Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis

Supplementary material

1 Implementation details

Stylization. The StyleNetV2 network is similar to one from Johnson *et al.* with all batch normalization layers replaced with instance normalization. The network starts with 512×512 image and first pads it using reflection padding to have 592×592 resolution. The convolutions are not padded and residuals are added to center-cropped image (to match the spatial dimensions of residuals) resulting in 512×512 output size. The style image is first scaled to 600×600 size before passing through VGG-19 network.

We use Torch7 to implement the proposed method. We train stylization networks for 20000 iterations using Adam optimizer with learning rate of 0.001, batch size of 3 to fit in the GPU memory. The training process takes about 4 hours using NVIDIA TITAN X Maxwell.

Texture synthesis. The architecture of TextureNetV2 is presented in table 1. We use samples from uniform distribution $z \sim U(0, 1)$ as inputs to the generator network. We train it with Adam optimizer for 5000 iterations starting with learning rate of 0.001 and lowering it down by a factor of 1.5 every 750 iterations. The batch size was set to 8 and image size to 256. The training takes no more than half an hour on NVIDIA TITAN X Maxwell.

#	Dim	Layer
0	256	Input
1	256	Linear
2	256	Linear
3	$16 \times 4 \times 4$	Reshape
4	$128 \times 8 \times 8$	FullConvolution 3×3 + BN + ReLU
5	$128 \times 16 \times 16$	FullConvolution 3×3 + BN + ReLU
6	$128 \times 32 \times 32$	FullConvolution 3×3 + BN + ReLU
7	$64 \times 64 \times 64$	Bilinear UpSampling + Convolution 3×3 + BN + ReLU
8	$32 \times 128 \times 128$	Bilinear UpSampling + Convolution 3×3 + BN + ReLU
9	$3 \times 256 \times 256$	Bilinear UpSampling + Convolution 3×3 + BN + ReLU

Table 1: TextureNetV2 architecture. The fully-connected layers at the start ensure huge receptive field.

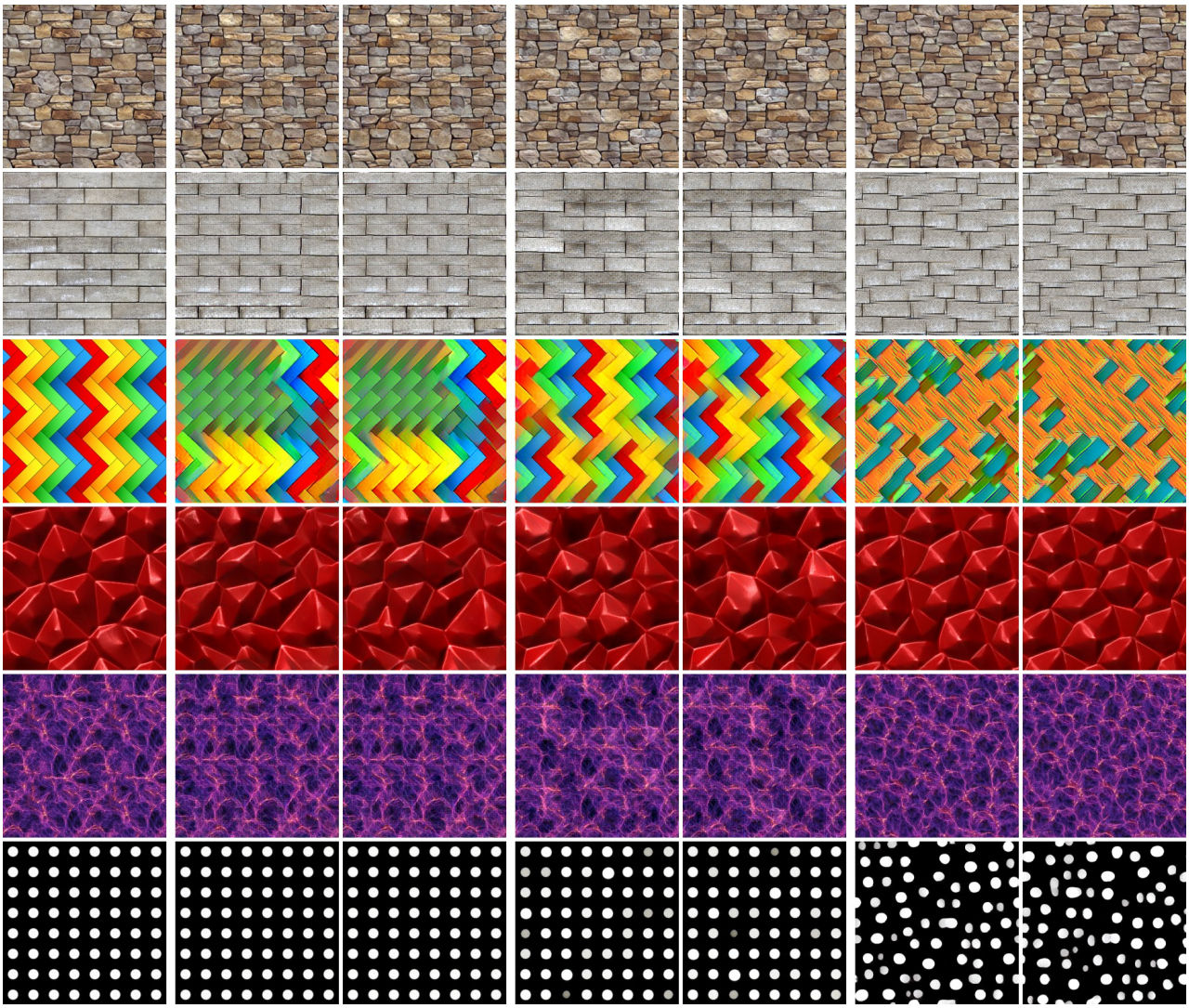


Figure 1: Additional examples for Fig.5 of the main paper.

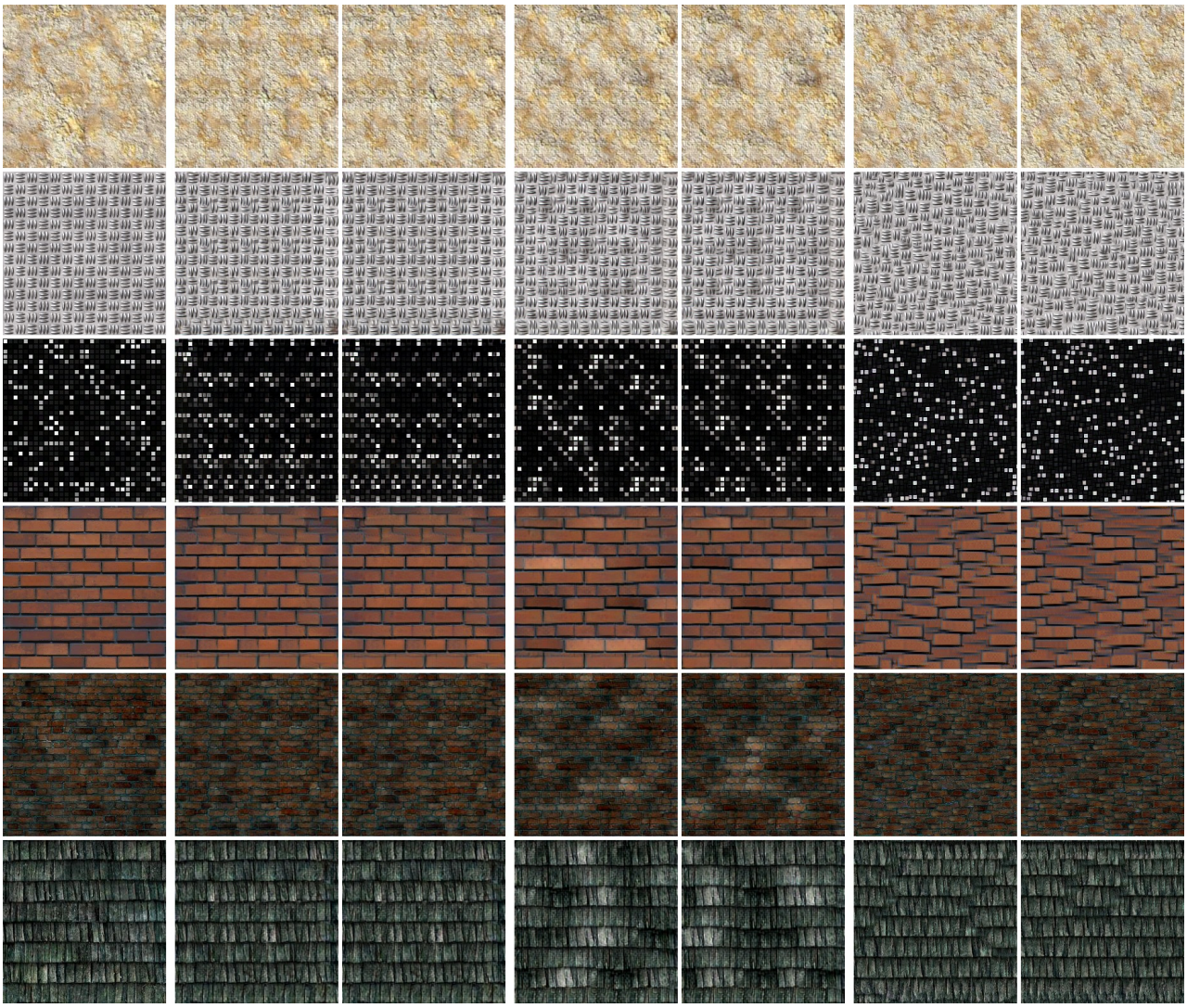
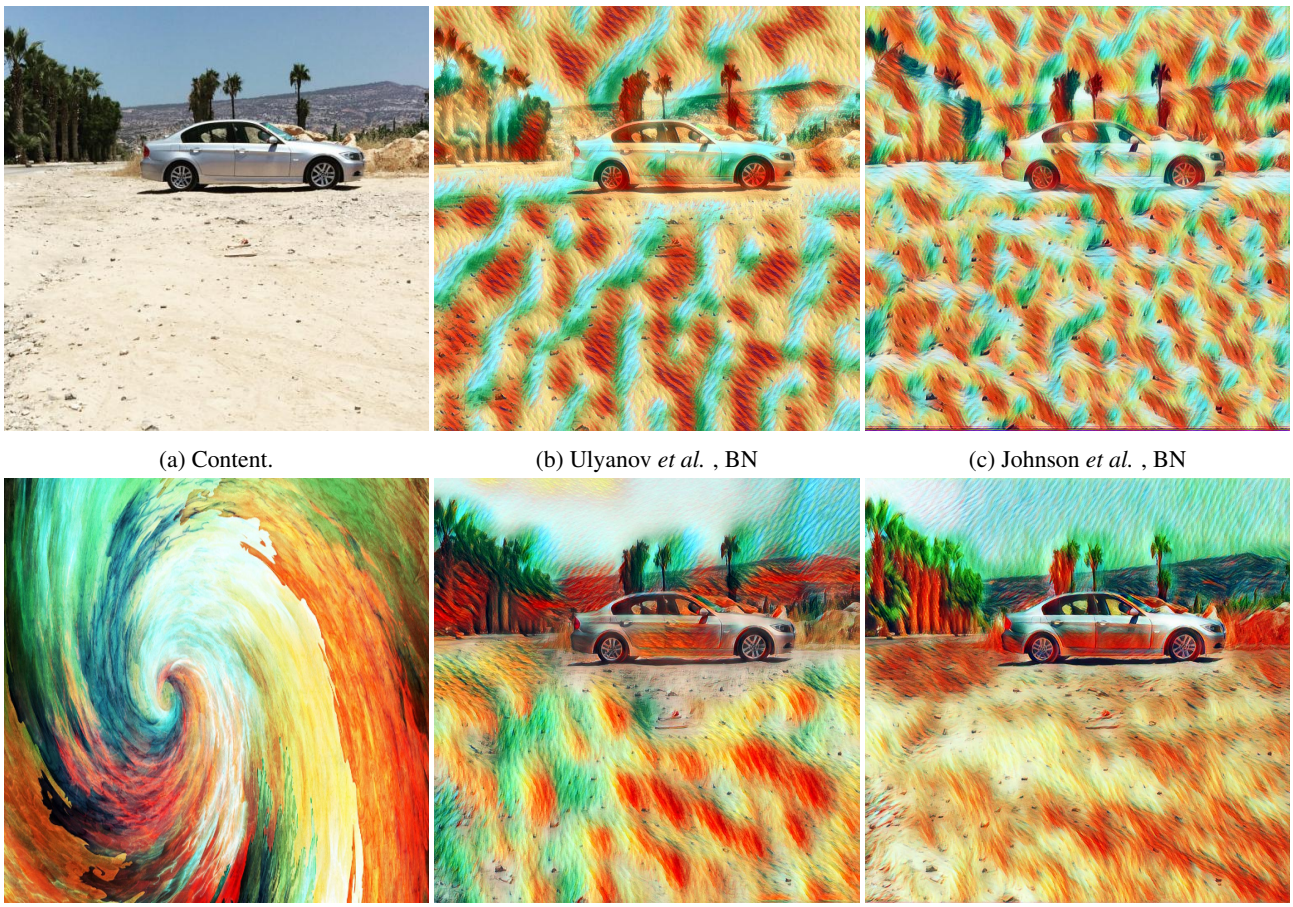


Figure 2: Additional examples for Fig.5 of the main paper.

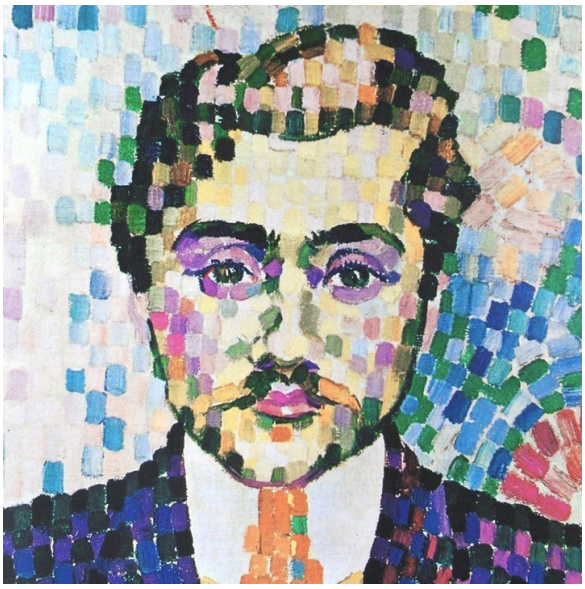


(a) Content. (b) Ulyanov *et al.* , BN (c) Johnson *et al.* , BN

(d) Style. (e) Ulyanov *et al.* , IN (f) Johnson *et al.* , IN.

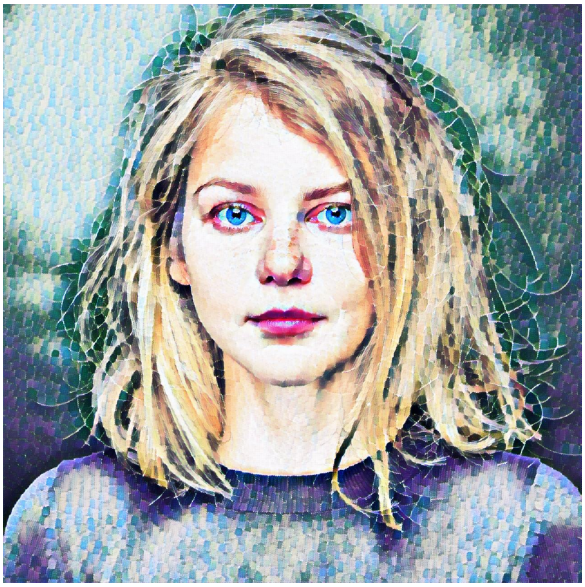
Figure 3: Qualitative comparison of generators proposed in Ulyanov *et al.* and Johnson *et al.* with batch normalization (BN) and instance normalization (IN). Both architectures benefit from instance normalization.

(a) Content.



(b) Style.

(c) Image size 512×512.



(d) Image size 1080×1080.

Figure 4: Processing a content image with StyleNet IN at different resolutions: 512 (c) and 1080 (d).

Figure 5: Content images for next four figures.

Figure 6: StyleNet IN astylization examples, part 1. The content images are given in fig. 5. Style images are shown in the first row.

Figure 7: StyleNet IN astylization examples, part 2. The content images are given in fig. 5. Style images are shown in the first row.

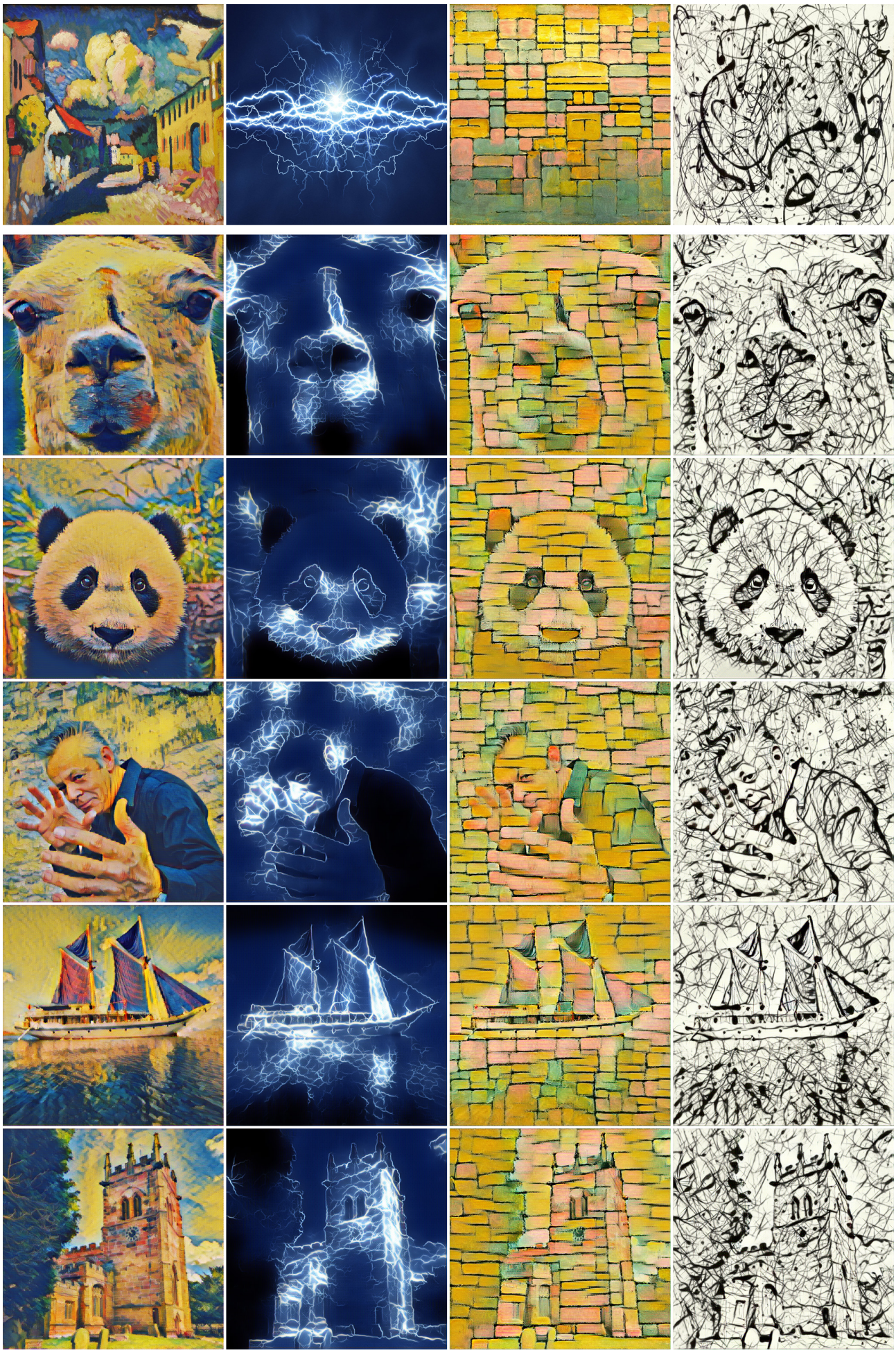


Figure 8: StyleNet IN astylization examples, part 3. The content images are given in fig. 5. Style images are shown in the first row.

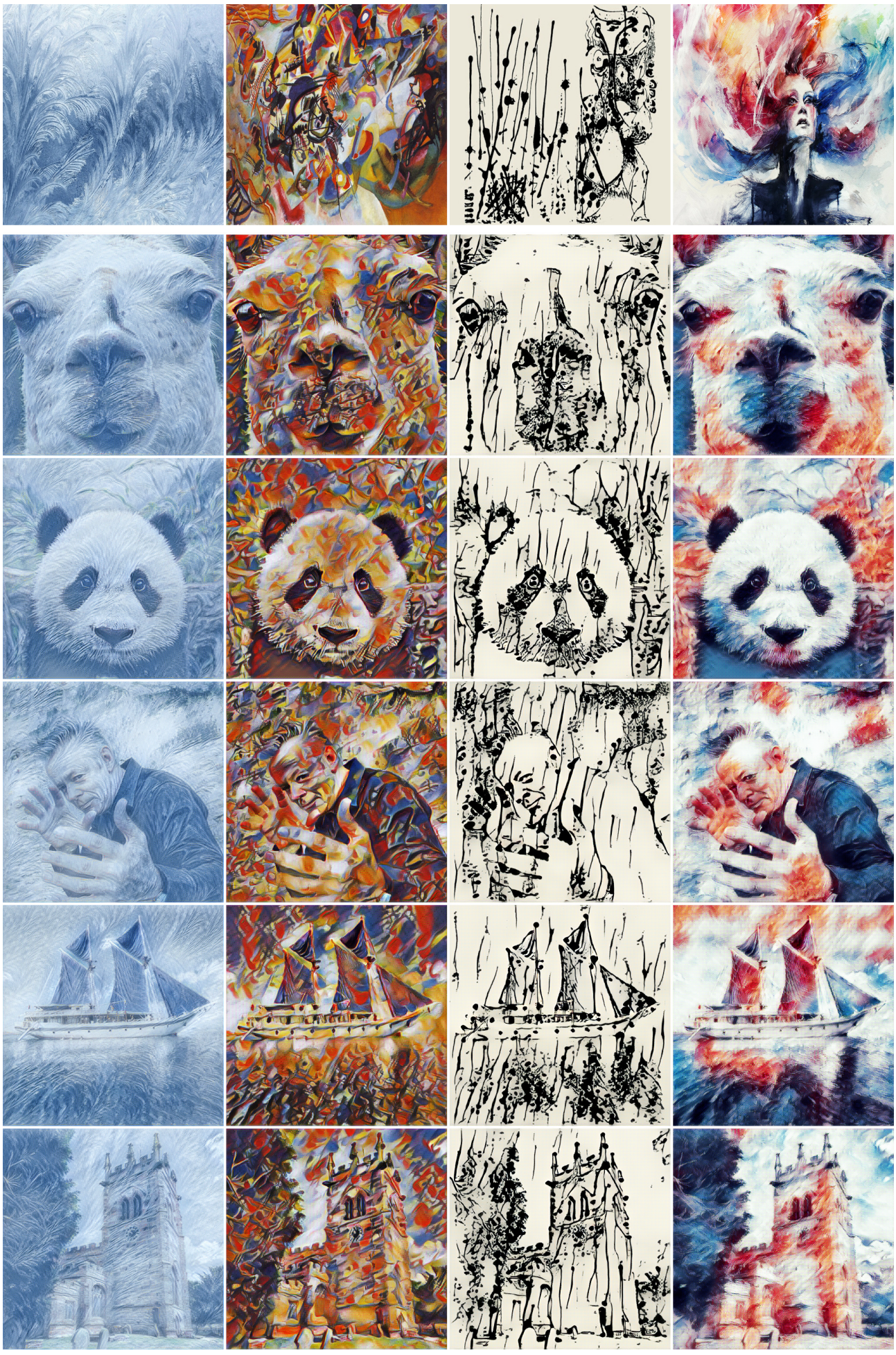


Figure 9: StyleNet IN astylization examples, part 4. The content images are given in fig. 5. Style images are shown in the first row.

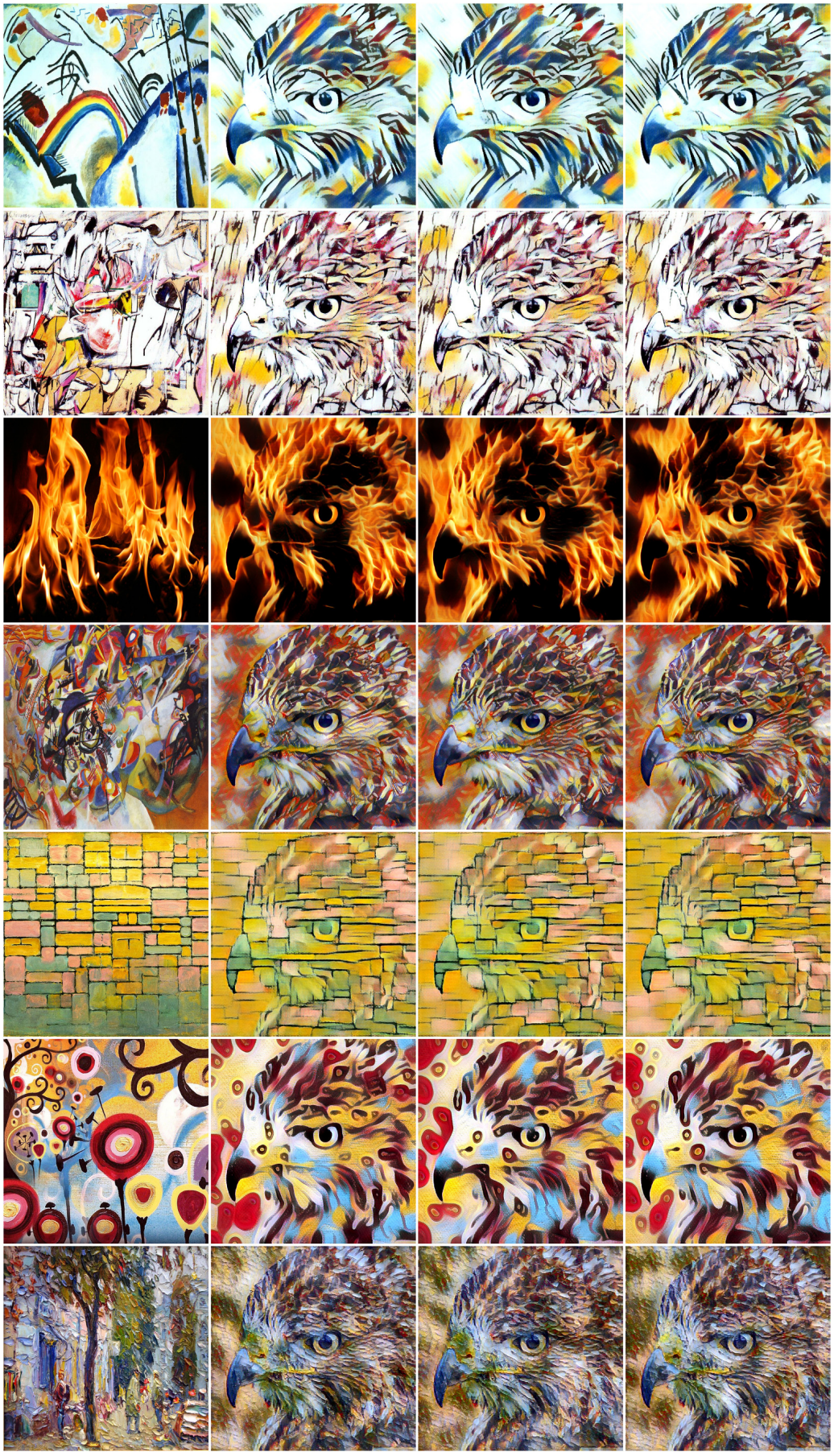


Figure 10: Style (left column) and three stylizations obtained with StyleNet IN trained with diversity loss.

Figure 11: An effect from concatenating noise to image. Left: content image, middle: stylization with generator whose input is formed by concatenating content image and noise, right: use only content image as input. We observe that flat regions are not stylized with 'no noise' generator. Style image can be found in fig. 10.