**GANs: Generative Adversarial Networks — An Advanced Solution for Data Generation**

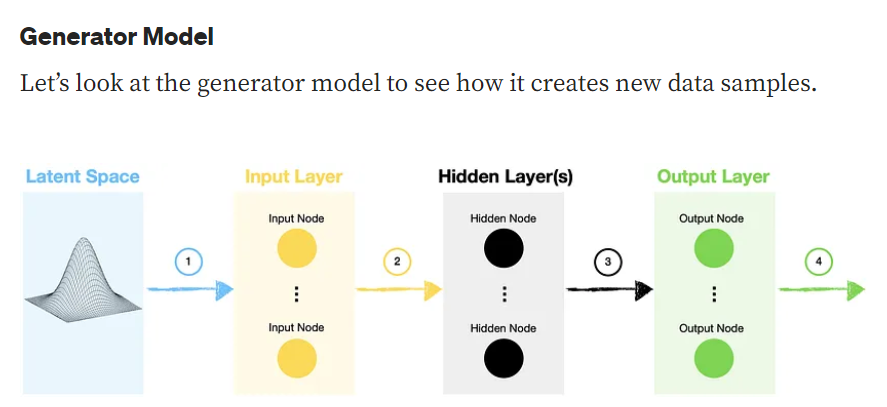
# Intro

There has been so much hype over Generative Adversarial Networks (GANs) in the Data Science community. But, as you start learning about them, you immediately see why. GAN architecture is a genius setup that has unlocked the potential for realistic data generation and augmentation.

# An intuitive explanation of GAN architecture and how it works

Generative Adversarial Networks are deep learning machines that combine two separate models into one architecture. The two components are:

* Generator Model
* Discriminator Model
* The two models compete against each other in a zero-sum game. The generator model tries to generate new data samples similar to those in the problem domain. Meanwhile, the discriminator tries to identify whether the example presented is fake (comes from a generator ) or real (comes from the actual data domain).
* The competition between the generator and the discriminator makes them adversaries, which gives the name to GANs.



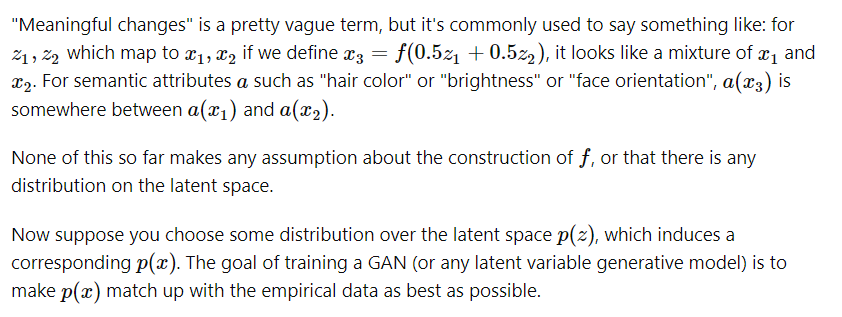
1. The generator model samples a random vector from the latent space. This space follows Gaussian distribution with the number of dimensions specified by us. The random vector seeds the generative process since we use it as input to our neural network.
2. The inputs follow a standard path through the network with one or multiple hidden layers. In the case of a simple GAN, this would be a bunch of densely connected layers, while Deep Convolutional GAN (DCGAN) would also incorporate convolutional layers.
3. The data flows into an output layer where we can make final adjustments to ensure that the generator output has the required shape to feed into a discriminator.
4. Finally, we can use these fake (generated) samples to try and fool the discriminator.

Latent space refers to an abstract multi-dimensional space containing feature values that we cannot interpret directly, but which encodes a meaningful internal representation of externally observed events.

The latent space itself has no meaning. Typically, it is a 100-dimensional hypersphere with each variable drawn from a Gaussian distribution.

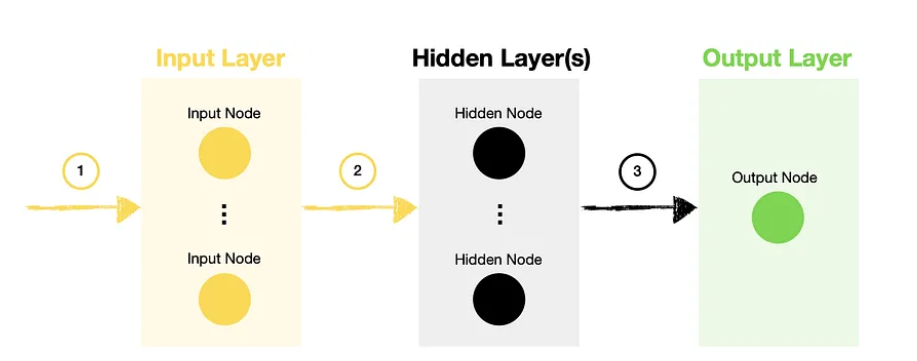
**1**

We say the latent state is a "meaningful internal representation" because manipulations and transformations to a latent vector z result in meaningful changes in the observed output x=f(z).



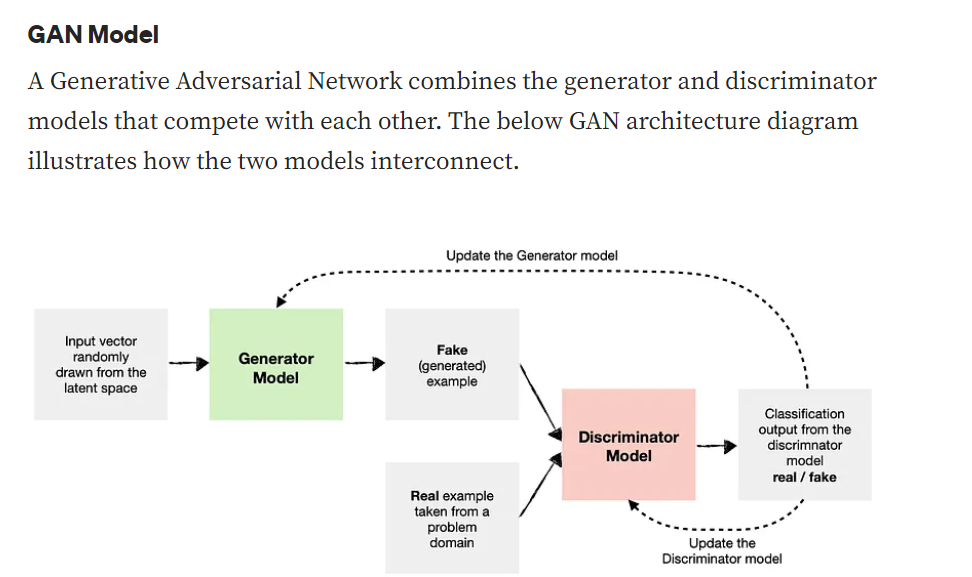
## **Discriminator Model**

Next, let’s see how the discriminator model is constructed.



1. The inputs to the discriminator model are a combination of real samples (drawn from the problem domain) and fake ones (created by the generator model).
2. The data goes through the network with one or multiple hidden layers, the same as what you would have in any other neural network.
3. Once we reach the output layer, the discriminator decides whether the sample is real or fake (generated).

In summary, the discriminator is no different from a standard neural network classification model.



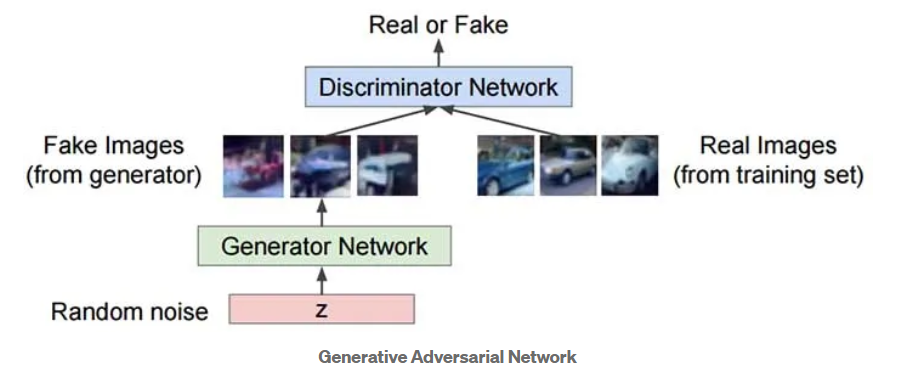
As you can see in the diagram, we feed fake (generated) and real examples to the discriminator model, training it to differentiate between the two classes.

As the discriminator becomes better at distinguishing between real and fake examples, the weights and biases of the generator model get updated to make it produce more convincing fakes.

The process continues for the specified number of epochs with the generator and discriminator trying to become better at their specific task. Finally, at the limit, outputs from the generator model become indistinguishable from the real ones, with the discriminator model converging towards a neutral prediction of 0.5.

The generator starts with a random values from the latent space. This space follows Gaussian distribution with the number of dimensions specified by us. While we train the generator, it changes the random selected values from the latent space, choosing better values that will create a vector as input that creates a better image with the intention of creating fake images more similar to the real ones through each epoch iteration.

GANs consists of two networks, a Generator **G(x)**, and a Discriminator **D(x)**. They both play an adversarial game where the generator tries to fool the discriminator by generating data similar to those in the training set. The Discriminator tries not to be fooled by identifying fake data from real data. They both work simultaneously to learn and train complex data like audio, video or image files.



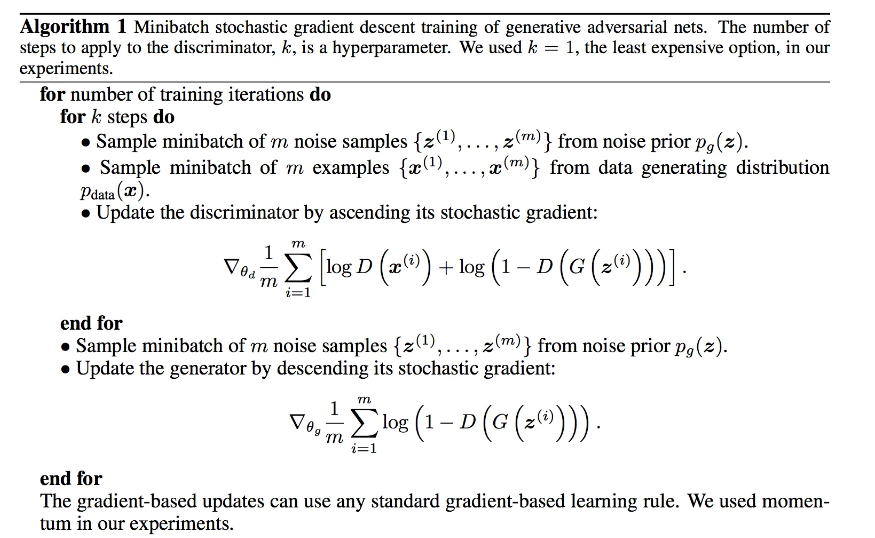
The generator model generates images from random noise(z) and then learns how to generate realistic images. Random noise which is input is sampled using uniform or normal distribution and then it is fed into the generator which generates an image. The generator output which are fake images and the real images from the training set is fed into the discriminator that learns how to differentiate fake images from real images. The output D(x) is the probability that the input is real. If the input is real, D(x) would be 1 and if it is generated, D(x) should be 0. So, the output of the generator is between 0 and 1.

D() gives us the probability that the given sample is from training data X. For the Generator, we want to minimize log(1-D(G(z)) i.e. when the value of D(G(z)) is high then D will assume that G(z) is nothing but X and this makes 1-D(G(z)) very low and we want to minimize it which this even lower. For the Discriminator, we want to maximize D(X) and (1-D(G(z))).

So the optimal state of D will be P(x)=0.5. **However, we want to train the generator G such that it will produce the results for the discriminator D so that D won’t be able to distinguish between z and X. Create Fake images as similar as the real ones.**

Now the question is why this is a minimax function. Here, the Discriminator tries to maximize the objective which is V while the Generator tries to minimize it, due to this minimizing/maximizing we get the minimax term. They both learn together by alternating gradient descent.

We will perform the iteration of gradient descent on D using real and generated images by fixing G. Then we fix D and train G for another single iteration to fool a fixed D. **We want to optimize our minimax function by iterating both G and D in alternating steps until we get good quality images from the generator and discriminator won’t be able to differentiate between real and fake images, Discriminator probability will be P(x)=0.5.**

****

Below is the pseudo-code which shows how GANs are trained.