Supplementary Material: Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks

Chuan Li and Michael Wand

Institut for Informatik, University of Mainz, Germany

1 Architecture details

Figure 1 illustrates the parameters of each layer. Including the *VGG* network till layer Relu4_1, the parameters of the our generative model takes 70 Mb memory. In practice, the required memory to decode a photo linearly depends on the photo's size: for a 256-by-256 picture it takes about 600 Mb, and for a 512-by-512 picture it requires about 2.5 Gb memory. Memory usage can be optimized by subdividing the input photo into blocks and run the decoding in a scanline fashion.

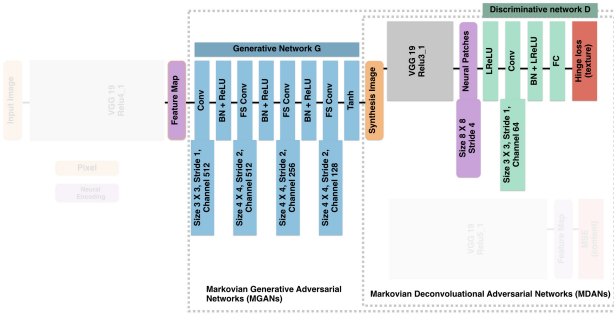

Fig. 1: This figure shows the parameters of our model.

2 Comparison with Deconvloutional Approaches

We first compare our results with existing deconvolutional approaches Li et al. [3] and Gatys et al. [2] (Figure 2 to 4). Speed-wise our method is significantly better than the other two methods (10000 times faster than Li et al. [3], and 1000 times faster than Gatys et al. [2]). The purpose here is to study the quality difference between these methods.

We use Justine Jason's implementation [1] to produce results for Gatys et al. [2], and the author's own implementation to produce results for Li et al. [3] [2].

[1] https://github.com/jcjohnson/neural-style
[2] https://github.com/chuanli11/CNNMRF

Our results are generated with the MGANs model trained with the example texture and 100 randomly selected ImageNet photos.

Our studies show that the favorable spectrum of these three methods differs: Li et al. [3] has the best performance for textures that are less random (Figure 2); Gatys et al. [2] works superiorly for stochastic textures (Figure 4). As a nice complement, our method gives the best results for the texture spectrum between these two cases (Figure 3). Please see the caption for each image for details.

3 Comparison with Generative Approach

Now we compare our method to Ulyanov et al. [1], a concurrent work that also uses feed-forward synthesis (Figure 5 and 6). Both methods runs at the same level of speed that is significantly faster than the deconvolutional approaches. For comparison, the input photos, example textures and the competitor's results are original pictures from the paper of [1]. Our results are generated with MGANs trained with the example texture and 100 randomly selected ImageNet photos.

By capturing a global feature distribution, Ulyanov et al. [1] are able to preserve the global color scheme of the example texture, and generate more natural appearance at background regions. In contrast, via learning a mapping from content to texture, our method applies the textures more coherently, and renders the foreground regions more profoundly. Please see the caption for each image for details.

4 Style Enhancement

Although our model is trained with a fixed weight for the content loss (α_1 in Equation (1) in the paper), it is still possible to control the stylization via post-processing. Here we offer two options for style enhancement: 1) Injecting low frequency noise to the input photo to get stronger textures; 2) Recursively decoding the image with the previous round's output as the next round's input. We provide a script (Model/demo_MGANs.lua) to test option 1), and show the outcome of option 2) in Figure 7. Notice how the style of the synthesized image gets stronger and stronger with more decoding iterations in Figure 7 .

References

1. Dmitry Ulyanov, Vadim Lebedev, A.V.V.L.: Texture networks: Feed-forward synthesis of textures and stylized images. CoRR abs/1603.03417 (March, 2016), http://arxiv.org/abs/1603.03417v1 2, 7, 8
2. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style (2015), arXiv preprint; http://arxiv.org/abs/1508.06576 1, 2, 4, 5, 6
3. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. CoRR abs/1601.04589 (2016), http://arxiv.org/abs/1601.04589 1, 2, 4, 5, 6

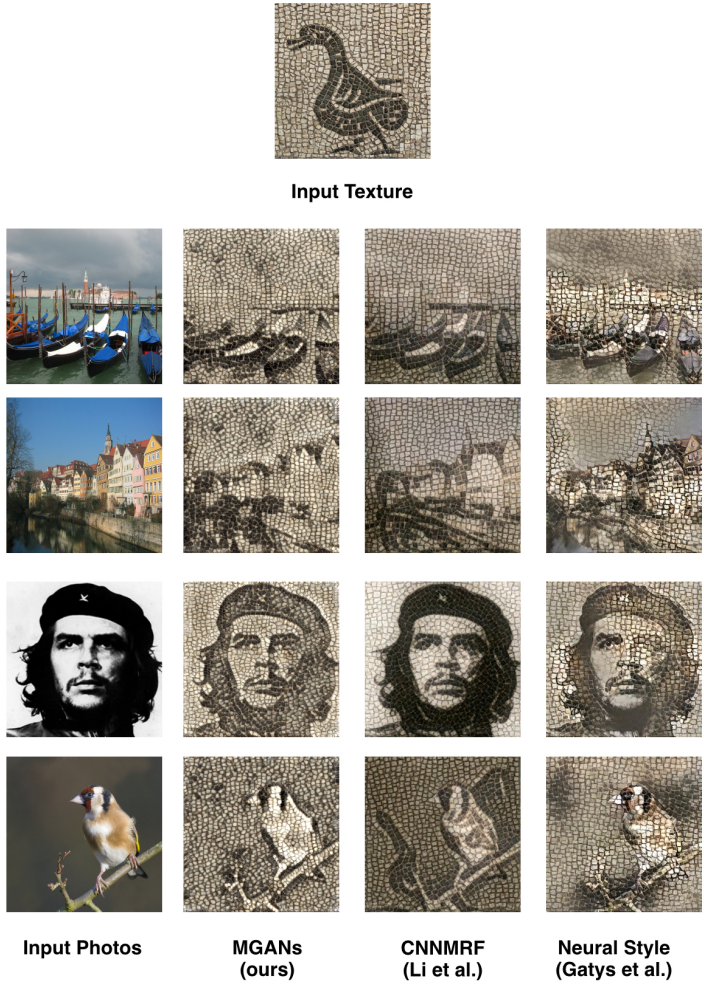

Fig. 2: This figure compares our results with deconvolutional methods [3,2]. The example texture contains two distinct components: bright stones and dark stones. In this case, the Gaussian based model [2] produces the least satisfactory results due to the miss-matching between the character of the texture and the model. Local nearest neighbor research [3] produces the best results by avoiding appearance distortion using rigid data copy. Our results are inferior to [3] but superior to [2].

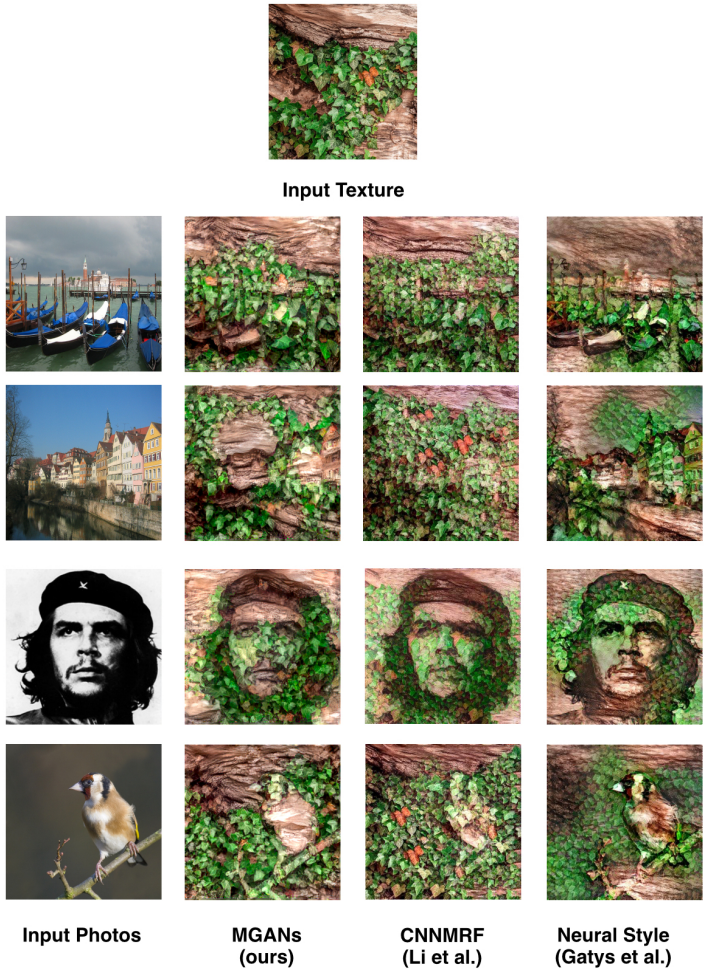

Fig. 3: This figure compares our results with deconvolutional methods [3,2]. In this example the texture is slightly more stochastic than the one in Figure 2. Notice Li et al. [3] appears too rigid to adapt the texture to the photos. Gatys et al. [2] get improvement but still creates lower quality than our method (for examples, they are not able to produce convincing leaves).

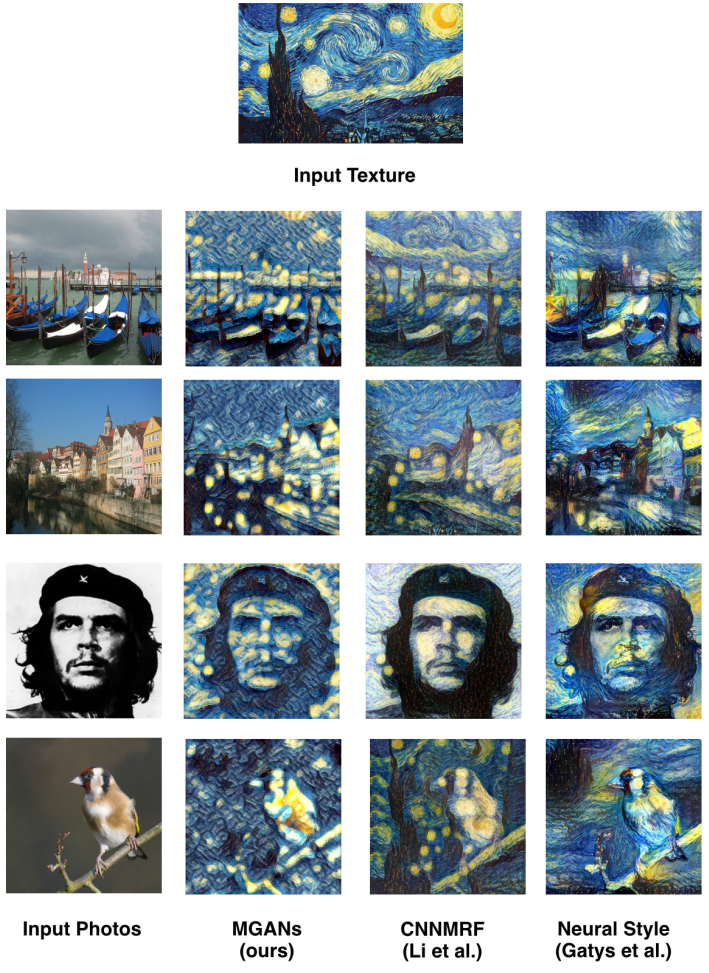

Fig. 4: This figure compares our results with deconvolutional methods [3,2]. The example texture is very stochastic and full of deformable brush strokes. In this case Gatys et al. [2] produce the best results. It also exposes the weakness of our method in dealing with backgrounds with flat color. Currently we deal with this problem by adding low frequency noise to the input image.

Fig. 5: This figure compares our results with another feed-forward method [1]. These examples show that our method is able to produce more profound foreground, while in Ulyanov et al. [1]'s results the content image has less control over the distribution of the texture, due to their model's Gaussian assumption.

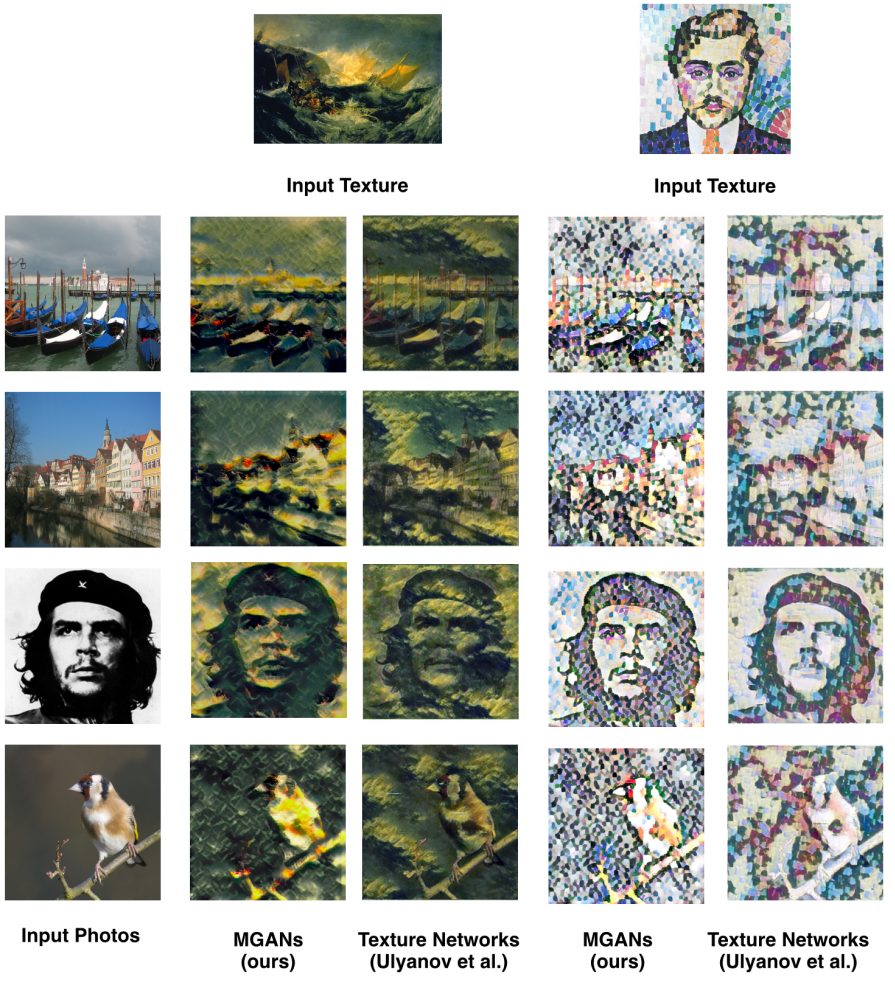

Fig. 6: This figure compares our results with another feed-forward method [1]. These examples show that Ulyanov et al. [1] match the global color of the example texture better, and produce more natural texture for flat color regions. In contrast, our method produce more paintly textures for the foregrounds. We believe better results can be generated by elaborating the strengths of these two approaches.

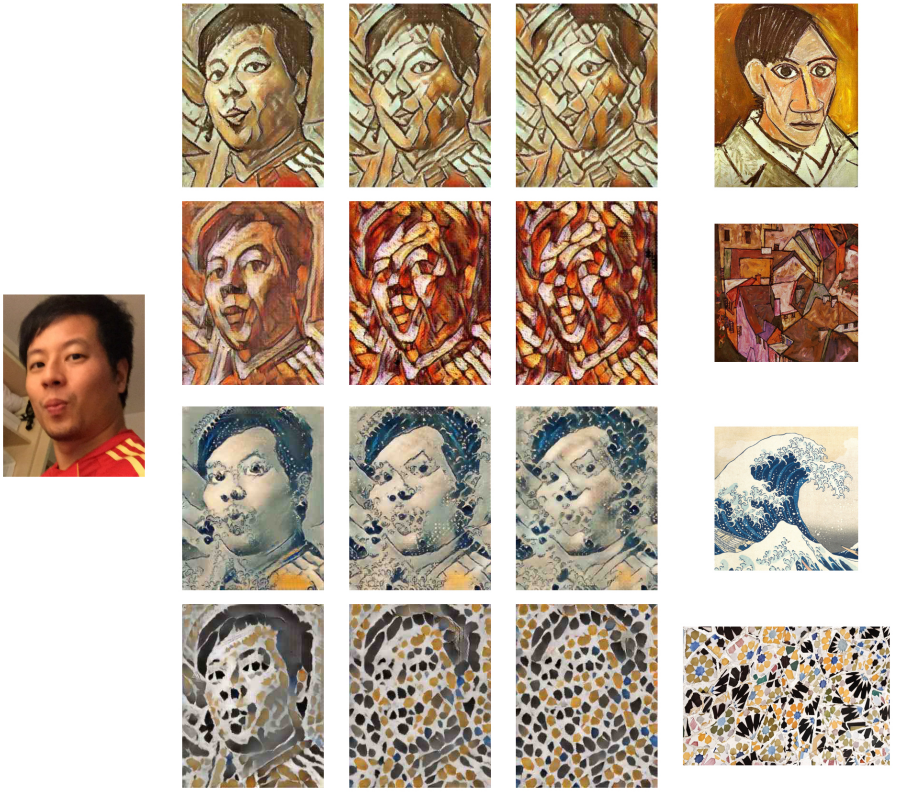

Fig. 7: This figure shows it is able to increase the stylization by recursively decoding the photo. Each picture on the right (apart from the rightmost example texture) is the decoding result of the picture on its left.