Learning Raw Image Reconstruction-Aware Deep Image Compressors

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

Abstract—Deep learning-based image compressors are actively being explored in an effort to supersede conventional image compression algorithms, such as JPEG. Conventional and deep learning-based compression algorithms focus on minimizing image fidelity errors in the nonlinear standard RGB (sRGB) color space. However, for many computer vision tasks, the sensor's linear raw-RGB image is desirable. Recent work has shown that the original raw-RGB image can be reconstructed using only small amounts of metadata embedded inside the JPEG image [1]. However, [1] relied on the conventional JPEG encoding that is unaware of the raw-RGB reconstruction task. In this paper, we examine the ability of deep image compressors to be "aware" of the additional objective of raw reconstruction. Towards this goal, we describe a general framework that enables deep networks targeting image compression to jointly consider *both* image fidelity errors and raw reconstruction errors. We describe this approach in two scenarios: (1) the network is trained from scratch using our proposed joint loss, and (2) a network originally trained only for sRGB fidelity loss is later fine-tuned to incorporate our raw reconstruction loss. When compared to sRGB fidelity-only compression, our combined loss leads to appreciable improvements in PSNR of the raw reconstruction with only minor impact on sRGB fidelity as measured by MS-SSIM.

Index Terms—image compression, radiometric calibration, raw image reconstruction, deep learning-based image compression

1 Introduction and Motivation

Camera images are compressed and saved in the highly processed standard RGB (sRGB) color space. The camera sensor itself, captures images in an unprocessed raw-RGB format that is linear with respect to sensor irradiance. Raw-RGB images are converted onboard the camera to sRGB through a number of steps, many nonlinear in nature, in order to improve the perceptual and aesthetic quality of the image. Many computer vision tasks (e.g., deblurring, photometric stereo, color constancy) work best in the linear raw-RGB format. While modern cameras allow images to be saved in unprocessed linear-raw format, most casual photographers do not shoot in raw because of prohibitive file sizes and storage requirements. In the absence of the raw file, radiometric calibration methods are usually required to convert sRGB values back to either linear values proportional to scene radiance or, in the case of more recent raw reconstruction techniques, the original raw values (e.g., [2], [3], [4], [5], [6], [7], [8]). Recent work by Nguyen and Brown [1] showed how to embed small amounts of metadata (e.g., only 64 KBs), computed from an sRGB-raw pair, inside the compressed JPEG sRGB image to allow the raw image to be reconstructed. Although their technique reduces the storage demands for obtaining raw information, in this scenario, JPEG encoding is assumed, and applied independently of the raw reconstruction objective. There is no opportunity to make the compressor aware of the additional goal of raw reconstruction.

While conventional schemes, such as JPEG compression [9], have long been the dominant choice for lossy image compression, there has been a renewed interest in developing the next generation of compression methods based on deep neural networks [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. This recent focus on developing a new class of image compressors based on neural networks offers the opportunity to "learn" image compression that targets not only perceptual fidelity but also the image's utility for computer vision algorithms. Towards this goal, we advocate a raw reconstruction loss that can be integrated into existing deep learning frameworks for image compression such that the compressor is also aware of the target of reconstructing the raw image.

To this end, we show that the mapping function estimated for a given sRGB-raw pair by the method of Nguyen and Brown [1] can be accurately represented by a $16\times16\times16$ 3D lookup table (LUT). The implications of this finding are three-fold. First, the mapping function from sRGB to raw, though difficult to analytically define and differentiate, is piecewise smooth since the raw reconstruction quality is well preserved even when reducing from a full-sized $256\times256\times256$ LUT (as required by an sRGB to raw color space mapping) to a $16\times16\times16$ LUT. As a result, the derivatives of the local interpolating functions encoded within the LUTs can provide meaningful gradients for backpropagation. Secondly, a reduced $16\times16\times16$ LUT imposes minimal burdens on storage and memory during training. Furthermore, these LUTs can be computed offline and stored before the training step. Lastly, lookup operations are fast, and, in our experiments, the inclusion of our raw reconstruction loss adds only an additional 25% overhead to the training time of the compression network.

We select the state-of-the-art network architecture of Toderici et al. [19] to demonstrate how our raw reconstruction loss can be incorporated into a deep image compression neural network. An overview of our proposed scheme is shown in Fig. 1. During training, we assume that the raw image corresponding to the sRGB input is available. We

• *A. Punnappurath and M. S. Brown are with the Department of Electrical Engineering and Computer Science, York University, Toronto, Canada. E-mail: pabhijith@eecs.yorku.ca, mbrown@eecs.yorku.ca*

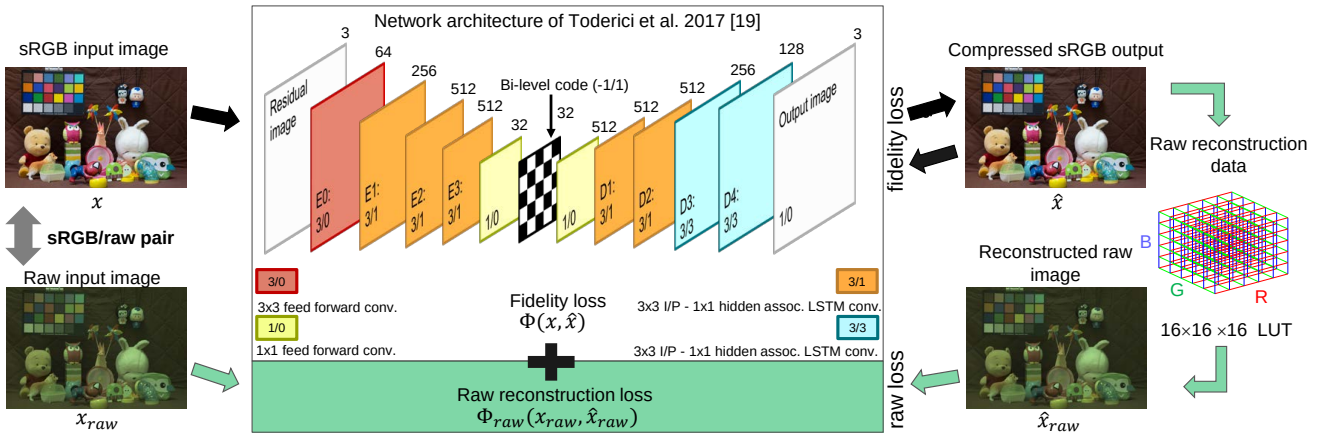


Fig. 1. An overview of our proposed scheme. The full-resolution image compression neural network of Toderici et al. [19], which is our architecture of choice to demonstrate the utility of our raw reconstruction loss, computes the sRGB fidelity loss between the input sRGB image and their compressed sRGB output. We first reconstruct the raw image from the sRGB output of their network via a lookup table, then compute our raw reconstruction loss between this estimated raw image and the input raw image, and append this raw loss to the existing fidelity loss. The path in green represents our contribution. (The illustration of the network architecture of [19] is modelled after the depiction of their network in [14].)

provide the uncompressed sRGB image as input, and apply the lookup operation on the estimated sRGB output of the network to recover its raw reconstruction. Our raw reconstruction loss calculated between the input raw image and our estimated raw image is added to the existing sRGB fidelity loss, and the combined error is backpropagated through the network. Note that raw images are required as input only during training.

Contribution This paper describes how to model raw image reconstruction into a neural network-based loss function such that any existing deep image compressor can be made aware of the additional objective of raw reconstruction. As part of this effort, we describe how to encode the metadata for reconstructing the raw image into a lookup table that can be differentiated locally to facilitate back propagation. We then show that this raw reconstruction LUT can be incorporated into a joint loss function that considers both image fidelity and the raw reconstruction. No changes to the underlying network architecture are needed since our proposed framework only needs to modify the loss function. We demonstrate the effectiveness of our approach on networks trained from scratch using the joint loss function, and networks trained only for image compression with a fidelity loss and then fine-tuned on the joint loss. Our results show that when compared to compression using only sRGB fidelity loss, we can obtain significant improvement in AUC (area under the rate-distortion curve) for the raw PSNR (peak signal-to-noise ratio) while incurring only modest losses in perceptual fidelity as measured by the AUC for MS-SSIM (multi-scale structural similarity) [20] metric.

We would like to highlight that achieving better compression than state-of-the-art image compression techniques is *not* the objective of our work. Due to the very nature by which our raw loss is appended to the existing loss (see Fig. 1), the compression rate achieved by our proposed method is limited by the compression rate of the chosen base architecture, which in the present case is [19]. The work in [19] is selected as a proof-of-concept of our proposed scheme because [19] was one of the first works to demonstrate high compression rates over variable bit rates on full-resolution images. We are aware that the architecture in [19] does not beat JPEG2000 [21], WebP [22], or BPG [23], and that other deep learning methods have recently appeared. We are also aware that recent work by [24] and [16] is able to outdo BPG, the top-performing conventional image compression algorithm. However, [24] and [16] have not released their code online at the time of submitting this work.

Our objective is to enable learning-based compressors to incorporate the goal of raw image reconstruction into their training procedure, and thereby endow them with the ability to perform better raw reconstruction. Current deep architectures perform compression in the perceptual sRGB space and produce a compressed sRGB image as output from an uncompressed sRGB input. To keep in line with this existing framework and provide a "plug and play" functionality, we use our LUT-based approach to map from the sRGB output of the network back to raw instead of compressing directly in the raw space or attempting direct raw recovery from sRGB by having the network try to learn this transformation. As such, our proposed loss function is generic and can be plugged into the current best-performing deep compressor, or future algorithms too for that matter.

2 Related Work

In this section, we review two areas of prior work: (1) raw reconstruction, and (2) image compression using deep neural networks.

Raw reconstruction Raw reconstruction is highly related to early work on radiometric calibration. Radiometric calibration methods (e.g., [3], [4], [7]) were designed to undo the non-linear processing applied onboard cameras to produce the final sRGB output. Radiometric calibration was developed before cameras allowed consumers access to the unprocessed raw-RGB data. As a result, radiometric calibration strived to linearize the sRGB data. As cameras started to allow capture in raw RGB, the goal shifted from linearizing the sRGB data to fully recovering the original

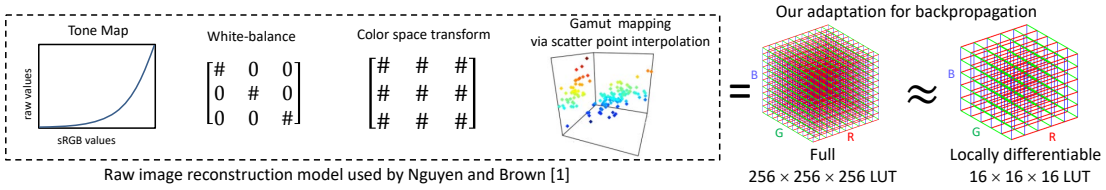


Fig. 2. An overview of the various stages in the framework of Nguyen and Brown [1]. Their method is equal to a full $256\times256\times256$ lookup operation, which we approximate by a locally differentiable $16\times16\times16$ lookup table for the purpose of backpropagation.

raw-RGB values (usually up to a quantization factor). Most notable in these methods was work by Chakrabarti et al. [2], [25], and Kim et al. [5] that introduced more complex onboard camera processing models in order to reconstruct the raw RGB. Similar work by Yuan and Sun [26] proposed a raw-RGB reconstruction method based on up-sampling using a low-resolution raw image stored along with the sRGB image to recover the full-resolution raw image. These aforementioned raw reconstruction methods require either careful camera calibration [2], [5], [25] or a small raw file to be stored along with the sRGB image [26]. Recent work by Nguyen and Brown [1] showed how the metadata to allow raw reconstruction could be computed directly (without the need for camera calibration) from an sRGB-raw image pair and could be embedded within a JPEG image using only 64KB of data. This method provided a practical solution to allow raw reconstruction from a self-contained JPEG image.

Deep learning-based image compression Image compression using deep neural networks has been attracting considerable attention recently from both the image processing and machine learning communities [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [24]. The two most commonly used architectures are auto-encoders [10], [11], [15], [16], [17], [24] and recurrent neural networks [13], [14], [18], [19]. The general strategy employed is to introduce a bitrate bottleneck layer, usually in the form of an autoencoder, in the encoder-decoder image compression pipeline. The networks are trained by minimizing a loss function that penalizes the difference between the original sRGB image and its reconstruction. The loss function is usually the mean-squared error between the original image and the compressed output image [10], [11], [15], [17], or is based on perceptual metrics, such as MS-SSIM [14], [16], [24].

The architectures of [17] and [11] are representative examples of the autoencoder category. They support multiple bitrates and employ a cascade of autoencoders. The output of the encoder is quantized, and while [17] adopts a smooth approximation of the derivative of the non-differentiable quantization function, [11] substitutes it with a continuous relaxation by adding uniform noise. [16] also employs an autoencoder architecture and explores the possibility of using adversarial training within a compression setting. In the recurrent neural network (RNN) framework, the model generates a progressive encoding that grows with the number of recurrent iterations. Depending on the target bitrate, only a subset of the progressive code is transmitted and multiple bitrates are achieved. At the decoder, the missing codes are either sampled from the learned distribution [13] or ignored [19]. Unlike [17] and [11], the methods of [14], [19] binarize the output of the encoder rather than quantizing it.

As discussed in Section 1, the work in this paper is inspired by Nguyen and Brown [1], where the information required to reconstruct the raw image is embedded inside a compressed JPEG image. However, unlike [1], where the raw image reconstruction and JPEG compression were applied independently, this paper explores the opportunity to modify a neural network-based compressor's objective to be aware of the raw reconstruction. This requires careful consideration as to how the sRGB-raw mapping can be modelled such that it can be differentiated in an appropriate manner to allow for backpropagation.

3 Raw Image Reconstruction as a Lookup Operation

The raw image reconstruction procedure of Nguyen and Brown [1] calculates the necessary parameters to reconstruct the original raw image (up to a quantization factor) from the corresponding sRGB image given an sRGB-raw image pair. More specifically, the parameters modelled are a white-balance matrix, a color correction matrix, a tone map operator that is applied to all color channels, and a gamut mapping that maps a 3D color value to a new 3D color value. Fig. 2 shows an illustration of these steps that reflects the typical processes that a raw image undergoes onboard a camera to produce the sRGB image. As with all other radiometric calibration/raw reconstruction techniques in the literature, the method of [1] does not explicitly model the compression and assumes the sRGB image used to estimate the mapping parameters is high-quality.

Based on the given sRGB-raw image pair, the work in [1] essentially computes a distinct mapping from the sRGB color space to the raw color space. Therefore, for a particular pixel value in the sRGB space, there exists a corresponding pixel intensity in the raw space which is fixed for that sRGB-raw pair. An implication of this is that it is possible to build a 3D lookup table that encodes this mapping from sRGB to the raw color space. Each axis of the 3D lattice represents one of the three color channels and an input sRGB pixel value defines a point inside the lattice. The output raw pixel values stored in the 3D LUT can thus be indexed by the input sRGB pixel intensities. We note that use of a 3D LUT for raw reconstruction has been demonstrated by Lin et al. [27]. However, they did not consider issues such as the degree of the interpolation function used in the LUT and its implication for differentiation, which is critical to our goal of using it as part of a deep learning-based loss function.

A full 3D LUT for an input sRGB image spanning the 0-255 intensity range requires $256^3\times3$ color channels ≈ 48 million parameters. If each entry is stored as type `float32`,

then the LUT size will be approximately 190 MB. This is highly inefficient in terms of storage and memory usage. A common alternative is to use a smaller LUT, and if an input value is not a lattice point, the result of the lookup operation is computed by interpolating the nearby lattice values.

To examine how raw reconstruction accuracy is affected by the size of the LUT and the degree of the local interpolation function, we applied the algorithm of Nguyen and Brown [1] on 100 randomly selected sRGB-raw image pairs from the NUS dataset [28], [29] and then used the parameters estimated by their method to build, for each pair, LUTs of various sizes and different degrees of interpolating polynomials. We tested tables of sizes 64, 32, 16, and 8, with interpolating polynomials whose variables in each term have degree 1, 2, and 3. We then computed the root mean squared error (RMSE) between the ground truth raw images from the NUS dataset and the raw images reconstructed using the LUTs. From our experiments, we found that even a LUT of size $16\times16\times16$ using a first-degree interpolating function produces an accurate raw reconstruction. This suggests that the sRGB-raw mapping function, although hard to analytically express and differentiate due to the various operations involved, is piecewise smooth. Therefore, meaningful gradients can be obtained by differentiating the local interpolating functions encoded within the LUT, and these can be backpropagated while training the network.

The coefficients of the derivatives of the interpolating functions themselves, just as the other entries of the table, can be pre-computed and stored within the LUT rendering both the raw reconstruction and the calculation of the derivative during backpropagation as fast and efficient lookup operations. For example, a LUT of size $16\times16\times16$ with a first-degree interpolating function requires $16^3\times(8\times3+4\times3)\approx0.14$ million parameters, where 3 represents the number of color channels, and 8 and 4 denote the number of coefficients of the interpolating function and the number of coefficients of the derivative of the interpolating function, per color channel. If each value is saved as type `float32`, then the LUT will be around half an MB in size. For all our experiments in Section 5, we used LUTs of size $16\times16\times16$ with a first-degree interpolating function as they offered the best compromise between raw reconstruction accuracy, training time, and memory requirements.

A LUT generated using the approach of [1] is specific to that sRGB-raw pair. This means that at training time, the raw data corresponding to each sRGB image must be available. While it is possible to construct generic camera-specific LUTs, this process requires careful camera calibration [5]. The technique of [1] has the advantage that it requires no knowledge of the camera parameters, and thus we select their method to provide a simple calibration-free pipeline.

4 Raw reconstruction loss for neural network-based image compression

As discussed in Section 1, we select the state-of-the-art image compression network of Toderici et al. [19] to demonstrate the utility of our proposed raw reconstruction loss. The architecture of Toderici et al. [19] is built on an RNN-based encoder and decoder, and a binarizer. At each recurrent iteration, the RNN encoder encodes the residual between the previous reconstruction of the sRGB image and the original uncompressed sRGB image. New information from the current residual is extracted and combined with the context stored in the hidden states of the recurrent layers at each step. The model thus generates a progressive encoding that grows with the number of recurrent iterations. The number of iterations that the decoder runs is determined by the number of recurrent iterations. Variable bitrates are thereby achieved – the decoder runs fewer iterations for lower bitrate encodings and generates a poorer-quality reconstruction as compared to higher bitrate encodings.

Within the same architecture, Toderici et al. [19] examine three different RNN types (LSTM, associative LSTM, and a hybrid of GRU and ResNet) and three different reconstruction methods (one-shot, additive, and residual scaling). For our experiments, we choose the associative LSTM as our recurrent unit variant, and the additive reconstruction framework to plug in our raw reconstruction loss. We would like to point out that this choice is arbitrary, and our proposed scheme is equally applicable to any of the models described in their paper.

We briefly review the recurrent model of Toderici et al. [19] before proceeding to an explanation of how our raw reconstruction loss can be incorporated within their framework. Following Toderici et al. [19], a single iteration of their network can be expressed as:

$$b_t = B(E_t(x-\hat{x}_{t-1})),\quad \hat{x}_t = D_t(b_t)+\hat{x}_{t-1},\quad \hat{x}_0=0, \quad (1)$$

where E, B, and D represent the encoder, binarizer, and decoder, respectively. The parameter t denotes the recurrent iteration, and b_t and $\hat{x}_t$ are the progressive binary representation and the progressive reconstruction, respectively, of the original image x.

See Fig. 1 for the network architecture of Toderici et al. [19]. The input is an sRGB patch of size $32\times32\times3$ pixels, which is reduced to a $2\times2\times32$ binarized representation at each recurrent iteration under the action of the binarizer. As a result, each iteration represents $\frac{1}{8}$ bit per pixel (bpp). The batch size is set to 32 and the number of RNN unroll iterations is selected as 16.

4.1 Loss function

Toderici et al. [19] impose an L_1 loss on the residuals generated at each iteration as

$$\Phi_t(x,\hat{x}_t) = |x-\hat{x}_t|. \quad (2)$$

To this existing fidelity loss in the sRGB space, we append our raw reconstruction loss as:

$$\Phi_{t_{\text{raw}}}(x_{\text{raw}},\hat{x}_{t_{\text{raw}}}) = |x_{\text{raw}}-\hat{x}_{t_{\text{raw}}}|, \quad (3)$$

where x_{raw} is the ground truth raw image. Note that we estimate the raw reconstruction $\hat{x}_{t_{\text{raw}}}$ at every recurrent iteration i.e., $\hat{x}_{t_{\text{raw}}}(=\Psi(\hat{x}_t))$ is obtained by applying the lookup operation (described in Section 3) on the sRBG reconstruction $\hat{x}_t$ produced by the network at iteration t. The total loss for the network during training is

$$\beta\left(\sum_t \Phi_t(x,\hat{x}_t)+\lambda\sum_t \Phi_{t_{\text{raw}}}(x_{\text{raw}},\hat{x}_{t_{\text{raw}}})\right), \quad (4)$$

where β, λ are positive scalars, and λ controls the contribution of our raw reconstruction loss to the total loss.

5 EXPERIMENTS

We validate the utility of our proposed approach on DSLRs as well as mobile phone images. For our experiments on DSLR data, we use the NUS dataset [28], [29], which contains raw images captured using several different DSLR cameras with approximately 200 to 350 images per camera. To evaluate our method on mobile phone data, we use the dataset of [30] comprising raw files acquired using smart phones with 100 images per camera.

5.1 Training setup

Since the raw-RGB color space is camera-specific, we train our models separately for each camera. For our first experiment, we choose the Samsung NX2000 DSLR camera from the NUS dataset, which has 307 images. We split these images randomly into training (60%), validation (20%), and test (20%) sets. To obtain the sRGB images from the raw files, we use the platform of Karaimer and Brown [31]. Their framework takes the raw image as input, and offers the flexibility to pause the camera pipeline before the JPEG compression stage and save the uncompressed sRGB output. Using this platform, we also obtain the demosaiced raw image before the white-balancing and color manipulation stages. This demosaiced unprocessed raw image serves as ground truth for both training and evaluating our models.

We downsize by a factor of 4 all the sRGB-raw pairs thus obtained to roughly match the image resolution used by [19][1]. Next, we use the raw image reconstruction parameters estimated by the method of Nguyen and Brown [1] to build lookup tables for all pairs in the training set. We then decompose the sRGB training images into non-overlapping 32×32 patches, and use for training 640 patches from each image that have the worst compression ratio when using the PNG compression algorithm. This is based on the observation by the authors of [19] that training on patches that are hard to compress yields a better compression model.

Corresponding patches from the raw images are also cropped and stored for computation of our raw reconstruction loss during training. Although the number of patches required for training is very high, the number of training images from which these patches are extracted is only of the order of a few hundred, and hence, the number of LUTs to be built and saved, before training, is not prohibitive.

Only the *pre-trained* TensorFlow model for the network architecture of Toderici at al. [19] is publicly available. Neither training data nor the code for training has been released by the authors. Therefore, we used a re-implementation of their network architecture in PyTorch, and appended our raw reconstruction loss to their framework to train all our own models described in the paper. All networks were trained using the Adam [32] optimizer. The learning rate was set to 5×10^{-4}, and we used a fixed value of $\lambda = 5$ for all our experiments. We experimented with β values of $\frac{1}{10}$, $\frac{1}{25}$, $\frac{1}{50}$, $\frac{1}{100}$, and $\frac{1}{200}$. For the Samsung NX2000, a β value of $\frac{1}{50}$ was selected. The training is performed from scratch on an nVidia Titan X GPU with 12 GB of memory. The training time per batch for the original architecture of [19] with only the fidelity loss is approximately 0.7 seconds. With our raw loss added, the training time increases by approximately 0.2 seconds. All our trained models and associated data used in this paper will be made publicly available.

1. The dataset of Toderici et al. [19] contains images of size 1280×720 pixels. The images from Samsung NX2000 after downsizing by 4 are of size 1344×896 pixels.

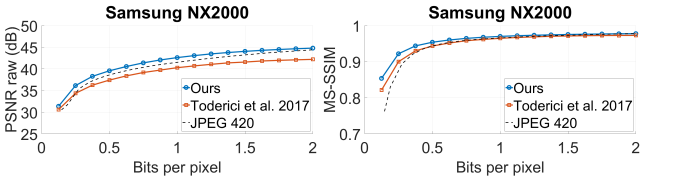


Fig. 3. Rate distortion curves on test images from Samsung NX2000 given as PSNR raw on the left, and MS-SSIM sRGB on the right, versus bpp.

5.2 Comparisons and evaluation metrics

To evaluate our trained model, we use the sRGB images from the test set. Note that raw images have to be provided to our network only during training. At test time, just as with other compression networks, including Toderici et al. [19], only the sRGB image needs to be input to our network.

We compare our results with the pre-trained TensorFlow model for the Residual GRU architecture made publicly available by the authors of [19]. Since we do not perform entropy coding on our results (as this is not the objective of our work), we also skip the entropy coding step described in [19] to allow for a fair comparison. Note that both models, ours and [19], would benefit by approximately the same amount if an entropy coding scheme is applied, and thereby our experimental findings remain unaffected even if this step is skipped. Of the various architectures described in their paper, only the pre-trained Residual GRU model has been released by the authors, and this according to their paper is one of their best-performing models. Following [19], we also include a baseline comparison against JPEG.

To examine the improvement in raw reconstruction using our approach, we first pre-compute the raw reconstruction parameters corresponding to each sRGB-raw image pair in the test set using the algorithm of [1]. Following [1], we assume that this 64 KB metadata is available to all methods (ours, [19], and JPEG) at test time. On the compressed sRGB output of our network, we apply the raw recovery technique of [1] using this metadata to obtain our raw reconstruction. Likewise, we use the same metadata to recover the corresponding raw reconstructions from the sRGB output of [19], as well as JPEG. We then compute the PSNR between the ground truth raw image and the raw reconstruction obtained using each of the three approaches.

It is worth noting that while it is possible to objectively assess raw reconstruction accuracy using PSNR, the same is not true of sRGB images [19] since the human visual system is more sensitive to certain types of distortions than others, a fact that JPEG also exploits. Therefore, following other recent works [11], [14], [16], [19], [24], we use MS-SSIM [20], which is a commonly employed perceptual full-reference image metric for comparing lossy image compression algorithms, for quantitative evaluation of our sRGB outputs.

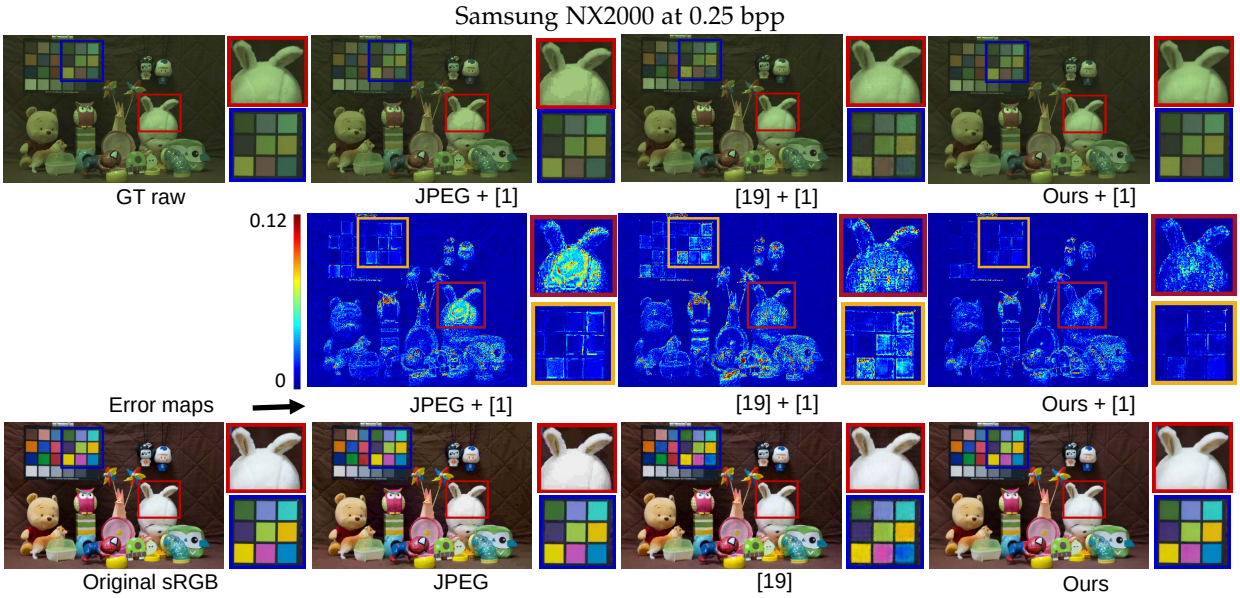


Fig. 4. Qualitative results of our method, along with comparisons. The ground truth (GT) raw image, and the raw reconstructions obtained by applying the raw image reconstruction technique of Nguyen and Brown [1] (with the same 64 KB raw reconstruction metadata) on JPEG, the output of Toderici et al. [19], and our output, respectively, are shown in the first row. The root squared error maps (with GT raw as reference) of the raw reconstructions in row one are presented in the second row. The third row shows the original sRGB, JPEG, output of [19], and our output. Zoomed-in regions are also displayed for comparison. Note that a gamma has been applied on the raw images for better visualization.

TABLE 1
Performance on various cameras measured as area under the curve (AUC) for the specified metric, up to 2 bits per pixel.

Camera	Samsung NX2000		Canon 1Ds Mark III		Olympus E-PL6		Google Pixel	
Metric AUC	PSNR raw (dB)	MS-SSIM sRGB	PSNR raw (dB)	MS-SSIM sRGB	PSNR raw (dB)	MS-SSIM sRGB	PSNR raw (dB)	MS-SSIM sRGB
JPEG 420	75.83	1.7670	84.22	1.7823	83.52	1.7670	70.79	1.7119
Toderici et al. [19]	74.06	1.7876	82.12	1.8022	81.63	1.7896	70.66	1.7274
Ours with both losses	**78.30**	1.8027	**85.98**	1.8229	**85.78**	1.8083	**72.02**	1.7365
Ours with only fidelity loss	75.93	**1.8102**	84.28	**1.8264**	83.65	**1.8138**	71.01	**1.7528**

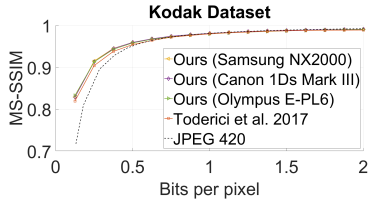
Both metrics – PSNR raw and MS-SSIM sRGB – are calculated over the reconstructed images after each iteration for both our own results and that of [19]. The results are presented in the plot of Fig. 3. Qualitative evaluations are also provided in Fig. 4, and it can be seen that our raw reconstruction has fewer errors, and our sRGB output is visually superior to both [19] and JPEG.

To rank the results, we use the same score reporting procedure as in [19] – we use an aggregate measure computed as the area under the rate-distortion curve as a single scalar-valued measure of performance. The AUC values of our model trained using both losses and those of [19] and JPEG are presented in Table 1. It can be observed from the results that JPEG produces a higher PSNR raw AUC than [19], but its MS-SSIM AUC scores are lower than [19]. Our model outperforms both competitors on both metrics. As mentioned in Section 1, since the work of [19], other methods have been published that outperform [19]. However, our objective is in determining the improvement in raw image reconstruction due to our joint loss given a particular network architecture, and as a result, we compare our results only with [19].

5.3 Effect of our raw reconstruction loss

In the previous experiment, the training and testing were specific to a camera – that is, the patches used for training and the images used for testing came from the same camera. It is natural to consider whether the improvement in reconstruction quality over the generic model of [19] is simply the result of the sRGB fidelity term in our joint loss being better constrained. To ascertain the contribution of our raw reconstruction loss in improving the raw reconstruction quality, we perform the following experiment. We turn off our raw reconstruction loss by setting $\lambda = 0$ in equation (4), and train from scratch using only the sRGB fidelity loss. We use the same Samsung NX2000 training patches and train for the same number of epochs as before. The PSNR raw and MS-SSIM sRGB AUCs obtained using our model trained with only the fidelity loss are provided in Table 1. It can be observed that while the MS-SSIM performance of our two models is nearly the same, our PSNR raw values improve substantially when trained using both losses – the PSNR raw AUC increases from 75.93 to 78.30. This clearly demonstrates the effectiveness of our raw reconstruction loss in improving the raw reconstruction quality. The MS-SSIM AUC of our model trained with just the fidelity loss at 1.8102 is only slightly greater than the 1.8027 AUC produced by our network trained using both losses.

We repeat the same experiments on the Canon 1Ds Mark III and the Olympus E-PL6 DSLRs from the NUS dataset. We train two separate models from scratch – the first with



Metric AUC	MS-SSIM sRGB
Samsung NX2000	1.8171
Canon 1Ds Mark III	1.8174
Olympus E-PL6	1.8157
Toderici et al. 2017	1.8120
JPEG 420	1.7990

Fig. 5. Left: rate distortion curves on the Kodak dataset [33] given as MS-SSIM, versus bpp. Right: AUC up to 2 bpp for the MS-SSIM plot.

our joint loss, and the second using only the fidelity loss. The corresponding AUC values are given in Table 1. Our joint loss model yet again outperforms by a notable margin our fidelity-only model in terms of PSNR raw. The drop in MS-SSIM is negligibly small, as can be seen from the results.

5.4 Generalizability

In all our experiments so far, the training and test images were drawn from the same dataset and the images are similar in terms of scene content. To determine the generalizability of our models trained using both losses, we test on the Kodak Photo CD dataset [33] that the authors of [19] have used to report their scores on. We would like to note that this dataset has also been used by other recent image compression works, such as [12], [14], [15], [16], [17], [24] for score reporting. Since the ground truth raw images are unavailable in this case, we report only the MS-SSIM values on the sRGB images. See the plot of Fig. 5. The AUC values are also provided alongside the plot. It can be observed that our models trained from scratch using both losses perform on par and even surpass [19], even though our training data did not contain images from a Kodak camera.

5.5 Mobile phone data

The NUS dataset contains only images captured using DSLRs. For our next experiment, we use the recent dataset of [30] that contains raw images captured using mobile phones. To test our framework on mobile data, we used the 100 raw images from this dataset captured using the Google Pixel mobile phone camera. Since the amount of training data is smaller in this case, we use a split of 80% training, 10% validation, and 10% test. We train two models once again from scratch (using our joint loss and only the fidelity loss). Mobile phone images tend to be noisier and of a poorer quality than DSLRs, and we do notice a drop in performance of our method as well as competing techniques. However, as seen from the AUC values in Table 1, our models still outperform [19] and JPEG on both metrics. Our joint loss network again scores favourably over the fidelity-only model with regard to PSNR raw, while providing competitive performance in terms of sRGB fidelity.

5.6 Fine-tuning

Since the raw-RGB color space is camera-dependent, our models have to be trained separately for each camera. In practice, sufficient training data may not be available to train the model from scratch, or it may be desirable to use an existing trained compression network in our raw reconstruction scenario. To address this issue, we advocate

TABLE 2
Performance comparison of our fine-tuned models versus our models trained from scratch.

Metric AUC	PSNR raw (dB)	MS-SSIM sRGB
Camera	Samsung NX2000	
Fine-tuned	77.63	1.8071
From scratch	78.30	1.8027
Camera	Canon 1Ds Mark III	
Fine-tuned	85.69	1.8240
From scratch	85.98	1.8229
Camera	Olympus E-PL6	
Fine-tuned	85.02	1.8115
From scratch	85.78	1.8083

TABLE 3
Peformance on Sony A57.

Metric AUC	PSNR raw (dB)	MS-SSIM sRGB
JPEG 420	83.00	1.7677
[19]	81.73	1.7889
Mixed model	83.89	1.8120
Fine-tuned	85.13	1.8101
From scratch	85.53	1.8029

a fine-tuning strategy where we first train a generic model with only the sRGB fidelity loss, using training patches from different cameras, and then fine-tune this model for the test camera. For this experiment, we pool an equal number of patches from Samsung NX2000, Canon 1Ds Mark III, Olympus E-PL6, and Nikon D5200 from the NUS dataset [28], [29]. The total number of patches is selected to be equal to the number of patches used for training each individual camera model from scratch in our earlier experiments in Section 5.1. We now train the network using this mixed set of patches with only the sRGB fidelity loss. Next, we fine-tune this model using patches from Samsung NX2000, which we select as our test camera. For fine-tuning, we use only about 30,000 training patches. We apply this fine-tuned model on the same test images from Samsung NX2000 as in Section 5.2, and compute the rate distortion curves. The corresponding PSNR raw and MS-SSIM sRGB AUC values are provided in Table 2. The AUC values from Table 1 on the same test images obtained using our original model trained from scratch using both losses are reproduced in Table 2 for ease of comparison. It can be seen that our fine-tuned model performs almost on par with the model trained from scratch in terms of PSNR raw. We performed fine-tuning on Canon 1Ds Mark III and Olympus E-PL6 as well and observed similar trends. The AUC values are provided in Table 2. We do note that the MS-SSIM sRGB AUC values of our fine-tuned networks are slightly higher than our corresponding models trained from scratch. This is because the fine-tuning is done on a network that has originally been trained to maximize sRGB fidelity. We do not run the fine-tuning experiment on mobile phone images because, as mentioned in Section 5.5, there are currently no large publicly available datasets with a sizeable number of raw images from mobile phones sufficient to train a representative mixed model.

To examine the efficacy of our fine-tuning framework in cases where the test camera does not belong to the mixed training set, we select a new camera, the Sony A57,

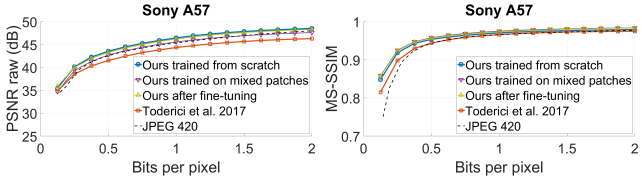


Fig. 6. Rate distortion curves on test images from Sony A57 given as PSNR raw on the left, and MS-SSIM sRGB on the right, versus bpp.

from the NUS dataset [28], [29]. We randomly divide the images into training, validation, and test sets. Next, we fine-tune our earlier mixed model, which did not include Sony, using patches extracted from the training images. The improvement in raw reconstruction accuracy over the mixed model as a result of fine-tuning is evident from the plots of Fig. 6 and the AUC values in Table 3. For comparison, we trained an independent model from scratch for the Sony A57 using both losses. It can be seen from the results that our fine-tuned model with a PSNR raw AUC of 85.13 performs almost on par with our network trained from scratch, which has a corresponding AUC value of 85.53.

For all our fine-tuning experiments, a fixed β value of $\frac{1}{50}$ was used. For our models trained from scratch, β was selected as $\frac{1}{25}$ for Canon 1Ds Mark III, $\frac{1}{50}$ for Olympus E-PL6 and Google Pixel, and $\frac{1}{100}$ for Sony A57. The rate distortion curves corresponding to the experiments on Canon 1Ds Mark III and Olympus E-PL6 in Section 5.3, and Google Pixel in Section 5.5 are provided in the appendix.

6 Summary

This paper has demonstrated how the loss used to train neural network-based compression architectures can be modified to consider both the sRGB image fidelity errors and raw reconstruction errors. Our proposed approach needs only to modify the loss function and requires no changes to the underlying network architecture. As part of this work, we analysed how the sRGB-raw mapping can be modelled as 3D LUTs that are locally differentiable. Our method was applied within two training contexts: (1) training from scratch using the joint loss, and (2) fine-tuning a network trained only on sRGB fidelity loss to also consider the joint loss. In both scenarios, we demonstrated improvements in the raw image reconstruction with modest impact on the sRGB image fidelity. The high-level goal of this paper is to encourage researchers developing the next generation of neural network-based image compression algorithms to consider expanding the utility of the compressed images beyond a photo-centric usage only (i.e., sRGB). To this end, we hope to influence the development of compression schemes and associated encoding formats that are mindful of low-level computer vision tasks that require the camera to serve as a scientific instrument that provides RGB values that are linearly related to the scene's real radiometric properties.

Acknowledgment

This study was funded in part by the Canada First Research Excellence Fund for the Vision: Science to Applications (VISTA) programme and an NSERC Discovery Grant.

References

[1] R. M. H. Nguyen and M. S. Brown, "Raw image reconstruction using a self-contained srgb-jpeg image with only 64 KB overhead," in *CVPR*, 2016.
[2] A. Chakrabarti, Y. Xiong, B. Sun, T. Darrell, D. Scharstein, T. Zickler, and K. Saenko, "Modeling radiometric uncertainty for vision with tone-mapped color images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2185–2198, 2014.
[3] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *SIGGRAPH*, 2008.
[4] M. D. Grossberg and S. K. Nayar, "Determining the camera response from images: What is knowable?" *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1455–1467, 2003.
[5] S. J. Kim, H. T. Lin, Z. Lu, S. Ssstrunk, S. Lin, and M. S. Brown, "A new in-camera imaging model for color computer vision and its application," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2289–2302, 2012.
[6] S. Mann and R. W. Picard, "On being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures," in *Proc. of IS&T*, 1995, pp. 442–448.
[7] T. Mitsunaga and S. K. Nayar, "Radiometric self calibration," in *CVPR*, 1999.
[8] S. Nam and S. J. Kim, "Modelling the scene dependent imaging in cameras with a deep neural network," in *ICCV*, 2017.
[9] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Trans. on Consumer Electronics*, vol. 38, no. 1, 1992.
[10] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *NIPS*, 2017.
[11] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *ICLR*, 2017.
[12] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *ICLR*, 2018.
[13] K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra, "Towards conceptual compression," in *NIPS*, 2016.
[14] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. T. Chinen, S. J. Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *arXiv*, 2017.
[15] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *arXiv*, 2017.
[16] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *ICML*, 2017.
[17] L. Theis, W. Shi, A. Cunningham, and F. Huszr, "Lossy image compression with compressive autoencoders," in *ICLR*, 2017.
[18] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *arXiv*, 2015.
[19] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *CVPR*, 2017.
[20] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers*, 2003.
[21] D. S. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.
[22] "WebP image format." [Online]. Available: https://developers.google.com/speed/webp/
[23] F. Bellard, "BPG image format." [Online]. Available: https://bellard.org/bpg/
[24] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional probability models for deep image compression," in *CVPR*, 2018.
[25] A. Chakrabarti, D. Scharstein, and T. E. Zickler, "An empirical camera model for internet color vision," in *BMVC*, 2009.
[26] L. Yuan and J. Sun, "High quality image reconstruction from raw and JPEG image pair," in *ICCV*, 2011.
[27] H. T. Lin, Z. Lu, S. J. Kim, and M. S. Brown, "Nonuniform lattice regression for modeling the camera imaging pipeline," in *ECCV*, 2012.
[28] D. Cheng, A. Abdelhamed, B. Price, S. Cohen, and M. S. Brown, "Two illuminant estimation and user correction preference," in *CVPR*, 2016.
[29] D. Cheng, B. Price, S. Cohen, and M. S. Brown, "Beyond white: Ground truth colors for color constancy correction," in *ICCV*, 2015.
[30] H. C. Karaimer and M. S. Brown, "Improving color reproduction accuracy on cameras," in *CVPR*, 2018.
[31] ——, "A software platform for manipulating the camera imaging pipeline," in *ECCV*, 2016.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv*, 2014.
[33] Eastman Kodak, "Kodak lossless true color image suite (PhotoCD PCD0992)."