Learning to Reconstruct and Generate Images

ECE 57000 — November 4, 2026

David I. Inouye

A model's objective and its use need separate evidence

- Generation feeds selected tokens back; decoding choices change outputs without retraining.
- Serving and training have different memory costs; parameter count alone predicts neither.
- Our short-email comparison supported both learned models, without a decisive architecture winner.

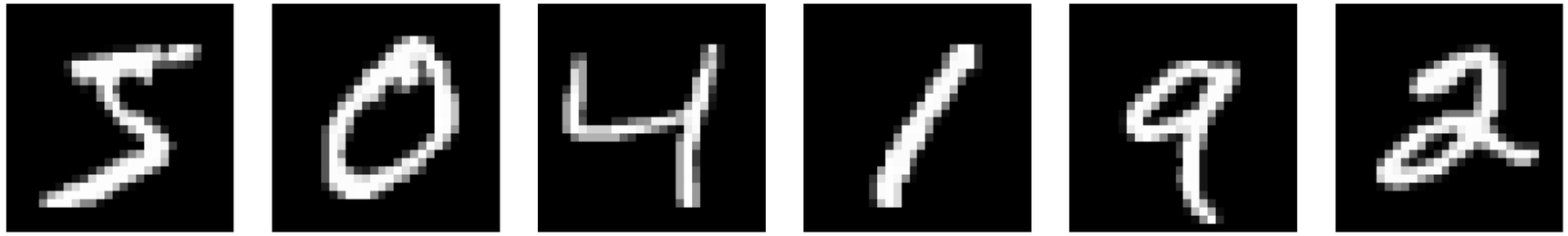
Next: move from generating text to generating images, beginning with models that reconstruct an image.

We will build from reconstructing images to generating new ones

We generated text one token at a time. Now we turn to **images**, starting with two simpler learning tasks.

1. **Reconstruct an image:** learn an encoder and decoder with an autoencoder. *(Today)*
2. **Restore a corrupted image:** train a denoising autoencoder to recover the clean image. *(Today)*
3. **Make the mappings probabilistic:** then add a generative model to develop VAEs.
4. **Evaluate the results:** compare image quality, diversity, usefulness and cost. *(Later)*

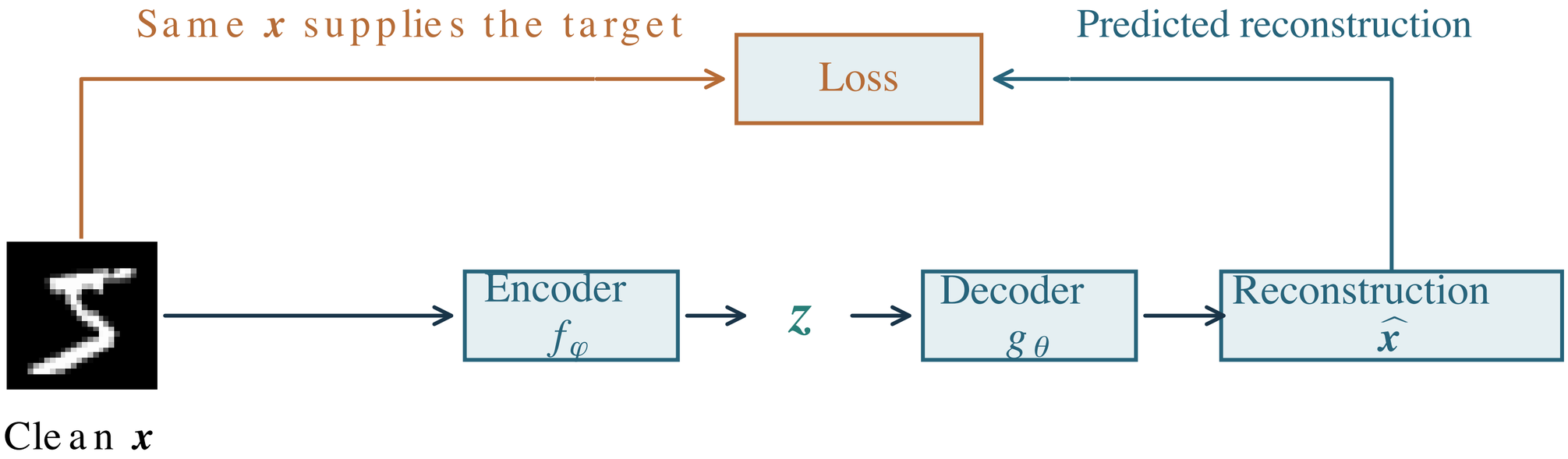
We will use handwritten digits and small color images as examples



- **MNIST:** trace a recognizable digit through encoding, corruption and reconstruction.
- **CIFAR-10:** compare models on 32 × 32 color images with varied objects and backgrounds.

Use the same data, evaluation protocol and declared compute budget for comparisons.

An autoencoder learns a code that supports reconstruction



MNIST training image 0; reconstruction is a symbol, not a trained output.

$$z = f_\phi(x), \qquad \hat{x} = g_\theta(z)$$

Reconstruction error trains both halves together

For images $\boldsymbol{x}_i \in \mathbb{R}^d$, one choice is mean squared error:

$$\min_{\phi,\theta} \frac{1}{nd} \sum_{i=1}^{n} \|\boldsymbol{x}_i - g_\theta(f_\phi(\boldsymbol{x}_i))\|_2^2 .$$

- The image supplies its own reconstruction target.
- Backpropagation updates encoder and decoder parameters through the same loss.

A small pixel error measures reconstruction, not every property we care about.

Perfect copying can solve the loss without a useful code

If both functions can be the identity,

$$f(\boldsymbol{x}) = \boldsymbol{x}, \qquad g(\boldsymbol{z}) = \boldsymbol{z} \quad \Longrightarrow \quad \|\boldsymbol{x} - g(f(\boldsymbol{x}))\|_2^2 = 0.$$

The reconstruction objective needs a constraint that makes learning useful structure worthwhile.

Our first choice: pass through a smaller code, $\boldsymbol{z} \in \mathbb{R}^k$ with $k < d$.

Remember: PCA encodes an input in fewer features and reconstructs it

We already know a **linear encoder and decoder**: PCA.

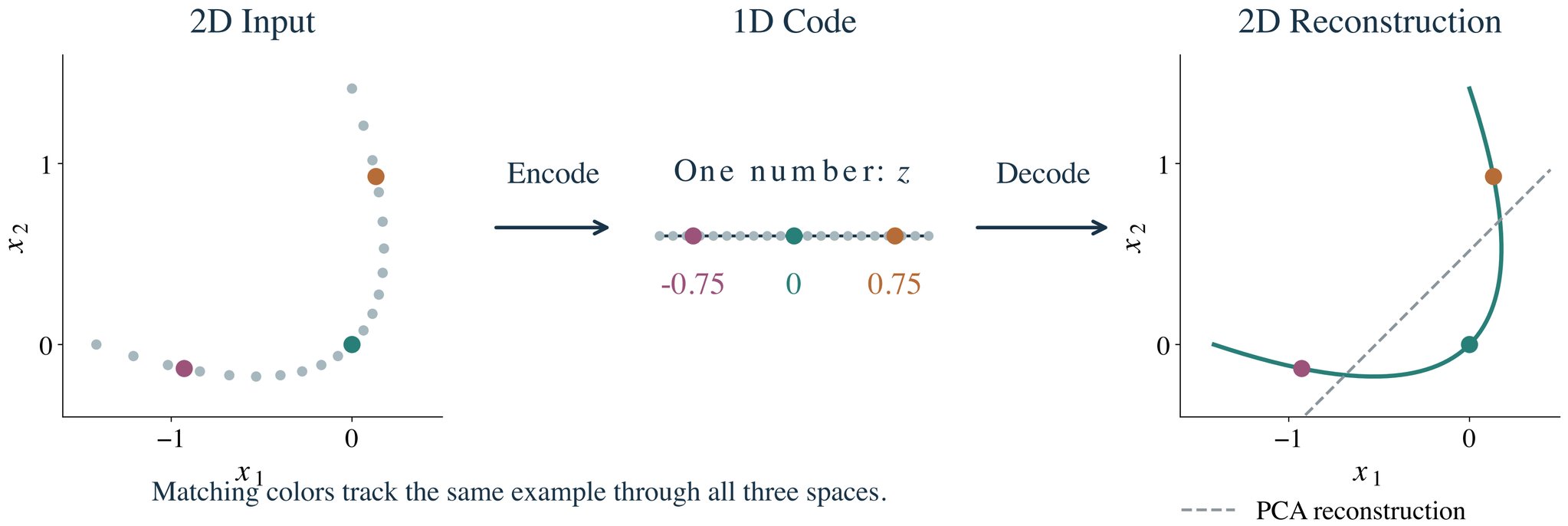
For a centered input $\boldsymbol{x}$ and principal directions $U_k \in \mathbb{R}^{d \times k}$:

$$\underbrace{\boldsymbol{z} = U_k^\top \boldsymbol{x}}_{\text{encode into } k \text{ features}} \quad , \qquad \underbrace{\widehat{\boldsymbol{x}} = U_k \boldsymbol{z}}_{\text{decode back to } d \text{ features}} \quad .$$

This gives an optimal rank-k linear reconstruction under squared error.

Learned linear AE factors can differ while producing the same reconstruction.

A nonlinear decoder maps a 1D code back to a 2D curve



Matching colors track the same example through all three spaces.

Encode: $z = (x_1 + x_2)/\sqrt{2}$. **Decode:** $\widehat{x} = \frac{1}{\sqrt{2}}[z - z^2,\ z + z^2]^\mathsf{T}$.

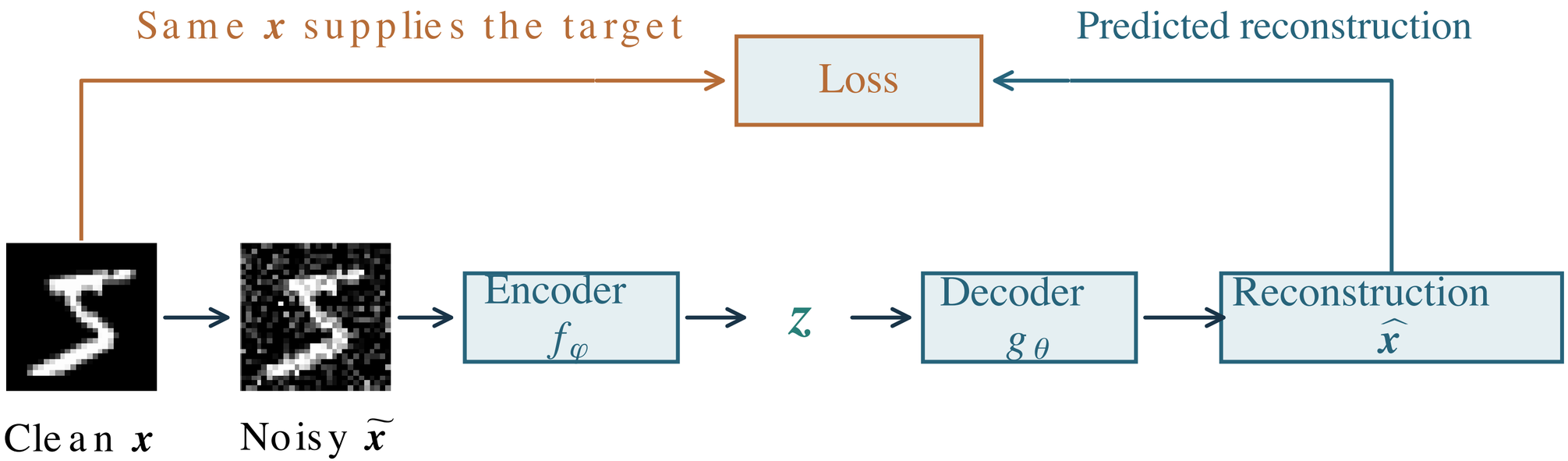
The code combines both input features. *Analytic example; no model was trained.*

What does a small reconstruction error tell us?

> **An AE reconstructs held-out images accurately.**
> **Which conclusion is supported?**
> **A. Its codes separate all useful classes.**
> **B. Its reconstructions preserve the measured image features well.**
> **C. Random Gaussian codes produce realistic images.**

B. Classification and generation need their own evidence.

A denoising AE receives corruption but targets the clean image



MNIST training image 0; reconstruction is a symbol, not a trained output.

Copying the corrupted input now disagrees with the clean target.

Denoising trains against the original, not the corrupted input

Construct $\widetilde{x} = c(x, e)$ using a sampled corruption:

$$\min_{\phi,\theta} \; \mathbb{E}_{\mathbf{x},\mathbf{e}} \left[\frac{1}{d} \|\mathbf{x} - g_\theta(f_\phi(c(\mathbf{x}, \mathbf{e})))\|_2^2 \right].$$

Input sent to encoder	Target used in loss
Corrupted image $\widetilde{x}$	Clean image x

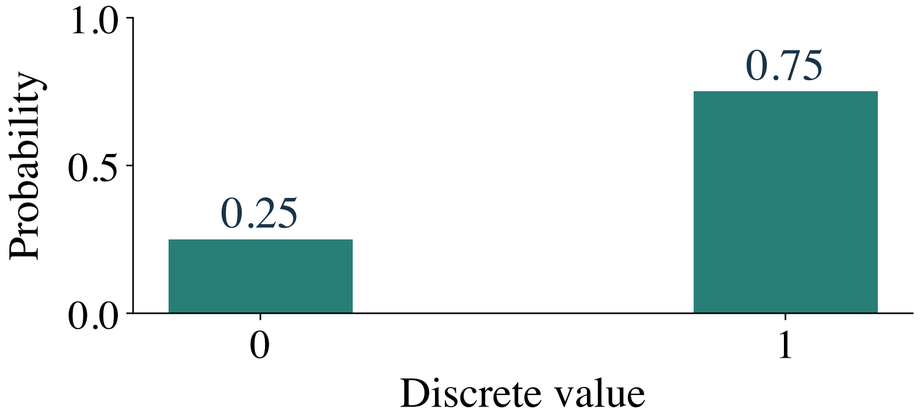
The chosen corruption shapes which structure helps solve the task.

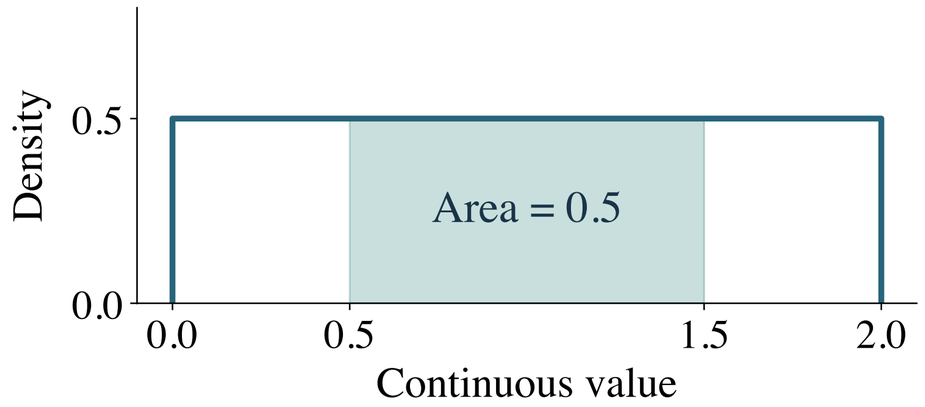
Probability primer

Distributions, expectations, likelihood, and KL

A distribution describes possibilities; a sample is one outcome

For a latent vector, $\mathbf{z} \sim p$ describes a random draw; z denotes one realized value.

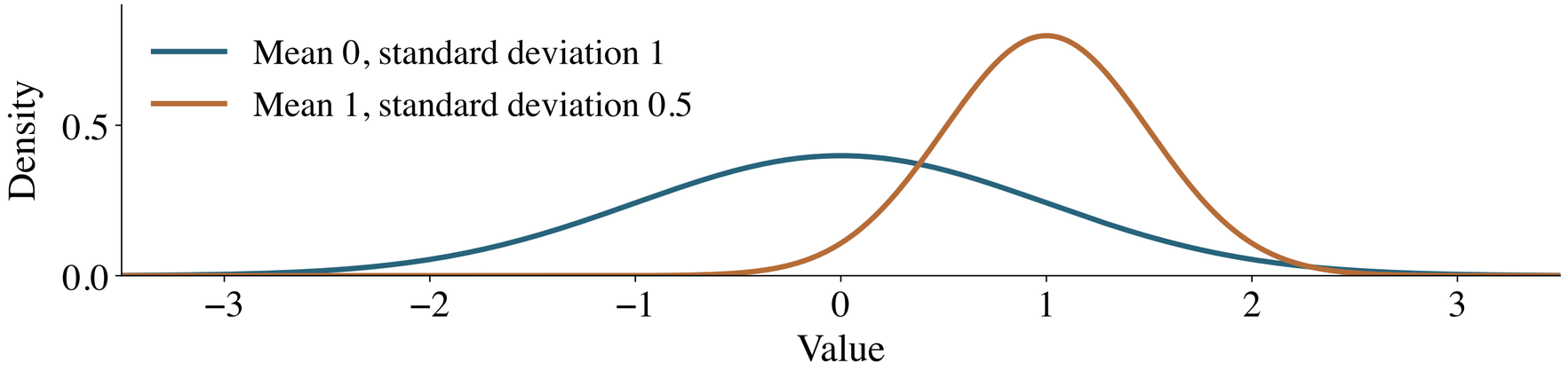




- **Discrete:** probabilities at individual values sum to one.
- **Continuous:** density integrates to one; probability is **area over a region**, not the height at a point.

Gaussian distributions specify a center and a spread

For one feature, $z \sim \mathcal{N}(\mu, \sigma^2)$: **mean** μ, **variance** σ^2, **standard deviation** σ.



For vectors, $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ uses a **covariance matrix**.

- Diagonal entries are feature variances; off-diagonal entries are covariances.
- $\Sigma = I$: independent Gaussian features, each with variance one.

A conditional distribution holds one quantity fixed

$p(x \mid z)$ describes possible images x **for a given code** z.

- Fix the code; the distribution over images still integrates to one.
- Changing the code can change the image distribution.
- $p(z \mid x)$ asks a different question: which codes are plausible for a given image?

The **joint distribution** describes images and codes together:

$$p(x, z) = p(z)\, p(x \mid z).$$

The vertical bar means **given**, not division or causation.

Marginalization removes a variable; Bayes' rule reverses conditioning

To describe images without specifying a code, average over possible codes:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} = \int p(\boldsymbol{z}) p(\boldsymbol{x} \mid \boldsymbol{z})\, d\boldsymbol{z}.$$

For an observed image, Bayes' rule gives

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{z})\, p(\boldsymbol{z})}{p(\boldsymbol{x})}.$$

- $p(\boldsymbol{z})$: code distribution before seeing the image (**prior**).
- $p(\boldsymbol{z} \mid \boldsymbol{x})$: code distribution after seeing it (**posterior**).

For discrete codes, replace the integral with a sum.

An expectation is a probability-weighted average

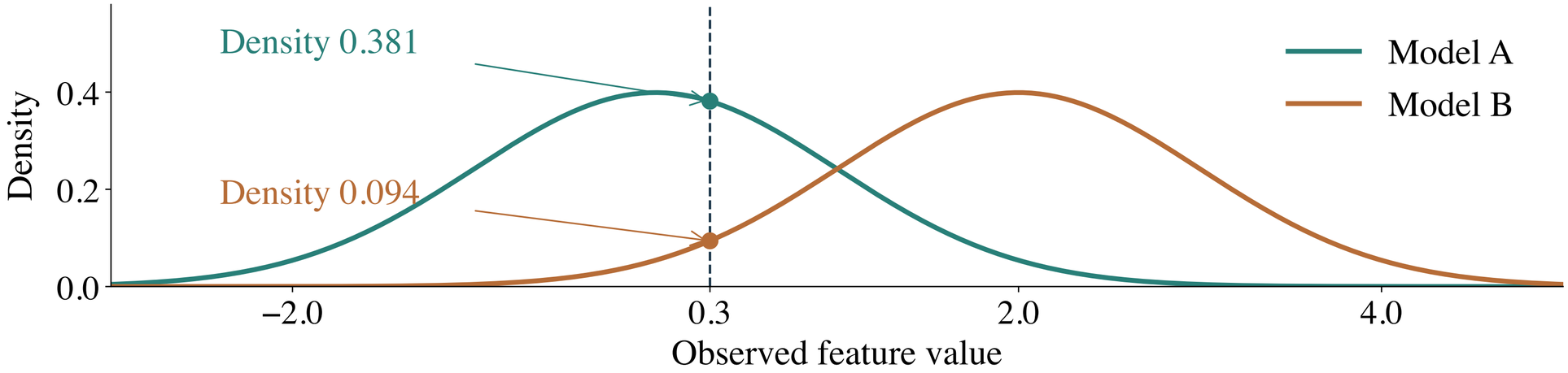
If a loss is 1 with probability 0.75 and 5 with probability 0.25, its expectation is $0.75(1) + 0.25(5) = 2$.

For a function of a continuous code,

$$\mathbb{E}_{\mathbf{z}\sim q}[h(\mathbf{z})] = \int h(\boldsymbol{z})q(\boldsymbol{z})\,d\boldsymbol{z} \;\approx\; \frac{1}{m}\sum_{j=1}^{m} h(\boldsymbol{z}^{(j)}), \qquad \boldsymbol{z}^{(j)} \text{ sampled from } q.$$

- The integral is the expectation; the sample average **estimates** it.
- One sample gives a noisy estimate, not the exact average.
- For reconstruction, fix the image and average its loss over codes drawn from its encoder.

Likelihood evaluates an observation we already have



- **Sampling:** hold the model fixed and draw a new observation.
- **Likelihood:** hold the observed $\boldsymbol{x}$ fixed and evaluate $p_\theta(\boldsymbol{x})$ as the model changes.
- A common loss is $-\log p_\theta(\boldsymbol{x})$: higher density gives lower loss.

For a decoder, evaluate $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ at the **original image**.

KL compares distributions by averaging a log-density ratio

The **Kullback–Leibler divergence** is

$$D_{\mathrm{KL}}(q\|p) = \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z})}\right].$$

- Average under q: disagreement where q puts weight matters.
- KL is nonnegative and zero when the distributions agree.
- It is directional: generally $D_{\mathrm{KL}}(q\|p) \neq D_{\mathrm{KL}}(p\|q)$.

For two outcomes, $q = (0.75, 0.25)$ and $p = (0.5, 0.5)$ give $D_{\mathrm{KL}}(q\|p) \approx 0.131$; setting $p = q$ gives zero.

We will use KL to compare **code distributions**, including their centers and spreads.