# A screenshot of a device Description automatically generated



**Hand Book**

**Foundation of Green Skilling**

**Disclaimer: The content is curated from online/offline resources and used for educational purpose only**

**Table of Contents**

[Learning Outcomes iii](#_Toc182037885)

[Chapter 7: Introduction to Generative AI 4](#_Toc182037886)

[7.1 Introduction 4](#_Toc182037887)

[7.2 Generative Models 6](#_Toc182037888)

[7.3 Training and Inference 9](#_Toc182037889)

[7.4 Generative Adversarial Networks (GANs) 12](#_Toc182037890)

[7.5 Variational Autoencoders (VAEs) 15](#_Toc182037891)

[7.6 Tools and Libraries for Generative AI 18](#_Toc182037892)

[7.7 TensorFlow Generative Model APIs 24](#_Toc182037893)

[7.8 Fine-Tuning Pre-trained Models (Transfer Learning) 26](#_Toc182037894)

[References: 30](#_Toc182037895)

# Premium Vector | White abstract background in 3d paper style | Abstract backgrounds, Abstract, Geometric backgroundLearning Outcomes

After completing this handbook, learner will be able to

Chapter 7: Introduction to Generative AI

|  |
| --- |
| **Learning Outcomes:**  By the end of this chapter, students will be able to:   * Recall the concepts of generative AI models, such as GANs and VAEs, and their applications. * Explain the significance of fine-tuning and transfer learning in optimizing pre-trained models for specific tasks. * Apply popular AI frameworks like TensorFlow, PyTorch, and Hugging Face to build and deploy generative models. * Analyze how fine-tuning generative models like GANs and VAEs can be adapted to specific use cases, such as image creation, text generation, or music production. * Evaluate the quality of generated outputs using metrics like Inception Score (IS) and Frechet Inception Distance (FID), and through visual inspection. * Discuss future trends in generative AI, including accessibility, interpretability, and ethical considerations in AI practices. * Analyze the impact of generative AI on various industries and its potential for innovation in creative arts, data science, and personalized solutions.​ * Explore the applications and challenges of Generative AI, particularly in content creation and ethical considerations in industries like customer service​ |

# 7.1 Introduction

The new advancement in artificial intelligence has produced cutting-edge technologies that no longer just process information but also generate completely novel content. Such innovations have made generative AI a forceful arena and changed industries from art and entertainment to healthcare and business.

## 7.1.1 What is Generative AI?

Generative AI refers to a class of AI that can generate new, novel content based on patterns they have learned from existing data. In contrast to regular AI, which is used for the most part toward classification or predictive tasks, generative AI is in charge of creating something new —and it does not matter what: generating images, creating text, music, and even video. This is not content mimicry, but creation, with possible novel outputs that may come to look similar to or an improvement upon the original.

At its core, generative AI relies on models and algorithms that learn and understand patterns and structures from vast datasets. The two most important architectures applied in generative AI are GANs and Transformers.

Two of the best-known examples of generative AI are: Text-to-image models, which turn a description in text into lifelike images, and large language models that generate human-like text following an input prompt. All of this is already changing the business landscape by enabling rapid content creation, personalized user interactions, and even new forms of art and design.

## 7.1.2 Traditional AI vs. Generative AI: What is the Difference?

Generative AI constitutes a part of artificial intelligence, but it is remarkably different in its purposes and methods from traditional AI. Traditional AI is largely designed based on data analysis and classification and prediction. Traditional applications include spam filters and language translation and recognition capabilities, where accuracy is vital.

But for generative AI, things take on a different turn. Creativity and the generation of new content are the focuses. Where a traditional AI might say the picture is of a “cat” or “dog,” the generative AI would have a realistic drawing of the creature from scratch or perhaps even come up with something completely new, half cat and half dog. That difference makes clear what value generative AI adds to the table, particularly in areas where innovation and originality are of great importance.

**Table1: Traditional AI vs. Generative AI Difference**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Traditional AI** | **Generative AI** |
| **Primary Purpose** | Analyze, classify, and predict based on structured data | Create new content by learning from data patterns |
| **Common Uses** | Spam filtering, language translation, fraud detection | Image generation, content creation, audio synthesis, prototyping |
| **Focus** | Accuracy, reliability, and decision-making | Creativity, innovation, and originality |
| **Output** | Identifies or categorizes (e.g., labels an image as “cat” or “dog”) | Generates realistic or imaginative content (e.g., creates an image of a “cat-dog” hybrid) |
| **Value Addition** | Supports data-driven decisions and automation | Enables creative solutions and novel ideas |

# 7.2 Generative Models

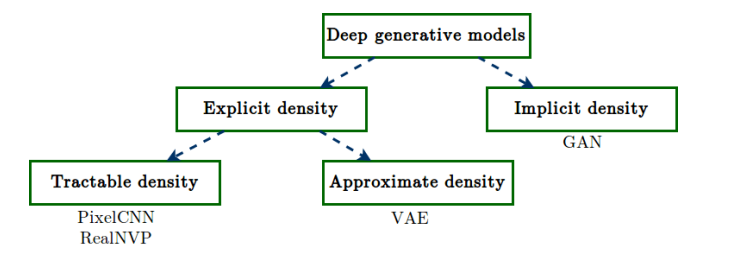
One of the machine learning model classes that produces new samples of data like some input training set is the generative model. Unlike discriminative models, which focus on distinguishing between classes or predicting specific labels, generative models focus on understanding and recreating the data distribution itself. Therefore, these models are capable of generating realistic samples, and they play an important role in all sorts of data synthesis applications including image generation, text generation, and style transfer.

### Key Points:

* **Input**: A dataset is fed to train the model, for example, images of cats.
* **Output**: After training, it can generate new samples looking similar to the original data, for example, images of cats that have never existed in the training set.
* **Objective**: Learn the underlying data distribution and then be able to generate similar data.

## 7.2.1 Types of Generative Models

Generative models broadly can be classified based on the type of probabilistic approach they use:



Source: <https://www.davidinouye.com/course/ece57000-fall-2020/lectures/gans.pdf>

### Explicit Density Models

These explicitly define a probability distribution over the data and often rest on statistical assumptions about the structure of the data. The following are some examples of explicit density models:

* **Variational Autoencoders:** VAEs approximate a probabilistic model of the data using a neural network, usually assuming a Gaussian distribution. They learn to encode input data into a compressed latent space and can generate new samples by decoding points from this space.
* **Autoregressive Models:** In these models, such as **PixelRNN** and **WaveNet**, data is generated one step at a time as a sequence. These models make an estimate of the conditional probability of each element in the sequence given all previous elements.

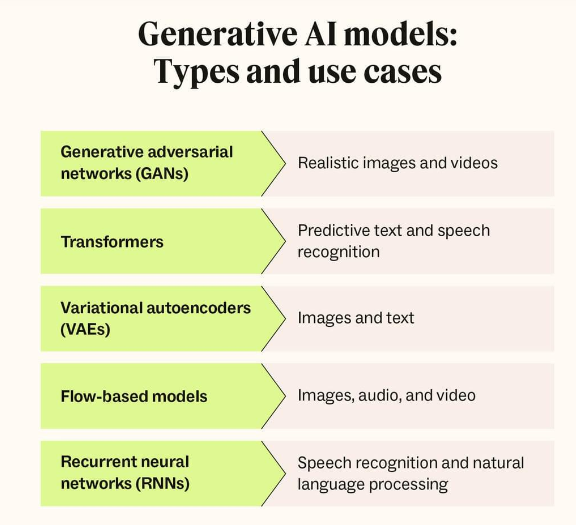
### Implicit Density Models

Implicit density models do not specify a probability density function directly but instead aim to learn to generate samples that are close to being realistic:

* **Generative Adversarial Networks (GANs):** GANs use two competing networks known as the generator and discriminator to produce realistic samples. The role of the generator is to generate apparently real fake data, while the discriminator's role is to be able to distinguish real and fake samples.
* **Energy-Based Models (EBMs):** Instead of defining directly a probability distribution, a function is defined that assigns lower energy to the real samples and higher energy to the fake ones.

## 7.2.2 Applications of Generative Models

Generative models have completely changed the way we work with, transform, and generate data.



https://www.zendesk.com/dk/blog/generative-ai-guide/

**Here are the most important application areas**:

#### Image Generation and Editing

* Generative models can now create high-quality images starting from scratch. Parts of the creative process can be automated by artists and designers.
* Examples: AI-generated art, StyleGAN-generated faces, and deepfake technology.

#### Text Generation and Translation

* These text generative models, for example, GPT, produce text that sounds as if it is written by a human. They used for applications such as chatbots, content generation, and translation.
* Examples: ChatGPT, DALL-E's captioning, and OpenAI Codex for programming.

#### Data Augmentation

* Generative models can be used to augment training datasets by generating new samples, which come in handy when there is limited data. For example, GANs can be used to create variations of existing images for training computer vision models.

#### Drugs Discovery and Healthcare

* In drug discovery, generative models can be used to generate novel molecular structures or design proteins with specific properties, accelerating research and development in healthcare.

#### Other Creative Applications

* Music composition, the development of video game character, virtual worlds, and animation are other fields that leverage generative models to create immersive and unique content.

**Examples**:

* **StyleGAN2**: A model capable of generating ultra-realistic human faces that don’t exist.
* **OpenAI’s DALL-E**: Generates images based on textual descriptions, opening new avenues for art and design.
* **GPT-3**: A language model that can generate coherent and contextually accurate text for a wide range of applications, from chatbots to writing assistants.

## 7.2.3 Challenges and Considerations

While generative models hold immense potential, they also come with challenges and ethical considerations:

### Technical Challenges

* **Training Instability:** GANs particularly are known to suffer from instability in training due to the adversarial nature, which creates mode collapse and gradient vanishing problems.
* **Computational Complexity:** Generative models often require large computational powers, especially for training over large data or generating quality images or text.
* **Evaluation Metrics:** It is extremely challenging to evaluate the quality of the generated data. Standard metrics such as accuracy do not apply. Specialized metrics such as Inception Score (IS) or Frechet Inception Distance (FID) are used.

### Ethical and Social Considerations

* **Misuse and Privacy Risks:** Generative models can create deepfakes or unauthorized realistic content, which raises concerns about privacy and misuse.
* **Bias in Generated Content:** If trained on biased data, generative models can reproduce and even amplify those biases to lead to unfair or discriminatory outputs.
* **Intellectual Property:** Ownership of AI-generated content is still a matter of debate. Questions on copyright, authorship, and responsibility are very pertinent, especially when these models are used for commercial purposes.

### Environmental Impact

* The computational power required in training large generative models, however, has serious implications in terms of energy; therefore, sustainable practices include model optimization and efficient training.

# 7.3 Training and Inference

Generative models have two fundamentally different phases in their lifecycles: training and inference.

* **Training**: This is a phase where the model learns from some dataset by adjusting its parameters for understanding what the underlying patterns or features of the data are. This is usually a process involving minimization of some loss function via optimization techniques.
* **Inference**: After training, the model enters the inference stage, where it can generate new samples or make predictions based on the patterns it learned during training.

For instance, given images of animals, a GAN trained on them during this inference, this GAN can generate plausible images about new animals akin to what is in training.

## 7.3.1 Training Process

Training a generative model is not an easy task. There are several key steps and components involved in the training process. Let's start to dive into the main components of the training process.

### Loss Functions

Loss functions play a very important role in training generative models. They define the objective the model tries to optimize. Each type of generative model has its own unique loss function.

1. **Mean Squared Error (MSE):** MSE is widely applied in autoencoders where it calculates the average squared difference between the original data and the reconstructed data.
2. **KL-Divergence:** In Variational Autoencoders (VAEs), KL-divergence is very often used for computing how one probability distribution deviates from a second, expected probability distribution. The VAE tries to minimize the KL-divergence so that the latent space follows the Gaussian distribution.
3. **Adversarial Loss (GANs):** GANs make use of adversarial loss for its generator and discriminator. In this case, the generator's loss is determined based on how well it can deceive the discriminator, while the loss for the discriminator is determined based on its ability to distinguish between the real and fake samples.

### Backpropagation and Optimization

* **Backpropagation:** This is the process through which the errors propagate back into the network in order to adjust the weights. It’s essential in training neural networks, as it allows the model to learn by correcting errors iteratively.
* **Optimization Algorithms:** The optimization algorithms update model parameters to minimize the loss function. Popular optimizers include:
  + **Stochastic Gradient Descent (SGD):** It calculates gradients and then updates the weights with tiny random batches of data.
  + **Adam:** A very widely used optimizer, combines the effects of momentum and adaptive learning rate adjustments.

### Metrics for Evaluating Generative Models

The measurement of quality of samples generated by the Generative models is very important but the traditional metrics like accuracy aren’t suitable for this quality measurement.

Here are some of the popular metrics that is used the measure the quality of the output:

1. **Inception Score (IS):** It measures the quality and diversity of the images generated. The higher IS is, the higher is the quality and diversity.

**How It Works:** This score is computed from a pre-trained Inception model, commonly Inception v3 applied to generated images. It calculates the entropy of the predicted class labels:

*Quality:* High-quality images are classified with high confidence, that is, low entropy within an image.

*Diversity:* Different output spreads across classes due to the high entropy across images.

**Interpretation:** A high Inception Score means the model is generating images of both high quality and high diversity.

1. **Frechet Inception Distance (FID):** It measures the similarity in distribution between the generated and real images. The smaller the value of FID, the more close the generated images are to real images.

**Purpose:** FID is used to measure the similarity between the distributions of real and generated data, hence providing a statistical view of quality.

It is basically comparing the distributions in feature vectors between real and generated images drawn from some certain layer in Inception v3 model. The distance was used as the Fréchet Distance, also referred to as Wasserstein-2 distance between the corresponding distributions:

* Real and generated images are each represented by a mean and covariance of feature vectors.
* The FID score is therefore computed as the distance of these means and covariances.

**Interpretation:** The lower FID scores indicate that the produced images are closer to the real ones and represent better quality and similarity.

1. **Reconstruction Loss (VAEs):** Measures how well output by the generator is comparable to the input. Or how well the VAE managed to capture the data distribution.

This quantifies how well the VAE learned to represent the data distribution; this could be especially useful for generative models in the reconstruction tasks.

**How does it work?** In VAEs, reconstruction loss is the difference between the original input and the generated output. Reconstruction loss is typically the Mean Squared Error for continuous input data and Binary Cross-Entropy for discrete input data.

A small reconstruction loss means that VAE is learning the inherent probability distribution of the data well enough to have good input reconstruction with zero, or close to zero error.

1. **Human Evaluation:** For certain applications like text generation, human subjective evaluation can be used for quality and coherence of the generated content.

In many domains, particularly with text, music, or other sequential data, human evaluation is necessary to assess quality beyond automated metrics.

How It Works: Human judges evaluate the output against criteria such as coherence, relevance, creativity, or fluency. Scoring could use a rubric or Likert-type scales to score coherence and fluency.

**Interpretation:** Human scoring can provide a qualitative means of rating the ability of the model to create meaningful and appropriately contextualized content, things that cannot be captured by auto metrics.

In practice, putting all these together will provide an evaluation that encompasses quality, diversity, fidelity, and subjective human preferences. That will help understand better the strengths and limitations of generative models in creating realistic, varied, and high-quality outputs.

## 7.3.2 Inference Techniques

Once a generative model is trained, it can be used to generate new samples. This process is called inference and has particular techniques for controlling and producing varied outputs.

### Sampling Methods

It's basically inference through the means of sampling because that enables one to generate new examples based on the data learned in distribution.

* **Random Sampling:** The random selection from latent space or noise distribution
* **Importance Sampling:** Sampling in a manner such that some samples will get more weights or fewer weights as it might better represent the actual distribution.
* **Markov Chain Monte Carlo (MCMC):** More advanced sampling technique for creating a sequence of samples such that each sample depends on the preceding one.

### Latent Space Exploration

In VAEs and GANs, latent space is an abstract compressed representation of data. During inference, we can explore this space to produce new data:

* **Latent Vector Manipulation:** We can, by manipulating the latent vectors' values, control some aspects of the output generated. For example, in a GAN trained on images of faces, changing the latent vector might change facial expressions or hair color.
* **Interpolation:** We can obtain intermediate outputs by interpolating between two points in the latent space. This technique is handy for tasks like image morphing or text blending.

### Conditional Generation

In models such as conditional GANs (cGANs) and conditional VAEs (CVAE), we can add more context or conditions to control the output. For example, we can generate images of cats or dogs based on a certain condition.

# 7.4 Generative Adversarial Networks (GANs)

Generative Adversarial Networks, or GANs for short, is one type of generative models where two neural nets with the names generator and discriminator are trained together to create a competitive framework for such models. This is called an adversarial setup in which GANs produce highly realistic data samples such as images and audio.

* **Generator (G):** This network receives a random noise vector and produces data that might seem like the training set.
* **Discriminator (D):** This network takes a real or generated sample as input and tries to classify it as real or fake.

The objective of GAN training is that the generator will be producing samples that are indistinguishable from real data while the discriminator gets better at telling real samples from the fake ones.

## 7.4.1 How GANs Work: The Adversarial Training Process

The central theme behind GANs is adversarial training, in which the generator and discriminator play a "game" that is locked in with one trying to outwit the other:

### 1. Training of Generator:

* The generator draws samples from noise and attempts to deceive the discriminator so that it believes these generated samples are real.
* It tries to maximize the chances of the discriminator labeling them as real.

### 2. Training of Discriminator:

* The discriminator is trained on real and fake samples. It tries to maximize its ability to correctly identify real samples as "real" and generated samples as "fake."

### 3. The Game:

* The generator and discriminator are trained alternately. The generator gets better at fooling the discriminator, and the discriminator gets better at distinguishing real samples from fakes.
* Training ideally becomes balanced such that the discriminator cannot classify real and generated samples with better-than-random accuracy.

## 7.4.2 Popular GAN Architectures

The GAN architecture has evolved quite a lot since its introduction, with numerous architectures improving upon the original design for particular use cases and performance enhancement.

Some of the popular GAN architectures are listed below:

### Deep Convolutional GAN (DCGAN)

* DCGAN is a modified version of GANs, where it employs the use of convolutional layers, which makes the entire process very efficient especially with the generation of images.
* Major contributions of DCGAN are:
  + Convert fully connected layers of generators and discriminators into the convolutional layers that have the ability to grasp the spatial hierarchies.
  + Make the use of batch normalization by stabilizing the training that enhances convergence.
* Applications: DCGAN is implemented mostly in applications such as generation of images, super resolution, and style transfer.

### Conditional GAN (cGAN)

* This is an extension of the GAN. In this case, the introduction of additional information to the generator and discriminator allows conditional data generation.
* The most illustrative example is found below:
  + In the case of image synthesis, labels like "dog" or "cat" can direct a GAN to produce the desired animal.
  + When it comes to image-to-image translation tasks, in the case of cGANs, control over the content that is generated is left entirely to the user.
* Applications:
  + Image-to-Image Translation (e.g., converting sketches to colored images).
  + Text-to-Image Generation (e.g., generating images based on text descriptions).

### StyleGAN

* StyleGAN, developed by NVIDIA, introduces a new generation of high-quality images with the concept of "style" layers, giving greater control over different aspects of the generated images, such as face shape, hair, lighting, etc. StyleGAN uses a unique generator architecture with a mapping network to produce a latent vector that controls various aspects of the generated image.
* Applications of StyleGAN
  + Photorealistic face generation.
  + Facial attribute manipulation.

## 7.4.3 Challenges in Training GANs

Training GANs is notoriously challenging due to several inherent difficulties.

Here are some of the most common issues:

### Mode Collapse

Mode collapse is a phenomenon in which the generator produces only a limited variety of samples, focusing on a few specific outputs instead of covering the full diversity of the data distribution. The model might generate similar-looking outputs repeatedly, meaning it becomes not very effective.

Solutions:

* **Mini-batch Discrimination:** It makes the generator to produce different samples within a mini-batch.
* **Unrolled GANs:** It is a method in which discriminator is "unrolled" over several steps so that it can look ahead and therefore doesn't encourage mode collapse.

### Training Instability

Training of a GAN is a sensitive balance between the generator and discriminator. The training becomes unstable if one network is too strong relatively to the other.

Solution:

* **Gradient Penalty:** Incorporated during the training of discriminators. This penalty had been efficient for Wasserstein GANs (WGAN).
* **Spectral Normalization:** Works directly on the discriminator's gradients and, thus makes training more stable.

### Vanishing Gradients

Gradients vanish whenever the discriminator is too correct because gradients passed through after this have become too weak to be learned; such can make learning extremely slow or get stuck

Solution:

* Wasserstein Loss: In the WGAN setting, replaces common cross-entropy loss due to a binary classification of the standard WGAN with its loss. It resolves vanishing gradient.

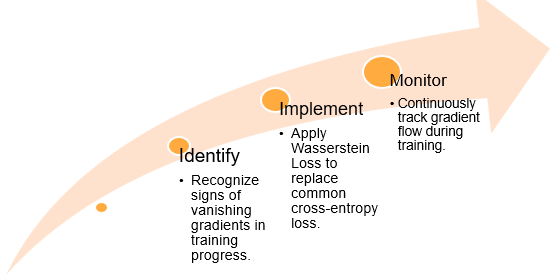


Fig: Challenges in Training GANs: Vanishing Gradients

# 7.5 Variational Autoencoders (VAEs)

Variational autoencoders are generative models in the sense that they enable the model to learn a probabilistic representation of data thereby allowing the generation of new data points. Basically, they use an architecture called autoencoders where encoder maps input information into a space with significantly fewer dimensions as follows. The decoder then expands that mapped input information so as to reconstruct the initial input into a form acceptable for that task.

However, unlike other autoencoders that give a unique determinate code for one input, VAE produces a probability distribution of values for every input in latent space. This has turned out to be a strength point for the VAE: it allows for excellent production of new data samples as well as interpolation among data samples within the latent space.

## 7.5.1 Variational Autoencoder (VAE) Architecture

Variational Autoencoder, also known as VAE, is a generative model that learns a structured, continuous "latent space" representation of the input data. VAEs consist of three essential parts:

### Encoder:

* The encoder is a neural network that compresses the input data into a simplified representation that can be called the latent space, much like that of an image or text.
* It produces two important values: the mean (μ) and the variance (σ²) of a Gaussian (bell curve) distribution. These parameters summarize the important information in the input data and represent the compressed form of the data.

### Latent Space Sampling:

* Having acquired the mean and variance from the encoder, we take a sampled point from the Gaussian distribution. A point sampled like this is called the latent vector that contains only key information in compact forms relating to the data.
* There forms a bottleneck, thus passing just the necessary points of input through; removing noise and redundancy that does not belong in such space.

### Decoder:

* There is a second neural network that tries to reconstruct the original data by taking the sampled point from the latent space - the decoder.
* This is where the reconstruction comes in, which essentially makes the VAE capable not only of compression but also of reconstructing the original data from the latent representation.

The latent space of a VAE is continuous and well-structured; similar data points lie close to each other in the latent space. This allows us to smoothly transition between one data point and another and, therefore, be able to "interpolate" between different samples, bringing about a gradual transformation or blend from one sample into another.

## 7.5.2 Training a VAE: Evidence Lower Bound (ELBO)

Training a VAE involves optimizing a special objective function known as the Evidence Lower Bound, or ELBO. The ELBO is a function that aims to balance two objectives: accurate reconstruction of the data and proper structuring of the latent space.

There are two major components in the ELBO:

### Reconstruction Loss:

* It measures the decoder's ability to reconstruct the original data from the latent vector. This is essentially an error measure and it shows how close the VAE output is to the actual input data.
* In case of image data, a few popular reconstruction loss functions are Mean Squared Error (MSE), which penalizes VAE according to how far every pixel is from its original value or Binary Cross-Entropy (BCE) that works well for binary images black and white.

### KL-Divergence Loss:

* The Kullback-Leibler Divergence is the difference by which the output distribution of the encoder differs from the target Gaussian. Minimizing this assists a VAE in structuring a latent space wherein data points that are closer to each other will lie near each other.
* Decreasing the divergence also structures the latent space in such a way that it gives a smooth transition and proper representation of the data.

ELBO is obtained by summing up the following two components:

This setup ensures that:

* The latent space is structured in a smooth, continuous way, making interpolations between data points possible.
* The decoder becomes skilled at recreating realistic samples from the training data.

## 7.5.3 Sampling and Latent Space Exploration in VAEs

Once trained, VAEs open the way to exploring the latent space in fascinating ways:

### Data Generation:

* By sampling random points in the latent space and feeding them to the decoder, we can generate new data samples that resemble the training data.
* For instance, if trained on images, the VAE could create new images with similar characteristics.

### Interpolation in Latent Space:

* By selecting points that lie between two known data samples in the latent space, we can create a smooth transformation from one sample to the other.
* This technique is used in applications like image morphing, where one image gradually changes into another.

### Anomaly Detection:

* A VAE trained on "normal" data can detect unusual or abnormal data by checking how well new samples fit within the latent space distribution. If a new sample doesn’t align well, it can be flagged as an anomaly, which is useful for applications like fraud detection.

## 7.5.4 Popular Implementations of VAE

Several variants of VAE have been developed in order to extend its basic capabilities. The most popular variant is Conditional Variational Autoencoder (CVAE):

### Conditional Variational Autoencoders (CVAE)

* Conditional VAEs increase the control of the generator by allowing both the encoder and decoder to use extra information (such as labels or conditions) in addition to the data input.
* For example, if we train a CVAE on images of the digits 0 through 9, we could supplement the input with the label of the digit we were training on, say, "7." We would let it generate images for us, specifying the desired label, which could be "7.".

### Beta-VAE

Beta-VAE introduces a hyperparameter, β, which scales the KL-divergence term in the loss function:

A higher β encourages the model to focus on disentangling latent features, which is helpful for interpretability and understanding how specific features of data affect the generated output.

# 7.6 Tools and Libraries for Generative AI

* Developing generative AI models from scratch is a very challenging task, and several frameworks and libraries have been developed to make the process easier. These tools offer basic building blocks in the form of neural network components, optimizers, data processing utilities, and more, which makes the process easier for developers to experiment and deploy generative models.
* Some of the most widely used frameworks and libraries are listed below:
  + **TensorFlow:** Google provides TensorFlow which is quite scalable and has good support for production environments. It does have broad tools for deep learning and specific libraries for generative models like GANs and VAEs.
  + **PyTorch:** PyTorch is from Facebook, and is especially favored in research because of the flexibility it provides, and also its intuitive design, especially when considering custom model architectures.
  + **Hugging Face Transformers:** A library which is more interested in the generation of natural language; it provides ready-to-use, powerful pre-trained models for text-related tasks.
  + **Diffusers:** A library which is also focused on diffusion models in image and text generation; it's also provided by Hugging Face.
* All the frameworks have their unique strengths and usually combine with other tools to make some generative tasks easier.

## 7.6.1 TensorFlow

* The most versatile deep learning framework, which provides many tools for building, training, and deploying generative models, is TensorFlow.
* It has strong support on production environments and is highly recommended for research as well as large-scale deployments.

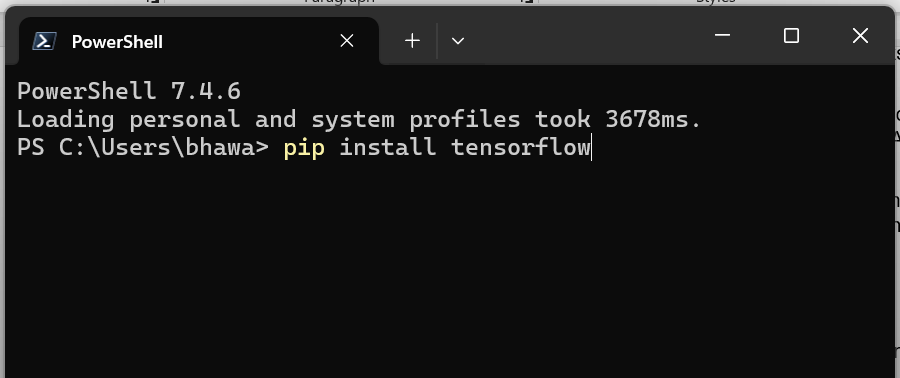
### Libraries and Extensions in TensorFlow:

* **TensorFlow Keras:** A higher-level API of TensorFlow to simplify model development. This is intuitive and easy to use, which makes it a popular tool for neural networks in applications such as image, audio, or text generation.
* **TensorFlow GAN (TF-GAN):** It is specifically used for GAN in TensorFlow. It provides several helper functions and pre-cooked architectures and evaluation metrics for GANs so that one can easily play with different designs of GANs.
* **TensorFlow Addons:** Brings about additional functionality from TensorFlow such as the utility of spectral normalization used in stabilizing GAN's training by constraining weights helping it to converge better than during training.

### Installation

You can install TensorFlow via pip. The command below installs the latest stable version of TensorFlow:

pip install tensorflow



If you plan on using TensorFlow-GAN and TensorFlow Addons for generative models:

pip install tensorflow-gan tensorflow-addons

A screenshot of a computer

Description automatically generated

### Basic Import

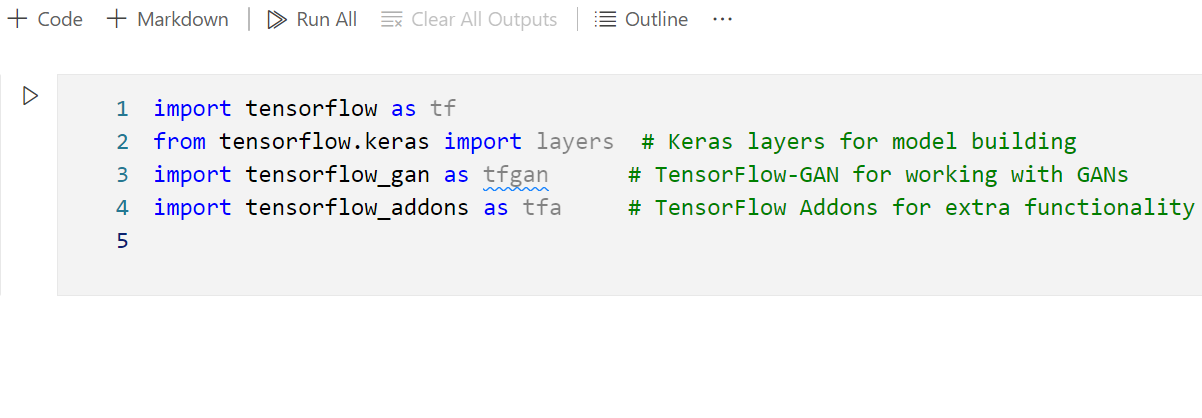
Once installed, you can import TensorFlow and any relevant submodules:

import tensorflow as tf

from tensorflow.keras import layers  # Keras layers for model building

import tensorflow\_gan as tfgan      # TensorFlow-GAN for working with GANs

import tensorflow\_addons as tfa     # TensorFlow Addons for extra functionality



### Checking Installation

You can confirm that TensorFlow is correctly installed by checking the version:

print(tf.\_\_version\_\_)

A screenshot of a computer program

Description automatically generated

## 7.6.2 PyTorch

* PyTorch is special for its flexibility along with a dynamic computational graph which allows developers to see their modifications on the fly allowing easy debugging and customization of models.
* PyTorch can be found in most major research work and proves valuable in trying out new architectures on generative models.

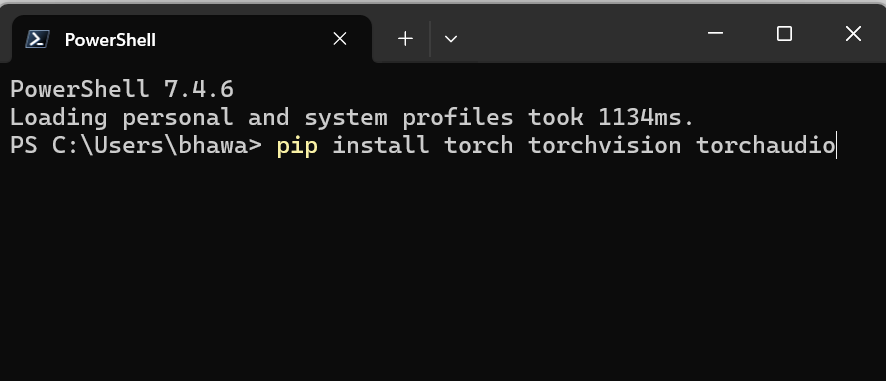
### Key Libraries and Extensions in PyTorch:

* **PyTorch Lightning:** It standardizes the process of training loops and comes with tools for logging and checkpointing, thereby simplifying model training, experimentation, and scaling. This leaves focus on model design and de-emphasizes boilerplate code.
* **torchvision:** This is the library for image processing including access to datasets of images and pre-trained models helpful in building and testing the generative models especially about image data.
* **Hugging Face Diffusers:** This is a library comprising a significant number of diffusion models with the capability of generating images and text in PyTorch. Pipelines and pre-trained models make Diffusers easy for training and experimenting with diffusion models.

### Installation

Install PyTorch by choosing the appropriate version on the official PyTorch website. For example, to install the latest version for CUDA 11:

pip install torch torchvision torchaudio



If you need Diffusers by Hugging Face:

pip install diffusers

A screenshot of a computer

Description automatically generated

### Basic Import

You can import PyTorch and its key components as follows:

import torch

import torch.nn as nn  # Neural network components

import torch.optim as optim  # Optimizers

import torchvision.transforms as transforms  # Image transformations

from diffusers import StableDiffusionPipeline  # Importing Diffusers for diffusion models

A screenshot of a computer

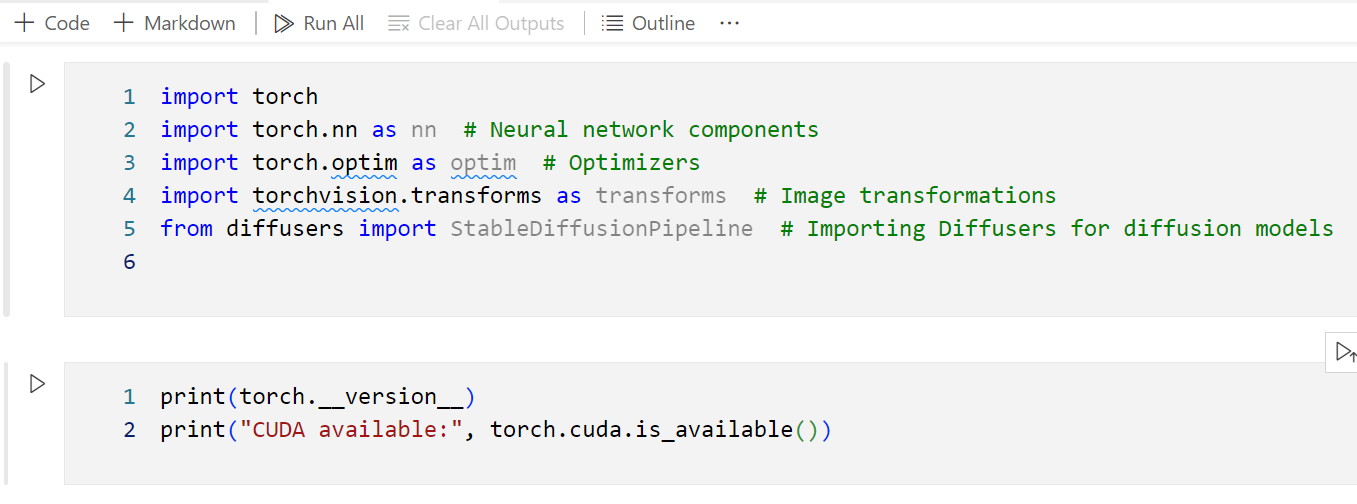
Description automatically generated

### Checking Installation

To confirm PyTorch and check if CUDA (for GPU) is available:

print(torch.\_\_version\_\_)

print("CUDA available:", torch.cuda.is\_available())



## 7.6.3 Hugging Face Transformers

* The Hugging Face Transformers library is a one-stop shop for all natural language processing and text generation tasks.
* It contains thousands of pre-trained models fine-tuned to achieve state-of-the-art results in a wide variety of NLP tasks, including GPT, BERT, and T5.

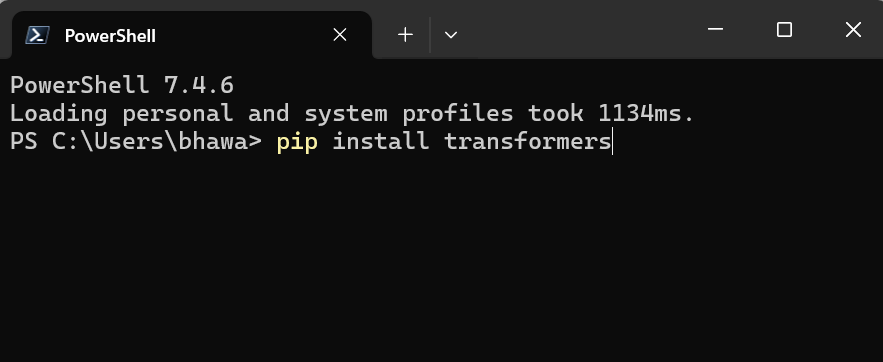
### Key Features:

* **Thousands of Pre-Trained Models:** access to models pre-trained on large text datasets, saving time and computational resources.
* **Fine-Tuning:** Users can fine-tune models on particular tasks. This enables it to support even more specialist applications like summarization, question answering, and sentiment analysis.
* **Hugging Face Model Hub:** Hugging Face provides a central location through which developers can share or retrieve models. That further enables community collaboration, therefore enhancing development.

### Installation

Hugging Face Transformers supports both PyTorch and TensorFlow. Install the library with:

pip install transformers



### Basic Import

Import the Transformers library and load a pre-trained model:

from transformers import pipeline, AutoModelForCausalLM, AutoTokenizer

# Example: setting up a text generation pipeline

text\_generator = pipeline("text-generation", model="gpt2")

A screenshot of a computer program

Description automatically generated

### Checking Installation

You can confirm the version with:

import transformers

print(transformers.\_\_version\_\_)

A screenshot of a computer

Description automatically generated

# 7.7 TensorFlow Generative Model APIs

TensorFlow creates, trains, and evaluates generative models through a set of APIs. These APIs make the design process easier with specific functions and models for various types of generative tasks.

Here are some APIs:

* **TensorFlow GAN (TF-GAN):** Specifically designed library to perform GANs. This library has already implemented loss functions, metrics, and model templates for one to use.
* **TensorFlow Probability (TFP):** Rich library for probabilistic models, including VAEs, that supports distributions and reparameterization tricks for latent space sampling and inference.
* **Hugging Face Diffusers API:** While originally built around PyTorch, the Hugging Face Diffusers API is now also usable with TensorFlow to investigate diffusion models, which have achieved spectacular successes on harder image and text generation tasks.

Let's dive deeper into each of these, with key features and example use cases.

## 7.7.1 TensorFlow GAN (TF-GAN)

* TF-GAN is intended to facilitate easier experimentation with GANs using TensorFlow. It offers facilities specific to GANs that obviate the complexity of designing the architecture and ensure the models are optimized for stable training as well as high-quality generation.
* In particular, GANs are notoriously difficult to train. Thus, TF-GAN has a set of tools specifically customized toward the stability of training and performance profiling of GANs.

### Features of TF-GAN

**GAN-Specific Loss Functions:** Loss functions play a very significant role in stabilizing the training of GANs.

TF-GAN provides several popular loss functions, including:

* **Wasserstein Loss:** This is often employed for training Wasserstein GANs (WGANs) to avoid mode collapse along with various instabilities.
* **Least-Squares Loss:** Often used in Least-Squares GANs (LSGANs) for the generation of sharper and stable images.
* **Hinge Loss:** Very successful when used in GANs, especially for better generation.

**Evaluation Metrics:** For judging the performance of a GAN, TF-GAN offers some specific metrics that are relevant to GAN:

* **Frechet Inception Distance (FID):** Evaluated the similarity between generated and real images to measure visual quality.
* **Inception Score (IS):** Evaluates the diversity and quality of images generated, a very good measure of how well a GAN can generate realistic and varied samples.

**Pre-built Architectures:** TF-GAN delivers template architectures for some of the most commonly used GANs, so getting started that much faster:

* **DCGAN:** One of the most commonly used GAN architectures used in image generation.
* **Conditional GAN (cGAN):** Allows the generation of images conditioned on a specific class, such as generating images of specific animals or objects.

## 7.7.2 Variational Autoencoders with TensorFlow Probability (TFP)

* TensorFlow Probability (TFP) is a library designed as an additive library for TensorFlow, to design probabilistic models.
* In this class, one of the members is the Variational Autoencoders (VAEs). VAEs rely on specific types of probabilistic layers to be able to handle the latent space and sample appropriately without sacrificing differentiability.
* It makes the task highly easy by offering probabilistic layers, distributions, and sampling methods.

### Features of TensorFlow Probability

* **Probabilistic Layers:** TFP contains probabilistic layers that enable latent variable sampling as well as the reparameterization of a gradient-based optimization. With these layers, defining an encoder/decoder for a VAE is made simple.
* **Distributions API:** This API owns a wide array of probabilistic distributions that are central to VAEs such as Gaussian, Bernoulli, and Categorical :
  + **Gaussian Distribution:** The most commonly implemented distribution for VAE. The latent space of a model is modeled with a normal distribution.
  + **Reparameterization Trick:** Which allows the VAE to backpropagate gradients through the stochastic layers in such a way that sampling and training occur without any problem.

## 7.7.3 Diffusion Models using TensorFlow and Hugging Face Diffusers

* Diffusion models are a new class of generative models based on the principle of iteratively adding noise to data and then learning to remove this noise step by step to generate samples.
* Hugging Face's library for PyTorch was called Diffusers, but it has now been extended to also support TensorFlow. That opens it up to anyone in the TensorFlow ecosystem who wants to try out these models.

### Features of Hugging Face Diffusers

* **Pre-trained Diffusion Models:** Diffusers offer a set of pre-trained diffusion models, like Stable Diffusion, that allow users to generate high-quality images starting from text or noise
* **Flexibility and Ease of Use:** Using Diffusers, a user will be able to control the diffusion process in minute detail, including the number of diffusion steps and the noise scheduler, as well as batch processing. This might especially prove very useful when fine-tuning the output quality of generated images or text.

# 7.8 Fine-Tuning Pre-trained Models (Transfer Learning)

* Transfer learning is the method by which one uses the model trained on the large, general-purpose dataset for the more specific, often smaller target task. It is particularly useful in generative AI, where the generation of good-quality, diverse data from the ground up can be quite computationally expensive.
* Through transfer learning, practitioners will be able to transfer the knowledge learned by pre-trained models those trained on gigantic datasets for example to other applications, using fewer training samples and time.

## 7.8.1 Why Fine Tune Pre-Trained Models?

Fine-tuning a pre-trained model has several advantages that make fine-tuning very advantageous in generative tasks:

* **Efficiency:** Training a new generative model from scratch is computationally expensive and time-consuming. Due to the fact that fine-tuning relies on the foundational knowledge which has been embedded in pre-trained models, training time and computational resources are greatly lower.
* **Improved Performance on Small Datasets:** When the target dataset is small or specialized, pre-trained models can significantly enhance performance. These models already capture a wide range of features from general data, which can be refined to fit the specific domain, especially useful for cases like rare image styles or niche text categories.
* **Knowledge Transfer:** Highly representative and complex models are developed with billions of images or words in a dataset. This knowledge transfer proves useful in many specialized generative tasks to be able to comprehend complex patterns.

## 7.8.2 Common Applications of Transfer Learning in Generative AI

Transfer learning for generative AI is very common across domains for quality, customized content creation:

* **Image:** For GANs, the style-specific image generation is more often applied through the transfer learning. Thus, the pre-trained GAN can be fine-tuned on the general images to be specific to the artistic style or a theme.
* **Text:** Fine-tuning a language model such as GPT or BERT could let it perform specially the generation of product descriptions or answer questions in a chatbot. One can even write in a specific tone.
* **Music and Art Creation:** Generative models may be fine-tuned to produce outputs in specific musical styles or even artistic forms, such as classical music or impressionistic painting. Finetuning can incorporate these stylistic nuances into the generated content.

## 7.8.3 Fine-Tuning Techniques

There are many approaches to fine-tuning, depending on the type of model, and for the best application goals.

Three techniques are found below:

### Feature Extraction

In feature extraction, the pre-trained model would act as a fixed feature extractor, retaining core knowledge in their layers while accommodating the new dataset's specific characteristics.

#### Steps of Feature Extraction:

* **Load the Pretrained Model:** Import the model and freeze its layers, which prevent them from updating the parameters during training.
* **Replace the Last Layers:** Set the output layer(s) to be whatever number of classes or output features that would be required by the target dataset. In the case of generative models, for instance, it may add further layers at the top that enhance the generator's output.
* **Train Only the New Layers:** As the pre-trained layers are frozen, it should train only on new layers to learn over the target dataset.

This method is effective because the target task is closely related to the original task, given that the model retains most of its learned knowledge while adapting the final layers to the new output requirements.

### Full Model Fine-Tuning

Full model fine-tuning provides much deeper adjustments by retraining all of the layers of the model. This is computationally heavier, but it can potentially yield better results, especially on very complex tasks or when the new dataset considerably differs from the original.

#### Steps for Full Model Fine-Tuning:

* **Load the Pretrained Model:** The model should be loaded without freezing the layers.
* **Selective Unfreezing:** Optionally, unfreeze only the latter layers, which are usually closer to the output, so that the low-level features from the pre-trained model are retained while the high-level features are adapted.
* **Training on the New Dataset:** Then train with a lower learning rate to not wash out the pre-trained knowledge and destabilize the weights of the model.

This approach allows fine-grained control over the features of the model and is useful if the new dataset is largely different in style or structure.

### Layer-wise Learning Rate Adjustment

Sometimes, different layers are updated at different rates. In this approach, layers closer to the input typically have lower learning rates to retain foundational knowledge, while the layers closer to the output are updated more aggressively to adapt to the new dataset.

This method is very useful for complex generative models, where some information acquired during the initialization with a pre-trained model (for instance, elementary shapes for image generation) should remain almost intact.

## 7.8.4 Fine-Tuning Pre-trained GANs with TensorFlow

GAN fine-tuning is the process of updating the generator and/or discriminator on a new dataset while preserving the general knowledge from the pre-trained model. With TF-GAN, this procedure can be much less labor-intensive in TensorFlow.

### Steps for Fine-tuning a Pre-trained GAN:

* **Load the pre-trained GAN model:** Load in the pre-trained GAN model including the generator and the discriminator
* **Freeze or Unfreeze layers**: Choose to freeze layers in the generator and discriminator based on the dataset similarity and the degree of adaptation needed.
* **Train With at Lower Learning Rate:** Train using a reduced learning rate to stabilize the training and keep the weights that were pre-trained.

This fine-tuning methodology has proven particularly useful for jobs such as style adaptation, in which a GAN already pre-trained on generic images is trained with regard to specific styles, perhaps anime or artistic sketches.

## 7.8.5 Fine-tuning Pre-trained VAEs using TensorFlow Probability

In the case of VAEs, the fine-tuning is largely focused on the updates within the latent space representation as well as the decoder itself to improve reconstruction quality over the new data set.

* **Load the Pre-trained VAE Model:** Load the pre-trained VAE and understand what parts of it - encoder or decoder - are required to be fine-tuned.
* **Freeze Layers for Feature Extraction or Unfreeze for Full Fine-Tuning:** Freeze layers in the encoder if one is interested in feature extraction or unfreeze layers across the model for deeper adjustments.
* **Focus on reconstructive loss:** The loss function must balance reconstructive accuracy so that the VAE ability of producing real samples from the latent space doesn't degrade.

## 7.8.6 Evaluating Fine-Tuned Models

After fine-tuning, how the model is adapting to the new dataset must be verified. In generative AI, both qualitative and quantitative evaluations are adopted.

### Evaluation Metrics

* **Inception Score (IS):** This measures the quality and diversity of the images generated. A large IS indicates that the model is able to generate quite different and realistic images.
* **Frechet Inception Distance (FID):** FID evaluates the similarity of the generated images to real images in terms of feature distributions, offering a more robust form of evaluation than IS, particularly for fidelity in the visuals.
* **Visual Inspection:** Samples generated during training are investigated quite often manually to establish their quality, relevance, and creative fidelity. This becomes crucial where numerical metrics fail to capture the true power of the model when it comes to creative or stylistic applications.

## 7.8.7 Visualization of Output

Visualization is an important tool in checking generative models, which allows better understanding of the latent space and the ability of the model to generate good-quality data.

* **Image Plots:** These plots show the output images with real images for comparison of visual quality and consistency in style.
* **Latent Space Embeddings:** Visualize the embeddings of generated and real data points in latent space to discover clustering or distribution patterns, which reveal whether the generated data closely aligns with the real data.

# References:

* <https://www.weforum.org/agenda/2024/02/green-jobs-green-skills-growth/>
* [Generative AI: A comprehensive guide](https://www.zendesk.com/dk/blog/generative-ai-guide/)
* <https://www.davidinouye.com/course/ece57000-fall-2020/lectures/gans.pdf>
* <https://www.zendesk.com/dk/blog/generative-ai-guide/>
* <https://www.coveo.com/blog/generative-models/>
* <https://www.altexsoft.com/blog/generative-ai/>
* <https://www.coveo.com/blog/generative-models/>
* <https://www.techtarget.com/searchenterpriseai/definition/generative-modeling>
* <https://www.datacamp.com/blog/what-is-a-generative-model>



**Disclaimer: The content is curated from online/offline resources and used for educational purpose only**