Lecture 18:
Generative AI Part 2

GANs & Diffusion

Administrative

- Milestone was due last week
- **Quiz 5** (last quiz) will take place last 30 minutes of next lecture
- **Assignment 5** due Friday

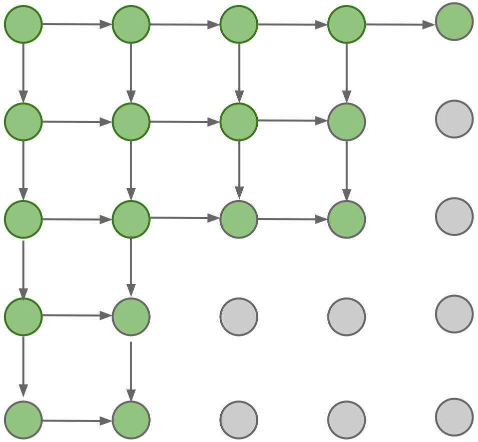
- If you want us to print your poster, check the course website for details!

Generative AI so far: Autoregressive models

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Very slow during both training and testing; N x N image requires 2N-1 sequential steps!

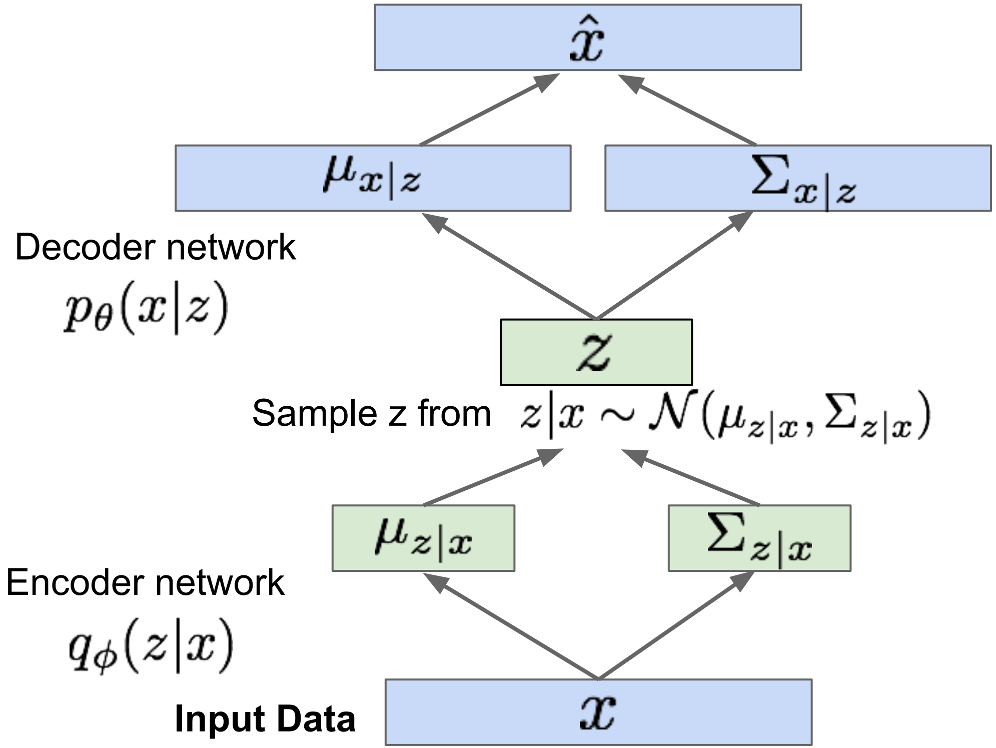


[van der Oord et al. 2016]

Generative AI so far: Variational Autoencoders

Maximizing the likelihood lower bound

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



$\hat{x}$

$\mu_{x|z}$ $\Sigma_{x|z}$

Decoder network

$p_\theta(x|z)$

z

Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$

$\mu_{z|x}$ $\Sigma_{z|x}$

Encoder network

$q_\phi(z|x)$

Input Data x

Comparing the two methods so far

Autoregressive model
- Directly maximize p(data)
- High-quality generated images
- Slow to generate images
- No explicit latent codes

Variational model
- Maximize lower bound on p(data)
- Generated images often blurry
- Very fast to generate images
- Learn rich latent codes
  - (although not really)

Comparing the two methods so far

Autoregressive model

- Directly maximize p(data)
- High-quality generated images
- Slow to generate images
- No explicit latent codes

Variational model

- Maximize lower bound on p(data)
- Generated images often blurry
- Very fast to generate images
- Learn rich latent codes
    - (although not really)

So far...

Autoregressive define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

What if we didn't try to model *p(data)* at all?

Generative Adversarial Networks (GANs)

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. We want to learn a function that samples from p_{data}.

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. We want to learn a function that samples from p_{data}.

Idea: Introduce a latent variable z with simple prior p(z) (e.g. assume z is a multivariate gaussian). Sample z ~ p(z) and pass to a Generator Network $x = G(z)$

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. We want to learn a function that samples from p_{data}.

Idea: Introduce a latent variable z with simple prior $p(z)$ (e.g. assume z is a multivariate gaussian). Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$

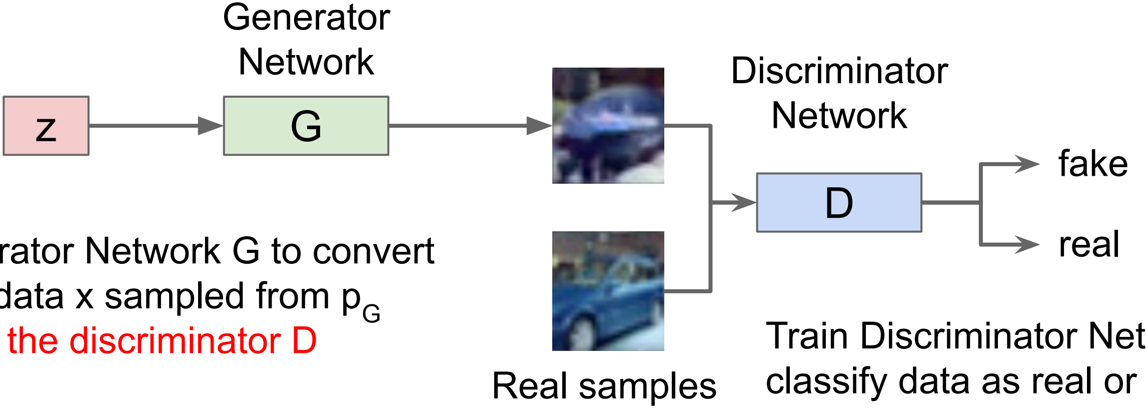
Then x is a sample from the Generator distribution p_G. **We just need to make sure $p_G = p_{data}$!**

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. We want to learn a function that samples from p_{data}.

Idea: Introduce a latent variable z with simple prior $p(z)$ (e.g. assume z is a multivariate gaussian). Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$

Then x is a sample from the Generator distribution p_G. **We just need to make sure $p_G = p_{data}$!**



Train Generator Network G to convert z into fake data x sampled from p_G by "fooling" the discriminator D

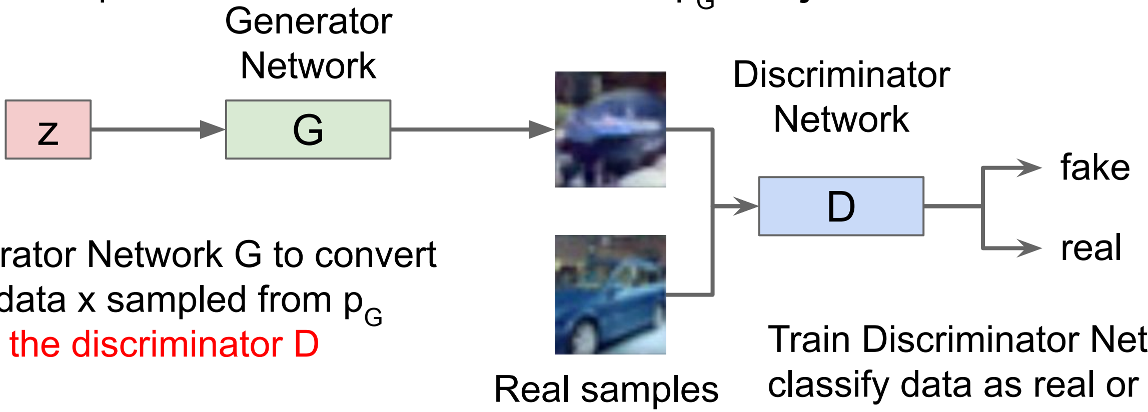
Train Discriminator Network D to classify data as real or fake (1/0)

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. We want to learn a function that samples from p_{data}.

Idea: Introduce a latent variable z with simple prior $p(z)$ (e.g. assume z is a multivariate gaussian). Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$

Then x is a sample from the Generator distribution p_G. **We just need to make sure $p_G = p_{data}$!**



Train Generator Network G to convert z into fake data x sampled from p_G by ”fooling” the discriminator D

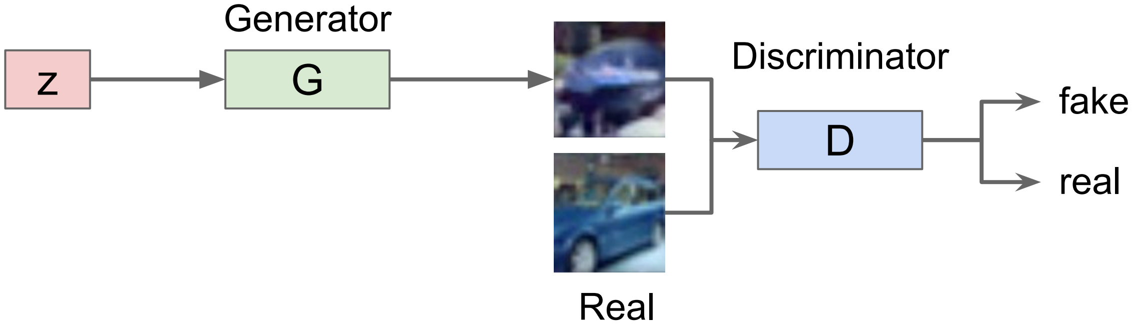
Train Discriminator Network D to classify data as real or fake (1/0)

Jointly train G and D. Hopefully p_G converges to p_{data}!

Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

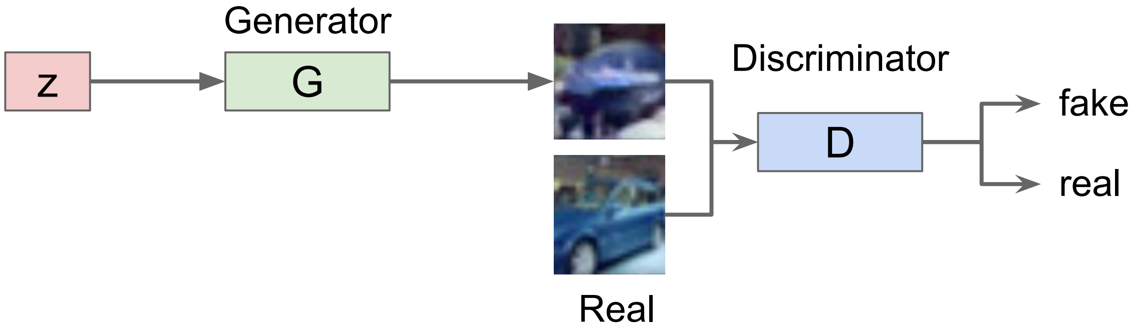
$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$



Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

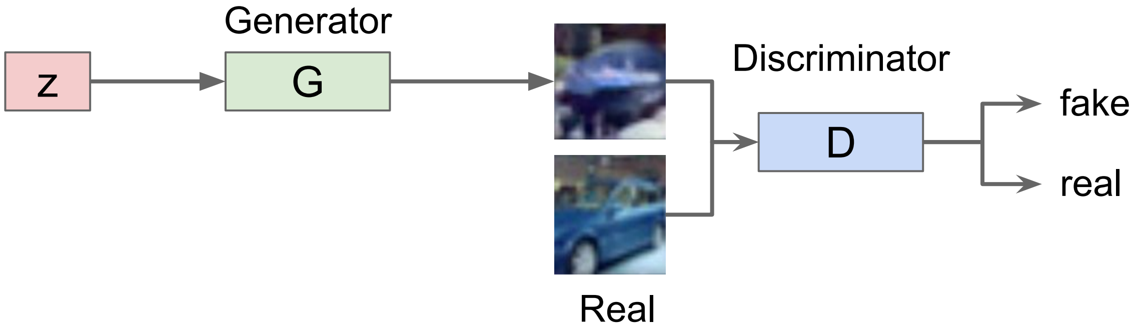
$$\min_{G} \boxed{\max_{D}} \left(\overbrace{E_{x\sim p_{data}}[\log D(x)]}^{\text{Discriminator wants D(x) = 1 for real data}} + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$



Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

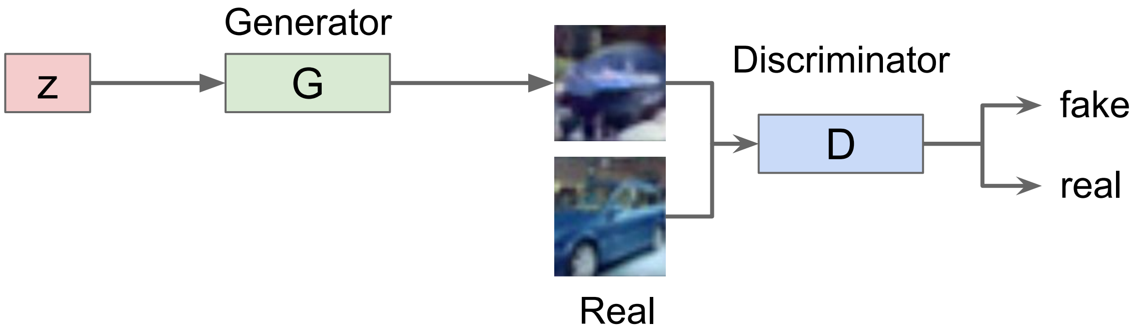
$$\min_{G} \boxed{\max_{D}} \left(\overbrace{E_{x\sim p_{data}}[\log D(x)]}^{\textbf{Discriminator wants } \mathbf{D(x) = 1} \textbf{ for real data}} + \overbrace{E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]}^{\textbf{Discriminator wants } \mathbf{D(x) = 0} \textbf{ for fake data}} \right)$$



Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

$$\boxed{\min_{G}}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + \overbrace{E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]}^{\textbf{Generator wants D(x) = 1 for fake data}}\right)$$



Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_{G} \max_{D} V(G, D)$$

Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_{G} \max_{D} V(G, D)$$

For t in 1, ... T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

$$= \min_{G}\max_{D} V(G, D)$$

For t in 1, ... T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

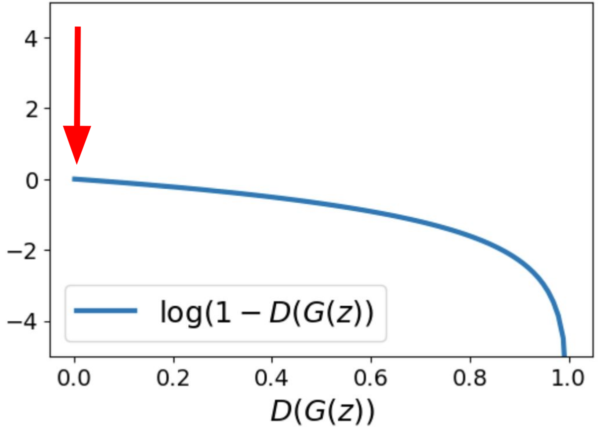
We are not minimizing any overall loss! No training curves to look at!

Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0



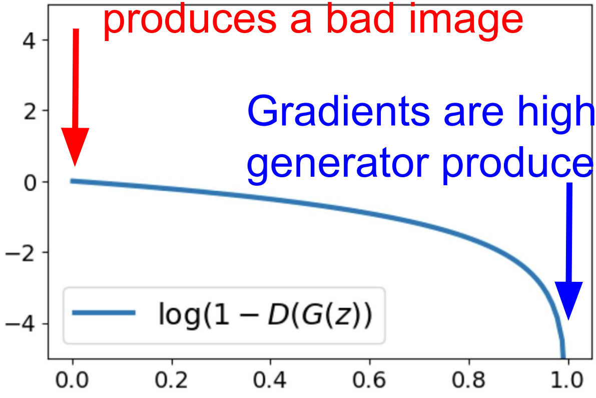
Generative Adversarial Networks

Jointly train generator G and discriminator D with a minimax game

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1-D(G(z))\right)\right]\right)$$

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0

Problem: Why is this a problem?



Gradients are near 0 when G produces a bad image

Gradients are high when generator produces good

Generative Adversarial Networks

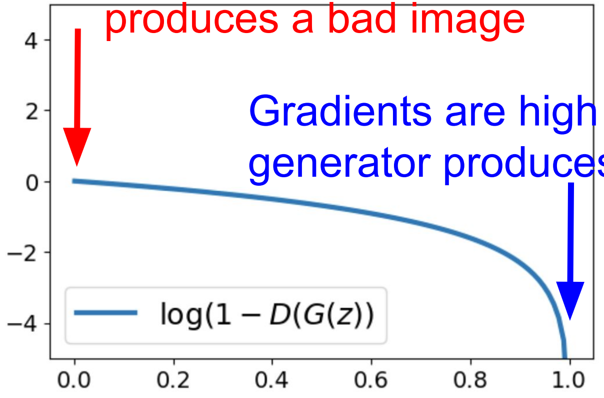
Jointly train generator G and discriminator D with a minimax game

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0

Problem: Vanishing gradients for G

How do we fix this?



Gradients are near 0 when G produces a bad image

Gradients are high when generator produces good

Generative Adversarial Networks

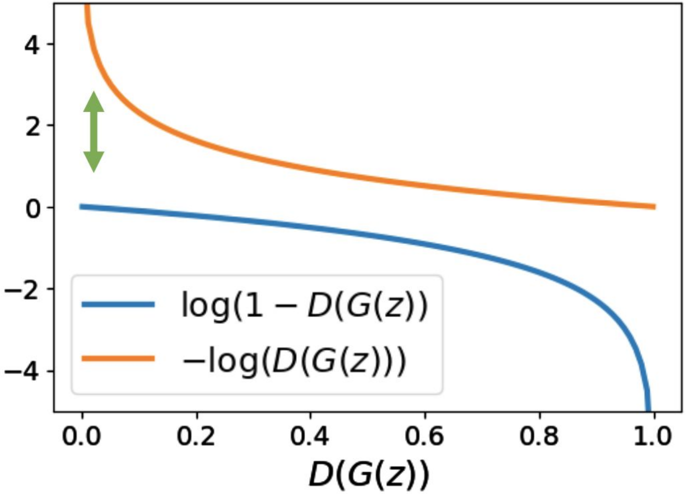
Jointly train generator G and discriminator D with a minimax game

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0

Problem: Vanishing gradients for G

Solution: Train G to minimize **–log(D(G(z))**, instead of **log(1-D(G(z))**. Then G gets strong gradients at start of training! (Wasserstein GAN)



Generative Adversarial Networks

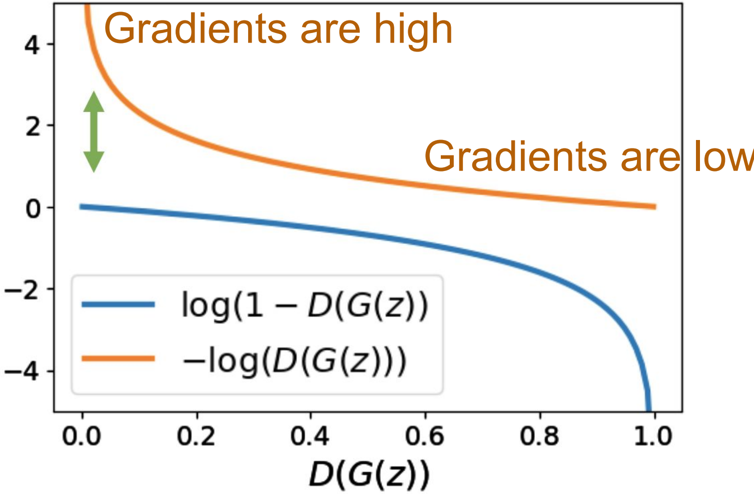
Jointly train generator G and discriminator D with a minimax game

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0

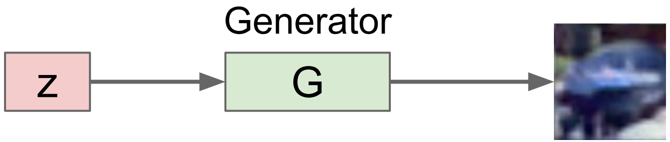
Problem: Vanishing gradients for G

Solution: Train G to minimize **–log(D(G(z))**, instead of **log(1-D(G(z))**. Then G gets strong gradients at start of training!



Generative adversarial networks

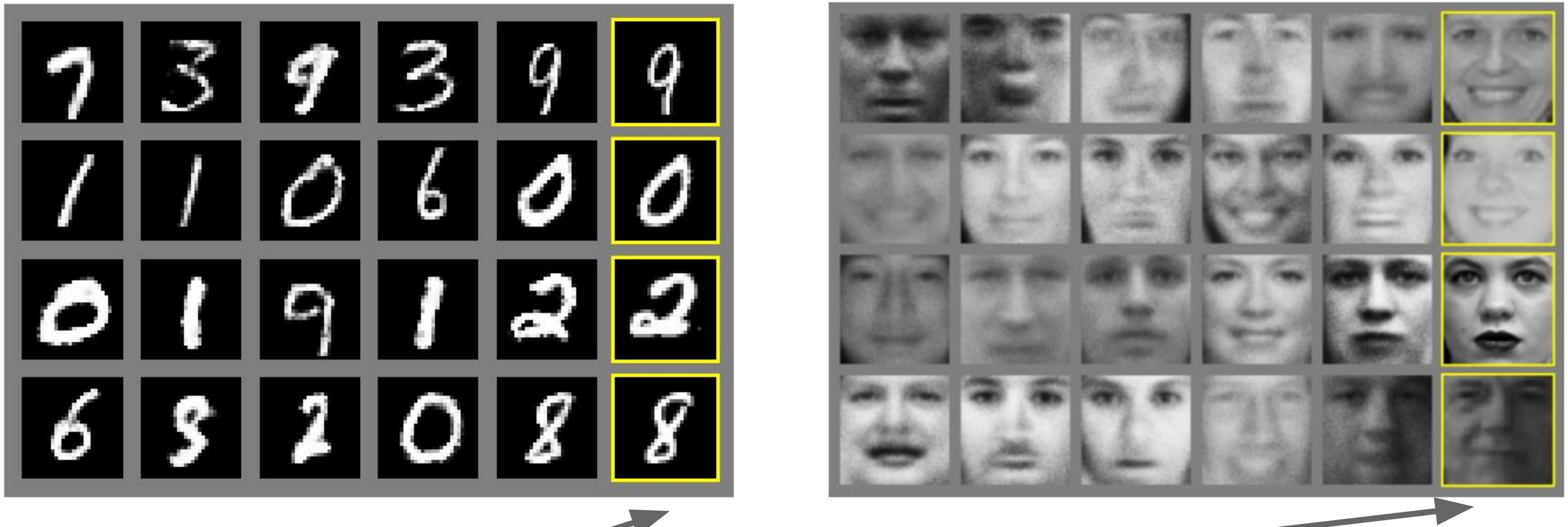
Once trained, throw away the discriminator and use G to generate new images



Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generative Adversarial Nets

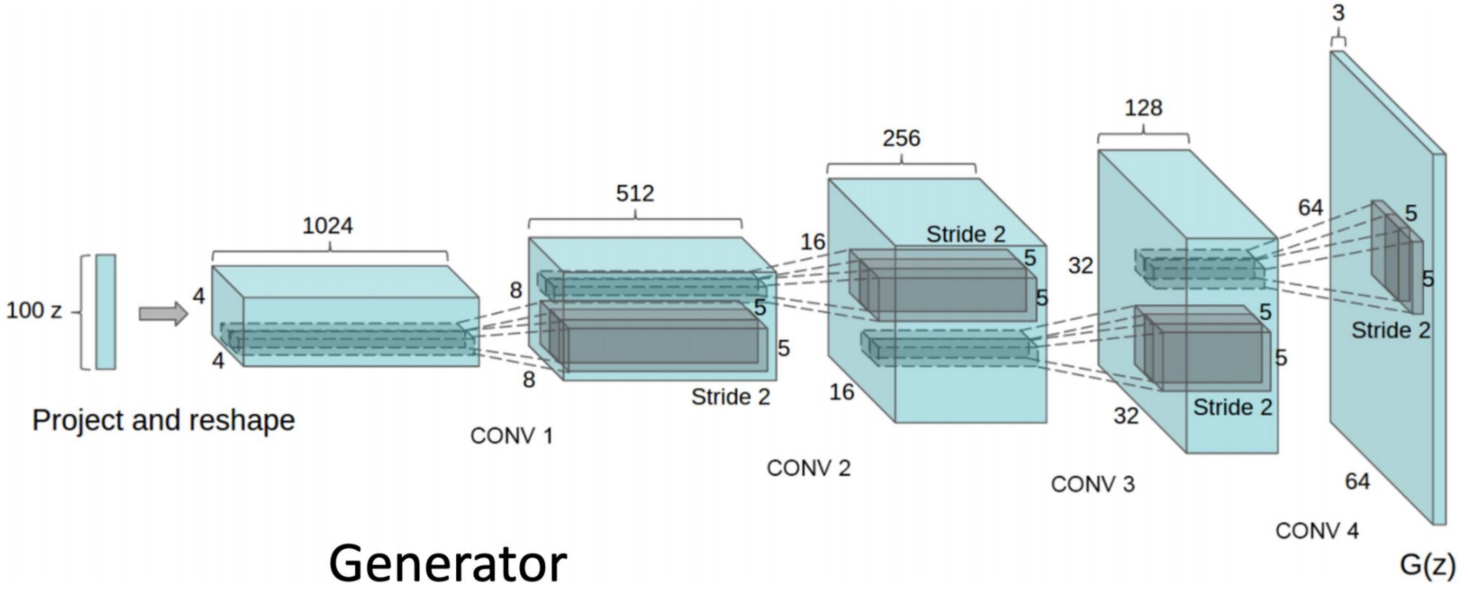
Generated samples



Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.

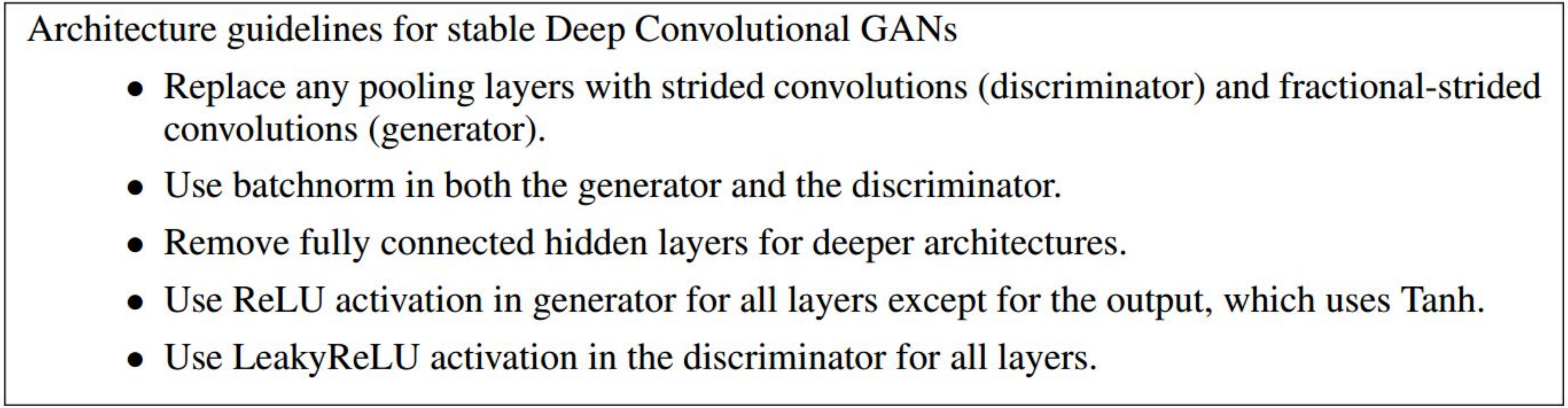
Generative Adversarial Nets: Convolutional Architectures



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network



Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

Radford et al, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in laten space

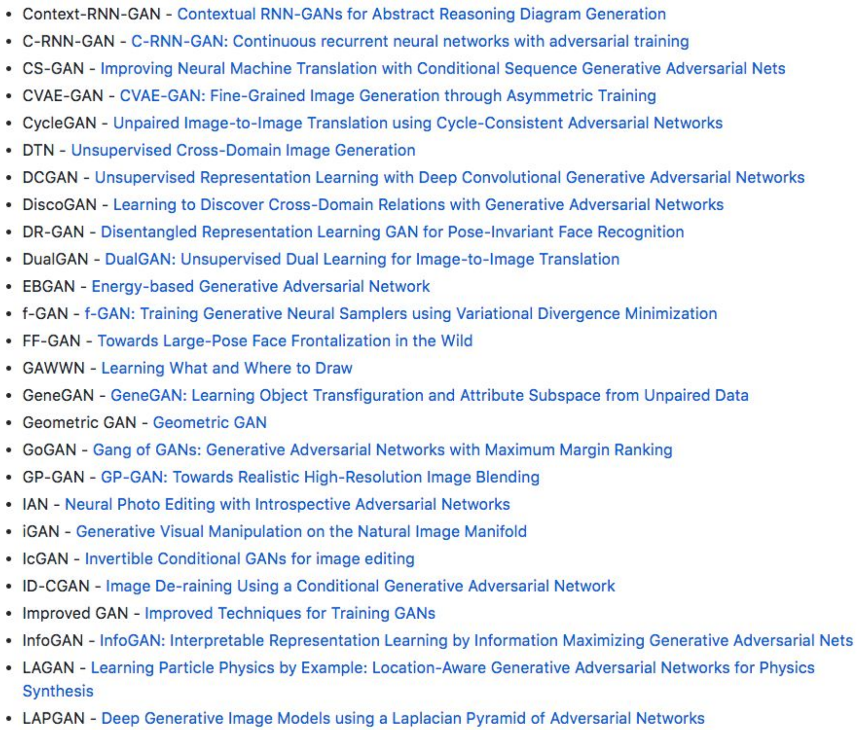
Radford et al,
ICLR 2016

Since then: Explosion of GANs

“The GAN Zoo”

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks



- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

GAN improvements: higher resolution

256 x 256 bedrooms

1024 x 1024 faces

Progressive GAN, Karras 2018.

GAN transformations

Source->Target domain transfer

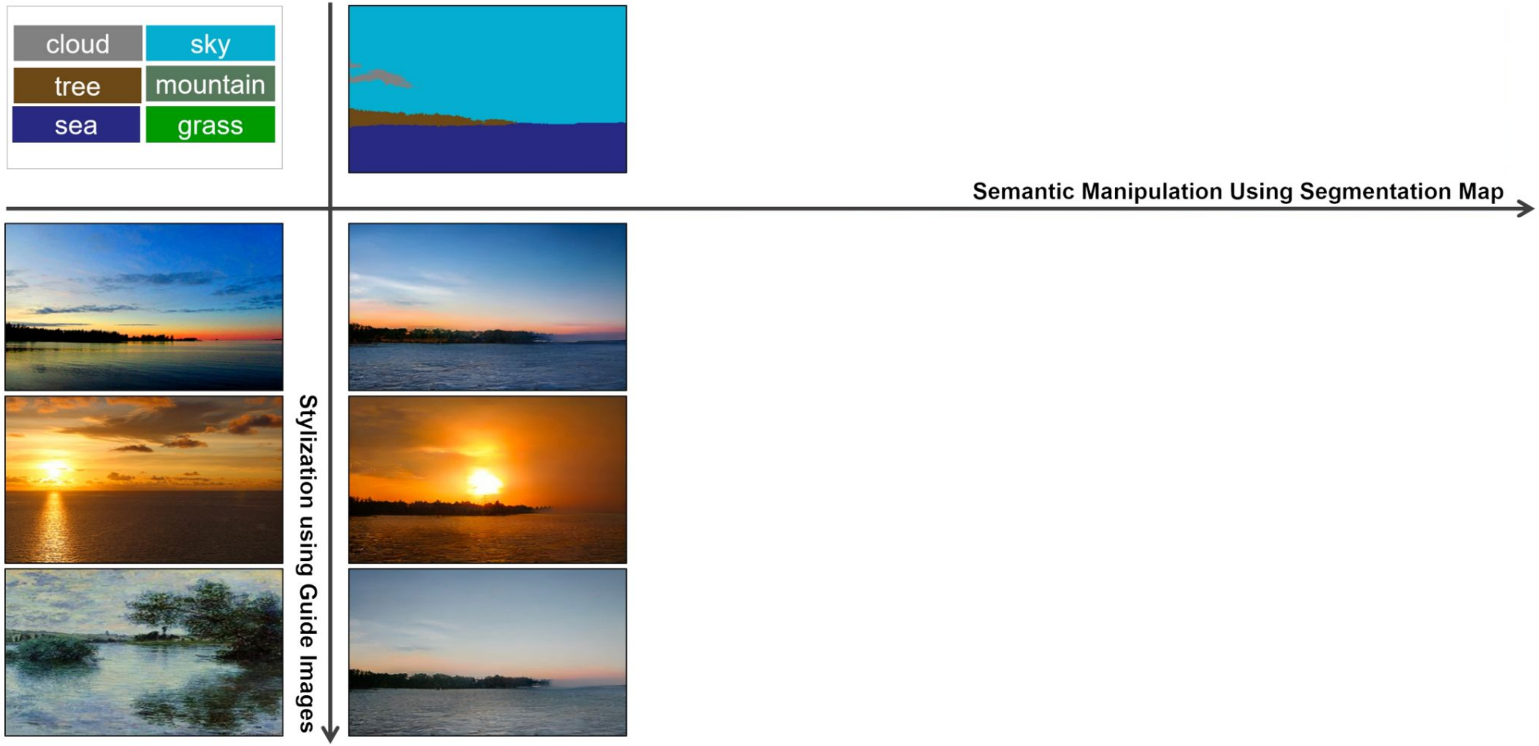
CycleGAN. Zhu et al. 2017.

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

BigGAN: 512x512 images

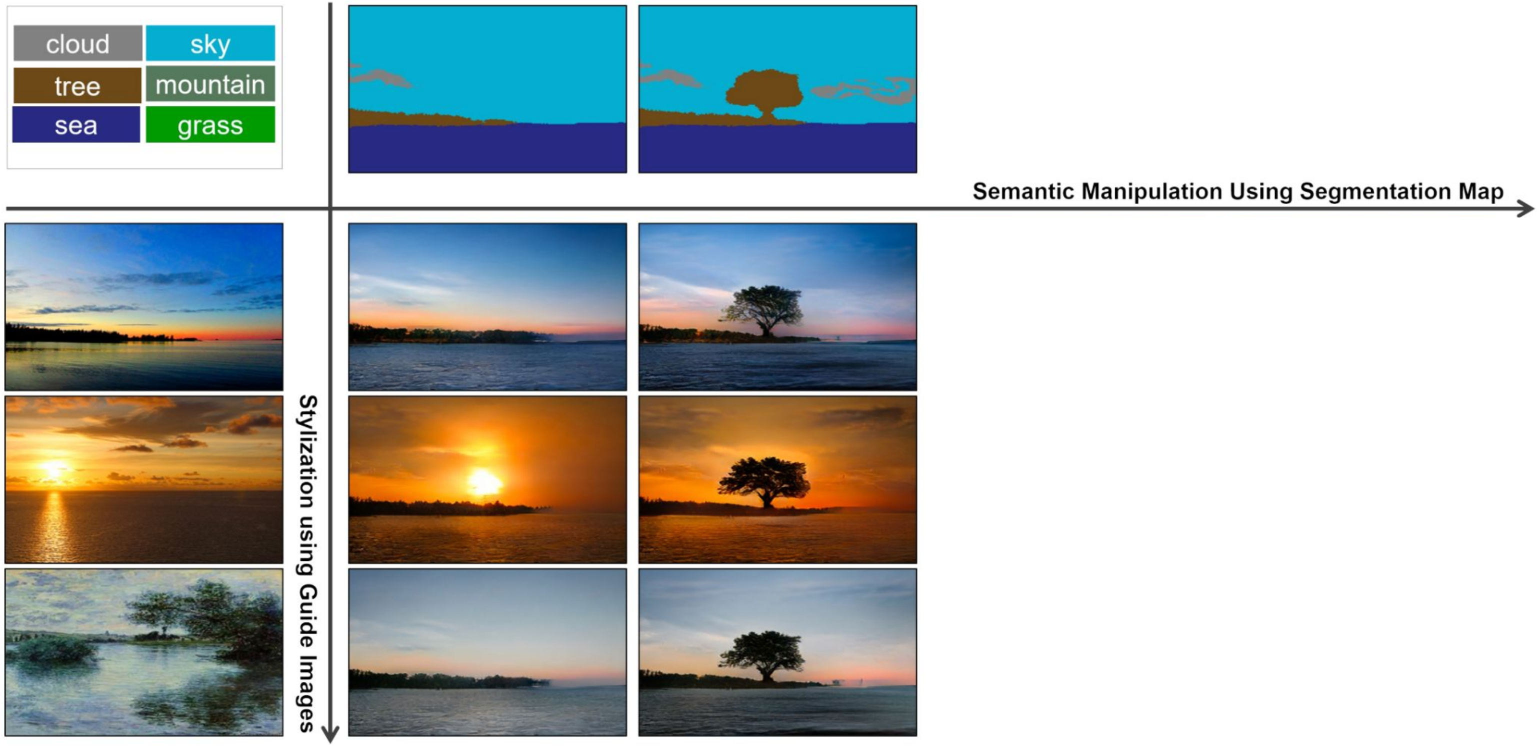
Brock et al., 2019

Controlled generation with GANs



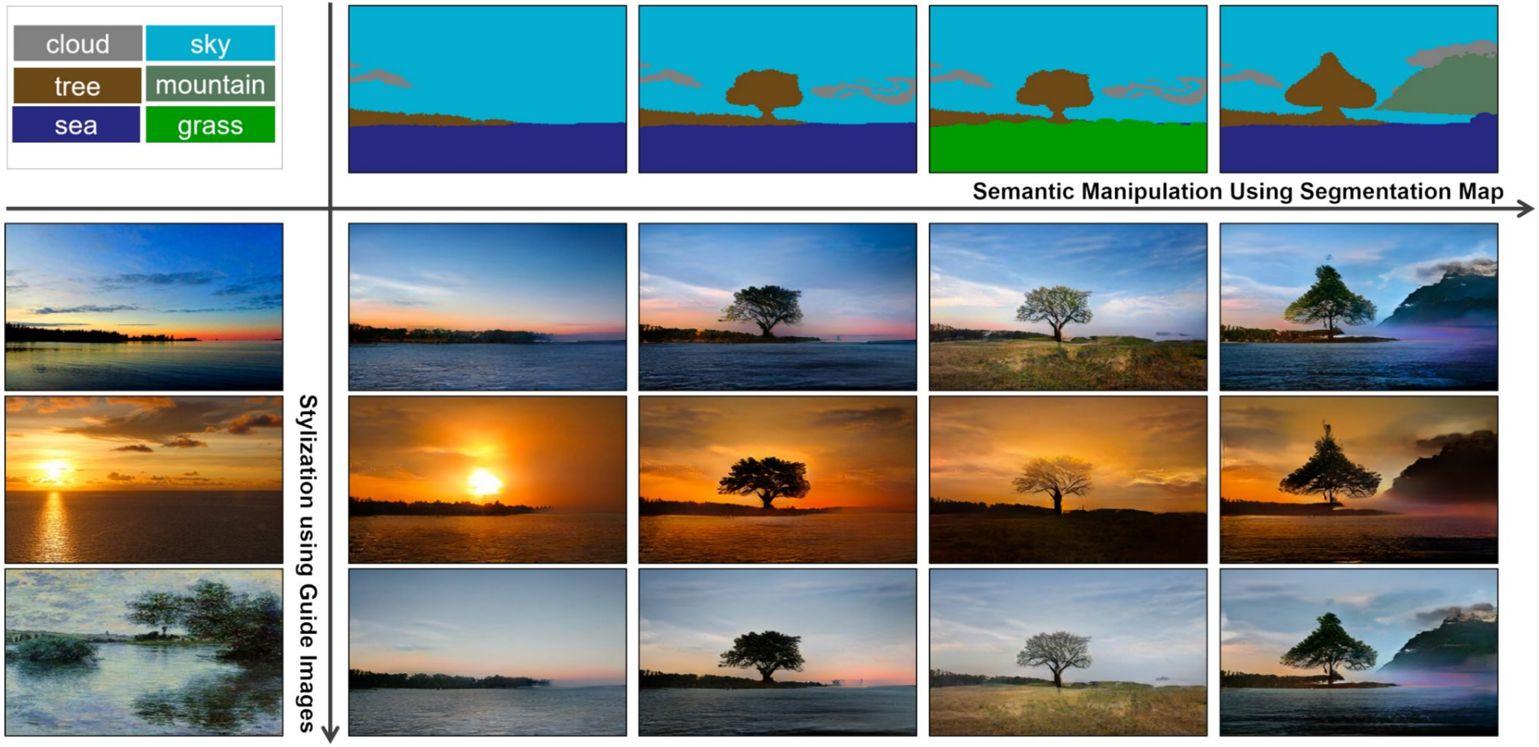
Park et al, “Semantic Image Synthesis with Spatially-Adaptive Normalization”, CVPR 2019

Controlled generation with GANs



Park et al, “Semantic Image Synthesis with Spatially-Adaptive Normalization”, CVPR 2019

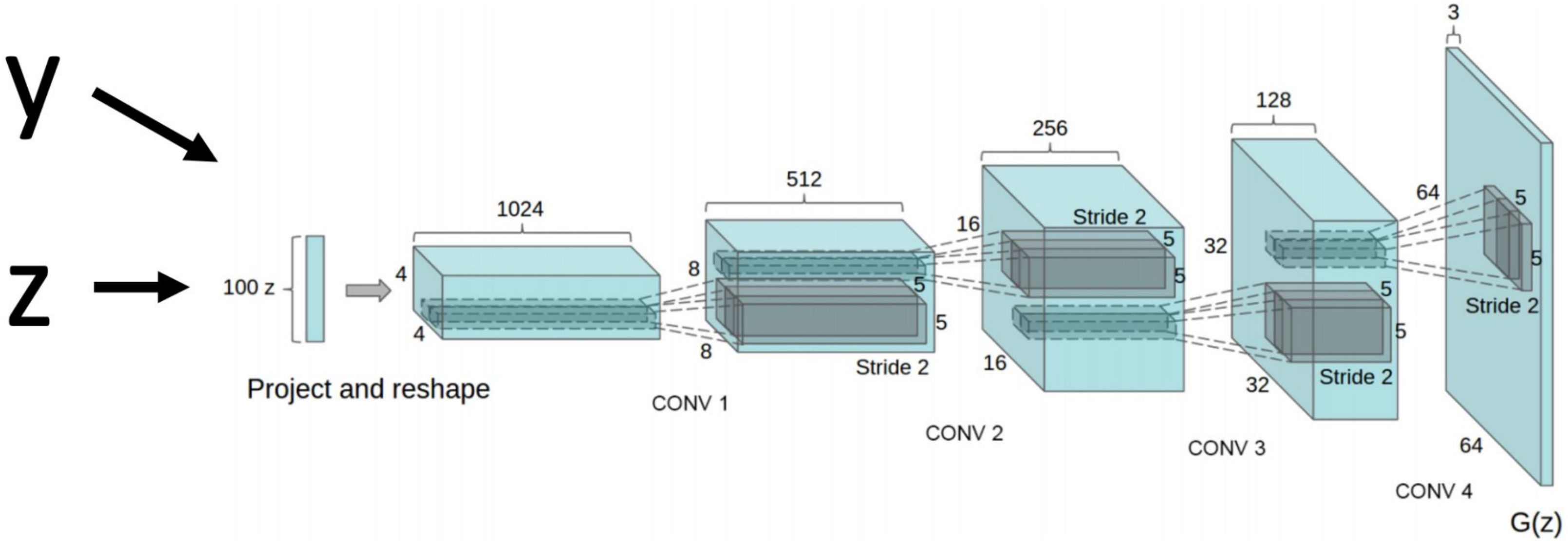
Controlled generation with GANs



Park et al, “Semantic Image Synthesis with Spatially-Adaptive Normalization”, CVPR 2019

Conditional GANs: StyleGAN

Y is text that describes the image you want to generate



Karras et al, “Analyzing and Improving the Image Quality of StyleGAN”, CVPR 2020

Conditional GANs: StyleGAN

Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_{i,j} - \mu_j\right)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Learn a separate scale and shift for each different label y

Conditional Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_{i,j} - \mu_j\right)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \boldsymbol{\gamma_j^y} \hat{x}_{i,j} + \boldsymbol{\beta_j^y}$$

Karras et al, "Analyzing and Improving the Image Quality of StyleGAN", CVPR 2020

Summary: GANs

Don't work with an explicit density function
Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:

- Beautiful, state-of-the-art samples!

Cons:

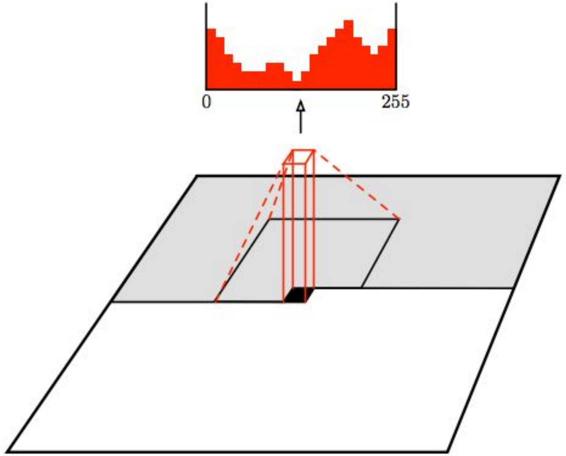
- Trickier / more unstable to train

Active areas of research:

- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

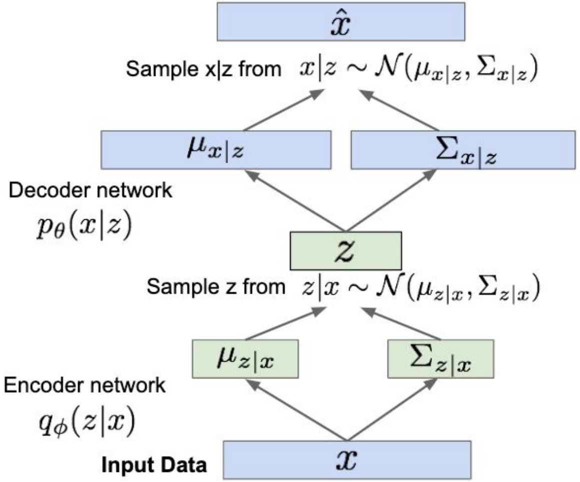
(Broader) Summary

Autoregressive models:
PixelRNN, PixelCNN



Van der Oord et al, “Conditional image generation with pixelCNN decoders”, NIPS 2016

Variational Autoencoders



$\hat{x}$

Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

$\mu_{x|z}$ $\Sigma_{x|z}$

Decoder network
$p_\theta(x|z)$

z

Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$

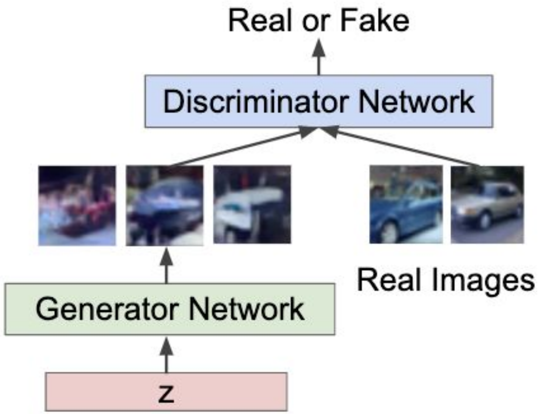
$\mu_{z|x}$ $\Sigma_{z|x}$

Encoder network
$q_\phi(z|x)$

Input Data x

Kingma and Welling, “Auto-encoding variational bayes”, ICLR 2013

Generative Adversarial Networks (GANs)



Real or Fake

Discriminator Network

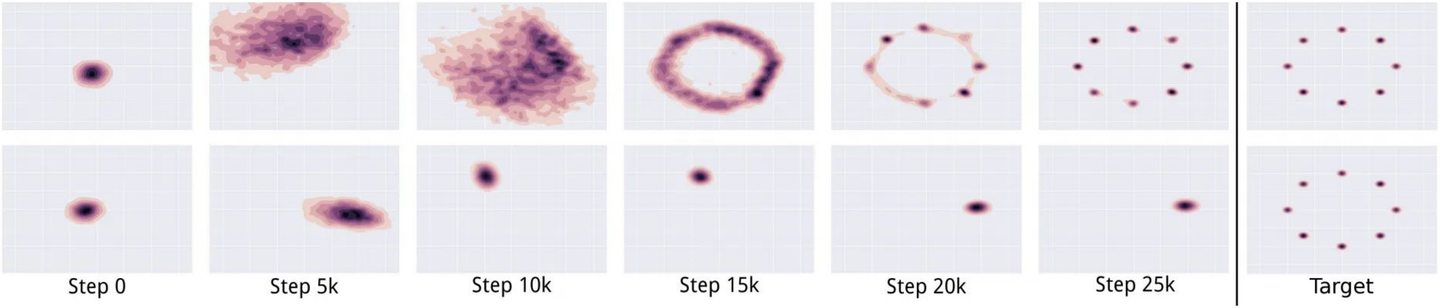
Real Images

Generator Network

z

Goodfellow et al, “Generative Adversarial Nets”, NIPS 2014

A problem with GANs: mode collapse



A problem with VAEs: posterior collapse

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

Enter Diffusion models

Diffusion Models are outperforming GANs

Dhariwal & Nichol. "Diffusion Models Beat GANs on Image Synthesis", OpenAI 2021

Ho et al. "Cascaded Diffusion Models for High Fidelity Image Generation", Google 2021

Text-to-Image (T2I) Generation

Dall-E2

“a teddy bear on a skateboard in times square”

Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents” 2022

Imagen

“A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.”

Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding” 2022

Text-to-Image (T2I) Generation

Stable Diffusion

Mega thread on Twitter/X about Stable Diffusion

Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models” 2022

But what is a diffusion model?

What takeaways do we have from previous models?

Autoregressive define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

GANs give up on explicitly modeling density and just learn to sample “real” data

What takeaways do we have from previous models?

Autoregressive define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

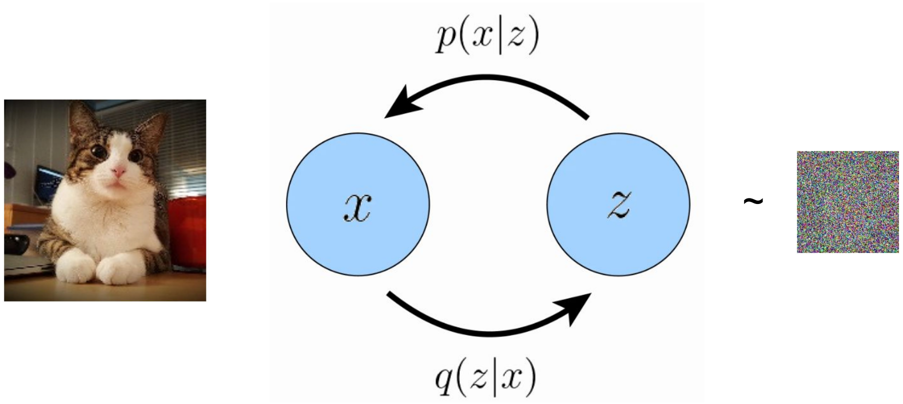
Cannot optimize directly, derive and optimize lower bound on likelihood instead

GANs give up on explicitly modeling density and just learns to sample “real” data

VAEs and GANs model in one forward step, but AR models use an iterative process

VAEs for images look like this

Decoder



Encoder

- We learn 2 networks, one to encode and one to decode
- We ensure that z is similar to a unit normal noise
- To sample new images, we can sample from the unit normal and decode in 1 step

The lower bound we derived last lecture

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Logarithms}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling (need some trick to differentiate through sampling).

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.

Two loss objectives for VAEs

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

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

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

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

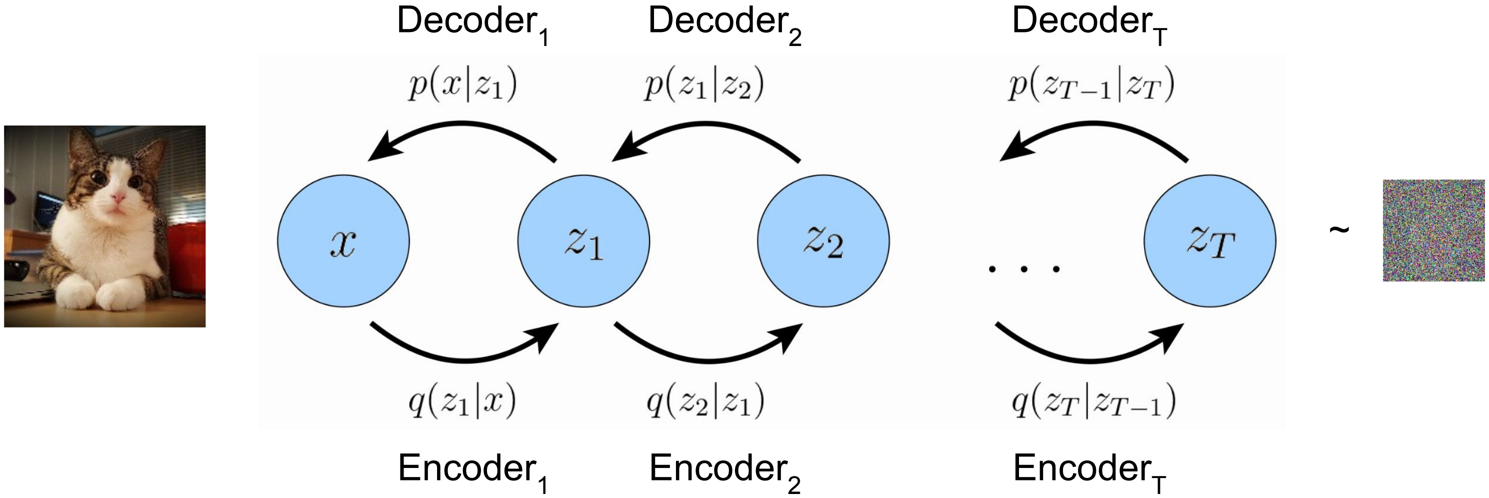
$$= \boxed{\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\boxed{\geq 0}}$$

Decoder:
reconstruct
the input data

Encoder:
make approximate
posterior distribution
close to prior

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Markovian Hierarchical VAEs



- We learn 2T networks, one to encode and one to decode
- We ensure that z_T is similar to a unit normal noise
- To sample new images, we can sample from the unit normal and decode in T step

Markovian Hierarchical VAEs - same derivation

$$\log p(x) = \mathbb{E}_{z_{1:T} \sim q_\phi(z_{1:T}|x)}[\log p_\theta(x^{(i)})]$$ $p_\theta(x)$ is independent of $z_{1:T}$

$$= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{p_\theta(z_{1:T}|x)}]$$ Bayes rule

Markovian Hierarchical VAEs - same derivation

$$\log p(x) = \mathbb{E}_{z_{1:T}\sim q_\phi(z_{1:T}|x)}[\log p_\theta(x^{(i)})] \qquad p_\theta(x) \text{ is independent of } z_{1:T}$$

$$= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{p_\theta(z_{1:T}|x)}] \qquad \text{Bayes rule}$$

$$= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{p_\theta(z_{1:T}|x)} \frac{q_\phi(z_{1:T}|x)}{q_\phi(z_{1:T}|x)}] \qquad \text{Multiplying by a constant}$$

$$= \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}] + \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T})}{p(z_{1:T}|x)}]$$

Markovian Hierarchical VAEs - same derivation

$$
\begin{aligned}
\log p(x) &= \mathbb{E}_{z_{1:T} \sim q_\phi(z_{1:T}|x)}[\log p_\theta(x^{(i)})] && p_\theta(x) \text{ is independent of } z_{1:T} \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{p_\theta(z_{1:T}|x)}] && \text{Bayes rule} \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{p_\theta(z_{1:T}|x)} \frac{q_\phi(z_{1:T}|x)}{q_\phi(z_{1:T}|x)}] && \text{Multiplying by a constant} \\
&= \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}] + \mathbb{E}_{z_{1:T}}[\log \cancel{\frac{q_\phi(z_{1:T})}{p(z_{1:T}|x)}}]
\end{aligned}
$$

↑ Reconstruction objective maximizes the likelihood of data $p_\theta(x|z)$

↑ This KL term (between Gaussians for encoder and z prior)

↑ $p_\theta(z|x)$ intractable but we know KL divergence always >= 0.

Markovian Hierarchical VAEs

Keeping just the first two terms:

$$\log p(x) \geq = \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}]$$

Markovian Hierarchical VAEs

Keeping just the first two terms:

$$
\begin{aligned}
\log p(x) \geq &= \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}] \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{q_\phi(z_{1:T}|x)}] \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x, z_{1:T})}{q_\phi(z_{1:T}|x)}]
\end{aligned}
$$

Markovian Hierarchical VAEs

Keeping just the first two terms:

$$
\begin{aligned}
\log p(x) \geq &= \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}] \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{q_\phi(z_{1:T}|x)}] \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x, z_{1:T})}{q_\phi(z_{1:T}|x)}]
\end{aligned}
$$

where the joint probability distribution is: $p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)$

This is very similar to the autoregressive model formula

Markovian Hierarchical VAEs

Keeping just the first two terms:

$$\begin{aligned}\log p(x) \geq &= \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}] \\ &= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{q_\phi(z_{1:T}|x)}] \\ &= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x, z_{1:T})}{q_\phi(z_{1:T}|x)}]\end{aligned}$$

where the joint probability distribution is: $p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)$

And the encoder posterior is: $q_\phi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}) \prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$

Markovian Hierarchical VAEs

Keeping just the first two terms:

$$
\begin{aligned}
\log p(x) \geq &= \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}] \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{q_\phi(z_{1:T}|x)}] \\
&= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x, z_{1:T})}{q_\phi(z_{1:T}|x)}]
\end{aligned}
$$

Why is this a hard objective to train?

where the joint probability distribution is: $p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)$

And the encoder posterior is: $q_\phi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x})\prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$

Markovian Hierarchical VAEs

Keeping just the first two terms:

$$\log p(x) \geq = \mathbb{E}_{z_{1:T}}[\log p_\theta(x|z_{1:T})] - \mathbb{E}_{z_{1:T}}[\log \frac{q_\phi(z_{1:T}|x)}{p_\theta(z_{1:T})}]$$

$$= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x|z_{1:T})p_\theta(z_{1:T})}{q_\phi(z_{1:T}|x)}]$$

$$= \mathbb{E}_{z_{1:T}}[\log \frac{p_\theta(x, z_{1:T})}{q_\phi(z_{1:T}|x)}]$$

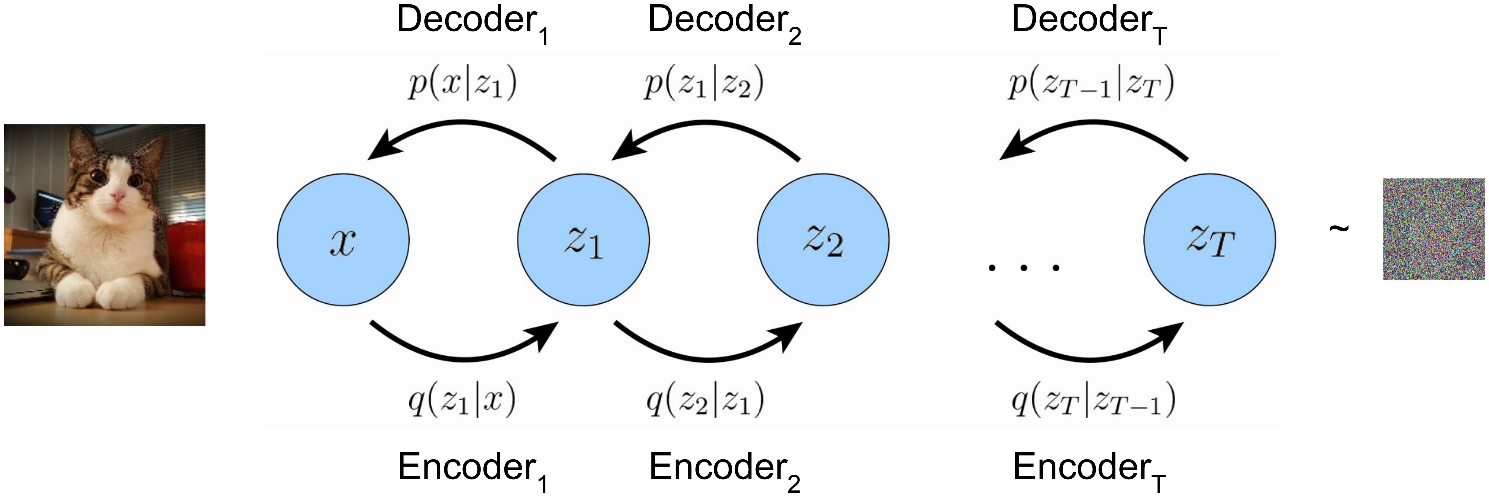
Why is this a hard objective to train?

1. There are too many networks to learn
2. The objective function is expensive!
3. It collapses easily!

where the joint probability distribution is: $p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)$

And the encoder posterior is: $q_\phi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x})\prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$

Markovian Hierarchical VAEs

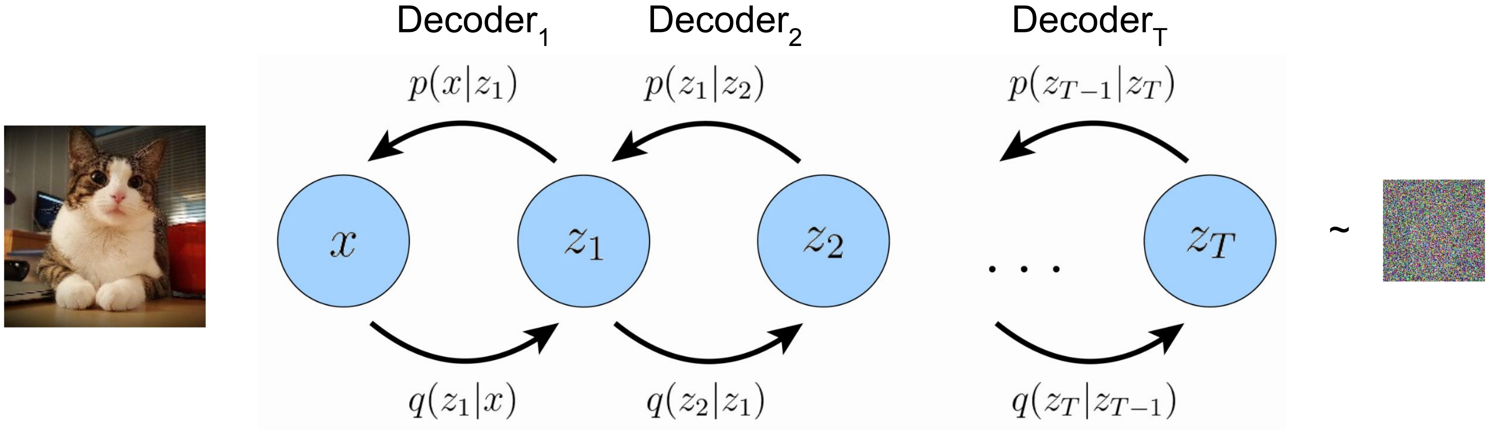


Decoder$_1$ Decoder$_2$ Decoder$_T$

Encoder$_1$ Encoder$_2$ Encoder$_T$

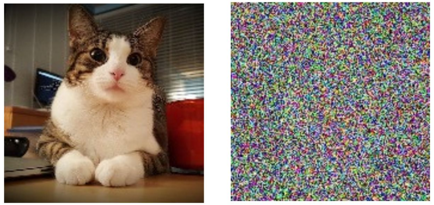
Diffusion models are a special case
With a more interpretable, simpler objective.

Markovian Hierarchical VAEs



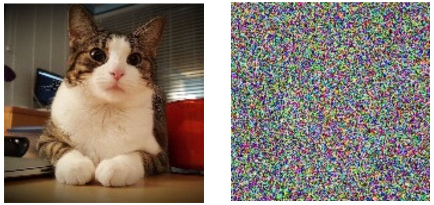
How do we do this? Get rid of the encoders! We're just transitioning to noise, how hard can it be?

How do we do this??

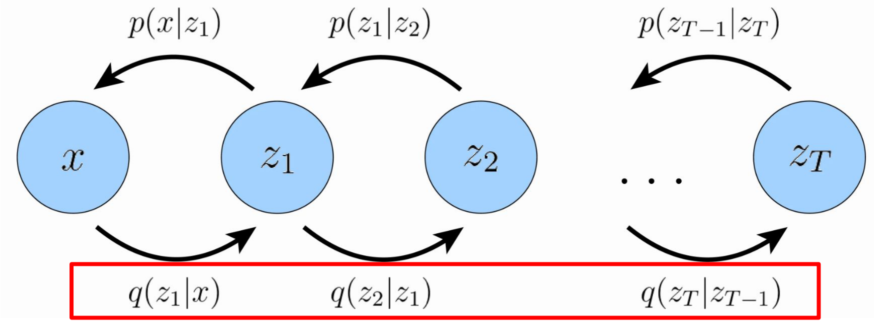


1. The latent dimension size is exactly equal to the data dimension

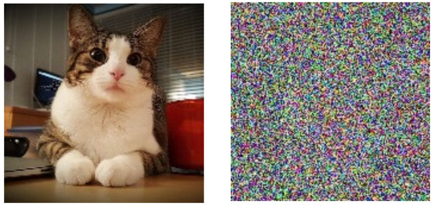
How are diffusion models different?



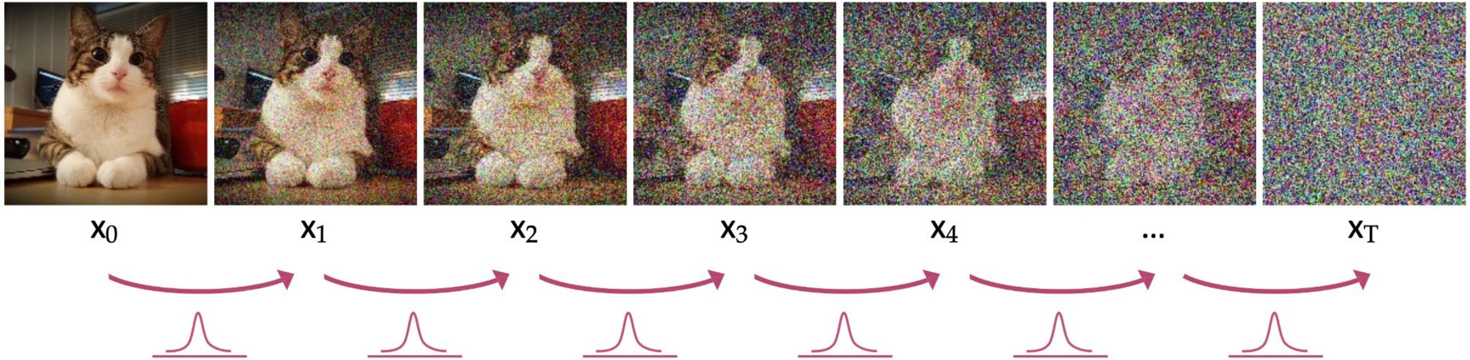
1. The latent dimension size is exactly equal to the data dimension
2. The encoders are pre-defined and not learned.



How are diffusion models different?

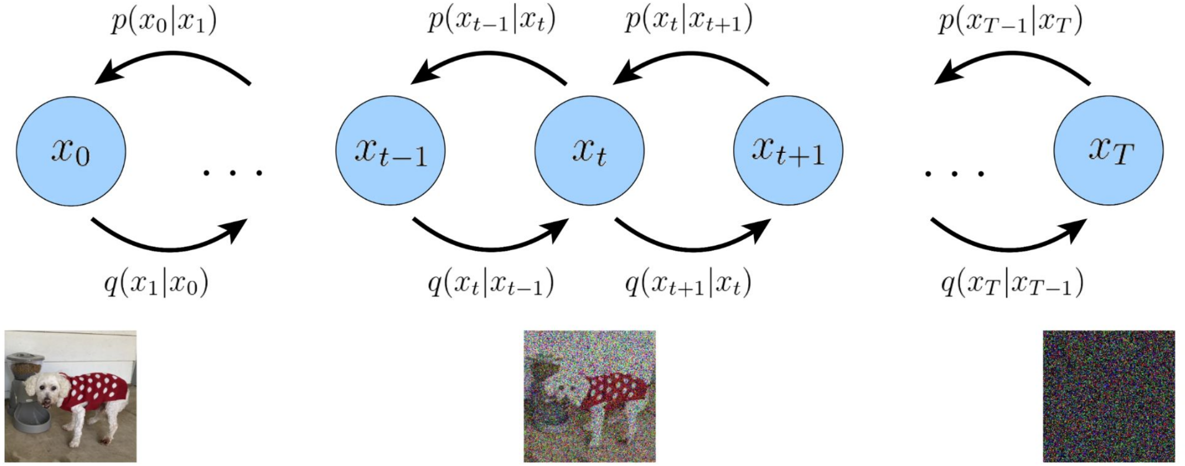


1. The latent dimension size is exactly equal to the data dimension
2. The encoders are pre-defined and not learned.
3. Encoders are designed as a linear Gaussian model conditioned on the time step: Add noise at every time step

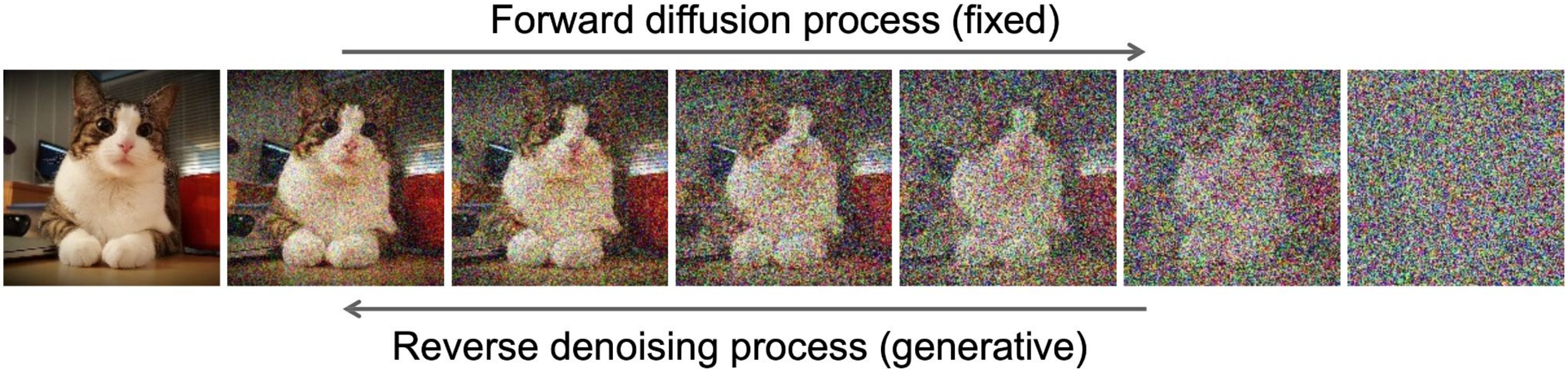


How are diffusion models different?

4. The Gaussian parameters vary over time in such a way that the distribution of the latent at final step T is a standard Gaussian



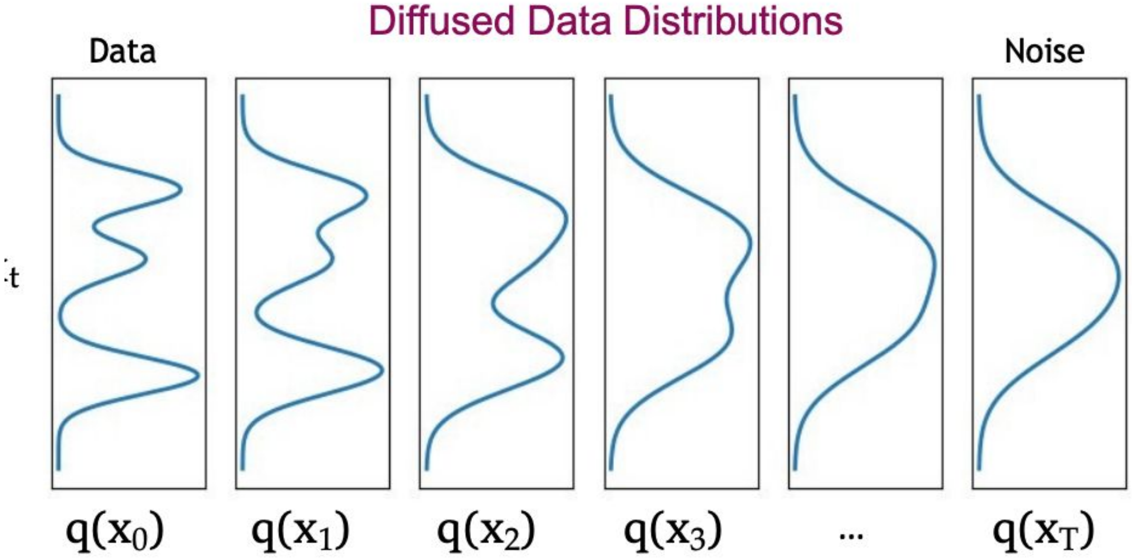
Terminology: Forward and backward process



Note: reverse or backward here doesn't mean the same thing as backpropagation

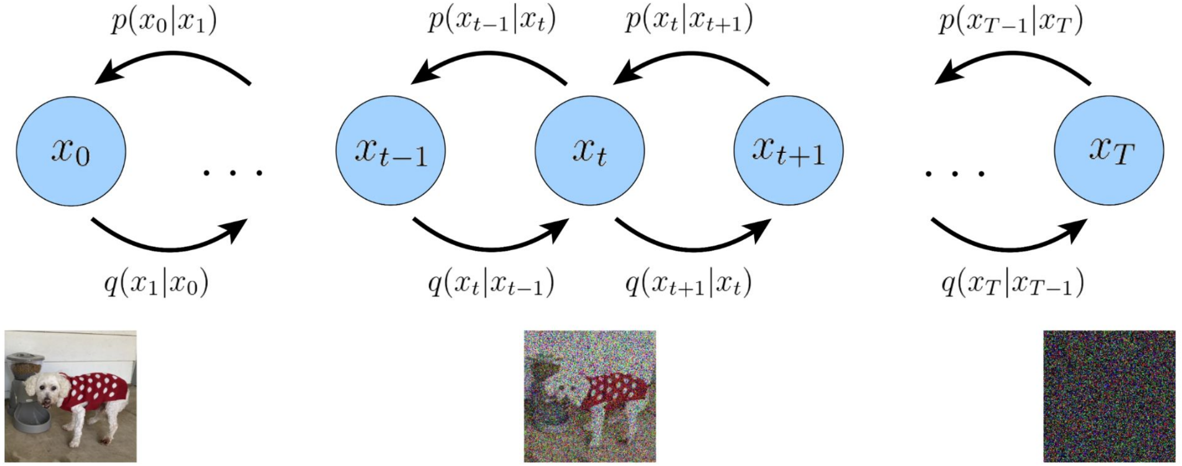
The distribution perspective

Over time, as we add more noise sampled from a Gaussian distribution, it begins to look more like a unit normal



How do we define a loss objective?

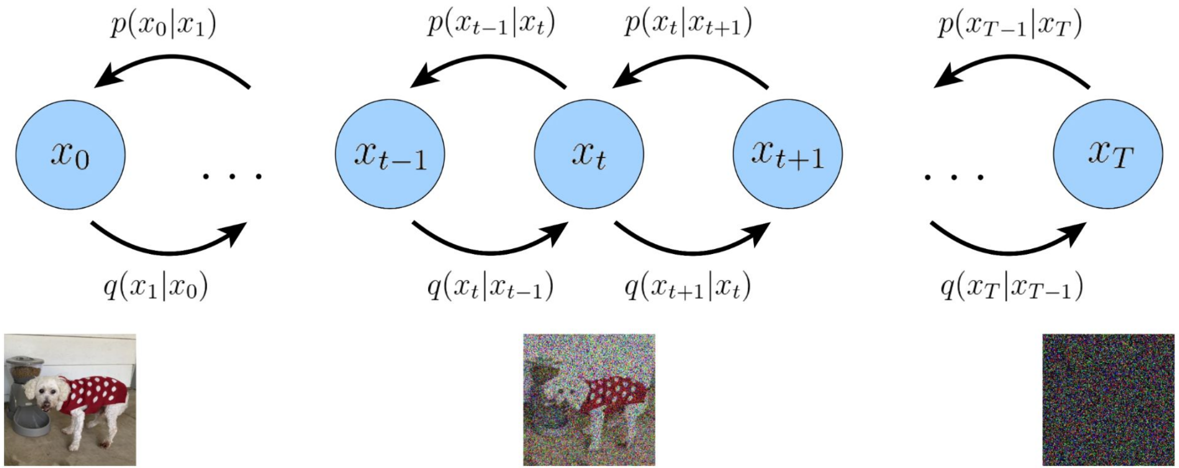
Q. What do we have to learn to generate new samples from noise?



How do we define a loss objective?

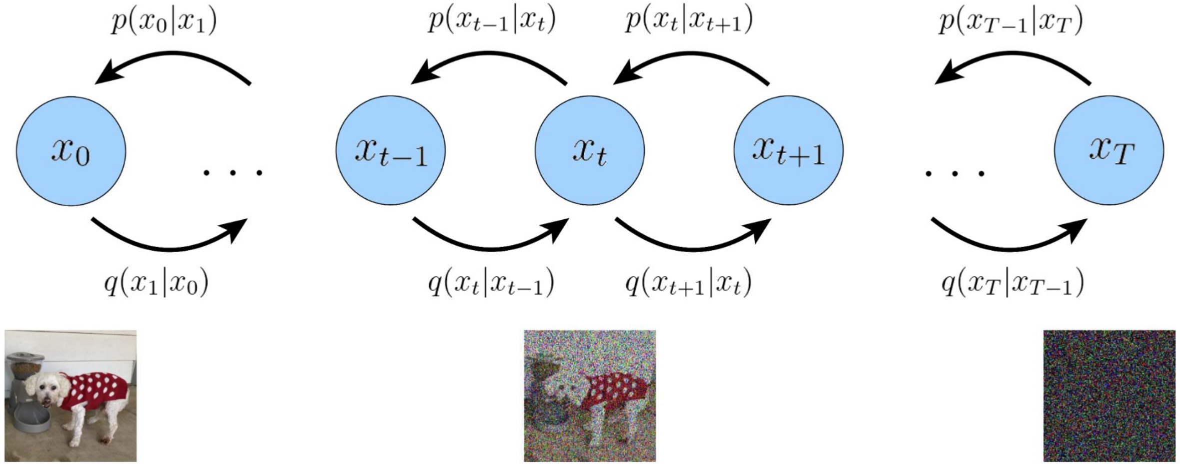
Q. What do we have to learn to generate new samples from noise?

A. We want to define a neural network to predict $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$



How do we define a loss objective?

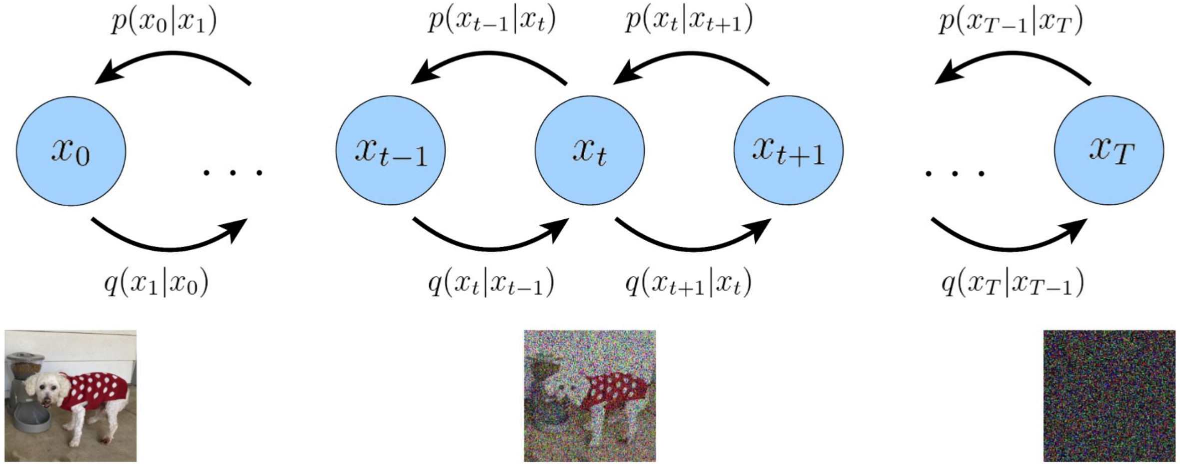
Q. How should we train $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$?



How do we define a loss objective?

Q. How should we train $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$?

A. We can get it to match $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$!



Ok so our loss function is:

Q. How should we train $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$?

A. We can get it to match $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$!

Minimize the distance between the two distributions:

$$\underset{\boldsymbol{\theta}}{\arg\min}\, \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))$$

Ok so our loss function is:

Q. How should we train $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$?

A. We can get it to match $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$!

Minimize the distance between the two distributions:

$$\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\text{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))$$

Problem: How do we estimate $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$?

The forward diffusion step

The distribution at step t is a Gaussian

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

The forward diffusion step

The distribution at step t is a Gaussian

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1 - \alpha_t)\mathbf{I})$$

The mean defined by x_{t-1}: $\boldsymbol{\mu}_t(\boldsymbol{x}_t) = \sqrt{\alpha_t}\boldsymbol{x}_{t-1}$

$\boldsymbol{\alpha}_t$ is a predefined value for each step t

The forward diffusion step

The distribution at step t is a Gaussian

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

The mean defined by x_{t-1}: $\boldsymbol{\mu}_t(\boldsymbol{x}_t) = \sqrt{\alpha_t}\boldsymbol{x}_{t-1}$

$\boldsymbol{\alpha}_t$ is a predefined value for each step t

The covariance is independent of x_{t-1} (an assumption)

$$\boldsymbol{\Sigma}_t(\boldsymbol{x}_t) = (1-\alpha_t)\mathbf{I}$$

How the forward step was designed:

The distribution at step t is a Gaussian

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

$$\boldsymbol{\mu}_t(\boldsymbol{x}_t) = \sqrt{\alpha_t}\boldsymbol{x}_{t-1}$$

$$\boldsymbol{\Sigma}_t(\boldsymbol{x}_t) = (1-\alpha_t)\mathbf{I}$$

So, given x_{t-1} we can sample x_t using:

$$x_t \sim \sqrt{\alpha_t}x_{t-1} + (1-\alpha_t)\epsilon$$

where $\epsilon \sim \mathcal{N}(x; 0, I)$

Why was it designed like this?

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

Why was it designed like this?

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \quad \longleftarrow \text{Substituting } x_{t-1}
\end{aligned}
$$

Why was it designed like this?

$$\begin{aligned}\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\ &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}\end{aligned}$$

← Substituting x_{t-1}

← Opening the parentheses

Why was it designed like this?

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \quad \longleftarrow \text{Substituting } x_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \boxed{\sqrt{\alpha_t-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}} \quad \longleftarrow \text{Opening the parentheses}
\end{aligned}
$$

We can interpret this $\sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ as a sample from $\mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I})$

We can interpret this $\sqrt{\alpha_t-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}$ as a sample from $\mathcal{N}(\mathbf{0}, (\alpha_t-\alpha_t\alpha_{t-1})\mathbf{I})$

Why was it designed like this?

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \quad \longleftarrow \text{Substituting } x_{t\text{-}1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \boxed{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}} \quad \longleftarrow \text{Opening the parentheses}
\end{aligned}
$$

We can interpret this $\sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ as a sample from $\mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I})$

We can interpret this $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}$ as a sample from $\mathcal{N}(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I})$

Notice that $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ is the sum of two Gaussian samples

Why was it designed like this?

$$\begin{aligned}\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\ &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \boxed{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}}\end{aligned}$$

← Substituting x_{t-1}

← Opening the parentheses

We can interpret this $\sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ as a sample from $\mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I})$

We can interpret this $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}$ as a sample from $\mathcal{N}(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I})$

Notice that $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ is the sum of two Gaussian samples

Using the property: $\mathcal{N}(x; 0, \sigma_1 I) + \mathcal{N}(x; 0, \sigma_2 I) = \mathcal{N}(x; 0, \sqrt{\sigma_1^2 + \sigma_2^2} I)$

Why was it designed like this?

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Substituting } x_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \boxed{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}} && \longleftarrow \text{Opening the parentheses}
\end{aligned}$$

We can interpret this $\sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ as a sample from $\mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I})$

We can interpret this $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}$ as a sample from $\mathcal{N}(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I})$

Notice that $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ is the sum of two Gaussian samples

Using the property: $\mathcal{N}(x; 0, \sigma_1 I) + \mathcal{N}(x; 0, \sigma_2 I) = \mathcal{N}(x; 0, \sqrt{\sigma_1^2 + \sigma_2^2} I)$

We can rewrite $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$ as $\sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2}$

Why was it designed like this?

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

$$= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \longleftarrow$$

Substituting x_{t-1}

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \longleftarrow$$

Opening the parentheses

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2} \longleftarrow$$

Sum of two Gaussians

Why was it designed like this?

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

$$= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

Substituting x_{t-1}

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

Opening the parentheses

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2}$$

Sum of two Gaussians

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\underline{\alpha_t - \alpha_t\alpha_{t-1}} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2}$$

Squaring the terms

Why was it designed like this?

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Substituting } x_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Opening the parentheses} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Sum of two Gaussians} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Squaring the terms} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Simplifying}
\end{aligned}
$$

Why was it designed like this?

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Substituting } x_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Opening the parentheses} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Sum of two Gaussians} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Squaring the terms} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Simplifying} \\
&= \ldots \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\boldsymbol{x}_0 + \sqrt{1 - \prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon}_0 && \longleftarrow \text{Substituting till } x_0
\end{aligned}
$$

Why was it designed like this?

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Substituting } x_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1} && \longleftarrow \text{Opening the parentheses} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Sum of two Gaussians} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Squaring the terms} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} && \longleftarrow \text{Simplifying} \\
&= \ldots \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\boldsymbol{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon}_0 && \longleftarrow \text{Substituting till } x_0 \\
&= \sqrt{\bar\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\bar\alpha_t}\boldsymbol{\epsilon}_0 && \longleftarrow \text{Let } \bar\alpha_t = \prod_{i=1}^{t}\alpha_i
\end{aligned}
$$

Why was it designed like this?

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

$$= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

Substituting x_{t-1}

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^*_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^*_{t-1}$$

Opening the parentheses

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2}$$

Sum of two Gaussians

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2}$$

Squaring the terms

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}$$

Simplifying

$$= \ldots$$

$$= \sqrt{\prod_{i=1}^{t}}\alpha_i\boldsymbol{x}_0 + \sqrt{1 - \prod_{i=1}^{t}}\alpha_i\boldsymbol{\epsilon}_0$$

Substituting till x_0

$$= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0$$

Let $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$

$$\sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

x_t is now a Gaussian characterized by x_0

Takeaway from the previous slides:

$$\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$$

We can instantly sample x_t given any input data x_0

What about the reverse?

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$$

What about the reverse?

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}$$

Applying Bayes rule

What about the reverse?

$$
\begin{aligned}
q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{\boxed{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\, q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)} \\
&= \frac{\boxed{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})}}{}
\end{aligned}
$$

The first term is just a single forward diffusion process:

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

What about the reverse?

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0)\boxed{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}$$

$$= \frac{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})\boxed{\mathcal{N}(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}}{}$$

The second term is also a Gaussian using the formula we just derived:

$$\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

What about the reverse?

$$
\begin{aligned}
q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{\boxed{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}} \\
&= \frac{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\boxed{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})}}
\end{aligned}
$$

The third term is also a Gaussian using the same formula:

$$\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

What about the reverse?

$$
\begin{aligned}
q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)} \\
&= \frac{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})} \\
&\propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})
\end{aligned}
$$

The product of these 3 Gaussian distributions simplify to a Gaussian as well!

Let's call its mean $\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ and variance $\boldsymbol{\Sigma}_q(t)$

Proof (out of scope for the class)

$$
\begin{aligned}
q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0)\, q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)} \\
&= \frac{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})\,\mathcal{N}(\boldsymbol{x}_{t-1}; \sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_0, (1-\bar\alpha_{t-1})\mathbf{I})}{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar\alpha_t}\boldsymbol{x}_0, (1-\bar\alpha_t)\mathbf{I})} \\
&\propto \exp\left\{-\left[\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{2(1-\alpha_t)} + \frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_0)^2}{2(1-\bar\alpha_{t-1})} - \frac{(\boldsymbol{x}_t - \sqrt{\bar\alpha_t}\boldsymbol{x}_0)^2}{2(1-\bar\alpha_t)}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{1-\alpha_t} + \frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_0)^2}{1-\bar\alpha_{t-1}} - \frac{(\boldsymbol{x}_t - \sqrt{\bar\alpha_t}\boldsymbol{x}_0)^2}{1-\bar\alpha_t}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{(-2\sqrt{\alpha_t}\boldsymbol{x}_t\boldsymbol{x}_{t-1} + \alpha_t\boldsymbol{x}_{t-1}^2)}{1-\alpha_t} + \frac{(\boldsymbol{x}_{t-1}^2 - 2\sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_{t-1}\boldsymbol{x}_0)}{1-\bar\alpha_{t-1}} + C(\boldsymbol{x}_t, \boldsymbol{x}_0)\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[-\frac{2\sqrt{\alpha_t}\boldsymbol{x}_t\boldsymbol{x}_{t-1}}{1-\alpha_t} + \frac{\alpha_t\boldsymbol{x}_{t-1}^2}{1-\alpha_t} + \frac{\boldsymbol{x}_{t-1}^2}{1-\bar\alpha_{t-1}} - \frac{2\sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_{t-1}\boldsymbol{x}_0}{1-\bar\alpha_{t-1}}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\left(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\bar\alpha_{t-1}}\right)\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_0}{1-\bar\alpha_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{\alpha_t(1-\bar\alpha_{t-1}) + 1 - \alpha_t}{(1-\alpha_t)(1-\bar\alpha_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar\alpha_{t-1}}\boldsymbol{x}_0}{1-\bar\alpha_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\}
\end{aligned}
$$

Proof (out of scope for the class)

$$= \exp\left\{-\frac{1}{2}\left[\frac{\alpha_t - \bar{\alpha}_t + 1 - \alpha_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\}$$

$$= \exp\left\{-\frac{1}{2}\left[\frac{1 - \bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\}$$

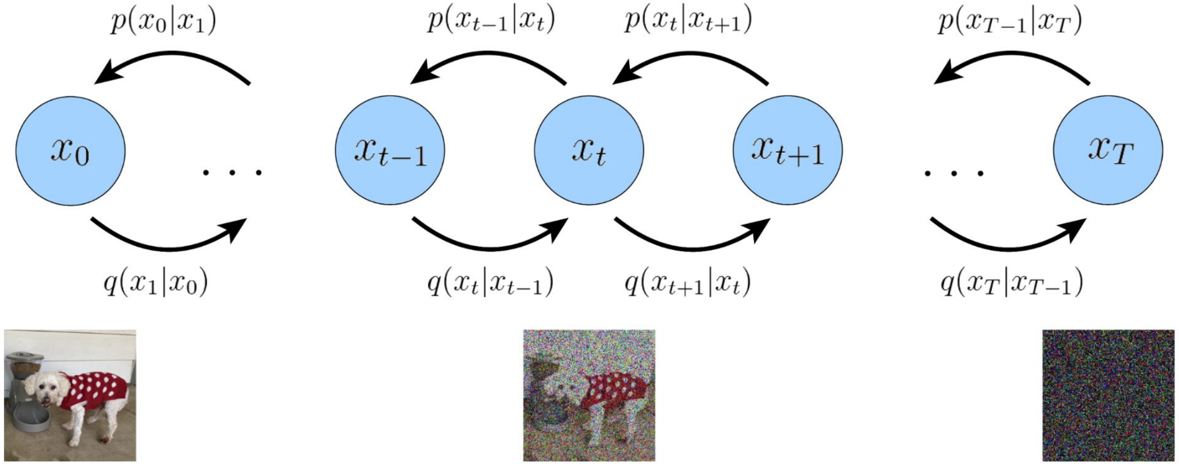
$$= \exp\left\{-\frac{1}{2}\left(\frac{1 - \bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)}{\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}}\boldsymbol{x}_{t-1}\right]\right\}$$

$$= \exp\left\{-\frac{1}{2}\left(\frac{1 - \bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right]\right\}$$

$$= \exp\left\{-\frac{1}{2}\left(\frac{1}{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right]\right\}$$

$$\propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

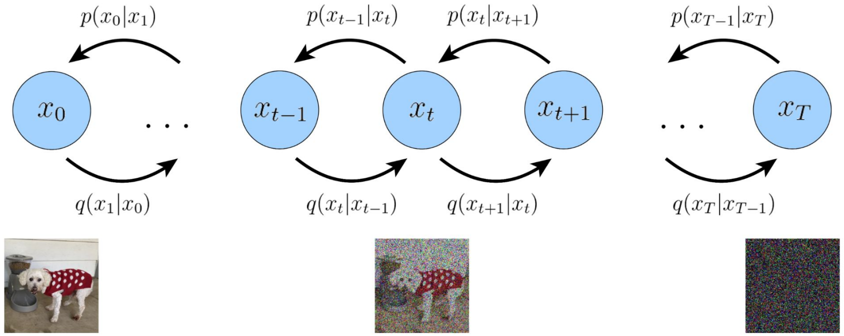
Let’s go back to the Markovian VAE



We are ready to set up a simple intuitive loss function to train the decoder!
Given an image x_0:

We want to generate $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ to match the Gaussian we just derived: $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$

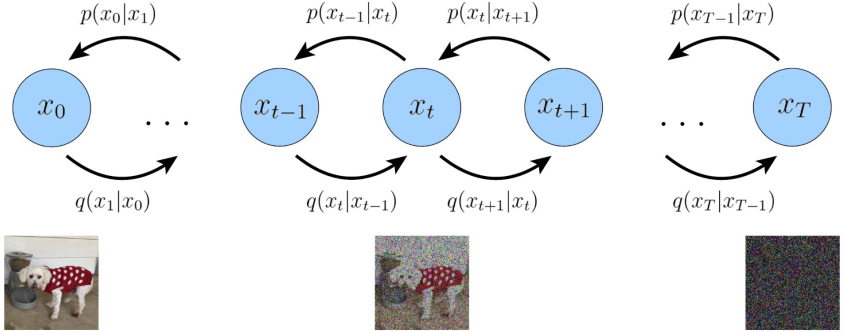
The loss function tries to match distributions



The loss function

$$\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))$$

We can model $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ as a Gaussian

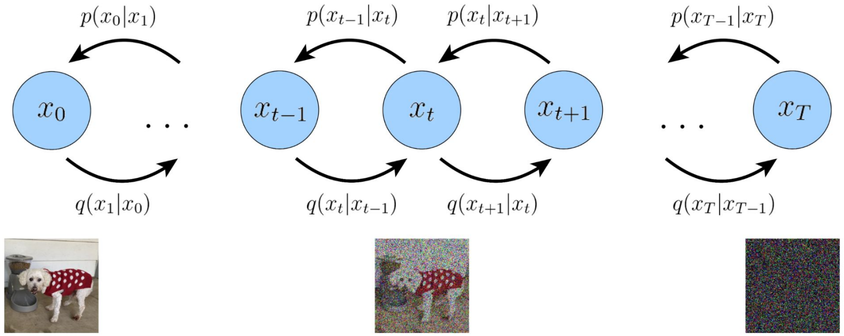


The loss function

$$\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\text{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\text{KL}} \left(\mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q\left(t\right)\right) \mid\mid \mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q\left(t\right)\right)\right)$$

We can model $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ as a Gaussian



The loss function

$$\begin{aligned}
&\underset{\boldsymbol{\theta}}{\arg\min}\, \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) \\
&= \underset{\boldsymbol{\theta}}{\arg\min}\, \mathcal{D}_{\mathrm{KL}} \left(\mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q\left(t\right)\right) \,||\, \mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q\left(t\right)\right)\right) \\
&= \underset{\boldsymbol{\theta}}{\arg\min}\, \frac{1}{2\sigma_q^2(t)} \left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right]
\end{aligned}$$

Proof (out of scope for class)

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) \\
=&\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q(t)) \mid\mid \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q(t))) \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_q(t)|}{|\boldsymbol{\Sigma}_q(t)|} - d + \mathrm{tr}(\boldsymbol{\Sigma}_q(t)^{-1}\boldsymbol{\Sigma}_q(t)) + (\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[\log 1 - d + d + (\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)^T\left(\sigma_q^2(t)\mathbf{I}\right)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\mid\mid \boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q \mid\mid_2^2\right]
\end{aligned}$$

Ok we are close to the objective:

The loss we want to minimize is $\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\left\| \boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q \right\|_2^2 \right]$

Ok we are close to the objective:

The loss we want to minimize is $\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2\right]$

From the previous slide, we got the mean from this:

$$\mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

Ok we are close to the objective:

The loss we want to minimize is $\underset{\boldsymbol{\theta}}{\arg\min}\, \frac{1}{2\sigma_q^2(t)}\left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right]$

From the previous slide, we got the mean from this:

$$\mathcal{N}(\boldsymbol{x}_{t-1}; \boxed{\underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

So, we can write the mean to be: $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$

Ok we are close to the objective:

The loss we want to minimize is $\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right]$

From the previous slide, we got the mean from this:

$$\mathcal{N}(\boldsymbol{x}_{t-1}; \boxed{\underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)}}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

So, we can write the mean to be:

$$\begin{aligned}\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \\ &= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0\end{aligned}$$

Our neural network can predict noise instead!

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0$$

We can also set our predicted mean to be:

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t)$$

Why is this helpful?

Our neural network can predict noise instead!

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0$$

We can also set our predicted mean to be:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Why is this helpful? Because now our model needs to predict the noise that was injected, which turns out to be empirically more stable of an objective than predicting the image mean.

The two loss objectives are equivalent

The loss function

$$\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \boxed{\left[\left\| \boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q \right\|_2^2 \right]}$$

Instead of predicting the mean image values

The two loss objectives are equivalent

The loss function

$$\arg\min_{\boldsymbol{\theta}} \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \boxed{\left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right]}$$

Instead of predicting the mean image values

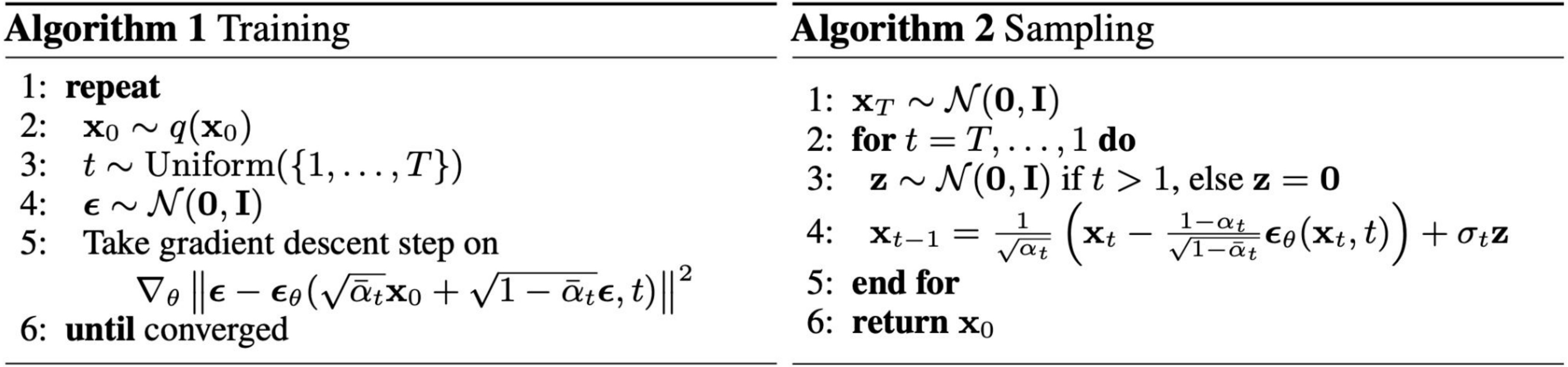
$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \boxed{\left[\left\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right\|_2^2\right]}$$

The neural network can predict the added noise

Proof: (out of scope)

$$
\begin{aligned}
&\underset{\boldsymbol{\theta}}{\arg\min}\, \mathcal{D}_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) \\
=&\underset{\boldsymbol{\theta}}{\arg\min}\, \mathcal{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q(t)) \mid\mid \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q(t))) \\
=&\underset{\boldsymbol{\theta}}{\arg\min}\, \frac{1}{2\sigma_q^2(t)} \left[\left\| \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 \right\|_2^2 \right] \\
=&\underset{\boldsymbol{\theta}}{\arg\min}\, \frac{1}{2\sigma_q^2(t)} \left[\left\| \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right\|_2^2 \right] \\
=&\underset{\boldsymbol{\theta}}{\arg\min}\, \frac{1}{2\sigma_q^2(t)} \left[\left\| \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}(\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)) \right\|_2^2 \right] \\
=&\underset{\boldsymbol{\theta}}{\arg\min}\, \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\mid\mid \boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \mid\mid_2^2 \right]
\end{aligned}
$$

The final algorithm (DDPM)



Algorithm 1 Training

```
repeat
    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
    $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Take gradient descent step on
        $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
until converged
```

Algorithm 2 Sampling

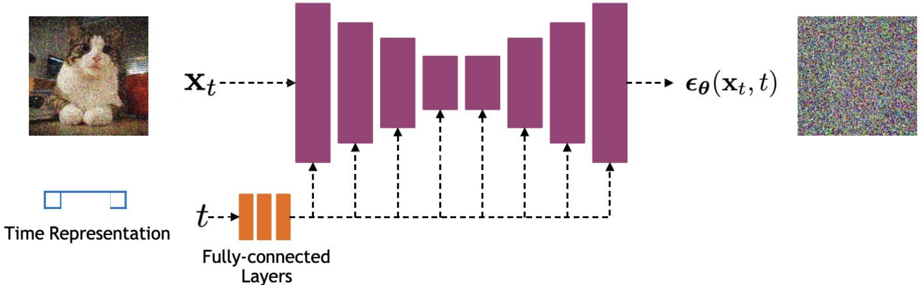
```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ do
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$
```

How is the time step input:

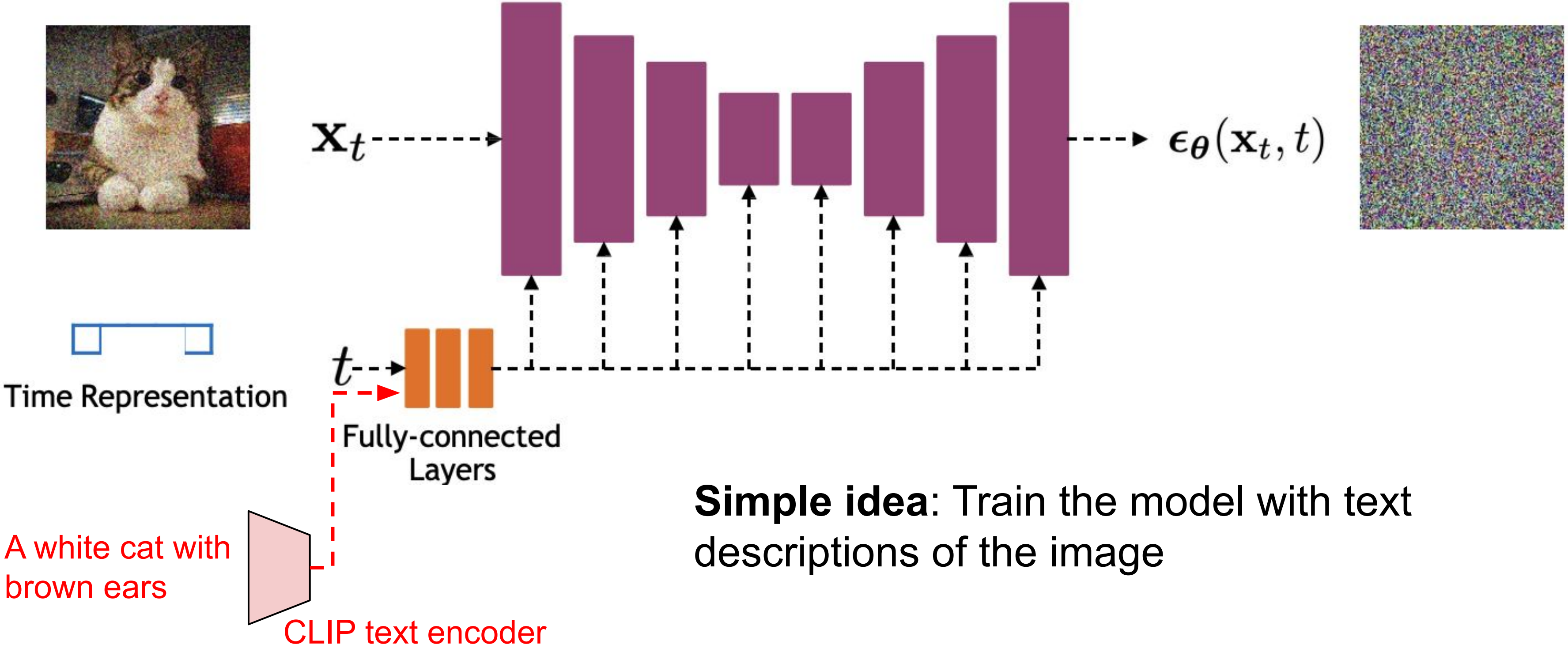
Time representation: sinusoidal positional embeddings.

Added in using: $\mathrm{AdaGN}(h, y) = y_s\,\mathrm{GroupNorm}(h) + y_b$

- *h* is the intermediate activations of the residual block following the first convolution in each layer,
- $y = [y_s, y_b]$ is obtained from a linear projection of the timestep

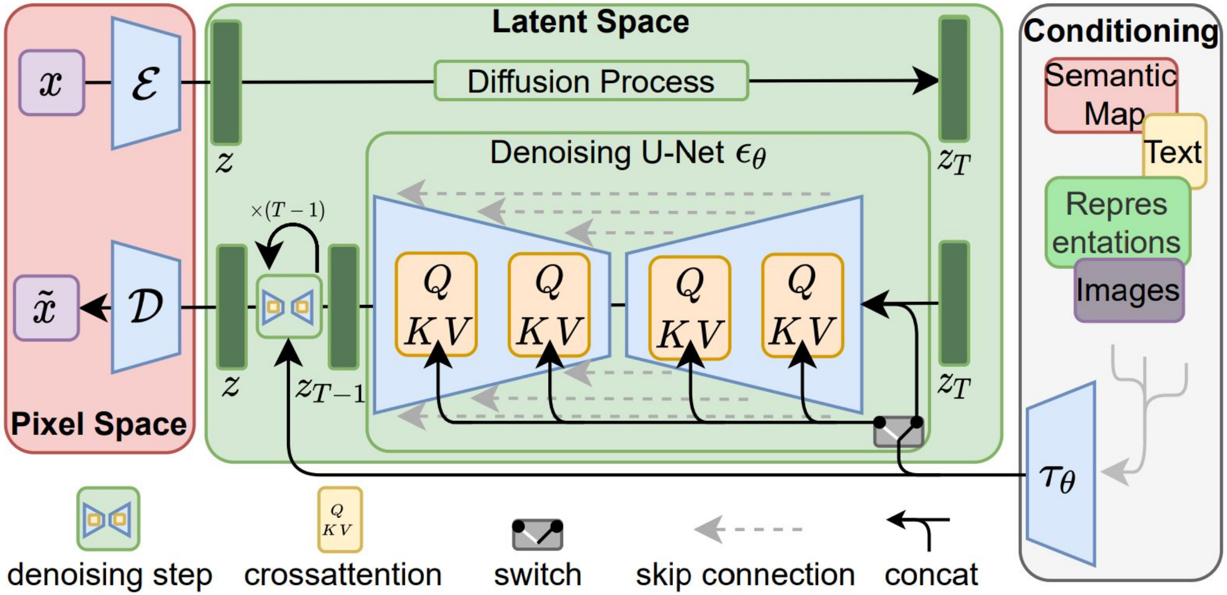


Text-conditioned generation

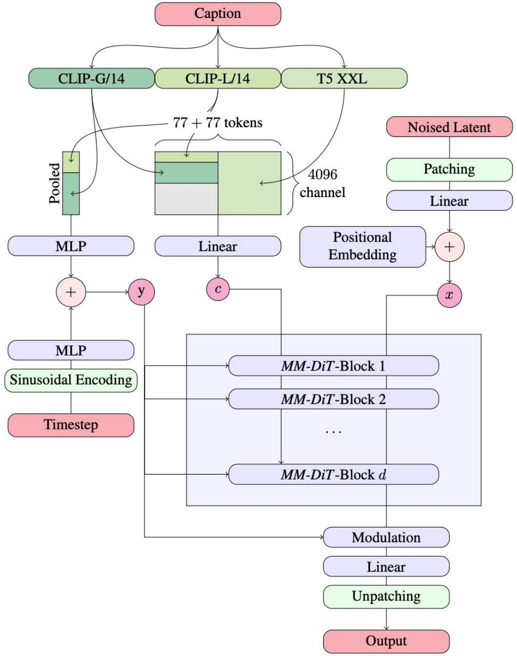


Simple idea: Train the model with text descriptions of the image

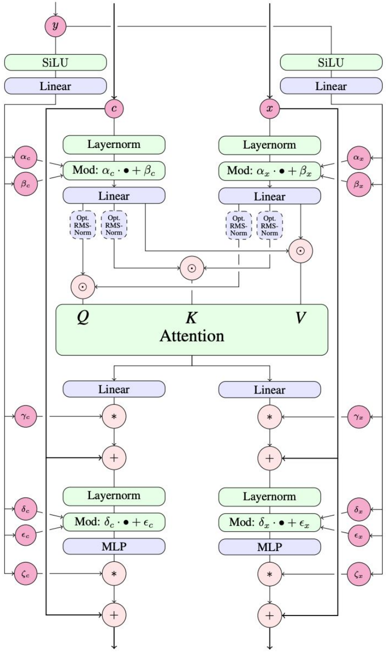
In practice, a bit more complicated



In practice, a bit more complicated

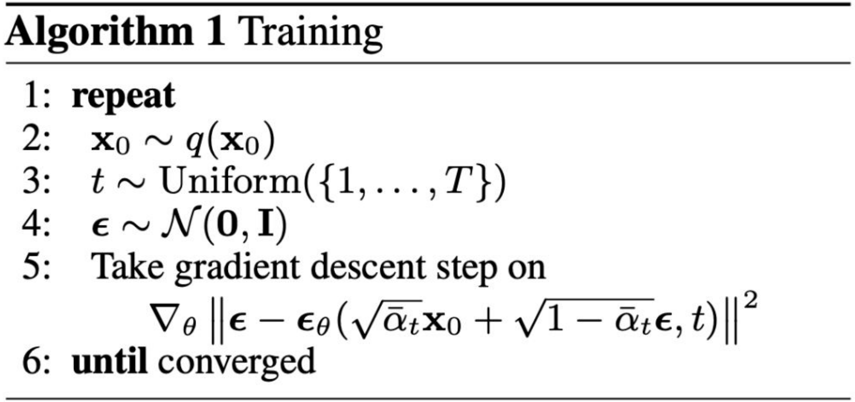


(a) Overview of all components.



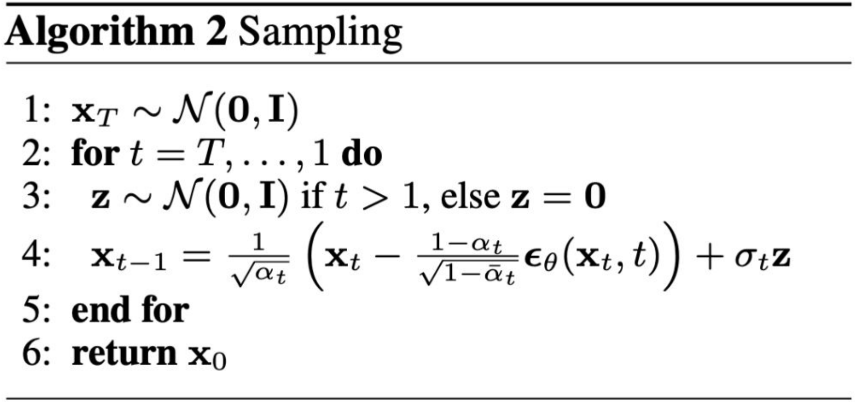
(b) One *MM-DiT* block

A simplified diffusion algorithm (DDIM)



Algorithm 1 Training

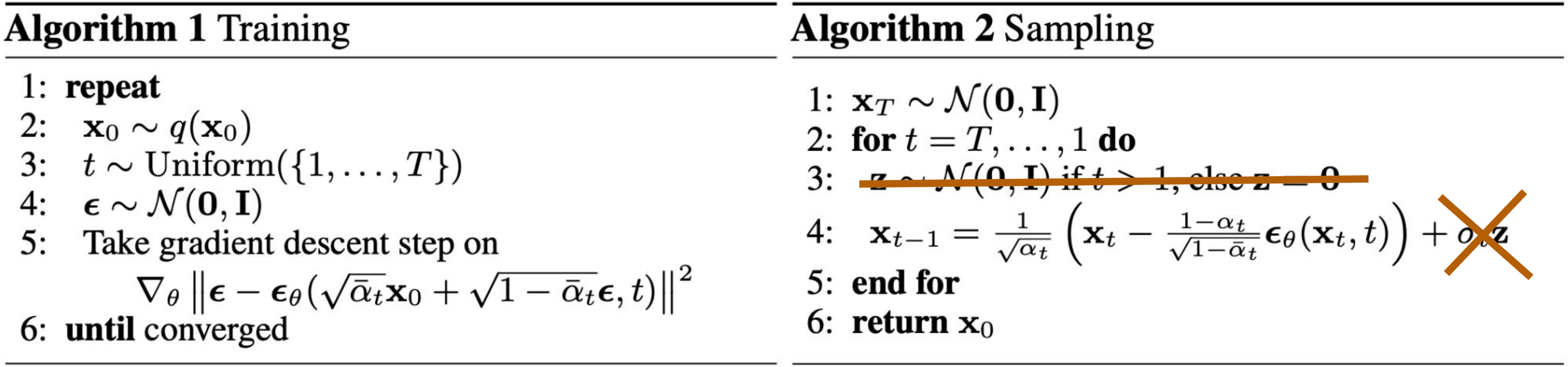
```
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
      $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar\alpha_t}\mathbf{x}_0 + \sqrt{1-\bar\alpha_t}\boldsymbol{\epsilon}, t) \right\|^2$
until converged
```



Algorithm 2 Sampling

```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ do
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$
```

A simplified diffusion algorithm (DDIM)



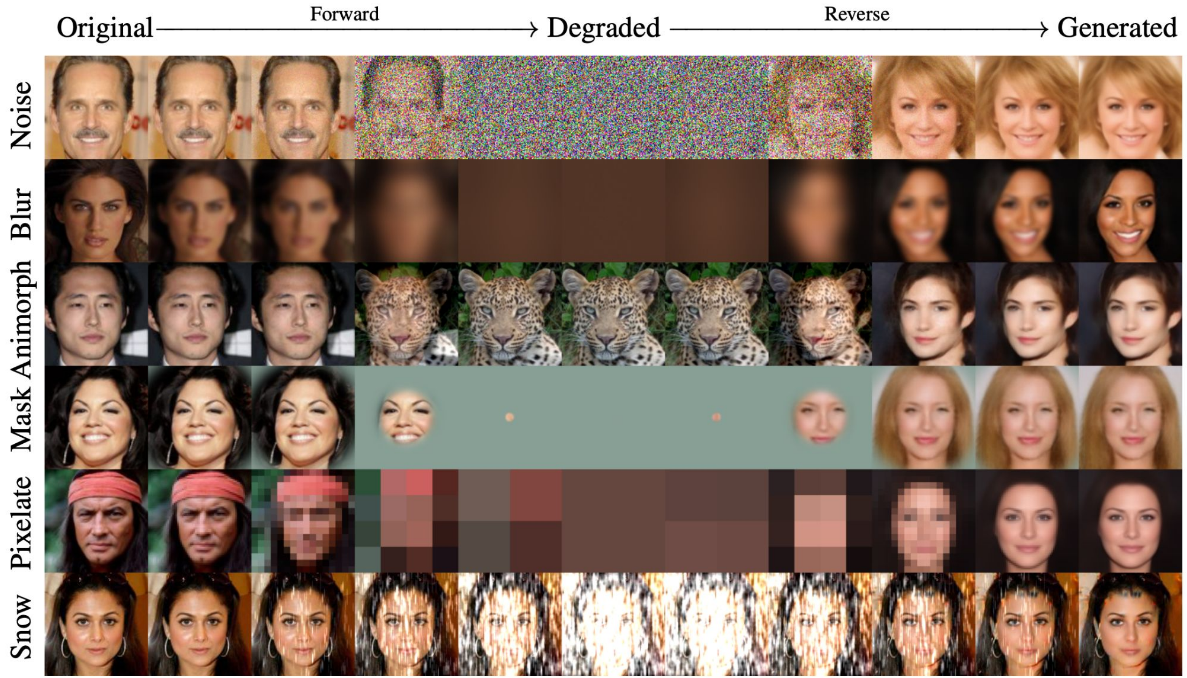
Algorithm 1 Training

```
repeat
    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
    $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Take gradient descent step on
        $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
until converged
```

Algorithm 2 Sampling

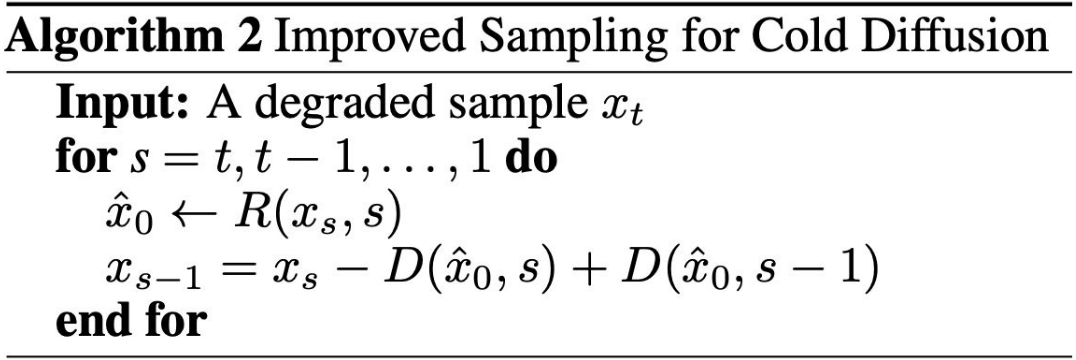
```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ do
    ~~$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$~~
    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \cancel{\sigma_t \mathbf{z}}$
end for
return $\mathbf{x}_0$
```

Did we really need any of this? (cold diffusion)



Did we really need any of this? (cold diffusion)

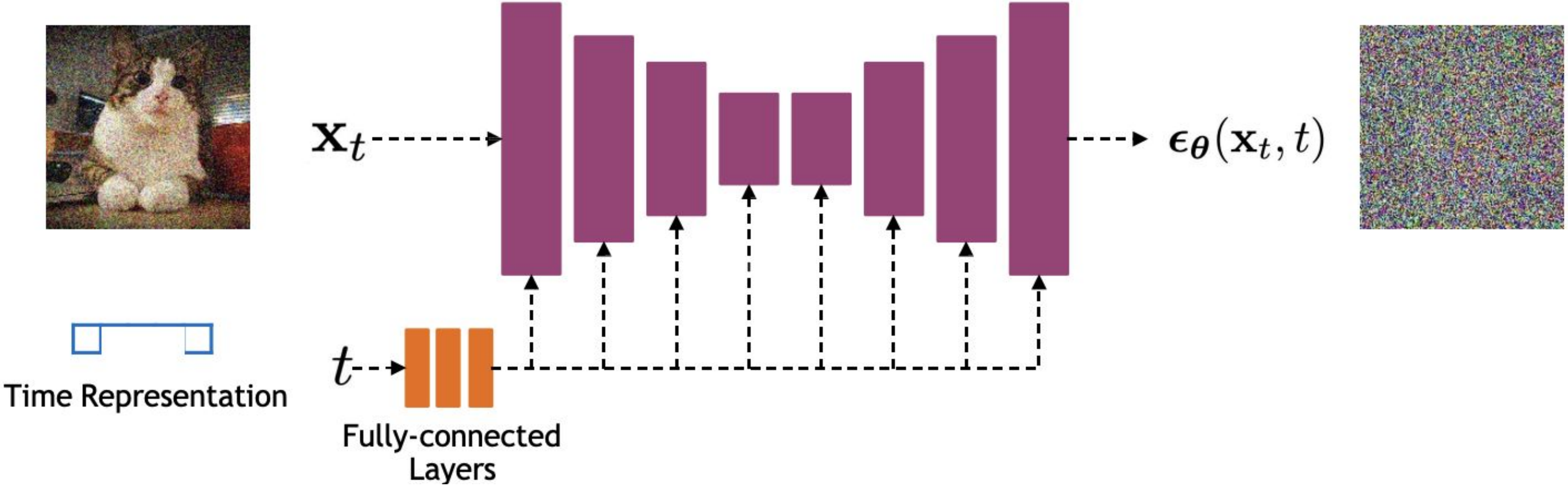
- Two components “restoration” network R, and a “degradation operator D.



Algorithm 2 Improved Sampling for Cold Diffusion

```
Input: A degraded sample $x_t$
for $s = t, t-1, \dots, 1$ do
    $\hat{x}_0 \leftarrow R(x_s, s)$
    $x_{s-1} = x_s - D(\hat{x}_0, s) + D(\hat{x}_0, s-1)$
end for
```

The denoising architecture



Time representation: sinusoidal positional embeddings.

How do we sample a new image?

```
Sample $x_T \sim \mathcal{N}(0, I)$
For $t = T \ldots 1$ do
    Predict $\hat{\epsilon}_t = p_\theta(x_t)$
    $\mu_{t-1} = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}\sqrt{\alpha_t}} \sqrt{1 - \bar{\alpha}_{t-1}} \hat{\epsilon}_t$
    Sample $x_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$
Return $x_0$
```

Application of diffusion: Image Super-resolution

Irish Setter

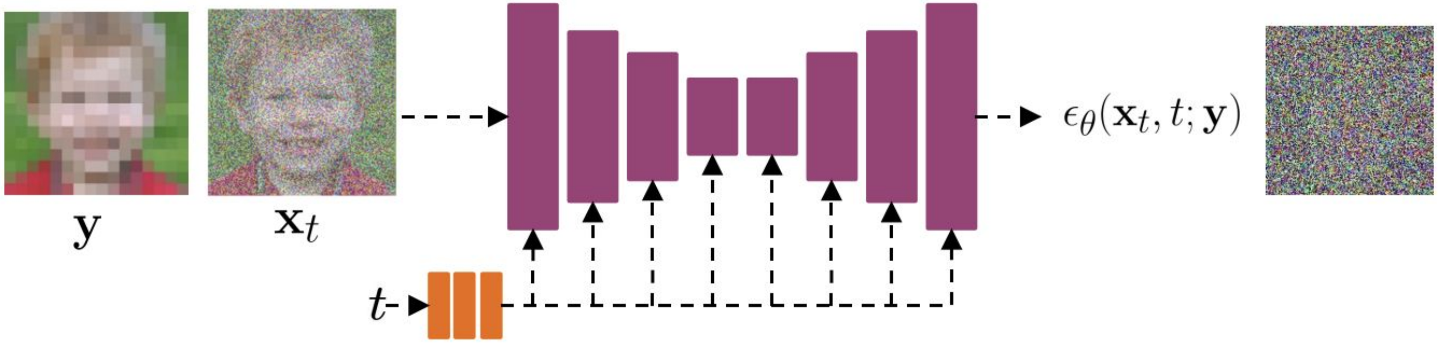
Saharia et al., Image Super-Resolution via Iterative Refinement, ICCV 2021

Gif on this slide is not displayed in pdf

Application: super-resolution

Learn a superresolution diffusion model conditioned on a low resolution image.
y is a low resolution input image, x is a high resolution output image

$$\mathbb{E}_{\mathbf{x},\mathbf{y}}\ \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\ \mathbb{E}_{t}\ ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_p^p$$

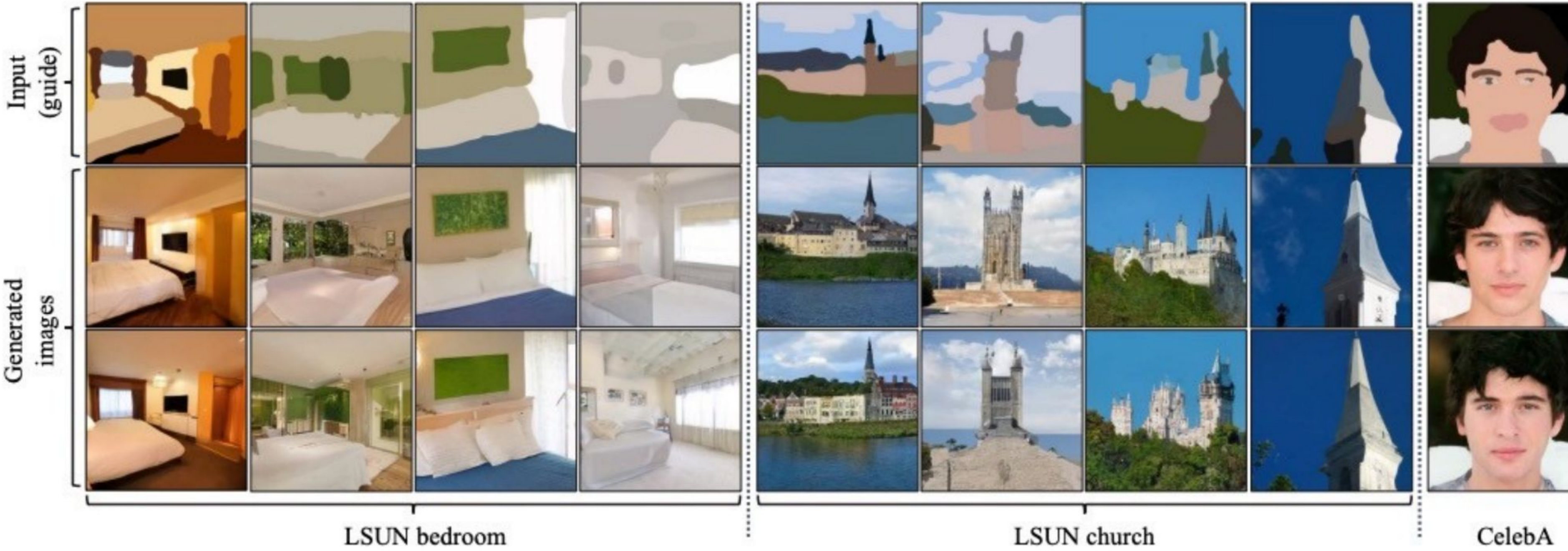


Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

Application: super resolution

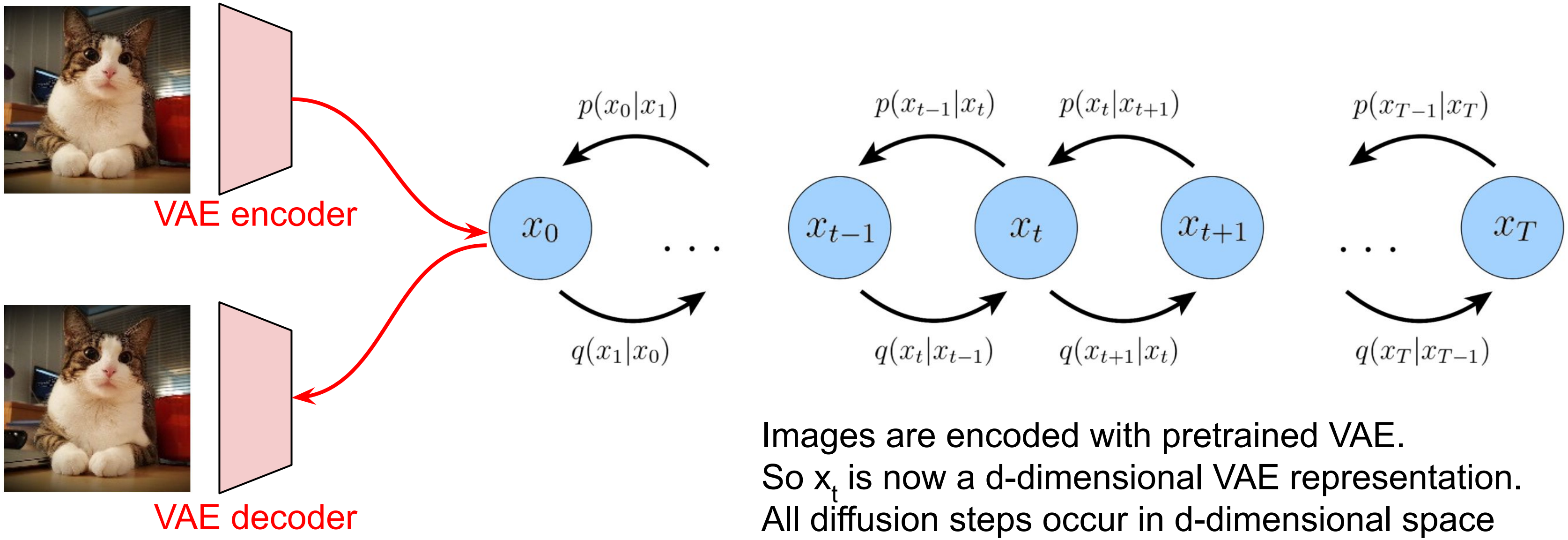
Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

Application: image editing



Meng et al., SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, ICLR 2022

Latent diffusion models: perform diffusion over latent VAE encodings

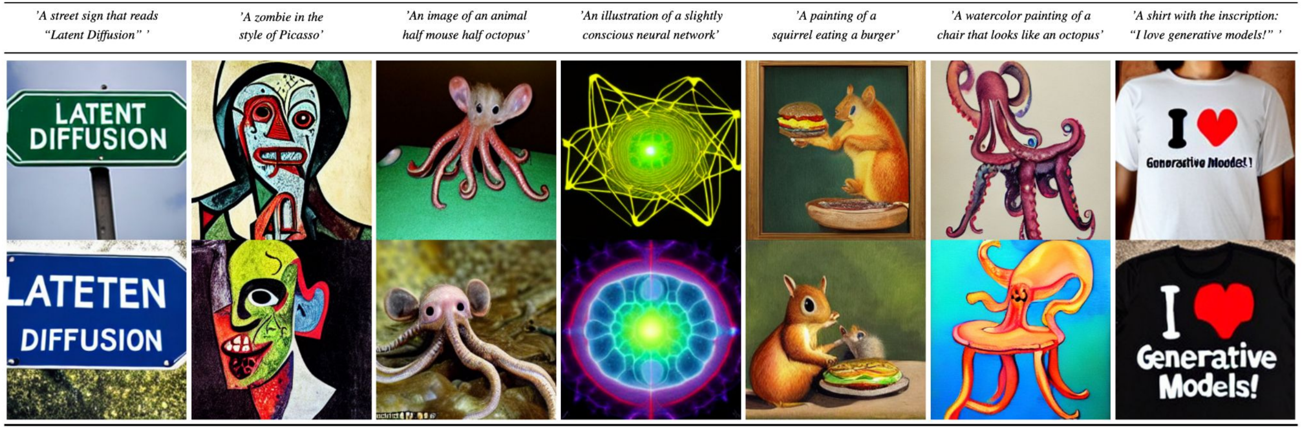


Images are encoded with pretrained VAE.
So x_t is now a d-dimensional VAE representation.
All diffusion steps occur in d-dimensional space
Memory and compute efficient

Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models ArXiv 2022

Stable diffusion - from Stability AI

- Open sourced diffusion model - main model used for research
- Produces 512x512 images
- UNet with 860M params
- ViT-L text encoder with 123M params
- Fits in 10GB VRAM - fits on most GPUs



Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models ArXiv 2022

Imagen examples

A dragon fruit wearing karate belt in the snow.

A relaxed garlic with a blindfold reading a newspaper while floating in a pool of tomato soup.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

Sora video diffusion model

https://openai.com/sora

How did they do it?

- More data (unknown data source)
- Replaced U-Net architecture with transformers

Comparing the different generative models

Q. Which ones are VAEs good at?

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

Comparing the different generative

VAEs are bad at generating high quality samples

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

Comparing the different generative models

Q. Which ones are GANs good at?

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

Comparing the different generative models

GANs suffer from mode collapse

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

Comparing the different generative models

Q. Which ones are Diffusion models good at?

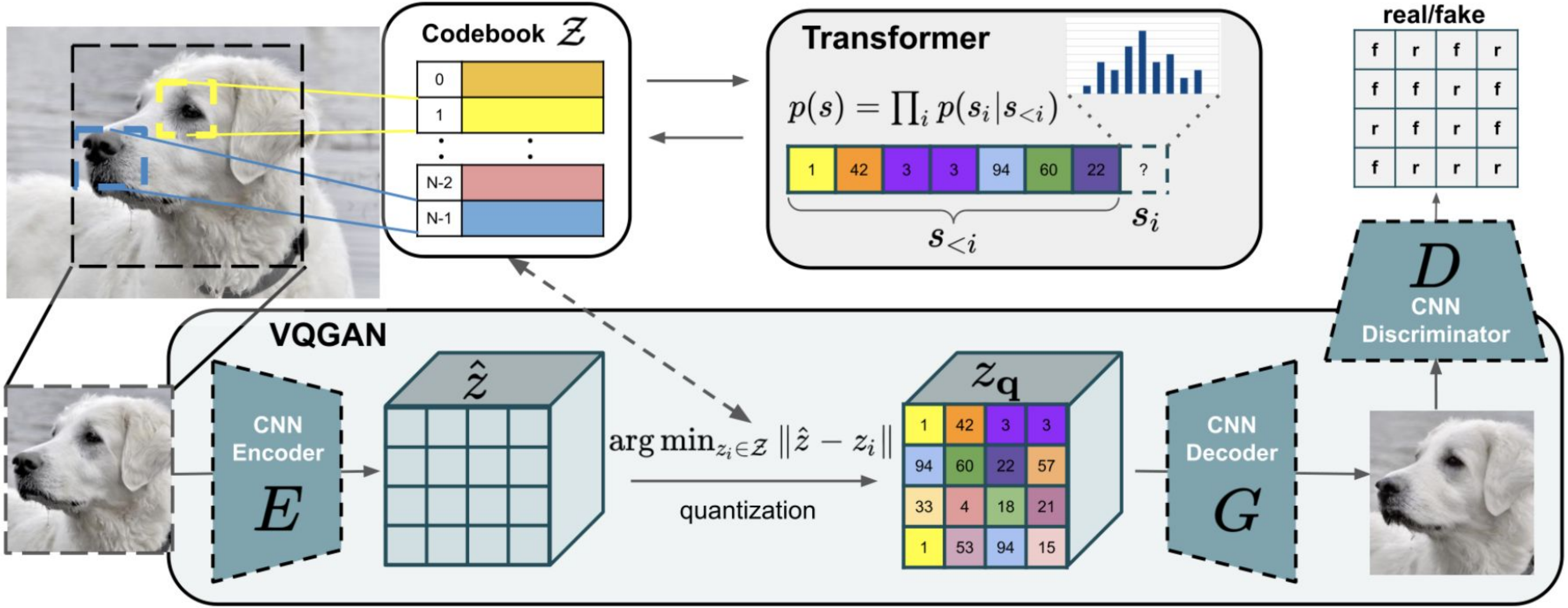
	VAEs	**GANs**	**Diffusion**
Mode coverage / diversity of generations	✅	❌	
Fast sampling	✅	✅	
High quality samples	❌	✅	

Comparing the different generative models

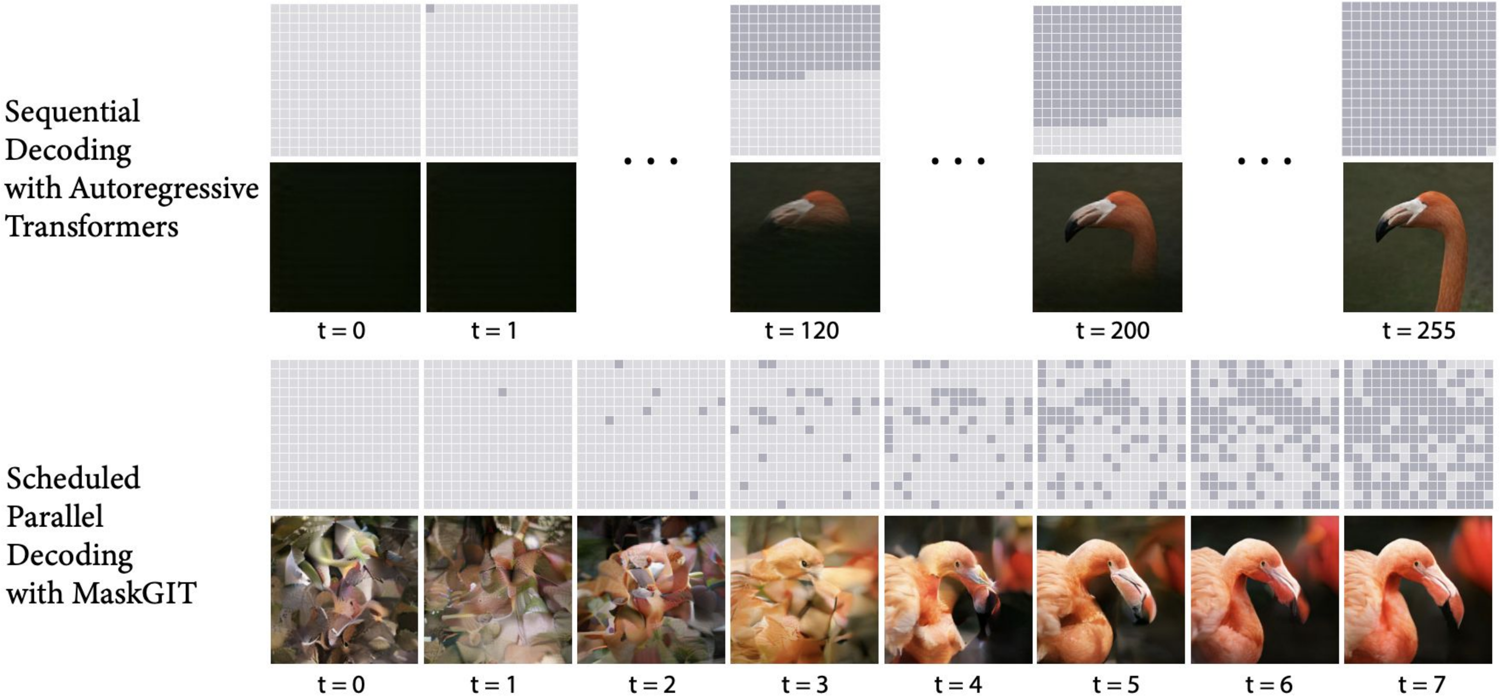
Diffusion models are bad at sampling fast.

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

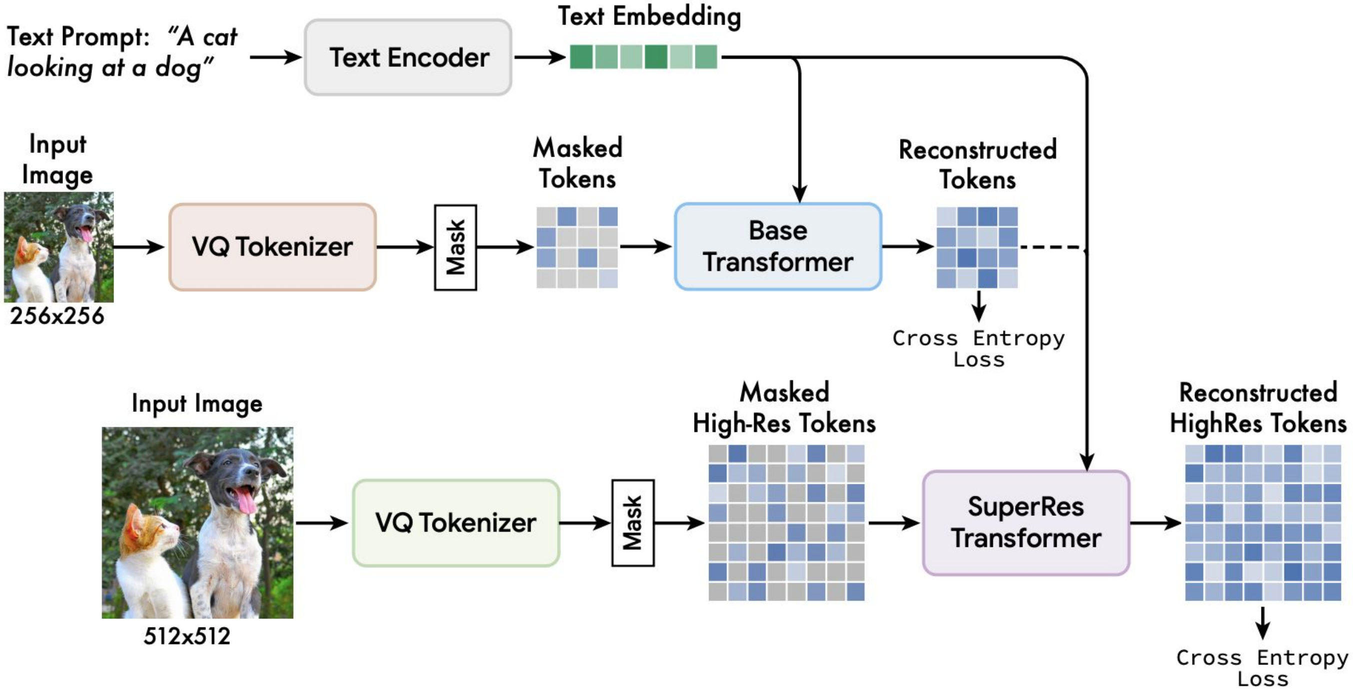
Discrete token generation models



Different “Stage 2” possibilities



Can be scaled pretty well



Next: Deep Reinforcement Learning