Comparison of GAN Architectures

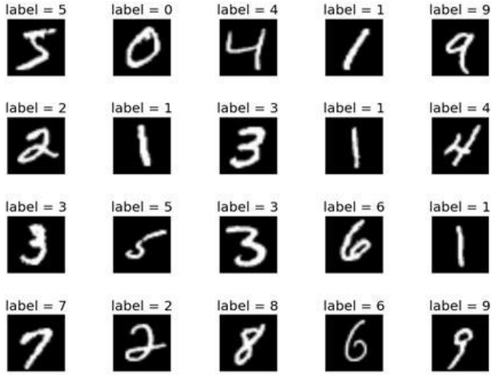
Ryan Missel
Jeffery Russell
Kyle Rivenburgh

Agenda

- Problem Statement
- What is a GAN?
- GAN Architectures
- Experiments/Results
- Conclusion

Research Questions

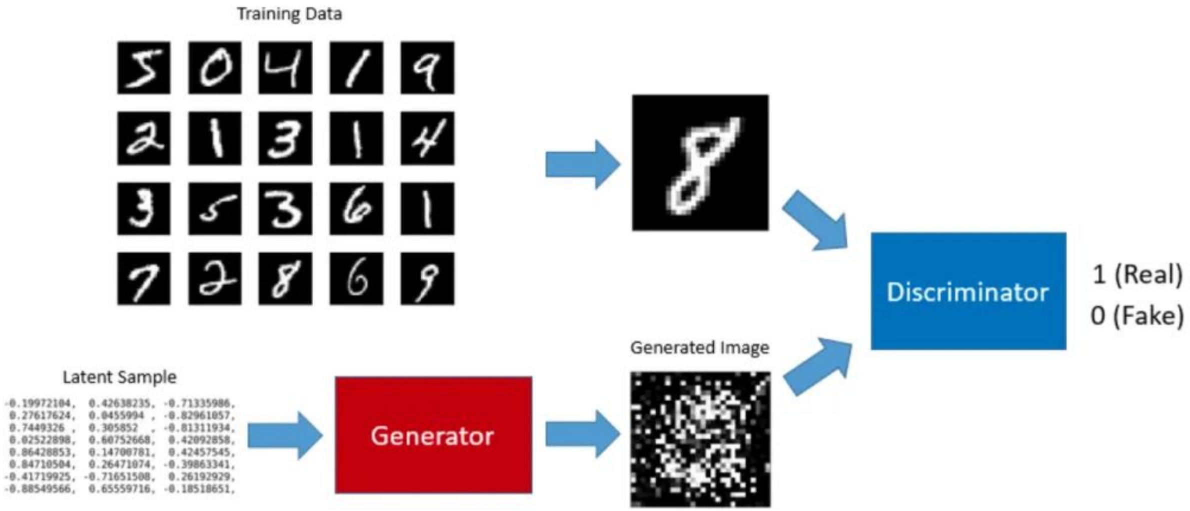
- Which common GAN architecture performed the best on the MNIST Dataset?
- What are the quantitative differences in terms of:
  - Stability?
  - Result Quality?
- How does training differ between architectures?



What is a GAN?

What is a GAN?

- Type of Neural Network called a Generative Adversarial Network
- Proposed by Ian Goodfellow in his 2014 thesis
- Consists of training 2 competing networks
    - One that learns to generate fake samples
    - One that learns to detect real vs. fake ones



What is a GAN?

- Popular uses of GANs include:
    - Image Generation
    - Music Generation
    - Style Transfer
    - Video Prediction
    - Super Resolution
    - Text-to-image

StyleGAN 2 Results

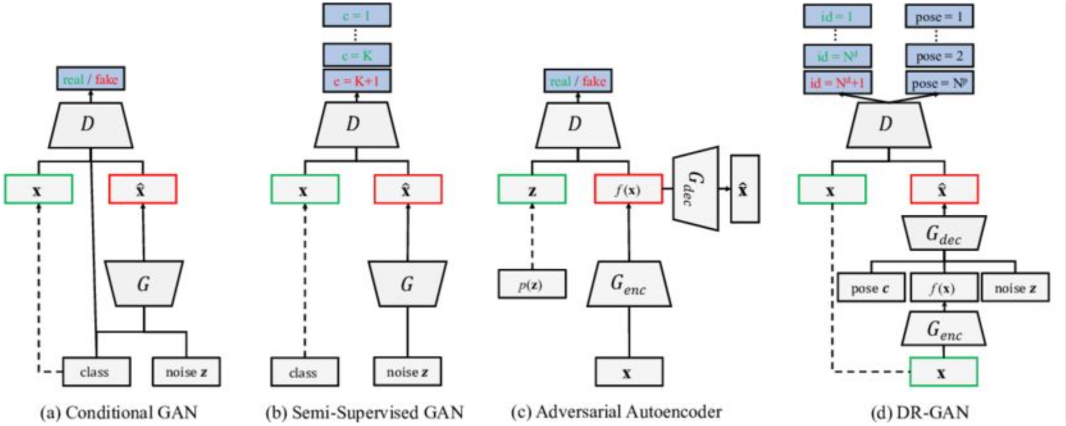
Example of Style Transfer

GAN Architectures

GAN Architectures

- There are many different GAN architectures proposed in the field
- In this work, we compared 3 common ones:
  - Vanilla GAN
  - DCGAN
  - WGAN
- The differences primarily lie in layer shapes and design

Other architectures
not explored

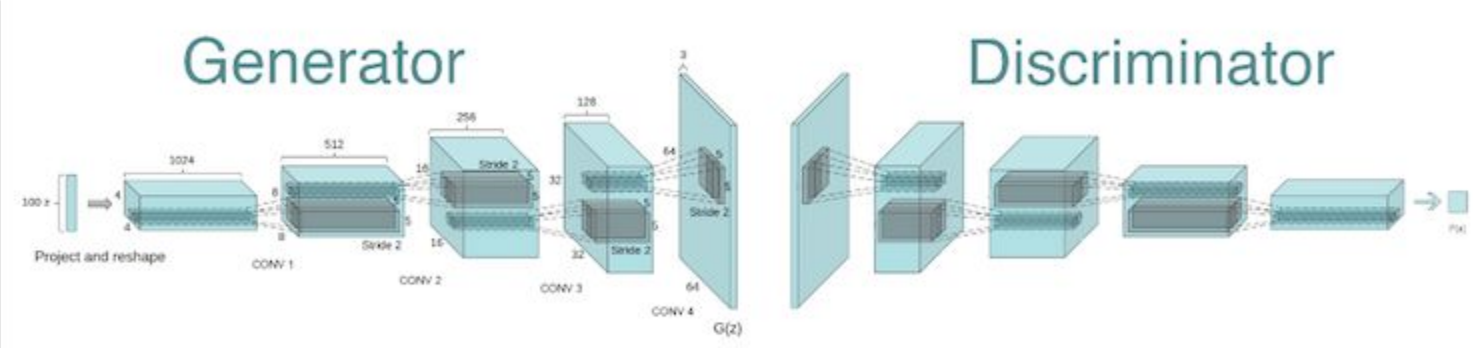


GAN Architectures - Vanilla

- The vanilla architecture is just a dense network for both the model
- Generator:
  - 5 fully-connected layers
  - 128 -> 128 -> 256 -> 512 -> 1024
  - Each hidden layer is activated with a Leaky ReLU
  - The output layer is activated with a TanH for pixel scaling
- Discriminator:
  - 3 fully-connected layesr
  - 1024 -> 512 -> 256 -> 1
  - Leaky ReLU for hidden layers, Sigmoid for output

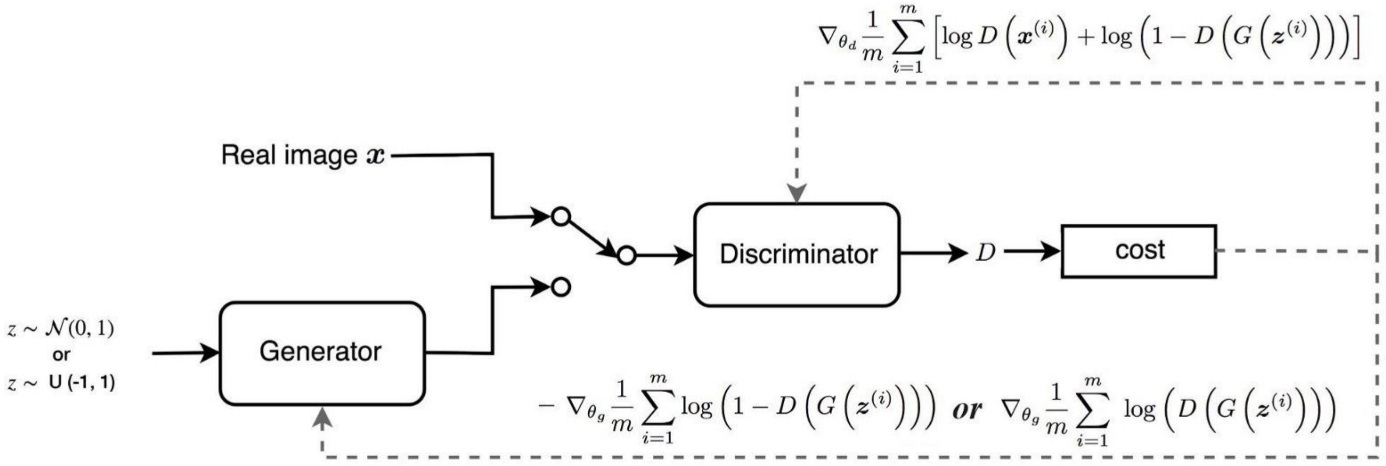
GAN Architectures - DCGAN

- Stands for “Deep Convolutional GAN”
- Relates to having a mirrored deep convolutional network in both models
- Instead of fully-connected layers, convolution layers make sense for image work



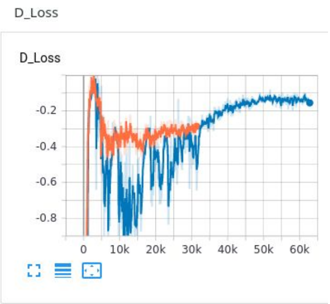
GAN Architectures - WGAN

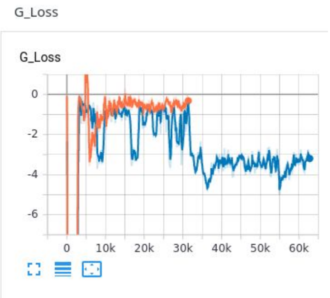
- Stands for “Wasserstein GAN”
- Changes the objective function of the discriminator to work off of “belief”
  - Ie. “An example is 60% real and 40% fake”
- Idea is to allow the Generator to minimize the distance between real and fake easier through a continuous output on the Discriminator side



Experiments/Results

Implementation



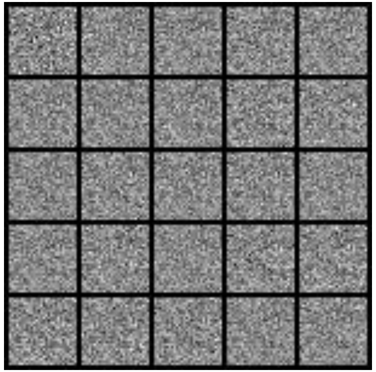


- Using identical hyperparameters and training methodologies, the three GAN architects were implemented in PyTorch.
- Tensorboard was used to view the performance measures of the discriminator and generator during training.
- We used the PyTorch built in loader for the MNIST dataset.

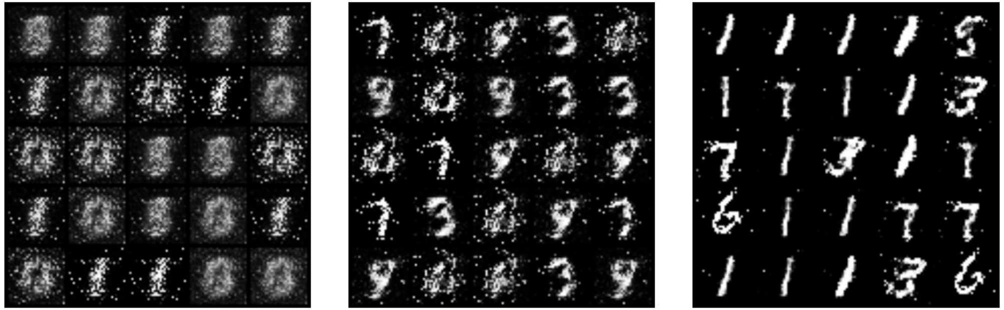
Training Time

- Rates vastly differed between models, both for convergence and time per epoch
- All models were trained on a GTX2060
- Vanilla GAN
  - Quickest time per epoch
- WGAN
  - Similar amount of time to converge as the Vanilla GAN
  - Similar time per epoch
  - Makes sense since architecture similar, just loss function differences
- DCGAN
  - Slowest training time due to massive architecture

Results with
no training

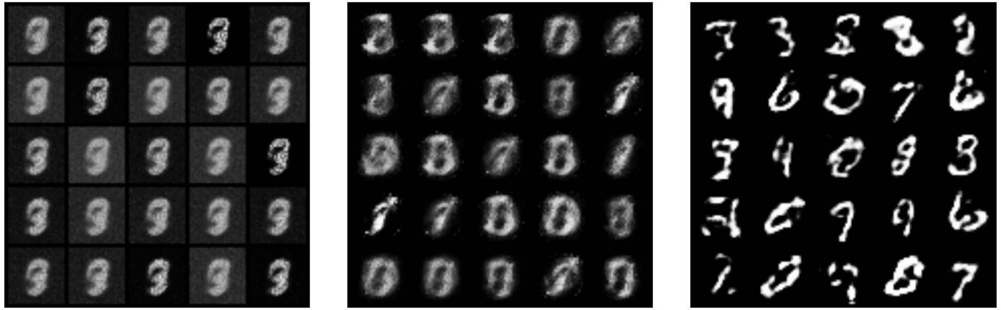


Training Rate



(a) 400 Batches (b) 6000 Batches (c) 187200 Batches

FIG. 3: GAN Results Sampled at Different Epochs



(a) 400 Batches (b) 6000 Batches (c) 187200 Batches

FIG. 4: WGAN Results Sampled at Different Epochs

- DCGANs were able to converge on legible results with the fewest numbers of epochs.
- Our vanilla GAN took the longest to converge on legible outputs

(a) 400 Batches (b) 6000 Batches (c) 187200 Batches

FIG. 4: DCGAN Results Sampled at Different Epochs

Perceived Quality

- To test the quality of the outputs, we sampled ratings from random participants
- Blind samples from a mix of the models
  - We took the last 6 results generated from each architecture -- after 200 epochs
  - Noted that the DCGAN had the best quality

GAN

WGAN

DCGAN

Vanilla GAN	WGAN	DCGAN
7	7	10
6	8	10
7	7	10
6	7	10
5	7	10
4	8	10

Training Data Used

- Restricted the training data to 1/6th the size to test data needed
- Had a massive effect on quality of the samples
    - GAN/WGAN affected most heavily
    - DCGAN performed decently

(a) GAN (b) WGAN (c) DCGAN

FIG. 6: Results with one Sixth of Training Set and trained for 200 Epochs

Questions?

Email us at:

Ryan Missel (rxm7244@rit.edu)
Jeffery Russell (jeffery@jrtechs.net)
Kyle Rivenburgh (ktr5669@rit.edu)

Code:
https://github.com/jrtechs/CSCI-431-final-GANs