Deep learning

Unpacking Transformers, LLMs and image generation

Session 6

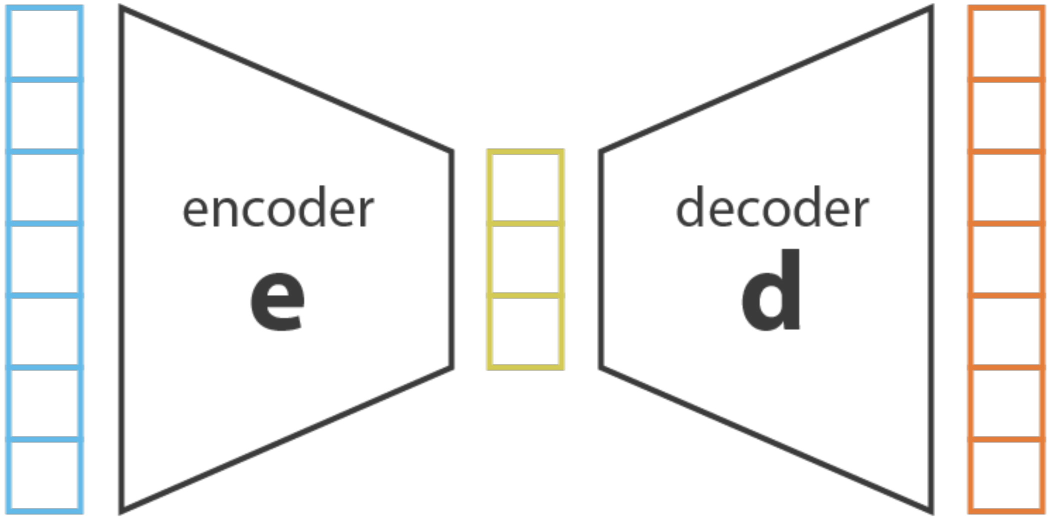
Syllabus

Session	Lecture	TP
1	Intro to DL Gradient descent and backprop	Intro to micrograd (*) fit the x^2 function
2	DL fundamentals I • Backprop • Loss functions • Neural Probabilistic Language Model (Bengio 2003)	Bigram model and MLP for next-character prediction (*) extend to tri-gram
3	DL fundamentals II • Activation function • Regularization • Initialization • Residual networks • Normalization Recurrent Neural Networks	Backprop ninja MLP in pytorch (*) add batchnorm to TP2
4	Attention and Transformers	GPT from scratch
5	DL for computer vision: convnets, unets	Convnets for CIFAR-10
6	VAE & Diffusion models	Diffusion from scratch **Quiz**

Agenda for today

1. VAEs
2. Diffusion

Auto-encoders



x | $e(x)$ | $d(e(x))$

initial data
in space R^n

encoded data
in latent space R^m (with $m<n$)

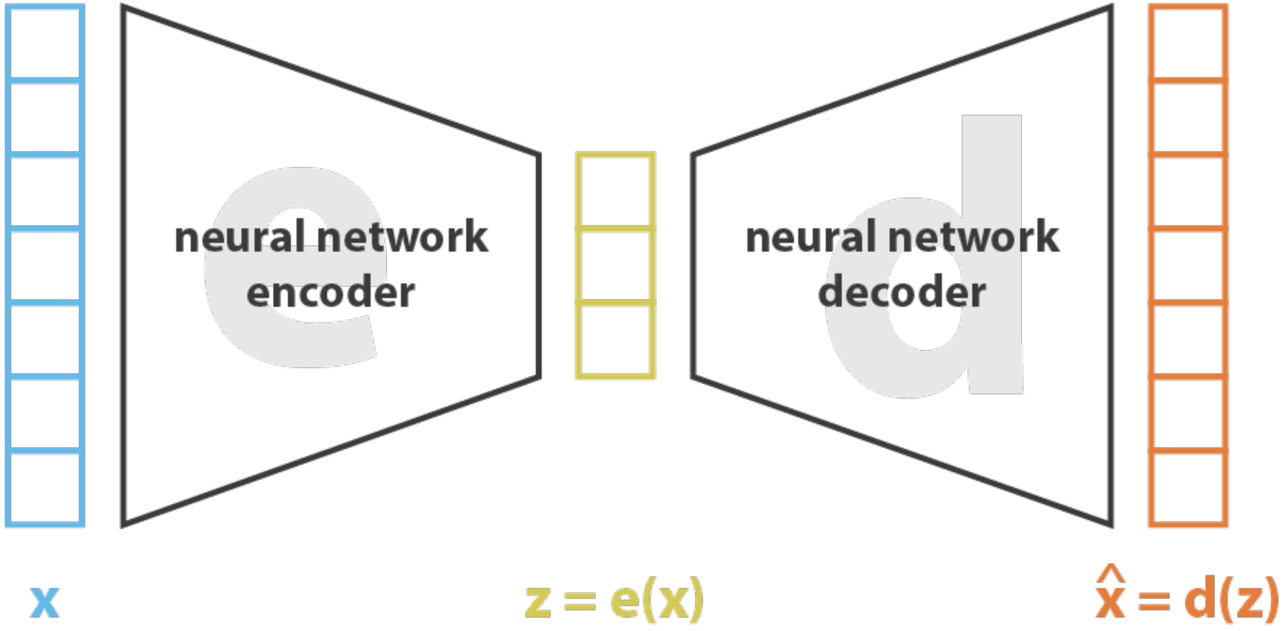
encoded-decoded data
back in the initial space R^n

$x = d(e(x))$ ➡ **lossless encoding**
no information is lost when reducing the number of dimensions

$x \neq d(e(x))$ ➡ **lossy encoding**
some information is lost when reducing the number of dimensions and can't be recovered later

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

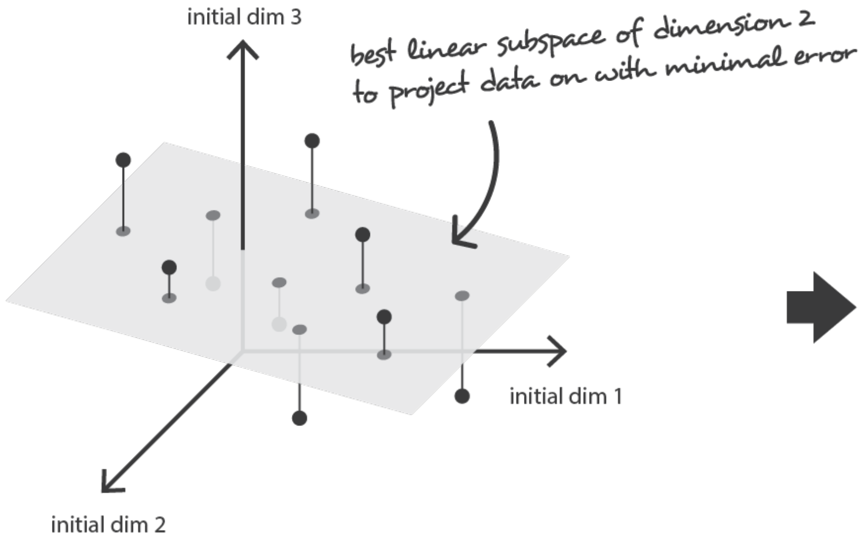
Auto-encoders



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

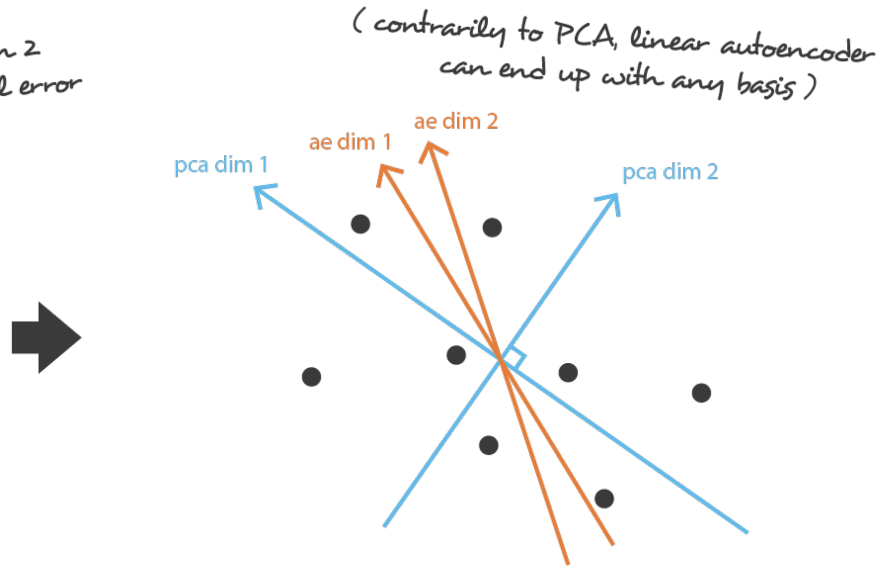
Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

Auto-encoders



Data in the full initial space

In order to reduce dimensionality, PCA and linear autoencoder target, in theory, the same optimal subspace to project data on...

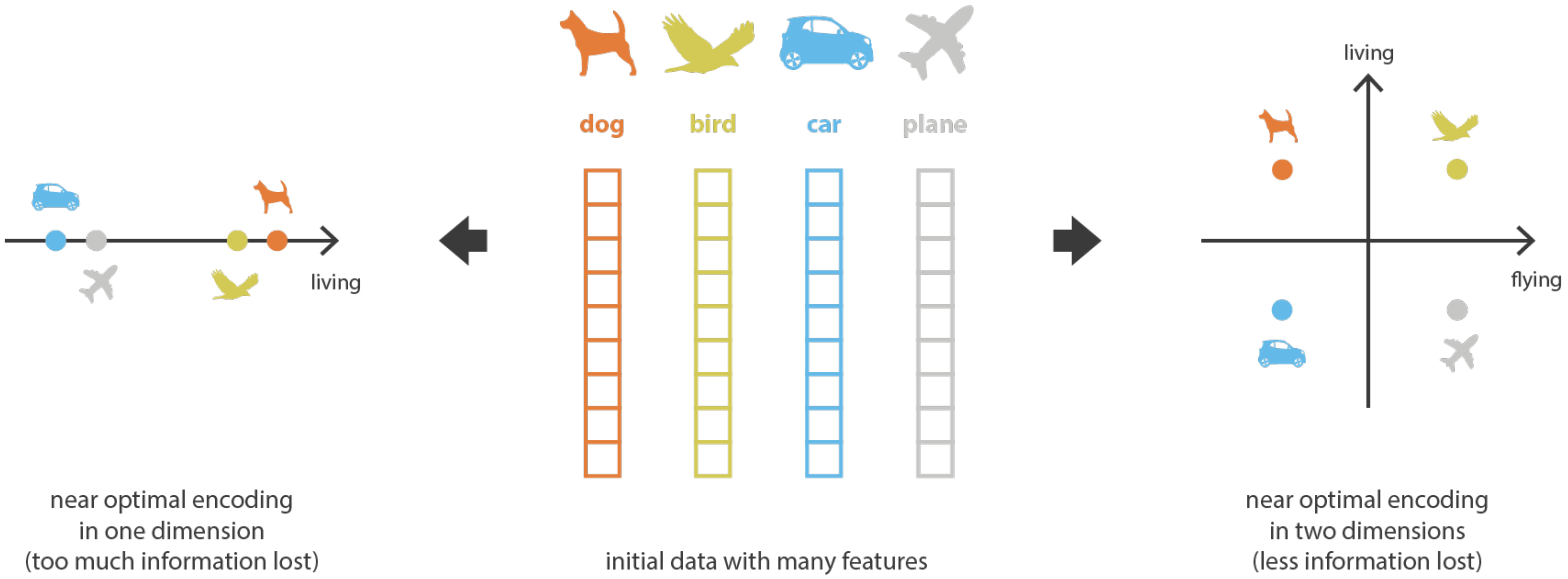


Data projected on the best linear subspace

... but not necessarily with the same basis due to different constraints (in PCA the first component is the one that explains the maximum of variance and components are orthogonal)

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

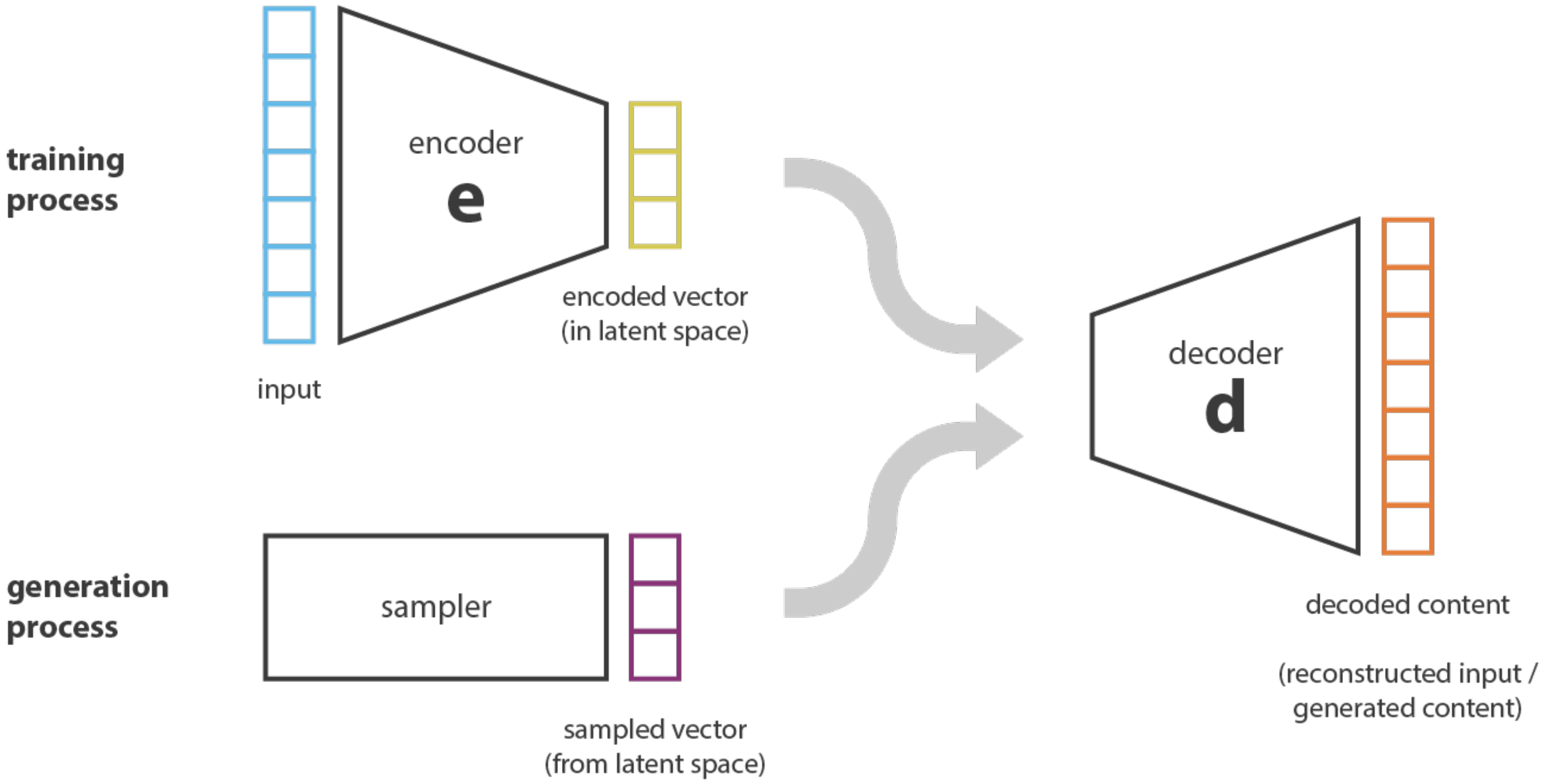
Auto-encoders



When reducing dimensionality, we want to keep the main structure there exists among the data.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

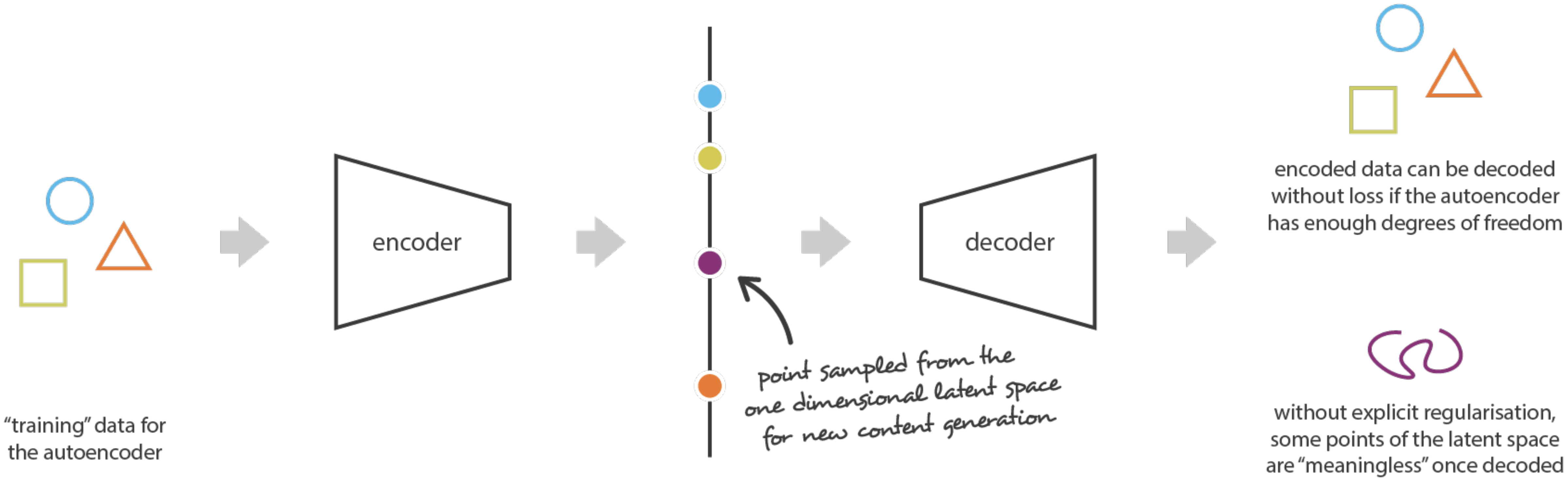
Auto-encoders



In theory, auto-encoders can be used for data generation.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

Auto-encoders



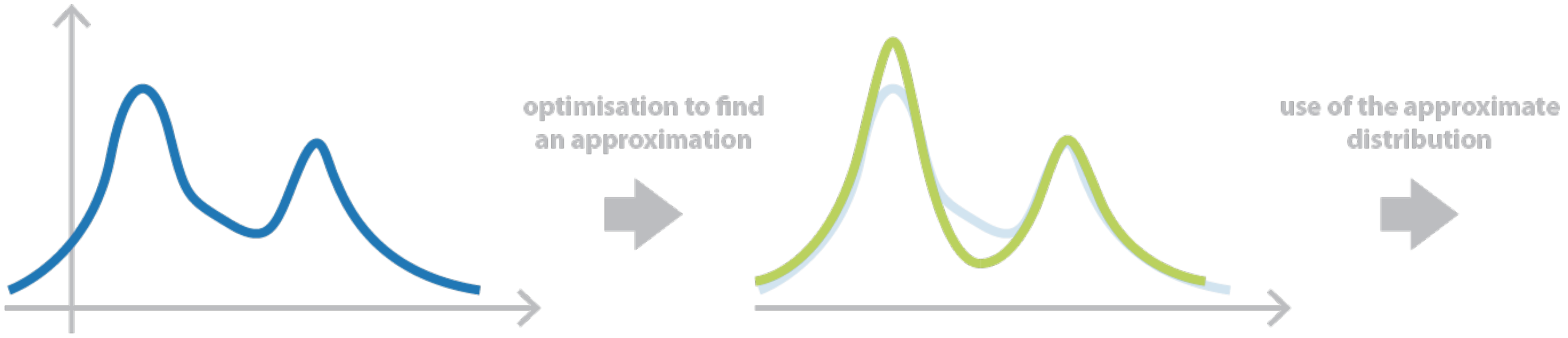
Irregular latent space prevent us from using autoencoder for new content generation.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

Variational auto-encoders

A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.

Variational inference



$$\int x.\pi(x)\, dx \approx \int x.f(x)\, dx$$

Unnormalised distribution (π)
whose normalisation factor
computation is intractable

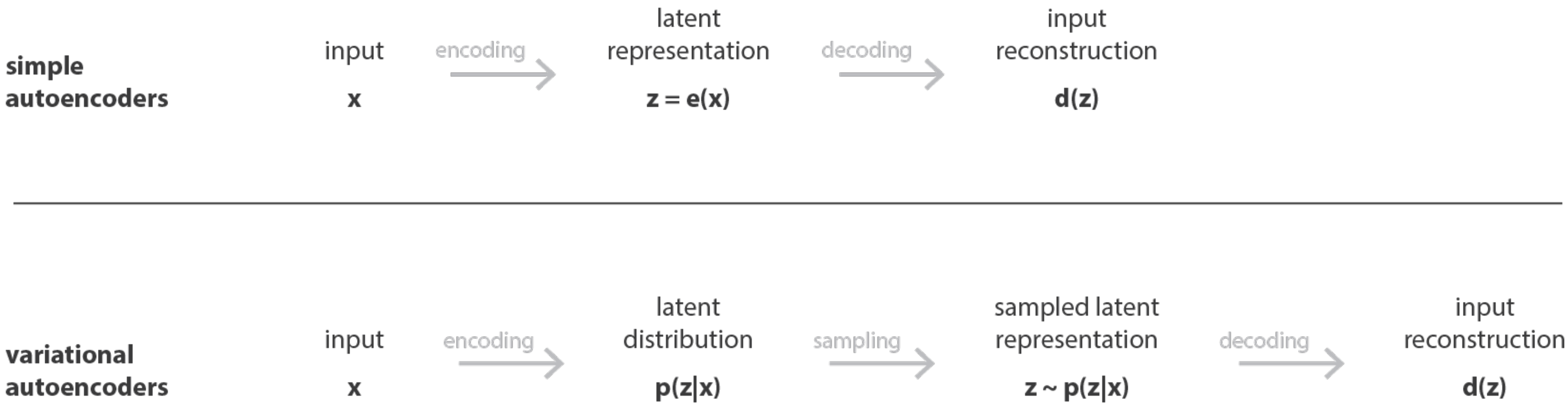
Best approximation (f)
among a parametrised family, obtained
by optimising over the family parameters
and without proceeding to the normalisation

Approximate computations
obtained by replacing the exact
distribution by the approximate one

Variational Inference methods that consist in finding the best approximation of a distribution among a parametrised family.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

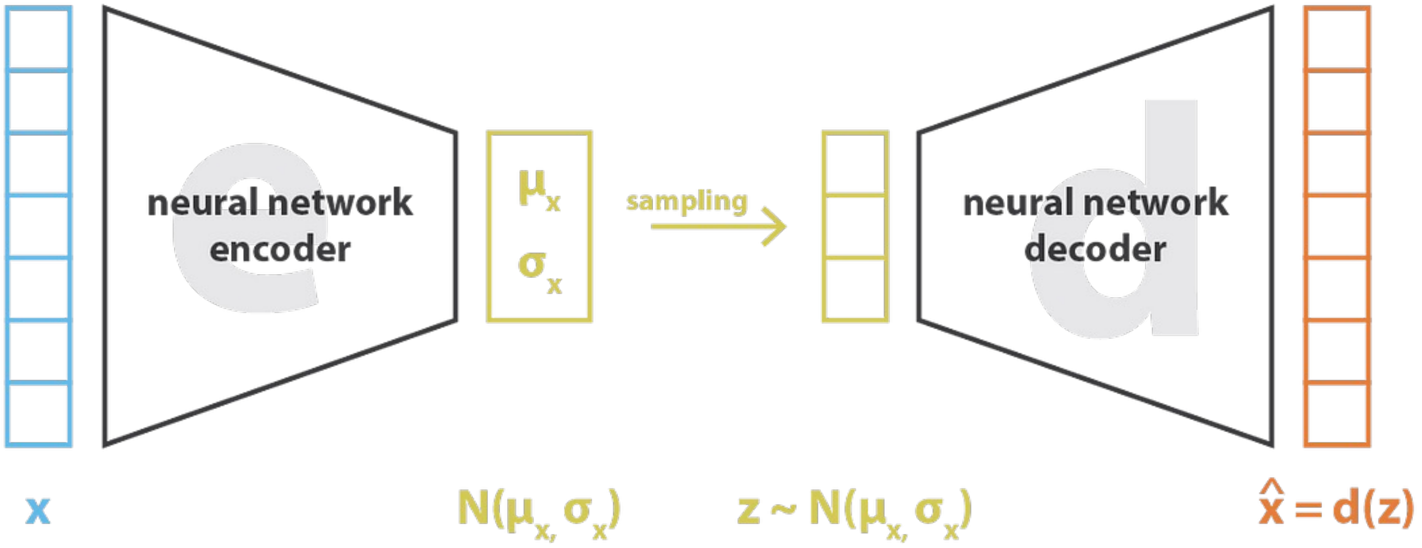
Variational Auto-encoders (VAEs)



Difference between autoencoder (deterministic) and variational autoencoder (probabilistic).

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

Variational Auto-encoders (VAEs)

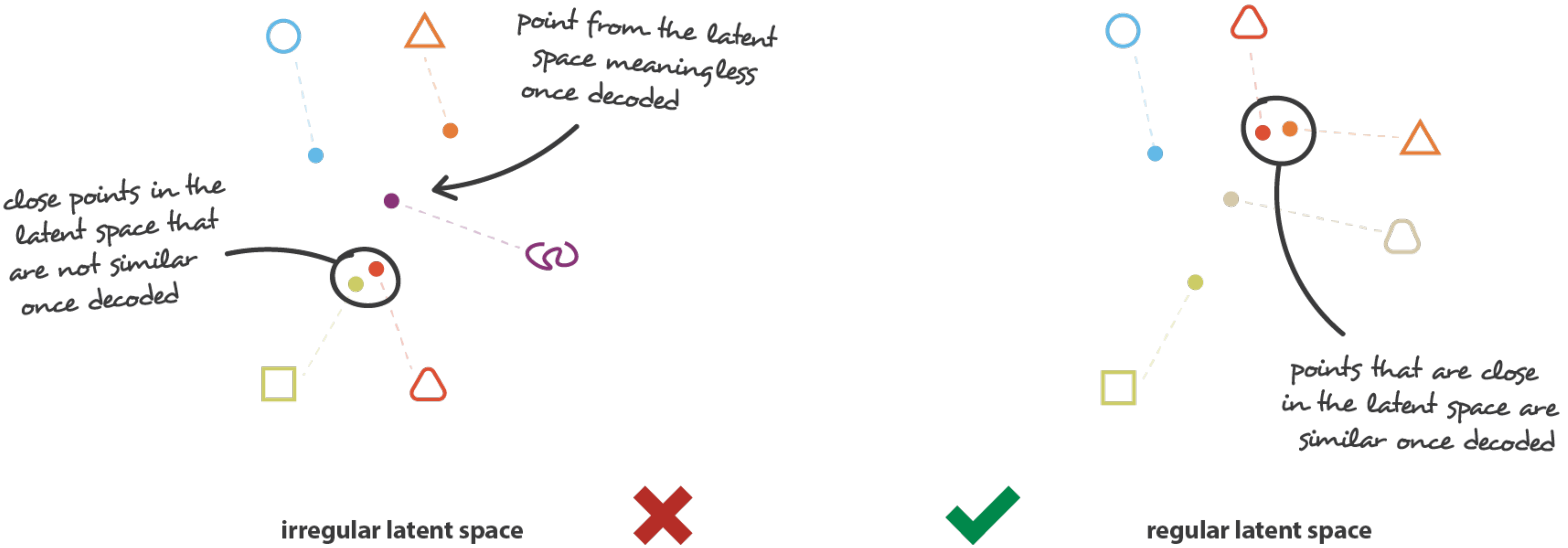


$$\text{loss} = \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

In variational autoencoders, the loss function is composed of a **reconstruction term** (that makes the encoding-decoding scheme efficient) and a **regularisation term** (that makes the latent space regular).

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

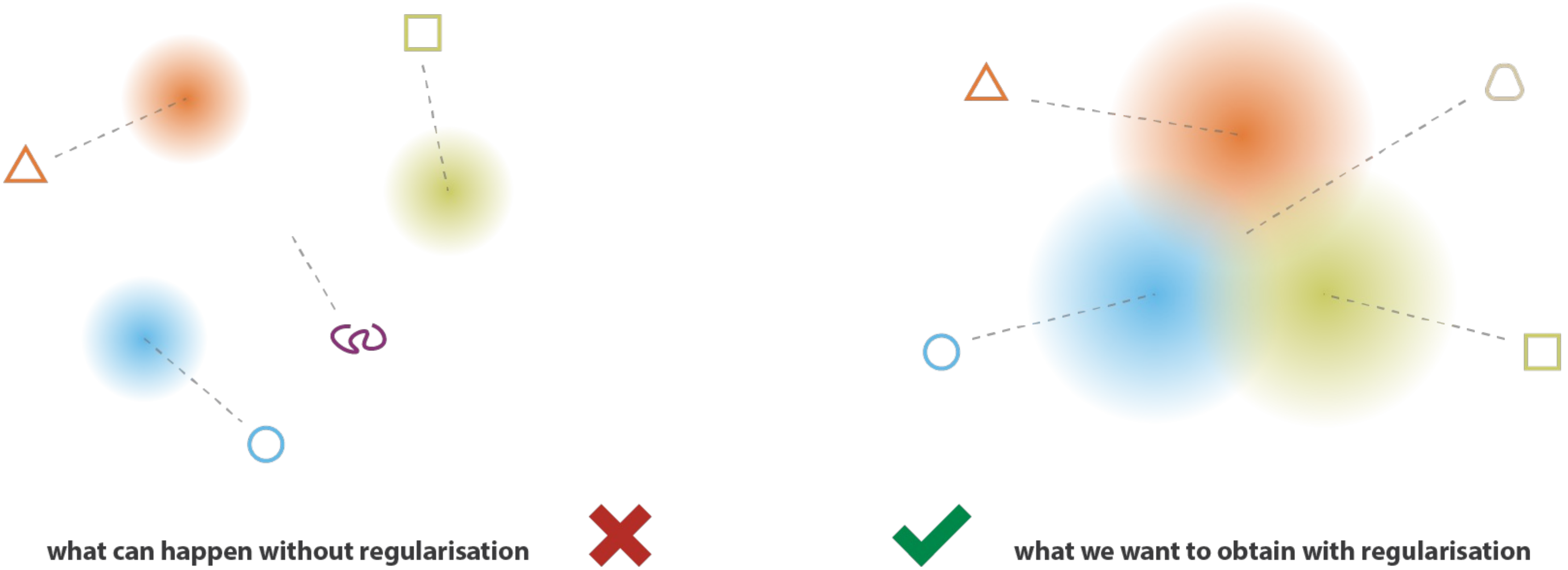
Variational Auto-encoders (VAEs)



Difference between a “regular” and an “irregular” latent space.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

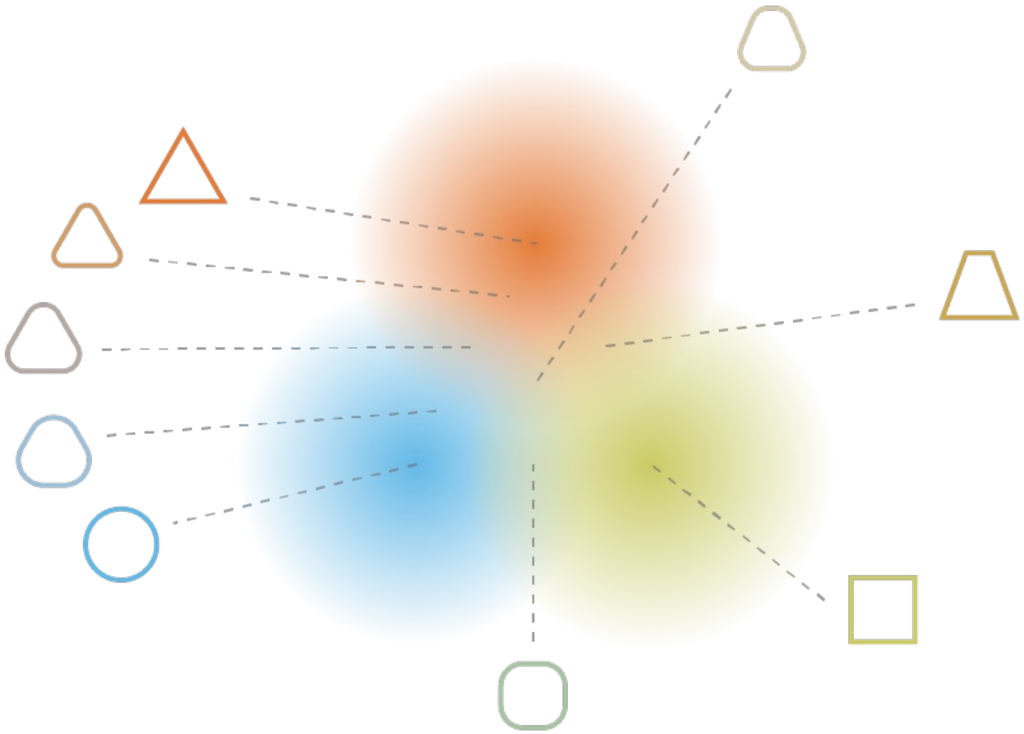
Variational Auto-encoders (VAEs)



The returned distributions of VAEs have to be regularised to obtain a latent space with good properties.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

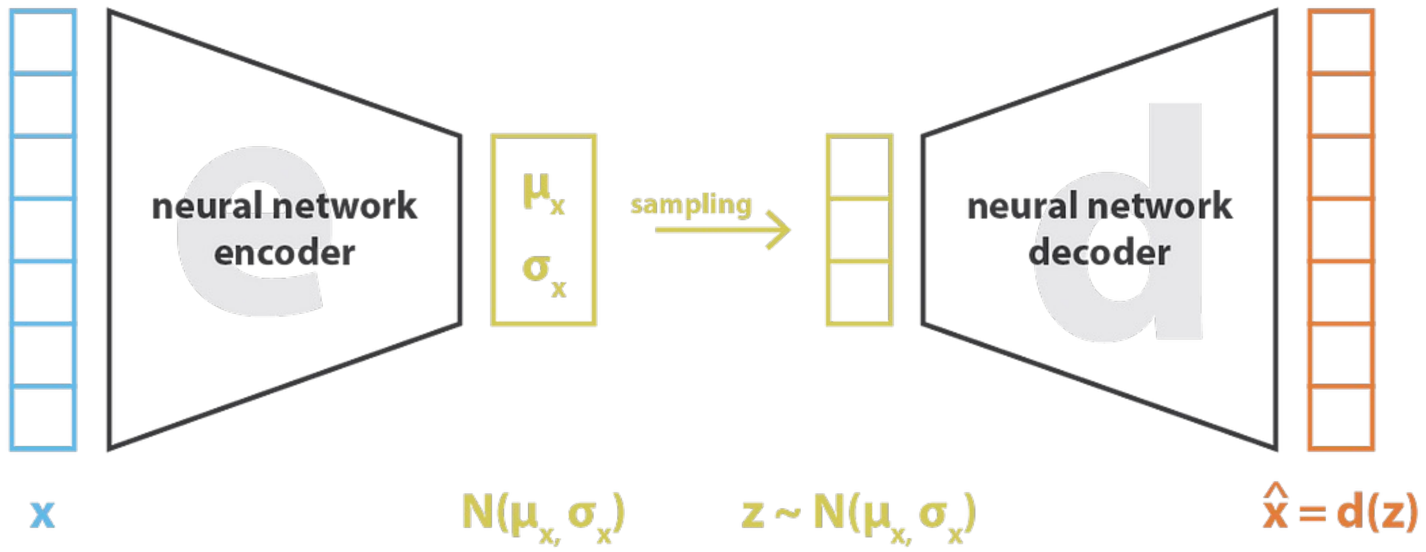
Variational Auto-encoders (VAEs)



Regularisation tends to create a “gradient” over the information encoded in the latent space.

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

Variational Auto-encoders (VAEs)

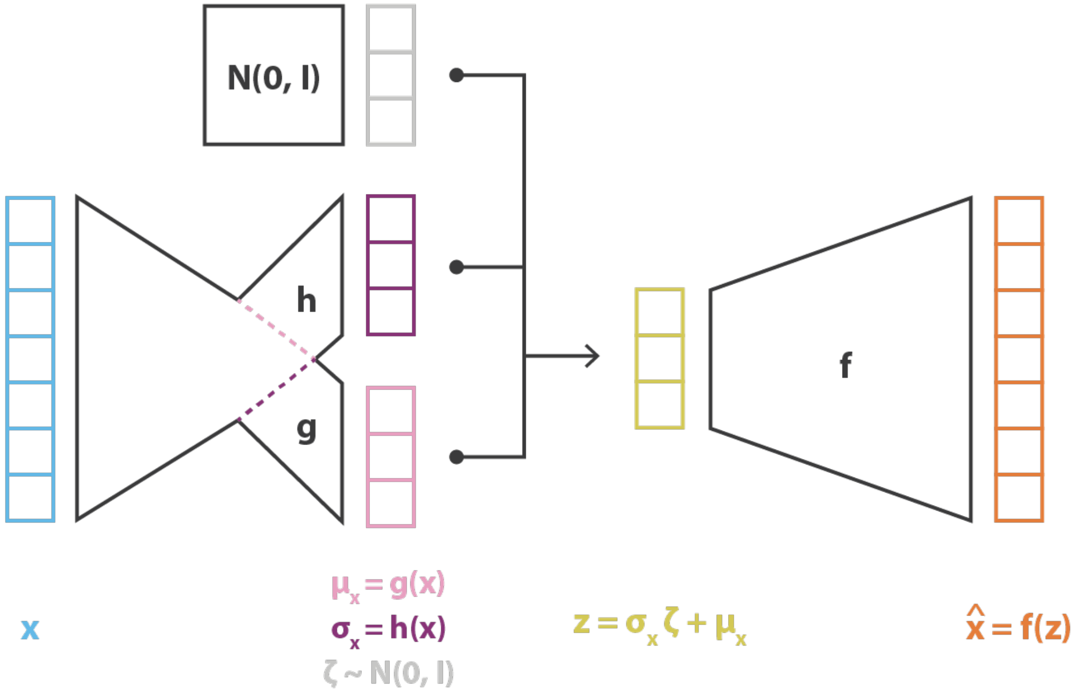


$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,]$$

In variational autoencoders, the loss function is composed of a **reconstruction term** (that makes the encoding-decoding scheme efficient) and a **regularisation term** (that makes the latent space regular).

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

Variational Auto-encoders (VAEs)



$$\text{loss} = C\,||\,x - \hat{x}\,||^2 + \mathrm{KL}[\,N(\mu_x, \sigma_x), N(0, I)\,] = C\,||\,x - f(z)\,||^2 + \mathrm{KL}[\,N(g(x), h(x)), N(0, I)\,]$$

Understanding Variational Autoencoders (VAEs), J. Rocca, 2019

VAEs for image generation

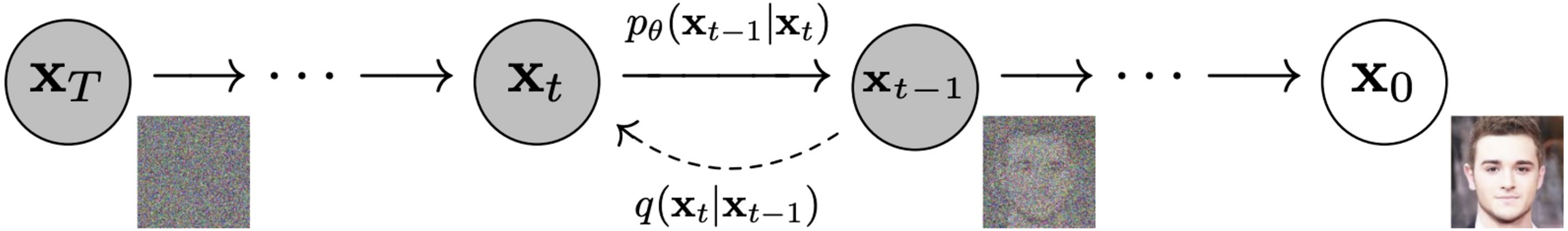
Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

Generating Diverse High-Fidelity Images with VQ-VAE-2, Ali Razavi, Aaron van der Oord, Oriol Vinyals, 2019

Agenda for today

1. VAEs
2. Diffusion

The principle of diffusion



Denoising Diffusion Probabilistic Models (DDPM), Jonathan Ho, Ajay Jain and Pieter Abbeel, NIPS 2020

Schedulers

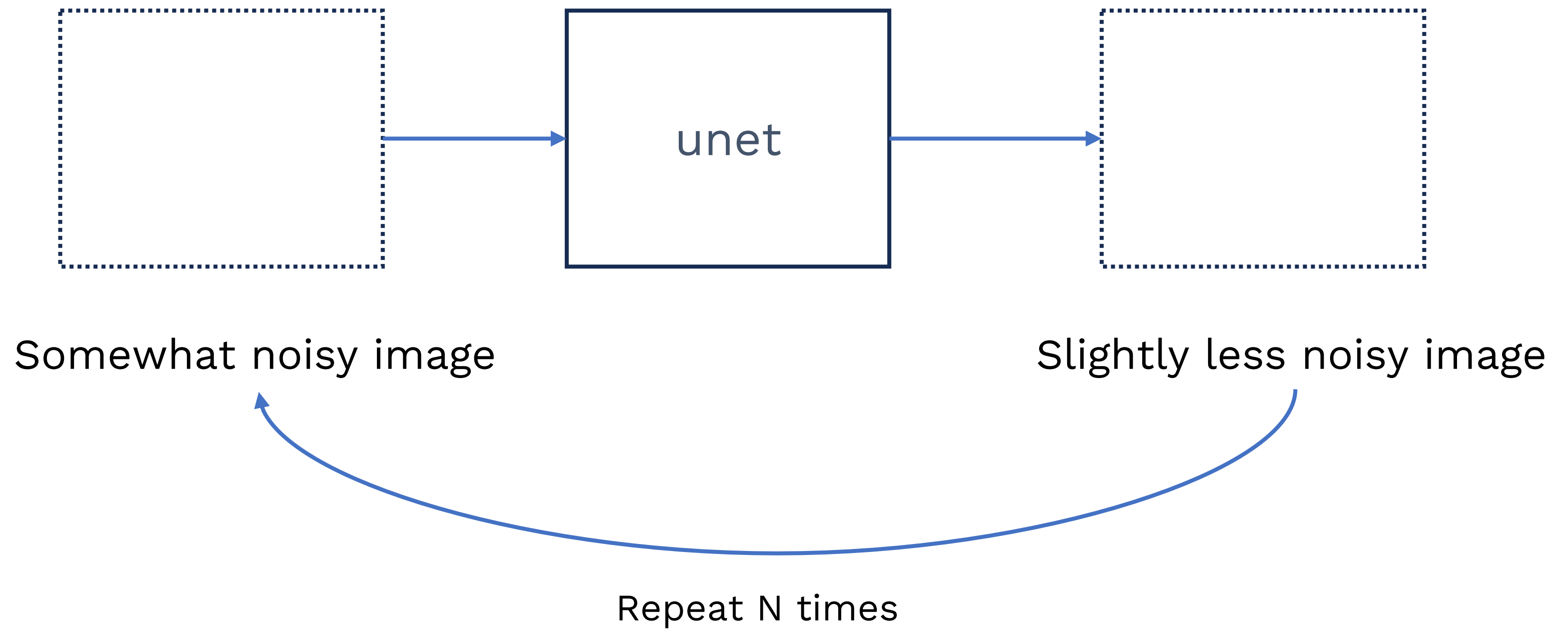
Schedulers define the methodology for iteratively adding noise to an image or for updating a sample based on model outputs.

- How to add noise for training
- How to update a sample based on an output from a pretrained model for inference

Schedulers are often defined by a *noise schedule* and an *update rule* to solve the differential equation solution.

Improved Denoising Diffusion Probabilistic Models, Nichol and Dhariwal, 2021

Stable Diffusion



U-Net

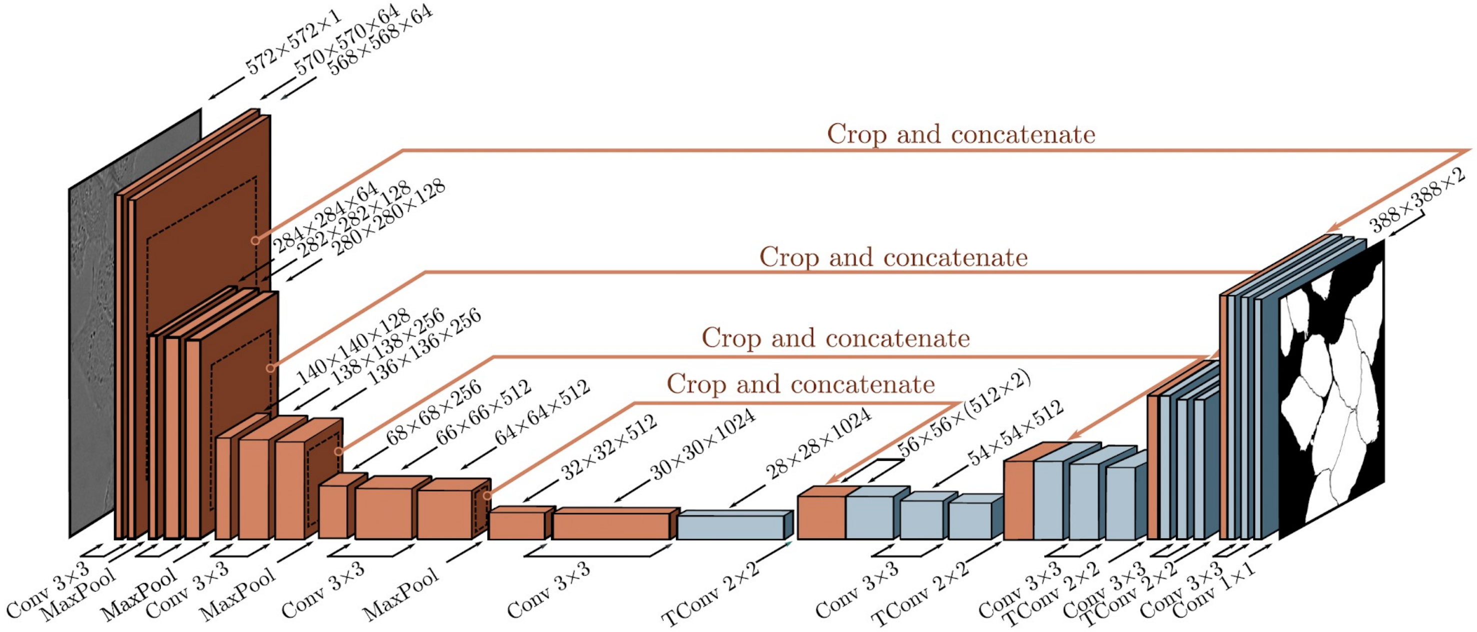
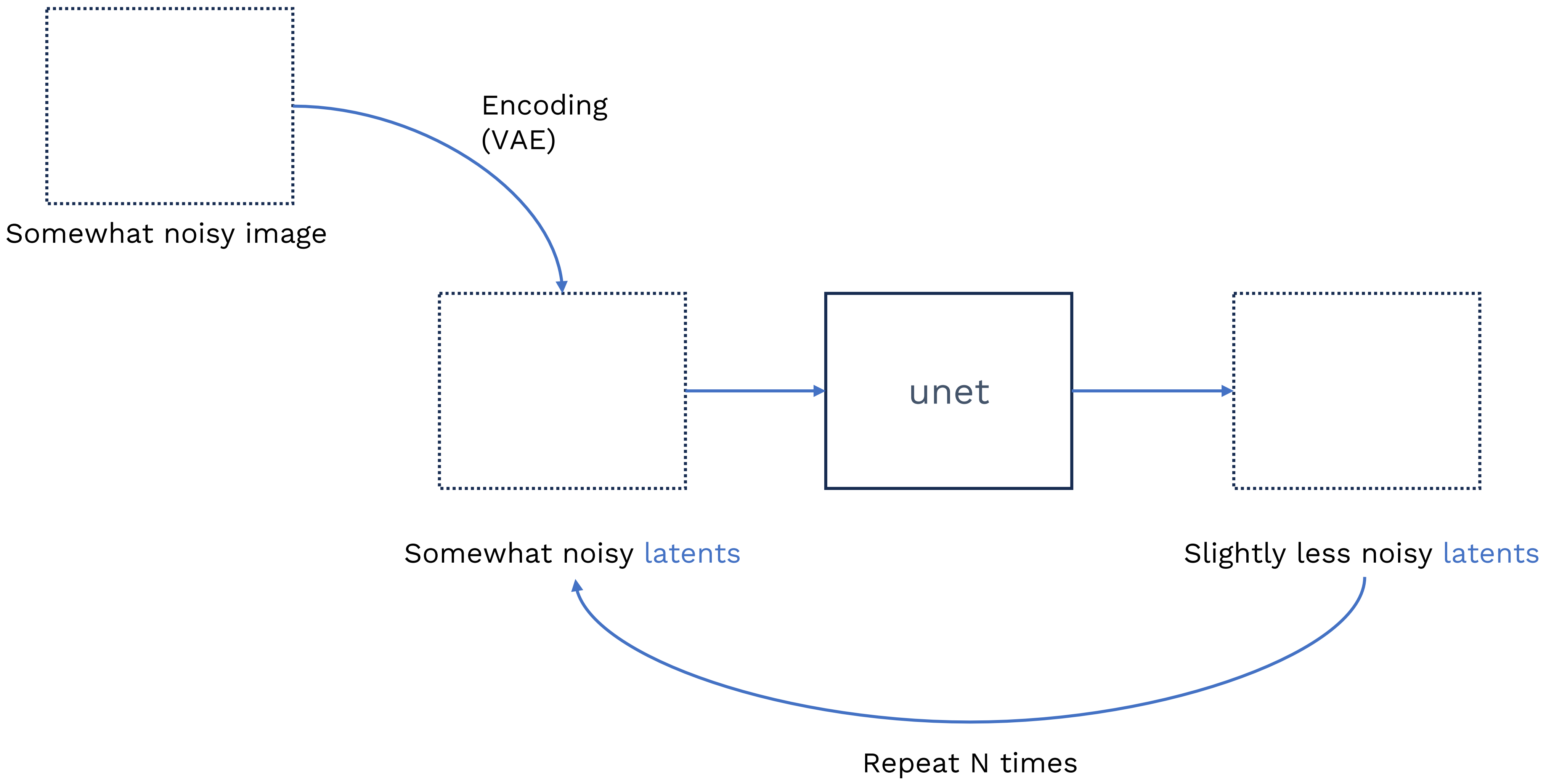


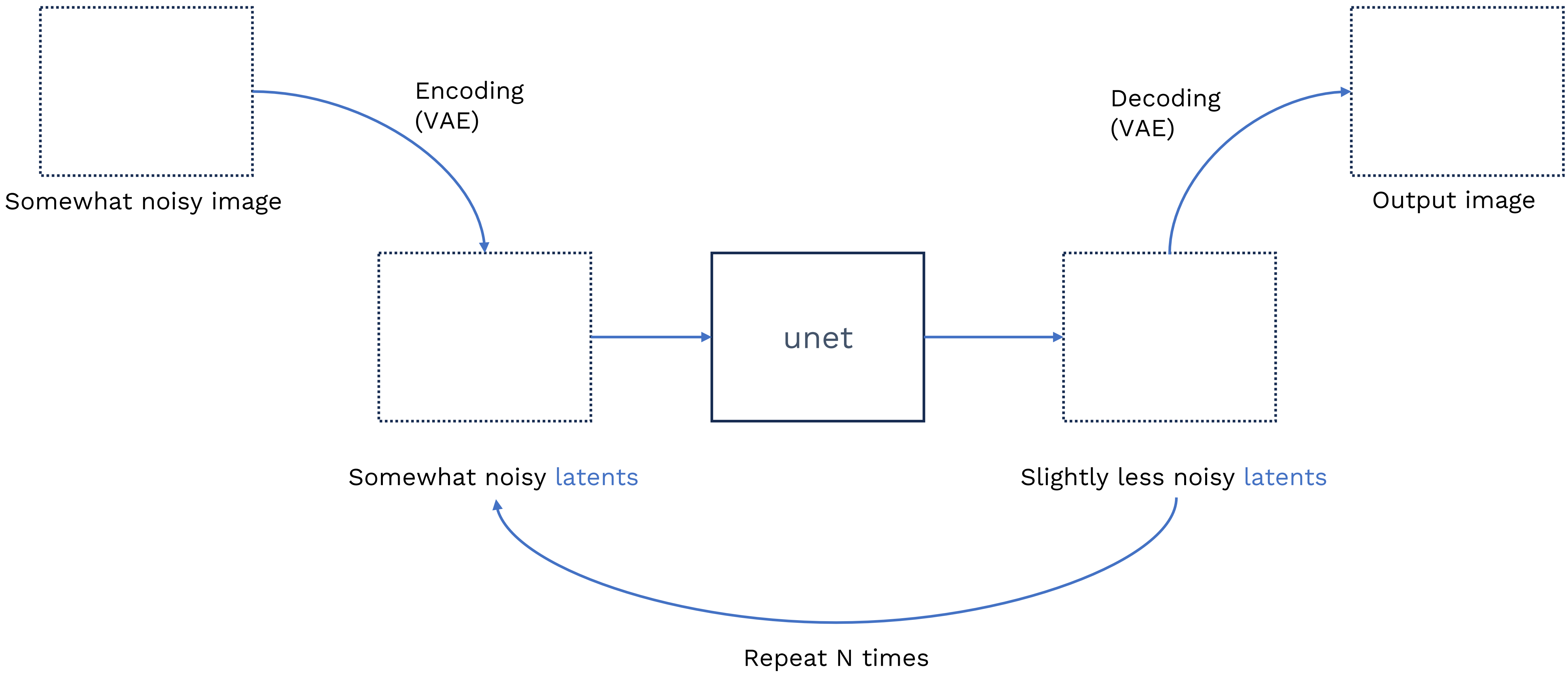
Figure 11.10 U-Net for segmenting HeLa cells. The U-Net has an encoder-decoder structure, in which the representation is downsampled (orange blocks) and then re-upsampled (blue blocks). The encoder uses regular convolutions, and the decoder uses transposed convolutions. Residual connections append the last representation at each scale in the encoder to the first representation at the same scale in the decoder (orange arrows). The original U-Net used "valid" convolutions, so the size decreased slightly with each layer, even without downsampling. Hence, the representations from the encoder were cropped (dashed squares) before appending to the decoder. Adapted from Ronneberger et al. (2015).

Source: Understanding Deep Learning, S. Prince, 2023

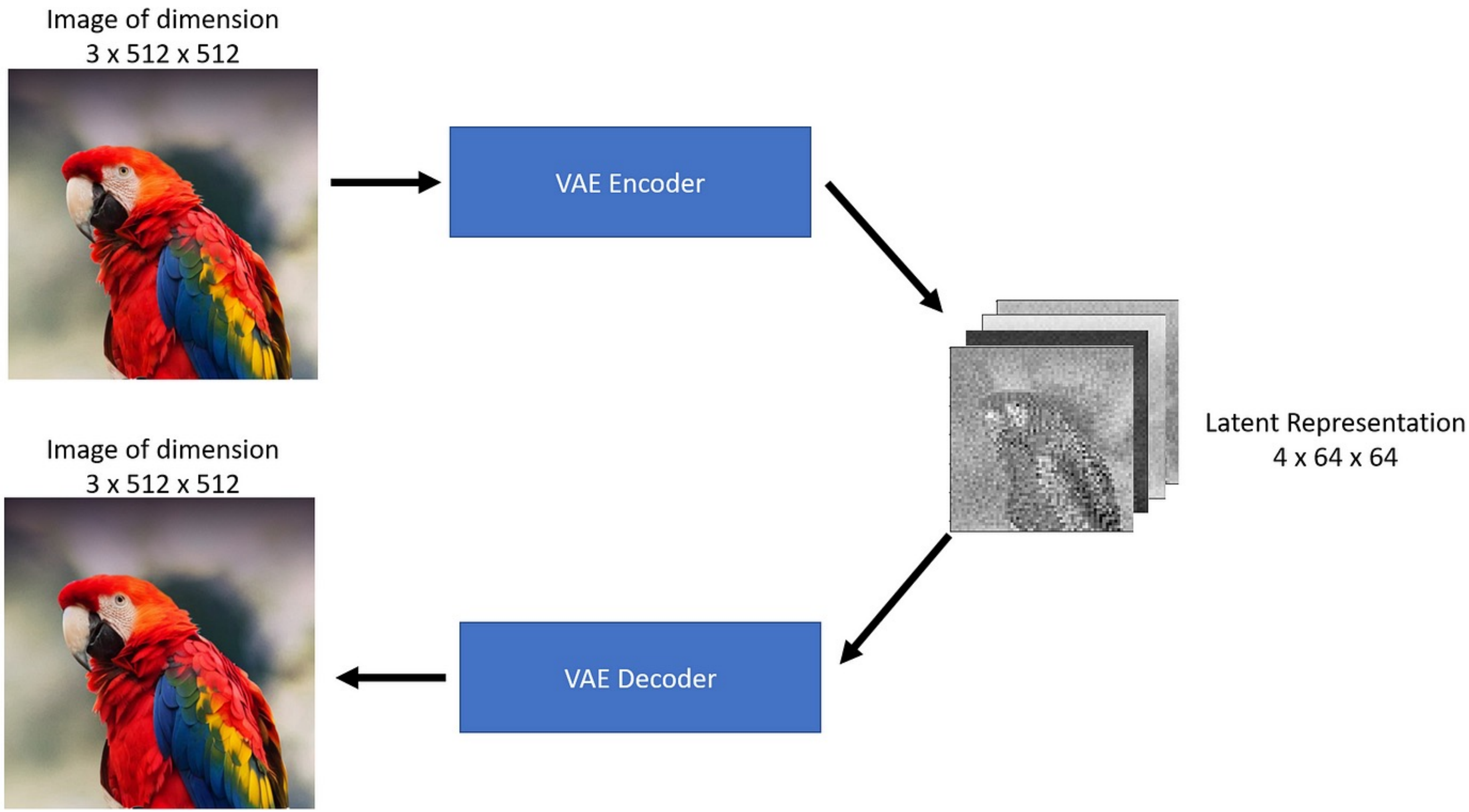
Stable Diffusion



Stable Diffusion

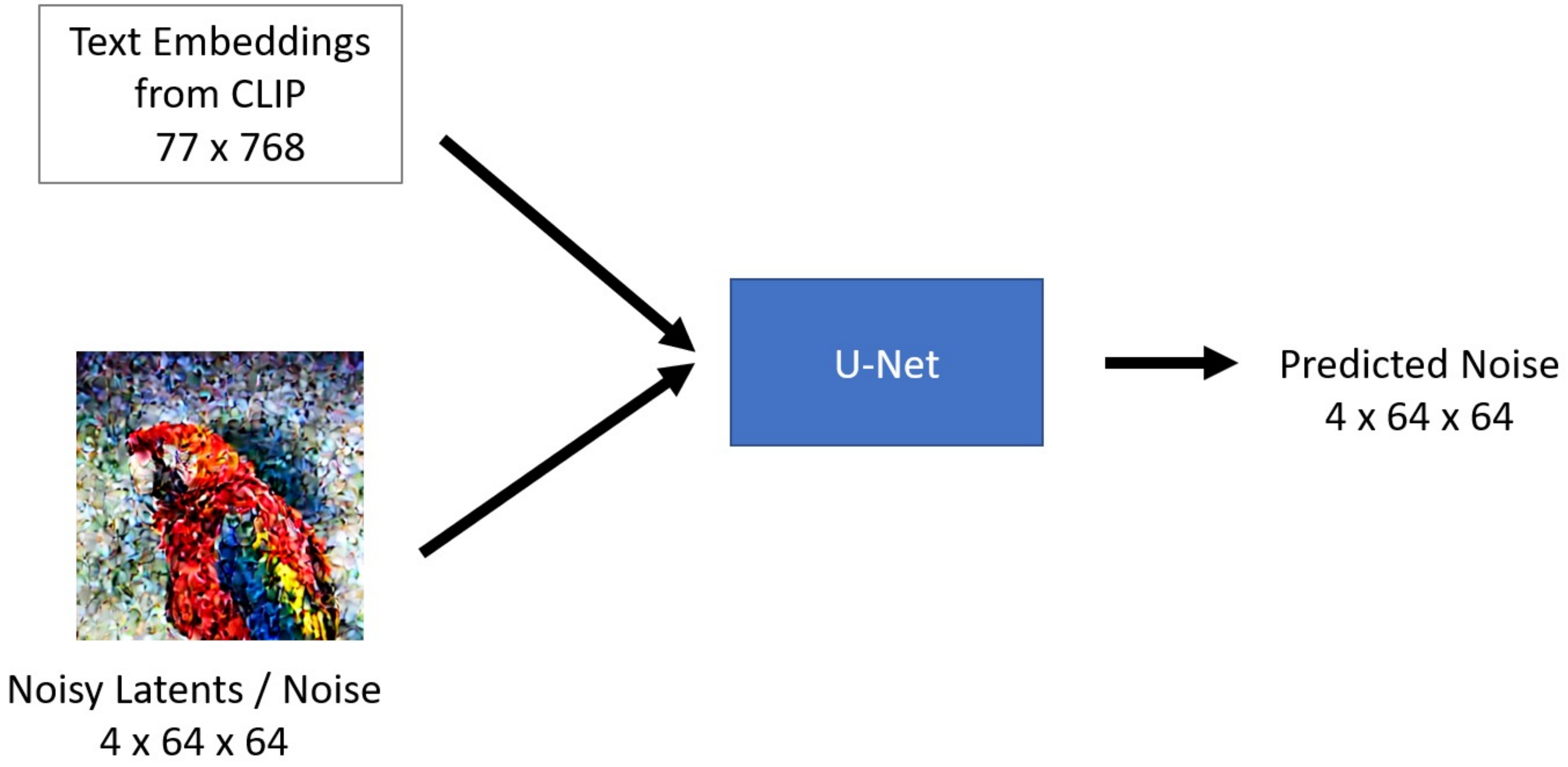


VAE Encoder



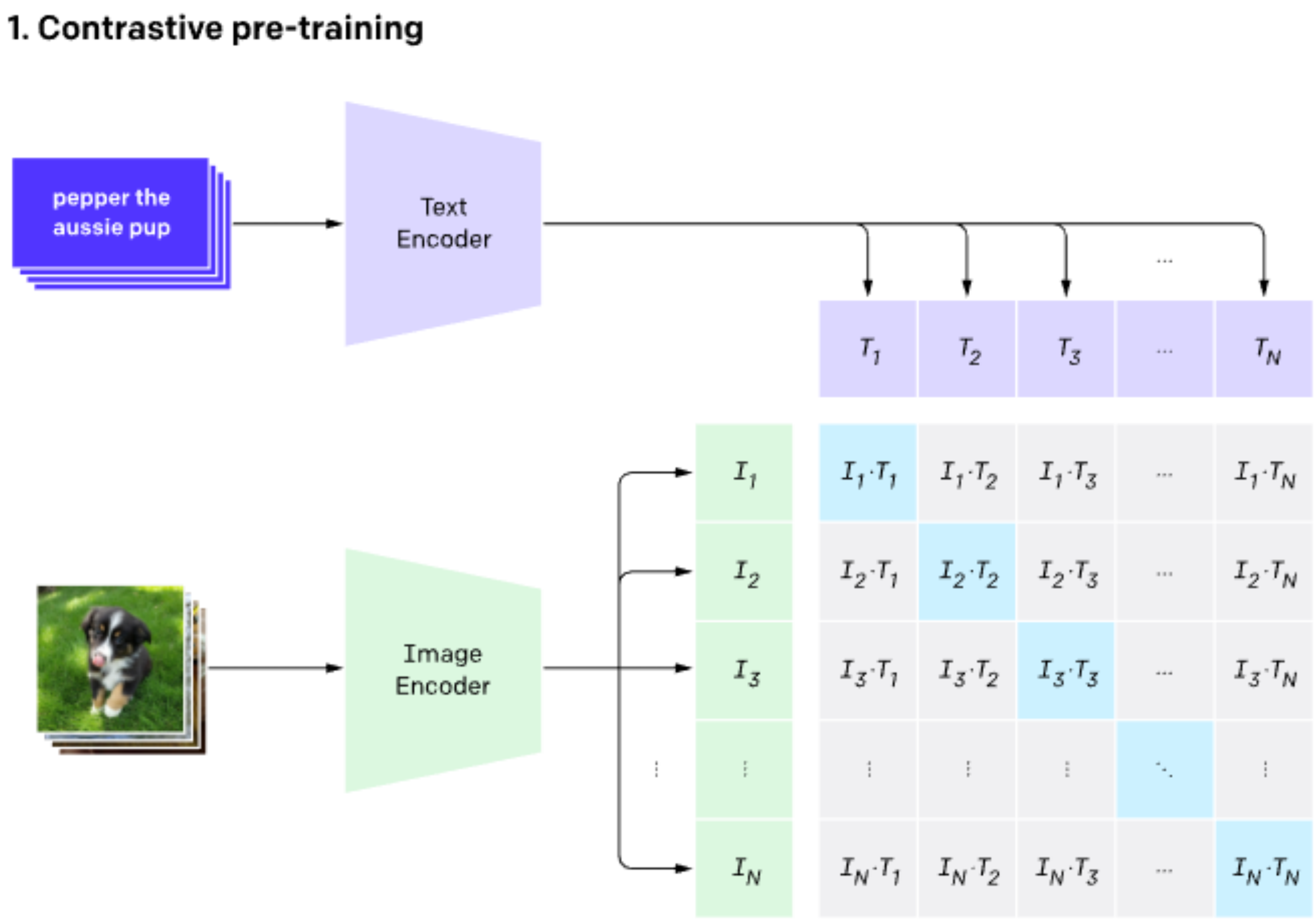
Original bird pic credit.

Guiding the diffusion with CLIP text encoding



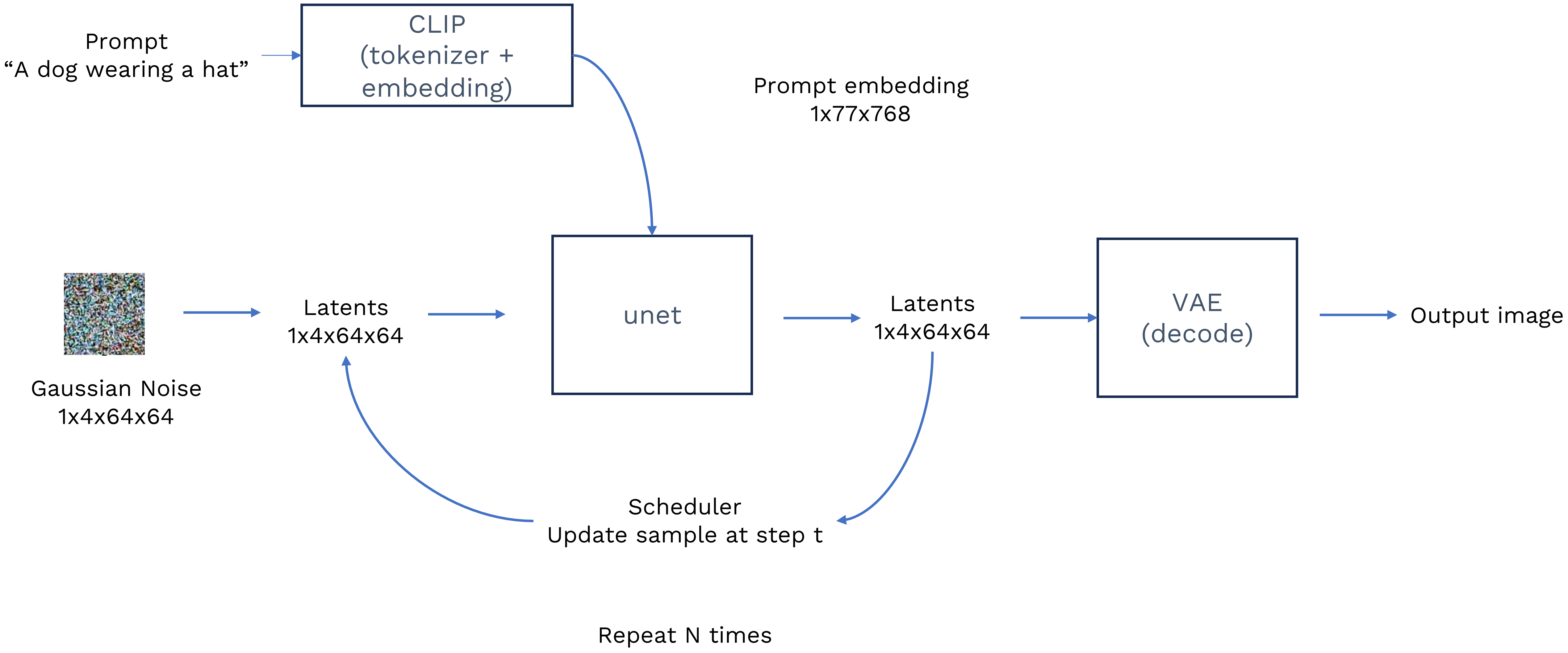
Original bird pic credit.

CLIP (Contrastive Language-Image Pretraining)

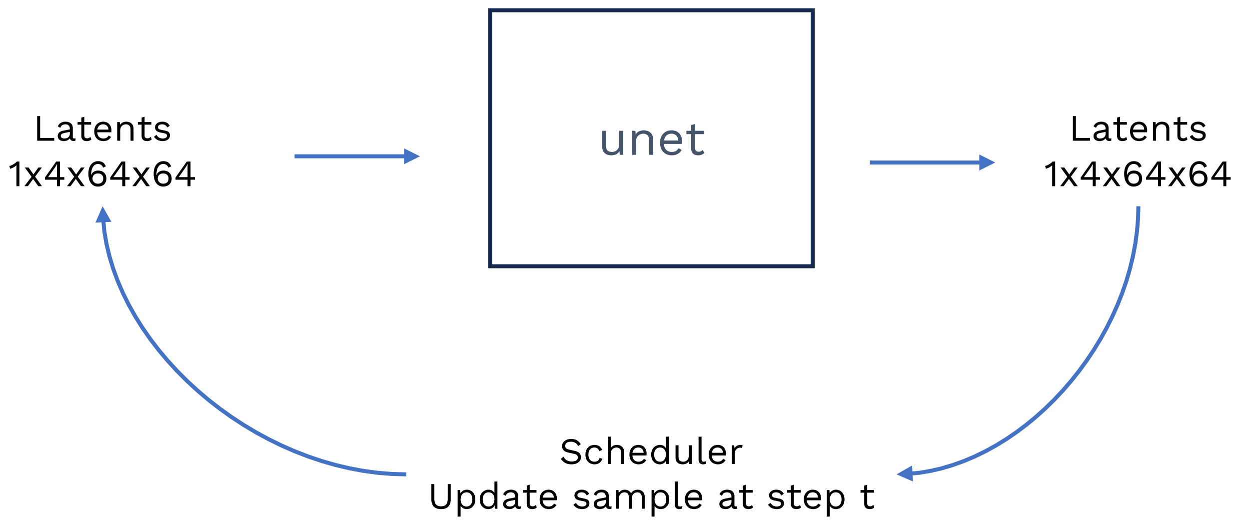


Credit — OpenAI

Stable Diffusion - Putting it all together



Common misconception



The U-Net does not predict noise between step t-1 and step t
The U-Net predicts the *entire* noise
The scheduler takes care of removing part of the noise
See [demo]

Stable Diffusion in 15 lines of code

```
tokenizer = CLIPTokenizer.from_pretrained("openai/clip-vit-large-patch14",
torch_dtype=torch.float16)

text_encoder = CLIPTextModel.from_pretrained("openai/clip-vit-large-patch14",
torch_dtype=torch.float16).to("cuda")

vae = AutoencoderKL.from_pretrained("stabilityai/sd-vae-ft-ema",
torch_dtype=torch.float16).to("cuda")

unet = UNet2DConditionModel.from_pretrained("CompVis/stable-diffusion-v1-4",
subfolder="unet", torch_dtype=torch.float16).to("cuda")

beta_start,beta_end = 0.00085,0.012

scheduler = LMSDiscreteScheduler(beta_start=beta_start, beta_end=beta_end,
beta_schedule="scaled_linear", num_train_timesteps=1000)
```

Stable Diffusion in 15 lines of code

```
bs = len(prompts)
text = text_enc(prompts)
uncond = text_enc([""] * bs, text.shape[1])
emb = torch.cat([uncond, text])

latents = torch.randn((bs, unet.in_channels, height//8, width//8))
scheduler.set_timesteps(steps)
latents = latents.to("cuda").half() * scheduler.init_noise_sigma

for i,ts in enumerate(tqdm(scheduler.timesteps)):
    inp = scheduler.scale_model_input(torch.cat([latents] * 2), ts)

    with torch.no_grad():
        u,t = unet(inp, ts, encoder_hidden_states=emb).sample.chunk(2)
    pred = u + g*(t-u)
    latents = scheduler.step(pred, ts, latents).prev_sample

with torch.no_grad():
    return vae.decode(1 / 0.18215 * latents).sample
```

Going deeper

The maths of diffusion by Lilian Weng

Classifier-free guidance (CFG)

Denoising Diffusion Implicit Models (DDIM)

Diffusion Models for Text Generation (AR-Diffusion)

Diffusion Models in Reinforcement Learning (DDPO)

Applications in Molecular Design (GaUDI)

Practical 6: let’s build a diffusion model from scratch!