SVKM'S NMIMS Deemed to be UNIVERSITY

GANs

B.Tech. Data Science, NMIMS

By,

Bilal Hungund, Data Scientist, Halliburton

Introduction

A generative adversarial network (GAN) has two parts:

- The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- The discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

Reference: https://developers.google.com/machine-learning/gan

Generated Data

Discriminator

Real Data

FAKE

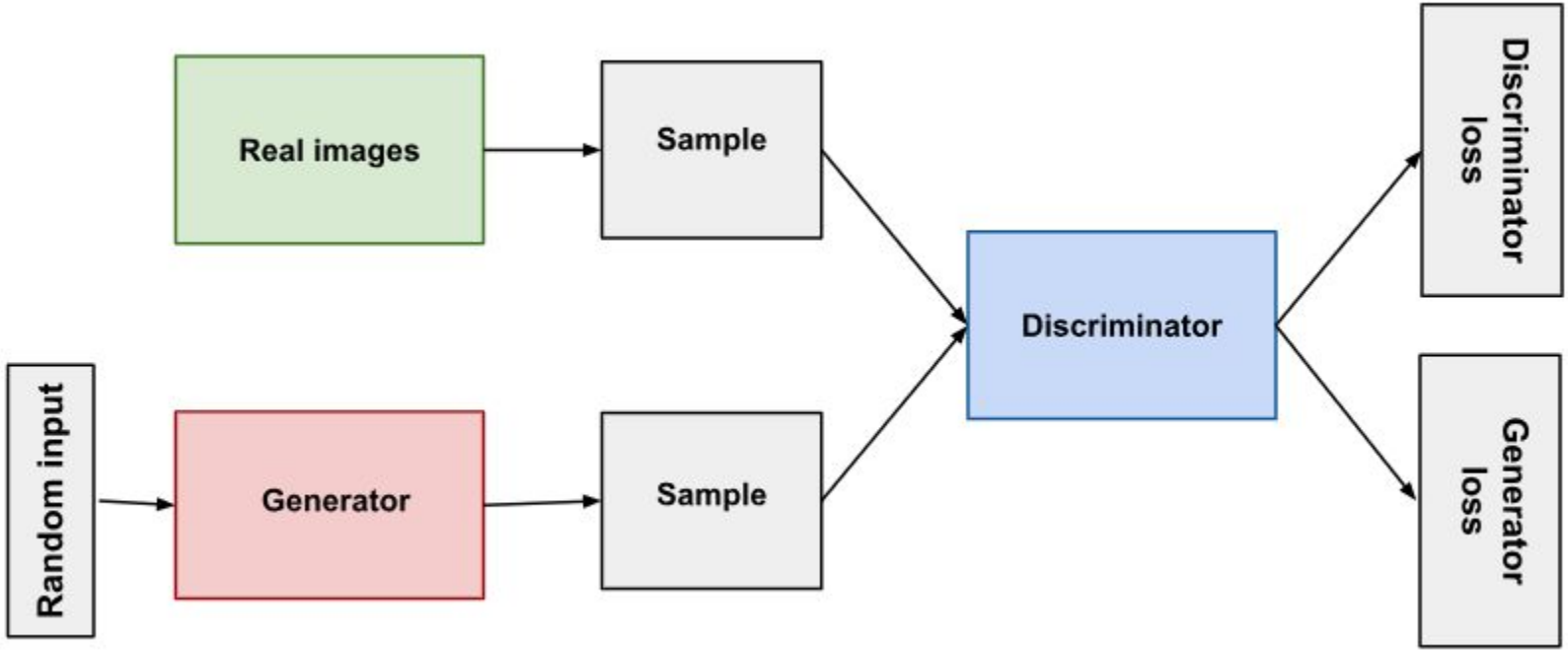
REAL

FAKE

REAL

REAL

REAL


Real images
Sample
Discriminator loss
Discriminator
Random input
Generator
Sample
Generator loss

Discriminator

The discriminator's training data comes from two sources:

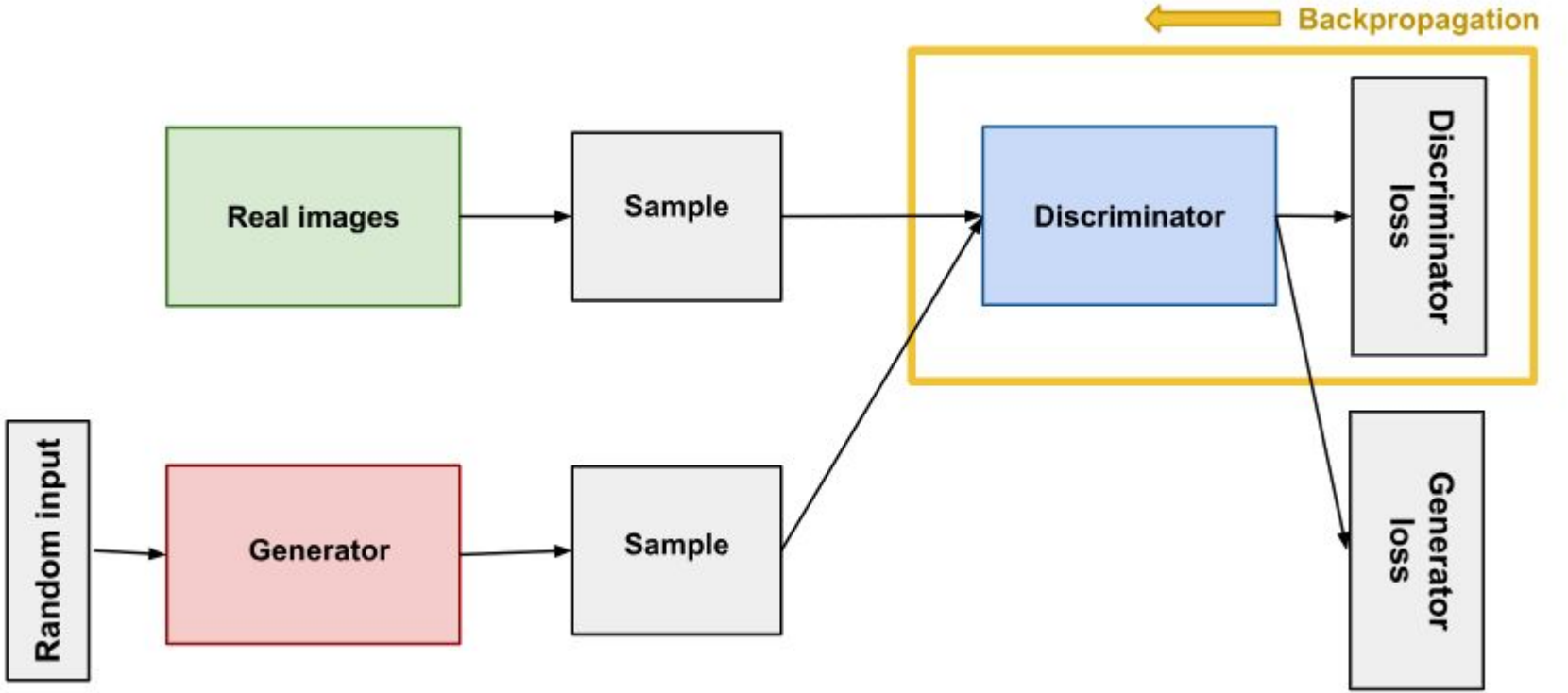
- Real data instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
- Fake data instances created by the generator. The discriminator uses these instances as negative examples during training.

The discriminator connects to two loss functions. During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss. We use the generator loss during generator training.

During discriminator training:

- The discriminator classifies both real data and fake data from the generator.
- The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.

The discriminator in a GAN is simply a classifier. It tries to distinguish real data from the data created by the generator. It could use any network architecture appropriate to the type of data it's classifying.

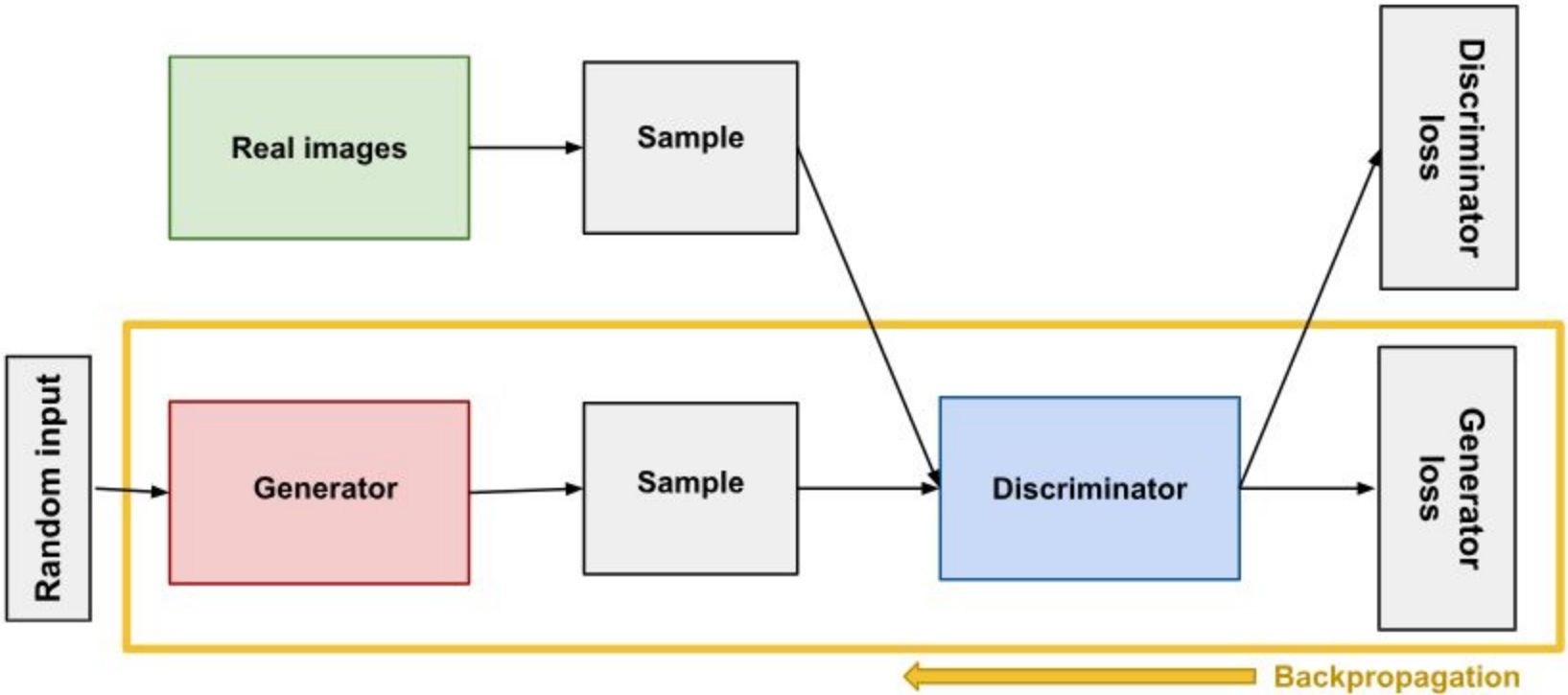


Generator

The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real.

Generator training requires tighter integration between the generator and the discriminator than discriminator training requires. The portion of the GAN that trains the generator includes:

1. random input
2. generator network, which transforms the random input into a data instance
3. discriminator network, which classifies the generated data
4. discriminator output
5. generator loss, which penalizes the generator for failing to fool the discriminator


Real images
Sample
Discriminator loss
Random input
Generator
Sample
Discriminator
Generator loss
Backpropagation

Training GANs

Which network should I train first?: Discriminator!

But with what training data?

The Discriminator is a Binary classifier.

- The Discriminator has two class - Real and Fake.
- The data for Real class if already given: THE TRAINING DATA
- The data for Fake class? -> generate from the Generator

But how? What's our training objective?

- Generate images from the Generator
- such that they are classified incorrectly by the Discriminator

Step 1: Train the Discriminator using the current ability of the Generator.

Step 2: Train the Generator to beat the Discriminator

Convergence

- As the generator improves with training, the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake. If the generator succeeds perfectly, then the discriminator has a 50% accuracy. In effect, the discriminator flips a coin to make its prediction.
- This progression poses a problem for convergence of the GAN as a whole: 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

GANs try to replicate a probability distribution. They should therefore use loss functions that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.

In the paper that introduced GANs, the generator tries to minimize the following function while the discriminator tries to maximize it:

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

- D(x) is the discriminator's estimate of the probability that real data instance x is real.
- Ex is the expected value over all real data instances.
- G(z) is the generator's output when given noise z.
- D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
- Ez is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
- The formula derives from the cross-entropy between the real and generated distributions.