Is Bigger Always Better? An Empirical Study on Efficient Architectures for Style Transfer and Beyond

Jie An
University of Rochester
Rochester, NY, USA
jan6@cs.rochester.edu

Tao Li
Peking University
Beijing, China
li_tao@pku.edu.cn

Haozhi Huang
Xverse Inc.
Shenzhen, Guangdong, China
huanghz08@gmail.com

Jinwen Ma
Peking University
Beijing, China
jwma@math.pku.edu.cn

Jiebo Luo
University of Rochester
Rochester, NY, USA
jluo@cs.rochester.edu

Abstract

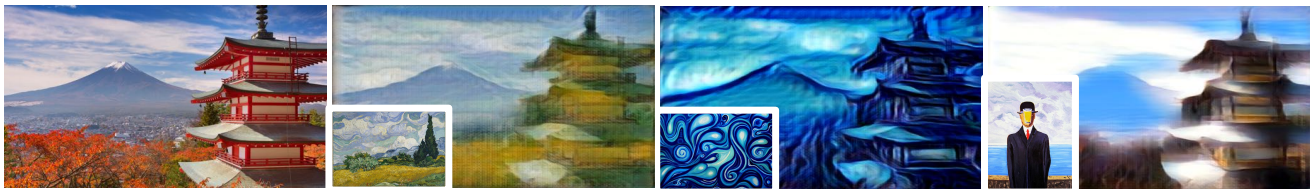
Network architecture plays a pivotal role in style transfer. Most existing algorithms use VGG19 as the feature extractor, which incurs a high computational cost. In this work, we conduct an empirical study on the popular network architectures and find that some more efficient networks can replace VGG19 while having comparable style transfer performance. Beyond that, we show that an efficient network can be further accelerated by removing its empty channels via a simple channel pruning method tweaked for style transfer. To prevent the potential performance drop due to using a more lightweight network and obtain better style transfer results, we introduce a more accurate deep feature alignment strategy to improve existing style transfer modules. Taking GoogLeNet as an exemplary efficient network, the pruned GoogLeNet with the improved style transfer module is $2.3 \sim 107.4\times$ *faster than the state-of-the-art approaches and can achieve 68.03 FPS on* 512×512 *images. Extensive experiments demonstrate that VGG19 can be replaced by a more lightweight network with significantly improved efficiency and comparable style transfer quality.*

1. Introduction

Neural style transfer is an image editing task that aims at changing the artistic style of an image according to a reference image. Given a pair of content and style images as the input, a style transfer method will generate an image with the scene of the content image and the visual effects (*e.g.*, colors, textures, strokes, etc.) of the style image. For example, in Fig. 1, we transfer a picture of "Chureito pagoda" into different styles. The difference between style transfer and GAN-based image translation is that style transfer model usually can be applied to any images but cannot transfer structure of the image. On the contrary, GAN-based image translation can only be used to generate images from certain domains where the training data comes from. The merit of GAN-based image translation is that it can change both structure and appearance of images.

Remarkable advances have been made in neural style transfer. The pioneering work is presented in [15, 18], where Gatys *et al*. make the first attempt to connect style representation to the Gram matrices of deep features. Following this line of research, many algorithms based on iterative optimization [16, 57, 55, 52, 40, 45, 27, 34, 41] and feed-forward neural networks [14, 65, 33, 64, 55, 66, 13, 4, 77, 70, 19, 78] have been proposed. While these algorithms can produce high-quality stylization results, they have to trade-off between the efficiency and the generalization ability. Recently, universal style transfer methods [7, 43, 28, 58, 44, 20, 74, 2, 32, 42, 71, 54, 5, 6, 8, 11, 25, 51, 46, 68] have been proposed to handle arbitrary styles and contents. To generate high-quality images while retaining the benefit of universal transfer, a few improved approaches such as multi-level stylization [43, 44, 42, 54], iterative EM process [20], wavelet transform [74], Normalization Flow [1], attention mechanism [51], and contrastive loss [5] are introduced, which make significant advances in style transfer.

The network architecture used to extract features plays a pivotal role in a style transfer algorithm [68]. A long-standing convention in style transfer is that most state-of-the-art algorithm uses VGG19 [60] as the backbone. For example, iterative methods [18, 16, 55, 52, 40, 45] use VGG19 as the feature extractor and compute loss terms accordingly. On the other hand, approaches based on feed-forward neural networks [43, 44, 20, 74] adopt VGG19 as the encoder



Content: Chureito Pagoda | Whaet Field with Cypresses | Blue Swirls | Beer

Figure 1: Style transfer results by using an efficient architecture on arbitrary content and style images.

part of auto-encoders. In terms of the style transfer quality, although VGG19 can indeed produce good style transfer images as demonstrated by the above-mentioned algorithms, Wang *et al*. [68] shows that ResNet-50 [22] with a softmax transformation trick is comparable to or even better than VGG19, which breaks the myth that VGG19 is always the best. When it comes to efficiency, VGG19 has 12.95 MB parameters and 189.50 GFLOPs, which imposes a big computational burden on style transfer algorithms, especially when we work on real-time style transfer applications. Can we find a network architecture that has a comparable style transfer quality with VGG19 but is more efficient? This study addresses this issue.

To find efficient and comparably high performance network architectures for style transfer, we first conduct an empirical study on the stylization effect and efficiency of 28 most popular network architectures [60, 36, 30, 26, 62, 61, 53, 56, 73, 76]. We observe that although the style transfer algorithms using VGG19 [60] as the backbone indeed produce high-quality stylization results, using more lightweight backbones such as GoogLeNet [61] and MobileNetv2 [56] can achieve similar stylization effects more efficiently.

Next, we reveal that the above-mentioned classical architectures can be further accelerated at little cost in style transfer. Taking GoogLeNet as an example, we find that deep features by the ReLU layers in the networks pre-trained on the ImageNet dataset [37] contain a few empty channels, which have no contributions to style transfer but still consume computation and memory resources. Inspired by [39], we remove those empty channels and the corresponding filters in Convolution and Batch Normalization operators ahead of the ReLU layer by a simple yet effective channel pruning method. The pruned GoogLeNet achieves $2\times$ acceleration at the cost of tiny performance degradation.

To prevent performance drop of using more efficient networks, we adopt a simple yet effective method, named *sandwich swap transform* ($\texttt{S}^2$), based on the style transfer module used by Avatar-Net [58]. $\texttt{S}^2$ is lightweight and can perform more accurate feature alignment between the content and style features compared with Avatar-Net [58], which is crucial for efficient networks since fewer feature maps can be used to perform the feature alignment. Experimental results show that $\texttt{S}^2$ has improved content preservation ability than the state-of-the-art methods.

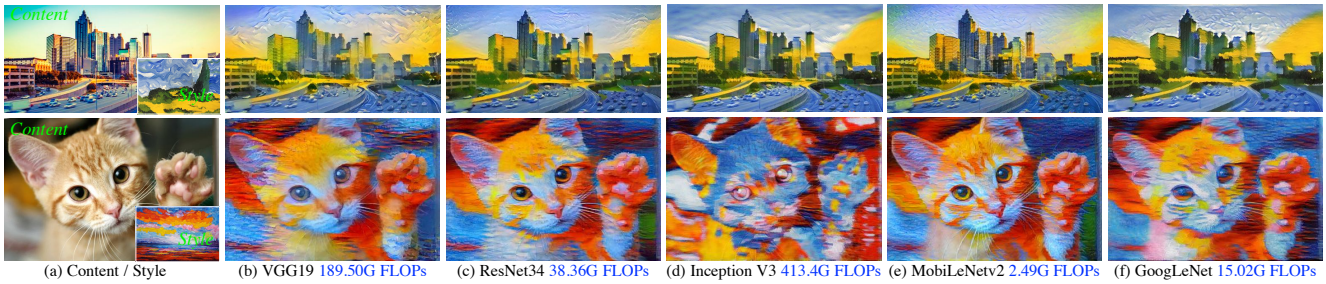
Our main contributions can be summarized as follows:

- We conduct an empirical study on the style transfer performance of different network architectures, revealing that we can use more efficient networks (*e.g*., GoogLeNet) to replace VGG19 for better efficiency.
- We remove the network's redundant parameters via a simple channel pruning method and therefore further improve the efficiency of style transfer while keeping the stylization quality almost intact.
- We propose a sandwich swap transform ($\texttt{S}^2$) module based on Avatar-Net [58], which is more suitable for efficient networks and thus improves the stylization quality of efficient networks.

2. Related Work

Universal Style Transfer. Universal style transfer algorithms can be categorized into two types: optimization-based methods and feed-forward neural network based methods. The algorithm by Gatys *et al*. [17] belongs to the first category. This kind of method performs optimization only on the encoder and does not have a decoder. We use the method of Gatys *et al*. [17] in the empirical study because it avoids the influence of the decoder architecture and decoder training, therefore can better reflect the style transfer ability of different feature extraction backbones. In contrast to the optimization-based methods, feed-forward neural network based algorithms [7, 28, 43, 44, 20, 58, 74, 67, 10, 72] usually consist of two parts: an auto-encoder and a style transfer module that works at the bottleneck. The auto-encoder architecture used by all these methods is the same, *i.e*., the VGG19 network [60] pre-trained on the ImageNet dataset [37] as the encoder and its inverted version as the decoder. In this work, we concentrate on finding and improving efficient network architectures to replace VGG19 while maintaining high-quality stylization effects.

Image-to-Image Translation. In addition to style transfer, image-to-image translation [31, 69, 50, 63, 59, 48, 79, 29, 49] can also be used to transfer image styles. To render a certain visual style, image-to-image translation methods usually need pre-transfer and post-transfer datasets to train generator and discriminator networks. However, universal style transfer methods can be used to make style transfer for arbitrary content and style images in a zero-shot fashion.



(a) Content / Style (b) VGG19 189.50G FLOPs (c) ResNet34 38.36G FLOPs (d) Inception V3 413.4G FLOPs (e) MobiLeNetv2 2.49G FLOPs (f) GoogLeNet 15.02G FLOPs

Figure 2: Style transfer results by efficient network architectures that are comparable with VGG19. Please zoom in on screen to see fine details. We take GoogLeNet as an exemplary network in the remaining parts since it has the a better quality-efficiency tradeoff.

Table 1: Statistics of user study on different networks. No one is significantly better than others.

Network	GoogLeNet	Inception	MobileNetv2	ResNet34	VGG19
# 1st Rank	403	413	408	429	419
Mean Rank	2.8786	2.9500	2.9143	3.0643	2.9929

Neural Network Pruning. The network pruning approaches can be categorized into two types: weight pruning and channel pruning. Weight pruning [21, 38, 3, 12] usually detect non-operative weight positions in filters and disable them by setting to zero. Channel pruning approaches [39, 23, 75, 24] delete information-wise redundant channels and their corresponding weights entirely. The channel pruning method we use is a simple variant of the method by Li *et al*. [39]. Since style transfer methods generally use feature maps produced by ReLU layers in networks, our method removes empty channels of every ReLU layer and hereby indirectly deletes the weights of the preceding convolution/batch normalization operators. This differs from [39] which focuses on pruning channels of convolutional layers directly. More importantly, since we are removing nonfunctional empty channels for style transfer, a fine-tuning process is not needed.

3. Is VGG19 Always the Best?

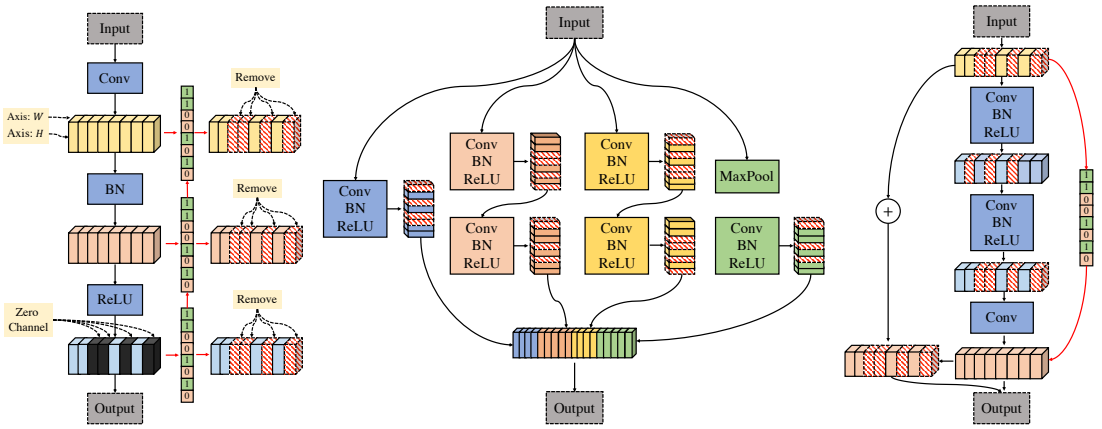
Existing state-of-the-art style transfer algorithms [43, 44, 20, 74] usually have a similar framework which consists of 1) an auto-encoder to extract (encoder) and invert (decoder) features, 2) a feature transfer module that works at the bottleneck of the auto-encoder. VGG19 [60] is commonly used as the feature extractor. As claimed by Wang *et al*. [68], the architecture of the feature extractor plays a central role in style transfer algorithms, which is even more important than network training. In terms of the style transfer quality, deep features extracted by the auto-encoder directly influence the quality of style transfer. VGG19 indeed can produce high-quality style transfer results. However, as demonstrated by [68], VGG19 is not always the best. For example, ResNet-50 [22] trained with the softmax transformation trick is comparable to or even better than VGG19 in terms of the style transfer quality. Concerning efficiency, VGG19 usually occupies a large portion of the overall time cost for a style transfer algorithm. For example, VGG19 occupies 72.73% time consumption of the inference time of AdaIN. Therefore, VGG19 is not perfect in terms of efficiency, and a more lightweight feature extractor is desired for faster style transfer.

4. Finding Efficient Architectures

To find an efficient network architecture with a strong style transfer ability, we perform an empirical study on 28 popular network architectures [60, 36, 30, 26, 62, 61, 53, 56, 73, 76]. We compare the style transfer performance and time consumption of different architectures based on a dataset that consists of 1092 content-style pairs (42 content; 26 styles). We first train each network on the ImageNet dataset [37], and then use these networks to generate stylized images based on the algorithm of Gatys *et al*. [17], respectively. Here we apply the algorithm proposed by Gatys *et al*. [17] since it does not need a decoder to invert features back to images, thus avoiding the bias introduced by the decoder training.

According to the visual comparison, VGG19/16/13/11, ResNet18/34, GoogLeNet, Inceptionv3, and MobileNetv2 outperform other architectures in terms of fine local texture generation. We show the visual comparison of the aforementioned superior architectures in Fig. 2 and the complete comparison results are shown in the supplementary material.

To reinforce our findings based on the visual comparison that MobileNetv2, GoogLeNet, Inceptionv3, and ResNet34 are comparable against VGG19 in terms of the style transfer results, we further make a user study to quantitatively assess the style transfer quality of the above-mentioned architectures. The user study is based on the dataset consisting of 1092 content-style pairs crawled from the Internet. In each question, we show five stylized images side-by-side in random order and ask the user to rank these images according to their style transfer quality. In the end, we collect 2072 rank results in total. Tab 1 shows the statistics of the user study. According to the result of the quantitative evaluation, the above five network architectures have a similar mean rank and the number of 1st ranks is also close to each other. Since GoogLeNet shows a good style transfer performance with



(a) Prune Conv-BN-ReLU (b) Prune Inception (c) Prune Inverted Residual Block

Figure 3: Removing empty channels of different modules.

$1/10$ FLOPs of VGG19, *w.l.o.g.*, we choose GoogLeNet as an exemplary efficient network to replace VGG19. Please note that the proposed analysis and method can be applied to any alternative efficient architectures.

5. Making Efficient Networks Faster

By employing GoogLeNet instead of VGG19 as the feature extractor, a style transfer algorithm (*e.g.*, Gatys [17] and AdaIN [28]) can achieve more than $3\times$ acceleration. However, further speed improvement is always desired for practical applications. So here comes a question: Can we further improve the efficiency of GoogLeNet? Motivated by this, we make an in-depth analysis of GoogLeNet by visualizing its feature maps per channel, we find that a few channels of ReLU layers (especially in shallower layers) are empty, *i.e.*, zero tensors. Fig. 4 shows the feature map of the ReLU_1_1 layer of GoogLeNet and the average response per channel. We can find that empty channels have no response for all random input images. Please refer to the supplementary material for quantitative empty channel analysis based on the MS_COCO dataset. According to quantitative analysis of a large number of images, we find that the positions of these empty channels remain unchanged given different input images, *i.e.*, they are data agnostic. During inference, empty channels deliver no information to the subsequent layers (please refer to the supplementary material for the mathematical proof). However, empty channels themselves, the corresponding convolution and BN operators in upper layers still waste GPU memories and computational resources. Since only the features of ReLU layers are used to make style transfer, we can accelerate the style transfer algorithm without damaging the transfer quality by pruning those empty channels and other corresponding parameters in upper convolutional and BN layers.

To remove empty channels in the features maps, the most intuitive way is to use a channel pruning method. However,

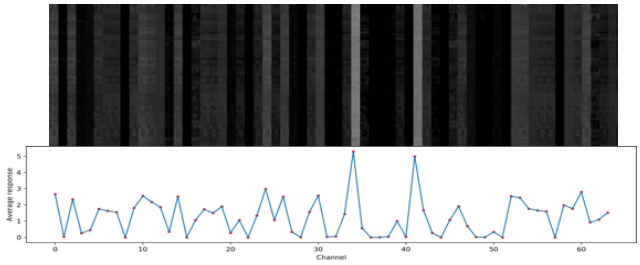
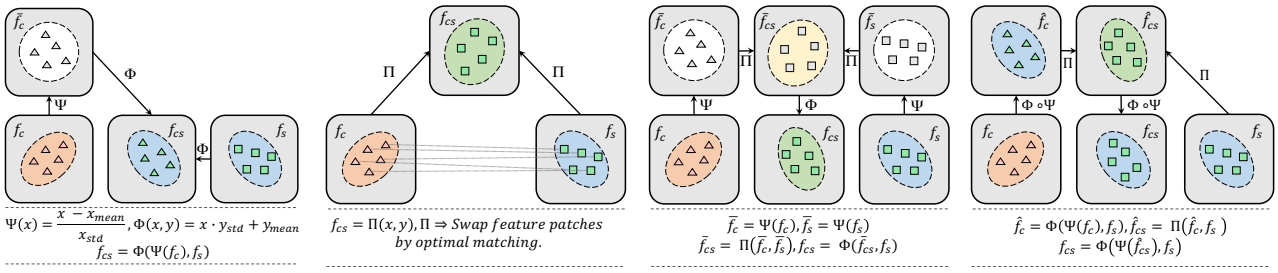


Figure 4: Visualization of the feature maps produced by ReLU_1_1 layer in GoogLeNet. We show the features produced by 16 input images vertically while features of each channel are shown side-by-side horizontally.

the widely-used filter pruning method proposed by Li *et al*. [39] cannot be used directly in style transfer because it focuses on pruning low-contribution filters in convolution layers and then the corresponding BN and ReLU layers while style transfer algorithms are usually based on feature maps of ReLU layers. Low-contribution filters in convolution layers usually cannot lead to empty channels in ReLU layers. To this end, we tweak the method by [39] to make it focus on pruning empty channels in ReLU layers and then removing the corresponding filters of Conv and BN. It is worth noting that since removing empty channels does not hurt the style transfer performance, there is no need to finetune the network after pruning while [39] does need.

The pruning method we used works on different architectures. Here we show the way to prune three typical network modules, *e.g.*, Conv-BN-ReLU layer, Inception, and Residual Block. Fig. 3(a) shows the pruning of the Conv-BN-ReLU module. We first store the indexes of every empty channel in a binary vector m. In m, every position is set to 1 or 0 to represent keeping/pruning the corresponding channel. For example, in Fig. 3(a), the value of m is set to "11001010". Then m is passed to the upper convolutional and BN layers before the ReLU layer. Based on m, all empty channels in features and the corresponding weights



(a) AdaIN (b) StyleSwap [7] (c) Style Decorator [58] (d) Ours S^2

Figure 5: Comparison of different feature transfer modules. The shape of dotted ellipses means the variances of features, and the color denotes the means of features. The triangles and squares represent feature patches. (a) AdaIN transfers holistic global appearance by adopting Ψ to normalize/whiten f_c and then using Φ to match/coloring $\bar{f}_{cs}$ with respect to f_s. The produced f_{cs} has the same mean μ and standard deviation σ as f_s, but cannot transfer complex textures directly. (b) StyleSwap creates f_{cs} by using the Π to get the optimal matching between f_c and f_s. It has the ability to create fine textures but cannot transfer holistic global appearance since it cannot match μ and σ between f_{cs} and f_s. (c) Avatar-Net also fails to match μ and σ, thus producing an impaired global appearance. Our S^2 can benefit from Π in rendering complicated textures and keep μ, σ of f_{cs} equal to f_s.

in Conv and BN operators are removed. To prune the Inception module, as Fig. 3(b) shows, we first prune every branch within the Inception module and then concatenate the pruned feature maps together. Fig. 3(c) shows the way we prune Inverted Residual Blocks within Mobilenetv2. To allow the residual connections from the input to the output, we pass m from the input feature to the output and prune other feature maps accordingly. By removing empty channels, we reduce the parameter size of GoogLeNet from 6.63 MB to 3.28 MB and Mobilenetv2 from 2.22 MB to 760.11 KB, respectively.

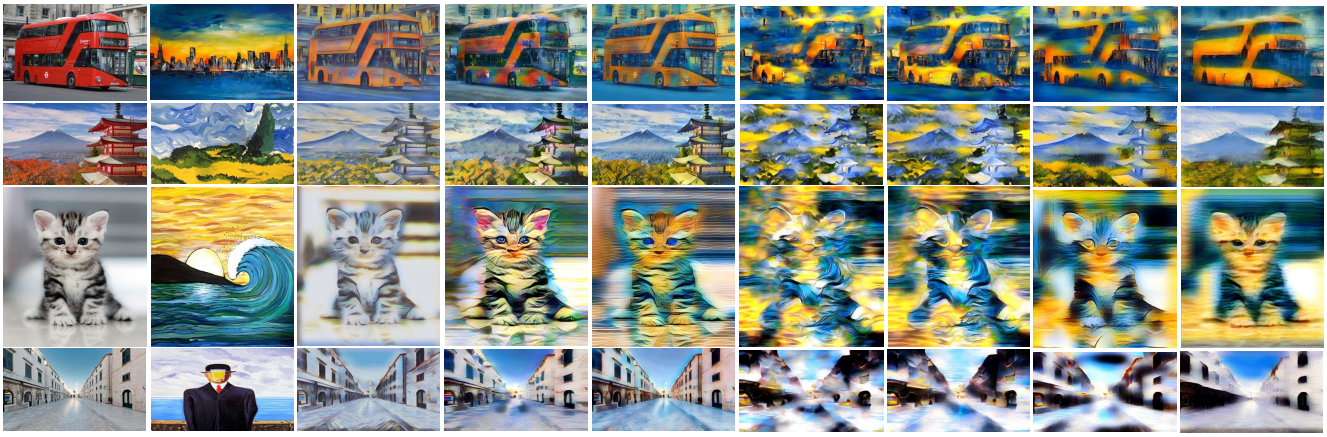
When we perform empty channel visualization and quantitative analysis, we find that VGG networks (VGG11/16/19) have fewer empty channels compared with other networks, *e.g.*, GoogLeNet and MobileNetv2. We conjecture the reason could be that VGG networks do not use residual connections, therefore, each channel has to learn more visual information than networks consisting of Inception and Residual Blocks. Furthermore, the number of empty channels of a network may be able to reflect its performance on style transfer. We leave this to future work.

6. S^2: Sandwich Swap Transform

So far we have improved the efficiency of style transfer by replacing VGG19 with a pruned GoogLeNet. To enable the efficient network to have comparable style transfer performance with state-of-the-art methods that use big backbones, we make a simple yet effective improvement on one of the most popular style transfer modules named StyleDecorator in Avatar-Net [58] so that it is more suitable for efficient networks. We name the improved style transfer module Sandwich Swap Transform (S^2) as it uses a three-layer structure similar to StyleDecorator [58]. As shown in Fig. 5(d), S^2 adopts *AdaIN-Swap-AdaIN* in a cascade to perform feature transfer. In S^2, we first adopt an AdaIN module to project f_c to the space of f_s, and then a swap module is used to directly copy textures/painting strokes from f_s to $\hat{f}_{cs}$. After that, another AdaIN module is introduced to correct color aberrations by aligning μ and σ between f_{cs} and f_s. S^2 works at the bottleneck of auto-encoder.

Our S^2 is motivated by AdaIN [28] and StyleSwap [7]. As Fig. 5(a) shows, AdaIN first normalizes the content feature f_c with Ψ and then re-colors it with respect to the style feature f_s by Φ. Specifically, AdaIN can maintain $\mu(f_{cs}) = \mu(f_s), \sigma(f_{cs}) = \sigma(f_s)$, where μ, σ represent the mean and standard deviation of the feature maps across H, W axes and have a shape of $C \times 1$. Note that the matching of μ and σ can reduce the Gram loss between f_{cs} and f_s, so AdaIN can transfer visual effects from the style to the content. However, AdaIN is good at transferring global appearance and usually degrades in rendering complicated textures. StyleSwap (Fig. 5(b)) directly borrows fine textures from f_s to f_{cs} according to the optimal matching between f_c and f_s. Since f_{cs} consists of a few selected patches of f_s, μ and σ of f_{cs} and f_s are different. Moreover, directly making swap on unnormalized f_c and f_s may bias the optimal matching [58]. Therefore, the results of StyleSwap contain great textures but usually have color aberrations. Our S^2 combines AdaIN and StyleSwap, which enjoys the merits and mitigates the drawbacks of both methods.

Sandwich Swap Transform (S^2) is based on the StyleDecorator in Avatar-Net, which can be regarded as a variant of StyleDecorator specifically for efficient networks. Our method makes a style swap in the space of f_s instead of the normalized space. In this way, the style swap procedure can achieve more accurate patch matching by taking both textures and colors to perform optimal matching while colors are not considered in StyleDecorator. For big models, we usually have enough number of feature maps in feature



(a) Content (b) Style (c) StyleSwap (d) AdaIN (e) LinearWCT (f) WCT (g) OptimalWCT (h) Avatar-Net **(i) Ours**

Figure 6: Style transfer result comparison against the state-of-the-art universal style transfer algorithms. All compared images are generated by the officially released codes of the corresponding methods.

(a) Input (b) GoogLeNet (c) GoogLeNet w/ pruning (d) Input (e) MobiLeNetv2 (f) MobiLeNetv2 w/ pruning

Figure 7: Comparison between the style transfer results with and without pruning. Top row: Image reconstruction results. Bottom row: Style transfer results.

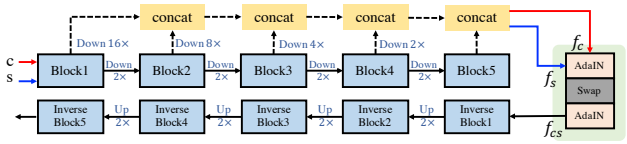


Figure 8: The framework of the style transfer used in our experiment. Here Block/Inverse Block are network sections split by every pooling/upsampling layer, respectively. For GoogLeNet, each block is an Inception modules.

alignment. Therefore, the lack of color information can be compensated by adequate textures since colors and textures are overlapped. However, because efficient networks usually have fewer feature maps, a more accurate feature alignment is crucial for efficient networks to create good style transfer results, which demonstrates the necessity of our improvement. We find that our style swap strategy achieves better style transfer quality on efficient networks and preserves more content information in style-transferred images.

Another improvement is inspired by An *et al.* [2], we concatenate deep features from all encoder blocks together and feed the concatenated f_c and f_s into $\mathtt{S}^2$ (Fig. 8). This strategy on the one hand enables $\mathtt{S}^2$ to make use of the features from high level to low level without introducing extra computational burden. On the other hand, it increases the number of feature maps used for making style transfer and therefore enhances the stylization effects. Please see Fig. 9(h) for the ablation result without this strategy. Because AdaIN is more focused on controlling colors and Style Swap is for transferring textures in $\mathtt{S}^2$, we find that $\mathtt{S}^2$ allows style transfer with a color reference and a texture reference separately. To achieve this, in $\mathtt{S}^2$, we firstly transfer the content feature f_c and texture reference feature f_{s_t}, to the domain of color reference feature f_{s_c} with the first AdaIN module. The produced content and texture reference features are $\hat{f}_c$ and $\hat{f}_{s_t}$, respectively. Then we make patch swap with respect to $\hat{f}_c$ and $\hat{f}_{s_t}$ and obtain $\hat{f}_{cs_t}$. Finally, we adopt the second AdaIN to transfer the color information of f_{s_c} to $\hat{f}_{s_t}$. The produced feature $\hat{f}_{cs_t,s_c}$ has the texture of f_{s_t} and colors of f_{s_c} while preserving the content information of f_c.

7. Experiments

In this section, we present evaluation results against state-of-the-art methods. Fig. 8 illustrates the network architecture we employ to conduct style transfer. The backbone network is based on the pruned GoogLeNet. We use the inverted version of the pruned GoogLeNet as the decoder. The style

Table 2: Quantitative evaluation results for universal stylization methods. Higher is better.

Method	StyleSwap	AdaIN	WCT	LinearWCT	OptimalWCT	Avatar-Net	**Ours**
SSIM ↑	**0.4851**	0.3525	0.2032	0.4363	0.2511	0.3829	0.4452
User Preference (%) ↑	7.38	8.20	3.28	26.50	4.37	14.48	**35.79**

Table 3: Computing-time comparison (Unit: Second). "OOM": out of the memory; "NA": not applicable at this resolution.

Method	StyleSwap	AdaIN	WCT	LinearWCT	OptimalWCT	Avatar-Net	**Ours**
128×128	0.0478	0.0037	2.6873	0.0051	0.5003	NA	**0.0142**
256×256	0.3068	0.0093	3.0805	0.0167	0.8793	0.1732	**0.0145**
512×512	1.5782	0.0344	4.1922	0.0603	1.8077	0.3718	**0.0147**
1024×1024	OOM	0.1363	OOM	0.2278	4.1589	OOM	**0.0775**

transfer module attached with the GoogLeNet is S^2. Because we are studying a more general way to accelerate style transfer, GoogLeNet can be replaced by other efficient networks such as MobileNetv2 and ResNet18. More results, analyses, and failure cases can be found in the supplementary material. All source codes will be made available to the public.

7.1. Experimental Settings

The auto-encoder we use adopts the pruned GoogLeNet as the encoder. As for the decoder, we introduce the architecturally inverted version of the encoder correspondingly. During network training, parameters of the encoder are frozen and the decoder is trained based on the MS_COCO [47] dataset to invert features back to images. We use the Frobenius norm of the original and inverted images as the reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \|I_{\text{in}} - I_{\text{out}}\|_{\text{F}}, \tag{1}$$

where I_{in} denotes the input image, I_{out} is the reconstructed image, and $\|\cdot\|_{\text{F}}$ represents the Frobenius norm. Inspired by Li et al. [43], we introduce the perceptual loss term [33] to improve the reconstruction quality of the decoder,

$$\mathcal{L}_{\text{percep}} = \sum_{i=1}^{5} \|\Phi_i\left(I_{\text{in}}\right) - \Phi_i\left(I_{\text{out}}\right)\|_F, \tag{2}$$

where $\Phi_i\left(\cdot\right)$ denotes the output of the i^{th} stage of the ImageNet [9] pre-trained VGG-19 [60]. The overall loss function is,

$$\mathcal{L} = \alpha\mathcal{L}_{\text{recon}} + \beta\mathcal{L}_{\text{percep}}, \tag{3}$$

where α and β balance two loss terms. The decoder is trained for five epochs. We use the Adam [35] algorithm with the fixed learning rate of 0.001 to minimize loss objectives.

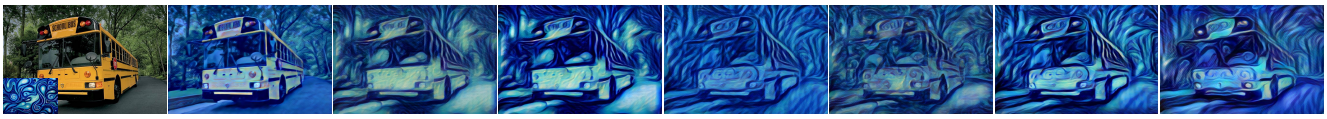
7.2. Visual Comparison

We evaluate the style transfer quality of the efficient style transfer method in comparison to state-of-the-art universal methods: StyleSwap [7], AdaIN [28], WCT [43], Avatar-Net [58], LinearWCT [42] and OptimalWCT [54].

StyleSwap does not transfer precise textures and colors as shown in Fig. 6(c). AdaIN (d) and LinearWCT (e) can generate complex details. However, the generated images are visually dissimilar to the corresponding style images in terms of colors (row 1/3) and local textures (row 2/4). WCT (f) and OptimalWCT (g) are good at creating visually pleasing local textures and bright colors. However, the style transfer results of these two approaches look fragmented such that the content is unrecognizable. Avatar-Net (h) improves the results of WCT in terms of preserving characters and objects in the content images. However, the transferred images are distorted. For example, in the first row, Avatar-Net paints the body of the bus with inconsistent blue and orange colors, but the color of the bus in the content image is uniform. Furthermore, in the second and fourth rows, Avatar-Net renders the sky in images with the colors of the land (row 2) and windows (row 4), causing artifacts in some cases. We argue that the artifacts of Avatar-Net are because the StyleDecorator module [58] may not always create accurate feature alignment as analyzed in Sec. 6. Because S^2 can make a more accurate feature alignment, our method has a more uniform stylization effect within every object of an image. For example, the bus and the street (row 1), the sky and the land (row 2) have distinct styles, but the visual effects within each object are consistent, which demonstrates the effectiveness of the improvement we make in S^2.

7.3. Quantitative Comparison

We conduct a quantitative comparison against state-of-the-art methods upon a dataset that consists of 1092 content-style pairs (42 content; 26 styles). Inspired by [74], we adopt the Structural Similarity Index (SSIM) between original contents and stylized images as the metric to measure the performance of the content preservation. Since the objective evaluation of the style transfer effect remains an open problem, we conduct a user study to subjectively assess the stylization effect by all the compared methods. We show the statistics of user study in Table 2. StyleSwap has the highest SSIM score but the lowest user preference. This is because the results of StyleSwap are more biased toward con-



(a) Input (b) AdaIN (c) Swap (d) Swap+AdaIN (e) AdaIN+Swap (f) StyleDecorator (g) Sandwich Swap (h) w/o FA

Figure 9: Ablation study for the S^2 module and feature aggregation (FA). Our result (g) is different from Avatar-Net (f).

tent preservation. Our efficient method has the highest user preference while its SSIM score ranks second to StyleSwap, suggesting that a more efficient network can achieve similar style transfer performance as heavy backbones.

7.4. Computational Time Comparison

We conduct a computing time comparison against the state-of-the-art universal methods to demonstrate the efficiency improvement by using an efficient backbone. All approaches are tested on the same computing platform which includes an NVIDIA TitanXp GPU card with 16GB RAM. We compare the computing time on content and style images of different resolutions. As Table 3 shows, our method outperforms the compared methods in terms of efficiency. Our efficient method can achieve real-time (around 68 FPS) style transfer at the resolution of 512×512.

7.5. Ablation Study

Empty channel pruning. In Fig. 7, we show the image reconstruction (top row) and style transfer (bottom row) results by using GoogLeNet *v.s.* pruned GoogLeNet (c) and MobileNetv2 (e) *v.s.* pruned MobileNetv2 (f). As shown in the top row of Fig. 7, removing empty channels does not harm the image reconstruction results of the auto-encoder. Please refer to the supplementary material for the quantitative comparison between the pruned and unpruned networks. More importantly, the style transfer results with/without removing empty channels are almost the same, which demonstrates that we can reduce the parameter size of the network by removing empty channels without hurting the quality of style transfer results. This also explains the reason that we do not need to fine-tune the model after removing empty channels.
Sandwich Swap Module. To demonstrate the effectiveness of S^2 module, we conduct an ablation study on each element of it. As shown in Fig. 9(b), the AdaIN [28] module can transfer holistic global appearance (*e.g.*, colors). However, it does not transfer fine textures. Fig. 9(c) shows the style transfer results only with the style swap module [7]. The results by style swap contain rich details but imperfect colors. The prior AdaIN in S^2 is to project f_c into the domain of f_s, and thus correct the bias of the optimal matching (Fig. 9(e)). In addition, the posterior AdaIN can rematch the mean and variance of the output feature after the style swapping to f_s, which can correct the color aberration introduced by the style swap module (Fig. 9(d)). Fig. 9(g) shows that the style transfer results by S^2 contain better textures and more accurate colors.

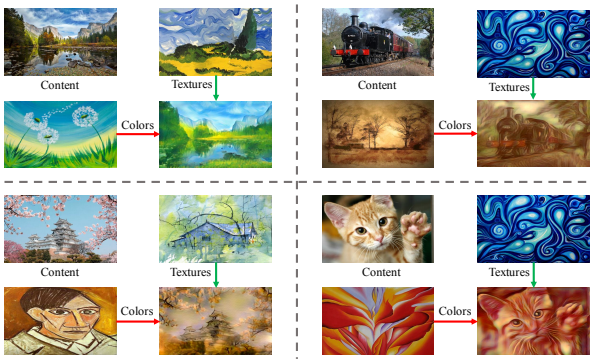


Figure 10: Style transfer results by mixing textures and colors from two separate reference images.

7.6. Transferring Color and Texture Separately

By using S^2, we can generate an image by mixing *textures* and *colors* from two separate reference images. Fig. 10 shows style mixing results. In the top-left example, the generated image brings green global color from the color reference on the left and has the texture of the reference image on the top. We find that textures in the generated images by style mixing are not as good as the style transfer results with only one reference. We think it is because colors and textures are implicitly overlapped and thus the transferred textures by style swap (2nd layer) will be disturbed when the posterior AdaIN (3rd and last layer) in S^2 is applied.

8. Conclusion

In this paper, we present an empirical study on efficient network architectures that can replace time-consuming VGG19 in style transfer. We find that some architectures such as GoogLeNet are more lightweight yet have comparable style transfer quality with VGG19. Furthermore, we show that the feature maps of efficient networks contain a few empty channels, and removing empty channels by a channel pruning method can further improve the efficiency. By removing the empty channels, GoogLeNet achieves real-time efficiency in style transfer on high-resolution images. Moreover, we introduce a sandwich swap transform (S^2) module based on StyleDecorator to transfer artistic styles at the bottleneck of the auto-encoder. S^2 improves the accuracy of the feature alignment in style transfer, thus leading to better content preservation in style transfer. The extensive experiments demonstrate that by replacing VGG19 with a more lightweight network (*e.g.*, the pruned GoogLeNet) together with S^2, we can accelerate style transfer significantly while maintaining a comparable style transfer quality.

References

[1] Jie An, Siyu Huang, Yibing Song, Dejing Dou, Wei Liu, and Jiebo Luo. Artflow: Unbiased image style transfer via reversible neural flows. In *CVPR*, 2021.

[2] Jie An, Haoyi Xiong, Jun Huan, and Jiebo Luo. Ultrafast photorealistic style transfer via neural architecture search. In *AAAI*, 2020.

[3] Miguel A Carreira-Perpinán and Yerlan Idelbayev. "learning-compression" algorithms for neural net pruning. In *CVPR*, 2018.

[4] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: an explicit representation for neural image style transfer. In *CVPR*, 2017.

[5] Haibo Chen, Zhizhong Wang, Huiming Zhang, Zhiwen Zuo, Ailin Li, Wei Xing, Dongming Lu, et al. Artistic style transfer with internal-external learning and contrastive learning. *NeurIPS*, 2021.

[6] Haibo Chen, Lei Zhao, Zhizhong Wang, Huiming Zhang, Zhiwen Zuo, Ailin Li, Wei Xing, and Dongming Lu. Dualast: Dual style-learning networks for artistic style transfer. In *CVPR*, 2021.

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

[8] Jiaxin Cheng, Ayush Jaiswal, Yue Wu, Pradeep Natarajan, and Prem Natarajan. Style-aware normalized loss for improving arbitrary style transfer. In *CVPR*, 2021.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: a large-scale hierarchical image database. In *CVPR*, 2009.

[10] Yingying Deng, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, and Changsheng Xu. Arbitrary video style transfer via multi-channel correlation. *arXiv preprint arXiv:2009.08003*, 2020.

[11] Yingying Deng, Fan Tang, Weiming Dong, Wen Sun, Feiyue Huang, and Changsheng Xu. Arbitrary style transfer via multi-adaptation network. In *ACM MM*, 2020.

[12] Xiaohan Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, Ji Liu, et al. Global sparse momentum sgd for pruning very deep neural networks. In *NeurIPS*, 2019.

[13] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

[14] Oriel Frigo, Neus Sabater, Julie Delon, and Pierre Hellier. Split and match: example-based adaptive patch sampling for unsupervised style transfer. In *CVPR*, 2016.

[15] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015.

[16] Leon A Gatys, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016.

[17] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[18] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[19] Xinyu Gong, Haozhi Huang, Lin Ma, Fumin Shen, Wei Liu, and Tong Zhang. Neural stereoscopic image style transfer. In *ECCV*, 2018.

[20] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary style transfer with deep feature reshuffle. In *CVPR*, 2018.

[21] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[23] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, 2018.

[24] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, 2019.

[25] Zhiyuan Hu, Jia Jia, Bei Liu, Yaohua Bu, and Jianlong Fu. Aesthetic-aware image style transfer. In *ACM MM*, 2020.

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[27] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. In *CVPR*, 2017.

[28] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[29] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[30] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[31] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[32] 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.

[33] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[34] Nikolai Kalischek, Jan D Wegner, and Konrad Schindler. In the light of feature distributions: moment matching for neural style transfer. In *CVPR*, 2021.

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

[36] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[38] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[39] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[40] Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. Laplacian-steered neural style transfer. In *ACM MM*, 2017.

[41] Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. Laplacian-steered neural style transfer. In *ACM MM*, 2017.

[42] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *CVPR*, 2019.

[43] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NeurIPS*, 2017.

[44] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *ECCV*, 2018.

[45] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[46] Tianwei Lin, Zhuoqi Ma, Fu Li, Dongliang He, Xin Li, Errui Ding, Nannan Wang, Jie Li, and Xinbo Gao. Drafting and revision: Laplacian pyramid network for fast high-quality artistic style transfer. In *CVPR*, 2021.

[47] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: common objects in context. In *ECCV*, 2014.

[48] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017.

[49] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *ICCV*, 2019.

[50] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, 2016.

[51] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *CVPR*, 2021.

[52] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *CVPR*, 2017.

[53] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.

[54] Lu Ming, Zhao Hao, Yao Anbang, Chen Yurong, Xu Feng, and Zhang Li. A closed-form solution to universal style transfer. In *ICCV*, 2019.

[55] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[56] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[57] Ahmed Selim, Mohamed Elgharib, and Linda Doyle. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics*, 2016.

[58] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018.

[59] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[60] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[63] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017.

[64] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[65] D Ulyanov, A Vedaldi, and VS Lempitsky. Instance normalization: the missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[66] Dmitry Ulyanov, Andrea Vedaldi, and Victor S Lempitsky. Improved texture networks: maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017.

[67] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative distillation for ultra-resolution universal style transfer. In *CVPR*, 2020.

[68] Pei Wang, Yijun Li, and Nuno Vasconcelos. Rethinking and improving the robustness of image style transfer. In *CVPR*, 2021.

[69] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[70] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. Multimodal transfer: a hierarchical deep convolutional neural network for fast artistic style transfer. In *CVPR*, 2017.

[71] Hao Wu, Zhengxing Sun, and Weihang Yuan. Direction-aware neural style transfer. In *ACM MM*, 2018.

[72] Xide Xia, Tianfan Xue, Wei-sheng Lai, Zheng Sun, Abby Chang, Brian Kulis, and Jiawen Chen. Real-time localized photorealistic video style transfer. In *WACV*, 2021.

[73] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[74] Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha. Photorealistic style transfer via wavelet transforms. In *ICCV*, 2019.

[75] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *CVPR*, 2018.

[76] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[77] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017.

[78] Yuheng Zhi, Huawei Wei, and Bingbing Ni. Structure guided photorealistic style transfer. In *ACM MM*, 2018.

[79] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.