Generative Models

Discriminative vs Generative

Understanding Machine Learning

Archetype: Naïve Bayes

probabilistic model:

$$P(Y|X_1,\cdots,X_p) = \frac{P(Y,X_1,\cdots,X_p)}{P(X_1,\cdots,X_p)} = \frac{P(Y)P(X_1,\cdots,X_p|Y)}{P(X_1,\cdots,X_p)} \propto P(Y)P(X_1,\cdots,X_p|Y)$$

Bayes' rule　　constant　　to be estimated

approach:

1. estimate $P(Y,\boldsymbol{X})$ → generative model (can be used to generate new samples)
2. calculate $P(Y|\boldsymbol{X})$ from $P(Y,\boldsymbol{X})$ → used for discriminative task (classification)

Independence Assumption

(naïve) assumption: conditional independence of features given target

$$P\left(X_j | Y, X_1, \cdots, X_{j-1}, X_{j+1}, \cdots, X_p\right) = P\left(X_j | Y\right)$$

$$\Longrightarrow P\left(Y | X_1, \cdots, X_p\right) = \frac{P(Y) \prod_{j=1}^{p} P\left(X_j | Y\right)}{P\left(X_1, \cdots, X_p\right)}$$

→ independent feature contributions (ignoring feature correlations)

→ robust against curse of dimensionality

Estimation of Feature Contributions

separate estimations of $P(X_j|Y)$ for each feature

requires assumption of distributions (e.g., Gaussian naïve Bayes) or non-parametric methods (kernel density estimation)

Gaussian feature likelihoods: $$P(x_{ij}|y) = \frac{1}{\sqrt{2\pi\sigma_{y,j}^2}} \exp\left(-\frac{(x_{ij}-\mu_{y,j})^2}{2\sigma_{y,j}^2}\right)$$

parameter estimation (e.g., mean and variance of Gaussians) can be done with maximum likelihood method (y known in training)

→ no Bayesian methods needed

Maximum a Posteriori Classification

$$\hat{y}_i = \underset{y}{\operatorname{argmax}} P(y) \prod_{j=1}^{p} P\left(x_{ij} | y\right)$$

despite potentially inaccurate probability estimates (due to naïve independence assumption), good identification of correct class via maximum probability

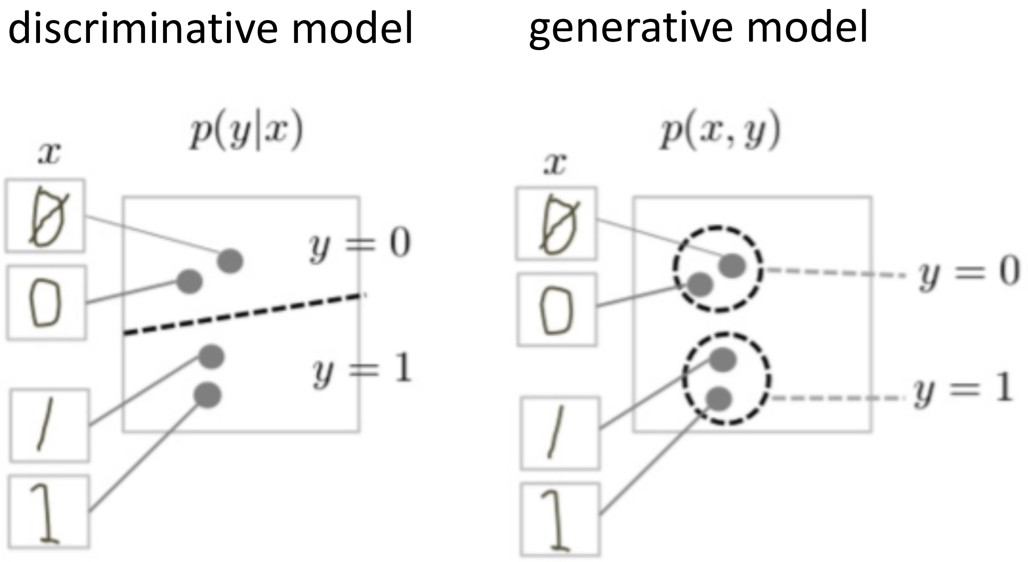
→ bad for regression tasks (if independence assumption is too naïve, i.e., features are correlated)

Generative vs Discriminative Models

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

discriminative models: predict conditional probability $P(Y|\boldsymbol{X})$ 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



source

Naïve Bayes and Logistic Regression

generative-discriminative pair of classification algorithms

- binary case: logit of naïve Bayes' outputs, $\log\left(\frac{P(y_i=1|\boldsymbol{x}_i)}{P(y_i=0|\boldsymbol{x}_i)}\right)$, corresponds to output of logistic regression's linear predictor
- for discrete inputs or Gaussian naïve Bayes: naïve Bayes can be reparametrized as linear classifier

for discriminative task: identical in asymptotic limit (infinite training samples) if independence assumption holds (otherwise naïve Bayes less accurate)

naïve Bayes has greater bias but lower variance than logistic regression → to be preferred for scarce training data (if bias, i.e., independence assumption, correct)

Data Generation

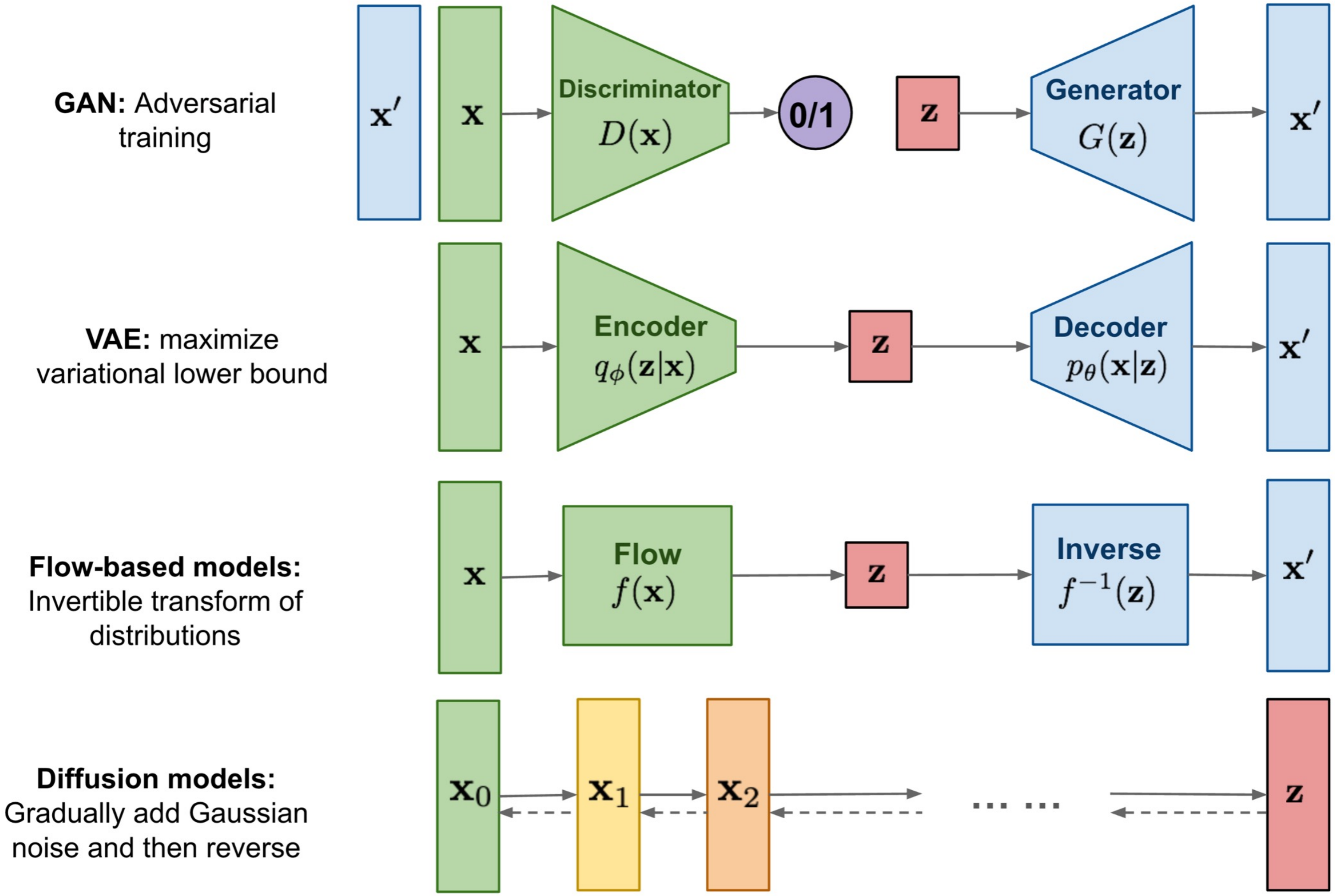
generative models can be used for discriminative tasks (although potentially inferior to direct discriminative methods)

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

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

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

Different Types of Generative Models



source

Variational Autoencoders (VAE)

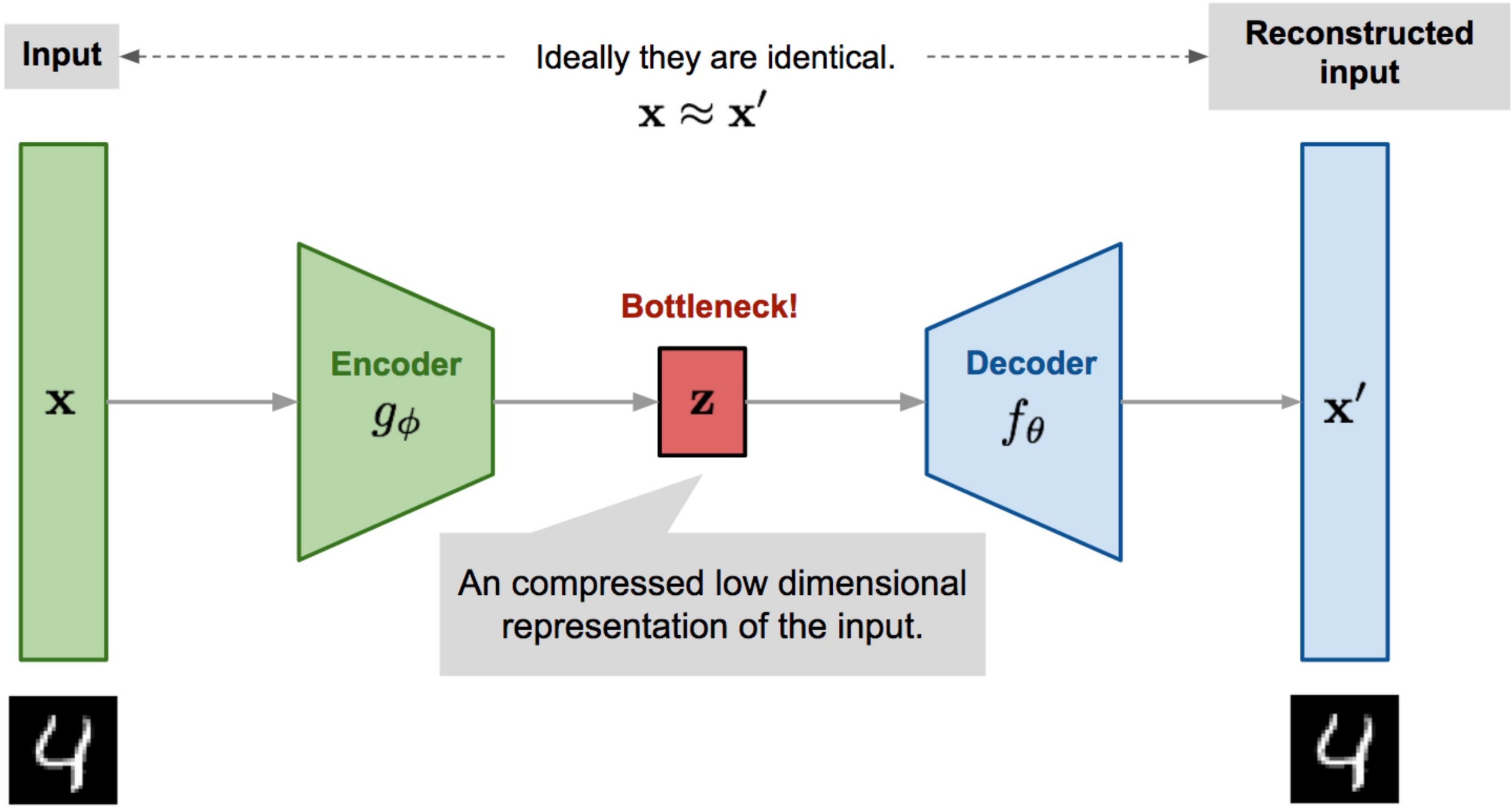
Recap: Autoencoder

(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

Autoencoder Architecture for Generative Tasks

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

→ learn variational distribution instead of identity function

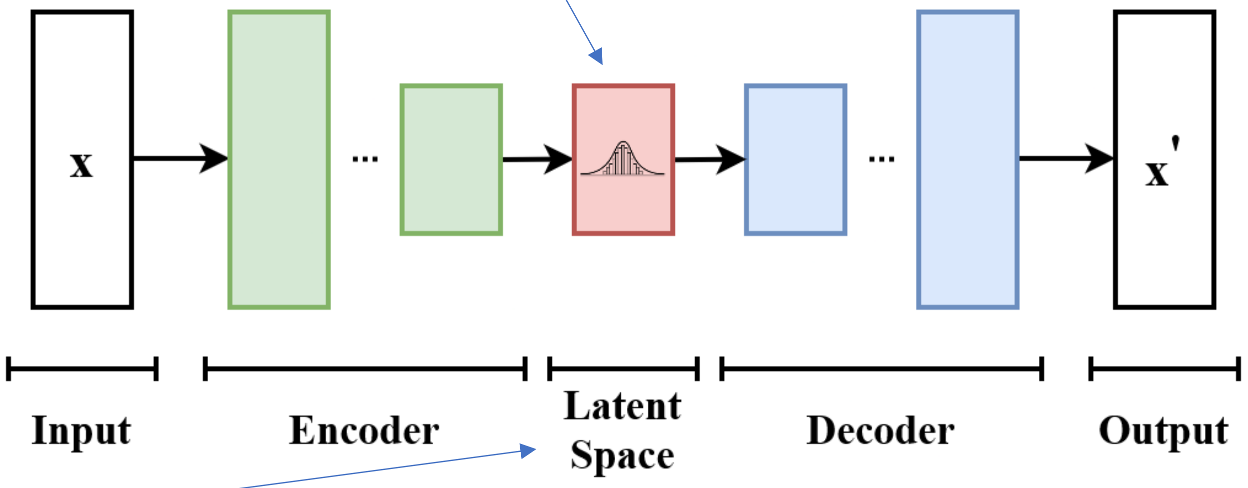
to be precise: parametrized variational distribution of latent encoding variables $\boldsymbol{z}$

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

Encoder and Decoder Networks

encoder: find posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$

unfortunately, generally intractable (integral over $\boldsymbol{z}$ expensive)

→ approximate by $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$

VAE: $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ expressed by neural network with weights $\boldsymbol{\phi}$

→ amortized inference:
$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ learned in training, $\boldsymbol{z}$ inferred from $\boldsymbol{x}$ in prediction (sharing variational parameters across all data points)

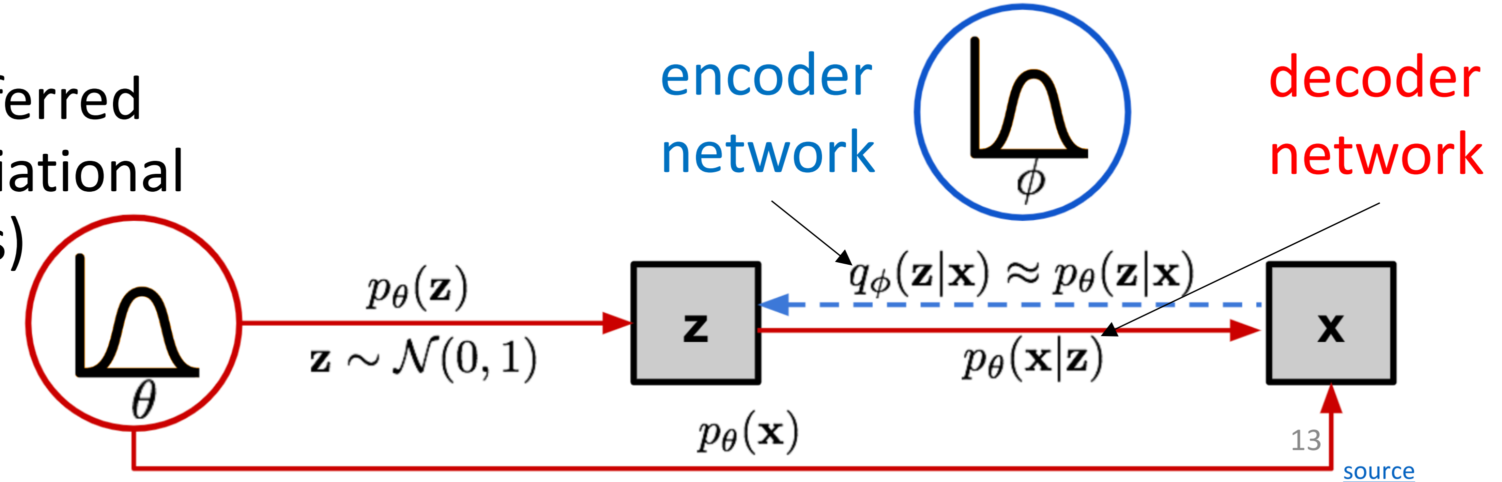
in VAE: network weights $\boldsymbol{\theta}$

decoder: generate new sample $\boldsymbol{x}_i$

1. sample $\boldsymbol{z}_i$ (from Gaussian)
2. generate $\boldsymbol{x}_i$ (similar to real data)

→ maximize: $p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z})\, p_{\boldsymbol{\theta}}(\boldsymbol{z}) dz$

(expensive → use only likely codes $\boldsymbol{z}$ given input $\boldsymbol{x}$: need for encoder)



source

VAE Loss: ELBO

VAE loss function to be minimized according to network weights:

$$L(\boldsymbol{x}_i; \boldsymbol{\theta}, \boldsymbol{\phi}) = -\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + D_{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}_i)\right)$$

maximize likelihood of observed data (minimize reconstruction error)

and

minimize difference of approximation $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)$ to exact posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}_i)$

can be interpreted as regularizer

corresponds to maximizing evidence lower bound (ELBO), i.e., maximizing lower bound of probability to generate real data sample:

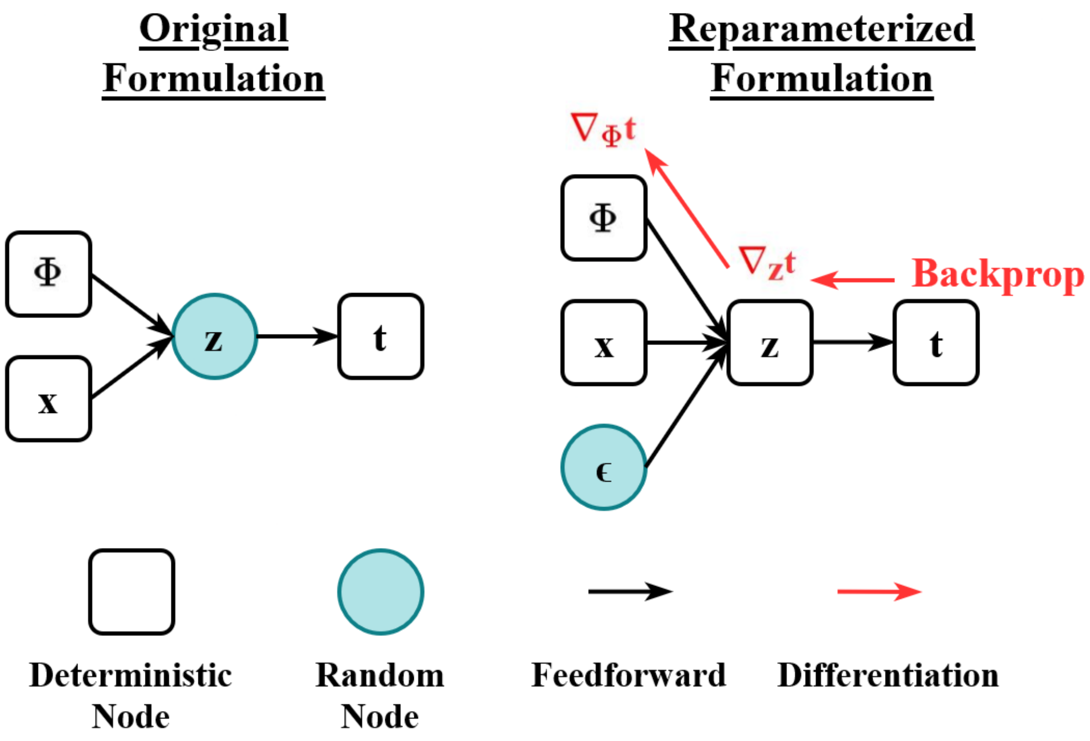
$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \underbrace{D_{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}_i)\right)}_{\text{non-negative}} = E_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)}\left[\ln \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)}\right]$$

Reparameterization Trick

→ gradient descent according to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$

issue: not readily possible for $\boldsymbol{\phi}$ (expecatation over $\boldsymbol{z}$, which is sampled from $q_{\boldsymbol{\phi}}$)

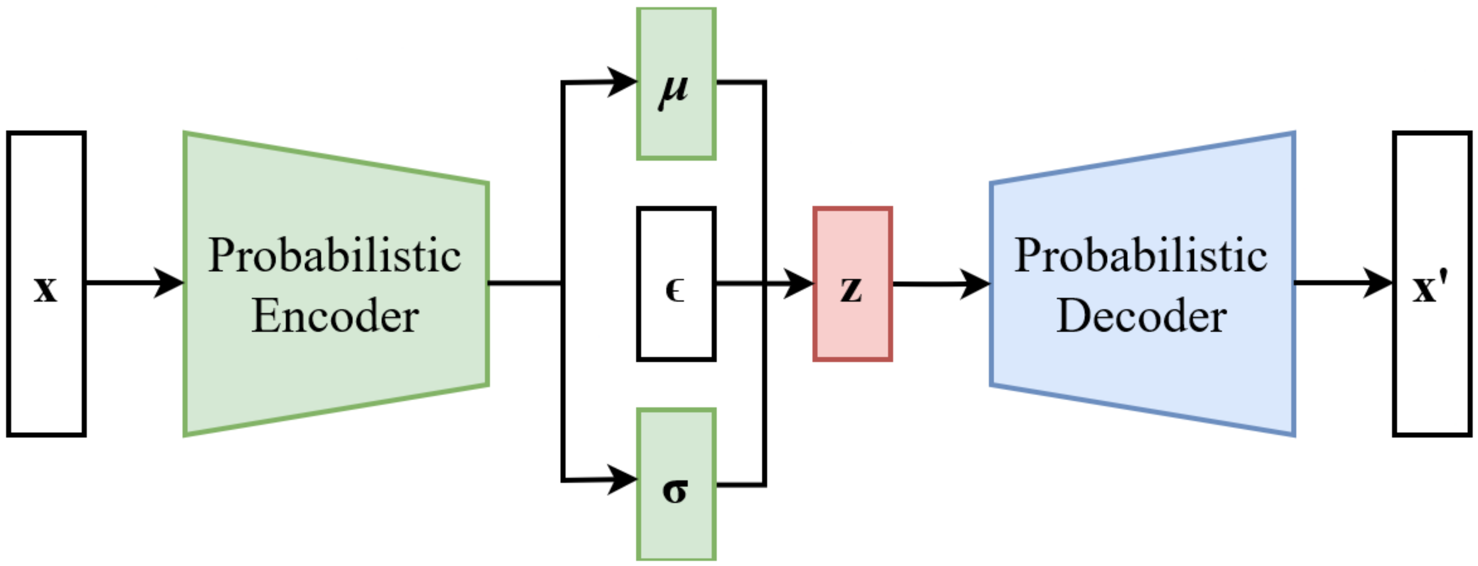
→ reparametrization to the rescue: express randomness in $\boldsymbol{z}$ by independent auxiliary variable $\boldsymbol{\varepsilon}$



from wikipedia

e.g., q_ϕ as multivariate Gaussian with diagonal covariance structure

→ learn mean and variance



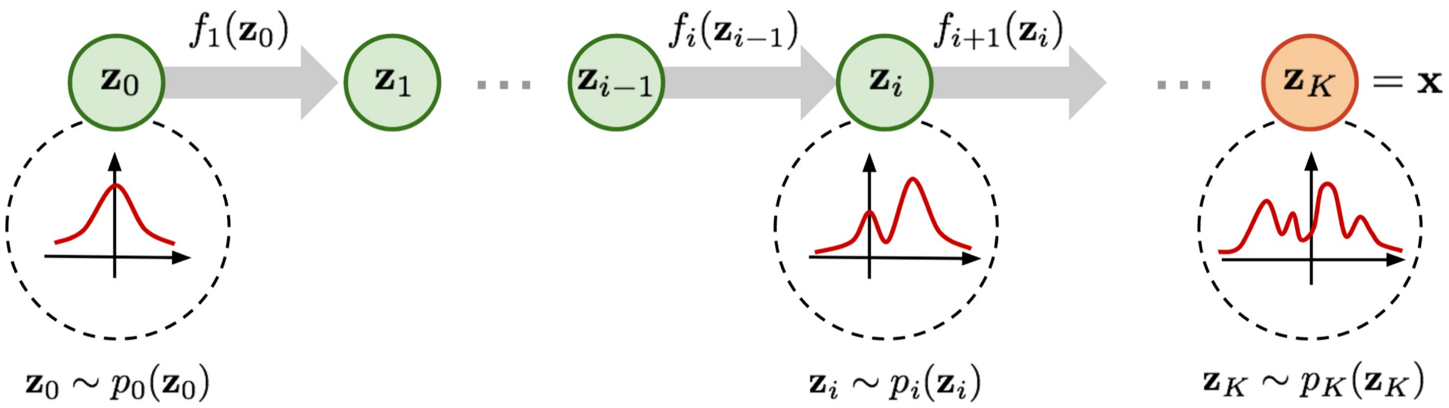
from wikipedia

Flow-Based Methods

Normalizing Flows

… beyond Gaussian …

…



source

- … specialized architectures to construct reversible transform

Generative Adversarial Networks (GAN)

Indirect Training via Discriminator

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

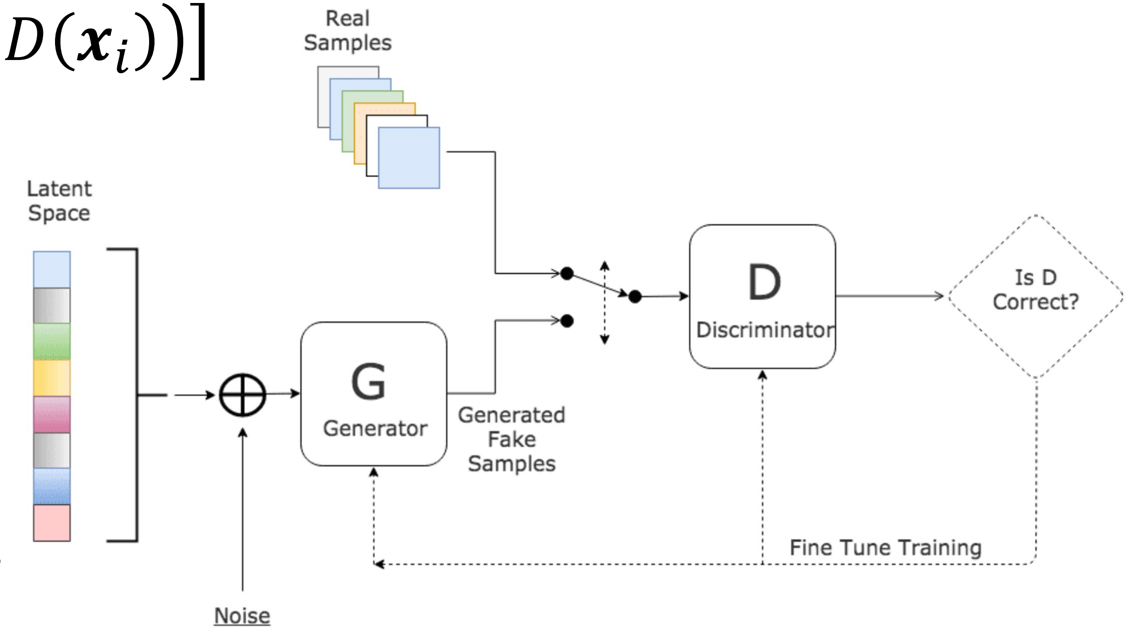
idea: G not trained directly to minimize reconstruction error of real samples, but to fool D → self-supervised approach

Formulation

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



decomposition into latent space (parameters of generator network) and noise (sampled from, e.g., Gaussian distribution): reparametrization trick

source

Properties

implicit generative model: do not estimate likelihood function

for optimal D, GAN loss quantifies similarity between generative data distribution p_g and real data distribution p_r by Jensen-Shannon divergence

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p||\frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q||\frac{p+q}{2}\right)$$

for optimal values of both G and D: $p_g = p_r$ and $D = 0.5$

issue: potentially unstable training

Diffusion Models

- inspired by non-equilibrium thermodynamics
- Markov chain of diffusion steps to slowly add random noise to data
- …chain of denoising autoencoders…

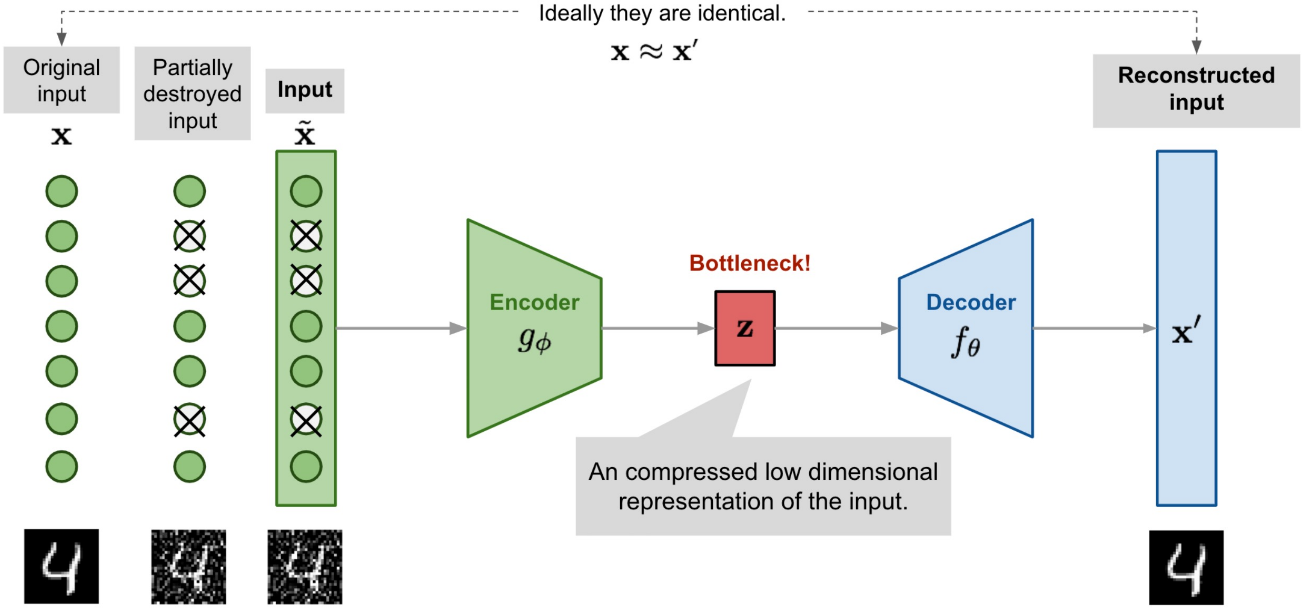
Denoising Autoencoder

goal: avoid overfitting and improve robustness of plain autoencoder

...

...

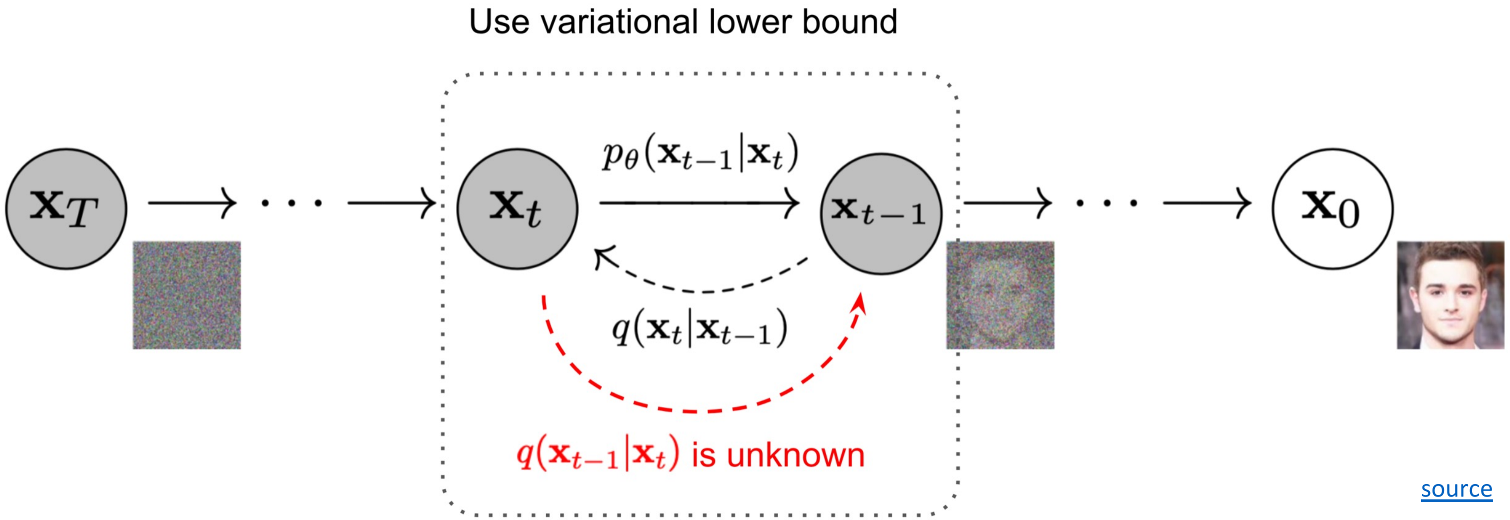
similar to dropout



source

- then learn to reverse the diffusion process to construct desired data samples from the noise
- Unlike VAE or flow models, diffusion models are learned with a fixed procedure and the latent variable has high dimensionality (same as the original data).

- ...

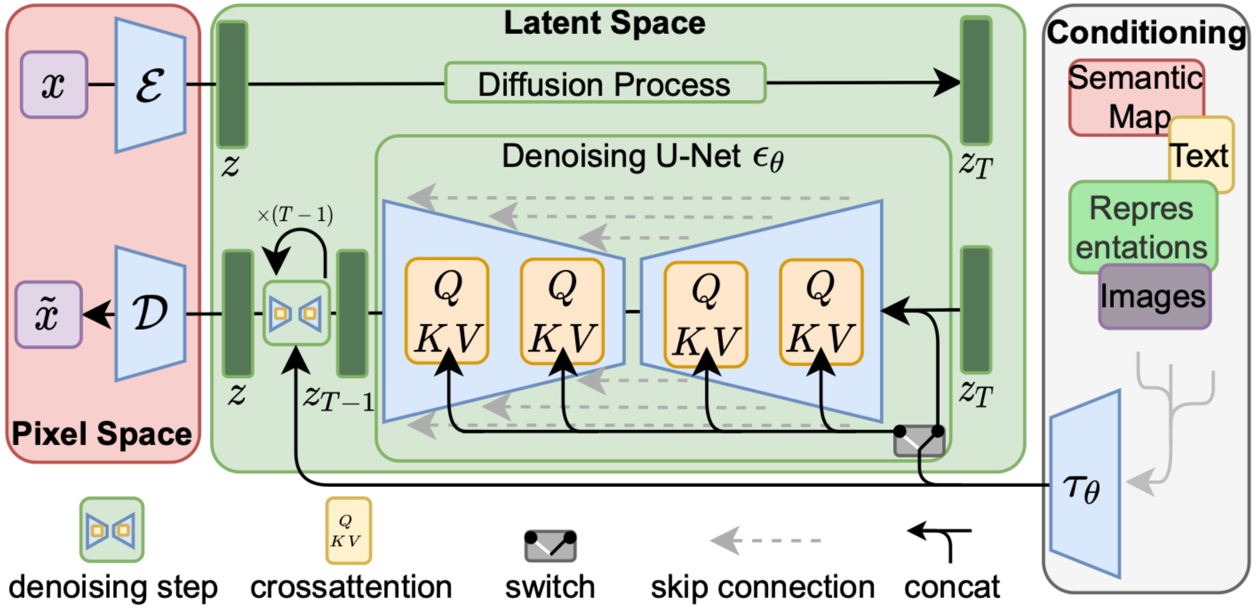

Use variational lower bound
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown
$\mathbf{x}_0$

source

Latent Diffusion Model

… idea: add noise to latent representation rather than raw data

… speedup



source

Conditioned Generation

as discussed so far, generative methods give no control over what kind of data is generated (limited usability)

→ need for conditional approach (e.g., conditioning on describing text)

…

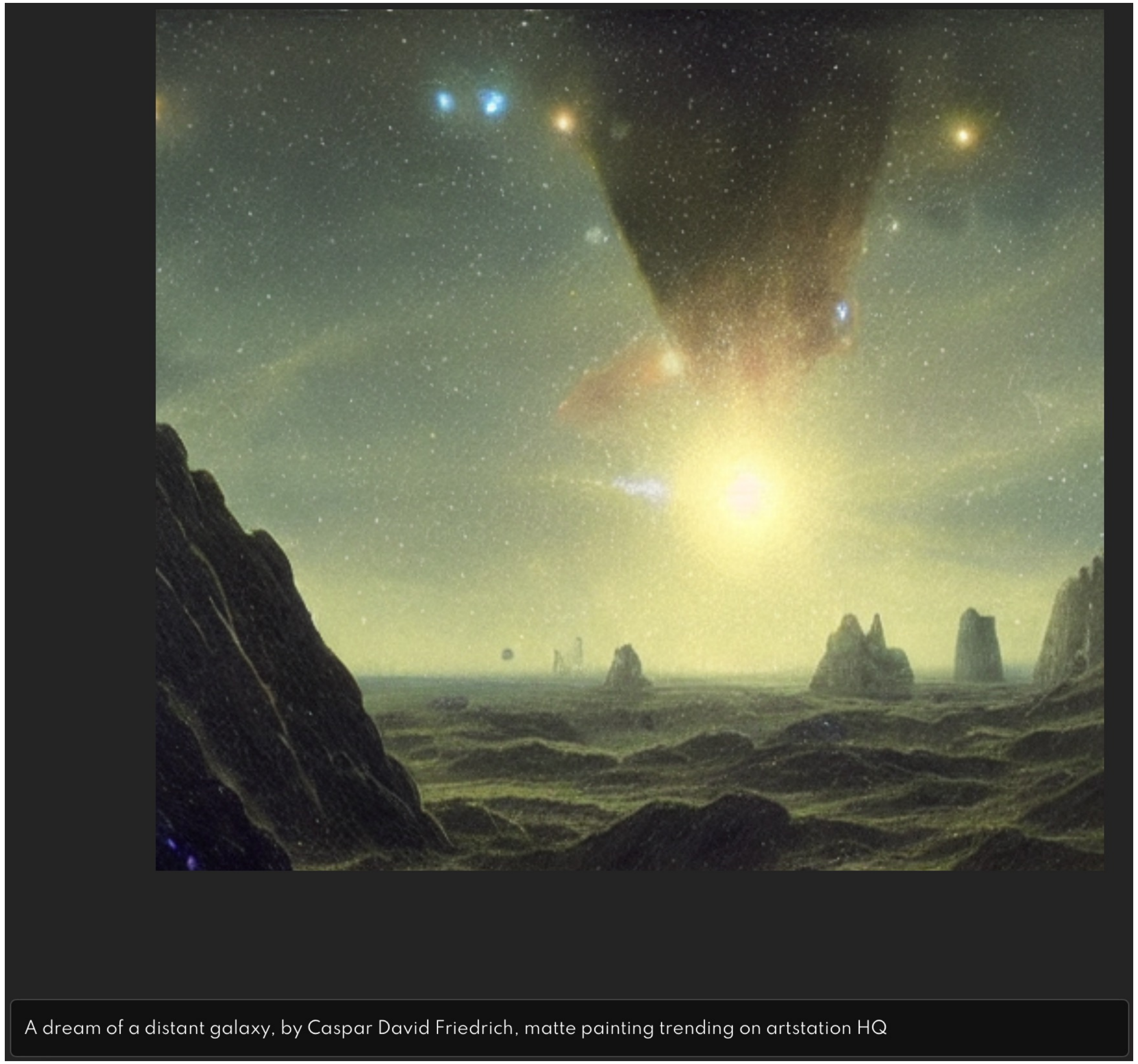
Applications

…

text-to-image:
DALL-E 2, Stable Diffusion

web app for Stable Diffusion:

DreamStudio



Literature

papers:

- variational autoencoder
- normalizing flows
- GAN
- latent diffusion

Movie-like Intelligence

emergent capabilities of complex systems almost impossible to foresee

mini examples in contemporary ML:

- large language models
- multi-agent reinforcement learning

one idea: reward is enough

philosophical: emotions or consciousness might also occur as emergent capabilities