Image Generation from Captions with Deep Convolutional Generative Adversarial Networks

Andac Demir
ECE Dept.
Northeastern Univ.
ademir@ece.neu.edu
https://github.com/andac-demir/Caption2Image-with-DCGAN

Abstract

In recent years, convolutional neural networks (CNNs) has seen a huge progress in computer vision applications. They greatly boosted the accuracy of image recognition models taking advantage of big data and high-performance computing resources. Meanwhile, deep generative adversarial networks (GANs) have begun to generate photo-realistic images in various categories including faces, maps, nature and commercial products. Creating intelligent models to generate images from natural language descriptions is a fundamental problem in many applications, such as art generation and computer aided design. This paper presents a class of deep convolutional generative adversarial networks (DCGANs) in translating visual concepts from characters in a text in the form of a single sentence human written descriptions to images. We train this network on generating flower images from text descriptions and aim to find a direct mapping from words to images by testing a set of constraints on the architectural topology of DCGANs to be stable to train and not memorizing the training data lazily.

keywords- generative adversarial networks, dcgan, image synthesis, generating natural images from text

1 Introduction

In this project, we aimed to translate texts in the form of single sentence human written descriptions into images. For example, "these white flowers are together on a long wine with purple leaves" or "white and yellow spiraling petals make up the grouping of flowers which have visible orange stamen". The problem of generating images from visual descriptions is an important object in Google Images search which returns images for a text query as well as for commercial purposes for instance suggesting merchandise based on the user search on e-commerce platforms.

We propose to solve this challenging problem in two steps: first learning a text feature representation that captures the important visual details and second using these features to generate images that a human might mistake for real (Reed et al., 2016). In training we use the Caltech-UCSD Birds dataset and the Oxford-102 Flowers dataset. The Caltech-UCSD Birds dataset consists of 200 bird species with 6,033 images. The Oxford-102 Flowers dataset consists of 102 flower categories, commonly seen in the United Kingdom. Each class consists of between 40 and 258 large scale images with variations in pose and brightness/contrast.

The reverse problem, generating texts from images, has been examined as an end-to-end framework in which the raw image is first encoded by a deep CNN which extracts the global visual feature vector that represents the semantic content of the overall image. Once the global visual vector is extracted, it is then fed into a recurrent neural network (RNN)-based decoder for caption generation which can be

Andac Demir is the only member of this project

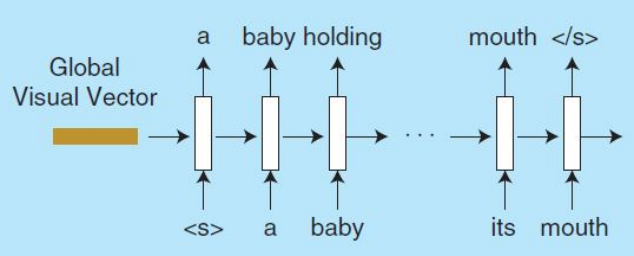

Figure 1: A CNN is trained for a 1,000-class image classification task on the large-scale ImageNet data set. The last layer contains 1,000 nodes, each corresponding to a category. The second last fully connected dense layer is extracted as the global visual feature vector, representing the semantic content of the overall images. This is then passed to the RNN-based decoder to learn to generate tokens.

chosen as a long-short memory network (LSTM) or gated recurrent unit (GRU). RNN-based caption decoder.

Global visual vector is passed to the RNN-based decoder and the sentence start symbol <s> is used as the first input to the hidden layer at the first step as shown in Figure 1 (He and Deng, 2017). Then the first word is generated from the hidden layer. Continuing this process, the word generated in the previous step becomes the input to the hidden layer at the next step to generate the next word. The text generation process iterates until until the sentence end symbol, </s>, is generated.

Recently, an attention based mechanism has been used to utilized this simple CNN encoder-RNN decoder approach (Luong et al., 2015). From a lower convolutional layer in the CNN, the attention based mechanism generates a set of visual vectors for subregions in the image. Then, in text generation, at each step of generating a new word, the RNN will refer to these subregion vectors and determine the likelihood that each of the subregions is relevant to the current state to generate the word. This is very useful for the RNN-decoder since it is fed with a contextual vector, which is a sum of subregional visual vectors weighted by the likelihood of relevance, instead of one global visual feature vector.

The reverse direction (image to text) decomposes word or character sequence with a chain rule model such that one trains the model to predict the next character/word conditioned on the image and all previous characters/words, which is a more well-defined prediction problem. However, in text to image problem the distribution of images conditioned on a text description is highly multimodal in the sense that we can generate many images that may correctly illustrate the text description (Reed et al., 2016).

They proposed to overcome this limitation by developing a simple text conditional convolutional GAN architecture and training mechanism that enables synthesis of bird and flower images from human-written descriptions.

To make the GAN training more stable and generate more realistic outputs yet unique and not lazily copying the training data, we make the following contributions mainly by implementing and evaluating a set of constraints on the architectural topology to develop a text conditional DCGAN:

- Batch normalization was used in both the generator and discriminator networks.
- Pooling layers were replaced with fractional strided convolutions in the generator network and strided convolutions in the discriminator network.
- Fully connected layers were removed for increased depth.
- In the generator network ReLU activation was used, except for the output which uses Tanh.
- In the discriminator network Leaky ReLU activation was used for all layers.

These items can be counted as architectural guidelines to develop a more stable text conditional DCGAN (Radford et al., 2015).

The remainder of this paper is organized as follows. Section 2 discusses GANs by introducing their network architecture and training details. Section 3 describes our methodology and introduces our model combining text conditional GANs and architectural guidelines of DCGANs. Section 4 reports our experimental results, evaluated on Caltech-UCSD Birds Oxford-102 Flowers datasets. Finally, Section 5 summarizes our contributions.

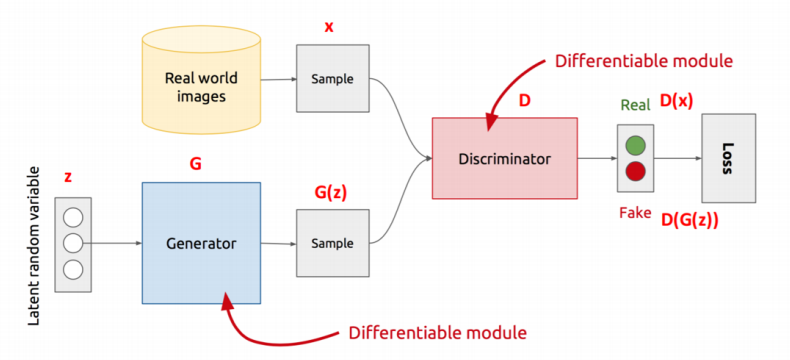

Figure 2: A high-level representation of the network architecture of the GANs first proposed in Goodfellow's paper (2014).

2 Background

In many use cases of applied math and engineering, we want to sample from a high dimensional and complex training distribution but there is no direct way to do this. One advantage of the generative adversarial networks is that no Markov chains needed neither for training the network nor for drawing samples and the major advantage is that they are the best in generating new samples. GANs draw samples from a simple distribution, for instance Gaussian noise and then learn a transformation from these simple distributions directly to the complex training distribution that we want using neural networks.

The way we train and learn this network is we look at this as a two-player game with adversaries: a generator network and a discriminator network. Generator network tries to generate samples that are intended to resemble those in the training distribution. It aims to fool the discriminator by generating real-looking images and the discriminator network inspects a sample and decides whether that is real or fake. Once the training is over, the discriminator network is no longer needed and we just generate samples with the generator network. The original paper (Goodfellow et al., 2014) explains this adversarial nets framework with an analogy: the generative model is like a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the cops, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles. In fact, the generator and discriminator are actually playing a game whose Nash equilibrium is achieved when the generator's distribution becomes same as the desired distribution.

We illustrate a simple GAN pipeline in Figure 2. First, it takes a random noise; this could be a vector or matrix or some dimension that we specify in the input. Then, we pass this through a generator network and get as output, a sample from the training distribution, hence for every input of random noise, we develop a learning strategy to correspond to a sample from training distribution. The idea is if the generator network is able to generate fake images that can successfully fool the discriminator network and the discriminator network can successfully distinguish between real and fake images, then we can have a good generative model in the sense that we can have images that look like images from the training set, but also different. Compared to variational autoencoders (VAEs), GANs generate more realistic images and the key to their success is they adopt an adversarial loss in learning the mapping such that the generated images are hard to distinguish from the images in the target domain (Mescheder et al., 2017).

Generative and discriminative networks are trained jointly in a minimax game formulation. This is the minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} V(\theta_g, \theta_d) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

The loss function was initially designed to be minimum over the parameters of the generative network and maximum over the parameters of the discriminative network. The first term is the likelihood of real data being real, and drawn from the data distribution $p_{data}(x)$ and the second term is the discriminator output for our generated fake data. Simply, the discriminator wants to maximize objective such that $D(x)$ is close to 1 for the real data and $D(G(z))$ is close to 0 for the fake data. If we minimize this objective over the parameters of the generative network, we can minimize this

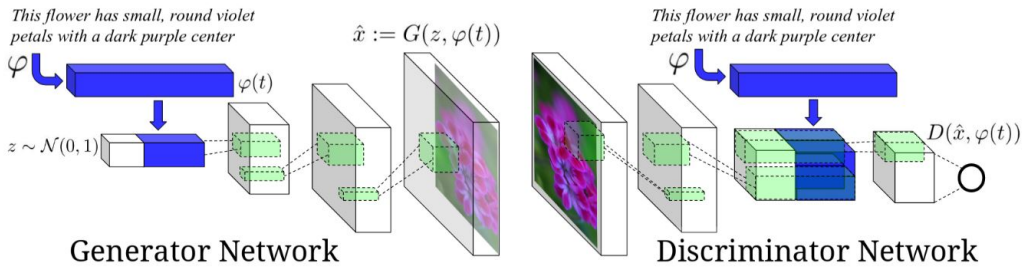

Figure 3: The network architecture of our text conditional convolutional DCGAN. Text input is compressed by an embedding layer $\varphi(t)$ and concatenated with samples from a noise prior. The resulting latent vector is used by both generator and discriminator. The generator network uses transpose convolutions to do upsampling and generate photo-realistic images, whereas the discriminator network is depth concatenated with image feature maps extracted after series of strided convolutions and the latent vector.

objective such that $D(G(z))$ is close to 1 so that the discriminator is fooled into thinking that fake image $G(z)$ is real. Similarly, if we maximize this objective over the parameters of the discriminative network, the discriminator can successfully distinguish between the real and fake data.

To train a GAN, we alternate between gradient descent on generator and gradient ascent on discriminator as shown in Equation 2 and 3:

$$\min_{\theta_g}[\mathbb{E}_{x\sim p_z(z)} \log(1 - D(G(z)))] \tag{2}$$

$$\max_{\theta_d}[\mathbb{E}_{x\sim p_{data}(x)} \log D(x) + \mathbb{E}_{x\sim p_z(z)} \log(1 - D(G(z)))] \tag{3}$$

We have only the second part of the minmax objective function for gradient descent on generator because the first part doesn't have the parameters of the generator network.

In practice it has been found this generator objective doesn't work well. The loss landscape of the existing generator objective is flatter at the region where generator isn't doing a good job in fooling discriminator and decreasing with a greater slope at the region where it is harder to distinguish the fake data from the real data (Goodfellow et al., 2014). Hence, the gradient term is dominated by the region where the generated data is already pretty good, whereas we want to learn greedily where the generated data is pretty bad. This slows the convergence of the training process and returns not realistic output images. To improve the learning what we use a slightly different objective function for the gradient where now we do gradient ascent instead. Instead of minimizing the likelihood of the discriminator being correct: $\log(1 - D(G(z)))$ which is what we had earlier, we flip it and maximize the likelihood of the discriminator being wrong: $\log(D(G(z)))$. This way, we have the same objective of fooling the discriminator, but now we have a higher gradient term for bad fake samples:

$$\max_{\theta_g}[\mathbb{E}_{x\sim p_z(z)} \log(D(G(z)))] \tag{4}$$

3 Method

We have a text conditional DCGAN that performs feed-forward inference both in the generator and the discriminator as demonstrated in Figure 3 (Reed et al., 2016). We draw 100 samples from the noise prior $z \sim \mathcal{N}(0,\ 1)$ and we use a fully connected layer of 1000 units, followed by batch normalization and leaky-ReLU with negative slope 0.2 to encode a text input to a vector of length 128. This embedding layer is represented with the symbol: $\varphi(t)$ where t is the input text. We concatenate the samples drawn from the noise prior and the embedded text vector in a latent vector of length 228. The synthetic image generated from the generator is denoted with the symbol $\hat{x} := G(z, \varphi(t))$.

Following the text embedding layer, the rest of the generator network is a series of five transpose convolutions (other names: upsampling, upconvolution, fractionally-strided convolution) with kernel size 4, stride 2 and padding 1. Some papers wrongly call these layers "deconvolution", which refers to the opposite convolution that reverses the effect of convolution on recorded data. As has been mentioned in the introduction, all transpose convolutions are followed by batch normalization and leaky-ReLU with negative slope 0.2 except for the output which uses Tanh.

In transpose convolutions, input elements are the weights that we multiply the upsampling matrix. Once the upsampling matrix is multiplied separately for each input value, they are placed on the

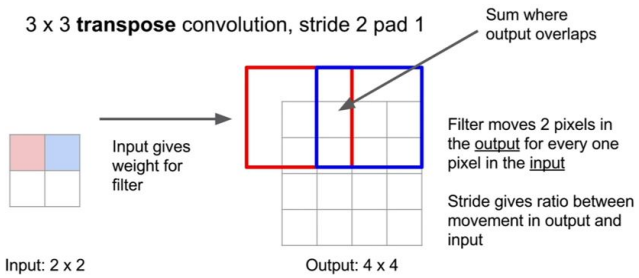

Figure 4: Learnable upsampling: Transpose convolution.

Algorithm 1 Text conditional DCGAN training algorithm. Optimized with minibatch SGD using learning rate α. BCE is used to denote binary cross entropy. In evaluating the BCE between the discriminator output and the text input, right texts are labeled as 1 and mismatching texts are labeled as 0.

Inputs: minibatch images x, matching text t, mismatching text $\hat{t}$ and number of batches S.

for n=1 to S **do**
- $h \leftarrow \varphi(t)$ {Matching text is encoded.}
- $\hat{h} \leftarrow \varphi(\hat{t})$ {Mismatching text is encoded.}
- $z \sim \mathcal{N}(0,\, 1)$ {Samples drawn from zero mean unit variance Gaussian.}
- $\hat{x} \leftarrow G(z, h)$ {Generator network generates fake images.}
- $s_r \leftarrow D(x, h)$ {BCE between discriminator output for real image and right text input pair.}
- $s_w \leftarrow D(x, \hat{h})$ {BCE between discriminator output for real image and arbitrary text input pair.}
- $s_f \leftarrow D(\hat{x}, h)$ {BCE between discriminator output for fake image and right text input pair.}
- $L_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$ {Discriminator loss.}
- $D \leftarrow D - \alpha \cdot \partial L_D / \partial D$ {Updating the discriminator network.}
- $L_G \leftarrow \log(s_f)$ {Generator loss.}
- $G \leftarrow G - \alpha \cdot \partial L_G / \partial G$ {Updating the generator network.}

end for

output matrix, after slided by some number of strides (Dumoulin and Visin, 2016). Over the areas of intersection, values are summed as shown in Figure 4.

In the discriminator network, first we perform a series of four convolutional layers with with spatial batch normalization followed by leaky ReLU with negative slope 0.2. All convolutional layers have kernels size 4, stride 2 and padding 1. The extracted visual feature representation has spatial dimensions 4 x 4 and depth 512 channels. At this stage, we again encode the input text using a fully connected layer of 1024 units followed by batch normalization and leaky ReLU with negative slope 0.2. The low dimensional embedded text is a vector of length 128. We replicate this vector spatially to match the spatial dimensions of the visual feature representation tensor so they can be concatenated. The output tensor after depth concatenation is passed to the final convolutional layer with kernel size 4, stride 1 and zero padding. It is fed with 512 channels and returns a single channel, three dimensional tensor of spatial dimensions 1 x 1 which is squeezed to a scalar value after passed through a leaky ReLU activation with negative slope 0.2.

The naive way to train a text conditional DCGAN is to take (text, image) pairs in the input as joint observations and train the discriminator to judge pairs as real or fake; however in the naive way of training the discriminator assumes input pair is either an embedded text that matches with the real training image or a synthetic image with an arbitrary embedded text. Yet in practice, learning dynamics are different. There is a third type of input that is real images with mismatched embedded text, which the discriminator must be trained to score as fake (Reed et al., 2016). Briefly, there are two sources of error in training: generated images are not realistic (for any text input) and realistic images don't match the semantics.

To improve the mapping from the input text to the output image, we add a third term in the loss function. Along with real image - corresponding text and fake image - arbitrary text pairs, we also train the network with real image - mismatched text pairs. This improves the matching between the text and the image in addition to generating more realistic images. We summarize the training

(a) Real flower images from Oxford-102 database.

(b) Fake flower images generated from the captions corresponding to the real images on the left.

Figure 5

procedure in Algorithm 1. First we encode the right text and the mismatching text to compress the text input in an embedding layer. After encoding the text, we draw samples from a zero mean, unit variance Gaussian and we concatenate the encoded text with the sampled noise in a latent vector. Generator network generates fake images from the latent vector. Then we compute three scores from the discriminator. Those are: s_r, which indicates the score of associating a real image and its corresponding sentence, s_w, which measures the score of associating a real image with an arbitrary sentence and s_f, which is the score of associating a fake image with its corresponding text. We evaluate these scores by finding the binary cross entropy between the discriminator output and text label, which is 1 for the matching text and 0 for the mismatching text. After computing the loss objective for the discriminator, we update the weights of the discriminator network using minibatch SGD for simplicity and likewise we update the weights of the generator network.

4 Experimental Results

In this section we present the results on the Oxford-102 dataset of flower images and the CUB dataset of bird images. Oxford-102 has 82 train + validation and 20 test classes, while CUB has 150 train + validation classes and 50 test classes. During the training, we used 5 captions per image for both datasets.

Results on the Oxford-102 Flowers dataset can be seen in Figure 5. Generated flower images with the right text are plausible and we can see that they match the real images in the sense that we see the same colors and similar petal types between corresponding flowers. The major problem observed is the low resolution in the generated images which occurs because of the early stopping in our training. Since our computational resources were scarce and time was limited, we trained our model for 120 epochs. In most GAN applications, model training lasts more than 600 epochs. In Figure 6 we see a mismatch between the real and generated bird images. We speculate that it is easier to generate flowers, perhaps because birds have stronger structural regularities across species that make it easier for the discriminator to spot a fake bird than to spot a fake flower (Reed et al., 2016) which results in slower convergence and poor performance in optimization while training the generator network.

We also investigated the extent to which our model can separate style and content. By content, we mean the visual attributes of the bird itself, such as shape, size and color of each body part. By style, we mean all of the other factors of variation in the image such as background color and the pose orientation of the bird. Embedded text vector mainly covers the content information and typically nothing about style. Captions in the dataset don't describe the background or the bird pose. Therefore, in order to generate realistic images, we concatenated the embedded text vector with samples drawn from a zero mean, unit variance Gaussian noise and created a latent vector. GAN must learn to use noise samples to account for style variations. This latent vector is different at every trial. As a consequence, we can generate unique, but also plausible images from the same text using the same pre-trained model as as illustrated in Figure 7.

(a) Real bird images from CUB database.

(b) Fake bird images generated from the captions corresponding to the real images on the left.

Figure 6

(a) Generated flower image given: "bright droopy yellow petals with burgundy streaks, and a yellow stigma"

(b) Generated bird image given: "this bird is different shades of brown all over with white and black spots on its head and back"

Figure 7

5 Conclusion

In this work we improved the stability of a simple and effective GAN model for generating images based on detailed visual descriptions. The DCGAN architecture we propose gives more stable results than the existing GAN architecture with a naive way of training. Yet, there are still some forms of model instability remaining that we still see some artifacts and blurriness in the generated images, which is an optimization problem and not much related to the network architecture. We optimized our network with SGD; however a Quasi-Newton method L-BFGS, which is an unconstrained, non-linear optimization that steers its search through variable space by using an estimate to the inverse Hessian matrix, can give better optimization results and more realistic fake images. Although L-BFGS is memory greedy and computationally intensive, since it is a second-order method, it finds the local minima better than the variants of stochastic optimization methods, resulting in lower loss (Zhu et al., 1997).

Using Caltech-UCSD Birds and Oxford-102 Flowers datasets, we demonstrated that the model can generate many plausible, photo-realistic images of a given text caption. We evaluated the generalizability of our model generating images with multiple objects and variable backgrounds. In future work, we aim to improve the model to generate higher resolution images using L-BFGS optimization and add more types of text.

6 Acknowledgments

We are fortunate and thankful for all the advice and guidance we have received during this work, especially that of Prof. Yanzhi Wang and the course TA Kaidi Xu.

References

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.

X. He and L. Deng, "Deep learning for image-to-text generation: A technical overview," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 109–116, Nov 2017.

M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks," *arXiv preprint arXiv:1701.04722*, 2017.

V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.