Rethinking and Improving the Robustness of Image Style Transfer

Pei Wang
UC, San Diego
pew062@ucsd.edu

Yijun Li
Adobe Research
yijli@adobe.com

Nuno Vasconcelos
UC, San Diego
nuno@ucsd.edu

Abstract

Extensive research in neural style transfer methods has shown that the correlation between features extracted by a pre-trained VGG network has a remarkable ability to capture the visual style of an image. Surprisingly, however, this stylization quality is not robust and often degrades significantly when applied to features from more advanced and lightweight networks, such as those in the ResNet family. By performing extensive experiments with different network architectures, we find that residual connections, which represent the main architectural difference between VGG and ResNet, produce feature maps of small entropy, which are not suitable for style transfer. To improve the robustness of the ResNet architecture, we then propose a simple yet effective solution based on a softmax transformation of the feature activations that enhances their entropy. Experimental results demonstrate that this small magic can greatly improve the quality of stylization results, even for networks with random weights. This suggests that the architecture used for feature extraction is more important than the use of learned weights for the task of style transfer.

1. Introduction

Image style transfer aims to map a content image into the style of a different reference image. It has received substantial attention, particularly with the introduction of neural style transfer algorithms based on deep networks. A consistent observation from this work [43, 24, 30, 23, 25, 19, 9, 2, 8] is that the correlation between the activations of a pre-trained VGG [33] network has remarkable ability to capture the visual style of an image. It is, however, puzzling that when the VGG is replaced by architectures of better performance in other tasks, e.g. classification, such as the ResNet [14, 44, 42], InceptionNet [35, 34, 36] or DenseNet [16], stylization performance degrades significantly. This is even more puzzling because, when implemented with the VGG, style transfer is very robust. For example, a VGG model with random weights performs comparably to a pre-trained model [14, 3].

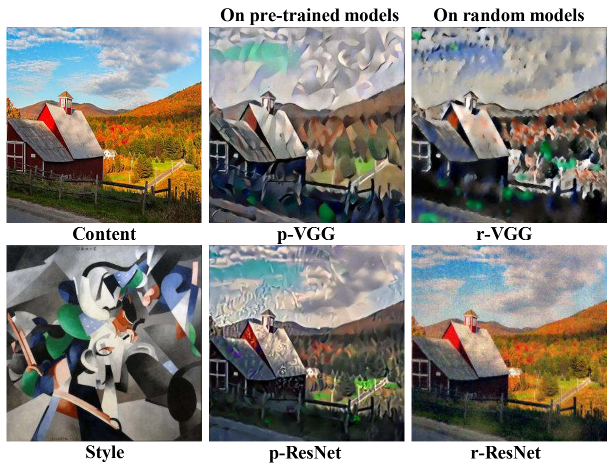


Figure 1: Neural style transfer by different architectures, using the methods of [7, 27] ('p-', 'r-' denotes pre-trained and randomly initialization. Please zoom in the picture for a detailed comparison).

Figure 1 shows an example of style transfer using different models. The VGG transfers style (color, texture, strokes) more faithfully than the ResNet, for both pre-trained and random weights. While these observations have spurred discussion in the literature, about why style transfer is much more effective for the VGG [27, 12, 4], there are still no clear answers. In particular, there has not been a comprehensive study of (i) what architectural differences between the VGG and other networks cause this striking performance difference, and (ii) what remedies could make non-VGG networks perform as well as the VGG. One explanation is that VGG features are more robust than others. To validate this conjecture, [27, 4] trained a ResNet with adversarial examples [10] to improve feature robustness. They found this can significantly improve stylization quality. However, the fact that the VGG with random weights can generate comparable results [13, 3] suggests that robustness is not a property of the training data, but inherent to the architecture.

In this work, we investigate this hypothesis by comparing stylizations produced with activations from different architectures. We seek the architectural properties that explain the differences between these activations, and how these could explain the discrepancy between stylization results. Taking the ResNet as a non-VGG architecture representative, we study the statistics of both activations and the derived Gram matrices, usually used to encode image style. A striking observation is that, when normalized into a probability distribution, the ResNet activations of deeper layers have large peaks and small entropy. This shows that they are dominated by a few feature channels and have nearly deterministic correlation patterns. It suggests that the optimization used to synthesize the stylized images is biased into replicating a few dominant patterns of the style image and ignoring the majority. This explains why the ResNet is unable to transfer high-level style patterns, such as strokes, that are usually captured in deeper layers of the network. In contrast, VGG activations are approximately uniform for all layers, capturing a much larger diversity of style patterns. We then analyze the architectural properties that could lead to very peaky activations, and conclude that they can be, in significant part, explained by the existence of residual or shortcut connections between layers. The fact that these connections are prevalent in most modern architectures explains why the robustness problem is so widespread. In summary, *residual connections are not good for style transfer.*

We then investigate whether it is possible to solve the robustness problem without changing the network architecture, and in a manner that is compatible with the large diversity of stylization losses in the literature. Taking inspiration from knowledge distillation, we propose to smooth the activations used in the computation of these loss functions. This can be implemented by adding a simple softmax transformation to existing losses. We denote the novel version of stylization as *Stylization With Activation smoothinG* (SWAG). Experiments show that SWAG is an important contribution at three levels. First, it improves the performance of several popular stylization algorithms for several popular architectures, including ResNet, Inception, and WideResNet. Second, for these architectures, it improves the performance of random networks to the level of pre-trained ones. Third, for pre-trained networks, non-VGG models with SWAG can even outperform the VGG with standard stylization.

2. Related Work

Stylization. For an extensive review of stylization methods please see [21]. Starting with [7], it has been shown that a pre-trained image classifier can be used as a feature extractor to drive style transfer [7, 22, 41]. Style transfer algorithms either implement an iterative optimization [7, 3, 13], or directly learn a feed-forward generator network [22, 40, 37]. Unlike all these papers, we do not propose a new stylization algorithm. Instead, we note that all previous algorithms use VGG pre-trained models [43, 30, 23, 25, 19, 41, 2, 8, 26]. In fact, even recent GAN-based proposals [46] use a VGG encoder. We seek to make these algorithms applicable to a broader set of network architectures. To the best of our knowledge, no previous work discusses the architecture robustness of style transfer.

Random Networks. Our work is partly motivated by some theoretical and practical studies on random weights [17, 18, 29, 28, 40, 39, 5]. For example, Gaier et al. [5] propose the weight agnostic neural network, by fixing randomly initialized weights and searching optimal network architectures, which is shown to achieve good results in several reinforcement and supervised learning tasks. [38] studies several low-level vision problems and uses random weights to show that the structure of a generator network is sufficient to capture image statistics. [1] compares the performance of many saliency algorithms based on random and pre-trained weights. He et al. [13] used a random weight network for texture synthesis and neural style transfer. However, their approach does not use genuinely random weights, as discussed in Section 3.2. In contrast, all random models and results presented in this work are based on purely random weights.

3. Robust Stylization

3.1. Preliminaries

Consider a color image $\mathbf{x}_0 \in \mathbb{R}^{W_0 \times H_0 \times 3}$, where W_0 and H_0 are the image width and height. A convolutional neural network (CNN) maps $\mathbf{x}_0$ into a set of feature maps $\{F^l(\mathbf{x}_0)\}_{l=1}^L$, where $F^l : \mathbb{R}^{W_0 \times H_0 \times 3} \to \mathbb{R}^{W_l \times H_l \times D_l}$ is the mapping from the image to the tensor of activations of the l^{th} layer, which has D_l channels of spatial dimensions $W_l \times H_l$. The activation tensor $F^l(\mathbf{x}_0)$ can also be reshaped into a matrix $F^l(\mathbf{x}_0) \in \mathbb{R}^{D_l \times M_l}$, where $M_l = W_l H_l$. Image style is frequently assumed to be encoded by a set of Gram matrices $\{G^l\}_{l=1}^L$ where $G^l \in \mathbb{R}^{D_l \times D_l}$ is derived from the activations F^l of layer l by computing the correlation between activation channels, i.e.

$$[G^l(F^l)]_{ij} = \sum_k F^l_{ik} F^l_{jk}. \tag{1}$$

To simplify the discussion, we focus on the Gram matrix loss [7, 24, 37, 3, 13]. However, in the experiment section, we show that all results hold for other losses. We consider the image stylization framework of [7], where given a content image $\mathbf{x}_0^c$ and a style image $\mathbf{x}_0^s$, an image $\mathbf{x}^*$ that presents the content of $\mathbf{x}_0^c$ under the style of $\mathbf{x}_0^s$ is synthesized by solving

$$\mathbf{x}^* = \underset{x \in \mathbb{R}^{W_0 \times H_0 \times 3}}{\operatorname{argmin}} \alpha \mathcal{L}_{\text{content}}(\mathbf{x}_0^c, \mathbf{x}) + \beta \mathcal{L}_{\text{style}}(\mathbf{x}_0^s, \mathbf{x}). \tag{2}$$

with

$$\mathcal{L}_{\text{content}}(\mathbf{x}_0^c, \mathbf{x}) = \frac{1}{2} ||F^l(\mathbf{x}) - F^l(\mathbf{x}_0^c)||_2^2, \tag{3}$$

$$\mathcal{L}_{\text{style}}(\mathbf{x}_0^s, \mathbf{x}) = \sum_{l=1}^{L} \frac{w_l}{4D_l^2 M_l^2} ||G^l(F^l(\mathbf{x})) - G^l(F^l(\mathbf{x}_0^s))||_2^2, \tag{4}$$

where $w_l \in \{0, 1\}$ are weighting factors of the contribution of each layer to the total loss. $w_l = 1$ represents that l^{th} is used otherwise ignored. l and w_l are pre-specified in (3) and (4). $\mathbf{x}$ is usually initialized to $\mathbf{x}_0^c$.

3.2. Importance of residual connections

We present results for both pre-trained models and networks initialized with random weights. Prefixes 'r-' and 'p-' are used to indicate if a model is initialized randomly or pre-trained on ImageNet, respectively. We follow the setup of [7] for the VGG model, simply replacing their pre-trained models with random networks in some experiments. For ResNet, we follow the setting of [27] but, in addition to the outputs of layers conv2_3, conv3_4, conv4_6, conv5_3, we also use that of layer conv1_2 in (4). This is for fair comparison to the VGG implementation of [7], which uses five layers, each selected from one of five layer-groups. VGG19 and ResNet-50 are used as VGG and ResNet models, because the former is the network of choice for stylization papers and the latter one of the most popular deep learning models. Random weights follow default PyTorch settings[1]. Note that this is unlike the random network set-up of [13], which samples several sets of random weights per layer, reconstructs the target image, and chooses the weights yielding the smallest loss[2] [13, 3].

Figures 2b to 2e present two examples of stylization by the r-VGG, p-VGG, r-ResNet and p-ResNet networks, showing that performance varies drastically with the network architecture. Compared to p-VGG, the p-ResNet transfers much lower-level color patterns and produces much noisier stylized images. This discrepancy is even more obvious for random models, with the r-ResNet simply failing to stylize the content image. To investigate the reasons behind the very different performance of two architectures, we performed an ablation study over many network components, including the use of residual connections, convolution kernels varying among size 1×1, 3×3 to 7×7, variable network depth, batch normalization, number of channels per layer, a fixed stride of 2 vs. maxpooling, etc. While detailed results are presented in the supplementary, the main conclusion was that the poor performance of the ResNet is mainly explained by its residual connections. This is interesting, since residual connections are usually seen as the main asset of this architecture for tasks like classification.

Figure 2 provides evidence for this claim, comparing the stylized outcomes by several architectures after deletion or addition of residual connections. Starting from the ResNet-50, we built a 'NoResNet' by removing all residual connections. As can be seen in Figure 2f, this drastically improves style transfer performance. r-NoResNet has much closer performance to r-VGG than to r-ResNet. We next considered the benefits of several other modifications that made the NoResNet more similar to the VGG: 1) replaced its 7×7 conv kernel with the 3×3 conv kernel of the VGG; 2) replaced the bottleneck module with the basicblock module of the ResNet-34 (see Figure 5) without the residual connection; 3) inserted a maxpooling layer between each stage to decrease the size of feature maps, as done in the VGG. The resulting architecture is denoted as 'pseudo-VGG'. As shown in Figure 2g, these modifications made the stylization performance even closer to that of the r-VGG. However, by comparing the r-ResNet, r-NoResNet, and r-pseudo-VGG stylizations, it is clear that the bulk of the gains are due to the deletion of residual connections, i.e., from ResNet to NoResNet. To further confirm this, we re-introduced the residual connections in the pseudo-VGG network, to create a 'pseudo-ResVGG'. Figure 2h shows that this destroyed all the gains of the pseudo-VGG, again producing undiscernable styles. In fact, this network produced the worst results of Figure 2, showing that there is no benefit to the pseudo-ResVGG over the ResNet. In summary, the weak ResNet performance is due to its residual connections.

3.3. Why do residual connections degrade performance?

We next try to understand why residual connections are so nefarious for stylization. Since the optimization of (4) is, in this case, solely based on the Gram matrices G^l of the network responses to the original and synthesized style, we start by visualizing the statistics of the network activations and their Gram matrices. Figure 3 presents the average of maximum values $\max_{i,k} F^l_{i,k}$ and $\max_{i,j} G^l_{i,j}$, as well as normalized entropies [11][3]

$$H(F^l_{i,k}) = -\frac{1}{\log(D_l M_l)^2} \sum_{i,k} p(F^l_{i,k}) \log p(F^l_{i,k}) \quad (5)$$

$$p(F^l_{i,k}) = \frac{e^{F^l_{i,k}}}{\sum_{m,n} e^{F^l_{m,n}}} \quad (6)$$

$$H(G^l_{i,j}) = -\frac{1}{\log D_l^2} \sum_{i,j} p(G^l_{i,j}) \log p(G^l_{i,j}), \quad (7)$$

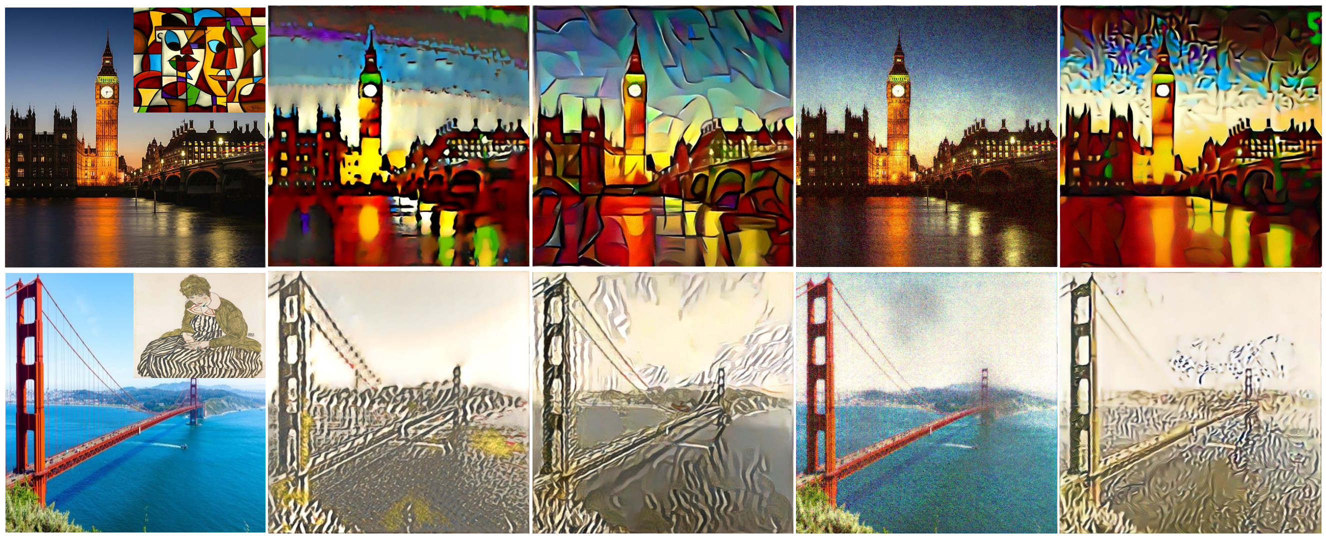
$$p(G^l_{i,j}) = \frac{e^{G^l_{i,j}}}{\sum_{m,n} e^{G^l_{m,n}}} \quad (8)$$

of the activations and Gram matrices, respectively, of the last layer of each network stage of the random models on 10 style images, where i, j, k are the spatial coordinates defined in (1). The figure shows that activations and Gram values have similar behavior. In both cases, the maximum value increases and the entropy decreases gradually with layer depth for the architectures with residual connections (ResNet and pseudo-ResVGG). This is unlike the networks without shortcuts (NoResNet and pseudo-VGG), where activations

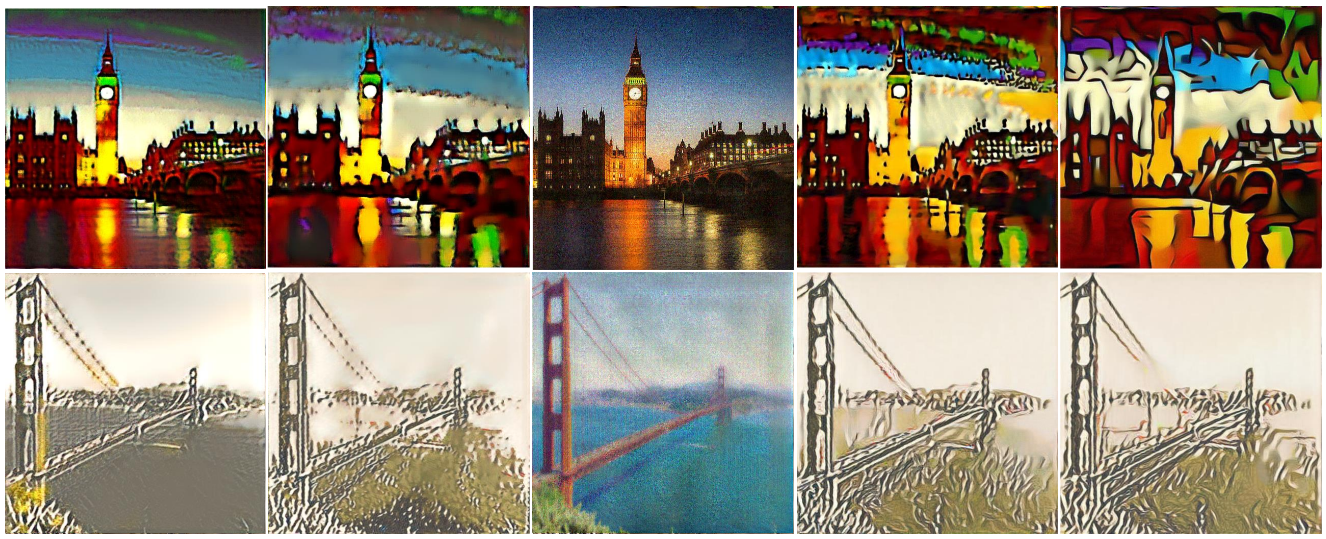
[1]Convolutional layers use kaiming initialization [32, 31]; batchNorm weights (biases) are set to 1 (0); weights of VGG fully connected layers are drawn from a normal distribution $\mathcal{N}(0, 0.01)$ and biases set to 0.

[2]We believe that this set-up leads to an unfair comparison to [7], since it leverages the target image to choose the best random weights and extremely time-consuming gradient computations to add weight factors in (4).

[3]We omit the dependency on $\mathbf{x}$ on these equations for simplicity.

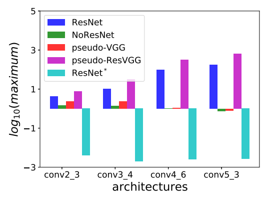


(a) content^{style} (b) r-VGG (c) p-VGG (d) r-ResNet (e) p-ResNet

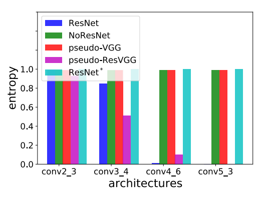


(f) r-NoResNet (g) r-pseudo-VGG (h) r-pseudo-ResVGG (i) r-ResNet* (j) p-ResNet*

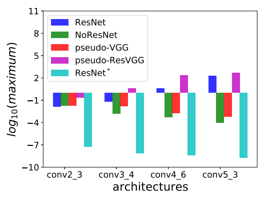
Figure 2: Stylization by different architectures. (‘p-’, ‘r-’ represent pre-trained and randomly initialized, ‘*’ denotes SWAG.)



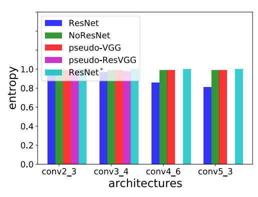
(a) Activation maxima.



(b) Activation entropy.



(c) Gram maxima.



(d) Gram entropy.

Figure 3: Activation statistics of different random architectures.

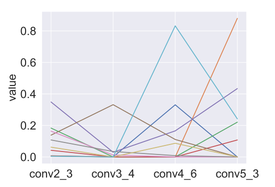


Figure 4: Activation tracks across layers.

tend to decrease and entropies remain fairly constant and much higher. In some cases, e.g. pseudo-ResVGG, the introduction of residual connections translates into large maximum and near-zero entropies for the deeper layers, i.e. activations that are dominated by a single feature channel and deterministic correlation patterns.

The observation of small entropies is consistent with at least two explanations for poor stylization performance. Both of these follow from the fact that the only variables in the optimization of (4) are the activations $F^l(\mathbf{x})$, which are encouraged to have Gram matrix as similar as possible to those of the style image $F^l(\mathbf{x}_0^s)$. A first explanation is derived from the well known outlier sensitivity of the L_2 distance [20]. Due to this outlier sensitivity, when the Gram matrices derived from $\mathbf{x}_0^s$ are "peaky" (low entropy), the optimization will focus mostly on creating equal peaks on the matrices produced by $\mathbf{x}^*$, while paying less attention to the remaining entries of the Gram matrix. From (1), the Gram matrix value of location i, j is a measure of similarity of the vectors of activations $F^l_{i,k}$ and $F^l_{j,k}$ across the channel depth dimension k. Hence, the Gram matrix peaks identify pairs of locations of strong activations that are highly correlated across the channel dimension. By giving disproportionate emphasis to these location pairs, the optimization overfits on a few style patterns, ignoring most of the remaining. This explains why small entropies degrade stylization. Note that the appearance of Gram peaks in the deeper layers is consistent with the stylization results of Figure 2. While the r-ResNet and r-pseudo-ResVGG can transfer the localized color patterns captured by the earlier layers, they fail to capture the long-range correlations that are essential for texture and style perception and only accessible in the later layers.

A second explanation derives from interpreting stylization as a knowledge distillation problem [15]. For classification, neural networks are typically trained to minimize a cross-entropy loss between the posterior distribution $\mathbf{q}$ and a target distribution $\mathbf{p}$, which is typically a one-hot code. Knowledge distillation aims to, instead, minimize the distance between the distribution of a student network $\mathbf{q}$ and that of a soft teacher network $\mathbf{p}$. [15] has shown that using the soft probability output of a pre-trained larger network as target $\mathbf{p}$ can improve training speed and converge to a better model than using the hard one-hot target. This is because a teacher distribution of high entropy produces gradients of much less variance during training. The same rationale can be applied to stylization, which can be seen as a form of knowledge distillation, where the optimization of (2) seeks the $\mathbf{x}^*$ that minimizes the distance between the 'student' ($F^l(\mathbf{x})$, $G^l(F^l(\mathbf{x}))$) and 'teacher' ($F^l(\mathbf{x}_0^c), G^l(F^l(\mathbf{x}_0^s))$) pairs of activations and Gram matrices. Under the distillation view, pairs of higher entropy should make learning easier.

3.4. Why are residual network activations and Gram matrices peaky?

We next investigate why residual networks produce peaky activations and Gram matrices. We start by recalling the details of the bottleneck ResNet block, shown in Figure 5. At layer l, this module computes a residual $R(F^{l-1}(\mathbf{x}))$, which is added to the output $F^{l-1}(\mathbf{x})$ of layer $l-1$ to create

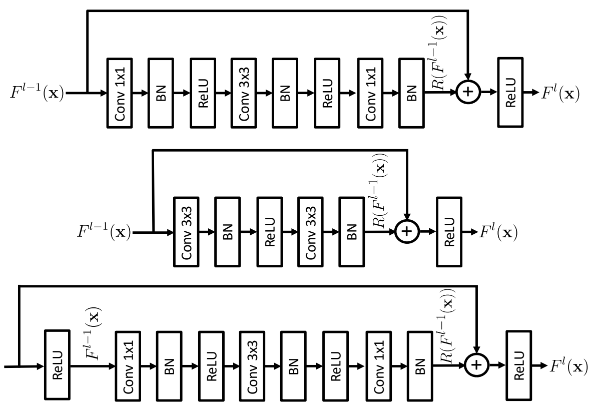


Figure 5: Top: bottleneck; Middle: Basicblock; Bottom: a variation of bottleneck.

the output of layer l

$$F^l(\mathbf{x}) = R(F^{l-1}(\mathbf{x})) + F^{l-1}(\mathbf{x}), \tag{9}$$

where $R(\cdot)$ is implemented with a sequence of convolutional (Conv), batch normalization (BN) and ReLU layers. An important detail is that the addition of (9) is computed *between* the last BN and ReLU layers in the module, as shown at the top and middle of Figure 5, i.e. (9) is effectively implemented as

$$F^l(\mathbf{x}) = ReLU\left(R(F^{l-1}(\mathbf{x})) + F^{l-1}(\mathbf{x})\right). \tag{10}$$

This design choice contributes to the existence of large activation maxima for deeper layers. Note that $F^{l-1}(\mathbf{x})$ is the output of a ReLU layer, i.e. a positive number, and $R_i(F^{l-1})$, a real number. It follows that, for any i, there are at least two ways in which $F^l_i(\mathbf{x})$ can be large: 1) a large residual $R_i(F^{l-1}(\mathbf{x}))$, or 2) a positive residual $R_i(F^{l-1}(\mathbf{x}))$ if $F^{l-1}_i(\mathbf{x})$ is already large. The second condition is particularly prone to creating large peaks, since positive residuals enable $F_i(\mathbf{x})$ to grow from layer to layer. In fact, the only way to cancel a large $F^{l-1}_i(\mathbf{x})$ is to make $R_i(F^{l-1}(\mathbf{x}))$ *large and negative.* However, it may be impossible to generate a large negative residual for one channel without generating large positive residuals for others. Hence, the attempt to correct for the second condition could create the first. This would imply that, once a large activation emerges at an intermediate layer, the network could be forced into a game of "whack-a-mole," producing large amplitudes for the subsequent layers.

To investigate this hypothesis, we tracked the evolution of activations through the network. For a randomly selected style image, we randomly sampled image positions and tracked the corresponding activation values accross the network layers, using nearest-neighbor interpolation. Figure 4 shows a typical random sample of 10 activation tracks. The "whack-a-mole" effect is visible even in this limited sample. The network attempts to mitigate the peaks that appear in intermediate layers, but this creates larger picks on other channels of the subsequent layers. Note how the maximum response increases with layer depth. Figure 4 also shows that, beyond the large picks, there are many small activations in the deeper layers. This is also consistent with the ”whack-a-mole” hypothesis. When combined with the ReLU of (10), the impetus to produce large negative residuals results in many activations $F^l(\mathbf{x})$ of small or zero value. Hence, as depth l increases, the activations become peakier and of lower entropy. From (1), the Gram matrices are then likely to have the same property.

It should be noted that this discussion applies to the standard residual connection structure of the ResNet [14], which is widely used. One possibility to improve stylization performance would be to change the architecture. An example is given at the bottom of Figure 5, where the skip connection is moved to precede the output ReLU of layer $l-1$. Since, in this case, the addition of activation and residual operates on two real numbers, the pressure for activations to grow would be smaller. We have tried several of these variations, but found that they tend to result in ineffective architectures for image recognition. The example modification of Figure 5 reduces the mean classification accuracy of the ResNet-50 on CIFAR-100 (ImageNet) from 77% to 65% (76% to 62%). Hence, even if it turned out to enhance stylization performance, it would result on different architectures for stylization and classification, which is undesirable. In what follows, we show that it is possible to improve stylization *while maintaining the widely used network architecture*.

3.5. Stylization With Activation Smoothing

In this work, we leverage the observations above to improve the stylization performance of networks with residual connections. We propose a very simple solution, inspired by the interpretation of stylization as knowledge distillation [15], where significant gains are observed by smoothing teaching distributions. In the same vein, we propose to avoid peaky activations of low entropy, by smoothing all activations with a softmax-based smoothing transformation [4]

$$\sigma(F^l_{ik}(\mathbf{x})) = \frac{e^{F^l_{ik}(\mathbf{x})}}{\sum_{m,n} e^{F^l_{mn}(\mathbf{x})}}, \tag{11}$$

Note that the softmax layer is *not* inserted in the network, which continues to be the original model, but only used to redefine the style and content loss functions of (2), which become

$$\mathcal{L}_{\text{content}}(\mathbf{x}^c_0, \mathbf{x}) = \frac{1}{2}||\sigma(F^l(\mathbf{x})) - \sigma(F^l(\mathbf{x}^c_0))||^2_2, \tag{12}$$

$$\begin{aligned}\mathcal{L}_{\text{style}}(\mathbf{x}^s_0, \mathbf{x}) = \sum_l^L \frac{w_l}{4D_l^2 M_l^2}||G^l(\sigma(F^l(\mathbf{x}))) \\ - G^l(\sigma(F^l(\mathbf{x}^s_0))||^2_2.\end{aligned} \tag{13}$$

The softmax transformation reduces large peaks and increases small values, creating a more uniform distribution. Since this can be seen as a form of smoothing, we denote this approach to stylization as *Stylization With Activation smoothinG* (SWAG), and denote the resulting models with a '$*$' superscript, e.g. r-ResNet*. ResNet* of figure 3 shows that SWAG successfully suppresses the maxima and increases entropies, especially on deeper layers.

The impact of activation smoothing on stylization performance is illustrated in Figure 2i and 2j, where SWAG results are presented for the r-ResNet* and p-ResNet* networks. In both cases, the quality of the stylized images improves substantially after smoothing, in the sense that more high-level style patterns are transferred. The results of the r-ResNet* approach those of the r-VGG and it could be claimed that the p-ResNet* outperforms the p-VGG. This is next evaluated quantitatively. It should be noted that there are other ways of smoothing activations and decreasing their entropy, e.g. softmax with different temperature, nested softmax, or even multiplying a small constant (< 0.1), that we found working in our experiments. We chose the softmax of (11) because it is simple, hyperparameter-free, and achieves similar effects with other smoothing methods.

4. Experimental Evaluation

In this section we discuss an experimental evaluation of the stylization gains of SWAG models. More results and implementation details can be found in the supplementary materials.

4.1. Qualitative evaluation

We start by evaluating SWAG for two popular non-VGG architectures, other than the ResNet, that also rely on shortcut connections: Inception-v3 [36] and Wide ResNet (WRN) [45]. WRN experiments use all settings of the ResNet, while conv2d_1a, conv2d_3b, mixed_5b, mixed_6a, mixed_7a leayers are used for Inception. This is, again, for consistency with the VGG model of [7]. We denote VGG, ResNet, WRN, and Inception-v3 networks as '**V**', '**R**', '**W**', and '**I**', for brevity. Figure 6 presents style transfer results on four different images, using the pre-trained versions of all networks, comparing results of SWAG ($*$) and standard

[4]We experimented with adding a temperature parameter, but this made no difference. The detailed discussion can be found in the supp.

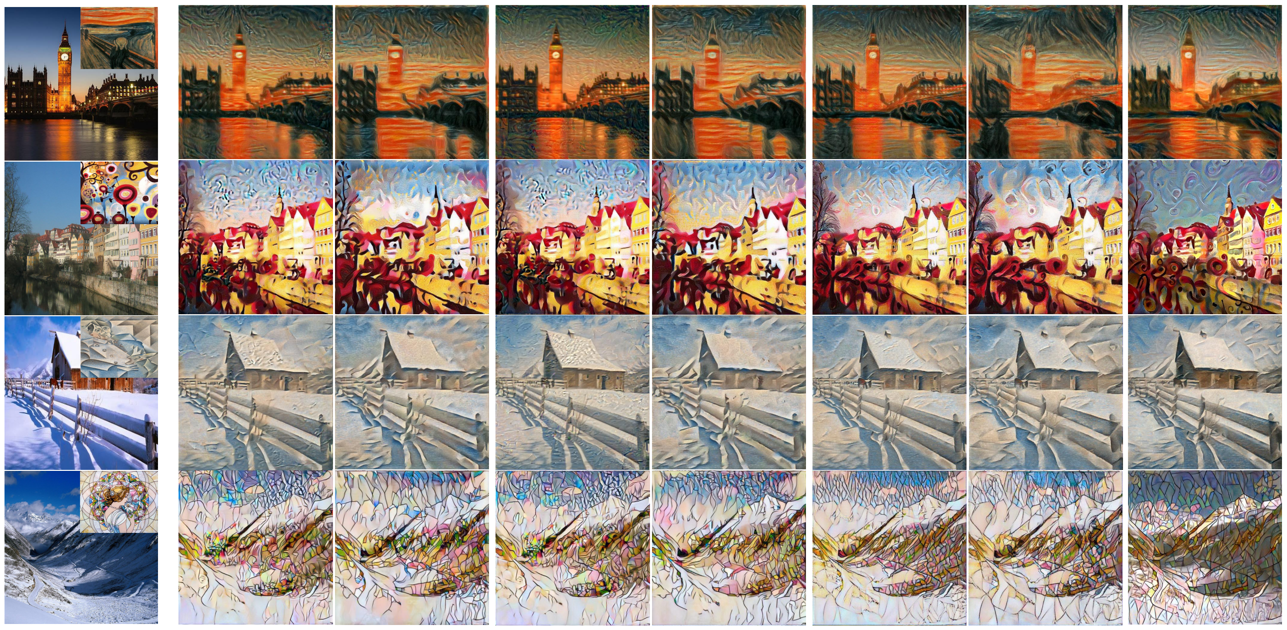

(a) content$^{\text{style}}$ (b) p-R vs p-R* (c) p-W vs p-W* (d) p-I vs p-I* (e) p-VGG

Figure 6: Comparison of neural style transfer performance between standard and SWAG (denoted with *) models, for different pre-trained architectures with shortcut connections (R: ResNet, W: WRN, I: Inception). The results of the standard VGG model are also shown for comparison.

Arch.	p-R / p-R*	p-I / p-I*	p-W / p-W*	p-V / p-V*	p-V / p-R*	p-V / p-I*	p-V / p-W*
Pref.(%)	17.0 / $\mathbf{83.0}_{7.1,4.3}$	34.7 / $\mathbf{65.3}_{6.2,5.4}$	32.3 / $\mathbf{67.7}_{3.4,5.3}$	48.3 / $\mathbf{51.7}_{1.2,5.5}$	33.3 / $\mathbf{66.7}_{3.7,5.3}$	34.7 / $\mathbf{65.3}_{4.2,5.4}$	37.3 / $\mathbf{62.7}_{3.1,5.5}$
Arch.	r-R / r-R*	r-I / r-I*	r-W / r-W*	r-V / r-V*	r-V / r-R*	r-V / r-I*	r-V / r-W*
Pref.(%)	15.7 / $\mathbf{84.3}_{4.6,4.1}$	10.7 / $\mathbf{89.3}_{2.3,3.5}$	5.4 / $\mathbf{94.6}_{5.3,2.6}$	44.3 / $\mathbf{55.7}_{2.3,6.2}$	23.3 / $\mathbf{76.7}_{6.7,4.8}$	18.7 / $\mathbf{81.3}_{2.1,4.4}$	25.3 / $\mathbf{74.7}_{3.4,4.9}$

Table 1: Comparison of user preference (%), mean$_{\text{std, confidence interval}}$ (conf. interval at 95% conf. level). (R: ResNet; I: Inception-v3; W: WRN; V: VGG. r-: random; p-: pre-trained. *: SWAG).

stylization. Note that p-R* and p-W* transfer more high-level style features, such as strokes, and textures. For p-I*, the improvement is smaller. We speculate that this is due to the somewhat different structure of the Inception whose basic module has multiple parallel connections and merges features of different solutions. This may make the stylization optimization harder. We have not investigated the issue in detail. However, the improvement is still visible.

We next evaluate SWAG with other two stylization algorithms [22, 24]. Unlike [7], [22] trains a style-specific transformer to directly transfer the target to the stylized image, using a perceptual loss. [24], which trains a single transformer for all types of styles, is the state-of-the-art for universal stylization. Figure 7 compares the results of SWAG implementations of the two algorithms, for different networks. For both random and pre-trained models, the performance of the two algorithms improves significantly under SWAG. For example, [24] fails for the p-ResNet, producing many repeated image patches. SWAG improves its results to a level comparable with VGG. These results show that SWAG is generally beneficial for stylization algorithms.

4.2. Quantitative evaluation

Stylization quality is difficult to evaluate quantitatively, since it is subjective. While some works only show synthesized images [7, 3, 14, 41, 2, 8], the user study has been mainly used for quantitative evaluation in this paper, where humans choose a preferred image among a set of candidates. This is consistent with the subjective nature of stylization.

We present results of a user study on Amazon Mechanical Turk, comparing pairs of images synthesized by networks with and without SWAG. Each comparison was assigned to 30 turkers. Each turker was asked to choose the image that more closely resembled a given style image. 10 randomly selected stylization comparisons were performed by turker. The results of the experiment are summarized in Table 1, which shows several interesting findings. First, for

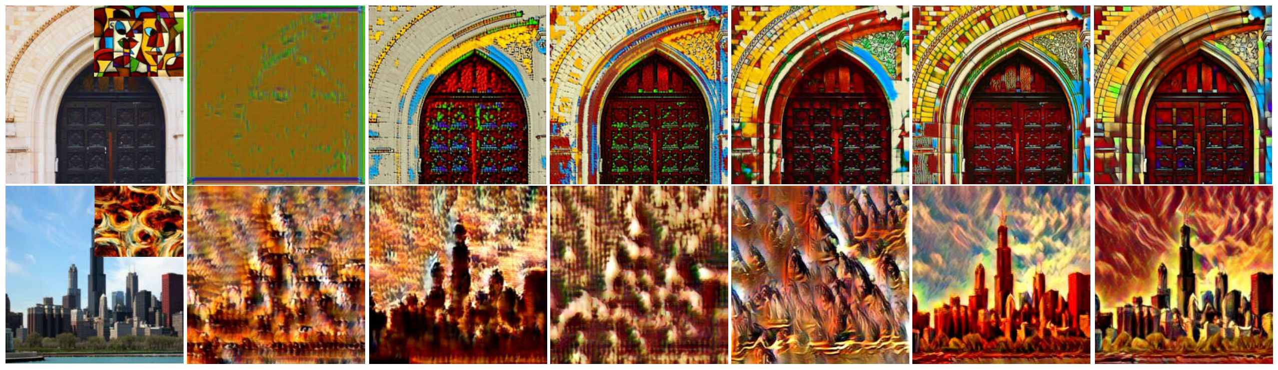


Figure 7: Comparison of neural style transfer performance between standard and SWAG (denoted with *) implementations of different stylization algorithms. Top: algorithm of [22]; Bottom: algorithm of [24]. The results of the VGG model are also shown for comparison.

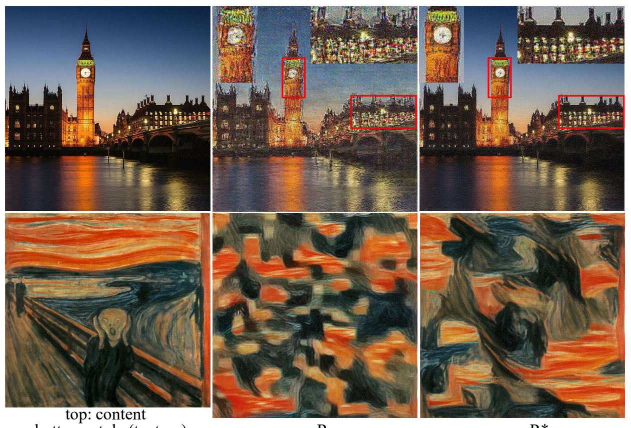


Figure 8: Top: reconstruction of the top-left content image; Bottom: texture synthesis of the bottom-left image.

both pre-trained and random networks, models with SWAG always earned more preferences than models without. Second, all models with SWAG significantly outperformed the standard VGG implementation. All these results suggest that SWAG eliminated the dependence of stylization on the VGG architecture.

4.3. Ablation study

For stylization, the target image is usually initialized with the content image [7, 13, 3], i.e. the optimization of (2) uses the content image x_0^c as initial condition. This reduces the difficulty in transferring content information to the target image. In fact, it can make the content loss negligible [27], to the point where the latter can be removed from the optimization altogether [3]. This creates difficulties to ablate the effectiveness of SWAG on the content and style loss individually.

For this, we leveraged two alternative tasks: image reconstruction and texture synthesis, using a random initial image. Following [7, 6], we matched conv3_4 features from the content images for image reconstruction, and multilayer features from the texture image for texture synthesis. As shown in Figure 8, SWAG produces better reconstructed images (average PSNR over 10 randomly selected content images of p-R/p-R*: $27.9(\pm 0.1)/28.3(\pm 0.1)$) and synthesizes textures with a larger diversity of patterns, including small-scale and large-scale style patch patterns. This suggests that 1) residual connections hurt both losses fail; 2) SWAG can match styles at deeper layers, which capture larger-scale features, to improve reconstruction and texture synthesis quality.

5. Conclusion

We have studied the lack of robustness of stylization algorithms for non-VGG architectures. This showed that a significant factor is the use of residual connections, which decreases the entropy of deeper layer activations. A simple solution was proposed by adding activation smoothing to the loss functions used for stylization, using a softmax function. This was denoted as SWAG, and shown to be effective for various architectures, forms of pre-training (random vs. ImageNet), and stylization algorithms. It was shown that, with the addition of SWAG, the lightweight non-VGG becomes a viable alternative to VGG in future stylization work.

Acknowledgement

This work was partially funded by NSF awards IIS-1924937, IIS-2041009, a gift from Amazon, a gift from Qualcomm, and NVIDIA GPU donations. We also acknowledge and thank the use of the Nautilus platform for some of the experiments discussed above.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31*, pages 9505–9515. 2018.

[2] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1897–1906, 2017.

[3] Len Du. How much deep learning does neural style transfer really need? an ablation study. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3150–3159, 2020.

[4] Logan Engstrom, Andrew Ilyas, Aleksander Madry, Shibani Santurkar, Brandon Tran, and Dimitris Tsipras. A discussion of 'adversarial examples are not bugs, they are features': Discussion and author responses. *Distill*, 2019. https://distill.pub/2019/advex-bugs-discussion/original-authors.

[5] Adam Gaier and David Ha. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, pages 5365–5379, 2019.

[6] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pages 262–270, 2015.

[7] 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*, pages 2414–2423, 2016.

[8] 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*, pages 3985–3993, 2017.

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

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Robert M Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

[12] gwern. https://www.reddit.com/r/MachineLearning/comments/7rrrk3/d_eat_your_vggtables_or_why_does_neural_style/.

[13] Kun He, Yan Wang, and John E. Hopcroft. A powerful generative model using random weights for the deep image representation. In *NIPS*, pages 631–639, 2016.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[17] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. IEEE, 2004.

[18] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[19] 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*, pages 1501–1510, 2017.

[20] Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004.

[21] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 2019.

[22] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[23] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10051–10060, 2019.

[24] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[25] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 453–468, 2018.

[26] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 768–783, 2018.

[27] Reiichiro Nakano. https://distill.pub/2019/advex-bugs-discussion/response-4/.

[28] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[29] Y-H Pao and Yoshiyasu Takefuji. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5):76–79, 1992.

[30] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5880–5888, 2019.

[31] PyTorch. https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py.

[32] PyTorch. https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py.

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

[34] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[37] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016.

[38] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[39] Dianhui Wang and Ming Li. Robust stochastic configuration networks with kernel density estimation for uncertain data regression. *Information Sciences*, 412:210–222, 2017.

[40] Dianhui Wang and Ming Li. Stochastic configuration networks: Fundamentals and algorithms. *IEEE transactions on cybernetics*, 47(10):3466–3479, 2017.

[41] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5239–5247, 2017.

[42] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[43] Yuan Yao, Jianqiang Ren, Xuansong Xie, Weidong Liu, Yong-Jin Liu, and Jun Wang. Attention-aware multi-stroke style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1467–1475, 2019.

[44] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016.

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

[46] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 506–511. IEEE, 2017.