To appear in The 38th Annual AAAI Conference on Artificial Intelligence (AAAI-24)

NILUT: Conditional Neural Implicit 3D Lookup Tables for Image Enhancement

Marcos V. Conde[1], Javier Vazquez-Corral[2,3], Michael S. Brown[4], Radu Timofte[1]
[1] Computer Vision Lab, CAIDAS, University of Würzburg [2] Computer Vision Center (CVC)
[3] Department of Computer Science, Universitat Autònoma de Barcelona [4] York University

Figure 1: NILUTs encode multiple 3D LUTs in a single representation with the ability to blend between "styles" implicitely.

Abstract

3D lookup tables (3D LUTs) are a key component for image enhancement. Modern image signal processors (ISPs) have dedicated support for these as part of the camera rendering pipeline. Cameras typically provide multiple options for picture styles, where each style is usually obtained by applying a unique handcrafted 3D LUT. Current approaches for learning and applying 3D LUTs are notably fast, yet not so memory-efficient, as storing multiple 3D LUTs is required. For this reason and other implementation limitations, their use on mobile devices is less popular.

In this work, we propose a Neural Implicit LUT (NILUT), an implicitly defined continuous 3D color transformation parameterized by a neural network. We show that NILUTs are capable of accurately emulating real 3D LUTs. Moreover, a NILUT can be extended to incorporate multiple styles into a single network with the ability to blend styles implicitly. Our novel approach is memory-efficient, controllable and can complement previous methods, including learned ISPs. Code, models and dataset available at: `https://github.com/mv-lab/nilut`.

1. Introduction

Image signal processors (ISP) are hardware units used in cameras to process the RAW sensor images to the final output image [18, 24, 39]. The ISP hardware applies a series of processing steps to render the RAW image to its final photo-finished output. 3D lookup tables (**3D LUTs**) are one of the core components used in conventional ISPs. Specifically, a 3D LUT is a global color operator that maps an RGB color to a new RGB color. 3D LUTs are commonly used to model a desired stylistic look as shown in Fig. 1. Most cameras can render the same image to several different pictures, where each picture style has its own associated 3D LUT to enhance the color and tone [13, 25, 50, 53].

Modern cameras use conventional ISPs [11, 13, 24] and apply further photo-finishing deep-learning models [23, 44]. These usually run on dedicated processors *e.g.* neural processing units (NPUs), with important limitations such as memory allocation and allowed operations [21]. In addition, there is already an active trend to replace many conventional ISP components with deep learning-based algorithms, *e.g.*, image denoising [8], color constancy [19], super-resolution [9, 22], and even the entire ISP [28, 31, 32].

In particular, methods for image enhancement via color and tone manipulation are often based on 3D LUTs [48, 50] due to their runtime efficiency. However, many of these methods are not suitable for mobile devices due to their memory limitations (*i.e.* storing multiple 3D LUTs would be too memory exhaustive) and required operations.

Contribution We propose **NILUT**, a novel application of implicit neural representations (**INRs**) [41] for color manipulation. Our NILUT is an implicitly defined, continuous 3D color transformation parameterized by a neural network. NILUTs can accurately mimic existing professional 3D LUTs, as shown in Fig. 2. Moreover, NILUTs can be extended to encode multiple styles into a single network. During inference, the NILUT can be conditioned on a particular picture style, and even blend between multiple styles implicitly. This novel multi-style formulation allows controllable image enhancement and customization. We believe NILUTs can complement previous methods for image enhancement [46, 48, 50, 53], including learned ISPs [20, 23] designed for smartphones. As part of this effort, we provide a dataset of curated 3D LUTs and images for evaluation.

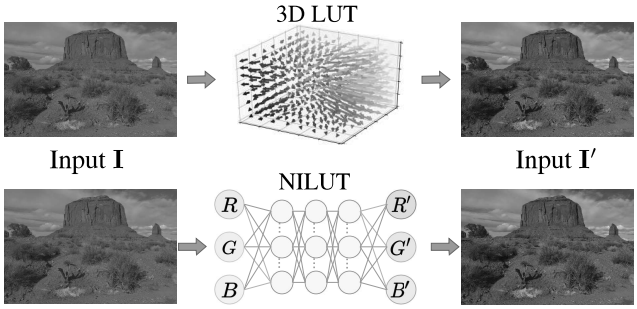


Figure 2: **Top:** shows a conventional 3D lookup table able to enhance the color and tone of the image. **Bottom:** shows the same functionality based on the proposed NILUT.

2. Related Work

2.1. 3D LUTs for Color Manipulation

3D LUTs are a mechanism to approximate a nonlinear 3D color transform by sparsely sampling the transformation by a discrete 3D lattice [24, 29, 50]. The model is defined as a mapping ϕ, usually in the RGB color space, where an input color $\mathbf{I} = [r, g, b]$ is mapped into $\mathbf{I}' = [r', g', b']$:

$$\phi : \mathbb{R}^3 \mapsto \mathbb{R}^3 \quad \phi(\mathbf{I}) = \mathbf{I}'. \tag{1}$$

3D LUTs appear at different stages of the image signal processor pipeline, as detailed in [24, 25]. 3D LUTs are often manually created by camera engineers and professional photographers. Methods such as lattice regression [14, 29] parameterize the 3D LUT's lattice from the sparse samples using various regularizations.

More recently, deep learning techniques have been employed for estimating 3D LUTs and learning interpolation and sampling strategies within the LUT with the goal of color image enhancement [30, 46, 49]. For example, Zeng *et al*. [50] proposed a method that learned the parameters of three 3D LUTs and an additional per-image adaption to blend between the LUTs. Similarly, Wang *et al*. [46] proposed a similar idea to learn the parameters of 3D LUTs, but included a spatial blending over the image, effectively implementing a spatially varying 3D LUT. Yang *et al*. AdaInt [48] focused on improving the sampling within the 3D LUTs (based on [50]). They proposed a learned method to improve the classical trilinear interpolation between uniform sampling points in the LUT [29]. Also, Yang *et al*. [49] proposed to learn separated component-correlated sub-transforms as 1D and 3D LUTs.

The aforementioned methods focus on conventional 3D LUTs and their interpolation mechanisms. In contrast, our NILUT is interested in providing the functionality of the 3D LUTs, but as a neural network to be more compatible with NPU-based hardware. In addition, we are interested in encoding multiple picture styles within the same network.

2.2. ISPs and Image Enhancement

In recent years, most low-level computer vision tasks have witnessed promising results from deep learning methods. For example, significant advances in image denoising [8, 51], image deblurring [38], image super-resolution [9, 27], and image enhancement [12, 16, 17, 33, 36, 45, 47, 52, 53] amongst many other tasks. Most of these tasks are crucial stages in modern smartphone ISPs.

Moreover, recent end-to-end learned ISPs [23, 28, 31, 32] have also obtained promising results. Due to this current deep learning trend, smartphone manufacturers are incorporating special neural processing units (NPUs) [2–7, 21].

This said, the previously introduced methods are designed to run on consumer GPUs (*e.g*. NVIDIA V100), and therefore integrating such powerful tools into a smartphone is extremely challenging, and sometimes impossible due to the memory and computational limitations [21].

Our NILUTs represent a new plug-and-play module for modern deep learning-based ISPs and image enhancement pipelines such as Ignatov *et al*. real-time image super-resolution [22] and end-to-end learned ISPs [20] tested in real commercial smartphones NPUs.

2.3. Implicit Neural Representations

In recent years, implicit neural representations (INRs) [15, 37, 41] have become increasingly popular in image processing as a novel way to parameterize an image. Also known as coordinate-based networks, these approaches use multilayer perceptrons (MLPs) to overfit to the image. Multiple works have demonstrated the potential of MLPs as continuous, memory-efficient implicit representations for images [41, 42]; we find especially inspiring SIREN [41] and Fourier Feature Networks [43]. This technique was also successfully applied to model shapes [15, 34] and 3D scenes [35, 37].

Conventional signal representations are usually discrete *e.g*., an image is a discrete grid of pixels (subspace of $\mathbb{R}^2$) with output values bounded in an $\mathbb{R}^3$ RGB space. In contrast, INRs parameterize a signal as a continuous function that maps the source domain $\mathcal{S}$ of the signal (*i.e*., a coordinate) to its corresponding value in the target $\mathcal{T}$ (*i.e*., the corresponding RGB intensity value). This function is approximated using a neural network, and therefore it is not analytically tractable. This can be formulated as:

$$\Phi : \mathbb{R}^2 \mapsto \mathbb{R}^3 \quad \mathbf{x} \rightarrow \Phi(\mathbf{x}) = [r, g, b], \tag{2}$$

where Φ is the learned INR function, the domains $\mathcal{S} \in \mathbb{R}^2$ and $\mathcal{T} \in \mathbb{R}^3$, the input coordinates $\mathbf{x}$, and the output RGB value $[r, g, b]$. Note that the behavior of this function is similar to a lookup table—see (1)— yet being continuous, differentiable, and learnable. Our new application of INRs consists of learning a mapping between two color representations.

3. Neural Implicit LUT

As discussed in the prior section, the most common INRs in the literature [41, 43] are coordinate-based representations of image signals, implemented using MLPs. In this work, our goal is to learn a 3D transformation in the RGB color space, therefore we map $\mathbb{R}^3$ coordinates (*i.e.* color values $\mathbf{I}$) from a source $\mathcal{S}$ to a target $\mathcal{T}$, being both 3D domains.

Specifically, we want a continuous function Φ with the following properties:

$$\Phi : \mathbb{R}^3 \mapsto \mathbb{R}^3 \quad \Phi(\mathbf{I}) \in [0,1]^3, \tag{3}$$

$$\Phi(\mathbf{I}) \approx \phi(\mathbf{I}), \tag{4}$$

$$\nabla\Phi \approx \nabla\phi. \tag{5}$$

We represent the RGB space as a set $\mathcal{X} = \{x_i\}$ of color pixels $x_i = (r_i, g_i, b_i)$. This set contains $\approx$ 16 million elements if we consider the complete RGB space (*i.e.* 256^3). To learn our continual representations Φ we minimize:

$$\mathcal{L} = \sum_i \|\Phi(x_i) - \phi(x_i)\|_1, \tag{6}$$

where ϕ is the real 3D LUT -see Eq. 1-.

This function Φ is an implicit neural representation of a 3D lookup table ϕ and can be formulated as:

$$\begin{aligned} \Phi(x) = \mathbf{W}_n(\varsigma_{n-1} \circ \varsigma_{n-2} \circ \ldots \circ \varsigma_0)(x) + \mathbf{b}_n \\ \varsigma_i(x_i) = \alpha\left(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i\right), \end{aligned} \tag{7}$$

where ς_i are the layers of the network (considering their corresponding weight matrix $\mathbf{W}$ and bias $\mathbf{b}$), and α is a nonlinear activation. We study three different networks: (i) simple ReLU-MLPs or Tanh-MLPs [41], (ii) SIREN [41] and (iii) Residual MLPs (referred as MLP-Res).

We note here that SIREN [41] has some drawbacks in terms of its implementation. First, it can not use INRs speed-up techniques, such as the one in [37], and second, their custom activations are still not supported for mobile devices accelerator hardware.

Another alternative to SIREN would be a residual-based MLP (MLP-Res). This approach assumes the 3D LUT does not make drastic color changes (e.g., red becoming blue), but instead performs color manipulation via reasonable displacements (residual) between input and output RGB values. Our visualizations of 3D LUTs in Fig. 3 help to reveal that this is often the case. This approach also serves to help regularize the MLP in regions with no changes (i.e., the residual is 0 over the three changes). Our ablations in Section 4 also reveal this is a good strategy.

Conditional Neural LUT We can further improve the proposed representation to model more complex relationships in the image color domain:

$$\Psi : \mathbb{R}^{3+m} \mapsto \mathbb{R}^3 \quad \Psi(\mathbf{z}) = \mathbf{I}', \tag{8}$$

where $\mathbf{z} = [\mathbf{I}, \mathbf{c}]$ is the concatenation of the input RGB intensity $\mathbf{I} \in \mathbb{R}^3$, and a condition vector $\mathbf{c} \in \mathbb{R}^m$ with m possible styles or LUTs using one-hot class encoding. Therefore this continual function Ψ maps an input intensity $\mathbf{I}$ into $\mathbf{I}'$ under the condition $\mathbf{c}$, representing a conditional neural implicit LUT, a more general and powerful representation than the previously introduced NILUT (Φ). We illustrate this in Fig. 3, where we show the codification of the style as a condition vector using one-hot encoding. The derivation of our CNILUT also allows us to blend among different styles by just modifying the values of the condition vector. Details of this are provided in Section 4.1.

3.1. Discussion

We end this section by discussing some benefits and differences between NILUTs over conventional 3D LUTs.

Firstly, previous approaches for learning 3D LUTs [14, 48, 50] are notably fast and faithful, allowing real-time processing using regular GPUs (*e.g.* 24Gb memory). We understand that a neural network (*e.g.* NILUT) is limited and cannot surpass the efficiency of a lookup operation. However, in the mobile devices scenario, 3D LUTs have two important limitations: **(i)** allocating in memory multiple 3D LUTs (usually 33-dim, thus $107K$ FP parameters [50]) is not possible due to the hard memory limitations found in mobile devices chips *e.g.* NPUs [21]. Note that dedicated processors such as the ISP usually have their own memory and typically only compact models can run on these [20, 22]. **(ii)** many operations, including Zeng *et al.* trilinear interpolation on CUDA [50] are not supported by PyTorch Mobile or TFLite, the most common frameworks for developing efficient mobile models. We believe these are the reasons why there are very few works about image enhancement on mobile devices using 3D LUTs. Thus, we aim to complement previous approaches and offer a memory-efficient alternative for mobile devices.

Secondly, NILUTs offer other benefits such as being naturally differentiable, allowing end-to-end learning, and are mobile-ready allowing to complement deep learning-based image processing pipelines as a plug-and-play module, in mobile devices. Finally, NILUTs, as a novel representation, are conditional, allowing a single compact neural network to deal with multiple styles or LUTS that are presented in the imaging process. This clearly contrasts with current LUTS, in which a different process should be run for each specific one. This is key to being memory-efficient as one NILUT can mimic the transformation of five real 3D LUTs. Also, this property allows us to blend among the different styles at inference time in contrast with current LUTs where each blending is individually computed either at the 3D LUT or at the image level (*e.g.* as in [50]).

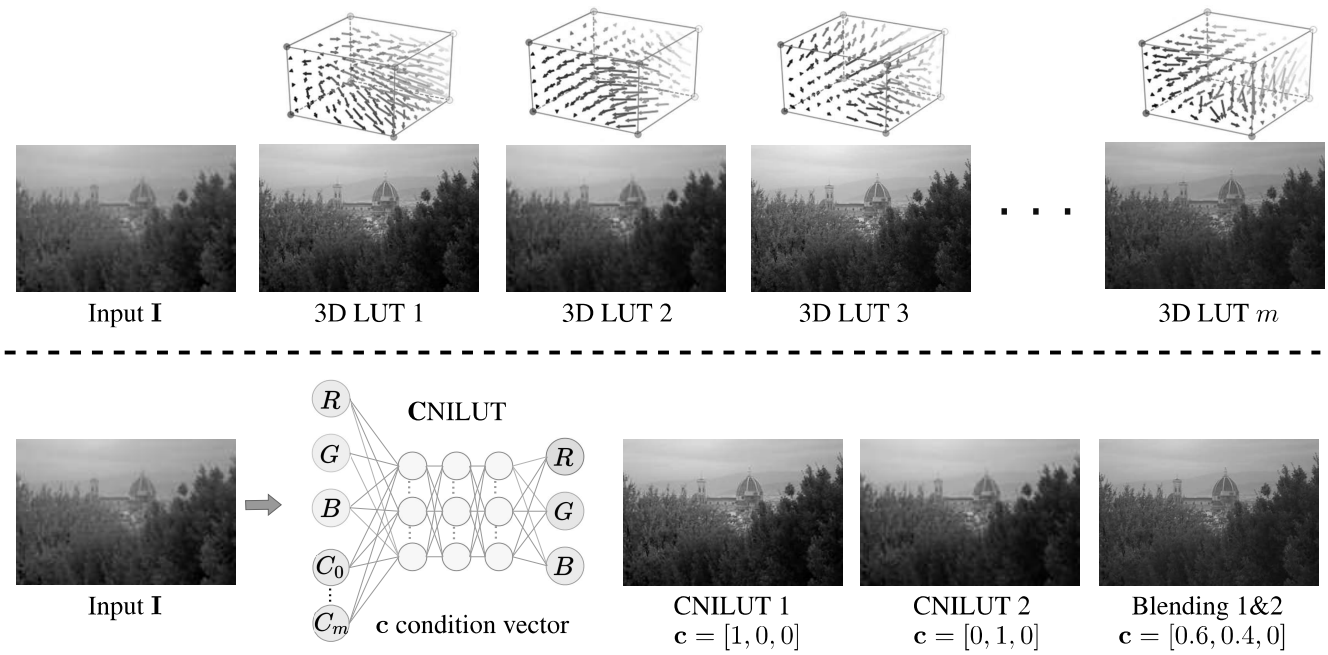


Figure 3: **Top:** A conventional 3D LUT framework. Each 3D LUT is stored and processed individually. **Bottom:** The new framework introduced by CNILUTs. We use as input both the image and a condition vector (one-hot encoding of the style), allowing for **i)** selection of multiple styles with a single network, and **ii)** blending different 3D LUTs styles by modifying the input condition vector. This happens implicitly without additional computational cost.

4. Experiments

Learning a complete 3D LUT transformation of the 8-bit RGB space requires $256^3 = 16.78$ million input (and output) colors. According to Eq. (6), we represent these $16M$ colors of the RGB space as a set $\mathcal{X} = \{x_i\}$ of size $16M \times 3$. This set is equivalent to an image of dimension $4096 \times 4096 \times 3$ that we call $\mathcal{M}$. We will denote this image as the RGB map, illustrated in Fig. 4. We then process image $\mathcal{M}$ using professional image editing software (Adobe Photoshop) and real 3D LUTs designed by photographers. These processed images are reshaped back to dimension $16M \times 3$, and correspond to $\phi(x_i)$ on Eq. (6), *i.e.* our ground-truth for the minimization.

The coordinate-based MLP Φ, as it is the standard practice with INRs [41], is trained to "overfit" the mapping between its result to the input colors ($\Phi(x_i)$) and the output of the real LUT $\phi(x_i)$, as previously introduced in Section 2.3. We provide visualizations of this training for a subset of the colors in $\mathcal{M}$ in the supplementary material.

Note that using this setup we do not require natural images to learn real 3D LUTs, just the corresponding RGB maps (Halds). This is indeed the way professional photographers create 3D LUTs [1].

Evaluation We consider two different metrics for our work. We report PSNR, and CIELAB ΔE error. We choose these two measures because i) PSNR is a standard fidelity metric in the literature and ii) ΔE is a perceptual color difference metric that measures differences between two colors [40], and therefore well suited for our problem. Considering that we fit our NILUTs using RGB maps as mentioned before, we evaluate the quality of our NILUT representation in two different ways: (i) RGB mapping quality (Fig. 4), (ii) using natural unseen images from MIT5K [10].

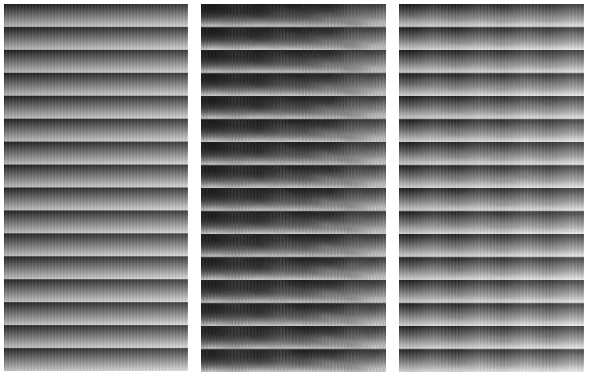


Figure 4: From left to right, original RGB map, output of 3D LUT "Cyberpunk", output of 3D LUT style "Nightcolors". We can appreciate the color transformation clearly. In graphics, this is referred to as a **Hald** image, a graphical representation of 3D LUT in the form of a color table that contains all of the color gradations of 3D LUT [1].

Method	N	L	$\text{PSNR}_{\text{rgb}} \uparrow$	$\Delta\text{E}_{\text{rgb}} \downarrow$	$\text{PSNR}_{\text{5k}} \uparrow$	$\Delta\text{E}_{\text{5k}} \downarrow$
SIREN [41]	128	2	44.43	1.04	<u>41.17</u>	**1.63**
SIREN [41]	64	2	**45.37**	**0.96**	40.35	1.82
MLP-Res	128	2	<u>45.34</u>	<u>0.97</u>	**42.04**	<u>1.65</u>
MLP-Res	64	2	43.84	1.11	40.34	1.93
MLP	128	2	43.18	1.19	39.70	2.20
MLP	64	2	41.41	1.41	38.34	2.36

Table 1: Evaluation of different NILUT architectures on both the RGB map image and on the MIT5K dataset [10]. We refer to the number of neurons as (N), number of layers as (L). Compact SIREN [41] and MLP-Res architectures can model complex 3D LUTs and approximate their transformations. Best results in bold. Second best underscored.

Our dataset consists on a set of 5 professional 3D LUTs. We report the average metrics over the five.

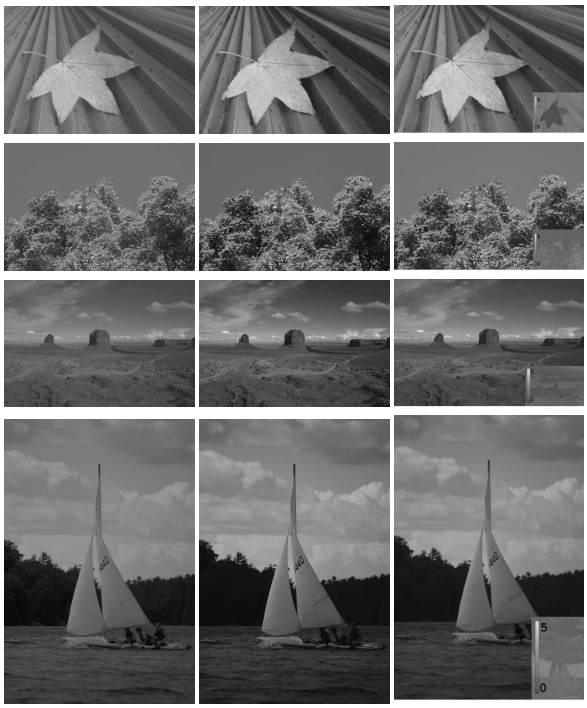
I) We compare the fidelity between the NILUT and real 3D LUT RGB maps. More in detail, we evaluate the differences between the NILUT-generated map $\Phi(\mathcal{M})$ and the real 3D LUT output map $\phi(\mathcal{M})$ -see Eq.(6)-. These results are referred as PSNR_{rgb} and $\Delta\text{E}_{\text{rgb}}$ in Table 1

II) We randomly selected 100 RAW images from the Adobe MIT5K dataset [10], captured using diverse DSLR cameras. We then processed the images using Adobe Photoshop and the same set of 3D LUTs discussed before. Once a NILUT is fitted using the corresponding RGB map, we apply it to this set of images and measure the fidelity between the real 3D LUT and NILUT processed images. These results are referred as PSNR_{5k} and $\Delta\text{E}_{\text{5k}}$ in Table 1.

Table 1 presents our results for different configurations of MLPs -a basic MLP, SIREN, and a Residual MLP-, under two different numbers of neurons (N) and layers (L). Note that differences in ΔE smaller than 2 are indistinguishable by human observers [40]. Also, results with more than 40 dB PSNR are considered of high quality. Attending to Table 1 results, we can affirm that NILUTs can mimic almost perfectly the RGB transformation of real 3D LUTs on natural images from photographers [10].

We also present in Fig. 5 results for the case of MLP-Res $N = 128$ and $L = 2$. We show from left to right, the input image, the ground-truth image processed by real a 3D LUT, and our result. We also display a miniature error map for each of our results. The error map is scaled between 0 and 5 ΔE Units. We provide more results in the supplementary material.

Ablation Studies We studied the results for a larger configuration of MLPs under the first evaluation scenario. Results are presented in Table 2. We can see how for all the cases studied we are able to obtain values that are higher than 40 dB in PSNR and smaller than 1.5 on ΔE, *i.e.* our results have high quality in terms of PSNR and are, on average, indistinguishable from the ground-truth for a human observer. Also, for all the studied architectures we looked at their convergence speed. In Fig. 6 we show for each of the architectures the number of iterations that they required to obtain a result of 40 dB. We can clearly see that the basic MLP needs a larger number of iterations in comparison to SIREN [41] and the Residual-MLP. For the last one, our main architecture MLP-Res, the training is more stable than for SIREN [41], and we can achieve almost perfect mapping in 4 minutes without using special INR acceleration [37].



Input Real 3D LUT Ours NILUT

Figure 5: Results for two different styles (1-2 rows, and 3-4 rows). We can see how the NILUTs are able to reproduce the different styles from professional 3D LUTs. Samples from MIT5K [10]. Note that the NILUT never saw these (or any) real images. Best in electronic version.

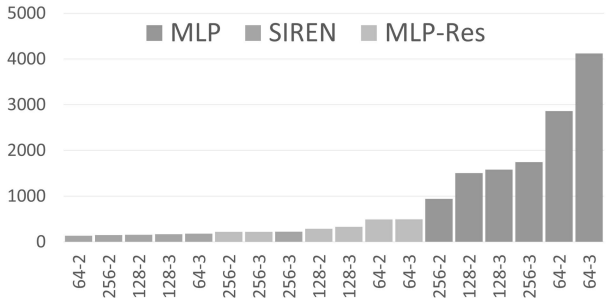


Figure 6: Convergence study. Required number of training steps for each architecture (N neurons, L layers) to achieve over 40dB PSNR at RGB mapping quality -see Fig. 4-.

Figure 7: Results for our conditional NILUTs (CNILUTs). A **single CNILUT** is able to represent three different styles in a single model. All the images in this figure have less than 3 ΔE error with respect to the real LUT. Best viewed in color.

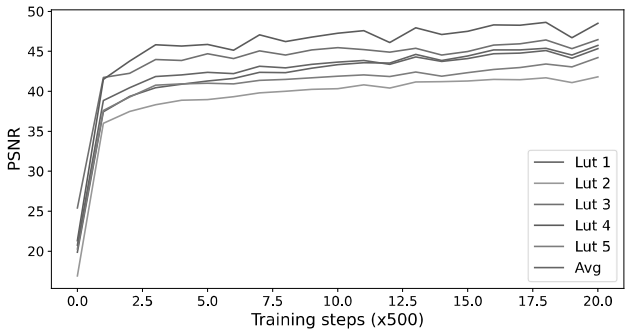


Figure 8: Performance evolution for the CNILUT fitting. Our single CNILUT can accurately learn five different styles from five real 3D LUTs. The reported PSNR is calculated over the five respective reference RGB maps ($\mathcal{M}$). The final average PSNR over the five learned LUTs is 45.34.

4.1. Conditional NILUT

As we introduced in Section 3, our NILUT can be conditioned to different styles (CNILUT), and by doing this, we can learn implicitly multiple 3D LUT transformations using a single NILUT. This novel feature would allow reducing the memory requirements of storing and calling multiple 3D LUTs in a camera pipeline [24]. In our experiments, we set to three and five the number of 3D LUTs to learn using this approach. This however denotes a clear *trade-off* since, in exchange of this ability, CNILUTS require longer and more complex training, and there is a slight performance degradation in comparison to learning three/five separated -and specialized- NILUTs. This said we are able to obtain consistent values larger than 42 dBs in PSNR and errors smaller than 1.5 ΔE in the RGB mapping, therefore being

Method	N	L	# Par. (K)	PSNR ↑	CIELAB ΔE ↓
SIREN [41]	256	2	133.3	41.95	1.45
SIREN [41]	256	3	199.1	43.00	1.33
SIREN [41]	128	2	33.90	44.43	1.04
SIREN [41]	128	3	50.40	42.74	1.43
SIREN [41]	64	2	8.700	45.37	<u>0.96</u>
SIREN [41]	64	3	12.90	43.03	1.35
MLP-Res	256	2	133.3	<u>45.84</u>	1.09
MLP-Res	256	3	199.1	**46.19**	**0.91**
MLP-Res	128	2	33.90	45.34	0.97
MLP-Res	128	3	50.40	45.47	<u>0.96</u>
MLP-Res	64	2	8.700	43.84	1.11
MLP-Res	64	3	12.90	43.55	1.14
MLP	256	2	133.3	44.34	1.00
MLP	256	3	199.1	44.49	1.03
MLP	128	2	33.90	43.18	1.19
MLP	128	3	50.40	42.26	1.34
MLP	64	2	8.700	41.41	1.41
MLP	64	3	12.90	40.86	1.49

Table 2: Ablation study of different MLPs. Results are computed as the differences between the NILUT generated map $\Phi(\mathcal{M})$ and the real 3D LUT map $\phi(\mathcal{M})$. Metrics are the average over the five 3D LUTs in our dataset.

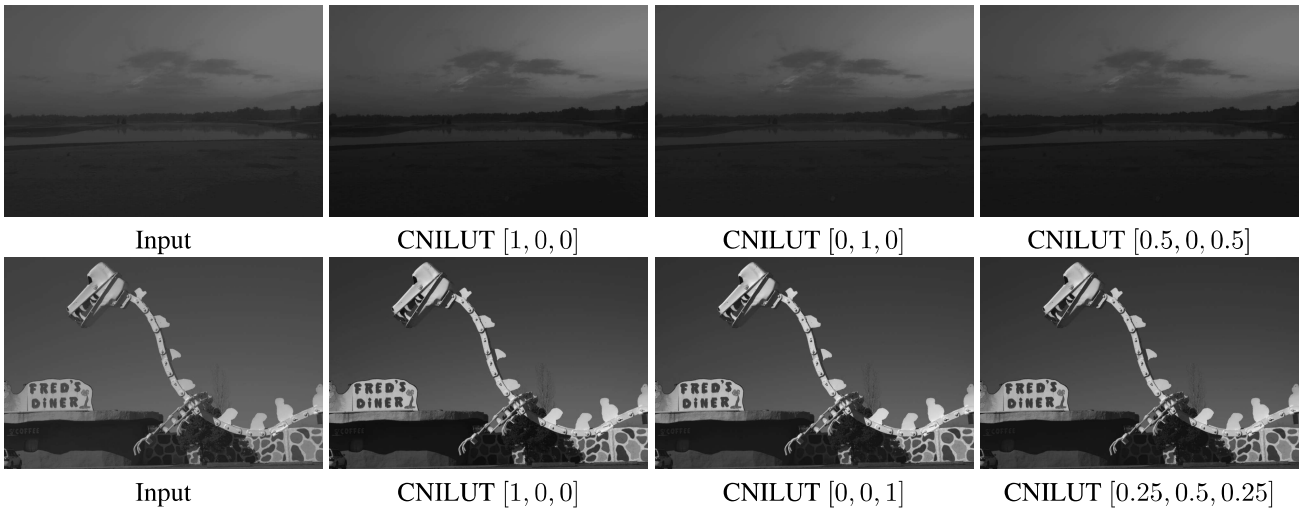


Figure 9: Example of the blending capabilities of the proposed CNILUT. In particular, the proposed MLP-Res 128x2 is trained on a set of three 3D LUTs, which achieves an average RGB mapping quality of 43.72dB. We show the three learned styles (one-hot encoded condition vectors), and two random "blendings" given the specified condition vectors.

almost unnoticeable for human observers. Fig. 8 shows the evolution of the training of the CNILUT using an MLP-Res 256x2. Note that as before, we train using five RGB maps, and use as the main metric -in this case- the average `avg` over the five representations. As we show, some LUTs are easier to learn than others, yet we can learn five LUTs with an average RGB mapping PSNR of 45.34dB, and ΔE 0.85. In Table 3, we provide an ablation study where we show the performance of different MLP-Res architectures when learning 3 and 5 different 3D LUTs. The error is computed as the average for the different 3D LUTs. Given the large PSNR and errors smaller than 1.2 ΔE in the RGB mapping, we can confirm that the difference between the CNILUT mapping and the corresponding three/five 3D LUTs is almost unnoticeable for human observers [40]. The main difference between CNILUT and NILUT is the training: CNILUT requires longer training to obtain good performance. We provide this study in the supplementary material.

Multi-Styles Qualitative Results In Fig. 7 we present some qualitative results of our CNILUT. A single CNILUT can successfully emulate the behavior of three different 3D LUTs by imposing the corresponding condition vectors.

Memory Efficiency From previous experiments, we can conclude that a CNILUT can compress the representation of five 3D LUTs, without additional computational cost *i.e.* the number of operations due to the integration of the condition vector does not affect the runtime. Considering a standard 33-dimensional 3D LUTs [48, 50] with $\approx 107K$ parameters, if stored as FP32 would require ≈ 0.43MB. Our NILUT has 8.7K parameters (≈ 0.032MB) and can accurately reproduce the behavior and properties of such complex 3D LUTs, which implies a compression of $13\times$. This facilitates its storage in mobile devices with limited on-chip memory.

Method	Learn **x3**		Learn **x5**	
	PSNR ↑	ΔE ↓	PSNR ↑	ΔE ↓
128x2	43.72	1.07	42.67	1.19
128x3	44.03	0.99	43.71	1.01
256x2	<u>45.37</u>	<u>0.90</u>	<u>45.34</u>	<u>0.85</u>
256x3	**47.26**	**0.74**	**46.05**	**0.8**

Table 3: Conditional NILUT ablation study. All the reported architectures are MLP-Res. We report results when learning 3 and 5 different 3D LUTs styles in a single CNILUT. We train our CNILUT for 10000 steps.

Blending Styles The derivation of our CNILUT also allows us to blend among different styles by just modifying the values of the condition vector. When this vector is a one-hot encoder we have one of the basis implicit 3D LUTs; but we can generalize this vector as a set of blending weights. The multi-style blending occurs as an implicit interpolation within the neural network itself. This is the same principle as in SIREN [41] (interpolation of pixels for 2D images) and NeRF [35]. We analyze the output of blending the test images with weights $[0.33, 0.33, 0.33]$, and compare with the linear interpolation of the real processed 2D RGB images, the PSNR is 40.05dB, indicating the implicit CNILUT blending is equivalent to explicit images interpolation.

We provide examples of our blending capabilities in Fig. 9. We believe the blending property itself represents an interesting future work. We provide more details in the supplementary material.

Figure 10: NILUT as plug-in module to further enhance learned ISPs. (Left) **RAW** image after demosaicing. (Middle) **RGB** image produced using a learned ISP designed for NPU [21, 22]. (Right) Enhanced image using our **NILUT**.

4.2. Plug-in deep learning based ISPs

We explore the integration of NILUTs into modern learned ISPs [11, 44]. Our NILUT is a possible plug-and-play module to further enhance the colors and apply different styles. Moreover, it is differentiable, which facilitates end-to-end ISP optimization. In Fig. 10, we show how to complement a learned ISP [22] designed to run -and tested- on smartphones, using our NILUTs to enhance the image further and manipulate colors.

This is a core step in any traditional ISP [24]. Further work on training or enhancing ISPs is out of the scope of this work, as it is a research topic by itself [23, 28, 32, 44].

4.3. Applications

We provide a demo application in Fig. 11. Following Section 4.2, we deploy a small CNILUT (32x2) on mobile devices using *AI Benchmark* [21] and provide the results in Tab. 4. The model can process 4K images in smartphones without memory problems at $\approx$ 10 FPS on regular GPUs (2020). The CNILUT (3 styles) is just 4KB in comparison to the 1.2 MB (3×0.4MB) of the three complete 3D LUTs.

This technique allows controllable color manipulation by just changing the input condition vectors. By definition, CNILUT is a pixel-wise transformation, therefore the structure of the image is perfectly preserved.

4.4. Limitations and Other Methods

Despite the promising results of learning 3D LUTs as INRs, there are some limitations that we must consider.

Firstly, as we previously discussed in Section 3.1, we understand that a neural network (*e.g.* NILUT) is limited and cannot surpass the efficiency of a lookup operation.

Secondly, by design and mathematical convenience, other methods for learning 3D LUTs such as Zeng *et al.* [50] also achieve "perfect" mapping (*i.e.* $>$ 50dB). However, despite the great fitting, the application of classical 3D LUTs on mobile devices is not trivial as discussed in Sec. 3.1.

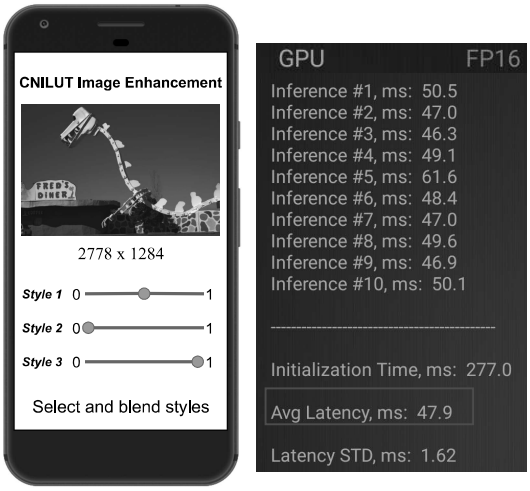


Figure 11: (L) Sample image enhancement application on mobile devices based on the proposed CNILUT. (R) Deployment using [21]. See also Section 4.1 and Figure 9.

Input resolution	Mali-G77 MC9 (ms) ↓	Adreno 620 (ms) ↓
1920×1080	47.9 ± 1.62	75.0 ± 3.28
2778×1284	73.9 ± 1.62	135.0 ± 3.63
3840×2160	121.0 ± 1.41	286.0 ± 2.92

Table 4: CNILUT deployment on two mid-level smartphone GPUs. We report the average image processing runtime over 10 runs $\pm$ the std. deviation (see Figure 11).

Thirdly, many previous approaches [48–50] did not focus on learning explicit 3D LUTs, but instead focused on constructing a mapping between RGB and enhanced RGB (*e.g.* "Expert C" in MIT5K [10]), which was not necessarily limited to 3D LUT operations. We aim to complement these approaches [46, 48–50, 53] with the proposed NILUT that offers multi-style functionality and is suitable for modern mobile devices.

5. Concluding remarks

We have introduced NILUTs, a new approach for modeling 3D LUTs as INRs. Our NILUTs are implicitly defined, continuous, 3D color transformations parameterized by a neural network. They present several advantages: i) are memory-efficient and can run on mobile devices, therefore, are easy to integrate into modern deep learning ISPs; ii) can perform multiple style modifications with a single architecture; and iii) can compute blend between different styles. The novel multi-style blending formulation allows controllable image enhancement and customization. Quantitative and qualitative results demonstrate the effectiveness of our newly defined NILUTs for image enhancement. We have also curated a dataset of 3D LUTs and images for evaluation of color manipulation methods.

Acknowledgements

This work was partially supported by the Alexander von Humboldt Foundation.

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.

JVC was supported by Grant PID2021-128178OB-I00 funded by MCIN/AEI/10.13039/501100011033, ERDF "A way of making Europe", the Departament de Recerca i Universitats from Generalitat de Catalunya with reference 2021SGR01499, and the "Ayudas para la recualificación del sistema universitario español" financed by the European Union-NextGenerationEU.

A. Implementation Details

We develop the models using PyTorch framework and two NVIDIA RTX 3090. The MLP networks are designed based on SIREN (and variants) [41]. The models are trained using fixed learning rate $1e^{-3}$ and Adam optimizer [26] until convergence (*e.g.* 5000 steps, $\sim$ 4 minutes). Note that as we show in Fig. 6, convergence depends on the architecture. For the experiments in Tables 1, 2, 3 we do not use RGB maps of dimension $4096 \times 4096 \times 3$ (*i.e.* complete 16M values), instead, we use a reduced map of dimension $2048 \times 1024 \times 3$ which contains one of each two possible values in the R,G, and B channels (*i.e.* 128^3 values), which requires less memory and allows faster experimentation.

For training the conditional NILUT we use three/five different 3D LUTs, at each step we feed the three/five condition vector and RGB map into the network and accumulate the three/five different loss terms (one for each learned LUT). We concatenate the map and condition vector to obtain an input with dimension $[h \times w, 6]/[h \times w, 8]$. Since we learn three/five different 3D LUTs using a single CNILUT, we need to train at least 4000 steps to obtain reasonable results. Once the CNILUT is trained, we can further finetune it to perform blending by yielding random condition vector weights (*i.e.* softmax weights) and the corresponding blended outputs; these represent plausible convex combinations of the three basis 3D LUTs.

References

[1] https://3dlutcreator.com/3d-lut-creator—materials-and-luts.html. Accessed: 2023-03-05. 4

[2] https://blog.google/products/pixel/introducing-google-tensor/. Accessed: 2023-03-05. 2

[3] https://machinelearning.apple.com/research/neural-engine-transformers. Accessed: 2023-03-05. 2

[4] https://research.ibm.com/blog/ibm-artificial-intelligence-unit-aiu. Accessed: 2023-03-05. 2

[5] https://semiconductor.samsung.com/newsroom/tech-blog/all-about-exynos-2-an-upgraded-mobile-experience-the-important-role-of-cpu-and-npu-in-smartphones. Accessed: 2023-03-05. 2

[6] https://technave.com/gadget/qualcomm-snapdragon-8-gen-2-will-offer-greatly-improved-gpu-npu-and-isp-31990.html. Accessed: 2023-03-05. 2

[7] https://www.sony.com/content/sony/en/en_us/sca/company-news/press-releases/sony-electronics/2022/sony-electronics-new-alpha-7r-v-camera-delivers-a-new-highresolution-imaging-experience-with-aibased-autofocus.html. Accessed: 2023-03-05. 2

[8] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, pages 1692–1700, 2018. 1, 2

[9] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, pages 126–135, 2017. 1, 2

[10] 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. 4, 5, 8

[11] Marcos V Conde, Steven McDonagh, Matteo Maggioni, Ales Leonardis, and Eduardo Pérez-Pellitero. Model-based image signal processors via learnable dictionaries. In *AAAI*, volume 36, pages 481–489, 2022. 1, 8

[12] Marcos V Conde, Florin Vasluianu, Javier Vazquez-Corral, and Radu Timofte. Perceptual image enhancement for smartphone real-time applications. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1848–1858, 2023. 2

[13] Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar. Mobile computational photography: A tour. *arXiv preprint arXiv:2102.09000*, 2021. 1

[14] Eric Garcia and Maya Gupta. Lattice regression. *NeurIPS*, 22, 2009. 2, 3

[15] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, pages 7154–7164, 2019. 2

[16] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017. 2

[17] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *CVPR*, pages 1780–1789, 2020. 2

[18] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pajak, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. Flexisp: A flexible camera image processing framework. *ACM Transactions on Graphics (ToG)*, 33(6):1–13, 2014. 1

[19] Daniel Hernandez-Juarez, Sarah Parisot, Benjamin Busam, Ales Leonardis, Gregory Slabaugh, and Steven McDonagh. A multi-hypothesis approach to color constancy. In *VPR*, 2020. 1

[20] Andrey Ignatov, Cheng-Ming Chiang, Hsien-Kai Kuo, Anastasia Sycheva, and Radu Timofte. Learned smartphone isp on mobile npus with deep learning, mobile ai 2021 challenge: Report. In *CVPR Workshops*, pages 2503–2514, 2021. 1, 2, 3

[21] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. Ai benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 1, 2, 3, 8

[22] Andrey Ignatov, Radu Timofte, Maurizio Denna, and Abdel Younes. Real-time quantized image super-resolution on mobile npus, mobile ai 2021 challenge: Report. In *CVPR*, pages 2525–2534, 2021. 1, 2, 3, 8

[23] Andrey Ignatov, Luc Van Gool, and Radu Timofte. Replacing mobile camera isp with a single deep learning model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 536–537, 2020. 1, 2, 8

[24] Hakki Can Karaimer and Michael S Brown. A software platform for manipulating the camera imaging pipeline. In *ECCV*, pages 429–444, 2016. 1, 2, 6, 8

[25] Hakki Can Karaimer and Michael S Brown. Improving color reproduction accuracy on cameras. In *CVPR*, 2018. 1, 2

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 9

[27] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021. 2

[28] Zhetong Liang, Jianrui Cai, Zisheng Cao, and Lei Zhang. Cameranet: A two-stage framework for effective camera isp learning. *IEEE Transactions on Image Processing*, 30:2248–2262, 2021. 1, 2, 8

[29] Hai Ting Lin, Zheng Lu, Seon Joo Kim, and Michael S Brown. Nonuniform lattice regression for modeling the camera imaging pipeline. In *ECCV*, pages 556–568, 2012. 2

[30] Chengxu Liu, Huan Yang, Jianlong Fu, and Xueming Qian. 4d lut: Learnable context-aware 4d lookup table for image enhancement. *arXiv preprint arXiv:2209.01749*, 2022. 2

[31] Shuai Liu, Chaoyu Feng, Xiaotao Wang, Hao Wang, Ran Zhu, Yongqiang Li, and Lei Lei. Deep-flexisp: A three-stage framework for night photography rendering. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1210–1219, 2022. 1, 2

[32] Shuai Liu, Chaoyu Feng, Xiaotao Wang, Hao Wang, Ran Zhu, Yongqiang Li, and Lei Lei. Deep-flexisp: A three-stage framework for night photography rendering. In *CVPR*, pages 1211–1220, 2022. 1, 2, 8

[33] Long Ma, Tengyu Ma, Risheng Liu, Xin Fan, and Zhongxuan Luo. Toward fast, flexible, and robust low-light image enhancement. In *CVPR*, pages 5637–5646, 2022. 2

[34] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *ICCV*, pages 4743–4752, 2019. 2

[35] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2, 7

[36] Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. Deeplpf: Deep local parametric filters for image enhancement. In *CVPR*, pages 12826–12835, 2020. 2

[37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 3, 5

[38] Seungjun Nah, Sanghyun Son, Suyoung Lee, Radu Timofte, and Kyoung Mu Lee. Ntire 2021 challenge on image deblurring. In *CVPR Workshops*, pages 149–165, 2021. 2

[39] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018. 1

[40] Gaurav Sharma and Raja Bala. *Digital color imaging handbook*. CRC press, 2017. 4, 5, 7

[41] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, volume 33, pages 7462–7473, 2020. 1, 2, 3, 4, 5, 7, 9

[42] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *ECCV*, pages 74–91, 2022. 2

[43] 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*, volume 33, pages 7537–7547, 2020. 2, 3

[44] Ethan Tseng, Yuxuan Zhang, Lars Jebe, Xuaner Zhang, Zhihao Xia, Yifei Fan, Felix Heide, and Jiawen Chen. Neural photo-finishing. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022. 1, 8

[45] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxim: Multi-axis mlp for image processing. In *CVPR*, pages 5769–5780, 2022. 2

[46] Tao Wang, Yong Li, Jingyang Peng, Yipeng Ma, Xian Wang, Fenglong Song, and Youliang Yan. Real-time image enhancer via learnable spatial-aware 3d lookup tables. In *ICCV*, pages 2471–2480, 2021. 1, 2, 8

[47] Wencheng Wang, Xiaojin Wu, Xiaohui Yuan, and Zairui Gao. An experiment-based review of low-light image enhancement methods. *Ieee Access*, 8:87884–87917, 2020. 2

[48] Canqian Yang, Meiguang Jin, Xu Jia, Yi Xu, and Ying Chen. Adaint: Learning adaptive intervals for 3d lookup tables on real-time image enhancement. In *CVPR*, pages 17522–17531, 2022. 1, 2, 3, 7, 8

[49] Canqian Yang, Meiguang Jin, Yi Xu, Rui Zhang, Ying Chen, and Huaida Liu. Seplut: Separable image-adaptive lookup tables for real-time image enhancement. In *ECCV*, pages 201–217, 2022. 2, 8

[50] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1, 2, 3, 7, 8

[51] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017. 2

[52] Zhaoyang Zhang, Yitong Jiang, Jun Jiang, Xiaogang Wang, Ping Luo, and Jinwei Gu. Star: A structure-aware lightweight transformer for real-time image enhancement. In *ICCV*, pages 4106–4115, 2021. 2

[53] Luxi Zhao, Abdelrahman Abdelhamed, and Michael S Brown. Learning tone curves for local image enhancement. *IEEE Access*, 2022. 1, 2, 8