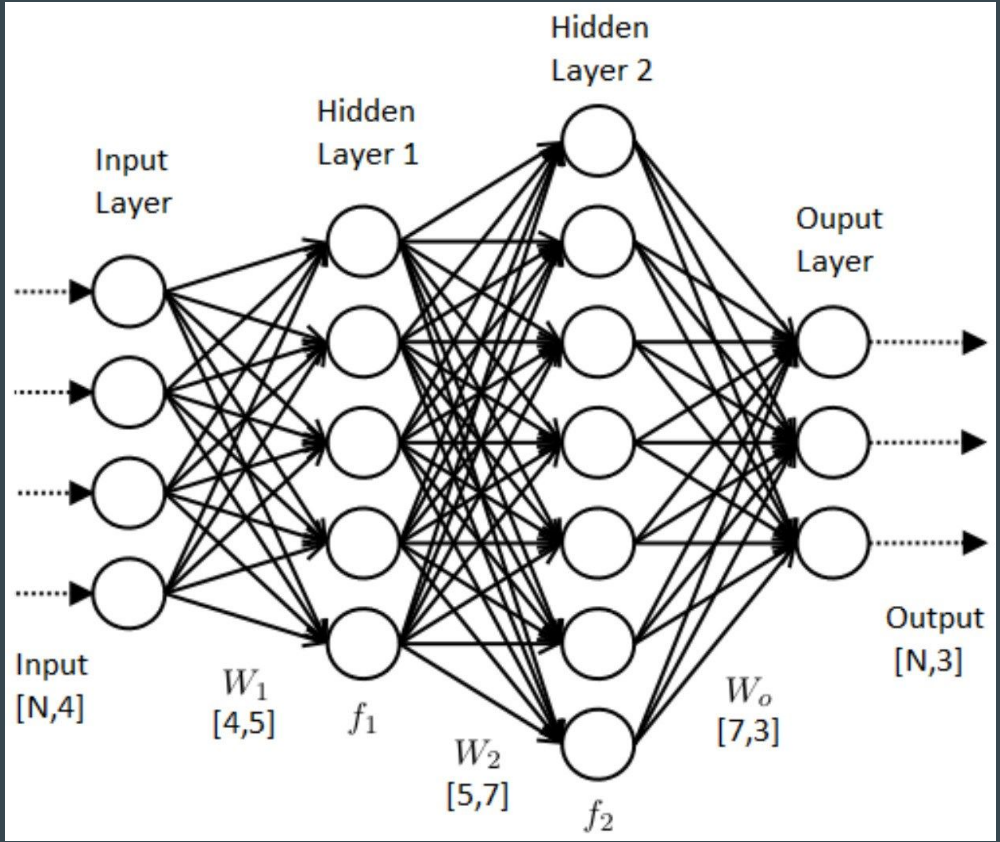
ShenaniGANs: A Discussion of Generative Adversarial Networks

Quinn Meier and John Pace

https://arxiv.org/pdf/1406.2661.pdf

Review of Neural Networks

Feature Vector -> Feed Into First Layer -> Matrix Multiplication with Weights -> Activation Function/Dropout -> Repeat Until Output -> Backpropagate, train until satisfactory



The Motivation

Deep learning does best with discriminative tasks; does less well with generative tasks

Authors posit that this is due to the relative ease of using piecewise linear units in classification tasks vs. generative tasks

So what if we put a discriminative task inside of the generative task?

Discussion of Related Work

Other deep generative models try to provide actual probability distributions with parameters learned by training

These often have ugly likelihood gradients to work with - intractable

'Generative Machines' don't specify an actual probability distribution - they just try to pop out a function that recreates samples

How to Make an Adversarial Net

To learn a generator's 'distribution' over some data:

1) Define a prior on input noise
2) Define the generator, a map from the noise space to the data space using a differentiable function modeled by a DNN
3) Define a discriminator function modeled by a DNN that returns the probability some vector came from the actual data
4) Train discriminator to maximize correct assignments, train generator to confuse discriminator

See Algorithm 1 in paper for pseudocode implementation

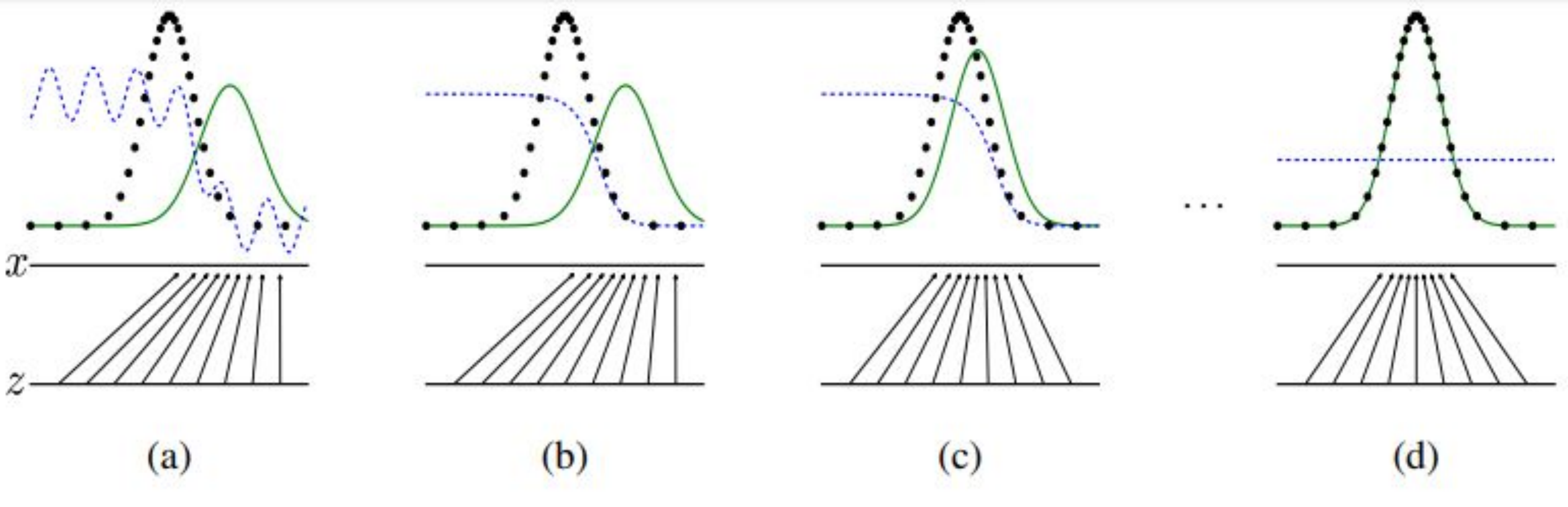


Figure 1 from the journal article - a visual representation of the training process

Discriminator: Blue
Generator: Green
Data: Black

Arrows on bottom represent the function of the generator

A Brief Foray Into Game Theory

All that corresponds to playing a ‘minimax game’:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

The generator tries to minimize the maximum performance of the discriminator. The first term corresponds to the discriminator guessing correctly actual data came from the dataset, the second term corresponds to the discriminator guessing correctly that generated data came from the generator.

Some Caveats

Can’t optimize discriminator to completion in the inner loop of training - will be horribly expensive and lead to overfitting

May need to use alternate criterion to train generator in early stages: if discriminator is too good, generator can’t learn using second term on previous slide

Theoretical Work

Not going to go into detail - involves measure theory and just a bunch of number shuffling

For annotated proof, see:

https://srome.github.io/An-Annotated-Proof-of-Generative-Adversarial-Networks-with-Implementation-Notes/

The Important Stuff

The theoretical work section proves that:

1) The minimax game has a global minimum when the generator's distribution is equal to the data's distribution
2) Given enough network capacity, the ability to update the generator's distribution to improve the minimax game's criterion, and suitable time for the discriminator to train towards its optimum for each generator update, the generator's distribution will converge to the data's distribution

Overview of Experiments

Given a few standard image datasets, the model was trained to create new images

Test set data was compared to a Gaussian Parzen window fit to generated samples using log likelihood

See paper for more generated images

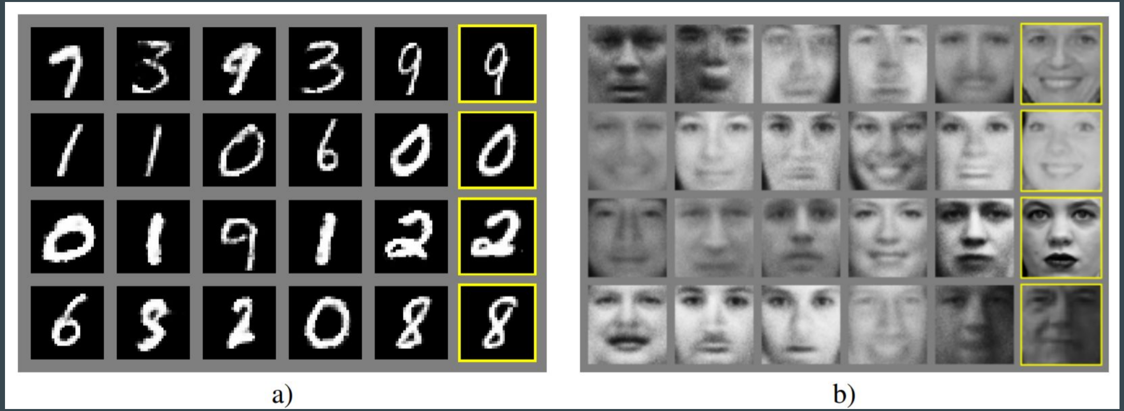


Image from Experiments section of article

Advantages/Disadvantages

Advantages:

- Doesn’t use Markov chains - efficient
- Only calculates gradients using backpropagation - efficient
- No inference during learning - efficient
- Model is very flexible - just needs to be differentiable
- Generator doesn’t see actual data - possibly helps with generalizability

Disadvantages:

- No explicit representation of the generator’s distribution over the data space
- Requires tuning of generator and discriminator’s dynamics - possible for generator to get stuck in degeneracies and lose representability

What Are GANs Being Used For?

Image 'super-resolution' - SRGAN: https://arxiv.org/abs/1609.04802

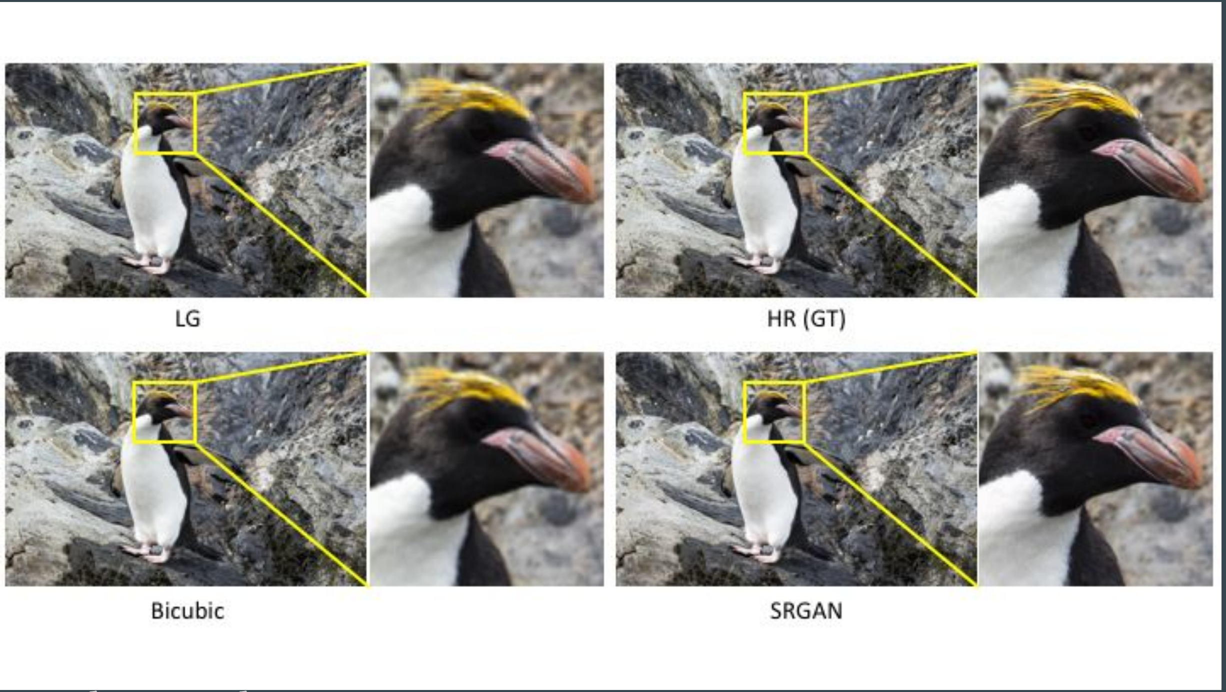


Image taken from https://modelzoo.co/model/srgan

For more examples, read: https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900

What Are GANs Being Used For?

Cross-domain transfer - CycleGAN: https://arxiv.org/abs/1703.10593

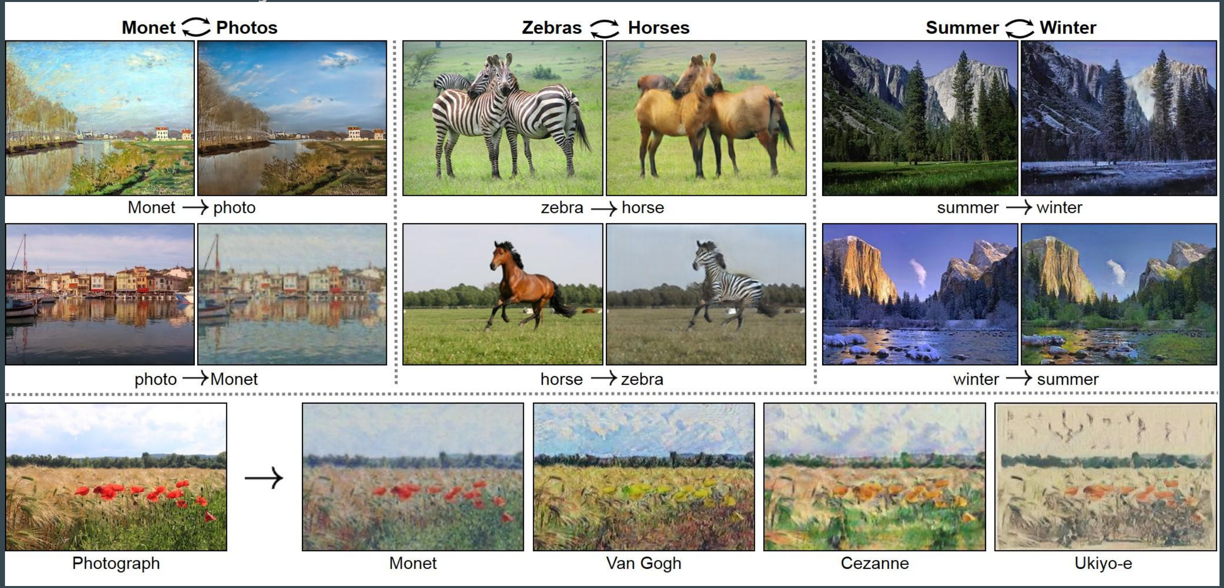


Image from linked CycleGAN journal article

(see also: how CycleGAN ‘cheats’: https://arxiv.org/abs/1712.02950)

For more examples, read: https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900

What Are GANs Being Used For?

Text-to-image translation - StackGAN: https://arxiv.org/abs/1612.03242

Image from linked StackGAN journal article

For more examples, read: https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900

Some fun with StyleGAN (https://arxiv.org/pdf/1812.04948.pdf)

https://www.thispersondoesnotexist.com/

https://thesecatsdonotexist.com/

http://www.whichfaceisreal.com/index.php