Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

Lecture 9.2: New Generative Models

Paul Liang

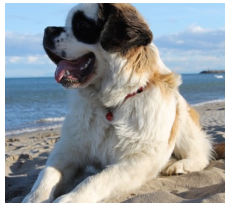
** Co-lecturer: Louis-Philippe Morency. Original course co-developed with Tadas Baltrusaitis. Spring 2021 and 2022 editions taught by Yonatan Bisk. Spring 2023 edition taught by Yonatan and Daniel Fried*

Generation

Definition: Learning a generative process to produce raw modalities that reflects cross-modal interactions, structure, and coherence.

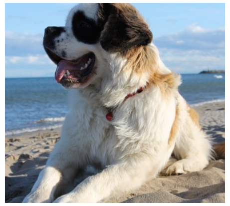
Summarization

(video)



Big dog on the beach

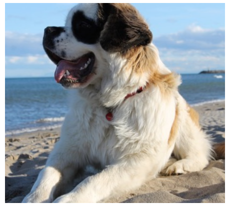
Translation



Big dog on the beach

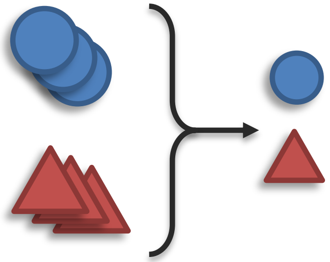
Creation

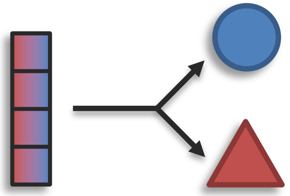
z_y



Big dog on the beach

'woof'
'crash?'





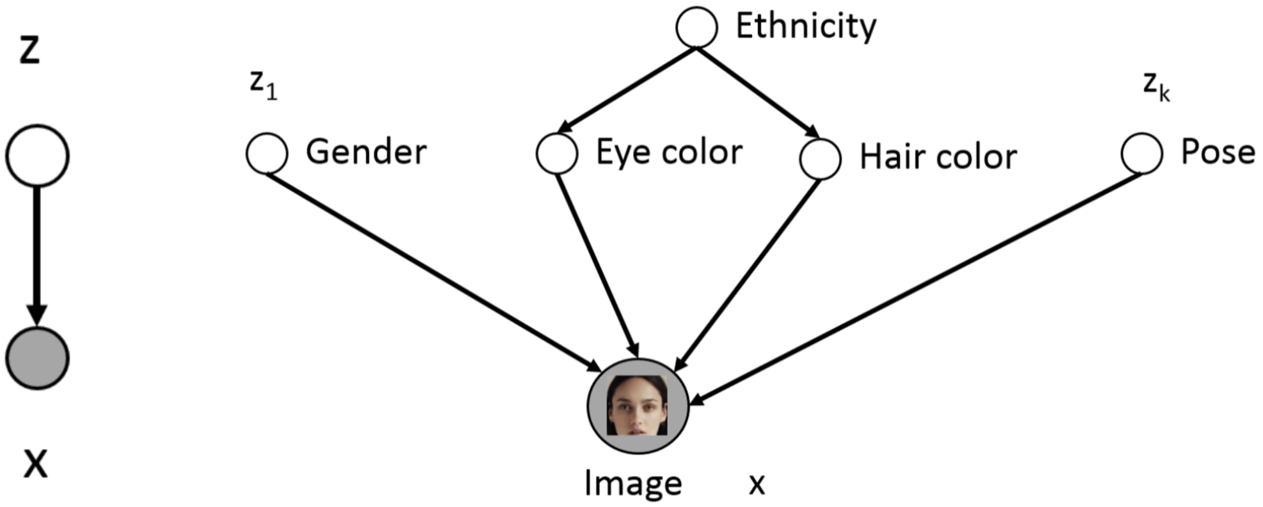
Information:
(content)

Reduction	Maintenance	Expansion
□ > □	□ = □	□ < □

Latent Variable Models

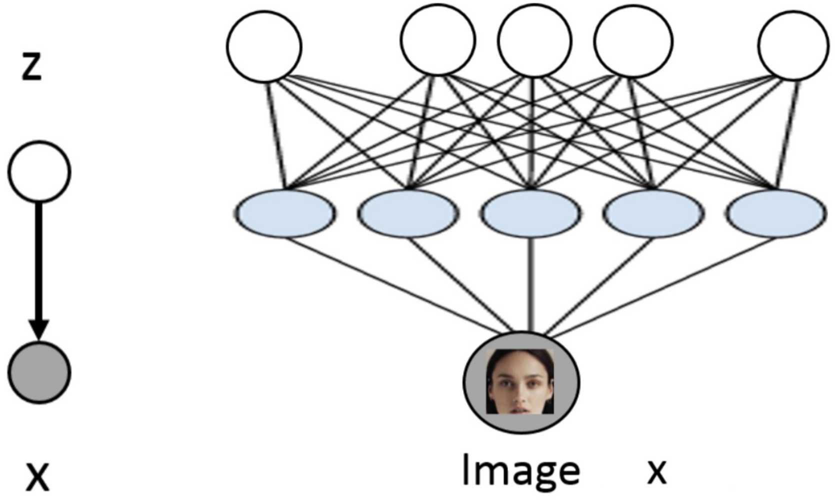
- Lots of variability in images **x** due to gender, eye color, hair color, pose, etc.
- However, unless images are annotated, these factors of variation are not explicitly available (latent).
- Idea: explicitly model these factors using latent variables **z**

Latent Variable Models



- Only shaded variables **x** are observed in the data
- Latent variables **z** are unobserved - correspond to high-level features
  - We want z to represent useful features e.g. hair color, pose, etc.
  - But very difficult to specify these conditionals by hand and they're unobserved
  - Let's **learn** them instead

Latent Variable Models



- Put a prior on z $\mathbf{z} \sim \mathcal{N}(0, I)$

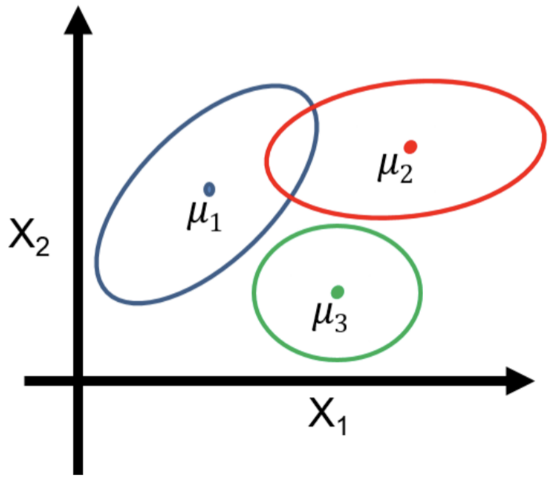
  $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
- Hope that after training, z will correspond to meaningful latent factors of variation - useful features for unsupervised representation learning
- Given a new image x, features can be extracted via p(zlx)
- Given a random z, a new x can be generated => control; if z is interpretable

Mixture of Gaussians

Mixture of Gaussians (Bayes network z -> x)

$$\mathbf{z} \sim \mathrm{Categorical}(1, \cdots, K)$$

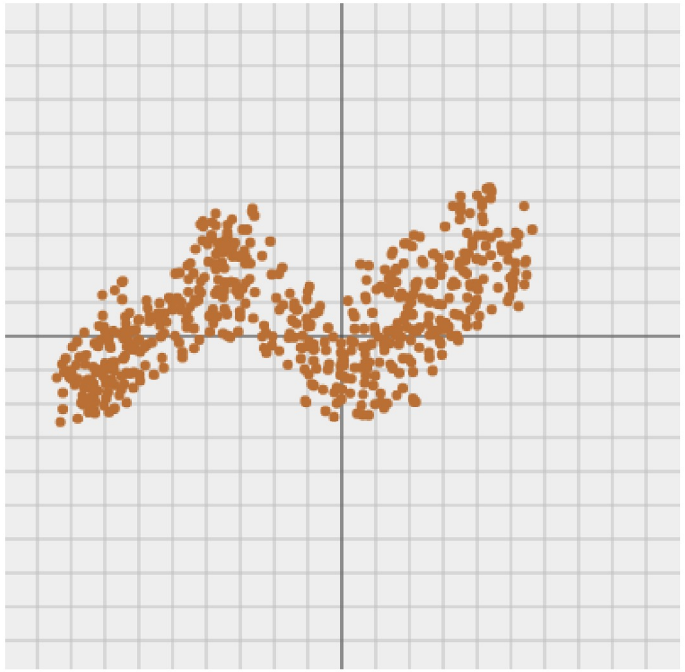
$$p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mu_k, \Sigma_k)$$

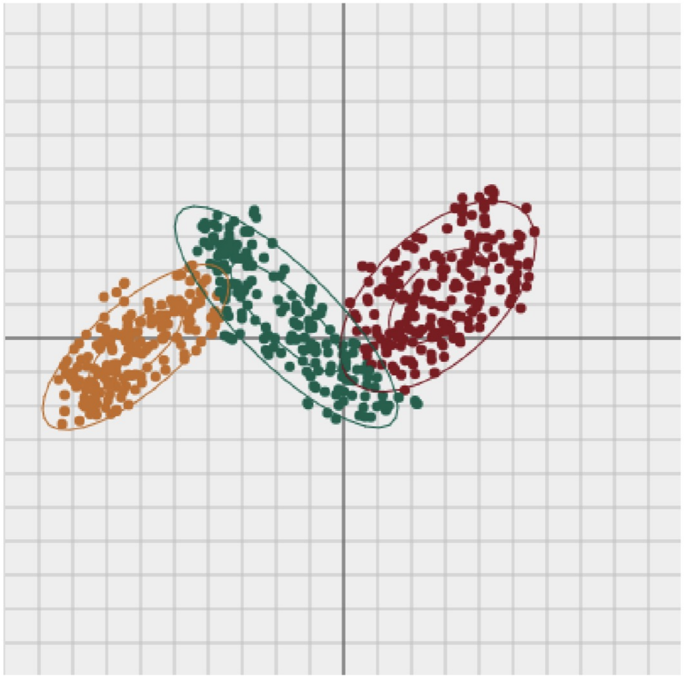


Generative process

1. Pick a mixture component by sampling z
2. Generate a data point by sampling from that Gaussian

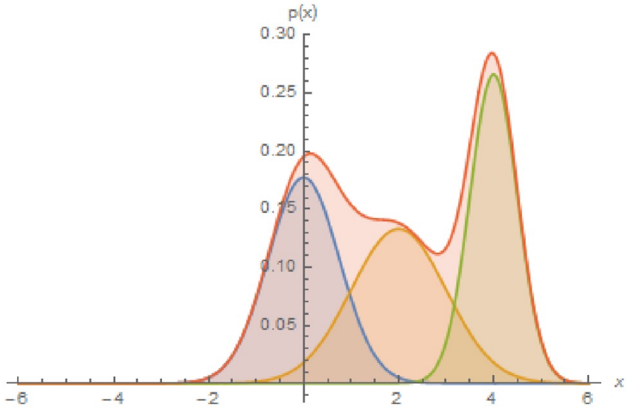
Mixture of Gaussians





Mixture of Gaussians

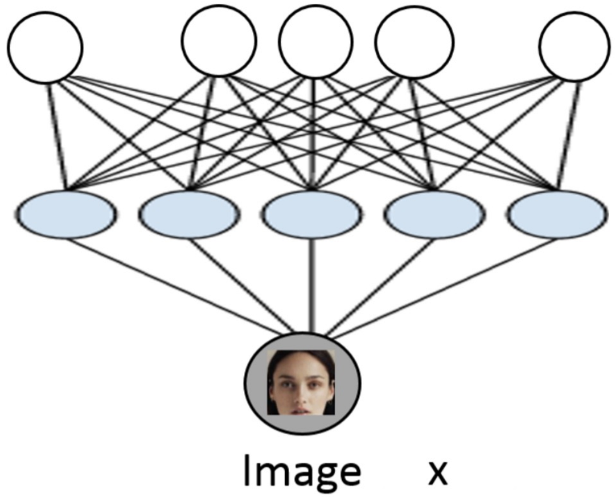
Combining simple models into more expressive ones



$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) = \sum_{k=1}^{K} p(\mathbf{z} = k) \underbrace{\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)}_{\text{component}}$$

can solve using expectation maximization
Expectation: use mean and variance to estimate p(z=k)
Maximization: use estimate p(z=k) to update mean and variance

From GMMs to VAEs



- Put a prior on z $\mathbf{z} \sim \mathcal{N}(0, I)$

  $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ, Σ_θ are neural networks
- Hope that after training, z will correspond to meaningful latent factors of variation - useful features for unsupervised representation learning
- Even though p(x|z) is simple, marginal p(x) is much richer/complex/flexible

Learning parameters of VAEs

- Learning parameters of VAE: we have a joint distribution $p(\mathbf{X}, \mathbf{Z}; \theta)$
- We have a dataset **D** where for each datapoint the **x** variables are observed (e.g. images, text) and the variables **z** are not observed (latent variables)
- We can try maximum likelihood estimation:

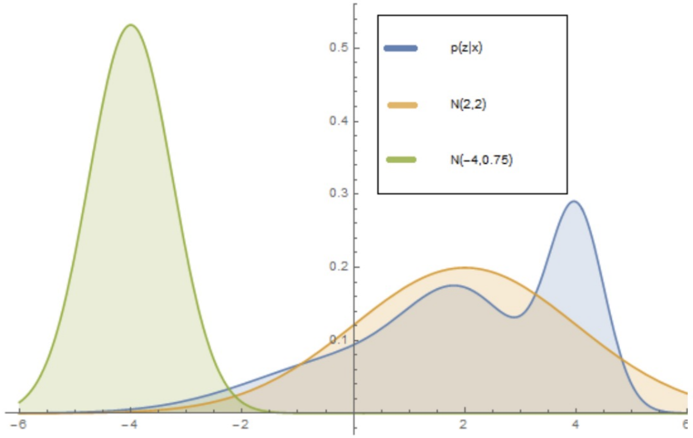
$$\log \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x}; \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}; \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log \underbrace{\sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta)}$$

Need cheaper approximations to optimize for VAE parameters

intractable :-(

- if z binary with 30 dimensions, need sum 2^30 terms
- if z continuous, integral is impossible

Variational Inference



Suppose $q(\mathbf{z};\phi)$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)

- For example, a Gaussian with mean and covariance specified by ϕ

$$q(\mathbf{z};\phi) = \mathcal{N}(\phi_1, \phi_2)$$

- Variational inference: optimize variational parameters so that $q(\mathbf{z};\phi)$ is as close as possible to $p(\mathbf{x},\mathbf{z};\theta)$ while being simple to compute
- E.g. in figure, posterior (in blue) is better approximated by orange Gaussian than green

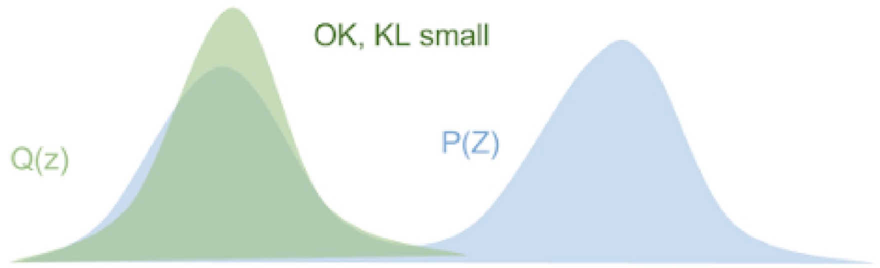
[Slides from Ermon and Grover]

KL Divergence

- The KL divergence for variational inference is:

$$\mathbf{D}_{KL}(q(z)\|p(z|x)) = \int q(z)\log\frac{q(z)}{p(z|x)}dz$$

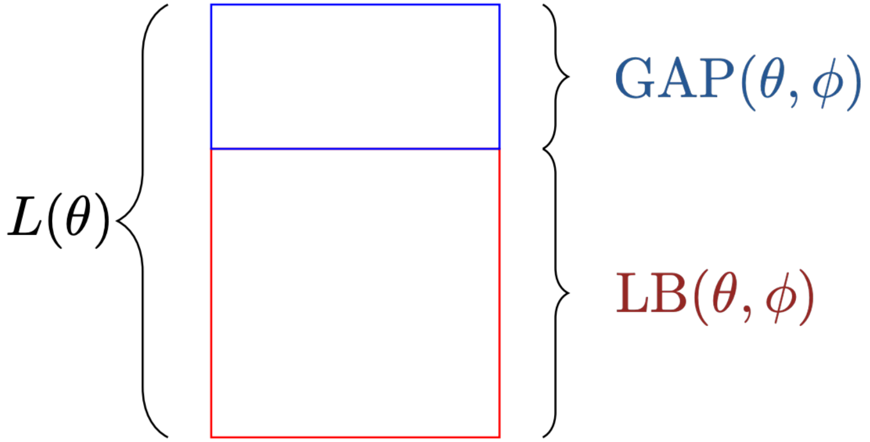
- Intuitively, there are three cases
  a. If **q** is low then we don't care (because of the expectation).
  b. If **q** is high and **p** is high then we are happy.
  c. If **q** is high and **p** is low then we pay a price.
- Note that p must be > 0 wherever q > 0



[Slides from Ermon and Grover]

Variational Inference

High-level: decompose objective into lower-bound and gap.



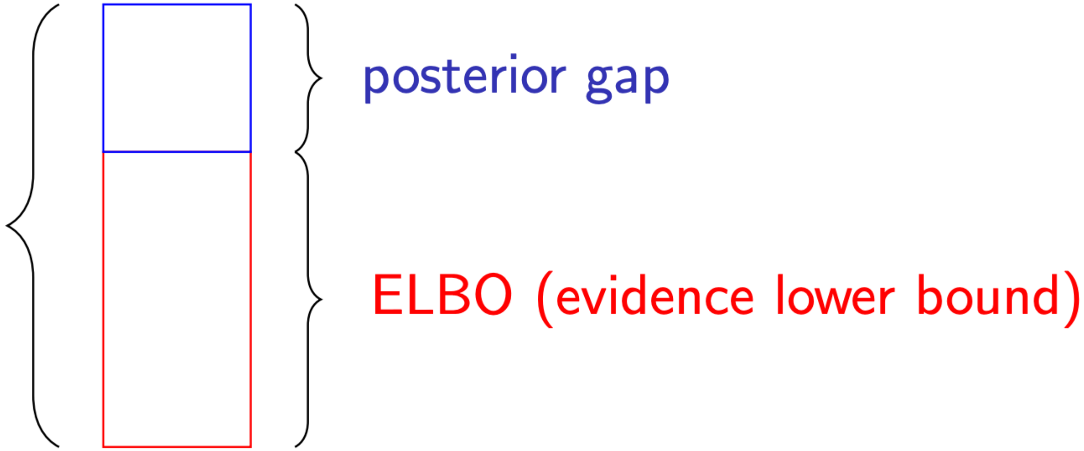
$$\log \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x}; \theta) = L(\theta) = \mathrm{LB}(\theta, \phi) + \mathrm{GAP}(\theta, \phi) \text{ for some } \phi$$

Provides framework for deriving a rich set of optimization algorithms.

Variational Inference

For any[1] distribution $q(z|x;\phi)$ over z,

$$L(\theta) = \mathbb{E}_q \log \frac{p(x,z;\theta)}{q(z|x;\phi)} + \mathrm{KL}[q(z|x;\phi) \,||\, p(z|x;\theta)]$$



Since KL is always non-negative, $L(\theta) \geq \mathrm{ELBO}$

[1] Technical condition: $\mathrm{supp}(q(z)) \subset \mathrm{supp}(p(z \mid x;\, \theta))$

Variational Inference

$$\log p(x;\,\theta) = \mathbb{E}_q \log p(x) \quad \textit{(Expectation over } z\textit{)}$$

$$= \mathbb{E}_q \log \frac{p(x,z)}{p(z\,|\,x)} \quad \textit{(Mult/div by } p(z|x)\textit{, combine numerator)}$$

$$= \mathbb{E}_q \log \left(\frac{p(x,z)}{q(z\,|\,x)} \frac{q(z\,|\,x)}{p(z\,|\,x)} \right) \quad \textit{(Mult/div by } q(z|x)\textit{)}$$

$$= \mathbb{E}_q \log \frac{p(x,z)}{q(z\,|\,x)} + \mathbb{E}_q \log \frac{q(z\,|\,x)}{p(z\,|\,x)} \quad \textit{(Split Log)}$$

$$= \underbrace{\mathbb{E}_q \log \frac{p(x,z;\theta)}{q(z|x;\phi)}}_{\text{Evidence Lower Bound (ELBO)}} + \underbrace{\mathrm{KL}[q(z|x;\phi)\;||\;p(z|x;\theta)]}_{\text{posterior gap}}$$

q: like an encoder network:
data -> latent

Huge number of algorithms from choices of q and decompositions of ELBO

We'll ignore this term from now on

Evidence Lower Bound (ELBO)
We'll choose q, and parameters θ, ϕ to maximize this

posterior gap
Typically uncomputable, but hopefully small if we chose q well

Slides from Kim, Wiseman, and Rush, https://nlp.seas.harvard.edu/latent-nlp-tutorial.html

Learning parameters of VAEs

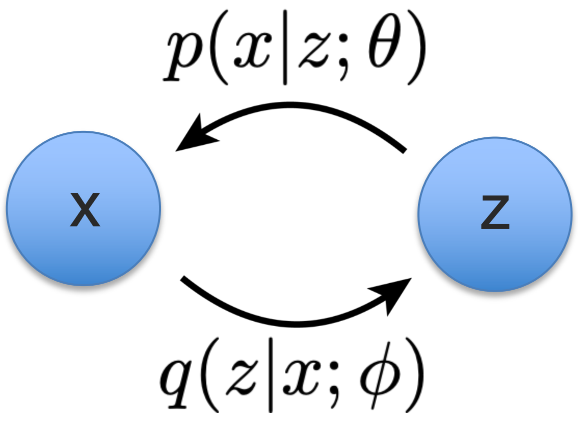
Evidence Lower Bound (ELBO)

$$\mathbb{E}_q \log \frac{p(x,z;\theta)}{q(z|x;\phi)} = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x})]$$

$$= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$$

$$= \underbrace{E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)]}_{\text{Reconstruct the input from features z}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{Prior prevents hint from being too informative (different KL than before!)}}$$

Generative/decoder



Inference/encoder

1. Take a data point $\mathbf{x}^i$
2. Map it to $\hat{\mathbf{z}}$ by sampling from $q_\phi(\mathbf{z}|\mathbf{x}^i)$ (*encoder*)
3. Reconstruct $\hat{\mathbf{x}}$ by sampling from $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$ (*decoder*)

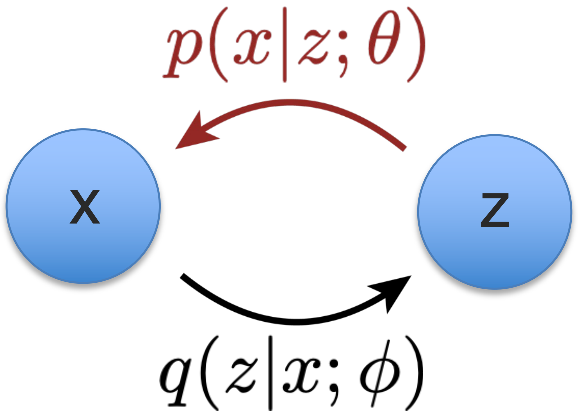
What does the training objective $\mathcal{L}(\mathbf{x};\theta,\phi)$ do?

- First term encourages $\hat{\mathbf{x}} \approx \mathbf{x}^i$ ($\mathbf{x}^i$ likely under $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$)
- Second term encourages $\hat{\mathbf{z}}$ to be likely under the prior $p(\mathbf{z})$

[Slides from Ermon and Grover]

Learning parameters of VAEs

$$\mathcal{L}(\mathbf{x};\theta,\phi) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

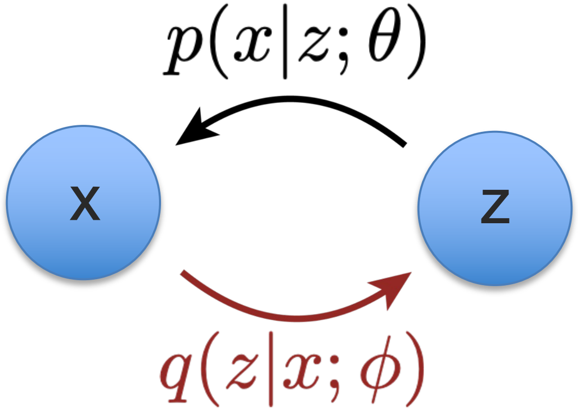


- We need to compute the gradients $\underbrace{\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{easy}}$ and $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\mathrm{x};\theta,\phi) &= \nabla_\theta E_{q_\phi(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x}|\mathrm{z};\theta)] - D_{KL}(q_\phi(\mathrm{z}|\mathrm{x})||p(\mathrm{z})) \\
&= \nabla_\theta E_{q_\phi(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x}|\mathrm{z};\theta)] \\
&= E_{q_\phi(\mathrm{z}|\mathrm{x})}[\nabla_\theta \log p(\mathrm{x}|\mathrm{z};\theta)] \\
&\approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p(\mathrm{x}|\mathrm{z}_i;\theta)
\end{aligned}$$

Learning parameters of VAEs

$$\mathcal{L}(\mathbf{x};\theta,\phi) \quad = \quad E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$



- We need to compute the gradients $\underbrace{\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{easy}}$ and $\underbrace{\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{tricky}}$

- Expectations also depend on ϕ

$$\nabla_\phi \mathcal{L}(\mathrm{x};\theta,\phi) = \nabla_\phi E_{q_\phi(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x}|\mathrm{z};\theta)] - D_{KL}(q_\phi(\mathrm{z}|\mathrm{x})||p(\mathrm{z}))$$

Reparameterization Trick

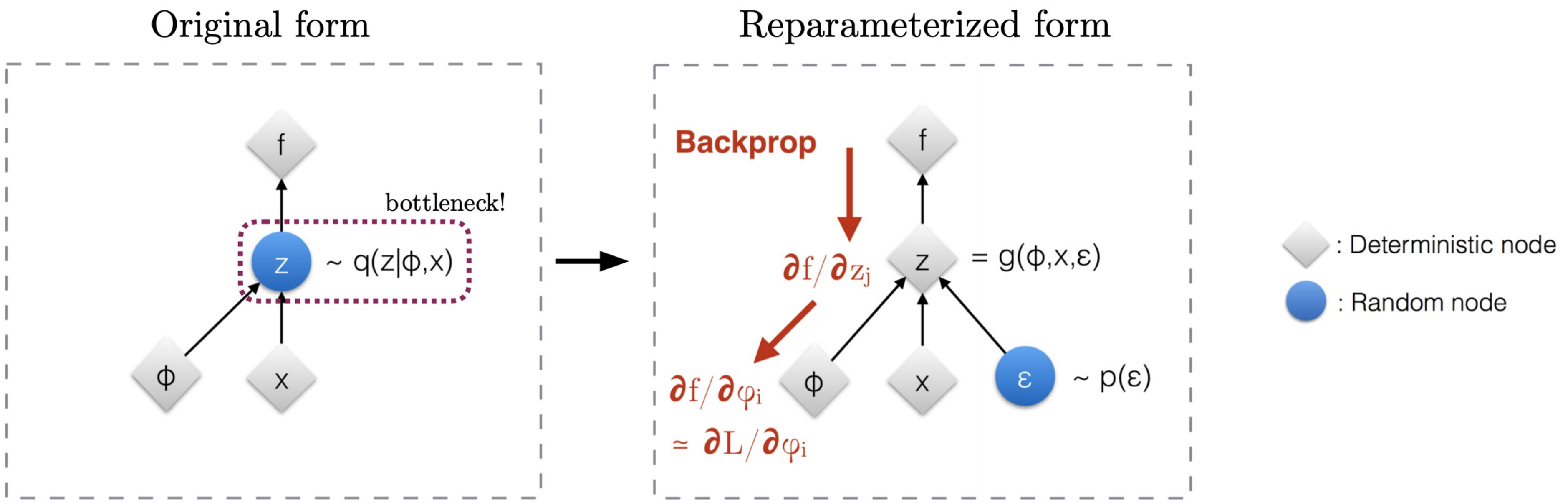
- Want to compute a gradient with respect to ϕ of
$$E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \int q(\mathbf{z};\phi)r(\mathbf{z})d\mathbf{z}$$
where $\mathbf{z}$ is now **continuous**
- Suppose $q(\mathbf{z};\phi) = \mathcal{N}(\mu, \sigma^2 I)$ is Gaussian with parameters $\phi = (\mu, \sigma)$. These are equivalent ways of sampling:
  - Sample $\mathbf{z} \sim q_\phi(\mathbf{z})$
  - Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon = g(\epsilon;\phi)$
- Using this equivalence we compute the expectation in two ways:
$$E_{\mathbf{z}\sim q(\mathbf{z};\phi)}[r(\mathbf{z})] = E_{\epsilon\sim\mathcal{N}(0,I)}[r(g(\epsilon;\phi))] = \int p(\epsilon)r(\mu+\sigma\epsilon)d\epsilon$$
$$\nabla_\phi E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \nabla_\phi E_\epsilon[r(g(\epsilon;\phi))] = E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))]$$
- Easy to estimate via Monte Carlo if r and g are differentiable w.r.t. ϕ and ϵ is easy to sample from (backpropagation)
- $E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))] \approx \frac{1}{k}\sum_k \nabla_\phi r(g(\epsilon^k;\phi))$ where $\epsilon^1, \cdots, \epsilon^k \sim \mathcal{N}(0, I)$.

[Slides from Ermon and Grover]

Reparameterization Trick

$$\nabla_\phi \mathcal{L}(\mathrm{x};\theta,\phi) = \nabla_\phi E_{q_\phi(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x}|\mathrm{z};\theta)] - D_{KL}(q_\phi(\mathrm{z}|\mathrm{x})||p(\mathrm{z}))$$

$$\begin{aligned}\nabla_\phi E_{q_\phi(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x}|\mathrm{z};\theta)] &= \nabla_\phi E_\epsilon[\log p(\mathrm{x}|\mu+\sigma\epsilon;\theta)] \quad \textbf{reparameterize} \\ &= E_\epsilon[\nabla_\phi \log p(\mathrm{x}|\mu+\sigma\epsilon;\theta)] \\ &\approx \frac{1}{n}\sum_{i=1}^{n}[\nabla_\phi \log p(\mathrm{x}|\mu+\sigma\epsilon_i;\theta)]\end{aligned}$$



[Slides from Ermon and Grover]

Stochastic Optimization

$$\max_{\phi} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

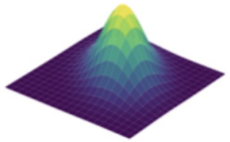
VAEs

$$\max_{\theta,\phi} \mathcal{L}(x;\theta,\phi)$$ Evidence lower bound

$$\max_{\theta,\phi} E_{q_\phi(z|x)}[\log p(x|z;\theta)]$$

Solve by reparameterization!

$$x \xleftarrow{p(x|z;\theta)} z \xleftarrow{q_\phi(z|x)} x$$



We require that:
- z is continuous
- q(z) is reparameterizable
- f(z) is differentiable wrt ϕ

- Sample $\mathbf{z} \sim q_\phi(\mathbf{z})$
- Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon$

RL

$$\max_{\phi} J(\phi)$$ Reward

$$\max_{\phi} E_{\tau\sim p(\tau;\phi)}[r(\tau)]$$

Reparameterization???

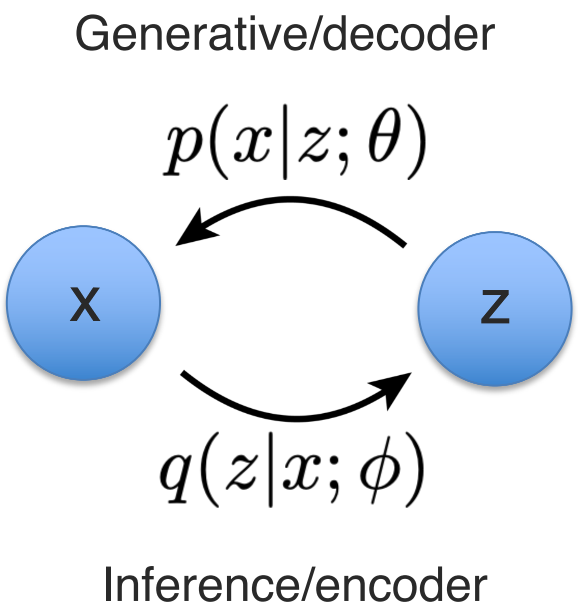
In RL (at least for discrete actions):
- T is a sequence of discrete actions
- p(T; ϕ) is not reparameterizable
- r(T) is a black box function
i.e. the environment

$$s \xrightarrow{\pi_\phi(a|s)} a \xrightarrow{???} r$$

REINFORCE is a general-purpose solution!

Learning parameters of VAEs

$$
\begin{aligned}
\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= \underbrace{E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{prior}}
\end{aligned}
$$

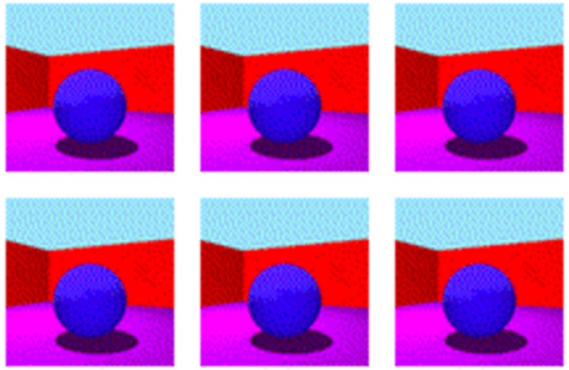


1. Take a datapoint x_i.
2. Map it to μ, σ using $q_\phi(z|x_i)$. **encoder**
3. Sample $\epsilon \sim N(0, I)$ and compute $\hat{z} = \mu + \sigma\epsilon$. **reparameterize**
4. Reconstruct $\hat{x}$ by sampling from $p(x|\hat{z};\theta)$. **decoder**

VAEs for Disentangled Generation

Disentangled representation learning

- Very useful for style transfer: disentangling **style** from **content**



disentanglement_lib

From negative to positive
consistently slow . consistently good . consistently fast .
my goodness it was so gross . my husband 's steak was phenomenal . my goodness was so awesome .
it was super dry and had a weird taste to the entire slice . it was a great meal and the tacos were very kind of good . it was super flavorful and had a nice texture of the whole side .

[Locatello et al., Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. ICML 2019]

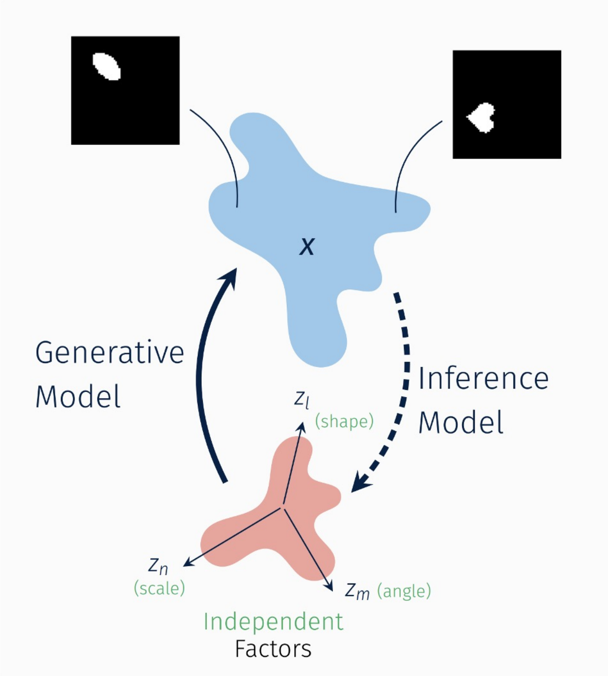
VAEs for Disentangled Generation

Disentangled representation learning

- Very useful for style transfer: disentangling **style** from **content**

$$\mathcal{L}_\beta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta \cdot \mathrm{KL}(q_\phi(z|x)||p(z))$$

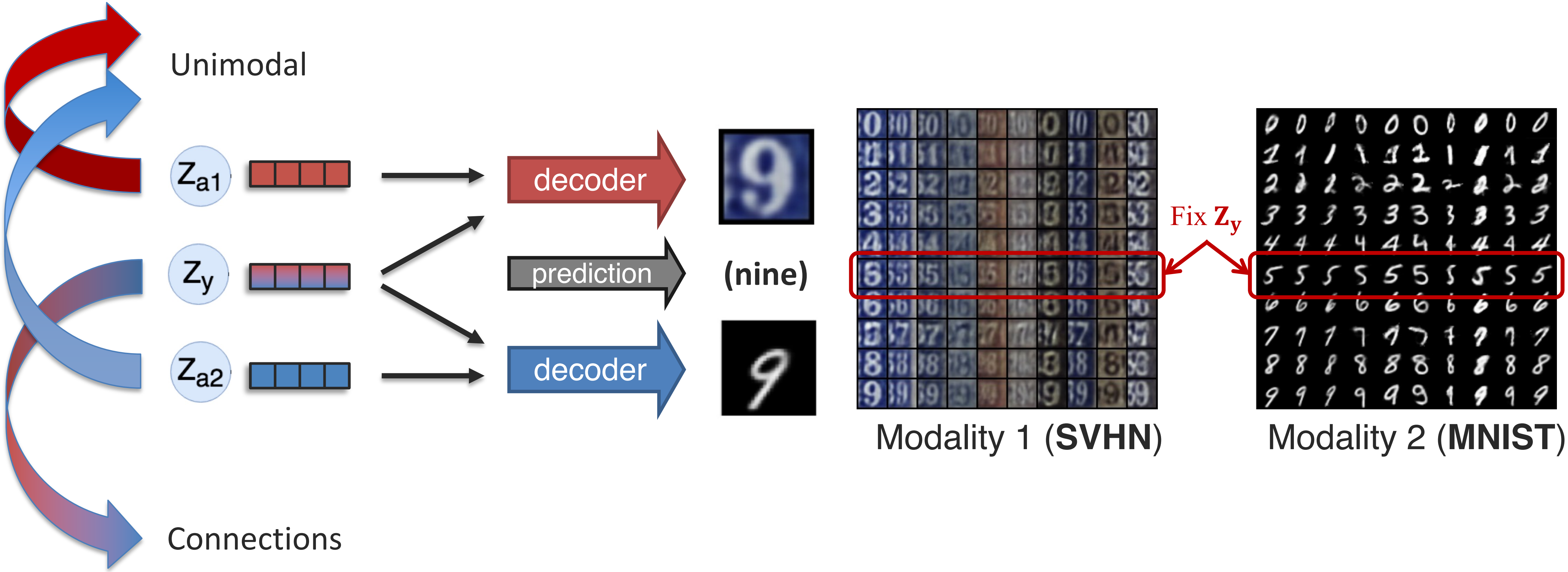
- beta-VAE: beta = 1 recovers VAE, beta > 1 imposes stronger constraint on the latent variables to have independent dimensions
- Difficult problem!
  - Positive results [Hu et al., 2016, Kulkarni et al., 2015]
  - Negative results [Mathieu et al., 2019, Locatello et al., 2019]
  - Better benchmarks & metrics to measure disentanglement [Higgins et al., 2017, Kim & Mnih 2018]



[Mathieu et al., Disentangling Disentanglement in Variational Autoencoders. ICML 2019]

VAEs for Multimodal Generation

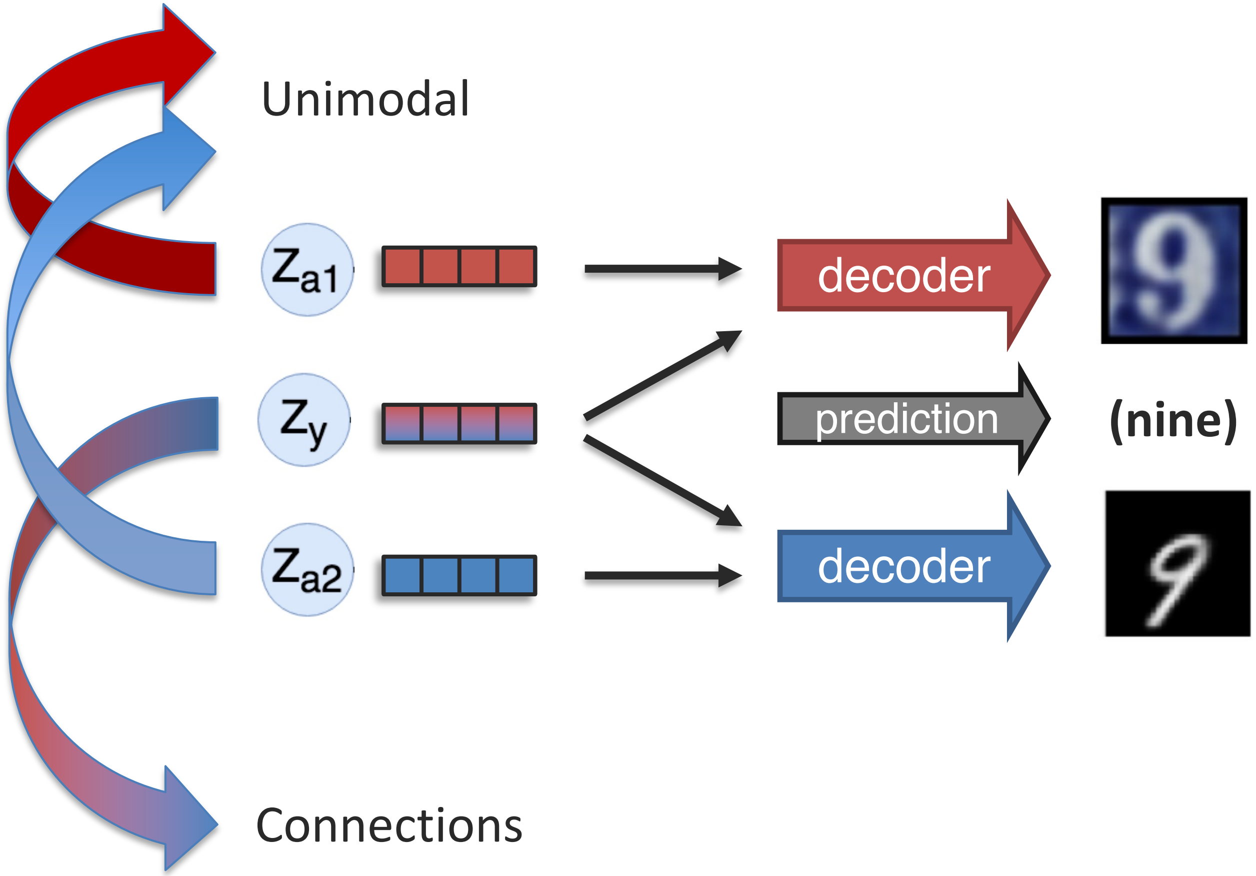
Some initial attempts: factorized generation

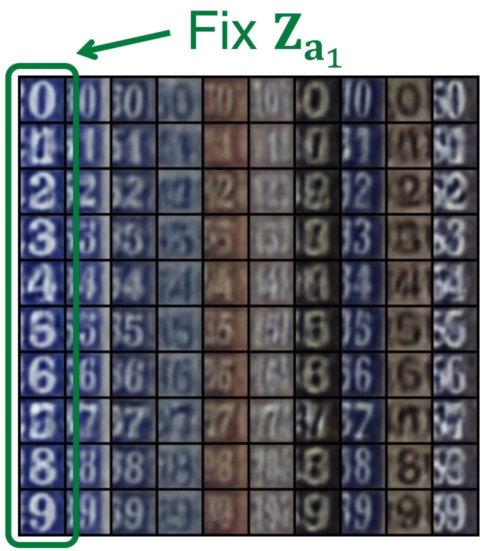


[Tsai et al., Learning Factorized Multimodal Representations. ICLR 2019]

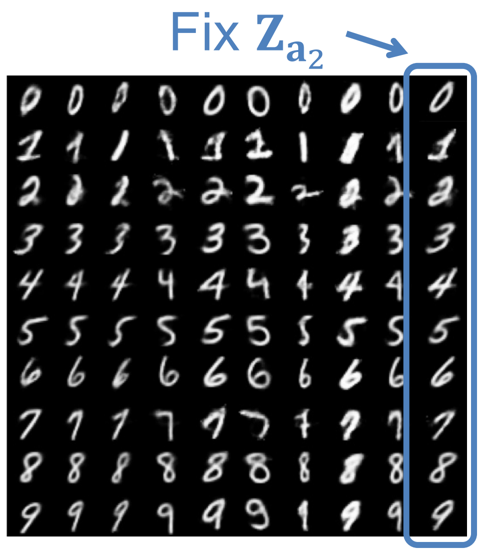
VAEs for Multimodal Generation

Some initial attempts: factorized generation





Modality 1 (**SVHN**)



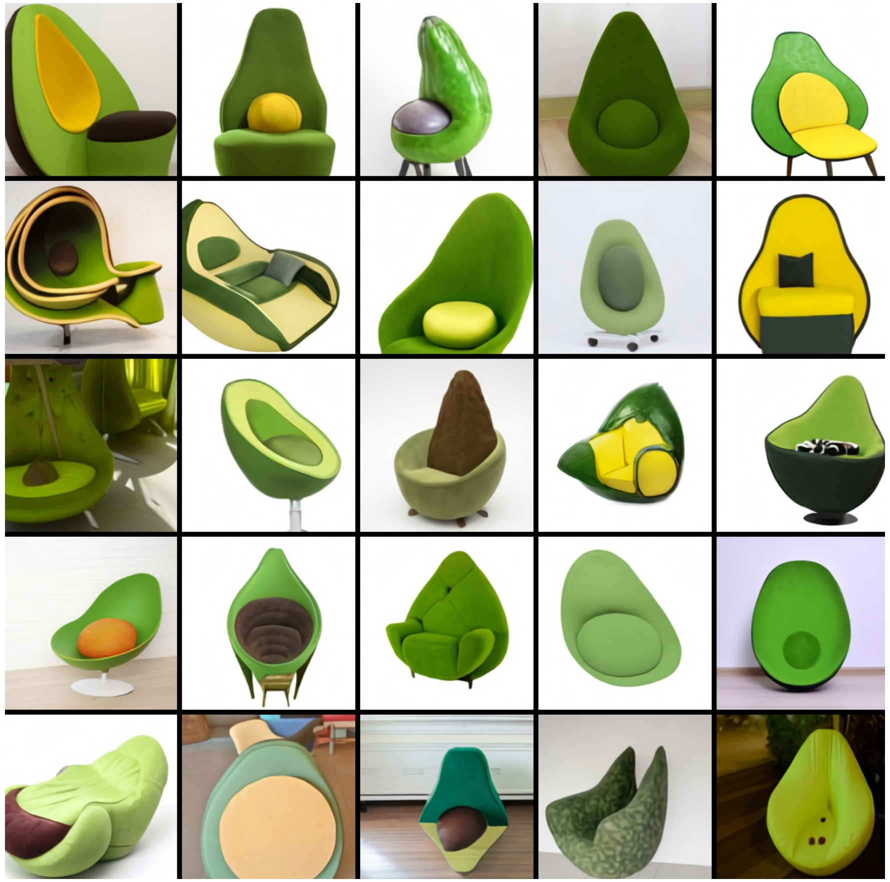
Modality 2 (**MNIST**)

[Tsai et al., Learning Factorized Multimodal Representations. ICLR 2019]

Image Tokens + Transformers

- Is this magic?

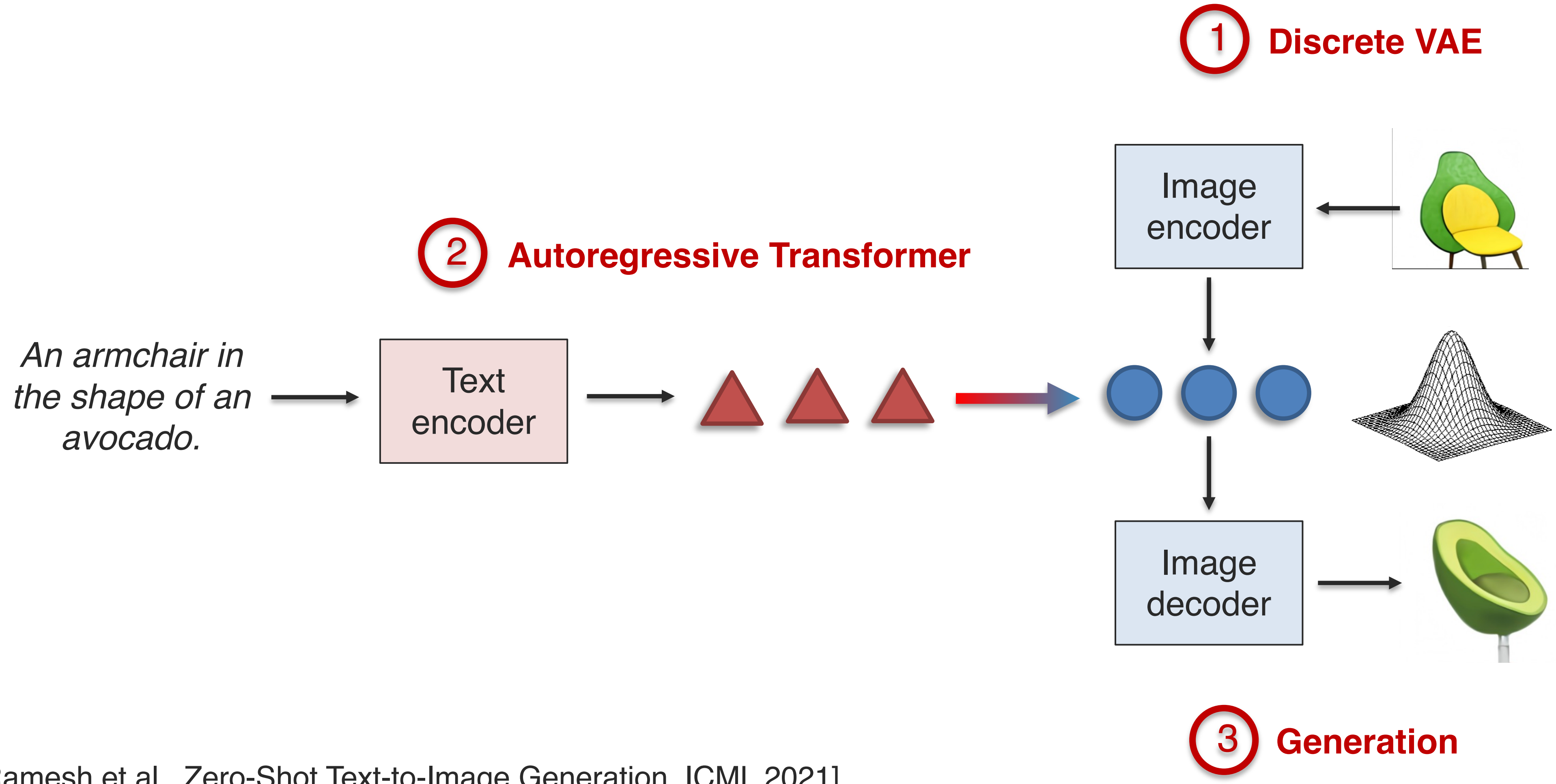
An armchair in the shape of an avocado →



[*DALL-E.* Ramesh et al., Zero-Shot Text-to-Image Generation. ICML 2021]

[see also, Esser et al. Taming Transformers for High Resolution Image Synthesis. CVPR 2021]

Image Tokens + Transformers



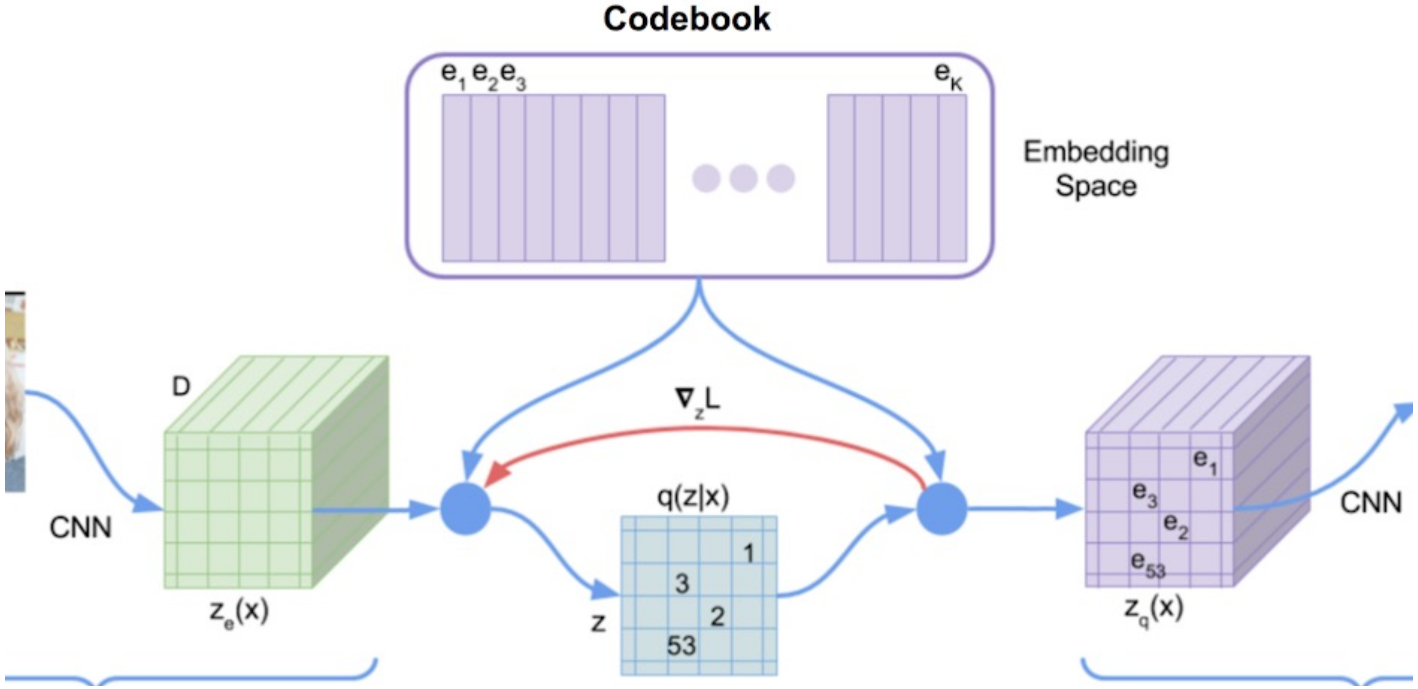
[*DALL-E.* Ramesh et al., Zero-Shot Text-to-Image Generation. ICML 2021]

[see also, Esser et al. Taming Transformers for High Resolution Image Synthesis. CVPR 2021]

https://arxiv.org/abs/2102.12092

DALL-E

(1) Discrete visual tokens from a VQ-VAE

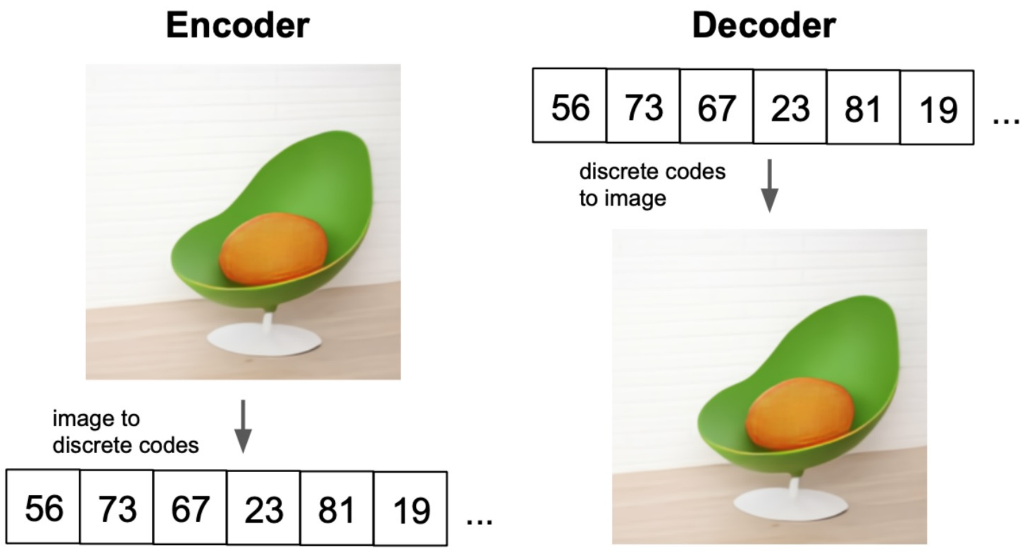


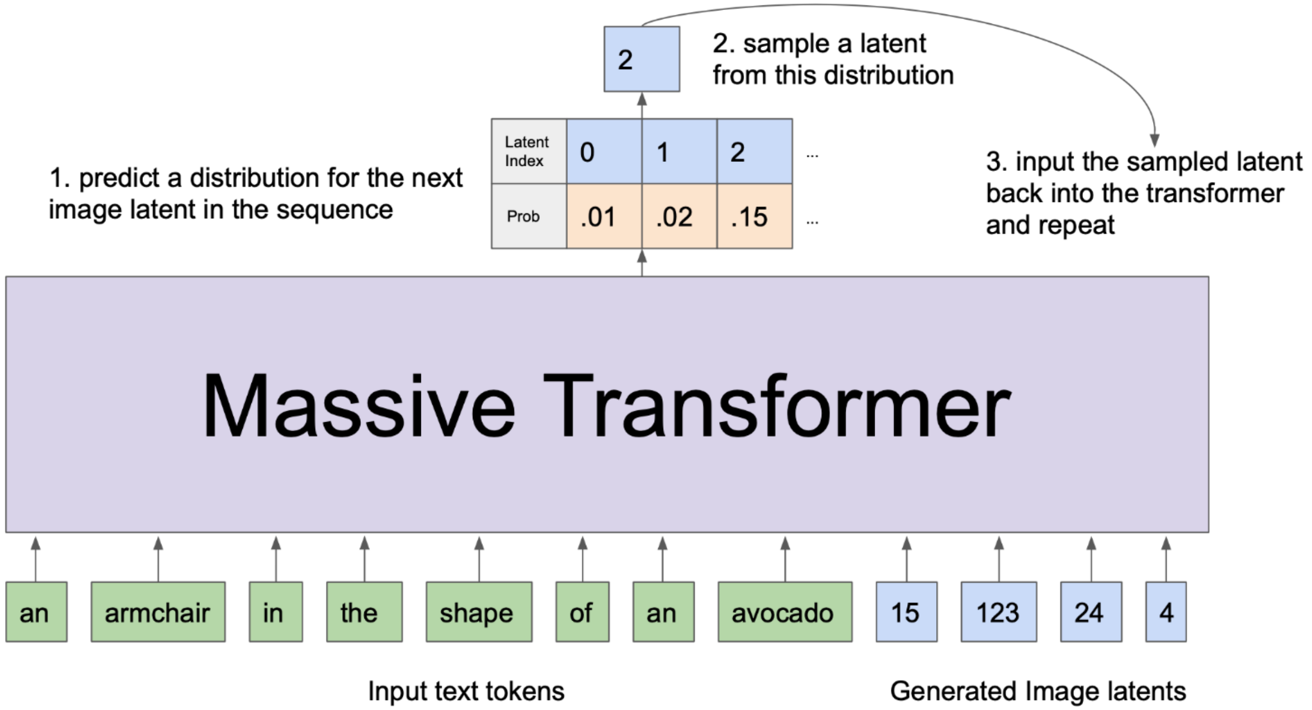
32 x 32 grid of digits, [0… 8192]
Each digit is a “visual token”

https://arxiv.org/abs/2102.12092, Figures from Charlie Snell, https://ml.berkeley.edu/blog/posts/vq-vae/

Image Tokens + Transformers

(2) Autoregressive generation of the tokens



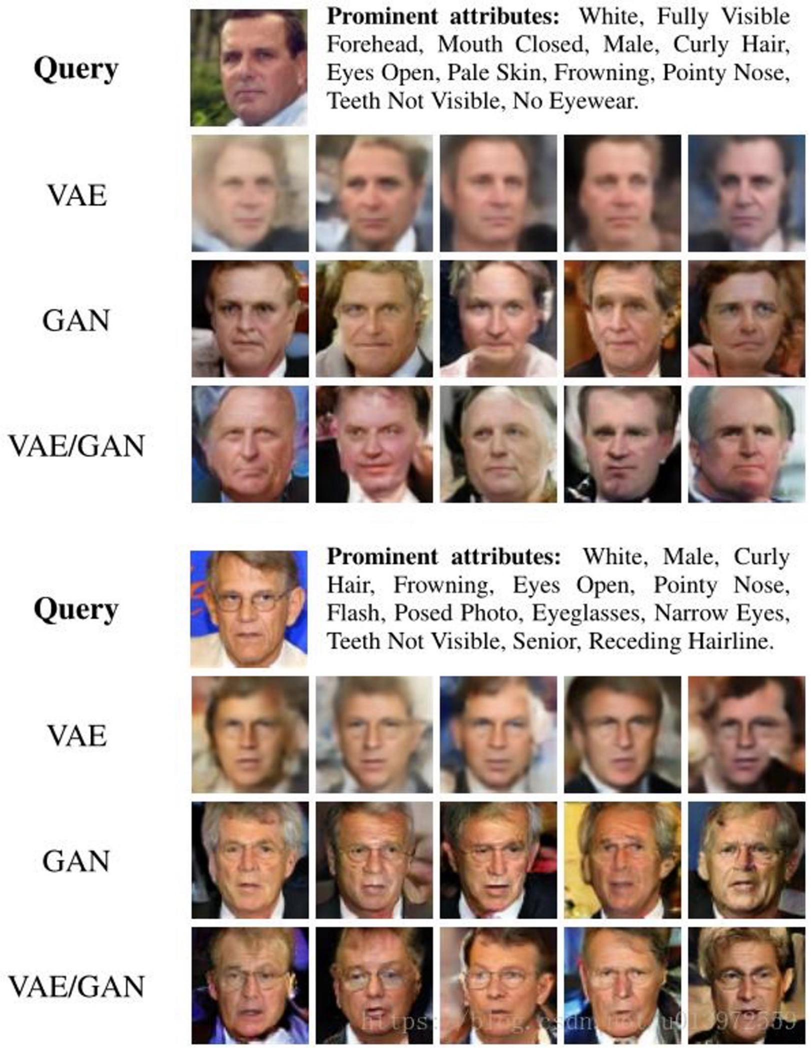


https://arxiv.org/abs/2102.12092, Figures from Charlie Snell,
https://ml.berkeley.edu/blog/posts/vq-vae/

Summary: Variational Autoencoders

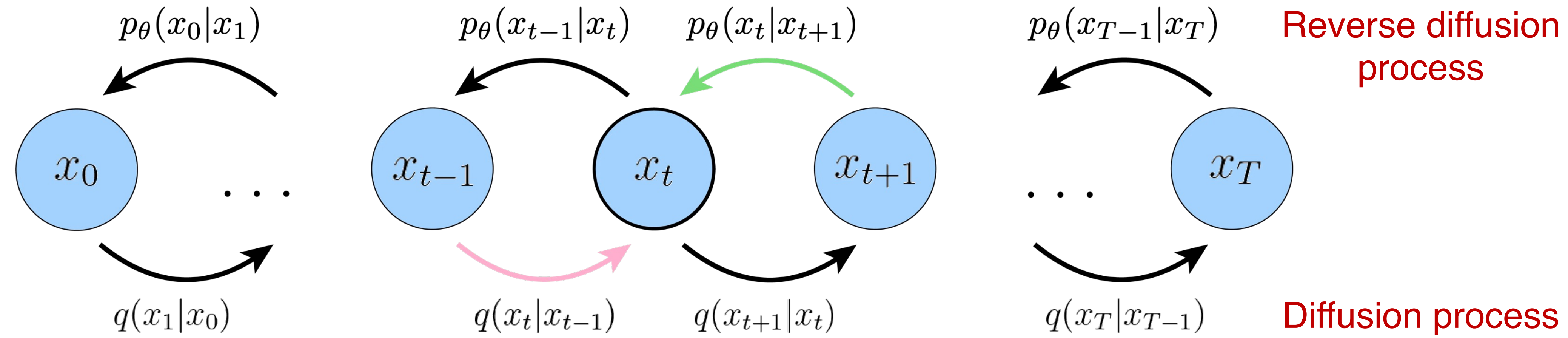
- Relatively easy to train.
- Explicit inference network q(zlx).

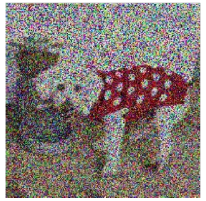
- More blurry images (due to reconstruction).

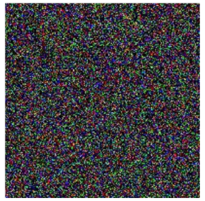


Diffusion Models

Generative modeling via denoising







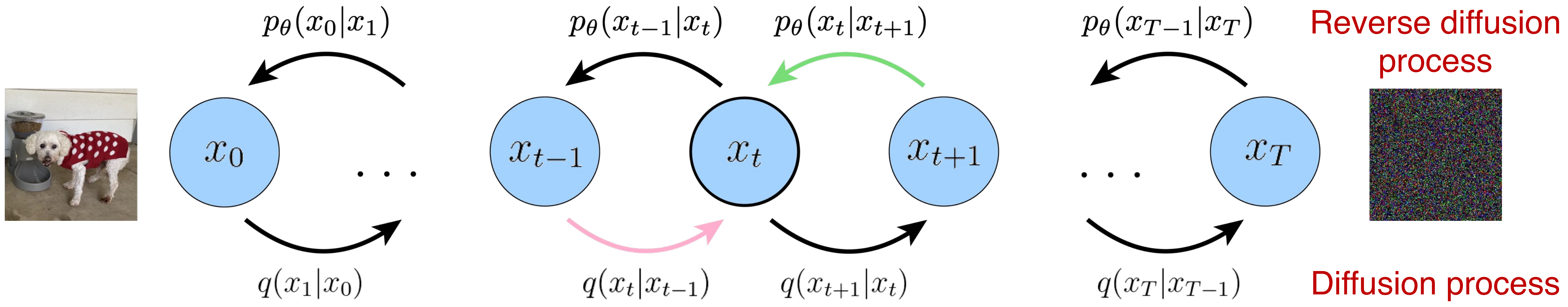
Encoding via adding noise: $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$ Noise parameters

Decoding via denoising: $p(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) \quad \text{where } p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$

[Tutorial by Calvin Luo and Yang Song]

Diffusion Models

Generative modeling via denoising



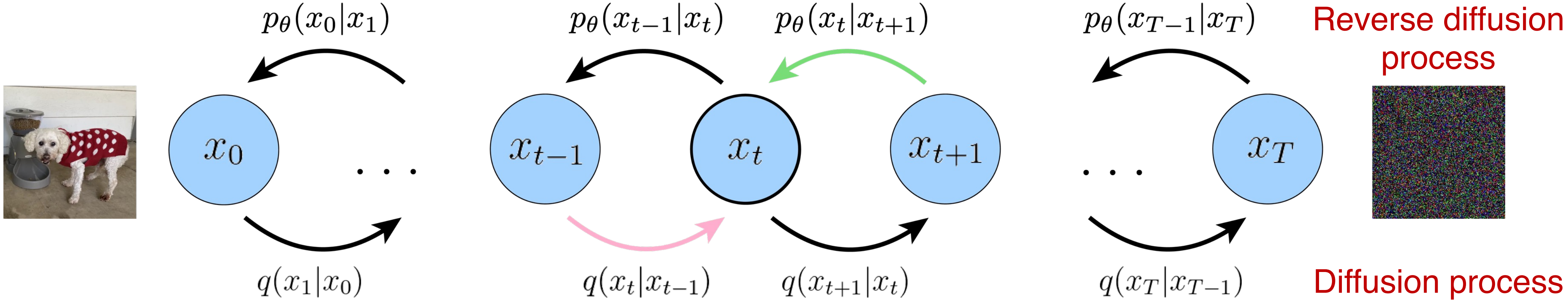
Similar to variational autoencoder, but:

1. The latent dimension is exactly equal to the data dimension.
2. Encoder q is not learned, but pre-defined as a Gaussian distribution centered around the output of previous timestep.
3. Gaussian parameters of latent encoders vary over time such that distribution of final latent is a standard Gaussian.

[Tutorial by Calvin Luo and Yang Song]

Learning Diffusion Models

Key idea: use variational inference



$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)}\right]$$

Our old friend the ELBO

$$= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{\mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \,||\, p(\boldsymbol{x}_T))}_{\text{prior matching term}}$$

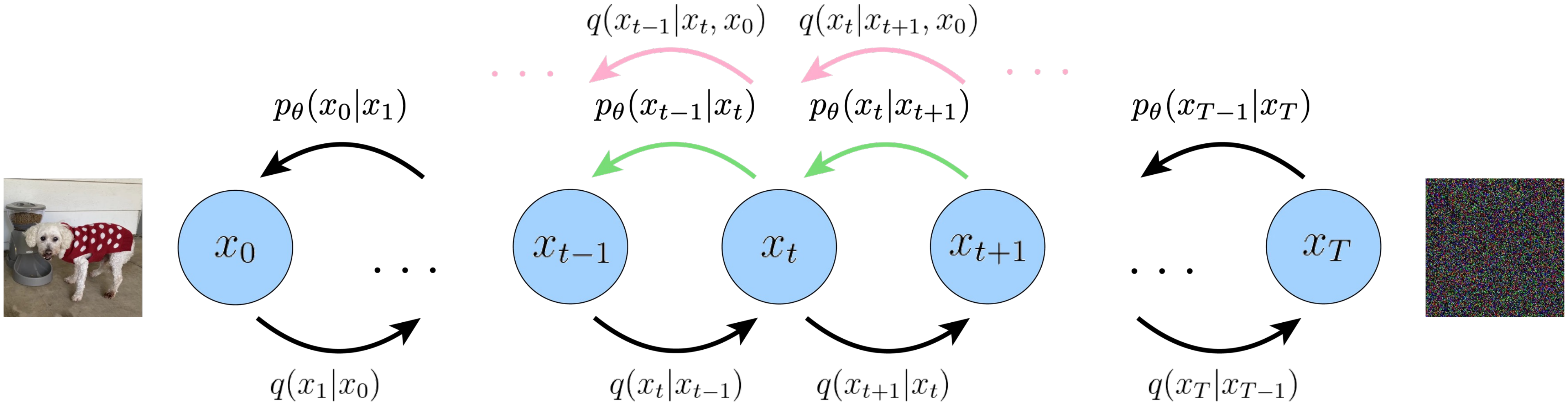
$$- \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Multi-level VAE!

[Tutorial by Calvin Luo and Yang Song]

Learning Diffusion Models

Key idea: use variational inference



Intuition: Neural network to predict cleaner image x_{t-1} from noisy image x_t at time t, consistent with the noise adding process.

Use Bayes rule to reverse, proportional to a Gaussian

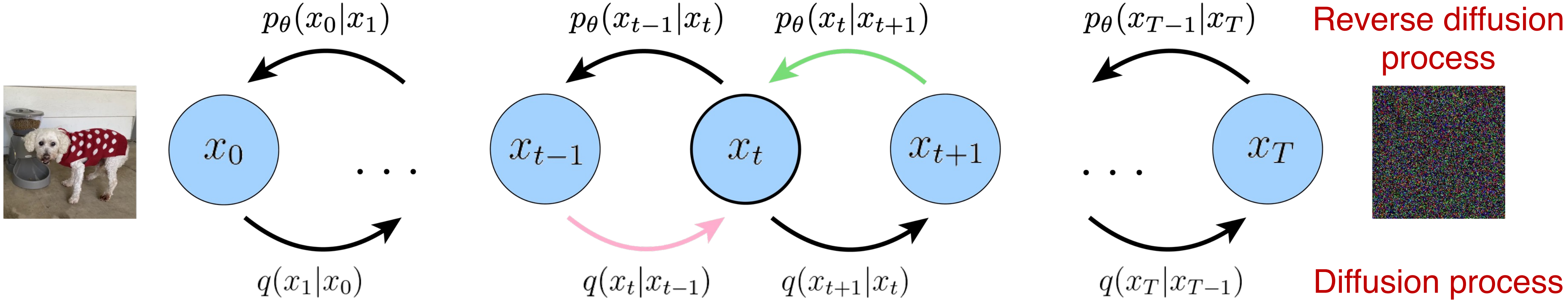
Also parameterize as Gaussian, use reparameteriztion trick

$$-\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

[Tutorial by Calvin Luo and Yang Song]

Learning Noise Parameters

Generative modeling via denoising



Encoding via adding noise: $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$ Noise parameters

Choose $\bar{\alpha}_1 > \cdots > \bar{\alpha}_T$

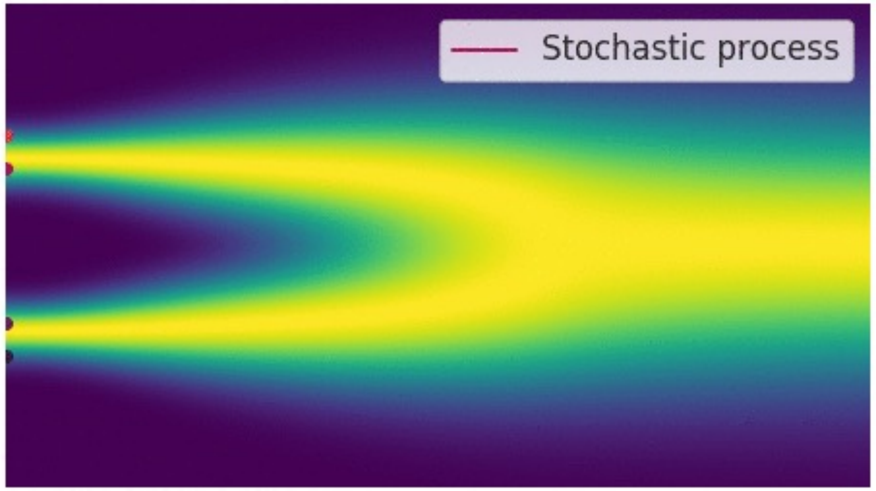
i.e., add smaller noise at the beginning of the diffusion process and gradually increase noise when the samples get noisier.

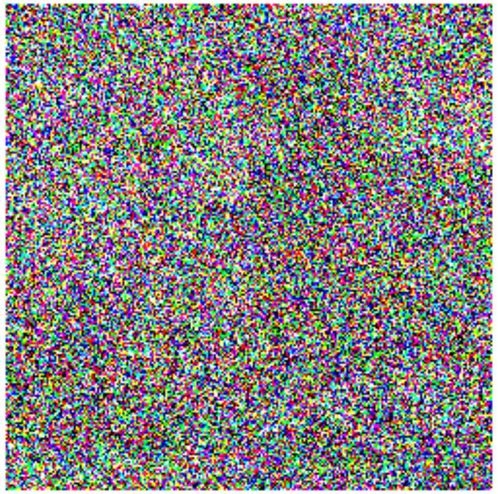
[Tutorial by Calvin Luo and Yang Song]

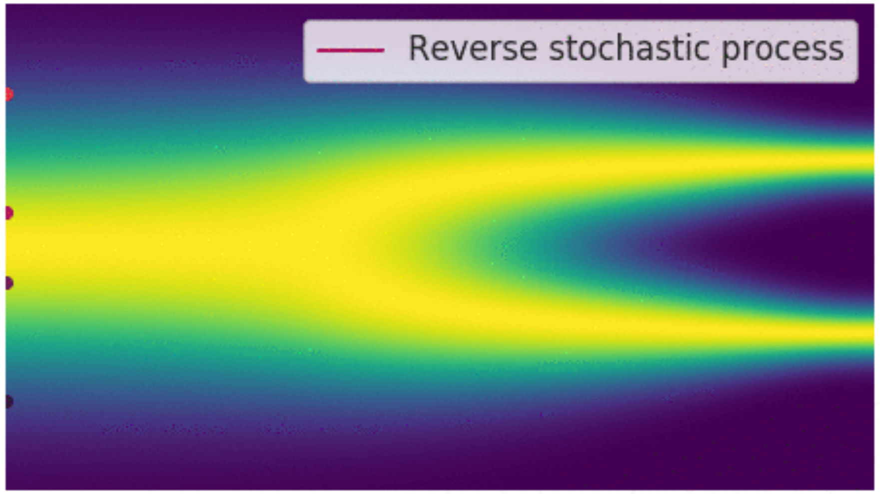
Diffusion Models as Differential Equations

From discrete diffusion process to continuous diffusion process

- Higher quality samples
- Exact log-likelihood
- Controllable generation







[Tutorial by Yang Song]

Diffusion Models as Differential Equations

From discrete diffusion process to continuous diffusion process

Forward SDE (data → noise)

$$\mathbf{x}(0) \longrightarrow \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w} \longrightarrow \mathbf{x}(T)$$

score function

$$\mathbf{x}(0) \longleftarrow \mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}} \longleftarrow \mathbf{x}(T)$$

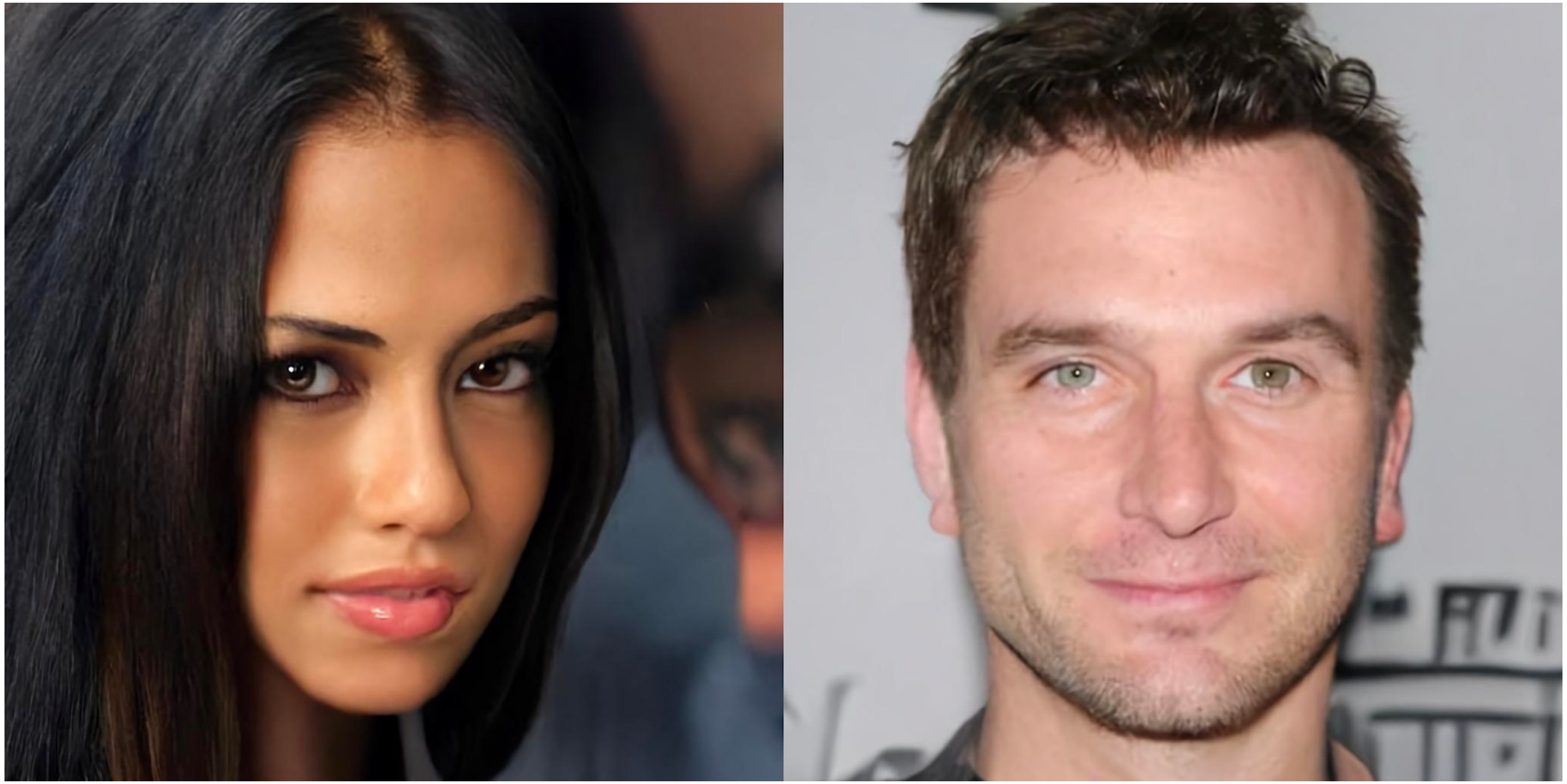
Reverse SDE (noise → data)

Think ‘infinite-layer’ latent variable model

[Tutorial by Calvin Luo and Yang Song]

Diffusion Models as Differential Equations

From discrete diffusion process to continuous diffusion process



[Tutorial by Calvin Luo and Yang Song]

Conditioning Diffusion Models

1. Directly training diffusion models with conditional information

$$p(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) \longrightarrow p(\boldsymbol{x}_{0:T} \mid y) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y)$$

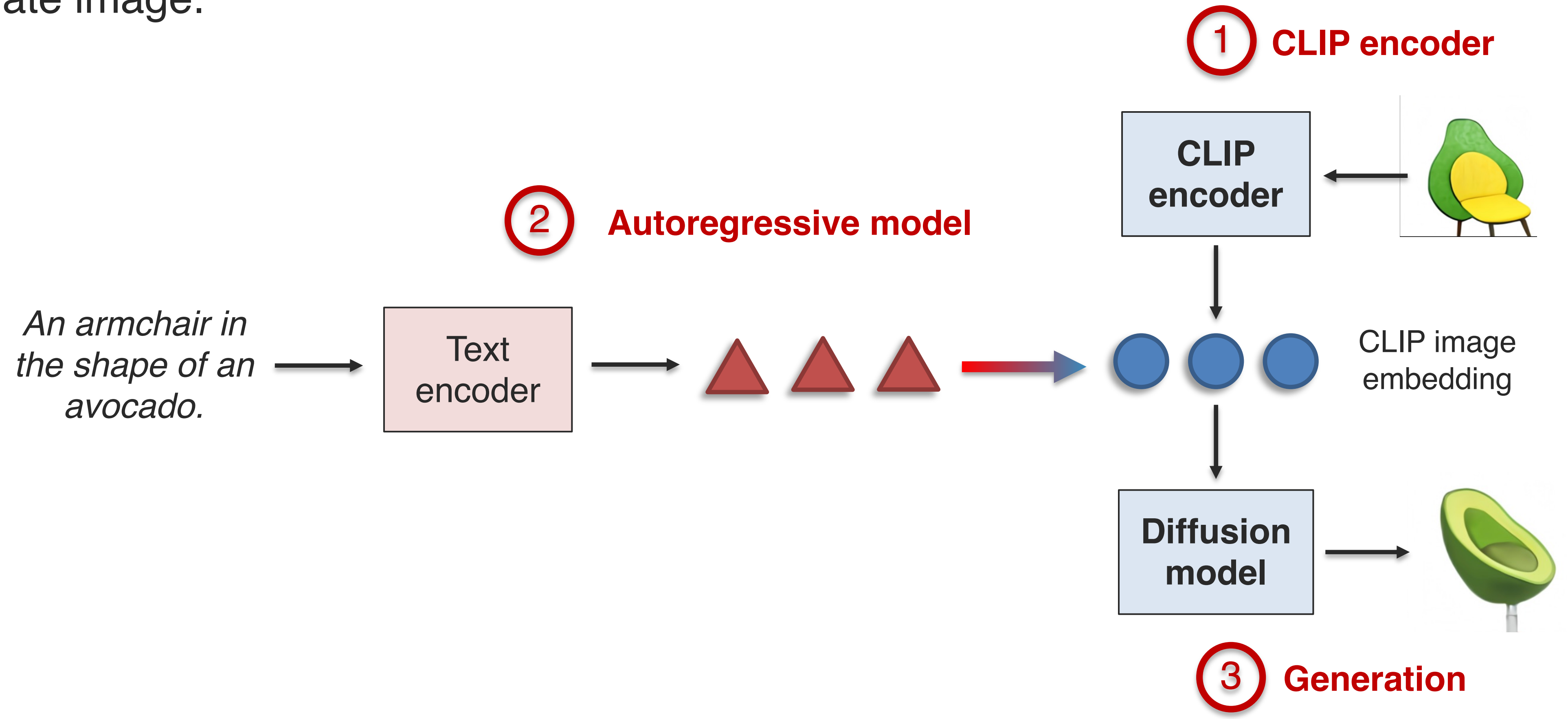
1. Conditional original image prediction $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, y) \approx \boldsymbol{x}_0$
2. Conditional noise prediction $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, y) \approx \boldsymbol{\epsilon}_0$
3. Conditional score function estimation $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, y) \approx \nabla \log p(\boldsymbol{x}_t \mid y)$

[Tutorial by Calvin Luo and Yang Song]

Conditioning Diffusion Models on Text

1. Directly training diffusion models with conditional information

Conditional latent variables are pretrained CLIP embeddings, then diffusion model to generate image.



[Ramesh et al., Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv 2022]

Conditioning Diffusion Models on Text

- DALL-E 2 https://cdn.openai.com/papers/dall-e-2.pdf
- Diffusion on top of frozen CLIP

Imagen https://arxiv.org/pdf/2205.11487.pdf

Diffusion on top of frozen T5 embeddings

A black apple and a green backpack.

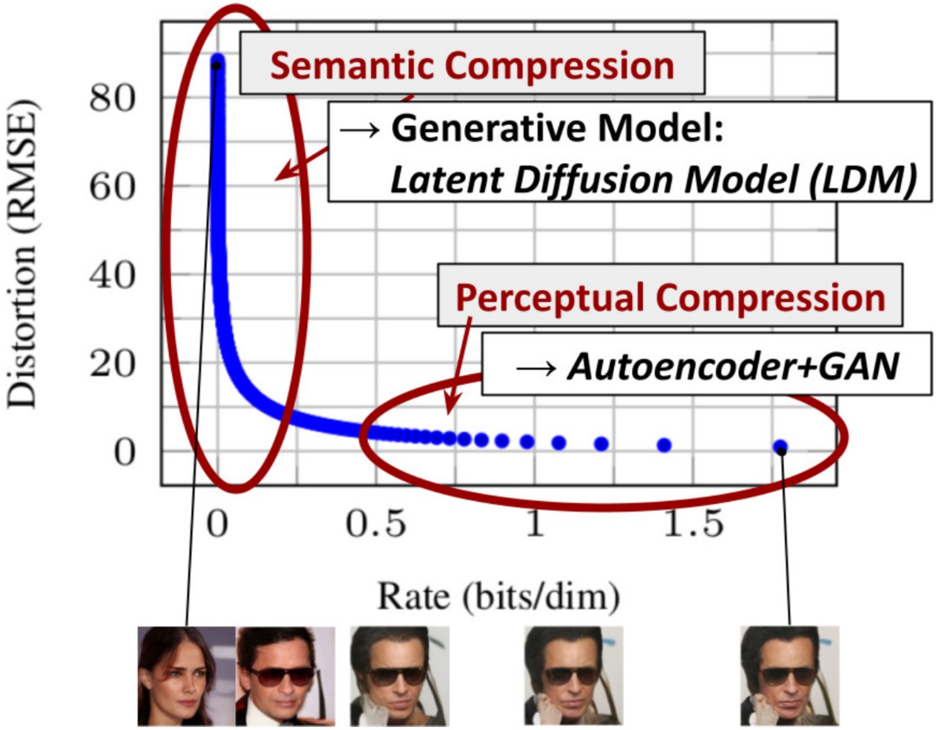
A horse riding an astronaut.

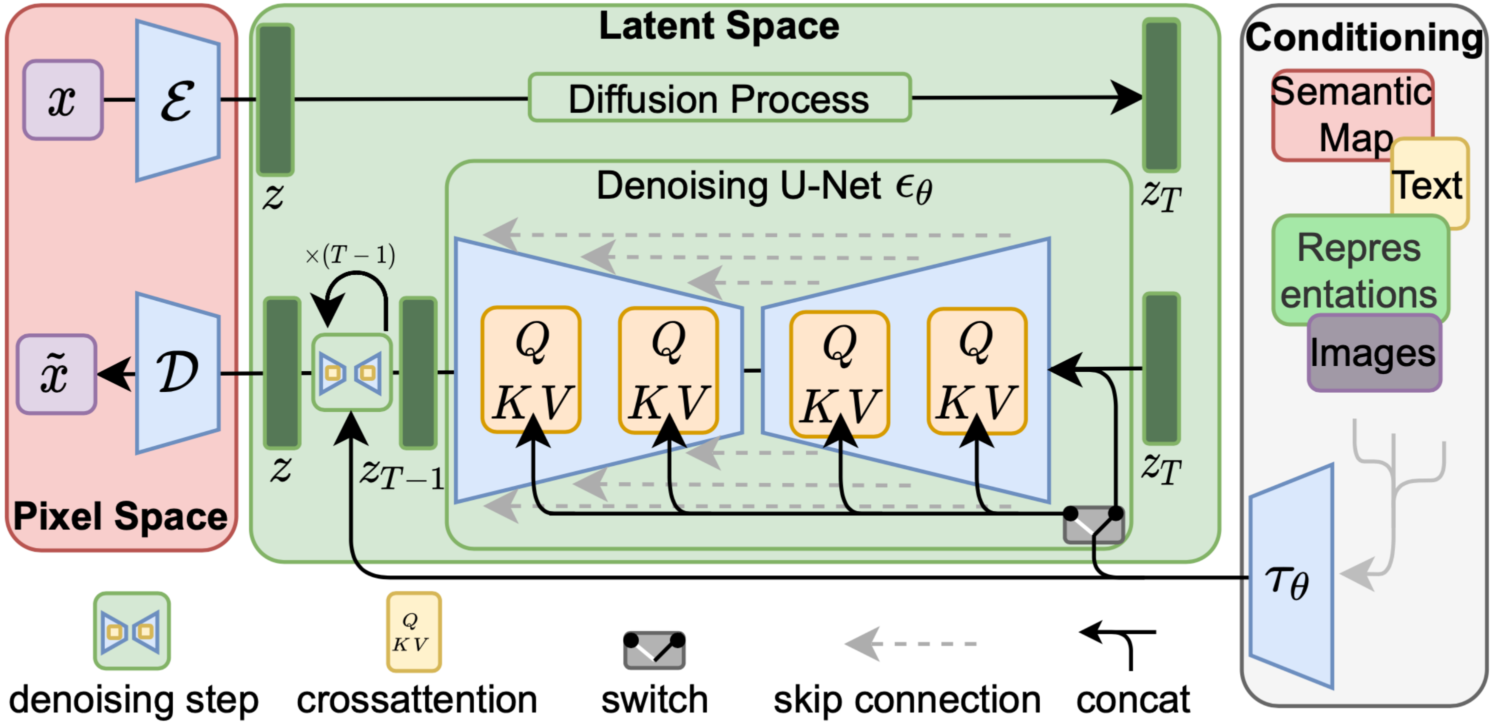
Text-to-Image Generation with Latent Diffusion

1. Directly training diffusion models with conditional information

Diffusion process in latent space instead of pixel space – faster training and inference.

Use autoencoder for perceptual compression, diffusion model for semantic compression.





[Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022]

Text-to-Image Generation with Latent Diffusion

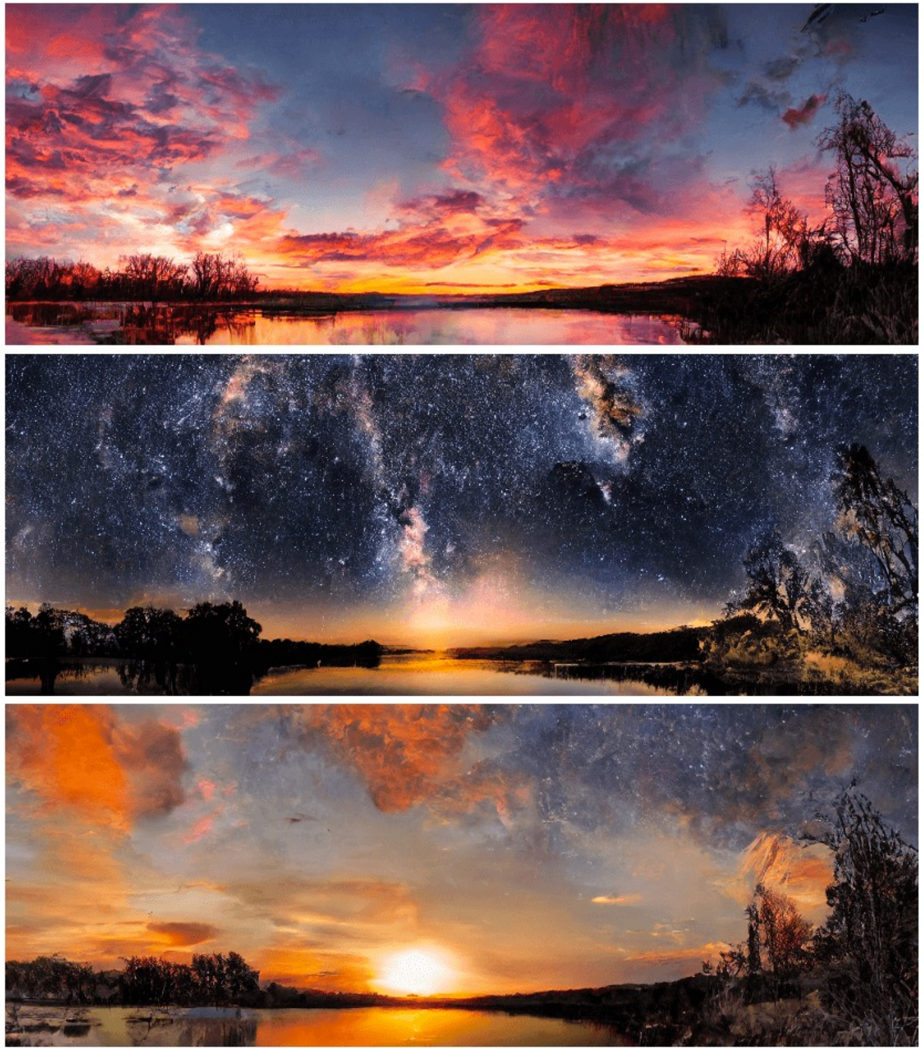
Text-to-Image Synthesis on LAION. 1.45B Model.

'A street sign that reads "Latent Diffusion" '	'A zombie in the style of Picasso'	'An image of an animal half mouse half octopus'	'An illustration of a slightly conscious neural network'	'A painting of a squirrel eating a burger'	'A watercolor painting of a chair that looks like an octopus'	'A shirt with the inscription: "I love generative models!" '

[Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022]

Text-to-Image Generation with Latent Diffusion

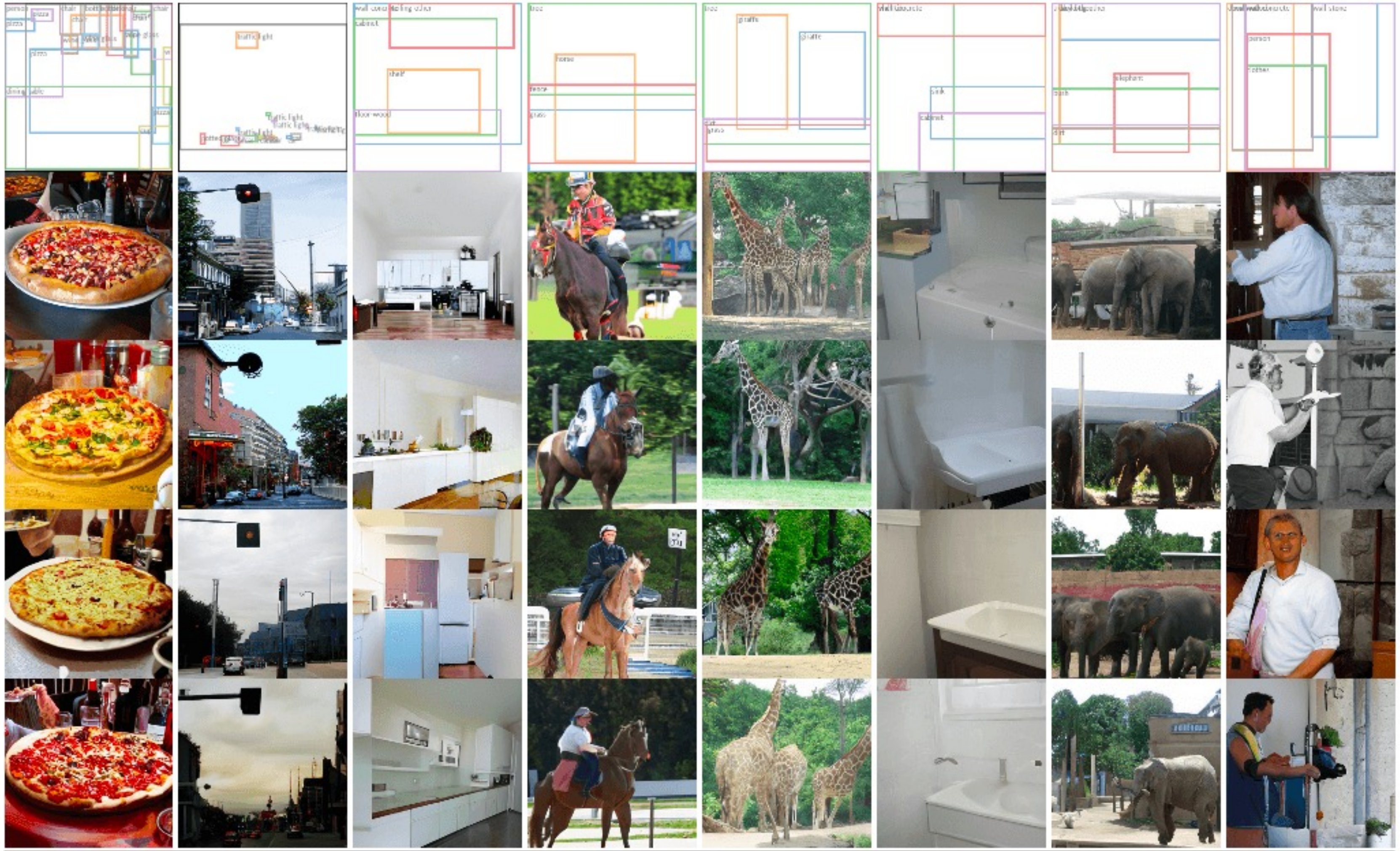
Semantic Synthesis on Flickr-Landscapes [21]



[Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022]

Text-to-Image Generation with Latent Diffusion

layout-to-image synthesis on the COCO dataset

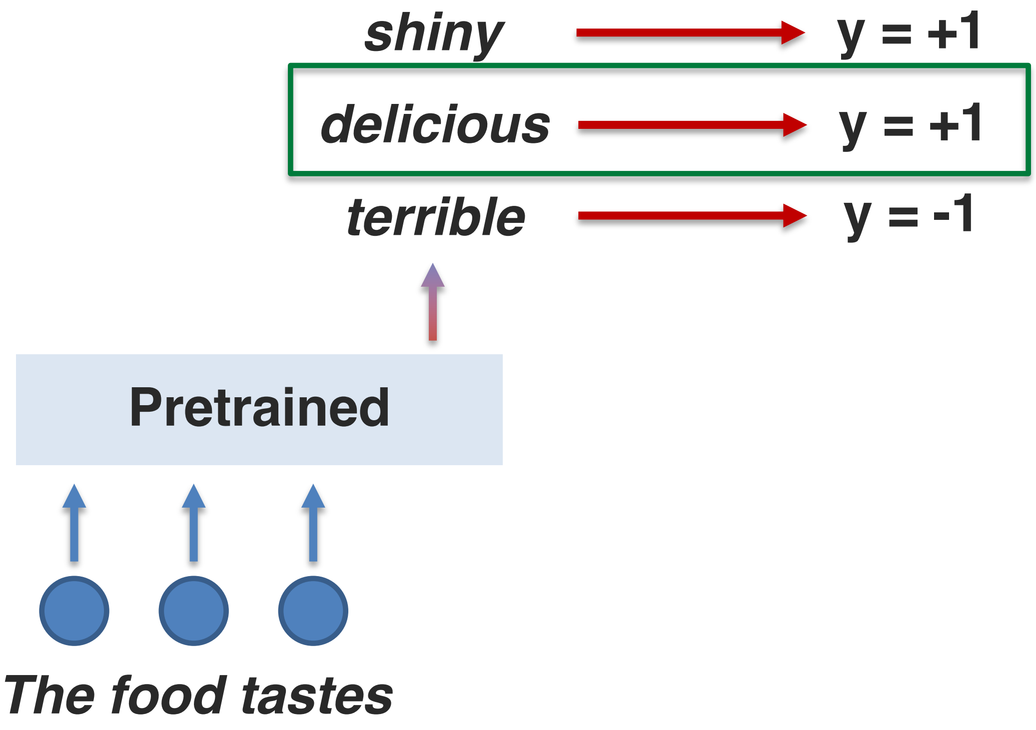


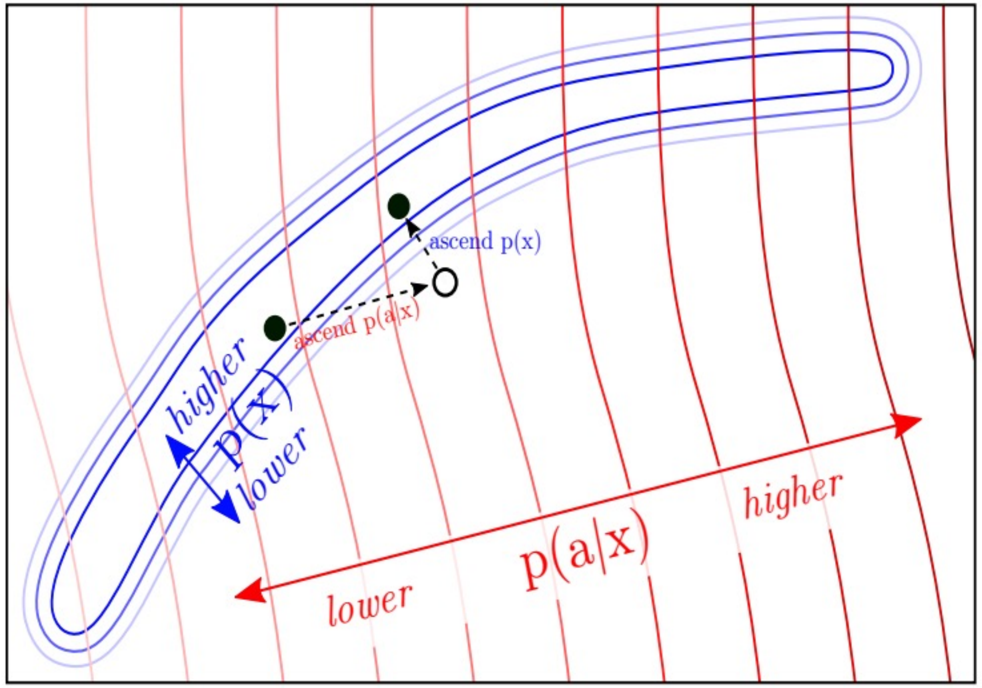
[Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022]

Conditioning Diffusion Models

2. Training unconditional diffusion model then classifier guidance

$$\nabla \log p(\boldsymbol{x}_t \mid y) = \underbrace{\nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y \mid \boldsymbol{x}_t)}_{\text{classifier gradient}}$$





[Dhariwal and Nichol, Diffusion Models Beat GANs on Image Synthesis. arXiv 2021]

Conditioning Diffusion Models

3. Training unconditional diffusion model then classifier-free guidance

$$\begin{aligned}
\nabla \log p(\boldsymbol{x}_t \mid y) &= \nabla \log p(\boldsymbol{x}_t) + \gamma \left(\nabla \log p(\boldsymbol{x}_t \mid y) - \nabla \log p(\boldsymbol{x}_t)\right) \\
&= \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(\boldsymbol{x}_t \mid y) - \gamma \nabla \log p(\boldsymbol{x}_t) \\
&= \underbrace{\gamma \nabla \log p(\boldsymbol{x}_t \mid y)}_{\text{conditional score}} + \underbrace{(1-\gamma) \nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}}
\end{aligned}$$

2 separate diffusion models, one conditional and one unconditional?

Just 1 diffusion model, unconditional training can be seen as setting y=constant

See empirical comparison by GLIDE paper – classifier-free guidance is more preferred

[Nichol et al., GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. arXiv 2022]

Summary: Generative Models

Likelihood-based		
1. Autoregressive models – exact inference via chain rule	Easy to train, exact likelihood	Slow to sample from
2. VAEs – approximate inference via evidence lower bound	Fast & easy to train	Lower generation quality
3. Diffusion model – approximate inference via modeling noise	High generation quality	Slow to sample from

Summary: Conditioning and Controlling Generative Models

1. Disentanglement

$$\mathcal{L}_\beta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta \cdot \mathrm{KL}(q_\phi(z|x)||p(z))$$

2. Conditioning

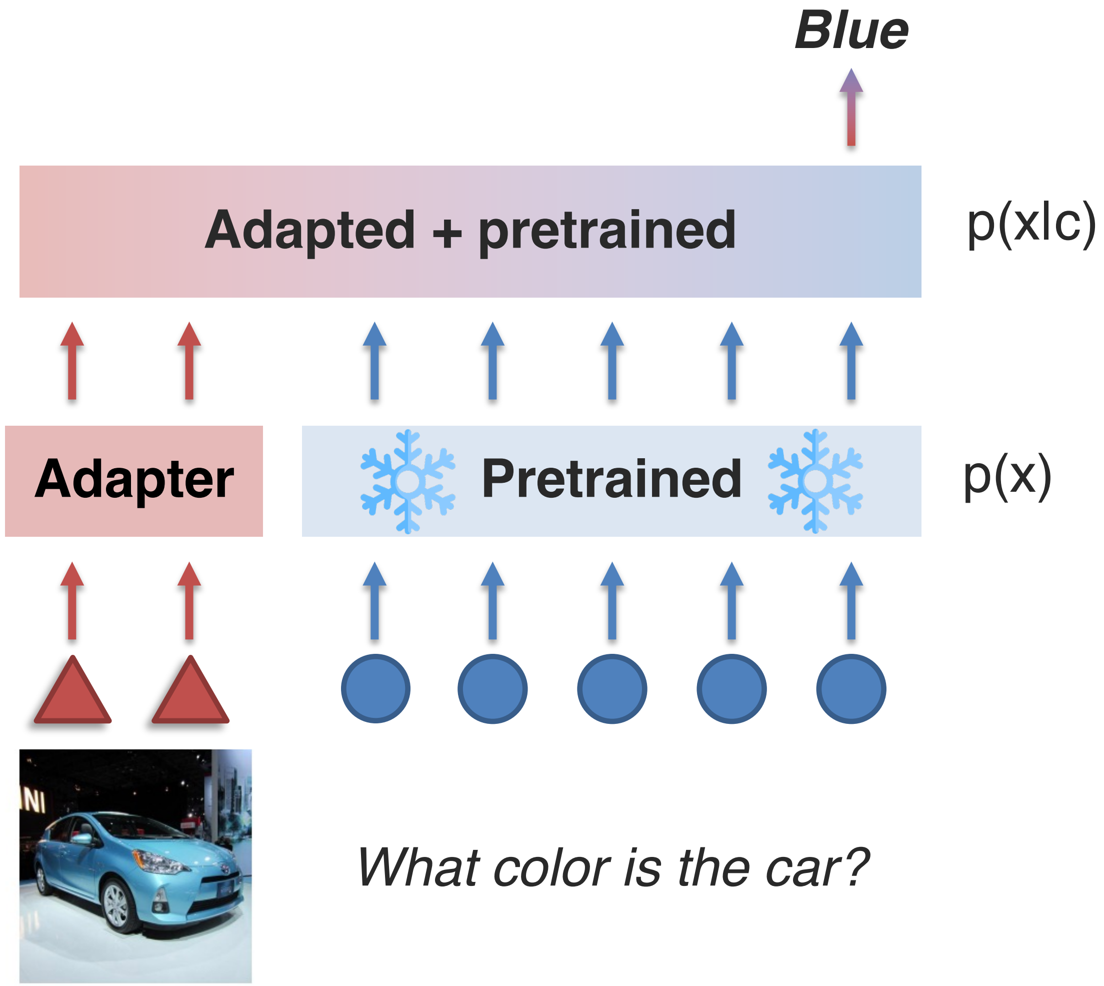
$$p(\boldsymbol{x}_{0:T} \mid y) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y)$$

Summary: Conditioning and Controlling Generative Models

1. Disentanglement

2. Conditioning

3. Prompt tuning



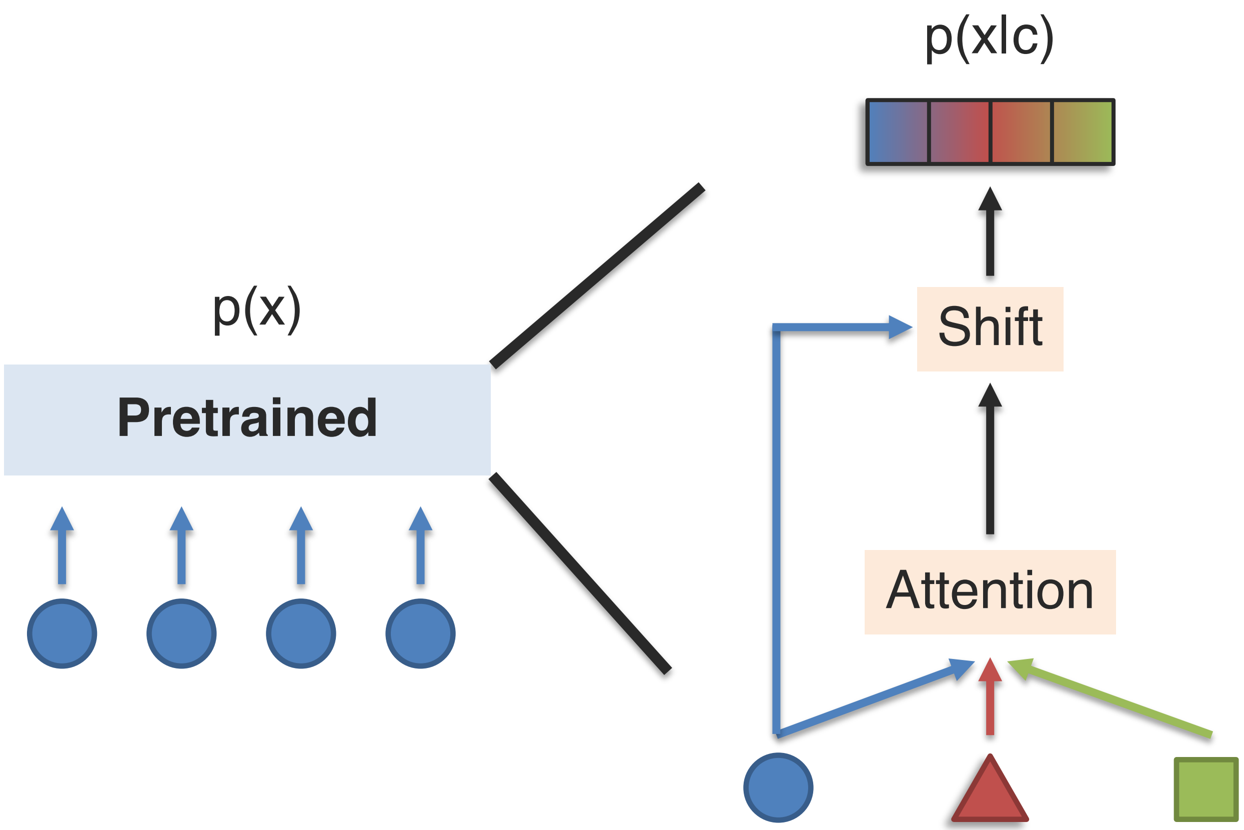
Summary: Conditioning and Controlling Generative Models

1. Disentanglement

2. Conditioning

3. Prompt tuning

4. Representation tuning



Summary: Conditioning and Controlling Generative Models

1. Disentanglement

$$\mathcal{L}_\beta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta \cdot \mathrm{KL}(q_\phi(z|x)||p(z))$$

2. Conditioning

$$p(\boldsymbol{x}_{0:T} \mid y) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y)$$

3. Prompt tuning

4. Representation tuning

5. Classifier gradient tuning

$$\nabla \log p(\boldsymbol{x}_t \mid y) = \underbrace{\nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y \mid \boldsymbol{x}_t)}_{\text{classifier gradient}}$$

6. Classifier-free tuning

$$\nabla \log p(\boldsymbol{x}_t \mid y) = \underbrace{\gamma \nabla \log p(\boldsymbol{x}_t \mid y)}_{\text{conditional score}} + \underbrace{(1-\gamma) \nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}}$$

Open Challenges

Open challenges

Definition: Simultaneously generating multiple modalities to increase information content while maintaining coherence within and across modalities.

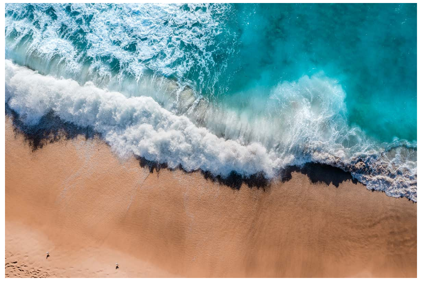
Many goals!

z_y

z_{a1}

z_{a2}

z_y



Cross-modal interactions

Big dog on the beach. Waves crashing, people playing volleyball, …

Cross-modal interactions

'woof' *'crash'* *'bounce'* *'whoosh'*

Temporal + causal + logical structure

Recall representation & alignment!

Recall reasoning!

Open Challenges

Open challenges

1. Synchronized generation over multiple modalities.
2. What's special about diffusion models from multimodal perspective?
2. Combining generation with explicit reasoning to enable compositional generation.
3. Better representation fusion and alignment in generation.
4. More control over large-scale generative models, fine-grained + few-shot control.
5. Human-centered evaluation of generative models.

More resources:

https://lilianweng.github.io/tags/generative-model/
https://yang-song.net/blog/2021/score/
https://blog.evjang.com/2018/01/nf1.html & https://blog.evjang.com/2018/01/nf2.html
https://deepgenerativemodels.github.io/syllabus.html
https://www.cs.cmu.edu/~epxing/Class/10708-20/lectures.html
https://cvpr2022-tutorial-diffusion-models.github.io/
https://huggingface.co/blog/annotated-diffusion
https://calvinyluo.com/2022/08/26/diffusion-tutorial.html
https://jmtomczak.github.io/blog/1/1_introduction.html