Supplementary material for Photographic Text-to-Image Synthesis with a Hierarchically-nested Adversarial Network

Anonymous CVPR submission

Paper ID 2823

1. Training and Architecture Details

The training procedure is similar to the one used in standard GANs, which alternatively updates the generator and discriminators until converge.

The Adam optimizer [3] is used. The initial learning rate is set as 0.0002 and decreased by half for every 100 epochs (50 for COCO). The model is trained for 500 epochs in total (200 epochs for COCO). We configure the side outputs at 4 different scales where the feature map resolution is equal to 64^2, 128^2, 256^2, and 512^2, respectively. For the local image loss of these 4 side outputs, we set $R_1 = 1, R_2 = 1, R_3 = 5$, and $R_4 = 5$. For example, R_1 refers to 64^2. These numbers are not fine-tuned but are set empirically. We believe there exists better configurations to be explored.

All intermediate conv layers, except from the specified ones in Section 3.4, use 3×3 kernels (with reflection padding). Some other normalization also layers are experimented (i.e. instance normalization [5] and layer normalization [1]) since they are used by recent advances [7, 2]. But the results are not satisfactory.

With respect to the generator, we use 1-repeat residual blocks for the generator till the 256^2 resolution. The input of the generator is a $1024\times4\times4$ tensor. As the feature map resolution increases by 2, the number of feature maps is halved at $8, 32, 128, 256$ sizes. To generate 512^2 images, we pre-train the generator to 256^2 due to the limitation of the GPU memory. We use a 3-repeat res-block followed by a stretching layer to upscale the feature map size to $32\times512\times512$. and a linear compression layer to generate images. Since the 256^2 image already captures the overall semantics and details, to boost the training and encourage the 512^2 maintain this information, we use a l1 reconstruction loss to 'self-regularize' the generator.

All source code will be released.

2. More Qualitative Results and Analysis

In this section, we demonstrate more sample results for the three datasets.

Figure 1 compares our results with StackGAN. For each input, 6 images are randomly sampled. Furthermore, we visualize zoomed-in samples compared with StackGAN in Figure 2. Our results demonstrate obviously better quality, less artifacts, and less sharp pixel transitions.

Figure 3 shows the results on the CUB bird dataset. All the outputs of a model with different resolutions are also shown. As can be observed in this two figures, our method can generate fairly vivid images with different poses, shape, background, etc. Moreover, the images with different resolutions, which are side outputs of a single model, have very consistent information. More and more image details can be observed as the resolution increases. Figure 4 shows the results on the Oxford-102 flower dataset. Very detailed petals can be generated with photographic colors and saturability.

Figure 5 shows some sampled results on the COCO dataset. COCO is much more challenging than the other two datasets since it contains natural images from a wide variety of scenes containing hundreds of different objects. As can be observed in the shown samples, our method can still generate semantically consistent images.

However, it is worth to notice that, although our method significantly improves existing methods [6, 4] on COCO, we realize that generating fine-grained details of complex natural scenes with various objects is still challenging. Based on this study, we expect to further address this problem as the future study.

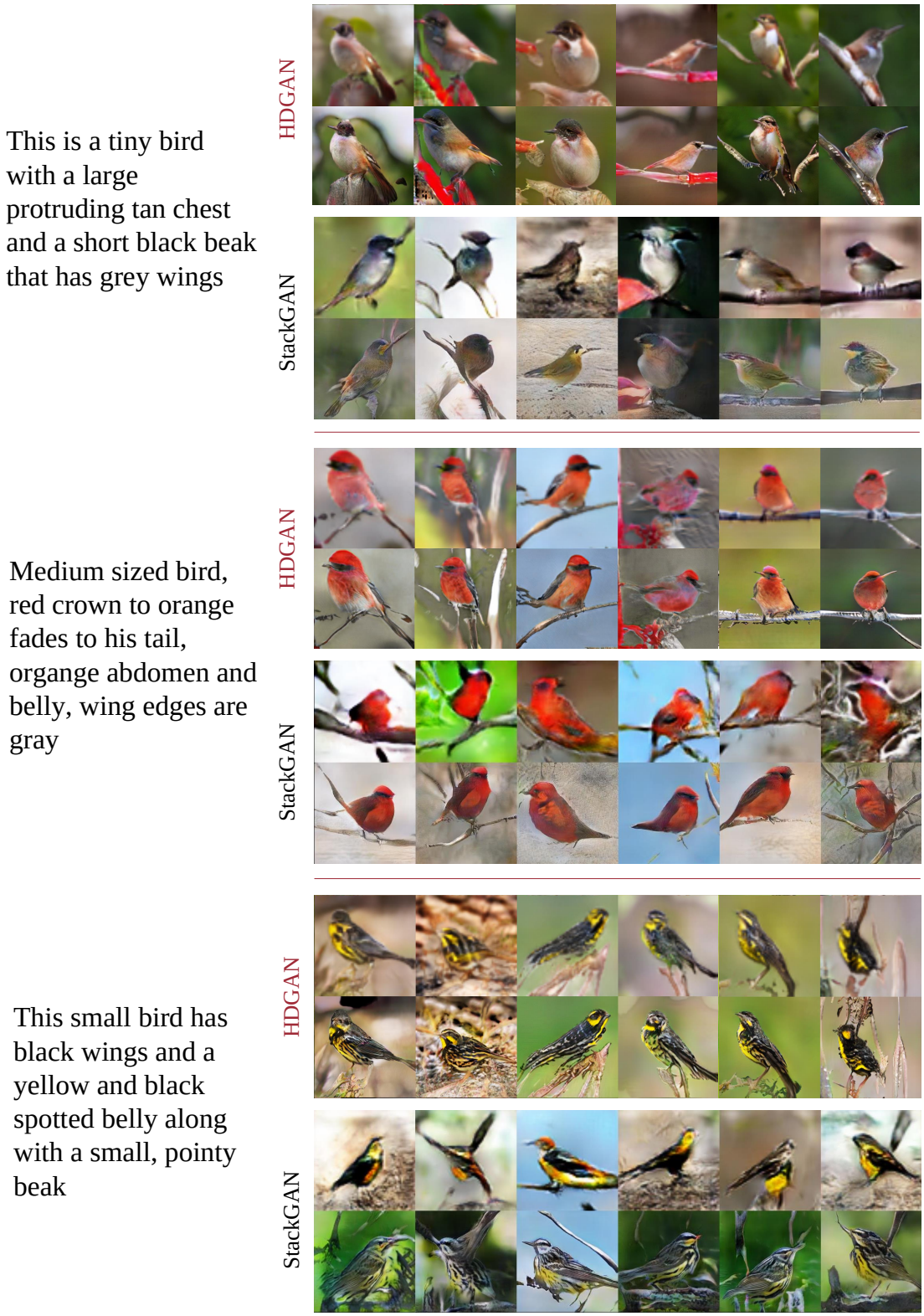

Figure 1. Sample results on CUB compared with StackGAN. For each input, 6 samples are shown with resolutions of 64^2 and 256^2, respectively. As can be obviously seen, our HDGAN can generate very consistent content in images of different resolutions. Moreover, our generated images show more photographic color and contrast.

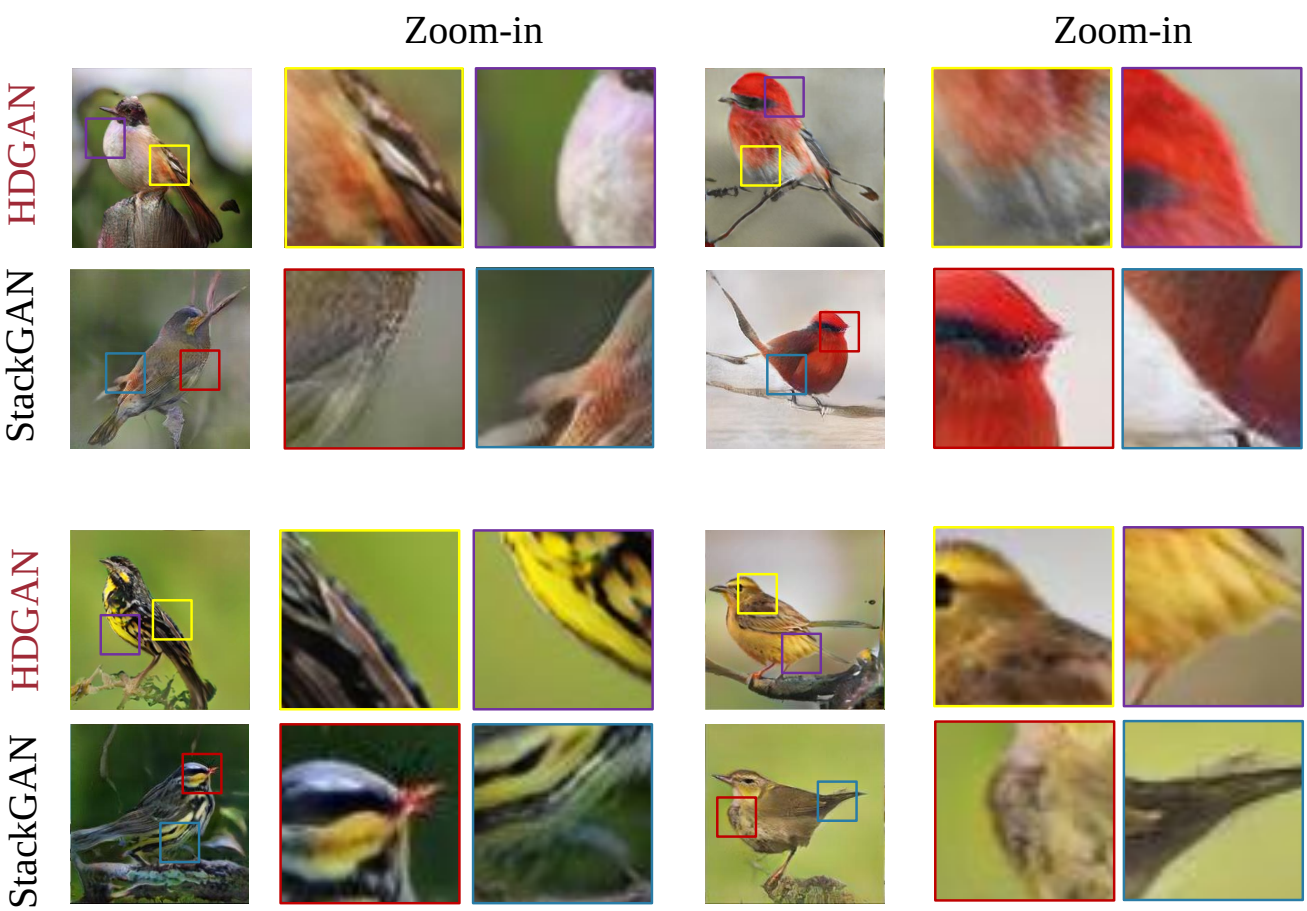

Figure 2. Zoomed-in samples compared with StackGAN. The best sample among 6 samples given an input text is selected. It can be clearly observed that our results show more smooth visual results. Especially, much less sharp pixel transitions exist in our results.

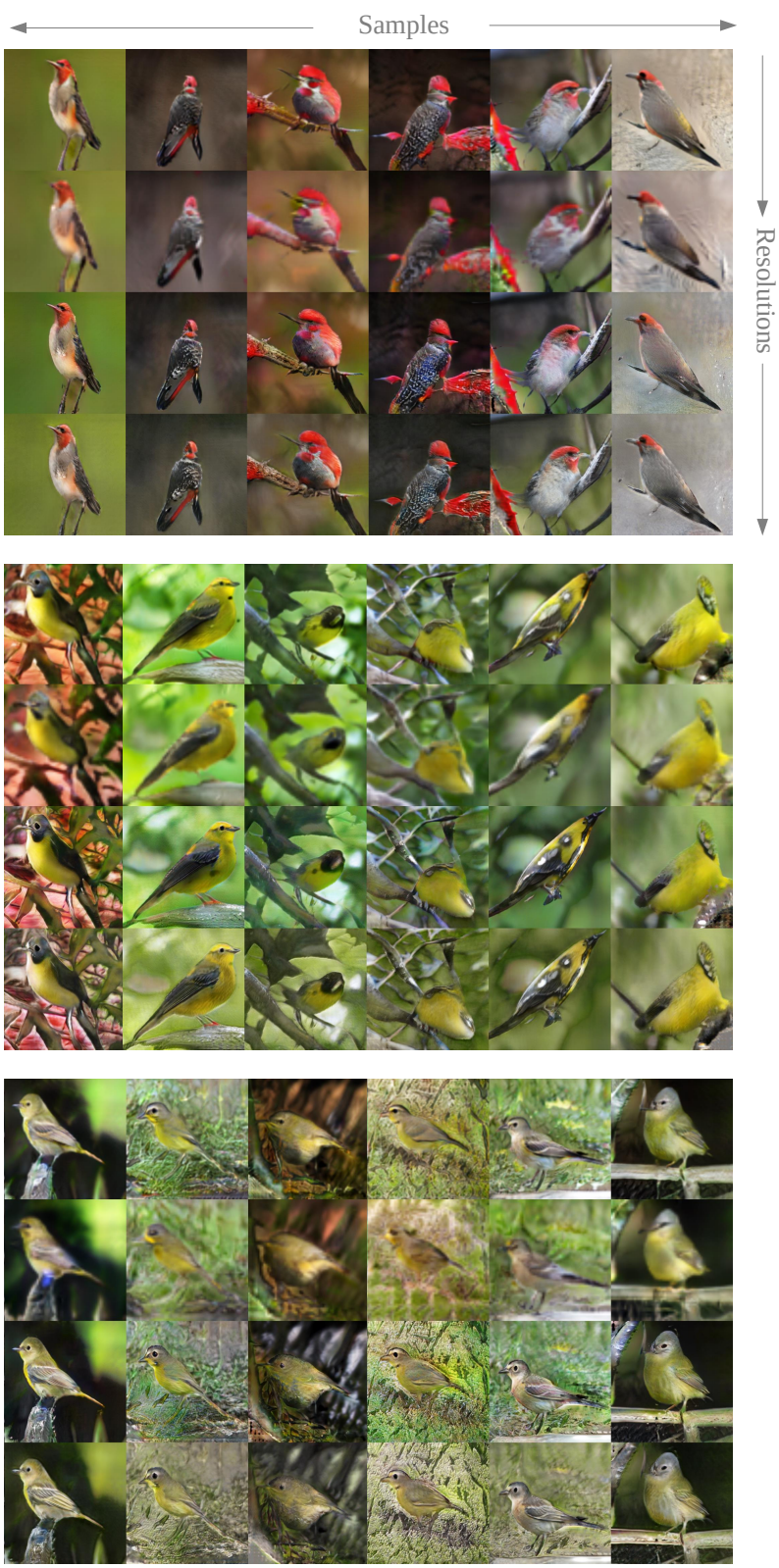

Figure 3. Sample results on CUB. For each input, 6 samples with resolutions of 64^2, 128^2, 256^2, and 512^2 are shown in 4 rows, respectively.

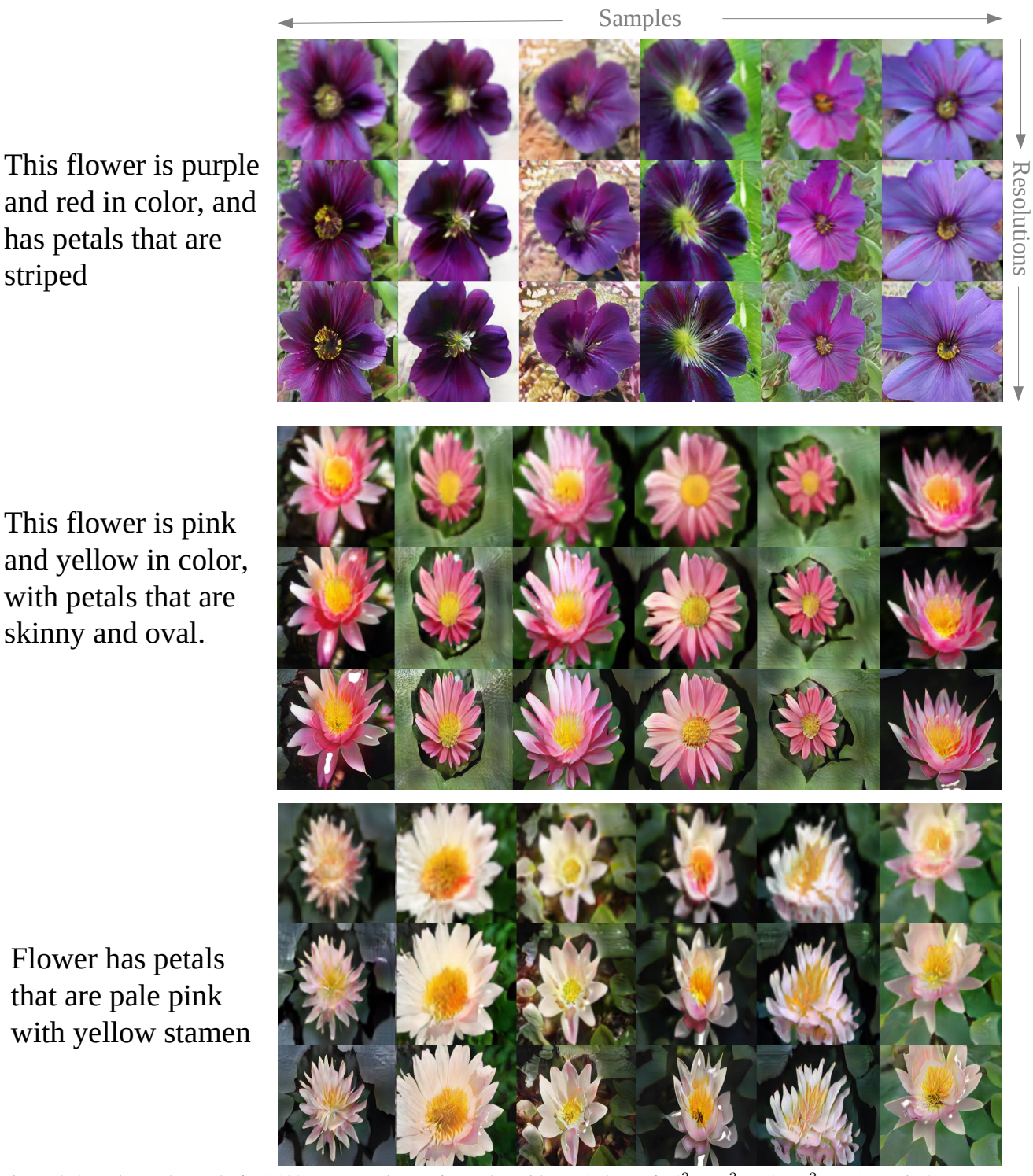

Figure 4. Sample results on Oxford-102. For each input, 6 samples with resolutions of 64^2, 128^2, and 256^2 are shown in 3 rows, respectively.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
[2] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *ICCV*, 2017.

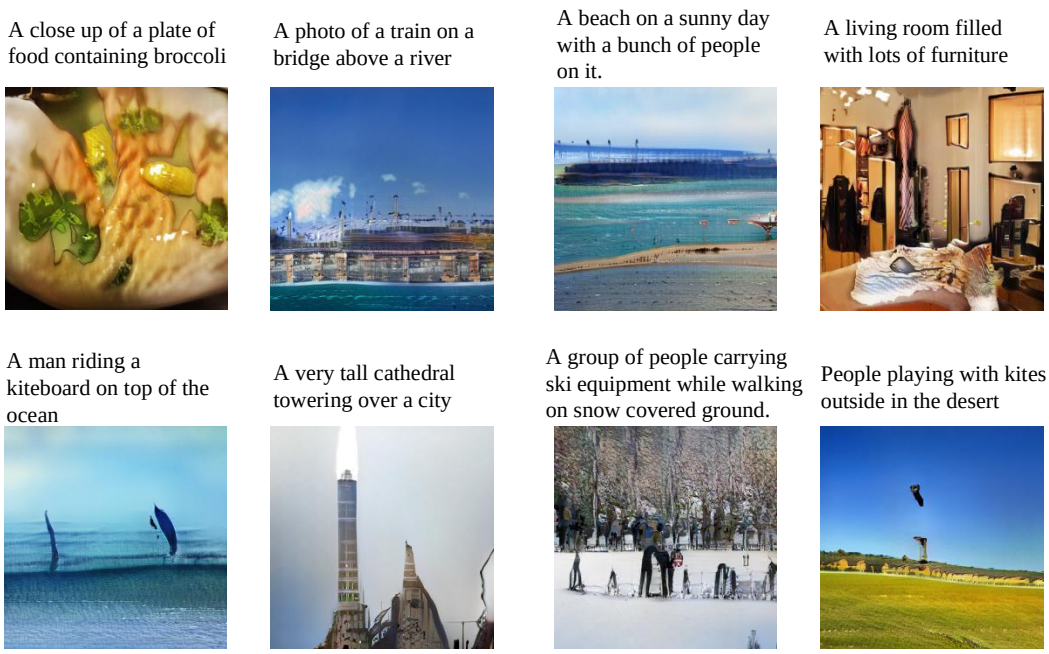

Figure 5. Sample results on COCO. We show 8 256^2 samples in very different scenes.

[3] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *ICML*, 2016.

[5] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[6] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017.