Generative Adversarial Text to Image Synthesis

PROJECT REPORT

submitted towards the partial fulfillment of the

requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

AMIT MANCHANDA
14116013

ANSHUL JAIN
14116016

Under the guidance of:

Dr. VINOD PANKAJAKSHAN

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)

May, 2018

CANDIDATE'S DECLARATION

We declare that the work presented in this report with title "**Generative Adversarial Text to Image Synthesis**" towards the fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** submitted in the **Dept. of Electronics & Communication Engineering**, **Indian Institute of Technology, Roorkee**, India is an authentic record of our own work carried out during the period **from August 2017 to May 2018** under the supervision of **DR. VINOD PANKAJAKSHAN**, Assistant Professor, IIT Roorkee. The content of this report has not been submitted by us for the award of any other degree of this or any other institute.

DATE : ..... SIGNED: .....

PLACE: ..... (AMIT MANCHANDA)
ENROLLMENT NO.: 14116013

DATE : ..... SIGNED: .....

PLACE: ..... (ANSHUL JAIN)
ENROLLMENT NO.: 14116067

CERTIFICATE

This is to certify that the statement made by the candidates is correct to the best of my knowledge and belief.

DATE : .....

SIGNED: .....

(DR. VINOD PANKAJAKSHAN)
ASSISTANT PROFESSOR
DEPT. OF ELECTRONICS AND COMMUNICATION
IIT ROORKEE

ACKNOWLEDGEMENTS

FIRST and foremost, we would like to express our sincere gratitude towards our guide **Dr. VINOD PANKAJAKSHAN**, Assistant Professor, Dept. of Electronics and Communication Engineering, IIT Roorkee for his ideal guidance throughout the entire period. We want to thank him for the insightful discussions and constructive criticisms which certainly enhanced our knowledge as well as improved our skills. His constant encouragement, support and motivation were key to overcome all the difficult and struggling phases. Although he had a lot of teaching and research responsibilities, he was always able to find time for discussion and advice that significantly improved our work and for which we are truly thankful.

Finally, we appreciate the help of all our friends for keeping us motivated and providing with valuable insights as a part of various healthy discussions.

Abstract

Generative Adversarial Networks have shown striking results in the unconditional and conditional task but have been limited by the size and is not able to generate fine-grained details from text. In this report, we implement several techniques to successfully obtain a model for synthesizing images using text descriptions. Attention Generative Adversarial Network, that uses attention mechanism at multiple stages to generate fine-grained images. It improves the details by selecting the words on which it need to focus. We also implement image-word loss, Deep Attentional Multimodal Similarity Modal, to be used while training of the model. It generates 64×64, 128×128 and 256×256 dimensions of photo-realistic quality. We have also explored the use of conditional Wasserstein Generative Adversarial Network for generating images as Wasserstein distance provide insightful representation of the distance in between two low dimension distributions and therefore have shown plausible results in generating non-conditional images.

Table of Contents

List of Figures

List of Tables

Table **Page**

CHAPTER 1

INTRODUCTION

Human beings have the ability to imagine and mentally picturize an image just by analyzing the textual description of the scene or the object. For example, *"An apple is lying on a wooden table."*, this sentence provides us with a mental image of the information that the sentence is trying to convey. However, there can be multiple possible outcomes which would be able to correctly depict the entire information of the sentence. Artificial synthesis of images through textual descriptions could have numerous applications in visual editing, picture based summarization, digital designing, and animation generation.

This challenging problems has two subproblems: first is to learn to represent textual sentences into an encoded embedding that captures the important visual details required to draw the image and second one is to develop a generative model that would use the encoded embedding to synthesize an image that could be mistaken for real. The recent advancements in Deep learning has made huge progress in both the domains i.e. Natural Language Processing and generative modeling.

In recent year, Recurrent Neural Networks and Long Short term Memory networks have been able to encode the text sequences very efficiently by retaining the long term temporal dependencies. Also, advancements in generative modeling by using Variational Autoencoders and Generative Adversarial Networks have begun to generate highly realistic images. We build on these previous works and employ techniques from the two fields.

We aim at generating image pixel values directly from raw text in the form of a caption - a single sentence description of the image. For example feeding an input *"A red flower with a yellow stigma"* should generate an image corresponding to this caption. To accomplish this translation of text to images, the caption needs to be encoded to capture the discriminative text features. The encoded sentence will then be used to condition a generative adversarial model to generate images corresponding to the caption.

One challenging aspect of text to image synthesis is that synthesizing images using Generative models, and GANs in particular, conditioned on textual description leads to the problem of multi-modality in output. A single caption could lead to many possible images each of which could be aesthetically correct and covers all the information in the caption.

Figure 1.1: Example results for image synthesizing model using captions.

The further report is organised as follows. We discuss related works in chapter 2 and in chapter 3, we briefly describe the necessary background work required to understand the concepts involved with the methods. In chapter 4, we discuss the approach and the algorithms. Chapter 5 explains the implementation details such as hyperparameter details. Chapter 6 shows the results of our experiments followed by future works and conclusion in chapter 7 and 8 respectively.

CHAPTER 2

RELATED WORK

Generating images from text is a multimodal learning in which we learn shared representation between different modalities and synthesis unavailable data in one modality conditioned on other. Audio and video multimodal learning has been achieved [10] using stacked autoencoders. Srivastava and Salakhutdinov [11] modeled text tags with images using a deep Boltzmann machine. The earliest work of multimodal prediction with mathematical justification was proposed in 2014[12]. Other Variational Autoencoder approaches have been used in this direction works on maximizing the lower bound of data likelihood. Researchers have also used deep deconvolutional network to generate 3D chair models[13]. The draw model of Variational Autoencoders(VAE) has also proved very successful in conditioning the VAE with text embeddings to produce images from captions [14]. They used skip-thought vectors for text embeddings for conditioning the VAE.

For a long time major focus in previous works was retrieval of images from a text query or vice versa. However, from the past couple of years people have been using recurrent neural network architectures for generating text descriptions of images [15]. They uses a Long-Short Term Memory [16] on the top of a deep convolutional neural network architecture to generate caption on real image using MS COCO [17] and other datasets. The trained encoder of an RNN can be coupled with a multimodal network involving other domains like images, sound etc. to train across multiple domains using transfer learning. Attention Mechanism [18] has also been introduced to identify the regions on which a particular word focuses on. Figure 2.1 shows the architecture of this multi-modal network.

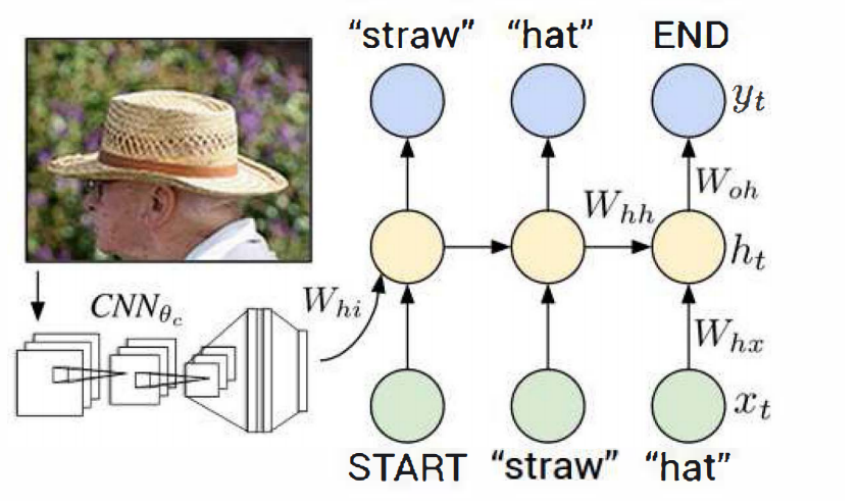

Figure 2.1: Image Captioning Using multi-modal networks (RNN and CNN) [1].

Recently, people have been looking into Generative Adversarial Networks[19] composed of deep recurrent and convolutional networks to generate realistic images with assuring results. GANs is composed of two separate models where both of them compete for a zero-sum game and end up improving each other based on Game Theory. But GANs are highly unstable and require adequate hyperparameter tuning to generate high resolution images. A lot of research had been made [2], [20], [21], [22], [23], [24] in stabilizing the training with Deep Convolutional Generative Adversarial Network (DCGAN) as a major breakthrough.

DCGAN generates images from random noise using a deconvolutional network as generator to synthesis images using principles of generative models. The work showed that it is possible to condition on image synthesis with class specific informations. Figure 2.2 shows the output of DCGAN architecture conditioned on images. They use all-convolutional layers with striding, removal of all dense layers after convolution, batch normalization at each step, along with ReLU activation at each stage except the last which uses tanh layer and LeakyReLU activation in case of discriminator.

[4] proposed the first differential conditional GANs architecture conditioned on text description in place of class labels, from character level to pixel level. It generated 64×64 realistic images with the help of matching aware discriminator and manifold interpolation regularizer. Using series of GANs for generating detailed images have also been studied [5]. The first stage generates the basic structure of the description and the job of the second stage GAN is to correct the defects of GAN-1 and complete object details. Therefore by

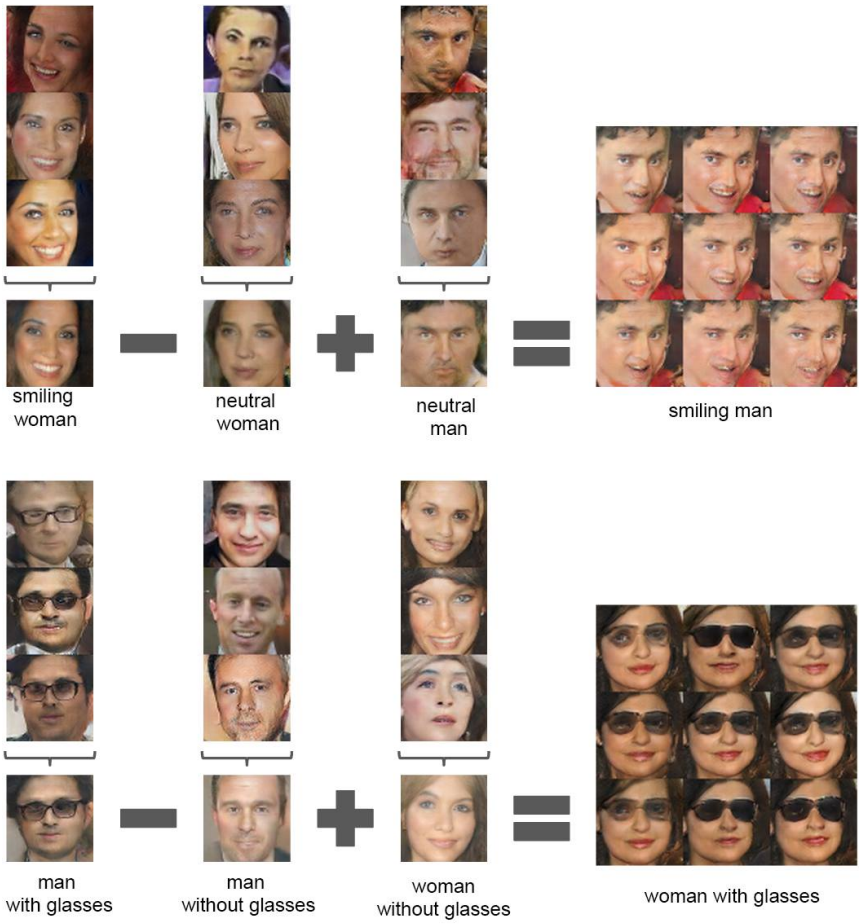

Figure 2.2: Images generated via DCGAN generator [2].

stacking GANs, people have tried generating 256×256 images.

Other uses of GANs include generating super-resolution images[25], for photo-editing[26] and domain transfer[27]. AlignDRAW[28], extented work of Deep Recurrent Attention Writer implemented a similar approach to attend to important words and PixelCNN[29] used multi-scale model to generate images but their iterative optimization process was inefficient. AlignDRAW has achieved results in a way that it is able to generate proper images according to the text description, but the images are not realistic. Other approaches tried adding class labels along with condition and by generating synthesized captions.

CHAPTER 3

BACKGROUND

3.1 Generative Adversarial Networks

Majority of work on deep generative models work on the principle of maximizing the log likelihood. One of the most successful of them is Deep Boltzmann Machines [30]. They provided a method for generating parametric specification of a probability distribution function which would be able to approximate the distribution of real training data. Such models generally require numerous approximations in the log-likelihood algorithm to work successfully. Motivated by these difficulties, "generative machines" were developed. They do not explicitly represented the log likelihood but use other approximation algorithms which do not require parametric specification. Generative Adversarial Networks[19] or GANs are based on the idea of a generative machine that can be trained by using the backpropagation algorithm. They are based on the idea of using a differentiable generator network which is paired along with a discriminator network. The model maps the samples of latent variables $\mathbf{z}$ to samples of real data or to distributions over real samples using a differentiable function, typically implemented by a neural network.

GANs work on the principle of game theory and set up a minimax game between two players namely generator and discriminator. The generator and discriminator compete against each other. The generator G parameterized by θ generates samples using the random noise $\mathbf{z}$, $G(\mathbf{z};\theta)$. The generated samples are intended to be drawn from the same distribution as of training data $\mathbf{x}$. The other player discriminator D tries to discriminate

between the samples drawn from the real training distribution and those drawn from the generator. The discriminator learns by classifying inputs two classes (real or fake). It emits a probability $D(\mathbf{x})$ or $D(G(\mathbf{z}))$ based on whether the input is from the training data or the generator. The generator's aim is to fool the discriminator by producing as real looking samples as possible and discriminator's aim is to correctly identify the synthesized samples from the real ones. In mathematical terms, GANs are structured probabilistic models having latent variables $\mathbf{z}$(random noise) and observed variables $\mathbf{x}$, as shown in Figure 3.1.

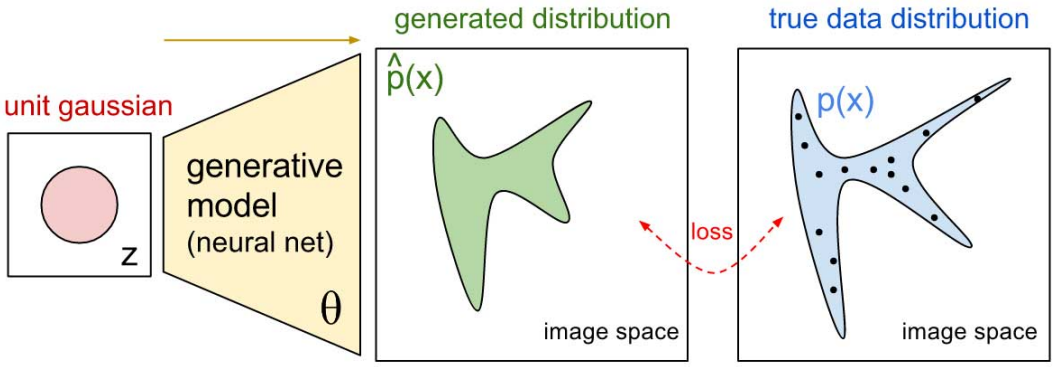

Figure 3.1: Generative adversarial Network Architecture

Both players are associated with a cost function which is also commonly called loss function that are defined in terms of their parameters. The discriminator has to minimize $J^{(D)}(\theta^{(D)},\theta^{(G)})$ and must do by adjusting only $\theta^{(D)}$. On the other hand, the generator has to minimize $J^{(G)}(\theta^{(D)},\theta^{(G)})$ by controlling $\theta^{(G)}$. The cost function of both players is also dependent on each others parameters but they cannot modify the other's parameters. This scenario can be better described as a game. The optimum solution to this game is a point in parameter space where the two mentioned cost functions are jointly minimized and all the other neighboring points endure greater or equal cost.

The cost function of discriminator is shown in Equation 3.1 where p_{data} represents the probability distribution of true data. It is same as the standard cross entropy loss.

$$J^{(D)}(\theta^{(D)},\theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D(G(z))) \tag{3.1}$$

The simplest version to the two player game of GAN is the zero-sum game in which the costs of all the players sum to zero. Hence,

$$J^{(G)} = -J^{(D)} \tag{3.2}$$

Zero sum games involves minimization of the total loss through one parameter and maximization through other. Hence they are also called minimax games. The value function which is to be optimized for GANs then becomes:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)} \log D(x) + \mathbb{E}_{z\sim p_z(z)} \log(1 - D(G(z))) \tag{3.3}$$

However, this generator loss is not optimal to be used in practical purposes. This is because, when the discriminator rejects the generators sample with high confidence, generator's gradient vanishes and does not allow the parameters to be trained for generator. In practice, following cost function is used for generator.

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{z\sim p_z} \log(D(G(z))) \tag{3.4}$$

3.1.1 DCGAN

The practical GAN architectures used today are mostly based on the DCGAN (Deep Convolutional GAN) architecture[2]. It specifies a set of guidelines which are important for stable implementation and training of GANs. Some of the key insights of DCGAN were to:

- The architecture is based upon the all-convolutional net which contains only convolutional layers. Strided convolution is used instead of pooling layers to increase the spatial dimension of the output.
- The fully connected hidden layers were removed for deeper architectures.
- Batch Normalization was extensively used to stabilize the training by normalizing the input to have zero mean and unit variance. This was essential in deep generators to initialize the training and avoiding the problem of mode-collapse.
- Using ReLU activation in generator and Leaky Relu in discriminator. The last layer of generator uses tanh activation. Both generator and discriminator use the Adam optimizer.

The architecture of DCGAN is also explained in Figure 3.2. They have shown good results when trained to generate images such as images of bedrooms and birds. They have also shown that their latent codes can indulge in simple arithmetic operations to generate meaningful outcomes as demonstrated in figure 2.2.

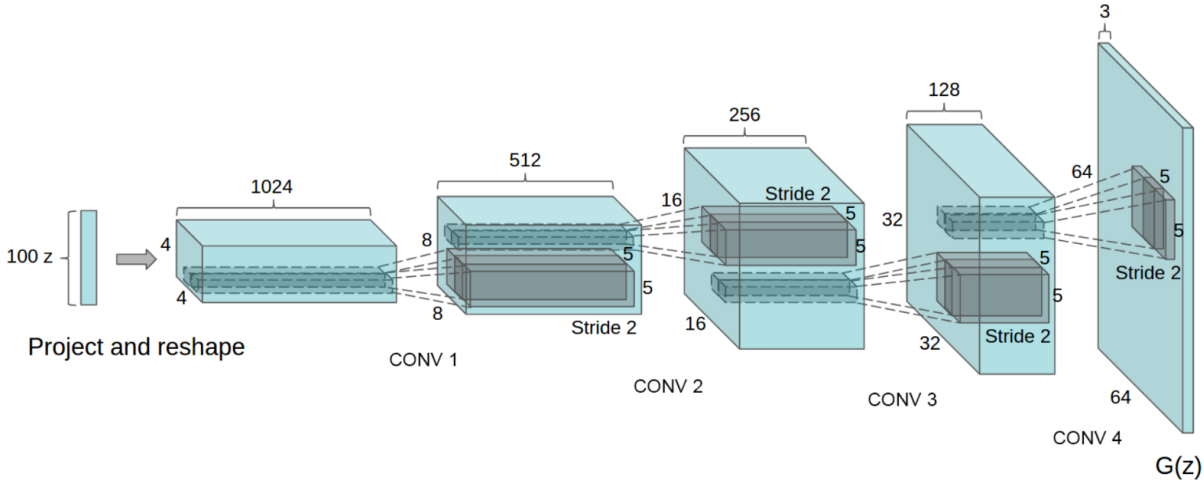

Figure 3.2: DCGAN generator used for LSUN dataset [2].

3.1.2 Conditional GANs

In an unconditional generative model, we do not have control on the data being generated. However, by providing some additional information about the kind of samples required, data generation could be directed. This additional information could be in the form of class or context of the image required. There has been recent work on Condtional GAN[31] to synthesize an image conditioned by a class label c. This is achieved by slight modification of the losses defined in the DCGAN above. The generator now has to generate an image which not only appears real, but also corresponds to the class label c. Also, the discriminator has to distinguish between a real and fake image and also make sure the image and the class label correspond to each other. This is achieved by the following objective function.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)} \log D(x|c) + \mathbb{E}_{z\sim p_z(z)} \log(1 - D(G(z|c))) \tag{3.5}$$

3.1.3 Wasserstein GANs

Any generative model aims to minimize the difference between the real distribution and learned distribution. However, this might not be possible in traditional GAN model if the true distribution and modeled distribution do not overlap. It might not be able to

learn the true distribution if the discriminator does not provide enough information to the generator. The Wasserstein GAN or WGAN[32] leverages on this fact and models a known distribution function to the desired distribution. To achieve this task, it needs to compute the distance between the real and model distributions.

WGAN make use of Wasserstein distance or commonly known as Earth-Mover distance to distinguish the two probability distributions and compute the amount of dissimilarity. To define EM distance intuitively, if we consider the probability distributions as the piles of dirt, then EM distance is the minimum "cost" of converting one pile into another. The cost is defined as the mass of dirt moved times the distance of movement.

We want to compute $P_\theta = g_\theta(z)$ to match P_r. Here, P_θ is the modeled distribution which is modeled as a generator network g dependent on parameter θ. We can model the critic f_w for the Wasserstein distance given a fixed g_θ. We can then use backpropagation to get the gradient for θ

$$\nabla_\theta W(P_r, P_\theta) = \nabla_\theta(\mathbb{E}_{x\sim p_r}[f_w(x)] + \mathbb{E}_{z\sim p_z}[f_w(g_\theta(z))]) \tag{3.6}$$

$$= -\mathbb{E}_{z\sim p_z}[\nabla_\theta f_w(g_\theta(z))] \tag{3.7}$$

The training can now be summarized in the following steps:

- Compute an approximate value of $W(P_r, P_\theta)$ by training f_w to convergence.
- Compute the gradient as shown in equation 3.7.
- Update the parameter θ and repeat the process.

WGANs have proved to improve stability. Here, the discriminator is trained many more times than generator which proves better in generator training for WGAN. Also, the authors of the paper[32] claim that they experienced no mode-collapse by this approach.

3.2 Text Embeddings

Algorithms does not understand string, it only understand numbers. Therefore before any Natural Language Processing work, a major task is to convert words/sentences to vector. The most simpler approach is Bag of Words in which we just show the presence of any word in a string. It doesn't hold the word order and thus a important part of information is lost. For it "This is cow" and "Is this cow" would be similar.

Distributed representation of words[33] in a N dimensional vector space helps to group similar words together. For example, vec("Delhi") - vec("India") + vec("France") is closest to vec("Paris"). Similar representations can be learnt for sentences with varying length also in which it maps a sentence into a fixed size feature vector.

3.2.1 Recurrent Neural Network

Recurrent Neural Networks(RNN) have been widely used for sequential data, when there is some information in the time domain of the signal. This is because the inner neuron stores/retains the overall learnt feature from the previous inputs. RNN has loops that carries old information back to the neuron while reading the new input. Figure 3.3 explain the structure and they way one can intuitively understand it.

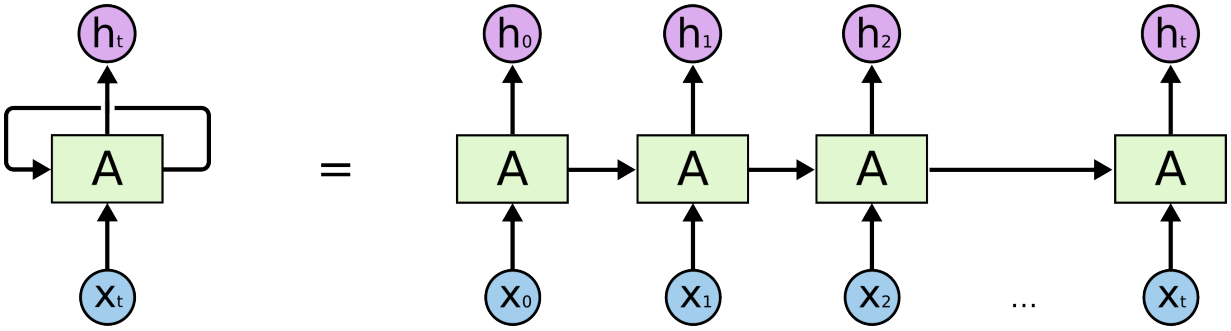

Figure 3.3: Unrolled Recurrent Neural Network

One of the major drawback of RNN is that, in native form it can't have long-term dependencies due to vanishing gradient. **Long Short-Term Memory (LSTM)** networks is a variation of RNN designed specifically for this purpose. Remembering long term information is the unique selling point. Native RNN have a single tanh function as the activation while LSTMs deploy 3 gates and 2 activation functions to keep the memory and current state saved. The three gates/mechanism are as follows:

1. **Forgetting Mechanism:** When a new input comes, it needs to decide which prior information is important and to be kept and which information can be forgotten or thrown away.
2. **Saving Mechanism:** This mechanism decides which information from the new input is important and is worth saving. Therefore first the LSTM mechanism throws away any long term information that is no longer required and then saves the important information from input.

3. **Extracting information from long term memory:** The model after the forgetting and saving mechanism identifies which information from the long-term memory is of immediate importance conditioned on the current input.

Figure 3.4 explains the above mentioned mechanism in a modular structure

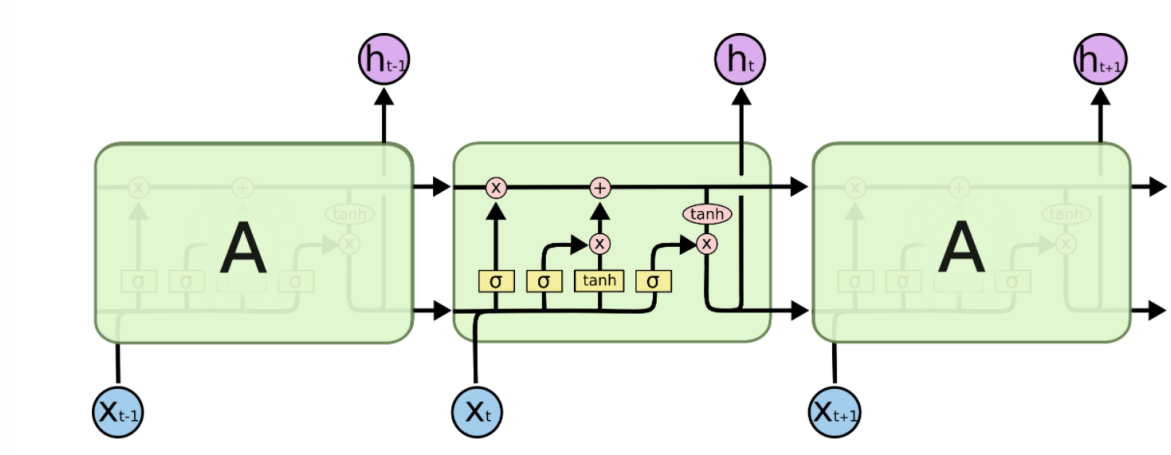

Figure 3.4: LSTM cell

3.2.2 Skip Thought Vectors

Skip-thought vector [3] is a unsupervised generic sentence encoder approach. It consists of a encoder-decoder model which generates the surrounding sentences based on a sentence as shown in Figure 3.5. The data is passed through a series of network, encoder, which extracts important information into an array smaller than input. This array is again passed to a series of network, decoder, which decodes and returns the desired output. This formula is simple, but effective. The basis of the algorithm is that the neighboring sentences share properties and are close in a N-dimensional space. It uses an Recurrent Neural network (RNN) with Gated Recurrent Unit (GRU) activations at encoder and decoder. But any recurrent model which can be backpropagrated can be used here.

Encoder: Let $w_i^1, w_i^2, ..., w_i^N$ represent the words in a i^{th} sentence, N being the length of the sentence. $\mathbf{h}_i^t$ represent the hidden state of the encoder of sentence i at an instant of $\mathbf{t}^{th}$ word being input. $\mathbf{h}_i^N$ represent the skip-thought vector of the whole sentence. The following equations explain the encoding way.

$$r^t = \sigma(W_r x^t + U_r h^{t-1}) \tag{3.8}$$

$$z^t = \sigma(W_z x^t + U_z h^{t-1}) \tag{3.9}$$

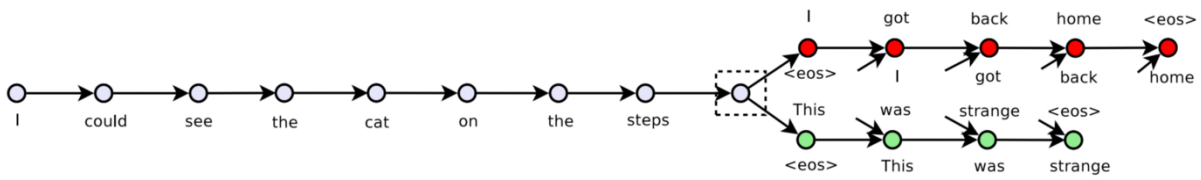

Figure 3.5: Skip-thought model. Showing the input and neighboring sentences [3].

$$\bar{h}^t = \tanh(Wx^t + U(r^t \odot h^{t-1}) \tag{3.10}$$

$$h^t = (1 - z^t) \odot h^{t-1} + z^t \odot \bar{h}^t \tag{3.11}$$

$\bar{h}^t \rightarrow$ state update at time t
$z^t \rightarrow$ update gate
$r^t \rightarrow$ reset gate

Decoder: The decoder model conditions on the hidden state **h** proposed by the encoder. $\mathbf{C}_r$, $\mathbf{C}_z$, **C** are used to bias the reset gate, update gate, and hidden state. Two decoders are used with separate parameters except the Vocabulary consisting of the word mapping with vectors. Two decoders are used for the previous and next sentence. Decoding uses the following equations

$$r^t = \sigma(W_r^d x^{t-1} + U_r^d h^{t-1} + C_r h_i) \tag{3.12}$$

$$z^t = \sigma(W_z^d x^{t-1} + U_z^d h^{t-1} + C_z h_i) \tag{3.13}$$

$$\bar{h}^t = \tanh(W^d x^{t-1} + U^d(r^t \odot h^{t-1} + C h_i) \tag{3.14}$$

$$h_{i+1}^t = (1 - z^t) \odot h^{t-1} + z^t \odot \bar{h}^t \tag{3.15}$$

Given previous t - 1 words, the probability of word w_{i+1}^t is:

$$P(w_{i+1}^t | w_{i+1}^{<t}, h_i) \propto exp(v_{w_{i+1}^t} h_{i+1}^t) \tag{3.16}$$

$h_{i+1}^t \rightarrow$ decoder hidden state at time t
$v_{w_{i+1}^t} \rightarrow$ vector corresponding to word w_{i+1}^t Similar computation is performed for the s_{i-1} sentence.

Using these equations our objective is to optimize the sum of forward and backward sentence log-probabilities:

$$\sum_t logP(w_{i+1}^t | w_{i+1}^{<t}, h_i) + \sum_t logP(w_{i-1}^t | w_{i-1}^{<t}, h_i) \tag{3.17}$$

3.3 Attention Mechanism

Attention Mechanisms[18] have been originated from humans where we tend to focus on a certain region of a bigger information to derive results, then change the focus point over time. Taking motivation from here, the attention mechanism in neural networks is developed to attend to important set of information from total. This has also helped in visualizing what the network is learning. It has shown state of the art results in speech translation, automatic speech recognition (ASR), reasoning and image captioning.

A attention model takes **n** inputs $y_1, y_2, ..., y_n$ along with a context and returns a vector **z** which is weighted sum of y_i. By focusing on the contextual information is picks and give more weight to specific $y_i's$. These weights are easily accessible and therefore are used to identify the regions of focus. Figure 3.6 shows the model of the attention mechanism.

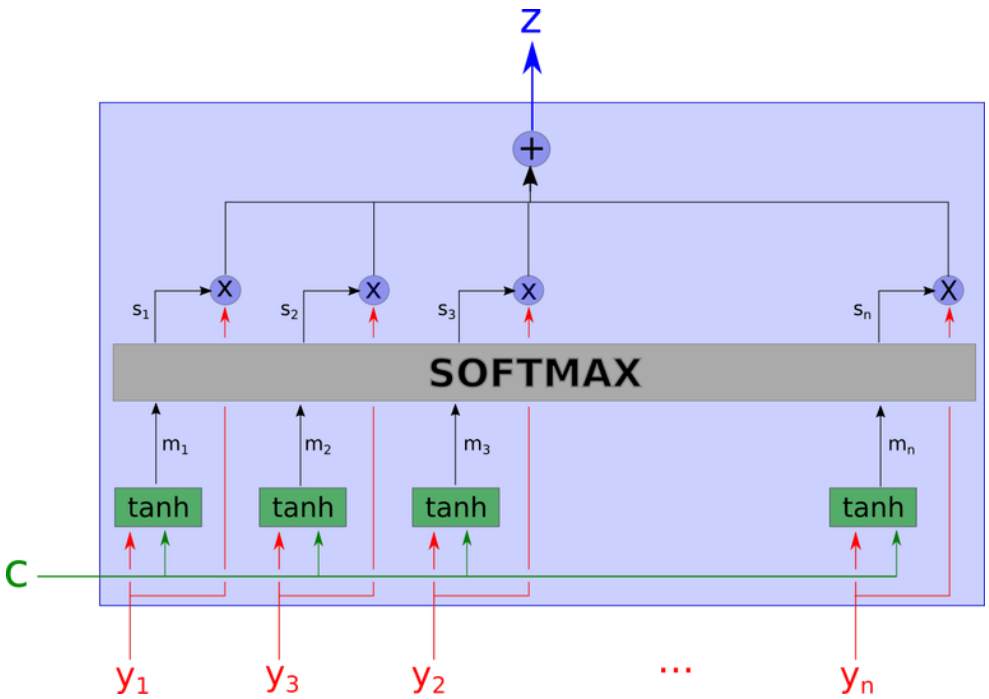

Figure 3.6: Attention Model

To begin with, we find a similarity or dissimilarity (depending on the use case) between the input $y_i's$ and context vector. An important thing to notice is that the $m_i's$ are calculated without looking at other $y_i's$. The $m_i's$ are then passed to a softmax layer to normalize them all. The output $\mathbf{z}$ is the weighted sum of s_i and y_i.

In [18] the author applied attention mechanism to generate text description from images. They used Convolutional Neural Networks and Long Short Term Memory Networks with Attention mechanism. In [34] authors used this mechanism in passage question answering system. [35] used the approach to translate a text written in English to French.

CHAPTER 4

APPROACH

4.1 Model Architectures

In this section, we briefly describe the architectures followed in various of our experiments.

4.1.1 Vanilla and Wasserstein Conditional GANs

Vanilla Conditional GAN was the first architecture for Text to Image synthesis using GANs followed by [4]. The architecture is shown in figure 4.1.

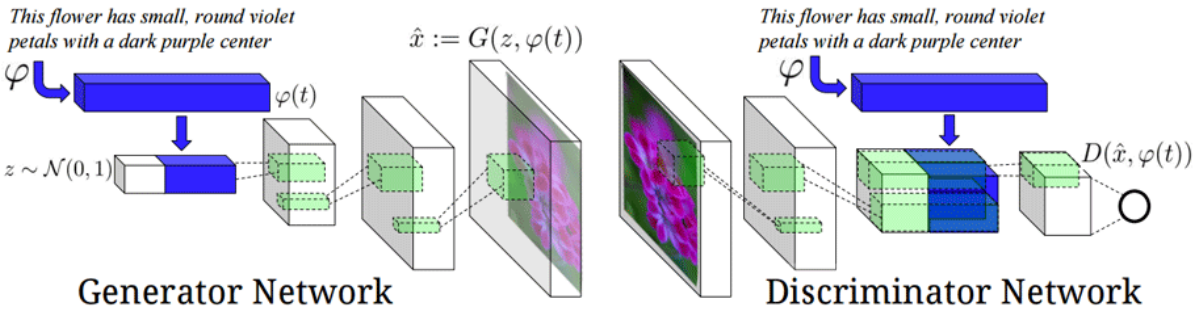

Figure 4.1: Text Conditional convolutional GAN architecture followed in [4]

The authors use a DCGAN architecture which is conditioned by text features. The text features are generated by encoding the text using a character level recurrent neural

network. The generator and discriminator architecture used in Vanilla Conditional GANs is explained below in 4.1.1.

1. Generator Network

    a) In the generator network we sample a Z dimensional noise(z) from a normal distribution.

    b) The caption is encoded with using a pretrained skip-thought model. The encoded caption is mapped to vector of smaller size and concatenated with the z.

    c) The concatenated embedding then goes through a feed forward deep deconvolutional network to obtain a synthetic image $\hat{x}$.

2. Discriminator Network

    a) The discriminator is a standard convolutional network which takes as input an image with 3 channels.

    b) The discriminator applies layer wise strided convolutions with batch normalization

    c) When the spatial dimension of the discriminator is 4×4, the skip-thought text embedding (after being reduced to a smaller dimension) is concatenated to discriminator layer.

    d) We then perform a 1×1 convolution, rectification, 4×4 convolution in the sequential order to compute the final score from D.

The architecture explain in Wasserstein GAN is same as the Vanilla Conditional GAN architecture mentioned in 4.1.1. The difference is in the loss functions of the generator and discriminator and the training process.

4.1.2 StackGAN

The images generated in [4] were of size 64×64 and were also not of optimum quality. The process used in StackGAN [5] is a two stage process to generate images of size 64×64 in stage 1 and 256×256 in stage 2. Also, the images generated are plausible enough to be treated as real images. The stage-1 GAN draws the primitive layout of image and shape of the object along with the colors, yielding a low resolution image. The stage-2 GAN conditions on the stage-1 output as well as the text embedding and complete the fine details

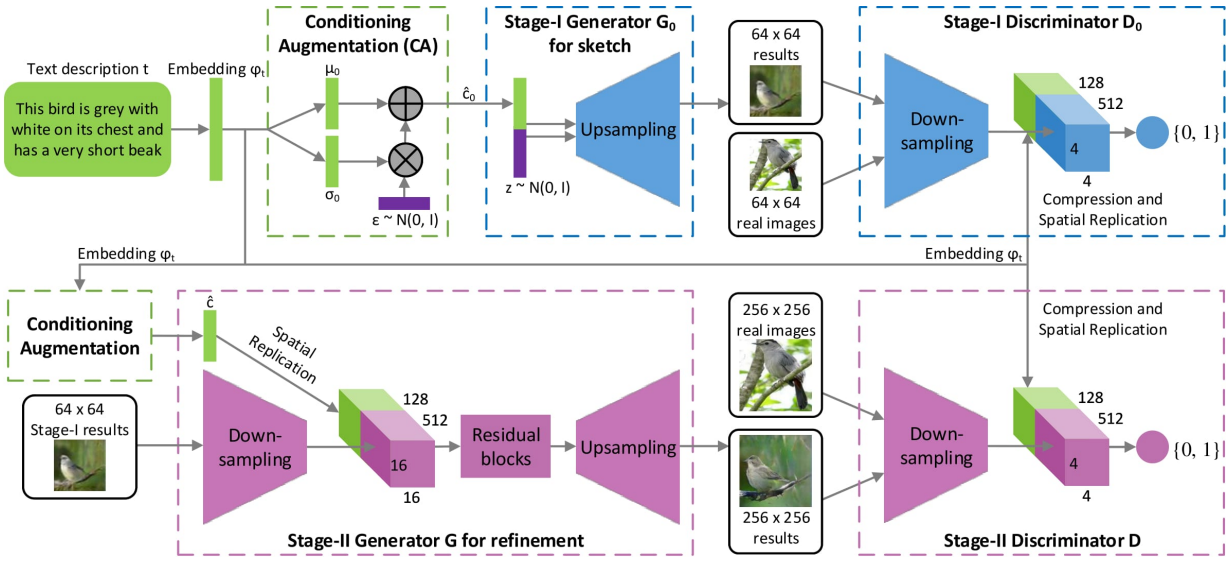

Figure 4.2: StackGAN Architecture [5]

and imparts minute features, giving a high resolution output. The model architecture is depicted in figure 4.2

In StackGAN, the conditioning variable $\hat{c}_0$ is sampled from the gaussian distribution $\mathcal{N}(\mu_0(\phi_t), \Sigma_0\phi_t)$ where ϕ_t is the text embeddings. This is called Conditioning Augmentation. The different stages of StackGAN is explained below:

1. Stage-1 GAN

   a) The text embedding is fed through a fully connected layer to get a Gaussian distribution from which the conditioning variable $\hat{c}_0$ is sampled.

   b) Then $\hat{c}_0$ is concatenated with the noise vector to generate a $W_0 \times H_0$ image by passing through a series of upsampling blocks.

   c) For the discriminator D_0, the text embedding is converted to N_d dimensions by passing it to a fully connected layer and then replicated spatially to form a tensor of dimension $M_d \times M_d \times N_d$. The real and fake images are then downsampled to $M_d \times M_d$ and then concatenated to the text tensor along the channel dimension.

   d) A single node fully connected layer produces the final decision score.

   e) This stage maximize the discriminator loss and minimize the generator loss as mentioned in equation 4.1 and 4.2.

$$L_{D_0} = \mathbb{E}_{(I_0,t)\sim p_{data}}[\log D_0(I_0, \phi_t)] + \mathbb{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_0(z, \hat{c}_0), \phi_t))] \quad (4.1)$$

$$L_{G_0} = \mathbb{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_0(z, c_0), \phi_t))] + \lambda D_{KL}(\mathcal{N}(\mu_0(\phi_t), \Sigma_0(\phi_t)||\mathcal{N}(0, I)) \tag{4.2}$$

2. Stage-2 GAN

   a) The result of Stage-1 is downsampled to $M_g \times M_g$ and concatenated along with $\hat{c}_0$.

   b) The tensor obtained is passed through residual blocks which learn multi-modal representation of text and images. Then upsampling blocks are used to generate a W × H image.

   c) The stage-2 discriminator is similar to stage-1 discriminator only with more downsampling blocks as the image size is larger now.

   d) The discriminator is also trained to learn the authenticity of image-text pair as described in [4]. The positive sample pairs consist of real images and corresponding text while negative samples consists of two groups. Synthesized images with real captions and real images with wrong captions.

   e) This stage maximize the discriminator loss and minimize the generator loss as mentioned in equation 4.3 and 4.4.

$$L_D = \mathbb{E}_{(I,t)\sim p_{data}}[\log D(I, \phi_t)] + \mathbb{E}_{s_0\sim pG_0, t\sim p_{data}}[\log(1 - D(G(s_0, \hat{c}), \phi_t))] \tag{4.3}$$

$$L_G = \mathbb{E}_{s_0\sim pG_0, t\sim p_{data}}[\log(1 - D(G(s_0, \hat{c}), \phi_t))] + \lambda D_{KL}(\mathcal{N}(\mu(\phi_t), \Sigma(\phi_t)||\mathcal{N}(0, I)) \tag{4.4}$$

Built upon the success of StackGAN, the authors proposed a modified version of StackGAN called StackGAN++[6]. In this Stackgan-V2, multiple generators and discriminators are arranged in a tree like structure and each branch of the tree generates an image from low-resolution to high-resolution. The discriminator analyzes whether an image is coming from the true data or from the generator. The generators are jointly trained to hierarchically generate images from random noise to high resolution. The architecture for StackGAN++ is shown in figure 4.3

4.1.3 Attention GANs

The images generated from StackGANs[5] were not able to capture small details in the later part of the generator. They used a common way to generate embedding of the whole

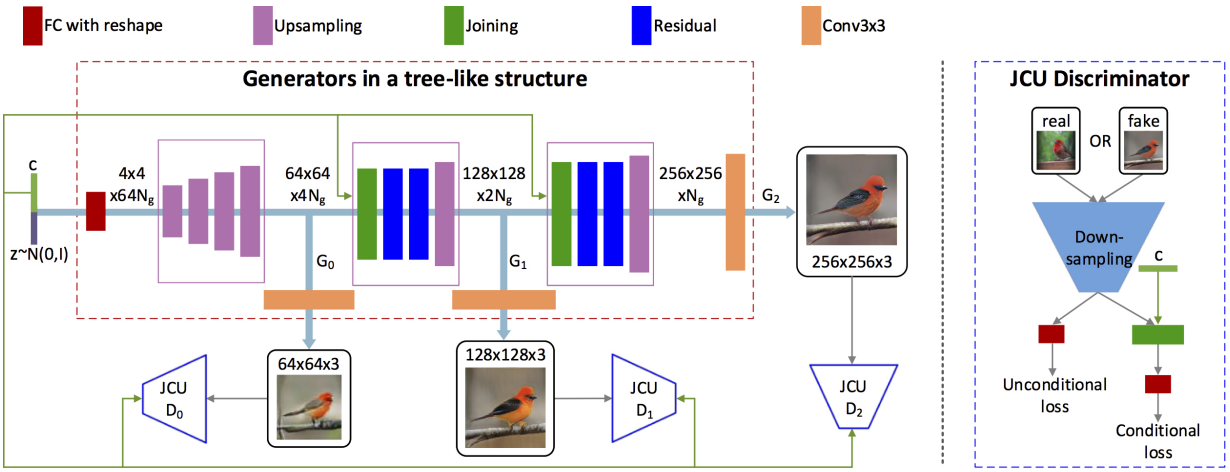

Figure 4.3: StackGAN++ framework showing a tree like structure [6].

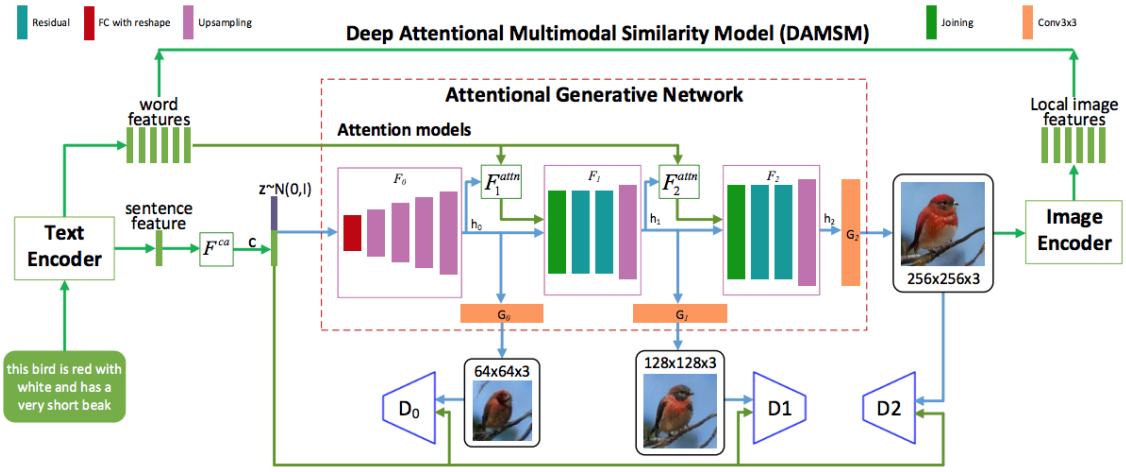

Figure 4.4: AttnGAN Architecture [7]

sentence into a global vector but this global vector miss out the fine-grain details at word level. Therefore [7] propose an multi-stage attention driven architecture for text to image generation. Architecture of the GAN is described in Figure. 4.4

It consists of two components. First, an attention generative network which generates different subregions of the image by focusing on the text embeddings of each word which is most relevant to them. The other component, Deep Attentional Multimodal Similarity Model(DAMSM), computes the losses between the generated image-sentence and generated image-words. This provides additional loss useful for generator training. Below we explain the the two components used in Attention GANs:

1. Deep Attentional Multimodal Similarity Model

   a) **Text Encoder** uses bi-directional Long-Short Term Memory networks to extract feature vectors from text descriptions. $e \in \mathbb{R}^{D \times T}$ represent the feature matrix of words where T and D represent the number of words and dimensions respectively. The Global sentence vector is the last state of bidirectional LSTMs, $\bar{e} \in \mathbb{R}^{D}$.

   b) **Image Encoder** uses part of Inception-v3 trained on ImageNet. First, we rescale the images to 299x299 pixels and then extract the outputs from the "mixed 6e" layer $f \in \mathbb{R}^{768 \times 289}$. The global feature vector $\bar{f} \in \mathbb{R}^{2048}$ is taken from the last pooling layer. A Dense layer is added to convert the feature maps extracted to the same vector space as text embeddings.

   c) **DAMSM loss** is calculated to find the matching between the whole images and entire sentences.

2. Attention Generative Network

   a) The model can have m generator-discriminator pair $(G_0D_0, G_1D_1, ..., G_{m-1}D_{m-1})$ as shown in the Figure 4.4. The i_{th} generator takes hidden inputs h_i as input and produces $\hat{x}_i$ image.

   $$\hat{x}_i = G_i(h_i) \tag{4.5}$$

   b) At the beginning, noise concatenated to the global conditional augmented vector is passed to couple Upsampling layers to generate hidden input.

   $$h_0 = F_0(z, F^{ca}(\bar{e})); \tag{4.6}$$

   $$h_i = F_i(h_{i-1}, F_i^{attn}(e, h_{i-1})) \text{ for } i = 1, 2, ..., m-1; \tag{4.7}$$

   c) The hidden input is passed to the attention mechanism F_i^{attn} along with the word features. The word-context vector, representation of words relevant to h, is calculated as follows.

   $$c_j = \sum_{i=0}^{T-1} = \beta_{j,i} e'_i \text{ where } \beta_{j,i} = \frac{exp(s'_{j,i})}{\sum_{k=0}^{T-1} exp(s'_{j,k})} \tag{4.8}$$

   d) This word context vector along with the hidden output is used to generate images for the next stage.

   e) The discriminators are defined at each node to classify the if the inputs are real or fake along with the authenticity of image-text pair.

4.2 Training the Model

We need to make the generator learn realistic images which correspond to the captions. A correctly trained discriminator should give a positive score for a real image with a corresponding caption and a negative score for synthetic images or real images which do not match the caption.

4.2.1 Vanilla Conditional GAN

This approach targets two sources of error, unrealistic images for any text and Realistic images which do not correspond to the conditioning text embedding. Therefore, by learning to optimize image / text matching in addition to the image realism, the discriminator can provide an additional signal to the generator. Algorithm 1 is used to implement the above mentioned framework using mini-batch training.

Algorithm 1 Training Algorithm

$x \leftarrow$ Mini batch of images
$t \leftarrow$ Matching Text
$\hat{t} \leftarrow$ Mis-matching text
for n = 1 to S **do**
    $h \leftarrow$ Matching text thought vector
    $\hat{h} \leftarrow$ Mis-Matching text thought vector
    $z \leftarrow$ Z dimensional random noise
    $\hat{x} \leftarrow$ G(z,h) generator synthesized image
    $s_r \leftarrow$ D(x, h) real image, right text
    $s_w \leftarrow$ D(x, $\hat{\text{h}}$) real image, wrong text
    $s_f \leftarrow$ D($\hat{\text{x}}$, h) fake image, right text
    $L_d \leftarrow log(s_r) + (log(1 - s_w) - log(1 - s_f))/2$
    $D \leftarrow D - \alpha * \partial L_d / \partial D$
    $L_g \leftarrow log(s_f)$
    $G \leftarrow G - \alpha * \partial L_g / \partial G$: update G twice
end for

4.2.2 Wasserstein Conditional GAN

The modified implementation of Vanilla Condition GAN used WGAN as the image synthesizing model, objective function for which is described in 3.1.3. In WGAN, the discriminator is trained much more number of times than generator to give some meaningful initial gradient to the generator at the start of the training. The Algorithm 2 described below is used implemented WGAN.

Algorithm 2 WGAN Training Algorithm

$G\_iteration \leftarrow 0$
for n = 1 to Epoch **do**
    $Steps \leftarrow$ number of mini batches of images
    for i = 1 to Steps **do**
        for p = 1 to Diter **do**
            $x \leftarrow$ Next Mini batch of images
            $t \leftarrow$ Matching Text
            $h \leftarrow$ Matching text thought vector
            $z \leftarrow$ Z dimensional random noise
            $\hat{x} \leftarrow$ G(z,h) generator synthesized image
            $s_r \leftarrow$ D(x, h) real image, right text
            $s_f \leftarrow$ D(x̂, h) fake image, right text
            $L_d \leftarrow -log(s_r) - log(1 - s_f))/2$
            $D \leftarrow D - \alpha * \partial L_d / \partial D$
            $p \leftarrow p + 1$
        end for
        If G_iter < 25 or G_iter % 500 == 0
        Diter ← 100
        Else Diter ← 5
        $L_g \leftarrow -log(s_f)$
        $G \leftarrow G - \alpha * \partial L_g / \partial G$: update G
        $G\_iter \leftarrow$ G_iter + 1
    end for
end for

4.2.3 Attention GANs

1. **Deep Attentional Multimodal Similarity Model (DAMSM):** Attention GANs consists of two different independent neural networks that needs to be trained. DAMSM model scales down the image and text in the same dimension to find the correlation loss between them. This loss is then used for training the final discriminator. Algorithm 3 explains the way the DAMSM loss can be achieved with mini batch training.

Algorithm 3 DAMSM Training Algorithm

Steps ← Mini batch of images
for n = 1 to Steps **do**
  $v \leftarrow$ Visual feature vector
  $\bar{v} \leftarrow$ Global feature vector of the image
  $e \leftarrow$ feature matrix of words
  $\bar{e} \leftarrow$ Global feature vector of text
  $s \leftarrow e^T v$, similarity matrix of words and sub-regions of images
  $\bar{s}_{i,j} \leftarrow \frac{exp(s_{i,j})}{\sum_{k=0}^{T-1} exp(s_{k,j})}$ Normalised similarity matrix
  $c_i \leftarrow \sum_{j=0}^{288} \alpha_j v_j$ where $\alpha_j = \frac{exp(\gamma_1 \bar{s}_{i,j})}{\sum_{k=0}^{288} exp(\gamma_1 \bar{s}_{i,j})}$ c_i represents subregion for the i^{th} word
  $RQ,D \leftarrow \log(\sum_{i=1}^{T-1} exp(\gamma_2 R(c_i e_i))^{\gamma_2^-1}$,attention-driven image-text matching score
  $P(D_i|Q_i) \leftarrow \frac{exp(\gamma_3 R(Q_i|D_i))}{\sum_{j=1}^{M} exp(\gamma_3 R(Q_i|D_j))}$, probability of sentence D_i matching the image Q_i
  $L_1^w \leftarrow -\sum_{i=1}^{M} \log P(D_i|Q_i)$, word loss 1
  $L_2^w \leftarrow -\sum_{i=1}^{M} \log P(Q_i|D_i)$, word loss 2
  L_1^s, $L_2^s \leftarrow$, can be calculated using $\bar{e}$, $\bar{v}$
  $L_{DAMSM} \leftarrow L_1^w + L_2^w + L_1^s + L_2^s$, backpropagate on the DAMSM loss to update weights
end for

2. **Attention Generative Network:** The second model, Attention Generative Netword, is a extension of StackGAN++[6] in which instead of using the global vector of sentence after every stage, it computes the weighted sum of word vectors using attention mechanism to let the model decide for itself on which detail to focus. Below Algorithm 4 explains the steps.

Algorithm 4 Attention GAN Training Algorithm

Steps $\leftarrow$ Mini batch of images
for n = 1 to Steps **do**
  $x \leftarrow$ Real Images
  $\bar{e} \leftarrow$ Matching sentence encoded text
  $\bar{\hat{e}} \leftarrow$ Mis-Matched sentence encoded text
  $z \leftarrow$ Z dimensional random noise
  for i = 1 to NumOfBranches **do**
    $\hat{x}_i \leftarrow G_i(z, \bar{e})$, Generator synthesized image
    $s_r \leftarrow D_i(x_i, \bar{e})$ real image, right text
    $s_w \leftarrow D_i(x_i, \bar{\hat{e}})$ real image, wrong text
    $s_f \leftarrow D_i(\hat{x}_i, \bar{e})$ fake image, right text
    $us_r \leftarrow D_i(x_i)$ real image
    $us_f \leftarrow D_i(\hat{x}_i)$ fake image
    IF i < NumOfBranches:
    $L_{di} \leftarrow log(s_r) + (log(1 - s_w) + log(1 - s_f))/2 + log(1 - us_f) + log(us_r)$
    ELSE:
    $L_{di} \leftarrow log(s_r) + (log(1 - s_w) + log(1 - s_f))/2 + log(1 - us_f) + log(us_r) + \lambda L_{DAMSM}$
    $D_i \leftarrow D_i - \alpha * \partial L_{di}/\partial D_i : UpdateD_i$
    $L_{gi} \leftarrow log(s_f) + log(us_f)$
    $G_i \leftarrow G_i - \alpha * \partial L_{gi}/\partial G_i : UpdateG_i$
  end for
end for

CHAPTER 5

EXPERIMENTAL DETAILS

5.1 Datasets

We used the Caltech-UCSD Birds(CUB) dataset[8] and Oxford-102[9] of flowers for our experiments. CUB dataset contains 11,788 birds images of 200 categories. 80% of the images have an object to image size ratio less than 0.5. Therefore the first step in the experiments was to crop the images to have more than 75% of object inside the image. Oxford-102 dataset contains 8,189 images from 102-different flower categories common in United Kingdom. Table 5.1 and 5.2 shows the split in the datasets.

Table 5.1: CUB dataset[8].

CUB	Train	Test
#samples	8,855	2,933
caption/images	10	10

Table 5.2: Oxford-102 dataset[9].

Oxford-102	Train	Test
#classes	82	20
#samples	6,142	2,047
caption/images	5	5

5.2 Vanilla Conditional GANs

We trained the model on The Oxford-102 flowers dataset [9]. We take a maximum of 5 captions per image from the dataset. For embedding, the text we use the pretrained skip-thought model and use the 2048 dimensional uni-skip encodings for the captions. The implementation was done using Keras framework with hyperparameter details mentioned below.

1. We used uni-skip vectors from the skip thought vectors. We have not tried training the model with combine-skip vectors.
2. The images generated are 64×64 in dimension.
3. The model was trained for 600 epochs with a batch size of 64 on a GPU with a learning rate $\alpha = 0.0002$ using Adam Optimizer.
4. While processing the batches before training, we flipped the images horizontally with a probability of 0.5.
5. The training-validation split was set to be 0.75.
6. During a single-mini batch iteration the weights of the generator are updated twice to prevent the discriminator loss going down to 0.

5.3 Wasserstein GANs

We trained the model on birds dataset of CUB200-2011[8] dataset. The text encoder was pre-computed using a character level hybrid CNN-RNN model on structured joint embeddings of captions with 1,024 dimensional GoogLeNet image embeddings [36]. The hybrid CNN-RNN model is described in [37]. Other implementation details are mentioned below.

1. The text encoder used a CNN input size (sequence length) of 201 and a text embedding layer of 1024 dimensions.
2. The model was trained for 600 epochs. The discriminator was trained 100 times for a single update of generator in the initial phase to provide substantial gradients to the generator and trigger the learning.

3. Images generated are of size 64×64.

4. Batch size used was 64 using a learning rate of 0.0002. Adam Optimizer was used.

5.4 Attention GANs

We used the CUB200-2011[8] dataset to evaluate the results. The text embeddings of 256 dimensions were calculated using the RNN_ENCODER along with the DAMSM and Attention Generative Network. Tuned hyperparameters are described below.

1. The model generates three images of 64×64, 128×128, 256×256 dimensions. It also generates two attention maps applied on 64×64, 128×128 dimension images.

2. The DAMSM model is trained for 600 epochs on GeForce GTX 1080Ti GPU with the learning rate of 0.002 with $\gamma_1, \gamma_2, \gamma_3$ equals to 4.0, 5.0, 10.0 respectively and Adam optimizer($\beta_1 = 0.5, \beta_2 = 0.999$)

3. The Attention Generative model is also trained for 600 epochs GeForce GTX 1080Ti GPU with the discriminator and generator learning rate of 0.002 and $\gamma_1, \gamma_2, \gamma_3, \lambda$ equals to 4.0, 5.0, 10.0, 5.0 respectively.

4. Maximum number of words are restricted to 18 in the captions.

5. Generator and Discriminators are trained on both conditional and unconditional losses with a batch size of 20 images with Adam Optimizer ($\beta_1 = 0.5, \beta_2 = 0.999$).

CHAPTER 6

RESULTS

6.1 Vanilla Conditional GAN

Figure 6.1 shows the 64×64 images generated with the Vanilla Conditional GAN along with the text description. Most of the flower images are realistic enough. However, the images are of suboptimal quality and the method also suffers from mode-collapse as many of the generated flowers have very similar features and do not show much variety.

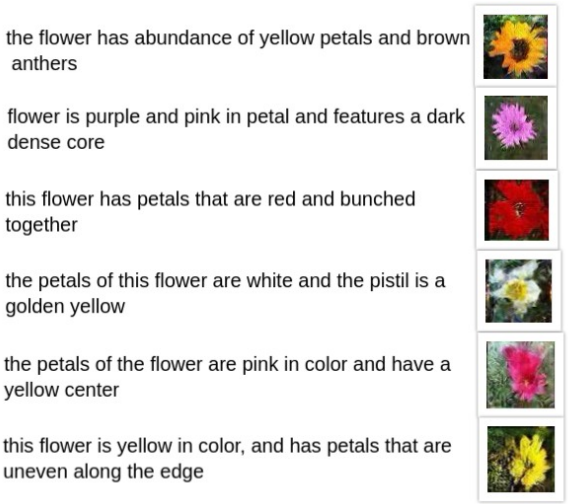

Figure 6.1: Text descriptions and the image generated with Vanilla Conditional GAN.

Figure 6.2 shows some more images generated from the Vanilla Conditional GAN. The text description are randomly chosen strings from the 1000 samples kept out for validation. However, as seen from the figure, similar flowers are generated multiple times, this is because of the text description being very generic which can be fit into a lot of variety of flowers, a single global sentence can't capture all the information of a sentence and mode collapse. These drawbacks have been rectified in the future Attention GAN mechanism.

Figure 6.2: 64×64 **images generated with Vanilla Conditional GAN.**

6.2 Wasserstein GAN

The training loss for discriminator and generator of WGAN is plotted in figure 6.3 and 6.4.

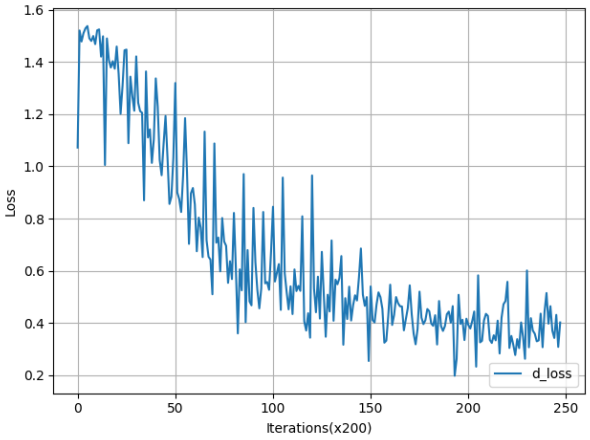

Figure 6.3: Wasserstein loss of WGAN

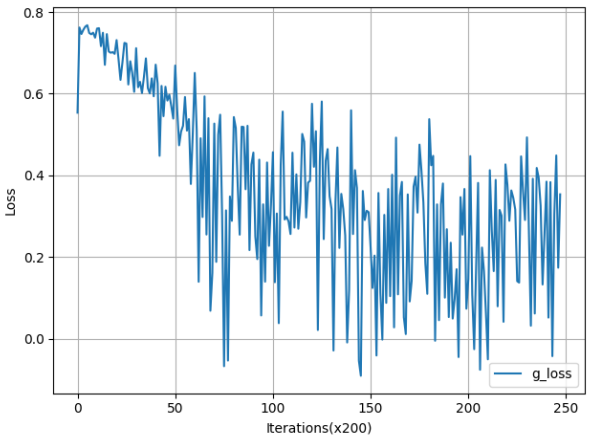

Figure 6.4: Generator loss of WGAN

Wasserstein GAN try to distinguish between real Image-real Caption, fake Image-real Caption and wrong Image-real Caption. The Wasserstein loss Figure 6.3 seems to decrease with the improvements in image quality. This is different as compared to the Vanilla GAN losses because here the loss($K * W(P_r, P_\theta)$) represents the closeness of the two distributions. Therefore, wasserstein loss is different and should not be confused with discriminator loss as discriminator loss would never be zero even for an ideal case when the distributions are same.

The instability of generator loss and the decreasing discriminator loss explains that the generator generates an improved image but discriminator is also trained to find flaws in that improved image. Therefore the generator has to improve the image even further.
Some results shown in the Figure 6.5 and 6.6 are able to generate the shape and colors of the birds. Even they lack minute details like legs, beak and other authentic details because of which they can't be called real. This is due of the partial training of the model because of limitation in the computation power. As we can see from the Figure. 6.3 the discriminator has not yet saturated and could be further trained to improve results.

Figure 6.5 shows the text description along with the image generated through conditional WGAN algorithm and figure 6.6 shows some more images of birds generated.

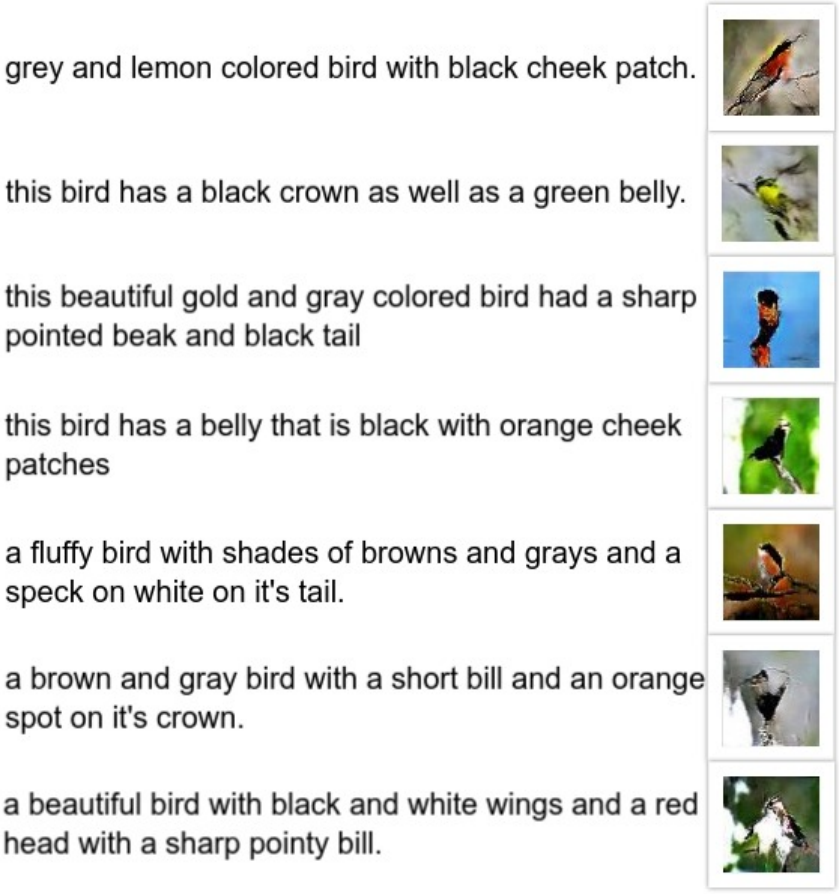

Figure 6.5: Text descriptions and the image generated with WGAN.

Figure 6.6: 64×64 **images generated from WGAN.**

6.3 Attention GAN

The Generator and Discriminator losses in Figure 6.7 and Figure 6.8 don't converge because with every iteration it is changing. At the start, the generator can be considered as a child and discriminator as a teacher. The child works hard and gets into the next standard but then the teacher will also respond by making the test/curricular harder. The generator has become smart but the tests have also become harder. If the generator produces kindergarten level work, the teacher/discriminator will punish more than it did in kindergarten. Therefore if the child never improves, the scores will go down. The fact that scores is not going down implies that the generator is improving.

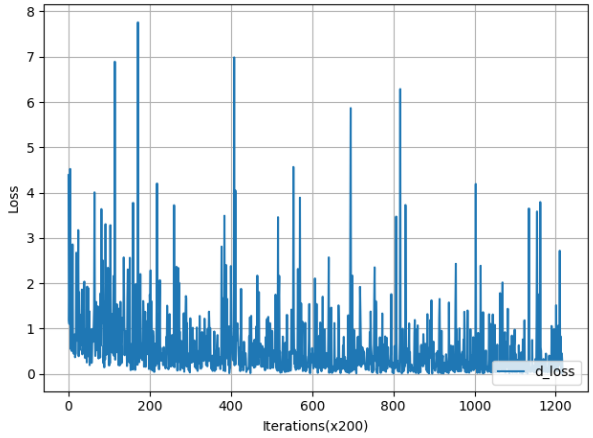

Figure 6.7: Discriminator loss of Attention GANs

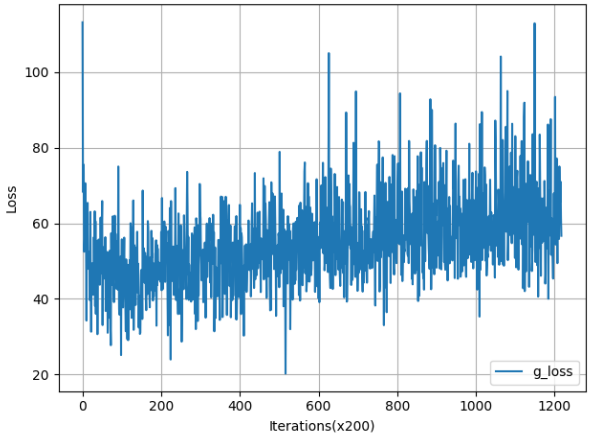

Figure 6.8: Generator loss of Attention GANs

As shown in the Figure 6.9, a low resolution image is generated first (by G_0) that shows only the shape and color of the object since global sentence vector was used. In the next stages (G_1,G_2), due to focus on words is able to rectify the drawbacks of the first image and generate a more detailed high resolution image. These figures shows the top-5 words taken by the algorithm along with the region to apply attention in the two stages. Figure 6.10 shows other examples of final stage AttnGAN. Attention Mechanism provide a dual benefit. First, it is able to extract the important words which it needs to focus on. Secondly, we are able to visualize and take insights about the position where each word is focusing. These images are able to capture and focus on minute details, like all the textures, of the bird along with the background. These results are definitely better than the current WGAN results.

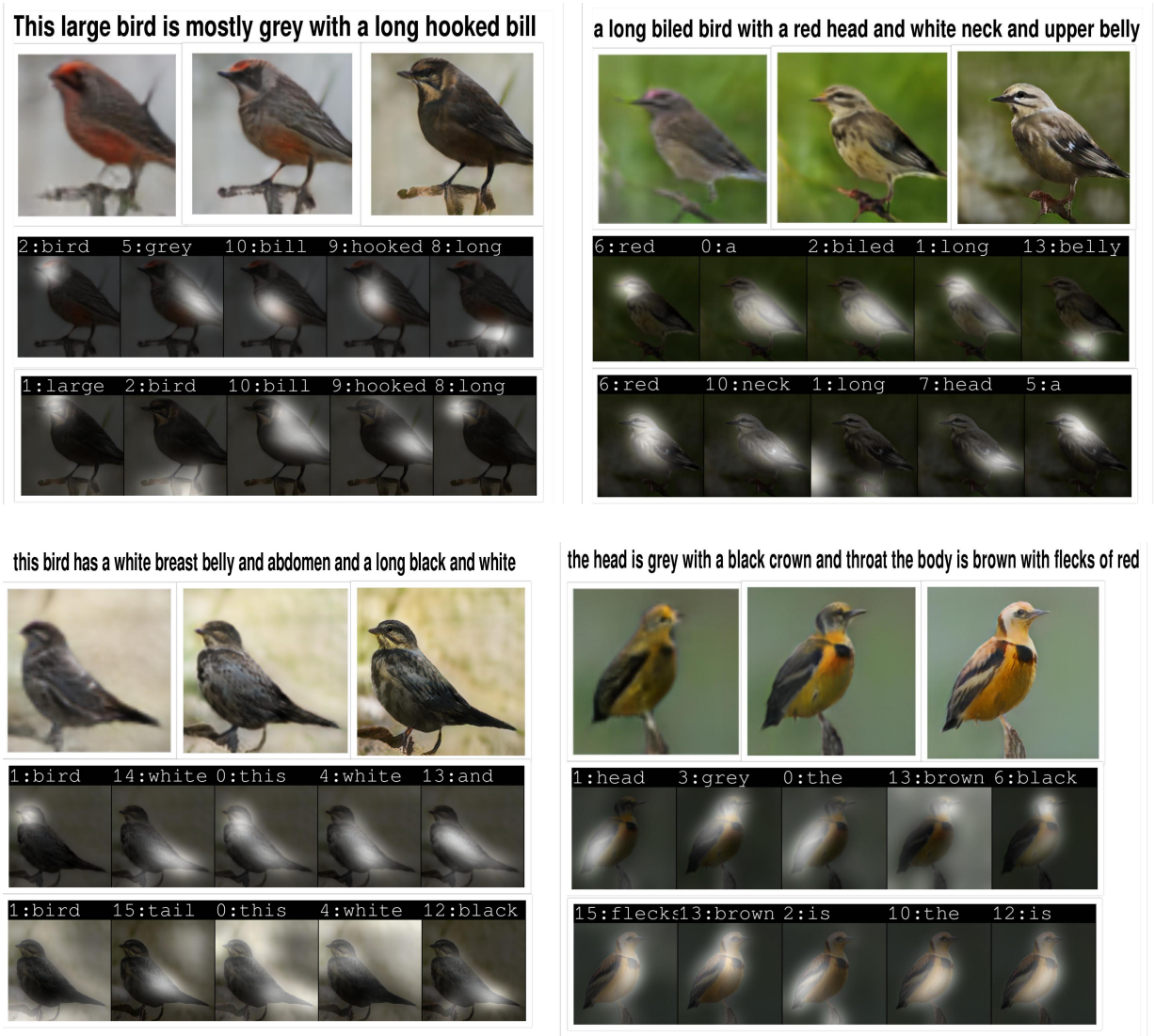

Figure 6.9: Images generated in multiple stages along with their attention maps. The attention maps placed above are of Stage 2 and below ones are from stage 3.

Figure 6.10: 256×256 **images generated using AttnGAN model trained on CUB.**

CHAPTER 7

FUTURE WORK

The current architecture has performed better than the vanilla architecture condition GANs. In this architecture we have explored the improvements in text embedding part. The way we could focus on specific words and then draw fine-details in the image. Below mention are some experiments that could be tried to improve the results.

- If we consider the way an artist draw an image, first a bounding box is created for an object, then the object inside a bounding box is given shape and then the whole image is generated. Therefore if we could break down our generation process into similar steps along with Attention Mechanism, better results could be hypothesized.
- Wasserstein GANs have become a trending word within the GAN community because for two low dimensional distributions, Wasserstein distance provide a smooth and meaningful representation of the distance in-between than Kullback-Leibler divergence or Jensen-Shannon divergence. But training WGAN takes more time and more computational power. Therefore, applying conditional WGAN with attention mechanism could produce better results.
- The results in this report are generated on CUB and Oxford-102 datasets. The same architecture with different hyper-parameters could be used to train on MS COCO dataset to produce generalized images.

CHAPTER 8

CONCLUSION

In our work, we implemented several techniques to successfully obtain a model for synthesizing images using text descriptions. Though each of the techniques has its own pros and cons, the final model Attention GAN performs the best among all as can be evident from the qualitative results shown. The generated images are of size 256×256 and of photo-realistic quality. This model is the state-of-the-art for synthesizing images based on textual description. We also implemented image-word loss, DAMSM, to be used while training of the model.

We also explored conditional Wasserstein GANs which can be used instead of normal GANs to generate images. The modified loss function in WGAN provides sufficient theoretical as well as practical proof for the success of WGANs. Although, the training process is slower as compared to Vanilla GAN model as the discriminator has to be trained many more number of times before the generator can effectively start learning, WGANs have proved to be resistant to the mode-collapse problem.

The current methods work well on datasets like CUB[8] and Oxford-102 flowers[9]. One limitation for the current methods is that the models only learn the distinguishable feature of a single object or features of the whole image but does not learn the concept of objects in it. Hence, besides improving the available models for text to image synthesis, one another possible area of research is to learn the concepts of individual objects in image for improving upon the problem of image generation through text.

BIBLIOGRAPHY

[1] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 664–676, Apr. 2017.

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

[3] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, (Cambridge, MA, USA), pp. 3294–3302, MIT Press, 2015.

[4] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1060–1069, JMLR.org, 2016.

[5] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *ICCV*, 2017.

[6] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan++: Realistic image synthesis with stacked generative adversarial networks," *arXiv: 1710.10916*, 2017.

[7] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," *CoRR*, vol. abs/1711.10485, 2017.

[8] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," tech. rep., 2011.

[9] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[10] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *ICML*, 2011.

[11] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *NIPS*, 2012.

[12] S. W. Sohn, K. and and H. Lee, "Improved multimodal deep learning with variation of information," in *NIPS*, 2014.

[13] A.Dosovitskiy, J.T.Springenberg, and T.Brox, "Learning to generate chairs with convolutional neural networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[15] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, June 2015.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, (Cham), pp. 740–755, Springer International Publishing, 2014.

[18] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[19] 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.

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016.

[21] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," in *ICLR*, 2017.

[22] J. J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," in *ICLR*, 2017.

[23] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," in *ICLR*, 2017.

[24] 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.

[25] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 105–114, IEEE Computer Society, 2017.

[26] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural photo editing with introspective adversarial networks," *CoRR*, vol. abs/1609.07093, 2016.

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

[28] E. Mansimov, E. Parisotto, L. J. Ba, and R. Salakhutdinov, "Generating images from captions with attention," in *ICLR*, 2016.

[29] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 4790–4798, Curran Associates, Inc., 2016.

[30] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics* (D. van Dyk and M. Welling, eds.), vol. 5 of *Proceedings of Machine Learning Research*, (Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA), pp. 448–455, PMLR, 16–18 Apr 2009.

[31] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[32] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 214–223, PMLR, 06–11 Aug 2017.

[33] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 3111–3119, Curran Associates, Inc., 2013.

[34] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 1693–1701, Curran Associates, Inc., 2015.

[35] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift.," in *ICML* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456, JMLR.org, 2015.

[37] S. E. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. de Freitas, "Parallel multiscale autoregressive density estimation," in *ICML*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 2912–2921, PMLR, 2017.