Deep Learning and Neural Networks

Demetrio Labate

April 8, 2026

Generative Modeling

Why generative modeling?

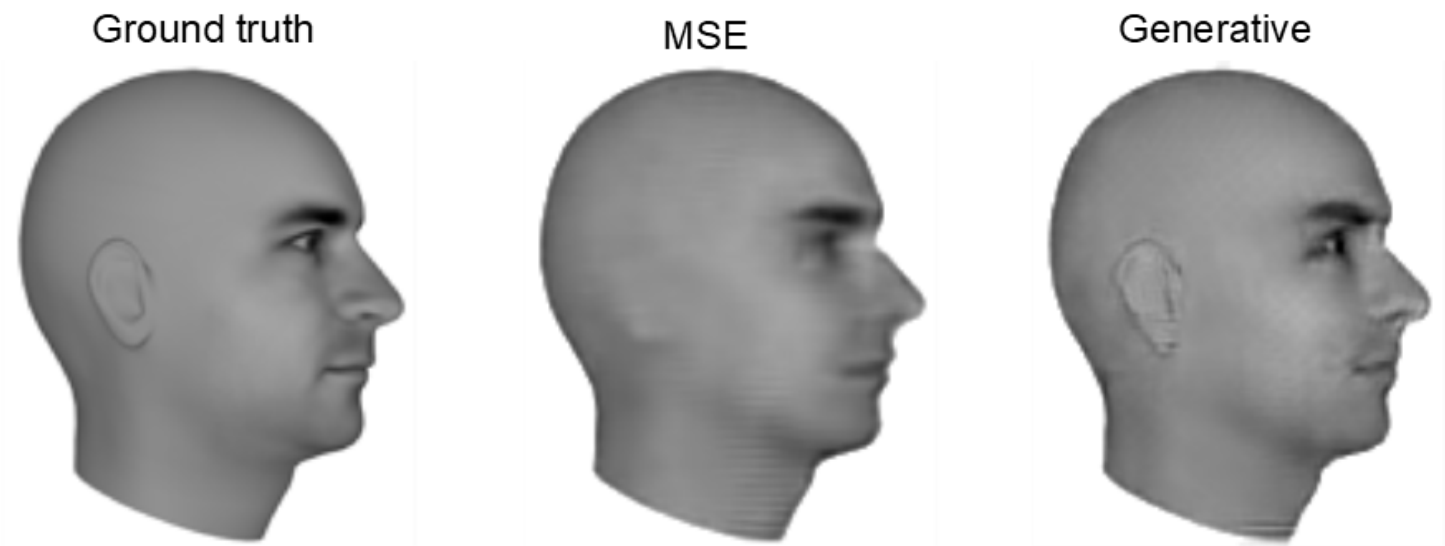
- **Understanding high-dimensional probability distributions.** Training and sampling from generative models can be used to test of our ability to represent and manipulate high-dimensional probability distributions. High-dimensional probability distributions are important objects in a wide variety of applied mathematics and engineering domains.
- **Missing data.** Generative models can be trained with missing data and can provide predictions on inputs that are missing data. One particularly interesting case of missing data is semi-supervised learning, in which the labels for many or even most training examples are missing.

Generative Modeling

- **Single image super-resolution.** The goal of super-resolution is to take a low-resolution image and synthesize a high-resolution equivalent.
  There are many possible high-resolution images corresponding to the low-resolution image. Choosing an image that is the average of all possible images would yield a blurry result. A generative model can choose an image that is a sample from the probability distribution over possible images.

Generative Modeling

- **Multi-modal outputs.** For many tasks, a single input may correspond to many different correct answers, each of which is acceptable. Traditional training machine learning models, such as minimizing the mean squared error between a desired output and the model's predicted output, are unable to train models that can produce multiple different correct answers. An example of such a scenario is the prediction of the next frame in a video.



GANs

Generative Adversarial Networks (GANs) were invented by Goodfellow et al. in 2014 and first described in the paper:

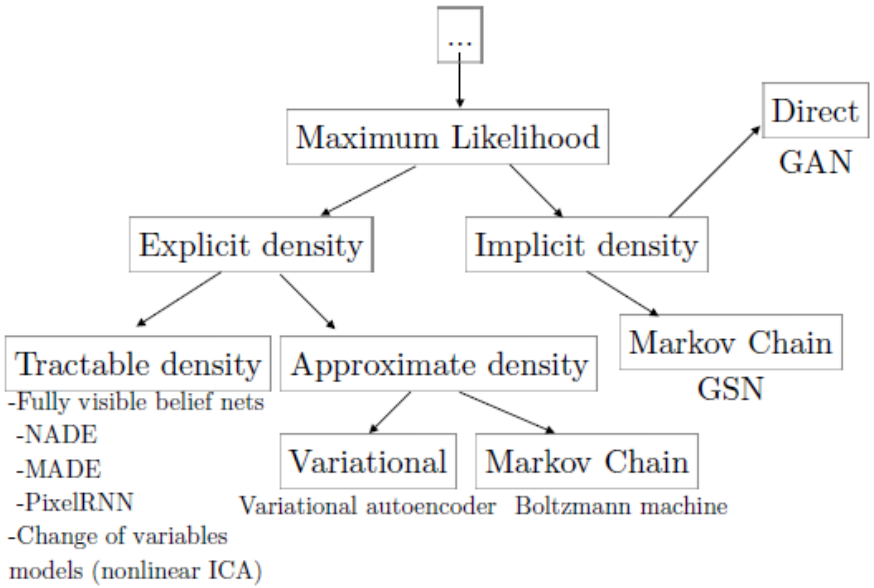
I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems, 2014, pp. 2672–2680

It is a framework for teaching a deep learning model to capture the training data distribution so we can generate new data from that same distribution.

Unlike other generative models (e.g., RBM) which aim to **explicitly** learn the distributions of data, GANs learn to capture the statistical distribution of training data **implicitly**.

GANs

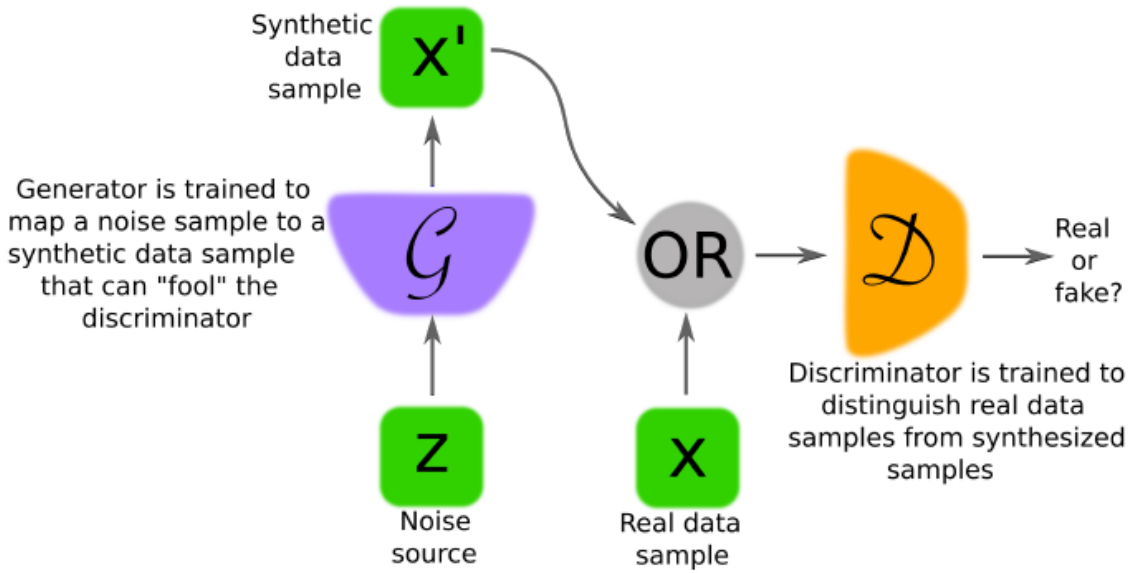
Deep generative models that can learn via the principle of maximimum likelihood differ with respect to how they represent or approximate the likelihood.



On the left branch, models construct an explicit density and an explicit likelihood which can be maximized. On the right, models do not explicitly represent a probability distribution but provide some way of interacting less directly with this probability distribution, e.g., by drawing samples.

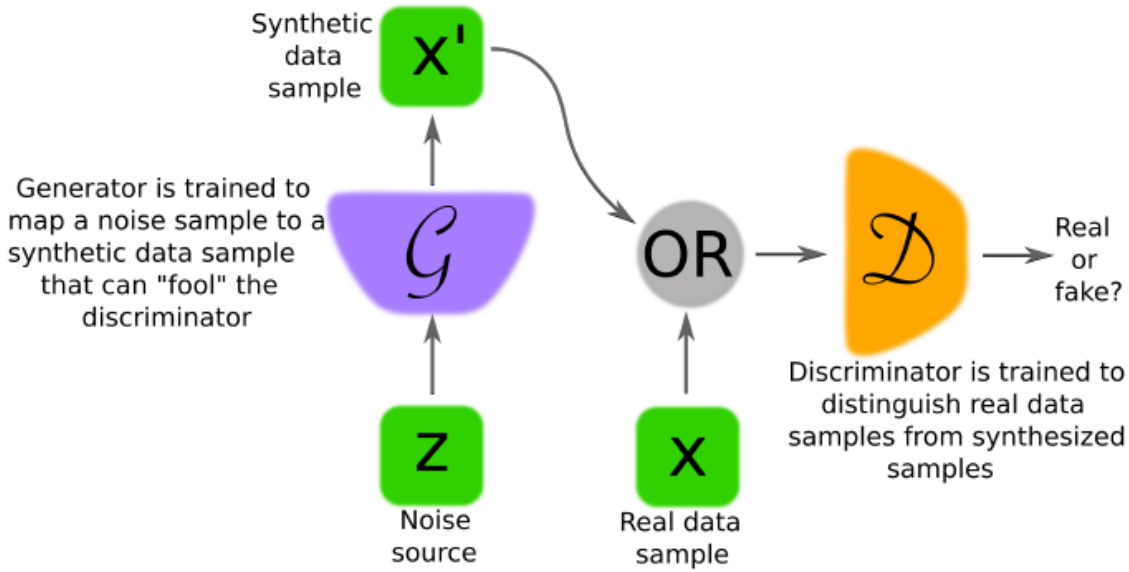
GANs

GANs are made of two distinct models, a **generator** and a **discriminator**, typically implemented with neural networks.



The job of the generator is to spawn 'fake' images that look like the training images.
The job of the discriminator is to look at an image and output whether or not it is a real training image or a fake image from the generator.

GANs



During training, the generator is trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to become a better detective and correctly classify real and fake images.
The equilibrium of this game is when the generator is generating perfect fakes and the discriminator is left to always guess at 50% confidence that the generator output is real or fake.

GAN Architectures

Fully Connected GANs

The first GAN architectures used fully connected neural networks for both the generator and discriminator [Goodfellow et al, 2014]

This type of architecture was applied to relatively simple image datasets, namely MNIST (hand written digits), CIFAR-10 (natural images) and the Toronto Face Database.

GAN Architectures

Convolutional GANs

Given that CNNs are extremely well suited to image data, it was natural to use CNN architectures with GANs.

Early experiments conducted on CIFAR-10 suggested that it was more difficult to train generator and discriminator networks using CNNs with the same level of capacity and representational power as the ones used for supervised learning.

Radford et al. in 2016 proposed a family of network architectures called DCGAN (Deep Convolutional GAN) which allows training a pair of deep convolutional generator and discriminator networks.

GANs Theory

We can model the generator network as mapping from some representation space, called a latent space, to the space of the data

$$\mathcal{G} : \mathcal{G}(z) \mapsto \mathbb{R}^N$$

where $z \in \mathbb{R}^m$ is a sample from the latent space and x is data point, e.g., an image.

The discriminator network, D, may be similarly characterized as a function that maps from image data to a probability that the image is from the real data distribution, rather than the generator distribution:

$$\mathcal{D} : \mathcal{D}(x) \mapsto [0, 1]$$

GANs Theory

For a fixed generator, $\mathcal{G}$, the discriminator, $\mathcal{D}$, is trained to classify images as either being from the training data (real, close to 1) or from a fixed generator (fake, close to 0).

When the discriminator is optimal, it can be frozen, and the generator, $\mathcal{G}$ can continue to be trained to lower the accuracy of the discriminator.
If the generator distribution is able to match the real data distribution perfectly, then the discriminator will be maximally confused, predicting 0.5 for all inputs. In practice, the discriminator might not be trained until it is optimal.

GANs Training

Training of GANs involves both finding the parameters of a discriminator that maximize its classification accuracy, and finding the parameters of a generator that maximally confuse the discriminator.

The cost of training is evaluated using a value function, $V(\mathcal{G}, \mathcal{D})$ that depends on both the generator and the discriminator

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D})$$

where

$$V(\mathcal{G}, \mathcal{D}) = E_{p_{data}} \log \mathcal{D}(x) + E_g \log(1 - \mathcal{D}(\mathcal{G}(z))$$

and p_{data} is the data distribution (represented by samples), p_g is the model distribution, modeled as Gaussian.

Note: the generator and discriminator networks must be differentiable.

GANs Training

Min-max value explanation

For the discriminator $\mathcal{D}$ to be accurate, we maximize

$$E_{p_{data}} \log \mathcal{D}(x)$$

$\mathcal{G}$ is trained to increase the probability of $\mathcal{D}$ for a fake image, that is, we minimize

$$E_g \log(1 - \mathcal{D}(\mathcal{G}(z))$$

Together, $\mathcal{D}$ and $\mathcal{G}$ are playing a minimax game

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \{V(\mathcal{G}, \mathcal{D}) = E_{p_{data}} \log \mathcal{D}(x) + E_g \log(1 - \mathcal{D}(\mathcal{G}(z))\}$$

GANs Training

During training, the parameters of one model are updated, while the parameters of the other are fixed.

Theory: Goodfellow et al. (2014) show that for a fixed generator **there is a unique optimal discriminator**,

$$\mathcal{D}^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \in [0, 1]$$

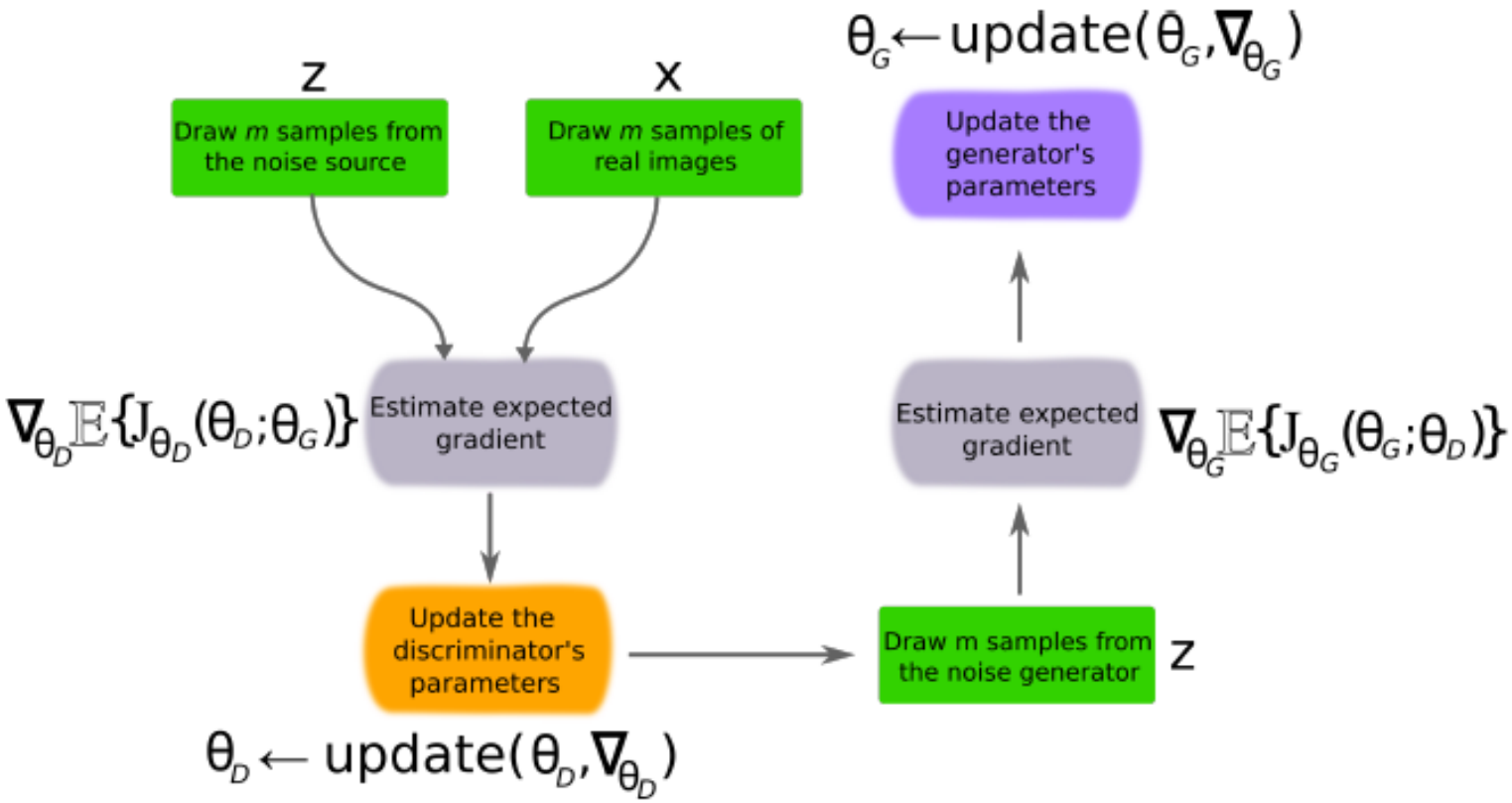
They also show that the generator, $\mathcal{G}$, is optimal when

$$p_{data}(x) = p_g(x)$$

which implies that the optimal discriminator $\mathcal{D}^*(x)$ predicts 0.5 for all samples drawn from x.

That is, the generator is optimal when the discriminator, $\mathcal{D}$, cannot distinguish real samples from fake ones.

GANs Training



Main loop of GAN training. Novel data samples, x', are drawn by passing random samples, z, through the generator network $\mathcal{G}$. Training alternates between k steps of optimizing $\mathcal{D}$ and one step of optimizing $\mathcal{G}$.

GANs Training

Ideally, the discriminator $\mathcal{D}$ is trained until optimality with respect to the current generator; next, the generator is again updated.

However, in practice, the discriminator $\mathcal{D}$ might not be trained until optimal, but rather may only be trained for a small number of iterations, and the generator is updated simultaneously with the discriminator.

Despite the theoretical existence of unique solutions,
GAN training is challenging and often unstable because it requires finding a delicate balance (Nash equilibrium) between two competing neural networks - the generator and discriminator - rather than converging to a single loss minimum.

Note: theory requires that $\mathcal{D}$ and $\mathcal{G}$ have enough capacity.

GANs Training

Common manifestations of GAN Instability

- **Vanishing Gradients:** If the discriminator becomes too "perfect," it rejects all generated samples entirely, leading to little or no feedback (gradients) for the generator to learn.
- **Mode Collapse:** This occurs when the generator finds a limited set of samples that fool the discriminator and fails to produce a diverse range of outputs, focusing on just one or a few modes of the data.
- **Non-convergence:** The generator and discriminator may constantly chase each other, with their losses oscillating wildly instead of reaching a stable equilibrium.

GANs Training

Common Solutions and Mitigations

- **Gradient Penalties:** Regularization techniques, such as weight-clipping or gradient penalties, prevent the discriminator from becoming too powerful.
- **Updating Frequencies:** Sometimes, updating the discriminator more or less frequently than the generator can help restore balance.
- **Wasserstein Loss (WGAN)** Uses a different loss function to provide useful gradients even when the discriminator is strong.

GANs

GANs PyTorch Tutorial