Story Illustration using Generative Adversarial Networks (GANs)

Grace Kim, Priscilla Lui
{yek1354, pslui88}@stanford.edu

June 2, 2021

1 Motivation

GANs are an incredible ML tool for generating novel, realistic images. GANs have even been extended to generate images from text input. The application of text-to-image GANs that we are exploring is transforming story narrative into illustrations, akin to a DIY storybook. To illustrate a story, one needs to generate a sequence of images that accurately depict their corresponding textual elements (coherency), but also maintain a cohesive style and character design throughout (consistency). We are especially motivated by the creative applications for end users, such as a web-app for fostering creativity in children.

2 Method

StoryGAN (2) is an existing recurrent GAN architecture for story illustration. In a previous quarter, one of us worked on a custom dataset to train StoryGAN on, but the outputs were a bit disappointing. This quarter, we tried to achieve better results in two ways: 1. We supplemented our dataset with story captions that are more descriptive and semantically faithful to the images and re-trained StoryGAN on this new dataset. 2. Instead of using a recurrent network that trains on 5 text/image pairings at a time, we used DF-GAN (4) pre-trained on COCO, which takes a single text to a single image, and used it to inference iteratively on each text/image pairing to form one story.

2.1 Dataset

Our dataset consists of captioned images organized into stories derived from Microsoft's Visual Storytelling Dataset (VIST) (1). Each story is a sequence of 5 images and 1 sentence caption per image. The original VIST consisted of high-resolution photos, but the version of VIST we used was inherited from a previous project. The images in this inherited dataset are the original images but style-transferred to lend a storybook aesthetic and downsized to 128x128 to increase training efficiency.

As for the text captions, VIST provides two different annotations files: story-in-sequence (sis) and description-in-isolation (dii). The sis annotations describe a sub-dataset of VIST that contains 49,000 stories with captions that flow like a story, whereas the dii annotations describe another sub-dataset of 24,000 stories but with captions that describe each picture in isolation (Fig. 1). Each dataset was split 80/10/10 into Train/Test/Val. The dataset we inherited was the sis dataset, but we hypothesized that our baseline StoryGAN's poor performance could be partially attributed to the poor semantic consistency between images and their sis caption. Thus, we built a second dataset consisting of only stories with description-in-isolation captions to leverage more descriptive text.

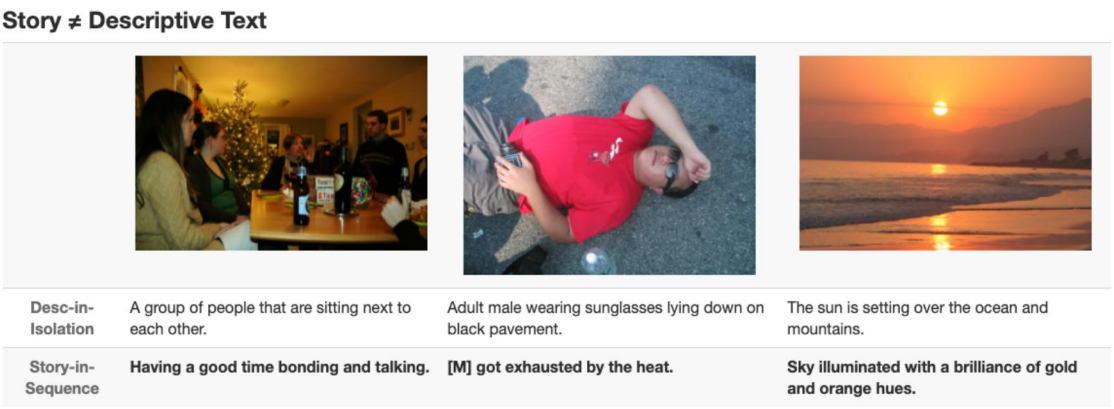

Figure 1: This is an example of three selected images that happen to have story-in-sequence and description-in-isolation captions. The dii captions are much more descriptive of the image content.

2.2 Data Processing

In terms of code contribution, we wrote scripts to parse the dii annotations file and generate the encodings for each dii caption. Since the dataset we inherited used CLIP (3) from OpenAI to transform the raw text to encodings, we did so too. CLIP is a state-of-the art encoder for extracting visual information from text. We used CLIP to tokenize each caption into 77 tokens since the CLIP encoder required that length. All the captions were tokenizable except for 3 captions that upon inspection, turned out to be the same sentence repeated multiple times, so we manually fixed those captions. Next, we used the CLIP encoder to turn the tokenized captions into 512-dim tensors. We saved the encodings in a Pytorch file. The second part of processing was creating a data structure to map image IDs to their encodings and organize them into their sequences (not a story, which only applies to the sis data). The final format was a list of lists, where each inner list represented a sequence of 5 tuples that looked like: (image id, index into the encodings file). We performed this process on the three JSON annotation files for train/test/val.

A challenge we had was that our GPU easily hit its memory limit, so we improved the script to automatically generate the encodings in batches and combine them into one large pytorch file in the end. In the following section, we describe in more depth how we used the sis and dii datasets to train StoryGAN.

2.3 StoryGAN Experiments

2.3.1 Training on Story-in-sequence (sis)

First, we describe what we worked on up until the milestone with respect to StoryGAN. At this point, we were only working with the sis dataset. Despite inheriting most of the starter code, we had to spend time familiarizing ourselves with the codebase. Further, we modified the code in two ways before training began: (1) Some of the image files were corrupted, and the training code would break. The reason for this was that the script to generate the annotations only checked if image files existed, not that they were open-able. Thus, we wrote a script to regenerate a cleaned version of the annotations file. (2) There was a remnant of an experiment involving zero-ing out the 'story context', which holds the concatenation of the text encodings for a whole story. The previous team noted that it worsened performance, so we undid this modification. After making these code changes,

we trained a baseline StoryGAN on just the sis dataset on a GCP instance with 4 CPUs and 1 Tesla K80 GPU for 16 epochs. We stopped it at 16 epochs because we observed the training loss plateauing and the images suffering from mode collapse.

2.3.2 Training on Description-in-isolation (dii)

After the milestone, we generated the dii dataset and started training StoryGAN again on this dataset. Rather than starting with random weights, we loaded in the weights from the 16th epoch from training StoryGAN on the sis dataset. Since many images are shared across the two datasets, we hoped that the introduction of more descriptive text captions would improve the GAN's performance. We trained for 3 additional epochs on the dii dataset but was cut short at 19 epochs because our GPU reached its memory limit every time we tried to train more. If we had more time, we would debug this issue and train longer.

2.4 DF-GAN Experiment

Recent literature has shown that DF-GAN outperforms many existing text-to-image GAN models. Some unique elements of DF-GAN architecture that may account for this improvement in performance are: (1) Instead of stacking generators and discriminators, DF-GAN utilizes one generator and one discriminator. Research has shown that the currently commonly used framework of stacked GANs actually causes the model to rely heavily on the output of the first generator since all subsequent generators rely on that. (2) DF-GAN utilizes MA-GP (Matching-Aware Gradient Penalty) in order to place real image, text-matching data points at the minimum of the loss function of the discriminator. (3) DF-GAN uses one-way output to only give the gradient that points to the real and matching inputs after backpropagation.(4) Lastly, DF-GAN takes a novel approach to more effectively fuse the text and image information through DFBlocks (Deep text-image Fusion Blocks).

For this project, we utilized the DF-GAN library found on this github repo (https://github.com/tobran/DF-GAN). We used the weights provided by the authors obtained by training on the COCO dataset and then sought to see how well it would perform when provided new user input. To test the model, we created a streamlit app, an interactive platform for users to input a text description of the image they'd like to see outputted (Fig. 2). We then made significant modifications to the code so that the model could inference on this new data input in real-time to output the corresponding, generated image. We also modified DF-GAN to accept VIST captions so that we could compare performance by this pre-trained model vs. by StoryGAN.

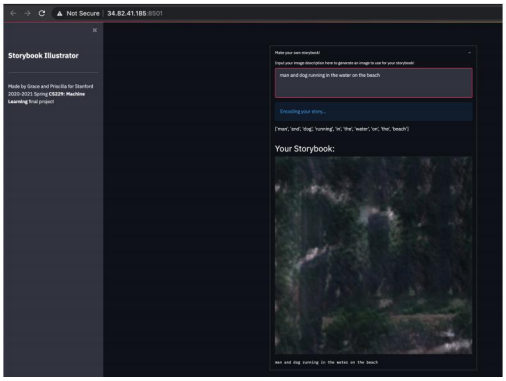

Figure 2: Screenshot of the Streamlit App which takes user text and outputs the generated image.

3 Results

Figure 3: This table shows the results of our various models on a story about a family at the beach. This story happened to have both sis and dii captions (row 1,2) but we passed only the dii captions as input to the GANs. Row 4 shows our StoryGAN trained on sis data for 16 epochs. Row 5 shows the results of StoryGAN after training an additonal 3 epochs on the dii data. Row 6 shows StoryGAN trained after 57 epochs on SIS data. Row 7 shows DF-GAN pre-trained on COCO data.

We compiled the results of our models into one table in Fig. 3. We chose an example story from our dataset about a family at the beach. Row 1 and 2 show the two sets of captions this story happens to have, but to generate the images, we used the dii captions. We were only able to train StoryGAN for 16 epochs on sis data and 3 more epochs with dii data, so the results are very preliminary. If we had more time to train, we hypothesize that the descriptive captions would help

the GAN learn associations better and output better images. We also pasted in the results from the work we built off of. The previous team was able to train until 57 epochs and achieve much nicer results, especially the blue colors in the last two images that are about bodies of water. The first image even features several black blobs which resemble people sitting together. We placed these results to hint at the potential of our GAN if we had time to train longer. The final row shows the results of DF-GAN, pre-trained on COCO, inferencing on the user inputted text description listed at the top of the figure. Interestingly, many of the outputted images had a strong green overtone. We hypothesize that this may be due to the prevalence of nature photos in the images the model was trained on but further work would need to be done to more fully examine this issue.

4 Contributions

Priscilla uploaded VIST and COCO data to the instance, processed the VIST dataset, and trained StoryGAN. Grace got DF-GAN working on COCO data, modified DF-GAN to take single, user-inputted data as well as VIST data, and created the platform for users to input data. Both contributed to the final paper equally.

5 GitHub

https://github.com/priscillalui/CS229-Project

References

[1] Ting-Hao K. Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Aishwarya Agrawal, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. Visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*, 2016.

[2] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David E. Carlson, and Jianfeng Gao. Storygan: A sequential conditional GAN for story visualization. *CoRR*, abs/1812.02784, 2018.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[4] Ming Tao, Hao Tang, Songsong Wu, Nicu Sebe, Fei Wu, and Xiao-Yuan Jing. DF-GAN: deep fusion generative adversarial networks for text-to-image synthesis. *CoRR*, abs/2008.05865, 2020.