StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks

Han Zhang*[1], Tao Xu*[2], Hongsheng Li[3],
Shaoting Zhang[4], Xiaogang Wang[3], Xiaolei Huang[2], Dimitris Metaxas[1]

[1]Rutgers University [2]Lehigh University [3]The Chinese University of Hong Kong [4]Baidu Research
{han.zhang, dnm}@cs.rutgers.edu, {tax313, xih206}@lehigh.edu
{hsli, xgwang}@ee.cuhk.edu.hk, zhangshaoting@baidu.com

Abstract

Although Generative Adversarial Networks (GANs) have shown remarkable success in various tasks, they still face challenges in generating high quality images. In this paper, we propose Stacked Generative Adversarial Networks (StackGAN) aimed at generating high-resolution photo-realistic images. First, we propose a two-stage generative adversarial network architecture, StackGAN-v1, for text-to-image synthesis. The Stage-I GAN sketches primitive shape and colors of the object based on given text description, yielding low-resolution images. The Stage-II GAN takes Stage-I results and text descriptions as inputs, and generates high-resolution images with photo-realistic details. Second, an advanced multi-stage generative adversarial network architecture, StackGAN-v2, is proposed for both conditional and unconditional generative tasks. Our StackGAN-v2 consists of multiple generators and discriminators in a tree-like structure; images at multiple scales corresponding to the same scene are generated from different branches of the tree. StackGAN-v2 shows more stable training behaviour than StackGAN-v1 by jointly approximating multiple distributions. Extensive experiments demonstrate that the proposed stacked generative adversarial networks significantly outperform other state-of-the-art methods in generating photo-realistic images.

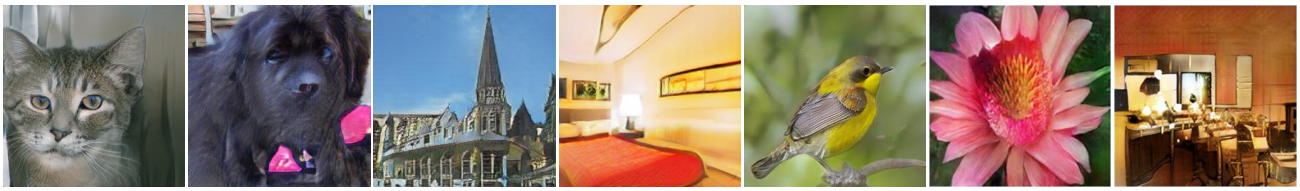
1. Introduction

Generative Adversarial Networks (GAN) is a generative model proposed by Goodfellow *et al*. [11]. In the original setting, GAN is composed of a generator and a discriminator that are trained with competing goals. The generator is trained to generate samples towards the true data distribution to fool the discriminator, while the discriminator is optimized to distinguish between real samples from the true data distribution and fake samples produced by the generator. Recently, GAN has shown great potential in simulating complex data distributions, such as those of texts [5], images [28] and videos [44].

Despite the success, GAN models are known to be difficult to train. The training process is usually unstable and sensitive to the choices of hyper-parameters. Several papers argued that the instability is partially due to the disjoint supports of the data distribution and the implied model distribution [38, 1]. This problem is more severe when training GAN to generate high-resolution (*e.g*., 256×256) images because the chance is very rare for the high-resolution image distribution and the model distribution to share supports in a high-dimensional space. Moreover, a common failure phenomenon for GAN training is *mode collapse*: many of the generated samples contain the same color or texture pattern.

In order to stabilize the GAN training process and improve sample diversity, several methods tried to address the challenges by proposing new network architectures [28], introducing heuristic tricks [36] or modifying the learning objectives [2, 4]. But most of the previous methods are designed to approximate a single data distribution (*e.g*., images of the same size). Due to the difficulty in directly approximating the high-resolution image data distribution because of rare overlap between model and data distributions in the very high-dimensional space, most previous methods are limited to generating low-resolution images. In this work, we observe that, real world data, especially natural images, can be modeled at different scales [34]. One can view multi-resolution digitized images as sampling from the same continuous image signal with different sampling rates. As the resolution goes lower, aliasing is more conspicuous thus many different high-resolution images may get subsampled to correspond to the same low resolution

*Indicates equal contribution



ImageNet cat [35] ImageNet dog [35] LSUN church [49] LSUN bedroom [49] CUB [45] Oxford-102 [26] MS COCO [20]

Figure 1. 256×256 photo-realistic images generated by the proposed StackGAN-v2 (leftmost five columns) and StackGAN-v1 (rightmost two columns) on different datasets.

image. Henceforth, the distributions of images at multiple discrete scales are related. Apart from multiple distributions of different scales, images coupled with or without auxiliary conditioning variables (*e.g.*, class labels or text descriptions) can be viewed as conditional distributions or unconditional distributions, which are also related distributions. Motivated by these observations, we argue that GAN can be stably trained to generate high resolution images by breaking the difficult generative task into sub-problems with progressive goals, *i.e.*, we propose Stacked Generative Adversarial Networks (StackGAN) to model a series of low-to-high-dimensional data distributions.

First, we propose a two-stage generative adversarial networks, StackGAN-v1, to generate images from text descriptions through a sketch-refinement process [50]. Low-resolution images are first generated by our Stage-I GAN. On the top of Stage-I GAN, we stack Stage-II GAN to generate high-resolution (*e.g.*, 256×256) images. By conditioning on the Stage-I result and the text again, Stage-II GAN learns to capture the text information that is omitted by Stage-I GAN and draws more details. For the text-to-image generation task, the limited number of training text-image pairs often results in sparsity in the text conditioning manifold and such sparsity makes it difficult to train GAN. Thus, we propose a novel Conditioning Augmentation technique to encourage smoothness in the latent conditioning manifold [50]. It allows small random perturbations in the conditioning manifold and increases the diversity of synthesized images.

Second, we propose an advanced multi-stage generative adversarial network architecture, StackGAN-v2, for both conditional and unconditional generative tasks. StackGAN-v2 has multiple generators that share most of their parameters in a tree-like structure. As shown in Figure 3, the input of the framework can be viewed as the root of the tree, and multi-scale images are generated from different branches of the tree. The generator at the deepest branch has the final goal of generating photo-realistic high-resolution images. Generators at intermediate branches have progressive goals of generating small to large images to help accomplish the final goal. The whole network is jointly trained to approximate different but highly related image distributions at different branches. The positive feedback from modeling one distribution can improve the learning of others. For conditional image generation tasks, our proposed StackGAN-v2 simultaneously approximates the unconditional image-only distribution and the conditional image distribution. Those two types of distributions are complementary to each other. Moreover, we propose a color-consistency regularization term to guide our generators to generate more coherent samples across different scales. The regularization provides additional constraints to facilitate multi-distribution approximation, which is especially useful in the unconditional setting where there is no instance-wise supervision between the image and the input noise vector.

In summary, the proposed Stacked Generative Adversarial Networks, StackGAN-v1 and StackGAN-v2, decompose the difficult problem of generating high-resolution images into more manageable sub-problems and significantly improve the state of the arts for both conditional and unconditional image generative tasks. (1) Our StackGAN-v1 for the first time generates images of 256×256 resolution with photo-realistic details from text descriptions. (2) A new Conditioning Augmentation technique is proposed to stabilize the conditional GAN training and also improves the diversity of the generated samples. (3) Our StackGAN-v2 jointly approximates multiple distributions to generate realistic images and stabilize GANs' training behavior. (4) For conditional image generation, in addition to multi-scale image distributions, our StackGAN-v2 simultaneously approximates the image-only distribution and its complementary conditional image distribution (*e.g.*, conditioned on text description or class label), resulting in better performance. (5) A color-consistency regularization term is proposed to provide additional constraints to multi-distribution approximation. It significantly improves the generated images' quality for unconditioned generation tasks.

In the rest of this paper, we first discuss the related work and preliminaries in section 2 and section 3, respectively. And then, we introduce our StackGAN-v1 [50] in section 4 and StackGAN-v2 in section 5. In section 6, extensive experiments are conducted to evaluate our method. Finally, we make our conclusions in section 7. Our StackGAN-v1 code is available at https://github.com/hanzhanggit/StackGAN. Our StackGAN-v2 code will be released soon.

2. Related Work

Generative image modeling is a fundamental problem in computer vision. There has been remarkable progress in this direction with the emergence of deep learning techniques. Variational Autoencoders (VAE) [17, 33] formulated the problem with probabilistic graphical models whose goal was to maximize the lower bound of data likelihood. Autoregressive models (*e.g.*, PixelRNN) [40] that utilized neural networks to model the conditional distribution of the pixel space have also generated appealing synthetic images. Recently, Generative Adversarial Networks (GAN) [11] have shown promising performance for generating sharper images. But training instability makes it hard for GAN models to generate high-resolution (*e.g.*, 256×256) images. A lot of works have been proposed to stabilize the training and improve the image qualities [28, 36, 23, 51, 4, 25].

Built upon these generative models, conditional image generation has also been studied. Most methods utilized simple conditioning variables such as attributes or class labels [47, 41, 6, 27]. There is also work conditioned on images to generate images, including photo editing [3, 52], domain transfer [39, 15] and super-resolution [38, 19]. However, super-resolution methods [38, 19] can only add limited details to low-resolution images and can not correct large defects as our proposed StackGAN does. Recently, several methods have been developed to generate images from unstructured text. Mansimov *et al*. [21] built an AlignDRAW model by learning to estimate alignment between text and the generating canvas. Reed *et al*. [32] used conditional PixelCNN to generate images using the text descriptions and object location constraints. Nguyen *et al*. [25] used an approximate Langevin sampling approach to generate images conditioned on text. However, their sampling approach requires an inefficient iterative optimization process. With conditional GAN, Reed *et al*. [31] successfully generated plausible 64×64 images for birds and flowers based on text descriptions. Their follow-up work [29] was able to generate 128×128 images by utilizing additional annotations on object part locations.

Given the difficulties of modeling details of natural images, many works have been proposed to use multiple GANs to improve the sample quality. Denton *et al*. [7] built a series of GANs within a Laplacian pyramid framework. At each level of the pyramid, a residual image was generated conditioned on the image of the previous stage and then added back to the input image to produce the input for the next stage. Wang *et al*. [46] utilized a structure GAN and a style GAN to synthesize images of the indoor scenes. Yang *et al*. [48] factorized image generation into foreground and background generation with layered recursive GANs. Concurrent to our work, Huang *et al*. [13] added several GANs to reconstruct the multi-level representations of a pre-trained discriminative model. But they were still unable to generate high resolution image with photo-realistic details. Durugkar *et al*. [9] used multiple discriminators along with one generator to increase the chance of the generator receiving effective feedback. However, all discriminators in their framework are trained to approximate a single data distribution instead of multiple distributions.

3. Preliminaries

Generative Adversarial Networks (GANs) [11] are composed of two models that are alternatively trained to compete with each other. The generator G is optimized to reproduce the true data distribution p_{data} by generating images that are difficult for the discriminator D to differentiate from real images. Meanwhile, D is optimized to distinguish real images and synthetic images generated by G. Overall, the training procedure is similar to a two-player min-max game with the following objective function,

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))], \quad (1)$$

where x is a real image from the true data distribution p_{data}, and z is a noise vector sampled from distribution p_z (*e.g.*, uniform or Gaussian distribution).

Conditional GAN [10, 24] is an extension of GAN where both the generator and discriminator receive additional conditioning variables c, yielding $G(z, c)$ and $D(x, c)$. This formulation allows G to generate images conditioned on variables c.

4. StackGAN-v1: Two-stage Generative Adversarial Networks

To generate high-resolution images with photo-realistic details, we propose a simple yet effective two-stage generative adversarial networks, StackGAN-v1. As shown in Figure 2, it decomposes the text-to-image generative process into two stages.

- **Stage-I GAN:** it sketches the primitive shape and basic colors of the object conditioned on the given text description, and draws the background layout from a random noise vector, yielding a low-resolution image.

- **Stage-II GAN:** it corrects defects in the low-resolution image from Stage-I and completes details of the object by reading the text description again, producing a high-resolution photo-realistic image.

4.1. Conditioning Augmentation

As shown in Figure 2, the text description t is first encoded by an encoder, yielding a text embedding φ_t. In previous works [31, 29], the text embedding is nonlinearly transformed to generate conditioning latent variables as the

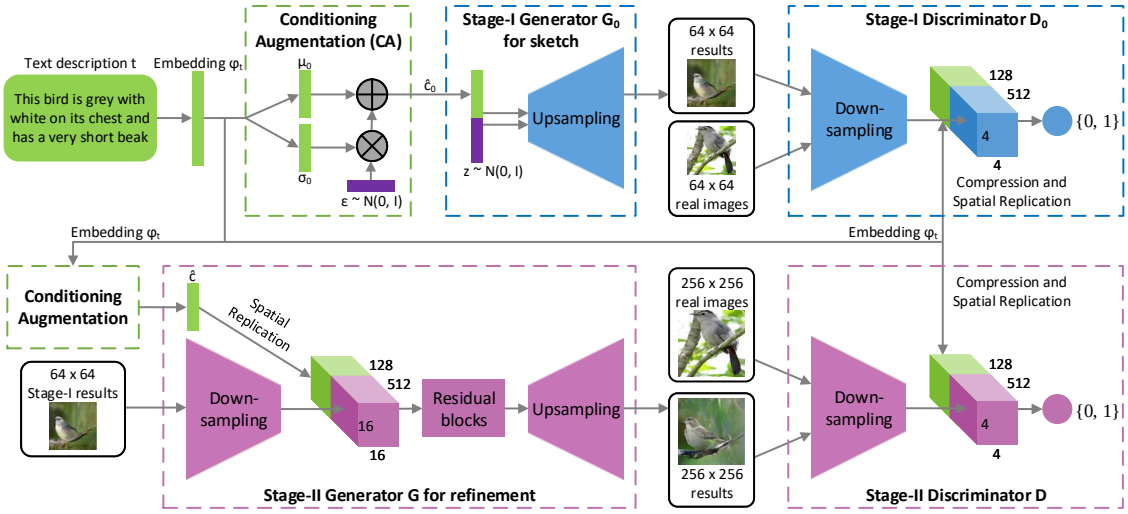


Figure 2. The architecture of the proposed StackGAN-v1. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

input of the generator. However, latent space for the text embedding is usually high dimensional (> 100 dimensions). With limited amount of data, it usually causes discontinuity in the latent data manifold, which is not desirable for learning the generator. To mitigate this problem, we introduce a Conditioning Augmentation technique to produce additional conditioning variables $\hat{c}$. In contrast to the fixed conditioning text variable c in [31, 29], we randomly sample the latent variables $\hat{c}$ from an independent Gaussian distribution $\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))$, where the mean $\mu(\varphi_t)$ and diagonal covariance matrix $\Sigma(\varphi_t)$ are functions of the text embedding φ_t. The proposed Conditioning Augmentation yields more training pairs given a small number of image-text pairs, and thus encourages robustness to small perturbations along the conditioning manifold. To further enforce the smoothness over the conditioning manifold and avoid overfitting [8, 18], we add the following regularization term to the objective of the generator during training,

$$D_{KL}\left(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) \,||\, \mathcal{N}(0, I)\right), \tag{2}$$

which is the Kullback-Leibler divergence (KL divergence) between the standard Gaussian distribution and the conditioning Gaussian distribution. The randomness introduced in the Conditioning Augmentation is beneficial for modeling text to image translation as the same sentence usually corresponds to objects with various poses and appearances.

4.2. Stage-I GAN

Instead of directly generating a high-resolution image conditioned on the text description, we simplify the task to first generate a low-resolution image with our Stage-I GAN, which focuses on drawing only rough shape and correct colors for the object.

Let φ_t be the text embedding of the given description, which is generated by a pre-trained encoder [30] in this paper. The Gaussian conditioning variables $\hat{c}_0$ for text embedding are sampled from $\mathcal{N}(\mu_0(\varphi_t), \Sigma_0(\varphi_t))$ to capture the meaning of φ_t with variations. Conditioned on $\hat{c}_0$ and random variable z, Stage-I GAN trains the discriminator D_0 and the generator G_0 by alternatively maximizing $\mathcal{L}_{D_0}$ in Eq. (3) and minimizing $\mathcal{L}_{G_0}$ in Eq. (4),

$$\begin{aligned}\mathcal{L}_{D_0} = &\mathbb{E}_{(I_0,t)\sim p_{data}}[\log D_0(I_0, \varphi_t)] + \\ &\mathbb{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))],\end{aligned} \tag{3}$$

$$\begin{aligned}\mathcal{L}_{G_0} = &\mathbb{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))] + \\ &\lambda D_{KL}(\mathcal{N}(\mu_0(\varphi_t), \Sigma_0(\varphi_t)) \,||\, \mathcal{N}(0, I)),\end{aligned} \tag{4}$$

where the real image I_0 and the text description t are from the true data distribution p_{data}. z is a noise vector randomly sampled from a given distribution p_z (Gaussian distribution in this paper). λ is a regularization parameter that balances the two terms in Eq. (4). We set $\lambda = 1$ for all our experiments. Using the reparameterization trick introduced in [17], both $\mu_0(\varphi_t)$ and $\Sigma_0(\varphi_t)$ are learned jointly with the rest of the network.

Model Architecture. For the generator G_0, to obtain text conditioning variable $\hat{c}_0$, the text embedding φ_t is first fed into a fully connected layer to generate μ_0 and σ_0 (σ_0 are the values in the diagonal of Σ_0) for the Gaussian distribution $\mathcal{N}(\mu_0(\varphi_t), \Sigma_0(\varphi_t))$. $\hat{c}_0$ are then sampled from the Gaussian distribution. Our N_g dimensional conditioning vector $\hat{c}_0$ is computed by $\hat{c}_0 = \mu_0 + \sigma_0 \odot \epsilon$ (where $\odot$ is the element-wise multiplication, $\epsilon \sim \mathcal{N}(0, I)$). Then, $\hat{c}_0$ is concatenated with a N_z dimensional noise vector to generate a $W_0 \times H_0$ image by a series of up-sampling blocks.

For the discriminator D_0, the text embedding φ_t is first compressed to N_d dimensions using a fully-connected layer and then spatially replicated to form a $M_d \times M_d \times N_d$ tensor. Meanwhile, the image is fed through a series of down-sampling blocks until it has $M_d \times M_d$ spatial dimension. Then, the image filter map is concatenated along the channel dimension with the text tensor. The resulting tensor is further fed to a 1×1 convolutional layer to jointly learn features across the image and the text. Finally, a fully-connected layer with one node is used to produce the decision score.

4.3. Stage-II GAN

Low-resolution images generated by Stage-I GAN usually lack vivid object parts and might contain shape distortions. Some details in the text might also be omitted in the first stage, which is vital for generating photo-realistic images. Our Stage-II GAN is built upon Stage-I GAN results to generate high-resolution images. It is conditioned on low-resolution images and also the text embedding again to correct defects in Stage-I results. The Stage-II GAN completes previously ignored text information to generate more photo-realistic details.

Conditioning on the low-resolution result $s_0 = G_0(z, \hat{c}_0)$ and Gaussian latent variables $\hat{c}$, the discriminator D and generator G in Stage-II GAN are trained by alternatively maximizing $\mathcal{L}_D$ in Eq. (5) and minimizing $\mathcal{L}_G$ in Eq. (6),

$$\begin{aligned}\mathcal{L}_D = &\mathbb{E}_{(I,t)\sim p_{data}}[\log D(I, \varphi_t)] + \\ &\mathbb{E}_{s_0\sim p_{G_0}, t\sim p_{data}}[\log(1 - D(G(s_0, \hat{c}), \varphi_t))],\end{aligned} \tag{5}$$

$$\begin{aligned}\mathcal{L}_G = &\mathbb{E}_{s_0\sim p_{G_0}, t\sim p_{data}}\left[\log(1 - D(G(s_0, \hat{c}), \varphi_t))\right] + \\ &\lambda D_{KL}\left(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) \,||\, \mathcal{N}(0, I)\right),\end{aligned} \tag{6}$$

Different from the original GAN formulation, the random noise z is not used in this stage with the assumption that the randomness has already been preserved by s_0. Gaussian conditioning variables $\hat{c}$ used in this stage and $\hat{c}_0$ used in Stage-I GAN share the same pre-trained text encoder, generating the same text embedding φ_t. However, Stage-I and Stage-II Conditioning Augmentation have different fully connected layers for generating different means and standard deviations. In this way, Stage-II GAN learns to capture useful information in the text embedding that is omitted by Stage-I GAN.

Model Architecture. We design Stage-II generator as an encoder-decoder network with residual blocks [12]. Similar to the previous stage, the text embedding φ_t is used to generate the N_g dimensional text conditioning vector $\hat{c}$, which is spatially replicated to form a $M_g \times M_g \times N_g$ tensor. Meanwhile, the Stage-I result s_0 generated by Stage-I GAN is fed into several down-sampling blocks (*i.e.*, encoder) until it has a spatial size of $M_g \times M_g$. The image features and the text features are concatenated along the channel dimension. The encoded image features coupled with text features are fed into several residual blocks, which are designed to learn multi-modal representations across image and text features. Finally, a series of up-sampling layers (*i.e.*, decoder) are used to generate a $W \times H$ high-resolution image. Such a generator is able to help rectify defects in the input image while add more details to generate the realistic high-resolution image.

For the discriminator, its structure is similar to that of Stage-I discriminator with only extra down-sampling blocks since the image size is larger in this stage. To explicitly enforce GAN to learn better alignment between the image and the conditioning text, rather than using the vanilla discriminator, we adopt the matching-aware discriminator proposed by Reed *et al.* [31] for both stages. During training, the discriminator takes real images and their corresponding text descriptions as positive sample pairs, whereas negative sample pairs consist of two groups. The first is real images with mismatched text embeddings, while the second is synthetic images with their corresponding text embeddings.

4.4. Implementation details

The up-sampling blocks consist of the nearest-neighbor upsampling followed by a 3×3 stride 1 convolution. Batch normalization [14] and ReLU activation are applied after every convolution except the last one. The residual blocks consist of 3×3 stride 1 convolutions, Batch normalization and ReLU. Two residual blocks are used in 128×128 StackGAN-v1 models while four are used in 256×256 models. The down-sampling blocks consist of 4×4 stride 2 convolutions, Batch normalization and LeakyReLU, except that the first one does not have Batch normalization.

By default, $N_g = 128$, $N_z = 100$, $M_g = 16$, $M_d = 4$, $N_d = 128$, $W_0 = H_0 = 64$ and $W = H = 256$. For training, we first iteratively train D_0 and G_0 of Stage-I GAN for 600 epochs by fixing Stage-II GAN. Then we iteratively train D and G of Stage-II GAN for another 600 epochs by fixing Stage-I GAN. All networks are trained using ADAM [16] solver with batch size 64 and an initial learning rate of 0.0002. The learning rate is decayed to $1/2$ of its previous value every 100 epochs.

5. StackGAN-v2: Multi-distribution Generative Adversarial Networks

As discussed above, our StackGAN-v1 has two separate networks, Stage-I GAN and Stage-II GAN, to model low to high resolution image distributions. An alternative method for gradually modeling low-to-high resolution image distributions is to build an end-to-end network as our StackGAN-v2. As shown in Figure 3, our StackGAN-v2 consists of multiple generators (Gs) and discriminators (Ds) in a tree-like structure. Images from low-resolution

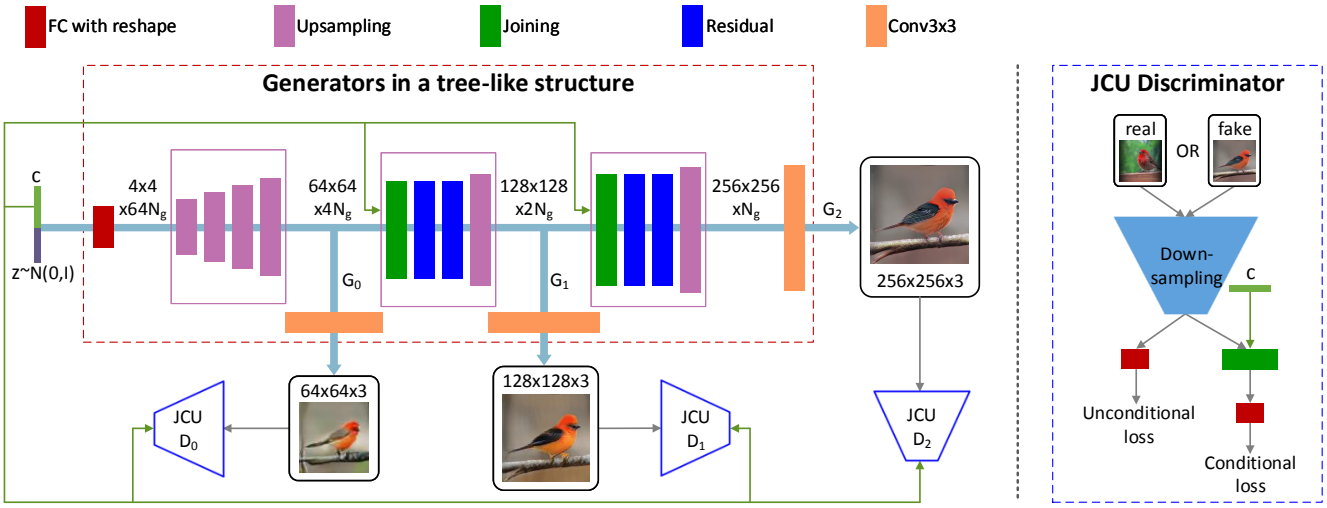


Figure 3. The overall framework of our proposed StackGAN-v2 for the conditional image synthesis task. (c is the vector of conditioning variables which can be computed from the class label, the text description, *etc.*; N_g and N_d are the numbers of channels of a tensor.)

to high-resolution are generated from different branches of the tree. At each branch, the generator captures the image distribution at that scale and the discriminator estimates the probability that a sample came from training images of that scale rather than the generator. The generators are jointly trained to approximate the multiple distributions, and the generators and discriminators are trained alternatively. In this section, we explore two types of multi-distributions: (1) multi-scale image distributions; and (2) joint conditional and unconditional image distributions.

5.1. Multi-scale image distributions approximation

Our StackGAN-v2 framework has a tree-like structure, which takes a noise vector $z \sim p_{noise}$ as the input and has multiple generators to produce images of different scales. The p_{noise} is a prior distribution, which is usually chosen as the standard normal distribution. The latent variables z are transformed to hidden features layer by layer. We compute the hidden features h_i for each generator G_i by a non-linear transformation,

$$h_0 = F_0(z); \quad h_i = F_i(h_{i-1}, z),\ i = 1, 3, ..., m-1, \quad (7)$$

where h_i represents hidden features for the i^{th} branch, m is the total number of branches, and F_i are modeled as neural networks with shared parameters (see Figure 3 for illustration). In order to capture information omitted in preceding branches, the noise vector z is concatenated to the hidden features h_{i-1} as the inputs of F_i for calculating h_i. Based on hidden features at different layers (h_0, h_1, ..., h_{m-1}), generators produce samples of small-to-large scales (s_0, s_1, ..., s_{m-1}),

$$s_i = G_i(h_i),\ i = 0, 1, ..., m-1, \quad (8)$$

where G_i is the generator for the i^{th} branch.

Following each generator G_i, a discriminator D_i, which takes a real image x_i or a fake sample s_i as input, is trained to classify inputs into two classes (real or fake) by minimizing the following the cross-entropy loss,

$$\mathcal{L}_{D_i} = -\frac{1}{2}\mathbb{E}_{x_i \sim p_{data_i}}[\log D_i(x_i)] - \frac{1}{2}\mathbb{E}_{s_i \sim p_{G_i}}[\log(1 - D_i(s_i)], \quad (9)$$

where x_i is from the true image distribution p_{data_i} at the i^{th} scale, and s_i is from the model distribution p_{G_i} at the same scale. The multiple discriminators are trained in parallel, and each of them focuses on a single image scale.

Guided by the trained discriminators, the generators are optimized to jointly approximate multi-scale image distributions ($p_{data_0}, p_{data_1}, ..., p_{data_{m-1}}$) by minimizing the following loss function,

$$\mathcal{L}_G = \sum_{i=1}^{m} \mathcal{L}_{G_i}, \quad \mathcal{L}_{G_i} = \frac{1}{2}\mathbb{E}_{s_i \sim p_{G_i}}[\log(1 - D_i(s_i)], \quad (10)$$

where $\mathcal{L}_{G_i}$ is the loss function for approximating the image distribution at the i^{th} scale (*i.e.*, p_{data_i}). In practice, the generator G_i is modified to maximize $\log(D_i(s_i))$ instead of minimizing $\log(1 - D_i(s_i))$ to mitigate the problem of gradient vanishing [11]. During the training process, the discriminators D_i and the generators G_i are alternately optimized till convergence.

The motivation of our proposed StackGAN-v2 is that jointly approximating multiple image distributions increases the chance of these data distributions to share supports with model distributions. Adding auxiliary generation tasks to intermediate branches provides more gradient signals for training the whole network. For instance, approximating the low-resolution image distribution at the first branch results in images with basic color and structures. Then the generators at the subsequent branches can focus on completing details for generating higher resolution images.

5.2. Joint conditional and unconditional distribution approximation

For unconditional image generation, discriminators in StackGAN-v2 are trained to distinguish real or fake images. To handle conditional image generation, conventionally, images and their corresponding conditioning variables are input into the discriminator to determine whether an image-condition pair matches or not, which guides the generator to approximate the conditional image distribution. We propose conditional StackGAN-v2 that jointly approximates conditional and unconditional image distributions.

For the generator of our conditional StackGAN-v2, F_0 and F_i are converted to take the conditioning variable c as input, such that $h_0 = F_0(c, z)$ and $h_i = F_i(h_{i-1}, c)$. For F_i, the conditioning variable c replaces the noise vector z to encourage the generators to draw images with more details according to the conditioning variable. Consequently, multi-scale samples are now generated by $s_i = G_i(h_i)$. The objective function of training the discriminator D_i for conditional StackGAN-v2 now consists of two terms, the unconditional loss and the conditional loss,

$$\mathcal{L}_{D_i} = \underbrace{-\frac{1}{2}\mathbb{E}_{x_i\sim p_{data_i}}\left[\log D_i(x_i)\right] - \frac{1}{2}\mathbb{E}_{s_i\sim p_{G_i}}\left[\log(1-D_i(s_i)\right]}_{\text{unconditional loss}} + \underbrace{-\frac{1}{2}\mathbb{E}_{x_i\sim p_{data_i}}\left[\log D_i(x_i, c)\right] - \frac{1}{2}\mathbb{E}_{s_i\sim p_{G_i}}\left[\log(1-D_i(s_i, c)\right]}_{\text{conditional loss}}. \tag{11}$$

The unconditional loss determines whether the image is real or fake while the conditional one determines whether the image and the condition match or not. Accordingly, the loss function for each generator G_i is converted to

$$\mathcal{L}_{G_i} = \underbrace{\frac{1}{2}\mathbb{E}_{s_i\sim p_{G_i}}\left[\log(1-D_i(s_i)\right]}_{\text{unconditional loss}} + \underbrace{\frac{1}{2}\mathbb{E}_{s_i\sim p_{G_i}}\left[\log(1-D_i(s_i, c)\right]}_{\text{conditional loss}}. \tag{12}$$

The generator G_i at each scale therefore jointly approximates unconditional and conditional image distributions. The final loss for jointly training generators of conditional StackGAN-v2 is computed by substituting Eq. (12) into Eq. (10).

5.3. Color-consistency regularization

As we increase the image resolution at different generators, the generated images at different scales should share similar basic structure and colors. A color-consistency regularization term is introduced to keep samples generated from the same input at different generators more consistent in color and thus to improve the quality of the generated images.

Let $\boldsymbol{x_k} = (R, G, B)^T$ represent a pixel in a generated image, then the mean and covariance of pixels of the given image can be defined by $\boldsymbol{\mu} = \sum_k \boldsymbol{x_k}/N$ and $\boldsymbol{\Sigma} = \sum_k (\boldsymbol{x_k} - \boldsymbol{\mu})(\boldsymbol{x_k} - \boldsymbol{\mu})^T/N$, where N is the number of pixels in the image. The color-consistency regularization term aims at minimizing the differences of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ between different scales to encourage the consistency, which is defined as

$$\mathcal{L}_{C_i} = \frac{1}{n}\sum_{j=1}^{n}\left(\lambda_1\|\boldsymbol{\mu}_{s_i^j} - \boldsymbol{\mu}_{s_{i-1}^j}\|_2^2 + \lambda_2\|\boldsymbol{\Sigma}_{s_i^j} - \boldsymbol{\Sigma}_{s_{i-1}^j}\|_F^2\right), \tag{13}$$

where n is the batch size, $\boldsymbol{\mu}_{s_i^j}$ and $\boldsymbol{\Sigma}_{s_i^j}$ are mean and covariance for the j^{th} sample generated by the i^{th} generator. Empirically, we set $\lambda_1 = 1$ and $\lambda_2 = 5$ by default. For the j^{th} input vector, multi-scale samples s_1^j, s_2^j, ..., s_m^j are generated from m generators of StackGAN-v2. $\mathcal{L}_{C_i}$ can be added to the loss function of the i^{th} generator defined in Eq. (10) or Eq. (12), where $i = 2, 3, ..., m$. Therefore, the final loss for training the i^{th} generator is defined as $\mathcal{L}'_{G_i} = \mathcal{L}_{G_i} + \mathcal{L}_{C_i}$.

5.4. Implementation details

As shown in Figure 3, our StackGAN-v2 models are designed to generate 256×256 images. The input vector (*i.e.*, z for unconditional StackGAN-v2 and the concatenated z and c for conditional StackGAN-v2) is first transformed to a $4\times4\times64N_g$ feature tensor, where N_g is the number of channels in the tensor. Then, this $4\times4\times64N_g$ tensor is gradually transformed to $64\times64\times4N_g$, $128\times128\times2N_g$, and eventually $256\times256\times1N_g$ tensors at different layers of the network by six up-sampling blocks. The intermediate $64\times64\times4N_g$, $128\times128\times2N_g$, and $256\times256\times1N_g$ features are used to generate images of corresponding scales with 3×3 convolutions. Conditioning variables c or unconditional variables z are also directly fed into intermediate layers of the network to ensure encoded information in c or z is not omitted. All the discriminators D_i have down-sampling blocks and 3×3 convolutions to transform the input image to a $4\times4\times8N_d$ tensor, and eventually the sigmoid function is used for outputting probabilities. For all datasets, we set $N_g = 32$, $N_d = 64$ and use two residual blocks between every two generators. ADAM [16] solver with a learning rate of 0.0002 is used for all models.

6. Experiments

We conduct extensive experiments to evaluate our method. In section 6.1, several baseline models are designed to investigate the overall design and important components of our StackGAN-v1. For the first baseline, we directly train Stage-I GAN for generating 64×64 and 256×256 images to investigate whether the proposed two-stage stacked structure and the Conditioning Augmentation are beneficial. Then we modify our StackGAN-v1 to generate 128×128 and 256×256 images to investigate whether larger images by our method result in higher image quality. We also investigate whether inputting text at both stages of StackGAN-v1 is useful. In section 6.2, experi-

Dataset	CUB [45]		Oxford-102 [26]		COCO [20]		LSUN [49]		ImageNet [35]	
	train	test	train	test	train	test	bedroom	church	dog	cat
#Samples	8,855	2,933	7,034	1,155	80,000	40,000	3,033,042	126,227	147,873	6,500

Table 1. Statistics of datasets. We do not split LSUN or ImageNet into train/test sets because they are utilized for the unconditional tasks.

Figure 4. Samples generated by our StackGAN-v1 from unseen texts in CUB test set. Each column lists the text description, images generated from the text by Stage-I and Stage-II of StackGAN-v1.

ments are designed to validate important components of our StackGAN-v2, including the designs with fewer multi-scale image distributions, the effect of jointly approximating conditional and unconditional distributions, and the effectiveness of the proposed color-consistency regularization. Finally, we compare our StackGAN-v2 with StackGAN-v1 in section 6.3. In addition, several state-of-the-art methods on text-to-image synthesis [31, 29] and on unconditional image synthesis [28, 51, 2, 22] are also compared with our proposed method.

Datasets. We evaluate our conditional StackGAN for text-to-image synthesis on the CUB [45], Oxford-102 [26] and MS COCO [20] datasets. CUB [45] contains 200 bird species with 11,788 images. Since 80% of birds in this dataset have object-image size ratios of less than 0.5 [45], as a pre-processing step, we crop all images to ensure that bounding boxes of birds have greater-than-0.75 object-image size ratios. Oxford-102 [26] contains 8,189 images of flowers from 102 different categories. To show the generalization capability of our approach, a more challenging dataset, MS COCO [20] is also utilized for evaluation. Different from CUB and Oxford-102, the MS COCO dataset contains images with multiple objects and various backgrounds. Each image in COCO has 5 descriptions, while 10 descriptions are provided by [30] for every image in CUB and Oxford-102 datasets. Following the experimental setup in [31], we directly use the training and validation sets provided by COCO, meanwhile we split CUB and Oxford-102 into class-disjoint training and test sets. Our unconditional StackGAN-v2 utilizes bedroom and church sub-datasets of LSUN [49], and a dog-breed (using the wordNet IDs provided by Vinyals *et al.*, [43]) and a cat-breed (using the wordNet IDs provided in our supplementary materials) sub-datasets of ImageNet [35] to synthesize different types of images. The Statistics of these datasets are presented in Table 1.

Evaluation metrics. It is difficult to evaluate the performance of generative models (*e.g.*, GAN). We choose a recently proposed numerical assessment approach "inception score" [36] for quantitative evaluation,

$$I = \exp\left(\mathbb{E}_{\boldsymbol{x}} D_{KL}\left(p\left(y|\boldsymbol{x}\right) \,||\, p\left(y\right)\right)\right), \tag{14}$$

where $\boldsymbol{x}$ denotes one generated sample, and y is the label predicted by the Inception model [37]. The intuition behind this metric is that good models should generate diverse but meaningful images. Therefore, the KL divergence between the marginal distribution $p(y)$ and the conditional distribution $p(y|\boldsymbol{x})$ should be large. In our experiments, we directly use the pre-trained Inception model for COCO dataset. For fine-grained datasets, CUB and Oxford-102, we fine-tune an Inception model for each of them. As suggested in [36], we evaluate this metric on a large number of samples (*i.e.*, 30k randomly selected samples) for each model.

Although the inception score has shown to well correlate with human perception on visual quality of samples [36], it cannot reflect whether the generated images are well conditioned on the given text descriptions. Therefore, we also conduct human evaluation. We randomly select 50 text de-

Method	CA	Text twice	Inception score
64×64 Stage-I GAN	no	/	2.66 ± .03
	yes	/	2.95 ± .02
256×256 Stage-I GAN	no	/	2.48 ± .00
	yes	/	3.02 ± .01
128×128 StackGAN-v1	yes	no	3.13 ± .03
	no	yes	3.20 ± .03
	yes	yes	3.35 ± .02
256×256 StackGAN-v1	yes	no	3.45 ± .02
	no	yes	3.31 ± .03
	yes	yes	3.70 ± .04

Table 2. Inception scores calculated with 30,000 samples generated by different baseline models of our StackGAN-v1.

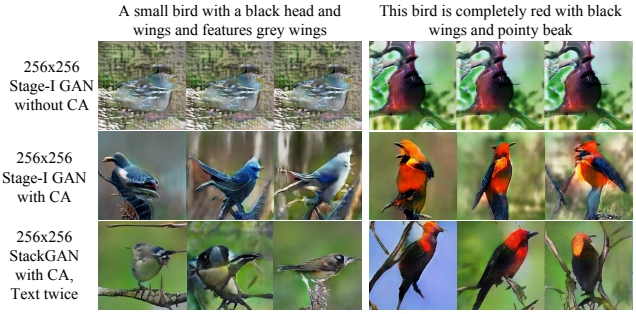


Figure 5. Conditioning Augmentation (CA) helps stabilize the training of conditional GAN and improves the diversity of the generated samples. (Row 1) without CA, Stage-I GAN fails to generate plausible 256×256 samples. Although different noise vector z is used for each column, the generated samples collapse to be the same for each input text description. (Row 2-3) with CA but fixing the noise vectors z, methods are still able to generate birds with different poses and viewpoints.

scriptions for each class of CUB and Oxford-102 test sets. For COCO dataset, 4k text descriptions are randomly selected from its validation set. For each sentence, 5 images are generated by each model. Given the same text descriptions, 10 users (not including any of the authors) are asked to rank the results by different methods. The average ranks by human users are calculated to evaluate all compared methods.

In addition, we use t-SNE [42] embedding method to visualize a large number (*e.g.*, 30,000 on the CUB test set) of high-dimensional images in a two-dimensional map. We observe that t-SNE is a good tool to examine the distribution of synthesized images and identify collapsed modes.

6.1. The component analysis of StackGAN-v1

In this section, we analyze different components of StackGAN-v1 on CUB dataset with our baseline models.

The design of StackGAN-v1. As shown in the first four rows of Table 2, if Stage-I GAN is directly used to generate images, the inception scores decrease significantly. Such performance drop can be well illustrated by results in Figure 5. As shown in the first row of Figure 5, Stage-I GAN fails to generate any plausible 256×256 samples without using Conditioning Augmentation (CA). Although Stage-I

Figure 6. For generated images (column 1), retrieving their nearest training images (columns 2-6) by utilizing Stage-II discriminator of StackGAN-v1 to extract visual features. The $L2$ distances between features are calculated for nearest-neighbor retrieval.

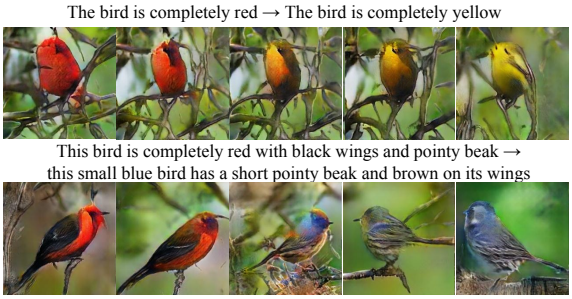


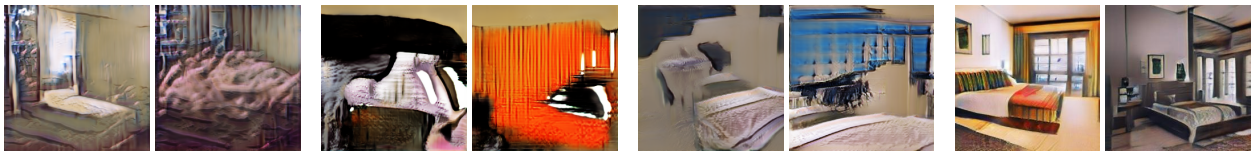
Figure 7. (Left to right) Images generated by interpolating two sentence embeddings. Gradual appearance changes from the first sentence's meaning to that of the second sentence can be observed. The noise vector z is fixed to be zeros for each row.

GAN with CA is able to generate more diverse 256×256 samples, those samples are not as realistic as samples generated by StackGAN-v1. It demonstrates the necessity of the proposed stacked structure. In addition, by decreasing the output resolution from 256×256 to 128×128, the inception score decreases from 3.70 to 3.35. Note that all images are scaled to 299 × 299 before calculating the inception score. Thus, if our StackGAN-v1 just increases the image size without adding more information, the inception score would remain the same for samples of different resolutions. Therefore, the decrease in inception score by 128×128 StackGAN-v1 demonstrates that our 256×256 StackGAN-v1 does add more details into the larger images. For the 256×256 StackGAN-v1, if the text is only input to Stage-I (denoted as "no Text twice"), the inception score decreases from 3.70 to 3.45. It indicates that processing text descriptions again at Stage-II helps refine Stage-I results. The same conclusion can be drawn from the results of 128×128 StackGAN-v1 models.

Figure 4 illustrates some examples of the Stage-I and Stage-II images generated by our StackGAN-v1. As shown in the first row of Figure 4, in most cases, Stage-I GAN is able to draw rough shapes and colors of objects given text descriptions. However, Stage-I images are usually blurry with various defects and missing details, especially for foreground objects. As shown in the second row, Stage-

Model	branch G_1	branch G_2	branch G_3	JCU	$\mathcal{L}_{C_i}$	inception score
StackGAN-v2	64×64	128×128	256×256	yes	yes	3.82 ± .06
StackGAN-v2-no-$\mathcal{L}_C$	64×64	128×128	256×256	yes	no	3.81± .04
StackGAN-v2-no-JCU	64×64	128×128	256×256	no	yes	3.77 ± .04
StackGAN-v2-G_3	removed	removed	256×256	yes	yes	3.49 ± .04
StackGAN-v2-3G_3	removed	removed	three 256×256	yes	yes	3.22 ± .02
StackGAN-v2-all256	256×256	256×256	256×256	yes	yes	2.89 ± .02

Table 3. Inception scores by our StackGAN-v2 and its baseline models on CUB test set. "JCU" means using the proposed discriminator that jointly approximates conditional and unconditional distributions.



(a) StackGAN-v2-all256 (b) StackGAN-v2-G_3 (c) StackGAN-v2-3G_3 (d) StackGAN-v2

This black and white and grey bird has a black bandit marking around it's eyes

(e) StackGAN-v2-all256 (f) StackGAN-v2-G_3 (g) StackGAN-v2-no-JCU (h) StackGAN-v2

Figure 8. Example images generated by the StackGAN-v2 and its baseline models on LSUN bedroom (top) and CUB (bottom) datasets.

II GAN generates 4× higher resolution images with more convincing details to better reflect corresponding text descriptions. For cases where Stage-I GAN has generated plausible shapes and colors, Stage-II GAN completes the details. For instance, in the 1st column of Figure 4, with a satisfactory Stage-I result, Stage-II GAN focuses on drawing the short beak and white color described in the text as well as details for the tail and legs. In all other examples, different degrees of details are added to Stage-II images. In many other cases, Stage-II GAN is able to correct the defects of Stage-I results by processing the text description again. For example, while the Stage-I image in the 5th column has a blue crown rather than the reddish brown crown described in the text, the defect is corrected by Stage-II GAN. In some extreme cases (*e.g.*, the 7th column of Figure 4), even when Stage-I GAN fails to draw a plausible shape, Stage-II GAN is able to generate reasonable objects. We also observe that StackGAN-v1 has the ability to transfer background from Stage-I images and fine-tune them to be more realistic with higher resolution at Stage-II.

Importantly, the StackGAN-v1 does not achieve good results by simply memorizing training samples but by capturing the complex underlying language-image relations. We extract visual features from our generated images and all training images by the Stage-II discriminator D of our StackGAN-v1. For each generated image, its nearest neighbors from the training set can be retrieved. By visually inspecting the retrieved images (see Figure 6), we conclude that the generated images have some similar characteristics with the training samples but are essentially different.

Conditioning Augmentation. We also investigate the efficacy of the proposed Conditioning Augmentation (CA). By removing it from StackGAN-v1 256×256 (denoted as "no CA" in Table 2), the inception score decreases from 3.70 to 3.31. Figure 5 also shows that 256×256 Stage-I GAN (and StackGAN-v1) with CA can generate birds with different poses and viewpoints from the same text embedding. In contrast, without using CA, samples generated by 256×256 Stage-I GAN collapse to nonsensical images due to the unstable training dynamics of GANs. Consequently, the proposed Conditioning Augmentation helps stabilize the conditional GAN training and improves the diversity of the generated samples because of its ability to encourage robustness to small perturbations along the latent manifold.

Sentence embedding interpolation. To further demonstrate that our StackGAN-v1 learns a smooth latent data manifold, we use it to generate images from linearly interpolated sentence embeddings, as shown in Figure 7. We fix the noise vector z, so the generated image is inferred from the given text description only. Images in the first row are generated by simple sentences made up by us. Those sentences contain only simple color descriptions. The results show that the generated images from interpolated embeddings can accurately reflect color changes and generate plausible bird shapes. The second row illustrates samples generated from more complex sentences, which contain

ImageNet dog [35] ImageNet cat [35] LSUN church [49]

Figure 9. Example images generated without (top) and with (bottom) the proposed color-consistency regularization for our StackGAN-v2 on ImageNet dog, ImageNet cat and LSUN church datasets. (Left to right) 64×64, 128×128, and 256×256 images by G_1, G_2, G_3, respectively.

Metric	Dataset	GAN-INT-CLS	GAWWN	Our StackGAN-v1	Our StackGAN-v2
Inception score	CUB	2.88 ± .04	3.62 ± .07	3.70 ± .04	**3.84 ± .06**
	Oxford	2.66 ± .03	/	**3.20 ± .01**	/
	COCO	7.88 ± .07	/	**8.45 ± .03**	/
Human rank	CUB	2.81 ± .03	1.99 ± .04	**1.37 ± .02**	/
	Oxford	1.87 ± .03	/	**1.13 ± .03**	/
	COCO	1.89 ± .04	/	**1.11 ± .03**	/

Table 4. Inception scores and average human ranks of our StackGAN-v1, StackGAN-v1, GAWWN [29], and GAN-INT-CLS [31] on CUB, Oxford-102, and MS-COCO datasets.

more details on bird appearances. The generated images change their primary color from red to blue, and change the wing color from black to brown.

6.2. The component analysis of StackGAN-v2

In this section, we analyze important components of the proposed StackGAN-v2. Table 3 lists models with different settings and their inception scores on the CUB test set. Figure 8 shows example images generated by different baseline models.

Our baseline models are built by removing or changing a certain component of the StackGAN-v2 model. By only approximating a single image distribution, the inception scores on CUB dramatically decrease from 3.82 to 3.49 for "StackGAN-v2-G_3" and to 2.89 for "StackGAN-v2-all256" (See Figures 8 (a-b) and (e-f)). Inspired by [9], we also build a baseline model with multiple discriminators, namely "StackGAN-v2-3G_3", as shown in Figure 8(c). Those discriminators have the same structure but different initializations. However, the results do not show improvement over "StackGAN-v2-G_3". "StackGAN-v2-no-JCU" replaces the joint conditional and unconditional discriminators with the conventional ones; its inception score is also lower than that of StackGAN-v2, which demonstrates the effectiveness of jointly approximating conditional and unconditional distributions. Another model "StackGAN-v2-no-$\mathcal{L}_{C_i}$" does not use the color-consistency regularization term. The inception score of this model on the conditional generation task on the CUB dataset does not show obvious decrease compared to StackGAN-v2, but qualitative results in Figure 9 do show that the color-consistency regularization has significant positive effects for the unconditional image synthesis task. It helps generators at different branches produce more coherent samples from the same noise input, resulting in much more realistic high-resolution images.

6.3. Comparison with state-of-the-art GAN models

To further demonstrate the effectiveness of the proposed method, we compare it with state-of-the-art GAN models on text-to-image synthesis [31, 29] and unconditional image synthesis [28, 51, 2, 22].

Text-to-image synthesis. We compare our results with the state-of-the-art text-to-image methods [29, 31] on CUB, Oxford-102 and COCO datasets. The inception scores and average human ranks for our proposed StackGAN and compared methods are reported in Table 4. Representative examples are compared in Figure 10, Figure 11 and Figure 12.

Compared with previous GANs models [31, 29], our StackGAN achieves the best inception score and average human rank on all three datasets. For example, compared with GAN-INT-CLS [31], StackGAN-v1 achieves 28.47% improvement in terms of inception score on CUB dataset (from 2.88 to 3.70), and 20.30% improvement on Oxford-102 (from 2.66 to 3.20). The better average human rank of our StackGAN also indicates our proposed method is able to generate more realistic samples conditioned on text descriptions. As shown in Figure 10, the 64×64 samples generated by GAN-INT-CLS [31] can only reflect the general shape and color of the birds. Their results lack vivid parts (*e.g.*, beak and legs) and convincing details in most cases, which make them neither realistic enough nor have sufficiently high resolution. By using additional conditioning variables on location constraints, GAWWN [29] obtains a better inception score on CUB dataset, which is still slightly

Figure 10. Example results by our StackGAN-v1, GAWWN [29], and GAN-INT-CLS [31] conditioned on text descriptions from CUB test set.

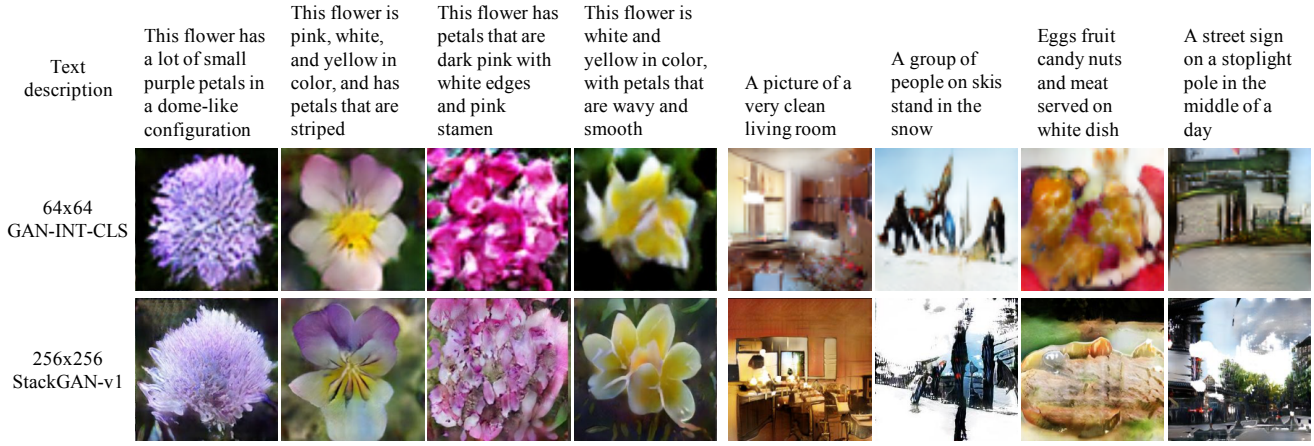
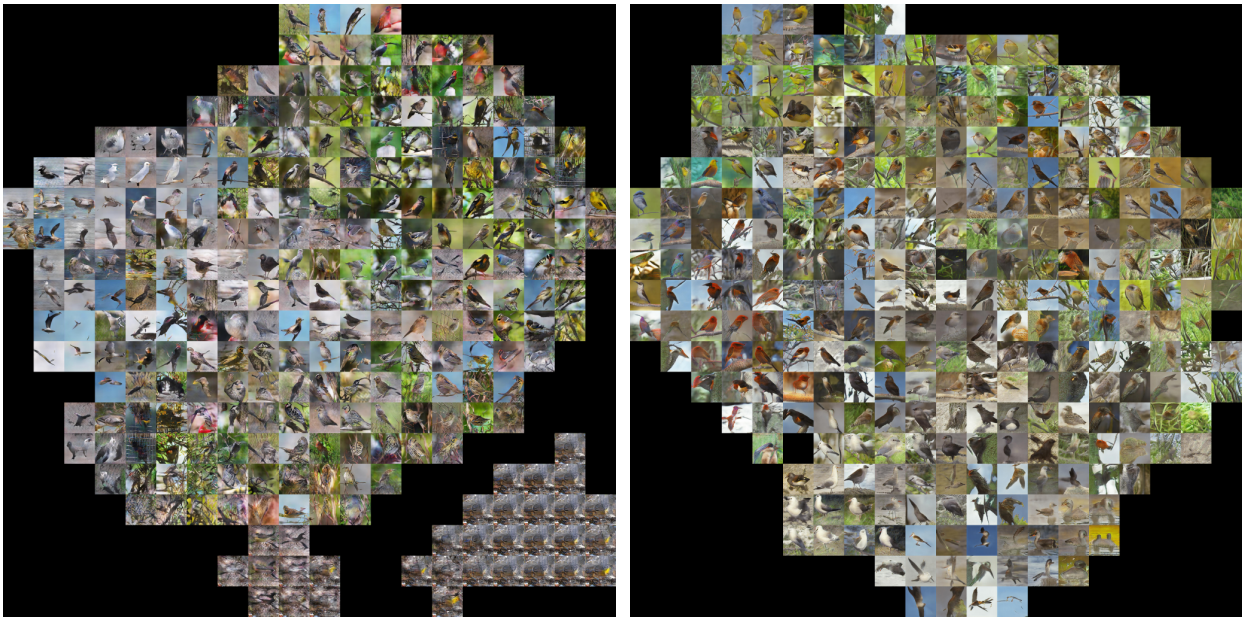


Figure 11. Example results by our StackGAN-v1 and GAN-INT-CLS [31] conditioned on text descriptions from Oxford-102 test set (leftmost four columns) and COCO validation set (rightmost four columns).

lower than ours. It generates higher resolution images with more details than GAN-INT-CLS, as shown in Figure 10. However, as mentioned by its authors, GAWWN fails to generate any plausible images when it is only conditioned on text descriptions [29]. In comparison, our StackGAN can generate 256×256 photo-realistic images from only text descriptions.

Compared with StackGAN-v1, our StackGAN-v2 further improves the inception score on on CUB dataset. We also utilize t-SNE [42] to conduct visualization and comparison. For each model, a large number of images are generated and embedded into the 2D plane. We first extract a 2048d CNN feature from each generated image using a pre-trained Inception model. Then, t-SNE algorithm is applied to embed the CNN features into a 2D plane, resulting a location for each image in the 2D plane. Due to page limits, Figure 12 only shows a 20×20 grid with a few compressed images for each dataset, where each generated image is mapped to its nearest grid location. As shown in Figure 12, StackGAN-v1 contains two collapsed modes (nonsensical images). Their model generates failure-case images if the Stage-I images are around a col-



(a) Samples by our StackGAN-v1 (b) Samples by our StackGAN-v2

Figure 12. Utilizing t-SNE to embed images generated by our StackGAN-v1 and StackGAN-v2 on the CUB test set.

lapsed mode. For our StackGAN-v2, since all the generators are trained jointly, they receive more regularization and thus result in more stable training behavior and less chance of mode collapse. We did not observe any collapsed nonsensical mode in the visualization of StackGAN-v2-generated images. While StackGAN-v2 is more advanced than StackGAN-v1 in many aspects, StackGAN-v1 has the advantage that it can be trained stage by stage. In the case that we want to generate very high resolution images, StackGAN-v1 can be generalized to stack many stages of GANs, and be trained stage by stage. However, StackGAN-v2 might be limited by the GPU memory because the whole end-to-end structure of StackGAN-v2 requires more memory than a single stage of StackGAN-v1.

Unconditional image synthesis. We compare our StackGAN-v2 with DCGAN [28], WGAN [2], EBGAN-PT [51], LSGAN [22] on the LSUN bedroom dataset. As shown in Figure 13, our StackGAN-v2 is able to generate 256×256 images with more photo-realistic details. In Figure 14, we also compare the 256×256 samples generated by StackGAN-v2 and EBGAN-PT. As shown in the figure, the samples generated by the two methods have the same resolution, but StackGAN-v2 generates more realistic ones (more recognizable dog faces with eyes and noses). These experiments demonstrate that our StackGAN-v2 outperforms the state-of-the-art methods for unconditional image generation in terms of image quality. Example images generated by our StackGAN-v2 on LSUN church and ImageNet cat datasets are presented in Figure 15.

7. Conclusions

In this paper, we propose Stacked Generative Adversarial Networks (StackGAN) for synthesizing *photo-realistic* images. The proposed StackGAN-v1 decomposes the text-to-image synthesis to a novel sketch-refinement process. Stage-I GAN sketches the object following basic color and shape constraints from given text descriptions. Stage-II GAN corrects the defects in Stage-I results and adds more details, yielding higher resolution images with better image quality. The proposed StackGAN-v2 jointly models multiple related distributions to stabilize training and further improve the quality of generated samples. Extensive quantitative and qualitative results demonstrate the effectiveness of our proposed method. While StackGAN-v2 is more advanced than StackGAN-v1 in many aspects, StackGAN-v1 has the potential to stack more stages of GANs. Compared to state-of-the-art GAN models, our method generates higher resolution images (*e.g.*, 256×256) with more photo-realistic details and diversity for both conditional and unconditional image generation tasks.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 1

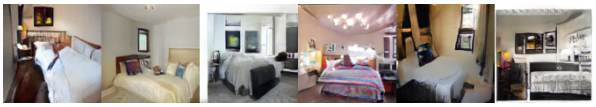
[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv:1701.07875*, 2017. 1, 8, 11, 13, 14

[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017. 3

Samples by DCGAN (Reported in [28])

Samples by WGAN (Reported in [2])

Samples by EBGAN-PT (Reported in [51])



Samples by LSGAN (Reported in [22])

256×256 Samples by our StackGAN-v2

Figure 13. Comparison of samples generated by models trained on LSUN bedroom [49] dataset.

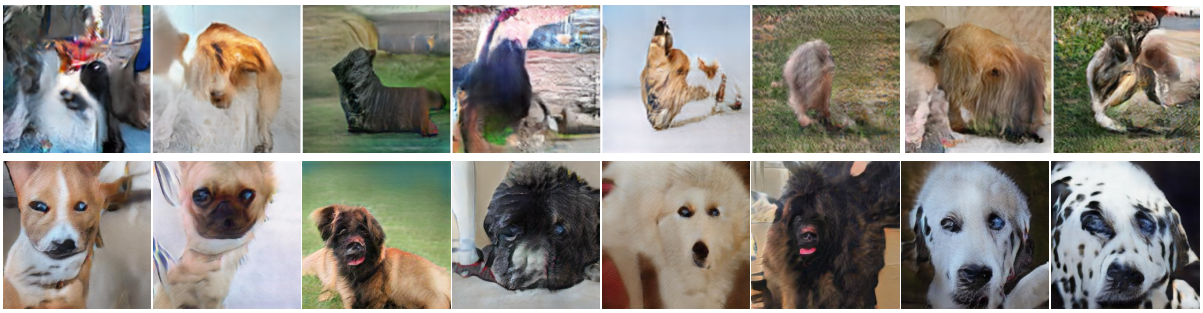


Figure 14. 256×256 samples generated by EBGAN-PT [51] (top) and our StackGAN-v2 (bottom) on ImageNet dog dataset [35].

Figure 15. 256×256 samples generated by our StackGAN-v2 on LSUN church [49] (top) and ImageNet cat [35] (bottom) datasets.

[4] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. In *ICLR*, 2017. 1, 3

[5] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv:1702.07983*, 2017. 1

[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016. 3

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

[8] C. Doersch. Tutorial on variational autoencoders. *arXiv:1606.05908*, 2016. 4

[9] I. P. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. In *ICLR*, 2017. 3, 11

[10] J. Gauthier. Conditional generative adversarial networks for convolutional face generation. *Technical report*, 2015. 3

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 1, 3, 6

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[13] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. In *CVPR*, 2017. 3

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 3

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5, 7

[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 3, 4

[18] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 4

[19] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 3

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 8

[21] E. Mansimov, E. Parisotto, L. J. Ba, and R. Salakhutdinov. Generating images from captions with attention. In *ICLR*, 2016. 3

[22] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *arXiv:1611.04076*, 2016. 8, 11, 13, 14

[23] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017. 3

[24] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014. 3

[25] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 2017. 3

[26] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICCVGIP*, 2008. 2, 8

[27] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In *ICML*, 2017. 3

[28] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 1, 3, 8, 11, 13, 14

[29] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *NIPS*, 2016. 3, 4, 8, 11, 12

[30] S. Reed, Z. Akata, B. Schiele, and H. Lee. Learning deep representations of fine-grained visual descriptions. In *CVPR*, 2016. 4, 8

[31] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *ICML*, 2016. 3, 4, 5, 8, 11, 12

[32] S. Reed, A. van den Oord, N. Kalchbrenner, V. Bapst, M. Botvinick, and N. de Freitas. Generating interpretable images with controllable structure. *Technical report*, 2016. 3

[33] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014. 3

[34] D. L. Ruderman. The statistics of natural images. *Network: Computation In Neural Systems*, 5(4):517–548, 1994. 1

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 8, 11, 14

[36] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016. 1, 3, 8

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 8

[38] C. K. Snderby, J. Caballero, L. Theis, W. Shi, and F. Huszar. Amortised map inference for image super-resolution. In *ICLR*, 2017. 1, 3

[39] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 3

[40] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016. 3

[41] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016. 3

[42] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9: 25792605, Nov 2008. 9, 12

[43] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, pages 3630–3638, 2016. 8

[44] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 1

[45] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 8

[46] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 3

[47] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016. 3

[48] J. Yang, A. Kannan, D. Batra, and D. Parikh. LR-GAN: layered recursive generative adversarial networks for image generation. In *ICLR*, 2017. 3

[49] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 2, 8, 11, 14

[50] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv:1612.03242*, 2016. 2

[51] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *ICLR*, 2017. 3, 8, 11, 13, 14

[52] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 3