Appendix

A. Implementation Details of Styleformer

We implemented our Styleformer on top of the StyleGAN2-ADA Pytorch implementation [1]. Most of the details have not changed except generator architecture. We used CIFAR-10 tuning version of StyleGAN2-ADA, which means disabling style mixing regularization, path length regularization, and residual connections in D when training. We also fixed mapping network's depth to 2. We used bilinear filtering in all upsampling layer used in Styleformer. We use data augmentation pipeline suggested in StyleGAN2-ADA, and did not use mixed-precision training for all experiments.

For Styleformer encoder, we add bias and noise at the end of the encoder block, then performing leaky RELU with $\alpha = 0.2$. After passing several encoder blocks, we reshaped it to the form of square feature map, i.e., Unflatten. We then proceed bilinear upsample operation as we said in the Section 3.1, but also convert these reshaped output for each resolution into an RGB channel, using ToRGB layer. We upsample each of RGB output and add to each other, creating an output-skip connection generator, similar to StyleGAN2 generator. Originally ToRGB layer converts high-dimensional per pixel data into RGB per pixel data via 1×1 convolution operation, which we replace it to same operation, linear operation. We initialize all weights in Styleformer encoder using same method used in Pytorch linear layer. Unlike StyleGAN2-ADA, we employ weight demodulation also in ToRGB layer. We perform all experiments with 4 Titan-RTX using Pytorch 1.7.1. All of our experiments presented in the paper including failure spent about four months.

As in Section 4.1, the number of the Styleformer encoder and hidden dimension size for each resolution can be chosen as hyperparameters. We call these two hyperparameters as "Layers" and "Hidden size", respectively.

Low-Resolution Synthesis with Styleformer For low-resolution synthesis experiment in Section 4.1, we use pure Styleformer. Each Layers and Hidden size used for CIFAR-10, STL-10, and CelebA are shown in 1. We trained Styleformer for 65M, 92M, 25M at CIFAR-10, STL-10 and CelebA, respectively.

For CIFAR-10 experiment, we use 50K images (32×32) at the training set, without using the label. For STL-10 experiment, we resize 96×96 image datasets to 48×48, and using 5k training images, 100k unlabeled images together as in [2] we change the size of the constant input from 8×8 to 12×12 to generate an image with a size of 48×48.

[1] https://github.com/NVlabs/stylegan2-ada-pytorch

Table 1. Details of Styleformer hyperparameters at low resolution synthesis. This model setting match with performance result at Table 2 in paper.

Datasets	Layers	Hidden size	FID
CIFAR-10	{1,3,3}	{1024,512,512}	2.82
STL-10	{1,2,2}	{1024,256,64}	15.17
CelebA	{1,2,1,1}	{1024,256,64,64}	3.92

For CelebA dataset, we use 200k unlabeled face images of the Align and Cropped version, which we resize to 64×64 resolution as in [2]. We start at 8×8 constant, as we train CIFAR-10 dataset.

Ablation study details As we said in paper, we use small version of Styleformer with CIFAR-10 dataset for ablation study. In more detail, we use 1,2,2 for Layers, 256, 64, 16 for Hidden size, trained for 20M images.

Number of head experiment We conduct experiment about number of heads effect using one layer Styleformer, as said in Figure 3 in paper. One Layer Styleformer is a model that starts with 32×32 learned constant and only have one Styleformer encoder with hidden dimension size 256. We trained the model for 20M images, same as ablation study.

B. Attention map analysis

Post-Layer normalization We analyze the results of the attention map experiment on layer normalization. We propose in Section 3.2 that if we perform layer normalization at the end of Styleformer encoder, it breaks the attention map. Figure 1 shows that if layer normalization is located at the end of the encoder, the attention map has not been properly learned. Therefore, we position layer normalization to the front of the encoder (i.e. after applying style modulation) so that the attention map can effectively learn the relationship between pixels. The experiment is all conducted on CIFAR-10, using small version of Styleformer, same as ablation study.

Demodulation for Query and Key As in Section 3.3, we demonstrate that when an attention map is created with Q, K from input scaled by style vector, specific value in attention map becomes very large. We show the attention map without demodulation operation to Q and K in Styleformer at Figure 1. We can see that without demodulation operation, attention is heavily concentrated on a particular pixel, preventing the attention operation from working properly.

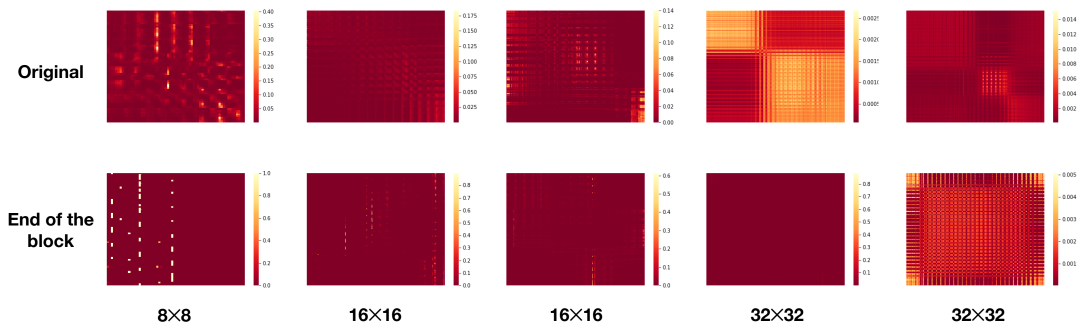

Figure 1. Comparison of attention map experimented based on Layer Normalization

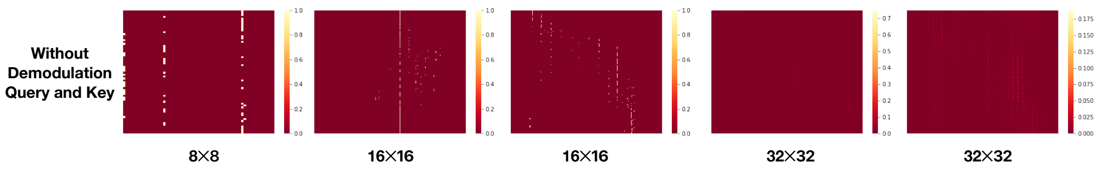

Figure 2. Attention map without demodulation operation to query and key. Comparing with Figure 1 upper row, we can see specific large value in attention map.

We overcome this problem with demodulation operation to Q, K, before creating attention map.

C. Intuition of Increased Multi-Head Attention

Transformer is designed for natural language processing (NLP). It is difficult to use original Transformer in the field of image generation. A convolution network can efficiently generate images, unlike a linear layer, because it proceeds both operations between pixels using kernels and between channels. Although not a convolution network structure, [4] has shown that architecture with pixel-to-pixel and channel-to-channel operations can be efficient in learning information about images.

Unlike the original convolution operation, in MobileStyleGAN [1], they show good performance in image generation even by separating the interpixel and interchannel operations with depthwise separable convolution.

Main operation of Transformer can be divided into a interpixel operation (i.e. pixel section) and a interchannel operation (i.e. channel section) and it can be expressed by the following formula.

$$A_i = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}}), \tag{1}$$

$$head_i = A_i V_i, \tag{2}$$

$$Multihead(Q,K,V) = Concat(head_1, \ldots, head_k)W^O, \tag{3}$$

In above formulation, Q_i, K_i, V_i is query, key, value for each head. d is the dimension of $(query, key, value)$, k is the number of heads, and d_k is d/k. A_i is attention map and W^O is the parameters of a linear layer that integrates multi-heads.

Pixel section corresponds for the self-attention operation between pixels (i.e., Equation 2), and channel section corresponds for integration of multi-head with linear layer, which operates between channels (i.e., Equation 3). In a Transformer, the pixel section is slightly different from depthwise convolution. In depthwise convolution, kernel weights exist for each channel, but in Transformer, attention map A acts like one huge kernel, which means applying equal kernel weight to all different channels in V. It is difficult to create a powerful generator using a Transformer because the same attention kernel is applied for each channel, unlike the generator using depthwise separable convolution.

Using increased multi-head attention, this problem could be overcome. We can generate various attention maps (i.e., kernels) by increasing the number of heads. However, increasing the number of heads inevitably leads to smaller depth, where depth is hidden channel dimension divided by the number of heads. Since depth is a dimension used to create the attention map, there exists a minimum depth required. However, due to differences in the properties of pixels and tokens, the required depth size in computer vision is smaller than NLP.

In the field of NLP, a single token has a lot of information, so the required depth dimension, which represents the token, must be large, but one pixel in an image has less information than a token, which means that the required depth is smaller than a traditional Transformer. As described in the caption of Figure 3, the left graph shows a hidden dimension of 256, and the right graph shows 32. In the left graph, until the number of heads is 8 (depth is 32), performance increases when the number of heads grows up, but after that, even if the number of heads is increased, the performance decreases because the depth is less than 32. Similarly, in the right graph, performance is best when the number of heads is 1 (depth is 32), and after that, performance degrades because the depth is less than 32. From these results, we demonstrate that increasing the number of heads too much results in poor performance, and fixing the depth to 32.

Meanwhile, the channel section is a layer to integrate the multi-head together, where the linear layer is exactly the same operation as the 1×1 convolution, i.e., pointwise convolution. Unlike convolution, the attention kernel is a kernel generated by the input itself, so it can create a more dense kernel, and it is advantageous to capture global features because it considers the relationship between all pixels. Therefore, by enhancing multi-head attention, the Styleformer can play a more powerful role as depthwise separable convolution, enabling generate high-quality images.

D. Demodulation for Encoder Output

As described in Section 3.3, self-attention conducts more operations than the convolution operation, making the demodulation process more complex. We show how standard deviation of encoder output can be derived (Demodulation for Encoder Output at Section 3.3), and the effect of the number of pixel in output standard deviation.

Derivation of encoder output standard deviation After demodulation operation to Q, K and V, Styleformer encoder performs style modulation to input V, weighted sum of V with attention map, and then performs linear operation. Let's consider the attention map matrix as A, and the weight matrix of linear operation as w. Since the matrix multiplication is associative, statistics of the output are the same even if the linear operation is calculated before multiplication with attention map :

$$output = [A(s_j \cdot V)]w = A[(s_j \cdot V)w]. \tag{4}$$

From now on, we think the linear operation is conducted before the multiplication with the attention map. Therefore, same as demodulation for query, key, and value, style modulation to V can be replaced to scaling linear weights:

$$w'_{jk} = s_j \cdot w_{jk}, \tag{5}$$

where s_j scales jth feature map of V, and k enumerates the flattened output feature map. Assuming that input V have a unit standard deviation, the standard deviation of output after linear operation can be derived as follows:

$$\sigma_k = \sqrt{\sum_j {w'_{jk}}^2}. \tag{6}$$

Finally, this output is multiplied with the attention map matrix A. When the attention score vector for lth pixel is expressed as $A_{l\cdot}$, standard deviation of encoder output is as follows:

$$\sigma'_{lk} = \sqrt{\sum_{\cdot} {A_{l\cdot}}^2 \cdot \sum_j {w'_{jk}}^2}. \tag{7}$$

Effect of the number of pixel Scaling the encoder output feature map k with $1/\sigma''_k$ where $\sigma''_k = \sqrt{\sum_j {w'_{jk}}^2}$, the standard deviation of output activations will be

$$\sigma_{lk} = \sqrt{\sum_{\cdot} {A_{l\cdot}}^2}. \tag{8}$$

Since the attention map A is matrix after softmax operation, $\sum_{\cdot} A_{l\cdot} = 1$. Assuming $A_{l\cdot}$ are i.i.d random variables from normal distribution with mean $\frac{1}{n}$ and variance $\frac{1}{n^2}$, and n denotes the number of pixel, ${\sigma_{lk}}^2$ can be derived as follows:

$${\sigma_{lk}}^2 = \sum_{\cdot} {A_{l\cdot}}^2 = \sum_{\cdot} (A_{l\cdot} - \frac{1}{n})^2 + \frac{1}{n}, \tag{9}$$

using $\sum_{\cdot} A_{l\cdot} = 1$. Then $(A_{l\cdot} - \frac{1}{n})$ become random variables from normal distribution with zero mean and variance $1/n^2$. Based on the property of normal distribution, $\sum_{\cdot} (A_{l\cdot} - \frac{1}{n})$ follows gamma distribution with shape parameter $\frac{n}{2}$ and scale parameter $\frac{2}{n^2}$. Therefore, using Chebyshev inequality, we have

$$\Pr(|\sum_{\cdot} (A_{l\cdot} - \frac{1}{n})^2 - \frac{1}{n}| \le \frac{1}{n}) \ge 1 - \frac{2}{n}, \tag{10}$$

meaning σ_{lk} approaches zero when the number of pixels(i.e., n) increase.

E. Implementation details of Styleformer-L

We introduce Styleformer-L, a model with Linformer applied to self-attention operation of Styleformer. When applying Linformer, we fix k (i.e., projection dimension for key, value) to 256, and apply to the encoder block above

32×32 resolution. For projection parameter sharing effect, we used Key-value sharing. It means we create single E projection matrix for each layer that applies equally to the key, value of each head. We project key and value to k dimension after demodulation for key and value, i.e., before creating attention map and weight sum of value. When using Linformer, to prevent augmentation leaking, we clamp augmentation probability to 0.7.

CelebA We conduct CelebA experiment using Styleformer-L in the same setting as CelebA using pure Styleformer for fair comparision (Table 3 in paper). We use $\{1, 2, 1, 1\}$ for Layers, $\{1024, 256, 64, 64\}$ for Hidden size, and training for 25M images.

LSUN-Church We use $\{1, 2, 1, 1, 1\}$ for Layers, $\{1024, 256, 64, 64, 64\}$ for Hidden size, training for 40M images.

F. Implementation Details of Styleformer-C

We introduce Styleformer-C, which combines Styleformer and StyleGAN2. We generate low-resolution parts (up to 32×32) of image using Styleformer Encoder, and the rest of the resolutions parts of image are generated by applying StyleGAN2 block. Fundamentally, in Styleformer-C, the detail of Styleformer part is the same as that of Appendix A, and the StyleGAN2 part is the same as that of StyleGAN2 [3] implementation. We experiment with CLEVR and Cityscapes which is high-resolution multi-object or compositional scene datasets with a image size of 256×256, and we also experiment AFHQ-Cat which is a high-resolution single-object dataset with a image size of 512×512. We performed all training runs on NVIDIA 2 Tesla V100 GPUs.

CLEVR and Cityscapes In CLEVR, we start at 8×8 learned constant input just like the default setting and used $\{1, 2, 1, 1, 1, 1\}$ for Layers, $\{1024, 256, 256, 256, 256, 128\}$ for Hidden size, training for 10M images. Here, what Layers means in StyleGAN2 is a block which has two convolution operations and what Hidden size means in StyleGAN2 is the number of channels. In Cityscapes, we apply StyleGAN2 in the same way as CLEVR. We also set Layers and Hidden size same as CLEVR experiment. Furthermore, we train Styleformer-C for 36M images.

AFHQ-Cat Likewise, we start with 8×8 learned constant input and generate an image of 512×512 size. We set Layers to $\{1, 2, 1, 1, 1, 1\}$ and Hidden size to $\{1024, 256, 256, 256, 256, 64\}$ to train Styleformer-C for 9M images.

G. Visual Samples

We show various samples generated with Styleformer, Styleformer-L and Styleformer-C in this section.

References

[1] Sergei Belousov. Mobilestylegan: A lightweight convolutional neural network for high-fidelity image synthesis, 2021. 2

[2] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan, 2021. 1

[3] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020. 4

[4] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021. 2

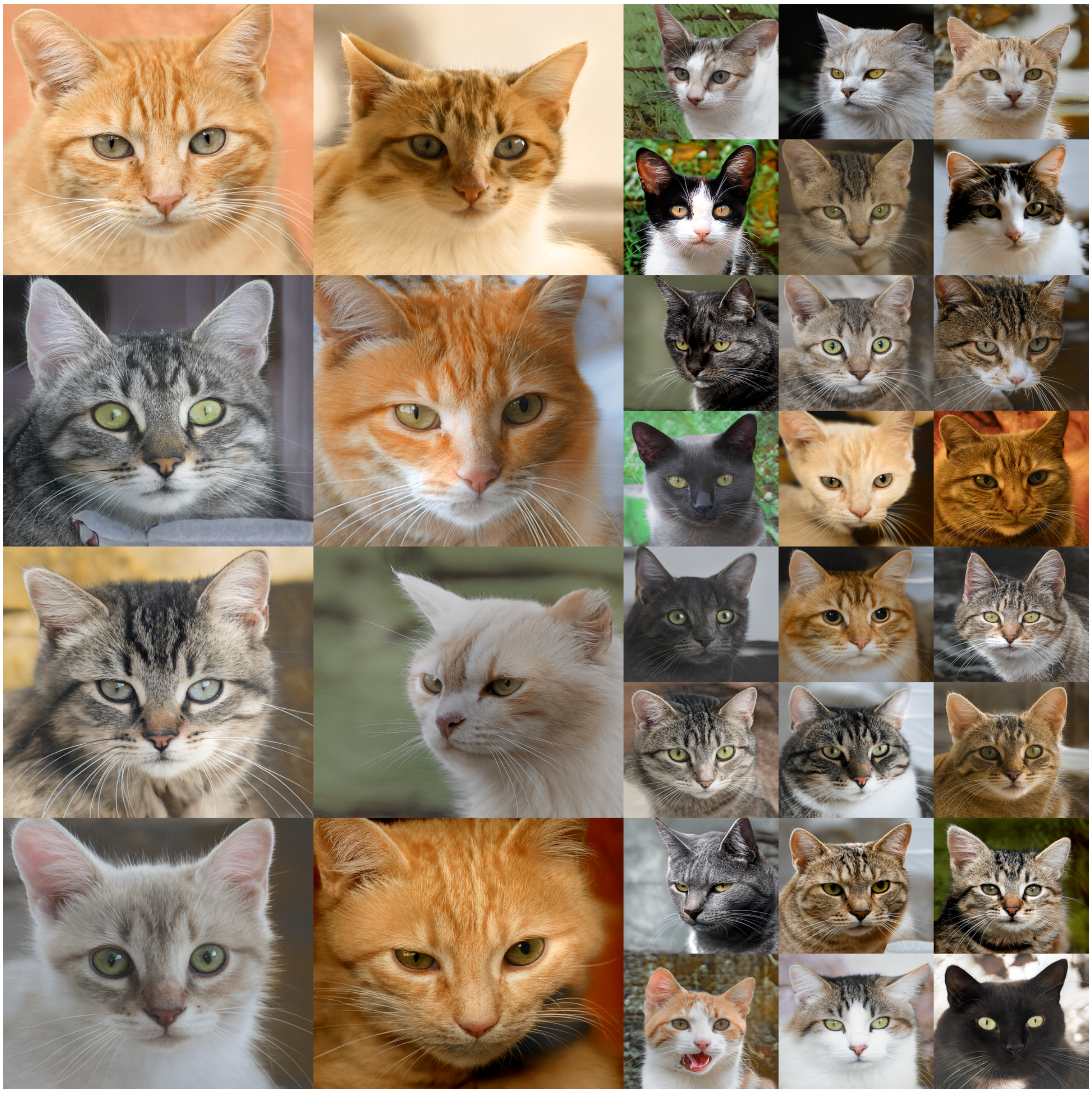
Figure 3. High-resolution samples generated by Styleformer-C on AFHQ-Cat.

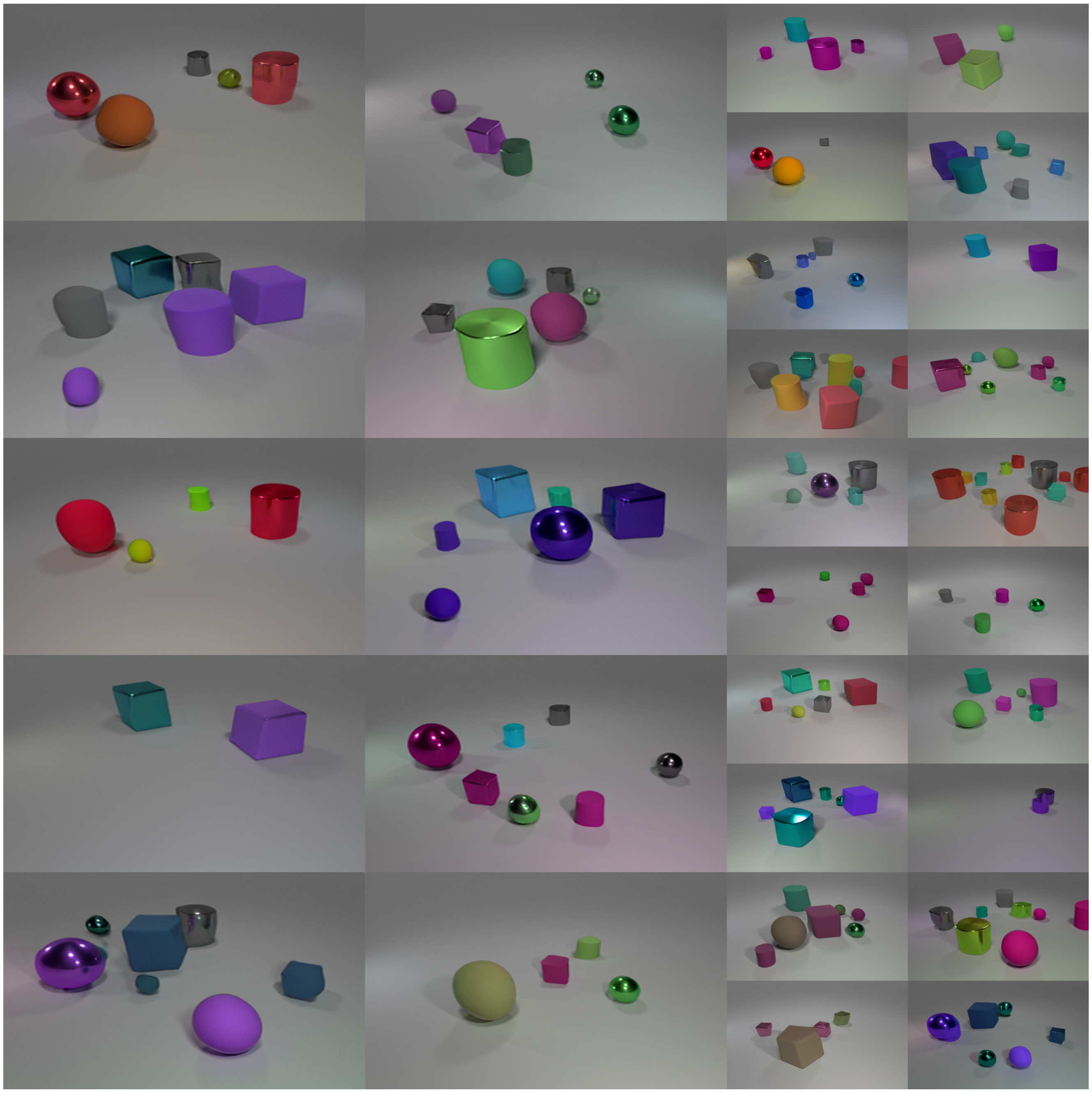

Figure 4. High-resolution samples generated by Styleformer-C on CLEVR.

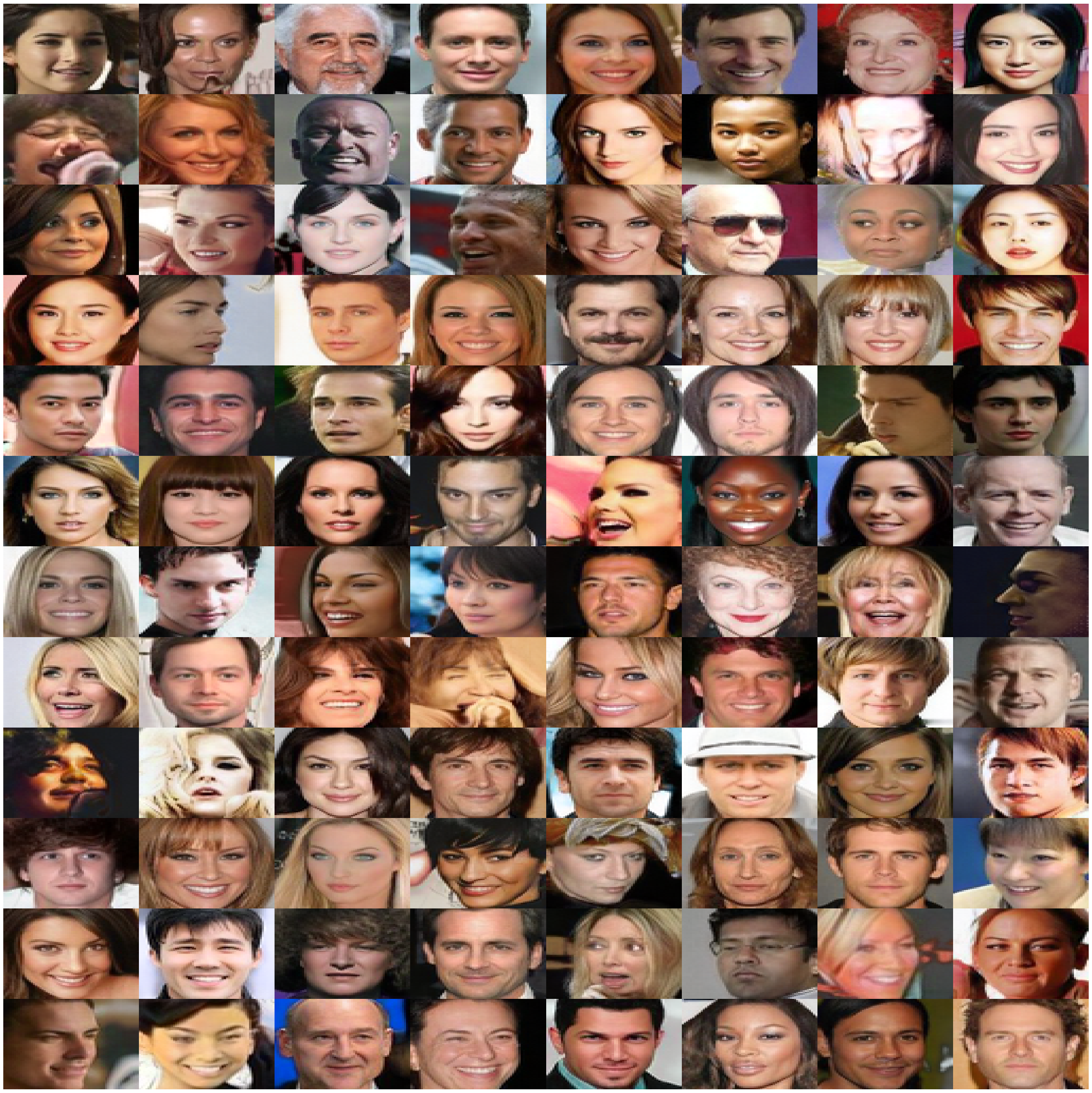

Figure 5. High-resolution samples generated by Styleformer-L on CelebA.

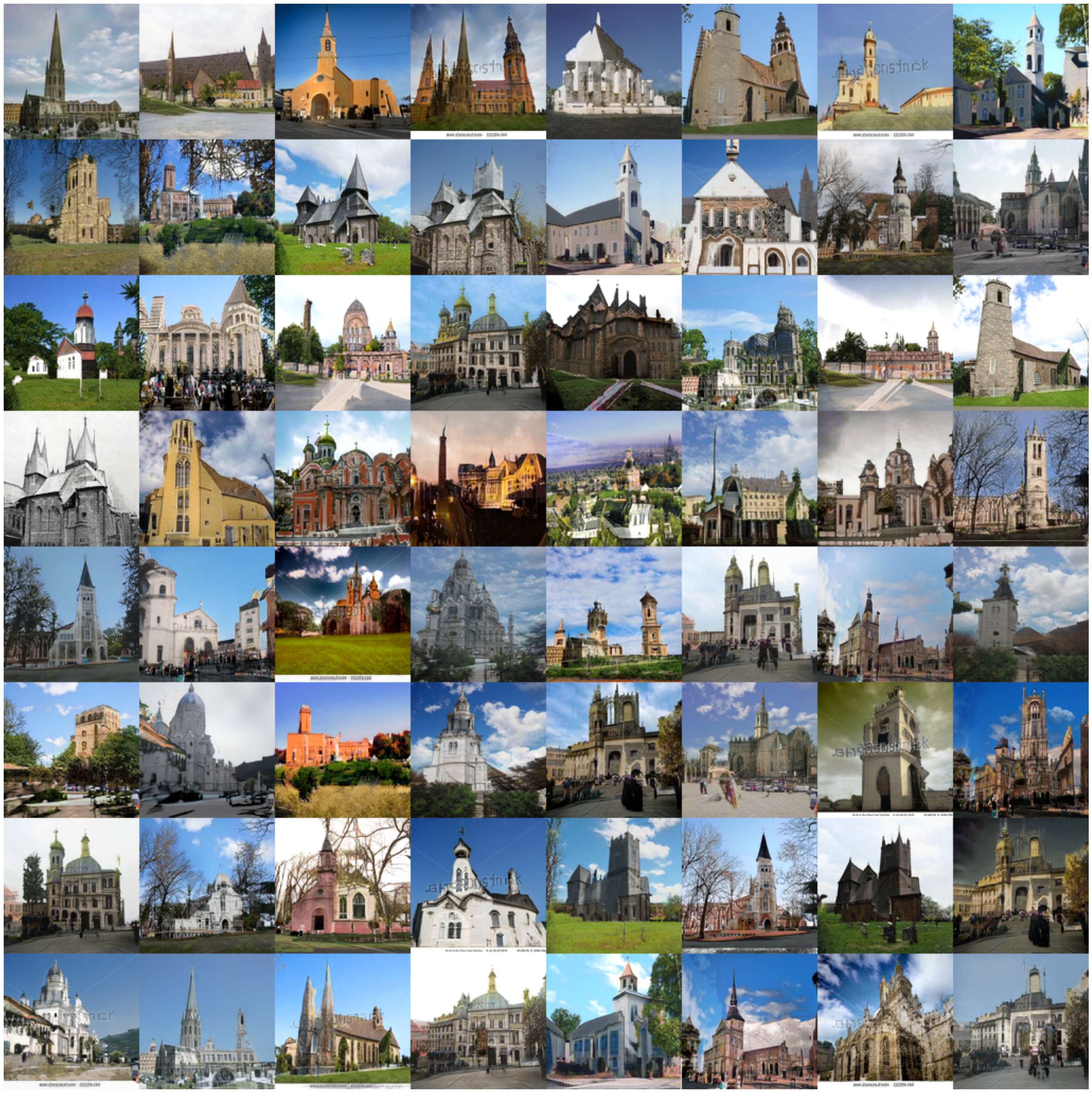
Figure 6. High-resolution samples generated by Styleformer-L on LSUN-church.

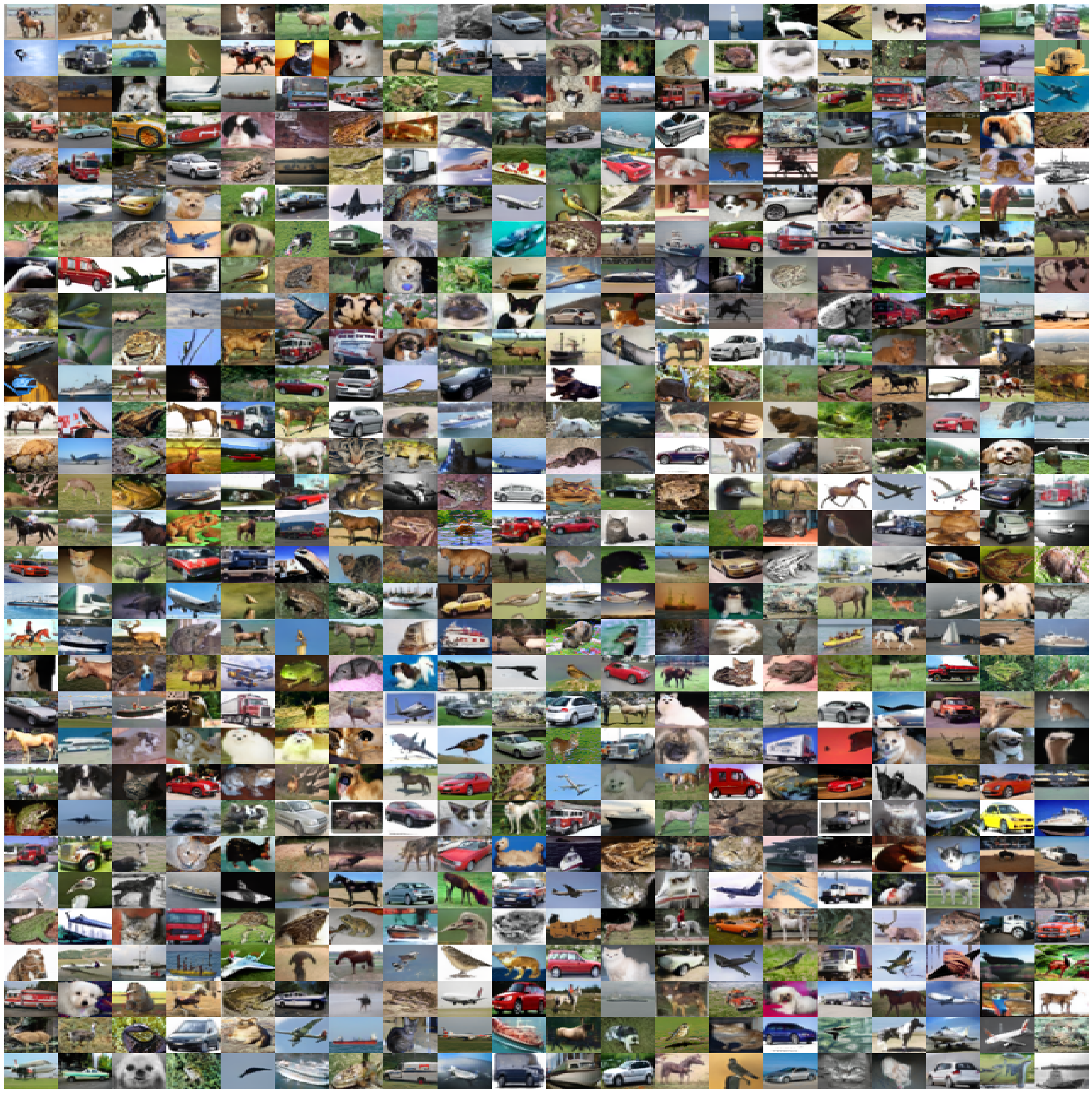
Figure 7. Samples generated by Styleformer on CIFAR-10.