# Guide: GAN Deep Learning



## GAN Deep Learning: A Practical Guide

Learn More About Computer Vision

VGG16
ResNet-50
ResNet
GAN Deep Learning
Generative Adversarial Networks

### What Is a Generative Adversarial Network (GAN)?

A Generative Adversarial Network (GAN) is a generative modeling method that automatically learns and discovers patterns in data inputs, generating plausible outputs based on the original dataset.

GANs can train generative models by emulating a supervised approach to learning problems. A GAN contains two sub-models that compete and feed off each other to produce more realistic outputs:

- **The generator model**—trained to generate new outputs.
- **The discriminator model**—classifies inputs as realistic or fake. It attempts to identify whether an input originates from the original dataset of the generator model.

This adversarial approach helps to improve the generator model's capabilities until the discriminator model cannot distinguish between real and generated inputs.

### In this article



## GAN Architecture

The architecture of a GAN consists of two main components. The generator is a neural network that generates data instances, and the discriminator attempts to determine their authenticity. The discriminator model decides if a data instance appears real (i.e., plausibly belongs to the original training data) or fake. The generator model attempts to fool the discriminator and trains on more data to produce plausible results.

This architecture is adversarial because the generator and discriminator work against each other with opposite objectives—one model tries to mimic reality while the other tries to identify fakes. These two components train simultaneously, improving their capabilities over time. They can learn to identify and reproduce complex training data such as image, audio, and video.

The following diagram represents an entire GAN architecture:

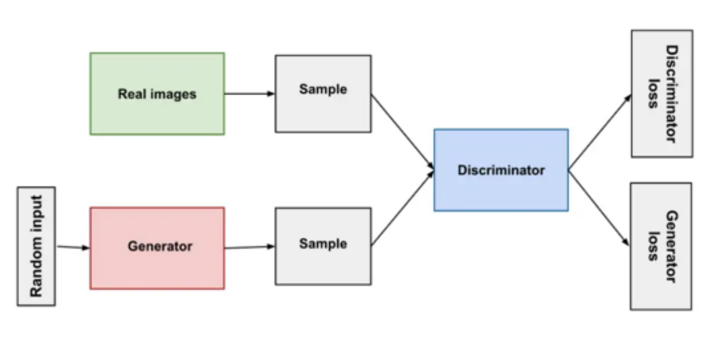



Image Source: developers.google.com

A GAN uses this basic workflow:

1. The generator ingests an input containing random numbers.
2. The generator processes the input to produce an image.
3. The discriminator ingests the image generated by the generator and additional, real images.
4. The discriminator compares the entire image set and attempts to determine which images are real or fake.
5. The discriminator returns a prediction for each image, using a number between 0 and 1 to express the probability of authenticity. A score of 0 indicates a fake image, while 1 indicates a real image.

This workflow creates a continuous feedback loop. The discriminator determines the ground truth (empirical truth) for image inputs, and the generator feeds the discriminator new and improved generated images.

## GANs, Autoencoders and VAEs

Here is a comparison between GANs and other neural network models:

- **GANs** – starts from a random input and learns to create realistic synthetic data.
- **Autoencoders** – encode data inputs as vectors to create a compressed or hidden representation of the original data. Autoencoders are useful for reducing dimensionality. When used alongside decoders, they enable reconstruction of input data based on the associated vector.
- **Variational Autoencoders (VAEs)** – A VAE is a generative algorithm that adds additional constraints to encode data—it normalizes the hidden representations of the input data. VAEs are commonly used to synthesize data similar to a GAN.

## GAN Applications and Use Cases

GANs can produce high-quality, realistic images that are indistinguishable from real photos. For some people, this ability to fake reality is a cause for concern, but generative technology has several important uses.

Various applications for the creation of realistic images include:

- **Face recognition**—the portrait generating capability can also help smartphones recognize their owners' faces in different conditions.
- **Pattern recognition**—a GAN can produce new artwork matching a given artistic style.
- **Content creation**—a cGAN can generate various forms of content to fill in gaps in images or complete presentations. For example: adding facades to buildings, recreating natural landscapes, generating apparel, and rendering fully-furnished interiors.
- **Virtual reality**—the highly detailed HD virtual environments made possible by GANs are useful for simulations and gaming.
- **Unstructured data search**—when used with an unstructured data repository, a GAN can identify similar images based on compressed representations.
- **Predictive imagery**—GANs can be used to simulate aging in images of people.
- **Text-based image generation**—GANs can create new images based on the descriptions in a text. It is possible to train a GAN by labeling GAN-generated images using a supervised learning algorithm. The GAN can then use manually produced text to generate images that match the descriptions.



## What is a Conditional GAN?

A Conditional Generative Adversarial Network (cGAN) is a type of GAN that generates images or other artifacts with conditional settings applied. Both the generator and the discriminator are conditioned on the same auxiliary information, such as class labels or input data.

A successful cGAN model can use this contextual information to learn multimodal mappings. For example, you can condition a cGAN model on a segmentation mask, to generate realistic images that match that mask.

The cGAN architecture has two advantages over traditional GAN. First, it converges faster, because it doesn't start from a completely random distribution. Second, it makes it possible to control the output of the generator by providing a label for the images that the model is expected to generate.

A simple example of cGAN is a GAN model that generates handwritten letters. In a traditional GAN, there is no control over which specific letter the model will generate. However, in a cGAN, you can add an input layer with one-hot encoded image labels, or a feature vector derived from the specific letters the model needs to generate. This guides the model to generate those specific letters.

## What is GAN Inversion?

The objective of GAN inversion is to retrieve a latent vector for a given image to ensure that it produces a near-real image when processed by the generator. Using this latent vector, it is possible to manipulate existing input images rather than randomly generated ones.This means the model is not restricted to using GAN-generated images retained from random samples and image generation.

The following formula describes the inversion problem:

In this context, **z** refers to the latent vector, while **G** refers to a generator. The metric used to describe the distance within the image (at the pixel level) is **l** – for instance, perceptual loss, **l1**, and **l2**.

## Image Editing with GAN

Early GAN architectures did not provide much control over the GAN outputs. There are several approaches that allow GAN users to "edit" or modify the images created by a GAN model. Let's briefly review three prominent attempts at achieving GAN image editing.

### EditGAN

One innovative semantic editing system, *EditGAN*, allows users to edit images with high precision. For example, users can sketch segmentation masks for objects in an image, and EditGAN will modify the masks for more detailed segmentation. It builds on the GAN framework that simultaneously models images and semantic segmentation based on minimal examples of segmentation labels. These capabilities make it a highly scalable editing tool.

EditGAN can embed images into the latent GAN space to conditionally optimize latent code based on the user's segmentation edits, thus modifying the images. The latent space contains editing vectors for amortizing optimization and implementing the edits. This framework lets users learn a set of editing vectors and apply them directly to other images at an interactive rate.

EditGAN uses a high degree of freedom and detail to transform images and preserve their quality. It also supports combining different edits to enable realistic editing beyond the initial training data.

Image Source: *NVIDIA Toronto AI Lab*

*Read the paper*

### GANSpace

GANSpace aims to allow more control over GAN models, by browsing through concepts that GAN networks have learned. It uses Principal Component Analysis (PCA) to understand the latent space for StyleGAN, and the feature space for BigGAN, a model that trains a GAN to generate high resolution, high fidelity images using class-conditional images.

GANSpace modifies BigGAN to allow layer-wise style mixing and control, like in StyleGAN. It enables layer decomposition that provides many controls not ordinarily available in GAN architectures. Users need to perform a one-time labeling effort to identify useful control directions.

Practically, GANSpace gives users control over properties like object pose and shape, as well as nuanced parameters like lighting, face attributes, and landscape attributes.

Image Source: *ResearchGate*

*Read the paper*

### InterFaceGAN

InterFaceGAN is an approach to interpret faces generated by current GAN models, identifying facial semantics encoded in the latent space, and modify them. It works by identifying linear subspaces in the GAN latent space, and realistically manipulating the facial attributes that correspond to those subspaces, without needing to retrain the model.

This technique makes it possible to manipulate the gender, age, facial expression, and accessories like eyeglasses or earrings. When a face is generated with the wrong artifacts, it can be used to correct the resulting images. This makes it possible to synthesize faces in a predictable, controllable process.

Image Source: *GenForce*

*Read the paper*

## GAN Libraries for Deep Learning

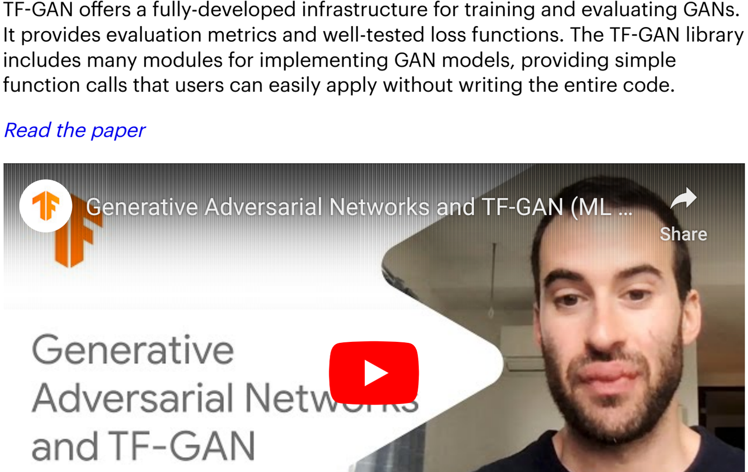
Here are some examples of libraries that provide GAN tools for deep learning use cases:

### TensorFlow-GAN (TF-GAN)

TensorFlow-GAN is a lightweight, open source python library developed by Google. It aims to facilitate the implementation of generative adversarial networks.

TF-GAN offers a fully-developed infrastructure for training and evaluating GANs. It provides evaluation metrics and well-tested loss functions. The TF-GAN library includes many modules for implementing GAN models, providing simple function calls that users can easily apply without writing the entire code.

*Read the paper*




### TorchGAN

TorchGAN is a Pytorch-based GAN design and development framework. It provides the basic building blocks for common GANs and allows users to customize their models for advanced research. The TorchGAN modular structure enables users to test popular GANs on their data and plugin new architectures and loss functions with older versions. It provides various logging backends to help visualize training projects.

*Source*

*Read the paper*

### Mimicry

Mimicry is a lightweight PyTorch-based library that emphasizes the reproducibility of GAN-related research. It helps users compare GANS and navigate the nuances in different offerings' implementation and evaluation methods, which may affect performance.

Mimicry provides the following features:

- Standardized implementations of widely used GANs reproducing reported scores closely.
- Baseline GAN scores based on training and evaluation in comparable conditions.
- A framework to help researchers implement GANs without rewriting the entire boilerplate code for GAN training.
- Support for various metrics to evaluate GANs.

Mimicry provides baselines and a model zoo to compare various GAN models of identical size, trained under identical conditions. It uses various metrics to evaluate popular GANs and ensure model reproducibility. Mimicry verifies the scores of GAN implementations models based on a library of reported scores.

*Read the paper*

### GAN Lab

GAN Lab is an interactive, user-friendly tool for visualizing and experimenting with GANs. It allows users to visualize the inner workings of GAN models while training them for 2D data distributions.

GAN Lab uses TensorFlow.js, a GPU-accelerated, in-browser deep learning library. It uses JavaScript to implement the entire GAN experiment, including visualization and training. Users can run GAN Lab with just a web browser. This implementation approach makes deep learning more accessible to the general public.

Source: GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation. ***Read the paper***

*Explore Datagen's Platform with Our Free Trial. Start Now!*