# Scene Style Network (SSN): Disentangling Layout and Texture for Image Synthesis

Kevin Tan, Ehsan Adeli, Juan Carlos Niebles

Stanford University
{kevintan,eadeli,jniebles}@cs.stanford.edu

**Abstract.** Controllable image synthesis is challenging because it requires an understanding of both the layout and texture of a visual scene. Previous works that perform layout-to-image generation offer control over the layout, but lack the ability to simultaneously control the style. In this work, we propose Scene Style Network (SSN), which explicitly separates between a *layout pathway* and *texture pathway* during the synthesis. The layout pathway obtains a layout representation given an input scene graph with a graph convolution network and mask network. The texture pathway leverages a joint layout-texture embedding space to generate the texture at multiple resolution scales. In this way, SSN enables improved layout-texture awareness for conditional image synthesis. We demonstrate that the learned joint layout-texture latent space exhibits higher disentanglement compared to that of prior works. Additionally, we evaluate the quality of generated images and demonstrate that SSN outperforms several baselines on the Visual Genome dataset.

## 1 Introduction

People understand the visual world as a sum of its parts. Our effortless mental ability to simulate and imagine what will happen crucially depends on a scene representation that is compositional with respect to objects and their relationships [31,38]. In cognitive science, structural description models represent visual concepts as compositions of parts and relationships, which provide a strong inductive bias for constructing models capable of reasoning about new objects [13]. However, current deep approaches to image synthesis struggle with generating realistic images without introducing unwanted artifacts. In this work, we argue that the compositionality of objects and their relations are crucial for image synthesis in a manner that offers control over both the layout and texture.

Controllable image synthesis is a key task in computer vision research with applications in image editing [46], 3D scene understanding [10], and inverse graphics [28]. A key challenge in controllable image generation is to learn representations that exhibit high disentanglement between the high-level contents of the scene (layout) and the fine-grained details (texture). Previous holistic-based works either disregard the texture when generating images from layout, or disregard the layout when generated styled images. Class-conditional GANs such as ACGAN [34] or BigGAN [4] are not able to interpolate between latent factors

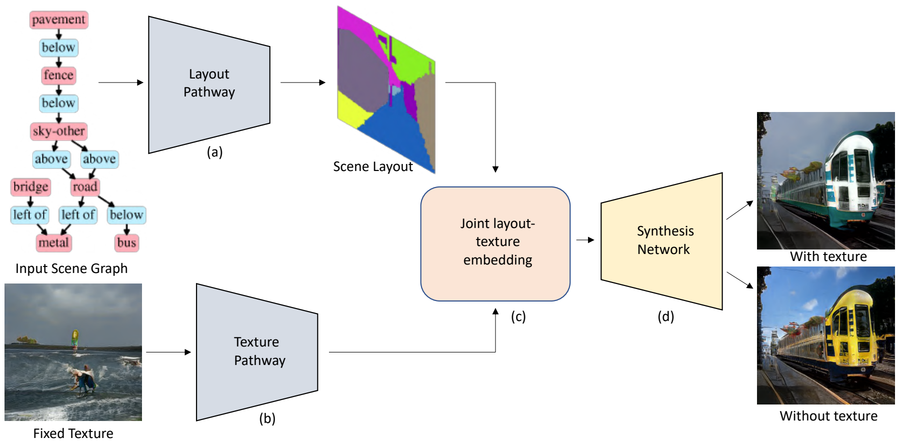


Fig. 1: Scene Style Network (SSN) takes as input a *scene graph* and *texture* and outputs a corresponding image that respects the spatial layout while preserving the visual texture. The (a) *layout* pathway obtains a layout with a graph convolution network and mask network $M$ from an input scene graph. The (b) *texture* pathway takes in a texture and produces a (c) joint layout-texture embedding code $\mathbf{w} \in \mathcal{W}$. The synthesis network (d) leverages $\mathbf{w}$ to fix the texture at multiple resolution scales to synthesize the output image.

of variation without changing the visual texture. Recently, Locatello et al. [29] demonstrated that unsupervised disentanglement learning is impossible without an additional model inductive bias. This motivates us to explore the inductive bias of compositional layout-awareness in order to disentangle the object layout from visual textures. With these insights, we observe that layout and texture should be decoupled from one another, and bridge the gap between them to demonstrate more controllable image synthesis.

To this end, we propose Scene Style Network (SSN) (Figure 1), a novel framework for image generation with control over layout and texture. The main technical innovation is to learn a joint layout-texture intermediate embedding space (Figure 1c) that compactly encodes the fine-grained details with spatial awareness. Given an input scene graph, the *layout* pathway (Figure 1a) obtains the scene layout with a graph convolution network (GCN) and mask network. The *texture* pathway (Figure 1b) samples a latent texture code $\mathbf{z}$, extracts a layout embedding with the *scene layout extractor* (Figure 4a), and passes the concatenation of the two latent codes through the mapping $f : \mathcal{Z} \to \mathcal{W}$ (Figure 4b) to obtain an intermediate joint layout-texture latent code $\mathbf{w} \in \mathcal{W}$. Then, the *synthesis network* (Figure 1d) injects $\mathbf{w}$ into progressively growing synthesis blocks (Figure 4c) to synthesize the output image. Crucially, the layout and textures are explicitly decoupled until the joint embedding. In this way, SSN automatically learns which textures align with which regions in the spatial layout, to generate a plausible output image satisfying the layout and texture constraints.

We perform several experiments to demonstrate the effectiveness of SSN. The generated images achieve comparable state-of-the-art performance on image generation tasks on the Visual Genome [27] dataset, a structured knowledge-base dataset of images with scene graphs. We show that when the mask network is trained on the Coco-Stuff [5] dataset, it is an effective way to bootstrap the training of the *synthesis network*. Style mixing experiments demonstrate that SSN is capable of indepdently sampling generated images that fixate on a single texture. In addition, SSN shows an advantage over prior works such as StyleGAN [23] by learning a joint layout-texture embedding space $\mathcal{W}$ that exhibits higher disentanglement as measured by the Perceptual Path Length [23].

To summarize our main contributions: (1) we introduce Scene Style Network (SSN) which consists of a *layout* pathway that processes the input scene graph via graph convolution, and a *texture* pathway that synthesizes the image by injecting style into progressively growing upsampling blocks; (2) we demonstrate that the learned joint layout-texture embedding space enables controlled image synthesis with respect to an input scene graph and fixed texture; (3) we propose a modification to Perceptual Path Length (PPL) [23] that is better suited for measuring the disentanglement of learned latent space in natural scenes as opposed to human faces; (4) we show that SSN outperforms previous baselines on image generation and disentanglement evaluations.

## 2 Related Work

**Conditional image synthesis** Conditional image generation is the task of generating images conditioned on an additional input source. Pix2pix [18] is a fully supervised method that requires aligned pairs of samples from two domains. Generative adversarial networks (GANs), autoregressive models, and variational autoencoders (VAEs) have achieved great success in conditional image generation. Various methods study how to feed the condition the additional input, e.g., class information [33,34,47], source image [25], or a text description [36,48]. In this work, we condition on scene layout by obtaining an intermediate joint layout-texture embedding before performing the generator forward pass.

**Deep image manipulation** Recent advancements in image manipulation by deep neural networks have enabled various image editing tasks such as style transfer [11], image-to-image translation [52], automatic colorization [49], and 3D-aware scene editing [28]. Most similar to our work is that of Ashual et al. [2] where the proposed method for image generation from scene graphs includes an intermediate layout embedding and appearance embedding. While they demonstrate the ability to import elements from other images and selecting appearance archetypes given user input, the manipulated synthesis results are sensitive to changes in texture, and often introduce unwanted artifacts.

**Deep learning on graphs** There are some works which propose to learn feature embeddings for graph nodes given a single large graph [12,35,45]. This is similar to word vector embeddings, e.g., word2vec [32], which are typically employed for natural language processing tasks. Our work uses a scene graph processing pipeline similar to Johnson et al. [20]. The main focus of our work is, however, not on how to obtain the scene layout, but rather how to use it to obtain a joint layout-texture embedding.

**Disentanglement learning** Learning disentangled representations in an unsupervised or self-supervised manner has recently received great attention. InfoGAN [7] presents an information-theoretic extension to GANs demonstrating learned disentangled factors. $\beta$-VAE [15] proposes a modification of the variational autoencoder (VAE) framework by introducing a weighting factor to balance the likelihood and KL divergence. However, these methods do not guarantee that the learned latent factors are semantically meaningful in a way that gives fine-grained control of the generation. The analysis of Locatello et al. [29] show that fully unsupervised disentanglement learning is impossible without the addition of a model inductive bias. In this work, we build upon this hypothesis by inducing the bias of explicitly decoupling the layout and texture before the joint layout-texture embedding.

**Image generation from layout** Several works have proposed to generate images consistent with layout. Some previous methods use the layout information as an intermediate representation between other input sources, e.g., text [16] and scene graphs [20], and the output image. Others use the layout as a complementary feature for image generation based on contextual text [21, 37, 44] or shape and lighting [8]. Typically, as done in [16] and [20], the scene layout is obtained through a multiple stage pipeline in which a semantic layout with bounding boxes and object masks is first obtained, then a separate image generator is used to reconstruct the layout to an RGB image.

Recently, [51] achieves state-of-the-art reconfigurable layout-to-image generation, but focuses on low-resolution images of $64 \times 64$ and requires a computationally expensive convolutional LSTM; [42] also focuses on low-resolution images and is unable to control the textures of the generated image. In contrast to these works, our method is able to (i) condition on an additional texture to obtain a joint layout-texture embedding, (ii) subsequently inject the layout-texture into a desired resolution during the progressively-growing synthesis (inspired by [23]), and (iii) quantitatively demonstrate a disentangled intermediate latent space.

## 3 Scene Style Network

We propose Scene Style Network (SSN) which generates a realistic image corresponding to the constraints of two inputs: (1) a *scene graph* describing objects and their relationships, and (2) an input *style* representing the texture of the desired image. Figure 1 provides an overview of the proposed method. The technical challenges are three-fold: first, we must develop a method for processing the graph structure to obtain a dense encoding that retains spatial structure; second, we must ensure that the layout is preserved throughout the each resolution of the progressive generator network; third, we must ensure that the textures are

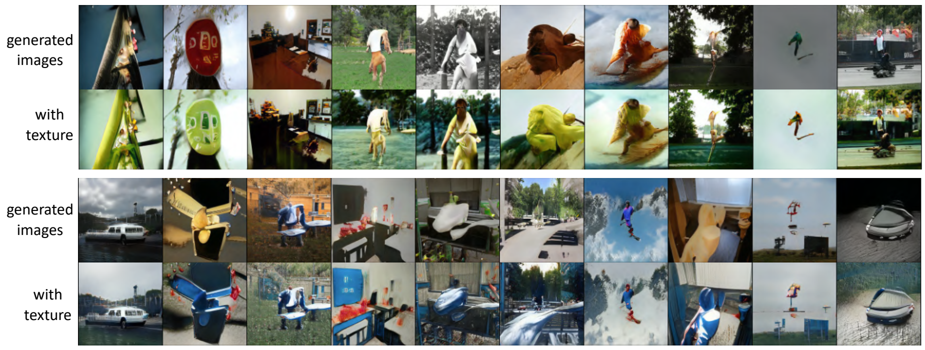


Fig. 2: The diversity across generated images and fixed texture obtained from our method on 256 × 256 Visual Genome generated images ($n = 4$). A texture $z \sim \mathcal{N}(0, 1) \in \mathbb{R}^{512}$ is fixed for each sample across the columns.

preserved in the output image. In the following, we will describe the architecture overview of SSN (Figure 1).

## 3.1 Layout Pathway

The goal of the layout pathway (Figure 1a) is to obtain a scene layout from the input scene graph. We describe, in turn, the architectural details of the scene graph input representation, graph convolution network, and mask network for obtaining the layout. See Figure 3 for an overview.

**Scene graphs** A *scene graph* [20] is a structured representation of an image, where nodes in a scene graph correspond to object bounding boxes with their object categories, and edges correspond to their pairwise relationships between objects. Formally, a *scene graph* is defined by a 3-tuple set $G = \{B, O, R\}$:

- $B = \{b_1, b_2, ..., b_n\}$ is the region candidate set, with elements $b_i \in \mathbb{R}^4$ denoting the bounding box of the $i^{th}$ region
- $O = \{o_1, o_2, ..., o_n\}$ is the object set, with element $o_i \in \mathbb{N}$ denoting the corresponding class label of region $b_i$
- $R = \{r_1, r_2, ..., r_m\}$ of pairwise relationships between those objects, where $r_k$ denotes a triplet of a start node $(b_i, o_i) \in B \times O$, an end node $(b_j, o_j) \in B \times O$, and a relationship label $x_{i \to j} \in \mathcal{R}$. $\mathcal{R}$ is the set of all possible predicate types.

For every image $I$ and the corresponding scene graph $G$, we obtain the object embeddings $e_{\text{obj}}$ and relation embeddings $e_{\text{rel}}$ via two embedding layers of dimensionality 128. Similar to processing input text in neural language models, the embedding layers are lookup tables where the keys are the the object or relation index and the values are the object or relation dense vectors of size 128.

**Graph convolution network** We utilize *graph convolution networks* [20, 40] (GCN) to aggregate contextual spatial information over the output of the relation proposal network. The purpose of the GCN is to transform the node

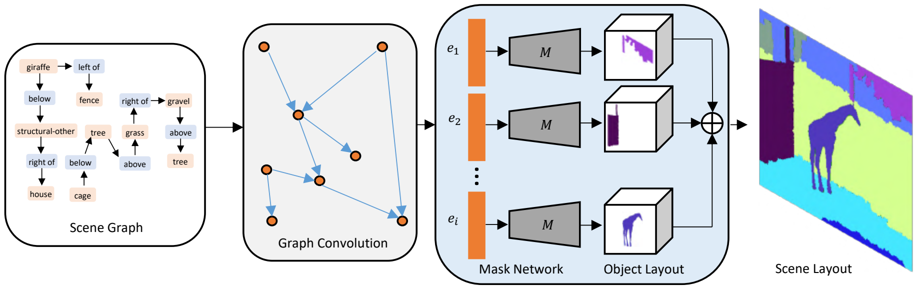


Fig. 3: Overview of the *layout pathway* (similar to [20]) which obtains a scene layout from scene graphs. The input scene graph is processed with a graph convolutional network that aggregates per-object contextual embeddings by passing along information of its neighboring edges. The object embeddings are then used to regress bounding boxes, predict masks, and compute a global scene layout.

embeddings into a new set of context-aware embeddings. For fair comparisons, we follow the same high-level architecture as proposed in [20]. Specifically, given input vectors $e_{\text{obj}}, e_{\text{rel}} \in \mathbb{R}^{D_{\text{in}}}$ for all objects $o_i \in O$ and edges $(o_i, r, o_j) \in E$, the new nodes and edges are computed using three functions $g_s, g_p, g_o$ which predict the subject $o_i$, predicate $r$, and object $o_j$, respectively. The forward pass implements $g_s, g_p, g_o$ with a 2-layer MLP that passes a concatenation of the triples input $[o_i, r, o_j]$ through a hidden layer with dimensionality 512 followed by three output heads. Since some objects can exist in many relationships, we address this by utilizing average pooling over the object embeddings.

**Mask network** The mask network [20] takes in the location embeddings of each object and feeds it into a *mask regression network* which predicts object pseudo-binary masks $\hat{m}_i \in M \times M$ as well as a *bounding box regressor* which outputs bounding box $\hat{b}_i = [x_1, y_1, x_2, y_2]^T \in [0,1]^4$ from a MLP. Note that the bounding box represents the coordinates as ratio of the image dimensions and not the absolute positions. To avoid overfitting, we follow [2] in injecting per-object random noise $z_i \sim \mathcal{N}(0,1) \in \mathbb{R}^{64}$ as additional input to the mask network $M$ to generate stochasticity in the masks. The mask regression network terminates with a sigmoid nonlinearity to force the masks to lie in range $(0,1)$.

To compute the object layout $t_i$ for a single object $o_i$, we shift and scale the mask $\hat{m}_i$ according to the ratio of bounding box $\hat{b}_i$ resulting in a mask $m'_i \in \mathbb{R}^{H \times W}$. Then, we compute the appearance embedding $a_i$ with the *appearance network* which is a CNN that extracts feature maps with output dimensionality set to 32 from the cropped image containing object $i$. The object layout $t_i$ is then the result of the tensor product between $m'_i$ and $a_i$.

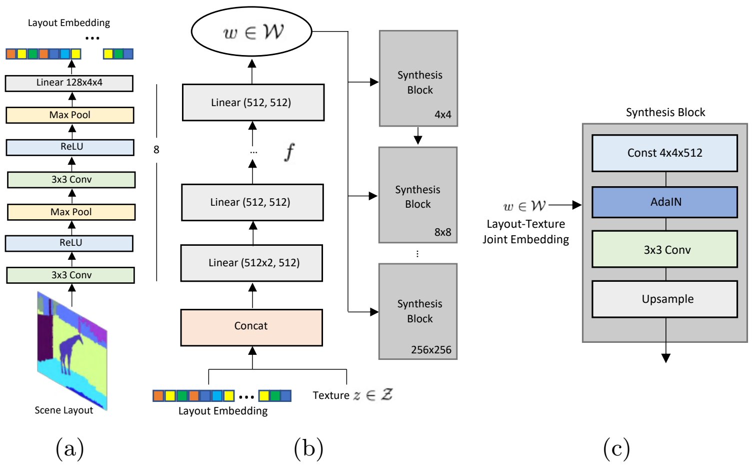


Fig. 4: Overview of the *texture pathway* which generates an image given a joint layout-texture latent code. The mapping $f$, parameterized by 8 fully-connected layers, learns a joint layout-texture embedding to produce $\mathbf{w} \in \mathcal{W}$. **(a)** *Scene layout extractor* obtains a compressed dense encoding of the scene that retains spatial structure. **(b)** *Synthesis network* injects the layout-texture joint embedding vector to progressively-growing [22] synthesis blocks. **(c)** *Synthesis block* for an arbitrary resolution to iteratively upsample the generated image.

## 3.2 Texture Pathway

The goal of the texture pathway (Figure 1b) is to generate an image given a joint layout-texture latent code $\mathbf{z} \in \mathcal{Z}$. To do this, we first obtain the joint layout-texture embedding via the *scene layout extractor*, then synthesize the output image by fixing the texture at multiple *synthesis blocks* of varying resolutions. Figure 4 provides a schematic overview, and Figure 2 demonstrates how the texture pathway is able to perform multiple layout-to-image generations while fixing control over the texture. Note that the image corresponding with the fixed texture in Figure 1 is for illustrative purposes only. In practice, we sample a latent code $\mathbf{z} \in \mathcal{Z}$ that represents the visual texture of the illustrated image.

**Scene layout extractor** We introduce a novel *scene layout extractor* module (Figure 4a) for obtaining a dense encoding of a scene layout that retains spatial structure. Specifically, it takes as input a scene layout $T = \sum_{i=1}^{C} t_i$ as a global layout representation for all objects, and outputs a latent code $\hat{\mathbf{z}} \in \mathbb{R}^{512}$. The scene layout $T$ is passed through two consecutive downsampling blocks which each consist of $3 \times 3$ convolutional layers, ReLU nonlinearity, and max pool layer, followed by a final linear layer with input size $128 \times 4 \times 4$ which outputs a latent embedding $\hat{\mathbf{z}} \in \mathbb{R}^{512}$ to be fed as input to the *synthesis network*.

**Joint layout-texture embedding** The *joint layout-texture embedding* (Figure 1c) aggregates the spatial structure from the layout pathway and fine-grained details from the texture pathway into a joint latent space. Importantly, the texture pathway is decoupled from the layout pathway up until this point.

## 3.3 Synthesis Network

**Generator** Given the scene layout and a style texture, we must synthesize an image that respects the layout while retaining the textures. The input to the generator is $\mathbf{z}' = [\mathbf{z}; \hat{\mathbf{z}}]$ where $\mathbf{z} \sim \mathcal{N}(0, 1)$ and $\hat{\mathbf{z}}$ is obtained via the scene layout extractor, and ; denotes concatenation along the feature axis. Then, $\mathbf{z}'$ is fed into the mapping network $f : \mathcal{Z} \to \mathcal{W}$ to obtain an intermediate latent encoding $\mathbf{w} \in \mathcal{W}$. In our experiments, $f$ is an 8-layer MLP. For fair comparison to StyleGAN, we set the embedding dimensionality of both $\mathcal{Z}$ and $\mathcal{W}$ to 512.

To synthesize the output image in RGB from $\mathbf{w}$, we use a *synthesis network* consisting of resolution-specific *synthesis blocks* inspired by StyleGAN [23], a style-based architecture for generative adversarial networks. Our generator augments StyleGAN to perform layout-conditional generation and learns a joint intermediate embedding space $\mathbf{w} \in \mathcal{W}$. We follow similar architecture design choices for the synthesis network (Figure 4b) and synthesis blocks (Figure 4c). The synthesis network $g$ consists of 2 layers per resolution from $4 \times 4, 8 \times 8, ..., 256 \times 256$. Joint layout-texture latent codes $\mathbf{w}$ control the synthesis blocks through adaptive instance normalization (AdaIN) [17] at each convolution layer. The final layer converts the output to RGB with a $1 \times 1$ convolution.

**Discriminators** Our method employs three discriminators:

- The *mask discriminator* $D_{\text{mask}}$ uses a Least Squares GAN (LSGAN) [30] conditioned on the object class $c_i$. The loss associated with the mask discriminator is given by:

$$\mathcal{L}_{D_{\text{mask}}} = [\log D_{\text{mask}}(m_i, c_i)] + \mathbb{E}_{z \sim \mathcal{N}(0,1)}[\log(1 - D_{\text{mask}}(M(e_i, z), c_i] \quad (1)$$

  where $m_i$ is the real mask of object $i$, $M(\cdot)$ is the *mask network*, $e_i$ is the per-object embedding.
- The *image discriminator* $D_{\text{image}}$ ensures that the generated image is realistic by using a multi-scale LSGAN to compute the sum of $\mathcal{L}_{D_{\text{image}}}$ across full and half scales. The loss of $D_{\text{image}}$ is a combination of the real loss and fake loss:

$$\mathcal{L}_{D_{\text{image}}} = \log D_{\text{image}}(I, t) - \log(1 - D_{\text{image}}(I', t)) \quad (2)$$

  where $I$ is the real image, $I'$ is the fake image, and $t$ is the layout.
- The *object discriminator* $D_{\text{object}}$ classifies each object as real or fake by cropping the input pixels and rescaling using bilinear interpolation [19]:

$$\mathcal{L}_{D_{\text{object}}} = \sum_{i=1}^{n} \log D_{\text{object}}(I'_i) - \log D_{\text{object}}(I_i) \quad (3)$$

## 4 Training

We employ a three-stage training strategy: first, we train the *layout pathway* which obtains a scene layout from a scene graph with an autoencoder network $R$ similar to [2]; second, training the *texture pathway* by replacing $R$ with the proposed *synthesis network*; third, jointly training the proposed *scene layout extractor* with the mapping network $f$. We use the autoencoder network $R$ to bootstrap the first stage of training. The layout feature embeddings extracted at the first layer of the information bottleneck of $R$ are used as input to the *scene layout extractor*. The total loss used to optimize the networks is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{pix}} + \lambda_2 \mathcal{L}_{\text{box}} + \lambda_3 \mathcal{L}_{\text{percept}} + \lambda_4 \mathcal{L}_{D_{\text{mask}}} + \lambda_5 \mathcal{L}_{D_{\text{image}}} + \lambda_6 \mathcal{L}_{D_{\text{object}}} \quad (4)$$

where $\mathcal{L}_{\text{pix}} = ||I - \hat{I}||_1$ is the $l_1$ difference between input and generated images, $\mathcal{L}_{\text{box}} = \sum_{i=1}^{n} ||b_i - \hat{b}_i||_1$ is the $l_1$ loss that penalizes the differences between the ground truth box and predicted box, and $\mathcal{L}_{\text{percept}} = \sum_{u \in U} \frac{1}{u} ||VGG^u(I) - VGG(I')||_1$ is the perceptual loss [50] which compares the $l_1$ between activations of a $VGG$ network at layer $u$, summed over all layers $u \in U$.

## 5 Experiments

In our experiments, we aim to show that our method generates images of complex scenes satisfying the object and relationship specification given by the input scene graph. Our experiments aim to answer the following questions: (1) How does the proposed joint layout-texture latent space $\mathcal{W}$ contribute to controllable synthesis? (2) How well does the object discriminator $D_{\text{object}}$ enforce that generated objects look one-by-one real? (3) How do the generated images compare with conditional image generation baselines on Visual Genome? In this section, we first discuss the datasets (Section 5.1) and implementation details (5.2), followed by both the qualitative results (Section 5.3) on the tasks of image generation and style mixing, and lastly the quantitative evaluations on image generation (Section 5.4) and disentanglement (Section 5.5).

### 5.1 Datasets

**Visual Genome** We experiment on Visual Genome (VG) [27] version 1.4 which comprises 108,077 images annotated with scene graphs. Following [20], we divide the data into 80% train, 10% val, and 10% test, resulting in 62,565 train, 5,506 val, and 5,088 test images. We use object and relationship categories occurring at least 2000 and 500 times respectively in the train set, leaving 178 object and 45 relationship types. The number of objects in the image ranges from 3 to 30.

**Coco-Stuff** We use the Coco-Stuff [5] dataset at resolution of $256 \times 256$ to train the mask network in the first stage of training since VG does not contain the ground truth masks. Using the same split as previous works [51], we partition the train, valid, and test sets to 25,972, 1,024, and 2,048, respectively. There are 171 objects, with roughly 3 to 8 objects per image.

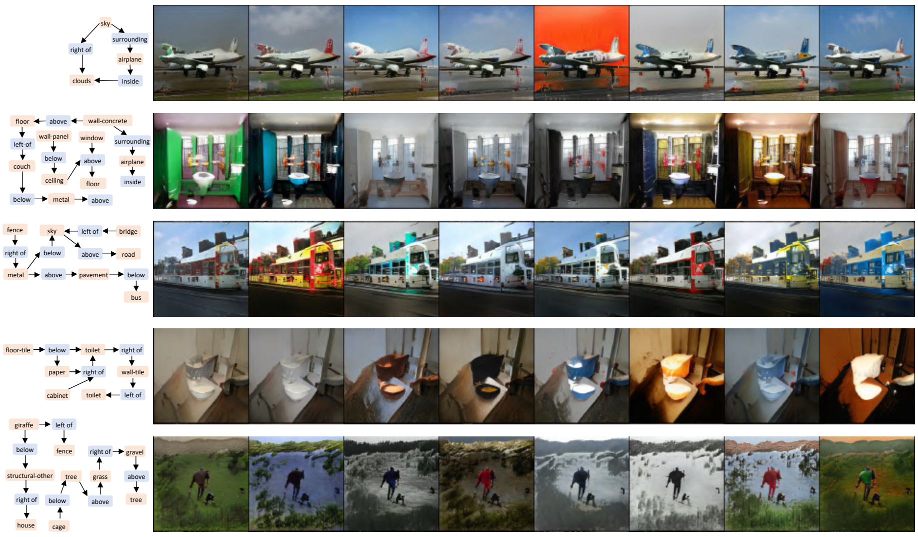


Fig. 5: Image generation results from an input scene graph on Visual Genome. For each example, we generate 10 samples to demonstrate control over the colorization. From an input scene graph, SSN is able to control both the spatial layout as well as the fine-grained textural details.

## 5.2 Implementation Details

In our experiments, the generator network shares the architecture of StyleGAN [24] aside from the mapping network $f$. We train on VG with resolution $256 \times 256$ and batch size 32. Total training time takes more than 6 days on 4 Titan RTX resulting in more than 8.3M images processed. We use Adam [26] optimizer with initial learning rate 0.002. We use an exponential moving average of the weights of the generator $G_{\text{ema}}$ with a channel multiplier of 2 between synthesis blocks. To learn the joint layout-texture embedding, we freeze the weights in synthesis blocks and discriminator and finetune the mapping network $f : \mathcal{Z} \rightarrow \mathcal{W}$, with the first dense layer containing $1024 \times 512 + 512 = 524,800$ parameters.

## 5.3 Qualitative Results

**Image Generation** Fig 5 shows qualitative image generation results from an input scene graph on Visual Genome ($256 \times 256$). Our model is able to synthesize an image consistent with the input scene graph while also retaining control over the fine-grained textural details, as shown by the diversity across the columns. In contrast to [51] and [42], we generate images at higher resolution and show sharper object boundaries when sampling multiple times for a single layout. In every row, the scene layout remains unchanged and the objects do not show morphing artifacts across independent samples of the texture $\mathbf{z} \in \mathcal{Z}$. This

Table 1: Evaluating the quality of generated images on Visual Genome (VG) with the Inception Score (IS) and Fréchet Inception Distance (FID) scores.

| Method | Resolution | IS | FID |
|---|---|---|---|
| Real Images | $64 \times 64$ | $13.9 \pm 0.5$ | — |
| pix2pix [18] | $64 \times 64$ | $2.7 \pm 0.02$ | 142.86 |
| sg2im [20] (GT Layout) | $64 \times 64$ | $6.3 \pm 0.2$ | 74.61 |
| layout2im [51] | $64 \times 64$ | $8.1 \pm 0.1$ | 31.25 |
| LostGAN [42] | $64 \times 64$ | $8.7 \pm 0.4$ | 34.75 |
| Real Images | $128 \times 128$ | $20.5 \pm 1.5$ | — |
| LostGAN [42] | $128 \times 128$ | $11.1 \pm 0.6$ | 29.36 |
| Real Images | $256 \times 256$ | $29.3 \pm 2.1$ | — |
| SSN (*ours*) (No concat) | $256 \times 256$ | **$17.0 \pm 1.1$** | **32.51** |
| SSN (*ours*) | $256 \times 256$ | **$21.6 \pm 1.6$** | **24.11** |

demonstrates the ability of SNN to control texture independently of layout. The converse, i.e., the ability to control layout independently of texture, is demonstrated in Figure 2, where in rows 2 and 4, the texture remains unchanged across various layouts. Together, these two demonstrate how the explicit decoupling of the layout and texture pathways enable improved control of the synthesis.

**Style Mixing** We perform style mixing experiments to demonstrate how varying style controls meaningful high-level attributes of the output image. To generate a style mixed image, we generate two random latent codes $\mathbf{z}_1, \mathbf{z}_2$ and obtain corresponding intermediate latent codes $\mathbf{w}_1, \mathbf{w}_2$ via the mapping network $f$ parameterized by an 8-layer MLP. The latent codes $\mathbf{w}_1, \mathbf{w}_2$ control the styles through the synthesis network by applying $\mathbf{w}_1$ to styles before a chosen injection index and applying $\mathbf{w}_2$ after. Specifically, the injection index $j \in \{1, ..., \log(N) * 2 - 2\}$ is a function of the max resolution size of the output image, e.g., $N = \{256, 512, 1024\}$.

Fig 6 shows the result of style mixing on $N = 256$ resolution images from Visual Genome. Given an batch of input scene graphs ($n = 8$) and a fixed texture $\mathbf{z} \in \mathcal{N}(0, 1) \in \mathbb{E}^{512}$, we generate $n$ style-mixed images for each injection position. From these examples, we observe that we can modify a generated image to a desired texture without substantially manipulating the scene layout, especially for the middle injections (rows 6-9). For example, the plane in row 7, column 3 retains the foreground content while successfully manipulating the background. Early injections (rows 1-5) may change the contents of the scene but the spatial layout remains in tact. Late injections (rows 10-12) do not change high-level contents, but rather small fine-grained details such as color. Note that injection of texture in the last position fully reconstructs the generated sample.

### 5.4 Evaluating Image Generation

**Metrics** We measure the image quality based on the Inception Score (IS) [39] and Fréchet inception distance (FID) Score [14]. The Inception Score measures

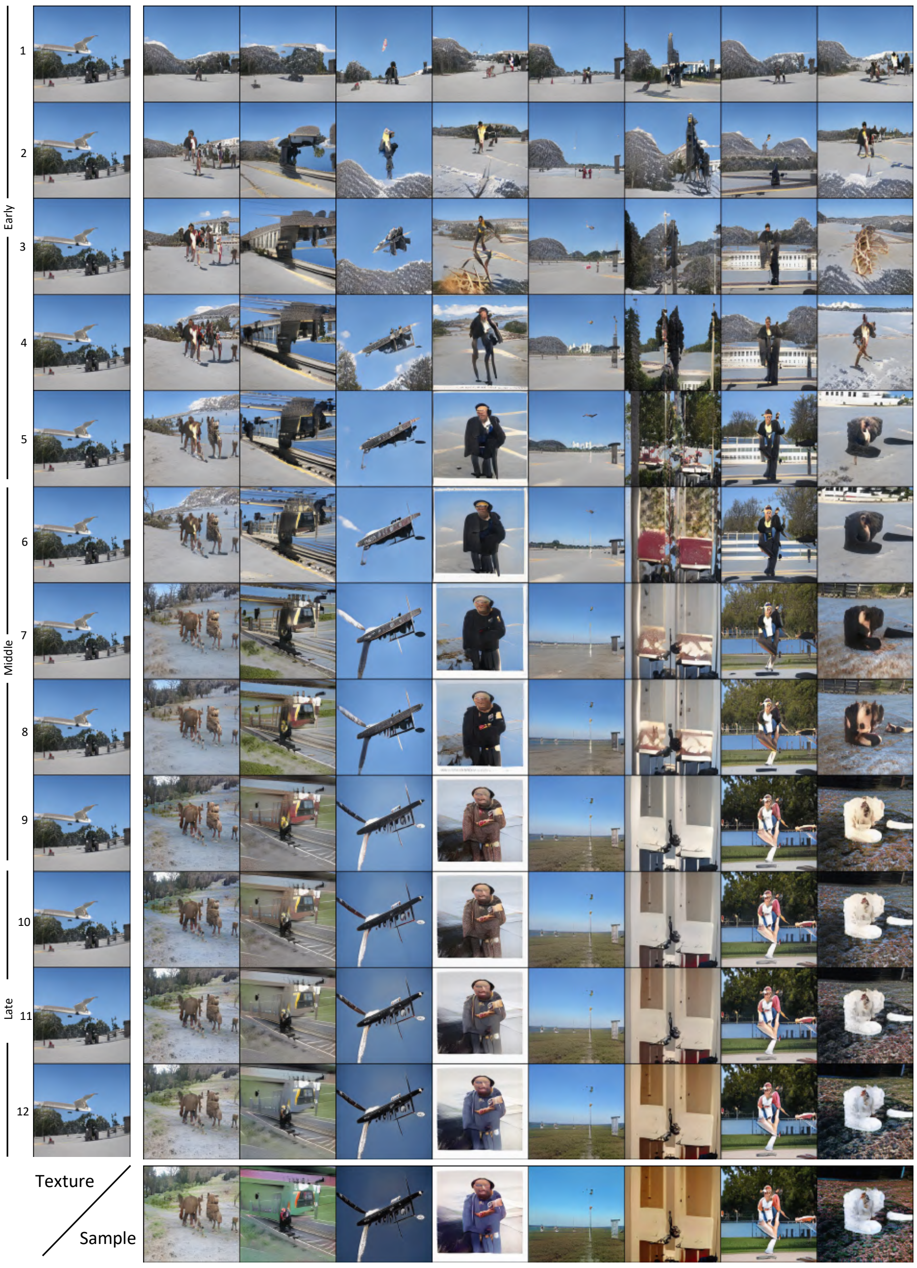


Fig. 6: Style mixing on 256 × 256 images from Visual Genome. Given a fixed texture (leftmost column), we generate $n = 8$ independent samples from the *synthesis network* by applying the texture across varying injection points corresponding to progressively growing resolutions.

both the quality of the generated images as well as their diversity. Following previous literature, a pre-trained Inception [43] network is used for computing feature activations. Higher Inception Scores are better. The FID score measures the distance between the distributions of the generated and real images when modelled as multivariate Gaussians. Fréchet inception distance is more robust to noise than Inception Score. Lower FID scores are better. Despite some flaws in Inception Score as pointed out by Barratt and Sharma et al. [3], we nevertheless employ these two evaluation metrics as approximate measures of sample quality due to their popularity and for fair comparison against prior works.

**Results** We show quantitative results on Inception Score (IS) and Fréchet Inception Distance (FID) Score on Visual Genome at the $64 \times 64$, $128 \times 128$, and $256 \times 256$ resolutions in Table 1. The results from prior works [18,20,42,51] are listed at the resolutions in which they are reported in the literature. We also report the IS on real images for comparison. For calculating FID, we use 50,000 samples and batch size of 32 due to memory constraints. The proposed method (SSN) outperforms several baselines on all metrics across two settings: with and without the concatenation of $\mathbf{z}' = [\mathbf{z}; \hat{\mathbf{z}}]$. Note that in this experimental setting without a fixed texture, choosing to not use concatenate the layout embedding is equivalent to StyleGAN [23]. We observe significantly better results when conditioning the synthesis network on the layout embedding. These experiments demonstrate the effectiveness of SSN on conditional image generation.

## 5.5 Evaluating Disentanglement

**Metrics** A natural desirable trait for learned generative models is for the latent space to exhibit high disentanglement, i.e., a latent space which consists of linear subspaces, each of which controls a single semantically-meaningful factor of variation. There are several definitions of disentanglement [1,6,9], but in this work we measure disentanglement with a modification of Perceptual Path Length (PPL) as proposed by [23]. Intuitively, since it should be easier to generate realistic images from a disentangled representation than an entangled representation, we posit that the generator network is incentivized to favor learning semantically-meaningful linear factors of variation through the mapping $f : \mathcal{Z} \to \mathcal{W}$. Lower PPL scores are a measure of a more disentangled space.

To compute the perceptual path length in $\mathcal{Z}$, we fix the scene layout for a single batch for fair comparison of images generated from interpolations. The perceptual path length of $\mathcal{Z}$ over all possible endpoints and $n$ layouts is:

$$l_{\mathcal{Z}} = \mathbb{E}\left[\frac{1}{\epsilon^2} d(G(\mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2; t), T), G(\mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2; t + \epsilon), T))\right] \quad (5)$$

where $\mathbf{z}_1, \mathbf{z}_2 \sim P(\mathbf{z}), t \sim U(0, 1)$, $T$ is the scene layout, `slerp` denotes spherical interpolation, and $G$ is the generator network. The distance metric $d(\cdot, \cdot)$ evaluates the perceptual distance between the resulting fake images given by a perceptually-based pairwise image distance [50] calculated via a weighted difference between two VGG16 [41] embeddings. Since latent vectors in $\mathcal{W}$ are not

Table 2: Perceptual Path Length (PPL) results on Visual Genome with resolution $256 \times 256$. Lower PPL is better for latent space $\mathcal{W}$. Higher PPL is better for latent space $\mathcal{Z}$.

| Method | Space | Crop | PPL |
|---|---|---|---|
| StyleGAN [23] | $\mathcal{W}$ | ✓ | 326.7 |
| StyleGAN [23] | $\mathcal{W}$ | ✗ | 302.4 |
| SSN (*ours*) | $\mathcal{W}$ | ✓ | **311.8** |
| SSN (*ours*) | $\mathcal{W}$ | ✗ | **291.4** |

| Method | Space | Crop | PPL |
|---|---|---|---|
| StyleGAN [23] | $\mathcal{Z}$ | ✓ | 769.2 |
| StyleGAN [23] | $\mathcal{Z}$ | ✗ | 737.1 |
| SSN (*ours*) | $\mathcal{Z}$ | ✓ | **819.7** |
| SSN (*ours*) | $\mathcal{Z}$ | ✗ | **738.2** |

normalized in any way, we compute perceptual path lengths $l_{\mathcal{W}}$ for $\mathcal{W}$ using linear interpolation instead of spherical interpolation.

**Results** Table 2 shows the perceptual path lengths (PPL) on Visual Genome with resolution $256 \times 256$. We approximately compute the expectation by taking 5000 samples with a batch size of 64 and $\epsilon = 1 \times 10^{-4}$. In contrast to [23], we perform experiments without cropping the generated images to prefer a face prior since we are dealing with natural scenes and not human faces.

Our method demonstrates lower path lengths $l_{\mathcal{W}}$ and higher disentanglement than [23] in $\mathcal{W}$. Note that for fair comparison, we control the dimensionality of $\mathbf{w} \in \mathcal{W}$ to be 512 in all experiments. We do not make any changes to the synthesis network besides the weights and bias of the first fully-connected layer. This demonstrates a smoother manifold of $\mathcal{W}$ when fixed on a particular scene layout. We also show $l_{\mathcal{Z}}$ and demonstrate that the latent space $\mathcal{Z}$ of our method is more entangled. This is expected as our strategy for performing conditional generation is to simply concatenate a latent vector $\mathbf{z}$ and scene layout of equal size. These disentanglement studies demonstrate the effectiveness of the joint layout-texture latent space in learning a smooth manifold of images conditioned on the layout.

## 6 Conclusion

In this work, we introduced Scene Style Network (SSN), a novel framework that leverages a joint layout-texture intermediate latent space to offer improved layout-texture control of conditional image synthesis. To achieve this, we proposed to decouple the *layout* pathway from the *texture* pathway, in order to subsequently obtain a joint layout-texture embedding to synthesize the output image. Importantly, the spatially-aware structure obtained by the layout pathway is explicitly disentangled from the fine-grained details obtained by the texture pathway until the joint layout-texture embedding. We showed empirically that SSN achieves comparable or better than state-of-the-art results for both conditional image generation and disentanglement tasks.

## References

1. Achille, A., Soatto, S.: Emergence of invariance and disentanglement in deep representations. Journal of Machine Learning Research **19**, 1–34 (09 2018)
2. Ashual, O., Wolf, L.: Specifying object attributes and relations in interactive scene generation. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
3. Barratt, S., Sharma, R.: A note on the inception score (01 2018)
4. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019)
5. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: CVPR (2018)
6. Chen, T.Q., Li, X., Grosse, R., Duvenaud, D.: Isolating sources of disentanglement in variational autoencoders (2018)
7. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2172–2180. Curran Associates, Inc. (2016)
8. Dosovitskiy, A., Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. Arxiv (11 2014)
9. Eastwood, C.: A framework for the quantitative evaluation of disentangled representations (03 2018)
10. Eslami, S.M.A., Rezende, D.J., Besse, F., Viola, F., Morcos, A.S., Garnelo, M., Ruderman, A., Rusu, A.A., Danihelka, I., Gregor, K., Reichert, D.P., Buesing, L., Weber, T., Vinyals, O., Rosenbaum, D., Rabinowitz, N.C., King, H., Hillier, C., Botvinick, M.M., Wierstra, D., Kavukcuoglu, K., Hassabis, D.: Neural scene representation and rendering. Science **360**, 1204–1210 (2018)
11. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style (2015)
12. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864. ACM (2016)
13. van den Hengel, A., Russell, C., Dick, A., Bastian, J., Pooley, D., Fleming, L., Agapito, L.: Part-based modelling of compound scenes from images. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
14. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 6626–6637. Curran Associates, Inc. (2017)
15. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M.M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
16. Hong, S., Yang, D., Choi, J., Lee, H.: Inferring semantic layout for hierarchical text-to-image synthesis (01 2018)
17. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV (2017)
18. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arxiv (2016)

19. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2017–2025. Curran Associates, Inc. (2015)
20. Johnson, J., Gupta, A., Fei-Fei, L.: Image generation from scene graphs. In: CVPR (2018)
21. Karacan, L., Akata, Z., Erdem, A., Erdem, E.: Learning to generate images of outdoor scenes from attributes and semantic layouts. ArXiv **abs/1612.00215** (2016)
22. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)
23. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. CoRR **abs/1812.04948** (2018)
24. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. CoRR **abs/1912.04958** (2019)
25. Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1857–1865. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017)
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
27. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalanditis, Y., Li, L.J., Shamma, D.A., Bernstein, M., Fei-Fei, L.: Visual genome: Connecting language and vision using crowdsourced dense image annotations (2016)
28. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2539–2547. Curran Associates, Inc. (2015)
29. Locatello, F., Bauer, S., Lučić, M., Rätsch, G., Gelly, S., Schölkopf, B., Bachem, O.F.: Challenging common assumptions in the unsupervised learning of disentangled representations. In: International Conference on Machine Learning (2019)
30. Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., Smolley, S.: Least squares generative adversarial networks. pp. 2813–2821 (10 2017). https://doi.org/10.1109/ICCV.2017.304
31. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Henry Holt and Co., Inc., New York, NY, USA (1982)
32. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013)
33. Miyato, T., Koyama, M.: cGANs with projection discriminator. In: International Conference on Learning Representations (2018)
34. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 2642–2651. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017)
35. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. CoRR **abs/1403.6652** (2014)

36. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1060–1069. PMLR, New York, New York, USA (20–22 Jun 2016)
37. Reed, S.E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., Lee, H.: Learning what and where to draw. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 217–225. Curran Associates, Inc. (2016)
38. Riesenhuber, M., Poggio, T.: Models of object recognition. Nature neuroscience **3 Suppl**, 1199–204 (12 2000). https://doi.org/10.1038/81479
39. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training gans. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2234–2242. Curran Associates, Inc. (2016)
40. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**, 61–80 (2009)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv 1409.1556 (09 2014)
42. Sun, W., Wu, T.: Image synthesis from reconfigurable layout and style. ArXiv **abs/1908.07500** (2019)
43. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015)
44. Tan, F., Feng, S., Ordonez, V.: Text2scene: Generating compositional scenes from textual descriptions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
45. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW '15 (2015)
46. Yao, S., Hsu, T.M.H., Zhu, J.Y., Wu, J., Torralba, A., Freeman, W.T., Tenenbaum, J.B.: 3d-aware scene manipulation via inverse graphics. In: Advances in neural information processing systems (2018)
47. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7354–7363. PMLR, Long Beach, California, USA (09–15 Jun 2019)
48. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV (2017)
49. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016)
50. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
51. Zhao, B., Meng, L., Yin, W., Sigal, L.: Image generation from layout. In: CVPR (2019)
52. Zhu, J.Y., Park, T., Isola, P., Efros, A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. pp. 2242–2251 (10 2017). https://doi.org/10.1109/ICCV.2017.244

## Supplementary Material

## 7 Additional Qualitative Results

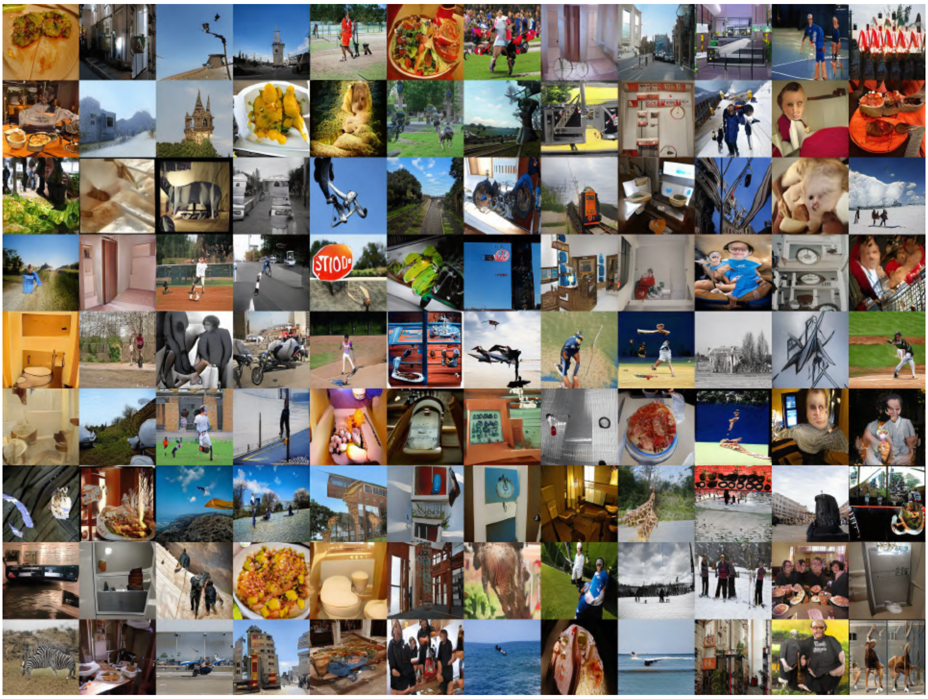
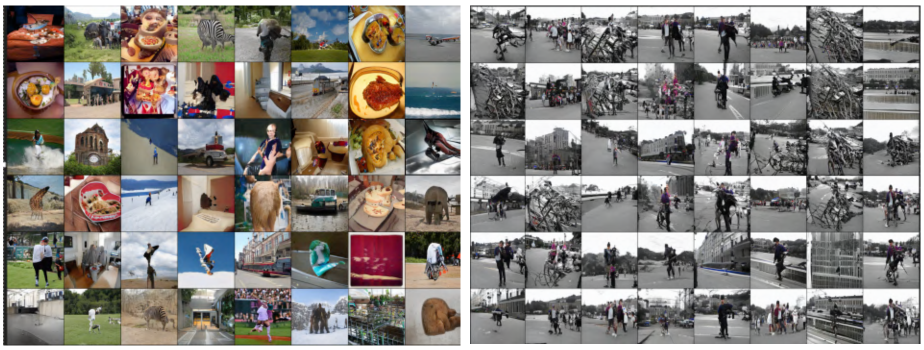
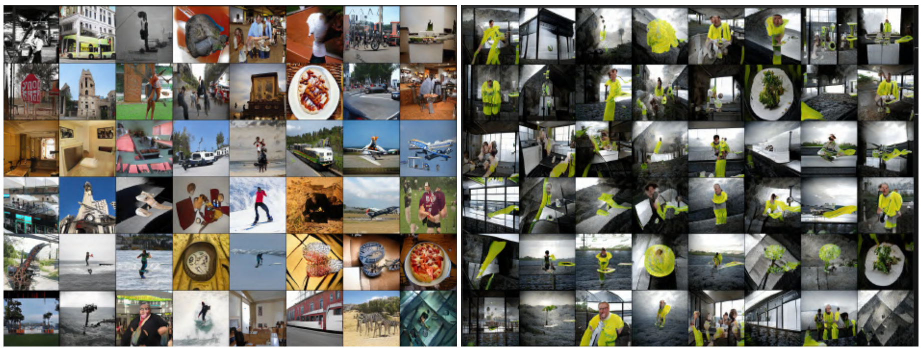


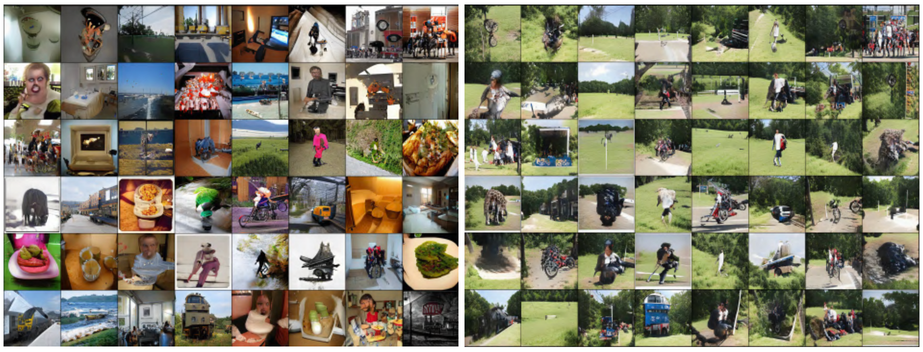
Fig. 7: Fake images of Visual Genome with StyleGAN. StyleGAN is trained on images from VG at resolution 256 × 256 for more than 7.8M images and 6 days on 4 TITAN RTX.



(a)



(b)



(c)

Fig. 8: Visualizing the diversity of fixed textures. For each of the pairs 10a, 10b, 8c, the left shows 48 unaltered generation images, and the right shows the corresponding images with fixed texture. Note that the image at $(i, j)$ in the left preserves the spatial layout at $(i, j)$ in the right.

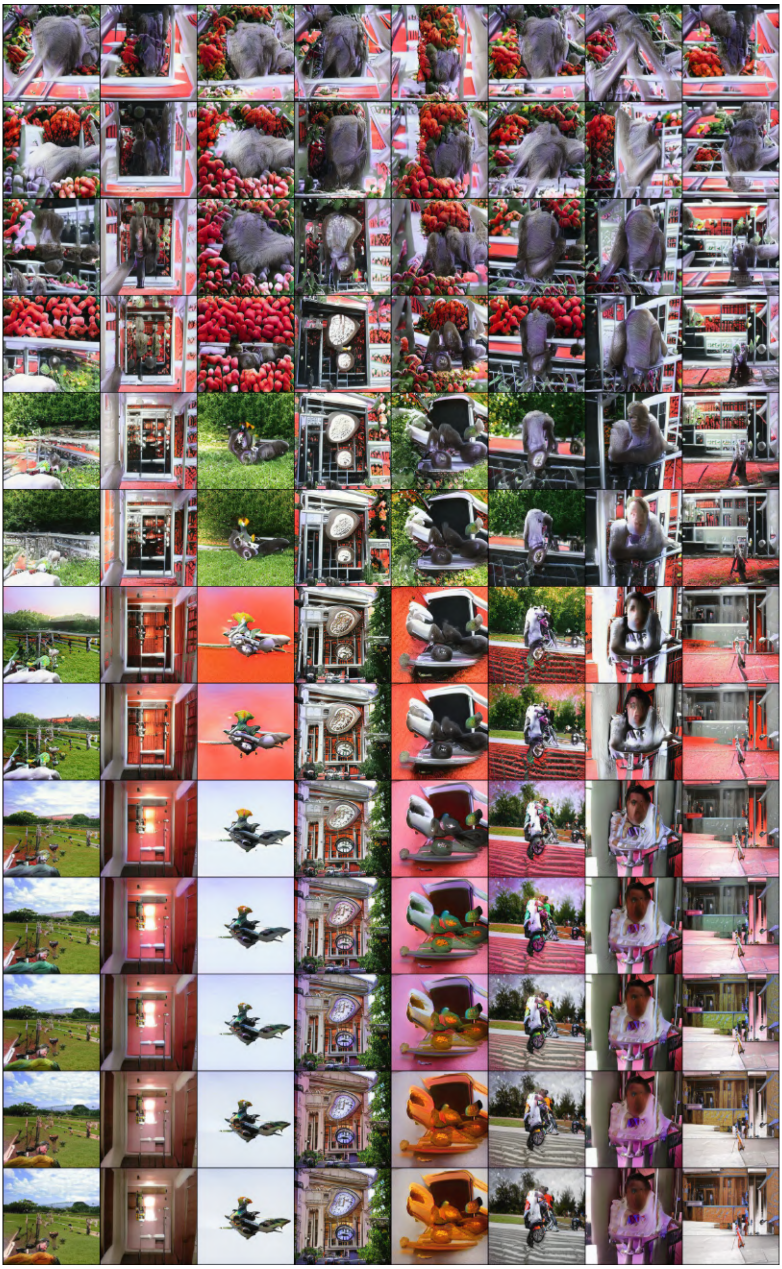
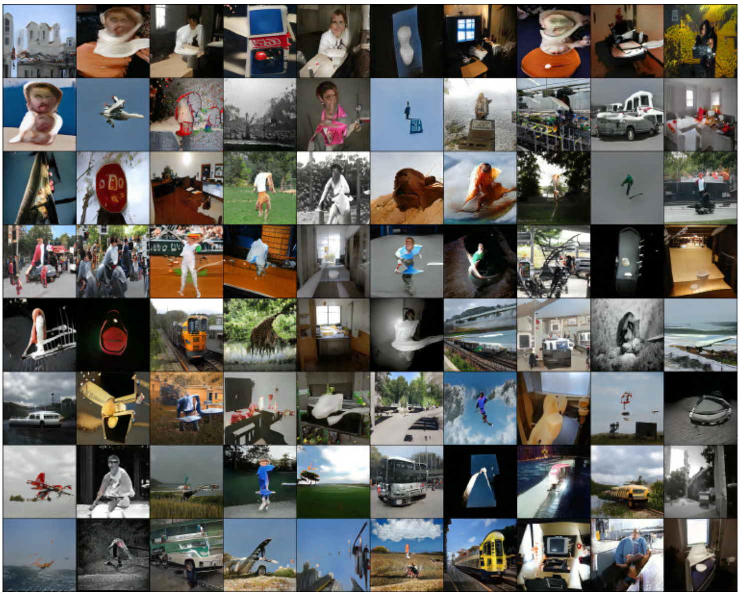
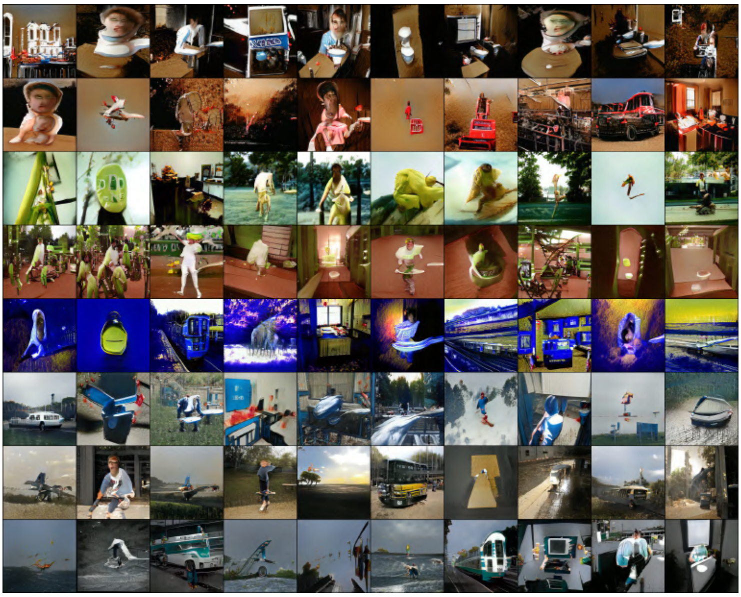


Fig. 9: Additional style mixing results similar to Figure 6.



(a)



(b)

Fig. 10: Additional results for fixed texture experiments similar to 2. (Top): unaltered generated images. (Bottom): generated images with fixed texture along every row.

## 8 Architecture Details

### Scene layout extractor

Table 3: Architecture of the scene layout extractor. The semantics of the Conv2d configuration is (in_channels, out_channels, kernel_size, stride, padding); for MaxPool2d it is (kernel_size, padding); for Linear it is (in_features, out_features).

| **Input** | **Config** | **Shape** |
|---|---|---|
| SceneLayout | — | $512 \times 32 \times 32$ |
| **Operation** | **Config** | **Shape** |
| Conv2d | 512, 256, 3, 1, 1 | $256 \times 32 \times 32$ |
| ReLU | — | $256 \times 32 \times 32$ |
| MaxPool2d | 2, 2 | $256 \times 16 \times 16$ |
| Conv2d | 256, 128, 3, 1, 1 | $128 \times 16 \times 16$ |
| ReLU | — | $128 \times 16 \times 16$ |
| MaxPool2d | 2, 2 | $128 \times 8 \times 8$ |
| MaxPool2d | 2, 2 | $128 \times 4 \times 4$ |
| Linear | $128 \times 4 \times 4$, 512 | 512 |
| **Output** | **Config** | **Shape** |
| Concatenate | $\mathbf{z} \in \mathcal{Z}$, SceneLayout | 1024 |

## Synthesis network

Table 4: Architecture of the synthesis network. The input $\mathbf{z}' = [\mathbf{z}; \hat{\mathbf{z}}]$ is the concatenation of a sampled texture code $z$ and the SceneLayout (see Table 3). The PixelNorm layer normalizes the feature vector in each pixel to unit length with no trainable weights. The ConstantInput layer introduces a trainable layer of constant input. LeakyReLU uses a negative slope of 0.2. All linear layers contain a bias parameter. See Table 5 for details of the SynthesisBlock.

| **Input** | **Config** | **Shape** |
|---|---|---|
| $\mathbf{z}' = [\mathbf{z}; \hat{\mathbf{z}}]$ | — | 1024 |
| **Operation** | **Config** | **Shape** |
| PixelNorm | — | 1024 |
| Linear1 | 1024, 512 | 512 |
| LeakyReLU | — | 512 |
| Linear2 | 512, 512 | 512 |
| LeakyReLU | — | 512 |
| ⋮ | ⋮ | ⋮ |
| Linear8 | 512, 512 | 512 |
| LeakyReLU | — | 512 |
| ConstantInput | 512, 4 | $512 \times 4 \times 4$ |
| SynthesisBlock | — | $3 \times 8 \times 8$ |
| ⋮ | ⋮ | ⋮ |
| SynthesisBlock | — | $3 \times 256 \times 256$ |

## Synthesis block

Table 5: Architecture of a synthesis block at resolution $n \times n$, where $n = 2^i$ for $i = [3, ..., 8]$. The notation for a ModulatedConv2d operation is defined as (in_channel, out_channel, kernel_size, upsample, downsample). NoiseInjection returns the previous feature activations shifted by a scaled trainable weight. Upsampling is achieved by nearest neighbor sampling.

| **Input** | **Config** | **Shape** |
|---|---|---|
| Input | — | $512 \times 2^i \times 2^i$ |
| Style | — | 512 |
| **Operation** | **Config** | **Shape** |
| ModulatedConv2d | $2^i$, $2^i$, 3, False, False | $512 \times 2^{i+1} \times 2^{i+1}$ |
| NoiseInjection | — | $512 \times 2^{i+1} \times 2^{i+1}$ |
| LeakyReLU | — | $512 \times 2^{i+1} \times 2^{i+1}$ |
| ModulatedConv2d | $2^i$, $2^i$, 3, True, False | $512 \times 2^{i+1} \times 2^{i+1}$ |
| NoiseInjection | — | $512 \times 2^{i+1} \times 2^{i+1}$ |
| LeakyReLU | — | $512 \times 2^{i+1} \times 2^{i+1}$ |
| Upsample | — | $3 \times 2^{i+1} \times 2^{i+1}$ |