Generating Faces Using Generative Adversarial Networks

<u>Proposed Project:</u>

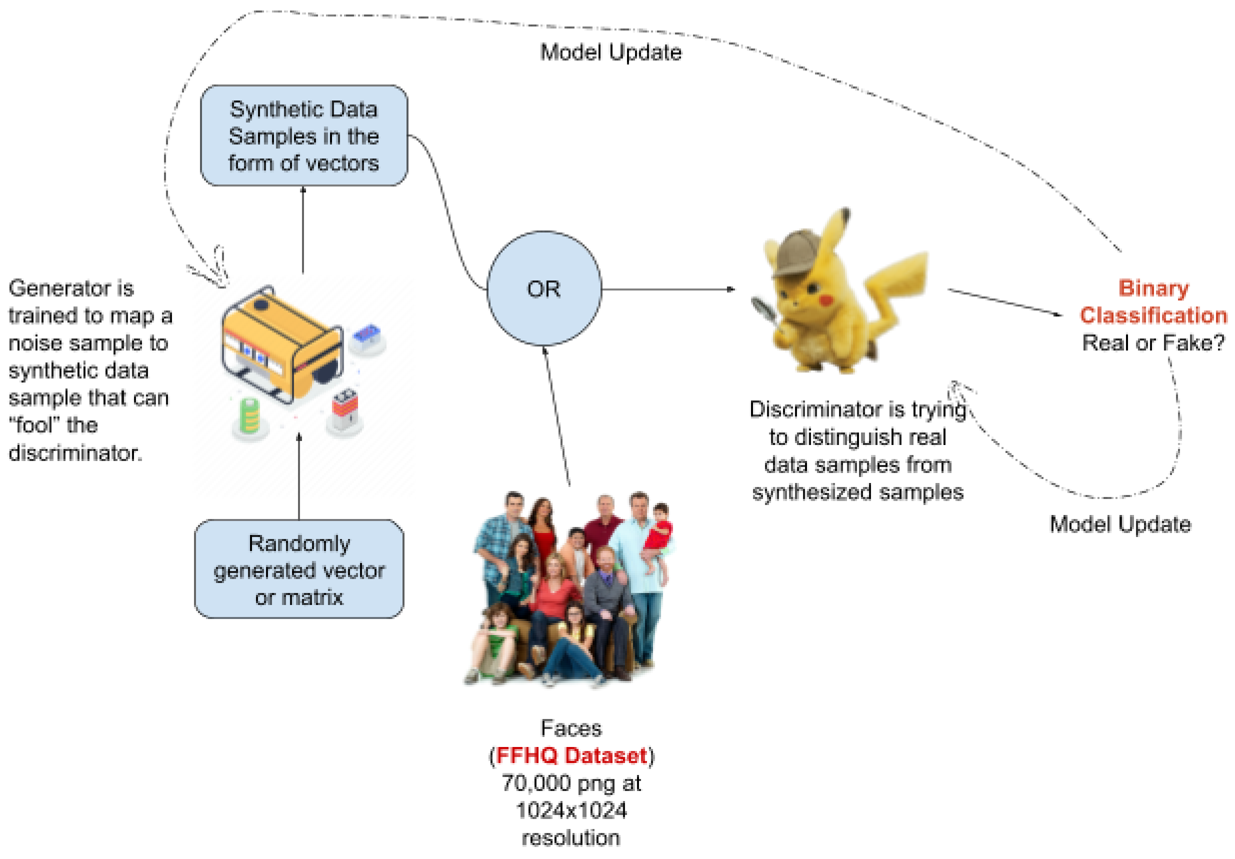
- Initially we believed our dataset can be created using one of the same techniques as we proposed in the beginning. However, upon reading your comments and discussing amongst ourselves we realized that choosing a static dataset made more sense. Your comments also helped us reiterate our end goal.

<u>Report Summary</u> <u>(Progress Report):</u>

- In this report, we would like to provide a brief overview on our progress of implementing the above mentioned network for the purpose of generating human faces. In addition to this, we will also be presenting the remaining challenges in theory and application.

- **Motivation for the Project**: Deep neural networks are also used in supervised learning, such as classification and regression. However, GANs, or Generative Adversarial Networks, use neural networks for a completely different purpose: Generative modeling. A good way to learn deep representations is by using generative adversarial networks, importantly when we want the training data not to be extensively annotated. Using backpropagation signals involving a pair of networks through a competitive process is a way of applying GANs. A multitude of applications exist on this particular representation for eg., image synthesis, semantic image editing, style transfer, image super-resolution and classification.

- **<u>Technical Description</u>**: In the figure below, the two models which are learned during the training process for a GAN are the discriminator (D) and the generator (G). These are typically implemented with neural networks, but they could be implemented by any form of differentiable system that maps data from one space to another.


Model Update
Synthetic Data Samples in the form of vectors
Generator is trained to map a noise sample to synthetic data sample that can "fool" the discriminator.
OR
Binary Classification
Real or Fake?
Discriminator is trying to distinguish real data samples from synthesized samples
Model Update
Randomly generated vector or matrix
Faces
(FFHQ Dataset)
70,000 png at 1024x1024 resolution

- **Dataset**: We checked on the following Datasets initially :
  - CelebA Dataset
  - Faces94 Dataset
  - Flickr-Faces-HQ Dataset (FFHQ) Dataset

  After checking the datasets and the implementations , we have decided to work on the Flickr-Faces-HQ Dataset (FFHQ) Dataset.

  Flickr-Faces-HQ Dataset (FFHQ) is a dataset which contains 70,000 high-quality PNG images relating to human faces at 1024×1024 resolution .This dataset includes more variation than CELEBA-HQ dataset in terms of age, ethnicity and image background, and also has much better coverage of accessories such as eyeglasses, sunglasses, hats, etc. The images were crawled from Flickr and then automatically aligned and cropped.

We will be needing UITS Carbonate Deep Learning Cluster as we need GPUs which improve our training Process.

- **Data Preprocessing**:
  - One huge benefit of using Flickr-Faces-HQ Dataset is that each of the images has been cropped to remove parts of the image that don't include a face using dlib. We then resize this down to 3x64x64 NumPy images. Resizing the data to a smaller size will make for faster training, while still creating convincing images of faces. The 3 is for RGB channels.
  - Created a dataset and gave a directory of images to process. We used PyTorch's ImageFolder wrapper, with a root directory - /input/flickrfaceshq-dataset-ffhq/ and data transformation steps are performed on this.
  - As we are resizing the Images are resized to 3 x 64 x 64. And the pixel value range of these images are made between (-1, 1), which is in the range of tanh activation.

- **DataLoader:**
  - We change the image size to 64 and the reasonable batch_size parameter is 128. After loading our normalised data we are all set to visualize some images!

In the Upcoming Milestones......

- In the coming days, two of us will be working on the “Generator” part of the code and the rest on the “Discriminator” part. We aim to produce convincing results where the discriminator will be giving out 0.5 as the prediction. (The achievement would be to produce as realistic images as possible).

- As such, there is no objective function or objective measure for the generator model. We will be using Manual inspection for validation as this might help us take steps in the right direction.
- However, we are also looking forward to reaching at least a substantial level of accuracy and comparing these levels using Inception score and Frechet Inception Distance to quantitatively summarize the quality of generated images.
- We are also looking forward to utilizing Carbonate's computing power.

<u>Caveats that we might face in the future:</u>

- Hyperparameter Tuning can be a major hiccup with various params such as a Batch Size , Number of epochs, Learning Rate, Activation functions in different layers, Optimization Algorithms, Loss function, Experimenting with the number of layers in both the networks.
- Mode collapse.
- Vanishing Gradients.
- Training two networks and converging their gradient updations is highly unstable.