Text To Image Synthesis

Neural Networks and Reinforcement Learning Project

H. Vijaya Sharvani (IMT2014022),
Nikunj Gupta (IMT2014037),
Dakshayani Vadari (IMT2014061)

December 7, 2018

Contents

1 Problem Statement

One of the most challenging problems in the world of Computer Vision is synthesizing high-quality images from text descriptions. No doubt, this is interesting and useful, but current AI systems are far from this goal. In the recent years, powerful neural network architectures like GANs (Generative Adversarial Networks) have been found to generate good results.[1] Samples generated by existing text-to-image approaches can roughly reflect the meaning of the given descriptions, but they fail to contain necessary details and vivid object parts.[2] Through this project we wanted to explore architectures that could help us achieve our task of **generating images from given text descriptions**. Generating photo-realistic images from text has tremendous applications, including photo-editing, computer-aided design, etc.

1.1 Example

Text description: This white and yellow flower have thin white petals and a round yellow stamen.

Corresponding Generated Image:

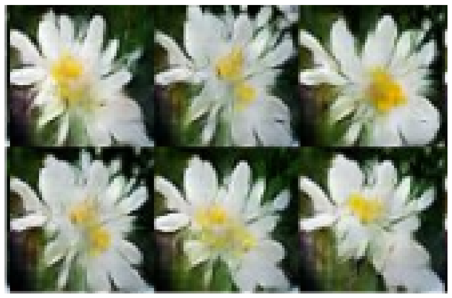


Figure 1: Generated Images

2 Dataset Description

2.1 Overview

Our results are presented on the the Oxford-102 dataset of flower images having 8,189 images of flowers from 102 different categories. The dataset has been created with flowers chosen to be commonly occurring in the United Kingdom. The images have large scale, pose and light variations. In addition, there are categories having large variations within the category and several very similar categories. The dataset is visualized using isomap with shape and colour features.[6]
Each image has ten text captions that describe the image of the flower in different ways. Each class consists of between 40 and 258 images.The details of the

categories and the number of images for each class can be found here: FLOWERS_IMAGES_LINK
5 captions were used for each image. The captions can be downloaded for the following FLOWERS_TEXT_LINK

2.2 Example Images from the Dataset

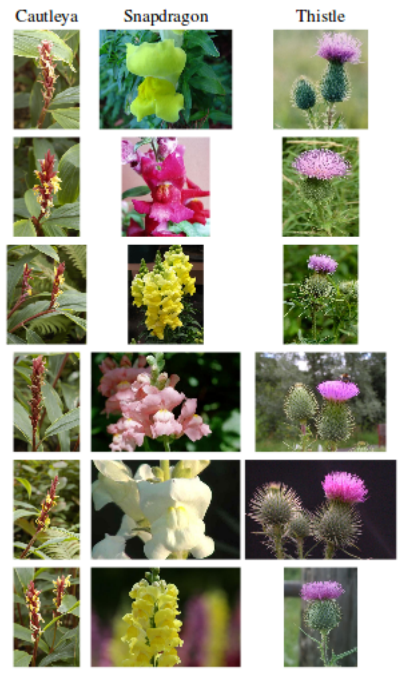
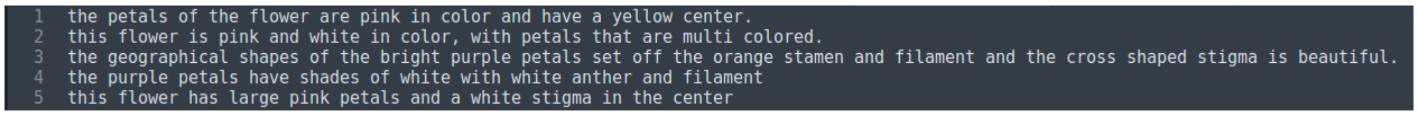


Figure 2: Example images from different classes

2.3 Examples of Text Descriptions for a given Image



```
the petals of the flower are pink in color and have a yellow center.
this flower is pink and white in color, with petals that are multi colored.
the geographical shapes of the bright purple petals set off the orange stamen and filament and the cross shaped stigma is beautiful.
the purple petals have shades of white with white anther and filament
this flower has large pink petals and a white stigma in the center
```

Figure 3: Sample Text Descriptions for one flower

3 Architecture Details

3.1 Generative Adversarial Networks[4]

The main idea behind generative adversarial networks is to learn two networks- a Generator network G which tries to generate images, and a Discriminator network D, which tries to distinguish between 'real' and 'fake' generated images. One can train these networks against each other in a min-max game where the generator seeks to maximally fool the discriminator while simultaneously the discriminator seeks to detect which examples are fake:

$$\min_{w_G} \max_{w_D} \mathbb{E}_{x \sim p_x(x)} [\log D(x; w_D)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z; w_G); w_D))]$$

Figure 4: Equation

where z is a latent "code" that is often sampled from a simple distribution (such as normal distribution). Conditional GAN is an extension of GAN where both generator and discriminator receive additional conditioning variables c, yielding G(z, c) and D(x, c). This formulation allows G to generate images conditioned on variables c.

3.2 Generative Adversarial Text-To-Image Synthesis[1]

Figure 5 shows the network architecture proposed by the authors of this paper. The paper's talks about training a deep convolutional generative adversarial network (DC-GAN) conditioned on text features. These text features are encoded by a hybrid character-level convolutional-recurrent neural network. Both the generator network G and the discriminator network D perform feed-forward inference conditioned on the text features. The encoded text description embedding is first compressed using a fully-connected layer to a small dimension followed by a leaky-ReLU and then concatenated to the noise vector z sampled in the Generator G. The following steps are same as in a generator netowrk in vanilla GAN; feed-forward through the deconvolutional network, generate a synthetic image conditioned on text query and noise sample.

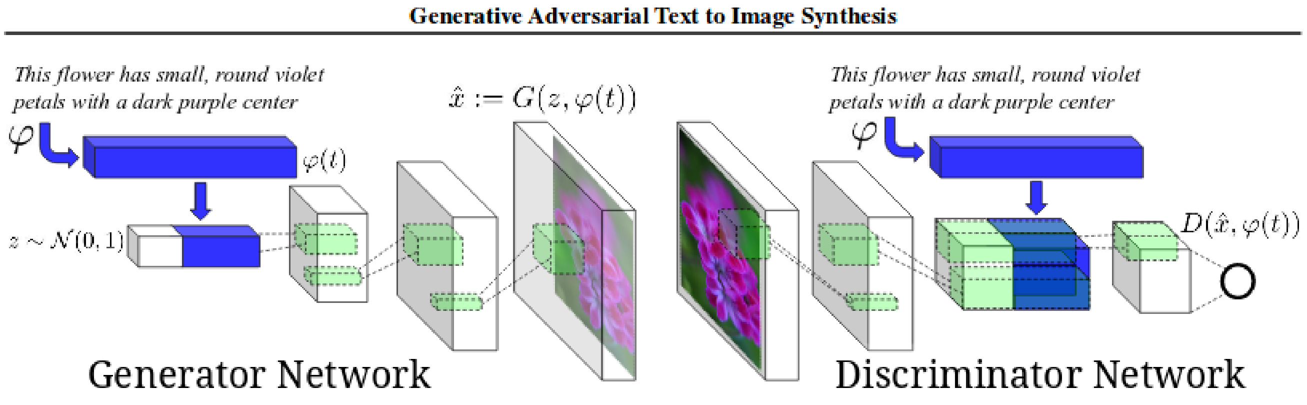
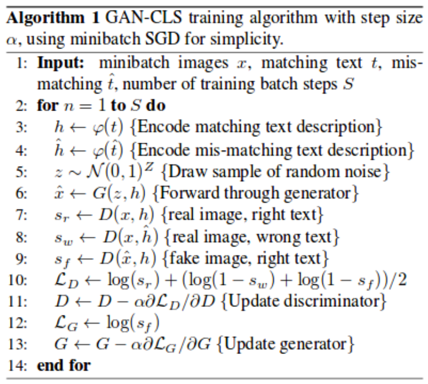

Generative Adversarial Text to Image Synthesis
This flower has small, round violet petals with a dark purple center
φ
$\varphi(t)$
$z \sim \mathcal{N}(0,1)$
$\hat{x} := G(z, \varphi(t))$
This flower has small, round violet petals with a dark purple center
φ
$D(\hat{x}, \varphi(t))$
Generator Network
Discriminator Network

Figure 5: Network Architecture

GAN-CLS

This is the first tweak proposed by the authors. The most straightforward way to train a conditional GAN is to view (text, image) pairs as joint observations and train the discriminator to judge pairs as real or fake. The discriminator has no explicit notion of whether real training images match the text embedding context. To account for this, in GAN-CLS, in addition to the real / fake inputs to the discriminator during training, a third type of input consisting of real images with mismatched text is added, which the discriminator must learn to score as fake. By learning to optimize image / text matching in addition to the image realism, the discriminator can provide an additional signal to the generator.



Algorithm 1 GAN-CLS training algorithm with step size α, using minibatch SGD for simplicity.

```
1: Input: minibatch images $x$, matching text $t$, mis-
   matching $\hat{t}$, number of training batch steps $S$
2: for $n = 1$ to $S$ do
3:     $h \leftarrow \varphi(t)$ {Encode matching text description}
4:     $\hat{h} \leftarrow \varphi(\hat{t})$ {Encode mis-matching text description}
5:     $z \sim \mathcal{N}(0,1)^Z$ {Draw sample of random noise}
6:     $\hat{x} \leftarrow G(z, h)$ {Forward through generator}
7:     $s_r \leftarrow D(x, h)$ {real image, right text}
8:     $s_w \leftarrow D(x, \hat{h})$ {real image, wrong text}
9:     $s_f \leftarrow D(\hat{x}, h)$ {fake image, right text}
10:    $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$
11:    $D \leftarrow D - \alpha\partial\mathcal{L}_D/\partial D$ {Update discriminator}
12:    $\mathcal{L}_G \leftarrow \log(s_f)$
13:    $G \leftarrow G - \alpha\partial\mathcal{L}_G/\partial G$ {Update generator}
14: end for
```

Figure 6: GAN-CLS Algorithm

GAN-INT

Papers have proved that deep netowrks learn representations in which interpolations between embedding pairs tend to be near the data manifold. Referencing these papers, the authors generated a large amount of additional text embeddings by simply interpolating between embeddings of training set captions. As the interpolated embeddings are synthetic, the discriminator D does not have corresponding "real" images and text pairs to train on. However, D learns to predict whether image and text pairs match or not.

3.3 StackGAN[2]

The aim here was to generate high-resolution images with photo-realistic details. The authors proposed an architecture where the process of generating images from text is decomposed into two stages as shown in Figure 7. The two stages are as follows:

- **Stage-I GAN:** The primitive shape and basic colors of the object (conditioned on the given text description) and the background layout from a random noise vector are drawn, yielding a low-resolution image.
- **Stage-II GAN:** The defects in the low-resolution image from Stage-I are corrected and details of the object by reading the text description again are given a finishing touch, producing a high- resolution photo-realistic image.

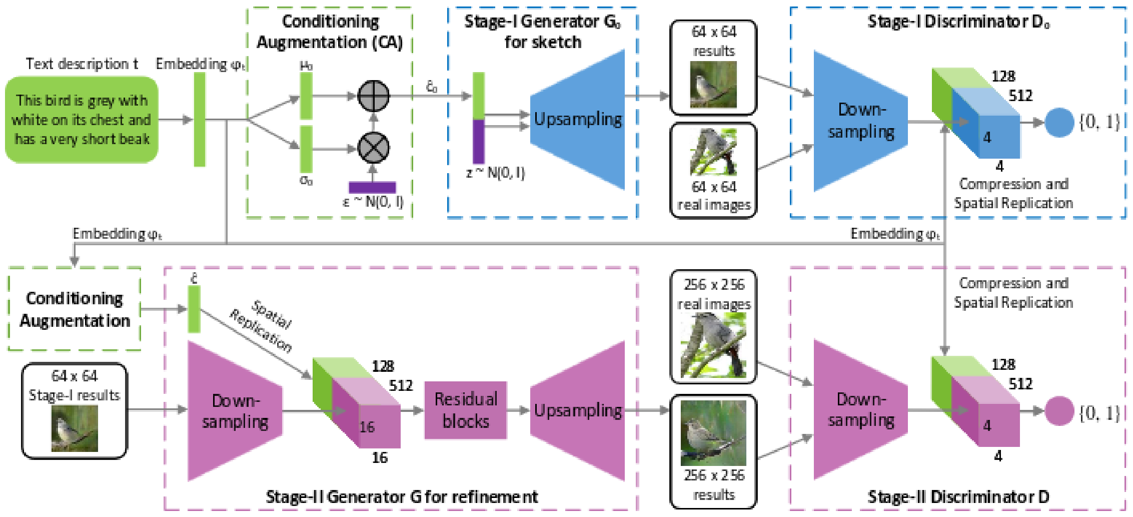


Figure 7: Network Architecture of StackGAN

3.4 StackGAN++ [5]

This is the version 2 of StackGAN talked about earlier. It is an advanced multi-stage generative adversarial network architecture consisting of multiple generators and multiple discriminators arranged in a tree-like structure. The architecture generates images at multiple scales for the same scene. Experiments demonstrate that this new proposed architecture significantly outperforms the other state-of-the-art methods in generating photo-realistic images. Figure 8 shows the architecture.

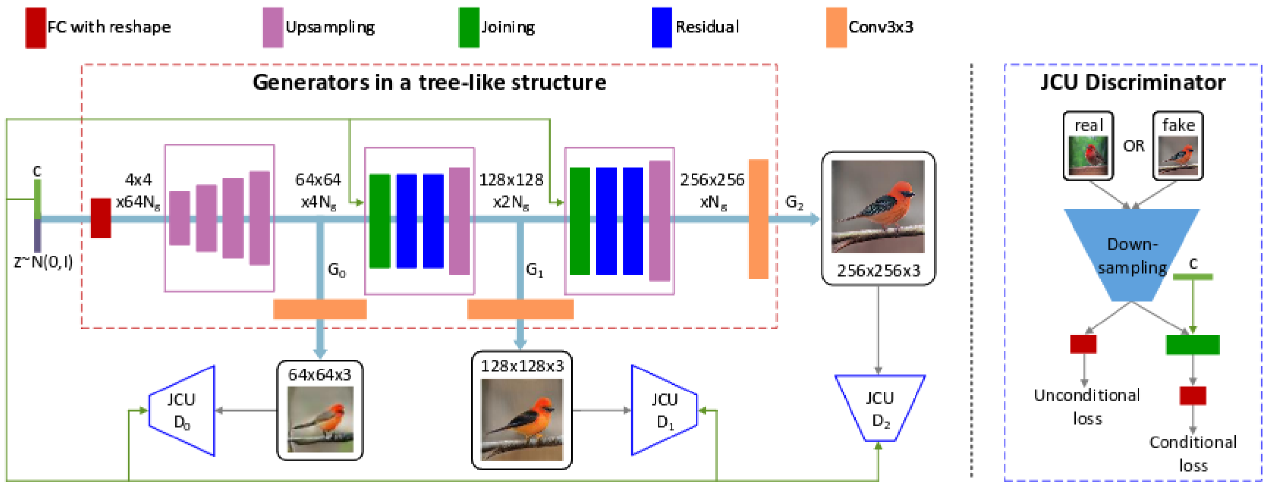


Figure 8: Network Architecture of StackGAN++

4 Implementation

We split the dataset into distinct training and test sets. Oxford-102 has 82 train+val and 20 test classes. We used 5 captions per image for training. During mini-batch selection for training we randomly pick an image view (e.g. crop, flip) of the image and one of the captions. For text features, we first pre-train a deep convolutional recurrent text encoder on structured joint embedding of text captions with 1,024-dimensional GoogLeNet image embedings. We used a hybrid of character-level ConvNet with a recurrent neural network (char-CNN-RNN). The reason for pre-training the text encoder was to increase the speed of training the other components for faster experimentation. The text encoder produced 1024 dimensional embeddings that were projected to 128 dimensions in both the generator and discriminator before depth concatenation into convolutional feature maps.[1]

The training image size was set to $32 \times 32 \times 3$. We have compressed the images from $500 \times 500 \times 3$ to the set size so that the training process would be fast.

The system configuration details are as follows:

- Intel®Core^{TM} i5-6200 CPU @ 2.30 GHz 2.40 GHz
- 64 bit OS, x64-based processor

We iteratively trained the GAN for 435 epochs. All networks are trained using SGD with batch size 128, a base learning rate of 0.0005, and ADAM solver with momentum 0.5. The generator noise was sampled from a 100-dimensional unit normal distribution.
In the discriminator, there are several convolutional layer, where convolution is performed with stride 2, along with spatial batch normalisation followed by leaky ReLU. The dimensionality of the description vector is reduced by using a separate fully connected layer. When the spatial dimension of the discriminator is 4 × 4, the description embedding is replicated spatially and concatenated depth-wise. Then a 1 × 1 convolution followed by rectification is performed and then a 4 × 4 convolution to compute the final score from the dicriminator D. Note that batch normalisation is performed on all convolutional layers. [1]

5 Results

In this section, we will describe the results, i.e., the images that have been generated using the test data. A few examples of text descriptions and their corresponding outputs that have been generated through our GAN can be seen in Figure 6. As you can see, the flower images that are produced (16 images in each picture) correspond to the text description accurately. One of the most straightforward and clear observations is that, the GAN gets the colours always correct - not only of the flowers, but also of leaves, anthers and stems. The model also produces images in accordance with the shape of petals as mentioned in the text descriptions. For example, in Figure 6, in the third image description, it is mentioned that 'petals are curved upward'.

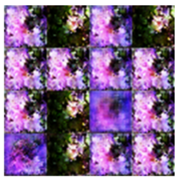
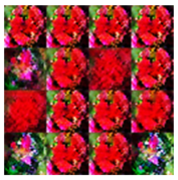
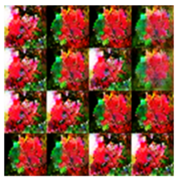
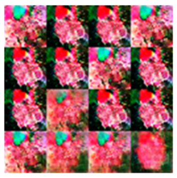
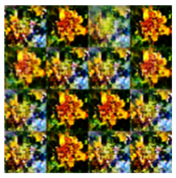
Text description	This is a flower with white petals and purple and white anthers.	The petals of the flower are maroon in colour and have green leaves.	This flower has petals that are pink and has yellow stamen	This flower is pink in colour, and has petals that are curled upward.	This flower is yellow and white in colour, with petals that are pointed at the tips.
DC - GAN					

Figure 9: Flower images generated using GAN

This method of evaluation is inspired from [1] and we understand that it is quite subjective to the viewer. Our observations is an attempt to be as objective as possible. The complete directory of the generated snapshots can be viewed in the following link: SNAPSHOTS.

6 Comments and Conclusions

In this project we make an attempt to explore techniques and architectures to achieve the goal of automatically synthesizing images from text descriptions. We implemented simple architectures like the GAN-CLS and played around with it a little to have our own conclusions of the results. We would like to mention here that the results which we have obtained for the given problem statement were on a very basic configuration of resources. Better results can be expected with higher configurations of resources like GPUs or TPUs. Though AI is catching up on quite a few domains, text to image synthesis probably still needs a few more years of extensive work to be able to get productionalized.

7 Acknowledgements

We would like to thank Prof. G Srinivasaraghavan for helping us throughout the project. This project was supported by our college- IIIT Bangalore.

8 References

[1] Reed, Scott, et al. "Generative adversarial text to image synthesis." arXiv preprint arXiv:1605.05396 (2016).

[2] Zhang, Han, et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks." arXiv preprint (2017).

[3] Nilsback, Maria-Elena, and Andrew Zisserman. "Automated flower classification over a large number of classes." Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on. IEEE, 2008.

[4] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[5] Zhang, Han, et al. "Stackgan++: Realistic image synthesis with stacked generative adversarial networks." arXiv preprint arXiv:1710.10916 (2017).

[6] Nilsback, Maria-Elena, and Andrew Zisserman. "Automated flower classification over a large number of classes." Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on. IEEE, 2008.