Object-driven Text-to-Image Synthesis via Adversarial Training

Wenbo Li[†*1,2]   Pengchuan Zhang[*2]   Lei Zhang[3]

Qiuyuan Huang[2]   Xiaodong He[4]   Siwei Lyu[1]   Jianfeng Gao[2]

[1]University at Albany, SUNY   [2]Microsoft Research AI   [3]Microsoft   [4]JD AI Research

{wli20,slyu}@albany.edu, {penzhan,leizhang,qihua,jfgao}@microsoft.com, xiaodong.he@jd.com

Abstract

In this paper, we propose Object-driven Attentive Generative Adversarial Newtorks (Obj-GANs) that allow object-centered text-to-image synthesis for complex scenes. Following the two-step (layout-image) generation process, a novel object-driven attentive image generator is proposed to synthesize salient objects by paying attention to the most relevant words in the text description and the pre-generated semantic layout. In addition, a new Fast R-CNN based object-wise discriminator is proposed to provide rich object-wise discrimination signals on whether the synthesized object matches the text description and the pre-generated layout. The proposed Obj-GAN significantly outperforms the previous state of the art in various metrics on the large-scale COCO benchmark, increasing the Inception score by 27% and decreasing the FID score by 11%. A thorough comparison between the traditional grid attention and the new object-driven attention is provided through analyzing their mechanisms and visualizing their attention layers, showing insights of how the proposed model generates complex scenes in high quality.

1. Introduction

Synthesizing images from text descriptions (known as *Text-to-Image synthesis*) is an important machine learning task, which requires handling ambiguous and incomplete information in natural language descriptions and learning across vision and language modalities. Approaches based on Generative Adversarial Networks (GANs) [5] have recently achieved promising results on this task [23, 22, 32, 33, 29, 16, 9, 12, 34]. Most GAN based methods synthesize the image conditioned only on a global sentence vector, which may miss important fine-grained information at the word level, and prevents the generation of high-quality images. More recently, AttnGAN [29] is proposed which introduces the attention mechanism [28, 30, 2, 27] into the GAN framework, thus allows attention-driven, multi-stage

† Work was performed when was an intern with Microsoft Research AI.
* indicates equal contributions.
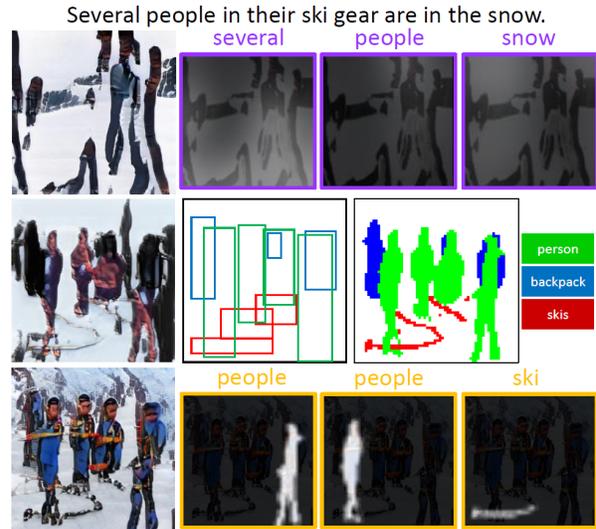

Figure 1: Top: AttnGAN [29] and its grid attention visualization. Middle: our modified implementation of two-step (layout-image) generation proposed in [9]. Bottom: our Obj-GAN and its object-driven attention visualization. The middle and bottom generations use the same *generated* semantic layout, and the only difference is the object-driven attention.

refinement for fine-grained text-to-image generation.

Although images with realistic texture have been synthesized on simple datasets, such as birds [29, 16] and flowers [33], most existing approaches do not specifically model objects and their relations in images and thus have difficulties in generating complex scenes such as those in the COCO dataset [15]. For example, generating images from a sentence "several people in their ski gear in the snow" requires modeling of different objects (people, ski gear) and their interactions (people on top of ski gear), as well as filling the missing information (*e.g.*, the rocks in the background). In the top row of Fig. 1, the image generated by AttnGAN does contain scattered texture of people and snow, but the shape of people are distorted and the picture's layout is semantically not meaningful. [9] remedies this problem by first constructing a semantic layout from the text and then synthesizing the image by a deconvolutional

image generator. However, the fine-grained word/object-level information is still not explicitly used for generation. Thus, the synthesized images do not contain enough details to make them look realistic (see the middle row of Fig. 1).

In this study, we aim to generate high-quality complex images with semantically meaningful layout and realistic objects. To this end, we propose a novel Object-driven Attentive Generative Adversarial Networks (Obj-GAN) that effectively capture and utilize fine-grained word/object-level information for text-to-image synthesis. The Obj-GAN consists of a pair of object-driven attentive image generator and object-wise discriminator, and a new object-driven attention mechanism. The proposed image generator takes as input the text description and a pre-generated semantic layout and synthesize high-resolution images via multiple-stage coarse-to-fine process. At *every* stage, the generator synthesizes the image region within a bounding box by focusing on words that are most relevant to the object in that bounding box, as illustrated in the bottom row of Fig. 1. More specifically, using a new object-driven attention layer, it uses the class label to query words in the sentences to form a word context vector, as illustrated in Fig. 4, and then synthesizes the image region conditioned on the class label and word context vector. The object-wise discriminator checks every bounding box to make sure that the generated object indeed matches the pre-generated semantic layout. To compute the discrimination losses for all bounding boxes simultaneously and efficiently, our object-wise discriminator is based on a Fast R-CNN [4], with a binary cross-entropy loss for each bounding box.

The contribution of this work is three-folded. (*i*) An Object-driven Attentive Generative Network (Obj-GAN) is proposed for synthesizing complex images from text descriptions. Specifically, two novel components are proposed, including the object-driven attentive generative network and the object-wise discriminator. (*ii*) Comprehensive evaluation on a large-scale COCO benchmark shows that our Obj-GAN significantly outperforms previous state-of-the-art text-to-image synthesis methods. Detailed ablation study is performed to empirically evaluate the effect of different components in Obj-GAN. (*iii*) A thorough analysis is performed through visualizing the attention layers of the Obj-GAN, showing insights of how the proposed model generates complex scenes in high quality. Compared with the previous work, our object-driven attention is more robust and interpretable, and significantly improves the object generation quality in complex scenes.

2. Related Work

Generating photo-realistic images from text descriptions, though challenging, is important to many real-world applications such as art generation and computer-aided design. There has been much research effort for this task

through different approaches, such as variational inference [17, 6], approximate Langevin process [24], conditional PixelCNN via maximal likelihood estimation [26, 24], and conditional generative adversarial networks [23, 22, 32, 33]. Compared with other approaches, Generative Adversarial Networks (GANs) [5] have shown better performance in image generation [21, 3, 25, 13, 11, 10]. However, existing GAN based text-to-image synthesis is usually conditioned only on the global sentence vector, which misses important fine-grained information at the word level, and thus lacks the ability to generate high-quality images. [29] uses the traditional grid visual attention mechanism in this task, which enables synthesizing fine-grained details at different image regions by paying attentions to the relevant words in the text description.

To explicitly encode the semantic layout into the generator, [9] proposes to decompose the generation process into two steps, in which it first constructs a semantic layout (bounding boxes and object shapes) from the text and then synthesizes an image conditioned on the layout and text description. [12] also proposes such a two-step process to generate images from scene graphs, and their process can be trained end-to-end. In this work, the proposed Obj-GAN follows the two-step generation process as [9]. However, [9] encodes the text into a single global sentence vector, which loses word-level fine-grained information. Moreover, it uses the image-level GAN loss for the discriminator, which is less effective at providing object-wise discrimination signal for generating salient objects. We propose a new object-driven attention mechanism to provide fine-grained information (words in the text description and objects in the layout) for different components, including an attentive seq2seq bounding box generator, an attentive image generator and an object-wise discriminator.

The attention mechanism has recently become a crucial part of vision-language multi-modal intelligence tasks. The traditional grid attention mechanism has been successfully used in modeling multi-level dependencies in image captioning [28], image question answering [30], text-to-image generation [29], unconditional image synthesis [31] and image-to-image translation [16], image/text retrieval [14]. In 2018, [1] proposes a bottom-up attention mechanism, which enables attention to be calculated over semantic meaningful regions/objects in the image, for image captioning and visual question-answering. Inspired by these works, we propose Obj-GAN which for the first time develops an object-driven attentive generator plus an object-wise discriminator, thus enables GANs to synthesize high-quality images of complicated scenes.

3. Object-driven Attentive GAN

As illustrated in Fig. 2, the Obj-GAN performs text-to-image synthesis in two steps: generating a semantic layout

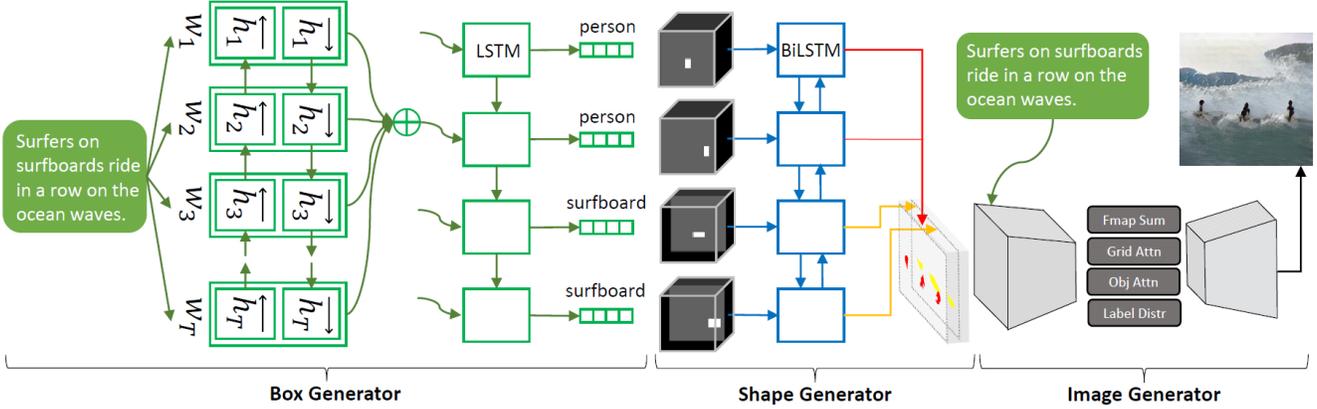**Box Generator**   **Shape Generator**   **Image Generator**

Figure 2: Obj-GAN completes the text-to-image synthesis in two steps: the layout generation and the image generation. The layout generation contains a bounding box generator and a shape generator. The image generation uses the object-driven attentive image generator.

(class labels, bounding boxes, shapes of salient objects), and then generating the image. In the image generation step, the object-driven attentive generator and object-wise discriminator are designed to enable image generation conditioned on the semantic layout generated in the first step.

The input of Obj-GAN is a sentence with T_s tokens. With a pre-trained bi-LSTM model, we encode its words as word vectors $e \in \mathbb{R}^{D \times T_s}$ and the entire sentence as a global sentence vector $\bar{e} \in \mathbb{R}^D$. We provide details of this pre-trained bi-LSTM model and the implementation details of other modules of Obj-GAN in § A.

3.1. Semantic layout generation

In the first step, the Obj-GAN takes the sentence as input and generates a semantic layout, a sequence of objects specified by their bounding boxes (with class labels) and shapes. As illustrated in Fig. 2, a box generator first generates a sequence of bounding boxes, and then a shape generator generates their shapes. This part resembles the bounding box generator and shape generator in [9], and we put our implementation details in § A.

Box generator. We train an attentive seq2seq model [2], also referring to Fig. 2, as the box generator:

$$B_{1:T} := [B_1, B_2, \ldots, B_T] \sim G_{\text{box}}(e). \tag{1}$$

Here, e are the pre-trained bi-LSTM word vectors, $B_t = (l_t, b_t)$ are the class label of the t's object and its bounding box $b = (x, y, w, h) \in \mathbb{R}^4$. In the rest of the paper, we will also call the label-box pair B_t as a bounding box when no confusion arises. Since most of the bounding boxes have corresponding words in the sentence, the attentive seq2seq model captures this correspondence better than the seq2seq model used in [9].

Shape generator. Given the bounding boxes $B_{1:T}$, the shape generator predicts the shape of each object in its bounding box, i.e.,

$$\widehat{M}_{1:T} = G_{\text{shape}}(B_{1:T}, z_{1:T}). \tag{2}$$

where $z_t \sim \mathcal{N}(0, 1)$ is a random noise vector. Since the generated shapes not only need to match the location and category information provided by $B_{1:T}$, but also should be aligned with its surrounding context, we build G_{shape} based on a bi-directional convolutional LSTM, as illustrated in Fig. 2. Training of G_{shape} is based on the GAN framework [9], in which a perceptual loss is also used to constrain the generated shapes and to stabilize the training.

3.2. Image generation

3.2.1 Attentive multistage image generator

As shown in Fig. 3, the proposed attentive multistage generative network has two generators (G_0, G_1). The base generator G_0 first generates a low-resolution image $\widehat{x}_0$ conditioned on the global sentence vector and the pre-generated semantic layout. The refiner G_1 then refines details in different regions by paying attention to most relevant words and pre-generated class labels and generates a higher resolution image $\widehat{x}_1$. Specifically,

$$h_0 = F_0(z, \quad \bar{e}, \quad \text{Enc}(M^0), c^{\text{obj}}, c^{\text{lab}}), \quad \widehat{x}_0 = G_0(h_0),$$

$$h_1 = F_1(c^{\text{pat}}, h_0 + \text{Enc}(M^1), c^{\text{obj}}, c^{\text{lab}}), \quad \widehat{x}_1 = G_1(h_1),$$

where (i) z is a random vector with standard normal distribution; (ii) $\text{Enc}(M^0)$ ($\text{Enc}(M^1)$) is the encoding of low-resolution shapes M^0 (higher-resolution shapes M^1); (iii) $c^{\text{pat}} = F_{\text{attn}}^{\text{grid}}(e, h_0)$ are the patch-wise context vectors from the traditional grid attention, (iv) $c^{\text{obj}} = F_{\text{attn}}^{\text{obj}}(e, e^g, l^g, M)$ are the object-wise context vectors from our new object-driven attention, and $c^{\text{lab}} = c^{\text{lab}}(l^g, M)$ are the label context vectors from class labels. We can stack more refiners to the generation process and get higher and higher resolution images. In this paper, we have two refiners (G_1 and G_2) and finally generate images with resolution 256×256.

Compute context vectors via attention. Both patch-wise context vectors c^{pat} and object-wise context vectors c^{obj} are attention-driven context vectors for specific image regions, and encode information from the words that are most relevant to that image region. Patch-wise context vectors are
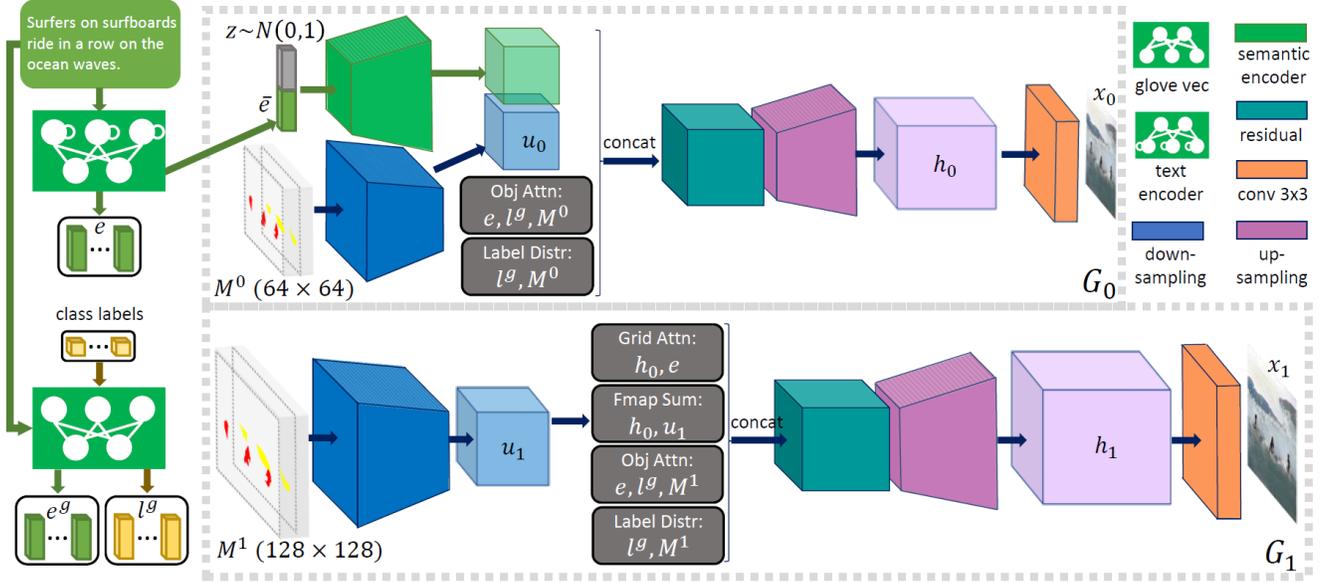
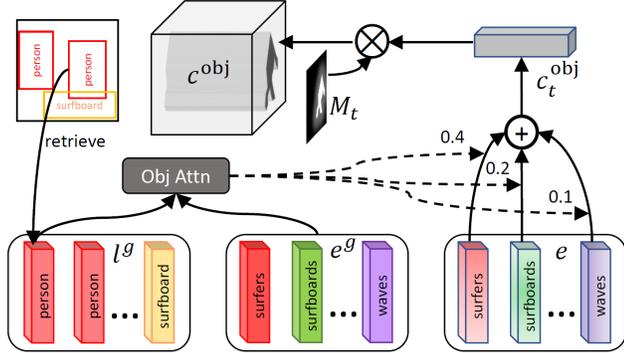Figure 3: The object-driven attentive image generator.

Figure 4: Object-driven attention.

for uniform-partitioned image patches determined by the uniform down-sampling/up-sampling structure of CNN, but these patches are not semantically meaningful. Object-wise context vectors are for semantically meaningful image regions specified by bounding boxes, but these regions are at different scales and may have overlaps.

Specifically, the patch-wise context vector c_j^{pat} (objective-wise context vector c_t^{obj}) is a dynamic representation of word vectors relevant to patch j (bounding box B_t), which is calculated by

$$c_j^{\text{pat}} = \sum_{i=1}^{T_s} \beta_{j,i}^{\text{pat}} e_i, \quad c_t^{\text{obj}} = \sum_{i=1}^{T_s} \beta_{t,i}^{\text{obj}} e_i. \quad (3)$$

Here, $\beta_{j,i}^{\text{pat}}$ ($\beta_{t,i}^{\text{obj}}$) indicates the weight the model attends to the i'th word when generating patch j (bounding box B_t) and is computed by

$$\beta_{j,i}^{\text{pat}} = \frac{\exp(s_{j,i}^{\text{pat}})}{\sum_{k=1}^{T_s} \exp(s_{j,k}^{\text{pat}})}, \quad s_{j,i}^{\text{pat}} = (h_j)^T e_i, \quad (4)$$

$$\beta_{t,i}^{\text{obj}} = \frac{\exp(s_{t,i}^{\text{obj}})}{\sum_{k=1}^{T_s} \exp(s_{t,k}^{\text{obj}})}, \quad s_{t,i}^{\text{obj}} = (l_t^g)^T e_i^g. \quad (5)$$

For the traditional grid attention, we use the image region feature h_j, which is one column in the previous hidden layer $h \in \mathbb{R}^{D^{\text{pat}} \times N^{\text{pat}}}$, to query the pre-trained bi-LSTM word vectors e. For the new object-driven attention, we use the GloVe embedding of object class label l_t^g to query the GloVe embedding of the words in the sentence, as illustrated in the lower part of Fig. 4.

Feature map concatenation. The patch-wise context vector c_j^{pat} can be directly concatenated with the image feature vector h_j in the previous layer. However, the object-wise context vector c_t^{obj} cannot, because they are associated with bounding boxes instead of pixels in the hidden feature map. We propose to copy the object-wise context vector c_t^{obj} to every pixel where the t'th object is present, i.e., $M_t \otimes c_t^{\text{obj}}$ where $\otimes$ is the vector outer-product, as illustrated in the upper-right part of Fig. 4. [1]

If there are multiple bounding boxes covering the same pixel, we have to decide whose context vector should be used on this pixel. In this case, we simply do a max-pooling across all the bounding boxes:

$$c^{\text{obj}} = \max_{t\,:\,1 \le t \le T} M_t \otimes c_t^{\text{obj}}. \quad (6)$$

Then c^{obj} can be concatenated with the feature map h and patch-wise context vectors c^{pat} for next-stage generation.

Label context vectors. Similarly, we distribute the class label information to the entire hidden feature map to get the label context vectors, i.e.,

$$c^{\text{lab}} = \max_{t\,:\,1 \le t \le T} M_t \otimes e_t^g. \quad (7)$$

Finally, we concatenate h, c^{pat}, c^{obj} and c^{lab} and pass the

---
[1]This operation can be viewed as an inverse of the pooling operator.

concatenated tensor through one up-sampling layer and several residual layers to generate a higher-resolution image.

Grid attention vs. object-driven attention. The process to compute the patch-wise context vectors above is the traditional grid attention mechanism used in AttnGAN [29]. Note that its attention weights $\beta_{j,i}^{\text{pat}}$ and context vector c_j^{pat} are useful only when the hidden feature h_j^{pat} in the G_0 stage correctly captures the content to be drawn in patch j. This essentially assumes that the generation in the G_0 stage already captures a rough sketch (semantic layout). This assumption is valid for simple datasets like birds [29], but fails for complex datasets like COCO [15] where the generated low-resolution image $\widehat{x}_0$ typically does *not* have a meaningful layout. In this case, the grid attention is even harmful, because patch-wise context vector is attended to a wrong word and thus generate the texture associated with that wrong word. This may be the reason why AttnGAN's generated image contains scattered patches of realistic texture but overall is semantically not meaningful; see Fig. 1 for example. Similar phenomenon is also observed in Deep-Dream [20]. On the contrary, in our object-driven attention, the attention weights $\beta_{t,i}^{\text{obj}}$ and context vector c_t^{obj} rely on the class label l_t^g of the bounding box and are independent of the generation in the G_0 stage. Therefore, the object-wise context vectors are always helpful to generate images that are consistent with the pre-generated semantic layout. Another benefit of this design is that the context vector c_t^{obj} can also be used in the discriminator, as we present in § 3.2.2.

3.2.2 Discriminators

We design patch-wise and object-wise discriminators to train the attentive multi-stage generator above. Given a patch from uniformly-partitioned image patches determined by the uniform down-sampling structure of CNN, the patch-wise discriminator is trying to determine whether this patch is realistic or not (unconditional) and whether this patch is consistent with the sentence description or not (conditional). Given a bounding box and the class label of the object within it, the object-wise discriminator is trying to determine whether this region is realistic or not (unconditional) and whether this region is consistent with the sentence description and given class label or not (conditional).

Patch-wise discriminators. Given an image-sentence pair $x, \bar{e}$ ($\bar{e}$ is the sentence vector), the patch-wise unconditional and text discriminator can be written as

$$p^{\text{pat,un}} = D_{\text{uncond.}}^{\text{pat}}(\text{Enc}(x)), \quad p^{\text{pat,con}} = D_{\text{text}}^{\text{pat}}(\text{Enc}(x), \bar{e}), \tag{8}$$

where Enc is a convolutional feature extractor that extracts patch-wise features, $D_{\text{uncond.}}$ ($D_{\text{text}}^{\text{pat}}$) determine whether the patch is realistic (consistent with the text description) or not.

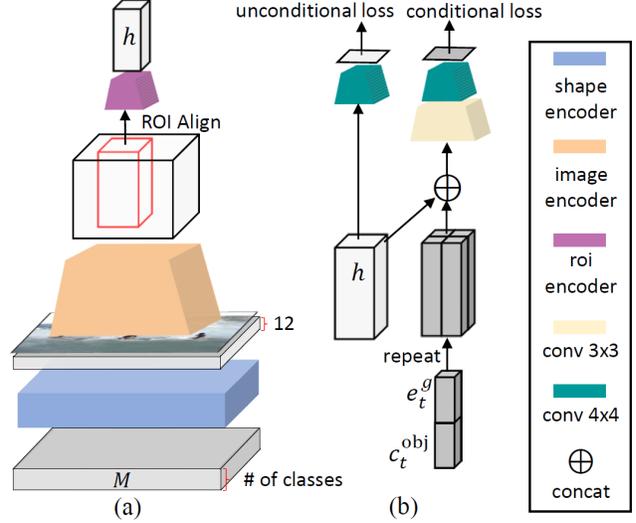**Shape discriminator.** In a similar manner, we have our patch-wise shape discriminator

Figure 5: Object-wise discriminator.

$$p^{\text{pix}} = D^{\text{pix}}(\text{Enc}(x, M)), \tag{9}$$

where we first concatenate the image x and shapes M in the channel dimension, and then extracts patch-wise features by another convolutional feature extractor Enc. The probabilities p^{pix} determine whether the patch is consistent with the given shape. Our patch-wise discriminators $D_{\text{uncond.}}^{\text{pat}}$, $D_{\text{text}}^{\text{pat}}$ and D^{pix} resembles the PatchGAN [11] for the image-to-image translation task. Compared with the global discriminators in AttnGAN [29], the patch-wise discriminators not only reduce the model size and thus enable generating higher resolution images, but also increase the quality of generated images; see Table 1 for experimental evidence.

Object-wise discriminators. Given an image x, bounding boxes of objects $B_{1:T}$ and their shapes M, we propose the following object-wise discriminators:

$$\{h_t^{\text{obj}}\}_{t=1}^T = \text{FastRCNN}(x, M, B_{1:T}),$$
$$p_t^{\text{obj,un}} = D_{\text{uncond.}}^{\text{obj}}(h_t^{\text{obj}}), \quad p_t^{\text{obj,con}} = D^{\text{obj}}(h_t^{\text{obj}}, e_t^g, c_t^{\text{obj}}). \tag{10}$$

Here, we first concatenate the image x and shapes M and extract a region feature vector h_t^{obj} for each bounding box through a Fast R-CNN model [4] with an ROI-align layer [7]; see Fig. 5(a). Then similar to the patch-wise discriminator (8), the unconditional (conditional) probabilities $p_t^{\text{obj,un}}$ ($p_t^{\text{obj,con}}$) determine whether the t'th object is realistic (consistent with its class label e_t^g and its text context information c_t^{obj}) or not; see Fig. 5(b). Here, e_t^g is the GloVe embedding of the class label and c_t^{obj} is its text context information defined in (3).

All discriminators are trained by the traditional cross entropy loss [5].

3.2.3 Loss function for the image generator

The generator's GAN loss is a weighted sum of these discriminators' loss, *i.e.*,

$$\mathcal{L}_{\text{GAN}}(G) = -\frac{\lambda_{\text{obj}}}{T} \sum_{t=1}^{T} \left(\underbrace{\log p_t^{\text{obj,un}}}_{\text{obj uncond. loss}} + \underbrace{\log p_t^{\text{obj,con}}}_{\text{obj cond. loss}} \right)$$

$$-\frac{1}{N^{\text{pat}}} \sum_{j=1}^{N^{\text{pat}}} \left(\underbrace{\log p_j^{\text{pat,un}}}_{\text{uncond. loss}} + \lambda_{\text{txt}} \underbrace{\log p_j^{\text{pat,con}}}_{\text{text cond. loss}} + \lambda_{\text{pix}} \underbrace{\log p_j^{\text{pix}}}_{\text{shape cond. loss}} \right).$$

Here, T is the number of bounding boxes, N^{pat} is the number of regular patches, $(\lambda_{\text{obj}}, \lambda_{\text{txt}}, \lambda_{\text{pix}})$ are the weights of the object-wise GAN loss, patch-wise text conditional loss and patch-wise shape conditional loss, respectively. We tried combining our discriminators with the spectral normalized projection discriminator [18, 19], but did not see significant performance improvement. We report performance of the spectral normalized version in § 4.1 and provide model architecture details in § A.

Combined with the deep multi-modal attentive similarity model (DAMSM) loss introduced in [29], our final image generator's loss is

$$\mathcal{L}_G = \mathcal{L}_{\text{GAN}} + \lambda_{\text{DAMSM}} \mathcal{L}_{\text{DAMSM}} \tag{11}$$

where λ_{damsm} is a hyper-parameter to be tuned. Here, the DAMSM loss is a word level fine-grained image-text matching loss computed, which will be elaborated in § A. Based on the experiments on a held-out validation set, we set the hyperparameters in this section as: $\lambda_{\text{obj}} = 0.1$, $\lambda_{\text{txt}} = 0.1$, $\lambda_{\text{pix}} = 1$ and $\lambda_{\text{damsm}} = 100$.

Remark 3.1. *Both the patch-wise and object-wise discriminators can be applied to different stages in the generation. We apply the patch-wise discriminator for every stage of the generation, following [33, 11], but only apply the object-wise discriminator at the final stage.*

4. Experiments

Dataset. We use the COCO dataset [15] for evaluation. It contains 80 object classes, where each image is associated with object-wise annotations (*i.e.*, bounding boxes and shapes) and 5 text descriptions. We use the official 2014 train (over 80K images) and validation (over 40K images) splits for training and test stages, respectively.

Evaluation metrics. We use the Inception score [25] and *Fréchet inception distance* (FID) [8] score as the quantitative evaluation metrics. In our experiments, we found that Inception score can be saturated, even over-fitted, while FID is a more robust measure and aligns better with human qualitative evaluation. Following [29], we also use R-precision,

Table 1: The quantitative experiments. Methods marked with 0, 1 and 2 respectively represent experiments using the predicted boxes and shapes, the ground-truth boxes and predicted shapes, and the ground-truth boxes and shapes. We use **bold**, ∗, and ∗∗ to highlight the best performance under these three settings, respectively. The results of methods marked with † are those reported in the original papers. ↑ (↓) means the higher (lower), the better.

Methods	Inception ↑	FID ↓	R-prcn (%) ↑
Obj-GAN[0]	**27.37 ± 0.22**	**25.85**	86.20 ± 2.98
Obj-GAN[1]	27.96 ± 0.39∗	24.19∗	88.36 ± 2.82
Obj-GAN[2]	29.89 ± 0.22∗∗	20.75∗∗	89.59 ± 2.67
P-AttnGAN w/ Lyt[0]	18.84 ± 0.29	59.02	65.71 ± 3.74
P-AttnGAN w/ Lyt[1]	19.32 ± 0.29	54.96	68.40 ± 3.79
P-AttnGAN w/ Lyt[2]	20.81 ± 0.16	48.47	70.94 ± 3.70
P-AttnGAN	26.31 ± 0.43	41.51	86.71 ± 2.97
Obj-GAN w/ SN[0]	26.97 ± 0.31	29.07	**86.84 ± 2.82**
Obj-GAN w/ SN[1]	27.41 ± 0.17	27.26	88.70 ± 2.65∗
Obj-GAN w/ SN[2]	28.75 ± 0.32	23.37	89.97 ± 2.56∗∗
Reed *et al.* [23]†	7.88 ± 0.07	n/a	n/a
StackGAN [32]†	8.45 ± 0.03	n/a	n/a
AttnGAN [29]	23.79 ± 0.32	28.76	82.98 ± 3.15
vmGAN [35]†	9.94 ± 0.12	n/a	n/a
Sg2Im [12]†	6.7 ± 0.1	n/a	n/a
Infer [9][0]†	11.46 ± 0.09	n/a	n/a
Infer [9][1]†	11.94 ± 0.09	n/a	n/a
Infer [9][2]†	12.40 ± 0.08	n/a	n/a
Obj-GAN-SOTA[0]	30.29 ± 0.33	25.64	91.05 ± 2.34
Obj-GAN-SOTA[1]	30.91 ± 0.29	24.28	92.54 ± 2.16
Obj-GAN-SOTA[2]	32.79 ± 0.21	21.21	93.39 ± 2.08

a common evaluation metric for ranking retrieval results, to evaluate whether the generated image is well conditioned on the given text description. More specifically, given a pre-trained image-to-text retrieval model, we use generated images to query their corresponding text descriptions. First, given generated image $\widehat{x}$ conditioned on sentence s and 99 random sampled sentences $\{s'_i : 1 \leq i \leq 99\}$, we rank these 100 sentences by the pre-trained image-to-text retrieval model. If the ground truth sentence s is ranked highest, we count this a success retrieval. For all the images in the test dataset, we perform this retrieval task once and finally count the percentage of success retrievals as the R-precision score.

It is important to point out that none of these quantitative metrics are perfect. Better metrics are required to evaluate image generation qualities in complicated scenes. In fact, the Inception score completely fails in evaluating the semantic layout of the generated images. The R-precision score depends on the pre-trained image-to-text retrieval model it uses, and can only capture the aspects that the retrieval model is able to capture. The pre-trained model we use is still limited in capturing the relations between objects in complicated scenes, so is our R-precision score.

Quantitative evaluation. We compute these three metrics under two settings for the full validation dataset.

Qualitative evaluation. Apart from the quantitative evaluation, we also visualize the outputs of all ablative versions of Obj-GAN and the state-of-the-art methods (*i.e.*, [29]) whose pre-trained models are publicly available.

Figure 6: The overall qualitative comparison. All images are generated without the usage of any ground-truth information.

4.1. Ablation study

In this section, we first evaluate the effectiveness of the object-driven attention. Next, we compare the object-driven attention mechanism with the grid attention mechanism. Then, we evaluate the impact of the spectral normalization for Obj-GAN. We use Fig. 6 and the higher half of Table 1 to present the comparison among different ablative versions of Obj-GAN. Note that all ablative versions have been trained with batch size 16 for 60 epochs. In addition, we use the lower half of Table 1 to show the comparison between Obj-GAN and previous methods. Finally, we validated the Obj-GAN's generalization ability on the novel text descriptions. **Object-driven attention.** To evaluate the efficacy of the object-driven attention mechanism, we implement a baseline, named P-AttnGAN w/ Lyt, by disabling the object-driven attention mechanism in Obj-GAN. In essence, P-AttnGAN w/ Lyt can be considered as an improved version of AttnGAN with the patch-wise discriminator (abbreviated as the prefix "P-" in name) and the modules (*e.g.*, shape discriminator) for handling the conditional layout (abbreviated as "Lyt"). Moreover, it can also be considered as a modified implementation of [9], which resembles their two-step (layout-image) generation. Note that there are three key differences between P-AttnGAN w/ Lyt and [9]: (i) P-AttnGAN w/ Lyt has a multi-stage image generator that gradually increases the generated resolution and refines the generated images, while [9] has a single-stage image generator. (ii) With the help of the grid attentive module, P-AttnGAN w/ Lyt is able to utilize the fine-grained word-level information, while [9] conditions on the global sentence information. (iii) The third difference lies in their loss

functions: P-AttnGAN w/ Lyt uses the DAMSM loss in (11) to penalize the mismatch between the generated images and the input text descriptions, while [9] uses the perceptual loss to penalize the mismatch between the generated images and the ground-truth images. As shown in Table 1, P-AttnGAN w/ Lyt yields higher Inception score than [9] does.

We compare Obj-GAN with P-AttnGAN w/ Lyt under three settings, with each corresponding to a set of conditional layout input, *i.e.*, the predicted boxes & shapes, the ground-truth boxes & predicted boxes, and the ground-truth boxes & shapes. As presented in Table 1, Obj-GAN consistently outperforms P-AttnGAN w/ Lyt on all three metrics. In Fig. 7, we use the same layout as the conditional input, and compare the visual quality of their generated images. An interesting phenomenon shown in Fig. 7 is that both the foreground objects (*e.g.*, airplane and train) and the background (*e.g.*, airport and trees) textures synthesized by Obj-GAN are much richer and smoother than those using P-AttnGAN w/ Lyt. The effectiveness of the object-driven attention for the foreground objects is easy to understand. The benefits for the background textures using the object-driven attention mechanism is probably due to the fact that it implicitly provides stronger signal that distinguishes the foreground. As such, the image generator may have richer guidance and clearer emphasis when synthesizing textures for a certain region.

Grid attention vs. object-driven attention. We compare Obj-GAN with P-AttnGAN herein, so as to compare the effects of the object-driven and the grid attention mechanisms. In Fig. 8, we show the generated image of each method as well as the corresponding attention maps aligned

Figure 7: Qualitative comparison with P-AttnGAN w/ Lyt.

Figure 8: Qualitative comparison with P-AttnGAN. The attention maps of each method are shown beside the generated image.

on the right side. In a grid attention map, the brightness of a region reflects how much this region attended to the word above the map. As for the object-driven attention map, the word above each attention map is the most attended word by the highlighted object. The highlighted region of an object-driven attention map is the object shape.

As analyzed in § 3.2.1, the reliability of grid attention weights depends on the quality of the previous layer's image region features. This makes the grid attention unreliable sometimes, especially for complex scenes. For example, the grid attention weights in Fig. 8 are unreliable because they are scattered (*e.g.*, the attention map for "man") and inaccurate. However, this is not a problem for the object-driven attention mechanism, because its attention weights are directly calculated from embedding vectors of words and class labels and are independent of image features. Moreover, as shown in Fig. 4 and Equ. (6), the impact region of the object-driven attention context vector is bounded by the object shapes, which further enhances its semantics meaningfulness. As a result, the instance-driven attention significantly improves the visual quality of the generated images, as demonstrated in Fig. 8. Moreover, the performance can be further improved if the semantic layout generation is improved. In the extreme case, Obj-GAN based on ground truth layout (Obj-GAN^2) has the best visual quality (the rightmost column of Fig. 8) and the best quantitative evaluation (Table 1).

Obj-GAN w/ SN vs. Obj-GAN. We present the comparison between the cases with or without spectral normalization in the discriminators in Table 1 and Fig. 6. We observe that there is no obvious improvement on the visual quality, but slightly worse on the quantitative metrics. We show

Figure 9: Generated images for novel descriptions.

more results and discussions in § A.

Comparison with previous methods. To compare Obj-GAN with the previous methods, initialized by the Obj-GAN models in the ablation study, we trained Obj-GAN-SOTA with batch size 64 for 10 more epochs. In order to evaluate AttnGAN on FID, we conducted the evaluation on the officially released pre-trained model. Note that the Sg2Im [12] focuses on generating images from scene graphs and conducted the evaluation on a different split of COCO. However, we still included Sg2Im's results to reflect the broader context of the related topic. As shown in Table 1, Obj-GAN-SOTA outperforms all previous methods significantly. We notice that the increment of batch size does boost the Inception score and R-precision, but does not improve FID. The possible explanation is: with a larger batch size, the DAMSM loss (a ranking loss in essence) in (11) plays a more important role and improves Inception and R-precision, but it does not focus on reducing FID between the generated images and the real ones.

Generalization ability. We further investigate if Obj-GAN just memorizes the scenarios in COCO or it indeed learns the relations between the objects and their surroundings. To this end, we compose several descriptions which reflect novel scenarios that are unlikely to happen in the real-world, *e.g.*, *a decker bus is floating on top of a lake*, or *a cat is catching a frisbee*. We use Obj-GAN to synthesize images for these rare scenes. The results in Fig. 9 further demonstrate the good generalization ability of Obj-GAN.

5. Conclusions

In this paper, we have presented a multi-stage Object-driven Attentive Generative Adversarial Networks (Obj-GANs) for synthesizing images with complex scenes from the text descriptions. With a novel object-driven attention layer at each stage, our generators are able to utilize the fine-grained word/object-level information to gradually refine the synthesized image. We also proposed the Fast R-CNN based object-wise discriminators, each of which is paired with a conditional input of the generator and provides object-wise discrimination signal for that condition. Our Obj-GAN significantly outperforms previous state-of-the-art GAN models on various metrics on the large-scale challenging COCO benchmark. Extensive experiments demonstrate the effectiveness and generalization ability of Obj-GAN on text-to-image generation for complex scenes.

References

[1] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and vqa. *CVPR*, 2018.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

[3] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[4] R. B. Girshick. Fast R-CNN. In *ICCV*, 2015.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[6] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, 2015.

[7] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, 2017.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a nash equilibrium. *NIPS*, 2017.

[9] S. Hong, D. Yang, J. Choi, and H. Lee. Inferring semantic layout for hierarchical text-to-image synthesis. *CVPR*, 2018.

[10] Q. Huang, P. Zhang, D. O. Wu, and L. Zhang. Turbo learning for captionbot and drawingbot. In *NeurIPS*, 2018.

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[12] J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018.

[13] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[14] K. Lee, X. Chen, G. Hua, H. Hu, and X. He. Stacked cross attention for image-text matching. *ECCV*, 2018.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[16] S. Ma, J. Fu, C. W. Chen, and T. Mei. DA-GAN: Instance-level image translation by deep attention generative adversarial networks. In *CVPR*, 2018.

[17] E. Mansimov, E. Parisotto, L. J. Ba, and R. Salakhutdinov. Generating images from captions with attention. In *ICLR*, 2016.

[18] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018.

[19] T. Miyato and M. Koyama. cgans with projection discriminator. *ICLR*, 2018.

[20] A. Mordvintsev, C. Olah, and M. Tyka. Deep dream, 2015, 2017.

[21] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[22] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *NIPS*, 2016.

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

[24] S. E. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. de Freitas. Parallel multiscale autoregressive density estimation. In *ICML*, 2017.

[25] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.

[26] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *NIPS*, 2017.

[28] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[29] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *CVPR*, 2018.

[30] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016.

[31] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[32] 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.

[33] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *TPAMI*, 2018.

[34] S. Zhang, H. Dong, W. Hu, Y. Guo, C. Wu, D. Xie, and F. Wu. Text-to-image synthesis via visual-memory creative adversarial network. In *PCM*, 2018.

[35] S. Zhang, H. Dong, W. Hu, Y. Guo, C. Wu, D. Xie, and F. Wu. Text-to-image synthesis via visual-memory creative adversarial network. In *PCM*, 2018.

A. Appendix

A.1. Obj-GAN vs. the ablative versions

In this section, we show more images generated by our Obj-GAN and its ablative versions on the COCO dataset. In Fig. 10 and Fig. 11, we provide more comparisons as the complementary for Fig. 6. It can be found that there are no obvious improvement on the visual quality when using the spectral normalization.

Figure 10: The overall qualitative comparison.

A.2. Visualization of attention maps

In Fig. 12, we visualize attention maps generated by P-AttnGAN and Obj-GAN as the complementary for Fig. 8.

A.3. Results based on the ground-truth layout

We show the results generated by Obj-GAN based on the ground-truth layout in Fig. 13, Fig. 14 and Fig. 15.

Real Image	P-AttnGAN	P-AttnGAN w/ Lyt	Obj-GAN w/ SN	Obj-GAN	Real Image	P-AttnGAN	P-AttnGAN w/ Lyt	Obj-GAN w/ SN	Obj-GAN

A polo player on a horse following the ball on the ground.

Pizza with tomato and green olives being cut by a big knife.

A parking meter that reads "fail" and a white car across from it.

A cat is lying on top of a suitcase.

A group of scattered sheep move down a country path.

A slice of casserole with steamed broccoli on a plate.

A couple of people in a field flying kites.

Skier in mid jump on mountain with poles extended.

A train traveling down the rail road tracks.

A small two toned blue airplane flying.

The bed and desk in a modern hotel room.

A cat plays with grass outside in the garden.

A kitchen area with a sink, refrigerator, and stove.

Boys are playing a soccer game on a field.

A neatly made bed with a red sheet.

A large air plane on a run way.

Closeup of a cat in a bathroom sink.

A park bench on a grassy green hillside.

A living room has a beige couch, and carpet.

A coupe of people stand on a rocky hill top.

Figure 11: The overall qualitative comparison.

P-AttnGAN	Obj-GAN⁰	Obj-GAN²

A field of cows grazing near a barn.

There is casserole dish filled with leftover vegetables

This young child is running in a field with a kite.

Hundreds of sheep walking in the water and a ranch.

A man riding a wave on a surfboard.

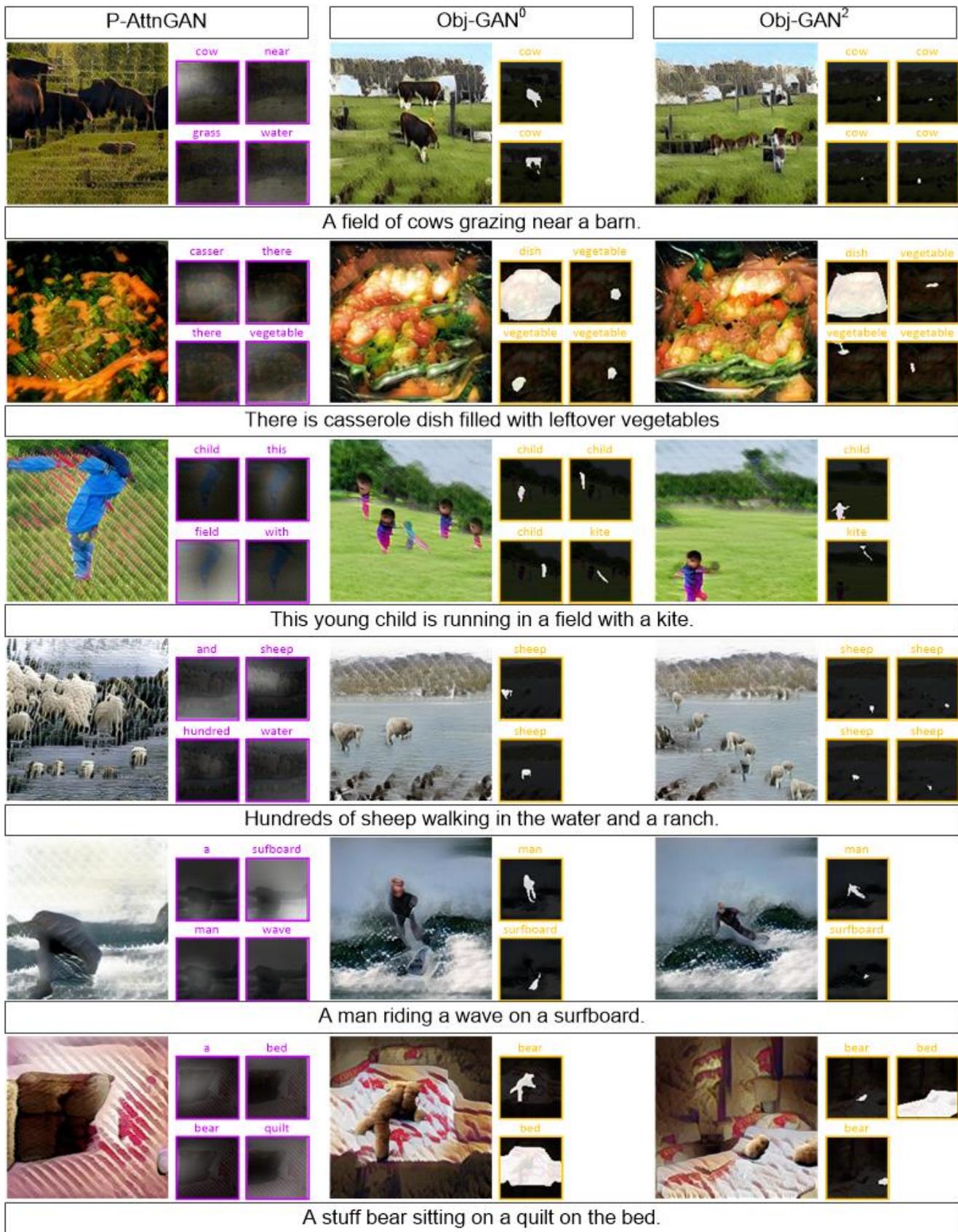A stuff bear sitting on a quilt on the bed.

Figure 12: Qualitative comparison with P-AttnGAN. The attention maps of each method are shown beside the generated image.

(1) A glass table with a bottle and glass of wine next to a chair.

(2) A train sitting on some tracks next to a sidewalk.

(3) Soccer player wearing green and orange hitting soccer ball.

(4) A kitchen with a very messy counter space.

(5) The people are on the beach getting ready to surf.

(6) A jet airliner waits its turn on the runway.

(7) Two cows are grazing in a dirt field.

(8) A small lightweight airplane flying through the sky.

(9) A cow running in a field next to a dog.

(10) Two people go into the water with their surfboards.

(11) A man in a helmet jumps a snowboard.

(12) A giraffe is standing all alone in a grassy area.

(13) The black dog is staring at the cat.

(14) A bunch of sheep are standing in a field.

(15) A bench sitting on top of a lush green hillside.

(16) A polar bear playing in the water at a wild life enclosure.

Figure 13: Results based on the ground-truth layout.

(1) A man on a soccer field next to a ball.

(2) A dog sitting on a bench in front of a garden.

(3) A black cat drinking water out of a water faucet.

(4) A cat laying on a TV in the middle of the room.

(5) Four people on skis below a mountain taking a picture.

(6) A man in outdoor winter clothes holds a snowboard.

(7) A orange before and after it was cu.

(8) A dog running with a frisbee in its mouth.

(9) A woman and a dog tussle over a frisbee.

(10) Man in a wetsuit on top of a blue and white surfboard.

(11) A white ship sails in the blue ocean water.

(12) A couple of men standing next to dogs near water.

(13) A man on a motorcycle in a carport.

(14) A group of people riding horses on a beach.

(15) A hipster wearing flood pants poses with his skateboard.

(16) A black dog holding a frisbee in its mouth.

Figure 14: Results based on the ground-truth layout.

(1) A big boat on the water near the shore.

(2) All the horses in the pen are grazing.

(3) A man riding a bike down the middle of a street.

(4) A bathroom with a sink and a toilet.

(5) A yellow school bus parked near a tree.

(6) A group of cows graze on some grass.

(7) A ship is sailing across an ocean filled with waves.

(8) Three skiers posing for a picture on the slope.

(9) A large green bus approaching a bus stop.

(10) A close view of a pizza, and a mug of beer.

(11) A cat is looking at a television displaying a dog in a cage.

(12) Three white sinks in a bathroom under mirrors.

(13) Three cranes standing on one leg in the water.

(14) A bear lying on a rock in its den, looking upward.

(15) Two bottles of soda sit near a sandwich.

(16) Someone on a snowboard coming to a stop.

Figure 15: Results based on the ground-truth layout.

A.4. Bi-LSTM text encoder, DAMSM and R-precision

We use the deep attentive multi-modal similarity model (DAMSM) proposed in [29], which learns a joint embedding of the image regions and words of a sentence in a common semantic space. The fine-grained conditional loss enforces the sub-region of the generated image to match the corresponding word in the sentence.

Bi-LSTM text encoder serves as the text encoder for both DAMSM and the box generator (see § A.5). Bi-LSTM text encoder is a bi-directional LSTM that extracts semantic vectors from the text description. In the Bi-LSTM, each word corresponds to two hidden states, one for each direction. Thus, we concatenate its two hidden states to represent the semantic meaning of a word. The feature matrix of all words is indicated by $\dot{e} \in \mathbb{R}^{D \times T_s}$. Its ith column $\dot{e}_i$ is the feature vector for the ith word. D is the dimension of the word vector and T_s is the number of words. Meanwhile, the last hidden states of the bi-directional LSTM are concatenated to be the global sentence vector, denoted by $\hat{e} \in \mathbb{R}^D$. We present the network architectures for the Bi-LSTM text encoder in Table 2.

The image encoder is a convolutional neural network that maps images to semantic vectors. The intermediate layers of the CNN model learns local features of different regions of the image, while the later layers learn global features of the image. More specifically, the image encoder is built upon Inception-v3 model pre-trained on ImageNet. We first rescale the input image to be 299×299 pixels. And then, we extract the local feature matrix $f \in \mathbb{R}^{768 \times 289}$ (reshaped from 768×17×17) from "$mixed\_6e$" layer of Inception-v3. Each column of f is the feature vector of a local image region. 768 is the dimension of the local feature vector, and 289 is the number of regions in the image. Meanwhile, the global feature vector $\overline{f} \in \mathbb{R}^{2048}$ is extracted from the last average pooling layer of Inception-v3. Finally, we convert the image features to the common semantic space of text features by adding a new layer perceptron as shown in Eq. (12),

$$v = Wf; \quad \overline{v} = \overline{W}\,\overline{f}, \tag{12}$$

where $v \in \mathbb{R}^{D \times 289}$ and its ith column v_i is the visual feature vector for the ith image region; $\overline{v} \in \mathbb{R}^D$ is the visual feature vector for the whole image. While v_i is the local image feature vector that corresponds to the word embedding, $\overline{v}$ is the global feature vector that is related to the sentence embedding. D is the dimension of the multimodal (*i.e.*, image and text modalities) feature space. For efficiency, all parameters in layers built from Inception-v3 model are fixed, and the parameters in newly added layers are jointly learned with the rest of networks.

The fine-grained conditional loss is designed to learn the correspondence between image regions and words.

However, it is difficult to obtain manual annotations. Actually, many words relate to concepts that may not easily be visually defined, such as *open* or *old*. One possible solution is to learn word-image correspondence in a semi-supervised manner, in which the only supervision is the correspondence between the entire image and the whole text description (a sequence of words).

We can first calculate the similarity matrix between all possible pairs of word and image region by Eq. (13),

$$s = \dot{e}^T v, \tag{13}$$

where $s \in \mathbb{R}^{T \times 289}$ and $s_{i,j}$ means the similarity between the ith word and the jth image region.

Generally, a sub-region of the image is described by none or several words of the text description, and it is not likely to be described by the whole sentence. Therefore, we normalize the similarity matrix by Eq. (14),

$$\overline{s}_{i,j} = \frac{\exp(s_{i,j})}{\sum_{k=0}^{T-1} \exp(s_{k,j})} \tag{14}$$

Second, we build an attention model to compute a context vector for each word (query). The context vector c_i is a dynamic representation of image regions related to the ith word of the text description. It is computed as the weighted sum over all visual feature vectors,

$$c_i = \sum_{j=0}^{288} \alpha_j v_j, \tag{15}$$

where we define the weight α_j via Eq. (16),

$$\alpha_j = \frac{\exp(\gamma_1 \overline{s}_{i,j})}{\sum_{k=0}^{288} \exp(\gamma_1 \overline{s}_{i,k})} \tag{16}$$

Here, γ_1 is a factor that decides how much more attention is paid to features of its relevant regions when computing the context vector for a word.

Finally, we define the relevance between the ith word and the image using the cosine similarity between c_i and $\dot{e}_i$, *i.e.*, $R(c_i, \dot{e}_i) = (c_i^T \dot{e}_i)/(||c_i||||\dot{e}_i||)$. The relevance between the entire image (Q) and the whole text description (U) is computed by Eq. (17),

$$R(Q, U) = \log\left(\sum_{i=1}^{T-1} \exp(\gamma_2 R(c_i, \dot{e}_i)) \right)^{\frac{1}{\gamma_2}}, \tag{17}$$

where γ_2 is a factor that determines how much to magnify the importance of the most relevant word-image pair. When $\gamma_2 \to \infty$, $R(Q, U)$ approximates to $\max_{i=1}^{T-1} R(c_i, \dot{e}_i)$.

For a text-image pair, we can compute the posterior probability of the text description (U) being matching with the image (Q) via,

$$P(U|Q) = \frac{\exp(\gamma_3 R(Q, U))}{\sum_{U' \in \mathbb{U}} \exp(\gamma_3 R(Q, U'))}, \tag{18}$$

where γ_3 is a smoothing factor determined by experiments. $\mathbb{U}$ denotes a minibatch of M text descriptions, in which only one description U^+ matches the image Q. Thus, for each image, there are $M - 1$ mismatching text descriptions. The objective function is to learn the model parameters Λ by minimizing the negative log posterior probability that the images are matched with their corresponding text descriptions (ground truth),

$$\mathcal{L}_1^w(\Lambda) = -\log \prod_{Q \in \mathbb{Q}} P(U^+|Q), \qquad (19)$$

where 'w' stands for "word".

Symmetrically, we can compute,

$$\mathcal{L}_2^w(\Lambda) = -\log \prod_{U \in \mathbb{U}} P(Q^+|U), \qquad (20)$$

where $P(Q|U) = \frac{\exp(\gamma_3 R(Q,U))}{\sum_{Q' \in \mathbb{Q}} \exp(\gamma_3 R(Q',U))}$.

If we redefine Eq. (17) by $R(Q, U) = \left(\bar{v}^T \hat{e}\right)/\left(||\bar{v}|| ||\hat{e}||\right)$ and substitute it to Eq. (18), Eq. (19), Eq. (20), we can obtain loss functions $\mathcal{L}_1^s$ and $\mathcal{L}_2^s$ (where 's' stands for "sentence") using the sentence embedding $\hat{e}$ and the global visual vector $\bar{v}$.

The fine-grained conditional loss is defined via Eq. (21),

$$\mathcal{L}_{DAMSM} = \mathcal{L}_1^w + \mathcal{L}_2^w + \mathcal{L}_1^s + \mathcal{L}_2^s \qquad (21)$$

The DAMSM is pre-trained by minimizing $\mathcal{L}_{DAMSM}$ using real image-text pairs. Since the size of images for pre-training DAMSM is not limited by the size of images that can be generated, real images of size 299×299 are utilized. Furthermore, the pre-trained DAMSM can provide visually-discriminative word features and a stable fine-grained conditional loss for the attention generative network.

The R-precision score. The DAMSM model is also used to compute the R-precision score. If there are R relevant documents for a query, we examine the top R ranked retrieval results of a system, and find that r are relevant, and then by definition, the R-precision (and also the precision and recall) is r/R. More specifically, we use generated images to query their corresponding text descriptions. First, the image encoder and Bi-LSTM text encoder learned in our pre-trained DAMSM are utilized to extract features of the generated images and the given text descriptions. Then, we compute cosine similarities between the image features and the text features. Finally, we rank candidates text descriptions for each image in descending similarity and find the top r relevant descriptions for computing the R-precision.

A.5. Network architectures for semantic layout generation

Box generator. We design our box generator by improving the one in [9] to be attentive. We denote the bounding box of the t-th object as $B_t = (b_t^x, b_t^y, b_t^w, b_t^h, \boldsymbol{l}_t)$. Then, we formulate the joint probability of sampling B_t from the box generator as

$$p(b_t^x, b_t^y, b_t^w, b_t^h, \boldsymbol{l}_t) = p(\boldsymbol{l}_t)p(b_t^x, b_t^y, b_t^w, b_t^h|\boldsymbol{l}_t). \qquad (22)$$

We implement $p(\boldsymbol{l}_t)$ as a categorical distribution, and implement $p(b_t^x, b_t^y, b_t^w, b_t^h|\boldsymbol{l}_t)$ as a mixture of quadravariate Gaussians. As described in [9], in order to reduce the parameter space, we decompose the box coordinate probability as $p(b_t^x, b_t^y, b_t^w, b_t^h|\boldsymbol{l}_t) = p(b_t^x, b_t^y|\boldsymbol{l}_t)p(b_t^w, b_t^h|b_t^x, b_t^y, \boldsymbol{l}_t)$, and approximate it with two bivariate Gasussian mixtures by

$$p(b_t^x, b_t^y|\boldsymbol{l}_t) = \sum_{k=1}^{K} \pi_{t,k}^{xy} \mathcal{N}(b_t^x, b_t^y; \boldsymbol{\mu}_{t,k}^{xy}, \Sigma_{t,k}^{xy}),$$

$$p(b_t^w, b_t^h|b_t^x, b_t^y, \boldsymbol{l}_t) = \sum_{k=1}^{K} \pi_{t,k}^{wh} \mathcal{N}(b_t^w, b_t^h; \boldsymbol{\mu}_{t,k}^{wh}, \Sigma_{t,k}^{wh}).$$

$$\qquad (23)$$

In practice, as in [9], we implement the box generator within a encoder-decoder framework. The encoder is the Bi-LSTM text encoder as mentioned in § A.4. The Gaussian Mixture Model (GMM) parameters for Eq. (22) are obtained from the decoder LSTM outputs. Given text encoder's final hidden state $h_{T_s}^{\text{Enc}} \in \mathbb{R}^D$ and output $H^{\text{Enc}} \in \mathbb{R}^{T_s \times D}$, we initialize the decoder's initial hidden state h_0 with $h_{T_s}^{\text{Enc}}$. As for H^{Enc}, we use it to compute the contextual input z_t for the decoder:

$$z_t = \sum_{i=1}^{T_s} \alpha_i h_i^{\text{Enc}}, \text{ with } \alpha_i = W_v \cdot (W_\alpha[h_{t-1}, h_i^{\text{Enc}}]),$$

$$\qquad (24)$$

where W_v is a learnable parameter, W_α is the parameter of a linear transformation, and $\cdot$ and $[\cdot, \cdot]$ represent the dot product and concatenation operation, respectively.

Then, the calculation of GMM parameters are shown as follows:

$$[h_t, c_t] = \text{LSTM}([B_{t-1}, z_t]; [h_{t-1}, c_{t-1}]), \qquad (25)$$

$$\boldsymbol{l}_t = W^l h_t + \mathbf{b}^l, \qquad (26)$$

$$\boldsymbol{\theta}_t^{xy} = W^{xy}[h_t, \boldsymbol{l}_t] + \mathbf{b}^{xy}, \qquad (27)$$

$$\boldsymbol{\theta}_t^{wh} = W^{wh}[h_t, \boldsymbol{l}_t, b_x, b_y] + \mathbf{b}^{wh}, \qquad (28)$$

where $\boldsymbol{\theta}_t = [\boldsymbol{\pi}_{t,1:K}, \boldsymbol{\mu}_{t,1:K}, \boldsymbol{\Sigma}_{t,1:K}]$ are the parameters for GMM concatenated to a vector. We use the same Adam optimizer and training hyperparameters (*i.e.*, learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$) as in [9].

Shape generator. We implement the shape generator in [9] with almost the same architecture except the upsample block. In [9], the upsample block is designed as [convtranspose 4×4 (pad 1, stride 2) - Instance Normalization

- ReLU]. We discovered that the usage of convtranspose would lead to unstable training which is reflected by the frequent severe grid artifacts. To this end, we replace this upsample block with that in our image generator (see Table 3) by switching the batch normalization to the instance one.

A.6. Network architectures for image generation

We present the network architecture for image generators in Table 4 and the network architectures for discriminators in Table 5, Table 6 and Table 7. They are built with basic blocks defined in Table 3. We set the hyperparameters of the network structures as: $N_g = 48$, $N_d = 96$, $N_c = 80$, $N_e = 256$, $N_l = 50$, $m_0 = 7$, $m_1 = 3$, and $m_2 = 3$.

We employ an Adam optimizer for the generators with learning rate 0.0002, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. For each discriminator, we also employ an Adam optimizer with the same hyperparameters.

We design the object-wise discriminators for small objects and large objects, respectively. We specify that if the maximum of width or height of an object is greater than one-third of the image size, then this object is large; otherwise, it is small.

A.7. Network architectures for spectral normalized projection discriminators

We combine our discriminators above with the spectral normalized projection discriminator in [18, 19]. The difference between the object-wise discriminator and the object-wise spectral normalized projection discriminator is illustrated in Figure 16. We present detailed network architectures of the spectral normalized projection discriminators in Table 8, Table 9 and Table 10, with basic blocks defined in Table 3.

Table 2: The architecture of Bi-LSTM text encoder.

Layer Name	Hyperparameters
Embedding	num embeddings = vocab size, embedding dim = 300
Dropout	prob = 0.5
LSTM	input size = 300, hidden size ($\frac{D}{2}$) = 128, num layers = 1, dropout prob = 0.5, bidirectional = True

Table 3: The basic blocks for architecture design. ("-" connects two consecutive layers; "+" means element-wise addition between two layers.)

Name	Operations / Layers
Interpolating (k)	Nearest neighbor upsampling layer (up-scaling the spatial size by k)
Upsampling (k)	Interpolating (2) - convolution 3×3 (stride 1, padding 1, decreasing ♯channels to k) - Batch Normalization (BN) - Gated Linear Unit (GLU).
Downsampling (k)	In Gs: convolution 3×3 (stride 2, increasing ♯channels to k) - BN - LeakyReLU. In Ds, the convolutional kernel size is 4. In the first block of Ds, BN is not applied.
Downsampling w/ SN (k)	Convolution 4×4 (spectral normalized, stride 2, increasing ♯channels to k) - BN - LeakyReLU. In the first block of Ds, BN is not applied.
Concat	Concatenate input tensors along the channel dimension.
Residual	Input + [Reflection Pad (RPad) 1 - convolution 3×3 (stride 1, doubling ♯channels) - Instance Normalization (IN) - GLU - RPad 1 - convolution 3×3 (stride 1, keeping ♯channels) - IN].
FC	At the beginning of Gs: fully connected layer - BN - GLU - reshape to 3D tensor.
FC w/ SN (k)	Fully connected layer (spectral normalized, changing ♯channels to k).
Outlogits	Convolution 4×4 (stride 2, decreasing ♯channels to 1) - sigmoid.
Repeat ($k \times k$)	Copy a vector $k \times k$ times.
Fmap Sum	Summing the two input feature maps element-wisely.
Fmap Mul	Multiplying the two input feature maps element-wisely.
Avg Pool (k)	Average pooling along the k-th dimension.
Conv 3×3 (k)	In Gs: convolution 3×3 (stride 1, padding 1, changing ♯channels to k) - Tanh. In Ds, convolution 3×3 (stride 1, padding 1, changing ♯channels to k) - BN - LeakyReLU.
Conv 4×4 w/ SN	Convolution 4×4 (spectral normalized, stride 2, keeping ♯channels).
Conv 1×1 w/ SN	Convolution 1×1 (spectral normalized, stride 1, decreasing ♯channels to 1).
F^{ca}	Conditioning augmentation that converts the sentence embedding $\widehat{e}$ to the conditioning vector $\overline{e}$: fully connected layer - ReLU.
$F^{\text{pat-attn}}$	Grid attention module. Refer to the paper for more details.
$F^{\text{obj-attn}}$	Object-driven attention module. Refer to the paper for more details.
$F^{\text{lab-distr}}$	Label distribution module. Refer to the paper for more details.
Shape Encoder (k)	RPad 1 - convolution 3×3 (stride 1, decreasing ♯channels to k) - IN - LeakyReLU.
Shape Encoder w/ SN (k)	RPad 1 - convolution 3×3 (spectral normalized, stride 1, decreasing ♯channels to k) - IN - LeakyReLU.
ROI Encoder	Convolution 4×4 (stride 1, padding 1, decreasing ♯channels to $N_d * 4$) - LeakyReLU.
ROI Encoder w/ SN	Convolution 4×4 (spectral normalized, stride 1, padding 1, decreasing ♯channels to $N_d * 4$) - LeakyReLU.
ROI Align (k)	Pooling $k \times k$ feature maps for ROI.

Table 4: The structure for generators of Obj-GAN.

Stage	Name	Input Tensors	Output Tensors
G_0	FC	100-dimensional z, and F^{ca}	$8 \times 8 \times 4N_g$
	Upsampling $(2N_g)$	$8 \times 8 \times 4N_g$	$16 \times 16 \times 2N_g$
	Upsampling (N_g)	$16 \times 16 \times 2N_g$	$c\,(32 \times 32 \times N_g)$
	Shape Encoder $(\frac{1}{2}N_g)$	$M^0\,(64 \times 64 \times N_c)$	$64 \times 64 \times \frac{1}{2}N_g$
	Downsampling (N_g)	$64 \times 64 \times \frac{1}{2}N_g$	$u_0\,(32 \times 32 \times N_g)$
	Concat	$c, u_0, F^{\text{obj-attn}}, F^{\text{lab-distr}}$	$32 \times 32 \times (3N_g + N_l)$
	m_0 Residual	$32 \times 32 \times (3N_g + N_l)$	$32 \times 32 \times (3N_g + N_l)$
	Upsampling (N_g)	$32 \times 32 \times (3N_g + N_l)$	$h_0\,(64 \times 64 \times N_g)$
	Conv 3×3 (3)	h_0	$x_0\,(64 \times 64 \times 3)$
G_1	Shape Encoder $(\frac{1}{2}N_g)$	$M^1\,(128 \times 128 \times N_c)$	$128 \times 128 \times \frac{1}{2}N_g$
	Downsampling (N_g)	$128 \times 128 \times \frac{1}{2}N_g$	$u_1\,(64 \times 64 \times N_g)$
	Fmap Sum	h_0, u_1	$h_0\,(64 \times 64 \times N_g)$
	Concat	$F^{\text{pat-attn}}, h_0, F^{\text{obj-attn}}, F^{\text{lab-distr}}$	$64 \times 64 \times (3N_g + N_l)$
	m_1 Residual	$64 \times 64 \times (3N_g + N_l)$	$64 \times 64 \times (3N_g + N_l)$
	Upsampling (N_g)	$64 \times 64 \times (3N_g + N_l)$	$h_1\,(128 \times 128 \times N_g)$
	Conv 3×3 (3)	h_1	$x_1\,(128 \times 128 \times 3)$
G_2	Shape Encoder $(\frac{1}{2}N_g)$	$M^2\,(256 \times 256 \times N_c)$	$256 \times 256 \times \frac{1}{2}N_g$
	Downsampling (N_g)	$256 \times 256 \times \frac{1}{2}N_g$	$u_2\,(128 \times 128 \times N_g)$
	Fmap Sum	h_1, u_2	$h_1\,(128 \times 128 \times N_g)$
	Concat	$F^{\text{pat-attn}}, h_1, F^{\text{obj-attn}}, F^{\text{lab-distr}}$	$128 \times 128 \times (3N_g + N_l)$
	m_2 Residual	$128 \times 128 \times (3N_g + N_l)$	$128 \times 128 \times (3N_g + N_l)$
	Upsampling (N_g)	$128 \times 128 \times (3N_g + N_l)$	$h_2\,(256 \times 256 \times N_g)$
	Conv 3×3 (3)	h_2	$x_2\,(256 \times 256 \times 3)$

Table 5: The structure for patch-wise discriminators of Obj-GAN. $\overline{e}$ is output by F^{ca}

Stage	Name	Input Tensors	Output Tensors
D_0	Downsampling (N_d)	$x_0\,(64 \times 64 \times 3)$	$32 \times 32 \times N_d$
	Downsampling $(2N_d)$	$32 \times 32 \times N_d$	$16 \times 16 \times 2N_d$
	Downsampling $(4N_d)$	$16 \times 16 \times 2N_d$	$8 \times 8 \times 4N_d$
	Downsampling $(8N_d)$	$8 \times 8 \times 4N_d$	$h_0\,(4 \times 4 \times 8N_d)$
	Repeat (4×4)	$\overline{e}\,(N_e)$	$4 \times 4 \times N_e$
	Concat - Conv 3×3 $(8N_d)$	$h_0, 4 \times 4 \times N_e$	$he_0\,(4 \times 4 \times 8N_d)$
	Outlogits (unconditional loss)	h_0	1
	Outlogits (conditional loss)	he_0	1
D_1	Downsampling (N_d)	$x_1\,(128 \times 128 \times 3)$	$64 \times 64 \times N_d$
	Downsampling $(2N_d)$	$64 \times 64 \times N_d$	$32 \times 32 \times 2N_d$
	Downsampling $(4N_d)$	$32 \times 32 \times 2N_d$	$16 \times 16 \times 4N_d$
	Downsampling $(8N_d)$	$16 \times 16 \times 4N_d$	$h_1\,(8 \times 8 \times 8N_d)$
	Repeat (8×8)	$\overline{e}\,(N_e)$	$8 \times 8 \times N_e$
	Concat - Conv 3×3 $(8N_d)$	$h_1, 8 \times 8 \times N_e$	$he_1\,(8 \times 8 \times 8N_d)$
	Outlogits (unconditional loss)	h_1	3×3
	Outlogits (conditional loss)	he_1	3×3
D_2	Downsampling (N_d)	$x_2\,(256 \times 256 \times 3)$	$128 \times 128 \times N_d$
	Downsampling $(2N_d)$	$128 \times 128 \times N_d$	$64 \times 64 \times 2N_d$
	Downsampling $(4N_d)$	$64 \times 64 \times 2N_d$	$32 \times 32 \times 4N_d$
	Downsampling $(8N_d)$	$32 \times 32 \times 4N_d$	$h_2\,(16 \times 16 \times 8N_d)$
	Repeat (16×16)	$\overline{e}\,(N_e)$	$16 \times 16 \times N_e$
	Concat - Conv 3×3 $(8N_d)$	$h_2, 16 \times 16 \times N_e$	$he_2\,(16 \times 16 \times 8N_d)$
	Outlogits (unconditional loss)	h_2	7×7
	Outlogits (conditional loss)	he_2	7×7

Table 6: The structure for shape discriminators of Obj-GAN.

Stage	Name	Input Tensors	Output Tensors
D_0	Shape Encoder ($\frac{1}{8}N_d$)	M^0 ($64 \times 64 \times N_c$)	$64 \times 64 \times \frac{1}{8}N_d$
	Concat	x_0 ($64 \times 64 \times 3$), $64 \times 64 \times \frac{1}{8}N_d$	$64 \times 64 \times (3 + \frac{1}{8}N_d)$
	Downsampling (N_d)	$64 \times 64 \times (3 + \frac{1}{8}N_d)$	$32 \times 32 \times N_d$
	Downsampling ($2N_d$)	$32 \times 32 \times N_d$	$16 \times 16 \times 2N_d$
	Downsampling ($4N_d$)	$16 \times 16 \times 2N_d$	$8 \times 8 \times 4N_d$
	Downsampling ($8N_d$)	$8 \times 8 \times 4N_d$	h_0 ($4 \times 4 \times 8N_d$)
	Outlogits (unconditional loss)	h_0	1
D_1	Shape Encoder ($\frac{1}{8}N_d$)	M^1 ($128 \times 128 \times N_c$)	$128 \times 128 \times \frac{1}{8}N_d$
	Concat	x_1 ($128 \times 128 \times 3$), $128 \times 128 \times \frac{1}{8}N_d$	$128 \times 128 \times (3 + \frac{1}{8}N_d)$
	Downsampling (N_d)	$128 \times 128 \times (3 + \frac{1}{8}N_d)$	$64 \times 64 \times N_d$
	Downsampling ($2N_d$)	$64 \times 64 \times N_d$	$32 \times 32 \times 2N_d$
	Downsampling ($4N_d$)	$32 \times 32 \times 2N_d$	$16 \times 16 \times 4N_d$
	Downsampling ($8N_d$)	$16 \times 16 \times 4N_d$	h_1 ($8 \times 8 \times 8N_d$)
	Outlogits (unconditional loss)	h_1	3×3
D_2	Shape Encoder ($\frac{1}{8}N_d$)	M^2 ($256 \times 256 \times N_c$)	$256 \times 256 \times \frac{1}{8}N_d$
	Concat	x_2 ($256 \times 256 \times 3$), $256 \times 256 \times \frac{1}{8}N_d$	$256 \times 256 \times (3 + \frac{1}{8}N_d)$
	Downsampling (N_d)	$256 \times 256 \times (3 + \frac{1}{8}N_d)$	$128 \times 128 \times N_d$
	Downsampling ($2N_d$)	$128 \times 128 \times N_d$	$64 \times 64 \times 2N_d$
	Downsampling ($4N_d$)	$64 \times 64 \times 2N_d$	$32 \times 32 \times 4N_d$
	Downsampling ($8N_d$)	$32 \times 32 \times 4N_d$	h_2 ($16 \times 16 \times 8N_d$)
	Outlogits (unconditional loss)	h_2	7×7

Table 7: The structure for object-wise discriminators of Obj-GAN. c^{obj} represents the intermediate context vectors of $F^{\text{obj-attn}}$, and e^{g} represents the embedding vectors the class labels.

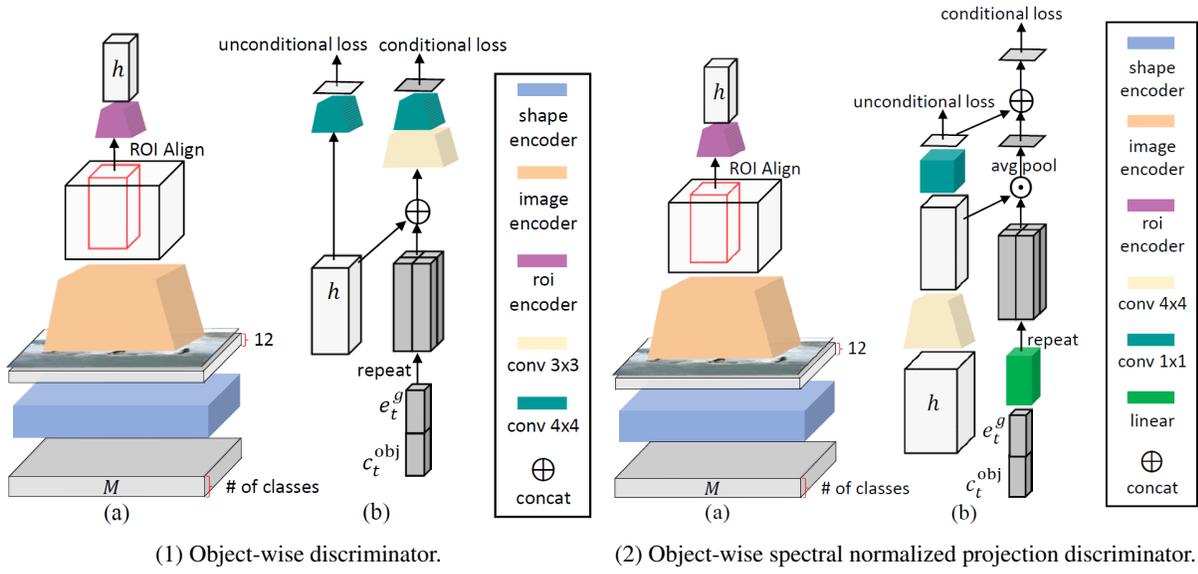| Stage	Name	Input Tensors	Output Tensors
small	Interpolating (2)	M^2 ($256 \times 256 \times N_c$)	$512 \times 512 \times N_c$
	Interpolating (2)	x^2 ($256 \times 256 \times 3$)	$512 \times 512 \times 3$
	Shape Encoder ($\frac{1}{8}N_d$)	$512 \times 512 \times N_c$	$512 \times 512 \times \frac{1}{8}N_d$
	Concat	$512 \times 512 \times 3$, $512 \times 512 \times \frac{1}{8}N_d$	$512 \times 512 \times (3 + \frac{1}{8}N_d)$
	Downsampling (N_d)	$512 \times 512 \times (3 + \frac{1}{8}N_d)$	$256 \times 256 \times N_d$
	Downsampling ($2N_d$)	$256 \times 256 \times N_d$	$128 \times 128 \times 2N_d$
	Downsampling ($4N_d$)	$128 \times 128 \times 2N_d$	$64 \times 64 \times 4N_d$
	ROI Align (5)	$64 \times 64 \times 4N_d$	$N_{\text{small}} \times 5 \times 5 \times 4N_d$
	ROI Encoder (5)	$N_{\text{small}} \times 5 \times 5 \times 4N_d$	h ($N_{\text{small}} \times 4 \times 4 \times 4N_d$)
	Repeat (4×4)	c^{obj} ($N_{\text{small}} \times N_g$)	$N_{\text{small}} \times 4 \times 4 \times N_g$
	Repeat (4×4)	e^{g} ($N_{\text{small}} \times N_l$)	$N_{\text{small}} \times 4 \times 4 \times N_l$
	Concat - Conv 3×3 ($4N_d$)	h, $N_{\text{small}} \times 4 \times 4 \times N_g$, $N_{\text{small}} \times 4 \times 4 \times N_l$	hc ($N_{\text{small}} \times 4 \times 4 \times 4N_d$)
	Outlogits (unconditional loss)	h	N_{small}
	Outlogits (conditional loss)	hc	N_{small}
large	Interpolating (2)	M^2 ($256 \times 256 \times N_c$)	$512 \times 512 \times N_c$
	Interpolating (2)	x^2 ($256 \times 256 \times 3$)	$512 \times 512 \times 3$
	Shape Encoder ($\frac{1}{8}N_d$)	$512 \times 512 \times N_c$	$512 \times 512 \times \frac{1}{8}N_d$
	Concat	$512 \times 512 \times 3$, $512 \times 512 \times \frac{1}{8}N_d$	$512 \times 512 \times (3 + \frac{1}{8}N_d)$
	Downsampling (N_d)	$512 \times 512 \times (3 + \frac{1}{8}N_d)$	$256 \times 256 \times N_d$
	Downsampling ($2N_d$)	$256 \times 256 \times N_d$	$128 \times 128 \times 2N_d$
	Downsampling ($4N_d$)	$128 \times 128 \times 2N_d$	$64 \times 64 \times 4N_d$
	Downsampling ($8N_d$)	$64 \times 64 \times 4N_d$	$32 \times 32 \times 8N_d$
	ROI Align (5)	$32 \times 32 \times 8N_d$	$N_{\text{large}} \times 5 \times 5 \times 8N_d$
	ROI Encoder (5)	$N_{\text{large}} \times 5 \times 5 \times 8N_d$	h ($N_{\text{large}} \times 4 \times 4 \times 4N_d$)
	Repeat (4×4)	c^{obj} ($N_{\text{large}} \times N_g$)	$N_{\text{large}} \times 4 \times 4 \times N_g$
	Repeat (4×4)	e^{g} ($N_{\text{large}} \times N_l$)	$N_{\text{large}} \times 4 \times 4 \times N_l$
	Concat - Conv 3×3 ($4N_d$)	h, $N_{\text{large}} \times 4 \times 4 \times N_g$, $N_{\text{large}} \times 4 \times 4 \times N_l$	hc ($N_{\text{large}} \times 4 \times 4 \times 4N_d$)
	Outlogits (unconditional loss)	h	N_{large}
	Outlogits (conditional loss)	hc	N_{large}

Figure 16: The comparison between the object-wise discriminator and its spectral normalized projection version. (a) extracts the region feature based on the Fast R-CNN model. (b) determines whether the t-th object is realistic (consistent with its label e_t^g and text context information c_t^{obj}) or not.

Table 8: The structure for patch-wise spectral normalized projection discriminators of Obj-GAN. $\bar{e}$ is output by F^{ca}

Stage	Name	Input Tensors	Output Tensors
D_0	Downsampling w/ SN (N_d)	x_0 ($64 \times 64 \times 3$)	$32 \times 32 \times N_d$
	Downsampling w/ SN ($2N_d$)	$32 \times 32 \times N_d$	$16 \times 16 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$16 \times 16 \times 2N_d$	$8 \times 8 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$8 \times 8 \times 4N_d$	$4 \times 4 \times 8N_d$
	Conv 4×4 w/ SN	$4 \times 4 \times 8N_d$	h_0 ($8N_d$)
	FC w/ SN ($8N_d$)	$\bar{e}$ (N_e)	c_0 ($8N_d$)
	Fmap Mul - Avg Pool (0)	h_0, c_0	hc_0 (1)
	Conv 1×1 w/ SN (unconditional loss)	h_0	o_0^{uncond} (1)
	Fmap Sum (conditional loss)	o_0^{uncond}, hc_0	o_0^{cond} (1)
D_1	Downsampling w/ SN (N_d)	x_1 ($128 \times 128 \times 3$)	$64 \times 64 \times N_d$
	Downsampling w/ SN ($2N_d$)	$64 \times 64 \times N_d$	$32 \times 32 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$32 \times 32 \times 2N_d$	$16 \times 16 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$16 \times 16 \times 4N_d$	$8 \times 8 \times 8N_d$
	Conv 4×4 w/ SN	$8 \times 8 \times 8N_d$	h_1 ($3 \times 3 \times 8N_d$)
	FC w/ SN ($8N_d$)	$\bar{e}$ (N_e)	$8N_d$
	Repeat (3×3)	$8N_d$	c_1 ($3 \times 3 \times 8N_d$)
	Fmap Mul - Avg Pool (2)	h_1, c_1	hc_1 (3×3)
	Conv 1×1 w/ SN (unconditional loss)	h_1	o_1^{uncond} (3×3)
	Fmap Sum (conditional loss)	o_1^{uncond}, hc_1	o_1^{cond} (3×3)
D_2	Downsampling w/ SN (N_d)	x_2 ($256 \times 256 \times 3$)	$128 \times 128 \times N_d$
	Downsampling w/ SN ($2N_d$)	$128 \times 128 \times N_d$	$64 \times 64 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$64 \times 64 \times 2N_d$	$32 \times 32 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$32 \times 32 \times 4N_d$	$16 \times 16 \times 8N_d$
	Conv 4×4 w/ SN	$16 \times 16 \times 8N_d$	h_2 ($7 \times 7 \times 8N_d$)
	FC w/ SN ($8N_d$)	$\bar{e}$ (N_e)	$8N_d$
	Repeat (7×7)	$8N_d$	c_2 ($7 \times 7 \times 8N_d$)
	Fmap Mul - Avg Pool (2)	h_2, c_2	hc_2 (7×7)
	Conv 1×1 w/ SN (unconditional loss)	h_2	o_2^{uncond} (7×7)
	Fmap Sum (conditional loss)	o_2^{uncond}, hc_2	o_2^{cond} (7×7)

Table 9: The structure for shape spectral normalized projection discriminators of Obj-GAN.

Stage	Name	Input Tensors	Output Tensors
D_0	Shape Encoder w/ SN ($\frac{1}{8}N_d$)	M^0 ($64 \times 64 \times N_c$)	$64 \times 64 \times \frac{1}{8}N_d$
	Concat	x_0 ($64 \times 64 \times 3$), $64 \times 64 \times \frac{1}{8}N_d$	$64 \times 64 \times (3 + \frac{1}{8}N_d)$
	Downsampling w/ SN (N_d)	$64 \times 64 \times (3 + \frac{1}{8}N_d)$	$32 \times 32 \times N_d$
	Downsampling w/ SN ($2N_d$)	$32 \times 32 \times N_d$	$16 \times 16 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$16 \times 16 \times 2N_d$	$8 \times 8 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$8 \times 8 \times 4N_d$	$4 \times 4 \times 8N_d$
	Conv 4×4 w/ SN	$4 \times 4 \times 8N_d$	h_0 ($8N_d$)
	Conv 1×1 w/ SN (unconditional loss)	h_0	1
D_1	Shape Encoder w/ SN ($\frac{1}{8}N_d$)	M^1 ($128 \times 128 \times N_c$)	$128 \times 128 \times \frac{1}{8}N_d$
	Concat	x_1 ($128 \times 128 \times 3$), $128 \times 128 \times \frac{1}{8}N_d$	$128 \times 128 \times (3 + \frac{1}{8}N_d)$
	Downsampling w/ SN (N_d)	$128 \times 128 \times (3 + \frac{1}{8}N_d)$	$64 \times 64 \times N_d$
	Downsampling w/ SN ($2N_d$)	$64 \times 64 \times N_d$	$32 \times 32 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$32 \times 32 \times 2N_d$	$16 \times 16 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$16 \times 16 \times 4N_d$	$8 \times 8 \times 8N_d$
	Conv 4×4 w/ SN	$8 \times 8 \times 8N_d$	h_1 ($3 \times 3 \times 8N_d$)
	Conv 1×1 w/ SN (unconditional loss)	h_1	3×3
D_2	Shape Encoder w/ SN ($\frac{1}{8}N_d$)	M^2 ($256 \times 256 \times N_c$)	$256 \times 256 \times \frac{1}{8}N_d$
	Concat	x_2 ($256 \times 256 \times 3$), $256 \times 256 \times \frac{1}{8}N_d$	$256 \times 256 \times (3 + \frac{1}{8}N_d)$
	Downsampling w/ SN (N_d)	$256 \times 256 \times (3 + \frac{1}{8}N_d)$	$128 \times 128 \times N_d$
	Downsampling w/ SN ($2N_d$)	$128 \times 128 \times N_d$	$64 \times 64 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$64 \times 64 \times 2N_d$	$32 \times 32 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$32 \times 32 \times 4N_d$	$16 \times 16 \times 8N_d$
	Conv 4×4 w/ SN	$16 \times 16 \times 8N_d$	h_2 ($7 \times 7 \times 8N_d$)
	Conv 1×1 w/ SN (unconditional loss)	h_2	7×7

Table 10: The structure for object-wise spectral normalized projection discriminators of Obj-GAN. c^{obj} represents the intermediate context vectors of $F^{\text{obj-attn}}$, and e^{g} represents the embedding vectors the class labels.

Stage	Name	Input Tensors	Output Tensors
small	Interpolating (2)	M^2 ($256 \times 256 \times N_c$)	$512 \times 512 \times N_c$
	Interpolating (2)	x^2 ($256 \times 256 \times 3$)	$512 \times 512 \times 3$
	Shape Encoder w/ SN ($\frac{1}{8}N_d$)	$512 \times 512 \times N_c$	$512 \times 512 \times \frac{1}{8}N_d$
	Concat	$512 \times 512 \times 3$, $512 \times 512 \times \frac{1}{8}N_d$	$512 \times 512 \times (3 + \frac{1}{8}N_d)$
	Downsampling w/ SN (N_d)	$512 \times 512 \times (3 + \frac{1}{8}N_d)$	$256 \times 256 \times N_d$
	Downsampling w/ SN ($2N_d$)	$256 \times 256 \times N_d$	$128 \times 128 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$128 \times 128 \times 2N_d$	$64 \times 64 \times 4N_d$
	ROI Align (5)	$64 \times 64 \times 4N_d$	$N_{\text{small}} \times 5 \times 5 \times 4N_d$
	ROI Encoder w/SN (5)	$N_{\text{small}} \times 5 \times 5 \times 4N_d$	$N_{\text{small}} \times 4 \times 4 \times 4N_d$
	Conv 4×4 w/ SN	$N_{\text{small}} \times 4 \times 4 \times 4N_d$	h ($N_{\text{small}} \times 4N_d$)
	Concat	c^{obj} ($N_{\text{small}} \times N_g$), e^{g} ($N_{\text{small}} \times N_l$)	$N_{\text{small}} \times (N_g + N_l)$
	FC w/ SN ($4N_d$)	$N_{\text{small}} \times (N_g + N_l)$	c ($N_{\text{small}} \times 4N_d$)
	Fmap Mul - Avg Pool (1)	h, c	hc (N_{small})
	Conv 1×1 w/ SN (unconditional loss)	h	o^{uncond} (N_{small})
	Fmap Sum (conditional loss)	o^{uncond}, hc	o^{cond} (N_{small})
large	Interpolating (2)	M^2 ($256 \times 256 \times N_c$)	$512 \times 512 \times N_c$
	Interpolating (2)	x^2 ($256 \times 256 \times 3$)	$512 \times 512 \times 3$
	Shape Encoder w/ SN ($\frac{1}{8}N_d$)	$512 \times 512 \times N_c$	$512 \times 512 \times \frac{1}{8}N_d$
	Concat	$512 \times 512 \times 3$, $512 \times 512 \times \frac{1}{8}N_d$	$512 \times 512 \times (3 + \frac{1}{8}N_d)$
	Downsampling w/ SN (N_d)	$512 \times 512 \times (3 + \frac{1}{8}N_d)$	$256 \times 256 \times N_d$
	Downsampling w/ SN ($2N_d$)	$256 \times 256 \times N_d$	$128 \times 128 \times 2N_d$
	Downsampling w/ SN ($4N_d$)	$128 \times 128 \times 2N_d$	$64 \times 64 \times 4N_d$
	Downsampling w/ SN ($8N_d$)	$64 \times 64 \times 4N_d$	$32 \times 32 \times 8N_d$
	ROI Align (5)	$32 \times 32 \times 8N_d$	$N_{\text{large}} \times 5 \times 5 \times 8N_d$
	ROI Encoder w/ SN (5)	$N_{\text{large}} \times 5 \times 5 \times 8N_d$	$N_{\text{large}} \times 4 \times 4 \times 4N_d$
	Conv 4×4 w/ SN	$N_{\text{large}} \times 4 \times 4 \times 4N_d$	h ($N_{\text{large}} \times 4N_d$)
	Concat	c^{obj} ($N_{\text{large}} \times N_g$), e^{g} ($N_{\text{large}} \times N_l$)	$N_{\text{large}} \times (N_g + N_l)$
	FC w/ SN ($4N_d$)	$N_{\text{large}} \times (N_g + N_l)$	c ($N_{\text{large}} \times 4N_d$)
	Fmap Mul - Avg Pool (1)	h, c	hc (N_{large})
	Conv 1×1 w/ SN (unconditional loss)	h	o^{uncond} (N_{large})
	Fmap Sum (conditional loss)	o^{uncond}, hc	o^{cond} (N_{large})