Name: Leela Sai Surya Peela

Student Id: 23088635

Git repository:

https://github.com/peelaleelasaisurya20/Machine_Learning_GAN_Assignment.git

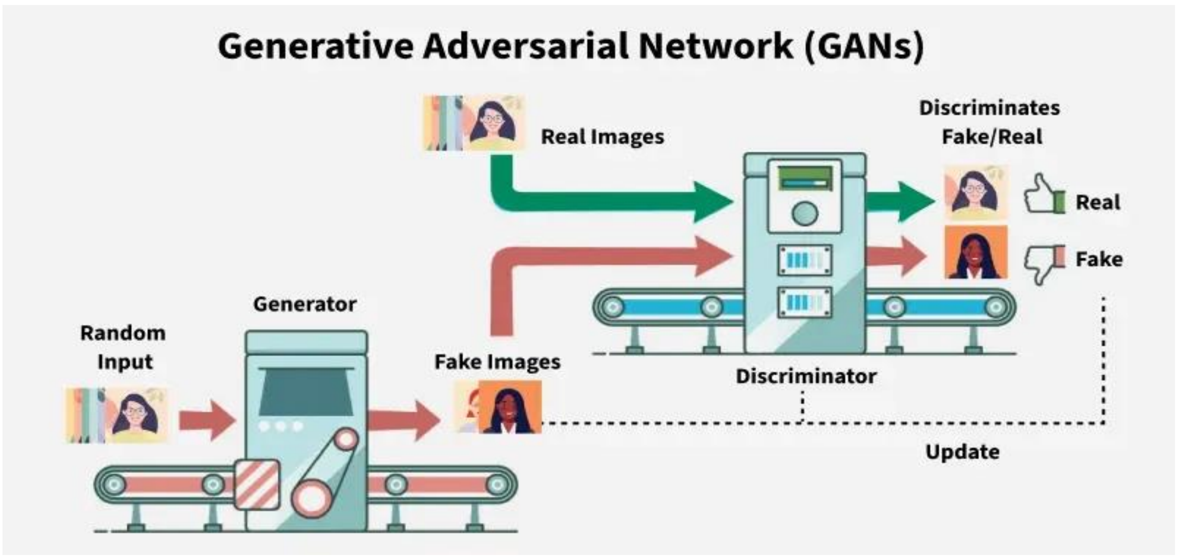
# Generative Adversarial Networks (GANs)

## Introduction

A class of machine learning models used for generating synthetic data that looks like real data are called Generative Adversarial Networks (GANs). Deep learning is a powerful application in image generation, style transfer and data augmentation. In 2014, Ian Goodfellow and his colleagues presented GANs as cornerstone in artificial intelligence research.

The two neural networks in GANs are competing against each other in the game theoretic sense. The latter evaluates the data generated by one network and generates increasingly realistic data as each iteration continues.

Generative adversarial network (GAN) is also a progress of artificial intelligence and forms very realistic artificial data. Generative adversarial networks (GANS) have emerged as an amazing tool for image generation that allows a machine to generate real images from random noise. In this project, we apply the Deep Convolutional GAN (DCGAN) with Tenseless and Cares Libraries to create images from the CIFAR -10 dataset. CIFAR-10 consists of small 32 × 32 color images in 10 classes, for example, animals and vehicles. Unlike traditional GNS, DCGNs employ conventional and transposed convolutional layers to reinforce image generation so that they can generate stable and high-quality images.

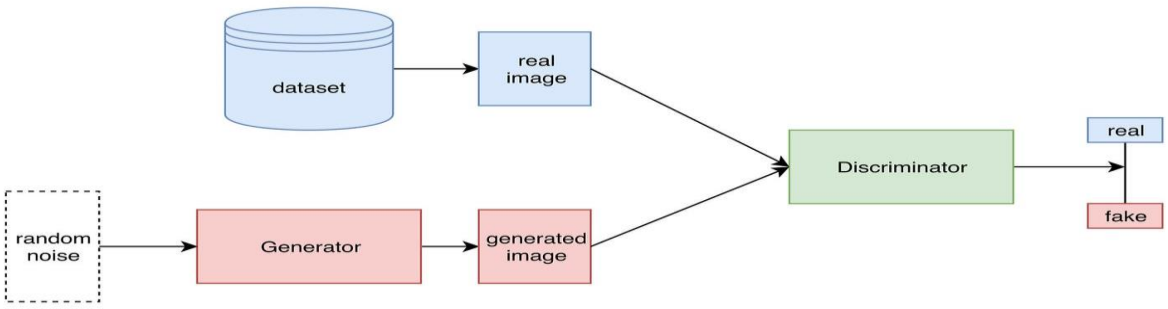


## Architecture:

Two artificial neural networks within GANs perform an adversarial mutual process. Two essential functional parts form a GAN architecture.

1. The generator (G) develops artificial information that reproduces authentic data patterns. The input begins with random noise that leads to an output which matches the appearance of actual data points.

2. The discriminator functions as an identification system between real data samples from actual sources and the generated fake outputs of the generator.



*Basic Architecture of Generative Adversarial Networks (GANs)*

This competition drives both networks to improve continuously - the generator produces increasingly realistic outputs while the discriminator becomes more sophisticated at detection. The training process is guided by carefully designed loss functions that quantify how well each network performs its task.

### The Training Process:

The GAN's training follows a minimax game, defined by the following objective function:

➢ $min_G max_D E_{x\sim pdata(x)}[log\ D(x)] + E_{z\sim pz(z)}[log(1-D(G(z)))]$

- The discriminator tries to maximize the probability of correctly classifying real and fake samples.
- The generator tries to minimize the chance of the discriminator correctly identifying the generated data as fake.

During training, the discriminator optimizes its ability to distinguish between authentic and artificial samples. The training process of the generator works to reduce how well the discriminator identifies fabricated data as phony samples. During training, the generator enhances its output realism production skills; simultaneously, the discriminator improves its capability to spot fraudulent data. The training process continues between generators and discriminators until the generated data becomes identical to real data. This framework demonstrates great success in discovering and duplicating detailed visual details for image-generation challenges.

## Types of GANs:

There are several types of GANs, each designed for specific tasks and applications, and a few of them are the most important types in GANs:

### Vanilla GAN:

The original GAN model was introduced by Ian Goodfellow. It uses a simple generator and discriminator network.

- **Generator**: Creates fake data from random noise.
- **Discriminator**: Classifies data as real or fake.
- **Applications**: Basic image generation tasks (e.g., generating simple shapes or handwritten digits like MNIST).

### Conditional GAN (CGAN):

A type of GAN wherein the generator and discriminator are conditioned on extra data, including elegance labels or different facts. This permits the generation of unique categories of records.

- **Applications:** Text-to-photo era (e.g., producing snapshots from textual descriptions). Image technology with unique labels (e.g., growing a particular breed of canine or a particular type of chicken). Style switch (reworking one style of painting into another).

### Deep Convolutional GAN (DCGAN):

A variation of GAN that uses convolutional layers in both the generator and discriminator. This improves the quality of images generated compared to the original GAN.

- **Applications:** Image generation (e.g., generating realistic images of faces, animals, etc.). Super-resolution (increasing low-resolution images for high resolution).

### CycleGAN:

It is designed for unpaired image-to-image translation, where there is no direct correspondence between the training images. It uses two generators and two discriminators to translate images between domains (e.g., from one style to another).

- **Applications**:
    - ✓ Photo enhancement (e.g., turning black-and-white photos into color).

- ✓ Image-to-image translation (e.g., converting summer images to winter landscapes).
- ✓ Artistic transformations (e.g., turning photos into paintings or vice versa).

## StyleGAN:

A GAN architecture focused on generating highly realistic images with control over image features (e.g., facial expressions, hairstyle, etc.). It uses a style-based generator that can control the style at different levels of the image generation process.

- **Applications**:
  - ✓ Face generation (e.g., creating photorealistic human faces, used in deepfakes).
  - ✓ Fashion design (e.g., generating clothing styles and fashion trends).
  - ✓ Artistic image generation (e.g., creating art based on specific attributes or styles).

Let's do one application using Deep Convolutional GAN (DCGAN):

## Dataset overview:

The CIFAR-10 benchmark contains 60,000 color images of 32x32 size that are split into 10 classes, which include airplane, automobile, bird, cat, deer, dog, along with frog, horse, ship, and truck. The image collection consists of 50,000 training pictures alongside 10,000 testing pictures. Now the workflow starts:

**Data Preprocessing:** Strong GAN training needs pixel value normalization of the CIFAR-10 dataset to remain between -1 and 1. Conversion of the dataset into a TensorFlow dataset object along with random shuffling enables numerous information distribution across the training process. The pictures get distributed across 64 samples for each batch regime to enhance training speed.

**Building the Generator:** The generator network serves the purpose of developing realistic images starting from random noise inputs. The generator accepts a randomly sampled, hundred-dimensional latent vector as its input before gradually enlarging the values using Conv2DTranspose layers. Every layer performs an activation function of LeakyReLU followed by batch normalization to preserve stability while fighting off mode collapse. A 32×32×3 pixel output picture (matching the CIFAR-10 image size) utilizes the tanh activation function to scale its values.

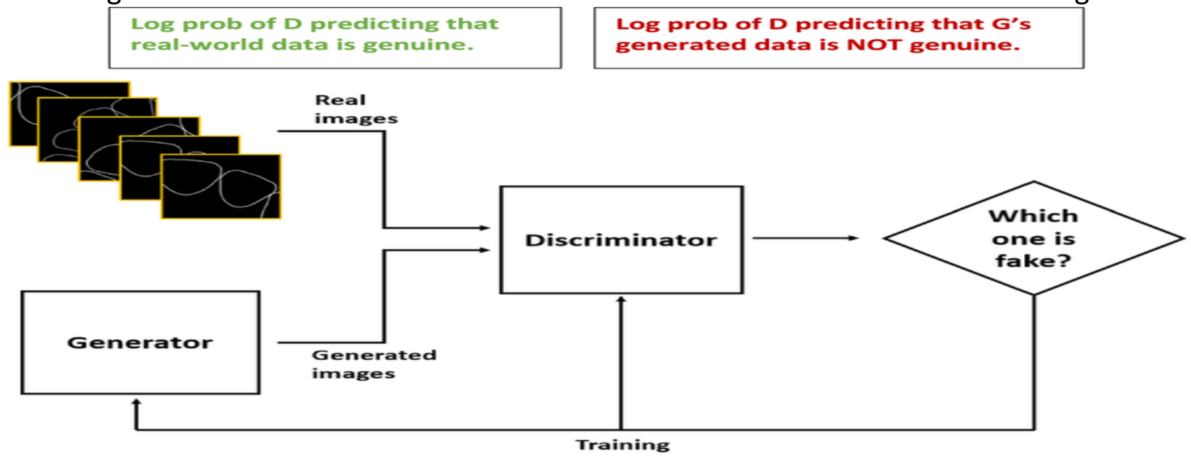
**Building the Discriminator:** The discriminator contains a convolutional neural network structure that decides between real images from CIFAR-10 and generator-produced fake images. The two convolutional layers are supplemented with LeakyReLU activations and dropout layers to ensure optimal model performance. Following the characteristic map flattening process, the network applies a sigmoid activation function to generate a probability score that shows how authentic the image is.

**Defining Loss Functions and Optimizers:** Both networks are skilled at using binary pass-entropy loss. The generator works towards keeping the discriminator mistaken, which prompts its loss function to output results classified as genuine. The discriminator derives its loss value from two components: it checks both actual images matched to the correct labels and artificially produced images matched to fake labels. The networks depend on Adam optimization along with a learning rate of 1e-4 to strengthen their training process.

**Training the DCGAN:** The training loop continues for 70 epochs, during which actual picture samples get selected from the dataset in every generation. A random set of noise vectors generates fake images through an application of the generator network. During evaluation, both real images and fake ones enter the discriminator, which modifies its weights through loss-based evaluation. The training of the generator relies on the gradients obtained from the discriminator's criticism.

**Generating and Saving Images**: During each epoch completion, the generator produces 16 fake PNG images from random noise before saving this data for storage. The generate_and_save_images() function implements matplotlib to arrange a 4x4 grid of visual outputs and names the saved files according to the epoch number.

The defined workflow enables DCGAN training on CIFAR-10 data through efficient implementation that demands low computational complexity. A dependent training technique built with batch normalization plus dropout together with proper activation features leads to stable training and avoids the general GAN challenges of mode collapse and vanishing gradients.


Log prob of D predicting that real-world data is genuine.
Log prob of D predicting that G's generated data is NOT genuine.
Real images
Discriminator
Which one is fake?
Generator
Generated images
Training

*Architecture used in this process*

The workflow applies to our dataset. The dataset of bird images(Birds_Images_dataset) retrieved from Kaggle ('https://www.kaggle.com/datasets/stealthtechnologies/birds-images-dataset/data') contains a variety of bird photographs.

The implementation includes four main elements, which are data loading and three neural networks titled generator network, discriminator network, and training loop. Training alternates between updating both networks. The system imports images through the image_dataset_from_directory mechanism within TensorFlow from "Birds" folders, which allocate dedicated training and testing sections. A normalization operation applies a [0,1] scaling to the images while operating at a pixel size of 64×64 and processing each batch with a size of 32. The collection contains 250 test images and 2,750 training images. A 100-D noise vector enters the generator network, leading it to build 64×64×3 images through transposed convolution layers in an upsampled manner.

The model uses LeakyReLU activations together with batch normalization throughout its structure for stable training before applying a tanh activation to produce output values in the [-1,1] range. The discriminator network utilizes convolutional layers to decrease images from 64×64×3 to provide binary real or fake classifications. The structures contain LeakyReLU activations together with dropout layers to enhance robustness performance. The final layer outputs a single value between 0 (fake) and 1 (real) using a sigmoid activation function. Each loss function depends on binary cross-entropy as the generator strives to trick the discriminator, and simultaneously, the discriminator works to identify genuine from artificial images. The training process extends to 70 epochs, employing the Adam optimizer under a learning rate condition of 1e-4, and the resultant image is shown in fig-1. The functionality of the system suffers from absent validation metrics and no model-saving or sample generation during training capabilities. The basic DCGAN produces believable bird images after undergoing enough training time. If training has been extended, then it will be as like as real images.

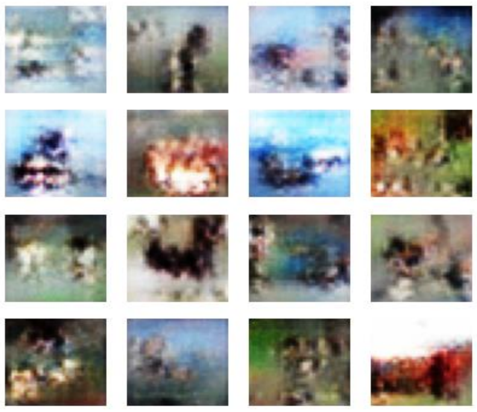


*Figure 1: images of cifar-10*

## How GANs Work in Real Life

GANs currently serve practical purposes across multiple domains, which include:

- **Image Generation:** Developing realistic images from textual instructions works through the combination of models, including DALL-E and MidJourney.
- **Data Augmentation**: Creating synthetic data for training machine learning models in cases where real data is limited.
- **Art and Creativity**: Producing artwork, music, and video content with AI-generated creativity.
- **Medical Imaging**: AI systems enable the improvement of medical images for clinical detection in addition to scientific research.
- **Gaming and Virtual Reality**: Through gaming and virtual reality, desarrollo creates realistic avatars alongside textures and virtual environments.
- **Image Super-Resolution**: Low-resolution images are improved to high-quality images through the application of Super-Resolution GANs (SRGAN).

## Conclusion

Artificial intelligence has been replaced by the generative advertisement network, which allows machines to produce realistic data. From practical use in Gans healthcare and entertainment to artistic applications, they are pushing forward the limits of AI capacity in various ways. It is difficult to train Gans, however, because they need to adjust their hyperparameters carefully and solve problems such as mode collapse (when the generator produces a slight variety in the output). The GANS machine is one of the newest and promising developments in learning, despite these difficulties. GANS is a major component of contemporary AI systems, and more study will improve their stability, effectiveness, and suitability for a wide range of industries. Artificial intelligence has been replaced by the generative advertisement network, which allows machines to produce realistic data. From practical use in Gans healthcare and entertainment to artistic applications, they are pushing forward the limits of AI capacity in various ways. It is difficult to train Gans, however, because they need to adjust their hyperparameters carefully and solve problems such as mode collapse (when the generator produces a slight variety in the output). The GANS machine is one of the newest and promising developments in learning, despite these difficulties. GANS is a major component of contemporary AI systems, and more study will improve their stability, effectiveness, and suitability for a wide range of industries.

## References:

1. *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10893935&isnumber=10893901*
2. *https://www.sciencedirect.com/science/article/pii/S2667096820300045*
3. *https://arxiv.org/abs/1406.2661*
4. *https://www.researchgate.net/publication/349828273_A_Review_on_Generative_Adversarial_Networks*
5. *https://ieeexplore.ieee.org/abstract/document/10692867*
6. *https://ieeexplore.ieee.org/document/10650943*
7. *https://ieeexplore.ieee.org/document/10010725*