Image Enhancement using Color Naming in Transformer-Based Models

Francisco Antonio Molina Bakhos

Abstract

Color naming, the process of categorizing colors into universal terms, plays a significant role in how humans perceive and describe images. Leveraging this concept, this thesis integrates color naming probability maps into transformer-based models to enhance image retouching. By embedding these maps into the deep learning pipeline of models like Restormer and PromptIR, the study aims to replicate the nuanced adjustments made by expert retouchers. The results demonstrate that incorporating color naming improves color accuracy and visual quality, providing a novel approach to automated image enhancement.

Index Terms

Color Naming · Image Enhancement · PromptIR · Restormer · Transformer · Visual Quality

I. Introduction

PHOTO enhancement is a critical component of professional photography, where it involves meticulous adjustments to brightness, contrast, color balance, and other elements to create visually appealing images. Expert retouchers rely on their skills and artistic vision to transform raw photographs into polished, aesthetically pleasing works of art. However, despite their expertise, the enhancement process is often time-consuming and requires a high-level of precision. This has spurred interest in developing tools to assist professionals with the enhancement process without sacrificing visual quality.

Image enhancement, as a broader concept, refers to the process of adjusting digital images to make them better suited for specific purposes—such as making them more visually appealing. In recent years, deep learning techniques have increasingly been employed in this domain, thanks in part to the availability of datasets consisting of images retouched by photographers or colorists. These datasets are designed to train models to emulate the style and quality of expert retouching consistently. However, other image-to-image tasks such as image restoration (e.g. denoising, deblurring, dehazing) have overshadowed image enhancement developing powerful deep learning models for primarily restoration tasks.

In this thesis, we aim to bridge this gap by repurposing state-of-the-art image restoration models for image enhancement. We explore how existing models, originally designed for restoration tasks, can be adapted and optimized for making images visually pleasant. Specifically, we leverage tools and techniques that allow these models to perform image enhancement effectively.

Both professionals and amateurs typically use software applications like Adobe Lightroom to manipulate images. These applications often allow them to manipulate the images based on a small set of fixed colors, which usually coincide with the ones universal across languages found by Berlin and Kay [1]. These 11 universal color names found in most societies are: orange, brown, yellow, white, grey, black, pink, purple, red, green, and blue. This area of study is known as *color naming*. Recent research by Serrano et al. [2] has demonstrated that color naming can be effectively utilized in image enhancement by decomposing the image into different color names and learning specific tone curves for each.

We hypothesize that by incorporating color naming as a visual feature to guide state-of-the-art transformer-based image restoration models, such as Restormer [3] and PromptIR [4], we can aid them in the task of mimicking the style of expert photographers. We conduct an extensive analysis of how color naming can be integrated into transformer-based architectures, demonstrating improvements on the well-known MIT-Adobe-FiveK dataset [5]. Our findings show that by incorporating color naming, we can enhance the performance of these models, bringing them closer to the state of the art in image enhancement.

II. State of the art

The camera imaging pipeline is designed to convert raw sensor data into visually appealing images by employing a series of image restoration and enhancement techniques. Traditional restoration methods include demosaicing, noise reduction, and lens distortion correction, while enhancement techniques often involve white balance adjustment and color correction [6]. While these processes are integral to producing high-quality images, they are not always sufficient to achieve optimal results, especially in challenging environments or under suboptimal shooting conditions. As a result, additional image restoration and enhancement methods are often applied in an sRGB-to-sRGB workflow, where the goal is to further refine the image after it has passed through the camera's default processing pipeline. In the following sections, we provide an in-depth review of the latest advancements in image restoration and enhancement methods, focusing on techniques that extend beyond the limitations of the camera pipeline.

Author: Francisco A. Molina Bakhos, franciscoantonio.molina@autonoma.cat
Advisor 1: Javier Vazquez Corral, Computer Vision Center, Universitat Autònoma de Barcelona
Advisor 2: David Serrano Lozano, Computer Vision Center, Universitat Autònoma de Barcelona
Thesis dissertation submitted: September 2024

A. Image restoration

Image restoration aims to recover the original image from a degraded version by reversing processes such as blurring, noise, or low resolution. It addresses both known and unknown forms of degradation, restoring the image to its intended quality.

With the advent of deep learning, data-driven approaches have become the state of the art for image restoration. CNN-based methods [7, 8] and generative-based models [9, 10] have shown great success in recent years. However, our focus is on transformer-based models, which have rapidly gained popularity due to their superior performance in capturing both local and global dependencies. As transformers have surged in use, several innovative architectures have been developed specifically for image restoration tasks. One such model is SwinIR, proposed by Liang et al. [11]. SwinIR features a shallow feature extraction module and a deep feature extraction module built from Residual Swin Transformer Blocks, which utilize Swin Transformer Layers [12]. It also includes a high-quality image reconstruction module, adaptable to specific restoration tasks, allowing for flexibility across a wide range of degradation challenges. Wang et al. [13] introduced Uformer, a hierarchical network with skip connections between the encoder and decoder stages. At each stage, stacks of Locally-enhanced Window (LeWin) Transformer blocks combine the self-attention mechanism with convolutional operators. This allows Uformer to capture both long-range dependencies and useful local context, making it highly effective for image restoration.

A key model in our research is Restormer, designed by Zamir et al. [3]. Restormer adopts a multiscale hierarchical structure with two specialized core modules: Multi-Dconv Head Transposed Attention (MDTA) for efficient feature interaction across channels, and the Gated-Dconv Feed-forward Network (GDFN), which uses a gating mechanism to propagate relevant information through depthwise convolutions. This architecture is known for its computational efficiency and high-quality restoration capabilities. Although other image restoration models only work for a specific known degradation, Restormer can effectively recover several degradations such as rain, haze, and noise.

We also utilize PromptIR, a more recent model proposed by Potlapalli et al. [4], which builds upon Restormer with one crucial enhancement. PromptIR introduces a "prompt block" within the decoder, consisting of two key modules: the Prompt Generation Module (PGM), which generates input-conditioned prompts based on input features, and the Prompt Interaction Module (PIM), which dynamically adapts the input features using the generated prompt via another transformer block. This addition allows for greater adaptability and improved restoration outcomes.

B. Image enhancement

The goal of image enhancement is to improve the visual appearance or perceptual quality of an image, emphasizing specific features to make it more suitable for a particular application or aesthetically pleasing to human viewers. Enhancements can range from correcting color balance and contrast to improving visibility in low-light or underwater conditions.

Like in image restoration, data-driven methods have become the state-of-the-art in image enhancement. Recent advancements include CNN-based, generative-based, and transformer-based architectures applied to various enhancement tasks, such as low-light image enhancement [14, 15, 16, 17, 18, 19] and underwater image enhancement [20, 21, 22, 23, 24, 25]. However, our focus is on mimicking the manual retouching process performed by experts to produce visually pleasing images.

Yan et al.[26] pioneered the use of deep neural networks to replicate photographic editing styles. They designed a multilayer perceptron (MLP) that used informative feature descriptors at pixel, contextual, and global levels to capture both low- and high-level image information. Their model outputted parameters to perform image adjustments that emulated expert retouching. Similarly, Wang et al.[27] proposed a CNN-based network that estimated an image-to-illumination mapping to enhance visual quality, while Moran et al. [28] developed DeepLPF, a CNN backbone that predicted three types of filters to be applied to the image mimicking the process of local editing.

The concept of pixel-wise curve estimation for dynamic range modification was introduced by Zero-DCE [15], inspiring further research into curve-based image enhancement methods, such as CURL [29], FlexiCurve [30], and NamedCurves [2]. Another widely used method involves learning lookup tables (LUTs) for image manipulation. Zeng et al.[31] and Wang et al.[32] proposed using 3D LUTs for real-time enhancement. Yang et al. [33] later introduced AdaInt, which increased the capability of 3D LUTs through dense sampling in the color space.

Image-to-image methods have also been applied to the enhancement task, often utilizing Generative Adversarial Networks (GANs). Studies such as those by Chen et al.[34], Ni et al.[35], and Jiang et al. [36] leveraged GANs to perform image enhancement by learning from paired or unpaired data.

In recent years, transformer-based approaches have emerged for image enhancement. Zhang et al.[37] introduced the Structure-Aware Transformer Network (STAR) for real-time image enhancement. In STAR, the image is divided into patches and tokenized, which are then fed into a long-short range transformer module. The output sequence from the transformer is processed by downstream tasks for enhancement. Cui et al.[38] introduced the Illumination Adaptive Transformer (IAT) for image enhancement and exposure correction. IAT employs a two-branch model: a local branch with depth-wise convolution for latent feature mapping, and a global branch using transformer attention to adjust Image Signal Processor (ISP) parameters, such as color and gamma correction. With only $\sim$90K parameters, IAT achieves adaptive enhancement across diverse lighting conditions with high processing speed.

C. Color naming

Color naming is the process of assigning labels or names to the colors we perceive, categorizing the visible part of the electromagnetic spectrum into discrete terms. It is influenced by universal perceptual mechanisms and cultural variations. In their seminal study, Berlin and Kay [1] explored whether there are universal patterns in the way different languages and cultures name colors. They discovered that color terms tend to emerge in a specific order across languages and identified 11 basic color names that are common: *black, white, red, green, yellow, blue, brown, purple, pink, orange*, and *gray*. Subsequent research has expanded on this concept by exploring the role of communicative needs [39, 40, 41], the economy and efficiency of language [42, 43, 44], and psychophysiological factors [45, 46] that may influence how color terms are structured and used.

In the field of computer vision, researchers have adapted the concept of color naming to develop algorithms that categorize and label colors in images. Mojsilovic [47] introduced a computational approach to color naming by designing a color-naming metric and creating an algorithm to extract the color composition of complex images based on human-labeled data. Benavente et al. performed a psychophysical test with different patches in order to develop a new model based on fuzzy measurements [48, 49]. Van de Weijer et al.[50] proposed learning color names directly from real-world images without relying on human test subjects. This approach was extended by using statistical and probabilistic models to better capture color naming dynamics[51, 52], and other researchers have focused on improving color naming accuracy in challenging visual conditions, such as shadowed or highlighted regions [53]. Deep learning techniques have further advanced the task of color naming. Recent studies have employed neural networks to learn color names and build efficient color-naming systems [54, 55, 56, 57]. These models are trained to map pixel values to color names, automating the process with greater precision and adaptability.

Beyond the direct task of color naming, researchers have applied this concept to enhance performance in other computer vision tasks, such as object detection and recognition [58, 59]. By incorporating color naming as an additional feature, these systems can achieve more robust results in scenarios where color plays a significant role.

In this thesis, we utilize color naming as a visual prompt to guide the end-to-end training of our model. By leveraging color naming, we aim to enhance the model's ability to produce results that align with human color perception, ultimately improving image enhancement.

III. Method

We aim to enhance a low-quality sRGB input image x by a learned model that will output an enhanced version $\hat{y}$, which is derived to be as similar as possible to the expert-retouched image y based on a loss function L.

Figure 1 shows our approach and all of the variants that are performed in this thesis. The variants are explained with further detail in section IV. The main components of our approach are the color naming module, which we use to decompose the image into six or eleven color probability maps, and the color naming encoder, which we use to encode the features of the color naming maps so that we can use them inside the different models. To obtain the color naming maps, we also considered the possibility of including the backbone that was used in [2], to standardize the input image into a canonical latent space first. When using the color naming probability maps, we also considered concatenating them into the channels of the input image itself, creating a nine- or fourteen-channel image, to check if the models themselves could take advantage of the information the maps provided without having to use the encoder to extract the features from them.

In the following sections, we detail the functionality of these modules, also explaining how we introduced the encoded features of the color naming maps in the different models we used.

A. Color Naming

By using color naming pixel-wise, we decompose an image perceptually in a set of color names. As we have mentioned in Section I, our hypothesis is that this decomposition might help state-of-the-art image restoration models to better learn images produced by image retouchers. Berlin and Kay [1] identified 11 basic colors that are common across languages: *black, white, red, green, yellow, blue, brown, purple, pink, orange* and *gray*. Color naming algorithms define the boundaries of these terms. We use Van de Weijer et al. [50] color classification method, as seen in Figure 2.

The Van de Weijer et al. [50] color naming model decomposes an image into probability maps, one for each color name. The method decomposes each pixel into an 11-dimensional vector that represents the probability of that pixel being classified as one of the universal colors. To visualize the result of the decomposition, we can look at Figure 3. The color naming probability maps are color coded to help with the visualization. Each map represents the value for each pixel in one of the 11 dimensions the color naming model provides. Just as in [2], we considered reducing the set of 11 probability maps to just 6 by grouping the color names that share similar hues, but differ in intensity. More in detail we joined i) pink and purple, ii) orange, brown and yellow, and iii) black, grey and white. Since they are probabilities, each 11-dimensional vector sums up to 1 and, therefore, by adding the corresponding maps we can reduce their amount in a simple yet effective way. Both approaches are explored in section IV.

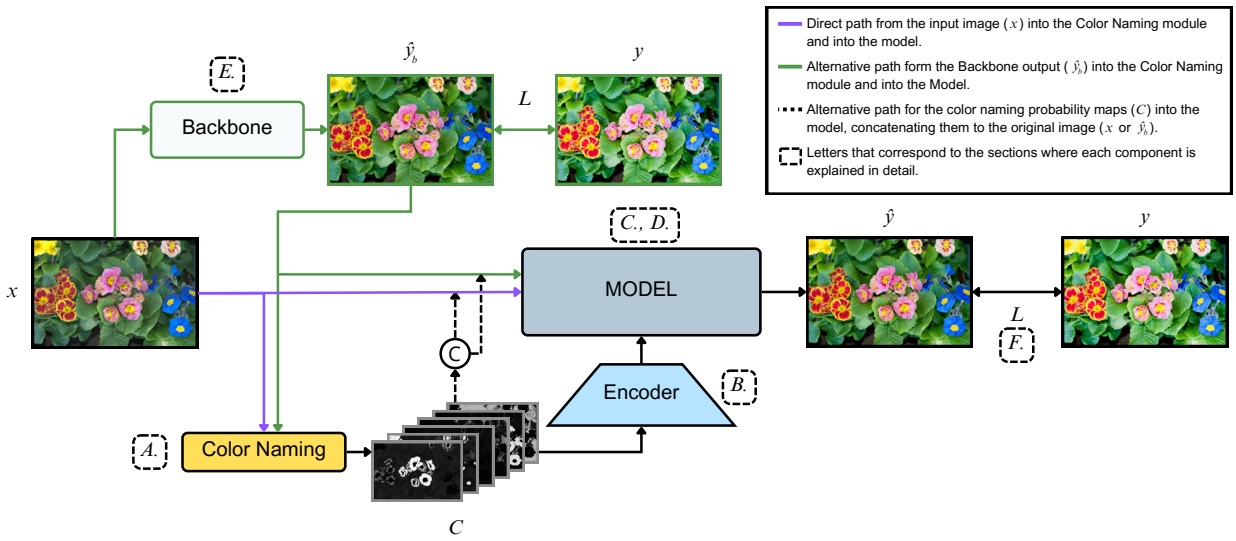


Fig. 1: An overview of the proposed method and all the variants analyzed in this thesis. The labeled components correspond to subsections in Section III, where each is detailed. The process begins with the color naming module *(A)*, which generates probability maps from the input image. These maps are then encoded *(B)* and used as visual features in the main baseline model *(C, D)*. We apply a lightweight backbone (E) to normalize scene conditions. The final enhanced image $\hat{y}$ is compared to the ground-truth y using a loss function L *(F)*.

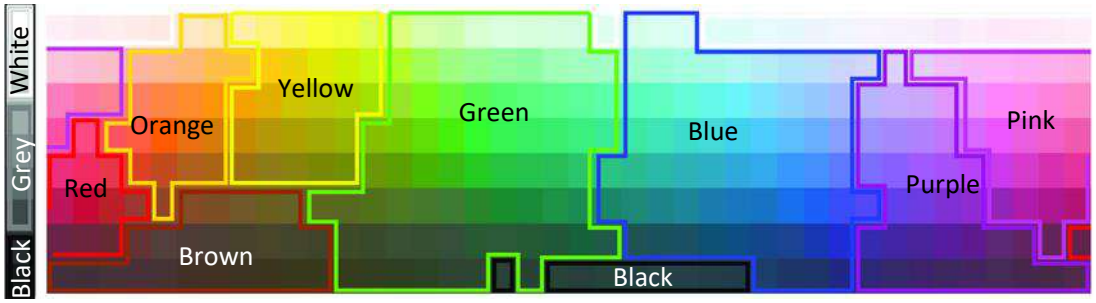


Fig. 2: Van de Weijer et al. [50] color names grouped in the Munsell color array.

B. Color Naming Encoder

The main module of our approach is the Color Naming Encoder (CNE). The CNE extract features from the color naming probability maps.

We first derive a basic CNE. It is shown in Figure 4a. This basic CNE is composed of a 3x3 convolution that duplicates the number of channels, an activation function to introduce non-linearities, and a pooling layer to reduce the spatial dimensions by half.

We propose also two new versions of the CNE. These new versions aim to reduce the number of parameters and diversify the receptive fields of the convolutions. The first version is denoted as Squeeze CNE and is shown in Figure 4b. In this case, the CNE is based on the Fire module developed by Iandola et al. [60] for SqueezeNet. The squeeze layer reduces the number of channels first, significantly reducing the number of parameters and learning a compressed set of features. Then, the expanding layer, composed of the 1x1 and the 3x3 convolutions, increases the dimensionality again, reintroduces complexity, and diversifies the receptive field of the encoder. The second version of the CNE is called Inception CNE and is shown in Figure 4c. This CNE is based on the inception module developed by Szegedy et al. [61]. The 1x1 pointwise convolutions reduce the number of channels and, therefore, the number of parameters. The 3x3 and 5x5 convolutions diversify the receptive fields of the filters for a more comprehensive feature extraction process and the parallel pooling path helps preserve spatial information from the input itself.

In the following subsections, we describe how we incorporate the features outputted by the CNE in both of the considered models—Restormer and PromptIR.

C. Restormer with Color Naming (RCN)

The Restormer [3], which is a transformer-based, U-Net-like network, was originally designed with the purpose of image restoration. It consists of a hierarchical design that incorporates transformer blocks with two designed core modules: a Multi-

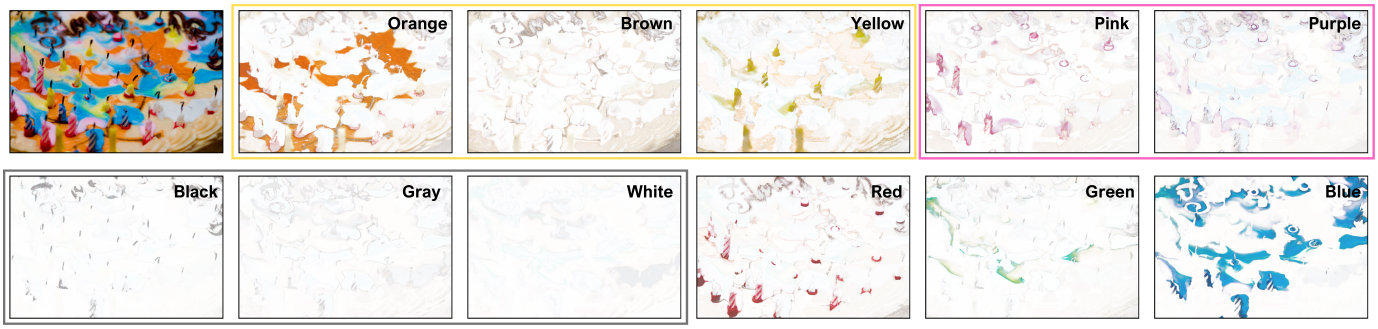
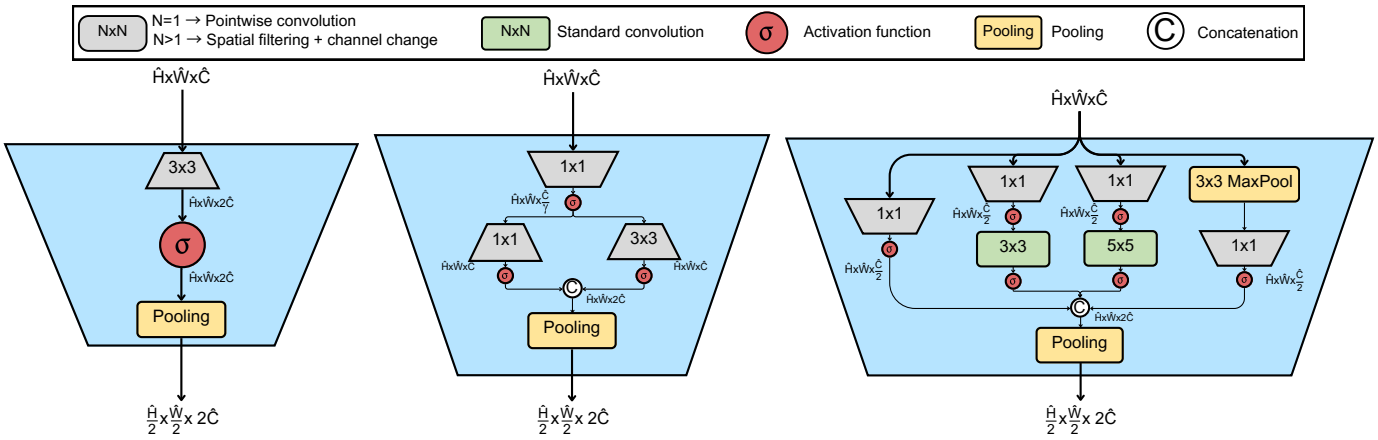


Fig. 3: Color naming probability maps obtained using the Van de Weijer et al. [50] color naming model. They were extracted from the expert-retouched image with the name a1307-WP_CRW_3693 in the MIT-Adobe FiveK dataset [5], which we see on the top left. The maps are color coded to help with the visualization. Grouped in colored rectangles are the maps that would be added when only using 6 probability maps, which would be *orange-brown-yellow*, *pink-purple*, *achromatic*, *red*, *green* and *blue*, from left-to-right and top-to-bottom.



(a) Basic (b) Squeeze (c) Inception

Fig. 4: The three types of encoders developed to extract features from the probability color naming maps. (a) **Basic CNE**. It is composed of a 3x3 pointwise convolution that will duplicate the channels of the input, an activation function to introduce non-linearity, and a pooling method to reduce the spatial dimensions of the input. (b) **Squeeze CNE**. Encoder based on the Fire module developed by Iandola et al. [60] for SqueezeNet. Composed by a 1x1 pointwise convolution (squeeze), a 1x1, and a 3x3 standard convolution (expand) whose outputs will be concatenated in the channel dimensions and pooling to reduce the spatial dimensions. (c) **Inception CNE**. Encoder based on the Inception module developed by Szegedy et al. [61] for their Inception architecture. Composed of the four branches and concatenation of the Inception module and pooling at the end to reduce the spatial dimensions by half.

Dconv head transposed attention (MDTA) that performs feature interaction across channels and a gated-DConv feed-forward network (GDFN) that introduces a gating mechanism to allow useful information to propagate further.

We modified the Restormer's architecture to inject the color-naming features, denoted as C, given by the color naming. To do so, we start by applying a 3x3 convolution to the color probability maps, mimicking what it is done in the original architecture with the input image. Then, to extract deeper features, we concatenate several CNEs, so that the color naming features have the corresponding spatial dimensions and number of channels so that they can be integrated in the attention mechanisms at the different levels of the encoder-decoder architecture. As we can see in Figure 5, the incorporation of these color naming features in the transformer blocks is represented with a **switch symbol**, going from 1 to 6 in each of the transformer blocks that was used. It was designed like this to use these features in different ways, for example, using color naming only in the encoder part of the network (closing switches 1, 2, 3, and 4), using color naming only in the decoder part of the network (closing switches 4, 5, and 6), or any other combination. For switch (1), the features after the first convolution can be used in the first transformer block. For switches (2) and (6), the features processed by the first CNE are used. For switches (3) and (5), the features have also been transformed by the second CNE, and for switch (4) the features have gone through all CNEs. The combinations explored are developed with more detail in Section IV.

To integrate the color-naming features C in the transformer blocks, we modified the MDTA module. First, C is processed with a normalization and a depth-wise convolution, the same transformation the image features, denoted as X, undergo in the standard Restormer module. After that, C is included in the attention in four different ways. The first version is shown in figure 5a, where we show how C is added to the query. This way, we do not lose the information that the original model provides from the query tensor. In the second version —shown in figure 5b— we use C as the whole query. In this case, only the features extracted from the color naming maps are used as the query. In the other two versions, we explore a similar approach than the one just explained, but considering the value tensor in the attention mechanism. These versions are shown

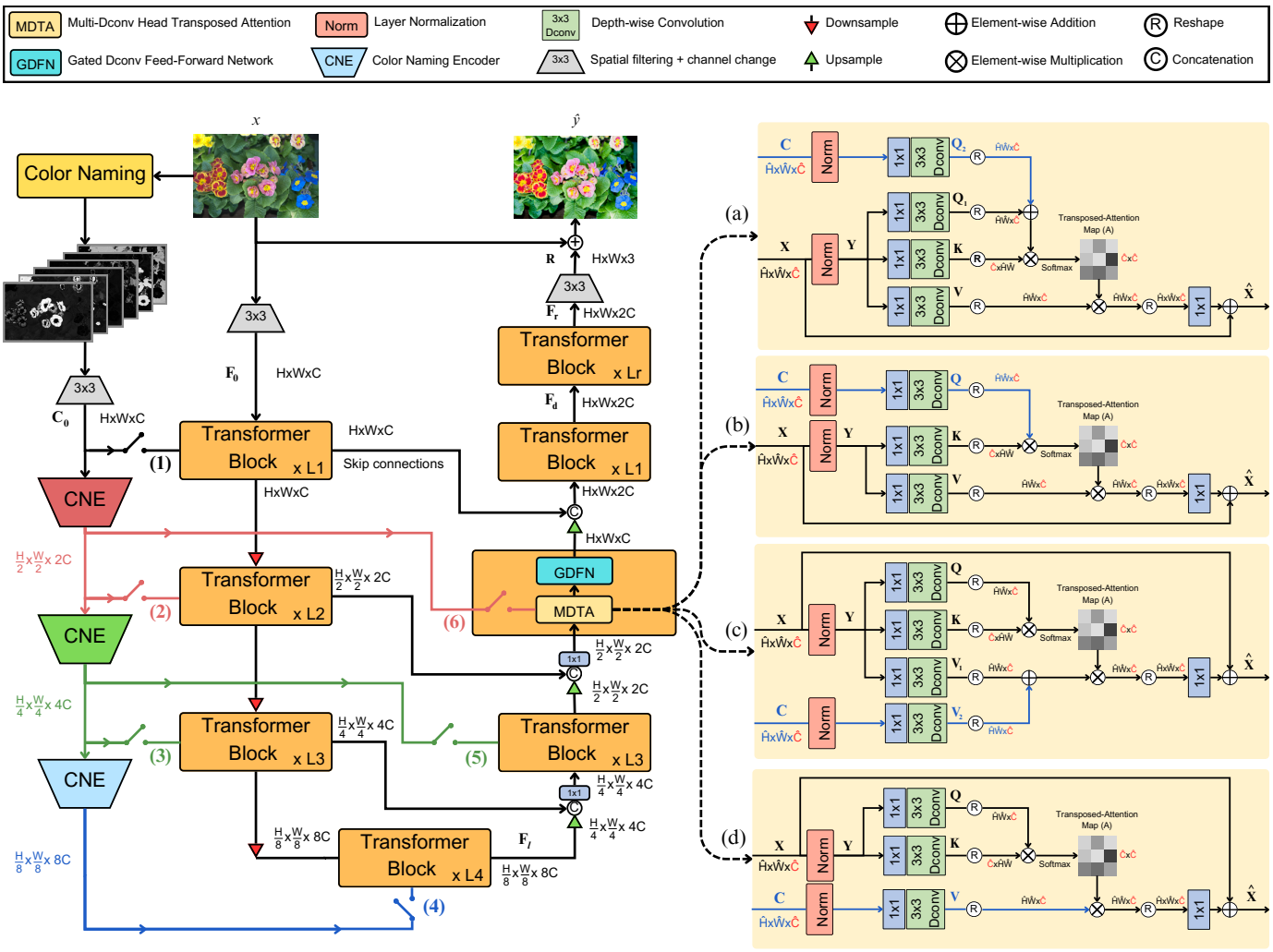


Fig. 5: Restormer with Color Naming (RCN). In the left part of the image, we present an overview of the architecture. The CNEs are introduced to extract the features of the color naming probability maps C. The use of C in each transformer block is represented with a switch (numbers 1 to 6) so that we can introduce these features in the model in different ways. Each level of color naming features C has been color coded so it is easier to visualize their path to the different modules where they can be used. To use them in the attention mechanism, the MDTA module was modified in four different ways: (a) adding C to the query, (b) using C as the query, (c) adding C to the value, and (d) using C as the value. In each of these cases, C undergoes the same transformation as X before being used in the attention, with a normalization and a depth-wise convolution.

in figure 5c where C is added to the value tensor, and in figure 5d where C is used as the value tensor. Please note that if the switch of the block is open, i.e., if C is not used in the transformer block, the original MDTA module is used, performing self-attention only on the input features X. Only one of these five configurations can be used at a time. Therefore, in section IV we describe with more detail how we explored their effects in the model.

D. PromptIR with Color Naming (PIRCN)

Potlapalli et al. [4] modified the architecture of the Restormer, adding a "prompt block" that enriched the input features with information related to the type of degradation the image had suffered, creating PromptIR.

Since Restormer [3] and PromptIR [4] are very similar models, we first did the same modifications explained before to PromptIR, integrating the color naming features C in the MDTA modules. Moreover, taking benefit from the particularities in PromptIR, in this case we also considered integrating the features in the prompt block.

The prompt block is composed of two distinct parts which: the Prompt Generation Module (PGM), which generates an input-conditioned prompt P using the input features F and a set of prompt components P_c, and the Prompt Interaction Module (PIM), which adapts the input features F using P with another transformer block.

Our approach modified the prompt block by using two sets of input features. Figure 6 shows our proposed modified to the prompt block. One of the sets of input features, Y, would be used so that the PGM can generate the prompt P conditioned by them. The other set, X, would then interact with P in the PIM, adapting X using the prompt P generated by the other set of

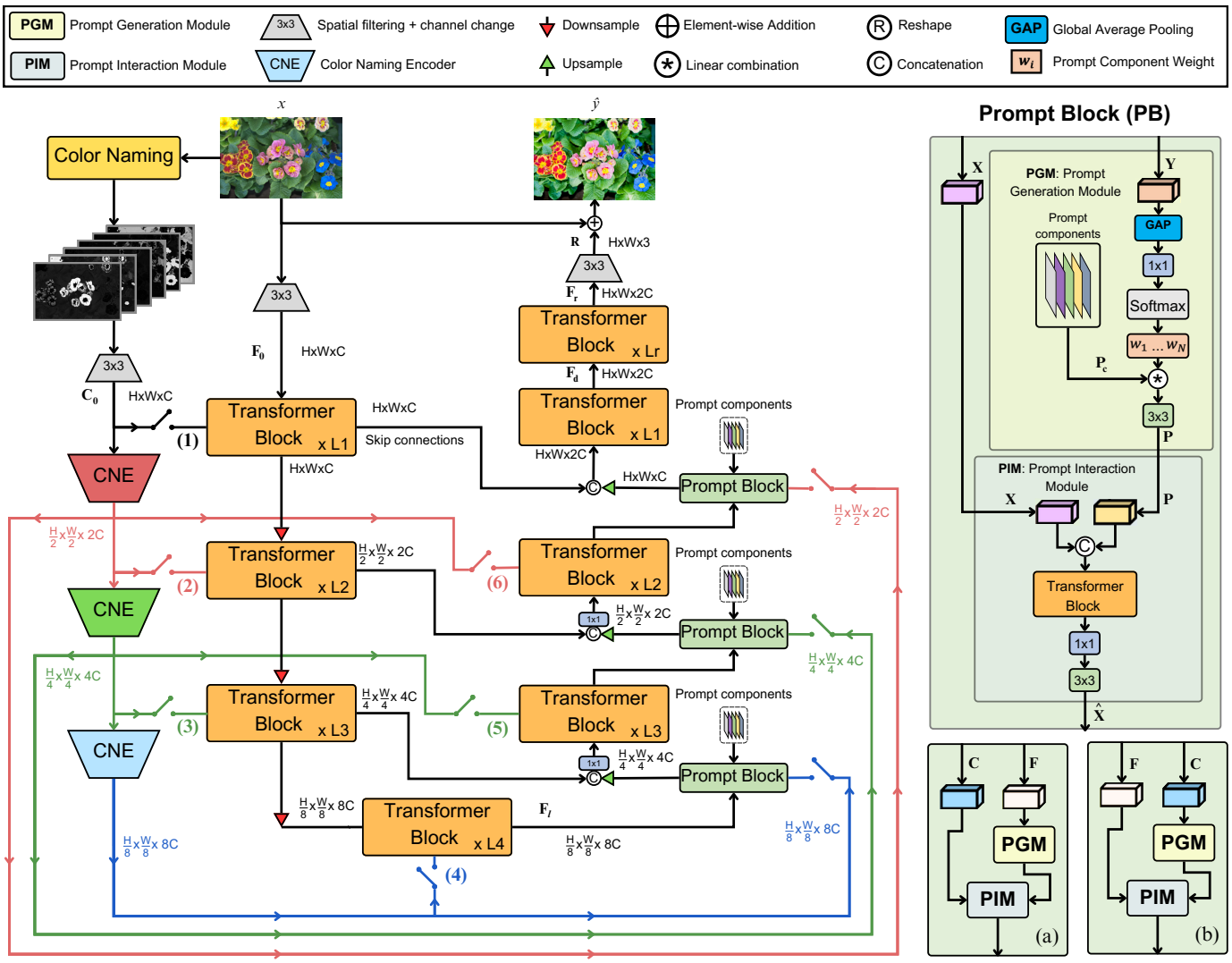


Fig. 6: PromptIR with Color Naming (PIRCN). In the left part of the image, we present an overview of the architecture. The CNEs extract the features from the color naming probability maps, C, to be used in the different levels of the transformer blocks and the prompt blocks in the decoder part of the network. Again, each level of color naming features C has been color coded. On the right part of the image, we see our modified prompt block. Two sets of input features are used: one of them, Y, goes through the PGM to generate prompt P, and the other one, X, interacts with P in the PIM, creating a new set of features $\hat{X}$ by using both of them. At the bottom of it, we see the two possibilities we have with the input features F and the color naming features C: (a) use F to generate the prompt that will interact with C or (b) use C to generate the prompt that will interact with F.

features Y. This way, we generate a prompt based on one of the sets of input features and we make it interact with the other set of features. Our goal with this modification is to combine the input features F that come from the transformer blocks and the color naming features C that come from the incorporated CNEs. Following this approach, we could make them interact with one another in two ways: like in figure 6a, generating a prompt from F and making it interact with C, or like in figure 6b, which is the converse case—generating a prompt from c and making it interact with C.

Again, we introduce the CNEs as we did in the previous section to extract features from the color naming probability maps. In all cases, the use of these features can be withdrawn, represented with the switch symbol again. Just as in the previous section with the transformer blocks, if the color naming features are not used in the prompt blocks, they act as in the original model. This way, we can see how these features affect the performance of the network when used in the different parts in which we considered their inclusion. The different combinations are explored with further detail in Section IV.

E. Backbone

Generally, when working in image restoration, the input images are under or over-exposed according to the scene conditions. This presents a problem for our particular approach with color naming, since the color naming probability maps of these types of images could be a misrepresentation of the actual colors in the image itself. The different camera settings and lighting conditions could affect our ability to consistently obtain the correct color names from the images.

For that reason, we considered the incorporation of a light and efficient backbone network in the pipeline. This backbone aims at pre-processing these images and get a version of them closer to the expert-retouched one, standardizing them into a canonical latent space. This way, the color naming probability maps will represent accurately the colors in the image.

The backbone we introduced is the one used in [2]. It was inspired by the LPIENet [62], a very light network created for image enhancement in smartphone real-time applications. The one we use consists of three encoder blocks and two decoder blocks. Each encoder block consists of two MobileNet [63] layers, a CBAM [64] module, and a max-pooling layer. The decoder follows the same structure, replacing the pooling with bilinear upsampling layers. The encoder blocks are connected to the decoder ones by multi-resolution skip connections, that consist of three parallel branches of convolutional layers with different dilation rates.

F. Loss Function

For the loss function we follow the same approach as in Serrano et al. work [2]. Our loss is composed of three terms. The initial term aims at obtaining a good enough standardized output by the backbone. This is done by calculating the L_2 loss between the backbone-standardized image $\hat{y}_b$ and the ground truth y, with a weighting factor α. This factor is set to 0 when there is no backbone present during training. The subsequent two terms assess the fidelity of the final output, and they compute the L_2 and $SSIM$ losses between the output of the full model y (backbone plus main net) and the ground truth y. Mathematically:

$$L(\hat{y_b}, \hat{y}, y) = \alpha||y - \hat{y_b}||_2 + ||y - \hat{y}||_2 + (1 - SSIM(y, \hat{y})). \tag{1}$$

IV. Experiments

In this section, we describe the details of the experiments.

A. Dataset

We used the MIT-Adobe-5K dataset [5] to compare our approaches with the baseline models. The MIT5K consists of 5,000 photographs taken with DSLR cameras by a set of different photographers, covering a broad range of scenes, subjects, and lighting conditions. Five photography students were hired to adjust the tone of the photos, each of them retouching the entire dataset, reaching what they considered visually pleasing renditions of the raw images.

This dataset has had different splits along the years. We shortly enumerate them here:

- MIT-Adobe-5K-DPE (DPE) version: DPE [34] split it into three partitions: 2,250 images and their retouched versions for supervised or paired training, 2,250 images for unpaired training (using this split as the target domain while using the first as source domain), and 500 images for testing in either setting. FlexiCurve [30] used the same split. Other studies [28, 29] that did not require paired and unpaired training similarly split the data to be able to compare their results to these SOTA methods, splitting into 2,250 images for training and 500 for testing, selecting 500 random images from the training set for hyperparameter optimization.
- MIT-Adobe-5K-UPE (UPE): UPE [27] established this partition, pre-processing the images to be under-exposed. DeepLPF [28] also used this partition and pre-processing process to compare results. This partition splits the data into 4,500 images for training and the other 500 for testing.
- MIT-Adobe-5K-UEGAN (UEGAN): UEGAN [35] followed the same split as UPE, however at a different resolution. Studies such as 3DLUT [31] and AdaInt [33] used this partition. In some of these cases, 500 images from the training set, randomly sampled, were used as the validation set.

In this work, we decided to use a different split than the previously mentioned, in order to combine the best of the different splits. Thus, we selected the same pre-processing and image sizes as in the UEGAN partition, but we used 3,500 images and their retouched versions for training, 750 for validation, and 750 for testing. This way, we get a larger set of images for training than the DPE partition while still having large enough validation and test sets. All the experiments to find the best approach to introduce the color naming maps in the pipelines of the selected models were performed with this partition. This said, once the best models were found, we also decided to train and test them with the UEGAN partition, in order to compare our approach with current state-of-the-art methods for image enhancement.

We also followed the current works [34, 28, 29, 30, 27, 33], and used as reference the version retouched by expert C, since he was ranked the highest at Bychokovsky et al. [5] user study.

B. Implementation details

In all experiments, unless it is specified otherwise, we use the following training parameters, which are the same as in the Restormer [3]. We trained the models with AdamW optimizer (β_1=0.9, β_2=0.999, weight decay $1e^{-4}$) and our loss function (equation 1) for 300K iterations with the initial learning rate $3e^{-4}$ gradually reduced to $1e^{-6}$ with the cosine annealing. For the same reasons stated in [3], we also performed progressive learning, where the models are trained on smaller image patches

in early epochs that gradually grow until the end of learning. The batch size is reduced as the patch size increases to maintain a similar time per optimization step. The patch and batch size pairs are updated to [(160^2,5), (192^2,4), (256^2,2), (320^2,1), (384^2,1)] at iterations [92K, 156K, 204K, 240K, 276K], starting with a patch size 128x128 and a batcj size of 8. For data augmentation, we use horizontal and vertical flips.

C. Experimental setup

Here we discuss how the experiments were developed. Note, that the experiments were conducted iteratively, where the results of one affected the next step in this pipeline.

1) Baseline

The first step was to establish a baseline with both Restormer [3] and PromptIR [4]. There are no results published with either of these models for the MIT-Adobe-5K dataset and our specific partition, so we trained both of them to establish the baseline.

2) Directly concatenating color naming

This variant was **only** tried with the Restormer [3]. By concatenating the color naming probability maps C to the input image we ensure whether the model is capable of extracting the perceptual information straightforwardly. We evaluated both having 11 and 6 color groups.

3) Incorporation of the CNE

We incorporated the CNEs in the pipeline of both models. We evaluated the three variants presented in Section III-B. We used the same setup and configurations for all the CNEs.

Basic CNE. Since Restormer and PromptIR are very similar models, with the primary difference being the inclusion of C in the prompt block for PromptIR, we conducted most of our experiments on Restormer to simplify the process. We explored four different variables: the combination of "switches" (1 to 6 in Figure 5), the activation function, the type of pooling (max or average), and the number of probability maps (6 or 11). For the switches, we considered the following specific combinations:

- **Encoder (E)**: Uses C throughout the entire encoder part of the network (switches 1 to 4 closed).
- **Encoder with Early Fusion (EEF)**: Uses C in the encoder with an early fusion approach (switches 1 to 3 closed).
- **Encoder with Late Fusion (ELF)**: Uses C in the encoder with a late fusion approach (switches 2 to 4 closed).
- **Decoder (D)**: Uses C in the decoder part of the network (switches 4 to 6 closed).
- **U part of the network (U)**: Uses C in most parts of the network, specifically in all transformer blocks except the first one (switches 2 to 6 closed).

The set of activation functions is specified in Section V. The best RCN combination is used to evaluate on PIRCN, also considering the use of C in the prompt blocks of the model.

Squeeze CNE. For this type of CNE we only explored the hyperparameter that is exclusive to it, which is γ (figure 4b). It controls how much we reduce the number of channels in the squeeze convolution. In the original Fire module [60], the activation function is always ReLU, so that is the one we used. For the switches and pooling, we used the best combination found with the Basic CNE. We explored its effect on both the RCN and PIRCN, considering what effect it had on the latter using C in the prompt block or not.

Inception CNE. This type of CNE has no specific hyperparameters to explore. In the original inception module [61], the activation function is also ReLU, so that is the one we used here. Again, for the switches and pooling we used the best combination found with the Basic CNE and we explored its effect on both the RCN and PIRCN, considering what effect it had on the latter using C in the prompt block or not.

4) Incorporation of the backbone

After finding the best combination of the CNE for both RCN and PIRCN, we considered the addition of the backbone in the pipeline. The goal was to standardize the input image into a canonical latent space before extracting the color naming probability maps from it. This way, the color maps will represent more accurately the colors in the images.

Untrained backbone. In this case, we trained both the backbone and the main network end-to-end, making use of the loss function presented in Equation 1. We explored its effects with the best combinations found for both RCN and PIRCN in the previous section.

Pretrained and frozen backbone. We also considered a two-step training scheme. First, we train the backbone and, then, with its weights frozen, we train the rest of the architecture. We explored this variant using the best combination of CNE and hyperparameters.

Unfrozen backbone. This approach follows the previous one except that we add a third step to further refine the architecture weights altogether. During this third step the model parameters have reached a considerable minimum, we reduce the learning rate to $3e^{-5}$ that would be gradually reduced to $1e^{-7}$.

To do an ablation study, we also experimented with taking these approaches with the original models, Restormer [3] and PromptIR [4], introducing the backbone at the beginning of their pipeline. This way, we could explore the effects of color naming and the backbone separately.

TABLE I: Performance metrics comparison of baseline models, Restormer [3] and PromptIR [4], in our partition of the MIT-Adobe-5K dataset. Mean for all images.

Model	**Evaluation metrics**				
	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
Restormer [3] (Baseline)	21.75	0.838	0.0861	8.25	11.29
PromptIR [4] (Baseline)	21.78	0.834	0.0865	8.24	11.38

TABLE II: Impact of concatenating color naming probability maps to the image on Restormer [3] performance metrics. N is the number of probability maps. Mean for all images.

Model	**Color Naming**	**Metrics**				
	N	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
Restormer [3] (Baseline)	-	**21.75-**	**0.838-**	**0.0861-**	**8.25-**	**11.29-**
Restormer [3] w/ 9 Inp. Ch.	6	21.58	0.834	0.0904	8.38	11.52
Restormer [3] w/ 14 Inp. Ch.	11	21.68	**0.838-**	0.0904	8.32	11.42

D. Evaluation metrics

To evaluate and compare the performance of the baseline models with our method, we used the following metrics: PSNR, SSIM, LPIPS [65], ΔE_{00} and ΔE_{ab}. Since our approach is based on color, we consider keenly the results of the two latter metrics; ΔE_{00} and ΔE_{ab} indicate the difference in color between the ground truth image and the output image.

V. Results

In this section, we report the results of all the experiments presented in Section IV following the same order. First, we aim to establish the baseline performance and the contributions of each module. We refer to the different models and variations using the column **Model** in all the tables. The corresponding name references the changes with respect to the best combination of the previous experiment, dubbed best until now (BUN). All results are obtained with the validation set of our partition of the MIT-Adobe-5K dataset.

A. Baseline results

As mentioned in Section IV, neither Restormer [3] nor PromptIR [4] were conceived for image enhancement, and they were never tested in the MIT-Adobe-FiveK dataset [5]. For that reason, we established a baseline for both models' performance with the dataset and our partition of the data. We show the results in Table I.

From now on, these results appear at the beginning of each table. We highlight the best metrics in bold. We also use the following symbols: (-) to indicate that it is the best metric until now and (↑) and (↓) to indicate what models have surpassed the performance of the best model until the moment.

B. Concatenating color naming

We concatenated the probability maps to the input image, creating a 9 or 14 input tensor, which modified the number of input channels in the Restormer [3].

We present the results in Table II. Concatenating the color naming maps directly to the input image did not improve the performance of the baseline Restormer [3]. The model is incapable of directly using the color naming information to improve the enhancing process.

C. Incorporation of the CNE

In this subsection, we look at how incorporating the CNE module affected the performance of the models. To do so, we first identified the best hyperparameter combination for the Basic CNE on the Restormer [3]. Then, we applied this best combination to PromptIR [4] with the corresponding prompt block. Finally, we tried the Squeeze and Inception CNEs on both models.

1) Basic CNE

We first explored the number of color names and switch combinations to find the best-performing option. In this initial set of experiments, we only explored using them in one part of the network, either the encoder or the decoder, using the E, EEF, ELF, and D combinations. We used our first approach (Figure 5a) for the MDTA module, without activation function and average pooling at the end. In Table III we show the results. In parenthesis in the Model column, we specify the changes in the CNE.

TABLE III: Performance impact of probability maps (N) and the use of color naming (combinations of switches, figure 5) in our RCN. Mean for all images. R is an acronym for Restormer.

Model	CNE details						Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
R. [3] (Baseline)	-	-	-	-	-	-	21.75-	**0.838-**	**0.0861-**	8.25-	**11.29-**
RCN (N=6, E)	Basic	6	(a)	**E**	Identity	Avg	21.72	0.837	0.0878	8.30	11.40
RCN (N=6, EEF)	Basic	6	(a)	**EEF**	Identity	Avg	21.73	0.832	0.0892	8.28	11.38
RCN (N=6, ELF)	Basic	6	(a)	**ELF**	Identity	Avg	**21.81**↑	0.833	0.0896	**8.20**↓	11.30
RCN (N=6, D)	Basic	6	(a)	**D**	Identity	Avg	21.68	0.835	0.0873	8.29	11.42
RCN (N=11, E)	Basic	11	(a)	**E**	Identity	Avg	21.66	0.834	0.0872	8.27	11.37
RCN (N=11, EEF)	Basic	11	(a)	**EEF**	Identity	Avg	21.59	0.831	0.0889	8.39	11.50
RCN (N=11, ELF)	Basic	11	(a)	**ELF**	Identity	Avg	21.65	0.836	0.0888	8.29	11.32
RCN (N=11, D)	Basic	11	(a)	**D**	Identity	Avg	21.72	0.832	0.0881	8.28	11.37

TABLE IV: Performance impact the MDTA module, the activation function and the pooling at the end in the Basic CNE. and the use of color naming (combinations of switches, figure 5) in our RCN. Mean for all images. R. is an acronym for Restormer.

Model	CNE details						Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
R. [3] (Baseline)	-	-	-	-	-	-	21.75	**0.838-**	**0.0861-**	8.25	11.29-
RCN (BUN)	Basic	6	(a)	ELF	Identity	Avg	21.81-	0.833	0.0896	8.20-	11.30
RCN (MDTA b)	Basic	6	**(b)**	ELF	Identity	Avg	21.66	0.836	0.0875	8.30	11.35
RCN (MDTA c)	Basic	6	**(c)**	ELF	Identity	Avg	21.69	0.834	0.0894	8.30	11.38
RCN (MDTA d)	Basic	6	**(d)**	ELF	Identity	Avg	21.82↑	0.835	0.0884	8.21	11.30
RCN (Act. GELU)	Basic	6	(a)	ELF	**GELU**	Avg	21.75	0.831	0.0906	8.27	11.36
RCN (Act. PReLU)	Basic	6	(a)	ELF	**PReLU**	Avg	21.73	0.835	0.0879	8.29	11.40
RCN (Act. ReLU)	Basic	6	(a)	ELF	**ReLU**	Avg	21.80	0.836	0.0874	8.22	11.31
RCN (Max pool)	Basic	6	(a)	ELF	Identity	**Max**	21.76	0.835	0.0876	8.20-	11.28↓
RCN (Act. GELU, MPool)	Basic	6	(a)	ELF	**GELU**	**Max**	21.67	0.833	0.0883	8.34	11.45
RCN (Act. ReLU, MPool)	Basic	6	(a)	ELF	**ReLU**	**Max**	**21.95**↑	0.837	0.0863	**8.03**↓	**11.10**↓
RCN (U, Act. ReLU, MPool)	Basic	6	(a)	**U**	**ReLU**	**Max**	21.76	0.833	0.0880	8.31	11.43
RCN (MDTA d, Act. ReLU, MPool)	Basic	6	**(d)**	ELF	**ReLU**	**Max**	21.66	0.836	0.0875	8.30	11.33

In Table III, there is one combination that stands out. We can conclude that the RCN (N=6, ELF) is better than any other combination. It improves the performance on both PSNR and ΔE_{00}, while it is the one that gets closer to the baseline in ΔE_{ab}. It is the only experiment where we saw an improvement in some of the metrics. For that reason, for the next iteration of experiments we chose this combination of parameters as the starting point.

Thereafter, we explored the effects of the attention (MDTA) approach — see Figure 5 a)-d) —, the activation function, and the pooling type. First, we evaluated the modifications independently and, then, we combined them. We present the results in Table IV.

In the first section of Table IV, MDTA-d is the variation that performed best. In the second section, ReLU outperformed the other activation functions. In the third section, the combination of ReLU and Max pooling improved the baseline results.

Finally, we considered trying the combination of ReLU and Max pooling in another two scenarios. The first one was by changing the combination of switches to U type, and the second one was by trying it with the (d) version of the MDTA, as RCN (MDTA d) managed to improved some metrics. However, neither of them yielded better results than the ReLU and Max pooling combination.

Then, we experimented with the effects of the Basic CNE in PIRCN instead of the RCN. In Table V we show the results. We chose to keep the best parameters until now for the Basic CNE, which consist of using only 6 probability maps (N=6), ReLU as the activation function and Max pooling at the end. We explored introducing the C features in different ways. First, only introducing them in the transformer blocks (ELF), just as we did with RCN, and letting the original prompt blocks of the PromptIR [4] act by themselves with the new features. Second, only introducing them in our version of the prompt block (both versions, as in figures 6a and 6b). Third, trying both things at the same time.

PIRCN (ELF) did not manage to improve any of the metrics. With PIRCN (PB a) and PIRCN (PB b), we managed to improve SSIM and ΔE_{ab} in both cases, with PIRCN (PB a) getting a lower value in the latter one. In terms of PSNR and LPIPS, PIRCN (PB a) performs slightly better than PIRCN (PB b), and PIRCN (ELF) performs better than both. Since the difference is barely noticeable between PIRCN (PB a) and PIRCN (PB b), we decided to stick with PIRCN (PB b) because we use C in the PGM, which generates a prompt based on them, that will interact in the PIM with the features that come from the transformer blocks F. This way, we can enrich C with PGM, since they are extracted with our simpler CNE, whereas F are extracted with the more complex transformer blocks.

TABLE V: Performance impact of the use of color naming in the transformer blocks (Use of C) and the prompt block (PB) by separate in our PIRCN. Mean for all images. P. IR is an acronym for PromptIR.

Model	CNE details							Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PB	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
P. IR [4] (Baseline)	-	-	-	-	-	-	-	**21.78-**	0.834-	**0.0865-**	8.24-	11.38-
PIRCN (ELF)	Basic	6	(a)	ELF	ReLU	Max	-	21.77	0.833	0.0878	8.31	11.42
PIRCN (PB a)	Basic	6	(a)	-	ReLU	Max	(a)	21.72	**0.836**↑	0.0866	8.24-	**11.33**↓
PIRCN (PB b)	Basic	6	(a)	-	ReLU	Max	(b)	21.71	**0.836**↑	0.0878	8.24-	11.36↓
PIRCN (ELF, PB b)	Basic	6	(a)	ELF	ReLU	Max	(b)	21.76	**0.836**↑	0.0870	**8.23**↓	**11.33**↓

TABLE VI: Performance impact of the Squeeze CNE in our RCN.

Model	CNE details						Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
R. [3] (Baseline)	-	-	-	-	-	-	21.75	**0.838-**	**0.0861-**	8.25	11.29
RCN (BUN)	Basic	6	(a)	ELF	ReLU	Max	**21.95-**	0.837	0.0863	**8.03-**	**11.10-**
RCN (Squeeze $\gamma = 6$)	**Squeeze** ($\gamma = 6$)	6	(a)	ELF	ReLU	Max	21.87	0.833	0.0898	8.25	11.37
RCN (Squeeze $\gamma = 8$)	**Squeeze** ($\gamma = 8$)	6	(a)	ELF	ReLU	Max	21.68	0.837	0.0875	8.30	11.43

Finally, we evaluated PIRCN (ELF, PB b). The results were still very similar between them, managing a slight improvement. We did not manage to improve the results with PIRCN as we did with RCN using the Basic CNE.

2) Squeeze CNE

After the Basic CNE, we explored the effects of the Squeeze CNE. In the original implementation of the fire module by Iandola et al. [60], ReLU is used as an activation function. Because of that, and because our results with Basic CNE using the same function, we used also ReLU for the Squeeze CNE. For the same reason, we used Max pooling as the pooling type. Thus, there is only one hyperparameter left to explore for the Squeeze CNE, γ. As we can see in figure 4b, it controls how much we reduce the number of input channels by the first convolution in the block. The higher the value, the more it is reduced. Looking at how SqueezeNet [60] is built, we can see that this parameter γ is usually between 4 and 8. Because the number of channels throughout our networks are always divisible by these values, we decided to try the two approaches that reduce the channels more aggressively: 6 and 8.

The results for RCN with the Squeeze CNE are presented in table VI. Neither RCN (Squeeze $\gamma = 6$) nor RCN (Squeeze $\gamma = 8$) managed to improve the results that we achieved with the Basic CNE, now RCN (BUN). The only thing worth mentioning is that RCN (Squeeze $\gamma = 6$) has a higher PSNR than Restormer [3] (Baseline) and the same value for ΔE_{00}.

The results for PIRCN with the Squeeze CNE are shown in table VII. First, we tried with both γ values while using C, the color naming features, in the prompt block as in figure 6b. PIRCN (Squeeze $\gamma = 8$) managed to improve all metrics from both PromptIR [4] (Baseline) and PIRCN (BUN), the highest difference being in the ΔEs. Just as we did with the experiments in table V, we also considered the use of C only in the encoder part of the network, letting the original prompt blocks work. However, in PIRCN (Squeeze $\gamma = 8$, no PB), the results were worse than PIRCN (Squeeze $\gamma = 8$) for all cases.

3) Inception CNE

Lastly, we experimented with the Inception CNE. We used ReLU as the activation function –again, in the original Inception implementation [61] is the one they use– and Max pooling for the same reasons as with the Squeeze CNE, fixing these hyperparameters. In this case there are no other hyperparameters to explore, so we compare the use of the Inception CNE with the baselines and the BUN models.

The results for RCN (Inception) are in table VIII. It improves Restormer [3] (Baseline) in PSNR, ΔE_{00} and ΔE_{ab}, significantly in the first metric, almost as much as the RCN (BUN). However, RCN (BUN) is still the best RCN.

The results for PIRCN with the Inception CNE are in table IX. Just as we did with both the Basic and Squeeze CNEs, we explored the impact of the Inception CNE by using it in the prompt block (PIRCN (Inception)) and not using it (PIRCN (Inception, no PB)). PIRCN (Inception, no PB) did not manage to improve the results for PromptIR [4] (Baseline), getting slightly worse results in all metrics. PIRCN (Inception), however, managed to reach the same PSNR as PIRCN (BUN), lagging a little behind in the rest of the metrics. However, all of the metrics are better than in PromptIR [4] (Baseline).

With the results until now, we can clearly see that the use of color naming is having a positive effect on both models. We have found that using 6 color probability maps ($N = 6$) works better than using 11, that combining the color naming features with the original query in the MDTA module as in figure 5a —i.e. when is used as part of the Query—is the better approach for the attention mechanism and that using the color naming features C with a late fusion approach in the encoder (ELF) is the best way to introduce them in the pipeline of both models. For RCN, the Basic CNE with the classic combination of ReLU as an activation function and Max pooling (RCN (Act. ReLU, Max pool) in table IV) has proved to be the best set of hyperparameters, having results showing a clear advantage with respect to the rest. For PIRCN, on the contrary, the Squeeze

TABLE VII: Performance impact of the Squeeze CNE in our PIRCN. Mean for all images. P. IR is an acronym for PromptIR.

Model	CNE details							Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PB	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
P. IR [4] (Baseline)	-	-	-	-	-	-	-	21.78-	0.834	0.0865-	8.24	11.38
PIRCN (BUN)	Basic	6	(a)	ELF	ReLU	Max	(b)	21.76	0.836-	0.0870	8.23-	11.33-
PIRCN (Squeeze $\gamma = 6$)	**Squeeze ($\gamma = 6$)**	6	(a)	ELF	ReLU	Max	(b)	21.76	0.836	0.0875	8.28	11.38
PIRCN (Squeeze $\gamma = 8$)	**Squeeze ($\gamma = 8$)**	6	(a)	ELF	ReLU	Max	(b)	**21.87**↑	**0.839**↑	**0.0858**↓	**8.09**↓	**11.13**↓
PIRCN (Squeeze $\gamma = 8$, no PB)	**Squeeze ($\gamma = 8$)**	6	(a)	ELF	ReLU	Max	**-**	21.75	0.836	0.0868	8.24	11.36

TABLE VIII: Performance impact of the Inception CNE in our PIRCN. Mean for all images. R. is an acronyms for Restormer.

Model	CNE details						Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
R. [3] (Baseline)	-	-	-	-	-	-	21.75	**0.838-**	**0.0861-**	8.25	11.29
RCN (BUN)	Basic	6	(a)	ELF	ReLU	Max	**21.95-**	0.837	0.0863	**8.03-**	**11.10-**
RCN (Inception)	**Inception**	6	(a)	ELF	ReLU	Max	21.91	0.834	0.0887	8.14	11.21

TABLE IX: Performance impact of the Inception CNE in our RCN. Mean for all images. P. IR is an acronym for PromptIR.

Model	CNE details							Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PB	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
P. IR [4] (Baseline)	-	-	-	-	-	-	-	21.78	0.834	0.0865	8.24	11.38
PIRCN (BUN)	**Squeeze ($\gamma = 8$)**	6	(a)	ELF	ReLU	Max	(b)	**21.87-**	**0.839-**	**0.0858-**	**8.09-**	**11.13-**
PIRCN (Inception)	**Inception**	6	(a)	ELF	ReLU	Max	(b)	**21.87-**	0.836	0.0863	8.15	11.27
PIRCN (Inception, no PB)	**Inception**	6	(a)	ELF	ReLU	Max	**-**	21.73	0.834	0.0868	8.25	11.39

CNE with a channel reduction factor of $\gamma = 8$ and the same combination of activation function and pooling (PIRCN (Squeeze $\gamma = 8$) in table VII) works better, with the advantage not being as clear as with RCN, since the Inception CNE (PIRCN (Inception) in table IX) also managed to improve the performance of the baseline model, although not as much.

D. Incorporation of the backbone

As we have mentioned before, the type of images we are dealing with are usually low-light or under-exposed images, which can affect our approach with color naming. In this section, we see the results of incorporating a light and fast backbone network in the pipeline, with the intention of pre-processing the images, standardizing them into a canonical space where the extraction of the probability maps would be more accurate. With the purpose of doing an ablation study, we also incorporated it in the baseline models, to see, separately, the effects of the color naming features and the backbone itself.

1) Untrained backbone

We explored first the effects of introducing the untrained backbone in both of the models. This means training both the backbone and the main network at the same time, with the loss in equation 1.

In the second section of rows in table X, we see the impact the untrained backbone makes on Restormer [3] (Baseline) and our RCN (BUN). The incorporation of the backbone on both Restormer [3] (Baseline, untrained BB) and RCN (BUN, untrained BB) had a positive effect on the PSNR value and the ΔEs. In neither of the cases it manages to improve the SSIM and LPIPS metrics.

For PIRCN, the results with the different types of CNE were not as clear as with RCN. For that reason, we decided to introduce the backbone in the PIRCN trying the three different types of CNE. We see the impact of the untrained backbone on PromptIR [4] (Baseline) and our different PIRCNs in the second section of rows of the table XI. The results show a very similar behaviour as with introduction of the backbone in both Restormer [3] and RCN. Most of the impact appears to be in the PSNR and the ΔEs. If we compare each model with their version with the untrained backbone incorporated, these three metrics are improved in all cases.

The impact of the backbone is clearly positive. It improves the performance of both the baseline models and the ones where we incorporated color naming. Furthermore, if we compare Tables XI-XII with Tables V-VI we can see that adding Color Naming improves similarly (RCN) or more (PIRCN) when using the backbone. We can conclude, therefore, that using the backbone for the standardization of the input image into a canonical latent space, getting a version of the image that is closest to the expert-retouched one where the colors will be more accurately represented, helps at the time of using the color naming probability maps as priors.

2) Pretrained and frozen backbone

We also considered training the backbone first in the same manner as the rest of the models and using it as an independent module, with its weights frozen. This way, we have an independent backbone that learns the task at hand, whose more

TABLE X: Performance impact of the backbone (untrained and pretrained) in Restormer (R.) [3] (Baseline) and RCN (BUN). Mean for all images.

Model	CNE details						BB	Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling		PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
R. [3] (Baseline)	-	-	-	-	-	-	No BB	21.75	0.838-	0.0861-	8.25	11.29
RCN (BUN)	Basic	6	(a)	ELF	ReLU	Max	No BB	21.95-	0.837	0.0863	8.03-	11.10-
R. [3] (Baseline, untrained BB)	-	-	-	-	-	-	**BB (untrained)**	21.96↑	0.836	0.0873	8.14	11.20
RCN (BUN, untrained BB)	Basic	6	(a)	ELF	ReLU	Max	**BB (untrained)**	22.11↑	0.835	0.0867	8.06	11.07↓
R. [3] (Baseline, pretrained BB)	-	-	-	-	-	-	**BB (pretrained)**	22.19↑	**0.841**↑	0.0871	7.87↓	10.90↓
RCN (BUN, pretrained BB)	Basic	6	(a)	ELF	ReLU	Max	**BB (pretrained)**	**22.32**↑	**0.841**↑	**0.0842**↓	**7.72**↓	**10.75**↓

TABLE XI: Performance impact of the backbone (untrained and pretrained) in PromptIR (P.IR) [4] (Baseline) and the best PIRCN with each type of CNE. Mean for all images.

Model	CNE details							BB	Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling	PB		PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
P. IR [4] (Baseline)	-	-	-	-	-	-	-	No BB	21.78	0.834	0.0865	8.24	11.38
PIRCN (Basic)	Basic	6	(a)	ELF	ReLU	Max	(b)	No BB	21.76	0.836	0.0870	8.23	11.33
PIRCN (Squeeze $\gamma = 8$, BUN)	Squeeze ($\gamma = 8$)	6	(a)	ELF	ReLU	Max	(b)	No BB	21.87-	0.839-	0.0858-	8.09-	11.13-
PIRCN (Inception)	Inception	6	(a)	ELF	ReLU	Max	(b)	No BB	21.87-	0.836	0.0863	8.15	11.27
P. IR [4] (Baseline, untrained BB)	-	-	-	-	-	-	-	**BB (untrained)**	22.01↑	0.837	0.0865	8.11	11.17
PIRCN (Basic, untrained BB)	Basic	6	(a)	ELF	ReLU	Max	(b)	**BB (untrained)**	22.13↑	0.838	0.0858-	7.96↓	10.96↓
PIRCN (Squeeze $\gamma = 8$, untrained BB)	Squeeze ($\gamma = 8$)	6	(a)	ELF	ReLU	Max	(b)	**BB (untrained)**	21.99↑	0.838	0.0865	8.05↓	11.08↓
PIRCN (Inception, untrained BB)	Inception	6	(a)	ELF	ReLU	Max	(b)	**BB (untrained)**	21.99↑	0.834	0.0862	8.14	11.22
P. IR [4] (Baseline, pretrained BB)	-	-	-	-	-	-	-	**BB (pretrained)**	22.02↑	0.840↑	0.0857↓	8.00↓	11.06↓
PIRCN (Basic, pretrained BB)	Basic	6	(a)	ELF	ReLU	Max	(b)	**BB (pretrained)**	22.06↑	0.839-	**0.0850**↓	7.97↓	11.01↓
PIRCN (Squeeze $\gamma = 8$, pretrained BB)	Squeeze ($\gamma = 8$)	6	(a)	ELF	ReLU	Max	(b)	**BB (pretrained)**	**22.19**↑	0.840↑	0.0851↓	**7.84**↓	**10.88**↓
PIRCN (Inception, pretrained BB)	Inception	6	(a)	ELF	ReLU	Max	(b)	**BB (pretrained)**	22.08↑	**0.841**↑	**0.0850**↓	7.92↓	10.98↓

TABLE XII: Performance impact of the unfrozen backbone in RCN (BUN). Mean for all images.

Model	CNE details						BB	Evaluation metrics				
	Type	N	MDTA	Use of C	Activation	Pooling		PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
RCN (BUN)	Basic	6	(a)	ELF	ReLU	Max	No BB	21.95-	0.837	0.0863	8.03-	11.10-
RCN (BUN, untrained BB)	Basic	6	(a)	ELF	ReLU	Max	BB (untrained)	22.11	0.835	0.0867	8.06	11.07
RCN (BUN, pretrained BB)	Basic	6	(a)	ELF	ReLU	Max	BB (pretrained)	**22.32-**	**0.841-**	**0.0842-**	**7.72-**	**10.75-**
RCN (BUN, unfrozen BB)	Basic	6	(a)	ELF	ReLU	Max	**BB (unfrozen)**	22.29	0.838	0.0859	7.76	10.78

color-accurate outputs can be used to extract the color naming probability maps and as inputs for the main network.

The results of incorporating the pretrained backbone to Restormer [3] (Baseline) and RCN (BUN) are in the third section of rows of table X. We observe a very similar change as when the backbone was untrained, but greater. Both the baseline and the RCN improve significantly in PSNR and the ΔEs, RCN (BUN, pretrained BB) being better than R. [3] (Baseline, pretrained BB). Both of them reach the same SSIM, which improves the best result we had achieved until now, and RCN (BUN, pretrained BB) gets the best value for LPIPS until now. Each of them also improved their respective versions with the untrained backbone in all metrics.

The results of incorporating the pretrained backbone to P. IR [4] (Baseline) and the different versions of PIRCN are in the third section of rows of table XI. In all cases, there is an improvement in all metrics with respect to their versions without any backbone. We also see that the models that use color naming have a better performance than the one that does not (P. IR [4] (Baseline, pretrained BB)). However, if we look at their versions with the untrained backbones, not all improve. P. IR [4] (Baseline, pretrained BB) has a slightly higher PSNR than P. IR [4] (Baseline, untrained BB), but the rest of the metrics improve the performance significantly. PIRCN (Basic, pretrained), however, has worse results than PIRCN (Basic, untrained) in PSNR, ΔE_{00} and ΔE_{ab}. The greatest impact the pretrained backbone had was on PIRCN (Squeeze $\gamma = 8$, BUN), which, if we do not consider the models with the untrained backbone incorporated, was the best model we had. All metrics are improved considerably. PIRCN (Squeeze $\gamma = 8$, pretrained BB) is the PIRCN model that has the best performance until now. The last one, PIRCN (Inception, pretrained BB), has also an improvement with respect to PIRCN (Inception, untrained BB), but it is not as significant as with PIRCN (Squeeze $\gamma = 8$, pretrained BB).

3) Unfrozen backbone

The last thing we considered doing with the backbone in the pipeline of the models was to take the model that was trained with the frozen backbone, unfreeze the backbone, and train again the whole thing with a reduced learning rate, considering that we would have already achieved a good minimum for the loss function.

This experiment was only conducted for the RCN (BUN) model, and the results can be seen in table XII. Unfreezing the backbone and training the whole thing again in RCN (BUN, unfrozen BB) leads to a slight worsening in all metrics with respect to RCN (BUN, pretrained BB).

TABLE XIII: Performance metrics comparison of baseline models and best versions of RCN and PIRCN on the test set of our partition of the MIT-Adobe-5K dataset. Mean for all images. P. IR is an acronym for PromptIR.

Model	Evaluation metrics				
	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
Restormer [3] (Baseline)	24.06-	0.862-	0.0681-	6.87-	9.50-
RCN (BUN)	**24.70**↑	**0.873**↑	**0.0653**↓	**6.41**↓	**8.88**↓
PromptIR [4] (Baseline)	23.92-	0.861-	0.0680-	6.96-	9.64-
PIRCN (BUN)	24.40↑	0.870↑	0.0658↓	6.60↓	9.11↓

TABLE XIV: Performance metrics comparison with other state-of-the-art methods with the test set of the UEGAN partition in the MIT-Adobe-5K dataset.

Model	Evaluation metrics				
	PSNR ↑	SSIM ↑	LPIPS ↓	ΔE_{00} ↓	ΔE_{ab} ↓
3DLUT [31]	25.29	0.923	0.043	6.76	7.55
AdaInt [33]	25.49	0.926	0.041	6.69	7.47
NamedCurves [2]	**25.59**	**0.936**	**0.038**	**6.07**	**7.40**
Restormer [3]	24.62	0.880	0.096	6.42	8.70
PromptIR [4]	24.35	0.879	0.079	6.55	8.87
RCN, pretrained and frozen backbone (Ours)	24.81	0.879	0.101	6.31	8.52
PIRCN, pretrained and frozen backbone (Ours)	24.44	0.879	0.084	6.48	8.76
RCN, pretrained and unfrozen backbone (Ours)	24.91	0.878	0.095	6.29	8.50
PIRCN, pretrained and unfrozen backbone (Ours)	24.95	0.881	0.082	6.22	8.43

E. Test results

In this section, we discuss the results in our test set from the MIT-Adobe-5K dataset [5]. We compare the baseline models with the best versions of RCN and PIRCN. Both RCN (BUN) and PIRCN (BUN) include the pretrained backbone in their pipeline, the use of 6 color naming probability maps, version (a) of the MDTA module (Figure 5a), a late fusion approach in the encoder (ELF), the ReLU activation and Max pooling at the end for the CNE. The differences are that RCN (BUN) uses the Basic CNE, and PIRCN (BUN) uses the Squeeze CNE with $\gamma = 8$, also using the color naming features in the version (b) of the prompt block (Figure 6b).

The results of the test set are in Table XIII. Our approaches outperform the corresponding baseline by 0.5 dBs in PSNR in both RCN (BUN) and PIRCN (BUN), the former having the best performance.

F. Comparison with state-of-the-art methods

To compare with other state-of-the-art methods, it is necessary to train our models with the proper data split. Since we have been using the images that were preprocessed with the methods of the UEGAN [35] partition, we will use this partition to train our models, which is 4500 images to train and 500 to test from the MIT-Adobe-5K dataset. This way, we also make use of a bigger set of images to train, which is always positive for data-hungry models such as the ones we have used. Another important change has to do with the LPIPS [65] metric. LPIPS compute the similarity between the activations of two image patches for some pre-defined network. Until now, in all of the previous tables, that network was AlexNet [66]. However, to be able to compare faithfully to other state-of-the-art methods, we have to use VGG-16 [67].

With respect to the models, we trained a backbone, Restormer [3], PromptIR [4], and the best versions of RCN and PIRCN we saw on the previous section with this split. This way, we can do a faithful comparison between every model we have used in this study with the state-of-the-art. We will also use this section for the qualitative results, looking at the restored images individually.

Table XIV presents the comparison with state-of-the-art methods using the UEGAN partition of the MIT-Adobe-5K dataset. Our method had a better performance than both baseline models, Restormer [3] and PromptIR [4], although the advantage is not as drastic as it was with our partition of the dataset. This is more noticeable in the SSIM and LPIPS metrics, where the baseline models managed a better performance. We also did a training with a pretrained and unfrozen backbone, to see if with this partition the frozen backbone was holding back the learning of the model. The results for this training are also in table XIV. In both cases, the performance is better with the unfrozen backbone that with respect to the frozen one. This is different from the results obtained when using our partition. We hypothesize that this is probably given by the fact that this partition has a larger number of images for training.

This said, the results are still far from other state-of-the-art methods. We only get close in the ΔE_{00} metric, improving the results of 3DLUT [31] and AdaInt [33], but not reaching NamedCurves [2].

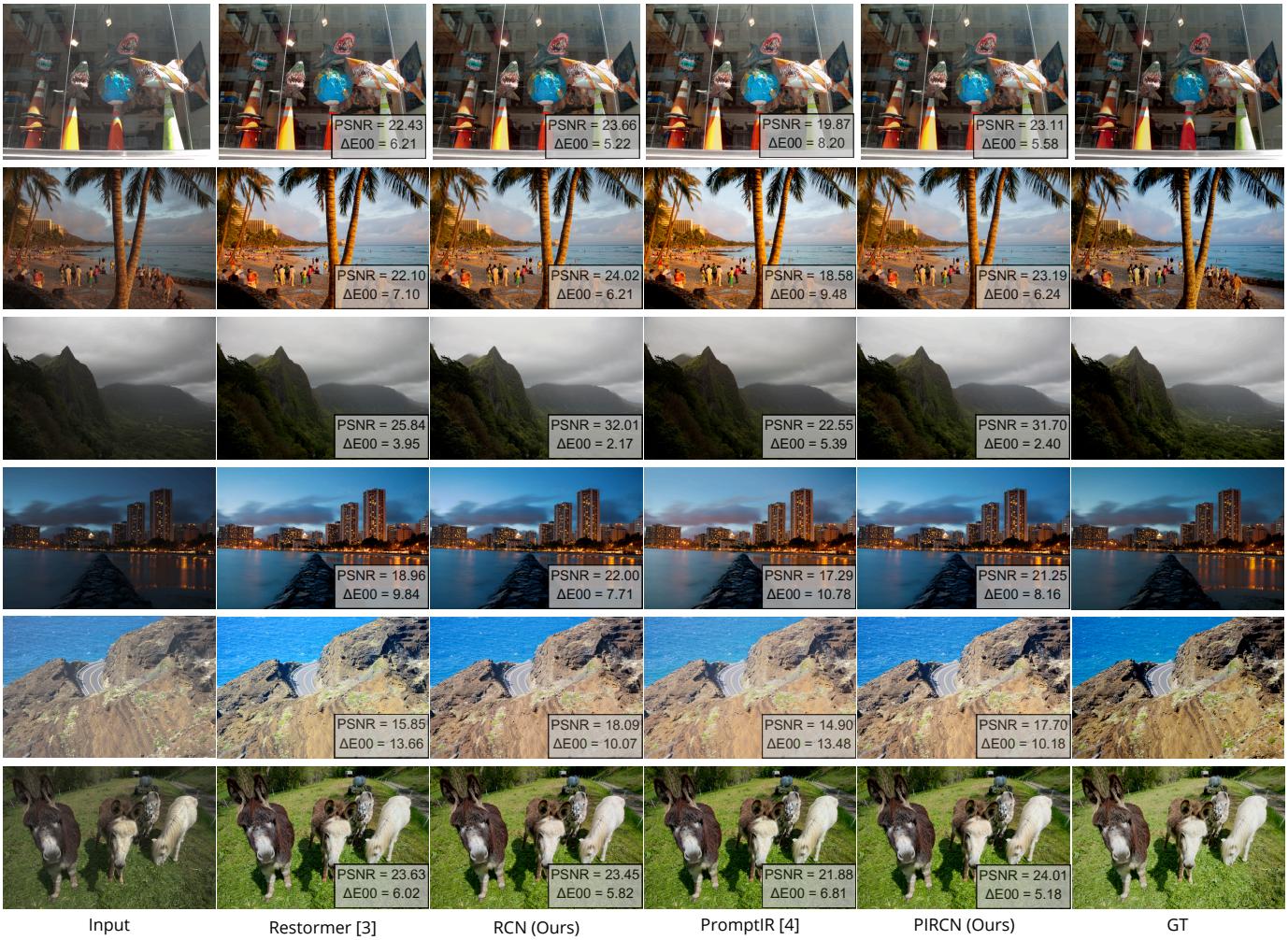


Fig. 7: Qualitative comparisons on the MIT-5K. We show results from Restormer [3], PromptIR [4], and our best two combinations of the corresponding methods. On the bottom-right of each image, we display the PSNR and ΔE00.

G. Qualitative results

Figures 7, 8, 9, and 10 present the qualitative results of our experiments. In Figure 7, we compare Restormer [3], PromptIR [4], and our two best method combinations. Incorporating color naming in the pipeline aids effectively in the enhancement of images in a wide variety of scenes. Figure 8 highlights a comparison between our best combinations and the methods AdaInt [33] and NamedCurves [2]. We can see that, in some instances, our models manage to improve the state-of-the-art performance, but it is not the norm. Figures 9 and 10 display additional results along with the corresponding ΔE2000 error maps. The results demonstrate that our approaches effectively remove color casts from the input images and enhance fine-grained details, yielding results closer to expert-retouched images than the baseline models.

VI. Conclusions

In our study, we explored the integration of color naming into transformer-based image restoration models, namely the Restormer [3] and the PromptIR [4] models. We demonstrated improvement over the baseline models, indicating that the integration of color naming into such architectures is promising.

This said, we did not achieve state-of-the-art performance. We believe that this is due to the nature of the models we used, since they were originally designed to perform image restoration and we repurposed them to perform image enhancement. Image restoration models are transformer-based and U-net-like architectures, which excel at tasks like image restoration by learning generalizable patterns from large datasets. They rely on learning features from the data itself, without incorporating more explicit, task-specific constraints. In contrast, state-of-the-art methods for image enhancement often leverage more explicit approaches, such as color curves [2] or look-up tables [31, 32, 33], which directly manipulate color and tone based on learned parameters that more closely model the physical properties of the process of image retouching. The latter approach can offer more precise control over the desired image output, and this precision likely contributes to their superior performance for the problem of image enhancement.

Fig. 8: Qualitative comparisons on the MIT-5K UEGAN split. We show results from AdaInt [33], NamedCurves [2], and our best two combinations of RCN and PIRCN. On the bottom-right of each image, we display the PSNR and ΔE00.

There is still room for experiments within the context of our study. More complex CNEs could be developed, and different ways to integrate the color naming features C in their pipelines could be tried. Due to time restrictions, we were not able to perform a hyperparameter optimization on our models; we still used the Restormer [3] configuration, and experimenting with it we could have get closer to state-of-the-art results.

Future work outside the context of our study may focus on developing new ways to integrate the use of color naming in both deep learning-based approaches, just as we did, and in more explicit, parametric methods for color and tone adjustment, as Serrano et al. [2] did. Hybrid models, combining the strength of both approaches, could also be developed, potentially bridging the gap between learned feature extraction and physically informed retouching techniques.

ACKNOWLEDGMENT

I would like to express my gratitude to Javier Vazquez Corral and David Serrano Lozano for their guidance, feedback and continuous support throughout the development of this thesis. Their expertise has been essential to the progress and completion of this work.

I would also like to thank the Computer Vision Center (CVC) for providing the necessary facilities and resources that made this study possible. The opportunity to work in such a collaborative environment lead to the success of this project.

REFERENCES

[1] B. Berlin and P. Kay, *Basic Color Terms: Their Universality and Evolution*. Berkeley, CA: University of California Press, 1969.
[2] D. Serrano-Lozano, L. Herranz, M. S. Brown, and J. Vazquez-Corral, "Namedcurves: Learned image enhancement via color naming," *arXiv preprint arXiv:2407.09892*, 2024.
[3] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5728–5739, 2022.
[4] V. Potlapalli, S. W. Zamir, S. Khan, and F. Khan, "Promptir: Prompting for all-in-one image restoration," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

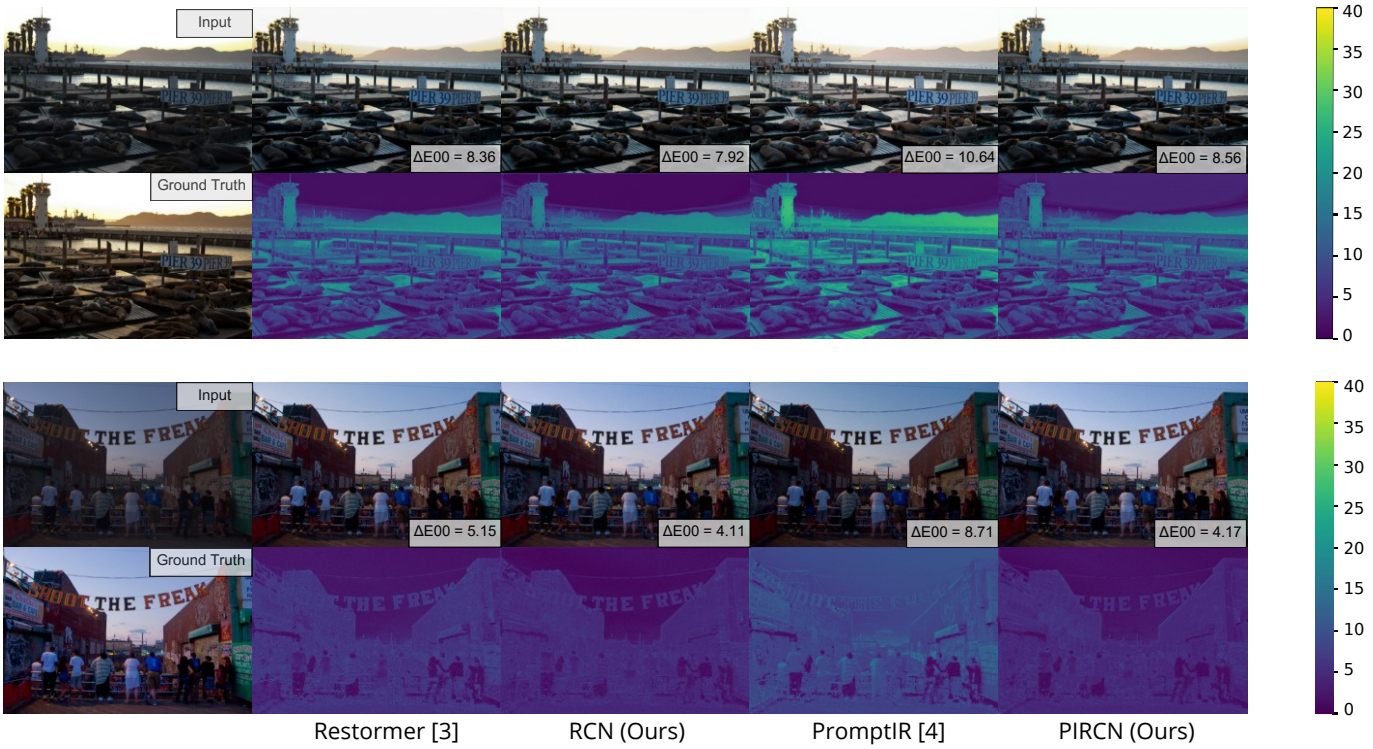


Fig. 9: Qualitative comparisons on the MIT-5K. We show results from Restormer [3], PromptIR [4], and our best two combinations of the corresponding methods. Below each result, we show the ΔE00 error map. On the bottom-right of each image, we display the ΔE00.

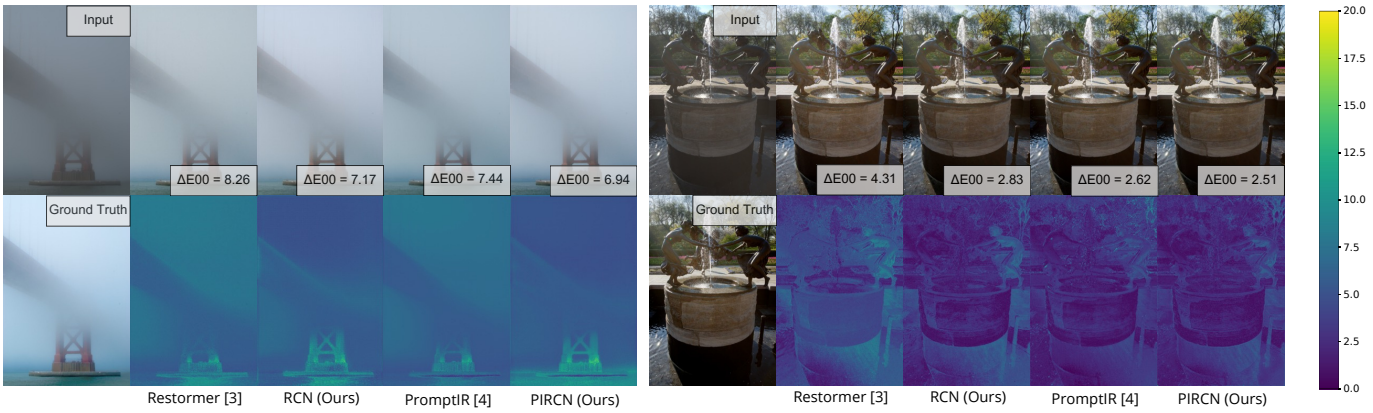


Fig. 10: Qualitative comparisons on the MIT-5K. We show results from Restormer [3], PromptIR [4], and our best two combinations of the corresponding methods. Below each result, we show the ΔE00 error map. On the bottom-right of each image, we display the ΔE00.

[5] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *CVPR 2011*, pp. 97–104, IEEE, 2011.

[6] M. Delbracio, D. Kelly, M. S. Brown, and P. Milanfar, "Mobile computational photography: A tour," *Annual Review of Vision Science*, vol. 7, pp. 571–604, 2021.

[7] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3929–3938, IEEE, 2017.

[8] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 773–782, IEEE, 2018.

[9] X. Pan, X. Zhan, B. Dai, D. Lin, C. C. Loy, and P. Luo, "Exploiting deep generative prior for versatile image restoration and manipulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7474–7489, 2021.

[10] B. Fei, Z. Lyu, L. Pan, J. Zhang, W. Yang, T. Luo, and B. Dai, "Generative diffusion prior for unified image restoration and enhancement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9935–9946, IEEE, 2023.

[11] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1833–1844, 2021.

[12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, IEEE, 2021.

[13] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general u-shaped transformer for image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17683–17693, 2022.

[14] L. Shen, Z. Yue, F. Feng, Q. Chen, S. Liu, and J. Ma, "Msr-net: Low-light image enhancement using deep convolutional network," *arXiv preprint arXiv:1711.02488*, 2017.

[15] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, "Zero-reference deep curve estimation for low-light image enhancement," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1780–1789, 2020.

[16] Z. Zhao, B. Xiong, L. Wang, Q. Ou, L. Yu, and F. Kuang, "Retinexdip: A unified deep framework for low-light image enhancement," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1076–1088, 2021.

[17] X. Yi, H. Xu, H. Zhang, L. Tang, and J. Ma, "Diff-retinex: Rethinking low-light image enhancement with a generative diffusion model," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12302–12311, 2023.

[18] T. Wang, K. Zhang, T. Shen, W. Luo, B. Stenger, and T. Lu, "Ultra-high-definition low-light image enhancement: A benchmark and transformer-based method," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 2654–2662, 2023.

[19] Y. Cai, H. Bian, J. Lin, H. Wang, R. Timofte, and Y. Zhang, "Retinexformer: One-stage retinex-based transformer for low-light image enhancement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12504–12513, 2023.

[20] Y. Wang, J. Zhang, Y. Cao, and Z. Wang, "A deep cnn method for underwater image enhancement," in *2017 IEEE international conference on image processing (ICIP)*, pp. 1382–1386, IEEE, 2017.

[21] Y. Wang, J. Guo, H. Gao, and H. Yue, "Uiecˆ 2-net: Cnn-based underwater image enhancement using two color space," *Signal Processing: Image Communication*, vol. 96, p. 116250, 2021.

[22] C. Fabbri, M. J. Islam, and J. Sattar, "Enhancing underwater imagery using generative adversarial networks," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 7159–7165, IEEE, 2018.

[23] Y. Guo, H. Li, and P. Zhuang, "Underwater image enhancement using a multiscale dense generative adversarial network," *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, pp. 862–870, 2019.

[24] Z. Huang, J. Li, Z. Hua, and L. Fan, "Underwater image enhancement via adaptive group attention-based multiscale cascade transformer," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–18, 2022.

[25] L. Peng, C. Zhu, and L. Bian, "U-shape transformer for underwater image enhancement," *IEEE Transactions on Image Processing*, 2023.

[26] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 2, pp. 1–15, 2016.

[27] R. Wang, Q. Zhang, C.-W. Fu, X. Shen, W.-S. Zheng, and J. Jia, "Underexposed photo enhancement using deep illumination estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6849–6857, 2019.

[28] S. Moran, P. Marza, S. McDonagh, S. Parisot, and G. Slabaugh, "Deeplpf: Deep local parametric filters for image enhancement," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12826–12835, 2020.

[29] S. Moran, S. McDonagh, and G. Slabaugh, "Curl: Neural curve layers for global image enhancement," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 9796–9803, IEEE, 2021.

[30] C. Li, C. Guo, S. Zhou, Q. Ai, R. Feng, and C. C. Loy, "Flexicurve: Flexible piecewise curves estimation for photo retouching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1092–1101, 2023.

[31] H. Zeng, J. Cai, L. Li, Z. Cao, and L. Zhang, "Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 4, pp. 2058–2073, 2020.

[32] T. Wang, Y. Li, J. Peng, Y. Ma, X. Wang, F. Song, and Y. Yan, "Real-time image enhancer via learnable spatial-aware 3d lookup tables," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2471–2480, 2021.

[33] C. Yang, M. Jin, X. Jia, Y. Xu, and Y. Chen, "Adaint: Learning adaptive intervals for 3d lookup tables on real-time image enhancement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17522–17531, 2022.

[34] Y.-S. Chen, Y.-C. Wang, M.-H. Kao, and Y.-Y. Chuang, "Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6306–6314, 2018.

[35] Z. Ni, W. Yang, S. Wang, L. Ma, and S. Kwong, "Towards unsupervised deep image enhancement with generative adversarial network," *IEEE Transactions on Image Processing*, vol. 29, pp. 9140–9151, 2020.

[36] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou, and Z. Wang, "Enlightengan: Deep light enhancement without paired supervision," *IEEE transactions on image processing*, vol. 30, pp. 2340–2349, 2021.

[37] Z. Zhang, Y. Jiang, J. Jiang, X. Wang, P. Luo, and J. Gu, "Star: A structure-aware lightweight transformer for real-time image enhancement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4106–4115, 2021.

[38] Z. Cui, K. Li, L. Gu, S. Su, P. Gao, Z. Jiang, Y. Qiao, and T. Harada, "You only need 90k parameters to adapt light: a light weight transformer for image enhancement and exposure correction," *arXiv preprint arXiv:2205.14871*, 2022.

[39] E. R. Heider, "Universals in color naming and memory.," *Journal of experimental psychology*, vol. 93, no. 1, p. 10, 1972.

[40] P. Kay, B. Berlin, L. Maffi, W. Merrifield, *et al.*, "Color naming across languages," *Color categories in thought and language*, vol. 21, no. 2, 1997.

[41] N. Zaslavsky, C. Kemp, N. Tishby, and T. Regier, "Color naming reflects both perceptual structure and communicative need," *Topics in cognitive science*, vol. 11, no. 1, pp. 207–219, 2019.

[42] T. Regier, P. Kay, and N. Khetarpal, "Color naming reflects optimal partitions of color space," *Proceedings of the National Academy of Sciences*, vol. 104, no. 4, pp. 1436–1441, 2007.

[43] E. Gibson, R. Futrell, J. Jara-Ettinger, K. Mahowald, L. Bergen, S. Ratnasingam, M. Gibson, S. T. Piantadosi, and B. R. Conway, "Color naming across languages reflects color use," *Proceedings of the National Academy of Sciences*, vol. 114, no. 40, pp. 10785–10790, 2017.

[44] N. Zaslavsky, C. Kemp, T. Regier, and N. Tishby, "Efficient compression in color naming and its evolution," *Proceedings of the National Academy of Sciences*, vol. 115, no. 31, pp. 7937–7942, 2018.

[45] M. H. Bornstein, "Color vision and color naming: a psychophysiological hypothesis of cultural difference.," *Psychological Bulletin*, vol. 80, no. 4, p. 257, 1973.

[46] M. H. Bornstein, "On the development of color naming in young children: Data and theory," *Brain and language*, vol. 26, no. 1, pp. 72–93, 1985.

[47] A. Mojsilovic, "A computational model for color naming and describing color composition of images," *IEEE Transactions on Image processing*, vol. 14, no. 5, pp. 690–699, 2005.

[48] R. Benavente, M. Vanrell, and R. Baldrich, "A data set for fuzzy colour naming," *Color Research & Application*, vol. 31, no. 1, pp. 48–56, 2006.

[49] R. Benavente, M. Vanrell, and R. Baldrich, "Parametric fuzzy sets for automatic color naming," *JOSA A*, vol. 25, no. 10, pp. 2582–2593, 2008.

[50] J. Van De Weijer, C. Schmid, J. Verbeek, and D. Larlus, "Learning color names for real-world applications," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1512–1523, 2009.

[51] D. Mylonas, L. MacDonald, and S. Wuerger, "Towards an online color naming model," in *Color and imaging conference*, vol. 18, pp. 140–144, Society of Imaging Science and Technology, 2010.

[52] A. Baronchelli, T. Gong, A. Puglisi, and V. Loreto, "Modeling the emergence of universality in color naming patterns," *Proceedings of the National Academy of Sciences*, vol. 107, no. 6, pp. 2403–2407, 2010.

[53] Z. Yuan, B. Chen, J. Xue, N. Zheng, *et al.*, "Illumination robust color naming via label propagation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 621–629, 2015.

[54] Y. Wang, J. Liu, J. Wang, Y. Li, and H. Lu, "Color names learning using convolutional neural networks," in *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 217–221, IEEE, 2015.

[55] Z. Cheng, X. Li, and C. C. Loy, "Pedestrian color naming via convolutional neural network," in *Computer Vision–ACCV 2016: 13th Asian Conference*

on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13, pp. 35–51, Springer, 2017.
[56] L. Yu, Y. Cheng, and J. van de Weijer, "Weakly supervised domain-specific color naming based on attention," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 3019–3024, IEEE, 2018.
[57] R. Chaabouni, E. Kharitonov, E. Dupoux, and M. Baroni, "Communicating artificial neural networks develop efficient color-naming systems," *Proceedings of the National Academy of Sciences*, vol. 118, no. 12, p. e2016569118, 2021.
[58] J. Van de Weijer and F. S. Khan, "Fusing color and shape for bag-of-words based object recognition," in *International Workshop on Computational Color Imaging*, pp. 25–34, Springer, 2013.
[59] J. Lou, H. Wang, L. Chen, F. Xu, Q. Xia, W. Zhu, and M. Ren, "Exploiting color name space for salient object detection," *Multimedia Tools and Applications*, vol. 79, pp. 10873–10897, 2020.
[60] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.
[61] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
[62] M. V. Conde, F. Vasluianu, J. Vazquez-Corral, and R. Timofte, "Perceptual image enhancement for smartphone real-time applications," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1848–1858, 2023.
[63] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[64] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.
[65] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.
[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.