Learning-Rate Decay for Generator

Mitch Hill, Erik Nijkamp

September 1, 2017

Abstract

The training of CoopNet on 32 × 32 image patches is non-trivial. The descriptor net appears to synthesize realistic and sharp images easily. The training seems relatively insensitive to varying learning parameters. In contrast, synthesis of sharp images by the generator net is hard. The training is very sensitive to variations in the parameters. We aim to identify relations between learning-rate and 'sharpness' from which we may formulate decay regimes.

Result

exp	approach	gamma2	loss
(1)	step-wise decay	(0.00035 : 100, 0.000035 : 200, 0.0000035 : 100)	unknown
(2)	step-wise decay ×10	$(0.0005 : 100, 0.0001 : 100, 0.00005 : 100, 0.00001 : 100, 0.000005 : 100) \cdot 10$	0.050311
(3)	step-wise decay ×2	$(0.0005 : 100, 0.0001 : 100, 0.00005 : 100, 0.00001 : 100, 0.000005 : 100) \cdot 2.0$	0.066391
(4)	step-wise decay ×1	$(0.0005 : 100, 0.0001 : 100, 0.00005 : 100, 0.00001 : 100, 0.000005 : 100) \cdot 1.0$	0.073952
(5)	step-wise decay ×.5	$(0.0005 : 100, 0.0001 : 100, 0.00005 : 100, 0.00001 : 100, 0.000005 : 100) \cdot 0.5$	0.085113
(6)	step-wise decay ×.1	$(0.0005 : 100, 0.0001 : 100, 0.00005 : 100, 0.00001 : 100, 0.000005 : 100) \cdot 0.1$	0.11982
(7)	(2) increased	$(0.0003 : 100, 0.0001 : 100, 0.00005 : 100, 0.00005 : 100, 0.00001 : 100) \cdot 10$	0.044032
(8)	step-wise decay ×20	$(0.0005 : 100, 0.0001 : 100, 0.00005 : 100, 0.00001 : 100, 0.000005 : 100) \cdot 20$	0.053683
(9)	log-space decay	$0.0005 \cdot logspace(-2, -3, 500) \cdot 100$	0.059591
(10)	log-space increased	$0.001 \cdot logspace(-2, -3, 500) \cdot 100$	0.052407
(11)	log-space long-tail	$0.001 \cdot logspace(-2, -4, 500) \cdot 100$	0.064684
(12)	lin-space decay	$0.0005 \cdot linspace(1, 0.01, 500)$	0.054532

Table 1: Experiments and preliminary results.

Parameters

```
config.nIteration = 500;
config.batchSize = 32;

% sampling parameters
config.num_syn = 32;

% descriptor net1 parameters
config.Delta = 0.3;
config.Gamma = [0.0005*ones(1,100), 0.00005*ones(1,100), 0.00001*ones(1,100), 0.000005*
    ones(1,100), 0.000001*ones(1,100)];
config.refsig = 1;
config.T = 40;

% generator net2 parameters
config.Delta2 = 0.3;
config.refsig2 = 1;
config.s = 0.3;
config.real_ref = 1;
config.cap2 = 8;

% how many layers to learn
config.layer_to_learn = 1;

% image size
config.im_size = 32;
```

Image patches from original texture

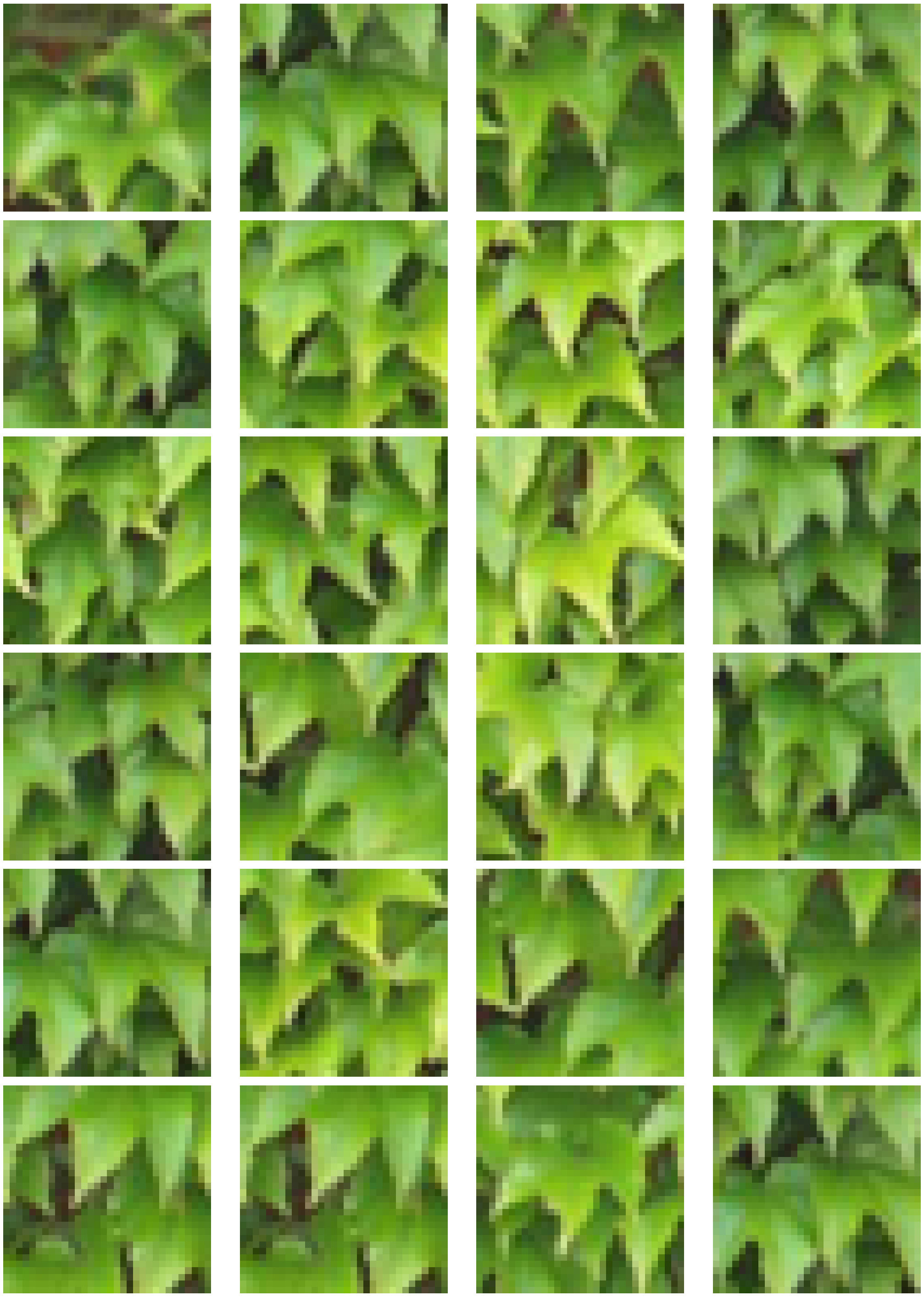

Table 2: Image patches sampled from original texture.

(1) exp_texture_512: step-wise decay - initial

```
config.Gamma2 = [0.00035*ones(1,100), 0.000035*ones(1,200), 0.0000035*ones(1,100)];
```

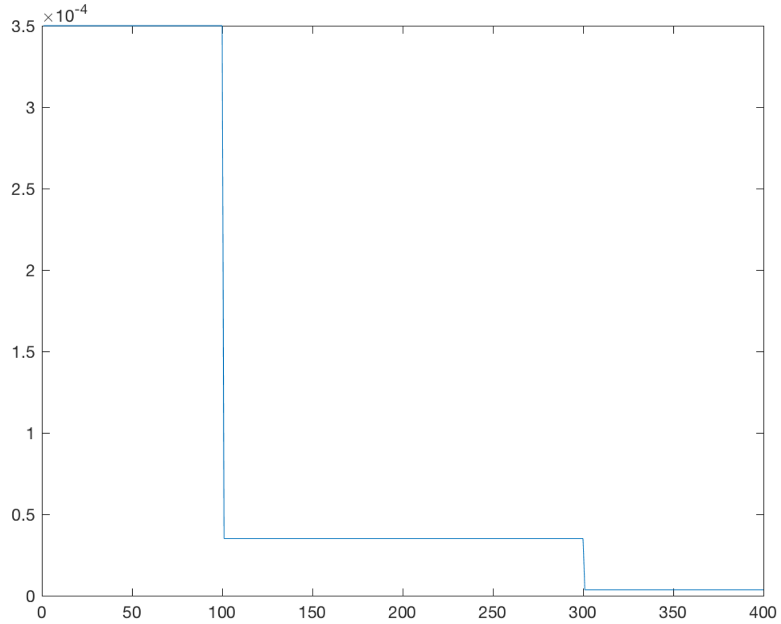

Figure 1: Gamma2.

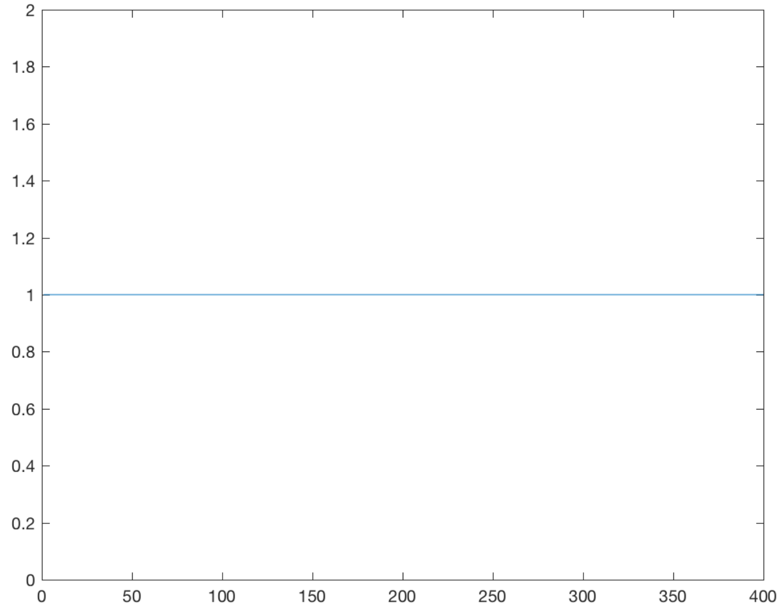

Figure 2: Loss (missing).

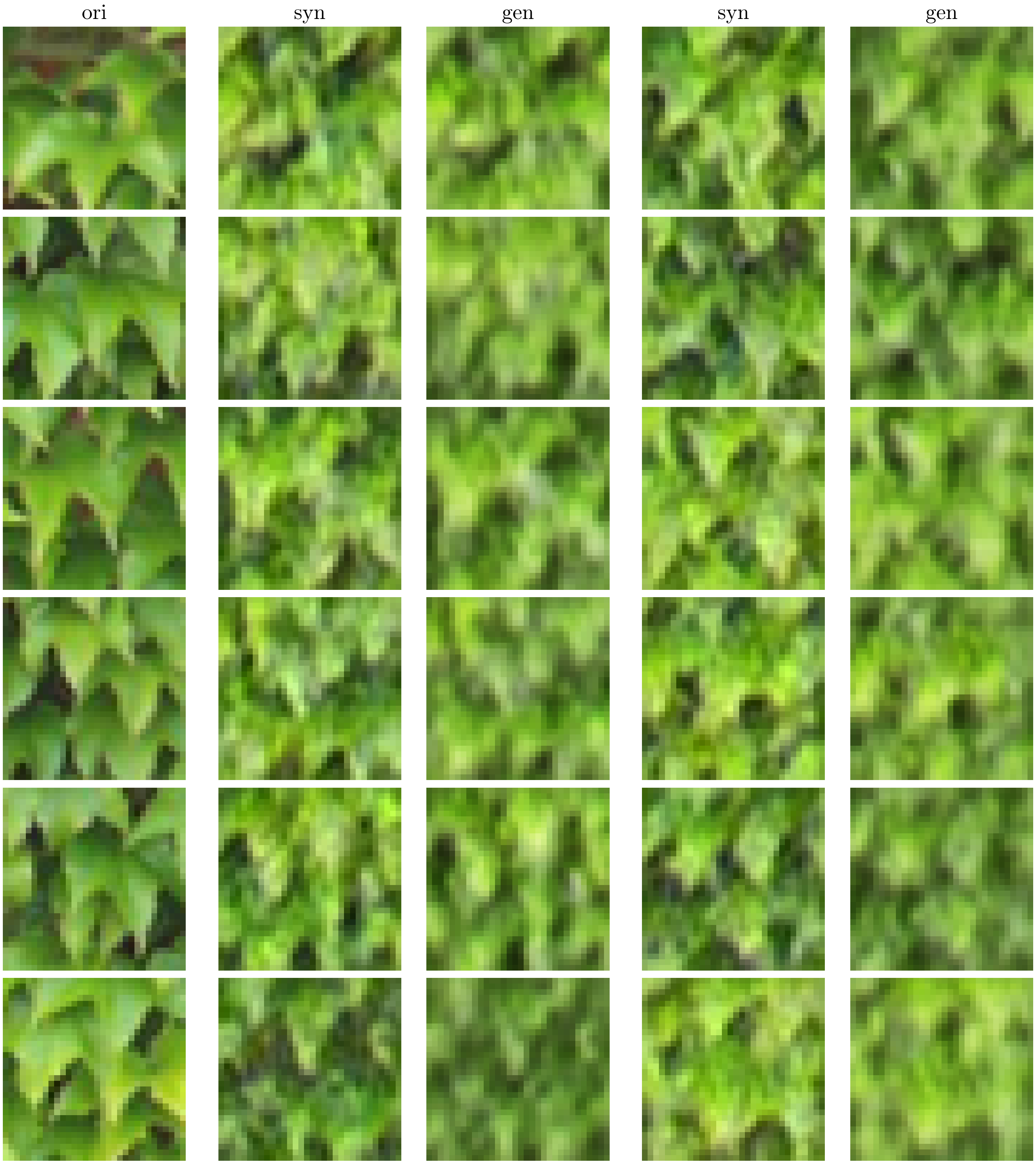

Table 3: Descriptor and generator images

(2) exp_texture_512_2: step-wise decay - ×10.0

```
config.Gamma2 = [0.0005*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00001*ones
    (1,100), 0.000005*ones(1,100)] * 10;
```

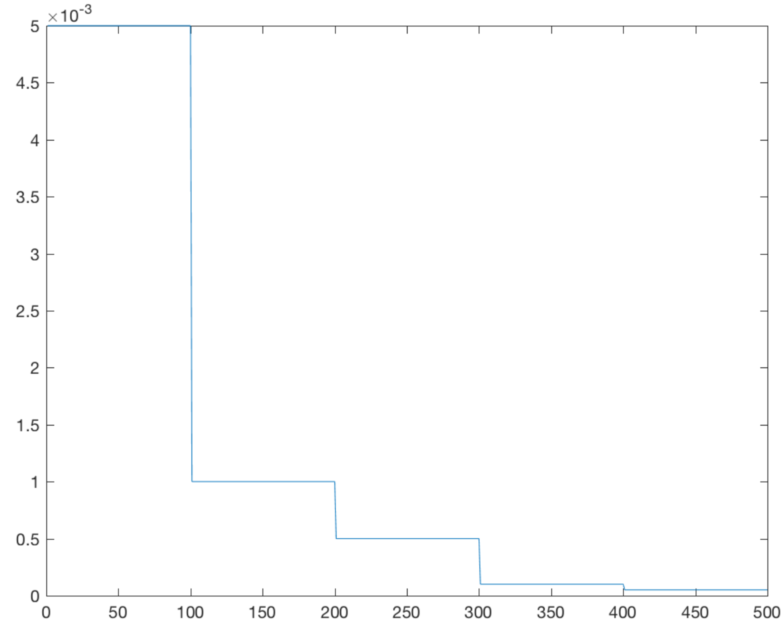

Figure 3: Gamma2.

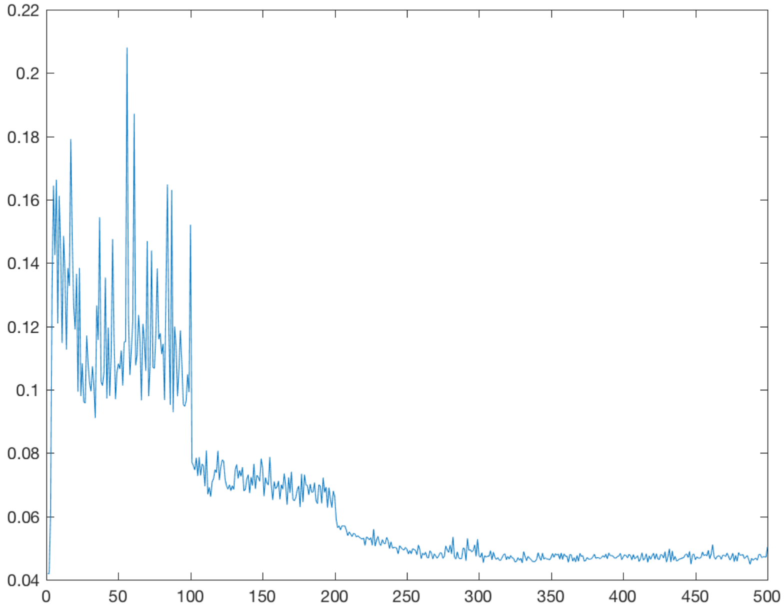

Figure 4: Loss.

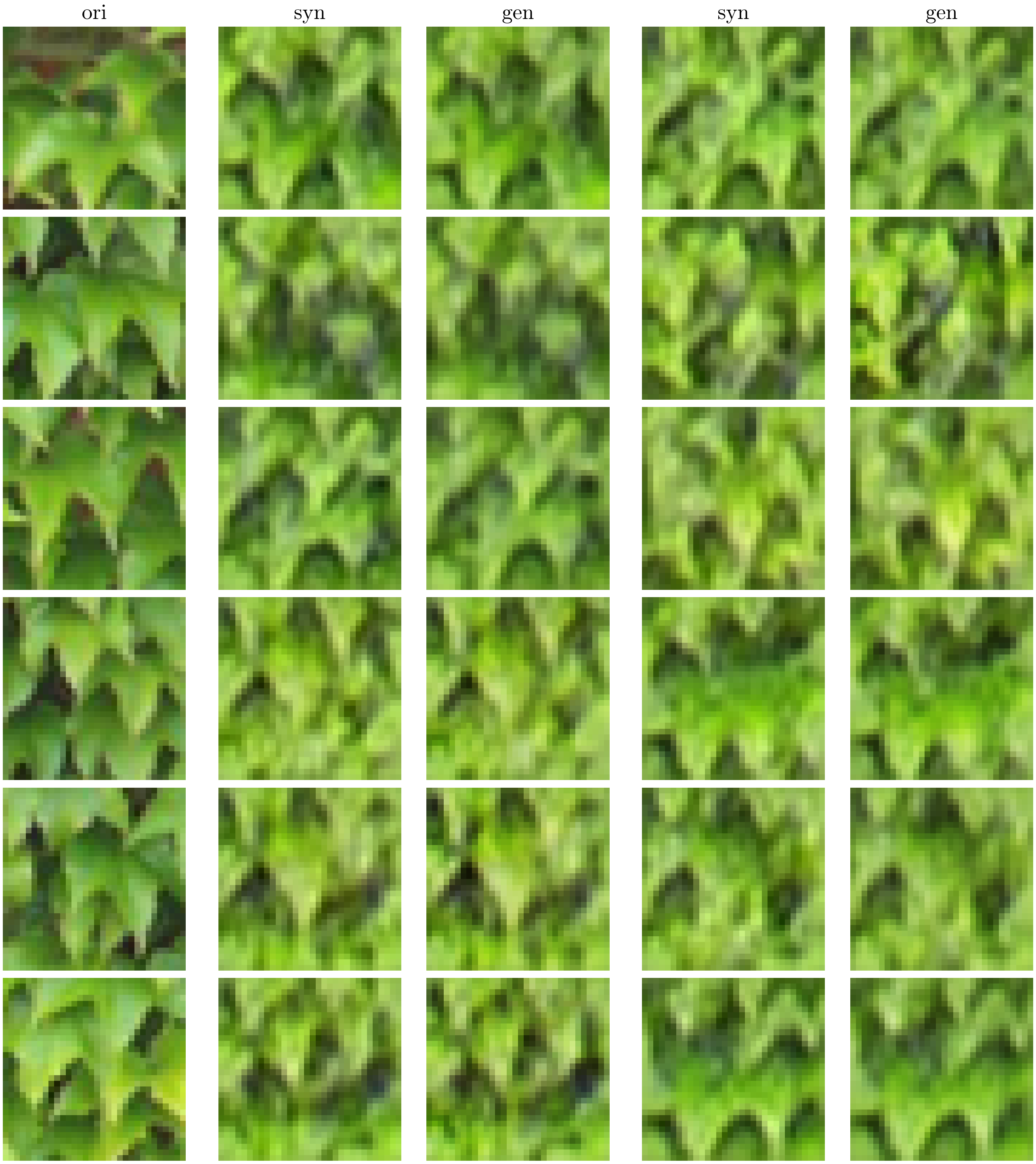

Table 4: Descriptor and generator images

(3) exp_texture_512_3: step-wise decay - ×2.0

```
config.Gamma2 = [0.0005*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00001*ones
    (1,100), 0.000005*ones(1,100)] * 2;
```

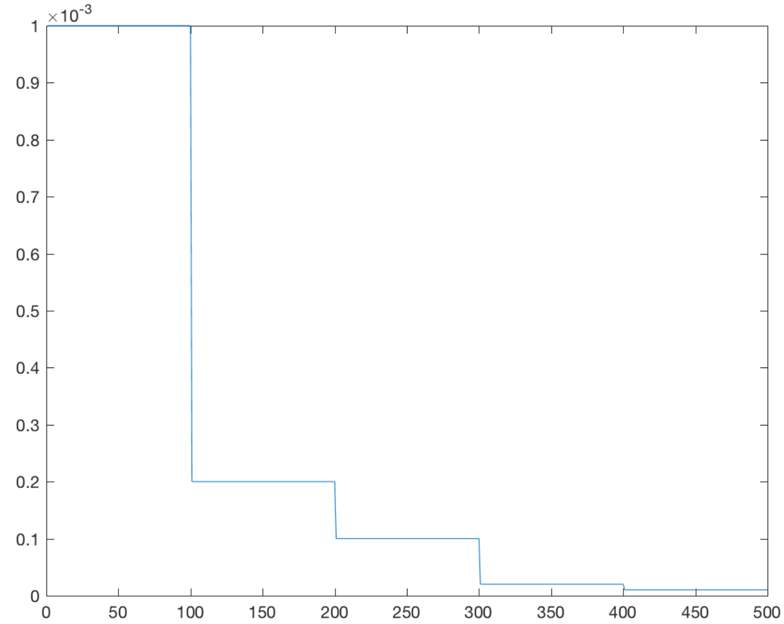

Figure 5: Gamma2.

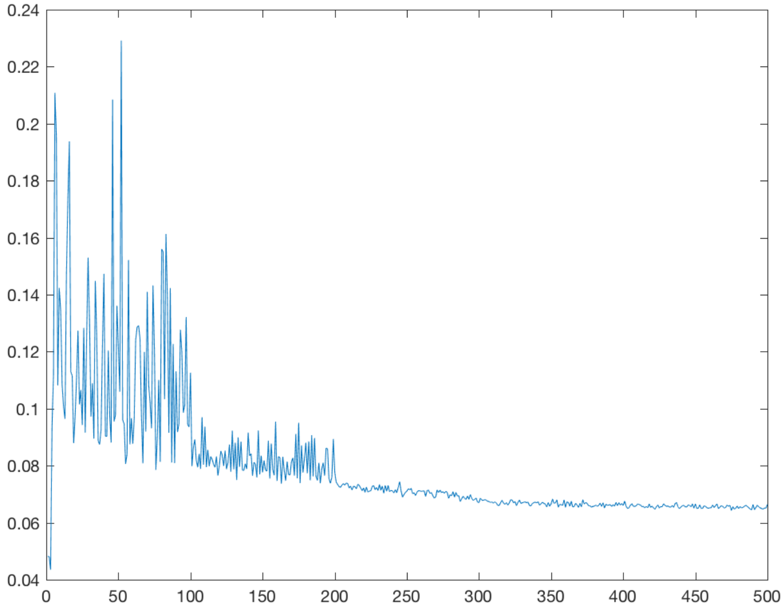

Figure 6: Loss.

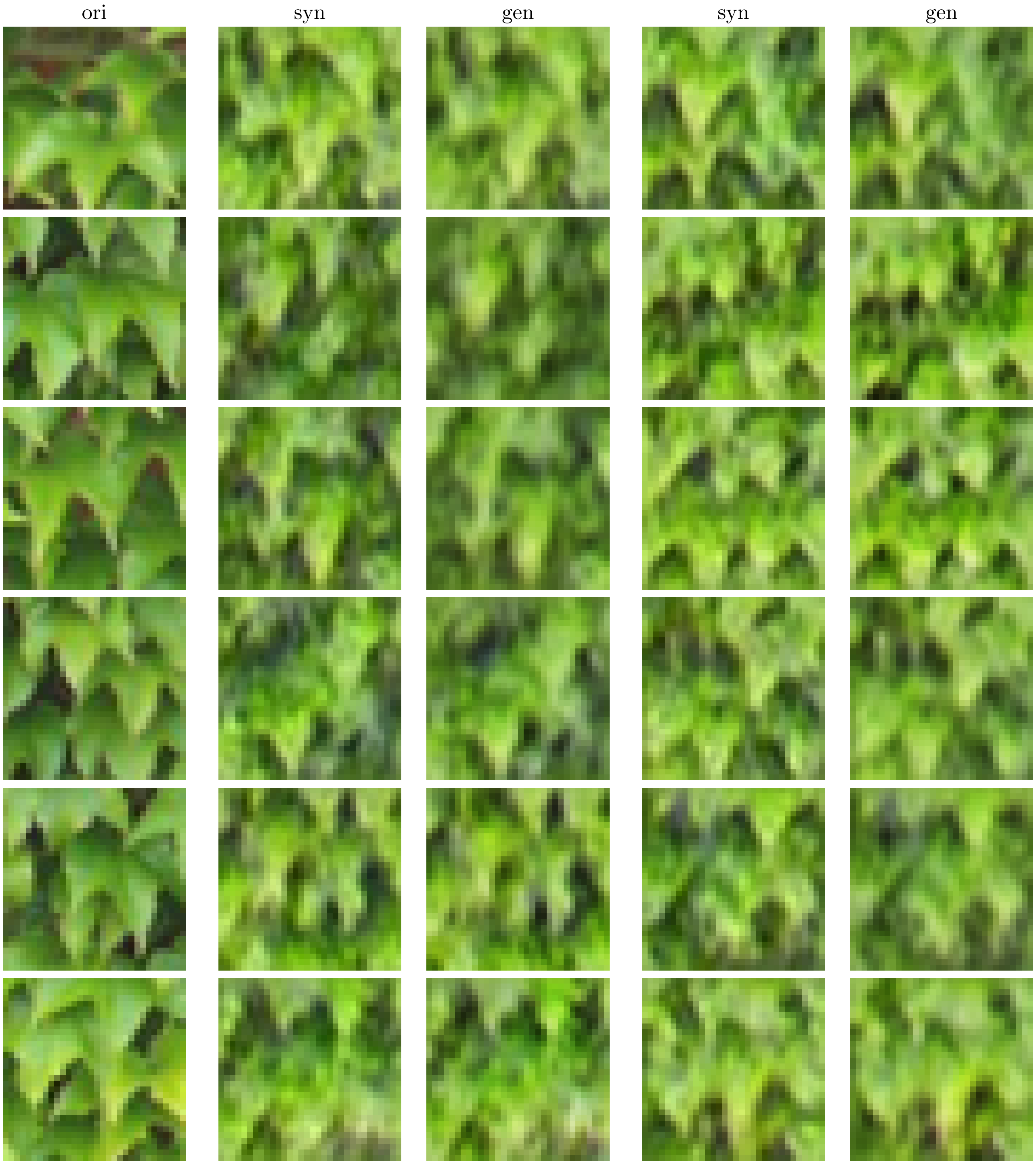

Table 5: Descriptor and generator images

(4) exp_texture_512_4: step-wise - ×1.0

```
config.Gamma2 = [0.0005*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00001*ones
    (1,100), 0.000005*ones(1,100)] * 1;
```

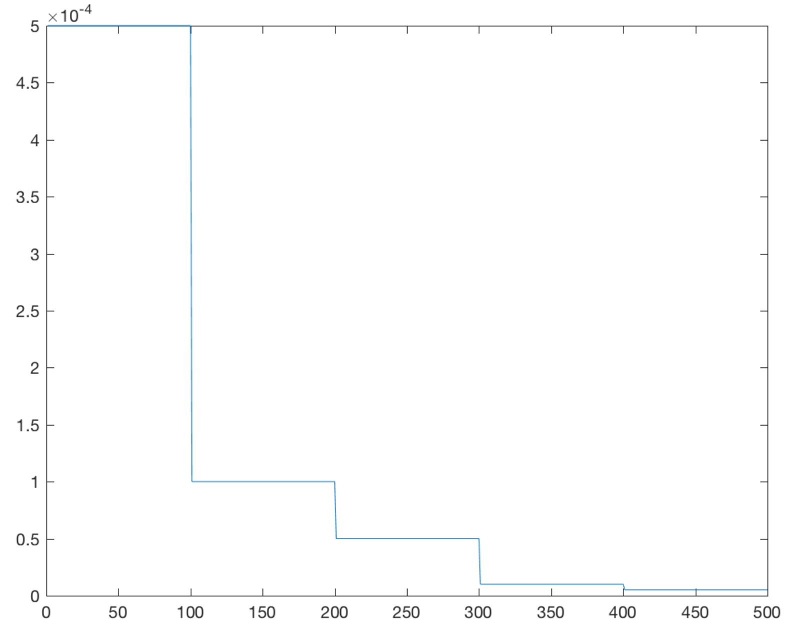

Figure 7: Gamma2.

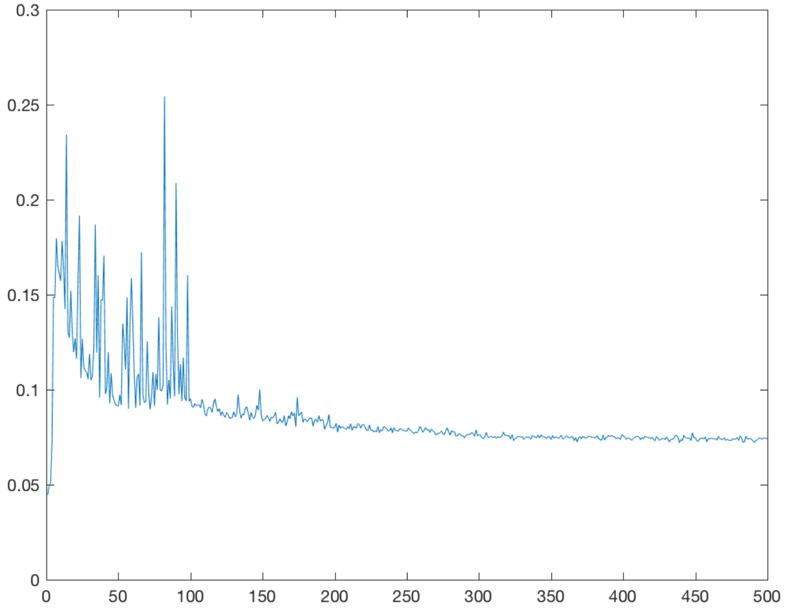

Figure 8: Loss.

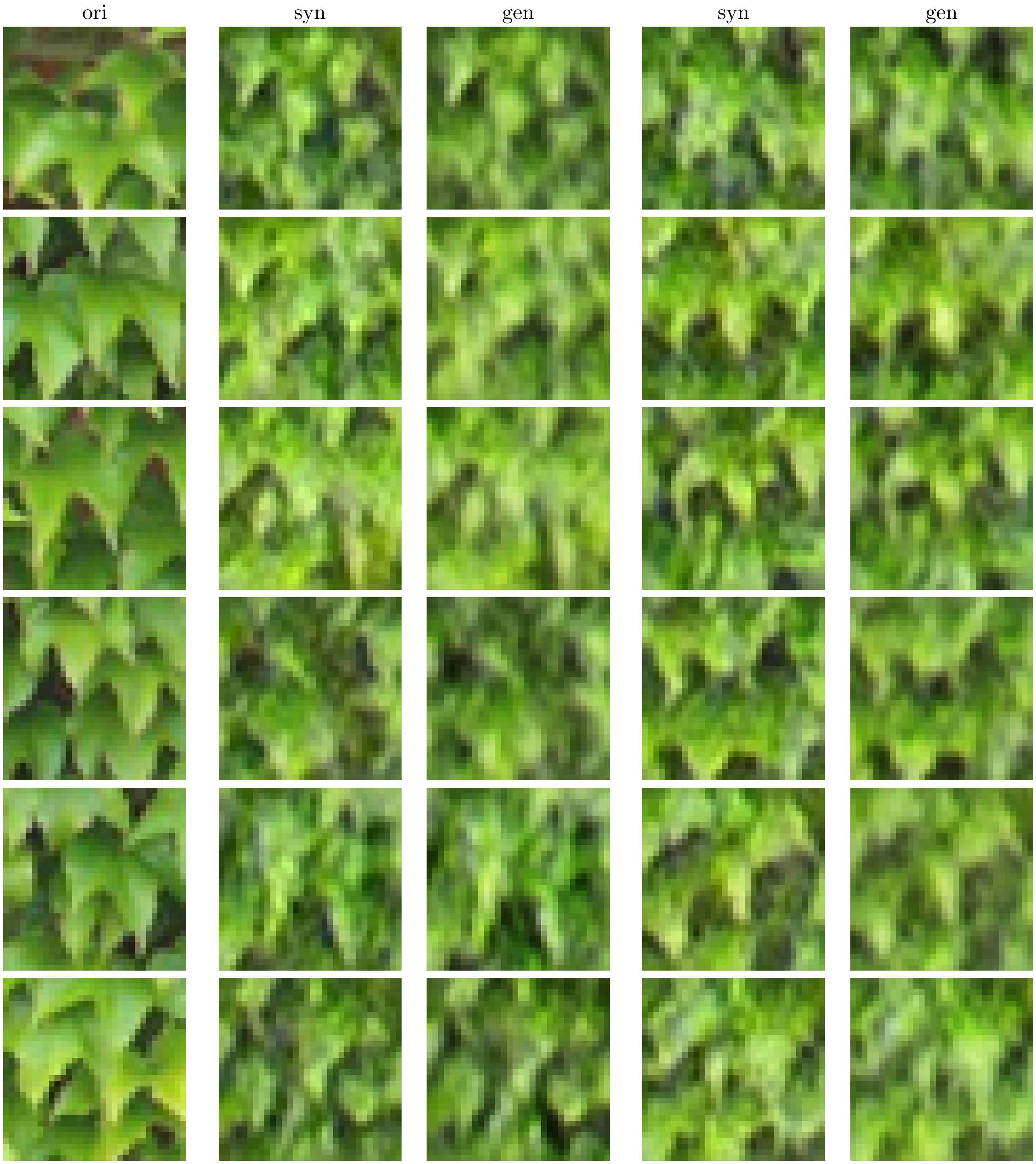

Table 6: Descriptor and generator images

(5) exp_texture_512_5: step-wise - ×0.5

```
config.Gamma2 = [0.0005*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00001*ones
    (1,100), 0.000005*ones(1,100)] / 2;
```

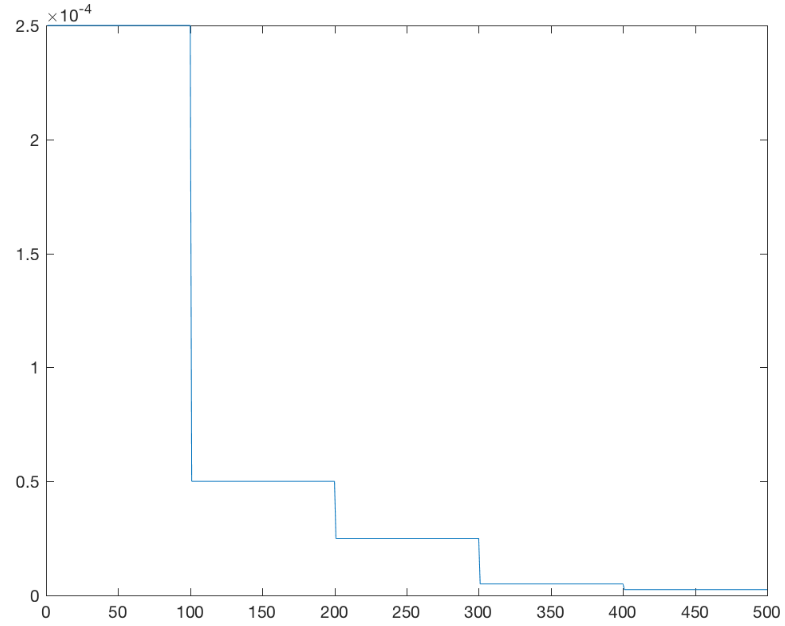

Figure 9: Gamma2.

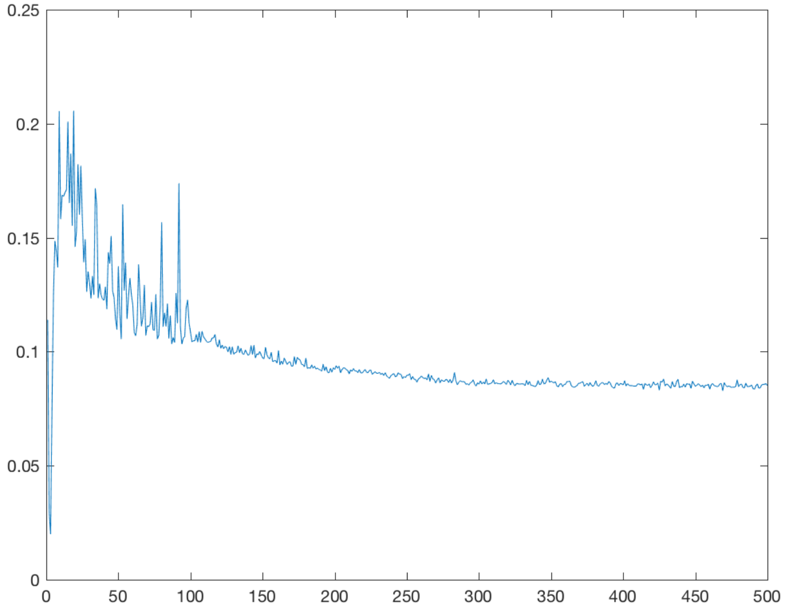

Figure 10: Loss.

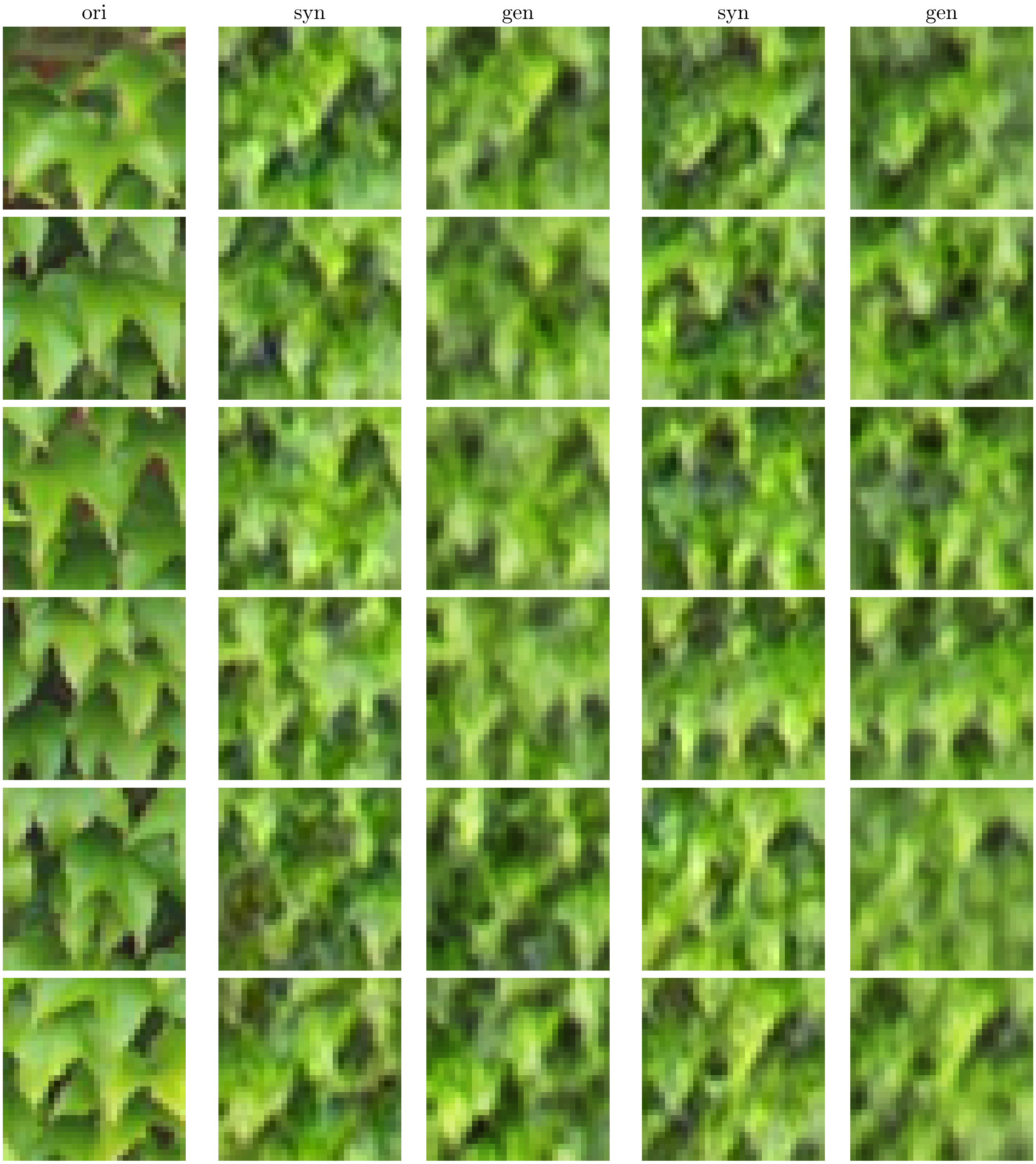

Table 7: Descriptor and generator images

(6) exp_texture_512_6: step-wise - ×0.1

```
config.Gamma2 = [0.0005*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00001*ones
    (1,100), 0.000005*ones(1,100)] / 10; % decreased
```

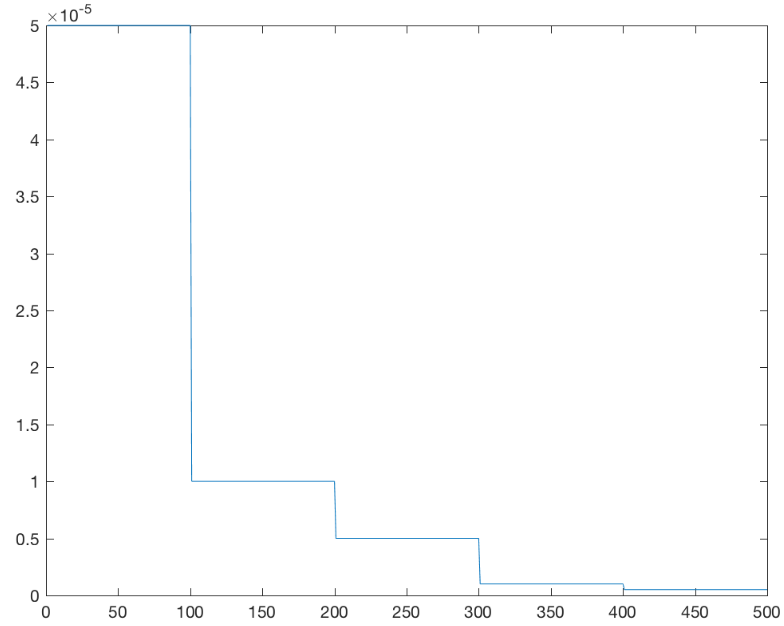

Figure 11: Gamma2.

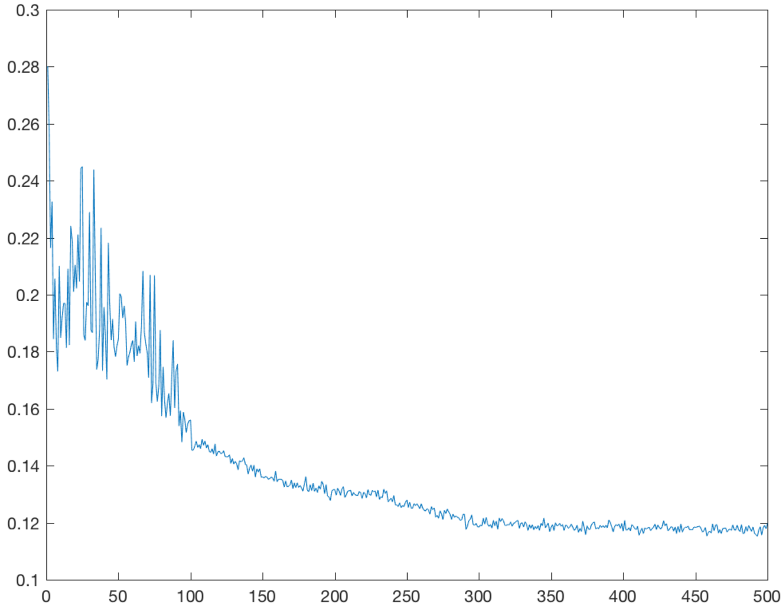

Figure 12: Loss.

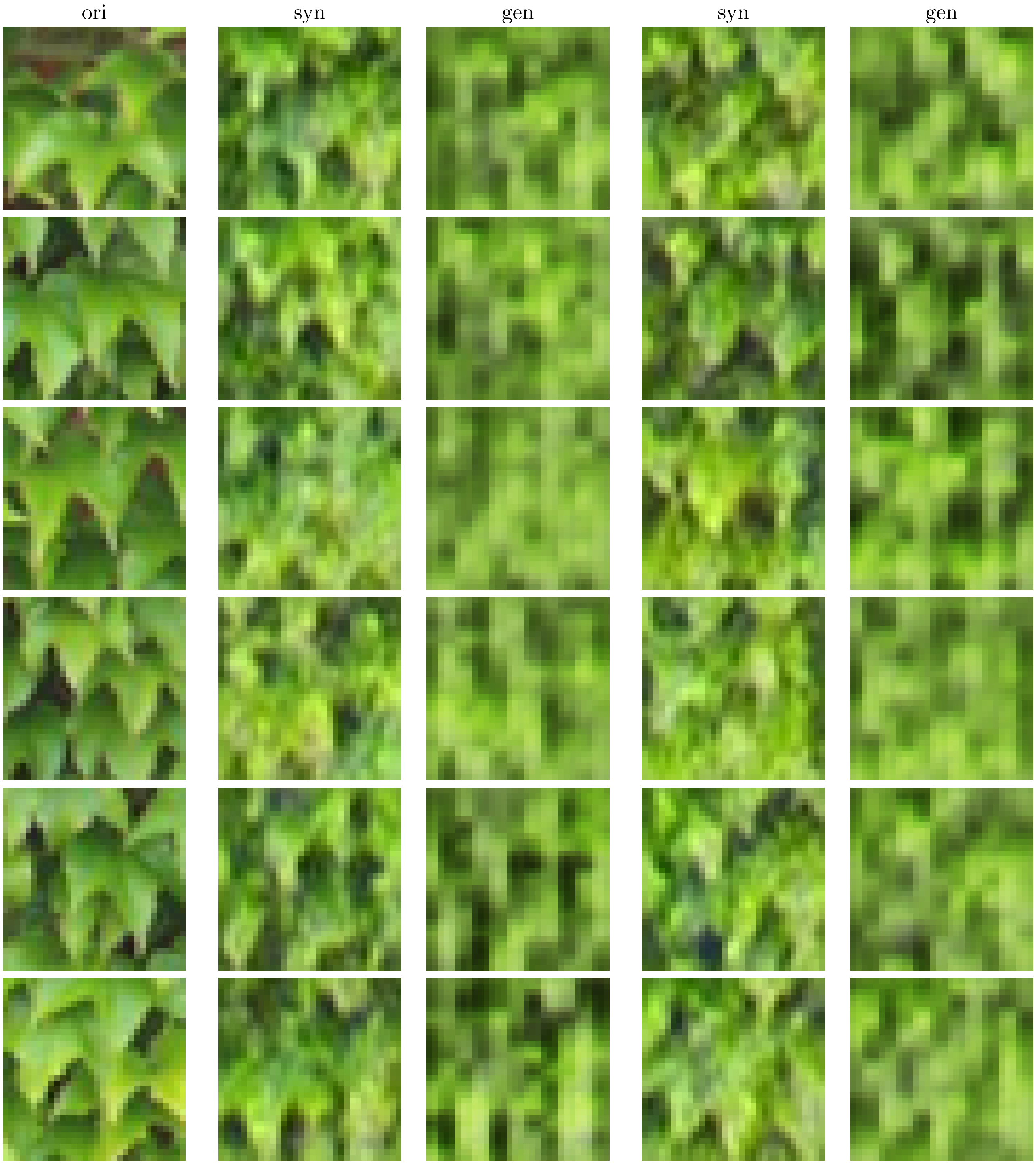

Table 8: Descriptor and generator images

(7) exp_texture_512_7: step-wise - (2) increased middle

```
config.Gamma2 = [0.0003*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00005*ones
    (1,100), 0.00001*ones(1,100)] * 10; % (2) lower gamma in beginning, higher in middle
```

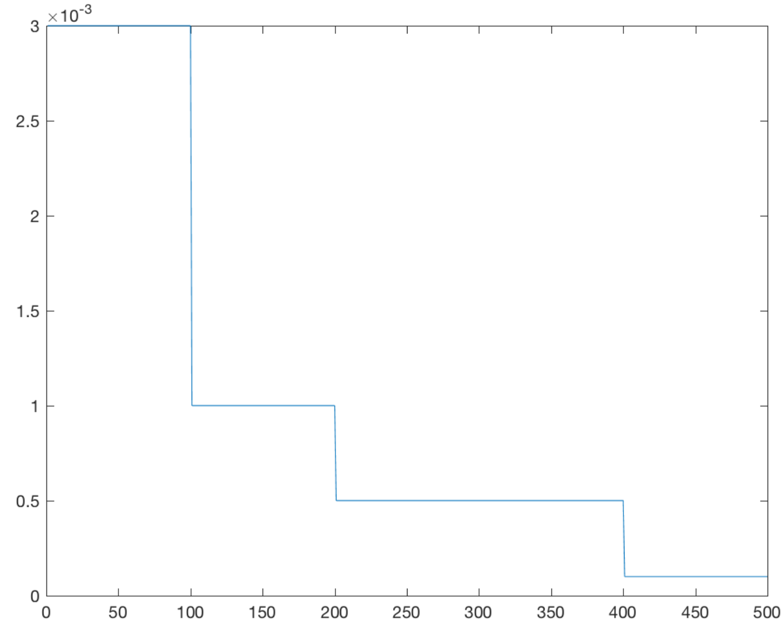

Figure 13: Gamma2.

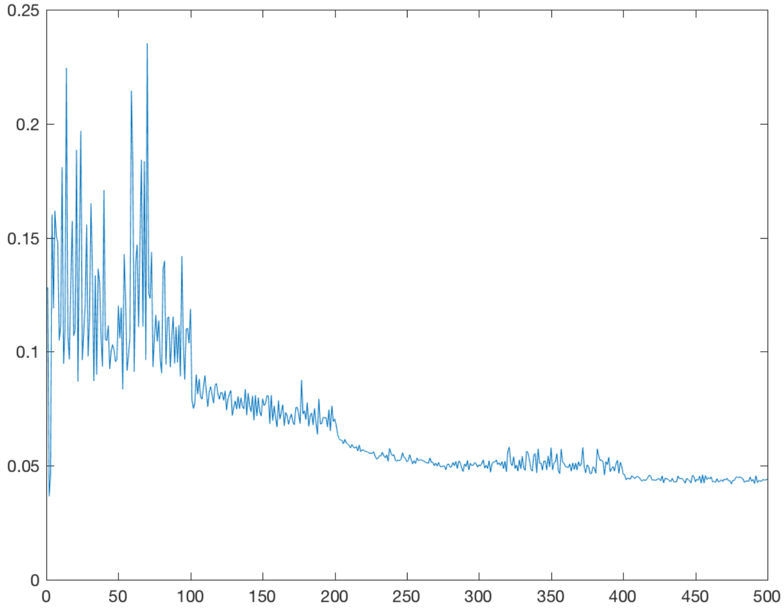

Figure 14: Loss.

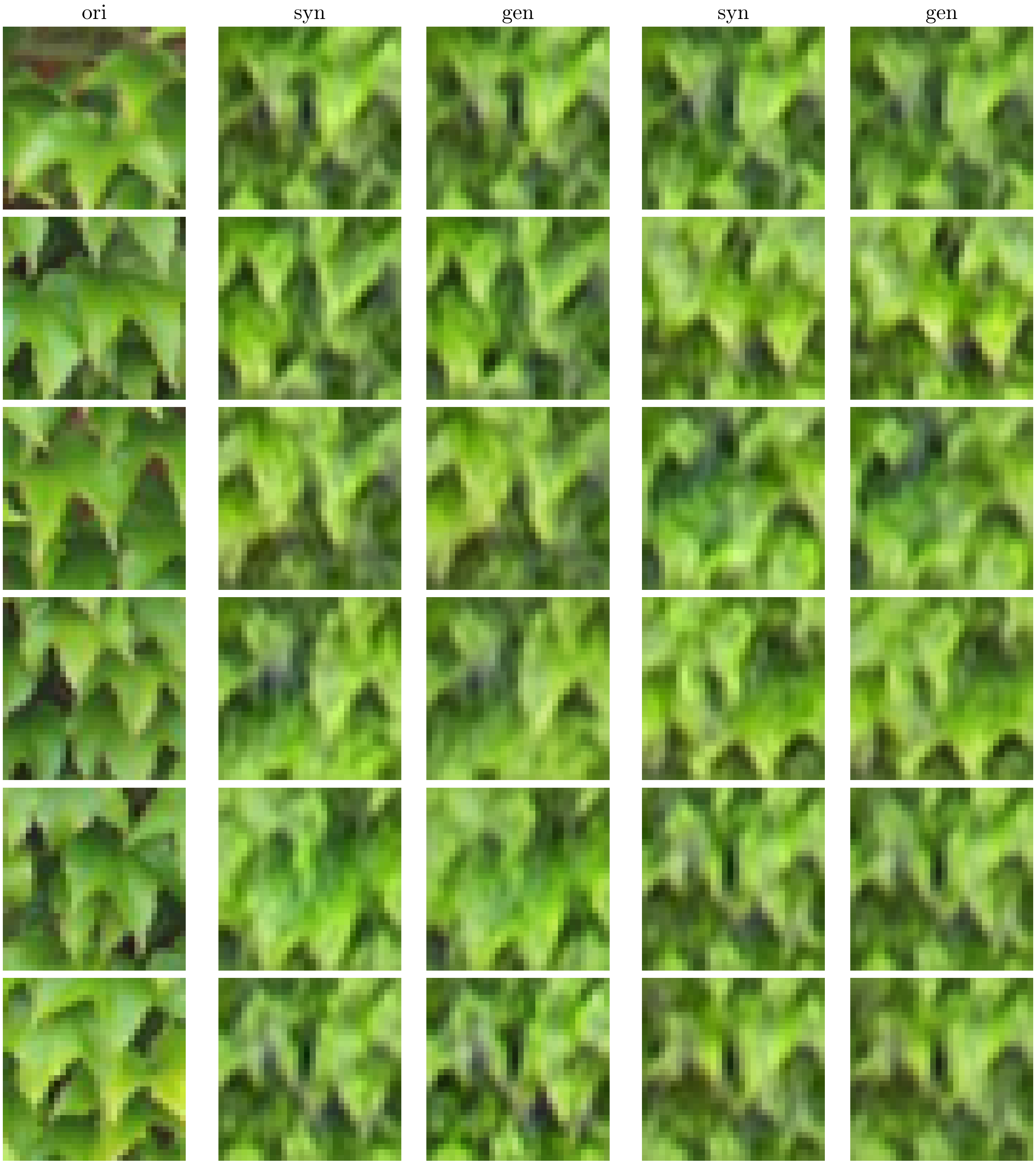

Table 9: Descriptor and generator images

(8) exp_texture_512_8: step-wise - ×20

```
config.Gamma2 = [0.0005*ones(1,100), 0.0001*ones(1,100), 0.00005*ones(1,100), 0.00001*ones
    (1,100), 0.000005*ones(1,100)] * 20; % increased
```

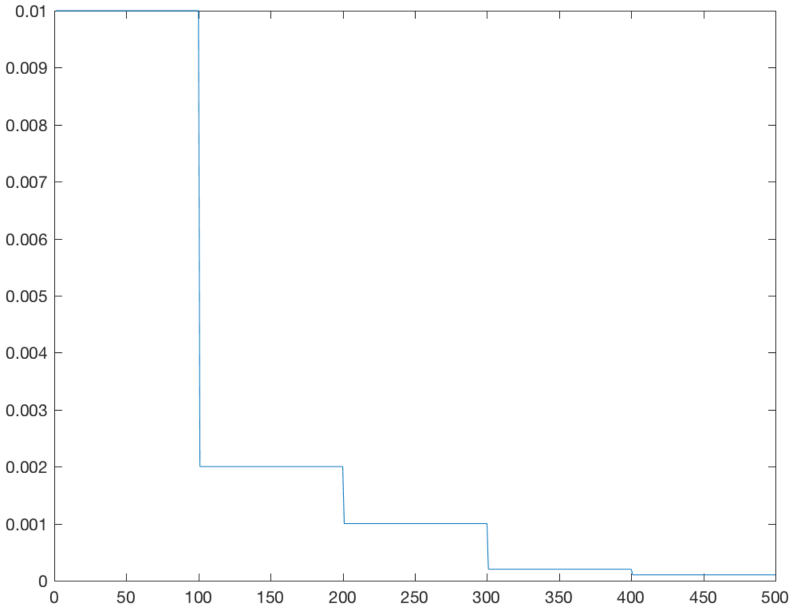

Figure 15: Gamma2.

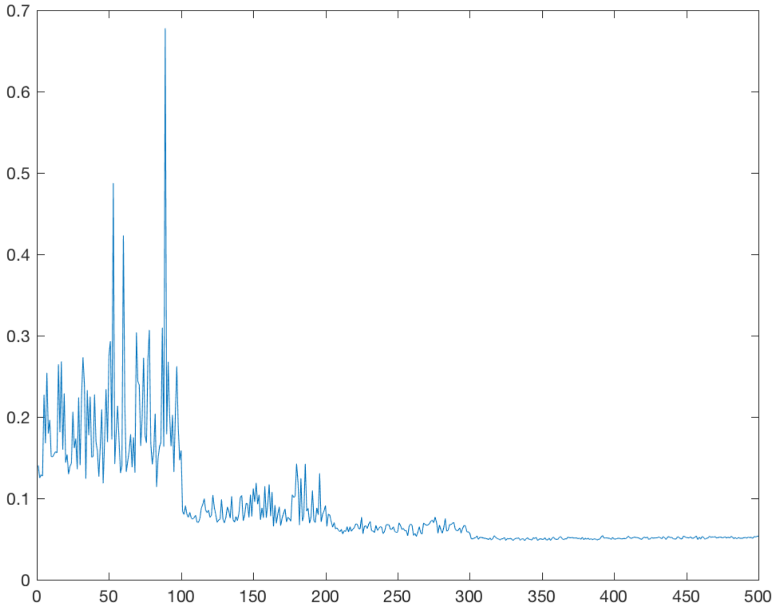

Figure 16: Loss.

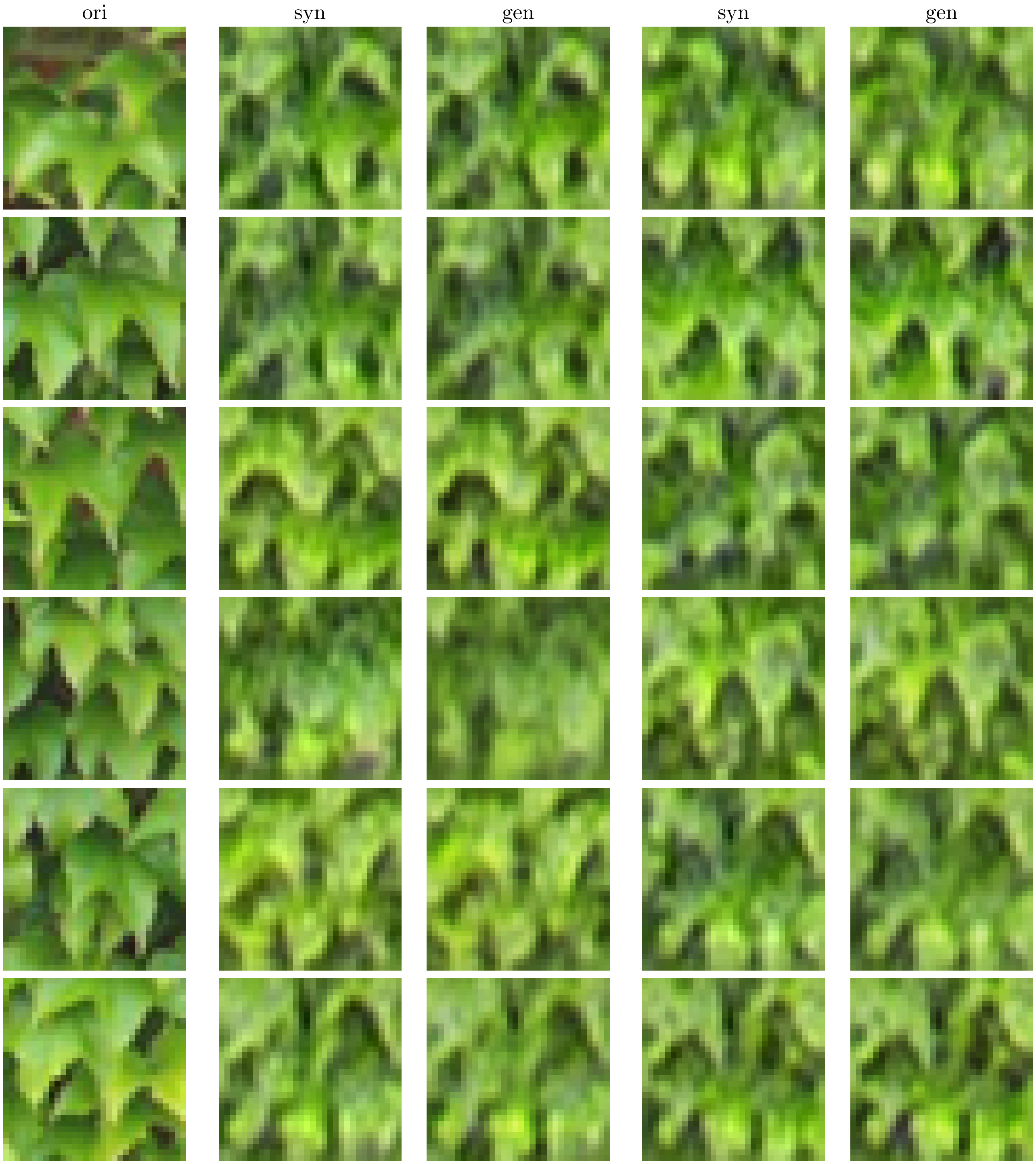

Table 10: Descriptor and generator images

(9) exp_texture_512_9: log-space

```
config.Gamma2 = 0.0005 * logspace(-2, -3, config.nIteration)*100; % logspace
```

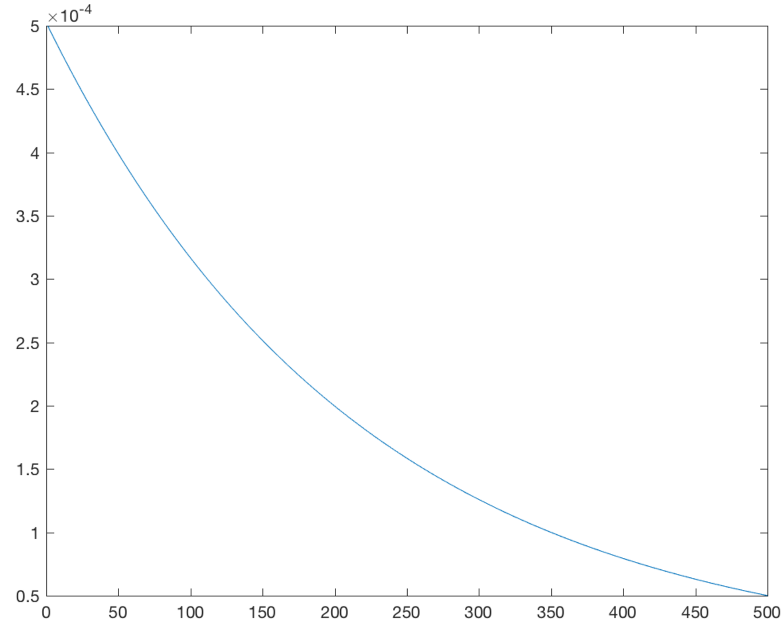

Figure 17: Gamma2.

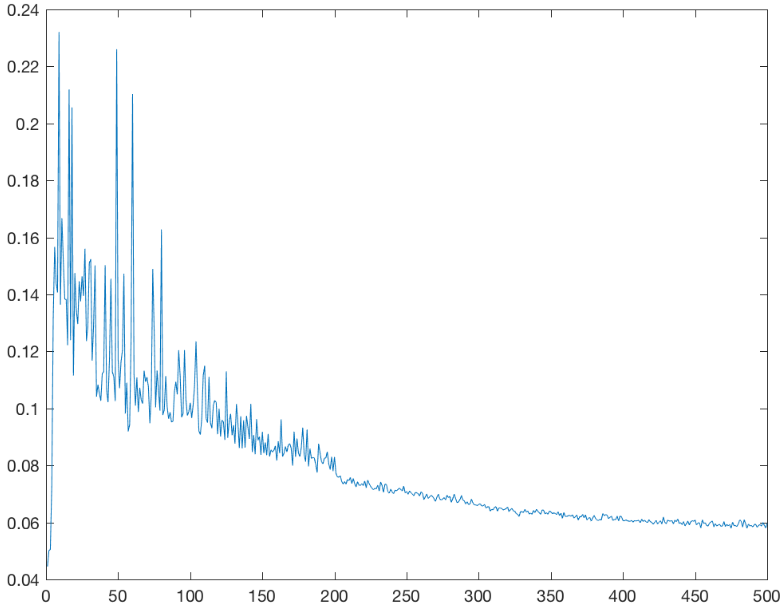

Figure 18: Loss.

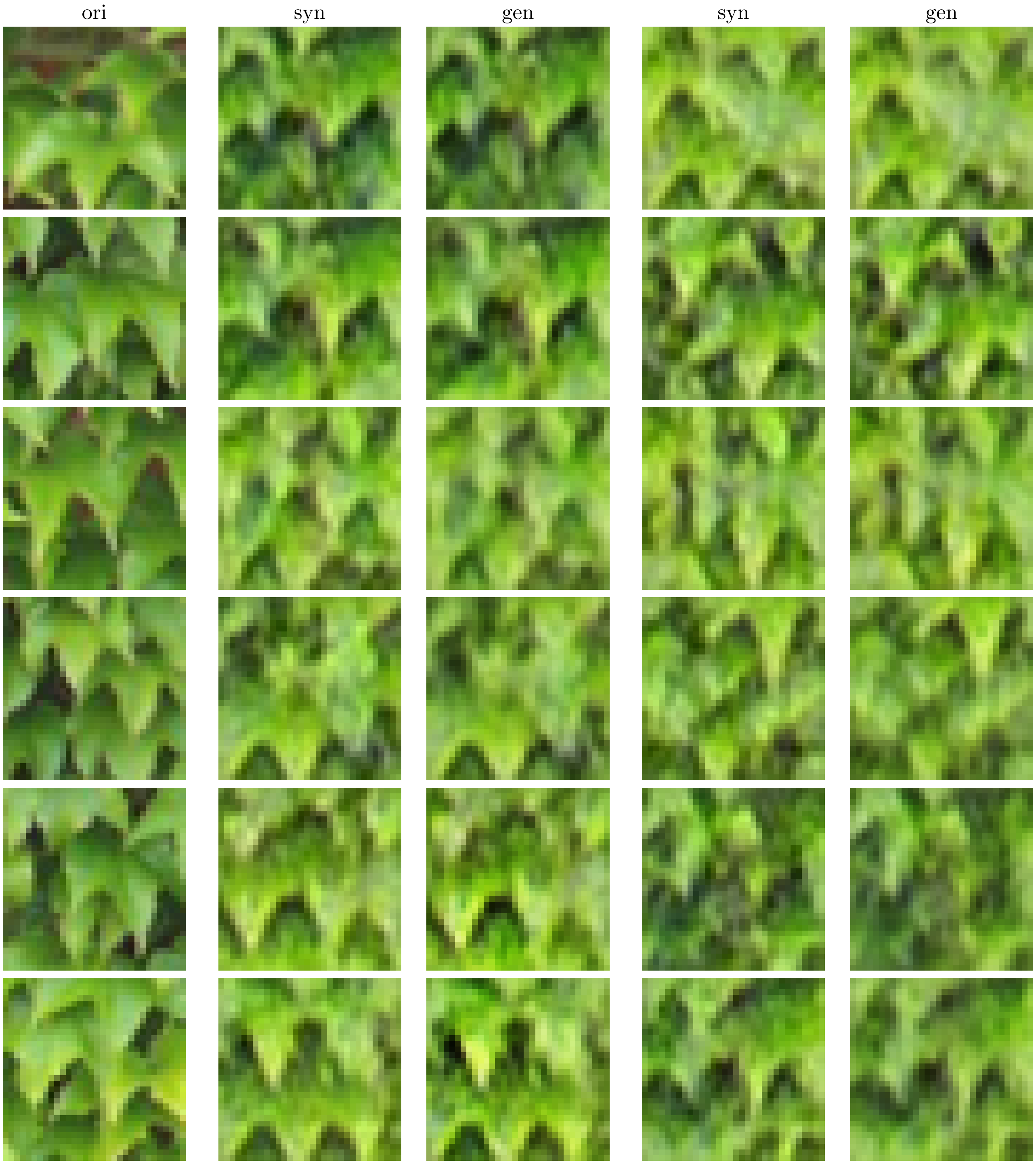

Table 11: Descriptor and generator images

(10) exp_texture_512_5: log-space - increased

```
config.Gamma2 = 0.001 * logspace(-2, -3, config.nIteration)*100; % logspace increased
```

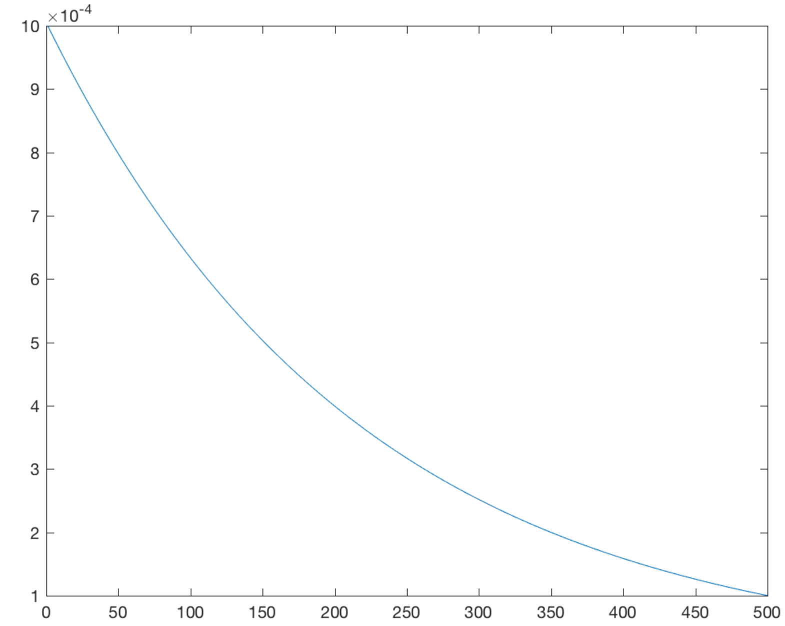

Figure 19: Gamma2.

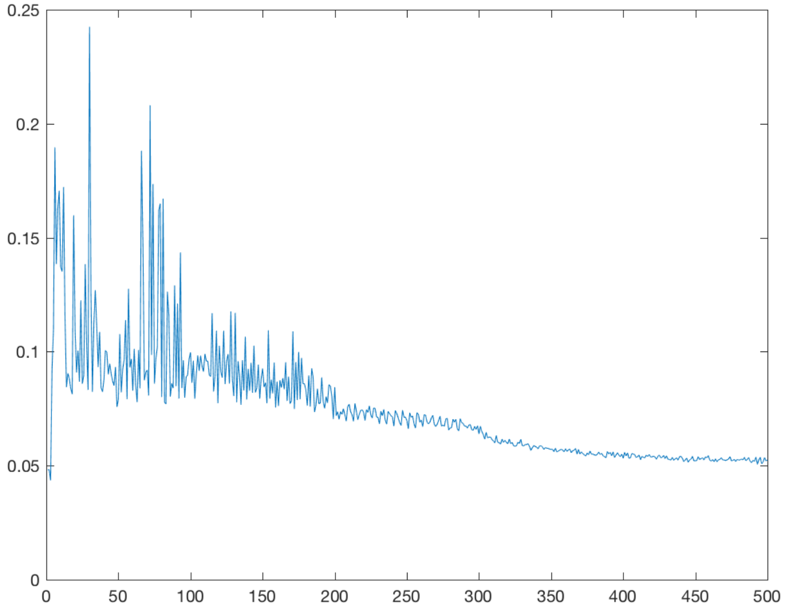

Figure 20: Loss.

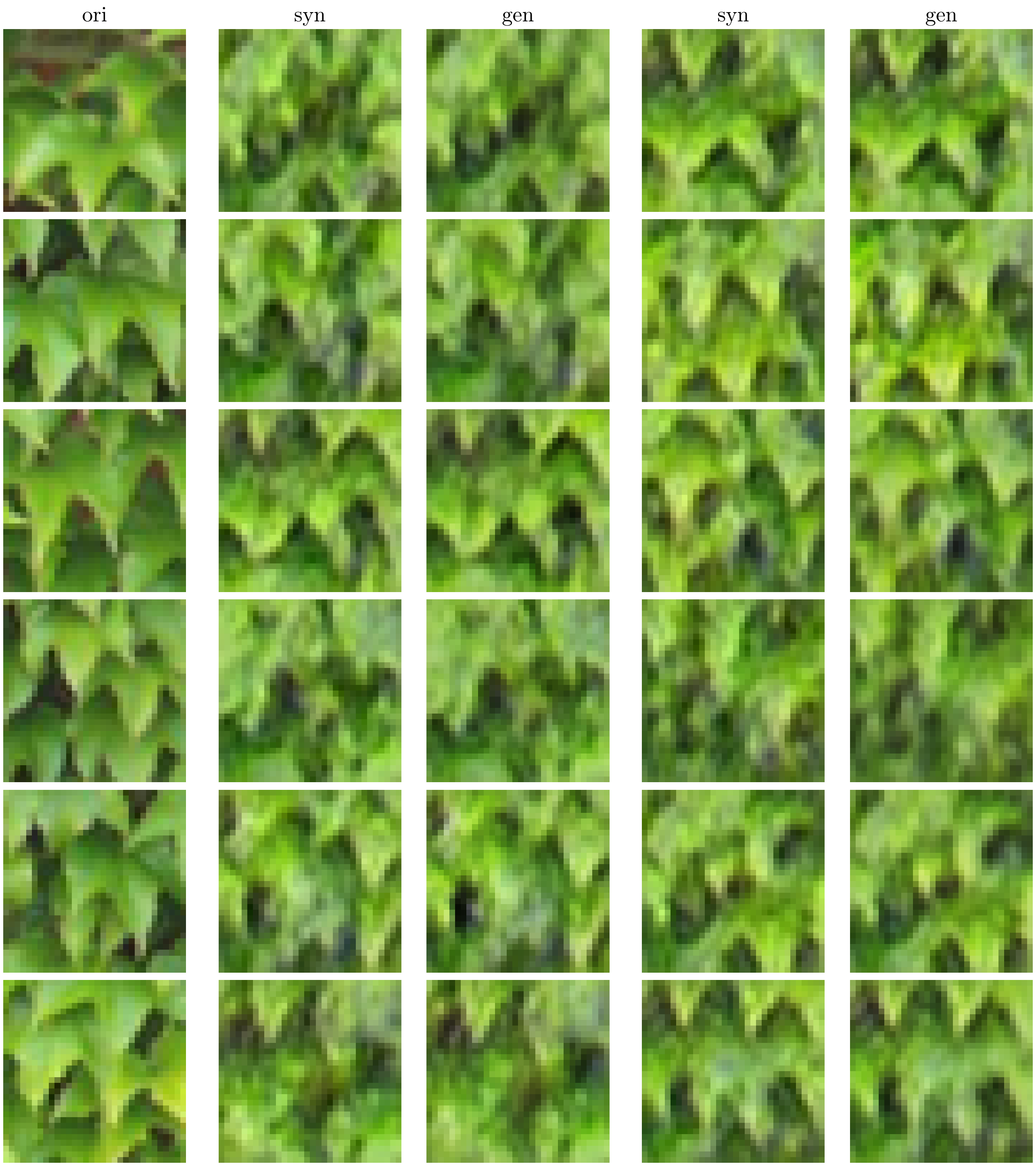

Table 12: Descriptor and generator images

(11) exp_texture_512_11: log-space - long-tail

```
config.Gamma2 = 0.001 * logspace(-2, -4, config.nIteration)*100; % logspace long-tail
```

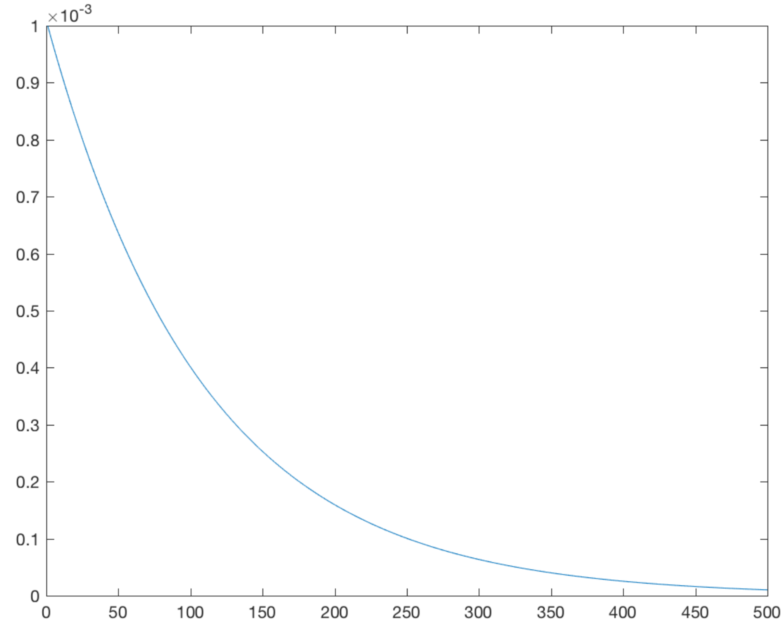

Figure 21: Gamma2.

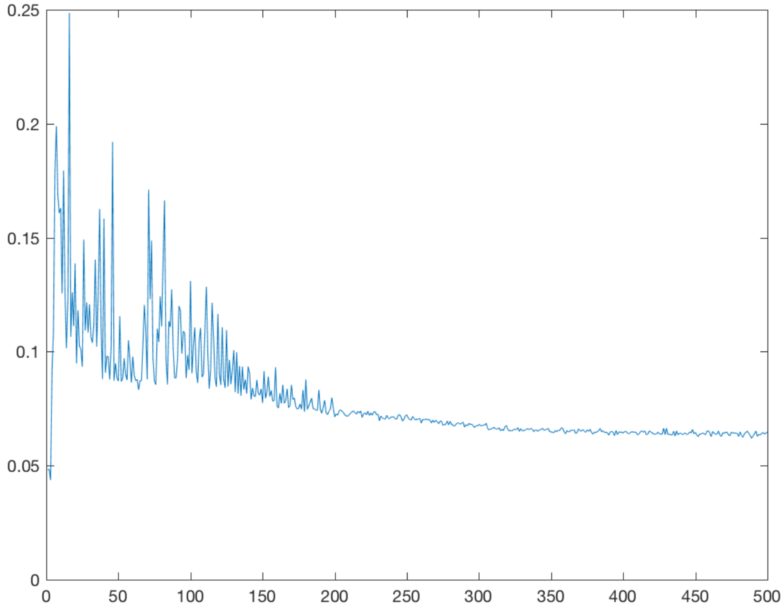

Figure 22: Loss.

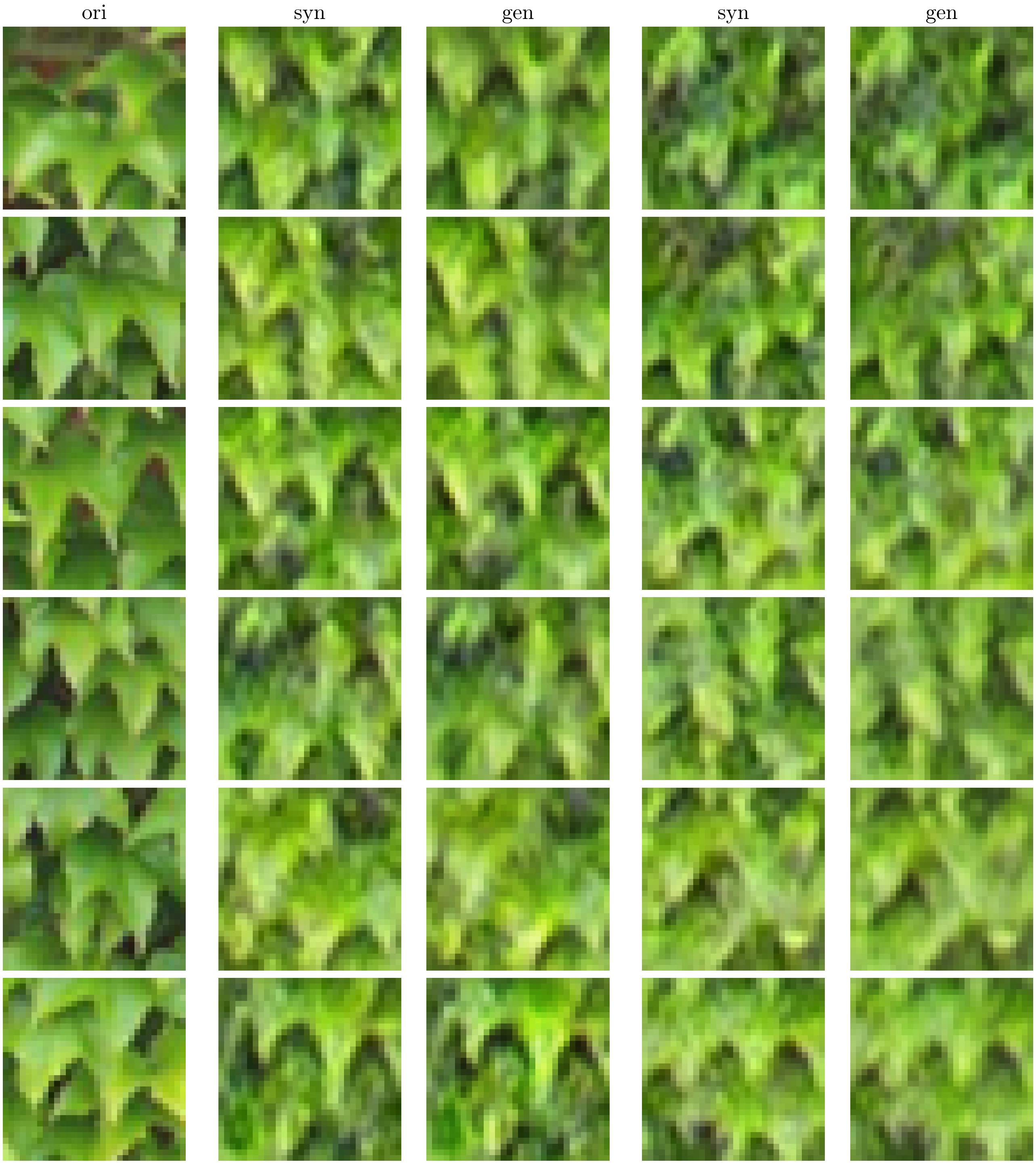

Table 13: Descriptor and generator images

(12) exp_texture_512_12: lin-space

```
config.Gamma2 = 0.0005 * linspace(1,0.01,config.nIteration); % linspace
```

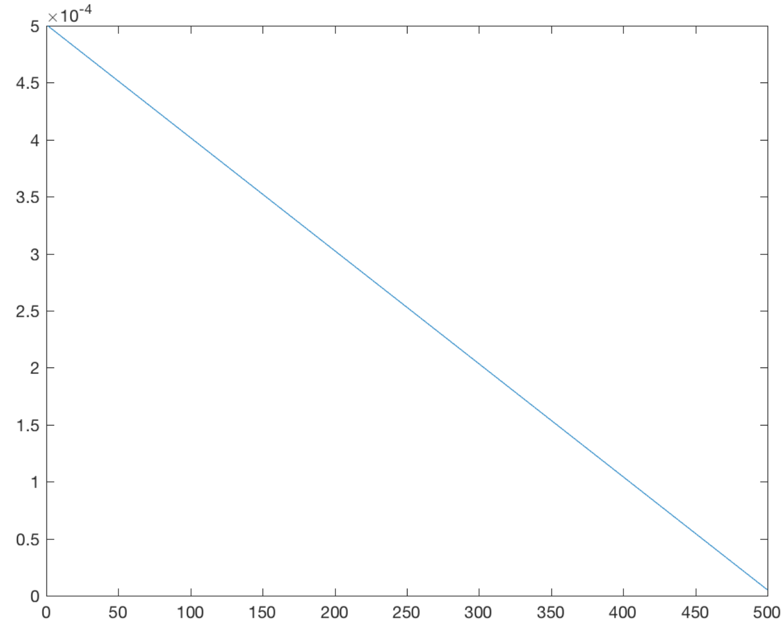

Figure 23: Gamma2.

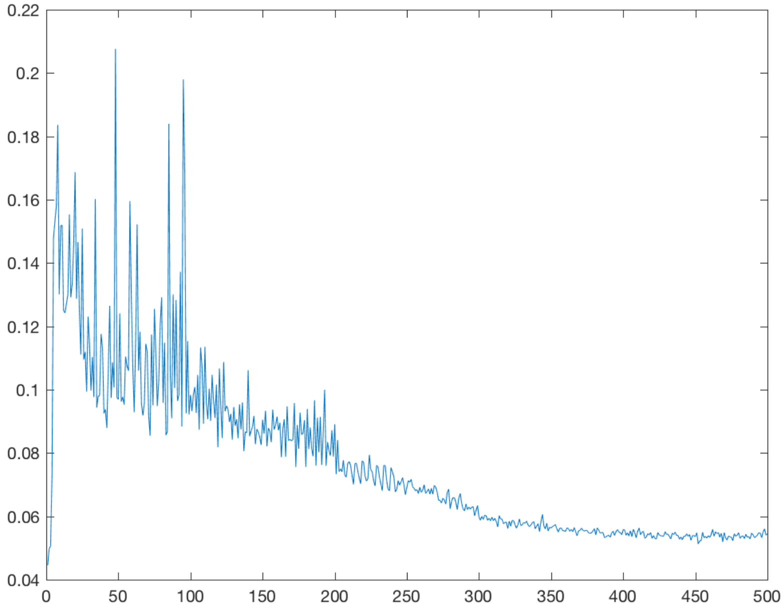

Figure 24: Loss.

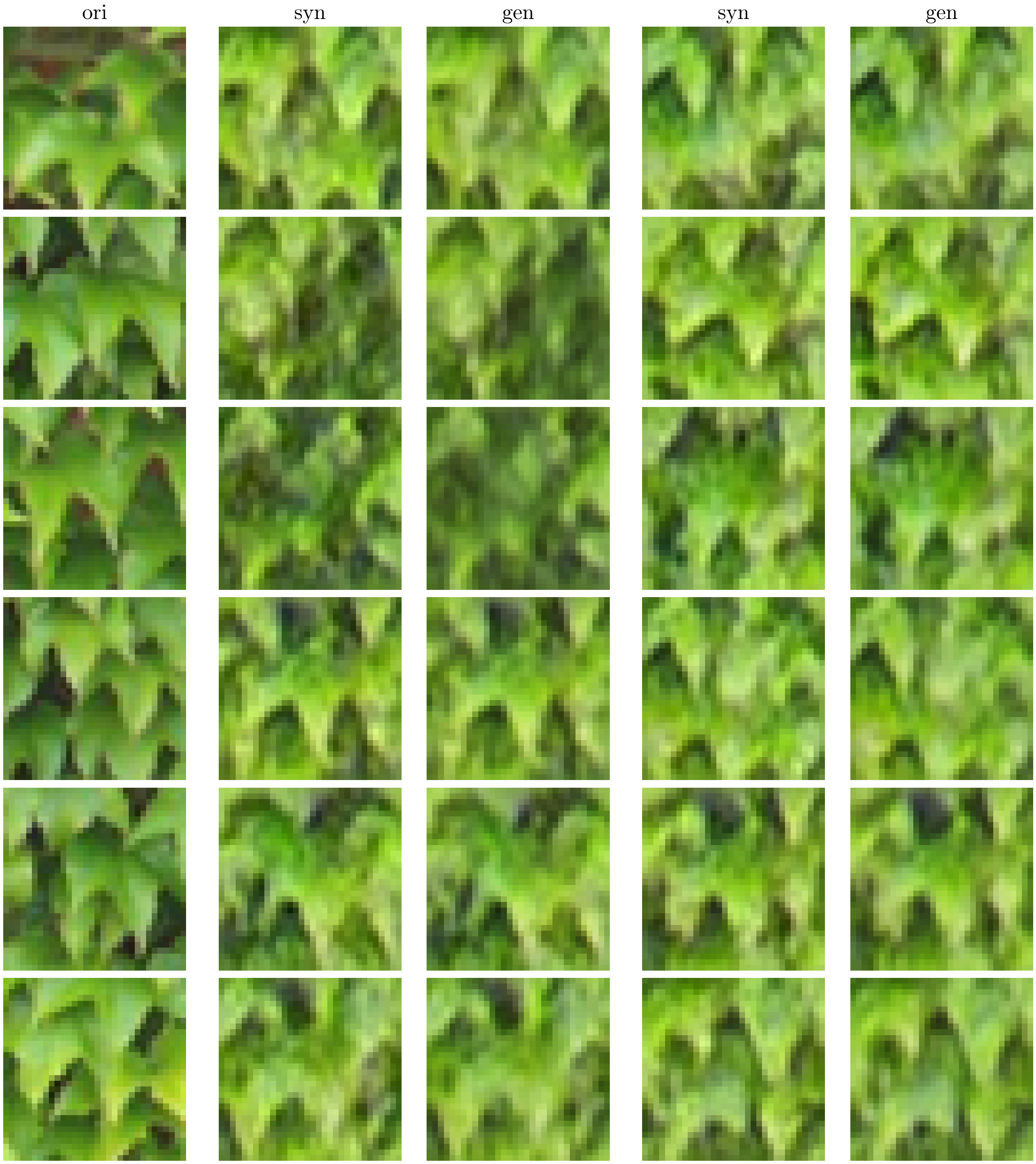

Table 14: Descriptor and generator images