Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.056079

ARTICLE

Explicitly Color-Inspired Neural Style Transfer Using Patchified AdaIN

Bumsoo Kim[1], Wonseop Shin[2], Yonghoon Jung[1], Youngsup Park[3] and Sanghyun Seo[1,4,*]

[1]Department of Applied Art and Technology, Chung-Ang University, Anseong, 17546, Republic of Korea

[2]Department of Advanced Imaging Science Multimedia & Film, Chung-Ang University, Seoul, 06974, Republic of Korea

[3]Innosimulation Co., Ltd., Gangseo-gu, 07794, Republic of Korea

[4]School of Art and Technology, Chung-Ang University, Anseong, 17546, Republic of Korea

*Corresponding Author: Sanghyun Seo. Email: sanghyun@cau.ac.kr

Received: 13 July 2024 Accepted: 25 September 2024 Published: 31 October 2024

ABSTRACT

Arbitrary style transfer aims to perceptually reflect the style of a reference image in artistic creations with visual aesthetics. Traditional style transfer models, particularly those using adaptive instance normalization (AdaIN) layer, rely on global statistics, which often fail to capture the spatially local color distribution, leading to outputs that lack variation despite geometric transformations. To address this, we introduce Patchified AdaIN, a color-inspired style transfer method that applies AdaIN to localized patches, utilizing local statistics to capture the spatial color distribution of the reference image. This approach enables enhanced color awareness in style transfer, adapting dynamically to geometric transformations by leveraging local image statistics. Since Patchified AdaIN builds on AdaIN, it integrates seamlessly into existing frameworks without the need for additional training, allowing users to control the output quality through adjustable blending parameters. Our comprehensive experiments demonstrate that Patchified AdaIN can reflect geometric transformations (e.g., translation, rotation, flipping) of images for style transfer, thereby achieving superior results compared to state-of-the-art methods. Additional experiments show the compatibility of Patchified AdaIN for integration into existing networks to enable spatial color-aware arbitrary style transfer by replacing the conventional AdaIN layer with the Patchified AdaIN layer.

KEYWORDS

Neural style transfer; image synthesis; image stylization

1 Introduction

Arbitrary style transfer (AST) [1–5] is used to change the appearance of a content image by referencing an external style image considering characteristics such as the painting/rendering style, brush strokes, patterns, and colorization. AST has enabled remarkable and attractive algorithms [6–8] for various computer vision [9–11] and graphics applications [12–16] owing to its unique aesthetic effects [17–21]. AST was pioneered in [22] with convolutional neural networks [23] pretrained on a large image dataset [24] for an upstream task (e.g., image reconstruction or recognition) and a mathematical or statistical model that leverages deep features in every convolutional layer. This method optimizes style transfer by iteratively minimizing the designated objective. Although fast

patch-based optimization has been proposed [25], it requires optimizing every style transfer image, remaining a time-consuming method. On the other hand, a simple feedforward method is adaptive instance normalization (AdaIN) [26]. It can perform AST without requiring separate optimization by realigning the statistics of deep features from the content image (e.g., mean and standard deviation) to those of deep features extracted from the style (reference) image in a latent space. AdaIN achieves fast AST inference with low memory resources. Thus, it has been enhanced or specialized for various AST tasks [27,28] such as domain enhanced style transfer [29,30], fast inference [31] with low computational cost [5,32–35], three-dimensional scene style transfer [12,18,20,36,37], video-level style transfer [1,6,38,39], shape- or pattern-aware style transfer [27,40,41], and high visual fidelity [42].

As a result of pioneering studies [22,26,43,44], AST has been rapidly specialized for several computer vision and graphics applications [8,17,18,20,38] by storm. Despite its advancements, AST remains challenging, especially when trying to enhance the results for a given input image to reach the desired appearance. Most existing AST methods fail to capture geometric transformations of input images to reflect regional color distributions. This is because AdaIN [26] manipulates global statistics, including the mean (or bias factor) and standard deviation (or scaling factor) [41]. Thus, available methods often ignore geometric changes in the input image for AST. At the service level, this limitation can undermine the user experience because perceptually similar results are obtained regardless of appearance changes in the input image. For example, if a user wants to control the output in terms of color distribution by rotating the input style image, the AST results are the same as the initial result. It hinders the user to find optimal wanted output. Without considering the color-awareness in AST, user always face the same results even though they are struggling to obtain the best output while geometrically changing the input images. This issue poses significant limitations to advance well-designed AI technology to real-world service and applications. In this point of view, although satisfactory visual quality has been achieved in previous studies, this limitation is important and necessitated. To address this, additional methods are required, e.g., attention module [45,46], etc. Therefore, we tackle this color-awareness issue in AST in this paper.

To address the abovementioned limitation, we propose a patchified method[1] for color-aware style transfer based on AdaIN or Patchified AdaIN for short. It explicitly divides the features into small patches and performs AdaIN in a latent space to capture the regional color distribution. Hence, spatial information can be preserved after AdaIN (Fig. 1). As Patchified AdaIN has no learnable parameters, it can be easily integrated into state-of-the-art (SOTA) models without additional training or finetuning while providing color-aware AST. Additionally, the patchified operation can be modified according to patch levels and types to adjust and generate the desired output image. Moreover, a code blending scheme provides a control factor to determine the naturalness of global-local color weights during inference. The key contributions of the proposed method can be summarized as follows:

1. We introduce patchify-based AdaIN operation, dubbed as Patchified AdaIN that explicitly divides the content and style features in a latent space to then apply AdaIN to every patch. After Patchified AdaIN, the computed output is decoded using spatial recombination. Hence, the proposed method can preserve spatial color information, establishing the first color-aware AST method that can be embedded in SOTA models.
2. Patchified AdaIN has no learnable parameters and can be easily integrated into existing AST models to add color-awareness without requiring finetuning.

[1]Patch-based AST (StyleSwap) [25] should be distinguished from our method. StyleSwap leveraged patches to replace content patch to the closest-matching style patch, establishing a style swap for fast optimization, as explained in Section 2.1. On the other hand, the proposed method relies on patches to preserve the regional information of color distribution by applying AdaIN to each content/style patch, showing spatial correspondence during inference (Section 4.2).

3. The user experience of AST applications with Patchified AdaIN is enhanced by enabling geometric transformations of the input image to obtain the desired results by adjusting parameters such as the patch level and type.

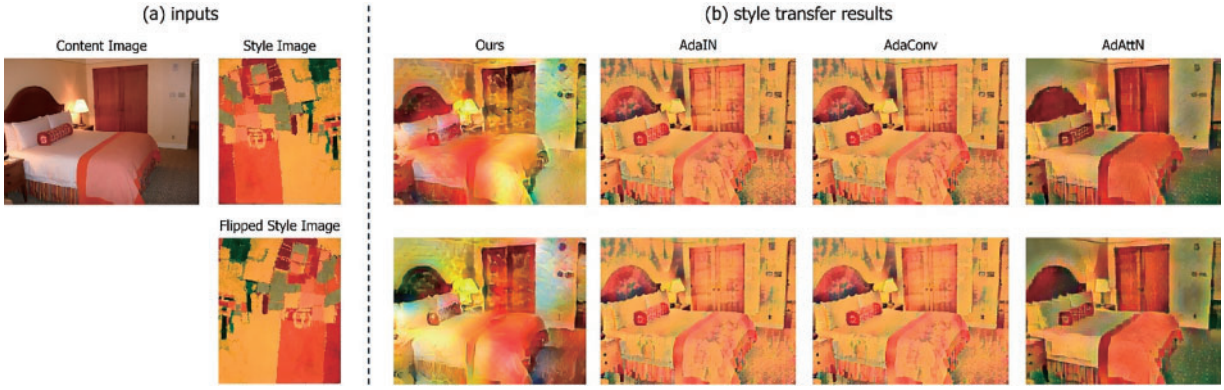

Figure 1: Abstracted our results with other network. Given geometrically transformed style image

2 Related Work

2.1 Arbitrary Style Transfer

Keeping pace with the development of deep neural networks, neural style transfer is active research topic. It redefines the style of a content image by referencing to a style image to achieve an artistic appearance, thus resembling artwork with an aesthetic appeal. AST emerged with an optimization method [22] based on pretrained networks from an upstream task. However, this method requires time-consuming iterative optimization for every style transfer image. To address this problem, various methods have been devised. In [44], perceptual losses are used for real-time style transfer. In [47], a hierarchical deep convolutional neural network is proposed for fast style transfer. In [25], patch-based style transfer for fast inference is introduced. In [48], generalized style transfer is achieved without compromising the visual quality with unseen styles. An unprecedented method is proposed in [26] to transfer the style using AdaIN in a single feedforward step for real-time style transfer. AdaIN enables fast and simple style transfer and has become a common and essential operation in style transfer. Hence, various studies have been focused on solving other important problems such as view or video style transfer [16,38,39,49,50], three-dimensional style transfer [19,20,36,37], text-driven style transfer [4,7,51,52], domain generalization [28,53], and domain enhancement [29,30]. More recently, various architecture including GANs [54], attention module [55] or CLIP [56] are leveraged for the controllability [57], fast style transfer [58], applications [59].

2.2 Something-Aware Style Transfer

Despite its high quality and applications, AST results may not be perceptually satisfactory in terms of user experience. To further improve AST, various methods implemented something-aware schemes to generate perceptually intuitive results. In [41], structure-aware style transfer with kernel prediction is proposed to enhance the structure style. In [42], visual fidelity is improved by using vector quantization with codebooks proposed in discrete image modeling [60,61]. For a relatively sparse shape or structure in an image, object shapes in an RGB (red-green-blue) image are considered with distance transform and patch matching modules for shape-aware style transfer. Structure guidance is used in [40] to focus on local regions. It enhances the content structure to avoid blurry or distorted

alignment by AdaIN. Recently, a pattern-aware scheme to discover the sweet spot of local and global style representations with pattern repeatability has been developed [62]. Previous studies have demonstrated intuitive outcomes using something-aware strategies for AST. However, user experience faces a common limitation. Existing AST methods generally fail to deliver the intended outcome if the user transforms the input style image, while the output image is expected to reflect any changes in the input images. To explore this limitation, we perform a pre-analysis of image transformation in Section 3.

3 Pre-Analysis

Consider a user manipulating a style transfer method in an exhibition hall as a representative AST example. The user might want to obtain various desired results from the same input image. However, this is difficult to achieve by manipulating the input images using an existing AST method. For instance, we can easily recognize that the user expects the appearance of resulting image to change when flipping or rotating the style or content image. In practice, user experience should be integrated into service-level applications. Nonetheless, most existing methods cannot address the abovementioned limitation because they cannot structurally reflect geometric changes in the input image.

To demonstrate this limitation through a short analysis, we applied geometric transformations (e.g., flipping, rotation, translation) to an input image. The images generated by AST methods always looked similar regardless of the applied transformation, as illustrated in Fig. 2. We first revisit the core component in style transfer. Simple sequential normalization and denormalization (i.e., conditional instance normalization) [26] is usually adopted as the main component for AST because representing style using statistics in the feature space is useful and effective. Given content image I_c and style image I_s, the resulting image has the following formulation in a latent space:

$$AdaIN\left(I_c, I_s\right) = \sigma\left(I_s\right) \times \frac{I_c - \mu\left(I_c\right)}{\sigma\left(I_c\right)} + \mu\left(I_s\right), \tag{1}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and standard deviation, respectively. When we transform the input images with transformation matrix $\mathcal{J}$, the output images can be expressed as $\mathcal{J} \times I_c$ and $\mathcal{J} \times I_s$. For simplicity, we denote $\mathcal{J} \times I_s$ as $I_{s,\mathcal{J}}$ and fix the content image with no transformations. From Eq. (1), the image statistics of the transformed style image can be expressed as $\mu\left(I_{s,\mathcal{J}}\right)$ and $\sigma\left(I_{s,\mathcal{J}}\right)$. To quantitatively compare the appearance of the output images according to the transformation of the style image, we calculated statistics of the output images with and without transformation (i.e., original style image) for 10,000 images.

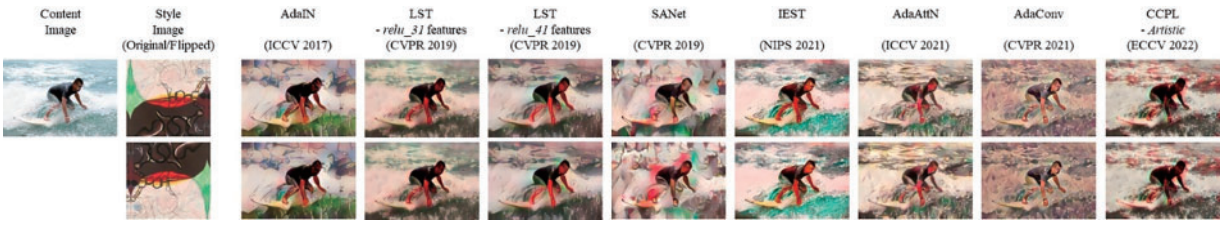

Figure 2: Limitation about image transformation. Regardless of style image transformation, the results always look the same

As a short experiment about the effect of geometric transformations on AST, we applied rotation, translation, and flipping to style images. As listed in Table 1, negligible differences in the statistics were obtained across the spatial transformations. More specifically, rotation provided a small difference

compared with the original statistics, while translation and flipping showed no changes in the image statistics because the absolute difference of the mean and standard deviation was zero. Hence, feature statistics after AdaIN provided highly similar representations regardless of spatial changes. Although the style image was drastically transformed in appearance, the resulting image was almost the same after AST. This can be explained by the image statistics before transformation, $\mu\left(I_{s,\mathcal{J}}\right)$ and $\sigma\left(I_{s,\mathcal{J}}\right)$, being almost equal to those after transformation, $\mu\left(I_s\right)$ and $\sigma\left(I_s\right)$. In other words, AdaIN globally aligns the image statistics without considering spatial distribution, thus failing to distinguish and ignoring spatial changes [41]. Considering the pre-analysis, we aimed to change statistics $\mu\left(I_{s,\mathcal{J}}\right)$ and $\sigma\left(I_{s,\mathcal{J}}\right)$ to reflect spatial changes. In Section 4.2, we explain how to expand global AdaIN to obtain variable local statistics according to spatial information.

Table 1: Comparison of image statistics between transformation and original. While style images are transformed with each corresponding $\mathcal{J}$ (e.g., rotation, translation, flip), content images are fixed as original image without transformation. Since the average $\boldsymbol{\mu}$ and average $\boldsymbol{\sigma}$ shows similar results for each RGB channel, we calculate the average about the results of 3 channels for simplicity

Transformations	Avg. $\mu(I_{s,\mathcal{J}})$	Avg. $\sigma(I_{s,\mathcal{J}})$	Abs. mean dist.	Abs. std. dist.
W/o transformation	103.43, 119.72, 133.55	11.08, 11.71, 12.27	.	.
Rotation $2/\pi$	103.87, 120.30, 134.22	11.50, 12.12, 12.70	0.44, 0.58, 0.67	0.42, 0.42, 0.43
Rotation π	103.43, 119.72, 133.55	11.08, 11.71, 12.27	0.00, 0.00, 0.00	0.00, 0.00, 0.00
Rotation random	103.50, 119.90, 133.84	10.23, 10.57, 10.96	0.07, 0.18, 0.29	0.85, 1.14, 1.31
Translation x, 0.5	103.43, 119.72, 133.55	11.08, 11.71, 12.27	0.00, 0.00, 0.00	0.00, 0.00, 0.00
Translation x, random	103.43, 119.72, 133.55	11.08, 11.71, 12.27	0.00, 0.00, 0.00	0.00, 0.00, 0.00
Flip x	103.43, 119.72, 133.55	11.08, 11.71, 12.27	0.00, 0.00, 0.00	0.00, 0.00, 0.00
Flip y	103.43, 119.72, 133.55	11.08, 11.71, 12.27	0.00, 0.00, 0.00	0.00, 0.00, 0.00

Note: Underline font means changed value compared from original value.

4 Proposed Method

From simple qualitative (Fig. 2) and quantitative (Table 1) analyses to unveil the limitations of existing AST methods, we identified the necessity to explicitly allow the statistics of a transformed image to change accordingly. To this end, various methods have been proposed for the transformed image, $I_{s,\mathcal{J}}$ to provide different statistics to those of the original image, I_s. Motivated by the findings in [63,64] and [27,62], we adopt an explicit method which allow each patch spatially divided from image to be participated into AdaIN operation. It is akin to strictly leverage a selected region in normalization process without considering another region within a whole image. By doing that, we argue that color preservation is simply yet effectively realized with easy-controllability, explainability, compatibility on other AdaIN-based style transfer networks. It has more efficacy than by incorporating advanced computationally large cost approach such as self-attention [64], contrastive learning [29], deformable convolution [41]. This explicit strategy has been shown to be simple and effective to handle local regions [25,27,62]. Our patch-based manipulation captures geometric transformations in the style image. We first divide the image into patches for segmenting the content and style images. Then, AdaIN is applied to every patch in a latent space. The obtained latent vectors are combined into an image with the original size in the latent space. Finally, the combined latent vector is decoded as the output image.

Additionally, in the latent space, we use code blending to mitigate non-plausible results when spatially combining the patches.

4.1 Pretraining Autoencoders

Before AST, autoencoders should be trained to perform reconstruction. As in [26], we adopt VGG-19 [23] as encoder E, which was pretrained on the MSCOCO dataset [24] and used to process the style image, and learnable decoder D with existing content and style loss functions. Unlike the method in [26], we do not use a shared encoder for the content and style images to avoid quality degradation that occurs by mapping the content and style images to the latent space using the same encoder [42].

As in [42], we first trained the content and style autoencoders for reconstruction on the MSCOCO [24] and WikiArt[2] datasets, respectively. For pretraining, each autoencoder was trained by reconstructing the input image as follows:

$$D_* \left(E_* \left(I_* \right) \right) = I_{*,rec}, \tag{2}$$

where $*$ is c for the content image and s for the style image. Each autoencoder uses the following reconstruction error:

$$\mathcal{L}_{rec} \left(I_* \right) = \left\| I_{*,rec} - I_* \right\|. \tag{3}$$

After pretraining, each encoder appropriately maps the corresponding image while considering the image characteristics. Among encoder and decoder for each image, we only fix the encoder for finetuning the decoder, as detailed in Section 4.3. During inference, we fix all the parameters in the encoders and decoder, except for the user-defined parameters.

4.2 Style Transfer via Patchified-AdaIN

We use the encoders for inference or finetuning by freezing the encoder weights. In detail, given input image pair (I_c, I_s), we generate style-transferred output image $\tilde{I}$. Considering a transformation, each input image I_* is transformed into $I_{*,\mathcal{J}}$ where $*$ is c or s, and $\mathcal{J}$ represents the transformation (i.e., rotation r, translations t_x and t_y, flipping along the horizontal f_h and vertical f_v, or zooming z) described in Section 5.1. The image is mapped onto the latent space by the encoder as follows:

$$z_c = E_c \left(I_{c,\tilde{\mathcal{J}}} \right), z_s = E_s \left(I_{s,\tilde{\mathcal{J}}} \right), \tag{4}$$

where $\tilde{\mathcal{J}}$ is a user-defined image transformation level, $z_* \in \mathbb{R}^{C_f \times H_f \times W_f}$ is the latent code (C_f is the number of channels, and H_f and W_f are the height and width of the encoded features), and E_c and E_s represent the pretrained content and style encoders, respectively. Then, the latent code is spatially divided into patches as follows:

$$\mathcal{C} = \mathbb{P} \left(z_c, M_h, M_w \right), \mathcal{S} = \mathbb{P} \left(z_c, M_h, M_w \right), \tag{5}$$

with $\mathbb{P} \left(z_*, M_h, M_w \right)$ being the patching layer formulated as:

$$\mathbb{P} \left(z_*, M_h, M_w \right) = [\mathrm{CH} \left(\mathrm{CH} \left(z_*, M_h, \mathrm{h} - \mathrm{axis} \right), \mathrm{w} - \mathrm{axis} \right)], \tag{6}$$

where CH is a chunk function. Hence, $\mathbb{P}$ divides latent code z_* into $M_h \times M_w$ patches, obtaining sets $\mathcal{C}$ and $\mathcal{S}$ of patches extracted from the content and style images, respectively. These sets can be represented as $\mathcal{C} = \{c_i\}_{i=1}^{M_h \times M_w}$, $\mathcal{S} = \{s_i\}_{i=1}^{M_h \times M_w}$ where c_i, $s_i \in \mathbb{R}^{C_f \times H_M \times W_M}$ are the i-th content and style patches, respectively, H_M and W_M are the dimensions of the divided patch calculated as $H_M = H_f / M_h$,

[2] https://www.wikiart.org/ (accessed on 24 September 2024).

$W_M = W_f / M_w$. We assume $M_h = M_w$ throughout the paper unless stated otherwise. AdaIN is applied to each patch with the corresponding positional patch in the other image calculated as:

$$\mathcal{T} = AdaIN_{\text{patch}}(\mathcal{C}, \mathcal{S}), t = AdaIN(z_c, z_s), \tag{7}$$

where t is the AdaIN output for code blending and $\mathcal{T}$ is AdaIN output set $\mathcal{T} = \{t_i\}_{i=1}^{M_h \times M_w}$ calculated by $AdaIN_{\text{patch}}(\mathcal{C}, \mathcal{S})$. As illustrated in Fig. 3, this operation can be formulated as:

$$AdaIN_{\text{patch}}(\mathcal{C}, \mathcal{S}) = \left\{ t_i | t_i = \sigma(s_i) \frac{c_i - \mu(c_i)}{\sigma(c_i)} + \mu(s_i) \right\}_{i=1}^{M}, \tag{8}$$

where $c_i \in \mathcal{C}$ and $s_i \in \mathcal{S}$. After applying AdaIN in the latent space, all the patches are aggregated and then blended using global feature t and local features $\mathcal{T}$ by user-defined parameter α as a global-local weight as follows:

$$\hat{z}_\alpha = (1 - \alpha) \mathbb{A}(\mathcal{T}) + \alpha t, \tag{9}$$

where $\hat{z}_\alpha$ is the final feature including user-defined factor α that sets the tradeoff between the global and local features. $\mathbb{A}(\cdot)$ is the de-patching layer that aggregates the input patch set while ensuring that $\mathbb{A}(\mathbb{P}(z_*, M_h, M_w)) = z_*$. During inference, the user can set the content image as the original one, that is, $I_{c,\mathcal{S}} = I_c$, while transforming the style image. Finally, output image $\tilde{I}_\alpha$ is obtained as $\tilde{I}_\alpha = D_s(\hat{z}_\alpha)$.

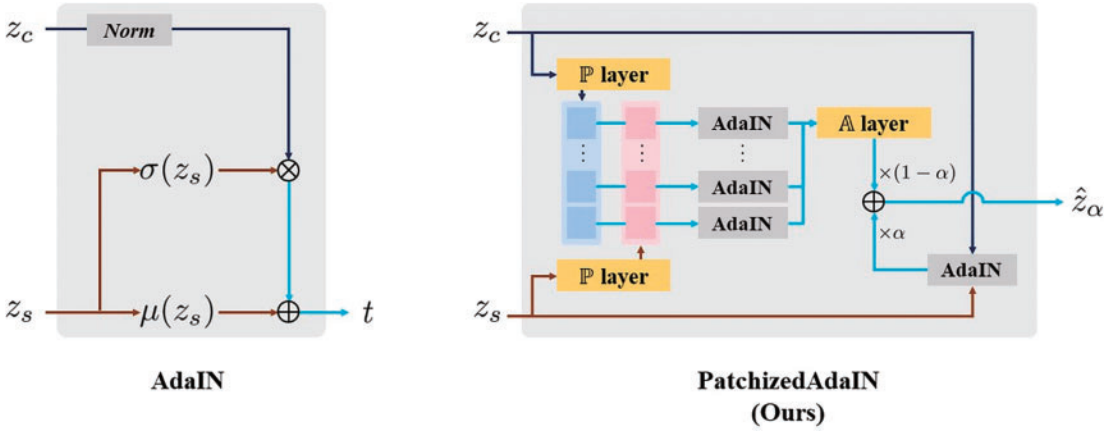

Figure 3: PatchizedAdaIN operation compared with AdaIN operation

4.3 Fine Tuning

The proposed Patchified AdaIN component can be easily used by replacing each AdaIN layer in an existing network without any other architecture modification or additional training or adaptation process. However, to enhance the regional quality of style transfer output, the decoder can be further tuned with an additional local loss using an existing loss function. To distinguish the loss functions, we refer to the existing AdaIN loss as the global loss throughout the paper. Similar to the global loss, our local loss is additionally computed from every patch of the content and style images. The local content loss is calculated as:

$$\mathcal{L}_{patch\_content} = \sum_{i=1}^{M} \|\mathcal{T} - \mathbb{P}(E_c(D_s(\mathbb{A}(\mathcal{T}))))\|_2, \tag{10}$$

where we omit the summation index for simplicity. Note that $\mathcal{T}$ and the output of $\mathbb{P}$ are sets including the patchified latent feature. In the local loss term for finetuning, we exclude code blending (i.e., $\alpha = 0$

in 9). Thus, $\hat{z}_\alpha$ becomes $\mathbb{A}(\mathcal{T})$ without considering the global features. Therefore, the local style loss can be calculated as:

$$\mathcal{L}_{patch\_style} = \sum_{i=1}^{M}\left(\sum_{j=1}^{L}\left\|\mu\left(\phi_j\left(D_s\left(\mathbb{A}\left(\mathcal{T}\right)\right)\right)\right)-\mu\left(\phi_j\left(I_s\right)\right)\right\|_2+\left\|\sigma\left(\phi_j\left(D_s\left(\mathbb{A}\left(\mathcal{T}\right)\right)\right)\right)-\sigma\left(\phi_j\left(I_s\right)\right)\right\|_2\right), \quad (11)$$

where $\phi_j(\cdot)$ represents the j-th layer in $E_s(\cdot)$ and L is the depth of the layer from $\phi_1(\cdot)$ in the encoder to be used as a loss. We calculate the content loss using relu4_1 and style loss as in [26]. We add the patch loss to the existing global loss [26]. Finally, the patch and global loss functions are formulated as follows:

$$\mathcal{L}_{patch} = \mathcal{L}_{patch\_content} + \lambda_{patch}\mathcal{L}_{patch\_style}, \mathcal{L}_{global} = \mathcal{L}_{global\_content} + \lambda_{global}\mathcal{L}_{global\_style}, \quad (12)$$

where λ is a style loss weight. We set every λ to 10 as in [26]. $\mathcal{L}_{global\_content}$ and $\mathcal{L}_{global\_style}$ are the same as the original loss functions of [26]. The complete loss, $\mathcal{L}_{total}$, is obtained by linearly combining the patch and global loss functions as follows:

$$\mathcal{L}_{total} = \gamma\mathcal{L}_{patch} + (1-\gamma)\mathcal{L}_{global}, \quad (13)$$

where γ is the patch loss scale. By changing γ, we can control the weight of the local spatial fidelity for finetuning. We set γ to 0.9. For plausible visual fidelity without explicit control, γ can be considered as a learnable parameter with an initial value of 1.0.

Additionally, we set M_h and M_w to 2 and obtain each patch by dividing the features, as shown in Fig. 4. Note that the user can control these factors (e.g., patch division strategy, number of patches), as analyzed in Section 5.5. Overall, the patch loss captures the AdaIN loss from local features s_i and c_i to obtain local statistics. Then, with the existing AdaIN loss [26], we weight the patch loss to set the contribution of the local style features to every patch. This affects not only backpropagation of the cost function during training but also inference for resulting image with AST, as analyzed in Section 5.4.

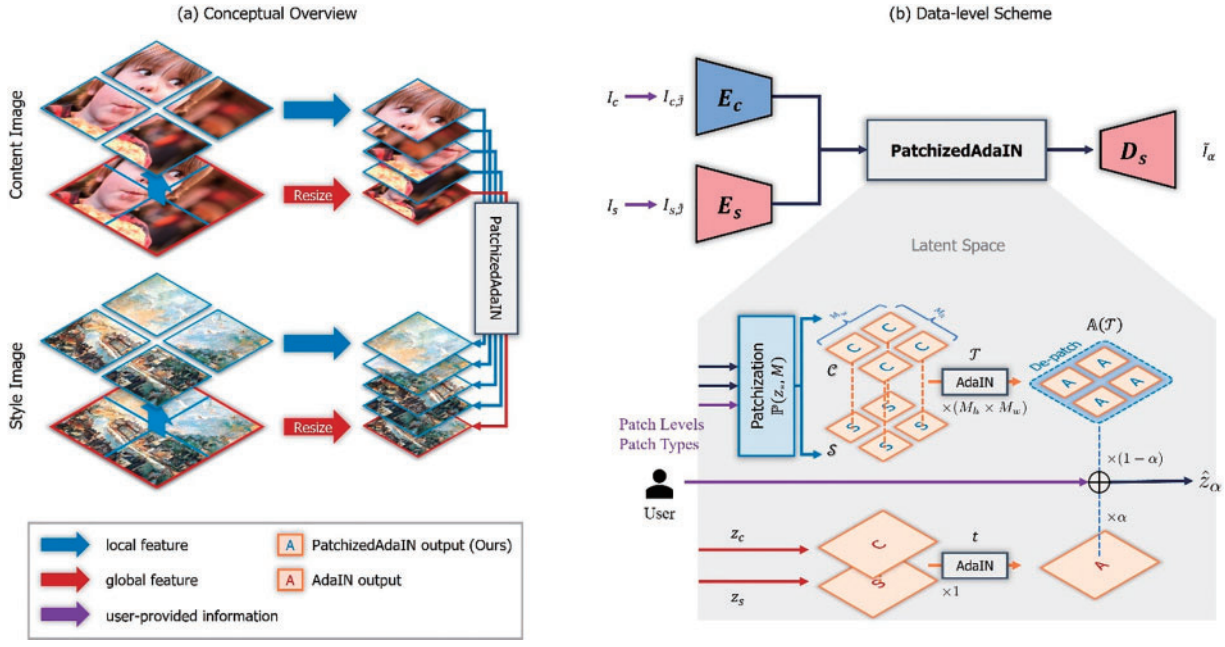

Figure 4: Proposed PatchizedAdaIN mechanism in inference step

4.4 Network Structure

We adopt the AdaIN network in [26] for the AST network as the baseline, but any other network relying on AdaIN can be used in our scheme by simply replacing AdaIN with the proposed Patchified AdaIN. For the encoder-decoder architecture, we use a nine-layer 3×3 filter convolution encoder f with up to 512 channels and decoder g with a symmetric architecture and the encoder based on [26]. The output image, $\tilde{I}$, is calculated as $\tilde{I} = D_s\left((1-\alpha)\,\mathbb{A}\left(\mathcal{T}\right) + \alpha t\right)$ for inference and $\tilde{I} = D_s\left(\mathbb{A}\left(\mathcal{T}\right)\right)$ for finetuning.

5 Experiments

We demonstrated the superiority of our method through comparisons with other models in terms of color-aware (i.e., geometric-transformation-aware) style transfer, and evaluated the influence of the number and types of patches on AST. First, we present the visual superiority of our proposal when transforming the style image in Section 5.1. Then, we obtain the color distributions before and after geometric transformations to show that our method visually reflects image transformations for AST in Section 5.2. For regional fidelity, we demonstrate finetuning results in Section 5.4. To evaluate user experience, we design a runtime control strategy for patching in Section 5.5. For practical application, we calculate the inference time by comparing our method with existing baselines in Section 5.8. Finally, we modify SOTA AST models using our Patchified AdaIN to integrate transformation-awareness, thereby demonstrating the extension and applicability provided by our proposal in Section 5.7. In training procedures, without patchization layer, we first pretrain the autoencoders for content and style, respectively. Then, the decoder of style autoencoder is further tuned based on Eq. (13).

5.1 Qualitative Evaluation under Geometric Transformations

To evaluate the capabilities of spatial-transformation-aware AST, we choose various geometric transformations that notably changed the regional color distribution. We applied four geometric transformations: rotation angle r, translations along the x axis, t_x, and y axis, t_y, flipping along the vertical, f_v, and horizontal, f_h, and zooming z. To prevent the global statistics of the image from excessively changing after a geometric transformation, we filled out empty areas with fictitious pixel insertion by, for example, reflecting symmetrical tiles for the rotation. Four cases were evaluated, with the transformation being given by $\mathcal{J} = \left(r, t_x, t_y, f_v, f_h, z\right)$, where r is a clockwise rotation angle from $-\pi$ to π, t_x and t_y are translation factors normalized from -1 to 1, f_v, and f_h, are flipping indicators being true or false, and z is a scaling and cropping factor from 0 to 0.5 with respect to the original image resolution. Examples of transformations are shown in Fig. 5.

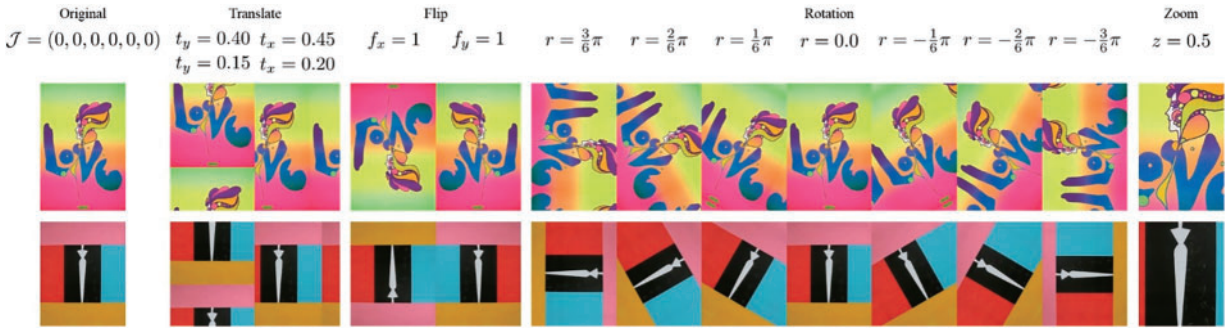

Figure 5: Geometric transformation cases

By applying predefined transformations, we performed AST using an original content image and transformed style image. For qualitative comparison, we selected AdaIN [26] as the baseline as well as LinearStyleTransfer (LST) [31], style-attentional network (SANET) [65], IEContraAST (IEST) [66], adaptive attention normalization (AdaAttN) [67], adaptive convolutions (AdaConv) [41], contrastive coherence preserving loss (CCPL) [30] as comparison networks. Because we focused on AST, we did not evaluate domain-enhanced (i.e., domain-specific) style transfer [29] and visual fidelity (whose results resemble image-to-image translation) [42]. As shown in Fig. 6, our method reflected the image transformations with varying color distributions in the output images. As expected from the pre-analysis (Section 3), no existing method could reflect the appearance change in the transformed style imaged, thus providing very similar results regardless of the applied transformation. On the other hand, the proposed method provided different results that reflected the transformations. For instance, our method generated an output image by locally reflecting the sky region (blue) in the style image in terms of spatial color distribution in the first row of Fig. 6. Similarly, the teddy bear showed varying red shades depending on the style image transformations in the fourth row, thus reflecting the variability provided by the proposed Patchified AdaIN.

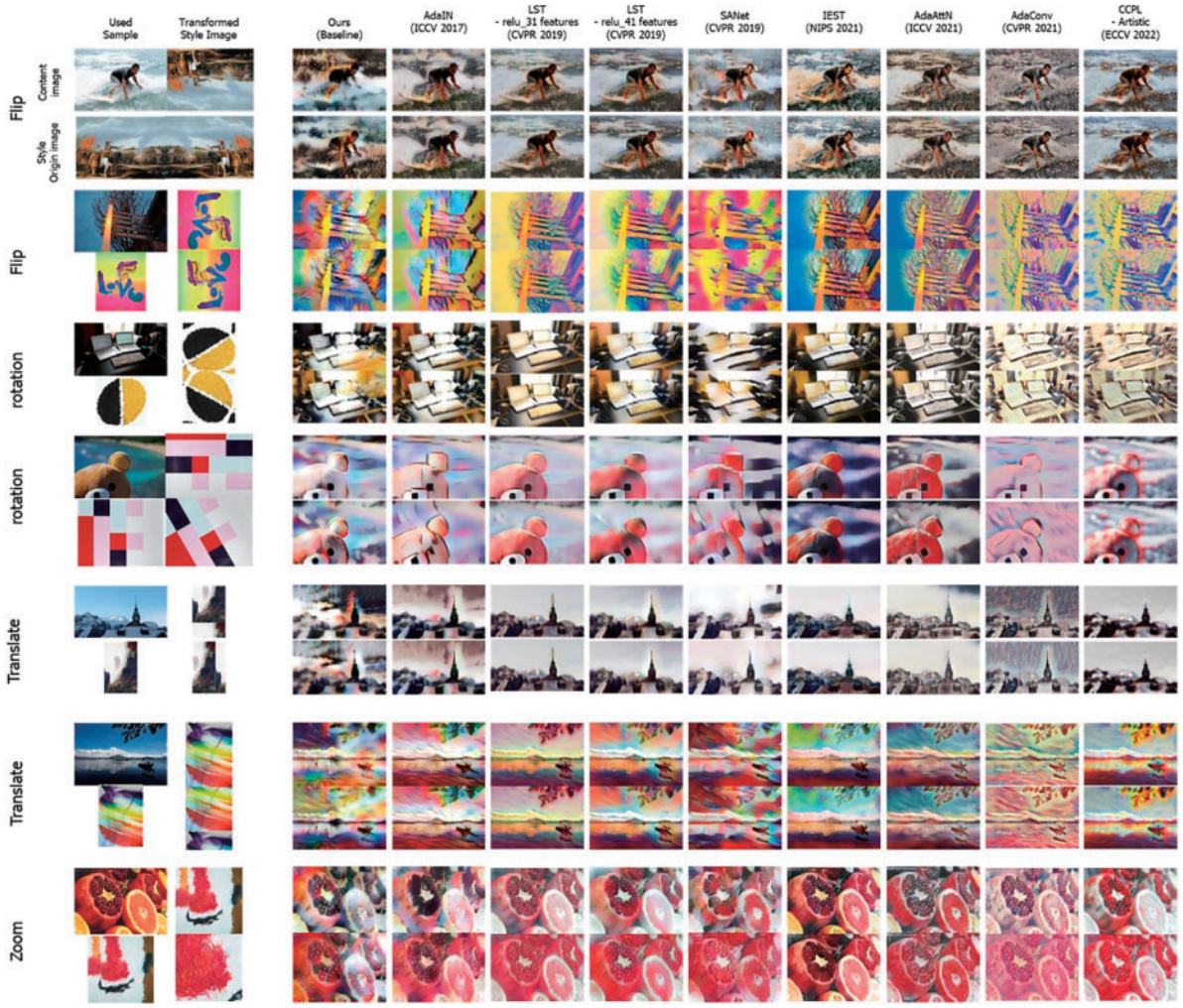

Figure 6: Qualitative comparison using our pre-defined geometric transformation cases

5.2 Quantitative Evaluation of Color-Awareness

The proposed Patchified AdaIN applies AdaIN to every spatially divided patch in the latent space. Nevertheless, it may generate images with a non-satisfactory appearance because the normalized latent code may be misaligned during decoding. To quantitatively evaluate this aspect, we measured the spatial color distribution according to the same patch division in the output image and used the same models as in Section 5.1 for comparison. The proposed method was finetuned using Eq. (13) for a value of $a = 0.0$ and $a = 0.5$ in 9. Here, we adopt two metrics: (a) spatial statistics (mean and variance), (b) Jensen-Shannon Divergence [68]. We used 1000 samples between the input style image and each output image per model. Statistical evaluation allows us to intuitively compare the distribution of pixel values. Meanwhile, JSD measure the difference of given two distributions, here image serve as a distribution. Concretely, we found that recent paper [68] leveraged Jensen-Shannon Divergence to measure the color distribution between two RGB images, we also adopt JSD metric in our quantitative evaluation. We used same images in statistical and JSD evaluation.

The quantitative results in Fig. 7 showed that our methods provided the lowest distance for the mean, indicating that our style transfer results had similar RGB mean with the style image among the evaluated models. On the other hand, the distance for the standard deviation was similar. Meanwhile, similar to statistical evaluation results (Fig. 7), JSD evaluation result showed that our method has best performance on reflecting spatial color distribution as shown Table 2. To determine whether this style statistics affected AST, we performed an evaluation considering human perception, as reported in Section 5.3.

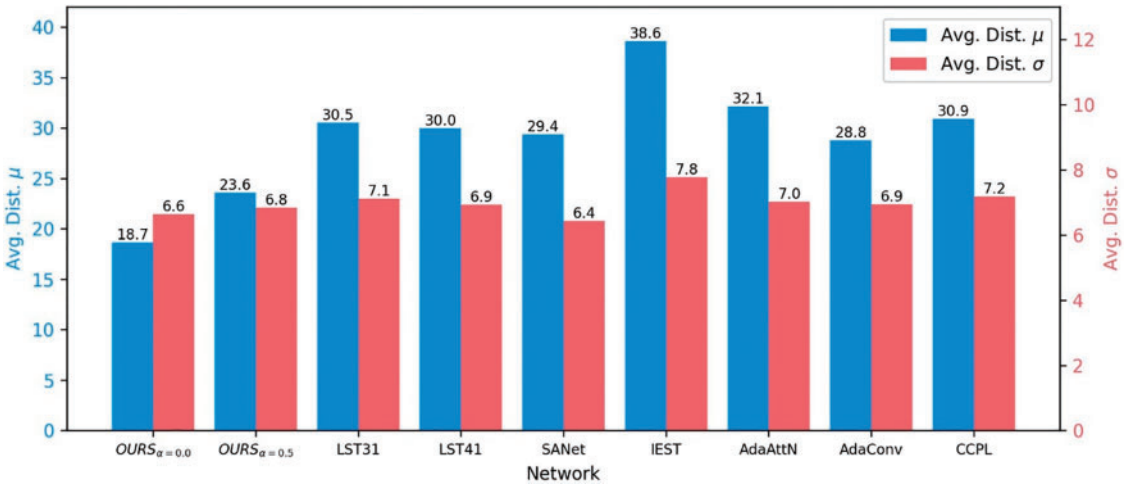

Figure 7: Quantitative evaluation results with statistical metric

Table 2: Quantitative evaluation results with JS-Divergence metric

Models	Ours$_{\alpha=0.0}$	Ours$_{\alpha=0.5}$	LST31	LST41	SANet	IEST	AdaAttN	AdaConv	CCPL
$JSD_R \downarrow$	0.077	0.105	0.103	0.102	0.096	0.398	0.094	0.293	0.449
$JSD_G \downarrow$	0.095	0.120	0.112	0.112	0.102	0.422	0.098	0.308	0.493
$JSD_B \downarrow$	0.080	0.111	0.108	0.109	0.102	0.429	0.097	0.302	0.487
$Avg.JSD \downarrow$	<u>0.084</u>	0.111	0.107	0.107	0.099	0.416	0.096	0.301	0.476

Note: <u>Underline</u> font means best model and ↓ indicates that lower value is better.

5.3 User Study

To evaluate human perception of color-aware AST, we conducted a user study considering style transfer outputs before and after transformation. We evaluated seven content images and two style images before and after applying transformations. We then conducted a seven-question survey considering random pairs of content and style images. One question involved 14 images, and it was responded for the outputs of the proposed and six comparison AST methods, obtaining 98 responses from 30 participants. We asked which output imaged better reflected the regional color distribution of the style image after transformation. The 30 participants included a junior artist, students enrolled in the art or computer science major, graphics/vision field researchers, and an immersive content creator. Every participant selected the best image among the outputs of all the methods considering the original and transformed style images for the same content image. From the user study, the proposed method achieved the best performance in terms of regional color distribution, as shown in Fig. 8. Almost half of the participants (45.71%) selected the proposed method as the best one among the evaluated methods. Remarkably, our proposal received dominant preference (>70%) for rotation and flipping transformations. On the other hand, regarding zooming with cropping, all the methods received similar proportions of preference. This was because the color distribution of the zoomed style images was distorted before AST, as illustrated in Fig. 6. Consequently, the output images of all the methods changed regardless of the ability to preserve the color distribution, causing highly variable preferences among the participants and similar preferences across methods after zooming the style image.

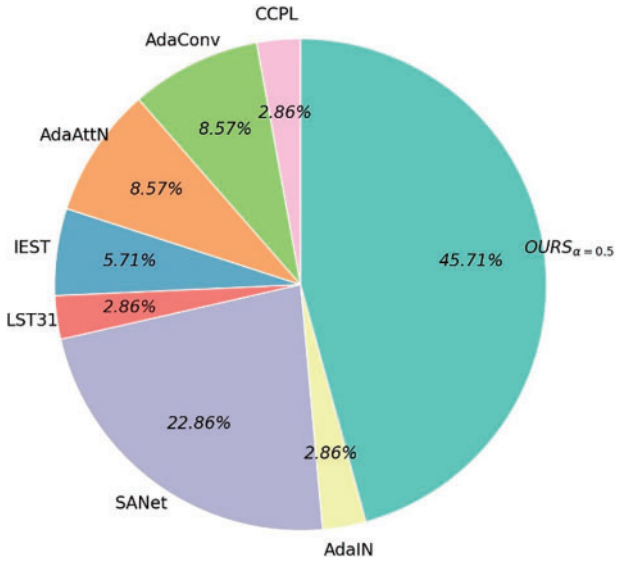

Figure 8: Result of user study in terms of color-distribution preservation

Along with the evaluation results presented in Section 5.2, we can conclude that the distance in standard deviation is not as important as that in mean regarding human perception of AST. In addition, the results in Figs. 7 and 8 suggest that the proposed method achieves SOTA performance and that the distance in mean allows to suitably evaluate the style similarity regarding human perception.

5.4 Local Image Style Transfer by Fine-Tuning

In addition to the existing AdaIN [26] training loss in Eq. (13), we introduce a local patch loss in Eq. (12). We evaluated the advantage of finetuning to highlight regional details in AST. We performed style transfer with two models, namely, a pretrained encoder-decoder replacing AdaIN with Patchified AdaIN without fine-tuning, whose weights were retrieved from an AdaIN PyTorch implementation [26], and a finetuned model considering our global-local loss in Eq. (13). As shown in Fig. 9, structurally distorted results were obtained without finetuning (Fig. 9b). Specifically, the boundaries between the cat and bathtub were blurry when using the pretrained model. In contrast, the finetuned model (Fig. 9a) preserved more structural details of the content image. Hence, finetuning enhanced the output image details without notably compromising the content image. Although blurry results may be artistically preferred in some contexts, preserving structural details is often preferred for human perception.

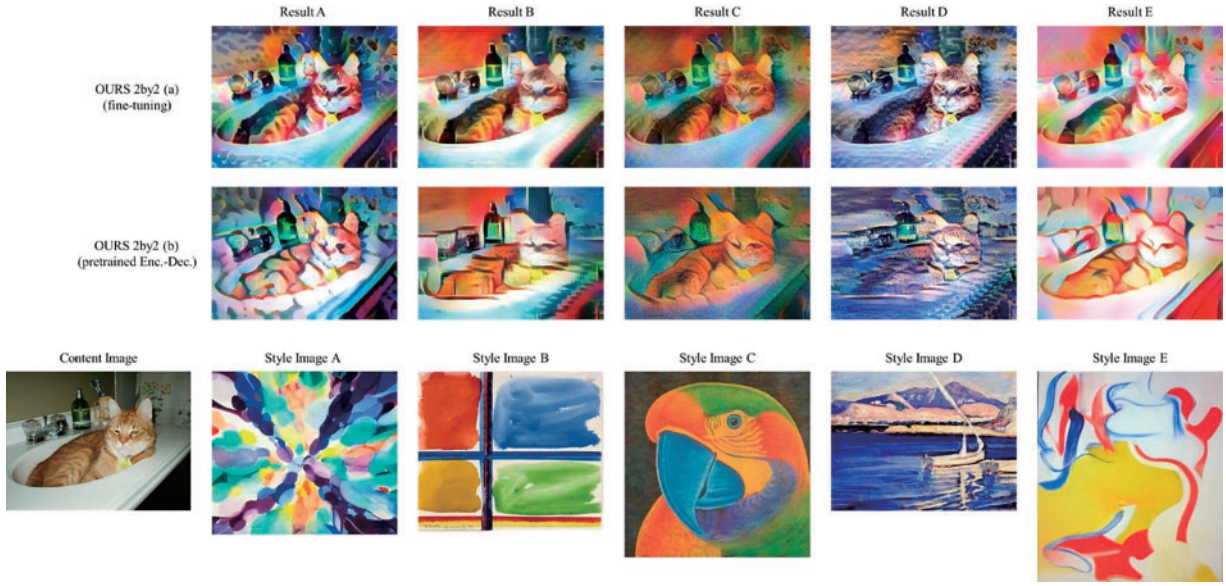

Figure 9: Structural details between w/ and w/o fine-tuning

5.5 Runtime Control for Patchization

To further evaluate the user experience, we explored runtime control strategies focusing on the Patchified AdaIN parameters of patch level and type. These controls may enhance the user experience during inference without requiring additional training.

5.5.1 Patch Level

Common segmentation into patches spatially divides features into 2 by 2 patches. As more patches may be useful in some cases, we evaluated the effects of increasing the patch number from $M_h = M_w = 2$ (default) to $M_h = M_w = 5$. Fig. 10 shows the corresponding results. When more patches were used, the structural details decreased for $\alpha = 0.0$. Hence, the apparent abstraction-level of the output increased because the model tried to preserve the regional color distribution. Thus, the structure of content image remained as edge information. On the other hand, the outputs reflecting global statistics ($\alpha \neq 0.0$) demonstrated a relatively clear and intuitive structure compared with patch-only AST, but the abstraction level of these results was highly dependent on the style image, i.e., they are unstable

and non-consistent. Nonetheless, we concluded that the local-global model (i.e., $\alpha = 0.5$) suitably preserved the structure of the content image regardless of the number of patches. Hence, a deep patch level led to unclear information near the content image edges. In addition, perspective or stereoscopic information easily collapsed for $a = 0.0$, likely providing a perceptually unpleasant impression from ambiguous results. We found that a model with H_f and W_f divided by 2 or 3 with $\alpha = 0.5$ provided appealing results in most cases. H_f and W_f divided by 4 or 5 can also be used, but these deep patch levels caused a global-local trade-off for $\alpha = 0.5$, impairing the user experience in terms of color-aware AST. Hence, we do not recommend using patch levels of 4 or 5. Patch level parameter do not require fine-tuning step, it can be simply used in inference step by user.

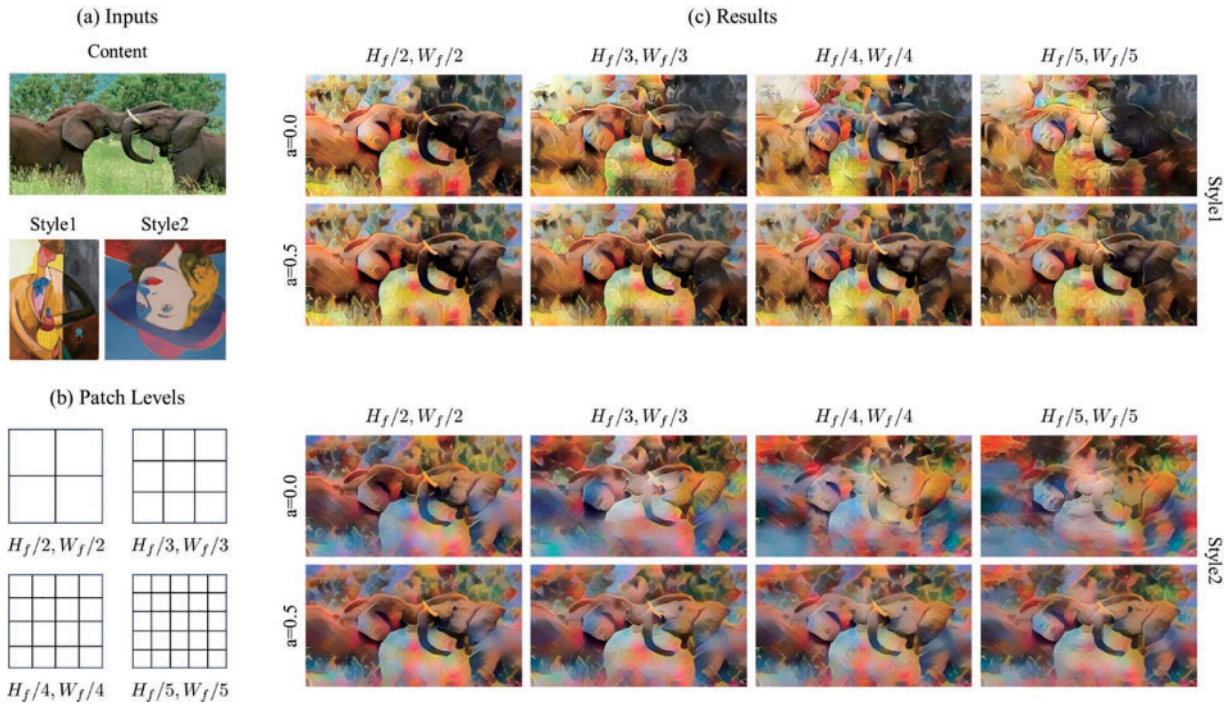

Figure 10: Stylized image according to different patch levels. In each patch levels (b), $\boldsymbol{M_h}$, $\boldsymbol{M_w}$ are 2, 3, 4 and 5

5.5.2 Patch Type

For patching in Patchified AdaIN, 2 by 2 division was the default setting (Fig. 3). However, different patch types can lead to different visual results. For evaluation, we applied three different patch types including the default one while preserving $M_h + M_w = 4$ in Eq. (5), obtaining the results shown in Fig. 11. In the feature space, we applied simple division methods to show the output changes according to the patch types, as shown in Fig. 11b. Type A was the most general method while and Types B and C corresponded to horizontal and vertical separation into patches, respectively. Fig. 11 shows that the patch types generated notably different output images. For instance, Type B provided a gloomy result, while Type C provided a bright sunny impression for Style 1. Hence, different impressions can be generated depending on the patch type for the same input image. Therefore, AST using the proposed method can provide a rich user experience. In fact, the user can flexibly select the runtime control parameters to achieve the desired style transfer results without additional training.

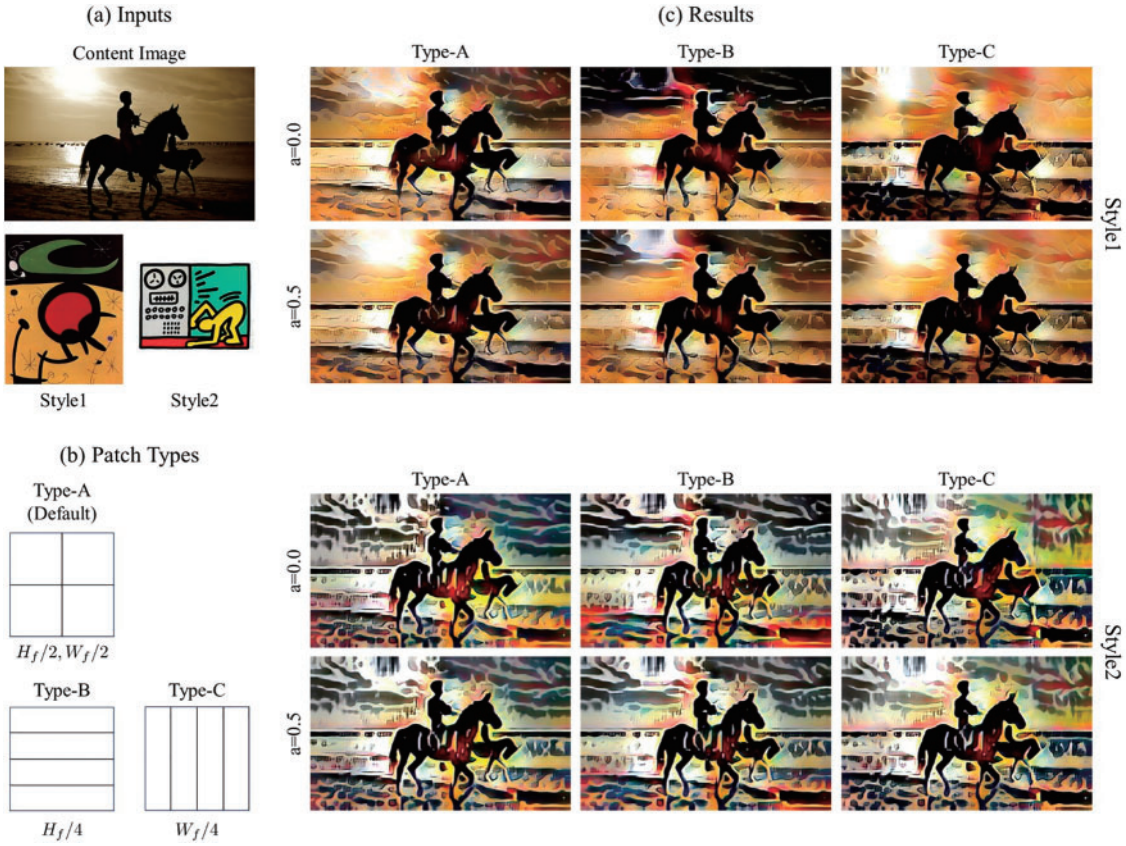

Figure 11: Stylized image according to different patch types. In Type-A, $(\boldsymbol{M}_h, \boldsymbol{M}_w) = (\mathbf{2}, \mathbf{2})$. Type-B and Type-C has $(\boldsymbol{M}_h, \boldsymbol{M}_w) = (\mathbf{4}, \mathbf{1})$ and $(\boldsymbol{M}_h, \boldsymbol{M}_w) = (\mathbf{1}, \mathbf{4})$, respectively

5.6 Inference Speed

To confirm the practicality of the proposed Patchified AdaIN, we measured the inference time according to different patch levels. The results were obtained using a computer equipped with an NVIDIA GeForce RTX 3090 GPU of 24 GB. We calculated were the average inference time over 1000 iterations.

We performed AST for patch levels from 2 to 5, as in Section 5.5. The baseline was the pretrained AdaIN model [26], and our method considered the finetuned model replacing AdaIN with Patchified AdaIN. For comparison, we normalized the AdaIN inference speed to 1. As listed in Table 3, our models had a negligible inference gap compared with the simpler baseline. Even the highest patch level provided a speed with a factor of 1.057. Hence, the proposed method incurs a small inference burden, suggesting that Patchified AdaIN can be integrated in SOTA models without notably compromising the inference speed.

5.7 Application to SOTA Models

As discussed in Section 5.8, Patchified AdaIN can be used in other models through simple replacement without notably increasing the inference time. We selected various SOTA AST models based on AdaIN for replacement with the proposed Patchified AdaIN. We aimed to demonstrate the extensive applicability of our method and the AST performance preservation while reflecting the regional

color distribution. For evaluation, we selected CCPL [30] with $\alpha = 0.0$ and LinearStyleTransfer [31] with $\alpha = 0.5$ for code blending using (9). Patchified AdaIN was integrated into the transformation or normalization block. As the evaluated models adopted different schemes for AST, we adapted Patchified AdaIN to their structure, establishing *modified* Patchified AdaIN implementations that preserved the core scheme of Patchified AdaIN of explicitly dividing the features into patches and aggregating them to obtain the output image. As shown in Fig. 12, the *modified* Patchified AdaIN endowed the SOTA models with the ability to perform spatially color-aware AST. The original SOTA models generated highly similar output images regardless of the transformed style image. After applying the *modified* Patchified AdaIN, the models provided different results reflecting the transformations, thus providing diverse appearances to the output images. These results demonstrate that our method is effective and easily applicable to other AST models without compromising performance. Nonetheless, it should be carefully considered when our method will be applied into existing networks.

Table 3: Inference speed according to the number of patches

Model	Time
AdaIN (baseline) [26]	1.000
Ours ($M_h = M_w = 2$)	1.005
Ours ($M_h = M_w = 3$)	1.008
Ours ($M_h = M_w = 4$)	1.031
Ours ($M_h = M_w = 5$)	1.057

Figure 12: Extension to SOTA models using modified PatchizedAdaIN. Our PatchizedAdaIN can be easily applied to other style transfer networks to provide the color-aware capability

5.8 Generalizability and Failure Case

For real-world use cases, it is important whether method can used in general and wide use cases, that is generalizailbity. To provide this capability, we provide additional experiment with some

images that have non-uniform and complex color distribution. Samples which have a high variance are chosen and we conduct style transfer under our model using those images to showcase potential generalizability in real-world use cases. As shown in Fig. 13, we observed that resulting stylized images has few visual changes even though different transformations are adopted. This issue can be the limitation of our method, but it might be addressed if user input the style images which have high contrast, vibrant color intensity. Meanwhile, our patchification strategy requires patch aggregation due to the spatial division in 2-dimensional space. Therefore, there is potential quality degradation in the edge of patch in aggregation step. Especially, if the structure of content images has semantically important information in the edge, artifacts in that region could prominently demonstrate visually non-harmonized appearance. Thus, this potential quality degradation should be carefully considered in inference step.

Figure 13: Failure cases on non-uniform and intricate color distribution. The resulting images may not be easily distinguishable despite various transformations when style images have complex color distribution

6 Conclusion

We propose a simple but effective scheme to reflect or preserve the color distribution of images that undergo transformations in AST. We divide the latent features into patches and aggregate them after applying AdaIN to every patch. This method can enlarge the user experience because Patchified AdaIN can generate different output images according to geometric transformations applied to the input images. Hence, the user can generate various desired results while using the same input image pairs. Additionally, we offer control over parameters to further increase the image variability and achieve the best results via iterative inference with different patch levels and types. A user study reveals that our method provides the preferred output images regarding human perception. Moreover, Patchified AdaIN can be easily integrated into existing models, demonstrating its applicability for extending existing AST models to preserve the spatial color distribution of transformed images. Overall, the proposed method achieves the best performance in terms of color-aware AST, establishing a SOTA approach without notable computational overhead during training and inference compared with baseline methods. However, there is a limitation in that ambiguous output occurs when the input image has a complex or non-uniform color distribution. This can be alleviated by having the end-users consider this technical limitation to achieve the desired appearance when using our algorithm in their applications.

Acknowledgement: This research was supported by the Chung-Ang University Graduate Research Scholarship in 2024.

Funding Statement: This research work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2022R1A2C1004657, Contribution Rate: 50%) and Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by Ministry of Culture Sports and Tourism in 2024 (Project Name: Developing Professionals for R&D in Contents Production Based on Generative Ai and Cloud, Project Number: RS-2024-00352578, Contribution Rate: 50%).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Bumsoo Kim, Sanghyun Seo; data collection and comparative experiments: Bumsoo Kim, Wonseop Shin, Yonghoon Jung; analysis and interpretation of results: Bumsoo Kim, Youngsup Park, Sanghyun Seo; draft manuscript preparation: Bumsoo Kim, Sanghyun Seo. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All the data used and analyzed is available in the manuscript.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Huang HP, Tseng HY, Saini S, Singh M, Yang MH. Learning to stylize novel views. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021; p. 13869–78. doi:10.1109/ICCV48922.2021.01361.
2. Chen Y, Yuan Q, Li Z, Liu Y, Wang W, Xie C, et al. UPST-NeRF: universal photorealistic style transfer of neural radiance fields for 3D scene; 2022. arXiv preprint arXiv: 220807059.
3. Wang Z, Zhang Z, Zhao L, Zuo Z, Li A, Xing W, et al. AesUST: towards aesthetic-enhanced universal style transfer. In: Proceedings of the 30th ACM International Conference on Multimedia, 2022; Lisbon, Portugal, p. 1095–106. doi:10.1145/3503161.3547939.
4. Liu ZS, Wang LW, Siu WC, Kalogeiton V. Name your style: an arbitrary artist-aware image style transfer; 2022. arXiv preprint arXiv: 220213562.
5. Wu X, Hu Z, Sheng L, Xu D. StyleFormer: real-time arbitrary style transfer via parametric style composition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021; p. 14618–27. doi:10.1109/ICCV48922.2021.01435.
6. Gao W, Li Y, Yin Y, Yang MH. Fast video multi-style transfer. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020; Snowmass Village, CO, USA. doi:10.1109/WACV45572.2020.9093420.
7. Kwon G, Ye JC. CLIPstyler: image style transfer with a single text condition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022; New Orleans, LA, USA; p. 18062–71. doi:10.1109/CVPR52688.2022.01753.
8. Zhang S, Su S, Li L, Lu J, Zhou Q, Chang CC. CSST-Net: an arbitrary image style transfer network of coverless steganography. Vis Comput. 2022;38:1–13. doi:10.1007/s00371-021-02272-6.
9. Zhang Y, Huang N, Tang F, Huang H, Ma C, Dong W, et al. Inversion-based style transfer with diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023; Vancouver, BC, Canada; p. 10146–56. doi:10.1109/CVPR52729.2023.00978.
10. Luan F, Paris S, Shechtman E, Bala K. Deep photo style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017; Honolulu, Hawaii; p. 4990–8. doi:10.1109/CVPR.2017.740.

11. Luo X, Han Z, Yang L, Zhang L. Consistent style transfer; 2022. arXiv preprint arXiv: 220102233.
12. Mu F, Wang J, Wu Y, Li Y. 3D photo stylization: learning to generate stylized novel views from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; New Orleans, LA, USA; p. 16273–82. doi:10.1109/CVPR52688.2022.01579.
13. Lu M, Xu F, Zhao H, Yao A, Chen Y, Zhang L. Exemplar-based portrait style transfer. IEEE Access. 2018;6:58532–42. doi:10.1109/ACCESS.2018.2874203.
14. Xie X, Li Y, Huang H, Fu H, Wang W, Guo Y. Artistic style discovery with independent components. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022; New Orleans, LA, USA; p. 19870–9. doi:10.1109/CVPR52688.2022.01925.
15. Huang S, Xiong H, Wang T, Wen B, Wang Q, Chen Z, et al. Parameter-free style projection for arbitrary image style transfer. In: ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022; Singapore, IEEE; p. 2070–4. doi:10.1109/ICASSP43922.2022.9746290.
16. Chen D, Yuan L, Liao J, Yu N, Hua G. Stereoscopic neural style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018; p. 6654–63. doi:10.1109/CVPR.2018.00696.
17. Han F, Ye S, He M, Chai M, Liao J. Exemplar-based 3D portrait stylization. IEEE Transact Visual Comput Graph. 2021;29(2):1371–83. doi:10.1109/TVCG.2021.3114308.
18. Höllein L, Johnson J, Nießner M. Stylemesh: style transfer for indoor 3D scene reconstructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; New Orleans, LA, USA; p. 6198–208. doi:10.1109/CVPR52688.2022.00610.
19. Chiang PZ, Tsai MS, Tseng HY, Lai WS, Chiu WC. Stylizing 3D scene via implicit representation and hypernetwork. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022; Waikoloa, HI, USA; p. 1475–84. doi:10.1109/WACV51458.2022.00029.
20. Zhang K, Kolkin N, Bi S, Luan F, Xu Z, Shechtman E, et al. ARF: artistic radiance fields. In: European Conference on Computer Vision, 2022; Tel Aviv, Israel, Springer; p. 717–33. doi:10.1007/978-3-031-19821-2_41.
21. Ma Y, Zhao C, Li X, Basu A. RAST: restorable arbitrary style transfer via multi-restoration. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023; Waikoloa, HI, USA; p. 331–40. doi:10.1145/3638770.
22. Gatys LA, Ecker AS, Bethge M. Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016; Las Vegas, NV, USA; p. 2414–23. doi:10.1109/CVPR.2016.265.
23. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition; 2014. arXiv preprint arXiv: 14091556.
24. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: common objects in context. In: European Conference on Computer Vision, 2014; Zurich, Switzerland, Springer; p. 740–55. doi:10.1007/978-3-319-10602-1_48.
25. Chen TQ, Schmidt M. Fast patch-based style transfer of arbitrary style; 2016. arXiv preprint arXiv: 161204337.
26. Huang X, Belongie S. Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision, 2017; Venice, Italy; p. 1501–10. doi:10.1109/ICCV.2017.167.
27. Hong K, Jeon S, Lee J, Ahn N, Kim K, Lee P, et al. AesPA-Net: aesthetic pattern-aware style transfer networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023; Paris, France; p. 22758–67. doi:10.1109/ICCV51070.2023.02080.
28. Chen J, Jiang M, Dou Q, Chen Q. Federated domain generalization for image recognition via cross-client style transfer. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023; Waikoloa, HI, USA; p. 361–70. doi:10.1109/WACV56688.2023.00044.

29. Zhang Y, Tang F, Dong W, Huang H, Ma C, Lee TY, et al. Domain enhanced arbitrary image style transfer via contrastive learning. In: ACM SIGGRAPH, 2022 Conference Proceedings, 2022; Vancouver, BC, Canada; p. 1–8. doi:10.1145/3528233.3530736.
30. Wu Z, Zhu Z, Du J, Bai X. CCPL: contrastive coherence preserving loss for versatile style transfer. In: European Conference on Computer Vision, 2022; Tel Aviv, Israel, Springer; p. 189–206. doi:10.48550/arXiv.2207.04808.
31. Li X, Liu S, Kautz J, Yang MH. Learning linear transformations for fast arbitrary style transfer; 2018. arXiv preprint arXiv: 180804537.
32. Jing Y, Liu Y, Yang Y, Feng Z, Yu Y, Tao D, et al. Stroke controllable fast style transfer with adaptive receptive fields. In: Proceedings of the European Conference on Computer Vision (ECCV), 2018; Munich, Germany. doi:10.1007/978-3-030-01261-8_15.
33. Puy G, Perez P. A flexible convolutional solver for fast style transfers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019; Long Beach, CA, USA. doi:10.1109/CVPR.2019.00917.
34. Huang Y, Jing M, Zhou J, Liu Y, Fan Y. LCCStyle: arbitrary style transfer with low computational complexity. IEEE Trans Multimedia. 2021;25. doi:10.1109/TMM.2021.3128058.
35. Wang Z, Zhao L, Zuo Z, Li A, Chen H, Xing W, et al. MicroAST: towards super-fast ultra-resolution arbitrary style transfer. Proc AAAI Conf Artif Intell. 2023;37:2742–50. doi:10.1609/aaai.v37i3.25374.
36. Zhang Z, Liu Y, Han C, Pan Y, Guo T, Yao T. Transforming radiance field with lipschitz network for photo-realistic 3D scene stylization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023; Vancouver, BC, Canada; p. 20712–21. doi:10.1109/CVPR52729.2023.01984.
37. Huang YH, He Y, Yuan YJ, Lai YK, Gao L. StylizedNeRF: consistent 3D scene stylization as stylized nerf via 2D-3D mutual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; New Orleans, LA, USA; p. 18342–52. doi:10.1109/CVPR52688.2022.01780.
38. Wang W, Yang S, Xu J, Liu J. Consistent video style transfer via relaxation and regularization. IEEE Transact Image Process. 2020;29:9125–39. doi:10.1109/TIP.2020.3024018.
39. Deng Y, Tang F, Dong W, Huang H, Ma C, Xu C. Arbitrary video style transfer via multi-channel correlation. Proc AAAI Conf Artif Intell. 2021;35(2):1210–7. doi:10.1609/aaai.v35i2.16208.
40. Atarsaikhan G, Iwana BK, Uchida S. Guided neural style transfer for shape stylization. PLoS One. 2020;15(6):e0233489. doi:10.1371/journal.pone.0233489.
41. Chandran P, Zoss G, Gotardo P, Gross M, Bradley D. Adaptive convolutions for structure-aware style transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 2021; p. 7972–81. doi:10.1109/CVPR46437.2021.00788.
42. Huang S, An J, Wei D, Luo J, Pfister H. QuantArt: quantizing image style transfer towards high visual fidelity. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023; Vancouver, BC, Canada; p. 5947–56. doi:10.48550/arXiv.2212.10431.
43. Jing Y, Liu X, Ding Y, Wang X, Ding E, Song M, et al. Dynamic instance normalization for arbitrary style transfer. Proc AAAI Conf Artif Intell. 2020;34(4):4369–76. doi:10.1609/aaai.v34i04.5862.
44. Johnson J, Alahi A, Fei-Fei L. Perceptual losses for real-time style transfer and super-resolution. In: Computer Vision–ECCV 2016: 14th European Conference, October 11–14, 2016, Amsterdam, The Netherlands, Springer; p. 694–711. doi:10.1007/978-3-319-46475-6_43.
45. Vaswani A. Attention is all you need. In: Advances in neural information processing systems 30 (NIPS 2017). Long Beach, CA, USA; 2017.
46. Zhang H, Goodfellow I, Metaxas D, Odena A. Self-attention generative adversarial networks. In: International Conference on Machine Learning, 2019; Long Beach, CA, USA: PMLR; p. 7354–63. doi:10.48550/arXiv.1805.08318.

47. Wang X, Oxholm G, Zhang D, Wang YF. Multimodal transfer: a hierarchical deep convolutional neural network for fast artistic style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017; Honolulu, HI, USA; p. 5239–47. doi:10.1109/CVPR.2017.759.
48. Li Y, Fang C, Yang J, Wang Z, Lu X, Yang MH. Universal style transfer via feature transforms. Adv Neural Inf Process Syst. 2017;30:385–95. doi:10.5555/3294771.3294808.
49. Tseng KW, Lee YC, Chen CS. Artistic style novel view synthesis based on a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; New Orleans, LA, USA; p. 2258–62. doi:10.1109/CVPRW56347.2022.00248.
50. Kong X, Deng Y, Tang F, Dong W, Ma C, Chen Y, et al. Exploring the temporal consistency of arbitrary style transfer: a channelwise perspective. IEEE Trans Neural Netw Learn Syst. 2023;35:8482–96. doi:10.1109/TNNLS.2022.3230084.
51. Jandial S, Deshmukh S, Java A, Shahid S, Krishnamurthy B. Gatha: relational loss for enhancing text-based style transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2023; Vancouver, BC, Canada; p. 3546–51. doi:10.1109/CVPRW59228.2023.00362.
52. Fu TJ, Wang XE, Wang WY. Language-driven artistic style transfer. In: European Conference on Computer Vision, 2022; Tel Aviv, Israel, Springer; p. 717–34. doi:10.1007/978-3-031-20059-5_41.
53. Zhang Y, Li M, Li R, Jia K, Zhang L. Exact feature distribution matching for arbitrary style transfer and domain generalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022; New Orleans, LA, USA; p. 8035–45. doi:10.1109/CVPR52688.2022.00787.
54. Zheng X, Yang X, Zhao Q, Zhang H, He X, Zhang J, et al. CFA-GAN: cross fusion attention and frequency loss for image style transfer. Displays. 2024;81:102588. doi:10.1016/j.displa.2023.102588.
55. Ge B, Hu Z, Xia C, Guan J. Arbitrary style transfer method with attentional feature distribution matching. Multimed Syst. 2024;30(2):96. doi:10.21203/rs.3.rs-3365364/v1.
56. Suresh AP, Jain S, Noinongyao P, Ganguly A, Watchareeruetai U, Samacoits A. FastCLIPstyler: optimisation-free text-based image style transfer using style representations. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024; Waikoloa, HI, USA; p. 7316–25. doi:10.1109/WACV57701.2024.00715.
57. Kim S, Min Y, Jung Y, Kim S. Controllable style transfer via test-time training of implicit neural representation. Pattern Recognit. 2024;146:109988. doi:10.1016/j.patcog.2023.109988.
58. Du X, Jia N, Du H. FST-OAM: a fast style transfer model using optimized self-attention mechanism. Signal, Image Video Process. 2024;18:1–13. doi:10.1007/s11760-024-03064-w.
59. Li Ha, Wang L, Liu J. Application of multi-level adaptive neural network based on optimization algorithm in image style transfer. Multimed Tools Appl. 2024;83:1–23. doi:10.1007/s11042-024-18451-1.
60. Van Den Oord A, Vinyals O, Kavukcuoglu K. Neural discrete representation learning. Adv Neural Inform Process Syst. 2017;30:6309–18. doi:10.48550/arXiv.1711.00937.
61. Esser P, Rombach R, Ommer B. Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021; p. 12873–83. doi:10.1109/CVPR46437.2021.01268.
62. Liu S, Zhu T. Structure-guided arbitrary style transfer for artistic image and video. IEEE Transact Multimed. 2021;24:1299–312. doi:10.1109/TMM.2021.3063605.
63. Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017; Honolulu, HI, USA; p. 1125–34. doi:10.1109/CVPR.2017.632.
64. He K, Chen X, Xie S, Li Y, Dollár P, Girshick R. Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; NewOrleans, LA, USA; p. 16000–9. doi:10.1109/CVPR52688.2022.01553.

65. Park DY, Lee KH. Arbitrary style transfer with style-attentional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019; Long Beach, CA, USA; p. 5880–8. doi:10.1109/CVPR.2019.00603.
66. Chen H, Wang Z, Zhang H, Zuo Z, Li A, Xing W, et al. Artistic style transfer with internal-external learning and contrastive learning. Adv Neural Inform Process Syst. 2021;34:26561–73. doi:10.5555/3540261.3542295.
67. Liu S, Lin T, He D, Li F, Wang M, Li X, et al. AdaAttN: revisit attention mechanism in arbitrary neural style transfer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021; p. 6649–58. doi:10.1109/ICCV48922.2021.00658.
68. Liu Y, Zhao H, Chan KC, Wang X, Loy CC, Qiao Y, et al. Temporally consistent video colorization with deep feature propagation and self-regularization learning. Comput Visual Media. 2024;10(2):375–95. doi:10.1007/s41095-023-0342-8.