




Layout2Image: Image Generation from Layout

Bo Zhao^{1,2,3}  · Weidong Yin^{1,3} · Lili Meng¹ · Leonid Sigal^{1,3}

Received: 14 April 2019 / Accepted: 2 February 2020 / Published online: 24 February 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Despite significant recent progress on generative models, controlled generation of images depicting multiple and complex object layouts is still a difficult problem. Among the core challenges are the diversity of appearance a given object may possess and, as a result, exponential set of images consistent with a specified layout. To address these challenges, we propose a novel approach for layout-based image generation; we call it Layout2Im. Given the coarse spatial layout (bounding boxes + object categories), our model can generate a set of realistic images which have the correct objects in the desired locations. The representation of each object is disentangled into a specified/certain part (category) and an unspecified/uncertain part (appearance). The category is encoded using a word embedding and the appearance is distilled into a low-dimensional vector sampled from a normal distribution. Individual object representations are composed together using convolutional LSTM, to obtain an encoding of the complete layout, and then decoded to an image. Several loss terms are introduced to encourage accurate and diverse image generation. The proposed Layout2Im model significantly outperforms the previous state-of-the-art, boosting the best reported inception score by 24.66% and 28.57% on the very challenging COCO-Stuff and Visual Genome datasets, respectively. Extensive experiments also demonstrate our model's ability to generate complex and diverse images with many objects.

Keywords Scene image generation · Image translation · Image generation · Generative adversarial networks

1 Introduction

Image generation of complex realistic scenes with multiple objects and desired layouts is one of the core frontiers for computer vision. Existence of such algorithms would not

only inform our designs for inference mechanisms, needed for visual understanding, but also provide practical application benefits in terms of automatic image generation for artists and users. In fact, such algorithms, if successful, may replace visual search and retrieval engines in their entirety. Why search the web for an image, if you can create one to user's specification?

For these reasons, image generation algorithms have been a major focus of recent research. Of specific relevance are approaches for text-to-image (Hong et al. 2018; Karacan et al. 2016; Mansimov et al. 2015; Reed et al. 2017; Tan et al. 2018; Zhang et al. 2017) generation. By allowing users to describe visual concepts in natural language, text-to-image generation provides natural and flexible interface for conditioned image generation. However, existing text-to-image approaches exhibit two drawbacks: (i) most approaches can only generate plausible results on simple datasets such as cats (Zhang et al. 2008), birds (Welinder et al. 2010) or flowers (Nilsback and Zisserman 2008). Generating complex, real-world images such as those in COCO-Stuff (Caesar et al. 2016) and Visual Genome (Krishna et al. 2017) datasets remains a challenge; (ii) the ambiguity of textual description

Communicated by Jun-Yan Zhu, Hongsheng Li, Eli Shechtman, Ming-Yu Liu, Jan Kautz, Antonio Torralba.

Bo Zhao and Weidong Yin have contributed equally to this work.

✉ Bo Zhao
zhaobo.cs@gmail.com

Weidong Yin
wdyin@cs.ubc.ca

Lili Meng
menglili@cs.ubc.ca

Leonid Sigal
lsigal@cs.ubc.ca

¹ Department of Computer Science, The University of British Columbia, Vancouver, Canada

² Bank of Montreal AI, Toronto, Canada

³ Vector Institute, Toronto, Canada

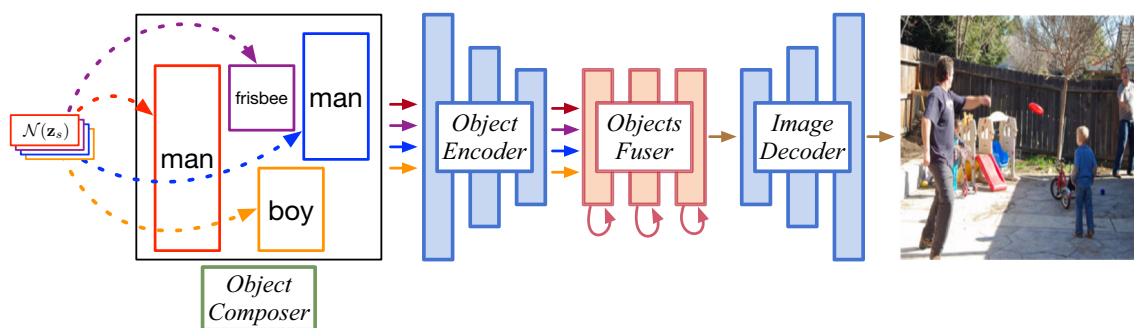


Fig. 1 Image generation from layout. Given the coarse layout (bounding boxes + object categories), the proposed Layout2Im model samples the appearance of each object from a normal distribution, and trans-

forms these inputs into a real image by a serial of components. Please refer to Sect. 3 for a detailed explanation

makes it more difficult to constrain complex generation process, e.g., locations and sizes of different objects are usually not given in the description.

Scene graphs are powerful structured representations that encode objects, their attributes and relationships. In Johnson et al. (2018) an approach for generating complex images with many objects and relationships is proposed by conditioning the generation on scene graphs. It addresses some of the aforementioned challenges. However, scene graphs are difficult to construct for a layman user and lack specification of core spatial properties, e.g., object size/position.

To overcome these limitations, we propose to generate complicated real-world images from layouts, as illustrated in Fig. 1. By simply specifying the coarse layout (bounding boxes + categories) of the expected image, our proposed model can generate an image which contains the desired objects in the correct locations. It is much more controllable and flexible to generate an image from layout than textual description.

With the new task comes new challenges. First, image generation from layout is a difficult one-to-many problem. Many images could be consistent with a specified layout; same layout may be realized by different appearance of objects, or even their interactions (e.g., a person next to the frisbee may be throwing it or be a bystander, see Fig. 1). Second, the information conveyed by a bounding box and corresponding label is very limited. The actual appearance of the object displayed in an image is not only determined by its category and location, but also its interactions and consistency with other objects. Moreover, spatially close objects may have overlapping bounding boxes. This leads to additional challenges of “separating” which object should contribute to individual pixels. A good generative model should take all these factors and challenges into account implicitly or explicitly.

We address these challenges using a novel variational inference approach. The representation of each object in the image is explicitly disentangled into a specified/certain part

(category) and an unspecified/uncertain part (appearance). The category is encoded using a word embedding and the appearance is distilled into a low-dimensional vector sampled from a normal distribution. Based on this representation and specification of object bounding box, we construct a feature map for each object. These feature maps are then composed using convolutional LSTM into a hidden feature map for the entire image, which subsequently is decoded into an output image. This set of modelling choices makes it easy to generate different and diverse images by sampling the appearance of individual objects, and/or adding, moving or deleting objects from the layout. Our proposed model is end-to-end learned using a loss that consists of a number of objectives. Specifically, a pair of discriminators are designed to discriminate the overall generated image and the generated objects within their specified bounding boxes, as real or fake. In addition, object discriminator is also trained to classify the categories of generated objects.

Contributions Our contributions are three-fold:

- We propose a novel approach for generating images from coarse layout (bounding boxes + object categories). This provides a flexible control mechanism for image generation;
- By disentangling the representation of objects into a category and (sampled) appearance, our model is capable of generating a diverse set of consistent images from the same layout;
- We propose an object-wise attention mechanism, which enables the network to model shape of different objects in an explicit manner and therefore enhances the overall visual quality.
- We show qualitative and quantitative results on COCO-Stuff (Caesar et al. 2016) and Visual Genome (Krishna et al. 2017) datasets, demonstrating our model’s ability to generate complex images with respect to object categories and their layout (without access to segmentation

masks (Hong et al. 2018; Johnson et al. 2018)). We also perform comprehensive ablations to validate each component in our approach.

Summary of Changes This work is an extension of our CVPR paper *Image Generation from Layout*. The main changes of this extended version are:

- Explicitly define the loss functions;
- Extend Object Feature Map Composition Module with Object-Wise Attention.
- Add one new section in the related work for semantic image generation.
- Add three new baselines for comparison: BicycleGAN, pix2pixHD and GauGAN (SPADE), and report their performance in inception score, FID, object classification accuracy and diversity score; add qualitative results on COCO and VG datasets.
- Add four ablation experiments: (a) remove the path of generating I' from normal distribution; (b) remove both latent code reconstruction loss and KL loss; (c) replace convolution LSTM with sum; (d) remove both latent code reconstruction loss, KL loss, and replace convolution LSTM with sum.
- Report FID scores for the ablation study in Table 6.

The source code of this work is available at <https://github.com/zhaobozb/layout2im>.

2 Related Work

2.1 Conditional Image Generation

Conditional image generation approaches generate images conditioned on additional input information, including entire source image (Isola et al. 2017; Liu et al. 2017; Pathak et al. 2016; Yang et al. 2017; Zhao et al. 2018; Zhu et al. 2017a, b), sketches (Isola et al. 2017; Sangkloy et al. 2017; Wang et al. 2018; Xian et al. 2018; Zhu et al. 2017b), scene graphs (Johnson et al. 2018), dialogues (Kim et al. 2017; Sharma et al. 2018) and text descriptions (Mansimov et al. 2015; Reed et al. 2017; Tan et al. 2018; Zhang et al. 2017). Variational Autoencoders (VAEs) (Kingma and Welling 2014; Mansimov et al. 2015; Sohn et al. 2015), autoregressive models (Oord et al. 2016; van den Oord et al. 2016) and GANs (Isola et al. 2017; Mirza and Osindero 2014; Wang et al. 2018; Zhu et al. 2017a) are powerful tools for conditional image generation and have shown promising results. However, many previous generative models (Isola et al. 2017; Pathak et al. 2016; Sangkloy et al. 2017; Xian et al. 2018; Yang et al. 2017; Zhu et al. 2017a) tend to largely ignore the random noise vector when conditioning on the same relevant context, making the gen-

erated images very similar to each other. By enforcing the bijection mapping between the latent and target space, BicycleGAN (Zhu et al. 2017b) pursues the diversity of generated images from the same input. Inspired by this idea, in our paper, we also explicitly regress the latent codes which are used to generate the different objects.

2.2 Image Generation from Layout

Existing models usually use key points of the object to generate specific type of image, i.e. human body (Ma et al. 2018b) or birds (Reed et al. 2016). However these models cannot be easily generalized to different objects in general image generation. The use of layout in image generation is a relative novel task. It is usually served as an intermediate representation between other input sources [e.g., text (Hong et al. 2018) or scene graphs (Johnson et al. 2018)] and the output images, or as a complementary feature for image generation based on context [e.g., text (Karacan et al. 2016; Reed et al. 2016; Tan et al. 2018), shape and lighting (Dosovitskiy et al. 2015)]. In Hong et al. (2018) and Johnson et al. (2018), instead of learning a direct mapping from textual description/scene graph to an image, the generation process is decomposed into multiple individual steps. They first construct a semantic layout (bounding boxes + object shapes) from the input, and then convert it to an image using an image generator. Both of them can generate an image from a coarse layout together with textual description/scene graph. However, Hong et al. (2018) requires detailed object instance segmentation masks to train its object shape generator. Getting such segmentation masks for large scale datasets is both time-consuming and labor-intensive. Different from Hong et al. (2018) and Johnson et al. (2018), we use the coarse layout without instance segmentation mask as a fundamental input modality for diverse image generation.

2.3 Semantic Image Generation

The task of image generation from semantic label maps has received lots of attention recently. Pix2pix (Isola et al. 2017) used an encoder-decoder generator with patch discriminator to translate input semantic label maps into images. Pix2pixHD (Wang et al. 2017) proposed a coarse-to-fine generator and multi-scale discriminators to generate high resolution images. GauGAN (Park et al. 2019) used input semantic label maps to generate normalization parameters for different layers. This track of work requires that accurate semantic label maps are provided as input where shape and location of each object is encoded inside the input. Different from these works, our method only requires coarse layout as input.

2.4 Disentangled Representations

Many papers (Chen et al. 2016; Cheung et al. 2015; Denton and Birodkar 2017; Lai et al. 2017; Lee et al. 2018; Ma et al. 2018a; Mathieu et al. 2016; Murez et al. 2018) have tried to learn disentangled representations as part of image generation. Disentangled representations model different factors of data variations, such as class-related and class-independent parts (Cheung et al. 2015; Lai et al. 2017; Lee et al. 2018; Mathieu et al. 2016; Murez et al. 2018). By manipulating the disentangled representations, images with different appearances can be generated easily. In Ma et al. (2018a), three factors (foreground, background and pose) are disentangled explicitly when generating person image. InfoGAN (Chen et al. 2016), DrNet (Denton and Birodkar 2017) and DRIT (Lee et al. 2018) learn the disentangled representations in an unsupervised manner, either by maximizing the mutual information (Chen et al. 2016) or adversarial losses (Denton and Birodkar 2017; Lee et al. 2018). In our work, we explicitly separate the representation of each object into a category-related and an appearance-related parts, and only the bounding boxes and category labels are used during both training and testing.

3 Image Generation from Layout

The overall **training** pipeline of the proposed approach is illustrated in Fig. 2. Given a ground-truth image \mathbf{I} and its corresponding layout \mathbf{L} , where $\mathbf{L}_i = (x_i, y_i, h_i, w_i)$ containing the top-left coordinate, height and width of the bounding box, our model first samples two latent codes \mathbf{z}_{ri} and \mathbf{z}_{si} for each object instance \mathbf{O}_i . The \mathbf{z}_{ri} is sampled from the posterior $Q(\mathbf{z}_r|\mathbf{O}_i)$ conditioned on object \mathbf{O}_i cropped from the input image according to \mathbf{L}_i . The \mathbf{z}_{si} is sampled from a normal prior distribution $\mathcal{N}(\mathbf{z}_s)$. Each object \mathbf{O}_i also has a word embedding \mathbf{w}_i , which is an embedding of its category label y_i . Based on the latent codes $\mathbf{z}_i \in \{\mathbf{z}_{ri}, \mathbf{z}_{si}\}$, word embedding \mathbf{w}_i , and layout \mathbf{L}_i , multiple object feature maps \mathbf{F}_i are constructed, and then fed into the object encoder and the objects fuser sequentially, generating a fused hidden feature map \mathbf{H} containing information from all specified objects. Finally, an image decoder D is used to reconstruct, $\hat{\mathbf{I}} = D(\mathbf{H})$, the input ground-truth image \mathbf{I} and generate a new image \mathbf{I}' , simultaneously; the former comes from $\mathbf{z}_r = \{\mathbf{z}_{ri}\}$ and the latter from $\mathbf{z}_s = \{\mathbf{z}_{si}\}$. Notably, both resulting images match the training image input layout. To make the mapping between the generated object \mathbf{O}'_i and the sampled latent code \mathbf{z}_{si} consistent, we make the object estimator regress the sampled latent codes \mathbf{z}_{si} based on the generated object \mathbf{O}'_i in \mathbf{I}' at locations \mathbf{L}_i . To train the model adversarially, we also introduce a pair of discriminators, D_{img} and D_{obj} , to classify the results at image and object level as being real or fake.

Once the model is trained, it can generate a new image from a layout by sampling object latent codes from the normal prior distribution $\mathcal{N}(\mathbf{z}_s)$ as illustrated in Fig. 1.

3.1 Object Latent Code Estimation

Object latent code posterior distributions are first estimated from the ground-truth image, and used to sample object latent code $\mathbf{z}_{ri} \sim \mathcal{N}(\mu(\mathbf{O}_i), \sigma(\mathbf{O}_i))$. These object latent codes model the ambiguity in object appearance in the ground-truth image, and play important roles in reconstructing the input image later.

Figure 3 illustrates the object latent code estimation process. First, each object \mathbf{O}_i is cropped, from the input image \mathbf{I} according to its bounding box \mathbf{L}_i , and then resized to fit the input dimensionality of object estimator using bilinear interpolation. The resized object crops are fed into an object estimator which consists of several convolutional layers and two fully-connected layers. The object estimator predicts the mean and variance of the posterior distribution for each input object \mathbf{O}_i . Finally, the predicted mean and variance are used to sample a latent code \mathbf{z}_{ri} for the input object \mathbf{O}_i . We sample latent code for every object in the input image.

3.2 Object Feature Map Composition

Given the object latent code $\mathbf{z}_i \in \mathbb{R}^m$ sampled from either posterior or the prior ($\mathbf{z}_i \in \{\mathbf{z}_{ri}, \mathbf{z}_{si}\}$), object category label y_i and corresponding bounding box information \mathbf{L}_i , the object composer module constructs a feature map \mathbf{F}_i for each object \mathbf{O}_i . Each feature map \mathbf{F}_i contains a region corresponding to \mathbf{L}_i filled with the disentangled representation of that object, consisting of object identity and appearance.

Figure 4 illustrates this module. The object category label y_i is first transformed to a corresponding word vector embedding $\mathbf{w}_i \in \mathbb{R}^n$, and then concatenated with the object latent vector \mathbf{z}_i . This results in the representation of the object which has two parts: object embedding and object latent code. Intuitively, the object embedding encodes the identity of the object, while the latent code encodes the appearance of a specific instance of that object. Jointly these two components encode sufficient information to reconstruct a specific instance of the object in an image. The object feature map \mathbf{F}_i is composed by simply filling the region within its bounding box with this object representation $(\mathbf{w}_i, \mathbf{z}_i) \in \mathbb{R}^{m+n}$. For each tuple $\langle y_i, \mathbf{z}_i, \mathbf{L}_i \rangle$ encoding object label, latent code and bounding box, we compose an object feature map \mathbf{F}_i . These object feature maps are downsampled by an object encoder network which contains several convolutional layers. Then an object fuser module is used to fuse all the downsampled object feature maps, generating a hidden feature map \mathbf{H} .

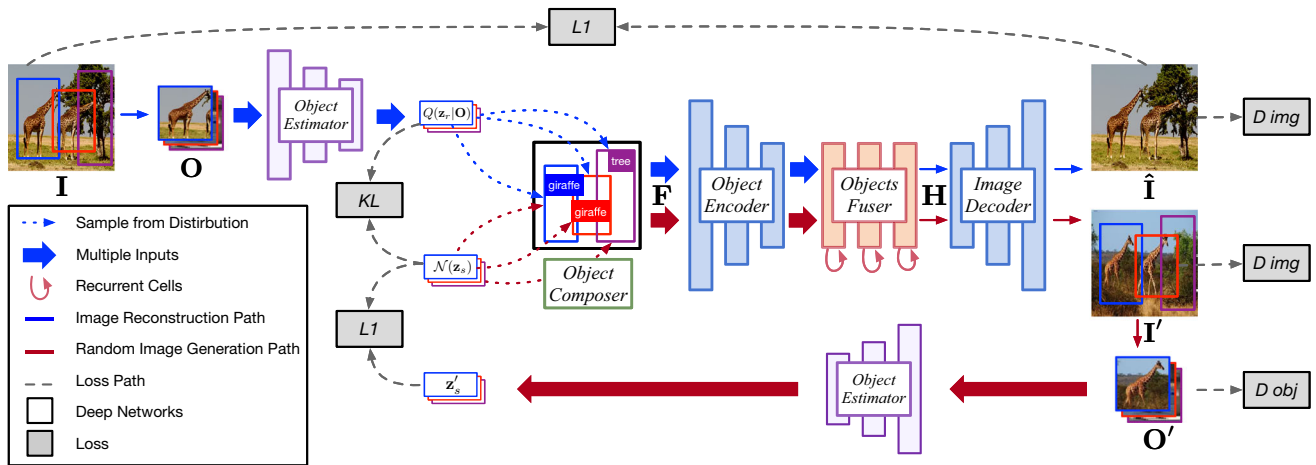
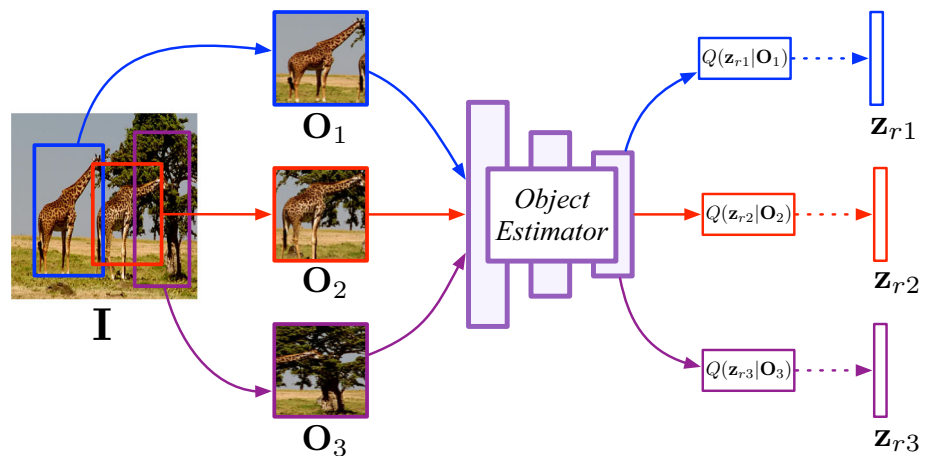


Fig. 2 Overview of our Layout2Img network for generating images from layout during training. The inputs to the model are the ground truth image with its layout. The objects are first cropped from the input image according to their bounding boxes, and then processed with the object estimator to predict a latent code for each object. After that, multiple object feature maps are prepared by the object composer based on the latent codes and layout, and processed with the object encoder, objects

fuser and image decoder to reconstruct the input image. Additional set of latent codes are also sampled from a normal distribution to generate a new image. Finally, objects in generated images are used to regress the sampled latent codes. The model is trained adversarially against a pair of discriminators and a number of objectives. For clarity, we omit D_{obj} for the objects cropped from \hat{I}

Fig. 3 Object latent code estimation. Given the input image and its layout, the objects are first cropped and resized from the input image. Then the object estimator predicts a distribution for each object from the object crops, and multiple latent codes are sampled from the estimated distribution



3.3 Object Feature Map Composition with Object-Wise Attention (OWA)

Our original method produces object feature map F_i by simply filling the region within its bounding boxes with object representations. This method has its limitations in that the shape of different classes of objects are all provided as rectangles and the network needs to figure out the shape implicitly with following fusion layers and decoders. To alleviate this problem, we proposed Object-Wise Attention(OWA). After we get the word embedding w_i representing different classes of objects, an attention decoder M is used to generate corresponding object wise attention mask $a_i = M(w_i)$ for different objects. Then we fill out each feature map F'_i with its object representations (w_i, z_i) using the corresponding atten-

tion mask a_i . Thus the shape of objects is modeled explicitly so that the network does not need to figure out the shape of different objects during decoding process. It can focus more on textures and coherence between objects to enhance visual quality. This is validated in following experiments (Fig. 5).

3.4 Object Feature Maps Fusion

Since the result image will be decoded from it, a good hidden feature map H is crucial to generating a realistic image. The properties of a good hidden feature map can be summarized as follows: (i) it should encode all object instances in the desired locations; (ii) it should coordinate object representations based on other objects in the image; (iii) it should be able to fill the unspecified regions, e.g., background, by implic-

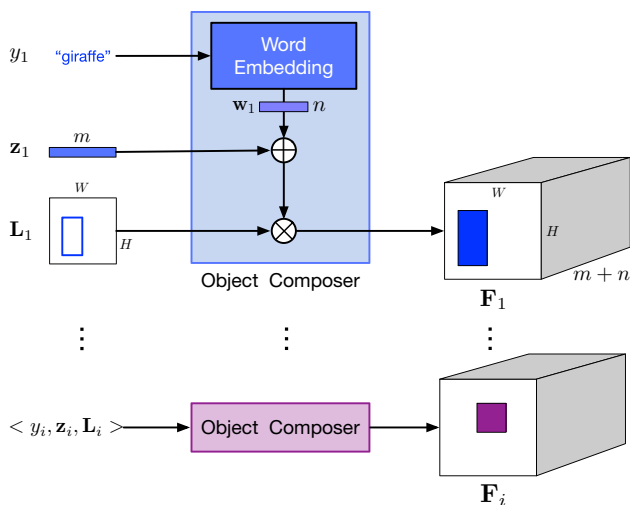


Fig. 4 Object feature map composition. The object category is first encoded by a word embedding. Then the object feature map is simply composed by filling the region within the object bounding box with the concatenation of category embedding and latent code. The rest of the feature map are all zeros. Symbol \oplus stands for the vector concatenation, and \otimes means replicating object representation within a bounding box

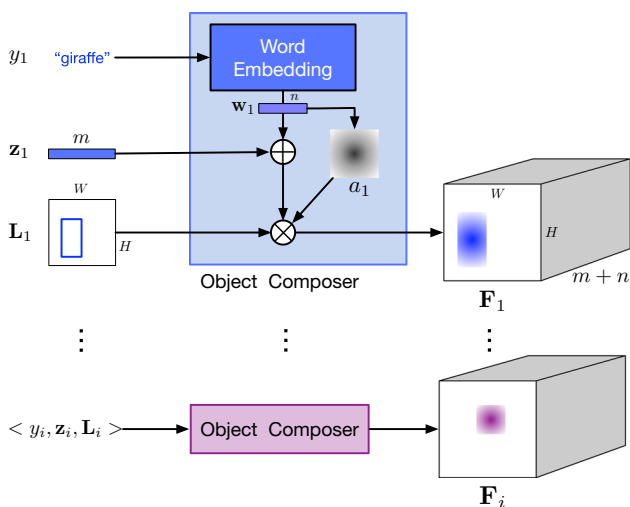


Fig. 5 Object Feature Map Composition with Object-Wise Attention(OWA). Instead of using the object bounding box to construct the object feature map as shown in Fig. 4, here we also predict an attention map for each object from the object embedding. By interpolating the attention map fitting into the bounding box, it provides finer guidance to generate target objects

itly reasoning about plausibility of the scene with respect to the specified objects.

To satisfy these requirements, we choose a multi-layer convolutional Long-Short-Term Memory (cLSTM) network (Shi et al. 2015) to fuse the downsampled object feature maps \mathbf{F} as illustrated in Fig. 6. Different from the traditional LSTM (Hochreiter and Schmidhuber 1997), the hidden states and cell states in cLSTM are both feature maps rather than vectors. The computation of different gates are also done by

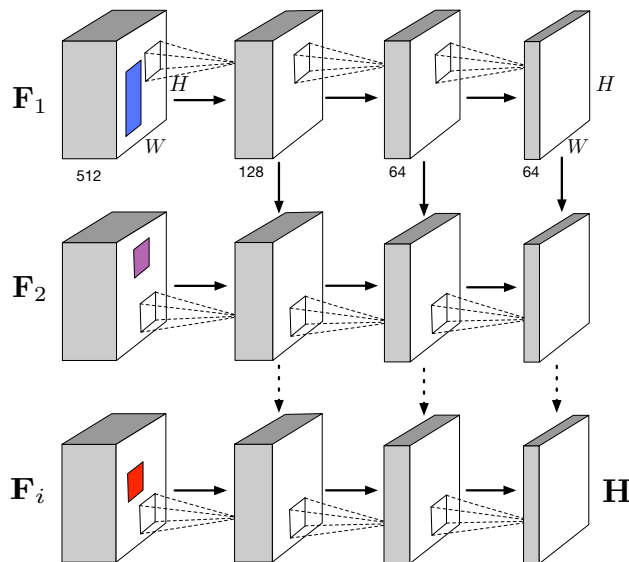


Fig. 6 Object Feature Maps Fusion. Three layers CLSTM is used to encode all object feature maps together

convolutional layers. Therefore, cLSTM can better preserve the spatial information compared with the traditional vector-based LSTM. The cLSTM acts like an encoder to integrate object feature maps one-by-one, and the last output of the cLSTM is used as the fused hidden layout \mathbf{H} , which incorporates the location and category information of all objects.

3.5 Image Decoder

Given the fused image hidden feature map \mathbf{H} , image decoder is tasked with generating a result image. As shown in Fig. 2, there are two paths (blue and red) in the networks. They differ in latent code estimation. The blue path reconstructs the input image using the object latent codes \mathbf{z}_r sampled from the posteriors $Q(\mathbf{z}_r|\mathbf{O})$ that are conditioned on the objects \mathbf{O} in the input image \mathbf{I} , while in the red one, the latent codes \mathbf{z}_s are directly sampled from prior distributions $\mathcal{N}(\mathbf{z}_s)$. As a result, two images are generated, i.e., $\hat{\mathbf{I}}$ and \mathbf{I}' , through the red and blue paths, respectively. Although they may differ in appearance, both of them share the same layout.

3.6 Object Latent Code Regression

To explicitly encourage the consistent connection between the latent codes and outputs, our model also tries to recover the random sampled latent codes from the objects generated along the red path. One can think of this as an inference network for the latent codes. This helps prevent a many-to-one mapping from the latent code to the output during training, and as a result, produces more diverse results.

To achieve this, we use the same input object bounding boxes \mathbf{L} to crop the objects \mathbf{O}' in the generated image \mathbf{I}' . The

resized \mathbf{O}' are then sent to an object latent code estimator (which shares weights with the one used in image reconstruction path), getting the estimated mean and variance vectors for the generated objects. We directly use the computed mean vectors, as the regressed latent codes \mathbf{z}'_s , and compare them with the sampled ones \mathbf{z}_s , for all objects.

3.7 Image and Object Discriminators

To make the generated images realistic, and the objects recognizable, we adopt a pair of discriminators D_{img} and D_{obj} . The discriminator is trained to classify an input image or object as real or fake. Meanwhile, the generator networks are trained to fool the discriminator.

The image discriminator D_{img} is applied to input images \mathbf{I} , reconstructed images $\hat{\mathbf{I}}$ and sampled images \mathbf{I}' , classifying them as real or fake. The object discriminator D_{obj} is designed to assess the quality and category of the real objects \mathbf{O} , reconstructed objects $\hat{\mathbf{O}}$ and sampled objects \mathbf{O}' at the same time. In addition, since $\hat{\mathbf{O}}$ and \mathbf{O}' are cropped from the reconstructed/sampled images according to the input bounding boxes \mathbf{L} , D_{obj} also encourages the generated objects to appear in their desired locations.

3.8 Loss Function

We end-to-end train the generator network and two discriminator networks in an adversarial manner. The generator network, with all described components, is trained to minimize the weighted sum of six losses:

KL Loss computes the KL-Divergence between the distribution $Q(\mathbf{z}_r|\mathbf{O})$ and the normal distribution $\mathcal{N}(\mathbf{z}_r)$, which is defined as:

$$\mathcal{L}_{\text{KL}} = \sum_{i=1}^o \mathbb{E}[\mathcal{D}_{\text{KL}}(Q(\mathbf{z}_{ri}|\mathbf{O}_i)||\mathcal{N}(\mathbf{z}_r))], \quad (1)$$

where o is the number of objects in the image/layout, \mathbf{O} are the cropped objects, and \mathbf{z} are the appearance codes of objects.

Image Reconstruction Loss penalizes the difference between ground-truth image \mathbf{I} and reconstructed image $\hat{\mathbf{I}}$. In our paper, \mathcal{L}_1 distance is chosen as shown in Eq. (2).

$$\mathcal{L}_1^{\text{img}} = \|\mathbf{I} - \hat{\mathbf{I}}\|_1. \quad (2)$$

Object Latent Code Reconstruction Loss encourages the connection between specific appearance and the latent code to be invertible, which is defined as the \mathcal{L}_1 distance between the randomly sampled $\mathbf{z}_s \sim N(\mathbf{z}_s)$ and the re-estimated \mathbf{z}'_s from the generated objects \mathbf{O}' :

$$\mathcal{L}_1^{\text{latent}} = \sum_{i=1}^o \|\mathbf{z}_{si} - \mathbf{z}'_{si}\|_1. \quad (3)$$

Image Adversarial Loss encourages the model to generate realistic images, which is defined as:

$$\begin{aligned} \mathcal{L}_{\text{GAN}}^{\text{img}} = & \mathbb{E}_{\mathbf{I} \sim p_{\text{real}}} \log D(\mathbf{I}) + 0.5 \mathbb{E}_{\hat{\mathbf{I}} \sim \hat{p}_{\text{fake}}} \log(1 - D(\hat{\mathbf{I}})) \\ & + 0.5 \mathbb{E}_{\mathbf{I}' \sim p'_{\text{fake}}} \log(1 - D(\mathbf{I}')), \end{aligned} \quad (4)$$

where \mathbf{I} is the ground truth image, $\hat{\mathbf{I}}$ is the reconstructed image and \mathbf{I}' is the sampled image.

Object Adversarial Loss encourages the model to generate realistic objects within an image, which is defined as:

$$\begin{aligned} \mathcal{L}_{\text{GAN}}^{\text{obj}} = & \mathbb{E}_{\mathbf{O} \sim p_{\text{real}}} \log D(\mathbf{O}) \\ & + 0.5 \mathbb{E}_{\hat{\mathbf{O}} \sim \hat{p}_{\text{fake}}} \log(1 - D(\hat{\mathbf{O}})) \\ & + 0.5 \mathbb{E}_{\mathbf{O}' \sim p'_{\text{fake}}} \log(1 - D(\mathbf{O}')), \end{aligned} \quad (5)$$

where \mathbf{O} are the objects cropped from the ground truth image \mathbf{I} , $\hat{\mathbf{O}}$ and \mathbf{O}' are objects cropped from the reconstructed image $\hat{\mathbf{I}}$ and sampled image \mathbf{I}' , respectively.

Auxiliary Classification Loss is adopted to classify the generated objects $\hat{\mathbf{O}}$ and \mathbf{O}' . It encourages them to be recognizable as their corresponding categories. The auxiliary classification loss is defined as:

$$\mathcal{L}_{\text{AC}}^{\text{obj}} = \mathbb{E}_{\hat{\mathbf{O}},c} [-\log D_{\text{obj}}(c|\hat{\mathbf{O}})] + \mathbb{E}_{\mathbf{O}',c} [-\log D_{\text{obj}}(c|\mathbf{O}')], \quad (6)$$

where c is the object class label.

Therefore, the final loss function of our model is defined as:

$$\begin{aligned} \mathcal{L} = & \lambda_1 \mathcal{L}_{\text{KL}} + \lambda_2 \mathcal{L}_1^{\text{img}} + \lambda_3 \mathcal{L}_1^{\text{latent}} + \lambda_4 \mathcal{L}_{\text{adv}}^{\text{img}} \\ & + \lambda_5 \mathcal{L}_{\text{adv}}^{\text{obj}} + \lambda_6 \mathcal{L}_{\text{AC}}^{\text{obj}}, \end{aligned} \quad (7)$$

where $\lambda_1 \sim \lambda_6$ are the parameters balancing different losses.

3.9 Implementation Details

We use SN-GAN (Miyato et al. 2018) for stable training. Batch normalization (Ioffe and Szegedy 2015) and ReLU are used in the object encoder, image decoder, and only ReLU is used in the discriminators (no batch normalization). Conditional batch normalization (de Vries et al. 2017) is used in the object estimator to better normalize the object feature map according to its category. After object fuser, we use six

Table 1 Statistics of COCO-Stuff and Visual Genome dataset

Dataset	Train	Val.	Test	# Obj.	# Obj. in Image
COCO (Caesar et al. 2016)	24,972	1024	2048	171	3–8
VG (Krishna et al. 2017)	62,565	5506	5088	178	3–30

residual blocks (He et al. 2016) to further refine the hidden image feature maps. We set both m and n to 64. The image and crop size are set to 64×64 and 32×32 , respectively. The $\lambda_1 \sim \lambda_6$ are set to 0.01, 1, 10, 1, 1 and 1 respectively.

We train all models using Adam (Kingma and Ba 2014) with learning rate of 0.0001 and batch size of 8 for 300,000 iterations; training takes about 3 days on a single Titan Xp GPU. Full details about our architecture can be found in “Appendix”, and code will be made publicly available.

4 Experiments

Extensive experiments are conducted to evaluate the proposed Layout2Im network. We first compare our proposed method and the proposed OWA extension with previous state-of-the-art models for scene image synthesis, and show its superiority in aspects of realism, recognition and diversity. Finally, the contributions of each loss for training our model are studied through ablation.

4.1 Datasets

The same as previous scene image generation method (Johnson et al. 2018), we evaluate our proposed model on the COCO-Stuff (Caesar et al. 2016) and Visual Genome (Krishna et al. 2017) datasets. We preprocess and split the two datasets the same as that in Johnson et al. (2018). Table 1 lists the datasets statistics. Each image in these datasets has multiple bounding boxes annotations with labels for the objects.

4.2 Baselines

We compare our approach with several state-of-the-art methods.

pix2pix (Isola et al. 2017) translates images between two domains. In this paper, we define the input domain as feature maps constructed from layout \mathbf{L} , and set the real images as the output domain. We construct the input feature map with the size of $C \times H \times W$ for each layout \mathbf{L} , where C is the number of object categories, $H \times W$ is the image size. A bounding box \mathbf{O}_i with label y_i will set the corresponding region within c -th channel (the channel for category y_i) of the feature map to 1 and others are all 0. The pix2pix model is learned to

translate the generated feature maps to real images.

BicycleGAN (Zhu et al. 2017b) models a distribution of possible outputs in a conditional generative modeling setting when a single input may correspond to multiple possible outputs. The ambiguity of the mapping is represented as a low-dimensional latent vector, which will be combined with the input, and be translated into the output. It explicitly encourages the connection between output and the latent code to be invertible, contributing to generating diversified translated images. In our paper, we construct the input the same as pix2pix, and generate real images.

sg2im (Johnson et al. 2018) is originally trained to generate images from scene graphs. However, it can also generate images from layout, simply replacing the predicted layout with ground truth layout. We list the Inception Score of sg2im using ground truth layouts as reported in their paper, and generate the results for other comparisons using their released model trained with ground truth layout. In other words, the input and training data for our and sg2im models is identical.

pix2pixHD (Wang et al. 2017) produces realistic images from given semantic label maps. It uses multi-scale patch wise discriminator and multi-scale generator to generate high resolution images. To manipulate object with different input style vectors, they use an encoder-decoder to generate latent vectors at each spatial location and perform instance-wise average pooling for each instance to get the style vector. Then it can be used to control style of different objects, such as colors. As only coarse layouts are provided, we use the same setting as pix2pix where the input is semantic map translated from layouts.

GauGAN (Park et al. 2019) is proposed to synthesize high-resolution images with realistic details. They propose that the original batch normalization will wash away semantic information during each layer, so spatial adaptive normalization is proposed to alleviate this problem. Different from previous conditional normalization layers, their proposed normalization layer applies a spatially varying affine transformation, making it suitable for image synthesis from spatially-varying semantic mask. We use the same setting as pix2pix and input is converted from layouts.

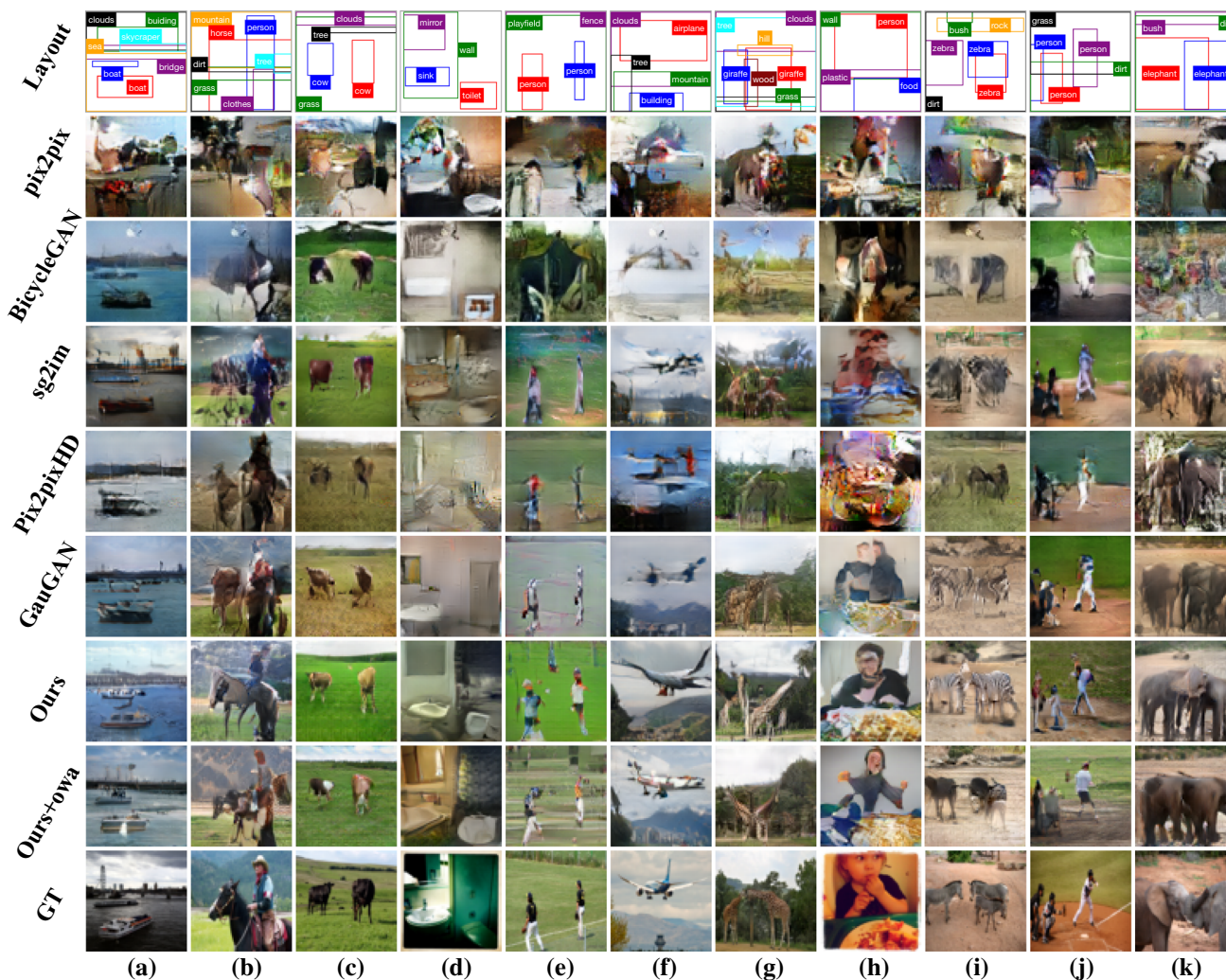


Fig. 7 Examples of 64×64 generated images from complex layouts on COCO-Stuff by our proposed method and baselines. For each example, we show the input layout, images generated by pix2pix, BicycleGAN, sg2im, pix2pixHD, GauGAN, results of our method, results of our

method with object wise attention added and the ground truth image in the datasets. Please zoom in to see the category of each object. Better view in color

4.3 Evaluation Metrics

Plausible images generated from layout should meet three requirements: be realistic, recognizable and diverse. Therefore we choose four different metrics, Inception Score (IS) (Salimans et al. 2016), Fréchet Inception Distance (FID) (Heusel et al. 2017), Object Classification Accuracy (Accu.) and Diversity Score (DS) (Zhang et al. 2018).

Inception Score (Salimans et al. 2016) is adopted to measure the quality, as well as diversity, of generated images. In our paper, we use the pre-trained VGG-net (Simonyan and Zisserman 2014) as the base model to compute the inception scores for our model and the baselines.

Fréchet Inception Distance (Heusel et al. 2017) uses 2nd order information of the final layer of the inception model, and calculates the similarity of generated images to real ones. Fréchet Inception Distance is more robust to noise than Inception Score.

Classification Accuracy measures the ability to generate recognizable objects, which is an important criteria for our task. We first train a ResNet-101 model (He et al. 2016) to classify objects. This is done using the real objects cropped and resized from ground truth images in the training set of



Fig. 8 Examples of 64×64 generated images from complex layouts on Visual Genome by our proposed method and baselines. For each example, we show the input layout, images generated by pix2pix, BicycleGAN, sg2im, pix2pixHD, GauGAN, results of our method, results

of our method with object wise attention added, the ground truth image in the datasets. Please zoom in to see the category of each object. Better view in color

each dataset. We then compute and report the object classification accuracy for objects in the generated images.

Diversity Score computes the perceptual similarity between two images in deep feature space. Different from the inception score which reflects the diversity across the entire generated images, diversity score measures the difference of a pair of images generated from the same input. We use the LPIPS metric (Zhang et al. 2018) for diversity score, and use AlexNet (Krizhevsky et al. 2012) for feature extraction as suggested in the paper.

4.4 Qualitative Results

Figures 7 and 8 show generated images using our method, as well as baselines on COCO and Visual Genome datasets

respectively. From these examples it is clear that our method can generate complex images with multiple objects, and even multiple instances of the same object type. For example, Fig. 7(a) shows two boats, (c) shows two cows, (e) and Fig. 8(r) contain two people. More qualitative results of our model of COCO and Visual Genome dataset are demonstrated in Figs. 9 and 10, respectively. These examples also show that our method generates images which respect the location constraints of the input bounding boxes, and the generated objects in the image are also recognizable and consistent with their input labels.

As we can see in Figs. 7 and 8, pix2pix fails to generate meaningful images, due to the extreme difficulty of directly mapping layout to a real image without detailed instance segmentation. Pix2PixHD and GauGAN provide more mean-

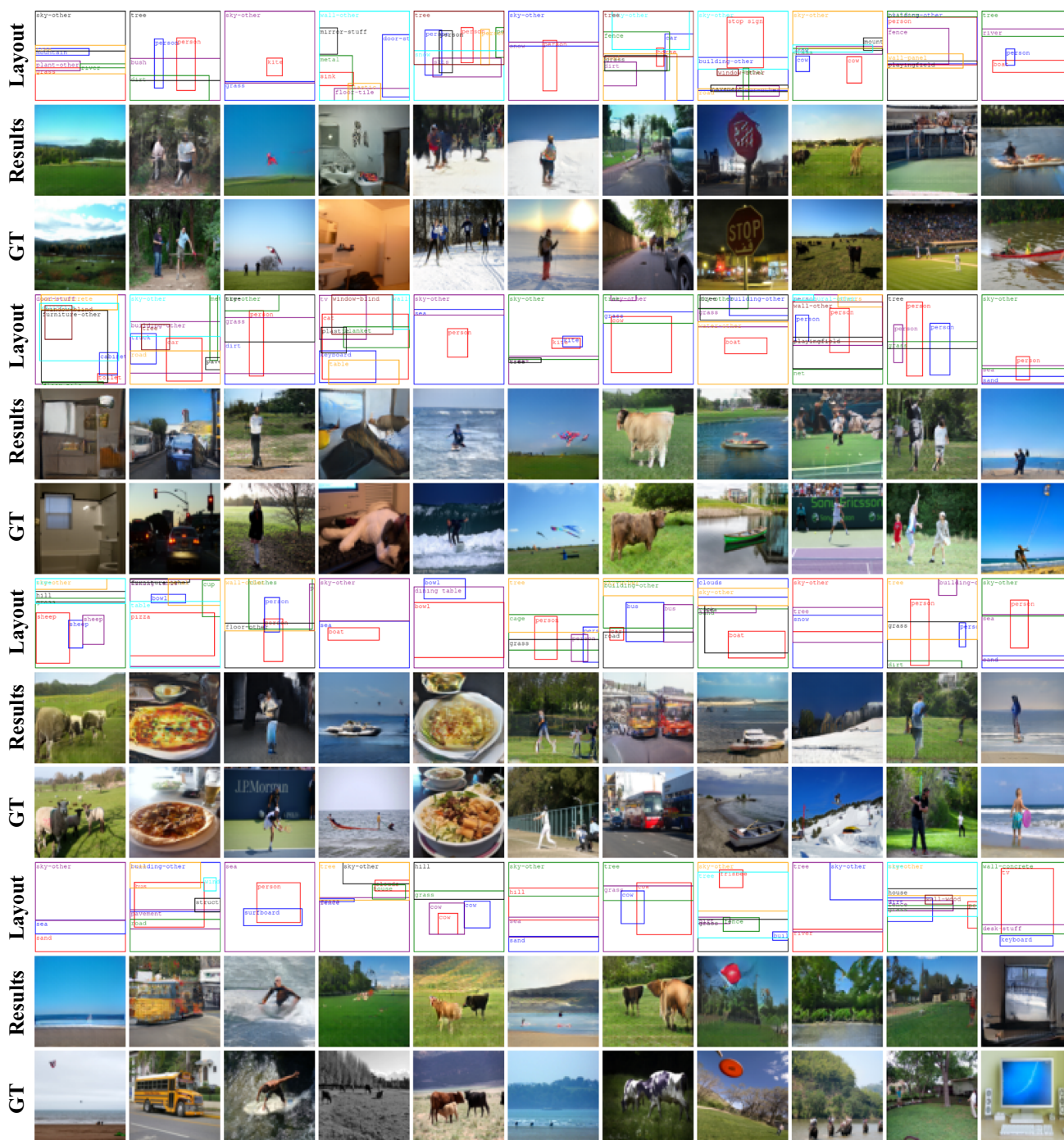


Fig. 9 More Qualitative Examples on COCO Dataset. Please zoom in to see the categories. Best viewed in color

ingful outputs but the shape of some object does not look realistic due to the lack of correct segmentation. For example in Fig. 7(g) the giraffe looks like a rectangle. The results generated by BicycleGAN and sg2im are also not as good as ours. For example, in Fig. 7(g), (i), the generated giraffe and zebra are difficult to recognize, and (l) contains lots of artifacts, making result look unrealistic. We also provide the result of OWA extension. The overall quality is better than

our original method and the shape of each object is more reasonable. As is shown in Fig. 7(f), the shape of airplane looks more reasonable thanks to the attention representation, which captures the shape of each object better.

In Fig. 11 we demonstrate our model’s ability to generate complex images by starting with simple layout and progressively adding new bounding box or moving existing bounding box, e.g., (g) and (k), to build/manipulate a

Table 2 Performance on COCO and VG in Inception Score (IS)

Method	COCO	VG
Real Images (64×64)	16.3 ± 0.4	13.9 ± 0.5
pix2pix (Isola et al. 2017)	3.5 ± 0.1	2.7 ± 0.02
BicycleGAN (Zhu et al. 2017b)	6.4 ± 0.1	5.6 ± 0.1
sg2im (GT Layout) (Johnson et al. 2018)	7.3 ± 0.1	6.3 ± 0.2
pix2pixHD (Wang et al. 2017)	4.7 ± 0.1	4.5 ± 0.1
GauGAN (Park et al. 2019)	8.3 ± 0.2	6.4 ± 0.2
Ours	9.1 ± 0.1	8.1 ± 0.1
Ours + OWA	9.7 ± 0.1	8.0 ± 0.2

Numbers in bold means best performance

The output size of all methods is 64×64 . We train the pix2pix, pix2pixhd and GauGAN from scratch by feeding layout, and generate image from the released sg2im model using ground truth layout

Table 3 Performance on COCO and VG in Fréchet Inception Distance (FID)

Method	COCO	VG
pix2pix (Isola et al. 2017)	121.97	142.86
BicycleGAN (Zhu et al. 2017b)	82.48	80.35
sg2im (GT Layout) (Johnson et al. 2018)	67.96	74.61
pix2pixHD (Wang et al. 2017)	121.21	83.12
GauGAN (Park et al. 2019)	49.27	47.74
Ours	44.19	39.68
Ours + OWA	40.19	33.54

Numbers in bold means best performance

The output size of all methods is 64×64 . We train the pix2pix, pix2pixhd and GauGAN from scratch by feeding layout, and generate image from the released sg2im model using ground truth layout

Our proposed method significantly outperforms baselines in all the evaluation metrics except Diversity Score. In terms of Inception Score and Fréchet Inception Distance, our method outperforms the existing approaches with a substantial margin, presumably because our method generates more recognizable objects as proved by object classification accuracy. By adding the object-wise attention mechanism, we can see a boost of Inception score on COCO dataset, and comparable performance on VG dataset. The FID scores, as well as the object classification accuracy are increased on both datasets with OWA. Please note that the object accuracy on real images is not the upper bound of object classification accuracy, since the object cannot be classified correctly in a real image does not necessarily mean it is also difficult to distinguish in a generated image. Pix2pix are deterministic so the diversity score is zero. We perform multimodal

Table 4 Performance on COCO and VG in Object Classification Accuracy

Method	COCO	VG
Real Images (64×64)	55.16	49.13
pix2pix (Isola et al. 2017)	12.06	9.20
BicycleGAN (Zhu et al. 2017b)	14.20	11.65
sg2im (GT Layout) (Johnson et al. 2018)	30.04	40.29
pix2pixHD (Wang et al. 2017)	21.26	22.59
GauGAN (Park et al. 2019)	35.82	34.43
Ours	50.84	48.09
Ours + OWA	54.95	51.49

Numbers in bold means best performance

The output size of all methods is 64×64 . We train the pix2pix, pix2pixhd and GauGAN from scratch by feeding layout, and generate image from the released sg2im model using ground truth layout

Table 5 Performance on COCO and VG in Diversity Score

Method	COCO	VG
pix2pix (Isola et al. 2017)	0	0
BicycleGAN (Zhu et al. 2017b)	0.33 ± 0.14	0.35 ± 0.12
sg2im (GT Layout) (Johnson et al. 2018)	0.02 ± 0.01	0.15 ± 0.12
pix2pixHD (Wang et al. 2017)	0.24 ± 0.10	0.26 ± 0.12
GauGAN (Park et al. 2019)	0.18 ± 0.06	0.18 ± 0.11
Ours	0.15 ± 0.06	0.17 ± 0.09
Ours + OWA	0.09 ± 0.05	0.09 ± 0.11

Numbers in bold means best performance

The output size of all methods is 64×64 . We generate image from the released sg2im model using ground truth layout. The other baselines do not provide diversity sampling so we do not compare with

sampling for pix2pixHD. Though the diversity is high as shown in Table 5, the image quality still low with different style codes because during training time the network cannot learn meaningful style code due to inaccurate correspondence between bounding boxes and ground truth image. We can sample diverse images from GauGAN by using VAE to provide global style information, but we cannot control style for each object with this global style code. Since BicycleGAN explicitly pursues the diversity of results, it can generate more diverse results, and has the highest diversity score. However, the generated objects of BicycleGAN are hard to recognize, as shown in the Figs. 7 and 8, which leads to poor performance on the rest evaluation metrics. By adding global noise to scene layout, sg2im can generate images with limited diversity. The diversity performance shows that our method can generate diverse results from the same layout. A

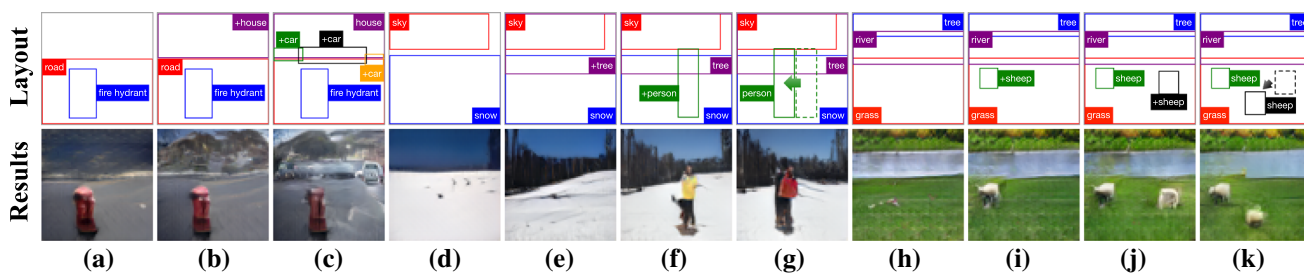


Fig. 11 Example of generated images by adding or moving bounding boxes based on previous layout. Three groups of images, (a)–(c), (d)–(g) and (h)–(k), are shown. In (g) and (k), original bounding boxes are drawn in dash. Please zoom in to see the category of each object

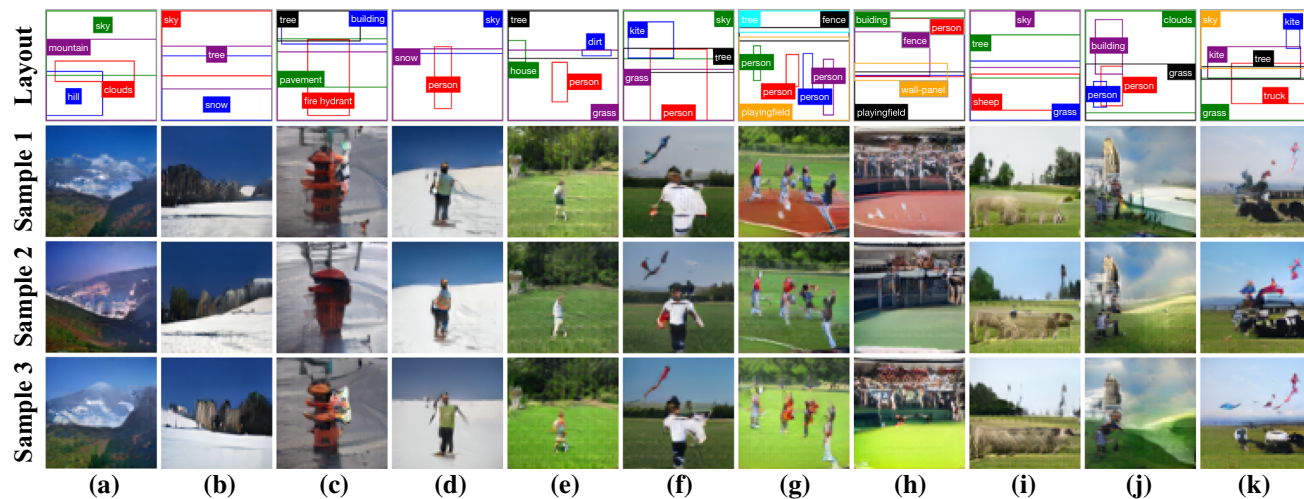


Fig. 12 Examples of diverse images generated from same layouts. For each layout, we sample 3 images. The generated images have different appearances, but sharing the same layout. Please zoom in to see the category of each object

very notable improvement is on COCO, where we achieve diversity score of 0.15 as compared to 0.02 for sg2im.

4.6 Ablation Study

We demonstrate the necessity of all components of our model by comparing the inception score, object classification accuracy, and diversity score of several ablated versions of our model trained on Visual Genome dataset:

- **w/o \mathcal{L}_1^{img}** reconstructs ground truth images without pixel regression;
- **w/o \mathcal{L}_1^{latent}** does not regress the latent codes which are used to generated objects in the result images;
- **w/o \mathcal{L}_{AC}^{obj}** does not classify the category of objects;
- **w/o \mathcal{L}_{adv}^{img}** removes the object adversarial loss when training the model;
- **w/o \mathcal{L}_{adv}^{obj}** removes the image adversarial loss when training the model.
- **w/o I'** removes the path which generates I' from prior distribution.

- **w/o \mathcal{L}_1^{latent} and \mathcal{L}_{KL}** removes both the KL loss and latent code regression loss.
- **w/o cLSTM** replaces the cLSTM in the object fusion module with simple object feature maps summation.
- **w/o \mathcal{L}_1^{latent} , \mathcal{L}_{KL} and cLSTM** removes both the KL loss and latent code regression loss. It also replaces the cLSTM in the object fusion module with simple object feature maps summation.

As shown in Table 6, removing any loss term will decrease the overall performance. Specifically, The model trained without \mathcal{L}_1^{img} or \mathcal{L}_1^{latent} generates less realistic images, which decreases the inception score. The object classification accuracy is still high because of the object classification loss. Without the constraint on reconstructed images or latent codes, the models get lower inception scores, but similar diversity scores. Removing the object classification loss degrade the inception score and object classification accuracy significantly, since the model cannot generate recognizable objects. Not surprisingly, this freedom results in higher diversity score. It is expected to see that removing the adversarial loss on image or object will decrease the inception score

Table 6 Ablation study of our model on Visual Genome dataset by removing different objectives

Method	IS	FID	Accuracy	DS
w/o $\mathcal{L}_1^{\text{img}}$	7.6 ± 0.2	39.29	49.03	0.17 ± 0.09
w/o $\mathcal{L}_1^{\text{latent}}$	7.5 ± 0.1	36.81	48.90	0.16 ± 0.09
w/o $\mathcal{L}_{AC}^{\text{obj}}$	6.5 ± 0.1	49.74	10.06	0.37 ± 0.11
w/o $\mathcal{L}_{adv}^{\text{img}}$	7.1 ± 0.1	43.14	56.17	0.13 ± 0.09
w/o $\mathcal{L}_{adv}^{\text{obj}}$	7.3 ± 0.1	44.01	57.74	0.14 ± 0.09
w/o I'	7.2 ± 0.3	39.04	44.64	0.28 ± 0.08
w/o $\mathcal{L}_1^{\text{latent}}$ & \mathcal{L}_{kl}	7.9 ± 0.2	36.37	50.65	0.16 ± 0.09
w/o cLSTM	7.5 ± 0.1	44.32	11.54	0.12 ± 0.10
w/o $\mathcal{L}_1^{\text{latent}}$ & \mathcal{L}_{kl} & cLSTM	7.9 ± 0.2	41.20	11.12	0.10 ± 0.11
Full model	8.1 ± 0.1	31.25	48.09	0.17 ± 0.09

Numbers in bold means best performance

IS is the inception score, Accuracy is the object classification accuracy, and DS is the diversity score

substantially. However, the object classification accuracy increases further comparing to the full model. We believe that without the realism requirement of image or object, the object classification loss could be tampered with adversarial attack. Without explicitly sampling images from prior distribution, the model is only learned from real data. A random latent code can not yield realistic images at test time since the KL loss may not be well optimized during training. Without both KL divergence loss and latent code regression, the model is not enforced to encode the appearance of different objects into prior distribution, and cannot generate diversified object appearance from prior distribution. It results to a drop on both inception score and FID. The performance drops in all four metrics when replacing the cLSTM in the Object Feature Maps Fusion module with simple feature maps summation, especially the object classification accuracy. Since object bonding boxes usually have overlaps, adding different object vectors together in the overlap region may confuse the image decoder to generate recognizable objects. We observe similar performance decrease in object classification accuracy when we remove both KL divergence loss and latent code regression, and replace cLSTM with summation. Trained with all the losses, our full model achieves a good balance across all four metrics.

5 Conclusion

In this paper we have introduced an end-to-end method for generating diverse images from layout (bounding boxes + categories). Our method can generate reasonable images which look realistic and contain recognizable objects at the desired locations. We also showed that we can control the

image generation process by adding/moving objects in the layout easily. To further improve the overall visual quality, we proposed an extension of our method called object wise attention, which helps the network model shape of different classes. Qualitative and quantitative results on COCO-Stuff (Caesar et al. 2016) and Visual Genome (Krishna et al. 2017) datasets demonstrated our model's ability to generate realistic complex images. Generating high resolution images from layouts will be our future work. Moreover, making the image generation process more controllable, such as specifying the fine-grained attributes of instances, would be an interesting future direction.

Acknowledgements This research was supported, in part, by NSERC Discovery, NSERC DAS and NSERC CFI grants. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

Appendix

Network Architecture

Here, we describe the detailed network architecture of all our model components in Tables 7, 8, 9, 10, 11 and 12. Here are some notations: CONV: convolutional layers; DECONV: transposed convolutional layers; FC: fully connected layer; CLSTM: convolutional LSTM; AVGPOOL: average pooling layer; CBN: conditional batch normalization; BN: batch normalization; ReLU: rectified linear unit; SUM: summation of feature maps along H & W axis; N: the number of output channels; K: kernel size; S: stride size; P: padding size.

Table 7 Architecture of object estimator

Index	Inputs	Operation	Output shape
(1)	–	Object Crops $\{\mathbf{O}_i\}$	$3 \times 32 \times 32$
(2)	(1)	CONV-(N64, K7, S1, P3), CBN, ReLU	$64 \times 32 \times 32$
(3)	(2)	CONV-(N128, K4, S2, P1), CBN, ReLU	$128 \times 16 \times 16$
(4)	(3)	CONV-(N256, K4, S2, P1), CBN, ReLU	$256 \times 8 \times 8$
(5)	(4)	CONV-(N512, K4, S2, P1), CBN, ReLU	$512 \times 4 \times 4$
(6)	(5)	CONV-(N1024, K4, S2, P1), CBN, ReLU	$1024 \times 2 \times 2$
(7)	(6)	AVGPOOL-(K2)	1024
(8)	(7)	FC-(1024, 64)	64
(9)	(7)	FC-(1024, 64)	64

Table 8 Architecture of object encoder

Index	Inputs	Operation	Output shape
(1)	–	Object Feature Maps $\{\mathbf{F}_i\}$	$(64+64) \times 64 \times 64$
(2)	(1)	CONV-(N64, K1, S1, P1), CBN, ReLU	$64 \times 64 \times 64$
(3)	(2)	CONV-(N128, K4, S2, P1), CBN, ReLU	$128 \times 32 \times 32$
(4)	(3)	CONV-(N256, K4, S2, P1), CBN, ReLU	$256 \times 16 \times 16$
(5)	(4)	CONV-(N512, K4, S2, P1), CBN, ReLU	$512 \times 8 \times 8$

Table 9 Architecture of objects fuser

Index	Inputs	Operation	Output shape
(1)	–	Downsampled $\{\mathbf{F}_i\}$	$512 \times 8 \times 8$
(2)	(1)	CLSTM-(N128, K5, S1, P2)	$128 \times 8 \times 8$
(3)	(2)	CLSTM-(N64, K5, S1, P2)	$64 \times 8 \times 8$
(4)	(3)	CLSTM-(N64, K5, S1, P2)	$64 \times 8 \times 8$
(5)	(4)	Residual Block: CONV-(N64, K3, S1, P1), BN, ReLU	$64 \times 8 \times 8$
(6)	(5)	Residual Block: CONV-(N64, K3, S1, P1), BN, ReLU	$64 \times 8 \times 8$
(7)	(6)	Residual Block: CONV-(N64, K3, S1, P1), BN, ReLU	$64 \times 8 \times 8$
(8)	(7)	Residual Block: CONV-(N64, K3, S1, P1), BN, ReLU	$64 \times 8 \times 8$
(9)	(8)	Residual Block: CONV-(N64, K3, S1, P1), BN, ReLU	$64 \times 8 \times 8$
(10)	(9)	Residual Block: CONV-(N64, K3, S1, P1), BN, ReLU	$64 \times 8 \times 8$

Table 10 Architecture of image decoder

Index	Inputs	Operation	Output shape
(1)	–	Image Hidden Feature Map \mathbf{H}	$64 \times 8 \times 8$
(2)	(1)	CONV-(N256, K3, S1, P1), BN, ReLU	$256 \times 8 \times 8$
(3)	(2)	DECONV-(N256, K4, S2, P1), BN, ReLU	$256 \times 16 \times 16$
(4)	(3)	DECONV-(N128, K4, S2, P1), BN, ReLU	$128 \times 32 \times 32$
(5)	(4)	DECONV-(N64, K4, S2, P1), BN, ReLU	$64 \times 64 \times 64$
(6)	(5)	CONV-(N3, K7, S1, P3)	$3 \times 64 \times 64$

Table 11 Architecture of image discriminator

Index	Inputs	Operation	Output shape
(1)	–	Ground Truth Image \mathbf{I} , Generated Image $\hat{\mathbf{I}}$ & \mathbf{I}'	$3 \times 64 \times 64$
(2)	(1)	Residual Block: CONV-(N64, K3, S1, P1), ReLU, AVGPOOL-(K2)	$64 \times 32 \times 32$
(3)	(2)	Residual Block: CONV-(N128, K3, S1, P1), ReLU, AVGPOOL-(K2)	$128 \times 16 \times 16$
(4)	(3)	Residual Block: CONV-(N256, K3, S1, P1), ReLU, AVGPOOL-(K2)	$256 \times 8 \times 8$
(5)	(4)	Residual Block: CONV-(N512, K3, S1, P1), ReLU, AVGPOOL-(K2)	$512 \times 4 \times 4$
(6)	(5)	Residual Block: CONV-(N1024, K3, S1, P1), ReLU, AVGPOOL-(K2)	$1024 \times 2 \times 2$
(7)	(6)	SUM-(K2)	1024
(8)	(7)	FC-(1024, 1)	1

Table 12 Architecture of object discriminator. C is the number of object categories

Index	Inputs	Operation	Output shape
(1)	–	Ground Truth Objects \mathbf{O} , Generated Objects $\hat{\mathbf{O}}$ & \mathbf{O}'	$3 \times 32 \times 32$
(2)	(1)	Residual Block: CONV-(N64, K3, S1, P1), ReLU	$64 \times 32 \times 32$
(3)	(2)	Residual Block: CONV-(N128, K3, S1, P1), ReLU, AVGPOOL-(K2)	$128 \times 16 \times 16$
(4)	(3)	Residual Block: CONV-(N256, K3, S1, P1), ReLU, AVGPOOL-(K2)	$256 \times 8 \times 8$
(5)	(4)	Residual Block: CONV-(N512, K3, S1, P1), ReLU, AVGPOOL-(K2)	$512 \times 4 \times 4$
(6)	(5)	Residual Block: CONV-(N1024, K3, S1, P1), ReLU, AVGPOOL-(K2)	$1024 \times 2 \times 2$
(7)	(6)	SUM-(K2)	1024
(8)	(7)	FC-(1024, 1)	1
(9)	(7)	FC-(1024, C)	C

References

- Caesar, H., Uijlings, J., & Ferrari, V. (2016). Coco-stuff: Thing and stuff classes in context. [arXiv: 1612.03716](https://arxiv.org/abs/1612.03716).
- Chen, X., Duan, Y., Houthoofd, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*.
- Cheung, B., Livezey, J. A., Bansal, A. K., & Olshausen, B.A. (2015). Discovering hidden factors of variation in deep networks. In *ICLR workshop*.
- de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., & Courville, A. (2017). Modulating early visual processing by language. In *NIPS*.
- Denton, E., & Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. In *NIPS*.
- Dosovitskiy, A., Tobias Springenberg, J., & Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *CVPR*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.
- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS*.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computing*, 9(8), 1735–1780.
- Hong, S., Yang, D., Choi, J., & Lee, H. (2018). Inferring semantic layout for hierarchical text-to-image synthesis. In *CVPR*.
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. [arXiv:1502.03167](https://arxiv.org/abs/1502.03167).
- Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *CVPR*.
- Johnson, J., Gupta, A., & Fei-Fei, L. (2018). Image generation from scene graphs. In *CVPR*.
- Karacan, L., Akata, Z., Erdem, A., & Erdem, E. (2016). Learning to generate images of outdoor scenes from attributes and semantic layouts. [arXiv:1612.00215](https://arxiv.org/abs/1612.00215).
- Kim, J. H., Parikh, D., Batra, D., Zhang, B. T., & Tian, Y. (2017). Codraw: visual dialog for collaborative drawing. [arXiv:1712.05558](https://arxiv.org/abs/1712.05558).
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *ICLR*.
- Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L. J., & Shamma, D. A., et al. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *IJCV*.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.
- Lai, W. S., Huang, J. B., Ahuja, N., & Yang, M. H. (2017). Deep Laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*.
- Lee, H. Y., Tseng, H. Y., Huang, J. B., Singh, M., & Yang, M. H. (2018). Diverse image-to-image translation via disentangled representations. In *ECCV*.
- Liu, M. Y., Breuel, T., & Kautz, J. (2017). Unsupervised image-to-image translation networks. In *NIPS*.
- Ma, L., Sun, Q., Georgoulis, S., Gool, L. V., Schiele, B., & Fritz, M. (2018a). Disentangled person image generation. In *CVPR*.
- Ma, L., Sun, Q., Georgoulis, S., Van Gool, L., Schiele, B., & Fritz, M. (2018b). Disentangled person image generation. In *IEEE conference on computer vision and pattern recognition*.

- Mansimov, E., Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2015). Generating images from captions with attention. [arXiv:1511.02793](#).
- Mathieu, M., Zhao, J., Sprechmann, P., Ramesh, A., & LeCun, Y. (2016). Disentangling factors of variation in deep representations using adversarial training. In *NIPS*.
- Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. [arXiv:1411.1784](#).
- Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *ICLR*.
- Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R., & Kim, K. (2018). Image to image translation for domain adaptation. In *CVPR*.
- Nilsback, M., & Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Indian conference on computer vision, graphics image processing*.
- Oord, A. v. d., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. [arXiv:1601.06759](#).
- Park, T., Liu, M. Y., Wang, T. C., & Zhu, J. Y. (2019). GauGAN: semantic image synthesis with spatially adaptive normalization. In *SIGGRAPH '19*.
- Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). Context encoders: feature learning by inpainting. In *CVPR*.
- Reed, S., Oord, A. v. d., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Belov, D., & de Freitas, N. (2017). Parallel multiscale autoregressive density estimation. [arXiv:1703.03664](#).
- Reed, S. E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., & Lee, H. (2016). Learning what and where to draw. In *NIPS*.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. In *NIPS*.
- Sangkloy, P., Lu, J., Fang, C., Yu, F., & Hays, J. (2017). Scribbler: Controlling deep image synthesis with sketch and color. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Sharma, S., Suhubdy, D., Michalski, V., Kahou, S. E., & Bengio, Y. (2018). ChatPainter: Improving text to image generation using dialogue. [arXiv:1802.08216](#).
- Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In *NIPS*.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. [arXiv:1409.1556](#).
- Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *NIPS*.
- Tan, F., Feng, S., & Ordonez, V. (2018). Text2scene: generating abstract scenes from textual descriptions. [arXiv:1809.01110](#).
- van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., & Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *NIPS*.
- Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A., Kautz, J., & Catanzaro, B. (2017). High-resolution image synthesis and semantic manipulation with conditional GANs. In *2018 IEEE/CVF conference on computer vision and pattern recognition* (pp. 8798–8807).
- Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A., Kautz, J., & Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8798–8807).
- Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., & Perona, P. (2010). Caltech-UCSD birds 200. Technical Report, CNS-TR-2010-001, California Institute of Technology.
- Xian, W., Sangkloy, P., Agrawal, V., Raj, A., Lu, J., Fang, C., Yu, F., & Hays, J. (2018). TextureGAN: Controlling deep image synthesis with texture patches. In *CVPR*.
- Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., & Li, H. (2017). High-resolution image inpainting using multi-scale neural patch synthesis. In *CVPR*.
- Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., & Metaxas, D. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*.
- Zhang, W., Sun, J., & Tang, X. (2008). Cat head detection—How to effectively exploit shape and texture features. In *ECCV*.
- Zhao, B., Chang, B., Jie, Z., & Sigal, L. (2018). Modular generative adversarial networks. In *ECCV*.
- Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017a). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*.
- Zhu, J. Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., & Shechtman, E. (2017b). Toward multimodal image-to-image translation. In *NIPS*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.