StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks

Report

Vinayak Gupta 9th July 2021

Introduction

Generating Realistic Images using just text descriptions is a challenging task. There has been a lot of research under this domain, but the model does not accurately represent the text description since it is unable to capture the important features from the text descriptions

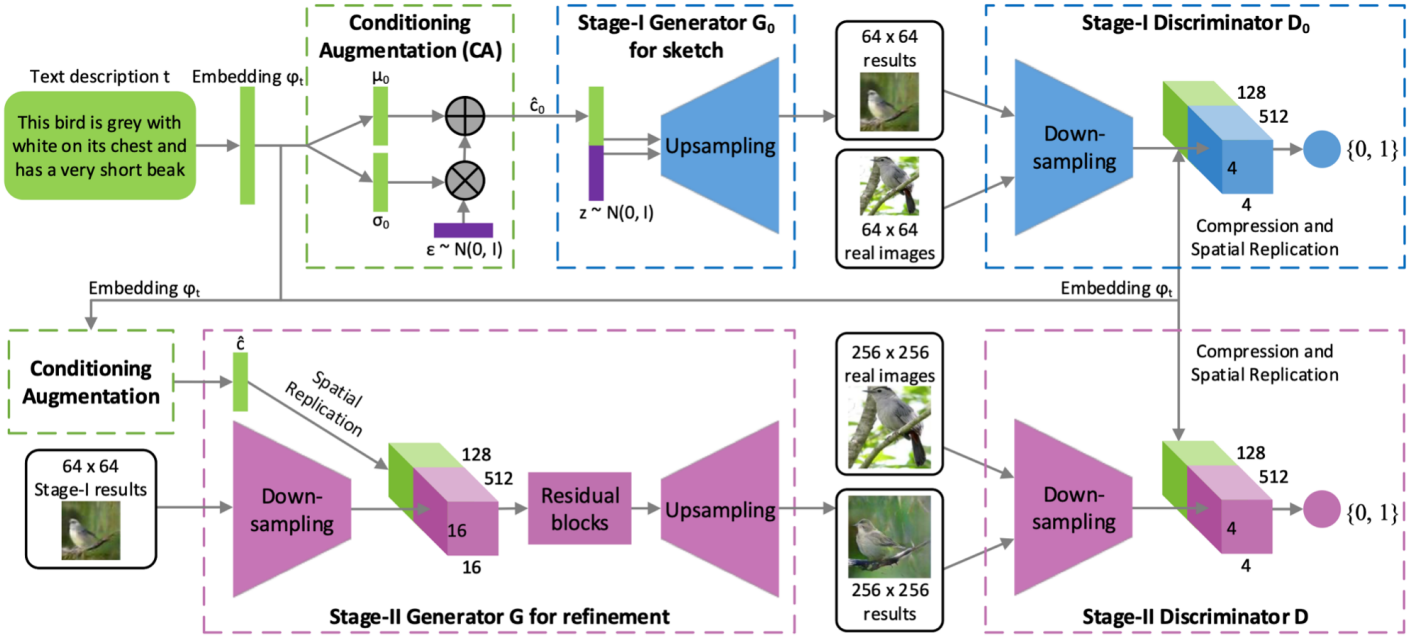
StackGAN solves the problem by generating realistic 256x256 images from just text descriptions. They divide the problem into 2 steps. Stage-I GAN outputs a 64x64 low-resolution image. The Stage-II GAN with the 64x64 image along with the text descriptions as input outputs a 256x256 high-resolution image.

They introduce a novel Conditioning Augmentation technique that encourages smoothness in the latent space and increases the randomness in the dataset so that the model can be Robust

Previous Work

- **Generative Adversarial Text to Image Synthesis: Paper**
- **Learning What and Where to Draw: Paper**

StackGAN Model



They have divided the problem into 2 modules:

- Stage-I GAN
- Stage-II GAN

We need to embed the text descriptions into an embedding vector so that they can be fed into the generator and discriminator. We can use Word2Vec or Glove for this process. But the authors recommend using the encoder used in this paper

It is important features from the text descriptions, otherwise we will miss those features while generating the image

Conditioning Augmentation

If we have fewer Text-Image pairs then there is a high chance that GAN models can collapse. To prevent this, the authors increase the dataset and hence encourages robustness to small disturbances

We use 2 fully connected layers, with the input as the text embedding and the output as the vector of the mean(μ0) and variance(σ0). Now we find the text conditioning variable cˆ0 by c^0 = μ0 + σ0*ε, where ε is sampled from a normal distribution of mean = 0 and variance = 1.

Stage-I GAN

This part sketches the main colours and draws a rough shape of the object using the text descriptions. The input for this part is a vector where c^0 and a vector z is concatenated where z is sampled from a normal distribution of mean = 0 and variance = 1. The random noise z is used to create a random background/scenary. The Stage-I GAN generator is made of upsampling layers to generate the 64x64 low-resolution image.

The discriminator takes the 64x64x3 image as input is downsampled to 4x4x512 volume. They also compress the 1024D text embedding into a 128D vector using a fully connected layer and then spatially replicate to 4x4x128 volume. These two volumes are concatenated along the channel direction and we apply a 1x1 convolution to learn the features across the image and text. A fully connected layer with one node at the end is used to predict whether the image is real or fake

Stage-II GAN

This part is mainly responsible to correct the defects predicted by Stage-I and capturing the important features from the text description which the Stage-I GAN left out

The output image of Stage-I GAN is used as the input along with the text embedding which fed into the Conditioning Augmentation to get the text conditioning variable c^. The Conditioning Augmentation network is not the same so that the Stage-II GAN can learn the important features left out by Stage-I GAN

The 64x64x3 image is downsampled to a 16x16x512 volume which is concatenated with spatial replication of the conditioning variable c^ to 16x16x128. This tensor(concatenated volume) is passed through a set of residual blocks that learns the representation of image and text together and it tries to learn the features from the text which the Stage-I GAN left. The output of the residual network is fed into a set of upsampling layers which outputs a 256x256 realistic image

The discriminator is modelled the same way as in the discriminator in Stage-II GAN except there are more downsampling layers since we are downsampling a 256x256x3 Image to a 4x4x512 volume in this case

Training

During training, the discriminator takes in a real image and its corresponding text embedding as a positive sample and for negative samples, there can be two ways: First is taking a real image and a mismatched text embedding and the second is taking a synthetic image from the generator and it's corresponding text embeddings

The generator uses the ReLU activation function whereas Discriminator uses the LeakyReLU activation function. We use LeakyReLU for smoother gradient flow through the architecture.

First, the Stage-I GAN is trained keeping Stage-II GAN fixed and then the Stage-II GAN is trained keeping Stage-I GAN fixed

Dataset

- CUB (Bird Species Dataset)
- Oxford-102 (Flowers Dataset)
- MS - COCO

Evaluation Metric (Inception Score)

$$I = \exp(\mathbb{E}_{\boldsymbol{x}} D_{KL}(p(y|\boldsymbol{x}) \,||\, p(y))),$$

x denotes one generated sample, and y is the label predicted by the Inception model. The intuition behind this metric is that good models should generate diverse but meaningful images. Therefore, the KL divergence between the marginal distribution p(y) and the conditional distribution p(y|x) should be large

End Note

To know more about StackGAN, check out the paper: StackGAN

To check out the Pytorch Implementation of the StackGAN model, check out my GitHub Repo: GitHub Repo

Results