Diffusion Models

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA’ DI PISA

DAVIDE.BACCIU@UNIPI.IT

Lecture Outline

- Introduction
  - Motivations
  - Learning to generate by denoising
- Denoising diffusion models
  - Forward & Reverse process
  - Training diffusion models
  - Implementation
- Advanced & Applications
  - Conditional generation
  - Multimodal

Why Diffusion Models?

“Diffusion Models Beat GANs on Image Synthesis” Dhariwal & Nichol, OpenAI, 2021

Ramesh et al., 2022

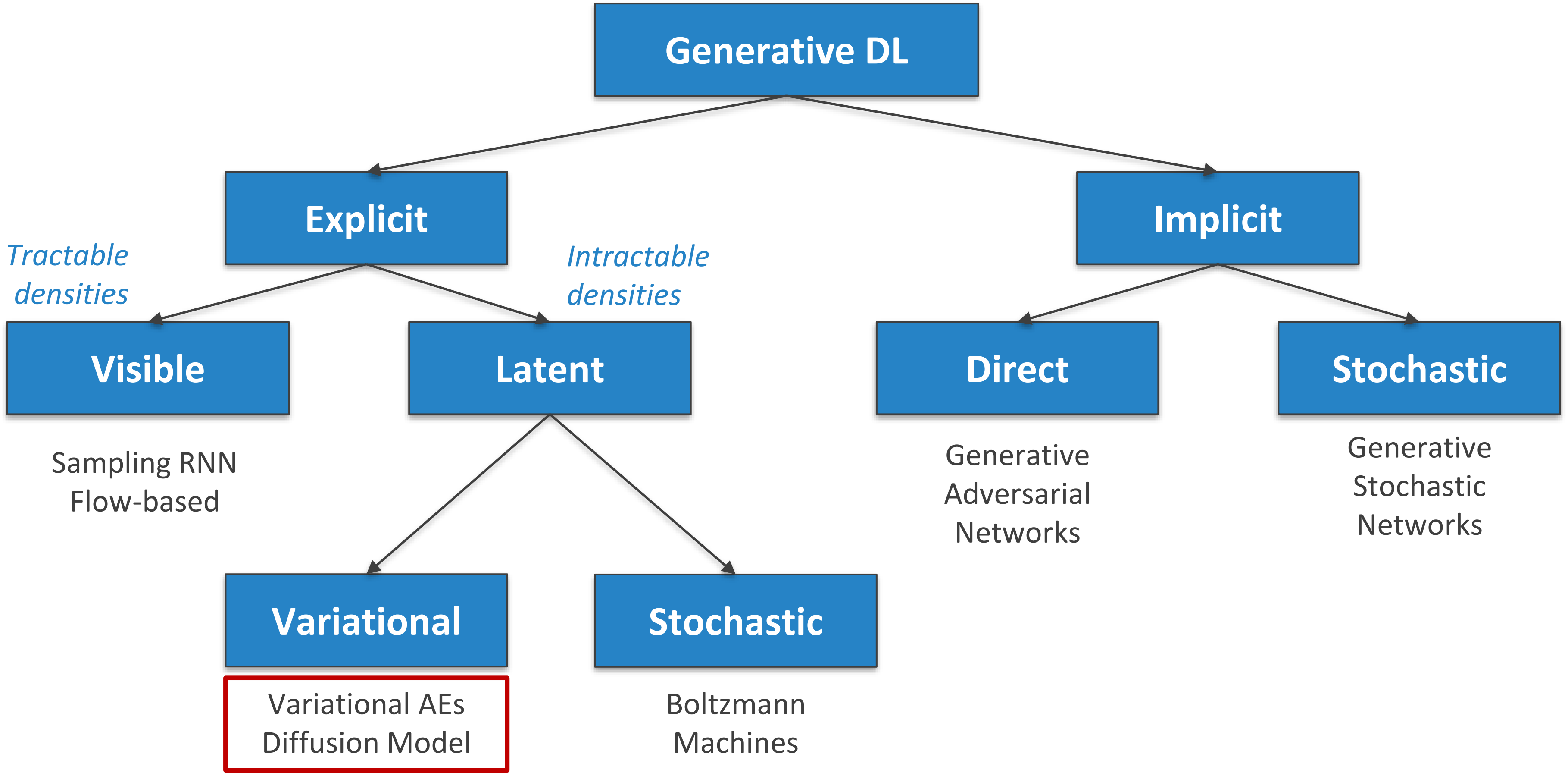
“a teddy bear on a skateboard in times square”

Why Diffusion Models?

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

A Taxonomy

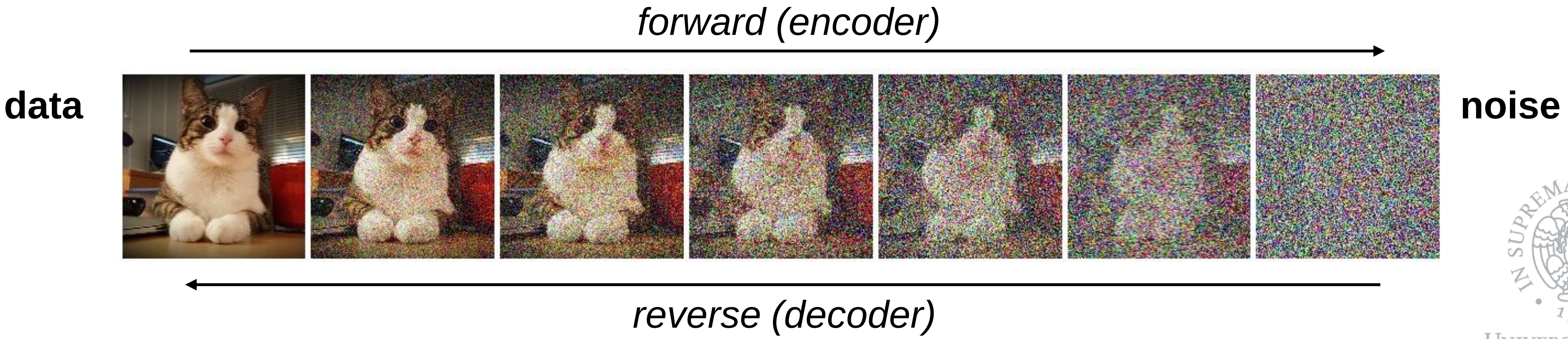
Diffusion models latent space has same size of data!



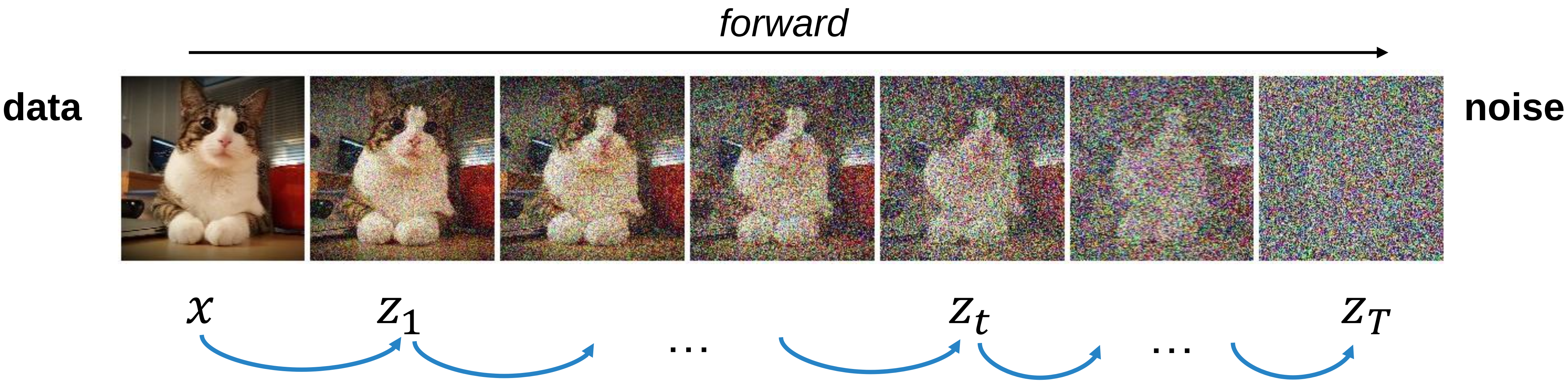
Adapted from I. Goodfellow, Tutorial on Generative Adversarial Networks, 2017

Learning to generate by denoising

- Two processes
  - Forward diffusion gradually adding noise to input
  - Reverse process reconstructs data from noise (generation)
- The key is how to do this efficiently

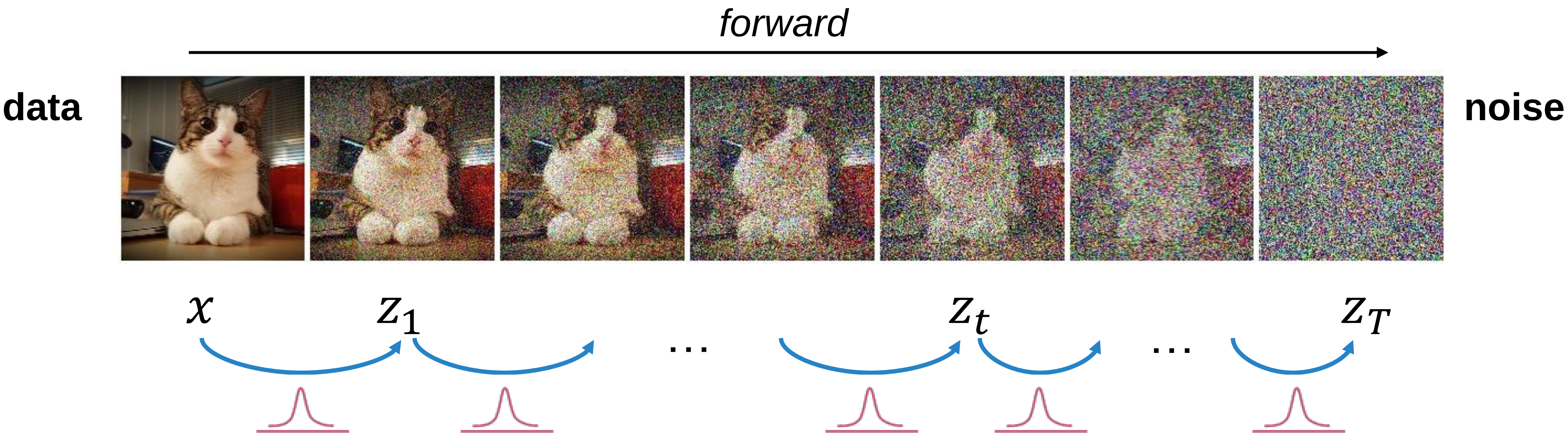


Forward Diffusion - Intuition



A fixed (i.e. non-adaptive) noise process in T steps mapping original data x into a same-sized latent variables z_t using simple additive noise

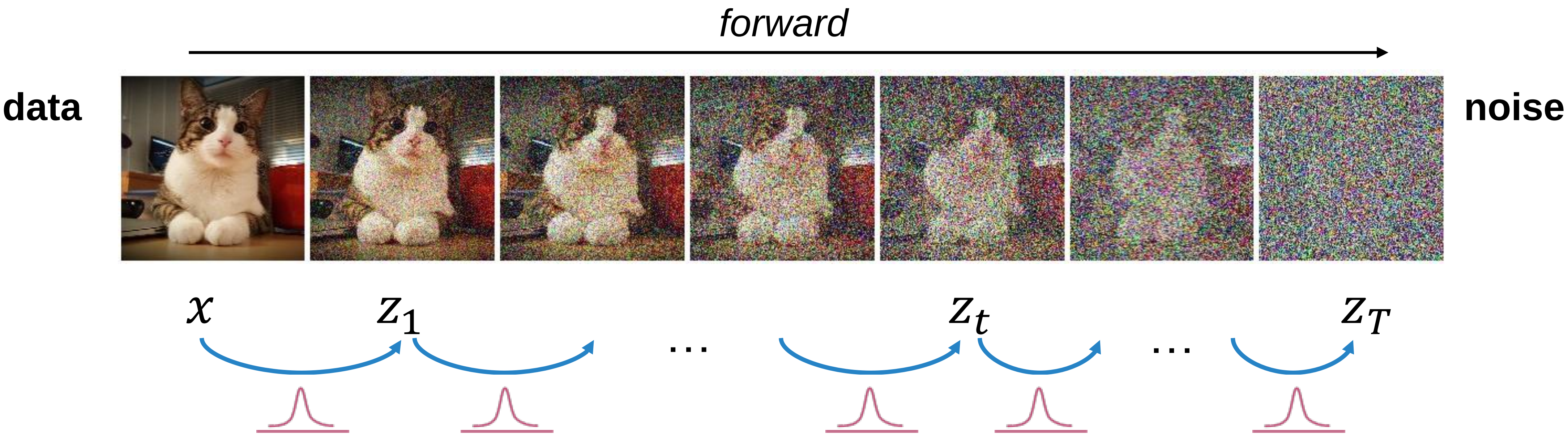
Forward Diffusion – Noise addition



$$\boldsymbol{z}_1 = \sqrt{1-\beta_1}\boldsymbol{x} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \qquad \boldsymbol{z}_t = \sqrt{1-\beta_t}\boldsymbol{z}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t$$

$\boldsymbol{\epsilon}_t \sim \mathcal{N}(0,1)$ $\quad$ $\beta_t \in [0,1]$ is the noise schedule

Forward Diffusion – Distributions



$$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}) \text{ where } \boldsymbol{z}_0 = \boldsymbol{x}$$

$$q(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T|\boldsymbol{x}) = q(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

Diffusion Kernel

Generating $\boldsymbol{z}_t$ sequentially is time-consuming so we use a closed-form solution for $q(\boldsymbol{z}_t|\cdot)$

$$q(\boldsymbol{z}_t|\boldsymbol{x}) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}, (1-\alpha_t)\boldsymbol{I}) \quad \text{(diffusion kernel)}$$

$$\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$$

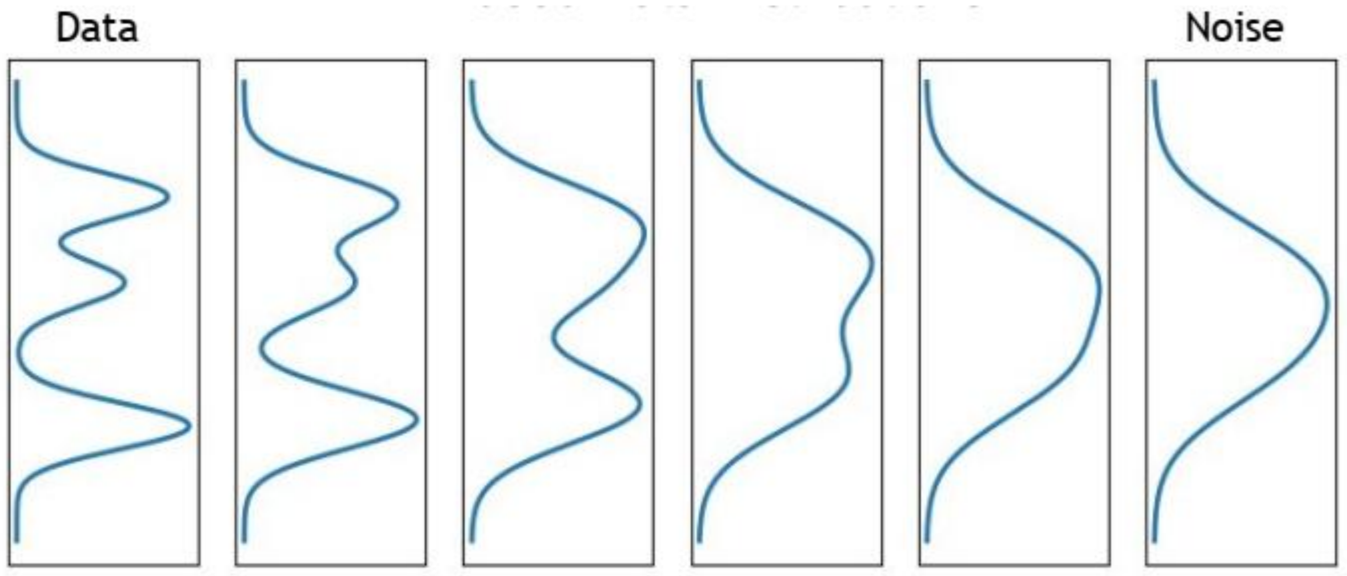
Which allows writing the marginal as

$$q(\boldsymbol{z}_t) = \int q(\boldsymbol{z}_t, \boldsymbol{x})d\boldsymbol{x} = \int \underbrace{q(\boldsymbol{x})}_{\text{data distribution}}q(\boldsymbol{z}_t|\boldsymbol{x})d\boldsymbol{x}$$

UNIVERSITÀ DI PISA

Evolution of diffused data distributions

$$q(\boldsymbol{z}_t) = \int q(\boldsymbol{z}_t, \boldsymbol{x})d\boldsymbol{x} = \int q(\boldsymbol{x})q(\boldsymbol{z}_t|\boldsymbol{x})d\boldsymbol{x}$$



$q(\boldsymbol{x})$ $q(\boldsymbol{z}_1)$ $q(\boldsymbol{z}_2)$ … $q(\boldsymbol{z}_T)$

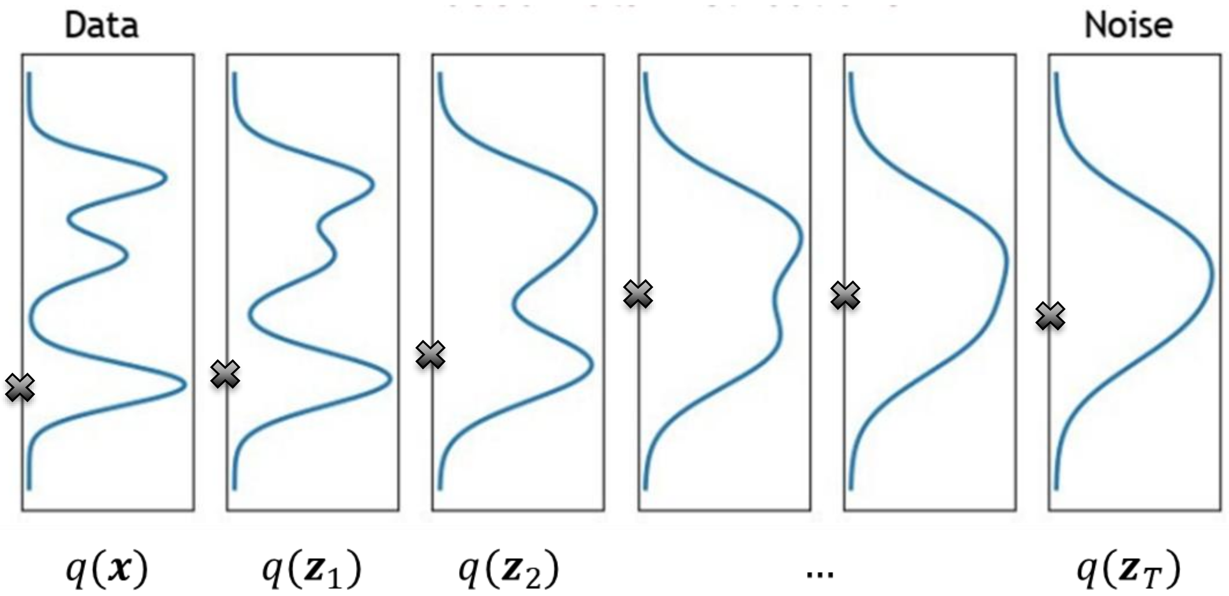
Denoising – Inverting the process

Sample $\boldsymbol{z}_T \sim \mathcal{N}(0,1)$

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

True denoising distribution is intractable

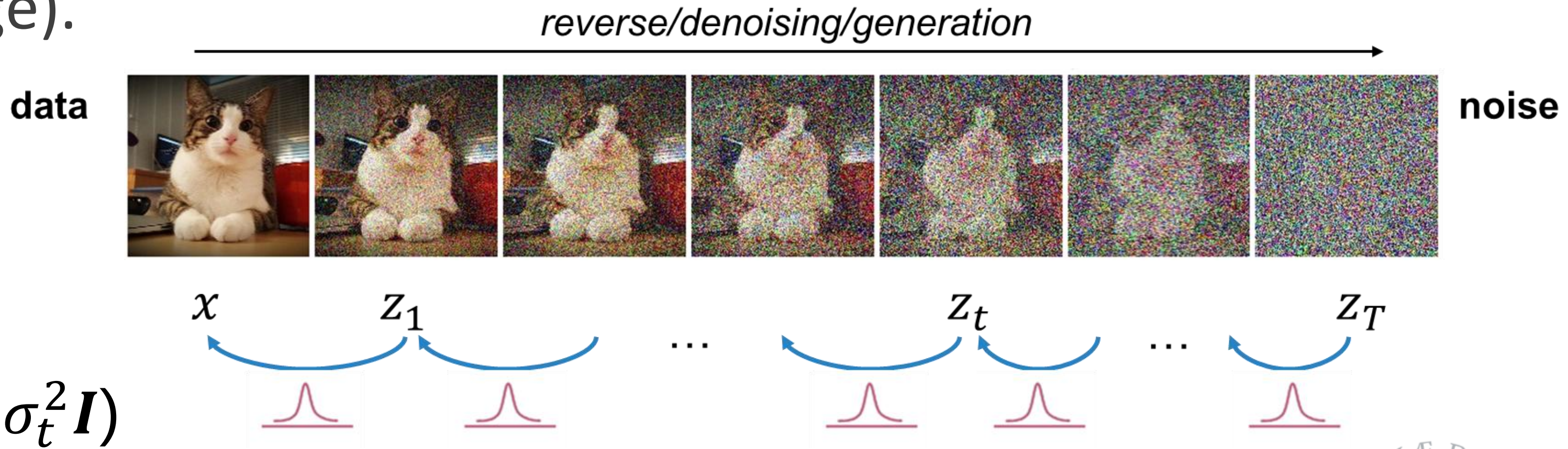
$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \frac{q(\boldsymbol{z}_{t-1})q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}{q(\boldsymbol{z}_t)}$$



We cannot de-mix noise if we don't know the starting point $\boldsymbol{x}$. If we do, then we can show that $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x})$ is Normal

Denoising – Reverse Process

- Reverse process learns an approximated denoising distribution (decoder)
- Assuming reverse distributions are approximately Normal (reasonable if β_t are small and T large).



$$P(\boldsymbol{z}_T) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$$P_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \mathcal{N}(\mu_\theta(\boldsymbol{z}_t, t), \sigma_t^2 \boldsymbol{I})$$

Mean of the denoised image $\boldsymbol{z}_{t-1}$ predicted by the θ-parameterized model given $\boldsymbol{z}_t$ (and time encoding)

Training

- Training follows the classical log-likelihood maximization view

$$\log \mathrm{P}_\theta(\boldsymbol{x}) = \log \int P_\theta(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T, \boldsymbol{x}) d\boldsymbol{z}_{1\dots T}$$

$$= \log \int P_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} P_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) P_\theta(\boldsymbol{z}_T)\, d\boldsymbol{z}_{1\dots T}$$

- …which is, of course, intractable

Training – ELBO at the rescue

Introduce the encoder distribution q (with $\bar{\boldsymbol{z}} = \boldsymbol{z}_1, \dots, \boldsymbol{z}_T$)

$$(\text{loglik}) \; \log \int P_\theta(\bar{\boldsymbol{z}}, \boldsymbol{x}) d\bar{\boldsymbol{z}} \geq \int q(\bar{\boldsymbol{z}}|\boldsymbol{x}) \log \left[\frac{P_\theta(\bar{\boldsymbol{z}}, \boldsymbol{x})}{q(\bar{\boldsymbol{z}}|\boldsymbol{x})} \right] d\bar{\boldsymbol{z}} \;\; (ELBO)$$

Sparing some derivation and simplifications, we approximate ELBO as:

$$\underbrace{\mathbb{E}_{q(\boldsymbol{z}_1|x)}[\log P_\theta(\boldsymbol{x}|\boldsymbol{z}_1)]}_{\text{Reconstruction term}} - \sum_{t=2}^{T} \underbrace{KL\big(q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x}) || P_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)\big)}_{\text{Aligns predicted and original (input-conditional) denoising densities}}$$

ELBO Loss Function

Distributions in KL are all Gaussians so can write the full form of the loss

$$\sum_{\boldsymbol{x}}\left(\underbrace{\left(-\log \mathcal{N}\left(\mu_\theta(\boldsymbol{z}_1,t),\sigma_1^2 \boldsymbol{I}\right)\right)}_{\text{Reconstruction}} + \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\underbrace{\left(\frac{(1-\alpha_{t-1})}{(1-\alpha_t)}\sqrt{1-\beta_t}\,\boldsymbol{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{(1-\alpha_t)}\boldsymbol{x}\right)}_{\text{Target mean of } q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})} - \underbrace{\mu_\theta(\boldsymbol{z}_t,t)}_{\text{predicted } \boldsymbol{z}_{t-1}}\right\|^2\right)$$

Minimize difference between estimate of $\boldsymbol{z}_{t-1}$ and the most likely value from ground truth-denoised data

Training – Practical view

Loss can be heavily simplified by reparameterizing so that the model predicts the noise $\epsilon_\theta(\cdot)$ that was mixed with the original data, rewriting $\boldsymbol{x}$ as

$$\boldsymbol{x} = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}_t$$

A network which predicts the unit noise given current noised input $\boldsymbol{z}_t$

Inserting $\boldsymbol{x}$ above into the ELBO yields (after a while)

$$Loss(\theta) = \sum_{\boldsymbol{x}}\sum_{t=1}^{T}\left\|\epsilon_\theta\big(\underbrace{\sqrt{\alpha_t}\boldsymbol{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t}_{\boldsymbol{z}_t}, t\big) - \boldsymbol{\epsilon}_t\right\|^2$$

J. Ho et al, NeurIPS 2020

Implementation - Training

Algorithm 18.1: Diffusion model training

Input: Training data $\mathbf{x}$
Output: Model parameters θ
repeat
  for $i \in \mathcal{B}$ **do** // For every training example index in batch
    $t \sim \text{Uniform}[1, \ldots T]$ // Sample random timestep
    $\boldsymbol{\epsilon} \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$ // Sample noise
    $\ell_i = \left\| \epsilon_\theta\left(\sqrt{\alpha_t}\boldsymbol{x_i} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, t\right) - \boldsymbol{\epsilon} \right\|^2$ // Compute individual loss
  Accumulate losses for batch and take gradient step
until converged

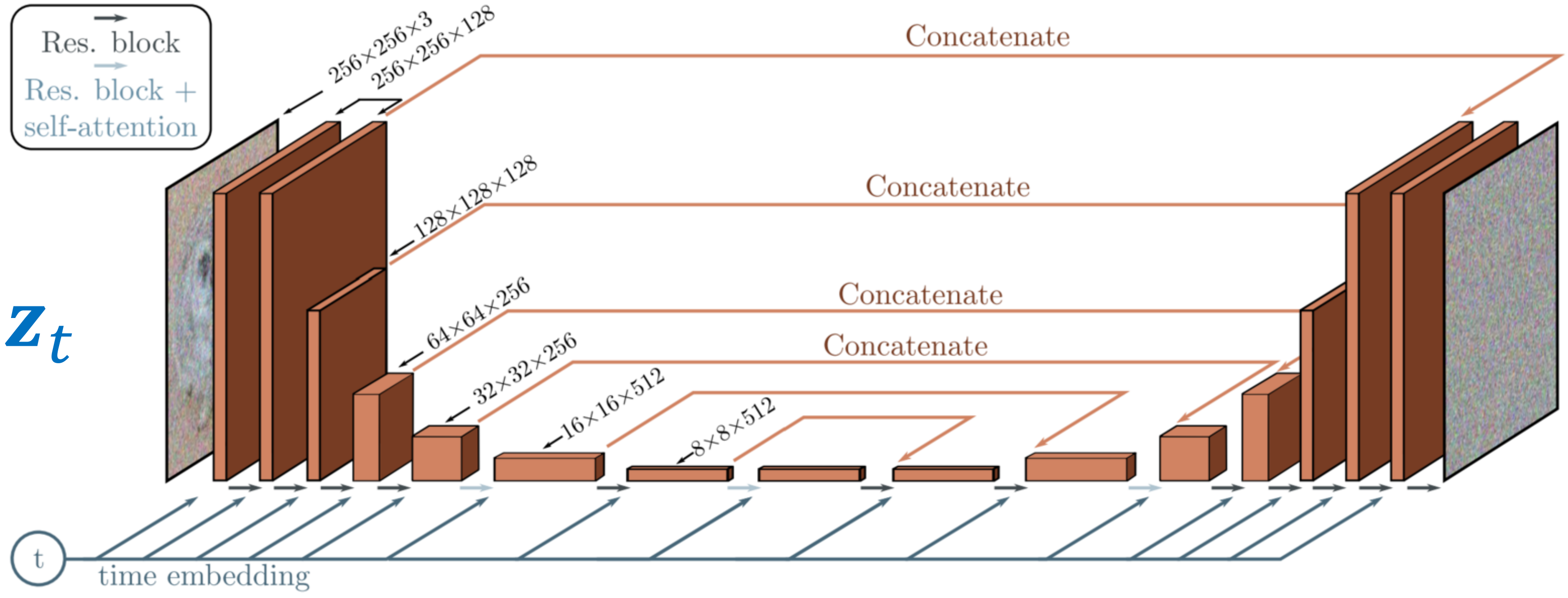
During forward we add noise to image. During reverse we predict that noise with a DNN and then subtract it from the image to denoise it.

Diffusion model for images

J. Ho et al, NeurIPS 2020
Dharivwal and Nichol NeurIPS 2021

U-Net architectures with ResNet blocks and self-attention layers

$\boldsymbol{z}_t$



$\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$

Sinusoidal positional embeddings

Time features are fed to residual blocks

Noise Schedules

- Terms β_t and σ_t control variance of forward diffusion and reverse denoising, respectively
  - β_t linear schedule
  - $\sigma_t^2 = \beta_t$
- Slowly increase the amount of added noise (as high-resolution information is corrupted first)
- Alternatives
  - σ_t can be learned by minimizing the bound
  - β_t can be learned by minimizing the variance of the training objective

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

$$P_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \mathcal{N}(\mu_\theta(\boldsymbol{z}_t, t), \sigma_t^{\boldsymbol{2}} \boldsymbol{I})$$

Guided Generation – Classifier Guidance

Guide diffusion process using auxiliary data c, using the gradient of a trained classifier as guidance

1. Train the diffusion model unconditionally
2. Train a classifier $P(c|\boldsymbol{z}_t)$ where c are conditioning labels
3. Add an extra term when sampling the diffusion model, i.e. when reconstructing $\boldsymbol{z}_{t-1}$ from $\boldsymbol{z}_t$, that modifies the reconstruction in the direction given by the gradient of a classifier

$$\boldsymbol{z}_{t-1} = \underbrace{\hat{\boldsymbol{z}}_{t-1} + \sigma_t \boldsymbol{\epsilon}}_{\text{Reversed diffusion}} + \underbrace{\sigma_t^2 \frac{\partial P(c|\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t}}_{\text{Classifier guidance}}$$

Classifier Guidance - Issues

- Classifier guidance comes from mixing the predicted score function of the unconditional diffusion model with the classifier gradients

$$\frac{\partial \log P_\gamma(\boldsymbol{z}_t|c)}{\partial \boldsymbol{z}_t} = \frac{\partial \log P(\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t} + \gamma \frac{\partial \log P(c|\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t}$$

  - γ guidance scale

- Classifier receives a noisy input $\boldsymbol{z}_t$ at each step (can't use pretrained ones)

- Most of $\boldsymbol{z}_t$ is of no use for predicting c => arbitrary classifier gradients

UNIVERSITÀ DI PISA

Classifier-free Guidance

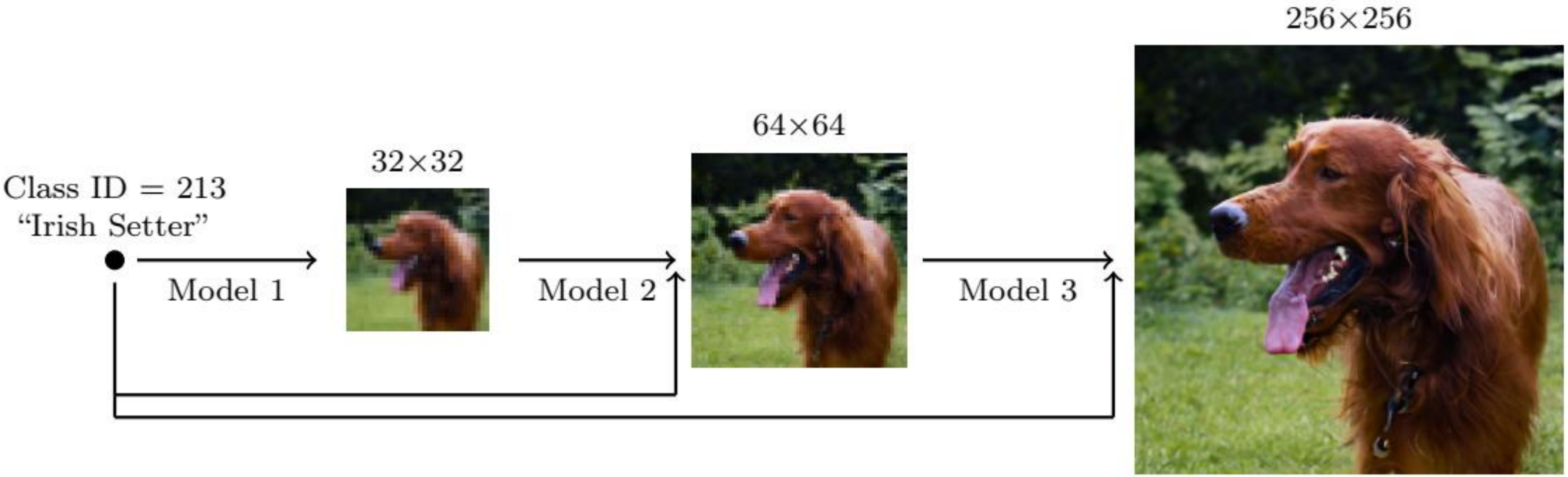
J. Ho, T. Salimans, NeurIPS 2021
Workshop DGMs Applications

Derive guidance from Bayes rule

$$\frac{\partial \log P_\gamma(\boldsymbol{z}_t|c)}{\partial \boldsymbol{z}_t} = (1-\gamma)\underbrace{\frac{\partial \log P(\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t}}_{\text{Unconditional diffusion score}} + \gamma \underbrace{\frac{\partial \log P(\boldsymbol{z}_t|c)}{\partial \boldsymbol{z}_t}}_{\text{Conditional diffusion score}}$$

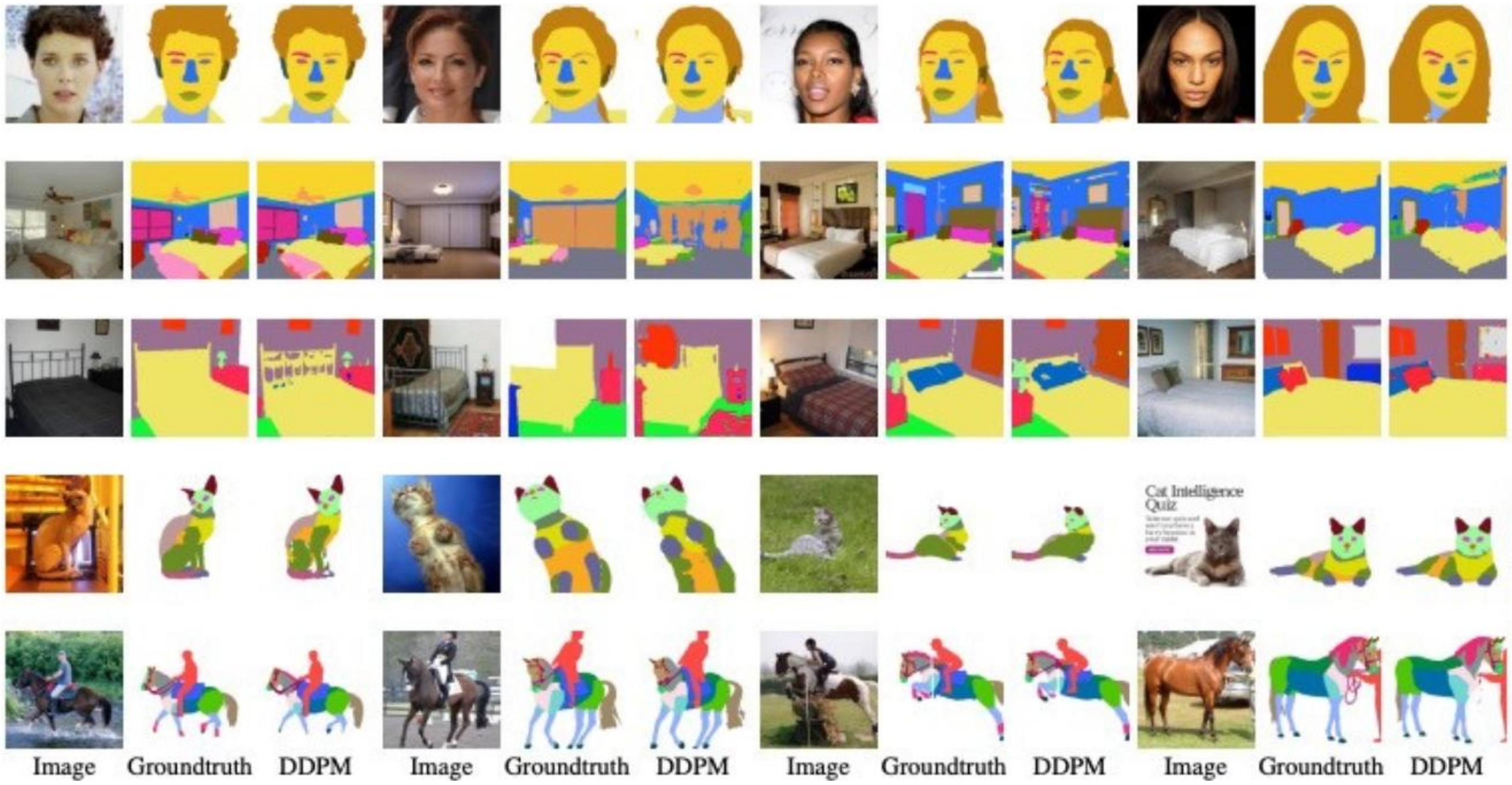
- Training conditional diffusion with dropout (randomly removing conditioning)
- Conditioning replaced by flag input (presence/absence of conditioning) => single model for conditional/unconditional diffusion

High-Resolution Image Generation



J. Ho et al, JMLR 2022

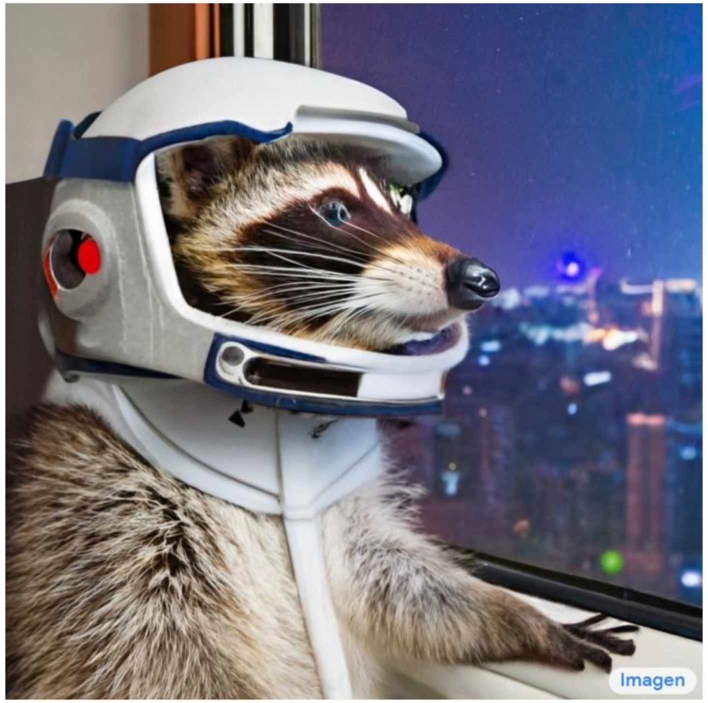
Diffusion-based Semantic Segmentation



Baranchuk et al, ICLR 2022

Conditional Generation

"A photo of a raccoon wearing an astronaut helmet, looking out of the window at night" (IMAGEN)



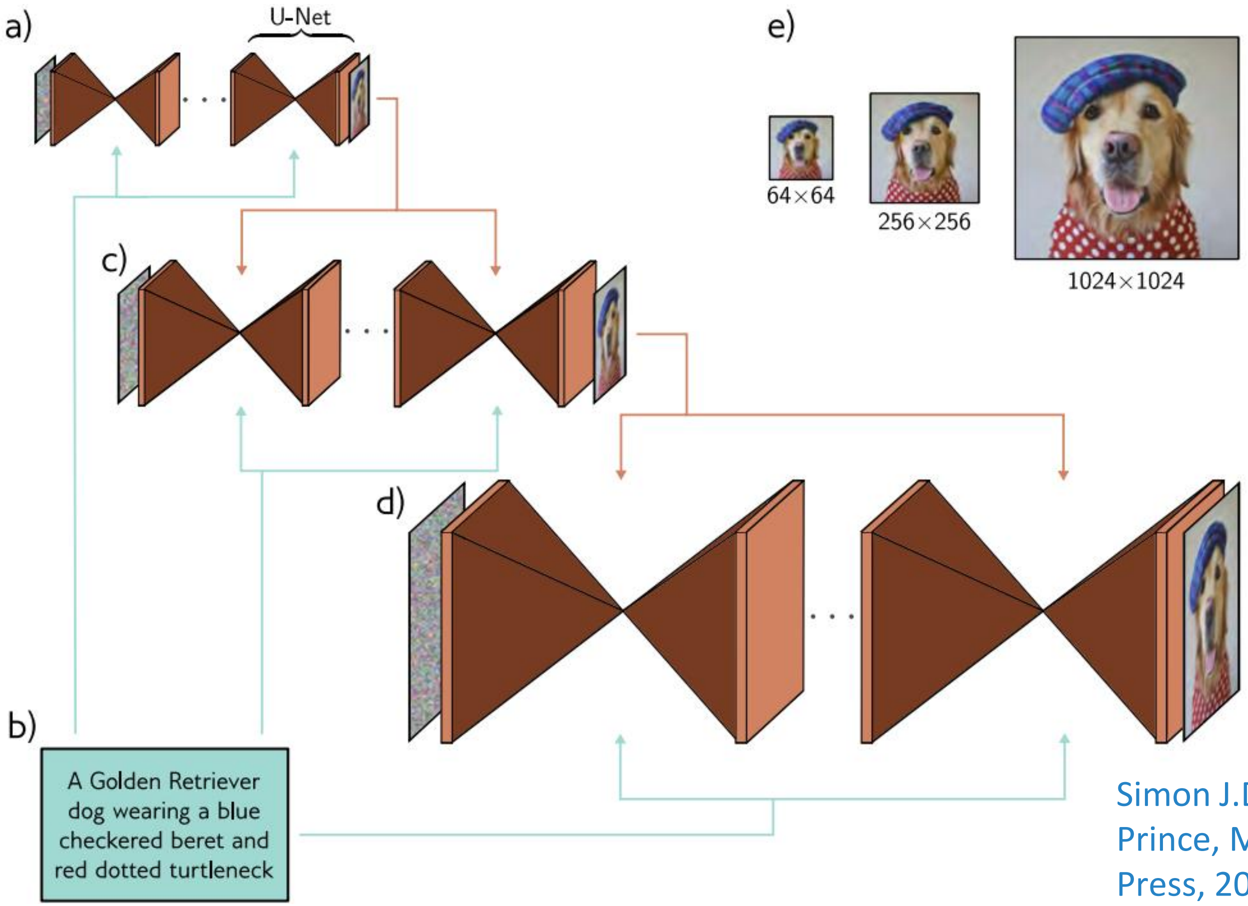
Text-2-image

Generated Input Generated

Panorama completion

UNIVERSITÀ DI PISA

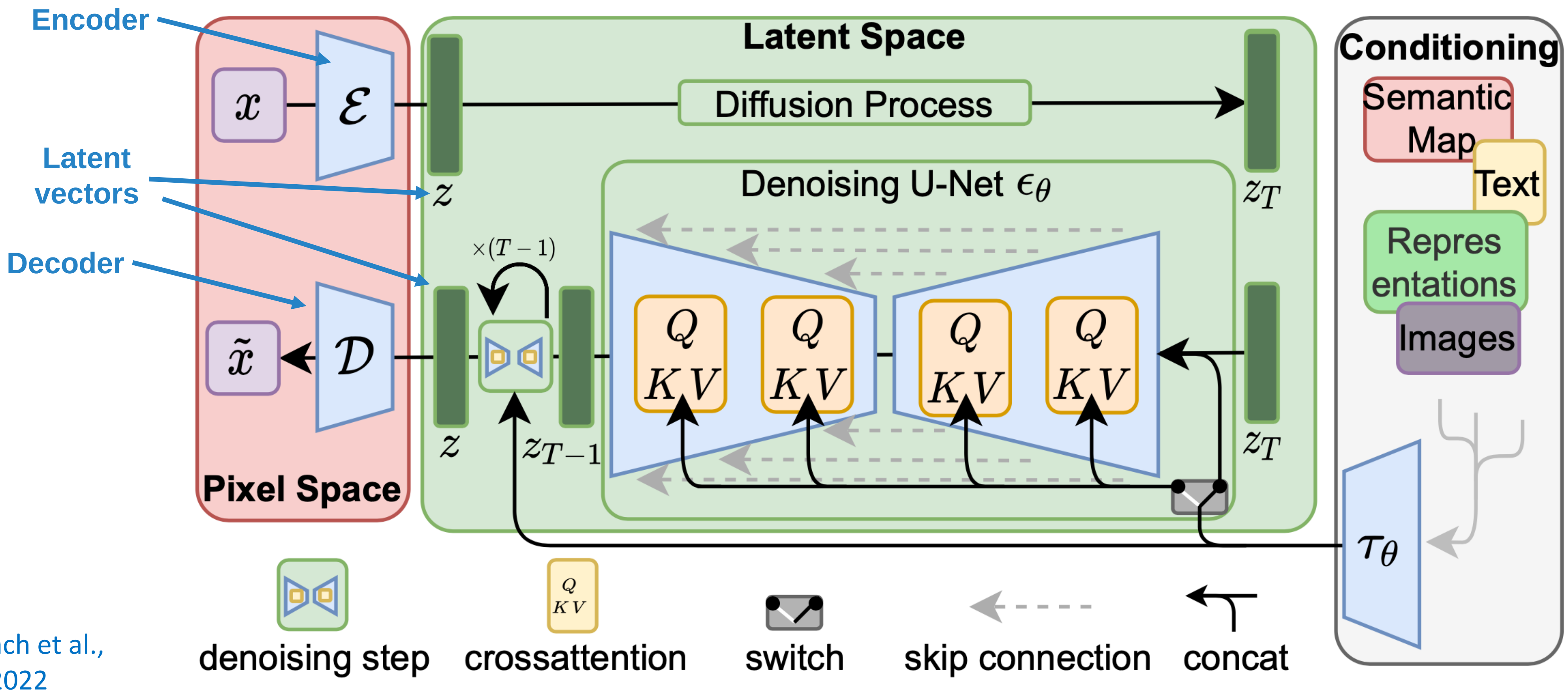
Cascaded Conditional Generation



Simon J.D. Prince, MIT Press, 2023

- Scalar – vector embedding + spatial addition (or adaptive group normalization)
- Image - channel-wise concatenation of the conditional image
- Text - vector embedding + spatial addition or cross-attention

Latent Space Diffusion

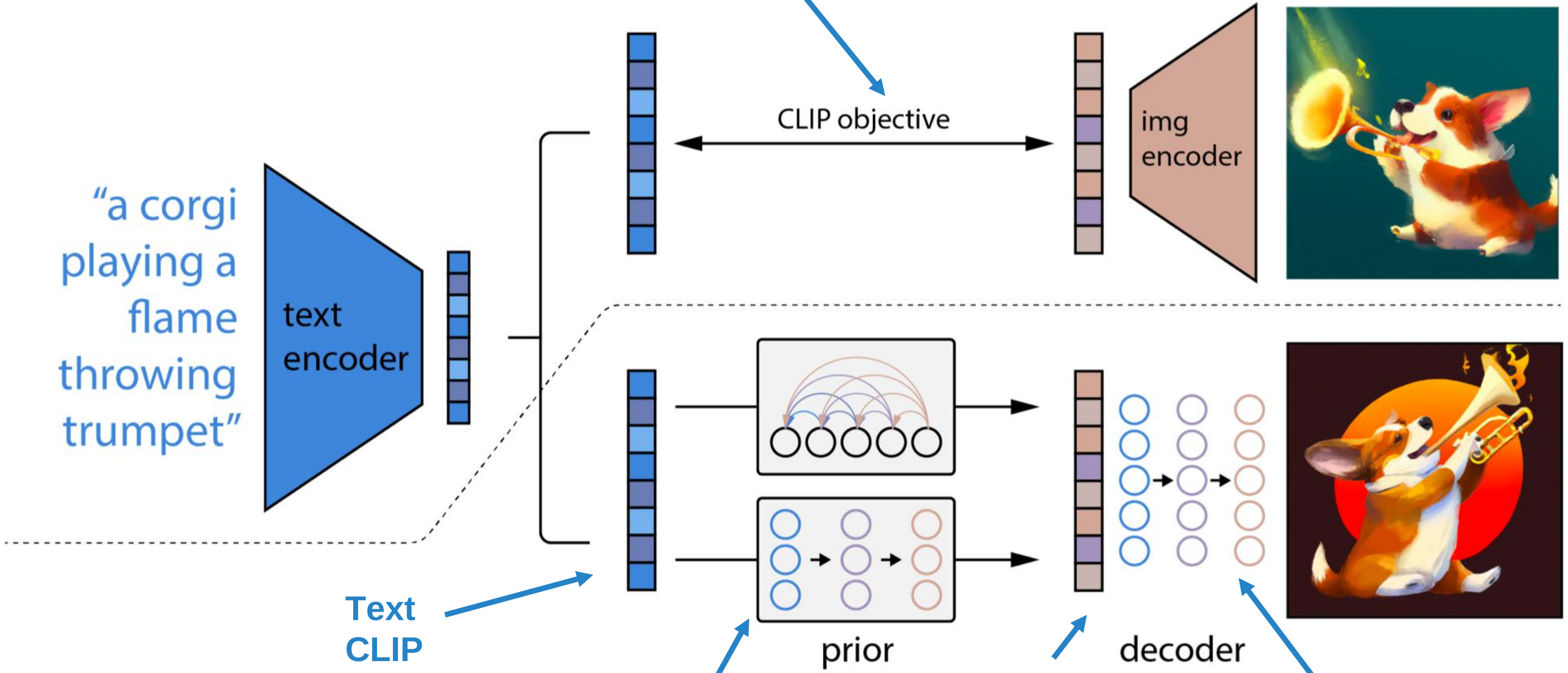


Run diffusion in the latent space instead of pixel space for cost saving

Rombach et al., CVPR 2022

DALL•E 2 – Diffusion Model

CLIP => scalable joint txt-img representations



joint representation space learning

text-to-image generation learning

Text CLIP

Autoregressive or diffusion prior

Image CLIP

Diffusion image decoding

Aditya Ramesh et al, OpenAI 2022

Take Home Messages

- Generate data from noise through a learned incremental denoising with fixed steps
  - Diffusion process can be reversed if the variance of the gaussian noise added at each step is small enough
  - Training goal is to make sure that the predicted noise map at each step is unit gaussian
  - During generation, subtract the predicted noise from the noisy image at time t to generate the image at time t-1
- Diffusion can be computationally involved
  - Need to take many small steps
  - Vanilla diffusion on a latent space same size as the original data
- Guided generation can improve sample quality (and reduce diversity)
- Latent space diffusion
  - Improves efficiency of generation
  - Generalizes which data that can be used (including discrete objects)
  - Allows introducing semantic structuring in latent space

Next Lecture(s)

Final lecture on deep generative models: normalizing flow

- Change of variable
- Coupling flows
- Masking & squeezing
- Invertible convolutions
- Autoregressive flows
- Normalizing flows and deep generative models wrap-up