Tutorial 6:
AE, VAE, cVAE

Practical Deep Learning for Science
30 May, 2023

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

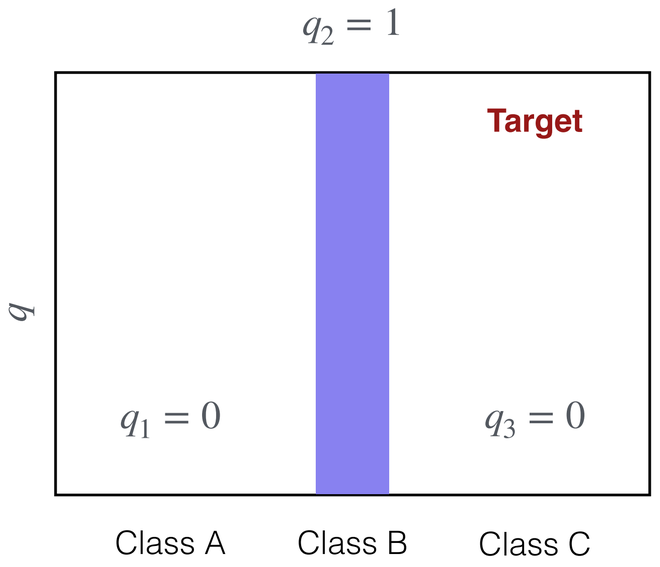
- Nilotpal

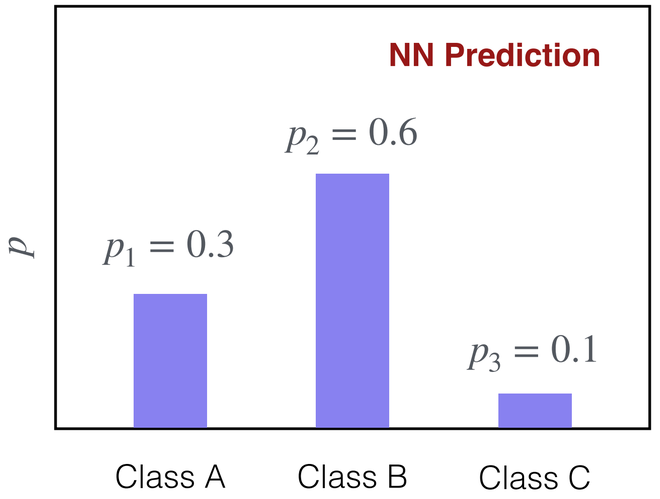
Quick side-note:
KLD, CE, BCE in classification

- Point-wise loss

- KLDivergence
  - $\sum_i q_i \log\left(\frac{q_i}{p_i}\right)$
  - $q_i \log(q_i)$ $\boxed{-\ q_i \log(p_i)}$
    Const **Cross Entropy**

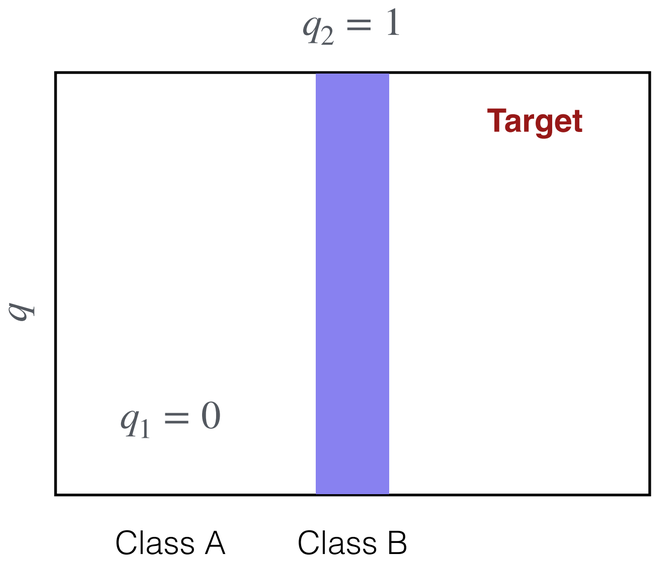
- Gradient (KLD) = Gradient (Cross Entropy)
  - For training, KLD and CE are (mostly) same

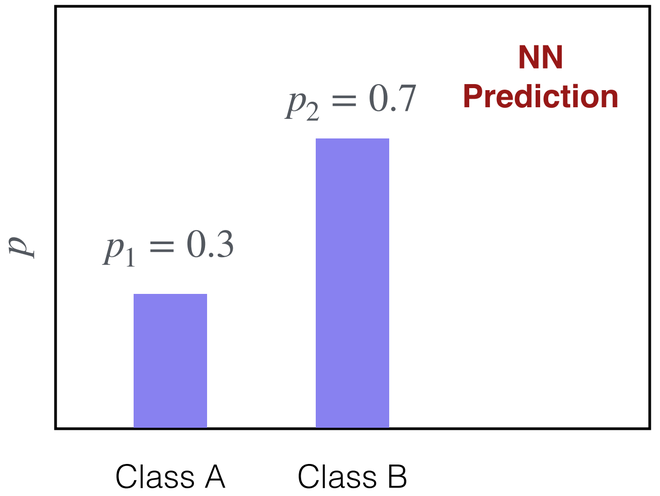




- If we have two classes

- Cross Entropy
  - $-q_1 \log(p_1) - q_2 \log(p_2)$
  - $-q_1 \log(p_1) - q_2 \log(1 - p_1)$
  - **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

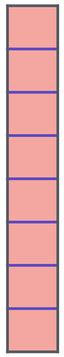
- For More than 2 classes,
  - `NN(…. Linear, LogSoftmax) → NLLLoss`
  - Or `NN(… Linear) → CrossEntropyLoss` (recommended)

AutoEncoder (AE)

During generation

input image
(x)

Encoder NN



Latent representation
(z)

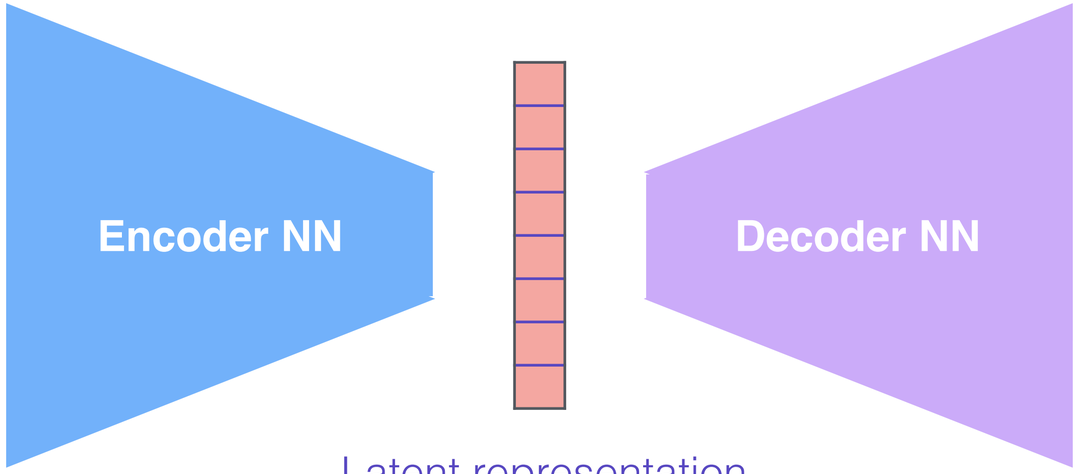
Decoder NN

output image
$(\bar{x})$

Loss $(x, \bar{x})$

Variational AutoEncoder (VAE)

input image
(x)



Latent representation
(z)

output image
$(\bar{x})$

Want this to be Normally distributed

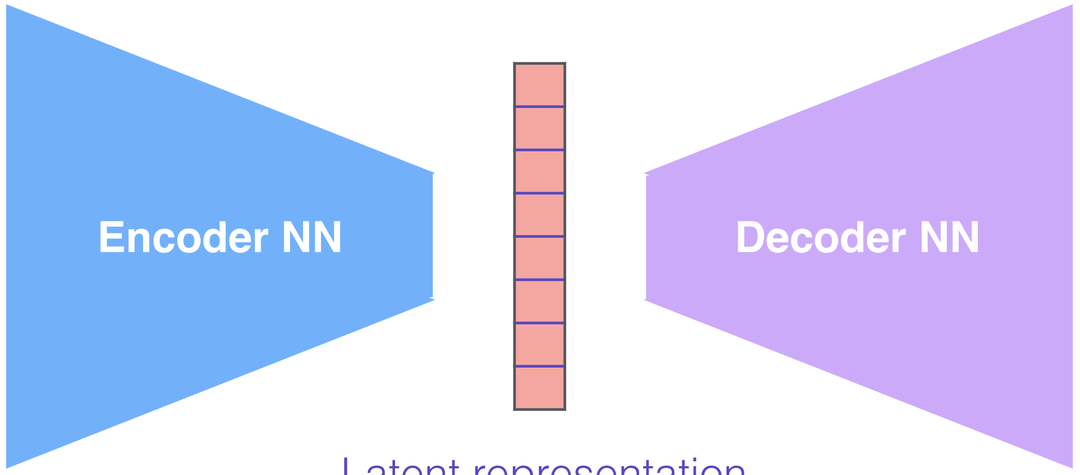
$$\text{Loss}\,(x, \bar{x}) + \text{KLD}(z, \text{Gauss})$$

Not pointwise KLD

- Target distribution $P = \mathcal{N}(0, I)$
- Predicted distribution $Q = \mathcal{N}(\mu, \Sigma)$
- KLD b/w the two
  - $KLD(P||Q)$
  - `kld_loss = -0.5 * sum(1 + log_var - mu^2 - exp(log_var))`
  - Derivation - https://stats.stackexchange.com/questions/318748/deriving-the-kl-divergence-loss-for-vaes/370048#370048

Conditional Variational AutoEncoder (cVAE)

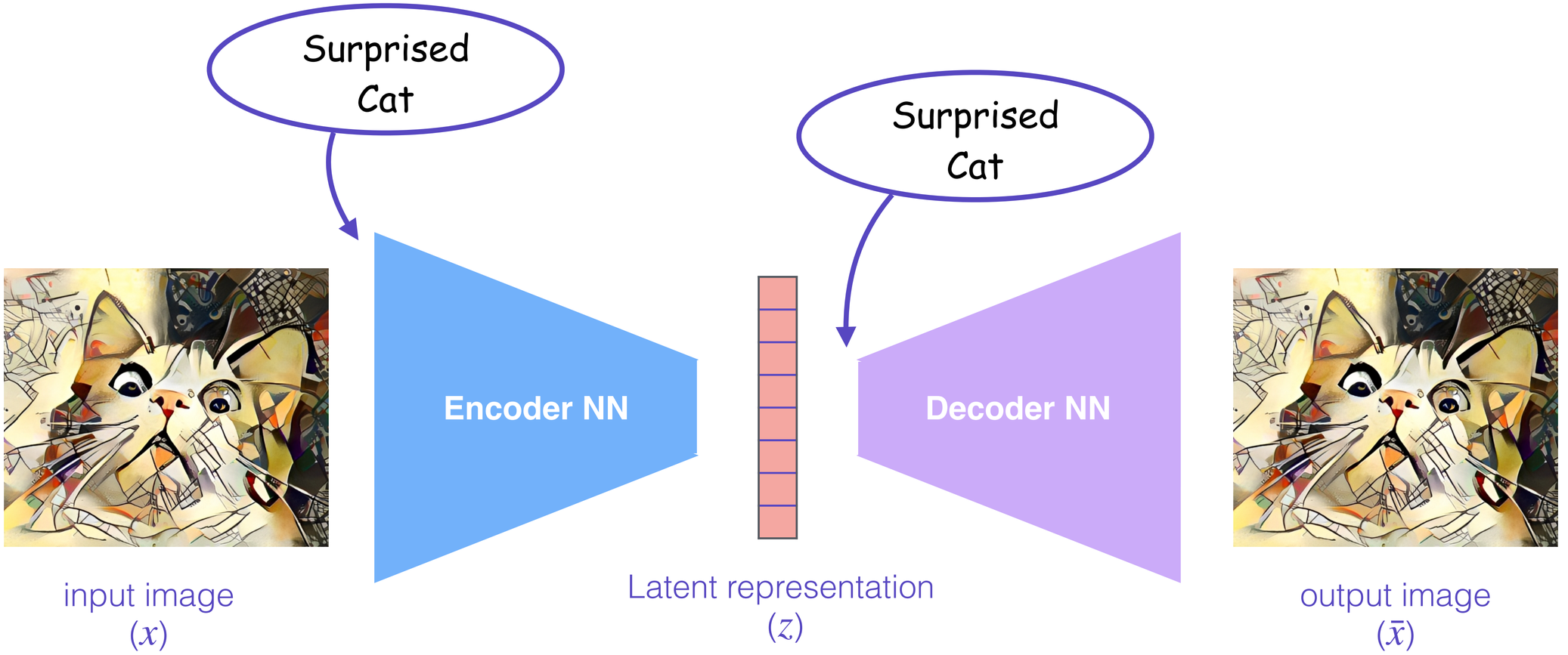
input image
(x)



Latent representation
(z)

output image
$(\bar{x})$

- 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

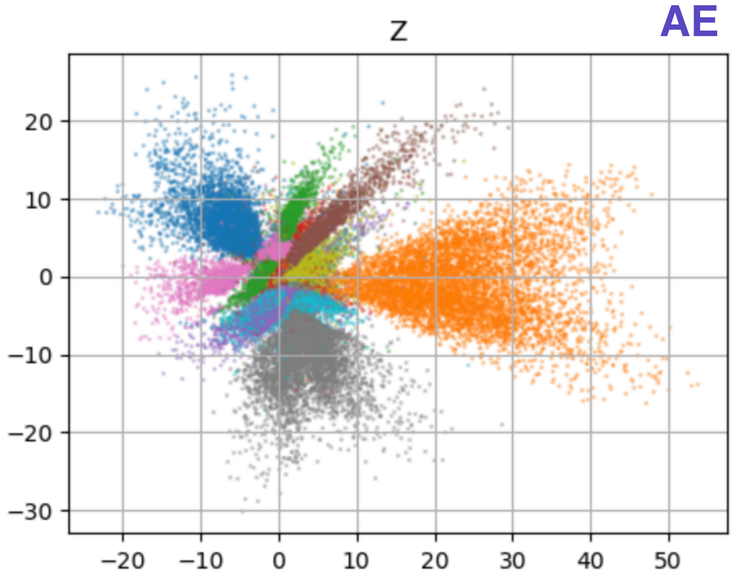
Encoding the conditional info

- One hot encoded (few classes)
  - 3 → (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
  - 9 → (0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
  - …

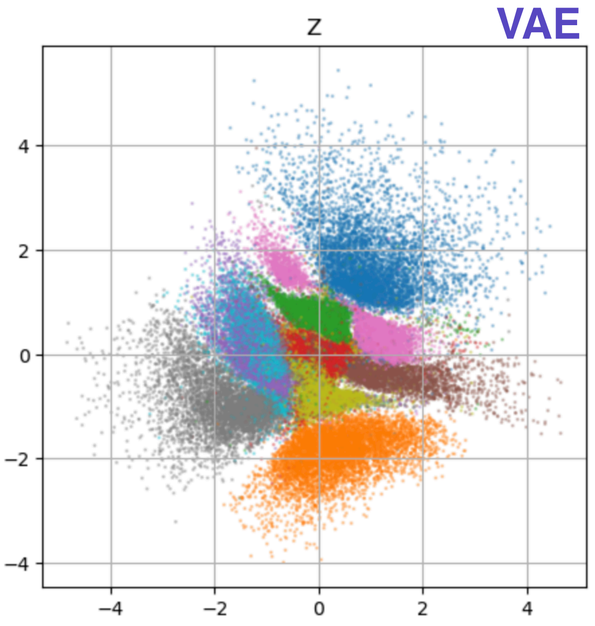
- Encoding class labels into a vector space,
  - `Torch.nn.Embedding()`
  - Helpful when we have a large number of classes

- Other Fancy encoding in text to image models (MIdjourney, DALLE etc)
  - Maybe later in the course (no promises!)

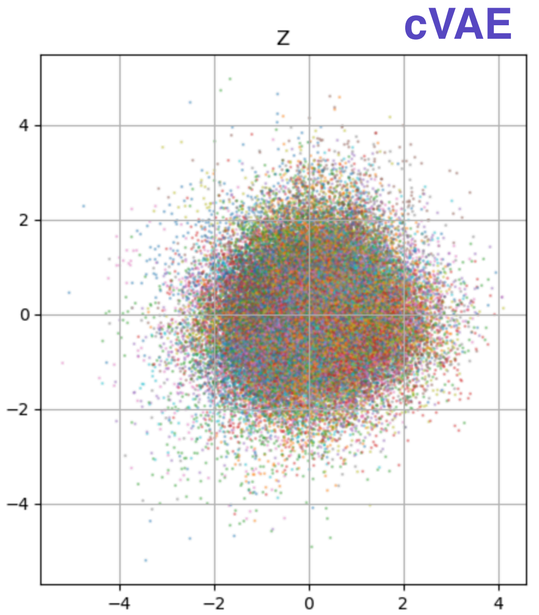
The latent space distributions



- No KLD → not Gaussian at all
- Empty spaces in between
  - ➡ Sampling is tricky



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



- KLD + conditional info → almost perfect Gaussian
- Doesn't need to keep the classes separate
  - ➡ Already knows which class to reconstruct from the conditional info