Text-Adaptive Generative Adversarial Networks

Emma Qin, Will Schwarzer, Kyra Wilson, Orlando Zuniga

October 16, 2019

Contents

1 Literature Review **1**
1.1 Network Architecture 1
1.1.1 Generator 2
1.1.2 Discriminator 3
1.2 Datasets 5

2 Goals **5**

3 Timeline and Role Breakdown **6**

1 Literature Review

Image synthesis is a difficult problem in computer vision, but the use of Generative Adversarial Networks (GAN) has led to great progress being made towards generating realistic images [1]. GANs typically consist of a generator, which is able to generate images, and a discriminator, which is able to discern whether an image is real or made by a computer. These two parts of the network typically work against each other, with the generator trying to make more and more accurate images to fool the discriminator and the Discriminator trying to get better and better at predicting what kinds of images are real or not.

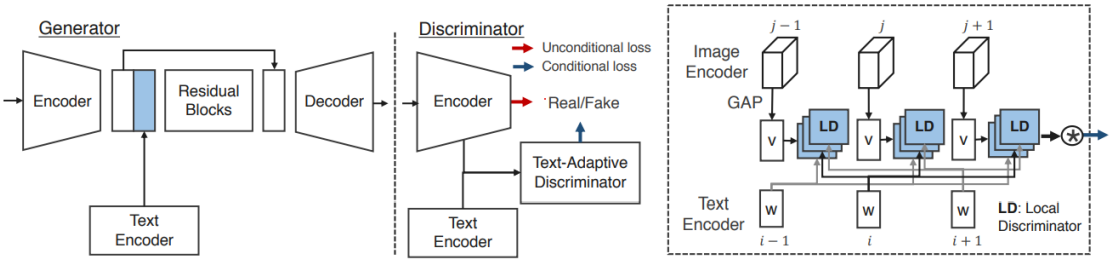
In their basic form, the images generated by GANs are not conditioned on any information other than random noise: the generator simply takes in a random noise vector as input, and generates a random image as output [1]. This means that the user is unable to control in any way what kind of image a GAN generates, aside from modifying the training data. Given this limitation, one of the most exciting extensions of the classical GAN paradigm is conditional GANs [1, 2], where the generator is instead given some input vector y as a starting point for its image generation, which could be anything from a class label for the image to be generated [2] to a free-form natural language description of the image [3]. When the discriminator is judging any given image, it then also learns to judge how well the image matches the condition it is associated with, based on the condition/image pairs in the training data. The feedback thus produced by the discriminator allows the generator to generate images that are not only lifelike, but also accurately depict the given input.

The problem that we propose working on in this project is a slight modification of this problem of generating images that correspond to natural language inputs: specifically, we propose instead modifying preexisting images in order to match a new text description that we supply. To do this, we will still feed in the text description to both the generator and the discriminator, and allow the discriminator to judge the generated images based on how well they match the description; however, we will also input a real image to the generator for it to base its generated image on. This approach is the subject of the paper we're replicating [4], as well as several previous papers.

1.1 Network Architecture

One proposed model for achieving natural language image modification is the Semantic Image Synthesis GAN (SISGAN) [5]. The SISGAN architecture is as follows: the generator consists of an encoder, a series of residual block

transformation units, and a decoder; the Discriminator consists of multiple convolutional layers which create image feature representations of a desired size and then predict the authenticity and correctness of an image and its text description. The network that we will be replicating in this project, the Text-Adaptive GAN[4], uses a slightly modified version of this generator, as well as a significantly modified discriminator.



(a) GAN structure (b) Text-adaptive discriminator

Figure 1: The network architecture used in [4].

1.1.1 Generator

In the generator, the encoder is used to make vectorized representations of images and natural language text descriptions. For images, this is done using a convolutional neural network (CNN), while text encoding is done using a bidirectional GRU (gated recurrent unit, a type of recurrent neural network (RNN)). In SISGAN, the RNN is pretrained before being used with the data of interest, but the TAGAN RNN is trained from scratch, though its embeddings are supplemented with pretrained embeddings from fastText [6]. While the authors of [4] do not state why they chose not to pretrain their RNN as was done in previous studies, we hypothesize that one possible reason for this is that the language used in the image captions is specialized enough that pretraining the neural network on more general natural language may not provide any benefit for the task at hand, and may even worsen its performance.

Another alteration made for text encoding with TAGAN is that conditioning augmentation [7] is used in order to generate more text-image pairs and minimize any outlying data in the training set. Conditioning augmentation uses a similar principle to variational autoencoders (VAEs): rather than encoding each raw data point to the same vector every time, the data are instead first encoded into vectors of normal distributions, from which individual representations are then sampled. This serves to "smush" the representations out to fill more of the representational space, forcing the generator to process textual inputs more flexibly, thereby allowing it to interpolate to novel sentences more easily.

Once both the image and the text have been encoded, the encodings are concatenated to create a visual-semantic text representation to be used as input for the residual blocks. These blocks, which are the part of the generator that actually modifies the image representation to match the language representation, are sets of standard multilayer perceptrons (MLPs, aka basic feedforward neural networks) with one particular quirk: each residual block saves the input vector it is given, and adds it to the output vector that it produces. This means that what each residual block is actually learning is the difference (residual) between the input and the output - i.e. the modification to be made to the input. The authors in Dong et al., 2017 [5], who also used residual blocks, assume that this will allow the generator to more easily keep description-irrelevant parts of the image, such as the background, unmodified.

Finally, the decoder returns the latent feature representation to an image of the desired size by passing it through several upsampling layers. A batch of such generated images is then passed to the discriminator, which

determines the validity of the images (whether or not they're real, and whether or not they match their descriptions); in the same epoch, a batch of real images is similarly given to the discriminator for validity checking.

1.1.2 Discriminator

In its basic structure, the discriminator used in this paper parallels that of Dong et al., 2017 [5]. Like all GANs, part of the discriminator is unconditional, i.e. it ignores the natural language descriptions of each image: it simply encodes the image (using a CNN, like the generator) and classifies it as either real or fake. However, like all conditional GANs, it also has a conditional (or "text-adaptive") part to it, which classifies whether or not the image accurately depicts the description associated with it. This is where the GAN architecture employed by this paper diverges most significantly from previous work. Whereas Dong et al., 2017 [5] simply encode the description as a whole, then ask the discriminator to classify the combined image and text representations, this paper uses a much more complicated word-level discriminator.

The basic idea behind this word-level discriminator is to consider each individual word of the description, determining the semantic fit between each word and the image in question. To do this, the paper uses a local discriminator, which takes in an image representation and a word representation as input, and returns a value representing whether or not the word in question (for example, "brown", or "bird") appears to be present in the image; this local discriminator is then used on the image with each individual word of the description, and the results of each of these trials are added together to determine the overall match between the image and the description. The goal of this finer-grained discriminator is to provide the generator with more precise feedback: rather than only being able to backpropagate the information that a given bird did not match its description, the word-level discriminator will be able to tell generator that the bird was not sufficiently black, or that its beak was too long.

That said, the above description glossed over a few details. First, the results of each local discrimination test are not just added together to get the overall result; instead, the network uses an attentional system to add together these results, allowing it to place more weight on words that it has learned are important. (For example, perhaps it has learned that it should really pay attention to whether or not the image matches the word "brown", but the word "the" tends to convey less information.) This turns the overall discrimination score into a weighted sum of the invidividual tests, where the weight for each word's local test is learned over time.

In addition, there is a bit of finesse in the exact image representations that the local discriminator takes as input: in fact, there are three different local discriminators, each taking in a different type of image representation! These different image representations represent different levels of detail in the image: "conv3" is the finest-grained (i.e. highest-dimensional), while "conv5" is the coarsest. These odd names stem from the details of how the representations are created by the network's CNN.

CNNs generally produce low-dimensional image representations from high-dimensional images by separating blocks of convolutional (sparsely connected) layers with pooling layers, which compress the dimensionality of the representation - often by a factor of 2 each time. (Specifically, they usually compress the width and height dimensions of the representation, while increasing its depth.) Thus, if conv3 is the last convolutional layer of the third convolutional block of the CNN, and if the image representation at this point has dimension (w, h, d), conv4 might have dimension (w/2, h/2, 2d), and conv5 would be (w/4, h/4, 4d). Generally, a CNN feeds an image through all of its convolutional blocks, before using several dense layers at the end to compress the representation fully to the desired size. However, you can also short-circuit the full process by instead converting the results of each convolutional block directly to a (1, 1,) representation. This is done using global average pooling (GAP), where the values of each (w, h) slice of the representation are averaged together to get a single number, turning the (w, h, d) representation in conv3 into a (1, 1, d) representation, while conv4 and conv5 produce (1, 1, 2d) and (1, 1, 4d) representations, respectively[8]. Since CNNs generally produce more and more coarse-grained representations while moving through their layers, it is reasonable to assume that the representation taken from conv3 will have the best low-level detail, while conv5 will be a more high-level representation of the image, as desired. The authors confirm this in the ablation study shown in Figure 2.

Once these different representations have been produced from the CNN, each of them is fed into its own corresponding local discriminator. This discriminator then attempts to determine whether or not the image at that particular scale matches the given word well: for example, conv3 might work well for determining whether or

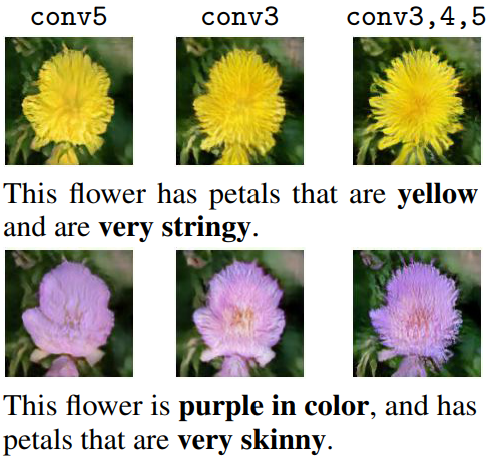


Figure 2: Results of only using certain image representations in the discriminator.

not a flower's petals are indeed "stringy", while conv5 might be good at determining whether or not the flower is purple. The results of these three local discriminators are then summed together attentionally, the final result of which is used as the value of the overall local discrimination test for that individual word. Once the discriminator has determined the total word-level fit of the description and the image, feedback on both the conditional and unconditional scores is provided to the generator, to allow it to produce more realistic and text-accurate images, while the discriminator receives feedback on whether or not the image was correctly classified as being real/fake and matching/not matching the description. The authors of the present paper show that their enhancements, especially (they claim) the word-level discriminator, result in significantly more accurate and natural (i.e. precise) image modification, as shown in Figure 2.

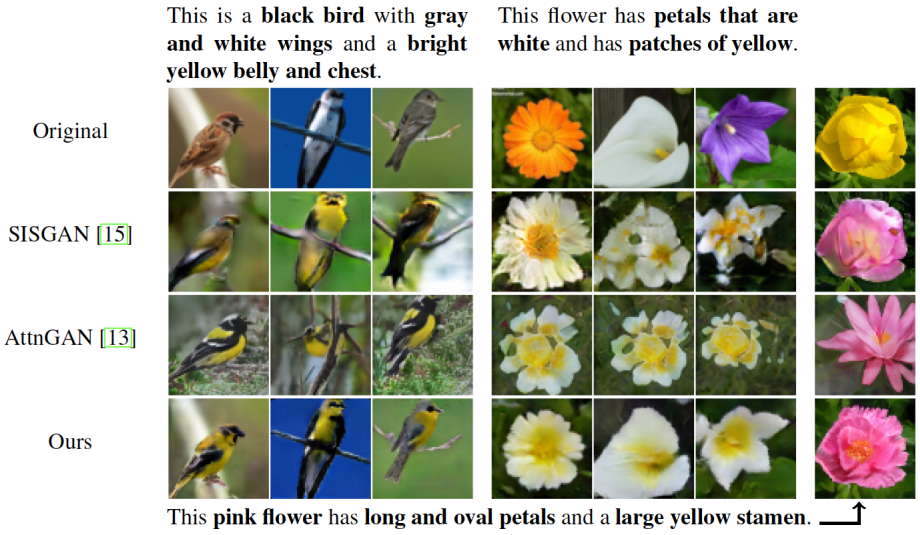


Figure 3: Qualitative results of the three methods.

1.2 Datasets

The datasets used to train and test TAGAN are CUB[9] and Oxford-102[10] datasets. These are both fairly standard and commonly used image datasets for machine learning tasks and in particular the predecessors of TAGAN[5]. For this project, we have chosen to focus on replicating the results using CUB as our primary goal, so further discussion of datasets is limited to that one. The CUB dataset includes 11,788 images of birds in 200 different categories, as well as annotations containing 15 part locations, 312 binary attributes, and 1 bounding box for each image. Because this paper aims to alter images based on descriptions that are natural language rather than sets of features, the CUB dataset must also be supplemented with natural language captions. The captions the authors used (which we also intend to utilize) were created for a different investigation which also employed CUB and Oxford-102 datasets[3]. These researchers created 10 natural language captions for each image in the dataset according to the annotated features in CUB. The annotated images and text are then used as inputs for the generator. In our replication, 80% of the data will be devoted to training and 20% will be devoted to testing. Furthermore, the authors of both the present paper and Dong et al., 2017 used standard data augmentation techniques for images, namely, cropping, rotation, and flipping, which we will also replicate.

2 Goals

Our major goal of this project is to recreate GAN structure, which includes generator and discriminator, and use the recreated GAN structure to modify and produce images according to text input. Due to time limitation, we will first focus on CUB dataset[9], and extend to Oxford-102 dataset[10] later. There are several measures used to test the performance of GAN, and in this project we will first focus on the L_2 error. L_2 error stands for Least Square Errors. It is used to minimize the error which is the sum of all the squared differences between the true value and the predicted value.

$$L_2 = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2.$$

Besides minimizing L_2 error in the project, we will try to replicate the result of the original paper on CUB dataset[9], shown in Figure 4, which is giving an image and annotation on birds, modifying the original image without changing the background of the original image.

Figure 4: Qualitative results of GAN on CUB dataset.

Given enough time, our primary reach goal would be to compare our GAN approach to the two baseline methods (AttnGan and SISGAN). AttnGAN is an Attentional Generative Adversarial Network (AttnGAN) that allows attention-driven, multi-stage refinement for fine-grained text-to-image generation [11]. In other words, AttnGAN can synthesize fine-grained details at different sub-regions of the image by paying attentions to the relevant words in the natural language description [11]. We would start by getting the L_2 error for each approach and comparing

it to ours. When we compare all three methods, the results should show that TAGAN outperforms the two baseline methods. Our results should also show that SISGAN and AttnGAN have similar performance based on their L_2 error. In addition to this, we would also like to compare the qualitative results for the three methods as seen in Figure 3.

Our second reach goal would be to use the Oxford-102 dataset [10] that contains the flower images. We would measure L_2 error for each approach and compare it to ours. The TAGAN approach should show a lower L_2 error in this dataset than the two baseline methods.

Our last reach goal would be to conduct a human evaluation on Amazon Mechanical Turk to gather quantitative results that measure the three GAN methods based on accuracy and naturalness [4]. This user study requires that we randomly select 10 images and 10 texts from the test set and produce 200 outputs from the dataset for each method. All output images would be resized to 64x64 to prevent the users from evaluating based on sharpness. The workers would then evaluate the images based on two criteria: whether the visual attributes (colors, textures) of the manipulated image match the test, and the background is preserved and whether the manipulated image looks natural and visually pleasing. We would then present each criterion as a score of accuracy and naturalness, where the lower number is better. This user study should conclude that the TAGAN method has a higher accuracy and naturalness score than the two baseline methods.

3 Timeline and Role Breakdown

First and foremost, here is the timeline of work distribution in Fall term. All the work listed in the table should be done by the beginning of the listed week.

Week #	Emma	Will	Kyra	Orlando
5	Setup libraries, finish project proposal	Setup libraries, finish project proposal	Get datasets, setup libraries, finish project proposal	Setup libraries, finish project proposal
6	Look at GitHub for SISGAN, make any alterations to the proposal if needed			
7	Image-decoder	Main	Making train and test splits,figuring out how to load them w/ PyTorch	Image-encoder
8	Plan for discriminator	Residual blocks	Text-encoder	Connect encoder and decoder/testing
9	"Good" progress on networks (encoders, decoders (generator), discriminator), "working-proof of concept complete", everyone focus on integrating parts of generator network			
10	Prep for short presentation			

Then, winter term timeline is more general, because the group will focus on the discriminator and general testing in winter term, which will be determined later this term.

Week #	Goal
1-3	3 people on Discriminator, 1 on reproducing SISGAN and AttnGAN as long as Emma still thinks its a good plan after 8^{th} week Fall term
4-5	Network done and training finished, start running experiments, start presentation
6	Reproduce L_2 error and Figure 3
7	Reach goals (potentially), run with flower data set (potentially), practice presentation for Anna
8	Entire network done, run some of the experiments

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In Proceedings of NIPS, pages 2672– 2680, 2014.

[2] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

[3] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text-to-image synthesis," in ICML, 2016.

[4] S. Nam, Y. Kim, and S. J. Kim. Text-adaptive generative adversarial networks: manipulating images with natural language. In Advances in Neural Information Processing Systems, pages 42–51, 2018.

[5] H. Dong, S. Yu, C. Wu, and Y. Guo, "Semantic image synthesis via adversarial learning," in ICCV, Oct 2017.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," arXiv preprint arXiv:1607.04606, 2016.

[7] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in ICCV, 2017.

[8] A. Cook. (2017). Global Average Pooling Layers for Object Localization. [Blog] Available at: https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/ [Accessed 16 Oct. 2019].

[9] C. Wah, S. Branson, P.Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.

[10] M.E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in ICCVGIP, Dec 2008.

[11] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. (2018). "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in Conference on Computer Vision and Pattern Recognition.