CS-4053 Recommender System

Fall 2023

Lecture 10: Generative Adversarial Network (GAN)

Course Instructor: Syed Zain Ul Hassan

National University of Computer and Emerging Sciences, Karachi

Email: zain.hassan@nu.edu.pk

Introduction

- **Generative Adversarial Networks** *(GANs)* is an unsupervised architecture that use two neural networks, competing against each other (thus the "*adversarial*") in order to generate new, synthetic instances of data that can pass for real data.

- GANs can learn to *mimic* any distribution of data.

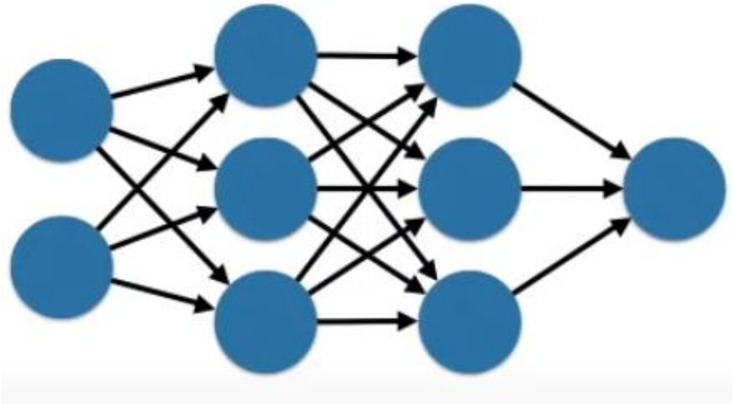
- It can be used for image generation, voice generation and video generation tasks.

Image Generation

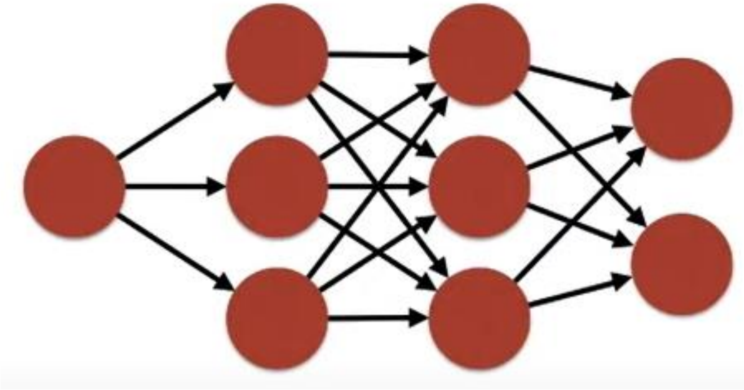
- [This Person Does Not Exist - Random Face Generator (this-person-does-not-exist.com)](this-person-does-not-exist.com)

Introduction

- A simple GAN architecture consists of two networks, a **generator** and a **discriminator**.



Generator



Discriminator

Introduction

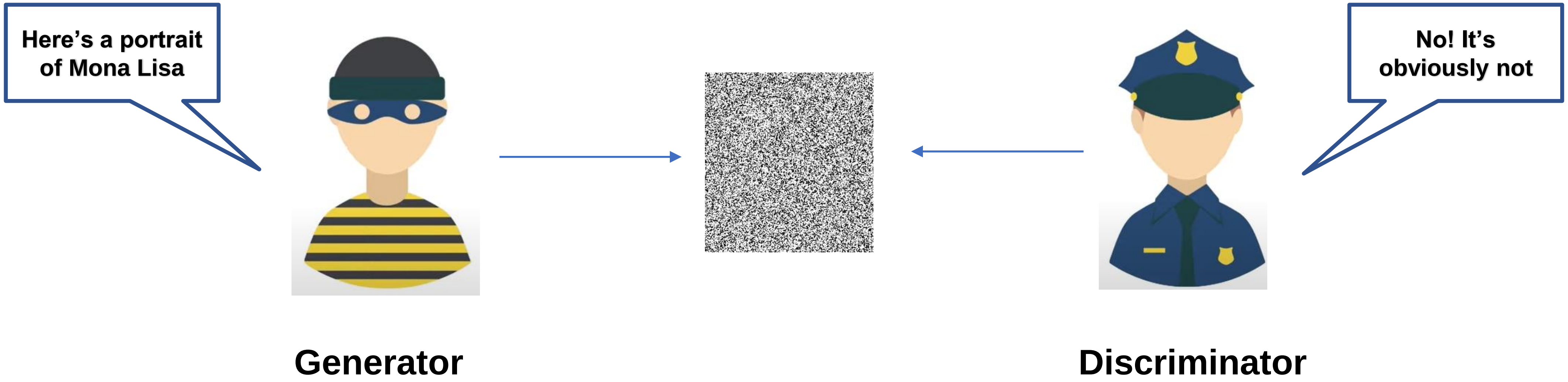
- A simple GAN architecture consists of two networks, a **generator** and a **discriminator**.

Generator

Discriminator

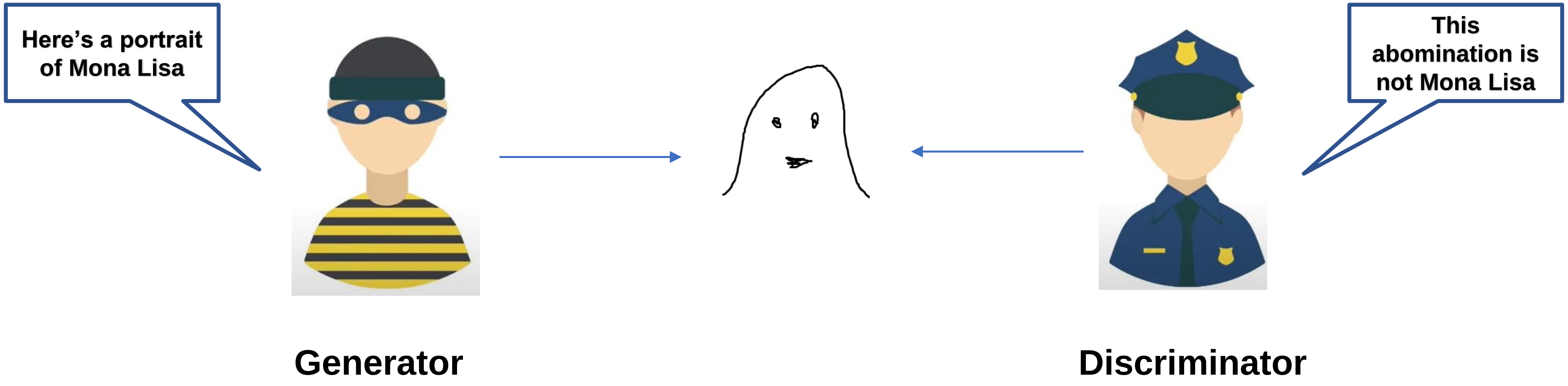
Adversarial Learning

- The **generator** starts with random noise and tries to mimic the target distribution *(e.g., image of a digit, image of Mona Lisa)*.



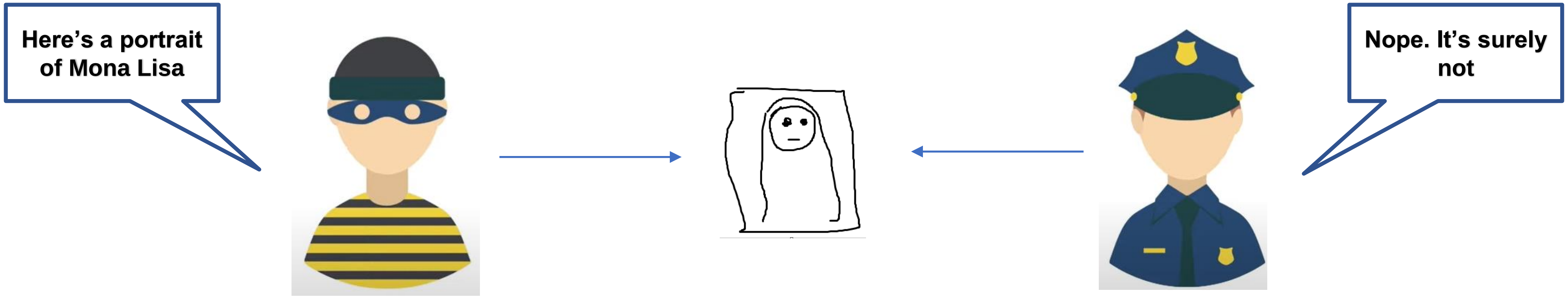
Adversarial Learning

- The **discriminator** has already seen the training data thus it knows what the target distribution is like thus it acts as a critic.



Adversarial Learning

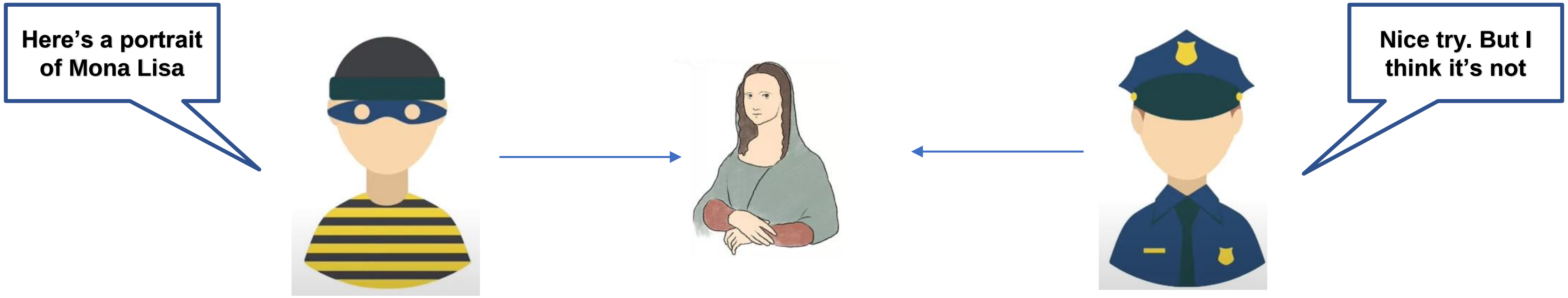
- The **generator** repeatedly tries to fool the discriminator into believing that the generated distribution (image) is a real one.



Generator **Discriminator**

Adversarial Learning

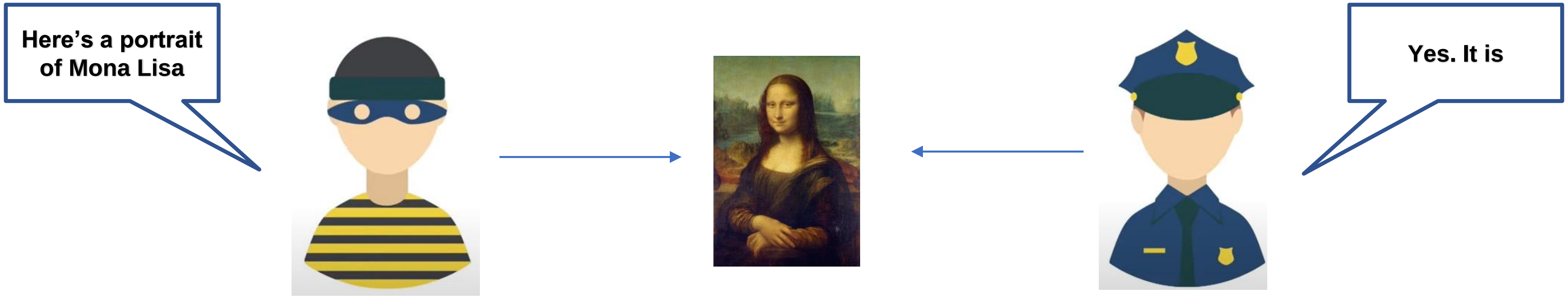
- Based on the feedback *(loss)* of the **discriminator** the quality of generated output is gradually improved.



Generator **Discriminator**

Adversarial Learning

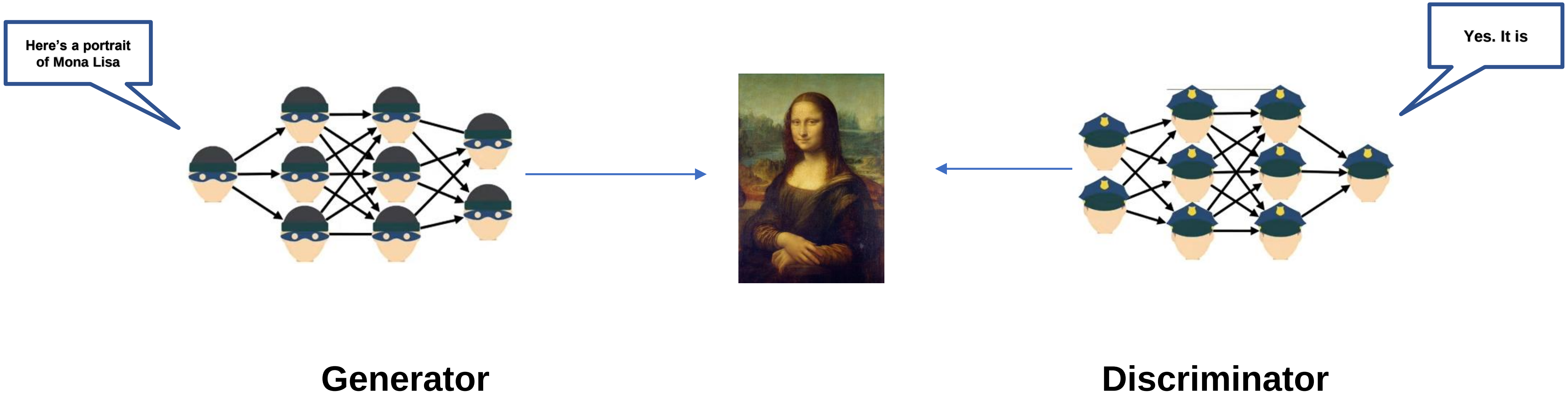
- This adversarial learning is a two-player game between the **generator** and the **discriminator**.



Generator **Discriminator**

Adversarial Learning

- This adversarial learning is a two-player game between the **generator** and the **discriminator**.



Steps taken by GAN:

- *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.*

$$\min_G \max_D V(D, G)$$

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

Steps taken by GAN:

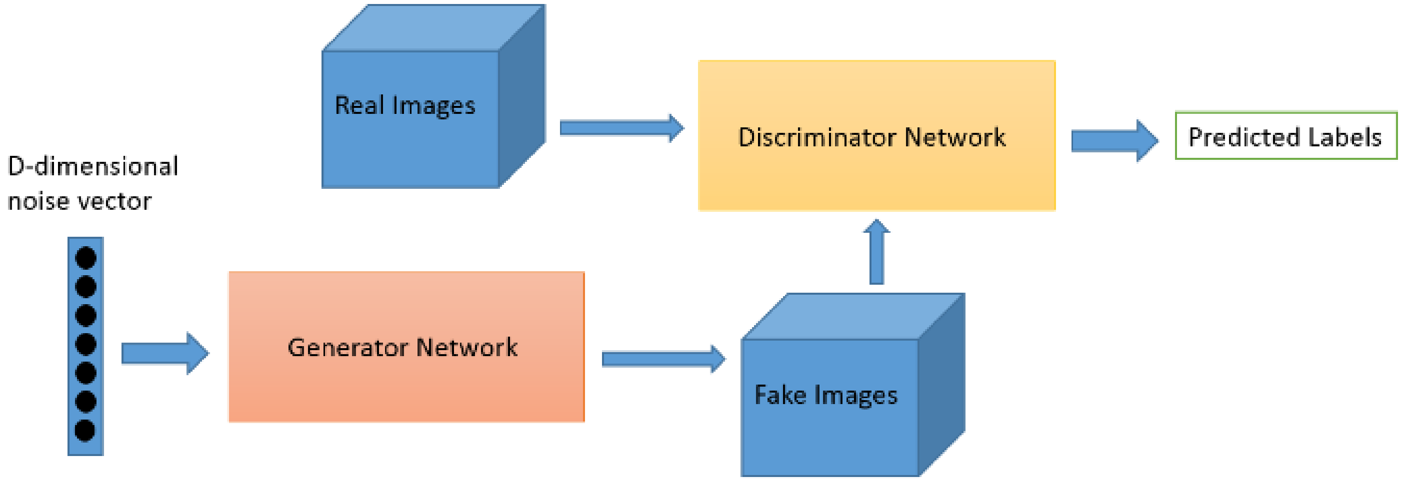


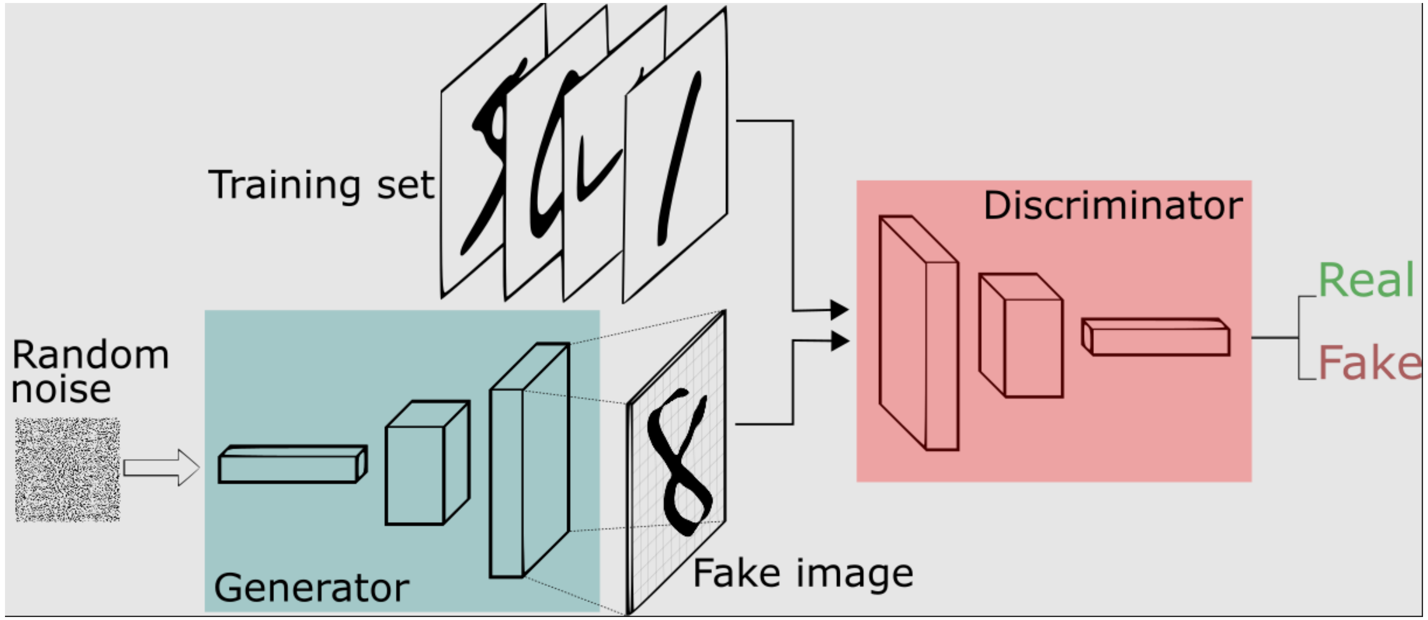
Image Credits: O'Reilly

Example: Generating MNIST digits

- For MNIST, the discriminator network is a standard convolutional network that can categorize the images fed to it, a binomial classifier labeling images as real or fake.

- The generator is an inverse convolutional network that takes a vector of random noise and upsamples it to an image

Example: Generating MNIST digits



How to train a GAN

- Build your generator and discriminator models
- Sample data from true distribution and train discriminator on this data
  - Discriminator works as a binary classifier
- Create input data from latent space (noise) and feed it to the generator
- The generator output serves as input to the discriminator
- The discriminator uses Sigmoid activation to classify the generated output as real or fake
- The discriminator loss is fed-back to the generator
- The generator is trained until it reaches convergence

Challenges

- GANs are really hard to train!
- Hard to meet convergence criteria.
- Vanilla GANs suffer from vanishing gradient problem.
- Overwhelming adversary i.e. one network should not "*overpower*" another
- Cannot count objects *(Stable Diffusion can help)*
- Sensitive to choice of hyperparameters
- *...and many other challenges*

Guidelines for training

- Use ***tanh*** activation for **generator** and ***sigmoid*** for **discriminator**
- Freeze **generator** for every other training iteration of **discriminator**
- Use ***LeakyReLU*** in dense layers
- Use Functional API (for Keras)
- Use dropout

Types of GANs

- Vanilla GANs (original model)
- CGAN
- Wasserstein GAN
- StyleGAN
- InfoGAN
- DCGAN
- DualGAN
- KDGAN
- *... many other variants*

Which architecture to use for RecSys?

- Vanilla GANs (original model)
- **CGAN**
- Wasserstein GAN
- **StyleGAN**
- InfoGAN
- DCGAN
- DualGAN
- KDGAN
- *... many other variants*

Which architecture to use for RecSys?

- Vanilla GANs (original model)
- **CGAN**
- Wasserstein GAN
- **StyleGAN**
- InfoGAN
- DCGAN
- DualGAN
- KDGAN
- *... many other variants*