“Text to Image Generation Using StackGAN”

A PROJECT REPORT

Submitted by

Mr. Shubham Sanjaykumar kale
Mr. Mahesh Sudhir Zarkar

in partial fulfillment for the award of the degree

of

Bachelor of Technology

in

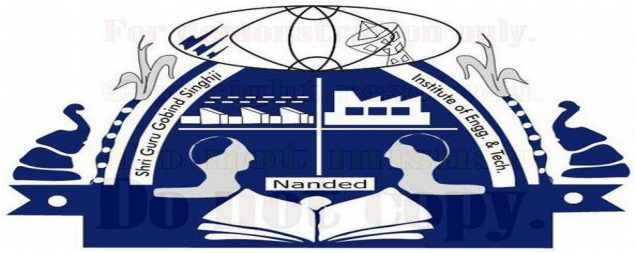
Computer Science and Engineering

at

SHRI GURU GOBIND SINGHJI
INSTITUTE OF ENGINEERING AND TECHNOLOGY,
VISHNUPURI, NANDED
(MAHARASHTRA STATE)
PIN 431 606 INDIA

May 2018

SHRI GURU GOBIND SINGHJI INSTITUTE OF ENGINEERING AND TECHNOLOGY



CERTIFICATE

This is to certify **Shubham Sanjaykumar Kale (2014BCS019), Mahesh Sudhir Zarkar (2014BCS152)** have carried out the project work entitled ***"Text To Image Generation Using StackGAN"*** for the award of degree of **Bachelor of Technology in Computer Science and Engineering** from Shri Guru Gobind Singhji Institute of Engineering & Technology, Vishnupuri, Nanded (M. S.) under my supervision. The project embodies results of original work, and studies are carried out by the student himself and the contents of the project work do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/ Institution.

Guide

Dr. Amit V. Nandedkar
(Assistant Professor)

Head

Prof. Dr. U. V. Kulkarni

Date: May 2018

ACKNOWLEDGEMENT

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time we are utilizing to thank all the people who have been concerned with this project.

Primarily we would like to thank our Guide **Dr. Amit V. Nandedkar**, whose valuable guidance has been the ones that helped us patch this project and make it full proof success. His suggestions and his instructions have served as the major contributor towards the completion of the project.

Then we would like to thank our parents who have helped us with their valuable suggestions and their guidance has been very helpful in various phases of the completion of the project. Last but not the least we would like to thank our classmates who have helped me a lot.

Shubham Sanjaykumar Kale
(2014BCS019)
Mahesh Sudhir Zarkar
(2014BCS152)

Contents

List of Figures

List of Tables

ABSTRACT

Synthesizing images from text descriptions remains a challenging problem in computer vision.Samples generated by old text-to-image approaches can roughly reflect the meaning of the given descriptions, but they fail to contain necessary details and vivid object parts. Contemporary AI systems have made remarkable breakthroughs in generating images of a single instance from a specific category. In this project, we implement improved stacked generative adversarial networks (StackGAN) model to delve into the text-to-image generation task with multiple instances from a broader variety of categories compared with previous researches, The StackGAN model demonstrates the capability to generate images with complex scene composition consisting of multiple objects based on the semantics of the given input text and proves the potential to generate high-resolution multi-instance images
.

Chapter 1

Introduction

1.1 Preamble

In this Project, we implement Stacked Generative Adversarial Networks (StackGAN) to generate 256x256 photo-realistic images conditioned on text descriptions. This method decomposes 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 multiforms. Exten-

sive 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.

1.2 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) formulated the problem with probabilistic graphical models whose goal was to maximize the lower bound of data likelihood.

Autoregressive models (e.g., PixelRNN) that utilized neural networks to model the conditional distribution of the pixel space have also generated appealing synthetic images. Recently, Generative Adversarial Networks (GAN) have shown promising performance for generating sharper images. But training instability makes it hard for GAN models to generate high-resolution images. Several techniques have been proposed to stabilize the training process and generate compelling results.

An energy-based GAN 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 . There is also work con-

ditioned on images to generate images, including photo editing domain transfer and super-resolution. However, super-resolution methods can only add limited details to low-resolution images and can not correct large defects as the implemented StackGAN does.

1.3 Problem Definition

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 . On the top of our Stage-I GAN, we stack Stage-II GAN to generate realistic high-resolution (e.g., 256x256) images conditioned on Stage-I results and text descriptions . 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.

1.4 Application

GANs have been used to produce samples of photorealistic images for the purposes of visualizing new interior/industrial design, shoes, bags and clothing items or items for computer games' scenes. These networks were reported to be used by Facebook. Recently, GANs have modeled patterns of motion in video. They have also been used to reconstruct 3D models of objects from images and to improve astronomical images.

In 2017 a fully convolutional feedforward GAN was used for image enhancement using automated texture synthesis in combination with perceptual loss. The system focused on realistic textures rather than pixel-accuracy. The result was a higher image quality at high magnification. The strength of GANs to generate samples that look like real data is investigated for path planning applications in the context of smart mobility and transportation applications. A GAN can also be trained on crowd-sourced trajectory data to recommend new paths to desired destinations.

Chapter 2

Literature Survey

2.1 Introduction

The promise of deep learning is to discover rich, hierarchical models that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. So far, the most striking successes in deep learning have involved discriminative models, usually those that map a high-dimensional, rich sensory input to a class label .

These striking successes have primarily been based on the backpropagation and dropout algorithms, using piecewise linear units which have a particularly well-behaved gradient . Deep generative models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise

linear units in the generative context.A new generative model estimation procedure that sidesteps these difficulties is GAN. In the adversarial nets framework proposed by Ian J Goodfellow, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution.

The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine artifacts.

2.2 Brief Literature Overview

The proposed new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the train-

ing data distribution and D equal to half everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation.

2.3 Existing system Overview

Recent years have witnessed tremendous success of deep neural networks (DNNs), especially the kind of bottom-up neural networks trained for discriminative tasks. In particular, Convolutional Neural Networks (CNNs) have achieved impressive accuracy on the challenging ImageNet classification benchmark. Interestingly, it has been shown that CNNs trained on ImageNet for classification can learn representations that are transferable to other tasks , and even to other modalities .

However, bottom-up discriminative models are focused on learning useful representations from data, being incapable of capturing the data distribution.Learning top-down generative models that can explain complex data distribution is a long-standing problem in machine learning research. The expressive power of deep neural networks makes them natural candidates for generative models, and several recent works have shown promising results. While state-of-the-art DNNs can rival human performance in certain discriminative tasks, current best deep generative models still fail when there are large

variations in the data distribution. A natural question therefore arises: can we leverage the hierarchical representations in a discriminatively trained model to help the learning of top-down generative models.

In this project, we implement a generative model named Stacked Generative Adversarial Networks (SGAN). Our model consists of a top-down stack of GANs, each trained to generate "plausible" lower-level representations conditioned on higher-level representations. Similar to the image discriminator in the original GAN model which is trained to distinguish "fake" images from "real" ones, a set of representation discriminators are introduced that are trained to distinguish "fake" representations from "real" representations.

The adversarial loss introduced by the representation discriminator forces the intermediate representations of the SGAN to lie on the manifold of the bottom-up DNN's representation space. In addition to the adversarial loss, we also introduce a conditional loss that imposes each generator to use the higher-level conditional information, and a novel entropy loss that encourages each generator to generate diverse representations. By stacking several GANs in a topdown way and using the top-most GAN to receive labels and the bottom-most GAN to generate images, SGAN can be trained to model the data distribution conditioned on class labels. Through extensive experiments performed, it is demonstrated that the SGAN

is able to generate images of much higher quality than a vanilla GAN. In particular, the models obtains state-of-the-art Inception scores on CIFAR-10 dataset.

Chapter 3

Implemented System

3.1 Project Analysis

3.1.1 Overview of Neural Networks

Neural Networks have made great progress. They now recognize images and voice at levels comparable to humans. They are also able to understand natural language with a good accuracy. But, even then, the talk of automating human tasks with machines looks a bit far fetched. After all, humans do much more than just recognizing image / voice or understanding what people around us are saying,Following are few examples where we need human creativity (at least as of now):

- Train an artificial author which can write an article and explain data science concepts to a community in a very simplistic manner by learning from past articles.
- You are not able to buy a painting from a famous

painter which might be too expensive. Can you create an artificial painter which can paint like any famous artist by learning from his / her past collections?

These are definitely difficult to automate tasks, but **Generative Adversarial Networks (GANs)** have started making some of these tasks possible.

3.1.2 What are GANs?

Generative adversarial networks (GANs) are a class of neural networks that are used in unsupervised machine learning. They help to solve such tasks as image generation from descriptions, getting high resolution images from low resolution ones, predicting which drug could treat a certain disease, retrieving images that contain a given pattern, etc.

GANs were introduced by Ian Goodfellow in 2014. They aren't the only approach of neural networks in unsupervised learning. There's also the Boltzmann machine (Geo rey Hinton and Terry Sejnowski, 1985) and Autoencoders (Dana H. Ballard, 1987). Both of them are dedicated to extract features from data by learning the identity function f(x) = x and both of them rely on Markov chains to train or to generate samples.

Generative adversarial networks were designed to avoid using Markov chains because of the high computational

cost of the latter. Another advantage relative to Boltzmann machines is that the Generator function has much fewer restrictions (there are only a few probability distributions that admit Markov chain sampling).

3.1.3 How do GANs work?

There are two main components of a GAN – Generator Neural Network and Discriminator Neural Network. The

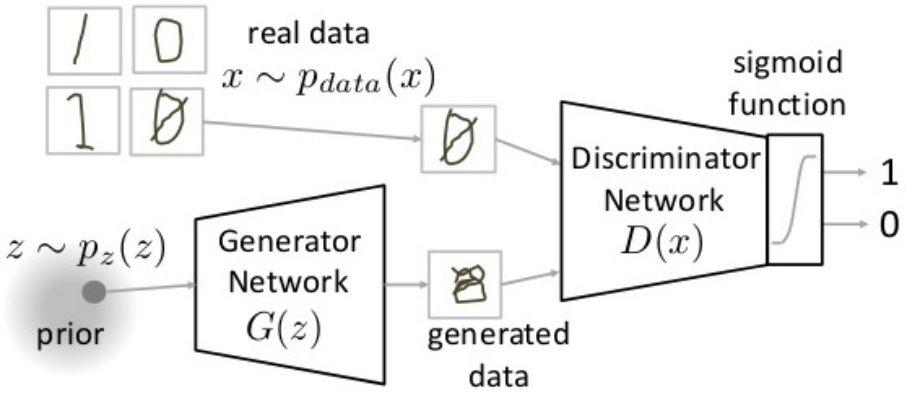


Figure 3.1: Architecture of Generative Adversarial Network

Generator Network takes an random input and tries to generate a sample of data. In the above image, we can see that generator G(z) takes a input z from p(z), where z is a sample from probability distribution p(z). It then generates a data which is then fed into a discriminator network D(x). The task of Discriminator Network is to take input either from the real data or from the generator and try to predict whether the input is real or generated. It takes an input x from $p_{data}(x)$ where $p_{data}(x)$is our

real data distribution. D(x) then solves a binary classification problem using sigmoid function giving output in the range 0 to 1.

Following are the notations that will be used to formalize the GAN, Now the training of GAN is done as

$p_{data}(x)$	the distribution of real data
x	sample from $p_{data}(x)$
P(z)	distribution of generator
Z	sample from p(z)
G(z)	Generator Network
D(x)	Discriminator Network

Table 3.1: Notations

a fight between generator and discriminator. This can be represented mathematically as

$$\min_G \max_D V(D,G)$$

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

Figure 3.2: Objective function

In the function V(D, G) the first term is entropy that the data from real distribution ($p_{data}(x)$) passes through the discriminator . The discriminator tries to maximize this to 1. The second term is entropy that the data from random input (p(z)) passes through the generator, which then generates a fake sample which is then passed through

the discriminator to identify the fakeness . In this term, discriminator tries to maximize it to 0 (i.e. the log probability that the data from generated is fake is equal to 0). So overall, the discriminator is trying to maximize our function V.

On the other hand, the task of generator is exactly opposite, i.e. it tries to minimize the function V so that the differentiation between real and fake data is bare minimum. This method of training a GAN is taken from game theory called the minimax game.

3.1.4 Parts of training GAN

So broadly a training phase has two main subparts and they are done sequentially

- Pass 1: Train discriminator and freeze generator (freezing means setting training as false. The network does only forward pass and no backpropagation is applied)

- Pass 2: Train generator and freeze discriminator

3.1.5 Steps to train a GAN

Step 1: Define the problem
Do you want to generate fake images or fake text. Here you should completely define the problem and collect data for it.

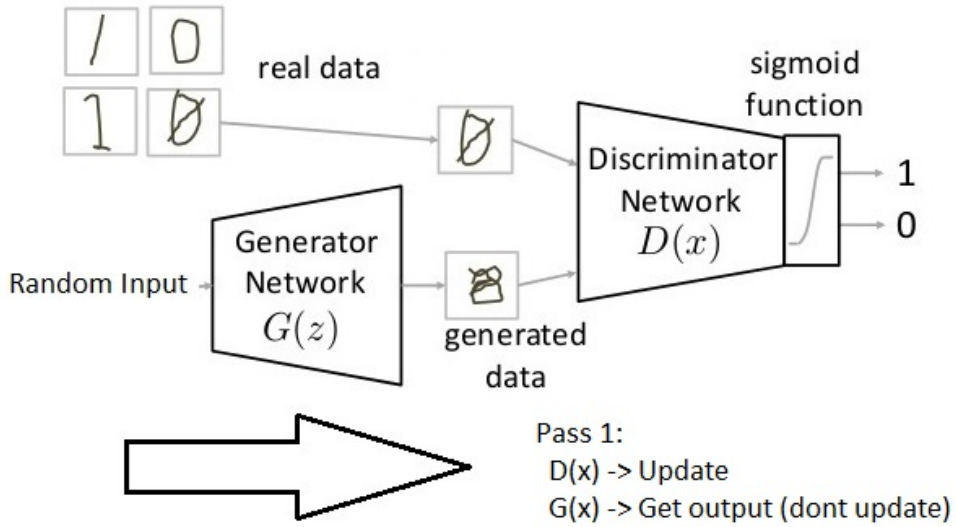


Figure 3.3: Pass-1

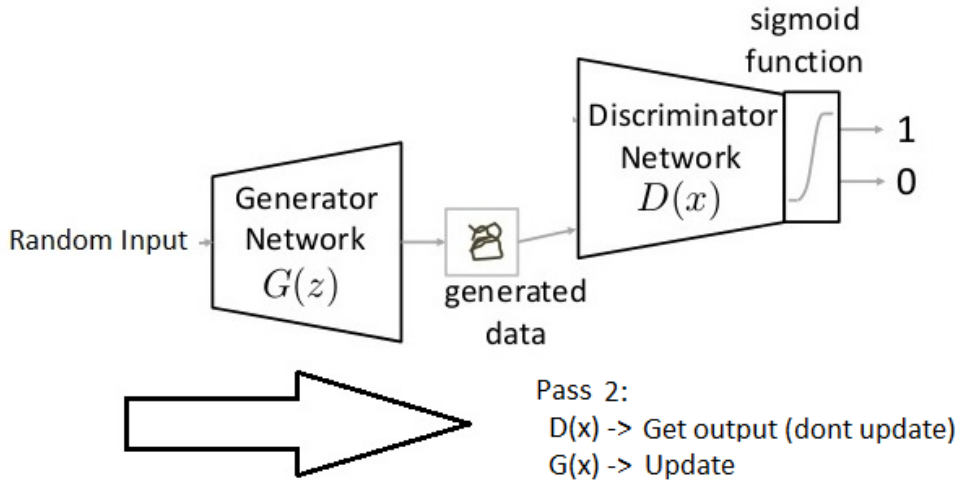


Figure 3.4: Pass-2

Step 2: Define architecture of GAN

Define how your GAN should look like. Should both your generator and discriminator be multi layer perceptrons, or convolutional neural networks? This step will depend on what problem you are trying to solve.

Step 3: Train Discriminator on real data for n epochs

Get the data you want to generate fake data and train the discriminator to correctly predict them as real. Here value N can be any natural number between 1 and infinity.

Step 4: Generate fake inputs for generator and train discriminator on fake data

Get generated data and let the discriminator correctly predict them as fake.

Step 5: Train generator with the output of discriminator

Now when the discriminator is trained, you can get its predictions and use it as an objective for training the generator. Train the generator to fool the discriminator.

Step 6: Repeat step 3 to step 5 for a few epochs

Step 7: Check if the fake data manually if it seems legit. If it seems appropriate, stop training, else go to step 3

This is a bit of a manual task, as hand evaluating the data is the best way to check the fakeness. When this step is

over, you can evaluate whether the GAN is performing well enough.

3.1.6 Challenges with GANs

The most important roadblock while training a GAN is stability. If you start to train a GAN, and the discriminator part is much powerful that its generator counterpart, the generator would fail to train effectively. This will in turn affect training of your GAN. On the other hand, if the discriminator is too lenient; it would let literally any image be generated. And this will mean that your GAN is useless.
Another way to glance at stability of GAN is to look as a holistic convergence problem. Both generator and discriminator are fighting against each other to get one step ahead of the other. Also, they are dependent on each other for efficient training. If one of them fails, the whole system fails. So you have to make sure they don't explode.

- **Problem with Counting :** GANs fail to differentiate how many of a particular object should occur at a location.
- **Problems with Perspective :** GANs fail to adapt to 3D objects. It doesn't understand perspective, i.e.difference between frontview and backview.It gives flat (2D) representation of 3D objects.

- **Problems with Global Structures :** Same as the problem with perspective, GANs do not understand a holistic structure.

3.2 Project Design

3.2.1 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 3.5)

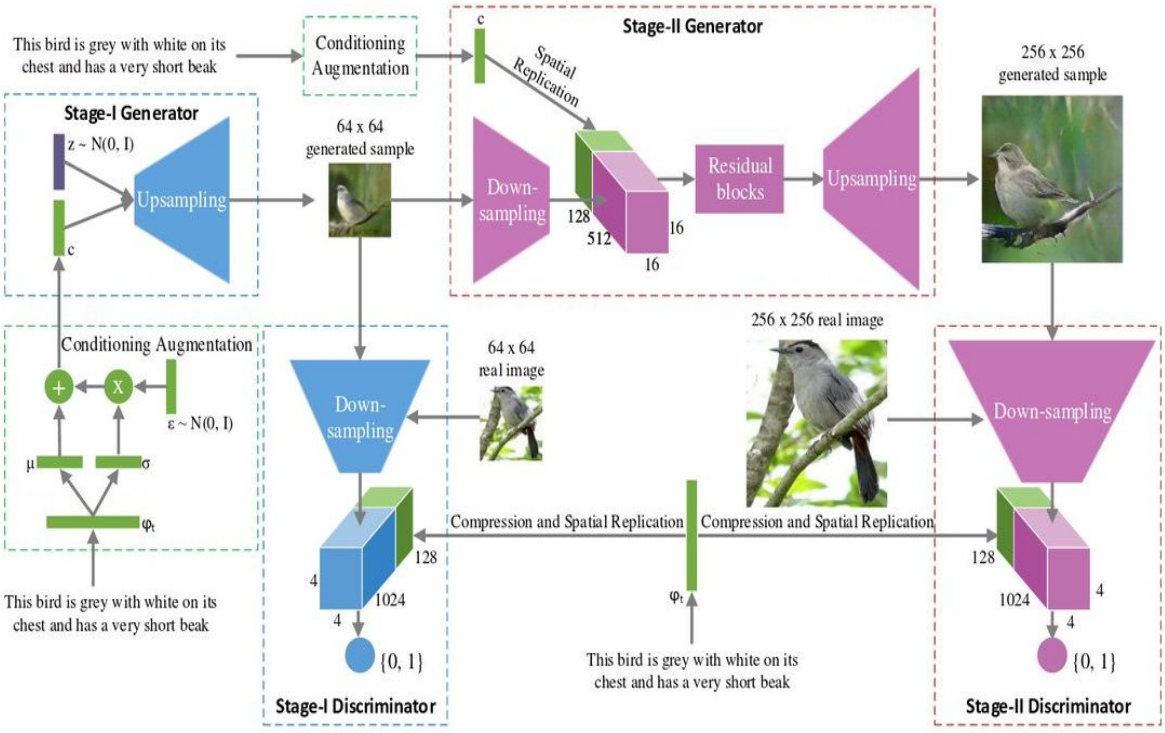


Figure 3.5: Architecture of StackGAN

- 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.

Conditional GAN is an extension of GAN where φ_t 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.

3.2.2 Conditioning Augmentation

As shown in Figure 3.5, the text description t is first encoded by an encoder, yielding a text embedding φ_t. In previous works , 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 (greater than 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 reduce this problem, we introduce a Conditioning Augmentation technique to produce additional conditioning variables ĉ. In contrast to the fixed conditioning text variable c in , we randomly sample the latent variables ĉ from an independent Gaussian distribution $N(\mu(\varphi_t),\Sigma(\varphi_t))$, 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 , we add the following regularization term to the objective of the generator during training,

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

,

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.

3.2.3 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 The Gaussian conditioning variables $\hat{c}_o$ for text embedding are sampled from $N(\mu_o(\varphi_t)$,$\Sigma_o(\varphi_t))$,to capture the meaning of φ_t with variations. Conditioned on $\hat{c}_o$and random variable z, Stage-I GAN trains the discriminatorD_o and the generator G_o by alternatively maximizing L_{D_o} and minimizing L_{G_o} ,

$$L_{D_o} = E_{(I_o,t)\simeq p_{data}}[logD_o(I_o, \varphi_t)]+ \\ E_{z\simeq p_z, t\simeq p_{data}}[log(1 - D_o(G_o(z, \hat{c}_o), \varphi_t))]$$

$$L_{G_o} = E_{z\simeq p_z, t\simeq p_{data}}[log(1 - D_o(G_o(z, \hat{c}_o), \varphi_t))]+ \\ \lambda D_{KL}(N(\mu(\varphi_t), \Sigma(\varphi_t))||N(0, I))$$

where the real image I_o 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). λ is a regularization parameter that bal-

ances the two terms . We set $\lambda = 1$ for all the experiments. Using the reparameterization trick , both $\mu_o(\varphi_t)$ and $\Sigma_o(\varphi_t)$ are learned jointly with the rest of the network.

Model Architecture

For the generator G_o , to obtain text conditioning variable $\hat{c}_o$, the textfed into a fully connected layer to generate μ_o and σ_o(σ_o are the values in the diagonal of Σ_o) for the Gaussian distribution $N(\mu_o(\varphi_t),\Sigma_o(\varphi_t))$ $\hat{c_o}$ are then sampled from the Gaussian distribution. Our N_g dimensional conditioning vector $\hat{c_o}$ is computed by $\hat{c_o} = \mu_o + \sigma_o \odot \epsilon$ (where $\odot$ is the element-wise multiplication, $\epsilon \simeq N$ (0, I)). Then, $\hat{c_o}$ is concatenated with a N_z dimensional noise vector to generate a W_o x H_o image by a series of up-sampling blocks.

For the discriminator D_o , 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 x M_d x N_d tensor. Meanwhile, the image is fed through a series of down-sampling blocks until it has M_d X 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 11 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.

3.2.4 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 $s_o = G_o$ (z, $\hat{c}_o$) and Gaussian latent variables ĉ,the discriminator D and generator G in Stage-II GAN are trained by alternatively maximizing L_D and minimizing L_G

$$L_D = E_{(I,t)\simeq p_{data}}[logD(I, \varphi_t)]+ \\ E_{s_o\simeq p_{G_o}, t\simeq p_{data}}[log(1 - D(G(s_o, \hat{c}), \varphi_t))]$$

$$L_G = E_{s_o\simeq p_{G_o}, t\simeq p_{data}}[log(1 - D(G(s_o, \hat{c}), \varphi_t))]+ \\ \lambda D_{KL}(N(\mu(\varphi_t), \Sigma(\varphi_t))||N(0, I))$$

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_o . Gaussian conditioning variables ĉ used in this stage and $\hat{c}_o$ 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. 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 x M_g x N_g tensor. Meanwhile, the Stage-I result s_o generated by Stage-I GAN is fed into several down-sampling blocks (i.e., encoder) until it has a spatial size of M_g x 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 x 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. 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.

3.2.5 Implementation details

The up-sampling blocks consist of the nearest-neighbor upsampling followed by a 3x3 stride 1 convolution. Batch normalization and ReLU activation are applied after every convolution except the last one. The residual blocks consist of 3x3 stride 1 convolutions, Batch normalization and ReLU. Two residual blocks are used in 128x128 StackGAN models while four are used in 256x256 models. The down-sampling blocks consist of 4x4 stride 2 convolutions, Batch normalization and LeakyReLU, except that the first one does not have Batch normalization.

By default, $N_g = 128$, $N_z = 100$, $M_g = 16$, $M_d = 4$, $N_d = 128$, $W_o = H_o = 64$ and W = H = 256. For training, we first iteratively train D_o and G_o 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 half of its previous value every 100 epochs.

3.3 Project Study

3.3.1 Dataset Specification

The CUB dataset 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 , 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. 10 descriptions are provided by for every image in CUB dataset. we split CUB dataset into class-disjoint training and test sets.

3.3.2 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" for quantitative evaluation,

$$I = exp(E_x D_{KL}(p(y|x)||p(y))),$$

where x denotes one generated sample, and y is the label predicted by the Inception model. The intuition behind this metric is that good models should generate diverse but meaningful images. Therefore, the KL divergence be-

tween the marginal distribution p(y) and the conditional distribution $p(y|x)$ should be large.

Although the inception score has shown to well correlate with human perception on visual quality of samples , it cannot reflect whether the generated images are well conditioned on the given text descriptions.

3.3.3 Significant Results

Following are some of the images generated from provided text descriptions :

Figure 3.6: Image-1

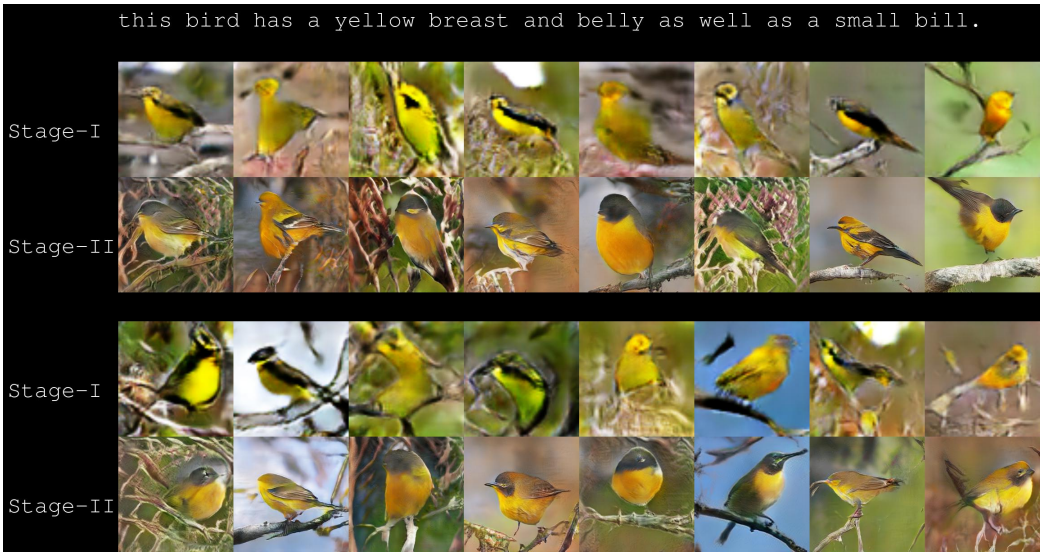


Figure 3.7: Image-2

Figure 3.8: Image-3

Figure 3.9: Image-4

Figure 3.10: Image-5

Chapter 4

Future Scope

GANs are used extensively in all the cases where generative models and techniques like Variational Auto Encoder(VAE), pixelRNNs, Deep Boltzmann Machine(DBM) are used. For example text, image, video generation, the advantage of using GANs is that they are faster and easier to train than traditional approaches like boltzman machines. GANs do not require any approximation and can be trained end-to-end through differentiable networks, which makes them a better effective models to be employed now and in the future. Many interesting use cases for GANs are in the field of Cryptography and Hacking (e.g encoder-decoder models).

Chapter 5

Conclusions

In this Project , we implement **Stacked Generative Adversarial Networks (StackGAN)** with Conditioning Augmentation for synthesizing photo-realistic images. The implemented 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 the implemented method. Compared to old text-to-image generative models, the method implemented generates higher resolution images (e.g. 256 x 256) with more photo-realistic details and diversity.

Chapter 6

References

1. *StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks*
   Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas

2. *Generative Adversarial Networks*
   Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

3. *Stacked Generative Adversarial Networks*
   Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, Serge Belongie

4. *Text-to-Image Generation Using Multi-Instance StackGan*
   Alex Fu, Yiju Hou Department of Computer Science Stanford University.

5. *Skip-Thought Vectors*
   Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard

S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler

6. *Deep Learning (Adaptive Computation and Machine Learning series)*[book]
   By Ian Goodfellow, Yoshua Bengio, Aaron Courville

7. *Neural Networks and Deep Learning*[book]
   By Michael Nielsen