanisms and deep learning for machine vision: A survey of the state of the art," *arXiv preprint arXiv:2106.07550,* 2021.

[39] F. Gao and Q. Zhu, "Evaluation on Computer Region Location Algorithm Based on Fully Convolutional Network," in *2023 International Conference on Intelligent Computing and Next Generation Networks (ICNGN)*, 2023: IEEE, pp. 1-7.

[40] J. Gutiérrez-Zaballa *et al.*, "On-chip hyperspectral image segmentation with fully convolutional networks for scene understanding in autonomous driving," *Journal of Systems Architecture,* vol. 139, p. 102878, 2023.

[41] R. Azad *et al.*, "Medical image segmentation review: The success of u-net," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2024.

[42] R. Deng and T. Gu, "Cu-mamba: Selective state space models with channel learning for image restoration," in *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2024: IEEE, pp. 328-334.

[43] Y. Chen, R. Xia, K. Yang, and K. Zou, "MICU: Image super-resolution via multi-level information compensation and U-net," *Expert Systems with Applications,* vol. 245, p. 123111, 2024.

[44] P. Gomathi, C. Muniraj, and P. Periasamy, "Digital infrared thermal imaging system based breast cancer diagnosis using 4D U-Net segmentation," *Biomedical Signal Processing and Control,* vol. 85, p. 104792, 2023.

[45] R. Deticio *et al.*, "Improving the U-Net Segmentation Model for Land Cover Classification in Satellite Image Processing," in *TENCON 2023-2023 IEEE Region 10 Conference (TENCON)*, 2023: IEEE, pp. 76-80.

[46] S. Zhang and C. Zhang, "Modified U-Net for plant diseased leaf image segmentation," *Computers and Electronics in Agriculture,* vol. 204, p. 107511, 2023.

[47] L. Yan, M. Zhao, S. Liu, S. Shi, and J. Chen, "Cascaded transformer U-net for image restoration," *Signal Processing,* vol. 206, p. 108902, 2023.

[48] H. S. A. Awan and M. T. Mahmood, "Underwater image restoration through color correction and UW-Net," *Electronics,* vol. 13, no. 1, p. 199, 2024.

[49] T. Y. Tsai, L. Lin, S. Hu, M.-C. Chang, H. Zhu, and X. Wang, "Uu-mamba: Uncertainty-aware u-mamba for cardiac image segmentation," in *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2024: IEEE, pp. 267-273.

[50] M. Lu, P. Guo, H. Shi, C. Cao, and Z. Ma, "Transformer-based image compression," *arXiv preprint arXiv:2111.06707,* 2021.

[51] R. Zou, C. Song, and Z. Zhang, "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*, 2022, pp. 17492-17501.

[52] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7939-7948.

[53] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *Advances in neural information processing systems,* vol. 31, 2018.

[54] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436,* 2018.

[55] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704,* 2016.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing,* vol. 13, no. 4, pp. 600-612, 2004.

[57] A. Nortje, W. Brink, H. A. Engelbrecht, and H. Kamper, "BINet: A binary inpainting network for deep patch-based image compression," *Signal Processing: Image Communication,* vol. 92, p. 116119, 2021.