A Little Bit More: Bitplane-Wise Bit-Depth Recovery

Abhijith Punnappurath and Michael S. Brown, Senior Member, IEEE

Abstract—Imaging sensors digitize incoming scene light at a dynamic range of 10–12 bits (i.e., 1024–4096 tonal values). The sensor image is then processed onboard the camera and finally quantized to only 8 bits (i.e., 256 tonal values) to conform to prevailing encoding standards. There are a number of important applications, such as high-bit-depth displays and photo editing, where it is beneficial to recover the lost bit depth. Deep neural networks are effective at this bit-depth reconstruction task. Given the quantized low-bit-depth image as input, existing deep learning methods employ a single-shot approach that attempts to either (1) directly estimate the high-bit-depth image, or (2) directly estimate the residual between the high- and low-bit-depth images. In contrast, we propose a training and inference strategy that recovers the residual image bitplane-by-bitplane. Our bit-plane-wise learning framework has the advantage of allowing for multiple levels of supervision during training and is able to obtain state-of-the-art results using a simple network architecture. We test our proposed method extensively on several image datasets and demonstrate an improvement from 0.5dB to 2.3dB PSNR over prior methods depending on the quantization level.

Index Terms—Bit depth, bitplane, quantization, image restoration

1 INTRODUCTION

The term bit depth refers to the number of bits used per color channel to encode the tonal values of an image. Most modern camera sensors capture images with bit depths between 10 and 12 bits (representing 1024 and 4096 tonal values, respectively). The captured sensor image is then processed onboard the camera to transform it to a standard RGB (sRGB) image that is compatible with prevailing encoding standards [1]. The final step in the in-camera processing pipeline is to quantize the sRGB image into an 8-bit-per-channel format to match the bit depth of consumer displays and printers, the vast majority of which are 8 bit.

Bit-depth recovery is the task of recovering the bits lost due to quantization. An example is shown in Fig. 1. Bit-depth recovery has important applications in high-bit-depth (HBD) displays and photo editing. While most monitors are still 8 bit, 10-bit display devices are becoming increasingly popular due to ever-growing consumer demand for finer and more nuanced color tones. Many TVs, and smart phones such as Samsung Galaxy S10 [2] and iPhone X [3], already support 10-bit displays to meet high dynamic range (HDR) standards. However, these displays remain under-utilized as most image and video content available is still 8 bit. Artifacts are observed if the bit depth is not restored before 8-bit data is displayed on 10-bit monitors. Likewise, restoring the bit depth prior to photo editing reduces artifacts due to the availability of additional tonal values.

Several classical methods (e.g., [4], [5], [6], [7], [8], [9]) for bit-depth recovery exist in the literature. More recently, deep learning algorithms [10], [11], [12], [13], [14], [15], [16] have been proposed. The strategy employed by most deep learning methods (e.g., [10], [12], [13], [14], [15], [16]) is to train the network using the quantized low-bit-depth (LBD) image as input and have the network predict the high-bit-depth image as output. A more recent alternative strategy [11] is to train the network to predict the residual between the ground truth HBD image and the quantized LBD image. At test time, this output residual image is added to the quantized LBD image to produce the desired HBD image. While these two strategies are equivalent, it is argued that from a deep neural network (DNN) training perspective, the latter is more efficient and provides faster convergence. We adopt this latter formulation in our work. However, as opposed to [11], who estimate the residual in a single shot, we propose to recover the residual image corresponding to the lost bits one bitplane at a time.

Because the numerical magnitude of a pixel’s tonal value decreases from the most significant bit to the least significant bit, the most significant bits dominate loss functions based
on mean squared error (MSE) or peak signal-to-noise ratio (PSNR). As a result, approaches that predict either the HBD image or the residual using single-shot training fail to capture the very fine details encoded by the lower-order bits. This observation is the impetus for our bitplane-wise training strategy that aims to overcome this limitation of existing methods.

Contributions Our proposed method is based on the observation that the residual between the high-bit-depth ground truth image and the quantized low-bit-depth input image can be expressed as a weighted summation of the bitplanes lost during quantization. Each bitplane is a binary map, and thus, the original task of predicting the residual can be reformulated into a series of better-constrained binary image segmentation problems. We train a separate neural network independently to predict each bitplane. This makes our method agnostic to the relative magnitude of the bit position, and overcome the limitation of the single-shot training strategy employed by existing approaches. The input images for training each network can be generated from the ground truth HBD image by applying the appropriate level of quantization. This multi-level-supervised training strategy outperforms state-of-the-art techniques using a simple network architecture. We compare our method against several classical as well as deep learning algorithms on five image datasets under a range of quantization levels and demonstrate that our method advances the state-of-the-art by a significant margin.

Scope Our method is focused strictly on bit-depth quantization and does not consider other strategies often used in bit-depth reduction, such as image half-toning [17], or color palette reduction, such as in GIF [18]. In addition, our method is not attempting inverse-tone mapping [19], [20], which is an image enhancement technique targeted at producing plausible high-dynamic-range images for HBD display from low-dynamic-range input data. In contrast, the aim of bit-depth recovery, and our algorithm in particular, is image restoration where the goal is to accurately recover the actual bits lost due to linear quantization.

2 RELATED WORK

Early work on bit-depth recovery was based on simple rules to fill the missing bits. Multiplication by an ideal gain is the most straightforward method, wherein the LBD image is simply multiplied by an ideal gain factor. Bit replication [4] fills the lost bits with copies of the current LBD bits. Although these methods are fast, they ignore the spatial characteristics of the image and produce visual artifacts. The minimum risk-based classification method [5] constructs prediction error histograms and assigns the value with the minimum associated risk as the HBD pixel intensity at that location. Interpolation-based methods [3], [7] have also been proposed. The contour region reconstruction algorithm [6] linearly interpolates pixel values by analyzing distances from upward contour edges and downward contour edges. A drawback of this method is that it fails in regions with local extrema. The content adaptive technique in [7] tackles this issue through the use of a virtual skeleton marking algorithm which converts the problematic 2D extrapolation in local extrema regions into simple 1D interpolation. Optimization-based approaches, such as [8], [21], are also popular. With this approach, the bit-depth recovery problem is formulated as a maximum a posteriori (MAP) estimation in [21]. The ACDC algorithm [8] applies graph signal processing to the bit-depth recovery task. The AC component of the desired HBD signal is first calculated using a MAP formulation, and the DC component is then computed using the AC estimate by applying a minimum MSE criterion. The IPAD technique [9] employs an intensity potential field to model the spatial correlation among the LBD pixels. The HBD image is recovered using a context-adaptive dequantization procedure that leverages this potential field.

Recently, deep learning-based approaches have gained in popularity [10], [11], [12], [13], [14], [15], [16]. Works by Hou and Qiu [14] and GG-DCNN [15] employ a U-Net style architecture to predict the HBD image given the LBD image. Their methods aim to restore very low bit depth images (e.g., 2 bits). The networks of BE-CNN [10] and Liu et al. [16] comprise a chain of deconvolution layers with long and short skip connections. BitNet [12] uses an encoder-decoder architecture with dilated convolutions and multi-scale feature integration. BE-CALF [11] uses a chain of convolutional-deconvolutional layers with dense concatenations of all level features. BDEN [13] uses a two-stream architecture, one for flat and another for non-flat regions, with a local adaptive adjustment preprocessing step for the flat areas. Their approach is tailored towards relatively smaller expansions of the bit depth – 6 to 8 bit, 8 to 10 bit – while other methods [10], [11], [12] target larger expansions – 4 to 16 bit, 6 to 16 bit. Different deep learning methods use different loss functions to train their models; MSE, mean absolute error (MAE), VGG features and so forth are the most common. It can be seen that existing deep learning methods resort to deeper and more complex architectures (U-Net style encoder-decoder, long-short skip connections, multi-stream networks, dense feature concatenations) to improve performance. In the following sections, we demonstrate how we can outperform these existing techniques across a variety of bit-depth ranges using a standard ResNet-style network architecture. This is made possible by our bitplane-wise training strategy, explained in the following section.

3 PROPOSED METHOD

In this section, we first analyze the bit-depth recovery problem from a bitplane perspective. We then introduce our bitplane-wise training and testing strategy. Our network architecture is discussed in Sec. 3.1.

Let the ground truth HBD image be denoted by O. Assume O has a bit depth of N. To quantize O to q bits, the following formula can be applied [14], [15], [23]:

$$I_q = \left\lfloor \frac{O}{2^{(N-q)}} \right\rfloor 2^{(N-q)},$$

(1)

where $I_q$ represents the q-bit quantized and zero padded LBD image, and $\lfloor . \rfloor$ represents the floor operation. Existing deep learning algorithms, such as [10], [12], [13], [14], [15], [16], train using ($I_q$, O) pairs as input and target. We can also express O as the summation of $I_q$ and a residual image R:

$$O = I_q + R.$$  

(2)
Existing deep learning approaches for bit-depth recovery

Equivalent interpretation in terms of bitplanes

Fig. 2. Bit-depth recovery from a bitplane perspective. Existing DNN methods are trained to predict a residual image which when added to the input q-bit quantized LBD image produces the desired N-bit HBD output image. Interpreted as bitplanes, recovering the residual is equivalent to recovering the \((N-q)\) bitplanes lost during quantization. As opposed to existing methods that predict the full residual in a single shot, we propose to recover it one bitplane at a time using a multi-level-supervised training strategy. Specifically, we train \((N-q)\) separate networks in a supervised fashion where the input/target pairs for each network are obtained by applying the appropriate level of quantization to the ground truth HBD image.

Liu et al. \cite{11} argue that training using \((I_q, R)\) pairs as input and target yields better results. At test time, the estimate of the residual \(\hat{R}\) can simply be added to the input \(I_q\) to generate the desired HBD image \(\hat{O}\). See row one of Fig. 2.

Our proposed method is based on the residual model in equation \(2\). In row two of Fig. 2, we present its equivalent interpretation in terms of bitplanes. The \(N\)-bit HBD ground truth image \(O\) can be expressed as

\[
\hat{O} = \sum_{k=1}^{q} 2^{N-k} \times B_{N-k} + \sum_{k=1}^{N-q} 2^{N-(q+k)} \times B_{N-(q+k)},
\]

where the first term corresponds to the \(q\) most significant bits and the second term corresponds to the \((N-q)\) least significant bits, and \(B\) denotes a binary map of 0s and 1s. When \(O\) is quantized to \(q\) bits using equation \(1\), we get

\[
I_q = \sum_{k=1}^{q} 2^{N-k} \times B_{N-k} + \sum_{k=1}^{N-q} 2^{N-(q+k)} \times Z_{N-(q+k)},
\]

where all entries of \(Z\) are zero. The residual \(\hat{R}\) predicted by the network is thus

\[
\hat{R} = \sum_{k=1}^{N-q} 2^{N-(q+k)} \times \hat{B}_{N-(q+k)},
\]

where \(\hat{B}\) is an estimate of the bitplanes. We use grayscale images in Fig. 2 for ease of visualization of the bitplanes – the extension to color images is straightforward.

Next, we describe in detail our training strategy. Given a \(q\)-bit quantized image \(I_q\), the objective is to recover the \((N-q)\) bitplanes \(\hat{B}\) lost during quantization, and restore \(I_q\) to its original bit depth of \(N\). Towards this goal, we train \((N-q)\) networks independently as shown in Fig. 3 where each network is trained to predict one single bitplane. Observe that \(I_q\) retains \(q\) most significant bits from \(O\). Therefore, the first network labelled DNN \((q+1)\) is trained to predict the bitplane \(N-(q+1)\). The input and target pairs for this network are the \(q\)-bit quantized image \(I_q\) and the ground truth binary map \(B_{N-(q+1)}\), respectively, both of which can be obtained from the ground truth HBD image \(O\). We use the binary cross entropy (BCE) loss, which is typically applied to binary image segmentation problems, to train our network. The loss is computed between the ground truth binary map \(B_{N-(q+1)}\) and the network’s prediction \(\hat{B}_{N-(q+1)}\).

In a similar manner, to predict the next least significant bit, we train DNN \((q+2)\) using the input-target pair \((I_{q+1}, B_{N-(q+2)})\), where \(I_{q+1}\) is obtained by quantizing \(O\) to \((q+1)\) bits using equation \(1\). This process is repeated for all \((N-q)\) bits – the last least significant bitplane is trained using DNN \((N)\) with an \((N-1)\)-bit quantized image \(I_{N-1}\) as input, and the ground truth binary map \(B_0\) as target. All our networks are trained using the same BCE loss as shown in Fig. 3.

At test time, the trained networks are applied sequentially as shown in Fig. 3. The \(q\)-bit quantized test image \(I_q\) is first fed to DNN \((q+1)\). The network’s prediction \(\hat{B}_{N-(q+1)}\) is multiplied by the appropriate weighting factor \(2^{N-(q+1)}\) and this result is added to the input image \(I_q\) to obtain our estimate \(\hat{I}_{q+1}\). Note that at this stage, \(\hat{I}_{q+1}\), we have restored the \(q\)-bit quantized input \(I_q\) to a bit depth of \((q+1)\). Next, we input this estimate \(\hat{I}_{q+1}\) to DNN \((q+2)\) to obtain its prediction \(B_{N-(q+2)}\), which is multiplied by \(2^{N-(q+2)}\) and added to \(\hat{I}_{q+1}\) to obtain our \((q+2)\)-bit-depth estimate, \(\hat{I}_{q+2}\). Repeating this process till the last bit produces our estimate of the HBD image \(O\).

### 3.1 Network architecture

Our network architecture is shown in Fig. 4. The input is a 3-channel RGB image \(I_{q+k-1}\). The network predicts a 3-channel output \(\hat{B}_{N-(q+k)}\) having the same size as the input.
Proposed approach: Training

\[
\text{Proposed approach: Testing}
\]

Fig. 3. Overview of our proposed approach. We train \((N-q)\) separate DNNs to recover the \((N-q)\) bitplanes lost during quantization. Here, \(N\) is the bit depth of the desired HBD image, and \(q\) the bit depth of the input LBD image. The input-target pairs \((I_{q+k}, O_{N-(q+k)})\), \(k=1\) to \((N-q)\), are used for training each network. The target \(O_{N-(q+k)}\) is a binary map, and the problem reduces to one of binary image segmentation, which we optimize using a binary cross entropy loss. At test time, the \(q\)-bit quantized input LBD image \(I_{q}\) is passed through the trained networks sequentially, with each network increasing the bit depth by one until the image is restored to its desired bit depth of \(N\). In particular, each individual network predicts the next bitplane, which is then weighted by the corresponding bit position, and added back to the input image to increment its bit depth by one. This estimated image forms the input to the next network, and so forth.

Here, \(k\) is the index that runs over the bitplanes – namely, \(k=1\) to \((N-q)\). We use the same architecture for all \((N-q)\) bitplanes. As previously mentioned, each of these \((N-q)\) networks is trained independently. Our architecture consists of an initial convolution layer (Conv), a series of \(D\) residual blocks, followed by a batch normalization (BN) layer and a final convolution layer. The last layer is a sigmoid activation restricting all pixel values to lie in the range \([0,1]\). Our residual blocks are styled after [13] and contain two convolution layers, two batch normalization layers, and a rectified linear unit (ReLU). There is an additive skip connection between the input and output of each residual block. All convolution filters are of size \(3 \times 3\) and padded such that the input and output are of the same size. We use a fixed number of 64 filters for all convolution layers. The depth of our network is determined primarily by the number of residual units \(D\). As we shall demonstrate in Sec. [4] we can already outperform all competing approaches with \(D=4\). We also experiment with \(D=16\) and show that this gives us a further boost in performance.

We note that our primary focus is not the choice of network architecture, but rather to evaluate the effectiveness of our bitplane-wise training strategy for bit-depth recovery. Other recent works have used more complex architectures, such as dense feature concatenations, multiple streams, and encoder-decoder structures. While such configurations are indeed possible, our aim is to demonstrate that a standard ResNet-style [22] architecture can produce state-of-the-art results with bitplane-wise training. We would also like to add that while our architecture resembles a single stream of BDEN [13], their overall network structure is more complex with two streams and a local adaptive adjustment preprocessing step applied to one stream.

Our method requires training separate models for each bitplane. While this may be seen as cumbersome, competing methods, such as [13], [14], also train separate models depending on the bit depth of the quantized input image.
4 RESULTS

In this section, we first provide details regarding how we train our networks. Next, we evaluate the performance of our algorithm against competing approaches both quantitatively and qualitatively using five publicly available image datasets.

4.1 Training

We use 2000 color images, 1000 each from the Sintel dataset [24] and the MIT-Adobe FiveK dataset [25], for training. Sintel is a short animation film, and the dataset contains more than 20,000 16-bit lossless images at a resolution of 436 × 1024 pixels. Following [10], [11], we select 1000 images at random from this dataset. The MIT-Adobe FiveK dataset is a natural image dataset consisting of 5000 images in raw format captured using different SLR cameras covering a broad range of scenes, subjects, and lighting conditions. Each raw image has been retouched in Adobe Lightroom by five professionals, and these five variations, saved as 16-bit lossless TIFF files, are available as part of the dataset. We use the first 1000 images (i.e., filenames a0001 to a1000) enhanced by a random expert for training. To ensure that images across both datasets are roughly the same resolution, all images from the MIT-Adobe dataset are downsized by a factor of 4. While some papers [12], [13], [14], [15] use natural images for training, others [10], [11] argue that animated images represent harder examples, and are thus more appropriate. As compared to training on only one category of images, we found that a balanced mix of animated and natural images gives the best generalization across both animated and natural image datasets.

For training, we crop non-overlapping patches of size 48 × 48 pixels from all 2000 images. Data augmentation is performed by randomly flipping the patches, or rotating them by 90°. We use the Adam optimizer [29] with parameters β1 = 0.9 and β2 = 0.999, a learning rate of 10−3, and a batch size of 128. The learning rate is divided by 5 after 16 epochs. All our models are trained for 30 epochs using these same settings. Our method is implemented using TensorFlow [30]. Our code and corresponding pre-trained models are publicly available at https://github.com/abhijithpunnapurath/a-little-bit-more.

4.2 Evaluation

We evaluate the performance of our method on three natural image datasets – MIT-Adobe FiveK [25], TESTIMAGES [26], and Kodak [27] – and two animated image datasets – Sintel [24] and ESPL v2 [28]. For quantitative evaluation,

| 4-16 bit | ZP | MIG | BR | MRC | CRR | CA | ACDC | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|----------|----|-----|----|-----|-----|----|------|------|--------|--------|--------|--------|--------|
| PSNR     | 42.8602 | 44.3357 | 44.4725 | 46.8886 | 46.1370 | 47.0376 | 46.6522 | 47.6593 |
| SSIM     | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 |

| 6-16 bit | ZP | MIG | BR | MRC | CRR | CA | ACDC | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|----------|----|-----|----|-----|-----|----|------|------|--------|--------|--------|--------|--------|
| PSNR     | 42.8602 | 44.3357 | 44.4725 | 46.8886 | 46.1370 | 47.0376 | 46.6522 | 47.6593 |
| SSIM     | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 |

| 8-16 bit | ZP | MIG | BR | MRC | CRR | CA | ACDC | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|----------|----|-----|----|-----|-----|----|------|------|--------|--------|--------|--------|--------|
| PSNR     | 42.8602 | 44.3357 | 44.4725 | 46.8886 | 46.1370 | 47.0376 | 46.6522 | 47.6593 |
| SSIM     | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 | 0.9990 |
implemented in Matlab (BitNet) \[12\]. The codes of the eight classical algorithms are (BE-CALF) \[11\], and (3) learning-based bit-depth expansion against eight classical (i.e., non-deep learning) algorithms are chosen as metrics. We compare our proposed method PSNR (dB) and structural similarity index (SSIM) \[31\] are chosen as metrics. We compare our proposed method against eight classical (i.e., non-deep learning) algorithms – (1) zero padding (ZP), (2) multiplication by ideal gain (MIG), (3) bit replication (BR) \[4\], (4) minimum risk-based classification (MRC) \[5\], (5) contour region reconstruction (CRR) \[6\], (6) content-adaptive image bit-depth enhancement (CA) \[7\], (7) maximum a posteriori estimation of AC signal (ACDC) \[8\], and (8) intensity potential for image de-quantization (IPAD) \[9\]. We also compare against three recent DNN methods – (1) bit-depth enhancement with CNN (BE-CNN) \[10\], (2) concatenating all level features (BE-CALF) \[11\], and (3) learning-based bit-depth expansion (BitNet) \[12\]. The codes of the eight classical algorithms are implemented in Matlab by the authors of \[9\]. For the DNN methods \[10\], \[11\], pre-trained models have been released by the authors for only two quantization levels – 4 to 16, and 8 to 16 bit recovery. Training code is not available. Hence, we report results for 4-bit and 8-bit quantized input images using their pre-trained models. For other quantization levels, we copy the results directly from the original papers, available for comparison against BitNet \[12\], we report scores using the code released by the authors.

We first report results on the Sintel \[24\] dataset. Following \[10\], \[11\], we select 50 random images (different from the training set) from this dataset for evaluation. Table \[1\] shows the scores of our proposed algorithm as well as various competing methods. Two variants of our proposed method corresponding to two different network depths are tested – D4 and D16, corresponding to \(D=4\) and \(D=16\) residual units, respectively, as explained in Sec. \[3.1\]. The best results are reported in bold and red. The second, third, and fourth best-performing methods are shown in green, blue, and yellow, respectively. Both \[10\] and \[11\] are trained exclusively on animated images from Sintel while we use a mix of natural and animated images for training. Yet, as can be seen from the results, both our models outperform all competing methods, including BE-CALF \[11\] and BE-CNN \[10\], by a sound margin across all quantization levels.

BitNet \[12\] has used the last 1000 images (i.e., filenames a4001 to a5000) of the MIT-Adobe dataset \[25\] enhanced by expert E for testing. The scores on these images are reported in Table \[7\]. Note that BitNet is trained exclusively on the first 4000 images enhanced by expert E, while we use only 1000 images by a random expert. Once again, it can be observed from the results that both our models outperform

### Table 3

Results on TESTIMAGES 1200 dataset \[26\].

|       | ZP | MIG | BR | MRC | CA | ACDC | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|-------|----|-----|----|-----|----|------|------|--------|---------|--------|---------|---------|
| 4-16 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 4-12 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 4-8 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 6-16 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 6-12 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 8-16 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |

### Table 4

Results on Kodak dataset \[27\]. NR denotes that a score was ‘not reported’ in the original paper.

|       | ZP | MIG | BR | MRC | CA | ACDC | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|-------|----|-----|----|-----|----|------|------|--------|---------|--------|---------|---------|
| 3-8 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 4-8 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |

### Table 5

Results on ESPL v2 dataset \[28\].

|       | ZP | MIG | BR | MRC | CA | ACDC | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|-------|----|-----|----|-----|----|------|------|--------|---------|--------|---------|---------|
| 3-8 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| 4-8 bit | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |

1. https://sites.google.com/site/jingliu198810/publication
The remaining three datasets, TESTIMAGES [24], Kodak [27], and ESPL v2 [28], are not part of our training set. We evaluate our method’s generalizability by testing on these images. The TESTIMAGES dataset [24] contains 40 1200 × 1200 16-bit natural images. The scores are reported in Table 3. The Kodak dataset [27] is a natural image dataset consisting of 24 768 × 512 images with a bit depth of 8. The results on this dataset are reported in Table 4. ESPL v2 [28] contains 25 1920 × 1080 animated images also with a bit depth of 8. Following [12], we used only the pristine images without any distortion, and the scores are presented in Table 5. It can be seen from the results in Tables 3, 4, and 5 that our method outperforms all competing algorithms on both animated and natural images across a diversity of bit depths and image resolutions, thereby validating the effectiveness of our bitplane-wise training strategy.

Qualitative results are shown in Fig. 5. Following prior literature [10], [12], [13], we use only images with a maximum bit depth of 8 for qualitative evaluation such that they are suitable for display on standard 8-bit monitors. In the first example from the Kodak dataset [27], it can be observed that the vertical lines on the sails of the boat are best restored, and points to the effectiveness of bitplane-wise training strategy, rather than the loss function. Our chosen BCE loss is more suitable since our target is a binary image, and the VGG loss, which is intended to be applied to natural images, performs poorly. At the same time, the MSE loss offers competitive performance. This validates that the performance gain is due to our bitplane-wise training strategy, rather than the loss function. Our chosen BCE loss is more suitable since our target is a binary bitplane, and it offers slightly higher accuracy than MSE.

Accumulating bitplanes: Table 7 shows that there is a performance gain as each lost bitplane is incrementally restored, and points to the effectiveness of bitplane-wise training. Since our method restores the image bitplane-by-bitplane, prediction errors can propagate. However, we found this effect to be minimal as also evident from our state-of-the-art results.

Running time: On an Nvidia Tesla V100 GPU with 32 GB of RAM, our D4 model takes approximately 0.08 seconds per bitplane to process a 768 × 512 resolution image, while our D4 variant, its PSNR is roughly 1 dB lower than D4.

Ablations on target and loss function: To validate that our method’s improvement accrues from our proposed bitplane-wise training strategy, we conducted two ablation studies. In the first ablation, we train \((N-q)\) networks to progressively improve image quality by predicting the next-bit quantized image \(I_{(q+k)}\), instead of the bitplane \(B_{N-(q+k)}\). The loss is changed to MSE since the target \(I_{(q+k)}\) is an image, and not a bitplane. All other training parameters are left unchanged. In Table 6 results are reported on four datasets for 4 to 8 bit recovery using the D4 model. Our proposed bitplane prediction, which is agnostic to the relative magnitude of the bit position, is more accurate than predicting the image \(I_{(q+k)}\). We also performed a second ablation by changing only the loss function. In particular, we tested a perceptual VGG loss [32] and an MSE loss to predict the bitplane \(B_{N-(q+k)}\). Bitplanes have a very different distribution from natural images, and the VGG loss, which is intended to be applied to natural images, performs poorly. At the same time, the MSE loss offers competitive performance. This validates that the performance gain is due to our bitplane-wise training strategy, rather than the loss function. Our chosen BCE loss is more suitable since our target is a binary bitplane, and it offers slightly higher accuracy than MSE.

Accuracy versus model size: The plot of Fig. 6 shows the PSNR (dB) versus number of model parameters on the Kodak dataset for the 4-8 bit scenario (see Table 4). Our method is compared against BE-CALF [11] and BitNet [12], which are our two closest competitors. Our D4 model has approximately 1.2 million parameters in total for 4 bitplanes, while D16 requires 4.77 million. In comparison, BE-CALF, at 5.18 million parameters, is roughly four times bigger than D4 while yielding a PSNR about half a dB lower than our models. While BitNet has roughly the same model size as our D4 variant, its PSNR is roughly 1 dB lower than D4.

Additional results are provided in the supplementary.
our D16 model takes 0.28 seconds on average. The running times of competing methods are reported in the supplementary material.

5 Conclusion

We have proposed a new DNN training strategy for bit-depth recovery where the residual image corresponding to the bits lost due to quantization is recovered bitplane by bitplane. Bitplane-wise learning makes our method independent of the relative magnitude of the bit position, overcoming the limitation of existing single-shot approaches. Experiments on five benchmark datasets demonstrate that our method outperforms the state of the art by up to 2.3 dB PSNR depending on the quantization level. As future work, we plan to explore the sharing of features across bitplanes instead of training networks completely independent of each other. We also plan to examine bitplane-wise training for other residual estimation problems.

References

[1] H. C. Karaimer and M. S. Brown, “A software platform for manipulating the camera imaging pipeline,” in ECCV, 2016.

[2] [Online]. Available: https://www.samsung.com/global/galaxy/galaxy-s10/specs/

[3] [Online]. Available: https://support.apple.com/kb/sp770?locale=en_US

[4] R. A. Ulrich and S. Cheung, “Pixel bit-depth increase by bit replication,” in Color Imaging: Device-Independent Color, Color HardCOPY, and GraphicArts, SPIE, 1998.

[5] G. Mittal, V. Jakheltya, S. Iaiswal, O. Au, A. Tiwari, and W. Dei, “Bit-depth expansion using minimum risk based classification,” in Visual Communications and Image Processing, 2012.

[6] C. Cheng, O. Au, C. H. Liu, and K. Yip, “Bit-depth expansion by contour region reconstruction,” in IEEE International Symposium on Circuits and Systems, 2009.

[7] P. Wan, O. Au, K. Tang, Y. Guo, and L. Fang, “From 2D extrapolation to 1D interpolation: Content adaptive image bit-depth expansion,” in IEEE International Conference on Multimedia and Expo, 2012.

[8] P. Wan, G. Cheung, D. Florencio, C. Zhang, and O. Au, “Image bit-depth enhancement via maximum a posteriori estimation of AC signal,” IEEE Transactions on Image Processing, vol. 25, no. 6, pp. 2896–2909, 2016.

[9] J. Liu, G. Zhai, X. Yang, and C.-W. Chen, “IPAD: Intensity potential for adaptive de-quantization,” IEEE Transactions on Image Processing, vol. 27, no. 10, pp. 4860–4872, 2018.

[10] Y. Su, W. Sun, J. Liu, G. Zhai, and P. Jing, “Photo-realistic image bit-depth enhancement via residual transposed convolutional neural network,” Neurocomputing, 2019.

[11] J. Liu, W. Sun, Y. Su, P. Jing, and X. Yang, “BE-CALF: Bit-depth enhancement by concatenating all level features of DNN,” IEEE Transactions on Image Processing, vol. 28, no. 10, pp. 4926–4940, 2019.

[12] J. Byun, K. Shim, and C. Kim, “BitNet: Learning-based bit-depth expansion,” in ACCV, 2019.

[13] Y. Zhao, R. Wang, W. Jia, W. Zuo, X. Liu, and W. Gao, “Deep reconstruction of least significant bits for bit-depth expansion,” IEEE Transactions on Image Processing, vol. 28, no. 6, pp. 2847–2859, 2019.

[14] X. Hou and G. Qiu, “Image compositing and inverse halftoning using deep convolutional neural networks,” arXiv, 2017.

[15] Y. Xiao, C. Pan, Y. Zheng, X. Zhu, Z. Qin, and J. Yuan, “Gradient-guided DCNN for inverse halftoning and image expanding,” in ACCV, 2019.

[16] J. Liu, W. Sun, and Y. Liu, “Bit-depth enhancement via convolutional neural network,” in Digital TV and Wireless Multimedia Communication, 2018.

[17] G. Sharma, Digital Color Imaging Handbook: CRC Press, Inc., 2002.

[18] “Graphics interchange format, version 89a,” [Online]. Available: https://www.w3.org/Graphics/GIF/spec-gif89a.txt.

[19] G. Eilertsen, J. Kronander, G. Denes, R. K. Mantiuk, and J. Unger, “HDR image reconstruction from a single exposure using deep CNNs,” ACM Transactions on Graphics, vol. 36, no. 6, pp. 1–15, 2017.

[20] S. Y. Kim, J. Oh, and M. Kim, “Deep SR-ITM: Joint learning of super-resolution and inverse tone-mapping for 4K UHD HDR applications,” in ICCV, 2019.

[21] M. Ikebe and A. Mizuno, “Bit-depth expansion for noisy contour reduction in natural images,” in ICASSP, 2016.

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016.

[23] Y. Li, L. Sharan, and E. H. Adelson, “Compressing and compositing high dynamic range images with subband architectures,” ACM Transactions on Graphics, vol. 24, no. 3, pp. 836–844, 2005.

[24] Xiph Foundation, “xiph.org,” [Online]. Available: http://www.xiph.org/

[25] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, “Learning photographic global tonal adjustment with a database of input / output image pairs,” in CVPR, 2011.

[26] N. Asuni and A. Gachetti, “TESTIMAGES: A large-scale archive for testing visual devices and basic image processing algorithms,” in Smart Tools and Apps for Graphics Conference, 2014.

[27] Eastman Kodak, “Kodak lossless true color image suite,” 1999, [Online]. Available: http://r0k.us/graphics/kodak/

[28] D. Kundu and B. Evans, “Full-reference visual quality assessment for synthetic images: A subjective study,” in ICIP, 2015.

[29] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” ICLR, 2014.

[30] M. et. al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, [Online]. Available: https://www.tensorflow.org/

[31] C. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, 2004.

[32] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time image denoising,” in ICLR, 2016.

[33] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015.

[34] T. L. et. al., “Microsoft COCO: Common objects in context,” arXiv, 2014, [Online]. Available: http://arxiv.org/abs/1405.0312

[35] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in ICCV, 2001.

[36] E. Agustsson and R. Timofte, “NTIRE 2017 challenge on single image super-resolution: Dataset and study,” CVPR Workshop, 2017.
This supplementary material provides additional results and details that could not be included in the main paper due to space constraints. Ablation studies on our network architecture are presented in Sec. S1. A single-shot baseline and a comparison against the standard U-Net [33] architecture are reported in Sec. S2. We also perform experiments to validate our training protocol in Sec. S3 and choice of training data in Sec. S4. In Secs. S5 and S6, we provide further quantitative and qualitative comparisons and results. The running times of various methods are reported in Sec. S7. An example highlighting the importance of bit-depth recovery for photo editing is furnished in Sec. S8.

**S1 ABLATIONS ON NETWORK ARCHITECTURE**

We performed ablation studies by varying the number of residual units $D$ in our network architecture (see Fig. 4 of our main paper). Specifically, we tested the following values $D = 2, 4, 8, 12, 16, 20$. The average PSNR (dB) and SSIM versus $D$ on the TESTIMAGES 1200 dataset [26] for 4 to 8 bit recovery are shown in the plots of Fig. S1. We have reported results, both in the main paper and this supplementary, for our D4 and D16 variants. We have showed that we already outperform existing methods with $D = 4$. Beyond $D = 16$, we did not observe an improvement in performance.

We also performed an ablation study where we varied the network architecture for each bitplane. In particular, we tested the 4 to 8 bit recovery scenario with each bitplane being recovered by either our D4 model or our D16 model. The plot of Fig. S2 shows SSIM versus the total number of network parameters (in millions) on the TESTIMAGES 1200 dataset [26]. In the first column with 1.20 million parameters, the D4 model was used to recover all four bitplanes. The number of D16 models used for recovery increases by one in each subsequent column, with the last column using D16 models for all four bitplanes. It can be observed from the plot that the most gains are achieved by using the larger D16 model for the more significant bit positions e.g., $2^3$ and $2^2$ in the present case.

**S2 BASELINE COMPARISONS**

We perform two baseline comparisons. The first is a single-shot approach based on our proposed network architecture, while the second is our proposed multi-level training strategy but with our model replaced by a standard U-Net [33].

**Single-shot baseline comparison:** Our proposed method employs $(N-q)$ networks with $D$ residual blocks each. For comparison, we trained a single network with the same total capacity by stacking together $(N-q) \times D$ residual blocks. The network was trained (under identical settings) to directly predict the residual using an MSE loss, with the final sigmoid layer removed. On the TESTIMAGES 1200 dataset [26], for example, for $D=4$, $N=8$, and $q=4$, the single-shot model obtained PSNR/SSIM values of 37.4822/0.9665, while our proposed method produced values of 39.6822/0.9691. This demonstrates the advantages of our bitplane-wise training strategy over the single-shot approach.

**Comparison against U-Net [33]:** Our proposed bitplane-wise training scheme is a general strategy that can be implemented using any network architecture of choice. In Table S1, we compare against the standard U-Net [33] model for 4 to 8 bit recovery. We varied the number of filters in the initial level of the U-Net (the number of filters in the subsequent layers are multiples thereof) such that the total number of parameters in the model is approximately equal to our proposed network. In particular, our D4 model has 301,248 parameters per bitplane while our D16 model has 1,193,664 parameters. For a fair comparison, we trained two U-Net models with 7 and 13 filters in the initial level for a total of 372,088 parameters (D4 equivalent) and 1,281,790 parameters (D16 equivalent) per bitplane, respectively. In can be observed from Table S1 that the U-Net models also offer competitive performance, but our proposed architecture is more accurate.
S3 TRAINING PROTOCOL

In our proposed framework, the \((N-q)\) bitplane networks are trained independently. This means that from the second network onwards, the distribution of the input \(I_{(q+k-1)}\) at test time is not identical to the distribution of the input \(I_{(q+k-1)}\) during training. To verify whether this difference in distribution has a significant impact on accuracy, we jointly trained all \((N-q)\) networks in an end-to-end fashion, and compared performance against our proposed independently trained networks. It is important to note that our proposed approach has the advantage that the network corresponding to each bitplane needs to be trained only once, whereas for joint end-to-end training, these networks have to be re-trained for each input quantization. For example, the network that predicts the 5th bitplane trained for 4 to 8 bit recovery cannot be re-used for 3 to 8 bit recovery under the joint training pipeline.

The results of joint training are presented in Table S2. In all cases, the D4 model was used. We first tested our method against a model trained jointly for 4 to 8 bit recovery. We also trained another joint model for 4 to 16 bit recovery, and tested it on the 4 to 16 and 4 to 12 bit recovery tasks. It can be observed that joint training is more accurate in some scenarios, while our proposed method works better in other cases. However, the difference in performance is very small. Overall, we did not find that joint training offers an improvement that outweighs the benefits of independent training, wherein each bitplane network needs to be trained only once.

S4 CHOICE OF TRAINING DATA

Our proposed models were trained on 2000 images, 1000 natural images from the MIT-Adobe FiveK dataset \([25]\) and 1000 animated images from the Sintel dataset \([24]\). We performed an ablation experiment where we trained two separate models – (1) on 2000 natural images from the MIT-Adobe FiveK dataset, and (2) on 2000 animated images from the Sintel dataset. The results on six different test sets are presented in Table S3. Scores reported are using our D4 model for 4 to 8 bit recovery. As expected, training on only animated images gives the best performance on the Sintel animation dataset, while training only on natural images produces the best accuracy on the MIT-Adobe FiveK natural image dataset. However, there is a 0.81 dB drop in PSNR when the model trained on natural images is tested on animated images, and similarly, a 0.85 dB drop when the model trained on animated images is tested on natural images. In comparison, our mixed model performs well on both natural and animated images with only a nominal drop of 0.14 dB and 0.10 dB, respectively, compared to the best model. Similar observations can be made on the other test sets. TESTIMAGES 1200 \([26]\), Kodak \([27]\) and BSD100 \([35]\) are natural image datasets, and the model trained on only animated images incurs a significant drop in performance, while our mixed model performs almost on par or even better than the model trained purely on natural images. Likewise, on the ESPL v2 \([28]\) animated dataset, the...
S5 ADDITIONAL QUANTITATIVE RESULTS

We compare against the deep learning methods of Hou et al. [14] and GG-DCNN [15], who have trained and tested their models on the Microsoft COCO [34] dataset. Their pre-trained models/codes are not available. Following GG-DCNN [15], we use 2000 random images from the testing fold of this dataset. The authors of [15] have reimplemented the method of [14], and in Table S2, the results of both [15] and [14] are copied from [15]. Note that [15] report results on downsampled images of size $256 \times 256$ and $512 \times 512$ (in line with their training settings). We reproduce the best scores of [15] and [14] in Table S4. Also, since resizing to square images in this manner alters the aspect ratio, the results of our method are reported on the original-sized images. It can be observed from the results in Table S4 that our PSNR values are significantly higher than [14] and [15].

The TESTIMAGES dataset [26] contains 40 16-bit natural images. While [10], [11] have reported results on the $1200 \times 1200$ resolution variant of this dataset, BitNet [12] has used $800 \times 800$ resolution images in their paper. We reported comparisons in Table 3 of our main paper on $1200 \times 1200$ resolution images. In Table S5 of this supplementary, we present comparisons on $800 \times 800$ resolution images. (Following [12], we used image files with the shifting indicator ‘B01C00’.) From the results, it can be observed that both our models, D4 and D16, outperform BitNet on both metrics.

In Table S6, we report results for 8 to 10 bit, and 8 to 12 bit recovery on the Sintel dataset [24] and TESTIMAGES dataset [26]. The vast majority of high-bit-depth (HBD) monitors available today are 10 bit, and therefore, 8 to 10 bit reconstruction represents the most common scenario of bit-depth recovery applied to high-bit-depth displays. Modern camera sensors have a dynamic range of 10–12 bits. Thus, 8 to 12 bit recovery is the typical range for photo editing applications. It can be seen that our proposed method obtains higher accuracy than competing approaches. Results on MIT-Adobe FiveK dataset [25] for the same quantization levels are provided in Table S7. Comparisons against CRR [6], CA [7], and ACDC [8] have been omitted since there are 1000 images in the test set, and the running times (see Table S4) of these methods are very high. Our method again outperforms competitors by a sound margin.

BDEN [13] has reported scores on the Kodak dataset [27] and the BSD100 dataset for 6 to 8 bit recovery. The BSD100 dataset consists of 100 images selected from the Berkeley Segmentation dataset [35]. A demo version of the code for the deep learning method in BDEN has been publicly released by the authors [13]. However, it works only on grayscale images, and the model has been trained using only the first 100 images from the DIV2K [36] dataset, while their actual model in the paper is trained on color images using all 800 images from this dataset. In consideration of these limitations, for fair comparison, we directly copy results reported in their paper instead of running their demo code. The results are presented in Table S8. We would like to note that our bitplane-wise recovery strategy offers the most advantage over single-shot methods for larger expansions of the bit depth e.g., 8–12 bit, 6–12 bit etc. In the present case, the expansion is only 2 bits (from 6 to 8), and hence, our PSNR values are only slightly higher than BDEN.

S6 ADDITIONAL QUALITATIVE RESULTS

Qualitative comparisons for 3 to 8 bit recovery on MIT-Adobe FiveK [25], Sintel [24], TESTIMAGES 1200 [26], Kodak [27], and ESPI v2 [28] datasets are provided in Figs. S3, S4, S5, S6, and S7 respectively. Zoomed-in patches are presented for better visualization. Representative structures including smooth color transitions, edges, and fine textures are shown.

In the first example of Fig. S3 it can be clearly observed that the horizontal lines are best recovered by our method. In the second example, competing approaches cannot reproduce the correct color tones, and have artifacts inside the circle, while our result is closest to the ground truth. In Fig. S4 from the Sintel dataset, the thin petal structures and the fine details in the hair (denoted by dotted rectangles on the zoomed-in ground truth patches) are best reconstructed.
### TABLE S5
Results on TESTIMAGES 800 dataset [26].

|          | ZP    | MIG | BR   | MRC  | CRR  | CA   | ACDC  | IPAD  | BitNet | Ours D4 | Ours D16 |
|----------|-------|-----|------|------|------|------|-------|-------|--------|---------|----------|
| 3-16 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |
| 4-16 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |
| 5-16 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |
| 6-16 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |

### TABLE S6
8–10 bit and 8–12 bit recovery.

|          | ZP    | MIG | BR   | MRC  | CRR  | CA   | ACDC  | IPAD  | BE-CNN | BE-CALF | BitNet   | Ours D4 | Ours D16 |
|----------|-------|-----|------|------|------|------|-------|-------|--------|---------|----------|---------|----------|
| 8-10 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |         |          |
| 8-12 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |         |          |

### TABLE S7
Results on MIT-Adobe FiveK dataset [25].

|          | ZP    | MIG | BR   | MRC  | IPAD | BE-CNN | BE-CALF | BitNet | Ours D4 | Ours D16 |
|----------|-------|-----|------|------|------|---------|---------|--------|---------|----------|
| 8-10 bit | PSNR  |     |      |      |      |         |         |        |         |          |
|          | SSIM  |     |      |      |      |         |         |        |         |          |
| 8-12 bit | PSNR  |     |      |      |      |         |         |        |         |          |
|          | SSIM  |     |      |      |      |         |         |        |         |          |

### TABLE S8
6–8 bit recovery.

|          | ZP    | MIG | BR   | MRC  | CRR  | CA   | IPAD  | BDEN | Ours D16 |
|----------|-------|-----|------|------|------|------|-------|------|----------|
| BSD100   | PSNR  |     |      |      |      |      |       |      |          |
|          | SSIM  |     |      |      |      |      |       |      |          |
| Kodak    | PSNR  |     |      |      |      |      |       |      |          |
|          | SSIM  |     |      |      |      |      |       |      |          |

### TABLE S9
Average running times of different algorithms on the Kodak dataset [27] for 4 to 8 bit recovery.

| Method   | Time (s) |
|----------|----------|
| 4-8 bit  | 0.006    | 0.008    | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  | 0.001  |

| Method   | ZP    | MIG | BR   | MRC  | CRR  | CA   | ACDC  | IPAD  | BE-CNN | BE-CALF | BitNet   | Ours D4 | Ours D16 |
|----------|-------|-----|------|------|------|------|-------|-------|--------|---------|----------|---------|----------|
| 4-8 bit  | PSNR  |     |      |      |      |      |       |       |        |         |          |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |         |          |
| 8-16 bit | PSNR  |     |      |      |      |      |       |       |        |         |          |         |          |
|          | SSIM  |     |      |      |      |      |       |       |        |         |          |         |          |
by our method as compared to competitors. Fig. S5 shows examples of smooth color transitions, and it can be seen that other algorithms introduce false edges (first image) or fail to detect the boundary of the eye (second image). In Fig. S6, comparison techniques fail to recover the fine lines on the top left in the first example, while artifacts can clearly be observed in the window blinds in the second example. Our approach produces an accurate reconstruction in both cases. In the first animated example from the ESPL v2 dataset in Fig. S7, it can be seen that most competing methods, including BitNet [12], produce artifacts along the edge of the face, particularly in the bottom left. Our result more closely matches the ground truth. The second image is a challenging example where we yet again produce compelling results, whereas comparison methods fail to recover the image structure.

S7 Running time

In Table S9, we report the average running times of eight classical methods and three DNN methods on the Kodak dataset [27], which contains images of size $768 \times 512$ pixels. The 4 to 8 bit recovery scenario is tested. As already mentioned in the main paper, we used the codes made publicly available by the authors of [9] to compare against the classical methods. The implementations are CPU-based. In the last two columns of Table S9, the average running times of our D4 and D16 models to process four bitplanes are reported. Our method is implemented on GPU. Although ZP, MIG, and BR [4] are extremely fast, their performance is poor. CRR [6], CA [7], and ACDC [8] have high computational complexity, and their outputs are inferior to ours both qualitatively and quantitatively. Although the difference in the testing environment (CPU versus GPU) biases a direct comparison of running times in our favour, it is still noteworthy that our proposed method is much faster than IPAD, which is the best performing classical method. More importantly, our method produces more accurate results than IPAD. The three DNN methods BE-CNN [10], BE-CALF [11] and BitNet [12], as well as our algorithm were tested on an Nvidia Tesla V100 GPU with 32 GB of RAM. While BE-CNN is faster than our method, its results are poor. In terms of accuracy, our two closest competitors are BE-CALF and BitNet. Our D16 model has nearly the same running time as BE-CALF, while D4 is faster. Both our models are significantly faster than BitNet.

S8 Photo editing application

The ability to edit an image post-capture can be adversely affected by quantization to 8 bits. Fig. S8(A) shows a common example of a captured 8-bit sRGB image having low contrast. Applying a simple histogram equalization operation on this 8-bit image to enhance its contrast produces the undesired “stair-step” effect shown in Fig. S8(B). This artifact is a direct consequence of the quantization process whereby tonal information was lost. Fig. S8(B) shows the results of histogram equalization when applied to the original unquantized 12-bit image (which is usually unavailable) and our recovered 12-bit result. Due to the additional tonal values in the 12-bit histograms, the banding artifacts are not present in these outputs. This example highlights the importance of bit-depth recovery for photo editing.
Fig. S3. Qualitative comparisons on the MIT-Adobe FiveK dataset [25] for 3 to 8 bit recovery.
Fig. S4. Qualitative comparisons on the Sintel dataset \cite{24} for 3 to 8 bit recovery.
Fig. S5. Qualitative comparisons on the TESTIMAGES 1200 dataset [26] for 3 to 8 bit recovery.
Fig. S6. Qualitative comparisons on the Kodak dataset for 3 to 8 bit recovery.
Fig. S7. Qualitative comparisons on the ESPL v2 dataset [28] for 3 to 8 bit recovery.
Fig. S8. (A) Shows a ground truth 12-bit HBD image, its 8-bit quantized LBD version, and our recovered 12-bit HBD result. The image has low contrast as shown by its image histogram. The green box denotes a zoomed-in region of the image. (B) Shows the same three images with histogram equalization applied. Due to its low bit depth, the 8-bit image has gaps in its histogram and exhibits visual banding. Our recovered 12-bit HBD image has a similar visual appearance and histogram profile as the ground truth 12-bit HBD image.