

# HistoGAN: Controlling Colors of GAN-Generated and Real Images via Color Histograms

Mahmoud Afifi

Marcus A. Brubaker

Michael S. Brown

York University

{mafifi, mab, mbrown}@eecs.yorku.ca

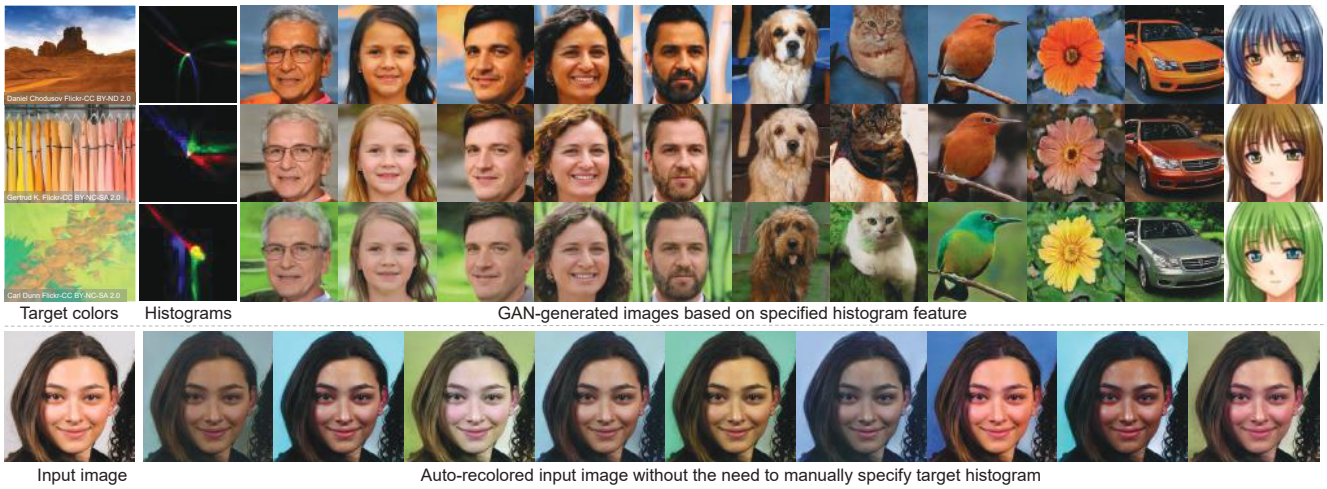


Figure 1: HistoGAN is a generative adversarial network (GAN) that learns to manipulate image colors based on histogram features. Top: GAN-generated images with color distributions controlled via target histogram features (left column). Bottom: Results of ReHistoGAN, an extension of HistoGAN to recolor real images, using sampled target histograms.

## Abstract

While generative adversarial networks (GANs) can successfully produce high-quality images, they can be challenging to control. Simplifying GAN-based image generation is critical for their adoption in graphic design and artistic work. This goal has led to significant interest in methods that can intuitively control the appearance of images generated by GANs. In this paper, we present HistoGAN, a color histogram-based method for controlling GAN-generated images’ colors. We focus on color histograms as they provide an intuitive way to describe image color while remaining decoupled from domain-specific semantics. Specifically, we introduce an effective modification of the recent StyleGAN architecture [34] to control the colors of GAN-generated images specified by a target color histogram feature. We then describe how to expand HistoGAN to recolor real images. For image recoloring, we jointly train an encoder network along with HistoGAN.

*The recoloring model, ReHistoGAN, is an unsupervised approach trained to encourage the network to keep the original image’s content while changing the colors based on the given target histogram. We show that this histogram-based approach offers a better way to control GAN-generated and real images’ colors while producing more compelling results compared to existing alternative strategies.*

## 1. Motivation and Related Work

Color histograms are an expressive and convenient representation of an image’s color content. Color histograms are routinely used by conventional color transfer methods (e.g., [21, 46, 52, 63]). These color transfer methods aim to manipulate the colors in an input image to match those of a target image, such that the images share a similar “look and feel”. In the color transfer literature, there are various forms of color histograms used to represent the color distribution

of an image, such as a direct 3D histogram [21, 52, 63], 2D histogram [4, 6, 10, 11], color palette [7, 13, 65] or color triad [59]. Despite the effectiveness of color histograms for color transfer, recent deep learning methods almost exclusively rely on image-based examples to control colors. While image exemplars impact the final colors of generative adversarial network (GAN)-generated images and deep recolored images, these methods – that mostly target image style transfer – also affect other style attributes, such as texture information and tonal values [23, 24, 30, 31, 43, 56, 61]. Consequently, the quality of the results produced by these methods often depends on the semantic similarity between the input and target images, or between a target image and a particular domain [28, 56].

In this paper, our attention is focused explicitly on controlling only the color attributes of images—this can be considered a sub-category of image style transfer. Specifically, our method does not require shared semantic content between the input/GAN-generated images and a target image or guide image. Instead, our method aims to assist the deep network through color histogram information only<sup>1</sup>. With this motivation, we first explore using color histograms to control the colors of images generated by GANs.

**Controlling Color in GAN-Generated Images** GANs are often used as “black boxes” that can transform samples from a simple distribution to a meaningful domain distribution without an explicit ability to control the details/style of the generated images [9, 25, 32, 41, 51]. Recently, methods have been proposed to control the style of the GAN-generated images. For example, StyleGAN [33, 34] proposed the idea of “style mixing”, where different latent style vectors are progressively fed to the GAN to control the style and appearance of the output image. To transfer a specific style in a target image to GAN-generated images, an optimization process can be used to project the target image to the generator network’s latent space to generate images that share some properties with the target image [1, 34]. However, this process requires expensive computations to find the latent code of the target image. Another direction is to jointly train an encoder-generator network to learn this projection [15, 39, 49]. More recently, methods have advocated different approaches to control the output of GANs, such as using the normalization flow [2], latent-to-domain-specific mapping [15], deep classification features [58], few-shot image-to-image translation [54], and a single-image training strategy [55]. Despite the performance improvements, most of these methods are limited to work with a single domain of both target and GAN-generated images [39, 49].

We seek to control GAN-generated images using color histograms as our specified representation of image style. Color histograms enable our method to accept target images

taken from *any* arbitrary domain. Figure 1-top shows GAN-generated examples using our method. As shown in Fig. 1, our generated images share the same color distribution as the target images without being restricted to, or influenced by, the semantic content of the target images.

**Recoloring Real Images** In addition to controlling the GAN-generated images, we seek to extend our approach to perform image recoloring within the GAN framework. In this context, our method accepts a real input image and a target histogram to produce an output image with the fine details of the input image but with the same color distribution given in the target histogram. Our method is trained in a fully unsupervised fashion, where no ground-truth recolored image is required. Instead, we propose a novel adversarial-based loss function to train our network to extract and consider the color information in the given target histogram while producing realistic recolored images. One of the key advantages of using the color histogram representation as our target colors can be shown in Fig. 1-bottom, where we can *automatically recolor* an image without directly having to specify a target color histogram. Auto-image recoloring is a less explored research area with only a few attempts in the literature (e.g., [7, 8, 18, 38]).

## 2. HistoGAN

We begin by describing the histogram feature used by our method (Sec. 2.1). Afterwards, we discuss the proposed modification to StyleGAN [34] to incorporate our histogram feature into the generator network (Sec. 2.2). Lastly, we explain how this method can be expanded to control colors of real input images to perform image recoloring (Sec. 2.3).

### 2.1. Histogram feature

The histogram feature used by HistoGAN is borrowed from the color constancy literature [4, 6, 11] and is constructed to be a differentiable histogram of colors in the log-chroma space due to better invariance to illumination changes [20, 22]. The feature is a 2D histogram of an image’s colors projected into a log-chroma space. This 2D histogram is parameterized by  $uv$  and conveys an image’s color information while being more compact than a typical 3D histogram defined in RGB space. A log-chroma space is defined by the intensity of one channel, normalized by the other two, giving three possible options of how it is defined. Instead of selecting only one such space, all three options can be used to construct three different histograms which are combined together into a histogram feature,  $\mathbf{H}$ , as an  $h \times h \times 3$  tensor [6].

The histogram is computed from a given input image,  $\mathbf{I}$ , by first converting it into the log-chroma space. For instance, selecting the R color channel as primary and nor-

<sup>1</sup>Project page: <https://github.com/mahmoudnafifi/HistoGAN>

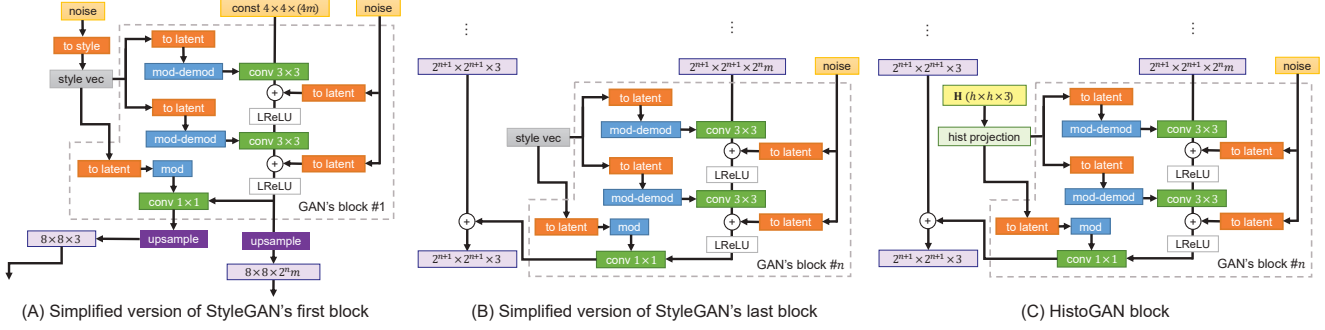


Figure 2: We inject our histogram into StyleGAN [34] to control the generated image colors. (A) and (B) are simplified versions of the StyleGAN’s first and last blocks. We modified the last two blocks of the StyleGAN by projecting our histogram feature into each block’s latent space, as shown in (C). The parameter  $m$  controls the capacity of the model.

malizing by G and B gives:

$$\mathbf{I}_{uR}(\mathbf{x}) = \log \left( \frac{\mathbf{I}_R(\mathbf{x}) + \epsilon}{\mathbf{I}_G(\mathbf{x}) + \epsilon} \right), \mathbf{I}_{vR}(\mathbf{x}) = \log \left( \frac{\mathbf{I}_R(\mathbf{x}) + \epsilon}{\mathbf{I}_B(\mathbf{x}) + \epsilon} \right), \quad (1)$$

where the R, G, B subscripts refer to the color channels of the image  $\mathbf{I}$ ,  $\epsilon$  is a small constant added for numerical stability,  $\mathbf{x}$  is the pixel index, and  $(uR, vR)$  are the  $uv$  coordinates based on using R as the primary channel. The other components  $\mathbf{I}_{uG}$ ,  $\mathbf{I}_{vG}$ ,  $\mathbf{I}_{uB}$ ,  $\mathbf{I}_{vB}$  are computed similarly by projecting the G and B color channels to the log-chroma space. In [6], the RGB- $uv$  histogram is computed by thresholding colors to a bin and computing the contribution of each pixel based on the intensity  $\mathbf{I}_y(\mathbf{x}) = \sqrt{\mathbf{I}_R^2(\mathbf{x}) + \mathbf{I}_G^2(\mathbf{x}) + \mathbf{I}_B^2(\mathbf{x})}$ . In order to make the representation differentiable, [4] replaced the thresholding operator with a kernel weighted contribution to each bin. The final unnormalized histogram is computed as:

$$\mathbf{H}(u, v, c) \propto \sum_{\mathbf{x}} k(\mathbf{I}_{uc}(\mathbf{x}), \mathbf{I}_{vc}(\mathbf{x}), u, v) \mathbf{I}_y(\mathbf{x}), \quad (2)$$

where  $c \in \{\mathbf{R}, \mathbf{G}, \mathbf{B}\}$  and  $k(\cdot)$  is a pre-defined kernel. While a Gaussian kernel was originally used in [4], we found that the inverse-quadratic kernel significantly improved training stability. The inverse-quadratic kernel is defined as:

$$k(\mathbf{I}_{uc}, \mathbf{I}_{vc}, u, v) = \left( 1 + (|\mathbf{I}_{uc} - u|/\tau)^2 \right)^{-1} \times \left( 1 + (|\mathbf{I}_{vc} - v|/\tau)^2 \right)^{-1}, \quad (3)$$

where  $\tau$  is a fall-off parameter to control the smoothness of the histogram’s bins. Finally, the histogram feature is normalized to sum to one, i.e.,  $\sum_{u,v,c} \mathbf{H}(u, v, c) = 1$ .

## 2.2. Color-controlled Image Generation

Our histogram feature is incorporated into an architecture based on StyleGAN [34]. Specifically, we modified

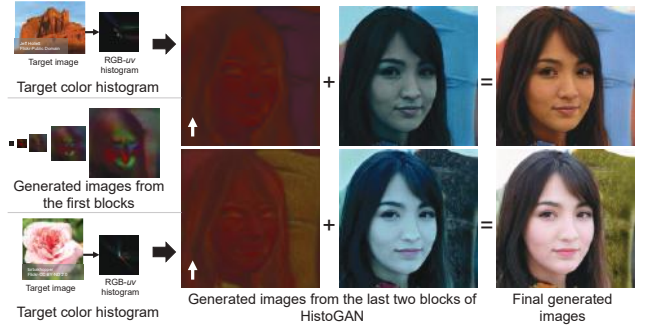


Figure 3: Progressively generated images using the HistoGAN modifications.

the original design of StyleGAN (Fig. 2-[A] and [B]) such that we can “inject” the histogram feature into the progressive construction of the output image. The last two blocks of the StyleGAN (Fig. 2-[B]) are modified by replacing the fine-style vector with the color histogram feature. The histogram feature is then projected into a lower-dimensional representation by a “histogram projection” network (Fig. 2-[C]). This network consists of eight fully connected layers with a leaky ReLU (LReLU) activation function [44]. The first layer has 1,024 units, while each of the remaining seven layers has 512. The “to-latent” block, shown in orange in Fig. 2, maps the projected histogram to the latent space of each block. This “to-latent” block consists of a single fc layer with  $2^m$  output neurons, where  $n$  is the block number, and  $m$  is a parameter used to control the entire capacity of the network.

To encourage generated images to match the target color histogram, a color matching loss is introduced to train the generator. Because of the differentiability of our histogram representation, the loss function,  $C(\mathbf{H}_g, \mathbf{H}_t)$ , can be any differentiable metric of similarity between the generated and target histograms  $\mathbf{H}_g$  and  $\mathbf{H}_t$ , respectively. For sim-



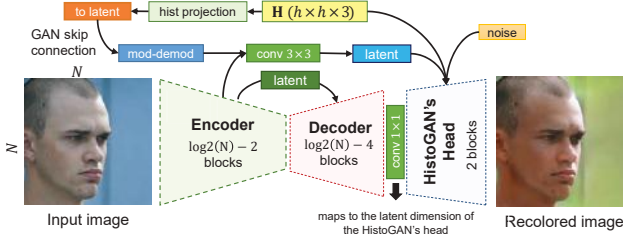


Figure 4: Our Recoloring-HistoGAN (ReHistoGAN) network. We map the input image into the HistoGAN’s latent space using an encoder-decoder network with skip connections between each encoder and decoder blocks. Additionally, we pass the latent feature of the first two encoder blocks to our GAN’s head after processing it with the histogram’s latent feature.

plicity, we use the Hellinger distance defined as:

$$C(\mathbf{H}_g, \mathbf{H}_t) = \frac{1}{\sqrt{2}} \left\| \mathbf{H}_g^{1/2} - \mathbf{H}_t^{1/2} \right\|_2, \quad (4)$$

where  $\| \cdot \|_2$  is the standard Euclidean norm and  $\mathbf{H}^{1/2}$  is an element-wise square root. Note that the Hellinger distance is closely related to the Bhattacharyya coefficient,  $B(\cdot)$ , where  $C(\mathbf{H}_g, \mathbf{H}_t) = (1 - B(\mathbf{H}_g, \mathbf{H}_t))^{1/2}$ .

This color-matching histogram loss function is combined with the discriminator to give the generator network loss:

$$\mathcal{L}_g = D(\mathbf{I}_g) + \alpha C(\mathbf{H}_g, \mathbf{H}_t), \quad (5)$$

where  $\mathbf{I}_g$  is the GAN-generated image,  $D(\cdot)$  is our discriminator network that produces a scalar feature given an image (see supp. materials for more details),  $\mathbf{H}_t$  is the target histogram feature (injected into the generator network),  $\mathbf{H}_g$  is the histogram feature of  $\mathbf{I}_g$ ,  $C(\cdot)$  is our histogram loss function, and  $\alpha$  is a scale factor to control the strength of the histogram loss term.

As our histogram feature is computed by a set of differentiable operations, our loss function (Eqs. 4 and 5) can be optimized using SGD. During training, different target histograms  $\mathbf{H}_t$  are required. To generate these for each generated image, we randomly select two images from the training set, compute their histograms  $\mathbf{H}_1$  and  $\mathbf{H}_2$ , and then randomly interpolate between them. Specifically, for each generated image during training, we generate a random target histogram as follows:

$$\mathbf{H}_t = \delta \mathbf{H}_1 + (1 - \delta) \mathbf{H}_2, \quad (6)$$

where  $\delta \sim U(0, 1)$  is sampled uniformly. The motivation behind this interpolation process is to expand the variety of histograms during training. This is a form of data augmentation for the histograms with the implicit assumption of the convexity of the histogram distribution in the target domain

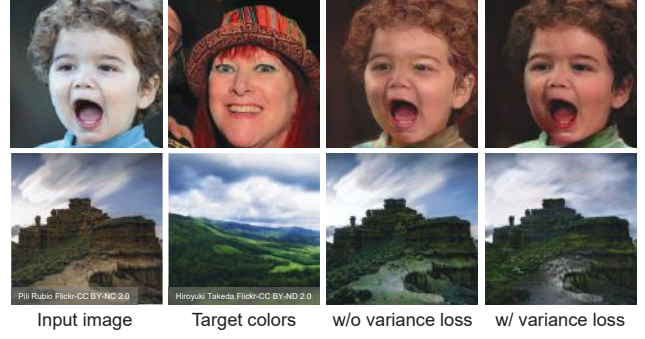


Figure 5: Results of training ReHistoGAN with and without the variance loss term described in Eq. 9.

(e.g., face images). We found this augmentation helped reduce overfitting to the histograms of the training images and ensured robustness at test time. We note that this assumption does not hold true for target domains with high diversity where the target histograms span a broad range in the log-chroma space and can be multimodal (e.g., landscape images). Nonetheless, we found that even in those cases the augmentation was still beneficial to the training.

With this modification to the original StyleGAN architecture, our method can control the colors of generated images using our color histogram features. Figure 3 shows the progressive construction of the generated image by HistoGAN. As can be seen, the outputs of the last two blocks are adjusted to consider the information conveyed by the target histogram to produce output images with the same color distribution represented in the target histogram.

### 2.3. Image Recoloring

We can also extend HistoGAN to recolor an input image, as shown in Fig. 1-bottom. Recoloring an existing input image,  $\mathbf{I}_i$ , is not straightforward because the randomly sampled noise and style vectors are not available as they are in a GAN-generated scenario. As shown in Fig. 3, the head of HistoGAN (i.e., the last two blocks) are responsible for controlling the colors of the output image. Instead of optimizing for noise and style vectors that could be used to generate a given image  $\mathbf{I}_i$ , we propose to train an encoding network that maps the input image into the necessary inputs of the head of HistoGAN. With this approach, the head block can



Figure 6: Results of image recoloring using the encoder-decoder reconstruction without skip connections and our ReHistoGAN using our proposed loss function.



Figure 7: Images generated by HistoGAN. For each input image shown in the left, we computed the corresponding target histogram (shown in the upper left corner of the left column) and used it to control colors of the generated images in each row.

be given different histogram inputs to produce a wide variety of recolored versions of the input image. We dub this extension the “Recoloring-HistoGAN” or ReHistoGAN for short. The architecture of ReHistoGAN is shown in Fig. 4. The “encoder” has a U-Net-like structure [53] with skip connections. To ensure that fine details are preserved in the recolored image,  $I_r$ , the early latent feature produced by the first two U-Net blocks are further provided as input into the HistoGAN’s head through skip connections.

The target color information is passed to the HistoGAN head blocks as described in Sec. 2.2. Additionally, we allow the target color information to influence through the

skip connections to go from the first two U-Net-encoder blocks to the HistoGAN’s head. We add an additional histogram projection network, along with a “to-latent” block, to project our target histogram to a latent representation. This latent code of the histogram is processed by weight modulation-demodulation operations [34] and is then convolved over the skipped latent of the U-Net-encoder’s first two blocks. We modified the HistoGAN block, described in Fig. 2, to accept this passed information (see supp. materials for more information). The leakage of the target color information helps ReHistoGAN to consider information from both the input image content and the target histogram in the



recoloring process.

We initialize our encoder-decoder network using He’s initialization [26], while the weights of the HistoGAN head are initialized based on a previously trained HistoGAN model (trained in Sec. 2.2). The entire ReHistoGAN is then jointly trained to minimize the following loss function:

$$\mathcal{L}_r = \beta R(\mathbf{I}_i, \mathbf{I}_r) + \gamma D(\mathbf{I}_r) + \alpha C(\mathbf{H}_r, \mathbf{H}_t) \quad (7)$$

where  $R(\cdot)$  is a reconstruction term, which encourages the preservation of image structure and  $\alpha$ ,  $\beta$ , and  $\gamma$  are hyperparameters used to control the strength of each loss term (see supp. materials for associated ablation study). The reconstruction loss term,  $R(\cdot)$ , computes the L1 norm between the second order derivative of our input and recolored images as:

$$R(\mathbf{I}_i, \mathbf{I}_r) = \|\mathbf{I}_i * \mathbf{L} - \mathbf{I}_r * \mathbf{L}\|_1 \quad (8)$$

where  $*\mathbf{L}$  denotes the application of the Laplacian operator. The idea of employing the image derivative was used initially to achieve image seamless cloning [48], where this Laplacian operator suppressed image color information while keeping the most significant perceptual details. Intuitively, ReHistoGAN is trained to consider the following aspects in the output image: (i) having a similar color distribution to the one represented in the target histogram, this is considered by  $C(\cdot)$ , (ii) being realistic, which is the goal of  $D(\cdot)$ , and (iii) having the same content of the input image, which is the goal of  $R(\cdot)$ .

Our model trained using the loss function described in Eq. 7 produces reasonable recoloring results. However, we noticed that, in some cases, our model tends to only apply a global color cast (i.e., shifting the recolored image’s histogram) to minimize  $C(\cdot)$ . To mitigate this behavior, we added variance loss term to Eq. 7. The variance loss can be described as:

$$V(\mathbf{I}_i, \mathbf{I}_r) = -w \sum_{c \in \{R, G, B\}} |\sigma(\mathbf{I}_{ic} * \mathbf{G}) - \sigma(\mathbf{I}_{rc} * \mathbf{G})|, \quad (9)$$

where  $\sigma(\cdot)$  computes the standard deviation of its input (in this case the blurred versions of  $\mathbf{I}_i$  and  $\mathbf{I}_r$  using a Gaussian blur kernel,  $\mathbf{G}$ , with a scale parameter of 15), and  $w = \|\mathbf{H}_t - \mathbf{H}_i\|_1$  is a weighting factor that increases as the target histogram and the input image’s histogram,  $\mathbf{H}_t$  and  $\mathbf{H}_i$ , become dissimilar and the global shift solution becomes more problematic. The variance loss encourages the network to avoid the global shifting solution by increasing the differences between the color variance in the input and recolored images. The reason behind using a blurred version of each image is to avoid having a contradiction between the variance loss and the reconstruction loss—the former aims to increase the differences between the variance of the *smoothed* colors in each image, while the latter aims to retain the similarity between the fine details of the input and

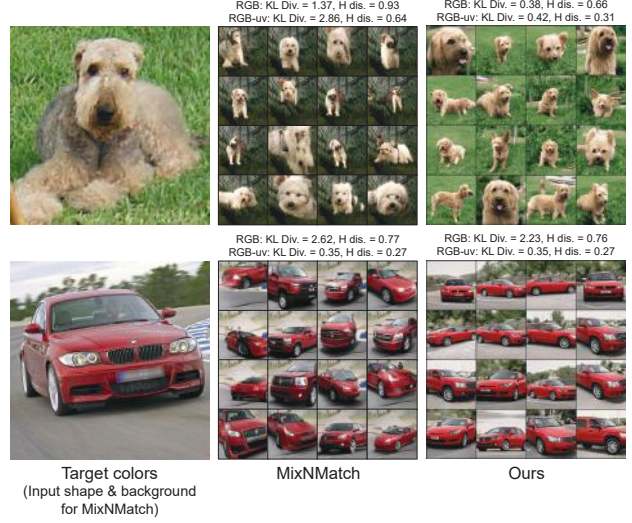


Figure 8: Comparison with the MixNMatch method [39]. In the shown results, the target images are used as input shape and background images for the MixNMatch method [39].

recolored images. Figure 5 shows recoloring results of our trained models with and without the variance loss term.

We train ReHistoGAN with target histograms sampled from the target domain dataset, as described earlier in Sec. 2.2 (Eq. 6).

A simpler architecture was experimented initially, which did not make use of the skip connections and the end-to-end fine tuning (i.e., the weights of the HistoGAN head were fixed). However, this approach gave unsatisfactory result, and generally failed to retain fine details of the input image. A comparison between this approach and the above ReHistoGAN architecture can be seen in Fig. 6.

### 3. Results and Discussion

This section discusses our results and comparisons with alternative methods proposed in the literature for controlling color. Due to hardware limitations, we used a lightweight version of the original StyleGAN [34] by setting  $m$  to 16, shown in Fig. 2. We begin by presenting our image generation results, followed by our results on image recoloring. Additional results, comparisons, and discussion are also available in the supp. materials.

**Image Generation** Figure 7 shows examples of our HistoGAN-generated images. Each row shows samples generated from different domains using the corresponding input target colors. For each domain, we fixed the style vectors responsible for the coarse and middle styles to show our HistoGAN’s response to changes in the target histograms. Qualitative comparisons with the recent MixNMatch method [39] are provided in Fig. 8.

To evaluate the potential improvement/degradation of

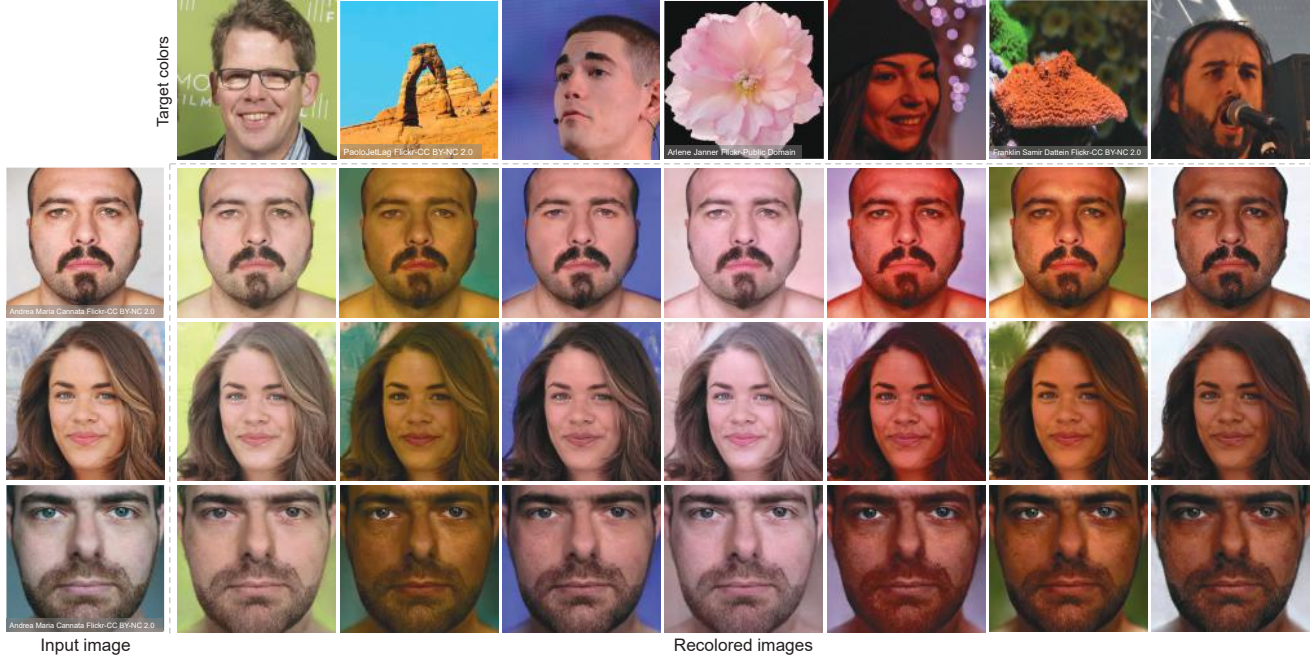


Figure 9: Results of our ReHistoGAN. The shown results are after recoloring input images (shown in the left column) using the target colors (shown in the top row).

Table 1: Comparison with StyleGAN [34]. The term ‘w/ proj.’ refers to projecting the target image colors into the latent space of StyleGAN. We computed the similarity between the target and generated histograms in RGB and projected RGB-uv color spaces. For each dataset, we report the number of training images. Note that StyleGAN results shown here *do not* represent the actual output of [34], as the used model here has less capacity ( $m = 16$ ).

Dataset	StyleGAN [34]						HistoGAN (ours)					
	FID		RGB hist. (w/ proj.)		RGB-uv hist. (w/ proj.)		FID	RGB hist. (w/ proj.)		RGB-uv hist. (w/ proj.)		
	w/o proj.	w/ proj.	KL Div.	H dis.	KL Div.	H dis.		KL Div.	H dis.	KL Div.	H dis.	
Faces (69,822) [33]	9.5018	14.194	1.3124	0.9710	1.2125	0.6724	8.9387	0.9810	0.7487	0.4470	0.3088	
Flowers (8,189) [47]	10.876	15.502	1.0304	0.9614	2.7110	0.7038	4.9572	0.8986	0.7353	0.3837	0.2957	
Cats (9,992) [17]	14.366	21.826	1.6659	0.9740	1.4051	0.5303	17.068	1.0054	0.7278	0.3461	0.2639	
Dogs (20,579) [35]	16.706	30.403	1.9042	0.9703	1.4856	0.5658	20.336	1.3565	0.7405	0.4321	0.3058	
Birds (9,053) [62]	3.5539	12.564	1.9035	0.9706	1.9134	0.6091	3.2251	1.4976	0.7819	0.4261	0.3064	
Anime (63,565) [16]	2.5002	9.8890	0.9747	0.9869	1.4323	0.5929	5.3757	0.8547	0.6211	0.1352	0.1798	
Hands (11,076) [3]	2.6853	2.7826	0.9387	0.9942	0.3654	0.3709	2.2438	0.3317	0.3655	0.0533	0.1085	
Landscape (4,316)	24.216	29.248	0.8811	0.9741	1.9492	0.6265	23.549	0.8315	0.8169	0.5445	0.3346	
Bedrooms (303,116) [64]	10.599	14.673	1.5709	0.9703	1.2690	0.5363	4.5320	1.3774	0.7278	0.2547	0.2464	
Cars (16,185) [36]	21.485	25.496	1.6871	0.9749	0.7364	0.4231	14.408	1.0743	0.7028	0.2923	0.2431	
Aerial Scenes (36,000) [45]	11.413	14.498	2.1142	0.9798	1.1462	0.5158	12.602	0.9889	0.5887	0.1757	0.1890	

the generated-image diversity and quality caused by our modification to StyleGAN, we trained StyleGAN [34] with  $m = 16$  (i.e., the same as our model capacity) without our histogram modification. We evaluated both models on different datasets, including our collected set of landscape images. For each dataset, we generated 10,000  $256 \times 256$  images using the StyleGAN and our HistoGAN. We evaluated the generated-image quality and diversity using the Fréchet inception distance (FID) metric [29] using the second max-pooling features of the Inception model [60].

We further evaluated the ability of StyleGAN to control colors of GAN-generated images by training a regression

deep neural network (ResNet [27]) to transform generated images back to the corresponding fine-style vectors. These fine-style vectors are used by the last two blocks of StyleGAN and are responsible for controlling delicate styles, such as colors and lights [33, 34].

The training was performed for each domain separately using 100,000 training StyleGAN-generated images and their corresponding “ground-truth” fine-style vectors. In the testing phase, we used the trained ResNet to predict the corresponding fine-style vectors of the target image—these target images were used to generate the target color histograms for HistoGAN’s experiments. We then generated output im-

ages based on the predicted fine-style vectors of each target image. In the evaluation of StyleGAN and HistoGAN, we used randomly selected target images from the same domain.

The Hellinger distance and KL divergence were used to measure the color errors between the histograms of the generated images and the target histogram; see Table 1.

**Image Recoloring** Figure 9 shows examples of image recoloring using our ReHistoGAN. A comparison with the recent high-resolution daytime translation (HiDT) method [8] is shown in Fig. 10. Additional comparisons with image recoloring and style transfer methods are shown in Fig. 11. Arguably, our ReHistoGAN produces image recoloring results that are visually more compelling than the results of other methods for image color/style transfer. As shown in Fig. 11, our ReHistoGAN produces realistic recoloring even when the target image is from a different domain than the input image, compared to other image style transfer methods (e.g., [24, 56]).

Lastly, we provide a qualitative comparison with the recent auto-recoloring method proposed by Afifi et al., [7] in Fig. 12. In the shown example, our target histograms were dynamically generated by sampling from a pre-defined set of histograms and applying a linear interpolation between the sampled histograms (see Eq. 6).

**What is Learned?** Our method learns to map color information, represented by the target color histogram, to an output image’s colors with a realism consideration in the recolored image. Maintaining realistic results is achieved by learning proper matching between the target colors and the input image’s semantic objects (e.g., grass can be green, but not blue). To demonstrate this, we examine a trained ReHistoGAN model for an image colorization task, where the input image is grayscale. The input of a grayscale image means that our ReHistoGAN model has no information regarding objects’ colors in the input image. Figure 13 shows outputs where the input has been “colorized”. As can be seen, the output images have been colorized with good semantic-color matching based on the image’s content.



Figure 10: Comparison with the high-resolution daytime translation (HiDT) method [8].

## 4. Conclusion

We have presented HistoGAN, a simple, yet effective, method for controlling colors of GAN-generated images. Our HistoGAN framework learns how to transfer the color information encapsulated in a target histogram feature to the colors of a generated output image. To the best of our knowledge, this is the first work to control the color of GAN-generated images directly from color histograms. Color histograms provide an abstract representation of image color that is decoupled from spatial information. This allows the histogram representation to be less restrictive and suitable for GAN-generation across arbitrary domains.

We have shown that HistoGAN can be extended to control colors of real images in the form of the ReHistoGAN model. Our recoloring results are visually more compelling than currently available solutions for image recoloring. Our image recoloring also enables “auto-recoloring” by sampling from a pre-defined set of histograms. This allows an image to be recolored to a wide range of visually plausible variations. HistoGAN can serve as a step towards intuitive color control for GAN-based graphic design and artistic endeavors.

## 5. Supplementary Material

### 5.1. Details of Our Networks

Our discriminator network, used in all of our experiments, consists of a sequence of  $\log_2(N)-1$  residual blocks, where  $N$  is the image width/height, and the last layer is an fully connected (fc) layer that produces a scalar feature. The first block accepts a three-channel input image and produce  $m$  output channels. Then, each block  $i$  produces  $2m_{i-1}$  output channels (i.e., duplicate the number of output channels of the previous block). The details of the residual blocks used to build our discriminator network are shown in Fig. S1.

Figure S2 provides the details of our encoder, decoder and GAN blocks used in our ReHistoGAN (used for image recoloring). As shown, we modified the last two blocks of our HistoGAN’s to accept the latent feature passed from the first two blocks of our encoder. This modification helps our HistoGAN’s head to consider both information of the input image structure and the target histogram in the recoloring process.

### 5.2. Training Details

We train our networks using an NVIDIA TITAN X (Pascal) GPU. For HistoGAN training, we optimized both the generator and discriminator networks using the diffGrad optimizer [19]. In all experiments, we set the histogram bin,  $h$ , to 64 and the fall-off parameter of our histogram’s bins,  $\tau$ , was set to 0.02. We adopted the exponential moving average of generator network’s weights [33, 34] with the path length





Figure 11: Comparisons between our ReHistoGAN and other image color/style transfer methods, which are: Reinhard et al., [52], Xiao et al., [63], Pitié and Kokaram [50], Nguyen et al., [46], Gatys et al., [24], and Sheng et al., [56].



Figure 12: Automatic recoloring comparison with the recent method by Afifi et al., [7].



Figure 13: Results of using our ReHistoGAN for a diverse image colorization.

penalty, introduced in StyleGAN [34], every 32 iterations to train our generator network. Due to the hardware limitation,

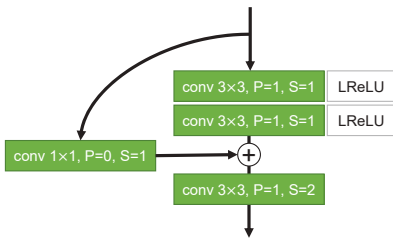


Figure S1: Details of the residual discriminator block used to reconstruct our discriminator network. The term P and S refer to the padding and stride used in each layer.

we used mini-batch of 2 with accumulated gradients every 16 iteration steps and we set the image’s dimension,  $N$ , to 256. We set the scale factor of the Hellinger distance loss,  $\alpha$ , to 2 (see Sec. 5.3 for an ablation study).

As mentioned in the main paper, we trained our HistoGAN using several domain datasets, including: human faces [33], flowers [47], cats [17], dogs [35], birds [62], anime faces [16], human hands [3], bedrooms [64], cars [36], and aerial scenes [45]. We further trained our HistoGAN using 4,316 landscape images collected from Flickr. The collected images have one of the following copyright licenses: no known copyright restrictions, Public Domain Dedication (CC0), or Public Domain Mark. See Fig. S3 for representative examples from the landscape set.

To train our ReHistoGAN, we used the diffGrad optimizer [19] with the same mini-batch size used to train our HistoGAN. We trained our network using the following hyperparameters  $\alpha = 2$ ,  $\beta = 1.5$ ,  $\gamma = 32$  for 100,000 iterations. Then, we continued training using  $\alpha = 2$ ,  $\beta = 1$ ,  $\gamma = 8$  for additional 30,000 iterations to reduce potential artifacts in recoloring (see Sec. 5.3 for an ablation study).

### 5.3. Ablation Studies

We carried out a set of ablation experiments to study the effect of different values of hyperparameters used in the main paper. Additionally, we show results obtained by variations in our loss terms.

We begin by studying the effect of the scale factor,  $\alpha$ , used in the loss function to train our HistoGAN. This scale factor was used to control strength of the histogram loss term. In this set of experiments, we used the 11K Hands dataset [3] to be our target domain and trained our HistoGAN with the following values of  $\alpha$ : 0.2, 2, 4, 8, and 16. Table S1 shows the evaluation results using the Frechét inception distance (FID) metric [29], the KL divergence, and

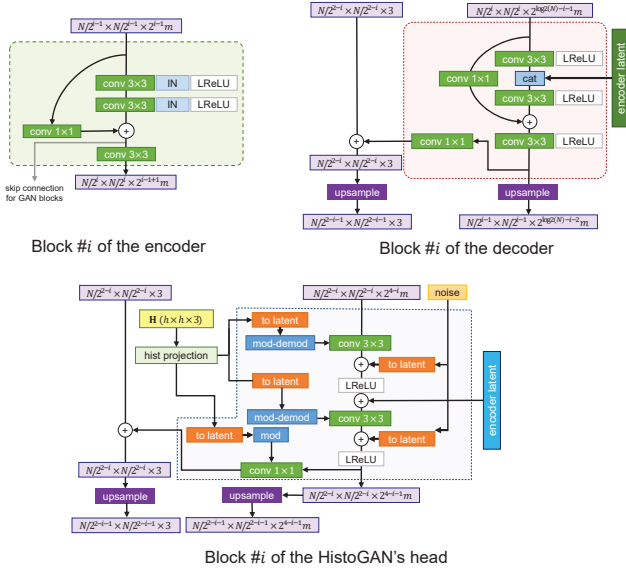


Figure S2: Details of our ReHistoGAN network. We modified the last two blocks of our HistoGAN by adding a gate for the processed skipped features from the first two blocks of our encoder.



Figure S3: Examples taken from our set of 4,316 landscape images collected from Flickr.

Hellinger distance. The KL divergence and Hellinger distance were used to measure the similarity between the target histogram and the histogram of GAN-generated images. Qualitative comparisons are shown in Fig. S4

Figure S5 shows examples of recoloring results obtained by trained ReHistoGAN models using different combination values of  $\alpha$ ,  $\beta$ ,  $\gamma$ . As can be seen, a lower value of the scale factor,  $\alpha$ , of the histogram loss term results in ignoring our network to the target colors, while higher values of the scale factor,  $\gamma$ , of the discriminator loss term, make our method too fixated on producing realistic output

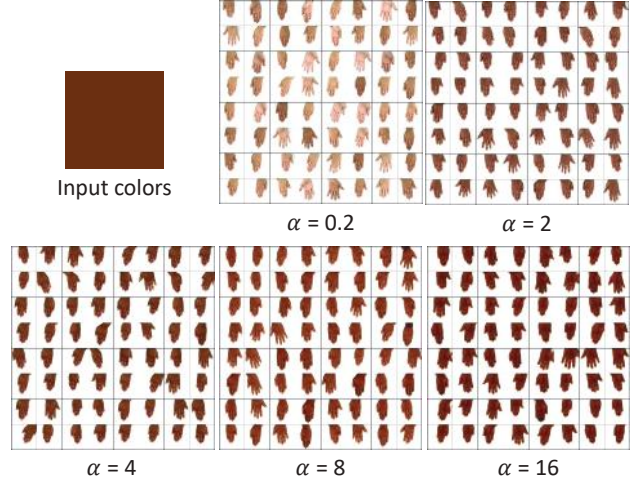


Figure S4: Results obtained by training our HistoGAN in hand images [3] using different values of  $\alpha$ .

Table S1: Results of our HistoGAN using different values of  $\alpha$ . In this set of experiments, we used the Hands dataset [3] as our target domain. The term FID stands for the Fréchet inception distance metric [29]. The term KL Div. refers to the KL divergence between the histograms of the input image and generated image, while the term H. dis. refers to Hellinger distance.

$\alpha$	FID	RGB-uv hist.	
		KL Div.	H dist.
0.2	1.9950	0.3935	0.3207
2	<b>2.2438</b>	<b>0.0533</b>	<b>0.1085</b>
4	6.8750	0.0408	0.0956
8	9.4101	0.0296	0.0822
16	15.747	0.0237	0.0743

images, regardless of achieving the recoloring (i.e., tending to re-produce the input image as is).

In the recoloring loss, we used a reconstruction loss term to retain the input image's spatial details in the output recolored image. Our reconstruction loss is based on the derivative of the input image. We have examined two different kernels, which are: the vertical and horizontal  $3 \times 3$  Sobel kernels (i.e., the first-order derivative approximation) and the  $3 \times 3$  Laplacian kernel (i.e., the second-order derivative). We found that training using both kernels give reasonably good results, while the Laplacian kernel produces more compelling results in most cases; see Fig. S6 for an example.

In the main paper, we introduced a variance loss term to encourage our network to avoid the global color cast solution for image recoloring. Figure S7 shows an example of the global color cast problem, where the network applies a global color shift to the input image to match the target histogram. As shown in Fig. S7 after training our network with



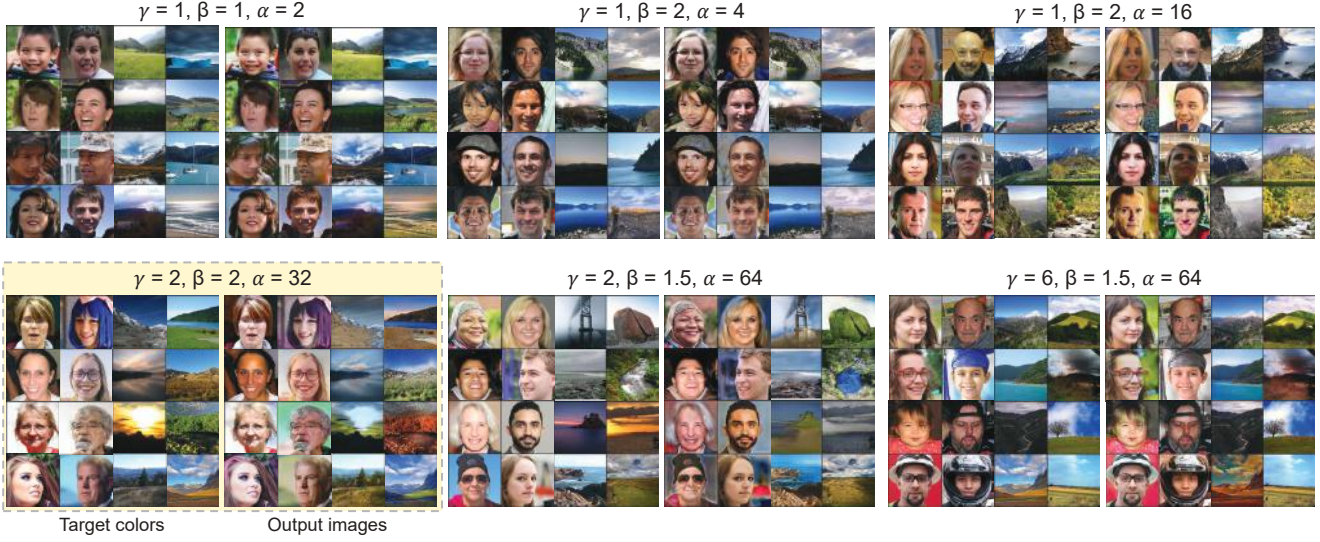


Figure S5: Results of recoloring by training our recoloring network using different values of  $\alpha$ ,  $\beta$ , and  $\gamma$  hyperparameters. The highlighted results refer to the settings used to produce the reported results in the main paper and the supplementary materials.



Figure S6: Results of two different kernels used to compute the reconstruction loss term.

the variance loss, this problem is reduced.

#### 5.4. Universal ReHistoGAN Model

As the case of most GAN methods, our ReHistoGAN targets a specific object domain to achieve the image recoloring task. This restriction may hinder the generalization of our method to deal with images taken from arbitrary domains. To deal with that, we collected images from a different domain, aiming to represent the “universal” object domain.

Specifically, our training set of images contains  $\sim 2.4$



Figure S7: The impact of the variance loss term. The shown results were obtained by training our ReHistoGAN with and without the variance loss term.

million images collected from different image datasets. These datasets are: collection from the Open Images dataset [37], the MIT-Adobe FiveK dataset [12], the Microsoft COCO dataset [40], the CelebA dataset [42], the Caltech-UCSD birds-200-2011 dataset [62], the Cats dataset [17], the Dogs dataset [35], the Cars dataset [36], the Oxford Flowers dataset [47], the LSUN dataset [64], the ADE20K dataset [66, 67], and the FFHQ dataset [33]. We also added Flickr images collected using the following keywords: landscape, people, person, portrait, field, city, sunset, beach, animals, living room, home, house, night, street, desert, food. We have excluded any grayscale image from the collected image set.

We trained our “universal” model using  $m = 18$  on this collected set of 2,402,006 images from several domains. The diffGrad optimizer [19] was used to minimize the same generator loss described in the main paper using the following hyperparameters  $\alpha = 2$ ,  $\beta = 1.5$ ,  $\gamma = 32$  for 150,000 iterations. Then, we used  $\alpha = 2$ ,  $\beta = 1$ ,  $\gamma = 8$  to train the model for additional 350,000 iterations. We set the mini-batch size to 8 with an accumulated gradient every 24 iterations. Figure S8 show results of our domain-specific and universal models for image recoloring. As can be seen, both models produce realistic recoloring, though the universal model tends to produce recolored images with less vivid colors compared to our domain-specific model. Additional examples of auto recoloring using our universal model are shown in Fig. S9.

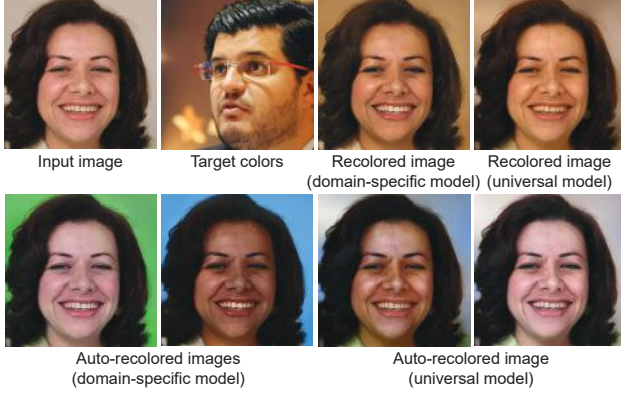


Figure S8: Results of domain-specific and universal ReHistoGAN models. We show results of using a given target histogram for recoloring and two examples of the auto recoloring results of each model.

## 5.5. Limitations

Our method fails in some cases, where the trained HistoGAN could not properly extract the target color information represented in the histogram feature. This problem is due to the inherent limitation of the 2D projected representation of the original target color distribution, where different colors are mapped to the same chromaticity value in the projected space. This is shown in Fig. S10-top, where the GAN-generated images do not have all colors in the given target histogram. Another failure case can occur in image recoloring, where the recolored images could have some color-bleeding artifacts due to errors in allocating the target/sampled histogram colors in the recolored image. This can be shown in Fig. S10-bottom

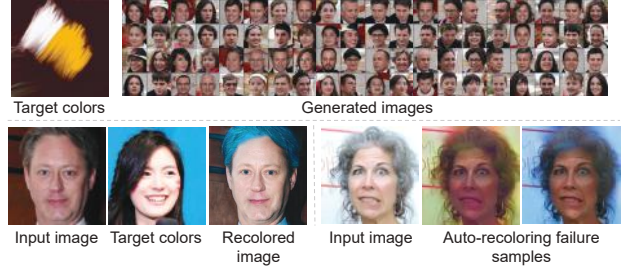


Figure S10: Failure cases of HistoGAN and ReHistoGAN. Our HistoGAN fails sometimes to consider all colors of target histogram in the generated image. Color bleeding is another problem that could occur in ReHistoGAN's results, where our network could not properly allocate the target (or sampled) histogram colors in the recolored image.

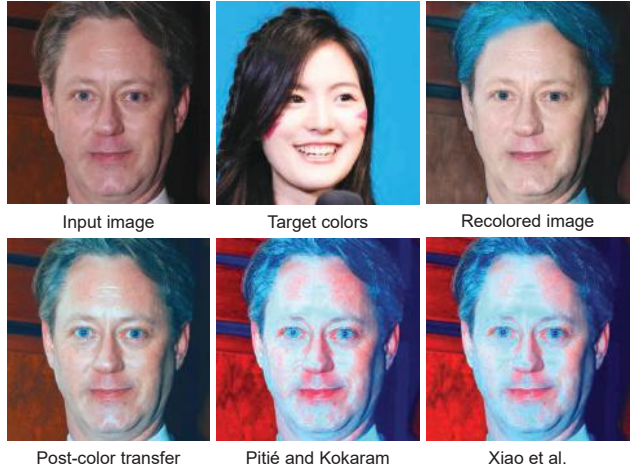


Figure S11: To reduce potential color bleeding artifacts, it is possible to apply a post-color transfer to our initial recolored image colors to the input image. The results of adopting this strategy are better than applying the color transfer to the input image in the first place. Here, we use the color transfer method proposed by Pitié and Kokaram [50] as our post-color transfer method. We also show the results of directly applying Pitié and Kokaram's [50] method to the input image.

## 5.6. Post-Processing

As discussed in Sec. 5.5, our method produces, in some times, results with color bleeding, especially when the target histogram feature has unsuitable color distribution for the content of the input image. This color-bleeding problem can be mitigated using a post-process color transfer between the input image and our initial recoloring. Surprisingly, this post-processing mapping produces results better than adopting the mapping in the first place—namely, applying the color transfer mapping without having our intermediate recoloring result.

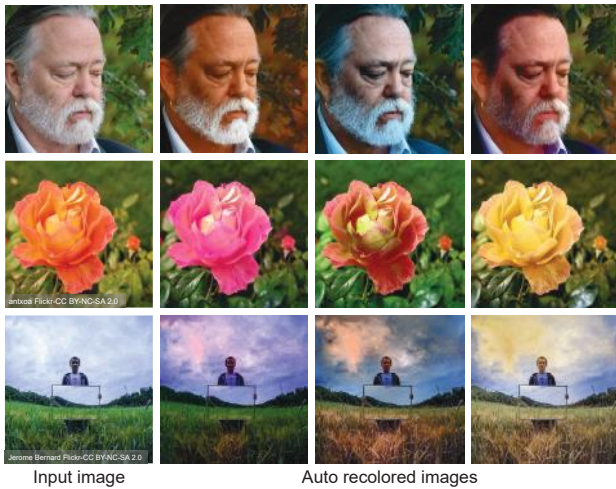


Figure S9: Auto recoloring using our universal ReHistoGAN model.



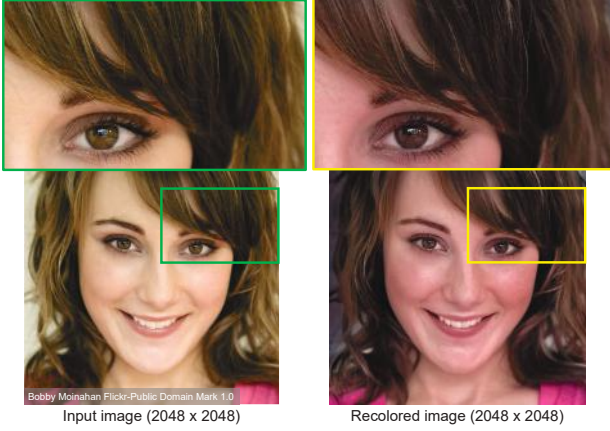


Figure S12: We apply the bilateral guided upsampling [14] as a post-processing to reduce potential artifacts of dealing with high-resolution images in the inference phase. In the shown example, we show our results of recoloring using an input image with  $2048 \times 2048$  pixels.

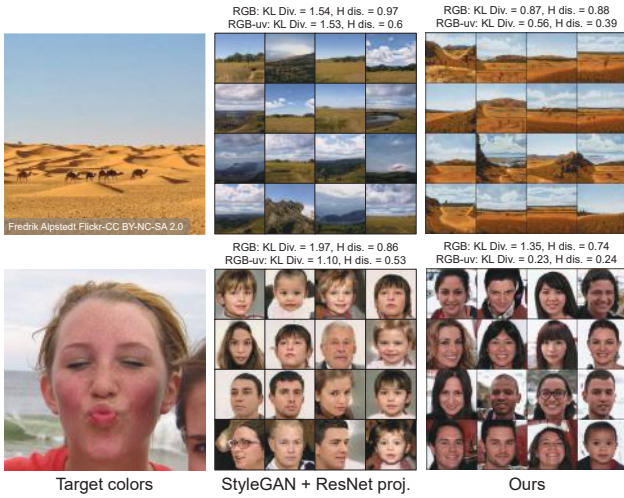


Figure S13: Comparison with generated images using StyleGAN [34] with latent space projection (see the main paper for more details) and our results.

Figure S11 shows an example of applying Pitié, and Kokaram’s method [50] as a post-processing color transfer to map the colors of the input image to the colors of our recolored image. In the shown figure, we also show the result of using the same color transfer method – namely, Pitié and Kokaram’s method [50] – to transfer the colors of the input image directly to the colors of the target image. As shown, the result of using our post-process strategy has a better perceptual quality.

Note that except for this figure (i.e., Fig. S11), we *did not* adopt this post-processing strategy to produce the reported results in the main paper or the supplementary mate-

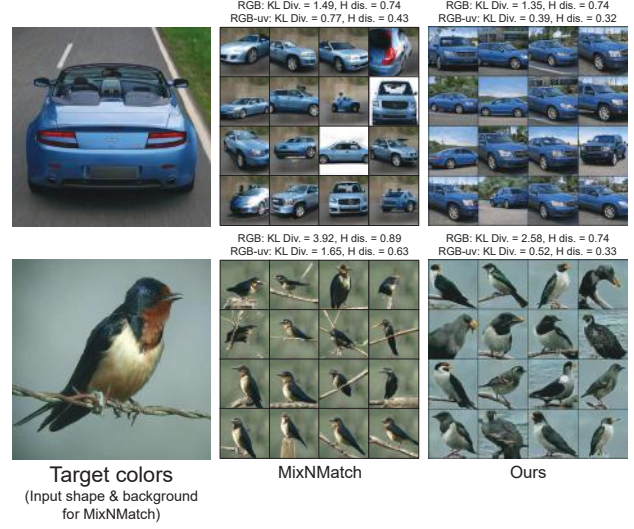


Figure S14: Additional comparison with the MixNMatch method [39]. In these examples, the target images were used as input shape and background images for the MixN-Match method.

rials. We discussed it here as a solution to reduce the potential color bleeding problem for completeness.

As our image-recoloring architecture is a fully convolutional network, we can process testing images in any arbitrary size. However, as we trained our models on a specific range of effective receptive fields (i.e., our input image size is 256), processing images with very high resolution may cause artifacts. To that end, we follow the post-processing approach used in [5] to deal with high-resolution images (e.g., 16-megapixel) without affecting the quality of the recolored image.

Specifically, we resize the input image to  $256 \times 256$  pixels before processing it with our network. Afterward, we apply the bilateral guided upsampling [14] to construct the mapping from the resized input image and our recoloring result. Then, we apply the constructed bilateral grid to the input image in its original dimensions. Figure S12 shows an example of our recoloring result for a high-resolution image ( $2048 \times 2048$  pixels). As can be seen, our result has the same resolution as the input image with no artifacts.

## 5.7. Additional Results

This section provides additional results generated by our HistoGAN and ReHistoGAN. As discussed in the main paper, we trained a regression ResNet [27] model to learn the back-projection from the generated images into the corresponding fine-style vectors of StyleGAN [34]. This regression model was used to compare HistoGAN and StyleGAN’s ability to control the generated images’ colors given a target color distribution. Figure S13 shows a qualita-



Figure S15: Our HistoGAN can be used to generate “unlimited” number of random samples, exactly like traditional StyleGANs [33, 34], by sampling from a pre-defined set of histograms to generate target histograms. In the shown figure, we show generated images by StyleGAN [33, 34] and our HistoGAN. In each row of the StyleGAN-generated images, we fixed the fine-style vector of the last two blocks of the StyleGAN, as these blocks are shown to control the fine-style of the generated image [34]. We also fixed the generated histogram for each row of our HistoGAN-generated images.

tive comparison between the results of our HistoGAN and StyleGAN with this projection approach. We show additional qualitative comparisons with the recent MixNMatch method [39] in Fig. S14. In the shown figures, we show the KL divergence and the Hellinger distance between the histograms of the GAN-generated images and the target his-

togram.

Our HistoGAN, along with the sampling procedure (used for auto recoloring in the main paper) can be used to turn our HistoGAN into a traditional GAN method, where there is no need for any user intervention to input the target histograms. Figure S15 shows an example of using our



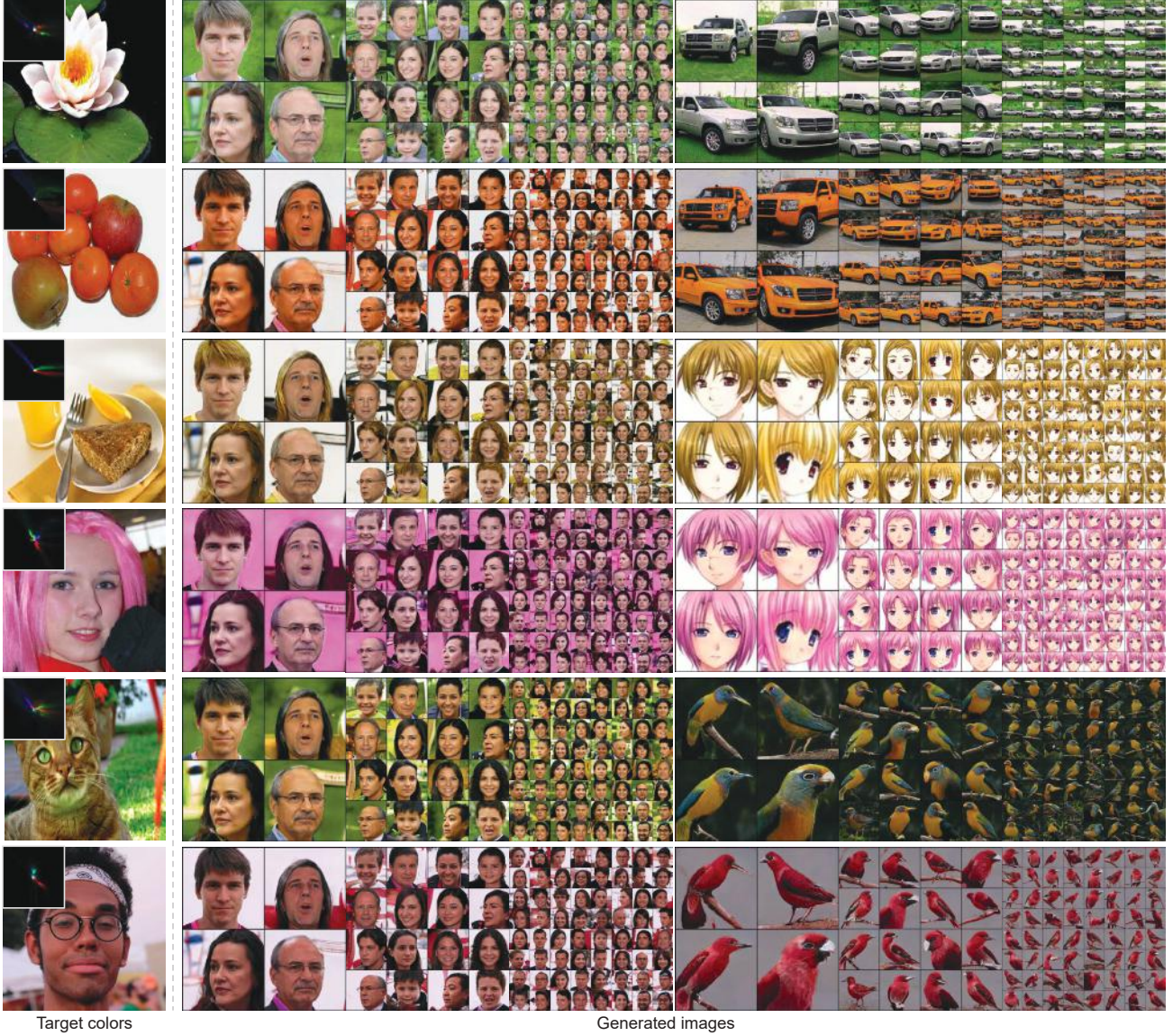


Figure S16: Additional examples of generated images using our HistoGAN. The target colors histograms were computed from the shown target images (left column).

sampling procedure to generate random histogram samples. The generated histogram samples are used by HistoGAN to generate “unlimited” number of samples. In the shown figure, we compare between our HistoGAN results, using the generated histograms, with StyleGAN [34]. In Fig.S15-(A), each row shows generated examples with a fixed fine-style vectors, which are used by the last two blocks of the StyleGAN as these blocks are shown to control the fine-style (e.g., colors, lighting, etc.) of the generated image [33,34]. In Fig.S15-(B), each row shows generated images using our HistoGAN with a fixed generated histogram. As shown in the figure, our HistoGAN generates samples with a higher color diversity compared to StyleGAN results.

Figure S16 shows additional HistoGAN-generated images from different domains. In each row, we show example images generated using the corresponding input target colors. We fixed the coarse- and middle-style vectors, for each domain, to show the response of our HistoGAN to changes in the target histograms.

In the main paper, we showed comparisons with different image recoloring and style transfer methods. Figure S17 shows additional qualitative comparisons. Note that Gaty et al.’s optimization method [23] takes  $\sim 4$  minutes to process a single image. In contrast, our ReHistoGAN processes a single image in  $\sim 0.5$  seconds without the guided upsampling procedure [14], and  $\sim 22$  seconds with an unoptimized im-





Figure S17: Additional comparisons with image recoloring/style transfer methods. We compare our results with results of the following methods: Reinhard et al., [52], Xiao et al., [63], Pitié and Kokaram [50], Nguyen et al., [46], and Sheng et al., [56].

plementation of the guided upsampling using a single GTX 1080 GPU. Further qualitative examples are shown in Fig. S18. As can be seen, our ReHistoGAN successfully transfers the target colors to the recolored images naturally.

As mentioned in the main paper, there are a few attempts to achieve auto recoloring (e.g., [7, 8, 18, 38]). The high-resolution daytime translation (HiDT) method [8], for example, achieves the auto-style transfer by sampling from a pre-defined set of target styles. We compared our method and the HiDT method in the main paper, where we used one of the pre-defined target styles as our target histogram. This idea of having a pre-defined set of target styles was originally proposed in [38], where a set of transient attributes are used to search in a dataset of different target styles. These methods, however, are restricted to the semantic content of the target styles to match the semantic content of training/testing input images. Unlike these auto recoloring/style transfer methods, our ReHistoGAN can deal with histograms taken from any arbitrary domain, as shown in our results in the main paper and these supplementary materials. In Fig. S19, we show qualitative comparisons of the recoloring results using our universal ReHistoGAN and the method proposed in [38].

Another strategy for image recoloring is to learn a diverse colorization model. That is, the input image is con-

verted to grayscale, and then a trained method for diverse colorization can generate different colored versions of the input image. In Fig. S20, we show a qualitative comparison with the diverse colorization method proposed by Deshpande et al., [18].

Lastly, we show additional qualitative comparisons with the recent auto-recoloring method proposed by Afifi et al., [7] in Fig. S21. The figure shows the results of domain-specific ReHistoGAN models (the first four rows) and the universal ReHistoGAN model (the last three rows). As can be seen from the shown figures, our ReHistoGAN arguably produces more realistic recoloring compared to the recoloring results produced by other auto recoloring methods.

## References

- [1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to embed images into the stylegan latent space? In *ICCV*, 2019. 2
- [2] Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. StyleFlow: Attribute-conditioned exploration of StyleGAN-generated images using conditional continuous normalizing flows. *arXiv preprint arXiv:2008.02401*, 2020. 2
- [3] Mahmoud Afifi. 11K hands: Gender recognition and biometric identification using a large dataset of hand images. *Mul-*



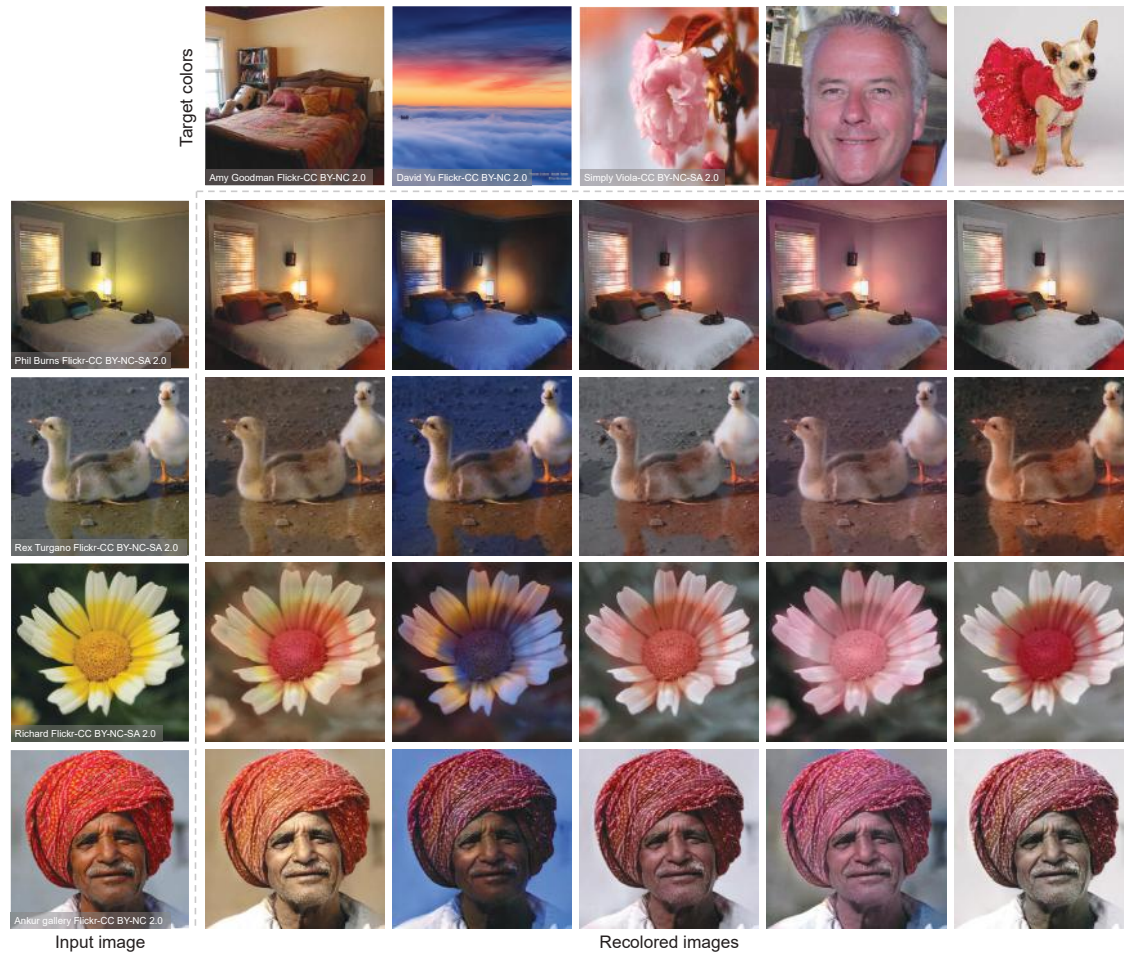


Figure S18: Additional results for image recoloring. We recolor input images, shown in the right by feeding our network with the target histograms of images shown in the top.

- imedia Tools and Applications*, 78(15):20835–20854, 2019. 7, 9, 10
- [4] Mahmoud Afifi and Michael S Brown. Sensor-independent illumination estimation for dnn models. In *BMVC*, 2019. 2, 3
  - [5] Mahmoud Afifi, Konstantinos G Derpanis, Björn Ommer, and Michael S Brown. Learning multi-scale photo exposure correction. In *CVPR*, 2021. 13
  - [6] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *CVPR*, 2019. 2, 3
  - [7] Mahmoud Afifi, Brian L Price, Scott Cohen, and Michael S Brown. Image recoloring based on object color distributions. In *Eurographics 2019 (short papers)*, 2019. 2, 8, 9, 16, 19
  - [8] Ivan Anokhin, Pavel Solovov, Denis Korzhenkov, Alexey Kharlamov, Taras Khakhulin, Aleksei Silvestrov, Sergey Nikolenko, Victor Lempitsky, and Gleb Sterkin. High-resolution daytime translation without domain labels. In *CVPR*, 2020. 2, 8, 16
  - [9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017. 2
  - [10] Mor Avi-Aharon, Assaf Arbelle, and Tammy Riklin Raviv. Deephist: Differentiable joint and color histogram layers for image-to-image translation. *arXiv preprint arXiv:2005.03995*, 2020. 2
  - [11] Jonathan T Barron. Convolutional color constancy. In *ICCV*, 2015. 2
  - [12] 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. 11
  - [13] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Transactions on Graphics (TOG)*, 34(4):139–1, 2015. 2
  - [14] Jiawen Chen, Andrew Adams, Neal Wadhwa, and Samuel W Hasinoff. Bilateral guided upsampling. *ACM Transactions on Graphics (TOG)*, 35(6):1–8, 2016. 13, 15
  - [15] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN V2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 2
  - [16] Spencer Churchill. Anime face dataset. <https://www.kaggle.com/splcher/animefacedataset>. [On-

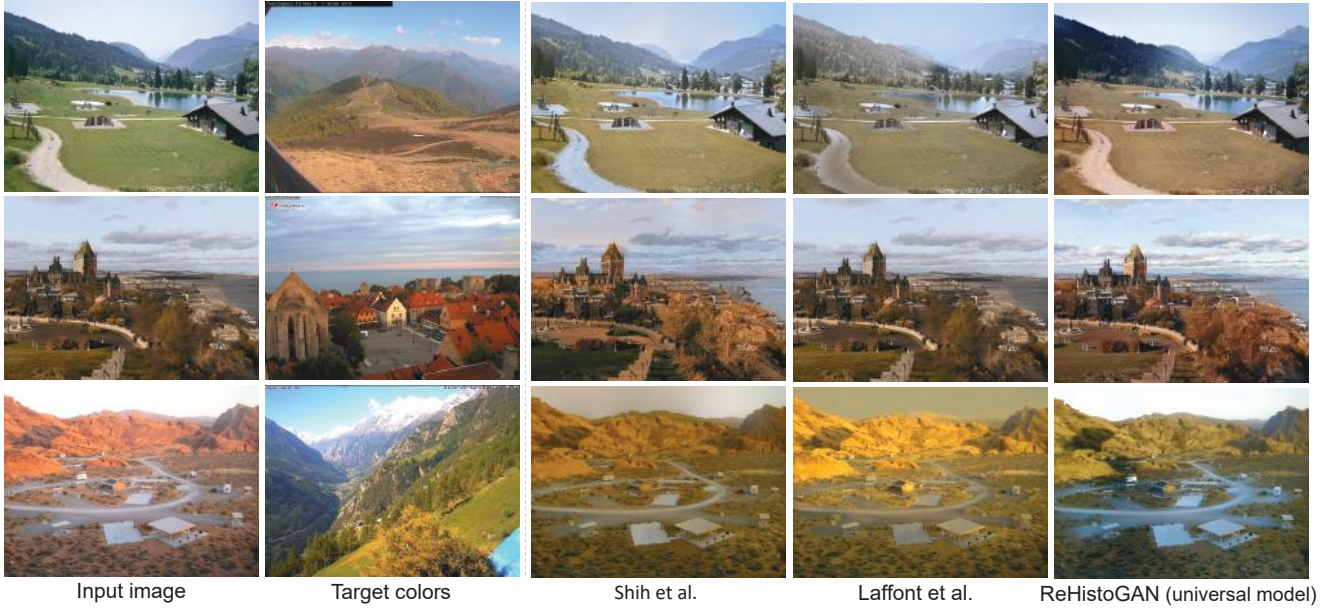


Figure S19: Comparisons between our universal ReHistoGAN and the methods proposed by Shih et al., [57] and Laffont et al., [38] for color transfer.



Figure S20: Comparisons between our ReHistoGAN and the diverse colorization method proposed by Deshpande et al., [18]. For our ReHistoGAN, we show the results of our domain-specific and universal models.

- line; accessed October 27, 2020]. 7, 9
- [17] Chris Crawford and NAIN. Cat dataset. <https://www.kaggle.com/crawford/cat-dataset>. [Online; accessed October 27, 2020]. 7, 9, 11
- [18] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. Learning diverse image colorization. In *CVPR*, 2017. 2, 16, 18
- [19] S. R. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, and B. B. Chaudhuri. diffGrad: An optimization method for convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4500–4511, 2020. 8, 9, 11
- [20] Eva Eibenberger and Elli Angelopoulou. The importance of the normalizing channel in log-chromaticity space. In *CIP*, 2012. 2
- [21] H Sheikh Faridul, Tania Pouli, Christel Chamaret, Jürgen Stauder, Erik Reinhard, Dmitry Kuzovkin, and Alain Trémeau. Colour mapping: A review of recent methods, extensions and applications. In *Computer Graphics Forum*, 2016. 1, 2
- [22] Graham D Finlayson and Steven D Hordley. Color constancy at a pixel. *JOSA A*, 2001. 2
- [23] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 2, 15
- [24] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 2, 8, 9
- [25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 6
- [27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7, 13
- [28] Mingming He, Jing Liao, Dongdong Chen, Lu Yuan, and Pedro V Sander. Progressive color transfer with dense seman-





Figure S21: Comparison with the auto image recoloring method proposed in [7]. Our recoloring results were produced by domain-specific networks, except for the last three rows, where our results were generated by the universal model.

- tic correspondences. *ACM Transactions on Graphics (TOG)*, 38(2):1–18, 2019. 2
- [29] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 7, 9, 10
- [30] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2
- [31] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2
- [32] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 2
- [33] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 2, 7, 8, 9, 11, 14, 15
- [34] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 1, 2, 3, 5, 6, 7, 8, 9, 13, 14, 15
- [35] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshops*, 2011. 7, 9, 11
- [36] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshops*, 2013. 7, 9, 11
- [37] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Alexander Kolesnikov, et al. The open images dataset V4. *International Journal of Computer Vision*, pages 1–26, 2020. 11

- [38] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on graphics (TOG)*, 33(4):1–11, 2014. 2, 16, 18
- [39] Yuheng Li, Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. MixNMatch: Multifactor disentanglement and encoding for conditional image generation. In *CVPR*, 2020. 2, 6, 13, 14
- [40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 11
- [41] Huidong Liu, Xianfeng Gu, and Dimitris Samaras. Wasserstein GAN with quadratic transport cost. In *ICCV*, 2019. 2
- [42] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 11
- [43] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *CVPR*, 2017. 2
- [44] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 3
- [45] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? The inria aerial image labeling benchmark. In *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017. 7, 9
- [46] Rang MH Nguyen, Seon Joo Kim, and Michael S Brown. Illuminant aware gamut-based color transfer. In *Computer Graphics Forum*, 2014. 1, 9, 16
- [47] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 7, 9, 11
- [48] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *SIGGRAPH*. 2003. 6
- [49] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020. 2
- [50] F. Pitie and A. Kokaram. The linear Monge-Kantorovitch linear colour mapping for example-based colour transfer. In *European Conference on Visual Media Production*, 2007. 9, 12, 13, 16
- [51] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2
- [52] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001. 1, 2, 9, 16
- [53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5
- [54] Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. COCO-FUNIT: Few-shot unsupervised image translation with a content conditioned style encoder. In *ECCV*, 2020. 2
- [55] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. In *ICCV*, 2019. 2
- [56] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. AvatarNet: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018. 2, 8, 9, 16
- [57] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):1–11, 2013. 18
- [58] Assaf Shocher, Yossi Gandelsman, Inbar Mosseri, Michal Yarom, Michal Irani, William T Freeman, and Tali Dekel. Semantic pyramid for image generation. In *CVPR*, 2020. 2
- [59] Maria Shugrina, Amlan Kar, Sanja Fidler, and Karan Singh. Nonlinear color triads for approximation, learning and direct manipulation of color distributions. *ACM Transactions on Graphics (TOG)*, 39(4):97–1, 2020. 2
- [60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 7
- [61] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2
- [62] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. 2011. 7, 9, 11
- [63] Xuezhong Xiao and Lizhuang Ma. Color transfer in correlated color space. In *International conference on Virtual reality continuum and its applications*, 2006. 1, 2, 9, 16
- [64] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 7, 9, 11
- [65] Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. Palette-based image recoloring using color decomposition optimization. *IEEE Transactions on Image Processing*, 26(4):1952–1964, 2017. 2
- [66] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 11
- [67] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 11