Image Generation from Layout

Bo Zhao Lili Meng Weidong Yin Leonid Sigal
University of British Columbia Vector Institute
{bzhao03, menglili, wdyin, lsigal}@cs.ubc.ca

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.

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 specification?

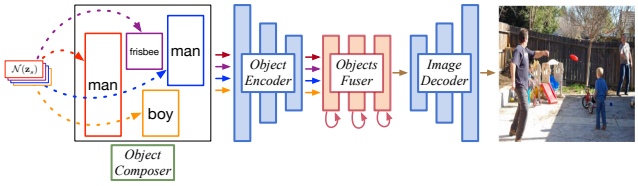

Figure 1. **Image generation from layout.** Given the coarse layout (bounding boxes + object categories), the proposed Layout2Im model samples the appearance of each object from a normal distribution, and transforms these inputs into a real image by a serial of components. Please refer to Section 3 for a detailed explanation.

For these reasons, image generation algorithms have been a major focus of recent research. Of specific relevance are approaches for text-to-image [11, 15, 25, 33, 41, 47] 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 [49], birds [44] or flowers [30]. Generating complex, real-world images such as those in COCO-Stuff [1] and Visual Genome [19] datasets remains a challenge; (ii) the ambiguity of textual description 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 [14] 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 Figure 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 ob-

jects 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 Figure 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: (1) We propose a novel approach for generating images from coarse layout (bounding boxes + object categories). This provides a flexible control mechanism for image generation. (2) 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. (3) We show qualitative and quantitative results on COCO-Stuff [1] and Visual Genome [19] datasets, demonstrating our model's ability to generate complex images with respect to object categories and their layout (without access to segmentation masks [11, 14]). We also preform comprehensive ablations to validate each component in our approach.

2. Related Work

Conditional Image Generation. Conditional image generation approaches generate images conditioned on additional input information, including entire source image [13, 23, 32, 46, 50, 51, 52], sketches [13, 36, 43, 45, 52], scene graphs [14], dialogues [16, 37] and text descriptions [25, 33, 41, 47]. Variational Autoencoders (VAEs) [18, 25, 40], autoregressive models [31, 42] and GANs [13, 27, 43, 51] are powerful tools for conditional image generation and have shown promising results. However, many previouse generative models [13, 32, 36, 45, 46, 51] tend to largely ignore the random noise vector when conditioning on the same relevant context, making the generated images very similar to each other. By enforcing the bijection mapping between the latent and target space, BicycleGAN [52] pursues the diversity of generated images from a same input. Inspired by this idea, in our paper, we also explicitly regress the latent codes which are used to generate the different objects.

Image Generation from Layout. The use of layout in image generation is a relative novel task. In prior works, it is usually served as an intermediate representation between other input sources (*e.g.*, text [11] or scene graphs [14]) and the output images, or as a complementary feature for image generation based on context (*e.g.*, text [15, 34, 41], shape and lighting [6]). In [11, 14], 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 [11] 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 and labor-intensive. Different from [11] and [14], we use the coarse layout without instance segmentation mask as a fundamental input modality for diverse image generation.

Disentangled Representations. Many papers [2, 3, 5, 21, 22, 24, 26, 29] 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 [3, 21, 22, 26, 29]. By manipulating the disentangled representations, images with different appearances can be generated easily. In [24], three factors (foreground, background and pose) are disentangled explicitly when generating person image. InfoGAN [2], DrNet [5] and DRIT [22] learn the disentangled representations in an unsupervised manner, either by maximizing the mutual information [2] or adversarial losses [5, 22]. In our work, we explicitly separate the representation of each object into a category-related and an

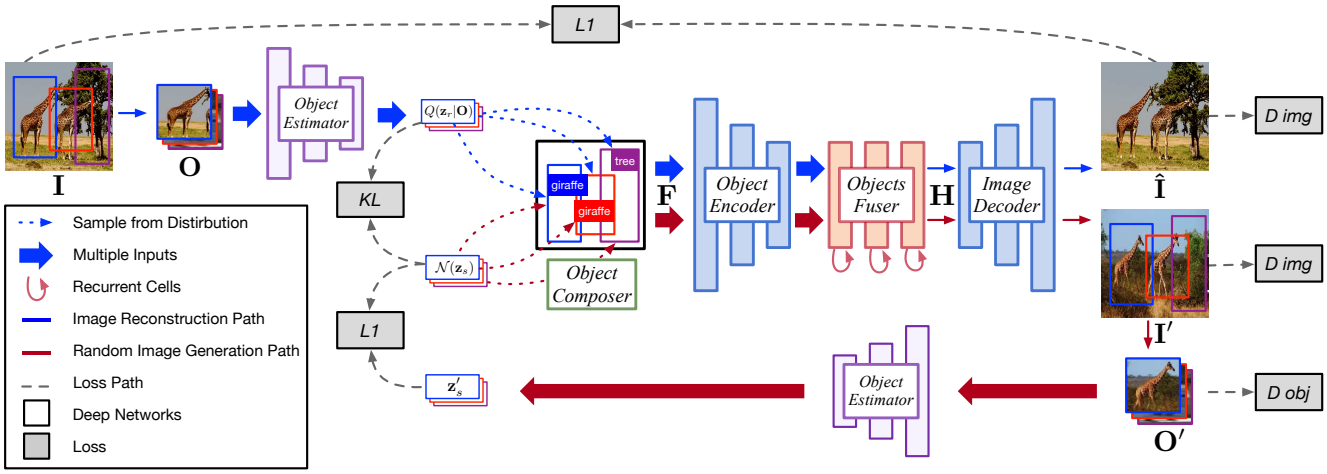

Figure 2. **Overview of our Layout2Im network** for generating images from layout during training. The inputs to the model are the ground truth image with its layout. The objects are first cropped from the input image according to their bounding boxes, and then processed with the object estimator to 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.

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 Figure 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 Figure 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 $\mathbf{z}_{ri} \sim Q(\mathbf{z}_{ri}|\mathbf{O}_i) = \mathcal{N}(\mu(\mathbf{O}_i), \sigma(\mathbf{O}_i))$. 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

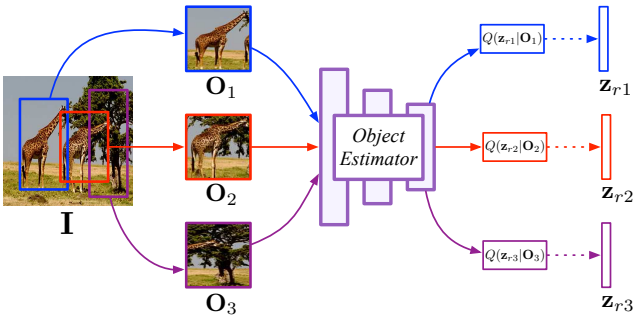

Figure 3. **Object latent code estimation.** Given the input image and its layout, the objects are first cropped and resized from the input image. Then the object estimator predicts a distribution for each object from the object crops, and multiple latent codes are sampled from the estimated distribution.

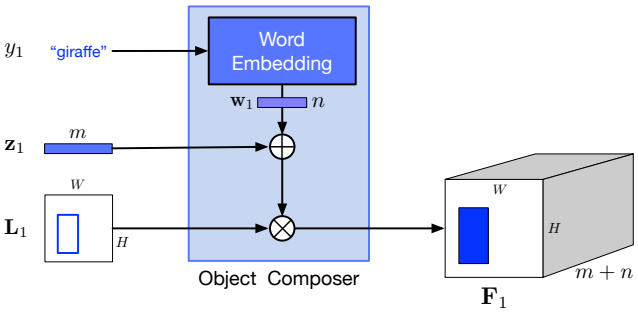

Figure 4. **Object feature map composition.** The object category is first encoded by a word embedding. Then the object feature map is simply composed by filling the region within the object bounding box with the concatenation of category embedding and latent code. The rest of the feature map are all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

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 $< y_i, \mathbf{z}_i, \mathbf{L}_i >$ 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 $\mathbf{H}$.

3.3. Object Feature Maps Fusion

Since the result image will be decoded from it, a good hidden feature map $\mathbf{H}$ 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 implicitly 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 [38] to fuse the downsampled object feature maps $\mathbf{F}$. Different from the traditional LSTM [10], the hidden states and cell states in cLSTM are both feature maps rather than vectors. The computation of different gates are also done by 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. Please refer to the supplementary material for the structure of multi-layer cLSTM networks.

3.4. Image Decoder

Given the fused image hidden feature map $\mathbf{H}$, image decoder is tasked with generating a result image. As shown in Figure 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.

3.5. 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.

3.6. 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 x or y as real or fake by maximizing the objective [7]:

$$\mathcal{L}_{\text{GAN}} = \mathop{\mathbb{E}}_{x\sim p_{\text{real}}} \log D(x) + \mathop{\mathbb{E}}_{y\sim p_{\text{fake}}} \log(1 - D(y)), \quad (1)$$

where x represents the real images and y represents the generated ones. Meanwhile, the generator networks are trained to minimizing $\mathcal{L}_{\text{GAN}}$. 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.

3.7. 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** $\mathcal{L}_{\text{KL}} = \sum_{i=1}^{o} \mathbb{E}[\mathcal{D}_{\text{KL}}(Q(\mathbf{z}_{ri}|\mathbf{O}_i)||\mathcal{N}(\mathbf{z}_r))]$ computes the KL-Divergence between the distribution $Q(\mathbf{z}_r|\mathbf{O})$ and the normal distribution $\mathcal{N}(\mathbf{z}_r)$, where o is the number of objects in the image/layout.
- **Image Reconstruction Loss** $\mathcal{L}_1^{\text{img}} = ||\mathbf{I} - \hat{\mathbf{I}}||_1$ penalizes the $\mathcal{L}_1$ difference between ground-truth image $\mathbf{I}$ and reconstructed image $\hat{\mathbf{I}}$.
- **Object Latent Code Reconstruction Loss** $\mathcal{L}_1^{\text{latent}} = \sum_{i=1}^{o} ||\mathbf{z}_{si} - \mathbf{z}'_{si}||_1$ penalizes the $\mathcal{L}_1$ difference between the randomly sampled $\mathbf{z}_s \sim N(\mathbf{z}_s)$ and the re-estimated $\mathbf{z}'_s$ from the generated objects $\mathbf{O}'$.
- **Image Adversarial Loss** $\mathcal{L}_{\text{GAN}}^{\text{img}}$ is defined as in Eq. (1), where x is the ground truth image $\mathbf{I}$, y is the reconstructed image $\hat{\mathbf{I}}$ and sampled image $\mathbf{I}'$.
- **Object Adversarial Loss** $\mathcal{L}_{\text{GAN}}^{\text{obj}}$ is also defined as in Eq. (1), where x is the objects $\mathbf{O}$ cropped from the ground truth image $\mathbf{I}$, y are $\hat{\mathbf{O}}$ and $\mathbf{O}'$ cropped from the reconstructed image $\hat{\mathbf{I}}$ and sampled image $\mathbf{I}'$.
- **Auxiliar Classification Loss** $\mathcal{L}_{\text{AC}}^{\text{obj}}$ from D_{obj} encourages the generated objects $\hat{\mathbf{O}}_i$ and $\mathbf{O}'_i$ to be recognizable as their corresponding categories.

Dataset	Train	Val.	Test	# Obj.	# Obj. in Image
COCO [1]	24,972	1,024	2,048	171	$3 \sim 8$
VG [19]	62,565	5,506	5,088	178	$3 \sim 30$

Table 1. **Statistics of COCO-Stuff and Visual Genome dataset.**

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

$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{KL}} + \lambda_2\mathcal{L}_1^{\text{img}} + \lambda_3\mathcal{L}_1^{\text{latent}} + \lambda_4\mathcal{L}_{\text{adv}}^{\text{img}} + \lambda_5\mathcal{L}_{\text{adv}}^{\text{obj}} + \lambda_6\mathcal{L}_{\text{AC}}^{\text{obj}},$$

where, λ_i are the parameters balancing different losses.

3.8. Implementation Details

We use SN-GAN [28] for stable training. Batch normalization [12] and ReLU are used in the object encoder, image decoder, and only ReLU is used in the discriminators (no batch normalization). Conditional batch normalization [4] is used in the object estimator to better normalize the object feature map according to its category. After object fuser, we use six residual blocks [8] 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 [17] 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 the supplementary material, 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 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 [14], we evaluate our proposed model on the COCO-Stuff [1] and Visual Genome [19] datasets. We preprocess and split the two datasets the same as that in [14]. 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 two state-of-the-art methods: pix2pix [13] and sg2im [14].

pix2pix [13] 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

Figure 5. **Examples of 64 $\times$ 64 generated images from complex layouts** on COCO-Stuff (top) and Visual Genome Datasets (bottom) by our proposed method and baselines. For each example, we show the input layout, images generated by pix2pix, sg2im and our method. Please zoom in to see the category of each object. The ground truth images and more examples can be found in the supplementary material.

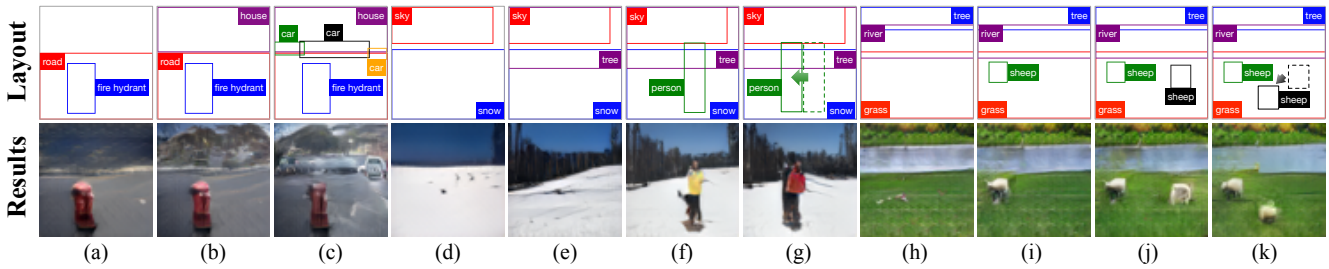

Figure 6. **Example of generated images by adding or moving bounding boxes based on previous layout.** Three groups of images, (a)-(c), (d)-(g) and (h)-(k), are shown. In (g) and (k), original bounding boxes are drawn in dash. Please zoom in to see the category of each object.

$C \times H \times W$ 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.

sg2im [14] 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

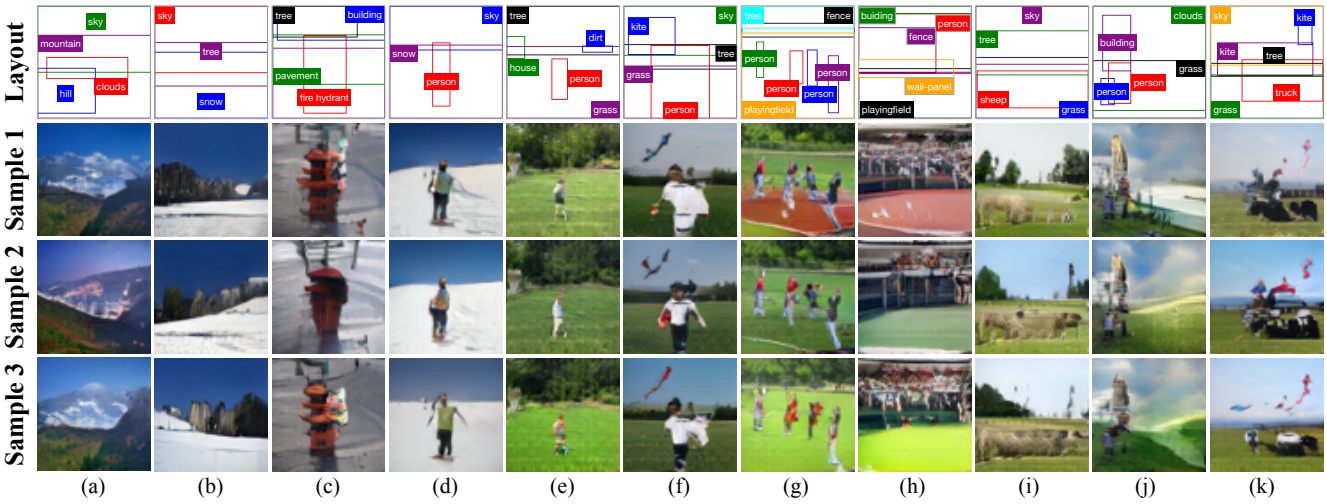

Figure 7. **Examples of diverse images generated from same layouts.** For each layout, we sample 3 images. The generated images have different appearances, but sharing the same layout. Please zoom in to see the category of each object.

trained with ground truth layout. In other words, the input and training data for our and sg2im models is identical.

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) [35], Fréchet Inception Distance (FID) [9], Object Classification Accuracy (Accu.) and Diversity Score (DS) [48].

Inception Score [35] is adopted to measure the quality, as well as diversity, of generated images. In our paper, we use the pre-trained VGG-net [39] as the base model to compute the inception scores for our model and the baselines.

Fréchet Inception Distance [9] 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 [8] to classify objects. This is done using the real objects cropped and resized from ground truth images in the training set of 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 [48] for diversity score, and use AlexNet [20] for feature extraction as suggested in the paper.

4.4. Qualitative results

Figure 5 shows generated images using our method, as well as baselines. 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, Figure 5(a) shows two boats, (c) shows two cows, (e) and (r) contain two people.

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 Figure 5, pix2pix fails to generate meaningful images, due to the extreme difficulty of directly mapping layout to a real image without detailed instance segmentation. The results generated by sg2im are also not as good as ours. For example, in Figure 5 (g) and (i), the generated giraffe and zebra are difficult to recognize, and (l) contains lots of artifacts, making result look unrealistic.

In Figure 6 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 complex image. From these examples we can see that new objects are drawn in the images at the desired locations, and existing objects are kept consistent as new content is added.

Figure 7 shows the diverse results generated from the same layouts. Given that the same layout may have many different possible real image realizations, the ability to sample diverse images is a key advantage of our model.

4.5. Quantitative results

Table 2 summarizes comparison results of the inception score, object classification accuracy and diversity score of baseline models and our model. We also report the incep-

Method	IS		FID		Accu.		DS	
	COCO	VG	COCO	VG	COCO	VG	COCO	VG
Real Images (64 × 64)	16.3 ± 0.4	13.9 ± 0.5	-	-	55.16	49.13	-	-
pix2pix [13]	3.5 ± 0.1	2.7 ± 0.02	121.97	142.86	12.06	9.20	0	0
sg2im (GT Layout) [14]	7.3 ± 0.1	6.3 ± 0.2	67.96	74.61	30.04	40.29	0.02 ± 0.01	0.15 ± 0.12
Ours	$\mathbf{9.1 \pm 0.1}$	$\mathbf{8.1 \pm 0.1}$	**38.14**	**31.25**	**50.84**	**48.09**	$\mathbf{0.15 \pm 0.06}$	$\mathbf{0.17 \pm 0.09}$

Table 2. **Performance on COCO and VG in Inception Score (IS), Fréchet Inception Distance (FID), Object Classification Accuracy (Accu.) and Diversity Score (DS).** The output size of all methods is 64 × 64. We train the pix2pix from scratch, and generate image from the released sg2im model using ground truth layout.

Method	IS	Accu.	DS
w/o $\mathcal{L}_1^{\text{img}}$	7.6 ± 0.2	49.03	0.17 ± 0.09
w/o $\mathcal{L}_1^{\text{latent}}$	7.5 ± 0.1	48.90	0.16 ± 0.09
w/o $\mathcal{L}_{\text{AC}}^{\text{obj}}$	6.5 ± 0.1	10.06	$\mathbf{0.37 \pm 0.11}$
w/o $\mathcal{L}_{\text{adv}}^{\text{img}}$	7.1 ± 0.1	56.17	0.13 ± 0.09
w/o $\mathcal{L}_{\text{adv}}^{\text{obj}}$	7.3 ± 0.1	**57.74**	0.14 ± 0.09
full model	$\mathbf{8.1 \pm 0.1}$	48.09	0.17 ± 0.09

Table 3. **Ablation study of our model** on Visual Genome dataset by removing different objectives. IS is the inception score, Accu. is the object classification accuracy, and DS is the diversity score.

tion score and object classification accuracy on real images.

The proposed method significantly outperforms baselines in all the three evaluation metrics. 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. 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. Since the pix2pix is deterministic, its diversity score is 0. 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 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** $\mathcal{L}_1^{\text{img}}$ reconstructs ground truth images without pixel regression.
- **w/o** $\mathcal{L}_1^{\text{latent}}$ does not regress the latent codes which are used to generated objects in the result images.
- **w/o** $\mathcal{L}_{\text{AC}}^{\text{obj}}$ does not classify the category of objects.
- **w/o** $\mathcal{L}_{\text{adv}}^{\text{img}}$ removes the object adversarial loss when training the model.
- **w/o** $\mathcal{L}_{\text{adv}}^{\text{obj}}$ removes the image adversarial loss when training the model.

As shown in Table 3, removing any loss term will decrease the overall performance. Specifically, The model trained without $\mathcal{L}_1^{\text{img}}$ or $\mathcal{L}_1^{\text{latent}}$ 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 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. Trained with all the losses, our full model achieves a good balance across all three 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. Qualitative and quantitative results on COCO-Stuff [1] and Visual Genome [19] 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.

Acknowledgement 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.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: thing and stuff classes in context. *arXiv: 1612.03716*, 2016. 1, 2, 5, 8

[2] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016. 2

[3] Brian Cheung, Jesse A. Livezey, Arjun K. Bansal, and Bruno A. Olshausen. Discovering hidden factors of variation in deep networks. In *ICLR workshop*, 2015. 2

[4] Harm de Vries, Florian Strub, Jeremie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. In *NIPS*, 2017. 5

[5] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017. 2

[6] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 2

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NIPS*, 2014. 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 7

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 7

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. 4

[11] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *CVPR*, 2018. 1, 2

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 5

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2, 5, 8

[14] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018. 1, 2, 5, 6, 8

[15] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv:1612.00215*, 2016. 1, 2

[16] Jin-Hwa Kim, Devi Parikh, Dhruv Batra, Byoung-Tak Zhang, and Yuandong Tian. Codraw: visual dialog for collaborative drawing. *arXiv:1712.05558*, 2017. 2

[17] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv:1412.6980*, 2014. 5

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[19] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017. 1, 2, 5, 8

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 7

[21] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 2

[22] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018. 2

[23] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017. 2

[24] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. Disentangled person image generation. In *CVPR*, 2018. 2

[25] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *arXiv:1511.02793*, 2015. 1, 2

[26] Michael Mathieu, Junbo Zhao, Pablo Sprechmann, Aditya Ramesh, and Yann LeCun. Disentangling factors of variation in deep representations using adversarial training. In *NIPS*, 2016. 2

[27] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014. 2

[28] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 5

[29] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *CVPR*, 2018. 2

[30] M. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics Image Processing*, 2008. 1

[31] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv:1601.06759*, 2016. 2

[32] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: feature learning by inpainting. In *CVPR*, 2016. 2

[33] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. *arXiv:1703.03664*, 2017. 1, 2

[34] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *NIPS*, 2016. 2

[35] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016. 7

[36] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: controlling deep image synthesis with sketch and color. In *CVPR*, 2017. 2

[37] Shikhar Sharma, Dendi Suhubdy, Vincent Michalski, Samira Ebrahimi Kahou, and Yoshua Bengio. Chatpainter: improving text to image generation using dialogue. *arXiv:1802.08216*, 2018. 2

[38] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: a machine learning approach for precipitation nowcasting. In *NIPS*, 2015. 4

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 7

[40] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015. 2

[41] Fuwen Tan, Song Feng, and Vicente Ordonez. Text2scene: generating abstract scenes from textual descriptions. *arXiv:1809.01110*, 2018. 1, 2

[42] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016. 2

[43] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv:1711.11585*, 2017. 2

[44] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 1

[45] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: controlling deep image synthesis with texture patches. In *CVPR*, 2018. 2

[46] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *CVPR*, 2017. 2

[47] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017. 1, 2

[48] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[49] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat head detection - how to effectively exploit shape and texture features. In *ECCV*, 2008. 1

[50] Bo Zhao, Bo Chang, Zequn Jie, and Leonid Sigal. Modular generative adversarial networks. In *ECCV*, 2018. 2

[51] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2

[52] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017. 2