Generative Adversarial Networks (GANs)

By

Dr Muhammad Atif Tahir

Introduction

- Generative adversarial networks (GANs) are an exciting recent innovation in machine learning

- GANs are *generative* models: they create new data instances that resemble your training data

- For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person

Sample Images Created by GAN

Images generated by a GAN created by NVIDIA

Introduction

- GANs achieve this level of realism by pairing a generator and discriminator
- Generator learns to produce the target output, with a discriminator, which learns to distinguish true data from the output of the generator
- The generator tries to fool the discriminator, and the discriminator tries to keep from being fooled

Main Objective of this Lecture

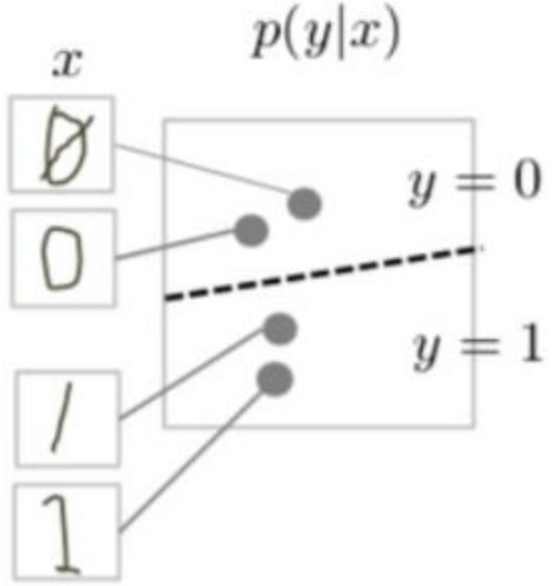
- Understand the difference between generative and discriminative models
- Identify problems that GANs can solve
- Understand the roles of the generator and discriminator in a GAN system
- Understand the advantages and disadvantages of common GAN loss functions
- Identify possible solutions to common problems with GAN training

Background

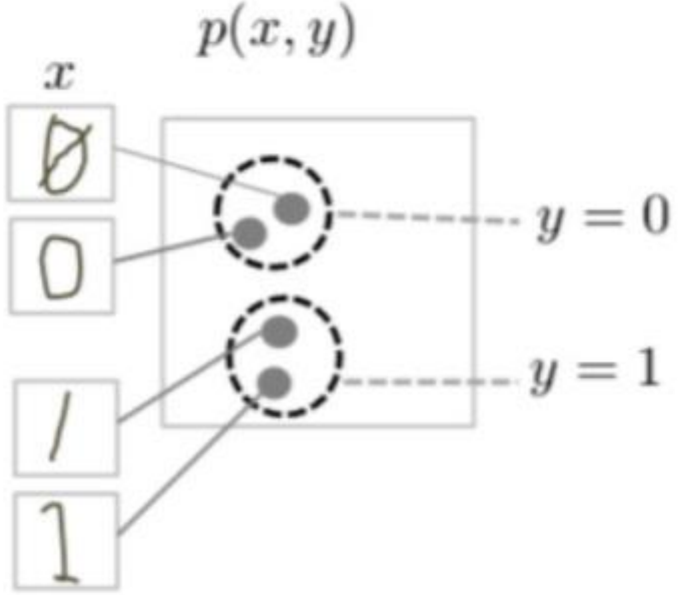
- Generative models can generate new data instances
- Discriminative models discriminate between different kinds of data instances
- More formally, given a set of data instances X and a set of labels Y:
  - Generative models capture the joint probability p(X, Y), or just p(X) if there are no labels
  - Discriminative models capture the conditional probability p(Y| X)

Background

- Discriminative Model



- Generative Model

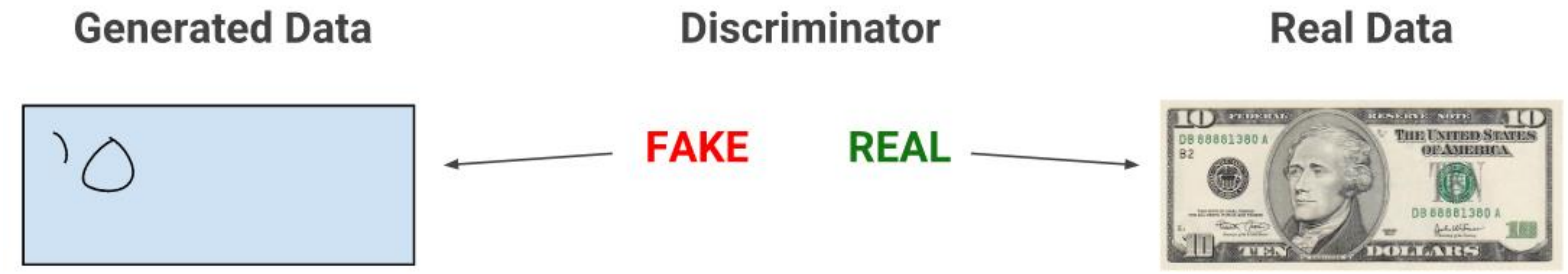


Overview of GAN Structure

- 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

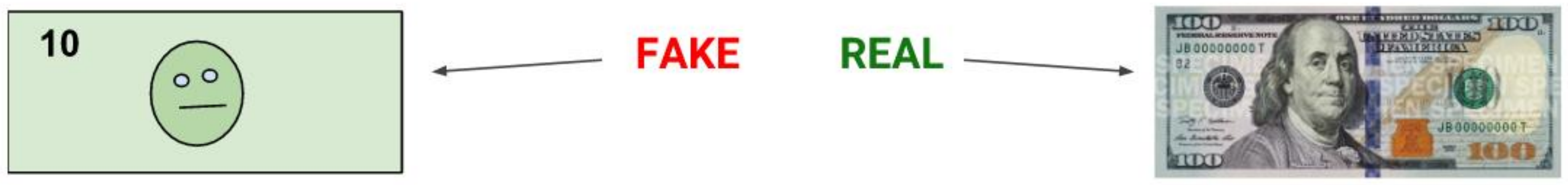
Overview of GAN Structure

- When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:

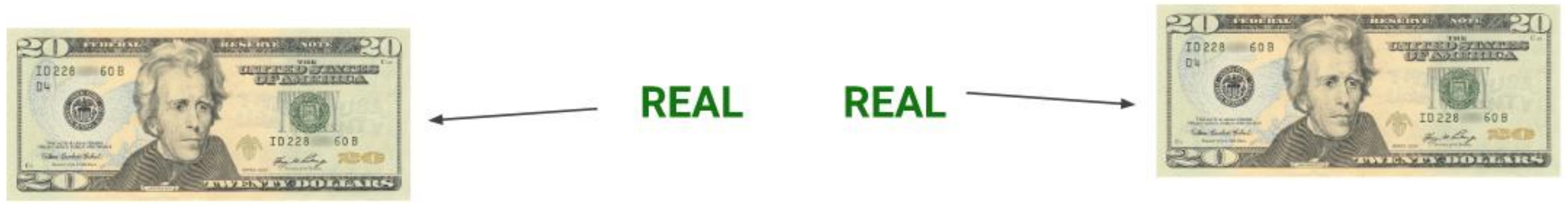


Overview of GAN Structure

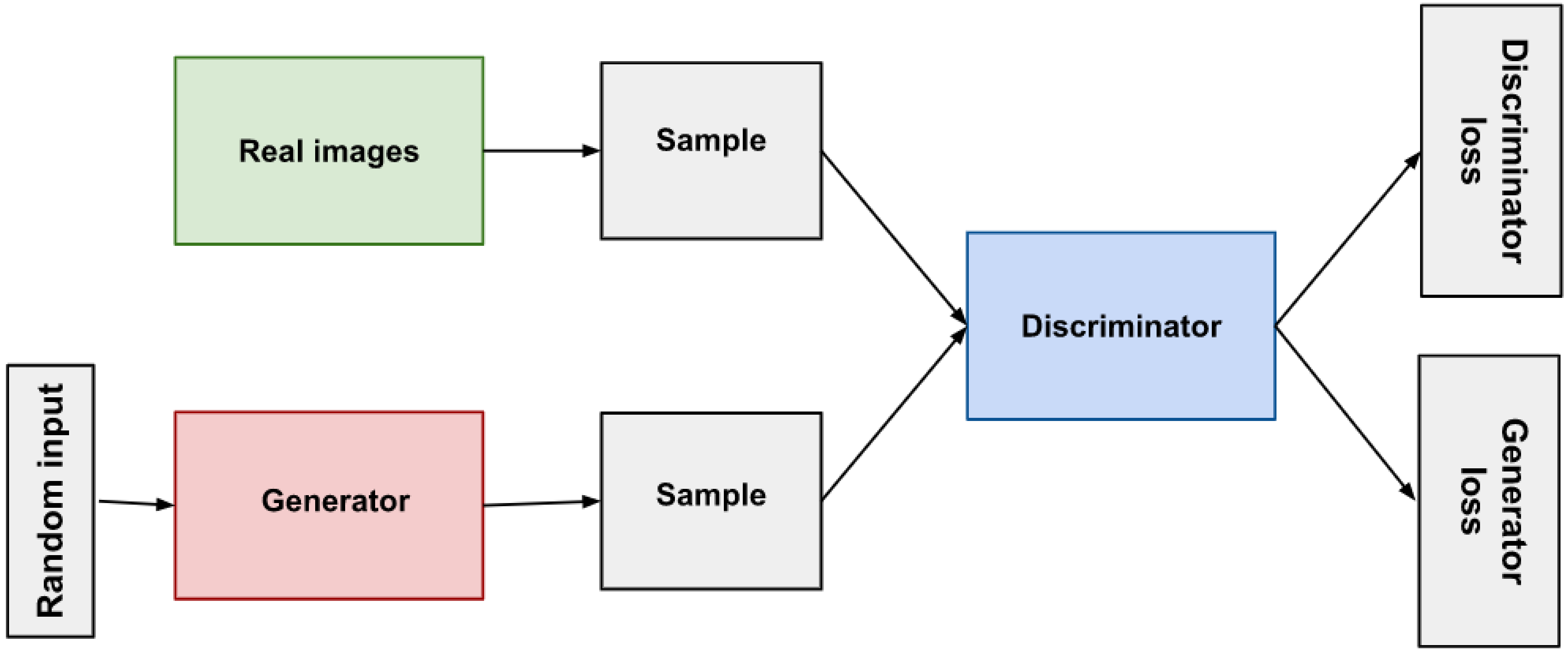
- As training progresses, the generator gets closer to producing output that can fool the discriminator:



Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases



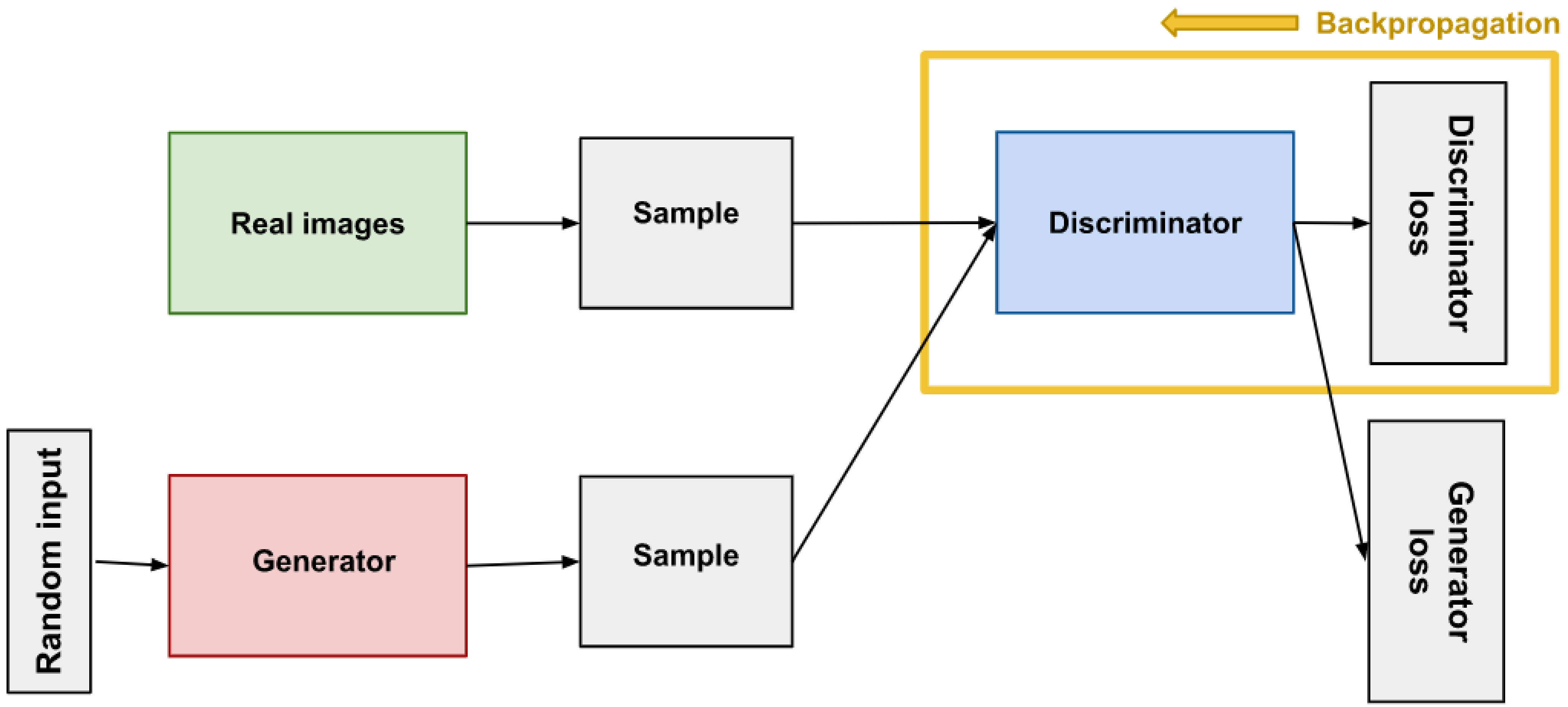
GAN Block Diagram



Overview of GAN Structure

- Both the generator and the discriminator are neural networks. The generator output is connected directly to the discriminator input
- Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights

Discriminator



Discriminator

- 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

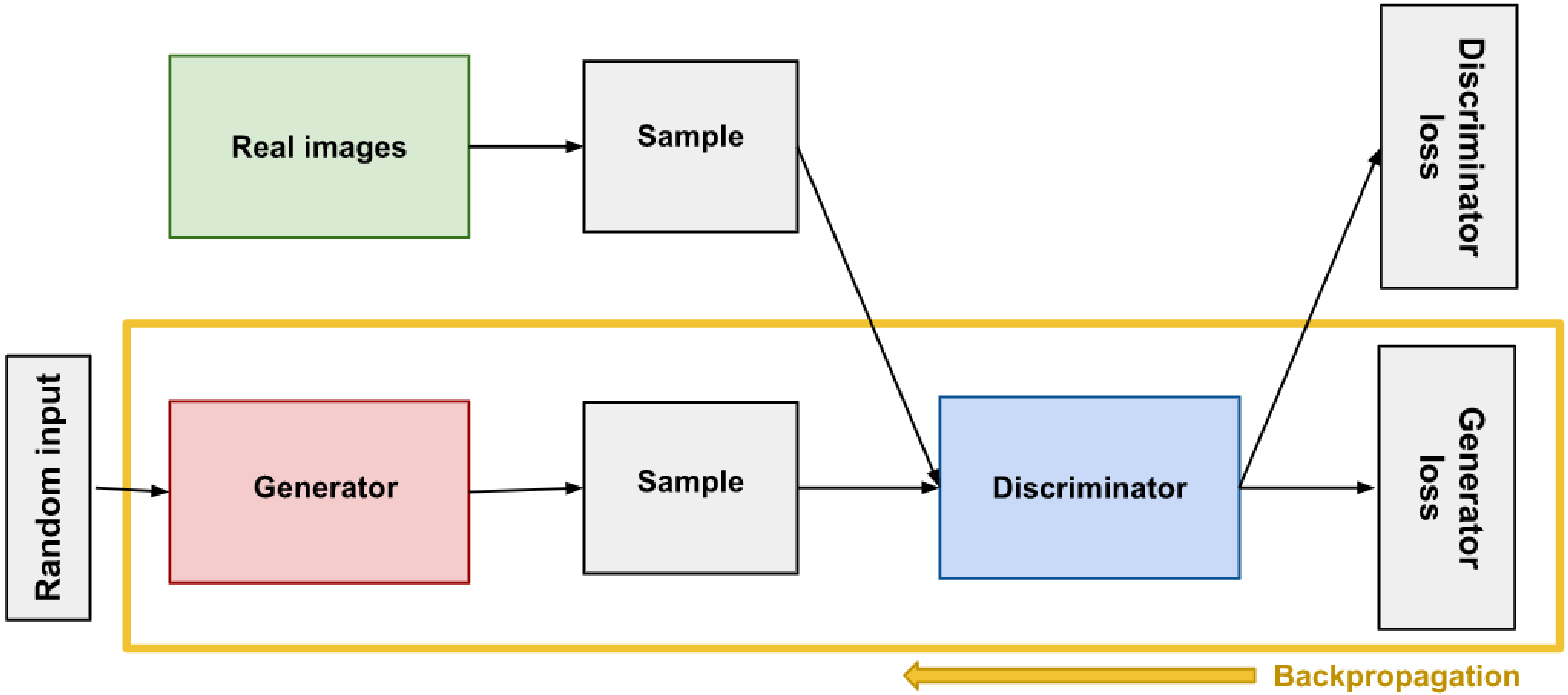
- The discriminator's training data comes from two sources:
  - Real data instances, such as real pictures of people. The discriminator uses these instances as positive examples during training
  - Fake data instances created by the generator. The discriminator uses these instances as negative examples during training

Discriminator Training

- The discriminator connects to two loss functions
- During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss.
- 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 updates its weights through backpropagation from the discriminator loss through the discriminator network

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

- Sample random noise
- Produce generator output from sampled random noise
- 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

- 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
- Careful with Convergence as not to overtrain the generator otherwise generator will start training with less meaningful discriminator

Loss Function

- Original GAN: minimax Loss

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

- 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
- The generator can't directly affect the log(D(x)) term in the function, so, for the generator, minimizing the loss is equivalent to minimizing log(1 - D(G(z)))

Modified MiniMax Loss

- The original GAN paper notes that the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy

- The paper therefore suggests modifying the generator loss so that the generator tries to maximize log D(G(z))

Wasserstein Loss

- Here no classification for discriminator
- Discriminator becomes “Critic”: Critic Loss: D(x) - D(G(z))
- The discriminator tries to maximize this function
- In other words, it tries to maximize the difference between its output on real instances and its output on fake instances
- Generator Loss: D(G(z))
- The generator tries to maximize this function. In other words, It tries to maximize the discriminator's output for its fake instances

Advantages of Wasserstein Loss

- Wasserstein GANs are less vulnerable to getting stuck than minimax-based GANs and avoid problems with vanishing gradients
- The earth mover distance also has the advantage of being a true metric: a measure of distance in a space of probability distributions
- Cross-entropy is not a metric in this sense.

Common Problems in GAN

- **Problem #1: Vanishing Gradient:** Research has suggested that if your discriminator is too good, then generator training can fail due to vanishing gradients

- In effect, an optimal discriminator doesn't provide enough information for the generator to make progress

- Sol: Wasserstein loss and Modified minimax loss

Common Problems in GAN

- **Problem #2: Mode collapse** When the generators rotate through a small set of output types, this form of GAN failure is called Mode Collapse

- In effect, an optimal discriminator doesn't provide enough information for the generator to make progress

- Sol: Wasserstein loss and Modified minimax loss

Common Problems in GAN

- **Problem #3: Failure to Converge:** Often problem in GAN

- Sol:

(i) Adding noise to discriminator inputs

(ii) Penalizing discriminator weights

GANs Variations

- Progressive GANs
- Conditional GANs
- Image-to-Image Translation
- CycleGAN
- Text-to-Image Synthesis
- Super-resolution
- Face Inpainting
- Text to Speech
- And Many More

Conclusions

- Generative models can generate new data instances
- Discriminative models discriminate between different kinds of data instances
- GAN architecture is discussed in this lecture which is fundamental to create new instances for data augmentation

References

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

- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Networks”, Machine Learning
- Martin Arjovsky, Soumith Chintala, Léon Bottou, “Wasserstein GAN” Machine Learning