Rethinking and Improving the Robustness of Image Style Transfer Supplementary Material

Pei Wang
UC, San Diego
pew062@ucsd.edu

Yijun Li
Adobe Research
yijli@adobe.com

Nuno Vasconcelos
UC, San Diego
nuno@ucsd.edu

1. Ablation study

To validate that the poor performance of the ResNet is mainly due to the residual connection, we perform the ablation study over other network components and their combinations: (1) **VGG-bn**: VGG with batch normalization (bn) where the *bn* layer is not used in the original VGG but is introduced in ResNet. (2) **VGG-c7**: Replacing the first layer conv3×3 in VGG with that conv7×7 in ResNet. There are three kinds of conv1×1 kernels in ResNet, conv1×1 that increases the channel number, conv1×1 that decreases the channel number, and conv1×1 that maintains the channel number, i.e. conv2_1. We denote them by 'Ic1', 'Dc1', 'c1' respectively. This introduces (3) **VGG-Ic1**: VGG with conv1x1 that increases the channel number, (4) **VGG-Dc1**: VGG with conv1x1 that decreases the channel number, and (5) **VGG-c1**: VGG with conv1x1 that maintains the channel number. We also investigate the influence of the combination of many factors. (6) **VGG-Ic1-Dc1**, (7) **VGG-c7-Ic1**, (8) **VGG-c7-Dc1**, (9) **VGG-c7-Ic1-Dc1**. The impact of other factors like the number of channels per layer or network depth have been discussed and shown to be less important in [1]. Figure 1 presents two stylization examples obtained by models (1)∼(9). It can be seen that the influence brought by those network components are much smaller than that by the residual connection (as shown in Figure 2 of the paper). Therefore, we conclude that the residual connection in ResNet is the root cause which results in the poor stylization performance.

2. More results

2.1. Style loss

We also compute the style loss for images synthesized by a pre-specified model (usually a pre-trained VGG) [4, 3]. Specifically, images are stylized using the different network architectures, with and without SWAG. Stylized images are then fed into a VGG pre-trained on ImageNet and the loss of (4) is computed. This measures the similarity between synthesized and style image, ignoring content information. Note that this metric has a certain bias towards the VGG.

Table 1 shows the results of the style loss comparison, based on activations from five layers (conv1_1, conv2_1, conv3_1, conv4_1, conv5_1) of the pre-trained VGG[1]. We randomly select 10 content images and 10 style images from [2, 1, 5], and compute the averaged style loss over all 100 content-style combinations. A few conclusions can be drawn. First, for both pre-trained and random models, SWAG improves the performance of each non-VGG network. Second, for random networks, the gains of SWAG are of two orders of magnitude. Third, for both random and pre-trained models, the ResNet with SWAG even outperforms the standard VGG model. Fourth, SWAG even slightly improves the performances of the VGG model, which does not suffer from a noticeable peaky large activation problem. Finally, SWAG significantly reduces the performance gap between random and pre-trained models.

2.2. Visual comparison

More comparisons of images synthesized by different networks corresponding to the 12 comparison pairs in Table 2 of the paper are shown in Figure 3∼14.

3. Ablation study on T in Eq. (11) of the paper

In general, T should 1) increase as H in Eq. (5) decreases and 2) be ≥ 1 (note that $H \in [0, 1]$). This is to guarantee SWAG can always increase the entropy and be more powerful for ultra-peaky activations. The ablation study experiment on temperature (T) in Eq. (11) is conducted. We found that mean entropy increases with T but saturates quickly for $T > 1$. The mean style loss across different architectures also decreases until saturation for $T > 1$. It indicates $T = 1$ is sufficient and larger T will not result in further improvement.

[1] The models, pre-trained on ImageNet, are those provided by PyTorch (https://pytorch.org/docs/stable/torchvision/models.html).

	Pre-trained				Random			
Arch.	ResNet	Inception	WRN	VGG	ResNet	Inception	WRN	VGG
Standard	3.8(9.1)e4	6.3(3.4)e4	3.8(4.5)e4	2.5(4.6)e4	1.8(1.3)e6	1.3(9.7)e6	1.3(7.7)e6	3.9(7.1)e4
SWAG	**2.3(4.3)e4**	**4.0(1.9)e4**	**2.6(6.3)e4**	**2.4(4.6)e4**	**3.4(1.3)e4**	**7.9(6.6)e4**	**6.3(3.6)e4**	**3.7(7.0)e4**

Table 1: Style loss comparison of different architectures (mean(std)).

4. Implementation details

The content and style image are subject to the standard normalization. Specifically, all images are first converted to $[0.0, 1.0]$ from $[0, 255]$ and then normalized by subtracting the mean ($[0.485, 0.456, 0.406]$) and divided by the standard deviation $(0.229, 0.224, 0.225])$ of each RGB color channel. All results are of size 512×512. All pre-trained models used in the paper are from PyTorch[2]. We follow the setup of [2] for the VGG model, i.e., using features at the conv1_1, conv2_1, conv3_1, conv4_1, conv5_1 layer for style loss of Eq. (4) and Eq. (13), and the conv4_2 layer for content loss of Eq. (3) and Eq. (12). We set $\alpha = 1$ and $\beta = 4e10$ in Eq. (2). For ResNet, we follow the setting of [6], but, in addition to the features at the conv2_3, conv3_4, conv4_6, conv5_3 layer, we also use the conv1_2 layer in Eq. (4) and Eq. (13). This is for fair comparison with the VGG implementation of [2], which uses five layers at different scales. The conv4_6 layer is used for the computation of content loss. We set $\alpha = 1$ and $\beta = 1e17$. The same setting is for WRN. On Inception v3, the conv2d_1a, conv2d_3b, mixed_5b, mixed_6a, mixed_7a leayers are used, again for consistency with the VGG model. The mixed_5b layer is for content loss computation, and we set $\alpha = 1$, $\beta = 4e10$.

References

[1] 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.

[2] 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.

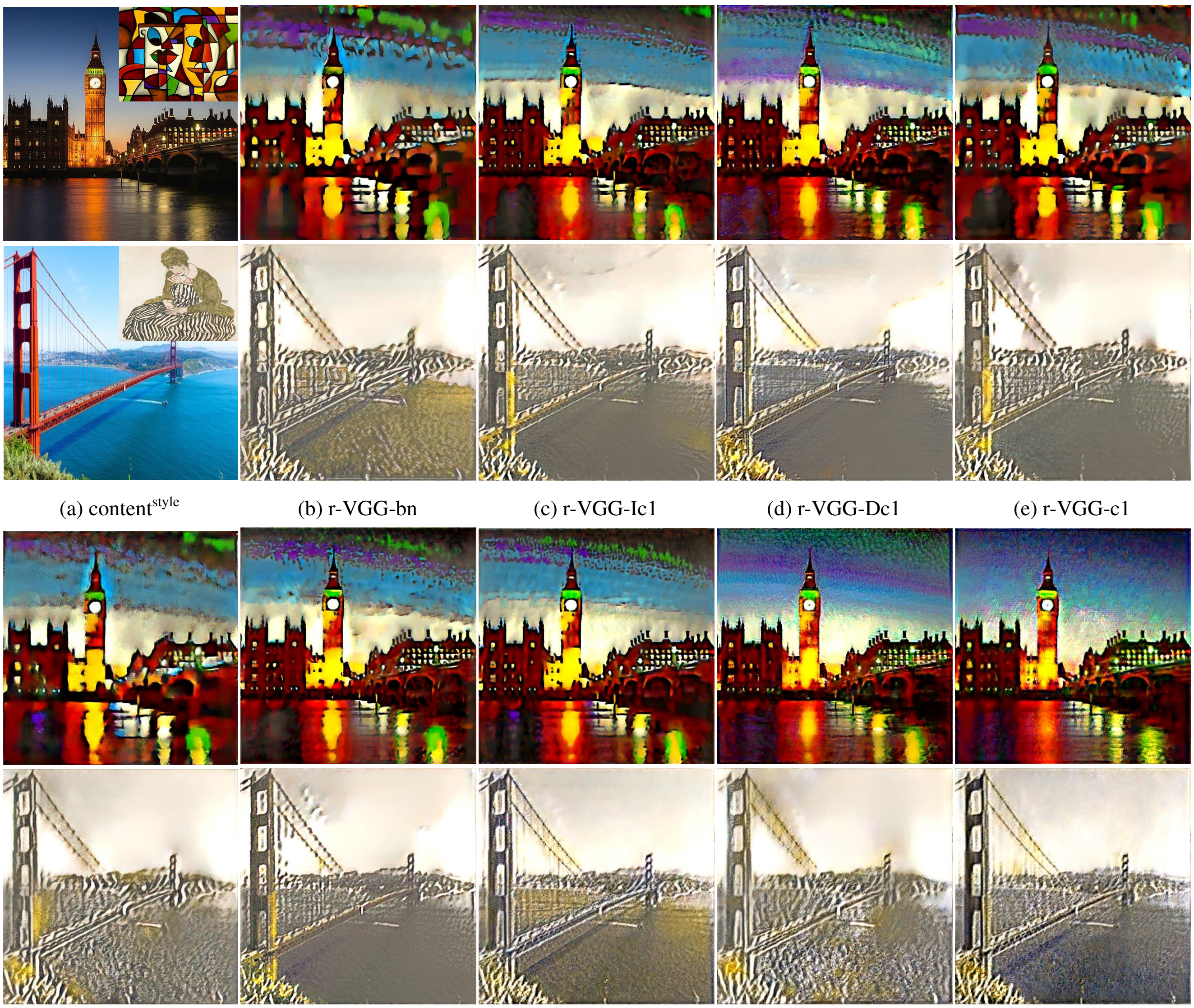
[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[4] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in neural information processing systems*, pages 386–396, 2017.

[5] 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.

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

[2] https://pytorch.org/docs/stable/torchvision/models.html



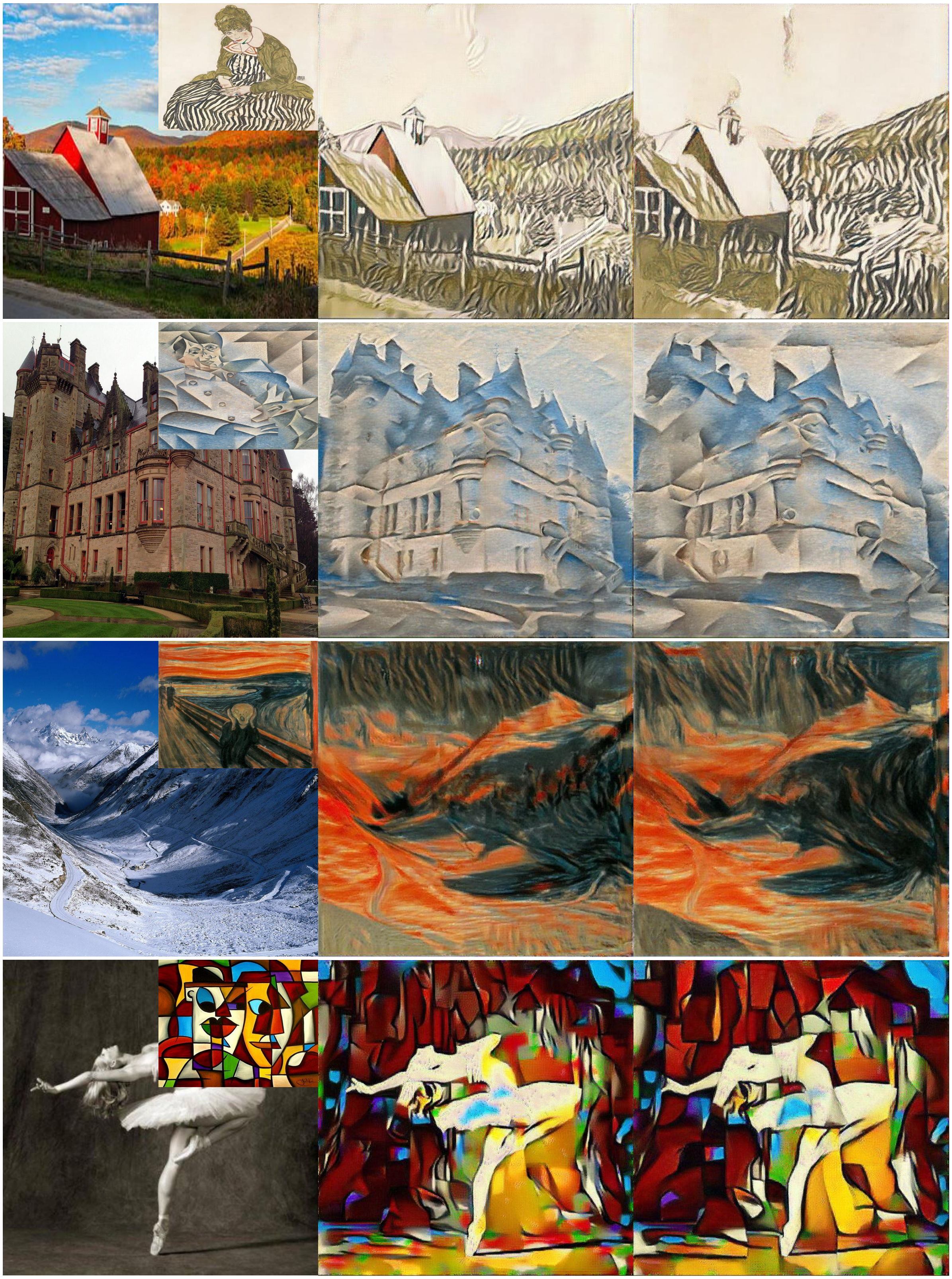
(a) content$^{\text{style}}$ (b) r-VGG-bn (c) r-VGG-Ic1 (d) r-VGG-Dc1 (e) r-VGG-c1

(f) r-VGG-c7 (g) r-VGG-Ic1-Dc1 (h) r-VGG-c7-Ic1 (i) r-VGG-c7-Dc1 (j) r-VGG-c7-Ic1-Dc1

Figure 1: Results comparisons of different architectures. ('r-' represent randomly initialized. Detailed comparison need to zoom in on pictures. '*' denotes our proposal.)

(a) content$^{\text{style}}$ (b) p-R (c) p-R*

Figure 2: Comparison of neural style transfer performance between p-R and p-R SWAG (denoted with *) models



(a) content$^{\text{style}}$ (b) p-I (c) p-I*

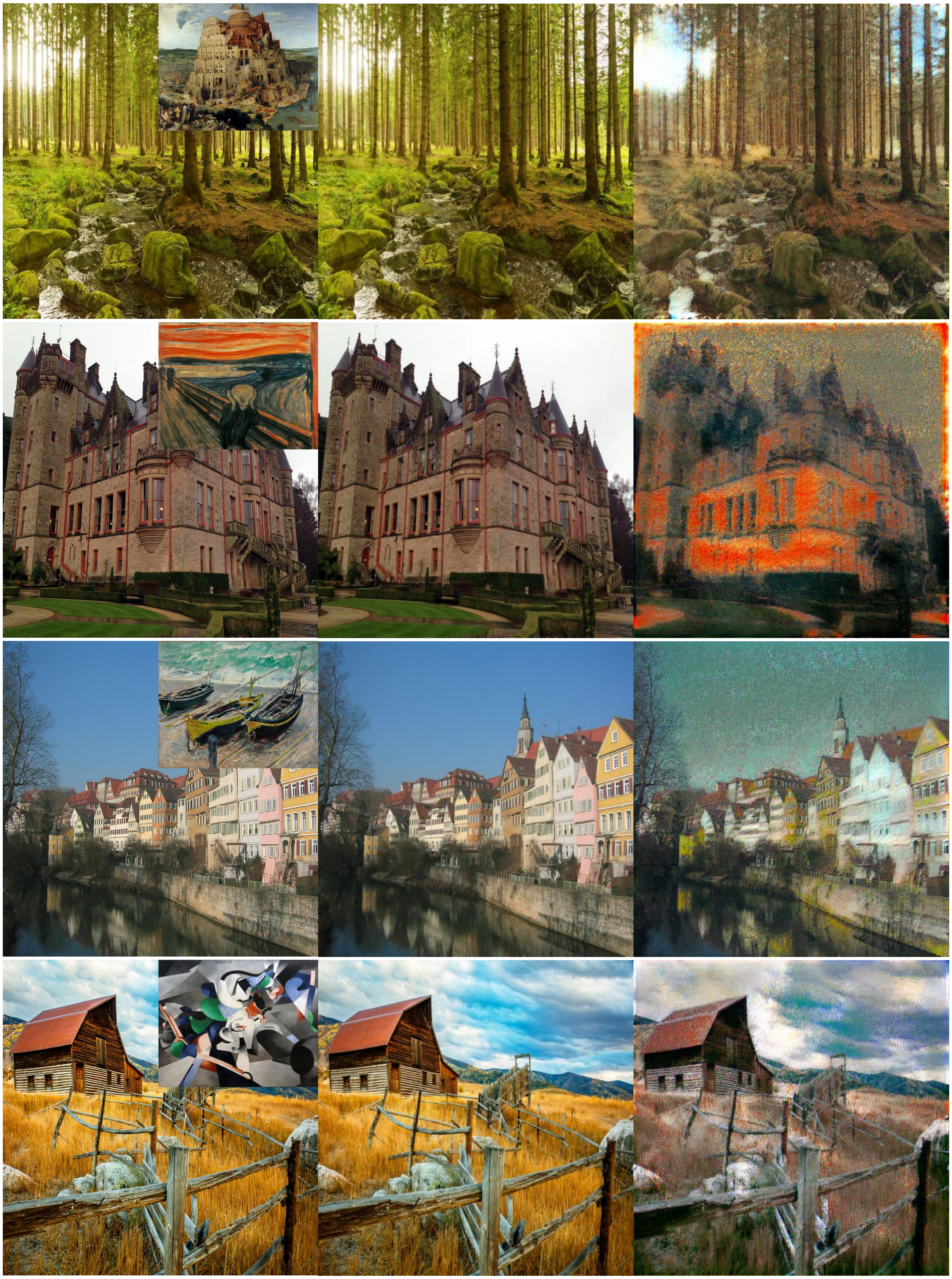
Figure 3: Comparison of neural style transfer performance between p-I and p-I SWAG (denoted with *) models

(a) content$^{\text{style}}$ (b) p-W (c) p-W*

Figure 4: Comparison of neural style transfer performance between p-W and p-W SWAG (denoted with *) models

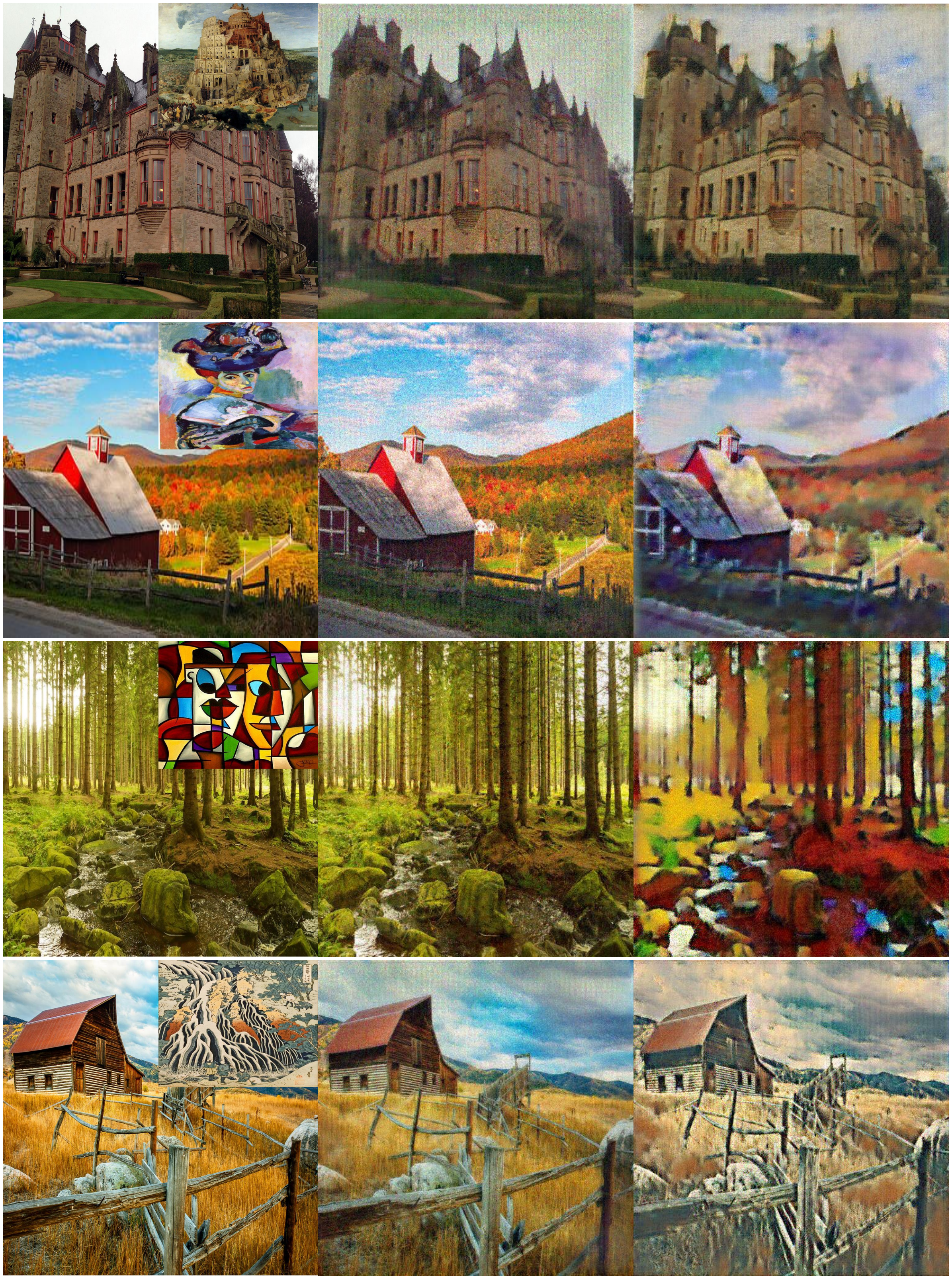
(a) content$^{\text{style}}$ (b) r-R (c) r-R*

Figure 5: Comparison of neural style transfer performance between r-R and r-R SWAG (denoted with *) models



(a) content$^{\text{style}}$ (b) r-I (c) r-I*

Figure 6: Comparison of neural style transfer performance between r-I and r-I SWAG (denoted with *) models



(a) content$^{\text{style}}$ (b) r-W (c) r-W*

Figure 7: Comparison of neural style transfer performance between r-W and r-W SWAG (denoted with *) models

(a) content$^{\text{style}}$ (b) p-V (c) p-R*

Figure 8: Comparison of neural style transfer performance between p-V and p-R SWAG (denoted with *) models

(a) content$^{\text{style}}$ (b) p-V (c) p-I*

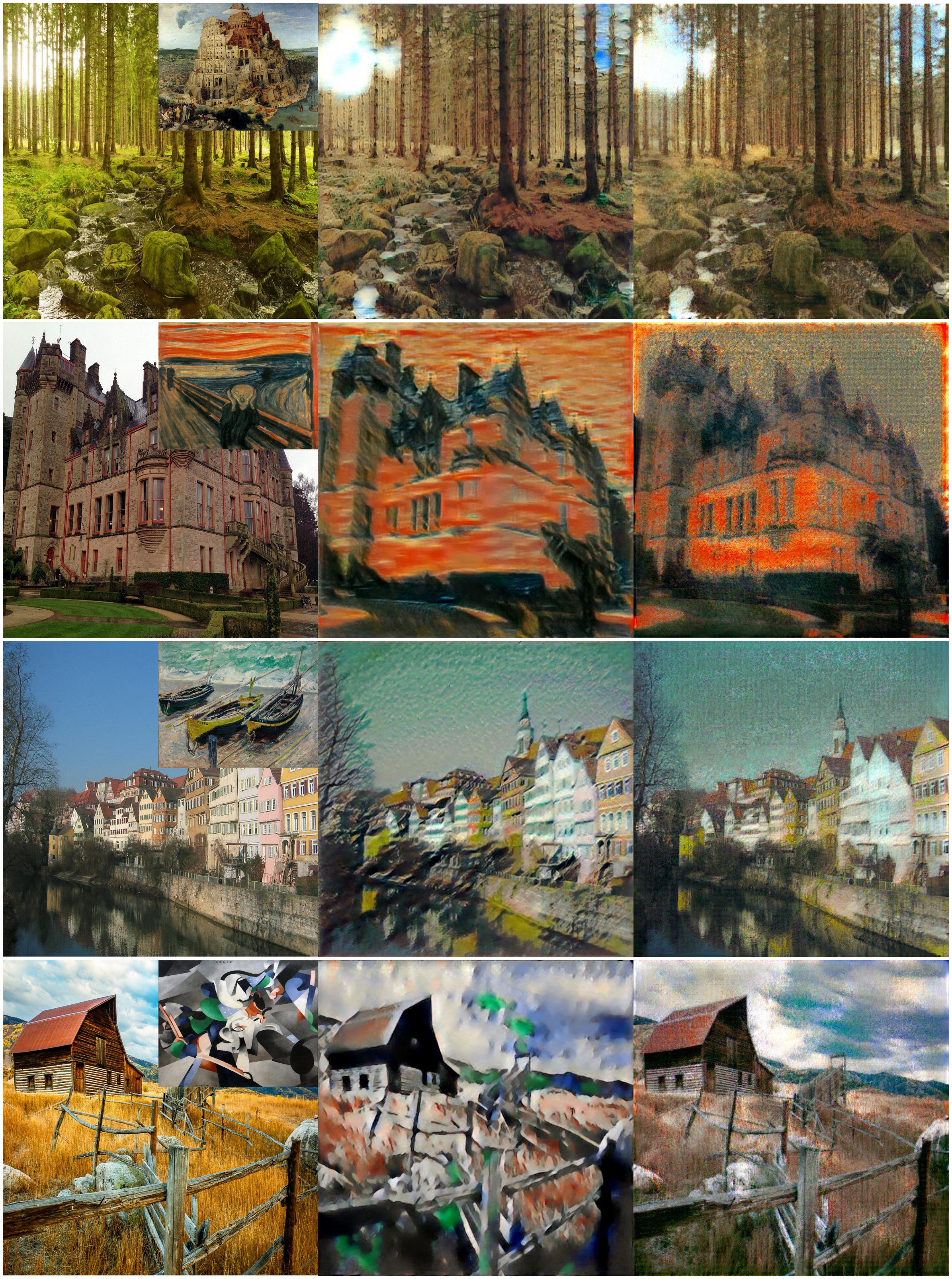
Figure 9: Comparison of neural style transfer performance between p-V and p-I SWAG (denoted with *) models

(a) content$^{\text{style}}$ (b) p-V (c) p-W*

Figure 10: Comparison of neural style transfer performance between p-V and p-W SWAG (denoted with *) models

(a) content$^{\text{style}}$ (b) r-V (c) r-R*

Figure 11: Comparison of neural style transfer performance between r-V and r-R SWAG (denoted with *) models



(a) content$^{\text{style}}$ (b) r-V (c) r-I*

Figure 12: Comparison of neural style transfer performance between r-V and r-I SWAG (denoted with *) models

(a) content$^{\text{style}}$ (b) r-V (c) r-W*

Figure 13: Comparison of neural style transfer performance between r-V and r-W SWAG (denoted with *) models