StackGAN: Text to Photo-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

Synthesizing high-quality images from text descriptions is a challenging problem in computer vision and has many practical applications. Samples generated by existing text-to-image approaches can roughly reflect the meaning of the given descriptions, but they fail to contain necessary details and vivid object parts. In this paper, we propose Stacked Generative Adversarial Networks (StackGAN) to generate 256×256 photo-realistic images conditioned on text descriptions. We decompose the hard problem into more manageable sub-problems through a sketch-refinement process. The Stage-I GAN sketches the primitive shape and colors of the object based on the given text description, yielding Stage-I 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. It is able to rectify defects in Stage-I results and add compelling details with the refinement process. To improve the diversity of the synthesized images and stabilize the training of the conditional-GAN, we introduce a novel Conditioning Augmentation technique that encourages smoothness in the latent conditioning manifold. Extensive experiments and comparisons with state-of-the-arts on benchmark datasets demonstrate that the proposed method achieves significant improvements on generating photo-realistic images conditioned on text descriptions.

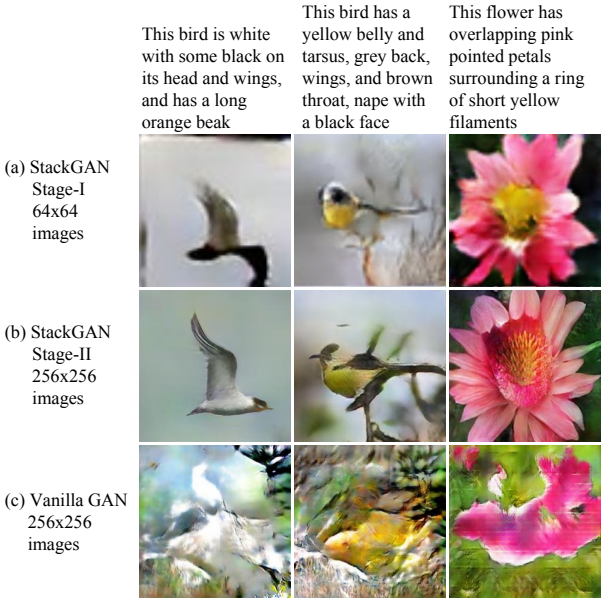


Figure 1. Comparison of the proposed StackGAN and a vanilla one-stage GAN for generating 256×256 images. (a) Given text descriptions, Stage-I of StackGAN sketches rough shapes and basic colors of objects, yielding low-resolution images. (b) Stage-II of StackGAN takes Stage-I results and text descriptions as inputs, and generates high-resolution images with photo-realistic details. (c) Results by a vanilla 256×256 GAN which simply adds more upsampling layers to state-of-the-art GAN-INT-CLS [26]. It is unable to generate any plausible images of 256×256 resolution.

1. Introduction

Generating photo-realistic images from text is an important problem and has tremendous applications, including photo-editing, computer-aided design, *etc*. Recently, Generative Adversarial Networks (GAN) [8, 5, 23] have shown promising results in synthesizing real-world images. Conditioned on given text descriptions, conditional-GANs [26, 24] are able to generate images that are highly related to the text meanings.

However, it is very difficult to train GAN to generate high-resolution photo-realistic images from text descriptions. Simply adding more upsampling layers in state-of-the-art GAN models for generating high-resolution (*e.g.*, 256×256) images generally results in training instability

and produces nonsensical outputs (see Figure 1(c)). The main difficulty for generating high-resolution images by GANs is that supports of natural image distribution and implied model distribution may not overlap in high dimensional pixel space [31, 1]. This problem is more severe as the image resolution increases. Reed *et al*. only succeeded in generating plausible 64×64 images conditioned on text descriptions [26], which usually lack details and vivid object parts, *e.g*., beaks and eyes of birds. Moreover, they were unable to synthesize higher resolution (*e.g*., 128×128) images without providing additional annotations of objects [24].

In analogy to how human painters draw, we decompose the problem of text to photo-realistic image synthesis into two more tractable sub-problems with Stacked Generative Adversarial Networks (StackGAN). Low-resolution images are first generated by our Stage-I GAN (see Figure 1(a)). On the top of our Stage-I GAN, we stack Stage-II GAN to generate realistic high-resolution (*e.g*., 256×256) images conditioned on Stage-I results and text descriptions (see Figure 1(b)). 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 object. The support of model distribution generated from a roughly aligned low-resolution image has better probability of intersecting with the support of image distribution. This is the underlying reason why Stage-II GAN is able to generate better high-resolution images.

In addition, 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. It allows small random perturbations in the conditioning manifold and increases the diversity of synthesized images.

The contribution of the proposed method is threefold: (1) We propose a novel Stacked Generative Adversarial Networks for synthesizing *photo-realistic* images from text descriptions. It decomposes the difficult problem of generating high-resolution images into more manageable subproblems and significantly improve the state of the art. The StackGAN 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) Extensive qualitative and quantitative experiments demonstrate the effectiveness of the overall model design as well as the effects of individual components, which provide useful information for designing future conditional GAN models. Our code is available at https://github.com/hanzhanggit/StackGAN.

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) [13, 28] formulated the problem with probabilistic graphical models whose goal was to maximize the lower bound of data likelihood. Autoregressive models (*e.g*., PixelRNN) [33] that utilized neural networks to model the conditional distribution of the pixel space have also generated appealing synthetic images. Recently, Generative Adversarial Networks (GAN) [8] 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. Several techniques [23, 29, 18, 1, 3] have been proposed to stabilize the training process and generate compelling results. An energy-based GAN [38] has also been proposed for more stable training behavior.

Built upon these generative models, conditional image generation has also been studied. Most methods utilized simple conditioning variables such as attributes or class labels [37, 34, 4, 22]. There is also work conditioned on images to generate images, including photo editing [2, 39], domain transfer [32, 12] and super-resolution [31, 15]. However, super-resolution methods [31, 15] 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*. [17] built an AlignDRAW model by learning to estimate alignment between text and the generating canvas. Reed *et al*. [27] used conditional PixelCNN to generate images using the text descriptions and object location constraints. Nguyen *et al*. [20] 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*. [26] successfully generated plausible 64×64 images for birds and flowers based on text descriptions. Their follow-up work [24] was able to generate 128×128 images by utilizing additional annotations on object part locations.

Besides using a single GAN for generating images, there is also work [36, 5, 10] that utilized a series of GANs for image generation. Wang *et al*. [36] factorized the indoor scene generation process into structure generation and style generation with the proposed S^2-GAN. In contrast, the second stage of our StackGAN aims to complete object details and correct defects of Stage-I results based on text descriptions. Denton *et al*. [5] 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. Concurrent to our

work, Huang *et al*. [10] also showed that they can generate better images by stacking several GANs to reconstruct the multi-level representations of a pre-trained discriminative model. However, they only succeeded in generating 32×32 images, while our method utilizes a simpler architecture to generate 256×256 images with photo-realistic details and sixty-four times more pixels.

3. Stacked Generative Adversarial Networks

To generate high-resolution images with photo-realistic details, we propose a simple yet effective Stacked Generative Adversarial Networks. It decomposes the text-to-image generative process into two stages (see Figure 2).

- **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.

3.1. Preliminaries

Generative Adversarial Networks (GAN) [8] 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,

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

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 [7, 19] 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.

3.2. 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 [26, 24], the text embedding is nonlinearly transformed to generate conditioning latent variables as the 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 [26, 24], 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 [6, 14], we add the following regularization term to the objective of the generator during training,

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

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.

3.3. 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 [25] 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),

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

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

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 [13], 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

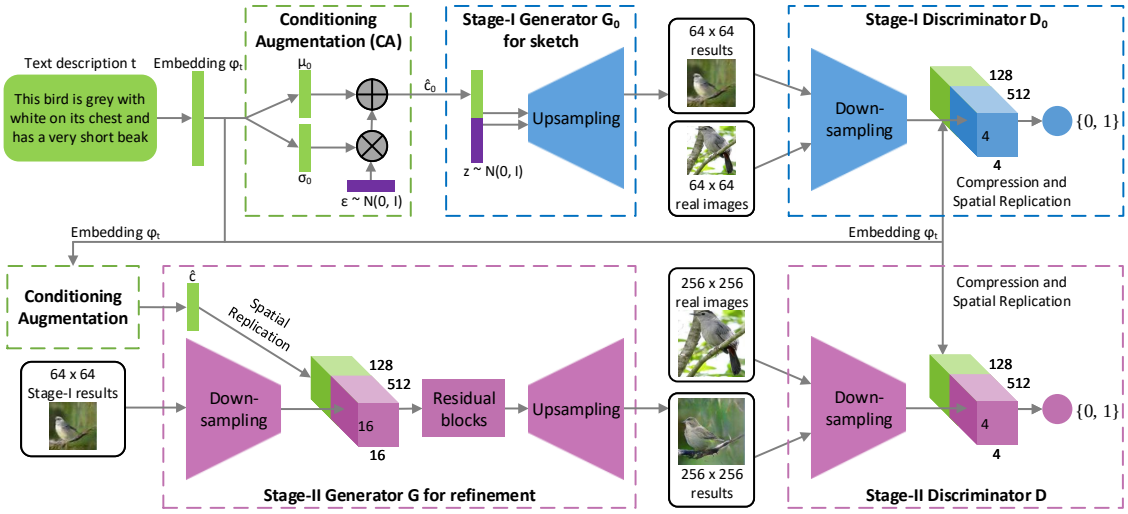


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects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

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.

3.4. 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),

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

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

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 [9]. 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*. [26] 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.

3.5. Implementation details

The up-sampling blocks consist of the nearest-neighbor upsampling followed by a 3×3 stride 1 convolution. Batch normalization [11] 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 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 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.

4. Experiments

To validate our method, we conduct extensive quantitative and qualitative evaluations. Two state-of-the-art methods on text-to-image synthesis, GAN-INT-CLS [26] and GAWWN [24], are compared. Results by the two compared methods are generated using the code released by their authors. In addition, we design several baseline models to investigate the overall design and important components of our proposed StackGAN. For the first baseline, we directly train Stage-I GAN for generating 64×64 and 256×256 images to investigate whether the proposed stacked structure and Conditioning Augmentation are beneficial. Then we modify our StackGAN 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 is useful.

4.1. Datasets and evaluation metrics

CUB [35] 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 [35], 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 [21] contains 8,189 images of flowers from 102 different categories. To show the generalization capability of our approach, a more challenging dataset, MS COCO [16] is also utilized for evaluation. Different from CUB and Oxford-102, the MS COCO dataset contains images with multiple objects and various backgrounds. It has a training set with 80k images and a validation set with 40k images. Each image in COCO has 5 descriptions, while 10 descriptions are provided by [25] for every image in CUB and Oxford-102 datasets. Following the experimental setup in [26], 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.

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" [29] for quantitative evaluation,

$$I = \exp(\mathbb{E}_{\boldsymbol{x}} D_{KL}(p(y|\boldsymbol{x}) \,||\, p(y))), \tag{7}$$

where $\boldsymbol{x}$ denotes one generated sample, and y is the label predicted by the Inception model [30]. 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 [29], 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 [29], 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 descriptions 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.

4.2. Quantitative and qualitative results

We compare our results with the state-of-the-art text-to-image methods [24, 26] on CUB, Oxford-102 and COCO

Figure 3. Example results by our StackGAN, GAWWN [24], and GAN-INT-CLS [26] conditioned on text descriptions from CUB test set.

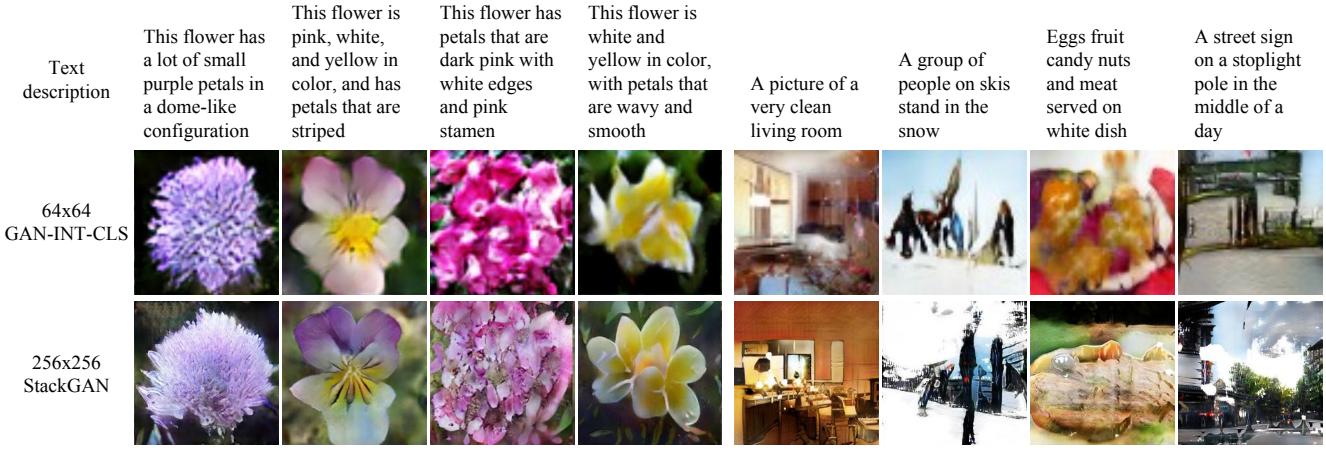


Figure 4. Example results by our StackGAN and GAN-INT-CLS [26] conditioned on text descriptions from Oxford-102 test set (leftmost four columns) and COCO validation set (rightmost four columns).

Metric	Dataset	GAN-INT-CLS	GAWWN	Our StackGAN
Inception score	CUB	2.88 ± .04	3.62 ± .07	**3.70 ± .04**
	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 1. Inception scores and average human ranks of our StackGAN, GAWWN [24], and GAN-INT-CLS [26] on CUB, Oxford-102, and MS-COCO datasets.

datasets. The inception scores and average human ranks for our proposed StackGAN and compared methods are reported in Table 1. Representative examples are compared in Figure 3 and Figure 4.

Our StackGAN achieves the best inception score and average human rank on all three datasets. Compared with GAN-INT-CLS [26], StackGAN 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 3, the 64×64 samples generated by GAN-INT-CLS [26] 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 loca-

Figure 5. Samples generated by our StackGAN 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.

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

tion constraints, GAWWN [24] obtains a better inception score on CUB dataset, which is still slightly lower than ours. It generates higher resolution images with more details than GAN-INT-CLS, as shown in Figure 3. However, as mentioned by its authors, GAWWN fails to generate any plausible images when it is only conditioned on text descriptions [24]. In comparison, our StackGAN can generate 256×256 photo-realistic images from only text descriptions.

Figure 5 illustrates some examples of the Stage-I and Stage-II images generated by our StackGAN. As shown in the first row of Figure 5, 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-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 5, 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 5), 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 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 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. For each generated image, its nearest neighbors from the training set can be retrieved. By visually inspecting the retrieved images (see Figure 6), we can conclude that the generated images have some similar characteristics with the training samples but are essentially different.

4.3. Component analysis

In this subsection, we analyze different components of StackGAN on CUB dataset with our baseline models. The inception scores for those baselines are reported in Table 2.

The design of StackGAN. 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 7. As shown in the first row of Figure 7, Stage-I GAN fails to generate any plausible 256×256 samples without

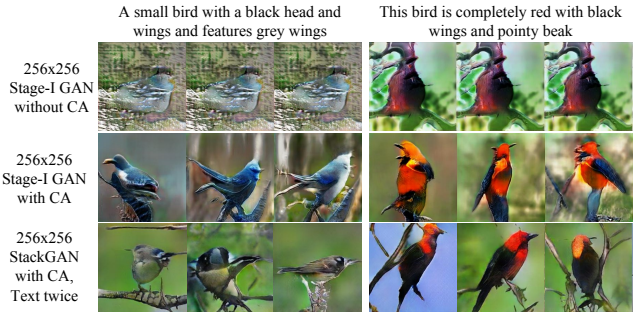


Figure 7. Conditioning Augmentation (CA) helps stabilize the training of conditional GAN and improves the diversity of the generated samples. (Row 1) without CA, Stage-I GAN fails to generate plausible 256×256 samples. Although different noise vector z is used for each column, the generated samples collapse 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.

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	yes	no	3.13 ± .03
	no	yes	3.20 ± .03
	yes	yes	3.35 ± .02
256×256 StackGAN	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.

using Conditioning Augmentation (CA). Although Stage-I GAN with CA is able to generate more diverse 256×256 samples, those samples are not as realistic as samples generated by StackGAN. 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 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 demonstrates that our 256×256 StackGAN does add more details into the larger images. For the 256×256 StackGAN, 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 models.

Conditioning Augmentation. We also investigate the efficacy of the proposed Conditioning Augmentation (CA). By removing it from StackGAN 256×256 (denoted as "no CA" in Table 2), the inception score decreases from 3.70 to 3.31. Figure 7 also shows that 256×256 Stage-I GAN (and StackGAN) 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.

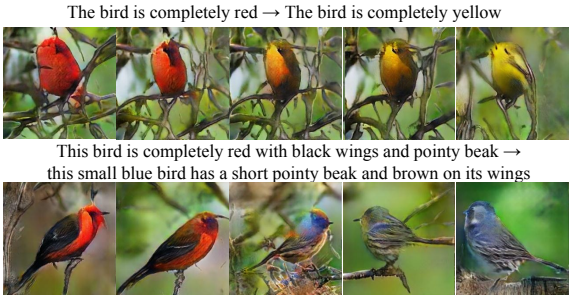


Figure 8. (Left to right) Images generated by interpolating two sentence embeddings. Gradual appearance changes from the first sentence's meaning to that of the second sentence can be observed. The noise vector z is fixed to be zeros for each row.

Sentence embedding interpolation. To further demonstrate that our StackGAN learns a smooth latent data manifold, we use it to generate images from linearly interpolated sentence embeddings, as shown in Figure 8. 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 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.

5. Conclusions

In this paper, we propose Stacked Generative Adversarial Networks (StackGAN) with Conditioning Augmentation for synthesizing *photo-realistic* images. The proposed method 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. Extensive quantitative and qualitative results demonstrate the effectiveness of our proposed method. Compared to existing text-to-image generative models, our method generates higher resolution images (*e.g.*, 256×256) with more photo-realistic details and diversity.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 2

[2] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017. 2

[3] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. In *ICLR*, 2017. 2

[4] 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. 2

[5] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 1, 2

[6] C. Doersch. Tutorial on variational autoencoders. *arXiv:1606.05908*, 2016. 3

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

[8] 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, 2, 3

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[10] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. In *CVPR*, 2017. 2, 3

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

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2, 3

[14] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 3

[15] 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. 2

[16] 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. 5

[17] E. Mansimov, E. Parisotto, L. J. Ba, and R. Salakhutdinov. Generating images from captions with attention. In *ICLR*, 2016. 2

[18] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017. 2

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

[20] 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. 2

[21] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICCVGIP*, 2008. 5

[22] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017. 2

[23] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 1, 2

[24] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *NIPS*, 2016. 1, 2, 3, 5, 6, 7

[25] S. Reed, Z. Akata, B. Schiele, and H. Lee. Learning deep representations of fine-grained visual descriptions. In *CVPR*, 2016. 3, 5

[26] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *ICML*, 2016. 1, 2, 3, 5, 6

[27] 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. 2

[28] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014. 2

[29] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016. 2, 5

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[31] C. K. Snderby, J. Caballero, L. Theis, W. Shi, and F. Huszar. Amortised map inference for image super-resolution. In *ICLR*, 2017. 2

[32] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 2

[33] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016. 2

[34] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016. 2

[35] 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. 5

[36] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 2

[37] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016. 2

[38] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *ICLR*, 2017. 2

[39] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2

Supplementary Materials

More Results of Birds and Flowers

Additional Results on CUB Dataset

This bird sits close to the ground with his short yellow tarsus and feet; his bill is long and is also yellow and his color is mostly white with a black crown and primary feathers

Stage-I images

Stage-II images

A large bird has large thighs and large wings that have white wingbars

Stage-I images

Stage-II images

This smaller brown bird has white stripes on the coverts, wingbars and secondaries

Stage-I images

Stage-II images

A cardinal looking bird, but fatter with gray wings, an orange head, and black eyerings

Stage-I images

Stage-II images

The small bird has a red head with feathers that fade from red to gray from head to tail

Stage-I images

Stage-II images

This bird is black with green and has a very short beak

Stage-I images

Stage-II images

This bird is light brown, gray, and yellow in color, with a light colored beak

Stage-I images

Stage-II images

This bird has wings that are black and has a white belly

Stage-I images

Stage-II images

Additional Results on Oxford-102 Dataset

This flower is yellow in color, with petals that are vertically layered

Stage-I images

Stage-II images

This flower has white petals with a yellow tip and a yellow pistil

Stage-I images

Stage-II images

A flower with small pink petals and a massive central orange and black stamen cluster

Stage-I images

Stage-II images

This flower is white, pink, and yellow in color, and has petals that are multi colored

Stage-I images

Stage-II images

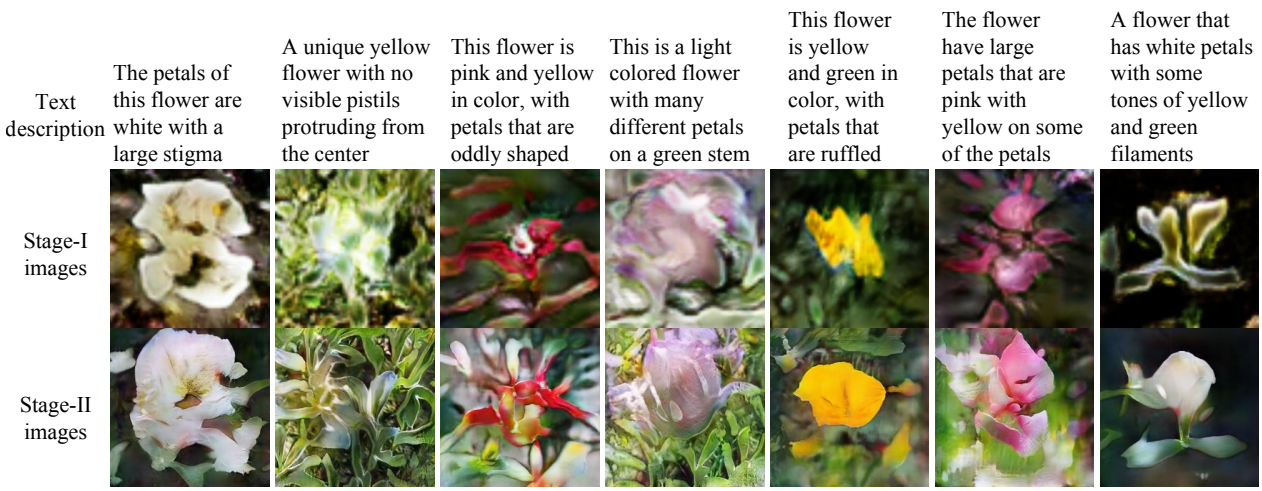
This flower has petals that are yellow with shades of orange

Failure Cases

The main reason for failure cases is that Stage-I GAN fails to generate plausible rough shapes or colors of the objects.

CUB failure cases:

Oxford-102 failure cases:



Beyond Birds and Flowers: Results on MS COCO

Results on COCO dataset demonstrate the generalization capability of our approach on images with multiple objects and complex backgrounds.

Diverse samples can be generated for each text description.

More results. We observe that StackGAN is able to synthesize reasonable images in various cases, although the image quality is lower than the results of birds and flowers. In the future work, we aim to further investigate more sophisticated stacked architectures for generating more complex scenes.