# An Overview of Generative Adversarial Networks (GANs)

# Introduction

- Generative Adversarial Networks were originally introduced by Goodfellow et al. in 2014.
- GANs were introduced as a new framework of generative models that use an adversarial process for training:
  - A **generative** model **G** captures the distribution of the data,
  - while a **discriminative** model **D** estimates whether the sample comes from the training data rather than **G**.

https://arxiv.org/abs/1406.2661.

# Introduction

- In other word, GAN are a particular neural network that **take random noise as input and generate picture as outputs**.

https://arxiv.org/abs/1406.2661.

# Introduction

- In other word, GAN are a particular neural network that **take random noise as input and generate picture as outputs**.
- These picture are **sampled** from the **training set distribution**

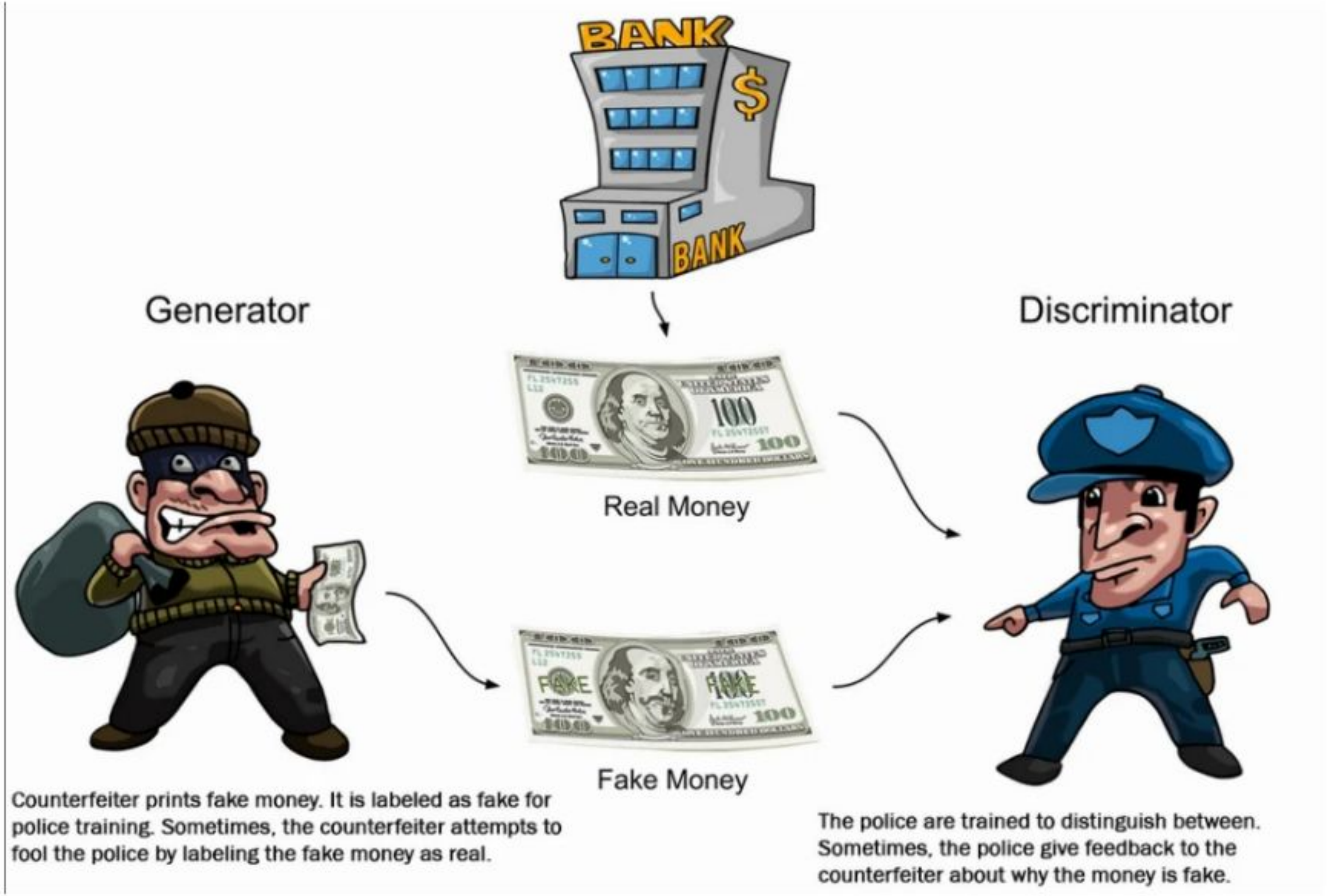
https://arxiv.org/abs/1406.2661.

# GANs Architecture

- A GAN structure consist in **two different** (and with different **scope**) **models** which are trained simultaneously and in an adversarial way:
    - A **Generative** model (also called **Generator**) that captures the training set distribution
    - A **Discriminative** model (also called **Discriminator**) that estimates the probability of a sample to be a **true** or **false** image (e.g., it comes from training data or is generated by another model).

# GANs Architecture

- The simplest explanation about GANs' training is to think at Generator as a *banknotes' counterfeiter* and at Discriminator as a *policeman specialized in banknote recognition*.


BANK
BANK
Generator
Discriminator
Real Money
Fake Money
FAKE
FAKE
Counterfeiter prints fake money. It is labeled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labeling the fake money as real.
The police are trained to distinguish between. Sometimes, the police give feedback to the counterfeiter about why the money is fake.

# GANs Architecture

- The simplest explanation about GANs' training is to think at Generator as a *banknotes' counterfeiter* and at Discriminator as a *policeman specialized in banknote recognition.*

- The policeman is taught **how to determine whether a dollar bill is genuine or counterfeit.**

# GANs Architecture

- The simplest explanation about GANs' training is to think at Generator as a *banknotes' counterfeiter* and at Discriminator as a *policeman specialized in banknote recognition*.

- The policeman is taught **how to determine whether a dollar bill is genuine or counterfeit.**
    - Samples of real dollar bills from the bank and fake money from the counterfeiter are used to train the policeman.

# GANs Architecture

- The simplest explanation about GANs' training is to think at Generator as a *banknotes' counterfeiter* and at Discriminator as a *policeman specialized in banknote recognition*.

- However, from time to time, the counterfeiter will attempt to pretend that he printed real dollar bills.

# GANs Architecture

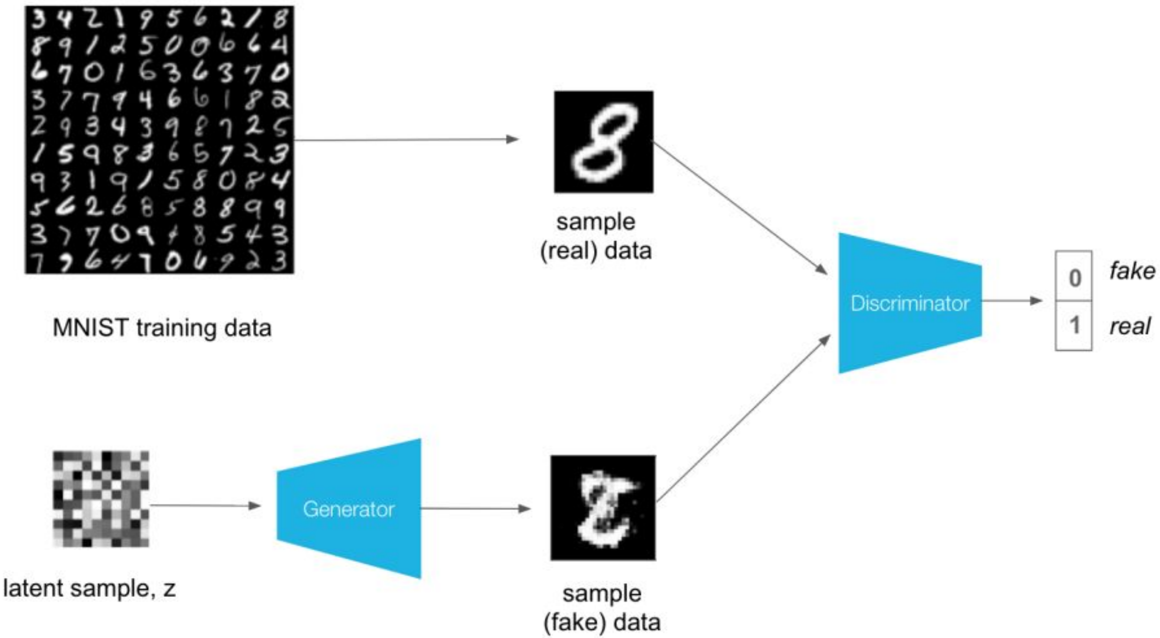
- The simplest explanation about GANs' training is to think at Generator as a *banknotes' counterfeiter* and at Discriminator as a *policeman specialized in banknote recognition*.

- However, from time to time, the counterfeiter will attempt to pretend that he printed real dollar bills.
- Initially, the police will not be fooled and will tell the counterfeiter **why the money is fake**. Taking into consideration this feedback, the counterfeiter hones his skills again and attempts to produce new fake dollar bills.

# GANs Architecture

- In other words, the **Generator is try**ing to learn the real data distribution (**by taking random noise as input**, and producing realistic-looking images) and is the network which we're usually interested in.
- On the other hand, the **Discriminator tries to classify whether the sample has come from the real dataset, or is fake** (generated by the generator).

# GANs Architecture



# Generator

- **Generator** is a Neural Network, which given a dataset $\mathbf{X}_{real}$ tries to capture his **distribution**, by producing images $\mathbf{X}_{fake}$ from **noise Z as input**.

# Generator

- **Generator** is a Neural Network, which given a dataset $\mathbf{X}_{\mathbf{real}}$ tries to capture his **distribution**, by producing images $\mathbf{X}_{\mathbf{fake}}$ from **noise Z as input**.
- **Noise** input is a **random** set of **values**, usually sampled from a **multivariate-gaussian distribution**.
    - This is often called **latent vector** and that vector space is called **latent space**.

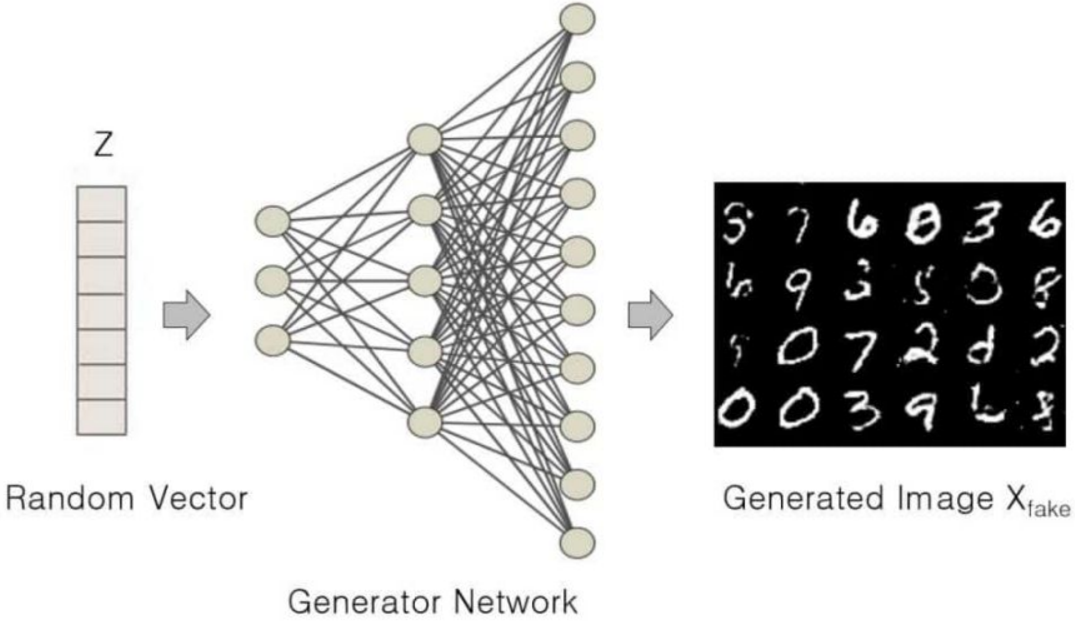
# Generator

- **Generator** is a Neural Network, which given a dataset $X_{real}$ tries to capture his **distribution**, by producing images $X_{fake}$ from **noise Z as input**.
- **Noise** input is a **random** set of **values**, usually sampled from a **multivariate-gaussian distribution**.
    - This is often called **latent vector** and that vector space is called **latent space**.
- The GAN's Generator acts quite like the decoder component of **VAE**: it **project latent space to an image**.
- The difference is that the Generator's latent space is not forced to learn exactly a Gaussian distribution but it can learn and model more complex distributions.

# Generator

- A problem that can bring GANs to poor performance is **Mode collapse**:
    - which happens when the **Generator can only produce a single type of output or a small set of outputs.**
- This may happen due to problems in training, such as the generator **finds a type of data that is easily able to fool the Discriminator and thus keeps generating that one type.**

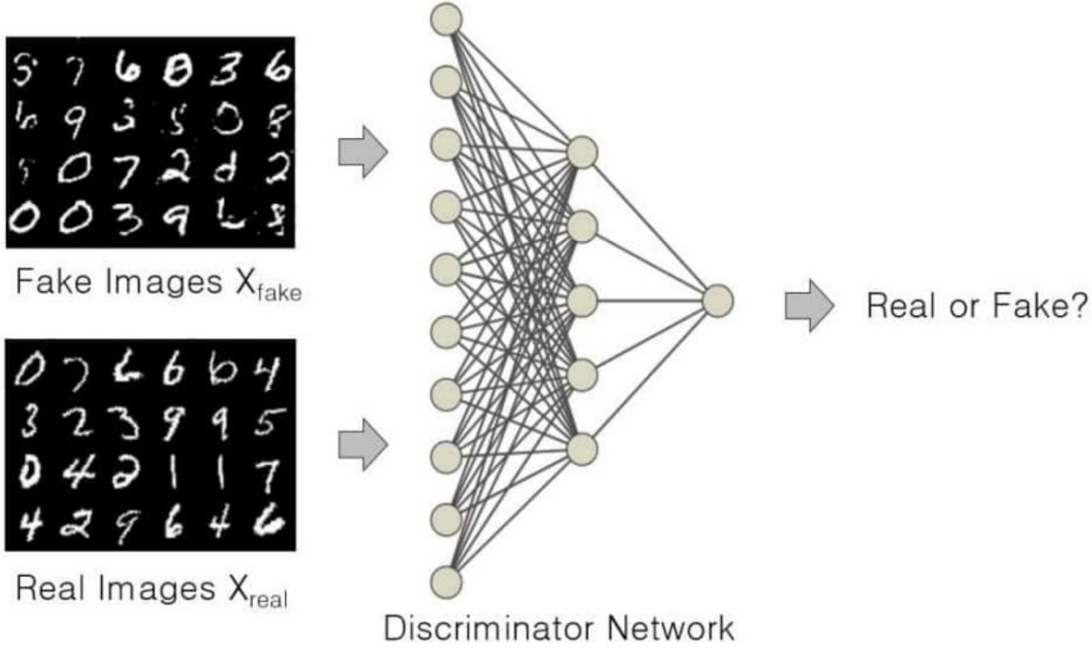
# Generator


Z
Random Vector
Generator Network
5 7 6 8 3 6
6 9 3 5 0 8
5 0 7 2 d 2
0 0 3 9 L 8
Generated Image $X_{fake}$

# Discriminator

- The Discriminator is like a model trained to classify if an input image is real or is generated by another model. In other words the Discriminator is a binary classifier trained with a supervisioned classification mode.
  - The Discriminator task is to predict a label (e.g. True or Fake or 0-1 label) from an input.
- While doing so, the supervisioned training gives to it a feedback and this permit the model to learn a set of parameters (weights and bias) in order to map the correct label.
- The main scope of the Discriminator is to give a feedback to the Generator model, and to improve its generation power.

# Discriminator



# Training Phase

Before start the explanation,

let’s denote some of the key elements:

# Training Phase

Before start the explanation,

let's denote some of the **key elements:**

- $X \rightarrow$ Training samples
- $Z \rightarrow$ Noise or Latent Vector
- $D \rightarrow$ Discriminator
- $G \rightarrow$ Generator
- $X_{real} \rightarrow$ a sample from training dataset
- $X_{fake}$ or $G(z) \rightarrow$ Generator's output
- $D(x) \in (0, 1) \rightarrow$ Discriminator's output

# Training Phase

Training of the two models (D and G) takes place alternately:

1. **Step 1**:

    - The Generator take $Z$ as input and produce $G(x)$ ($X_{fake}$).
    - $X_{fake}$ and a sample $X_{real}$ are passed to the Discriminator.
    - The Discriminator gives $D(x)$ as output. This tells the probability score of each input passed to $D$ to be real or fake.
    - As a normal supervised training, the predictions are compared with the ground truth, and a **Loss** is calculate (e.g. Binary Cross-Entropy).
    - The Gradient is the backpropagated only through the Discriminator and is parameters are updated.

# Training Phase

Training of the two models (D and G) takes place alternately:

2. **Step 2**:

    - The Generator produce an image $G(x)$, that is again passed through the Discriminator.
    - The Discriminator gives a prediction $D(x)$ as before, but only for image passed by the Generator.
    - The Loss is computed as before.
    - In this step, because we want to enforce the Generator to produce images, as similar to the real images as possible (i.e. close to the true distribution), the Loss is backpropagated only through the Generator network, and its parameters are optimized suitably.

# Training Phase

As described, both the **Generator and the Discriminator's training is based on the classification score given as output by the last layer of D model.** The output score is based on a binary classification problem, so the **Binary Cross-Entropy function is used to compute the Loss**:

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y_i} + (1 - y_i)\log(1 - \hat{y_i})]$$

# Training Phase

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y_i} + (1 - y_i)\log(1 - \hat{y_i})]$$

- The negative sign at the beginning of the equation is there because the output of a neural network is normalized between 0 and 1, compute the *log* of a number in that range result in a value less than zero.
- The $\frac{1}{N}$ term identify the Loss' means computed across the batch ($N$ refer to batch size).
- $\hat{y_i}$ is the Discriminator's prediction.
- $y_i$ is the true label.
- The equation's first term is valid when the true label is 1 (real), and the second is valid when the true label is 0 (fake).

# Training Phase

We can describe GAN's training phase as a **minmax** game played by Generator and Discriminator. Is possible to explain it with the following value function $V(G, D)$ [4]:

$$\min_G max_D V(D, G) = E_{x \sim p_{data}(x)}[log D(x)] + E_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

# Some GAN’s variants

- After Goodfellow’s article on GANs, many researchers have become interested in the Generative Adversarial Networks.
- Many variants of the original GAN architecture have been proposed and studied.
    - **Deep Convolutional GAN (DCGAN)**
    - **Conditional GAN (cGAN)**

# Deep Convolutional GAN (DCGAN)

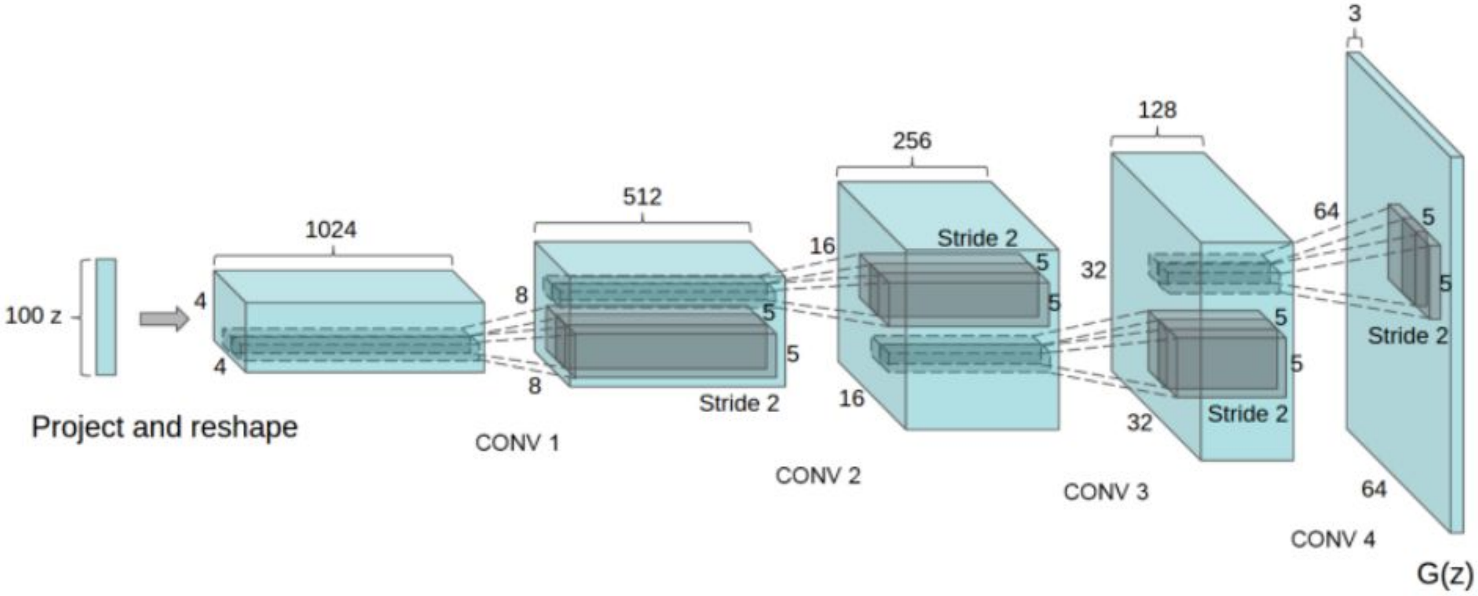
- **Deep Convolutional Generative Adversarial Network**, also know as **DCGAN** is a particular and new GAN architecture that **improve his quality using convolutional layers**.
- It was introduced around **2016** by Alec Radford and other researcher in a paper published at ICLR conference.

https://arxiv.org/abs/1511.06434

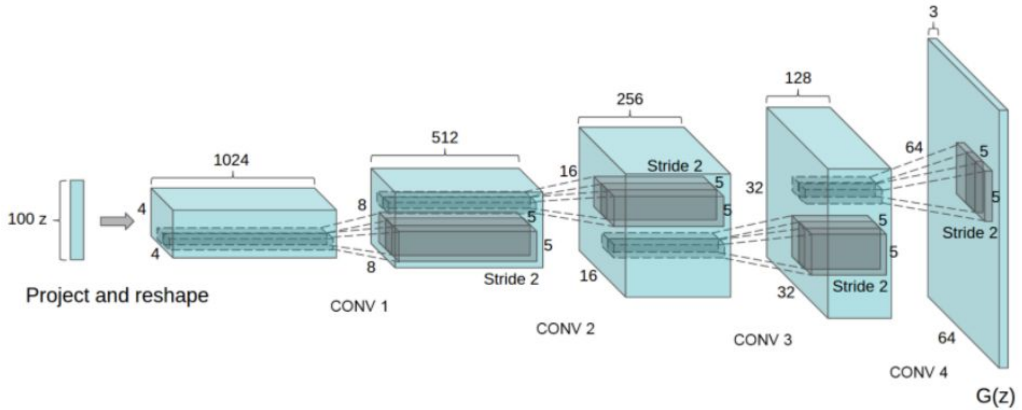
# Deep Convolutional GAN (DCGAN)

- Deep Convolutional Generative Adversarial Network, also know as DCGAN is a particular and new GAN architecture that improve his quality using convolutional layers.
- The main components introduced in DCGAN are:
  - **fractionally-strided convolutional layers in the Generator**, with the scope of **upsample** the images (see Figure 2.1).
  - **strided convolutional layers in the Discriminator**, with the scope of **downsample** the images.

# Deep Convolutional GAN (DCGAN)



# Deep Convolutional GAN (DCGAN)




A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps.

A series of four fractionally-strided convolutions then convert this high level representation into a 64 × 64 pixel image.

No fully connected or pooling layers are used.

# Deep Convolutional GAN (DCGAN)

- In DCGAN the authors used a **Stride of 2**, meaning the filter slides thorough the image moving 2 pixels per step.
- The use of **strided convolutional layer replaced** the use of **pooling** operation like **maxPooling** or **avgPooling** because these operation have not learnable parameter.
- The reason why they used a strided convolutional layer is because **in this manner the network** (the Discriminator in that case) **is allowed to learn its own spatial downsampling**

# Deep Convolutional GAN (DCGAN)

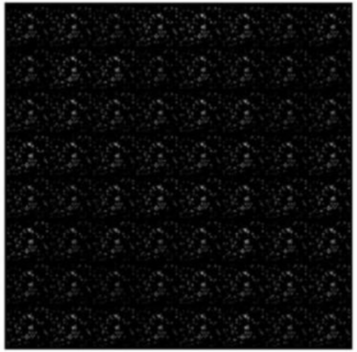
- The **Fractionally-strided convolution operation is essentially the opposite of a convolution operations**.
- In a traditional convolution operation there is a downsampling of the input dimension in the output.
    - Instead, in a fractionally strided convolution operation, is produced **a larger output from the input.**
- The power of this upsampling operation is that **fractionally-strided convolution has learnable parameters**.
    - This parameters can be trained in order to follow the objective function and maximize the performance.

# Conditional GAN (cGAN)

- Traditional GAN is trained in a completely unsupervised and unconditional fashion, meaning **no labels involved in the training process**.
- Though the GAN model is capable of generating new realistic samples for a particular dataset, **we have zero control over the type of images that are generated**.

# Conditional GAN (cGAN)

- Traditional GAN is trained in a completely unsupervised and unconditional fashion, meaning **no labels involved in the training process**.
- Though the GAN model is capable of generating new realistic samples for a particular dataset, **we have zero control over the type of images that are generated**.



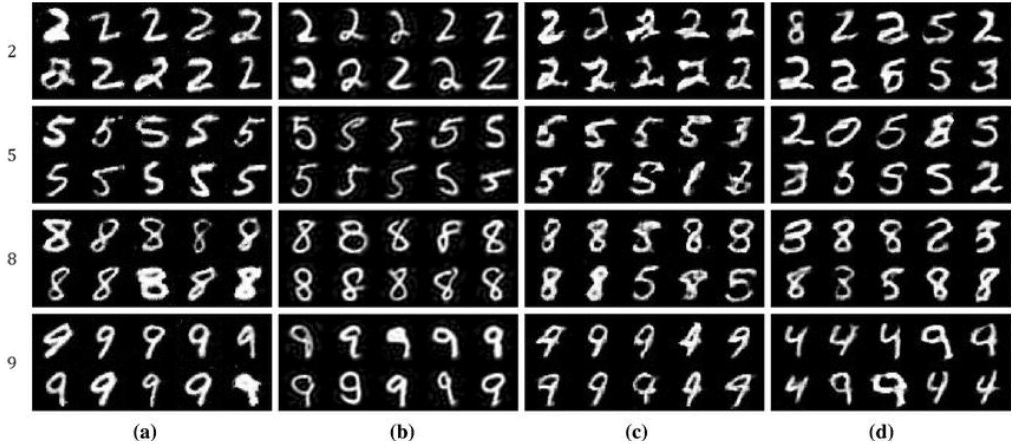
Epoch 1

Epoch 100

Epoch 200

# Conditional GAN (cGAN)

- Traditional GAN is trained in a completely unsupervised and unconditional fashion, meaning **no labels involved in the training process**.
- Though the GAN model is capable of generating new realistic samples for a particular dataset, **we have zero control over the type of images that are generated**.



# Conditional GAN (cGAN)

- Traditional GAN is trained in a completely unsupervised and unconditional fashion, meaning **no labels involved in the training process**.
- Though the GAN model is capable of generating new realistic samples for a particular dataset, **we have zero control over the type of images that are generated**.
- To understand the cGANs mechanism, think about a traditional GAN trained to generate images that match a dataset with handwritten number (e.g. MNIST dataset).
  - With a GAN you pass a noise vector as input and you get an images as output.
  - You don't have power on the number that GAN will output to you (e.g. it may be a 9 or it may be a 1).

# Conditional GAN (cGAN)

- Of course you can have explore the latent space and **find the area of the distribution which you can sample a latent vector** that will give you a specific number as output, but to obtain this you have to do a type of **brute-force** until you find the area which corresponds to a certain number.

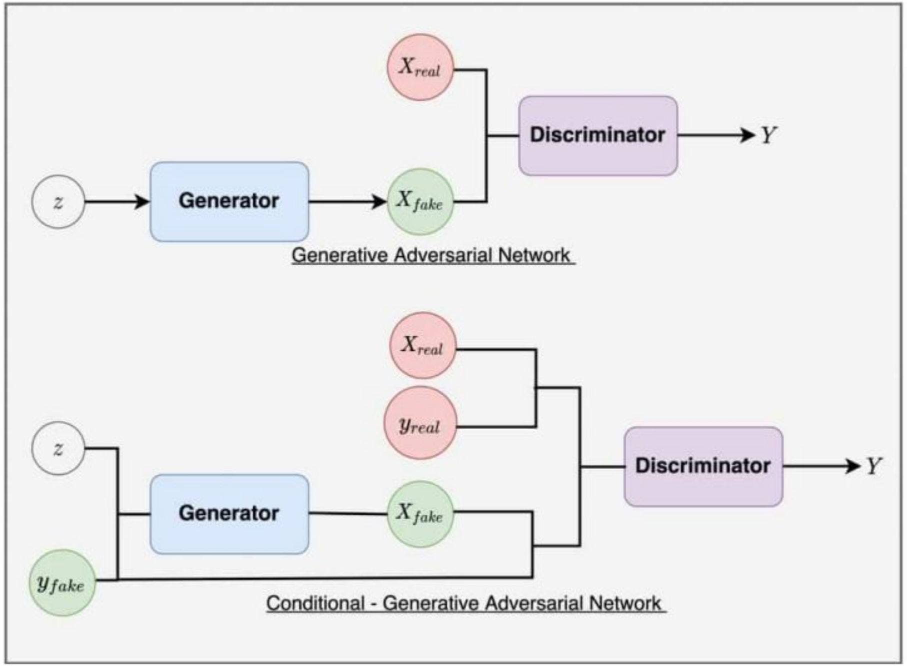
# Conditional GAN (cGAN)

- Of course you can have explore the latent space and **find the area of the distribution which you can sample a latent vector** that will give you a specific number as output, but to obtain this you have to do a type of **brute-force** until you find the area which corresponds to a certain number.
- This problem occurs with any dataset you used to train a GAN.

# Conditional GAN (cGAN)

- Of course you can have explore the latent space and **find the area of the distribution which you can sample a latent vector** that will give you a specific number as output, but to obtain this you have to do a type of **brute-force** until you find the area which corresponds to a certain number.
- This problem occurs with any dataset you used to train a GAN.
- The cGAN is a solution to this by giving some type of control on generation phase.

# Conditional GAN (cGAN)



# Conclusion

There are a lot of applications where GANs are useful:

- **Data Augmentation**: in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.
    - It acts as a regularizer and helps reduce overfitting when training a machine learning model.
    - It is closely related to oversampling in data analysis.
- Generate **high-resolution** versions of input images.
- Creating images, sketches, painting, and more.
- **Image-to-Image Translation**: The ability to translate photographs across domains, such as day to night, summer to winter, and more.

# Conclusion

Drawbacks of GANs:

- **Training Instability**: GANs can be challenging to train due to their competitive nature, with the risk of training instability and convergence to low-quality solutions.
- **Computational Complexity**: Training GANs can require significant computational resources, including time and computing power, especially for large-scale models.