Bottleneck

Data $x \in \mathbb{R}^D$

Encoder: $z = f_\theta(x)$: $z \in \mathbb{R}^d$, $d << D$

Decoder: $\hat{x} = g_\phi(z)$

Training: $\min_{\theta,\phi} \mathbb{E}_x[\mathcal{L}_{rec}(x, \hat{x})]$

$z_e \in \mathbb{R}^d$ needs accuracy → + Bits ∝ + computational cost.

Solution: Codebook $E = \{e_1, \dots, e_K\}$: $e_K \in \mathbb{R}^d$

Improvement!

Instead of giving z_e to $g_\phi(\cdot)$, we quantize it.

$K^*(z) = \arg\min_K \|z - e_K\|^2$ (Nearest neighbor) ; $z_q = e_K$

Stores an index, not a specific value. (~ $\log_2 K$ bits per simbol)

New Problem: The size K of the codebook

- Small K: A few bits, but high error $\|z_e - z_q\|$ → worse reconstruction.
- Big K: Better accuracy $\|z_e - z_q\| \to 0$, but gigantic codebook → + computational cost, + Bits.

New Solution: Make a codebook family $\{E^{(m)}\}_{m=1}^{M}$

$E^{(m)} : \{e_1^{(m)}, \dots, e_K^{(m)}\} \in \mathbb{R}^d$; $m = 1, \dots, M$ → Quantization Levels.

Now, the quantized latent:

$$z_q = \sum_{m=1}^{M} q^{(m)} \; ; \; q^{(m)} \in E^{(m)}$$

The "compressed" x is not a single index, now it is a tuple:

$(K_1, K_2, \dots, K_M)$: $K_m \in \{1, \dots, K_m\}$ → Not necesary every codebook has the same lenght.

And its computational cost is:

$$R \approx \sum_{m=1}^{M} \log_2 K_m$$

$$(K_1^*, \dots, K_M^*) = \arg\min_{K_1, \dots, K_M} \Big\| \underbrace{z -}_{\text{Residual}} \sum_{m=1}^{M} e_{K_m}^{(m)} \Big\|^2$$

Nearest Neighbor for every codebook.

Quantized