Learning Layout and Style Reconfigurable GANs for Controllable Image Synthesis

Wei Sun and Tianfu Wu

Abstract—With the remarkable recent progress on learning deep generative models, it becomes increasingly interesting to develop models for controllable image synthesis from reconfigurable inputs. This paper focuses on a recent emerged task, layout-to-image, to learn generative models that are capable of synthesizing photo-realistic images from spatial layout (i.e., object bounding boxes configured in an image lattice) and style (i.e., structural and appearance variations encoded by latent vectors). This paper first proposes an intuitive paradigm for the task, layout-to-mask-to-image, to learn to unfold object masks of given bounding boxes in an input layout to bridge the gap between the input layout and synthesized images. Then, this paper presents a method built on Generative Adversarial Networks for the proposed layout-to-mask-to-image with style control at both image and mask levels. Object masks are learned from the input layout and iteratively refined along stages in the generator network. Style control at the image level is the same as in vanilla GANs, while style control at the object mask level is realized by a proposed novel feature normalization scheme, Instance-Sensitive and Layout-Aware Normalization. In experiments, the proposed method is tested in the COCO-Stuff dataset and the Visual Genome dataset with state-of-the-art performance obtained.

Index Terms—Image Synthesis; Layout-to-Image; Layout-to-Mask-to-Image; Deep Generative Learning; GAN; ISLA-Norm.

1 INTRODUCTION

1.1 Motivation and Objective

Remarkable recent progress has been made on both unconditional and conditional image synthesis [1], [2], [3], [4], [5], [6], [7], [8]. The former aims to generate high-fidelity images from some random latent codes/vectors (e.g., sampled from the standard multivariate Gaussian distribution). The latter needs to do so with given conditions satisfied in terms of some consistency metrics. The conditions may take many forms such as category labels [3], [9], paired or unpaired source images [10], [11], [12], [13], semantic maps [14], [15], text description [16], [17] and scene graphs [18], [19]. Conditional image synthesis, especially with coarse yet complicated and reconfigurable conditions, remains a long-standing problem. As illustrated in Fig. 1, we shall only focus on conditional image synthesis from spatial layout and style latent codes, so-called layout-to-image [20]. Powerful systems, once developed, can pave a way for computers to truly understand visual patterns and their compositions via a comprehensive and systematic “analysis-by-synthesis” scheme. Those systems will also enable a wide range of practical applications, e.g., generating high-fidelity data for long-tail scenarios in different vision tasks such as autonomous driving.

In layout-to-image, the layout that a synthesized image needs to satisfy consists of labeled bounding boxes configured in an image lattice (e.g., 256 x 256 pixels). The style of a synthesized image refers to structural and appearance variations at both image and object levels, which is often encoded by some latent codes. Generating images from a spatial layout represents a sweet spot in conditional image synthesis. Spatial layouts are usually used as intermediate representations for other conditional image synthesis tasks such as text-to-image [17], [21] and scene-graph-to-image [18], [19]. And, layouts are more flexible, less constrained and easier to collect than other conditions such as semantic segmentation maps [11], [14]. Existing object detection benchmarks can be exploited in training.

The generative learning task of layout-to-image was recently proposed and only a few work have been proposed in the very recent literature [18], [19], [20], [23]. Although relatively new, it has been well recognized in the computer vision community. For example, the work (Grid2Im) by Ashua and Wolf [19] won the best paper honorable mentions at ICCV 2019. The layout-to-image task was emerged under the context of remarkable progress made in conditional image synthesis with relatively less complicated conditions such as the class-conditional image synthesis in ImageNet by the BigGAN [8], and the amazing style control for specific
objects (e.g., faces and cars) by the StyleGAN [8] 1. Despite the big successes achieved by BigGANs and StyleGANs, learning generative models for layout-to-image entails more research. In addition to realness, generative models for layout-to-image need to tackle many spatial and semantic relationships among multiple objects (combinatorial in general). Specifically, learning layout-to-image requires addressing the problems of learning one-to-many mapping (i.e., one layout covers many plausible realizations in image synthesis to preserve the intrinsic uncertainty), and of handling consistent multi-object generation (e.g., occlusion handling for overlapped bounding boxes and uneven, especially long-tail distributions of objects). Because of those, it is difficult to capture underlying probability distributions defined in the solution space of layout-to-image.

In this paper, we further focus on controllable image synthesis from reconfigurable spatial layouts and style codes. As illustrated in Fig. 2, by controllable and reconfigurable, it means a generative model is capable of (i) Style Control – the model can preserve the intrinsic one-to-many mapping from a given layout to multiple plausible images with sufficiently different structural and appearance styles (i.e., diversity), at both image and object levels, and (ii) Layout Control – the model is also adaptive with respect to changes of layouts (e.g., adding new objects), or perturbations of bounding boxes in a given layout, as well as the styles associated with the changes of spatial layouts. Prior arts on layout-to-image mainly focus on low resolution (64 × 64) [18], [20], except for the very recent Grid2Im method [19] which can synthesize images at a resolution of 256 × 256. Main drawbacks of existing methods are in two-fold: the diversity of styles in generated images is not sufficiently high to preserve the intrinsic one-to-many mapping for a given layout, and the controllability of styles along consecutive layout changes is often not sufficiently strong, as illustrated in Fig. 2. We aim to address these issues in this paper. Specifically, as we shall elaborate, the proposed method takes a step forward to address the problem of layout-to-image by learning layout-to-mask-to-image, while leveraging some of the best practice developed in state-of-the-art cGANs [6], BigGANs [5], StyleGANs [8] and Mask R-CNNs [25].

1. We view the StyleGAN as an implicitly conditional image synthesis framework since only one category is usually handled in training.
termed the proposed method LostGAN in our previous conference paper presented at ICCV 2019, entitled Image Synthesis from Reconfigurable Layout and Style [26]. We shall call the conference version LostGAN-V1 and the updated model LostGAN-V2 in this paper. We first give an overview of our LostGAN and then summarize the changes of LostGAN-V2.

The proposed LostGAN addresses the layout-to-image problem by learning layout-to-mask-to-image. To account for the gap between bounding boxes in a layout and underlying object shapes, learning layout-to-mask is an intuitive and straightforward intermediate step to induce fine-grained style control of objects in a synthesized image, which also helps decouple learning of object geometry and learning of object appearance. Layout-to-mask itself is a relatively easier task than direct layout-to-image since object appearance is ignored. In the meanwhile, motivated by the impressive recent progress on conditional image synthesis from semantic label maps [11], [14], [15], it also makes sense to integrate layout-to-mask. If a reasonably good mask can be inferred for an input layout, learning mask-to-image can then leverage the best practice in conditional image synthesis from semantic label maps. A naive approach is to develop two-stage generators, which may provide less effective solutions. Instead, we present a joint learning paradigm (i.e., using a single generator). Fig. 3 illustrates the overall workflow of the proposed LostGAN. Fig. 4 illustrates the joint learning of layout-to-mask-to-image.

The generator has three inputs as commonly used in generative learning of layout-to-image: (i) a spatial layout, \( L \) consisting of a number of object bounding boxes in an image lattice, (ii) a latent vector, \( z_{img} \) for style control at the image level, and (iii) a bucket of latent vectors, \( z_{obj}\)'s, each of which is used for style control of an object instance. The latent vectors are randomly sampled from the standard multivariate Gaussian distribution. The generator takes the image latent vector as its direct input for overall style control, while utilizing a novel feature normalization scheme for object-level style control, which takes as input the concatenation of the bucket of object latent vectors and the label embedding of objects in the layout. The object latent codes are involved in each stage of the generator for better style control, similar in spirit to the StyleGAN [8]. The objective of the generator is to capture the underlying probability distribution, \( p(I_{reg} | L, z_{img}, z_{obj1}, \ldots, z_{objm}) \). While straightforward for synthesizing images (using a single phase of forward computation), the generator involves a challenging inference step entailed in estimating the model parameters, that is to compute the latent codes for a real image \( I^{real} \) by sampling the posterior distribution, \( p(z_{img}, z_{obj1}, \ldots, z_{objm} | I^{real}, L) \).

To get around the difficulty of the posterior inference, GANs utilize an adversarial training paradigm by introducing an extra discriminator. Detailed specifications of the generator are shown in Fig. 4.

The discriminator has two inputs: a given image, either synthesized or real, and the corresponding spatial layout. It consists of three components: (i) a feature backbone extracting features from the input image, (ii) an image head classifier computing the image realness score based on the extracted features (the higher the score is, the more real an image is), and (iii) an object head classifier computing the realness score for each object instance. The features for an object instance is computed by the RoIAlign operation [25] using the given layout. Motivated by the projection-based cGANs [6] and the practice in the BigGAN [5], a label projection-based score is added to the realness score of each object instance. Detailed specifications of the generator are shown in Fig. 5.

The loss function consists of both image and object adversarial hinge loss terms [4], [6], [27], [28] (balanced by a trade-off parameter, \( \lambda \)). The hinge loss aims to push the realness score of a synthesized image sufficiently away from that of a real image by a predefined margin. Under the two-player minmax game setting of GANs, the hinge loss works better to enforce both the generator and the discriminator more aggressive, leading to synthesized images of higher fidelity.
The Instance-Sensitive and Layout-Aware Feature Normalization (ISLA-Norm) scheme is presented to realize the proposed layout-to-mask-to-image pipeline in our LostGAN. Fig. 4 illustrates the proposed ISLA-Norm. As a feature normalization scheme, it consists of two components: feature standarization, and feature recalibration by learning an affine transformation. The former is done as the BatchNorm [29] in which channel-wise mean and standard deviation are computed in a mini-batch. The latter is different from BatchNorm. Unlike BatchNorm in which channel-wise affine transformation parameters, $\beta$ (for re-shifting) and $\gamma$ (for re-scaling) are learned as model parameters and shared across spatial dimensions by all instances, in our ISLA-Norm, we first learn object instance-sensitive channel-wise affine transformations from the concatenation of object label embedding and object style latent vectors, as shown by the arrows in blue in Fig. 4, similar in spirit to the Adaptive Instance Normalization (AdaIN) used in StyleGANs [8] and the projection-based conditional BatchNorm used in cGANs [5]. We also learn the mask for an input layout in two pathways: one pathway learns the mask from the concatenation of object label embedding and object style latent vectors, and the other learns the mask from feature maps at different stage in the generator. A learnable weighted sum of the two masks are used as the inferred mask at a stage in the generator. Then, to obtain fine-grained spatially-distributed multi-object style control for an input layout, we place the object instance-sensitive channel-wise affine transformations in the learned mask, leading to the instance-sensitive and layout-aware affine transformations for feature recalibration in the generator, as illustrated by the light-grey cube in Fig. 4.

Summary of Changes. Compared to LostGAN-V1, the main changes of LostGAN-V2 are as follows.
- The ISLA-Norm is extended by integrating masks learned from feature maps at different stages in the generator.
- The experiments are significantly extended by training models at higher resolutions and by comparing with the prior arts including the Grid2Im [19] and the GauGAN [15].
- The paper is thoroughly rewritten with much more details on different aspects of the LostGAN and on the experimental settings, together with new figures of the model.
- Several ablation studies are added to analyze the proposed LostGAN and ISLA-Norm.

Our source code and pretrained models have been made publicly available at https://github.com/iVMCL/LostGANs.

1.3 Related Work

Generative models have been studied widely in recent years such as Autoregressive models, Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). For image generation, Autoregressive models such as Pixel-RNN [30] and PixelCNN [31] synthesize images pixel by pixel based on conditional distribution over pixels. VAEs [32], [33] jointly train an encoder and decoder where the former maps images into latent distribution and the latter generates images based on the latent distribution. GANs [1] are able to synthesize realistic and high resolution images under various settings, including both unconditional [2], [7], [8], [34] and conditional tasks [5], [6], [9]. Typically, a GAN consists of a Generator that produces realistic fake images from input (e.g., random noise) and a Discriminator that distinguishes generated images from real ones. More recently, a unified divergence triangle framework is proposed for joint training of generator model, energy-based model, and inference model for generative tasks [35].
**Conditional Image Synthesis.** Conditional Image synthesis takes additional information (*i.e.*, class information [3], [5], [6], [9], [36], source image [10], [12], [13], [37], text description [16], [17], [38], [39], scene graph [18], [40], etc) as input. How to feed conditional information to a GAN model has been studied in various ways. In [9], [16], the conditional information are encoded to a vector, concatenated with noise, and then passed to the generator. In [6], projection-based methods, which incorporate conditional information to the discriminator by inner product between feature and learned class embedding, effectively improve the quality of class conditional image generation. In [16], [17], [41], conditional information is utilized by the discriminator using simple concatenation with the input or intermediate feature maps. In [5], [15], [42], [43], conditional information is provided to the generator by conditional gains and bias in the Batch Normalization [29] layers. The concurrent work, GauGAN [15] learns spatially adaptive normalization parameters from annotated semantic masks, while our proposed ISLA-Norm learns from coarse layout information.

**Image Synthesis from Layout.** Image synthesis from coarse layout, which usually used as intermediate representation for other conditional image synthesis tasks, has been studied in the very recent literature and proven a difficult task. In [18], [23], [39], [44], layout and object information are utilized in text to image or scene graph to image generation. In [44], [45], locations of multiple objects are controlled in text-to-image generation by adding an extra object pathway in both the generator and discriminator. In [18], [23], [39], a two-step approach is used in image synthesis: generating the semantic layout (class label, bounding boxes and semantic mask) from a text description or a scene graph, and synthesizing images conditioned on the predicted semantic layout and text description (if present). However, in [19], [23], [39], pixel-level instance segmentation annotations are needed in training, while our proposed method does not require pixel-level annotations and can learn semantic masks in a weakly-supervised manner. The layout-to-image task was first studied in [20] at the resolution of 64×64, which uses a variational autoencoders based network, together with long-short term memory (LSTM), for object feature fusion. In [40], an external memory bank is introduced, consisting of objects cropped from real images in training, which are retrieved and pasted in generating images from layouts at the resolution of 64×64.

### 1.4 Our Contributions

This paper makes the following main contributions to the field of conditional image synthesis.

- **It presents a layout- and style-based architecture for GANs (termed LostGANs) which address the problem of layout-to-image by learning layout-to-mask-to-image and realize controllable image synthesis from reconfigurable layouts and styles.** The outputs of our LostGANs cover both image and semantic mask synthesis.

- **It presents an object instance-sensitive and layout-aware feature normalization scheme (termed ISLA-Norm) which explicitly and jointly accounts for the learning of layout-to-mask and the learning of spatially-distributed affine transformations for feature recalibration at an object mask level.**

- **It can synthesize images at a resolution of up to 512×512 and shows state-of-the-art performance in terms of the Inception Score [46], the Fréchet Inception Distance [47], the Diversity Score base on the LPIPS metric [48] and the classification accuracy [49] on two widely used datasets, the COCO-Stuff [22] and the Visual Genome [50].**

### 1.5 Paper Organization

In the remainder of this paper, Section 2 presents the problem formulation of layout-to-image and technical details of our proposed LostGAN and ISLA-Norm. Section 3 shows the experimental settings, quantitative and qualitative results, together with an ablation study. Section 4 concludes this paper.

### 2 The Proposed Method

In this section, we first define the problem and then present details of our LostGAN and ISLA-Norm.

#### 2.1 Problem Formulation

Denote by Λ an image lattice (e.g., 256 × 256) and by I an image defined on the lattice. Let $L = \{ (ℓ_i, bbox_i) \}_{i=1}^m$ be a layout consisting of n labeled bounding boxes, where label $ℓ_i \in C$ (e.g., $|C| = 171$ in the COCO-Stuff dataset [22]), and a bounding box $bbox_i \subseteq Λ$. Different bounding boxes may overlap and thus have undetermined partial-order of occlusions. Let $z_{img}$ be the latent code controlling image style and $z_{obj}$, the latent code controlling object instance style for $(ℓ_i, bbox_i)$ (e.g., the latent codes are randomly sampled from the standard Gaussian distribution, $N(0, 1)$ under the i.i.d. setting). Denote by $Z_{obj} = \{ z_{obj} \}_{i=1}^m$ the set of object instance style latent codes.

Image synthesis from layout and style is to learn a generation function capable of synthesizing an image for a given input $(L, z_{img}, Z_{obj})$,

$$I^{syn} = G(L, z_{img}, Z_{obj}; Θ_G)$$  \hspace{1cm} (1)$$

where $Θ_G$ represents the parameters of the generation function. In general, a generator model $G(·)$ is expected to capture the underlying conditional data distribution $p(I|L, z_{img}, Z_{obj}; Θ_G)$ in the high-dimensional space.

**Reconfigurability of a generator model $G(·)$**. We are interested in three aspects in this paper:

- **Image style reconfiguration:** If we fix the layout $L$, is the generator $G(·)$ capable of synthesizing images with different styles for different $(z_{img}, Z_{obj})$, while retaining the object configuration conditioned on $L$?

- **Object style reconfiguration:** If we fix the layout $L$, is the generator $G(·)$ capable of generating consistent images with different styles for the object $(ℓ_i, bbox_i)$ using different $z_{obj}$, while retaining the object configuration conditioned on $L$ and the styles of the remaining objects?

- **Layout reconfiguration:** Given an input tuple $(L, z_{img}, Z_{obj})$, is the generator $G(·)$ capable of generating consistent images for different $(L', z_{img}, Z_{obj})$'s, where we can add a new object to $L'$ or just change the location...
and/or label of an existing bounding box? When a new object is added, we also sample a new \( z_{obj} \) to add in \( Z_{obj}^+ \). When only the bounding box location changes, we keep all latent codes unchanged (i.e., \( Z_{obj}^+ = Z_{obj} \)).

It is a challenging problem to address the three aspects by learning a single generator model \( \mathcal{G}() \). Intuitively, it might be even difficult for well-trained artistic people to do so at scale (e.g., handling the 171 categories in the COCO-Stuff dataset). Due to the complexity that the generator model (Eqn. 1) needs to handle, it is often implemented using expressive and over-parameterized deep neural networks (DNNs). It is also well-known that training a DNN-based generator model individually is a extremely difficult task (due to the difficulty of sampling the posterior). Adversarial training is a popular workaround and GANs [1] are widely used in practice, which are formulated under a two-player minmax game setting.

### 2.2 The LostGAN

In this section, we present the technical details of our LostGAN.

#### 2.2.1 The Generator

As illustrated in Fig. 4, the generator \( \mathcal{G}() \) consists of a linear full-connected (FC) layer, followed by a number of residual building blocks (ResBlocks) [24] depending on the target resolution of image synthesis, and a “ToRGB” module outputting a synthesized image.

- The image style latent code \( z_{img} \in \mathbb{R}^{d_{img}} \) is a \( d_{img} \)-dim vector (e.g., \( d_{img} = 128 \) in our experiments). The linear FC layer projects \( z_{img} \) to a \( 4 \times 4 \times (16 \times ch) \) dimensional vector. The projected vector is then reshaped as a tensor of dimensions \( (4, 4, 16 \times ch) \) (representing height, width and channels), where \( ch \) is a hyperparameter to control the model complexity (e.g., \( ch = 64 \) used in our experiments).
- The right-top of Fig. 4 shows the detail of a ResBlock. The ResBlock uses the basic block design in ResNets [24], as adopted in the projection-based cGAN [6] and the BigGAN [5]. Each ResBlock upsamples its input once by a factor of 2. So, we will need \( B \) ResBlocks to generate images at a resolution of \( 4^B \times 4^B \) (e.g., \( B = 4 \) for the resolution of \( 64 \times 64 \)). Each ResBlock also reduces the number of feature channels of its input by a factor of 2.
- The final “ToRGB” module consists of a BatchNorm [29].

#### 2.2.2 The ISLA-Norm

The are two ISLA-Norm modules in a ResBlock. Denote by \( x \) an input 4D feature map of ISLA-Norm, and \( x_{n,c,h,w} \) the feature response at a position \( (n, c, h, w) \) (using the convention order of axes for batch, channel, and spatial height and width). We have \( n \in [0, N - 1], c \in [0, C - 1], h \in [0, H - 1], w \in [0, W - 1] \), where \( N \) is the mini-batch size or the accumulated size of synchronized mini-batches, and \( C, H, W \) depend on the stage of a ResBlock.

**Feature Standardization.** Our ISLA-Norm first computes the channel-wise mean and standard deviation as done in the BatchNorm [29]. In training, ISLA-Norm first normalizes \( x_{n,c,h,w} \) by

\[
    \hat{x}_{n,c,h,w} = \frac{x_{n,c,h,w} - \mu_c}{\sigma_c},
\]

where the channel-wise batch mean \( \mu_c = \frac{1}{NH} \sum_{n,h,w} x_{n,c,h,w} \) and standard deviation \( \sigma_c = \sqrt{\frac{1}{NH} \sum_{n,h,w} (x_{n,c,h,w} - \mu_c)^2 + \epsilon} \) (\( \epsilon \) is a small positive constant for numeric stability).

**Feature Recalibration.** In the vanilla BatchNorm [29], the recalibration is done by learning channel-wise affine transformations, consisting of the re-scaling parameter, \( \gamma_c \)'s and the re-shifting parameters, \( \beta_c \)'s. We have

\[
    \hat{x}_{n,c,h,w}^{BN} = \gamma_c \cdot \hat{x}_{n,c,h,w} + \beta_c.
\]

In our ISLA-Norm, we will learn instance-sensitive and layout-aware affine transformation parameters, \( \gamma_{n,c,h,w} \)'s and \( \beta_{n,c,h,w} \)'s, and we have

\[
    \hat{x}_{n,c,h,w} = \gamma_{n,c,h,w} \cdot \hat{x}_{n,c,h,w} + \beta_{n,c,h,w}.
\]

#### 2.2.2.1 Computing \( \gamma_{n,c,h,w} \) and \( \beta_{n,c,h,w} \)

Without loss of generality, we show how to compute the gamma and beta parameters for one sample, i.e., \( \gamma_{c,h,w} \) and \( \beta_{c,h,w} \). As shown in the left of Fig. 4, we have the following six components.

1. **Label Embedding.** We use one-hot label vector for the \( m \) object instances in a layout \( L \), which results in a one-hot label matrix, denoted by \( Y \), of the size \( m \times d_e \), where \( d_e \) is the number of object categories (e.g., \( d_e = 171 \) in the COCO-Stuff dataset). Label embedding is to learn a \( d_e \times d_e \) embedding matrix, denoted by \( W \), to compute the vectorized representation for labels,

\[
    \Upsilon = Y \cdot W,
\]

where \( \Upsilon \) is a \( m \times d_e \) matrix and \( d_e \) represents the embedding dimension (e.g., \( d_e = 128 \) in our experiments).

2. **Joint Label and Style Encoding.** We sample from the standard Gaussian distribution the object style latent codes \( Z_{obj} \) which is a \( m \times d_{obj} \) noise matrix (e.g., \( d_{obj} = 128 \) the same as \( z_{img} \) in our experiments). We concatenate the label-to-vector matrix \( \Upsilon \) and the object latent style matrix as the joint label and style encoding,

\[
    \mathbb{S} = (\Upsilon, Z_{obj}),
\]

which is a \( m \times (d_e + d_{obj}) \) matrix. So, the object instance style depends on both the label embedding (semantics) and i.i.d. latent codes (accounting for style variations).

3. **Mask Generation from the joint label and style encoding \( \mathbb{S} \).** We first generate an mask for each object instance in a layout \( L \) at some predefined size, \( s \times s \) (e.g., \( s = 32 \)) individually. Then, we resize the generated masks to the sizes of corresponding bounding boxes at a ResBlock stage in the generator.

- The mask generation process consists of two components: one is a simplified generator model (the small trapezoid in purple in Fig. 4), and the other is a simple “ToMask” operation. The former composes a linear FC layer projecting \( \mathbb{S} \) to a tensor with a resolution of \( 4 \times 4 \) (after reshaping), and a few stages of Conv3x3+BatchNorm+ReLU (where Conv3x3 refers to a convolution operation with a kernel of spatial dimensions, \( 3 \times 3 \)). The “ToMask” operation is
then implemented by Conv3x3+Sigmod, whose output is \( m \times s \times s \), representing a \( s \times s \) mask for each of the \( m \) object instances. Based on the mask size \( s \), we also upsample the feature map after a Conv3x3+BatchNorm+ReLU by a factor of 2 for a number of stages as needed.

- After resizing and reassembling the generated mask for a ResBlock stage, we obtain a mask tensor of dimensions \( (m, H, W) \), denoted by \( M_{\beta} \), each slice of which has zeros outside the corresponding bounding box, \( \text{bbox} \). For the visualization purpose (e.g., those shown in Fig. 4), when reassembling the resized object masks, we use arg max across the \( m \) channels of \( M_{\beta} \) to assign the label index for a pixel occupied by more than one objects due to occlusions.

iv) Mask Generation from the feature maps in a generator. For a ResBlock stage, we learn a mask from its input feature map using a simple “ToMask” operation implemented by Conv3x3+Sigmod, where the out channel of the Conv3x3 kernel is \( d_\ell \) (i.e., the number of categories in a dataset). The mask is represented by a tensor of sizes \( (d_\ell, H, W) \). We clip the mask based on the layout by only keeping values unchanged within the bounding boxes of the object instances in a layout (and zeroing out the remainder). Denote by \( M_{\beta}(L) \) the mask tensor of sizes \( (m, H, W) \) after the clipping (omitting the index for a ResBlock in the generator without loss of generality). For the second ISLA-Norm in a ResBlock (the right-top of Fig. 4), we just upsample \( M_{\beta}(L) \) by a factor of 2 for simplicity.

v) Object instance-sensitive channel-wise affine transformations learned from the joint label and style encoding \( S \). We adopt a linear projection with a learnable \( (d_\ell + d_{\text{obj}}) \times 2C \) projection matrix \( A \), where \( C \) is the number of channels, and we have,

\[
T = S \cdot A. \tag{7}
\]

which is a matrix of sizes \( (m, 2C) \). Let \( T_\beta \) and \( T_\gamma \) be the column-wise first and second half of \( T \). We unsqueeze both \( T_\beta \) and \( T_\gamma \) to the size of \( (m, C, H, W) \) by replicating values across the spatial dimensions. Learning the affine transformations in this way leads to stronger style control of our LostGAN than other layout-to-image methods, since the style latent codes get involved in every stage of the generator, rather than being used as input only to the first stage of the generator in other layout-to-image methods.

vi) Computing ISLA \( \gamma_{c,h,w} \) and \( \beta_{c,h,w} \). We first unsqueeze the two masks, \( M_{\delta} \) and \( M_{\beta}(L) \), to the sizes \( (m, C, H, W) \) by replicating \( C \) channels. Then, we have,

\[
\gamma_{c,h,w} = \frac{1}{M_{h,w}} \sum_{i=1}^{m} \text{Min}(i, c, h, w) \times T_\gamma(i, c, h, w), \tag{8}
\]

\[
\beta_{c,h,w} = \frac{1}{M_{h,w}} \sum_{i=1}^{m} \text{Min}(i, c, h, w) \times T_\beta(i, c, h, w), \tag{9}
\]

where \( \text{Min}(\cdot) = [(1 - \alpha) \cdot M_{\delta} + \alpha \cdot M_{\beta}(L)](\cdot) \) with \( \alpha \) being a learnable weight to balance the two masks, and \( M_{h,w} = \sum_{i=1}^{m} \text{Min}(i, 0, h, w) \) if the pixel \( (h, w) \) is occupied by multiple object bounding boxes \( (M_{h,w} \text{ is then to normalize the sum of mask values}), otherwise \( M_{h,w} = 1 \). We note that unlike the arg max operation that we used for the overlapping regions in visualizing the learned masks, the average is used in computing the spatially-distributed affine transformations.

Handling Background. To account for the situation in which all object instances do not occupy the entire image lattice (e.g., in the VG dataset [50]), we introduce a background class \( \ell_0 \) with \( \text{bbox}_0 = \Lambda \).

2.2.3 The Discriminator

As shown in Fig. 5, our discriminator consists of three components: a shared ResNet-based feature backbone, an image head classifier and an object head classifier.

- The shared ResNet-based feature backbone consists of a number of basic ResBlocks [24] with the BatchNorm modules removed (as commonly adopted in prior arts such as the SNGAN [4] and the BigGAN [5]). The number of ResBlocks is subject to the target resolution of layout-to-image.

- The image head classifier consists of an image-level feature backbone, a global average pooling layer, and an one-output FC layer. The image-level feature backbone includes a number of ResBlocks. The output of the FC layer is a scalar realness score, denoted by \( p_{\text{img}} \).

- The object head classifier is a simplified version of R-CNN based object detector [25], [52] with bounding boxes being

Fig. 5. Illustration of the discriminator (left). The shared feature backbone, the image-level feature backbone and the object-level feature pyramid use ResBlocks (right), and each consists of a number of ResBlocks depending on the target resolution of layout-to-image (e.g., \( 128 \times 128 \) or \( 256 \times 256 \)). In the ResBlock, “[op]” means an operation is optional subject to the settings. The object-level feature pyramid is used for placing object instances of different sizes at different feature layers (e.g., smaller bounding boxes placed at lower feature layers as done in the FPN [51]), such that the RoIAlign operation is meaningful. “FC” represents a fully-connected layer (with either a scalar output shown by a grey triangle or a vector output shown by a grey trapezoid). “AvgPool” represents a global channel-wise average pooling over the spatial dimensions. See text for details.
given in the input layout. It consists of an object-level feature pyramid, a RoIAlign layer [25], and two FC layers. The one-output FC layer computes a scalar realness score for each object instance, denoted by $p_{\text{obj}}^r (i \in [1, m])$. To encode object label semantic information, we also compute a label projection score [5], [6] for each object instance, denoted by $p_{\text{obj}}^\ell$, which is the inner product between the label embedding (calculated in the same as Eqn. 5) and the linear projection (using a FC layer) of the RoIAlign feature vector. The overall score of an object instance is the sum: $p_{\text{obj}} = p_{\text{obj}}^r + p_{\text{obj}}^\ell$.

In sum, denote by $\mathcal{D}(\cdot; \Theta_D)$ the discriminator with parameters $\Theta_D$. Given an image $I$ (real or synthesized) and a layout $L$, the discriminator computes a list of scores,

$$\left(p_{\text{img}}, p_{\text{obj}}^1, \ldots, p_{\text{obj}}^m\right) = \mathcal{D}(I, L; \Theta_D)$$ (10)
2.2.4 The Loss Functions

Under the mini-batch based SGD framework, for the generator, the loss function of $\Theta_G$ is defined by,

$$L(\Theta_G|\Theta_D) = -\sum_{(L,I^{syn},I^gt)\in\mathbb{B}} [P_D(I^{syn},L;\Theta_D) - ||I^{syn} - I^gt||_1 - ||F(I^{syn}) - F(I^gt)||_1],$$

where $\mathbb{B}$ represents a mini-batch, $I^{syn}$ and $I^gt$ represent a synthesized image (Eqn. 1) and the ground-truth image for the spatial layout $L$, $P_D(I^{syn},L;\Theta_D) = \lambda \cdot p_{img} + \frac{1}{m} \sum_{i=1}^{m} p_{obj}$ with a trade-off parameter $\lambda$ (0.1 used in our experiments), the second term in the right-hand side is the reconstruction loss, and the last term is the perceptual loss [53] which measure L1 difference between features, $F(\cdot)$ of generated image and ground truth images by an ImageNet pretrained network such as the VGG network [54]. Minimizing $L(\Theta_G|\Theta_D)$ is trying to fool the discriminator by generating high fidelity images.

For the discriminator, we utilize the hinge version [27], [28] of the standard adversarial loss [1],

$$l_t(I,L) = \begin{cases} \max(0,1-p_t) & \text{if } I \text{ is a real image} \\ \max(0,1+p_t) & \text{if } I \text{ is a fake image} \end{cases}$$

where $t \in \{img, obj_1, \cdots, obj_m\}$. In the hinge loss, no penalty will occur if the score of a real image (or a real object instance) is greater than or equal to 1, and the score of a fake image (or a fake object instance) is less than or equal to -1. The hinge loss is more aggressive than the real vs fake binary classification in the vanilla GAN. The overall loss is,

$$l(I,L) = \lambda \cdot l_{img}(I,L) + \frac{1}{m} \sum_{i=1}^{m} l_{obj_i}(I,L).$$

The loss function of $\Theta_D$ is defined by,

$$L(\Theta_D|\Theta_G) = \sum_{(L,I^{syn},I^gt)\in\mathbb{B}} [l(I^gt,L) + l(I^{syn},L)],$$

where $p(I, L)$ represents both the real and fake (synthesized by the generator) data. Minimizing $L(\Theta_D|\Theta_G)$ is trying to tell apart the real and fake images.

2.2.5 Implementation Details

In implementation and training, we follow the practice used in [3], [5], [6]. Synchronized Batch Normalization [29], where batch statistics for feature standarization are computed over all devices, is adopted in our ISLA-Norm. The Spectral Normalization [4] of model parameters is also applied in both the Generator and the Discriminator to stabilize training. Parameters of the Generator and the Discriminator are initialized using the Orthogonal Initialization method [55]. The Adam optimizer [56] is used with $\beta_1 = 0$ and $\beta_2 = 0.999$. 

Fig. 7. Generated samples from given layouts on VG by different models. From top to bottom shows Input Layout, and Images generated by Layout2Im [20] $64 \times 64$, our LostGAN-V1 $128 \times 128$, our LostGAN-V2 $256 \times 256$, and Ground Truth.
The learning rate is set constant $10^{-4}$ for both the Generator and the Discriminator. We use batch size of 128 based on our computing hardware resource. Training our LostGAN takes about 2-3 days, e.g., either for the models generating $128 \times 128$ images on 2 NVIDIA V100 GPUs, or for the models synthesizing $256 \times 256$ images on 4 NVIDIA V100 GPUs.

## 3 Experiments

We test our LostGAN in the COCO-Stuff dataset [22] and the Visual Genome (VG) dataset [50]. We evaluate LostGAN-V1 at two resolutions ($64 \times 64$ and $128 \times 128$) and LostGAN-V2 at three resolutions ($128 \times 128$, $256 \times 256$ and $512 \times 512$). Our LostGAN-V2 obtains state-of-the-art performance.

### 3.1 Datasets

The COCO-Stuff 2017 [22] augments the COCO dataset with pixel-level stuff annotations. The annotation contains 80 thing classes (person, car, etc.) and 91 stuff classes (sky, road, etc.) Following settings of [18], objects covering less than 2% of the image area are ignored, and we use images with 3 to 8 objects. For the Visual Genome (VG) dataset [50], we follow the settings of [18] to remove small and infrequent objects, which results in 62,565 images for training, 5,056 images for validation and 5,088 images for testing, with 3 to 30 objects from 178 categories in each image.

### 3.2 Methods in Comparison

We compare with four prior arts: i) The pix2pix method [11] learns to map images between two domains. We reuse the pix2pix results reported in the Layout2Im [20] in our comparisons, where a pix2pix model is trained to synthesize images from a feature map learned to encode the layout. The number of channels of the feature map is the number of categories (e.g., 171 in COCO-Stuff). ii) The scene graph to image (sg2im) method [18] synthesizes images from input scene graphs with an intermediate scene-graph-to-layout module. We compare with sg2im using the ground-truth (GT) layouts. iii) The Layout2Img method [20] is the first to synthesize images directly from input layouts. These three methods have only been evaluated at the resolution of $64 \times 64$. iv) The Grid2Im [19] method [19] extends the sg2im method, which has been tested at two resolutions, $128 \times 128$ and $256 \times 256$, in the COCO-Stuff dataset only since ground-truth masks are needed in training. We also compare with Grid2Im using the GT layouts. Fig. 6 and 7 show examples of synthesized images by different methods.

### 3.3 Evaluation Metrics

It remains a challenging problem to automatically evaluated DNN-based generator models in general. For layout-to-image, we adopt four state-of-the-art metrics as follows.

The Inception Score (IS) [46] uses an Inception V3 network pre-trained on the ImageNet-1000 classification benchmark and computes a score (statistics) of the network’s outputs with $N$ synthesized images $I_i$’s of a generator model $G_i$,

$$ IS(G) = \exp\left\{ \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{1000} p(y = j | I_i) \log \frac{p(y = j | I_i)}{\hat{p}(y = j)} \right\}, \quad (15) $$

where $\hat{p}(y = j) = \frac{1}{N} \sum_{i=1}^{N} p(y = j | I_i)$. The IS aims to capture two desirable qualities of image synthesis: Synthesized image should contain clear and meaningful objects (subject to the ImageNet-1000 training datasets), and diverse images from all the different categories in ImageNet should be observed in synthesized images. So, the larger the IS is, the better a generator model is. Multiple runs are usually used to calculate the mean±std evaluation (e.g., 5 runs are typically used). The IS does not leverage the statistics of real images.

The Fréchet Inception Distance (FID) [47] has been proposed to improve IS by incorporating statistics from real images. It also uses an ImageNet-pretrained Inception V3

| Methods       | IS↑ COCO | IS↑ VG | FID↓ COCO | FID↓ VG | DS↑ COCO | DS↑ VG |
|---------------|---------|-------|-----------|--------|----------|--------|
| Real Images   | $16.30 \pm 0.40$ | $13.90 \pm 0.50$ | -          | -      | -        | -      |
| Real Images   | $22.30 \pm 0.50$ | $20.50 \pm 1.50$ | -          | -      | -        | -      |
| Real Images   | $28.10 \pm 1.60$ | $28.60 \pm 1.20$ | -          | -      | -        | -      |
| (pixel-to-pixel [11]) $64 \times 64$ | $34.50 \pm 1.70$ | $34.20 \pm 1.10$ | -          | -      | -        | -      |
| sg2im (GT Layout) [18] $64 \times 64$ | $7.30 \pm 0.10$ | $6.30 \pm 0.20$ | $67.96 \pm 74.61$ | $0.02 \pm 0.01$ | $0.15 \pm 0.12$ | -      |
| Layout2Im [20] $64 \times 64$ | $9.10 \pm 0.10$ | $8.10 \pm 0.10$ | $44.19 \pm 39.68$ | $0.15 \pm 0.06$ | $0.17 \pm 0.09$ | -      |
| Layout2Im + OWA [20] $64 \times 64$ | $9.70 \pm 0.10$ | $8.00 \pm 0.20$ | $40.19 \pm 33.54$ | $0.09 \pm 0.05$ | $0.09 \pm 0.11$ | -      |
| LostGAN-V1 [26] $64 \times 64$ | $9.80 \pm 0.20$ | $9.20 \pm 0.40$ | $34.31 \pm 34.75$ | $0.25 \pm 0.09$ | $0.34 \pm 0.10$ | -      |
| Grid2Im [20] (GT Layout) $128 \times 128$ | $12.60 \pm 1.15$ | $12.50 \pm 1.25$ | -         | -      | $0.26 \pm 0.09$ | -      |
| LostGAN-V1 [26] $128 \times 128$ | $13.80 \pm 0.40$ | $11.10 \pm 0.60$ | $29.65 \pm 29.36$ | $0.40 \pm 0.09$ | $0.43 \pm 0.09$ | -      |
| LostGAN-V2 [128 x 128] $128 \times 128$ | $14.21 \pm 0.40$ | $10.71 \pm 0.26$ | $24.76 \pm 29.09$ | $0.45 \pm 0.09$ | $0.42 \pm 0.09$ | -      |
| Grid2Im [19] (GT Layout) $256 \times 256$ | $15.23 \pm 0.11$ | - | $65.95$ | $0.34 \pm 0.13$ | - | - |
| LostGAN-V1 [26] $256 \times 256$ | $18.01 \pm 0.50$ | $14.42 \pm 0.38$ | $42.55 \pm 47.62$ | $0.55 \pm 0.09$ | $0.53 \pm 0.09$ | -      |
| LostGAN-V2 [256 x 256] $256 \times 256$ | $19.50 \pm 0.40$ | $14.40 \pm 0.46$ | $51.99 \pm 52.73$ | $0.66 \pm 0.11$ | $0.61 \pm 0.09$ | -      |
| LostGAN-V2 $512 \times 512$ | $17.55 \pm 0.23$ | $14.40 \pm 0.46$ | $51.99 \pm 52.73$ | $0.66 \pm 0.11$ | $0.61 \pm 0.09$ | -      |

### TABLE 1

Quantitative comparisons using the Inception Score (IS, higher is better), FID (lower is better) and Diversity Score (DS, higher is better) evaluation metrics in the COCO-Stuff [22] and VG [50] datasets. See text for details.

### TABLE 2

Comparisons of the CAS. See text for details.
network and computes the Fréchet distance [57] between two Gaussian distributions fitted to synthesized images and real images respectively. Denote by \((\mu^0, \Sigma^0)\) and \((\mu^1, \Sigma^1)\) the mean vector and the covariance matrix of the Gaussian distribution fitted on synthesized images and real images respectively. The Fréchet distance is defined by,

\[
FID^2((\mu^0, \Sigma^0), (\mu^1, \Sigma^1)) = ||\mu^0 - \mu^1||_2^2 + Tr(\Sigma^0 + \Sigma^1 - 2(\Sigma^0\Sigma^1)^{1/2}). \tag{16}
\]

So, the lower the FID is, the better a generator model is. Both the IS and FID do not explicitly measure the quality of one-to-many mapping in layout-to-image.

The Diversity Score (DS) aims to compare the perceptual similarity in a DNN feature space between two images, \(I_1\) and \(I_2\), generated from the same layout. We adopt the LPIPS metric [48] in computing the DS,

\[
DS(I_1, I_2) = \sum_{i=1}^{n} \frac{1}{|A_i|} \sum_{p \in A_i} ||\omega_i \odot (x^i_1(p) - x^i_2(p))||_2^2, \tag{17}
\]

where \(n\) layers of unit-normalized features (in channel dimension), \(x^i_1\)'s, from a pre-trained DNN (e.g., the VGG network [54]) are used, \(|A_i|\) the spatial area of a feature map, and \(\omega_i\) the learned parameters by the LPIPS method. So, the higher the DS is, the better a generator model is. Similarly, the mean±std evaluation across multiple runs is used.

The Classification Accuracy Score (CAS) [49]. One long-term goal of generative learning in practice is to leverage synthesized images in training discriminative models. The CAS aims to verify how well a classification model trained only on synthesized images can perform on real testing images. So, the higher the CAS is, the better a generator model is. In contrast to the CAS, the classification accuracy metric used in the Layout2Im [20] is based on models trained with real image and tested on synthesized images, which may overlook the diversity of synthesized images.

### 3.4 Quantitative results

We first present the quantitative results and analyses. Table 1 and Table 2 summarize comparisons in terms of the our metrics in the two datasets.

At the resolution of 64 \(\times\) 64, our LostGAN-V1 obtains the best performance in comparison. It obtains slightly better Inception Score in both datasets and FID in the VG dataset than the Layout2Im. It obtains significantly better FID in the COCO-Stuff dataset (by more than 5 points reduction) and DS in both datasets. The diversity score of our LostGAN-V1 outperforms the Layout2Im by relative 288.9% and 277.8% in the two datasets respectively. There are a few other methods tested at the resolution of 64 \(\times\) 64 in the Layout2Im [20], including the pix2pixHD [14], BicycleGAN [58] and GauGAN [15], which are outperformed by the Layout2Im and thus not included here for the clarity of the table.

At the resolution of 128 \(\times\) 128, our LostGAN-V1 obtains better results than the Grid2Im method in the COCO-Stuff dataset, especially by more than 33 points reduction in FID and by relative 42.9% increase in DS. Our LostGAN-V2 further improves the results of LostGAN-V1, except for the IS and DS in the VG dataset. **Remarks.** We empirically observed that the VG dataset includes more diverse object configurations (e.g., bounding boxes may severely overlap in an image such as those for people, cloth and pants), and in general the bounding box annotations in the VG dataset are of lower quality than those in the COCO-Stuff dataset (e.g., they may have significant offsets for certain object instances). Those factors may affect the layout-to-mask component, especially the module of predicting masks from feature maps in the generator, which we think is the reason of LostGAN-V1 slightly outperforming LostGAN-V2 in the VG dataset. Similarly, Layout2Im+OWA [20] suffers a slight drop of performance in the VG dataset after introducing an object-wise attention mechanism to model shape of different objects. Considering these, we only test our LostGAN-V2 at higher resolutions than 128 \(\times\) 128.

At the resolution of 256 \(\times\) 256, our LostGAN-V2 also obtains better results than the Grid2Im method by more than 2 points increase in IS, 23 points reduction in FID and relative 61.8% increase in DS in the COCO-Stuff dataset.

At the resolution of 512 \(\times\) 512, there is no results from other baselines. Our LostGAN-V2 obtains better DS than the DS at the resolution of 256 \(\times\) 256. However, our LostGAN-V2 obtains slightly worse results than those obtained at the resolution of 256 \(\times\) 256 in terms of IS and FID. This phenomenon has been also observed in the BigGAN [5], which indicates, on the one hand, that more research are entailed to improve the quality of high resolution image synthesis, and on the other hand, that the models (Inception V3 pretrained in ImageNet at the resolution of 300 \(\times\) 300) used in computing IS and FID may need to change.

To compare the CAS, we train the ResNet-101 [24] on cropped and resized objects at a resolution of 32 \(\times\) 32 from generated images (five samples generated for each
layout in the testing set) and evaluate the trained model on objects cropped and resized from real testing images. We follow the widely used settings of ResNet-101 on the CIFAR-10/100 (with images at the resolution $32 \times 32$). We train a 171-category classification ResNet-101 in the COCO-Stuff dataset and a 178-category ResNet-101 in the VG dataset. For synthesized images at the three resolutions, our LostGANs obtain the best accuracy, often by large margin. These results are aligned with the higher DS results consistently obtained by our methods. Hopefully, with more research in the future work, we will be able to generate high-fidelity and high-resolution images from reconfigurable layouts and styles to facilitate more powerful discriminative learning, especially for handling some long-tail or corner situations.

We also compare with the state-of-the-art semantic-map-to-image method, the GauGAN [15]. Instead of using ground-truth semantic maps, we use the masks learned by our LostGAN-V2. Fig. 8 shows some examples, from which we can see the generator in our LostGAN-V2 works reasonably good, comparing to the GauGAN that are trained with ground-truth masks. Table 3 shows the comparisons in terms of IS, FID and DS. GauGAN obtains slightly better IS and FID than our LostGAN-V2, while our LostGAN-V2 achieves better DS.

### 3.5 Qualitative results

Fig. 6 and 7 show images synthesized by different models from the same layout in COCO-Stuff and VG respectively.
Fig. 10. Style Control: multiple samples generated from the same layout with different styles. Synthesized images have various visual appearance while preserving objects at desired locations. (a) Layout, (b) GT Image and (c-f) Synthesized images by our LostGAN-V2 256 × 256.

The input layouts are quite complex. Our LostGAN-V2 can generate visually more appealing images with more recognizable objects that are consistent with input layouts at resolution 256×256. We show more examples of layout and style control in our LostGAN-V2, in addition to Fig. 2.

In Fig. 9, layout control is demonstrated by adding object to, or moving a bounding box in a layout. When adding extra objects or moving the bounding box of one instance, our model can generate reasonable objects at desired position while keeping existing objects unchanged as we keep the input style of existing objects fixed. When moving the bounding box of one object, style of generated object in new position can also be kept consistent, e.g., in the top-right of Fig. 9, the person is moved while keep style feature like pose and color of clothes unaffected.

In Fig. 10, we show image-level style control of our model by synthesizing images with different visual appearance for a given layout while preserving objects at desired
locations. In Fig. 11, we show **object-level style control** of our model by gradually morphing styles of one instance in different images.

In Fig. 12, we show some selected examples of synthesized images at the resolution of $512 \times 512$. We observed that it is more difficult to generate realistic looking images at the resolution of $512 \times 512$.

### 3.6 Ablation Study

We conduct an ablation study on the mask generation component. Fig. 13 shows examples of learned masks, in which even for complex scene with multiple overlapping objects, synthesized images and learned masks are consistent and semantically reasonable. Compared to the input bounding boxes, the learned masks help reduce the semantic gap in layout-to-image. Those masks are learned jointly with image synthesis in a single generator in a weakly-supervised manner, verifying our proposed pipeline of simultaneously learning layout-to-mask-to-image.

Fig. 14 shows examples of mask refinement in the process of generation. The initial mask generation can produce reasonably good results, which are refined in the cascade of integrating masks learned from feature maps, especially for object boundaries (e.g., comparing (b) and (f)). This mask refinement is the main technical improvement between our LostGAN-V1 and LostGAN-V2, which also verifies the overall improvement by the updated LostGAN-V2 in experiments. To further investigate the effects of mask refinement, after training, we compare the performance of different models with some of mask refinement stages removed. As shown in Table 4, the last row shows the full model with all the mask components, $m_0 \cdots m_5$. In a backward way, if we remove the mask refinement stage by stage in the generator, the performance (Inception Score and FID) are indeed negatively affected. However, if we remove all the mask refinement stages and only use the initial masks, the performance is better than the model with mask refinement in the first stage, $m_0 + m_1$. One potential reason is that the resolution of first stage is very low, from which the learned masks may overlook objects of small sizes and introduce artifacts in the predicted masks. After observing this in the ablation study, we re-trained a model without using $m_1$ in COCO-Stuff and did not observe performance improvement, so we did not re-train all the models used in
Fig. 12. Some selected examples of synthesized images at the resolution of 512 × 512 in COCO-Stuff by our LostGAN-V2.

| Mask branch | IS (↑) | FID (↓) |
|-------------|--------|---------|
| $m_0$       | 16.68 ± 0.42 | 48.54 |
| $m_0 + m_1$ | 14.14 ± 0.33 | 63.96 |
| $m_0 + m_2$ | 17.10 ± 0.56 | 48.94 |
| $m_0 + m_3$ | 17.46 ± 0.34 | 44.38 |
| $m_0 + m_4$ | 17.51 ± 0.41 | 42.49 |
| $m_0 + m_5$ | 18.01 ± 0.50 | 42.55 |

TABLE 4
Effects of the mask refinement in our LostGAN-V2 256 × 256 in COCO-Stuff. $m_0$ represents initial masks generated from the joint label and style encoding. $m_i$ represents the refined masks at the $i$-th stage of the generator. See text for details.

TABLE 5
mIoU between masks and their nearest neighbor in ground truth masks.

| Category | person | car | plane | bus | train | truck | boat |
|----------|--------|-----|-------|-----|-------|-------|------|
| 53.8     | 66.5   | 68.0| 75.0  | 70.8| 66.1  | 63.1  |
| zebra    | 66.9   | 59.2| 77.7  | 62.3| 57.0  | 62.8  |

To investigate the quality of learned masks, we resort to the intersection-over-union (IoU) metric used in object semantic segmentation. We measure the IoU performance in the COCO-Stuff training dataset. We first crop masks for each category and then resize all the masks to the same resolution of 32 × 32. After training the LostGAN-V2 256 × 256, we run inference on each layout in the training dataset (one...
run is used for simplicity) and obtain the learned masks. We then crop and resize object masks in the same way as done for the ground-truth object masks. For each learned object mask, we retrieve the top-$k$ nearest neighbors in terms of mask IoU in the set of ground-truth object masks. Fig. 15 shows four examples with top-10 nearest neighbors. Table 5 shows the mean IoUs for 13 selected object categories which have reasonably high IoUs.

3.7 Failure Examples

Fig. 16 shows some failure examples. We observe that our proposed model is not capable of capturing interactions between person and small objects, e.g., person and baseball bat, tennis racket, handbag, etc. From the learned masks, we can also see why the model cannot synthesize good images. We leave this to our future work by investigating methods of learning fine-grained part-level masks.

Remarks. The generative learning of layout-to-image is still at an early stage of development in terms of synthesizing high-fidelity images, compared to the results of BigGANs [5] in ImageNet and StyleGANs [8] for faces. Overall, we can observe the quality of image generation from layout is still not sufficiently good, especially for articulated objects (such as people) and fine-grained object-object interactions at high resolution (e.g., examples in Fig. 16). In the meanwhile, we also note that the differences between the goals of BigGANs and StyleGANs and those of controllable layout-to-image are non-trivial. For example, we can use a trained BigGAN to generate cat images, and as long as the generated images look realistic and sharp with one or more than one cats, we shall think it does a great job (without requiring how many cats should appear and where they should be). Similarly, we can train a StyleGAN to generate face images, and we shall be happy if realistic and sharp face images are generated with a natural style (e.g., smiling or sad). Controllable layout-to-image has more fine-grained requirements. Those being said, based on the promising results of GauGANs [15] using annotated semantic maps in image synthesis, we think
Fig. 14. Examples of mask refinement in the generator. (a) Layouts, (b) Initial masks generated form the joint label and style encoding, (c-f) Mask refinement using masks generated from feature maps at different stages in the generator, (h) Synthesized images.

Fig. 15. Examples of learned masks and their nearest neighbors in the ground-truth masks in the COCO-Stuff training dataset: truck, airplane, hydrant and person (from top to bottom). (a) Masks learned by our LostGAN-V2 256 × 256 and (b-k) top-10 nearest neighbors. All masks are cropped and resized to the resolution of 32 × 32. See text for details.
the proposed layout-to-mask-to-image pipeline and LostGAN worth further explorations of seeking more powerful weakly-supervised learning of layout-to-mask. For example, we can develop more sophisticated mask generators and the “ToMask” modules in Fig. 4, and explore different consistency constraints between the “ToMask” modules, similar to the recently proposed PointRend method for improving Mask-RCNNs [59]. We leave those for the future work.

4 Conclusions

This paper studies the generative learning problem of layout-to-image with a focus on controllable image synthesis from reconfigurable layouts and styles. We first propose an intuitive pipeline of learning layout-to-mask-to-image. We then present a layout- and style-based architecture for generative adversarial networks (termed LostGANS). Our proposed LostGAN can be trained end-to-end to generate images from reconfigurable layout and style with strong style and layout controllability. Our proposed LostGAN can also learn fine-grained object masks in a weakly-supervised manner to bridge the gap between layouts and images by a novel object instance-sensitive layout-aware feature normalization (ISLA-Norm) scheme. State-of-the-art performance is obtained in the COCO-Stuff and Visual Genome datasets.

Acknowledgements

This work was supported in part by NSF IIS-1909644 and ARO Grant W911NF1810295. The views presented in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672–2680.
[2] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in International conference on Learning Representations (ICLR), 2016.
[3] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 7354–7363.
[4] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in International Conference on Learning Representations, 2018.
[5] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in International Conference on Learning Representations, 2019.
[6] T. Miyato and M. Koyama, “cGANs with projection discriminator,” in International Conference on Learning Representations, 2018.
[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” in International Conference on Learning Representations, 2018.
[8] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.
[9] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017, pp. 2642–2651.
[10] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, “Learning to discover cross-domain relations with generative adversarial networks,” in Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017, pp. 1857–1865.
[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.
[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2223–2232.
[13] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, “Multimodal unsupervised image-to-image translation,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 172–189.
[14] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-resolution image synthesis and semantic manipulation with conditional gans,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.
[15] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, “Semantic image synthesis with spatially-adaptive normalization,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2337–2346.
