1 Implementation details

Stylization. The StyleNetV2 network is similar to one from (Johnson *et al.*, 2016) with all batch normalization layers replaced with instance normalization. Three channels of $U(0,1)$ noise are concatenated to each image beforehand, so an input tensor has dimensionality `Bx6xHxW`. 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 used the same VGG loss setup as in methods we compare to. We used `relu4_2` layer features to compute content loss and `relu1_2`, `relu2_2`, `relu3_2`, `relu4_2` layers for style loss. Content loss weight was fixed to 1 while we slowly annealed style loss from 0 to 100 and picked the most visually pleasing checkpoint. This way we did not need to perform a grid search for a good alpha.

We used Torch7 to implement the proposed method. We trained stylization networks for 20000 iterations using Adam optimizer with learning rate of 0.001 and batch size of 3 to fit in the GPU memory. The training process takes about 4 hours using NVIDIA TITAN X Maxwell. At run-time, IN approximately as fast as BN, so run-time complexity of StyleNet IN is approximately the same as StyleNet BN, which, in turn, has the same complexity as the method of (Johnson *et al.*, 2016).

Texture synthesis. The architecture of TextureNetV2 is presented in table 1. We used samples from uniform distribution $z \sim U(0,1)$ as inputs to the generator network. We trained 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.

2 Additional examples

More examples are available at the project page https://dmitryulyanov.github.io/texture_nets_v2.

Figure 1: Textures, used for fig. 2.

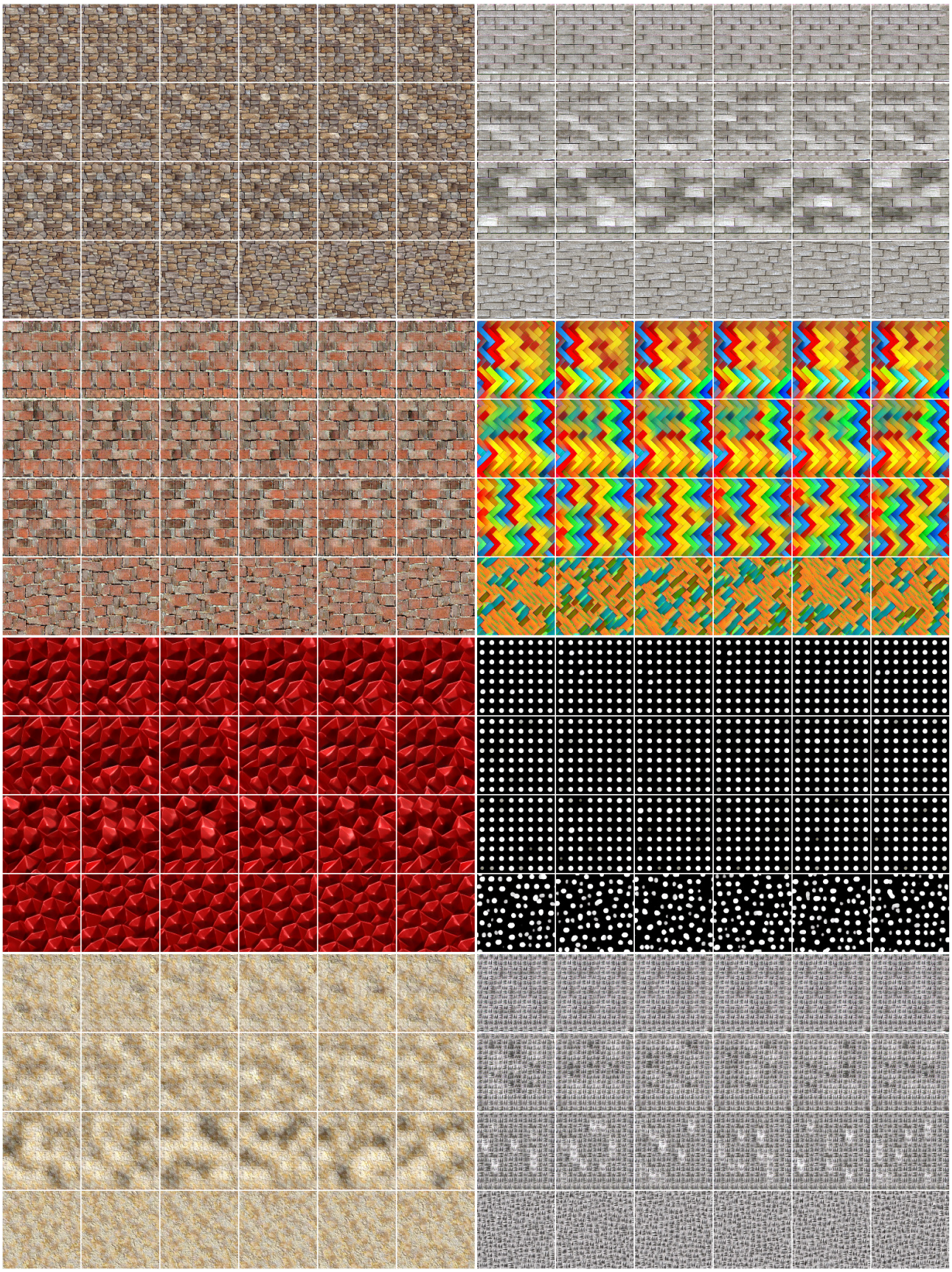
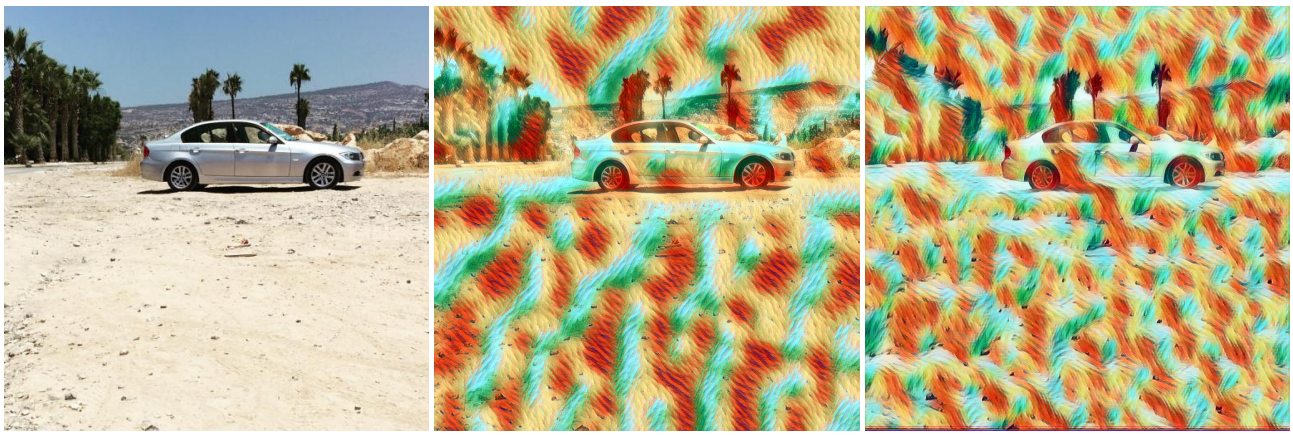


Figure 2: An effect of changing diversity parameter λ. For each texture first three rows show TextureNetV2 results for $\lambda = 5, 10, 15$; row four shows textures generated with TextureNetV1.

StyleNet IN, 512px | StyleNet BN, 512px | StyleNet IN, 256px | StyleNet BN, 256px (Johnson *et al.*, 2016)

Figure 3: We use 512×512 images to train StyleNet IN and compare to StyleNet BN trained on 512×512 images to emphasize the value of IN (columns one and two, same as in fig. 4 of main paper). This is different to (Ulyanov *et al.*, 2016) and (Johnson *et al.*, 2016) where 512×512 resolution was used. We show that IN also helps when 256×256 resolution is used (columns three and four). Methods of (Ulyanov *et al.*, 2016) and (Johnson *et al.*, 2016) differ only in generator structure and produce similar results. For that reason we compare only with (Johnson *et al.*., 2016).

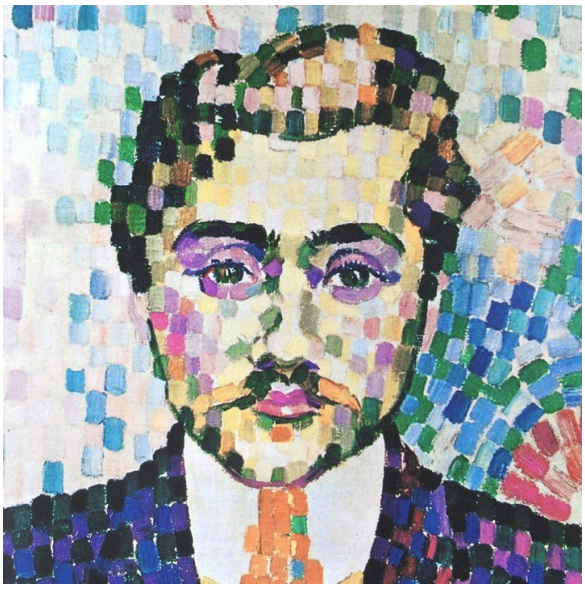


(a) Content. (b) Ulyanov *et al.*, BN (c) Johnson *et al.*, BN

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

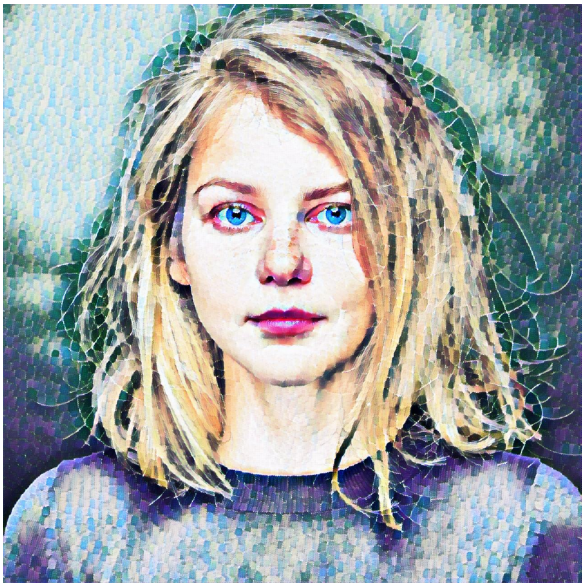
Figure 4: Qualitative comparison of generators proposed in (Ulyanov *et al.*, 2016) and (Johnson *et al.*, 2016) with batch normalization (BN) and instance normalization (IN). Both architectures benefit from instance normalization.

(a) Content.



(b) Style.

(c) Image size 512 × 512.



(d) Image size 1080 × 1080.

Figure 5: Processing a content image with StyleNet IN at different resolutions: 512 (c) and 1080 (d).

Figure 6: Content images for next four figures.

Figure 7: StyleNet IN astylization examples, part 1. The content images are given in fig. 6. Style images are shown in the first row.

Figure 8: StyleNet IN astylization examples, part 2. The content images are given in fig. 6. Style images are shown in the first row.

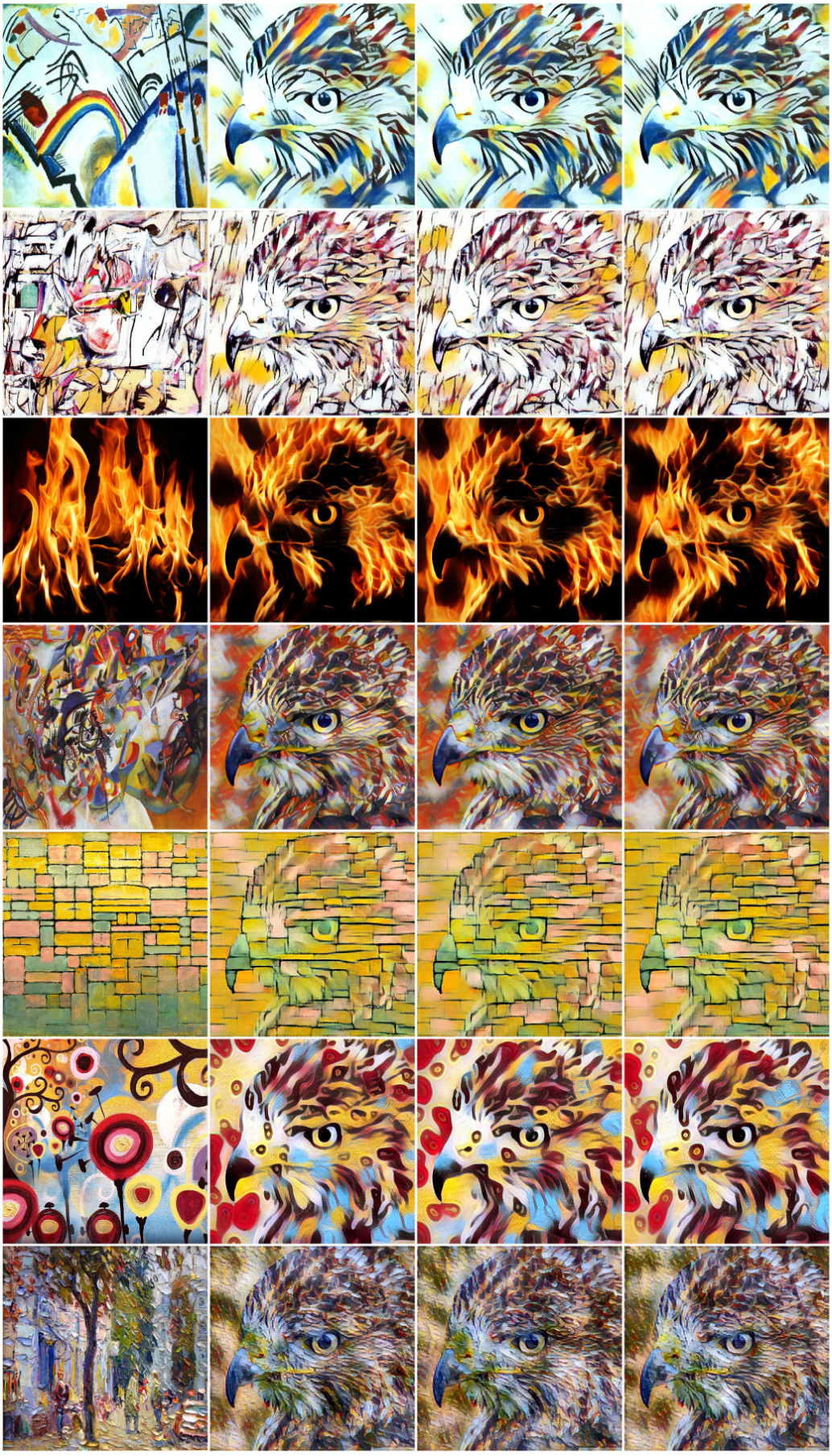


Figure 9: Style (left column) and three stylizations obtained with StyleNet IN trained with diversity loss.