

PixelBrush: Art Generation from text with GANs

Shani Ben Baruch, Omer Cohen,

Abstract

Briefly describe your problem, approach, and key results. Should be no more than 300 words.

1 Introduction

Describe the problem you are working on, why it's important, and an overview of your results

2 Related Work

Discuss published work that relates to your project. How is your approach similar or different from others.

In the past few years, there were many works which try to generate images using convolutional neural network (CNN) architecture.

Gatys et al. [gatys2016image] proposed a neural style transfer where new images are been synthesized using a content image and style image. However, this method requires an image as an input.

Goodfellow et al. [goodfellow2014generative] proposed a methods to generate images given a noise vector. The network proposed contain a generator G and a discriminator D that are been trained together. G synthesizes samples from a target distribution, while D needs to distinguishes between samples generated by G and a training set from the target distribution. The goal of G is to create samples that are classified by D as real samples. Radford et al. [radford2015unsupervised] created a successful architecture, based on Goodfellow's work, called deep convolutional generative adversarial network (DCGAN). Mirza & Osindero [mirza2014conditional] introduced a new method called conditional generative adversarial network (cGAN) to generate samples from a specific class.

Reed et al. [reed2016generative] were able to generate images based on textual description. They also created a new kind of discriminator called matching-aware discriminator (GAN-CLS), which learn to evaluate whether samples from the generator meet the conditional constraint. They introduced three kind of losses: one, that computes a real image with matching text; second, that computes a synthetic image with arbitrary text; and third, that computes a real image with mismatching text (which the discriminator must learn to score as fake).

Another set of works that generate images from textual description proposed by Zhang et al. [zhang2017stackgan, zhang2017stackgan++]. They were able to generate 256x256 high resolution images.

In our work we aim to generate art images from textual description. This idea is based on Jiale's idea [Zhi2017PixelBrush]. Jaile used Reed et al. [reed2016generative] basic model and change the architecture. Jaile used tensorflow for his work, and did not publish his implementation.

3 Background

3.1 Generative Adversarial Networks

Generative adversarial networks (GANs) have dramatically expanded the dimension of deep learning research since they were first introduced by Ian Goodfellow et al. [goodfellow2014generative].

The GAN model consists of two main components: a generator network G which produces samples $G(z)$ from the input of random noise z from distribution p_z , and a discriminator network D which determines whether a given sample is real from the true data distribution p_{data} or is artificially created. Both networks are trained simultaneously, and try to optimize the following objective function,

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p_z} [\log(1 - D(G(z)))] \quad (1)$$

Conditional GAN [mirza2014conditional] is an extension of GAN where both the generator and the discriminator receive additional information c , yielding $G(z, c)$ and $D(z, c)$. This additional information allows the generator to generate images conditioned on that information.

3.2 Skip-thoughts Text Embedding

In order to use textual description in our network, we need to convert the description into a vector representation. We use a pre-trained skip-thoughts model proposed by Kiros et al. [kiros2015skip]. This model was trained on the bookCorpus dataset [zhu2015aligning].

The skip-thoughts architecture contains one encoder and two decoders. Given a sentence tuple (s_{i-1}, s_i, s_{i+1}) , the model takes an input s_i and tries to predict the previous sentence s_{i-1} and the following sentence s_{i+1} .

There are two separate models of the encoder. The first is uni-skip model, which refers to unidirectional encoder with 2400 dimensions. The second is bi-skip model, which refers to a bidirectional model with forward and backward encoders of 1200 dimensions each, the outputs are then concatenated to form a 2400 dimensional vector. There is a third model, combine-skip, which is a concatenation of the uni-skip model and the bi-skip model, resulting in a 4800 dimensional vector. In our work we use the combine-skip model.

4 Dataset

4.1 Oxford Painting Dataset

We use Oxford paintings dataset [Crowley14, Crowley14a] which contains 8629 images in 10 categories: airplane, bird, boat, chair, cow, table, dog, horse, sheep and train. The dataset is split into train, validation and test sets. Because we followed Jaile’s work, we merged the different datasets (train, validation and test) and picked around 3200 images from only 4 categories - cow, dog, horse and sheep. 85% of the merged dataset has been used for training, and the other 15% for testing.

4.2 Adding Descriptions

This dataset only contains images with titles and categories. However, this is not enough to generate artwork, and we also need textual description for each image. As proposed by Jaile we used a two-step solution to create those descriptions. First, we generate descriptions using Neuraltalk2, a pre-trained network (trained on MSCOCO dataset), proposed by Karpathy et al. [karpathy2015deep]. Second, we manually reviewed all the generated descriptions and fixed the defects.

5 Methods

Discuss your approach for solving the problems that you set up in the introduction. Why is your approach the right thing to do? Did you consider alternative approaches? You should demonstrate that you have applied ideas and skills built up during the quarter to tackling your problem of choice. It may be helpful to include figures, diagrams, or tables to describe your method or compare it with other methods

5.1 Network Architecture

As discussed before, we followed the architecture proposed by Jaile [Zhi2017PixelBrush], who drew it from Reed et al. [reed2016generative]. The architecture is presented in Figure 1.

First, we create text embedding from a given description, using skip-thoughts model. Second, we concatenate that embedding with random noise vector z . And third, we train a generative adversarial network. The generator aims to create images conditioned on text description. The discriminator aims to determine whether the input images are real or fake based on their descriptions (see GAN-CLS in section 2 for more details).

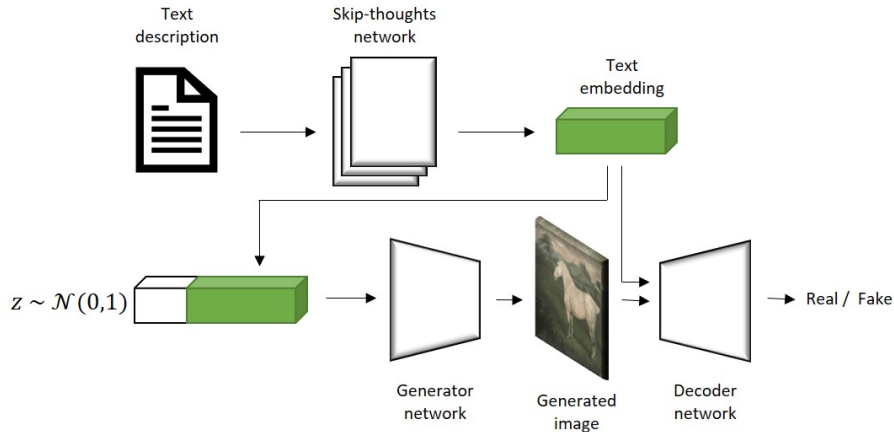


Figure 1: Overview of the network architecture

5.2 Generator architecture

There are 3 different kinds of generators we built: simple, normal and deep (see Table 1 for more details). The input for each generator is a concatenation of a random noise vector with 100 dimensions and a text embedding with 4800 dimensions. The output is an image of size $3 \times 64 \times 64$.

We use transpose convolution with stride 2 for up-sampling, and convolutional layers with stride 1. Each layer, except the last layer, follow by batch normalization and ReLU activation. The method for creating the normal and deep networks is similar to ResNet [he2016deep], which allows us to create deeper networks.

5.3 Discriminator architecture

The discriminator is built from convolutional layers follow by batch normalization and leakyReLU activation. The last convolutional layer is only follow by a sigmoid layer. The input is a $3 \times 64 \times 64$ image and a text embedding vector, and the output is a score that indicated whether the image is real or fake (see Figure 2).

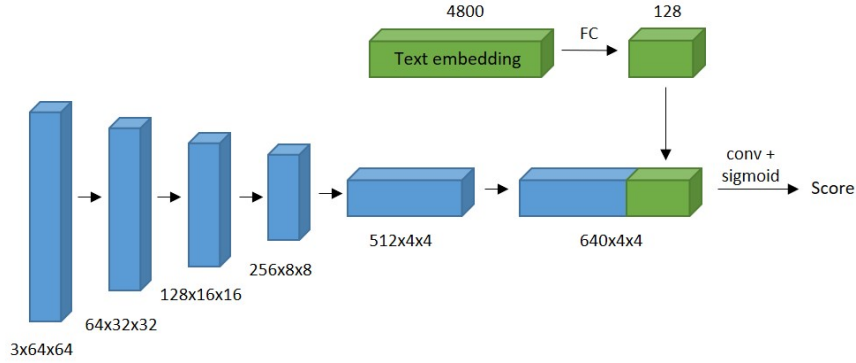


Figure 2: Overview of discriminator architecture

6 Experiments

Discuss the experiments that you performed to demonstrate that your approach solves the problem. The exact experiments will vary depending on the project, but you might compare with previously published methods, perform an ablation study to determine the impact of various components of your system, experiment with different hyperparameters or architectural choices, use visualization techniques to gain insight into how your model works, discuss common failure modes of your model, etc. You should include graphs, tables, or other figures to illustrate your experimental results

6.1 Evaluation metrics

It is difficult to evaluate performance on GAN models. For this project we follow Jaile suggestion and choose inception score (IS) [salimans2016improved],

$$I = \exp(E_x D_{KL}(p(y|x)||p(y))), \quad (2)$$

where x denote the generated image, and y is the label predicted by the IS. 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.

6.2 Comparison between different generators

In one of the experiments that Jaile did, he shows that generator has a strong affect on the image quality. To understand how generated affects generated image quality, we trained our network with three different Generator: "simple", "normal" and "deep" as denoted in 5.2 we trained each of the model on Oxford paintings dataset for 350 epochs, in contrast to Jaile that train his dataset for 250 epochs. Despite the difference in experimental length, we can see that we received the same results he received. From table 2 we can see that roughly inception score increase with the number of training iterations. In addition, inception score increases with the generator work goes deeper. In direct proportion to the numerical results, we also can see that the generate images looks more realistic with the generator work goes deeper.

6.3 Comparison against baseline

We tested our work against a GAN model that does not use textual descriptions. We follow the network of DCGAN [radford2015unsupervised], and compared it to our "simple" generator architecture. Results after 250 epochs are presented on table 1. We can see that in contrast to Jaile's result, our DCGAN network generates better images than the 'simple' architecture. This is reflected both in the inception score and also in the visibility of the images themselves. We are not sure why we received different results that Jaile on this section. Jaile didn't shared his written project, and therefore we can't know how precisely he implemented his networks. Maybe some difference between our project cause that various results.

7 Conclusion

Summarize your key results - what have you learned? Suggest ideas for future extensions or new applications of your ideas

Generator Architecture				
Simple	Normal		Deep	
input (4900-dimensional vector)				
FC-4096				
reshape to 256 x 4 x4				
upconv4-256	conv1-128	identity	conv1-128	identity
upconv4-128	conv3-128		conv3-128	
upconv4-64	conv3-512		conv3-512	
upconv4-3	+		+	
	upconv4-256		upconv4-256	
	conv3-256		conv3-256	
	conv1-64	identity	conv1-64	identity
	conv3-64		conv3-64	
	conv3-256		conv3-256	
	+		+	
	upconv4-256		upconv4-256	
	conv3-128		conv3-128	
	upconv4-128		conv1-64	identity
	conv3-64		conv3-64	
	upconv4-64		conv3-128	
	conv3-3		+	
			upconv4-128	
			conv3-64	
			conv1-32	identity
			conv3-32	
			conv3-64	
			+	
			upconv4-64	
			conv3-3	
tanh				

Table 1: Generator network architectures (simple, normal and deep). The (up)conv layers denote as "(up)conv(filter size) - (number of channels)"

epochs	simple	normal	deep
20	1.55 \pm 0.05	2.02 \pm 0.11	1.96 \pm 0.13
40	1.87 \pm 0.10	1.96 \pm 0.15	2.09 \pm 0.12
60	1.88 \pm 0.14	2.08 \pm 0.19	2.29 \pm 0.19
80	1.93 \pm 0.12	2.20 \pm 0.13	2.24 \pm 0.13
100	2.07 \pm 0.10	1.97 \pm 0.10	2.27 \pm 0.19

Table 2: Inception score for 3 kinds of generator networks. Higher inception score means better image quality.