

# Tutorial 6: AE, VAE, cVAE

---



**Practical Deep Learning for Science**  
**30 May, 2023**

# Quick side-note: KLD, CE, BCE

---

♦ Point-wise loss

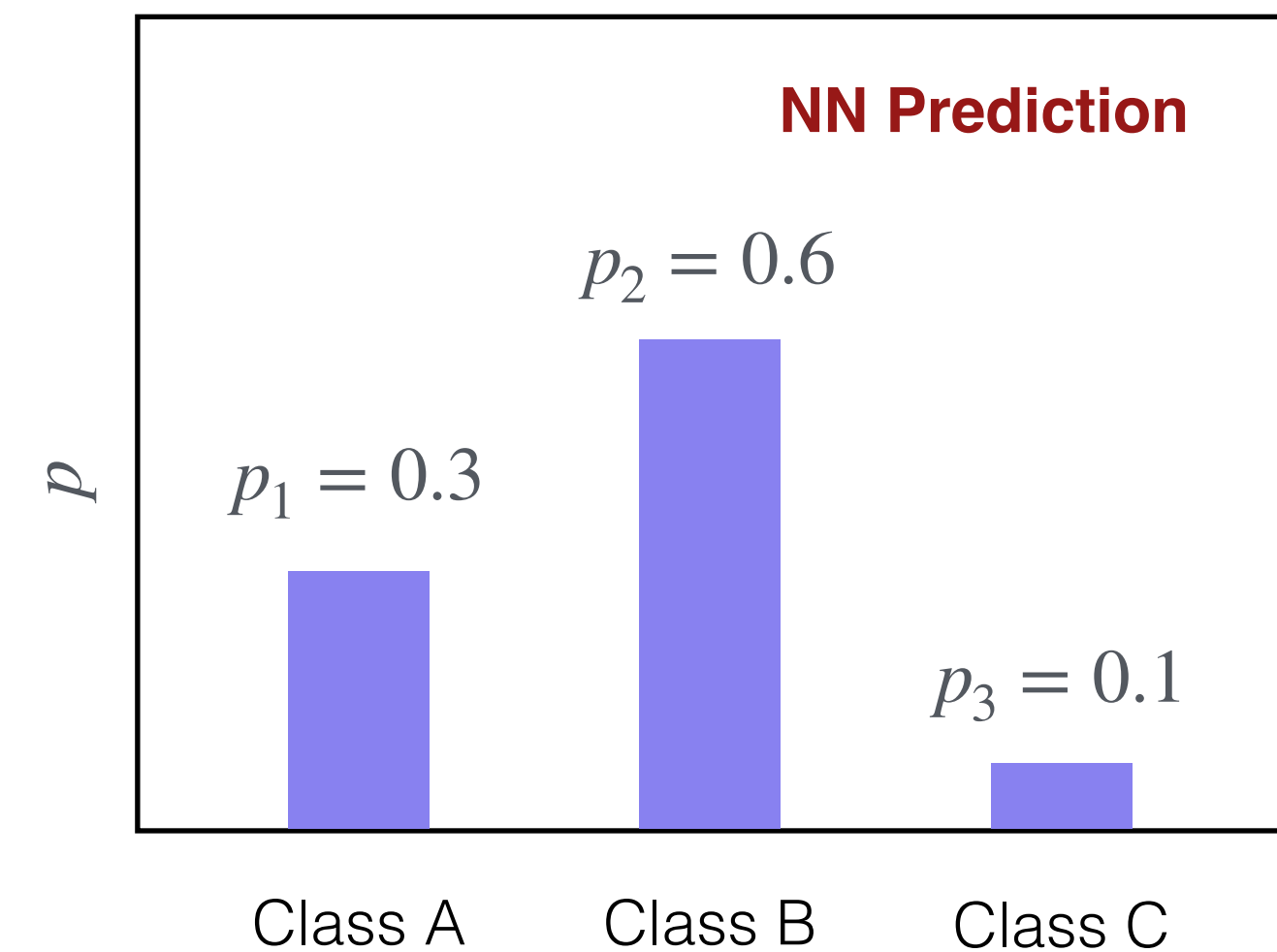
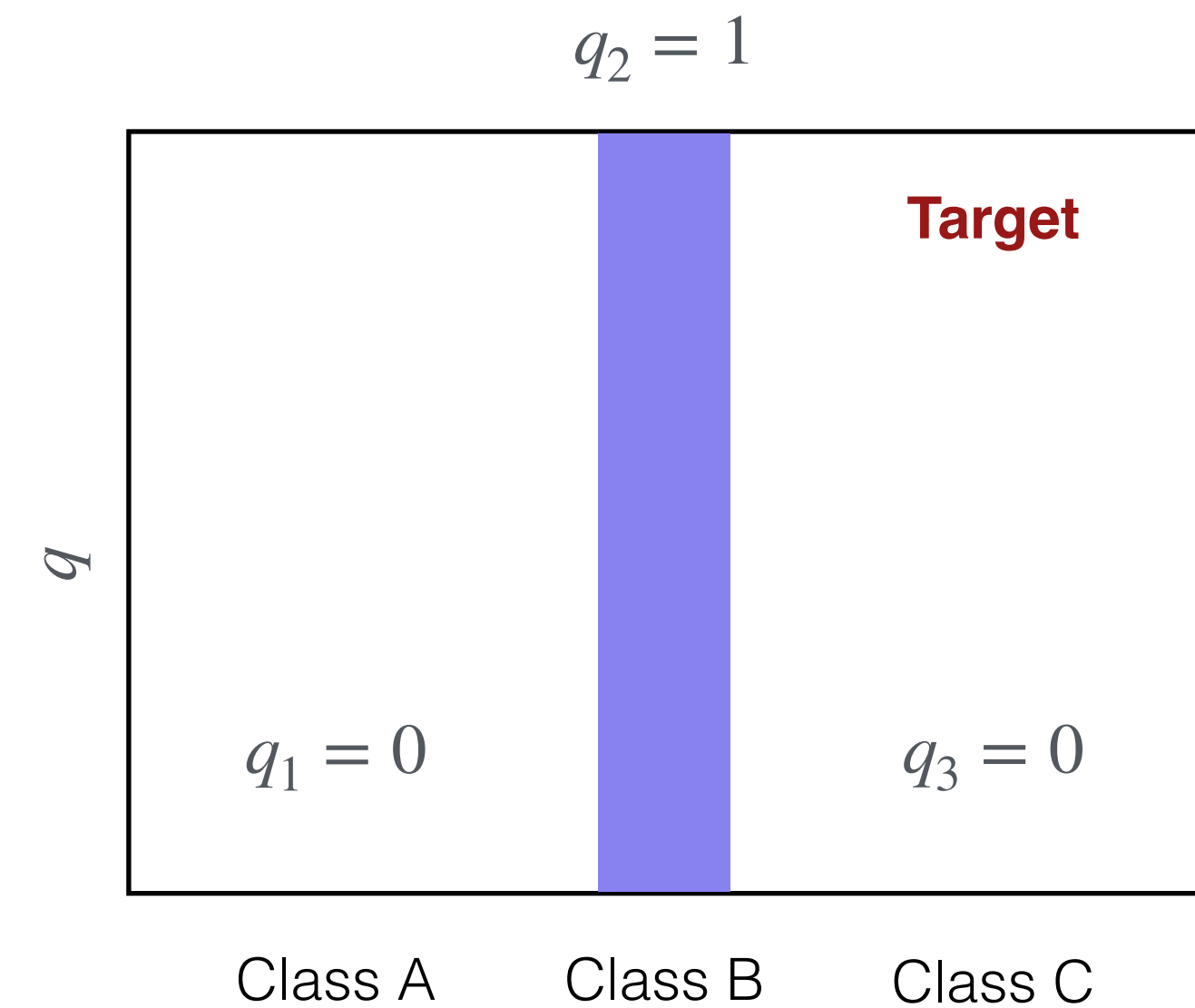
♦ KLDivergence

$$\rightarrow \sum_i q_i \log \left( \frac{q_i}{p_i} \right)$$

$$\rightarrow \underbrace{q_i \log(q_i)}_{\text{Const}} - \underbrace{q_i \log(p_i)}_{\text{Cross Entropy}}$$

♦ Gradient (KLD) = Gradient (Cross Entropy)

➔ For training, KLD and CE are (mostly) same



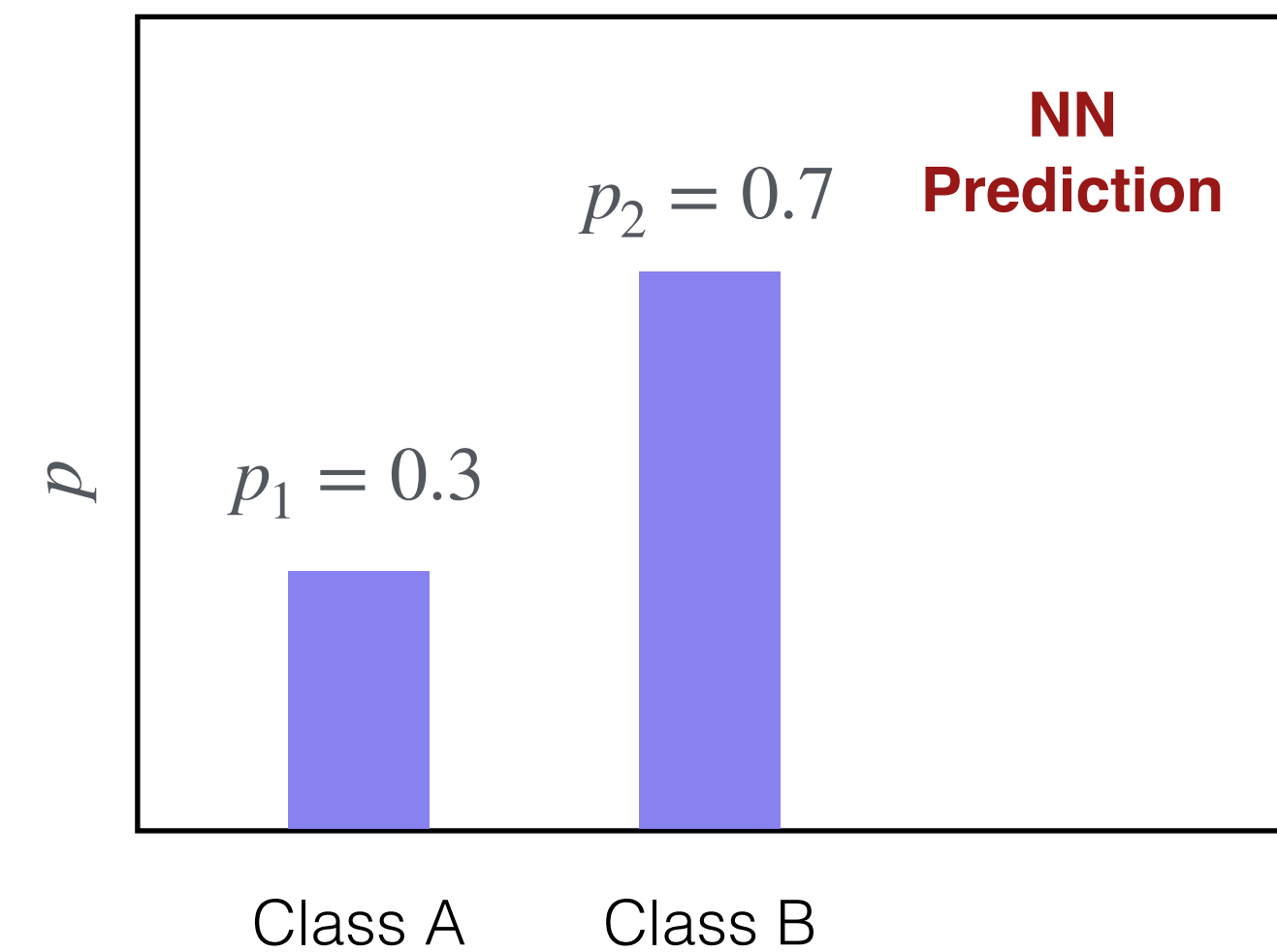
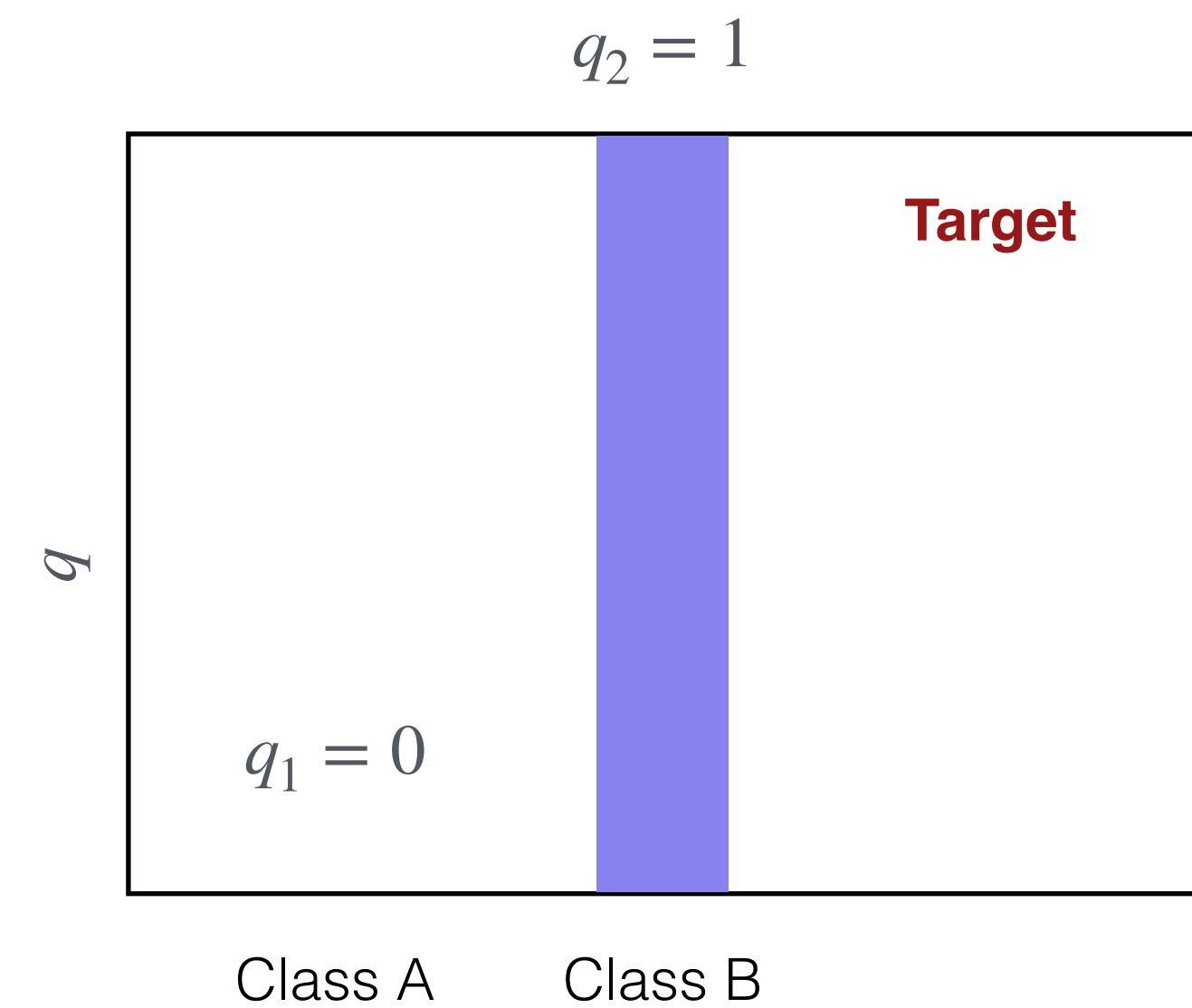
♦ If we have two classes

♦ Cross Entropy

$$\rightarrow -q_1 \log(p_1) - q_2 \log(p_2)$$

$$\rightarrow -q_1 \log(p_1) - q_2 \log(1 - p_1)$$

→ **Binary Cross Entropy**



# In torch...

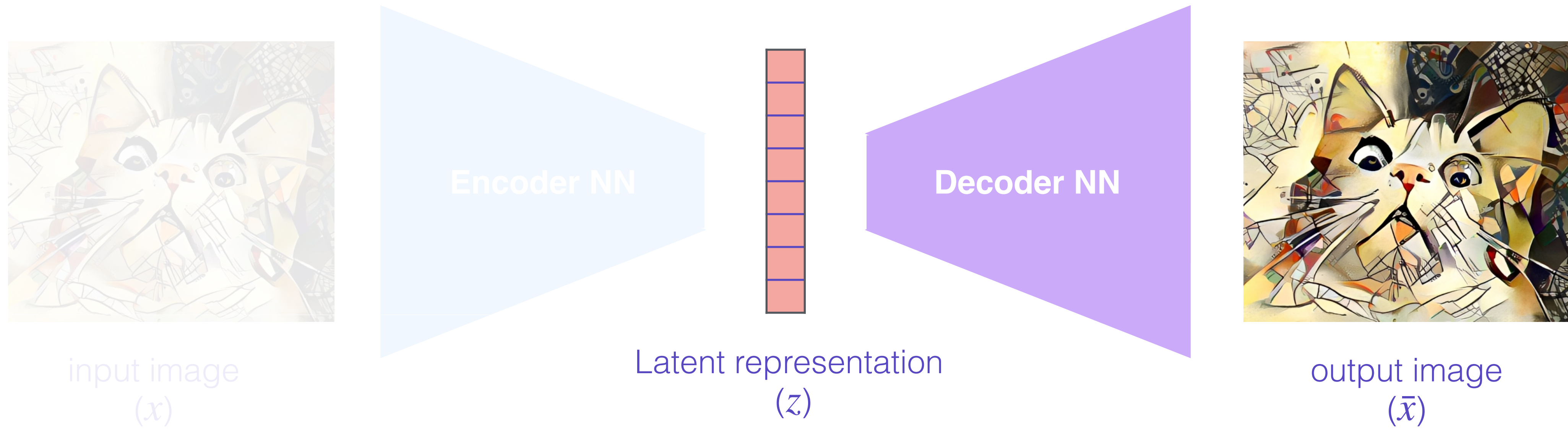
- ✦ We need probabilities to compute loss
- ✦ For binary classification,
  - ➔ `NN(... Linear, Sigmoid) → BCELoss`
  - ➔ Or `NN(... Linear) → BCEWithLogitsLoss` (recommended)
  - ➔ `BCEWithLogitsLoss = Sigmoid + BCELoss`
- ✦ For More than 2 classes,
  - ➔ `NN(... Linear, LogSoftmax) → NLLLoss`
  - ➔ Or `NN(... Linear) → CrossEntropyLoss` (recommended)

# AutoEncoder (AE)

---



## During generation



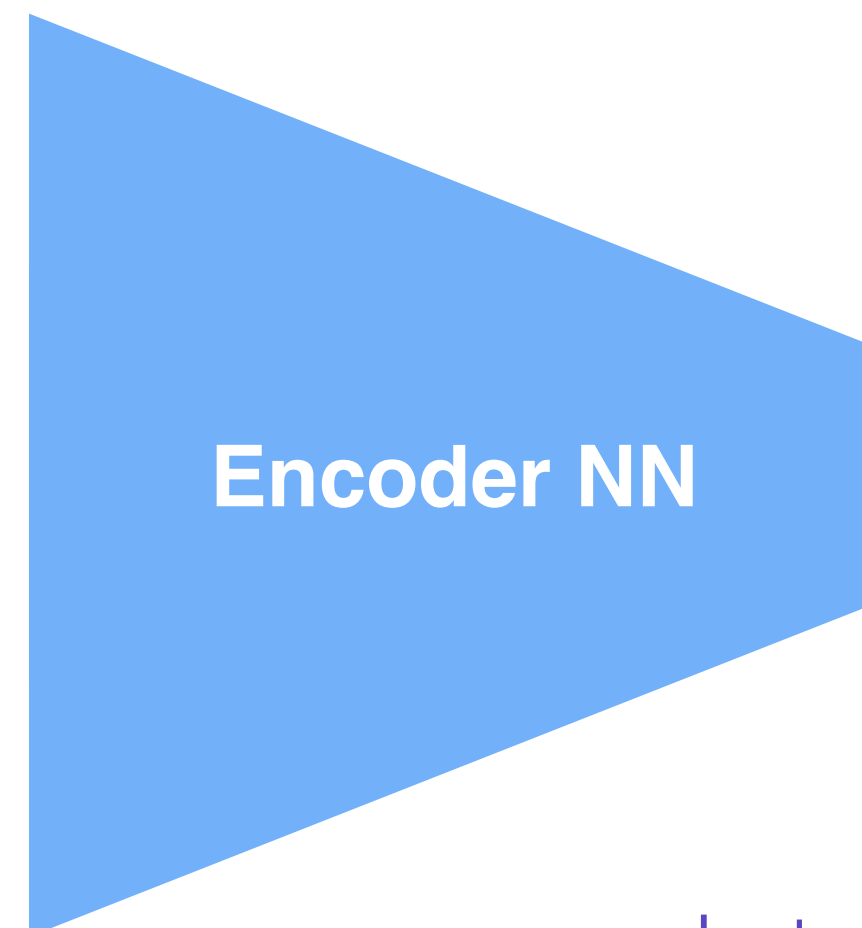
# Variational AutoEncoder (VAE)

---

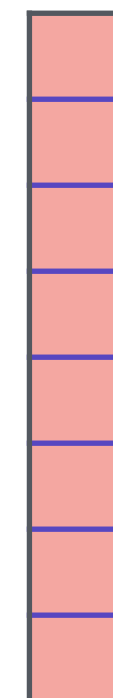




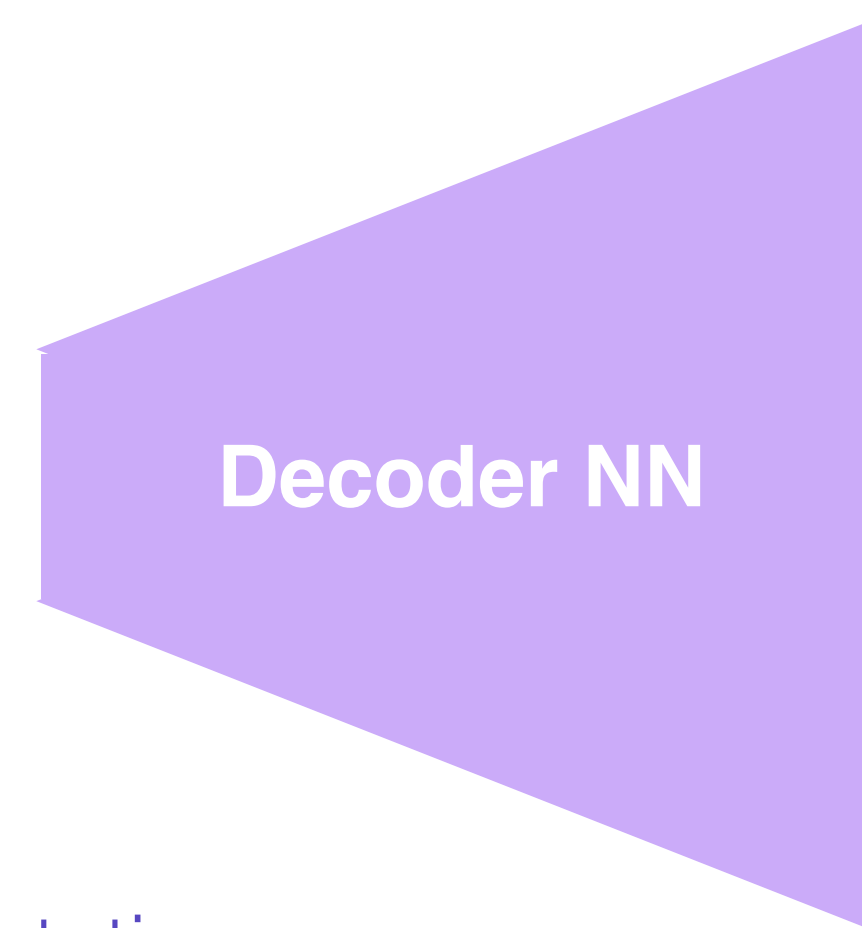
input image  
( $x$ )



Encoder NN



Latent representation  
( $z$ )



Decoder NN



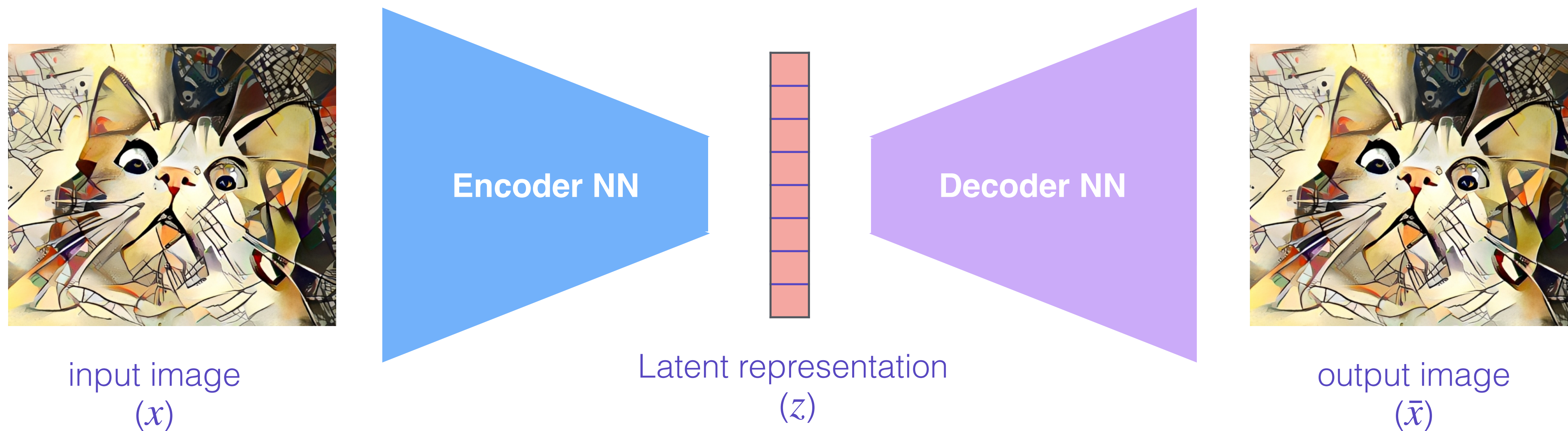
output image  
( $\bar{x}$ )

Want this to be Normally distributed

$$\text{Loss}(x, \bar{x}) + \text{KLD}(z, \text{Gauss})$$

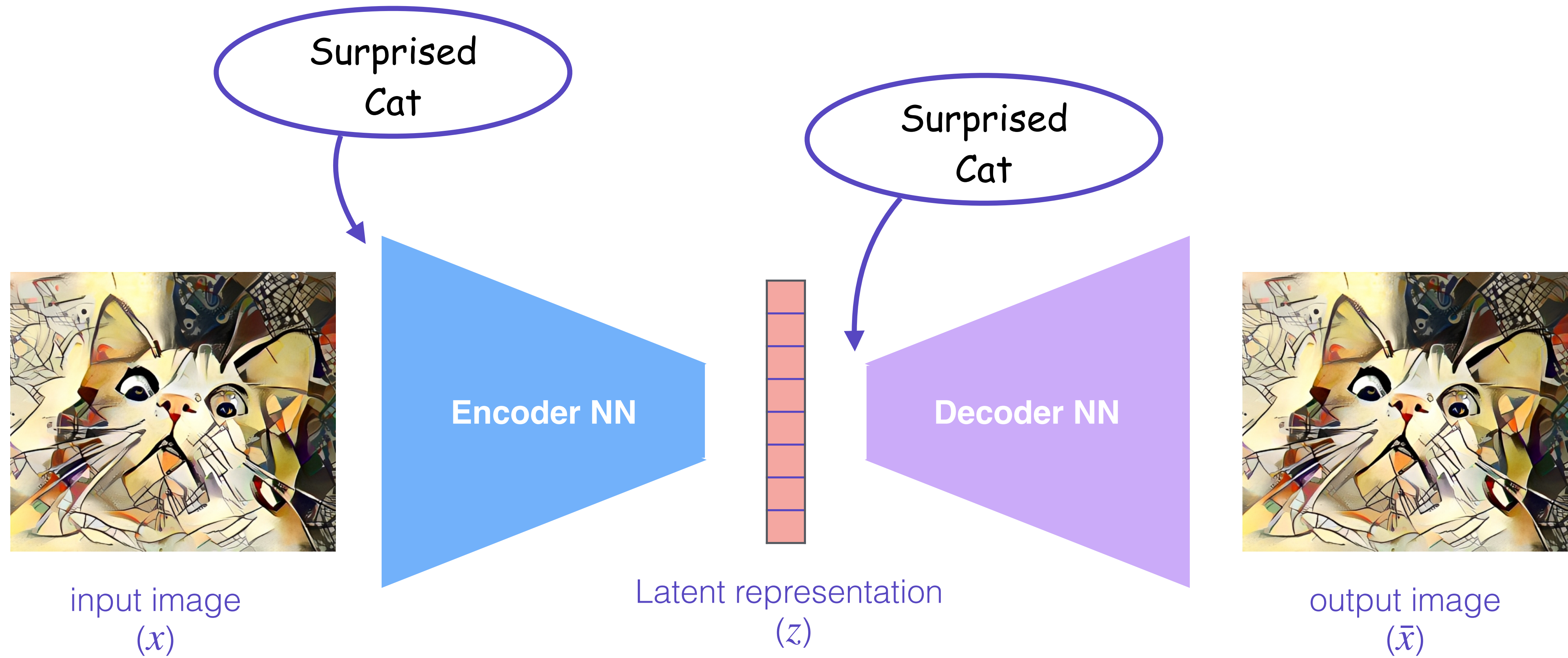
# Conditional Variational AutoEncoder (cVAE)

---



- Right now, we have no control over the generated images during inference
  - In the MNIST example, it generates random numbers
  - But let's say we want to generate specific numbers
  - Generation needs to be **conditioned on** what we want → **conditional VAE**





- We need to pass the conditional info as input to both Encoder and Decoder

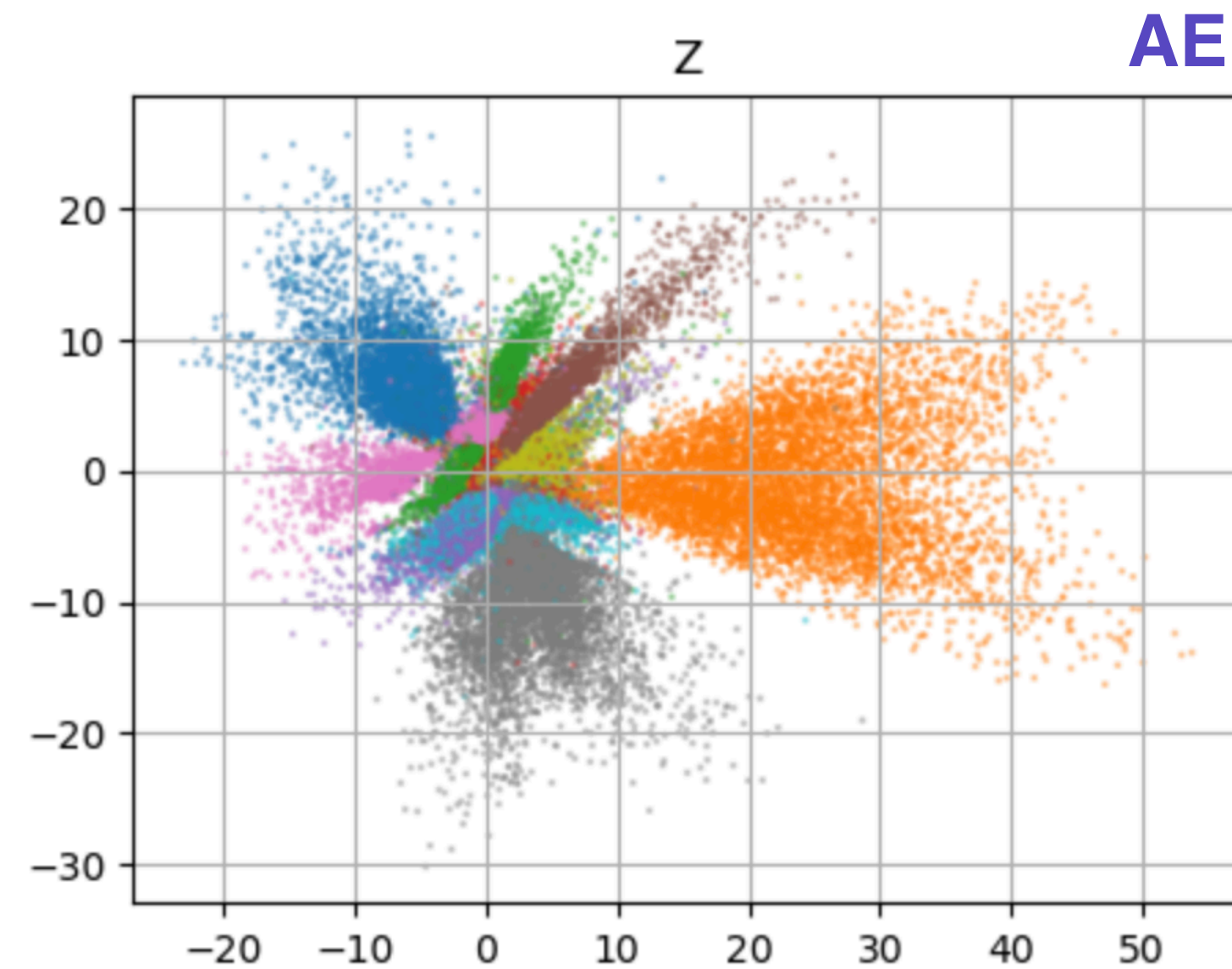
# Encoding the conditional info

- ✦ One hot encoded (few classes)
  - ➔  $3 \rightarrow (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$
  - ➔  $9 \rightarrow (0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$
  - ➔ ...
- ✦ Encoding class labels into a vector space,
  - ➔ `Torch.nn.Embedding()`
  - ➔ Helpful when we have a large number of classes
- ✦ Other Fancy encoding in text to image models (Mldjourney, DALLE etc)
  - ➔ Maybe later in the course (no promises!)

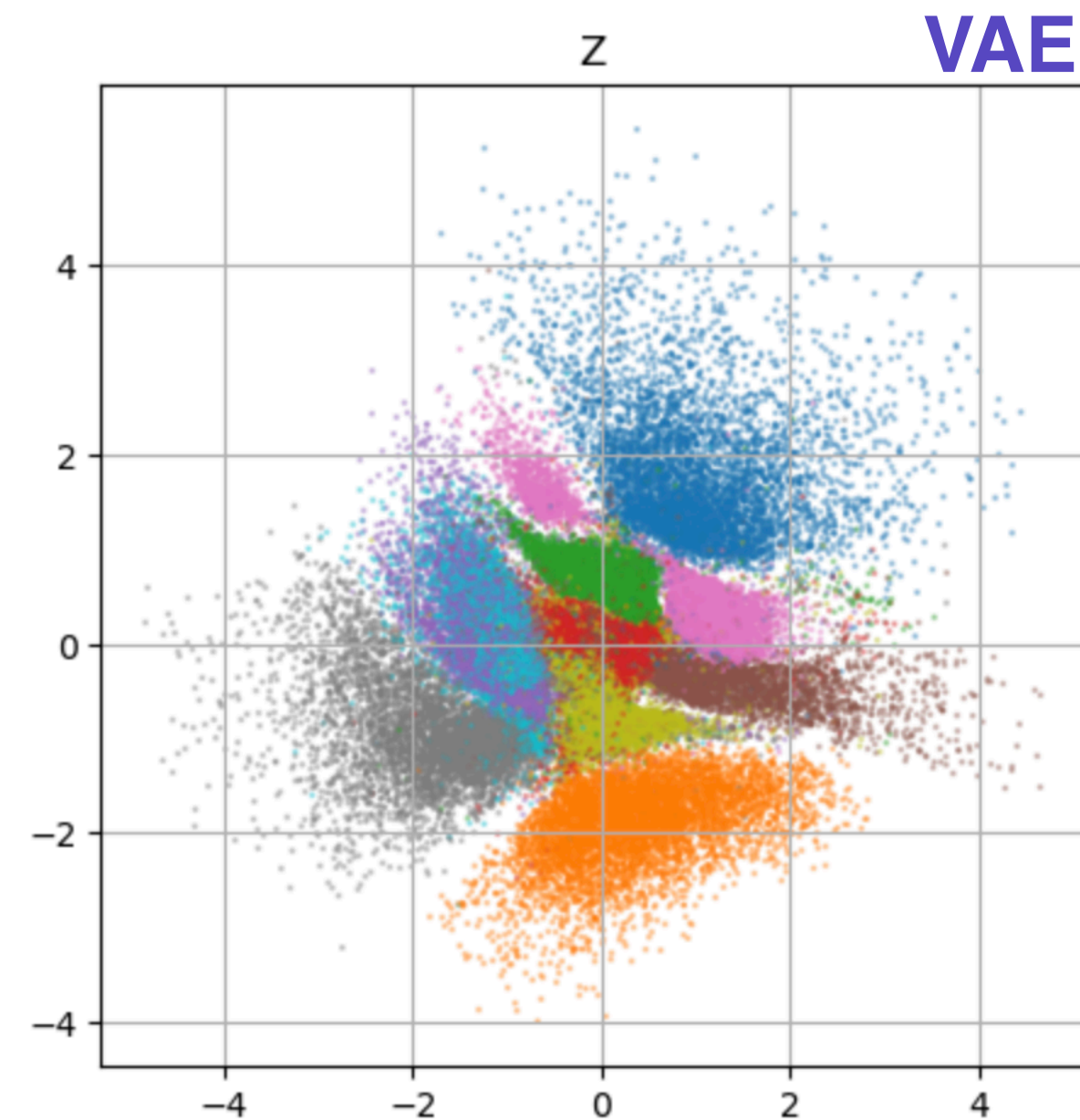
# The latent space distributions

---

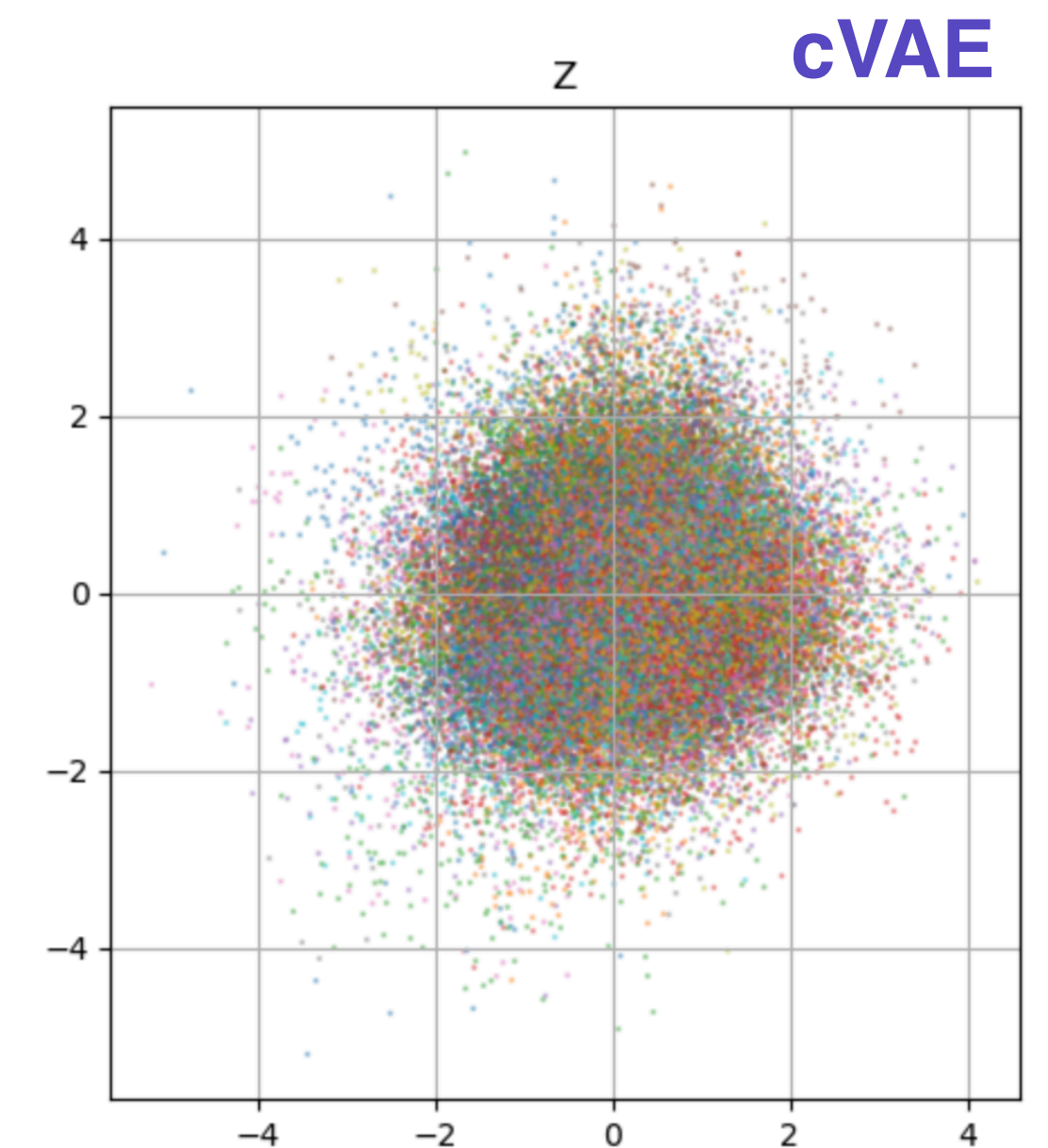




- ◆ No KLD  $\rightarrow$  not Gaussian at all
- ◆ Empty spaces in between
  - $\rightarrow$  Sampling is tricky



- ◆ KLD  $\rightarrow$  Gaussian like
- ◆ Less empty spaces in between
  - $\rightarrow$  Easy to sample
- ◆ Needs to keep the classes separate
  - $\rightarrow$  A 7 must be reconstructed as a 7



- ◆ KLD + conditional info  $\rightarrow$  almost perfect Gaussian
- ◆ Doesn't need to keep the classes separate
  - $\rightarrow$  Already knows which class to reconstruct from the conditional info