What a Generative Models?

Yuyang Zhang

2025-04-30

Contents

1 What is the Generative Models and Generative AI?

Generative models, as the name indicated, are models that can *generative* new content. Unlike **discriminate models**, the generative models are sometime hard to train. But why we need generative models in the first place? We want generative models because:

- Density Estimation: Estimate the probability density function of the data.
- Anomaly Detection: Detect the anomaly data points.
- Imputation: Fill in the missing data.
- Data Augmentation: Generate new data to increase the size of the dataset.
- Data Generation: Generate new data to train the model.
- Data Compression: Compress the data to save the storage.
- Data Denoising: Remove the noise from the data.
- Latent Space Exploration: Explore the latent space of the data.
- Latent Space Interpolation: Interpolate between the data points.

Generative Models can solve **inverse problems**. For example, the medical image reconstruction. Herea re some example of the generative models in the real world:

- Text to Image Model: this is common in the current AI, for example, the Stable Diffusion Model and Dalles model, below are the example of the generation of Chat-4o model:
- Text to Video model such as Sora.

In the blog, we will learn three things:

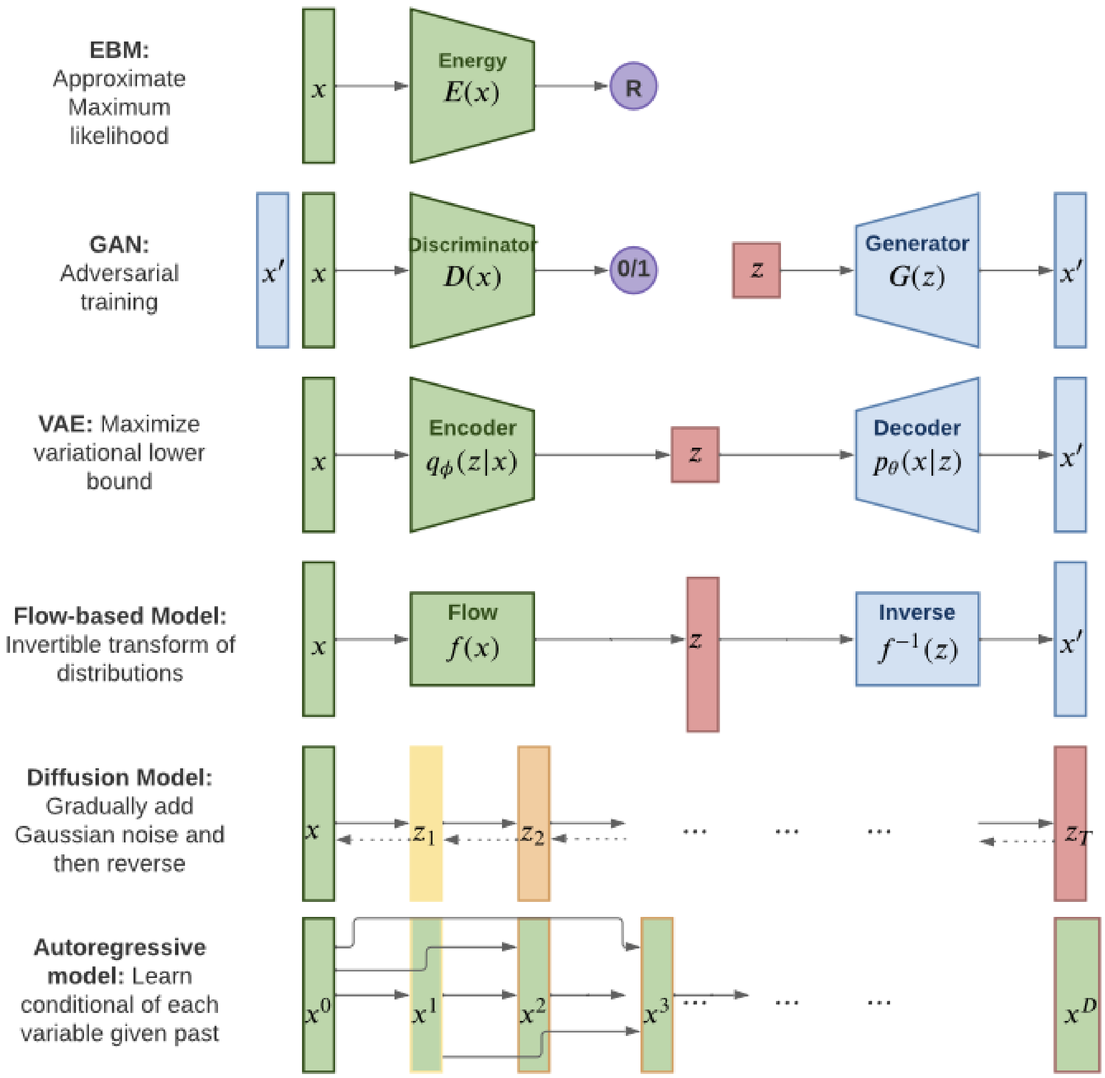


Figure 1: Summary of various kinds of deep generative models. (Image Source: Probabilistic Machine Learning)

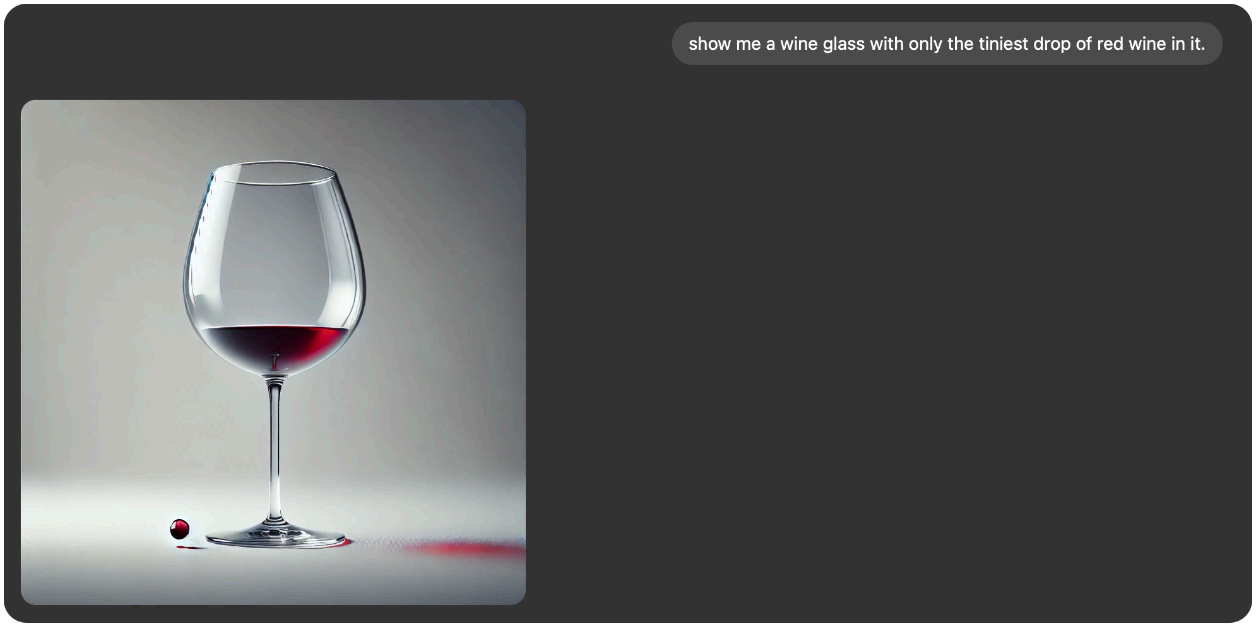


Figure 2: Image Generation through ChatGPT-4o model.

images/sora-generation.mp4

Figure 3

- Representation: how to model the joint distribution of many random variables
- Learning: how to learn and compare the different probability distribution
- Inference: how to invert the generation process (recover high-level description (latent variables) from raw data(images, text...)

Besides the three main topics, we will introduce 6 different generative models as showed in the Figure **??**. In this article, we will go through those 6 different types. Get into the details of each different types and compare models. How to combine those model to get more complex modeling ability.

Note

In this blog, we only going through the *main ideas* of the different models, if you want to dig into different topics more deeply, please check my following blogs:

- Variational AutoEncoder Models
- Diffusion Models
- Auto-Regressive Models (Coming soon...)
- Flow Matching(Coming soon...)

- GAN(Coming soon...)

2 Maximum Likelihood Learning

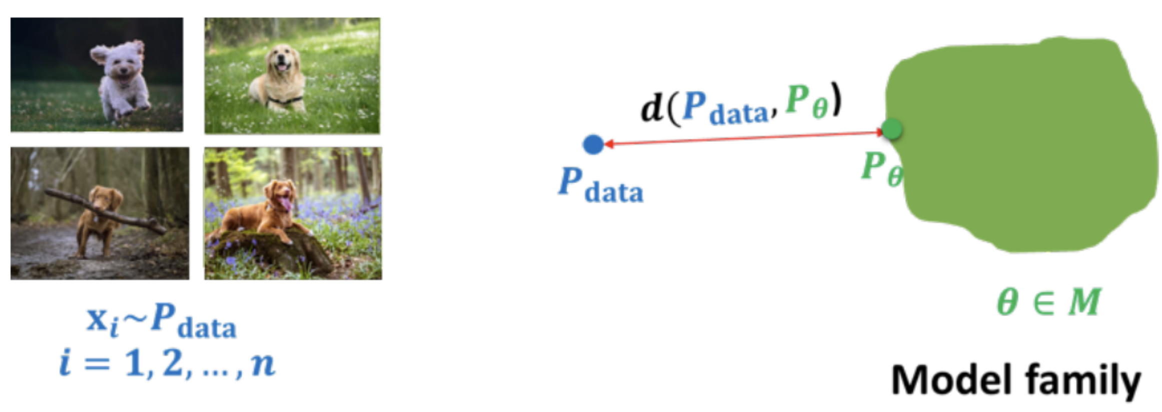


Figure 4: Compare modelled distribution with true distribution (Image Source: Stanford CS236 Deep Generative)

The purpose of the most generative models is to learn the probability distribution P_θ that is close to the true distribution P_{data} which we don't know. There are different form of the P_θ, how do we choose the "best" model to represent the P_{data}. We can use the Kullback-Leibler divergence (KL-divergence) between two distribution to measure how different those two distributions are. The KL-Divergence is defined as:

$$\begin{aligned} D_{KL}(P \| Q) &= \mathbb{E}_P\left[\log \frac{P(x)}{Q(x)}\right] \\ &= \mathbb{E}_P[\log P(x)] - \mathbb{E}_P[\log Q(x)] \end{aligned} \tag{1}$$

Two of the good property of the KL-Divergence are:

- $D_{KL} \geq 0$: when $Q = P$ we can get the equal
- $\mathbb{E}_P\left[-\log \frac{Q}{P}\right] \geq -\log \mathbb{E}_P\left[\frac{Q}{P}\right]$: due to theJensen's inequality and $-\log$ is the convex function.

So, we can measure how the P_{data} and P_θ are different:

$$
\begin{aligned}
D_{KL}(P_{\text{data}} \| P_\theta) &= \mathbb{E}_{x\sim P_{\text{data}}}\left[\log \frac{P_{data}(x)}{P_\theta(x)}\right] \\
&= \mathbb{E}_{x\sim P_{\text{data}}}[\log P_{\text{data}}(x)] - \mathbb{E}_{x\sim P_{\text{data}}}[\log P_\theta(x)]
\end{aligned} \tag{2}
$$

As we can see, the first term is not related to the θ, which means const across all the models and we don't know the value. We only need to **maximizing** the $\mathbb{E}_{x\sim P_{\text{data}}}[\log P_\theta(x)]$ in order to minimizing the D_{KL}.

> **Note**
>
> Notes that, although we can compare models, we are still not know how close we are to the true distribution P_{data}.

How to calculate $\mathbb{E}_{x\sim P_{\text{data}}}[\log P_\theta(x)]$, one way is to approximate the expected log-likelihood with empirical log-likelihood:

$$
\mathbb{E}_{x\sim P_{\text{data}}}[\log P_\theta(x)] \approx \mathbb{E}_{\mathcal{D}}[\log P_\theta(x)] = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \log P_\theta(x)
$$

So, the maximum likelihood learning become:

$$
\max_{P_\theta} \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \log P_\theta(x)
$$

The maximum of the likelihood function is equal to minimizing the **negative log-likelihood(NLL)**:

$$
\max_{P_\theta} \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \log P_\theta(x) \quad \Longleftrightarrow \quad \min_{P_\theta} \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} -\log P_\theta(x)
$$

So, the loss function is:

$$
\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} -\log P_\theta(x)
$$

Depending on what form $p_\theta(x)$ takes, this NLL simplifies into familiar losses like **MSE** or **CrossEntropy**.

When translating the above mathematical formulation into code, we commonly used some form:

- Mean Squared Loss (MSE): Diffusion Models
- Cross-Entropy Loss: GAN(Binary Cross Entropy)

i Why we can convert MLE to loss fucn

Assume the model outputs the **mean** of a Gaussian distribution:

$$p_\theta(x) = \mathcal{N}(x; \mu_\theta, \sigma^2 I)$$

Then the log-likelihood is:

$$\log p_\theta(x) = -\frac{1}{2\sigma^2}\|x - \mu_\theta\|^2 + \text{const}$$

So the **negative log-likelihood** becomes:

$$-\log p_\theta(x) = \frac{1}{2\sigma^2}\|x - \mu_\theta\|^2 + \text{const}$$

One of the problem with the MLE is that, it can easily overfit the data, that means it might not generalize well on the un-seen data set. There are several way to avoid the overfitting:

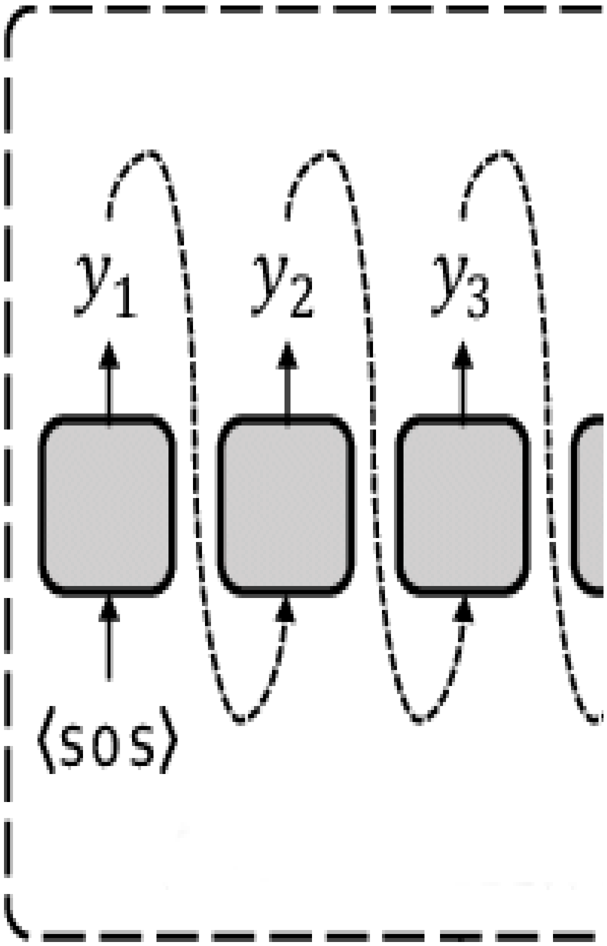
- Hard Constraints: limit the choice of the NN
- Soft preference for "Simpler" Models
- Augment the objective function with regularization

$$\text{obj}(x, \mathcal{M}) = \text{loss}(x, \mathcal{M}) + R(\mathcal{M})$$

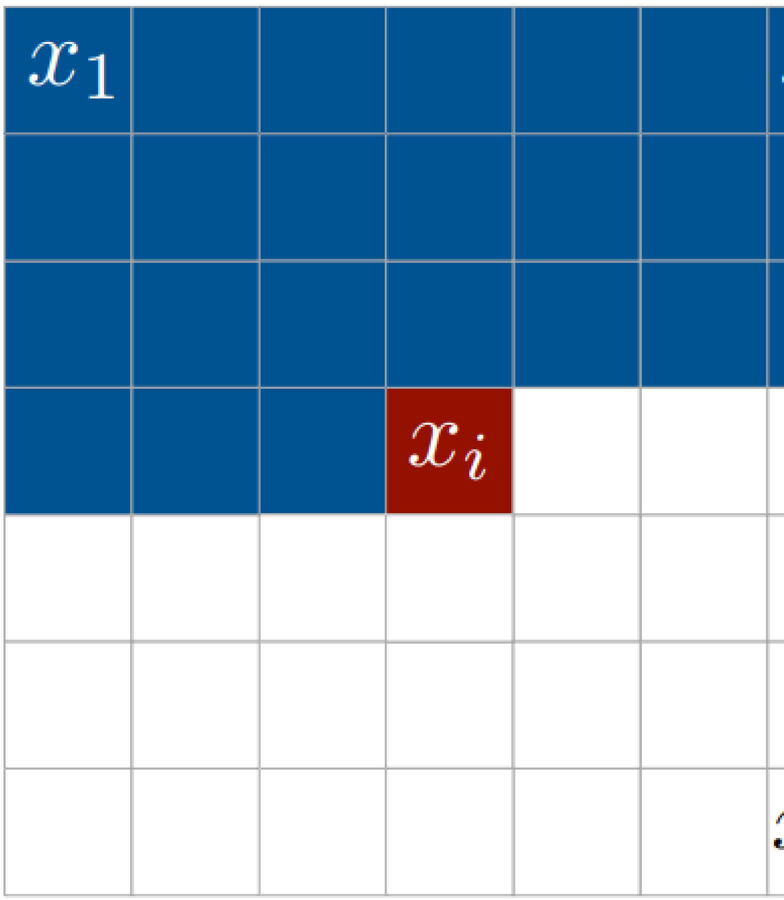
- Evaluate generalization performance on a held-out validation set.

3 Auto-Regressive Models

Theorem 3.1 (Auto Regressive Model)**.** *An* ***auto-regressive generative model*** *is a type of* ***generative model*** *that models the* ***joint probability distribution*** *of a sequence (e.g., words, pixels, audio samples) by factorizing it into a* ***product of conditional probabilities****—each conditioned on the previous elements Mathematically, given a sequence*



(a) Auto-Regressive Model of Language Model



(a) PixelCNN for the Image Modeling