A. Domain Adaptation: from Game to Real.

The linear transformation based arbitrary style transfer shares a lot of resemblance with typical high-level ideas of domain adaptation (e.g., by Sun et al. [29]), in which both are proposed to align the second-order statistics between two different domains, e.g., style and content images [20]. A pioneer work proposed by Liu et.al [5] successfully utilize the WCT-based style transfer technique in the game-to-real domain adaptation with an unsupervised setting. In this section, we validate that the proposed model can significantly narrow down the domain gap for semantic segmentation between the game images and the real images.

First, we apply the PSPNet semantic segmentation model [35], which is well-trained on the Cityscapes [2] dataset, to segment the images from the GTA [26] which contains similar traffic scenes, but with different image quality of computer games. We then transfer the game images to "real" images, based on the proposed method. Specifically, we implement the task as photo-realistic style transfer by randomly taking one image from the GTA dataset as content input, and another image from the Cityscapes dataset as style input. The transformation is conducted w.r.t the corresponding semantic mask, where all non-shared labels from both image are considered as belonging to an additional class. We then process semantic segmentation using the same PSPNet model on the transferred results, as shown in Figure 15. The segmentation results on column 2 (segmentation results before domain adaptation) and column 4 (segmentation results after domain adaptation) demonstrate that the proposed algorithm is effective to adapt the domain of game images to real images in the semantic space. We note that narrowing the gaps between these two datasets (especially adapting games images to real world scenes) is practical for increasing training samples for semantic/instance-level segmentation, flow and depth estimation, to name a few, due to the fact that the dense annotations of these tasks in the real-world scenes are difficult to obtain. Our algorithm provide an efficient and low-cost solution to generate numerous training data, which could potentially benefit for various vision problems.

B. More Applications

We show more related applications that can be achieved via our method, including style interpolation (Figure 13) and user-controllable stylization (Figure 16).

B.1. Style Interpolation

We use $F_d{}' = \sum_{i=1}^{K} w_i F_d^i$ to interpolate between K styles with corresponding weights $w_1, \ldots, w_k$, where F_d^i is the transferred features for style i, similar to [12]. We feed the "weighted" feature F_d' into the decoder to obtain transferred image and show interpolation results between four styles in Figure 13.

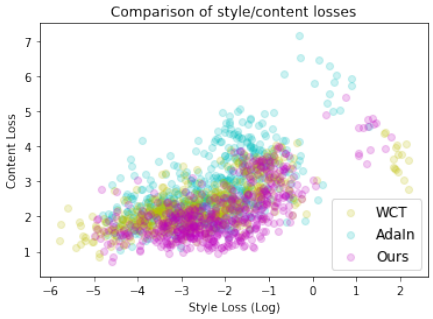


Figure 12. Content/Style loss comparison on holdout images. Magenta dots denote our algorithm.

Method	Style	Content	Speed (s)	Preference
AdaIn [4]	1.8	2.79	0.07	14.7%
WCT [5]	3.2	2.34	1.08	28.8%
Ours	2.6	**2.11**	**0.036**	**35.3%**

Table 2. Summary of evaluation between our algorithm and state-of-the-art methods. The style loss is shown without multiplying with 10^{-4}.

B.2. User-controllable Stylization

Our algorithm allows flexible controlling from user, by adjusting the weights between content and style. This is achieved by using $F_d{}' = \alpha F_d + (1-\alpha) F_c$, where F_c and F_d are content features and transferred features, and α controls the degree of stylization in synthesized images. We show results using different α in Figure 16. As shown in this figure, larger α leads to more stylized images.

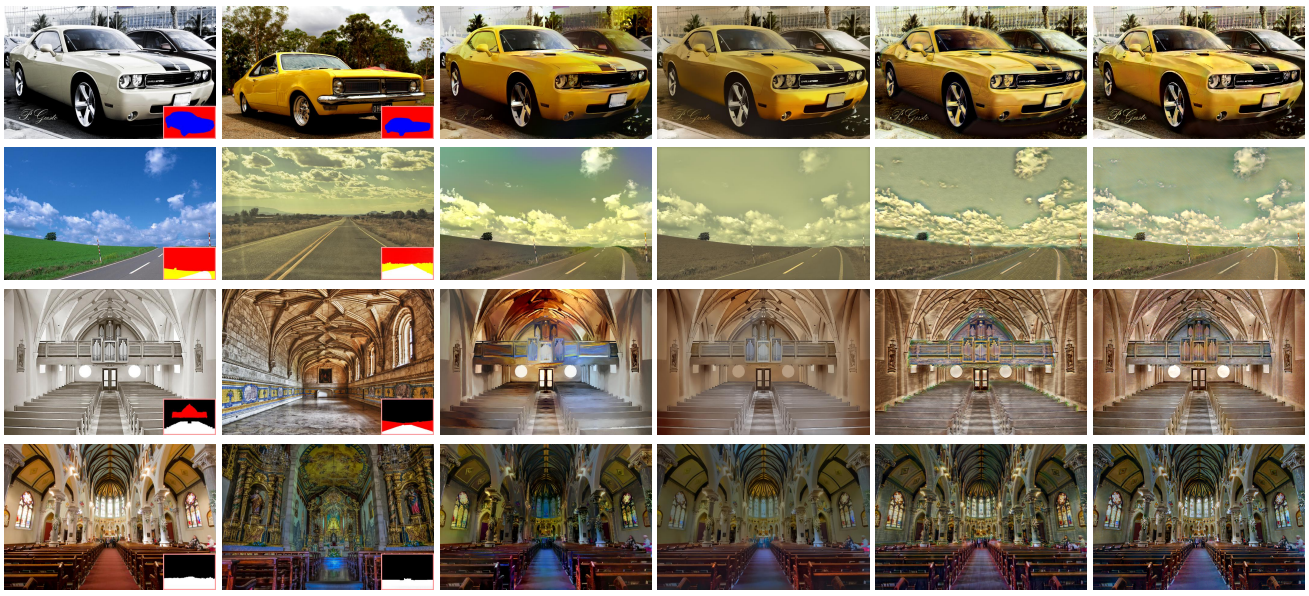
C. Experimental Results

In Figure 17, we show more comparisons between the state-of-the-art methods [10, 8, 30, 12, 18] and our algorithm for artistic style transfer. Unlike the methods [10, 12], our method does not introduce undesired color. For instance, the method proposed by Ghiasi et al. [10] introduces *undesired redness* in row 2, 8 and 9 in Figure 17, while our transferred results is consistent with the overall color distributions in the style images. We show more comparisons with the state-of-the-art methods and our algorithm for photo-realistic style transfer in Figure 14. Thanks to the end-to-end architecture and the data-driven training strategy, our algorithm is not only faster, but also preserves the overall contrast compared to the other methods [23, 19] that contain two separate stages.

D. Quantitative evaluation.

We summarize various quantitative evaluation metrics including the style transfer loss, inference time and user study preference percentage of different methods in Table 2. We also visualize loss comparisons following the work of Ghiasi et al [10] and WCT [18] in Figure 12. Table 2 and Figure 12 show that our algorithm not only achieves *lower loss* but also *faster speed* than [12, 18].

Figure 13. Style Interpolation. Our algorithm allows flexible combination of styles by feeding the decoder "weighted" transferred features of these styles.



Content Style Luan [23] Li [19] Ours +SPN

Figure 14. Photo-realistic style transfer results comparison. Spatial mask is displayed at the right bottom corner of each content and style image. "+SPN" in last column means results filtered by a SPN after stylization. Zoom-in to see the details.

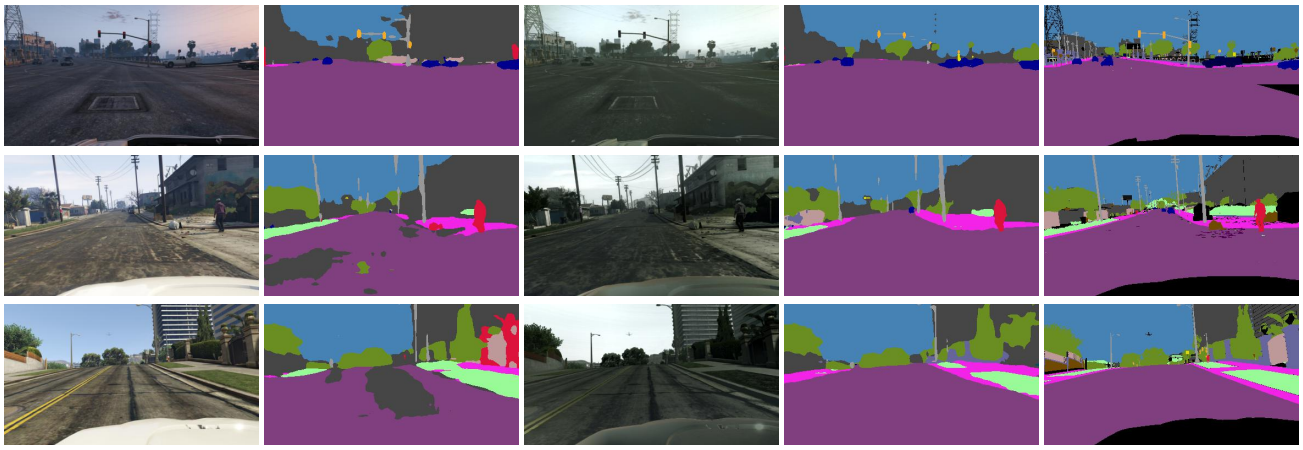


Figure 15. Segmentation results by the PSPNet [35] before and after domain adaptation by our method. Column 1 shows game images while column 3 shows transferred results by taking images from column 1 as content inputs and images from the Cityscapes [2] dataset as style inputs. The segmentation results in column 4 are better than results in column 2, showing the usefulness of our algorithm in domain adaptation.

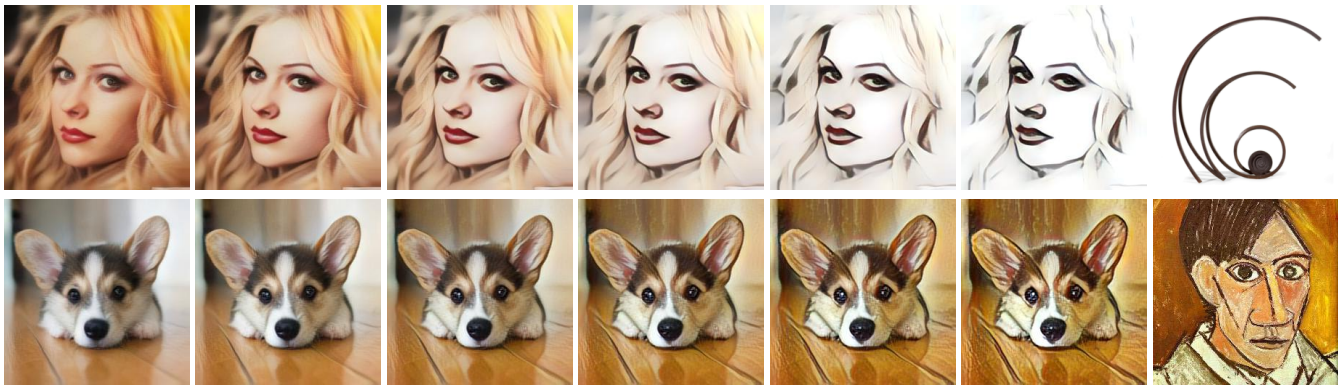


Figure 16. Content style trade-off. Our algorithm allows content-style trade-off at inference time by using different α. Larger α leads to more stylized results.

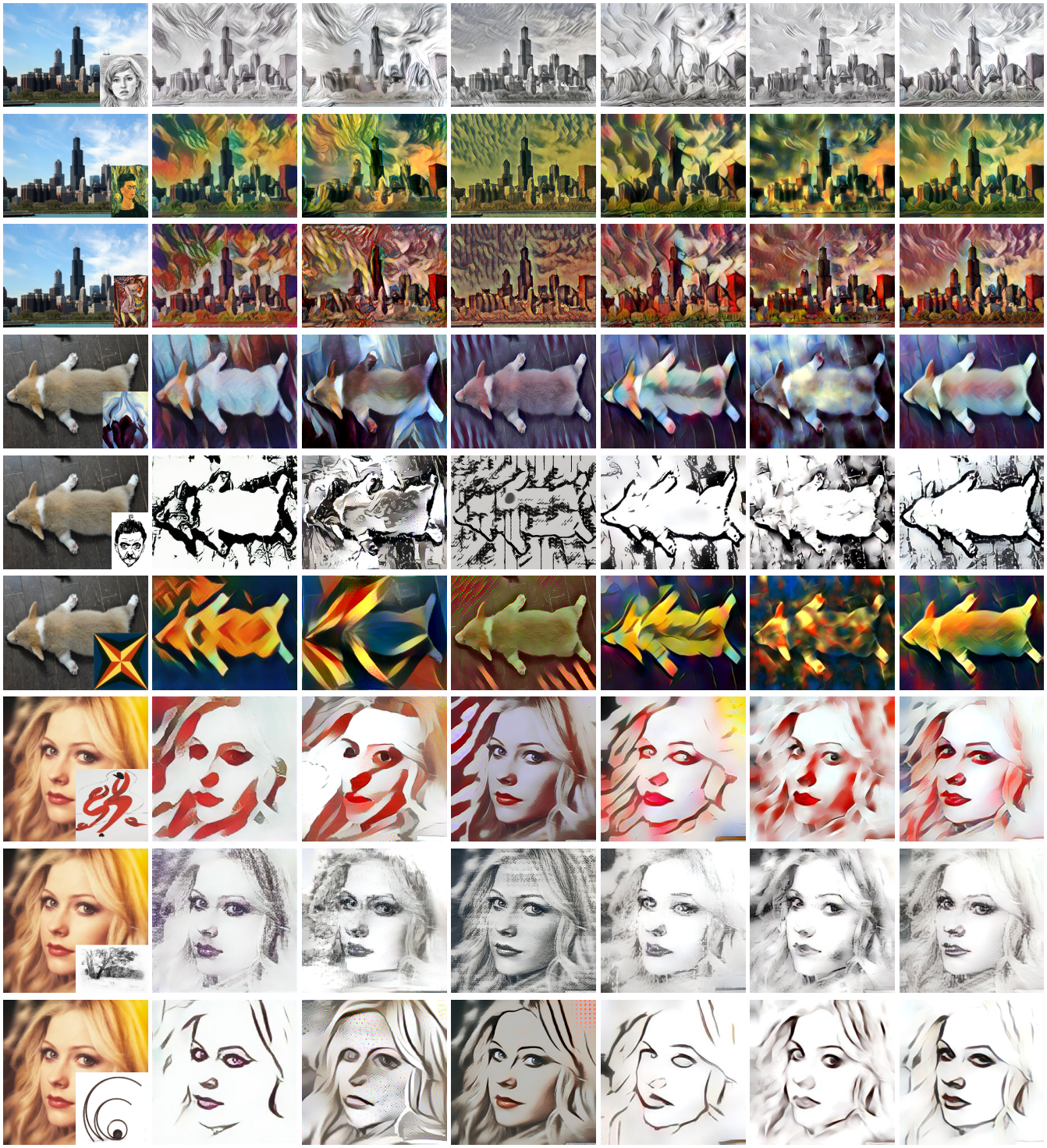


Figure 17. Comparison between the results by our style transfer algorithm and other methods. Our model is trained to transfer the content features from *relu4_1*, and with style losses computed on the *relu1_1*, *relu2_1*, *relu3_1*, *relu4_1* layers of VGG-19. No content image as well as style image presented here is included in the training set.