Supplementary Material for Image Synthesis from Layout with Locality-Aware Mask Adaption

Zejian Li[1] , Jingyu Wu[1] , Immanuel Koh[2] , Yongchuan Tang[1] , Lingyun Sun[1,3]
[1]Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Zhejiang University
[2]Singapore University of Technology and Design
[3]Collaborative Innovation Center of AI by MOE and Zhejiang Provincial Government
{zejianlee, wujingyu}@zju.edu.cn, immanuel_koh@sutd.edu.sg, {yctang, sunly}@zju.edu.cn

Abstract

This supplermentary material introduces more details of our layout-to-image model (Secition S1), discusses several topics of our model (Seciton S2) and presents more experimental results (Section S3).

S1. Method

In this section, we provide more detailed description of our proposed method, including mask generation (Section S1.1), mask-to-image generation (Section S1.2) and the training loss (Section S1.3).

S1.1. Mask Generation

In the layout-to-image pipeline, a semantic mask is generated first. To form a raw segmentation mask, the masks of objects are generated separately and mapped to bounding boxes. We assume that object masks are affected by the categories and the sizes with stochastic variations. An object mask indeed represents the object shape in the bounding box, and a reasonable shape may be related to the width and height. Given a slim and small bounding box with the class "person", its shape tends to be a whole human body. But given a nearly squared and big bounding box, the "person" may be in a close shot and the shape tends to be an upper half body. This insight drives us to consider box size in object mask generation.

Formally, for the i^{th} object, $\boldsymbol{w_i}$ denotes the width of the bounding box $\boldsymbol{b_i}$ and $\boldsymbol{h_i}$ denotes the height. The category $\boldsymbol{c_i}$ is embedded as a latent vector $\boldsymbol{y_i} \in \mathbb{R}^{d_y}$, and $\boldsymbol{z_i} \in \mathbb{R}^{d_z}$ represents stochastic variation. The generator of object masks $\boldsymbol{F_{om}}: \mathbb{R}^d \mapsto (0,1)^{32\times 32}$ takes the concatenation $\boldsymbol{o_i} = [y_i, z_i, w_i, h_i]$ (object feature) as input and gives an object mask of size $\mathbf{32}\times\mathbf{32}$. We define $\boldsymbol{d} = d_y + d_z + 2$, $d_y = d_z = 128$. Our implementation is built on top of LostGAN-V1 [29].

The architecture of $\boldsymbol{F_{om}}$ mainly consists of several ResNet [8] blocks and a sigmoid operator to transform the input vector to an object mask. Given a set of object features $\boldsymbol{O} = [o_1, \ldots, o_m]$, the generator $\boldsymbol{F_{om}}$ gives the mask of $\boldsymbol{m}$ objects, which are resized and mapped to the corresponding bounding boxes to form the raw mask $\tilde{\boldsymbol{M}}$ as shown in Figure 3 in the main paper.

The raw mask is adapted with our proposed Locality-Aware Mask Adaption module. We summarize the PyTorch [25] pseudo code of our LAMA in Figure S1. To reduce the computational cost, we resize the raw mask to 64×64 in the adaption, and the scaling factors are resized back to $H \times W$ and applied to the raw mask.

S1.2. Mask-to-Image Generation

In this stage, an image is synthesized according to the generated mask. Formally, given the mask $\boldsymbol{M}$, the object features $\boldsymbol{O}$ and a image variation $\boldsymbol{z_{img}}$, the mask-to-image generation module gives an image accordingly. The image variation $\boldsymbol{z_{img}} \in \mathbb{R}^{d_z}$ is to capture image style.

Our ResNet block. The mask-to-image generator stacks serval ResNet [8] blocks, whose architecture is shown in Figure S2. The block has a pre-activation [9] architecture, with Batch-Group Normalization to injection mask and category information. $\boldsymbol{O}^\mathsf{T} \otimes \boldsymbol{M}$ means the object features are aggregated in pixels.

Batch-Group Normalization. We use Batch-Group Normalization (BGN) to utilize information aross channels when the batch size is small. Given a batch-normalized [13] feature $\hat{\boldsymbol{x}}^{(BN)}$ and a group-normalized [34] one $\hat{\boldsymbol{x}}^{(GN)}$, BGN gives the feature $\hat{\boldsymbol{x}}^{(BGN)}$ as

$$\hat{\boldsymbol{x}}^{(BGN)} = \left(\boldsymbol{\rho}\cdot\hat{\boldsymbol{x}}^{(BN)} + (1-\boldsymbol{\rho})\cdot\hat{\boldsymbol{x}}^{(GN)}\right)\cdot\boldsymbol{\gamma} + \boldsymbol{\beta} \quad (1)$$

Here $\boldsymbol{\rho} \in [0,1]$ controls the weight of two normalizations, initialized as 0.1. The modulation parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$

```python
import torch
from torch.nn import functional as F
def LAMA(self, raw_mask, obj_feat):
   # raw_mask: the raw semantic masks with shape [b,m,H,W]
   # obj_feat: the object features to generate masks with shape [b, m, d]
   # b is the batch size, m the number of object and H, W height and width.

   b, m, H, W = raw_mask.size()
   obj_feat = obj_feat.view(b * m, -1)
   # object key
   obj_key = self.fc_key(obj_feat).view(b, m, -1)
   # transposed object query
   obj_query = self.fc_query(obj_feat).view(b, m, -1).permute(0, 2, 1)

   # resize the raw mask to 64 * 64
   resized_raw_mask = F.interpolate(raw_mask, size=(64, 64)).view(b, m, 64 * 64)
   # calculate pixel query in Eq (2)
   pixel_query = torch.bmm(obj_query, resized_raw_mask).view(b, -1, 64, 64)
   # calculate local query with ResNet blocks
   local_query = self.res_blocks(pixel_query).view(b, -1, 64 * 64)

   # Eq (3)
   E = torch.bmm(obj_key, local_query).view(b, m, 64, 64)
   # calculating scaling factors in Eq (4)
   factors = F.tanh(E * self.alpha) + 1
   # resize factors back to H * W
   factors = F.interpolate(factors, size=(H, W))
   # apply the factors in Eq (1)
   adapted_mask = raw_mask * factors

   # normalization
   adapted_mask = F.normalize(adapted_mask, p=1, dim=1)

   return adapted_Mask
```

Figure S1. Python code of LAMA based on PyTorch [25].

are transformed from $\boldsymbol{O}^{\mathsf{T}} \otimes \boldsymbol{M}$, as suggested by ISLA-norm [30]. The Batch-Group Normalization is inspired by Batch-Instance Normalization [24]. It is different from a recent work with the same name [40], who normalizes feature in groups of channels across a batch.

Noise Injection [14]. Noise injection is to add spatially-uncorrelated Gaussian noise to hidden feature maps after each convolution. The noise is in single-channel and broadcasted to all channels using learned deviations. Thus we encourage the deviations to be non-negative with the softplus operation. The noise injection is to provide intermediate stochasticity for each layer to adopt and thus enrich the hidden features.

ReZero [1]. ReZero is to smooth gradients in the early stage of training and accelerate the convergence. It designs a parameter τ initialized as 0, which scales the result of the main transformation in the ResNet. Formally, given a hidden feature $\boldsymbol{X}$ and a transformation $\boldsymbol{\mathcal{F}}$, ReZero gives a result as

$$\boldsymbol{X} + \tau \times \boldsymbol{\mathcal{F}}(\boldsymbol{X}) \tag{2}$$

As τ can be absorbed by convolutions in $\boldsymbol{\mathcal{F}}$, it does not affect the final solution.

The whole generator. The generator takes $\boldsymbol{z_{img}}$ as input, and gradually enriches the hidden features with the stacked ResNet blocks. Finally, the hidden features are transformed into the image with a $\tanh$ function. Our implementation is built on top of LostGAN-V1's generator [29].

S1.3. Loss

We use the adversarial training strategy [6, 22] to train our layout-to-image generation model with a discriminator $\boldsymbol{D}$. The discriminator has two branches, the image branch $\boldsymbol{D_{img}}$ and the object branch $\boldsymbol{D_{img}}$. The discriminator consists of ResNet [8] blocks with Spectral Normalization [22] to form the image branch $\boldsymbol{D_{img}}$. To classify objects in bounding boxes $\boldsymbol{L}$, the object branch $\boldsymbol{D_{obj}}$ uses ROI Align [7] to extract feature maps and identifies the objects with a projection discriminator [23]. The whole training loss consists of the adversarial hinge loss and a classification loss (3).

$$\begin{aligned}
\mathcal{L} =& \mathcal{L}_{img} + \mathcal{L}_{obj} \\
\mathcal{L}_{img} =& \mathbb{E}_{x\sim P_d} \min(0, 1 - D_{img}(x)) \\
&+ \mathbb{E}_{x\sim P_g} \min(0, 1 + D_{img}(x)) \\
\mathcal{L}_{obj} =& \mathbb{E}_{x\sim P_d} \min(0, 1 - D_{obj}(x, L)) \\
&+ \mathbb{E}_{x\sim P_g} \min(0, 1 + D_{obj}(x, L))
\end{aligned} \tag{3}$$

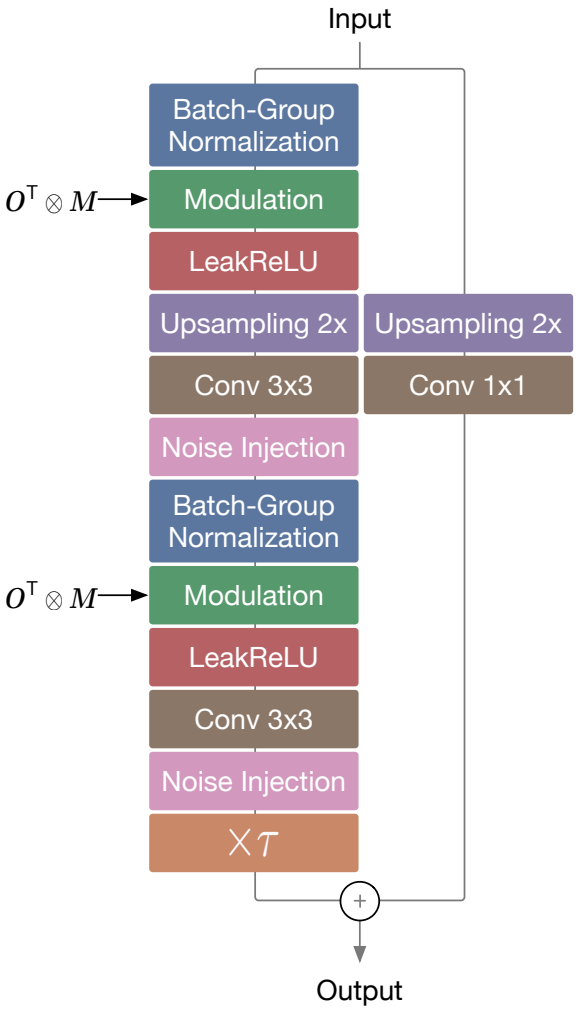

Figure S2. The architecture of ResNet blocks in mask-to-image generator. Object features are aggregated with the generated masks and transformed as affine parameters in the modulation parts after normalizations.

Here $\boldsymbol{P_d}$ and $\boldsymbol{P_g}$ are the real and generated distribution.

S2. Discussion

In this section, we discuss several topics on layout-to-image generation and our method.

Causal relation. LAMA mainly captures the correlation of objects' appearance. A more profound objects' interrelation is causal relation [21]. The objects' causal relation studies the causality of appearance between objects. In the layout-to-image scenario, the causal relations are mainly reflected by the shape of masks. Take the example of giraffes and bush in Figure S3. In (a) the bush in the foreground causes the shape of two giraffes to remove legs, while in (b) and (c) the giraffe in the foreground causes the background bush to disappear in the corresponding area. From the causal perspective, a more robust relation of appearance may be captured. We will explore this in our future work.

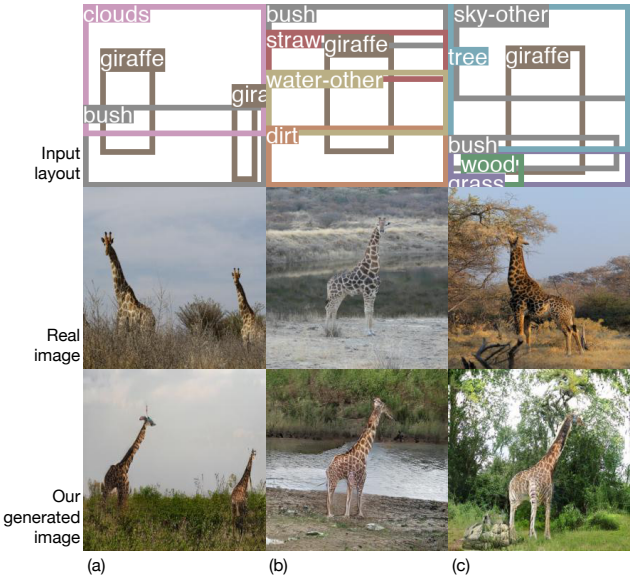

Figure S3. Example scenes where objects' visibility is different when their bounding boxes overlap. Best viewed magnified. See Section S2 for details.

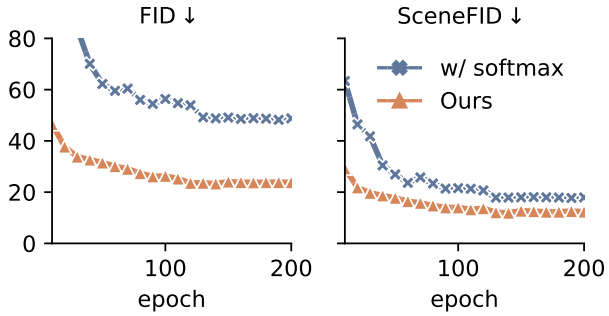

Figure S4. Performances of a LAMA variant to infer new masks directly with a softmax operation as output in Section S2.

Inferring new masks. Why do we adapt the raw mask rather than infer a clean mask directly? The reason is the object mask generator $\boldsymbol{F_{om}}$ is not aware of overlaps before object masks are mapped to bounding boxes. Thus, in our scenario overlaps cannot be avoided until the raw mask is formed. ConvLSTM [28] modelling relations can be used to generate masks [12, 20], but the training requires segmentation annotation. Except scaling, an alternative implementation is to translate the raw mask to a new one by applying softmax to each $\boldsymbol{E}_{\cdot j}$ feature. However, gradients may vanish in the softmax layer, and there is no shortcut to bypass gradients to $\boldsymbol{F_{om}}$ like the current design does, causing difficulty for convergence. To support our state-

ment, an experiment on the discussed variant is conducted on COCO [3] with 128×128. The difficulty for convergence is not revealed by loss plots, which are dominated by the discriminator. It is reflected by the performance degrade (Figure S4), and the result supports our current design.

Similarity to the attention mechanism. LAMA's computation seems similar to the attention mechanism [32] in SAGAN [36], but there are two differences. On the one hand, LAMA matches pixels and objects with a complexity of $\mathcal{O}(mWH)$, while SAGAN matches the key and the query both of pixels with $\mathcal{O}(W^2H^2)$. On the other hand, the matching value $\boldsymbol{E}$ has a different meaning. In LAMA, $\boldsymbol{E}$ determines scaling factors directly through a $\tanh$ function, while in SAGAN it is transformed to weights of attention with a softmax operation.

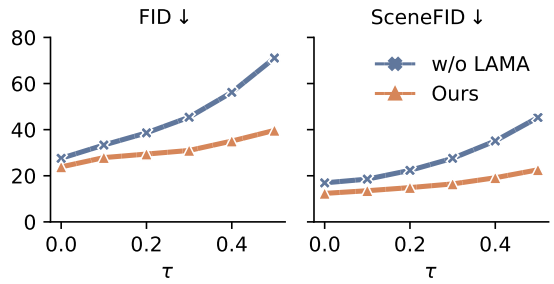

Figure S5. Performances with tainted raw masks in Section S2.

Refining tainted raw masks. To show LAMA is robust to mask ambiguity, we perform experiments on LAMA with taint raw masks before adaption, similar to those experiments in Section 3. Specifically, the raw mask $\tilde{\boldsymbol{M}}$ is tainted as $(1-\tau) \times \tilde{\boldsymbol{M}} + \frac{\tau}{m} \times \mathbf{1}(\tilde{\boldsymbol{M}})$, where $\boldsymbol{\tau} \in [0,1]$ and $\mathbf{1}(\tilde{\boldsymbol{M}})$ is the all-one tensor with the same size as $\tilde{\boldsymbol{M}}$. The tainted masks are then refined by LAMA and injected into the image generation pipeline. Empirically, our model is more robust to mask taint with LAMA than without (Figure S5). This exemplifies the robustness of LAMA and LAMA's ability to alleviate mask unclarity.

Potential scaling rules. Our paper proposes an adaption approach to aggregate overlapped object masks. A natural question is whether there is a simple adaption rule, which may strengthen or shrink masks according to a predefined object priority. We argue such priority may not be available in the layout-to-image scenario. For example, Figure S3 depicts images of giraffes and bush, where bounding boxes of "giraffe" and that of "bush" overlap. In (a) giraffes are standing behind the bush, and in this case we should have the mask of "bush" strengthen. However, in (b) and (c) we have giraffes standing before bush, and the adaption rule should be reversed. This exemplifies in the real scenario object relations are too complex to be described by a simple priority. In the third row of Figure S3 we show LAMA generates plausible images in this case.

Softmax fusion. Another simple adaption rule is to apply a pixelwise softmax fusion with a low temperature on the raw mask. The softmax fusion may strengthen or shrinken the mask values. However, we hypothesize overlaped objects with large mask values has similar new values after softmax fusion, even with a low temperature. Besides, as raw masks are in $(0,1)$ with differences not big enough, masks after softmax are close to $\frac{1}{\boldsymbol{m}}$ and not sharper, with $\boldsymbol{m}$ the number of objects. To verify our hypothesis, we train a LostGAN-V1 variant [29] with a softmax mask fusion on COCO [3] in the size of 128×128, with a learnable temperature in $(0,1]$ from 0.9. The resulting model has an FID 34.74 and a SceneFID 19.99, while the original LostGANv1 has 29.65 and 20.03. Besides, the resulting model has a mask entropy 2.01 ± 0.03, higher than the original 0.27 ± 0.15. The finally learned temperature is 1.0, and the model does not optimize a lower one. Thus, the softmax fusion may not help in this scenario.

Disentangling shape and texture. The designed layout-mask-image pipeline implicitly disentangles object shapes and texture with the inductive bias in the model. Specifically, the semantic mask generator provides only the shape information of objects, as the mask only represents the pixelwise visibility strength of objects. On the other hand, the mask-to-image generator only considers the texture. Although translation models like pix2pixHD [33] or CRN [5] are able to learn both shapes and texture simultaneously with convolution operations, their models accept inputs only in the beginning. In our generator, the generated semantic masks are injected into all hidden normalization layers, with which the shape information is constantly maintained. If the model severely changes the object shapes with convolution, the following normalization layer will strengthen the originally generated mask again. Thus, the mask-to-image generator only provides texture information. This is how shape and texture are disentangled. Such disentanglement is based on the inductive bias of model architecture instead of independence priors [11].

In the training, gradients about the appearance of objects are back-propagated to the mask generator and mask-to-image generator guided by bounding boxes. Similarly, the shape and texture information in gradients are disentangled, and that is why semantic masks can be learned in a weakly-supervised manner.

Reconfigurability. Reconfigurability means that a model can preserve most generated objects unchanged given per-

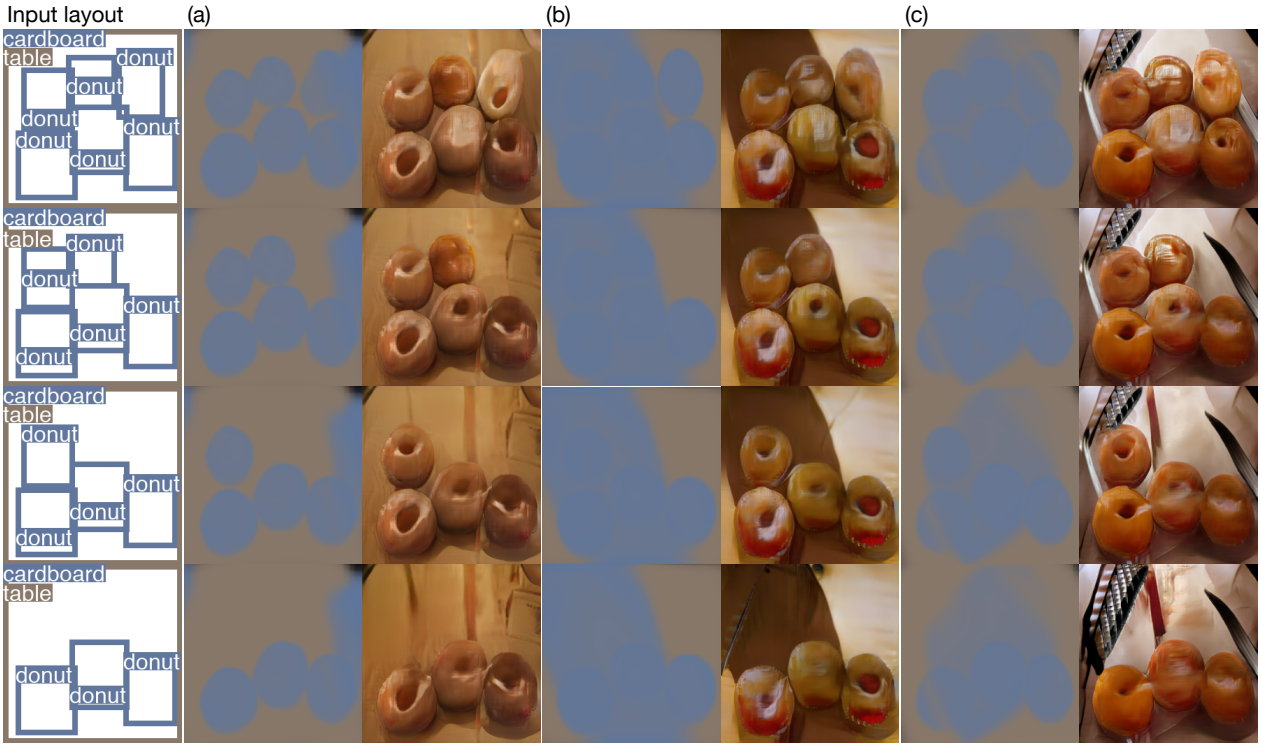

Figure S6. Examples of reconfigurability. By adding donuts, existing donuts and the background table remains steady.

turbations of layout and object style [29, 30]. Reconfigurability is intuitive and improves the controllability of generated images. With reconfigurability, it is possible to manually adjust the generated result as needed. Without this, the change of a bounding box like moving or scaling can cause the change of other objects. Our model is locality-aware as only adjacent or overlapped objects are correlated, and thus our model is reconfigurable. We demonstrate examples of reconfigurability in Figure S6 and S7.

Model complexity. LAMA introcuces extra parameters of complexity $\mathcal{O}(dd' + k'^2 d'^2)$ and an computation complexity of $\mathcal{O}(md'WH + k'^2 d'^2 WH + mdd')$. Parameters are in ResNet Blocks for local query and transformations to object query and object key. Here $\boldsymbol{k' = 3}$ is the kernel size of convolutions in ResNet blocks. Recall that $\boldsymbol{m}$ is the number of objects, $\boldsymbol{d}$ is the size of object features and $\boldsymbol{d'}$ is the size of object key and query in LAMA. In the implementation, the LAMA module has about 0.1M parameters.

S3. More Experimental Results

S3.1. Inception Score

We present the evaluation with Inception Score [27] in Table S1. Inception Score is to estimate the overall visual quality. The performance of our model is comparable with other leading methods.

Table S1. Evaluation of Inception Score [27] on COCO-Stuff [3] and Visual Genome (VG) [16]. A higher value is better. Best performances are highlighted.

Size	Methods	Datasets	
		COCO	VG
64×64	Real Images	16.30 ± 0.40	13.90 ± 0.50
	Layout2Im [38]	9.10 ± 0.10	8.10 ± 0.10
	Layout2Im+OWA [39]	9.70 ± 0.10	8.00 ± 0.20
	LostGAN-V1 [29]	9.80 ± 0.20	8.70 ± 0.40
	OC-GAN [31]	$\mathbf{10.80} \pm 0.50$	$\mathbf{9.30 \pm 0.20}$
	Ours	9.77 ± 0.28	8.12 ± 0.08
128×128	Real Images	22.30 ± 0.50	20.50 ± 1.50
	LostGAN-V1 [29]	13.80 ± 0.40	11.10 ± 0.60
	LostGAN-V2 [30]	14.21 ± 0.40	10.71 ± 0.26
	OC-GAN [31]	$\mathbf{14.46 \pm 0.37}$	$\mathbf{12.30 \pm 0.40}$
	Ours	14.46 ± 0.37	10.74 ± 0.09
256×256	Real Images	28.10 ± 1.60	28.60 ± 1.20
	LostGAN-V2 [30]	18.01 ± 0.50	14.10 ± 0.38
	OC-GAN [31]	17.80 ± 0.20	$\mathbf{14.70 \pm 0.20}$
	Ours	$\mathbf{19.13 \pm 0.41}$	13.52 ± 0.15

S3.2. More Metrics

We conduct extra experiments along metrics introduced in Casanova *et al.* [4]. Specifically, we follow the protocol to use ResNext-101 [35] features and approximate manifolds on real or generated images with k-nearest-neighborhood graph, with $k = 5$. Then we use improved

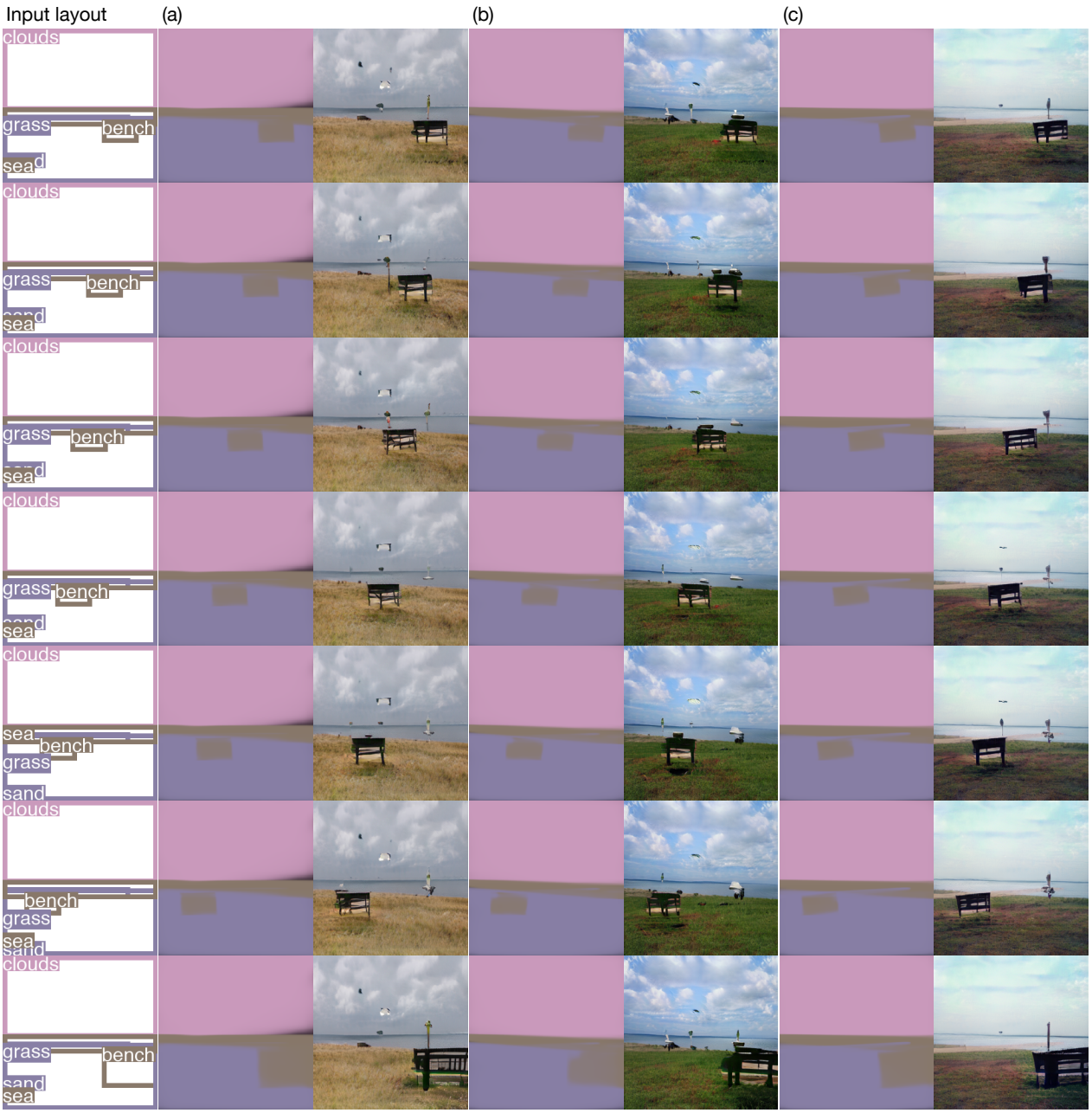

Figure S7. Examples of reconfigurability. By moving and scaling the bench bounding box, the background grassfield, sea and sky remain steady.

precision and recall [19] [1] and also layout consistency [4] to estimate generative quality. The improved precision estimates whether generated images lie on the real manifold, while the recall estimates whether real ones lie on the generated manifold. The consistency measures whether a generated image shares the same layout with nearby real images. Higher values are preferred for these three metrics. The results on scenes images are shown in Table S2 and on object crops in Table S3. Best performances are highlighted. Our method has better performances in most cases.

S3.3. A Variant with ConvLSTM

We further examine a variant of our model by replacing LAMA with ConvLSTM [28]. Particularly, we give the raw semantic mask as the initial hidden state. Given the object i, its object feature is spatially broadcasted to the resized ob-

[1] We use the implementation in github.com/youngjung/improved-precision-and-recall-metric-pytorch

Table S2. Precision, recall and consistenty on generated scene images (Section S3.2).

Size	Methods	Precision ↑		Recall ↑		Consistency ↑	
		COCO	VG	COCO	VG	COCO	VG
64 × 64	LostGAN-V1 [29]	**0.8269**	**0.7890**	**0.5172**	**0.4798**	**0.2639**	**0.1052**
	Ours	0.7904	0.7805	0.5011	0.4782	0.1315	0.1009
128 × 128	LostGAN-V1 [29]	**0.7484**	0.6531	0.2786	0.2850	**0.2104**	0.0749
	LostGAN-V2 [30]	0.6561	0.7099	0.3810	0.4045	0.0927	**0.1492**
	Ours	0.6703	**0.7224**	**0.3994**	**0.4529**	0.0953	0.0886
256 × 256	LostGAN-V2 [30]	0.7103	**0.6774**	**0.4824**	0.4061	**0.2254**	**0.1536**
	Ours	**0.7319**	0.6598	0.4204	**0.4407**	0.1182	0.0777

Table S3. Precision, recall and consistenty on generated object crops (Section S3.2).

Size	Methods	Precision ↑		Recall ↑		Consistency ↑	
		COCO	VG	COCO	VG	COCO	VG
64 × 64	LostGAN-V1 [29]	**0.7845**	**0.8124**	**0.5306**	**0.6343**	**0.3540**	**0.2835**
	Ours	0.7218	0.7932	0.3747	0.6597	0.2512	0.2380
128 × 128	LostGAN-V1 [29]	**0.7224**	0.7564	0.3187	0.4261	0.3104	0.2524
	LostGAN-V2 [30]	0.7146	0.7528	0.4178	0.4533	0.3324	0.2507
	Ours	0.7158	**0.7576**	**0.4679**	**0.5377**	**0.3598**	**0.3086**
256 × 256	LostGAN-V2 [30]	0.6321	0.7019	**0.3833**	0.3901	0.2895	0.2609
	Ours	**0.6547**	**0.7238**	0.3652	**0.4448**	**0.3336**	**0.3169**

ject mask, and this spatial object feature is given as the input of LSTM model. We stack three layers of ConvLSTM, and take the number of objects as hidden dimension. Finally, the output of ConvLSTM is treated as the adaption and added to the raw mask to form the final one. Such residual design helps to propagate gradient to object mask generator as our original design does. Following Layout2Im [38], we shuffle the order of objects and form a random input sequence. The quantitative result is summarized in Table S4, and the performance decays.

S3.4. Mask Entropy

Mask Entropy evaluates the pixelwise semantic clarity of a mask, introduced in Section 5.3 in the main paper. We present the comparison on mask entropies of LostGANs [29, 30] and ours in Table S5. We used official pretrained models of LostGANs to generate masks. Specifically, LostGAN-V2 refines masks based on feature maps in different scales of the mask-to-image generator. We present the average value of masks across scales. Results show our generated mask enjoys lower mask entropy values.

S3.5. Generated Masks Demonstration

We display generated images with their raw and adapted masks to show the mask adaption capacity of the proposed LAMA module (Figure S9). The masks are more semantically clear and clean after adaption with little ambiguity and sharp boundaries.

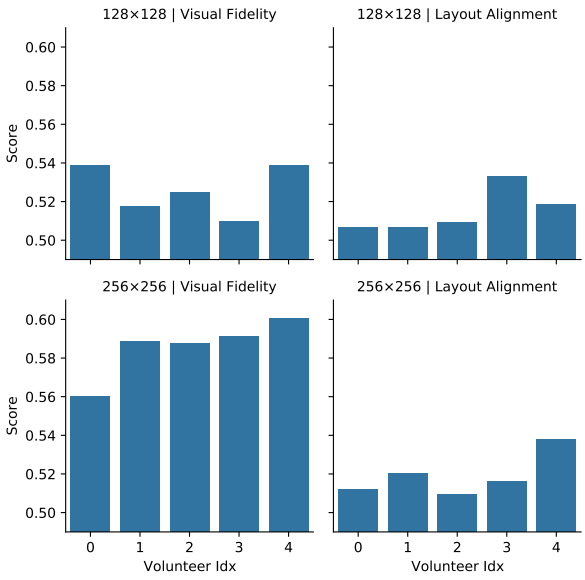

Figure S8. Individual preference scores in human evaluation experiments.

S3.6. User Studies

We conduct user studies to evaluate visual fidelity or layout alignment separately. Experiments are performed on biao.jd.com/wise, an online crowdsourcing platform. Images are generated by LostGAN-V2 [30] and our model according to 3,097 layouts of COCO val2017. Same-layout images are shuffled and displayed. Four groups of anonymous workers rank images in 128 or 256 based on visual fi-

Table S4. Comparison with a variant of our model by replacing LAMA with ConvLSTM [28].

Methods	IS ↑	FID ↓	DS ↑	CAS ↑	SceneFID ↓	AP ↑	AP_{50} ↑	AP_{75} ↑
Ours	14.46 ± 0.37	23.85	0.46 ± 0.09	34.15	12.35	7.9%	12.0%	8.9%
Ours w/ ConvLSTM [28]	11.10 ± 0.35	37.28	0.55 ± 0.08	31.24	20.78	2.3%	3.6%	2.3%

Table S5. Mask entropies of masks generated by LostGANs [29, 30] and our method.

Size	Datasets	LostGAN-V1 [29]	LostGAN-V2 [30]	Ours
64×64	COCO	0.33 ± 0.18	-	0.11 ± 0.07
	VG	0.57 ± 0.25	-	0.10 ± 0.09
128×128	COCO	0.27 ± 0.15	0.30 ± 0.16	0.14 ± 0.11
	VG	0.51 ± 0.23	0.55 ± 0.19	0.12 ± 0.09
256×256	COCO	-	0.29 ± 0.16	0.14 ± 0.10
	VG	-	0.52 ± 0.19	0.26 ± 0.17

Table S6. Mean preference scores in human evaluation experiments.

Size	Visual Fidelity ↑	Layout Alignment ↑
128 × 128	52.58% ± 1.28%	51.50% ± 1.13%
256 × 256	58.58% ± 1.36%	51.94% ± 0.99%

delity or layout alignment separately. Each group has three males and two females, aged from 20 to 28, with majors in different disciplines. They are given unlimited time to make choices. Table S6 summarizes the results, in which the mean is the average of five scores. Our method has higher mean scores. In fact, the percentage that each worker favors ours is over 50% in all cases (Figure S8).

S3.7. Diversity

This part presents more qualitative results to demonstrate the generative diversity of our proposed model. Given an input layout, the generated images are determined by the image style z_{img}, object variations z_i and noise injection in the generative process. We gradually fix these components to show the diversity and the visual quality of the model.

Image diversity of different layouts. We demonstrate the general image diversity of the same layout in Figure S10 (64 × 64), Figure S11 (128 × 128) and Figure S12 (256 × 256).

Image diversity of different image styles z_{img}. This is to show the style diversity of image styles by displaying images of different z_{img} (Figure S13). Besides, given the same layout and z_{img}, the diversity introduced by z_i's is shown in Figure S14.

Image diversity of an fixed image style z_{img} and fixed z_i's. This is to show the diversity introduced by noise injection given the same z_{img} and z_i's (Figure S15).

S3.8. Failure Cases

We present some failure cases in Figure S16. Failure includes mode collapse, artifacts and the failure to generate complex objects. Especially, our model suffers from failure in generating "zebra" objects. This may result from the difficulty to generate high-frequency zebra stripes.

S3.9. Metric Details

We introduce more implementational details about the adopted metrics in experiments.

Inception Score (IS) [27], **Frèchet Inception Distance (FID)** [10] and **SceneFID** [31]. For each layout, the model generate five new images. This augmented generated set of images and the validatation set are used to calculate the IS and FID value. For SceneFID, objects in images are cropped and resized to 224 × 224, and the FID is computed with 224 × 224 validatation object crops.

Diversity Score (DS). We mainly use LPIPS [37] with a pre-trained AlexNet [18] to compare two groups of generated images.

Classification Accuracy Score (CAS) [38]. To compute CAS score, a ResNet-101 [8] is trained on generated object crops and tested on validation crops. For each layout in the validation set, five images are generated and the objects are cropped and resized to 32 × 32. This forms the training set. The same cropping and resizing process is also applied to the validation images to form the testing set. The training strategy is designed for CIFAR10/100 [17] in `github.com/hysts/pytorch_image_classification`. Finally, the top-1 testing accuracy is reported as the CAS score.

YOLO scores. We use the official implementation and pre-trained YOLOv4 [2] model in `github.com/AlexeyAB/darknet`. For each layout, only one image is generated and resized to 512 × 512. The generated images are measured with AP (average precision), AP_{50} and AP_{75} with the COCO API in `github.com/cocodataset/cocoapi`.

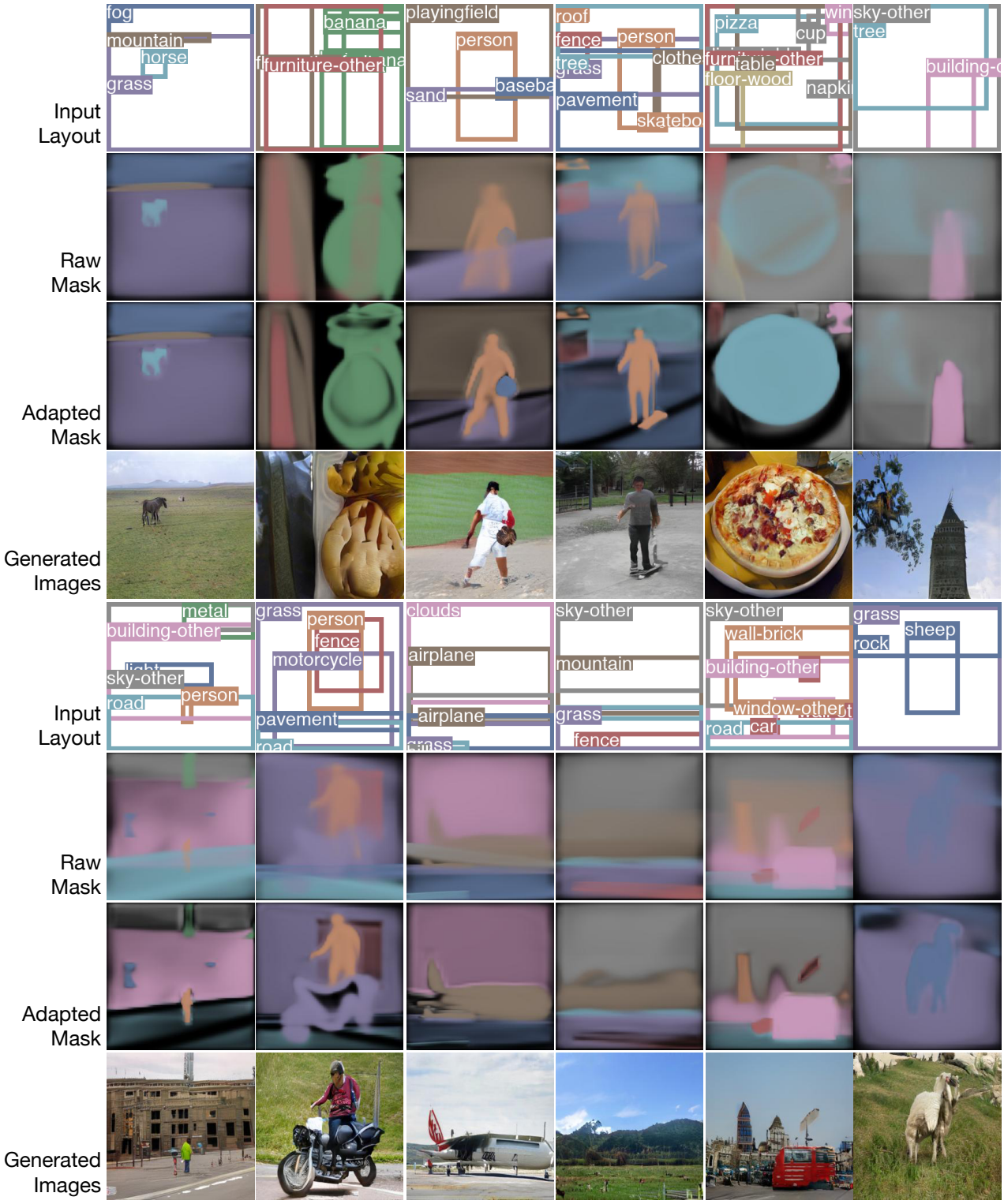

Figure S9. Generated 256 × 256 raw and adapted masks. The raw masks seem blur and ambiguous, while the adapted masks are more semantically clear.

References

[1] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence (UAI)*. Proceedings of

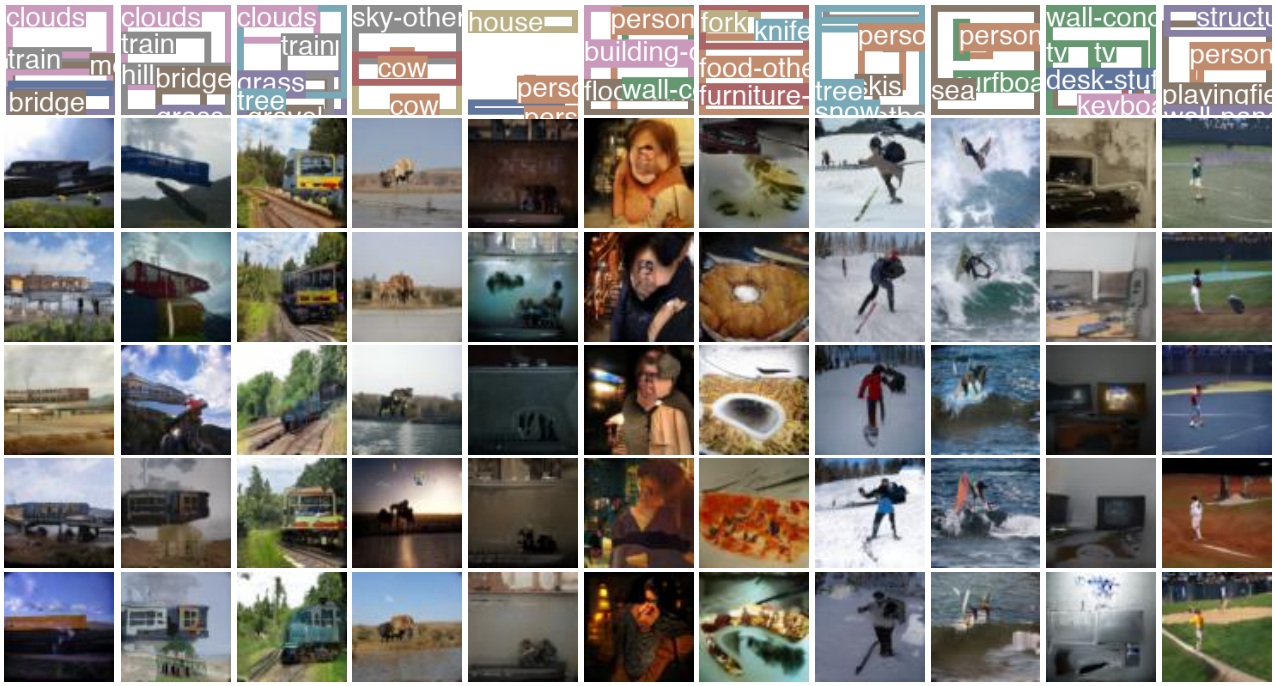

Figure S10. Generated 64×64 images given different layouts.

Machine Learning Research, 2021. 2

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: optimal speed and accuracy of object detection. *ArXiv*, abs/2004.10934, 2020. 8

[3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and stuff classes in context. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4, 5

[4] Arantxa Casanova, Michal Drozdzal, and Adriana Romero-Soriano. Generating unseen complex scenes: are we there yet? *ArXiv*, abs/2012.04027, 2020. 5, 6

[5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. *IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1529, 2017. 4

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680. Curran Associates, Inc., 2014. 2

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 2, 8

[9] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–646. Springer, 2016. 1

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6626–6637. Curran Associates, Inc., 2017. 8

[11] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017. 4

[12] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and H. Lee. Inferring semantic layout for hierarchical text-to-image synthesis. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7986–7994, 2018. 3

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015. 1

[14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 2

[15] Mahyar Khayatkhoei and Ahmed Elgammal. Spatial frequency bias in convolutional generative adversarial networks. *ArXiv*, abs/2010.01473, 2020. 16

[16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 123(1):32–73, 2017. 5

Figure S11. Generated 128×128 images given different layouts.

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009. 8

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 1097–1105. Curran Associates Inc., 2012. 8

[19] T. Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 6

[20] Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. Object-driven text-to-image synthesis via adversarial training. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12166–12174, 2019. 3

[21] David Lopez-Paz, Robert Nishihara, Soumith Chintala, B. Schölkopf, and Le on Bottou. Discovering causal signals in images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 58–66, 2017. 3

[22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[23] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[24] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2558–2567. Curran Associates Inc., 2018. 2

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035. Curran Associates, Inc., 2019. 1, 2

[26] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and

Figure S12. Generated 256×256 images given different layouts.

Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2019. 16

[27] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2234–2242. Curran Associates, Inc., 2016. 5, 8

[28] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 1–9. Curran Associates, Inc., 2015. 3, 6, 8

[29] Wei Sun and Tianfu Wu. Image synthesis from reconfigurable layout and style. In *IEEE International Conference on Computer Vision (ICCV)*, pages 10531–10540, 2019. 1, 2, 4, 5, 7, 8

[30] Wei Sun and Tianfu Wu. Learning layout and style reconfigurable GANs for controllable image synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 2, 5, 7, 8

[31] Tristan Sylvain, Pengchuan Zhang, Yoshua Bengio, R. Devon Hjelm, and Shikhar Sharma. Object-centric image gen-

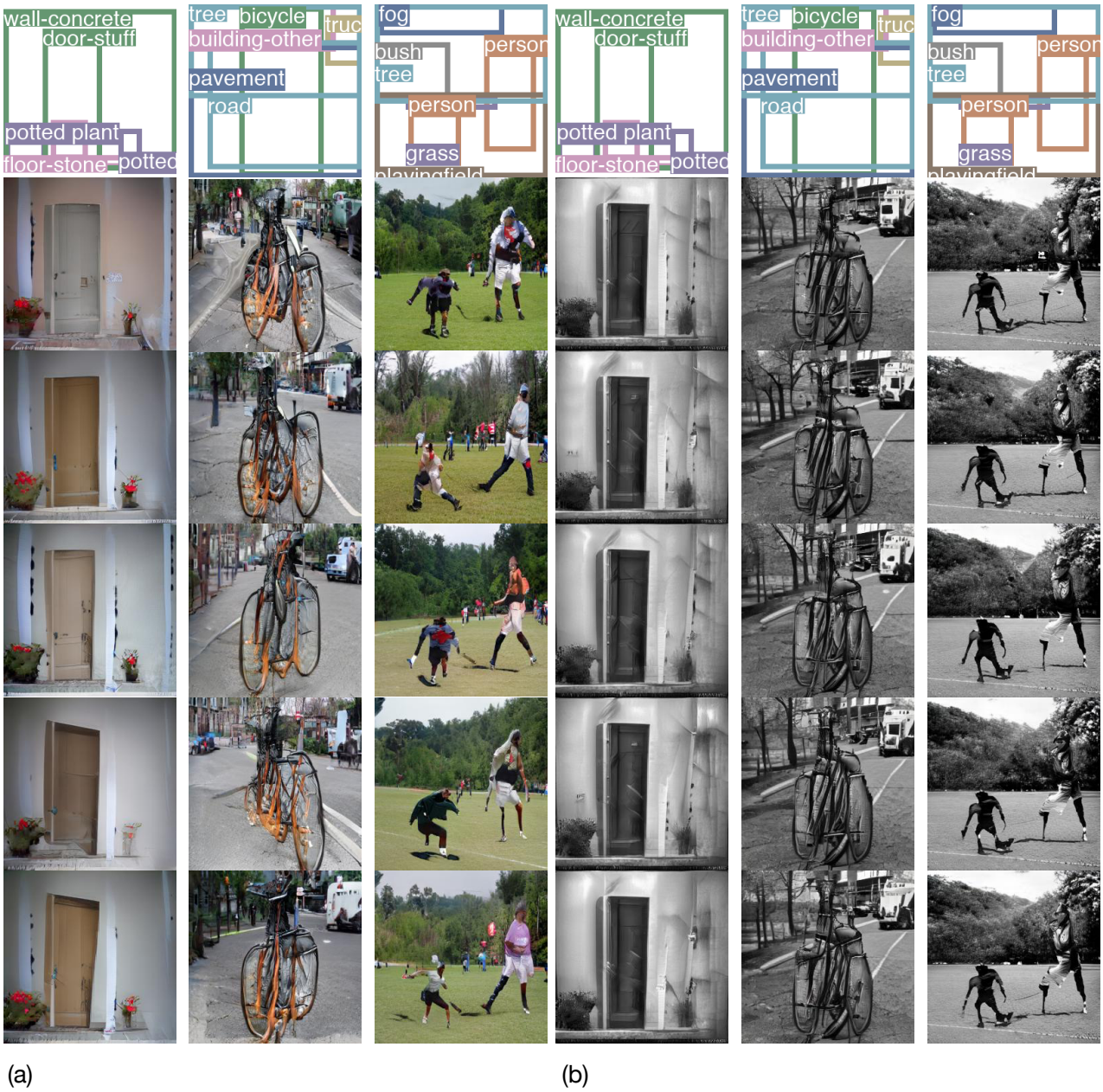

Figure S13. Generated 256×256 images given two different z_{img}'s in (a) and (b).

eration from layouts. In *International Conference on Learning Representations (ICLR)*, 2021. 5, 8

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 6000–6010. Curran Associates Inc., 2017. 4

[33] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4

[34] Yuxin Wu and Kaiming He. Group normalization. *International Journal of Computer Vision (IJCV)*, 128:742–755, 2019. 1

[35] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017. 5

[36] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 7354–7363. PMLR, 2019. 4

[37] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shecht-

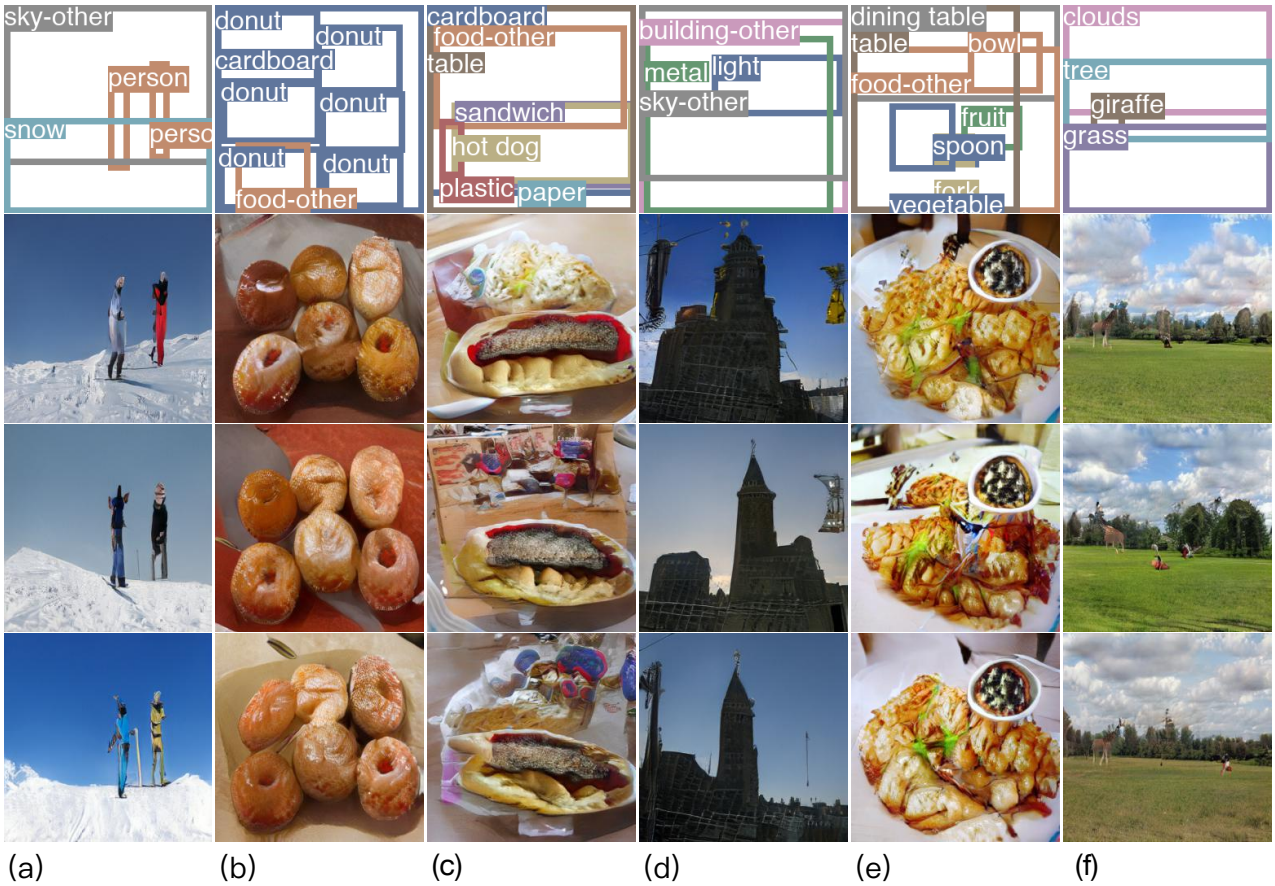

Figure S14. Generated 256×256 images of multiple z_{img}'s. Each column has the same z_{img}, and these are to show the diversity introduced by object variation z_i's.

man, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 8

[38] Bo Zhao, Lili Meng, Weidong Yin, and Leonid Sigal. Image generation from layout. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8584–8593, 2019. 5, 7, 8

[39] Bo Zhao, Weidong Yin, Lili Meng, and Leonid Sigal. Layout2image: Image generation from layout. *International Journal of Computer Vision (IJCV)*, pages 1–18, 2020. 5

[40] Xiaoyun Zhou, Jiacheng Sun, Nanyang Ye, X. Lan, Q. Luo, Bolin Lai, Pedro M. Esperança, G. Yang, and Zhenguo Li. Batch group normalization. *ArXiv*, abs/2012.02782, 2020. 2

Figure S15. Generated 256×256 images given the same z_{img} and z_i's. Each column of images share the same z_{img} and z_i's. These are to show fine-grained stochastic variation introduced by noise injection in the generator.

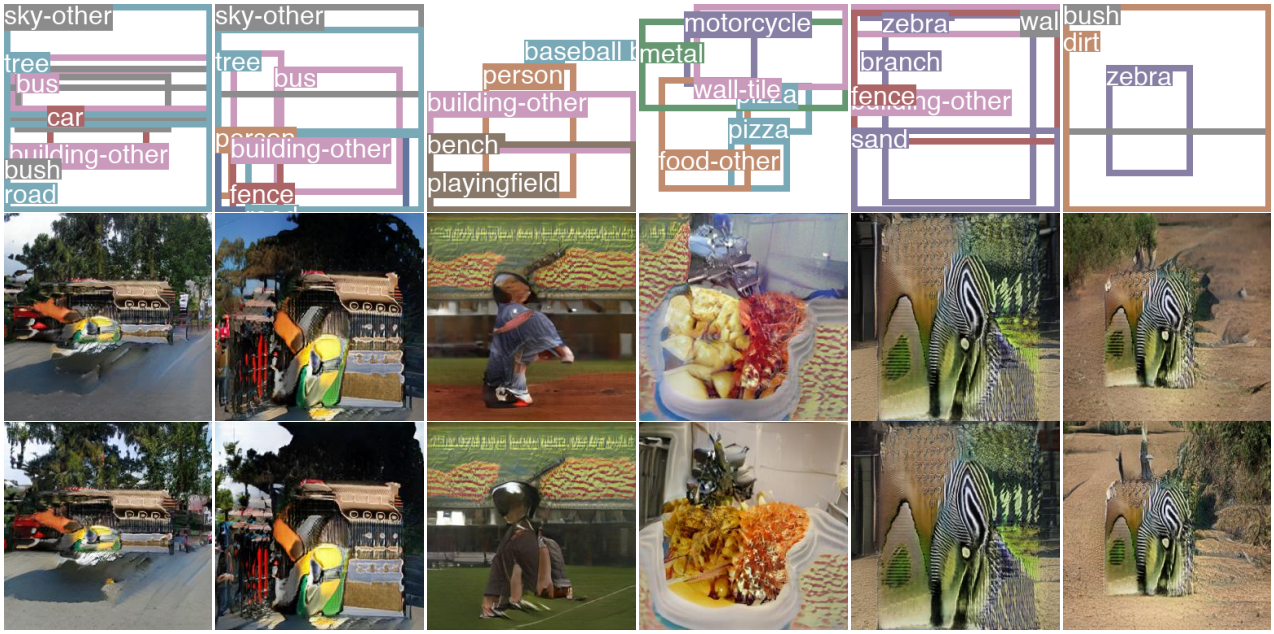

Figure S16. Generated 256×256 failure cases. The typical failure cases of the proposed model include mode collapse (a), artifacts (b) and the inability to learn complex objects (c). (a) The model ignores the stochastic variation in the model and generates nearly the same images. (b) In the blank area not covered by bounding boxes, the model generates artifacts. (c) The model fails to generate zebras, which has a complex texture. Such texture is high-frequency signals, which the model may fail to fit and simply ignore [15, 26].