INF721

2024/2

Deep Learning

L20: Variational Autoencoders

Logistics

Last Lecture

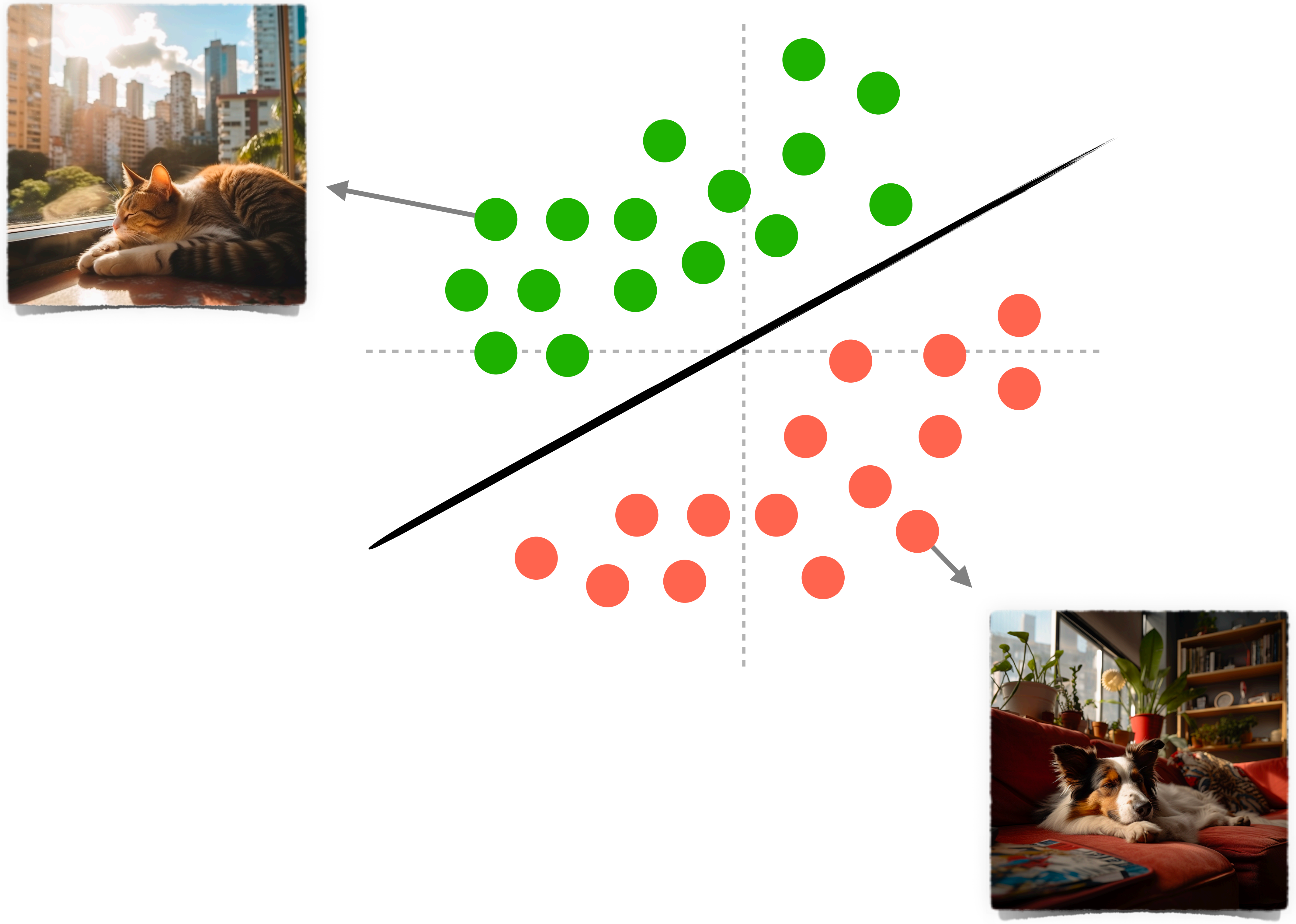
- Multimodal Learning
- Visual Transformers (ViT)
- Text-to-Speech
- Text-to-Image
- CLIP

Lecture Outline

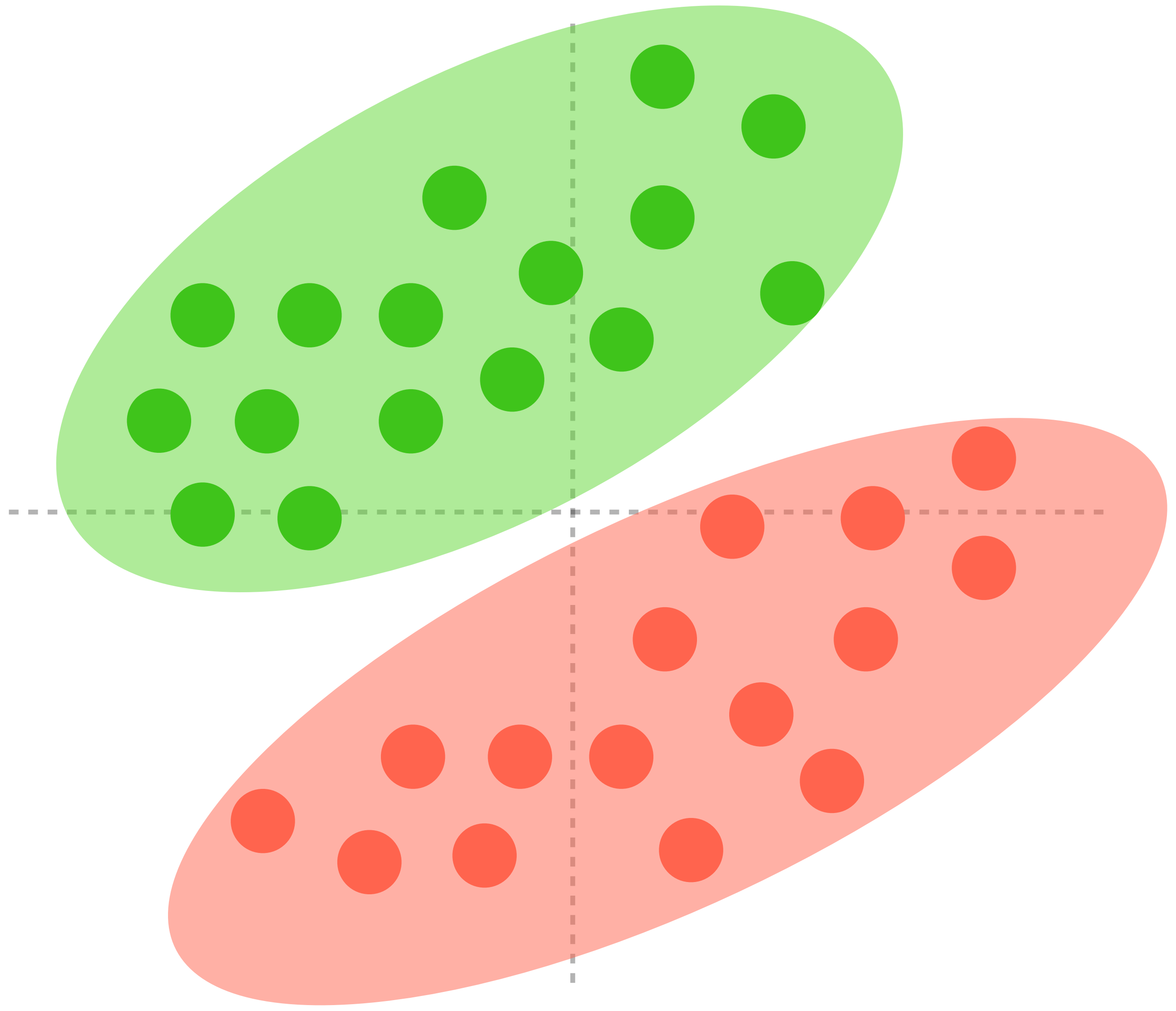
- Generative Modeling
- Text Generation
- Image Generation
- Autoencoders
  - Reconstruction Loss
- Variational Autoencoders
  - Kullback-Leibler Divergence

Discriminative vs. Generative Models

Learn a function $y = h(\mathbf{x})$ that maps an input feature vector $\mathbf{x}$ into a lavel y



Learn a function $h(\mathbf{x})$ that approximates the distribution that generated the samples $\mathbf{x}$



- Autoregressive Models
- Variational Autoencoders
- Generativa Adversarial Networks
- Diffusion Models

Autoregressive Models

In privivous lectures, we've discusses RNNs and Transformers as autoregressive models $P(x^{<t>} \mid x^{<t-1>}, \ldots, x^{<1>})$ to generate sequences:

- Text in Natural Language
- Source code in Programming Language
- Image Generation (as a sequence of pixels or patches)
- Music Generation
- Speech
- ...

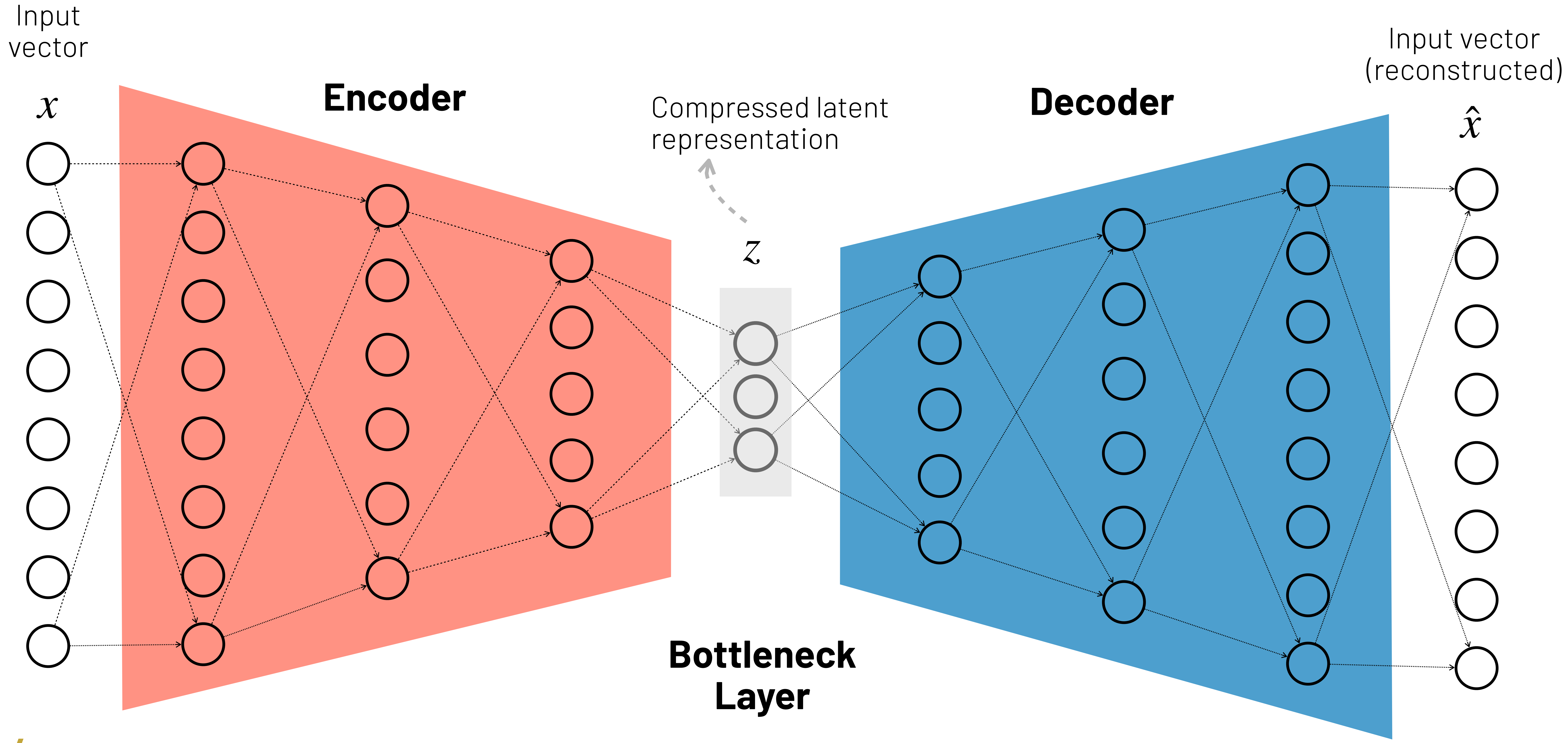
Image Generation

In next lectures, we will discuss image generation models that learn the probability distribution of pixels without treating them as a sequence:

- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models

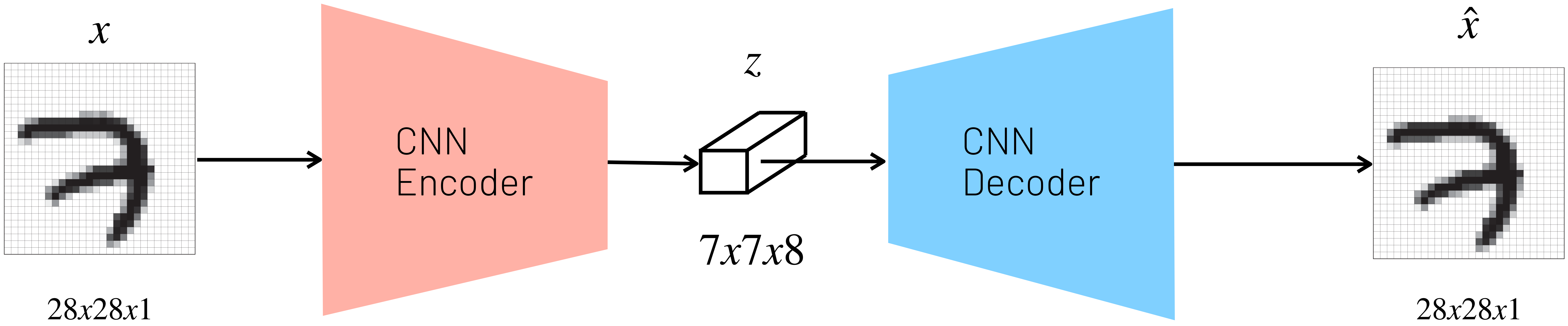
Autoencoder

Autoencoder is an encoder-decoder model designed to learn to compress and reconstruct data


Input vector
x
Encoder
Compressed latent representation
z
Decoder
Input vector (reconstructed)
$\hat{x}$
Bottleneck Layer

Example: Image Compression

We can apply autoencoders to learn compressed representations of images using a CNN as an encoder and decoder

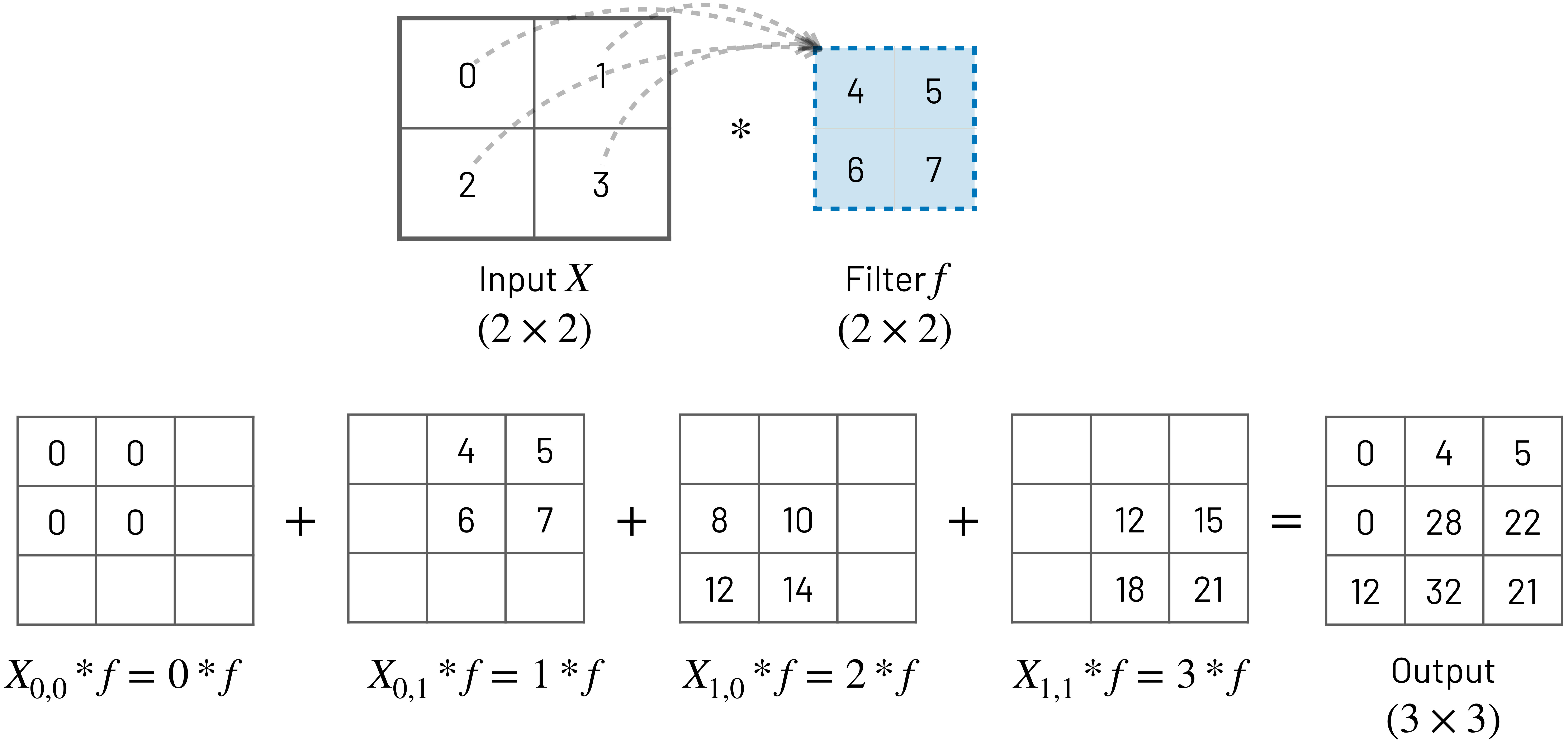


How to map a feature map back to an image?

Transposed Convolution!

Transposed Convolutions

A transposed convolutional layer slides the input over the kernel and performs element-wise multiplication and summation:



0	1
2	3

Input X
(2×2)

$*$

4	5
6	7

Filter f
(2×2)

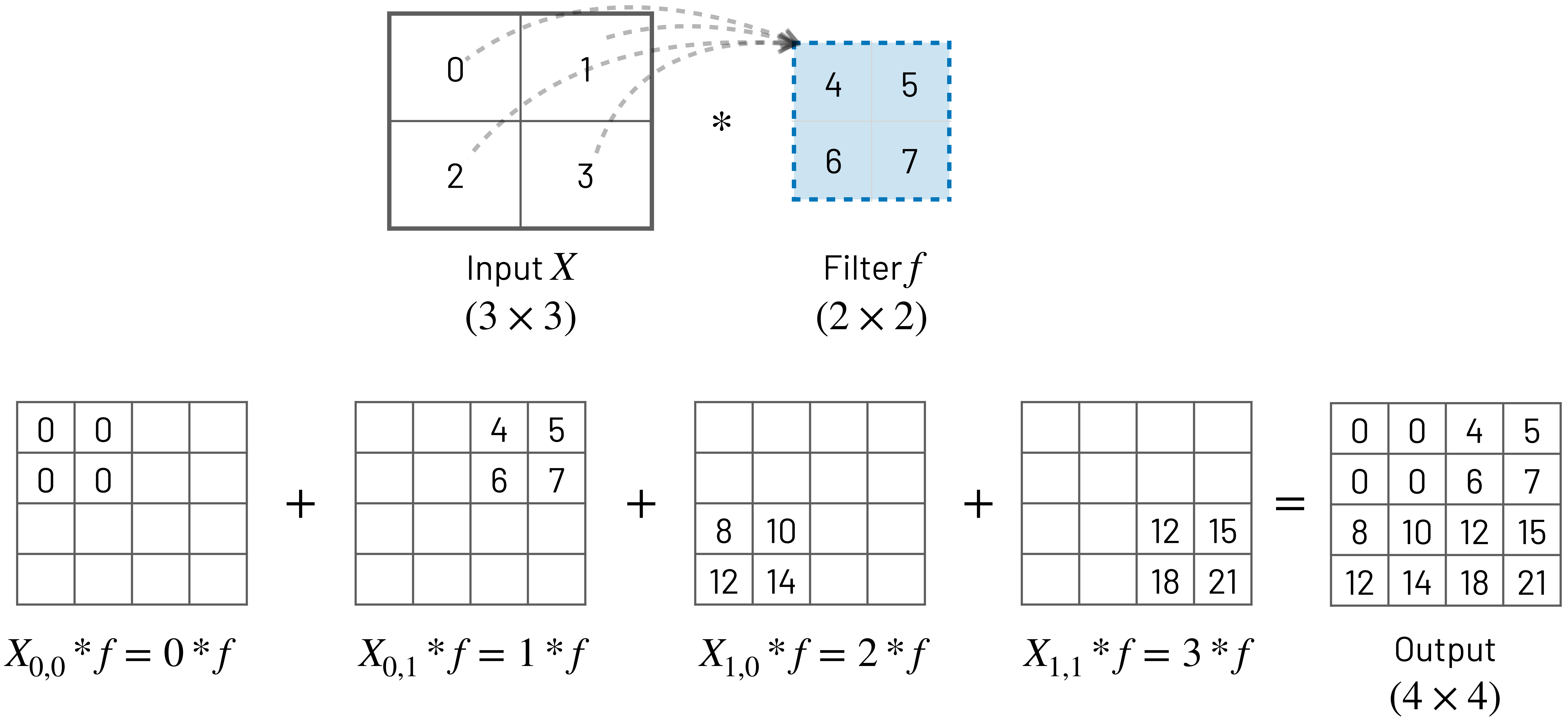
$$\begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & 0 & \\ \hline & & \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline & 4 & 5 \\ \hline & 6 & 7 \\ \hline & & \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline & & \\ \hline 8 & 10 & \\ \hline 12 & 14 & \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline & & \\ \hline & 12 & 15 \\ \hline & 18 & 21 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 0 & 4 & 5 \\ \hline 0 & 28 & 22 \\ \hline 12 & 32 & 21 \\ \hline \end{array}$$

$X_{0,0} * f = 0 * f$ $\quad X_{0,1} * f = 1 * f$ $\quad X_{1,0} * f = 2 * f$ $\quad X_{1,1} * f = 3 * f$

Output
(3×3)

Transposed Convolutions with Stride

In the transposed convolution, strides are specified for the the output, not for input. Let's performed the previous transposed convolution with stide of 2



0	1
2	3

Input X (3×3)

$*$

4	5
6	7

Filter f (2×2)

0	0		
0	0		

$X_{0,0} * f = 0 * f$

$+$

		4	5
		6	7

$X_{0,1} * f = 1 * f$

$+$

8	10		
12	14		

$X_{1,0} * f = 2 * f$

$+$

		12	15
		18	21

$X_{1,1} * f = 3 * f$

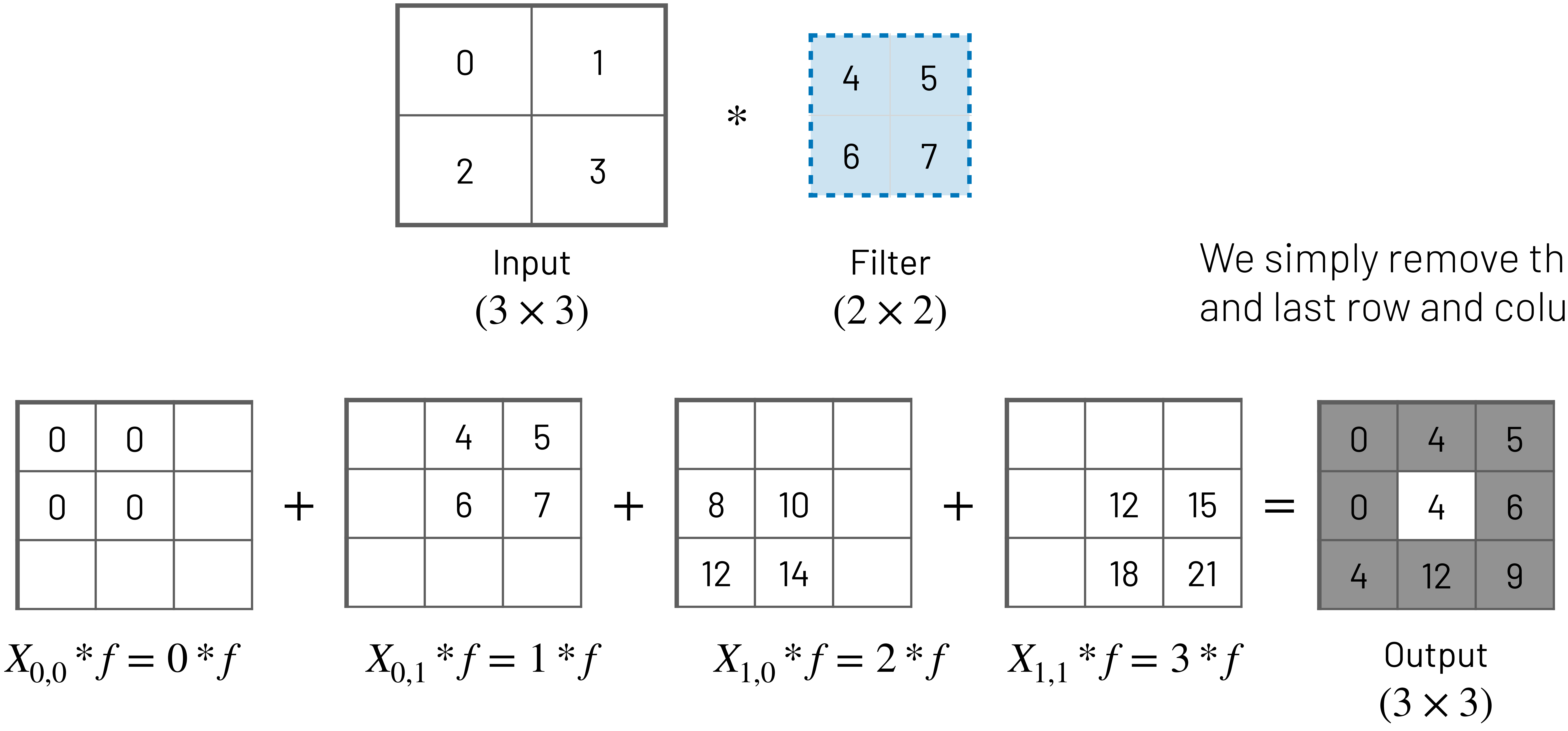
$=$

0	0	4	5
0	0	6	7
8	10	12	15
12	14	18	21

Output (4×4)

Transposed Convolutions with Padding

Different from in the regular convolution where padding is applied to input, it is applied to output in the transposed convolution.



$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}$$

Input (3×3)

Filter (2×2)

We simply remove the first and last row and column

$$\begin{bmatrix} 0 & 0 & \\ 0 & 0 & \\ & & \end{bmatrix} + \begin{bmatrix} & 4 & 5 \\ & 6 & 7 \\ & & \end{bmatrix} + \begin{bmatrix} & & \\ 8 & 10 & \\ 12 & 14 & \end{bmatrix} + \begin{bmatrix} & & \\ & 12 & 15 \\ & 18 & 21 \end{bmatrix} = \begin{bmatrix} 0 & 4 & 5 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{bmatrix}$$

$X_{0,0} * f = 0 * f$

$X_{0,1} * f = 1 * f$

$X_{1,0} * f = 2 * f$

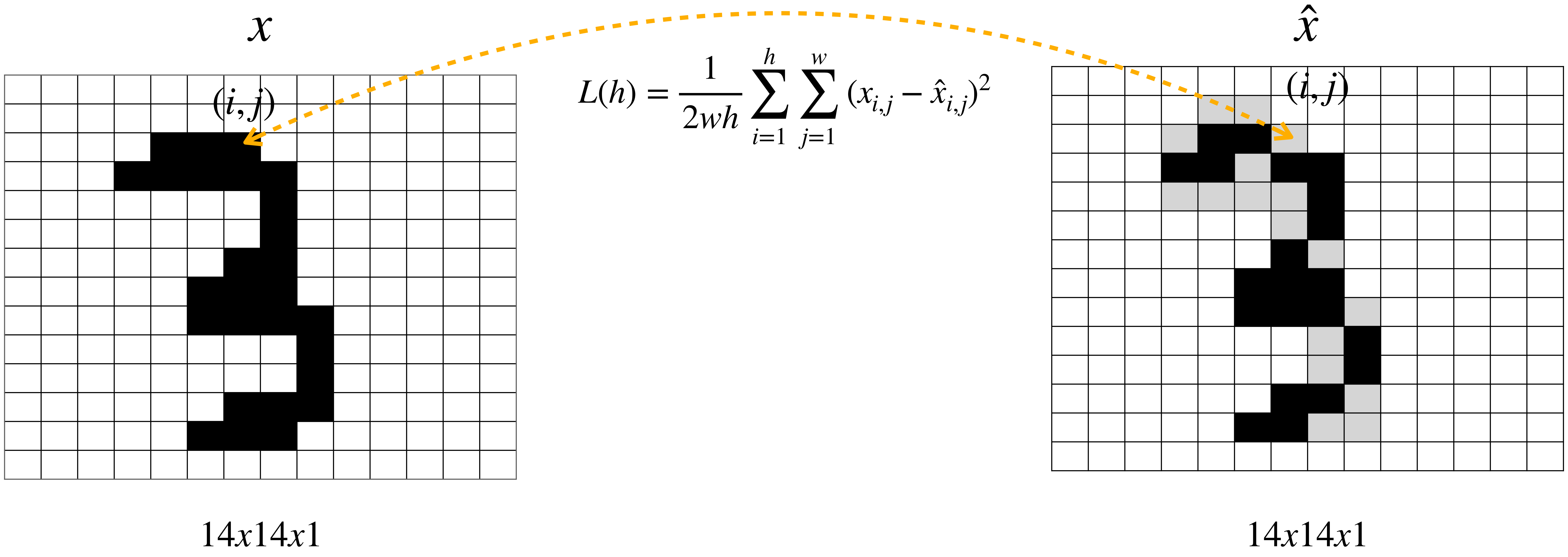
$X_{1,1} * f = 3 * f$

Output (3×3)

Resconstruction Loss

We train an autoencoder with a reconstruction loss, which measures the error between the original input image x from the reconstructed output $\hat{x}$

Error is computed pixel-by-pixel with MSE (or Binary Cross-Entropy)

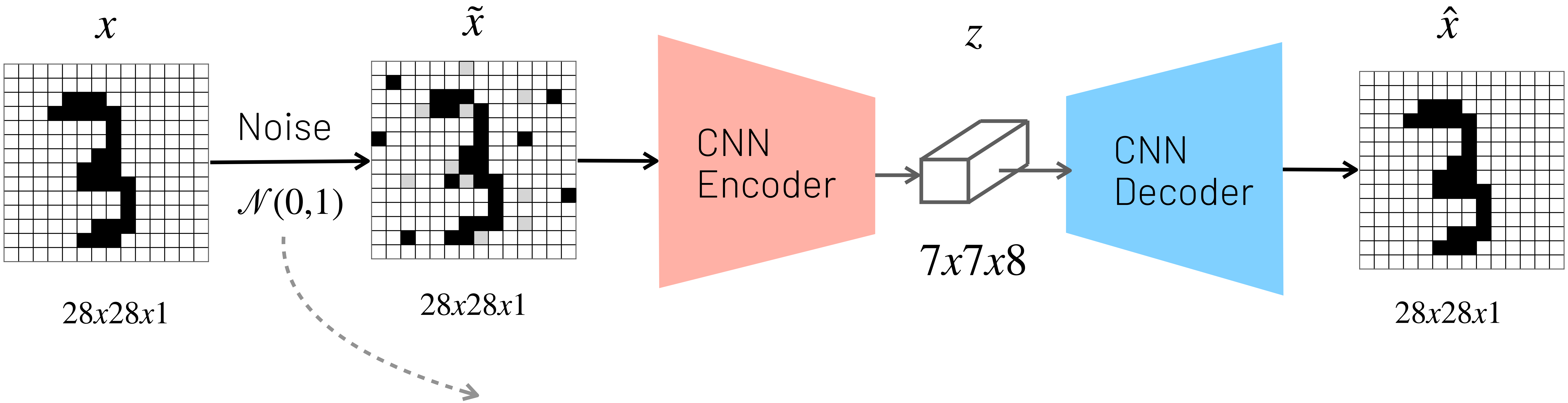


$$L(h) = \frac{1}{2wh} \sum_{i=1}^{h} \sum_{j=1}^{w} (x_{i,j} - \hat{x}_{i,j})^2$$

UFV

Denoising Autoencoders

Denoising Autoencoders are a type of autoencoder where we train the model to remove noise from the original data (e.g., image)



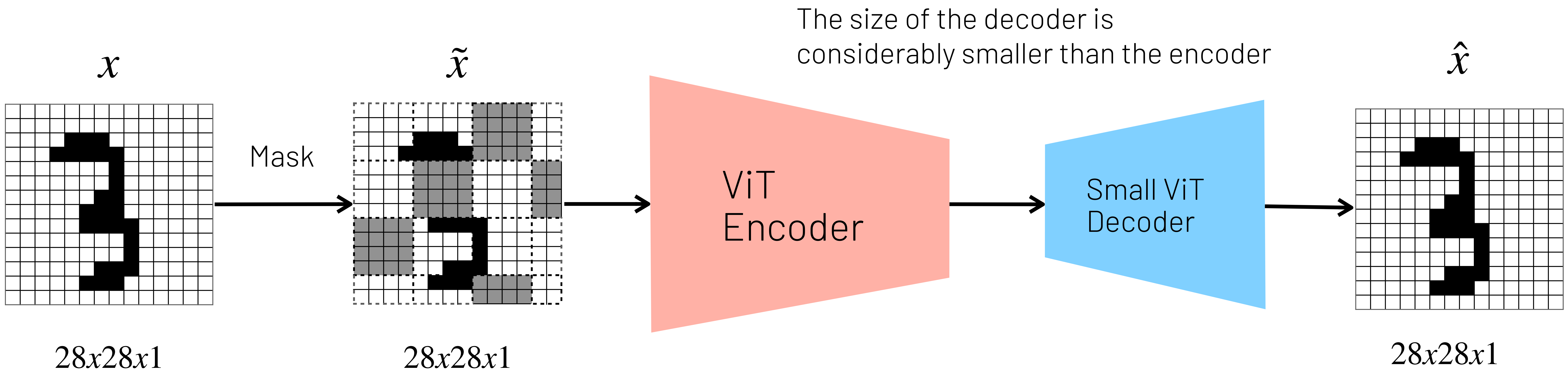
Adding noise forces the model to learn the internal structure of the data. To correct a given pixel, the model has to learn the correlation between its nearby pixels.

Vincent et al. 2008, Extracting and Composing Robust Features with Denoising Autoencoders

[1] This task is also called inpainting

Masked Autoencoders

Masked Autoencoders are another type of autoencoder where we train the model to complete[1] a masked patch of the original image (similar to BERT).



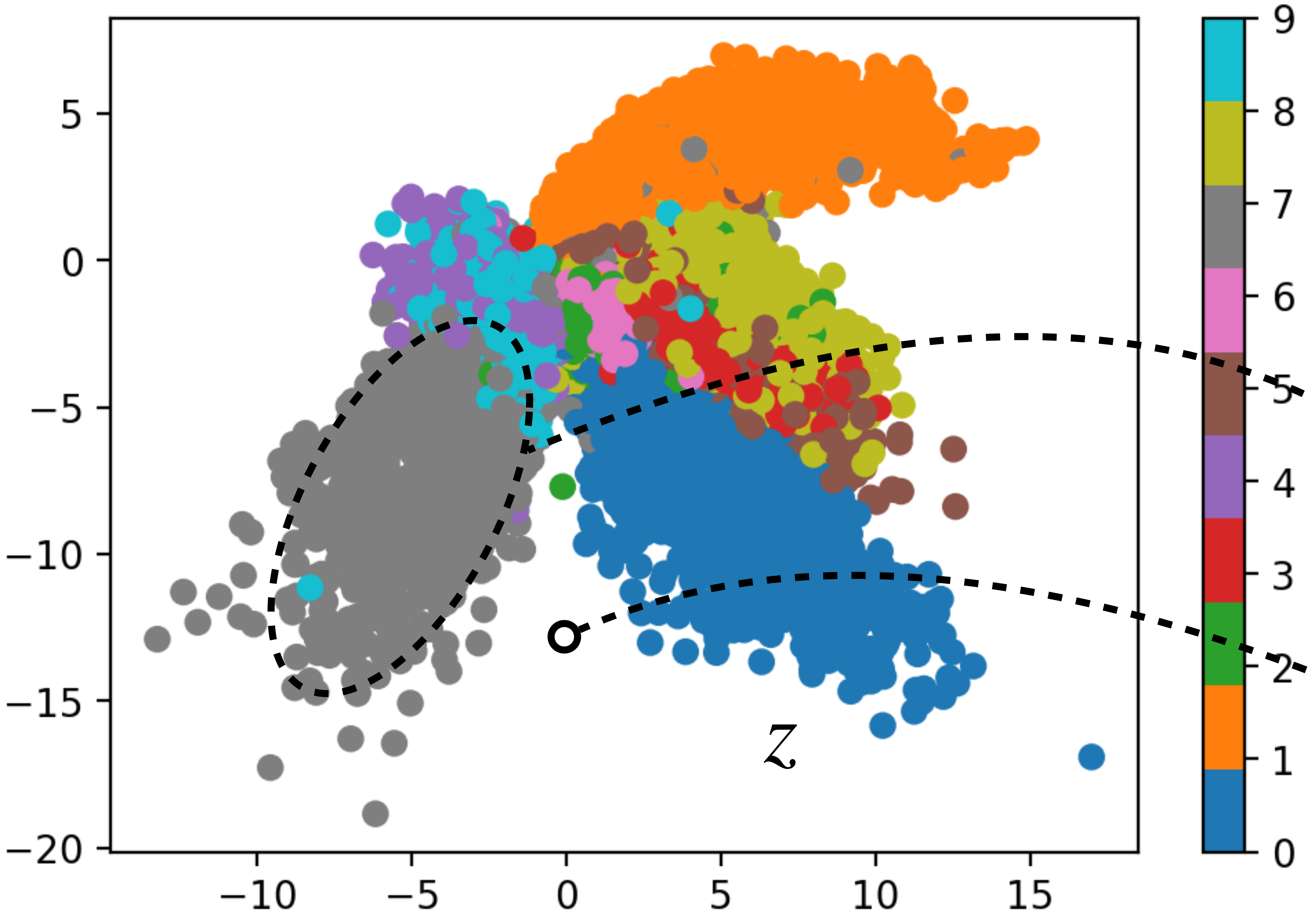
The size of the decoder is considerably smaller than the encoder

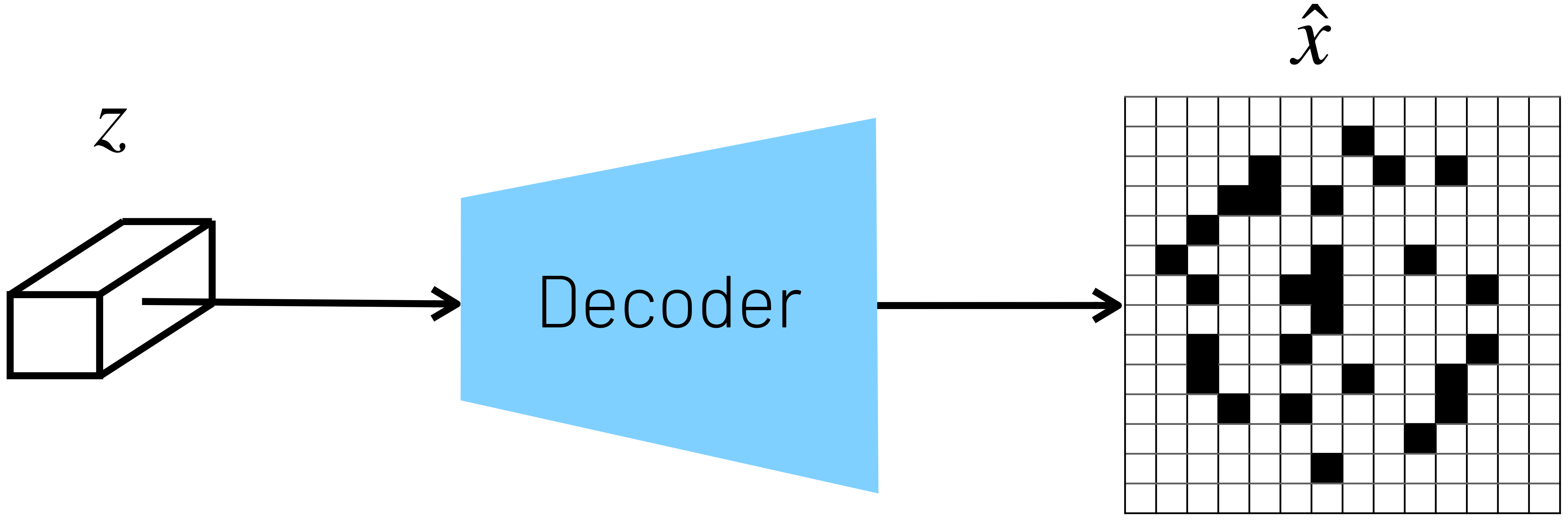
Like adding noise, masking is a form of imposing a constraint to avoid the model learning the identity function.

He et al., J. 2021, Masked Autoencoders Are Scalable Vision Learners

Decoding random latent images

After training an aucoencoder, we could try to decode random latent features maps z to generate new images, however the learned latent space might not be continuous!



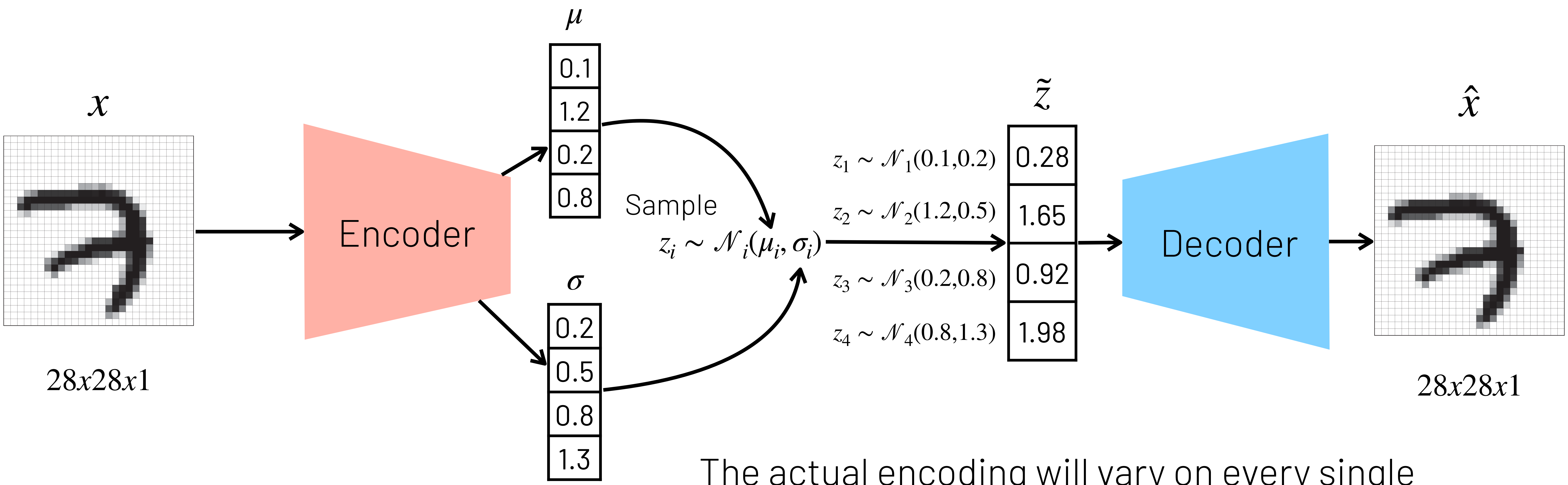


Examples are clustered in different regions of the learned latent space

But it has discontinuities, so if we try to decode a random latent vector z to generate an image, it might not be a realistic imge

Variational autoencoders (VAEs)

To solve that discontinuity problem, Variational Autoencoders (VAE) use the encoder to learn a probability distribuition P explicitly, and sample the latent representation z from P



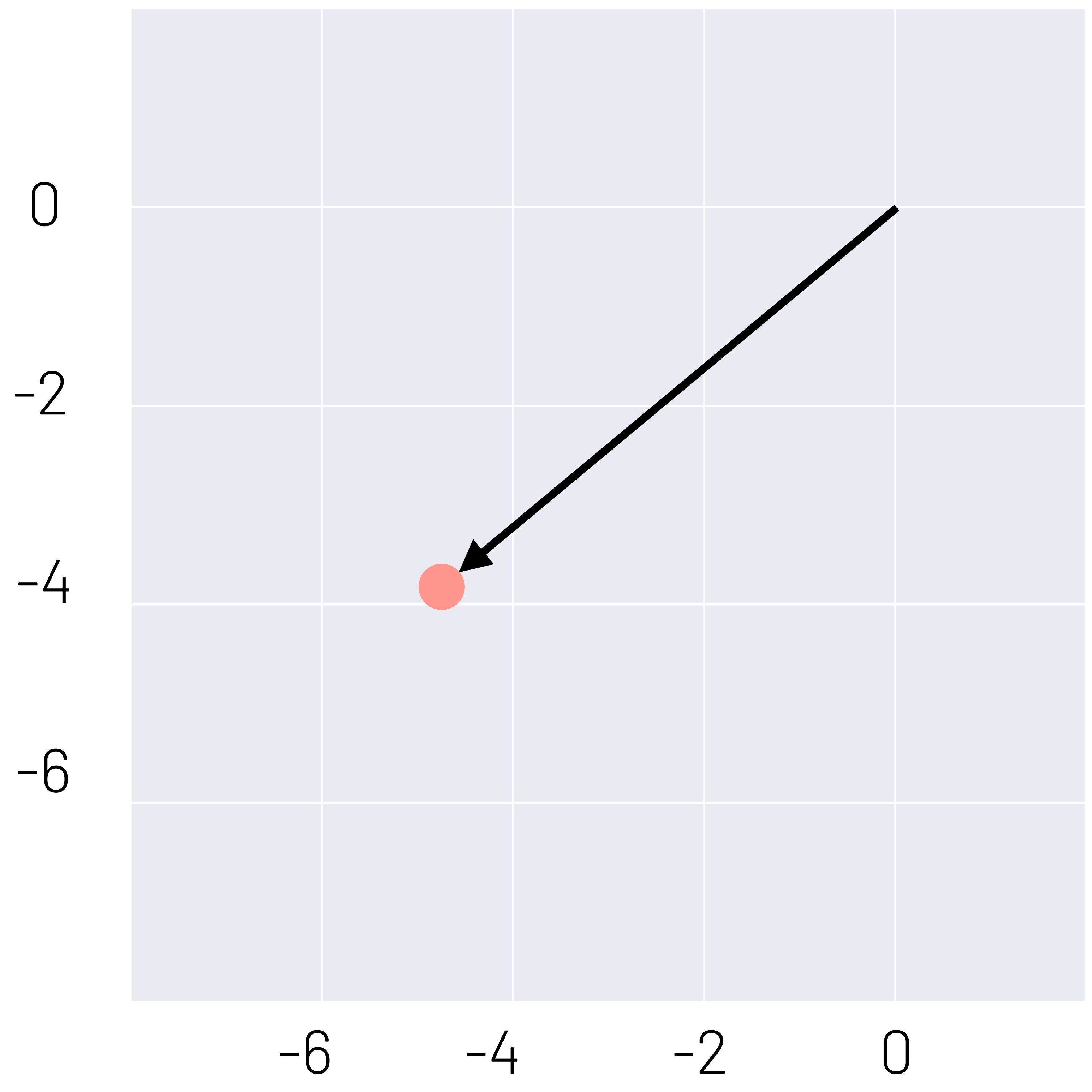
x
$28x28x1$
Encoder
μ
0.1
1.2
0.2
0.8
σ
0.2
0.5
0.8
1.3
Sample
$z_i \sim \mathcal{N}_i(\mu_i, \sigma_i)$
$z_1 \sim \mathcal{N}_1(0.1,0.2)$
$z_2 \sim \mathcal{N}_2(1.2,0.5)$
$z_3 \sim \mathcal{N}_3(0.2,0.8)$
$z_4 \sim \mathcal{N}_4(0.8,1.3)$
$\tilde{z}$
0.28
1.65
0.92
1.98
Decoder
$\hat{x}$
$28x28x1$

The actual encoding will vary on every single pass due to sampling.

Kingma, D. P. Welling, M. 2013, Auto-Encoding Variational Bayes

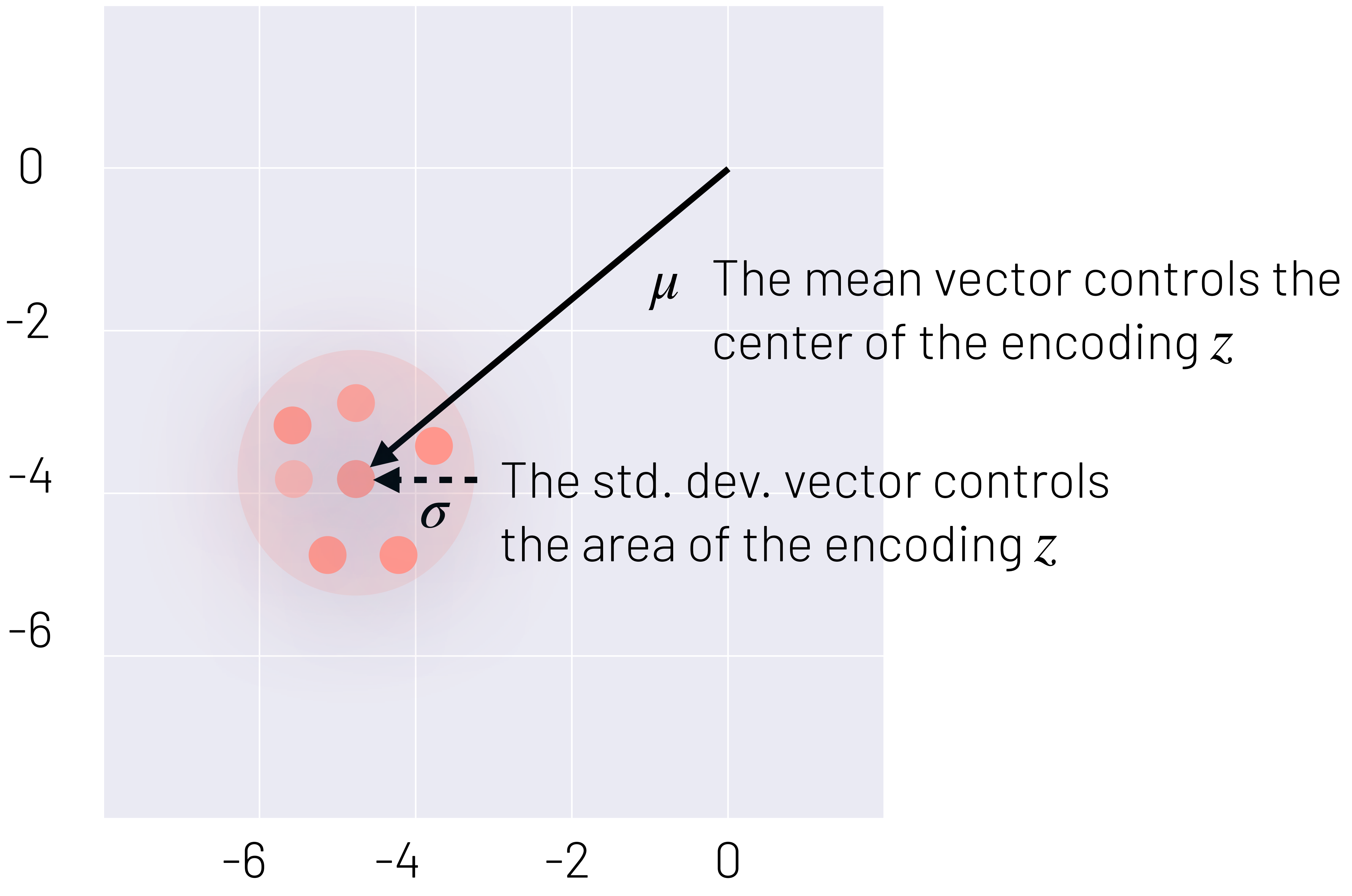
Autoencoders vs. VAEs

Autoencoders
(direct encoding coordinates)



- Only a single point in latent space refets to a sample of that class (discontinuity)

VAEs
(μ and σ define a probability distribution)



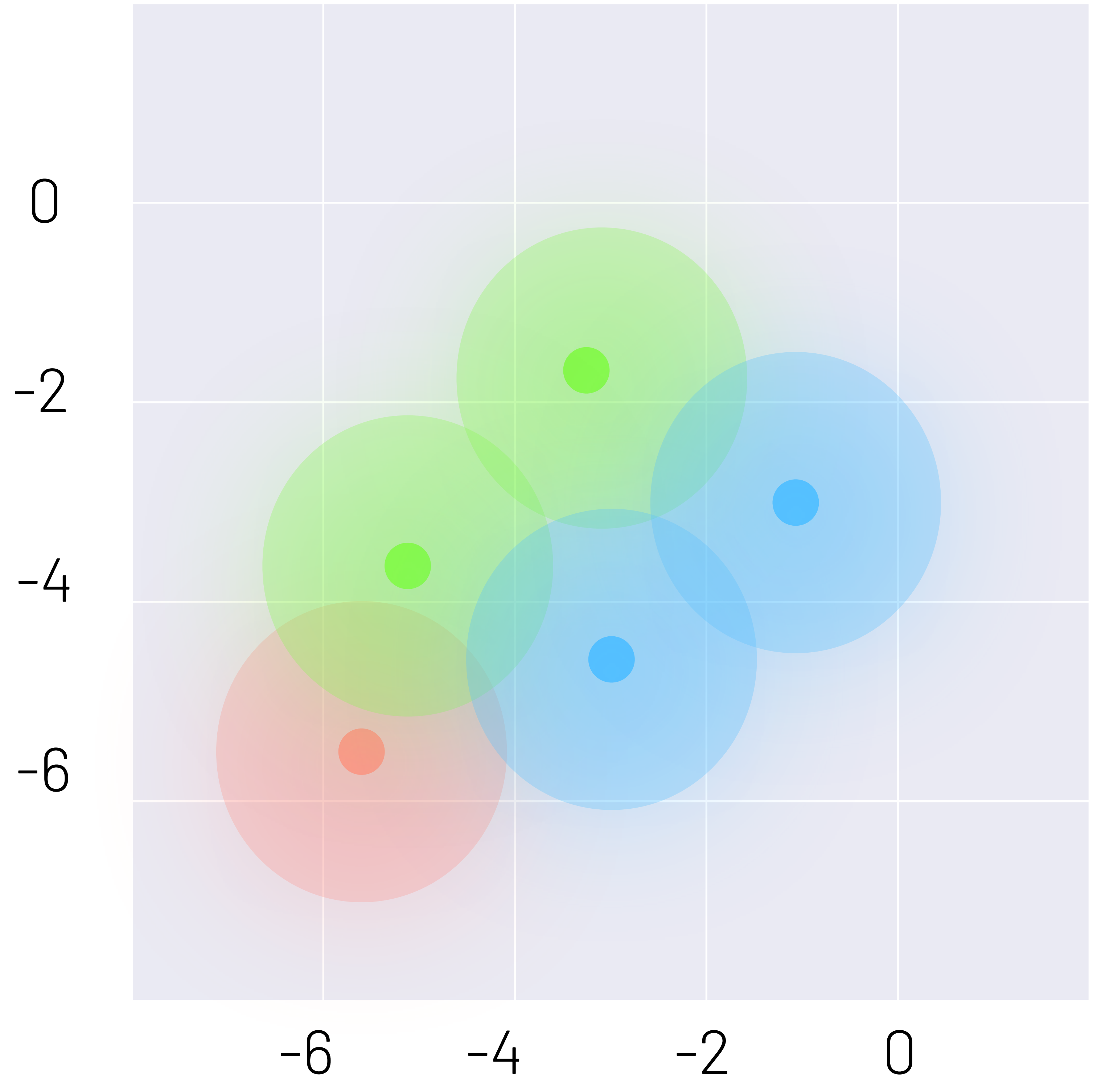
μ The mean vector controls the center of the encoding z

The std. dev. vector controls the area of the encoding z

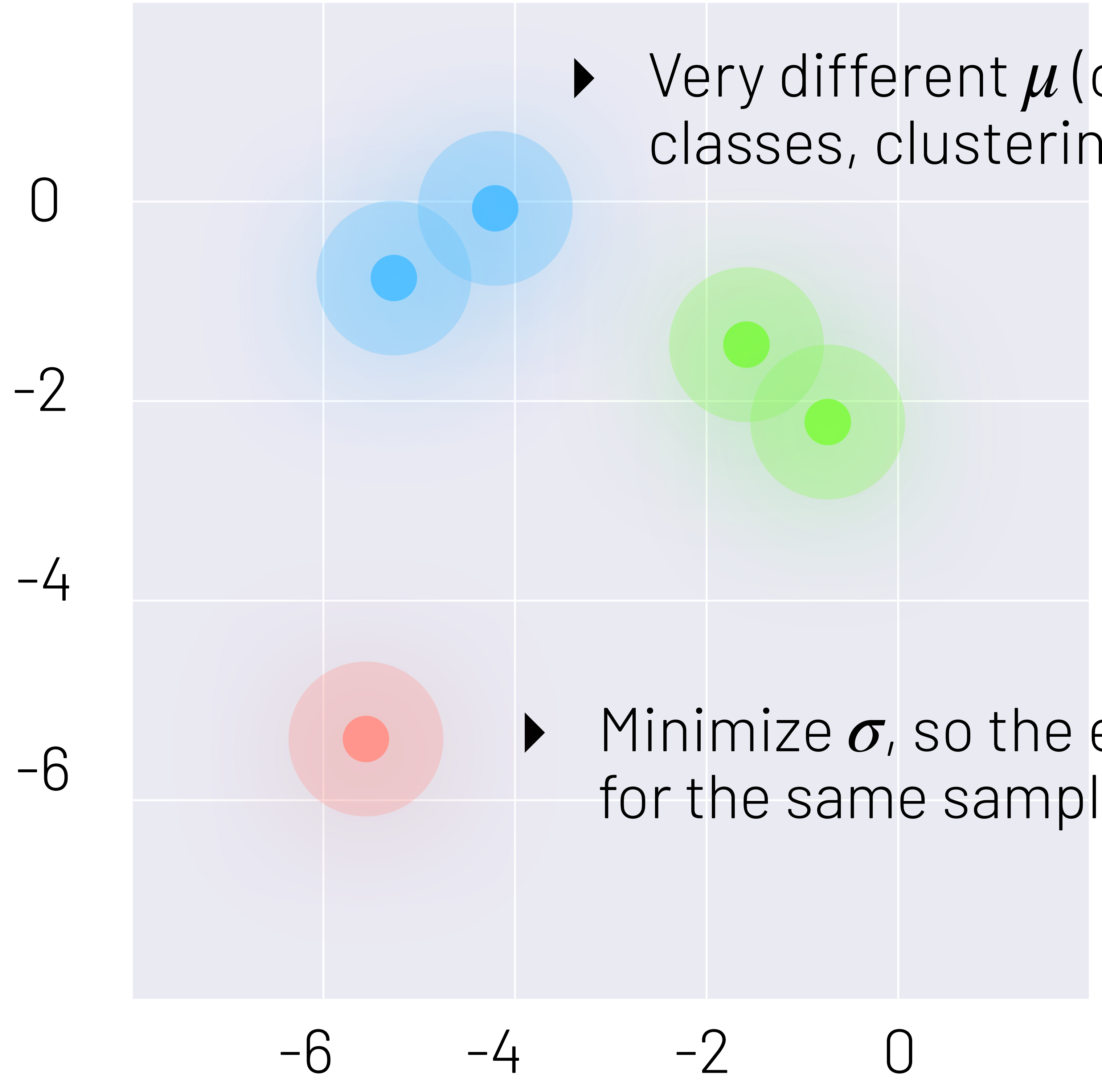
- All nearby points refer to the same sample of the class (continuity)

Ideal latent space

Ideally, we want overlap between samples that are not very similar too, in order to interpolate *between* classes.



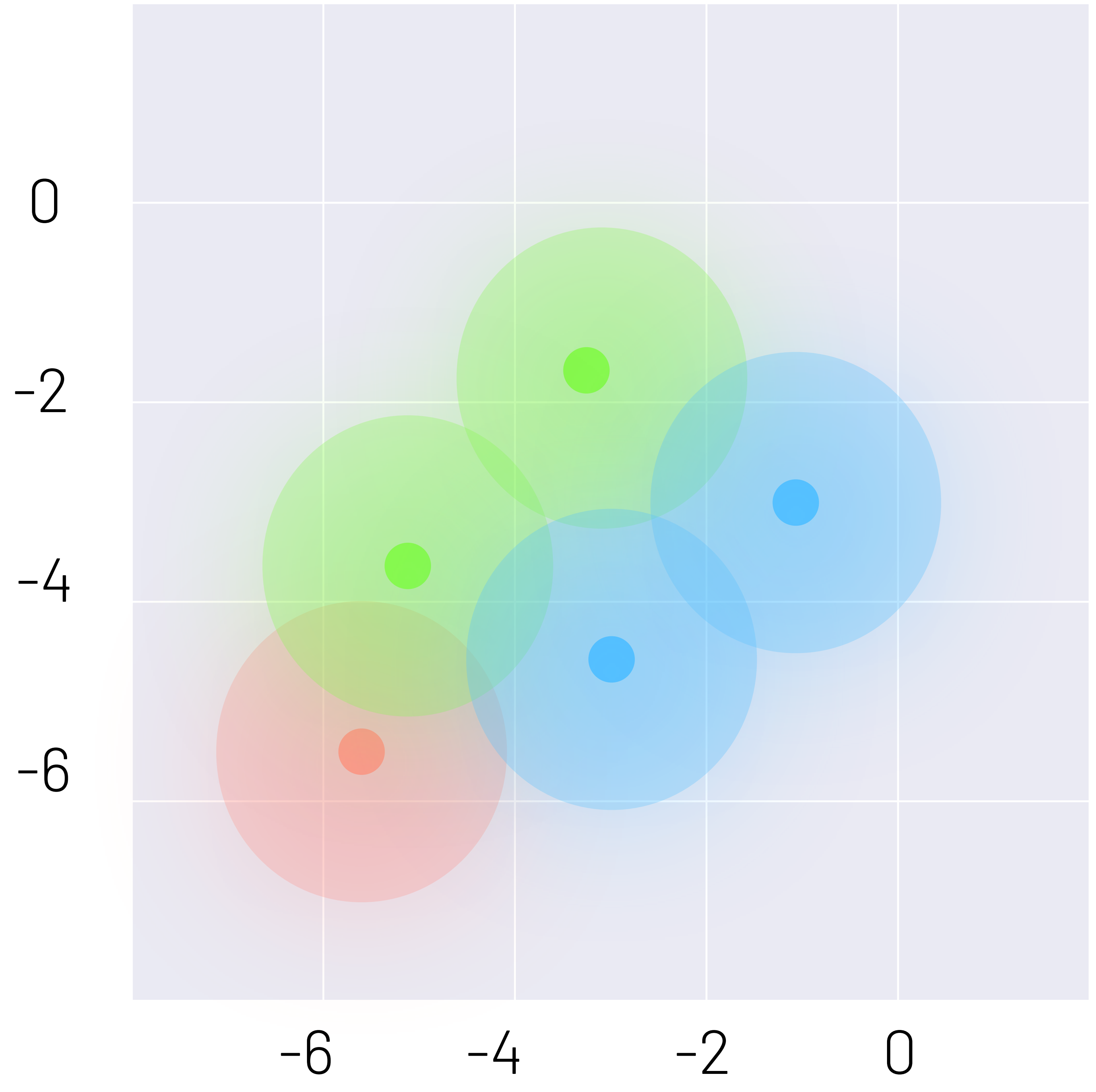
However, since there are *no limits* on what values vectors μ and σ can take on, the encoder can learn:



- Very different μ (centers) for different classes, clustering them apart
- Minimize σ, so the encodings don't vary much for the same sample

VAE Loss

To enforce overlap between samples, we add the **Kullback-Leibler divergence** (KL divergence) to the reconstruction loss function:



- Autoencoder Loss :

$$L(h) = MSE(x, \hat{x}) = \frac{1}{2wh} \sum_{i=1}^{h} \sum_{j=1}^{w} (x_{i,j} - \hat{x}_{i,j})^2$$

- VAE Loss:

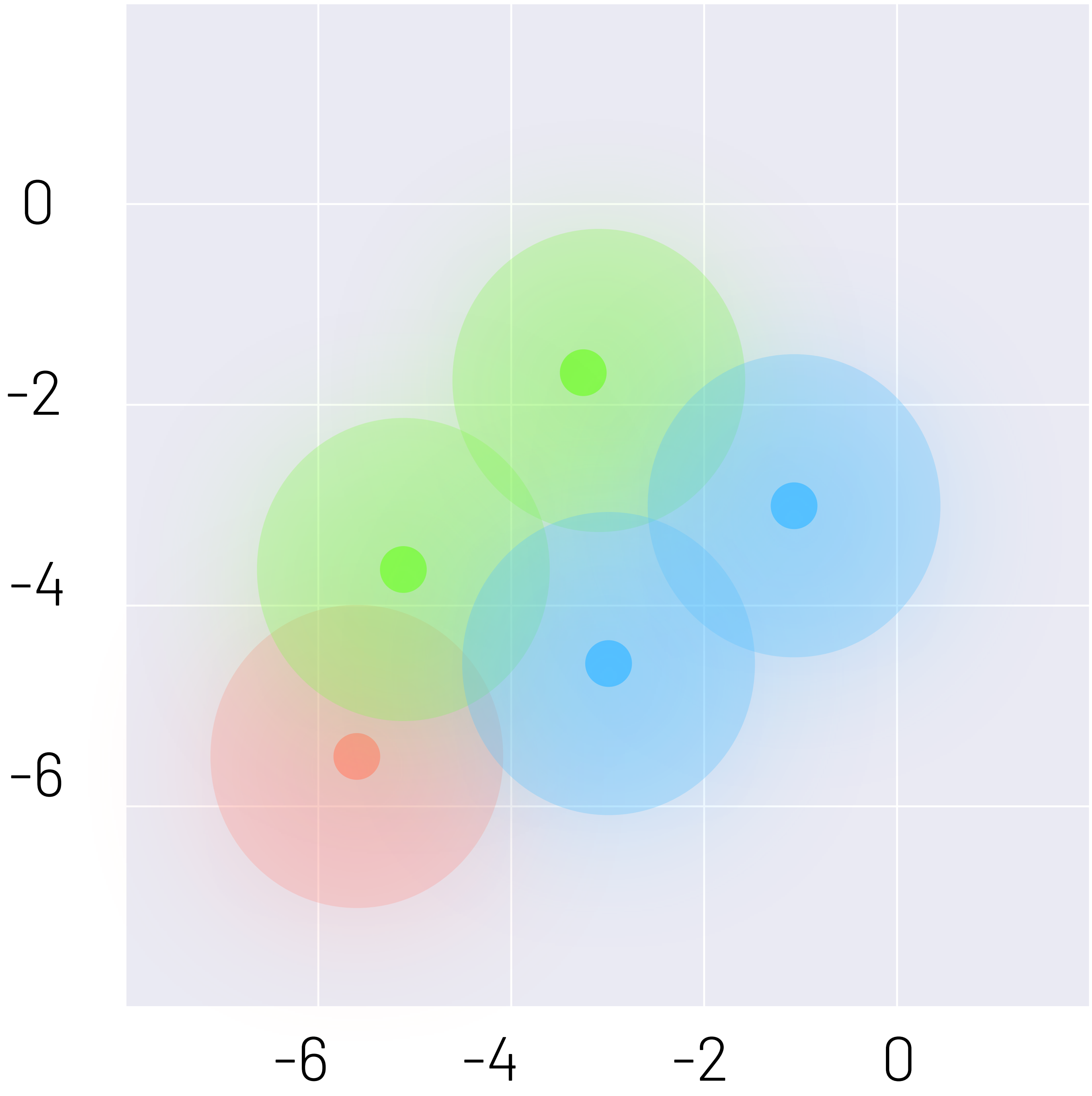
$$L(h) = MSE(x, \hat{x}) + \underline{KLD(N(\mu, \sigma), N(0,1))}$$

The KL Divergence is a "distance" metric to compare two distributions.

- We want to compare the learned distribution $N(\mu, \sigma)$ against a standard normal $N(0,1)$
- This loss encourages the encoder to distribute all encodings evenly around the center of the latent space.

VAE Loss

To enforce overlap between samples, we add the **Kullback-Leibler Divergence** (KL divergence) to the reconstruction loss function:



- Autoencoder Loss :

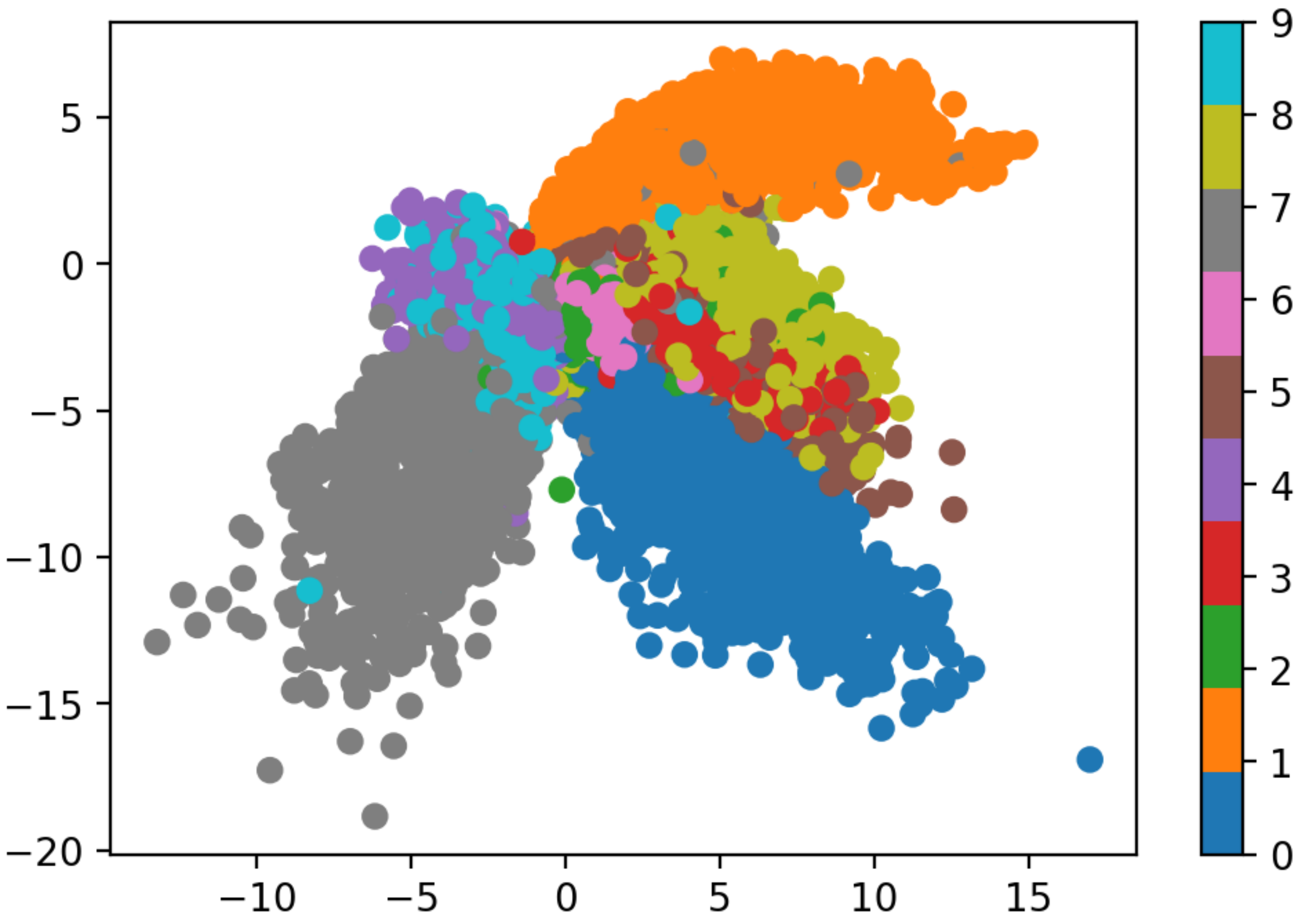
$$L(h) = MSE(x, \hat{x}) = \frac{1}{2wh} \sum_{i=1}^{h} \sum_{j=1}^{w} (x_{i,j} - \hat{x}_{i,j})^2$$

- VAE Loss:

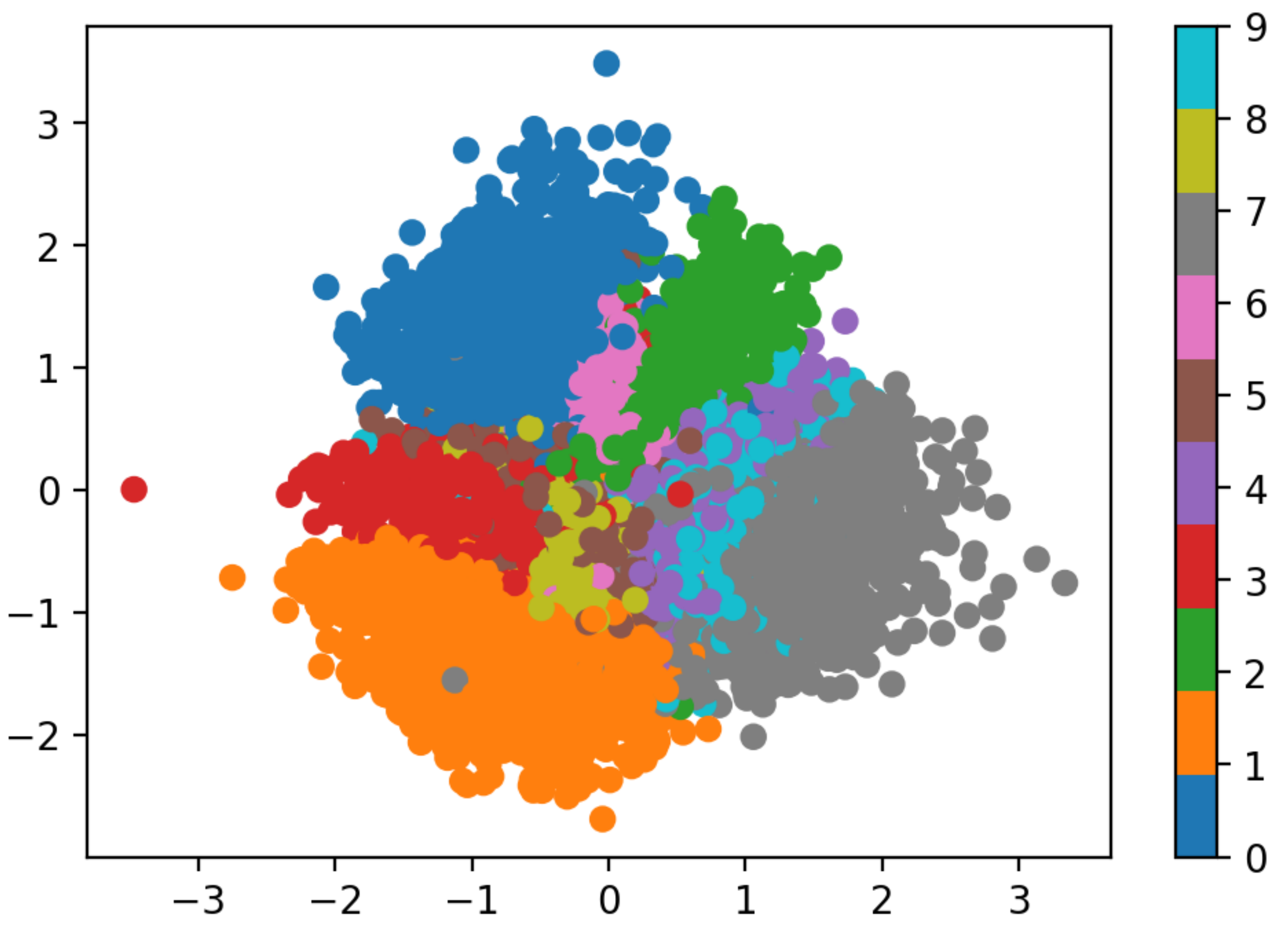
$$L(h) = MSE(x, \hat{x}) + KLD(N(\mu, \sigma), N(0,1))$$

$$KLD(N(\mu, \sigma), N(0,1)) = \sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1$$

Visualizing the learned latent space



Autoencoders

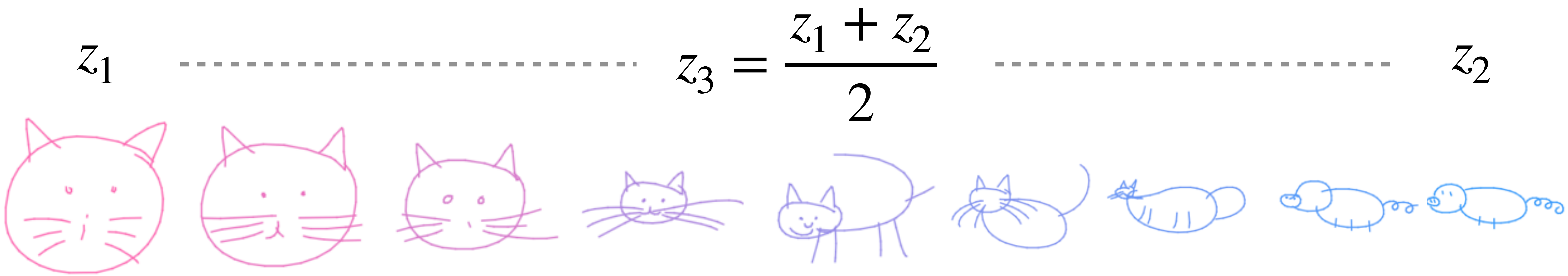


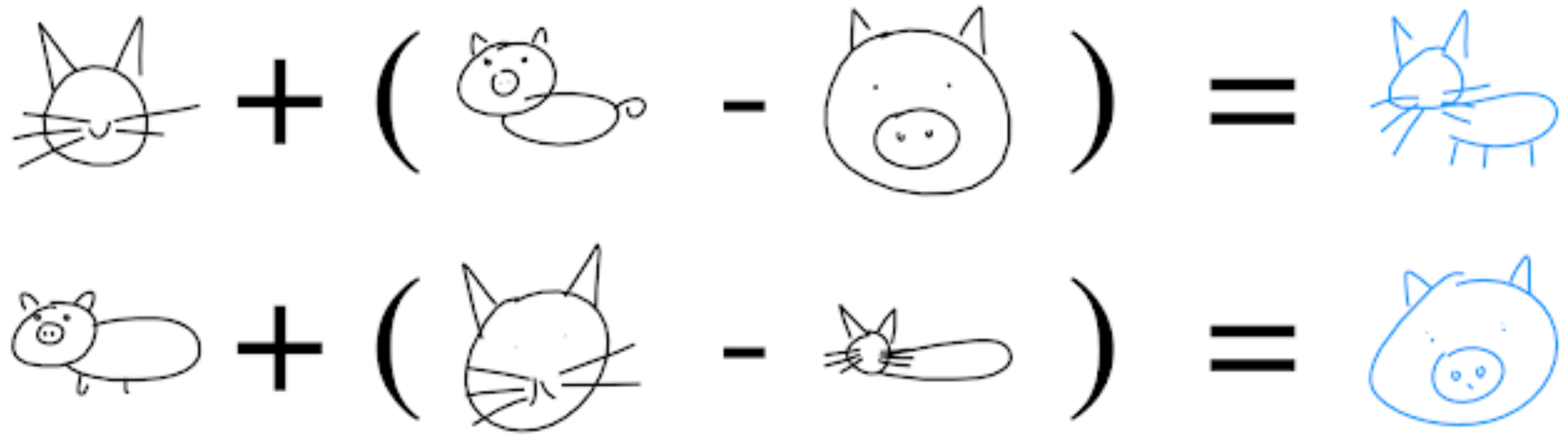
VAEs

- The VAE latent space maintains the similarity of nearby encodings on the *local scale* via clustering
- Yet *globally*, is very densely packed near the latent space origin

Interpolating samples in the latent space

Since the space is continuous and smoothly transitions samples classes, we can interpolate different latent vectors z_1 and z_2 and get a semantically meagninful result:





Ha, D. and Eck, D. 2017, A Neural Representation of Sketch Drawings

Next Lecture

L21: GANs

Generating images with Generative Adversarial Networks (GANs)