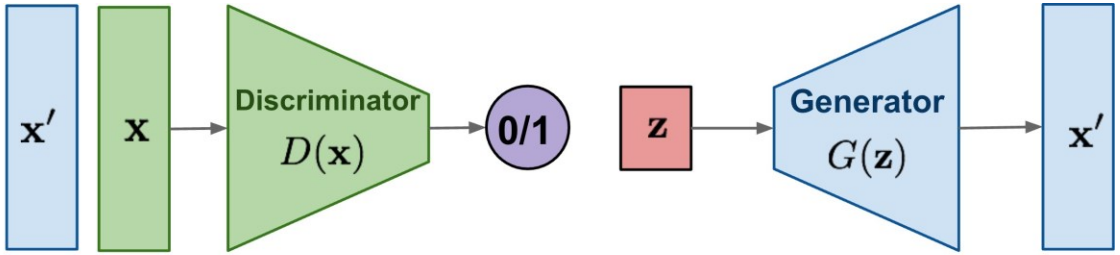
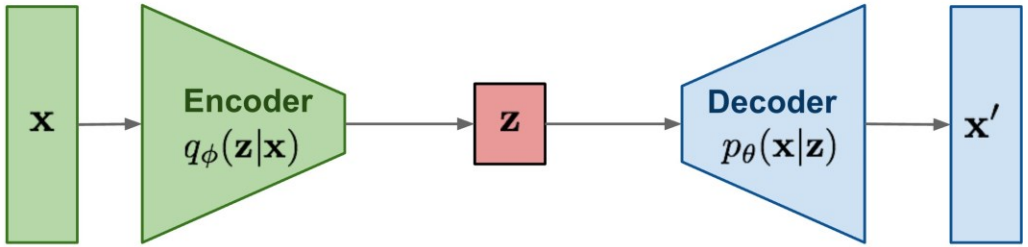
Probabilistic diffusion models

Overview of methods

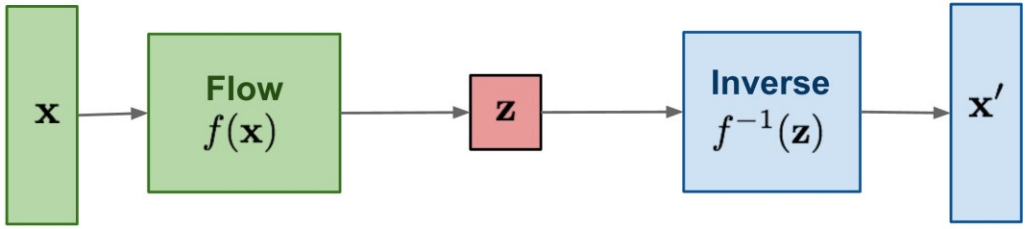
GAN: Adversarial training



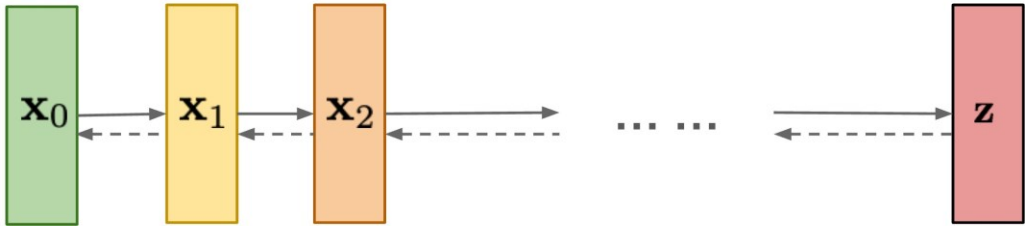
VAE: maximize variational lower bound



Flow-based models: Invertible transform of distributions

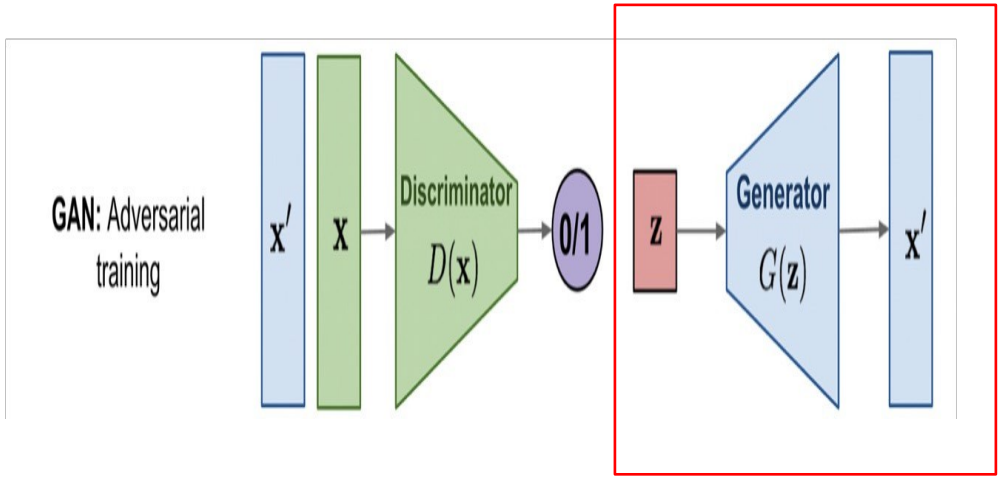


Diffusion models: Gradually add Gaussian noise and then reverse



GANs

- Convert latent noise vector to target distribution in one step

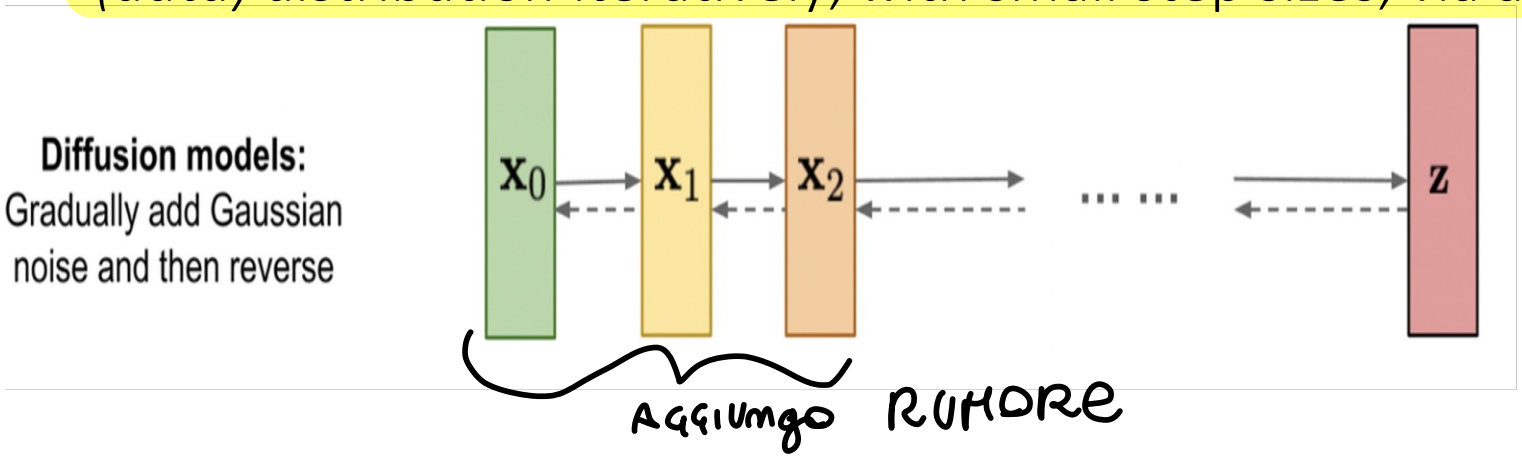


- Lots of issues with training:
  - Vanishing gradients: if discriminator is too good, gradients go to zero
  - Mode collapse
  - Learning Instability

Probabilistic diffusion

Idea

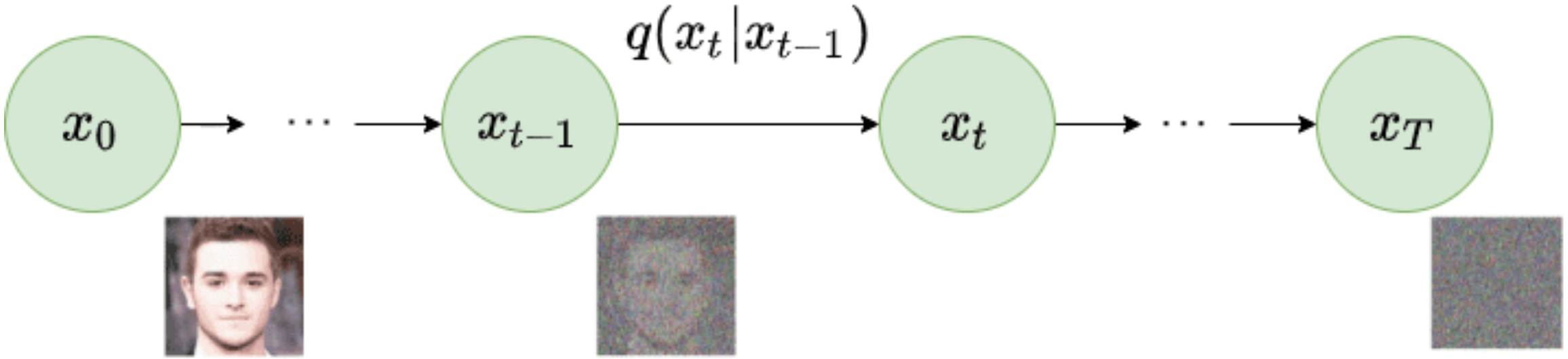
- Estimating and analyzing small step sizes is more tractable/easier than a single step from random noise to the learned distribution
- Convert a well-known and simple *base distribution* (like a Gaussian) to the *target (data) distribution* iteratively, with small step sizes, via a Markov chain:



- Markov chain: outlines the probability associated with a sequence of events occurring based on the state in the previous event.

Basic idea: learning how to denoise

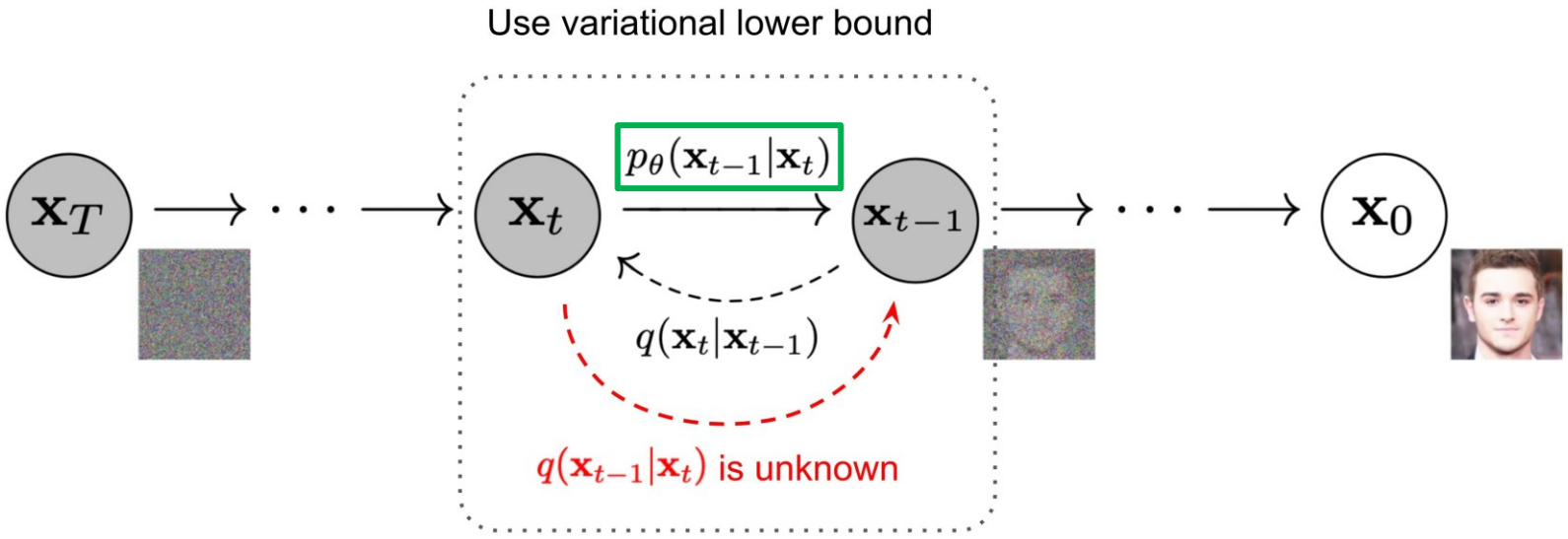
- Imagine we gradually add Gaussian noise to an image, until it's completely Gaussian normally distributed.
  - $q(x_t|x_{t-1})$ is the probability of having a noised image x_t conditioned on the current image x_{t-1}



Basic idea: learning how to denoise

- Now if we could simply learn how to reverse the process, we could generate images
  - By starting from random noise
  - Since the noise is gaussian distributed, then we know how to generate it

  - $p_0(x_{t-1} | x_t)$ is the verted probability that gives the probability of the denoised image x_{t-1} conditioned on the current image x_t



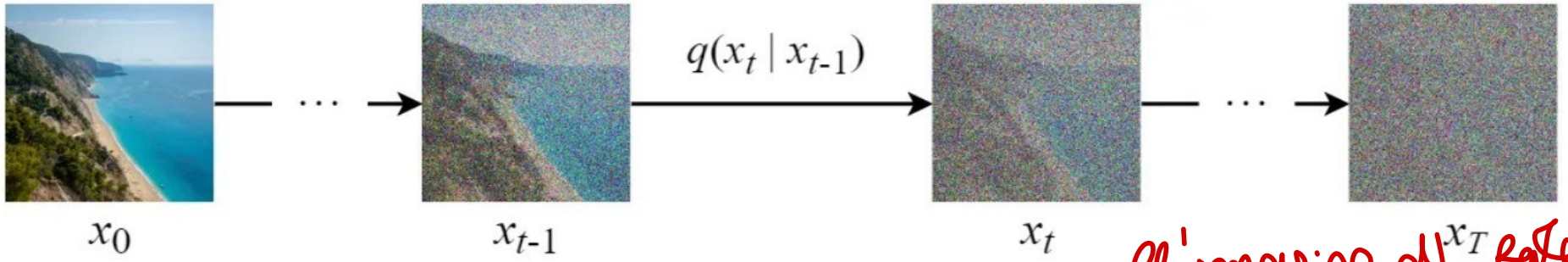
Diffusion Model

The intuitive idea

- Build a sort Stochastic autoencoder
  - encoder introduces noise and decoder denoises the data
  - We will see that the encoder is a just a stochastic proess (not a neural network), whereas the decoder is neural network (a Unet)

- Generates better data than variational autoencoders
- Easier to train than generative adversarial networks and does not suffer from mode collapse, learning instability

Forward Diffusion Process



Distribution of the noised images | Output | Mean μ_t | Variance Σ_t

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

Notations:

- t : time step (from 0 to T)
- x_0 : a data sampled from the real data distrition $q(x)$ (i.e. $x_0 \sim q(x)$)
- β_t : variance schedule ($0 \leq \beta_t \leq 1$, and β_0 = small number, β_T = large number)
- I : identity matrix

From Steins (medium.com)

Forward diffusion process

- Noise added can be parameterized by:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \{\beta_t \in (0,1)\}_{t=1}^{T}$$

- Vary the parameters of the Gaussian according to a *noise schedule:*
  - β_t changes with time t
  - It has been tested linea scheduler $\beta_t = t\beta + \alpha$

- You can prove with some math that as T approaches infinity, you eventually end up with an Isotropic Gaussian (i.e. pure random noise)
- Note: forward process is fixed not trained

↓
siccare la distribuzione e gaussiana ⇒ senza buchi: ⇒ se prendo un punto a caso sono sicuro che è associato qualche immagine

Stochastic Transformation

- With reparametrization trick, we can more explicitly define how a noise image depends on the previous one

  - When $x \sim P(x) = N(x|\mu, \sigma^2)$
    then $x = \sigma\epsilon + \mu$ where $\epsilon \sim N(\epsilon|0,1)$

- Since $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = N\left(\boldsymbol{x}_t\middle|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I}\right)$
  Then $\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{I})$

Stochastic Transformation

- The noised image x_t at time t is defined by adding noise t times, but we can obtain it adding the whole noise in a single step

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

where $\boldsymbol{\epsilon} \sim N(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{1})$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i \text{ and } \alpha_i = 1 - \beta_i$$

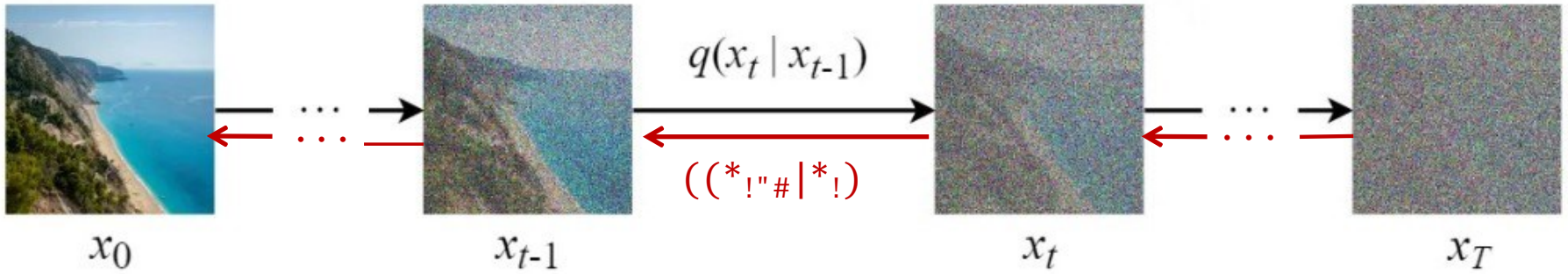
- In the limit x_t is a random vector from an isotropic Gaussian

$$\lim_{t\to\infty} \boldsymbol{x}_t = \sqrt{\bar{\alpha}_\infty}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_\infty}\boldsymbol{\epsilon} = \boldsymbol{\epsilon} \text{ since } \bar{\alpha}_\infty \to 0$$

in addestramento
genero immagini rumore
percorrendo un passo casuale
($T=1,\dots$)
es: prendo un
immagine e
vado a $t=25$
prendo un altra
immagine e vado
a $t=100$

Reverse Denoising Process

From Steins (medium.com)



- The nloising process is Gaussian
- Forward factorization: $q(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = q(\boldsymbol{x}_0) \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$
- Reverse factorization: $q(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \prod_{t=1}^{T} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\, q(\boldsymbol{x}_T)$
  - Since joint distribution is Gaussian then $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is also Gaussian
  - $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = N(\boldsymbol{x}_{t-1}|\tilde{\mu}_t(\boldsymbol{x}_t, t), \sigma_t \boldsymbol{I})$

Reverse Conditional Gaussian

- The reverse conditional $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = N(\boldsymbol{x}_{t-1}|\tilde{\mu}_t(\boldsymbol{x}_t, t), \sigma_t \boldsymbol{I})$ does not have a closed form, but Ho, Jain and Abbeel (2020) derived the following approximation

$$\tilde{\mu}_t(\boldsymbol{x}_t, t) \approx \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$$

where $\boldsymbol{\epsilon}_t = N(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{I})$ is the noise introduced at step t

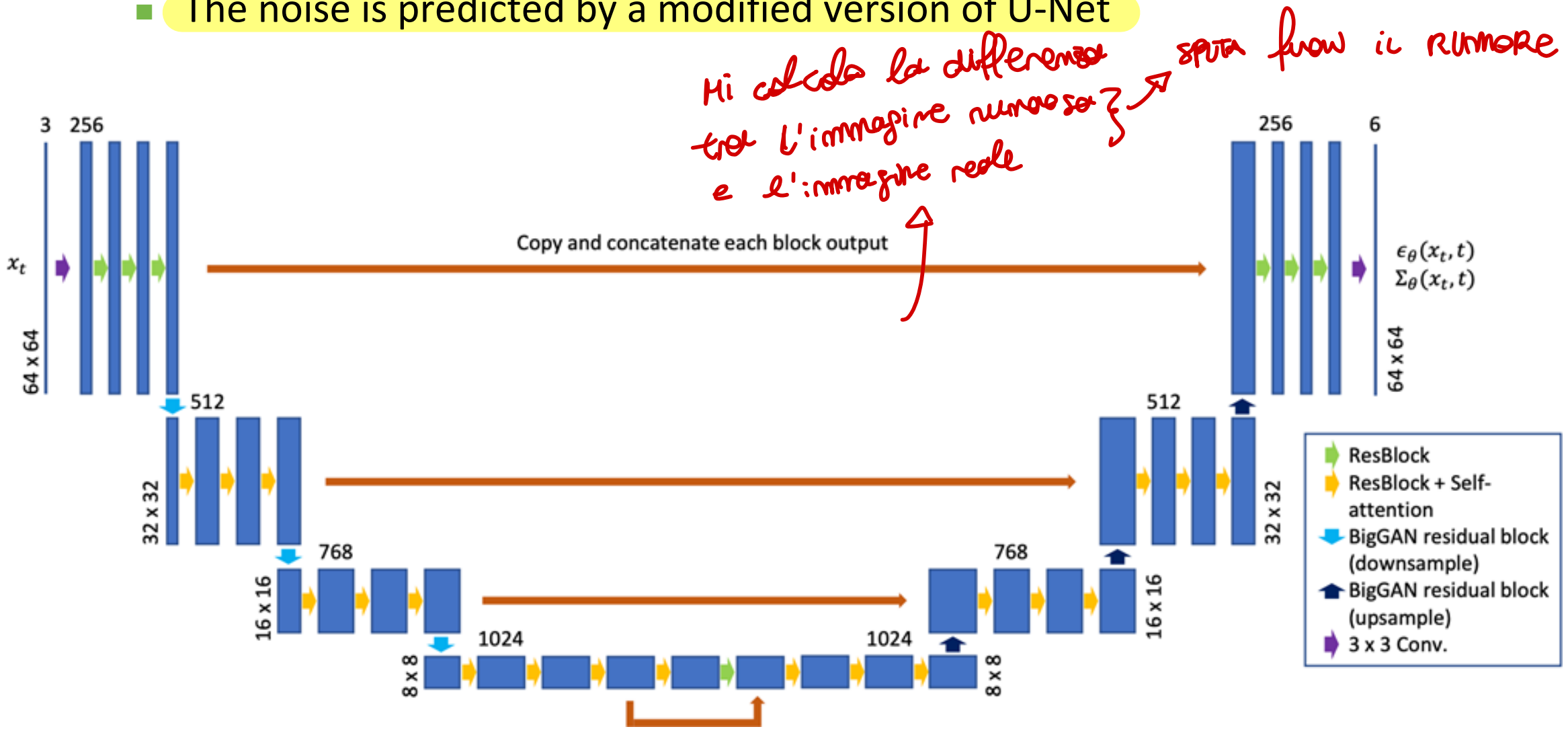
- We do not know $\boldsymbol{\epsilon}_t$, but we can train a neural network $\epsilon_\theta(\boldsymbol{x}_t, t)$ to approximate it:

$$\text{Minimize } L(\theta) = \left|\left|\boldsymbol{\epsilon}_t - \epsilon_\theta(\boldsymbol{x}_t, t)\right|\right|_2^2$$

target　　noise predictor

UNet

- The noise is predicted by a modified version of U-Net



Why Minimize $L(\theta) = ||\epsilon_t - \epsilon_\theta(x_t, t)||_2^2$?

- As it usually happens in generative models, we want to otimize

$$L = -log(p_\theta(x_0))$$

immagine d'ingresso

- It can be proved (not in this slides) that the following is a variational lower bount to

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$

where

$$L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$

$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$

$$L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$$

non la considero perché immagine reale e immagine completamente rumorosa

- Moreover, L_T does not depend on the neural network and can be skiped
- Experimentally L_0 does not improve the results
- Thus, only L_t has to be considered

Why ?

$$L_t = D_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$

- Intutively, the KL divergence implies that the two distributions are as close as possible
- By the defintion, p and q are close if ε_t and $\varepsilon_0(x_t,t)$ are close:
  - This is because p and q are Gaussian distributions with mean and deviations that depends p and q have gaussian distribution that depend on $\varepsilon_0(x_t,t)$ and ε_t , respectively
- Thus, we can minimize

$$\text{Minimize } L(\theta) = \left|\left|\boldsymbol{\epsilon}_t - \epsilon_\theta(\boldsymbol{x}_t, t)\right|\right|_2^2$$

Training Algorithm

Training: repeat

- Select an image
- Pick a time t and a noise ε from
- Add the noise to the image with the formula for time t
- Make a learning step (ε is the target to predict from image and time)

Repeat

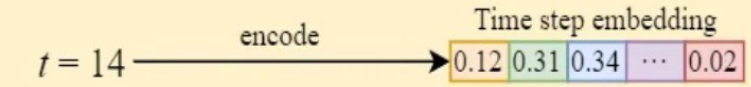
- $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
- $t \sim uniform(\{1, \dots, T\})$
- $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{I})$
- $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \nabla_{\boldsymbol{\theta}} \left|\left|\epsilon - \epsilon_\theta(\boldsymbol{x}_t, t)\right|\right|_2^2$

Until convergence

Training Algorithm

For each training step:

1. Randomly select a time step & encode it



2. Add noise to image

$$x_t = \sqrt{\bar{a}_t}\, x_0 + \sqrt{1 - \bar{a}_t}\, \varepsilon$$

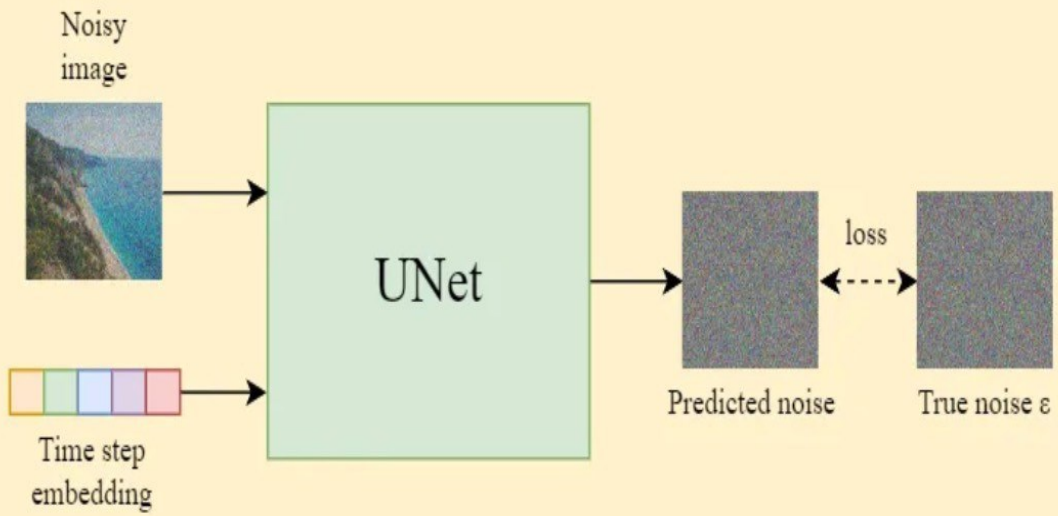
Adjust the amount of noise according to the time step t

$$\varepsilon \sim \mathcal{N}(0, 1)$$

$$\alpha_t = 1 - \beta_t$$

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

3. Train the UNet



From Steins (medium.com)

Training Algorithm

why

- we are not learning to reverse the whole denoising sequence, but a random single step at a time
  - It is easier to get varied distribution of the examples in the batches/tran set
- We are leaning to predict the noise ε and not the noised image x_{t-1}
  - Diffusion models employ a U-Net for noise prediction. With skip connection, the U-Net, at the last layer, can predict x_{t-1} and subtract it from input x_{t-1} ouputting noise ε, or it could not subract so that the output is x_{t-1}. Thus, the two approaches are alm[illegible] being bette of being normally distributed.

Repeat

- $\boldsymbol{x}_0 \sim q(\boldsymbol{x_0})$
- $t \sim uniform(\{1, \dots, T\})$
- $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{I})$
- $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta\nabla_{\boldsymbol{\theta}}\left|\left|\epsilon - \epsilon_\theta(\boldsymbol{x}_t, t)\right|\right|_2^2$

Until convergence

Data Generation Algorithm

- After the U-Net is trained, it can be used to generate new images from noise
- The images are generated iteratevely removing the noise, which is predicted by the U-Net

- $\boldsymbol{x}_T \sim N(\boldsymbol{x}|\mathbf{0}, \boldsymbol{I})$
- For $t = T, \ldots, 1$ do
  - $\boldsymbol{\epsilon} \sim N(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\boldsymbol{\epsilon} = \mathbf{0}$
  - $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)\right) + \sqrt{\sigma_t}\boldsymbol{\epsilon}$
- Return $\boldsymbol{x}_0$

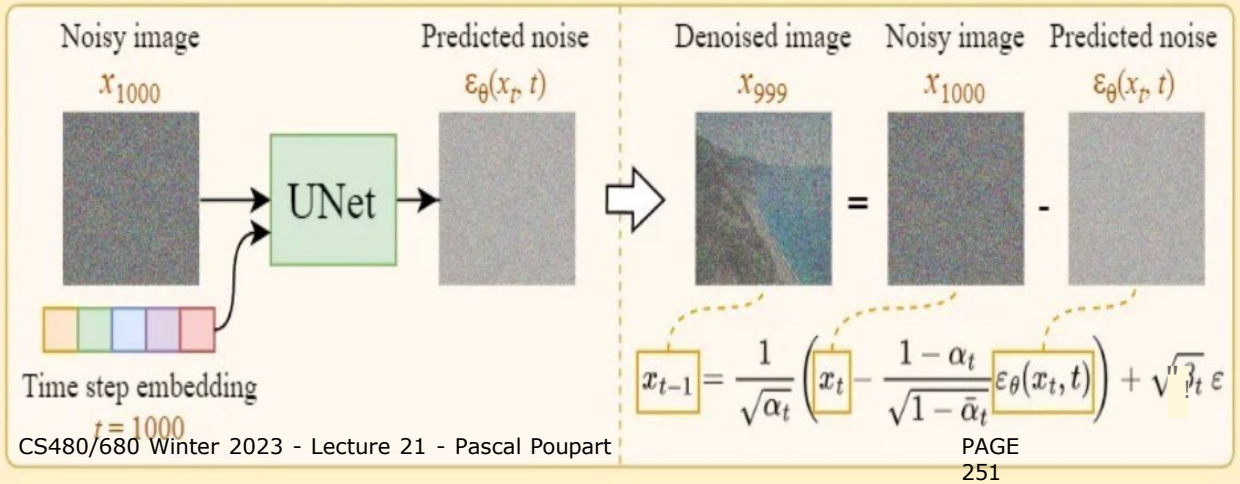
Data Generation Algorithm

1. Sample a Gaussian noise

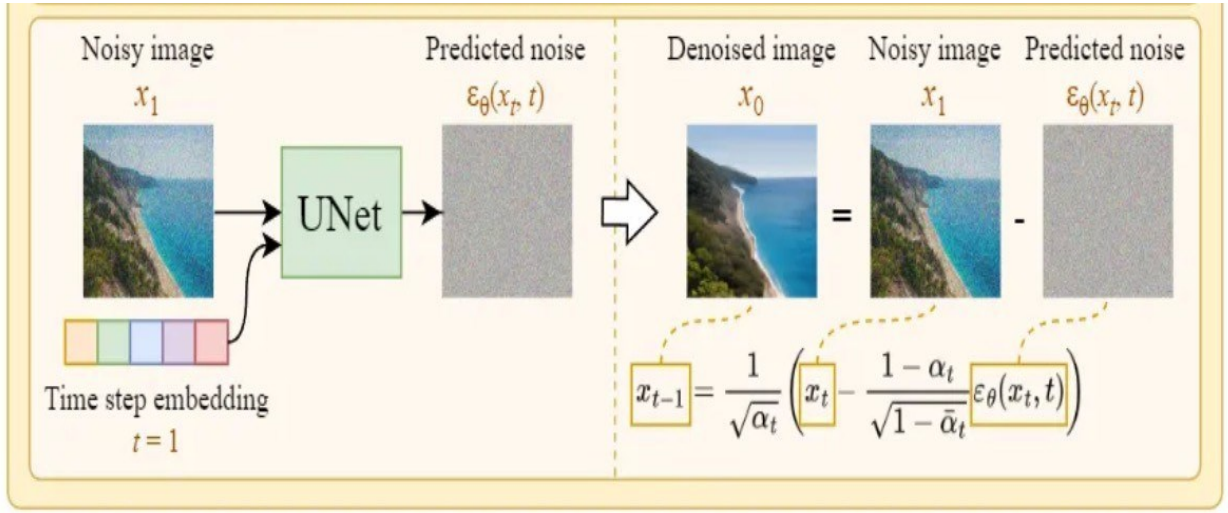
$x_T \sim N(0, I)$ E.g. $T = 1000$

$x_{1000} \sim N(0, I)$

2. Iteratively denoise the image



. . .



3. Output the denoised image

Denoised image x_0

From Steins (medium.com)

The performance

- Diffusion models are now the state of the art in conditional and unconditional image generation
- Even if training is very expensive

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

Results

Results

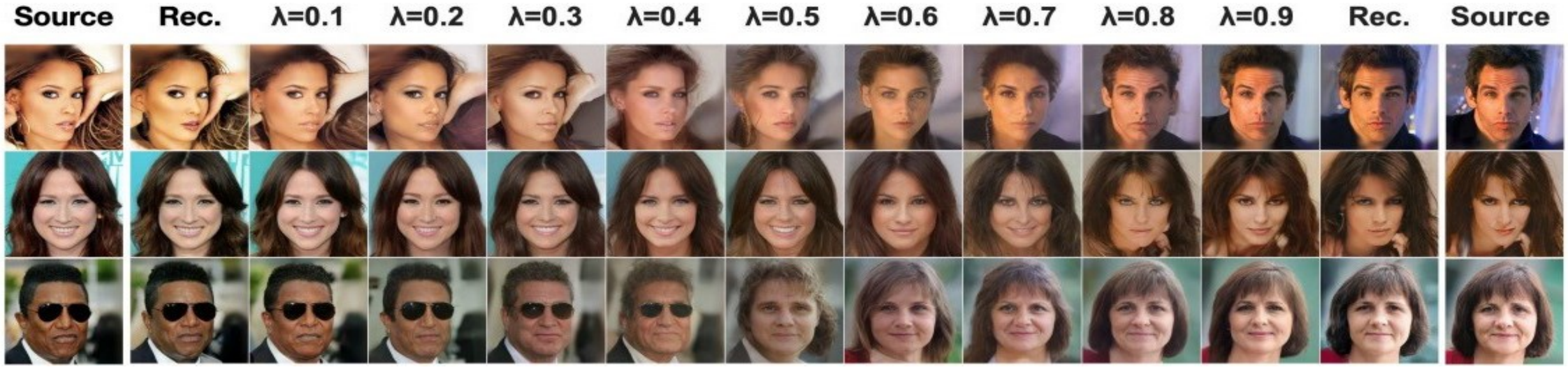
Ho, Jain and Abbeel (2020)

Figure 3: LSUN Church samples. FID=7.89

Figure 4: LSUN Bedroom samples. FID=4.90

Results

- Similarly to other approaches, we can generate newimages by interpolation between two



Latent diffusion models (a.k.a. Stable Diffusion)

- State of the art for image generation
- Stable diffusion, DALL E 3… are based on this approach
- Particularly suited for text-to-image generation

Main ideas

- Diffusion is carried in a latent space and not directly on images
- The conditioning is added to the inner layers of denoising UNet, not in input
- The architecture of the denoisinig networks exploit most recent solutions, including transformers layers and other tricks

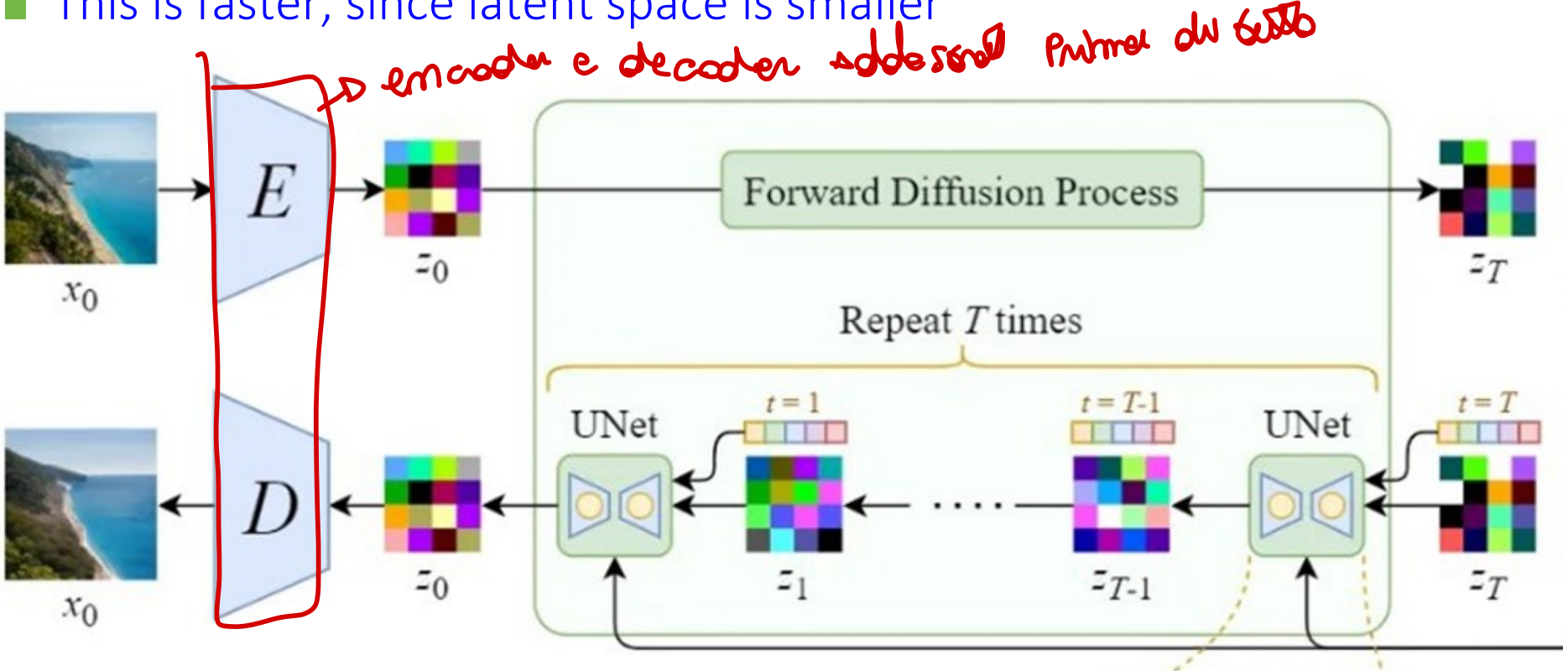
Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models

Latent Diffusion Model

Diffusion is carried in a latent space and not directly on images

- An encoder E and a decoder D project the input image into a latent space
- Noising and denoising is carried out in this space

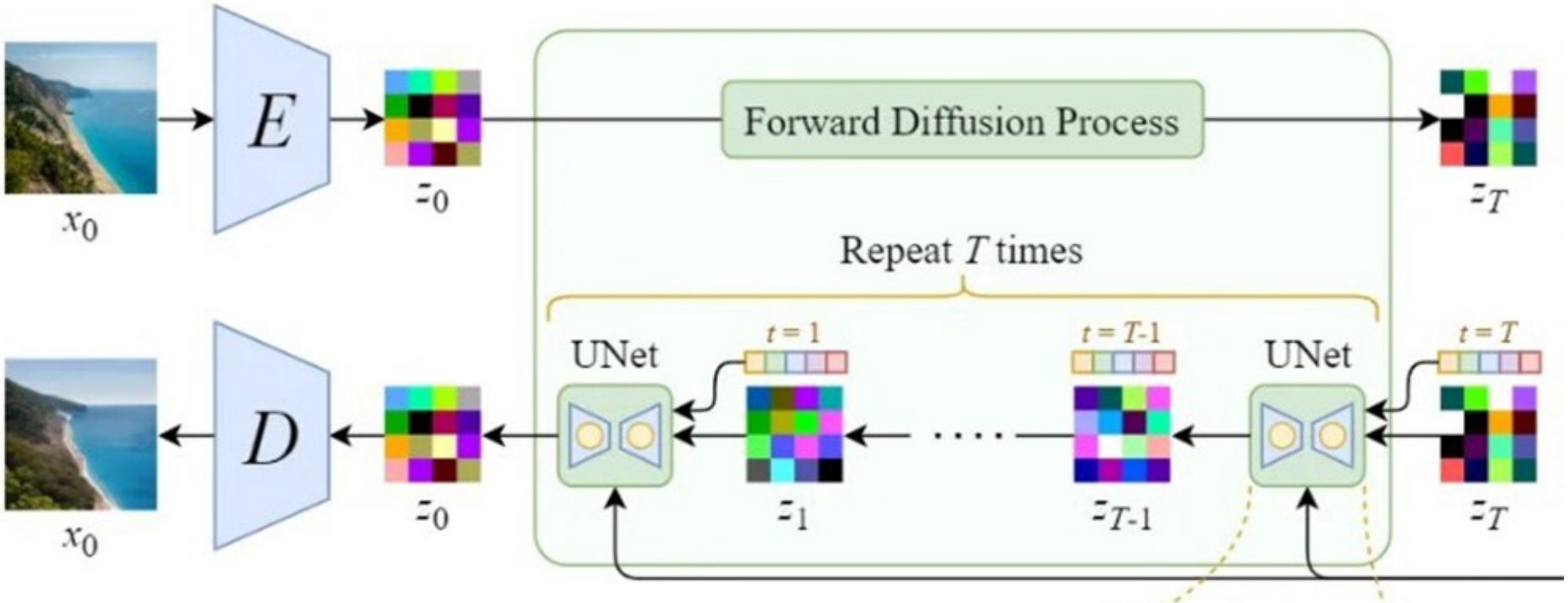
- This is faster, since latent space is smaller



Latent Diffusion Model

Diffusion is carried in a latent space and not directly on images

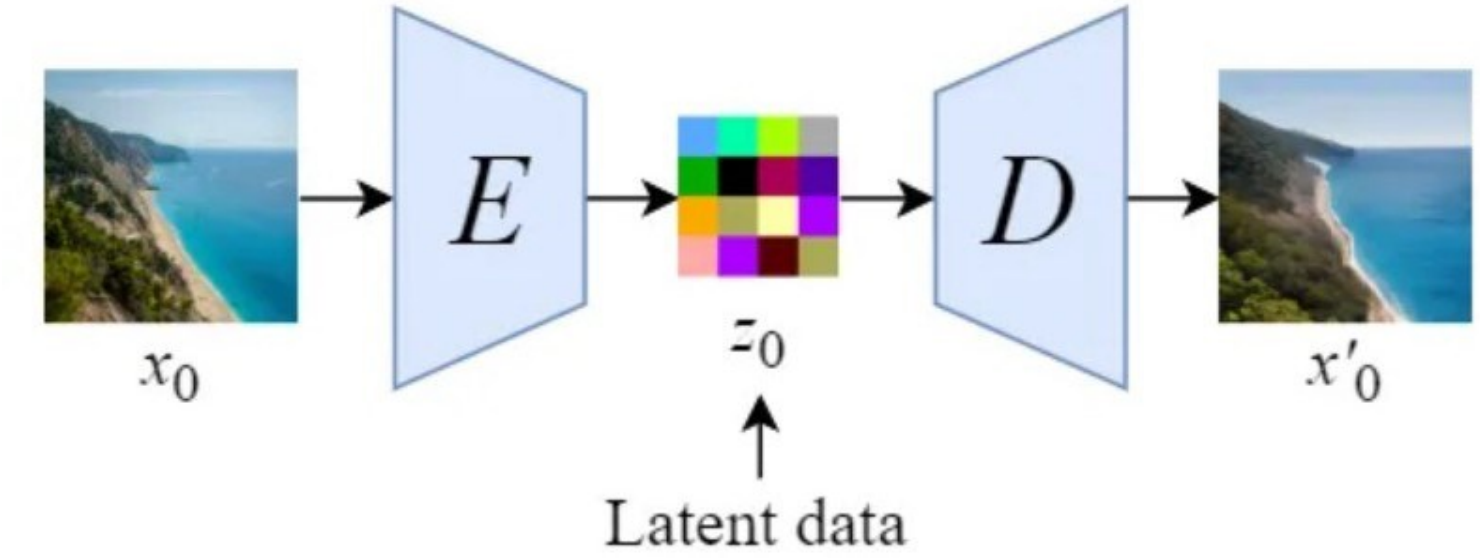
1. The image is transformed to latent space by E
2. In training, a diffusion process forward process is applied to latent representation, adding noise
3. A Unet implement the backward diffusion and denoise the latent representation
4. The denoised latent representation is transformed back to an image by D



Latent Diffusion Model

Diffusion is carried in a latent space and not directly on images

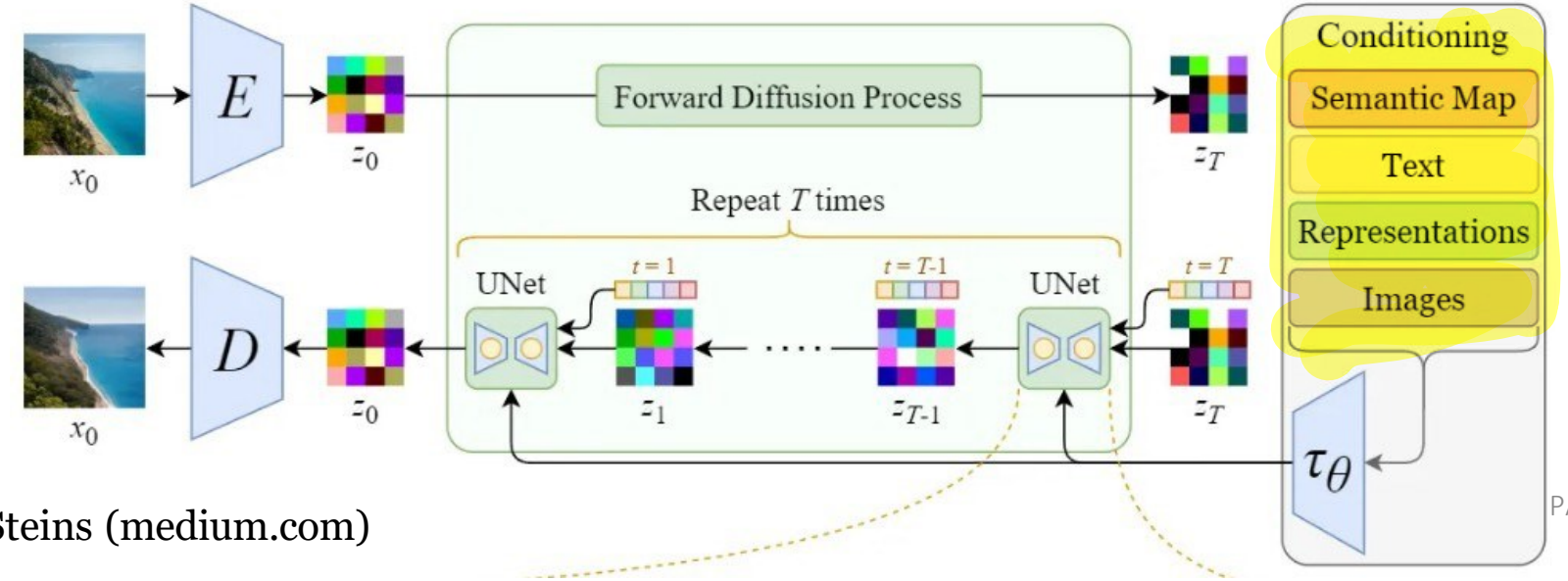
- The encoder-decoder E-D is a variational autoencoder
- E-D is trained independently, before the latent diffusion



From Steins (medium.com)

Conditioning

- The conditioning on text is obtained by giving in input to the Unet a encoding of the text
- The text encoder is CLIP
  - It has been designed to allign image and text encoding
  - It exploits a language model and an image encoder (only the former is used here)
- Other encoders can be used in the same way, for conditioning on image,semantic maps,….

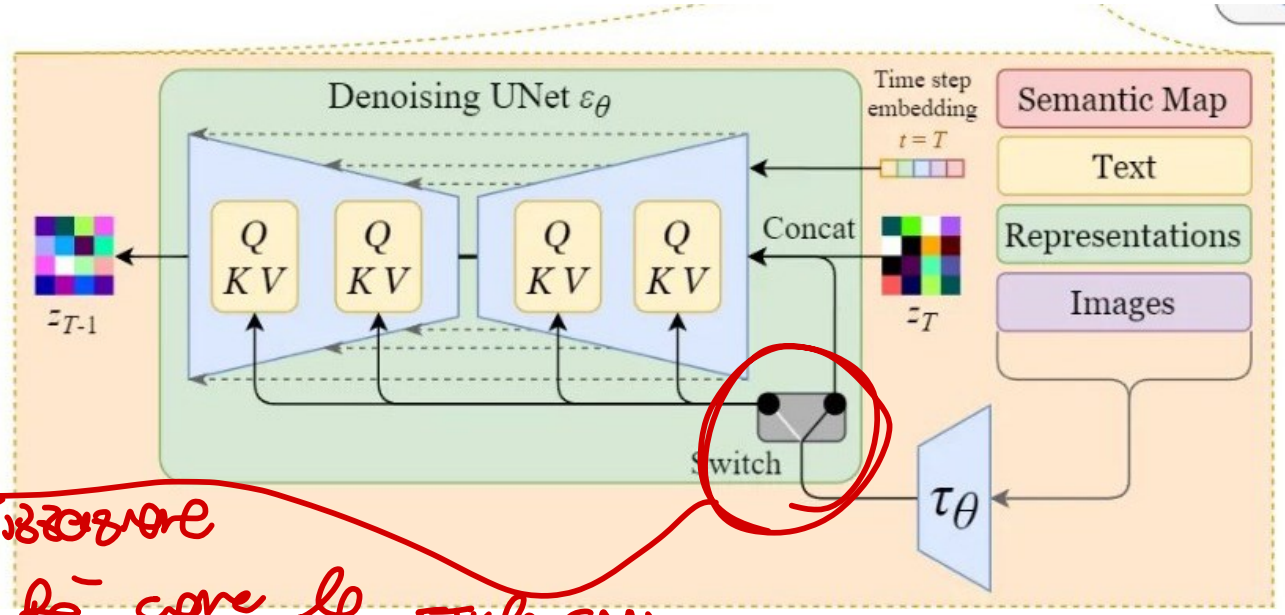


From Steins (medium.com)

Conditioning

- The conditioning can be added directly in input or to the inner layers of denosing Unet,
  - The former outperform the latter
  - In the former case, transformers are used to conditioning the feature maps

- The Unet is a recent version, including transformers and other tricks



From Steins (medium.com)

Trainining time (measured in days on VT100 GPU)

Method	Generator Compute	Classifier Compute	Overall Compute	Inference Throughput*	N_{params}	FID↓	IS↑	Precision↑	Recall↑
LSUN Churches 256^2									
StyleGAN2 [42]†	64	-	64	-	59M	3.86	-	-	-
LDM-8 (ours, 100 steps, 410K)	18	-	18	6.80	256M	4.02	-	0.64	0.52
LSUN Bedrooms 256^2									
ADM [15]† (1000 steps)	232	-	232	0.03	552M	1.9	-	0.66	0.51
LDM-4 (ours, 200 steps, 1.9M)	60	-	55	1.07	274M	2.95	-	0.66	0.48
CelebA-HQ 256^2									
LDM-4 (ours, 500 steps, 410K)	14.4	-	14.4	0.43	274M	5.11	-	0.72	0.49
FFHQ 256^2									
StyleGAN2 [42]	32.13‡	-	32.13†	-	59M	3.8	-	-	-
LDM-4 (ours, 200 steps, 635K)	26	-	26	1.07	274M	4.98	-	0.73	0.50
ImageNet 256^2									
VQGAN-f-4 (ours, first stage)	29	-	29	-	55M	0.58††	-	-	-
VQGAN-f-8 (ours, first stage)	66	-	66	-	68M	1.14††	-	-	-
BigGAN-deep [3]†	128-256		128-256	-	340M	6.95	$203.6_{\pm 2.6}$	0.87	0.28
ADM [15] (250 steps) †	916	-	916	0.12	554M	10.94	100.98	0.69	0.63
ADM-G [15] (25 steps) †	916	46	962	0.7	608M	5.58	-	0.81	0.49
ADM-G [15] (250 steps)†	916	46	962	0.07	608M	4.59	186.7	0.82	0.52
ADM-G,ADM-U [15] (250 steps)†	329	30	349	n/a	n/a	3.85	221.72	0.84	0.53
LDM-8-G (ours, 100, 2.9M)	79	12	91	1.93	506M	8.11	$190.4_{\pm 2.6}$	0.83	0.36
LDM-8 (ours, 200 ddim steps 2.9M, batch size 64)	79	-	79	1.9	395M	17.41	72.92	0.65	0.62
LDM-4 (ours, 250 ddim steps 178K, batch size 1200)	271	-	271	0.7	400M	10.56	$103.49_{\pm 1.24}$	0.71	0.62
LDM-4-G (ours, 250 ddim steps 178K, batch size 1200, classifier-free guidance [32] scale 1.25)	271	-	271	0.4	400M	3.95	$178.22_{\pm 2.43}$	0.81	0.55
LDM-4-G (ours, 250 ddim steps 178K, batch size 1200, classifier-free guidance [32] scale 1.5)	271	-	271	0.4	400M	3.60	$247.67_{\pm 5.59}$	0.87	0.48

Results

Rombach, Blattman et al., 2022

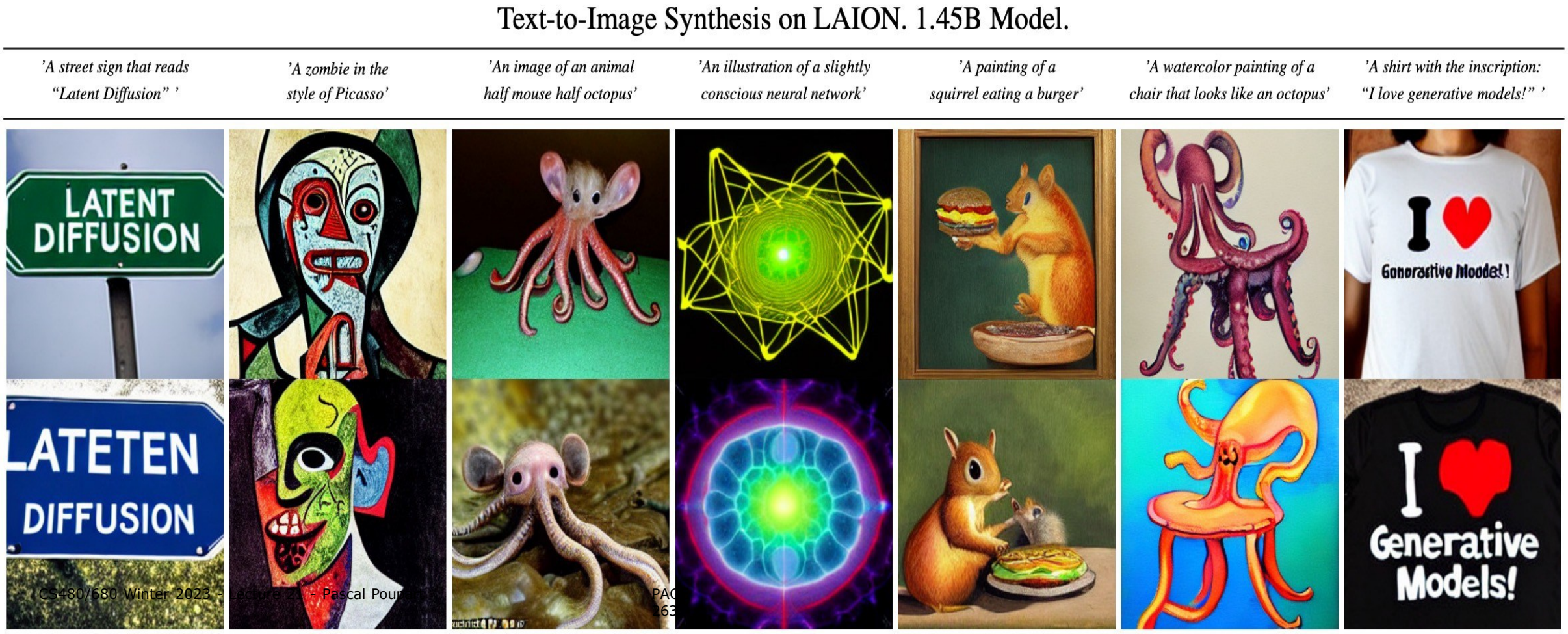
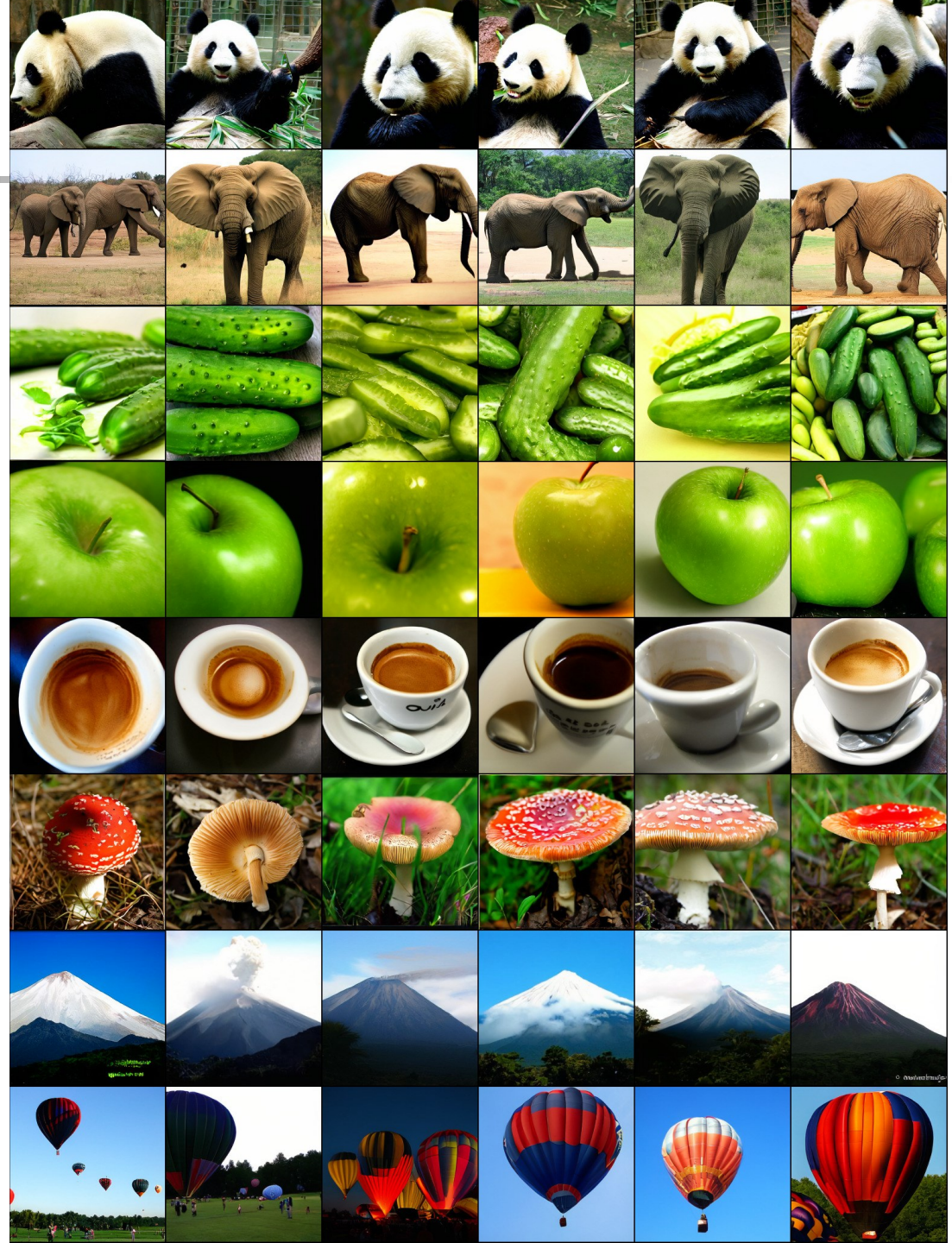


Figure 5. Samples for user-defined text prompts from our model for text-to-image synthesis, *LDM-8 (KL)*, which was trained on the LAION [78] database. Samples generated with 200 DDIM steps and $\eta = 1.0$. We use unconditional guidance [32] with $s = 10.0$.

Conditional class on ImageNet



Other applications, namely other conditioning: semantic generation

Inpainting

Inpainting