GENERATIVE ADVERSARIAL NETWORKS(GAN)

- A generative adversarial network, or GAN, is a deep neural network framework which is able to learn from a set of training data and generate new data with the same characteristics as the training data.

- Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning.

- It was developed and introduced by Ian J. Goodfellow in 2014.

- GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset.

- For example, a generative adversarial network trained on photographs of human faces can generate realistic-looking faces which are entirely fictitious. (even though the faces don't belong to any real person.)

- ## Generative Adversarial Network Architecture

- Generative adversarial networks consist of **two neural networks**,

the **generator and the discriminator,** which compete against each other.

  - The generator is trained to produce fake data, and

  - the discriminator is trained to distinguish the generator's fake data from real examples.

  - If the generator produces fake data that the discriminator can easily recognize as implausible, such as an image that is clearly

not a face, the generator is penalized. Over time, the generator learns to generate more plausible examples.

→GANs achieve this level of realism by pairing a generator, which 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.

Why were GANs developed?

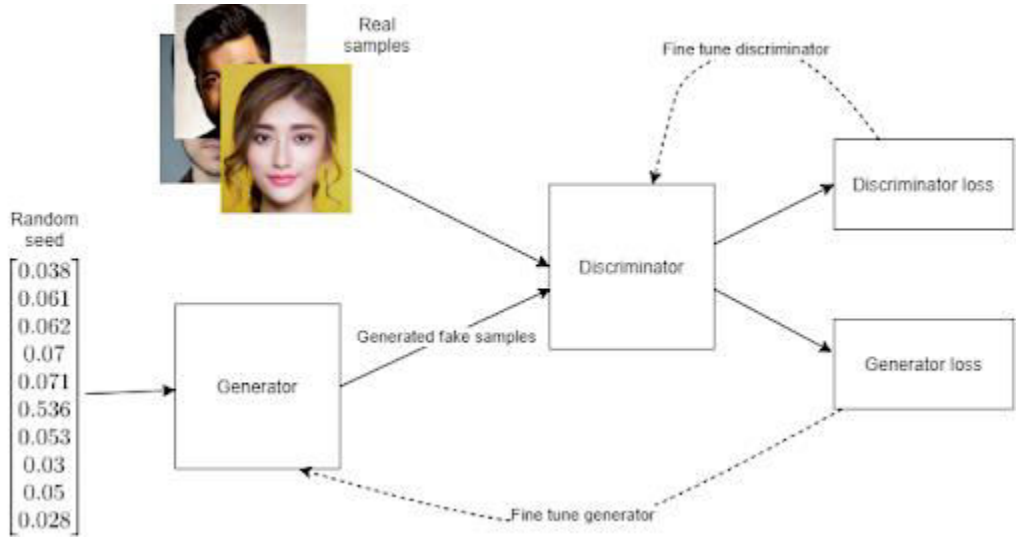
- The reason for such adversary is that most machine learning models learn from a limited amount of data, which is a huge drawback, as it is prone to overfitting.
- Also, the mapping between the input and the output is almost linear.
- Although, it may seem that the boundaries of separation between the various classes are linear, but in reality, they are composed of linearities and even a small change in a point in the feature space might lead to misclassification of data.

Generative Adversarial Networks (GANs) can be broken down into three parts:

- **Generative:** To learn a generative model, which describes how data is generated in terms of a probabilistic model.
- **Adversarial:** The training of a model is done in an adversarial setting. **(Characterized by conflict)**
- **Networks:** Use deep neural networks as the artificial intelligence (AI) algorithms for training purpose.

EXAMPLE:

The generator's fake examples, and the training set of real examples, are both fed randomly into the discriminator network. The discriminator does not know whether a particular input originated from the generator or from the training set.



The `seed()` method is used to initialize the random number generator.

- Initially, before training has begun, the generator's fake output is very easy for the discriminator to recognize.
- Since the output of the generator is fed directly into the discriminator as input, this means that when the discriminator classifies an output of the generator, we can apply the backpropagation algorithm through the whole system and update the generator's weights.
- Over time, the generator's output becomes more realistic and the generator gets better at fooling the discriminator. Eventually, the

generator's outputs are so realistic, that the discriminator is unable to distinguish them from the real examples.

The Discriminator in a Generative Adversarial Network

- The discriminator is simply a binary classifier, ending with a suitable function such as the softmax function.
- The discriminator outputs an array such as where the two numbers indicate the discriminator's estimate of the probability of the input example being real or fake.

- The discriminator's input may come from two sources:

1)The training set, such as real photos of faces, or real audio recordings.

2)The generator, such as generated synthetic faces, or fake audio recordings.

- While we are training the discriminator, we do not train the generator, but hold the generator's weights constant and use it to produce negative examples for the discriminator.

The process for training the discriminator in a GAN

- Pass some real examples, and some fake examples from the generator, into the discriminator as input.
- The discriminator classifies them into real and fake.
- Calculate the discriminator loss using a suitable function such as the cross-entropy loss.
- Update the discriminator's weights through backpropagation.

This process is the same as the process for training any other kind of binary classifier, such as a convolutional neural network in the case of computer vision.

The Generator in a Generative Adversarial Network

- The generator network is a feedforward neural network learns over time to produce plausible fake data, such as fake faces.
- It uses feedback from the discriminator to gradually improve its output, until ideally, the discriminator is unable to distinguish its output from real data.

The process of training the generator in a GAN

- At the start of training, we initialize both the generator and discriminator with random weights.
- For each training iteration, we pass a random seed into the generator as input. The random noise is propagated through the generator and outputs a synthetic example, such as an image.
- The generator output is then passed as input into the discriminator network, and the discriminator classifies the example as 'real' or 'fake'.

We calculate a loss function for the generator. The generator's loss function represents how good the generator was at tricking the discriminator.

- We use the backpropagation algorithm through both the discriminator and generator, to determine how to adjust the only generator's weights in order to improve the generator loss function.

→At this point we do not adjust the discriminator's weights, because the discriminator needs to stay static while we are training the generator. If we did not do this, training the generator would be like trying to hit a moving target.

How does training a generative adversarial network work?

There are two aspects that make generative adversarial networks more complex to train than a standard feedforward neural network:

- The generator and the discriminator are really two neural networks which must be trained separately, but they also interact so they cannot be trained completely independently of each other.

- It is hard to identify exactly when a generative adversarial network has converged.
- Since the generator and discriminator have their own separate loss functions, we have to train them separately.
- We can do this by alternating between the two:

→We train the discriminator for one or more epochs, keeping the generator weights constant.

→We train the generator for one or more epochs, keeping the discriminator weights constant.

We repeat steps (1) and (2) until we determine that the network has converged.

- **Part 1:** The Discriminator is trained while the Generator is idle. In this phase, the network is only forward propagated and no back-propagation is done. The Discriminator is trained on real data for n epochs, and see if it can correctly predict them as real. Also, in this phase, the Discriminator is also trained on the fake generated data from the Generator and see if it can correctly predict them as fake.

- **Part 2:** The Generator is trained while the Discriminator is idle. After the Discriminator is trained by the generated fake data of the Generator, we can get its predictions and use the results for training

the Generator and get better from the previous state to try and fool the Discriminator.
The above method is repeated for a few epochs and then manually check the fake data if it seems genuine. If it seems acceptable, then the training is stopped, otherwise, its allowed to continue for few more epochs.

Convergence in a Generative Adversarial Network

- Once the generator is able to produce fakes that are indistinguishable from real examples, the discriminator has a much more difficult task.
- In fact, for a perfect generator, the discriminator will have only 50% accuracy in distinguishing fakes from genuine examples.
- This means that the discriminator feedback, which we are using to train the generator, becomes less meaningful over time, and eventually becomes completely random, like a coin toss.
- If we continue to train the network beyond this point, then the discriminator's feedback can actually cause the generator's quality to go down. For this reason, it is important to monitor the quality of the generated output and stop training once the discriminator has 'lost' the game to the generator.

Loss Function of a Generative Adversarial Network

- The loss function used by Ian Goodfellow and his colleagues in their 2014 paper that introduced generative adversarial networks is as follows:

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

Generative adversarial network loss function

- The generator tries to minimize the output of the above loss function and the discriminator tries to maximize it.

- This way a single loss function can be used for both the generator and discriminator.

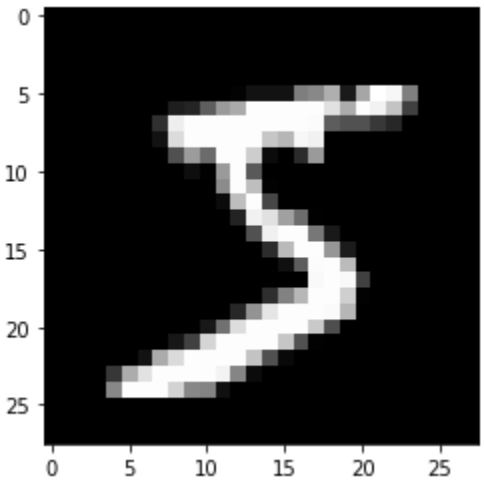
Loss Function Symbols Explained

$D(z)$	The estimate by the discriminator of the probability that an input example x is real
E_x	The expected value over all genuine examples
$G(z)$	The fake example produced by the generator for random seed z
$D(G(z))$	The estimate by the discriminator of the probability that a fake input example, G(z), from the generator is real.
E_z	The expected value over all random inputs to the generator.

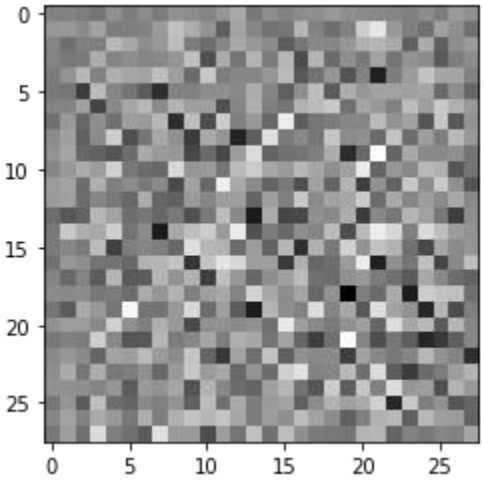
The generator is only able to minimize the second term in the loss function, since the first term depends only on the discriminator.

Example of Training a Generative Adversarial Network

- Let us take the example of training a generative adversarial network to synthesize handwritten digits.
- Below is a sample handwritten number 5 from the MNIST dataset.
- The MNIST dataset is a database of 60,000 images of handwritten digits 0 to 9, with dimensions 28x28 pixels.

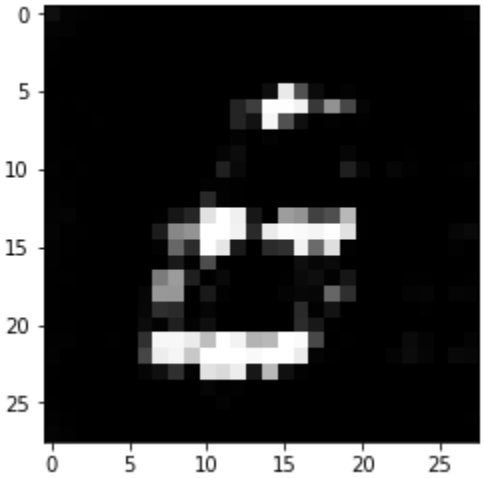


- When we initialize the generative adversarial network, initially the images produced by the generator will be pure noise, like this:

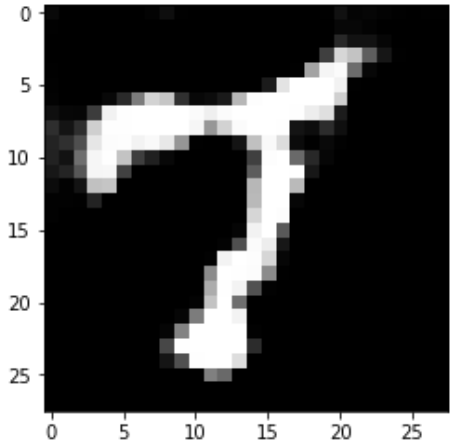


- Because this noise is very different from a handwritten digit, the discriminator immediately learns to tell the generated and fake data apart.

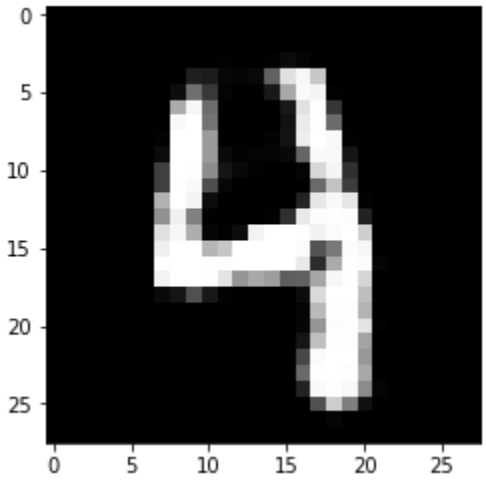
- The generator then begins to learn how to fool the discriminator. After four epochs (passing the whole MNIST dataset through the generative adversarial network four times, which takes a minute or so on a GPU), the generator starts producing random images that begin to resemble numbers. The discriminator's task is getting trickier.

- 

After a further 20 epochs, the generator's output begins to look recognizable:



- Below is the generator's output at 45 epochs.
- We can see that it is hard even for a human to recognize that this image is artificial, and at this point the discriminator's ability to recognize the fake examples has dropped to zero.



Applications of Generative Adversarial Networks

1)Generative Adversarial Networks for Synthetic training data

- Generative adversarial networks can be used to generate synthetic training data for machine learning applications where training data is scarce. It is often time consuming and costly to gather training data for many machine learning applications, so using a generative adversarial network to generate random faces is sometimes an attractive alternative.

Three synthetic faces generated by the generative adversarial network StyleGAN, developed by NVIDIA. These are not real people. StyleGAN was trained on the Flickr-Faces-HQ faces dataset.

2)Generative Adversarial Networks for Image Style Transfer

- Generating random images resembling a training dataset, generative adversarial networks can be used for style transfer.
- In 2019 a team at NVIDIA led by Tero Karras published a generative adversarial network architecture called StyleGAN which can be used to transform images from one style to another.
- This network can be used to morph a human face from one gender to another, or change facial orientation. It is also capable of replacing all horses in a photograph with zebras, for example, or turning a painting into the style of Monet.

3)Generative Adversarial Networks for Audio Style Transfer

- It is even possible to apply generative adversarial networks to audio data.
- With this technique, it is possible to morph audio from one speaker's voice to another, or to 'transfer' a piece of music from classical into a jazz style.

Different types of GANs:

GANs are now a very active topic of research and there have been many different types of GAN implementation. Some of the important ones that are actively being used currently are described below:

1. **Vanilla GAN:** This is the simplest type GAN. Here, the Generator and the Discriminator are simple multi-layer perceptrons. In vanilla GAN, the algorithm is really simple, it tries to optimize the mathematical equation using stochastic gradient descent.

2. **Conditional GAN (CGAN):** CGAN can be described as a deep learning method in which some conditional parameters are put into

place. In CGAN, an additional parameter ‘y’ is added to the Generator for generating the corresponding data. Labels are also put into the input to the Discriminator in order for the Discriminator to help distinguish the real data from the fake generated data.

3. **Deep Convolutional GAN (DCGAN):** DCGAN is one of the most popular also the most successful implementation of GAN. It is composed of ConvNets in place of multi-layer perceptrons. The ConvNets are implemented without max pooling, which is in fact replaced by convolutional stride. Also, the layers are not fully connected.

4. **Laplacian Pyramid GAN (LAPGAN):** The Laplacian pyramid is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual. This approach uses multiple numbers of Generator and Discriminator networks and different levels of the Laplacian Pyramid. This approach is mainly used because it produces very high-quality images. The image is down-sampled at first at each layer of the pyramid and then it is again up-scaled at each layer in a backward pass where the image acquires some noise from the Conditional GAN at these layers until it reaches its original size.

5. **Super Resolution GAN (SRGAN):** SRGAN as the name suggests is a way of designing a GAN in which a deep neural network is used along with an adversarial network in order to produce higher resolution images. This type of GAN is particularly useful in optimally up-scaling native low-resolution images to enhance its details minimizing errors while doing so.