Raw-JPEG Adapter: Efficient Raw Image Compression with JPEG

Mahmoud Afifi*, Ran Zhang[1], and Michael S. Brown[1]

AI Center-Toronto, Samsung Electronics

m.3afifi@gmail.com

{ran.zhang, michael.b1}@samsung.com

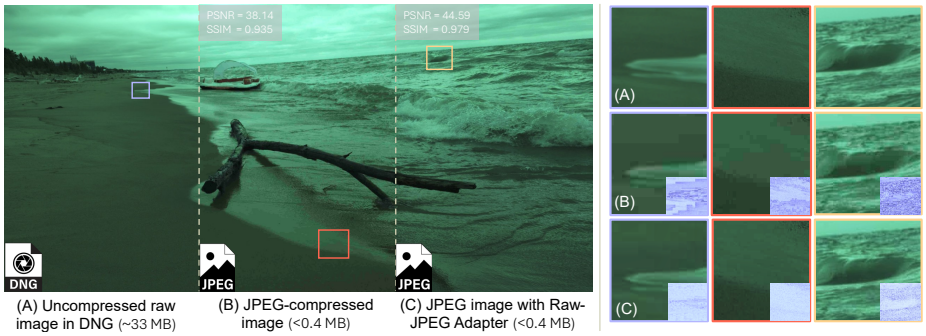

Fig. 1: We present Raw-JPEG Adapter, a lightweight, learnable pre-processing pipeline that adapts raw images before standard JPEG compression using spatial and optionally frequency domain transforms. The operations are fully invertible, with parameters fitting in the JPEG comment field (<64 KB), enabling accurate raw reconstruction after JPEG decoding and significantly reducing file size. In this figure, (A) shows the original raw (DNG), stored as JPEG with high compression (quality 25) without our method in (B), and with our method in (C). Error maps for (B) and (C) are shown on the right. All raw images shown in this paper are tone-mapped for visualization.

Abstract. Digital cameras digitize scene light into linear raw representations, which the image signal processor (ISP) converts into display-ready outputs. While raw data preserves full sensor information—valuable for editing and vision tasks—formats such as Digital Negative (DNG) require large storage, making them impractical in constrained scenarios. In contrast, JPEG is a widely supported format, offering high compression efficiency and broad compatibility, but it is not well-suited for raw storage. This paper presents Raw-JPEG Adapter, a lightweight, learnable, and invertible pre-processing pipeline that adapts raw images for standard JPEG compression. Our method applies spatial and optional frequency-domain transforms, with compact parameters stored in the JPEG comment field, enabling accurate raw reconstruction. Experiments across multiple datasets show that our method achieves higher fidelity than direct JPEG storage, supports other codecs, and provides a favorable trade-off between compression ratio and reconstruction accuracy.

* Work done while at AI Center-Toronto, Samsung Electronics

1 Introduction

Modern digital cameras capture incoming light and digitize it into linear raw sensor measurements, which are subsequently processed by the camera's image signal processor (ISP) to produce display-ready images (e.g., 8-bit sRGB). Raw data, however, is typically recorded at 12–14 bits per channel, preserving the full sensor information and offering greater flexibility for post-capture editing and computer vision applications [4, 10, 18, 22, 37].

Despite their benefits, raw images remain challenging to store in consumer workflows because of the absence of practical storage formats. The most widely used option is the Digital Negative (DNG) format [1], which encapsulates raw sensor data together with metadata. While DNG is lossless and widely supported in professional photography software, its file sizes are typically tens of megabytes per image, making it inefficient for large-scale storage or sharing. Alternative approaches include saving raw data in lossless 16-bit image formats such as PNG-16 or TIFF, which likewise preserve the original content but incur substantial storage overhead. Even with modern storage hardware, such file sizes are prohibitive for mobile capture, cloud synchronization, or large-scale datasets in computer vision research.

In principle, JPEG [17], the most widely used lossy image compression standard [14, 41], could be employed to reduce raw storage requirements. However, JPEG was fundamentally designed for 8-bit gamma-corrected images, whereas raw sensor data is linear and typically stored at higher bit-depths. Applying JPEG directly to raw signals results in severe degradation, including banding, clipping, and irreversible artifacts that preclude accurate raw reconstruction (see Fig. 1-B). By contrast, newer formats such as JPEG 2000, JPEG XL, and HEIC offer technical improvements but remain only sparsely adopted due to limited ecosystem and platform support [23, 28, 40, 42].

As a result, there remains no widely supported and efficient format for storing raw images at manageable file sizes without sacrificing fidelity. This motivates us to seek a solution that leverages the ubiquity of standard JPEG compression while addressing its shortcomings for raw storage. Such a solution should be accurate enough to overcome JPEG's quantization and banding limitations, yet lightweight, requiring minimal processing time for both encoding and decoding. Importantly, it should maintain compatibility with the standard decoding pipeline, requiring only lightweight modifications. To this end, we propose Raw-JPEG Adapter, a learnable, compact, and fully invertible pre-processing pipeline that adapts raw images for JPEG storage. At decoding time, the pre-processing is inverted using only lightweight operations—without any reliance on deep models—to reconstruct the original raw data with high fidelity. Our method is both efficient and effective, delivering substantial improvements in compression efficiency and reconstruction accuracy compared to existing alternatives.

Contributions: In this paper, we introduce a method for adapting raw images before JPEG storage, enabling high-fidelity reconstruction after decoding. Our method is a learnable and fully invertible pre-processing pipeline applied before standard JPEG compression. The pipeline learns the coefficients of invertible operators in a self-supervised manner, effectively mitigating JPEG artifacts and improving raw reconstruction accuracy. The method is lightweight ($\sim$37K parameters), incurs only $\sim$0.1s runtime over-

head, and requires less than 64 KB to store the learned coefficients, which are embedded directly in the JPEG comment (COM) segment. We extensively validate our method across multiple datasets, demonstrating strong generalization and significant improvements over raw pre-processing baselines and state-of-the-art raw reconstruction methods. Finally, we show that the Raw-JPEG Adapter can also be applied with alternative compression schemes, yielding consistent improvements. The code and trained models will be released upon acceptance.

2 Related work

The focus of this paper *is not* on developing a new raw compression algorithm, but rather on adapting existing compression schemes—particularly JPEG, the most widely adopted format in practice [21]—to better preserve raw data fidelity. To situate our work, we briefly review related research in three areas: 1) raw data compression, 2) raw image reconstruction, and 3) bidirectional neural ISPs.

2.1 Raw data compression

Prior work on raw image compression has largely focused on codec-specific engineering [8]. Such methods are commonly categorized as compression-first, which operate directly on mosaiced color filter array (CFA) data before coding, or demosaicing-first, which first convert to RGB and then apply handcrafted chroma subsampling or luma adjustment schemes [7–9, 24, 25]. Among demosaicing-first approaches, the Optimal Luma Modification (OLM) method adjusts the luma component so that, after upsampling and RGB conversion, the reconstructed image better matches the original [9]. Another line of work adapts the JPEG XS codec [12] for Bayer CFA data by replacing its component decorrelation stage with a nonlinear point transform and the Star-Tetrix transform, which approximates gamma-like behavior (2.2) while remaining hardware-friendly [36].

Although effective, these approaches are mainly driven by real-time video transport and streaming use cases (e.g., broadcast, HDMI/SDI, VR/AR), where low latency and codec compatibility outweigh storage efficiency. In contrast, we focus on compact raw image storage for photography and computer vision, where archiving, portability, and ecosystem support are key. Our Raw-JPEG Adapter provides a lightweight, learnable, and invertible pre-processing pipeline that works seamlessly with standard JPEG, offering a storage-oriented alternative.

2.2 Raw image reconstruction

Raw image reconstruction methods aim to recover raw sensor data in post-capture from display-referred images (e.g., sRGB) [33]. This line of work can be viewed as a form of compressed raw storage, where additional side information is stored to enable recovery from standard 8-bit images. The nature, scale, and role of this auxiliary information vary substantially across approaches. Early works operated under the 64 KB JPEG comment

constraint [31, 32], using the side information to compensate for irreversible ISP operations applied before JPEG encoding. Subsequent methods stored sparse raw samples (<2%) to assist reconstruction [20, 30, 34]. More recent learning-based approaches rely on richer latent features, often requiring 1–2 MB of auxiliary data per image in addition to the sRGB image itself [44, 45]. In these settings, metadata plays a central role in reconstructing raw data from ISP-rendered images, as it is required to compensate for the irreversible transformations introduced by tone mapping, gamma correction, and other non-linear ISP operations.

Our method shares the high-level goal of enabling storage-efficient raw preservation, but differs fundamentally in formulation. Rather than reconstructing raw from ISP-processed sRGB with substantial auxiliary data, we bypass sRGB rendering and store the raw image directly in the JPEG format. The JPEG quality parameter provides a standard user-controlled size–quality trade-off. We additionally embed a small amount of side information (well below 64 KB) in the COM segment; however, this information consists only of coefficients of the forward pre-processing operators that adapt raw data to JPEG quantization behavior. It is not used to compensate for irreversible ISP operations, nor does it encode latent representations of the image. Raw-JPEG therefore preserves raw content directly, while using lightweight operator coefficients solely to improve robustness to quantization, resulting in a simple and storage-efficient pipeline.

2.3 Bidirectional neural ISPs

Camera ISPs apply a sequence of highly nonlinear operators to raw images to generate display-referred outputs, typically 8-bit sRGB [11]. Neural ISPs learn to approximate this pipeline using deep neural networks [15, 16, 35, 50]. Some work extends this idea by proposing bidirectional frameworks [2, 49], enabling an approximate inversion of the ISP to reconstruct raw data from sRGB without storing additional metadata. These methods either jointly train two networks to minimize both the display-image rendering loss and the raw reconstruction loss [2], or design invertible neural ISP blocks that explicitly allow reversing the rendering operations [48].

While these approaches provide a path to raw recovery from compact storage (i.e., saving only the display image in formats such as JPEG), the reconstructed raw is typically limited in accuracy due to the highly nonlinear operations applied during neural ISP rendering. In contrast, our method bypasses ISP rendering entirely: we store the raw data itself, transformed only by a lightweight pipeline before JPEG compression. This enables higher reconstruction fidelity while still keeping storage requirements affordable (see Fig. 1-C).

3 Method

Given an input RGB demosaiced raw image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, after applying black-level normalization, our goal is to compress the image using a standard image codec (with a primary focus on JPEG), such that:

$$\mathbf{I}' = \texttt{Dec}\left(\texttt{Enc}\left(\mathbf{I}\right)\right) \approx \mathbf{I}, \tag{1}$$

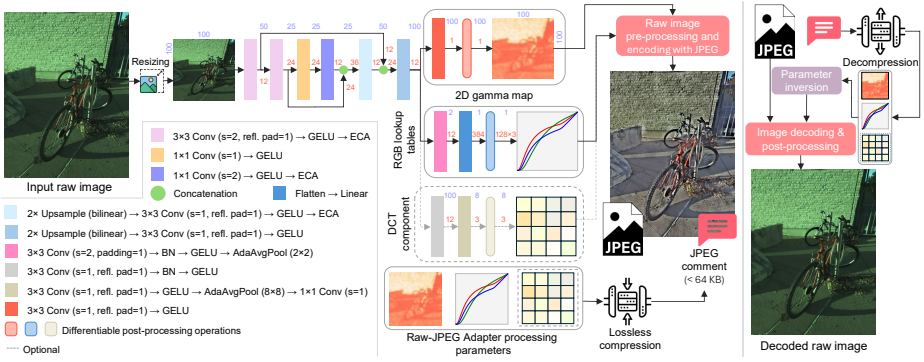

Fig. 2: Our Raw-JPEG Adapter uses a lightweight network (∼37K parameters) to process a thumbnail of the raw image and produce parameters for a pixel-wise gamma operator, RGB 1D tone-mapping lookup tables, and an optional DCT-based component applied globally in the frequency domain over 8×8 blocks. These transformations are applied before saving the image as a JPEG, while the associated parameters are compressed and embedded in the JPEG file's comment (COM) segment (<64 KB). All intermediate feature dimensionalities are annotated in the figure, with channel counts shown in red and spatial dimensions in purple. At decoding time, the parameters are retrieved, inverted, and applied to the stored image to reconstruct the original raw content. During training, the JPEG step is replaced with a differentiable simulator, and the network is optimized in a self-supervised manner.

where $\texttt{Enc}(\cdot)$ and $\texttt{Dec}(\cdot)$ denote the encoder and decoder of the compression scheme, and $\mathbf{I}'$ is the reconstructed raw image. Since our design centers on JPEG, we first highlight the challenges of directly using JPEG for raw storage. Raw sensor data is typically recorded at 12–14 bits per channel, with most values concentrated in the lower intensity range due to the linear sensor response. The data also exhibits strong color casts arising from sensor sensitivity bias and scene illumination, both of which remain uncorrected at the raw stage. In contrast, JPEG is designed for 8-bit per channel RGB images in display-referred space (sRGB). Its pipeline applies color space conversion, chroma sub-sampling, blockwise 8×8 discrete cosine transform (DCT), and quantization using tables tuned for sRGB image statistics.

Directly saving raw data with JPEG therefore compresses 12–14 bits information to 8 bits, subjects the data to suboptimal bucket sizes misaligned with raw distributions. While the JPEG quality parameter $Q \in [1, 100]$ controls the aggressiveness of quantization—higher Q values reduce loss at the cost of larger file sizes—this trade-off is not well suited to raw images. Even at the highest setting ($Q = 100$), banding in dark regions, clipping of highlights, and color distortions often remain because the quantization tables and color transforms are not adapted to sensor-domain data. Moreover, at lower Q values, these artifacts become even more pronounced, severely degrading downstream rendering quality (see Fig. 1-B).

To improve the fidelity of the reconstructed raw image, we introduce invertible preprocessing operators applied to the raw image $\mathbf{I}$ before compression and inverted after decoding. Accordingly, Eq. 1 is updated as:

$$\hat{\mathbf{I}} = F^{-1}\left(\texttt{Dec}\left(\texttt{Enc}\left(F\left(\mathbf{I};\theta\right)\right);\theta\right)\right), \tag{2}$$

where F denotes the invertible pre-processing operators, parameterized by coefficients θ, and $\hat{\mathbf{I}}$ is the reconstructed raw image by our method. These coefficients are compactly serialized using zlib compression with Base64 encoding and embedded in the JPEG comment segment, constrained to remain under 64 KB.

3.1 Pre-encoding

Before JPEG compression, the raw image is processed by a pipeline of invertible pre-processing operators. The parameters of these operators are predicted by a learnable network (Sec. 3.3). The pipeline balances expressiveness with minimal storage overhead and negligible decoding cost. It consists of three components:

Channel-wise 1D look-up tables (LuTs): 1D LuTs are applied to each of the three color channels in the demosaiced raw image to modulate the global tone and intensity distribution. Each LuT is constrained to 128×1 values:

$$\mathbf{I}^{\texttt{LuT}}_{c(x,y)} = \texttt{LuT}_c\big(\mathbf{I}_{c(x,y)}\big), \quad c \in \{R, G, B\}. \tag{3}$$

Blockwise DCT component scaling: The image is split into non-overlapping 8×8 blocks, and a single global 8×8 scaling matrix $\mathbf{S}$ is applied element-wise to the DCT coefficients of each block. The modified coefficients are then transformed back to the spatial domain via the inverse DCT:

$$\mathbf{I}^{\texttt{DCT}}_b = \texttt{DCT}^{-1}(\mathbf{S} \odot \texttt{DCT}(\mathbf{I}^{\texttt{LuT}}_b)), \tag{4}$$

where b indexes the image blocks and $\odot$ denotes element-wise multiplication. Since training is performed on data from a specific camera, $\mathbf{S}$ implicitly adapts to that camera's characteristics (e.g., sensor noise profile and color distribution). This frequency-domain adjustment is useful because JPEG applies fixed quantization tables that are not optimized for raw distributions. By selectively rescaling DCT coefficients before compression, $\mathbf{S}$ helps align the transformed raw data with JPEG's quantization behavior. A drawback, however, is that $\mathbf{S}$ encodes biases specific to the training camera, limiting generalization to new camera with different characteristics. For completeness, we therefore report results both with and without this component in our camera-specific and cross-camera evaluations (Sec. 4.1).

Pixel-wise gamma mapping: A 100×100 gamma map $\boldsymbol{\Gamma}$ stores local gamma scalar factors, which are bilinearly upsampled to the input resolution and applied in a pixel-wise manner:

$$\mathbf{I}^{\Gamma}_{(x,y)} = \mathbf{I}^{\texttt{DCT}}_{(x,y)}{}^{\boldsymbol{\Gamma}_{(x,y)}}. \tag{5}$$

Our pre-encoding pipeline is designed to overcome JPEG's bottlenecks. The 1D LuTs adjust tonal balance and redistribute intensity values, improving robustness to quantization from bit-depth reduction. The DCT scaling adapts frequency-domain statistics to camera characteristics, mitigating the mismatch with JPEG's quantization tables,

which were tuned for natural images in display space. Finally, the pixel-wise gamma map applies local nonlinear corrections that help alleviate banding in dark regions, reduce color shifts, and further compensate for information loss due to reduced bit-depth. Together, these operators address the key limitations of applying JPEG to raw data, enabling high-fidelity reconstruction while remaining lightweight and fully invertible. Note that the DCT component is optional, as it naturally encodes camera-specific characteristics. If omitted, $\mathbf{I}^{\texttt{DCT}}$ is replaced with $\mathbf{I}^{\texttt{LuT}}$ in Eq. 5.

3.2 Post-decoding

At decoding time, we apply the inverse sequence of operations to reconstruct the raw image from the JPEG output. Given a decoded JPEG image $\mathbf{I}'$, the reconstruction proceeds in three steps:

Pixel-wise gamma inversion: The gamma map $\boldsymbol{\Gamma}$ is bilinearly upsampled to the resolution of $\mathbf{I}'$, and an element-wise inverse gamma correction is applied:

$$\mathbf{I}^{\Gamma}_{(x,y)} = \hat{\mathbf{I}}^{\,1/\boldsymbol{\Gamma}_{(x,y)}}_{(x,y)}. \tag{6}$$

Inverse blockwise DCT scaling: If the DCT scaling component was used during encoding, we invert it by dividing the blockwise DCT coefficients by the scaling matrix $\mathbf{S}$ before returning to the spatial domain:

$$\mathbf{I}^{\texttt{IDCT}}_{b} = \texttt{DCT}^{-1}\big(\texttt{DCT}(\mathbf{I}^{\Gamma}_{b}) \oslash \mathbf{S}\big)\,, \tag{7}$$

where $\oslash$ denotes element-wise division, b indexes the 8×8 blocks, and $\mathbf{S}$ is the global scaling matrix estimated during encoding.

Inverse channel-wise 1D LuTs: Finally, channel-wise inverse LuTs are applied to restore the original distribution of each color channel:

$$\hat{\mathbf{I}}_{c(x,y)} = \texttt{LuT}^{-1}_{c}\Big(\mathbf{I}^{\texttt{IDCT}}_{c(x,y)}\Big)\,, \quad c \in \{R, G, B\}. \tag{8}$$

If the DCT component is not used, $\mathbf{I}^{\texttt{IDCT}}$ is replaced by $\mathbf{I}^{\Gamma}$.

Conceptually, the post-decoding pipeline starts from the 8-bit quantized JPEG pixels and progressively restores the original raw domain. The inverse gamma step expands the limited 8-bit range back toward the higher dynamic range of raw data, counteracting banding introduced during quantization. The DCT scaling then re-balances frequency-domain statistics that JPEG's fixed quantization distorts, while the inverse LuTs recover the channel-wise tonal mapping applied at encoding. Together, these operations exactly invert the pre-processing steps. The parameters ($\boldsymbol{\Gamma}$, $\mathbf{S}$, $\texttt{LuTs}$) are compactly stored in the JPEG comment segment, and decoding requires only applying their inverse operators, without running any neural networks. As a result, the runtime overhead at decoding is negligible, while enabling high-fidelity reconstruction of the original raw image.

3.3 Network Design

The parameters of our three operators ($\boldsymbol{\Gamma}$, $\mathbf{S}$, and $\texttt{LuTs}$) are predicted by a compact network, illustrated in Fig. 2. The architecture follows a convolutional encoder-decoder design with GELU activations [13], ECA channel-attention blocks [43], and skip connections. The decoder branches into three lightweight heads, each predicting the parameters of one operator (two heads if the DCT component is omitted). The network operates on a thumbnail of the input image to produce parameters adapted to that image. To ensure stability and invertibility, the raw outputs are passed through differentiable post-processing that maps them into valid coefficient ranges.

Gamma map: The predicted gamma values are stabilized via:

$$\boldsymbol{\Gamma}_{(x,y)} = \exp\left(2\tanh(g_{\theta_{(x,y)}})\right), \tag{9}$$

where g_θ denotes the network output corresponding to the gamma parameters. This mapping constrains the predictions to the valid range $[0.14, 7.4]$, ensuring strictly positive exponents while avoiding degenerate or excessively large corrections.

DCT scaling matrix: Frequency-domain scaling is constrained as:

$$\mathbf{S} = \exp\left(0.7\tanh(s_\theta)\right), \tag{10}$$

where s_θ denotes the network output corresponding to the DCT scaling matrix. This mapping constrains the coefficients to the range $[0.5, 2.0]$, enabling controlled frequency amplification or attenuation while avoiding instability.

Channel-wise 1D LuTs: To guarantee monotonicity (and thus invertibility), each LuT is parameterized using unconstrained network outputs h_θ, which are mapped to positive increments via $\texttt{softplus}$. The cumulative sum then enforces a strictly increasing mapping. Specifically, the sampled LuT values $\mathbf{L}$ are constructed as:

$$\mathbf{L}(i) = \texttt{cumsum}(\texttt{softplus}(h_\theta))_i, \tag{11}$$

where i indexes the discretized entries of the LuT. The resulting $\mathbf{L}$ represents the response values of the channel-wise look-up tables, which are then min–max normalized to $[0, 1]$. This ensures that each channel-wise look-up table, $\texttt{LuT}_c$, represents a valid increasing mapping, while remaining differentiable and stable during training.

3.4 Training

Our framework is trained in a self-supervised manner. Given an uncompressed raw training image $\mathbf{I}$, we predict the operator parameters $(\mathbf{S}, \boldsymbol{\Gamma}, \texttt{LuTs})$ using the network in Sec. 3.3. The image is transformed with these operators (Sec. 3.1), passed through a differentiable JPEG simulator [48], and inverted (Sec. 3.2) to yield the reconstruction $\hat{\mathbf{I}}$. The supervision signal is simply the original input $\mathbf{I}$, enabling end-to-end training without external ground truth pairs.

To improve robustness across cameras with different sensor sensitivities and varying responses to incident light, we apply two lightweight augmentations during training (in

addition to standard geometric augmentations): 1) intensity scaling, where the image is multiplied by a random factor in $[0.85, 1.15]$ to simulate brightness variations, and 2) color augmentation, where a random 3×3 matrix is applied to the RGB channels. To mimic the bias of real camera response functions toward the green channel—arising from sensor spectral sensitivities—we add a small fixed offset to the green row of the random color transformation matrix before normalization (i.e., +0.05). This enforces a slight green boost, reflecting the fact that most cameras capture proportionally more green light.

During training, we optimize the composite loss:

$$\mathcal{L} = \lambda_{\text{L1}}\,\mathcal{L}_{\text{L1}} + \lambda_{\text{SSIM}}\,\mathcal{L}_{\text{SSIM}} + \lambda_{\text{FFT}}\,\mathcal{L}_{\text{FFT}}, \tag{12}$$

where $\mathcal{L}_{\text{L1}}$ enforces pixel fidelity, $\mathcal{L}_{\text{SSIM}}$ promotes structural similarity, and $\mathcal{L}_{\text{FFT}}$ ensures frequency-domain consistency by applying an ℓ_1 loss to the real and imaginary components of the 2D FFT of $\hat{\mathbf{I}}$ and $\mathbf{I}$.

4 Experiments

Table 1: Ablation study results for different variations of our proposed design, evaluated on the S24 test set [3] with JPEG quality set to 75. Best results are highlighted in **green**.

Gamma (100×100)	LuT (128×3)	DCT	ECA	JPEG sim.	Gamma (64×64)	LuT (64×3)	LuT (128×1)	L1 loss	SSIM loss	FFT loss	PSNR	SSIM (×100)
✓								✓	✓	✓	42.32	96.11
	✓							✓	✓	✓	40.93	95.56
✓	✓							✓	✓	✓	42.50	97.12
✓	✓	✓						✓	✓	✓	42.70	97.24
✓	✓	✓	✓					✓	✓	✓	42.80	97.19
✓	✓	✓	✓	✓				✓	✓	✓	**43.58**	**97.67**
	✓	✓	✓	✓	✓						43.56	97.64
✓		✓	✓	✓		✓					43.49	97.66
✓		✓	✓	✓			✓				43.42	97.63
✓	✓	✓	✓	✓				✓		✓	43.06	97.35
✓	✓	✓	✓	✓				✓	✓		43.34	97.59
✓	✓	✓	✓	✓					✓	✓	41.72	97.62

We trained our model on the S24 training set [3], which contains 2,619 raw images from the Samsung S24 Ultra main camera. Training ran for 100 epochs with Adam [19] (learning rate 0.001, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 10^{-4}). A cosine scheduler [27] decayed the rate to 10^{-5}. Non-overlapping 512×512 patches were used. Loss weights were $\lambda_{\text{L1}} = 1.0$, $\lambda_{\text{FFT}} = 0.1$, and $\lambda_{\text{SSIM}} = 0.1$.

As in-domain evaluation, we used the S24 test set (400 images). For cross-camera generalization, we tested on three external datasets: S7 (220 smartphone images) [38], MIT-Adobe FiveK (5,000 images from 35 DSLR cameras) [5], and NUS (1,736 images from 8 DSLR cameras) [6]. All Bayer raws were demosaiced with Menon's algorithm [29]. We report results at JPEG qualities 100, 95, 75, 50, and 25, training a separate model for each. Comparisons include invertible pre-processing alternatives, metadata-based methods, and bidirectional neural ISPs. Our method *is not* a new codec

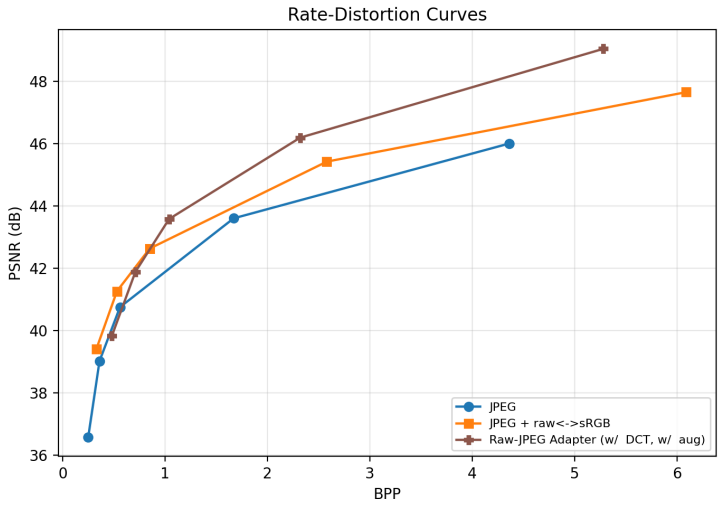

Fig. 3: Rate-Distortion Curves (PSNR vs. BPP) showing advantage of Raw-JPEG Adapter over competitive pre-processing baselines

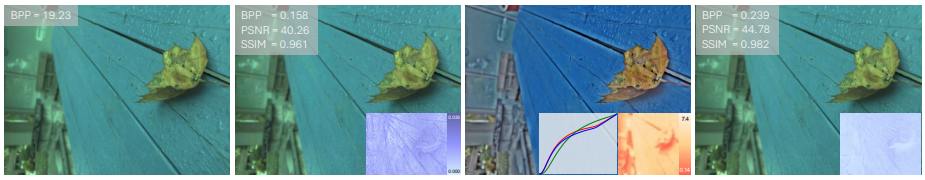

Fig. 4: Qualitative example from the S24 test set [3] at JPEG quality 25. (A) Uncompressed demosaiced raw image. (B) JPEG-compressed raw without pre-/post-processing, with its error map. (C) JPEG image produced by our Raw-JPEG Adapter using the predicted RGB LuTs and gamma map. (D) Decoded raw image by our Raw-JPEG Adapter, along with the corresponding error map.

but a fidelity-oriented pre-processing stage for JPEG, which naturally inherits JPEG's storage efficiency. Bits-per-pixel (BPP) is determined by the quality setting, but our Raw-JPEG Adapter consistently achieves higher fidelity than alternatives while maintaining comparable BPP and significantly outperforming direct JPEG on raw.

For metrics, we compute PSNR, SSIM, and MS-SSIM [46, 47] against uncompressed raw, and measure compression ratio (CR) relative to 16-bit uncompressed raw PNGs. We report results with and without the DCT component and color/intensity augmentations to assess their impact (see Table 1 for additional ablation studies).

4.1 Results

The results on the S24 test set [3] are shown in Table 2 for JPEG quality 100. This setting represents the best-case scenario for storing raw images with JPEG, as the quan-

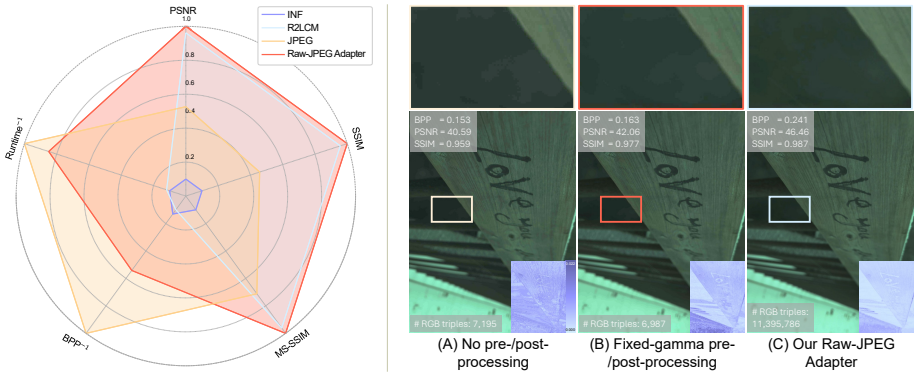

Fig. 5: Raw-JPEG Adapter improves raw image fidelity under JPEG compression while maintaining a favorable trade-off between compression efficiency, reconstruction quality, and runtime. Right: Visual comparison showing that, although our method significantly reduces artifacts, minor compression artifacts may still persist. (A) JPEG-decoded raw image without pre-/post-processing and its error map. (B) JPEG-decoded raw image with fixed 2.2 gamma applied before encoding and inverted after decoding, with its error map. (C) Decoded raw image produced by our method, along with the corresponding error map. Left: Comparison of compression methods in terms of normalized metrics (PSNR, SSIM, MS-SSIM, inverted BPP, and inverted runtime overhead) on the S24 dataset [3], including JPEG, INF [20], R2LCM [44, 45], and Raw-JPEG Adapter.

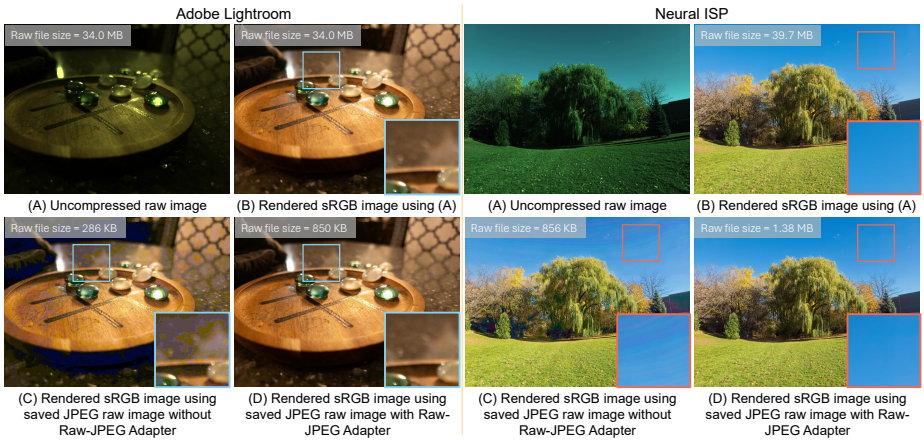

Fig. 6: On the left and right, we show sRGB images rendered by Adobe Lightroom and LiteISP [50], respectively, from: (B) the uncompressed raw image in (A) stored as a DNG file, (C) the JPEG-compressed raw image without our method, and (D) the JPEG-compressed raw image with our method. The raw JPEG images used to produce (C) and (D) were saved at quality level 50.

tization tables are scaled such that no quantization loss occurs in the DCT coefficients. We compare our method against three pre-processing baselines: 1) OLM [9], applied prior to JPEG compression; 2) raw↔sRGB mapping, where raw data are converted to

Table 2: Quantitative results on the S24 test set [3]. We report average PSNR, SSIM, and MS-SSIM [46, 47] between uncompressed raw images and decoded output at JPEG quality 100. Compression ratio (CR) is measured relative to the uncompressed 16-bit RGB PNG, and BPP corresponds to the JPEG file size. For our method, Raw-JPEG Adapter, we show results with and without the DCT component and our color/intensity augmentations. Best results are highlighted in **green**, and second-best results are shown in **bold**.

Method	Quality = 100				
	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	CR
JPEG	46.00	98.47	99.72	4.36	6.51
JPEG + OLM [9]	43.65	97.74	99.55	4.21	6.73
JPEG + raw↔sRGB	47.65	99.25	99.86	6.09	4.70
JPEG + fixed gamma	48.38	99.27	**99.87**	5.25	5.47
Raw-JPEG Adapter (w/o DCT, w/o aug)	48.49	**99.28**	**99.87**	5.28	5.43
Raw-JPEG Adapter (w/o DCT, w/ aug)	48.44	**99.28**	**99.87**	5.30	5.41
Raw-JPEG Adapter (w/ DCT, w/o aug)	**48.99**	**99.35**	**99.89**	5.28	5.40
Raw-JPEG Adapter (w/ DCT, w/ aug)	**49.04**	**99.35**	**99.89**	5.28	5.41

Table 3: BD-rate (%) of Raw-JPEG Adapter vs. competitive preprocessing baselines. Negative values indicate bitrate savings at equal PSNR.

Method	vs. JPEG	vs. JPEG + raw↔sRGB
Raw-JPEG Adapter (w/ DCT, w/ aug)	−27.88	−13.56

sRGB using white-balance gains, a color correction matrix (CCM) provided in each DNG file, and a fixed 2.2 gamma correction; and 3) fixed gamma correction, where a 2.2 gamma is applied before saving and inverted at decoding. For the raw↔sRGB baseline, the white-balance and CCM parameters are stored in the COM segment, similar to our method, and inverted at decoding time. Rate-distortion curve and RD-rate against competitive pre-processing baselines are shown in Fig. 3 and Table 3. For completeness, we also report results at additional JPEG quality levels in Table 4; see Fig. 4 for qualitative comparisons, with further results provided in the supp. material.

To ensure fair comparison under practical storage constraints, we further evaluate our method against standard JPEG compression under controlled bitrate settings. For each target BPP, we perform per-image rate control and evaluate at closely matched achieved bitrates. Under this setting, Raw-JPEG Adapter improves PSNR by approximately 2.6 dB on average compared to direct JPEG compression at matched BPP levels. Additional details are provided in Sec. 2.2 of the supp. material.

We further examine our method in combination with deep image compression by replacing the JPEG codec with a learning-based alternative. Specifically, we adopt the LIC-TCM method [26], a deep learning–based image compression model, in place of JPEG. We report results for our Raw-JPEG Adapter trained with a JPEG simulator, as well as after fine-tuning it with the pre-trained, fixed-weight LIC-TCM model. As shown in Table 5, our method achieves consistent improvements across different JPEG qualities and even when paired with deep image compression (see supp. material for a qualitative example). We also evaluated our model trained with the JPEG simulator on

Table 4: Quantitative results on the S24 test set [3] across different JPEG quality levels. For Raw-JPEG Adapter, we report results with and without DCT and color/intensity augmentations. Best results are highlighted in **green**, and second-best in **bold**.

Method	Quality = 25					Quality = 50					Quality = 75					Quality = 95				
	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	CR	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	CR	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	CR	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	CR
JPEG	36.58	90.08	95.79	0.25	121.15	39.02	93.57	97.74	0.36	86.57	40.75	95.40	98.62	0.56	56.42	43.60	97.21	99.35	1.67	17.78
JPEG + OLM [9]	36.53	90.09	95.77	0.25	121.87	38.90	93.53	97.71	0.36	87.43	40.33	95.17	98.56	0.55	57.20	42.42	96.84	99.25	1.64	17.98
JPEG + raw↔sRGB	39.41	95.25	98.39	0.33	92.99	41.26	96.58	99.07	0.53	60.23	42.63	97.37	99.39	0.85	37.15	45.42	98.52	99.62	2.58	11.83
JPEG + fixed gamma	38.30	94.51	97.82	0.30	101.32	40.74	96.20	98.81	0.47	68.00	42.43	97.16	99.25	0.73	43.06	45.42	98.42	99.65	2.15	14.06
Raw-JPEG Adapter (w/o DCT, w/o aug)	39.35	95.71	98.69	0.48	63.78	41.69	96.91	99.22	0.70	44.11	43.31	97.61	99.46	1.03	30.19	45.97	98.58	99.71	2.40	12.58
Raw-JPEG Adapter (w/o DCT, w/ aug)	39.49	95.70	98.69	0.47	65.13	41.73	96.87	99.20	0.69	45.10	43.36	97.60	99.46	1.01	30.75	46.12	**98.61**	**99.72**	2.46	12.29
Raw-JPEG Adapter (w/ DCT, w/o aug)	**39.70**	**95.83**	**98.77**	0.49	62.92	**41.92**	**96.99**	**99.26**	0.71	43.43	**43.54**	**97.68**	**99.49**	1.04	29.82	**46.22**	**98.60**	**99.72**	2.35	12.87
Raw-JPEG Adapter (w/ DCT, w/ aug)	**39.82**	**95.88**	**98.78**	0.48	63.84	**41.88**	**96.98**	**99.25**	0.71	43.73	**43.58**	**97.67**	**99.48**	1.04	29.85	**46.19**	98.58	**99.71**	2.32	13.02

Table 5: Quantitative results on the S24 test set [3] using the deep learning-based LIC-TCM method [26] for compression instead of JPEG. We evaluate our method with/without tuning for LIC-TCM. Best results are highlighted in **green**.

Pre-/post-processing	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	CR
None	42.95	96.56	99.03	0.31	160.47
Fixed gamma	43.55	97.38	99.32	0.63	90.27
Ours (w/o DCT)	45.11	98.12	99.58	1.16	42.14
Ours (w/ DCT)	45.11	98.12	99.58	1.16	42.14
Ours (w/o DCT) - tuned	45.43	98.53	99.61	1.17	41.52
Ours (w/ DCT) - tuned	**46.03**	**98.75**	**99.82**	1.26	40.97

JPEG 2000, where it likewise showed improvements; additional details are provided in the supp. material.

When comparing our method with alternative approaches, including bidirectional neural ISPs [2, 48] that reconstruct raw from sRGB, and metadata-based methods [20, 44, 45] that store auxiliary data alongside sRGB images, Table 6 shows that our method achieves the highest reconstruction accuracy while incurring minimal runtime overhead and maintaining comparable BPP.

In the results reported in Table 6, the bidirectional neural ISPs [2, 48] were trained on the S24 dataset [3]. Among the metadata-based methods, INF [20] does not require training, as it performs self-supervised fitting of an implicit neural function at test time. The R2LCM method [44, 45] was trained on the S24 dataset using paired JPEG-compressed sRGB images rendered with an advanced rendering pipeline that includes local tone mapping and complex color processing, along with the corresponding raw images. Both INF and R2LCM were evaluated on JPEG-compressed sRGB test images from the S24 dataset generated using the same rendering pipeline, reflecting practical imaging scenarios in which sRGB images are rendered through a complex rendering pipeline and subsequently JPEG-compressed. Additional discussion is provided in the supp. material (Sec. 1.4). A radar chart comparison is shown in Fig. 5, and qualitative examples are included in the supp. material.

We further evaluate our model trained on S24 images across different datasets to examine cross-camera generalization, as shown in Table 7. As described in Sec. 3, the DCT component is partially camera-specific, capturing sensor-dependent characteristics such as noise distribution. While this specialization can be advantageous for device-specific deployments, it generalizes less effectively across cameras. The results

Table 6: Comparisons with alternative methods on the S24 test set [3]. We compare Raw-JPEG Adapter against bidirectional neural ISPs [2, 48], metadata-based methods [20, 44, 45], and our method applied with JPEG or deep compression [26]. We report PSNR, SSIM, MS-SSIM [46, 47], BPP (total storage, with/without sRGB when applicable), and raw reconstruction runtime (seconds). Best results are in **green**, second-best in **bold**.

Method	PSNR	SSIM (×100)	MS-SSIM (×100)	BPP	Runtime overhead
CIE XYZ Net [2]	20.47	81.61	91.84	(2.340 / 0.000)	0.301
Invertible ISP [48]	43.71	98.40	99.56	(2.340 / 0.000)	7.932
INF [20]	41.24	96.29	98.56	(3.764 / 0.776)	25.93
R2LCM (drop 0) [44, 45]	46.01	98.49	99.67	(3.929 / 1.701)	2.815
R2LCM (drop 1) [44, 45]	44.8	98.34	99.61	(3.867 / 1.639)	2.786
R2LCM (drop 2) [44, 45]	38.42	96.28	98.55	(2.831 / 0.602)	2.261
R2LCM (drop 4) [44, 45]	33.65	94.56	96.28	(2.379 / 0.150)	2.023
JPEG + ours (quality = 50)	41.92	96.99	99.26	(0.713 / 0.713)	0.120
JPEG + ours (quality = 75)	43.58	97.67	99.48	(1.036 / 1.036)	0.120
JPEG + ours (quality = 95)	**46.22**	**98.60**	**99.72**	(2.345 / 2.345)	0.120
LIC-TCM [26] + ours	**46.03**	**98.75**	**99.82**	(1.260 / 1.260)	0.120

confirm this: the learned DCT component improves fidelity on the S24 dataset (the training camera; see Tables 2 and 4), but provides limited or no gains on other cameras, whereas the spatial-domain pipeline alone proves more robust. Qualitative examples are provided in the supp. material.

5 Conclusion and discussion

We presented a lightweight method that significantly improves raw image storage using JPEG, leveraging its lossy compression for small file sizes while enhancing fidelity and reconstruction accuracy. The results in Sec. 4.1 demonstrate consistent improvements over alternative approaches and strong generalization across datasets. Furthermore, we showed that our framework integrates seamlessly with alternative compression methods, while remaining lightweight with negligible runtime overhead.

Applications: Our method enables storing raw images at much smaller sizes while still supporting post-capture re-rendering and editing. To demonstrate this, we evaluated a pair of images from the S24 test set [3] compressed at JPEG quality 50, both with and without our method, and compared the rendered sRGB outputs against those obtained from uncompressed raw data. For rendering, we used LiteISP [50], trained on S24 raw images, and also tested Adobe Lightroom's raw-to-sRGB pipeline. Since Lightroom requires proprietary decoding, we adopted the approach of [39]: our decoded JPEGs (with and without our method) were stored as temporary DNGs by replacing the image data, which were then rendered to sRGB in Lightroom. As shown in Fig. 6, our method produces perceptually similar results to uncompressed DNGs (>30 MB) while reducing storage to under 2 MB. In contrast, directly storing raw data as JPEG introduces visible artifacts. Additional examples and discussion are provided in the supp. material.

Table 7: Quantitative results on cross-camera generalization to unseen cameras during training. We report results across different JPEG quality levels. Best results are highlighted in **green**.

Method	Quality = 25			Quality = 50			Quality = 75			Quality = 95		
	PSNR	SSIM (×100)	BPP	PSNR	SSIM (×100)	BPP	PSNR	SSIM (×100)	BPP	PSNR	SSIM (×100)	BPP
S7 dataset [38] (220 images taken by 1 smartphone camera)												
JPEG	37.54	85.81	0.18	39.58	90.02	0.27	40.94	92.55	0.47	43.84	96.41	1.75
Raw↔sRGB	38.29	91.75	0.33	39.13	93.45	0.64	39.95	95.05	1.16	42.76	98.00	3.70
Fixed gamma	39.77	91.20	0.29	41.26	93.13	0.52	42.46	94.85	0.93	46.38	98.08	2.98
Ours (w/ DCT)	39.86	**93.05**	0.51	40.85	**94.62**	0.86	42.15	**96.05**	1.38	44.99	98.21	3.18
Ours (w/o DCT)	**41.04**	92.98	0.49	**42.23**	94.56	0.81	**43.46**	95.98	1.34	**47.24**	**98.39**	3.37
MIT-Adobe 5K dataset [5] (5,000 images taken by 35 DSLR cameras)												
JPEG	38.96	91.13	0.19	41.90	95.02	0.25	43.91	96.69	0.35	46.38	97.83	1.14
Raw↔sRGB	38.42	96.26	0.25	39.69	97.04	0.38	40.43	97.46	0.63	41.73	98.17	2.18
Fixed gamma	41.88	96.66	0.23	44.42	97.72	0.33	46.09	98.27	0.53	48.96	99.05	1.80
Ours (w/ DCT)	40.96	**97.47**	0.36	42.47	**98.11**	0.52	43.99	98.53	0.78	45.73	99.11	1.95
Ours (w/o DCT)	**43.07**	97.42	0.35	**45.21**	98.10	0.50	**46.75**	**98.54**	0.76	**49.67**	**99.20**	2.08
NUS dataset [6] (1,736 images taken by 8 DSLR cameras)												
JPEG	37.92	89.56	0.21	40.54	93.78	0.29	42.32	95.83	0.42	44.59	97.15	1.22
Raw↔sRGB	40.10	96.97	0.30	41.77	97.87	0.44	42.86	98.34	0.66	44.49	98.88	1.92
Fixed gamma	41.26	96.46	0.28	43.58	97.61	0.41	45.16	98.19	0.62	47.68	98.84	1.80
Ours (w/ DCT)	41.48	**97.50**	0.50	43.17	**98.14**	0.68	44.64	**98.51**	0.93	46.25	98.92	1.97
Ours (w/o DCT)	**42.45**	97.41	0.49	**44.45**	98.06	0.66	**45.97**	98.49	0.91	**48.32**	**98.97**	2.11

Limitations and trade-offs: Despite its improvements, our method does not fully eliminate compression artifacts. Compared to direct JPEG decoding (Fig. 5-A) and the fixed 2.2 gamma baseline (Fig. 5-B), artifacts are clearly reduced, yet subtle distortions may still persist (Fig. 5-C). Our method introduces a small BPP overhead, since storing pre-processing parameters in the COM segment (typically $\sim$40 KB, always $<$64 KB) slightly increases file size. In practice, this overhead is negligible relative to uncompressed DNG files (tens of MBs) and is justified by the fidelity gains. Controlled bitrate experiments (Sec. 2.2 of the supp. material) further confirm that our method consistently outperforms direct JPEG compression at matched BPP levels. Furthermore, conventional BPP does not fully reflect perceptual quality: our method reconstructs a substantially wider range of unique RGB triples (i.e., richer tonal distributions) than baseline JPEGs, as shown in Fig. 5. To better capture this effect, Sec. 2.1 of the supp. material reports a weighted BPP metric that accounts for tonal diversity, providing a fairer measure of storage cost versus reconstruction quality. More broadly, the results in Fig. 5 highlight the central trade-off addressed in this work: enabling efficient raw storage at manageable file sizes while preserving high-fidelity reconstruction and maintaining compatibility with standard JPEG decoding. Our Raw-JPEG Adapter achieves a favorable balance among compression efficiency, reconstruction quality, and runtime overhead compared to existing alternatives, while requiring only lightweight processing during both encoding and decoding.

References

1. Adobe Systems Incorporated: Digital Negative (DNG) Specification (2004), latest version available at https://helpx.adobe.com/photoshop/digital-negative.html 2
2. Afifi, M., Abdelhamed, A., Abuolaim, A., Punnappurath, A., Brown, M.S.: CIE XYZ Net: Unprocessing images for low-level computer vision tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(9), 4688–4700 (2021) 4, 13, 14
3. Afifi, M., Zhao, L., Punnappurath, A., Abdelsalam, M.A., Zhang, R., Brown, M.S.: Time-aware auto white balance in mobile photography. In: ICCV (2025) 9, 10, 11, 12, 13, 14
4. Berdan, R., Besbinar, B., Reinders, C., Otsuka, J., Iso, D.: ReRAW: RGB-to-RAW image reconstruction via stratified sampling for efficient object detection on the edge. In: CVPR (2025) 2
5. Bychkovsky, V., Paris, S., Chan, E., Durand, F.: Learning photographic global tonal adjustment with a database of input/output image pairs. In: CVPR (2011) 9, 15
6. Cheng, D., Prasad, D.K., Brown, M.S.: Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution. Journal of the Optical Society of America A **31**(5), 1049–1058 (2014) 9, 15
7. Chiu, Y.H., Chung, K.L., Lin, C.H.: An improved universal subsampling strategy for compressing mosaic videos with arbitrary RGB color filter arrays in H. 264/AVC. Journal of Visual Communication and Image Representation **25**(7), 1791–1799 (2014) 3
8. Chung, K.L., Chen, H.Y., Hsieh, T.L., Chen, Y.B.: Compression for bayer CFA images: Review and performance comparison. Sensors **22**(21), 8362 (2022) 3
9. Chung, K.L., Hsu, T.C., Huang, C.C.: Joint chroma subsampling and distortion-minimization-based luma modification for RGB color images with application. IEEE Transactions on Image Processing **26**(10), 4626–4638 (2017) 3, 11, 12, 13
10. Cui, Z., Harada, T.: RAW-adapter: Adapting pre-trained visual model to camera RAW images. In: ECCV (2024) 2
11. Delbracio, M., Kelly, D., Brown, M.S., Milanfar, P.: Mobile computational photography: A tour. Annual review of vision science **7**(1), 571–604 (2021) 4
12. Descampe, A., Richter, T., Ebrahimi, T., Foessel, S., Keinert, J., Bruylants, T., Pellegrin, P., Buysschaert, C., Rouvroy, G.: JPEG XS—a new standard for visually lossless low-latency lightweight image coding. Proceedings of the IEEE **109**(9), 1559–1577 (2021) 3
13. Hendrycks, D., Gimpel, K.: Gaussian error linear units (GELUs). arXiv preprint arXiv:1606.08415 (2016) 8
14. Hudson, G., Léger, A., Niss, B., Sebestyén, I.: JPEG at 25: Still going strong. IEEE MultiMedia **24**(2), 96–103 (2017) 2
15. Ignatov, A., Malivenko, G., Timofte, R., Tseng, Y., Xu, Y.S., Yu, P.H., Chiang, C.M., Kuo, H.K., Chen, M.H., Cheng, C.M., et al.: PyNet-V2 mobile: Efficient on-device photo processing with neural networks. In: ICPR 4
16. Ignatov, A., Sycheva, A., Timofte, R., Tseng, Y., Xu, Y.S., Yu, P.H., Chiang, C.M., Kuo, H.K., Chen, M.H., Cheng, C.M., et al.: MicroISP: processing 32mp photos on mobile devices with deep learning. In: ECCV (2022) 4
17. ISO/IEC JTC 1/SC 29: Information technology – digital compression and coding of continuous-tone still images: Requirements and guidelines (1994), iSO/IEC 10918-1:1994, ITU-T Recommendation T.81 (1992) 2
18. Kee, E., Pikielny, A., Blackburn-Matzen, K., Levoy, M.: Removing reflections from RAW photos. In: CVPR (2025) 2
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 9

20. Li, L., Qiao, H., Ye, Q., Yang, Q.: Metadata-based raw reconstruction via implicit neural functions. In: CVPR (2023) 4, 11, 13, 14
21. Li, P., Lo, K.T.: Survey on JPEG compatible joint image compression and encryption algorithms. IET Signal Processing **14**(8), 475–488 (2020) 3
22. Li, Z., Lu, M., Zhang, X., Feng, X., Asif, M.S., Ma, Z.: Efficient visual computing with camera raw snapshots. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(7), 4684–4701 (2024) 2
23. Liam Proven: Google kills forthcoming JPEG XL image format in chromium (2022), https://www.theregister.com/2022/10/31/jpeg_xl_axed_chrome/, accessed: 2026-02-17 2
24. Lin, C.H., Chung, K.L., Yu, C.W.: Novel chroma subsampling strategy based on mathematical optimization for compressing mosaic videos with arbitrary RGB color filter arrays in H. 264/AVC and HEVC. IEEE Transactions on Circuits and Systems for Video Technology **26**(9), 1722–1733 (2015) 3
25. Lin, T.L., Yu, Y.C., Jiang, K.H., Liang, C.F., Liaw, P.S.: Novel chroma sampling methods for CFA video compression in AVC, HEVC and VVC. IEEE Transactions on Circuits and Systems for Video Technology **30**(9), 3167–3180 (2019) 3
26. Liu, J., Sun, H., Katto, J.: Learned image compression with mixed transformer-CNN architectures. In: CVPR (2023) 12, 13, 14
27. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 9
28. Lowe, D.B., Bennett, M.J.: A status report on JPEG 2000 implementation for still images: The uconn survey. In: Archiving Conference. vol. 6, pp. 209–212 (2009) 2
29. Menon, D., Andriani, S., Calvagno, G.: Demosaicing with directional filtering and a posteriori decision. IEEE Transactions on Image Processing **16**(1), 132–141 (2006) 9
30. Nam, S., Punnappurath, A., Brubaker, M.A., Brown, M.S.: Learning sRGB-to-raw-RGB derendering with content-aware metadata. In: CVPR. pp. 17704–17713 (2022) 4
31. Nguyen, R.M., Brown, M.S.: RAW image reconstruction using a self-contained srgb-jpeg image with only 64 KB overhead. In: CVPR (2016) 4
32. Nguyen, R.M., Brown, M.S.: Raw image reconstruction using a self-contained sRGB–JPEG image with small memory overhead. International Journal of Computer Vision **126**(6), 637–650 (2018) 4
33. Punnappurath, A., Brown, M.S.: Learning raw image reconstruction-aware deep image compressors. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(4), 1013–1019 (2019) 3
34. Punnappurath, A., Brown, M.S.: Spatially aware metadata for raw reconstruction. In: WACV (2021) 4
35. Ren, Y., Jiang, H., Yang, M., Li, W., Liu, S.: ISPDiffuser: Learning RAW-to-sRGB mappings with texture-aware diffusion models and histogram-guided color consistency. In: AAAI (2025) 4
36. Richter, T., Fößel, S., Descampe, A., Rouvroy, G.: Bayer CFA pattern compression with JPEG XS. IEEE Transactions on Image Processing **30**, 6557–6569 (2021) 3
37. Schewe, J.: The digital negative: Raw image processing in Lightroom, Camera Raw, and Photoshop. Peachpit Press (2012) 2
38. Schwartz, E., Giryes, R., Bronstein, A.M.: DeepISP: Toward learning an end-to-end image processing pipeline. IEEE Transactions on Image Processing **28**(2), 912–923 (2018) 9, 15
39. Seo, D., Punnappurath, A., Zhao, L., Abdelhamed, A., Tedla, S.K., Park, S., Choe, J., Brown, M.S.: Graphics2RAW: Mapping computer graphics images to sensor raw images. In: ICCV (2023) 14

40. SharinPix: What is the HEIC format and what are its limits? (2024), https://www.sharinpix.com/blog/what-is-the-heic-format-and-what-are-its-limits, accessed: 2026-02-17 2
41. Smith, E.: Why JPEGs still rule the web: 30 years ago, the standard became the dominant way we share digital photos. IEEE Spectrum **62**(8), 42–47 (2025) 2
42. Sneyers, J.: The case for JPEG XL (2022), https://cloudinary.com/blog/the-case-for-jpeg-xl, accessed: 2026-02-17 2
43. Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., Hu, Q.: ECA-Net: Efficient channel attention for deep convolutional neural networks. In: CVPR (2020) 8
44. Wang, Y., Yu, Y., Yang, W., Guo, L., Chau, L.P., Kot, A.C., Wen, B.: Raw image reconstruction with learned compact metadata. In: CVPR (2023) 4, 11, 13, 14
45. Wang, Y., Yu, Y., Yang, W., Guo, L., Chau, L.P., Kot, A.C., Wen, B.: Beyond learned metadata-based raw image reconstruction. International Journal of Computer Vision **132**(12), 5514–5533 (2024) 4, 11, 13, 14
46. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004) 10, 12, 14
47. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers (2003) 10, 12, 14
48. Xing, Y., Qian, Z., Chen, Q.: Invertible image signal processing. In: CVPR (2021) 4, 8, 13, 14
49. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: CycleISP: Real image restoration via improved data synthesis. In: CVPR (2020) 4
50. Zhang, Z., Wang, H., Liu, M., Wang, R., Zhang, J., Zuo, W.: Learning raw-to-sRGB mappings with inaccurately aligned supervision. In: ICCV (2021) 4, 11, 14