Text to Image Synthesis using Stack GAN

Under the guidance of: **Dr. V. Dinesh Reddy,** Assistant Professor, CSE, SRM university - AP

Desu Yasaswini
Department of Computer Engineering
SRM UNIVERSITY, AP
yasaswini_desu@srmap.edu.in

Sindhu Medarametla
Department of Computer Engineering
SRM UNIVERSITY, AP
sindhu_p@srmap.edu.in

Vamsee Chilukuri
Department of Computer Engineering
SRM UNIVERSITY, AP
vamsee_chilukuri@srmap.edu.in

Girish Neelissetti
Department of Computer Engineering
SRM UNIVERSITY, AP
girish_neelisetti@srmap.edu.in

Abstract

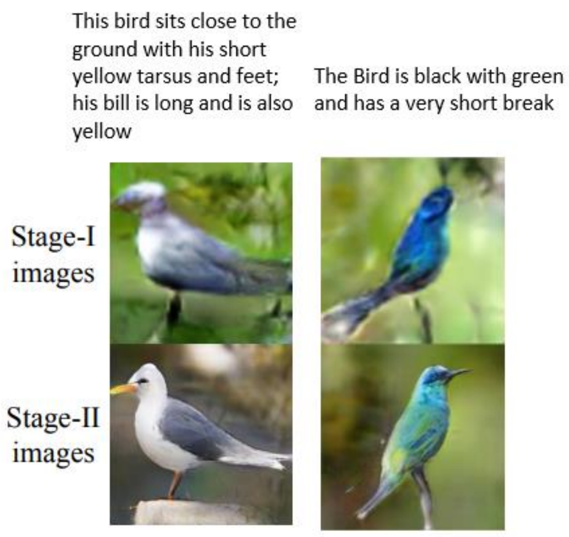
Automatically generating naturalistic images from the text would be quite valuable and interesting, these days AI is currently improving a lot, but this synthesis is still a long way off. It is one of the arduous problems in the computer vision (open CV) which also had many practical applications. Many recent studies which are based on text-to-image synthesis can roughly solve half part of the problem, but they fail to contain all the necessary details in it. In this study, we present this problem using stacked Generative Adversarial Networks (Stack GAN) to produce photo-realistic images based on the given text. In recent times Stack GAN generated highly appealing imagery in specialised categories such as room interiors, album covers etc. The Stage-I GAN creates low-resolution images by depicting the rudimentary/basic shape and colours of a scene based on a text description. Stage-II GAN generates high-resolution images with photo-realistic features using Stage-I findings and the text description as inputs. The output which are generated by this technique are credible than many other techniques which are already in use. More importantly Stack GAN produces 256 x 256 images based on only text descriptions, while the existing algorithms produces 128 x 128. To show the advancement and effectiveness of this technique, we have conducted an experiment on CUB dataset, which contain required object presence used for text-image synthesis.

Keywords

Generative Adversarial Networks (Stack GAN), text-to-image synthesis, generative models, multi distribution approximation. CUB dataset, generator, discriminator, inception score, ADAM Solver, Evaluation Metrics.

1. Introduction

Generative Adversarial Networks (GAN) were introduced by Goodfellow. They are comprised of a generator and a discriminator, which work parallelly with a competitive goal. The generator is designed in such a way that it keeps on generating the samples towards the original data distribution to bypass or fool the discriminator. Whereas discriminator is designed in such a way that it always tries to identify real data samples over the generated fake samples. We are interested to work on translating the single-sentence text into its equivalent image pixels. For example: *"A white Bird with black crown and yellow peak"* GANs have numerous applications in the real world like photo editing, image quality enhancement, computer-assisted design, etc. But they failed to spawn the high-resolution images using the text descriptions.


This bird sits close to the ground with his short yellow tarsus and feet; his bill is long and is also yellow
The Bird is black with green and has a very short break
Stage-I images
Stage-II images

Recently, GANs gave a positive outcome to this hurdle by generating 64 x 64 images based on given text inputs. However, the generated images were also lacking in detailing parts like many of the image's pixels failed to be synthesized with high-quality examples: 128 x 128 and 256 x 256). So, to handle these circumstances we tried to decompose the problem into two more subproblems where we can manage to produce high synthesized images with the assistance of **Stack Generative Adversarial Network**. Initially, a low-resolution image will get generated with the assistance of Stage – 1 GAN where the generator will produce a rudimentary/basic shape and primary colours of the object image according to the given text input and generate the background area. The generated fake sample object images will have many defects, for example, the object's parts may not be formed properly, and the shape of the object can be experienced with distortions.

So, to enhance our image quality and rectify all the defects we are going to use Stage-2 Stack GAN which produces a high-synthesized image as output (256 x 256). Now, Stage 2 Stack GAN will only focus on delivering the high-quality image by adding more details and rectifying the defects of the existing image. So, we thought this could be a easy method to develop a high-resolution images from scratch. The main idea of writing this paper is to implement Stacked Generative Adversarial Network which can produce High Synthesized images. When we try to compare the existing works related to text-image generation using different methodologies our GAN generated more realistic images.

2. Related Work

Image synthesis and image modelling is an important part of computer vision. We know that, in the recent times deep learning techniques emerged on a large scale and there was tremendous development in image generation. Dosovitskiy [1] has developed a deconvolutional neural network that can generate a 3D view of various chairs, cars, and tables with the help of respective datasets. They implemented this neural network with standard backpropagation to handle Euclidian reconstruction error. They implemented this neural network with standard backpropagation to handle Euclidian reconstruction error. This convolutional neural network is developed in such a manner that it can handle the intermediate images which permits smooth morph between intermediate stages. The architecture of this neural network is pretty simple but it is showcasing the complex behaviour

Generating a 3D object from a lone source image can be a very difficult task for both graphics and vision synthesizing of the 3D image. J. Yang [2] and some other groups came up with deterministic Neural networks for 3d view synthesis with a recurrent convolutional encoder-decoder network whose task is to provide the rotated objects starting from a single image. It allows to get long-term dependencies in addition to sequence of transformations. D.P. Kingma [3] has come up with Auto-Encoding Variational Bayes which can compute the probabilistic graphical models to achieve the maximum lower bound of the likelihood. In continuation to the work done on VAE, Gregor [4] proposed a DRAW model which can be applied to existing VAE for generating realistic images.

Generative Adversarial Networks performed well in generating sharper object samples, but the main issue with GANs was their instability during training dynamics. T. Salimans [5] has proposed various techniques to improve GANs training and has shown promising results in generating room interiors and synthesizing human faces. The generative adversarial neural network was lacking with stable data training and lack of evaluation metrics. So, to improve the training of GANs There are numerous techniques like Minibatch discrimination, Feature matching, historical averaging and virtual batch normalization. And coming to the evaluation metrics we have the Inception score which provides a partial solution for the problems with GANs. J. Zhao [6] has proposed a new technique called Energy-based GAN for making the training part even more stable.

Emily L. Denton [7] implemented deep generative neural network models with the aid of the Laplacian pyramid technique of Adversarial Networks. This technique has shown adequate results in generating high-quality image samples. To generate images in a course-to-fine manner they extended their model architecture by incorporating cascades of convolutional neural networks called convents with Laplacian pyramid technique. They trained distinct generative convnet models at every level of the Laplacian pyramid with the help of Generative Adversarial Nets (GAN). Our model's samples are of much greater quality than those from other models. Their CIFAR10 samples were mistaken for real images around 40% of the time in a quantitative assessment by human evaluators, compared to 10% for GAN samples. K. Gregor [4] introduces the Deep Recurrent Attentive Writer (DRAW) neural network architecture for image generation, it mimics the foveation of human eye, with a sequential variational auto-encoding framework. This can be further edited using Andrew Brock [8] neural photo editing approach with introspective adversarial networks, because most it is one of the most feasible applications beyond image generation. It is an interface that holds the power of generative neural networks to make large changes to existing images. It works based on hybridization of VAN and GAN.

Kaiming He [9] proposed a new framework called Residual learning, which assisted in effortless training of deep neural networks when compared to previous models. Instead of using the unreferenced functions they explicitly redefined the input layers as learning residual functions. This can be showcased as comprehensive support for these residual networks that can be optimized easily and can obtain better accuracy by considerably increasing the depth. CARL DOERSCH [10] presented a tutorial on variational autoencoders, which come out to be one of the most concrete approaches to unsupervised learning for intricate distributions. These are most used, VAEs are built on the top standard functional approximations, and can be trained with gradient descent approach.

3. Background

Stacked Generative Adversarial Networks (Stack GAN)

Intending to produce high resolution images with photo-realistic details, we use stacked GAN. GAN is comprised of generator and discriminator which works parallelly with a competitive goal.

- **Generator G**: optimized to generate images that are difficult for the discriminator D to evolve from real images.
- **Discriminator D**: optimized to distinguish real images from synthetic images generated by G.

It decomposes the text-to-image generation into two stages:

- **Stage-I GAN:** It gives the basic shape and colours of the object based on the text given; this stage draws the background layout which yields a low-resolution image. It also provides the structural information of the image. This stage usually generates 64 x 64 images.
- **Stage-II GAN:** It alters and corrects the defects in the low-resolution image and gives the complete detailed image of the object by analysing the text, producing a high-resolution image-realistic image. It uses the output of stage-1 as input, this stage samples to 256 x 256.

3.1. Preliminaries

GAN is composed of G and D. G is optimized to duplicate the true data p_{data} by producing the images that are difficult for D to differentiate from real images. D, on the other hand, is tuned to distinguish between actual and synthetic images generated by G.

Loss Functions

Scores from the Discriminator D:

$$\mathbf{S_r \leftarrow D(x,h)\ \{real\ image,\ right\ text\}}$$

$$\mathbf{S_w \leftarrow D(x, \hat{h})\ \{real\ image,\ wrong\ text\}}$$

$$\mathbf{S_f \leftarrow D(\hat{x},h)\ \{fake\ image,\ right\ text\}}$$

This entire training is similar to a min-max game with the below function,

$$\min_{G} \max_{D} V(D,G) = E_{X\sim Pdata}[\log D(x)] + E_{z\sim Pz}[\log(1-D(G(z)))] \quad (1)$$

where, x = real image from true data distribution $p_{data.}$

z = noise vector samples from distribution p_z.

3.2. Stage-I GAN:

As discussed above, the stage-1 GAN only focuses on developing the rough shape of the object image with primary colours. By the end of this stage 1, we can generate a low-resolution image of resolution 64 x 64. But before implementing stage-1 we have to perform conditioning argumentation on the text description. Previously [22], the work done on GAN used non-linear text embeddings to generate latent variables for the generator which forms high dimensional data.

It is a difficult job to train our generator using the high-dimensional text of smaller size data. So instead of using the non-linear transformation technique, we are going to use conditioning argumentation to train our generator. Initially, we will pass text description *t* to the encoder which gives ϕt text embeddings to the generator.

In stage-1, we going to select the random sample variables from the gaussian distribution $N(\mu(\phi t), \Sigma(\phi t)),$ where $\mu(\phi t)$ is the mean of the data and the diagonal covariance matrix is $\Sigma(\phi t)$. In addition to these two functions, we are going to add some of the regulations to the existing notation. Which maintains robustness and avoids overfitting of the model during generator training.

$$D_{KL}(N(\mu(\phi_t), \Sigma(\phi_t)) \,||\, N(0, I)) \qquad (2)$$

The above equation refers to Kullback-Leibler divergence (difference between standard gaussian distribution and conditioning gaussian distribution) So, the generator and discriminator will keep on trying to attain the minimum L_{Go} in Eq (3) and maximum L_{Do} *in Eq (4)* respectively.

$$L_{G0} = E_{z\sim pz\,,t\sim Pdata}\,[\log(1 - D_0(G_0(z, c_0), \phi t))] + \lambda D_{KL}(N(\mu(\phi_t), \Sigma(\phi_t)) \,||\, N(0, I)) \quad (3)$$

$$L_{D0} = E(I_0,t)\sim P_{data}\,[\log D_0(I_0, \phi t)] + E_{z\sim pz\,,t\sim Pdata}\,[\log(1 - D_0(G_0(z, c_0), \phi t))] \ (4)$$

Where the text description t is getting from the true distribution P_{data}. I_0 is the real image, Z is the noise vector which got from randomly sampled gaussian distribution data P_z. λ is the parameter which regularize the balance in Kullback-Leibler equation and gaussian distribution equation, we generally use λ=1. Φt is the text embeddings and C_0 is the gaussian conditioning variable

Model Architecture: Initially the text embeddings Φt are given as input via a connected layer to the generator to generate μ_0 and σ_0 for the Gaussian distribution equation. Thereafter C0 will compute the Ng dimensional vector using the formulae $c_0 = \mu_0 + \sigma_0 \odot \varepsilon$ (Where $\odot$ represents the multiplication of every element and ε will be in the range of 0 – 1). Then Ng dimensional vector is concatenated with N_z dimensional noise vector to compute W_0 X H_0 of the image's samples.

Now for Discriminator, the existing text embeddings Φt are compressed into to Nd dimensional vector using a fully connected tensor layer to form M_d X M_d X N_d tensor. Meanwhile, the discriminator will keep on blocking the image sample which was given in the form of series down-sampling blocks until the image sample has M_d X M_d dimensions. So, a new tensor will be created by adding an image filter map. 1 X 1 convolutional layer input will be given to the new tensor to compute features of the image and text. In the end, a fully-connected layer with a single node will give us the decision score.

3.3. Stage-II GAN:

Stage-II GAN is constructed upon Stage-I GAN to come up with a photo-realistic high-resolution images. Low resolution images that are produced by Stage-I GAN lack vivid object components and on addition with it the generated images will be dis-oriented or distorted, within the Stage-II of GAN methodology we tend to contemplate these low-resolution images that are generated within the early stages.

And also, the inputs i.e., the text embeddings to correct the mistakes and also the distortions within the Stage-I results are associated to train the model to gather and collage the data of the unheeded text input to put forward with the more realistic high-resolution images which incorporates the complete information given as an input in the beginning.

Conditioning on the low resolution sample s0 and Gaussian latent variables c, the discriminator D and generator G in Stage-II GAN is trained by alternatively maximizing L_D in Eq.(5) and minimizing L_G in Eq. (6),

$$L_D = E_{(I,t)\sim Pdata} [\log(D_0(I,\phi t))] + E_{s0\sim PG0,t\sim Pdata}[\log(1-D(G(s0,c),\phi t))) \quad (5)$$

$$L_G = E_{s0\sim PG0,t\sim Pdata}[\log(1-D(G(s0,c),\phi t))) + \lambda D_{KL}(N(\mu(\phi_t), \Sigma(\phi_t)) \,||\, N(0, I)), \quad (6)$$

Where s0 = G0(z, c0) is produced in the previous stage i.e., by Stage- I GAN, at which we haven't considered the random noise factor z on assuming that the uncertainty has already preserved in s0, which diverged it form the original GAN formulation. Gaussian conditioning variable c which is used in the Stage-II and c0 used in Stage-I GAN inherits the same pre-trained text encoder, which at the end produce the same text embeddings ϕt. Perhaps, they make use of separate 100% integrated layers for the productions of various averages and standard deviations. Hence, Stage-II follow the above procedure to learn how to capture useful information in the text embeddings from the results of the Stage-I GAN.

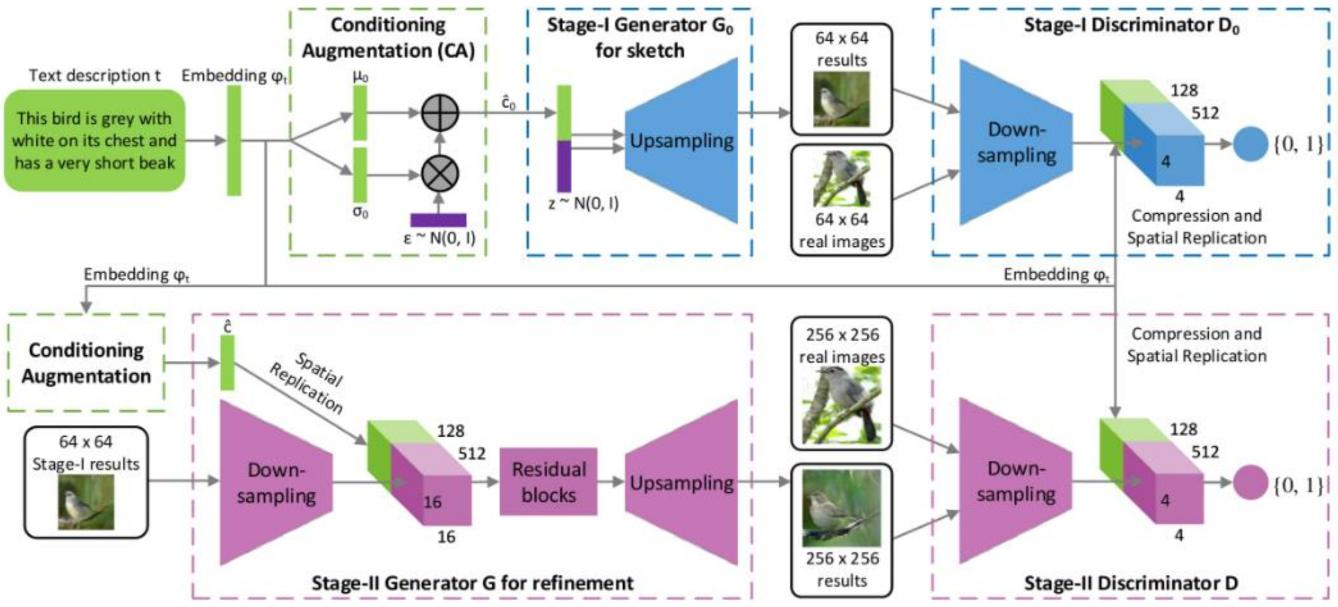
Model Architecture: For the Stage-II generator, ϕt is employed to come up with the Gaussian conditioning vector c of dimension N_g similar to the Stage-I, that is dimensionally derived to form a M_g x M_g x N_g tensor. The sample that's created from Stage-I GAN s0 is updated into many down-sampling blocks till its spatial size of M_g x M_g. The image filter map and also the text tensor are joined on the channel dimension. The resultant is passed into various residual blocks [11,9] combinedly encode the image and also the text attributes, and eventually a sequence of up-sampling blocks is utilized to create a W x H image.

The structure of discriminator is a close design with a little difference of having extra down-sampling blocks of the discriminator used in the Stage-I GAN generation, as the image size is comparatively big in this stage. Rather than following the naive discriminator, we can use the matching-aware discriminator prosed by Reed et al. for both the stages. As a part of training of the model the discriminator takes real images and their respective text as positive sample pairs, where as a negative sample pair are divided into 2 groups and are given as inputs to the discriminator. At start we input real images with the ill-assorted text embeddings, and the second are synthesized images with conditioning text embeddings.

4. Implementation details

Before going to implement the Stack GAN we need to convert the user text input into embedding using pre-defined character embeddings. There are different types of text embeddings. In this paper we tried to implement two dissimilar types of text embeddings they are char CNN text embeddings and Bidirectional Encoder Representations from Transformers.

We know the Stacked Generative Adversarial Neural Network is divided into two stages. Initially, we will



convert the user text input into embeddings with the help of any pre-trained character embeddings. Then, we pass the existing text embeddings to conditional Augmentation (CA), and then the Augmented input will be again passed as input for the Stage-1 Generator, which produces 64*64 resolution images.

The up-sampling blocks contain nearest neighbours followed by 3 x 3 convolution. We apply different normalization techniques for every convolution except the last layer. The down-sampling consists of 4 x 4 convolutions and performs the normalization except for the first layer. After completing the Stage-1 implementation we will store the stage-1 model weights and results we will start implementing Stage-2 which produces 256 x 256 resolution images.

By default, we initialize the Ng = Nd = 128, Mg =16, Md =4, W0 = H0 = 64, W = H = 256, and Learning rate = 0.0002. For training stage 1 and stage 2 we first iteratively train the discriminator (D0) and Generator (G0) for 800 epochs by fixing Stage-2 GAN. Thereafter we Iteratively train the Discriminator and Generator of Stage-2 for 600 epochs by keeping stage-1 constant. All the neural networks were trained with the assistance of ADAM Solver (which optimizes the neural networks in terms of computation, memory, and large datasets) with a batch size of 64, and learning rate will be decayed by half of its preceding values for every 100 epochs.

5. Experiments

To substantiate our methodology, we performed large-scale Quantitative and qualitative evaluations on the Birds dataset (Caltech – UCSD) that is widely used in text-to-image synthesis. We tried many existing base models to redesign our model architecture. We have implemented only stage 1 GAN for the first baseline to check whether the Stack structure performance is generating 64 x 64 resolution images. Then we tried to implement the existing GAN for generating 128 x 128 images but they were less optimal when compared to 64 x 64 images. We tried to give input of text at both stages but it was not shown a promising result.

CUB [19] (Birds dataset) consists of 200 varieties of bird species with 11,788 sample images. The object image ratios are < 0.5 so as a part of pre-processing the data we crop all the images to make sure the boundary boxes > 0.75. We divided the CUB dataset into two parts, training (with 150 classes) and testing (50 test classes). **Evaluation Metrics.** It is quite a difficult job to analyze the performance of Generative Adversarial Neural Networks. The best method to record the performance of GAN models is by asking Human volunteers to determine the image quality and other parameters. Recently, [5] a new type of score was introduced to evaluate the GANs called *the Inception Score* for quantitative Evaluation.

$$I = \exp (Ex\ DKL\ (p\ (y \mid x) \parallel p(y))),$$

Where *x* refers to our generated sample and *y* refers to the predicted label by the model. The divination beyond using this metric is a perfect model can produce better accuracy. So, the KL divergence between conditional and marginal distribution must be huge. In our experiments, we performed this process with different text inputs and recorded the average inception score of our model. To crossvalidate our model with the existing GANs, we gathered multiple inception scores of various models which can perform the text to image synthesis. Although inception scores are not 100% precise, we tried to verify the outputs by showcasing them to multiple humans to correlate with their perspectives and they gave satisfactory feedback.

6. Quantitative and Qualitative Results

We tried various inputs to compare the inception score of our proposed Stack GAN model with other on CUB (Birds dataset). You can look a Table 1 to get a complete overview of our inception scores comparison with various GAN Models.

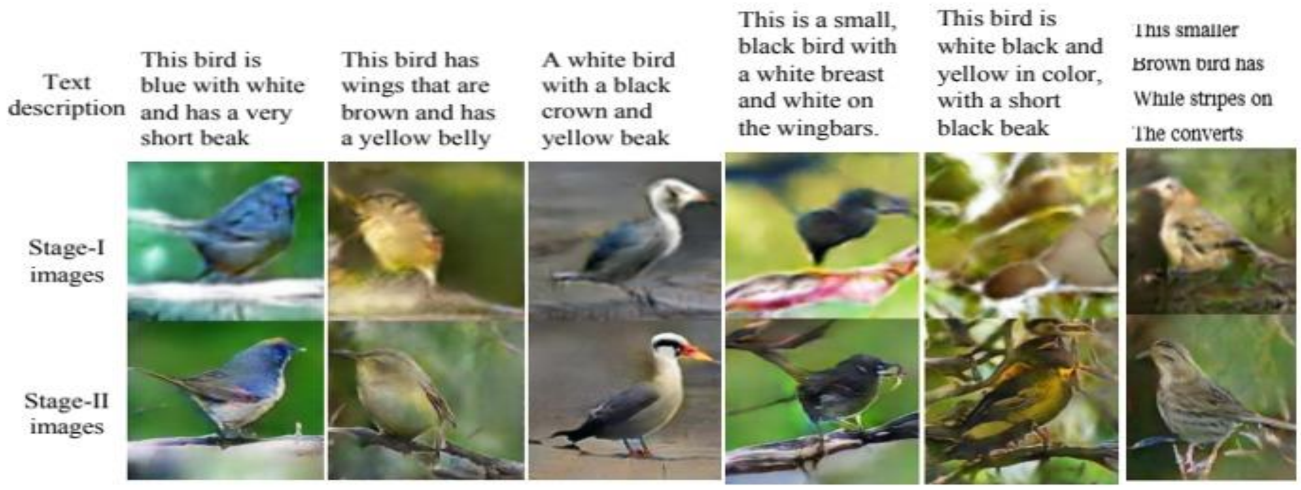
Method	**Inception Scores**
GAN – INT – CLS [13]	2.88 ± 0.4
Attn – GAN [15]	2.36 ± 0.21
Mirror GAN [16]	2.64 ± 0.41
DM – GAN [17]	3.23 ±0.23
OP – GAN [18]	2.78 ± 0.12
GAWW N [14]	3.62 ± 0.7
Our Stack GAN	3.70 ± 0.4

Our Stack GAN achieves best inception score on CUB dataset than other methods. Comparing with GAN-INT-CLS[13], our Stack GAN achieved 26.29% improvement in terms of inception score on CUB dataset.

In the above figure, the 64 x 64 samples are produced by GAN-INT-CLS can only mirror the generate colour and bird layout. But we observed that their results lack some parts of the birds like legs, feathers etc and the plausible details are most of the images were looking in non-realistic manner.

We have observed that GAWWAN [14] obtained somewhat better results on CUB dataset, which is slightly lesser than Stack GAN. But produced high resolution images than GAN-INT-CLS. But the only problem with GAWWAN is that it failed to produce the believable images when the text input is given. But the thing is, in our Stack GAN, the stage-1 produced blurry images with low-resolution, it also includes missing details and various defects. And in Stage-2, it generates high resolution images and it brought about 4 x higher resolution images with more conclusive details to enhance and mirror the corresponding text details. Stage-II GAN fills in the details where Stage-I GAN has generated realistic forms and colours. For example, in Figure 5, in the first column, with a Stage-II GAN concentrates on providing the small beak and white colour stated in the text as a suitable Stage-I outcome. Details for the tail and legs are also included. In any alternative case, Stage-II photos have various levels of detail added to them. In many other circumstances, by processing the text description, Stage-II GAN can address the flaws in Stage-I results again. In all other comparison studies, the results are not as good as Stack GAN.

Importantly, the Stack GAN achieves good results by capturing the intricate underlying language-image interactions rather than just memorising training data. We extract the actual visual features from our output produced images and all the trained images by Stage-II discriminator D of our model. For all the output images, its nearest neighbours from the training set can be reclaimed. By analysing the training images, we can wind up that the generated images in the drive have some similar characteristics with the retrieved training images but are actually different.


Text description
This bird is blue with white and has a very short beak
This bird has wings that are brown and has a yellow belly
A white bird with a black crown and yellow beak
This is a small, black bird with a white breast and white on the wingbars.
This bird is white black and yellow in color, with a short black beak
This smaller Brown bird has While stripes on The converts
Stage-I images
Stage-II images

7. Conclusion

In this paper, we proposed Stacked Generative Adversarial Networks for text-to-image synthesis. This methodology is very helpful in synthesizing high-quality images. We divided this complex implementation part into two chunks where Stage-1 Completely focuses on developing the outline and primary colours of the image. Whereas, the Stage-2 GAN will completely focus on image quality enhancement like adding more photorealistic details by rectifying the defects. We have performed an extensive qualitative and quantitative analysis to illustrate the performance of our model. We computed the Inception Score as a part of evaluation metrics. When compared with the other exiting GAN models our stack GAN has shown promising results in generating higher resolution images (256 x 256) with more photo–realistic details.

References

1. A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In CVPR, 2015.

2. J. Yang, S. E. Reed, M. Yang, and H. Lee. Weakly [1] supervised disentangling with recurrent transformations for 3d view synthesis. In NIPS, 2015.

3. D. P. Kingma and M. Welling. Auto-encoding variational bayes. In ICLR, 2014.

4. K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In ICML, 2015

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

6. J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. arXiv:1609.03126, 2016.

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

8. Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. arXiv: 1612.03242vl, 2016

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

10. C. Doersch. Tutorial on variational autoencoders. arXiv:1606.05908, 2016.

11. A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. arXiv:1609.07093, 2016.

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

13. S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In NIPS, 2016.

14. S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In ICML, 2016.

15. 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 Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1316–1324.

16. T. Qiao, J. Zhang, D. Xu, and D. Tao, “Mirrorgan: Learning text-toimage generation by redescription,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 1505–1514.

17. M. Zhu, P. Pan, W. Chen, and Y. Yang, “Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 5802–5810.

18. G. Yin, B. Liu, L. Sheng, N. Yu, X. Wang, and J. Shao, “Semantics disentangling for text-to-image generation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2327–2336.

19. S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In ICML, 2016.