Adversarial Open Domain Adaptation for Sketch-to-Photo Synthesis Supplementary Material

Xiaoyu Xiang[1]*, Ding Liu[2], Xiao Yang[2], Yiheng Zhu[2], Xiaohui Shen[2], Jan P. Allebach[1]
[1]Purdue University, [2]ByteDance Inc.
{xiang43,allebach}@purdue.edu,
{liuding,yangxiao.0,yiheng.zhu,shenxiaohui}@bytedance.com

Contents

A. Experimental Details

In this section, we first illustrate the architectures of our framework, including generators, discriminators, and a classifier in Section A.1. Then, we present the objective functions for training them in Section A.2. The training settings of each dataset and additional implementation details are described in Section A.3 and Section A.4.

A.1. Architecture

Note that our proposed solution is not limited to certain network architecture. In this work, we select the CycleGAN [22] as a baseline to illustrate the effectiveness of our proposed solution. Thus we only modify the G_p into a multi-class generator and keep the rest structures unchanged, as introduced below.

Photo-to-Sketch Generator G_s We adopt the architecture of the photo-to-sketch generator from Johnson *et al.* [8]. It includes one convolution layer to map the RGB image to feature space, two downsampling layers, nine residual blocks, two upsampling layers, and one convolution layer that maps features back to the RGB image. Instance normalization [19] is used in this network. This network is also adopted as the sketch extractor for the compared method in the main paper Section 3.1.

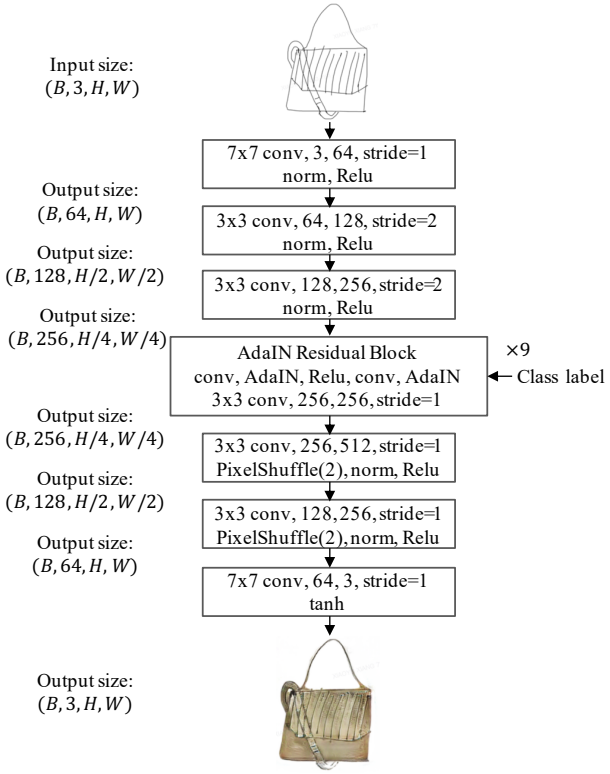

Figure 1: The architecture of our multi-class sketch-to-photo generator.

Multi-class Sketch-to-Photo Generator G_p The overall structure of this network is similar to G_s: a feature-mapping convolution, two downsampling layers, a few residual blocks, two upsampling layers, and the RGB-mapping convolution. We make the following modifications on the residual blocks and upsampling layers for the multi-class photo generation, as illustrated in Figure 1. To make the network capable of accepting class label information, we change the normalization layers of the residual blocks into adaptive instance normalization (AdaIN) [7]. The sketch input serves

*The author was with Purdue University when conducting the work in this paper during an internship at ByteDance. She is now with Facebook.

as the content input for AdaIN, and the class label is the style input ensuring that the network learns the correct textures and colors for each category. In addition, we use convolution and PixelShuffle layers [17], instead of commonly used transposed convolution, to upsample the features. The sub-pixel convolution can alleviate the checkerboard artifacts in generated photos while reducing the number of parameters as well as computations [1].

Discriminators We use the PatchGAN [11, 10] classifier as the architecture for the two discriminators in our framework. It includes five convolutional layers and turns a 256×256 input image into an output tensor of size 30×30, where each value represents the prediction result for a 70×70 patch of the input image. The final prediction output of the whole image is the average value of every patch.

Photo Classifier We adopt the architecture of HRNet [20] for photo classification and change its output size of the last fully-connected (FC) layer according to the number of classes in our training data. This network takes a 256×256 image as input and outputs an n-dim vector as the prediction result. We choose the HRNet because of its superior performance in maintaining high-resolution representations through the whole process while fusing the multi-resolution information at different stages of the network.

A.2. Objective Function

The loss for training the generator is composed of four parts: the adversarial loss of photo-to-sketch generation $\mathcal{L}_{G_s}$, the adversarial loss of sketch-to-photo translation $\mathcal{L}_{G_p}$, the pixel-wise consistency of photo reconstruction $\mathcal{L}_{pix}$, and the classification loss for synthesized photo $\mathcal{L}_{\eta}$:

$$\mathcal{L}_{GAN} = \lambda_s \mathcal{L}_{G_s}(G_s, D_s, p) + \lambda_p \mathcal{L}_{G_p}(G_p, D_p, s, \eta_s) + \lambda_{pix} \mathcal{L}_{pix}(G_s, G_p, p, \eta_p) + \lambda_{\eta} \mathcal{L}_{\eta}(R, G_p, s, \eta_s), \quad (1)$$

where

$$\mathcal{L}_{G_s}(G_s, D_s, p) = -\mathbb{E}_{p \sim P_{data}(p)}[\log_{D_s}(G_s(p))], \quad (2)$$

$$\mathcal{L}_{G_p}(G_p, D_p, s, \eta_s) = -\mathbb{E}_{s \sim P_{data}(s)}[\log_{D_p}(G_p(s, \eta_s))], \quad (3)$$

$$\mathcal{L}_{pix}(G_s, G_p, p, \eta_p) = \mathbb{E}_{p \sim P_{data}(p)}[||G_p(G_s(p), \eta_p) - p||_1], \quad (4)$$

$$\mathcal{L}_{\eta}(R, G_p, s, \eta_s) = \mathbb{E}[\log P(R(G_p(s, \eta_s)) = \eta_s | G_p(s, \eta_s))]. \quad (5)$$

Note that only the classification loss of the generated photo $G_p(s, \eta_s)$ is used to optimize the generators.

Then we update the discriminators D_s and D_p with the following loss functions, respectively:

$$\mathcal{L}_{D_s}(G_s, D_s, p, s) = -\mathbb{E}_{s \sim P_{data}(s)}[\log D_s(s)] + \mathbb{E}_{p \sim P_{data}(p)}[\log D_s(G_s(p))], \quad (6)$$

$$\mathcal{L}_{D_p}(G_p, D_p, s, p, \eta_s) = -\mathbb{E}_{p \sim P_{data}(p)}[\log D_p(p)] + \mathbb{E}_{s \sim P_{data}(s)}[\log D_p(G_p(s, \eta_s))]. \quad (7)$$

Then we calculate the classification loss of both real and synthesized photos and optimize the classifier:

$$\mathcal{L}_R(R, G_p, s, p, \eta_s, \eta_p) = \mathbb{E}[\log P(R(p) = \eta_p | p)] + \mathbb{E}[\log P(R(G_p(s, \eta_s)) = \eta_s | G_p(s, \eta_s))]. \quad (8)$$

Real images and their labels enable the classifier to learn the decision boundary for each class, and the synthesized images can force the classifier to treat the fake images as the real ones and provide discriminant outputs regardless of their domain gap. For this reason, the classifier needs to be trained jointly with the other parts of our framework.

We adopt the binary cross-entropy loss for discriminators and focal loss [12] for classification. The pixel-wise loss for photo reconstruction is measured by $\mathcal{L}1$-distance.

A.3. Datasets

We train our model on three datasets: Scribble [5] (10 classes), QMUL-Sketch [21, 18, 13] (3 classes), and SketchyCOCO [4] (14 classes of objects). During the training stage, the sketches of certain classes are completely removed to meet the open-domain settings.

Scribble This dataset contains ten classes of objects, including white-background photos and simple outline sketches. Six out of ten object classes have similar round outlines, which imposes more stringent requirements on the network: whether it can generate the correct structure and texture conditioned on the input class label. In our open-domain setting, we only have the sketches of four classes for training: *pineapple* (151 images), *cookie* (147 images), *orange* (146 images), and *watermelon* (146 images). We set the input image size to 256×256 and train all the compared networks for 200 epochs. We apply the Adam [9] optimizer with batch size$= 1$, and the learning rate is set to $2e-4$ for the first 100 epochs, and it decreases linearly to zero in the second 100 epochs.

QMUL-Sketch We construct it by combing three datasets: handbags [18] with 400 photos and sketches, ShoeV2 [21] with 2000 photos and 6648 sketches, and ChairV2 [13] with 400 photos and 1297 sketches. For the open-domain training setting, we completely remove the sketches of the ChairV2. We train the networks for 400 epochs.

SketchyCOCO This dataset includes 14 object classes, where the sketches are collected from the Sketchy dataset [16], TU-Berlin dataset [3], and $Quick!Draw$ dataset [6]. The 14,081 photos for each object class are segmented from the natural images of COCO Stuff [2] under unconstrained conditions, thereby making it more difficult for existing methods to map the freehand sketches to the photo domain. In our open-domain setting, we remove the sketches of two classes during training: $sheep$ and $giraffe$. We use EdgeGAN weights released by the author. All the other networks are trained for 100 epochs.

A.4. Implementation Details

Our model is implemented in PyTorch [14, 15]. We train our networks with the standard Adam [9] using 1 NVIDIA V100 GPU. The batch size and initial learning rate are set to 1 and $2e-4$ for all datasets. The epoch numbers are 200, 400, and 100 for the Scribble [5], QMUL-Sketch [21, 18, 13], and SketchyCOCO [4], respectively. The learning rates drop by multiplying 0.5 in the second half of epochs. For the compared method EdgeGAN [4], we use the official implementation in https://github.com/sysu-imsl/EdgeGAN for data preprocessing and training. It is trained for 100 epochs on Scribble and QMUL datasets using one NVIDIA GTX 2080 GPU. The batch size is set to 1 due to memory limitation.

B. Experimental Results

We first show more sketch-to-photo results on the QMUL-Sketch dataset in Section B.1 and briefly discuss these results. At last, we show more sketch-to-photo synthesis results of our method in Section B.2.

B.1. Comparison on QMUL-Sketch Dataset

We compare our method with the same baseline methods as described in the main paper: (a) CycleGAN as the baseline, (b) conditional CycleGAN that takes sketch and class label as input, and (c) EdgeGAN [4] trained on this dataset. Different from the Scribble dataset, the sketches in QMUL-Sketch are from three different datasets with rich strokes. Thus, the sketch itself already contains sufficient class information [13]. As shown in Figure 2, most compared methods can generate high-quality photos. Still, all of these methods change the structure of the open-domain class ($Chair$), as shown in the bottom two rows of columns (a), (b), and (c) of Figure 2. Compared with them, our model can maintain the natural shape in the original sketch and generate realistic photos.

The quantitative results are shown in Table 1. We can see that our model is preferred by more users than the other compared methods. While in terms of the FID score and classification accuracy, ours is the second-best. This is because the sketches in the QMUL-Sketch dataset are three times more than the photos (especially for *shoes*), which is not consistent with our motivation of enriching the missing sketches with abundant photos. Under this scenario, the asymmetry within the framework and strategies' design does not bring too many benefits.

Metric	Method	full	in-domain	open-domain
FID ↓	CycleGAN	97.9	87.7	151.7
	conditional CycleGAN	**91.6**	88.2	**107.5**
	EdgeGAN	243.0	281.3	268.3
	Ours	92.4	**76.9**	142.6
Acc (%) ↑	CycleGAN	72.6	64.7	78.2
	conditioanl CycleGAN	78.4	58.0	**92.6**
	EdgeGAN	62.7	**100.0**	36.8
	Ours	**89.7**	91.5	88.5
Human (%) ↑	CycleGAN	4.00	4.57	2.67
	conditional CycleGAN	21.20	18.86	26.67
	EdgeGAN	0.00	0.00	0.00
	Ours	**74.80**	**76.57**	**70.67**

Table 1: Results of quantitative evaluation and user preference study on QMUL-Sketch dataset. Best results are shown in **bold**.

Figure 2: Results on the QMUL-Sketch dataset. Compared with methods (a) CycleGAN [22], (b) conditional CycleGAN, and (c) EdgeGAN [4], our model can faithfully maintain the natural shapes in sketch inputs and synthesize realistic photos.

B.2. More Sketch-to-Photo Results

Here we show more 256×256 sketch-to-photo results of our model in Figure 3, 4 and 5. Previous sketch-to-photo synthesis works usually have output sizes $= 64 \times 64$ or 128×128. Leveraging the output size makes the problem even more challenging for two reasons: (1) the difficulty of correcting larger shape deformation, and (2) generating richer details and realistic textures for each image composition. The results in the following pages suggest that AODA is able to synthesize 256×256 photo-realistic images.

In addition, Figure 6 shows the in-domain results obtained on the full dataset of Scribble [5] without removing any sketch. Our network not only handle the open-domain training problem, but also perform even better under a common multi-class sketch-to-photo generation setting.

References

[1] Andrew Aitken, Christian Ledig, Lucas Theis, Jose Caballero, Zehan Wang, and Wenzhe Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017. 2

[2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1209–1218, 2018. 3

[3] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012. 3

[4] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: Image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5174–5183, 2020. 2, 3

[5] Arnab Ghosh, Richard Zhang, Puneet K Dokania, Oliver Wang, Alexei A Efros, Philip HS Torr, and Eli Shechtman. Interactive sketch & fill: Multiclass sketch-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1171–1180, 2019. 2, 3, 4

[6] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017. 3

[7] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 1

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 694–711, 2016. 1

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2, 3

[10] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017. 2

[11] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 702–716, 2016. 2

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 2

[13] Runtao Liu, Qian Yu, and Stella Yu. Unsupervised sketch-to-photo synthesis. *arXiv preprint arXiv:1909.08313*, 2019. 2, 3

[14] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019. 3

[16] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 3

[17] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 2

[18] Jifei Song, Qian Yu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Deep spatial-semantic attention for fine-grained sketch-based image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5551–5560, 2017. 2, 3

[19] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 1

[20] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 2020. 2

[21] Qian Yu, Feng Liu, Yi-Zhe SonG, Tao Xiang, Timothy Hospedales, and Chen Change Loy. Sketch me that shoe. In *Computer Vision and Pattern Recognition*, 2016. 2, 3

[22] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017. 1, 3

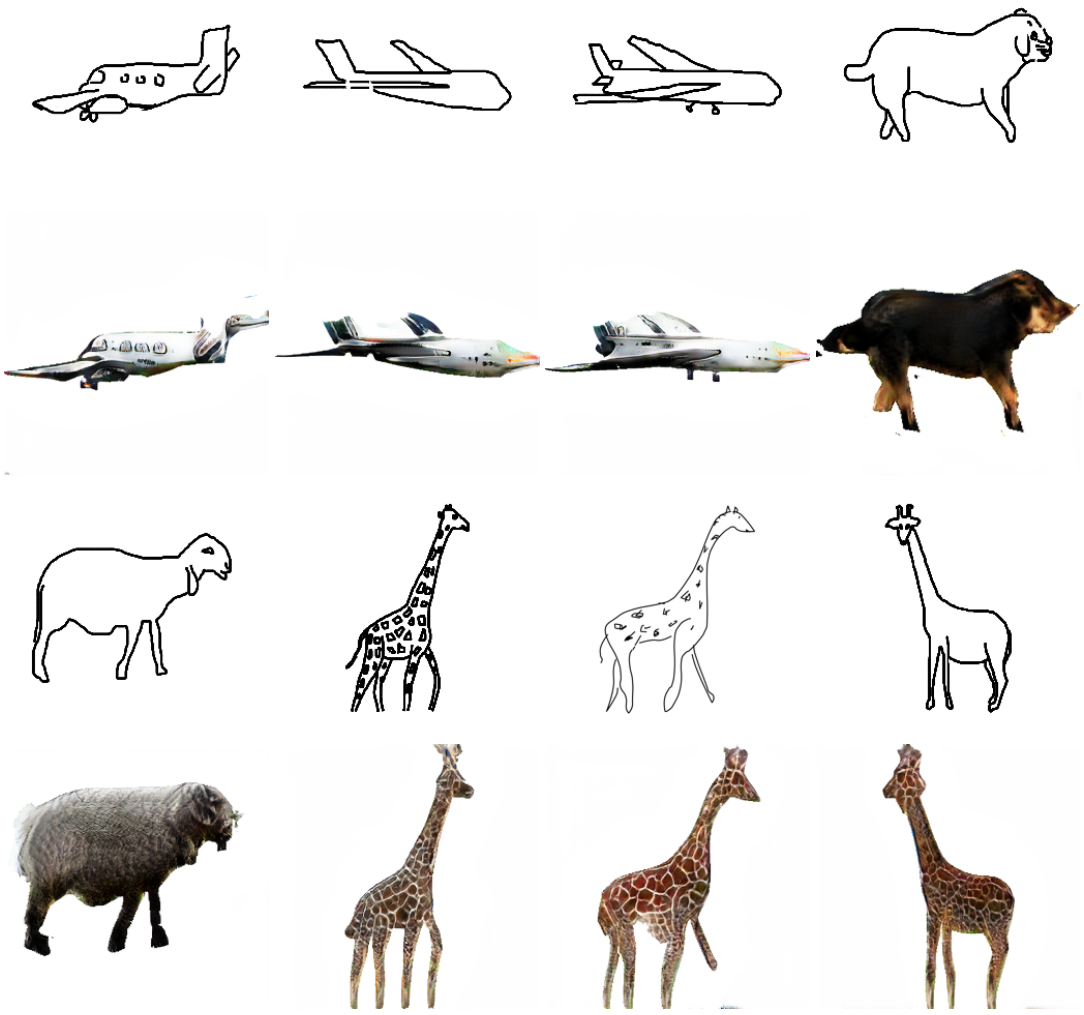

Figure 3: More 256×256 results on the SketchyCOCO dataset.

Figure 4: More 256×256 results on the QMUL-Sketch dataset.

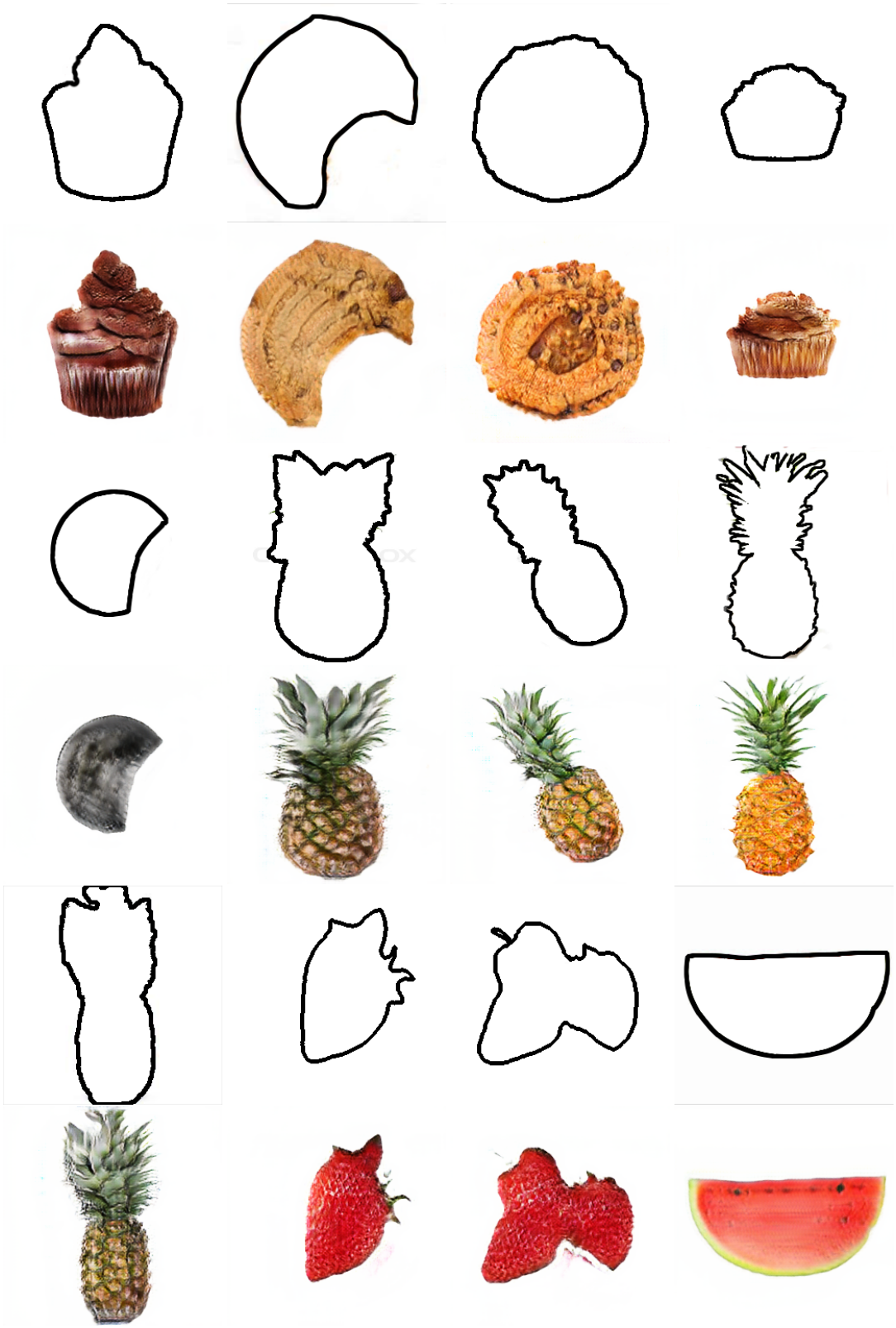

Figure 5: More 256×256 results on the Scribble dataset.

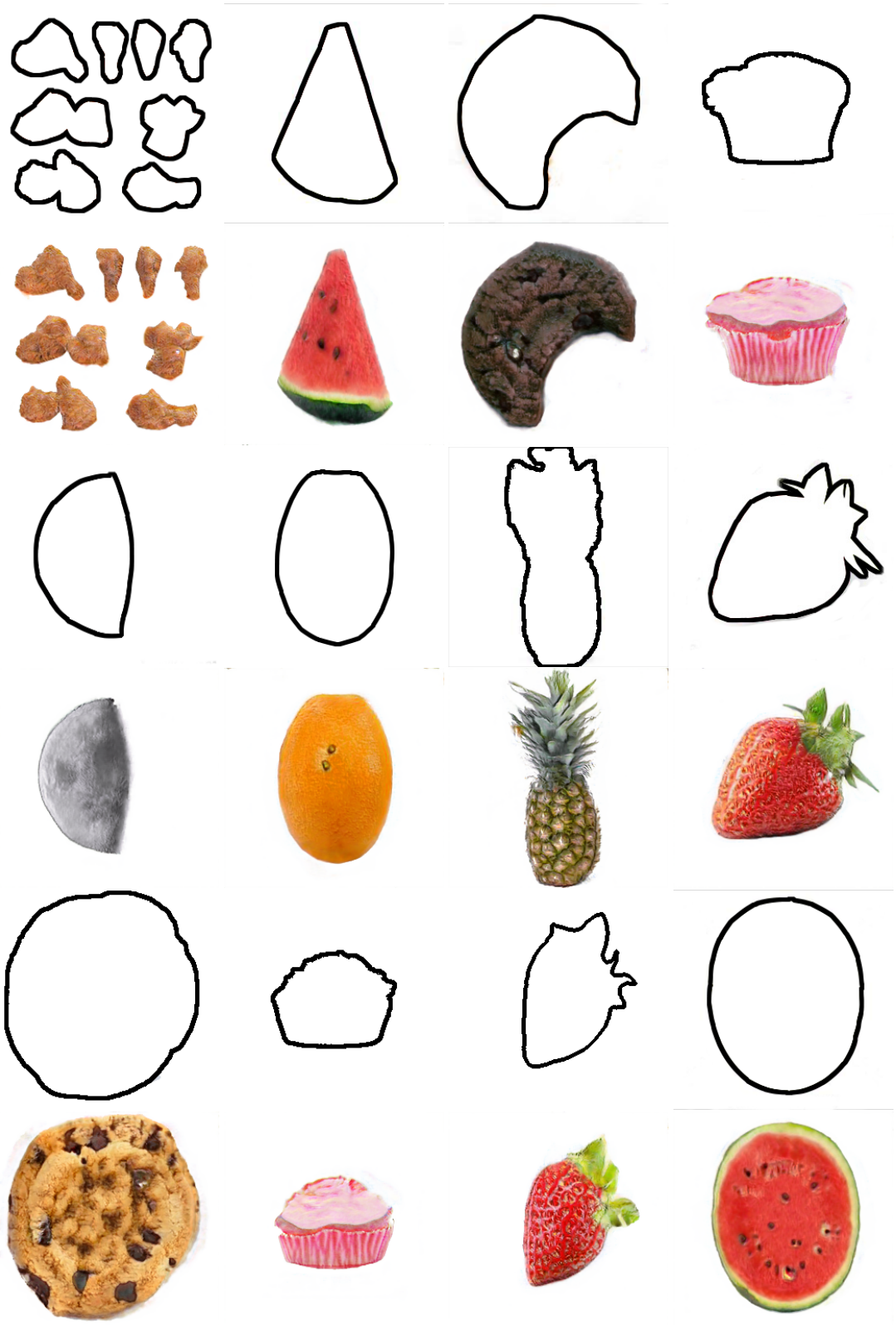

Figure 6: In-domain 256×256 results on the Scribble dataset.