PixelBrush: Art Generation from text with GANs

Jiale Zhi
Stanford University
jz2@stanford.edu

Abstract

Recently, generative adversarial networks (GANs) have been shown to be very effective in generating photo-realistic images. And recent work from Reed et al. also allows people to generate images with text descriptions. In this work, we propose a tool called PixelBrush that generates artwork from text descriptions using generative adversarial networks. Also, we evaluated the performance of our model and existing models on generating artworks. We also compared different generator architecture to see how the depth of the network can affect generated image quality.

1. Introduction

Fine art, especially painting are an import skill that human has mastered during a long time of evolution. As of today, only human can create paintings from ideas or descriptions. Artists can draw a painting of a bird giving a description "Draw a painting with a bird flying in the sky" without any difficulties. Currently, this skill is only limited to human. There is no known way that describes how paintings can be drawn algorithmically given an input description. It would be interesting to see how this process can be learned by computer algorithms and the replicated to create more artistic paintings.

There are a couple of challenges around this problem. First, computer algorithm needs to understand what objects or scene need to be drawn on the painting and also the relationships between different objects and scenes. Second, given the algorithm understand the description, generate an artistic image according to the description that provided. Note that the mapping between description to images is not one to one, one description can be mapped to an infinite amount of images.

In this work, we propose a new tool called PixelBrush that given a description of a painting, generates an artistic image from that description. The input to our tool is a piece of short text, like "a bird flying in the sky". We then use RNN to process the input text into a vector and then use generative adversarial networks to generate artistic images.

Figure 1: Images generated from text descriptions on test set from our work, first column are real images, the other columns are generated images. Those images are generated from the text descriptions from each corresponding row.

The main contributions of our work:

- We trained a conditional generative adversarial network to generate realistic painting images from text descriptions. Our result shows that generated paintings are consistent with input text descriptions.
- We compared our result with DCGAN to show that by adding condition to GAN, it helps with generated image quality.
- We compared how different generator affects generated images quantity by training our network on three different GANs with same discriminator architecture and different generator architecture.
- We provided another angle to look at the complexity of generated with evaluating the entropy of generated images showed how generated image's entropy changes through time and how it compared with real image entropy.

2. Related Work

Traditionally, people have been using computer algorithms to generate artistic paintings. Those generated work are usually random, abstract, or full of patterns due to the limitations of the algorithm. It's hard to customize those artwork based on different people's needs. Some examples of computer-generated art are available at Syntopia[1].

Recently, with the development of computer vision, people have been using neural networks to generate artistic images from real world images. Gatys *et al*. [7] proposed a convolutional neural network (CNN) based method to transfer styles from a painting to a photo. This method has been shown to generate artwork with high perceptual quality. And it can generate very high-resolution images which most of the other neural networks failed to do. However, this still requires a photo as input so generated artwork, and the content of the output image is fully based on input photo. Also, the style photo and the image photo may not work together very well because the color in the content image may affect the output of the style transfer. Causing the generated image to lose some artistic effects.

While deep style transfer provided a way to generate artwork using provided content image, generative adversarial networks (GANs)[9] provided another way of generating highly compelling images. Radford *et al*. [18] introduced a class of GAN called deep convolutional generative adversarial networks (DCGANs). Their work shows that GAN is able to learn a hierarchy of representations from object parts to scenes. Additionally, GAN can be used to generate near photo-realistic images of bedrooms. However, DCGAN generated images have low resolution and still suffer from different kinds of defects.

Nguyen *et al*. [17] proposed a new method called Plug & Play Generative networks (PPGN). Their works show that by introducing an additional prior on the latent code, sample quality and sample diversity can be improved, leading to a model that produces high-quality images at high resolution (227×227).

Although GAN has been successfully used in a lot of areas[6], the early age GANs doesn't provide the ability to generate images according to some input features. To help with this issue, Mirza *et al*. [16] proposed a new kind of GAN called conditional generative adversarial network (cGAN). They introduced an additional condition information and showed that cGAN can generate MNIST digits conditioned on class labels. Gauthier [8] showed that compared to vanilla GAN, conditional GAN can use conditional information as a deterministic control to deterministically control the output of the generator.

There are also developments in the synthesis of realistic images from text using GANs. Reed *et al*. [19] proposed a way to modify GAN's generator network to not only accept input random noise z, but also the description embedding $\varphi(t)$, so that generated images will be conditioned on text features. They also created a new kind of discriminator called matching-aware discriminator (GAN-CLS) to not only discriminate whether an image is real or fake but also discriminate image and text pairs. So that both discriminator network and generator network learns the relationship between image and text. Their experiments showed that their trained network is able to generate plausible images that match with input text descriptions. However, their network is limited to only generate limited kinds of objects: flower and birds. Also, generated image's resolution is low.

Zhang *et al*.'s recent work StackGAN [26] bridged resolution gap between text to image synthesis GAN and models like PPGN. In their work, they proposed a novel two staged approach for text to image synthesis. The first stage network is able to generate low-resolution plausible images from text descriptions. The second stage network then takes the generated image from the first stage network, and then refine the image to generate a more realistic and much higher resolution image. In their experiments, they were able to generate 256×256 high-resolution images from just a text description.

In our work, we propose a way of generation artwork from text descriptions. We will use natural language as input, so people can describe what kind of artwork they want, and then our tool PixelBrush will generate an image according to the description that provided.

3. Background

3.1. Generative Adversarial Networks

Generative adversarial network consists of a generator network G and a discriminator network D. Given training data x, G takes input from a random noise z and tries to generate data that has the similar distribution as x. Discriminator network D takes input from both training data x and generated data from G, it estimates the probability of a sample came from training data rather than G.

To learn the generator's distribution p_z over data x, the generator builds a mapping from a prior noise distribution $p_z(z)$ to data space as $G(z;\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. Discriminator network D is also represented as a multilayer perception $D(x;\theta_d)$ where θ_d is the parameters of the multiplayer perception.

G and D are trained simultaneously: we adjust parameters of D to maximize the probability of assigning the correct label for both training examples and samples from G and adjust parameters of G to minimize $\log(1 - D(G(z))$. In other words, D and G play the following two-player minmax game with value function V(G, D):

[1] http://blog.hvidtfeldts.net

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

3.2. Conditional Generative Adversarial Nets

Conditional generative adversarial nets are a variant of GAN that introduced additional information y, so that both generator G and discriminator D are conditioned on y. y could be any kind of auxiliary information such as class labels or other information.

The prior noise $p_z(z)$ and y are combined together to form a hidden representation as input to generator G. Gauthier *et al*. [8] show that the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. x and y are also combined as input to discriminator network.

The modified two-player minmax game value function V(G, D) would be:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]. \quad (2)$$

3.3. Text embeddings

In order to use text as condition to conditional GANs, we first need to convert text into a text embedding vector. There are a lot of ways of doings this. We use skip-thought vectors proposed by Kiros *et al*. [14]. Skip-thoughts use a encoder-decoder model. Given a sentence tuple (s_{i-1}, s_i, s_{i+1}), the encoder takes input s_i and produces a vector that is a representation of s_i. The decoder then takes the encoded vector as input, and tries to reproduce the previous sentence s_{i-1} and the following sentence s_{i+1} of s_i.

There is a wide selection of encoder/decoder pairs that can be chosen from, including ConvNet-RNN [11], RNN-RNN[2] and LSTM-LSTM [22]. The authors of skip-thoughts chose to use RNN encoder with GRU [3] activations and an RNN decoder with a conditional GRU.

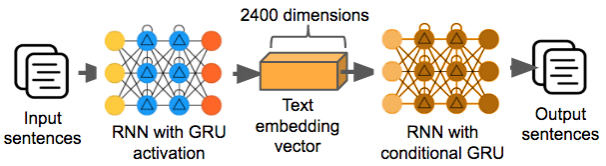

Figure 2: Overview of skip-thought embedding network

3.3.1 Encoder

The encoder uses a standard GRU network. Let s_i be a sentence that consists of words $w_i^1, ..., w_i^N$ where N is the number of words in the sentence. When encoding sentence s_i, at each time step, the encoder takes w_i^t as input, and produces a hidden state h_i^t. h_i^t can be viewed as a state that captured the information that represents $w_i^1, ..., w_i^t$. Then at next step, both w_i^{t+1} and h_i^t are feed to the network and produces a new hidden state that represents $w_i^1, ..., w_{i+1}^t$. In the end, when all the words are feed to the network, h_i^N then represents the whole sentence. To encode a sentence, we iterate through the following equations (dropping the subscript i):

$$\begin{aligned} r^t &= \sigma(W_r x^t + U_r h^{t-1}) \\ z^t &= \sigma(W_z x^t + U_z h^{t-1}) \\ \bar{h}^t &= \tanh(W x^t + U(r^t \odot h^{t-1})) \\ h^t &= (1 - z^t) \odot h^{t-1} + z^t \odot \bar{h}^t \end{aligned} \quad (3)$$

3.3.2 Decoder

The decoder is also a GRU network where it takes the output from the encoder h_i as a condition. The computation is similar to encoder network except that three new matrices C_z, C_r and C are introduced to condition the update gate, reset gate and hidden state computation by the sentence vector. In order to produce previous sentence s_{i-1} and following sentence s_{i+1}, two separate GRU networks are trained, one for s_{i-1}, one for s_{i+1}. There is also a vocabulary matrix V that is used to produce words distribution from hidden state h_{i-1}^t and h_{i+1}^t. To get hidden state $h_t^t + 1$ at time t, we iterative through the following sequence of equations (dropping the subscript i):

$$\begin{aligned} r^t &= \sigma(W_r^d x^{t-1} + U_r^d h^{t-1} + C_r h_i) \\ z^t &= \sigma(W_z^d x^{t-1} + U_z^d h^{t-1} + C_z H_i) \\ \bar{h}^t &= \tanh(W_z^d x^{t-1} + U^d(r^t \odot h^{t-1}) + C h_i) \\ h_{t+1}^t &= (1 - z^t) \odot h^{t-1} + z^t \odot \bar{h}^t \end{aligned} \quad (4)$$

Given h_{i+1}^t, let v_{i+1}^t denote the row of V corresponding to the word w_{i+1}^t. The probability of word w_{i+1}^t given the previous $t-1$ words and the encoder vector is. The same method can be applied to sentence s_{i-1}.

$$P(w_{t+1}^t | w_{t+1}^{<t}, h_i) \propto \exp(v_{v_{i+1}^t} h_{i+1}^t) \quad (5)$$

3.3.3 Objective

Given input sentence (s_{i-1}, s_i, s_{i+1}), the objective is to maximize the sum of the log-probabilities for the previous and following setences conditioned on the encoder representation:

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, h_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, h_i) \tag{6}$$

And the total objective is summed over all input training tuples.

3.3.4 Model details

In our work, we use a pertained skip-thougth encoder to encode text description into 4800-dimension vectors. The vector is a combination of uni-skp and bi-skip vectors as described in [14], both are 2400-dimension.

4. Methods

Our approach is to train conditional generative adversarial nets conditioned on text features encoded by an skip-thought vector text embedding encoded from a text descriptions. Given training data x and text y, both generator and discriminator networks are trained together so that distribution of x can be learned.

4.1. Network architecture

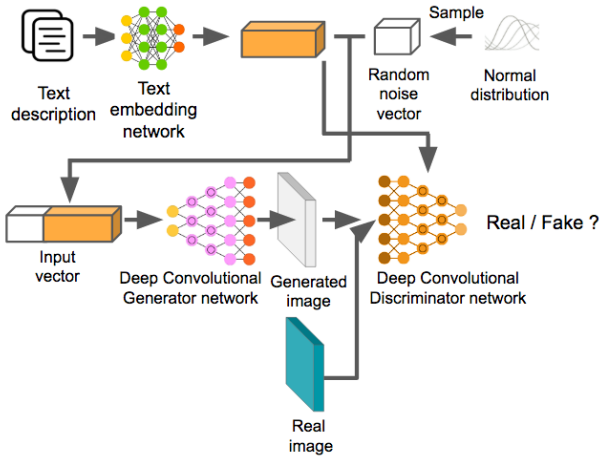

Figure 3: Overview of network architecture of our network that generates painting images from text descriptions.

Overall, we follow the architecture proposed by Reed *et al*. [19]. For each training iteration, we use a skip-thought text embedding network to convert text descriptions into text embedding vectors v. And feed this text embedding with random noise z to generator network. Generator network then generates images conditioned on the text description. These generated images are later feed into a discriminator network together with their text embeddings. The discriminator network then tells whether the input images are real or fake.

4.1.1 Generator architecture

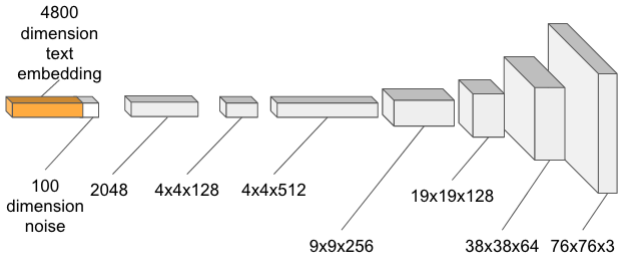

Figure 4: Overview of "simple" generator network

We use a multi-layer deep convolutional neural network to generate images. The network takes a random noise z and a text embedding vector v. We use a fully connected layer to convert the input to a 2048-dimension vector. Then we use transposed convolutional layer to convert this vector into an image. At each transposed convolutional layer, we use ReLU activations and spatial batch normalization before each layer. For the last layer, we use $\tanh$ activation. Network architecture are based on StackGAN [2].

To compare how generator network depth affects generated image quality. We created 3 kinds of different generator: "simple", "normal" and "deep". An overview of "simple" network architecture are showed in figure 4. To get deeper generator network, we use a method similar to Resnet[25]. By allowing output of previous layer splitting into two flows and skipping part of the network, it created better gradient flow and allows us to create deeper generator network.

4.1.2 Discriminator architecture

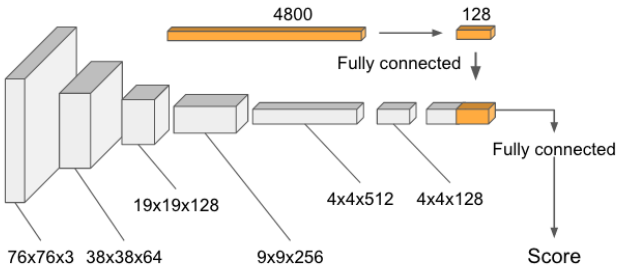

Figure 5: Overview of discriminator network

[2]https://github.com/hanzhanggit/StackGAN

Generator architecture				
Simple	Normal		Deep	
5 layers	1 layers		1 layers	
input (4900-dimension vector)				
FC-8192				
reshape to $4 \times 4 \times 256$				
upconv4-256 upconv4-128 upconv4-64	conv1-128 conv3-128 conv3-512	identity	conv1-128 conv3-128 conv3-512	identity
upconv4-3	$\oplus$		$\oplus$	
	upconv4-256-2		upconv4-256	
	conv3-256-1		conv3-256	
	conv1-64 conv3-64 conv3-256	identity	conv1-64 conv3-64 conv3-256	identity
	$\oplus$		$\oplus$	
	upconv4-256		upconv4-256	
	conv3-128		conv3-128	
	upconv4-128		conv1-64 conv3-64 conv3-128	identity
	conv3-64			
	upconv4-64			
	conv3-3		$\oplus$	
			upconv4-128	
			conv3-64	
			conv1-32 conv3-32 conv3-64	identity
			$\oplus$	
			upconv4-64	
			conv3-3	
tanh				

Table 1: Generator network architecture. From "simple" to "deep", the depth of the network increases. The (up)conv layer are denoted as "(up)conv⟨ filter size ⟩-⟨ number of channels. ⟩". All conv layers are using stride 1 and all upconv layers are using stride 2. There is also a batch normalization layer right after each fully connected layer or convolutional layer except the last layer (not shown in the table). ReLU activation is applied after batch normalization layer except the last layer where tanh activation is applied.

We also use a multi-layer deep convolutional neural network for our discriminator. The network takes an image and a text embedding vector v as condition, and outputs the probability of the image being real. We then apply the loss from equation 2 to compute the loss for discriminator. Detailed network architecture are showed in figure 5.

4.1.3 Training details

We use Adam optimizer[13] to train both discriminator and generator. We follow the advise from Chintala *et al*.[3] and at each round train both generator and discriminator once. We use a learning rate of 0.0002 for both discriminator and generator. We use a batch size of 64 and decay learning rate to half for every 150 epochs.

5. Dataset

We use Oxford paintings dataset [5, 4] to help us develop our algorithm. This dataset contains 8629 images in 10 categories: aero, bird, boat, chair, cow, table, dog, horse, sheep, train. It is split into training, validation, and test sets. Details statistics on images is showed on table 2. Some sample paintings are shown in Figure 6.

Because the focus of this work is to generate realistic painting images, we merged training, val and test set together, and selected around 3000 images from Cow, Dog, Horse, Sheep for training, and around 600 images from those 4 categories for testing.

5.1. Image pre-processing

The images in the dataset are in variety of sizes: 624×478, 358×544, 624×413, 624×453 etc. We unified the images by resize them to 76×76 with no cropping. Although this will change the height to width ratio of original image, but in our experiment, we found this isn't a problem and doesn't have a big effect on our results.

We also removed all the black-white images that only has one color channel. So all the images after pre-processing have size $76 \times 76 \times 3$.

5.2. Adding descriptions

One drawbacks of Oxford paintings dataset for our application is it only contains images with categories, and painting titles. But painting titles are usually short and most of the titles doesn't capture the whole scene on the image and thus it's hard to used them as image descriptions. For our application, although all images in this dataset are labeled, but in order to use this to generate artworks, we also need to provide captions to those images.

In order to do this, we use a two-step solution to this problem. First, we use image captioning network Neuraltalk2 proposed by Karpathy *et al*. [12] to generate captions for all the painting images. Although Neuraltalk2 is trained on a completely different dataset MSCOCO, we found it also works well for generating text descriptions for painting images in Oxford paintings dataset. Second, we human-reviewed all the generated text descriptions and manually fixed all the defects in the generated captions.

[3] https://github.com/soumith/ganhacks

	Aero	Bird	Boat	Chair	Cow	Table	Dog	Horse	Sheep	Train	Total
Train	74	319	862	493	255	485	483	656	270	130	3463
Val	13	72	222	140	52	130	113	127	76	35	865
Test	113	414	1059	569	318	586	549	710	405	164	4301
Total	200	805	2143	1202	625	1201	1145	1493	751	329	8629

Table 2: Number of images in the Oxford paintings dataset containing an instance of a particular class, as well as total number of images for each subsett

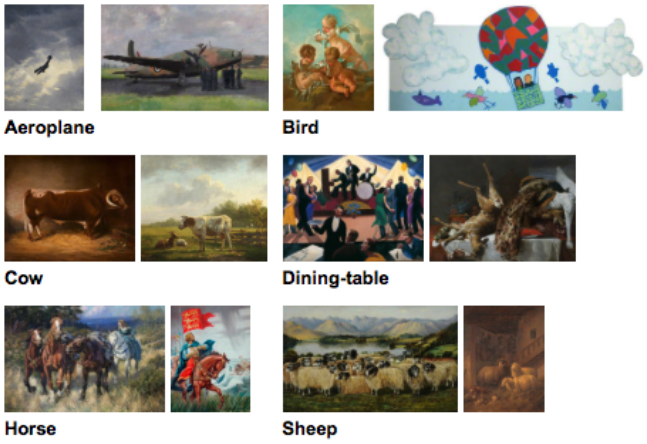

Figure 6: Sample images from Oxford Paintings dataset

In addition, we also use Caltech-UCSD Birds 200 dataset[23] as a comparison to show how it's different to generate images from paintings images and bird images. Caltech-UCSD Birds 200 dataset is now a standard in text-to-image generation area and a lot of work are reported on this dataset[19, 26, 20].

6. Experiments

6.1. Evaluation methods

Evaluation the quality of generated images is a hard problem. Salimans *et al*. [21] proposed two ways to evaluate the quality of generated images.

The first metric of performance is can human annotators tell the difference between computer generated images and real images. The downside is that this metrics varies depending on the setup of the task. And also it takes time to finish this evaluation process.

The second method is use machine learning to tackle this problem. They propose to use an off-the-shelf image classifier to evaluate the quality of generated images.

We plan to use the second method to evaluate the quality of generated artwork. To get a quantitive score for our generated images, we use inception scores proposed by Salimans *et al*. [21].

$$I = \exp(\mathbb{E}_x \mathrm{KL}(p(y|x)||p(y))) \tag{7}$$

The intuition behind this that we want out model to generate meaningful objects. So given input image x, the label conditional label distribution $p(y|x)$ should have low entropy. And we also want out model to generate varied images, so the marginal distribution $p(y)$ should have high entropy. Therefore, the DL divergence between $p(y|x)$ and $p(y)$ should be large.

6.2. Comparison against baselines

DCGAN[18] This is a kind of GAN that is widely used because of the quality of the generated images. DCGAN uses a series of four fractionally-strided convolutions to convert data in high level representation into 64 $\times$ 64 pixel images. We use DCGAN-tensorflow[4] as our baseline implementation.

Figure 7: Image generated from baseline. DCGAN trained on Oxford paintings dataset for 240 epochs with Adam optimizer, learning rate 0.0002, momentum term of Adam 0.5 and batch size 64.

[4]https://github.com/carpedm20/DCGAN-tensorflow

Method	Inception score
DCGAN	1.0012 ± 0.0003
Our work	2.7113 ± 0.3367

Table 3: Inception score for DCGAN generated images and our conditional GAN. The inception score correlates to perceptional image quality. Higher inception score means better image quality

To compare DCGAN generated images with our work, we use the same generator architecture "Simple" for both DCGAN and our conditional GAN, generator architecture can be found from Table 1

Figure 8: Image generated from our work.

We can see from figure 7 and 8, with the same number of training epochs and same generator setting, both GAN generated relative sharp images. The different is that DCGAN generated images seem to have everything mixed together, and there is no recognizable object in it. While our work leverages the help with labels(text descriptions), so generated images looks more real and looks really artistic compared to DCGAN generated results. And some images shows recognizable objects in it.

The intuition behind this is DCGAN doesn't use labels, so all the input images and features are more likely to be mixed together. With the help of labels, conditional GAN knows how to generate objects according to text description, so generated images looks more real. Although the quality of generated images is still low, we believe the quality can be improved through longer training time.

6.3. Comparison between different generators

To understand how generated affects generated image quality, we trained our network with three different generators as denoted in 4.1.1 "Simple", "Normal" and "Deep". We use the same discriminator to pair with each of the generator and use the same setup as we described in 4.1.3. Because of limited time and compute resources, we trained each of the model on Oxford paintings dataset for 5000 iterations (106 epochs). We compared the inception score of each of the model on each 1000 iterations.

Iterations	Generator		
	Simple	Normal	Deep
1000	2.33, 0.42	2.24, 0.23	2.67 ± 0.20
2000	2.43, 0.33	2.69, 0.49	2.56, 0.33
3000	2.70, 0.31	2.76, 0.58	2.47, 0.37
4000	2.60, 0.23	2.91, 0.38	2.67, 0.51
5000	2.74, 0.42	2.96, 0.33	3.05, 0.32

Table 4: Inception score for 3 kind of generator network generated images. Result are samples every 1000 iterations.

From table 4 we can see that roughly inception score increase with the number of training iterations. Because we only trained the network for 5000 iteration, we can't draw a direct conclusion from the data that we collected. But roughly, inception score increase with the number of training iterations. And also, inception score increases with the generator work goes deeper.

6.4. Entropy analysis

In information theory, entropy is a measure of unpredictability of the state, or equivalently, of its average information content. To better understand how the much information contained in generated image, we are curious about how entropy of generated images changes through time and how that compared to real images. In order to get this information, we calculated entropy for every minibatch of real and fake images that are feed to the discriminator.

From Figure 9, 10, we can see that real images' entropy, their value are near 1600 and keeps stable from different batches of real images. On the other hand, the entropy of generated image are very low at the beginning. Denoting that the generator hasn't learnt the distribution of real images and generated images has a low entropy. And usually at this stage, generated images contains large blocks of colors and lack variety. But through the process of training, generator learns to generate images that has higher entropy quickly. At iteration 300, the generator generated images already have similar entropy as real images.

One interesting result from this analysis is that the entropy of generated images stays stable after iteration 1000. But the entropy of fake images are consistently higher that real images. It would be interesting to see why generator generated images are having higher entropy constantly.

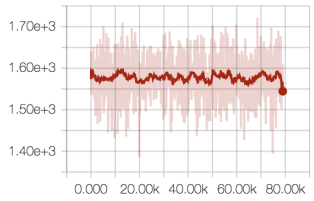

Figure 9: Image entropy ofReal images. Horizontal axis is number of training iterations. Vertical axis is the average information entropy of images in the same batch.

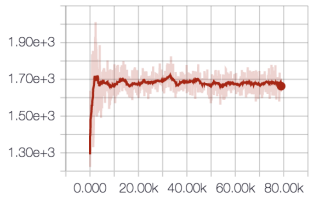

Figure 10: Image entropy of generated images. Horizontal axis is number of training iterations. Vertical axis is the average information entropy of images in the same batch.

7. Conclusion

In this work, we proposed a new tool PixelBrush that uses conditional generative adversarial networks to generate realistic and artistic images from text descriptions. We shows that by adding text description as a condition, generated images looks more realistic and more recognized objects can be seen in generated images. We also showed that how different generator network depth affects generated image quality by using inception score as an evaluation. The major contribution of this work is showing that conditional GAN can generate realistic and artistic images on a more complex dataset – Oxford paintings dataset despite the normally used CUB 200 dataset [23].

8. Future Work

There is a newly collected BAM! The Behance Artistic Media Dataset[24] which is more art focused. It contains short image descriptions/captions for 74,000 images from the crowd. It would be interesting to test our algorithm on that dataset because it contains more images but more importantly, it comes with text descriptions for painting images.

Our work shows that we can generate realistic images from text descriptions, but in order to put this for application, we need to generate larger images with higher resolutions. Currently the out image resolution is limited to 64 $\times$ 64. It would be interesting to see how image resolution can be increased by image super resulution methods proposed by Ledig*et al*. [15] or through a staged GAN proposed by Zhang *et al*. [26].

There are also methods proposed by Arjovsky *et al*. [1] and Gulrajani *et al*. [10] that shows how vanilla GAN's loss function have different problems causing GAN training to be unstable from theory. It would be interesting to see how their methods can be applied to conditional GAN, especially GAN-CLS method proposed by Reed *et al*. [19].

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. 2017.

[2] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014.

[3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. pages 1–9, 2014.

[4] E. J. Crowley and A. Zisserman. The State of the Art: Object Retrieval in Paintings using Discriminative Regions. *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[5] E. J. Crowley and A. Zisserman. In Search of Art. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8925:54–70, 2015.

[6] D. Duvenaud. Applications of GANs. *CSC 2541 Slides*, 2016.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style. *arXiv preprint*, pages 3–7, 2015.

[8] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014*, 2015.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. pages 1–19.

[11] N. Kalchbrenner and P. Blunsom. Recurrent Continuous Translation Models. *Emnlp*, (October):1700–1709, 2013.

[12] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676, 2017.

[13] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. pages 1–15, 2014.

[14] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-Thought Vectors. (786):1–11, 2015.

[15] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 2016.

[16] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. pages 1–7, 2014.

[17] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space.

[18] A. Radford, L. Metz, and S. Chintala. UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS.

[19] S. Reed, Z. Akata, X. Yan, L. Logeswaran REEDSCOT, B. Schiele, and H. Lee SCHIELE. Generative Adversarial Text to Image Synthesis.

[20] S. Reed, A. Van Den Oord, N. Kalchbrenner, V. Bapst, M. Botvinick, N. De Freitas, and G. Deepmind. GENERATING INTERPRETABLE IMAGES WITH CONTROLLABLE STRUCTURE.

[21] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. pages 1–10, 2016.

[22] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. *Nips*, pages 3104–3112, 2014.

[23] P. Welinder, S. Branson, T. Mita, and C. Wah. Caltech-UCSD birds 200. *CalTech*, 200:1–15, 2010.

[24] M. J. Wilber, C. Fang, H. Jin, A. Hertzmann, J. Collomosse, and S. Belongie. BAM! The Behance Artistic Media Dataset for Recognition Beyond Photography. 2017.

[25] S. Wu, S. Zhong, and Y. Liu. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, pages 1–17, 2017.

[26] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks.