Dynamic-Net: Tuning the Objective Without Re-training for Synthesis Tasks —Supplementary—

Alon Shoshan
Technion, Israel
shoshan@campus.technion.ac.il

Roey Mechrez
Technion, Israel
roey@campus.technion.ac.il

Lihi Zelnik-Manor
Technion & Alibaba Group
lihi@technion.ac.il

1. Implementation Details

In this section we provide the implementation details for all applications presented in the paper. We added our tuning-blocks to known network architectures without changing the main network. The connection to the main network is done before a convolution layer, if the next layer is a residual layer, then the tuning-block is connected before the residual shortcut.

1.1. Style Transfer

We adopt Johnson et al. [4][1] [2] Fast style transfer as the main network. We added three tuning-blocks: (i) before the input of the third residual layer, (ii) before the first de-convolution layer and (iii) before the last de-convolution layer. We, first, train the main network for two epochs using 82K images from MS-COCO2014 [6] train set, the batch size is set to four. Then, we freeze the main network and train the tuning-blocks for two epochs. We use Adam optimizer [5] with the default parameters and learning rate 10^{-3} for the main network and 10^{-4} for the tuning-blocks.

1.2. Face Generation

We adopt the architecture suggested in DCGAN [8] as the main network. We added a single tuning-block before the input of the forth $conv$ layer. First, we train the main network for 20 epochs only on images with a specific attribute (e.g. dark hair) from CelebA [7] dataset, with batch size of 128. Than, we freeze the main network and train the tuning-block for additional 20 epochs only on images with the "opposite" attribute (e.g. blond hair). We use Adam optimizer [5] with the default parameters and learning rate $2e^{-4}$ for both the main network and the tuning-blocks.

1.3. Image Completion

In this experiment we adopted the architecture suggested in Isola et al. [3] for image-to-image transformation[3]. but with a modified training scheme. We added three tuning-blocks: (i) before the fifth down-sampling layer, (ii) between the last down-sampling layer and the first up-sampling layer and (iii) before the forth from the end up-sampling layer. We used CelebA [7] dataset for training and left out the first 1000 images for validation. We used Adam optimizer [5] and batch size of 64. During optimization we compute the loss term only on the completion region in the output image (i.e. the hole). The discriminator was fed with the completion region dilated by four pixels, that is from the given region around the hole.

[1] PyTorch implementation https://github.com/pytorch/examples/tree/master/fast_neural_style
[2] PyTorch implementation https://github.com/ceshine/fast-neural-style
[3] Authors release https://github.com/phillipi/pix2pix

2. Additional Results

We first present additional **style transfer** experiments, best presented on zoomed screen. We experiment with tuning each tuning-block individually in Figure 1, using our user interface[4]. We observe that each tuning-block has its own effect on the blending of the two styles. The user interface runs in real-time on a Nvidia GeForce GTX 1080 Ti. Figure 2 shows additional comparison results of our method, Arbitrary Style Transfer using AdaIN [2], Conditional-IN [1] and image interpolation.

We present additional **Image completion** results in Figure 3 and 4 and **Face generation** results in Figure 5.

We presented additional **style transfer** results; Figure 6 and 7 presents additional result of controlling the style. Figures 8, 9 present additional results of interpolation between two styles. Figures 10 and 11 present additional results of interpolation and extrapolation in the scale space of the style image, i.e. the resolution. Finally Figure 12 presents the style images used in our experiments.

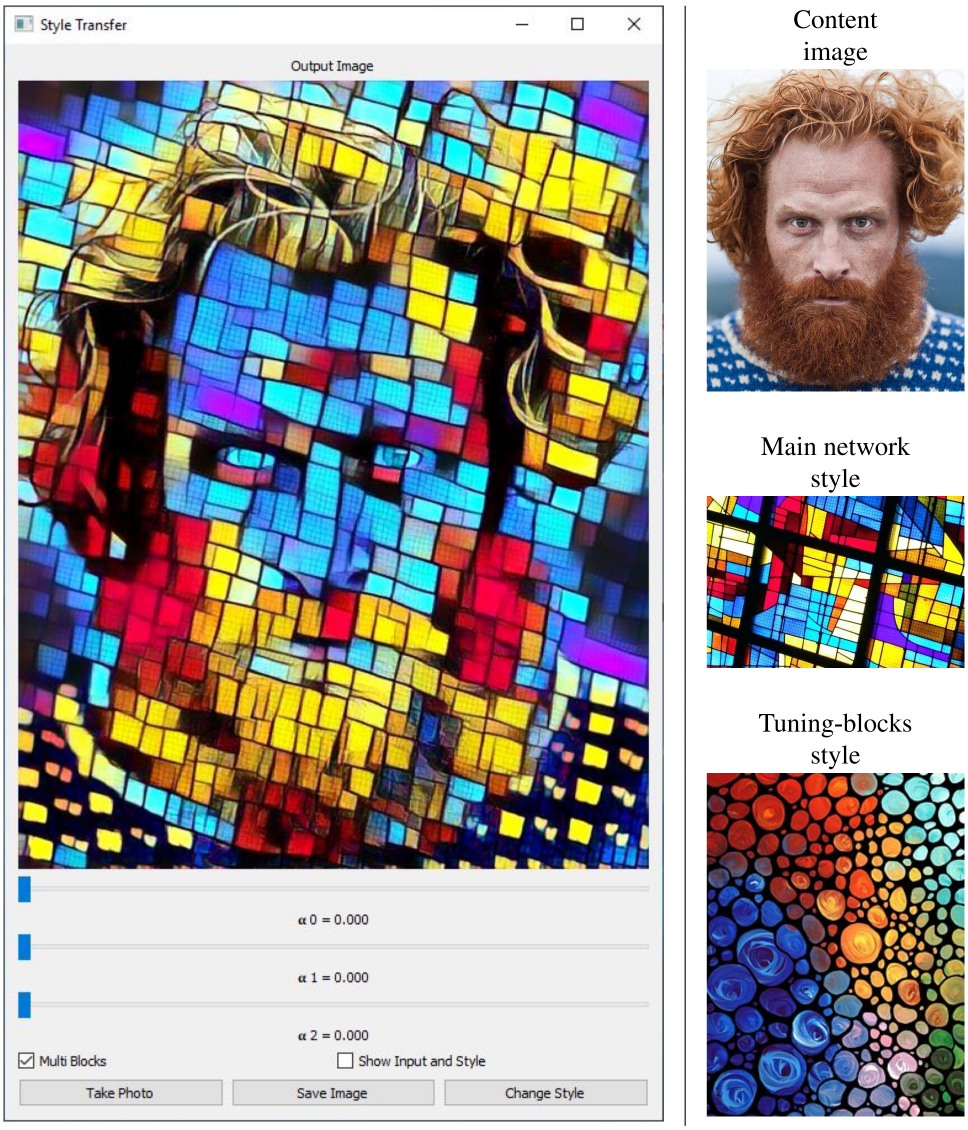

Figure 1: **User interface animation:** Using our user interface we can change each tuning-block α value individually (user interface runs in real-time on GPU). We observe that each tuning-block has a different impact on the output. The third tuning-block (α_2) is mostly responsible for color change (for ease of visualization we leave the same color in most of the animation). α_0 and α_1 are responsible for texture in different scales. [Animated figure, please view in Acrobat Reader].

[4] We developed a simple user interface to interactively change the values of α per tuning-block. Our code is available at https://github.com/AlonShoshan10/dynamic_net.

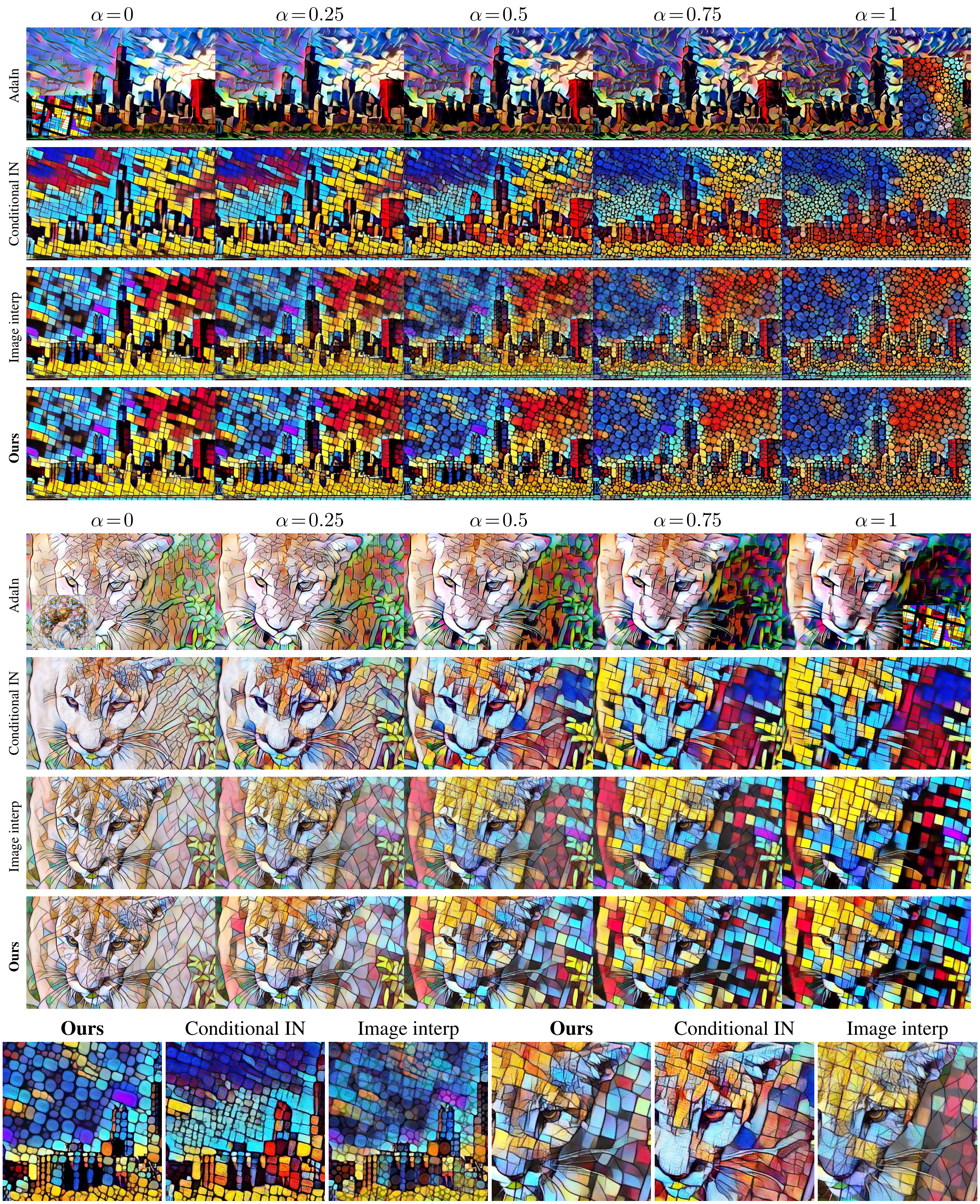

Figure 2: **Traversing between styles:** results of four different methods. Last row shows a zoomed-in patch of our method, conditional IN amd image interpolation for $\alpha = 0.5$. Best viewed zoomed-in.

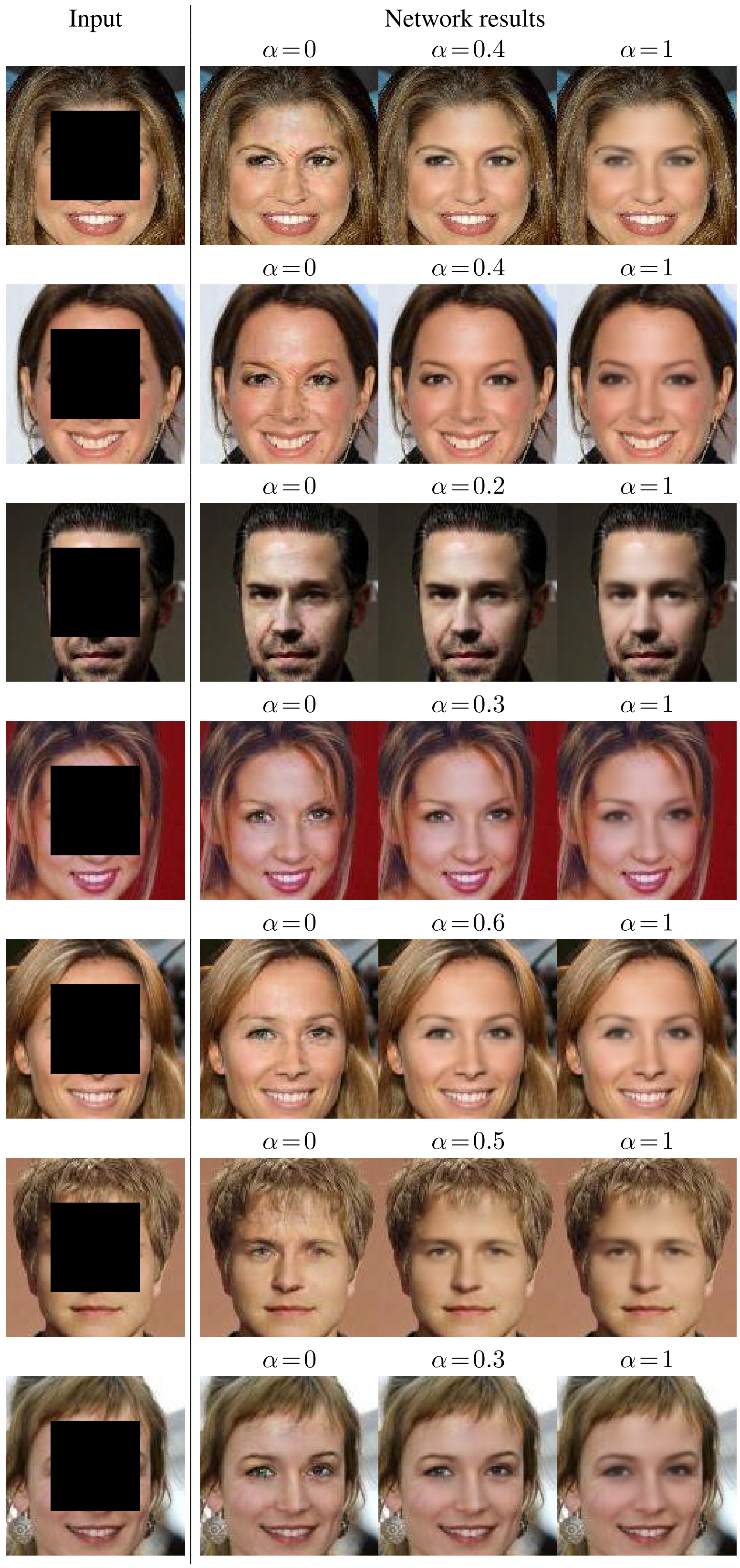

Figure 3: **Robustness to hyper-parameter:** The main network ($\alpha = 0$) produces artifacts common to adversarial training while $\alpha = 1$ produces blurry images common to L1 loss. Using $0 < \alpha < 1$ results in high quality images preventing the need to retrain the main network numerous times with different objectives to achieve high quality results.

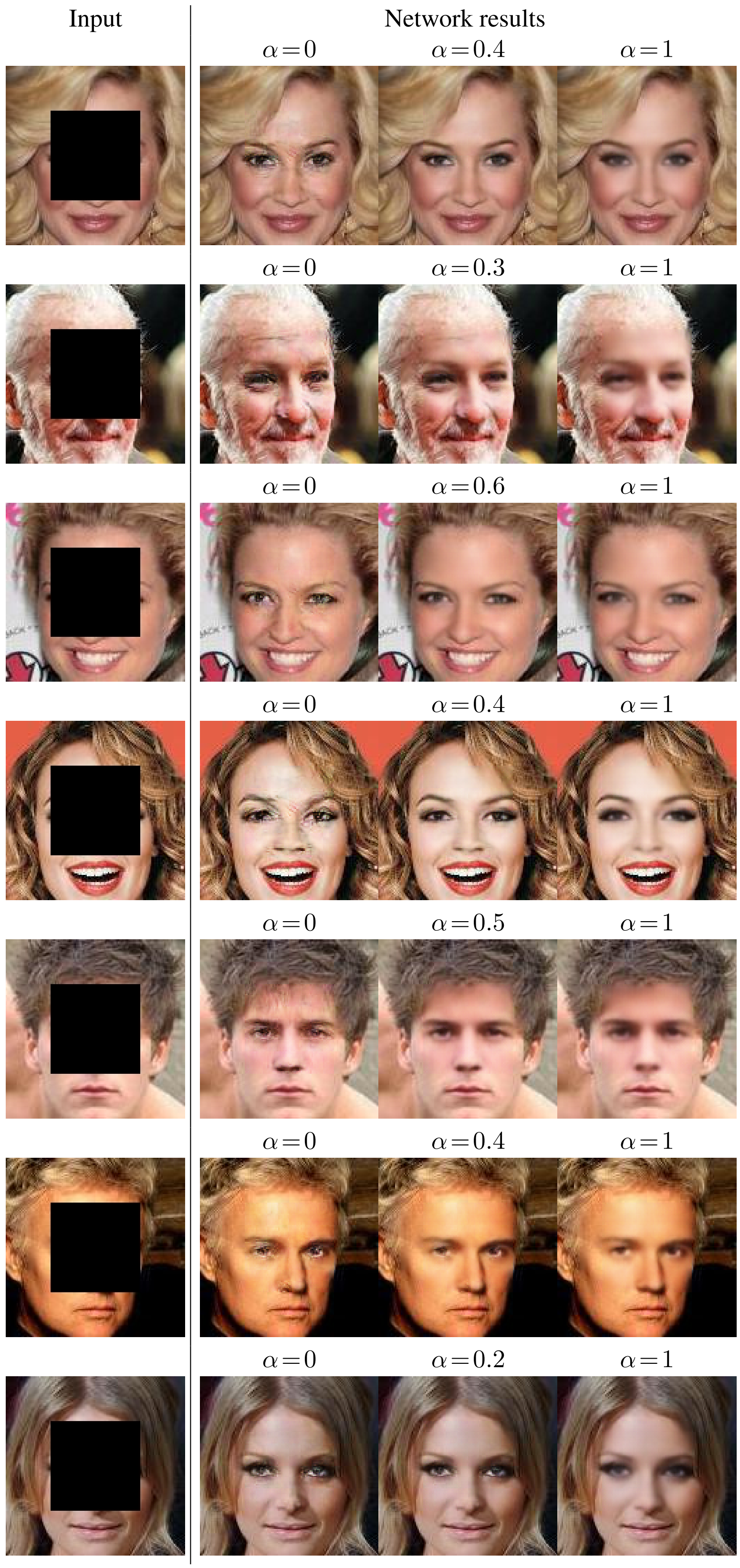

Figure 4: **Robustness to hyper-parameter:** The main network ($\alpha = 0$) produces artifacts common to adversarial training while $\alpha = 1$ produces blurry images common to L1 loss. Using $0 < \alpha < 1$ results in high quality images preventing the need to retrain the main network numerous times with different objectives to achieve high quality results.

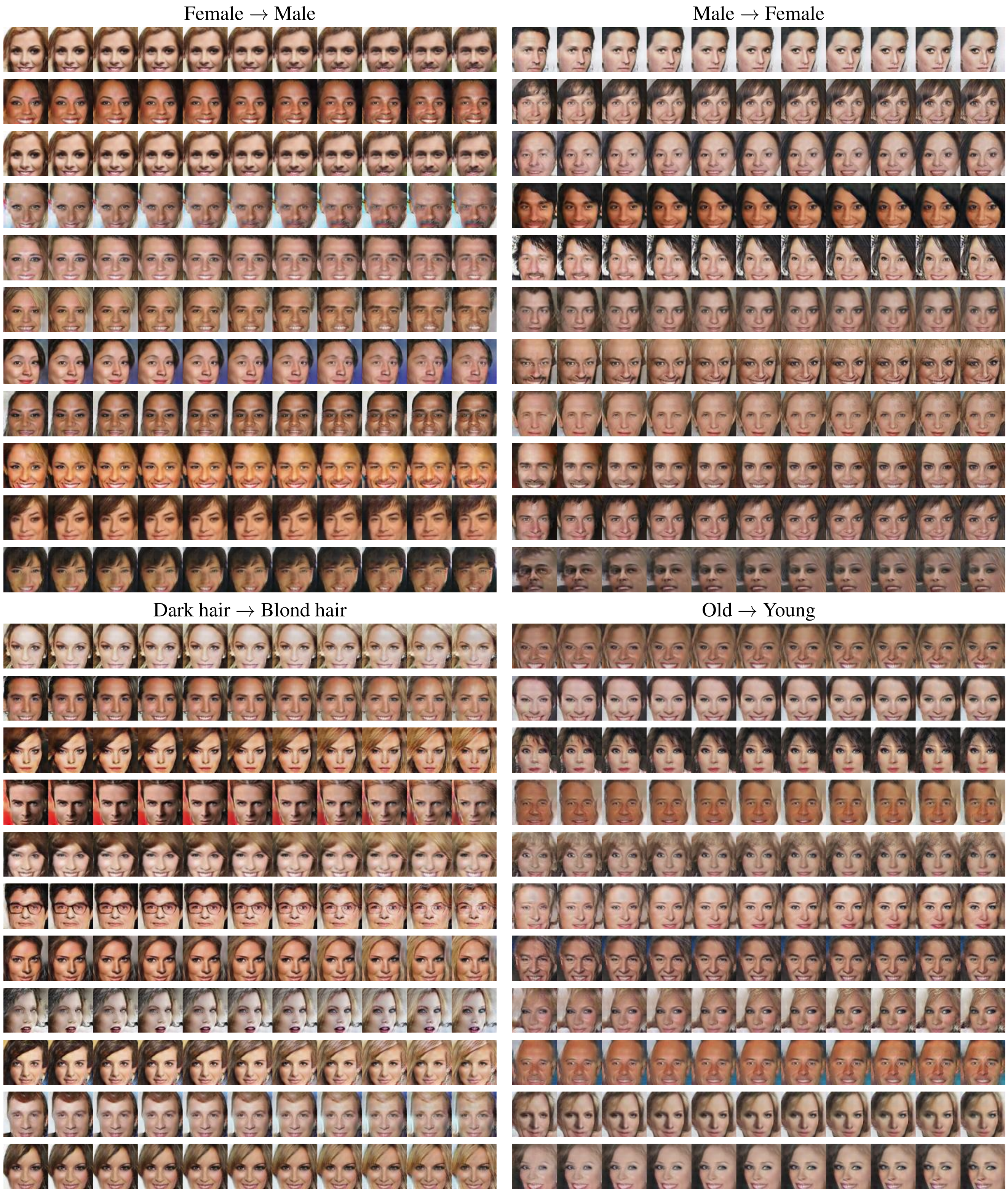

Figure 5: **Dynamic-DCGAN:** The proposed method allow us to interpolate between different facial attributes. The values of α are gradually increasing from left to right, results in a monotonic change of the specific attribute. Most left: $\alpha = 0$ correspond to the baseline result of DCGAN [8].

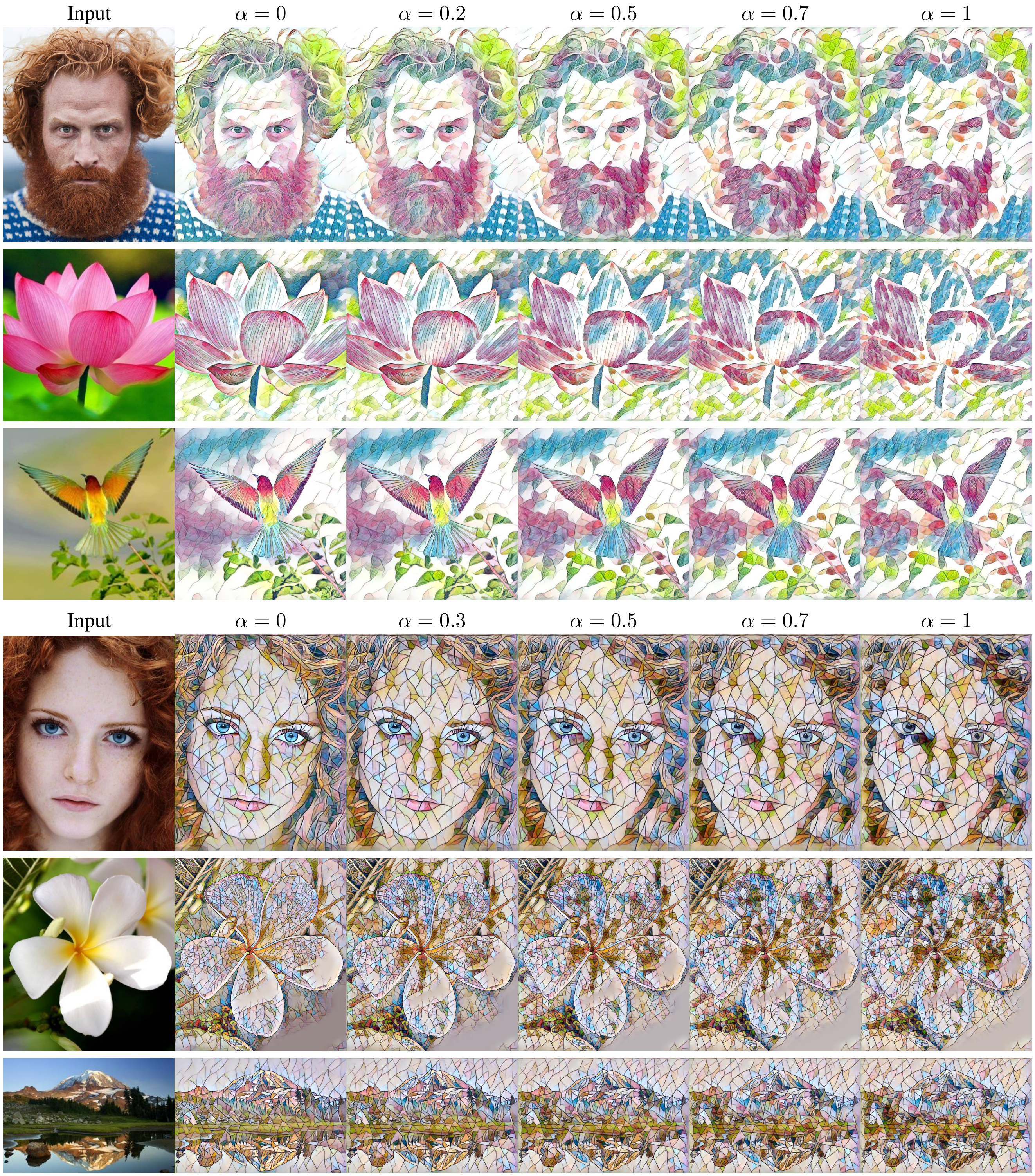

Figure 6: **Dynamic Style transfer:** Using tuning-blocks we achieve a monotonic change in style over a wide range of style levels. (top) style image 12a and $\lambda_0 = 10^4$, $\lambda_1 = 5 \times 10^6$. (bottom) style image 12b and $\lambda_0 = 10^5$, $\lambda_1 = 10^7$.

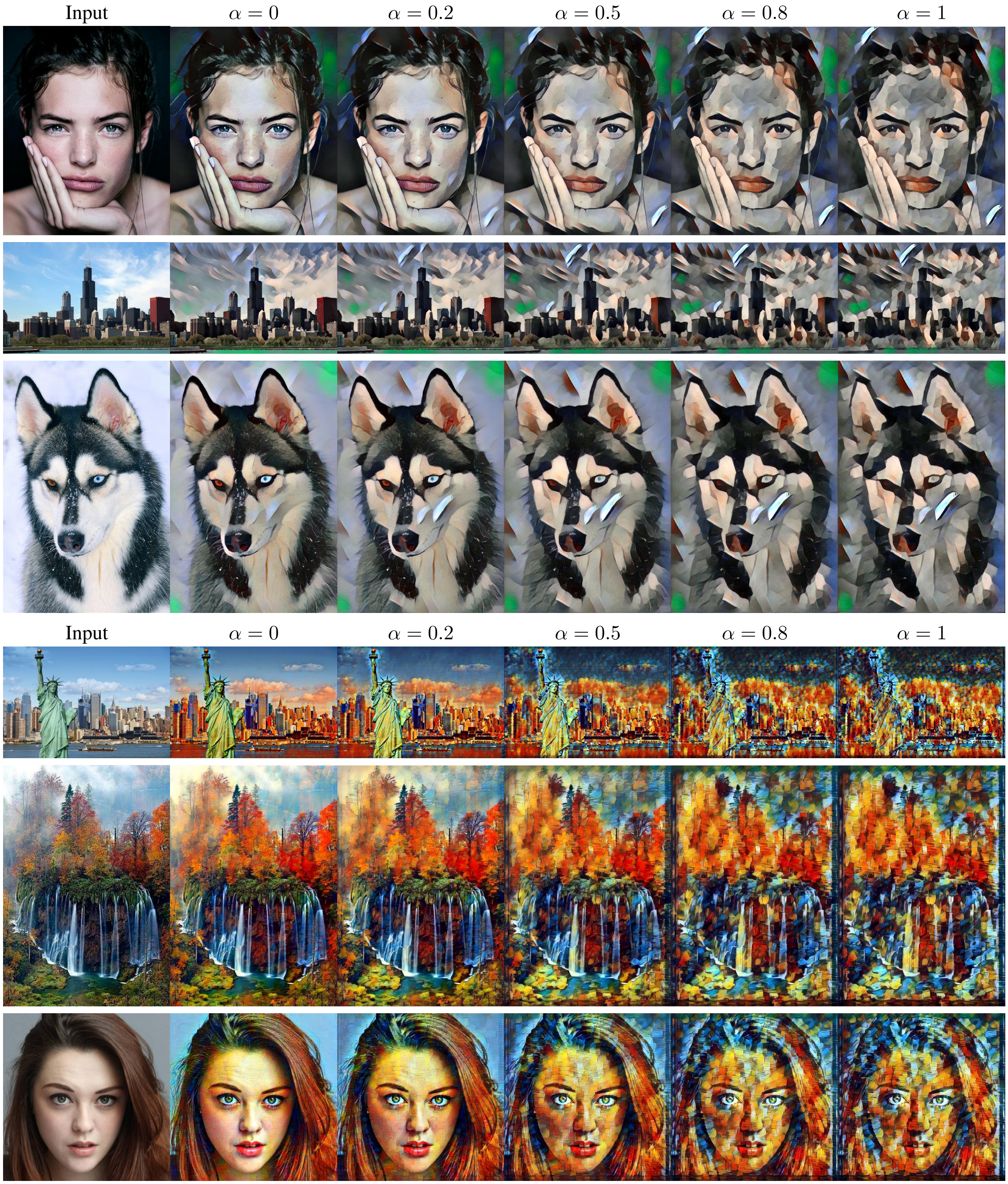

Figure 7: **Dynamic Style transfer:** Using tuning-blocks we achieve a monotonic change in style over a wide range of style levels. (top) style image 12c and $\lambda_0 = 10^4$, $\lambda_1 = 10^6$. (bottom) style image 12d and $\lambda_0 = 10^4$, $\lambda_1 = 10^6$.

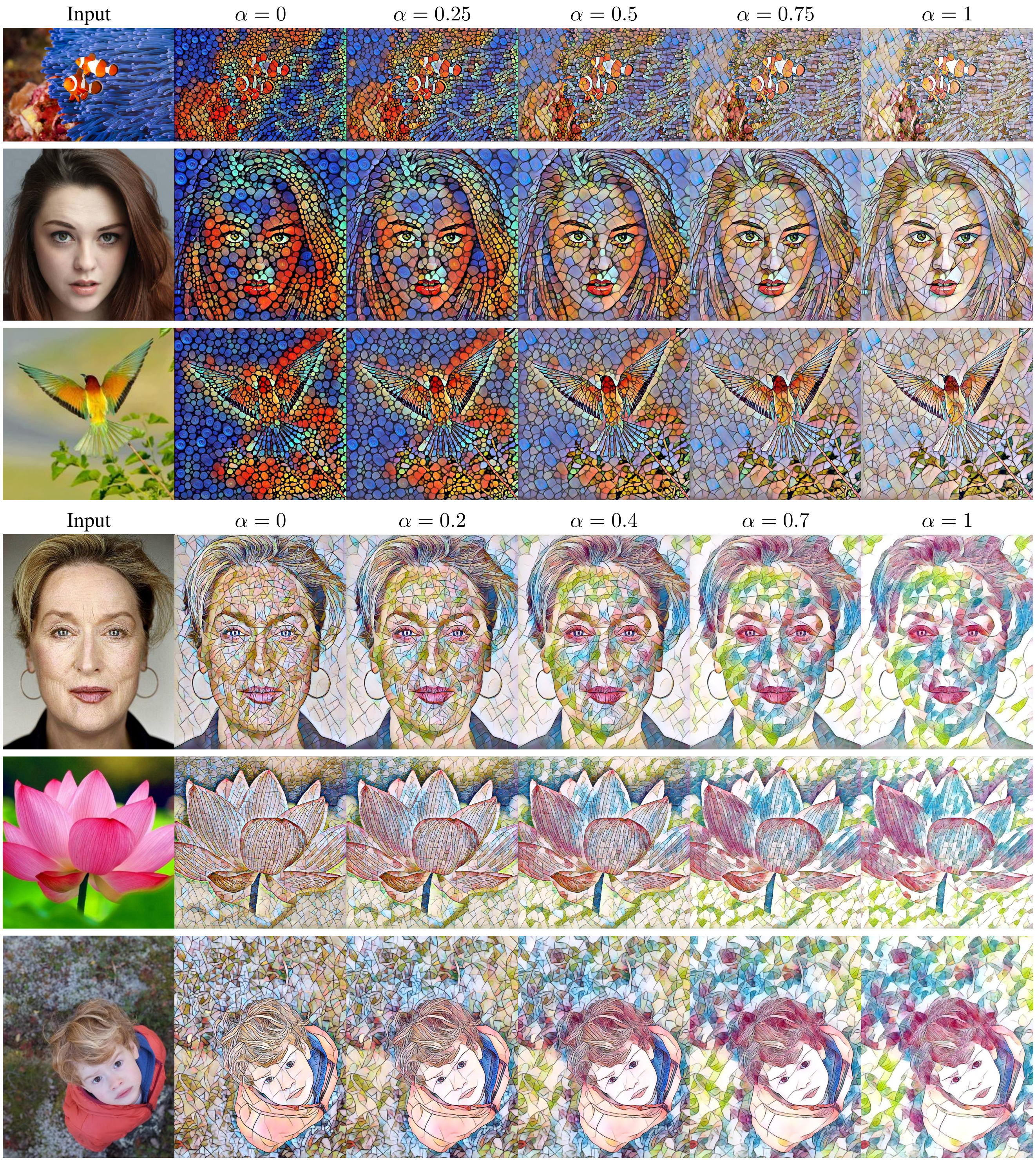

Figure 8: **Interpolation between two styles**: (top) Interpolation between style image 12j and style image 12a. (bottom) Interpolation between style image 12a and style image 12b.

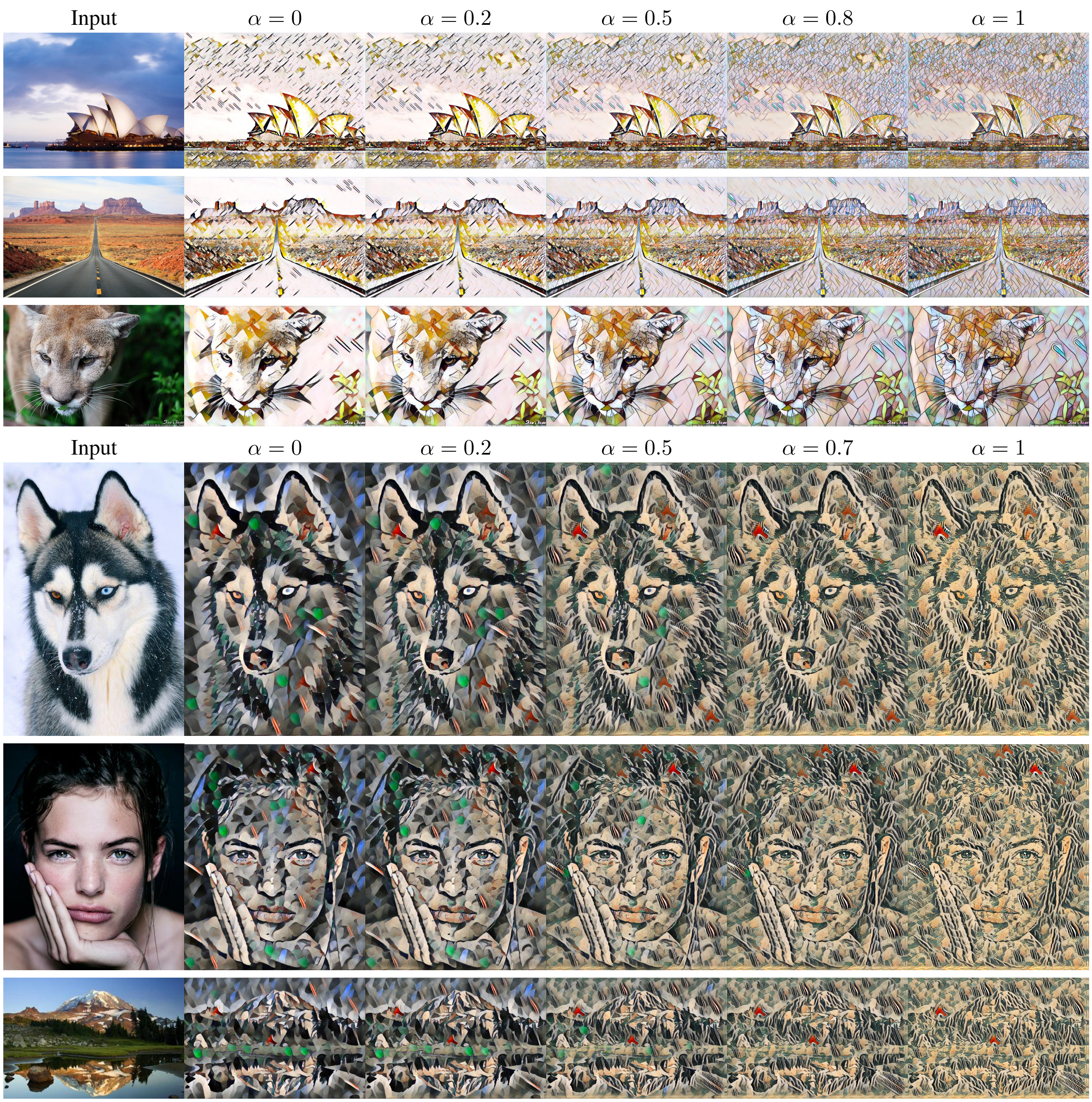

Figure 9: **Interpolation between two styles**: (top) Interpolation between style image 12g and style image 12a. (bottom) Interpolation between style image 12c and style image 12f.

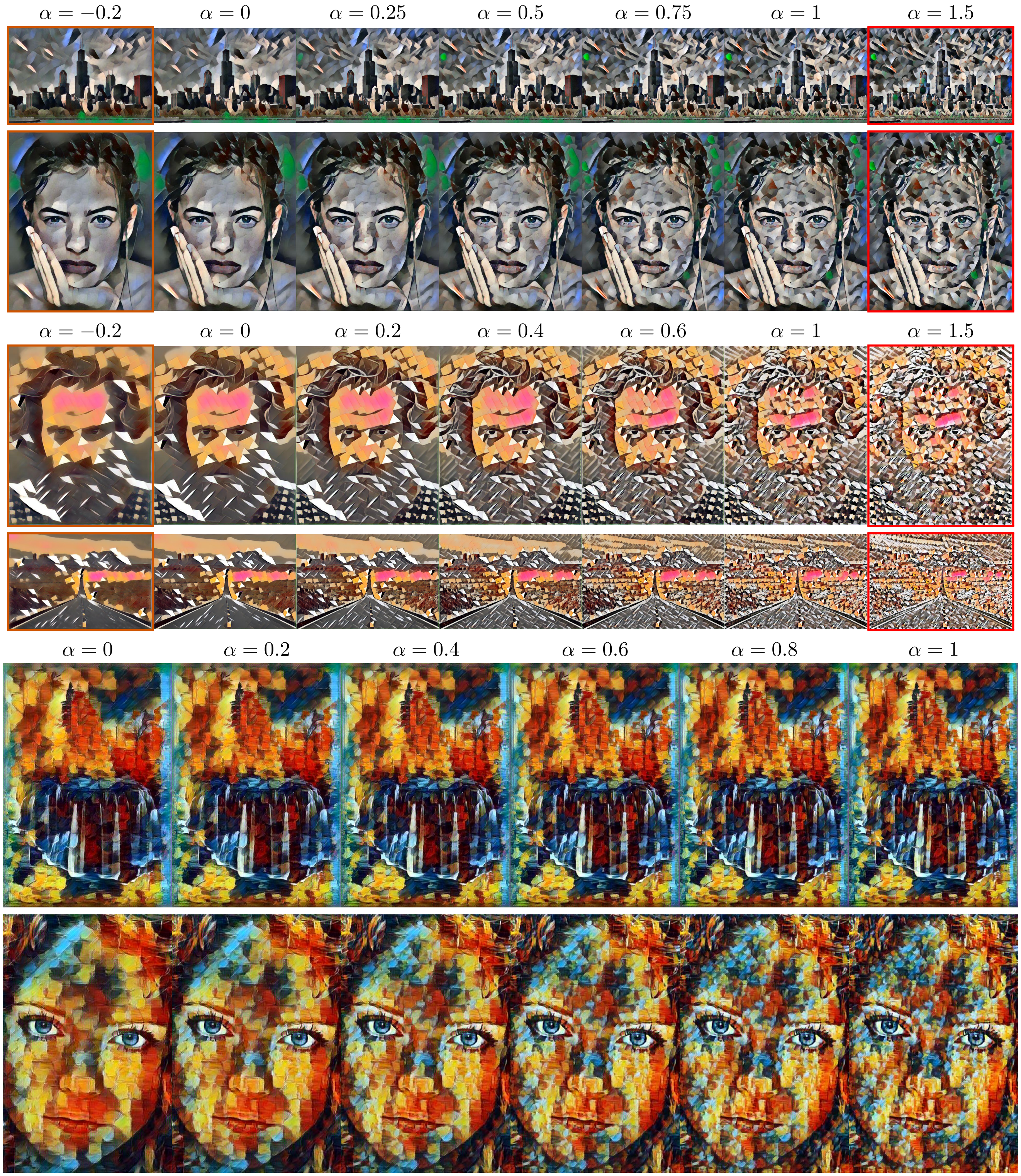

Figure 10: **Interpolation and extrapolation in style scale space**: (top) Interpolation and extrapolation of style image 12c. (middle) Interpolation and extrapolation of style image 12h. (middle) Interpolation of style image 12d. Red and orange rectangles show extrapolation results.

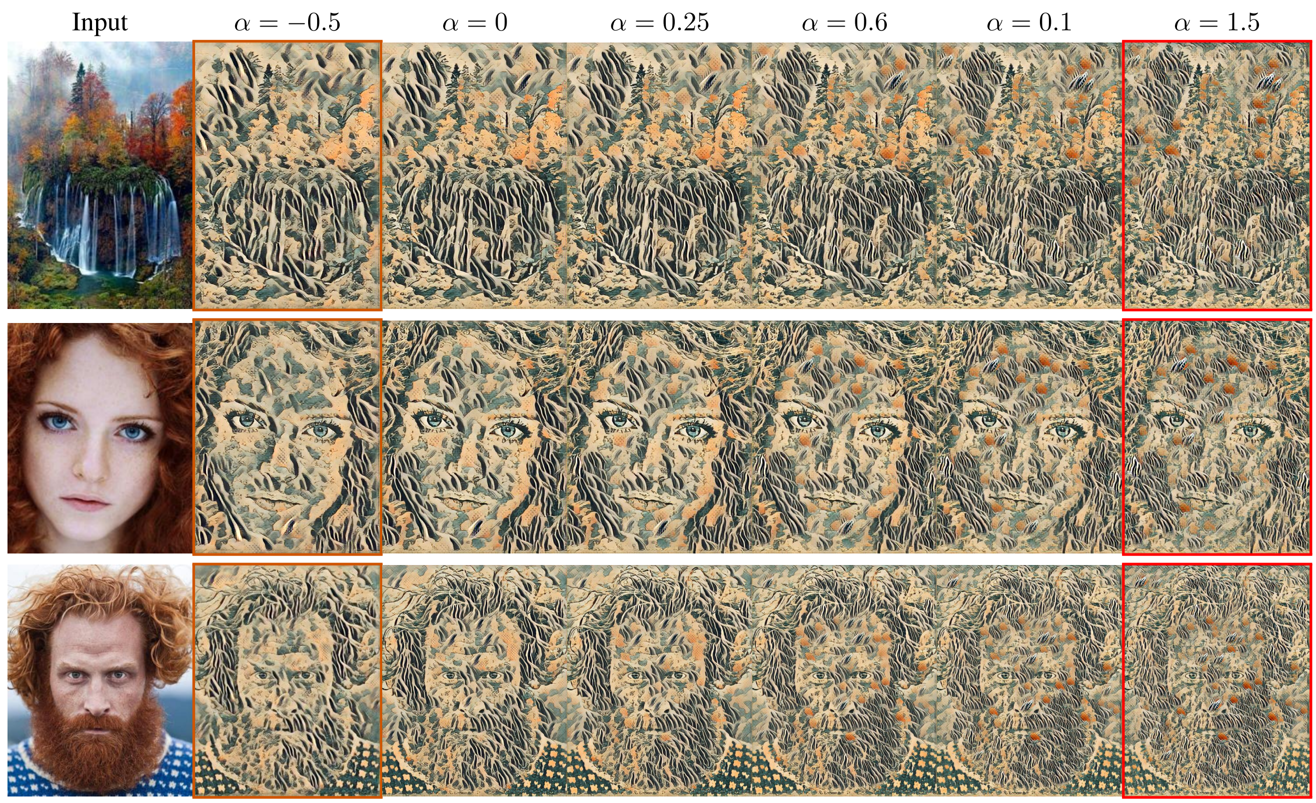

Figure 11: **Interpolation and extrapolation in style scale space**: Interpolation and extrapolation of style image 12f. Red and orange rectangles show extrapolation results.

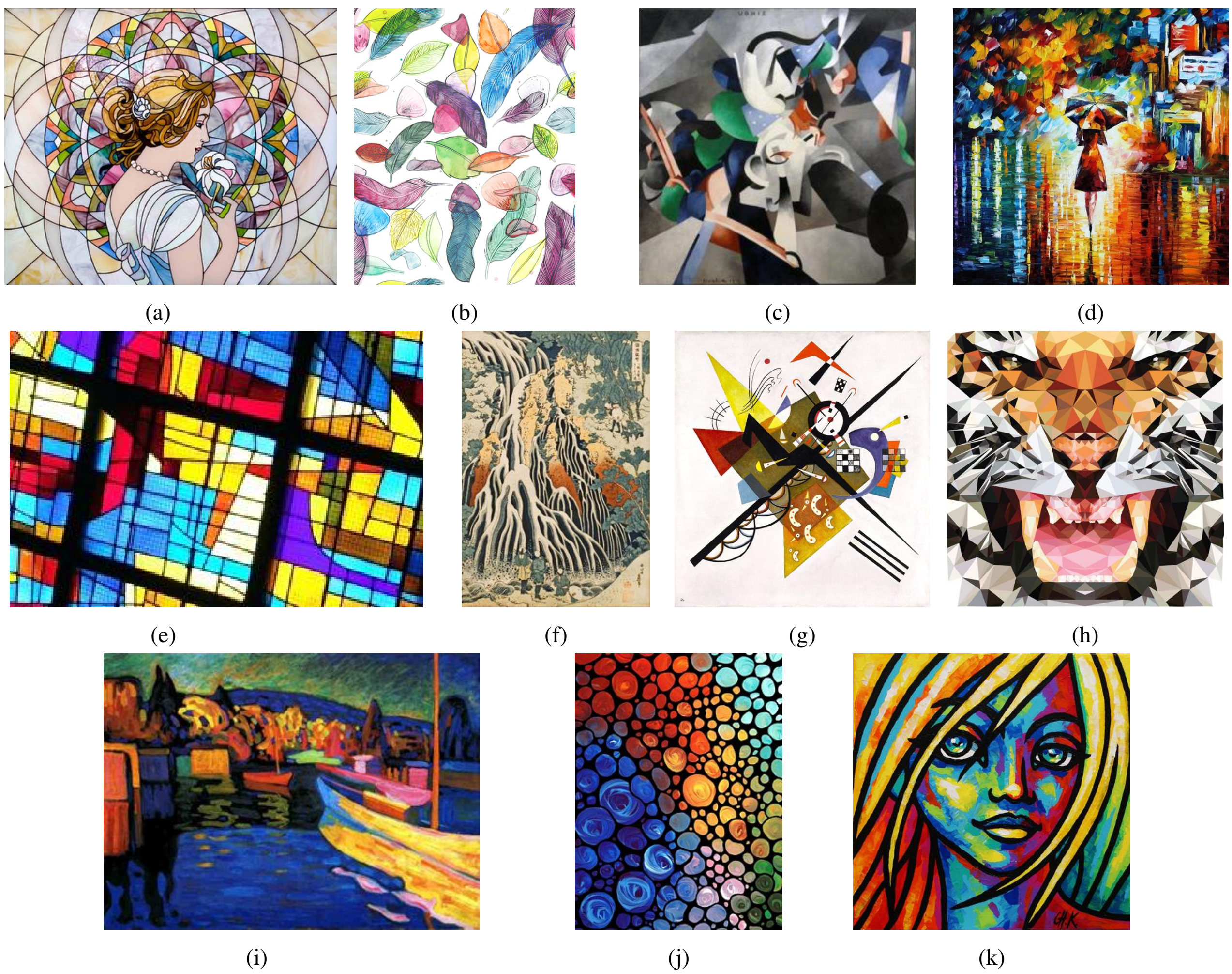

Figure 12: Style Images used in the paper and in the supplementary.

References

[1] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *Proc. of ICLR*, 2, 2017. 2

[2] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017. 2

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[6] 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. 1

[7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 1

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 1, 6