IMAGE GENERATION FROM TEXT USING GENERATIVE ADVERSARIAL NETWORKS AND BERT

A PREPRINT

Nitinram Velraj*
Department of Biomedical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Raghavendar Suriyanarayanan
Heinz College, School of Information Systems
Carnegie Mellon University
Pittsburgh, PA 15213

January 29, 2020

ABSTRACT

Generative Adversarial Networks (GANs) have predominantly been used for generating new images given an existing image. It also has a wide utility in the field of generating textual data. Recent works have also demonstrated the ablity of using Deep Convolutional Generative networks along with use of text based captions to generate images out of them. In this project we explore some models which could be used to generate images based on the text data. Mainly, we look at the possibility of using different text representations of the captions to generate different types of flowers. We have also provided some recommendations on how to improve over our work.

***K*eywords** Generative Adversarial Networks, Bidirectional Encoder Representations from Transformers (BERT), Skip Thoughts

1 Introduction

The problem of generating images from text is intricate and difficult because of the highly multimodal nature of the underlying distribution. To understand the problem clearly, consider the sentence 'a flower with petals that are bright pinkish purple with white stigma'. There multiple different possible images which can correspond to this very sentence. These difference can arise from just the orientation of the image, different flowers which have similar descriptions and other small differences. This is exact problem we will try to address in this project.

Recent developments in machine learning and deep learning have given us the tools to address this problem. These developments describe methods to use GAN architectures effectively for this purpose and the training algorithm necessary to train such an architecture. We will be using one such architecture developed by Reed et al., 2016 to perform this task.

For this purpose we will be using data from the Oxford-102 Flowers dataset. This dataset consists of 102 flower categories. The flowers chosen to be flower commonly occurring in the United Kingdom. Each class consists of between 40 and 258 images. The text captions and the word embedding for these images are provided along with the work done by Reed et al., 2016 (ref).

The input consists of textual captions describing the image. For Example "This flower has small, round violet petals with a dark purple center" is a sample input at the test time. However in train time the input consists of five different valid representations for an image.

In the test time, the output is the output of the generator. However in train time the output from the genera-

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

tor is fed back to the discriminator . Hence at the train time the output of the generator is the image and the output of the discriminator is a 0/1.

The results are interpreted using qualitative metrics. The qualitative front we rely on is visual inspection of the images. There are also quantitative metrics (inception score) which can be used to measure the performance of generative models but we do not use them here.

2 Background Literature

2.1 Generative Adversarial Networks

We will rely heavily on the Generative Adversarial Networks (GANs) (Goodfellow et al.,) to perform our task. Our discriminator network will be tasked with distinguishing real images conditioned on text from synthetic images conditioned on text. Our generator network will be tasked with the task generating synthetic images to fool the discriminator. In mathematical terms our generator and discriminator will play the following game:

$$min_{(G)} \max_{(D)} V(D,G) = E_{x \tilde{} p_{data}(x)} logD(x) + E_{x \tilde{} p_z(z)}[log(1 - D(G(z)))]$$

Equation 1: GAN objective function

The solution to this optimization is exactly at $p_g = p_{data}$. This was proved by Goodfellow et al.,. Reed et al,. and suggests that in practice however maximizing the log(D(G(z))) works better than minimizing log(1-D(G(z))).

2.2 Deep Representations of Fine-Grained Visual Descriptions

Each text description and image used has an text embedding associated with it. We will use the pre-generated embedding provided by Reed et al., in this project. The embeddings are generated by using a deep convolutional and recurrent text encoder which learns the correspondence between a given text description and an image. The intuition here is that a text encoding will have a high score with an image from the corresponding class than it will have with images from other classes.

2.3 Skip Thoughts Sentence Embedding

Skip Thoughts is an unsupervised learning algorithm to learn the sentence level embeddings. There are three components in a skipthought model which are encoder, prev decoder and next decoder. All the three are recurrent neural networks (either GRU or LSTM). The encoder takes as input the current sentence and generates a representation of that sentence. The prev decoder takes as input the generated representation from the encoder and tries to generate the previous sentence. The error is propagated and continues till a good representaion of the previous sentence is learnt. The next decoder takes the earlier hidden representation and tries to generate the next sentence from that hidden representation. Even in this case the error is propagated until the right sentence is generated. This is illustrated in the figure below [10].

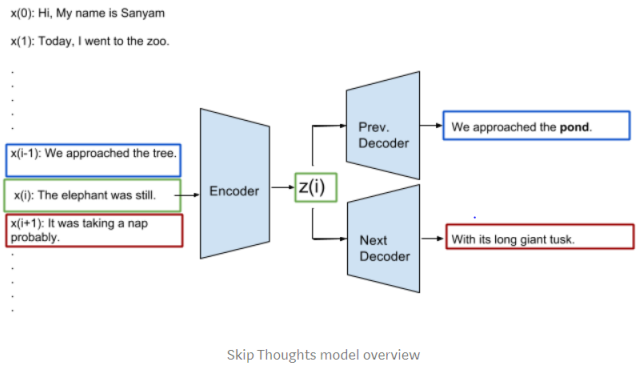


Figure 1: Skip thougths vector generation

2.4 BERT embeddings

We are using BERT, a word level embedding which uses attention. This was developed by Google AI. It leverages two unsupervised learning algorithms: masked language model (MLM) and next sentence prediction (NSP). BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. (2017). The BERT language model out performs previous language models is most tasks as shown by the developers in [9]. This is why we are evaluating the performance of BERT on the text-to-image task. The pre-trained model of BERT encoder that is available will be used in this project.

The reason for choosing BERT is that BERT embedding takes into account all the words surrounding the given words similar to the skip thoughts and the convolutional recurrent neural network. Also BERT offers a pretrained model which has been trained on the Wikipedia dataset and the Book corpus and hence allows us to avoid training the model from scratch and thereby saving a substantial training time. Also the BERT embedding has shown to be successful for the various applications in Natural Language Processing [9].

As shown in the image below the BERT model takes in individual tokens as parameters with some tokens being masked out. Also the tokens are preprocessed by splitting the words that end with "ing" like "playing" into "play" + "ing". Then each of the tokens are assigned a vocabulary ID from the look up table. Then the sentence embeddings which are used as distinguishers between the sentence are added and then followed by the positional embedding which indicate the position of each word in a sequence. Finally the BERT then passes these embeddings to a 12 layer Transformer encoder network with each output per token being used as a word embedding

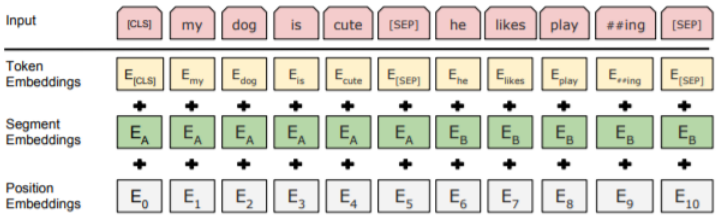


Figure 2: BERT pretrained word vector generation

2.5 Other works

As mentioned previously, we will use the work of Reed et al., as a main reference. But there are also other works that have been done in this area. Elman Mansimov et al., explores how attention can be used to build text to image generator.

The complete network used by Reed et al., is shown in Figure 3. A detailed description of the network architecture will be given in the methods section. Here, highlights of what is different in this work from that of Reed et al., will be provided. The Discriminator in the Reed et al., model consists of several layers of stride 2 convolution with spatial batch normalization (Ioffe Szegedy, 2015) followed by leaky ReLU. We will explore how the using different text represntation mechanism as described in class affects the performance of this entire network architecture. Other variations in network architecture will also be explored to see if we can improve the performance of this model if time permits.

3 Methods

In this paper we explore the approach of training a Deep Convolutional Generative Adversarial Network (DCGAN) which is conditioned on the pre trained embeddings which are obtained by using a hybrid character level convolutional recurrent neural network.

As can be seen from the Figure 3, obtained from the works of Reed et al., there are essentially 3 parts in the network architecture. The first being the generation of word vectors for the input text. There has been a lot of work done on converting the words which are very high dimensional to vector forms. Some of the popular methods include Bag of words , word2vec model. However the major problem with these models is the ability to not establish a semantic

relationship across words. Thus in this project we explore the method proposed by Reed et al. to use convolutional recurrent networks which can be trained to generate the word embedding.

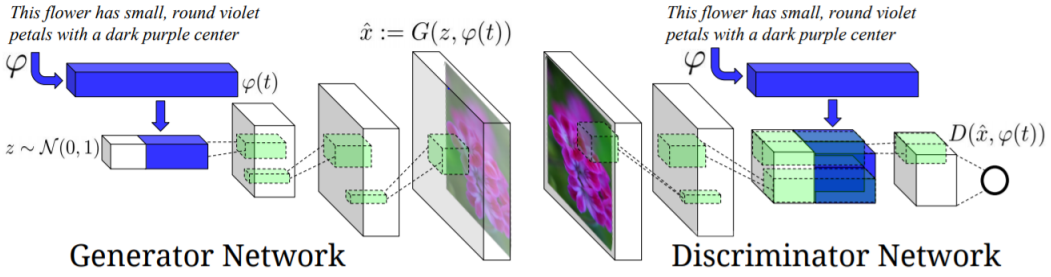


Figure 3: Network Architecture, (Reed et al.,)

The second important component of the network is the generator which acts as a feed forward network in the test time. Before feeding the embedded vectors to the generator we add a small noise prior ~N(0,1). Since word embeddings are high dimensional, we convert them to lower dimensions using a fully connected layer. The results are then passed through a deconvolution layer and unpooling layer to obtain the image.

The resultant image is then fed to the third important component which is the discriminator D. The discriminator takes the image as the input and outputs a binary function 0/1 based on whether the image is correct or not. The model uses a leaky ReLU for the activation inorder to avoid the dying ReLU problem and to speed up the training process.

The convolutional recurrent network used by Reed et al. to generate sentence embedding works well. In this work, we have made an effort to evaluate if the BERT encoding performs better. We believe the BERT embedding will perform better for this task as it has been shown to outperform most language models as shown in [9]. For the purpose of this project we are using the pretrained BERT model made available by Google. Each image contains 5 sentences describing the image. The BERT embedding model generates a word level embedding $_i$ for each word, i, in each sentence. Each word is embedded into a [1,768] length vector. To generate the sentence level embedding $\phi(t)$ for each sentence we take the average of all word embedding in a sentence. If L is the total length of the sentence, we have the equation shown below

$$\phi(t) = \frac{1}{L} \sum_{i=1}^{L} h_i.$$

Equation 2: Generating sentence level embedding from word level embedding

This results in a [5,768] matrix image embedding for each image. This is much smaller than the embedding generated by the Reed et in their model.

3.1 Baseline Results

Here we show our baseline results, which are the results obtained by Reed et al., (Figure 4) using the embedding method described in [6] and the ones we obtained using DCGAN architecture and skip thought embedding (Figure 5). Figure 4 was taken from the github link provided in the Reed et al., To obtain these results the model was trained for 600 epochs.

The DCGAN architecture used in both models are the same. The only change is in the embedding scheme. The method used in [6] is a word level embedding scheme. The experimental implementation shown in [7] uses skip thought vectors, which is sentence level embedding scheme as described in [8]. This model was trained for 200 epochs and the results obtained from this model for the same sentences are shown in Figure 5. Each sentence is embedded from the left to right using a [1,2400] dimensional vector and then from right to left using another [1,2400] dimensional vector and then the two vectors are concatenated to give a [1,4800] vector in order to capture the context of the sentence in a better manner than a unidirectional sweep. The same test sentences were used to generate Figure 5 as well. We did not train it further because of lack of resources. The model trained using skip thought vectors seems to not perform well when multiple colors are given as input. In the second sentence, the color yellow totally over powers the dark purple.

Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Figure 4: DCGAN model using Deep Representations of Fine-grained Visual Descriptions trained for 600 epochs, (Reed et al.,)

Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Figure 5: DCGAN model using skip thought vectors trained for 200 epochs

4 Results

The model we trained from scratch was done using BERT embedding and was trained for 200 epochs initially (Figure 6). Each word is embedded into a [1,768] dimensional vector and then all the embeddings are averaged to give a [1,768] dimensional representation of each sentence. The BERT model captures the information in the text input reasonably well. For instance, in the third set of images the description says lots of small pink petals. The generated images does have a lot of small petals, but the color information is not captured well. Skip thought does a better job at capturing both when trained for the same amount of time. This model also does not seem to capture color when multiple colors are given as input (like the skip thought model). Our assumption is that if trained for longer a period, some of these problems will be addressed. However, our initial assumption was that the model using BERT would perform better than the model trained using skip thought vectors embedding scheme when trained for the same amount of epochs.

Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Figure 6: DCGAN model using BERT trained for 200 epochs

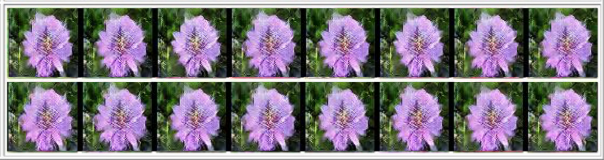
Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Figure 7: DCGAN model using BERT trained for 400 epochs

Figure 7 shows the results obtained by training the model with BERT embedding for an additional 200 epochs. The generated images now are much better than the ones obtained by using the skip thoughts embedding (trained for 200 epochs) and is comparable in performance to the baseline model given by Reed et al., in 200 fewer epochs. This validates our assumption.

5 Discussion and Analysis

In order to generate the above results we have used the DC-GAN as our primary model along with various text representations like skip thoughts, BERT and convolutional recurrent network. The DC-GAN is the primary network in the model and consists of the generator and the discriminator. In the generator, the text embedding generated by the model of our choice is passed through a linear layer. The output from the linear layer is then fed through a relu activation in order to remove the negative pixel values and for faster training. The output of the activation function is then fed to 4 deconvolutional layers with the filter size decreasing with every layer. Or in other words the size of the image keeps increasing by a factor of 2 as we pass through each layer. For the final deconvolution layer we use the 'tanh' activation function. However for the discriminator portion , the network is network is a start contrast to the generator. We use the a 4 layered convolutional network with relu as the activation and followed by a linear layer with a sigmoid activation since the discriminator predicts the probability of the image being fake. Also we normalize the output of the convolution at every layer and thereby making sure that the learning is much faster.

The above mentioned model parameters was used to obtain all the results. As noted previously the Figure 6 was generated using the BERT embedding which uses a 768 dimensional representation of each sentence. This model was trained for 200 epochs. Comparing to the baseline model skip thought which uses a 4800 dimensional vector for each sentence, we clearly make savings in terms of memory required to train the entire DC-GAN. This also significantly speeds up the training time considering the resource constraints. Also the output (.ckpt) file generated after performing the BERT embedding is much less in size (around 150 MB) compared to the baseline skipthoughts model(around 400 MB) .

In Figure 7, we have the model with bert embedding trained for 400 epochs. Here a significant increase in the performance was noted. We were able to capture a wider spectrum of colors like purple and yellow. Another important observation is that we were to generate multi coloured flowers. This shows that increasing the epochs increases the model performance and helps the bert embedding performs better than the skip thoughts model (Skip thought trained for more epochs may get better too).

However the model does have some limitations. The BERT pretrained model is trained on the Book Corpus and Wikipedia dataset. Hence it might not provide the exact representation of the flowers dataset at lower epochs. One way to solve this issue is by training for more epochs. However this might not always be a disadvantage since we noted that training the BERT based GAN model takes less time than the skip thoughts based GAN model. Another way to circumvent this would be to train the BERT model from scratch for the flowers dataset. This however has an effect on the training time of the whole model. The entire model may take days to train and may be infeasible considering the resource constraints. The pretrained model was trained using a TPU by Google.

The model also makes an assumptions that the 768 dimensional representation may capture the entire context in a much lower dimensional space than the skip thoughts model. This was not the case when both models were trained for the same amount of time. Also we assumed that BERT would perform better on the current flower dataset, since BERT is a highly used model in the NLP space and based on the previous results from BERT as mentioned in [9]. The generated images are all of size 64x64.

6 Future Work

There are some things which we did not cover in this work and would recommend to do in the future. The first is the fine tuning of BERT. Since we obtained the pretrained embeddings, fine tuning by training for more epochs would definitely improve the results. The second option is to train the bert from scratch with a higher dimensional feature representation. This would ideally bring a significant improvement in the performance at the cost of training time. The time taken to train the BERT from scratch is estimated to be around 4 days even if we run using a 12 cloud Google Colab TPU [9]. This will result in better generation in the image. The third option would be to train the model for higher epochs. For example in the baseline model by Reed et al., the model was trained for 600 epochs. However we have trained for only 400 epochs. Adding more training epochs would improve the accuracy.

Another option would be to take more than 5 captions per instead of restricting ourselves to 5. Since we have around 10 captions per image in the training data captions file, using all the 10 captions would potentially provide better image generations since the model would have seen additional descriptions for an image. Also before using the word embedding we could do some naive NLP preprocessing like tokenization by removing the stop words like "the", "a" , "and" etc and any other punctuation marks. Tokenization removes the unnecessary noise which would not be useful in final feature generation. Also we could perform lemmatization which involves choping the word to their roots so that we could get same representations for similar words in a sentence.

7 Work Split

The work was split throughout the project time. The literature survey was done by both Raghavendar Suriyanarayanan and Nitinram Velraj. The implementation and training (skip thoughts) of the baseline work was done by Raghavendar Suriyanarayanan. The BERT training and implementation was done by Nitinram Velraj. The Introduction section in the paper was written by Nitinram Velraj and Raghavendar Suriyanarayanan wrote the Methods section. The Background Literature, Results, Future Work and Work Spilt was written with the contribution of both authors.

References

[1] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[2] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *arXiv preprint arXiv:1511.02793*, 2015.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–58, 2016.

[7] Parth Neekhara. Text to image synthesis using thought vectors, January 2018.

[8] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NIPS*, 2015.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[10] Sanyam Agarwal. My thoughts on skip-thoughts, December.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]