Deep Learning

A course about <u>theory</u> & <u>practice</u>

Generative Networks: Diffusion Models

Marco Piastra

UNIVERSITÀ DI PAVIA

Images from text

- **DALL-E2**

  Diffusion Models: generating images from text

  «A teapot in the shape of an avocado»

[Image from https://www.nytimes.com/2022/04/06/technology/openai-images-dall-e.html]

Videos from text

- **SORA**

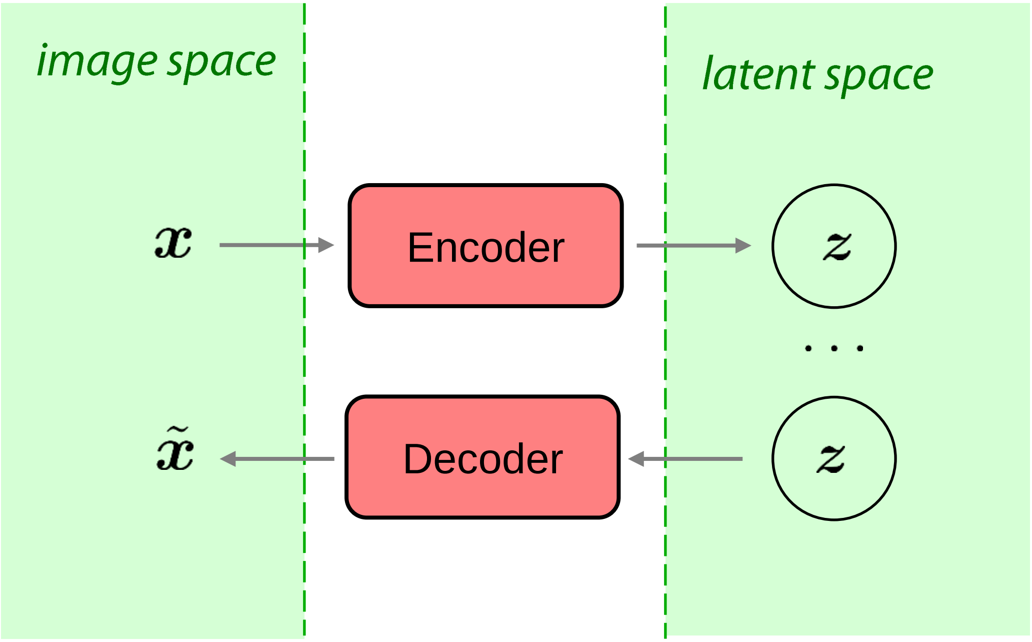
  Generating videos from text prompts

«A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.»

[Video clip from https://openai.com/index/sora/]

Diffusion Models are <u>autoencoders</u>

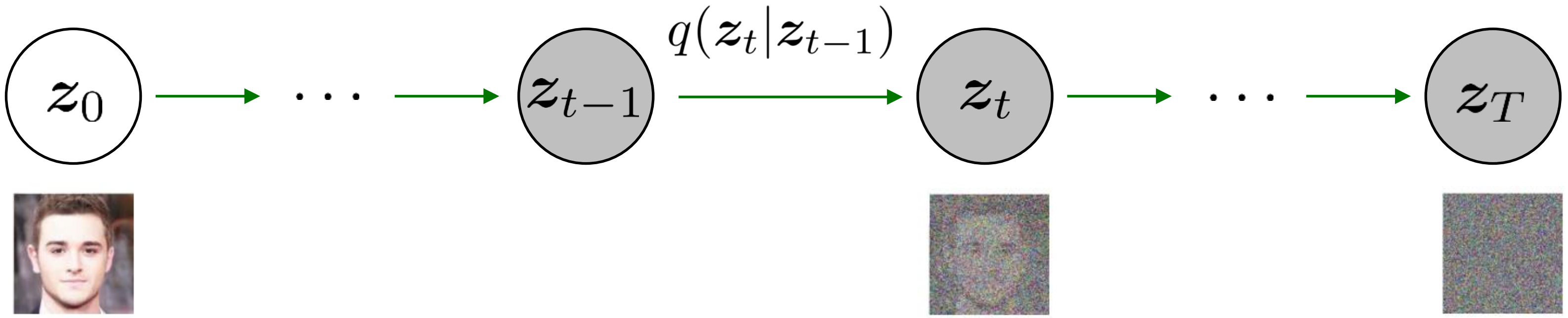
The basic, intuitive idea is to perform diffusion in the *latent space*



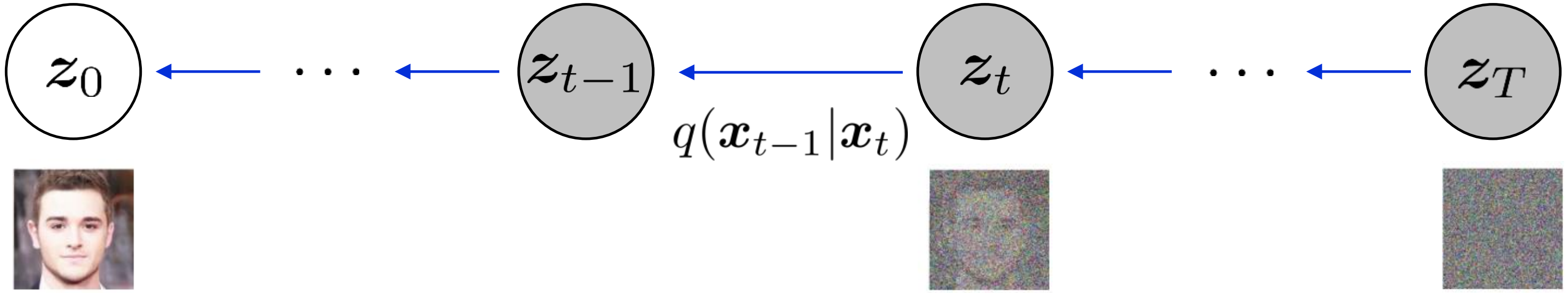
Diffusion Models

Basic idea

Forward Diffusion
Assume that images
are corrupted by
Gaussian noise
with known parameters



The idea behind
Denoising Diffusion Probabilistic Models (DDPM)
is learning how to reverse the process



Starting from the end: the DDPM training algorithm

Forward Diffusion
Assume that images
are corrupted by
Gaussian noise
with known parameters

The idea behind
Denoising Diffusion Probabilistic Models
is learning how to reverse the process

Note that
The neural network $g(\boldsymbol{z}_t, \boldsymbol{\vartheta}, t)$
is expected to predict the noise ε
that has been applied

Algorithm 20.1: Training a denoising diffusion probabilistic model

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$
Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Network parameters $\mathbf{w}$

---

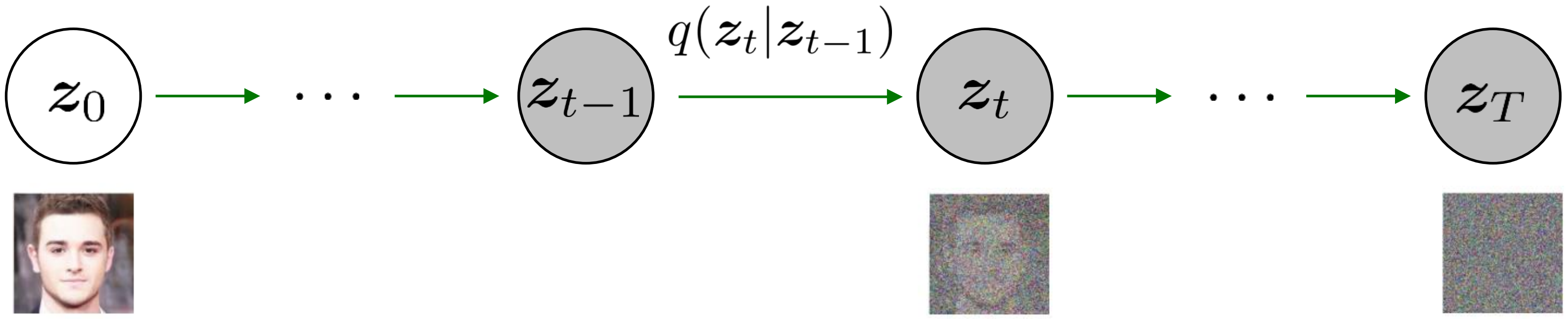
for $t \in \{1, \ldots, T\}$ **do**
  $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1 - \beta_\tau)$ `// Calculate alphas from betas`
end for
repeat
  $\mathbf{x} \sim \mathcal{D}$ `// Sample a data point`
  $t \sim \{1, \ldots, T\}$ `// Sample a point along the Markov chain`
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ `// Sample a noise vector`
  $\mathbf{z}_t \leftarrow \sqrt{\alpha_t}\mathbf{x} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}$ `// Evaluate noisy latent variable`
  $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_t, \boldsymbol{\vartheta}, t) - \boldsymbol{\epsilon}\|^2$ `// Compute loss term`
  Take optimizer step
until converged
return $\mathbf{w}$

Neural network
with suitable architecture

[Image from https://www.bishopbook.com/]

Forward diffusion

Forward Diffusion
Assume that images are corrupted by Gaussian noise with known parameters



$$\underbrace{\boldsymbol{z}_t \sim \mathcal{N}\left(\sqrt{1-\beta_t}\,\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}\right)}_{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}$$

$$\beta_t \in (0,1),\ \forall t$$

$$\beta_1 < \beta_2 < \cdots < \beta_T$$

'Noise schedule' : Hyperparameters

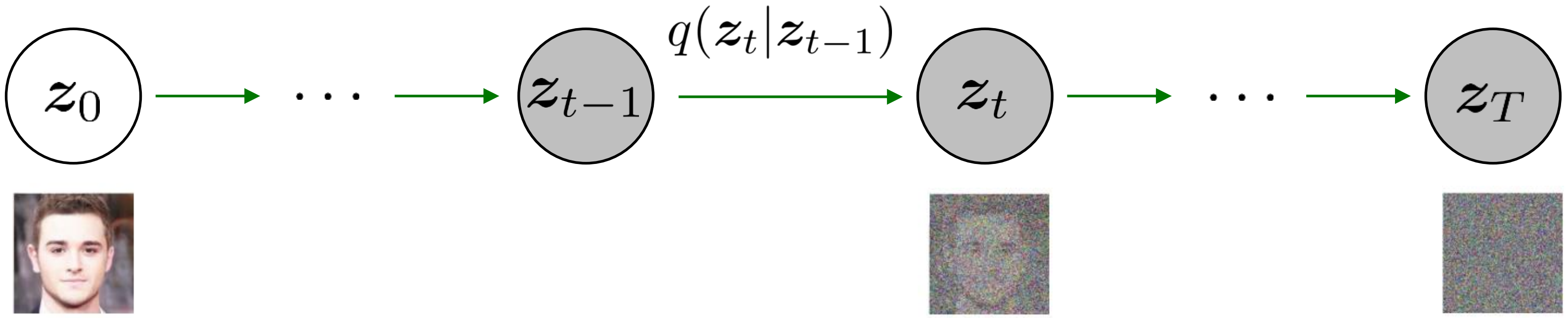
Could be rewritten as

$$\boldsymbol{z}_t = \sqrt{1-\beta_t}\,\boldsymbol{z}_{t-1} + \sqrt{\beta_t}\boldsymbol{\varepsilon}$$

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

Forward diffusion

Forward Diffusion
Assume that images
are corrupted by
Gaussian noise
with known parameters



At any forward step t, the diffusion sequence can be compacted as

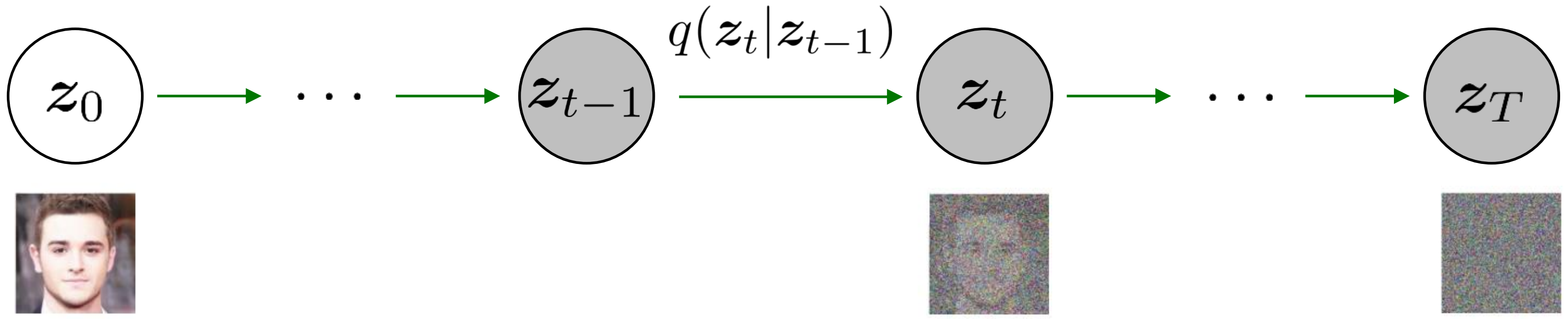
$$\boldsymbol{z}_t \sim \mathcal{N}\left(\sqrt{\alpha_t}\,\boldsymbol{z}_0, (1-\alpha_t)\boldsymbol{I}\right)$$

where:

$$\alpha_t = \prod_{\tau=1}^{t} (1-\beta_\tau)$$

Going backward: denoising

Backward Denoising
A neural network
is at the core
of the backward process



We assume that:

Neural Network

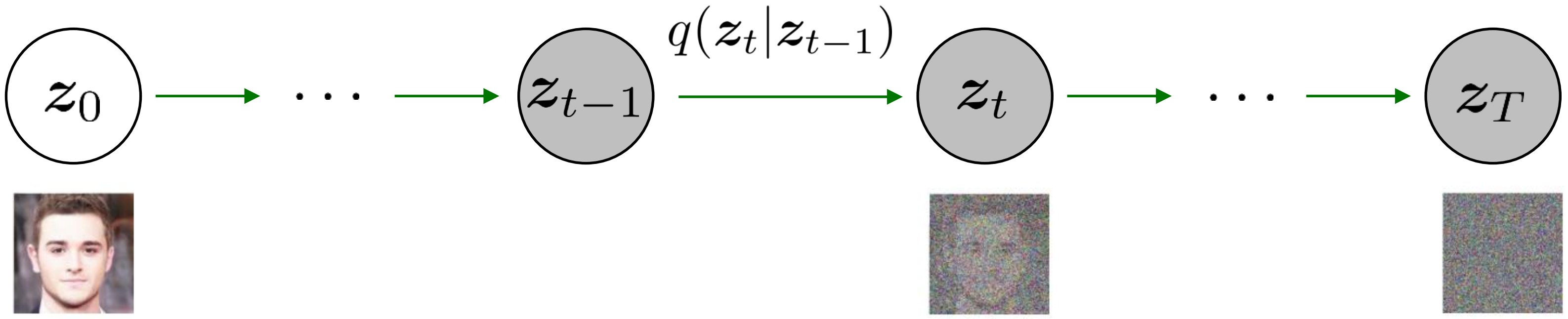
$$\boldsymbol{z}_{t-1} = \boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) + \sqrt{\beta_t}\, \boldsymbol{\varepsilon}$$

$$\boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) = \frac{1}{\sqrt{1-\beta_t}} \left\{ \boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) \right\}$$

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

Going backward: denoising

Backward Denoising
A neural network
is at the core
of the backward process



We assume that:

$$\boldsymbol{z}_{t-1} = \boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) + \sqrt{\beta_t}\, \boldsymbol{\varepsilon}$$

$$\boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) = \frac{1}{\sqrt{1-\beta_t}} \left\{ \boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \overbrace{\boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})}^{\textit{Neural Network}} \right\}$$

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

How can the neural network be trained?
(A suitable loss function is needed)

Going backward: denoising

$\tilde{q}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$

An approximation to $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$

We assume that:

$$\boldsymbol{z}_{t-1} \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}), \beta_t \boldsymbol{I}\right)$$

During <u>training</u>, $\boldsymbol{z}_0$ is known. Then we can sample $\boldsymbol{\varepsilon}_t$

$$\boldsymbol{z}_t = \sqrt{\alpha_t}\, \boldsymbol{z}_0 + \sqrt{1-\alpha_t}\, \boldsymbol{\varepsilon}_t$$

Noise added at step t

Therefore, it can be proven that:

$$\boldsymbol{m}(\boldsymbol{z}_{t-1}) = \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\, \boldsymbol{\varepsilon}_t\right)$$

is the true mean of:

$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$$

Then the Kullback-Leibler divergence is:

$$\begin{aligned}&\mathrm{KL}\left(q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \parallel \tilde{q}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)\right)\\ &\quad = \frac{1}{2\beta_t}\|\boldsymbol{m}(\boldsymbol{z}_{t-1}) - \boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})\|^2 + \mathrm{const}\end{aligned}$$

Going backward: denoising

$$\begin{aligned}\mathrm{KL}\left(q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \parallel \tilde{q}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)\right) \\ = \frac{1}{2\beta_t}\|\boldsymbol{m}(\boldsymbol{z}_{t-1}) - \boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})\|^2 + \mathrm{const}\end{aligned}$$

Therefore, given previous assumptions:

$$\boldsymbol{m}(\boldsymbol{z}_{t-1}) = \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon}_t\right)$$

$$\boldsymbol{\mu}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) = \frac{1}{\sqrt{1-\beta_t}}\left\{\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})\right\}$$

$$\mathrm{KL}\left(\tilde{q}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \parallel q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)\right) \propto \|\, \boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) - \boldsymbol{\varepsilon}_t\|^2$$

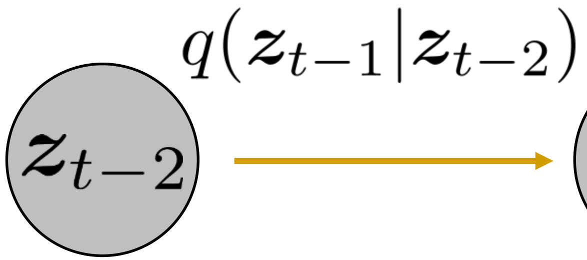
$$L(\boldsymbol{\vartheta}) := \|\, \boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) - \boldsymbol{\varepsilon}_t\|^2$$

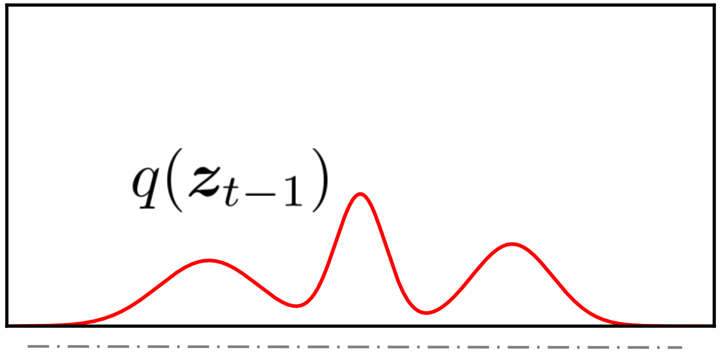
To be minimized

Fundamental: this line of reasoning works provided that $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$, $\tilde{q}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ *are Gaussians ...*

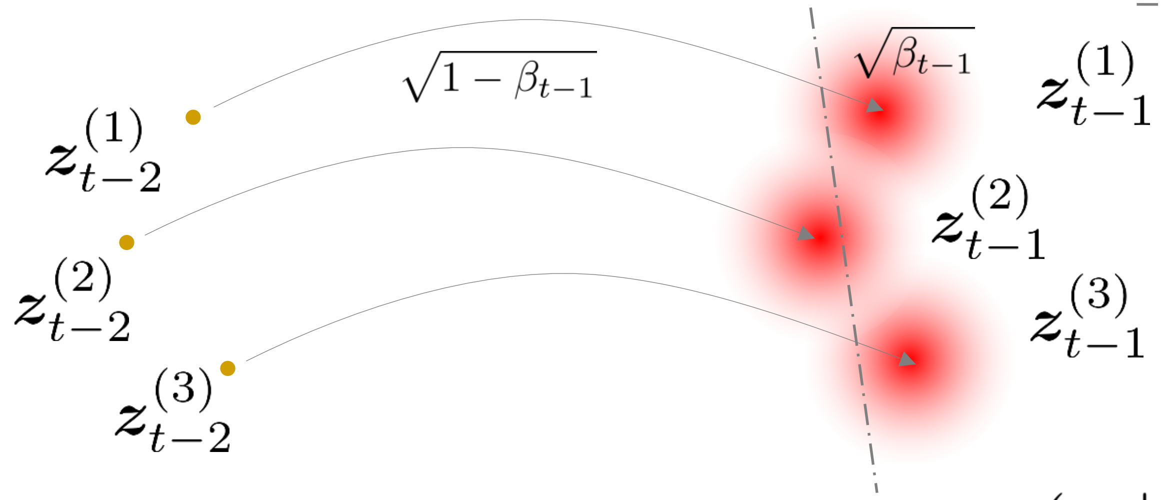
Diffusion Models: Why so many steps?

Going forward: adding noise

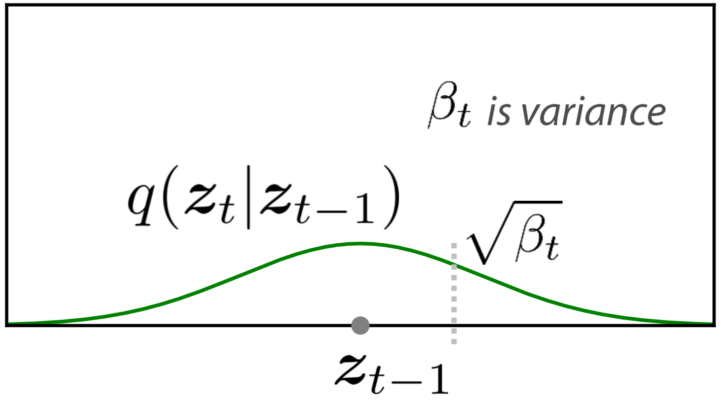


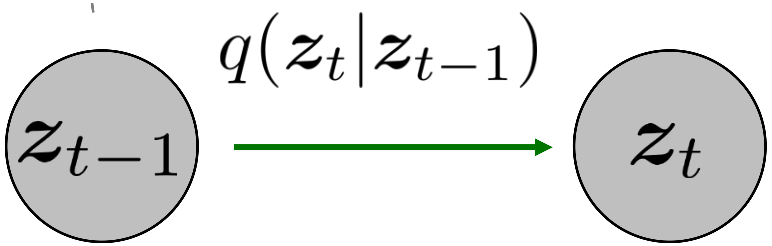


Marginal distribution
(cross-section)
at step $t-1$



Forward probability distribution
(Gaussian)

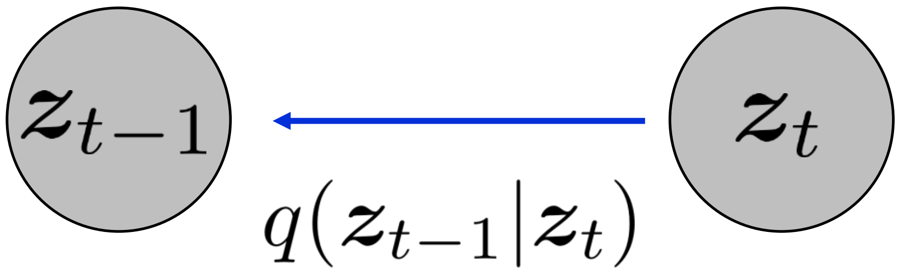


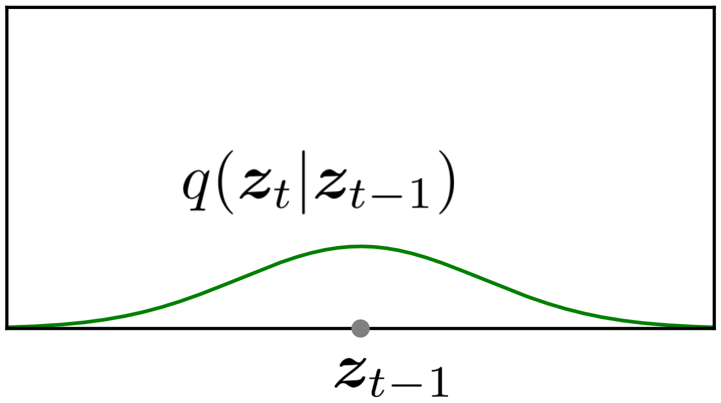


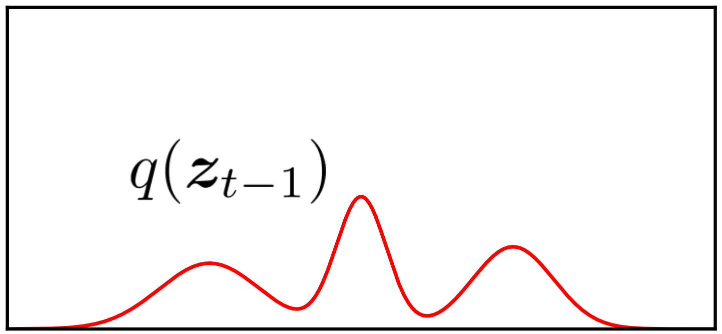
$$\mathbf{z}_t = \sqrt{1-\beta_t}\;\mathbf{z}_{t-1} + \sqrt{\beta_t}\;\varepsilon$$

Going backward: denoising

The backward probability distribution can be computed from forward and marginals using Bayes' theorem





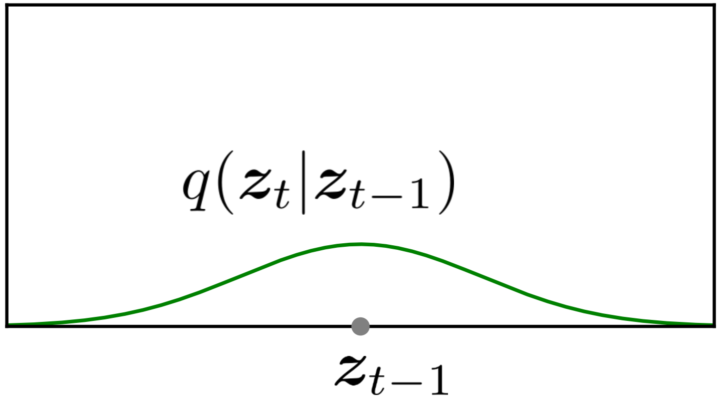


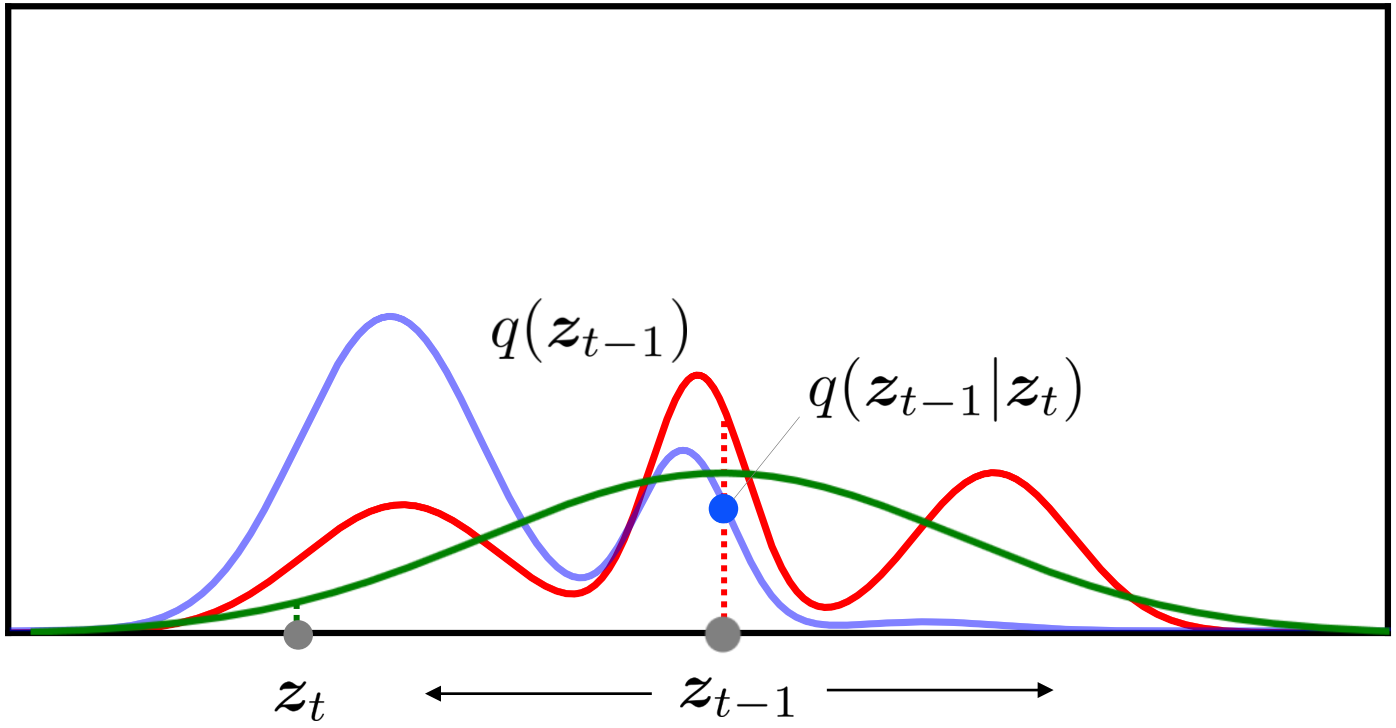
This is what we want to learn *This is what we know, by design*

$$\underbrace{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)} = \frac{\overbrace{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})q(\boldsymbol{z}_{t-1})}}{q(\boldsymbol{z}_t)}$$

Bayes' Theorem

Going backward: denoising





At training time, z_t is known (dataset + forward diffusion)

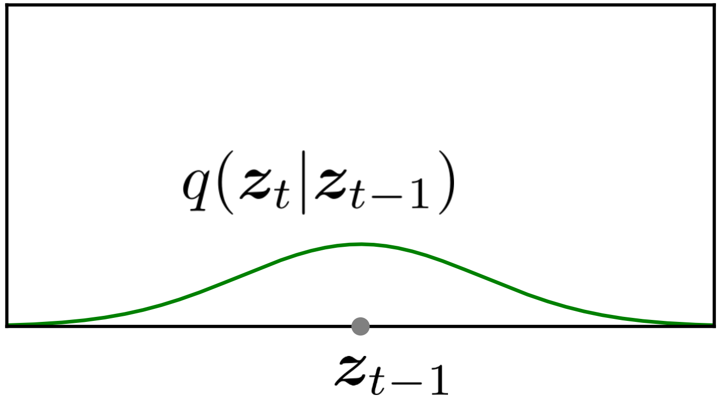
is the free variable, at this step

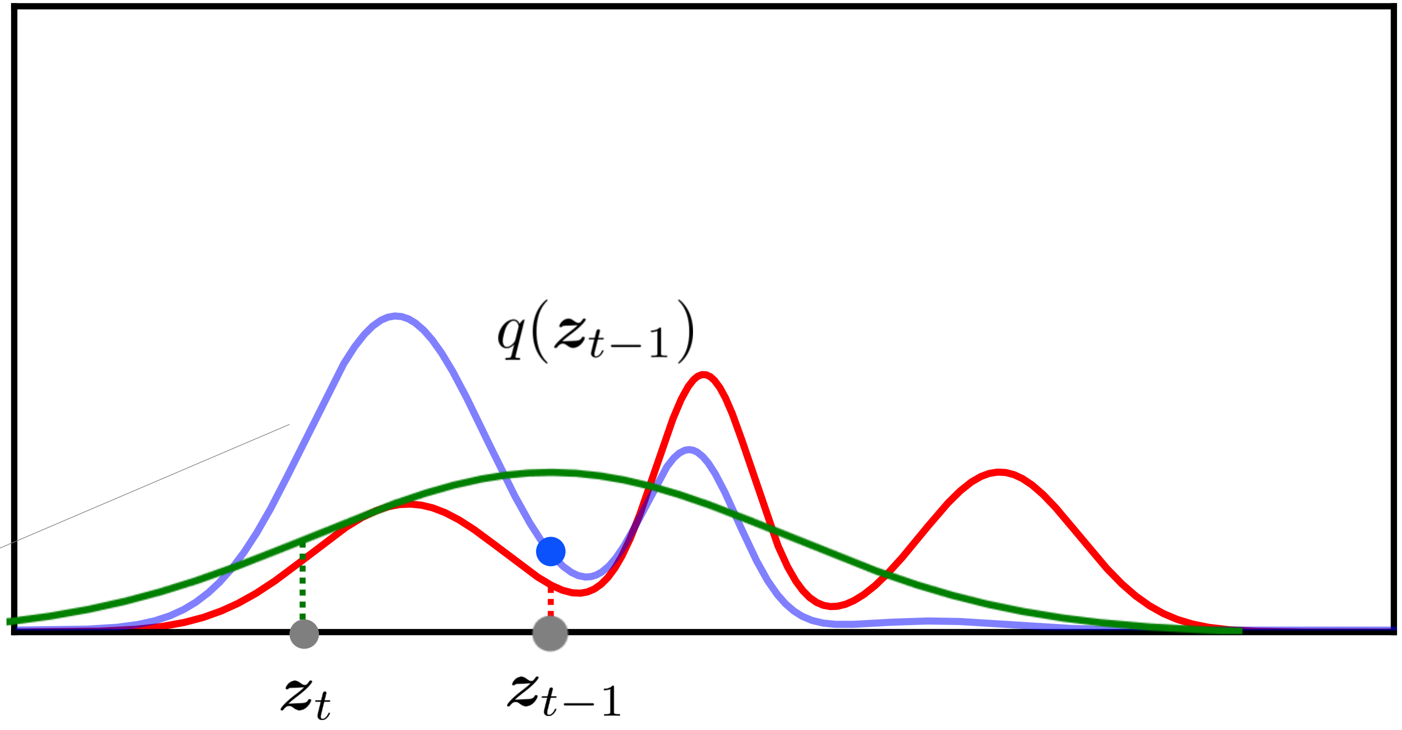
$$q(z_{t-1}|z_t) = \frac{q(z_t|z_{t-1})q(z_{t-1})}{q(z_t)} \propto q(z_t|z_{t-1})q(z_{t-1})$$

z_t *is known, hence this is constant*

[Image from https://www.bishopbook.com/]

Going backward: denoising





At training time, $\boldsymbol{z}_t$ is known (dataset + forward diffusion)

The reverse probability is the blue curve

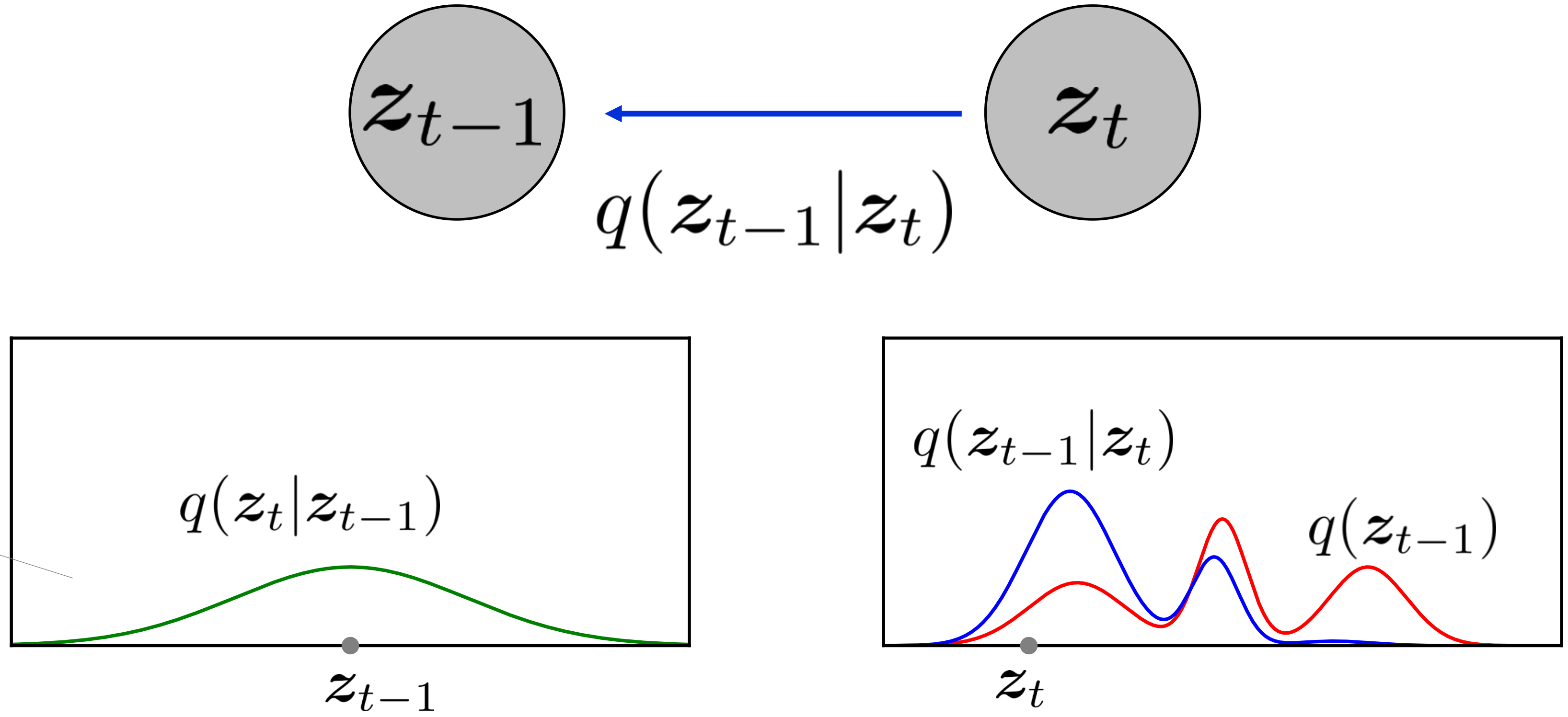
$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})q(\boldsymbol{z}_{t-1})}{q(\boldsymbol{z}_t)}$$

[Image from https://www.bishopbook.com/]

Going backward: denoising

When β_t *is large*
$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$
becomes very different from a Gaussian, hence unsuitable for training

β_t *large*

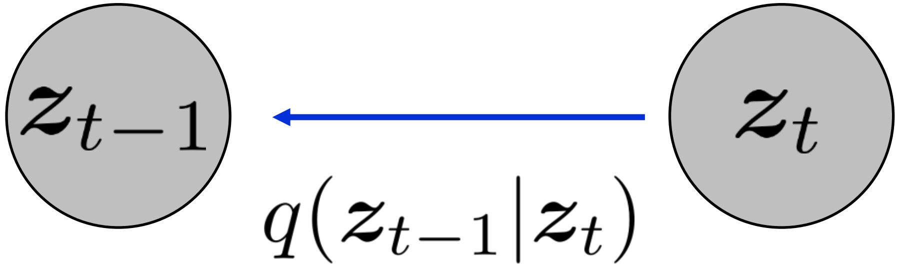


$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})q(\boldsymbol{z}_{t-1})}{q(\boldsymbol{z}_t)}$$

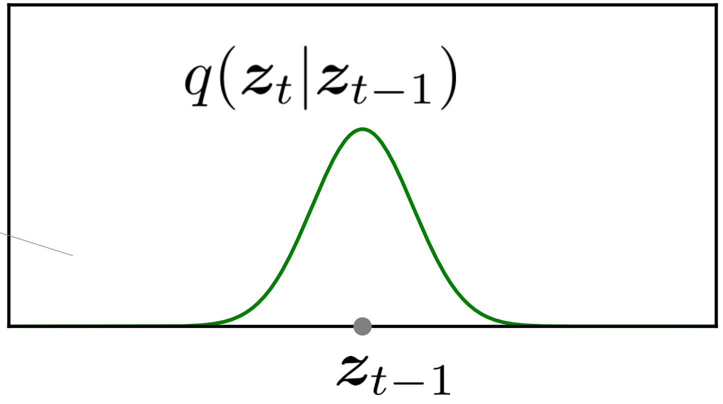
[Image from https://www.bishopbook.com/]

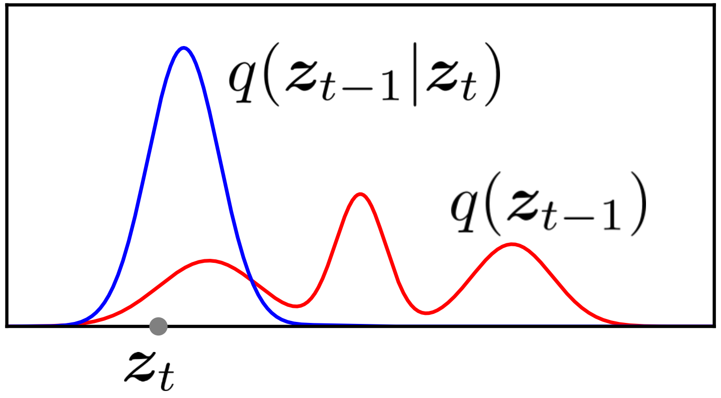
Going backward: denoising

When β_t is small
$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$
is <u>approximately Gaussian</u>



β_t small





$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})q(\boldsymbol{z}_{t-1})}{q(\boldsymbol{z}_t)}$$

Links

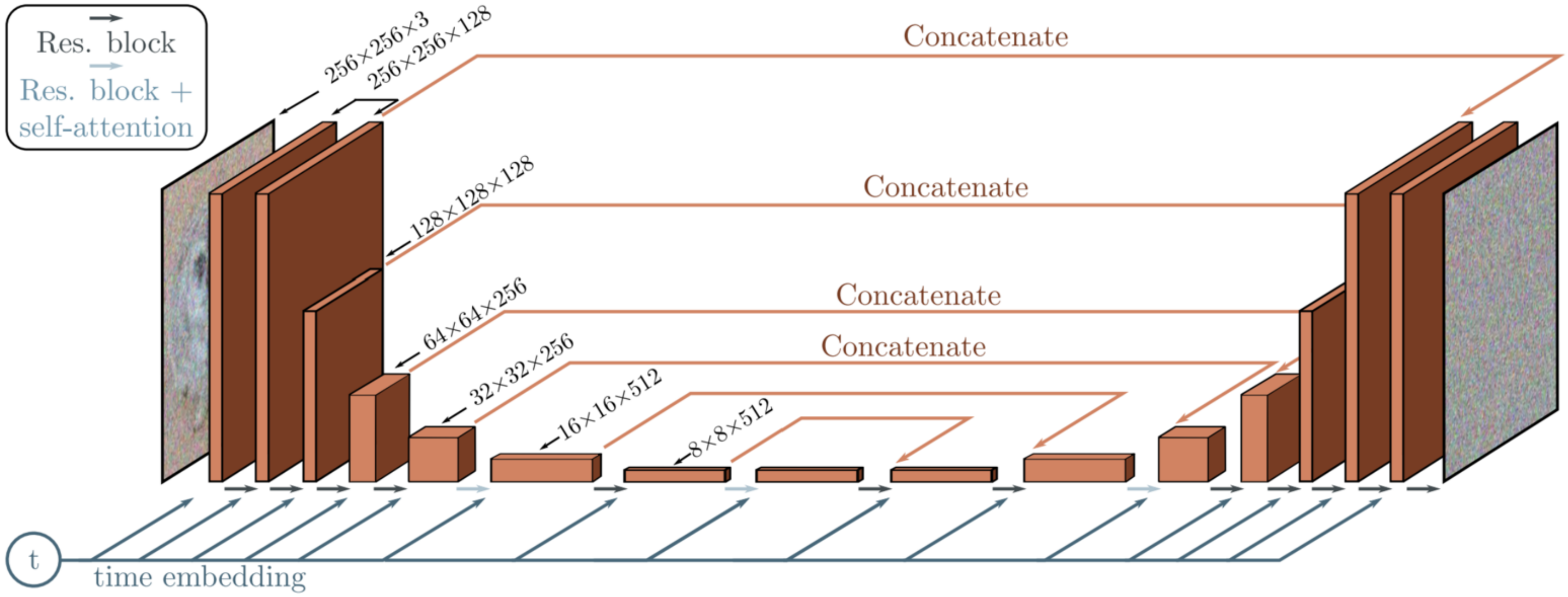
https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
https://www.superannotate.com/blog/diffusion-models
https://encord.com/blog/diffusion-models/

Practical Implementation

Conditional U-Net as basic denoising block

Loss function: $L(\boldsymbol{\vartheta}) := \| \, \boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) - \boldsymbol{\varepsilon}_t \|^2$

The network architecture for $\boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})$ is a U-Net with time embedding



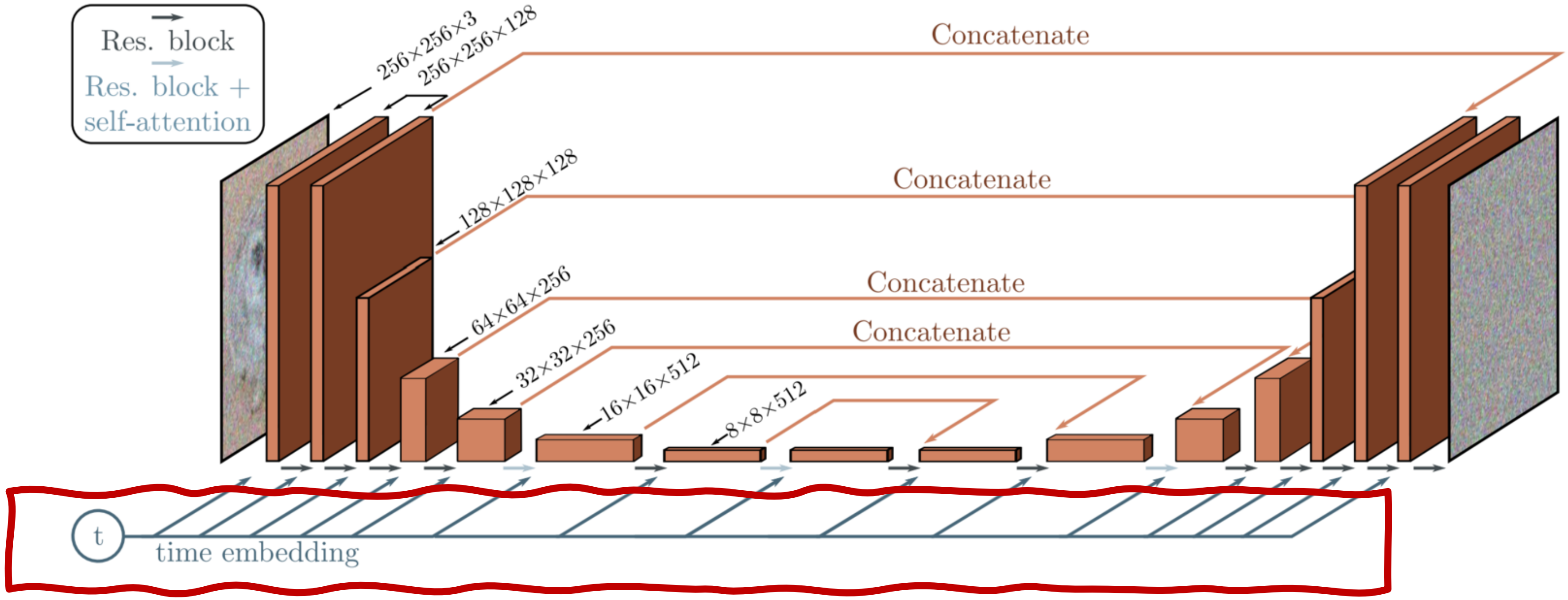
[Image from https://learnopencv.com/denoising-diffusion-probabilistic-models/]

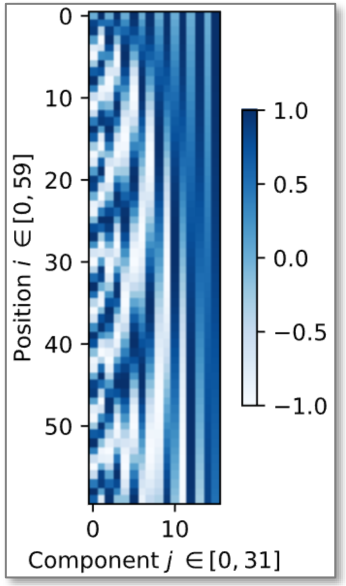
[Ho, Jain & Abbeel, 2020 - https://arxiv.org/pdf/2006.11239]

Conditional U-Net as basic denoising block

Loss function: $L(\boldsymbol{\vartheta}) := \| \boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) - \boldsymbol{\varepsilon}_t \|^2$

The network architecture for $\boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})$ *is a U-Net with time embedding*





The U-Net is conditioned by the time parameter which is embedded with sinusoidal positioning and added to each residual block

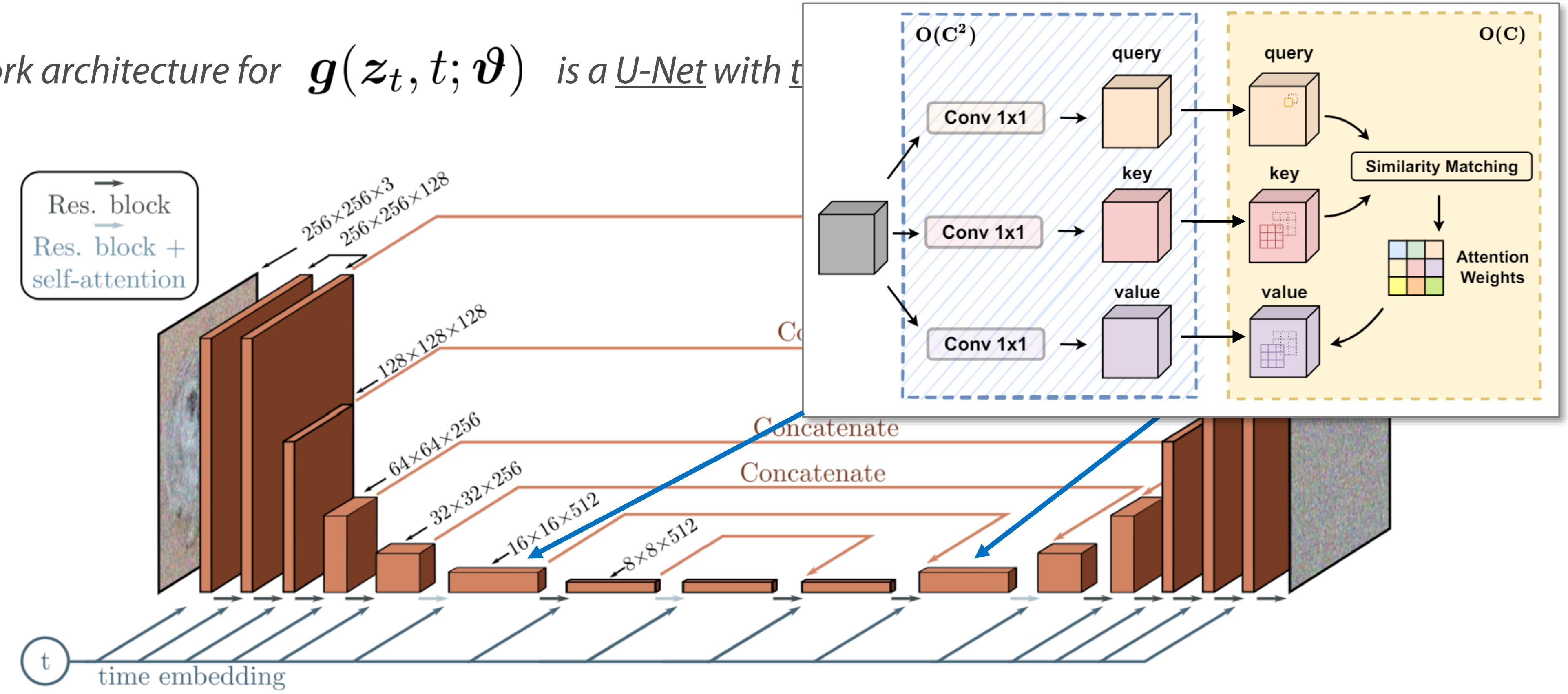
[Image from https://learnopencv.com/denoising-diffusion-probabilistic-models/]

[Ho, Jain & Abbeel, 2020 - https://arxiv.org/pdf/2006.11239]

Conditional U-Net as basic denoising block

Loss function: $L(\boldsymbol{\vartheta}) := \| \, \boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta}) - \boldsymbol{\varepsilon}_t \|^2$

The network architecture for $\boldsymbol{g}(\boldsymbol{z}_t, t; \boldsymbol{\vartheta})$ is a U-Net with t



[Image from https://learnopencv.com/denoising-diffusion-probabilistic-models/]

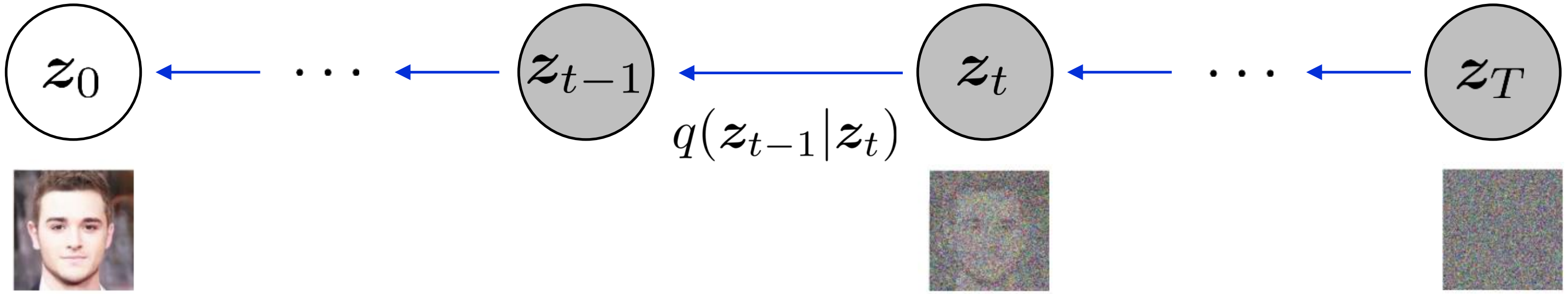
Self- Attention modules
are interspersed with convolutional blocks in the pipeline

[Ho, Jain & Abbeel, 2020 - https://arxiv.org/pdf/2006.11239]

Latent Diffusion Models

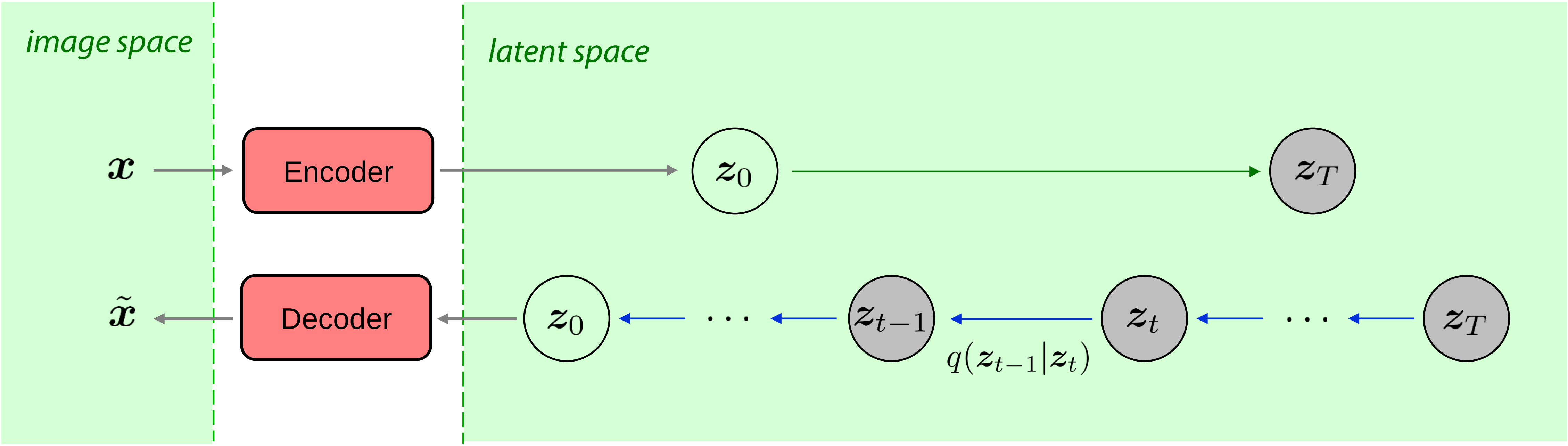
Forward Diffusion
It is relatively easy and inexpensive
(It can be performed in one step)

Backward Denoising
Must be performed in small steps
and is quite expensive,
in particular with high-resolution images



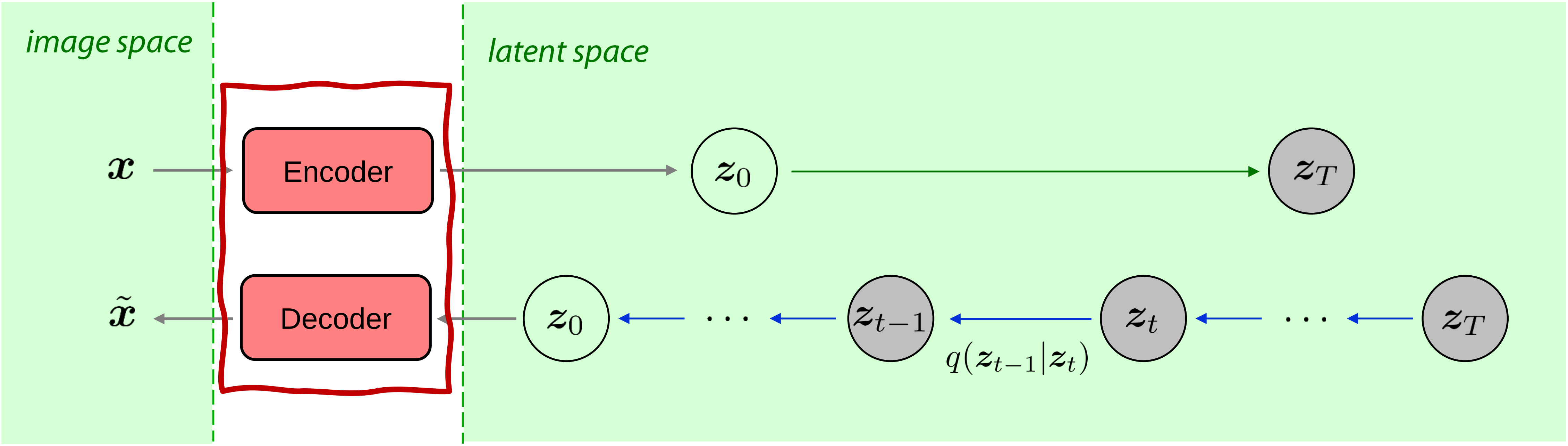
Latent Diffusion Models

Latent Diffusion Model
The intuitive idea is to perform diffusion in the *latent space*



Latent Diffusion Models

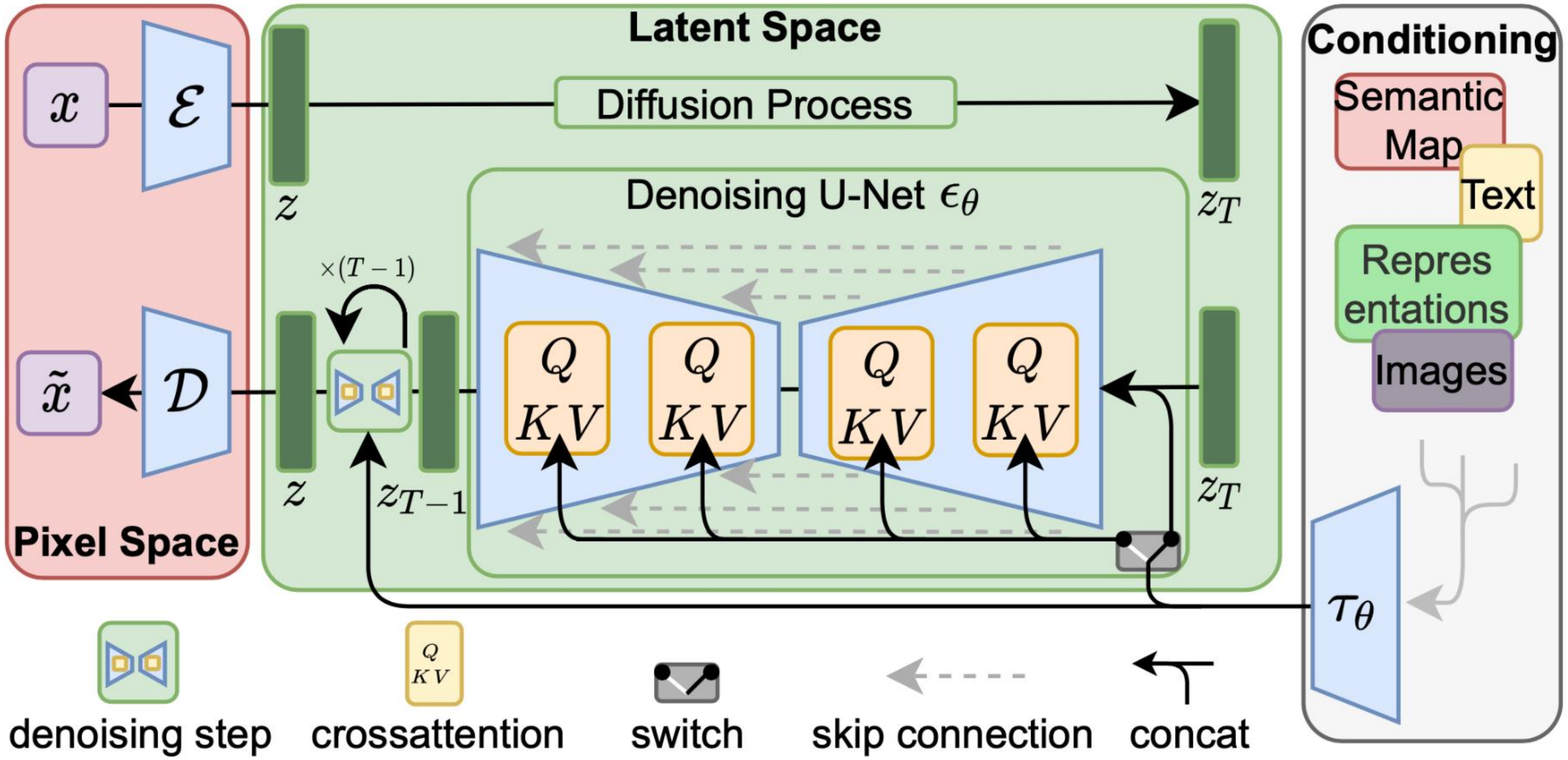
Latent Diffusion Model
The intuitive idea is to perform diffusion in the *latent space*



A pre-trained VAE is used to encode and decode high-resolution images into a suitable (reduced) latent format

Conditioning on Multimodal Labels

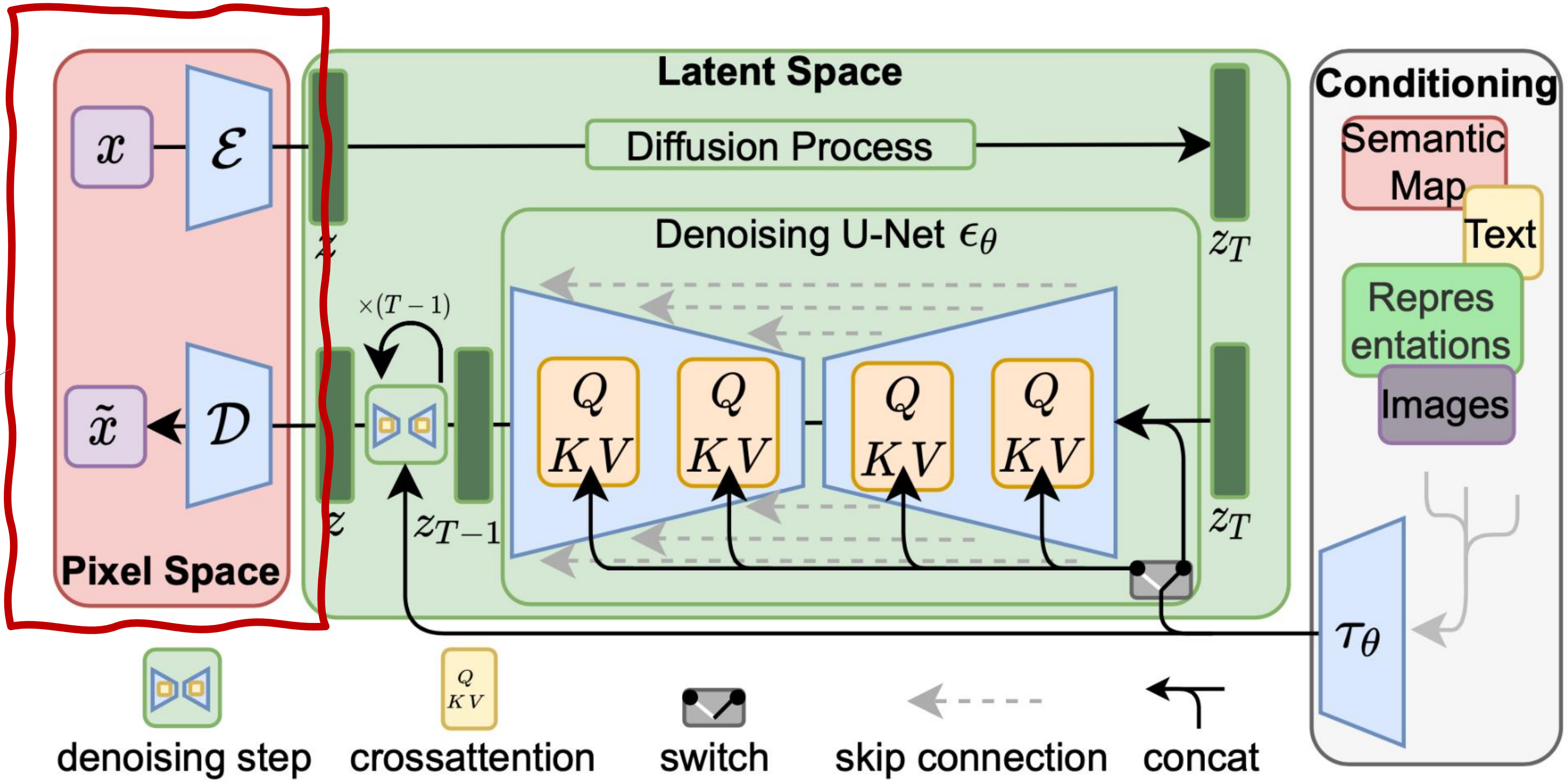
Latent Diffusion Models with Conditioning



Label $\boldsymbol{y}$ (multimodal)

[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

Latent Diffusion Models with Conditioning

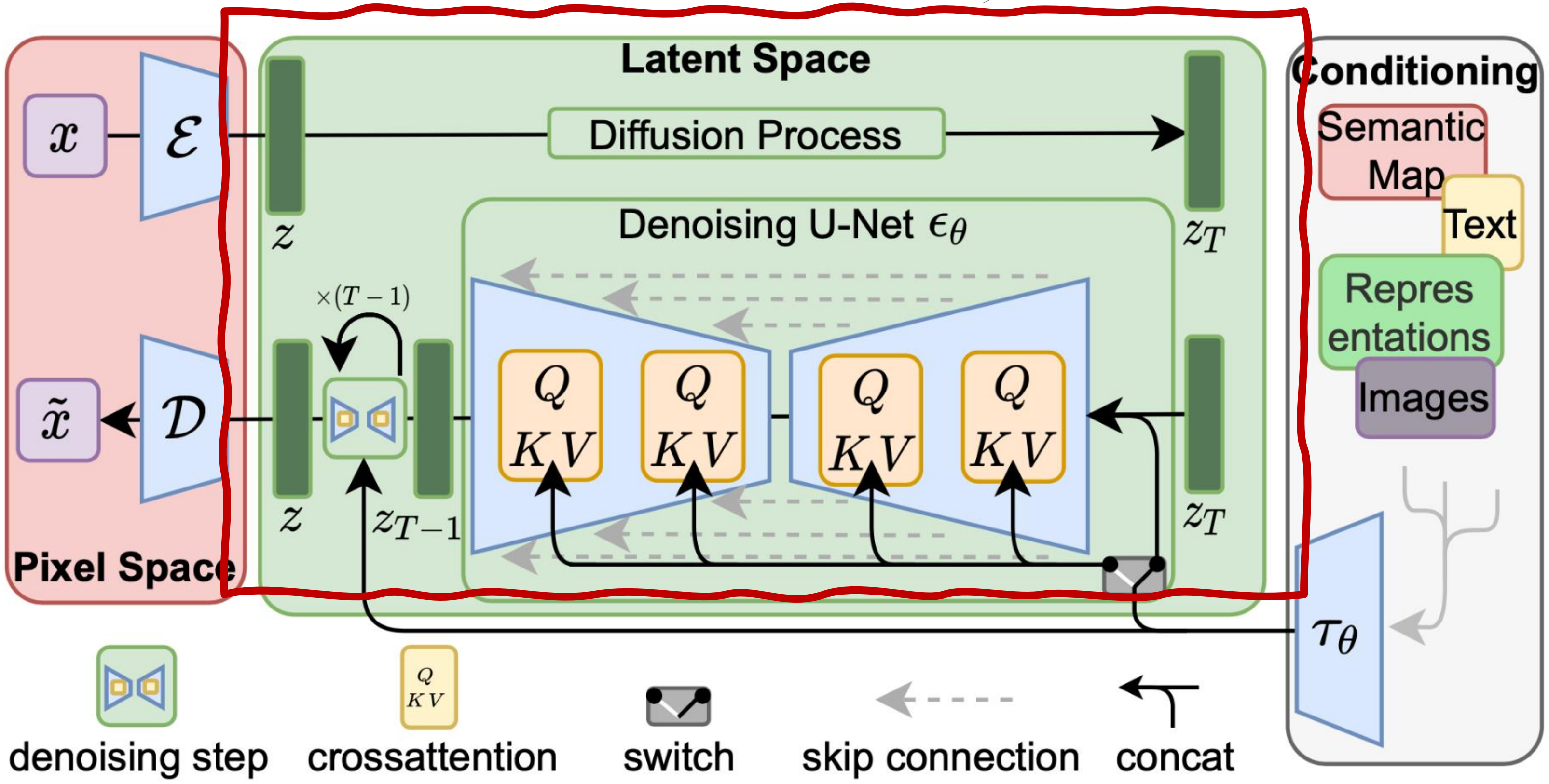


A suitable, VAE-style Encoder-Decoder is <u>pre-trained</u>

[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

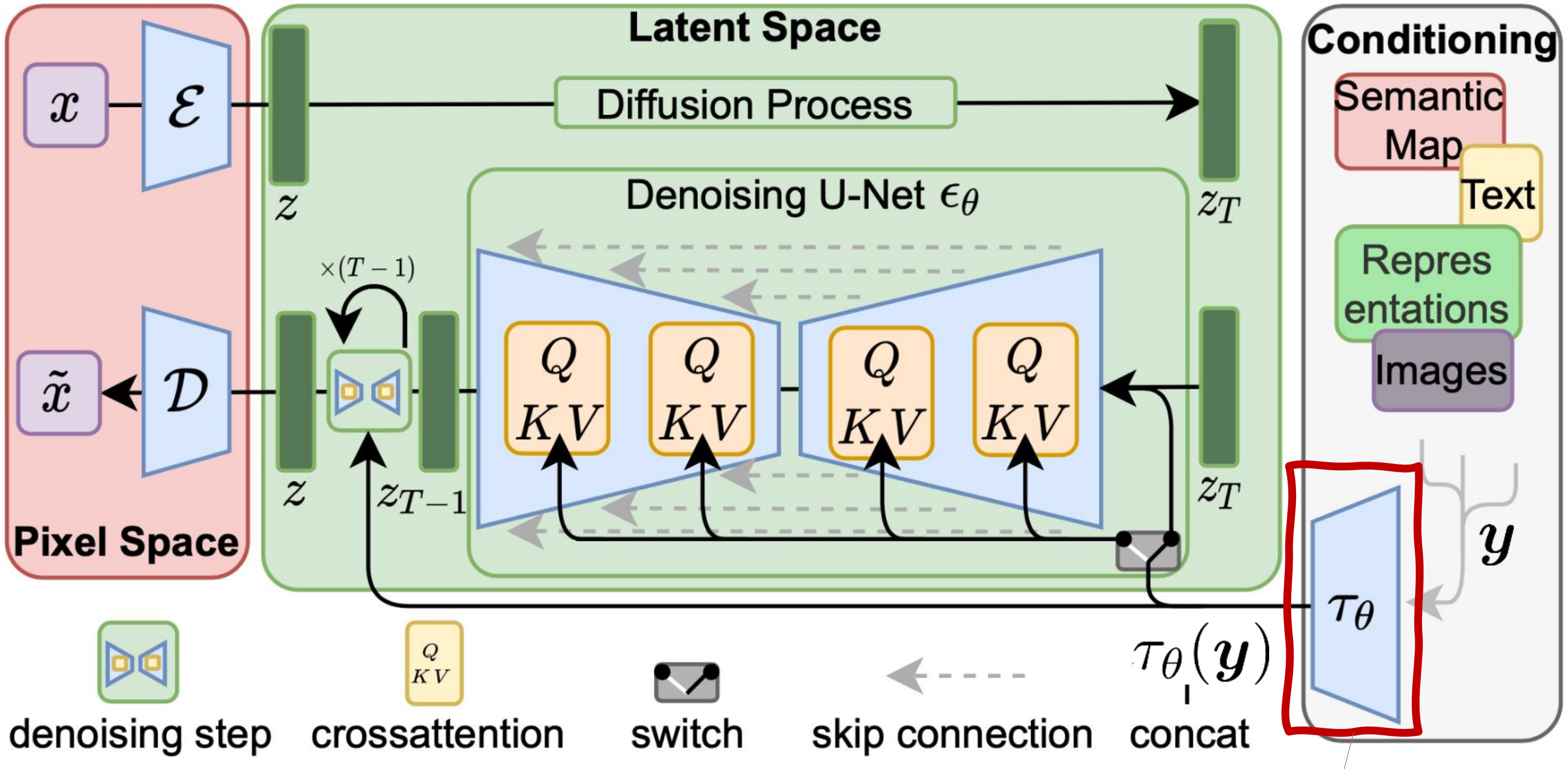
Latent Diffusion Models with Conditioning

The latent diffusion model is then pre-trained (without conditioning)



[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

Latent Diffusion Models with Conditioning

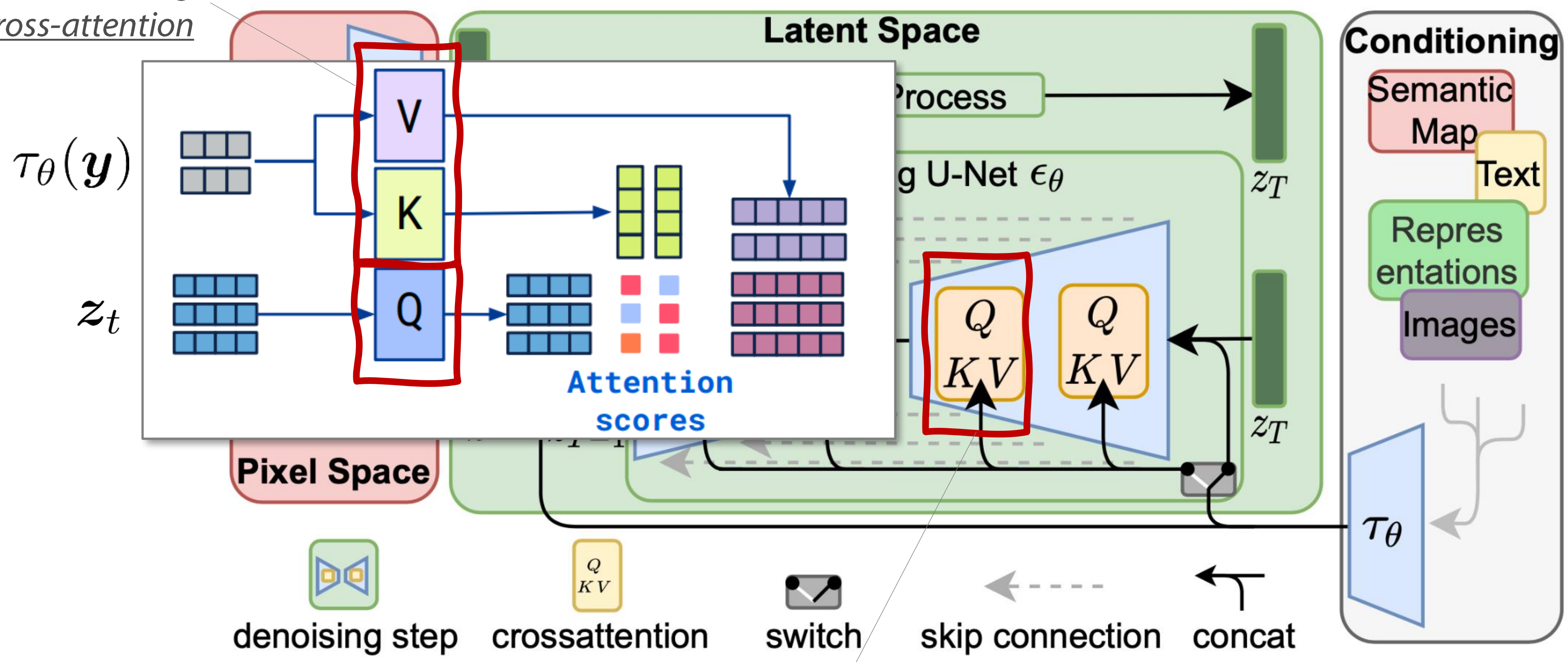


A suitable encoder
of the conditioning elements
is ***pre-trained*** *separately*

[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

Latent Diffusion Models with Conditioning

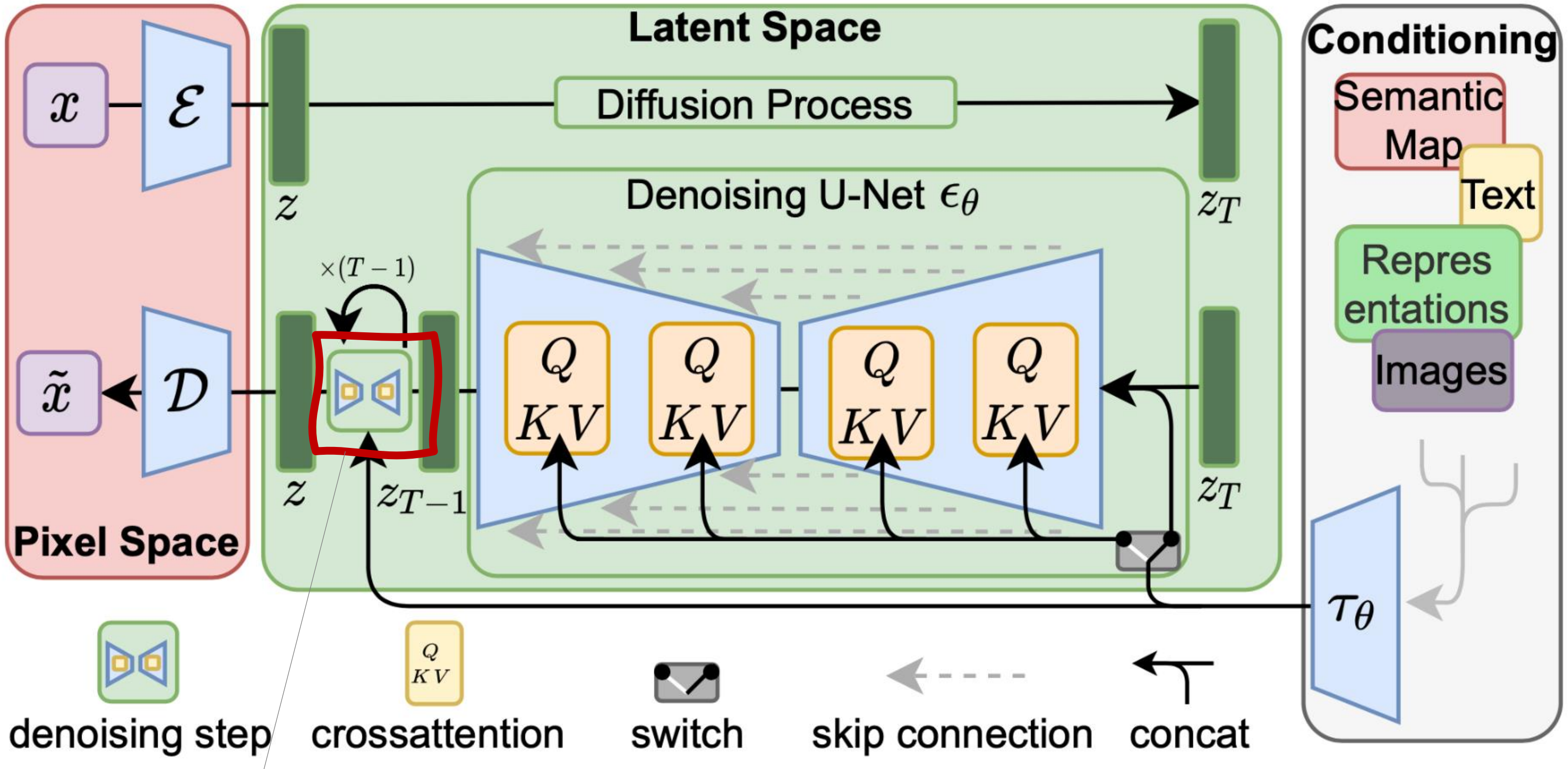
Conditioning
via <u>cross-attention</u>



Latent-space representations and embedded condition elements are combined via <u>cross-attention</u>

[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

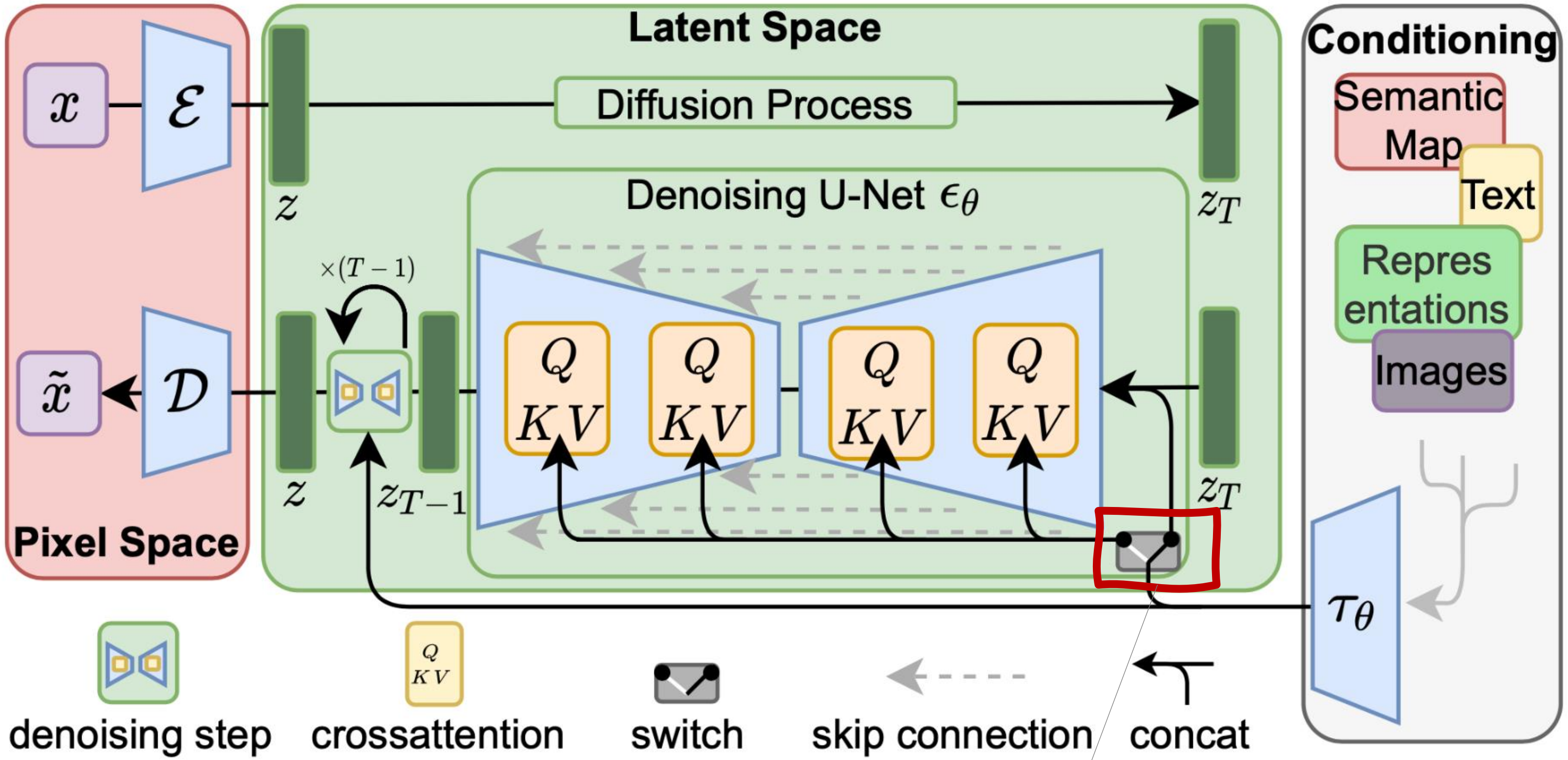
Latent Diffusion Models with Conditioning



The same step is iterated
$T-1$ *more times*

[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

Latent Diffusion Models with Conditioning

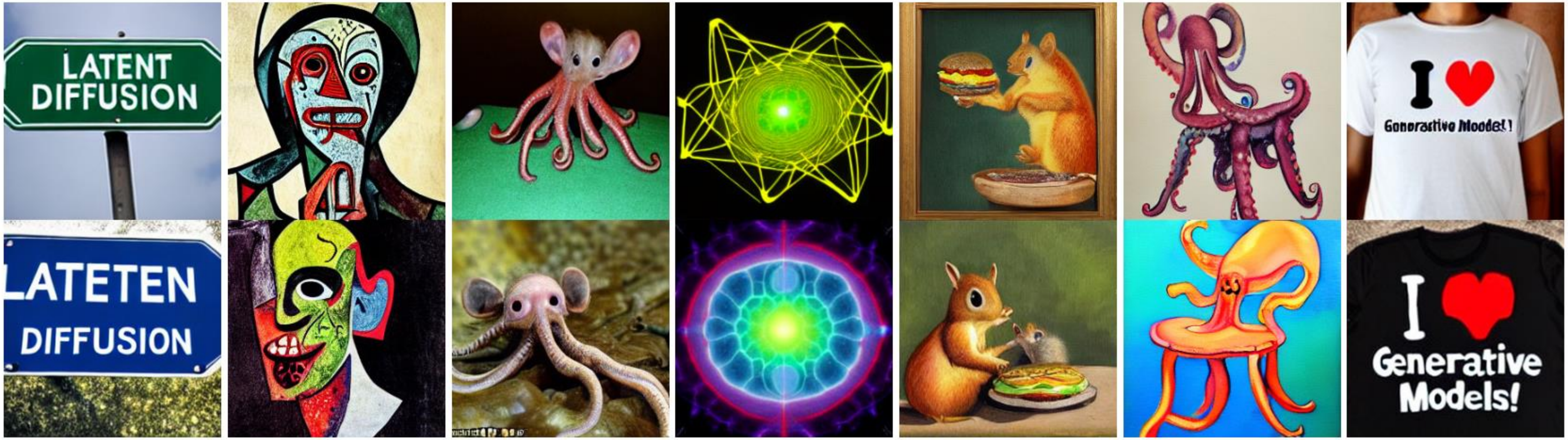


The switch is for multi-modality:
if the conditioning element is a class or text, use cross-attention,
if the input is an image, use concatenation

[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

Latent Diffusion Models with Conditioning

Text-to-Image Synthesis on LAION. 1.45B Model.						
'A street sign that reads "Latent Diffusion" '	'A zombie in the style of Picasso'	'An image of an animal half mouse half octopus'	'An illustration of a slightly conscious neural network'	'A painting of a squirrel eating a burger'	'A watercolor painting of a chair that looks like an octopus'	'A shirt with the inscription: "I love generative models!" '



[Image from Rombach et al., 2022 - https://arxiv.org/pdf/2112.10752]

Links

https://poloclub.github.io/diffusion-explainer/
https://blog.marvik.ai/2023/11/28/an-introduction-to-diffusion-models-and-stable-diffusion/
https://theaisummer.com/diffusion-models/
https://learnopencv.com/denoising-diffusion-probabilistic-models/

https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
https://www.superannotate.com/blog/diffusion-models
https://encord.com/blog/diffusion-models/