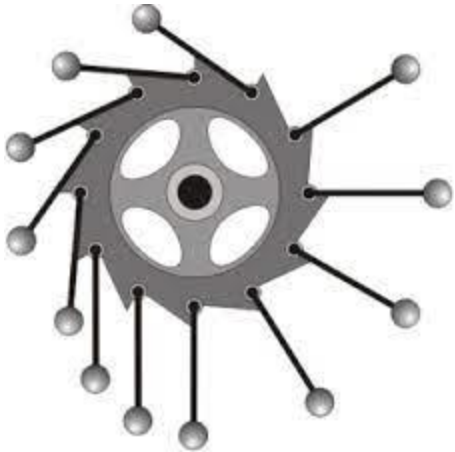
Deep Generative Models

Yadong Mu
Wangxuan Institute of Computer Technology
Peking University



Most slides are adapted from related courses or tutorials.
Internal use only. Please do not distribute the slides!

Deep Generative Models

- ability to produce highly realistic pieces of content of various kind, such as images, texts and sounds
- Manipulation the latent code
- Deep generative models vs. prob. graphical models

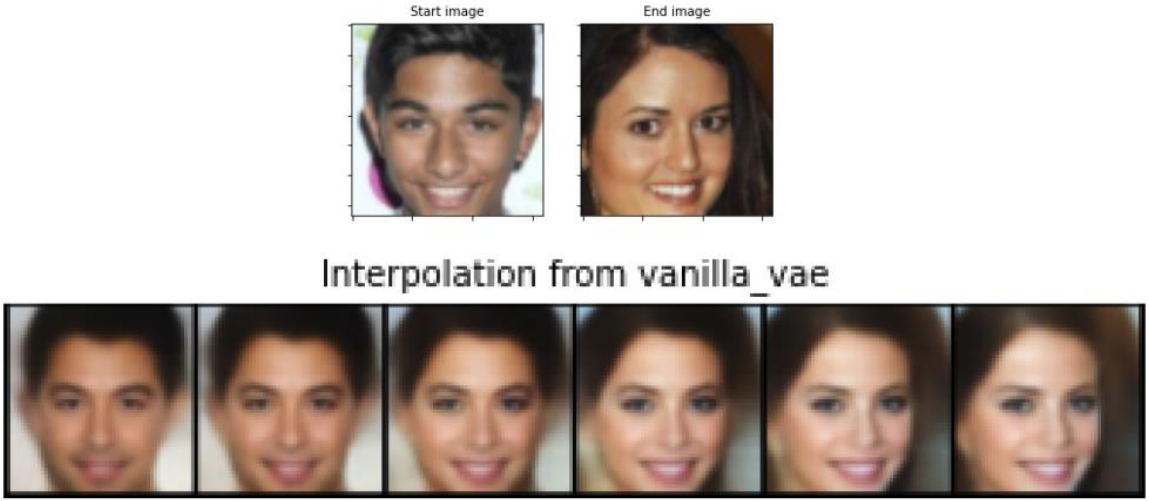


Figure 19.6: Interpolation between two images (top row) in the latent space of a VAE. Generated by vae_compare_results.ipynb.

Deep Generative Models

- ability to produce highly realistic pieces of content of various kind, such as images, texts and sounds
- Manipulation the latent code
- Deep generative models vs. prob. graphical models

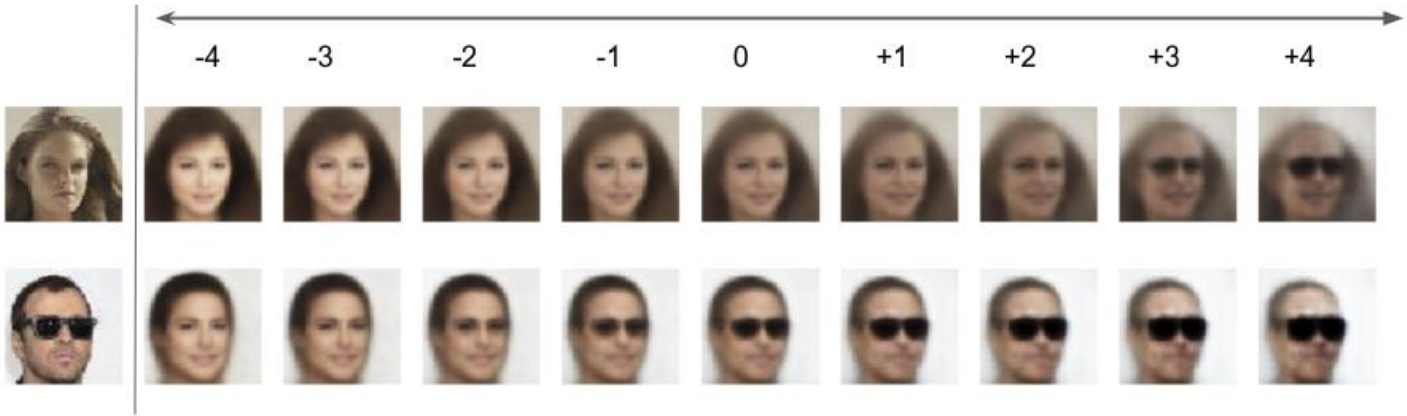
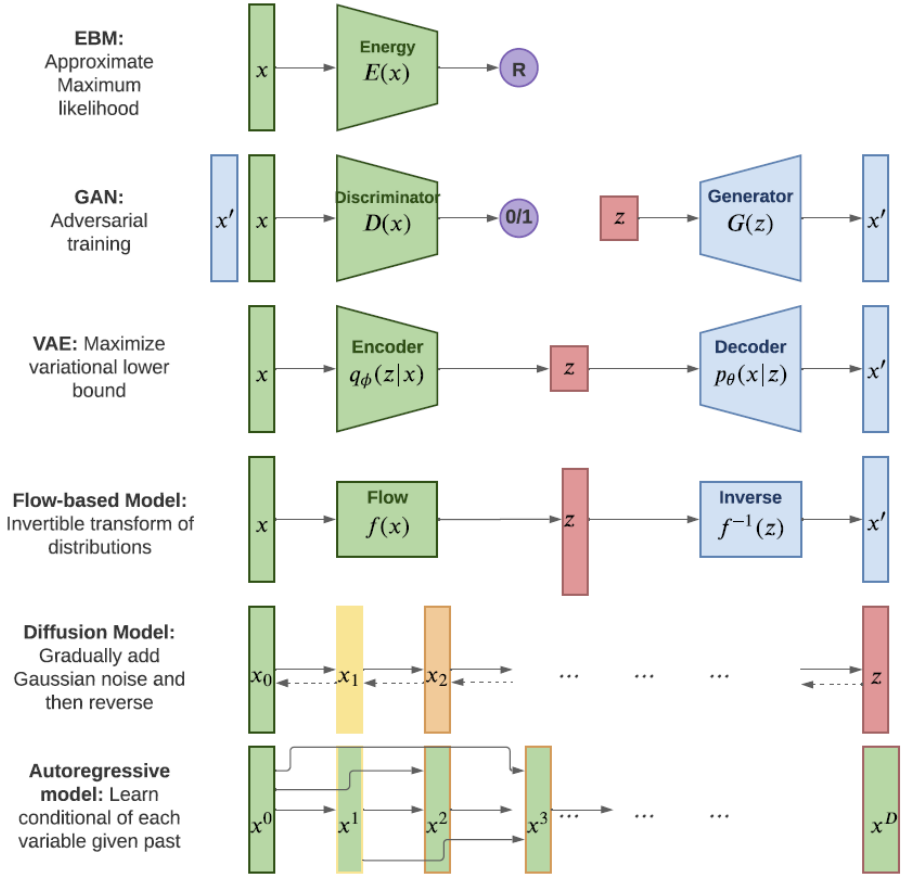


Figure 19.7: Arithmetic in the latent space of a VAE . The first column is an input image, with embedding z. Subsequent columns show the decoding of $z + s\Delta$, where $s \in \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ and $\Delta = \overline{z}^{+} - \overline{z}^{-}$ is the difference in the average embeddings of images with or without a certain attribute (here, wearing sunglasses). Generated by vae_ celeba_ lightning.ipynb.

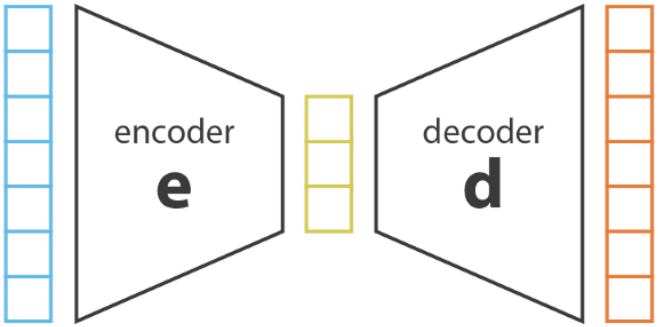
Overview of Generative Models



Credit: Prof. Kevin Murphy and https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Encoder-Decoder View of Dimension Reduction

- Encoder-decoder view of dimension reduction



x

initial data
in space R^n

e(x)

encoded data
in latent space R^m (with m<n)

d(e(x))

encoded-decoded data
back in the initial space R^n

$x = d(e(x))$ ➡ **lossless encoding**
no information is lost when reducing the number of dimensions

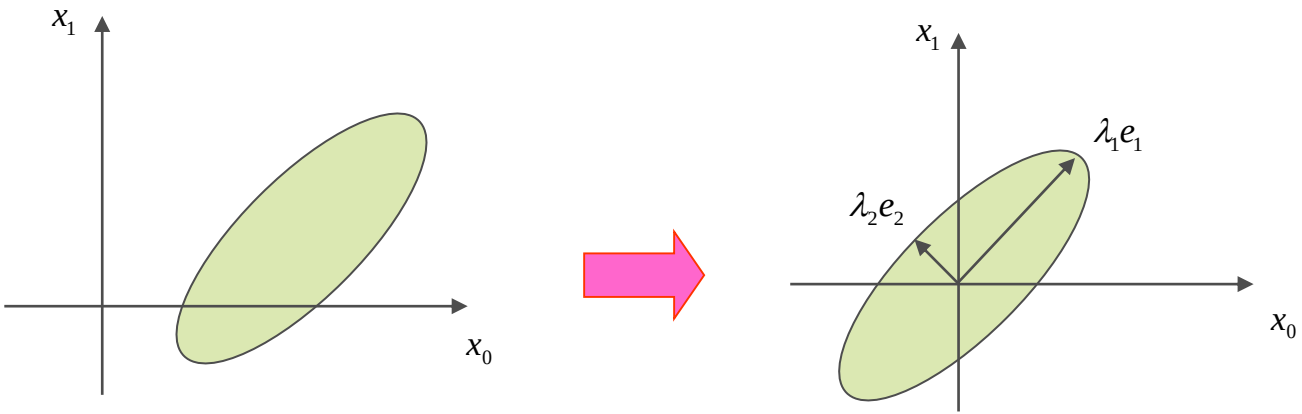
$x \neq d(e(x))$ ➡ **lossy encoding**
some information is lost when reducing the number of dimensions and can't be recovered later

Illustration of the dimensionality reduction principle with encoder and decoder.

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Principal Component Analysis (PCA)

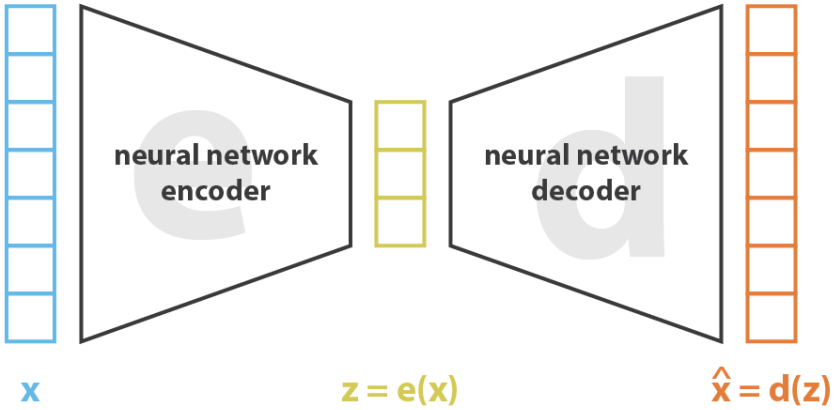
- Find a low-dimensional embedding of the data points that best preserves their variance as measured in the high-dimensional input space.



λ_k is the marginal variance along the principle direction e_k

AutoEncoder

- Replace orthonormal vectors by non-linear neural networks

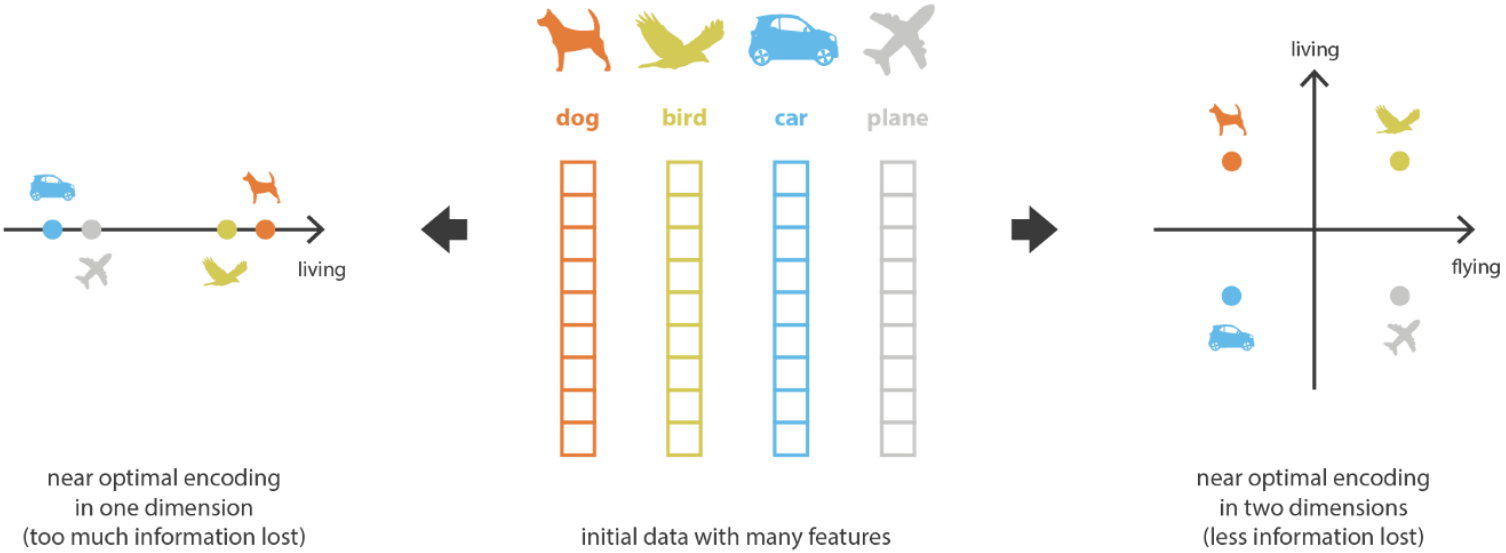


$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Illustration of an autoencoder with its loss function.

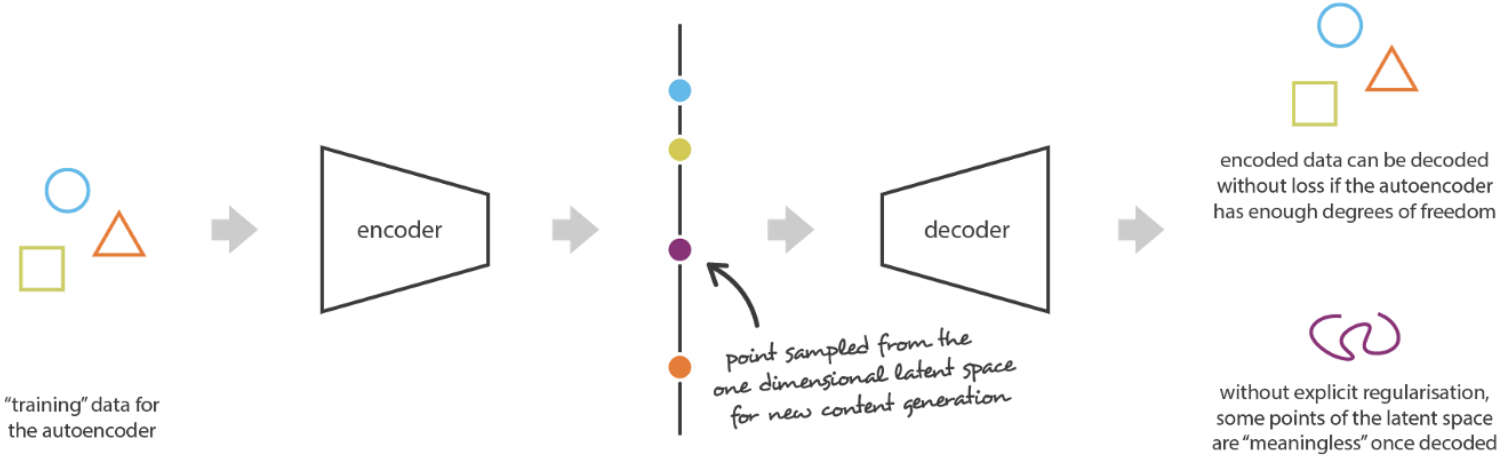
Intuition about Regular Latent Space

- When reducing dimensionality, we want to keep the main structure there exists among the data.
- classic autoencoders tend to over-fit



Intuition about Regular Latent Space

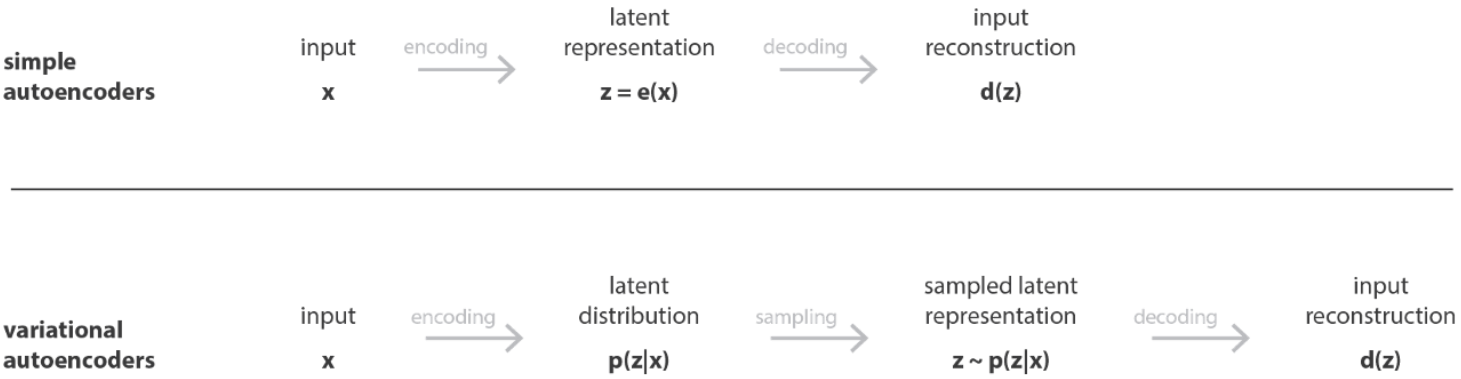
- Latent space shall support sampling from the whole space (extrapolation)



Irregular latent space prevent us from using autoencoder for new content generation.

Probabilistic Encoding in VAE

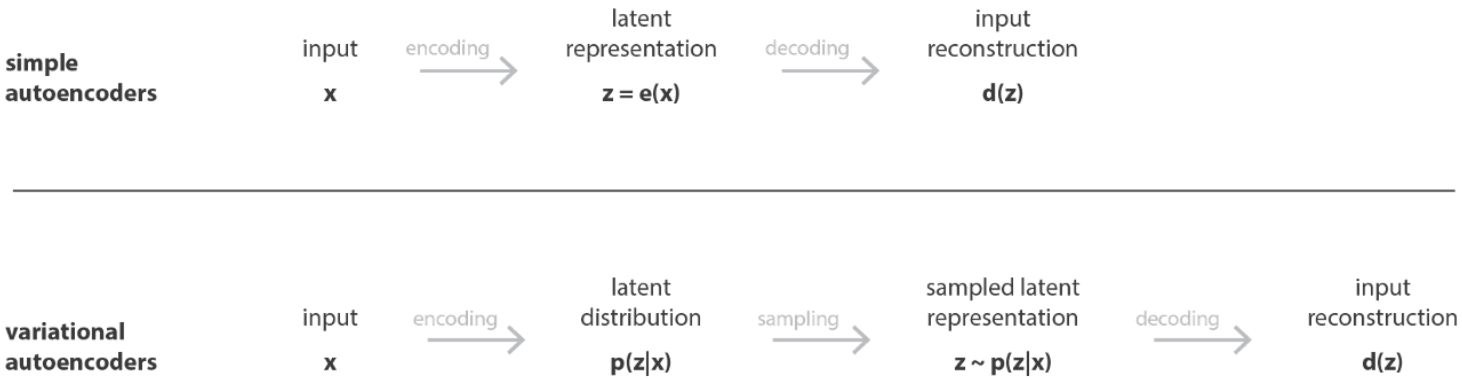
- Encoding as a point (as in naïve AE) and distribution (as in VAE)



Difference between autoencoder (deterministic) and variational autoencoder (probabilistic).

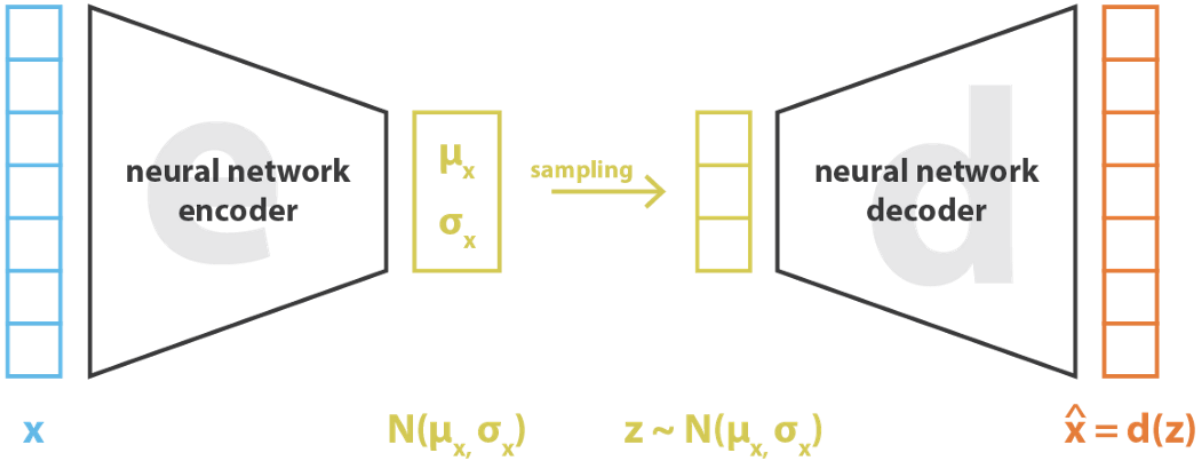
Probabilistic Encoding in VAE

- first, the input is encoded as distribution over the latent space
- second, a point from the latent space is sampled from that distribution
- third, the sampled point is decoded and the reconstruction error can be computed
- finally, the reconstruction error is backpropagated through the network



Loss of VAE

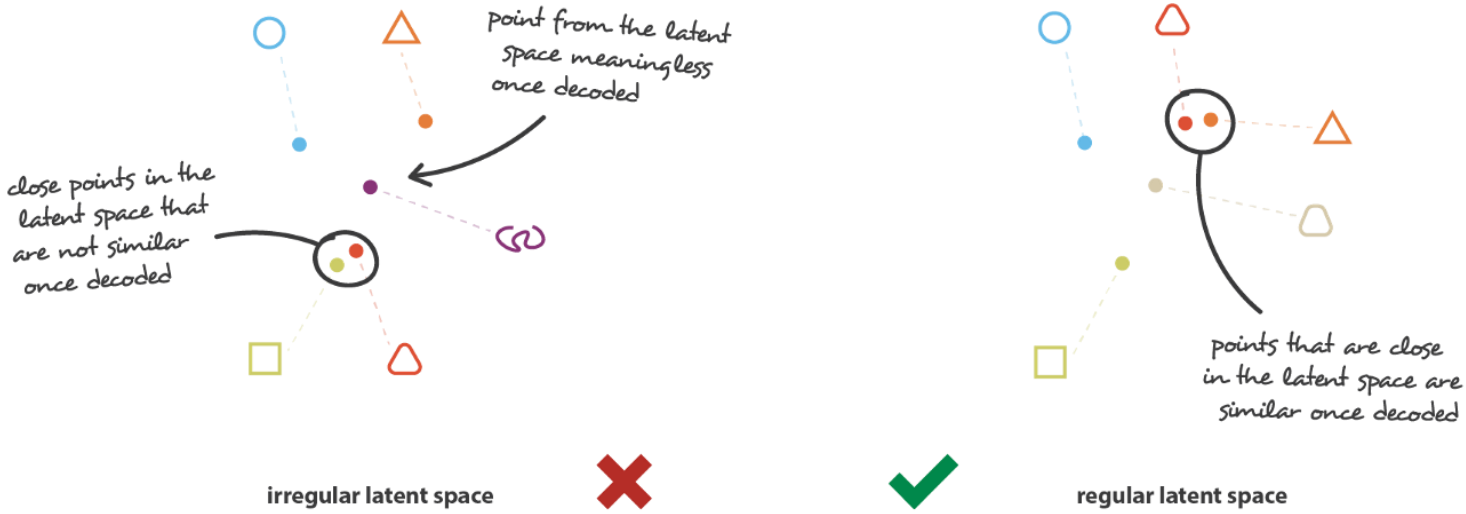
- the loss function is composed of a reconstruction term (that makes the encoding-decoding scheme efficient) and a regularization term (that makes the latent space regular).



$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Regularization of Latent Code

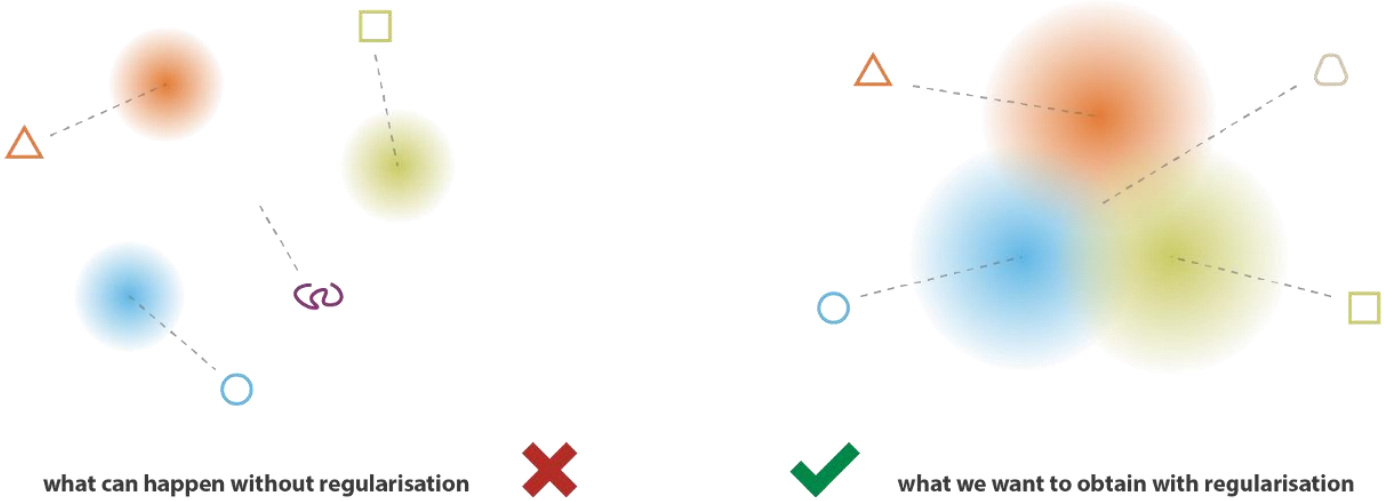
- **continuity** (two close points in the latent space should not give two completely different contents once decoded)
- **completeness** (for a chosen distribution, a point sampled from the latent space should give “meaningful” content once decoded)



Difference between a “regular” and an “irregular” latent space.

Regularization of Latent Code

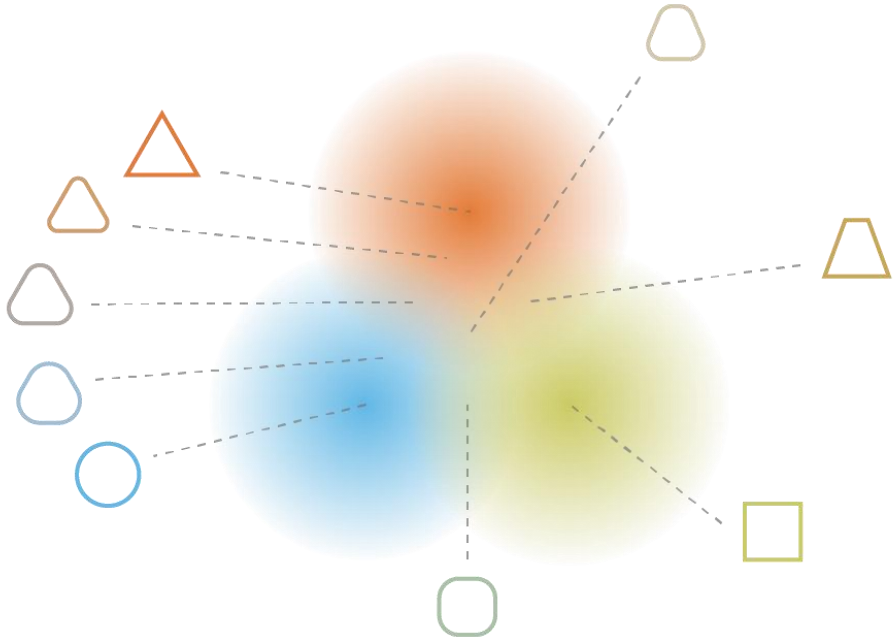
- prevent the model to encode data far apart in the latent space and encourage as much as possible returned distributions to “overlap”
- Tradeoff between reconstruction error / regularization



The returned distributions of VAEs have to be regularised to obtain a latent space with good properties.

Regularization of Latent Code

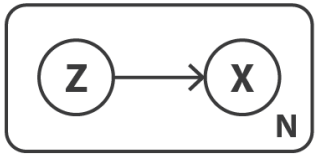
- Regularize both the mean and variance (covariance)
- Set an ideal distribution and calculate the KL-divergence with it



Regularisation tends to create a “gradient” over the information encoded in the latent space.

Formal Derivation of VAE

- denote by x the variable that represents our data and assume that x is generated from a latent variable z (the encoded representation)
- The “probabilistic decoder” is naturally defined by p(x|z), that describes the distribution of the decoded variable given the encoded one
- the “probabilistic encoder” is defined by p(z|x), that describes the distribution of the encoded variable given the decoded one.



Graphical model of the data generation process.

Formal Derivation of VAE

- Bayes theorem that makes the link between the prior p(z), the likelihood p(x|z), and the posterior p(z|x)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du}$$

- Assume p(z) is a standard Gaussian distribution and that p(x|z) is a Gaussian distribution whose mean is defined by a deterministic function f of the variable of z and whose covariance matrix has the form of a positive constant c that multiplies the identity matrix I.

$$p(z) \equiv \mathcal{N}(0, I)$$
$$p(x|z) \equiv \mathcal{N}(f(z), cI) \qquad f \in F \qquad c > 0$$

Variational Inference

- The idea of VI: set a parametrised family of distribution and to look for the best approximation of our target distribution among this family
- approximate the (otherwise intractable) p(z|x) by a Gaussian distribution q_x(z)

$$q_x(z) \equiv \mathcal{N}(g(x), h(x)) \qquad g \in G \qquad h \in H$$

Approximating the posterior

- Derivation using KL-divergence

$$
\begin{aligned}
(g^*, h^*) &= \underset{(g,h)\in G\times H}{\arg\min}\ KL(q_x(z), p(z|x)) \\
&= \underset{(g,h)\in G\times H}{\arg\min}\left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)}\right)\right) \\
&= \underset{(g,h)\in G\times H}{\arg\min}\left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x|z)) + \mathbb{E}_{z\sim q_x}(\log p(x))\right) \\
&= \underset{(g,h)\in G\times H}{\arg\max}\left(\mathbb{E}_{z\sim q_x}(\log p(x|z)) - KL(q_x(z), p(z))\right) \\
&= \underset{(g,h)\in G\times H}{\arg\max}\left(\mathbb{E}_{z\sim q_x}\left(-\frac{||x - f(z)||^2}{2c}\right) - KL(q_x(z), p(z))\right)
\end{aligned}
$$

staying close to the prior distribution

maximizing the likelihood of the “observations”

Approximating the posterior

- For any function f, let the best approximation of p(z|x) be q*_x(z)
- we want to choose the function f that maximizes the expected log-likelihood of x given z when z is sampled from q*_x(z)

$$f^* = \arg\max_{f \in F} \mathbb{E}_{z \sim q_x^*}(\log p(x|z))$$
$$= \arg\max_{f \in F} \mathbb{E}_{z \sim q_x^*}\left(-\frac{||x - f(z)||^2}{2c}\right)$$

- Gathering all the pieces together, we are looking for optimal f*, g* and h* such that

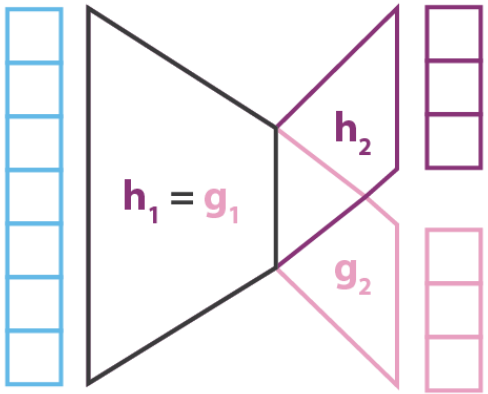
regularization term given by the KL divergence

$$(f^*, g^*, h^*) = \arg\max_{(f,g,h) \in F \times G \times H}\left(\mathbb{E}_{z \sim q_x}\left(-\frac{||x - f(z)||^2}{2c}\right) - KL(q_x(z), p(z))\right)$$

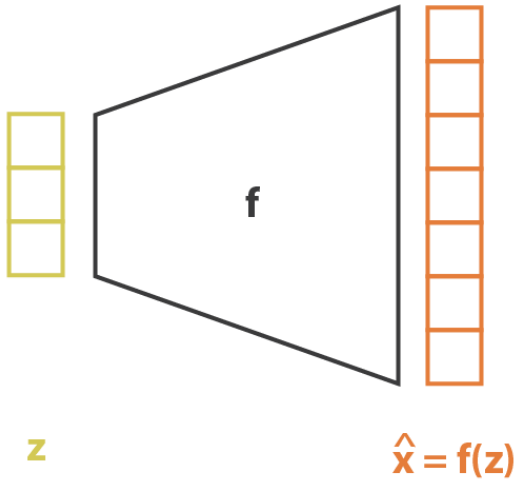
the reconstruction error between x and f(z)

Bringing neural networks into the model

- F, G and H correspond respectively to the families of functions defined by the networks architectures and the optimization is done over the parameters of these networks.



Encoder part of the VAE.



Decoder part of the VAE.

The Reparametrization Trick

- using the fact that if z is a random variable following a Gaussian distribution with mean g(x) and with covariance H(x)=h(x)

$$z = h(x)\zeta + g(x) \qquad \zeta \sim \mathcal{N}(0, I)$$

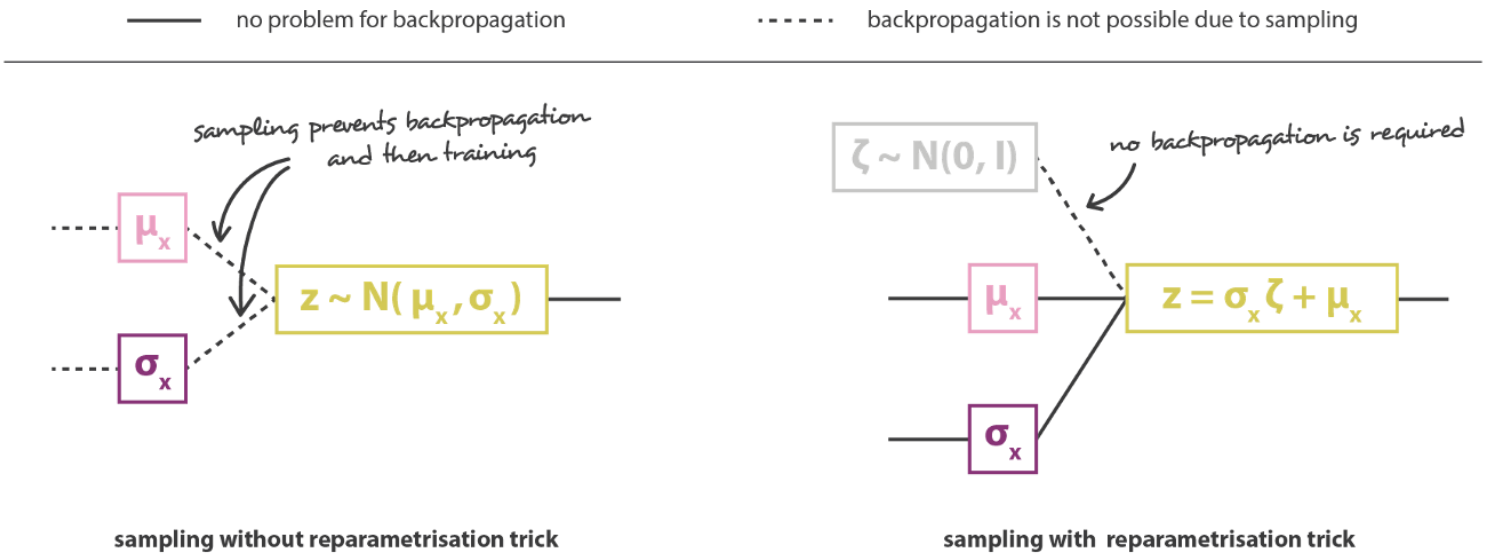
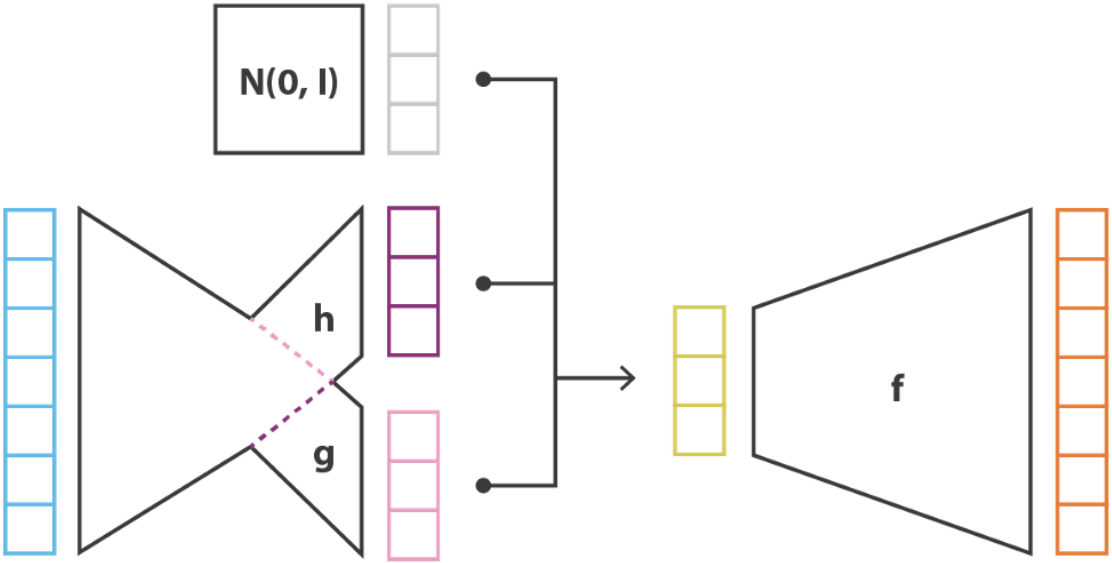


Illustration of the reparametrizsation trick.

The Final Objective



$$\text{loss} = C \,||\, x - \hat{x} \,||^2 + KL[\, N(\mu_x, \sigma_x), N(0, I) \,] = C \,||\, x - f(z) \,||^2 + KL[\, N(g(x), h(x)), N(0, I) \,]$$

Rethinking the KL-Divergence

The log likelihood for a point x is:

$$\log p\,(x) = \log E_{z\sim q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

Jensen's inequality (log is a concave function):

$$\log E\,(\cdots) \geq E \log(\cdots)$$

$$\geq E_{z\sim q_\phi(z|x)}\left[\log\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

Instead of maximizing $\log p\,(x)$, we maximize this *variational lower bound*. It has the form:

$$\mathcal{L}(\theta,\phi;x) = E_{z\sim q_\phi(z|x)}[\mathcal{L}(\theta,\phi;x,\mathrm{z})]$$

where

$$\mathcal{L}(\theta,\phi;x,\mathrm{z}) = \log\frac{p_\theta(x,z)}{q_\phi(z|x)} = \log p_\theta(x,z) - \log q_\phi(z|x)$$

Aside: Jensen's Inequality

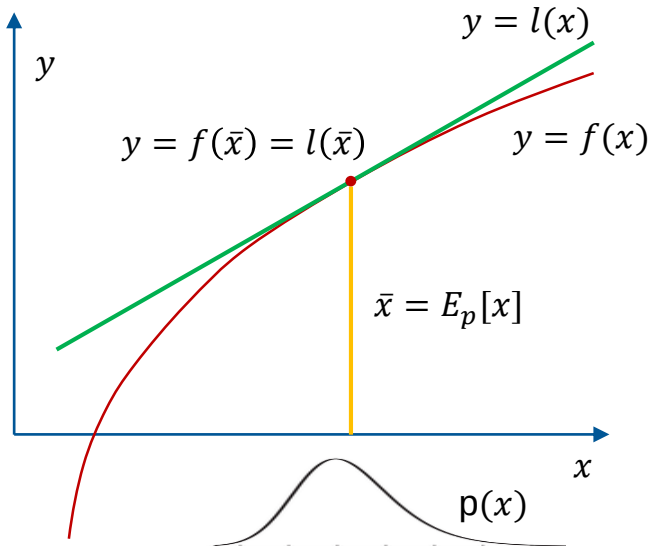
For concave $f(x)$ i.e. $f'' < 0$, (here its $\log x$) construct a linear function $l(x)$ tangent to $f(x)$ at $\bar{x} = E_p[x]$. By construction of the tangent line: $f(\bar{x}) = l(\bar{x})$ and $f(x) \leq l(x)$.

For a linear function $l\big(E_p[x]\big) = E_p[l(x)]$,
so $f\big(E_p[x]\big) = l\big(E_p[x]\big) = E_p[l(x)]$

Since $f(x) \leq l(x)$ it follows
$f\big(E_p[x]\big) = E_p[l(x)] \geq E_p[f(x)]$

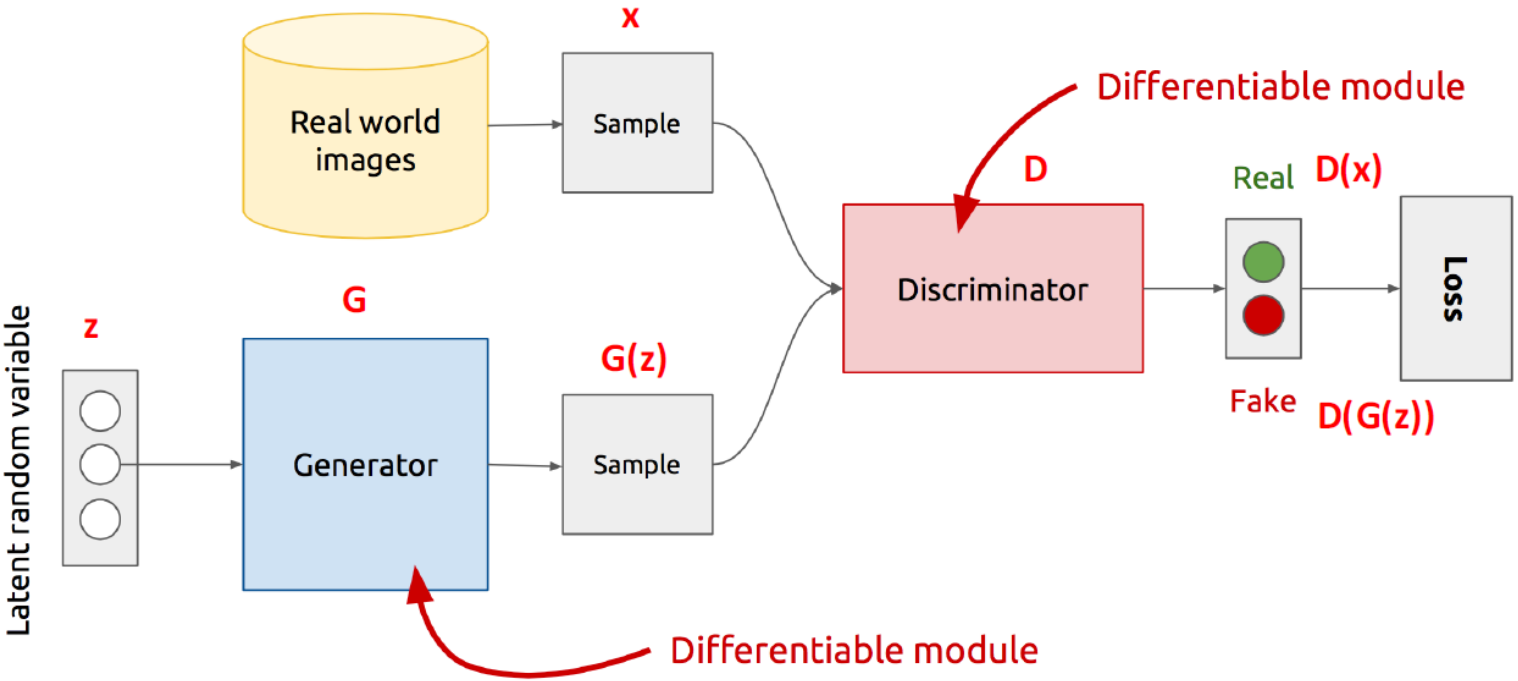
Similarly if $f(x)$ is convex, $f\big(E_p[x]\big) \leq E_p[f(x)]$

Finally, for the multivariate case $f(E_{z\sim q(z)}[h(z)])$ simply take $x = h(z)$ and for $p(x)$ the distribution on x induced by sampling $z \sim q(z)$.

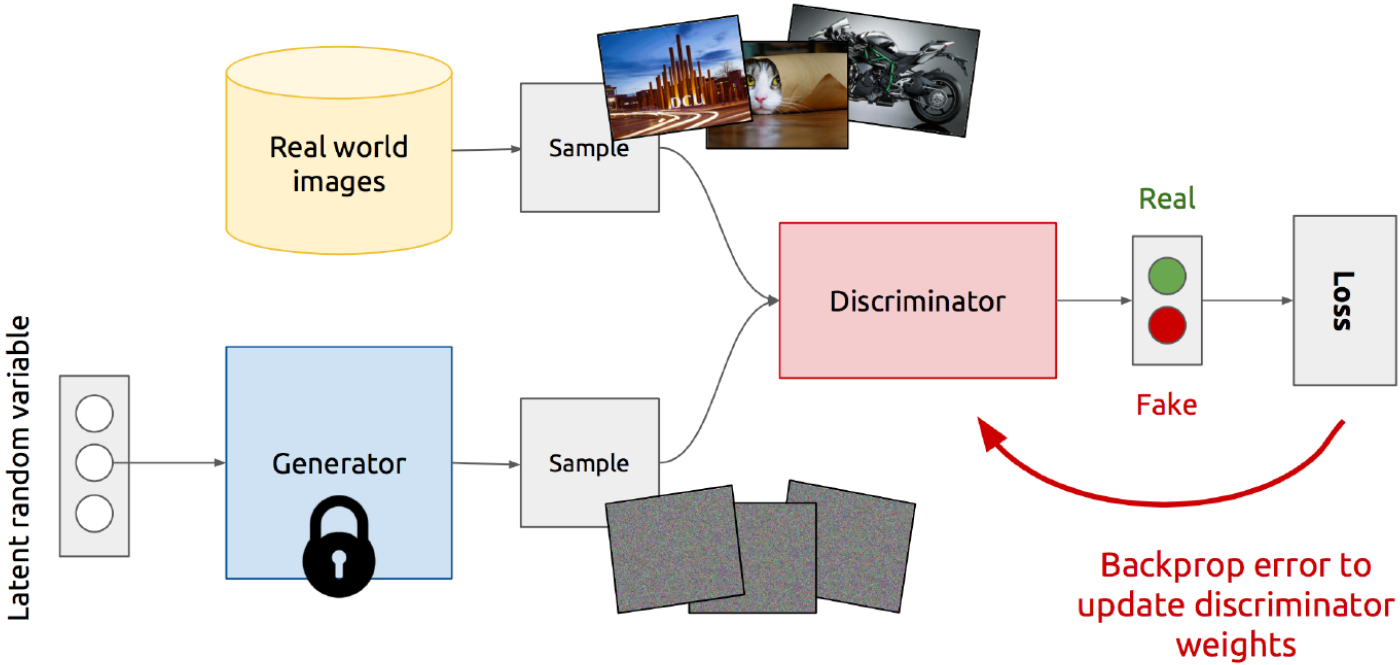


Introduction to Generative Adversarial Network

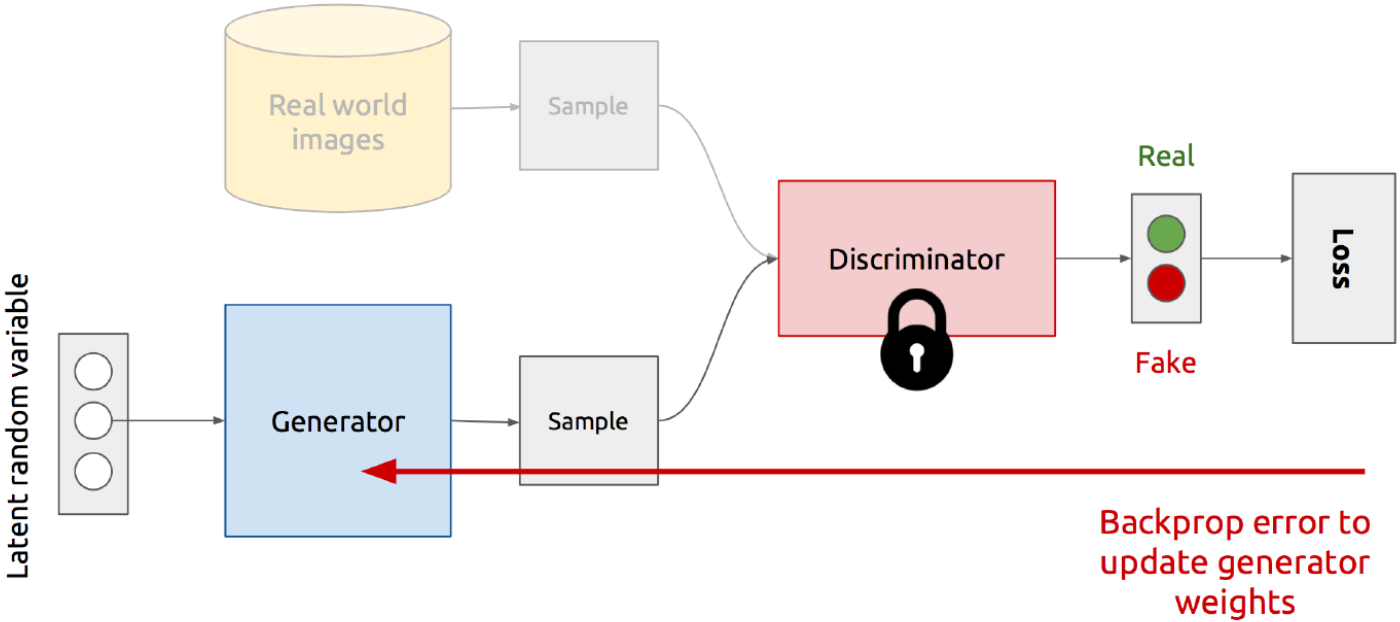
- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image



Training Discriminator



Training Generator



GAN's formulation

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

GAN's Pseudo-Code

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

Discriminator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

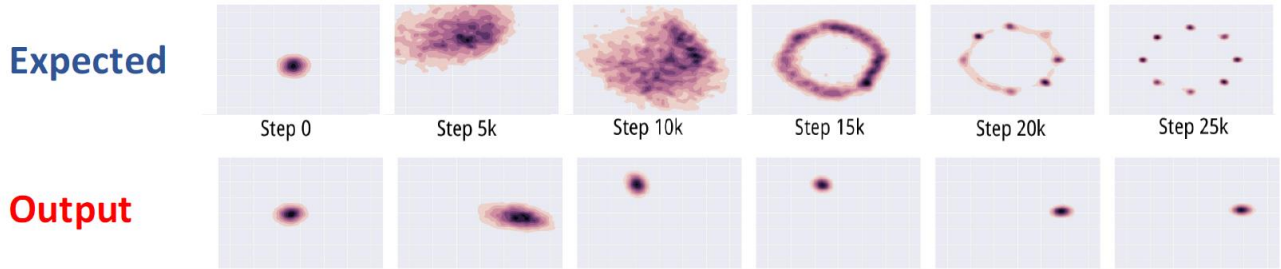
Instability of GAN

- Vanishing gradient

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

  - $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
  - Gradient goes to 0 if D is confident, i.e. $D\left(G(z)\right) \to 0$
  - Minimize $-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]$ for **Generator** instead (keep Discriminator as it is)

- Mode collapse



The Idea of GAN Inversion

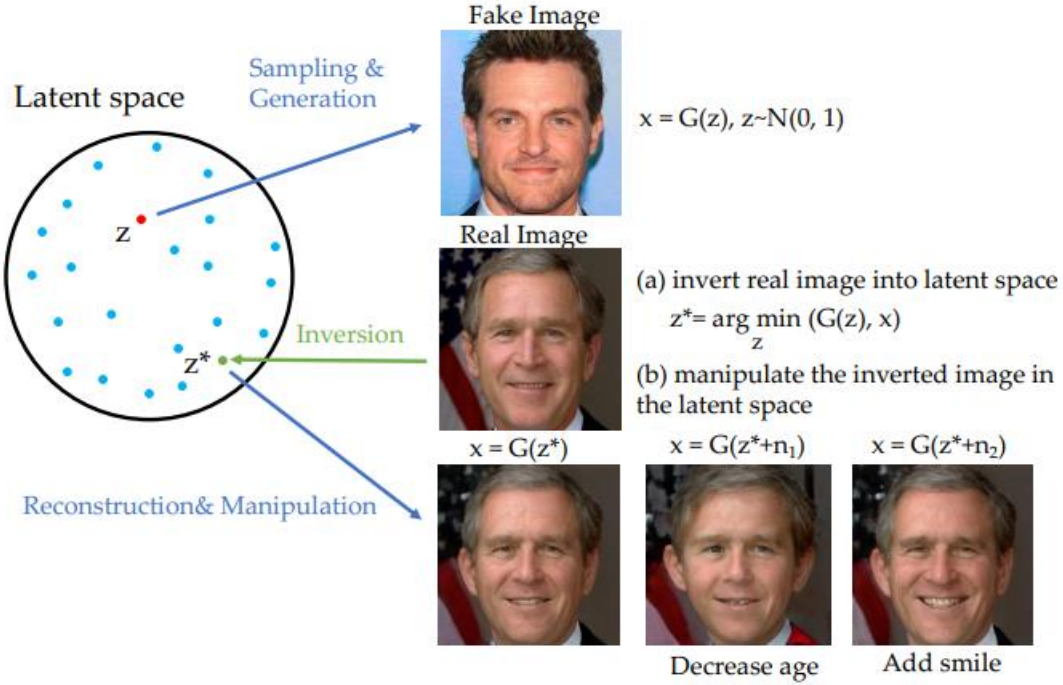
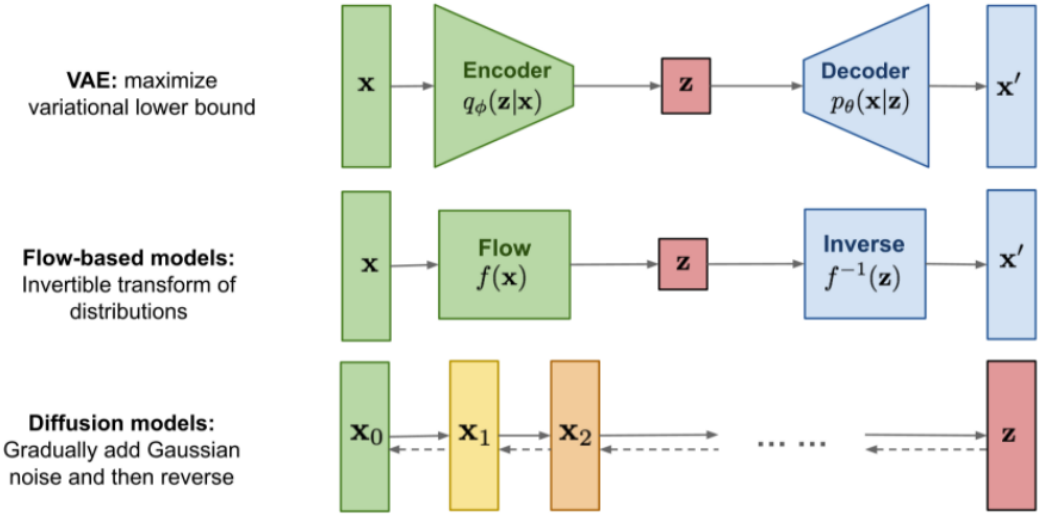


Fig. 1. **Illustration of GAN inversion.** Different from the conventional sampling and generation process using trained generator G, GAN inversion maps a given real image x to the latent space and obtains the latent code $\mathbf{z}^*$. The reconstructed image x^* is then obtained by $x^* = G(\mathbf{z}^*)$. By varying the latent code $\mathbf{z}^*$ in different interpretable directions *e.g.*, $\mathbf{z}^* + \mathbf{n_1}$ and $\mathbf{z}^* + \mathbf{n_2}$ where $\mathbf{n_1}$ and $\mathbf{n_2}$ model the age and smile in the latent space respectively, we can edit the corresponding attribute of the real image. The reconstructed results are from [22].

Diffusion Models

- Inspired by non-equilibrium thermodynamics
- construct reversible transform without specialized architectures



Credit: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

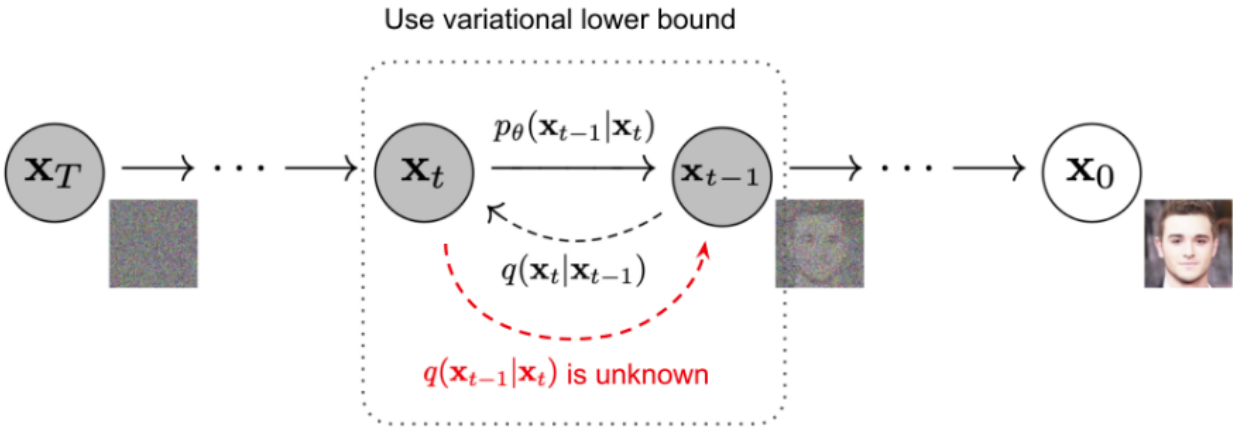
diffusion probabilistic models (Sohl-Dickstein et al., 2015), noise-conditioned score network (NCSN; Yang & Ermon, 2019), and denoising diffusion probabilistic models (DDPM; Ho et al. 2020).

Forward diffusion process

- add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

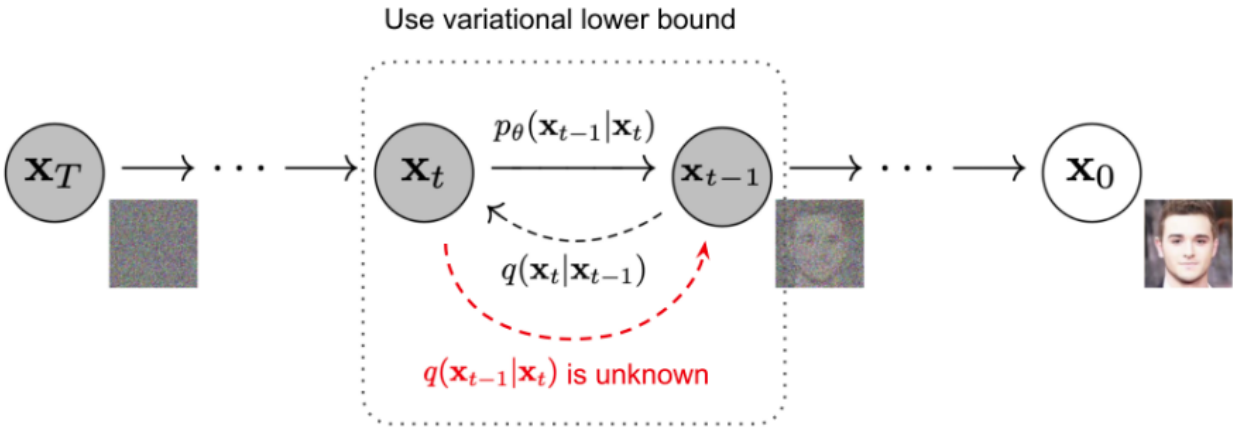
- the data sample x_0 gradually loses its distinguishable features as the step t becomes larger. Eventually when $T \to \infty$, x_T is equivalent to an isotropic Gaussian distribution.



Reverse diffusion process

- learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$



Reverse diffusion process

It is noteworthy that the reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

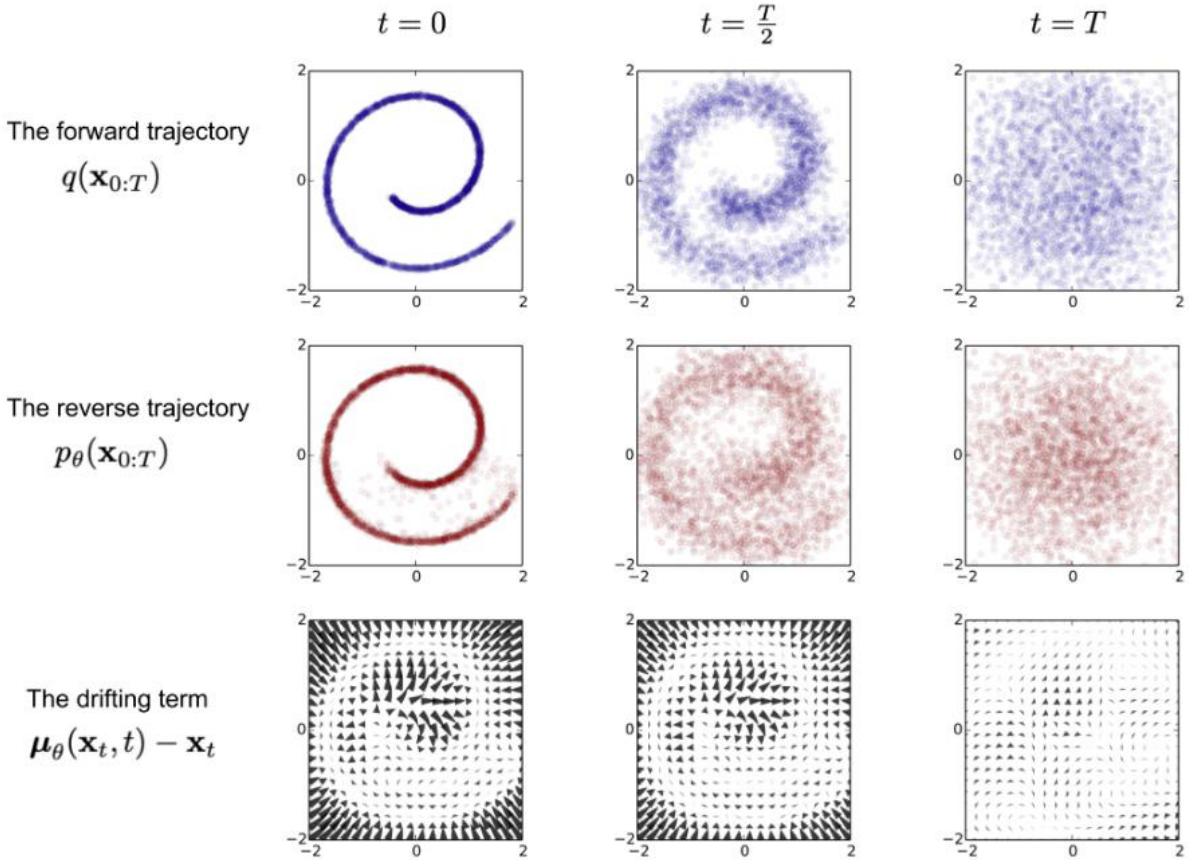
Using Bayes' rule, we have:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &\propto \exp\Big(-\frac{1}{2} \big(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t} \big) \Big) \\ &= \exp\Big(-\frac{1}{2} \big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) \mathbf{x}_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0) \mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0) \big) \Big) \end{aligned}$$

where $C(\mathbf{x}_t, \mathbf{x}_0)$ is some function not involving $\mathbf{x}_{t-1}$ and details are omitted. Following the standard Gaussian density function, the mean and variance can be parameterized as follows:

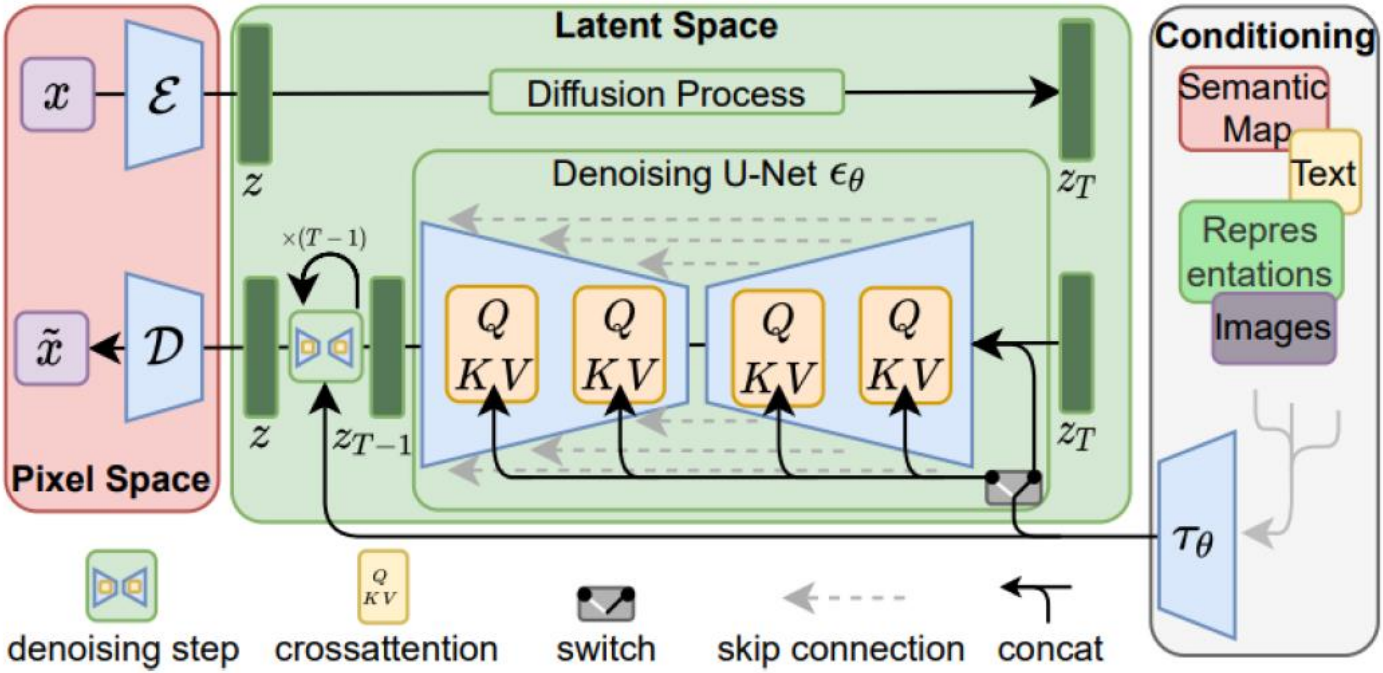
$$\begin{aligned} \tilde{\beta}_t &= 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0) / (\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 \end{aligned}$$

Diffusion Examples



Latent Diffusion Models

- Decouple perceptual and semantic signals

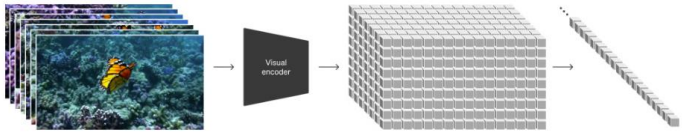


OpenAI Sora: State-of-The-Art Diffusion Model

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.

https://openai.com/sora

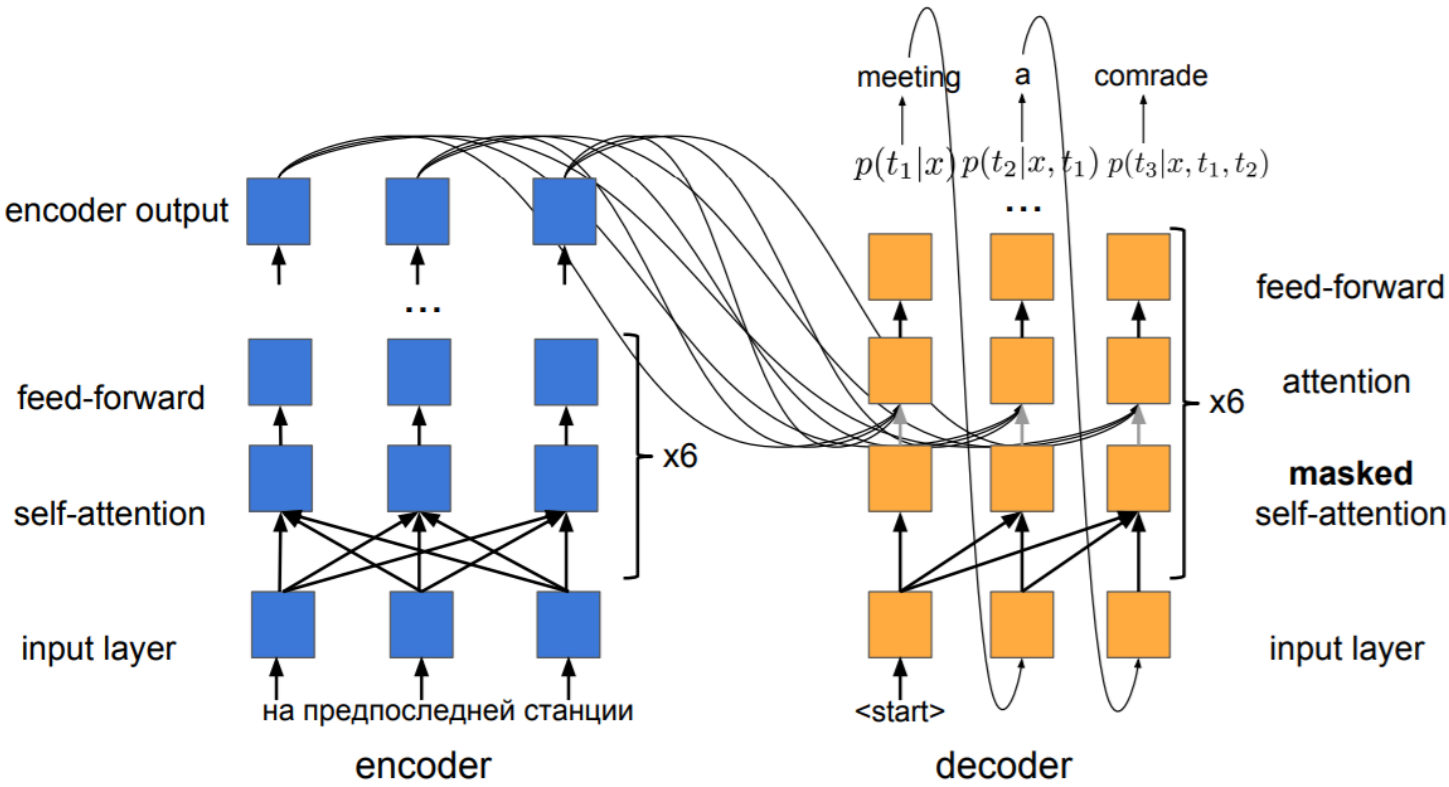
Latent spatio-temporal patch



Diffusion with Transformer (DiT)

Robin Rombach et al. High-resolution image synthesis with latent diffusion models, 2021

Recall: Transformer: “Attention is All You Need”



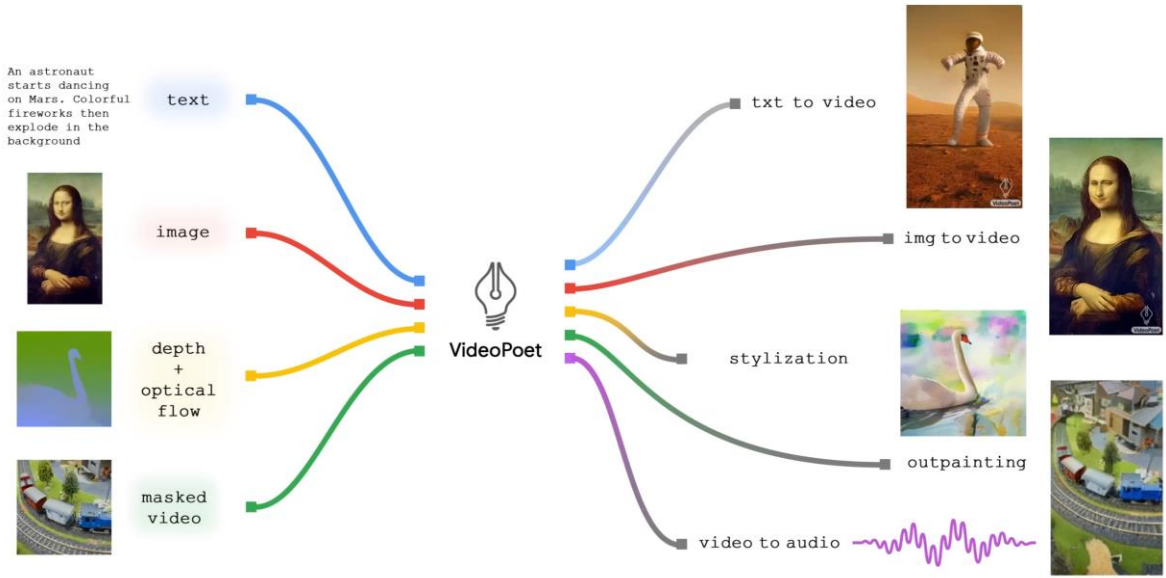
Training of GPT-Style Generative Models

预训练阶段	指令微调	奖励函数	强化学习
原始数据 数千亿单词：图书、百科、网页等	标注用户指令 数万用户指令和对应的答案	标注对比对 百万标注对比对	用户指令 十万用户指令
语言模型预训练	语言模型预训练	二分类模型	强化学习方法
基础模型	SFT 模型	RM 模型	RL 模型
1000+GPU 月级别训练时间 GPT 3.0、LLaMa、PaLM	1-100GPU 天级别训练时间 MOSS、ChatGLM6b、Vicuna-13B等	1-100GPU 天级别训练时间	1-100GPU 天级别训练时间 ChatGPT、Claude

State of GPT, Microsoft Build 2023, Andrej Karpathy

Autoregressive Models

- Representative works: Phenaki (Google 2022), Make-A-Video, NUWA, VideoGPT and CogVideo, VideoPoet (Google), Video-LaVIT (PKU & Kuaishou)
- **VideoPoet**: An autoregressive language model learns across video, image, audio, and text modalities to autoregressively predict the next video or audio token in the sequence.



Questions?

Email: myd@pku.edu.cn
Website: http://www.muyadong.com