GANs: Very hard to train!

By Zach, Will, and Jonathan

What is a GAN

- A Generative Adversarial Network (GAN) is a type of machine learning model
  - A new, exciting example of unsupervised learning
  - A novel application of zero-sum game theory
  - Composed of two neural networks that aim to outcompete one another
    - There is the generator and the discriminator
  - Both employ deep learning statistical methods to minimize expected prediction error

What is a Generator?

- A convolutional neural network
    - Specific for the processing and analysis of data with a grid-like structure (e.g. images)
- Goal is to create fake images that are so similar to the real images that they cannot be accurately distinguished by the discriminator
- The generator takes a random noise vector as input and then employs transposed convolutions to increase the spatial resolution of the image to match that of the way the training data is distributed

What is a Discriminator?

- A deconvolutional neural network
  - Specific for object recognition and the segmentation of images
- Goal is to accurately distinguish between the fake images that have been created by the generator and the real images
- The discriminator takes an image as the input and then functions as a binary classifier that assigns a probability to the image
  - The interpretation of the probability is context-specific and depends on both the quality of the generator and the accuracy of the discriminator

Loss function of GANS

- Two different loss functions
  - One for the generator and one for the discriminator
- Generator LF:
  - $L(G) = \log(1 - D(G(z))$
- Discriminator LF:
  - $L(D) = -\log(D(x)) - \log(1 - D(G(z)))$
- D = discriminator network, G = generator network, z = a noise vector, x = a real image

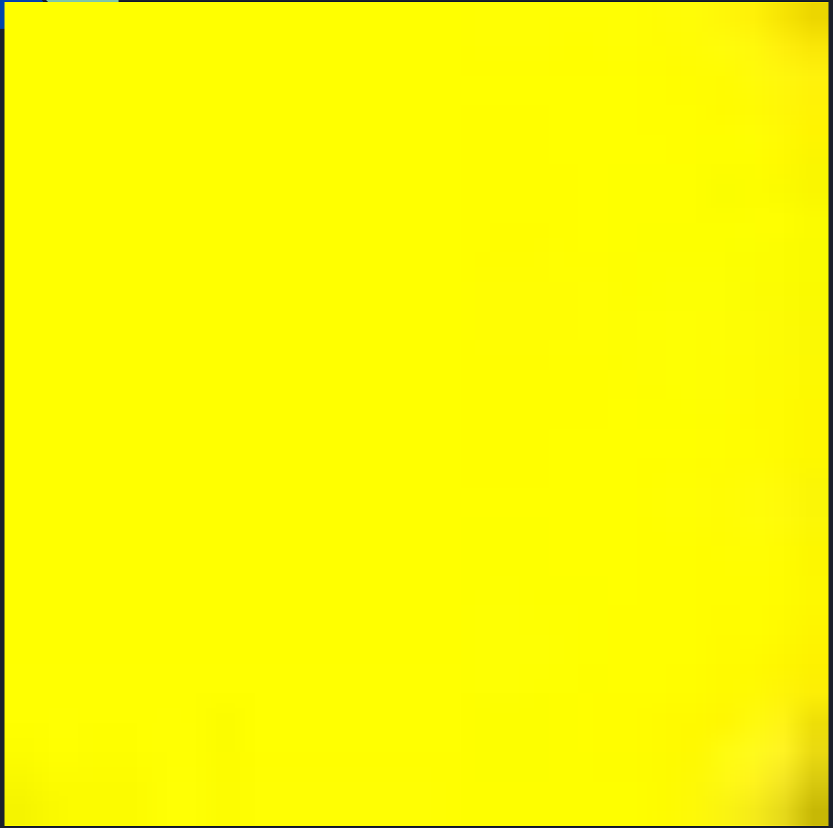
Our code and Process:
Attempt 1

- Followed https://github.com/nicknochnack/GANBasics
- Changed machine learning architecture to suite 3 color channels
- Scaled images to 28 by 28 and processed images to have rgb values between 0 and 1.
- Ran around 300 epochs with 1000 images
- First “flowers” did not appear until 100 generations or so

Attempt 2

- Mirrored the coding and neural network structure done in Attempt 1
- Images were scaled to 64 x 64 instead of 28 x 28
- Much more costly computation / epoch process
- First few flowers (structure and color) appeared around epoch 50

Mode Collapse



- Had many generations where GAN produced same output
- Generator effectively can outcompete discriminator by using only certain images instead of goal of many different images
- Mode collapse can be remedied by training discrimantor more, but other methods are available

Created images(28 x 28 x 3(RGB))

Problems/Constraints

- GANs are computationally heavy and we ran out of google colab credits
- As image gets larger takes longer to compute
- Google colab randomly shutting down kernel
- Flower data set only worked due to flowers having same orientation
- Datasets with images rotated were not as successful
- Mode collapse causing generator to create same images for many epochs

Future directions/changes

- Buy an NVIDIA graphics card to increase number of epochs and train larger dataset
  - Increase computing power overall
- Save model every certain number of epoch
- Find solution to stop mode collapsing for many epochs (possibly different loss function i.e. Wasserstein)

More Information / Reproducibility!

For more information and an in-depth look at code as well as other background information / example projects, please refer to the GitHub repository at the following link:

https://github.com/jlinschool/FirstGANProject