Photo via Art and Artificial Intelligence Laboratory, Rutgers University

Generative Adversarial Networks

Advanced Topics in Machine Learning

Generative Adversarial Network

- Generative adversarial networks (GANs) provide a way to learn deep representations without extensively annotated training data.
- They achieve this through deriving backpropagation signals through a competitive process involving a pair of networks.

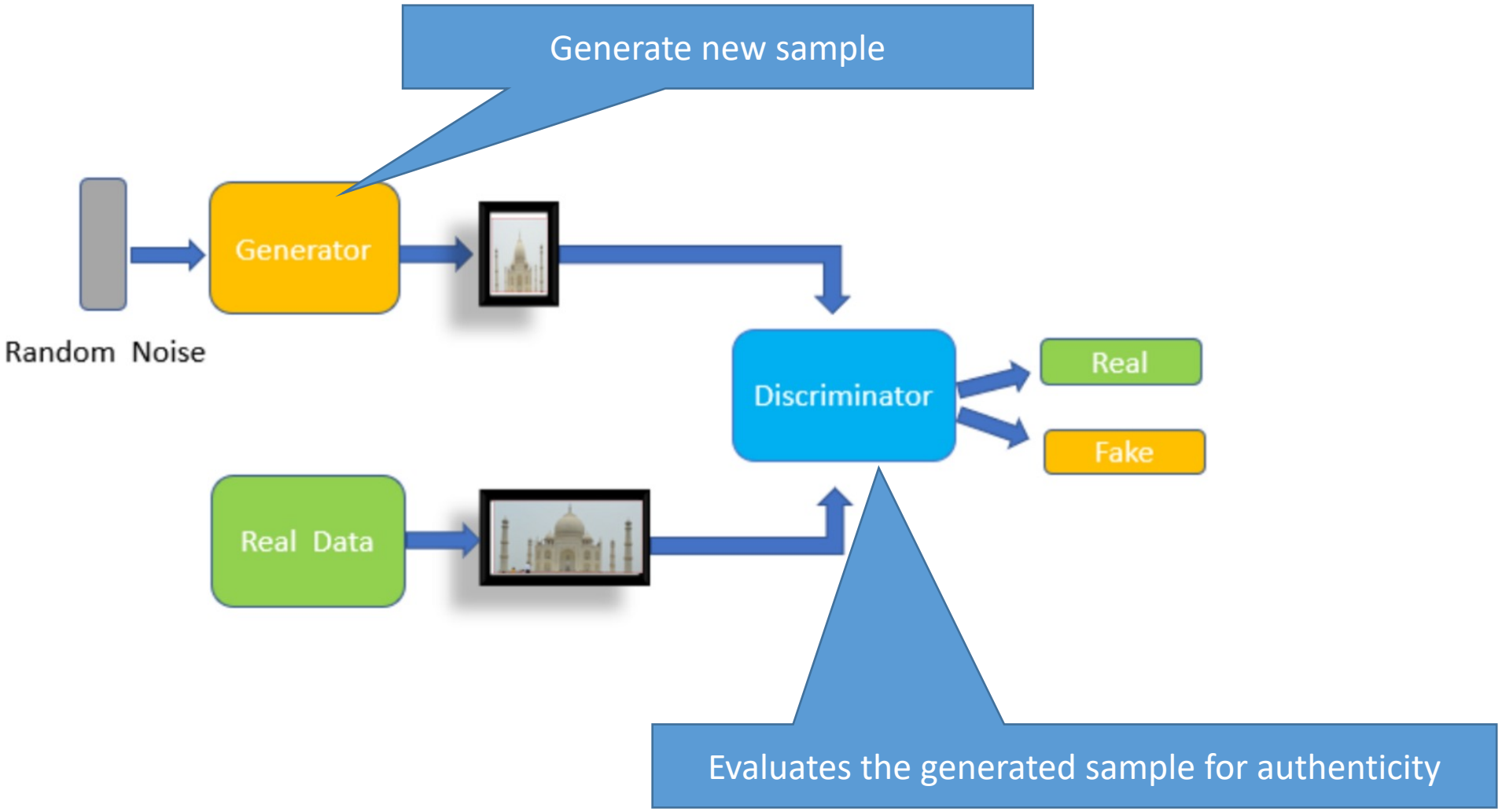
- Unsupervised learning or semi-supervised learning

Generative vs. Discriminative

- Discriminative model map features to labels.
  - They are concerned solely with that correlation. $\hat{y} = f(x)$
- Generative model attempt to predict features given a certain label, instead of predicting a label given certain features
  - They care about how you get x when given a certain label

- Another way to think about it is to distinguish discriminative from generative
  - Discriminative models learn the boundary between classes
  - Generative models model the distribution of individual classes

GAN


Generate new sample
Generator
Random Noise
Discriminator
Real
Fake
Real Data
Evaluates the generated sample for authenticity

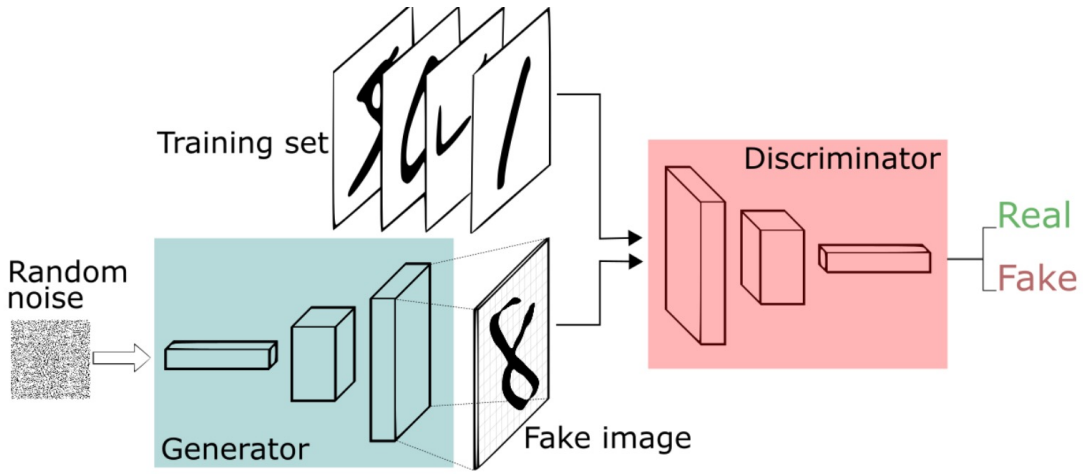
Steps a GAN takes

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

MNIST Example

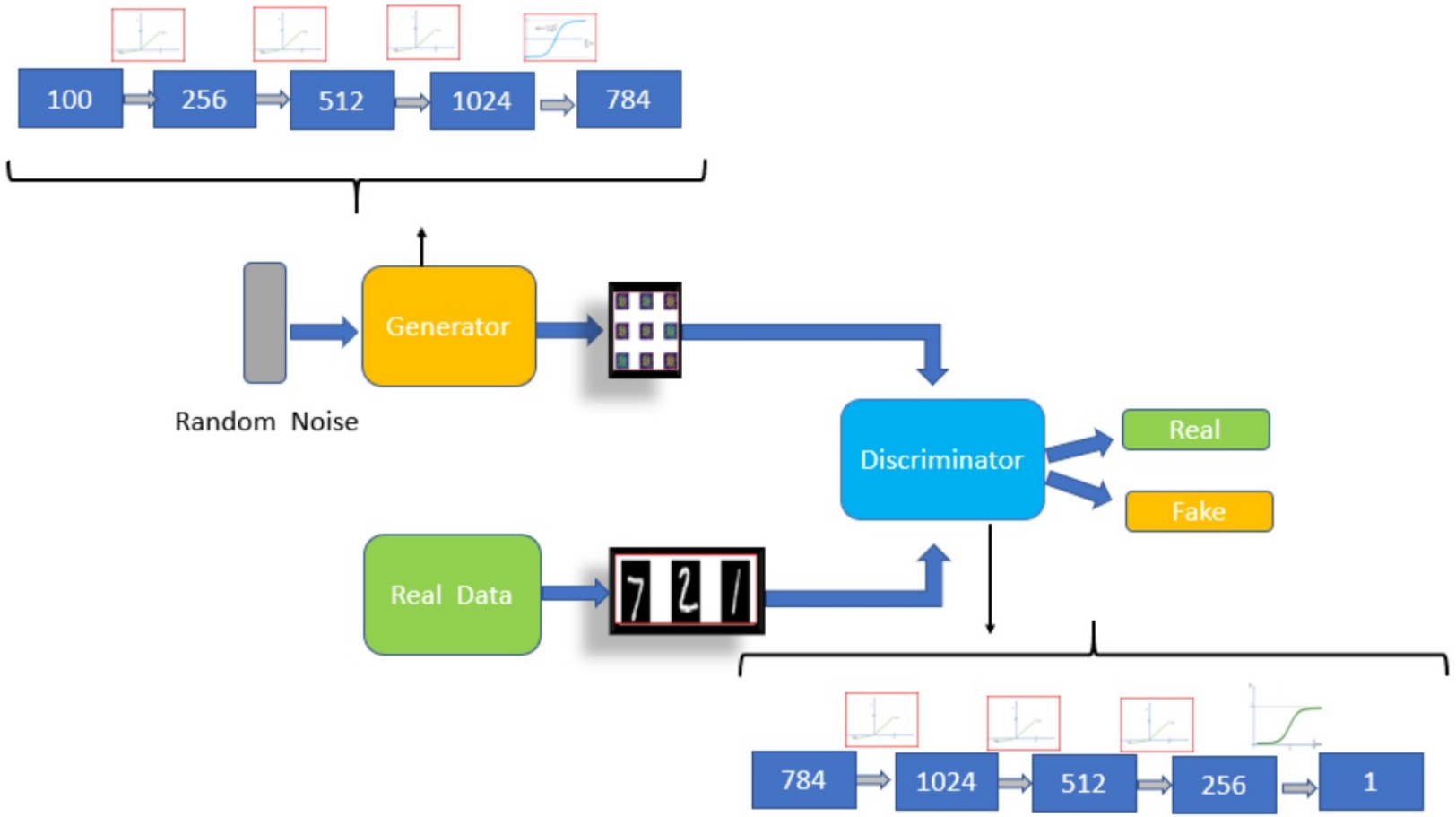
The **discriminator** network is a standard convolutional network that can categorize the input images as real or fake.

Both nets are trying to optimize a different and opposing objective function
As the discriminator changes its behaviour, so does the generator, and vice versa.
Their losses push against each other.



The **generator** is an inverse convolutional network, the generator takes a vector of random noise and upsamples it to an image.

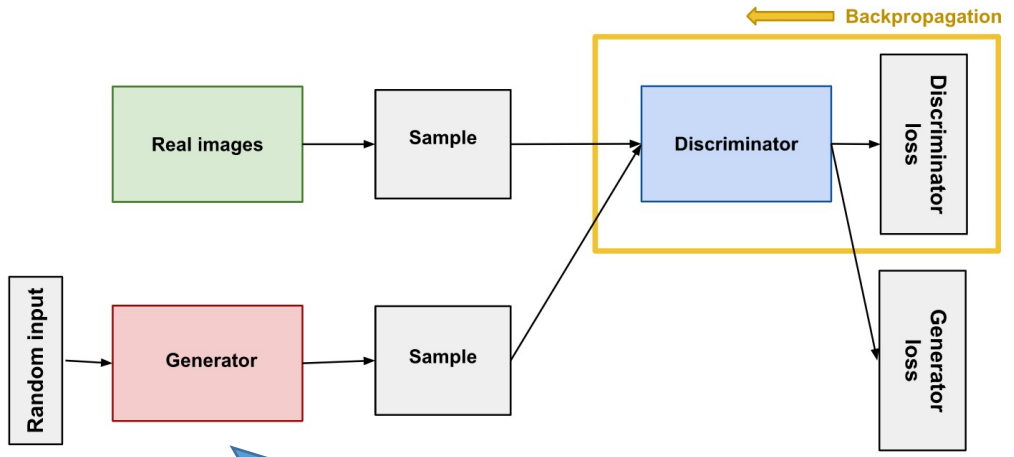
GAN for MNIST



Discriminator

- The discriminator in a GAN is simply a classifier.
- It could use any network architecture appropriate to the type of data it's classifying.

- Training data:
  - Real data – positive samples
  - Fake data – negative samples



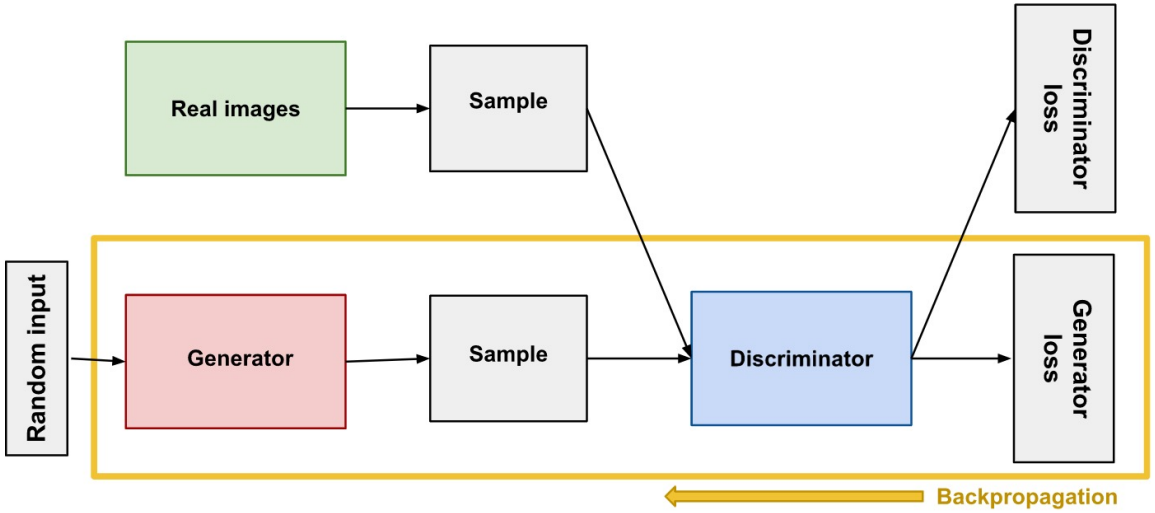
During discriminator training the generator does not train. Its weights remain constant while it produces samples for the discriminator to train on.

Generator - input

- A GAN takes random noise as its input
- it produces a wide variety of data, sampling from different places in the target distribution.

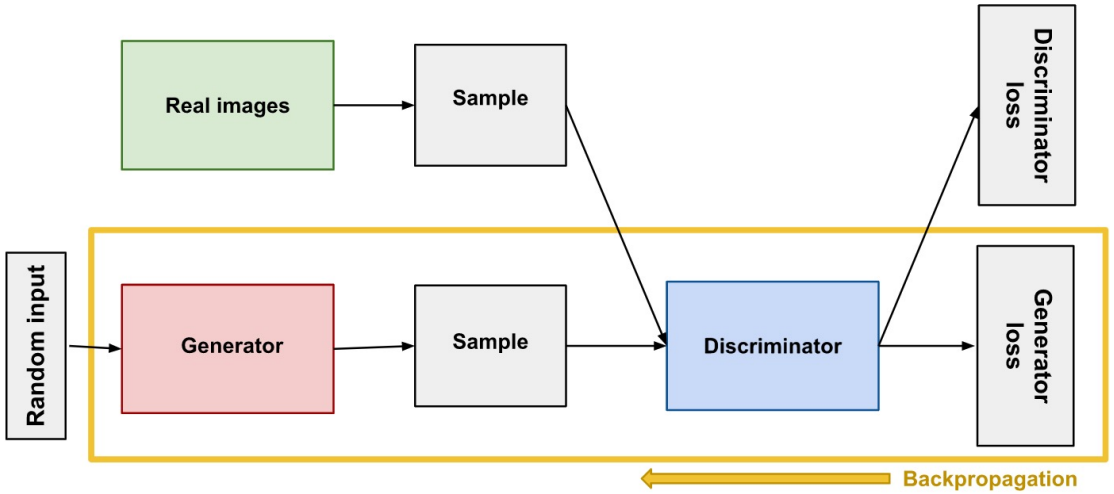
Generator - train

- The generator is not directly connected to the loss.
- The generated samples feeds into the discriminator net, and the discriminator produces the output, forming the Generator loss.
- Discriminator must be included in backpropagation.
- However, the output of Discriminator changes when value of weights change.
- Use gradients to change only the generator weights



Generator – train steps

- One iteration
  - Sample random noise.
  - Produce generator output
  - Get discriminator "Real" or "Fake" classification for generator output.
  - Calculate loss from discriminator classification.
  - Backpropagate through both the discriminator and generator to obtain gradients.
  - Use gradients to change only the generator weights.



GAN Training

- Train a GAN as a whole
  - The discriminator trains for one or more epochs.
  - The generator trains for one or more epochs.
  - Repeat steps 1 and 2 to continue to train the generator and discriminator networks.

This alternating training allows GANs to tackle intractable generative problems.

Convergence

- As the generator improves with training, the discriminator performance gets worse.
  - the discriminator can't easily tell the difference between real and fake.
  - If the generator succeeds perfectly, then the discriminator has a 50% accuracy.
- The discriminator feedback gets less meaningful over time
  - If the GAN continues training past the point when the discriminator is giving completely random feedback, then the generator starts to train on junk feedback, and its own quality may collapse.

  - For a GAN, convergence is often a fleeting, rather than stable state.

Loss Function

- Loss functions should reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.
- TF-GAN implements many loss functions
  - Minimax loss
  - Wasserstein loss

Minimax Loss

- The generator tries to minimize the loss function while the discriminator tries to maximize it

$$E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$$

- $D(x)$: output of discriminator
- $G(z)$: output of generators when given noise z
- E_x: expected value over all real data instances
- E_z: expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).

Wasserstein GAN

- A modification of the GAN scheme
  - The discriminator does not actually classify instances.
  - For each instance it outputs a number.
  - This number does not have to be in [0..1], so we can't use 0.5 as a threshold to decide whether an instance is real or fake.
  - Discriminator training just tries to make the output bigger for real instances than for fake instances.

Wasserstein Loss

- Critic(Discriminator) Loss
  - Maximize $D(x) - D(G(z))$
  - [average critic score on real images] – [average critic score on fake images]
- Generator Loss
  - Maximize the function $D(G(z))$
  - [average critic score on fake images]

Where the average scores are calculated across a mini-batch of samples.

Applications

Pix2Pix:
https://affinelayer.com/pixsrv/

Face2Ramen
https://www.youtube.com/watch?v=YGBzYsf61QY

Adversarial Examples In The Physical World - Demo
https://www.youtube.com/watch?v=zQ_uMenoBCk

Interview Ian Goodfellow
https://www.youtube.com/watch?v=pWAc9B2zJS4