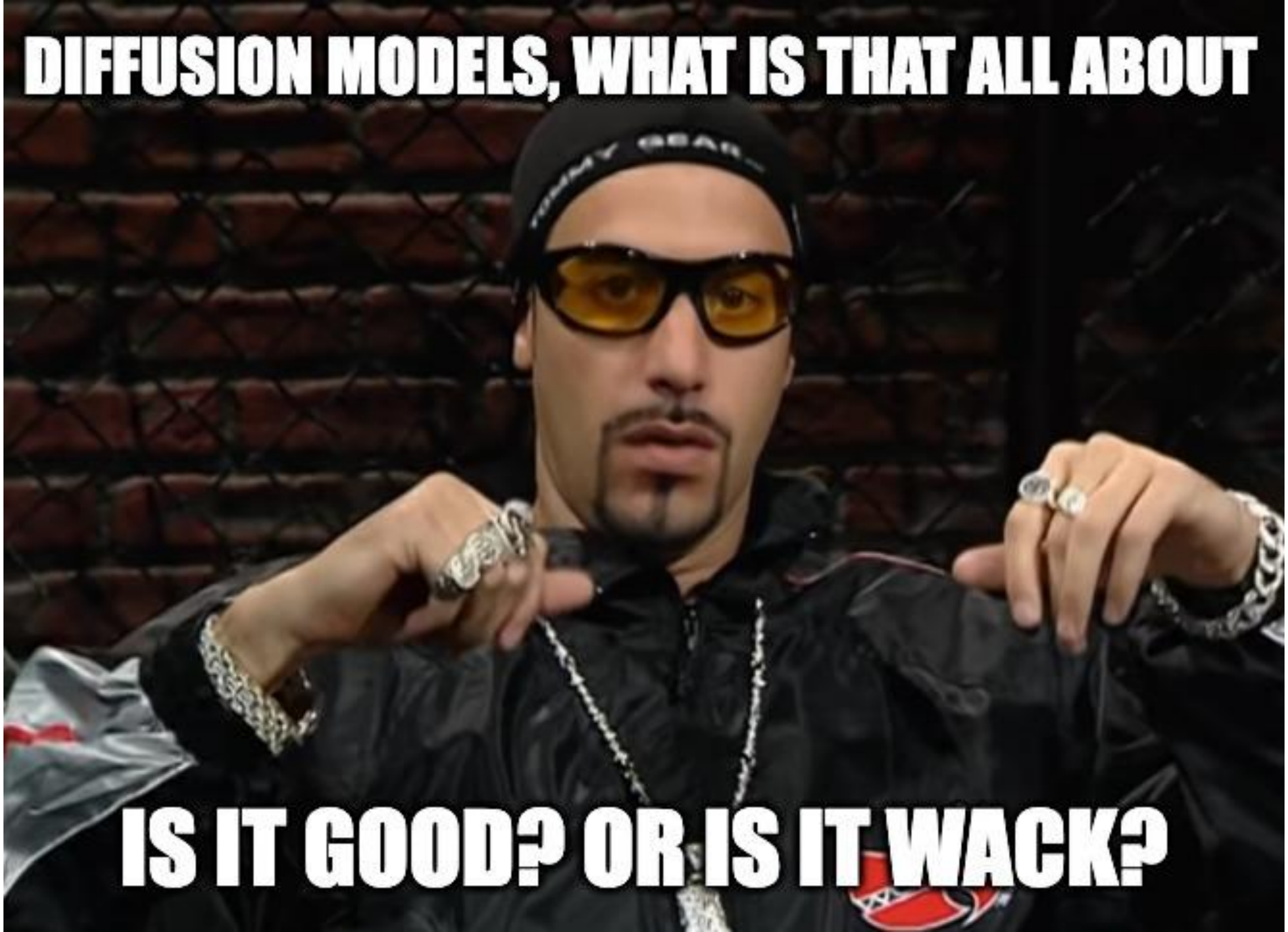

DIFFUSION MODELS, WHAT IS THAT ALL ABOUT
IS IT GOOD? OR IS IT WACK?

About Me

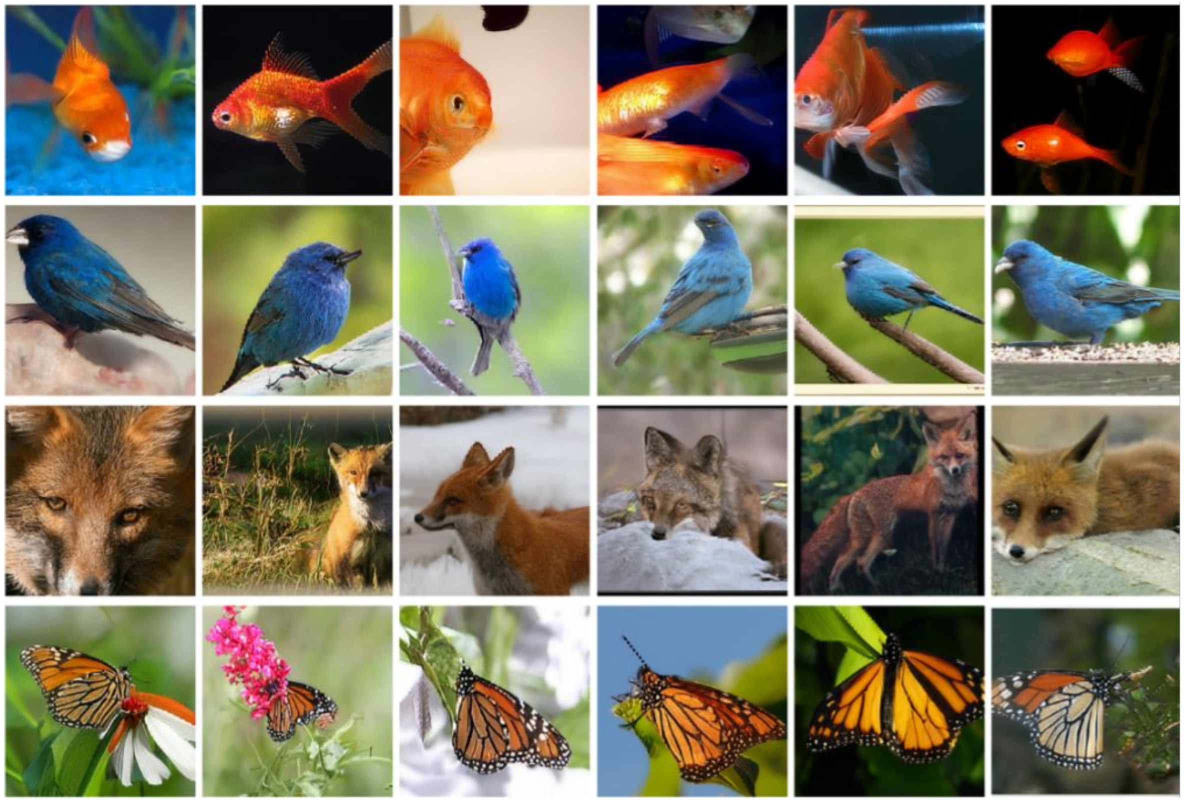
I'm Calvin, a 3rd year PhD student here at Brown, advised by Chen Sun

- I work on Diffusion Modeling and Reinforcement Learning

Diffusion models are really good at learning conditional distributions.

$$p(x \mid y)$$

Use Case: Class-Conditioned Generation $p(image \mid class\_label)$



source: [Image Super-Resolution via Iterative Refinement]

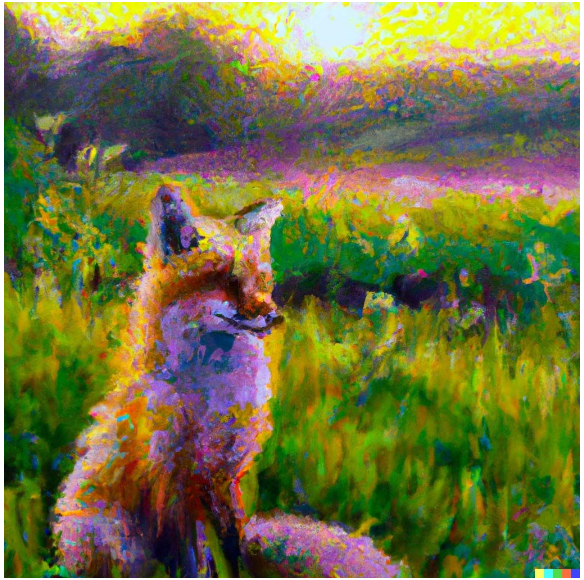
Use Case: Text-to-Image Generation

$p(image \mid text\_caption)$

"a painting of a fox sitting in a field at sunrise in the style of Claude Monet"

Parti (but pretend it is ImageN)

StableDiffusion

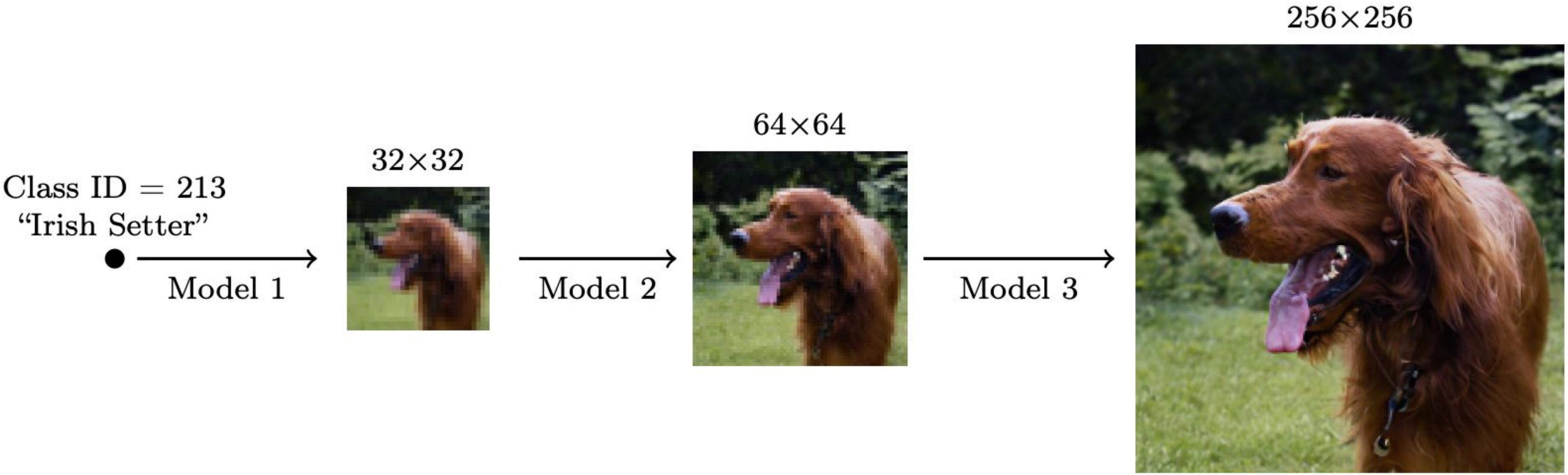


Dall-E 2.0

source: ImageN, StableDiffusion, Dall-E 2.0

Use Case: Super Resolution

$p(image \mid low\_res)$



source: [Cascaded Diffusion Models]()

Recap: Generative Modeling

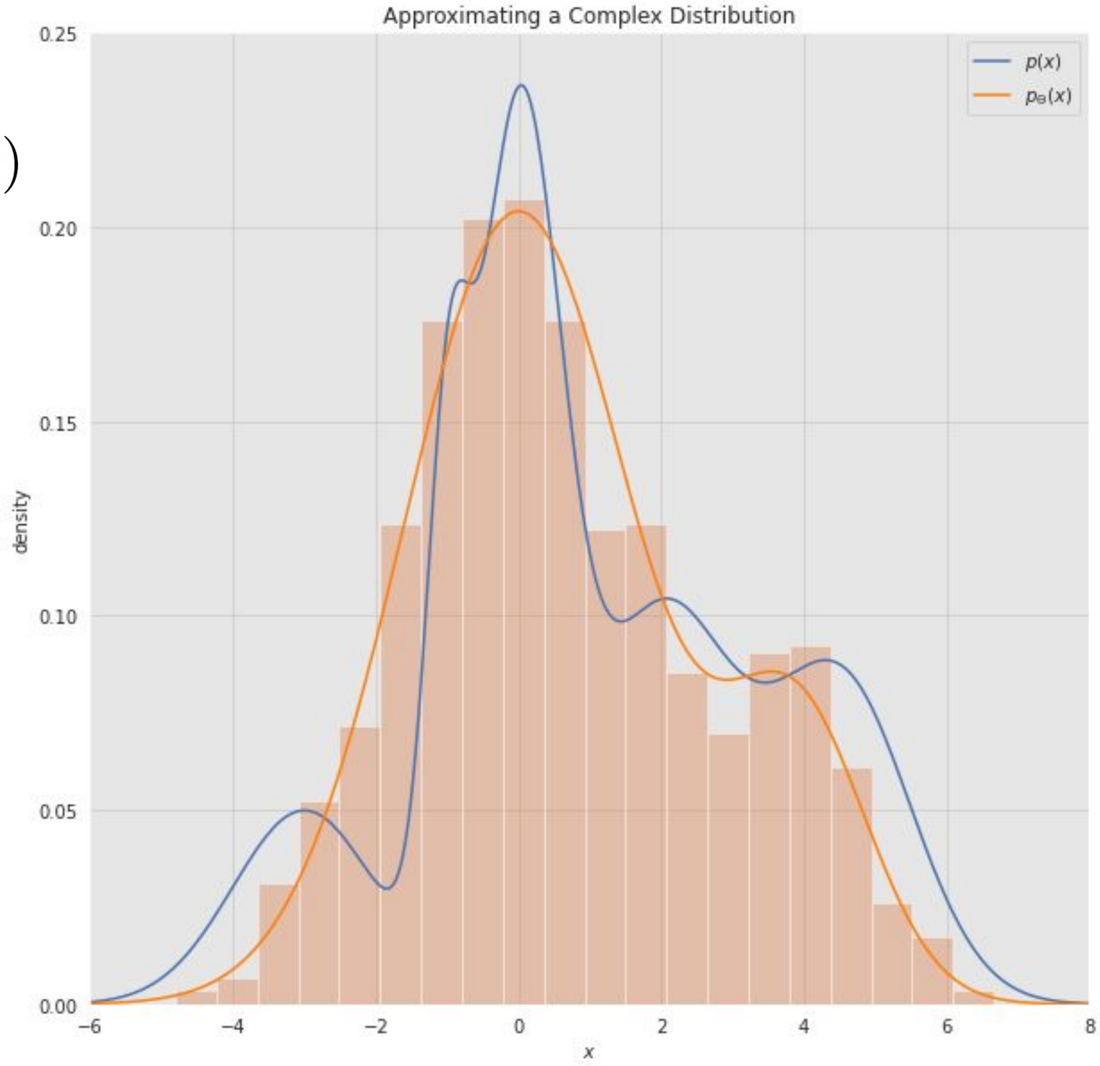
Recall the goal of generative modeling - learning a *model* of a distribution from which we can *generate* new samples.

Given $\boldsymbol{x} \sim p(\boldsymbol{x})$ we might want to learn $p_\theta(\boldsymbol{x}) \approx p(\boldsymbol{x})$ (*modeling*)

Then, we can generate new samples $\boldsymbol{x}^* \sim p_\theta(\boldsymbol{x})$ (*generation*)

Why is this useful?

Given $\boldsymbol{x} \sim p(\boldsymbol{x})$



Generative Modeling: Themes

What are some common themes of generative modeling?

- We want to learn some **complex** distribution $p_\theta(\boldsymbol{x}) \approx p(\boldsymbol{x})$
- But we only have access to some **simple** distributions (such as Gaussians)

Generative Modeling: Themes

What are some common themes of generative modeling?

- We want to learn some **complex** distribution $p_\theta(\boldsymbol{x}) \approx p(\boldsymbol{x})$
- But we only have access to some **simple** distributions (such as Gaussians)

Idea: Let's learn a complex function (aka a neural network) to transform a simple distribution sample into a complex one!

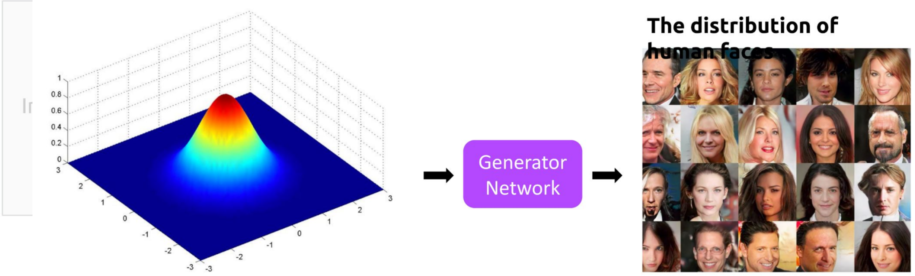
- Gaussian Sample ==(neural net)==> Data Sample

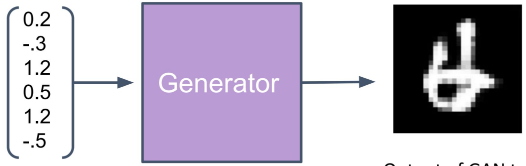
Generative Modeling: Themes

Idea: Let's learn a complex function (aka a neural network) to transform a simple distribution sample into a complex one!

- Gaussian Sample ==(neural net)==> Data Sample

You have seen this before in:





Random Vector (sampled from unit normal distribution)

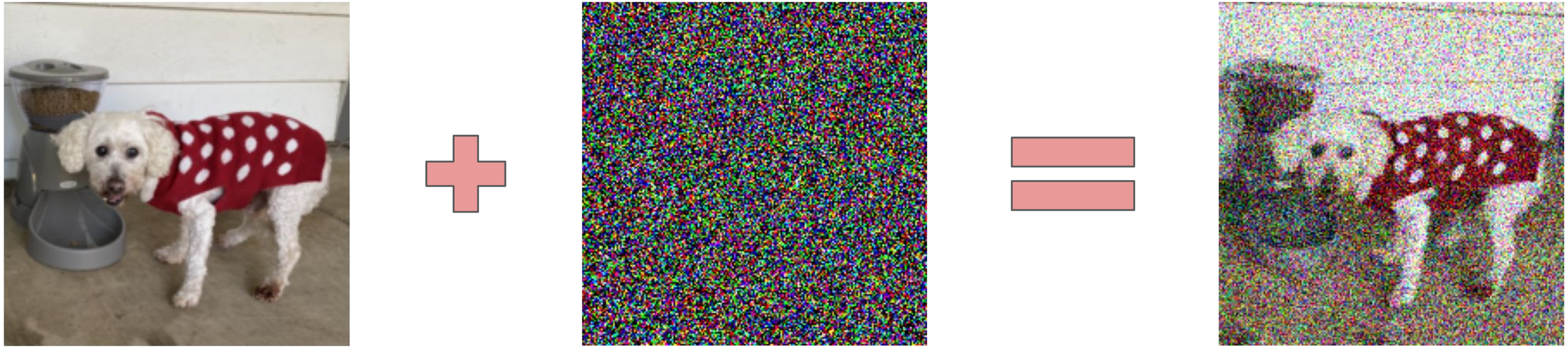
Output of GAN trained on MNIST images

Diffusion Models: a TLDR

An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.

Diffusion Models: a TLDR

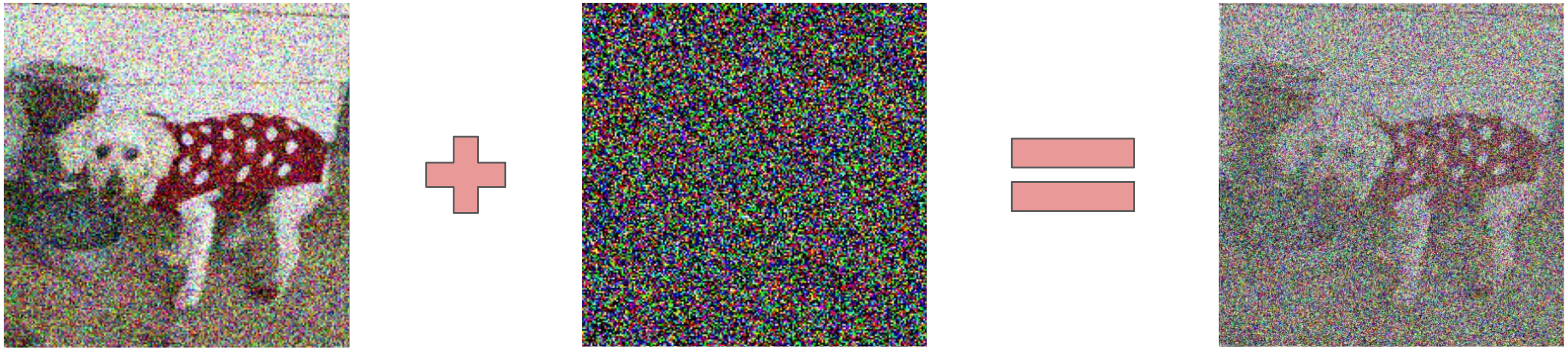
An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.



One Step

Diffusion Models: a TLDR

An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.



Another Step

Diffusion Models: a TLDR

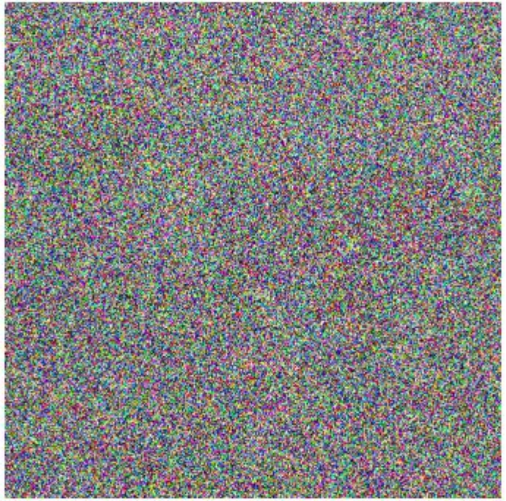
An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.

Many Steps

Diffusion Models: a TLDR

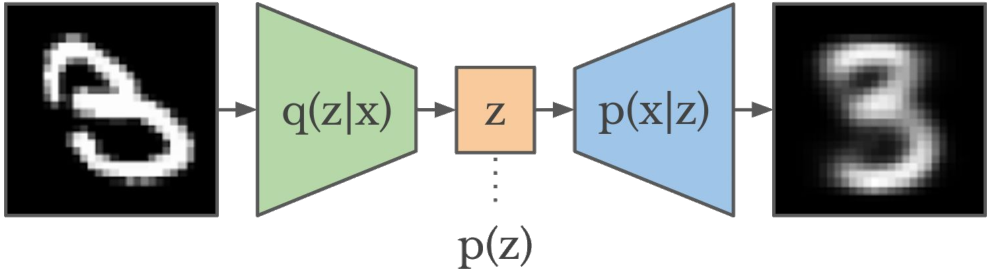
An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.

- Diffusion models simply learn to **reverse** this procedure over many timesteps



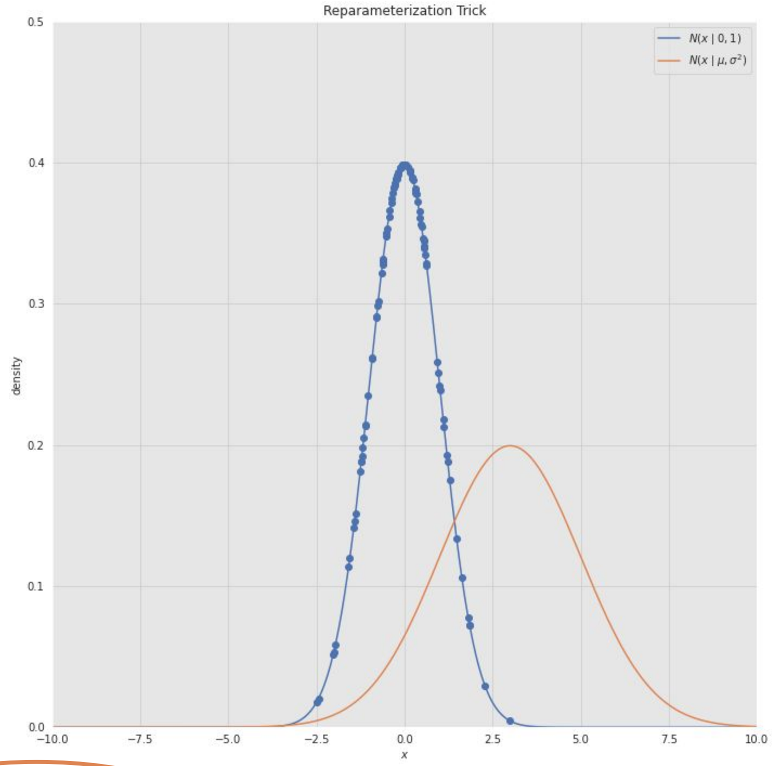
Recap: Variational Autoencoders

Visually, we often see a VAE as:



How do we perform backpropagation through samples?

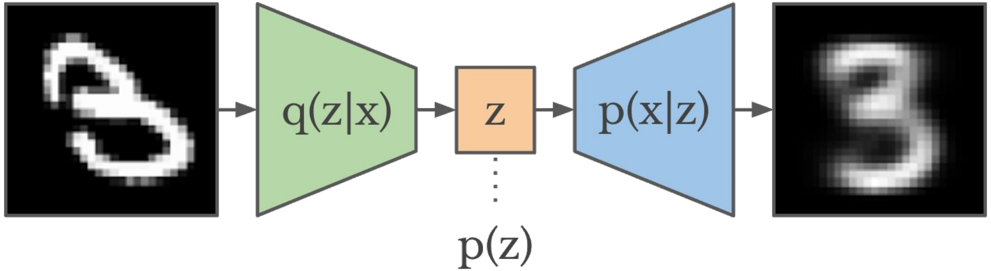
Recap: Reparameterization Trick



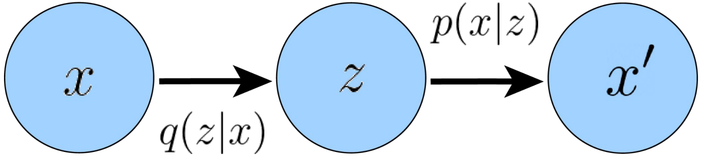
For $x \sim \mathcal{N}(x \mid \mu, \sigma^2)$, ϵ , where $\epsilon \sim \mathcal{N}(x \mid 0, \mathrm{I})$

Recap: Variational Autoencoders

Visually, we often see a VAE as:

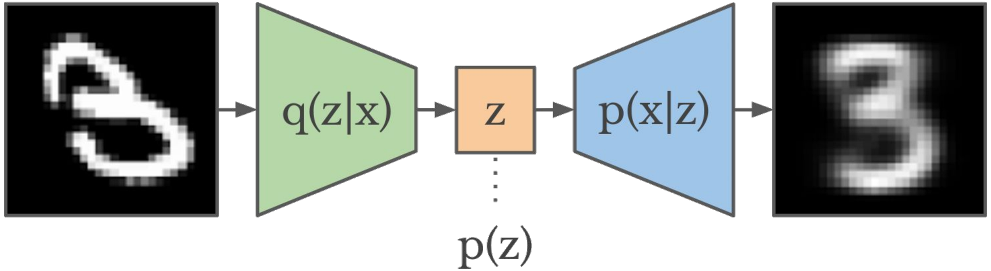


Or as:

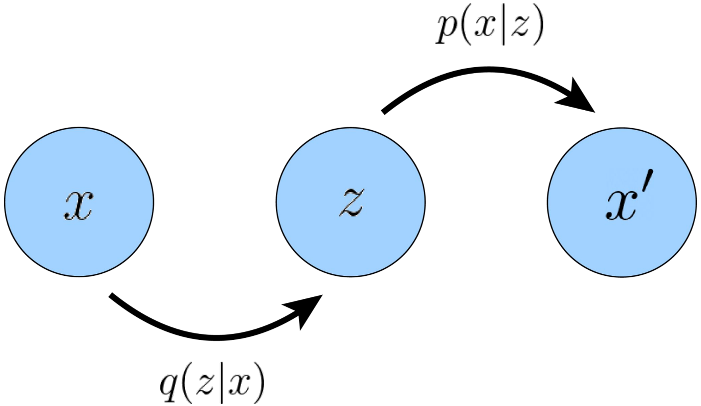


Recap: Variational Autoencoders

Visually, we often see a VAE as:

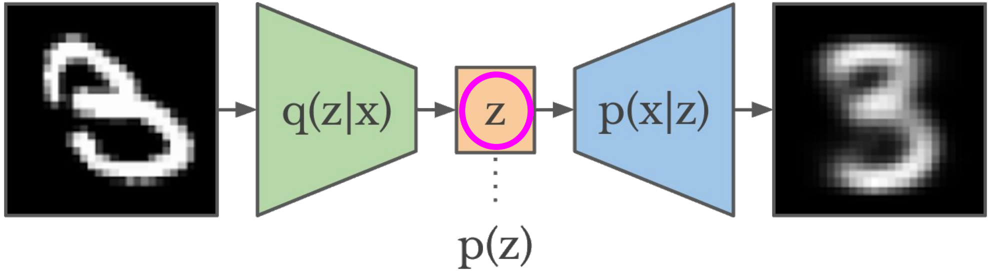


Or as:

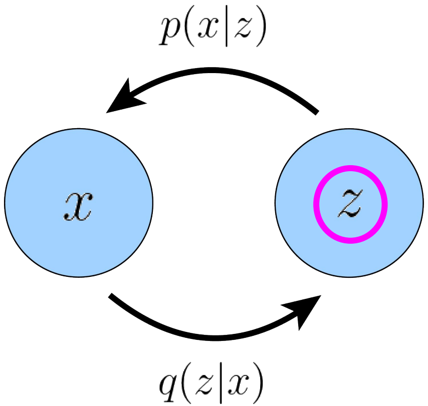


Recap: Variational Autoencoders

Visually, we often see a VAE as:



Or as:

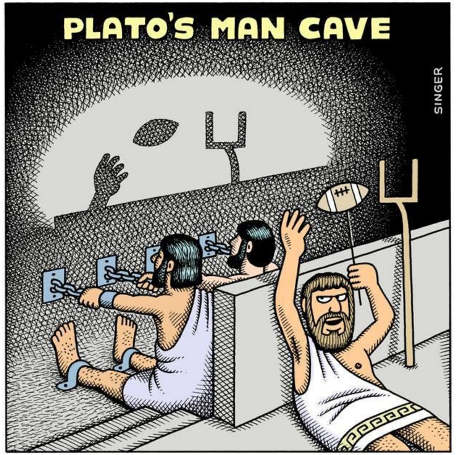


…but what's the intuition behind what is learned?

Generative Modeling with Latent Variables

Given $\boldsymbol{x} \sim p(\boldsymbol{x})$ we might want to learn $p_\theta(\boldsymbol{x}) \approx p(\boldsymbol{x})$ (*modeling*)

What if we assume latent variables $\boldsymbol{z}$ exist?



Elon has an idea…

Elon has an idea…

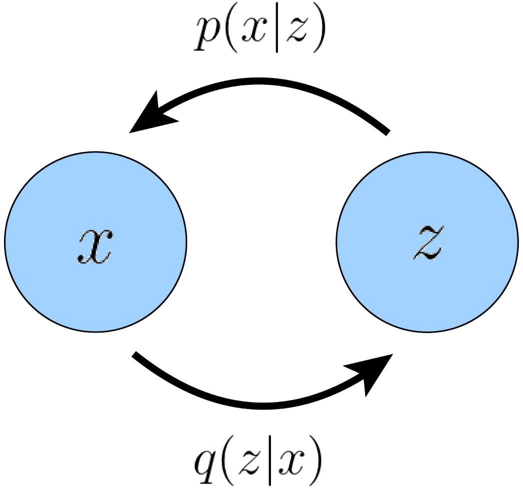
Elon has an idea…

Elon has an idea…

Hierarchical VAEs

Generalize VAEs by enabling a hierarchy of latents $z = z_1, ...z_T$

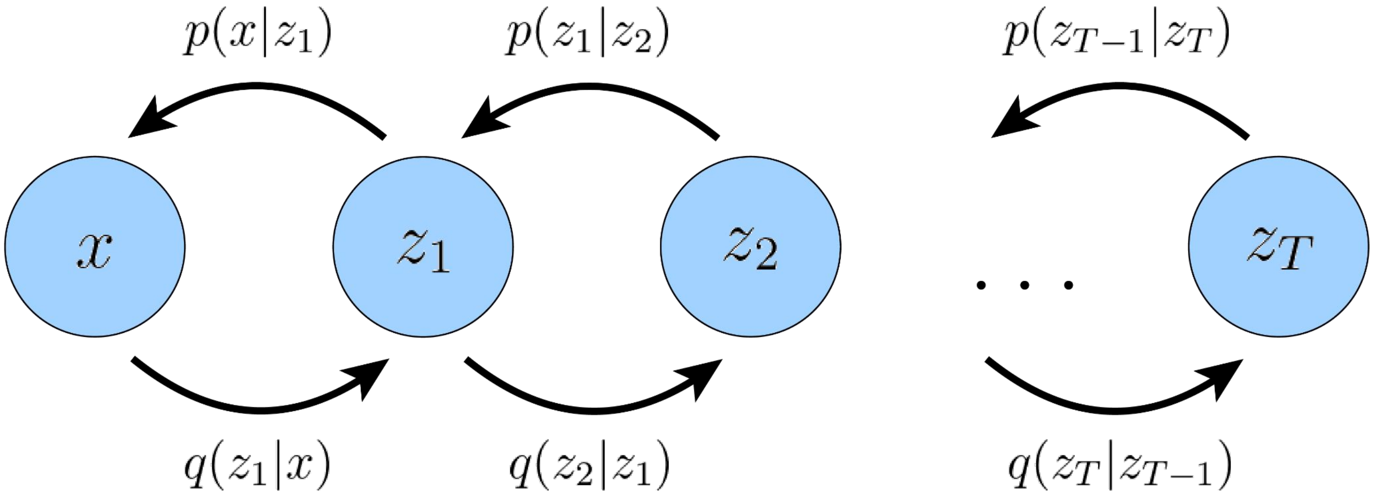
This is essentially learning a bunch of stacked VAEs



Hierarchical VAEs

Generalize VAEs by enabling a hierarchy of latents $z = z_1, \ldots z_T$

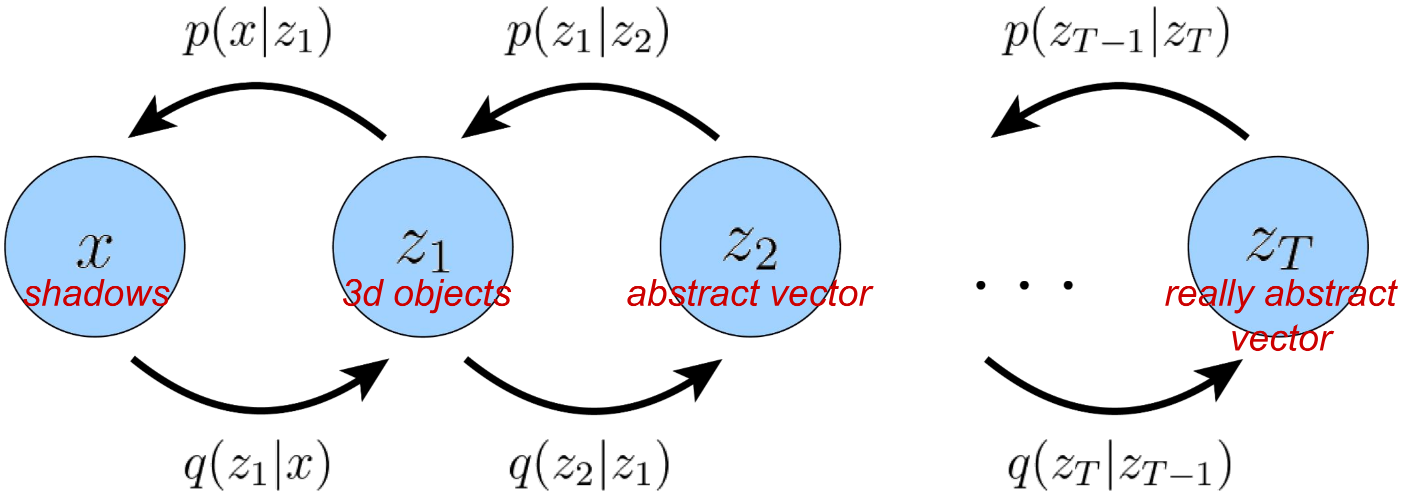
This is essentially learning a bunch of stacked VAEs



Disclaimer: Elon did not actually come up with this idea.

Hierarchical VAEs

Let's think like a caveman…

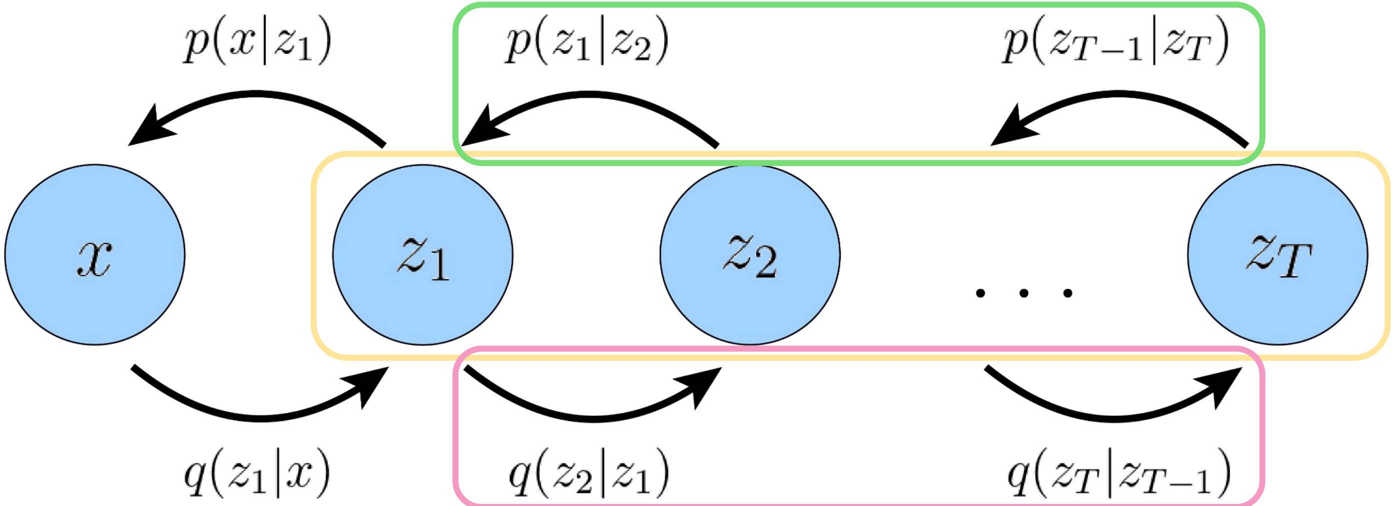


Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

…what if we assume all latent dimensions are the same?

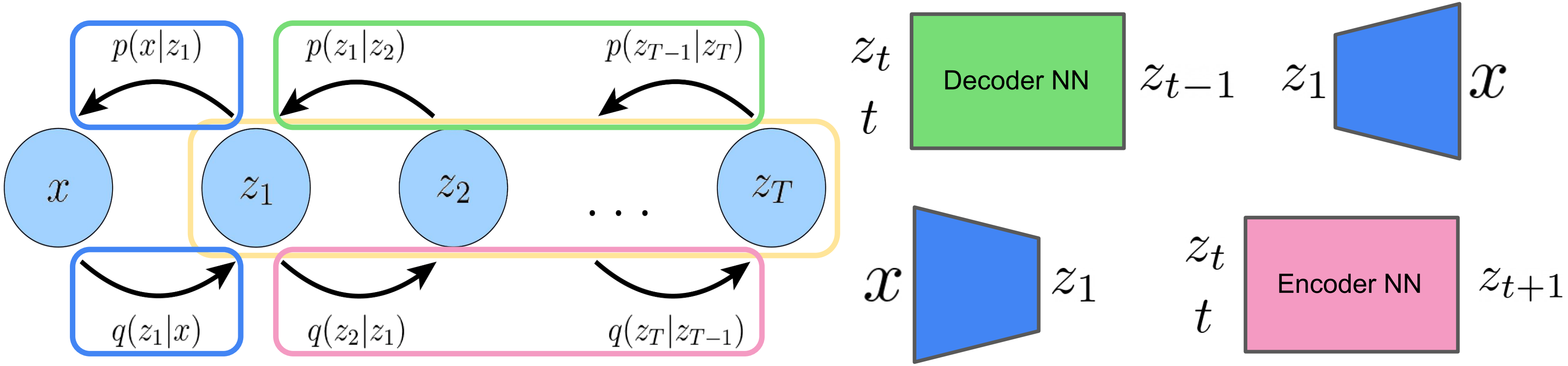


Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

…what if we assume all latent dimensions are the same?

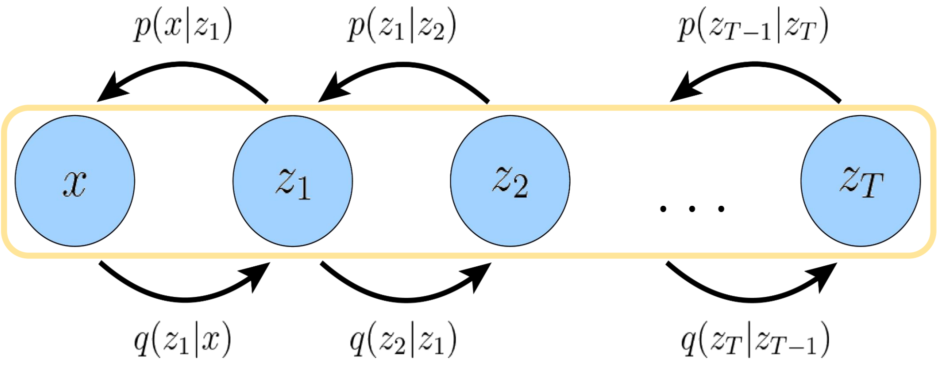


Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

*…what if we assume **all** dimensions are the same?*

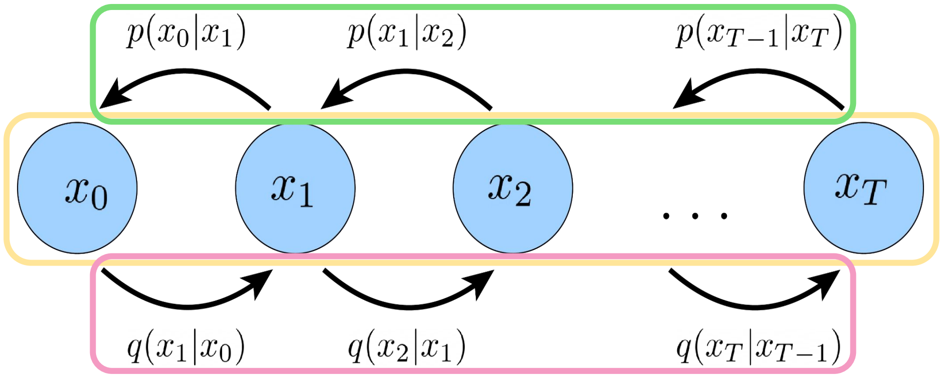


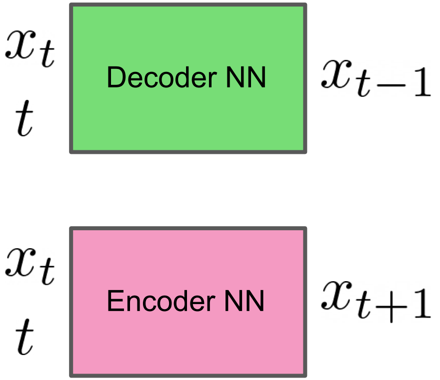
Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

*…what if we assume **all** dimensions are the same?*



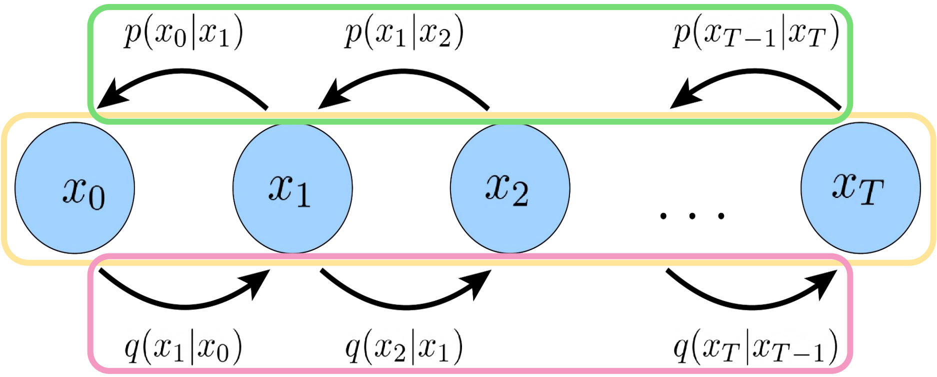


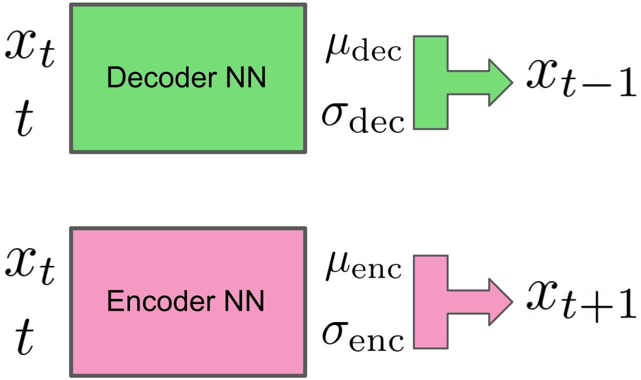
Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

*…what if we assume **all** dimensions are the same?*





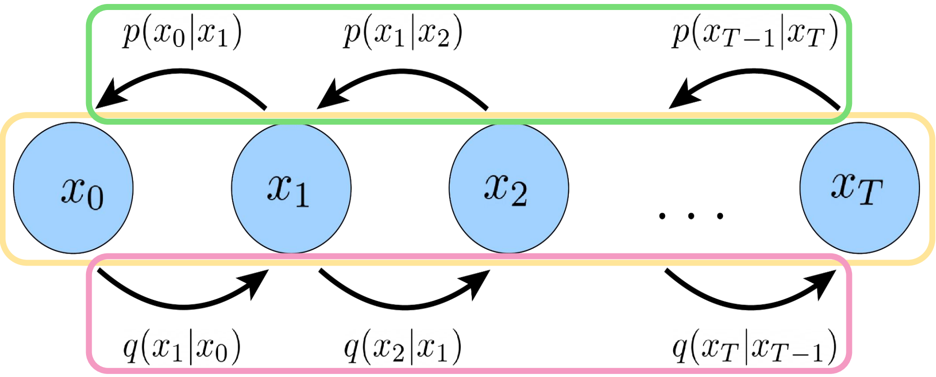
Hierarchical VAEs

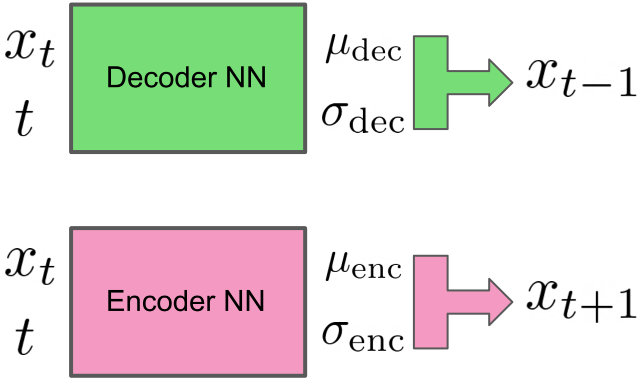
Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

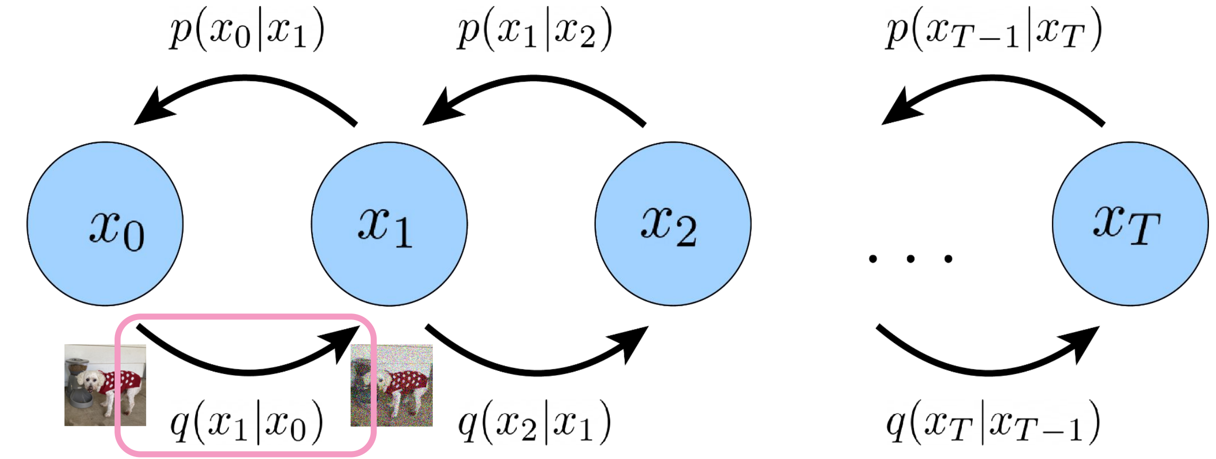
*…what if we assume **all** dimensions are the same?*

…what if we assume all encoder transitions are known Gaussians centered around their previous input?





Let's take a look at one encoding

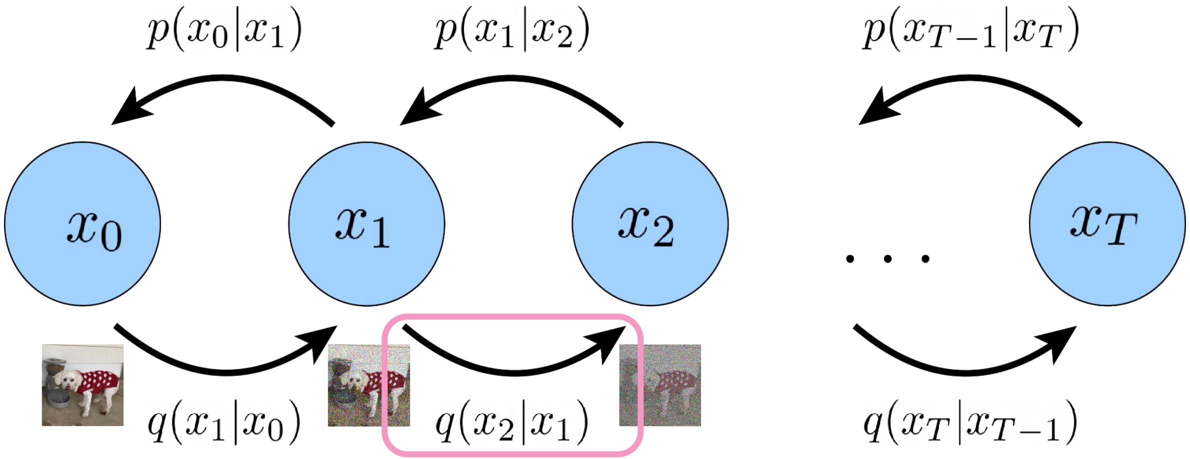


$$q(x_1|x_0) = \mathcal{N}(x_1|x_0, \sigma_1^2\mathrm{I})$$

$x_1 \sim q(x_1|x_0)$ $\boldsymbol{x}_0$ $\boldsymbol{\epsilon}_0$

reparam. trick!

Let's take a look at one encoding

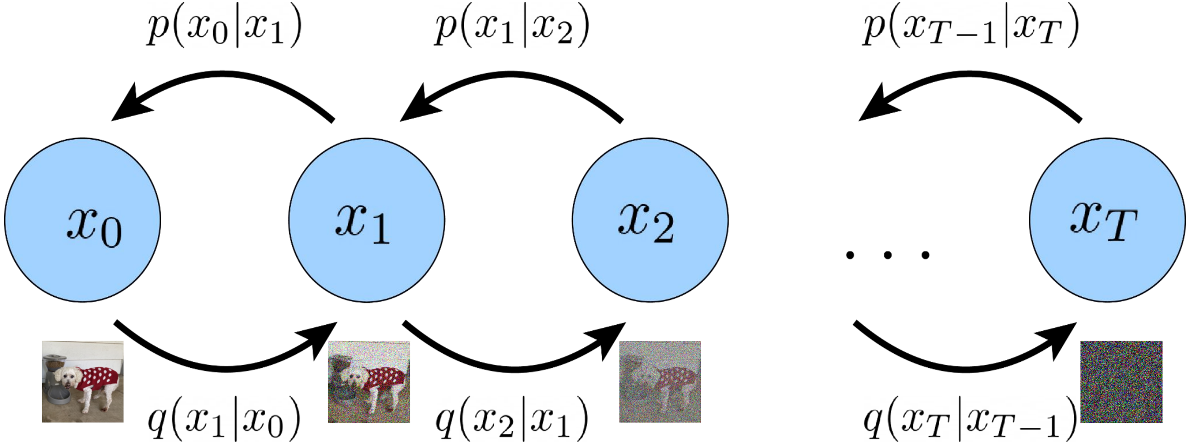


$$q(x_2|x_1) = \mathcal{N}(x_2|x_1, \sigma_2^2 \mathrm{I})$$

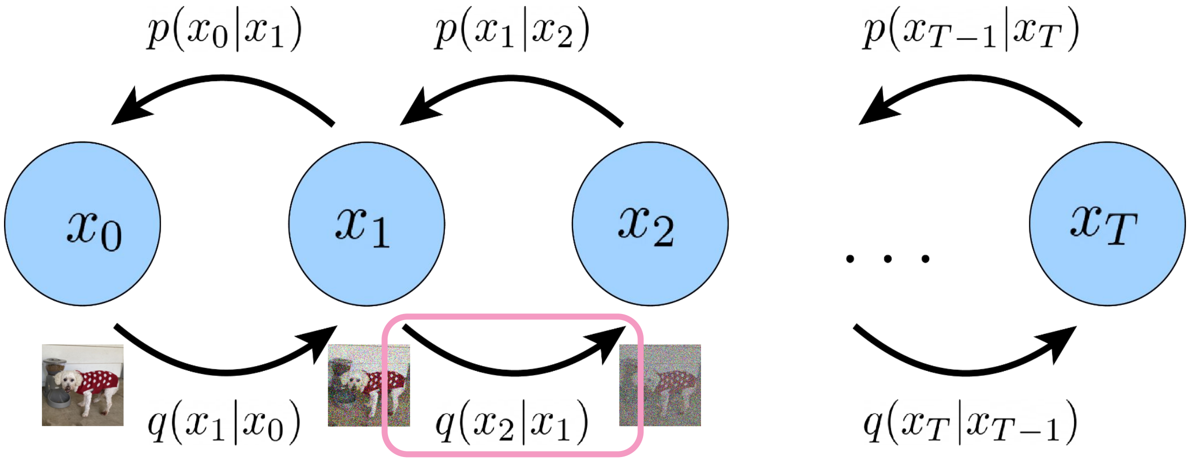
$x_2 \sim q(x_2|x_1)$ $\quad$ x_1 $\quad$ $\boldsymbol{\epsilon}_0$

reparam. trick!

Let's take a look at one encoding

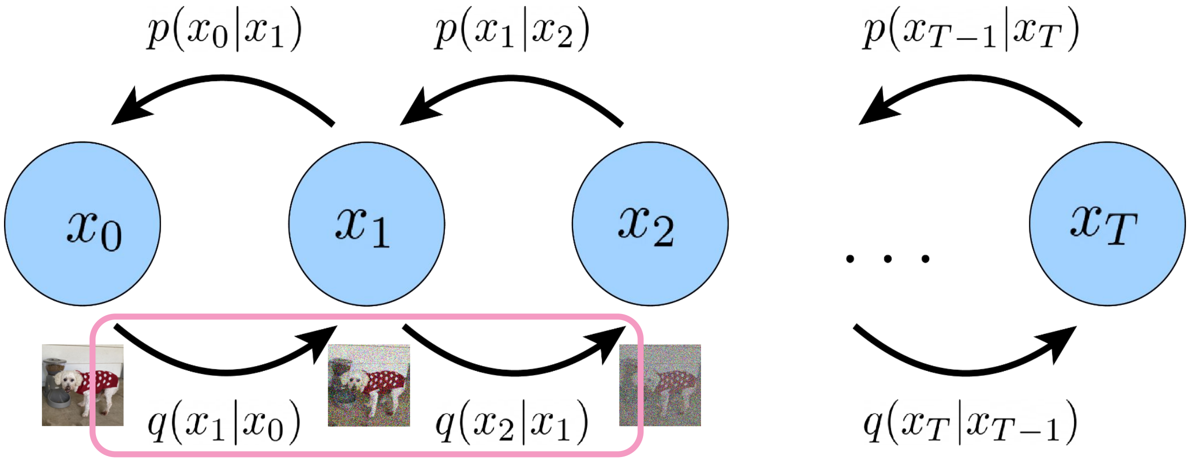


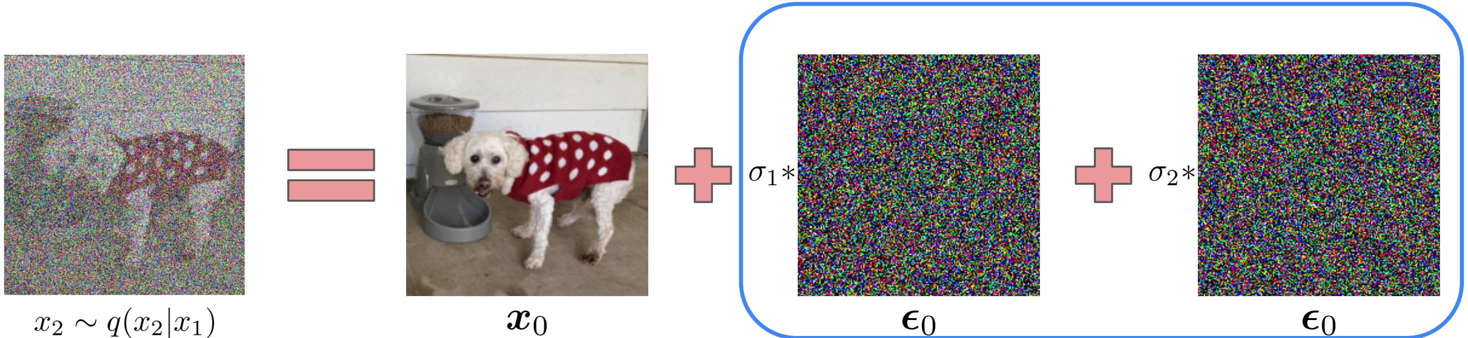
Let's take a look at one encoding



reparam. trick!

Let's take a look at one encoding

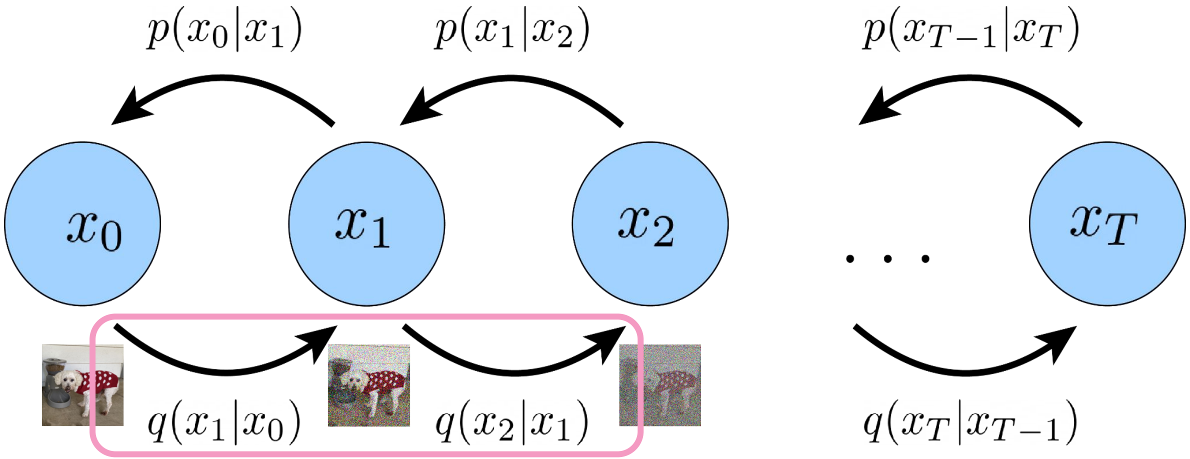




Aggregate into 1 sample!

reparam. trick!

Let's take a look at one encoding



where,

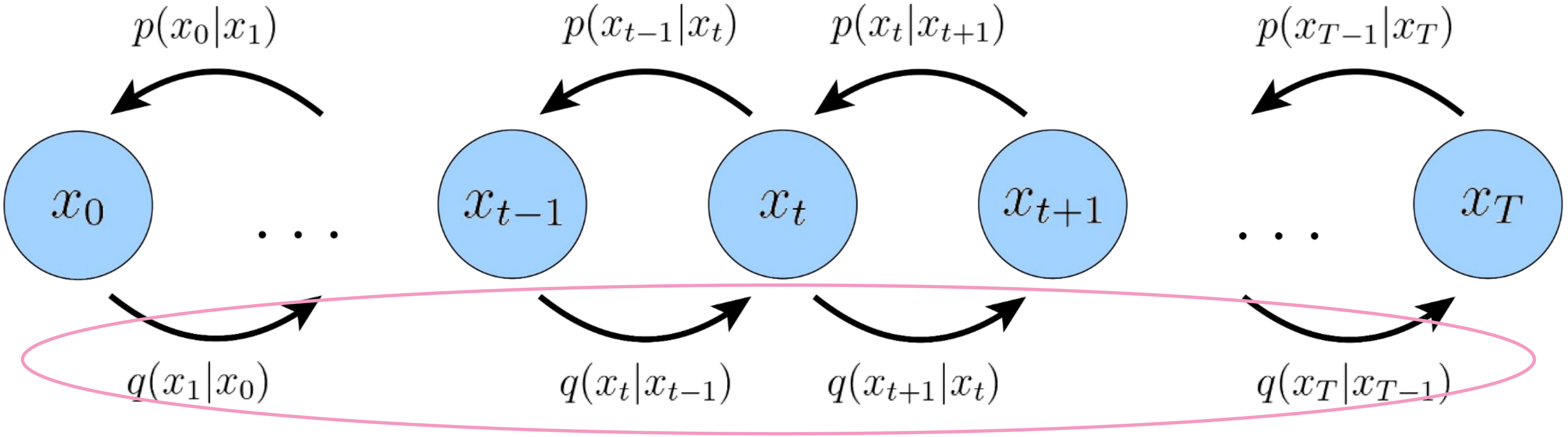
$$\alpha_2 = \sqrt{\sigma_1^2 + \sigma_2^2}$$

and,

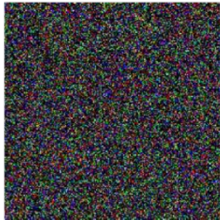
$$q(x_2|x_0) = \mathcal{N}(x_2|x_0, \alpha_2^2)$$

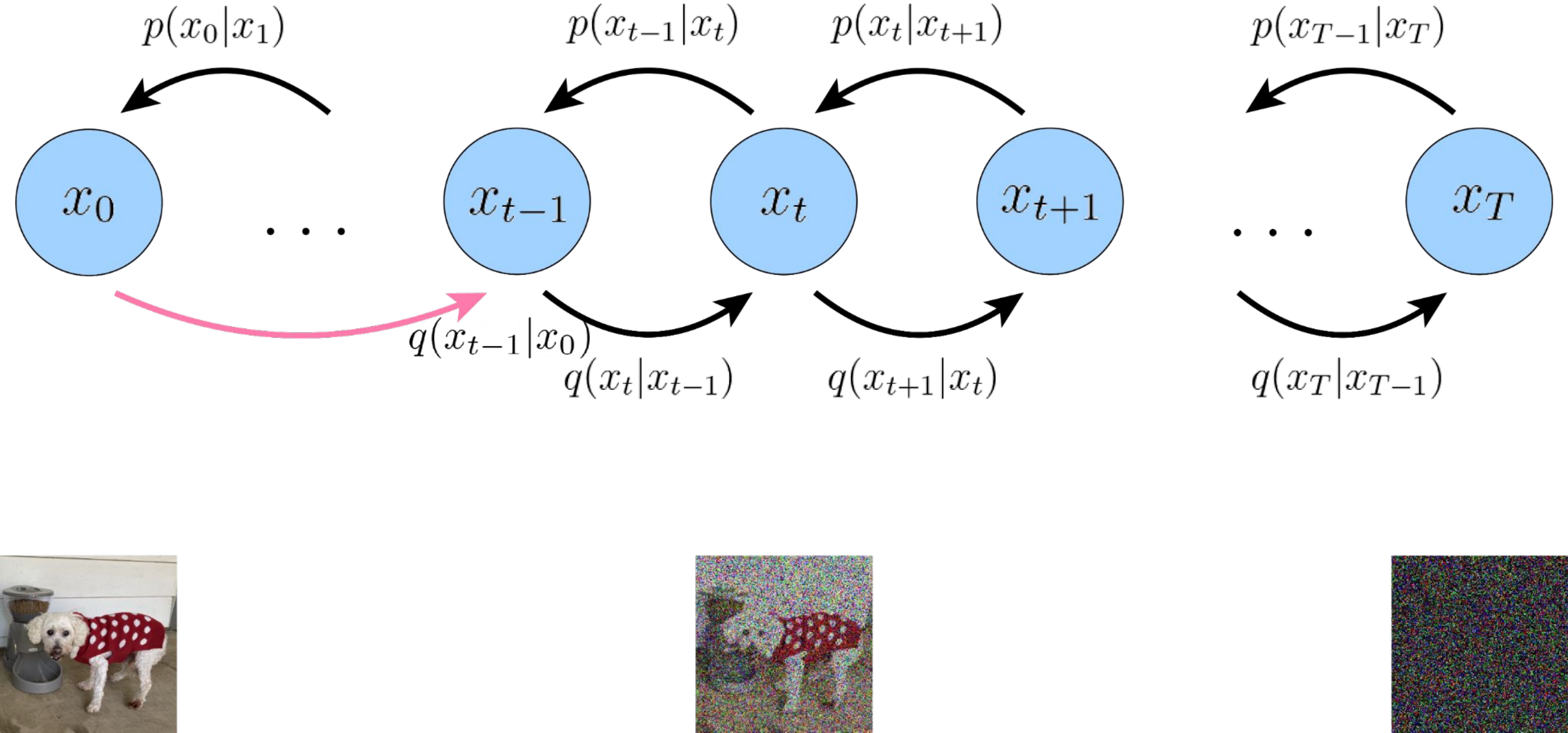
Aggregate into 1 sample!

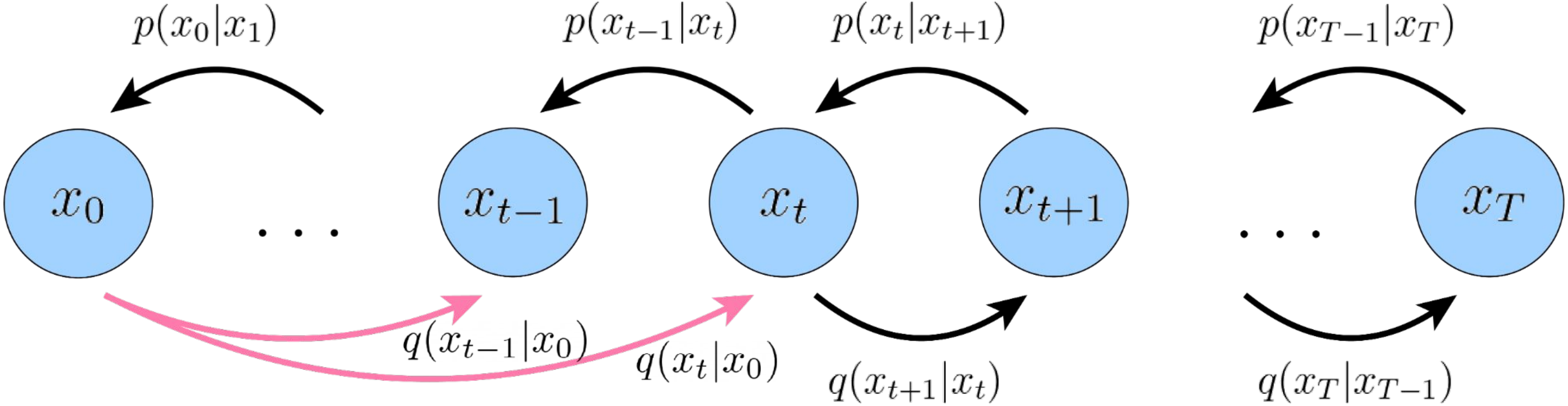
reparam. trick!

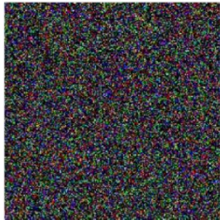


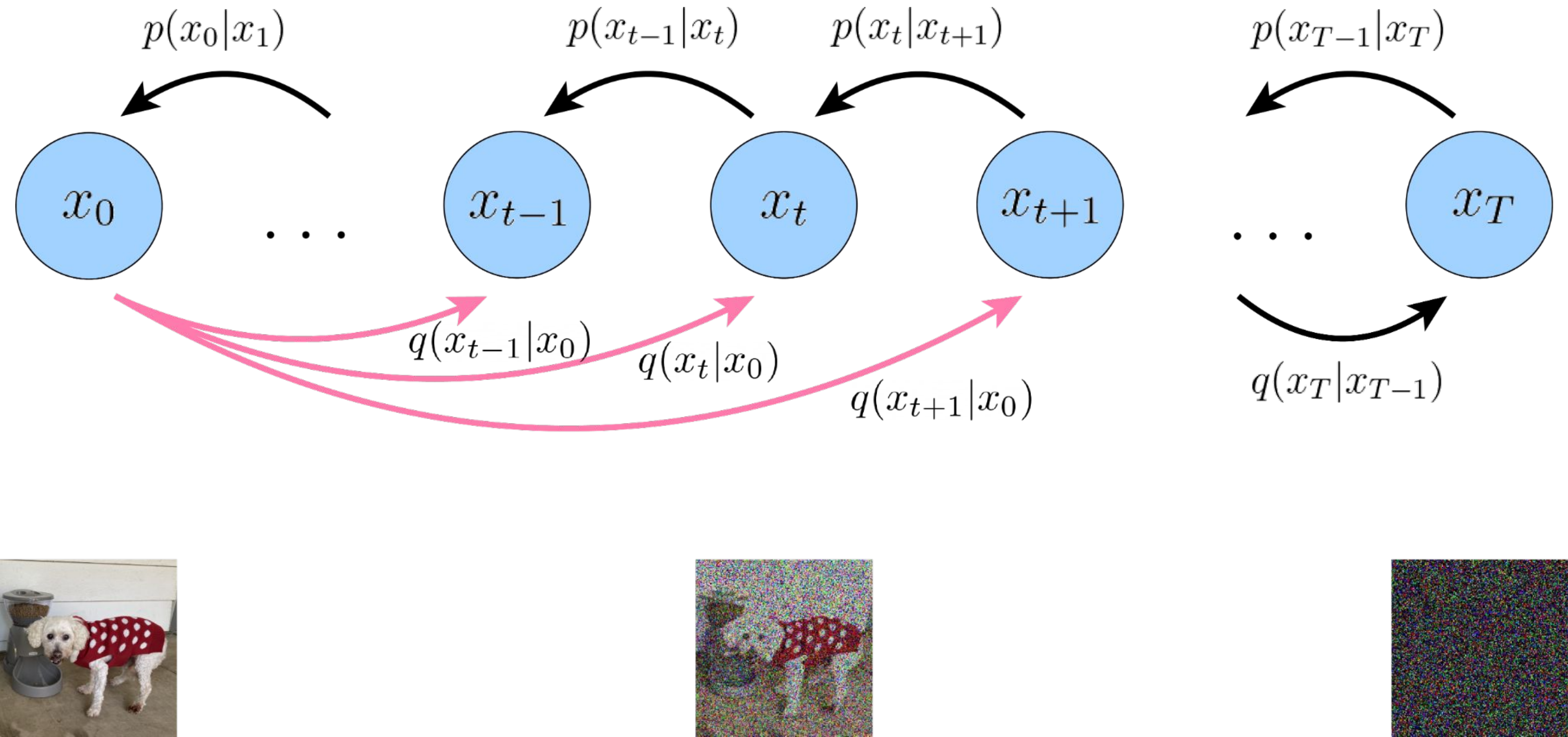
Individual (known) Gaussians!




$p(x_0|x_1)$
$p(x_{t-1}|x_t)$
$p(x_t|x_{t+1})$
$p(x_{T-1}|x_T)$
x_0
$\dots$
x_{t-1}
x_t
x_{t+1}
$\dots$
x_T
$q(x_{t-1}|x_0)$
$q(x_t|x_{t-1})$
$q(x_{t+1}|x_t)$
$q(x_T|x_{T-1})$


$p(x_0|x_1)$
$p(x_{t-1}|x_t)$
$p(x_t|x_{t+1})$
$p(x_{T-1}|x_T)$
x_0
$\dots$
x_{t-1}
x_t
x_{t+1}
$\dots$
x_T
$q(x_{t-1}|x_0)$
$q(x_t|x_0)$
$q(x_{t+1}|x_t)$
$q(x_T|x_{T-1})$

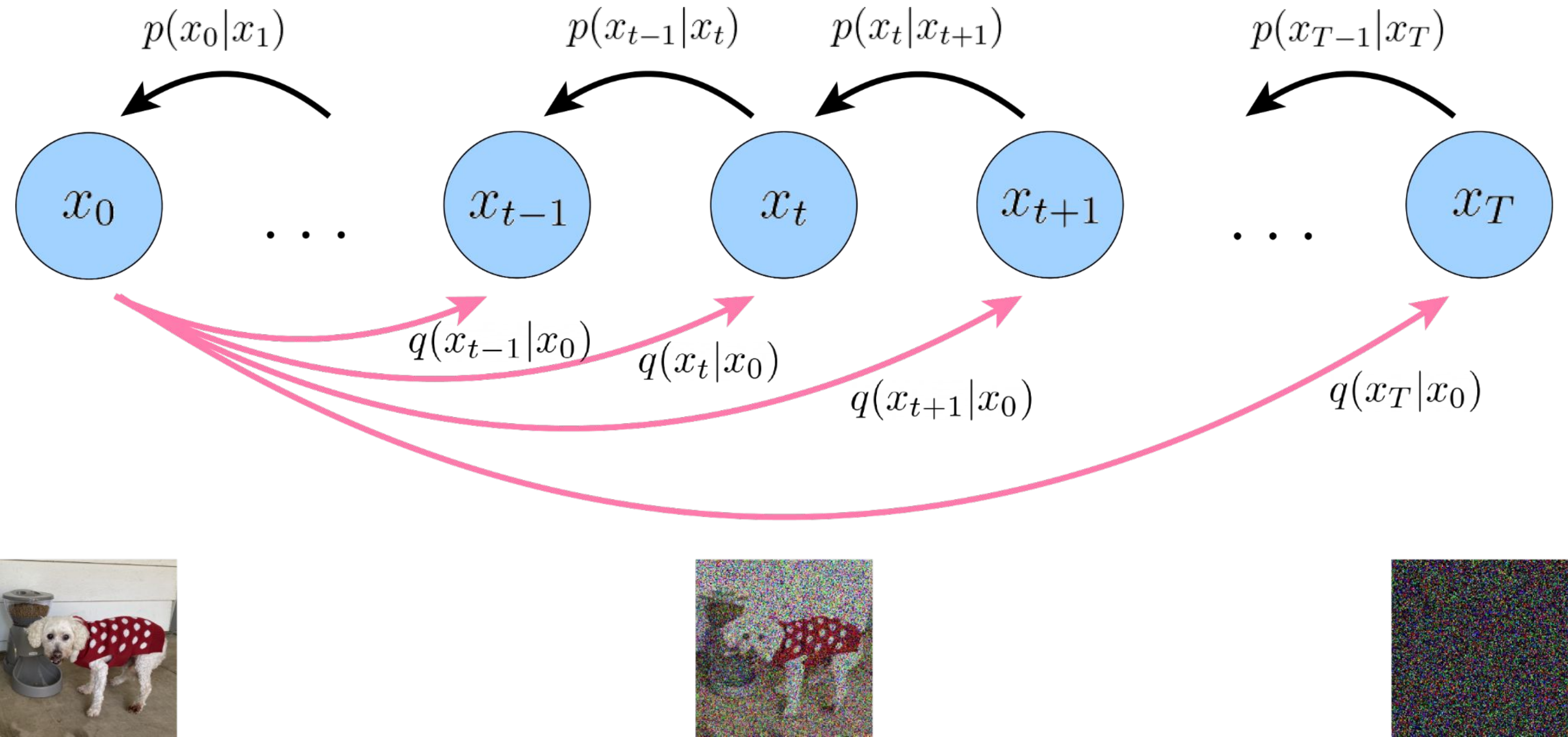




$p(x_0|x_1)$
$p(x_{t-1}|x_t)$
$p(x_t|x_{t+1})$
$p(x_{T-1}|x_T)$
x_0
$\dots$
x_{t-1}
x_t
x_{t+1}
$\dots$
x_T
$q(x_{t-1}|x_0)$
$q(x_t|x_0)$
$q(x_{t+1}|x_0)$
$q(x_T|x_{T-1})$

$q(x_t|x_0)$ is a Gaussian, for arbitrary t!

$q(x_t|x_0) = \mathcal{N}(x_t|x_0, \alpha_t^2 \mathrm{I})$, where $\alpha_0, \alpha_1, \ldots \alpha_T$ are all known/fixed.



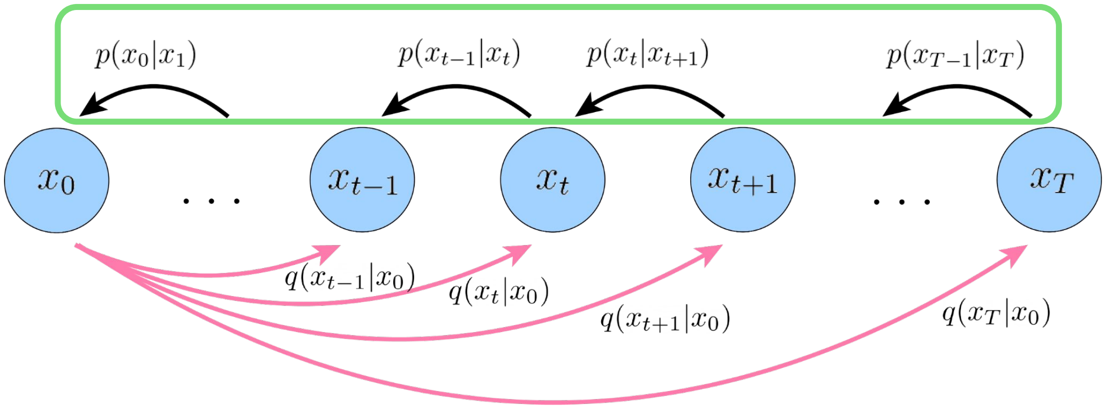
Hierarchical VAEs

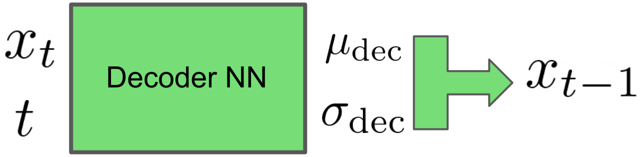
Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

*…what if we assume **all** dimensions are the same?*

…what if we assume all encoder transitions are known Gaussians centered around their previous input?





…then we can aggregate and simplify the distribution of each intermediate “latent”!

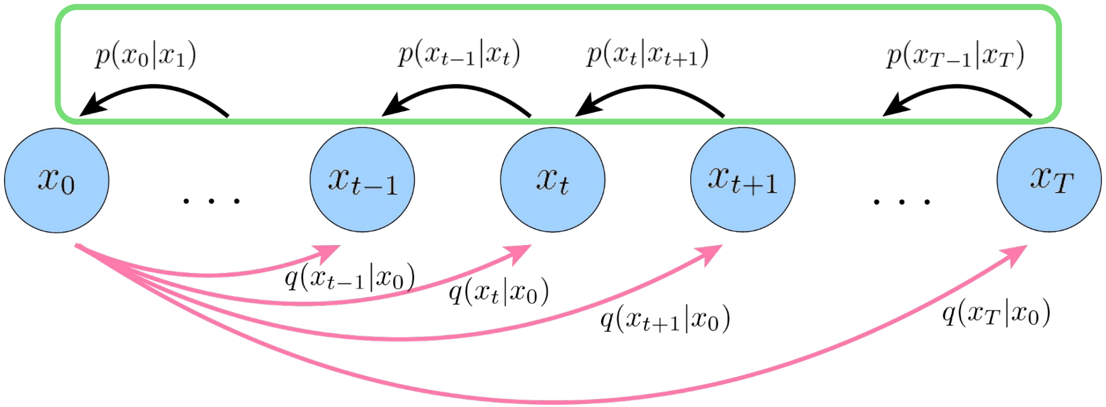
Diffusion Models

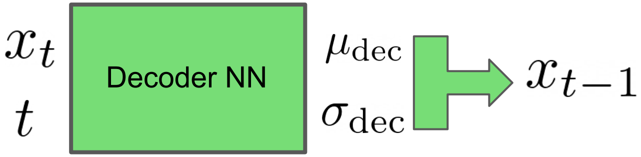
It turns out, that this is exactly what a diffusion model is!

- A Hierarchical VAE with these assumptions:

*…what if we assume **all** dimensions are the same?*

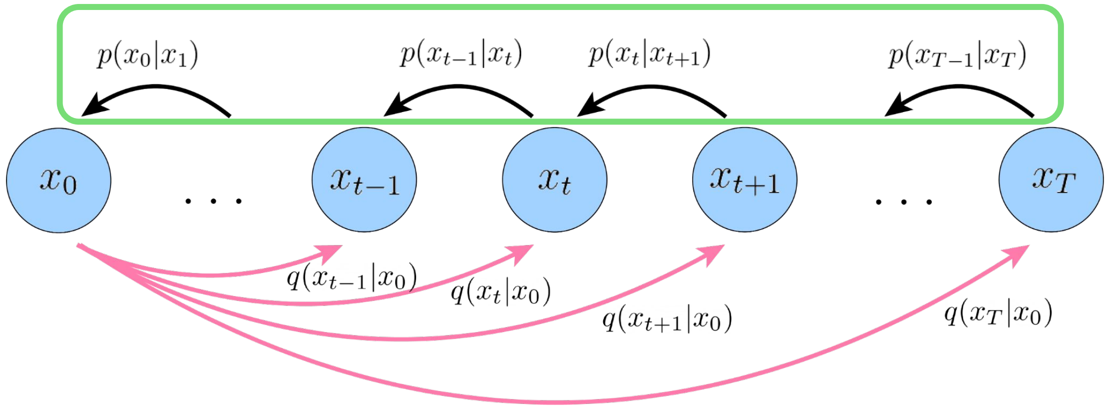
…what if we assume all encoder transitions are known Gaussians centered around their previous input?

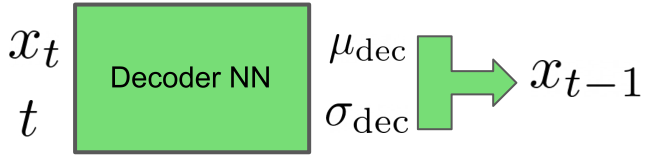




Diffusion Models

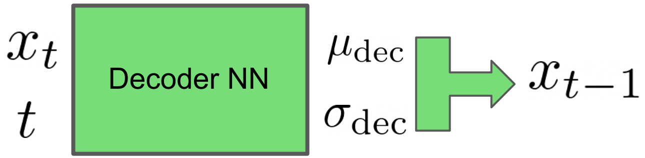
A diffusion model is implemented as a single neural network (the decoder)





Optimization?

We want to learn a denoising decoder:

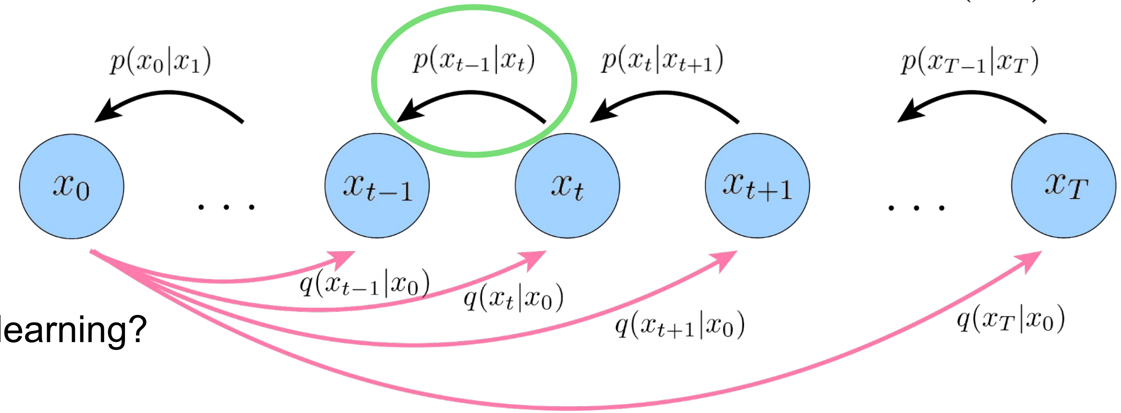


$$\hat{x}_{t-1} = \mu_{\text{dec}} + \sigma_{\text{dec}} * \epsilon$$

reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

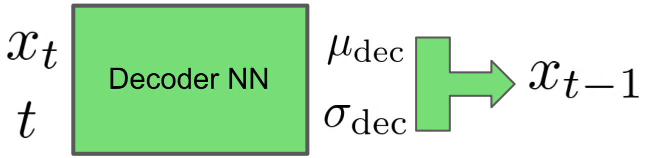
But what is the form of x_{t-1}?



…can we formulate this as supervised learning?

Optimization?

We want to learn a denoising decoder:



$$\hat{x}_{t-1} = \mu_{\text{dec}} + \sigma_{\text{dec}} * \epsilon$$

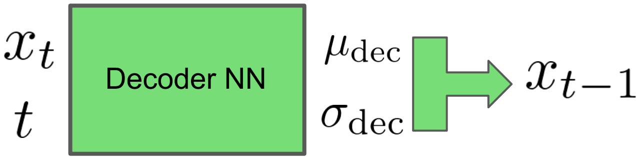
reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

But what is the form of x_{t-1}?

Optimization?

We want to learn a denoising decoder:



$$\hat{x}_{t-1} = \mu_{\text{dec}} + \sigma_{\text{dec}} * \epsilon$$

reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

But what is the form of x_{t-1}?

Recall that:

$$\boxed{\begin{gathered} q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}|x_0, \alpha_{t-1}^2 \mathrm{I}) \\ \therefore x_{t-1} = x_0 + \alpha_{t-1} * \epsilon \end{gathered}}$$

reparam. trick!

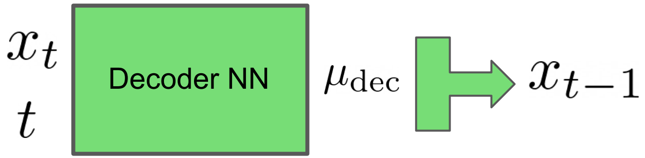
$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

Do we really need to predict σ_{dec}*?*

What is the ground truth signal for μ_{dec}*?*

Optimization?

We want to learn a denoising decoder:



$$\hat{x}_{t-1} = \hat{x}_0 + \alpha_{t-1} * \epsilon$$

reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

But what is the form of x_{t-1}?

Recall that:

$$q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}|x_0, \alpha_{t-1}^2 \mathrm{I})$$

$$\therefore x_{t-1} = x_0 + \alpha_{t-1} * \epsilon$$

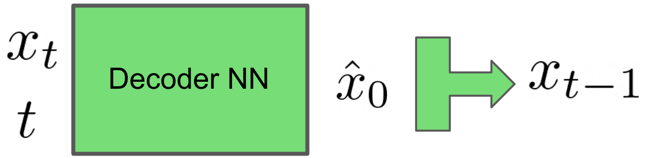
reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

Do we really need to predict σ_{dec}? *What is the ground truth signal for μ_{dec}?*

Optimization?

We want to learn a denoising decoder:



$$\hat{x}_{t-1} = \hat{x}_0 + \alpha_{t-1} * \epsilon$$

reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

But what is the form of x_{t-1}?

Recall that:

$$q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}|x_0, \alpha_{t-1}^2 \mathrm{I})$$

$$\therefore x_{t-1} = x_0 + \alpha_{t-1} * \epsilon$$

reparam. trick!

$\epsilon \sim \mathcal{N}(0, \mathrm{I})$

Do we really need to predict σ_{dec}?

What is the ground truth signal for μ_{dec}?

Optimization?

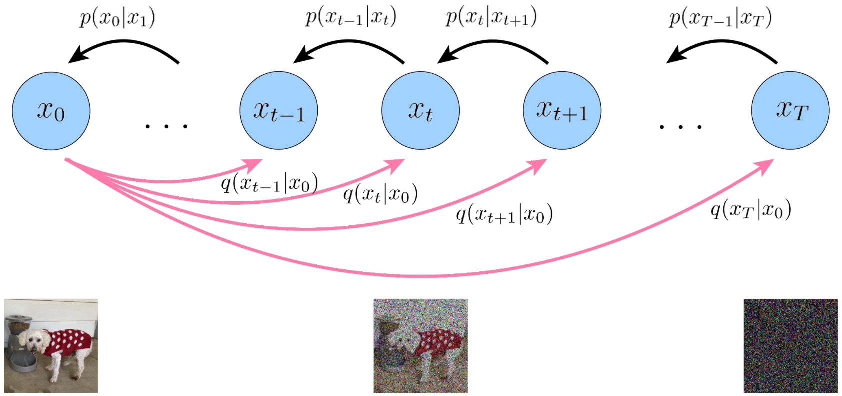
We want to learn a denoising decoder:

So in the end, a diffusion model is simply *one* Neural Network that predicts a clean image x_0 from arbitrary noisified image x_t.

Diffusion Models: A Summary

A Diffusion Model is:

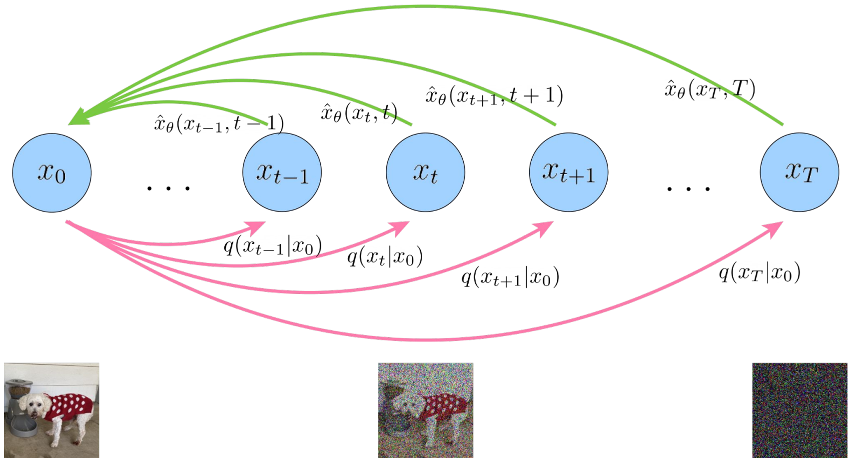
- One NN that predicts a clean image from a noisy version of the image



Diffusion Models: A Summary

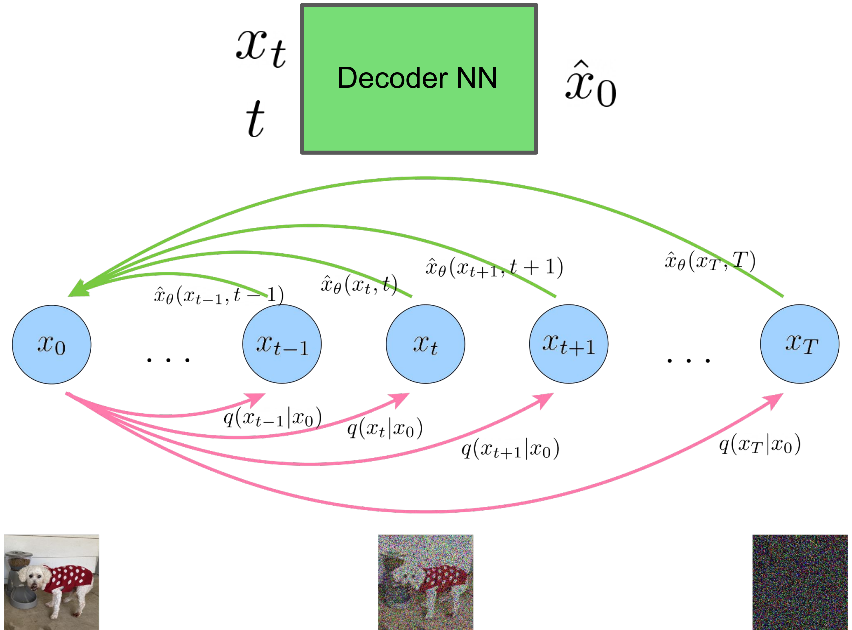
A Diffusion Model is:

- One NN that predicts a clean image from a noisy version of the image

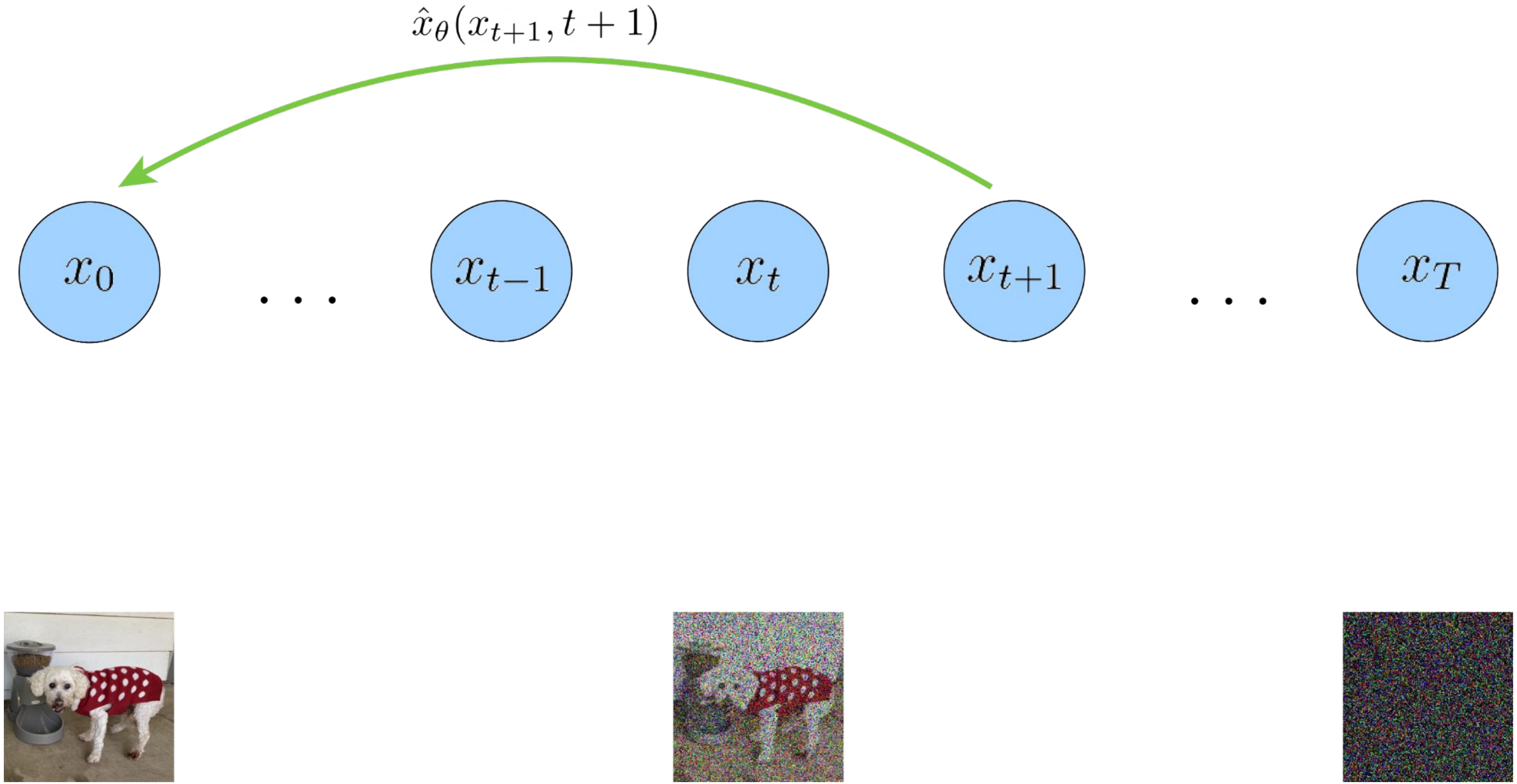


Diffusion Models: A Summary

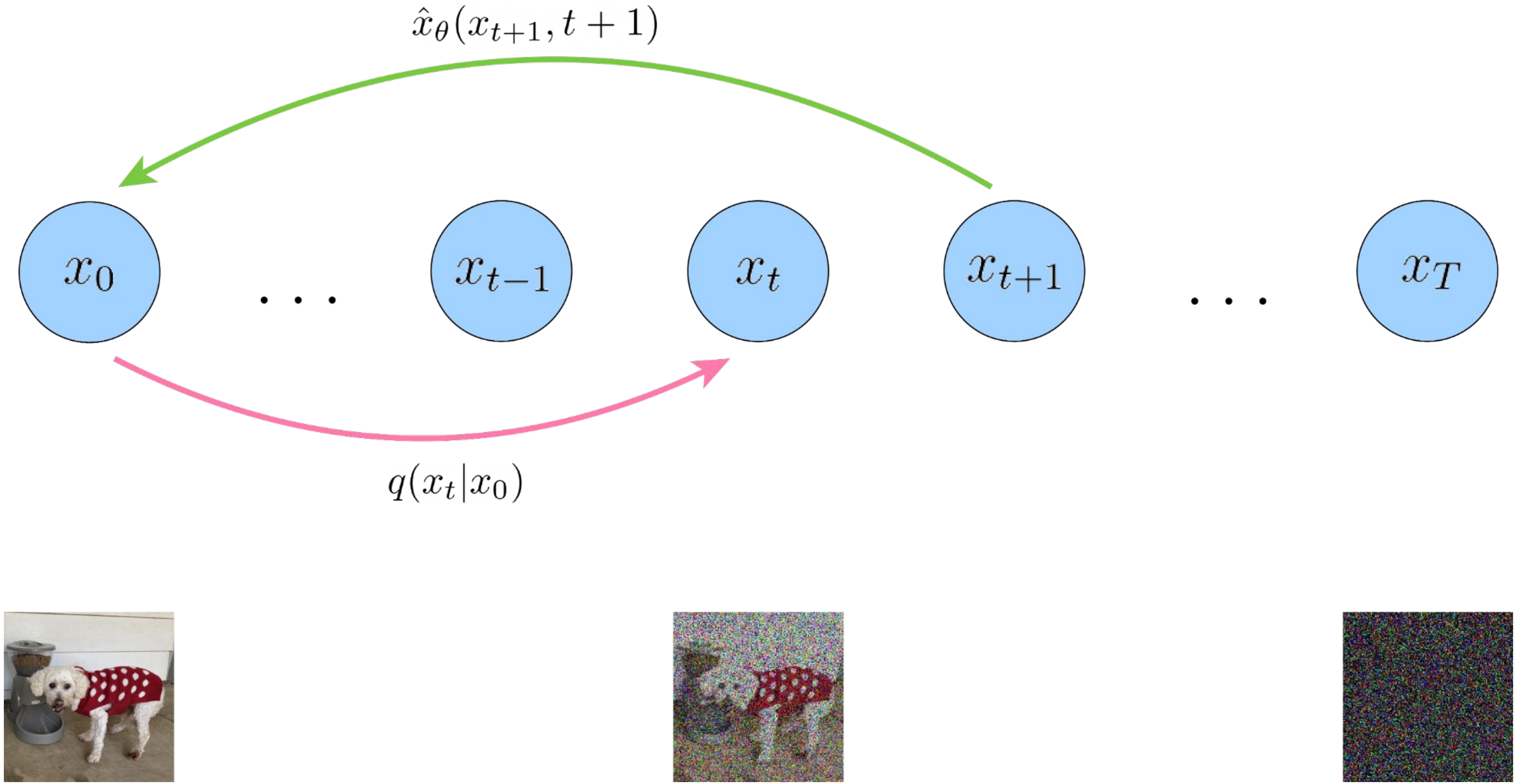
How do we perform sampling?



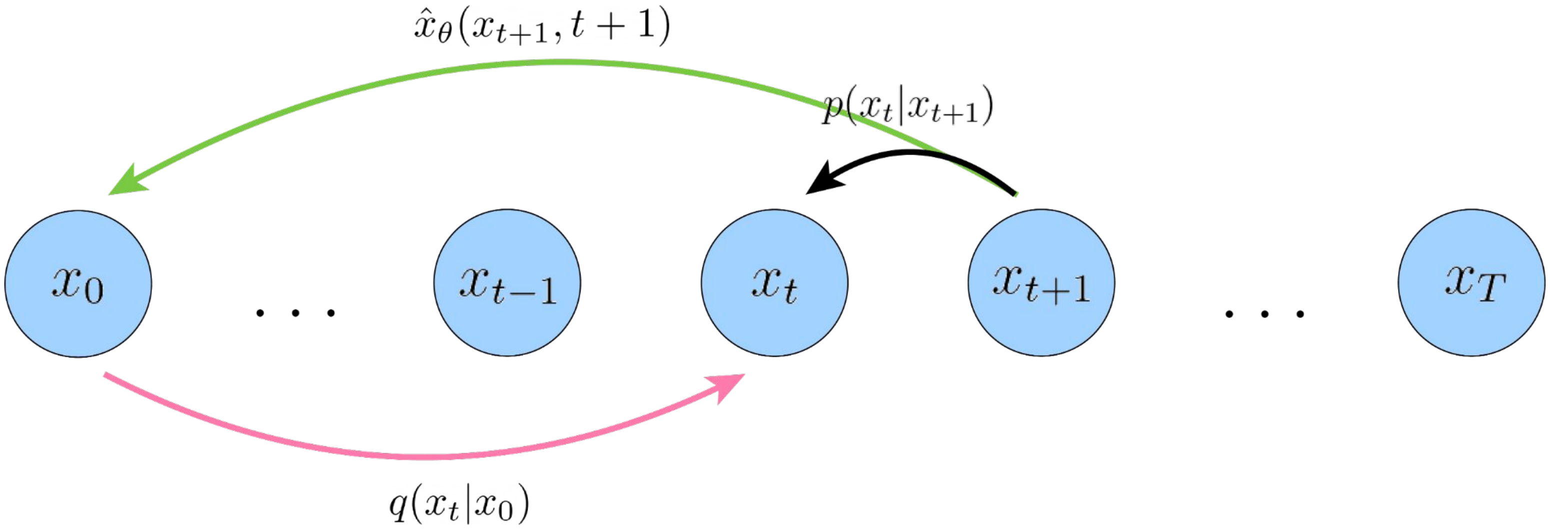
Sampling

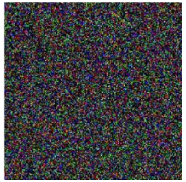


Sampling

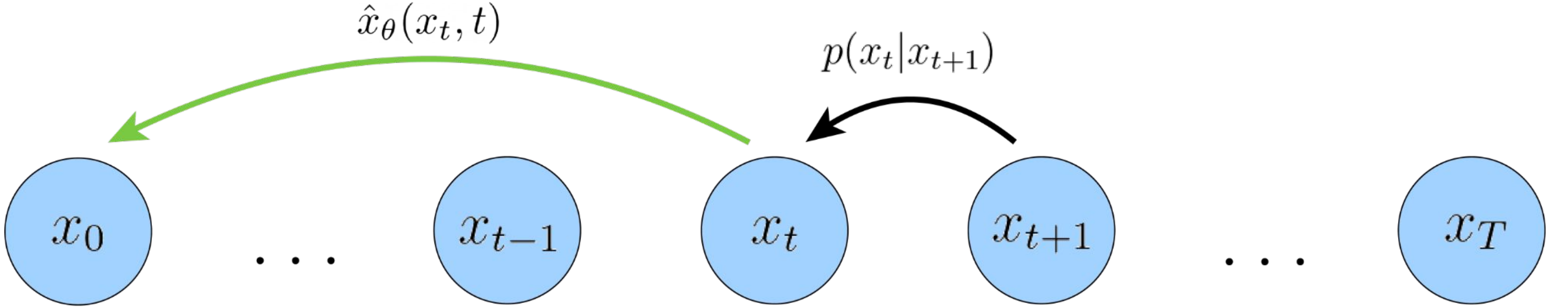


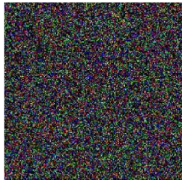
Sampling



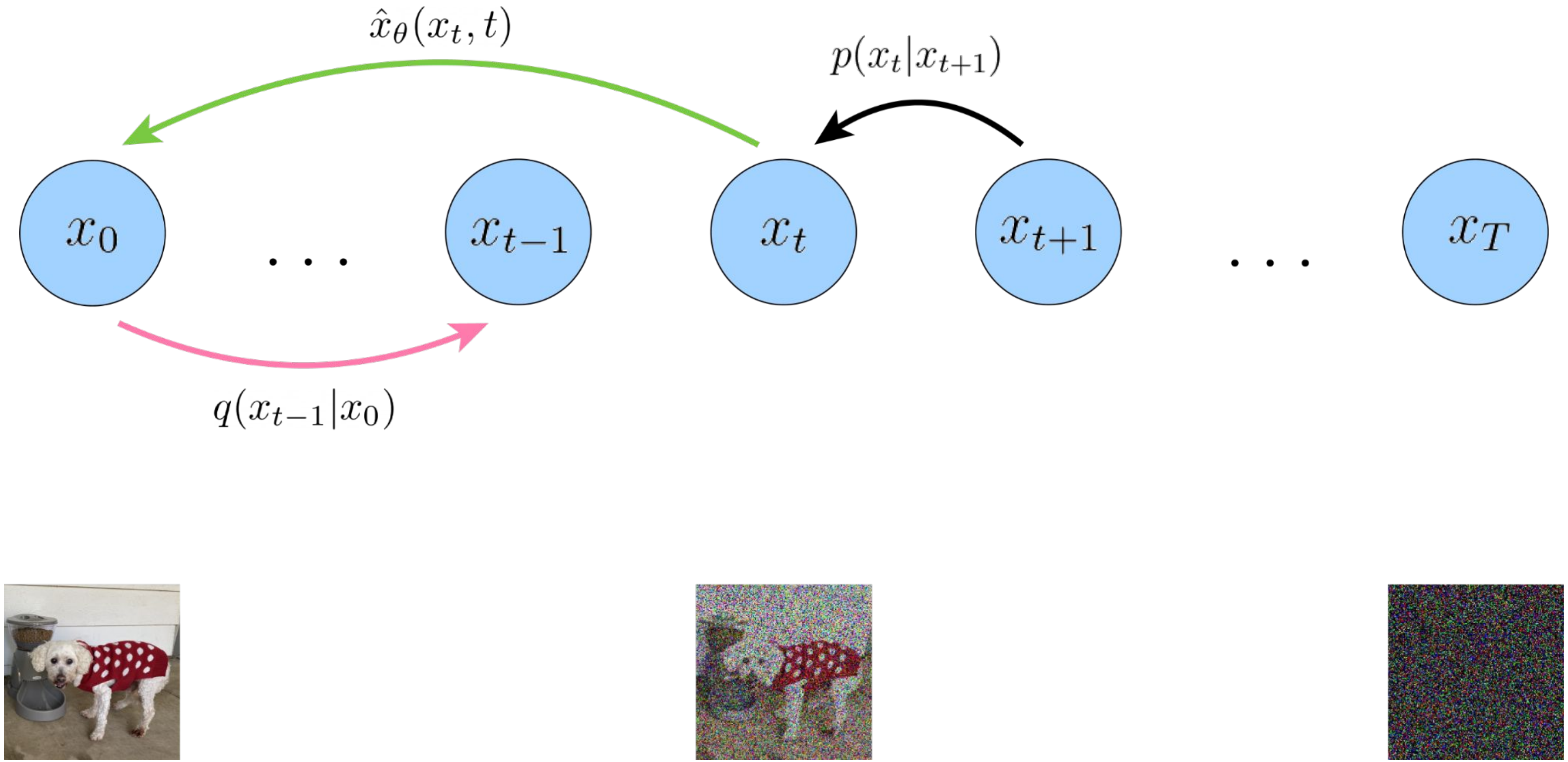


Sampling

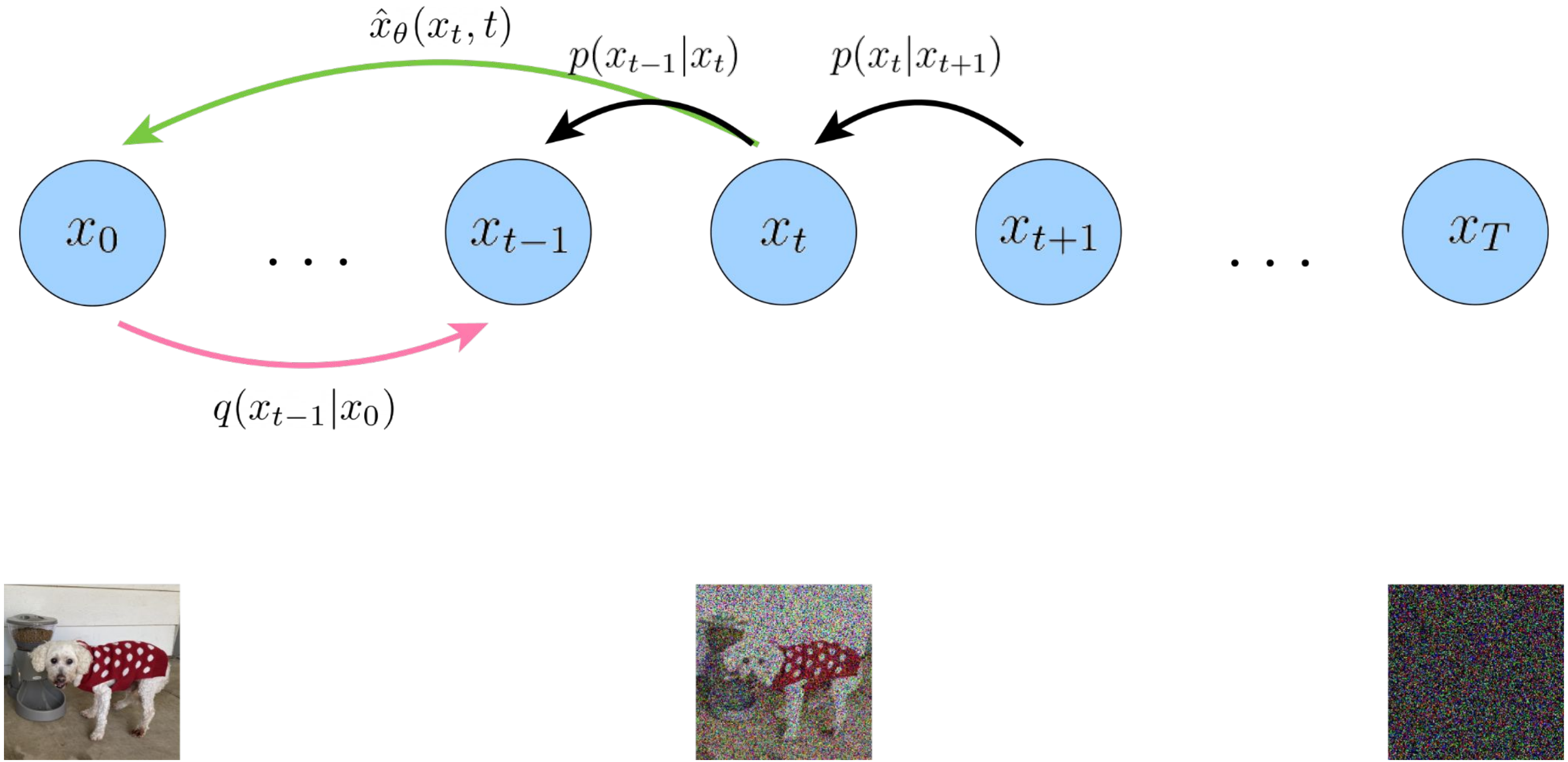




Sampling

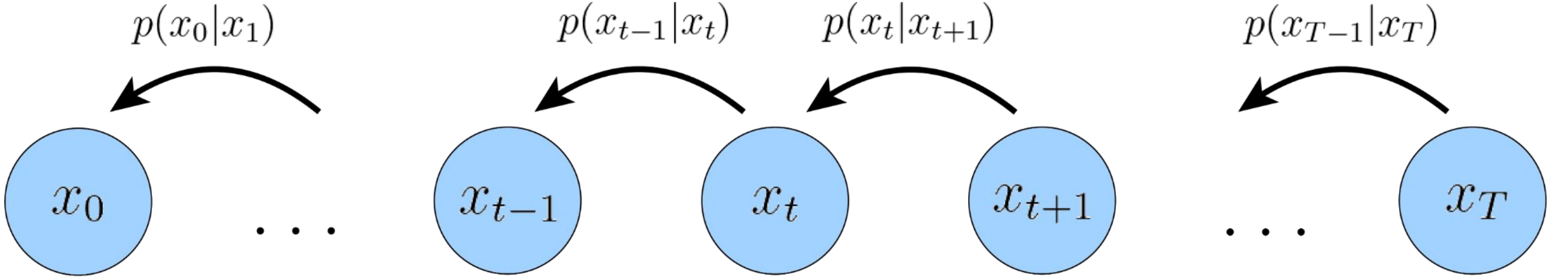


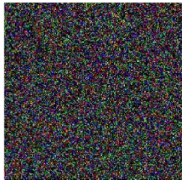
Sampling



$\hat{x}_\theta(x_t, t)$
$p(x_{t-1}|x_t)$
$p(x_t|x_{t+1})$
x_0
$\ldots$
x_{t-1}
x_t
x_{t+1}
$\ldots$
x_T
$q(x_{t-1}|x_0)$

Sampling





Pseudocode

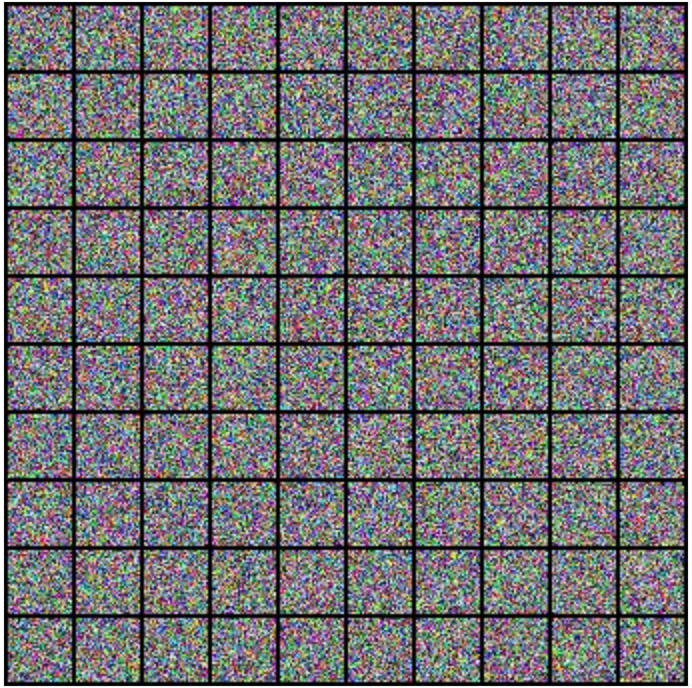
Algorithm 1 Training

repeat
  $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
  $t \sim \texttt{Uniform}(1, ..., T)$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta ||\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_0 + \alpha_t \boldsymbol{\epsilon}, t)||^2$
until converged

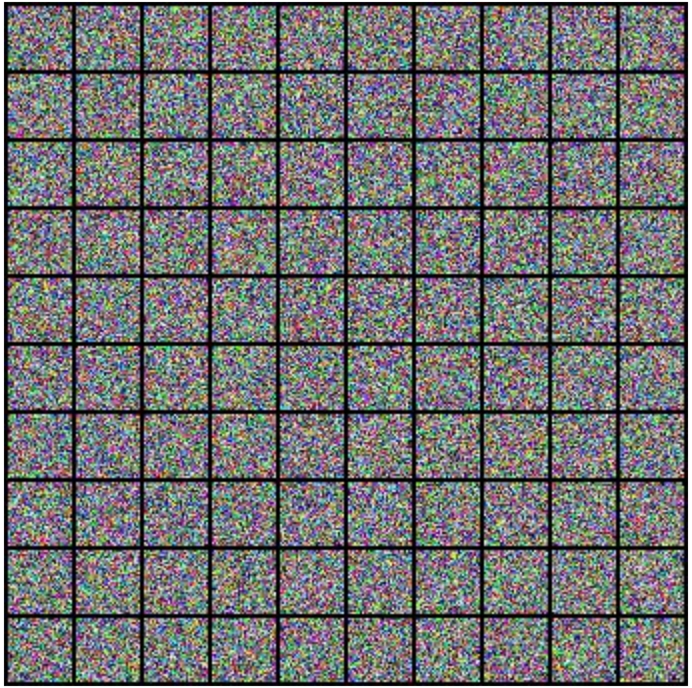
Algorithm 2 Sampling

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, ..., 1$:
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{\epsilon} = 0$
  $\boldsymbol{x}_{t-1} = \hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t) + \alpha_{t-1}\epsilon$
end for
return $\boldsymbol{x}_0$

Examples!



Celeb-A



CIFAR-10

source: [Generative Modeling by Estimating Gradients of the Data Distribution]

Examples!

1024x1024 samples

source: *Generative Modeling by Estimating Gradients of the Data Distribution*

Three Different Interpretations

It turns out, training a DiffModel can be done using three different interpretations:

- Predicting original image 🖼️ (we just did this)
- Predicting noise 🔊 (coming up!)
- Predicting score function 💯 (coming up!)

Diffusion Models as a Noise Predictor 🔊

Recall that our objective is to predict $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \approx \boldsymbol{x}_0$

Image 🖼️ and Noise 🔊? They are the same!

What does it mean intuitively?

For arbitrary $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$, we can rewrite it as $\boldsymbol{x}_t = \boldsymbol{x}_0 + \alpha_t \boldsymbol{\epsilon}_0$

Predicting $\boldsymbol{x}_0$ *determines* $\boldsymbol{\epsilon}_0$ and vice-versa, since they sum to the same thing!

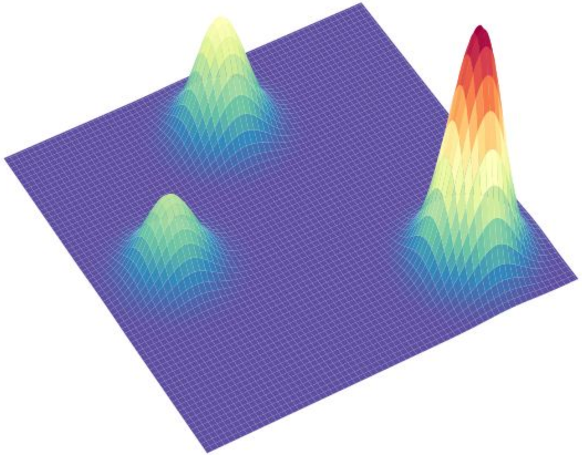
$\boldsymbol{x}_t \sim q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$ $\boldsymbol{x}_0$ $\boldsymbol{\epsilon}_0$

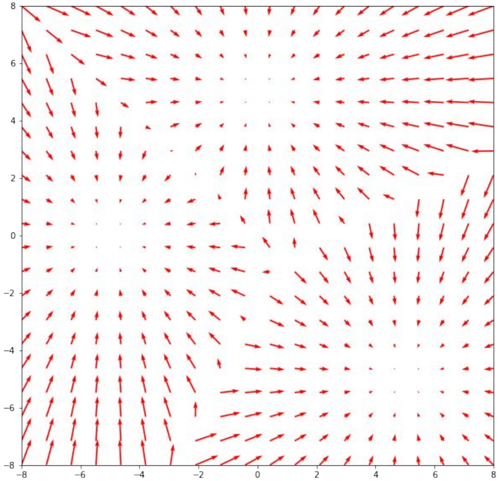
Score Functions 💯

What are score functions?

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$$

Intuitively, they describe how to move in data space to improve the (log) likelihood.





Tweedie's Formula

Mathematically, for a Gaussian variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$Tweedie's formula states:

$$\mathbb{E}\left[\boldsymbol{\mu}_z \mid \boldsymbol{z}\right] = \boldsymbol{z} + \boldsymbol{\Sigma}_z \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z})$$

Then, since we have previously shown that:

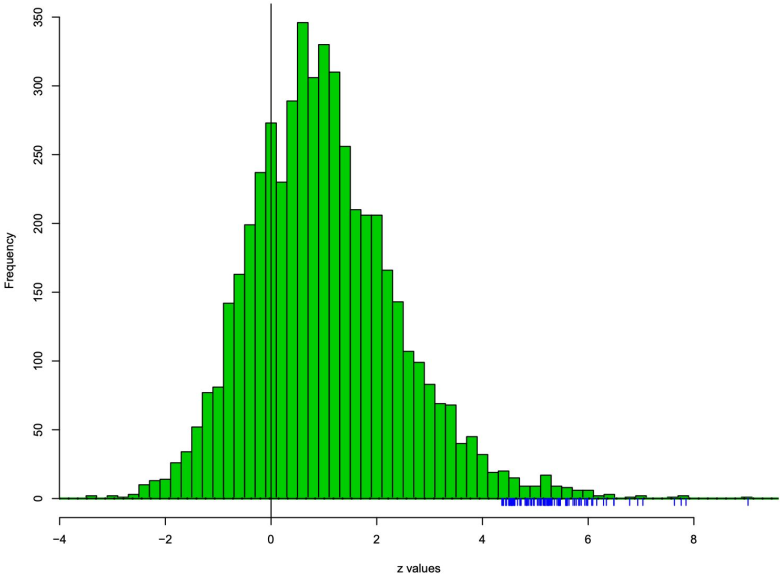
$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}_0, \alpha_t^2\mathbf{I})$$

By Tweedie's Formula, we derive:

$$\mathbb{E}[\boldsymbol{\mu}_{\boldsymbol{x}_t} \mid \boldsymbol{x}_t] = \boldsymbol{x}_t + \alpha_t^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$

The best estimate for the true mean is $\boldsymbol{\mu}_{\boldsymbol{x}_t} = \boldsymbol{x}_0$

$$\boldsymbol{x}_0 = \boldsymbol{x}_t + \alpha_t^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$



Tweedie's Formula

There exists a mathematical formula that states that:

$$\boldsymbol{x}_0 \approx \boldsymbol{x}_t + \alpha_t^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$

Due to the fact that the distribution is Gaussian:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{x}_0, \alpha_t^2 \mathbf{I})$$

Diffusion Models as a Score Predictor 💯

Recall that our objective is to predict $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \approx \boldsymbol{x}_0$

Score 💯 and Noise 🔊?

There is a relationship between the score and the noise, which we can derive by equating Tweedie's formula with the Reparameterization Trick.

$$\begin{aligned}\boldsymbol{x}_0 = \boldsymbol{x}_t + \alpha_t^2 \nabla \log p(\boldsymbol{x}_t) &= \boldsymbol{x}_t - \alpha_t \boldsymbol{\epsilon}_0 \\ \therefore\ \alpha_t^2 \nabla \log p(\boldsymbol{x}_t) &= -\alpha_t \boldsymbol{\epsilon}_0 \\ \nabla \log p(\boldsymbol{x}_t) &= -\frac{1}{\alpha_t} \boldsymbol{\epsilon}_0\end{aligned}$$

Intuitively, the direction to move in data space towards a natural image is the negative noise term that was added.

Three Different Interpretations

It turns out, training a DiffModel can be implemented as a neural net that:

- 🖼️ Predicts original image $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \approx \boldsymbol{x}_0$
- 🔊 Predicts noise epsilon $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \approx \boldsymbol{\epsilon}_0$
- 💯 Predicts score function $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \approx \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$

A Summary

We have learned that a diffusion model is simply one neural network that predicts a clean image from a noisy image.

Objective: $\arg\min_{\boldsymbol{\theta}} \; ||x_0 - \hat{x}_\theta(x_t, t)||^2$

Sampling: