Arun Agarwal, Eric Nguyen, George Aeillo

Professor Abha Belorkar

CIS 4496 - Projects in Data Science

March 14th, 2023

Project Charter

1. **Problem Description**:

Have you ever thought to yourself, after taking a picture of beautiful scenery, what would it look like if a famous painter was there and made a painting of it? We aim to solve this data science issue of style transfer, in which a user-generated image will be transferred to a "Monet-style" painting. The unique style of Claude Monet, such as his color choices and brush strokes, can now be imitated thanks to algorithms called Generative Adversarial Networks (GANs) (3). As shown in the diagram below, the first component of the network acts as a "detective" that is trained to distinguish between real examples and fake examples that are generated by the second "generator" component. In training, the "generator" component learns to train fake examples that are identical to the real examples, so the "detective" component is only 50% accurate (2). Our task is to build a GAN that generates 7,000 to 10,000 realistic Monet-style images of dimensions 256 X 256 X 3 (RGB) (3).

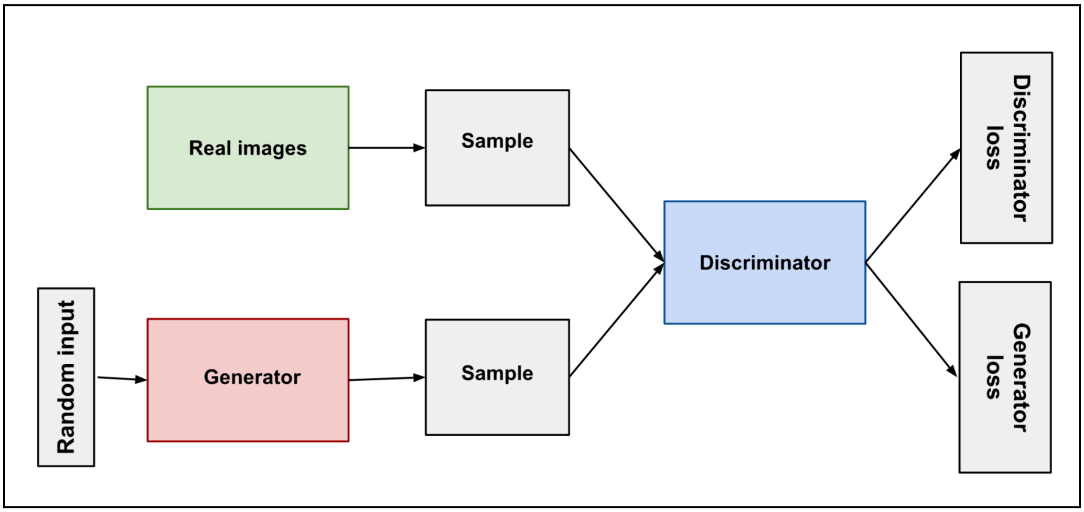

Diagram of GAN Process (2)

2. **Project Scope**:

The end user will be able to upload an image to a self-created website (with a UI) and download a stylized version of the image. For this project, we are restricting the user's input image size to 256x256x3, and we are focusing on emulating French painter Claude Monet's art

style when generating stylized images (3). However, our objective is to also allow for the style of multiple artists, such as Van Gogh, Cezzane, and Ukioyo-e. Additionally, we are aiming for the end user to receive their images relatively quickly, on the order of seconds, through the use of a web application with a user interface. Due to the bidirectional nature of CycleGANs, we hope to integrate artist-to-artist style transfers as a stretch goal (8). First, users will upload one of our four artists' images, and this image will be transferred into the style of a photograph. Then, this photograph can be transferred into the style of a different artist.

3. Metrics:

The success of the project will be quantified using MiFID (Memorization informed Fréchet Inception Distance), a modification of FID (Fréchet Inception Distance) created by Kaggle. FID is a common metric used to assess the quality of images created by a generative model, such as a GAN. Unlike the Inception Score (IS)–another common metric for GAN evaluation described later–the FID compares the distribution of generated images with the distribution of a set of real/ground truth images. Specifically, FID computes the Fréchet distance between two Gaussian distributions fitted to feature representations of the Inception network (7). Here, one uses the Inception network to extract features from an intermediate layer. From there, one models the data distribution for these features using a multivariate Gaussian distribution with mean μ and covariance Σ. As provided by Kaggle, the FID between the real images r and the generated images g is computed as

$$\mathrm{FID} = ||\mu_r - \mu_g||^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

(3)

where *Tr* is the sum of the diagonal elements.

On top of FID, this Kaggle competition takes into account training sample memorization in the performance metric. First, the memorization distance is calculated as the minimum cosine distance of the training samples in the feature space, averaged across all user-generated image samples (3). This distance is assigned a value of 1 if the distance exceeds a pre-defined epsilon. MiFID is then the FID metric multiplied by the inverse of the memorization distance (with the implemented threshold).

$$MiFID = FID \cdot \frac{1}{d_{thr}}$$

(3)

Kaggle calculates public MiFID scores with the pre-trained neural network Inception, and the public images used for evaluation are the rest of the TFDS Monet Paintings. The competition calculates the MiFID score after we submit our code/solution on Kaggle, so we cannot recreate

the calculations for our personal use (mostly due to the memorization distance). That is, this competition keeps our performance hidden from us until after submission, which becomes problematic as the code takes 2+ hours to run. We would also need a method to measure our performance to complete the proposed steps that extend beyond the scope of the competition. Therefore, we will look into how to calculate FID ourselves, and write corresponding functions in our scripts to do so. FID is also a common scoring metric for GANs, so we can use this formulation to compare our models and scores with previous/related work. This then will help us understand how to boost our performance.

While FID is the most common metric used by others in the domain, Inception Score is also popular. This score takes a list of images and returns a single floating-point number, which is a score of how realistic the GAN's output is (6). The score measures the variety of the images as well as their distinct quality (each image looks like an actual distinct entity). The Inception Score is high when both of these quality scores are high. Unfortunately, IS does not capture how synthetic images compare to real images; that is, IS will only evaluate the distribution of the generated images. Therefore, FID was developed to evaluate synthetic images based on a comparison of the synthetic images to the real images from the target domain (6). While it is true that FID commonly produces high bias, this is no less true for Inception Score. This and the fact that Inception Score is limited by what the Inception (or other networks) classifier can detect, FID is more popularly used to score GANs today and is what we will use for our project.

4. Architecture:

Our data will primarily consist of 256x256 images of two groups: artist paintings and camera photos. Specifically, our initial dataset as provided by Kaggle includes 300 Monet paintings and 7028 photos in both JPEG and TFRecord format (TensorFlow's custom data format) (3). We will primarily work with the TFRecord format; however, we will have the JPEG files as a fallback should any difficulties arise when using the TFRecord format. Our group is currently working on gathering additional paintings for not only Monet but other artists as well.

The total data including both formats amounts to 385.87 MB—small enough to be stored on a personal computer. However, given that the problem requires intense computation within a limited timeframe, we will need to make the data accessible by TPU-equipped machines for the best results. Kaggle conveniently provides access to the initial dataset stored in their systems through their Kaggle Notebooks (Jupyter Notebook service provided by Kaggle) which provide free limited access to TPU computers. Indeed, we will use Kaggle Notebooks to operate on the data and perform the majority of the computation required for the project. So far in the project, we have not run into rate limitation issues on Kaggle, but we have run into a variety of problems involving the use of the Temple HPC node, such as the other users hogging the GPUs on the node, along with issues finding access to any useable GPU on the Node. Due to the relative reliance on Kaggle compared with the Temple HPC node, we anticipate working on Kaggle for the remainder of the project.

For the entirety of our project, we have used CycleGAN for this problem. While our team initially considered utilizing state-of-the-artist models, such as UVCGAN, we ultimately decided against it. For example, the large training times, the amount of data used, and the large quantity of time required to understand other parts of deep learning, such as vision transformers, would have required our group to conduct in-depth-research and retool our project from scratch, rather than a simple swap of models (5). Due to this, our group in the second phase has focused mainly on data acquisition and data augmentation and developing models for different artists.

Once the models are trained on the image data, they will be ready to be deployed as a service for end users to access via an online web interface. Our web application will be powered by TensorFlow.js, a library that enables training and deployment of TensorFlow models on the web, which will load pre-trained models and make predictions on user-inputted images—this approach is completely client-side and does not require any cloud infrastructure; however should we find that TensorFlow.js is not sufficient for our project requirements, we may employ cloud services such as GCP or AWS to store our models and make predictions through the cloud. For the web interface itself, we will use Next.js–a web application framework–along with NextUI–a React component library–since these tools are what we are familiar with for building websites. For the scope of this project, the only functionality we are providing to the end user is a single stylized version of their uploaded input image as generated by our model.

Given time, we may consider adding features including batch processing of multiple input images, generating multiple stylized versions of an input image, etc. Additionally, we are considering using TensorBoard, a dashboard that allows us to monitor and visualize metrics of our model such as losses and outputs at different epochs, in addition to our interactive website.

5. Plan:

We have been taking a vertical splicing approach to this project, where each feature will be developed and delivered in phases:

1. Phase 1 (Feb 5 - Feb 22): Baseline model demo
2. Phase 2 (Feb 23 - Mar 29): Improved model demo
3. Phase 3 (Apr 3 - May 1): Finalized model demo with interactive web interface

In Phase 1, we constructed the baseline model using CycleGAN that will allow us to transfer images to a Monet-style painting. This allowed us to develop a baseline working model in order to base our progress. In this phase, we developed a Project Charter, researched and presented on a related topic related to GANs, and performed our initial project demo.

In Phase 2, we are currently improving and expanding our project in two main ways. First, we experiment with our base model by testing the amount of Monet paintings to incorporate into it. We also perform various image augmentations. After this, we expand the number of artists that a user would potentially be able to translate an image to, such as

“Van-Gogh-ifying” an image. Other tasks for this phase include writing our first bimonthly progress report, revising our project charter, and demonstrating our improved changes to our model with the class.

In the final phase, we will make any finishing touches to our model, as needed. Once our model is completely finalized, we will train and save the weights of the model’s generators in TensorFlow’s “SavedModel” format and convert the saved models to a format that TensorFlow.js can read. Once we have the models saved in a TensorFlow.js-compatible format, we can use TensorFlow.js to load the models onto the website, from which we can accept user-uploaded 256x256 JPG images and produce a style translation on the user images. The user input and generated output will be displayed side-by-side through our web interface, where the user will be given the option to download their generated image. In this phase, the deliverables that we will produce include the second bimonthly progress report, the data report, the model and performance report, and a final demonstration of our project including our interactive dashboard.

6. Personnel:

- George - Responsible for business understanding of the project, researching and gathering code for GANS, CycleGANs, other style-transfer models, data-augmentation techniques, and datasets to be used by the team, code documentation, code maintenance, and implementation of FID metrics in Kaggle.
- Eric - Responsible for model implementation, website implementation, GitHub repository ownership, Conda environment/Singularity container configuration, code quality, technical assistance, exploratory data analysis, and deployment.
- Arun - Responsible for understanding of the model scoring, writing and designing project-related presentations, working on majority of written assignments (bimonthly progress reports, project charter, etc.), and code documentation.
- Note: These are broad generalizations of our work to the project because all of us help each other whenever it is needed in other categories. This is especially true as our project progresses, as we are moving away from the research phase and presentation phase of the project, and the majority of all of our time is being used on writing and implementing code.

7. Communication:

The communication around the project will be mostly handled through online services such as Slack (messaging app), Trello (project management tool), and GitHub (code-hosting platform). In Slack, we have channels for messaging one another individually and general channels to communicate as a group. We can post research papers, relevant repositories, and resource links in the “References” channel and hold project discussions in the general chat. Furthermore, Slack can connect with Trello, which will allow us to receive notifications about upcoming tasks and deadlines on our Trello board. On the Trello board, project tasks are

separated into three phases (based on the demo schedule), which currently hold multiple sections: “To Do”, “Doing”, “Done”, “Long-Term”, “Decision List”, and “Scratched Decisions”. The “Decision List” describes decisions relating to our project that have yet to be formally decided, while “Scratched Decisions” are a place for our team to formally indicate that we will not be proceeding with a previously planned or considered step. We also communicate in person inside and outside the classroom weekly (1-2 times per week). Finally, GitHub will be used as the repository for our code.

8. Related Works:

We discuss related works for this project to emphasize the concept’s popularity, but also to demonstrate works we can reference to build our model. One almost identical work to this competition was performed by a research team at UC Berkeley, in which they used an ‘Unpaired Image-to-Image Translation’ technique to turn photographs into Monet-style paintings, as shown below:

(1)

While this is the same task that we are asked to perform in this competition, UC Berkeley's algorithm is also capable of a “reverse-Prima” trick of taking the painting and transforming it into a photograph (1). As this is a potential future step for our project, this work will serve as a great source to reference. Furthermore, their algorithm is capable of transposing style elements–to make horses into zebras, for example–and applying depth of field (1).

References:

1. Cade, D. L. (2017, April 3). *This AI Can Convert Paintings Into Photos and Summer Into Winter*. PetaPixel. Retrieved March 14, 2023, from https://petapixel.com/2017/04/03/ai-can-convert-paintings-photos-summer-winter/
2. Google Developers. (2022, July 18). *Overview of GAN Structure*. Google. Retrieved March 14, 2023, from https://developers.google.com/machine-learning/gan/gan_structure
3. Kaggle. (2020, August 28). *I'm Something of a Painter Myself*. Kaggle. Retrieved March 14, 2023, from https://www.kaggle.com/competitions/gan-getting-started
4. Kan, W. (2020, September 26). *Demo MiFID metric for Dog image generation comp*. Kaggle. Retrieved March 14, 2023, from https://www.kaggle.com/code/wendykan/demo-mifid-metric-for-dog-image-generation-comp/notebook
5. LS4GAN Group. (2022, August 9). *LS4GAN/Benchmarking*. GitHub. Retrieved March 14, 2023, from https://github.com/LS4GAN/benchmarking
6. Mack, D. (2019, March 7). *A simple explanation of the Inception Score*. Medium. Retrieved March 14, 2023, from https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a
7. Wikipedia. (2023, February 28). *Fréchet inception distance*. Wikipedia. Retrieved March 14, 2023, from https://en.wikipedia.org/wiki/Fr%c3%a9chet_inception_distance
8. Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2020, August 24). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. arXiv.org. Retrieved March 14, 2023, from https://arxiv.org/abs/1703.10593