Self-Contained Stylization via Steganography for Reverse and Serial Style Transfer
Supplementary Materials

Hung-Yu Chen[1*†] I-Sheng Fang[1*†] Chia-Ming Cheng[2] Wei-Chen Chiu[1]
[1]National Chiao Tung University, Taiwan [2]MediaTek Inc., Taiwan
chen3381@purdue.edu nf0126@gmail.com walon@cs.nctu.edu.tw

	Reverse Style Transfer			Serial Style Transfer		
	L2	SSIM	LPIPS	L2	SSIM	LPIPS
Gatys *et al*. [1]	4.4331	0.2033	0.3684	7.5239	0.0472	0.4317
AdaIN [2]	0.0368	0.3818	0.4614	0.0213	0.5477	0.3637
WCT [5]	0.0597	0.3042	0.5534	0.0568	0.3318	0.5048
Extended baseline (AdaIN w/ cycle consistency)	0.0502	0.2931	0.5809	0.0273	0.4140	0.4314
Our two-stage	**0.0187**	0.4796	**0.3323**	0.0148	0.7143	0.2437
Our end-to-end	0.0193	**0.5945**	0.3802	**0.0104**	**0.8523**	**0.1487**

Table 1: The average L2 distance, structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS [11]) between the results produced by different models and their corresponding expectations. Regarding extended baseline (AdaIN with cycle consistency), please refer to the Section 3 in this supplement for more detailed description.

1. More Results

1.1. Regular, Reverse and Serial Style Transfer

1.1.1 Qualitative Evaluation

First, we provide three more sets of results in Figure 6, demonstrating the differences between the results of regular, reverse, and serial style transfer performed by different methods. Moreover, we provide in the Figure 7 more qualitative results, based on diverse sets of content and style images from MS-COCO [6] and WikiArt [7] datasets respectively. In which these results show that our proposed methods are working fine to perform regular, reverse, and serial style transfer on various images.

1.1.2 Quantitative Evaluation

As mentioned in the Section 4.3 of our main manuscript, here we provide more quantitative evaluations in Table 1, based on L2 distance, structural similarity (SSIM), and LPIPS [11]. Both our methods in the tasks of reverse and serial stylization perform better than the baselines in terms of different metrics. Please note that although Gatys *et al*. [1] can obtain also good performance for the task of reverse style transfer in terms of LPIPS metric (based on the similarity in semantic feature representation), it needs to use the original image as the style reference to perform the reverse style transfer, which is actually impractical.

1.2. Serial Style Transfer for Multiple Times

To further demonstrate the ability of preserving the content information of our models, we perform serial style transfer on an image for multiple times. There are three sets of results in Figure 8 for comparing the results generated by different methods. It can be seen that Gatys *et al*. [1] and AdaIN [2] fail to distinguish the contour of the content objects from the edges caused by the stylization, thus the results deviate further from the original content when serial style transfer is applied. As for our two-stage and end-to-end model, the content is still nicely preserved even in the final results after a series of style transfer. It clearly indicates that our models provide better solutions to the issue of serial style transfer.

[†]Hung-Yu Chen and I-Sheng Fang are now with Purdue University and National Cheng Chi University respectively.
[*]Both authors contribute equally.

2. More Ablation Study

2.1. Two-Stage Model

2.1.1 Quantitative Evaluation of Identity Mapping

We evaluate the effect of having *identity mapping* (Section 3.1.1 in the main paper) in our proposed two-stage model based on the average L2 distance, structural similarity (SSIM), and learned perceptual image patch similarity (LPIPS [11]).The results are provided in Table 2. It clearly shows that adding identity mapping in the training of AdaIN decoder D_{AdaIN} enhances the performance of reverse and serial style transfer.

2.1.2 Training with and without Adversarial Learning

As mentioned in Section 3.1.3 of the main paper, the architectures of our message encoder E_{msg} and decoder D_{msg} in the steganography stage are the same as the ones used in HiDDeN [12], while HiDDeN [12] additionally utilizes adversarial learning to improve the performance of encoding. Here we experiment to train our steganography stage with adversarial learning as well, where two losses $\{\mathcal{L}_{discriminator}, \mathcal{L}_{generator}\}$ are added to our object function as follows.

$$\begin{aligned}\mathcal{L}_{\text{discriminator}} =& \mathbb{E}\left[\left(Dis\left(I_t\right)-\mathbb{E}\left(Dis\left(I_e\right)\right)-1\right)^2\right]+ \\ & \mathbb{E}\left[\left(Dis\left(I_e\right)-\mathbb{E}\left(Dis\left(I_t\right)\right)+1\right)^2\right]\end{aligned} \tag{1}$$

$$\begin{aligned}\mathcal{L}_{\text{generator}} =& \mathbb{E}\left[\left(Dis\left(I_e\right)-\mathbb{E}\left(Dis\left(I_t\right)\right)-1\right)^2\right]+ \\ & \mathbb{E}\left[\left(Dis\left(I_t\right)-\mathbb{E}\left(Dis\left(I_e\right)\right)+1\right)^2\right]\end{aligned} \tag{2}$$

where Dis denotes the discriminator used in adversarial learning. Here in our experiment, the architecture of the discriminator is identical to the one used in HiDDeN [12], and we adopt the optimization procedure proposed in [3] for adversarial learning.

Afterward, we perform qualitative and quantitative evaluations on the results, as shown in Figure 9 and Table 2 respectively. We observe that adding adversarial learning does not enhance the quantitative performance. Similarly, we remark that the results are visually similar according to the qualitative examples as shown in Figure 9.

2.1.3 Serial Style Transfer with De-Stylized Image

As mentioned in the main paper (cf. Section 3.1.3), we stylize the image generated from the decoded message to perform serial style transfer. However, we can also resolve the issue of serial style transfer in a different way. Figure 1 shows that we can implement serial style transfer by stylizing the de-stylized image from the result of reserve style transfer. For comparison, we qualitatively evaluate the results generated with the de-stylized image and the decoded message. Figure 2 shows that the results of these two methods are nearly identical. Since the model using decoded message (as in the main paper) is simpler than the other, we choose to adopt it in our proposed method. The quantitative evaluation is also provided in the Table 3, based on the metrics of average L2 distance, structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS [11]). We can see that our model of using decoded message performs better than the one of using de-stylized image, in which this observation thus verifies our design choice.

2.2. End-to-End Model

2.2.1 Quantitative evaluation of using E_{inv} to recover v_t from I_{st} in end-to-end model

We evaluate the effect of having E_{inv} (please refer to the Section 4.4 in the main paper) in our proposed end-to-end model based on the metrics of average L2 distance, structural similarity (SSIM), and learned perceptual image patch similarity (LPIPS [11]). The results are provided in Table 4. It clearly shows that using E_{inv} instead of E_{VGG} enhances the performance of reverse and serial style transfer, which thus verifies our design choice of having E_{inv} in our end-to-end model.

2.2.2 Decoding with Plain Image Decoder or AdaIN Decoder for Reverse Style Transfer

It is mentioned in Section 3.2 of the main paper that the training of a plain image decoder D_{plain} in the end-to-end model shares the same idea with the identity mapping, which is used in learning AdaIN decoder D_{AdaIN} of the two-stage model. However, although they both are trained to reconstruct the image I_c with its own feature $E_{VGG}(I_c)$, these two decoders accentuate different parts of the given feature during the reconstruction. The AdaIN decoder is trained to decode the results of regular and reverse style transfer simultaneously, but with an emphasis on the stylization, considering that identity mapping is only activated occasionally during the training. It is optimized toward both content and style features based on the perceptual loss in order to evaluate the effect of the stylization. As for the plain image decoder, it is solely trained for reconstructing the image with the given content feature, and optimized with the L2 distance to the original image. Such distinction brings differences to the images decoded from the same feature by these two decoders, as shown in Figure 3 and Table 5.

Comparing to the results generated by the plain image decoder, the images decoded by the AdaIN decoder have sharper edges and more fine-grained details, but sometimes

	Reverse Style Transfer			Serial Style Transfer		
	L2	SSIM	LPIPS	L2	SSIM	LPIPS
Our two-stage (w/ identity mapping)	**0.0187**	**0.4796**	**0.3323**	**0.0148**	**0.7143**	**0.2437**
Our two-stage (w/o identity mapping)	0.0226	0.4596	0.3637	0.0152	0.6990	0.2560
Our two-stage (w/ adversarial learning)	0.0271	0.4292	0.3878	0.0168	0.5946	0.3236

Table 2: The average L2 distance, structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS [11]) between the expected results and the ones which are obtained by our two-stage model and its variants of having identity mapping AdaIN decoder or adversarial learning.

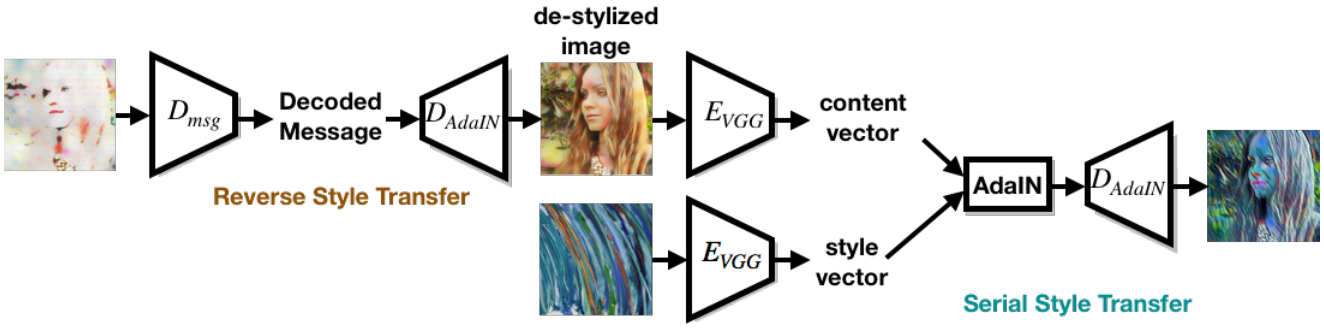

Figure 1: Illustrations of how to apply our two-stage model in the task of serial style transfer with de-stylized image.

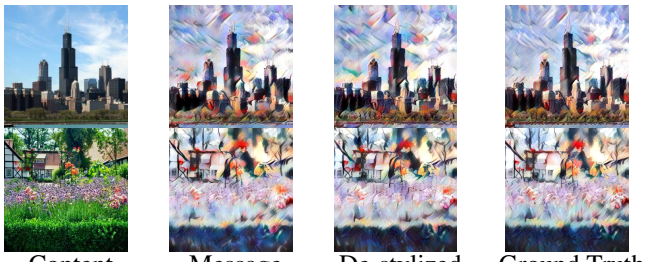

Figure 2: Comparison between the results of serial style transfer generated with decoded messages and the de-stylized images.

	L2	SSIM	LPIPS
w/ de-stylized image	0.02558	0.48694	0.40362
w/ decoded message	**0.01480**	**0.71430**	**0.24370**

Table 3: The average L2 distance, structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS [11]) between expected results and the ones which are produced by our two-stage model with performing serial style transfer w/ de-stylized image or w/ decoded message.

the straight lines are distorted and the contours of the objects are not in the same place as they are in the original image, harming the consistency of the overall content structure. Examples can be found in Figure 3, especially on the boundaries of the buildings. The quantitative evaluation provided in Table 5 also shows that using plain image decoder could provide better performance than adopting AdaIN decoder in terms of different metrics.The benefit of introducing the plain image decoder for reverse style transfer of end-to-end model is therefore verified.

	Reverse Style Transfer			Serial Style Transfer		
	L2	SSIM	LPIPS	L2	SSIM	LPIPS
E_{inv}	**0.0193**	**0.5945**	**0.3802**	**0.0104**	**0.8523**	**0.1487**
E_{VGG}	0.0241	0.5190	0.4727	0.0149	0.7525	0.2362

Table 4: The average L2 distance, structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS [11]) between expected results and the ones which are obtained by our end-to-end model of using E_{inv} or E_{VGG}.

	L2	SSIM	LPIPS
Plain image decoder	**0.0193**	**0.5945**	**0.3802**
AdaIN decoder	0.0349	0.4261	0.4141

Table 5: The average L2 distance, structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS [11]) between expected results and the ones which are obtained by our end-to-end model of using plain image decoder D_{plain} or VGG decoder for reverse style transfer.

Figure 3: Comparison between the images decoded from the same feature vectors by (1) the AdaIN decoder D_{AdaIN} in two-stage model and (2) the plain image decoder D_{plain} in end-to-end model. In set (a), the features given to the decoders are the content features extracted from the images in the top row by pre-trained VGG19 [9], which is $E_{VGG}(I_c)$. As for set (b), the given feature vectors are the ones derived from the stylized images (the second row) with the end-to-end model, i.e $\hat{v}_c$.

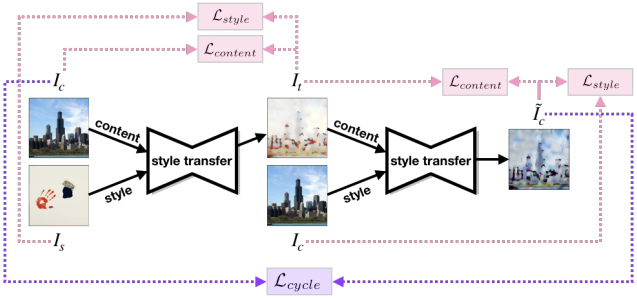

Figure 4: Illustration of the framework and training objectives for the extended baseline for reverse style transfer, which is based on a typical style transfer approach (i.e. AdaIN) and the cycle consistency objective $\mathcal{L}_{cycle}$.

3. Extended Baseline

Typical Style Transfer Approach Extended with Cycle Consistency. As mentioned in the Section 4.2.2. and Figure.2 of our main manuscript, the naïve baselines built upon the typical style transfer approaches (i.e. Gatys *et al*. [1] and AdaIN [2]) are not able to resolve the task of reverse style transfer, which is analogous to perform de-stylization on a stylized image back to its original photo. For further exploration of the capacity of the naïve baselines for reverse style transfer, here we provide another extended baseline for comparison.

The framework of this extended baseline is illustrated in Figure 4, where the AdaIN style transfer component is composed of a pre-trained VGG19 encoder and a decoder D_{AdaIN}. First, given a content photo I_c and a style image I_s, D_{AdaIN} is trained for making the stylized image I_t to have similar content and style as I_c and I_s respectively, where the content loss $\mathcal{L}_{content}(I_c, I_t)$ and the style loss $\mathcal{L}_{style}(I_s, I_t)$ are used (please refer to Equation.3 and 4 in the main manuscript). Second, I_t and I_c are taken as the source of content and style respectively to produce a de-stylized output $\tilde{I}_c$, where D_{AdaIN} is now trained to minimize $\mathcal{L}_{content}(I_t, \tilde{I}_c)$ and $\mathcal{L}_{style}(I_c, \tilde{I}_c)$. Last, the cycle consistency objective $\mathcal{L}_{cycle} = \left\| \tilde{I}_c - I_c \right\|$ is introduced for updating D_{AdaIN} in order to encourage $\tilde{I}_c$ and I_c to be identical, i.e. reverse style transfer or de-stylization.

As shown in Figure 5, even if the extended baseline is trained with the cycle consistency objective, it is still not able to resolve the task of reverse style transfer. The quantitative results provided in the Table 1 also indicate the inferior performance of this extended baseline. These results also demonstrate that the content information is lost during the procedure of the typical style transfer and can not be easily recovered, which further emphasize the contribution and the novelty of our proposed models based on the steganography idea. Please also note that the extended baseline needs to take the original content photo I_c as the source of style for performing de-stylization, while our proposed models are self-contained without any additional input.

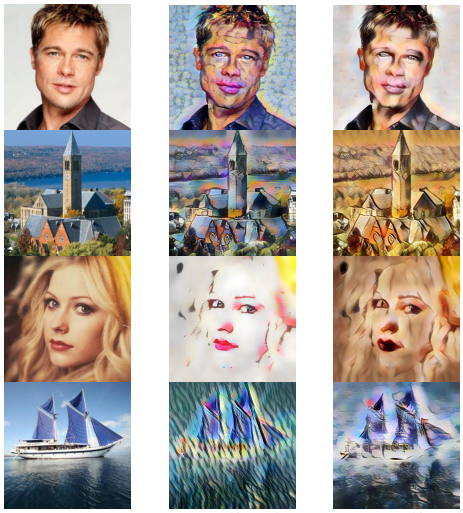

Figure 5: The results produced by the extended baseline of reverse style transfer which is trained with cycle consistency loss.

4. Replacing AdaIN with Other Stylization Methods for Two-Stage Model

To verify the adaptability of our two-stage model, we replace AdaIN, which is originally adopted in the style transfer stage, with WCT [5] and instance normalization [10], and compare their results to the ones of the original implementation. Denote the selected style transfer method (e.g.WCT [5]) by f, we can get the stylized image $I_t = f(I_c, I_s)$ in the style transfer stage. Meanwhile, the content feature $v_c = D_{VGG}(I_c)$ remains to be `relu4_1` extracted by VGG19 from the content image, and is encrypted into the stylized image I_t by $I_e = E_{message}(I_t, v_c)$ later in the steganography stage.

As the content feature encrypted in I_e is retrievable by $v_c' = D_{msg}(I_e)$ just like the original implementation, reverse style transfer can be intuitively done by $I_c' = D_{AdaIN}(v_c')$. When it comes to serial style transfer, we simply need to further stylize the reconstructed image I_c' with another style I_s' by computing $I_t' = f(I_c', I_s')$. As the results shown in Figure 10, our two-stage model still performs well when adapted to WCT [5] and instance normalization [10]. Their results are closer to the corresponding expectations, have less artifacts, and preserve more content structure and detail than the ones of naïve approaches, as the original implementation with AdaIN does. Please note that all these replacements are done without the need of any additional training. The encoders/decoders trained with the original implementation can be directly inherited without further modification.

5. Limitations

The main limitation of our proposed methods, which stem from the idea of steganography, is being unavoidable to have errors in the decrypted message through the procedure of encryption and decryption. In our two-stage model, since it needs to hide the whole content feature of the original image into its stylized output, the errors in the decrypted message would cause inconsistent color patches in the results of reverse style transfer. For instance in Figure 6, as can be seen from the reverse style transfer results of the sailboat image produced by our two-stage model, there are different color patches in the sky which ideally should be homogeneous. While for our end-to-end model, it aims to encrypt the statistic (i.e., mean and variance) of the content feature of the original image into the stylized output, the errors in the decrypted message now lead to the color shift issue when performing reverse style transfer, which is also observable in the Figure 6. We would seek for other network architecture designs or training techniques (e.g. add random noise during network training, as used in [12]) in order to have better robustness of our models against the errors caused by encryption and decryption.

6. Implementation Details

Here we provide some implementation details of our two-stage and end-to-end model. We use PyTorch [8] 0.4.1 as our environment of developing deep learning framework. All the source code and trained models will be publicly available for reproductivity once the paper is accepted.

6.1. Two-Stage Model

Architectures. D_{AdaIN} has the same architecture as the decoder used in the original implementation of AdaIN [2]. It consists of 3 nearest up-sampling layers, 9 convolutional layers with the kernels of size 3×3, and ReLU activations after each conv-layer except the last one. Our E_{msg} and D_{msg} also inherit the architecture of the encoder and decoder in the implementation of HiDDeN [12]. E_{msg} has 4 convolution blocks. Each convolution block includes a convolutional layer with a 3×3 kernel, a batch normalization layer and a ReLU activation (except the last block). The message to encrypt is first reshaped, then concatenated to the output of the first convolution block. D_{msg} has 8 convolution blocks. Each convolution block includes a convolutional layer with a 3×3 kernel, a batch normalization layer and a ReLU activation (except the first and the last block). The dimension of the decrypted message is recovered by adaptive average pooling and reshaping after the final block.

Hyperparameters. The learning rate used in our model training is 10^{-4}. We adopt Adam optimizer [4] with hyper-parameters $\{\beta_1 = 0.5, \beta_2 = 0.999\}$. The batch-

size is set to 8. The λ parameters for the objective function $\mathcal{L}_{steganography}$ in the steganography stage are set as $\{\lambda_{img} = 2000, \lambda_{msg} = 10^{-}5\}$.

6.2. End-to-End Model

Architectures. $D_{encrypt}$ is a deeper version of the decoder used in the original AdaIN implementation. It consists of 3 nearest up-sampling layers, 13 convolutional layers with kernels of size 3×3, and ReLU activations after each conv-layer except the last one. $E_{decrypt}$ stacks up 8 building blocks, where each building block contains a convolutional layer with kernels of size 3×3, a batch normalization layer, a ReLU activation, and a max-pooling layer with kernel of size 3×3.

Hyperparameters. The learning rate used in our model training is 10^{-4}. We adopt Adam optimizer [4] with hyper-parameters $\{\beta_1 = 0.5, \beta_2 = 0.999\}$. The batch-size is set to 8. The λ parameters for the objective function $\mathcal{L}_{end2end}$ are set as $\{\lambda_c = 2, \lambda_s = 10, \lambda_{dec} = 30, \lambda_{inv} = 5, \lambda_{des} = 5, \lambda_p = 1\}$.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 4, 7, 9

[2] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 4, 5, 7, 9, 11

[3] A. Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 5, 6

[5] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1, 5, 11

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 1, 8

[7] K. Nichol. Painter by numbers, wikiart. https://www.kaggle.com/c/painter-by-numbers, 2016. 1, 8

[8] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 5

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 4

[10] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5, 11

[11] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3

[12] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2, 5

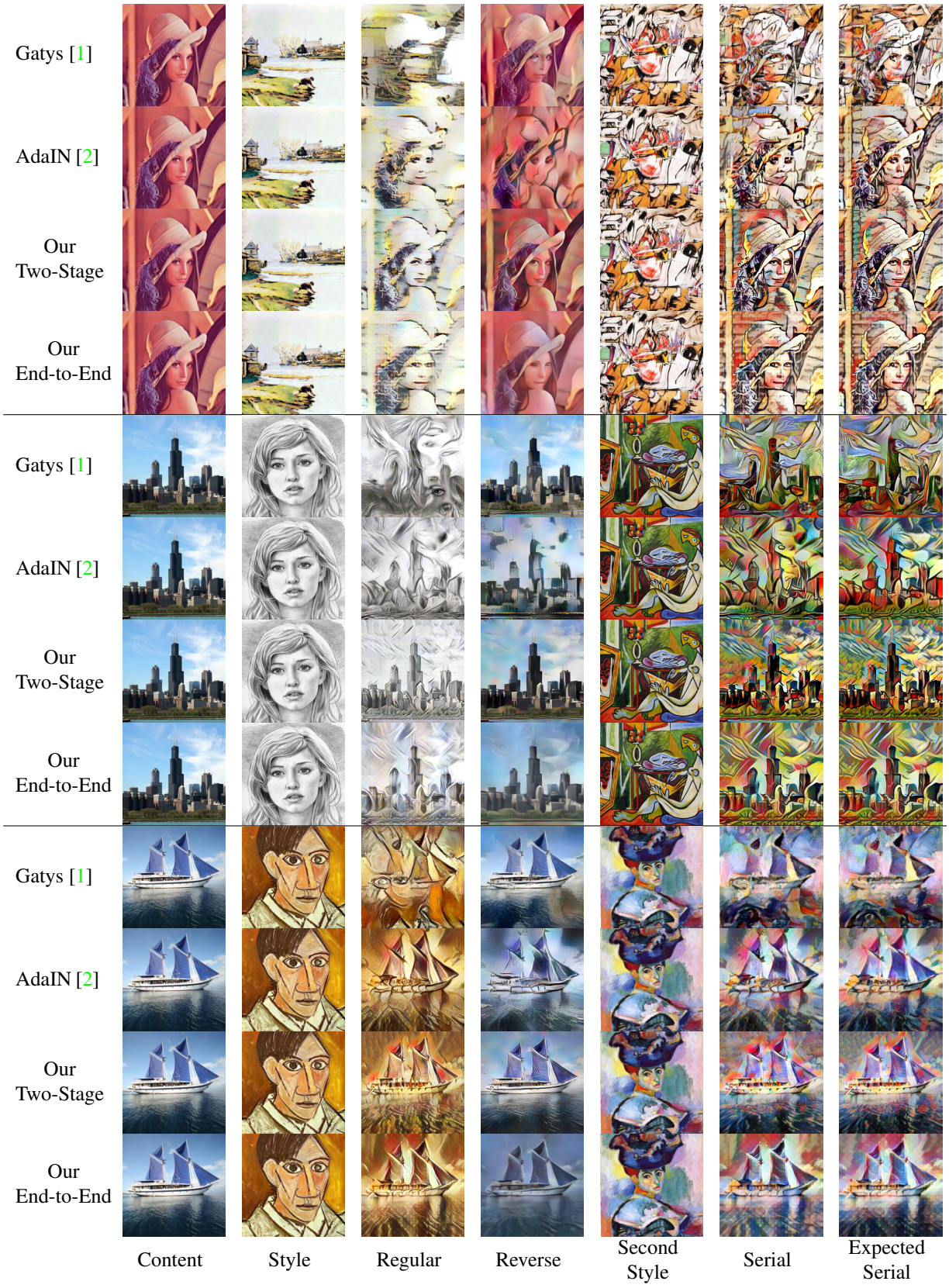

Figure 6: Three sets of additional results to demonstrate the comparison between different methods for regular, reverse, and serial style transfer. The rows in each set sequentially show the results generated by (1) Gatys *et al*. [1], (2) AdaIN [2], (3) our two-stage model, and (4) our end-to-end model.

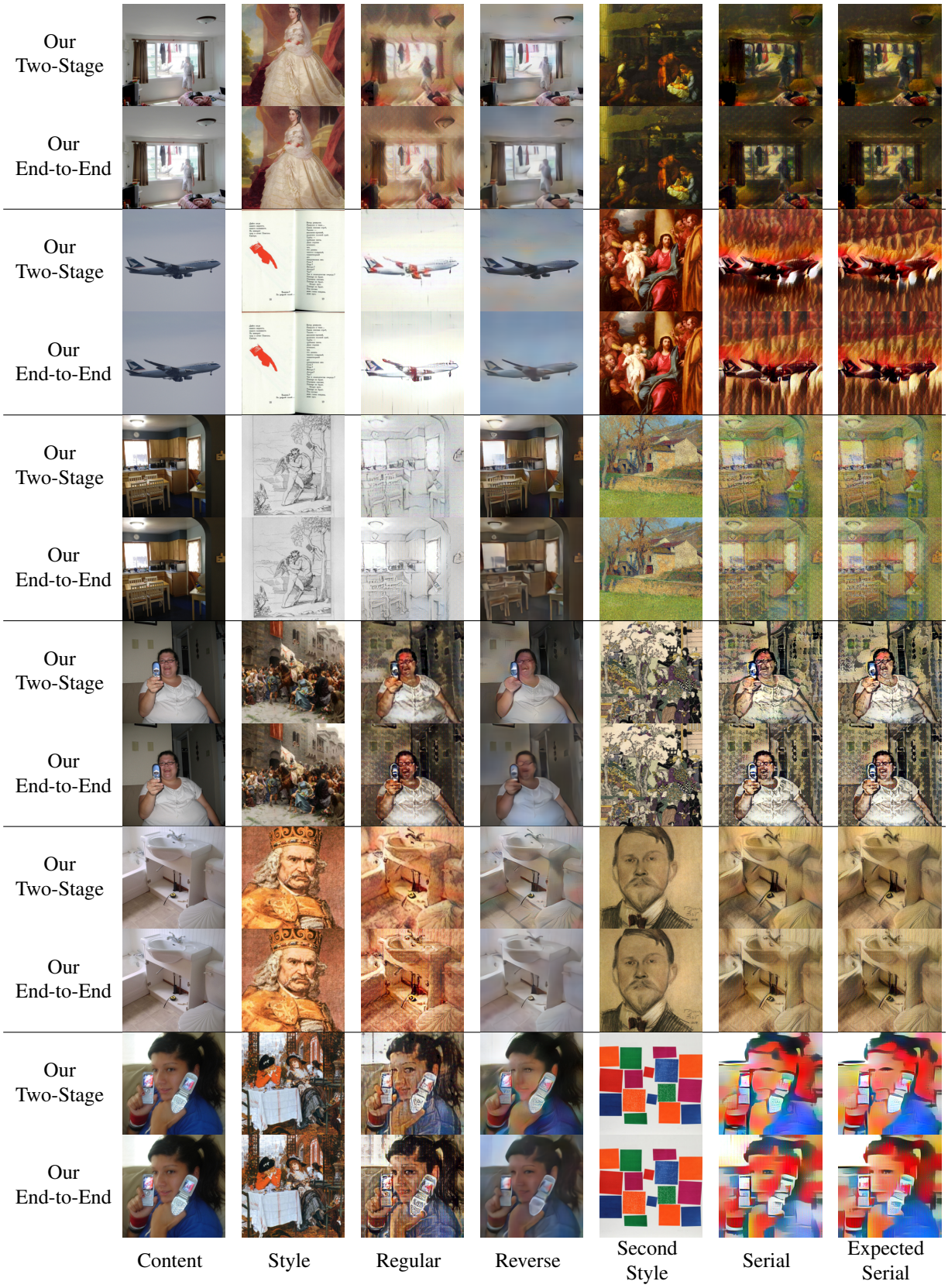

Figure 7: Example results of our proposed models in regular, reverse, and serial style transfer, based on diverse sets of content and style images from MS-COCO [6] and WikiArt [7] datasets respectively.

Figure 8: Three sets of example results of serial style transfer for multiple times. The top row contains the style images used in each serial style transfer. The rows in each set sequentially show the results generated by (1) Gatys *et al*. [1], (2) AdaIN [2], (3) our two-stage model, and (4) our end-to-end model. Except the leftmost column, which are the content images, every stylized image is generated with the content feature of the image in its left, and the style feature of the image at the top of the column. The content of the results produced by our proposed models are less distorted by the intermediate style transfer operations.

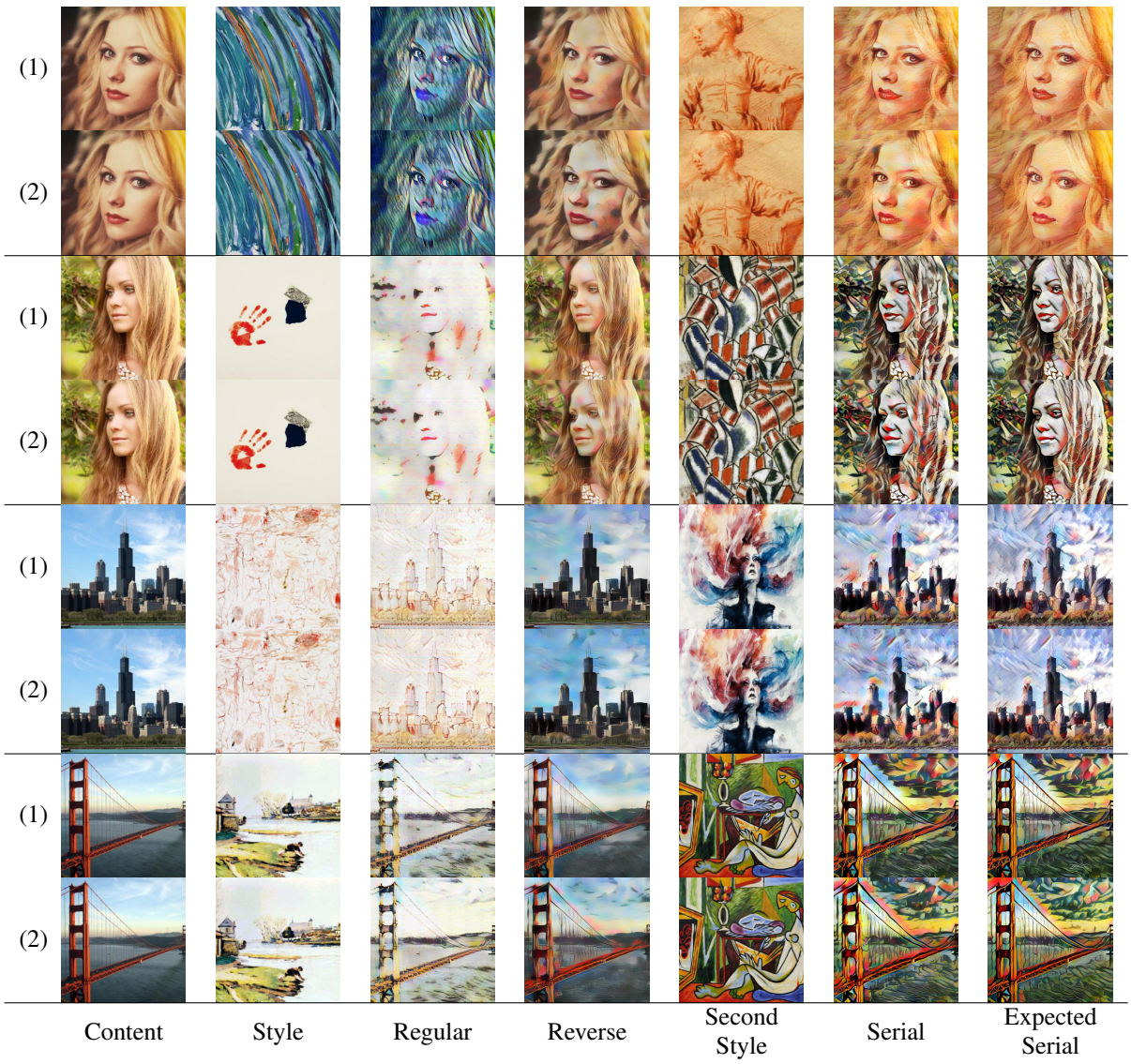

Figure 9: Comparison between the expected results for reverse and serial issues and the actual results generated w/ adversarial learning (1) and w/o adversarial learning (2).

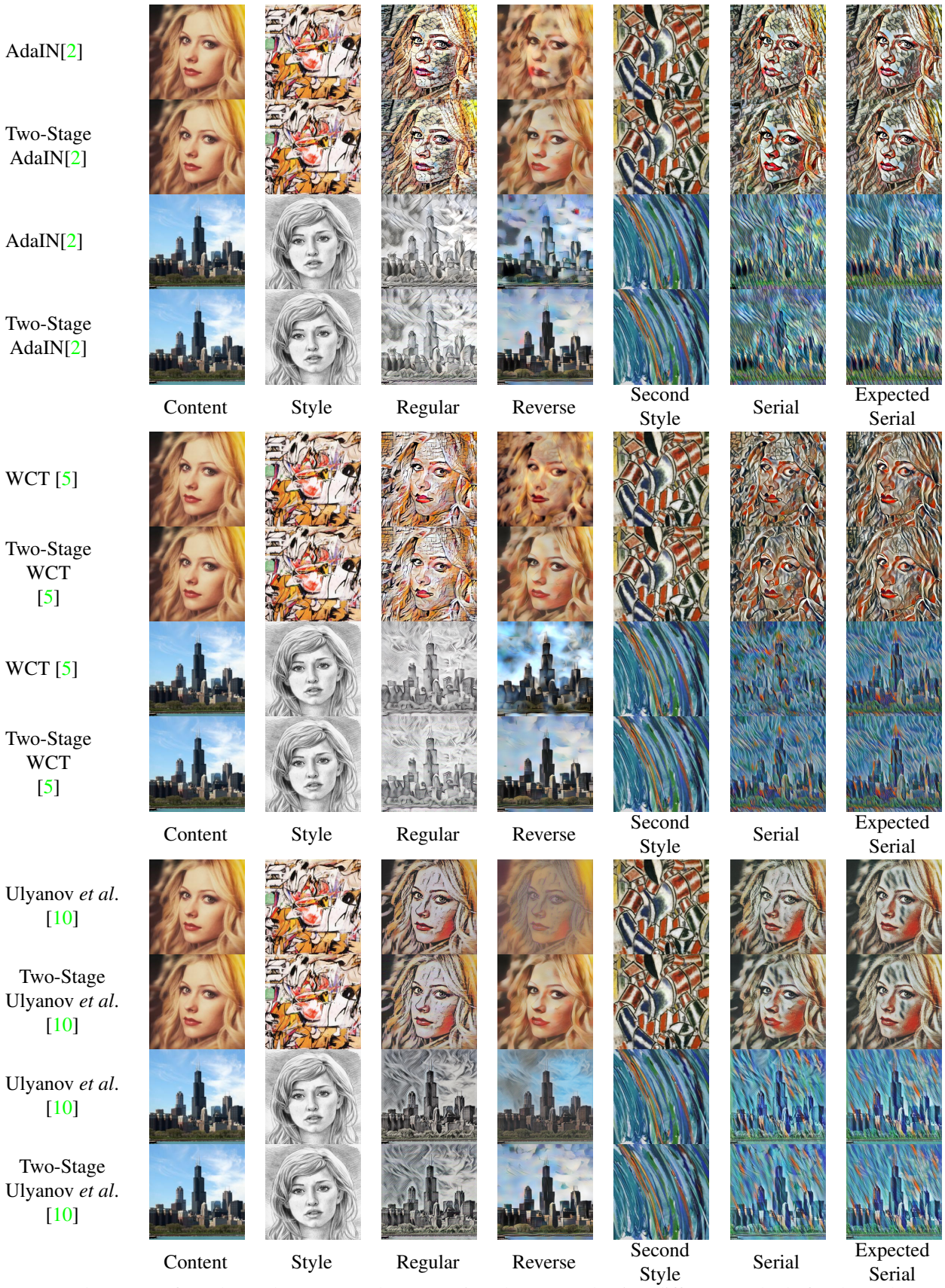

Figure 10: Three sets of results to demonstrate the comparison between adopting different methods for the style transfer stage of our two-stage model to perform regular, reverse, and serial style transfer. The rows in each set sequentially show the results generated by (1) AdaIN [2] and our two-stage model with AdaIN [2], (2) WCT [5] and our two-stage model with WCT [5], and (3) instance normalization [10] and our two-stage with instance normalization [10]