Group Members: Dhruv Saligram (dhruvks2)

Project Title

- Old Title: Reproducing Generative Adversarial Text to Image Synthesis
- New Title: Modernizing Generative Adversarial Text to Image Synthesis

Updated Problem Definition & Plan

As outlined in my project proposal, my previous project problem definition revolved around reimplementing the GAN-CLS architecture described in the Generative Adversarial Text to Image Synthesis paper on the Flowers 102 dataset and comparing the quality of my output to the one shown in the paper. However, after consulting with Professor Lazebnik and realizing that the code for this implementation was already publicly available in Lua using Torch, I decided to slightly alter my problem definition.

The foundation of my project remains the same – I will be reimplementing the GAN-CLS architecture and training it specifically on the Flowers 102 dataset. However, having the existing code in Lua / Torch presented an opportunity to modernize the technique and implement it in PyTorch – hence the updated project title. Part of my new work will involve specifically qualifying the usefulness of the architecture being written in PyTorch as opposed to Torch. However, it is very clear from the outset that PyTorch presents a more modern, usable, and adaptable framework that makes this paper's architecture more accessible to the public.

Given that I now have existing code to build from, my project aims to extend beyond solely reimplementation and qualitatively comparing my output to that from the paper. In fully embracing the "modernization" theme of my project, I hope to test newer findings with GANs and new toolkits that could help improve the architecture and the resulting output. This will mainly involve building on top of the PyTorch code, implementing new technologies or techniques, and evaluating / analyzing the output. These mainly involve things that the initial paper lacked a discussion on, including prompt structures to better output (a topic that has become much more prominent recently), different text encoders that massively reduce the overhead required for the project, design changes (mainly derived from https://github.com/soumith/ganhacks – a repository that details "GAN Hacks" that help improve the quality of output), and quantitative evaluation metrics.

Key Resources

My key resources for this project are as follows:

- The initial paper that inspired this project and contains information about the architecture

    - I initially used this paper for inspiration and direction for reimplementation
- The repository associated with this paper that has the source code for the project written in Lua using Torch
    - I am using the code from the main_cls.lua file to inform my PyTorch reimplementation
    - Based on my PyTorch reimplementation, I will be training my own model from scratch
- The Flowers102 image dataset and accompanying captions for training
- I am currently using an existing Sentence-BERT model through Python's sentence-transformers library to encode image captions for training and testing
- I plan to use Google Colab to run my code, and am considering using Colab Pro to speed up runtime and increase GPU access for training

Summary Of Work Done To Date

Almost the entirety of my time spent working on the project so far has been spent reimplementing the architecture in PyTorch based on the Torch implementation in Lua. While having the code has certainly sped up and aided the process, the architecture still remains advanced and required a lot of time to understand. Additionally, Torch has functions that were used in the original code (such as ConcatTable, CAddTable, and Replicate) that I had to research and learn how to reimplement in Python using PyTorch.

As of now, I have managed to successfully complete this reimplementation and integrate an off-the-shelf sentence encoder into the project. This has led to me being able to successfully run a training loop for the model and produce output from the trained model conditioned on text. The issue, however, is that my code is currently far too slow. Given how long it took me to reimplement the architecture, I only had a day to try and train the model. However, my code currently runs at around 30 seconds for a single batch with a batch size of 64 (the value which was used in the original implementation). Given that there are 8,189 images in the dataset (leading to around 128 batches) and the original implementation trained the model for 600 epochs, the current runtime would mean that it is completely infeasible to properly train the model. Having run the model only for 2 epochs, I was able to generate the following output:

While this is clearly just noise, it did help increase confidence that the code for training and generating images is running correctly and could produce successful results if the model were trained properly.

Because of this, my next immediate step in the project will be to find ways to massively speed up the runtime beyond just upgrading to a faster GPU. If I cannot achieve a significant enough runtime boost, however, my current fall-back plan is to train on a far more limited set of the available data.

Plan For The Remainder Of The Project

With the changes that I have made to my project in mind, my new project milestones are:

1. Processing image and text data from Flowers102
2. Finding and integrating a suitable text encoder
3. Implementing the GAN-CLS architecture
4. Finding ways to speed up training time
5. Training on the Flowers102 data
6. Testing / generating output of flowers from text captions
7. Qualitatively comparing resulting output to the output from the original paper
8. Creating a way to quantitatively evaluate output
    a. Researching and implementing evaluation choices which could include choices from Inception Score, Inception Distance, CLIP scores, or evaluating how well generated output can be classified by a classifier trained on the Flowers102 dataset with a high test accuracy
9. Quantitatively scoring my output
10. Evaluate different text encoders and see if they have an effect on the quality of output
    a. This could include fine-tuning off-the-shelf text encoders specifically on the Flowers102 dataset and evaluating the effect
11. Implement certain design changes from the “GAN Hacks” repository
12. Train a new model using the optimally chosen text encoder & design choices
13. Quantitatively score new output and evaluate the effect that the changes made
14. Test the effect that prompt structure can have on quality of output through similar quantitative and qualitative tests

As mentioned in the previous section, milestones 1-3 are completed and milestone 4 is currently in progress.

My minimum goal for this project is to complete through milestone 7. This scenario would likely only be achieved if I cannot find any way to significantly speed up the training time for the model. However, if this is the case, I will likely only use a much smaller subset of the data for training and then still continue to test to see how the proceeding milestones could increase the quality of the output and evaluation (recognizing that the final quality would not measure up to the initial paper due to the limited training).

My maximum goals are completing all of the milestones listed above. I believe that they encompass an extremely comprehensive amount of evaluation across multiple levels of the architecture. The main barrier to achieving these goals remains the model's training time. If it is infeasible to train a second, more optimal model, then milestone 11 may be left uncompleted.

The target and most realistic goal currently is to complete all milestones except 11 and 12. Instead, I aim to evaluate the text encoders by themselves, pick an optimal one, and then train the model once using it. This would avoid the need to train two different models but would still allow for a comprehensive evaluation at different levels.

As stated previously, the largest mitigation strategy I have is to train on a subset of the available data. In the scenario where I am only able to train the model once without any real changes from the original implementation or quantitative evaluation metrics, then I would likely take the project in a slightly different direction and instead focus the discussion on the direct benefit of having the architecture reimplemented in Python with PyTorch as opposed to Lua with Torch.

Member Roles: All work is being done by myself due to working alone.

Reference List

- Reed Scott's paper & implementation repository
- Flowers102 dataset
- "GAN Hacks" repository