Supplementary Materials

Anonymous CVPR submission

Paper ID 4318

1. Objective Function

Let $\tilde{s}$ be the output edge-image pair and s be the real edge-image pair, z be the noise vector, and $\hat{s}$ be the random sample. Based on our preliminary results, we leverage WGAN-GP as the basis in our network model to achieve stable and effective training. The loss function of WGAN-GP is defined as follows:

$$\mathcal{L}_{D_J}(D) = \mathop{\mathbb{E}}_{\tilde{s}\sim\mathbb{P}_g}[D_J(\tilde{s})] - \mathop{\mathbb{E}}_{s\sim\mathbb{P}_r}[D_J(s)] + \lambda \mathop{\mathbb{E}}_{\hat{s}\sim\mathbb{P}_{\hat{s}}}[(||\nabla_{\hat{s}} D_J(\hat{s})||_2 - 1)^2]. \tag{1}$$

$$\mathcal{L}_{D_J}(G) = \mathop{\mathbb{E}}_{\tilde{s}\sim\mathbb{P}_g}[-D_J(\tilde{s})]. \tag{2}$$

Let $\tilde{x}$, x and $\hat{x}$ be the generated edge, real edge and random generated edge, and $\tilde{y}$, y and $\hat{y}$ be the generated natural image, real image and random generated natural image, respectively. Since the discriminators D_E and D_I adopt the same architecture as D_J, we can define their losses as:

$$\mathcal{L}_{D_E}(D) = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[D_E(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[D_E(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D_E(\hat{x})||_2 - 1)^2]. \tag{3}$$

$$\mathcal{L}_{D_E}(G) = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[-D_E(\tilde{x})]. \tag{4}$$

$$\mathcal{L}_{D_I} = \mathop{\mathbb{E}}_{\tilde{y}\sim\mathbb{P}_g}[D_I(\tilde{y})] - \mathop{\mathbb{E}}_{y\sim\mathbb{P}_r}[D_I(y)] + \lambda \mathop{\mathbb{E}}_{\hat{y}\sim\mathbb{P}_{\hat{y}}}[(||\nabla_{\hat{y}} D_I(\hat{y})||_2 - 1)^2]. \tag{5}$$

$$\mathcal{L}_{D_I}(G) = \mathop{\mathbb{E}}_{\tilde{y}\sim\mathbb{P}_g}[-D_I(\tilde{y})]. \tag{6}$$

During training, D_J, D_E and D_I are updated to minimize Equation 1, Equation 3 and Equation 5 separately. Our classifier is used to predict class labels. We use the focal loss [5] as the classification loss. Let c be the predicted class label. Formally, the loss function between the ground-truth label and the predicted label is defined as:

$$\mathcal{L}_{ac}(D) = \mathbb{E}[logP(C = c|y)]. \tag{7}$$

Classifier is trained to maximize Equation 7. The generator also maximizes $\mathcal{L}_{ac}(G) = \mathcal{L}_{ac}(D)$ when the classifier is fixed.

We train the encoder E with the $L1$ loss between the random input vector z and generated vector $\tilde{z}$. $\tilde{z}$ is from encoding the edge $\tilde{x}$ which is the output of the generator G_E. Formally, the loss function is defined as:

$$\mathcal{L}_1^{\text{latent}}(E) = \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}||z - E(\tilde{x})||_1. \tag{8}$$

In summary, the loss function of the generator G_E is:

$$\mathcal{L}_{G_E}(G) = \mathcal{L}_{D_J}(G) + \mathcal{L}_{D_E}(G). \tag{9}$$

and the loss function of the generator G_I is:

$$\mathcal{L}_{G_I}(G) = \mathcal{L}_{D_J}(G) + \mathcal{L}_{D_I}(G) - \mathcal{L}_{ac}(G). \tag{10}$$

The generator G_E minimizes Equation 9 and G_I minimizes Equation 10.

2. Implementation Details

In the stage of instance generation, we train the model with 100 epochs, and randomly generate latent vectors in the normal distribution with zero mean and variance of 1.0. We train the generator and the discriminators with instance normalization. The encoder is implemented with ResNet blocks using instance normalization and ReLU, and the classifier is implemented with MRU block [2]. We use ReLU and Tanh for the generator while Leaky ReLU for the discriminator. In the instance of DCGAN, we use the Adam optimizer with a learning rate of 0.0002 and a beta of 0.5 for all the networks. In the instance of WGAN [1], we clamp the weights between -0.01 and 0.01 after each gradient update and use the RMSprop optimizer with a learning rate of 0.0002 for all the networks. In the instance of WGAN-GP [3], we set the weight of gradient penalty λ to 10, using the RMSprop optimizer with a learning rate of 0.0002 for all the networks. For background generation, we train the pix2pix model with the 110 epochs. We use XDoG [7] to obtain the edge maps of objects for training data.

3. Representative Samples from Sketchy-COCO

We show more examples of SketchyCOCO including five-tuple ground truth data in Fig. 1.

4. Object Level Results

4.1. More object level comparison results

We compare edgeGAN with ContextualGAN [6], SketchyGAN [2], and pix2pix [4] under different training strategies. Fig. 3 shows the comparison results. This figure is a supplement to Fig. 6 in the paper.

4.2. Some 128×128 results in the object level

We have trained the model on images of resolution 128×128. What is different from training on images of

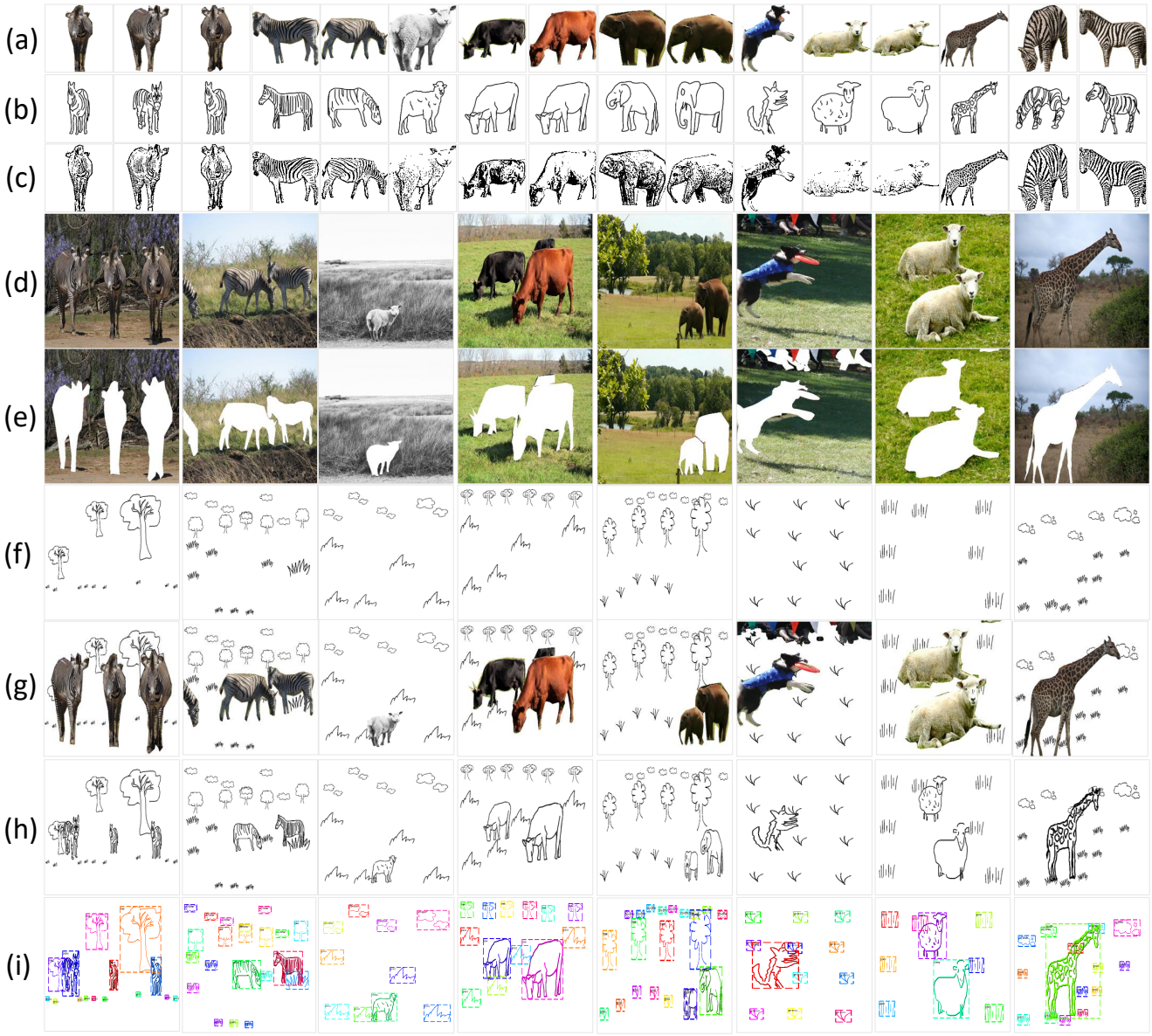


Figure 1: More exmples of the five-tuple ground truth data of SketchyCOCO, i.e., (1) {foreground image(a), foreground sketch(b), foreground edge maps(c)}, (2) {background image(e), background sketch(f)}, (3) {scene image(d), foreground image & background sketch(g)}, (4) {scene image(d), scene sketch(h)}, and (5) sketch segmentation(i). Unlike foreground sketches depicting single objects, background sketches such as grass and trees are purposefully designed to depict a specific region (e.g., several tree sketches depict a forest).

resolution 64×64 is that we use one more discriminator, whose structure is a copy of D_I. And we set the size of its input to 64×64 to guarantee the global information. In addition, we also set the size of D_I's input to 256×256 so that the model can pay more attention to local details. Some results are shown in Fig. 4.

5. Scene Level Results

In this section, we show more 128×128 scene-level results in Fig. 5, which are generated based on the 64×64 object level results, as well as more 256×256 results in Fig. 6, which are generated based on the 128×128 object level results.

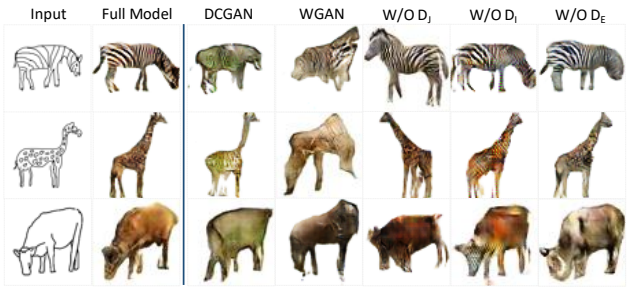


Figure 2: The results of network ablation. The full model is based on WGAN-gp and contains D_J, D_I and D_E. WGAN and DCGAN are the structures replacing WGAN-gp with WGAN and DCGAN, respectively.

Table 1: Object level scores in the ablation study.

Model(object)	FID	Accuracy	Shape Similarity
Full Model	**87.59**	**0.8866**	**2.294e+04**
W/O D_J	95.63	0.8361	2.457e+04
W/O D_I	110.17	0.6964	2.331e+04
W/O D_E	91.12	0.8204	2.341e+04
DCGAN	108.86	0.6429	2.335e+04
WGAN	106.67	0.3172	2.471e+04

6. Ablation Study

- **How the joint discriminator D_J works?** We concat the outputs of the edge generator and the image generator in the width channel as a joint image, which is used as the fake input of D_J. The real edge-image pair is taken as the real input. Therefore, the generated edge and image from the same vector respect each other under the constraint of the adversarial loss. In the inference stage, the attribute vector, which can be mapped to an edge image close to the input sketch, also can be mapped to a natural image with reasonable pose and shape. As shown in Fig. 2, the pose and shape of the generated image are not correct without D_J.

- **Which GAN model suits our approach the best?** WGAN-gp was proved to be more suitable for small data sets than DCGAN and WGAN, making training more stable and producing higher quality results. As shown in Fig. 2, when we change it to DCGAN or WGAN, the results get worse in both faithfulness and realism. So our network is based on WGAN-gp. More quantitative results are shown in Table 1.

- **Whether multi-scale discriminators can be used to improve the results?** We use multi-scale discriminators to improve the quality of generated images. For resolution 64×64, we add an edge discriminator (D_E) and an image discriminator (D_I), the inputs of which are enlarged edge (128×128) and image (128×128) respectively. As a result, the model can learn smaller receptive fields and thus pay more attention to local details. As shown in Fig. 2, the quality of local details is not as good as that of the full model when without D_I or D_E. As shown in Table 1, the full model outperforms the model without the multi-scale discriminators on both realism and faithfulness metrics.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv:1701.07875*, 2017. 1

[2] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *CVPR*, pages 9416–9425, 2018. 1

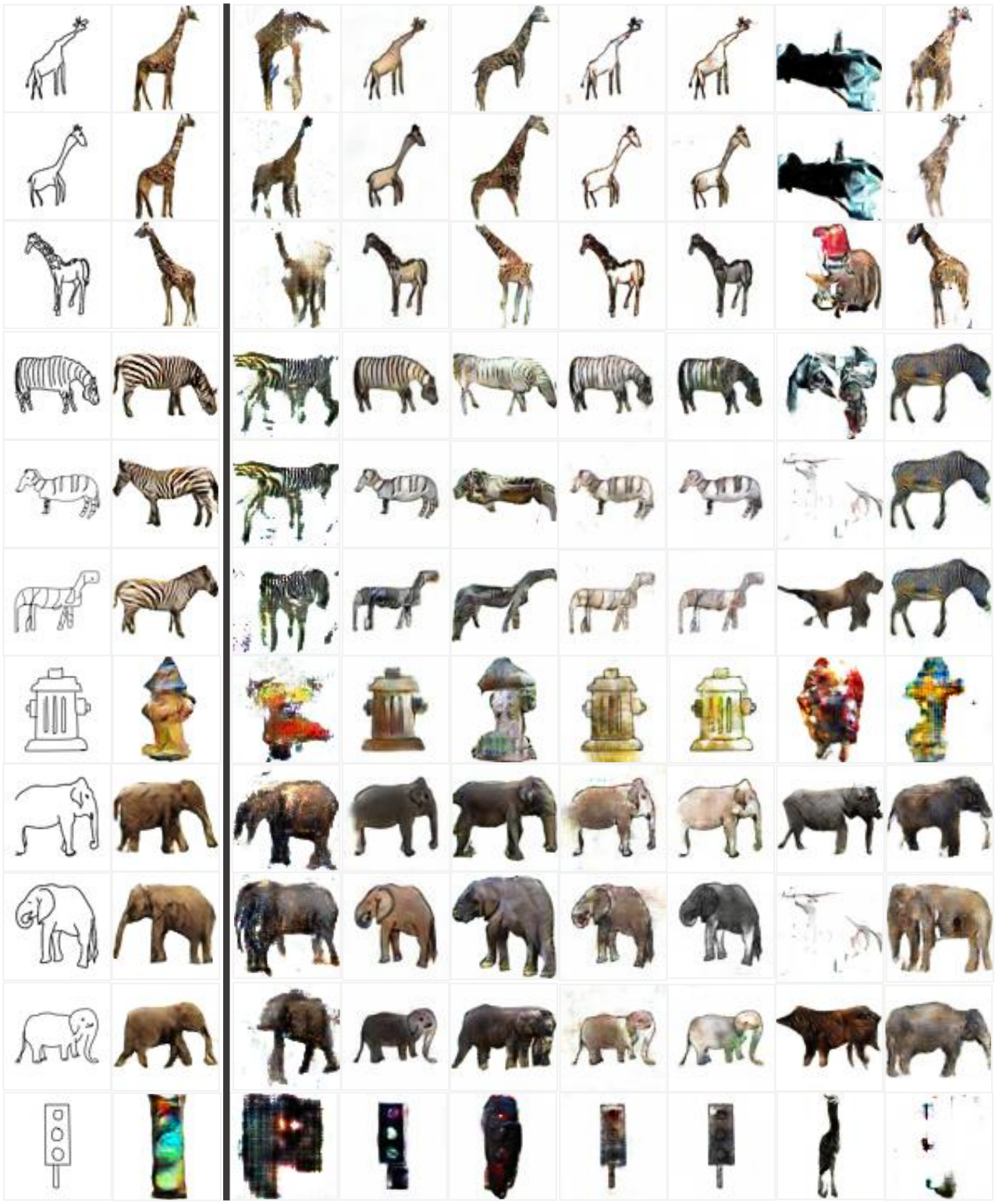
[3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, pages 5767–5777, 2017. 1

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1

[5] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *TPAMI*, 2018. 1

[6] Yongyi Lu, Shangzhe Wu, Yu-Wing Tai, and Chi-Keung Tang. Image generation from sketch constraint using contextual gan. In *ECCV*, pages 205–220, 2018. 1

[7] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012. 1


Input
Ours
(a)
(b1)
(b2)
(c1)
(c2)
(c3)
(c4)

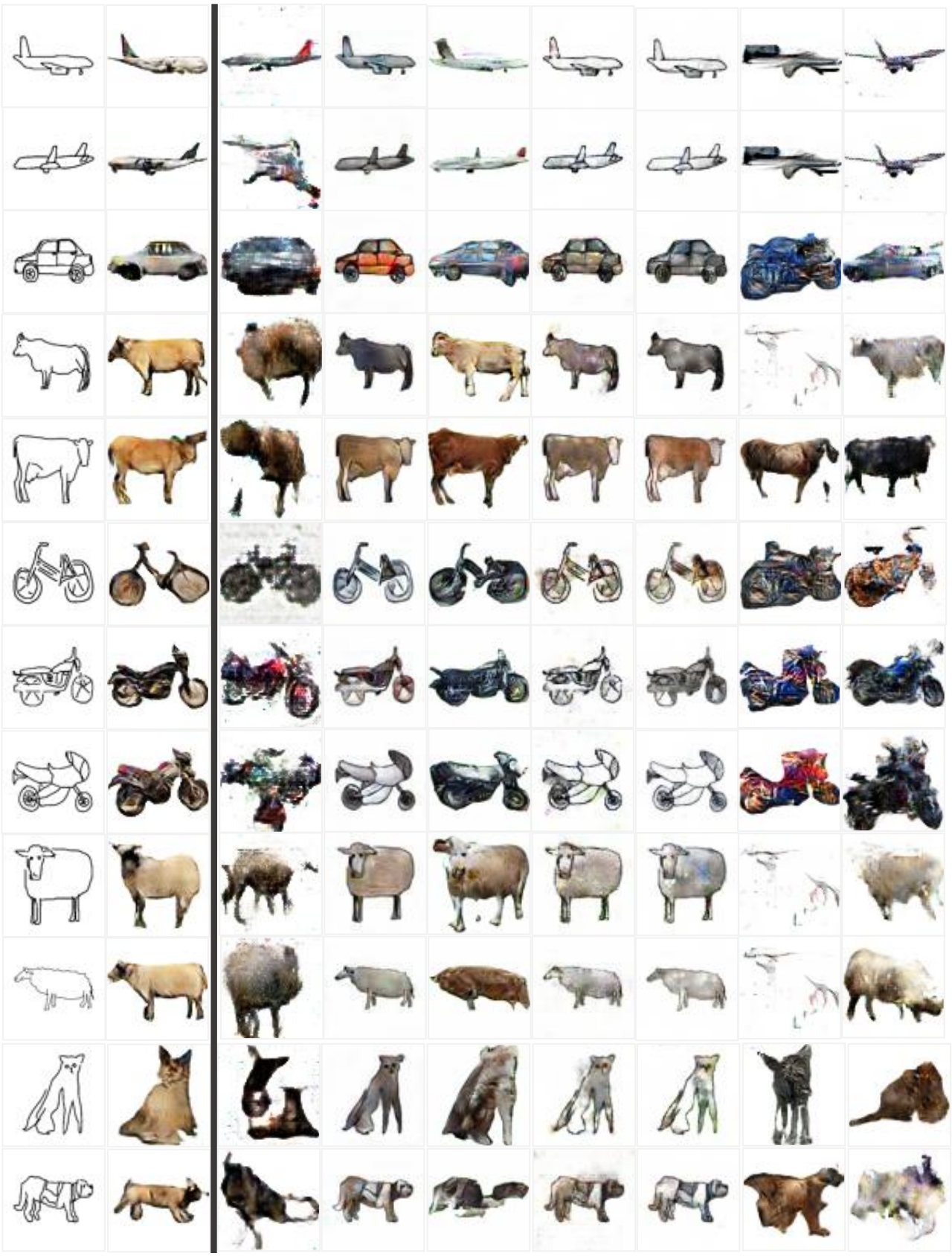

Input
Ours
(a)
(b1)
(b2)
(c1)
(c2)
(c3)
(c4)

Figure 3: From left to right: input sketches, results from edgeGAN, ContextualGAN (a), two training modes of SketchyGAN (i.e., SketchyGAN-E (b1) and SketchyGAN-E&S) (b2), and four training modes of pix2pix (i.e., pix2pix-E-SEP (c1), pix2pix-E-MIX (c2), pix2pix-S-MIX (c3), and pix2pix-S-SEP (c4)).

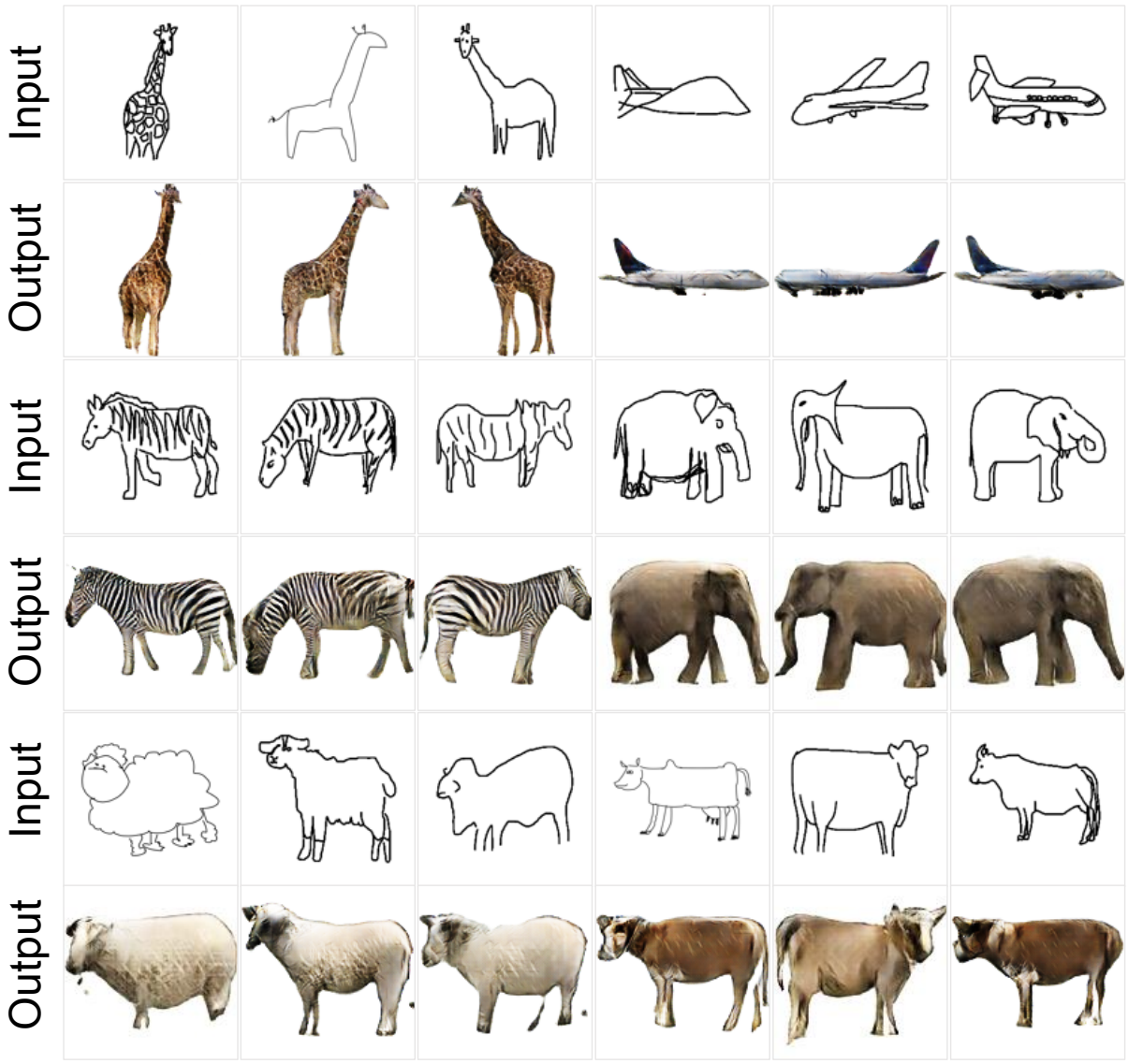


Figure 4: More 128×128 results in the object level.

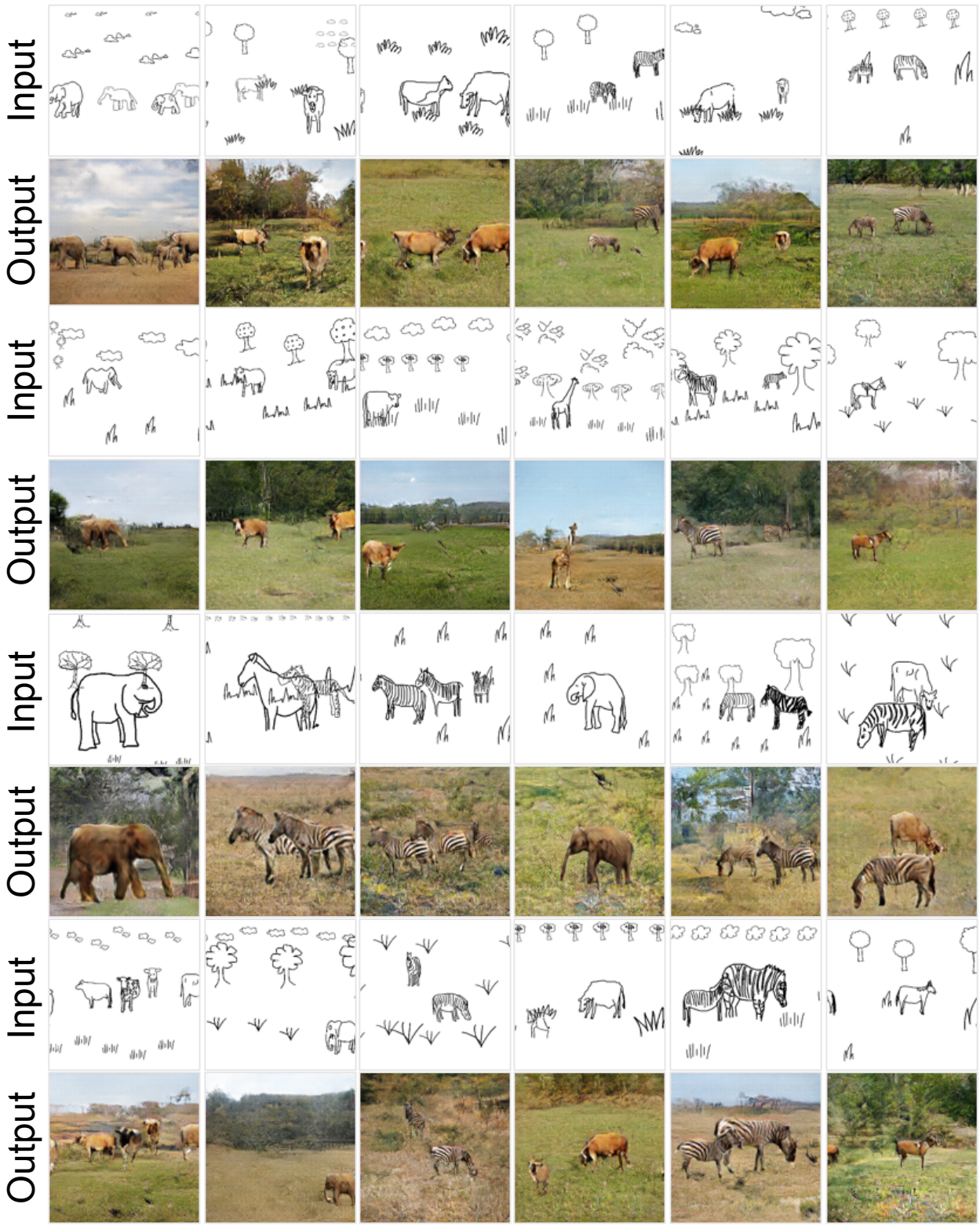
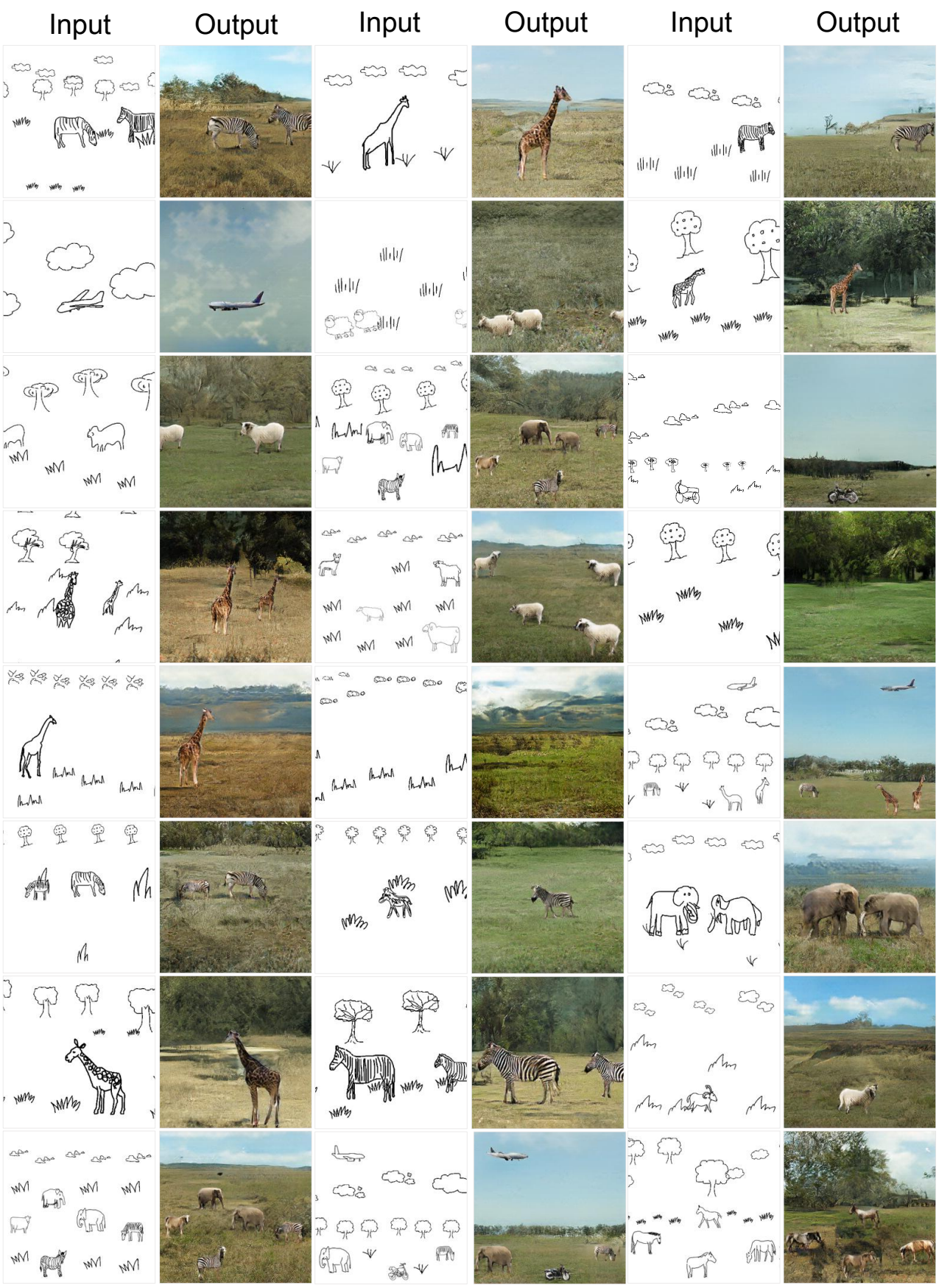


Figure 5: More 128×128 results in the scene level.


Input
Output
Input
Output
Input
Output

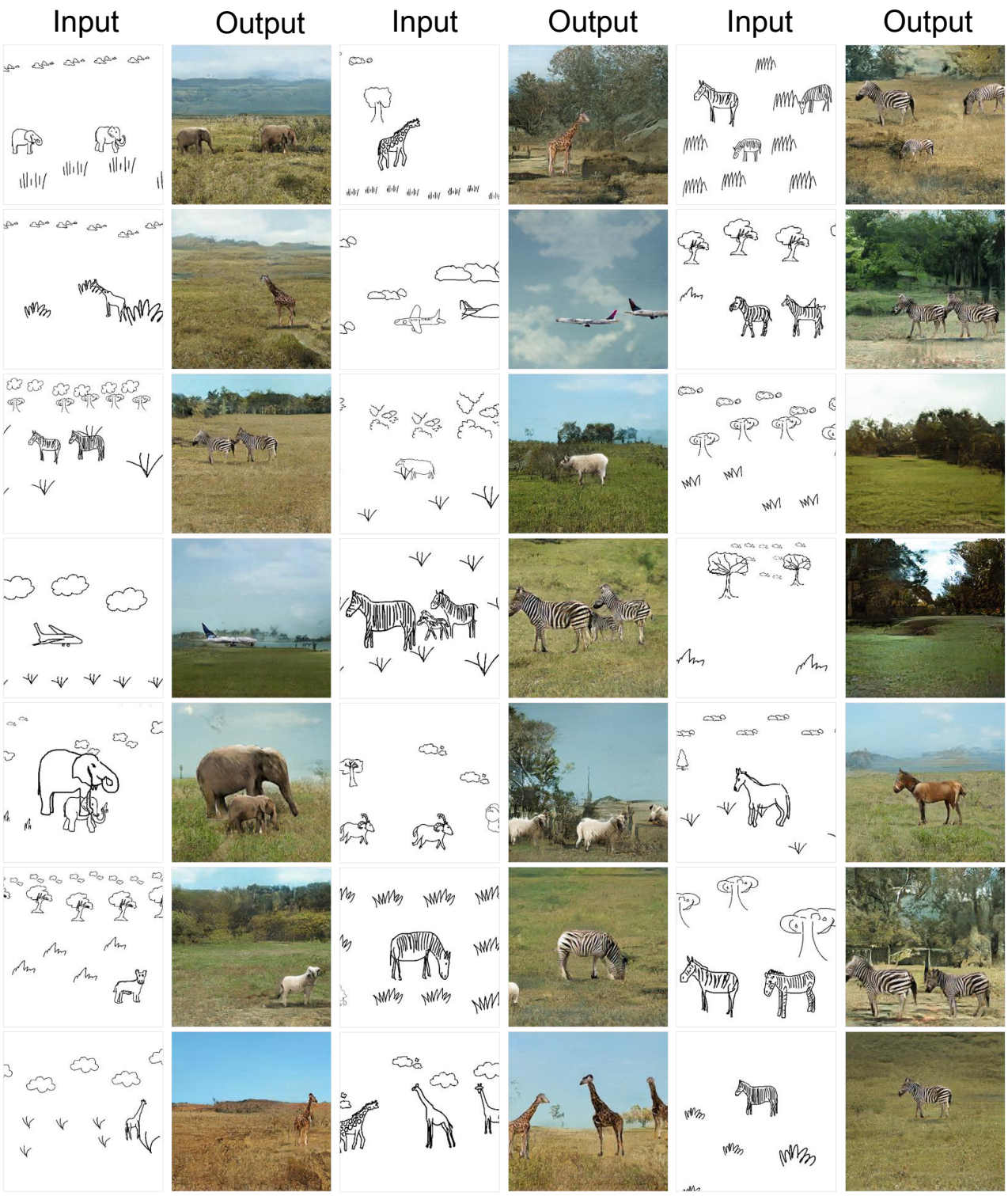


Figure 6: More 256×256 results in the scene level.