Generating Image from Text Captions

Faizaan Charania
Stony Brook Unviersity
fcharania@cs.stonybrook.edu

Jay Bhatt
Stony Brook University
jbhatt@cs.stonybrook.edu

I. Introduction

In the field of Computer Vision, the use of deep learning is growing on a daily basis. Automatic synthesis of realistic images from text would be interesting and useful. In recent years generic and powerful recurrent neural network architectures have been developed to learn discriminative text feature representations. Meanwhile, deep convolutional generative adversarial networks (DCGANs) have begun to generate highly compelling images of specific categories, such as faces, album covers, and room interiors. In this work, we develop a similar deep architecture and GAN formulation to do the generate the images. We are trying to replicate the results of the paper Generative Adversarial Text to Image Synthesis [1] and possibly make it better by fine tuning the parameters. We demonstrate the capability of our model to generate plausible images of birds from detailed text descriptions.

The following reasons make tackling this problem challenging:

- The project attempts to replicate the results of a paper that was published in ICML 2016 [1]
- GANs is a fairly recent and advanced concept in the field of Computer Vision, which the team had to learn on the job.
- The problem statement involves virtually using an English sentence and generating a compelling color image to match it.

II. Related Work

A. Generative Adversarial Text to Image Synthesis

This paper by Reed et.al. is the basis of the current project. They have attempted to translate text in the form of single-sentence human-written descriptions directly into image pixels. For example, ”a long bodied bird, with a black face and upper body, with a yellow underbelly and lower body” or ”a small bird has a small sharp bill, a spotted crown, and legs with a large tarsus”. The problem of generating images from visual descriptions gained interest in the research community, but it is far from being solved.

B. Learning Deep Representations of Fine-grained Visual Descriptions

This is another peice of work by Reed et. al.[2]. This paper, published in CVPR 2016 works primarily on taking an English sentence and converting it into a 1024 bit text embedding. Their proposed model trains end-to-end to align with the fine-grained and category-specific content of images. We have used pre-generated text embeddings for captions provided in our dataset.

III. Dataset

We are using the Caltech-UCSD birds database [3]. This data set consists of 200 birds categories and a total of 11,788 images. The dataset also contains 10 human generated text captions for every image. These captions are then used in [2] and the embeddings generated by the text encoder are then used in our model to generate images from it.

IV. Method

For using a Deep Convolutional Generative Adversarial Network for image synthesis from text captions, the first step is to create sufficiently good text representations, such that the images can be conditioned to generate on the given text. In this regard we leverage the work done in Learning Deep Representations of Fine-grained Visual Descriptions [2]. In this paper the authors have used a CNN-RNN text encoding method in which they stack a recurrent network on top of a mid-level temporal CNN hidden layer. As a result of this, the CNN hidden activation is split along the time dimension (the dimension was reduced to 8 steps) and treated as an input sequence of vectors. This text encoding network creates a 1024 bit text embedding for every caption. As making use of a 1024 bit embedding would've required a lot of computational power and a deeper network, we first compress this 1024 bit embedding to 256 bits using a dense neural network. We use a single layer with a LeakyReLU activation applied on it.

Once, the text embedding is created, the next step is to insert this into the initial conditioning vector that is used by the generator and also insert it at the final stage of Discriminator for further convolutional processing. In case of generator, we don't directly pass the encoding to it, but also include a 100 bit noise vector that is normally distributed. This is done so that the GAN doesn't always create exactly same images for a given text caption. Adding a different noise vector will lead to the generator creating new images every time.

However, for Discriminator, it is not so trivial. We have to choose a particular level at which the convolutional image feature maps are to be concatenated with the same text embedding. Depending on the level, the text embedding has to be replicated in the certain dimensions.

After adding the text embedding to the model, we have to decide on a loss function for the networks.

According to the paper there are 3 different cases the discriminator should learn to identify:

- S_r - Real image, Right text
- S_w - Real Image, Wrong text
- S_f - Fake Image, Right text

$$L_d = log(S_r) + (log(1 - S_w) + log(1 - S_f))/2$$

And for the Generator: S_f - Fake Image, Right text

$$L_g = log(S_f)$$

Once, all the above things are setup, the network is trained for 600 epochs and the results generated are shown in the next section.

V. Architecture

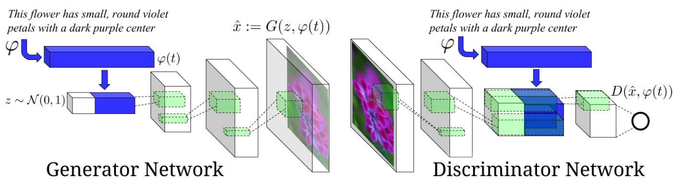

This flower has small, round violet petals with a dark purple center
φ
$\varphi(t)$
$z \sim \mathcal{N}(0,1)$
$\hat{x} := G(z, \varphi(t))$
This flower has small, round violet petals with a dark purple center
φ
$D(\hat{x}, \varphi(t))$
Generator Network
Discriminator Network

Fig. 1. The architecture of the GAN model[1]

A. Generator Network Architecture

We used the following architecture for the Generator:

- The input to the network is the compressed 256 bit text embedding concatenated with a 100 bit noise vector. This is passed to a deconvolutional layer that gives an output of dimensions (64*8) x 4 x 4
- This is followed by 3 convolutional layers and then another de-convolutional
- In the generator G, first we sample from the noise prior $z \in \mathbf{R}^z$ $N(0,1)$ and we encode the text query t using text encoder φ.
- The description embedding $\varphi(t)$ is first compressed using a fully-connected layer to a small dimension (in practice we used 256) followed by leaky-ReLU and then concatenated to the noise vector z.
- Following this, inference proceeds as in a normal deconvolutional network: we feed-forward it through the generator G; a synthetic image $\hat{x}$ is generated via $\hat{x} \leftarrow G(z, \varphi(t))$.
- Image generation corresponds to feed-forward inference in the generator G conditioned on query text and a noise sample.

B. Discriminator Network Architecture

The configuration of the discriminator network is as follows:

- In the discriminator D, we perform several layers of stride-2 convolution with spatial batch normalization [4] followed by leaky ReLU.
- We again reduce the dimensionality of the description embedding $\varphi(t)$ in a (separate) fully-connected layer followed by rectification.
- When the spatial dimension of the discriminator is 4x4, we replicate the description embedding spatially and perform a depth concatenation.
- We then perform a 1x1 convolution followed by rectification and a 4x4 convolution to compute the final score from D.
- Batch normalization is performed on all convolutional layers.

While making the GAN we made sure to follow the tips and advice given in the article[5]. This helped us tune the network better and get an improvement in results

VI. Results and Analysis

A. Samples generated by the model

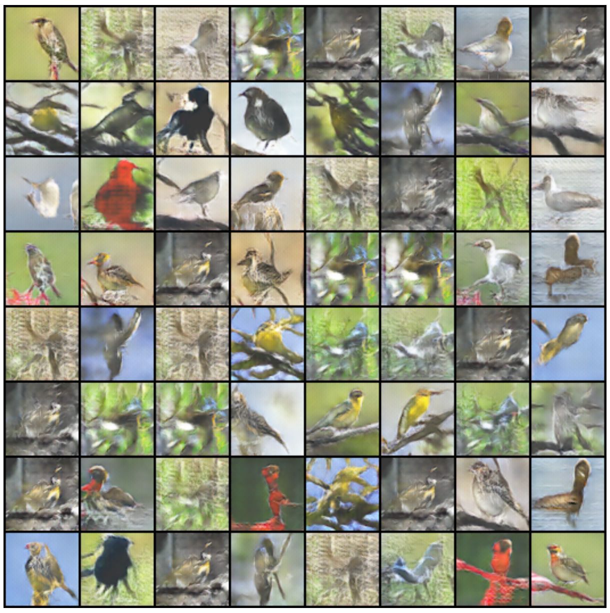
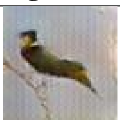
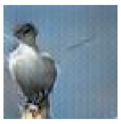
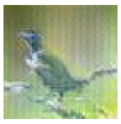
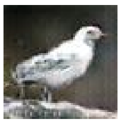
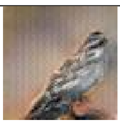
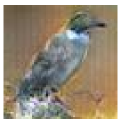
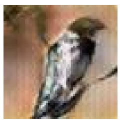
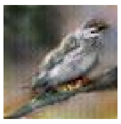


Fig. 2. A grid of 64 images generated after a particular epoch of the model

B. Generated samples matched with the input captions

Figure	Caption
	A long bodied bird with a black face and upper body with a yellow underbelly and lower body
	A medium bird with a white belly black rectrices and a bright yellow throat
	A short billed fully breasted bird with a beautifully contrasting crown this green and gold
	A skinny white bird with brown and black nape and back sits perched on a thin branch

	A small bird has a small sharp bill a spotted crown and legs with a large tarsus
	A small bird with a blue breast and a grey belly with a sharp small beak
	A small bird with a bright green belly and dark green body covering
	A small bird with a long pointed bill feathers along its belly are gray white and brown

VII. CONCLUSIONS

The model was working decently on the CUB dataset. It generated fairly sharp images which were visible as birds. The results were comparable to the GAN-CLS algorithm mentioned in the paper. Apart from that, we learnt how to implement, GAN in a real world examples. We expected it to be moderately difficult, as we were new to pytorch and GAN is something we recently studied. Hence, we thought to explore it in detail. And as it turns out it was a bit more difficult than we thought. As there were small technical implementation details [give example if something in mind] which we did not know about, until we started the implementing ourselves.

ACKNOWLEDGEMENT

We would like to show our gratitude to Prof. Roy Shilkrot who was very supportive of us picking picking this project in spite of us not being certain of how achievable the project was. We would also like to thank him for his continued help and guidance throughout the implementation of the project.

REFERENCES

[1] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[2] "Learning deep representations of fine-grained visual descriptions, booktitle = IEEE Computer Vision and Pattern Recognition, year = 2016, author = Scott Reed and Zeynep Akata and Bernt Schiele and Honglak Lee."

[3] (2011) The Caltech-UCSD Birds-200-2011 Dataset. [Online]. Available: http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[5] (2016) How to Train a GAN? Tips and tricks to make GANs work. [Online]. Available: https://github.com/soumith/ganhacks