TEXT TO IMAGE SYNTHESIS USING GENERATIVE ADVERSARIAL NETWORKS

A thesis submitted

in partial fulfillment for the award of the degree of

Master of Technology

in

MACHINE LEARNING AND COMPUTING

by

SRIKANT NAYAK

Department of Mathematics

Indian Institute of Space Science and Technology

Thiruvananthapuram, India

MAY 2020

Certificate

This is to certify that the thesis titled ***TEXT TO IMAGE SYNTHESIS USING GENERATIVE ADVERSARIAL NETWORKS*** submitted by **SRIKANT NAYAK**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Master of Technology** in **MACHINE LEARNING AND COMPUTING** is a bonafide record of the original work carried out by him/her under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr S Sumitra
Supervisor
Associate Professor
Department of Mathematics

Dr.Deepak Mishra
Co-Supervisor
Associate Professor
Department of Avionics

Dr.Nicholas Sabu
Professor &
Head of Department
Department of Mathematics

Place: Thiruvananthapuram
Date: MAY 2020

Declaration

I declare that this thesis titled ***TEXT TO IMAGE SYNTHESIS USING GENERATIVE ADVERSARIAL NETWORKS*** submitted in partial fulfillment for the award of the degree of **Master of Technology** in **MACHINE LEARNING AND COMPUTING** is a record of the original work carried out by me under the supervision of **Dr S Sumitra** and **Dr Deepak Mishra** has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: MAY 2020

SRIKANT NAYAK
(SC18M005)

Abstract

The text to image synthesis has many practical applications like photo editing and computer-aided creation. Most of the existing text-to-image approaches fail to reflect the entire details of the given text description. Recently conditional generative models are used to generate images from text.

This thesis describes the application of conditional-GAN, that comprises of two stages, for text to image conversion. In stage-I of this process, image is generated according to the given text description (conditional image generation) using Wasserstein GAN. The Wasserstein distance which provides stability is used in this technique, as it provides a meaningful value when two distributions are disjoint compared to KL divergence and JS divergence. In stage-II, using the output of stage-I, high-resolution image (256*256) is generated using super-resolution GAN (SRGAN). This stage is also able to rectify defects found in stage-I results. To improve the diversity of the synthesized images and stabilize the training of the conditional-GAN, conditioning augmentation technique, that is, data augmentation technique conditioned on text description has been applied. Extensive experiments on benchmark datasets and comparison of those results with that of other state-of-the-art techniques demonstrate the ability of the proposed method for conditional image generation.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter contains a short introduction of text to image synthesis, Generative model, and an overview of the thesis.

1.1 Text to Image Synthesis

Image captioning is one of the most common and challenging problems in the world of computer vision and natural language processing field i.e given an image, a text description is generated. The reverse process is the text to image synthesis, which synthesizing high quality of the image from the given text description.No doubt this is useful and interesting, but current AI is far from this goal. In recent years, a powerful deep learning model like a generative adversarial network(GANs) has been generating promising results. The sample that generates from text to image synthesis gives the rough idea or meaning of the given text description, but it fails to generate the high quality of images and also contain unnecessary details and vivid object. From a high-level perspective text to image is not different from the language translation problem. [1]

In spite of this, these problems are entirely different from image-text, as text to the image are highly multimodel problems. If one tries to translate a sentence such as “My name is Srikant Nayak ” to Chinese, then there are not many sentences that could be a valid translation. If one could try to produce a sentimental image of this description, then there will be a large number of image will produce in that description. Though this multimodel behavior is also present in the image captioning problem. There the difficulty level is less than the text to image synthesis because the language is mostly sequential. This problem is exploited under the conditioning of the generation of the new words on the previous words. Because of this text to image is very difficult then image captioning. Generating an image from the text description has many applications in the future once the technology comes in

commercial purposes.

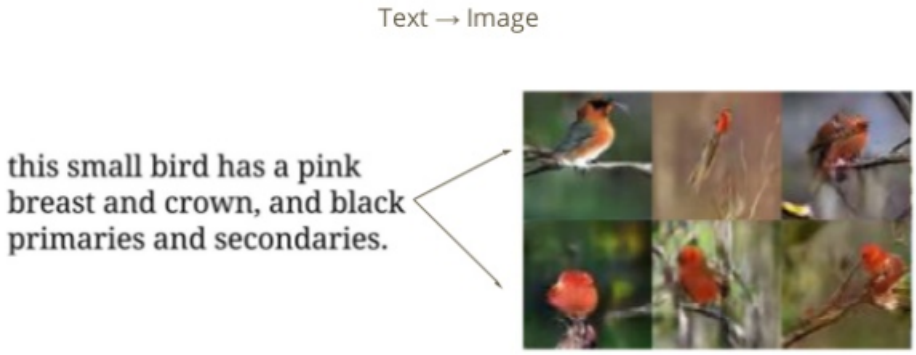


Figure 1.1: Text to image

1.2 Dataset

In this project, we are using the Caltech CUB-200 birds dataset and the OXFORD-102 flower dataset. These 2 datasets are the ones which usually used in the text to image synthesis. OXFORD-102 flower dataset contains 8192 flower images from 102 categories of the flower which have large scale and light variations. These datasets commonly occur in the united kingdom. There are some categories of the flower which have large variation, several similar categories and visualized with several shapes and color features.
CUB-200 birds dataset have 11,788 birds images of 200 different type of birds. These datasets include only photos but no description. For image description, we are using a publicly available caption that used in Amazon Mechanical Turk. In this description, each image contains 5 different types of descriptions having a length of 10 words of each sentence. They do not describe the backgrounds and also not mention the species of birds or flowers.It just gave the labels of images.[2]

As the number of images is less, we are applying data argumentation in each image by random cropping and right-left flipping so that we can generate more number of images with the same description. After data argumentation, we can split the data into training and testing such that they contain disjoint classes of image. The dataset is summarized in the below table

This pink flower has overlapping petals and yellow florets growing in the middle.

This bird is white with blue on its back and has a long, pointy beak.

Figure 1.2: Dataset for text to image synthesis

Dataset/Number of images	Train	Test	Total
Flowers	7,034	1,155	8,192
Augmented flowers (256x256)	675,264	110,880	786,432
Birds	8,855	2,933	11,788
Augmented birds (256x256)	850,080	281,568	1,131,648

Figure 1.3: Summary statistics of the datasets.

1.3 Generative Model

The term 'Generative' means to generate a new sample of data. A generative model describes, how a sample of data is generated with reference to the probabilistic model.

Suppose, We have a dataset of 1000 images of the horse. We try to build a model which generates new images of horse that have not exist in the real world. but still, it looks as real because the model has trained , that regulate the features of the horse. This kind of problem is solved through the generative model. A summary of the generative model is shown in the figure below,

The text to image perfectly fits the problem generative model attempt to solve. The generative model has various types such as variational autoencoder, generative adversarial network(GANs), and many more. The best generative model that is used in our project is the generative adversarial network(GANs). Before introducing the architecture of GANS, we will give some more information about the generative model in a few paragraphs.

Consider a dataset X = $x^{(1)}, x^{(2)}, ...x^{(n)}$ where n is the number of samples and $x^{(i)}$ is a vector which is an image that encoded as a vector of the pixel value. The dataset which is fed to the input that sampling the image compared to the unknown data distribution is

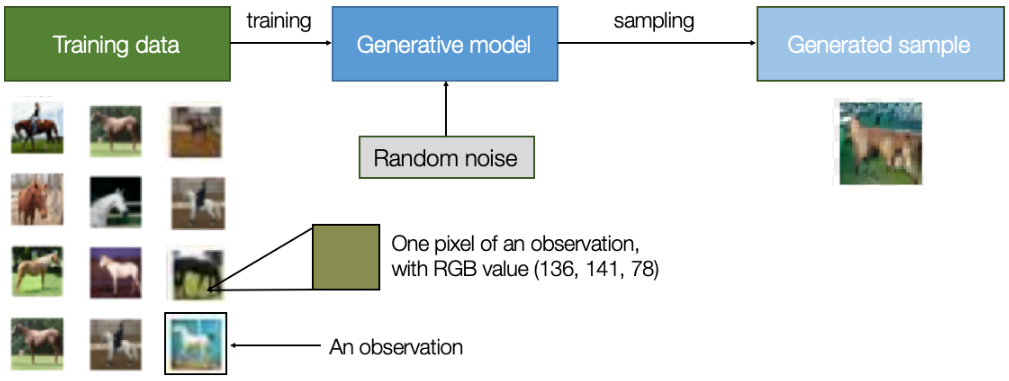


Figure 1.4: General architecture of generative model

P_r where r stands for the real. A generative model is a model that generates the estimated distribution of data P_g compare to the real distribution P_r. The model distribution P_g is the hypothesis about the real data distribution P_r.[3]

Most Generative model learn the model distribution P_g by maximizing the expected log likelihood $E_{x \sim P_r} log(P_g(x \mid \theta)$ where θ is the parameter of the model. The intuition behind the maximizing the log likelihood is equivalent to minimizing the kullback-leibler divergence $D_{KL}(P_r \parallel P_g) = \int_x P_r log \frac{P_r}{P_g} dx$ where $P_r$and $P_g$are data distribution. GANs is the another way of generating model that takes a different approach based on game theory.

1.4 Thesis Outline

The thesis is prepared in several chapters and the content of these chapters are given in brief as per the details given below.

1.4.0.1 Chapter-2

This chapter discusses some of the basic concepts which will be used through- out the chapter.

1.4.1 Chapter-3

This chapter discusses the recent works done in the field of text to image synthesis specifically in deep learning models. It also discusses how generative models like GANs can be used for text to image.

1.4.2 Chapter-4

In this chapter, we apply a new model of text to image synthesis that generate a low resolution images.

1.4.3 Chapter-5

After getting low resolution image , we try to convert it into high resolution image by using super resolution generating adversarial network.

1.4.4 Chapter-6

Here we concludes the thesis and discusses the future work that are to be done related to the thesis.

1.5 Contributions

The main contributions of the thesis are as follows:

1. To introduce a novel Text to image synthesis by using Wasserstein GAN. The method effectively reduces the mode collapse and also increases the quality of the image. To the best of our knowledge, this is the first approach which uses text to image synthesis.

2.Enhance the resolution by using Super resolution GAN

Chapter 2

Preliminaries

2.1 Kullback–Leibler Divergence(KL Divergence)

It is the measure of how one probability distribution is different from the second one .

Let P(x) and Q(x) be the two different probability distribution then the KL divergence of P(x) on Q(x) will be defined as:

$$D_{kl}(P \parallel Q) = \sum_{x} P(X = x) log \frac{P(X = x)}{Q(X = x)}$$

In other word it is the expectation of logarithmic differance between the two probability distribution P and Q , where expectation is taken from the probability P.

Suppose we have two multivarient normal distribution defined as :

$$P(x) = N(x; \mu_1, \Sigma_1)$$

$$Q(x) = N(x; \mu_2, \Sigma_2)$$

where μ_1 , μ_2 are the mean . Σ_1 , Σ_2 are the covarience of the two probability distribution of P(x) and Q(x).

so by applying K-L divergence ,

$$D_{kl}(P(x) \parallel Q(x)) = \frac{1}{2}[log \frac{\Sigma_1}{\Sigma_2} - K + tr(\Sigma^{-1}\Sigma) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)]$$

where K=$tr[I_k]$

2.1.1 Properties

- $D_{kl}(P \parallel Q) \geq 0$

- $D_{kl}(P \parallel Q)$ is not equal to $D_{kl}(Q \parallel P)$

- The KL divergence is suitable for continuous distributions, and also invariant under parameter transformations

- The KL divergence is additive for independent distributions . If P1 ,P2 are the independent distributions, with the joint distribution P(x) = P1(x)P2(x) and If Q1 ,Q2 are the independent distributions, with the joint distribution Q(x) = Q1(x)Q2(x) then KL divergence is defined as:

$$D_{kl}(P \parallel Q) = D_{kl}(P1 \parallel Q1) + D_{kl}(P2 \parallel Q2)$$

2.1.2 Drawbacks

In the defination of kl divergence ,

$$D_{kl}(P \parallel Q) = \sum_{x} P(X = x) log \frac{P(X = x)}{Q(X = x)}$$

where Q(x) > 0 and if P(x) = 0 then the KL divergence blown up, which means MLE assigns an extremely low cost to the scenarios, where the model generates some samples that do not locate on the data distribution. So KL divergence is not able to handle zero distribution .To handle zero distribution we are using Jensen-Shannon divergence.

2.2 Jensen-Shannon Divergence(JSD)

It is the improved version of the KL divergence . It also measure the similarity between the probability distribution . It is also known as Information radius or total divergence to the average. Let P(x) and Q(x) be the two different probability distribution then the Jensen-Shannon divergence of P(x) on Q(x) will be defined as:

$$JSD(P \parallel Q) = D_{kl}(P \parallel M) + D_{kl}(Q \parallel M)$$

where M = $\frac{1}{2}(P + Q)$

2.2.1 Why Jensen-Shennon Divergence(JSD) is preferred over KL Divergence ?

Jenson Shannon divergence is preferred over KL divergence because, in KL divergence, it cannot handle the zero probability distribution(the biggest drawback of KL divergence) , while in JS divergence this can easily handle zero distribution. That's why we use JS divergence in measuring the performance of the generative network.
Here we can prove it by mathematically:

$$JSD(P \parallel Q) = D_{kl}(P \parallel \frac{1}{2}(P + Q)) + D_{kl}(Q \parallel \frac{1}{2}(P + Q))$$

$$JSD(P \parallel Q) = \frac{1}{2}\sum_{x \epsilon \Omega}[p(x)log(\frac{2p(x)}{p(x) + q(x)}) + q(x)log(\frac{2q(x)}{p(x) + q(x)})]$$

Now assume one distribution will be 0 . Let say P(x) =0 . Put the value of P(x) in the above equation.we get,

$$JSD(P \parallel Q) = q(x)\frac{log2}{2}$$

This prove that JS divergence handle zero distribution.

2.3 Variational Autoencoder (VAE)

Variational Autoencoder is one of the generating model to generate a new sample of data. It is the probabilistic version of autoencoder. The goal of VAE is to find a distribution q(z/x) of some latent variable z which we can sample from $z_\phi(z/x)$ to generate new sample $x^{'} \sim p_\theta(x/z)$.

In the above architecture we found that there are 3 parts of variational autoencoder :

- **Probabilistic Encoder** : it is given as $q_\phi(z/x)$ where it converts the input sample of data (x) to a sample latent vector(z). it is also called a recognition model

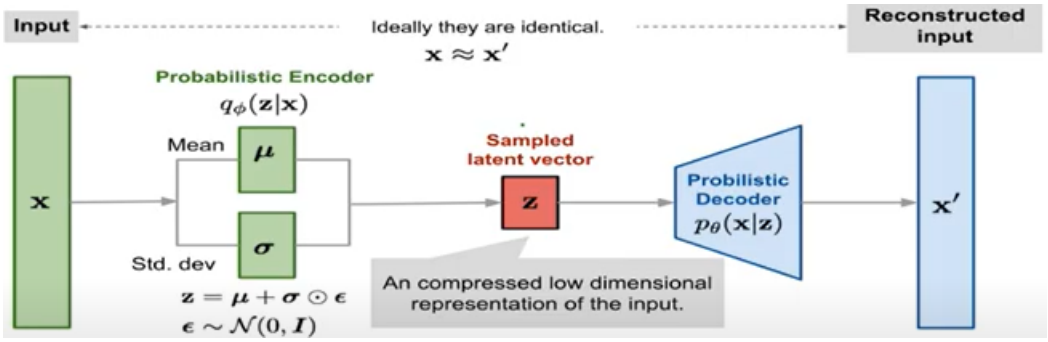


Figure 2.1: Architecture of Variational Autoencoder

- **Latent Variable** : It is denoted as z which corresponding to the real feature of the object that has not been measure(maybe technology is not able to do that)

- **Probabilistic Decoder** : It is denoted as $p_\theta(x/z)$, where it converts the latent variable(z) to the reconstructed image $(x^{'})$. It is also called as generating model.

Here θ and ϕ are the learning parameter. We have to learn the parameter by applying backpropagation. So, the loss function of the variational autoencoder is

$$L(\theta, \phi) = -E_{z \sim q_\phi(z/x)}[p_\theta(x/z)] + D_{KL}(q_\phi(z/x) \parallel p_\theta(z))$$

The target of VAE is to find the optimalϕthat minimizes the KL divergence between q and p.

Note : Because of noise, imperfect reconstruction, and with the standard decoder, the generated samples data are much more blurred than those coming from GANs

2.4 Generative Adversarial Network (GAN)

Generative Adversarial Networks are generative models which learns the distribution implicitly using a game-theoretic approach. They are the state- of-the-art models that generate extremely realistic samples from complex, high dimensional distributions. Various architectures of GAN have been proposed over the past two years which can be used for applications such as image generation, text synthesis, pose estimation etc. The architecture of GAN consist of two parts : Generator and Discriminator which competes with each other.

Generator generates fake images and discriminator tries to distinguish fake samples from real samples. The generator starts generating realistic images once it reaches the point of Nash equilibrium. The main advantage of GAN is that it does not require any approximate inference or approximate partition function of gradient. The basic architecture of GAN is as shown in the below Figure . The objective function of GAN is defined by:

$$L(D;G) = min_{\theta_g} max_{\theta_d} [E_{x_{data}} log D_{\theta_d}(x) + E_{\sim P_z} log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

where θ_g and θ_d are the parameters of Generator and Discriminator respectively.

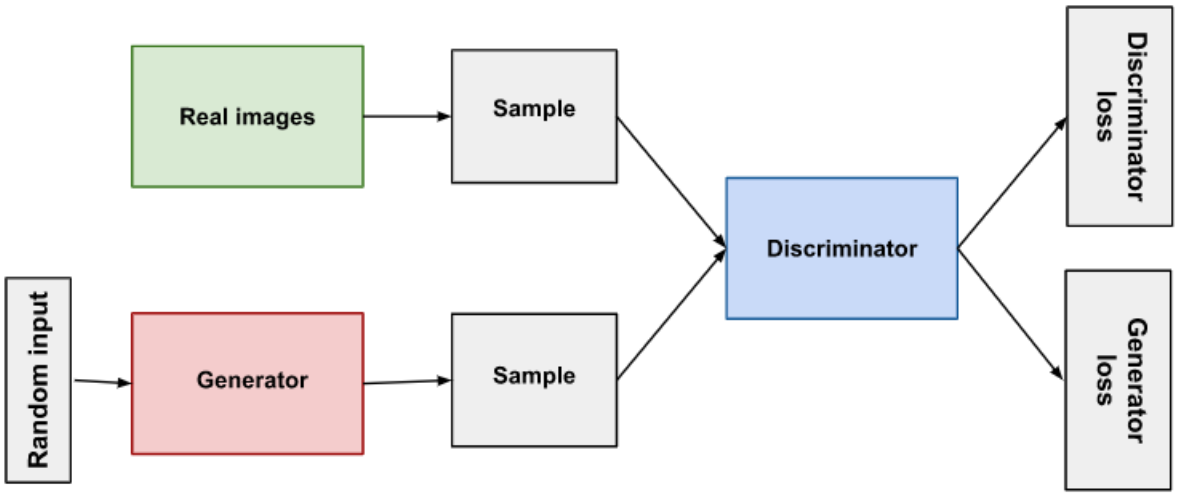


Figure 2.2: Architecture of GAN

Some of the recent architectures of GAN are discusses briefly in the next section.

2.5 Conditional GAN (CGAN)

Conditional GAN is the extension of the GAN model. This GAN model will generate the samples according to the given condition. The discriminator and the generator both receive some condition. The condition may be some column label or some property. For example, if we train a GAN model with mnist dataset, then it will generate a new mnist images. The generated image have no control over which specific digit will produce in the generator. so there is no mechanism on how to produce a specific number from the generator. This kind of problem can be solved a specific type of GAN called a condition GAN(CGAN). We add an additional input layer with the values of one hot encoded image label.

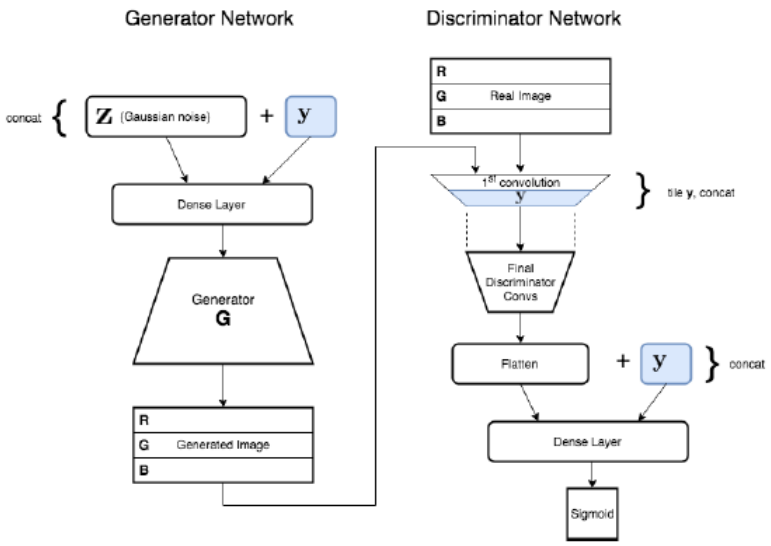


Figure 2.3: THE CGAN Architecture

Let c be the vector that to be conditioned on both the network. The input of the generator network is the random noise (z) with conditioned to the vector c. The output will be the G(z). This output will be fed to the discriminator model along with the real data sample x with conditioned to the vector c which gives the output as D(G(z)). This networks will learn to adapt and adjust their parameters to these additional inputs. The loss function of both the network will be as follows

$$L_G = -\mathbb{E}_{\mathbb{Z}\sim\mathbb{P}_\mathbb{Z}}[log(D_w(G_\theta(z \mid c)))]$$

$$L_D = -\mathbb{E}_{\mathbb{Z}\sim\mathbb{P}_\mathbb{Z}}[log(1 - D_w(G_\theta(z \mid c)))] - \mathbb{E}_{\mathbb{X}\sim\mathbb{P}_\mathbb{R}}[log(D_w(x \mid c))]$$

by conditioning the model on additional information, it is possible to direct the data generation process. Such conditioning could be based on class labels or any other process also.

2.6 Deep Convolution GAN (DCGAN)

The images generated from normal Generative Adversarial Networks are noisy and incomprehensible. Deep Convolutional GAN was introduced in which the architecture of normal

GAN is modified by adding convolution layers to the generator without max pooling or fully connected layers. Convolutional stride and transposed convolution are used for the downsampling and the upsampling respectively. Architecture guidelines proposed by for stable training of Deep Convolutional GANs are as follows
[4]

- The pooling layer should be replaced with fractional-strided convolutions, in case of the generator and strided convolution, in case of discrimination.
- we use batchnorm in both the generator and the discriminator.
- All fully connected should be removed for the deeper architecture
- Here activation function is the ReLu function for every hidden layer in the generator except for the output. In the output, we use Tanh function.
- For the discriminator, the activation function will be Leaky ReLu for all layers.

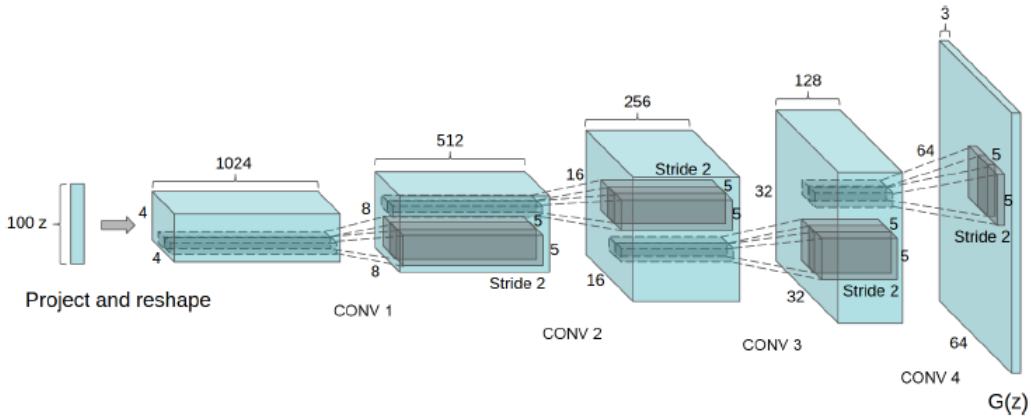


Figure 2.4: THE DCGAN Architecture

The modified architecture of DCGAN gives better results than that of GAN. It generates the good quality of images. However, DCGAN suffers from Mode collapse for the small dataset and is independent of noise input, that results in the non-convergence of the model. To avoid such problem we introduce Wasserstein GAN(WGAN)

2.7 Wasserstein GAN(WGAN)

As discussed above, DCGAN suffers from mode collapse and the results might even oscillate and never converge. Wasserstein GAN is the alternate way of training GAN which addresses the MODE COLLAPSE and gives the stability of the training. It shares the same architecture of the GAN except that the discriminator, which does not have the output sigmoid function. Here in WGAN, we are using Wasserstein distance or Earth-Mover(EM) distance instead of JS divergence. This Earth-Mover distance is the stable distance metric for PDF comparison. It measure the similarity between the two probability distribution by calculating the distance between them horizontally not vertically as we are doing in KL and JS divergence. Wasserstein distance between two distributions Pr and Pg is given by:
[5]

$$W(P_r, P_g) = inf_{\gamma \epsilon (P_r, P_g)}[\|x - y\|]$$

This Wasserstein metric gives smooth gradients in all the regions and solves the problems of vanishing and exploding gradients. Therefore it gives stability, even when the generator is not generating good images. The discriminator is referred to as critic. The architecture of WGAN is shown below figure :

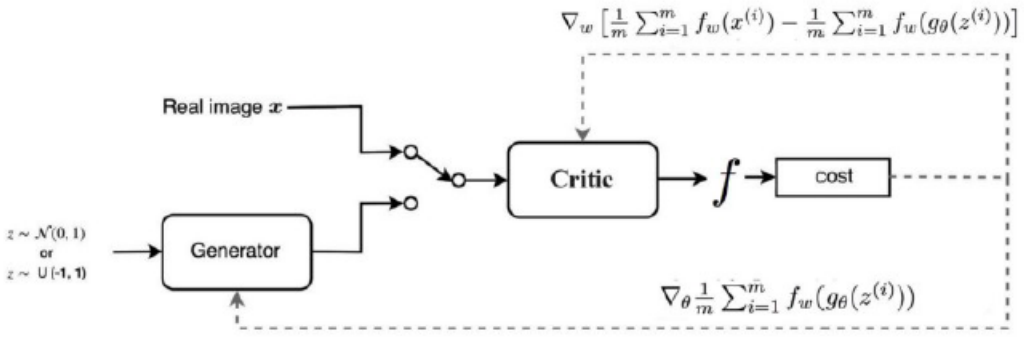


Figure 2.5: THE WGAN Architecture

the objective of the WGAN is to minimize $(min_\theta w(p_x, p_\theta))$ where θ is the parameter of the generator network which will try to tune the p_θ to generate the probability distribution as close possible to P_r.So main target of WGAN to minimize the Wasserstein distance. Since the equation for Wasserstein distance is intractable, it is simplified using Kantorovich-Rubinstein duality,

$$W(P_r, P_g) = Sup_{\|D\| \leq 1} E[D(x) - D(g_\theta(z))]$$

where sup is the Least Upper bound function and D is a 1-Lipschitz. To enforce this constraint, the weights of discriminator are clipped to a certain range controlled by hyper-parameter c

$$w \leftarrow w + \alpha.RMSProp(w; gw)$$

$$w \leftarrow clip(w, -c, c)$$

The performance of GAN is highly sensitive to the hyperparameters. So the demerit of the clipping weight is that it is very important to tune the hyperparameter which otherwise gives the poor result and generating a rough image.

Another type of WGAN is the Gradient penalty(WGAN-GP). Instead of clipping weight, the model is penalized, if the norm of the gradient value moves away from 1. Although the gradient penalty is computationally expensive and produces high-resolution images.

The difference between the Wasserstein GAN and the GAN is as follows:

- In WGAN we are removing the sigmoid activation function in the last layer of the discriminator network, as we are working in the regression problem.
- There is no log term in the loss function of WGAN network. As in the GAN, log term is there
- In WGAN, the clipping of the parameter 'W' lies in the range of [-c,c]. The region for doing this is to satisfy the constraints in order to have K Lipschitz constant
- In WGAN, we are using RMS PROP as optimize function whereas in GAN, we use ADAM as the optimizer.

2.8 Conclusion

In this chapter, we discuss KL divergence and JS divergence. We come to know how KL divergence problems solved by JS divergence. We also discuss variational autoencoder

and generative adversarial networks and come to know how GAN is better for a generative model then VAE. We also discuss conditional GAN, DCGAN, WGAN, and also proved WGAN is better then DCGAN for the generative model.

Chapter 3

Related Works

Majority of the text to image algorithms select data from the pool. Recently, a new approach which uses Generative Adversarial Networks for text to image synthesis was introduced. This chapter discusses two different approaches which uses GANs for text to image.

3.1 Text Embedding

The text is converted to a vector before going to any model. This vectorization is commonly referred to as a text embedding. As text embedding is not the main focus of this work, so we directly use the pre-trained network of character-level CNN-RNN. The encoder maps the images and the captions to a common embedding space such that images and descriptions that match are mapped to vectors with a high inner product. For this mapping, a CNN works on the images, and a Convolutional-Recurrent Neural Network converts into the text. This architecture is similar to the Image caption. Here we can VGG-16 model as CNN architecture and LSTM for text description.
[6]

For CNN, we use VGG-16 architecture for images and LSTM for text description. The main function of VGG-16 is to extract the features of the image. In LSTM, we can translate the features(given by our image-based CNN model) to the embedded vector form. Then both of these features are fed to the decoder to get the vector embedding as output.

An alternative method is also there named Skip-Through Vector which is a language-based model. This model gives sentences with similar semantics and syntax to similar vectors. In spite of that, the char-CNN-RNN encoder is suited better than that of Skip - Through vector for vision tasks because it gives the corresponding images of the descriptions as well. The vectorized form is similar to the convolutional features of the images

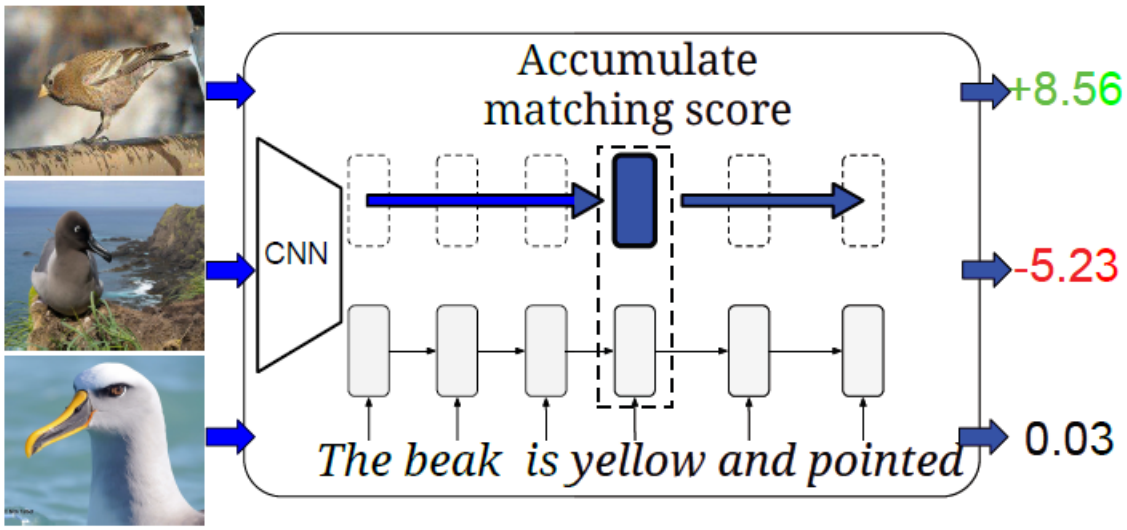


Figure 3.1: Architecture of text embedding

that correlate with, which makes them visually discriminative. This property shows good performance when the embeddings are works inside CNN.

3.2 GAN Based Text to Image(GAN-CLS)

Text to image synthesis is one of the practical applications of the GAN family. This is first introduced in 2016 using the generative adversarial network(GAN). The main advantage of GAN-CLS is that it increases learning speed by generating instances for querying. We use DCGAN conditioned on the text description.

In GAN-CLS, architecture divides into two-part. First, we will find the visually discriminative representation for the text description, and from that result, we generate the images.

In the above figure, it shows 2 fully connected layers compress the text embedding vector $\phi(t)$ and append in both the discriminator and the generator. In the discriminator, the compressed embedding is spatially replicated before going to append in depth. The function D(x) and G(z) are the output of discriminator and the generator that come across in regular GAN become in context with conditional GAN. By applying the conditional GAN with the text embedding $\phi(t)$, the output of the generator and the discriminator will

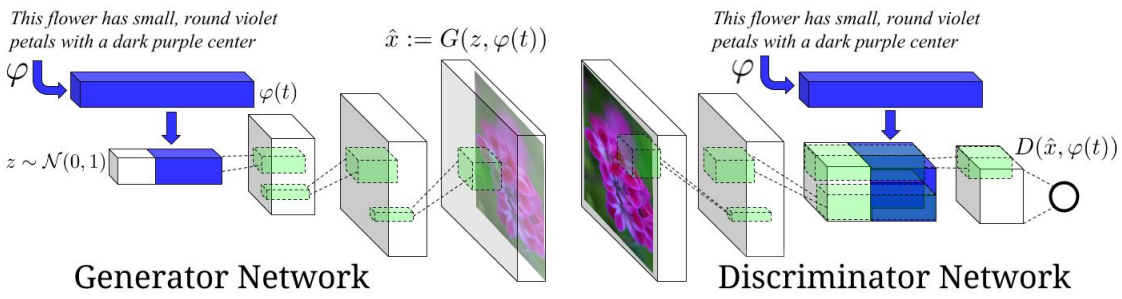


Figure 3.2: Architecture of text to image using GAN

be $G(z, \phi(t)) and D(x, \phi(t))$ where $\phi : \Sigma^* \rightarrow \mathbb{R}^{\mathbb{N}_\phi}$ is the char CNN-RNN encoder, t is the text description which gives the vector of character,$\Sigma$is the alphabet of the text description and $N_\phi$is the dimension of text embedding. The text embedding is the conditioned vector c.

3.2.1 Model Architecture

GAN based Text to image synthesis uses a deep convolution network both in the discriminator and the generator similar to DCGAN.

The text description 't' is passed to a function ϕ gives the output $\phi(t)$. It gives an embedded dimension of 4098. Then the embedded vector willcompressed with a dimension of 128 by applying activation function of Leaky ReLu and concatenated with a noise vector(z), which has dimension 128 having a normal distribution of mean 0 and unit covariance. After concatenate, the result is fed to the generator network as input. In the generative network, the input vector is transformed with a linear projection and passed through many deconvolution layers with Leaky ReLu activation function, until a final tensor having a dimension of $64 \times 64 \times 3$ is obtained. After the final layer is formed, the result will be passed to tanh activation function to bring the pixel value in the range of[-1,1].

The real image is passed to a discriminator who has a series of deconvolution until the spatial resolution becomes 4×4. Then the text embedding will be compressed to a dimension of 128 using a fully connected layer by applying the activation function of Leaky Relu as same as in the generator. After getting the compressed vector embedding, it spatially replicated and concatenate in-depth to the convolutional features of the network.The resultant tensor is then passed to a series of convolutions until a scalar is formed. Then the result is passed to a sigmoid activation function which is in the range of [0,1], gives a valid

probability.

3.2.1.1 Loss Function

The GAN-CLS loss function has a small difference compared to the loss function of conditional GAN. The goal of this modification is to better enforce the text to image synthesis. the loss function will be

$$L_D = -\mathbb{E}_{(\boldsymbol{X},\boldsymbol{E})\sim\mathbb{P}_{r-mat}}[\log(D(\boldsymbol{x},\boldsymbol{e}))] - \frac{1}{2}(\mathbb{E}_{(\boldsymbol{X},\boldsymbol{E})\sim\mathbb{P}_{ge}}[\log(1-D(\boldsymbol{x},\boldsymbol{e}))] + \mathbb{E}_{(\boldsymbol{X},\boldsymbol{E})\sim\mathbb{P}_{r-mis}}[\log(1-D(\boldsymbol{x},\boldsymbol{e}))])$$

whereas,

P_{r-mat} is the joint distribution of text embedding and the image which matches.

P_{r-mis} is the joint distribution of text embedding and the image which mismatch.

P_{ge} is the joint distribution of text embedding and the image that was generated

This shows that the discriminator has double functionality, which distinguishes both fake and real image. It also distinguishes the text image pair that match and that not match. That is the KL divergence between the conditioning Gaussian distribution and the standard Gaussian distribution.

3.2.2 Limitation

- The output of the generator is an image of 64×64 resolution conditioned on the text description. This resolution is not sufficient to describe the text that we provided in the input. It contains a lot of noise and false information. To rectify these problems, we convert a high-resolution image of 256×256 with the help of an architecture named stackgan which will be described in detail in the next topic.

- Mode collapse occurs very frequently.

3.3 StackGAN

As we have seen in the previous architecture GAN-CLS, unable to generate a high-resolution image. Because of this, its fail to give useful information according to the text description. So,to overcome this problem, we use STACKGAN, that gives a high-resolution image of 256×256. The advantage of this architecture is that it generates a photo-realistic image, and give clear information of text description.[**?**]

Stackgan contains 2 stages of GAN. In stage 1 GAN, is similar to GAN-CLS that draws the shape and color of an object based on a given text description which results stage1 low-resolution image. In stage 2 GAN, the input will be the output of stage1 GAN image concatenate with the text description and the result will be the high-resolution image. To increase the input image and stabilize the training of the conditional GAN, we use the conditional augmentation technique in the text embedding, which gives smoothness in the latent conditioning manifold. Before going to discuss architecture, first, we will discuss the text embedding augmentation in detail.

3.3.1 Text Embedding Augmentation

The text description 't' is first encoded into a function ϕ which gives a text embedding $\phi(t)$. This text embedding is non linearly transformed that generates a conditioning latent variable. This latent variable will be fed as an input to a generator. However the shape and size of the text embedding is usually high dimension(> 100 dimensions), but the latent variable is not enough size to generate an image. With a limited amount of data, it occurs discontinuity in latent data, which fails to generate an image in the generator. To solve this problem we introduce the conditional augmentation technique which generates an additional conditional variable. Now the conditioning text variable C is randomly sample the text data which is a independent normal distribution$\mathcal{N}(\mu(\phi(t)), \Sigma(\phi(t)))$ where$\mu(\phi(t))$ is the mean of the normal distribution and $\Sigma(\phi(t)$ is the covariance matrix of the normal distribution . These mean and the covariance matrix are the functions of text embedding. This augmentation produces a large number of text image pair, which given a small number of text image pair and thus support the robustness and the flexibility to small perturbation with conditioning manifold. To avoid the overfitting the model and the increase in the smoothness over the conditioning manifold we apply the regularization term i.e KL divergence. [7]

$$\mathcal{D}_{\mathcal{KL}}(\mathcal{N}(\mu(\phi(t)), \Sigma(\phi(t))) \parallel \mathcal{N}(0, I))$$

which is the Kullback-Leibler divergence (KL divergence)between the standard normal distribution and the conditioning normal distribution.

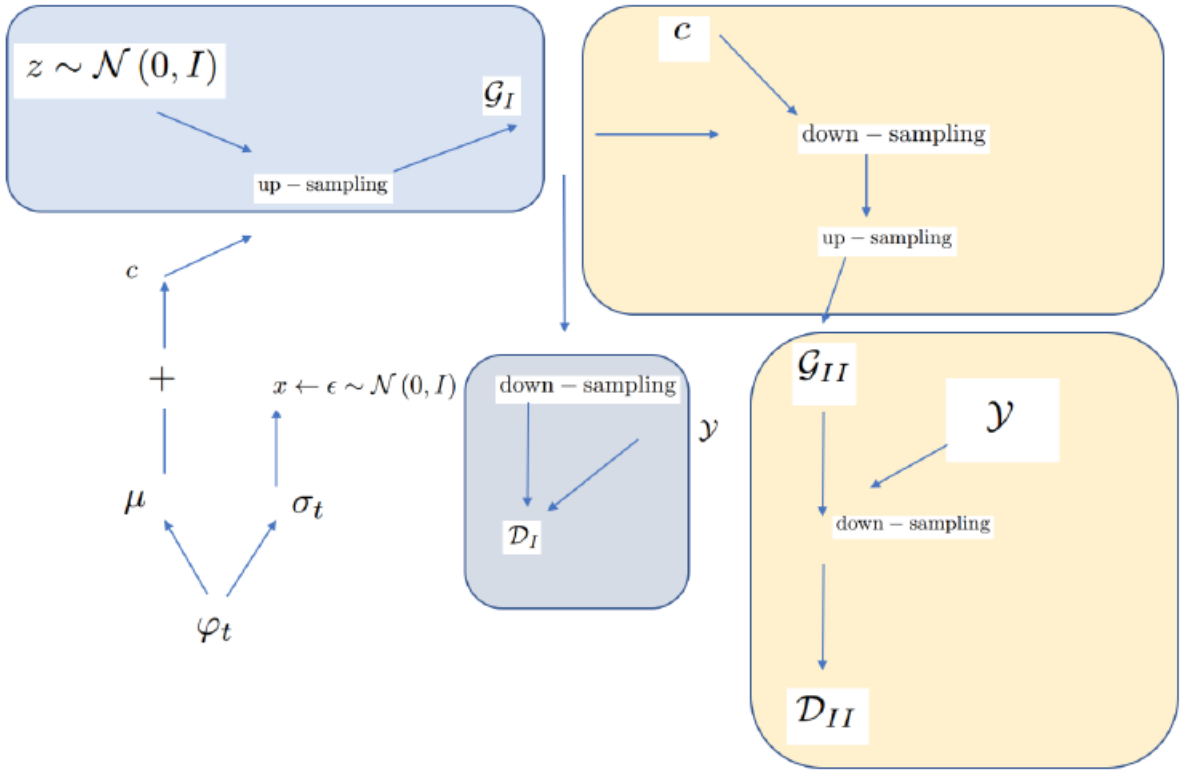


Figure 3.3: Flow graph of stackgan

3.3.2 Model Architecture

In this achitecture we apply 2 stage of GAN namely stage 1 and stage2 GAN.The architecture of stage1 gan is same as the architecture of GAN-CLS with the addition of the conditioning augmentation which are discussed previously.The loss function of stage1 gan is

$$\mathcal{L}_{D_0} = \mathbb{E}_{(I_0,t)\sim P_{data}}[logD_0(I_0, \phi_t)] + \mathbb{E}_{z\sim P_z, t\sim P_{data}}[log(1 - D_0(G_0(z, \hat{c_0}), \phi_t))]$$

$$\mathcal{L}_{G_0} = \mathbb{E}_{z\sim P_z, t\sim P_{data}}[log(1 - D_0(G_0(z, \hat{c_0}), \phi_t))] + \lambda D_{KL}(\mathcal{N}(\mu_0(\phi_t), \Sigma_0(\phi_t) \parallel \mathcal{N}(0, I))$$

whereas,

$I_0$is the real image

$P_{data}$is the real probability distribution

z is the randomly noise vector

p_z fake probability distribution

λ is a regularization parameter that balances the two terms. We set λ = 1 for all our experiments

After getting the low-resolution image in stage1, In stage 2 the generator starts with the downsampling of the stage1 output image of 64×64, until it reaches a spatial resolution of 4×4. After getting 4×4 blocks, it concatenates with the conditional augmentation of text embedding in depth which will improve the text image matching of stage1 GAN. Then the concatenate block will pass through 3 residual blocks and then it passes through the upsampling layer until the final tensor will be $256 \times 256 \times 3$ is obtained. After the final block is obtained the last layer will be the tanh activation function that gives the output in the range [-1,1].[8]

As we discuss stage1 discriminator architecture is similar to the GAN-CLS discriminator. The stage2 discriminator is also similar with the addition to 3 more downsampling convolution layer is used for higher resolution. In stage 2, the ReLu activation function is used in the generator and the Leaky ReLu activation function is used in the discriminator. We use Batch Normalization in both the discriminator and the generator. The loss function of stage2 GAN will be.

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

$\mathrm{L}_G = \mathbb{E}_{s_0\sim P_{G_0},t\sim P_{data}}[log(1 - D(G(s_0,\hat{c}),\phi_t))] + \lambda D_{KL}(\mathcal{N}(\mu(\phi_t),\Sigma(\phi_t) \parallel \mathcal{N}(0,I))$
whereas,

I will be the real image
$\hat{c}$ is the conditional augmentation of text embedding
$s_0 = G(z)$which is the output of the generator

In this way, Stage2 GAN learns to generate useful information, where in Stage1 GAN fails to generate.

3.3.3 Drawbacks

- Training a stage1 GAN takes huge time
- Mode collapse occurs frequently.So, before training the stackgan, tune the hyperparameter properly

3.4 Conclusion

In this chapter, we discussed two variants of GAN based approach for Text to Image synthesis. Next chapter discusses a proposed modication in GAN-CLS that takes visual interpretability of images into account.

Chapter 4

Low Resolution Text To Image Synthesis

In this chapter, we try to implement a new model of text to image synthesis which address some of the current research problem .

4.1 Proposed Approach

In the previous chapter, we discuss the various architecture of text to image synthesis. There, We suffer many problems such as failure of useful image, instability during training, and mode collapse. On taking into all these accounts, we will use a Wasserstein GAN on condition with a text description. As we know the Wasserstein GAN will generate an image better than that of DCGAN because of Wasserstein distance, which acts as a regularization. we discussed the architecture and the mathematical formulation of Wasserstein GAN in the preliminary part. In this section, we directly implement the approach towards text to an image .
[9]

In this model, the text embedding and the Conditioning Augmentation of the text embedding will be the same as in the case of StackGAN. The main difference will be the loss function of the discriminator. The loss function will be the loss function of the conditioned wasserstein GAN loss function and that is,

$$L_D = \mathbb{E}_{(X,E)\sim\mathbb{P}_{ge}}[D(x,e)]+\alpha\mathbb{E}_{(X,E)\sim\mathbb{P}_{r-mis}}[D(x,e)]-(1+\alpha)\mathbb{E}_{(X,E)\sim\mathbb{P}_{r-mat}}[D(x,e)]+\lambda L_{LP}$$

where

α is the parameter that controls the level of text-image matching.

L_{LP} is another regularization term that enforces the Lipschitz constraint.

The objective of the loss function is to minimize $W(P_{r-mat}, P_{r-mis})$ i.e the distance between the two distribution. If the distance is increased, it damages the gradient penalty and it become so small that the gradient norm will out of control. To control this, We increasing λ, but, it not robust to change, instead we use regularization parameter L_{LP}(LP - Lipschitz Penalty). This parameter not only decreases the gradient norm to 1, but also allows for a larger value of λ without harms the model. Moreover it also helpful for converging faster, high stability and reduce the chance of mode collapse.L_{LP} is given as :
[10]

$$L_{LP} = \mathbb{E}_{(\hat{X},E)\sim\mathbb{P}_\eta}[max(0, \|\nabla_{\hat{x}} D(\bar{x}, e)\| - 1)^2 + max(0, \|\nabla_e D(\hat{x}, e\| - 1)^2]$$

$\mathbb{P}_\eta$is the joint distribution of image text pairs (,e).
$\hat{x} = tG(z, e) + (1 - t)x$ which is the linear interpolation with t and e is the matching text embedding of the real image x.
$D(\hat{x}, e)$ is the function of text embedding e.

Now for the generator ,the loss function will be ,

$$L_G = -\mathbb{E}_{(X,E)\sim\mathbb{P}_g}[D(x, e)] + \mathbb{E}_{T\sim p_r}[\rho KL(N(0, I) \parallel N(\mu(\phi(t)), \Sigma(\phi(t))))]$$

This loss function is the same as the loss function of the WGAN on an addition to the conditional augmentation loss that maintains the standard normal distribution over the conditional latent space.

Now, the architecture of the generator is similar to that of stage1 StackGAN. For the discriminator, we remove the batch normalization because of gradient penalty to work. With the presence of batch normalization, the gradient penalty fails to assume a unique gradient for each sample as Wasserstein distance is used. To keep the training balanced, the discriminator no longer needs to be crippled. In the discriminator, we add one more convolution layer after the text embedding concatenation and also use the same number of convolution filters as in the generator.

GENERATOR	DISCRIMINATOR
input :$noise(z^{100}) + textembedded(1024 \times 1)$	input1 : $64 \times 64 \times numberofchannel$
FC 128×1 , ReLU	4×4 Conv, 64, LeakyReLU, stride 2
FC , 128×1, BatchNormalize	4×4 Conv, 128, LeakyReLU, stride 2
FC, 128×1,Leaky ReLU	4×4 Conv, 256, LeakyReLU, stride 2
UpSampling2D (size=(2, 2))	4×4 Conv, 512, LeakyReLU, stride 2
3×3 ConvTranspose, 516, ReLU, stride 1	input2 :Input(shape=(4, 4, 128))
UpSampling2D (size=(2, 2))	input=input1 + input2
3×3 ConvTranspose, 256, ReLU, stride 1	1×1 conv2D ,512, LeakyReLu ,stride 1
UpSampling2D (size=(2, 2))	FC , ReLu,512
3×3 ConvTranspose, 128, ReLU, stride 1	FC , Sigmoid , 1
UpSampling2D (size=(2, 2))	
3×3 ConvTranspose, 64, ReLU, stride 1	
3×3 ConvTranspose, 3, tanh, stride 1	

Table 4.1: Conditional WGAN Architecture

4.2 Experiment Setup

Text to Image is done using the proposed approach. The experiments are done for two different variants of GAN, DCGAN and WGAN with Gradient Penalty. For training the discriminator, we have to fixed the generator. The better the approximation of $W(P_{r-mat}, P_{r-mis})$, the closer the discriminator to gets optimal.Therefore, we take critic-iteration = 5 . It means that the discriminator is trained 5 times for every generator update. We take the noise dimension is 100 and embedding dimension is 1048. The maximum number of epochs is set to 120,000 with a batch size of 64. The learning rate is set to 0.002 and the optimizer used is Adam.

Stochastic gradient descent is employed to optimize the cost function[11] . Instead of fixing the number of iterations for gradient updates for the latent variable, the minimization is done till convergence. The learning rate is set to 0.002 and threshold value for convergence is set to $1e^{-4}$.β_1 and β_2 is a hyperparameter of both generator and discriminator and after tuning, the value is set to 0.5 and 0.9.The other parameter like the generator regularization term ρ is set to 10,the discriminator regularization term λ_{LP} is set to 150 and $\alpha = 1$.

The architectures of deep networks involved in the experiment are given above.

4.3 Results

The results of the proposed approach compared with GAN-CLS and Random Sampling from GAN are shown in the below Figure . The GAN architecture used in this case is Conditional WGAN. The proposed approach is generated for two different values i.e β_1 = .5 and β_2 = .9 . It can be seen that the Image for proposed algorithm (for both values of β) which adds visual interpretability to the images generated by GAN-CLS . Hence, it can be concluded that the performance of the algorithm that is proposed is comparable to that of GAN-CLS is better. Here some of the image generated by conditional WGAN as follows:

TEXT : small bird with propotionate head and a small pointed bill

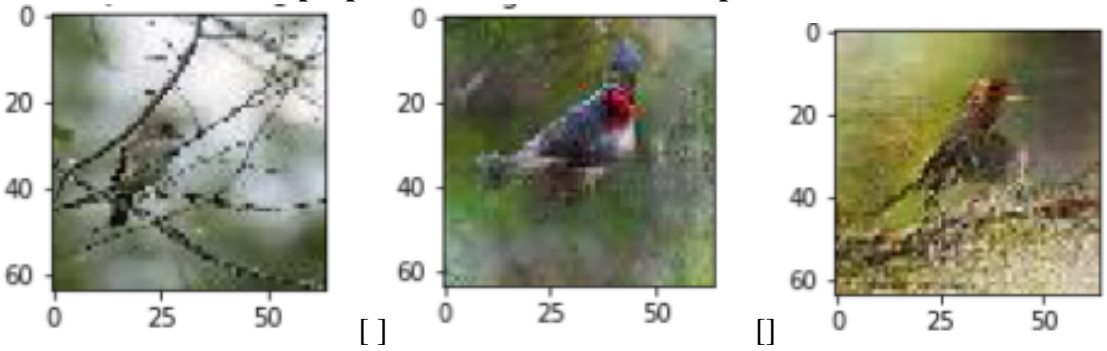


Figure 4.1: comparison between. (a) EXPECTED IMAGE (b) GAN-CLS (c) CONDITIONAL WGAN.

TEXT : Bird has small body and large head , black and gray wingspan and tail

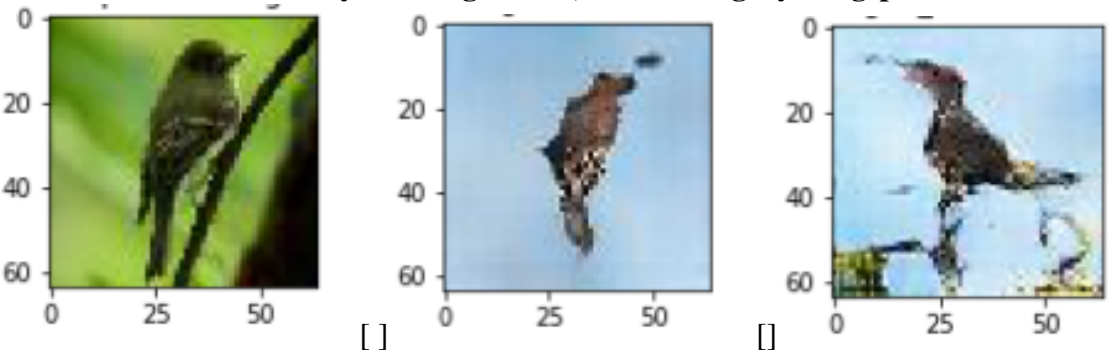
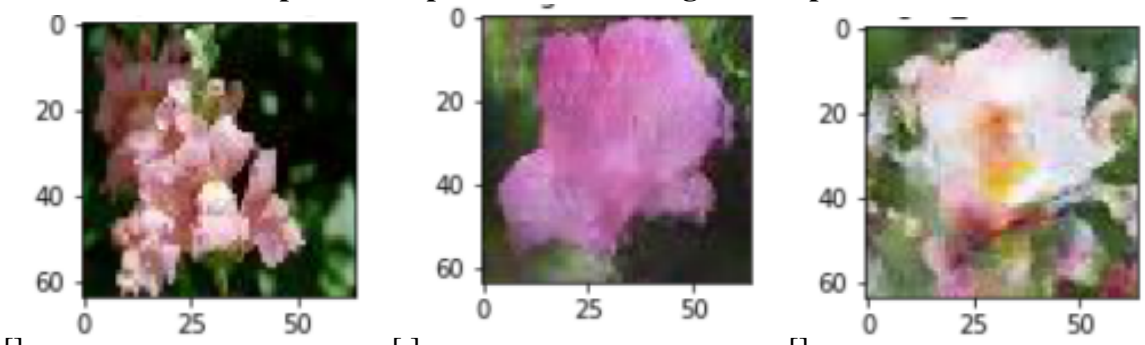


Figure 4.2: comparison between. (a) EXPECTED IMAGE (b) GAN-CLS (c) CONDITIONAL WGAN.

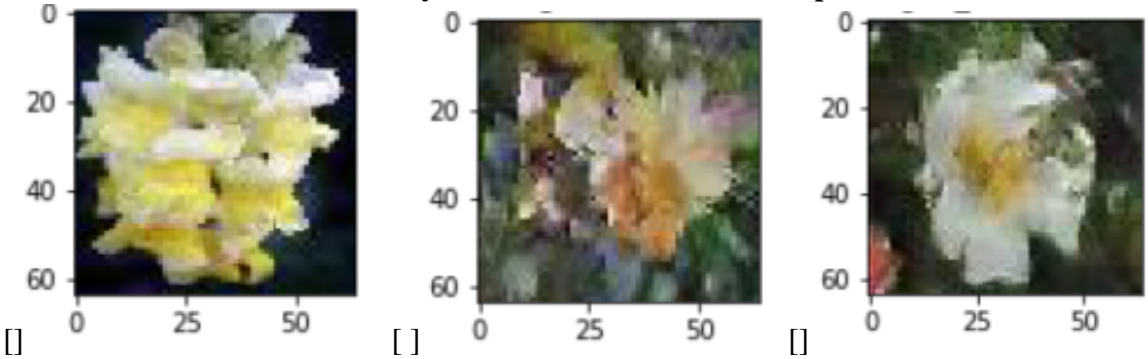
TEXT : flower have pink color petal and have irregular shape



[] [ ] []

Figure 4.3: comparison between. (a) EXPECTED IMAGE (b) GAN-CLS (c) CONDITIONAL WGAN.

TEXT : flower have white and yellow in color and have soft patel



[] [ ] []

Figure 4.4: Comparison between (a) EXPECTED IMAGE (b) GAN-CLS (c) CONDITIONAL WGAN.

In conditional WGAN, we are taking the learning rate = .002 on both generator and the discriminator network.The other parameter ,the generator regularization term ρ is set to 10,the discriminator regularization term λ_{LP} is set to 150 and $\alpha = 1$. We have plot how the GAN loss is varied with the iteration.

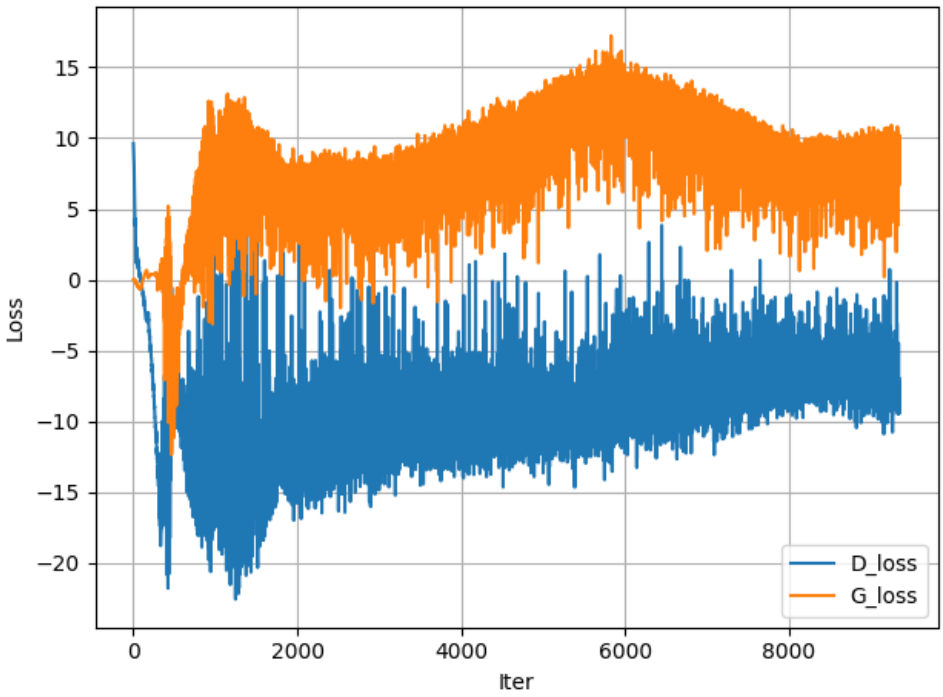


Figure 4.5: Analysis of the generator and the discriminator loss against with the iteration

.

4.4 Conclusion

we have successfully generate a low resolution image using conditional WGAN and also compare the image of GAN-CLS model .we conclude that the conditional WGAN model is better then GAN-CLS . In the next chapter we will generate a high resolution image taking the input of low resolution image.

Chapter 5

High resolution Text To Image

Previously,We successfully generated a Low-resolution image using conditional Wasserstein GAN. In this chapter, we will generate a high-resolution image by taking low resolution image. Before going to the implementation part, First, we like to discuss one of the most successful GAN architecture i.e super-resolution generative adversarial network(SRGAN)

5.1 Super resolution Generative adversarial network (SRGAN)

Super-resolution generative adversarial network(SRGAN) is one of the GAN architecture that will generate the super-resolution images with finer detail and high quality image from the low-resolution image. Before SRGAN, CNN is able to generate high-quality images that train fast and achieve high accuracy.However, in some cases it fails to generate finer details and generate blurry images. Like GAN, SRGAN contains a generator network and discriminator network. Both networks are the deep convolution network.[12]

In **Generator**, the input of this network is a low-resolution image of shape $(64 \times 64 \times 3)$. This image will go through the deep network contains a series of convolution and upsampling layers and gives the output as a high-resolution image of shape $(256 \times 256 \times 3)$.

In **Discriminator**, the input of this network is a high resolution of real data and the generated image (output of the generator). This network tries to identify whether the generated image is real(from the real data sample) or fake(generated from the generator).

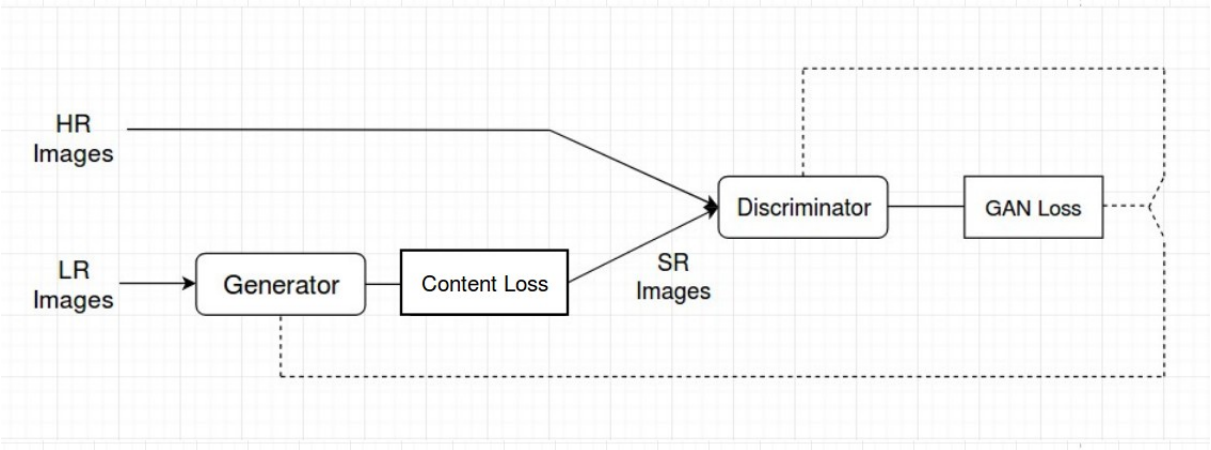


Figure 5.1: Flowgraph OF SRGAN

5.1.1 Architecture

Here we will discuss the architecture of the generator and the discriminator network of super-resolution GAN.

Both the networks are the deep convolution neural network which contains both convolution layer and the upsampling layer. Each convolution layer is followed by batch normalization and the activation layer.

The architecture of the generator contains the following blocks:

- **Pre-Residual Block**: This block contains a 2d convolution layer followed by a ReLu activation layer
- **Residual Block**: This block contains two 2d convolution layer and each convolution layer is followed by a batch normalization with a momentum of .8. The last layer will be the additional layer which calculates the sum of input tensor to that block and the output of the last batch normalization layer. the generator contains 16 residual block[**?**]
- **Post Residual Block**: This block contains a 2d convolution layer followed by a ReLu activation layer and a batch normalization with a momentum of .8[13]
- **Upsampling Block**: This block contains two 2d upsampling blocks. Each upsampling block contains one upsampling layer followed by a 2d convolution layer and the ReLu activation function

- **Last Convolution Layer**: This layer contains one 2d convolution layer that use tanh as an activation function.generates a high-resolution image of shape $(256 \times 256 \times 3)$.

The architecture of the discriminator contains 8 convolution block followed by a 2 dense layer (fully connected layer). Each convolution block is followed with a batch normalization layer. The last layer predicts the probability of the image that belongs to a fake image or a real image.[14]

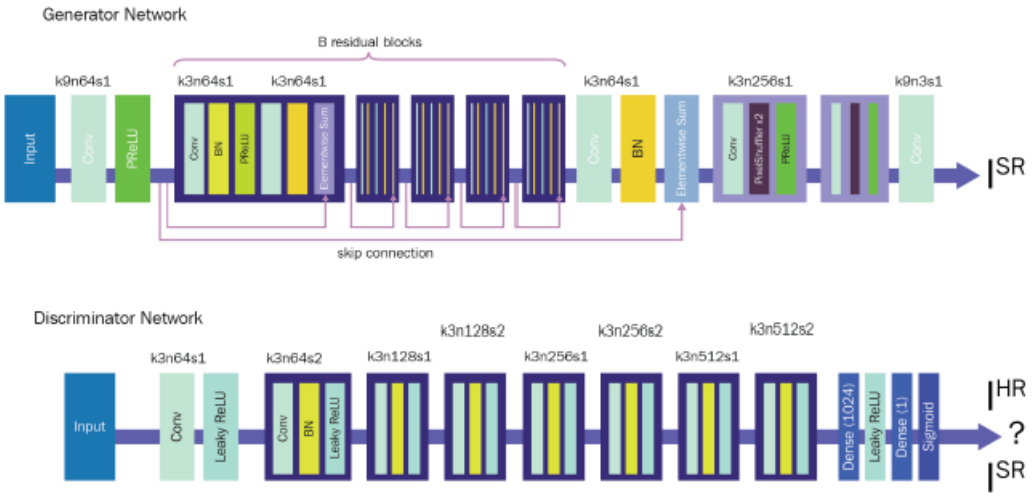


Figure 5.2: Architecture OF SRGAN

5.1.2 Loss Function

In SRGAN, there is two kinds of loss are present as shown in the figure above, i.e content loss and the GAN loss(also called adversarial loss).

Content Loss:

There are two types of content loss:

1. Pixel wise MSE(mean squared error) loss
2. VGG19 Loss

Pixel wise MSE loss:

This type of content loss is calculated between each pixel value of the generated image and each pixel value of the real image. It shows how different the generated image from the

real image. The mathematical expression of the pixel-wise MSE loss is as follows:

$$l_{MSE}^{SR} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I_{x,y}^{HR} - G_{\theta_G}(I^{LR})x,y)^2$$

whereas,

$G_{\theta_G}(I^{LR})$ is the high-resolution image generated by the generator.
I^{HR}= is the high-resolution image sampled from the real data.

VGG19 Loss

This is another type of content loss which is applied over real image and the generated image. VGG19 is a popular deep neural network that is mostly used for image classification. We can used the model for extract feature map of the real image and the generated image. The mathmatical expression for the VGG19 loss is as follows:

$$l_{VGG/i,j}^{SR} = \frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}(\Phi_{i,j}(I^{HR})x,y - \Phi_{i,j}(G_{\theta_G}(I^{LR}))x,y)^2$$

whereas,

$\Phi_{i,j}$ is the feature map generated by the VGG19 model

$\Phi_{i,j}(I^{HR})$ is the extracted feature map of the real image

$\Phi_{i,j}(G_{\theta_G}(I^{LR})$ is the extracted feature map for the generated image

$l_{VGG/i,j}^{SR}$ is the euclidean distance of the extracted feature map of the real image and the general image

we can use either of the content loss. In our project, we are using VGG19 loss as the content loss

Adversarial Loss

This loss is calculated on the probabilities returned by the discriminated network. In the adversarial model the input of the discriminator model is the output of the generator model.The mathmatical expression of the adversarial loss is as follows:

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -logD_{\theta_D}(G_{\theta_G}(I^{LR}))$$

Where as ,

$G_{\theta_G}(I^{LR})$ is the generated image

$D_{\theta_D}(G_{\theta_G}(I^{LR}))$ shows the probability that the generated image is the real image

Now the perceptual loss is the weighted sum of the adversarial loss and the content loss which is represented by,

$$l^{SR} = .001 * l_{Gen}^{SR} + 1 * l_{X}^{SR}$$

whereas ,

l^{SR} is the perceptual loss

l_{Gen}^{SR} is the GAN loss and l_{X}^{SR} is the content loss

The objective of the SRGAN is to minimize the perceptual loss. If this happens, then the generator makes fool to the discriminator. If the perceptual loss decrease, the generated image will generate a more realistic image.

5.2 Experiment Setup

We run this model by taking the low-resolution image conditioned to the text description as an input. To Train this network, it contain two way process. In the first step, the discriminator network will train by keeping the generator network off. In the second step, the generator network will train. Overfitting is prevented by using the Early Stopping tech-

GENERATOR	DISCRIMINATOR
Input : low resolution image $(64 \times 64 \times 3)$	Input : real image $(256 \times 256 \times 3)$
3×3 convolution ,64 , ReLu , stride 1	3×3 convolution , LeakyReLu , stride 1
3×3 convolution ,64 , ReLu , stride 1	3×3 convolution , LeakyReLu , stride 2
batch normalization , momentum .8	batch normalization , momentum .8
3×3 convolution ,64 , ReLu , stride 1	3×3 convolution , LeakyReLu , stride 1
batch normalization , momentum .8	batch normalization , momentum .8
Additional layer , None	3×3 convolution , LeakyReLu , stride 2
3×3 convolution ,64 , ReLu , stride 1	batch normalization , momentum .8
batch normalization , momentum .8	3×3 convolution , LeakyReLu , stride 1
upsampling size=2	batch normalization , momentum .8
3×3 convolution ,128 , ReLu , stride 1	3×3 convolution , LeakyReLu , stride 2
upsampling size=2	3×3 convolution , LeakyReLu , stride 1
3×3 convolution ,256 , ReLu , stride 1	batch normalization , momentum .8
9×9 convolution ,256 , tanh , stride 1	3×3 convolution , LeakyReLu , stride 2
	batch normalization , momentum .8
	Fully connected ,1024 , LeakyReLu
	Fully connected,1,sigmoid

Table 5.1: Super resolution GAN Architecture

nique. The maximum number of epochs is set to 2000 which is even reduced with the patience parameter in early stopping. The patience parameter is set to 10. The optimizer used is Adam and the learning rate is set to 0.002. The architecture of deep networks involved in the experiment is given below.

5.3 Results

As epoch increases the generated image will give more clear images.By training large number of epoch , the generator have start generating clear image . Lets have a look :

- After 1000 epoch the generated image as shown in the first row:
- after 1500 epoch the generated image as shown in the second row:
- After 5000 epoch the generated image as shown in the third row:

to generate a photo realistic image , train the network of 30000 to 40000 epoch.

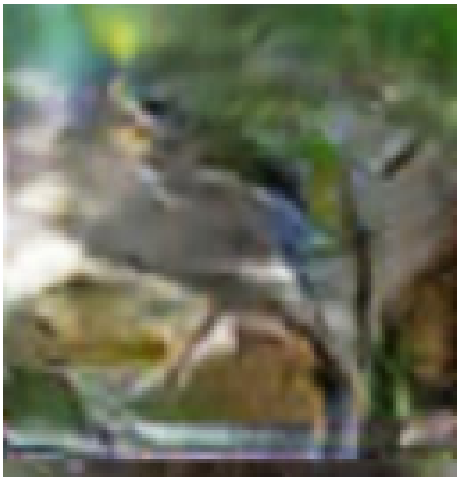


Figure 5.3: low resolution.

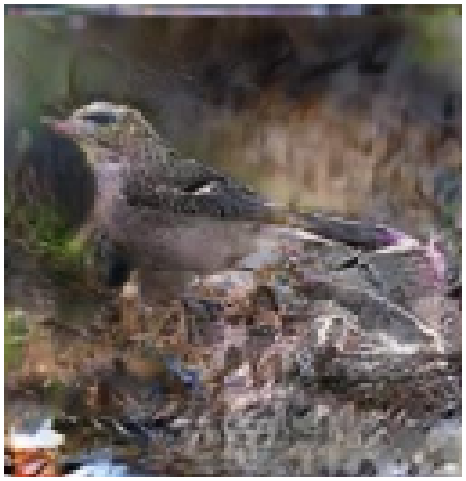


Figure 5.4: Generated Image

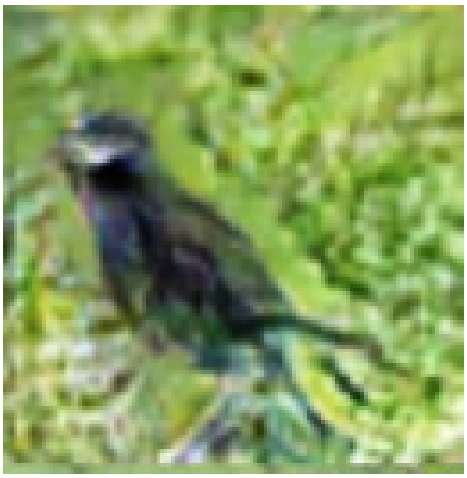


Figure 5.5: low resolution

Figure 5.6: Generated Image

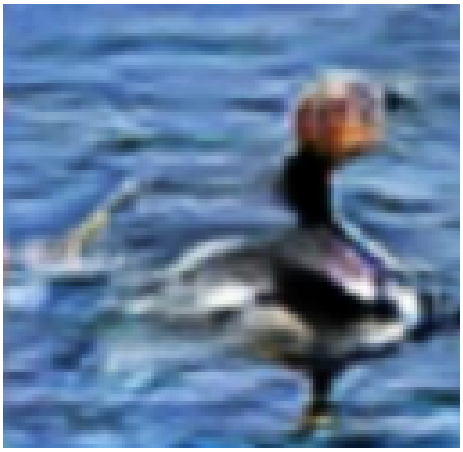


Figure 5.7: low resolution

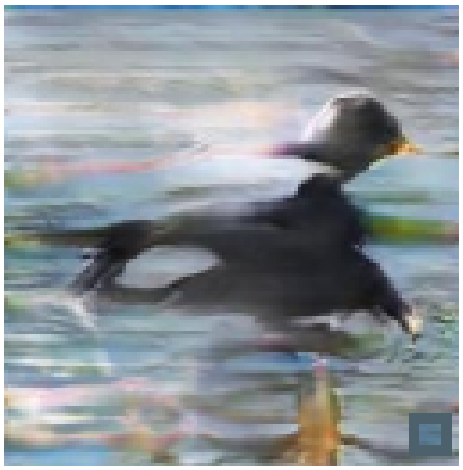


Figure 5.8: Generated Image

To visualize the losses for the training , we use tensorboard that gives the plotting of every loss that used in SRGAN.WE set the learning rate of .002 in both generator and the discriminator network .we set the maximum iteration of 200000.In generator network the loss is the content loss and the adversarial loss in SRGAN.

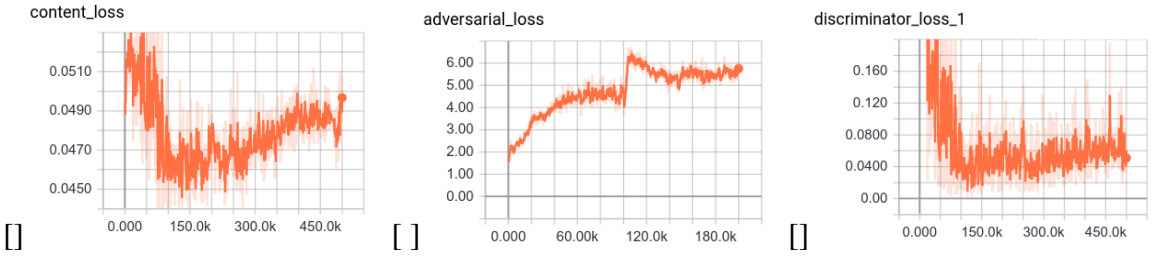


Figure 5.9: graph between loss of SRGAN vs iteration

5.4 Conclusion

In this chapter, we began by introducing high resolution text to image synthesis. Then, we described detail architecture of discriminator and the generator network of SRGAN .we also described every loss function used in SRGAN. Then, we carried out the required setup for the project. After that, we implemented the project in Keras and optimize the trained model by hyperparameter optimization techniques.For visualizing the loss, we use tensorboard.

Chapter 6
Conclusion And Future Work

6.1 Conclusion

This thesis mainly focused on overcoming a few limitations faced by the GAN based approaches for Text to Image Synthesis. Initially, a study was done to gain knowledge about various Text to Image synthesis techniques in traditional deep learning as well as machine learning algorithms. Extensive research was done to understand different criteria used to query labels in text to image synthesis and its effects on the results. We did a thorough literature survey on various pool and generation based Text to Image synthesis approaches.

Usage of Generative Adversarial Networks (GAN) is one of the recent approaches for Text to Image Synthesis. Implementation of GAN-CLS was done and the result was analyzed. The images generated from GAN-CLS were uncertain but not visually interpretable which would make it very difficult to understand. We tried using different variants of GAN such as DCGAN and WGAN and analyzed how the quality of results was affected.

We proposed a modification of the loss function which generated images that are uncertain and are sampled from the high-density regions of the data distribution, hoping for better visual interpretability of images. The accuracy of the proposed image compared to the GAN-CLS on birds dataset and the image generates was interpretable. Thus overcoming one of the disadvantages of GAN-CLS. Next, we approached a high-resolution image of text to image synthesis using SRGAN.

6.2 Future Work

In the text to image synthesis, we just generate the high-resolution image from the low-resolution image. Because of this, we cannot find the information that lost in the low-resolution image. Therefore in the future, we will find how the conditional Wasserstein loss function can be used in a more advanced model.

we will also try a model named progressive growing conditional Wasserstein gan for a better image.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[3] Y. Bengio, "The consciousness prior," *arXiv e-prints*, vol. 1709.08568, Sep. 2017. [Online]. Available: https://arxiv.org/abs/1709.08568

[4] L. M. Alec Radford and S. Chintala., "Unsupervised representationlearning with deep convolutional generative adversarial networks," *CoRR, abs/1511.06434,*, 2015.

[5] A. F. Henning Petzka and D. Lukovnikov., "On the regularization ofwasserstein gans." *In International Conference on Learning Representations*, 2018.

[6] B. S. Scott Reed, Zeynep Akata and H. Lee., "Learning deep rep-resentations of fine-grained visual descriptions." *In IEEE Computer Vision and PatternRecognition*, 2016.

[7] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan++: Realistic image synthesis with stacked generative adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1947–1962, 2019.

[8] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5908–5916.

[9] 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," in

2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1316–1324.

[10] T. S. et al., "Improved techniques for training gans," *n: NIPS.*, 2016.

[11] D. P. Kingma and J. Ba., "Adam: A method for stochastic optimization." *CoRR, abs/1412.6980*, 2014.

[12] C. Ledig, L. Theis, F. Huszár, J. A. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, 2017.

[13] S. R. Kaiming He, Xiangyu Zhang and J. Sun, "Identity mappings indeep residual networks," *Identity mappings indeep residual networks*, 2016.

[14] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.