Exploring Latent Space through a Pipeline of Face Synthesis and Detection

Alyssa Riter
Advisor: Patrick Flynn
Date: 12/02/2024

Introduction - Problem Statement

The goal of this project has been to put the synthesis of faces into a pipeline with a face detector to understand how far into the vector space we must walk to get images that appear to be "monsters" and are no longer recognizable as faces. Additionally, we delve into understanding the yield of confidently recognizable faces from a random sample of the latent space. The latent space is a mathematical region in which latent vectors exist, and it is a low-dimensional, abstract space where images are represented in a simplified form. This space captures underlying patterns and structure of the data, even if the data itself is high dimensional, exploring this space can give us insight into the confidence rate that facial detectors exude when analyzing the generated images and the percent yield of recognizable faces in the space. The latent space explored in this project is 512-dimensional which is also the dimensionality of each vector generated.

Background - Generative Models & The Latent Space

For the face synthesis portion of this project, StyleGAN2 with adaptive discriminator augmentation (ADA) was utilized. The main purpose of StyleGAN2 is to reduce the appearance of artifacts (e.g., water droplets) generated in the original StyleGAN image. In general, the purpose of StyleGAN is to synthesize realistic photo images. It is a very robust general adversarial network (GAN) architecture as it generates extremely realistic images at a high resolution. GANs consist of two neural networks, a generator and a discriminator that are trained together. The generator creates images from random noise while the discriminator tries to distinguish between real images and the fake ones produced by the generator. The primary component is the use of adaptive instance normalization (AdaIN) which is a normalization method that aligns the mean and variance of the content features with those of the style features. In the case of my programs, it provides a mapping network from a latent vector into W. We see progressive changes from low-resolution images to high-resolution images and as we walk along the latent space, some features such as the eyes are fixed into place which can lead to undesired artifacts in the generated images. It was suggested that the droplet artifacts found in the original StyleGAN design are a result of the generator intentionally creating a strong spike to scale the signal as it likes elsewhere. These undesired results led to the development of StyleGAN2.

The developers of StyleGAN2 remove the AdaIN operator and replace it with weight modulation and demodulation in each step. Compared to instance normalization, the demodulation technique is less powerful as

it relies on statistical assumptions about the signal rather than the actual contents of the feature maps leading to fewer artifacts found in the generated images. Along with this development, there was the implementation of StyleGAN2-ADA which is a method of training GAN with a limited set of data and reduces the amount of discriminator overfitting that can be found in GANs, especially when the amount of available data is low. Data augmentation involves randomly applying transformations that maintain the integrity of the data to the input data to generate various realistic versions of it, thus increasing the amount of training data available and reducing the likelihood of overfitting.

After using StyleGAN2-ADA for the face synthesis, InsightFace's SCRFD was used for facial recognition and embeddings. InsightFace generates face embeddings using deep convolutional neural networks, after performing feature extraction the network can take them and convert them into a compact, numerical vector - the face embedding. The vector embedding represents the face's identity in a multi-dimensional space. The key principles behind SCRFD are feature extraction, sample and computation redistribution, bounding box regression, and non-maximum suppression. Feature extraction involves the network analyzing the input image, and extracting features like edges, textures, and shapes in various scales. Sample and computation redistribution strategically allocates more computational resources to areas of the image more likely to contain faces, overall improving efficiency. Bounding box regression allows the network to predict the location and size of potential faces by drawing a bounding box around the face. Lastly, non-maximum suppression ensures that the most accurate bounding box for each face is kept, removing overlapping or redundant boxes.

Methodology

To conduct the experiments and obtain the percent yield and the confidence score for each set of generations we ran the face synthesis algorithm (StyleGAN2-ada) on a random walk of 1000 images. We simulated this random walk 4 times using a different random seed each time to get a variety of image sets. Starting at the origin, we traversed the latent space using a step size of 0.01. After generating these images we ran them through InsightFace to get the confidence score, bounding region, and embeddings for each image. The extraction of vector embeddings will allow us to visualize the vector space and detect any clusters forming in the region. To better visualize the results we graphed the confidence scores for each batch of 1000 images, with undetectable faces yielding a confidence score of 0.00.

Figure 1: Face Generated at the Origin

Figure 2: Face That Yielded No Detection & No Confidence Score

Experimental Process

Defining Experimental Parameters: Before running the scripts to generate the faces we set some key parameters such as the starting point (defined as w_avg) and boundaries (max_norm) in the latent space to define the scope of the walk and keep the results controlled and more realistic. We utilize a step size of 0.01 to explore the space, this step size controls the magnitude of change for each step in the latent space. A value of 0.01 was chosen because it is small enough to allow for a smooth transition between images while also giving us diversity in the produced images. The number of steps chosen for this experiment determines the length of the walk, and how many latent vectors, which correspond to images, will be generated. In this experiment, we have a script that runs in batches to generate images in groups for memory usage optimization. The generated images are then stored in an output directory named after the date, hour, and minute the script ran.

Loading StyleGAN2-ADA Network: The pre-trained StyleGAN2 model is loaded from a pickle file and includes a mapping network as well as a synthesis network. The mapping network projects random input vectors (Z) into the latent space (W), while the synthesis network converts the latent vector (W) into high-resolution images. As mentioned during the discussion of StyleGAN2-ADA above, the model uses a "W" latent space rather than the "Z" latent space for smoother generation and exploration; the W-space is derived from Z-space through the mapping network. This choice avoids undesirable artifacts in generated images. This is important to the experiment because it ensures that the walk stays within a region of the latent space where the generator produces meaningful, artifact-free outputs. Additionally, each step in the random walk results in smooth changes in the generated images allowing for clear transitions for analysis. Working in the Z-space could lead to unnatural transitions and low-quality images as intermediate steps in the walk.

Computing the Average Latent Vector: The average latent vector (w_avg) is computed using random samples from the latent space, this is done dynamically during execution. In the script, we generate 10,000 random vectors and map them to obtain the W-space using StyleGAN2's network. Then, we compute the mean of these w vectors to get w_avg. This is necessary because this average vector will represent the center of the latent space we are exploring and serves as the foundation and baseline for the space. Furthermore, the norm of the average vector will help us ensure the random walk stays within a meaningful and visually coherent region of the latent space.

Random Walk in Latent Space: We will start at the average vector to initialize the walk,

and then we will iterate for each step by applying a small random perturbation to the current vector we are using. To ensure we stay within a defined hypersphere of the latent space we will continuously check if the updated vector's norm exceeds our calculated maximum norm (max_norm), if so, we will scale it back to stay within bounds and then store the updated vector. By capping the norm, the walk cannot go into regions of the latent space that are undesirable or produce artifacts and we maintain a relatively small region of space to explore, giving us greater insights if there are unrecognizable faces found. The small step size ensures that consecutive vectors produce images with minimal changes since they are close together in the vector space. Once again, this small step size gives us greater insight into how far away we need to go in the vector space before we get a face that looks like a "monster". After completing the random walk, the latent vectors (w_out) are saved to .npz files to allow for reproducibility and portability for future experiments.

Image Synthesis: Each latent vector is fed into the StyleGAN2 synthesis network to generate the images. This is done by converting the latent vector into a tensor and repeating it across the layers of the network. Once again, we render images in batches to increase memory efficiency. Each generated image is saved as a .png file and named after which step it is in the walk.

Face Detection: Using InsightFace we create a script to perform facial recognition on all the .png image files we obtained from the previous step. The script is designed to analyze image files and produce a CSV file containing information about detected faces. The script accepts two command-line arguments, one is a glob pattern which is a file regular expression to specify the input files that are stored in the output directory. The other script input is the output file which is where we will save the results in CSV format. In this script, we configure the GPUs and workers to specify which GPU we want to use for computation. The workers indicate the number of workers per GPU to perform face detection on the images and exploit parallelism in the process. The CSV file contains the header "Filename, score, embeddings" to initialize the fields we are writing into the file. To allow for multiprocessing we initialize a queue with (GPU and worker) pairs for assigning tasks to workers, the ThreadPoolExecutor module manages workers, each processing one file at a time. We also initialize the InsightFace FaceAnalysis framework for each worker, correlating it to a specific GPU and worker ID.

Worker Threads: Each worker will load the image file and process a single frame at a time. Then the worker will run face detection through InsightFace using the command local.app.get(local.img). The results of each frame include the bounding box of the detected face, the key points (e.g. eyes, nose, mouth), the detection confidence score, and the face embedding. The results are stored as a dictionary for each frame, including the filename, frame number, and detection results which are then returned after the file is processed. Having the embeddings associated with a confidence score will allow for future analysis in

performing clustering or similarity searches based on the facial embeddings and grants us the opportunity to map where regions of "monsters" (unrecognizable faces) occur.

Graphing: To visualize the results of the facial recognition we can plot a histogram for each of the 1000 image cases run. We ran the script with different random seeds so that we could get a variation in the sets of data we collected to guide our analysis. We graphed the confidence score of each case using a histogram revealing how confident InsightFace's network was in asserting that it was looking at a face as well as how many faces were yielded from the sample.

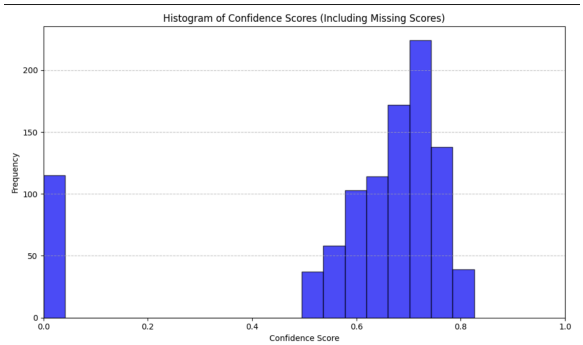

Figure 3: Histogram of Confidence Score for One Batch of 1000 Images

Results & Discussion

Overall, out of several runs of 1000 sample images, between 895 and 918 detectable faces were produced per 1000 images. The high rate of face detectability reflects the structured nature of StyleGAN's latent space, which has been trained to generate coherent faces. However, roughly 10% of undetectable images indicate a wandering pattern into regions of the latent space that correspond to artifacts, noise, and other features that distort the facial generation. These regions give rise to a question of how far into the vector space we need to go to find these uninterpretable faces and how frequently they occur in the vector space. The occurrence of undetectable images suggests that through our random walk, we went into the boundaries of the "face manifold" which represents a transition from a region of discernible faces to indiscernible. Since we apply constraints to the region of exploration using a norm boundary to keep the walk somewhat near the origin, the fact that we have a face detectability of approximately 90% reveals that we are begging to breach into the region where faces become monstrous. The confidence scores associated with the detections range from 0.00 - 0.82, with the majority of the images lying around the 0.70 mark. This indicates that the model has created images with questionable facial features and some un-human-like characteristics. The existence of generated images with a confidence score of 0.0 indicates that there are limits to generalization in both StyleGAN and InsightFace and that there exist undetectable and interesting images within the expanse of the vector space.

Figure 4: "Monster Face" With A Confidence Score of 0.54

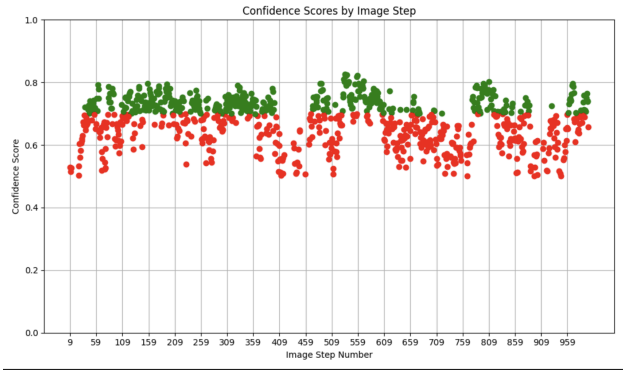

Figure 5: Confidence Score Plot, Green is a Confidence Score greater than 0.70

When the program was initially run it had a large step size of 0.1, which contributed to the frequency of undetectable images. Larger steps in the latent space led to greater derivations from face regions. Switching the step size to 0.01 allowed for more recognizable images to be generated and kept the walk within a more recognizable subspace, however, there were still distorted faces found, highlighting the strange and interesting nature of the latent space. Overall, the results found in the experiment reveal that the latent space is richly structured but not uniformly optimized for generating valid faces. This exploration into why some images are undetectable and what else lies within the region can give us greater insight into the boundaries of face generation and latent space traversal.

Future Work

In the future, we would like to capture even more data so that we can assert some places in the hypervolume where undetectable faces exist and explore them further. Beyond that, it would be beneficial to train the same network with a different starting configuration to do a similar analysis and see if there are any dependencies to the results we have already acquired.