EXSINGAN: LEARNING AN EXPLAINABLE GENERATIVE MODEL FROM A SINGLE IMAGE SUPPLEMENTARY

Anonymous authors

CONTENTS

1 Introduction **2**

2 Parameter selection **2**

2.1 Miscellaneous 2

2.2 GAN inversion 2

2.3 Different rescaling methods 4

3 Quantitative evaluation on *SIFID*, *LPIPS* and *Diversity* **6**

4 Unconditional generation at arbitrary sizes **7**

5 Comparison of DGP, SinGAN, ConSinGAN and ExSinGAN **9**

6 Datasets: *Places50*, *LSUN50*, *ImageNet50* **14**

7 Comparison of SinGAN, ConSinGAN, ExSinGAN on *Places50* **16**

8 Comparison of SinGAN, ConSinGAN, ExSinGAN on *LSUN50* **26**

9 Comparison of SinGAN, ConSinGAN, ExSinGAN on *ImageNet50* **37**

10 Output of different stages **47**

1 INTRODUCTION

This article is the supplementary of *ExSinGAN: Learning an Explainable Generative Model from a Single Image*. We are here to elaborate on some of the issues not covered by the paper, and present more syntheses for comparison. In section 2, we introduce some details about parameter configuration. In section 3, we show quantitative evaluation of three methods. In section 4, we show more details and results about unconditional generation at arbitrary sizes. In section 5, we show the random syntheses of DGP, SinGAN, ConSinGAN and ExSinGAN, which have been presented in the paper. In section 6, we show the datasets *Places50*, *LSUN50*, *ImageNet50*, which are used in the paper. In section 7, 8, 9, we show the random syntheses from SinGAN, ConSinGAN, and ExSinGAN on *Places50*, *LSUN50*, *ImageNet50*.

2 PARAMETER SELECTION

2.1 MISCELLANEOUS

We conduct our experiments on Tesla V100, and the average training time is 40 min per image. Each stage of generator and discriminator is constituted simply by the Conv-IN-LeakyReLU blocks. We set the negative slope of LeakyReLU to 0 for structural GAN and 0.2 for others. A key point is that the size and amount of our inversion results are both small, hence we put the training data of structural GAN on the GPU storage and directly read data from GPU rather than reading data back from local storage at each epoch. Instance Normalization has positive effects on speeding up the training and better results. Batch size has a great influence on structural GAN. A small batch size like 16 or 32 gets more diverse and delicate results than large bath size like 64. The reasons can be ascribed to that a larger batch size makes GAN converge faster and suffer from model collapse.

2.2 GAN INVERSION

Different from the original DGP, we change the parameters of DGP to make the jittering results more reasonable and controllable. We find that fine-tuning the generator by different layers of discriminator has different jittering results. Note Fig 1, supposed that we know the class label of the original image, when using the last layer to fine-tune the generator, the syntheses are more free and uncontrollable. with the layers to fine-tune the generator increased, the structures of syntheses have less changes. However, since we do not know the real label of the given image in practical, and the random label inputting to the generator determines the semantics of syntheses primarily, we need something like the pixel-level loss to constrain the syntheses from deviating too much from the original image in pixel space. Our choice is to use the last five layers for fine-tuning the generator in the paper (Fig. 1). Reducing the number of levels also has a positive effect to the results, *e.g.*, setting the standard derivations to $0.1, 0.2, 0.3$.

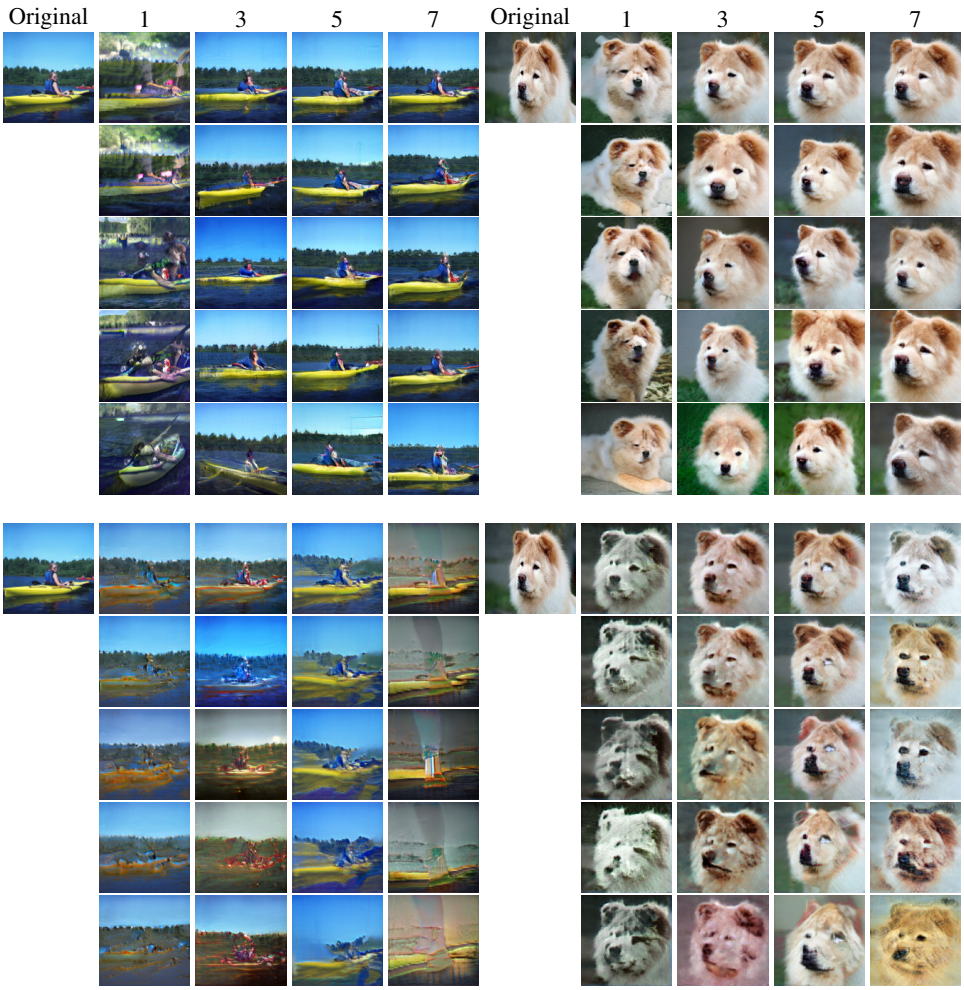


Figure 1: Above: known class label, fine-tuning the generator of BigGAN with the different number of layers of the coupled discriminator. Numbers 1,3,5,7 represent using the last 1,3,5,7 layers of discriminator respectively. Below: random class label, fine-tuning the generator of BigGAN with the different number of layers of the coupled discriminator. For each part, from top to bottom, the standard deviation ranges from 0.1 to 0.5.

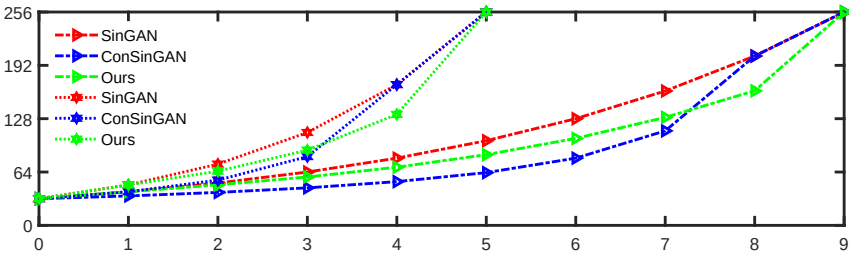


Figure 2: Comparison of different rescaling methods when $N = 5$ and 9, the horizontal and vertical axes respectively represent stage and pixel.

2.3 DIFFERENT RESCALING METHODS

The image rescaling method influences both quality and time consuming very much. SinGAN take the basic rescaling method

$$H_s = H_N \times r^{N-s}, s = 1, \ldots, N. \tag{1}$$

ConSinGAN designs a complicated formula

$$H_s = H_N \times r^{((N-1)/\log(N))*\log(N-s)+1}, s = 0, \ldots, N-1. \tag{2}$$

Our rescaling method by analogous Taylor approximation of equation (1)

$$H_s = H_0 \times (1 + st + \frac{s(s-1)(s-2)}{2}t^3), s = 0, \ldots, N, t = \frac{1}{r} - 1. \tag{3}$$

As we have discussed in the paper, equation (2) has a gentle slope when s is small, making it pay more attention to the small scale of the image compared with (Fig. 2). The intention of this design is to concentrate on the layout of the image. However, it will cause serious artifacts when N is a large number, *e.g.*, $N = 9$, see Fig 3. And when s is large, the computation of forwarding and backward propagation costs more time. Therefore, to reduce training time, we need to avoid paying too much attention to the large scales. Our rescaling method effectively reduce the training time without losing the performance, see Fig 3 and Table 1.

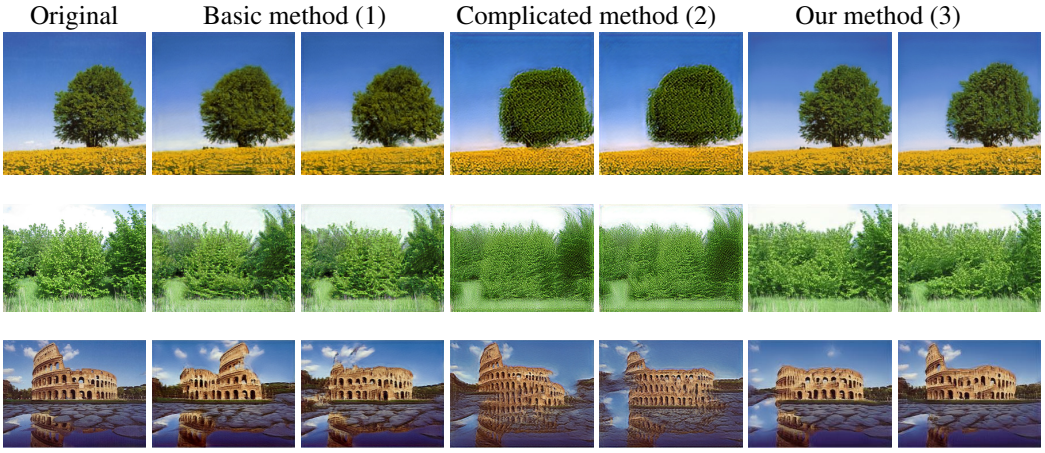


Figure 3: The comparison of the basic rescaling method1, complicated rescaling method 2, and our rescaling method 3 on 10-stage SinGAN.

Rescaling Method	**ExSinGAN** 7 stages	**SinGAN** 10 stages
SinGAN	60 min	80 min
ConSinGAN	55 min	78 min
Ours	40 min	60 min

Table 1: The training time with different rescaling methods. Our rescaling method costs less time on training ExSinGAN and SinGAN.

3 QUANTITATIVE EVALUATION ON *SIFID*, *LPIPS* AND *Diversity*

The Fréchet Inception Distance (FID) uses the activation vector after the last pooling layer in the Inception Network for comparing the distribution of given images and their syntheses. SinGAN proposed *SIFID* for image synthesis by replacing the activation vector with the output of the convolutional layer before the second pooling layer, and the *Diversity* to compare the diversity of syntheses with original images. Following the implementation of SinGAN and ConSinGAN, we sample 50 syntheses for each image to evaluate these metrics. We have reported the *SIFID* and *LPIPS* in the paper, and Table 2, 3, 4, also reported their standard deviations and the *Diversity*. Combining with the examples in paper, it seems that the model with lower *Diversity* is better. A low *Diversity* obviously has more regular syntheses, while a high *Diversity* indicates that the syntheses of the model are very chaotic.

Model	***Places50***	***LSUN50***	***ImageNet50***
SinGAN	0.09 ± 0.07	0.23 ± 0.15	0.60 ± 0.30
ConSinGAN	**0.06 ± 0.03**	0.11 ± 0.06	0.56 ± 0.26
ExSinGAN	0.10 ± 0.08	**0.11 ± 0.07**	**0.45 ± 0.24**

Table 2: Evaluations of *SIFID* on the *Places50*, *LSUN50*, and *ImageNet50* dataset.

Model	***Places50***	***LSUN50***	***ImageNet50***
SinGAN	0.25 ± 0.08	0.33 ± 0.10	0.33 ± 0.12
ConSinGAN	0.30 ± 0.10	0.32 ± 0.12	0.39 ± 0.14
ExSinGAN	**0.23 ± 0.05**	**0.25 ± 0.05**	**0.24 ± 0.07**

Table 3: Evaluations of *LPIPS* on the *Places50*, *LSUN50*, and *ImageNet50* dataset.

Model	***Places50***	***LSUN50***	***ImageNet50***
SinGAN	0.52	0.64	0.61
ConSinGAN	0.50	0.54	0.66
ExSinGAN	0.47	0.50	0.56

Table 4: Evaluations of *Diversity* on the *Places50*, *LSUN50*, and *ImageNet50* dataset.

4 UNCONDITIONAL GENERATION AT ARBITRARY SIZES

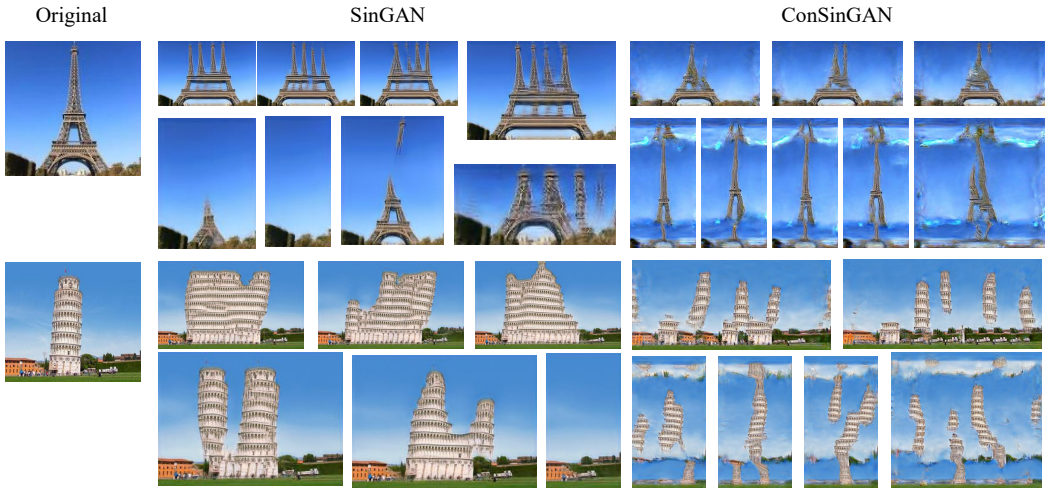


Figure 4: Unconditional generation at arbitrary sizes of SinGAN and ConSinGAN.

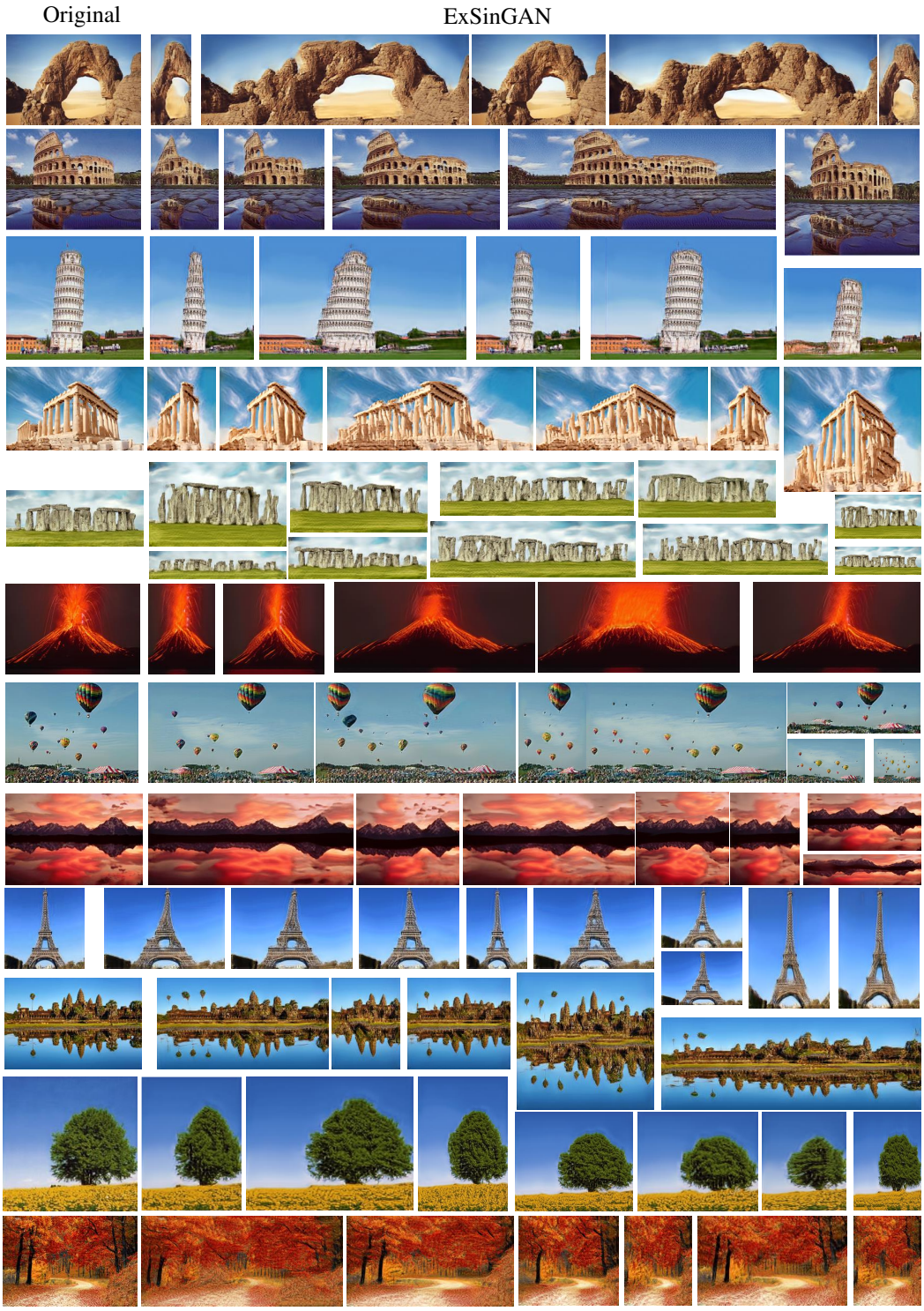


Figure 5: Unconditional generation at arbitrary sizes of ExSinGAN

5 Comparison of DGP, SinGAN, ConSinGAN and ExSinGAN

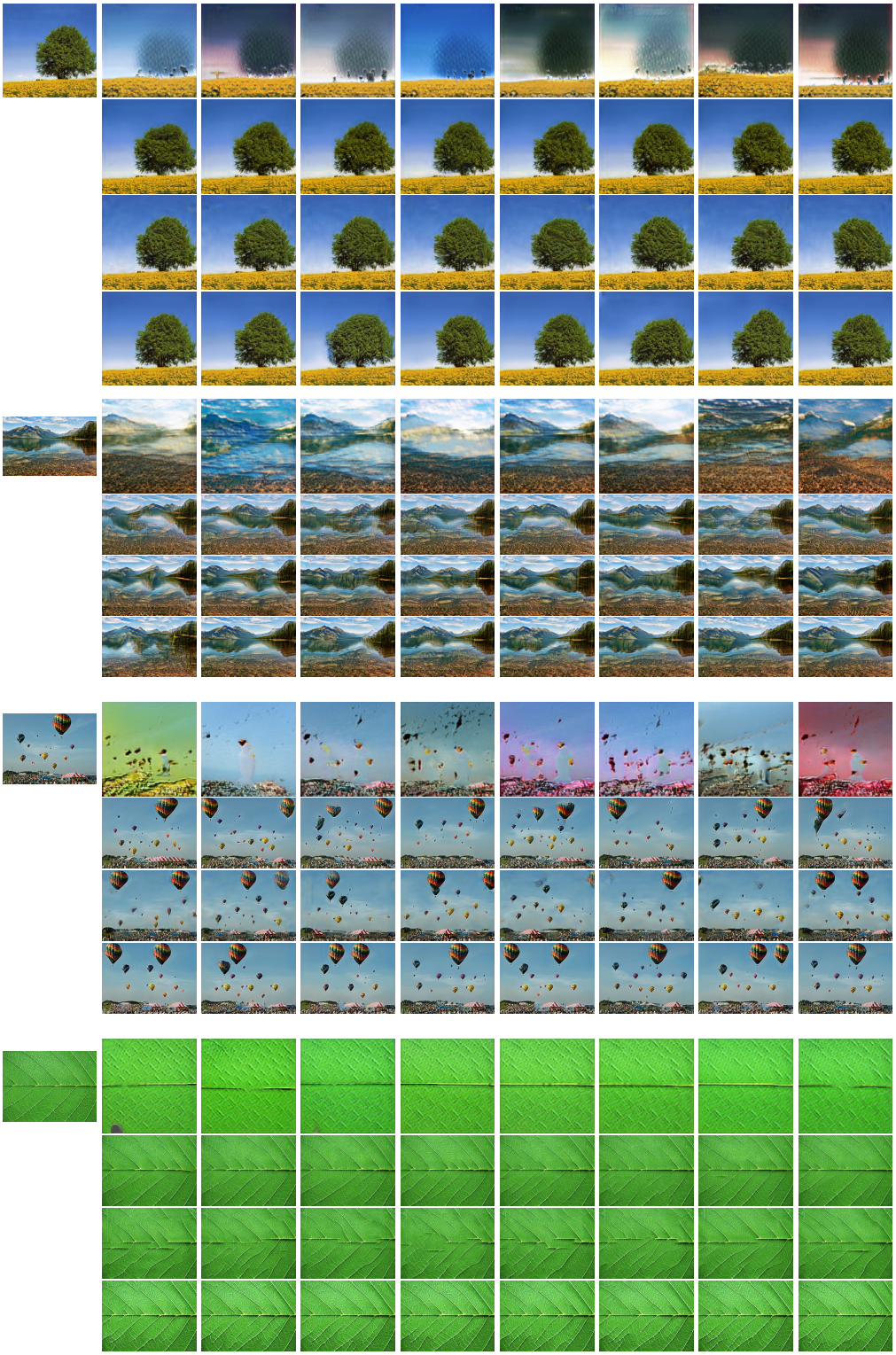


Figure 6: Comparison of various methods on the texture images. The first column of images is original. For each input, there are four rows of images, which are random samples of DGP, SinGAN, ConSinGAN, and ExSinGAN.

Figure 7: Comparison of various methods on semantic images. The first column of images is original. For each input, there are four rows of images, which are random samples of DGP, SinGAN, ConSinGAN, and ExSinGAN.

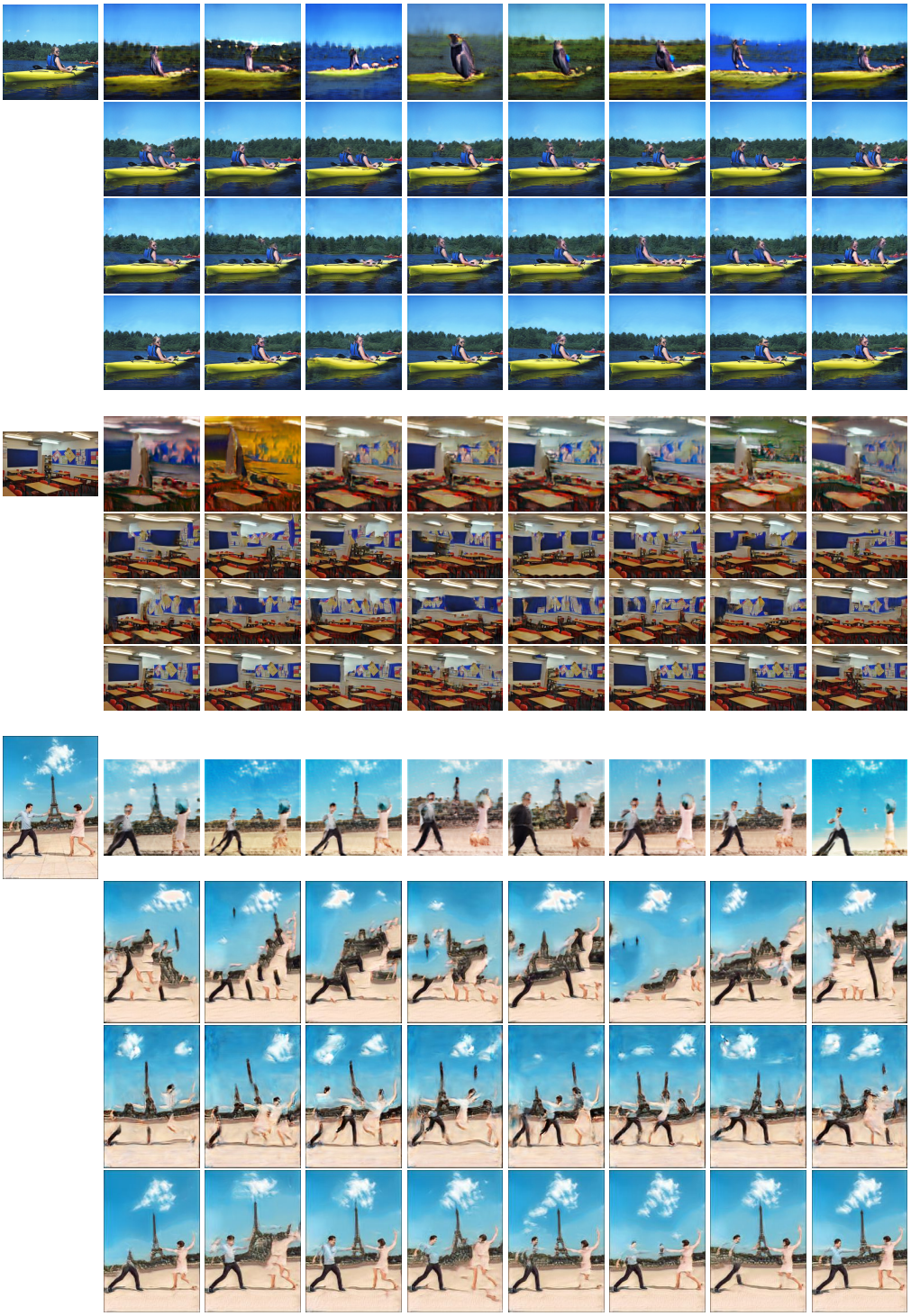


Figure 8: Comparison of various methods on structural images. The first column of images is original. For each input, there are four rows of images, which are random samples of DGP, SinGAN, ConSinGAN, and ExSinGAN.

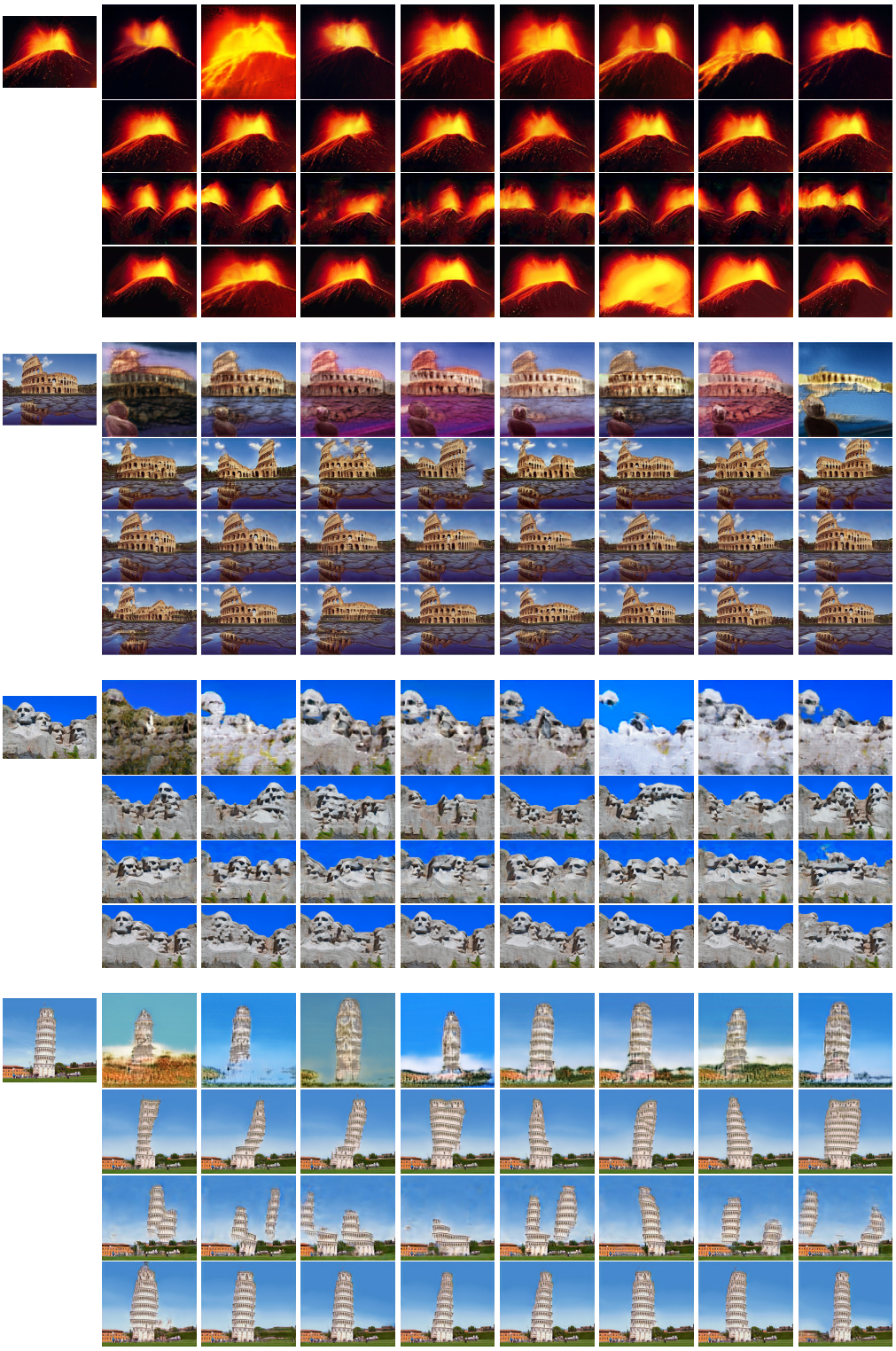


Figure 9: Comparison of various methods on images synthesis. The first column of images is original. For each input, there are four rows of images, which are random samples of DGP, SinGAN, ConSinGAN, and ExSinGAN.

Figure 10: Comparison of various methods on image synthesis. The first column of images is original. For each input, there are four rows of images, which are random samples of DGP, SinGAN, ConSinGAN, and ExSinGAN.

6 DATASETS: *Places50*, *LSUN50*, *ImageNet50*

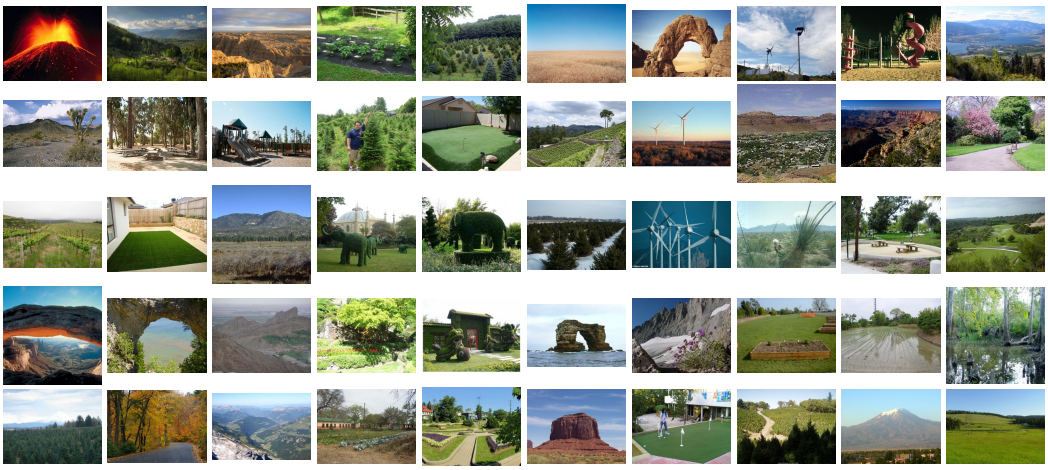


Figure 11: *Places50*

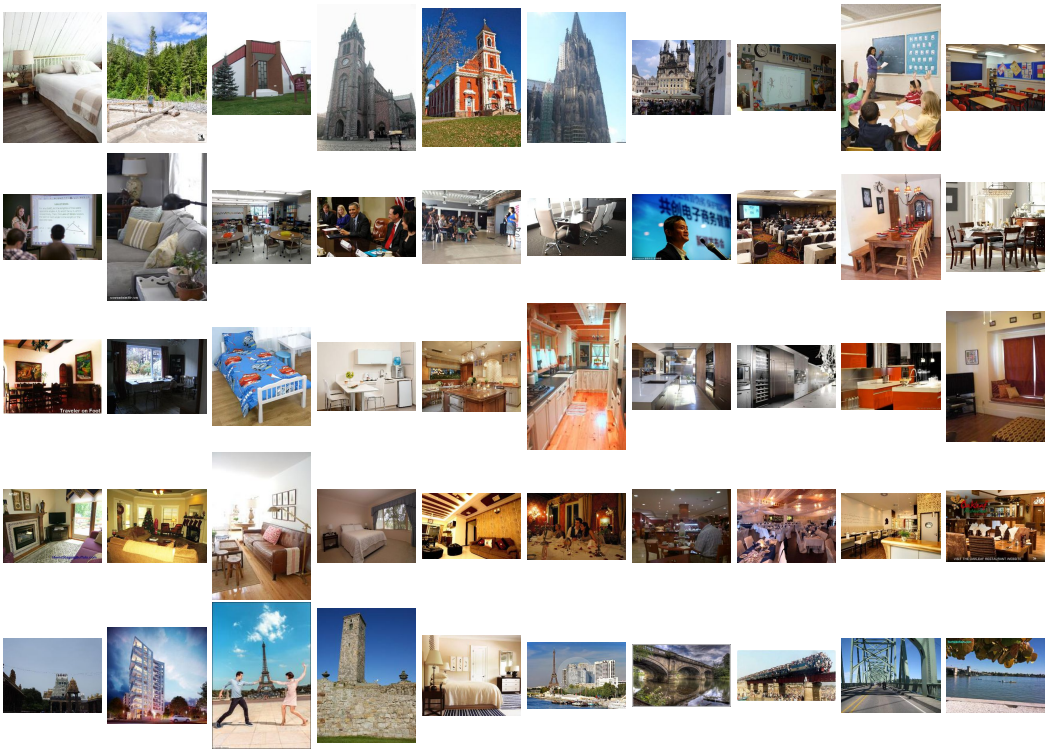


Figure 12: *LSUN50*

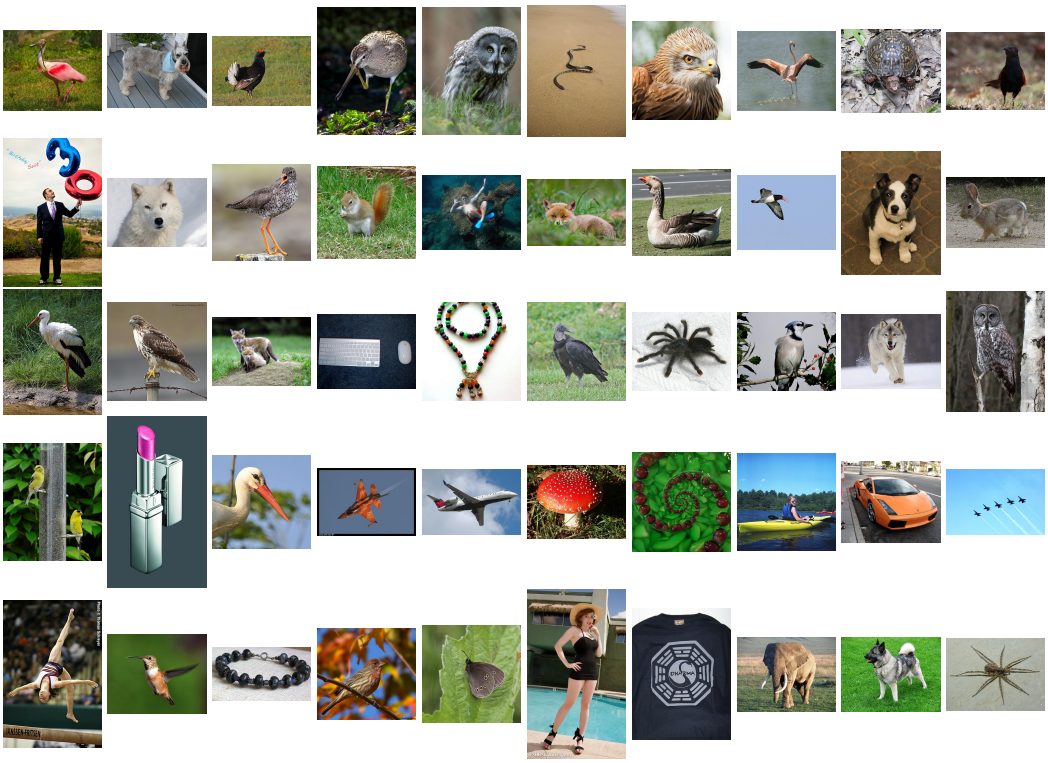


Figure 13: *ImageNet50*

7 COMPARISON OF SINGAN, CONSINGAN, EXSINGAN ON *Places50*

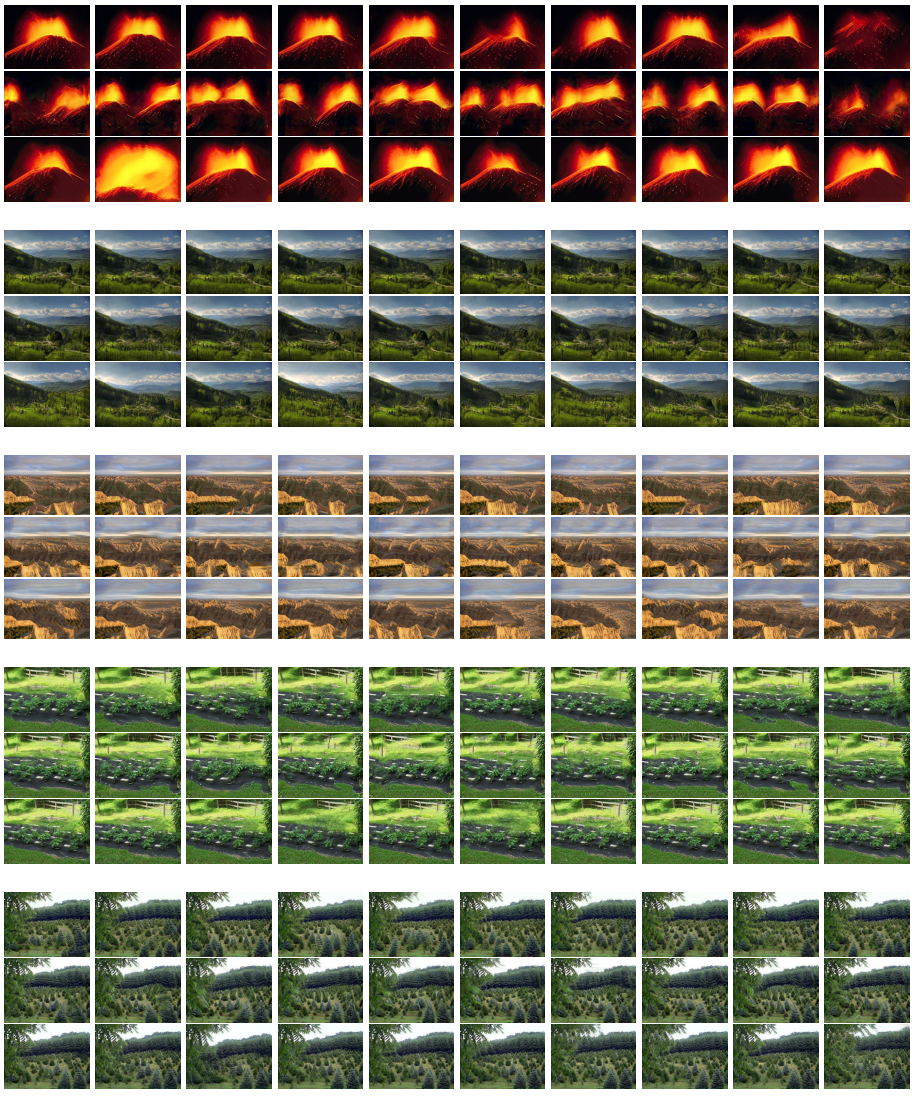


Figure 14: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

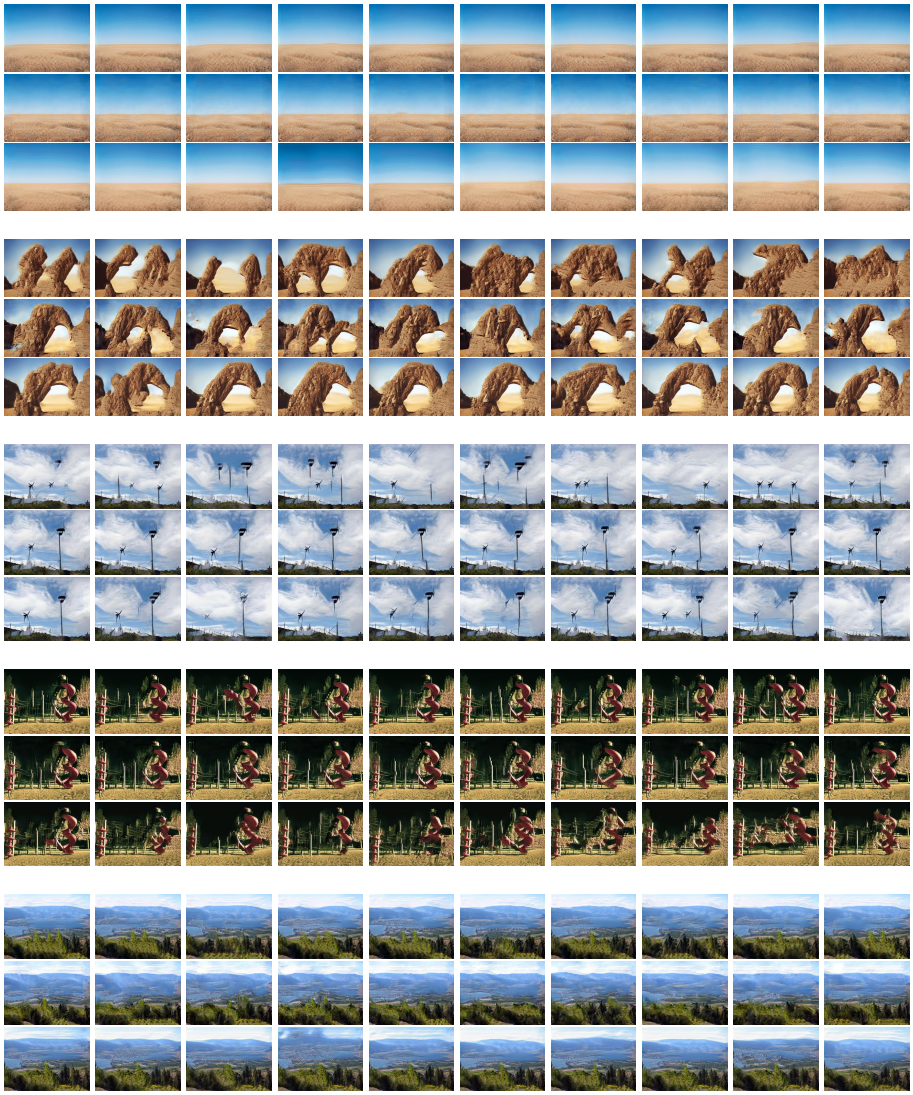


Figure 15: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

Figure 16: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

Figure 17: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

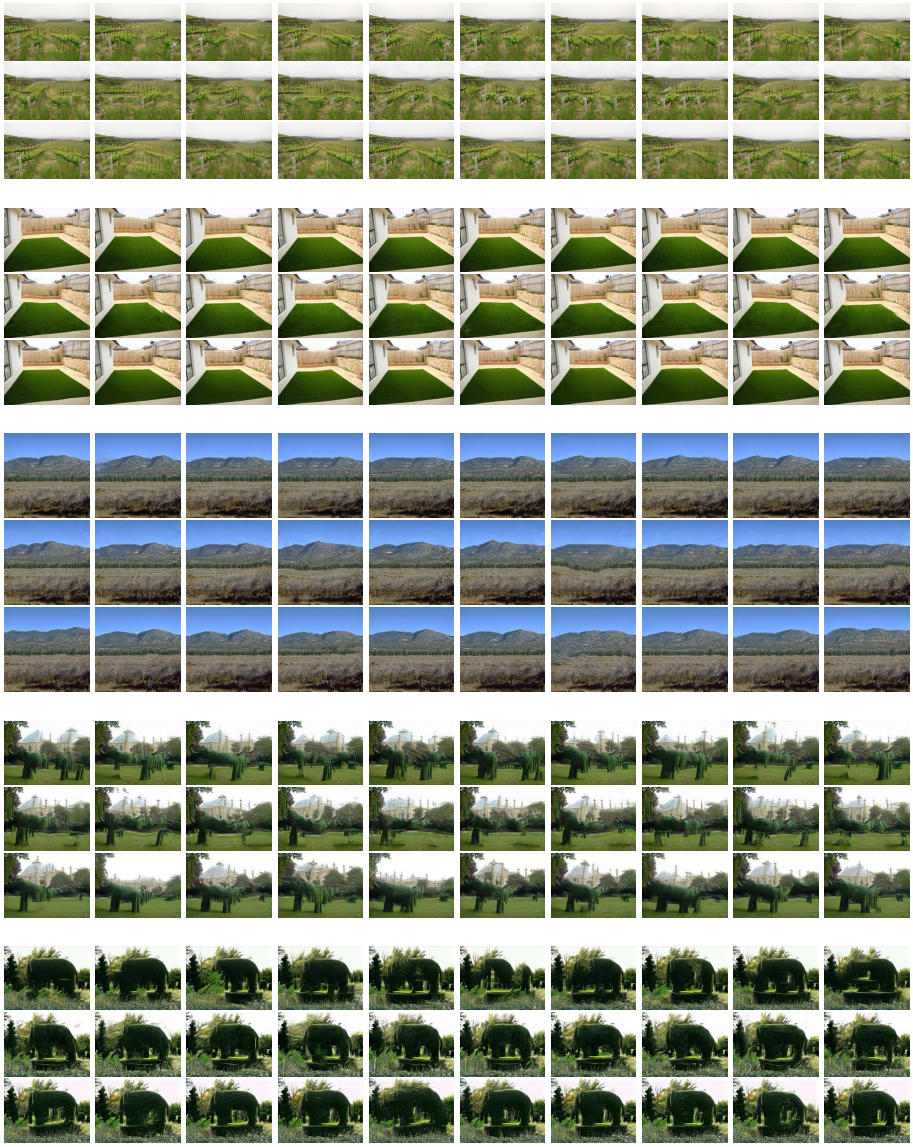


Figure 18: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

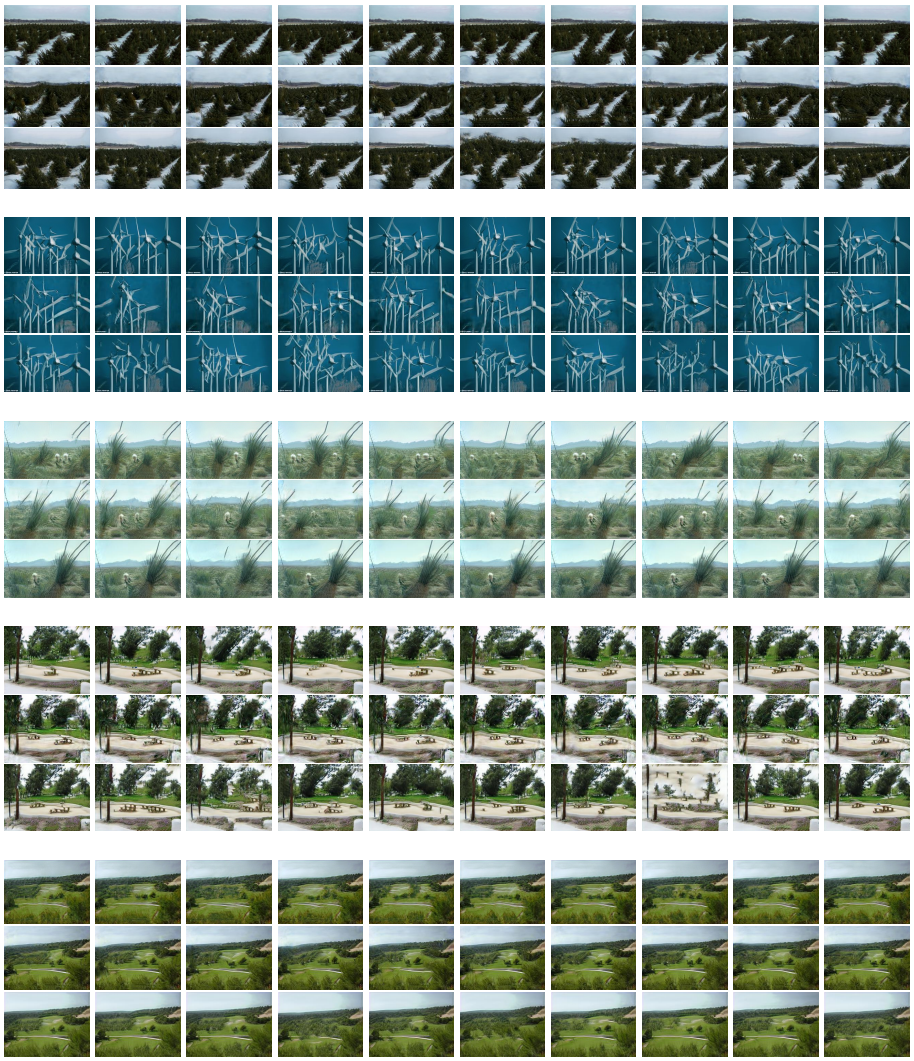


Figure 19: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

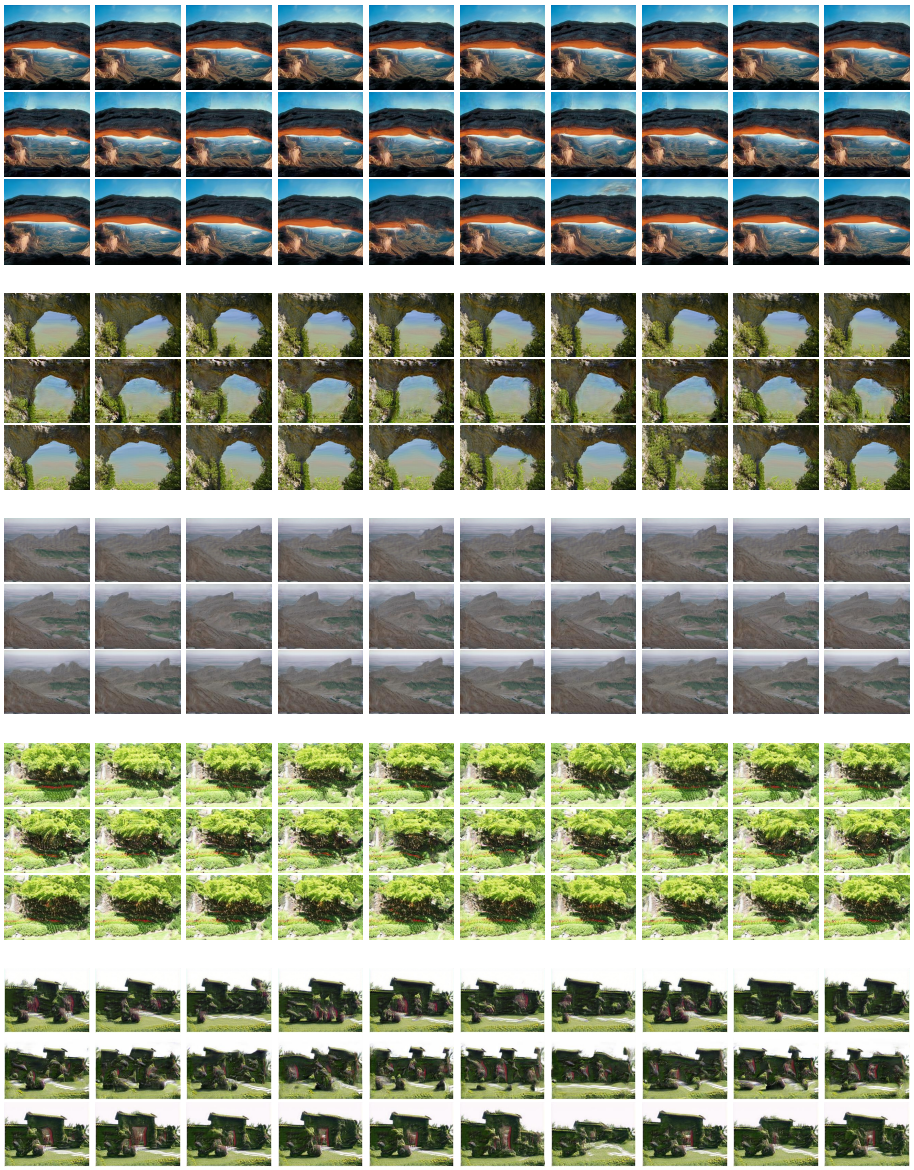


Figure 20: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

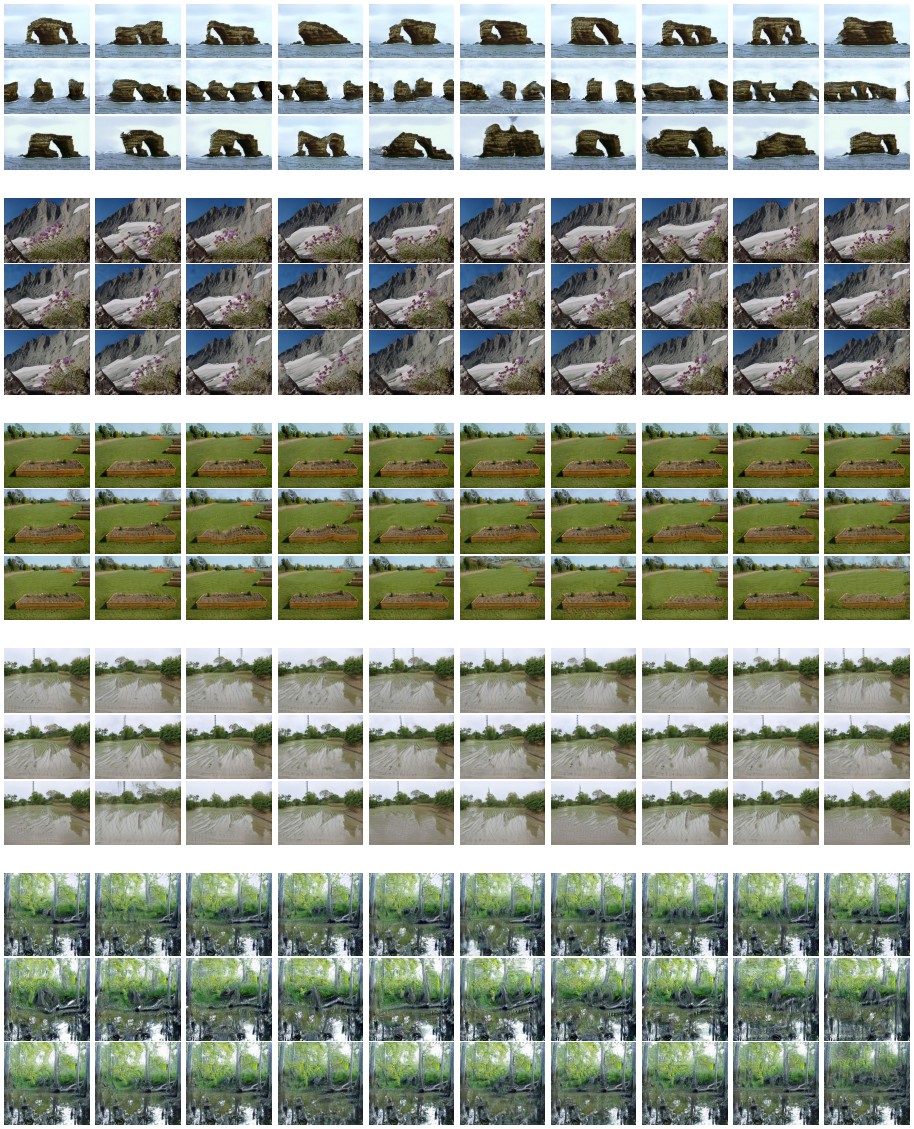


Figure 21: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

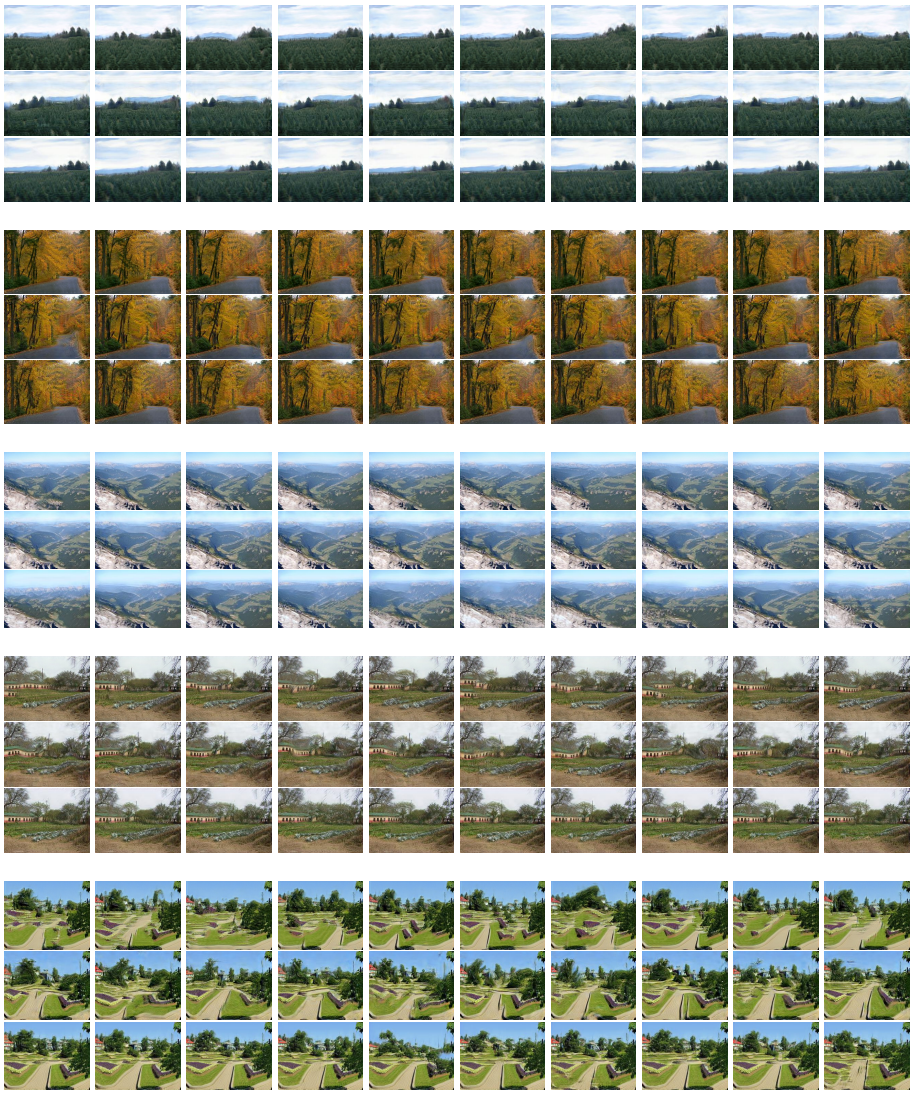


Figure 22: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

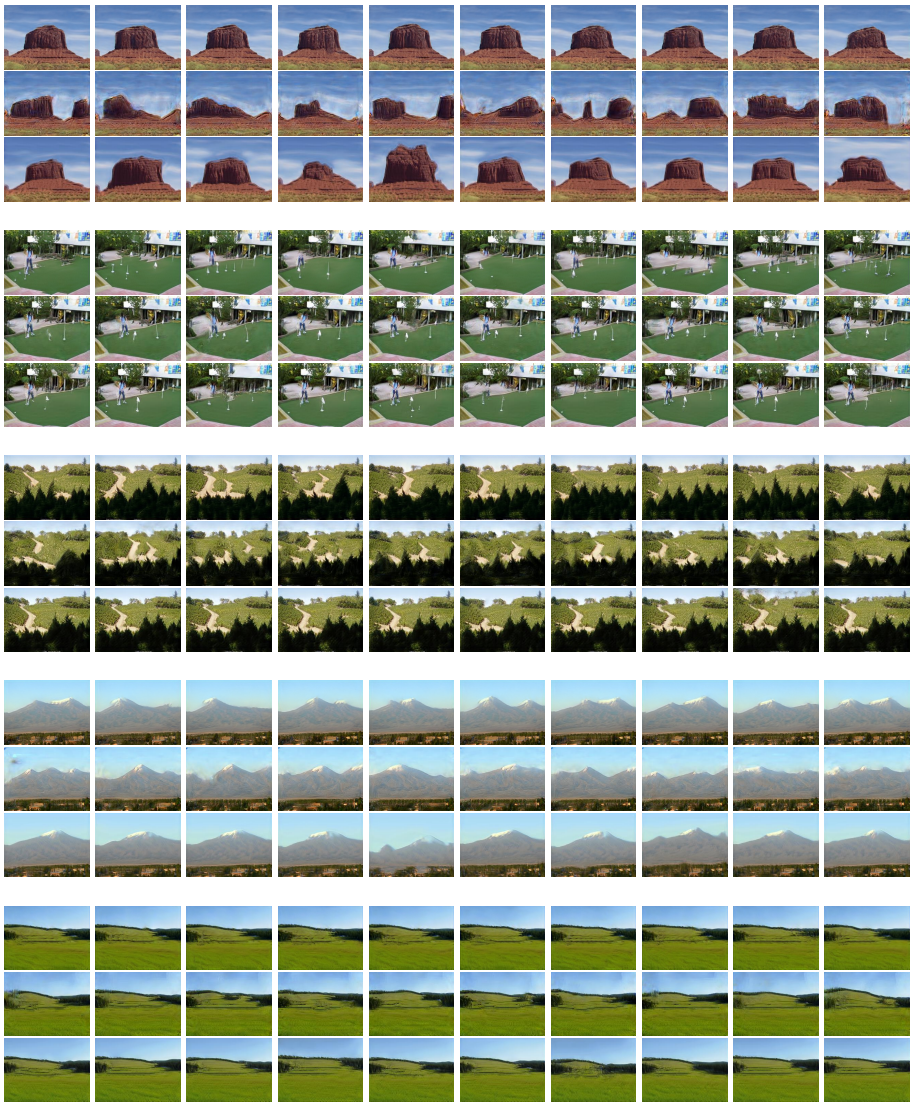


Figure 23: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *Places50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

8 COMPARISON OF SINGAN, CONSINGAN, EXSINGAN ON *LSUN50*

Since ConSinGAN has provided ten samples of each image in *LSUN50* officially for user-study, in order to exclude the impact of hardware or platform, we present their official samples here. However, in the quantitative comparison, since the number of samples is less than 50, we used their official code to sample 50 examples to make a fair comparison.

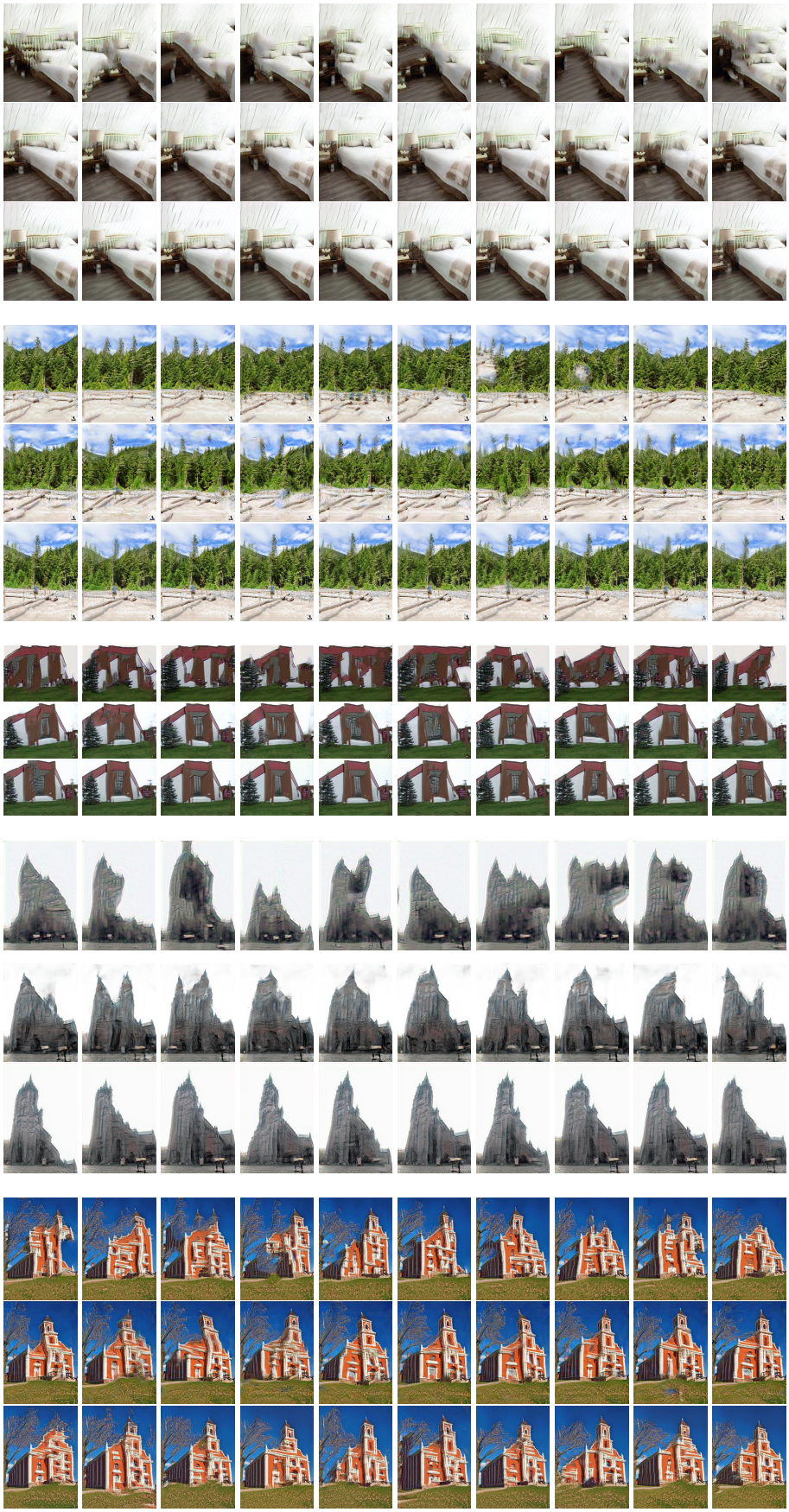


Figure 24: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

Figure 25: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

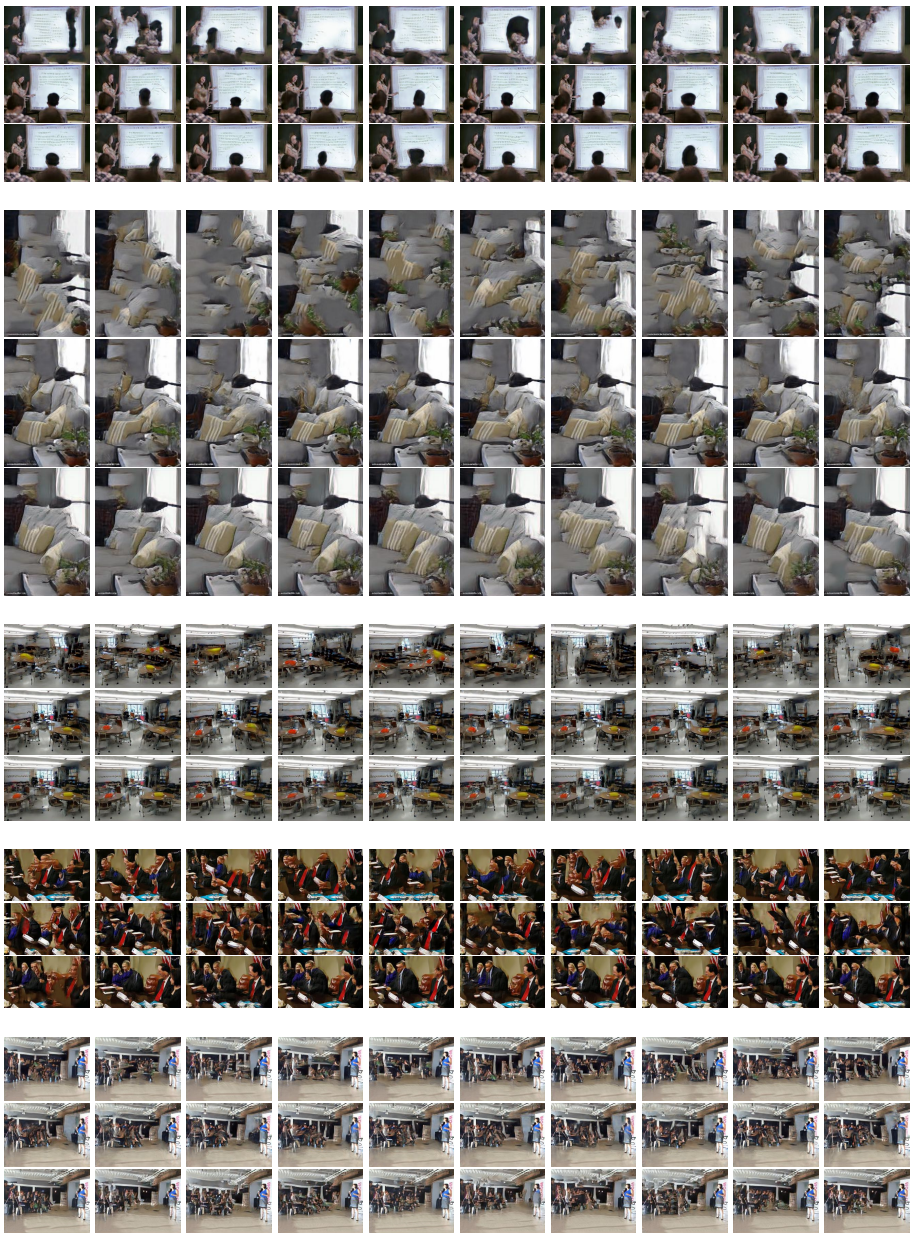


Figure 26: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

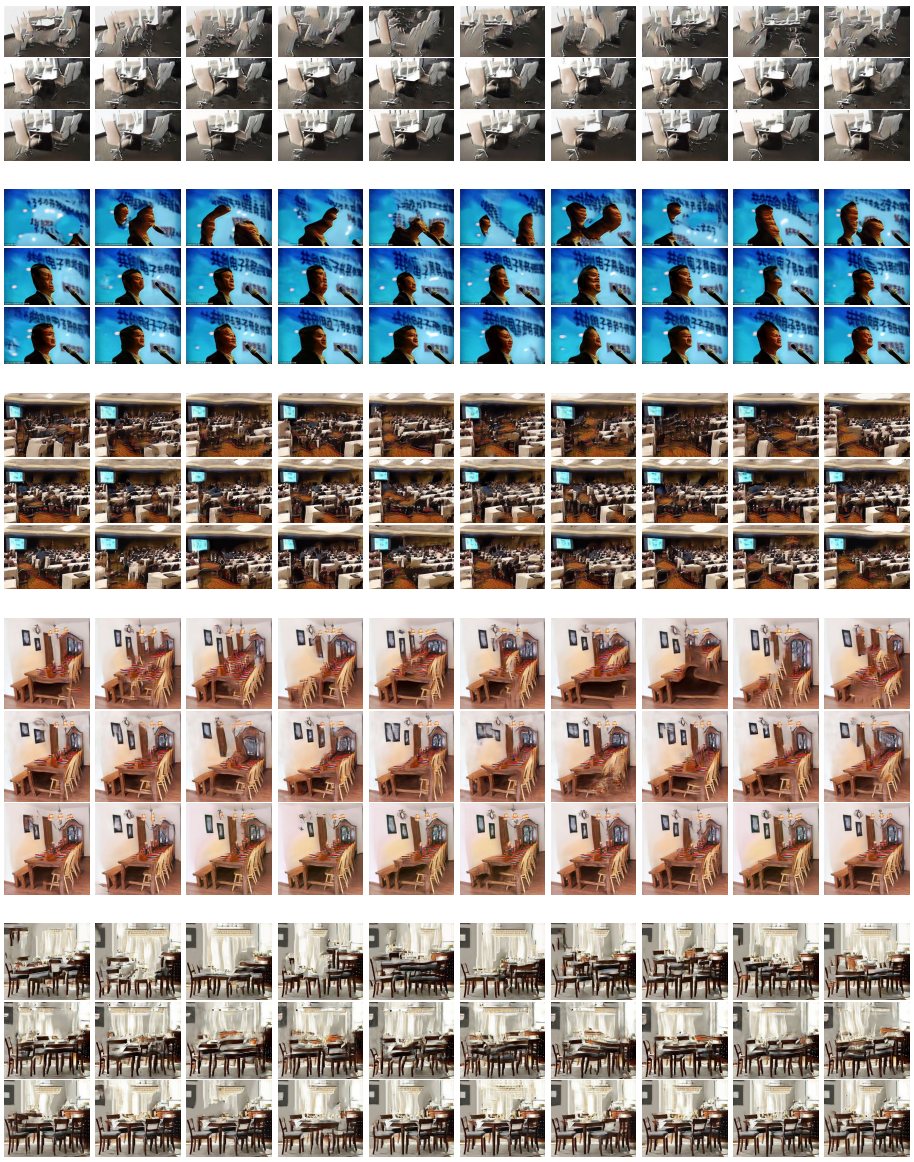


Figure 27: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

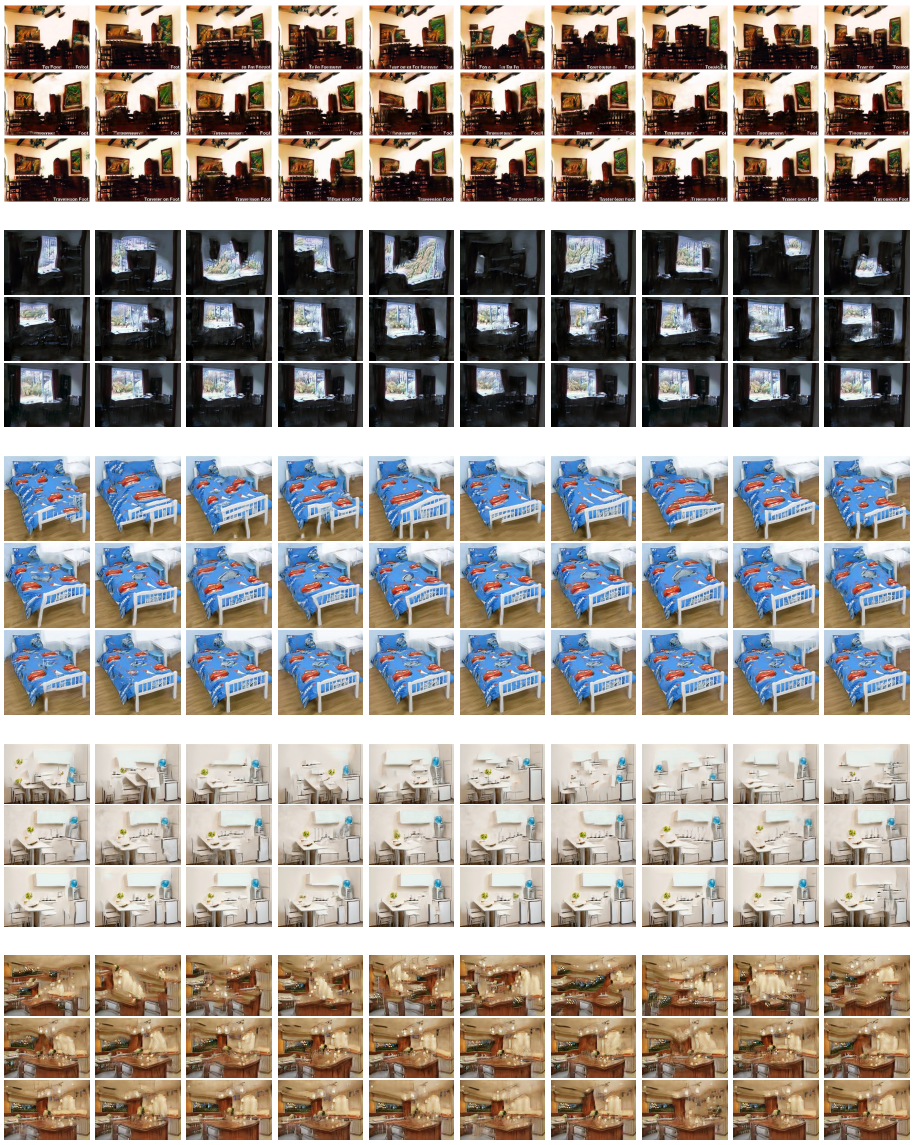


Figure 28: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

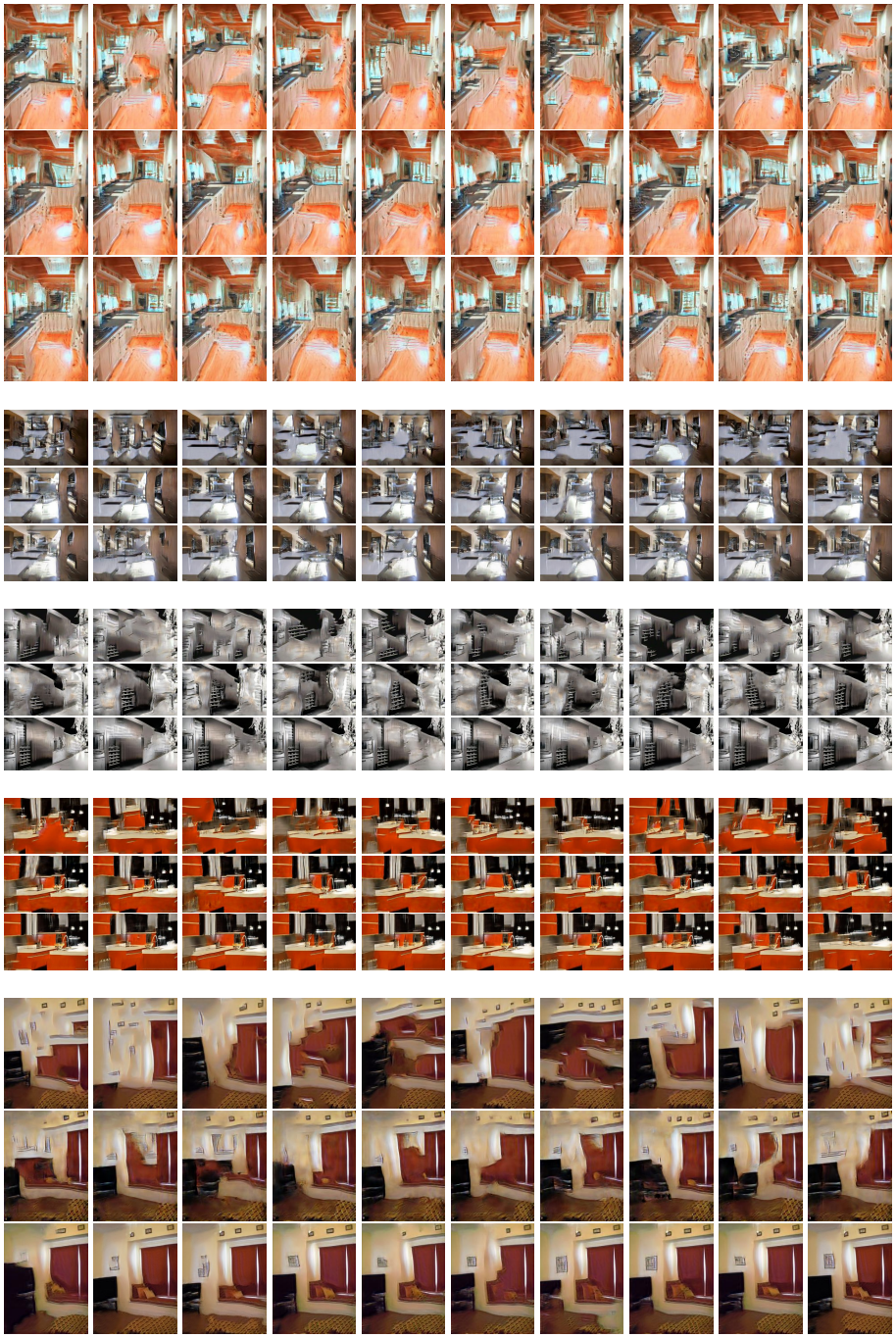


Figure 29: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

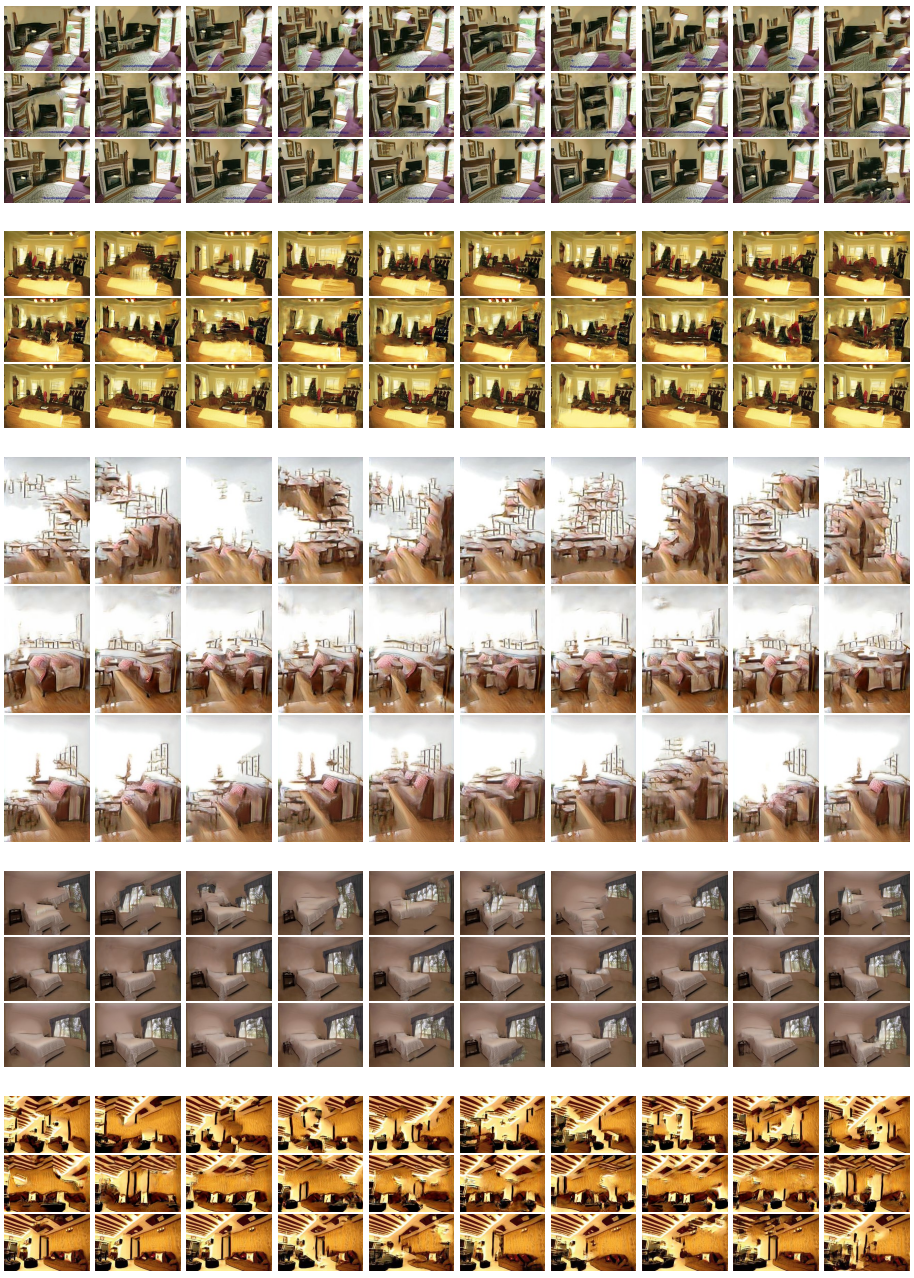


Figure 30: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

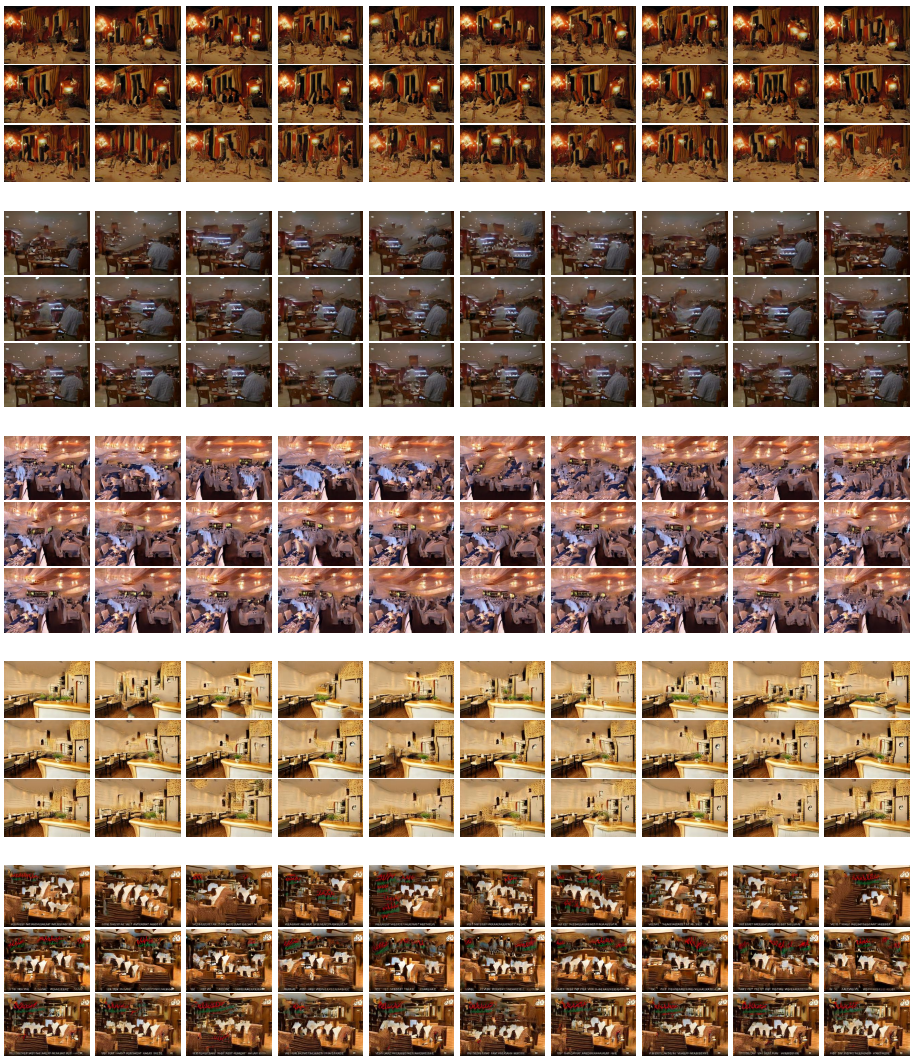


Figure 31: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

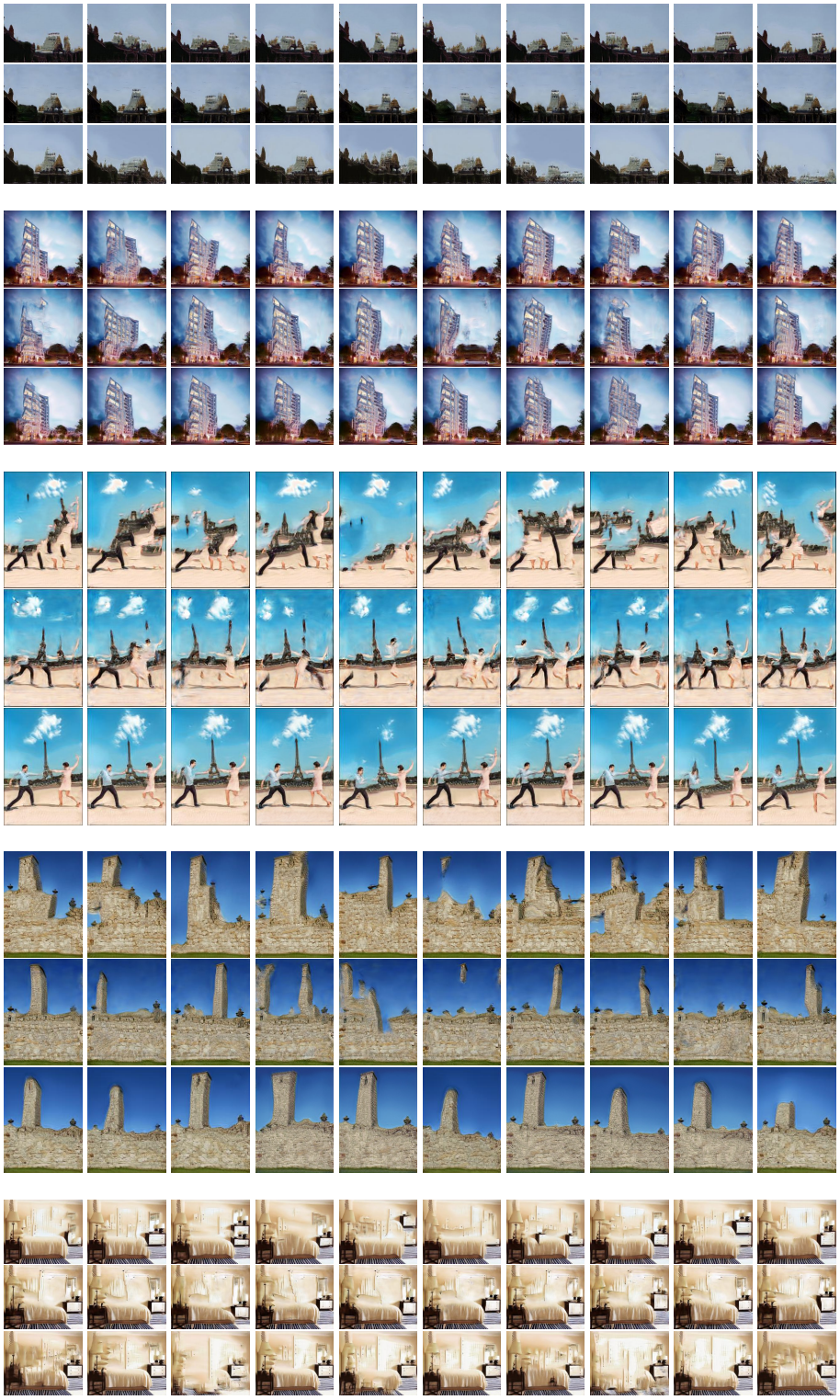


Figure 32: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

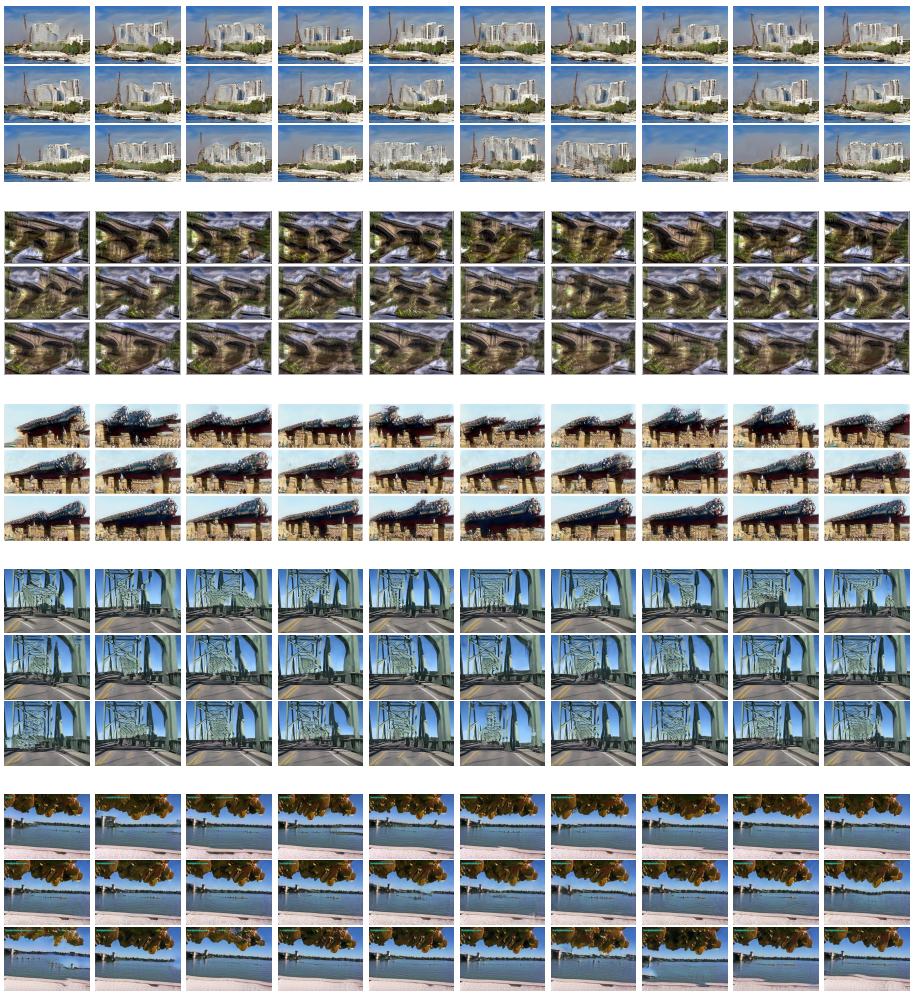


Figure 33: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *LSUN50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

9 COMPARISON OF SINGAN, CONSINGAN, EXSINGAN ON *ImageNet50*

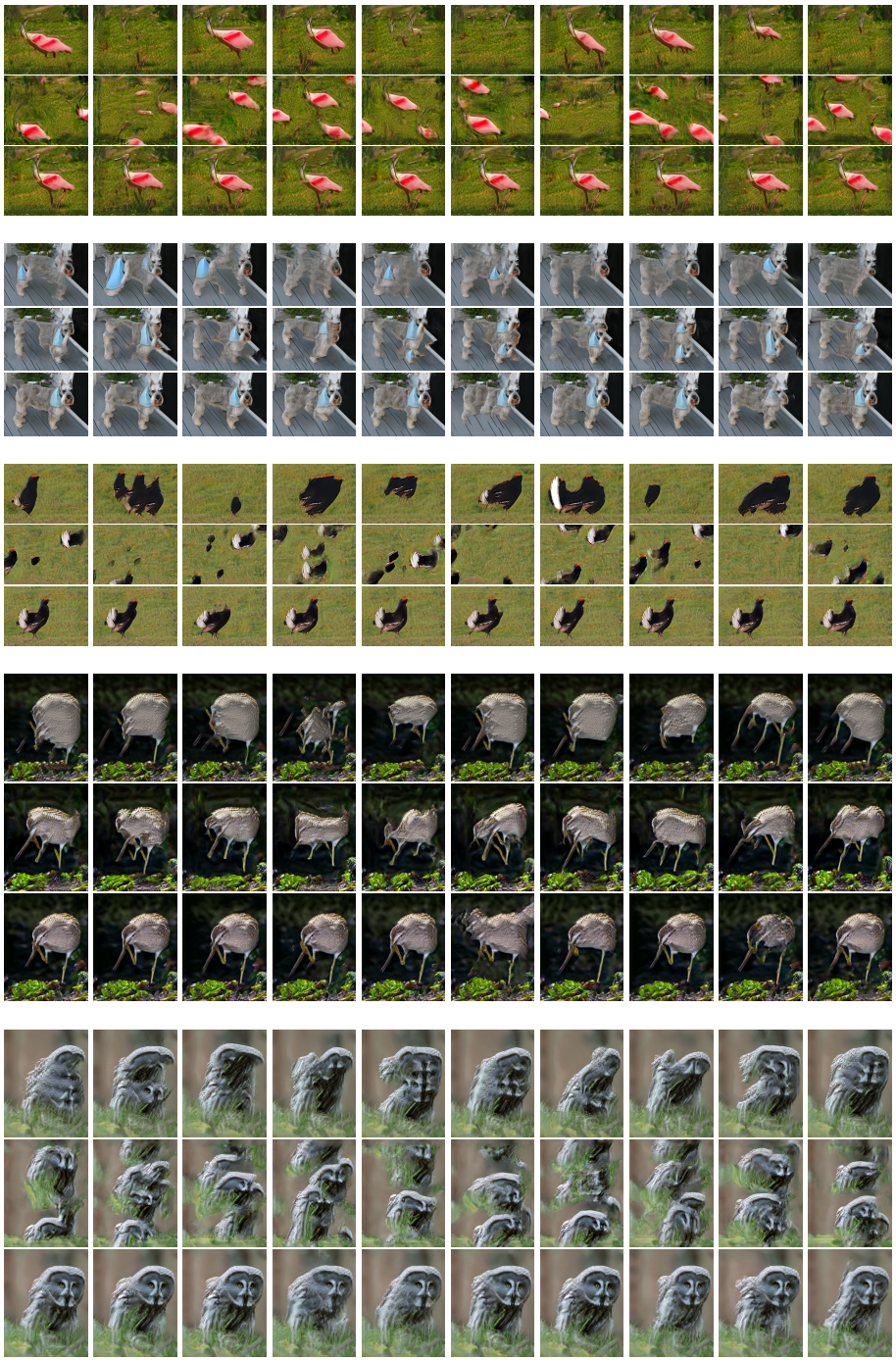


Figure 34: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

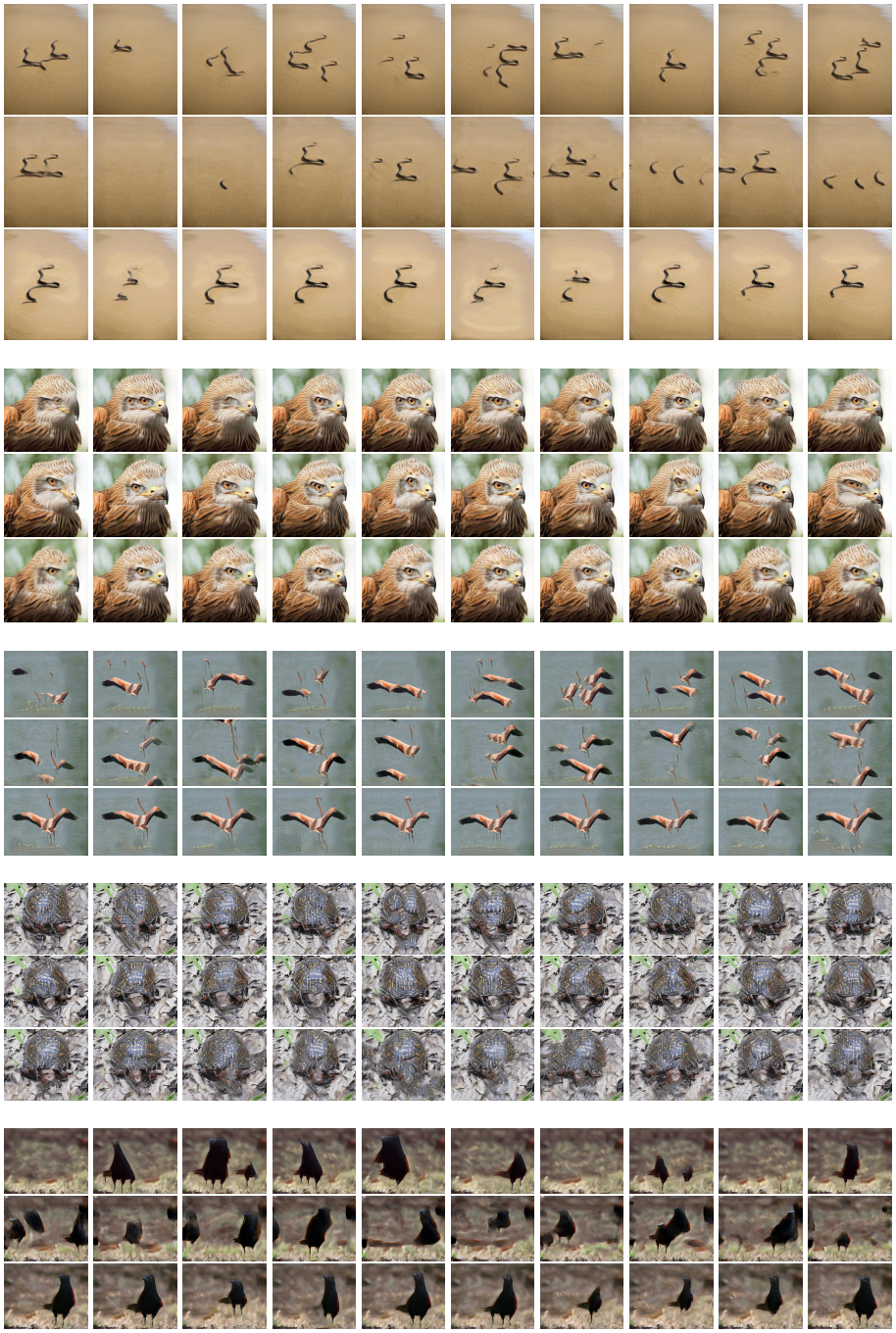


Figure 35: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

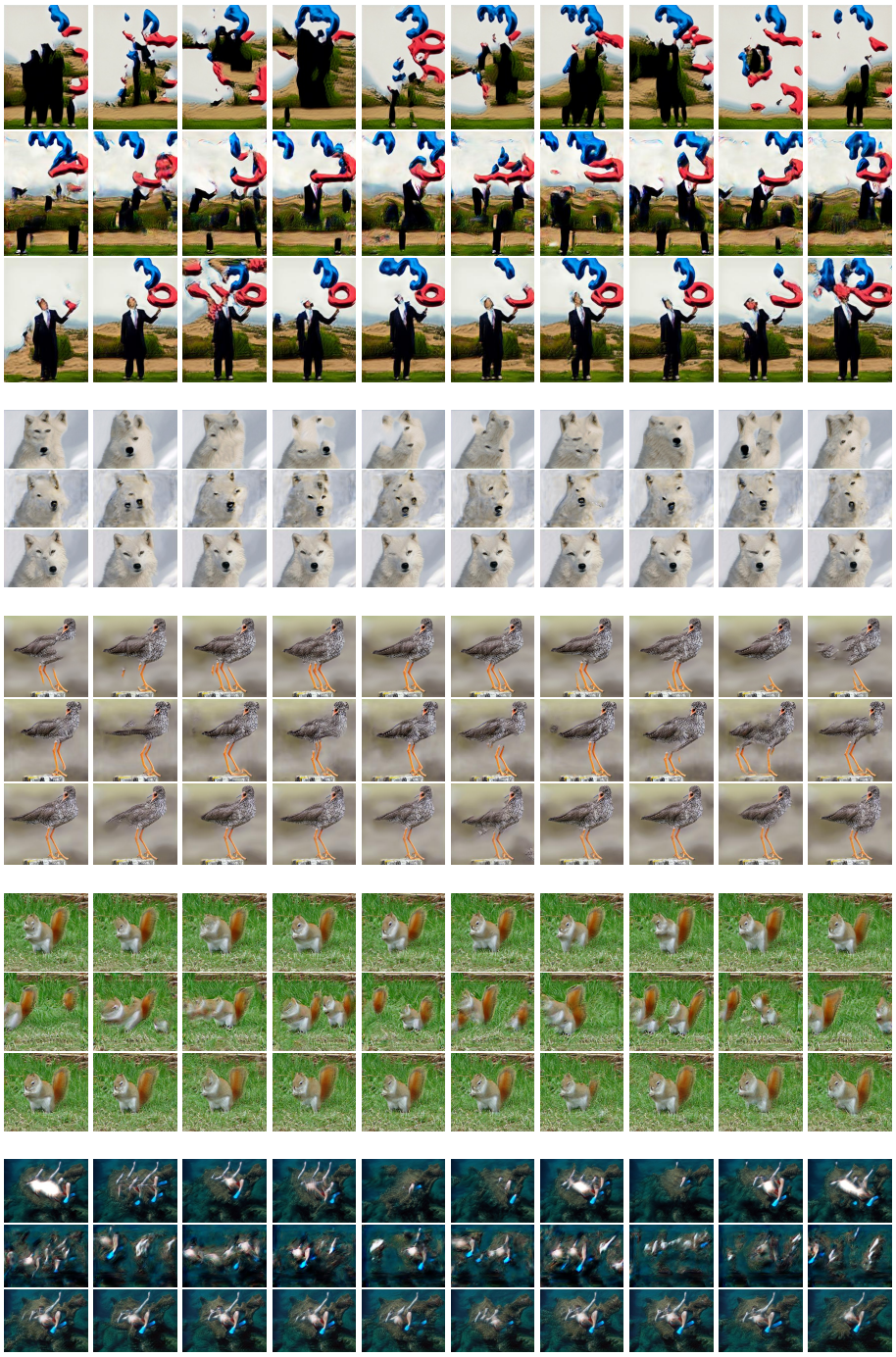


Figure 36: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

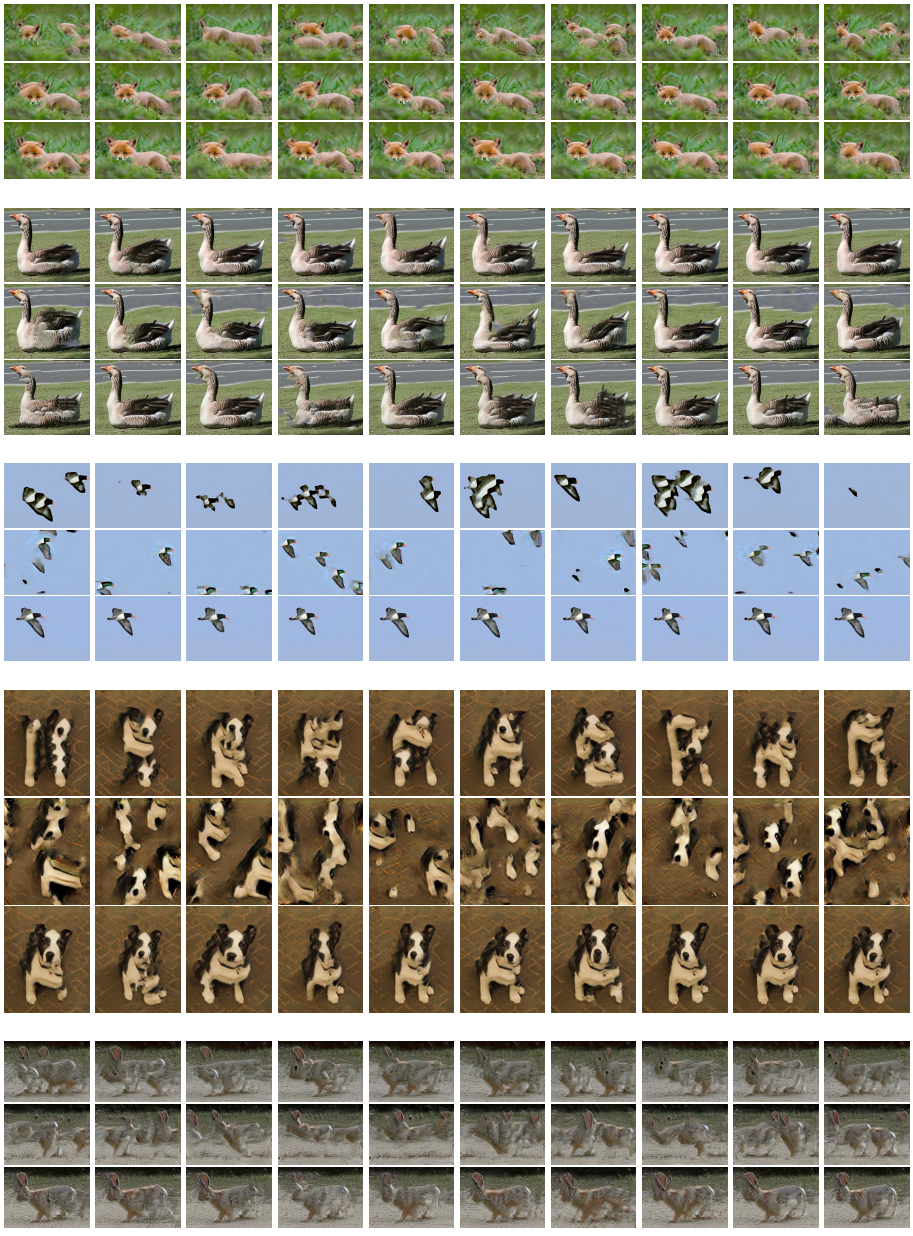


Figure 37: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

Figure 38: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

Figure 39: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

Figure 40: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

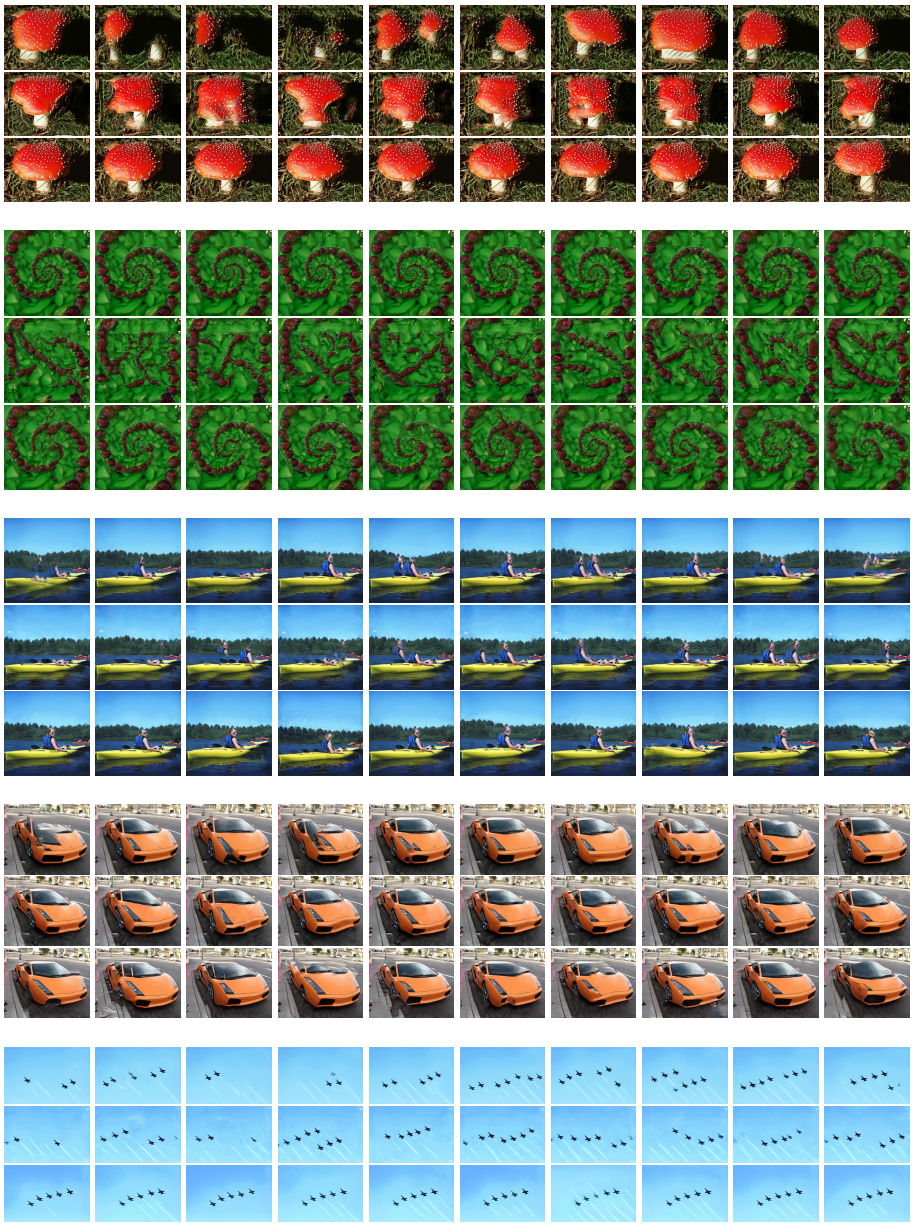


Figure 41: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

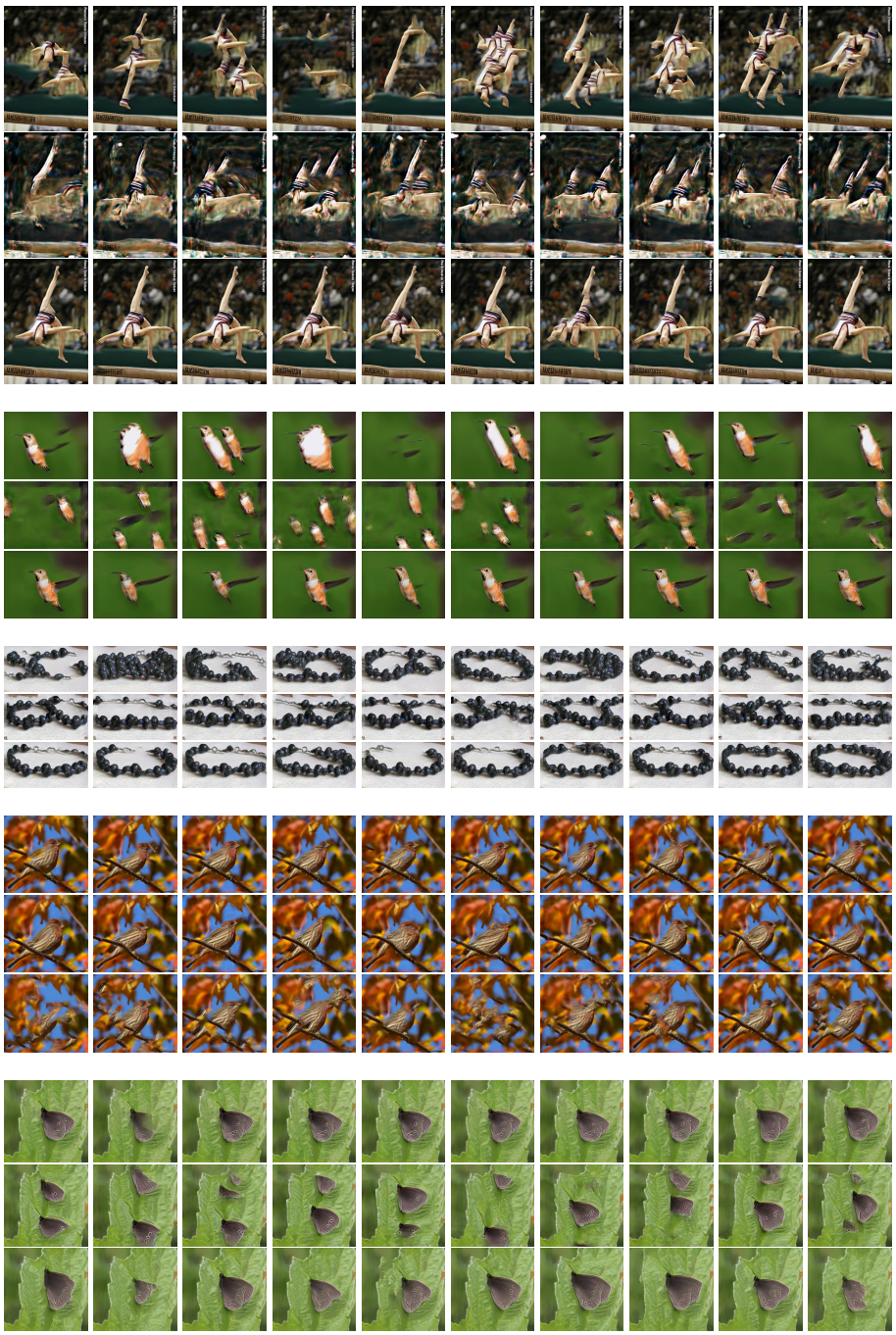


Figure 42: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

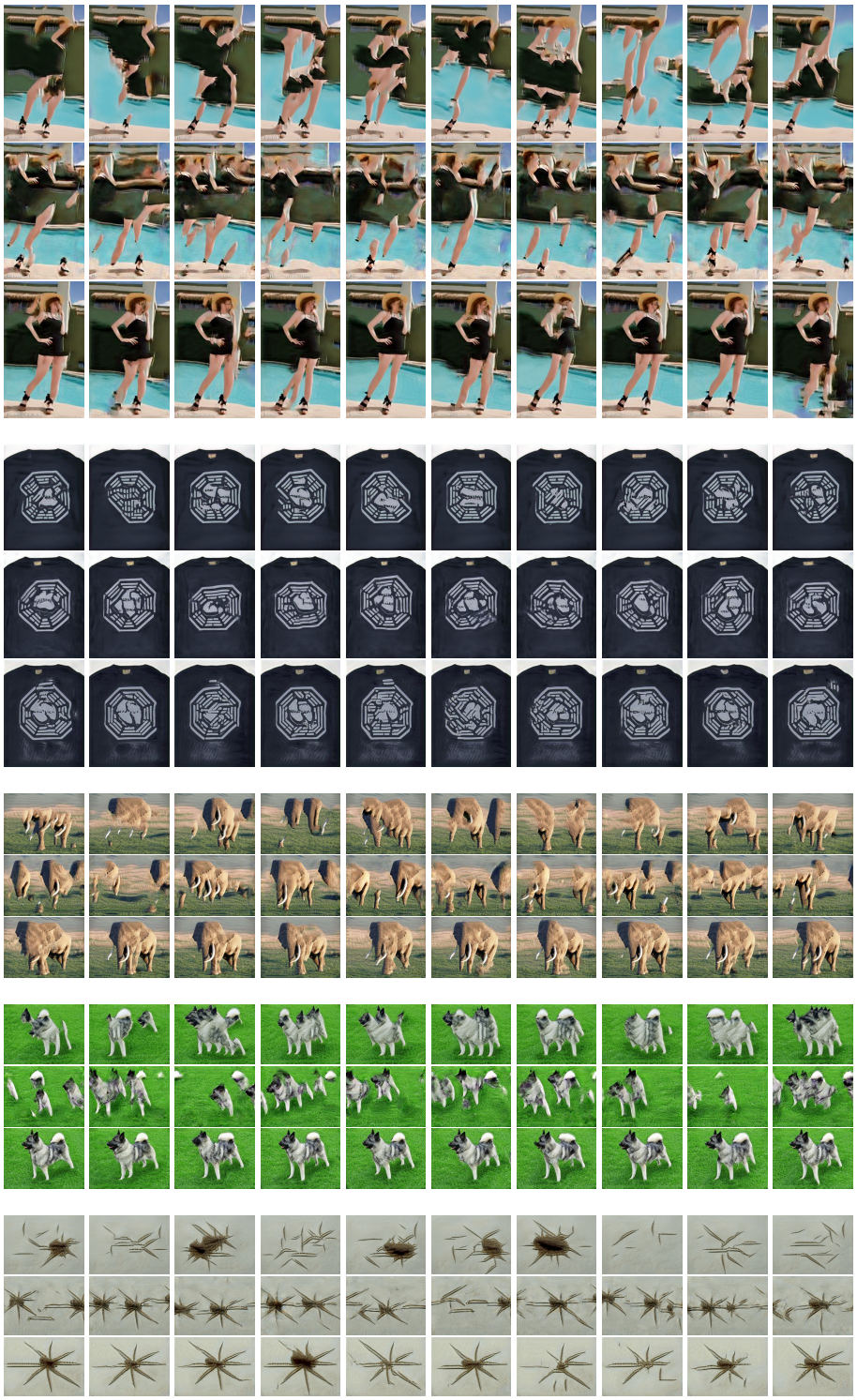
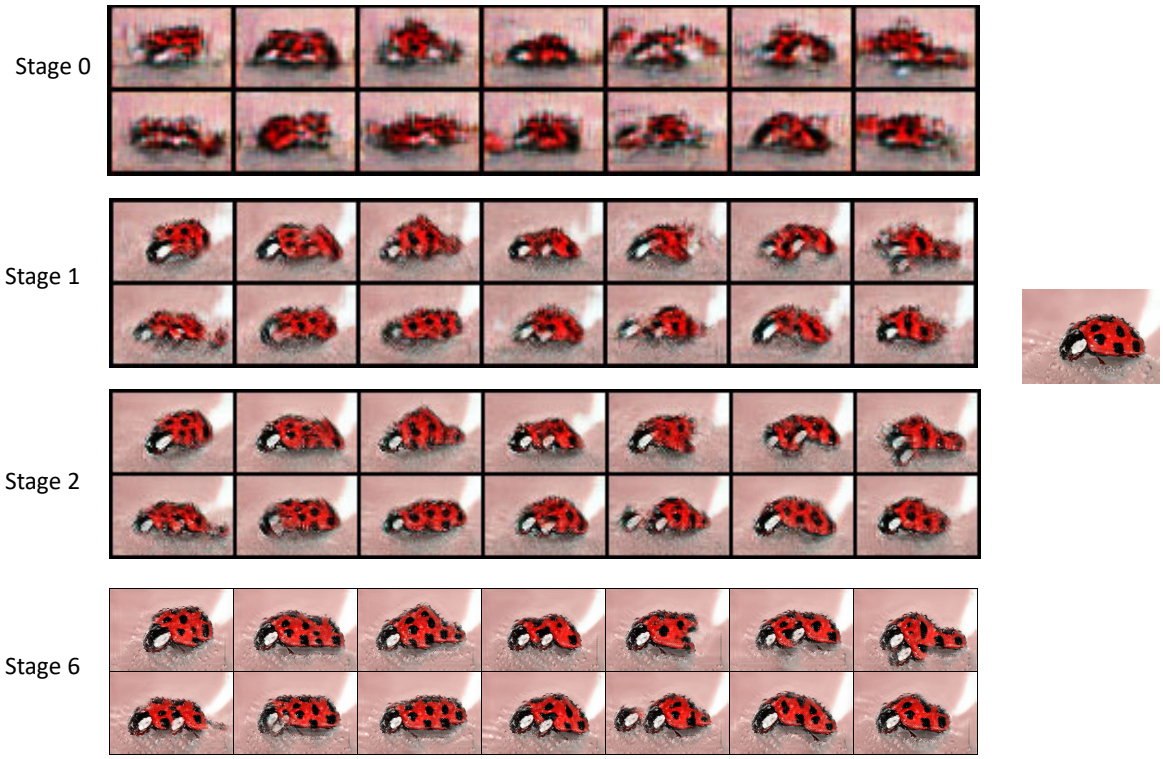


Figure 43: Comparison of SinGAN, ConSinGAN, and ExSinGAN on *ImageNet50*. For each part, the first, second and third row syntheses are respectively from SinGAN, ConSinGAN, and ExSinGAN.

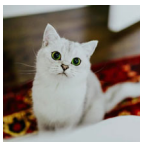


10 OUTPUT OF DIFFERENT STAGES

Stage 0 represents structural generator, stage 1 and Stage 2 are semantic generators.

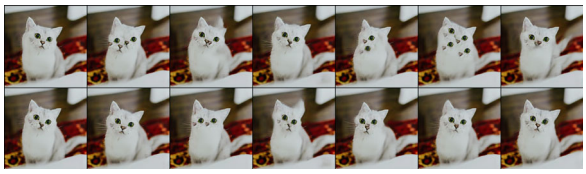
Stage 0

Stage 1

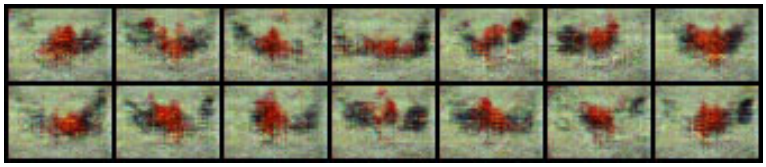


Stage 2

Stage 6



Stage 0



Stage 1

Stage 2

Stage 6

Stage 0

Stage 1

Stage 2

Stage 6