Generative AI

Generative vs Discriminative Models

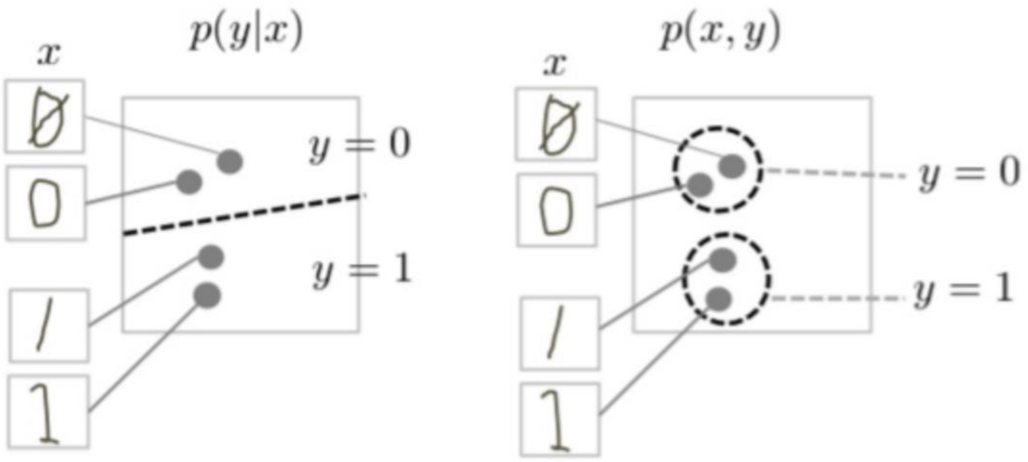
or just $P(\boldsymbol{X})$ → unsupervised (or self-supervised) learning

generative models: predict joint probability $P(Y, \boldsymbol{X})$ (what allows to create new data samples) or directly generate new data samples

discriminative models: predict conditional probability (or probability distribution for regression) $P(Y|\boldsymbol{X})$ or directly output (label for classification, real value for regression)

task of generative models more difficult: model full data distribution rather than merely find patterns in inputs to distinguish outputs

discriminative model generative model



source

Data Generation

generative models can be used for discriminative tasks (although potentially inferior to direct discriminative methods)

but generative methods do more than discriminative ones: model full data distribution

→ allows generation of new data samples (can be images, text, video, audio, code like SQL or Python, proteins, materials, time series, structured data, …)

large (auto-regressive) language models examples of generative models

Deep Learning for Generative AI

Depending on the application, there are currently two dominant approaches for generative AI:

- text generation: LLMs
- image synthesis: diffusion models (usually conditioned on text by transformers)

note the difference between image synthesis and multimodal understanding in LLMs

Image Synthesis

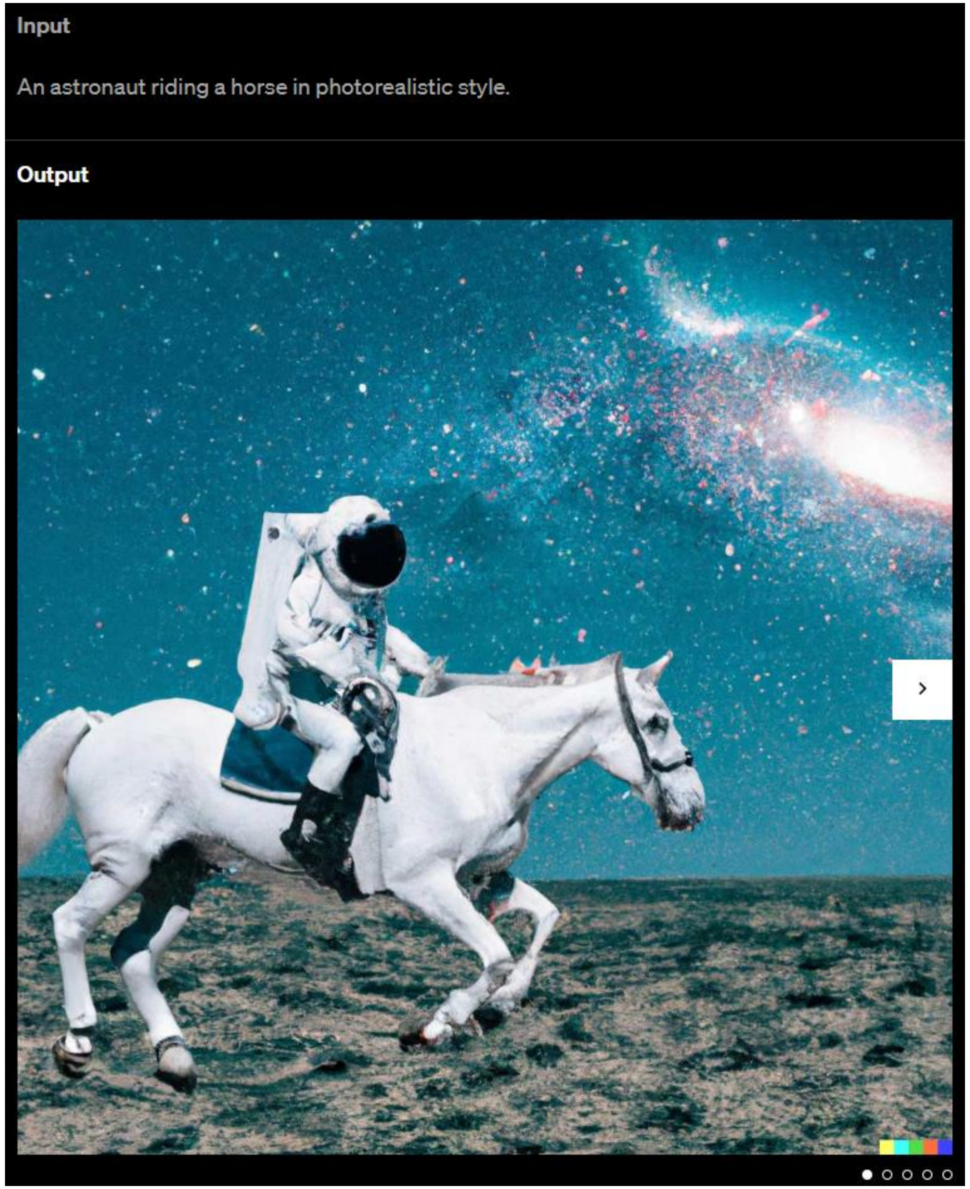
idea: generate new images as variations of training data

condition generation on text prompts: text-to-image

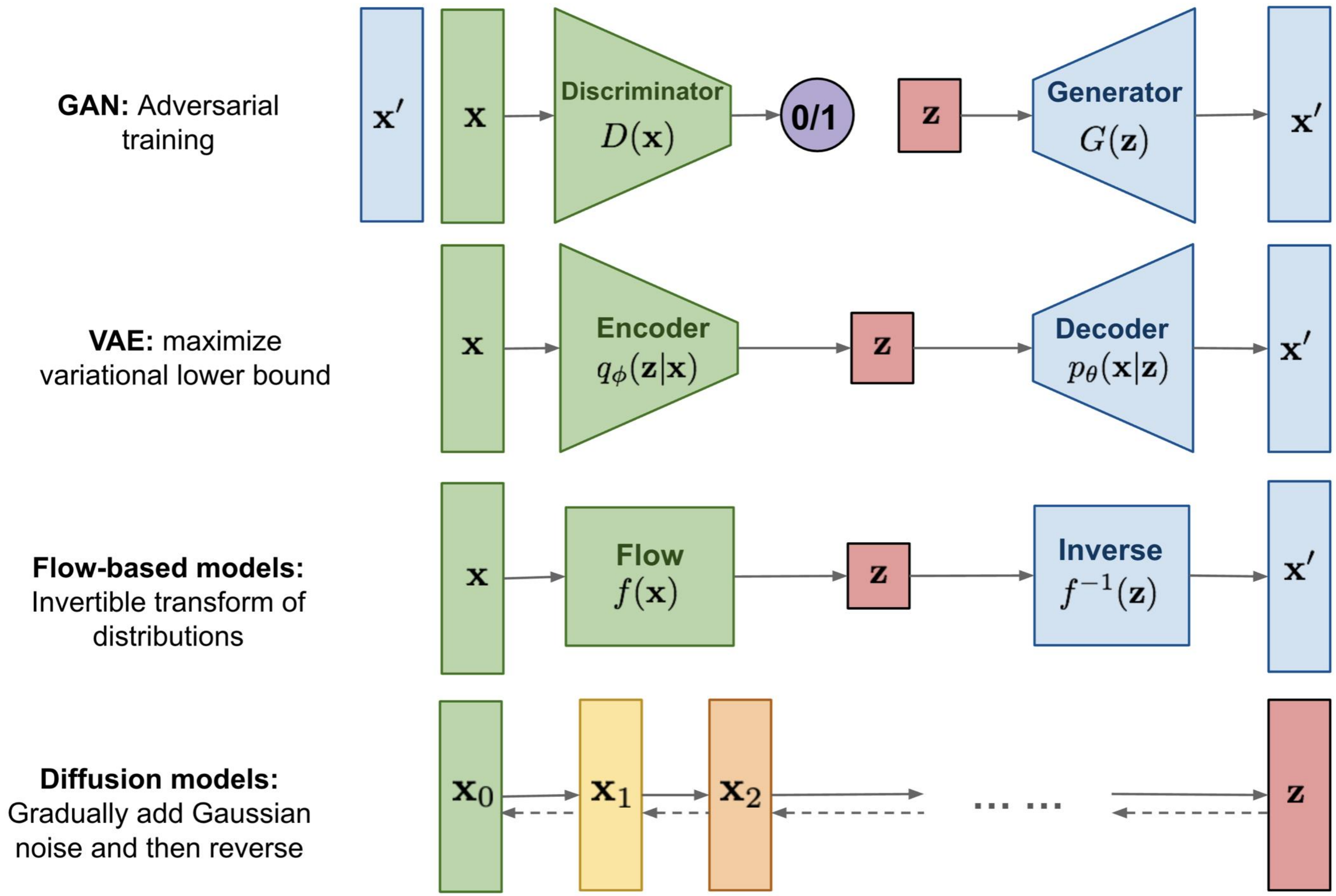
trade-off between diversity and fidelity

SOTA: (guided) diffusion models

example: DALL-E 2



Different Model Types for Image Synthesis



two neural networks playing a zero-sum game

learn variational distribution (not just replicating inputs)

more complex distributions by applying change-of-variable technique (need for specialized architecture)

chain of denoising autoencoders

source

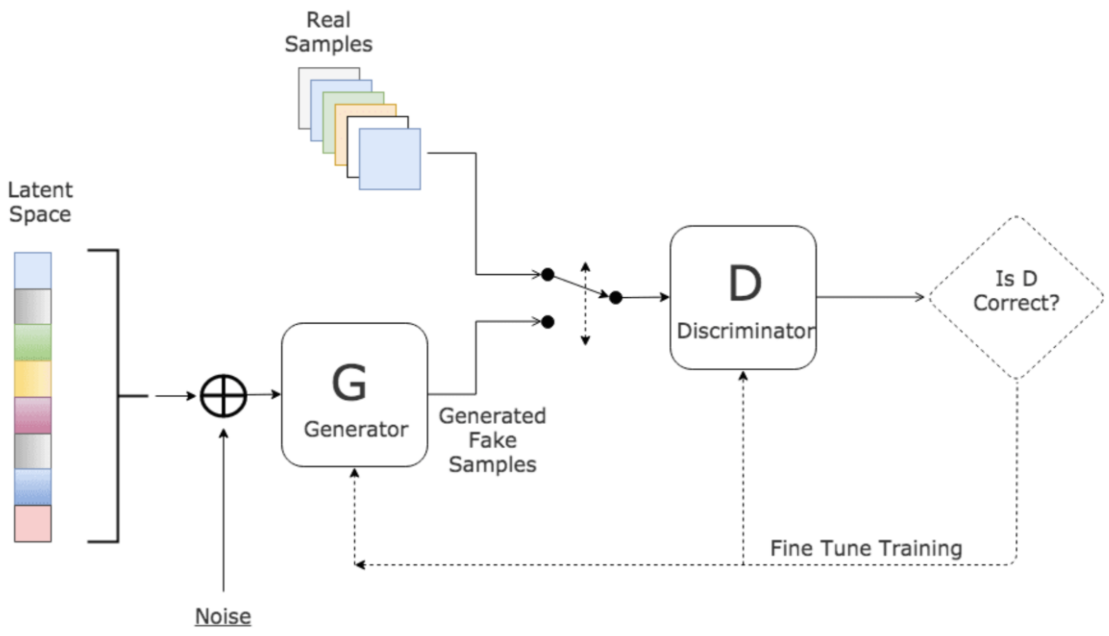
→ generalization: flow matching

Generative Adversarial Networks (GAN)

two neural networks playing a zero-sum game:

- the generator network G generating new (fake) samples
- the discriminator network D trying to distinguish between real and fake samples

indirect training via D: G not trained directly to minimize reconstruction error of real samples, but to fool D → self-supervised approach



source

common loss for generator and discriminator:

$$L(\boldsymbol{x}_i) = E_{\boldsymbol{x} \sim p_r(\boldsymbol{x})}[\ln D(\boldsymbol{x}_i)] + E_{\boldsymbol{x} \sim p_g(\boldsymbol{x})}\left[\ln\left(1 - D(\boldsymbol{x}_i)\right)\right]$$

G trying to minimize

D trying to maximize

Side Note: Autoencoder

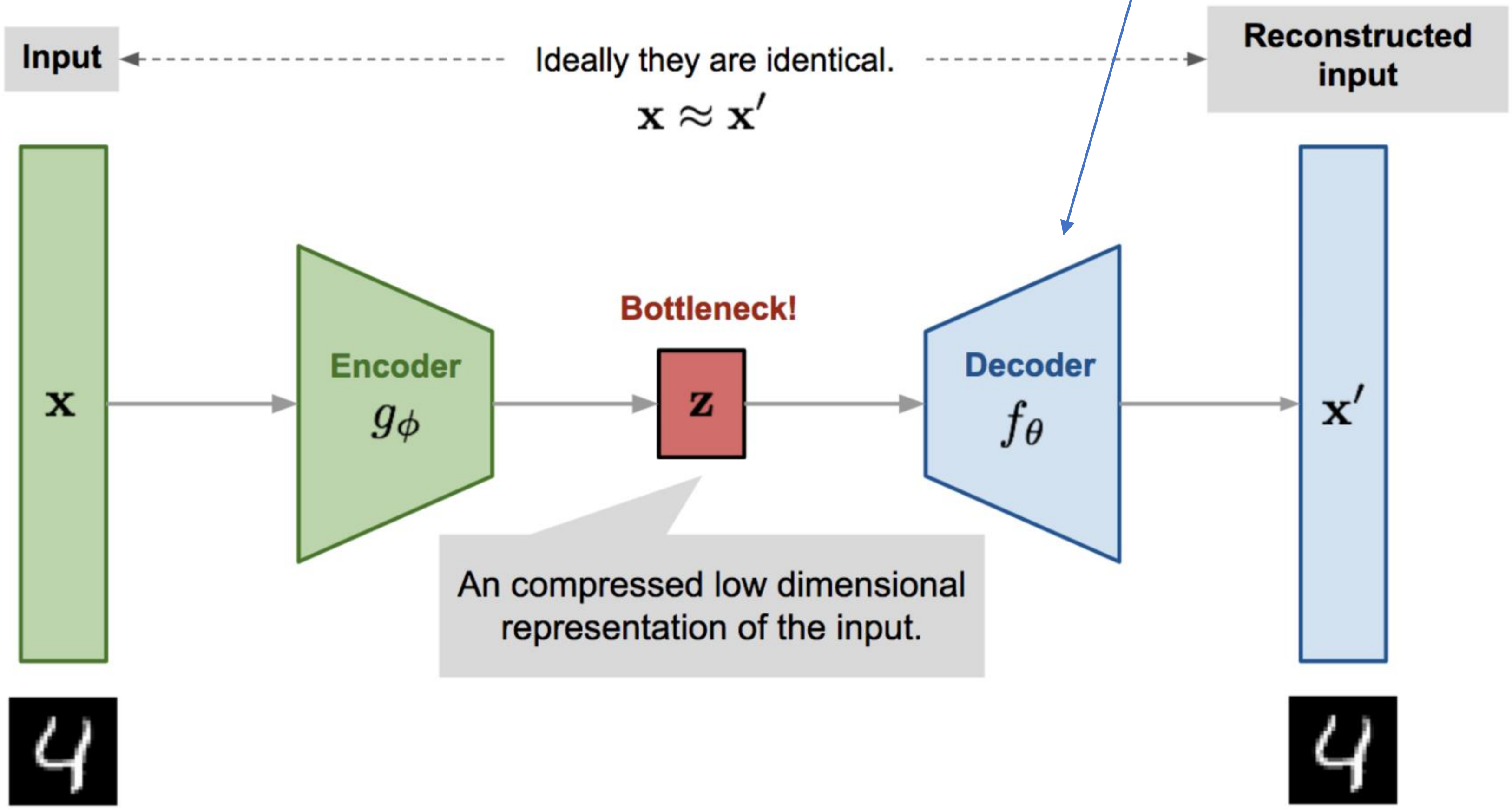
up-sampling, for example,
by transposed convolutions

(deep) encoder network

(deep) decoder network

learned together by minimizing differences between original input and reconstructed input (expressed as losses)

compressed intermediate representation: dimensionality reduction



source

Variational Autoencoder (VAE)

goal: generation of variations of input data rather than compressed representation

→ learn variational distribution instead of identity function

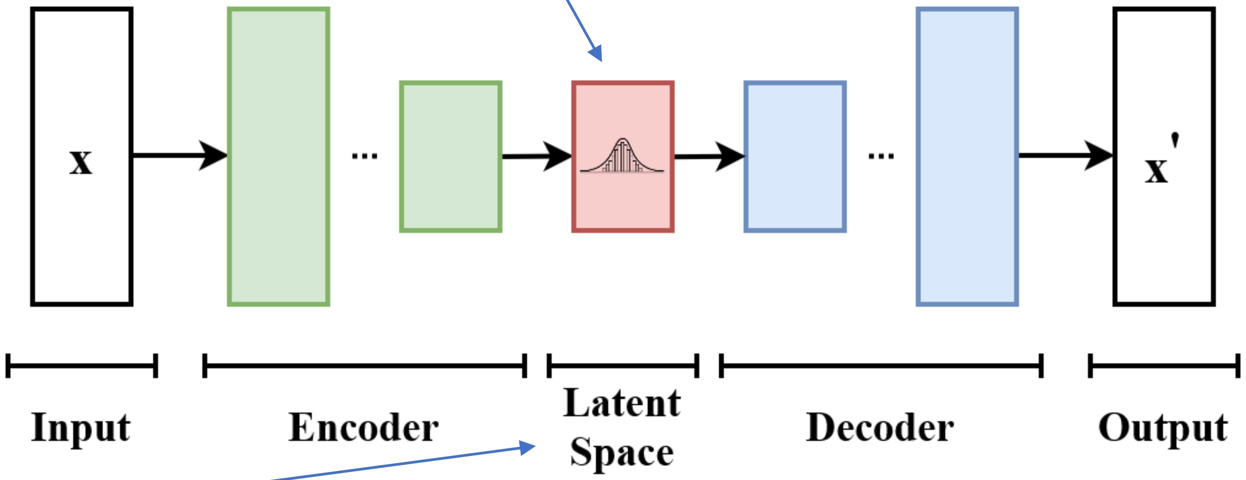
to be precise: parametrized variational distribution of latent encoding variables $\boldsymbol{z}$

prior (simple distribution, in usual VAE: Gaussian): $p_{\boldsymbol{\theta}}(\boldsymbol{z})$

posterior: $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{z})}{\int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{z})d\boldsymbol{z}}$

$p_{\boldsymbol{\theta}}(\boldsymbol{x})$: mixture of Gaussians

from which to sample

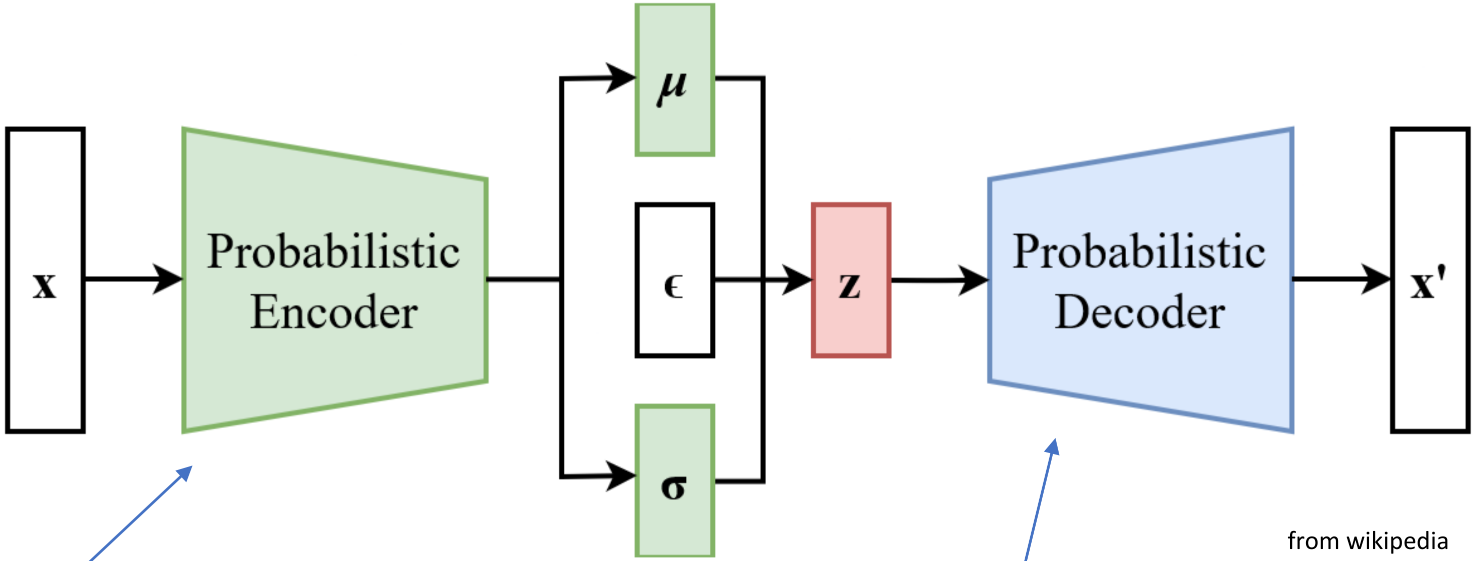


from wikipedia

Variational Bayesian Method

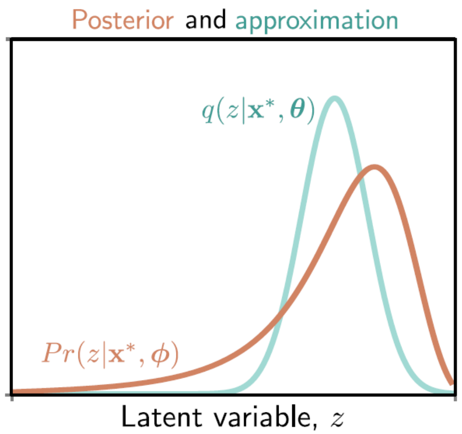
Gaussian Approximation

learn mean and variance of multivariate Gaussian with diagonal covariance structure

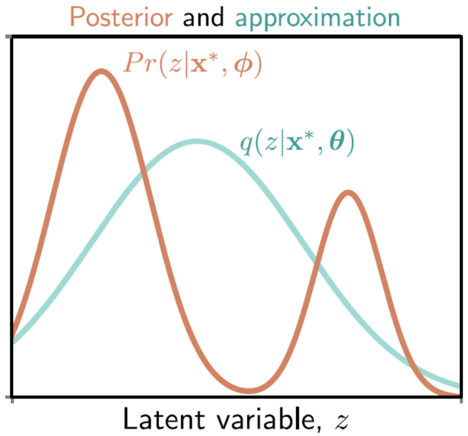


from wikipedia

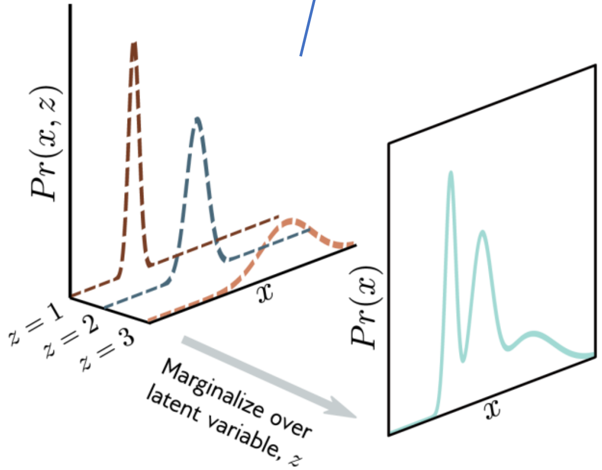
good approximation:



poor approximation:



source

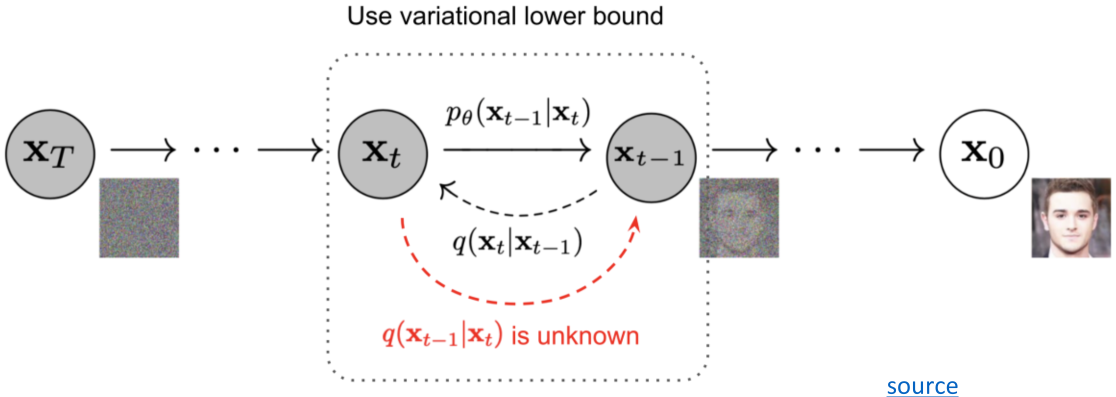


AlphaFold 3 uses diffusion-based architecture for protein structure prediction

Diffusion

training: distort training data by successively adding random noise, then learn to reverse this process (denoising)

generation: sample random noise and run through the learned denoising process



source

advantages: easy to train, produce high-quality/realistic samples

can be interpreted as special case of hierarchical VAE (one latent variable generates another) with fixed encoder and latent space of same size as the data

→ more sophisticated latent space than just Gaussian mixture in VAE

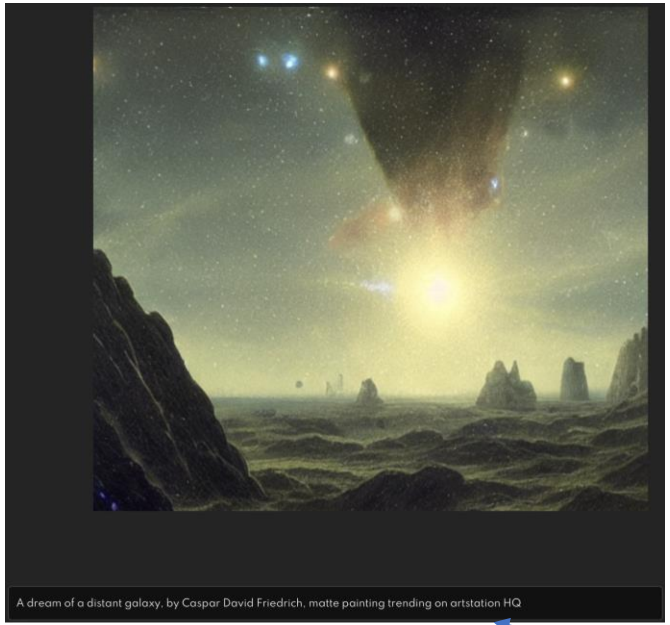
Text-to-Image

plenty of applications: DALL-E, Stable Diffusion, ImageGen, Midjourney, ...

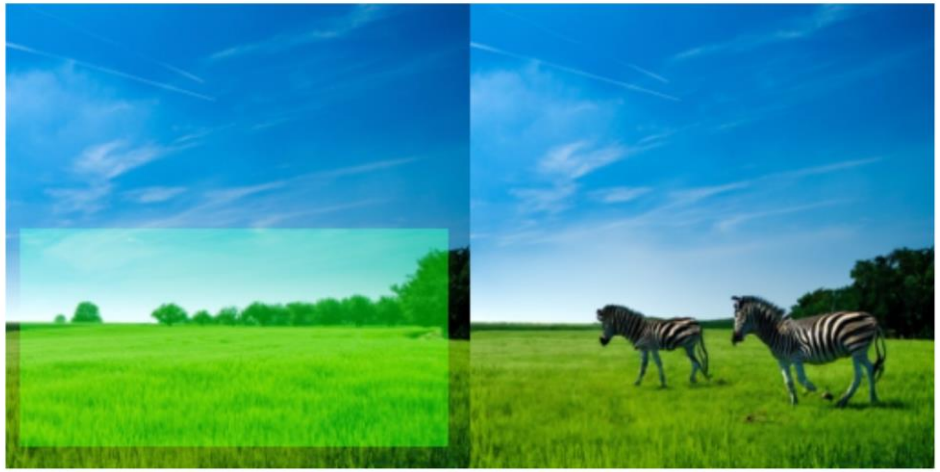
rather "translations"

also text-to-speech (VALL-E, Speech T5, ...) (and speech recognition, e.g., Whisper), text-to-video (Make-A-Video, Lumiere, Sora, ...) → dynamics/physics understanding/simulation

web app for Stable Diffusion: DreamStudio



inpainting example (GLIDE):



"zebras roaming in the field"

source

prompt