Adaptive Convolutions for Structure-Aware Style Transfer

Prashanth Chandran[1,2] Gaspard Zoss[1,2] Paulo Gotardo[1] Markus Gross[1,2] Derek Bradley[1]

1) DisneyResearch|Studios, Zurich
2) Department of Computer Science, ETH Zurich

{chandrap,gaspard.zoss,grossm}@inf.ethz.ch, {paulo.gotardo,derek.bradley}@disneyresearch.com

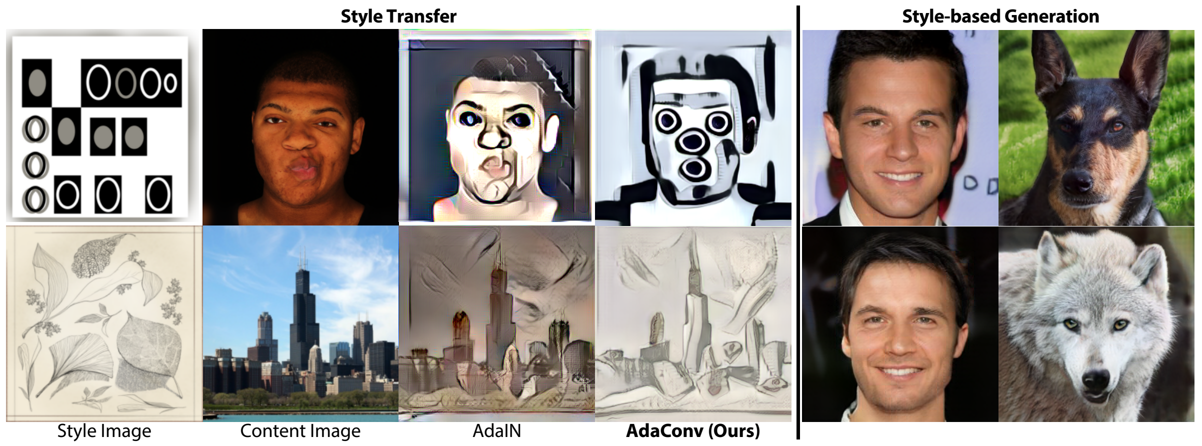

Figure 1: We propose Adaptive Convolutions (AdaConv), an extension of Adaptive Instance Normalization (AdaIN) for image style transfer, which is able to transfer both statistical and *structural* style elements. AdaConv can also be applied to generative models such as StyleGAN for photorealistic image synthesis on a multitude of datasets[1].

Abstract

Style transfer between images is an artistic application of CNNs, where the 'style' of one image is transferred onto another image while preserving the latter's content. The state of the art in neural style transfer is based on Adaptive Instance Normalization (AdaIN), a technique that transfers the statistical properties of style features to a content image, and can transfer a large number of styles in real time. However, AdaIN is a global operation; thus local geometric structures in the style image are often ignored during the transfer. We propose Adaptive Convolutions (AdaConv), a generic extension of AdaIN, to allow for the simultaneous transfer of both statistical and structural styles in real time. Apart from style transfer, our method can also be readily extended to style-based image generation, and other tasks where AdaIN has already been adopted.

1. Introduction

In recent years, convolutional neural networks (CNNs) have been used to explore and manipulate the *style* of images. Image style is often defined by image features such as overall color and local structure of brush strokes, in the context of paintings, or the pose and expression of a face in generative image applications. Style is also defined at different resolutions, and thus can include both the global identity of a face as well as the local structure of freckles on the skin. Research in this area gained a lot of momentum with the advent of *neural style transfer*, originally proposed by Gatys *et al.* [8], where a CNN is trained to reproduce the content of one input image, but rendered with the style of another image. In a similar spirit, generative adversarial networks (GANs) have been used to produce realistic synthetic images with style defined by a random vector input, for example in the creation of synthetic face images [18].

The widespread approach for manipulating style is through adaptive instance normalization (AdaIN), a method that transforms the mean and variance of image features. For example, AdaIN is often used to transfer feature statistics of a style image onto a content image. Since its definition by Huang *et al.* in 2017 [13], this operation has already become commonplace in CNN-based image manipulation literature. One major drawback of AdaIN, however, is that the statistic computation is a global operation; thus localized spatial structure in the style cannot be effectively captured and transferred. A concrete example is shown in Fig. 1 (row 1) where the style image has distinct features like black and white circles and squares. The AdaIN result transfers the statistics of that image to the content images, but the result lacks any structure of the style. A similar phenomenon can be seen in row 2, for a different style image.

In this work we introduce an extension to AdaIN called *Adaptive Convolutions (AdaConv)*, which allows for the simultaneous adaptation of both statistical and *structural* style. In the context of style transfer, instead of transferring a simple pair of global statistics (mean and standard deviation) from each style feature, our approach estimates full convolution kernels and bias values from the style image, which are then convolved on the features of the content image. As these kernels better capture localized spatial structure in the style, AdaConv can more faithfully transfer structural elements of the style image to the content image, as illustrated in Fig. 1 (columns 4 and 7).

The concept of predicting convolution kernels for deep learning tasks has already shown some promise in fields such as video frame interpolation [26, 27, 28] and denoising [1, 35]. Here we leverage this idea to extend AdaIN for more general image style manipulation. AdaConv can replace AdaIN in virtually every application where the latter has been adopted, providing a new, generic building block in CNN-based image generation and style manipulation. To illustrate the generality of AdaConv, we demonstrate its application in both style transfer as well as style-based generative face modeling (StyleGAN [18]).

2. Related Work

This section reviews prior work in the domains of neural style transfer, modulation layers in generative models and kernel prediction that are more closely related to our work.

Neural Style Transfer based on CNNs was initially proposed by Gatys *et al*. [8]. While their method allowed for transferring arbitrary style between images, it was based on an slow optimization procedure. Johnson *et al*. [17] addressed this issue with the introduction of perceptual losses, allowing for a significant speed-up of the optimization and achieving real-time results. In parallel, Ulyanov *et al*. [33] proposed a style transfer method that speeds up inference even further, by evaluating a feed-forward neural network that is style-specific and pre-trained. In a follow up work [34], they also replaced batch normalization (BN) layers by instance normalization (IN) to produce higher quality results without impacting speed. To improve control over the style transfer result, Gatys *et al*. [9] subsequently introduced explicit color, scale and spatial control by reformulating the loss function in both optimization-based and feed-forward methods [9]. Following up on the idea of IN, Dumoulin *et al*. [7] proposed conditional instance normalization (CIN) and conditioned the normalization layer on the style, allowing a single model to perform style transfer from one of 32 pre-defined styles or their interpolation. Ghiasi *et al*. [11] further extended CIN to allow transfers to arbitrary styles, unseen at training time; this was achieved using a large corpus of styles to train an encoder that transforms a style image into the conditioning latent vector. Cheng *et al*. [6] proposed patched-based style swap method for arbitrary style transfer. Concurrently, Huang *et al*. [13] proposed a method for arbitrary style transfer, by effectively making IN adaptive to the mean and standard deviation of the style features, thus leading to AdaIN. Li *et al*. [22] extended this method by whitening and coloring the latent features given the style. This idea was further extended by Sheng *et al*. [31] with a style decorator module and multi-scale style adaptation. Other works also looked at meta networks for style transfer [30], faster style transfer using learned linear transformations [21] and style transfer of stereoscopic images [4]. More recently, Jing *et al*. [15] noticed that directly replacing the statistics of the content features with those of the style features may be sub-optimal; instead, their dynamic instance normalization (DIN) method trains a style encoder to output the new statistics for the content features, while also adjusting the size and sampling locations of subsequent convolutions layer. Besides instance normalization, adversarial learning was also explored by Kotovenko *et al*. [20] to better disentangle style from content. Additional in-depth descriptions of other *Neural Style Transfer* methods are presented in the recent review paper by Jing *et al*. [16]. The aim of our work is to further extend AdaIN by predicting entire convolution kernels and biases according to the style image, to transfer both the statistics as well as the local structure of the style.

Modulation layers in generative models have also contributed to other breakthroughs beyond style transfer. Most notably, StyleGAN [18] uses the original version of AdaIN, but the input style statistics are predicted by an MLP from a Gaussian noise vector. To mitigate some visible artifacts caused by AdaIN, StyleGAN2 [19] replaces it with a weight demodulation layer, which only normalizes and modulates the standard deviation, without changing the mean. As AdaIN and its variants only transform global statistics, they are insensitive to localized, spatial semantics in the style input. To address this limitation, new methods have been proposed to predict spatially-varying normalization parameters from an input spatial layout image [29, 39, 15]. SPADE [29] replaces the global affine tranformation of AdaIN with per-pixel transformations regressed from an input semantic mask. SEAN [39] further extends SPADE by considering an additional style vector with the input layout mask. Both SPADE and SEAN preserve the conditioning spatial layout for the purpose of semantic image generation; they effectively control how each kernel is emphasized or suppressed at particular image locations. In contrast, our AdaConv method generates completely new kernels at test time. Also, SPADE and SEAN are not directly suitable for application in style transfer, where the spatial layout of the content image must be preserved.

Kernel prediction has also been explored in previous work. Note that all of the above methods for feature normalization and modulation follow a similar procedure: they define scalar affine transformations that are applied on each feature channel independently. The main differences are found in: (i) whether or not the transformation parameters are hand-crafted, learned during training, or predicted at test time; and (ii) whether the per-channel transformations are global or spatially-varying. Those methods that regress global transformations can also be understood as predicting 1×1 2D kernels at test time. For style transfer, Chen *et al*. [3, 5] learn style-specific filter banks that are convolved on the content image's features. Their method is limited to filter banks learned at training time; it cannot generate new kernels for unseen styles given only at test time. Jing et. al [15] claim to be able to regress dynamic convolutions from the input, using their generic DIN blocks; however, the reported experimental results are limited to 1×1 transformations. Related work on kernel prediction also goes beyond style transfer. Jia *et al*. [14] present dynamic convolutions for video and stereo image prediction, in which test-time features are reshaped into new filters that are applied either convolutionally or in a location-specific way. State-of-the-art methods for denoising Monte Carlo renderings [1, 35, 10] use neural networks to predict dynamic kernels applied to reconstruct the final denoised frame. Neural networks were also proposed to predict denoising kernel for natural images taken in *burst* mode with a handheld camera [24, 36]. Niklaus *et al*. [26] predict *frame interpolation kernels* for video; they later extended this work to predict separable convolution parameters [27, 28]. Xue *et al*. [37] use a CNN to predict *motion kernels* from random Gaussian variables used to synthetic plausible next frame. Esquivel *et al*. [38] predict *adaptive kernels* which are used to reduce the number of layers required to accurately classify images under limited computational resources. In the remainder of this paper, we explore a similar idea that leverages kernel prediction at test time to improve style transfer and style-based modulation in generative models.

3. Feature Modulation with AdaConv

We now describe AdaConv and our kernel predictors, showing how AdaConv generalizes and extends the typical 1×1 affine transformations in style-based feature modulation. We begin by drawing a parallel with AdaIN, in the context of style transfer, and then show how AdaConv allows for better conditioning on local feature structure, for better transfer of spatial style, while also being applicable in high-quality generative models outside style transfer.

3.1. Overview

Consider the usual style representation $\{a, b\} \in \mathbb{R}^2$, where a and b represent the style as scale and bias terms, respectively (*e.g.*, for style transfer, a and b are the mean and standard deviation of the style image features). Given an input feature channel with values $x \in \mathbb{R}$ and the desired style, AdaIN applies a style-defined affine transformation to normalized input features,

$$\mathrm{AdaIN}(x; a, b) = a\left(\frac{x - \mu_x}{\sigma_x}\right) + b\,, \tag{1}$$

where μ_x and σ_x are the mean and standard deviation over the feature channel. Thus, AdaIN changes only the global statistics of each channel based on the conditioning style parameters $\{a, b\}$. Note that the whole channel is modulated equally, regardless of the *spatial distribution* (structure) of the feature values around each sample x.

Our first step towards extending AdaIN is thus to introduce a conditioning *2D style filter* $\mathbf{f} \in \mathbb{R}^{k_h \times k_w}$, replacing the scale term and yielding extended style parameters $\{\mathbf{f}, b\}$. This filter allows for modulating the feature channel in a spatially-varying way, according to the local structure in a neighborhood $\mathcal{N}(x)$ around sample x,

$$\begin{aligned}\mathrm{AdaConv}_{\mathrm{dw}}(x; \mathbf{f}, b) &= \sum_{x_i \in \mathcal{N}(x)} f_i \left(\frac{x_i - \mu_x}{\sigma_x}\right) + b\,, \quad (2)\\ &= \sum_{x_i \in \mathcal{N}(x)} \mathrm{AdaIN}(x; f_i, b)\,.\end{aligned}$$

Note that this *depthwise* AdaConv variant subsumes AdaIN, which is a special case with a 1×1 filter $\mathbf{f}$ and $\mathcal{N}(x) = \{x\}$.

Our second and final step towards our full AdaConv modulation extends this depthwise variant by expanding the input style parameters to also include a separable, pointwise convolution tensor $\mathbf{p} \in \mathbb{R}^C$, for an input with C feature channels. This enables AdaConv to perform modulation based on a style that captures not only global statistics and spatial structure but also correlations across features x_c in different input channels c,

$$\mathrm{AdaConv}(x; \mathbf{p}, \mathbf{f}, \mathbf{b}) = \sum_c p_c \, \mathrm{AdaConv}_{\mathrm{dw}}(x_c; \mathbf{f}_c, b_c)\,. \tag{3}$$

The input style $\{\mathbf{p}, \mathbf{f}, b\}$ for AdaConv effectively includes a *depthwise-separable 3D kernel* [12], with depthwise and pointwise convolution components, and per-channel biases. The actual number of depthwise and pointwise convolutional kernels used to modulate an input is a design choice and can be arbitrarily large. As we later describe in Sec. 3.2.2, this can be controlled using the number of groups n_g in a *depthwise-separable* convolutional layer.

In the following, we propose a kernel prediction framework for AdaConv and show how it can be used as a general replacement for AdaIN to achieve more comprehensive style-based conditioning in style transfer and also in other high-quality generative models.

3.2. Style Transfer with AdaConv

For style transfer, we begin with the original architecture of Huang *et al.* [13] and also apply the same content and style losses during training. However, instead of directly mapping the global style statistics onto those of the content features using AdaIN, we use our new kernel predictors with AdaConv to more comprehensively transfer different properties of the style. An overview of our style transfer architecture is given in Fig. 2.

The input style and content images are encoded using a pre-trained VGG-19 [32] encoder to obtain latent features of style $\mathcal{S}$ and content $\mathcal{C}$. For kernel prediction, the style features $\mathcal{S}$ are encoded further by a style encoder E_S to obtain a global style descriptor $\mathcal{W}$. From $\mathcal{W}$, our *kernel prediction networks* $K = \{K_1, K_2, .., K_N\}$ output *depthwise-separable convolutional kernels* [12] with per-channel biases. These predictions are ingested into all layers of the decoder D, which outputs the style-transferred result.

Our style transfer architecture employs 4 kernel predictors that operate at 4 different resolutions of the decoded image, with kernels of different dimensions. Each decoding layer has an adaptive convolution block (Fig. 3), in which the predicted depthwise and pointwise convolutions precede a standard convolution. These standard convolutional layers are responsible for learning style-independent kernels that are useful to reconstruct natural images and remain fixed at test time. The encoder E_S, the kernel predictors K and the decoder D are trained jointly to minimize the same weighted sum of content and style losses in [13], within the VGG-19 latent feature space.

3.2.1 Style Encoder

We now turn to the goal of predicting convolutional kernels from the style features $\mathcal{S}$, to be applied to the content features $\mathcal{C}$ at every scale of our image decoder. Here, an intermediary step is to compute a style representation $\mathcal{W}$ that comprehensively describes the style image at different scales, while being guided by the style transfer loss. This design choice is also motivated via analogy with state-of-the-art generative modeling [18, 19], where the term 'style' denotes both global and local properties of an image.

The pre-trained VGG-19 network translates the original input style image with (channels, height, width) dimensions equal to (3, 256, 256) into a style tensor $\mathcal{S}$ with dimensions

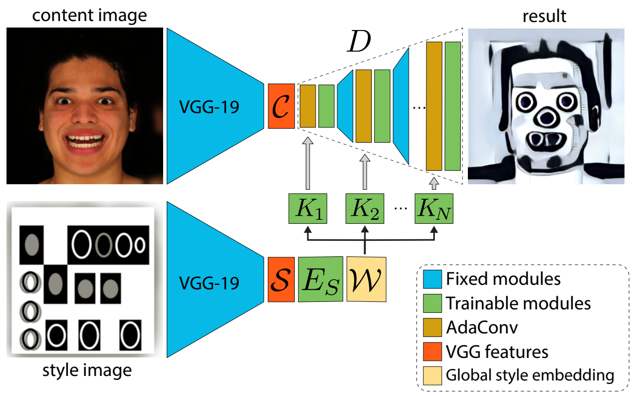

Figure 2: Network architecture with our new kernel predictors and AdaConv for structure-aware style transfer.

(512, 32, 32) at the VGG-19 $relu4\_1$ layer. Here, the receptive field does not cover the whole style image. Hence, we reduce $\mathcal{S}$ into our global embedding $\mathcal{W}$ by training an additional encoder component, E_S, as shown in Fig. 3.

Our style encoder E_S includes 3 initial blocks, each one with a 3 x 3 convolution, an average pooling operation, and a leaky ReLU activation. The output is then reshaped and fed to a final, fully-connected layer that provides the global style descriptor, which is in turn reshaped back into an output tensor $\mathcal{W}$ of size (s_d, s_h, s_w). The dimensions of this embedding are hyper-parameters and defined as a factor of the size of the kernel to be predicted.

Due to the use of this fully-connected layer, our network is limited to work with input style images of fixed dimensions (3, 256, 256). However, the dimensions of the content image are not restricted, since it flows through a fully convolutional component of the network.

3.2.2 Predicting Depthwise-Separable Convolutions

Each one of our kernel predictors K in Fig. 2 is a simple convolutional network whose input is the style descriptor $\mathcal{W}$, while the output is a depthwise-separable kernel. The choice of predicting depthwise-separable kernels [12] is motivated by the desire to keep the kernel predictor simple and computationally efficient, while also making the subsequent convolution faster.

A standard convolutional layer takes an input feature tensor of dimensions $(1, c_{in},\ h,\ w)$ and convolves it with a kernel tensor of size $(c_{out},\ c_{in},\ k_h,\ k_w)$, where c_{in} and c_{out} are the number of input and output channels. A per-channel bias is also added to the output. Thus, the number of weights required by such layer is $c_{out} \times c_{in} \times k_h \times k_w + c_{out}$. Depthwise-separable convolution reduces this number by collecting the input channels into n_g independent groups and by applying separate spatial and pointwise kernels that learn structural and cross-

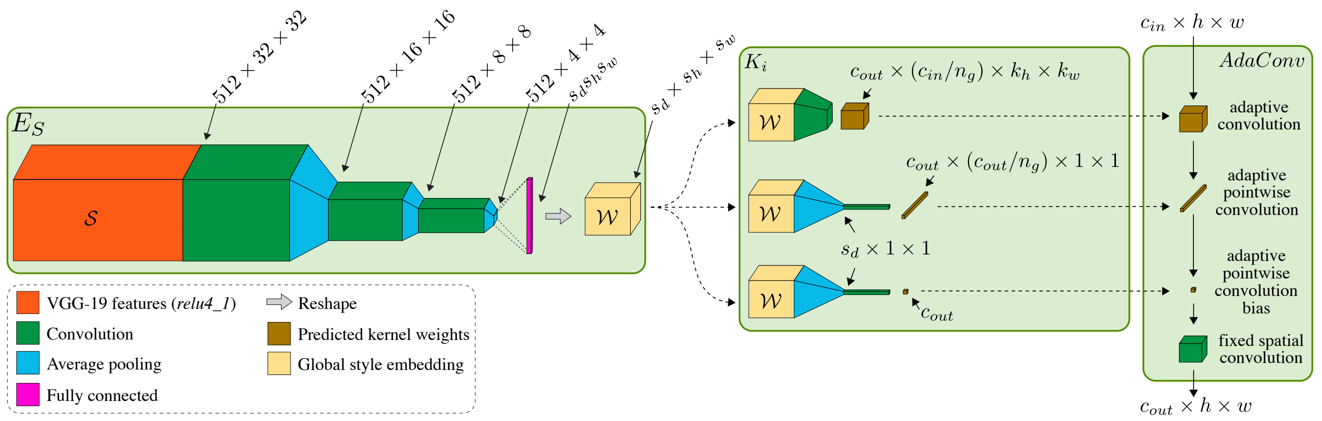
E_S
$512 \times 32 \times 32$
$512 \times 16 \times 16$
$512 \times 8 \times 8$
$512 \times 4 \times 4$
$s_d s_h s_w$
$s_d \times s_h \times s_w$
$\mathcal{S}$
$\mathcal{W}$
K_i
$c_{out} \times (c_{in}/n_g) \times k_h \times k_w$
$c_{out} \times (c_{out}/n_g) \times 1 \times 1$
$s_d \times 1 \times 1$
c_{out}
$c_{in} \times h \times w$
AdaConv
adaptive convolution
adaptive pointwise convolution
adaptive pointwise convolution bias
fixed spatial convolution
$c_{out} \times h \times w$
🟧 VGG-19 features (*relu4_1*)
🟩 Convolution
🟦 Average pooling
🟪 Fully connected
⇨ Reshape
🟫 Predicted kernel weights
🟨 Global style embedding

Figure 3: Architectures of the global style encoder E_S, kernel predictor K_i, and structural modulation in an AdaConv block with the resulting *depthwise-separable* convolutional kernel applied on input content features (top right).

channel correlations respectively. The required number of weights is reduced to $c_{out} \times \frac{c_{in}}{n_g} \times k_h \times k_w + c_{out}$. For a depthwise convolutional layer with $n_g = c_{in}$, each channel of the input is convolved with its own set of c_{out}/c_{in} filters. This is followed by a pointwise convolution with a 1×1 kernel to expands the number of channels in the output, adding a per-channel bias to the final output.

Here, it is important to note that the four AdaConv layers in our decoder have c_{in} equal to 512, 256, 128, and 64, decreasing as the spatial resolution increases. Thus, the kernel predictor at the lowest spatial resolution would usually have the highest number of parameters. To uniformly distribute our network's capacity over the successive resolution layers, we set a larger $n_g \in \{c_{in}, \frac{c_{in}}{2}, \frac{c_{in}}{4}, \frac{c_{in}}{8}\}$ at lower resolutions and gradually decrease it over successive layers, leading to better results (a comparison with constant $n_g = c_{in}$ is given in the supplement). n_g is set identically for both the depthwise and pointwise convolutional kernels.

Thus, each kernel predictor K outputs the necessary weights for the depthwise convolutional AdaConv layer in that scale of the decoder. These weights include: (i) spatial kernels of size $(c_{out}, \frac{c_{in}}{n_g}, k_h, k_w)$, (ii) pointwise kernels of size $(c_{out}, \frac{c_{out}}{n_g}, 1, 1)$, and (iii) a bias term $\mathbf{b} \in \mathbb{R}^{c_{out}}$.

The input to each kernel predictor K is the global style descriptor $\mathcal{W}$ of size (s_d, s_h, s_w), which is fed through convolutional and pooling layers that output spatial kernels of the target dimension, Fig. 3. These layers may consist of standard or transposed convolutions, whose parameters are determined at design time and depend on the size of kernels to be predicted. To predict pointwise 1×1 kernels, we pool $\mathcal{W}$ to a size $(s_d, 1, 1)$ and then perform a 1D convolution to predict c_{out} pointwise kernels. We use a separate predictor for per-channel biases, similar to that for pointwise kernels. Once the kernels and biases are predicted, they are used to modulate an input as shown in the right half of Fig. 3.

3.3. Training

To compare against existing techniques in style transfer (see Fig. 4), we train our method using the COCO dataset [23] as content images and the WikiArt dataset [25] as style images. For the remaining comparisons to AdaIN, we used a custom content dataset of around 4000 human faces captured in a controlled studio setting as content images and continued to use the WikiArt dataset as style images. For the experiments where we use faces as content, we re-train both AdaIN and AdaConv from scratch for a fair comparison. To train our method, we use the Adam optimizer with a learning rate of 1e-4 and a batch size of 8. For AdaIN we use the same settings as in [13]. Additional details on our training are presented in the supplementary material.

4. Results

We now show results of using AdaConv as an extension of AdaIN for style transfer and generative modeling.

4.1. Style Transfer

Our work is primarily motivated by the application of image style transfer, much like the original AdaIN [13]. In this section, all our results are created with a style descriptor size $s_d = 512$ and a kernel size of 3×3.

Qualitative Comparisons. We first compare AdaConv with several style transfer methods, including Huang and Belongie's AdaIN [13], Chen and Schmidt [6], Ulyanov *et al.* [34], Gatys *et al.* [8], Jing *et al.* [15], Li *et al.* [22], Sheng *et al.* [31], and Johnson *et al.* [17]. Fig. 4 shows that our approach performs comparably with the current state of the art, and is notably strong in preserving the structure of the style image. For instance, the structure of the water in the sailboat image (first row) resembles the hair strands in the style image; the structure of the brush strokes in the artistic paintings are transferred naturally to the content images.

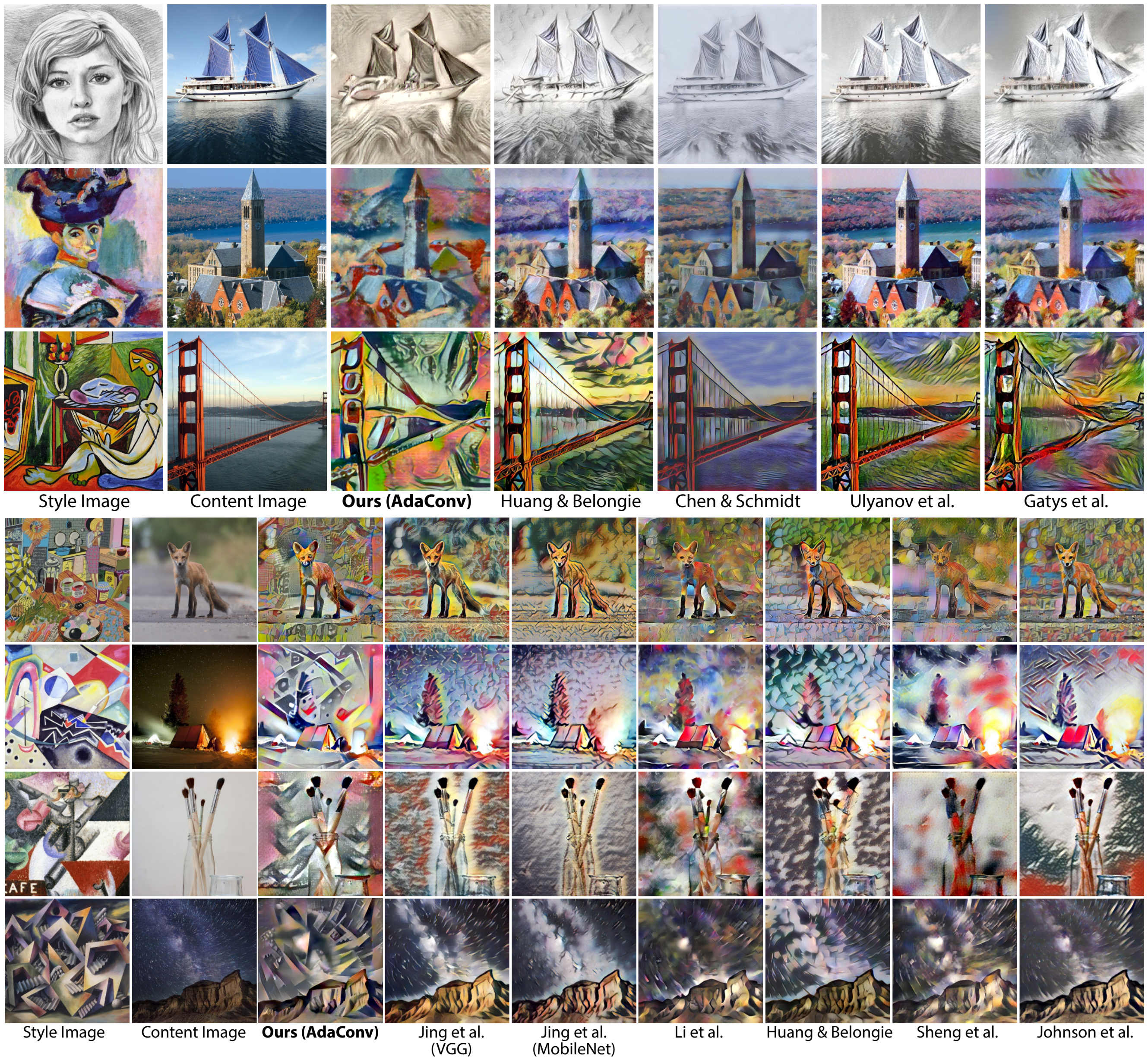

Figure 4: AdaConv performs comparably with current state-of-the-art methods[1,2]. Our method is particularly good at transferring the local structure of the style image to the content image.

As AdaConv extends AdaIN, we perform a more thorough comparison in Fig. 5. In all cases, AdaConv renders a content image that is more faithful to the structure (local spatial distribution) of the style image, while also transferring the global statistics of the style. AdaIN cannot transfer the style structure, only the global statistics of the style.

[1]Portions of this figure are © 2017 IEEE. Reprinted, with permission, from Huang and Belongie [13].

[2]Portions of this figure from Jing et al. [15] are © 2020, Association for the Advancement of Artificial Intelligence. All rights reserved. Permission to reuse the figure for other purposes (or granting it to others) is NOT allowed.

Style Rotations. We further highlight the benefit of AdaConv in preserving style image structure by applying the same style image under various degrees of rotation. Arguably, a rotated style image is in fact a different style. However, this notion is largely lost when transferring the style using AdaIN, since global feature statistics remain mostly the same under rotation. We illustrate this effect in Fig. 6, where we transfer a style image under four different rotations to the same content image (taken from the last row of Fig. 5). As we can see, AdaConv successfully

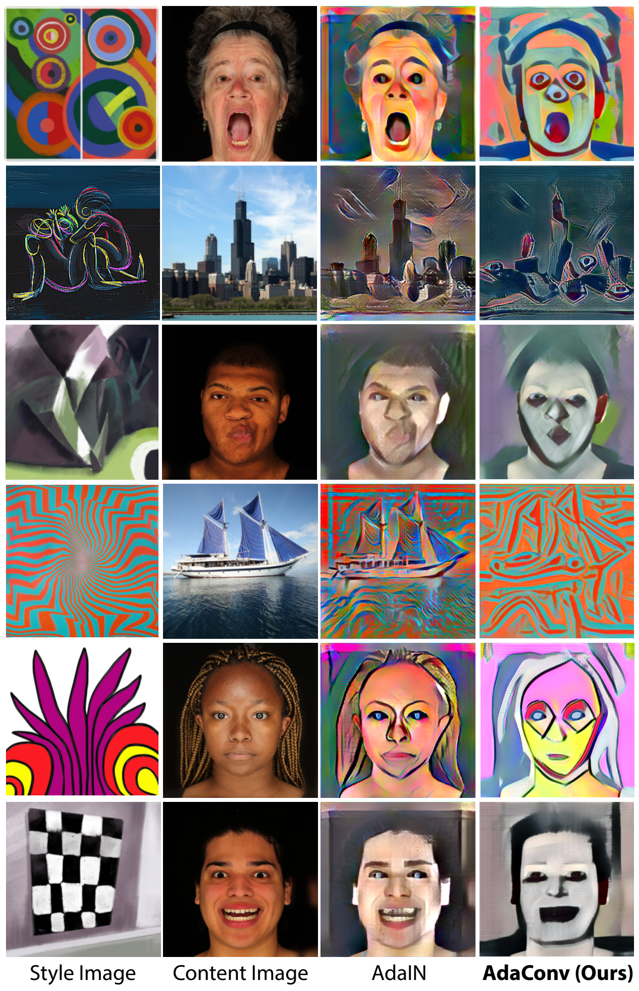

Figure 5: Compared to AdaIN [13], our AdaConv extension is better at maintaining the structure of the style image thanks to our kernel prediction approach.

preserves the spatial orientation of the style image in the transferred result, whereas the AdaIN results look mostly the same independent of rotation. We encourage the reader to view more rotation results in the supplemental video.

Style Interpolation. As with AdaIN, we can also interpolate in style space to generate results that mix multiple input styles. In the case of AdaConv, we interpolate the output of the style feature encoder, before the kernel predictor. The interpolated style descriptor then produces kernels that alter the structure of the decoded result. As a result, structural elements of the style images are smoothly interpolated spatially. This can be observed in Fig. 7, where we interpolate between two style images that have very different structure, and apply the result to a content image of a face. As compared to AdaIN, AdaConv generates in-between results that have structure which is also *in-between* the structure of the two style images. For example, one can easily see structural

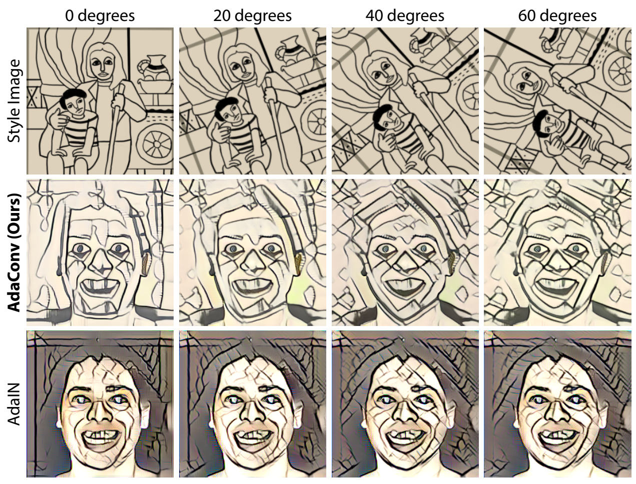

Figure 6: When rotating the style image, the style orientation is transferred well to the content image using AdaConv, while AdaIN results are mostly rotation-invariant, since the global statistics do not vary much under rotation.

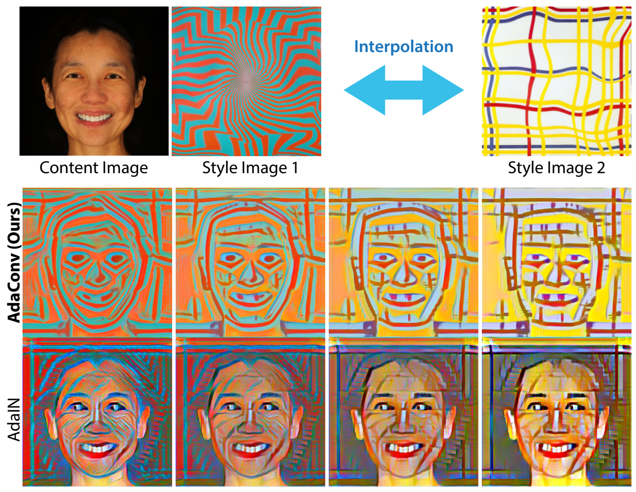

Figure 7: As we interpolate between two style images, the AdaConv results are smoother than AdaIN, and we can follow individual structures as they deform spatially from one result to the other with the AdaConv method.

elements like thick lines actually deforming and warping from one result to the other using AdaConv.

User Study. We also ran a user study to compare results of AdaConv and AdaIN. Participants evaluated a total of 10 side-by-side style transfer results obtained by AdaIN and AdaConv, with the two results displayed in a randomized order. The participants were asked to choose a result based on the following 3 questions: *(1) Which style transfer result preserves the content image better? (2) Which style transfer result preserves the style structure from the style image better? (3) Which style transfer result is overall doing a better job at transferring the* *style to the content image?* A total of 185 participants from multiple countries, age groups and backgrounds took part in our online survey. As expected, **93.9%** of the participants felt that AdaIN is better at content preservation, while **92%** felt that AdaConv was capturing the style structure better. Overall, a strong majority of the participants, **71.8%**, said that AdaConv did a better style transfer job.

Video Style Transfer. Finally, as seen on the supplemental video, AdaConv performs style transfer on video sequences with good temporal stability even when naïvely applying transfer to each frame independently. Improved temporal stability could be achieved by extending AdaConv with an optical flow technique for video style transfer [2].

4.2. Extensions to Generative Models

Though originally proposed for style transfer, AdaIN has found its way into a multitude of applications including generative models like StyleGAN [18] and StyleGAN2 [19], where it has been used to inject 'style' into a generator network that is trained in an adversarial fashion. As AdaConv is an extension of AdaIN, we demonstrate its suitability for generative networks by incorporating it, together with our kernel predictors, into a StyleGAN2-like network.

At each scale of the StyleGAN2 generator, the per-channel mean and standard deviation (A) predicted by an MLP are used to modulate the convolutional layer's weights using AdaIN (Fig. 8, left). Note, however, that the kernel weights are learned during training and only their scaling is adapted at test time. In contrast, our AdaConv blocks predict full depthwise-convolutional kernels from the input style parameters at test time. Thus, we replace each weight demodulation block in StyleGAN2 with an AdaConv block that performs 'style-based' depthwise-separable convolution on the up-sampled input from the previous layer (Fig. 8, right). A noise vector is also transformed through an MLP into an input 'style' $\mathcal{W}$ for the kernel predictor in each AdaConv block. Since depthwise convolutions have fewer parameters than standard convolutions, we follow the adaptive convolution with a standard 2D convolution in the same block. Then per-channel biases and Gaussian noise are added and the output is fed into the next AdaConv block.

We trained this modified StyleGAN2 generator on the FFHQ, CelebHQ, AFHQ-wild and AFHQ-dog datasets at a resolutions of (256×256). Our modified generator and the discriminator from StyleGAN2 are trained with the same hyperparameters and loss functions as [19]. We trained our generative network on a single Nvidia2080Ti GPU for 300K iterations ($\sim$1.2m real images) at a batch size of 4. We show some examples of synthetic faces and wild animals in Fig. 9. These results were generated with a style descriptor size $s_d = 128$ and a kernel size of 3×3. Additional results of using AdaConv in a generative setting are provided in our supplementary material.

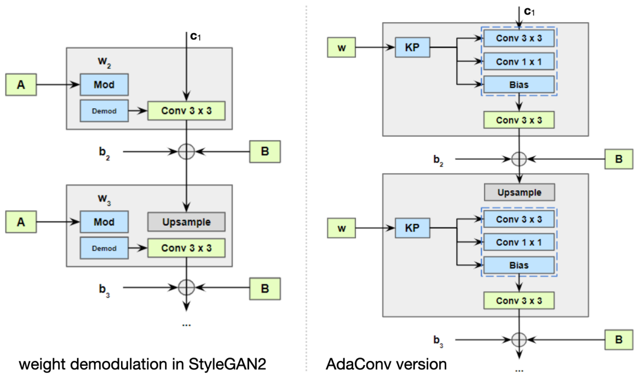

Figure 8: Demodulation blocks in StyleGAN2 [19] and our alternative network with AdaConv blocks.

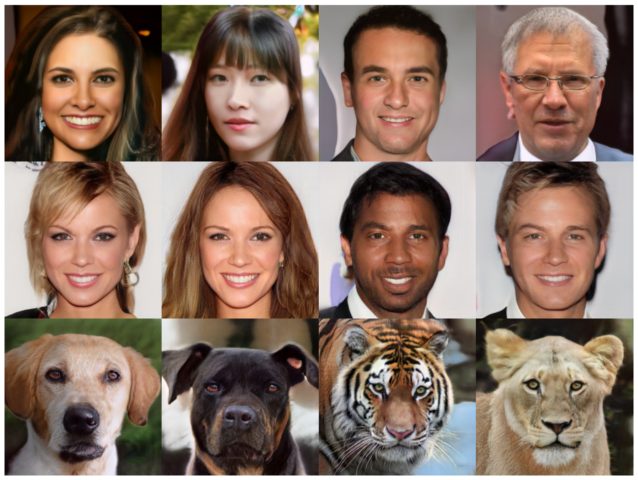

Figure 9: AdaConv can also be applied to generative architectures like StyleGAN2 [19] for realistic image synthesis.

5. Conclusion

In this work we propose Adaptive Convolutions (AdaConv) for structure-aware style manipulation. As an extension to adaptive instance normalization (AdaIN), AdaConv predicts convolution kernels and biases from a given style embedding, which can be woven into the layers of an image decoder to better adjust its behavior at test time. In the context of neural style transfer, AdaConv can transfer not only global statistics but also the spatial structure of a style image onto a content image. In addition, AdaConv is also applicable in style-based image generation (e.g. StyleGAN), as we have demonstrated, and virtually everywhere AdaIN has been employed. It provides a new, generic building block for ingesting conditioning input data into CNN-based image generation and style manipulation.

Acknowledgements. We would like to thank Maurizio Nitti for creating our style images.

References

[1] Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. volume 36, New York, NY, USA, July 2017. Association for Computing Machinery. 2, 3

[2] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1114–1123, 2017. 8

[3] Dongdong Chen, L. Yuan, Jing Liao, Nenghai Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2770–2779, 2017. 3

[4] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stereoscopic neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[5] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and G. Hua. Explicit filterbank learning for neural image style transfer and image processing. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 3

[6] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016. 2, 5

[7] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. 2017. 2

[8] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 5

[9] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[10] Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. Sample-based monte carlo denoising using a kernel-splatting network. volume 38, pages 125:1–125:12. ACM, 2019. 3

[11] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. 2017. 2

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. volume abs/1704.04861, 2017. 3, 4

[13] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2, 4, 5, 6, 7

[14] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in neural information processing systems*, pages 667–675, 2016. 3

[15] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In *AAAI*, 2020. 2, 3, 5, 6

[16] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. 2019. 2

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. 2, 5

[18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2, 4, 8

[19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 4, 8

[20] Dmytro Kotovenko, Artsiom Sanakoyeu, Sabine Lang, and Bjorn Ommer. Content and style disentanglement for artistic style transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[21] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[22] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 385–395, Red Hook, NY, USA, 2017. Curran Associates Inc. 2, 5

[23] 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 David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 5

[24] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2502–2510, 2018. 3

[25] K. Nichol. *Painter by numbers, Wikiart*, 2016. https://www.kaggle.com/c/painter-by-numbers. 5

[26] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2270–2279, 2017. 2, 3

[27] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 3

[28] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting adaptive convolutions for video frame interpolation. *arXiv preprint arXiv:2011.01280*, 2020. 2, 3

[29] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[30] Falong Shen, Shuicheng Yan, and Gang Zeng. Neural style transfer via meta networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8061–8069, 2018. 2

[31] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8242–8250, 2018. 2, 5

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 4

[33] Dmitry Ulyanov, Vadim Lebedev, Andrea, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. volume 48 of *Proceedings of Machine Learning Research*, pages 1349–1357, New York, New York, USA, 20–22 Jun 2016. PMLR. 2

[34] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4105–4113. IEEE Computer Society, 2017. 2, 5

[35] Thijs Vogels, Fabrice Rousselle, Brian Mcwilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with kernel prediction and asymmetric loss functions. volume 37, New York, NY, USA, July 2018. Association for Computing Machinery. 2, 3

[36] Zhihao Xia, Federico Perazzi, Michael Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. Basis prediction networks for effective burst denoising with large kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[37] Tianfan Xue, Jiajun Wu, Katherine L Bouman, and William T Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances In Neural Information Processing Systems*, 2016. 3

[38] Julio Zamora Esquivel, Adan Cruz Vargas, Paulo Lopez Meyer, and Omesh Tickoo. Adaptive convolutional kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019. 3

[39] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2