CSCI 1470

Eric Ewing

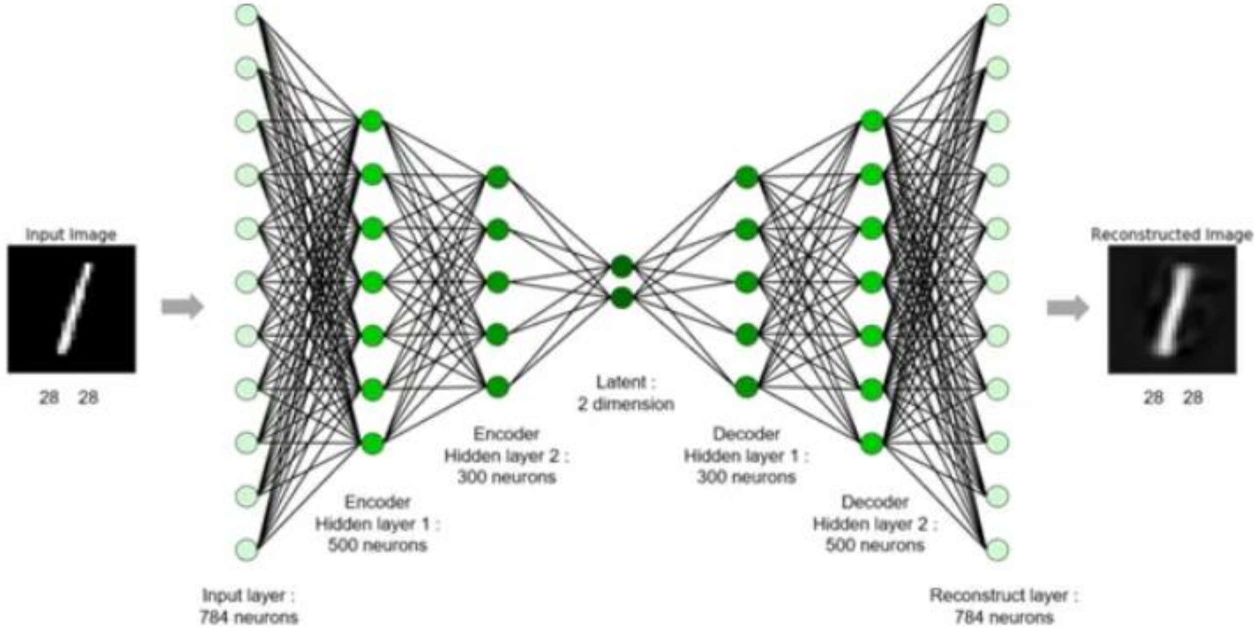
Monday, 4/7/25

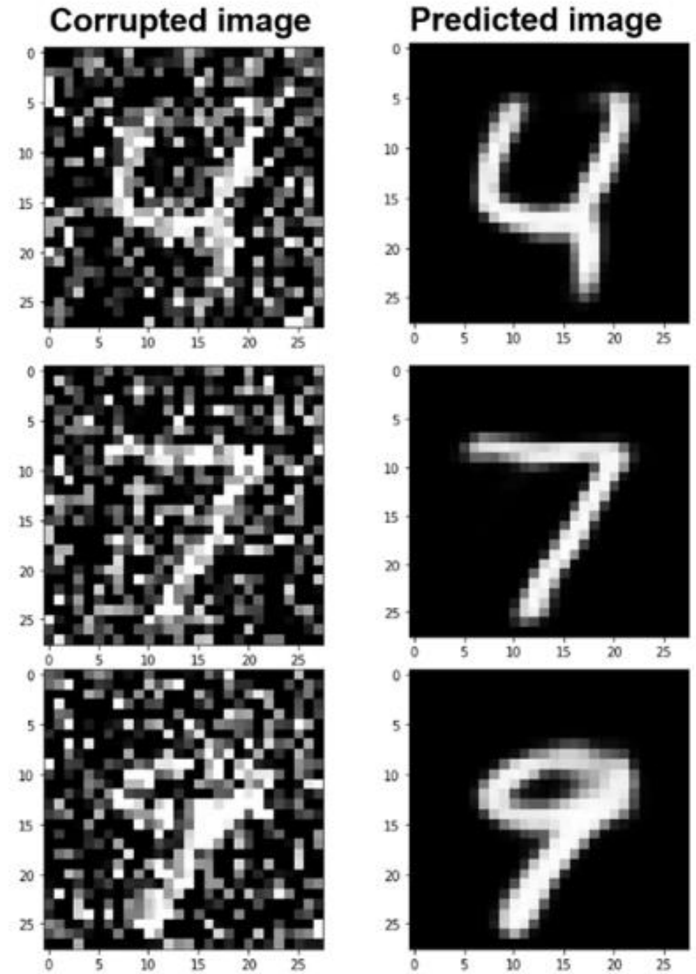
Deep Learning

Day 25: Diffusion Models

ChatGPT: “Generate an image of deep space” 4/7/25

Denoising Auto Encoders (DAEs)

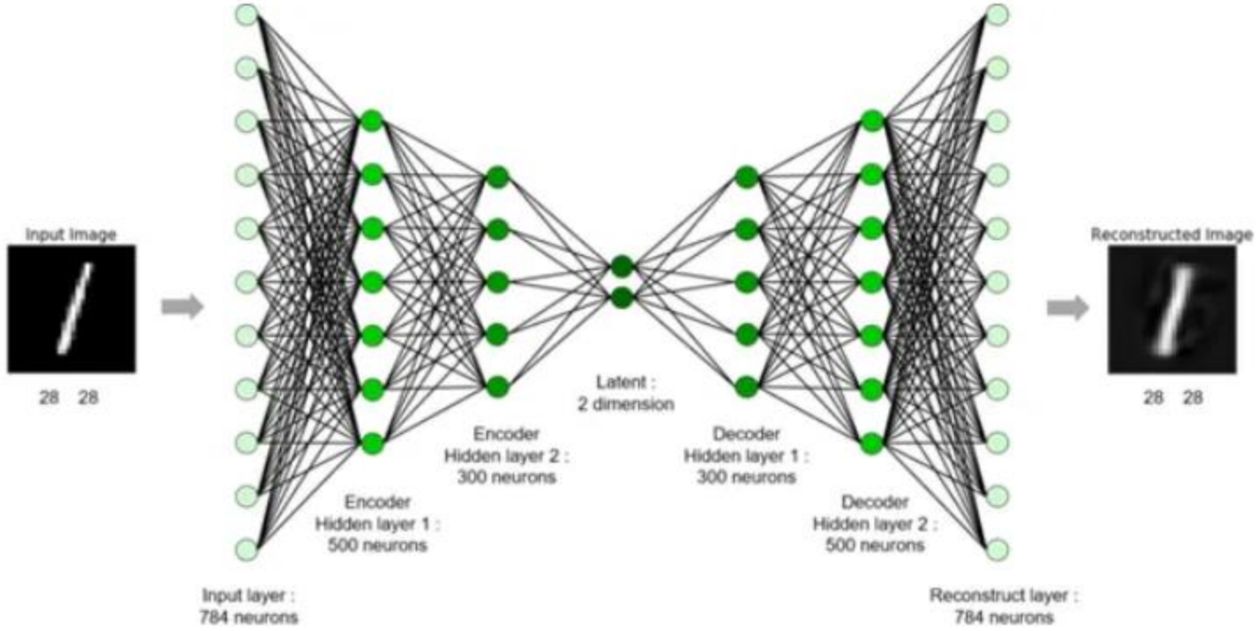


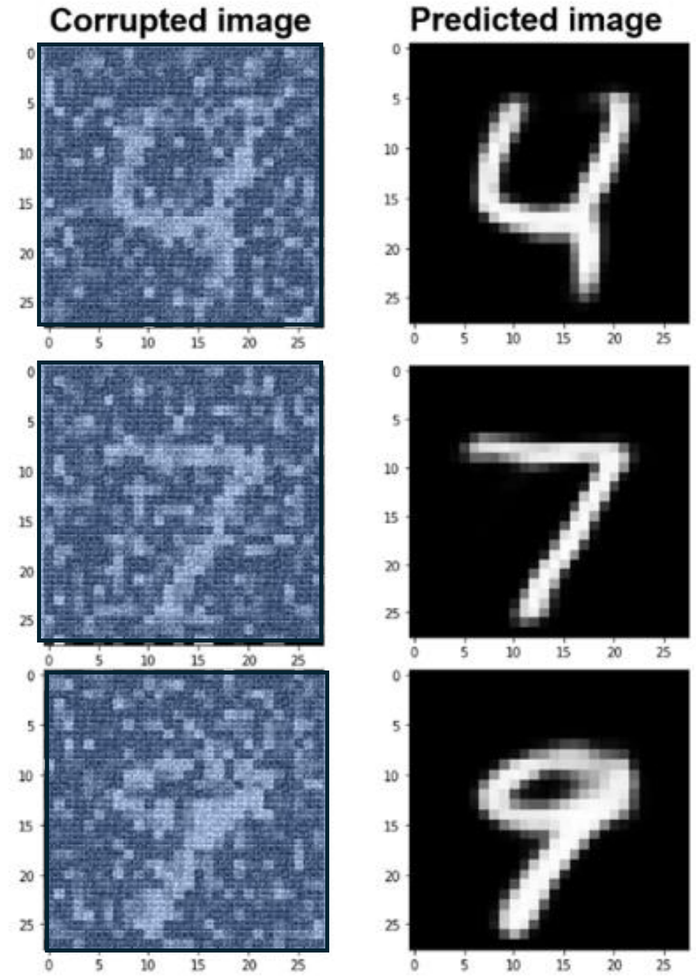


https://www.omdena.com/blog/denoising-autoencoders

Denoising Auto Encoders (DAEs)

What if our images were even noisier?



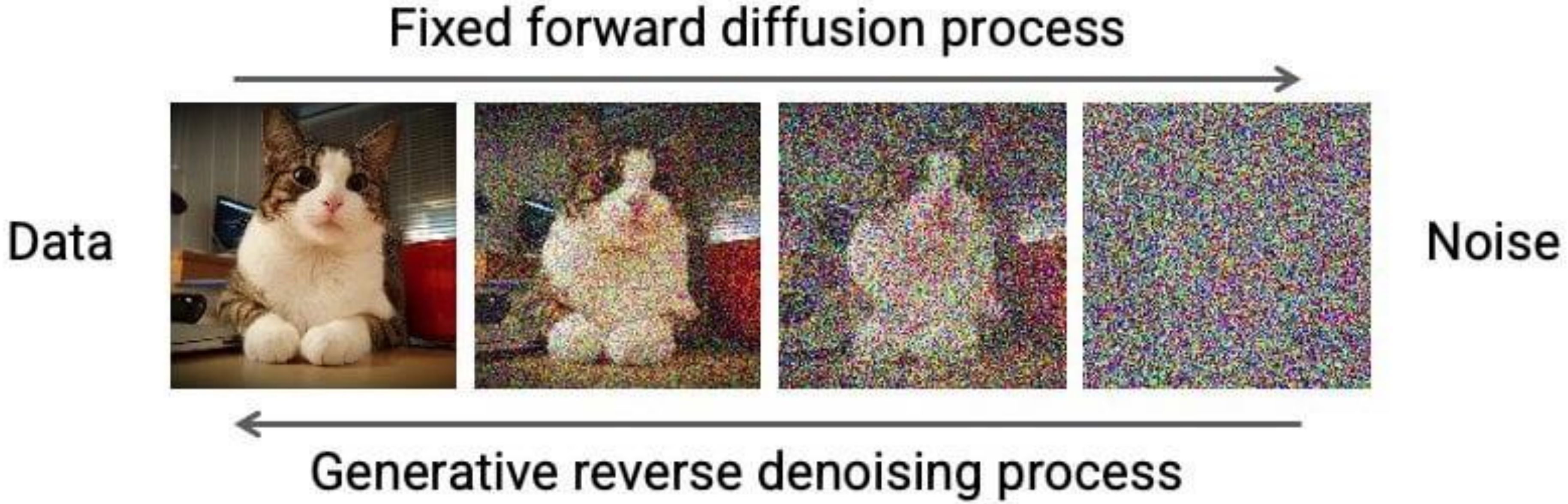


https://www.omdena.com/blog/denoising-autoencoders

Denoising Auto Encoders

Denoising all the noise in one step may be too hard to learn.

What if we added and removed noise incrementally?

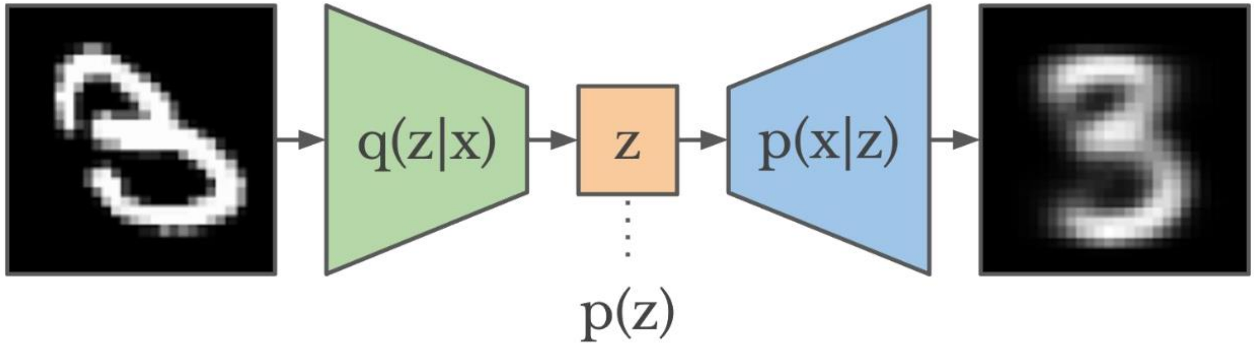


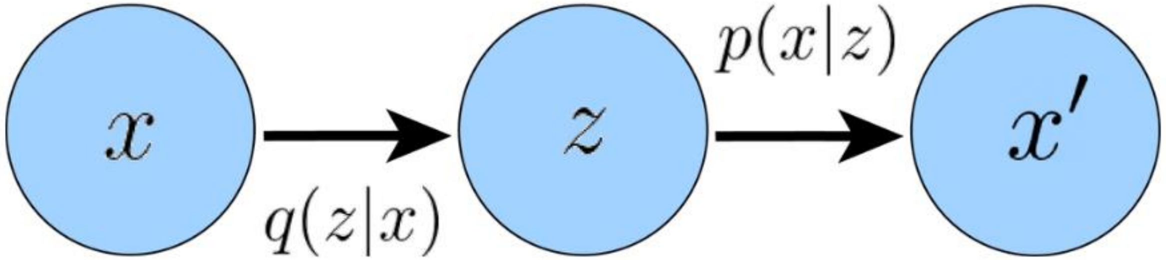
Variational Autoencoders

We can represent a VAE as a probabilistic graphical model

Encoder generates probability distribution $q(z|x)$

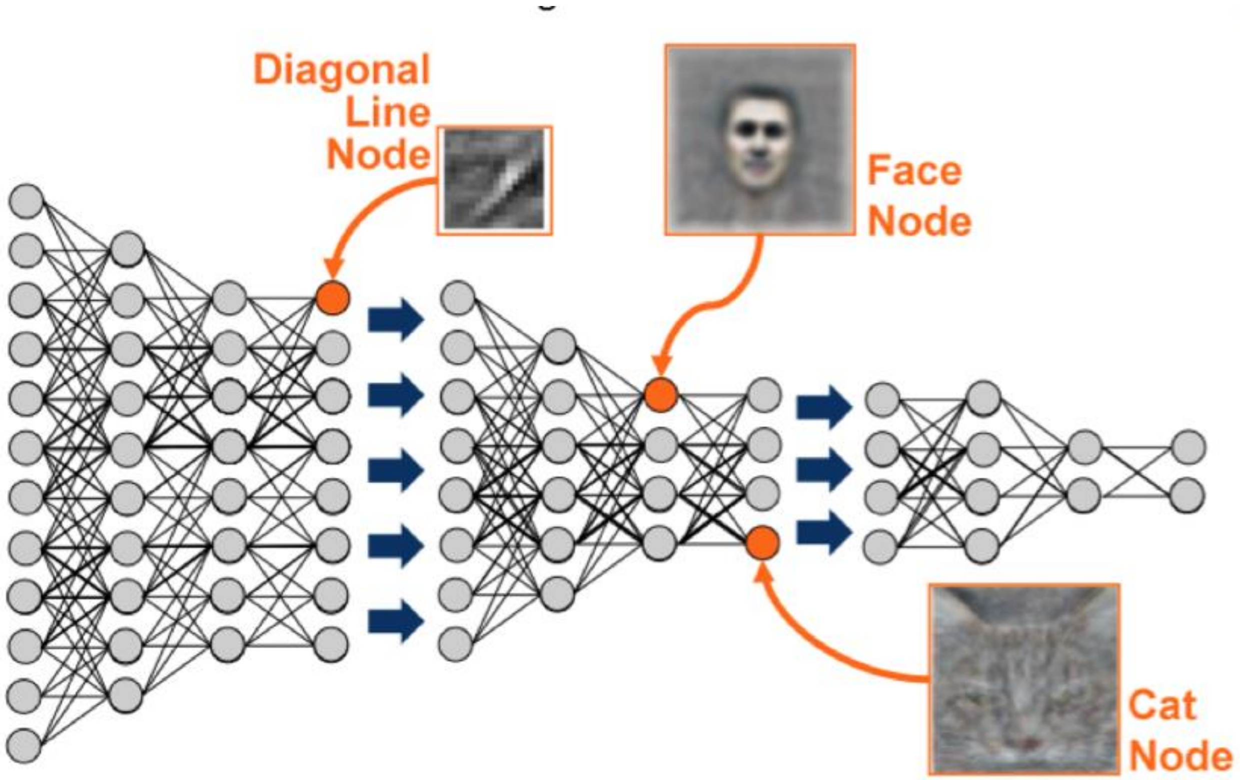
Decoder estimates $p(x|z)$





Hierarchical Features

Each intermediate layer in a neural network can be seen as learning a set of **intermediate features** based on the previous **intermediate outputs**.

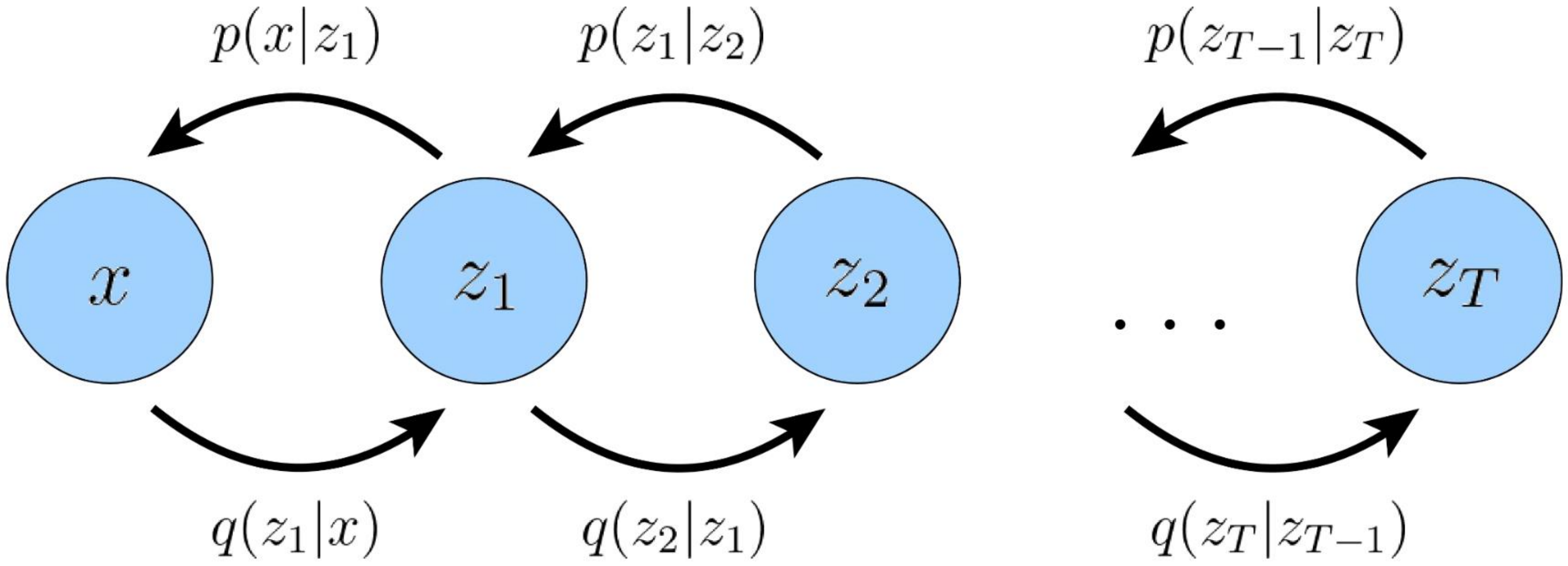


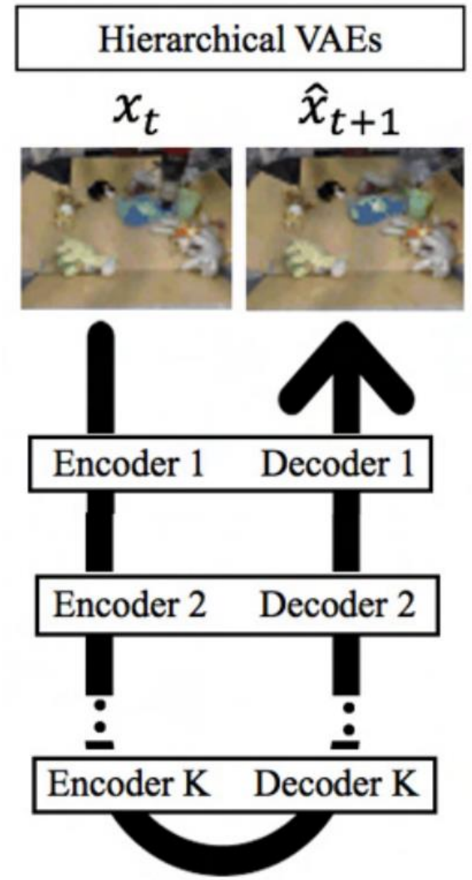
Source: https://ermongroup.github.io/blog/hierarchy/

Hierarchical VAEs

Idea: train K pairs of encoders and decoders

Each decoder is trained to reverse the associated encoder's operations





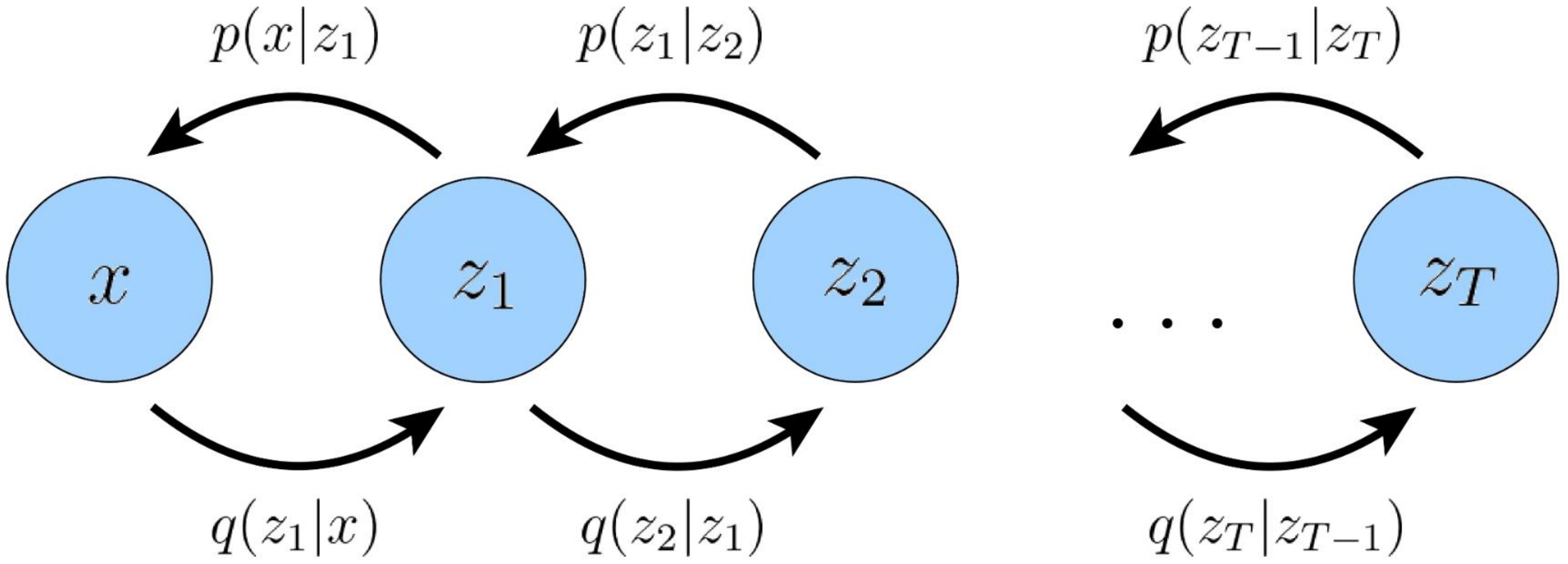
Hierarchical VAEs

How many encoders and decoders do we need to learn?

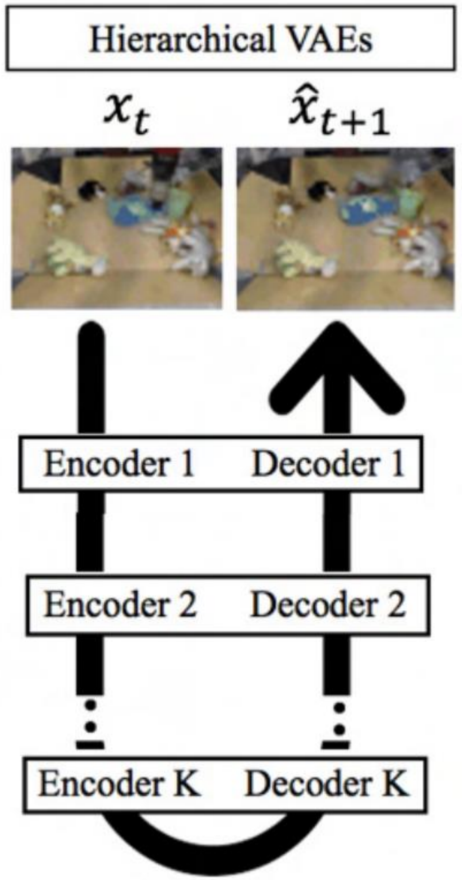
Idea: train K pairs of encoders and decoders

Each decoder is trained to reverse the associated encoder's operations

Decoders (reverse process)



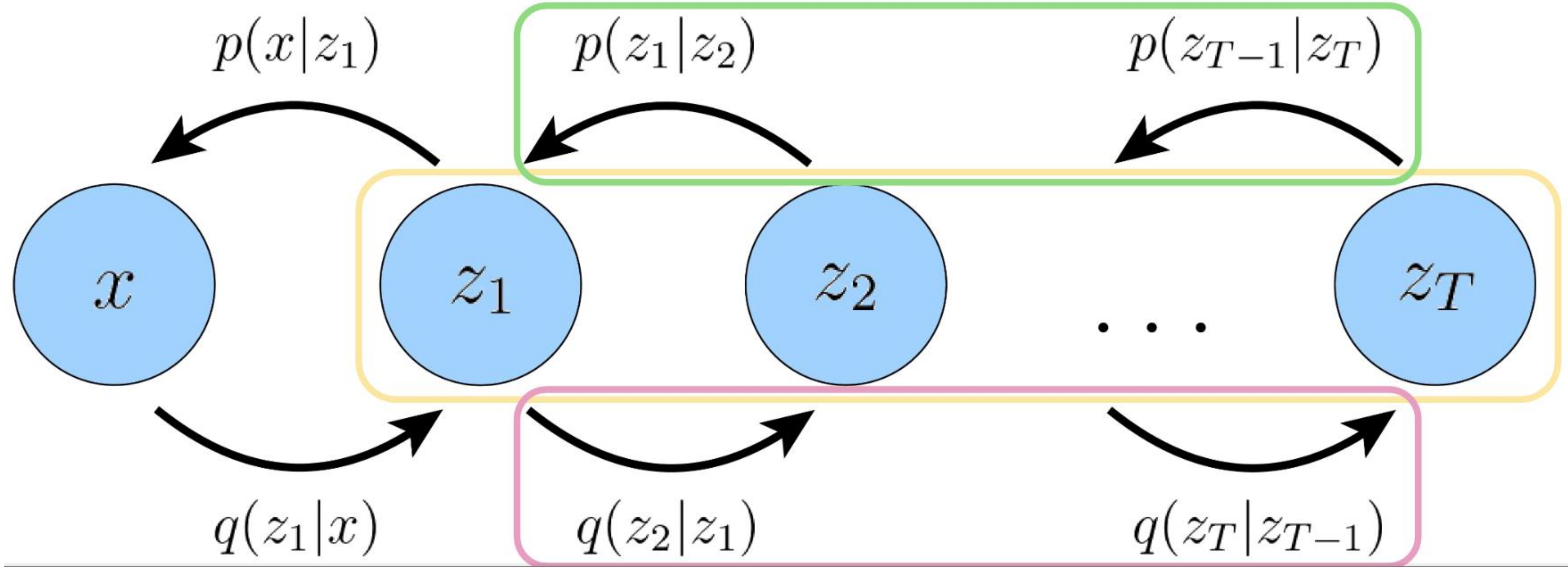
Encoders (forward process)



Hierarchical VAEs

What if all latent variables z have the same size?

Most encoders and decoders have the same input/output size

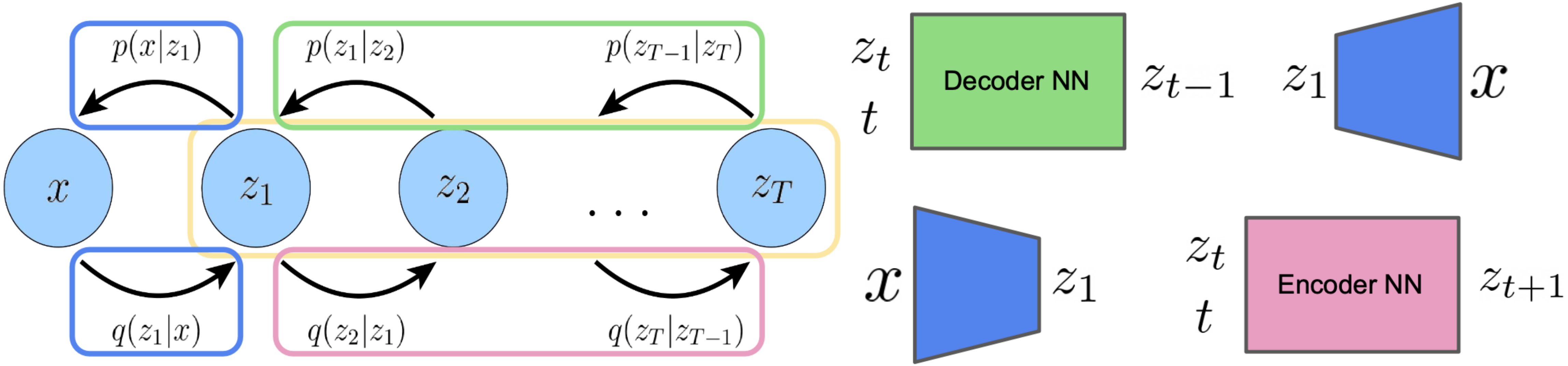


Hierarchical VAEs

But what if ***everything*** had the same dimensions

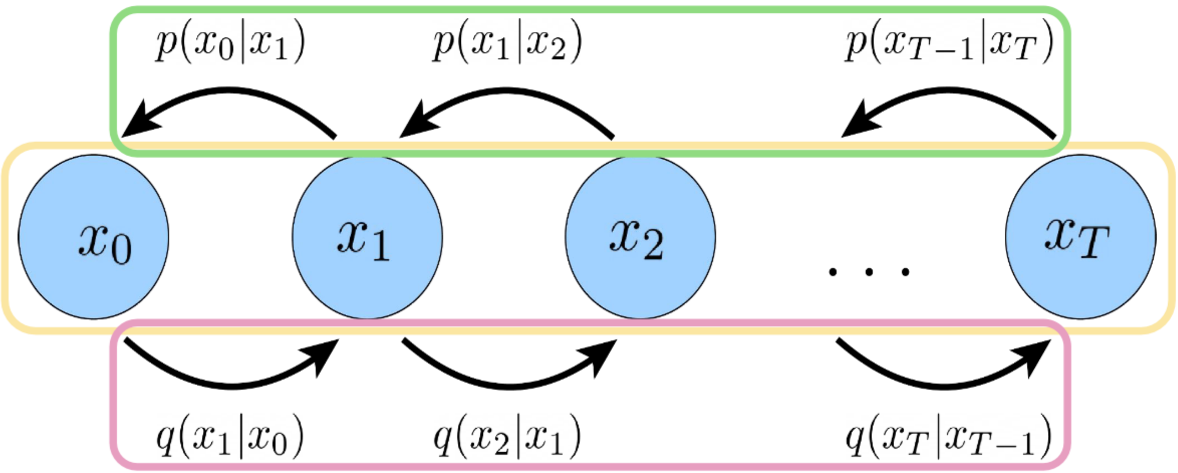
What if all latent variables z have the same size?

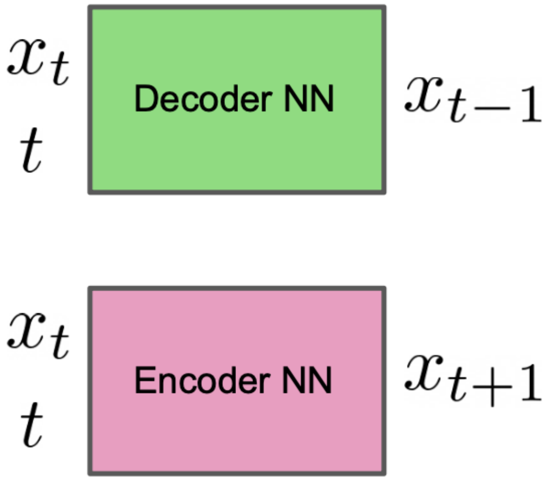
Need special **first encoder**, special **last decoder** to go from embedding size to original input size



Hierarchical VAEs

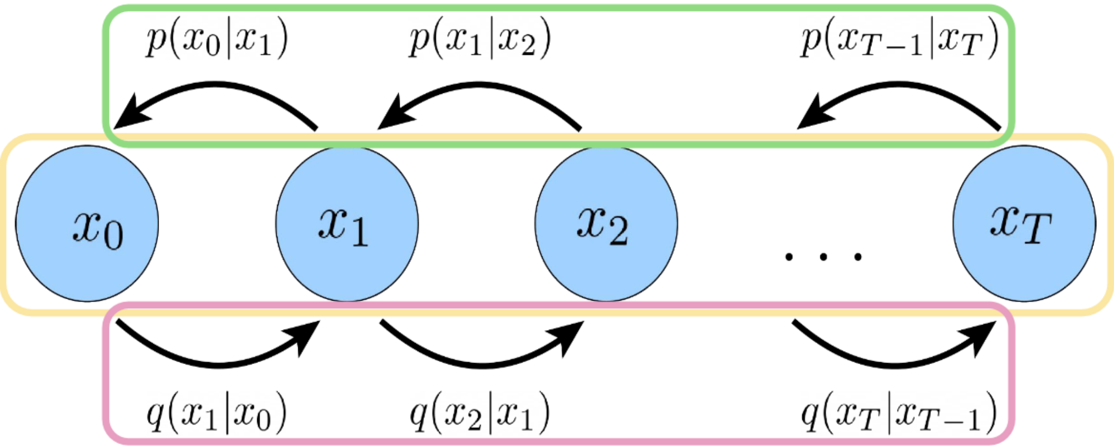
A hierarchical VAE that keeps the dimensions the same can use the same decoder and encoder for all steps

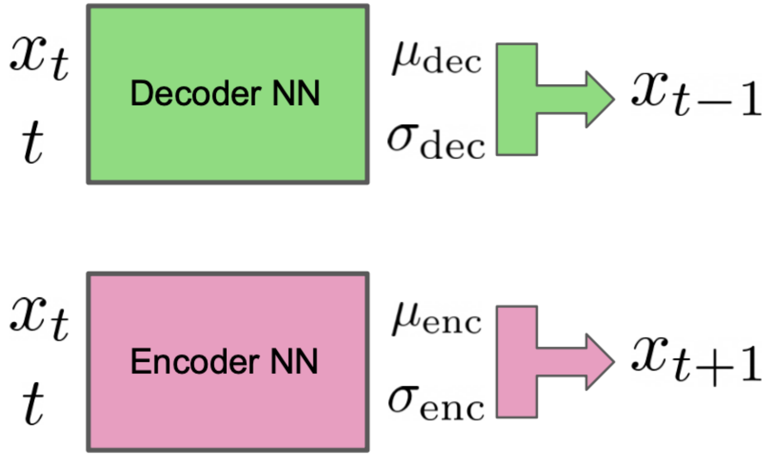




Hierarchical VAEs

A hierarchical VAE that keeps the dimensions the same can use the same decoder and encoder for all steps





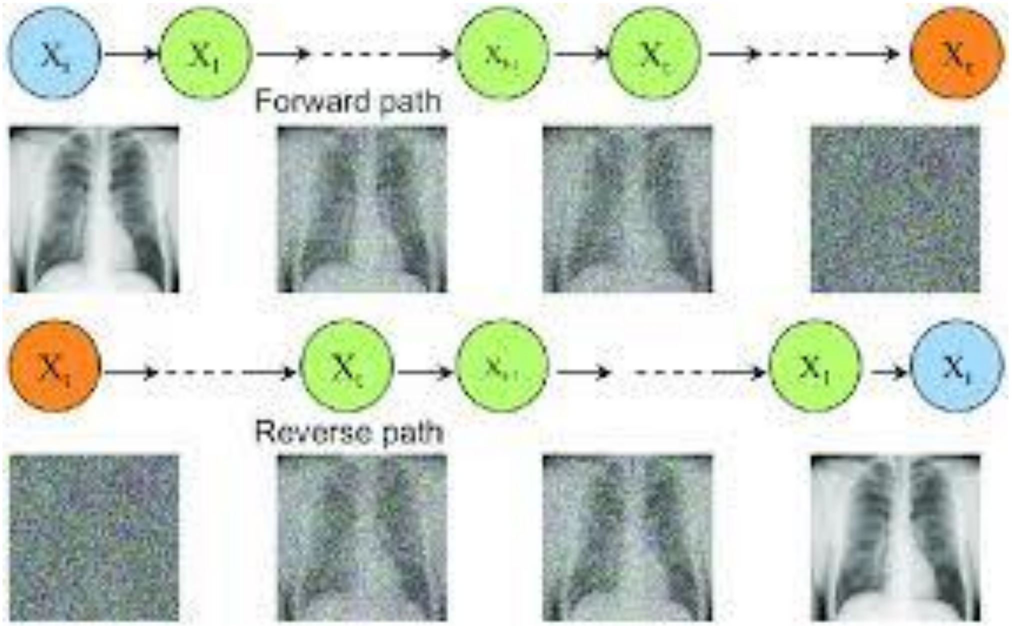
Denoising VAEs

If the size of the encoding in a VAE is the same size as the input, the model is no longer doing dimensionality reduction…

So why would we want this?

The forward process adds noise
The reverse path reverses this process

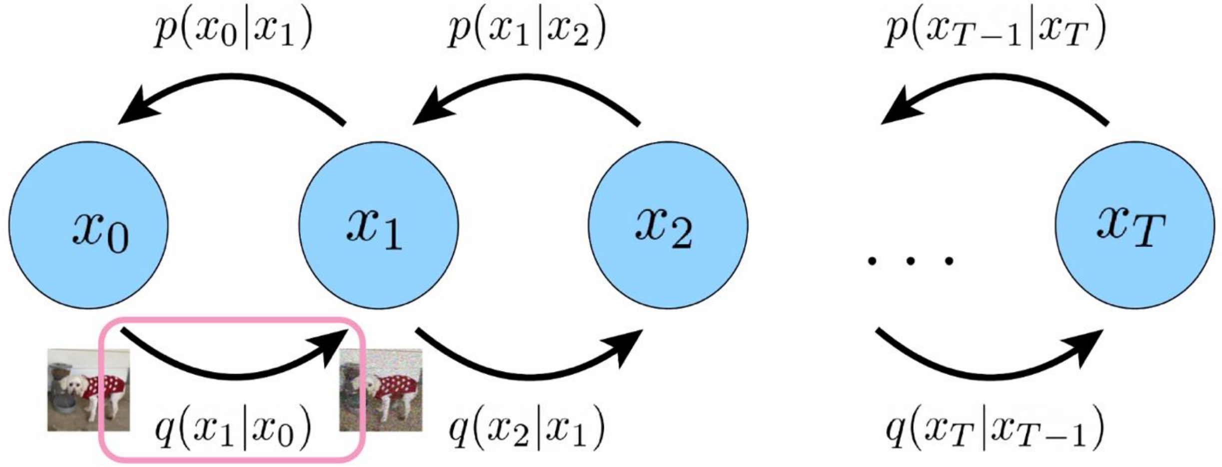
But we don’t need to learn the encoders, adding noise isn’t a learned process!



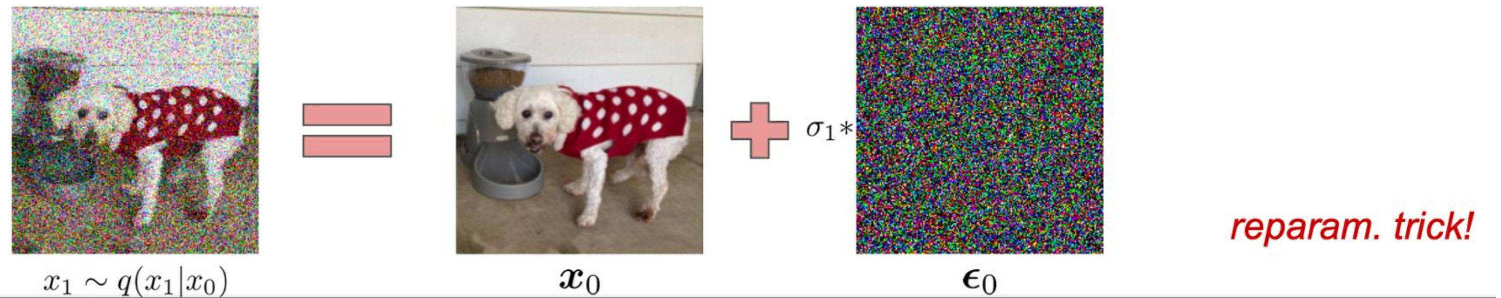
Source: https://www.researchgate.net/figure/Schematic-of-the-diffusion-model-training-process_fig1_383920783

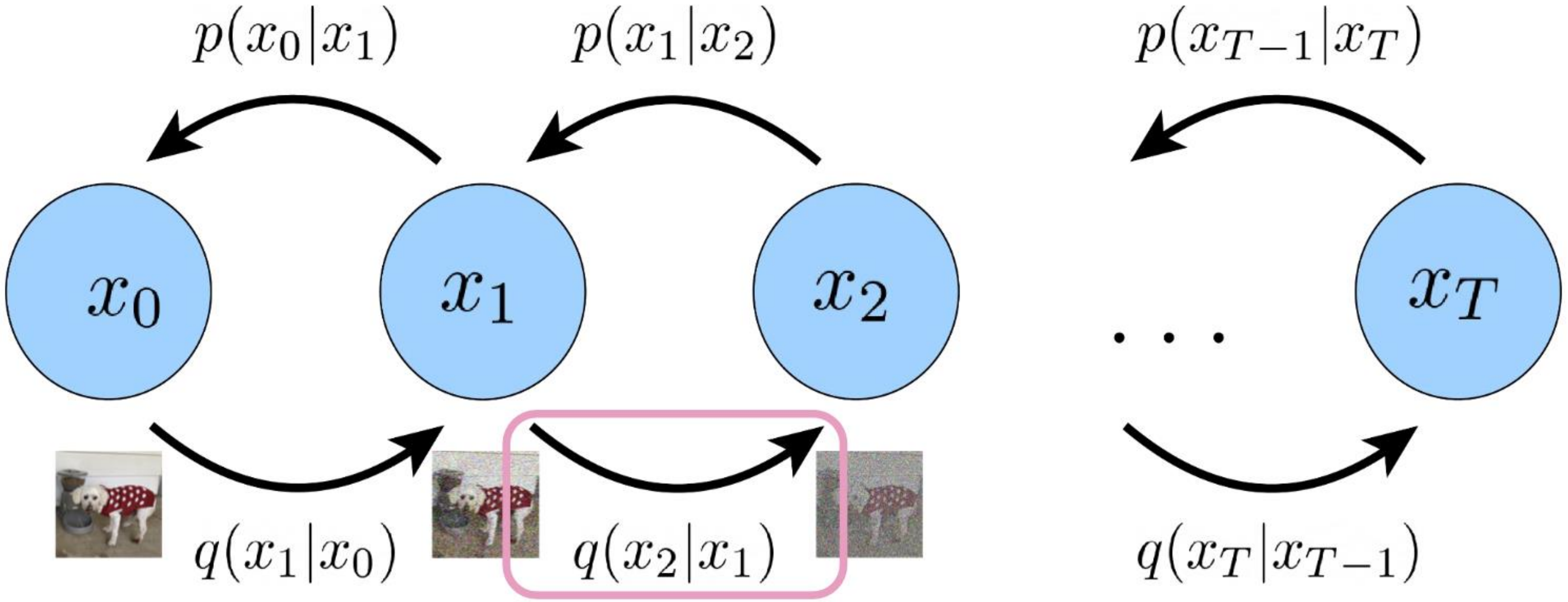
Adding Noise

What if each encoder transitions sample the input with some gaussian noise?



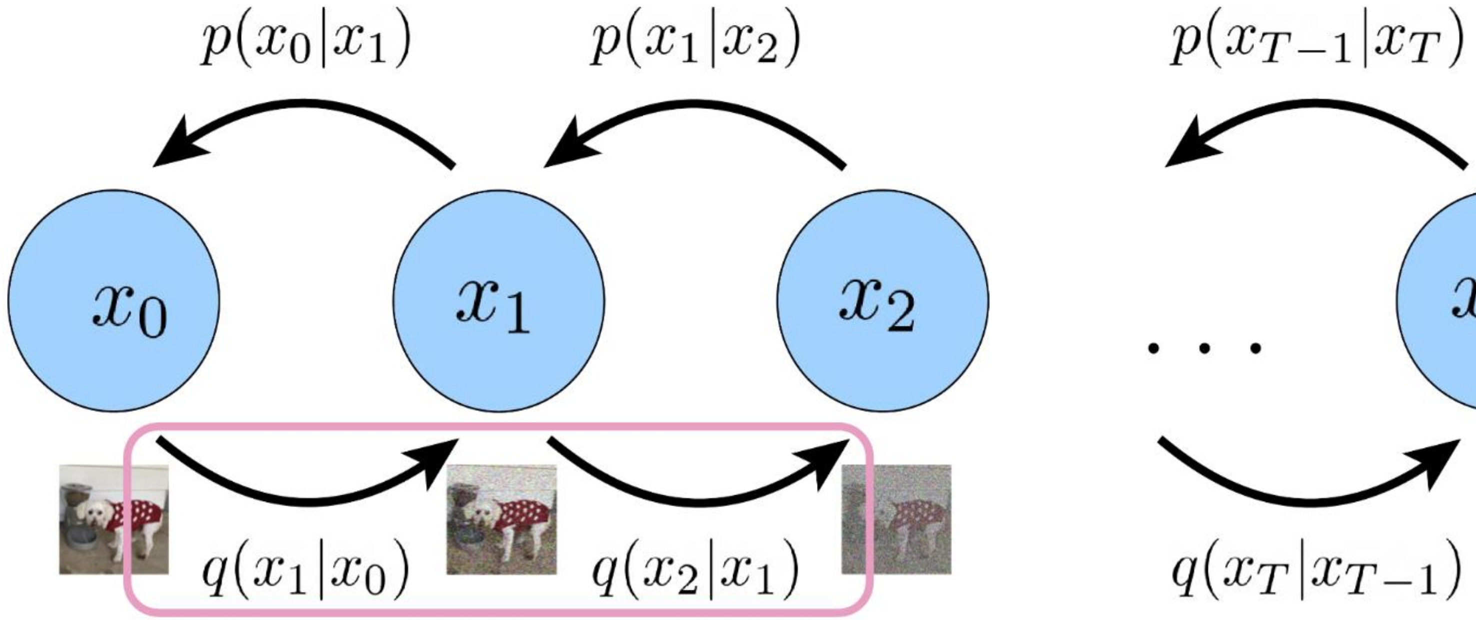
$$q(x_1|x_0) = \mathcal{N}(x_1|x_0, \sigma_1^2 \mathrm{I})$$

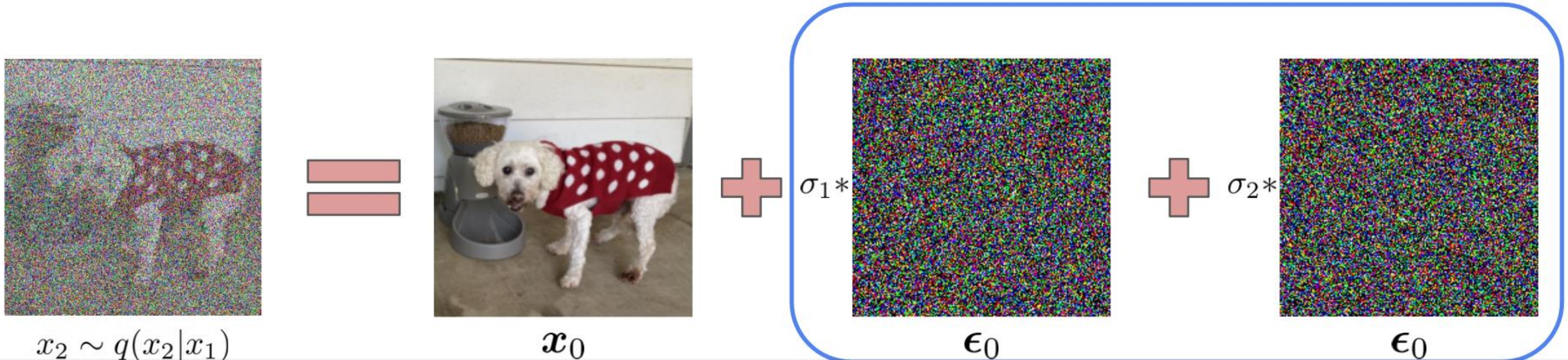




$$q(x_2|x_1) = \mathcal{N}(x_2|x_1, \sigma_2^2 \mathrm{I})$$

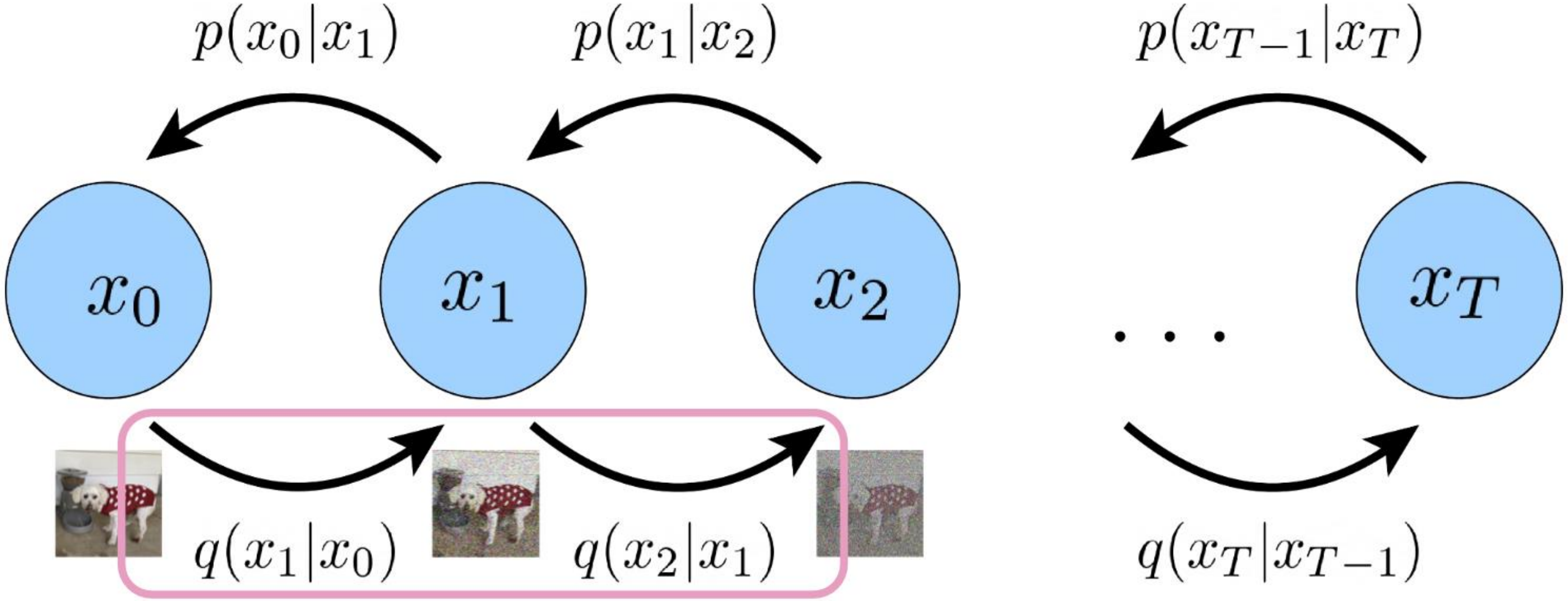
reparam. trick!

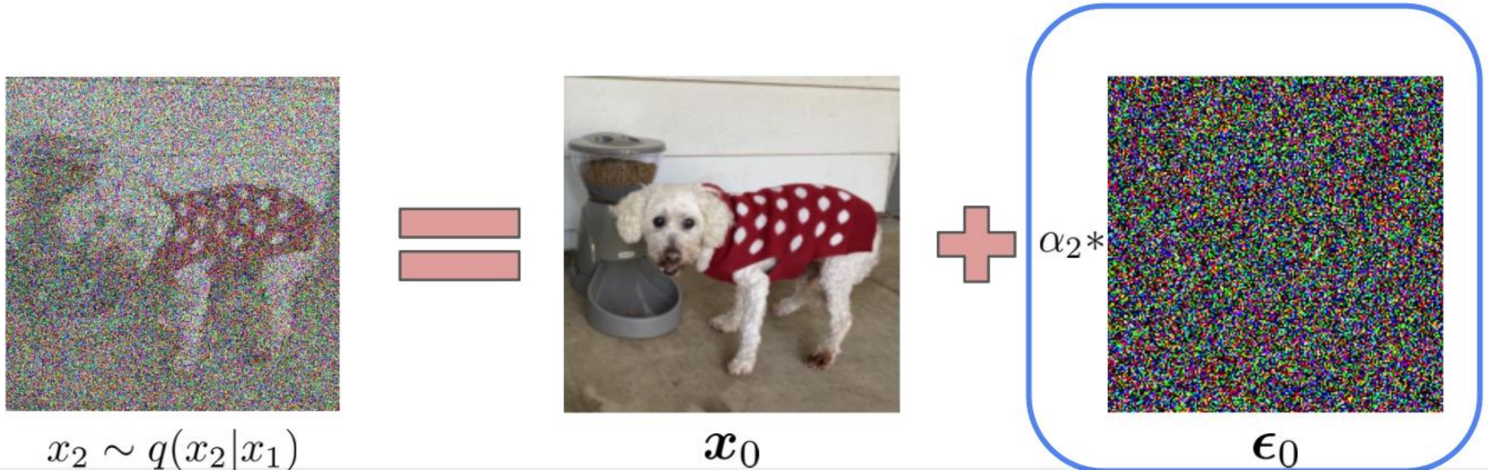




Aggregate into 1 sample!

reparam. trick!





where,

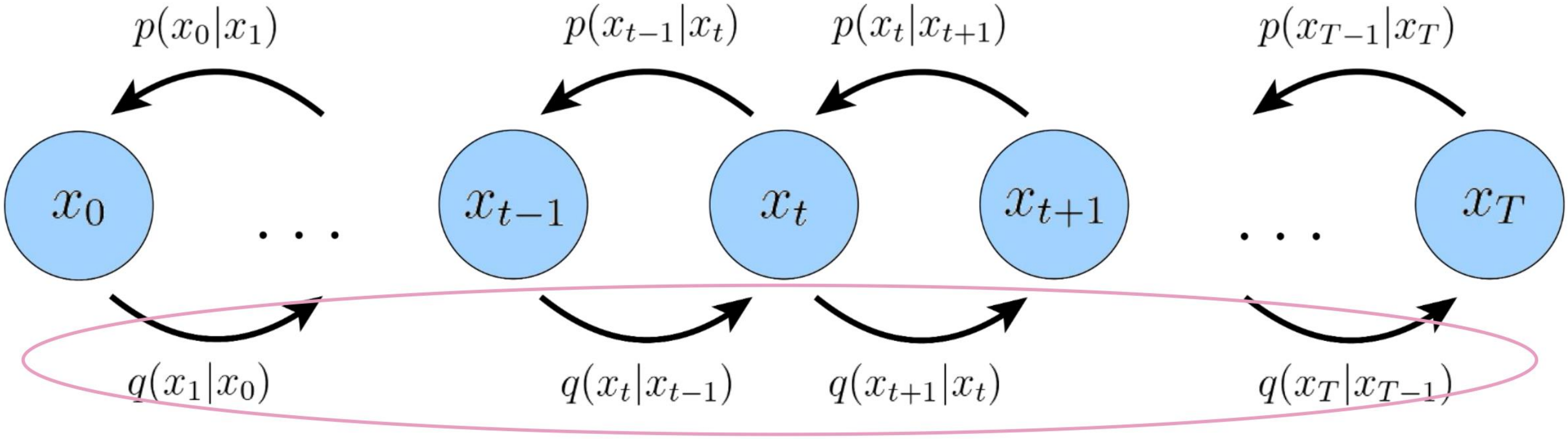
$$\alpha_2 = \sqrt{\sigma_1^2 + \sigma_2^2}$$

and,

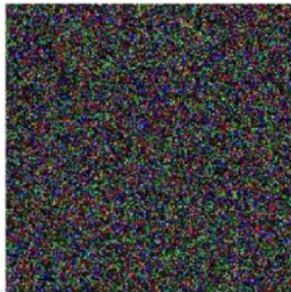
$$q(x_2|x_0) = \mathcal{N}(x_2|x_0, \alpha_2^2)$$

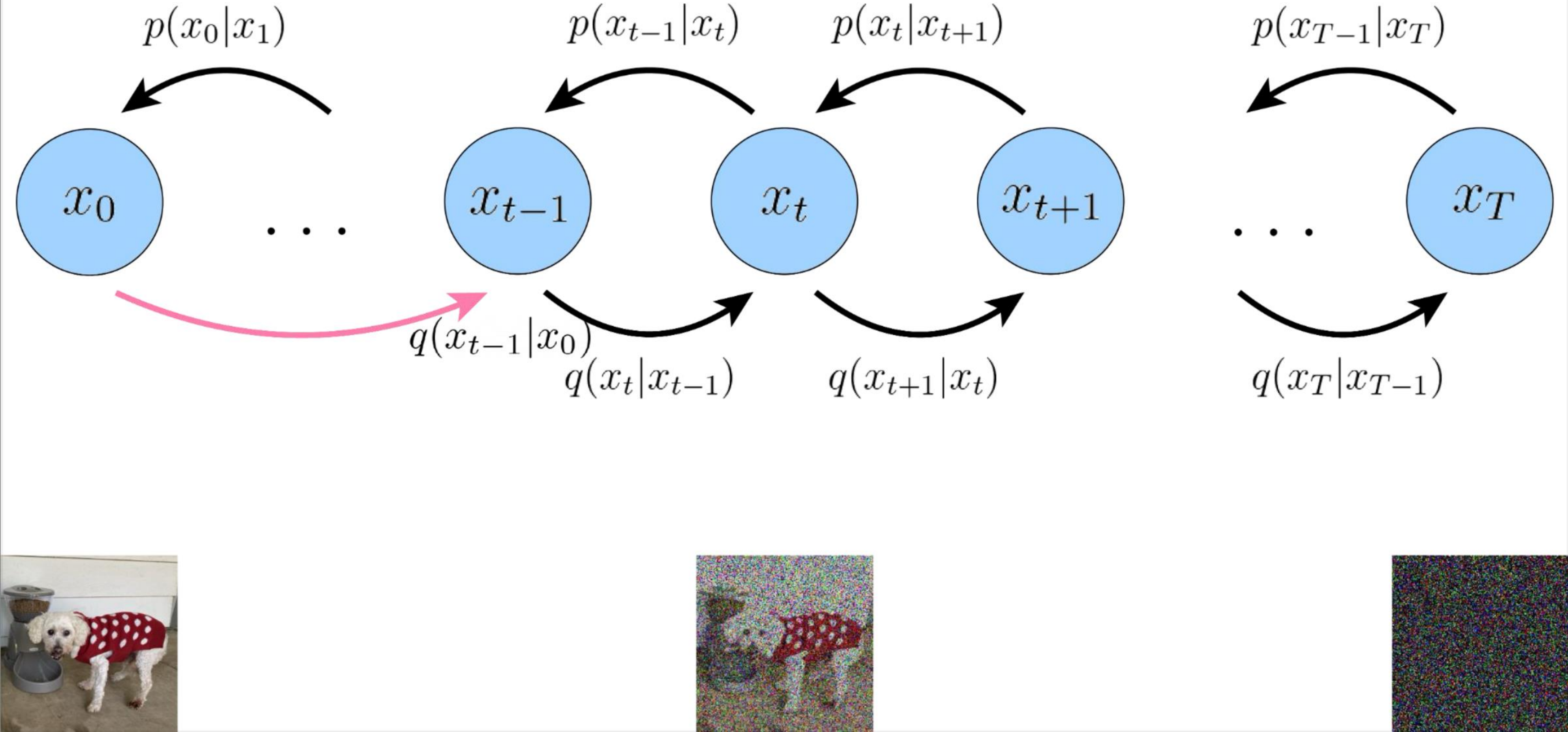
Aggregate into 1 sample!

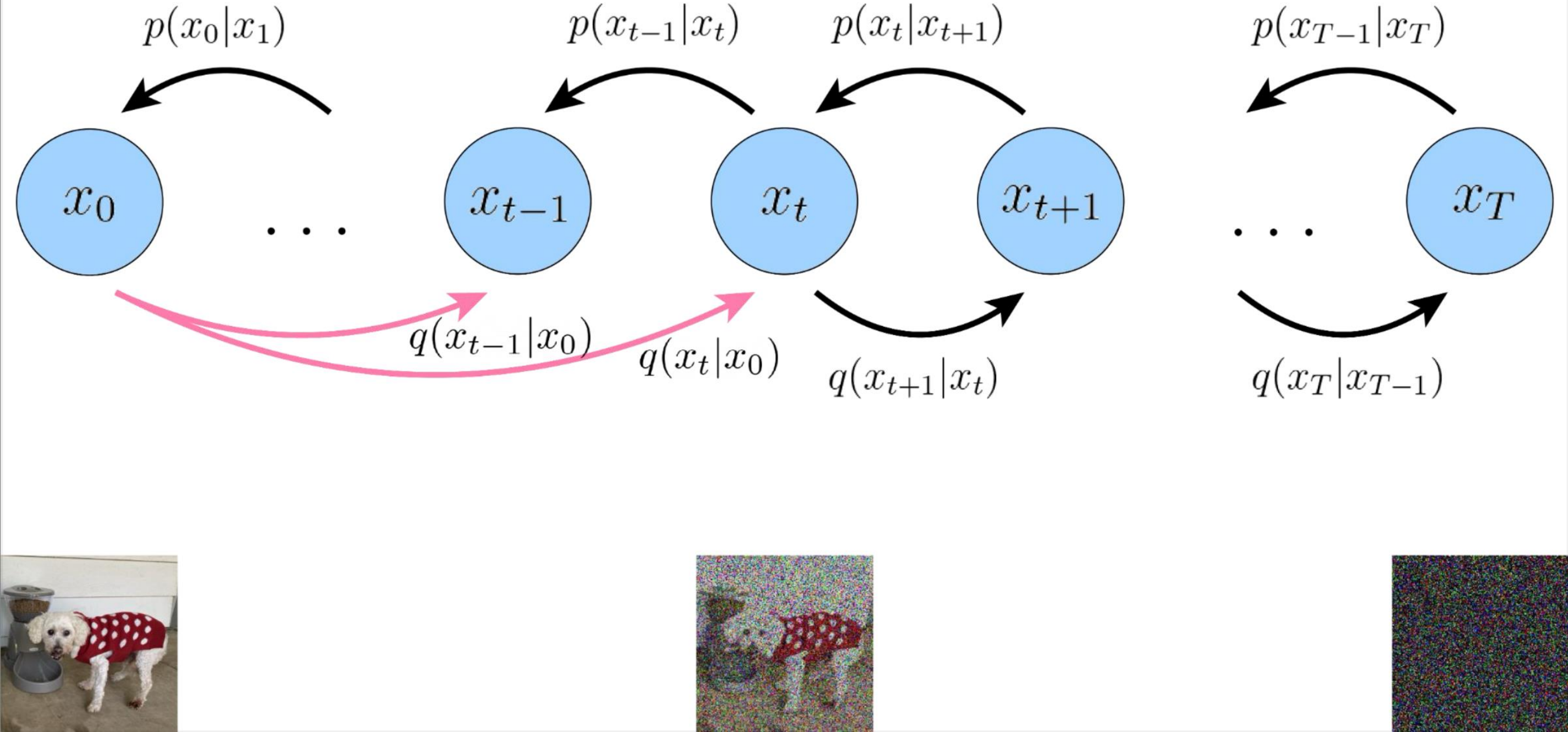
reparam. trick!



Individual (known) Gaussians!



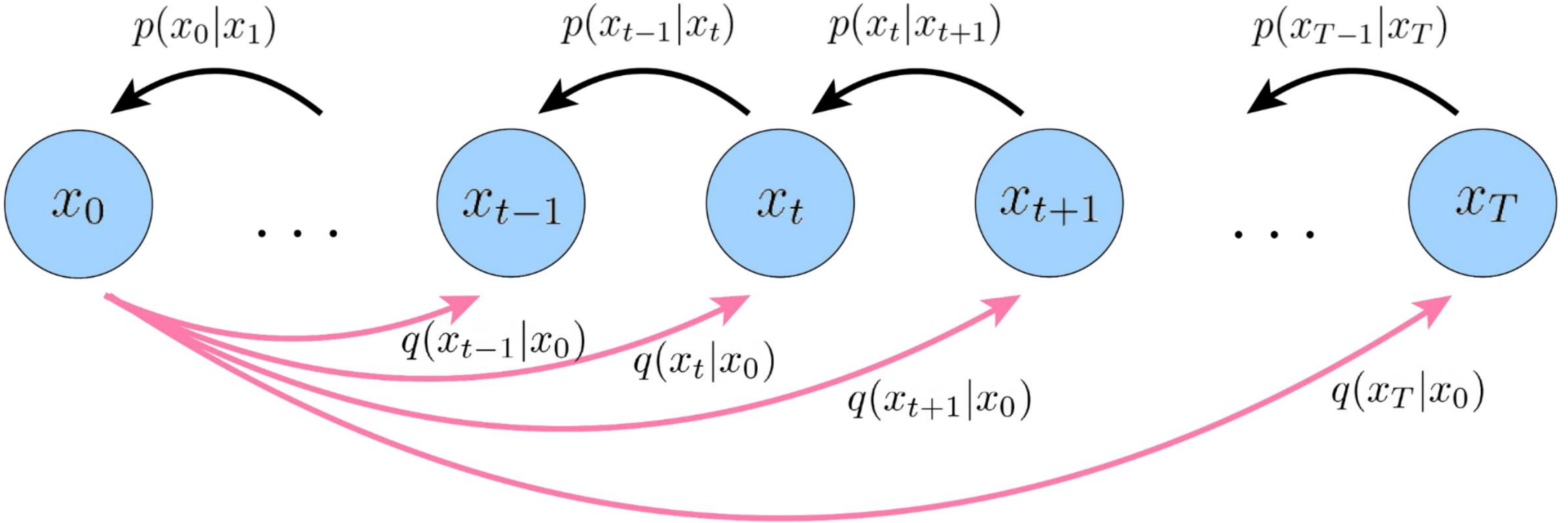

$p(x_0|x_1)$
$p(x_{t-1}|x_t)$
$p(x_t|x_{t+1})$
$p(x_{T-1}|x_T)$
x_0
$\cdots$
x_{t-1}
x_t
x_{t+1}
$\cdots$
x_T
$q(x_{t-1}|x_0)$
$q(x_t|x_{t-1})$
$q(x_{t+1}|x_t)$
$q(x_T|x_{T-1})$

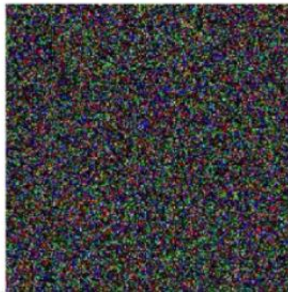


$p(x_0|x_1)$
$p(x_{t-1}|x_t)$
$p(x_t|x_{t+1})$
$p(x_{T-1}|x_T)$
x_0
$\cdots$
x_{t-1}
x_t
x_{t+1}
$\cdots$
x_T
$q(x_{t-1}|x_0)$
$q(x_t|x_0)$
$q(x_{t+1}|x_t)$
$q(x_T|x_{T-1})$

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

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

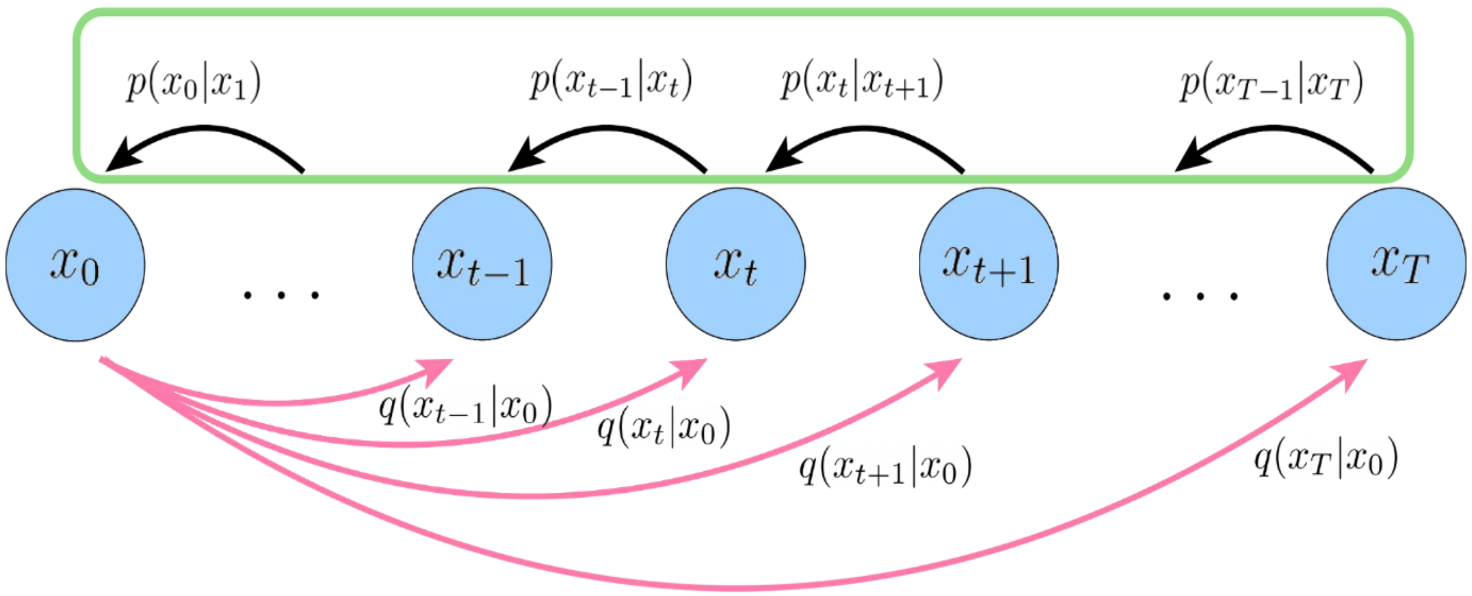


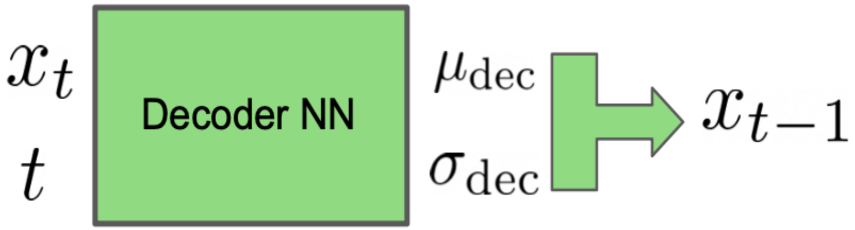


Diffusion Models

Are hierarchical VAEs with the following assumptions:

- All dimensions are the same (input size, encoding size)
- Encoder transitions are known gaussians centered around their previous inputs



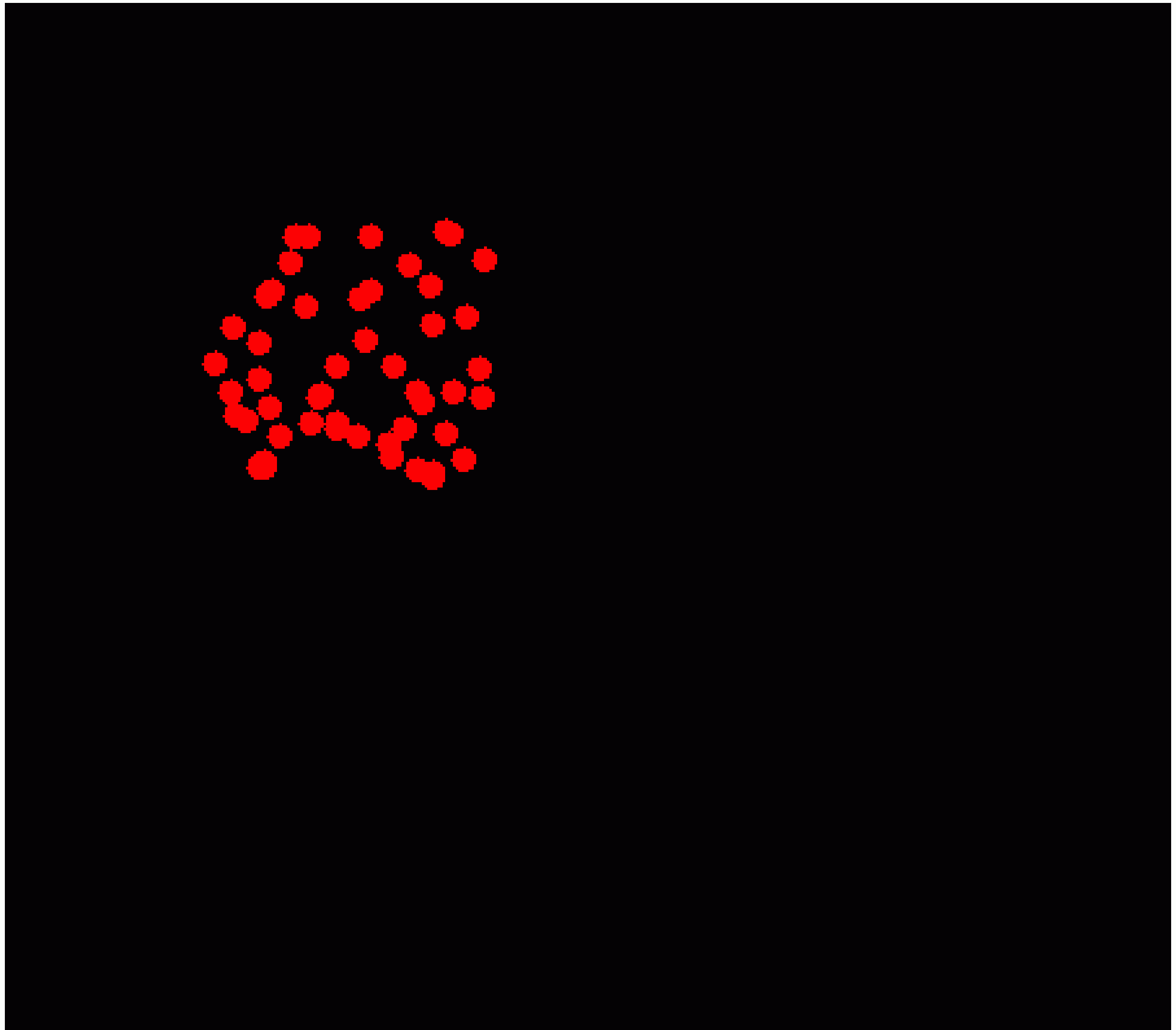


Why Call it Diffusion?

Diffusion of gas particles (and other physical things)

Start off organized

Transitions to “random noise”

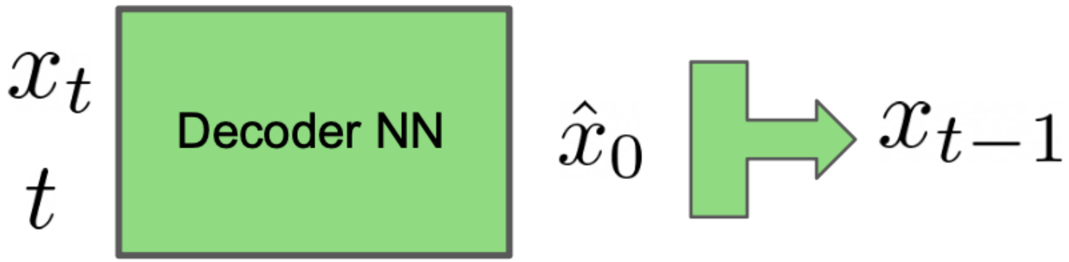


The Decoder

Diffusion models seek to learn a single neural network: a de-noising decoder

What form does x_{t-1} take?

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



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

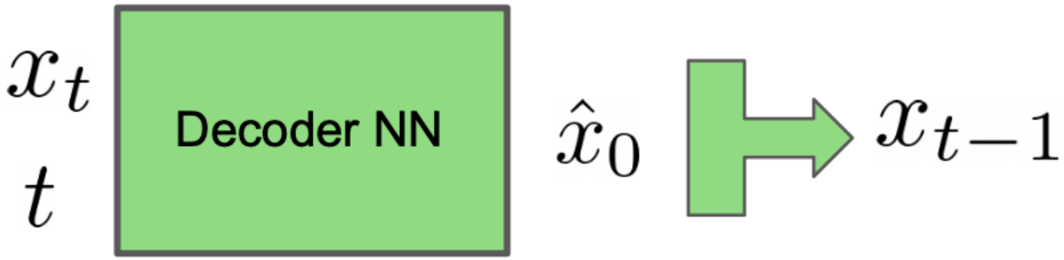
What is the ground truth for μ_{t-1}?

Does the decoder even need to predict σ?

Denoising Diffusion Models

Learn a single decoder network

- Input is image with added noise
- Output is predicted image with noise removed



How To Generate New Samples

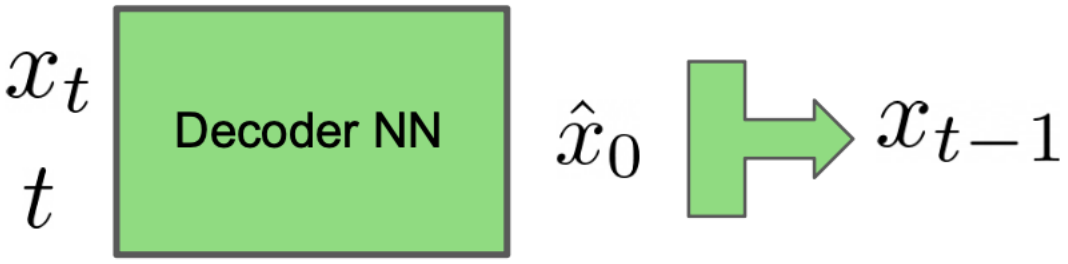
Idea 1: sample point from $\mathcal{N}(0, I)$, run decoder with t=T to generate $\hat{x}_0$

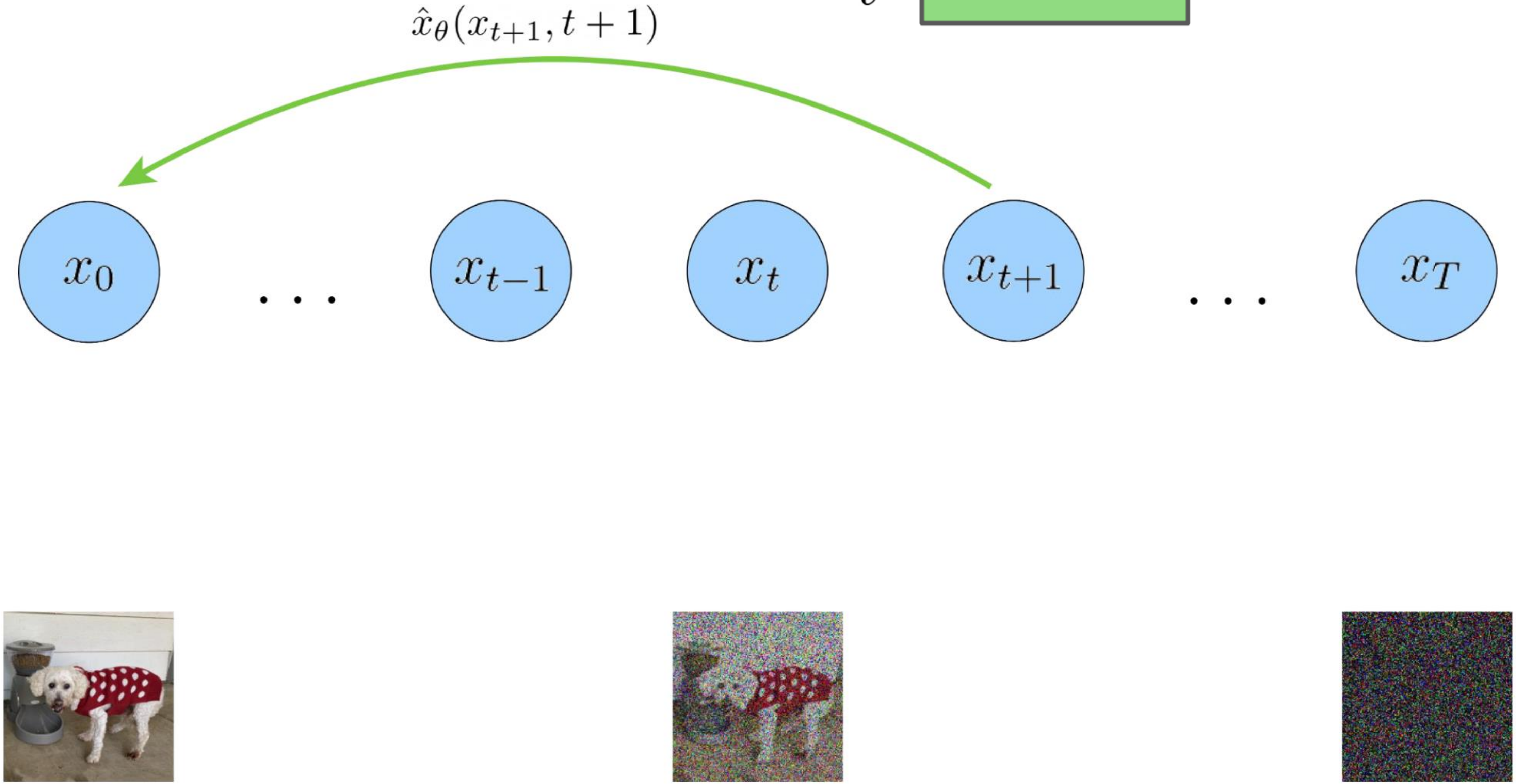
Issue: Doesn't work that well… that was the entire motivation for having multiple steps

Idea 2: sample point from $\mathcal{N}(0, I)$, iteratively generate $\hat{x}_{t-1}$

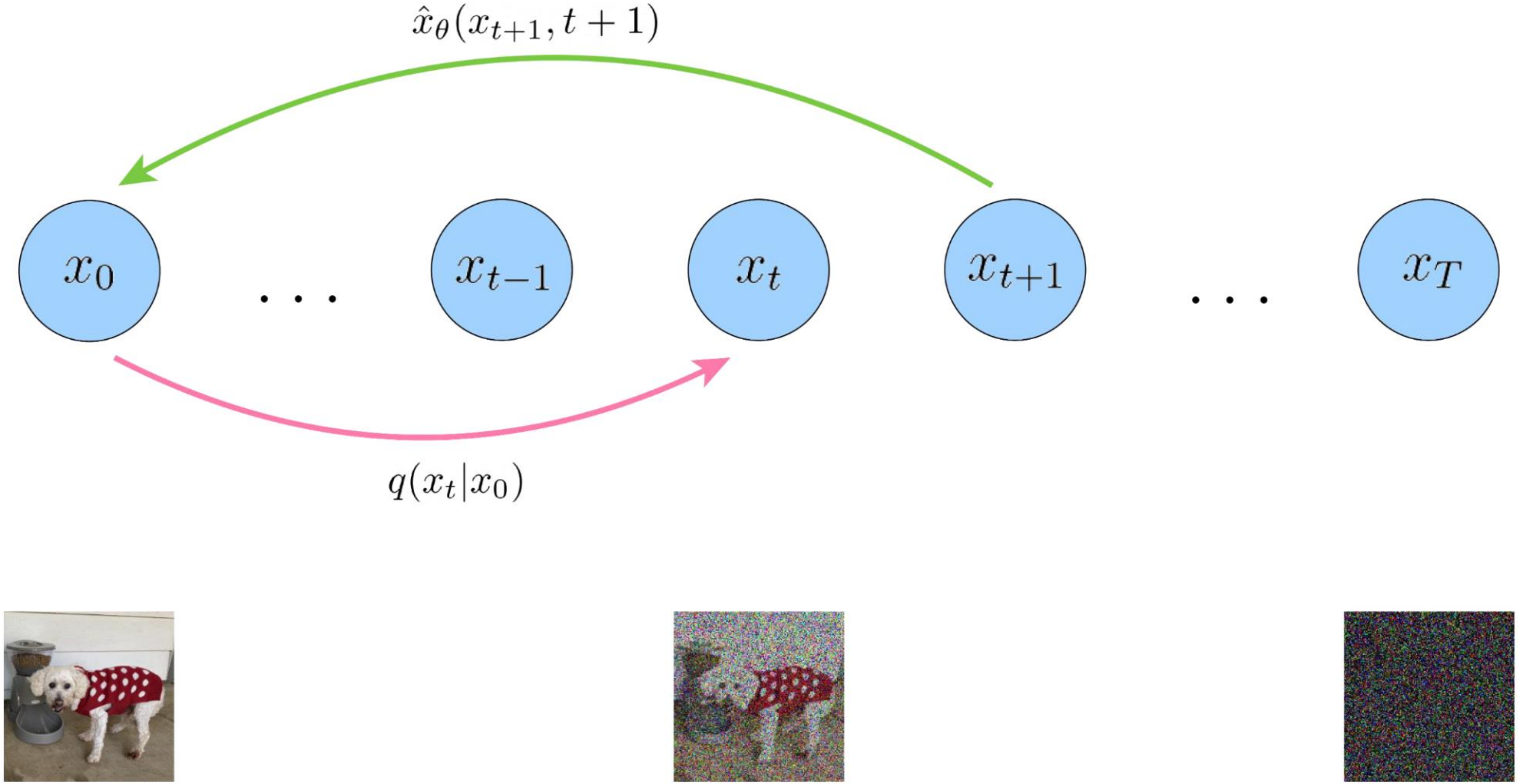
But how do we actually generate $\hat{x}_{t-1}$ when our decoder generates $\hat{x}_0$?

Sampling

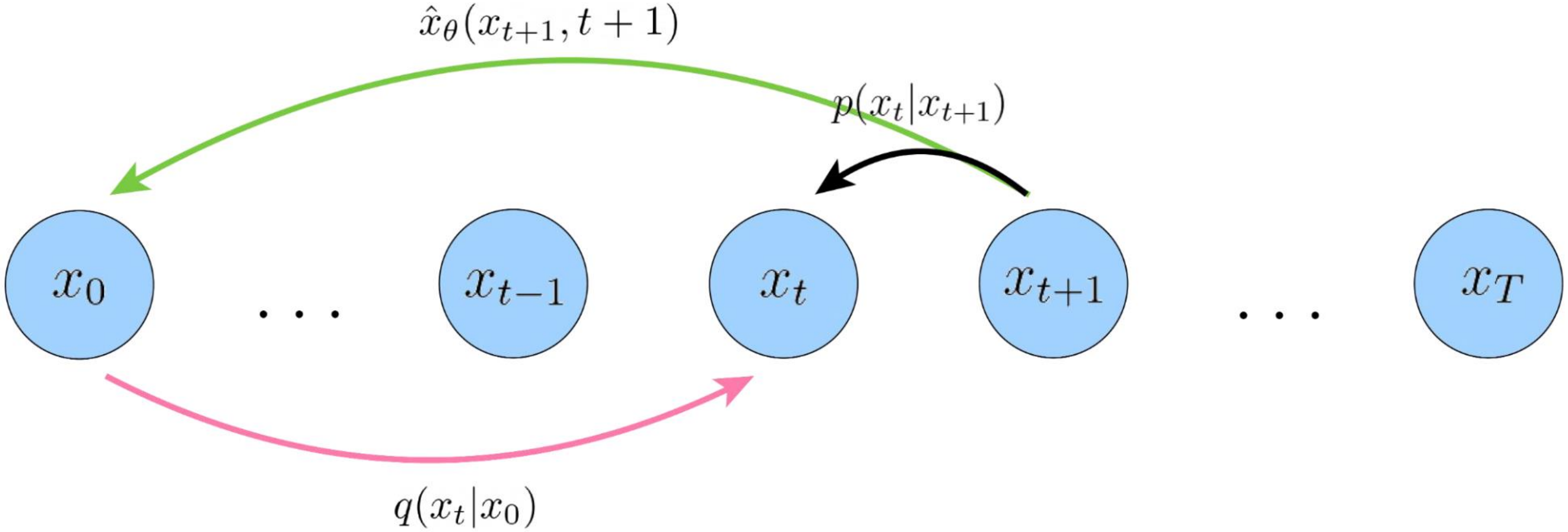


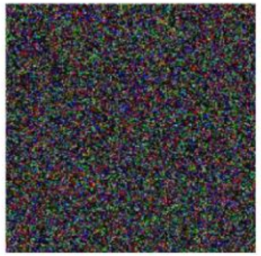


Sampling

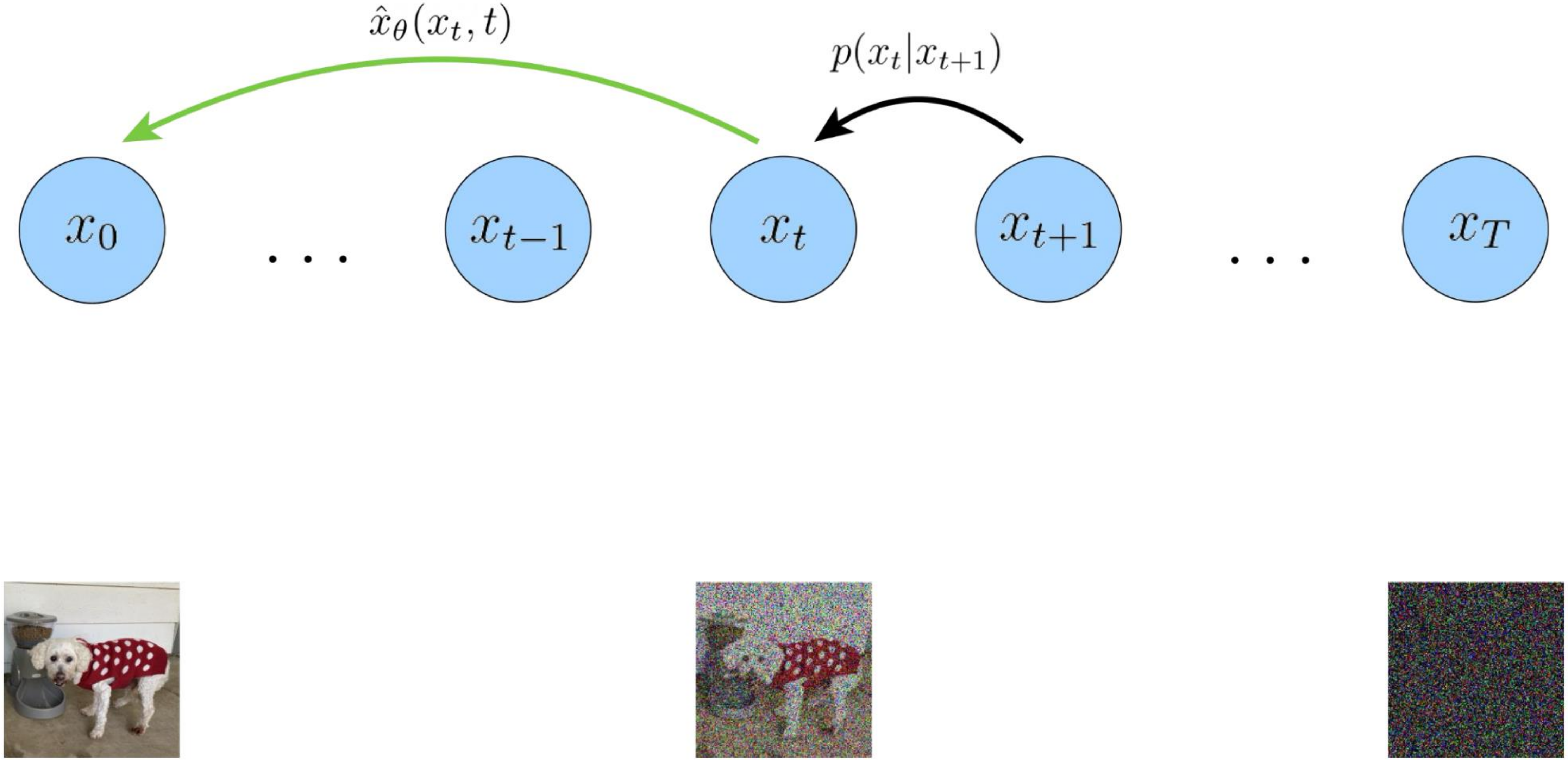


Sampling

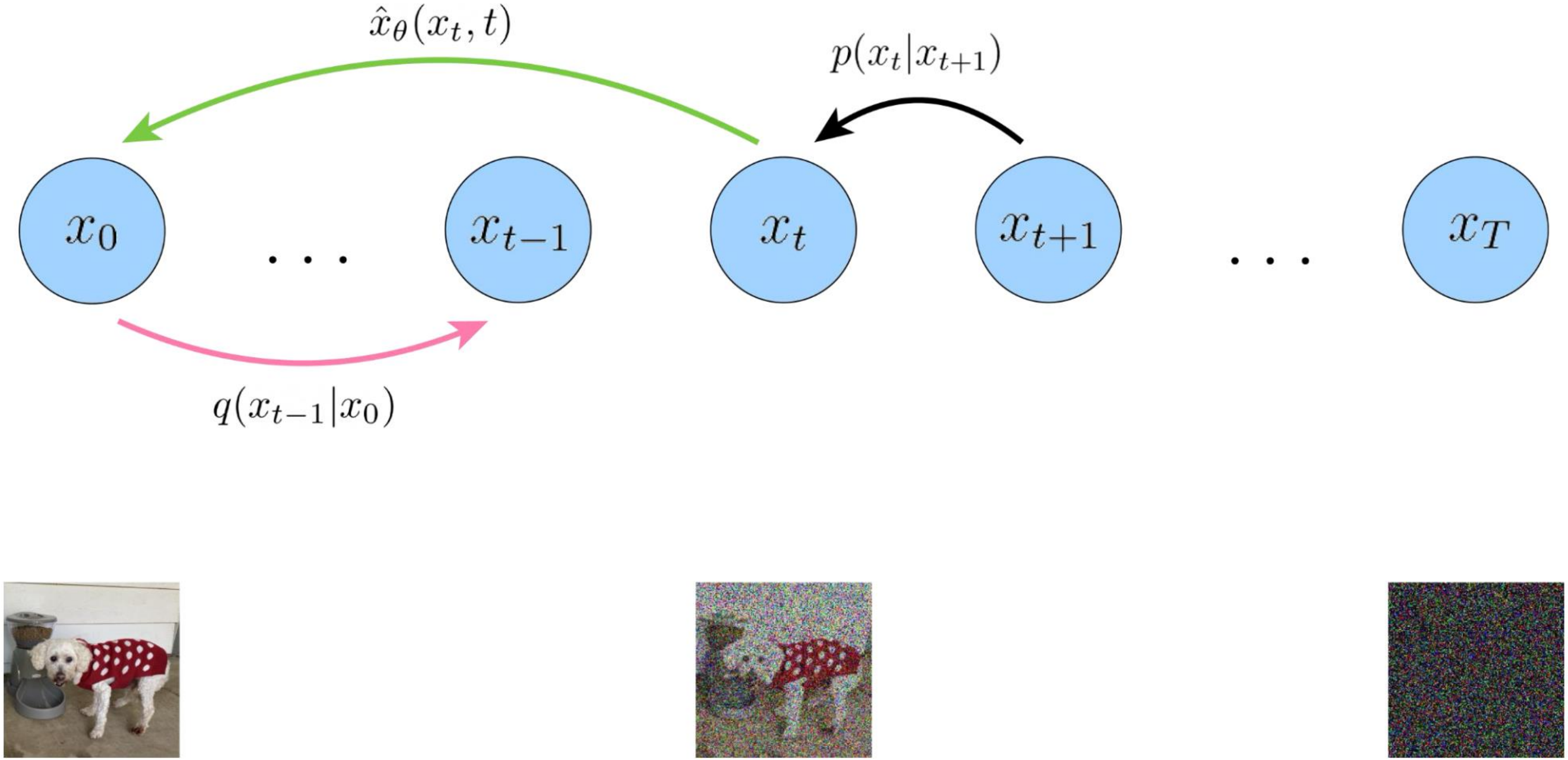




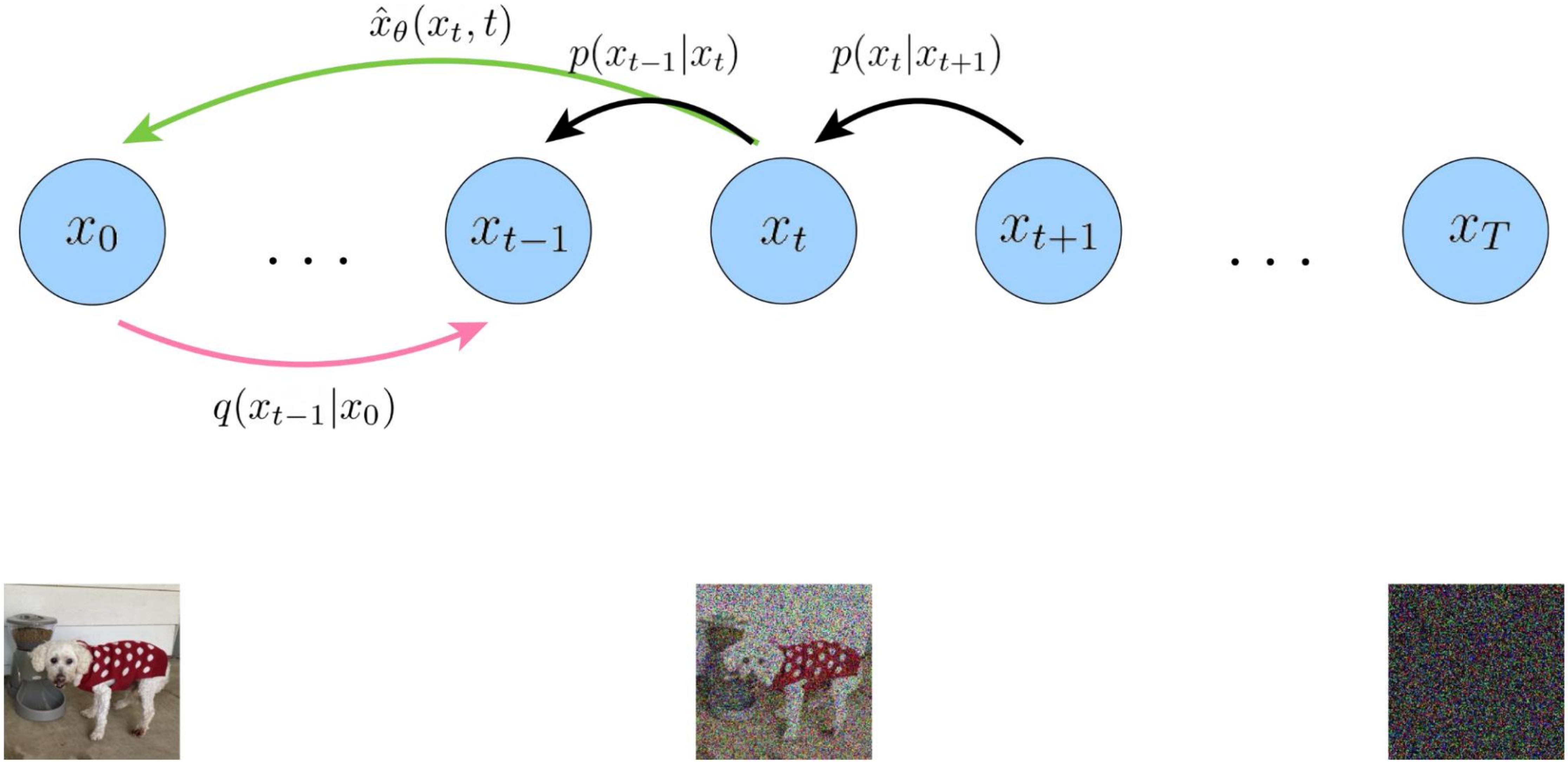
Sampling



Sampling

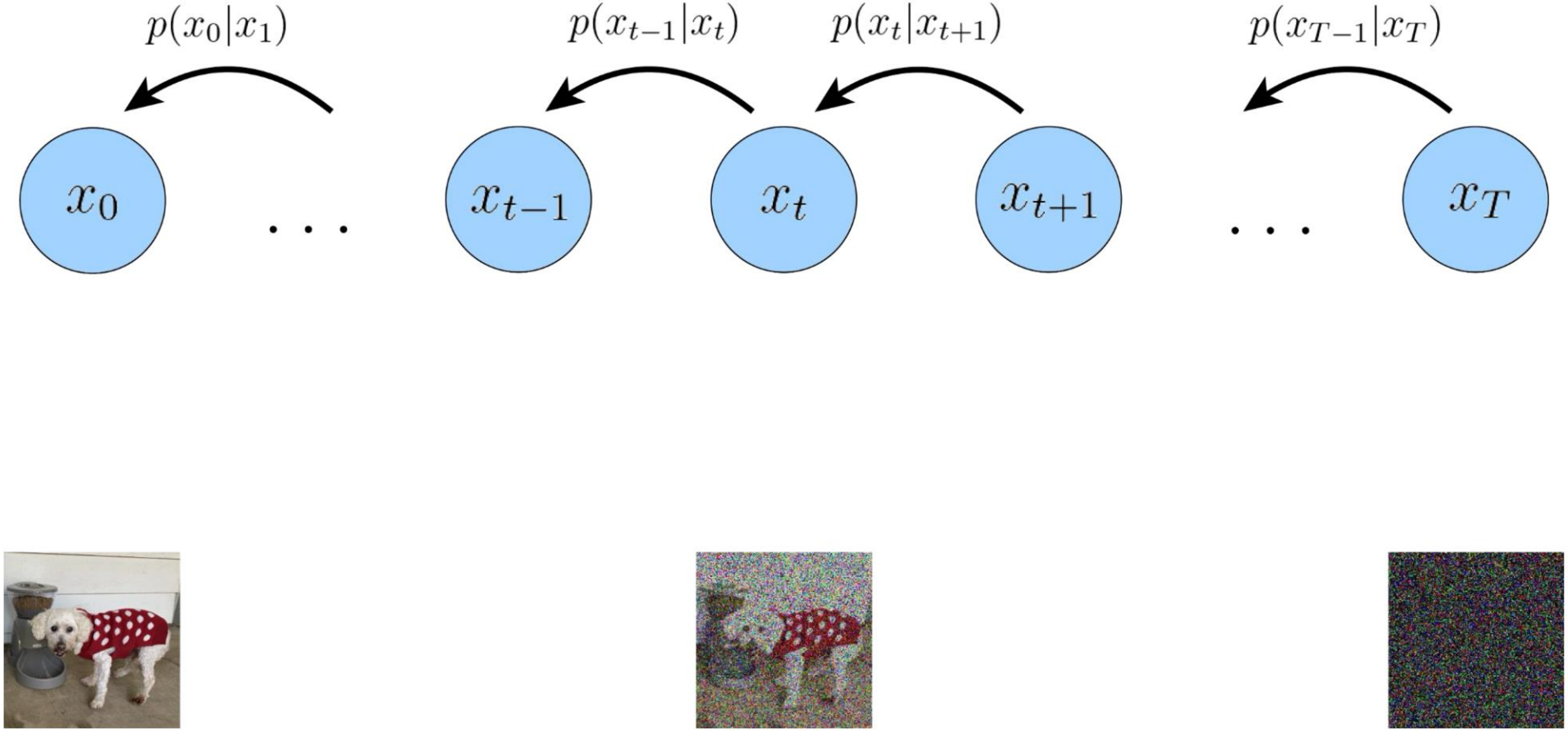


Sampling



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

Sampling

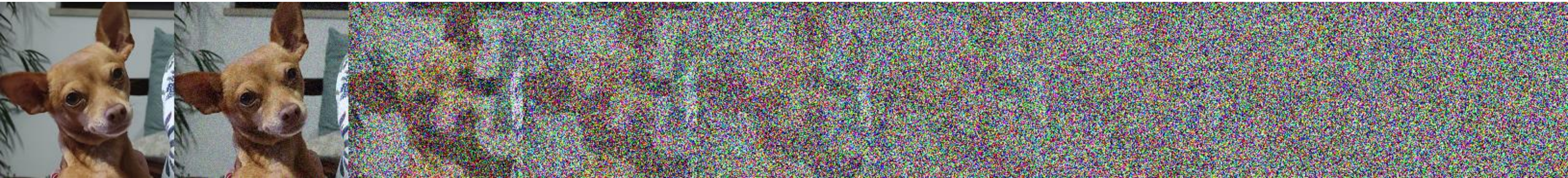


Noise Schedules

- What should the value of T be?
- How many steps of forward/reverse processes should we run?
- How much noise should be added at each step?

Amount of noise and number of steps determined by a *noise schedule (*hyperparameter)

Linear Schedule (equal noise added at each timestep)



Noise Schedules

- What should the value of T be?
- How many steps of forward/reverse processes should we run?
- How much noise should be added at each step?

Amount of noise and number of steps determined by a *noise schedule (*hyperparameter)

Cosine Schedule (small amounts of noise first, then fast)

Diffusion Training

Algorithm 1 Training

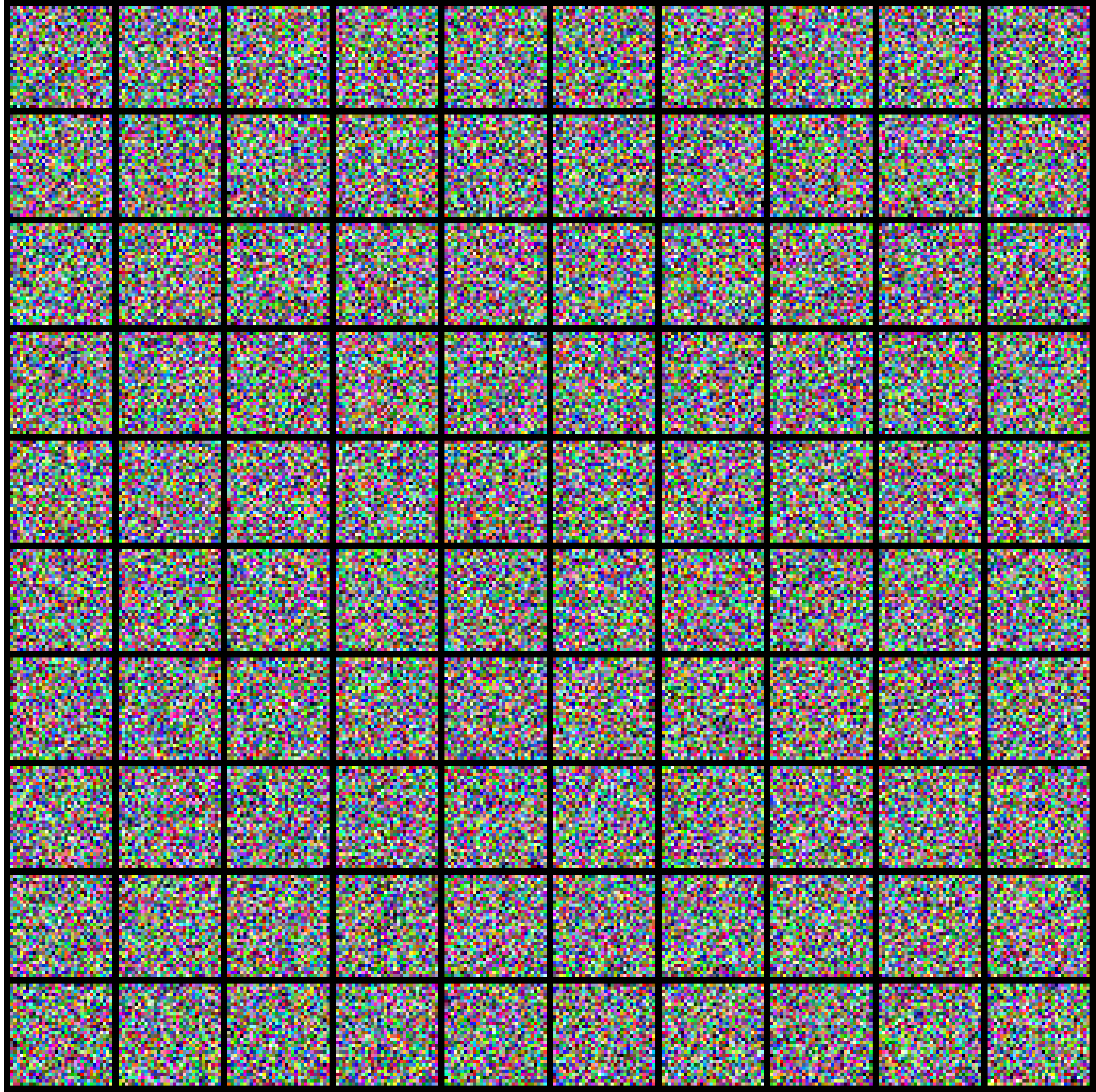
repeat
    $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
    $t \sim \texttt{Uniform}(1, ..., T)$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Take gradient descent step on
        $\nabla_\theta || \boldsymbol{x}_0 - \hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_0 + \alpha_t \boldsymbol{\epsilon},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

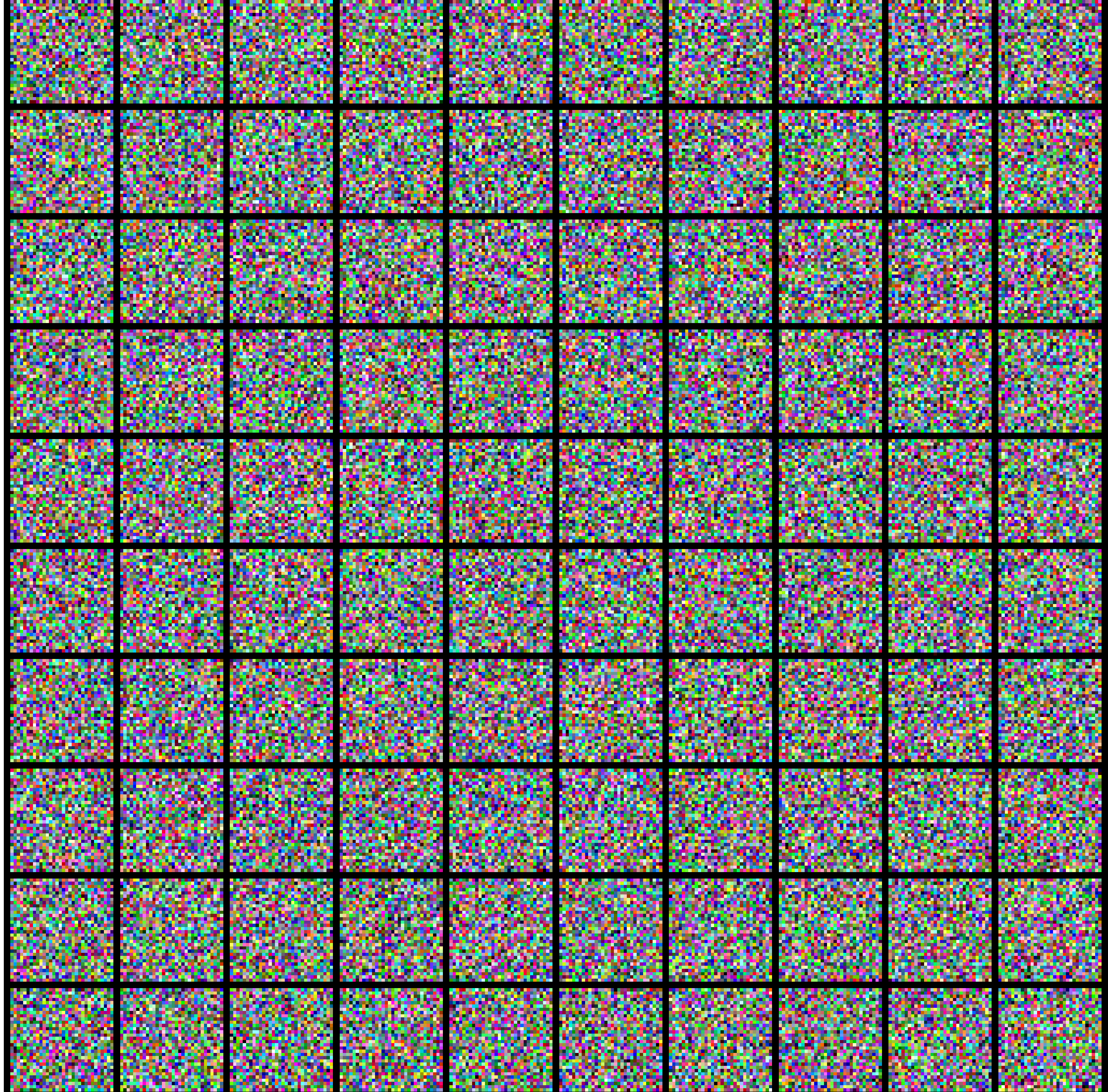
- Model trained on CelebA dataset



Source: https://yang-song.net/blog/2021/score/

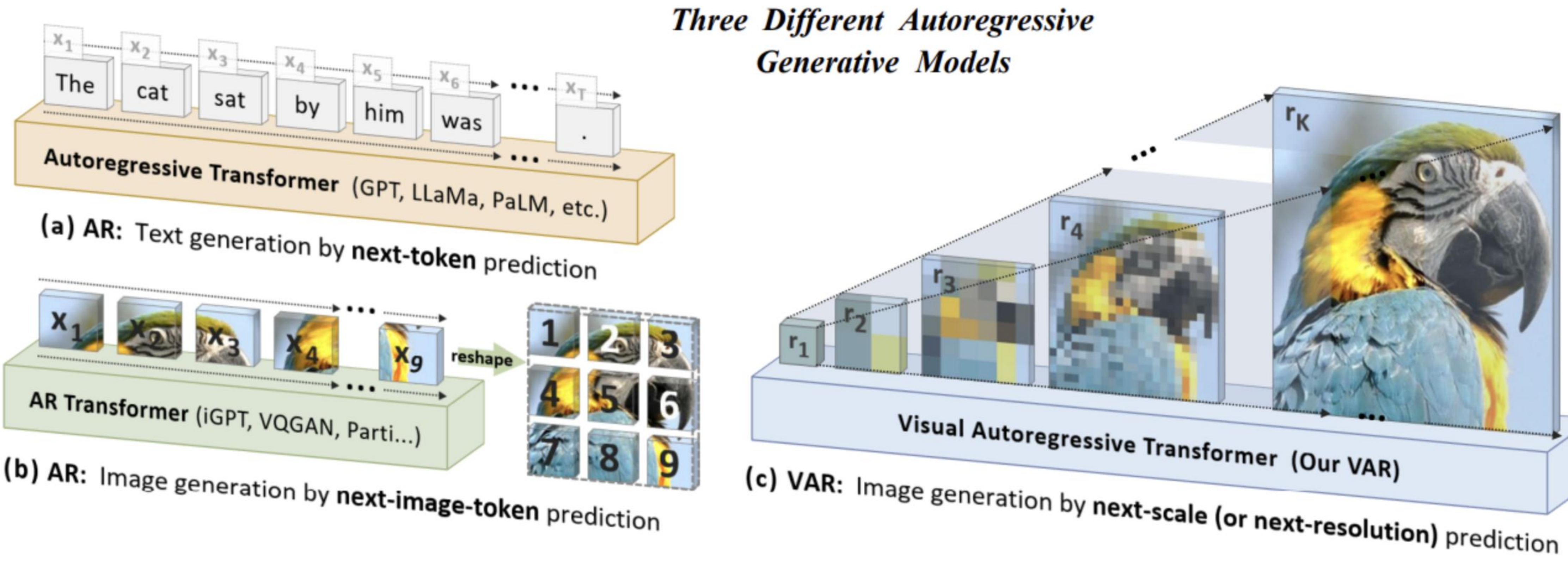
Examples

Model trained on CIFAR-10



Source: https://yang-song.net/blog/2021/score/

Visual Auto-Regressive Generation



Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction