International Journal of Computer Vision (2026) 134:374
https://doi.org/10.1007/s11263-026-02953-6

Ultra-Lightweight Adaptive Bitrate Deep Video Compression

Ali Hojjat[1,2] · **Janek Haberer**[1] · **Olaf Landsiedel**[1,2,3]

Received: 22 January 2026 / Accepted: 2 July 2026

Abstract

The rapid growth of camera-based IoT devices demands the need for efficient video compression, particularly for edge applications where devices face hardware constraints, often with only 1 or 2 MB of RAM and unstable internet connections. Traditional and deep video compression methods are designed for high-end hardware, exceeding the capabilities of these constrained devices. Consequently, video compression in these scenarios is often limited to M-JPEG due to its high hardware efficiency and low5 complexity. This paper introduces `MCUCoder`, an open-source adaptive bitrate video compression model tailored for resource-limited IoT settings. `MCUCoder` features an ultra-lightweight encoder with only 10.5K parameters and a minimal 350KB memory footprint, making it well-suited for edge devices and MCUs. While `MCUCoder` uses a similar amount of energy as M-JPEG, it reduces bitrate by 55.65% on the MCL-JCV dataset and 55.59% on the UVG dataset, measured in MS-SSIM. `MCUCoder` supports adaptive-bitrate transmission by generating a latent representation sorted by importance, allowing the transmitted data to be adjusted according to available bandwidth. We further show that `MCUCoder` compression outperforms M-JPEG compression when evaluated on downstream AI tasks, including image classification, object detection, and image captioning. Source code available at https://github.com/ds-kiel/MCUCoder.

Keywords Deep video compression · Internet of things · Adaptive bitrate encoding

1 Introduction

Motivation: The number of camera-based IoTs devices using always-on MCU is growing rapidly, reaching tens of billions (Lin et al., 2020). These devices are widely used in applications such as surveillance cameras (Hu et al., 2020; Josephson et al., 2019; Naderiparizi et al., 2018), wearable cameras (Veluri et al., 2023), robotics (Nakanoya et al., 2023), wildlife monitoring (Iyer et al., 2020), road monitoring (Hojjat et al., 2024), and smart farming (Koh et al., 2021). Typically, they capture raw frames through a camera sensor, encode them, and transmit the compressed version to a server via the Internet for further processing, including human observation or AI tasks such as object detection and classification (Hojjat et al., 2024). Therefore, a video encoder is necessary to efficiently compress the captured frames before transmission. However, in IoT environments, there are two primary limitations: constrained hardware resources and limited communication bandwidth.

Limited Hardware: Although traditional video codecs like H.264 (Wiegand et al., 2003), H.265 (Sullivan et al., 2012), and the newer H.266 (Bross et al., 2021) provide excellent performance, they demand significant hardware for extracting the intra and inter-frame correlations. For example, H.265 encoding involves highly computationally intensive tasks such as motion estimation with sub-pixel accuracy, Rate Distortion Optimization (RDO) for choosing optimal intra-prediction modes, and Context Adaptive Binary Arithmetic Coding (CABAC) for entropy coding. Additionally, a single video frame at 224 × 224 resolution

Communicated by Svetlana Lazebnik.

✉ Ali Hojjat
ali.hojjat@cs.uni-kiel.de

Janek Haberer
janek.haberer@cs.uni-kiel.de

Olaf Landsiedel
olaf.landsiedel@tuhh.de

[1] Department of Computer Science, Kiel University, Kiel, Germany

[2] Institute for Networked Cyber-Physical Systems, Hamburg University of Technology (TUHH), Hamburg, Germany

[3] United Nations University Hub on Engineering to Face Climate Change at the Hamburg University of Technology, United Nations University Institute for Water, Environment and Health (UNU-INWEH), Hamburg, Germany

Published online: 30 July 2026

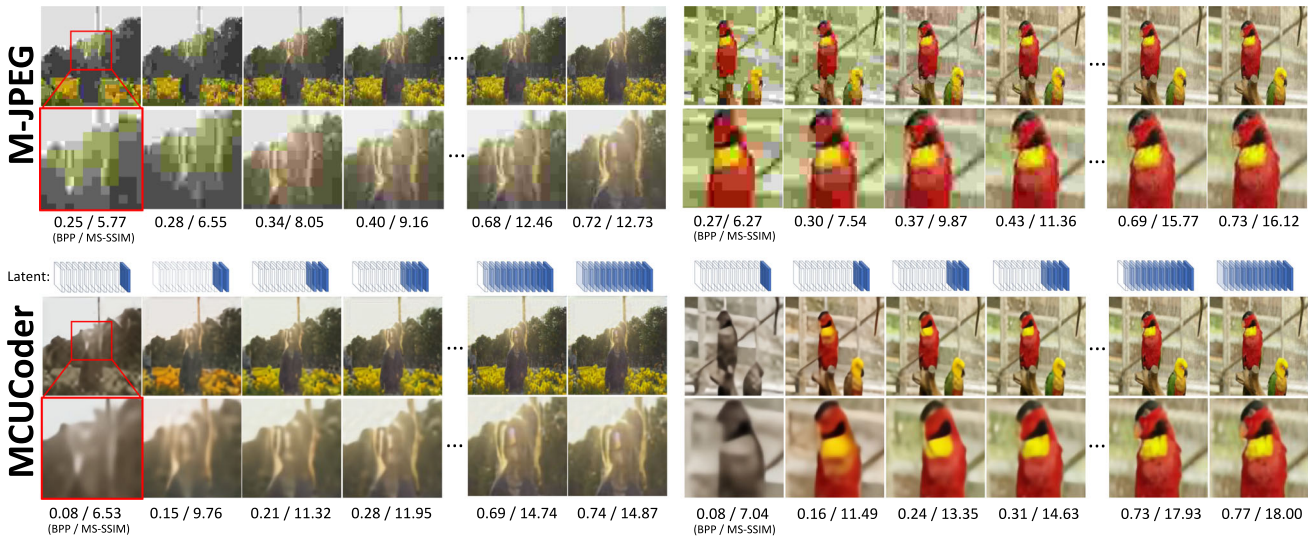


Fig. 1 Qualitative comparison of `MCUCoder` and M-JPEG across various compression rates on two videos from the MCL-JCV (Wang et al., 2016) and UVG (Mercat et al., 2020) datasets. As we can see, `MCUCoder` offers a significantly better trade-off between MS-SSIM and bit per pixel (bpp). For instance, at 0.15 bpp in the left example, with `MCUCoder` we can see the person's face whereas with M-JPEG we need at least 0.34 bpp to make out the face. Note that the images in each column do not necessarily have the same bitrate. More examples are reported in Figure 16

requires about 150 KB of RAM, which is a lot for the low-cost, low-energy MCUs used in IoT devices that typically have only **1–2 MB of RAM**. Consequently, inter-frame compression or any other kind of multi-frame analysis is not practically feasible on such constrained devices. Similarly, while Neural Networks (NNs) and AI-based compression methods outperform traditional models (Agustsson et al., 2020; Lu et al., 2019), they also often require considerable RAM and GPU resources. For instance, just storing a model with 1M parameters requires around 4 MB of RAM; see Fig. 3. As a result, in such settings, devices are typically limited to using Motion-JPEG (M-JPEG) (Pennebaker & Mitchell, 1992), a video compression format where each frame is compressed individually as a JPEG image, which is efficient and hardware-friendly.

Limited Internet: Many IoT devices are located in remote areas where Internet connection is weak and unstable, making it necessary for the encoder to have an **Adaptive Bitrate Encoding** that can generate video streams with varying bitrate. This feature allows the encoder to adjust the transmitted quality according to the available bandwidth, which is useful in bandwidth-constrained IoT applications such as remote monitoring. However, implementing an adaptive bitrate encoder adds complexity, as it requires mechanisms to prioritize bit stream information based on its impact on frame quality (e.g., Peak Signal to Noise Ratio (PSNR) or MS-SSIM), which is challenging for constrained devices.

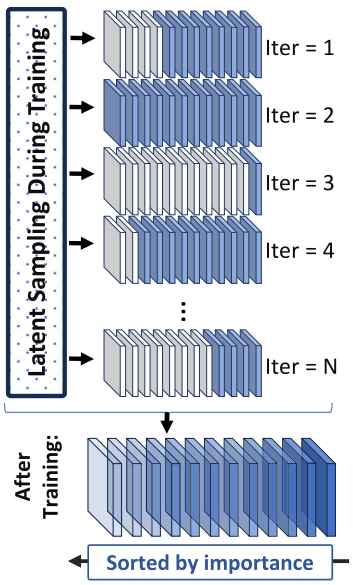


Fig. 2 Stochastic dropout training

Approach: To address these challenges, we introduce `MCUCoder`, an adaptive bitrate deep video compression codec tailored for resource-limited IoT devices. Our approach focuses on creating an "asymmetric" compression model that features an ultra-lightweight encoder designed to be both computationally efficient and memory-friendly. Moreover, to produce an "adaptive" bitstream, we train the `MCUCoder`'s latent representation with *stochastic channel dropout*, see Figure 2. This procedure sorts the N latent feature maps by

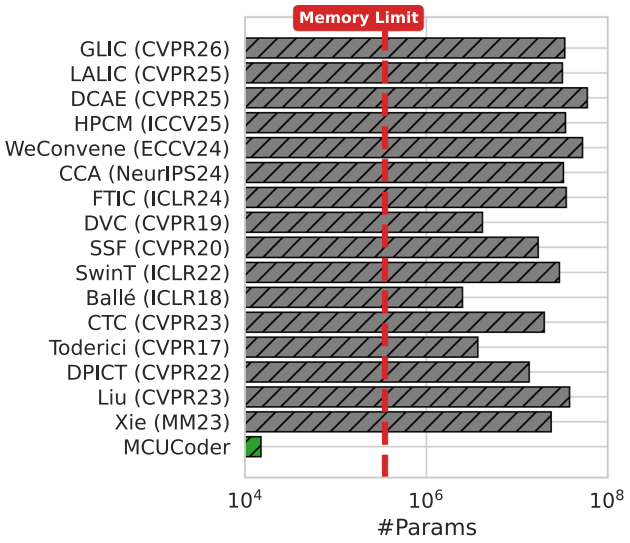


Fig. 3 Number of parameters of `MCUCoder` and other learned image compression (Ballé et al., 2018; Chen et al., 2026; Feng et al., 2025; Fu et al., 2024; Han et al., 2024; Jeon et al., 2023; Lee et al., 2022; Li et al., 2024, 2025; Liu et al., 2023; Lu et al., 2025; Toderici et al., 2017; Xie et al., 2021; Zhu et al., 2022) and video compression models (Agustsson et al., 2020; Lu et al., 2019)

their contribution to reconstruction quality, thereby forcing the model to produce a *progressive bitstream*: early channels in the latent carry the most salient information, and later channels refine the result.

At inference time, the MCU transmits only the first $k \leq N$ channels that fit the instantaneous bandwidth; the decoder then reconstructs the frame from this subset (see Fig. 1). This approach is beneficial for low-power MCUs since it shifts the complexity of identifying important data to the training phase rather than the inference phase. Also, by employing stochastic dropout training, the decoder can reconstruct the frame even with partial data availability, which is essential for maintaining smooth and uninterrupted video transmission in streaming applications, where network conditions can vary. Additionally, `MCUCoder`'s encoder is INT8 quantized, allowing it to utilize Digital Signal Processor (DSP) and CMSIS-NN (ARM-software, 2024) accelerators for faster processing and reduced power consumption.

Contributions:

1. `MCUCoder` has an ultra-lightweight encoder with only 10.5K parameters and a minimal memory footprint of roughly 350KB RAM on nRF5340 and STM32F7 MCUs, making it suitable for such low-resource IoT devices.
2. `MCUCoder` has an energy-efficient INT8 quantized encoder, which leverages the MCU's DSP and CMSIS-NN accelerators to achieve JPEG-level energy efficiency. Compared to its main baseline, M-JPEG, it saves 55.65% overall bit rate on the MCL-JCV dataset and 55.59% on the UVG dataset, measured in MS-SSIM.
3. `MCUCoder` produces a progressive bitstream that enables adaptive-bitrate transmission under varying network conditions
4. `MCUCoder` outperforms M-JPEG when evaluated on downstream AI tasks, including image classification, object detection, and image captioning, highlighting its suitability for intelligent edge vision applications.

2 Related Work

Traditional and NN based video compression: Video compression is a field that has been evolving for decades. Beyond traditional codecs like H.264 (Wiegand et al., 2003), H.265 (Sullivan et al., 2012), and H.266 (Bross et al., 2021), deep learning-based approaches often replace conventional modules such as motion compensation (Agustsson et al., 2020; Yang et al., 2020), transform coding (Gao et al., 2021; Zhu et al., 2022), and entropy coding (Mentzer et al., 2022; Xiang et al., 2023). Also, some work has been done regarding the end-to-end optimization of video compression models (He et al., 2020; Khani et al., 2021; Van Rozendaal et al., 2021). Lu et al. (2019) introduce DVC, the first end-to-end deep video compression model. Hu et al. (2022, 2021) extend DVC to operate in both pixel and feature domains. Li et al. (2021) and Liu et al. (2020) reduce bitrates by modeling probabilities over video frames using conditional coding. Also, in recent years, there has been growing interest in using implicit neural representations for video compression (Chen et al., 2021; Kwan et al., 2023). However, due to their substantial hardware requirements, these models are unsuitable for deployment on low-resource IoT devices. In Figure 3, we compare the number of parameters of `MCUCoder` with several recent learned video compression baselines. The results show that these models have substantially larger parameter counts than `MCUCoder`, exceeding the memory budget of typical MCU platforms.

Video compression for IoT: We can categorize IoT-based video encoders into two parts: hardware-based and software-based. Hardware approaches primarily focus on designing more power-efficient camera sensors (Bejarano-Carbo et al., 2022; Ji et al., 2016; Morishita et al., 2021) and more efficient MCU circuits and processors (Lefebvre et al., 2021; Rossi et al., 2021; Xu et al., 2020). Due to its simplicity, scalability, low latency, and very low energy consumption, the most common software-based video encoder on IoT devices is M-JPEG (Pennebaker & Mitchell, 1992). Nevertheless, there have been few works exploring alternative software-based models: Veluri et al. (2023) employ M-JPEG on the encoder to capture black-and-white and colorized frames at two different resolutions and uses super-resolution methods to interpolate and colorize frames on the decoder. However, unlike `MCUCoder`, it is not adaptive and relies on a JPEG

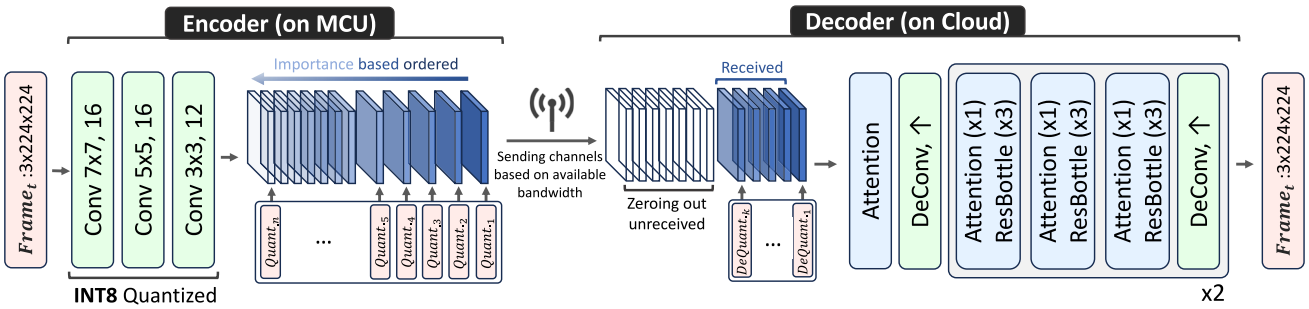


Fig. 4 Overview of `MCUCoder` architecture. The encoder compresses the input frame into a sorted latent space. Afterward, channels are independently quantized and transmitted based on available bandwidth. The decoder reconstructs the frame by zeroing out missing channels

encoder on MCUs. Hu et al. (2020) propose a deep image encoder for MCUs under unreliable network conditions with packet loss. However, their method is non-adaptive in bitrate. Moreover, it processes the image patch by patch, which increases encoding time and limits its practicality for video compression on highly constrained devices. LimitNet (Hojjat et al., 2024) proposes an object-based image compression for downstream AI tasks. It transmits only task-relevant objects and discards the background. In contrast, `MCUCoder` reconstructs the full frame and is designed for Human Perception by optimizing reconstruction metrics such as PSNR and MS-SSIM.

Overall, `MCUCoder` combines the advantages of both worlds: it offers the adaptive bitrate feature of more complex encoders, while maintaining the efficiency necessary for low-resource devices, making it a suitable solution for IoT video compression.

3 `MCUCoder`

In this section, we introduce `MCUCoder`, an adaptive bitrate asymmetric video compression model, specifically designed for IoT settings. We begin by detailing the asymmetric encoder-decoder architecture of `MCUCoder`, including the customized quantization processes. Then, we present the stochastic dropout training method, which trains the encoder of `MCUCoder` to store information in its channels based on importance.

3.1 Asymmetric Compression

MCUs are characterized by highly constrained hardware resources, such as limited RAM, CPU, FLASH, and power availability. Additionally, existing MCU-specific NN frameworks like TFLite Micro support only a limited set of NN layers (Hu et al., 2020). To address these constraints, we propose an asymmetric (Yao et al., 2020) encoder-decoder architecture optimized for constrained devices. Due to hardware constraints, `MCUCoder` encodes each frame independently, as inter-frame compression is not feasible. The encoder contains only 10.5K parameters, while the decoder utilizes approximately 3M parameters and leverages SOTA image decompression blocks; see Fig. 4. The encoding process begins by passing input frame f_t through three convolutional layers. To maximize the data range for subsequent quantization, no activation function is applied in the

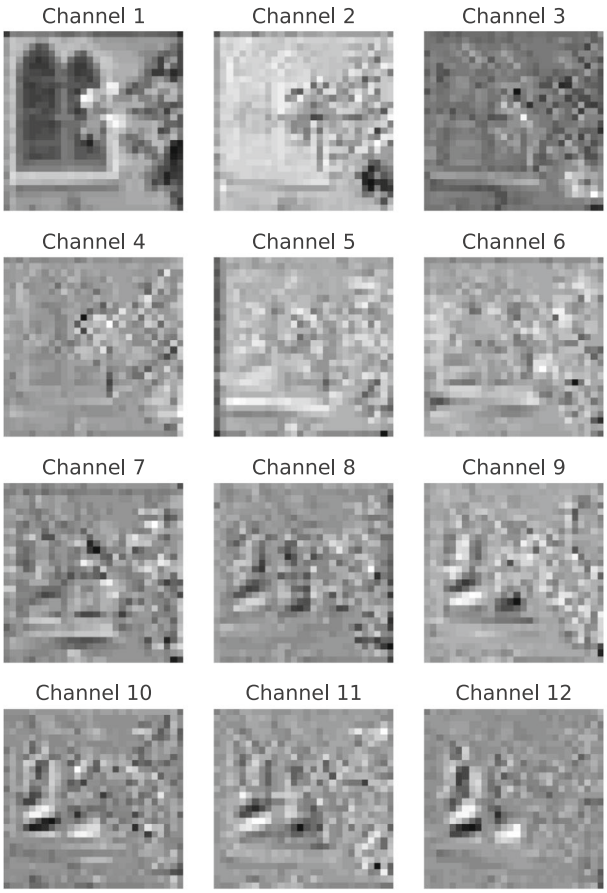


Fig. 5 `MCUCoder` latent channels: Early channels (important ones) capture low-frequency features, while later channels capture high-frequency features, similar to the DCT in JPEG

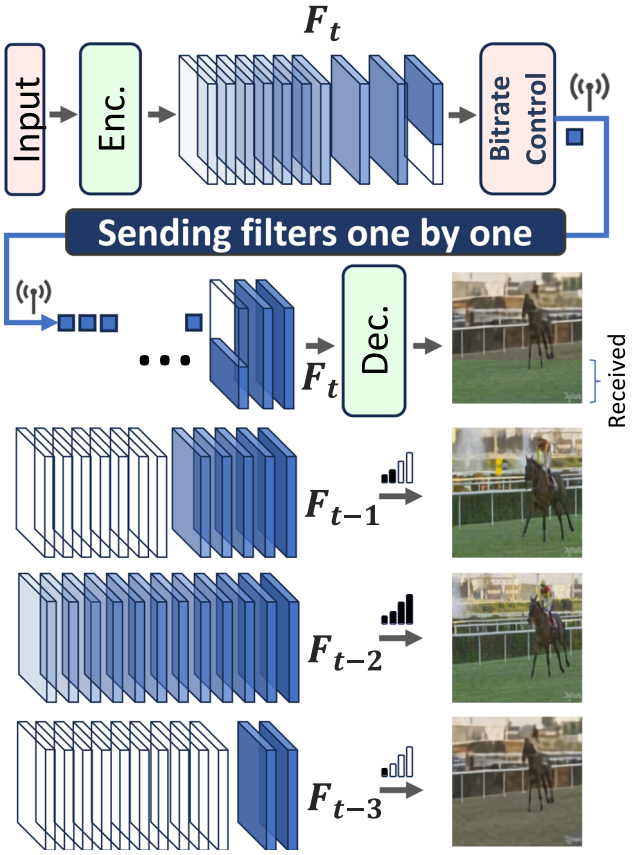


Fig. 6 An example of `MCUCoder` bitrate adaptation under dynamic network bandwidth, where the bitrate control module acts as a gate to determine the number of channels to send

final encoder layer, avoiding the negative truncation caused by ReLU. Afterward, each channel of the latent is quantized into INT8 individually, followed by a further reduction to 5-bit precision to enhance compression efficiency. For the decoder, inspired by He et al. (2022), we integrate a combination of attention blocks (Cheng et al., 2020) and residual bottleneck blocks (He et al., 2016) to reconstruct the frame; see Fig. 4.

3.2 Stochastic dropout training

Bitrate adaptation is a feature that typically introduces additional complexity to the encoding process, which can be challenging to implement on MCUs due to resource constraints. In the literature, dropout (Srivastava et al., 2014) serves as a powerful tool for enhancing generalization in NNs. Building on this insight, we employ a "biased" version of dropout to train `MCUCoder` in a way that instead of random dropping, it drops from the tail of the latent (Haberer et al., 2024; Hojjat et al., 2023). Specifically, on each iteration, after the encoder E gets the input frame f_t, it generates the latent representation z_N, where N is the number of the channels of the latent. Afterward, from a uniform distribution, denoted as $\mathcal{U}_{(0,1)}$, it generates a number, denoted as k, and drops (zero out) the last $N - \lfloor k \times N \rfloor$ channels from z_N. As a result, instead of z_N, the decoder D gets $z_{[0:\lfloor k \times N \rfloor]}$, fills the missing channels with zero, and then reconstructs the output.

$$f_t \rightarrow E(f_t) \rightarrow z_N \xrightarrow{k \sim \mathcal{U}(0,1)} z_{[0:\lfloor k \times N \rfloor]} \rightarrow D(z_{[0:\lfloor k \times N \rfloor]}) \rightarrow \hat{f}_t \tag{1}$$

This tailored version of dropout biases the training to prioritize the earlier channels over the later ones. Consequently, the encoder learns to encode more critical information (low frequency) in the initial feature maps and less important (high frequency) details in the subsequent ones; see Fig. 2. This prioritization enables flexible bitrate adaptation: upon encoding each frame, the encoder starts transmitting the most significant channels first. Depending on the available bandwidth, the bitrate control module determines how many channels need to be sent to the decoder under the current bandwidth budget; see Fig. 6. Importantly, because the latent features are pre-ordered by significance, the bitrate control module basically acts like a simple gate and does not add any extra computational complexity to the encoder.

3.3 Loss Functions

We train separate models using MS-SSIM, which emphasizes multi-scale structural fidelity, MSE, which focuses on pixel-level reconstruction accuracy, and a VGG-based perceptual loss, which preserves semantically meaningful features for downstream AI tasks. Each loss function can be selected based on the requirements of the target application. Finally, we incorporated entropy-aware regularization into the loss function to encourage compact latent representations and improve compression efficiency.

Human Perception: To optimize reconstruction quality under the constraints of IoT-oriented video compression, we train `MCUCoder` with perceptual distortion objectives that emphasize either structural fidelity or pixel-level accuracy. Specifically, we investigate MS-SSIM and Mean Squared Error (MSE) as training objectives. MS-SSIM emphasizes the preservation of structural information across multiple spatial scales, which is particularly important at low bitrates where maintaining scene layout and object boundaries is more critical than reproducing fine-grained details. In contrast, MSE enforces pixel-wise accuracy and aligns with traditional rate-distortion optimization. We train `MCUCoder` using each loss function and report their respective performance in Sect. 4.

Perceptual Loss: Beyond pixel-domain reconstruction quality, we consider *compression for AI-oriented applications*, where reconstructed frames are primarily consumed

Table 1 BD-rate results (Quantized). The anchor is M-JPEG

Type	Dataset	MS-SSIM	PSNR
Video	MCL-JCV	-55.65%	-47.39%
	UVG	-55.59%	-35.28%
Image	KODAK	-55.75%	-43.01%
	CLIC	-49.54%	-38.02%

by downstream machine perception models. To this end, we employ a Visual Geometry Group (VGG)-based perceptual loss, which optimizes the discrepancy between intermediate feature representations of a pretrained VGG network (Simonyan & Zisserman, 2014) for the original frame f_t and its reconstruction $\hat{f}_t$. This feature-space objective encourages the preservation of semantic information that is more relevant for neural inference than raw pixel similarity. The perceptual loss is defined as:

$$\mathcal{L}_{\text{VGG}} = \sum_{l \in \mathcal{L}} \left\| \phi_l(f_t) - \phi_l(\hat{f}_t) \right\|_2^2 \tag{2}$$

where $\phi_l(\cdot)$ denotes the feature map extracted at layer l of a pretrained VGG network and $\mathcal{L}$ denotes the set of selected layers spanning multiple network depths. In `MCUCoder`, we compute the loss using feature representations from early to mid-level convolutional layers, capturing both low-level visual patterns and higher-level semantic cues.

Training the model directly from scratch using only the VGG perceptual loss is insufficient, as feature-space supervision alone provides weak constraints on low-level image structure and leads to unstable optimization (Hojjat et al., 2024; Hu et al., 2020). We therefore adopt a two-stage training strategy. In the first stage, the model is trained using MS-SSIM to enforce structural fidelity and stable reconstruction at low bitrates, and subsequently fine-tuned using the VGG perceptual loss to optimize the learned representation for Perceptual Loss. Results are reported in Section 4.5.

Entropy-aware latent regularization: To further improve compression efficiency, inspired by Ballé et al. (2016), Ballé et al. (2015), we incorporate an entropy-aware regularization term that encourages more compressible latent representations. The motivation is to bias the encoder toward producing latent activations with lower estimated coding cost. To do so, the latent values are modeled with a Gaussian distribution, and their negative log-likelihood is used as an estimate of the coding cost. This cost is converted to bits and normalized to obtain the proxy bits-per-pixel term. This regularization operates only during training and complements the reconstruction objective with a bitrate proxy. Therefore, the training objective is defined as

Table 2 Resource demands of `MCUCoder` on nRF5340 and STM32F7 MCU

	nRF5340	STM32F7
Exec (ms)	1,969	237
RAM (KB)	344 (67%)	360 (36%)
Flash (KB)	100 (10%)	107 (5%)

Table 3 `MCUCoder` complexity in terms of parameters and GFLOPs

Module	Parameters	GFLOPs
Encoder	10.5K	0.72
Decoder	3M	130

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \, \hat{R}(z), \tag{3}$$

where $\mathcal{L}_{\text{rec}}$ denotes the reconstruction loss (e.g., MS-SSIM, MSE, or VGG), z is the encoder latent representation, $\hat{R}(z)$ is the differentiable proxy rate term, and λ controls the strength of the regularization. Since $\hat{R}(z)$ is differentiable, its gradients are backpropagated to the encoder together with the reconstruction loss. This term is used only during training and therefore does not introduce additional computational overhead on the encoder during inference.

4 Evaluation

We train `MCUCoder` on the 300K largest ImageNet images (Deng et al., 2009) and apply noise-downsampling preprocessing (Ballé et al., 2018; He et al., 2021). We use Adam with an initial learning rate of 10^{-4} and a batch size of 16, and train for 1M iterations, lowering the learning rate to 10^{-5} in the final 50K iterations (He et al., 2022). To address quantization effects, we add random noise to the latent. We also quantize inputs, weights, and activations to INT8 for RAM efficiency and to leverage DSP and CMSIS-NN accelerators (ARM-software, 2024) in MCUs. We use post-training quantization available in TensorFlow Lite Micro ((TensorFlow, 2023)) to reduce latency, processing power, and model size with minimal degradation in model accuracy. For all comparisons, we report performance metrics for both the FLOAT32 and INT8 models.

4.1 Quantitative results

Due to the limited hardware resources of MCUs, inter-frame compression is not practically feasible. As a result, in such devices, video compression is limited to M-JPEG where each frame is compressed independently. Therefore, in addition

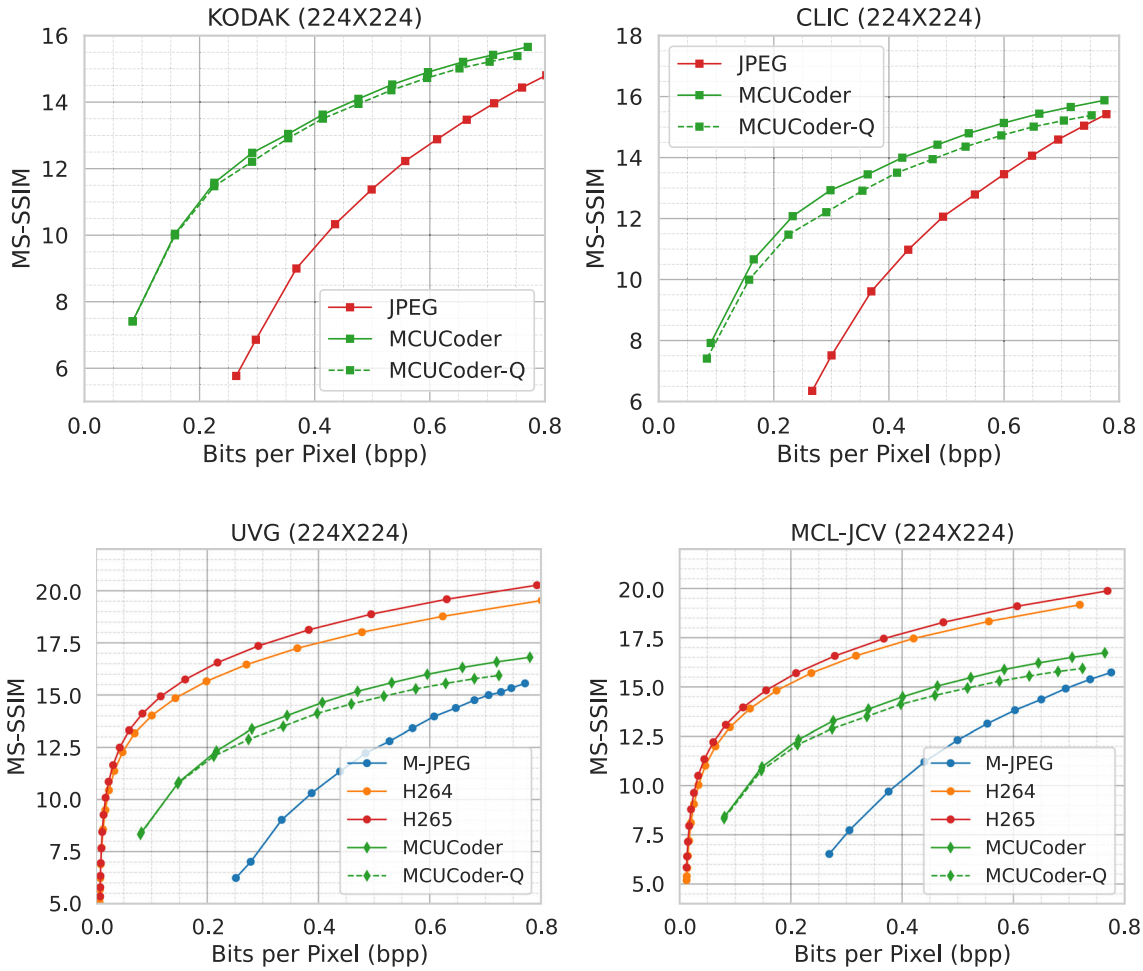


Fig. 7 Comparison of `MCUCoder` (quantized and non-quantized model) and baselines on the image (KODAK (Eastman Kodak, 1993), CLIC Workshop and Challenge on Learned Image Compression (2020)) and video (MCL-JCV (Wang et al., 2016), UVG (Mercat et al., 2020)) compression datasets. For context, we also compare with H.264 and H.265 on video datasets, despite being impractical for MCUs due to high hardware demands. All datasets are resized to 224 × 224

to evaluating `MCUCoder` and its baselines from the perspective of video compression, we also assess its performance on image compression datasets.

Input size: All datasets are evaluated at a resolution of 224 × 224 as `MCUCoder` targets highly constrained battery-powered IoT devices with very limited on-chip memory. For example, common MCU platforms provide only a small amount of available RAM, such as 512 KB on the nRF5340 and 1 MB on the STM32F7. A single raw RGB frame at 224 × 224 resolution requires about 224 × 224 × 3 ≈ 150 KB of memory before any intermediate activations, buffers, or model parameters are allocated. Therefore, increasing the input resolution substantially increases the memory footprint and quickly becomes infeasible on such devices. This limitation becomes more severe for high-resolution inputs. For instance, a single raw RGB frame at 1920 × 1080 resolution requires approximately 1920 × 1080 × 3 ≈ 6.2 MB of memory, which already exceeds the total available RAM of typical target MCU platforms. As a result, such devices cannot store even one uncompressed full-HD frame in memory, let alone execute a neural compression model with additional feature maps and runtime buffers. The 224 × 224 setting is therefore chosen as a practical operating point that reflects the memory constraints of the target embedded platforms while still allowing meaningful evaluation across datasets.

Video compression: We evaluate `MCUCoder` (trained with MS-SSIM loss), on the UVG (Mercat et al., 2020) and MCL-JCV (Wang et al., 2016) datasets, comparing its performance to M-JPEG, see Fig. 7. For additional context, we also include comparisons with traditional video codecs such as H.264 (Wiegand et al., 2003) and H.265 (Sullivan et al., 2012). However, these codecs are not practical deployment targets for the highly constrained MCUs considered in this work. In particular, H.264 and H.265 rely on compu-

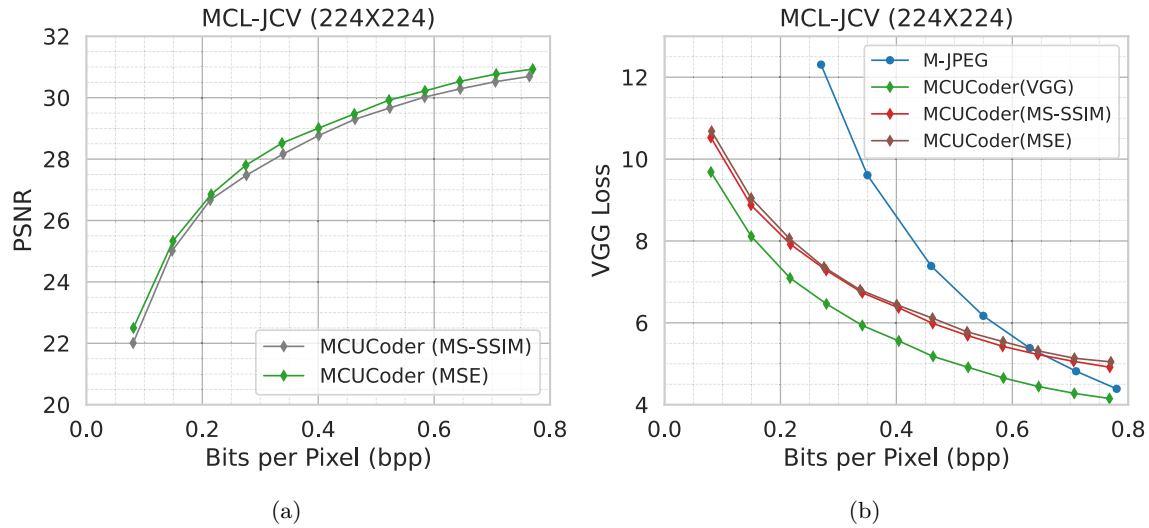


Fig. 8 **(a)** Comparison of PSNR for `MCUCoder` trained with MSE and MS-SSIM losses. Training with MSE results in higher PSNR, indicating improved pixel-level reconstruction quality compared to MS-SSIM. **(b)** Comparison of VGG perceptual loss (AI-oriented metric) for `MCUCoder` trained with MSE, MS-SSIM, and VGG loss. The model trained with the VGG loss achieves the lowest feature-space distortion, indicating better preservation of semantic information and improved suitability for downstream AI tasks

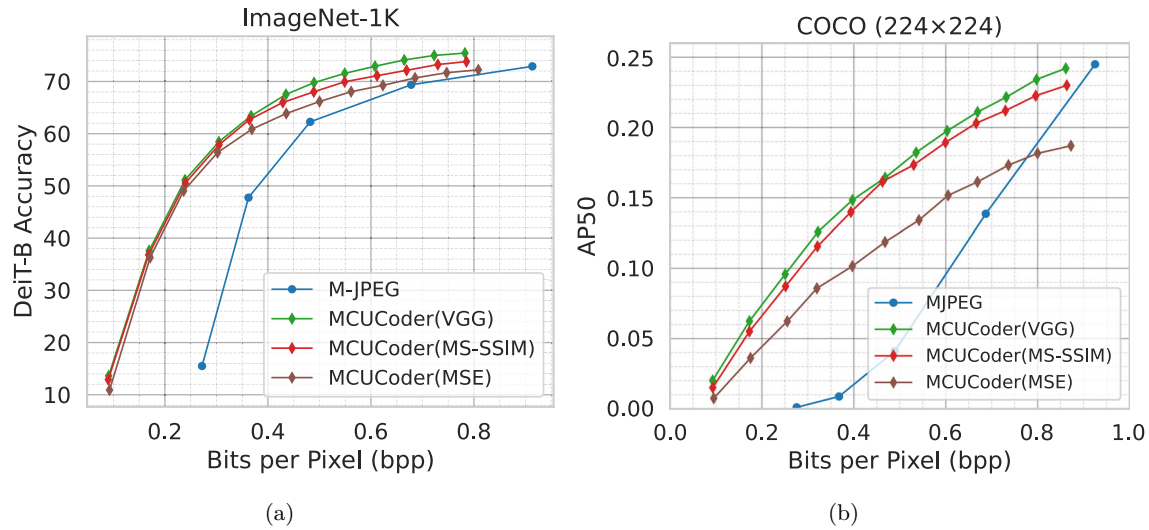
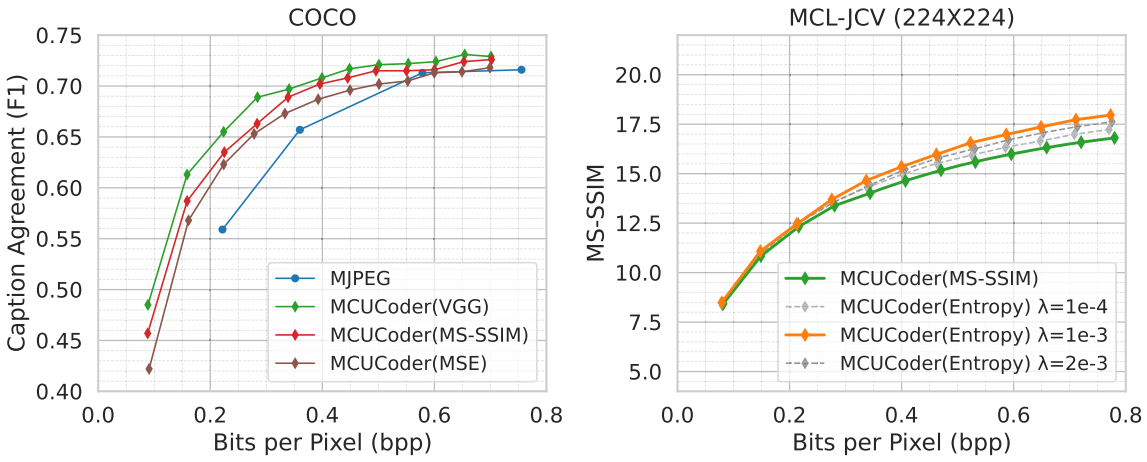


Fig. 9 **(a)** Top-1 classification accuracy of a DeiT-Base classifier pretrained on ImageNet-1K, evaluated on decoded images produced by M-JPEG and `MCUCoder` trained with different loss functions. `MCUCoder` consistently outperforms M-JPEG, and the model fine-tuned with the VGG loss achieves the best classification performance. **(b)** Object detection performance on the COCO dataset, measured in mAP@50. The results demonstrate the superior detection accuracy of `MCUCoder` compared to M-JPEG, highlighting its effectiveness for downstream AI tasks

tationally expensive operations such as motion estimation, motion compensation, transform coding, in-loop filtering, and entropy coding. These components typically require substantially higher processing capability, memory bandwidth, and hardware support than what is available on low-power battery-operated IoT devices.

We report the Bjøntegaard Delta (BD) rate (Bjøntegaard, 2001) for both datasets in Table 1. The results indicate that `MCUCoder` achieves a significantly higher MS-SSIM per bit compared to M-JPEG, highlighting its ability to deliver better video quality at lower bitrates. This is especially valuable for IoT applications, where achieving high compression rates with minimal computational overhead is crucial due to limited hardware resources. Additionally, `MCUCoder` has 12 "stacked" channels in its latent space, which provides 12 levels of quality that can be dynamically adjusted based on



(a) (b)

Fig. 10 **(a)**: Comparison of COCO (Chen et al., 2015) image captioning performance using BLIP-2 (Li et al., 2023) on decoded images produced by M-JPEG and `MCUCoder`. Performance is quantified using the token-level F1 agreement between captions generated from the original images and their decoded counterparts. `MCUCoder` consistently outperforms M-JPEG, demonstrating improved semantic preservation in the reconstructed images. **(b)**: Comparison of models trained with MS-SSIM alone and with MS-SSIM plus entropy regularization. The introduced entropy regularization term improves rate-distortion performance by encouraging more compressible latent representations. We also evaluate different values of the regularization weight λ and observe that $\lambda = 1e-3$ achieves the best overall performance

the available network bandwidth. In Fig. 14, we illustrate the bpp and MS-SSIM for each frame in a video from the UVG dataset for all 12 levels of quality. The results show that using more channels for decoding leads to a higher MS-SSIM, which verifies the effectiveness of the proposed stochastic dropout training. The PSNR results of `MCUCoder` (trained with MS-SSIM loss) are reported in Figure 13.

While MS-SSIM targets perceptual quality, some applications, such as traditional vision models, benefit from optimizing pixel-domain fidelity. We therefore also train `MCUCoder` with the MSE loss and evaluate it on the MCL-JCV dataset. As shown in Fig. 8a, MSE-based training achieves higher PSNR than MS-SSIM, confirming improved pixel-level reconstruction.

Image compression: To assess the image compression capabilities of `MCUCoder`, we conduct experiments on the CLIC (Workshop and Challenge on Learned Image Compression, 2020) and KODAK (Eastman Kodak, 1993) datasets, see Fig. 7. The results in Table 1 show that `MCUCoder` achieves an impressive average bitrate reduction of 55.75% on the KODAK dataset and 49.54% on the CLIC dataset, compared to JPEG. The PSNR results are reported in Figure 13.

4.2 Effect of entropy regularization.

To evaluate the impact of entropy-aware regularization, we compare the model trained solely with the MS-SSIM loss against a variant trained with the proposed entropy-regularized objective (MS-SSIM + entropy). As shown in Fig. 10b, incorporating entropy regularization improves performance at the higher bitrate ranges. This improvement indicates that encouraging lower-entropy latent representations leads to more efficient compression, allowing the model to achieve higher perceptual quality for a given bitrate. These results demonstrate that entropy regularization is an effective extension to the original training objective, improving rate-distortion performance without modifying the encoder architecture or increasing inference-time complexity. We also evaluate different values of the regularization weight λ and observe that $\lambda = 1e-3$ achieves the best overall performance.

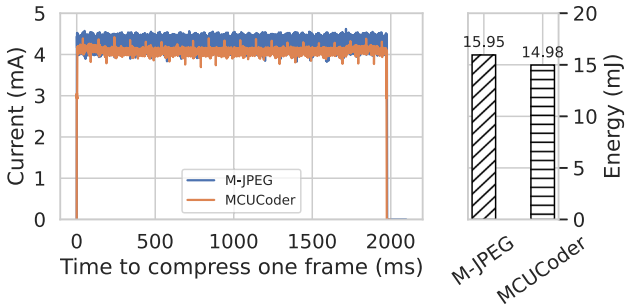


Fig. 11 Energy (Millijoule) and current (Milliampere) consumption of `MCUCoder` compared to M-JPEG for compressing one frame on the nRF5340. `MCUCoder` achieves comparable energy efficiency to M-JPEG while exceeding it in BD-rate

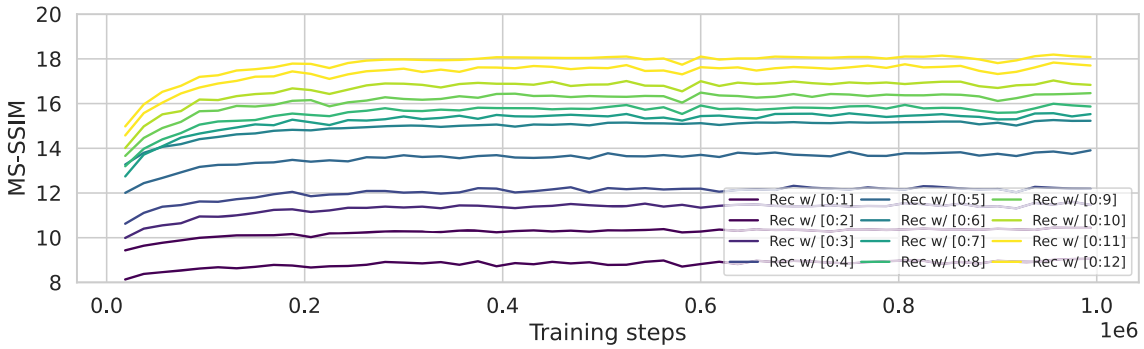


Fig. 12 MS-SSIM values on the KODAK dataset during training. The notation $[0 : k]$ represents the MS-SSIM of the reconstructed image using the first k latent channels out of a total of 12. As shown, with stochastic dropout training, all the sub-latents can be trained simultaneously without overfitting to any particular sub-latent

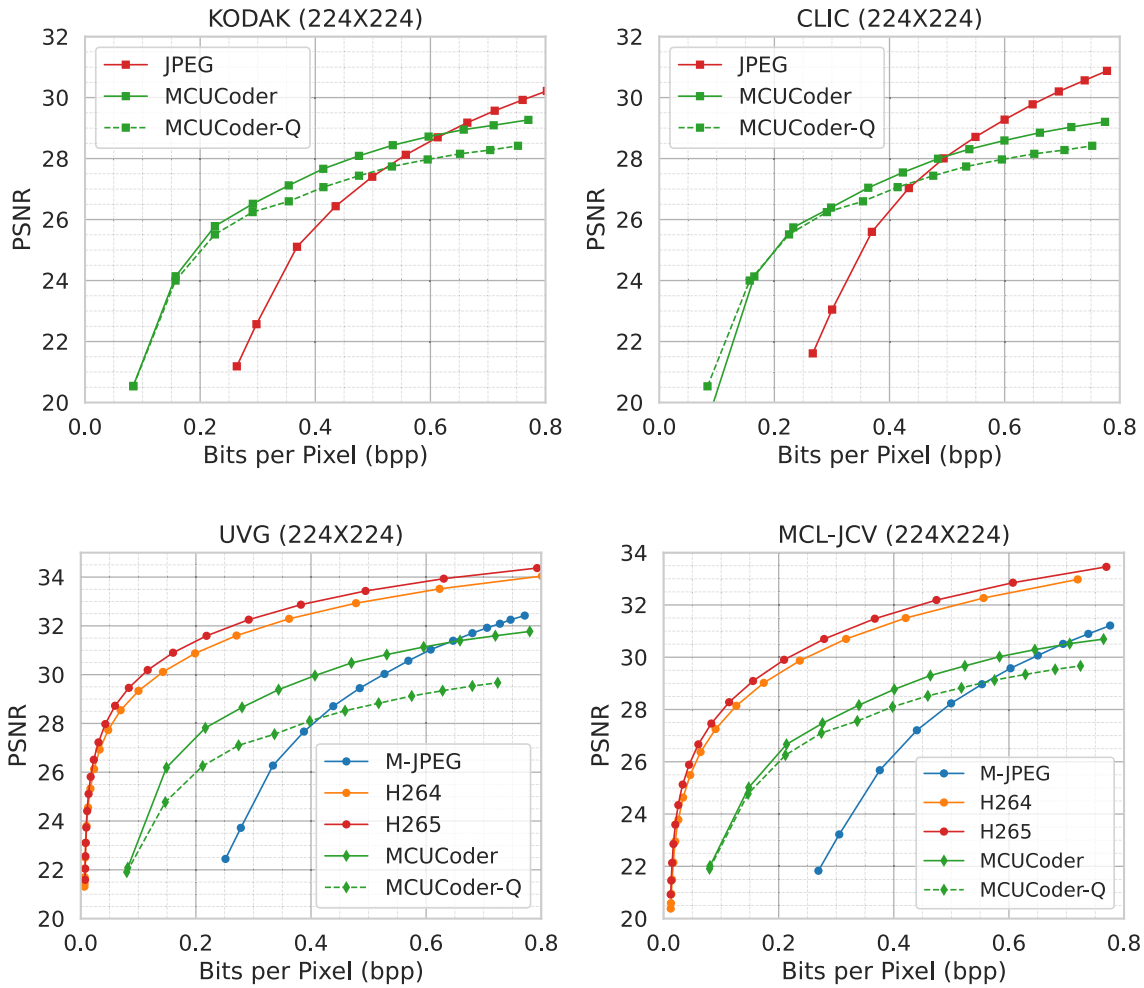


Fig. 13 PSNR comparison of `MCUCoder` and baselines across image (KODAK Eastman Kodak (1993), CLIC (Workshop and Challenge on Learned Image Compression, 2020)) and video (MCL-JCV (Wang et al., 2016), UVG (Mercat et al., 2020)) compression datasets

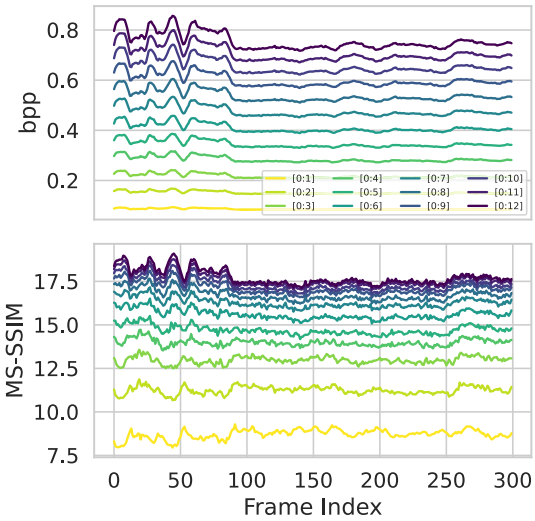


Fig. 14 MS-SSIM and bpp for the SunBath video from UVG (Mercat et al., 2020) dataset. $[0:k]$ shows the use of the first k channels (out of 12) for decoding

4.3 Comparison with high-complexity learned codecs

To further position `MCUCoder` relative to recent learned video compression methods, we compare it with DCVC-DC (Li et al., 2023) and DCVC-FM (Li et al., 2024) on the UVG dataset (Mercat et al., 2020). As shown in Fig. 15, these high-complexity learned codecs achieve stronger MS–SSIM rate–distortion performance than `MCUCoder`, which is expected given their substantially larger encoder architectures. In particular, DCVC-DC and DCVC-FM have approximately 8.0M and 5.9M encoder parameters, respectively, corresponding to about 758× and 560× more parameters than the 10.5K-parameter encoder of `MCUCoder`. This comparison highlights the intended trade-off of our design that `MCUCoder` does not target the unconstrained rate–distortion frontier, but instead prioritizes a minimal encoder that can be deployed on MCU-class devices with severe memory and compute constraints.

4.4 Latent ordering and DCT-JPEG alignment

Figure 5 illustrates the 12 latent channels derived from training with the stochastic dropout method. These channels display an intriguing hierarchical structure, where the early channels capture broad, low-frequency features, while the later channels progressively focus on finer, high-frequency details. This pattern closely resembles the Discrete Cosine Transform (DCT) basis matrix utilized in JPEG compression. In JPEG, the DCT plays a pivotal role in transforming image data into frequency components, allowing for efficient compression by prioritizing lower frequencies, which tend to carry more significant visual information. Similarly to `MCUCoder`, progressive JPEG leverages this frequency ordering, encoding data in a manner that allows the decoder to initially reconstruct the image using only low-frequency components, and as decoding progresses, higher-frequency details are incrementally added, resulting in a progressively refined image reconstruction.

4.5 Compression for AI

Image classification: We evaluate the impact of `MCUCoder` compression on image classification using a pretrained DeiT-Base (Touvron et al., 2021) model on the ImageNet-1K (Deng et al., 2009) dataset. Classification performance is assessed by measuring the top-1 accuracy of the classification model. As shown in Fig. 9a, `MCUCoder` consistently outperforms M-JPEG, demonstrating improved preservation of discriminative visual features. Furthermore, the model trained with the VGG perceptual loss achieves the best classification performance compared to those trained with MS-SSIM and MSE, indicating that VGG-based optimization is more effective for AI-oriented classification tasks.

Object detection: We evaluate the impact of `MCUCoder` compression on object detection using the COCO (Chen et al., 2015) validation set and a pretrained Faster R-CNN with a ResNet-50-FPN backbone (Ren et al., 2015). Detection performance is measured using AP@50 on reconstructed images. As shown in Fig. 9b, `MCUCoder` consistently outperforms M-JPEG, indicating improved preservation of object-level semantics under compression.

Vision–Language Model (VLM): We study the impact of `MCUCoder` compression on VLMs using the COCO Caption validation set (Lin et al., 2014) and BLIP-2 (FLAN-T5-XL) (Li et al., 2023), a pretrained vision-language model that generates image captions by conditioning a Large Language Model (LLM) on visual features extracted from the input image. Specifically, BLIP-2 bridges a frozen vision encoder and a frozen LLM through a lightweight query-based transformer, enabling semantic reasoning over visual content while remaining robust to low-level pixel variations. Captioning performance is evaluated using the token-level F1 agreement between captions generated from the original images and those generated from their reconstructed versions. This setup helps isolate the impact of compression on the VLM output, independent of the absolute captioning quality of the VLM itself. We therefore do not present this metric as a reference-caption quality score, but as a measure of whether compression changes the semantic output produced by the same model. As shown in Fig. 10a, `MCUCoder` outperforms M-JPEG, indicating improved preservation of semantic content. Moreover, the model trained with the VGG perceptual loss achieves the highest captioning performance

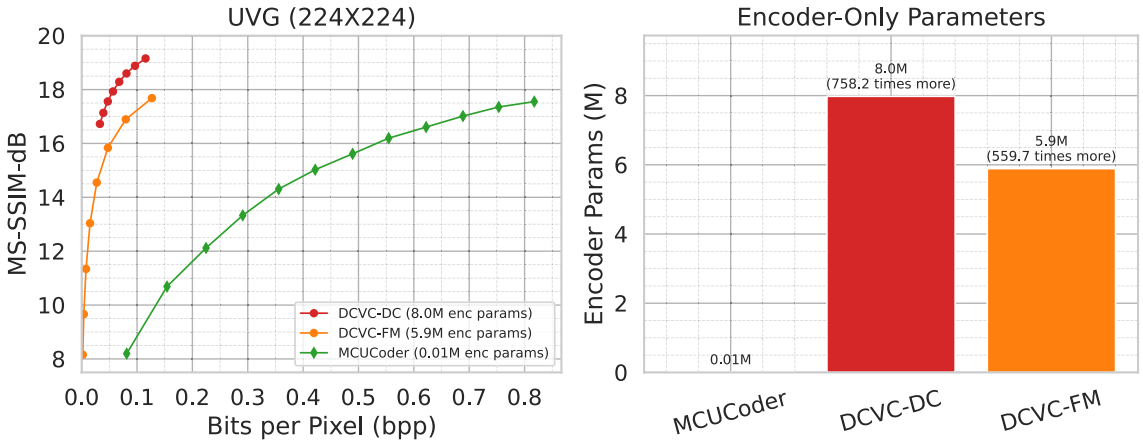


Fig. 15 Comparison of `MCUCoder` with recent high-complexity, GPU-oriented learned video compression baselines on UVG (Mercat et al., 2020). The left plot shows that DCVC-DC (Li et al., 2023) and DCVC-FM (Li et al., 2024) achieve stronger MS–SSIM rate–distortion performance, which is expected given their substantially larger and more complex encoders. The right plot shows that their encoders have approximately 758× and 560× more parameters than `MCUCoder`, respectively. This complexity gap is central to the comparison as these models target much more capable hardware, whereas `MCUCoder` is designed to remain deployable on MCU-class devices with severe memory and compute constraints

compared to models trained with MS-SSIM and MSE, highlighting the effectiveness of VGG-based optimization for AI-oriented tasks.

4.6 Model complexity

Table 3 summarizes the computational complexity of `MCUCoder` in terms of parameters and GFLOPs. The encoder is intentionally lightweight, with only 10.5K parameters and 0.72 GFLOPs, making it feasible for execution on resource-constrained MCUs. In contrast, the decoder is substantially larger, with 3M parameters and higher computational cost, and is designed to run on more capable cloud or edge servers. This asymmetric design shifts computational complexity away from the MCU, enabling efficient on-device encoding while maintaining high reconstruction quality at the decoder.

4.7 Performance on MCUs

We implement `MCUCoder` on two widely-used MCU platforms, the STM32F7 and nRF5340 MCUs, using TFLite-Micro and Zephyr RTOS. The STM32F7 features 2 MB of Flash memory, 1 MB of RAM, and a Cortex-M7 processor, while the nRF5340 is equipped with 1 MB of Flash, 512 KB of RAM, and a Cortex-M33 processor. Both MCUs support DSP and CMSIS-NN acceleration, making them well-suited for running lightweight deep learning models. As detailed in Table 2, `MCUCoder` demonstrates a low memory footprint, consuming 360 KB of RAM on the STM32F7 and 344 KB on the nRF5340, which is efficient for such constrained devices. This compact memory usage highlights the suitability of `MCUCoder` for low-power, resource-constrained IoT applications. To assess the energy efficiency of `MCUCoder`, we conducted a comparative analysis against M-JPEG. Specifically, we measured the energy consumption of `MCUCoder` and an optimized JPEG encoder for the Cortex-M series[1] on the nRF5340 platform; see Fig. 11. The results indicate that `MCUCoder` achieves comparable energy consumption to JPEG, while providing superior performance in terms of BD-rate, as shown in Table 1. However, the nRF5340 exhibits noticeably slower processing performance compared to the STM32F7 for both `MCUCoder` and M-JPEG. This discrepancy suggests that while the nRF5340 is energy-efficient, its lower computational capability makes it more suitable for event-driven or low-duty-cycle sensing applications rather than real-time, high-frame-rate video streaming. The STM32F7, on the other hand, provides higher throughput and is therefore better suited for scenarios that require more frequent frame transmission under low-bitrate constraints. However, given the measured latency, neither platform is suitable for high-frame-rate real-time video streaming, which is expected for such battery-powered devices.

4.8 Stochastic dropout training analysis

One potential challenge with stochastic dropout training is the risk of overfitting to specific loss functions when optimizing multiple losses concurrently. To evaluate this, we track the MS-SSIM of `MCUCoder` on the KODAK (Eastman Kodak, 1993) dataset across varying numbers of active

[1] https://github.com/noritsuna/JPEGEncoder4Cortex-M

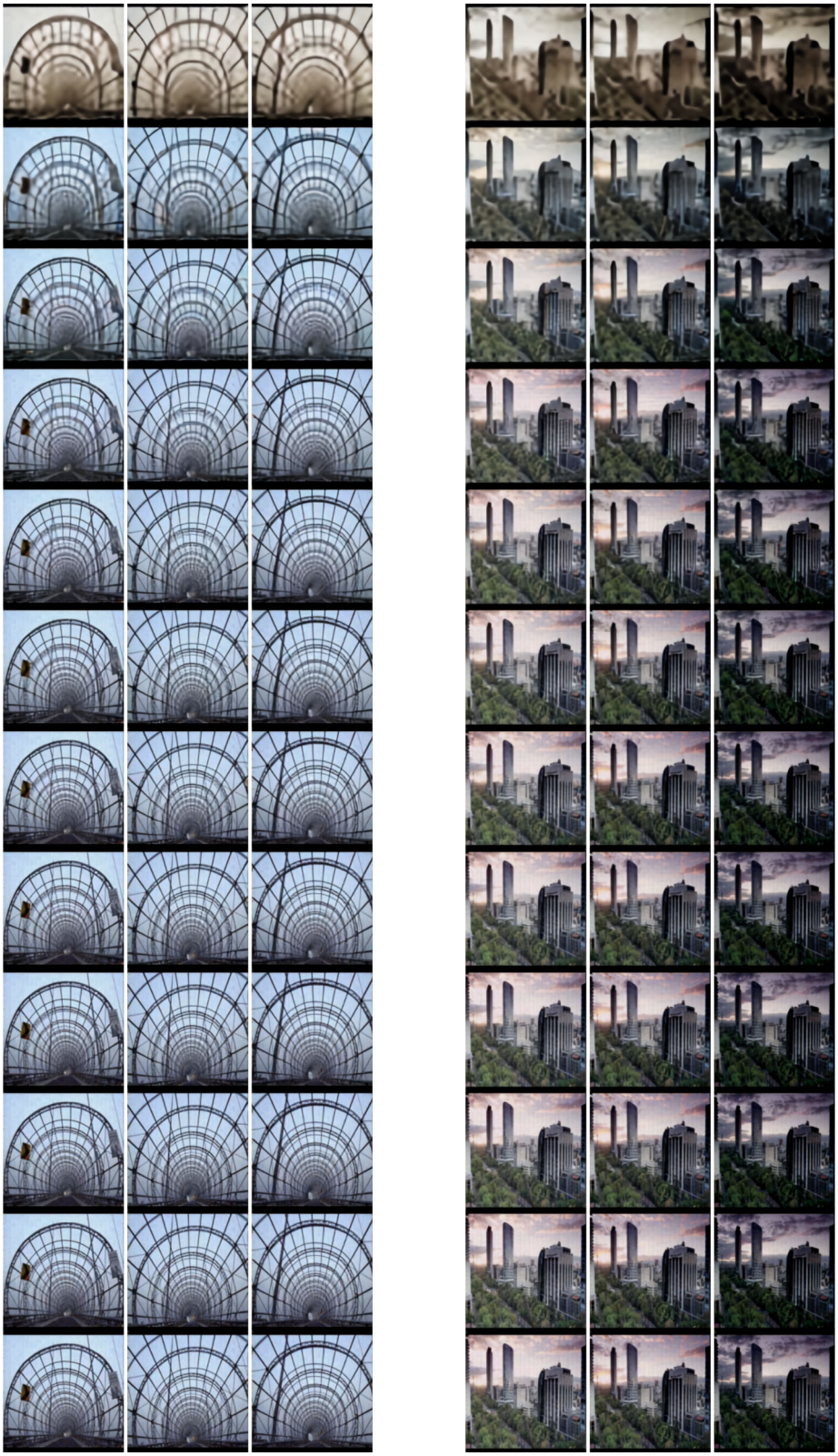


Fig. 16 Some samples from the MCL-JCV (Wang et al., 2016) dataset. The columns represent different frames, while the rows display progressively improving levels of quality from top to bottom, produced by `MCUCoder`

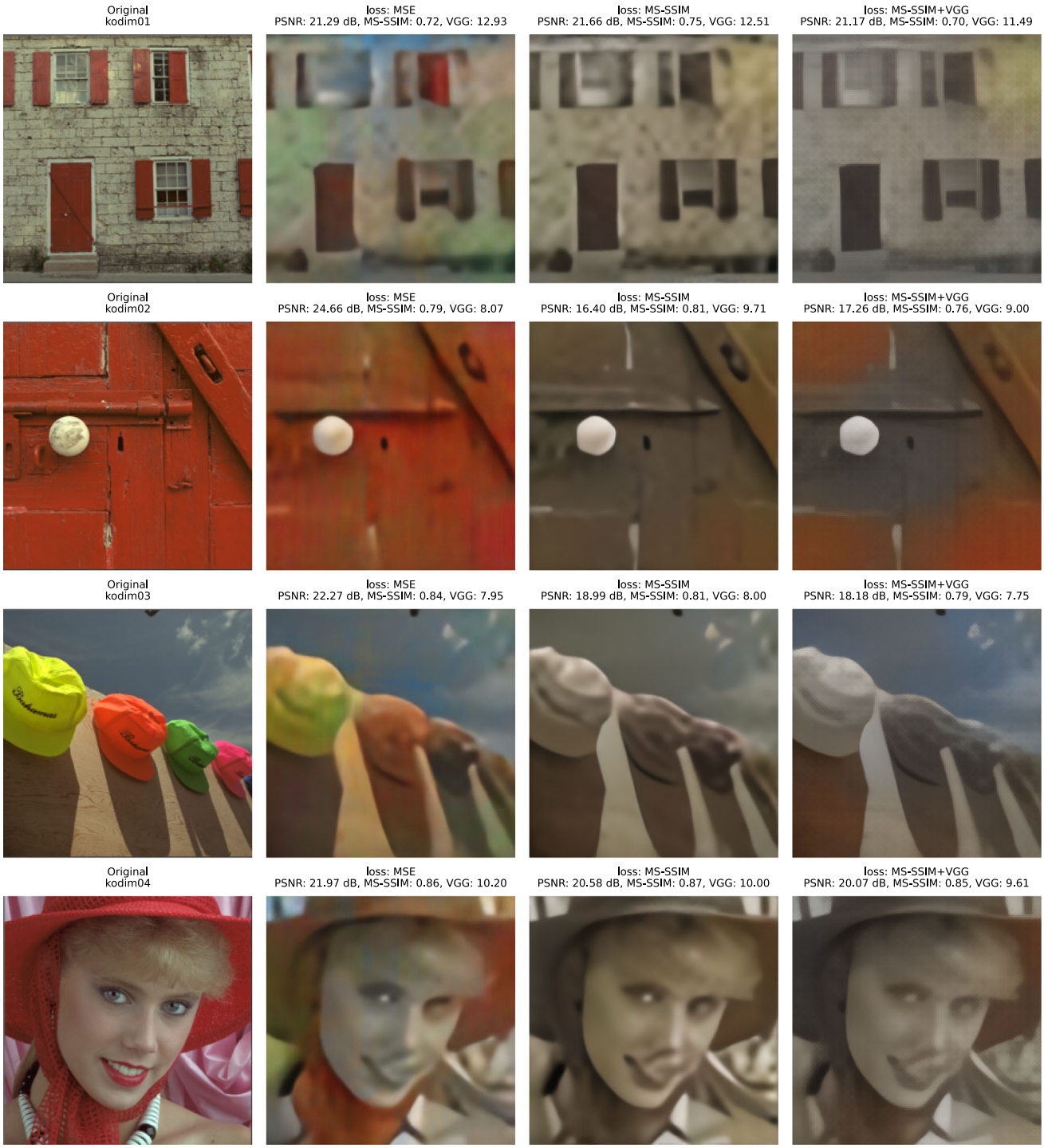


Fig. 17 Qualitative comparison for 1 transmitted latent channel out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

latent channels during training. The training logs, shown in Fig. 12, demonstrate that all the sub-latents are trained in parallel without overfitting to any specific sub-latent, which verifies the effectiveness of the uniform latent sampling strategy employed in the training.

4.9 Qualitative comparison of training losses

Figures 17–28 compare the qualitative reconstruction results of `MCUCoder` models trained with different loss functions, namely MSE, MS-SSIM, and perceptual loss. The results show that the choice of training objective significantly affects

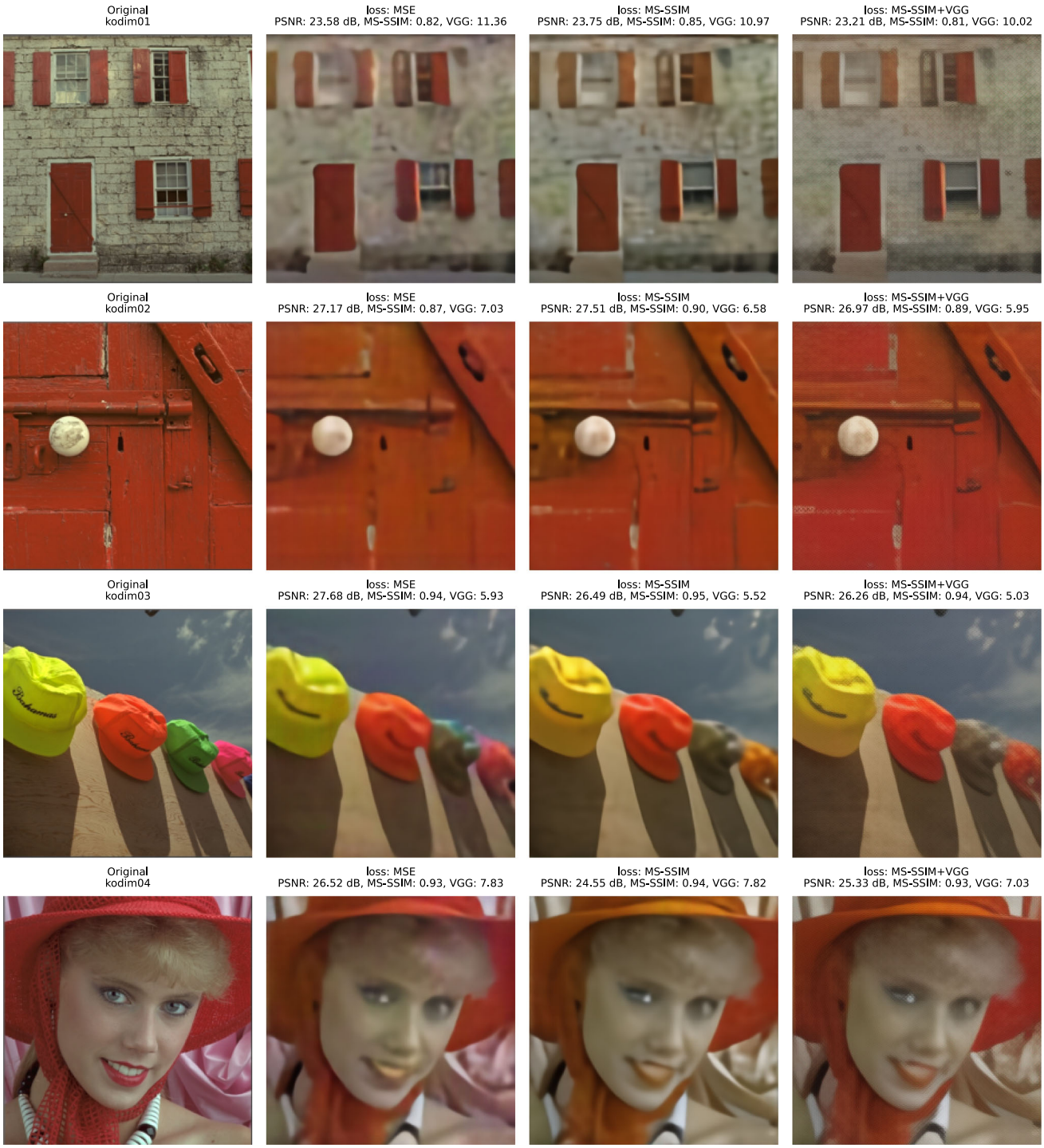


Fig. 18 Qualitative comparison for 2 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

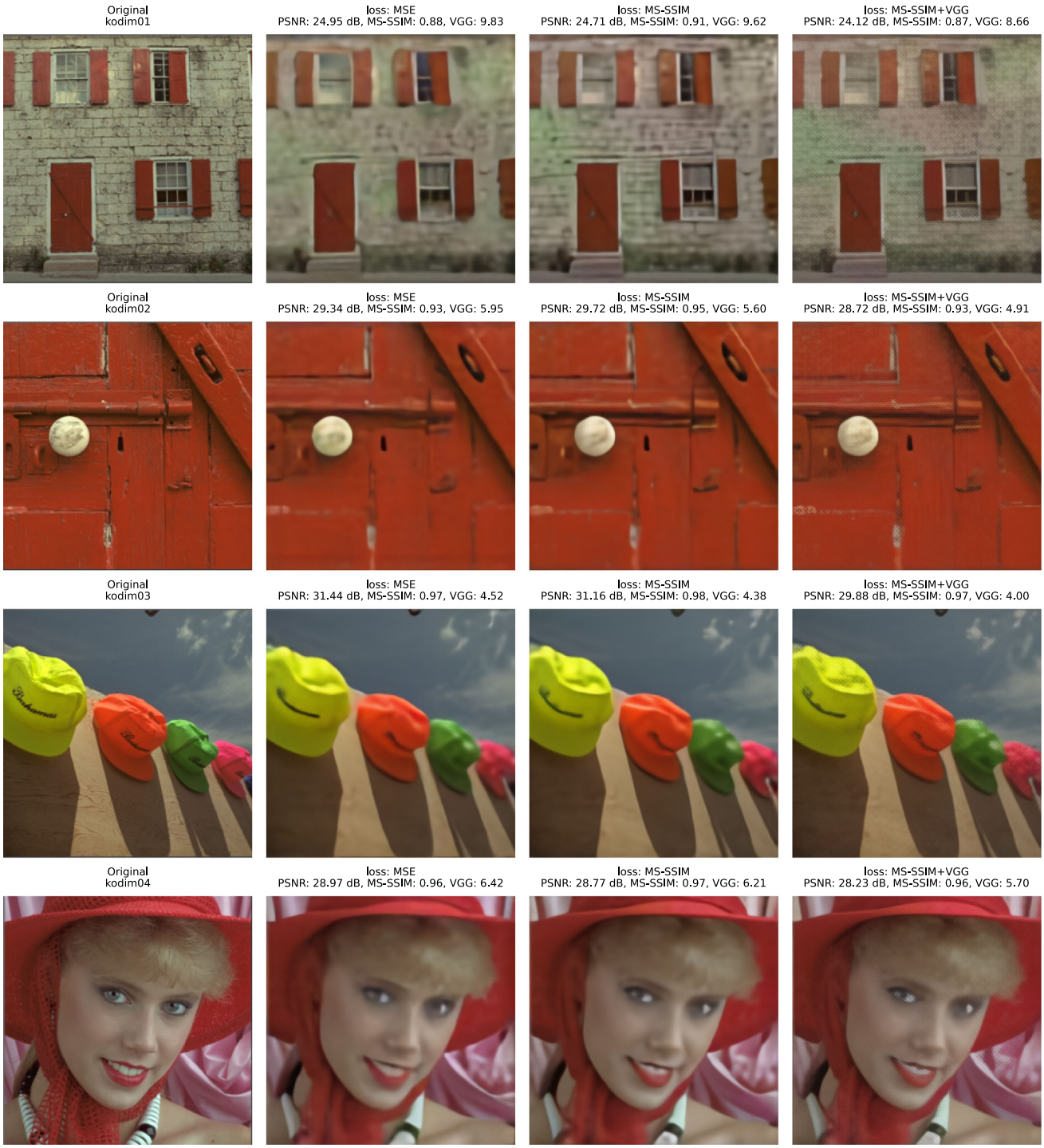


Fig. 19 Qualitative comparison for 3 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

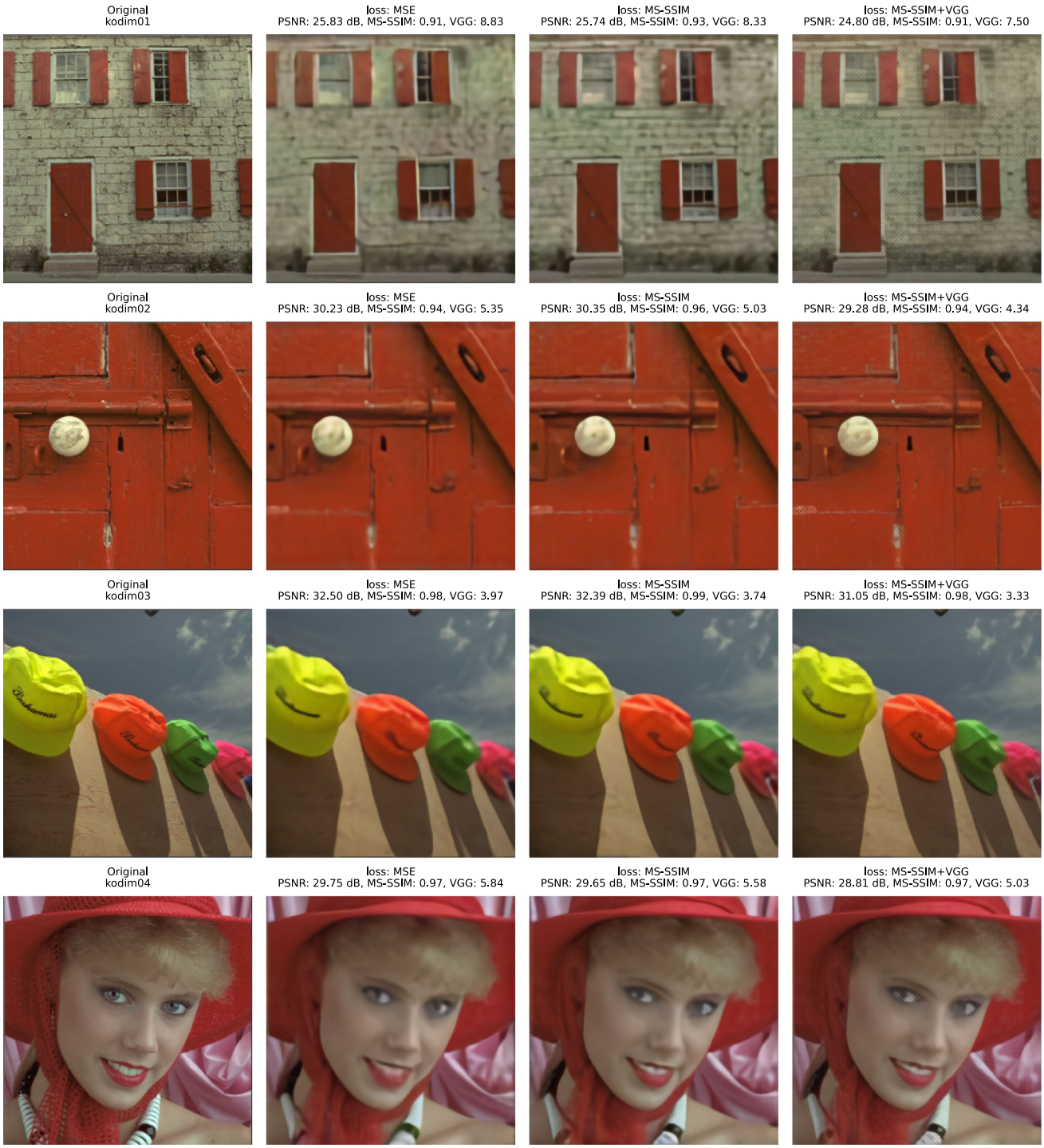


Fig. 20 Qualitative comparison for 4 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

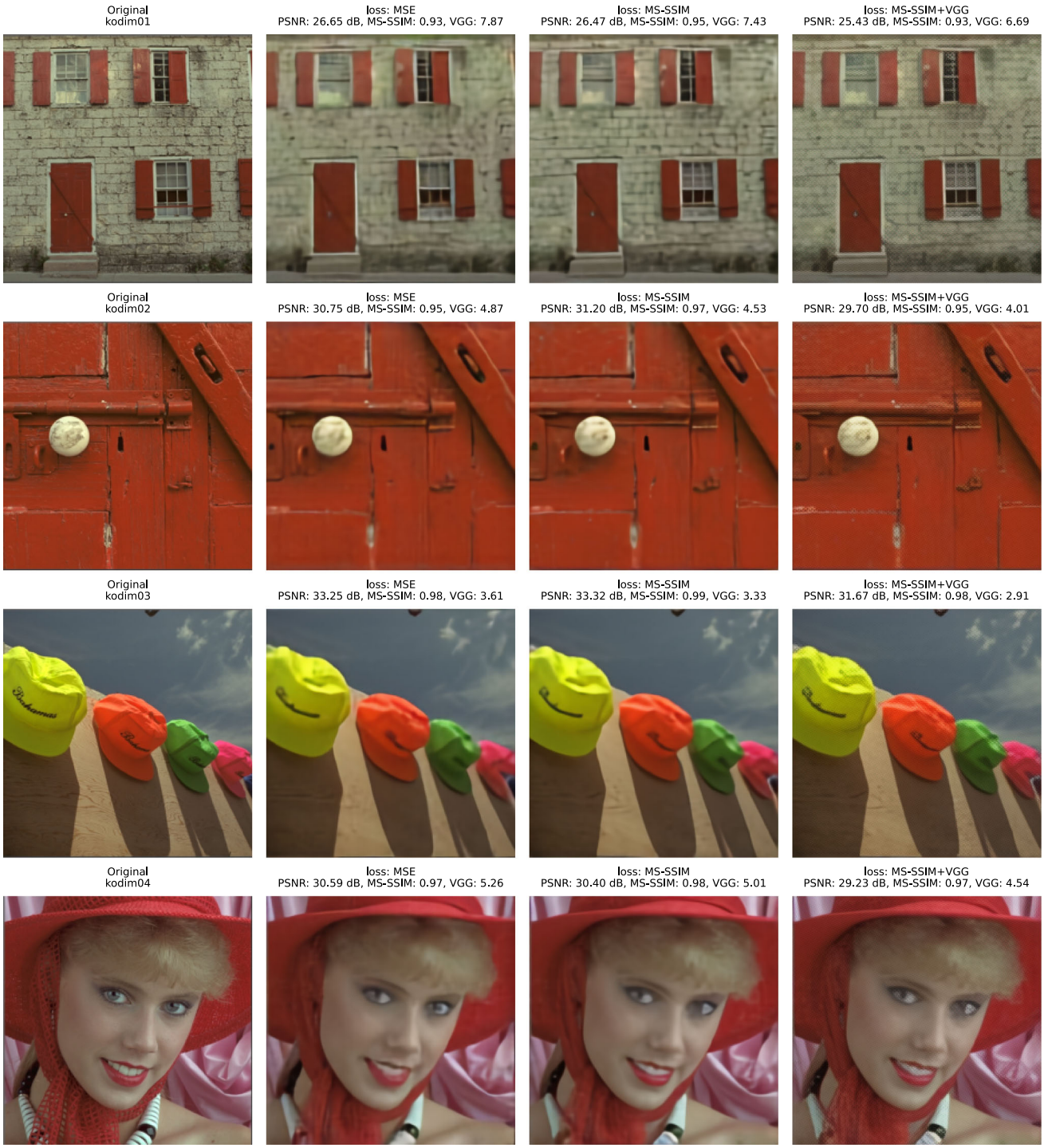


Fig. 21 Qualitative comparison for 5 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

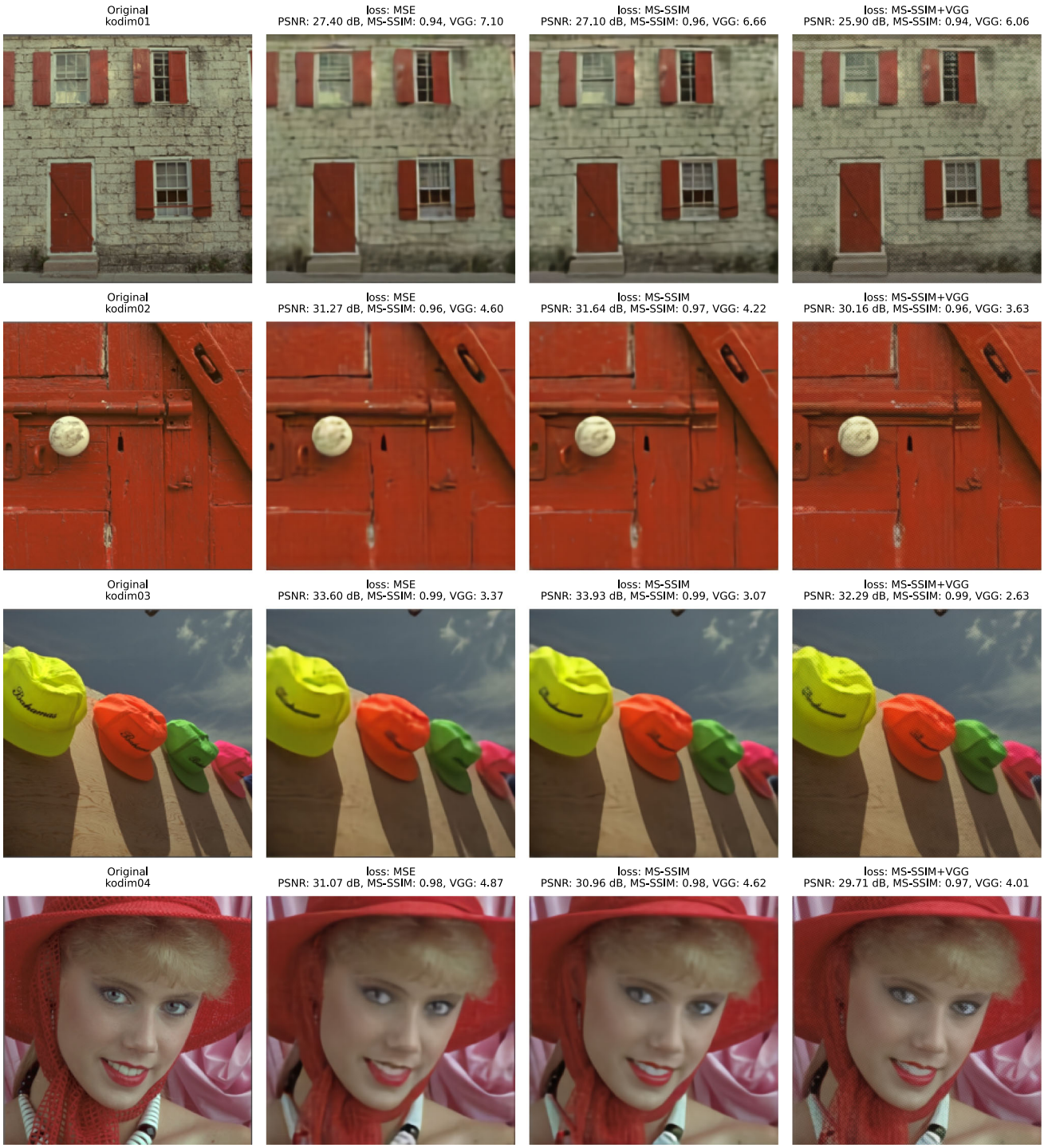


Fig. 22 Qualitative comparison for 6 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

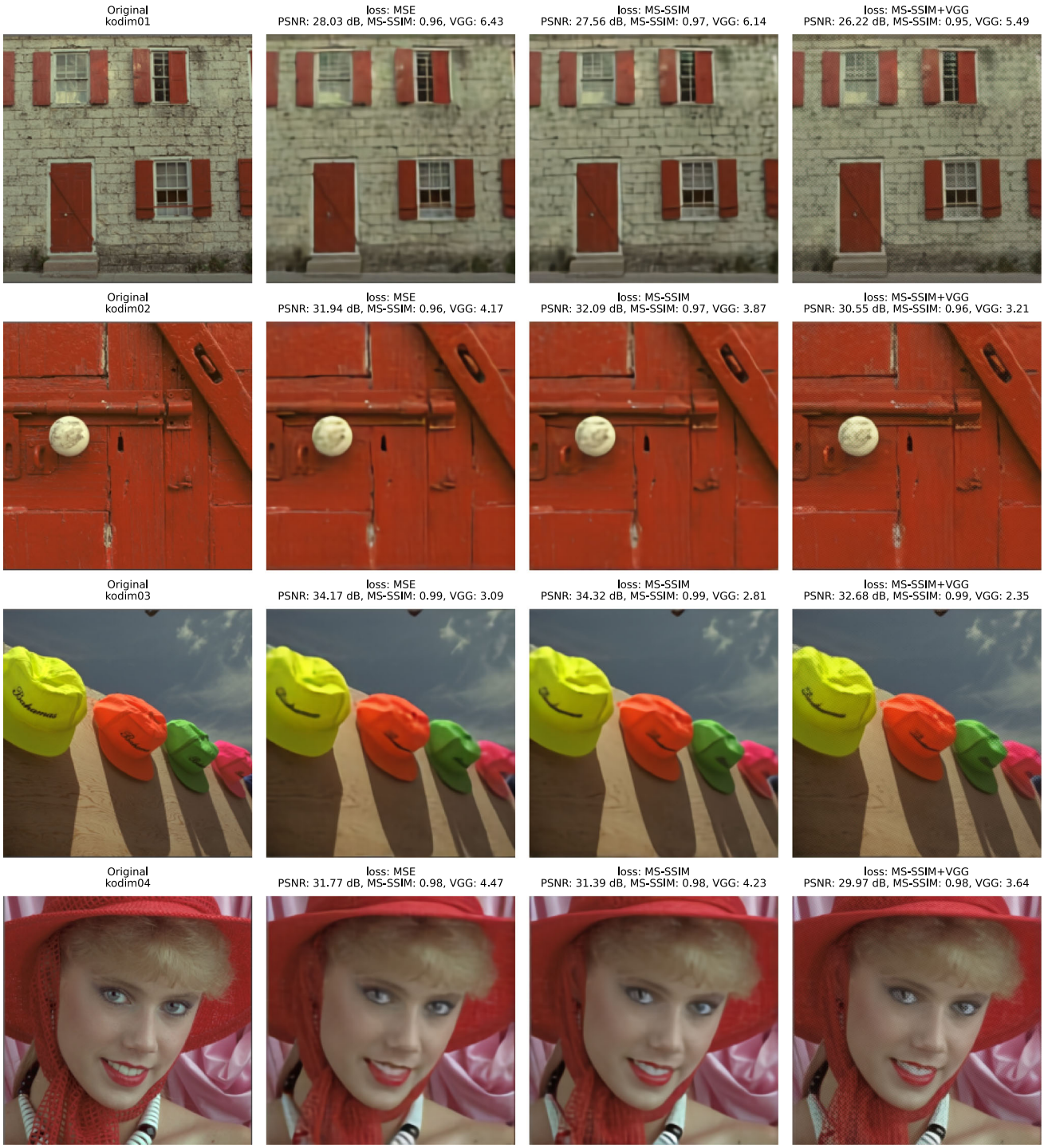


Fig. 23 Qualitative comparison for 7 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

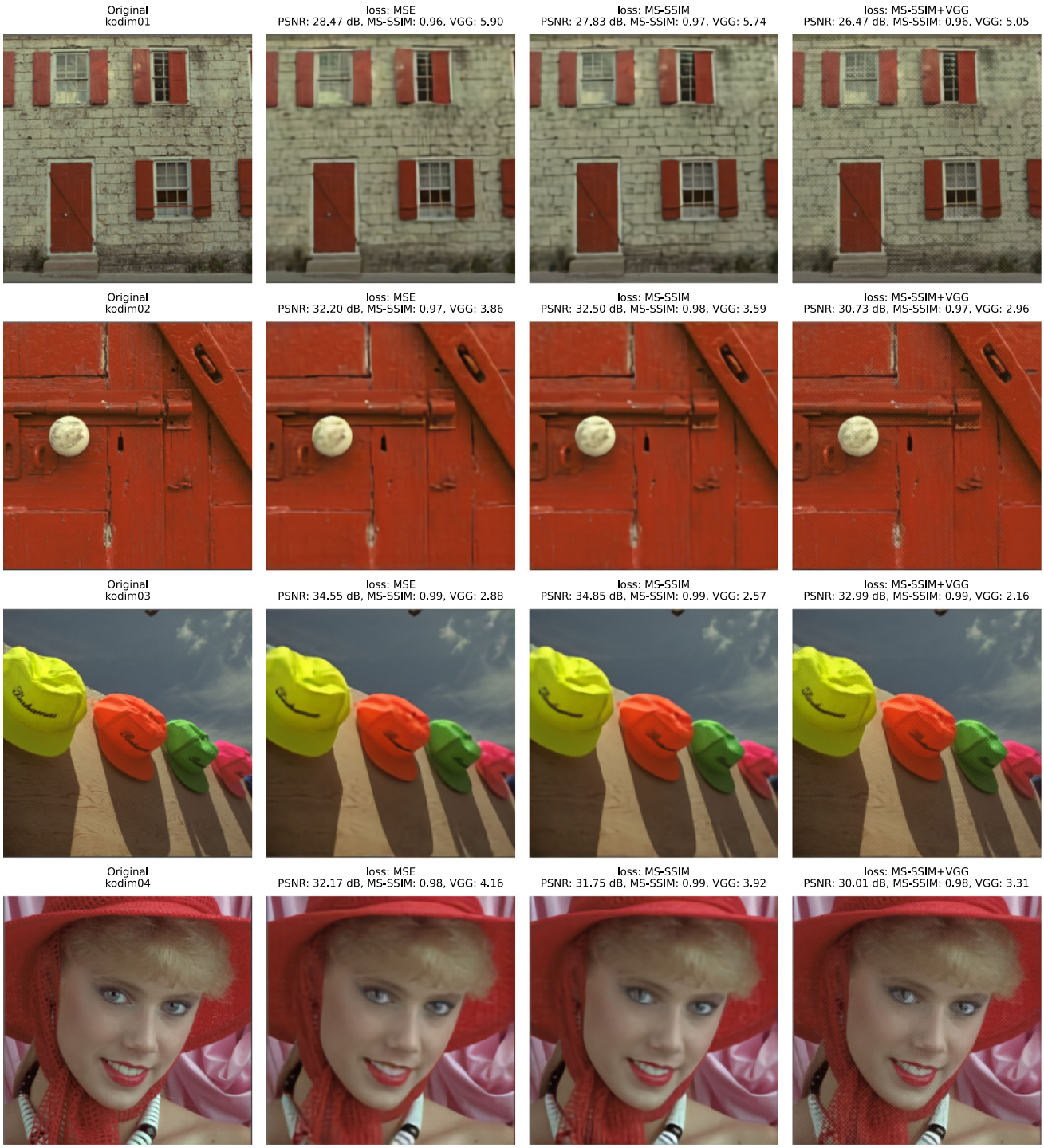


Fig. 24 Qualitative comparison for 8 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

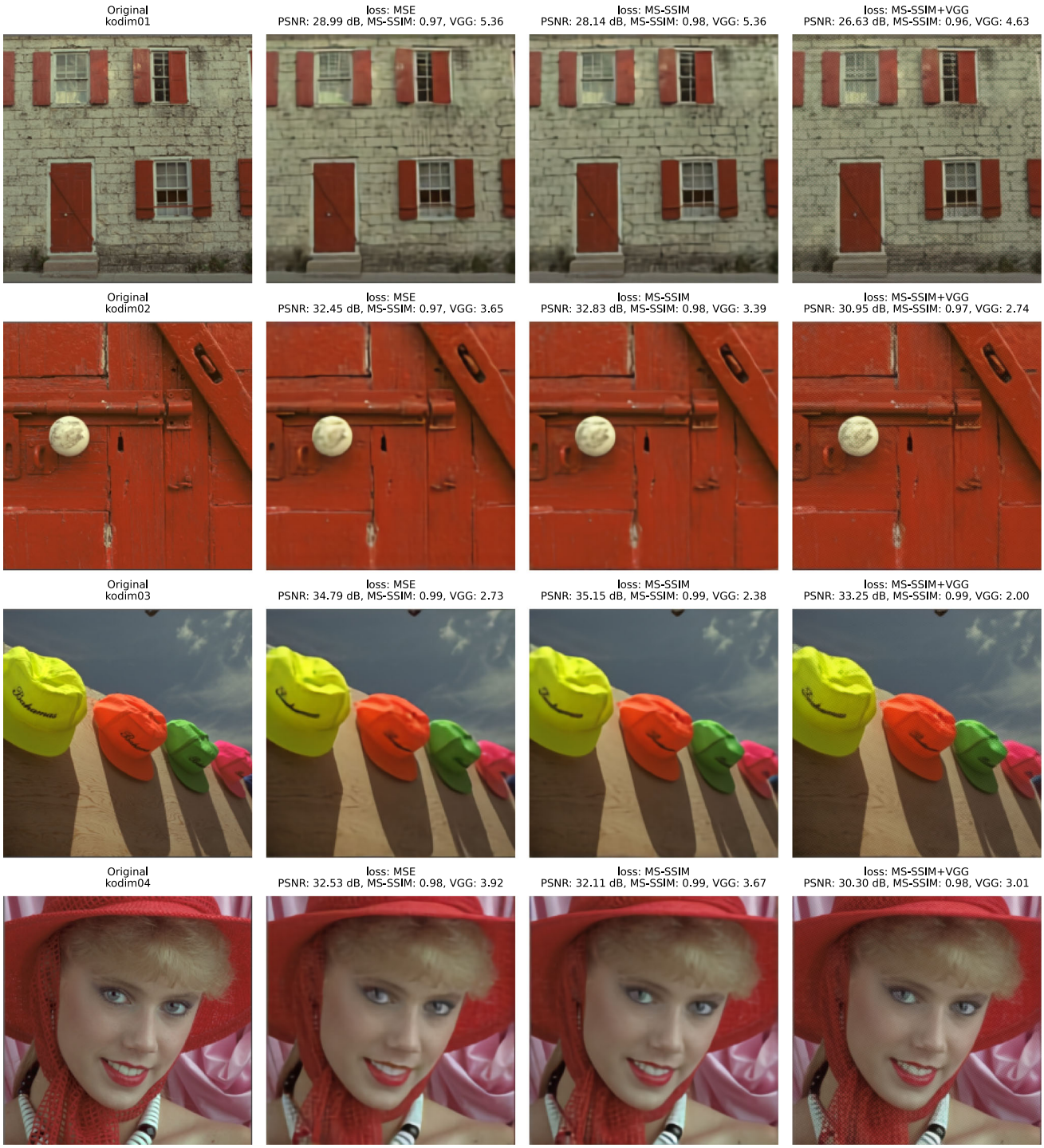


Fig. 25 Qualitative comparison for 9 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

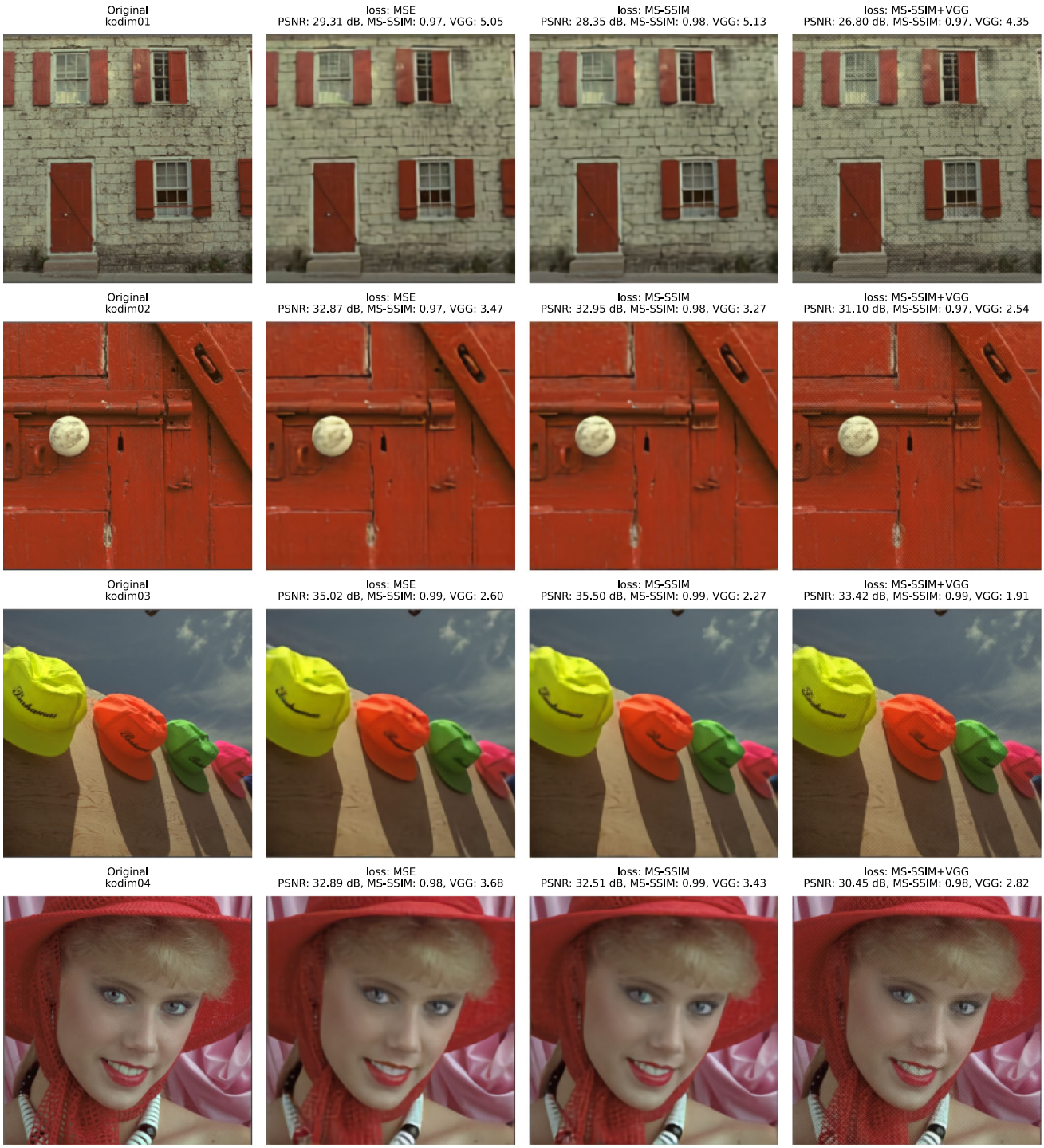


Fig. 26 Qualitative comparison for 10 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

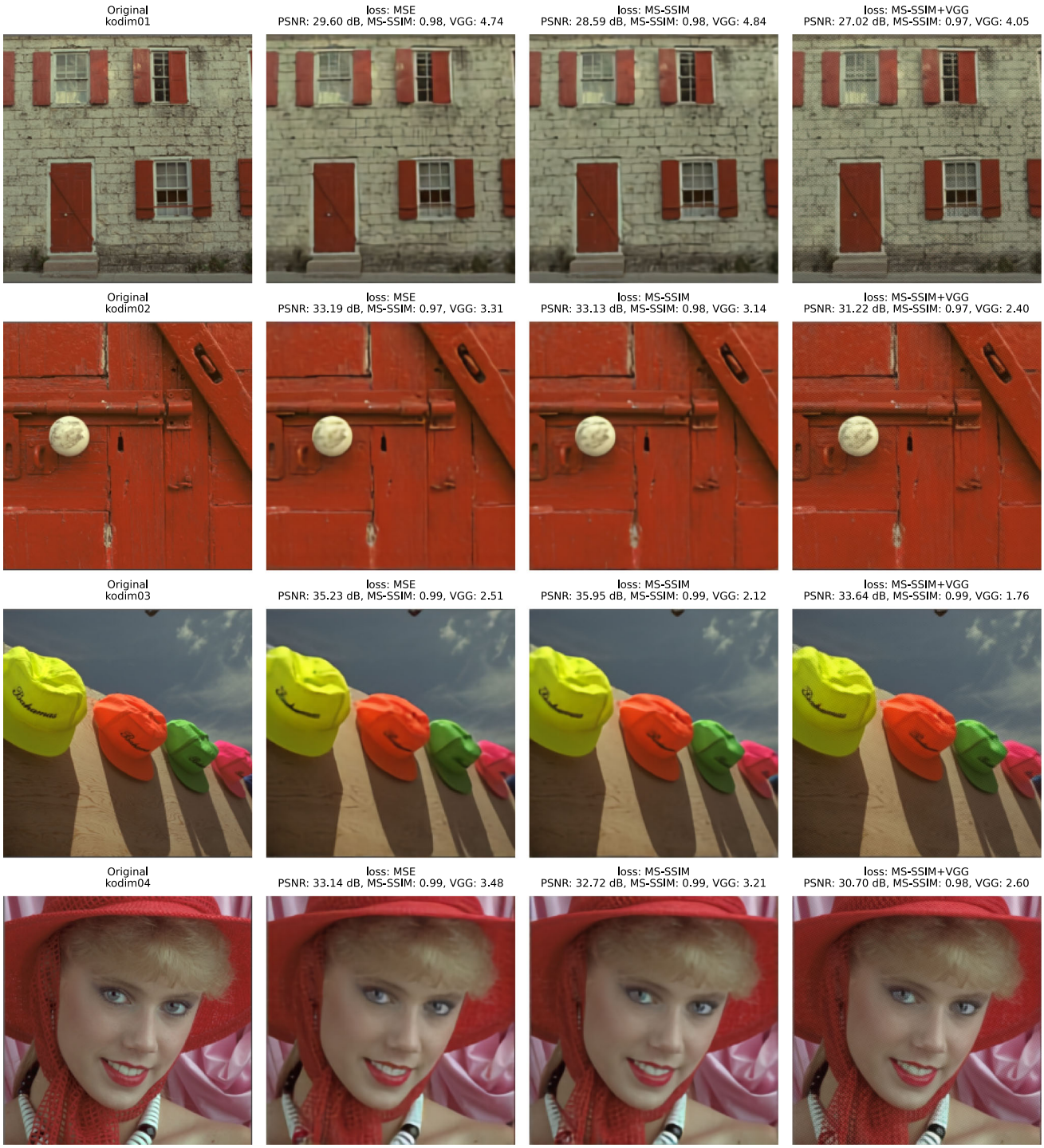


Fig. 27 Qualitative comparison for 11 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

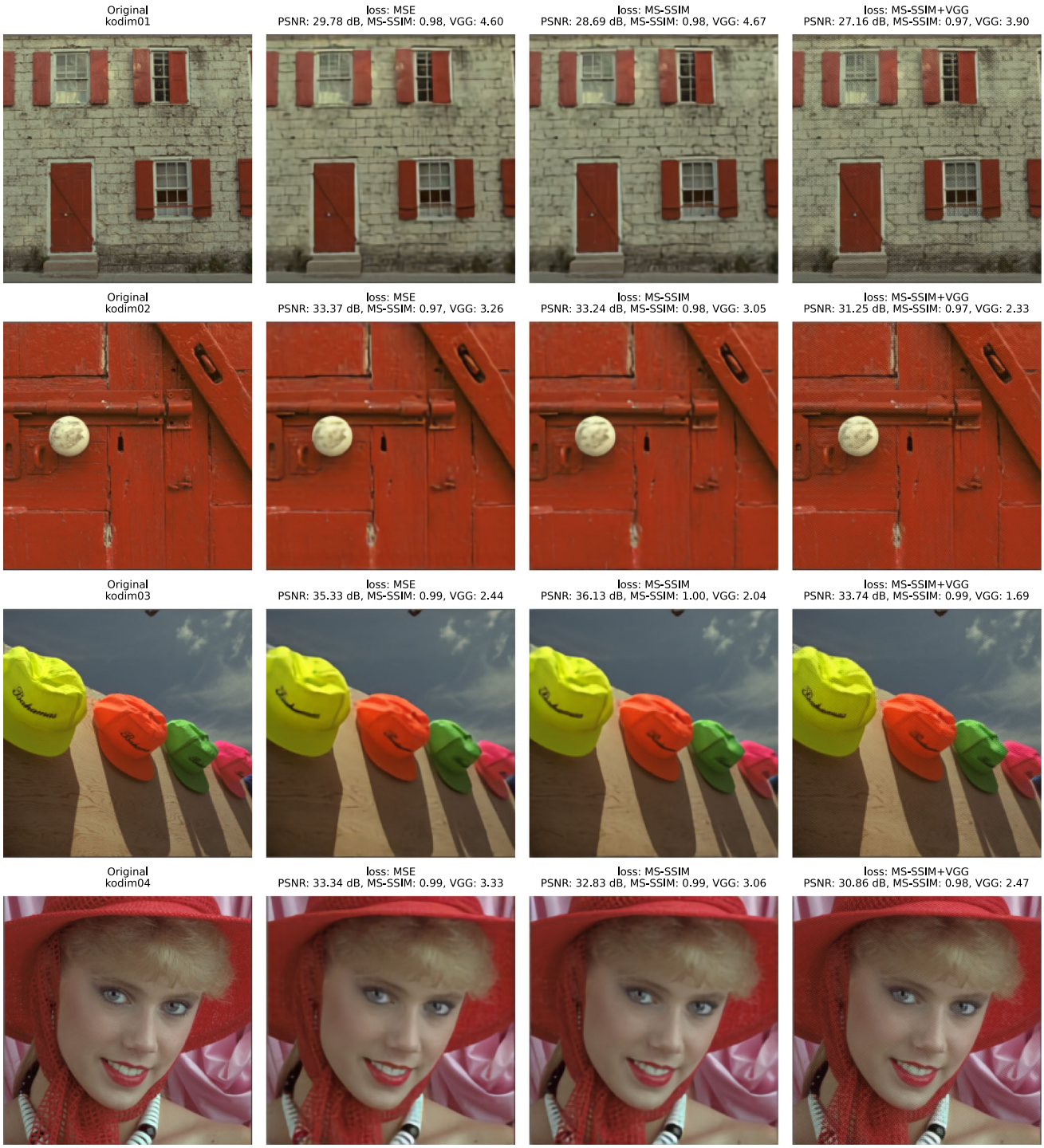


Fig. 28 Qualitative comparison for 12 transmitted latent channels out of 12. The images are from the KODAK (Eastman Kodak, 1993) dataset

the visual appearance of the reconstructed frames. The model trained with MSE tends to produce smoother reconstructions with more noticeable color distortions, especially when the bitrate is very low. This behavior is expected because MSE minimizes the pixel-wise difference between the reconstructed and original frames. Under a highly constrained latent representation, the model therefore favors averaged pixel values that reduce the overall numerical error, which can lead to blurred details and color shifts.27

The model trained with MS-SSIM produces reconstructions that are less colorful and closer to grayscale, while better preserving the global image structure. Since MS-SSIM measures structural similarity across multiple scales, it encourages the model to preserve luminance, contrast, and spatial structure rather than exact pixel-level values. As a result, the reconstructed frames better maintain the layout and edges of the scene, although some regions may appear coarser or less detailed.

Since the model trained with perceptual loss is first initialized using the MS-SSIM objective and then fine-tuned with the VGG-based perceptual loss, it tends to preserve the structure of objects while also introducing more color information in the reconstructed frames at low bitrates. This is beneficial for AI-based perception tasks, because many vision models depend on object boundaries, shapes, textures, and semantic features rather than exact pixel-wise reconstruction. Therefore, preserving the structural content while recovering visually meaningful color and texture cues can make the reconstructed frames more informative for downstream machine perception.

5 Conclusion

We introduced `MCUCoder`, an ultra-lightweight asymmetric video compression model for resource-constrained IoT devices. With just 10.5K parameters and a 350KB memory footprint, compared to M-JPEG, `MCUCoder` reduces bitrate by over 55% on both the MCL-JCV and UVG datasets while matching the efficiency of M-JPEG. We further show that `MCUCoder` outperforms M-JPEG on downstream AI tasks, including image classification, object detection, and image captioning. Its adaptive-bitrate design allows the transmitted representation to be adjusted under fluctuating network conditions, making it suitable for low-power edge applications with constrained bandwidth.

Author Contributions Ali Hojjat was responsible for the main conceptualization, methodology, implementation, experimental work, evaluation, and drafting of the manuscript. Janek Haberer contributed to the development of the research idea, study design, evaluation, and manuscript writing and revision. Olaf Landsiedel supervised the project and contributed to reviewing and revising the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This research received funding from the Federal Ministry for Digital and Transport under the CAPTN-Förde 5G project (grant no. 45FGU139H), the German Ministry of Transport and Digital Infrastructure through the CAPTN Förde Areal II project (grant no. 45DTWV08D), the Federal Ministry for Economic Affairs and Energy under the CAPTN X-FERRY project (grant no. 03SX612A), and the Federal Ministry for Economic Affairs and Climate Action under the Marispace-X project (grant no. 68GX21002E). The work was supported in part by high-performance computing resources provided by the Kiel University Computing Centre and the Hydra computing cluster, funded by the German Research Foundation (grant no. 442268015) and the Petersen Foundation (grant no. 602157).

Data Availability The code, training data links, and pretrained model weights used in this study are publicly available at: https://github.com/ds-kiel/MCUCoder

Declarations

Competing Interests The authors declare that they have no competing interests.

Ethics Approval This research does not involve human participants or animals.

Consent to Participate Not applicable, as this research does not involve human participants.

Consent for Publication Not applicable, as this study does not contain any identifiable individual data.

References

Agustsson, E., Minnen, D., Johnston, N., Balle, J., Hwang, S. J., & Toderici, G. (2020). Scale-space flow for end-to-end optimized video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8503–8512.

ARM-software: CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs. Accessed: 2024-09-22 (2024). https://github.com/ARM-software/CMSIS-NN

Ballé, J., Laparra, V., & Simoncelli, E. P. (2015). Density modeling of images using a generalized normalization transformation. Preprint arXiv:1511.06281.

Ballé, J., Laparra, V., & Simoncelli, E. P. (2016). End-to-end optimized image compression. Preprint arXiv:1611.01704.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., & Johnston, N. (2018). Variational image compression with a scale hyperprior. Preprint arXiv:1802.01436.

Bejarano-Carbo, A., An, H., Choo, K., Liu, S., Zhang, Q., Sylvester, D., Blaauw, D., & Kim, H.-S. (2022). Millimeter-scale ultra-low-power imaging system for intelligent edge monitoring. Preprint arXiv:2203.04496.

Bjøntegaard, G. (2001). Calculation of average psnr differences between rd-curves. https://api.semanticscholar.org/CorpusID:61598325

Bross, B., Wang, Y.-K., Ye, Y., Liu, S., Chen, J., Sullivan, G. J., & Ohm, J.-R. (2021). Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology, 31*(10), 3736–3764.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., & Zitnick, C. L. (2015). Microsoft coco captions: Data collection and evaluation server. Preprint arXiv:1504.00325.

Chen, Y., He, B., Lyu, Z., Hu, H., Gu, Q., Tian, Y., & Lu, G. (2026). Adaptive learned image compression with graph neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12150–12161.

Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2020). Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7939–7948.

Chen, H., He, B., Wang, H., Ren, Y., Lim, S. N., & Shrivastava, A. (2021). Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems, 34*, 21557–21568.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE.

Eastman Kodak: Kodak Lossless True Color Image Suite (PhotoCD PCD0992). http://r0k.us/graphics/kodak. 6 (1993)

Feng, D., Cheng, Z., Wang, S., Wu, R., Hu, H., Lu, G., & Song, L. (2025). Linear attention modeling for learned image compression. In: Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 7623–7632.

Fu, H., Liang, J., Fang, Z., Han, J., Liang, F., & Zhang, G. Weconvene: Learned image compression with wavelet-domain convolution and entropy model. In: European Conference on Computer Vision, pp. 37–53 (2024). Springer

Gao, G., You, P., Pan, R., Han, S., Zhang, Y., Dai, Y., & Lee, H. (2021). Neural image compression via attentional multi-scale back projection and frequency decomposition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14677–14686.

Haberer, J., Hojjat, A., & Landsiedel, O. (2024). Hydravit: Stacking heads for a scalable vit. Preprint arXiv:2409.17978.

Han, M., Jiang, S., Li, S., Deng, X., Xu, M., Zhu, C., & Gu, S. (2024). Causal Context Adjustment Loss for Learned Image Compression. In: Advances in Neural Information Processing Systems, vol. 37. https://doi.org/10.52202/079017-4234

He, G., Wu, C., Li, L., Zhou, J., Wang, X., Zheng, Y., Yu, B., & Xie, W. (2020). A video compression framework using an overfitted restoration neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 148–149.

He, D., Yang, Z., Peng, W., Ma, R., Qin, H., & Wang, Y. (2022). Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5718–5727.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

He, D., Zheng, Y., Sun, B., Wang, Y., & Qin, H. (2021). Checkerboard context model for efficient learned image compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14771–14780.

Hojjat, A., Haberer, J., & Landsiedel, O. (2023). Progdtd: Progressive learned image compression with double-tail-drop training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 1130–1139.

Hojjat, A., Haberer, J., Zainab, T., & Landsiedel, O. (2024). Limitnet: Progressive, content-aware image offloading for extremely weak devices & networks. In: Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services, pp. 519–533.

Hu, P., Im, J., Asgar, Z., & Katti, S. (2020). Starfish: Resilient image compression for aiot cameras. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems, pp. 395–408.

Hu, Z., Lu, G., & Xu, D. (2021). Fvc: A new framework towards deep video compression in feature space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1502–1511.

Hu, Z., Lu, G., Guo, J., Liu, S., Jiang, W., & Xu, D. (2022). Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5921–5930.

Iyer, V., Najafi, A., James, J., Fuller, S., & Gollakota, S. (2020). Wireless steerable vision for live insects and insect-scale robots. *Science robotics, 5*(44), 0839.

Jeon, S., Choi, K. P., Park, Y., & Kim, C.-S. (2023). Context-based trit-plane coding for progressive image compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14348–14357.

Ji, S., Pu, J., Lim, B. C., & Horowitz, M. (2016). A 220pj/pixel/frame cmos image sensor with partial settling readout architecture. In: 2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits), pp. 1–2. IEEE.

Josephson, C., Yang, L., Zhang, P., & Katti, S. (2019). Wireless computer vision using commodity radios. In: Proceedings of the 18th International Conference on Information Processing in Sensor Networks. pp. 229–240.

Khani, M., Sivaraman, V., & Alizadeh, M. (2021). Efficient video compression via content-adaptive super-resolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4521–4530.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I. and others (2021). Wilds: A benchmark of in-the-wild distribution shifts. In: International Conference on Machine Learning, pp. 5637–5664. PMLR.

Kwan, H. M., Gao, G., Zhang, F., Gower, A., & Bull, D. (2023). Hinerv: Video compression with hierarchical encoding-based neural representation. *Advances in Neural Information Processing Systems, 36*, 72692–72704.

Lee, J.-H., Jeon, S., Choi, K. P., Park, Y., & Kim, C.-S. (2022). Dpict: Deep progressive image compression using trit-planes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16113–16122.

Lefebvre, M., Moreau, L., Dekimpe, R., & Bol, D. (2021). 7.7 a 0.2-to-3.6 tops/w programmable convolutional imager soc with in-sensor current-domain ternary-weighted mac operations for feature extraction and region-of-interest detection. In: 2021 IEEE International Solid-State Circuits Conference (ISSCC), vol. 64, pp. 118–120. IEEE.

Li, J., Li, B., & Lu, Y. (2023). Neural video compression with diverse contexts. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22616–22626.

Li, J., Li, B., & Lu, Y. (2024). Neural video compression with feature modulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 26099–26108.

Li, H., Li, S., Dai, W., Li, C., Zou, J., & Xiong, H. (2024). Frequency-Aware Transformer for Learned Image Compression. In: The Twelfth International Conference on Learning Representations. https://openreview.net/forum?id=HKGQDDTuvZ

Li, J., Li, D., Savarese, S., & Hoi, S. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: International Conference on Machine Learning, pp. 19730–19742. PMLR.

Li, Y., Zhang, H., Li, L., & Liu, D. (2025). Learned image compression with hierarchical progressive context modeling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 18834–18843.

Li, J., Li, B., & Lu, Y. (2021). Deep contextual video compression. *Advances in Neural Information Processing Systems, 34*, 18114–18125.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In: European Conference on Computer Vision, pp. 740–755. Springer.

Lin, J., Chen, W.-M., Lin, Y., Gan, C., Han, S., et al. (2020). Mcunet: Tiny deep learning on iot devices. *Advances in neural information processing systems, 33*, 11711–11722.

Liu, J., Sun, H., & Katto, J. (2023). Learned image compression with mixed transformer-cnn architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14388–14397.

Liu, J., Wang, S., Ma, W.-C., Shah, M., Hu, R., Dhawan, P., & Urtasun, R. (2020). Conditional entropy coding for efficient video compression. In: European Conference on Computer Vision, pp. 453–468. Springer.

Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., & Gao, Z. (2019). Dvc: An end-to-end deep video compression framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11006–11015.

Lu, J., Zhang, L., Zhou, X., Li, M., Li, W., & Gu, S. (2025). Learned image compression with dictionary-based entropy model. In: Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 12850–12859.

Mentzer, F., Toderici, G., Minnen, D., Hwang, S.-J., Caelles, S., Lucic, M., & Agustsson, E. (2022). Vct: A video compression transformer arXiv preprint arXiv:2206.07307.

Mercat, A., Viitanen, M., & Vanne, J. (2020). Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference, pp. 297–302 (2020)

Morishita, F., Kato, N., Okubo, S., Toi, T., Hiraki, M., Otani, S., Abe, H., Shinohara, Y., & Kondo, H. (2021). A cmos image sensor and an ai accelerator for realizing edge-computing-based surveillance camera systems. In: 2021 Symposium on VLSI Circuits, pp. 1–2. IEEE.

Naderiparizi, S., Hessar, M., Talla, V., Gollakota, S., & Smith, J.R. (2018). Towards {Battery-Free}{HD} video streaming. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), pp. 233–247.

Nakanoya, M., Narasimhan, S. S., Bhat, S., Anemogiannis, A., Datta, A., Katti, S., Chinchali, S., & Pavone, M. (2023). Co-design of communication and machine inference for cloud robotics. *Autonomous Robots, 47*(5), 579–594.

Pennebaker, W.B., & Mitchell, J.L. (1992). JPEG: Still image data compression standard. Springer Science & Business Media.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Rossi, D., Conti, F., Eggiman, M., Mach, S., Di Mauro, A., Guermandi, M., Tagliavini, G., Pullini, A., Loi, I., Chen, J., and others (2021) 4.4 a 1.3 tops/w@ 32gops fully integrated 10-core soc for iot end-nodes with 1.7 μw cognitive wake-up from mram-based state-retentive sleep mode. In: 2021 IEEE International Solid-State Circuits Conference (ISSCC), vol. 64, pp. 60–62. IEEE.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*(1), 1929–1958.

Sullivan, G. J., Ohm, J.-R., Han, W.-J., & Wiegand, T. (2012). Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology, 22*(12), 1649–1668.

TensorFlow (2023). TensorFlow Lite for Microcontrollers. https://github.com/tensorflow/tflite-micro. Accessed: [2023]

Toderici, G., Vincent, D., Johnston, N., Hwang, S.J., Minnen, D., Shor, J., & Covell, M. (2017). Full resolution image compression with recurrent neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5306–5314.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning, pp. 10347–10357. PMLR.

Van Rozendaal, T., Brehmer, J., Zhang, Y., Pourreza, R., Wiggers, A., & Cohen, T. S. (2021). Instance-adaptive video compression: Improving neural codecs by training on the test set, arXiv preprint arXiv:2111.10302.

Veluri, B., Pernu, C., Saffari, A., Smith, J., Taylor, M., & Gollakota, S. (2023). Neuricam: Key-frame video super-resolution and colorization for iot cameras. In: Proceedings of the 29th Annual International Conference on Mobile Computing and Networking. pp. 1–17.

Wang, H., Gan, W., Hu, S., Lin, J.Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., & Kuo, C.-C.J. (2016). Mcl-jcv: A jnd-based h.264/avc video quality assessment dataset. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 1509–1513. https://doi.org/10.1109/ICIP.2016.7532610

Wiegand, T., Sullivan, G. J., Bjontegaard, G., & Luthra, A. (2003). Overview of the h 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology, 13*(7), 560–576.

Workshop and Challenge on Learned Image Compression (CLIC). http://www.compression.cc (2020)

Xiang, J., Tian, K., & Zhang, J. (2023). Mimt: Masked image modeling transformer for video compression. In: The Eleventh International Conference on Learning Representations.

Xie, Y., Cheng, K. L., & Chen, Q. (2021). Enhanced invertible encoding for learned image compression. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 162–170

Xu, H., Li, Z., Lin, N., Wei, Q., Qiao, F., Yin, X., & Yang, H. (2020). Macsen: A processing-in-sensor architecture integrating mac operations into image sensor for ultra-low-power bnn-based intelligent visual perception. *IEEE Transactions on Circuits and Systems II: Express Briefs, 68*(2), 627–631.

Yang, R., Mentzer, F., Gool, L. V., & Timofte, R. (2020). Learning for video compression with hierarchical quality and recurrent enhancement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6628–6637.

Yao, S., Li, J., Liu, D., Wang, T., Liu, S., Shao, H., & Abdelzaher, T. (2020). Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems, pp. 476–488.

Zhu, Y., Yang, Y., & Cohen, T. (2022). Transformer-based transform coding. In: International Conference on Learning Representations.