GamutMLP: A Lightweight MLP for Color Loss Recovery

Hoang M. Le[1] Brian Price[2] Scott Cohen[2] Michael S. Brown[1]
[1]York University [2]Adobe Research
{hminle,mbrown}@yorku.ca, {bprice,scohen}@adobe.com

Abstract

Cameras and image-editing software often process images in the wide-gamut ProPhoto color space, encompassing 90% of all visible colors. However, when images are encoded for sharing, this color-rich representation is transformed and clipped to fit within the small-gamut standard RGB (sRGB) color space, representing only 30% of visible colors. Recovering the lost color information is challenging due to the clipping procedure. Inspired by neural implicit representations for 2D images, we propose a method that optimizes a lightweight multi-layer-perceptron (MLP) model during the gamut reduction step to predict the clipped values. GamutMLP takes approximately 2 seconds to optimize and requires only 23 KB of storage. The small memory footprint allows our GamutMLP model to be saved as metadata in the sRGB image—the model can be extracted when needed to restore wide-gamut color values. We demonstrate the effectiveness of our approach for color recovery and compare it with alternative strategies, including pre-trained DNN-based gamut expansion networks and other implicit neural representation methods. As part of this effort, we introduce a new color gamut dataset of 2200 wide-gamut/small-gamut images for training and testing.

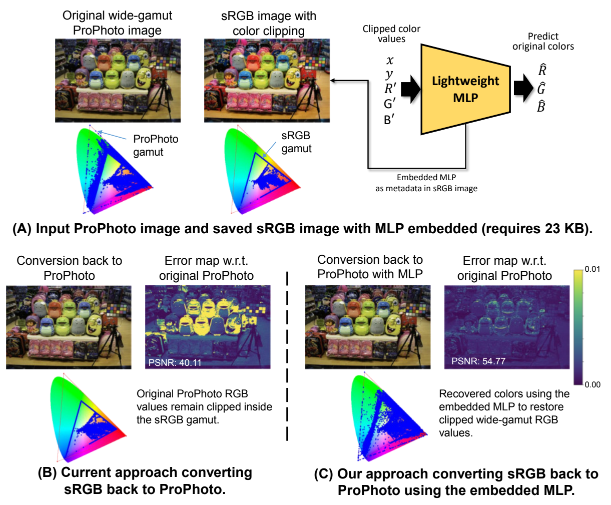

Original wide-gamut ProPhoto image
sRGB image with color clipping
Clipped color values
Predict original colors
x
y
R'
G'
B'
Lightweight MLP
$\hat{R}$
$\hat{G}$
$\hat{B}$
ProPhoto gamut
sRGB gamut
Embedded MLP as metadata in sRGB image
(A) Input ProPhoto image and saved sRGB image with MLP embedded (requires 23 KB).
Conversion back to ProPhoto
Error map w.r.t. original ProPhoto
Conversion back to ProPhoto with MLP
Error map w.r.t. original ProPhoto
0.01
0.00
PSNR: 40.11
PSNR: 54.77
Original ProPhoto RGB values remain clipped inside the sRGB gamut.
Recovered colors using the embedded MLP to restore clipped wide-gamut RGB values.
(B) Current approach converting sRGB back to ProPhoto.
(C) Our approach converting sRGB back to ProPhoto using the embedded MLP.

Figure 1. (A) shows a wide-gamut (ProPhoto) image that has been converted and saved as a small-gamut (sRGB) image; color clipping is required to fit the smaller sRGB gamut (as shown in the chromaticity diagrams). (B) Standard color conversion back to the wide-gamut color space is not able to recover the clipped colors. (C) Conversion back to the wide-gamut RGB using our lightweight GamutMLP (23 KB) can recover the clipped color values back to their original values.

1. Introduction

The RGB values of our color images do not represent the entire range of visible colors. The span of visible colors that can be reproduced by a particular color space's RGB primaries is called a gamut. Currently, the vast majority of color images are encoded using the standard RGB (sRGB) color space [7]. The sRGB gamut is capable of reproducing approximately 30% of the visible colors and was optimized for the display hardware of the 1990s. Close to 30 years later, this small-gamut color space still dominates how images are saved, even though modern display hardware is capable of much wider gamuts.

Interestingly, most modern DSLR and smartphone cameras internally encode images using the ProPhoto color space [12]. ProPhoto RGB primaries define a wide gamut capable of representing 90% of all visible colors [33]. Image-processing software, such as Adobe Photoshop, also uses this color-rich space to manipulate images, especially when processing camera RAW-DNG files. By processing images in the wide-gamut ProPhoto space, cameras and editing software allow users the option to save an image in other color spaces—such as AdobeRGB, UHD, and Display-P3—that have much wider color gamuts than sRGB. However, these color spaces are still rare, and most images are ultimately saved in sRGB. To convert color values between ProPhoto and sRGB, a gamut reduction step is applied that clips the wide-gamut color values to fit the smaller sRGB color gamut. Once gamut reduction is applied, it is challenging to recover the original wide-gamut values. As a result, when images are converted back to a wide-gamut color space for editing or display, much of the color fidelity is lost, as shown in Figure 1.

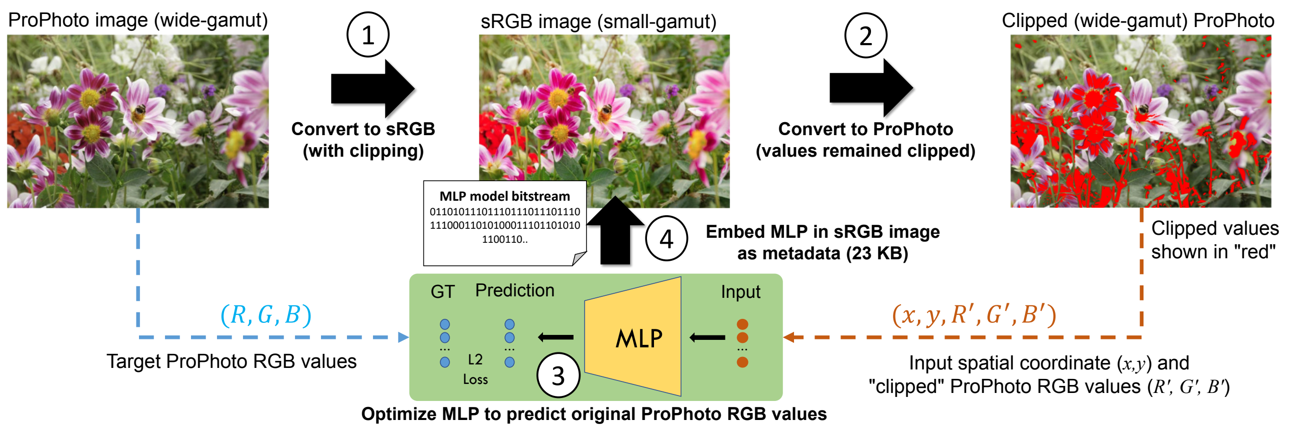

ProPhoto image (wide-gamut)
1
Convert to sRGB (with clipping)
sRGB image (small-gamut)
2
Convert to ProPhoto (values remained clipped)
Clipped (wide-gamut) ProPhoto
MLP model bitstream
0110101110111011101110111011101110
11100011010100011101101010
1100110..
4
Embed MLP in sRGB image as metadata (23 KB)
Clipped values shown in "red"
(R, G, B)
Target ProPhoto RGB values
GT
Prediction
Input
MLP
L2 Loss
3
(x, y, R′, G′, B′)
Input spatial coordinate (x,y) and "clipped" ProPhoto RGB values (R′, G′, B′)
Optimize MLP to predict original ProPhoto RGB values

Figure 2. An overview of the *gamut reduction* stage in our framework. This phase shows the gamut reduction step, where the wide-gamut ProPhoto is converted to the small-gamut sRGB. While saving the sRGB image, an MLP is optimized based on the original and clipped ProPhoto color values. The MLP is embedded in the sRGB image as metadata.

Contribution We address the problem of recovering the RGB colors in sRGB images back to their original wide-gamut RGB representation. Our work is inspired by coordinate-based implicit neural image representations that use multilayer perceptrons (MLPs) as a differentiable image representation. We propose to optimize a lightweight (23 KB) MLP model that takes the gamut-reduced RGB values and their spatial coordinates as input and predicts the original wide-gamut RGB values. The idea is to optimize the MLP model when the ProPhoto image is saved to sRGB and embed the MLP model parameters in the sRGB image as a comment field. The lightweight MLP model is extracted and used to recover the wide-gamut color values when needed. We describe an optimization process for the MLP that requires ∼2 seconds per full-sized image. We demonstrate the effectiveness of our method against several different approaches, including other neural image representations and pre-trained deep-learning-based models. As part of this work, we have created a dataset of 2200 wide-gamut/small-gamut image pairs for training and testing.

2. Related work

The following discusses three areas related to our work: (1) gamut reduction and expansion, (2) RAW image recovery methods, and (3) coordinate-based implicit neural functions.

Gamut reduction/expansion When converting between color spaces, it is necessary to address the gamut mismatch. There are many strategies for gamut reduction and expansion in the literature (e.g., [1, 18, 20, 21, 28]). The most common approach for both gamut-reduction and gamut-expansion uses *absolute colorimetric* intent, where the goal is to minimize color distortion between the two gamuts. For example, in the case of gamut reduction from ProPhoto to sRGB, colorimetric errors are minimized by projecting (and clipping) out-of-gamut (OG) ProPhoto color values to the boundary of the sRGB space, as shown in Figure 1. When converting back from sRGB to ProPhoto, the absolute colorimetric strategy minimizes color error by leaving the clipped values untouched. Absolute colorimetric reduction and expansion is a common strategy used by consumer cameras and image-editing software, such as Adobe Lightroom, DarkTable, and RawTherapee. Correcting the loss of color fidelity for color expansion is the goal of this paper. A less common approach for reduction and expansion is to use soft-clipping [20], where the out-of-gamut values in ProPhoto are compressed to fit within a specified region in the sRGB gamut. For example, instead of clipping out-of-gamut ProPhoto values, they are compressed to fit within the outer 10% of the sRGB gamut. On gamut expansion, the compressed region is expanded back to fill the ProPhoto gamut. While soft-clipping helps restore wide-gamut color values, it incurs colorimetric error during the gamut reduction to sRGB; as a result, most cameras and software do not use this. Recent work by Le et al. [16] proposed a DNN-based network to learn gamut-expansion based on a large dataset of ProPhoto and sRGB images. While this can improve gamut expansion, we show that our MLP optimization outperforms such pre-trained DNN networks by a wide margin.

RAW recovery/derendering Also related to our task are approaches for sRGB *derendering*, where the goal is to recover the original RAW sensor image from an sRGB input. Early approaches to this problem carefully modeled the in-camera rendering process to perform derendering [4, 5, 10, 13], while recent methods train DNN-based models for this task [17, 23]. Our method is similar to works that save small amounts of specialized metadata in the sRGB image to assist in the recovery problem. Such metadata can be in the form of a parametric model [25, 26]

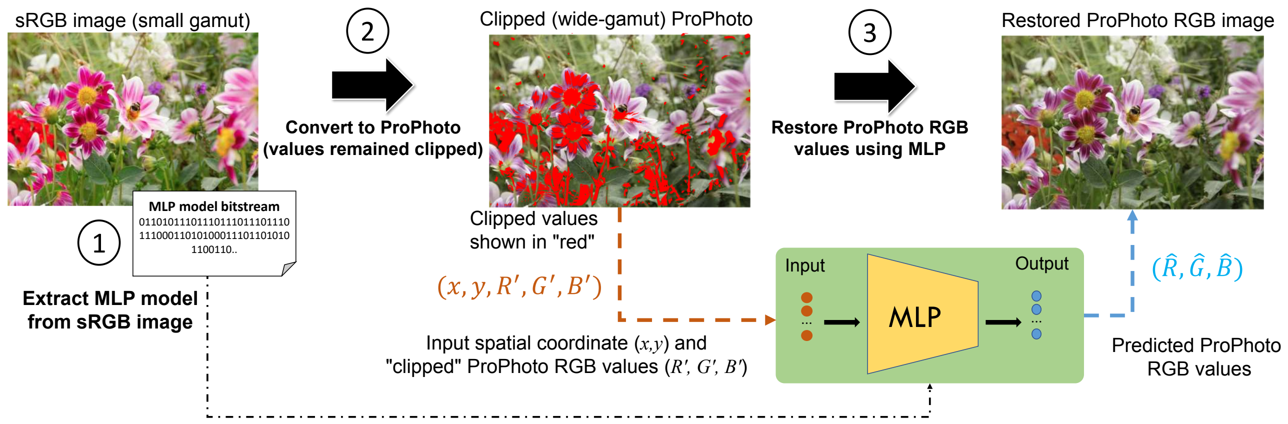

sRGB image (small gamut)
2
Convert to ProPhoto (values remained clipped)
Clipped (wide-gamut) ProPhoto
3
Restore ProPhoto RGB values using MLP
Restored ProPhoto RGB image
MLP model bitstream
0110101110111011101110111110
1110001101010001110110101010
1100110..
1
Extract MLP model from sRGB image
Clipped values shown in "red"
(x, y, R', G', B')
Input
MLP
Output
$(\hat{R}, \hat{G}, \hat{B})$
Input spatial coordinate (x,y) and "clipped" ProPhoto RGB values (R', G', B')
Predicted ProPhoto RGB values

Figure 3. This figure provides an overview of the *gamut expansion* phase in our framework. In particular, the MLP network is extracted and used to recover the clipped ProPhoto values.

or RAW pixel samples [24, 29]. While having access to a reconstructed RAW image would allow it to be re-rendered back to a wide-gamut ProPhoto format, such rendering requires camera-specific photo-finishing parameters that are often not readily available. Instead, we compare our method with that of [15], which proposed to store out-of-gamut ProPhoto samples as metadata in the sRGB. The out-of-gamut samples were used to estimate a polynomial function for gamut expansion. We show that our MLP-based approach provides significantly better results than those obtained by [15] with a smaller memory footprint.

Neural implicit functions Finally, our approach is also inspired by recent advances in neural implicit functions, such as Fourier features [35], SIREN [32], and NeRF [19]. We show that SIREN works well for this task, but requires a much larger model and slower optimization time.

3. GamutMLP for color recovery

We begin with a high-level overview of our gamut recovery framework in Section 3.1. Details on the MLP architecture and optimization are provided in Section 3.2.

3.1. Framework overview

Figure 2 and Figure 3 show illustrations of our framework's two steps: gamut reduction and gamut expansion.

Gamut reduction The gamut reduction step is performed when the image is being converted from ProPhoto to sRGB, either on a camera or image editing software. We assume the input to be a wide-gamut ProPhoto RGB image denoted as $\mathbf{I}_{\text{PP}} \in \mathbb{R}^{3\times N}$, where N is the number of pixels. Gamut reduction is performed using the absolute colorimetric intent described in the previous section. The original ProPhoto image is transformed to the unclipped sRGB image using a 3×3 matrix such that the in-gamut sRGB values fall within the range $[0,1]$. Out-of-gamut sRGB values are then clipped and processed with a gamma encoding to produce the final sRGB image. This procedure can be written as:

$$\mathbf{I}_{\text{sRGB}} = g(\text{clip}(\mathbf{M}\mathbf{I}_{\text{PP}}, min=0, max=1)), \tag{1}$$

where $\mathbf{M}$ is the matrix that maps between ProPhoto and the unclipped sRGB, clip() is the clipping operation, and g is the gamma-encoding for sRGB [7].

When converting sRGB colors back to ProPhoto RGB using the inverse transforms, the clipped color values will not be recovered. We refer to this clipped ProPhoto image as $\mathbf{I}_{\text{ClippedPP}} \in \mathbb{R}^{3\times N}$. Pixels with clipped values are illustrated in red in Figure 2. We express the mapping from sRGB to clipped ProPhoto as:

$$\mathbf{I}_{\text{ClippedPP}} = \mathbf{M}^{-1}g^{-1}(\mathbf{I}_{\text{sRGB}}), \tag{2}$$

where $g^{-1}(\cdot)$ is a de-gamma function for the input sRGB image, and $\mathbf{M}^{-1}$ is the inverse transform to convert the sRGB image back to the ProPhoto color space.

Applying Eqs. 1 and 2, we have the original ProPhoto image, $\mathbf{I}_{\text{PP}}$, and its clipped $\mathbf{I}_{\text{ClippedPP}}$. We also know which values were clipped. Using these two images, we optimize a lightweight MLP (GamutMLP) to predict a residual value that, when added to the $\mathbf{I}_{\text{ClippedPP}}$ recovers $\mathbf{I}_{\text{PP}}$. The GamutMLP model parameters are embedded in the sRGB image when it is saved. Since the parameters of our MLP require only 23 KB of memory, this can easily be embedded as a comment field in the image.

Gamut expansion step Given an sRGB image with an embedded GamutMLP model, we extract the model and perform the standard color space conversion described in Eq. 2 to compute $\mathbf{I}_{\text{ClippedPP}}$. The extracted model predicts the residuals of all pixels and adds them to the $\mathbf{I}_{\text{ClippedPP}}$ to recover the color values as shown in Figure 3.

The following section provides details of the GamutMLP model architecture and its optimization.

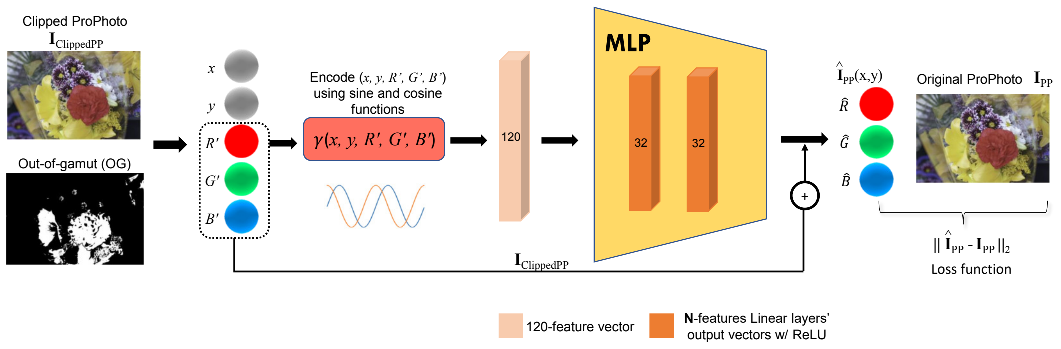

Clipped ProPhoto
$\mathbf{I}_{\text{ClippedPP}}$
Out-of-gamut (OG)
x
y
R'
G'
B'
Encode (x, y, R', G', B') using sine and cosine functions
$\gamma(x, y, R', G', B')$
120
MLP
32
32
$\mathbf{I}_{\text{ClippedPP}}$
+
$\hat{\mathbf{I}}_{\text{PP}}(x,y)$
$\hat{R}$
$\hat{G}$
$\hat{B}$
Original ProPhoto $\mathbf{I}_{\text{PP}}$
$\|\hat{\mathbf{I}}_{\text{PP}} - \mathbf{I}_{\text{PP}}\|_2$
Loss function
120-feature vector
N-features Linear layers' output vectors w/ ReLU

Figure 4. This figure shows the GamutMLP architecture. Given the clipped ProPhoto image, we optimize the MLP using samples from in-gamut and out-of-gamut pixels. The 5D coordinate and color input (x, y, R', G', B') is encoded as a 120D-feature vector before passing it to the MLP. The MLP has three linear layers, with the final layer predicting a residual to add to the R', G', B' input. The loss is computed against the original ProPhoto image R, G, B values.

3.2. GamutMLP model and optimization

Figure 4 shows a diagram of the GamutMLP architecture. When converting from ProPhoto to sRGB, we keep track of the transformed RGB values that lie outside the sRGB gamut. These pixel locations are denoted in the out-of-gamut mask in Figure 4. Out-of-gamut pixels will be clipped to fit within the sRGB gamut.

The MLP input is a 5D vector of a pixel's spatial coordinates (x, y) and color value (R', B', G') from the clipped wide-gamut ProPhoto image. GamutMLP predicts the residual that needs to be added to $\mathbf{I}_{\text{ClippedPP}}$ to recover the wide-gamut original (R, G, B). We can express GamutMLP as follows:

$$\hat{\mathbf{I}}_{\text{PP}}(\mathbf{x}) = f_\theta(\mathbf{x}, \mathbf{I}_{\text{ClippedPP}}(\mathbf{x})) + \mathbf{I}_{\text{ClippedPP}}(\mathbf{x}), \qquad (3)$$

where f_θ represents the *GamutMLP*, θ is the model's parameters, and $\hat{\mathbf{I}}_{\text{PP}}$ is the final recovered ProPhoto image. The MLP's input values (x, y, R', G', B') are normalized to the range $[-1, 1]$, and then pass to the encoding function γ. The use of an encoding function has been shown effective in improving neural implicit representations optimization [19, 30]. Prior neural implicit representations tend to have only 2D coordinates as input, while we apply the encoding function to both spatial coordinates and RGB values. We found the following mapping worked well for our task:

$$\gamma(m) = (\sin(2^0\pi m), \cos(2^0\pi m), ..., \sin(2^{K-1}\pi m), \cos(2^{K-1}\pi m)), \qquad (4)$$

where m is a spatial coordinate or RGB value. In our experiment, we choose K = 12. The γ function projects each of the 5D input values to a 24-dimension encoding, resulting in a final 120D feature vector for each input. The GamutMLP has three linear layers. The first two are fully connected ReLU layers with 32 output features. The last layer outputs three values and has no activation function. Our MLP is optimized with an L_2 loss function computed between the predicted ProPhoto image and the original ProPhoto image:

$$\mathcal{L}_{gamut} = \sum_{\mathbf{x}} ||(\hat{\mathbf{I}}_{\text{PP}}(\mathbf{x}) - \mathbf{I}_{\text{PP}}(\mathbf{x}))||_2^2. \qquad (5)$$

Pixel sampling and standard optimization We describe two optimization strategies: standard and fast. For our standard optimization, model parameters are randomly initialized. At optimization time, we know which pixels are out-of-gamut (OG) and which are in-gamut (IG). We found that training the MLP on both out-of-gamut pixels and in-gamut (non-clipped) pixels gave the best result. We optimized our model by sampling 2% of the IG pixels and 20% of the OG pixels uniformly over their spatial coordinates. For all reported results, the model was optimized for 9,000 iterations with a learning rate of $1e{-}3$ using the Adam optimizer [14].

Faster MLP optimization To speed up our optimization, we used the recent methods proposed by [9, 22] to improve optimization time. We also incorporated a meta-learning strategy [27] to pre-train a generic GamutMLP whose model parameters can be used for initialization. Such initialization has been shown to help coordinate-based MLPs converge faster during optimization [34].

To pre-train a meta-MLP model, we used images from our training dataset described in the following section. For each meta epoch, the meta-MLP fits each image with 10,000 iterations using a larger learning rate $1e - 2$ and the SGD optimizer. When a per-image GamutMLP is initialized with this pre-trained MLP, our optimization requires only 1,200 iterations instead of 9,000. Combining our meta-MLP for initialization with [9, 22] significantly reduced optimization time.

4. Dataset and results

We first describe our dataset generation for evaluating this work. Our dataset images are used to train our meta-GamutMLP model (for weight initialization) and competing DNN-based methods and to test results.

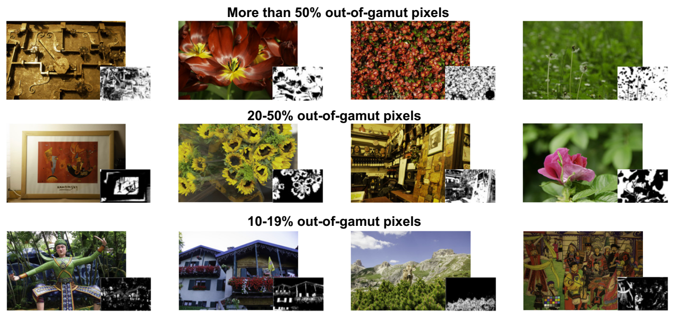
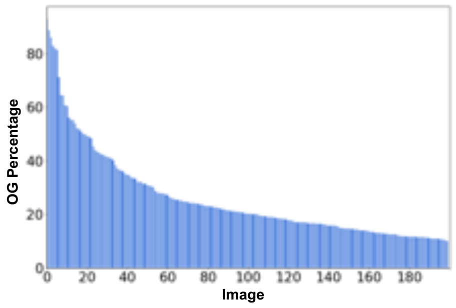

More than 50% out-of-gamut pixels
20-50% out-of-gamut pixels
10-19% out-of-gamut pixels


Figure 5. (Left) Examples from our dataset showing different amounts of out-of-gamut pixels (OG pixels are white in the masks). (Right) A histogram of our dataset in terms of percentage of out-of-gamut pixels.

4.1. Dataset

To prepare wide-gamut ProPhoto images, we followed a procedure used in our prior work [16] that processed RAW images from the MIT-Adobe FiveK [3], RAISE [8], Cube+ [2], and NUS [6] public datasets. There are 16,599 RAW images from these four datasets, representing a wide range of scene content. We use Adobe Camera RAW (ACR) to mimic a camera ISP to render the RAW images to 16-bit wide-gamut ProPhoto images. ACR can apply different photo-finishing styles when rendering RAW images. We use four picture styles—Adobe Standard, Adobe Landscape, Adobe Color, and Adobe Vivid—to generate ProPhoto images with vivid colors. These are saved in their original full resolution size, ranging from 2000×3000 to 4000×6000.

From these rendered images, we chose 2000 images for the training set and 200 for the testing set. Images were selected such that they have at least 10% out-of-gamut pixels. Figure 5 shows a sample of the images from our dataset and a plot of the percentage of out-of-gamut pixels in each image in the dataset. The minimum number of out-of-gamut pixels for an image in our testing set is over a million pixels. The final breakdown of images selected from the starting datasets is: 11% Adobe FiveK, 62% RAISE, 12% Cube+, and 15% NUS.

4.2. Comparisons

The following describes methods used for comparisons against the GamutMLP approach.

Conventional methods The two baseline methods are clipping and soft-clipping [20], described in Section 2. Clipping is currently the de facto method for gamut reduction. While soft-clipping aids in gamut expansion, it is not commonly used because it distorts the colors in the sRGB.

Pre-trained deep networks As described in Section 2, GamutNet [16] is a DNN-based method targeting gamut recovery. For the sake of completeness, we also compare against several image-to-image-translation methods: pix2pix [11], pix2pixHD [36], and ASAPNet [31]. We can consider our problem of clipped-ProPhoto to ProPhoto conversion as a special application for image-to-image translation methods. For all methods that need to be pre-trained, we train them with 512×512 crops from training images (clipped ProPhoto and ProPhoto pairs). Cropping is performed such that at least 10% of the cropped image has out-of-gamut pixels. At inference time, the DNN-based methods are applied to the full-sized testing images.

Per-image optimization We also compare with [15], another metadata approach that uniformly samples pixels (135KB) of the original ProPhoto image for recovery using polynomial color correction functions. We also compare with several variants of the SIREN [32] coordinate-based neural implicit function. In particular, we start with the original SIREN, which uses 2D coordinates for input, and has five fully connected linear layers, each with 256 channels and periodic activation functions; the last linear output layer has only three channels for RGB values. The SIREN model requires 796 KB. We optimize a SIREN-residual variant that predicts the residual between the clipped and ground-truth ProPhoto images. Finally, we try a variant of SIREN-residual that is limited to a small model size (69 KB) to mimic a smaller model. The SIREN models are optimized based on the loss $\mathcal{L}_{gamut}$ described in Equation 5. For the other hyperparameters, we adopt default settings for image-fitting tasks from SIREN [32]. Unfortunately, we could not use the fast implementation of MLP [9, 22] for the original SIREN and variants since the fast implementation API does not support the sinusoidal activation function used by SIREN.

For our GamutMLP approach, we show the results of our MLP variants with encoded inputs (i.e., Equation 4), and with and without optimization plus the meta-GamutMLP initialization. The pre-trained DNNs methods and MLP-based approaches are trained or optimized using a NVIDIA Quadro RTX 6000.

4.3. Quantitative results

Table 1 summarizes the performance of the tested methods. Results are reported as the average root mean square error (RMSE) and peak signal-to-noise ratio (PSNR) between the predicted and ground truth test images. Metrics

Method	Metadata↓	RMSE↓	RMSE OG↓	PSNR↑	PSNR OG↑	Optim. Time↓
Conventional						
Clip	-	0.0069	0.0126	43.22	37.98	-
Soft Clip	-	0.0039	0.0042	48.17	47.54	-
Pre-trained DNN						
Pix2pix [11]	-	0.0087	0.0167	41.24	35.55	-
Pix2pixHD [36]	-	0.0157	0.0314	36.08	30.07	-
ASAPNet [31]	-	0.0518	0.0993	25.72	20.06	-
GamutNet [16]	-	0.0052	0.0088	45.75	41.08	-
Optimized per image						
ProPhoto-Sampled [15]	135 KB	0.0032	0.0051	49.78	45.90	-
SIREN [32]	796 KB	0.0648	0.0421	23.77	27.52	115.67 mins
SIREN-residual	796 KB	0.0033	0.0044	49.72	47.20	118.98 mins
SIREN (small)-residual	69 KB	0.0040	0.0052	47.98	45.66	94.62 mins
MLP + enc. (no optimization)	48 KB	0.0021	0.0031	53.57	50.17	37.05 sec
MLP (53KB) + enc.	53 KB	0.0021	0.0030	53.73	50.33	16.29 sec
MLP (23 KB) + enc.	23 KB	0.0021	0.0031	53.65	50.04	16.32 sec
MLP (53KB) + enc. + meta init.	53 KB	0.0021	0.0032	53.40	50.00	1.94 sec
MLP (23 KB) + enc. + meta init.	23 KB	0.0021	0.0032	53.36	49.93	1.90 sec

Table 1. This table shows results on various methods used for wide-gamut color recovery. The reported numbers are the average results computed against the 200 16-bit ProPhoto ground-truth full-size images. RMSE and PSNR are provided for the whole image and out-of-gamut (OG) pixels. For the per-image methods, we provide associated metadata (size in KB) and optimization time. For pre-trained DNNs, we compare with Pix2pix [11], Pix2PixHD [36], ASAPNet [31], and GamutNet [16]. For the per-image methods, we compare with ProPhoto-Sampled [15], SIREN [32] variants, and our GamutMLP variants: MLP (no optimization), sizes (53 KB) and (23 KB), and with "meta" initialization.

are provided for the entire image and for only out-of-gamut (OG) pixels. We provide the associated metadata size (KB) and optimization times for the relevant methods.

The table reveals that simple image-to-image translation does not work well for this task. Pix2pix [11] gives better results than Pix2pixHD [36], since Pix2pix is a general-purpose method for image-to-image translation, while Pix2pixHD was designed to synthesize images from semantic label maps. The DNN-based GamutNet [16] method explicitly designed for gamut recovery performs better than the clipping baseline. The best results are obtained from the per-image optimized methods. In particular, the full-sized SIREN-residual MLP model produces results similar to ours. However, the model requires 796KB and is slow to optimize even compared with our equivalent standard optimized MLP. Our GamutMLP with feature encoding and pre-trained initialization gives the best results and the fastest optimization performance. Note that the ProPhoto-sampled method [15] is optimized per image; however, the computationally intensive component is during the gamut expansion phase instead of the reduction phase. In addition, the method's implementation uses Matlab and runs on CPU, so we do not include its running time since the comparison is unfair.

4.4. Qualitative results

We show qualitative visual outputs of selected approaches in Figure 6. See supplemental materials for additional results. For each method, the first row shows the predicted ProPhoto image. The second row shows an error map of the RMSE of each pixel between the predicted and original ProPhoto. The third row shows the out-of-gamut colors after restoration on a CIE-xy chromaticity diagram. Similar to the quantitative results, we see our approach achieves better results in terms of RMSE and PSNR. In addition, the CIE-xy chromaticity diagram shows the recovered colors appear more like the ground truth wide-gamut images.

While not the goal of our work, on careful observation, it can be noticed that our method and the ProPhoto-sampled method [15] provide a slight improvement in the in-gamut pixels. This is attributed to a slight recovery of some of the 16-bit values that were quantized when the image was saved in 8-bit sRGB in the gamut reduction step. Recall that this method includes in-gamut pixels in the optimization and is applied by the GamutMLP to all pixels. Similarly, the work [15] uniformly samples ProPhoto pixels, including in-gamut and out-of-gamut pixels. The most significant improvements, however, are still with the clipped out-of-gamut pixels, which incur the most error in the gamut reduction step.

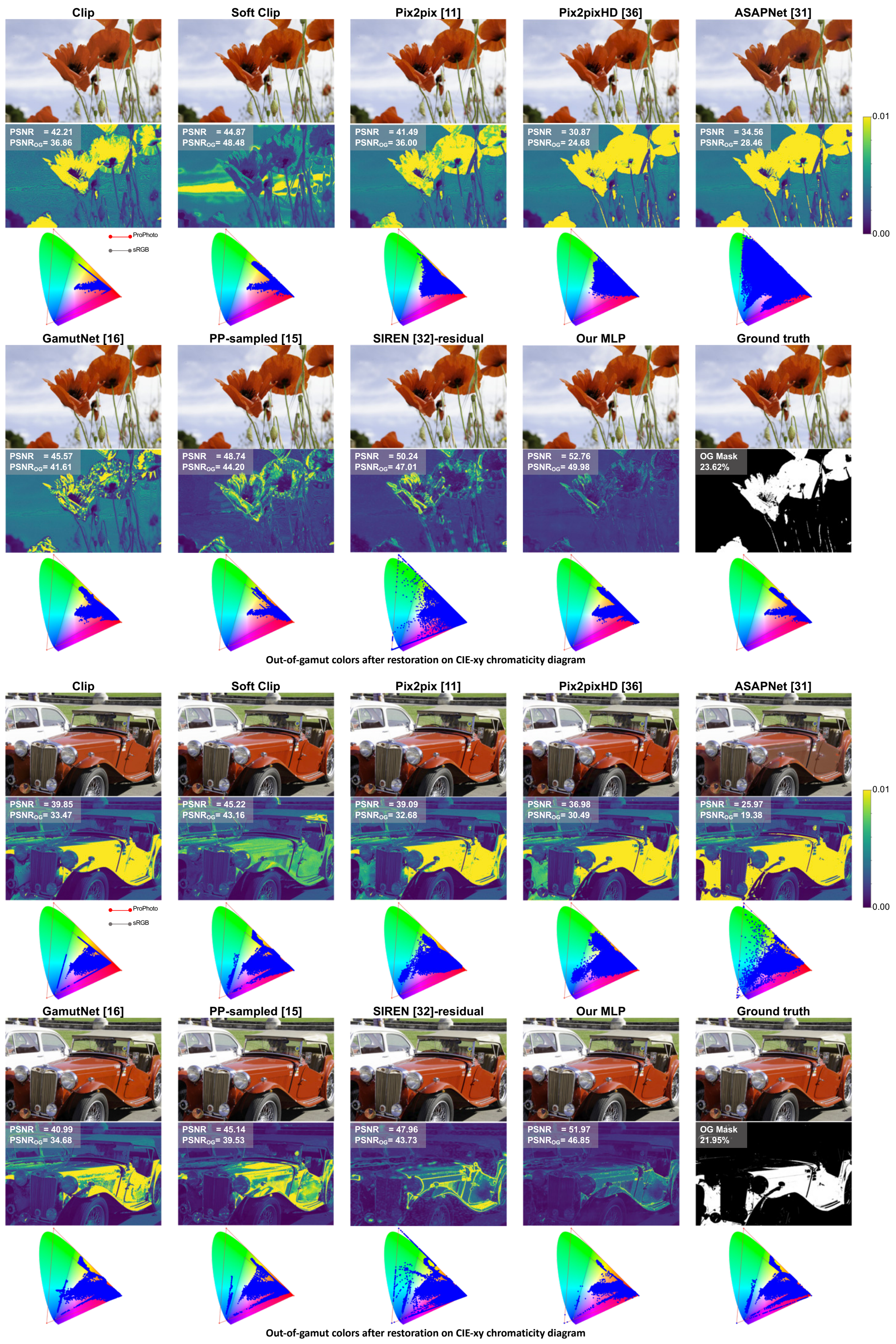


Figure 6. Qualitative comparisons between the predicted ProPhoto full-size output of Clip, Soft Clip, Pix2pix [11], Pix2PixHD [36], ASAPNet [31], GamutNet [16], PP-sampled [15], SIREN [32]-residual, and our optimized GamutMLP. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.

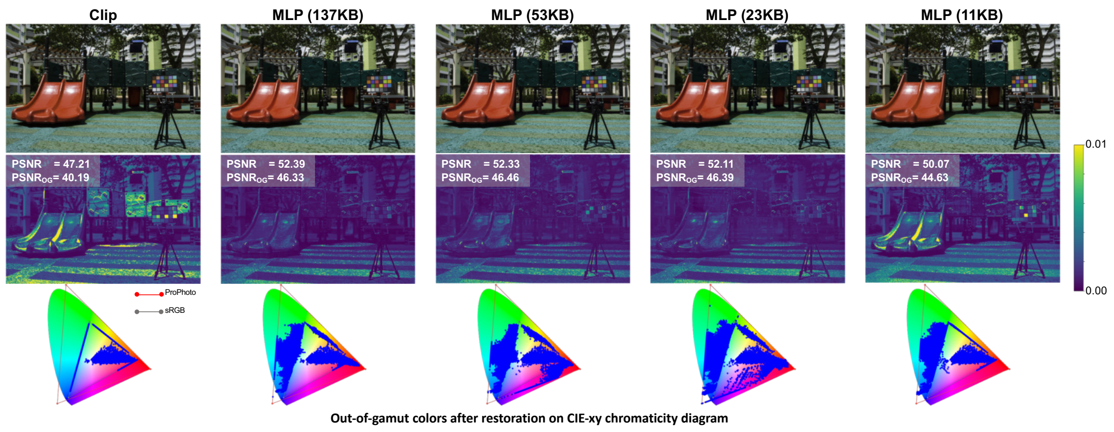


Figure 7. Qualitative comparisons between the predicted ProPhoto full-size output of Clip and our various versions of MLP (137 KB, 53 KB, 23 KB, and 11 KB).

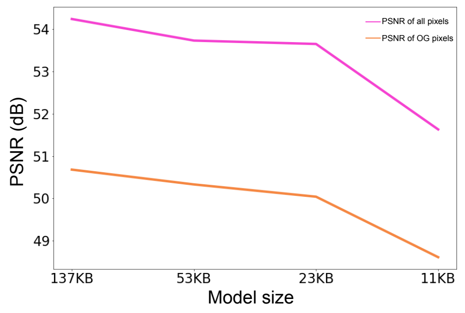


Figure 8. Ablation examining MLP model sizes. The 23 KB MLP provided good results for model size.

4.5. Ablations

Input coordinates We trained our proposed MLP with various input types: only coordinates (x,y), only color values (R,G,B), and our 5D vector (x,y,R,G,B) in Table 2. The use of the coordinate and color values provides excellent results with a small network.

Method	RMSE↓	RMSE OG↓	PSNR↑	PSNR OG↑
MLP (23 KB) [xy]	0.0037	0.0048	48.57	46.38
MLP (23 KB) [RGB]	0.0028	0.0045	51.10	46.98
MLP (23 KB) [xyRGB]	0.0021	0.0031	53.65	50.04

Table 2. This table shows the results of our variant MLPs with various types of inputs: *xy*, *RGB*, *xyRGB*. RMSE and PSNR for the whole image as well as OG pixels are computed and averaged over 200 test images.

GamutMLP model sizes Our goal was to find a model size that was compact and had a reasonable optimization time. In particular, we tried to vary the hidden features from 128, 64, 32 to 16, and their corresponding model sizes are 137 KB, 53 KB, 23 KB, and 11 KB. Figure 8 shows a plot of the average PSNR computed over all these 200 test images for each model. Results are shown for the entire image and out-of-gamut pixels only. Figure 7 shows qualitative results on a test image for the different model sizes. While the larger model gave a slightly better performance, the 23 KB model provided a comparable result, with memory size that can easily be included as a comment field in an sRGB image. The model below 23 KB performed poorly. See supplemental materials for additional ablations.

5. Discussion and concluding remarks

We have presented a framework to recover wide-gamut color values lost due to the gamut reduction step applied when converting a ProPhoto image to a sRGB image. We cast our task as a restoration problem with the goal of restoring the original color loss due to gamut clipping. By integrating our approach into the gamut reduction step when the image is converted to sRGB, we have the opportunity to optimize a model directly against the known restoration target—a luxury most restoration problems such as deblurring and image sampling do not have. This allowed us to use a lightweight MLP network to predict the original color signal. Compared to the overall sRGB image size (typically 2–5 MB), the 23 KB memory overhead for the GamutMLP model is negligible and means that the color fidelity recovery is obtained virtually for free.

Our experiments show that per-image MLP optimization provides much better results than pre-trained DNN models for color recovery. Furthermore, our small GamutMLP provides comparable performance in PSNR compared with larger MLP-based neural implicit functions but requires significantly less memory size and optimization time (within 2 seconds). As part of this effort, we have also generated a new dataset of 2200 images with high-color fidelity that will be useful in advancing research in this area. Our code and dataset can be found on the project website: https://gamut-mlp.github.io.

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

References

[1] Arne Bakke, Ivar Farup, and Jon Hardeberg. Evaluation of algorithms for the determination of color gamut boundaries. *Journal of Imaging Science and Technology*, 54(5):50502–1, 2010. 2

[2] Nikola Banic and Sven Loncaric. Unsupervised learning for color constancy. *CoRR*, 2017. 5

[3] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011. 5

[4] Ayan Chakrabarti, Daniel Scharstein, and Todd E. Zickler. An empirical camera model for internet color vision. In *BMVC*, 2009. 2

[5] Ayan Chakrabarti, Ying Xiong, Baochen Sun, Trevor Darrell, Daniel Scharstein, Todd Zickler, and Kate Saenko. Modeling radiometric uncertainty for vision with tone-mapped color images. *IEEE TPAMI*, 36(11):2185–2198, 2014. 2

[6] Dongliang Cheng, Dilip K. Prasad, and Michael S. Brown. Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution. *JOSA-(A)*, 31(5):1049–1058, 2014. 5

[7] International Electrotechnical Commission. IEC 61966-2-1 default RGB colour space – sRGB. *IEC*, 1999. 1, 3

[8] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. RAISE: A raw image dataset for digital image forensics. In *ACM Multimedia*, 2015. 5

[9] Thomas Müller et al. Instant neural graphics primitives with a multiresolution hash encoding. *ACM ToG*, July 2022. 4, 5

[10] Han Gong, Graham D. Finlayson, Maryam M. Darrodi, and Robert B. Fisher. Rank-based radiometric calibration. In *Color Imaging Conference*, 2018. 2

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 5, 6, 7

[12] Hakki Can Karaimer and Michael S. Brown. A software platform for manipulating the camera imaging pipeline. In *ECCV*, 2016. 1

[13] Seon Joo Kim, Hai Ting Lin, Zheng Lu, Sabine Süsstrunk, Stephen Lin, and Michael S. Brown. A new in-camera imaging model for color computer vision and its application. *IEEE TPAMI*, 34(12):2289–2302, 2012. 2

[14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 4

[15] Hoang Le, Mahmoud Afifi, and Michael S Brown. Improving color space conversion for camera-captured images via wide-gamut metadata. *Color and Imaging Conference*, 2020. 3, 5, 6, 7

[16] Hoang Le, Taehong Jeong, Abdelrahman Abdelhamed, Hyun Joon Shin, and Michael S Brown. GamutNet: Restoring wide-gamut colors for camera-captured images. *Color and Imaging Conference*, 2021. 2, 5, 6, 7

[17] Yu-Lun Liu, Wei-Sheng Lai, Yu Sheng Chen, Yi-Lung Kao, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Single-image HDR reconstruction by learning to reverse the camera pipeline. In *CVPR*, 2020. 2

[18] John J McCann. Color gamut mapping using spatial comparisons. In *Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts VI*, 2000. 2

[19] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3, 4

[20] Ján Morovič. *Color gamut mapping*. John Wiley & Sons, 2008. 2, 5

[21] J. Morovic and M. R. Luo. Evaluating gamut mapping algorithms for universal applicability. *Color Research & Application*, 26:85 – 102, 12 2000. 2

[22] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Trans. Graph.*, 40(4):36:1–36:16, Aug. 2021. 4, 5

[23] Seonghyeon Nam and Seon Joo Kim. Modelling the scene dependent imaging in cameras with a deep neural network. In *ICCV*, 2017. 2

[24] Seonghyeon Nam, Abhijith Punnappurath, Marcus A. Brubaker, and Michael S. Brown. Learning srgb-to-raw-rgb de-rendering with content-aware metadata. In *CVPR*, 2022. 3

[25] Rang M.H. Nguyen and Michael S. Brown. Raw image reconstruction using a self-contained sRGB–JPEG image with small memory overhead. *IJCV*, 126(6):637–650, 2018. 2

[26] Rang M. H. Nguyen and Michael S. Brown. RAW image reconstruction using a self-contained sRGB-JPEG image with only 64 KB overhead. In *CVPR*, 2016. 2

[27] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, 2018. 4

[28] Jens Preiss, Mark D. Fairchild, James A. Ferwerda, and Philipp Urban. Gamut mapping in a high-dynamic-range color space. In *Color Imaging XIX: Displaying, Processing, Hardcopy, and Applications*, 2014. 2

[29] Abhijith Punnappurath and Michael S. Brown. Spatially aware metadata for raw reconstruction. In *WACV*, 2021. 3

[30] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, 2019. 4

[31] Tamar Rott Shaham, Michaël Gharbi, Richard Zhang, Eli Shechtman, and Tomer Michaeli. Spatially-adaptive pixelwise networks for fast image translation. In *CVPR*, 2021. 5, 6, 7

[32] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. 3, 5, 6, 7

[33] Kevin E. Spaulding, Geoffrey J. Woolfe, and Edward J. Giorgianni. Reference input/output medium metric rgb color encodings. In *Color Imaging Conference*, 2000. 1

[34] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021. 4

[35] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020. 3

[36] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 5, 6, 7