Topics:

- Variational Autoencoders

CS 4803-DL / 7643-A
ZSOLT KIRA

- **Assignment 3**
  - Due **July 13th 11:59pm EST**
- **Projects**
  - Project proposal due **July 26th**
- Next Meta office hours today 3pm ET on bias/fairness
  - NOT recorded!

W9: July 7	Generative Models (Part I): Generative Adversarial Networks READING: LLaVA-OneVision
W9: July 9	Generative Models (Part II): Diffusion Models PS3/HW3 due July 13th 11:59pm (grace period July 15th)
W10: July 14	Variational Autoencoders (VAEs)
W10: July 16	Open topic! (please suggest). Otherwise default is Object Detection and Segmentation
W11: July 21	Fairness/Bias Discussion, open topics not covered, Wrap-up Final Project Due July 26th 11:59pm (grace period July 28th)

Back to Generative Models

Supervised Learning

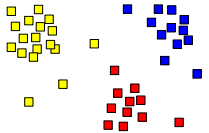
- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification

Sheep
Dog
Cat
Lion
Giraffe

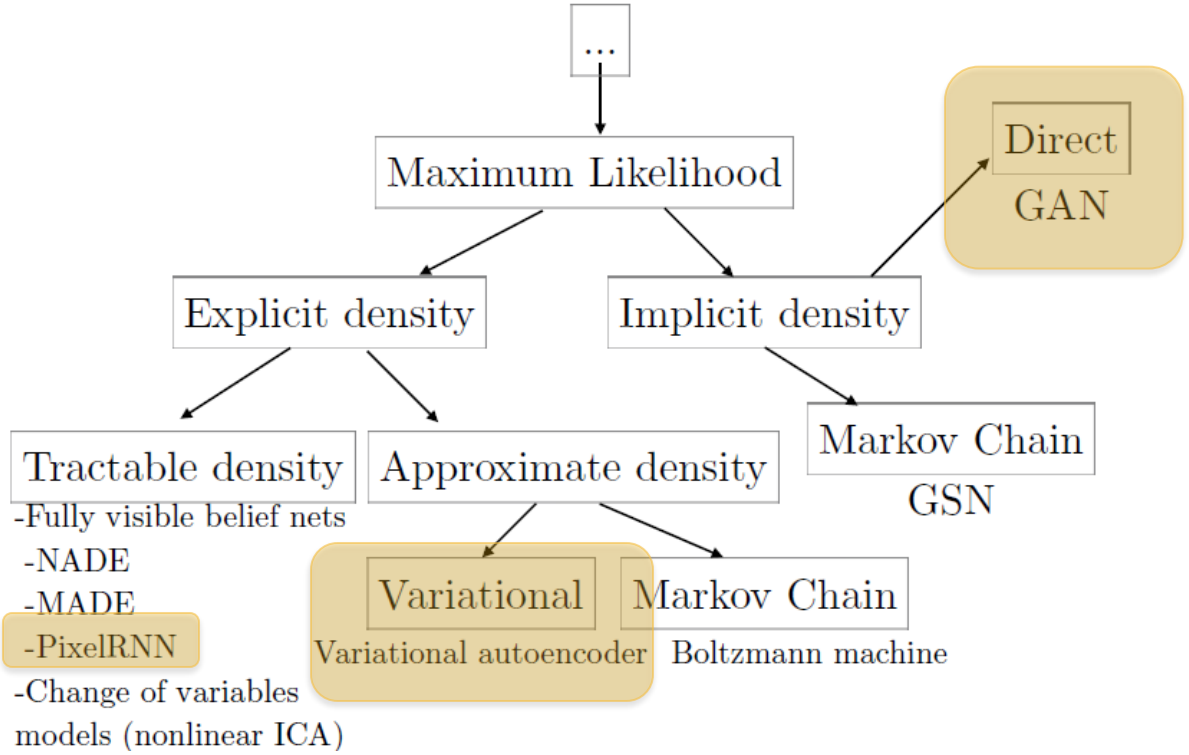
Less Labels

Unsupervised Learning

- Input: $\{X\}$
- Learning output: $P(x)$
- Example: Clustering, density estimation, etc.



Spectrum of Low-Labeled Learning

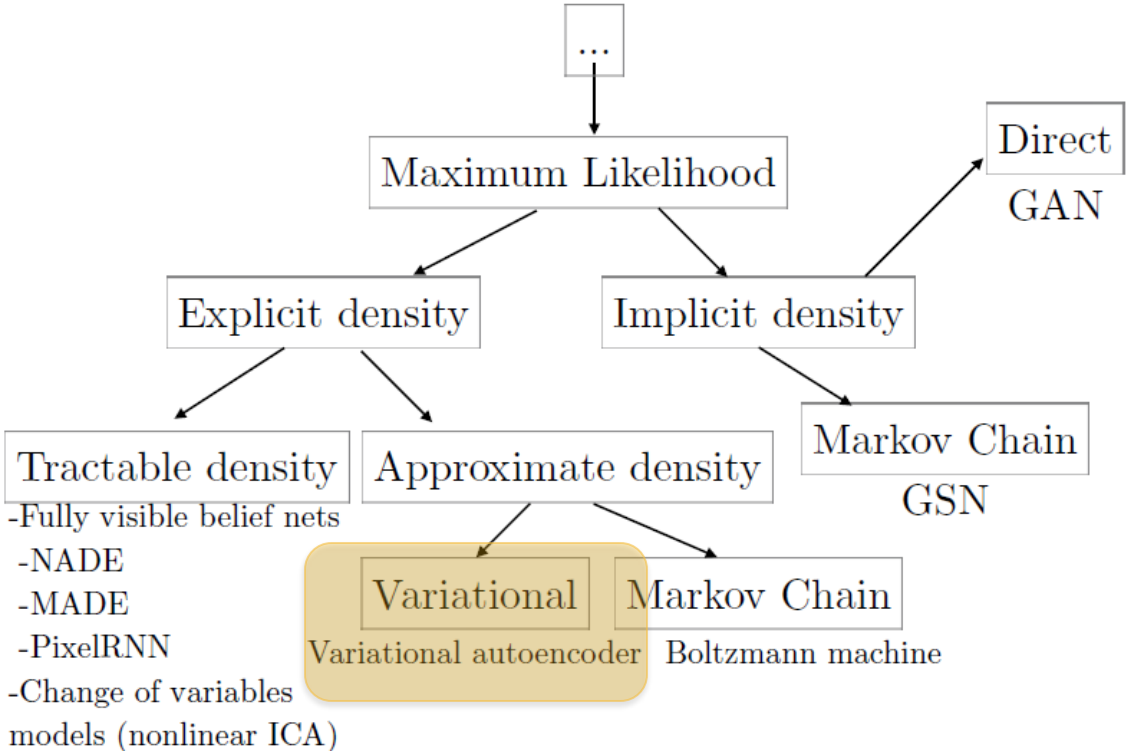


Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

Georgia Tech

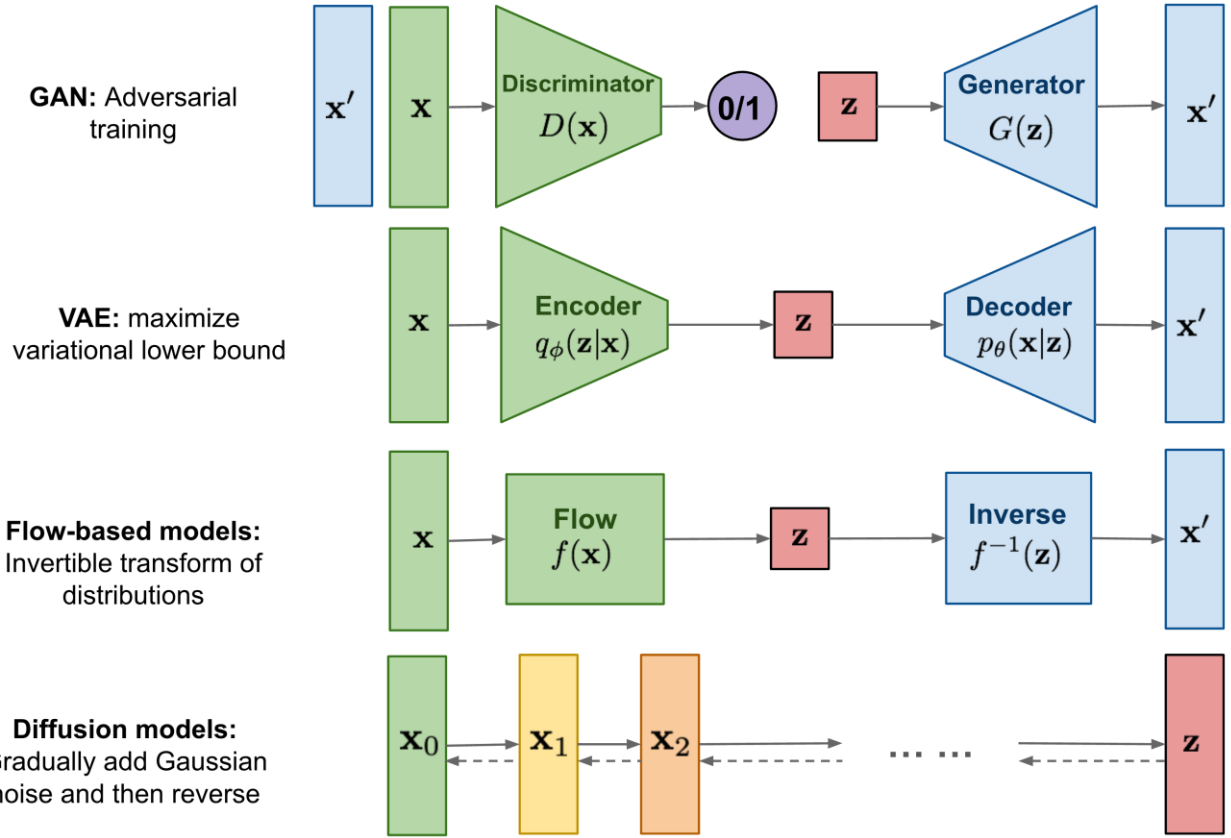
Variational Autoencoders (VAEs)

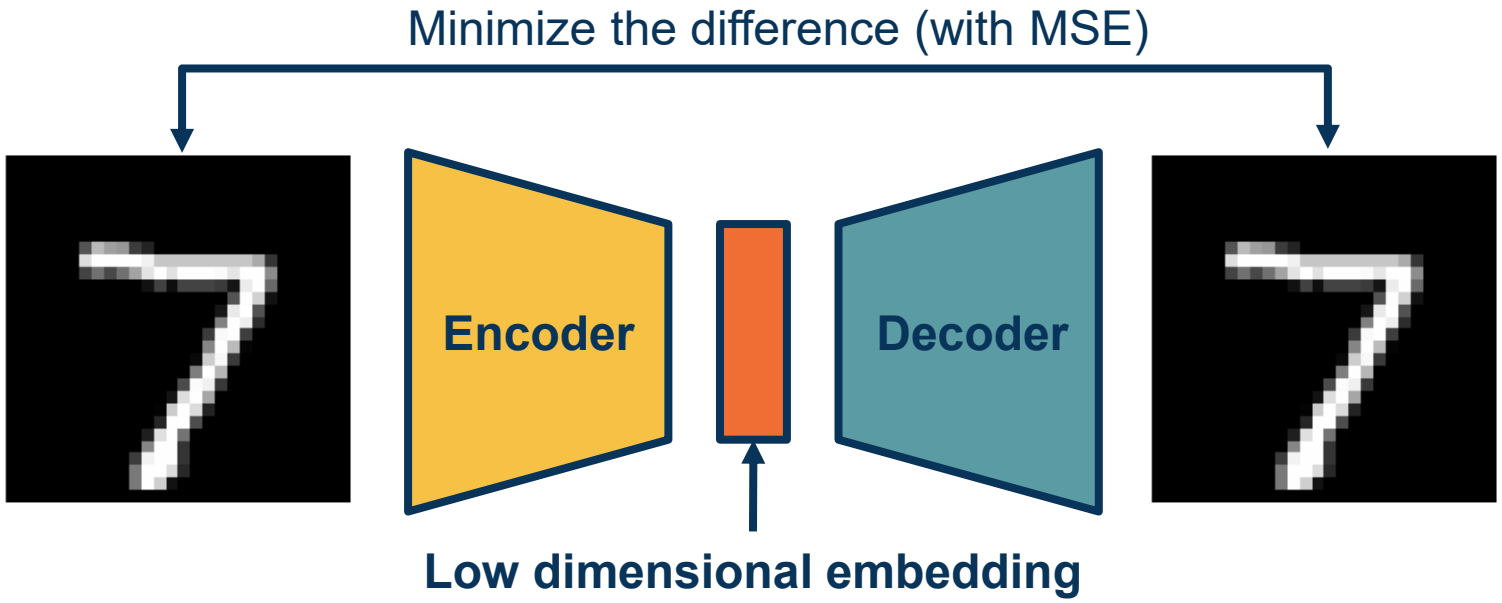


Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

Comparison

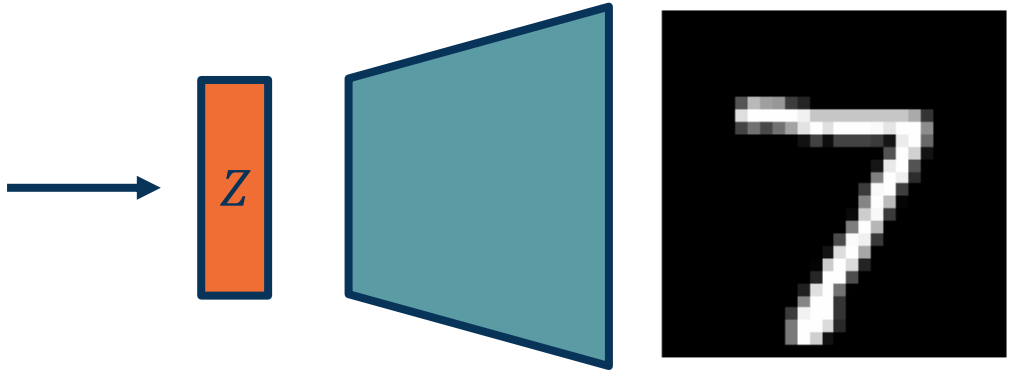

GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$



Linear layers with reduced dimension or Conv-2d layers with stride

Linear layers with increasing dimension or Conv-2d layers with bilinear upsampling

What is this?
Hidden/Latent variables
Factors of variation that produce an image:
(digit, orientation, scale, etc.)



$$P(X) = \int P(X|Z;\theta)P(Z)dZ$$

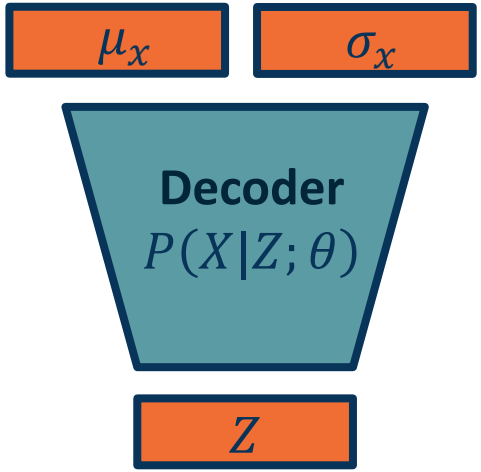
- We cannot maximize this likelihood due to the integral
- Instead we maximize a variational *lower bound* (VLB) that we *can* compute

Kingma & Welling, Auto-Encoding Variational Bayes

Formalizing the Generative Model

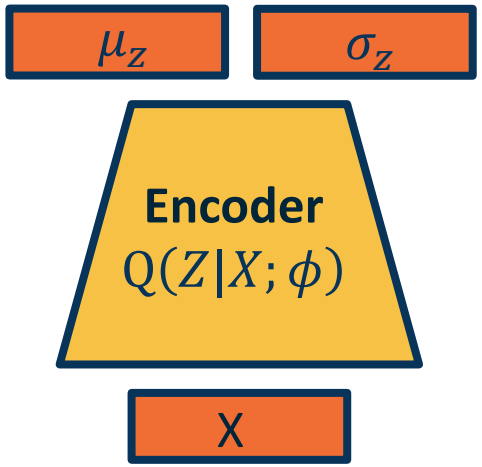
Georgia Tech

- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization
- Just as before, sample Z from simpler distribution
- We can also output parameters of a probability distribution!
  - **Example**: μ, σ of Gaussian distribution
  - For multi-dimensional version output diagonal covariance
- How can we maximize $P(X) = \int P(X|Z;\theta)P(Z)dZ$



Variational Autoencoder: Decoder

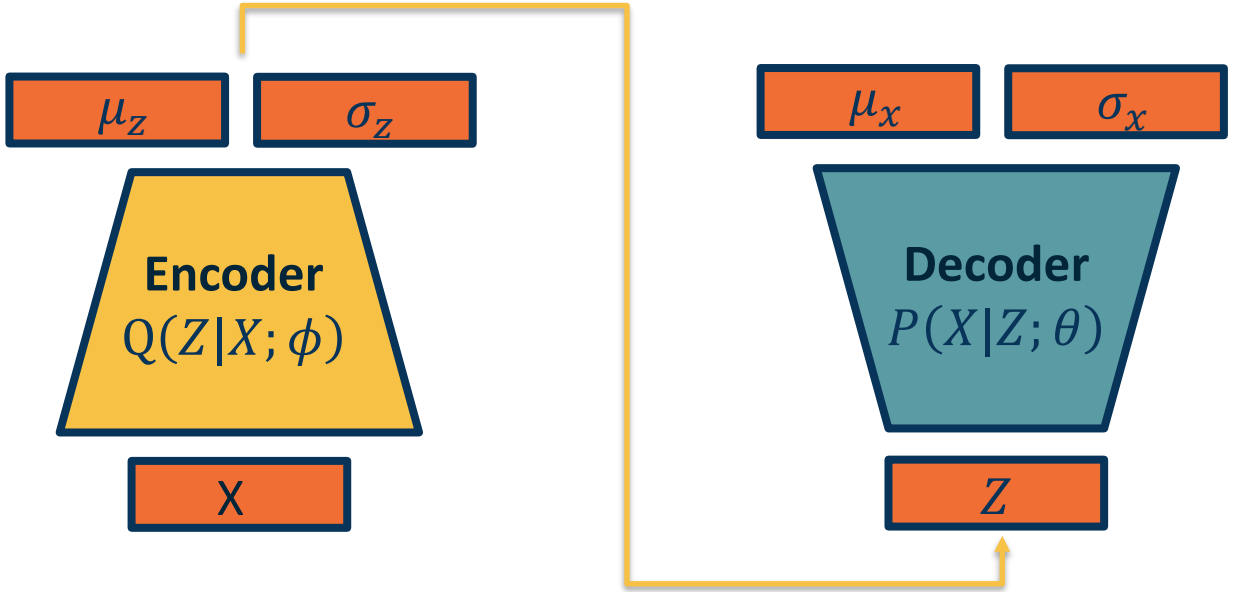
- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization



- Given an image, estimate Z
- Again, output *parameters of a distribution*

Variational Autoencoder: Encoder

- We can tie the encoder and decoder together into a probabilistic autoencoder
  - Given data (X), estimate μ_z, σ_z and sample from $N(\mu_z, \sigma_z)$
  - Given Z, estimate μ_x, σ_x and sample from $N(\mu_x, \sigma_x)$



Putting Them Together

- How can we optimize the parameters of the two networks?

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

Aside: KL Divergence (distance measure for distributions), always >= 0

$$KL(a||b) = H_c(a,b) - H(a) = \sum a\,(x)\log a(x) - \sum a(x)\log b(x)$$

Definition of Expectation

$$\mathbb{E}[f] = \mathbb{E}_{x\sim q}[f(x)] = \sum_{x\in\Omega} q(x)f(x)$$

$$KL(a||b) = E[\log\ a(x)] - E[log\ b(x)] = E[log\frac{a(x)}{b(x)}]$$

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

The expectation wrt. z (using encoder network) let us write nice KL terms

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

Georgia Tech

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{> 0}$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

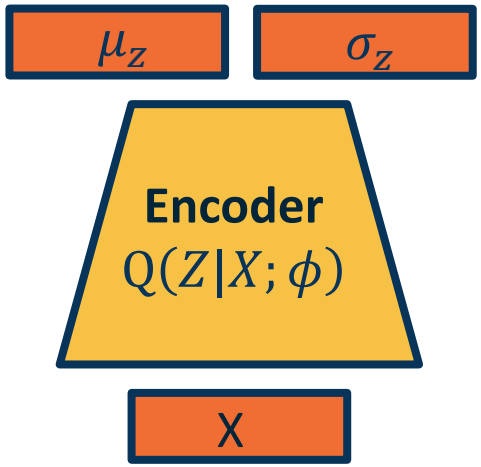
From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

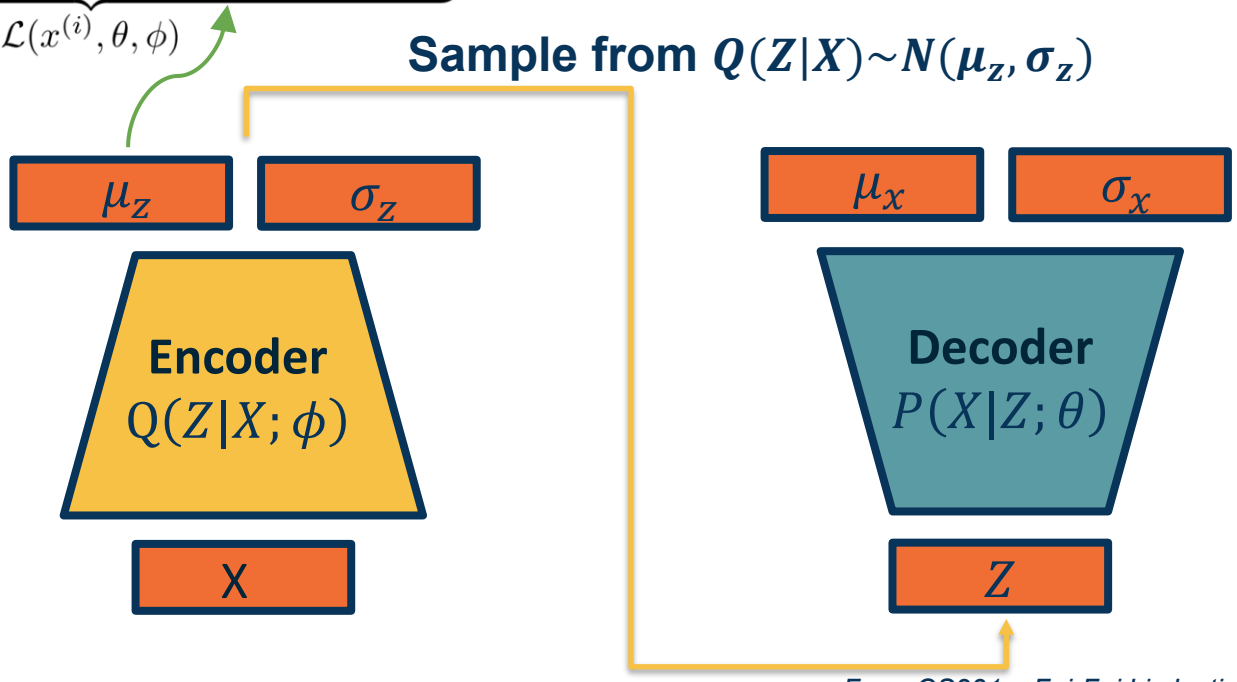
Make approximate posterior distribution close to prior



From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

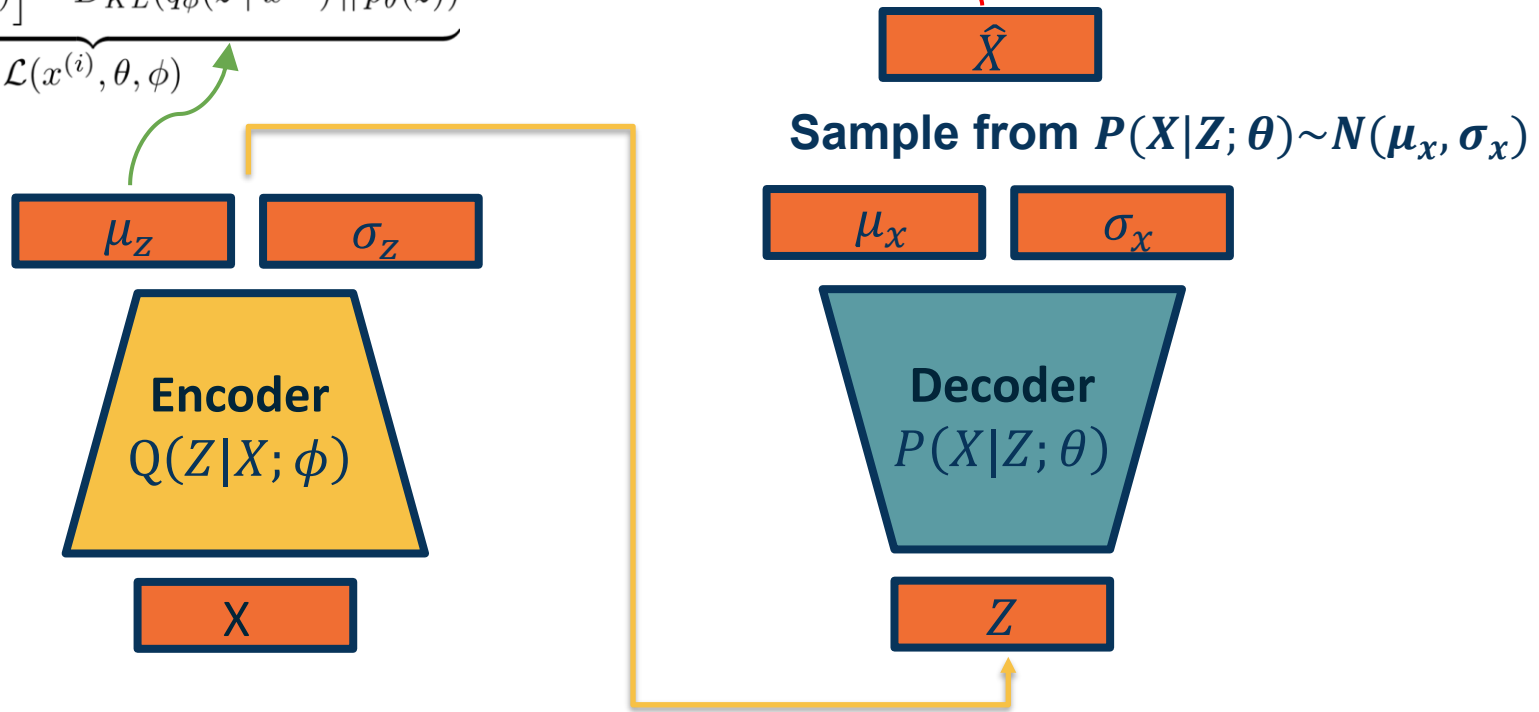


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Maximize likelihood of original input being reconstructed

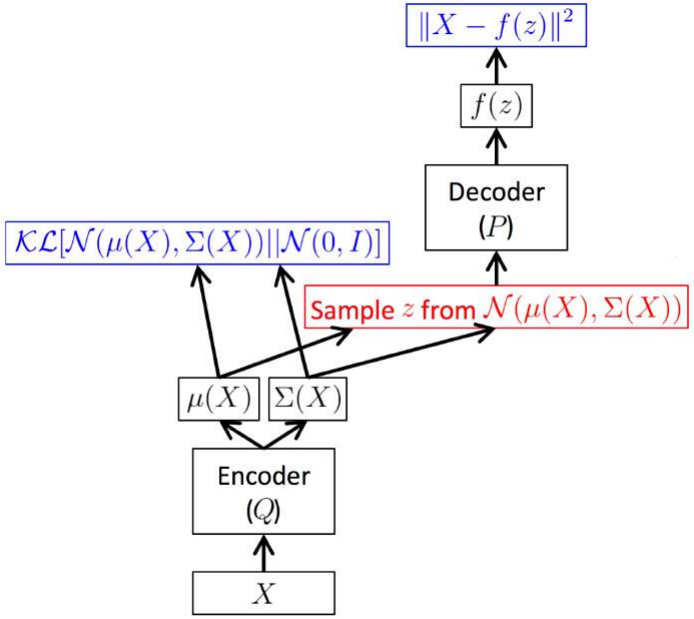


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

- Problem with respect to the VLB: updating ϕ

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{z},\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]$$
$$= -D_{\text{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}$$

- $Z \sim Q(Z|X;\phi)$: need to differentiate through the sampling process w.r.t ϕ (encoder is probabilistic)

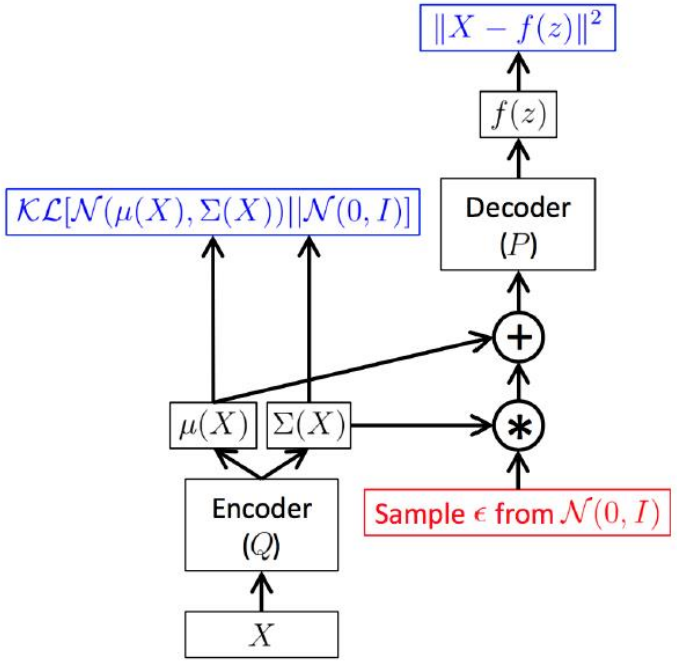


From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

From: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

Problem

- Solution: make the randomness independent of encoder output, making the encoder deterministic
- Gaussian distribution example:
  - Previously: encoder output = random variable $z \sim N(\mu, \sigma)$
  - Now encoder output = distribution parameter $[\mu, \sigma]$
  - $z = \mu + \epsilon * \sigma, \epsilon \sim N(0,1)$



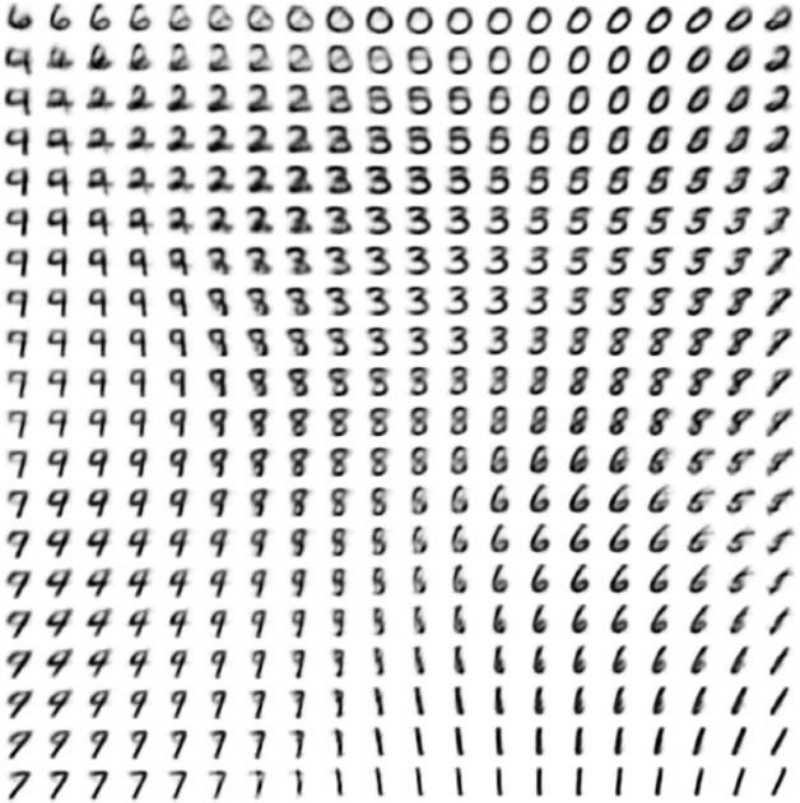
From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

From: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

Reparameterization Trick: Solution

z_1

z_2



Kingma & Welling, Auto-Encoding Variational Bayes

Interpretability of Latent Vector

- Variational Autoencoders (VAEs) provide a principled way to perform approximate maximum likelihood optimization
  - Requires some assumptions (e.g. Gaussian distributions)
- Samples are often not as competitive as diffusion models or GANs
- Latent features (learned in an unsupervised way!) often good for downstream tasks:
  - Example: World models for reinforcement learning (Ha et al., 2018)

Ha & Schmidhuber, World Models, 2018

De-noising Auto-encoder

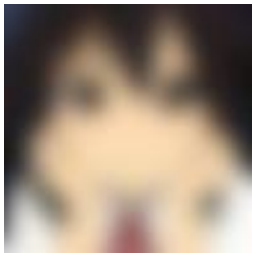
As close as possible

Add noises

NN Encoder

vector

NN Decoder

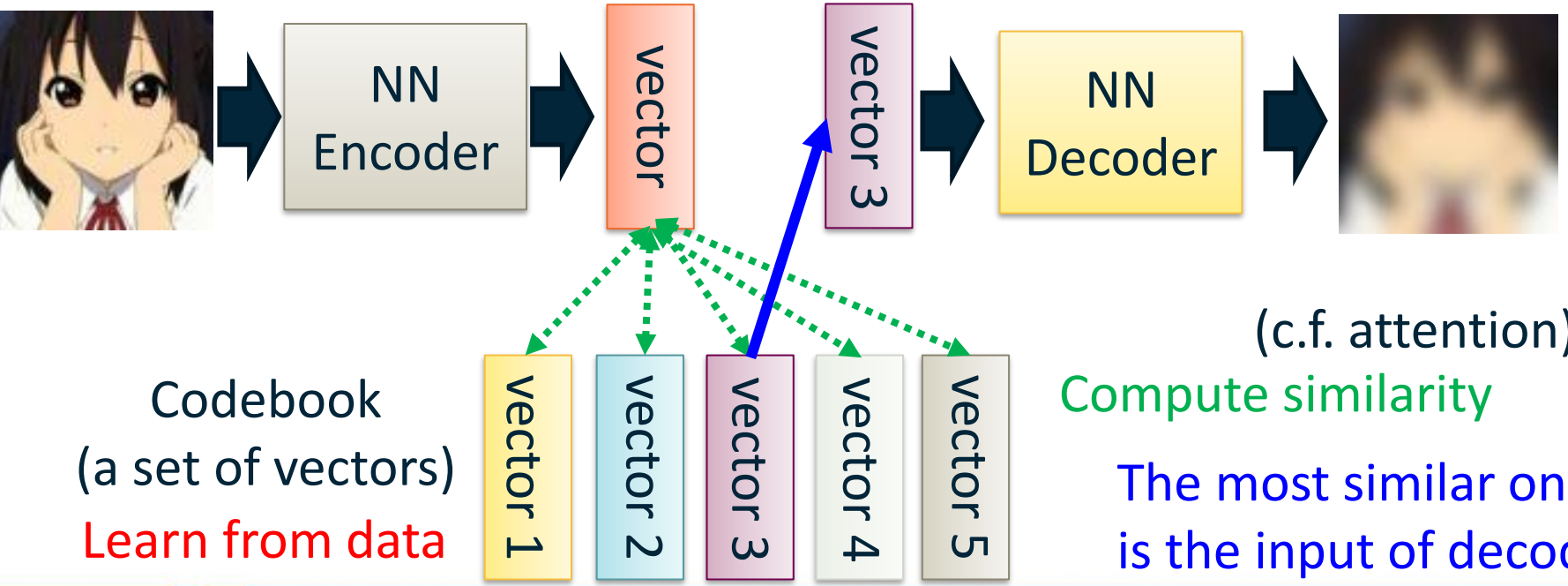


Slide by Hung-yi Lee

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.

Discrete Representation

- Vector Quantized Variational Auto-encoder (VQVAE)



Codebook
(a set of vectors)
Learn from data

(c.f. attention)
Compute similarity

The most similar one
is the input of decoder.

- Variational Autoencoders (VAEs) provide a principled way to perform approximate maximum likelihood optimization
  - Requires some assumptions (e.g. Gaussian distributions)

- Samples are often not as competitive as GANs

- Latent features (learned in an unsupervised way!) often good for downstream tasks:
  - Example: World models for reinforcement learning (Ha et al., 2018)

Ha & Schmidhuber, World Models, 2018

Comparing the different generative models

Q. Which ones are VAEs good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations			
Fast sampling			
High quality samples			

Slide by Krishna & Pratt

Comparing the different generative

VAEs are bad at generating high quality samples

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅		
Fast sampling	✅		
High quality samples	❌		

Slide by Krishna & Pratt

Comparing the different generative models

Q. Which ones are GANs good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅		
Fast sampling	✅		
High quality samples	❌		

Slide by Krishna & Pratt

Comparing the different generative models

GANs suffer from mode collapse

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅	❌	
Fast sampling	✅	✅	
High quality samples	❌	✅	

Slide by Krishna & Pratt

Comparing the different generative models

Q. Which ones are Diffusion models good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅	❌	
Fast sampling	✅	✅	
High quality samples	❌	✅	

Slide by Krishna & Pratt

Comparing the different generative models

Diffusion models are bad at sampling fast.

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅	❌	✅
Fast sampling	✅	✅	❌
High quality samples	❌	✅	✅

Slide by Krishna & Pratt

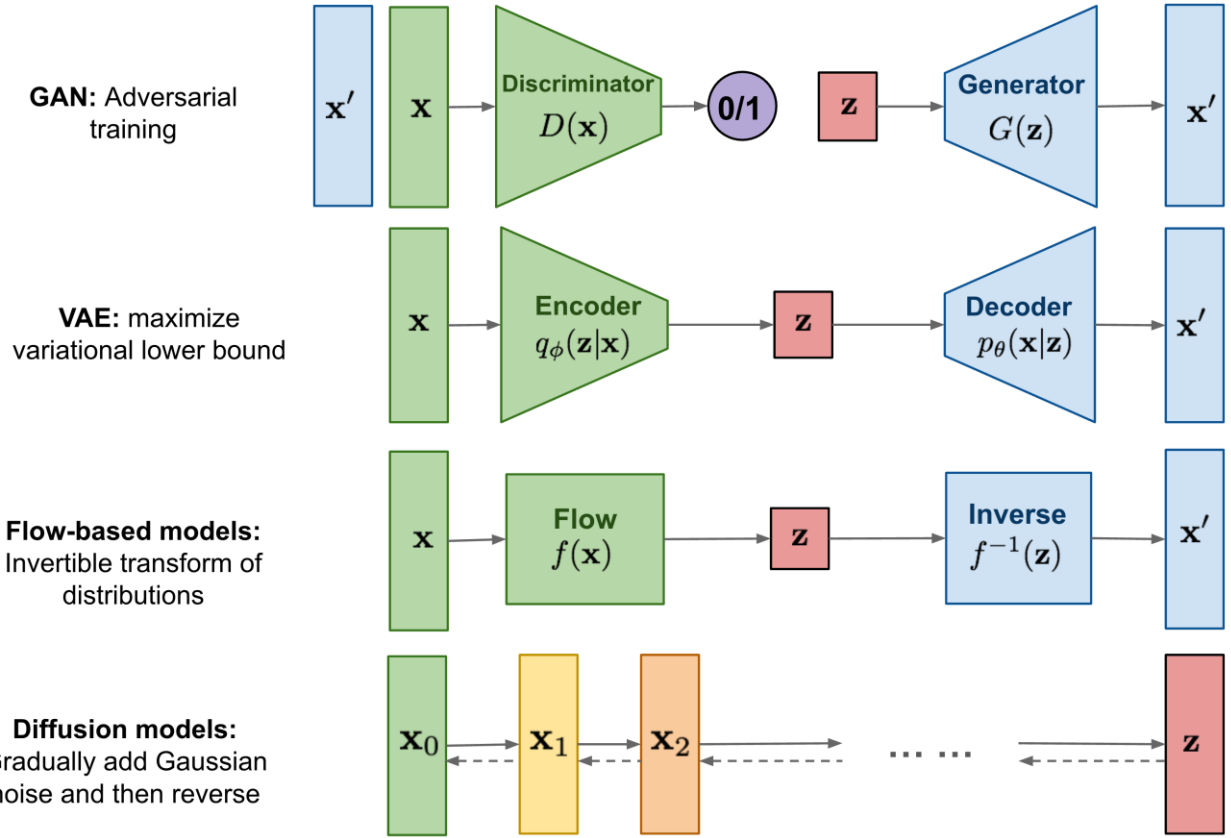
- Several ways to learn *generative* models via deep learning

- **Generative Adversarial Networks (GANs):**
  - Pro: Amazing results across many image modalities
  - Con: Unstable/difficult training process, computationally heavy for good results
  - Con: Limited success for discrete distributions (language)
  - Con: Hard to evaluate (implicit model)
- **Variational Autoencoders:**
  - Pro: Principled mathematical formulation
  - Pro: Results in disentangled latent representations
  - Con: Approximation inference, results in somewhat lower quality reconstructions
- **Diffusion Models**
  - Pro: Great results and diversity!
  - Con: Slow generation (though lots of tricks to address)

Ha & Schmidhuber, World Models, 2018

Overall Summary

Comparison


GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$

Taxonomy of Generative Models

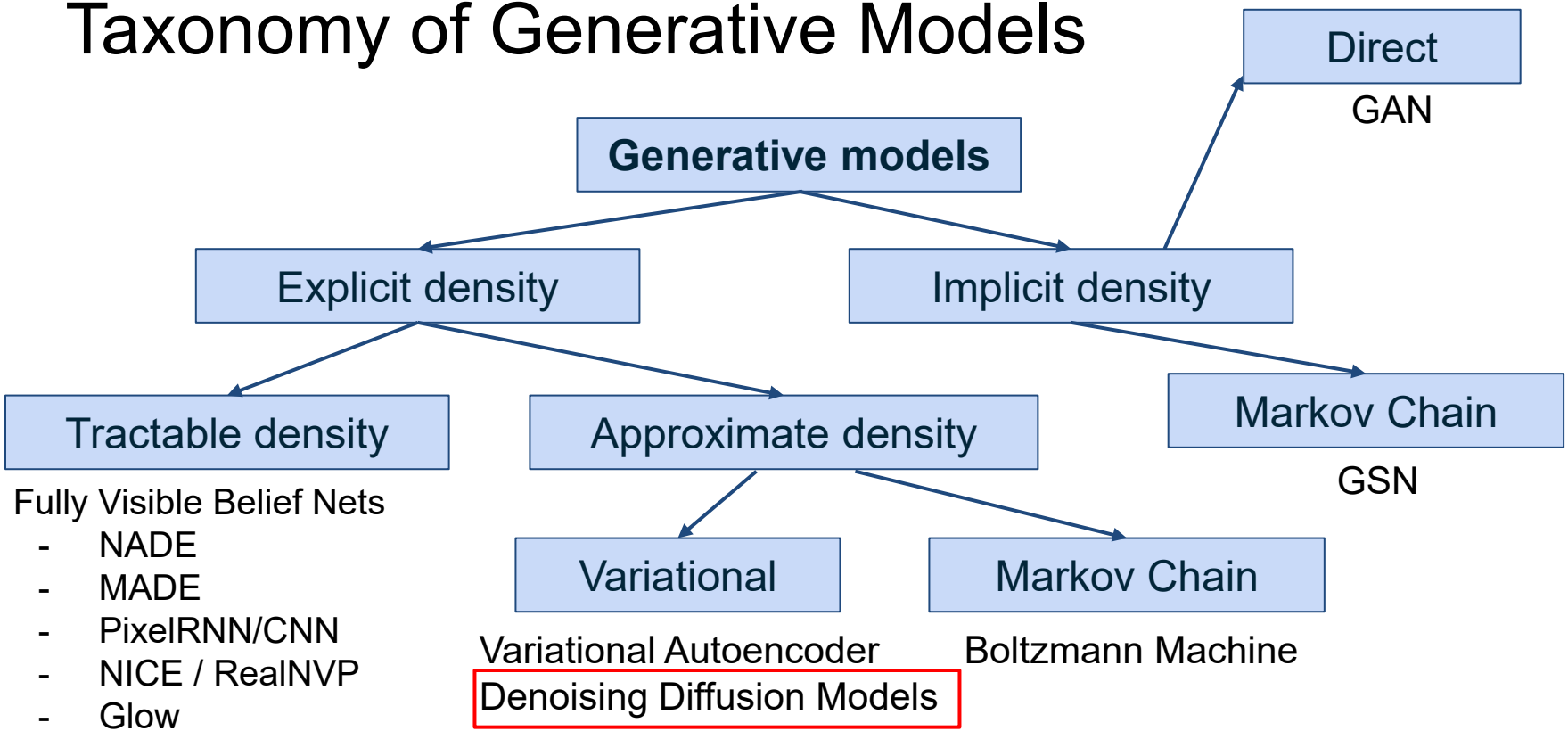


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Denoising Diffusion Probabilistic Models (DDPMs)

And Conditional Diffusion Models

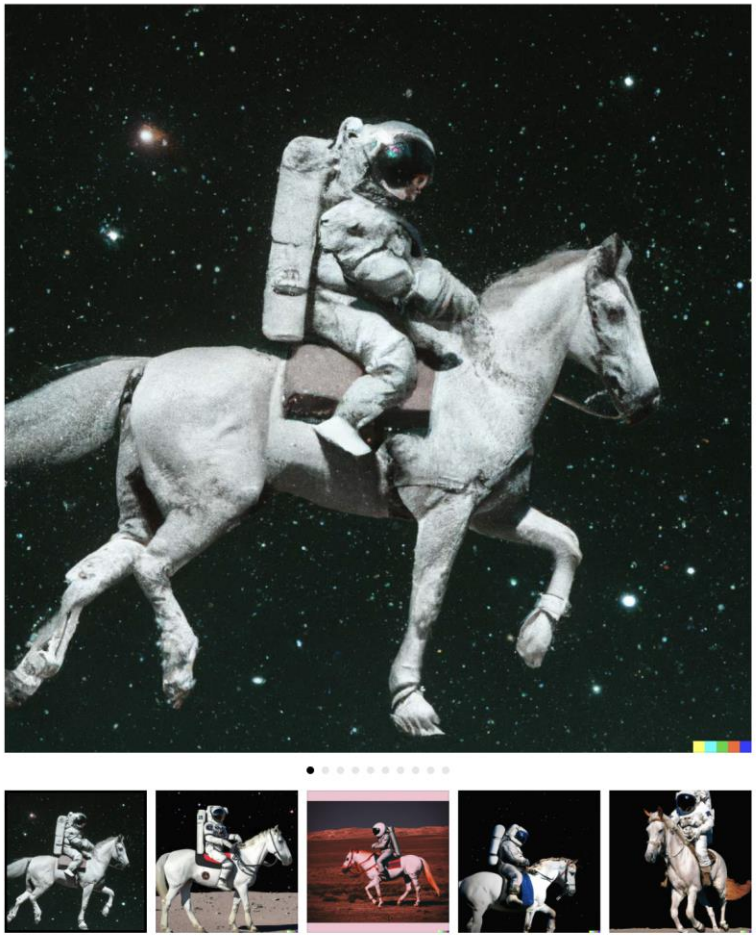
TEXT DESCRIPTION

An astronaut Teddy bears A bowl of soup

riding a horse lounging in a tropical resort in space playing basketball with cats in space

in a photorealistic style in the style of Andy Warhol as a pencil drawing

DALL·E 2



https://openai.com/dall-e-2/

Landscape Highlights of Diffusion Models (Nov 2022)

basic principles

- *Diffusion probabilistic models* (Sohl-Dickstein et al., 2015)
- *Noise-conditioned score network* (**NCSN**; Yang & Ermon, 2019)
- *Denoising diffusion probabilistic models* (**DDPM**; Ho et al. 2020)

conditional & high-res image generation

- *Classifier-guided conditional generation* (Dhariwal and Nichole, 2021)
- *Classifier-free Diffusion Guidance* (Ho and Salimans, 2022)
- *Latent-space Diffusion* (**StableDiffusion**; Rombach and Blattmann et al., 2022)

new applications

- *Planning with Diffusion for Flexible Behavior Synthesis* (**Diffuser**; Janner et al., 2022)
- *DreamFusion: Text-to-3D using 2D Diffusion* (Poole and Jain et al., 2022)
- *Make-A-Video: Text-to-Video Generation without Text-Video Data* (Singer et al., 2022)

How to make a new generative model

- **Setting:** Given unlabeled dataset of data, I want to learn to sample from P(x)
- Define the generative process
- Parameterize it
- Maximum likelihood (often -> KL-divergence)
- Approximations
- Optimize parameters!
- Add conditioning, e.g. text

Landscape Highlights of Diffusion Models (Nov 2022)

basic principles

- *Diffusion probabilistic models* (Sohl-Dickstein et al., 2015)
- *Noise-conditioned score network* (**NCSN**; Yang & Ermon, 2019)
- *Denoising diffusion probabilistic models* (**DDPM**; Ho et al. 2020)

conditional & high-res image generation

- *Classifier-guided conditional generation* (Dhariwal and Nichole, 2021)
- *Classifier-free Diffusion Guidance* (Ho and Salimans, 2022)
- *Latent-space Diffusion* (**StableDiffusion**; Rombach and Blattmann et al., 2022)

new applications

- *Planning with Diffusion for Flexible Behavior Synthesis* (**Diffuser**; Janner et al., 2022)
- *DreamFusion: Text-to-3D using 2D Diffusion* (Poole and Jain et al., 2022)
- *Make-A-Video: Text-to-Video Generation without Text-Video Data* (Singer et al., 2022)

The Denoising Diffusion Process

image from dataset

x_0

The Denoising Diffusion Process

image from dataset

The “forward diffusion” process: add Gaussian noise each step

x_0 → x_1 → …

The Denoising Diffusion Process

image from dataset

The “forward diffusion” process: add Gaussian noise each step

noise $\mathcal{N}(0, I)$

$x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_{T-1} \longrightarrow x_T$

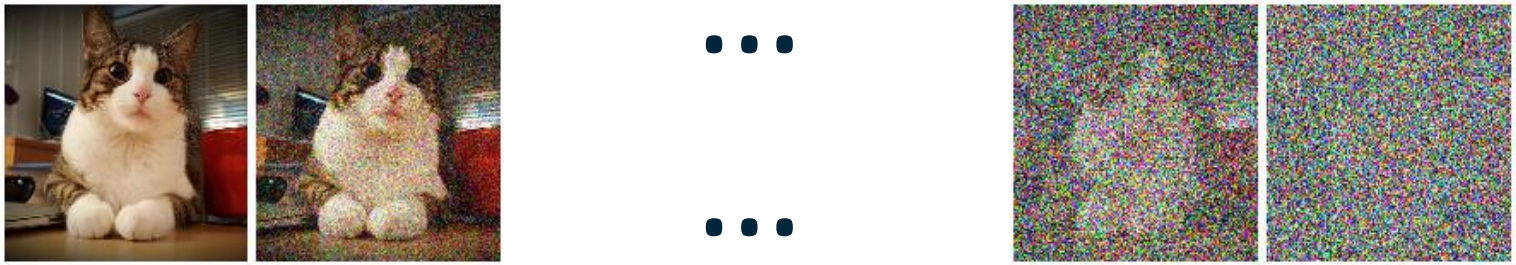
The Denoising Diffusion Process

image from dataset

The "forward diffusion" process: add Gaussian noise each step

noise $\mathcal{N}(0, I)$

$x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{T-1} \rightarrow x_T$

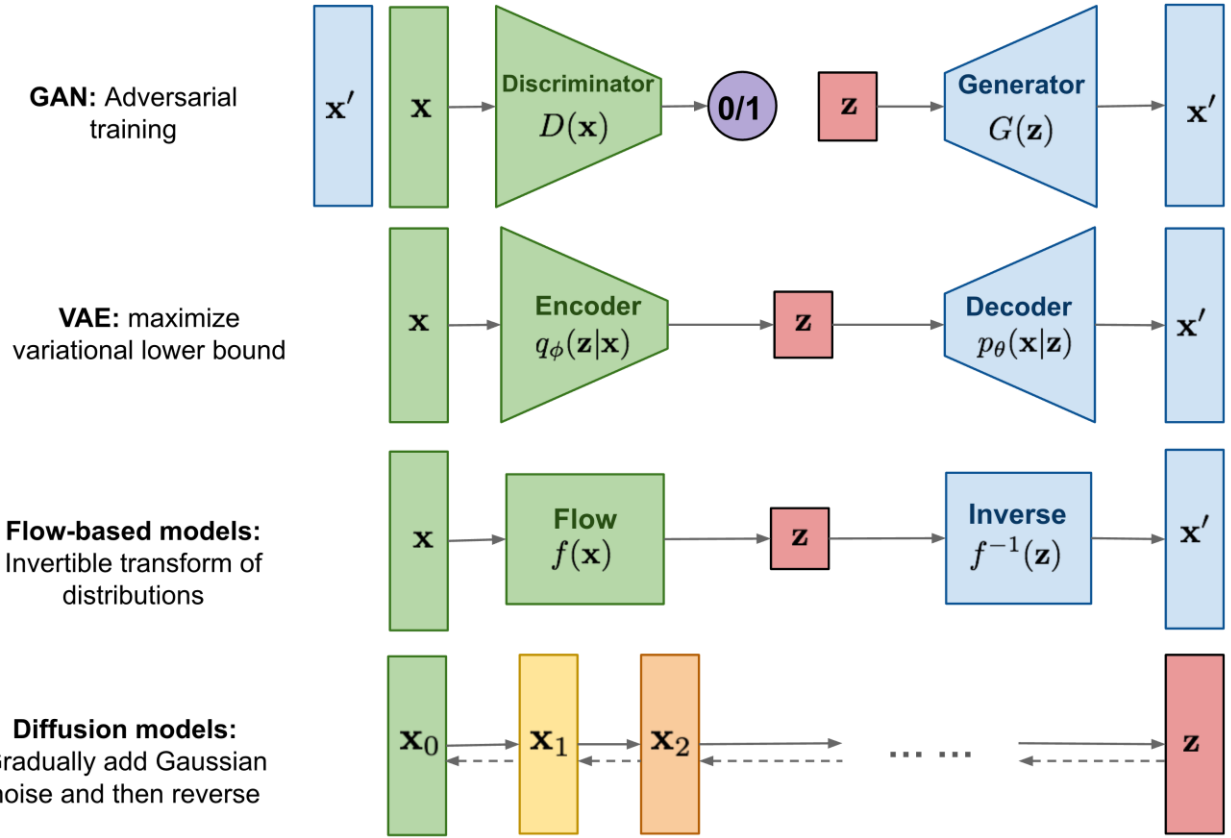


$x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_{T-1} \leftarrow x_T$

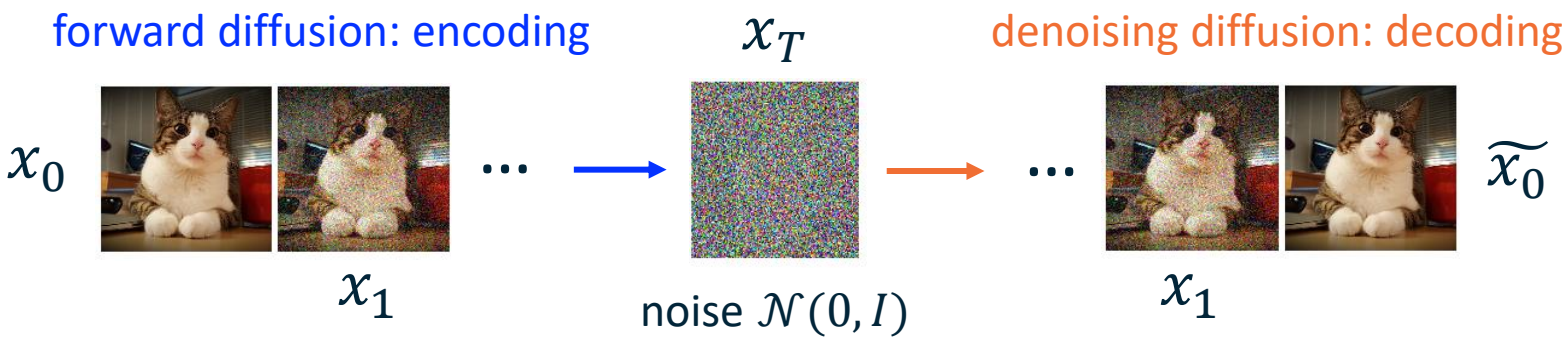
The "denoising diffusion" process: generate an image from noise by *denoising* the gaussian noises

Ties/inspiration form Annealed Imporantce Sampling in physics

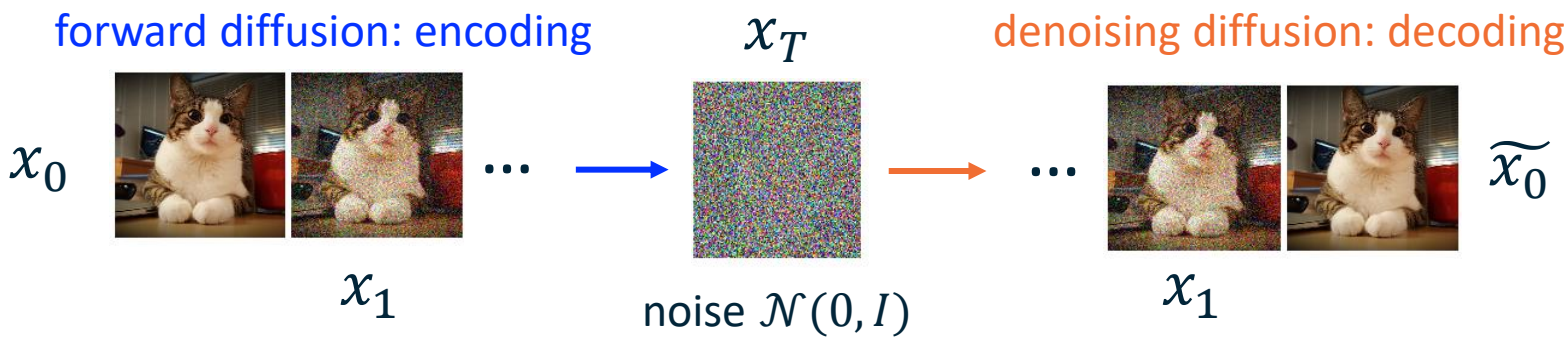
Comparison


GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$

Forward/Reverse Processes



Forward/Reverse Processes

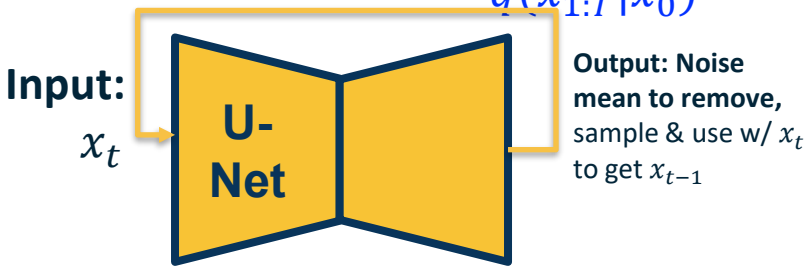


Known / predefined:

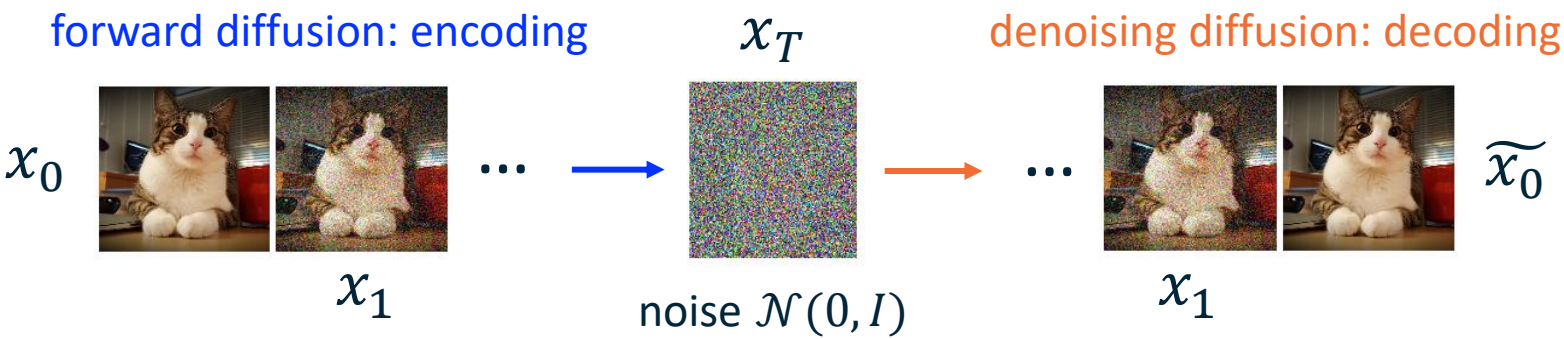
$q(x_{1:T}|x_0)$

Unknown / learned:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$



Forward/Reverse Processes



Known / predefined:

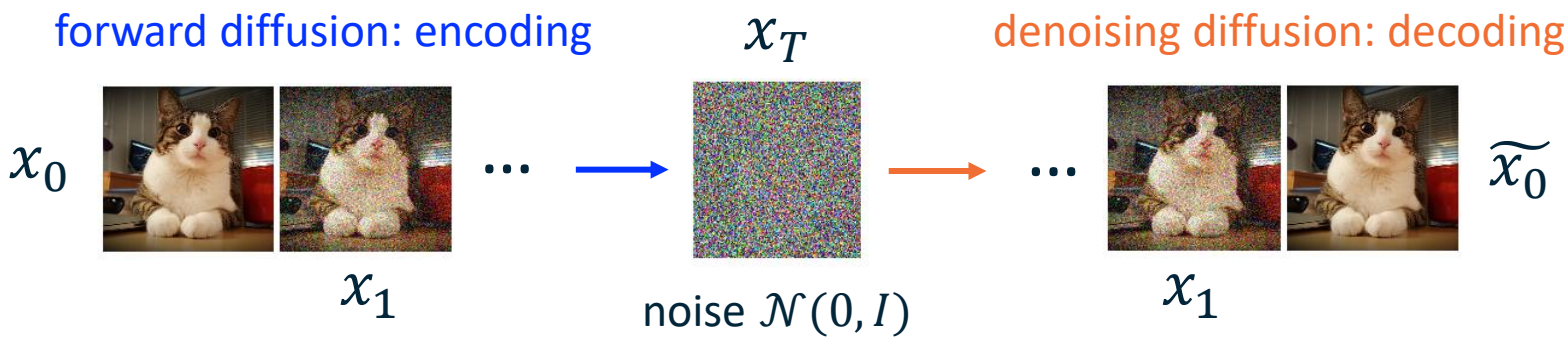
$q(x_{1:T}|x_0)$

Unknown / learned:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

Use the denoising decoding process to generate new images.

Forward/Reverse Processes



Known / predefined:
$q(x_{1:T}|x_0)$

Unknown / learned:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

Notation: A Gaussian distribution "for" x_t

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

β_t is the *variance schedule* at the diffusion step t

The Diffusion (Encoding) Process

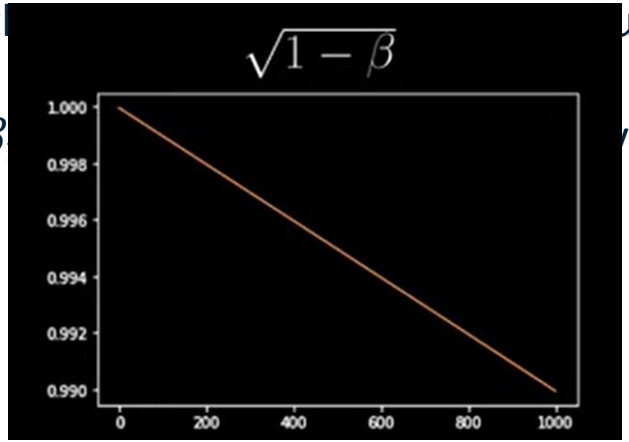
The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

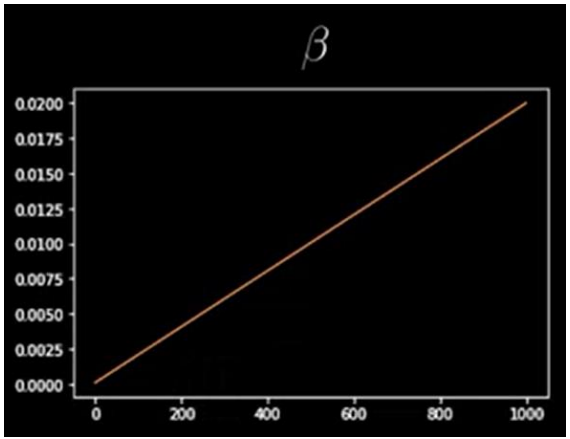
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

β_t is t…usion

$0 < \beta$… …value … $= 1000$





https://www.youtube.com/watch?v=HoKDTa5jHvg&t=517s

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

β_t is the *variance schedule* at the diffusion step t

$0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$, typical value range $[0.0001, 0.02]$, with $T = 1000$

$x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_{T-1} \longrightarrow x_T$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

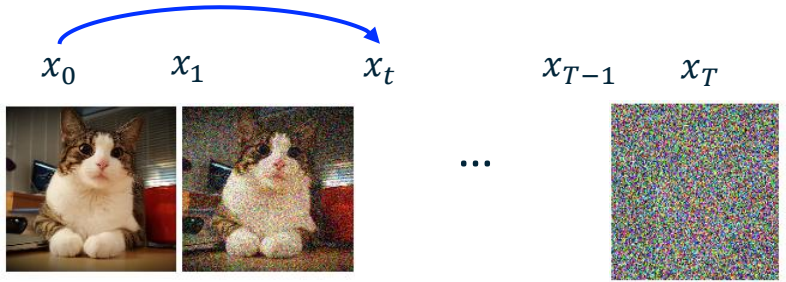
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

, where $\alpha_t = (1-\beta_t)$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$



The Diffusion (Encoding

https://www.youtube.com/watch?v=HoKDTa5jHvg&t=517s

$$\bar{\alpha}_t = \prod_{s=1} a_s$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$= \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$$

$$= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\,\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}\, x_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\,\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_1\alpha_0}\, x_0 + \sqrt{1-\alpha_t\alpha_{t-1}\ldots\alpha_1\alpha_0}\,\epsilon$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t)I) \longleftarrow \boxed{= \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}$$

The **known** forward process x_0 —

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probab

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

Gaussian reparameterization trick:

$$z = \mu + \epsilon * \sigma, \epsilon \sim N(0,1)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

Intuition: We know all distributions in forward process, and can in fact directly compute for any t based on X_0

(square root appears because reparameterization trick has just σ)

The Diffusion and Denoising Process