From Probabilistic Autoencoders to VAEs

David I. Inouye

Dynamic Content · Last Updated: November 9, 2026

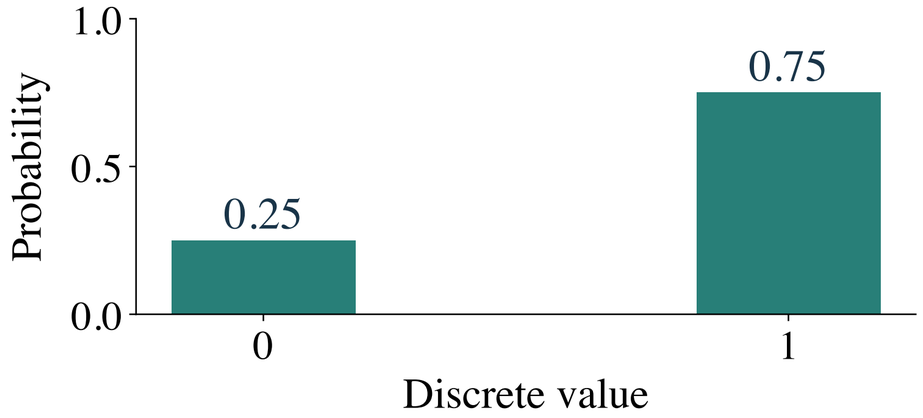
Arc 7 · Image generation and evaluation

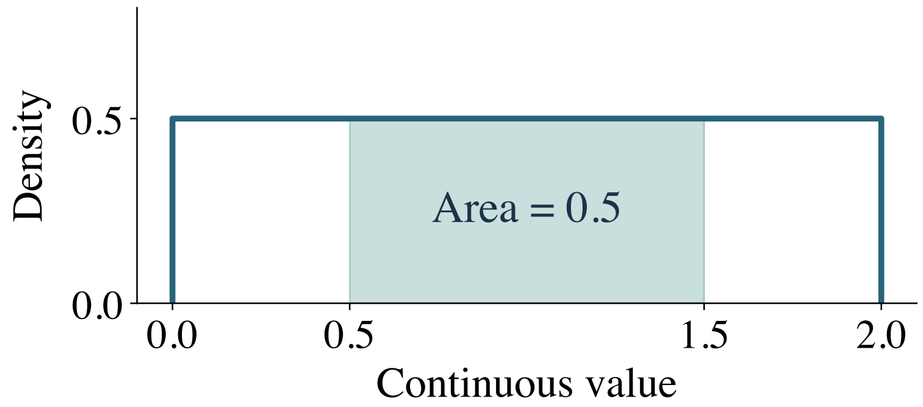
Probability primer

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.

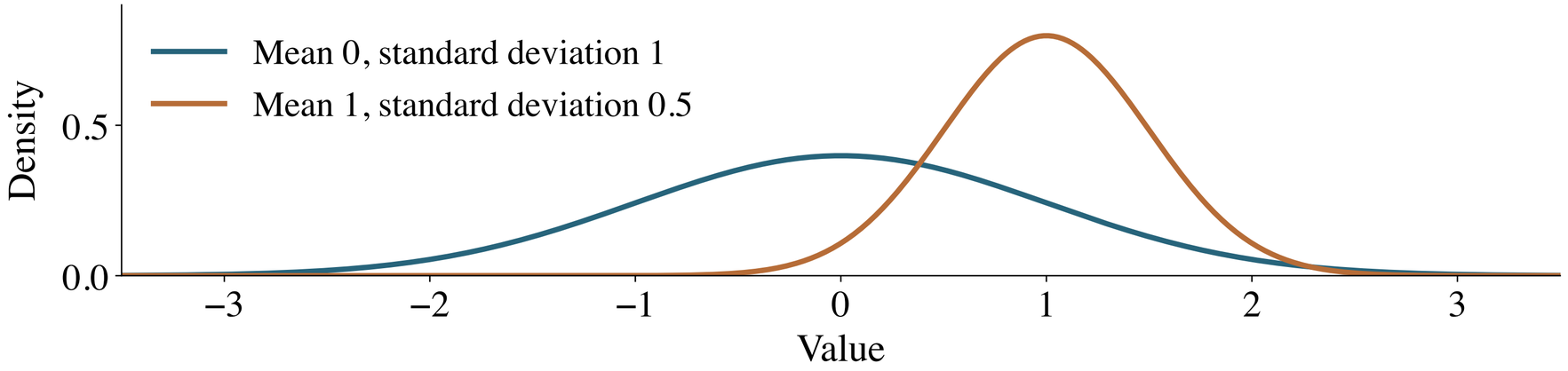




- **Discrete:** $p(x) \geq 0$ and $\sum_x p(x) = 1$.
- **Continuous:** $p(x) \geq 0$ and $\int p(x)\, dx = 1$; probability is **area**, not density height.

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.

Marginalization sums or integrates out a variable

The **joint distribution** $p(\boldsymbol{x}, \boldsymbol{z})$ describes images and codes together. To describe images alone, remove the code variable:

Discrete codes:

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

Continuous codes:

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

Similarly, $p(\boldsymbol{z}) = \int p(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{x}$ for continuous images.

A conditional distribution holds one quantity fixed

Divide the joint by the marginal of the quantity being held fixed:

$$p(\boldsymbol{x} \mid \boldsymbol{z}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{z})} \quad \text{for } p(\boldsymbol{z}) > 0.$$

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})} \quad \text{for } p(\boldsymbol{x}) > 0.$$

- $p(\boldsymbol{x} \mid \boldsymbol{z})$: possible images for a **fixed code**.
- $p(\boldsymbol{z} \mid \boldsymbol{x})$: possible codes for a **fixed image**.

For fixed $\boldsymbol{z}$, $\int p(\boldsymbol{x} \mid \boldsymbol{z})\, d\boldsymbol{x} = 1$; use a sum for discrete images.

The chain rule and Bayes' rule use the same joint in two directions

Multiply each conditional definition by its denominator:

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

Rearrange to get **Bayes' rule**, for $p(\boldsymbol{x}) > 0$:

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

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

The denominator comes from marginalization: $p(\boldsymbol{x}) = \int p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z})\, d\boldsymbol{z}$.

An expectation is a probability-weighted average

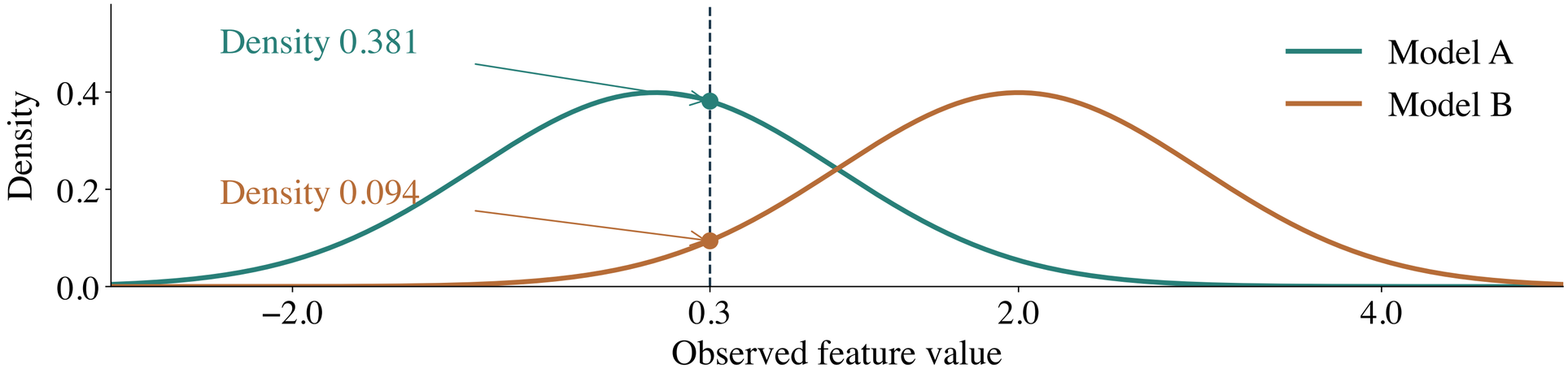
If a loss is 1 with probability 0.75 and 5 with probability 0.25, its expectation is $0.75(1) + 0.25(5) = 2$.

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.

1. **Nonnegative:** $D_{\mathrm{KL}}(q\|p) \geq 0$.
2. **Zero when the distributions agree:** $D_{\mathrm{KL}}(q\|q) = \mathbb{E}_{\mathbf{z}\sim q}[\log 1] = 0$.

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.

Probabilistic autoencoders

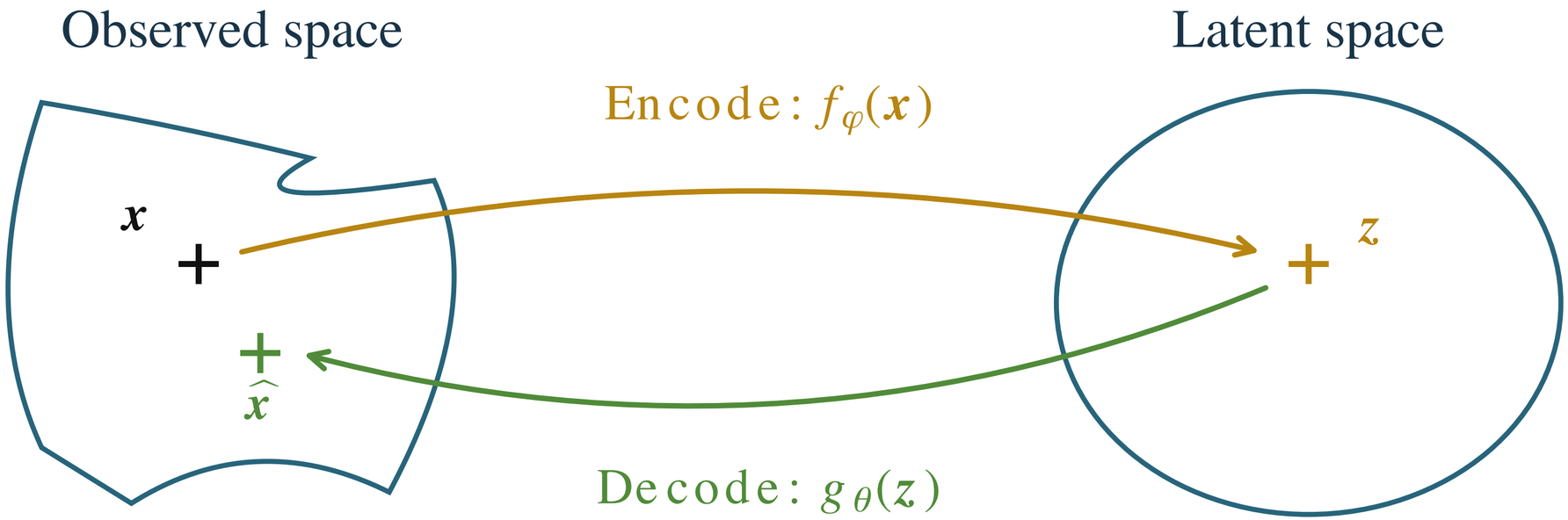
Replace point outputs with distributions

We can make both encoder and decoder probabilistic

1. **Probabilistic AE:** replace each point output with a distribution.
2. **VAE generative model:** add an assumption about where latent codes come from.
3. **VAE training:** fit the generative model using the encoder to infer codes.

First, encode a point into latent space, then decode back into the **same observed space**.

A deterministic AE maps one point to one point

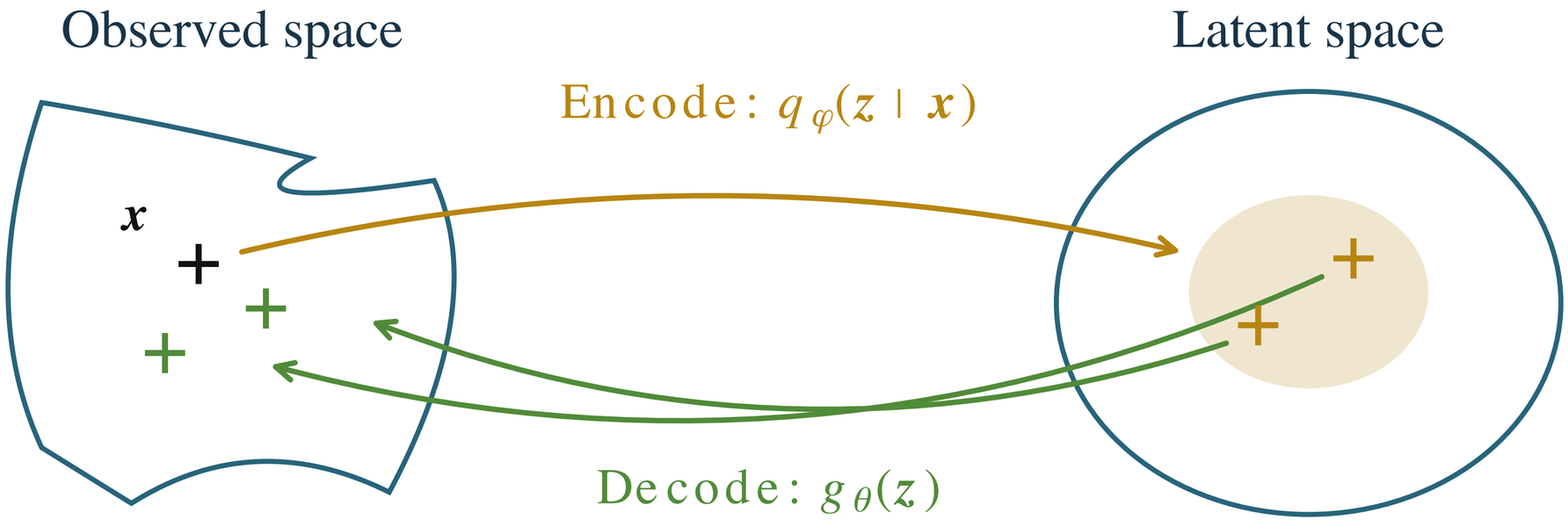


The reconstruction returns to the same observed space.

$$\boldsymbol{z} = f_\phi(\boldsymbol{x}), \qquad \widehat{\boldsymbol{x}} = g_\theta(\boldsymbol{z}).$$

Each **+** is one example or code. The outlines show the spaces where they live.

A probabilistic encoder maps an image to a distribution of codes

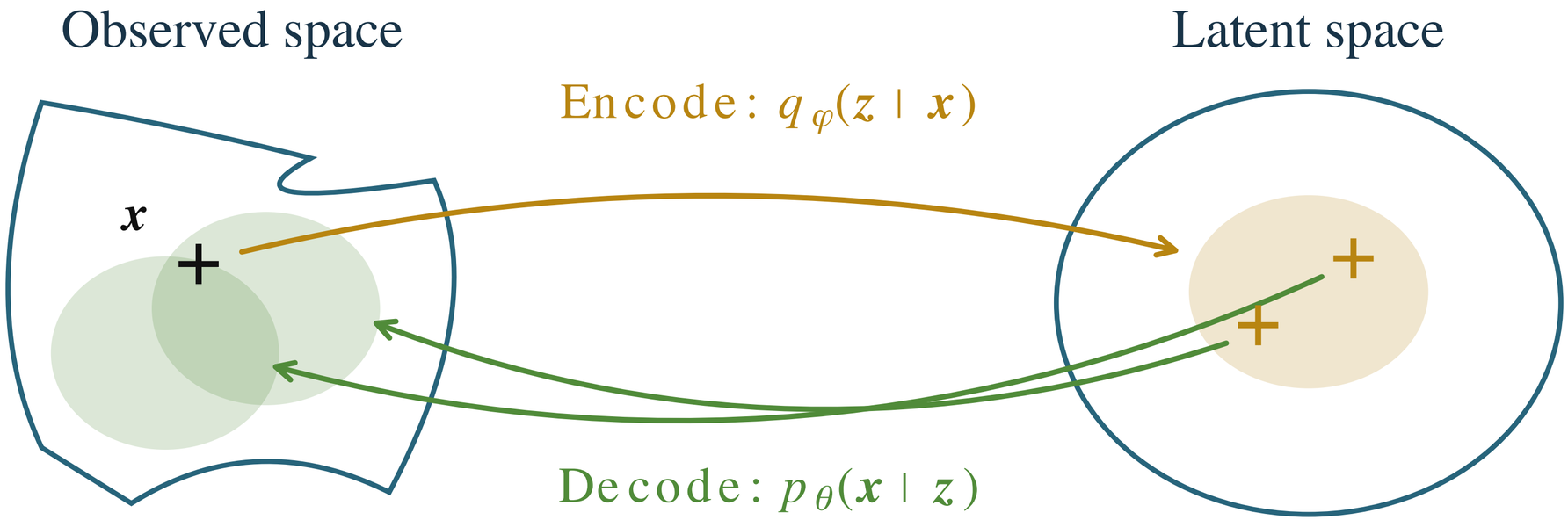


Sample latent codes; each decodes to a point.

Replace $z = f_\phi(x)$ with a conditional distribution $q_\phi(z \mid x)$.

The yellow circle is a code distribution; its **+ marks are sampled codes**. The decoder still returns one point per code to the original observed space.

A probabilistic decoder maps each code to a distribution of images



Training: evaluate the original + under each green distribution.

Replace $\widehat{x} = g_\theta(z)$ with $p_\theta(x \mid z)$.

During training, measure **the likelihood of the original input image** under the distributions represented by the green circles. We sample latent codes, **not new images from the decoder**.

Gaussian mappings let the networks predict the distribution means

For a first example, hold both covariance matrices fixed at the identity:

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; f_\phi(\boldsymbol{x}), I) \,.$$

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; g_\theta(\boldsymbol{z}), I) \,.$$

- f_ϕ predicts the center of the latent distribution.
- g_θ predicts the center of the image distribution.
- We can also learn the spreads; first keep them fixed to see the mechanism.

A probabilistic AE can train with expected reconstruction loss

For one observed image, average over sampled codes:

$$\mathcal{J}_{\mathrm{PAE}}(\boldsymbol{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\cdot|\boldsymbol{x})}[-\log p_\theta(\boldsymbol{x} \mid \mathbf{z})].$$

With the fixed Gaussian mappings, one noise draw gives the estimate

$$\tfrac{1}{2}\|\boldsymbol{x} - g_\theta(f_\phi(\boldsymbol{x}) + \boldsymbol{\epsilon})\|_2^2 + C, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I).$$

Evaluate the original $\boldsymbol{x}$ under the green decoder distribution. **Sample codes, not output images.**

Where does the noise enter these two autoencoders?

> **Both models reconstruct the original clean image.**
>
> **Which description matches the models we just defined?**
>
> **A. Denoising AE: noisy input; probabilistic AE: sampled latent code.**
>
> **B. Denoising AE: sampled latent code; probabilistic AE: noisy input.**
>
> **C. Both models must use the noisy input as their target.**

A. We changed the input for denoising, then changed the mappings for probabilistic autoencoding.

VAEs: a generative model

Pair a latent prior with a probabilistic decoder

A prior and decoder define an assumed image-generation process

- We **assume a generative process** as a way to model images; it need not match how real images are created.
- Pair a probabilistic decoder with a **prior** on latent codes to define a two-stage sampling process:

1. Sample a latent code: $\mathbf{z} \sim p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, I)$.
2. Sample an image: $\mathbf{x} \sim p_\theta(\boldsymbol{x} \mid \mathbf{z})$.

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

We will show why an encoder is useful in a few slides.

Fitting the generative model means averaging over possible codes

For an observed image, the model assigns density

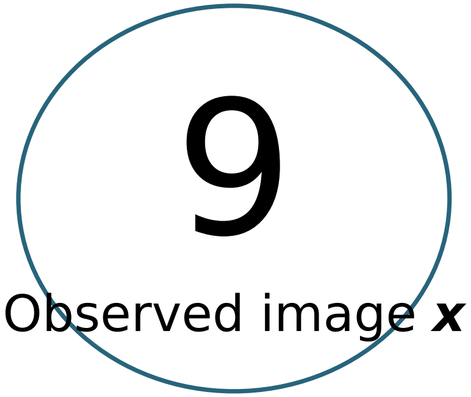
$$p_\theta(\boldsymbol{x}) = \int p(\boldsymbol{z})\, p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\, d\boldsymbol{z}.$$

- We would like to maximize $\sum_i \log p_\theta(\boldsymbol{x}_i)$.
- The code that produced an observed image is unknown.
- With a nonlinear decoder, this integral is generally difficult to compute exactly.

The encoder helps identify codes that could explain each observed image.

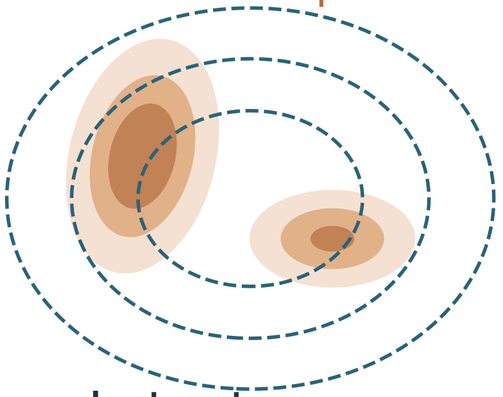
The VAE encoder approximates the codes that explain an image

Observe an image



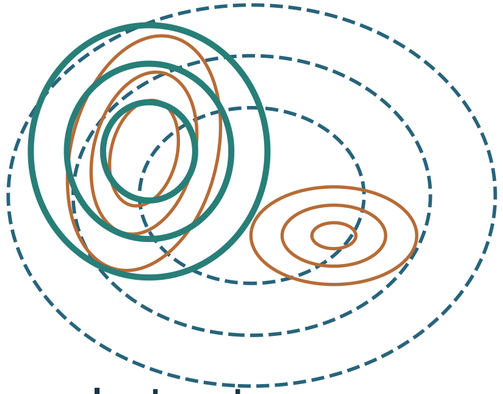
Prior → posterior
$p(\boldsymbol{z}) \rightarrow p_\theta(\boldsymbol{z} \mid \boldsymbol{x})$

Several codes explain this 9



Gaussian approximation
$q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$

Other modes can be missed



Schematic: dashed blue = prior · orange = complex posterior · teal = Gaussian approximation

The **prior** $p(z)$ is shared; the **posterior** $p_\theta(z \mid x)$ depends on this image.

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z}; \boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x}))\big) \quad \text{approximates } p_\theta(\boldsymbol{z} \mid \boldsymbol{x}).$$

Training a VAE

Use the encoder to fit the generative model

Encoder samples let us estimate a usable training objective

For a fixed image, draw codes from $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and define the **evidence lower bound (ELBO)**

$$\mathcal{L}(\theta, \phi; \boldsymbol{x}) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\cdot \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \boldsymbol{x}) \right].$$

- Evaluate the joint with $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z}) p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$.
- These densities are available; the difficult marginal $p_\theta(\boldsymbol{x})$ is absent.
- Average sampled values to estimate $\mathcal{L}$.

Training objective: maximize $\mathcal{L}$ over $ heta,$ (equivalently, minimize $-\mathcal{L}$).

Next, derive why it is a lower bound on image log likelihood.

Multiplying and dividing by the encoder separates the ELBO and its gap

Fix $\boldsymbol{x}$; abbreviate $q = q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$. Every expectation below is under q.

$$
\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \mathbb{E}_q[\log p_\theta(\boldsymbol{x})] && \text{(fixed image)} \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\boldsymbol{x},\mathbf{z})}{p_\theta(\mathbf{z}|\boldsymbol{x})}\right] && \text{(chain rule)} \\
&= \mathbb{E}_q\left[\log \left(\frac{p_\theta(\boldsymbol{x},\mathbf{z})}{q_\phi(\mathbf{z}|\boldsymbol{x})}\,\frac{q_\phi(\mathbf{z}|\boldsymbol{x})}{p_\theta(\mathbf{z}|\boldsymbol{x})}\right)\right] && \text{(multiply by } q/q\text{)} \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\boldsymbol{x},\mathbf{z})}{q_\phi(\mathbf{z}|\boldsymbol{x})}\right] + \mathbb{E}_q\left[\log \frac{q_\phi(\mathbf{z}|\boldsymbol{x})}{p_\theta(\mathbf{z}|\boldsymbol{x})}\right] && \text{(split the log)} \\
&= \mathcal{L}(\theta, \phi; \boldsymbol{x}) + D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \,\|\, p_\theta(\cdot \mid \boldsymbol{x})) && \text{(definitions)}
\end{aligned}
$$

The ELBO gives a lower bound on image log likelihood

For one image, the following identity connects inference and generation:

$$\log p_\theta(\boldsymbol{x}) = \underbrace{\mathcal{L}(\theta, \phi; \boldsymbol{x})}_{\text{ELBO}} + \underbrace{D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \,\|\, p_\theta(\cdot \mid \boldsymbol{x}))}_{\text{gap from approximate inference}}.$$

Because KL is nonnegative, $\mathcal{L}(\theta, \phi; \boldsymbol{x}) \leq \log p_\theta(\boldsymbol{x})$.

- Fit θ and ϕ by maximizing this **evidence lower bound**.
- The bound is tight when the encoder equals the true posterior.

The VAE loss balances reconstruction and agreement with the prior

Factor $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ inside the ELBO:

$$\mathcal{L} = \mathbb{E}_q[\log p_\theta(\boldsymbol{x} \mid \mathbf{z})] + \mathbb{E}_q\left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z} \mid \boldsymbol{x})}\right].$$

Minimize the negative ELBO:

$$\mathcal{J} = -\mathcal{L} = \underbrace{\mathbb{E}_q[-\log p_\theta(\boldsymbol{x} \mid \mathbf{z})]}_{\text{reconstruction penalty}} + \underbrace{D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \,\|\, p(\cdot))}_{\text{prior regularization}}.$$

Two different KL terms	Meaning
$D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \| p_\theta(\cdot \mid \boldsymbol{x}))$	**True posterior:** inference error for this image
$D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \| p(\cdot))$	**Prior:** regularization toward shared sampling codes

β controls how strongly codes must agree with the prior

$$\mathcal{J}_\beta = \underbrace{\mathbb{E}_q[-\log p_\theta(\boldsymbol{x} \mid \mathbf{z})]}_{\text{reconstruction}} + \beta \underbrace{D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \| p(\cdot))}_{\text{prior regularization}}.$$

Weight	What we optimize
$\beta = 0$	Probabilistic autoencoder: reconstruction alone
$0 < \beta < 1$	Weaker prior regularization
$\beta = 1$	Standard VAE: **exactly the negative ELBO**
$\beta > 1$	Stronger prior regularization (**β-VAE**)

More prior pressure can reduce information about the image and worsen reconstruction.

Side point: as $\beta \to \infty$, $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = p(\boldsymbol{z})$ gives zero KL. A sufficiently expressive decoder could still fit images with $p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = p_\theta(\boldsymbol{x})$, although $\mathbf{x}$ and $\mathbf{z}$ are independent.

A Gaussian decoder turns likelihood into scaled squared error

Part 1 of the VAE objective: reconstruction.

Let the decoder predict an image mean $g_\theta(\boldsymbol{z})$ with fixed variance τ^2:

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}; g_\theta(\boldsymbol{z}), \tau^2 I\right).$$

Then, for a d-feature image,

$$-\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \frac{\|\boldsymbol{x} - g_\theta(\boldsymbol{z})\|_2^2}{2\tau^2} + \frac{d}{2}\log(2\pi\tau^2).$$

- τ is the **decoder observation spread**; $\boldsymbol{\sigma}$ describes encoder codes.
- With fixed τ, the last term is constant during training.
- Changing the error scale changes its balance with KL.

For Gaussian codes, we can calculate KL directly

Part 2 of the VAE objective: regularization. Compare the **encoder's approximate posterior** with the **prior**:

$$q_\phi(\cdot \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \operatorname{diag}(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}))\right), \qquad p(\cdot) = \mathcal{N}(\mathbf{0}, I).$$

$$D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \,\|\, p(\cdot)) = \frac{1}{2} \sum_{j=1}^{k} \left(\mu_j^2 + \sigma_j^2 - 1 - \log \sigma_j^2\right).$$

Here $\boldsymbol{\mu} = \boldsymbol{\mu}_\phi(\boldsymbol{x})$ and $\boldsymbol{\sigma} = \boldsymbol{\sigma}_\phi(\boldsymbol{x})$ are this image's encoder outputs.

- Mean $\mathbf{0}$ and spread $\mathbf{1}$ match the prior and give zero KL.
- This term is analytic: **no latent-code sampling is needed**.

The encoder changes the distribution inside the expectation

To maximize the ELBO, we need $\nabla_{\theta,\phi}\mathcal{L}$ (or minimize $\mathcal{J} = -\mathcal{L}$). The Gaussian KL is explicit; reconstruction still needs an expectation.

Let $\ell_\theta(\boldsymbol{z}) = -\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ for fixed $\boldsymbol{x}$:

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[\ell_\theta(\mathbf{z})] = \nabla_\phi \int q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\ell_\theta(\boldsymbol{z})\, d\boldsymbol{z} \neq \mathbb{E}_{q_\phi}[\nabla_\phi \ell_\theta(\mathbf{z})] \quad \text{in general.}$$

- At a fixed code, ℓ_θ has no direct ϕ dependence; the naive right side is zero.
- But changing the encoder changes **which codes are likely**.
- We need to move ϕ into a differentiable function of **parameter-independent noise**.

Reparameterization makes sampling a differentiable calculation

For a diagonal Gaussian encoder, an **exact alternative sampling procedure** is

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I), \qquad \boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}.$$

1. Draw noise from a distribution independent of the encoder parameters.
2. Scale it by the encoder's standard deviations.
3. Shift it by the encoder's means.

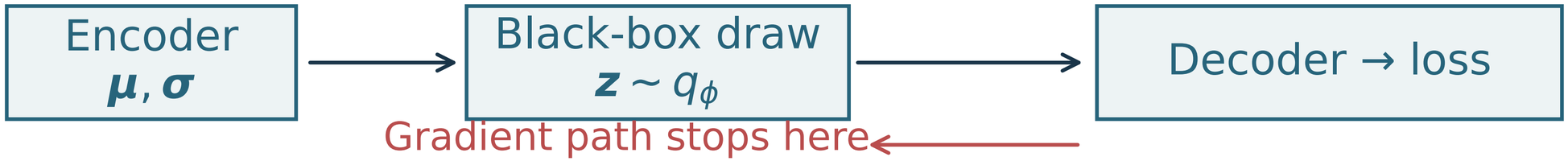
The transformed code has exactly distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$:

$$\nabla_\phi \mathbb{E}_{q_\phi}[\ell_\theta(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)}[\nabla_\phi \ell_\theta(\boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\epsilon})] \,.$$

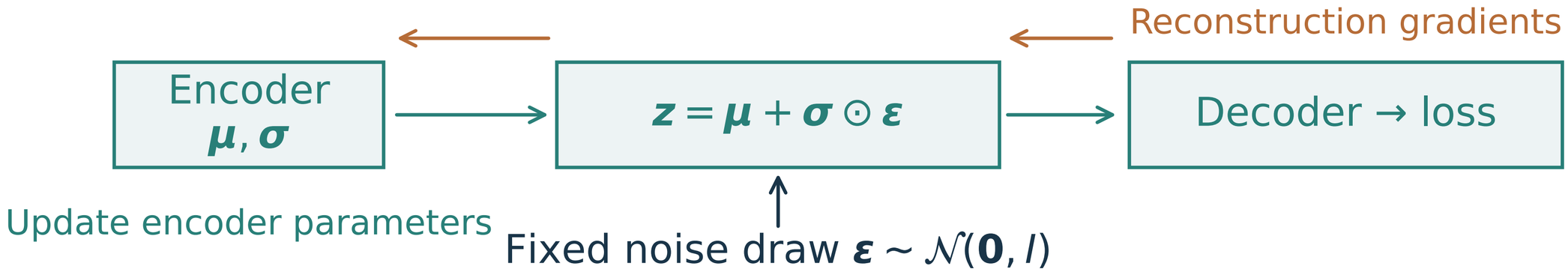
Now the noise distribution is fixed. More complex reparameterizations exist; this Gaussian case is the simplest common one.

Fixed noise lets backpropagation follow the encoder-to-loss path

Direct sampling: no ordinary pathwise derivative through the draw



Reparameterized sampling: differentiate the scale and shift



Hold ϵ fixed **for this gradient calculation**; differentiate the scale and shift, not the random draw.

Monte Carlo sampling estimates the expectation and its gradient

Draw M independent noises and form codes **via the reparameterization trick**:

$$\boldsymbol{\epsilon}^{(m)} \sim \mathcal{N}(\mathbf{0}, I), \qquad \boldsymbol{z}^{(m)} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}^{(m)}.$$

$$\widehat{\mathcal{J}}_\beta = \underbrace{\frac{1}{M}\sum_{m=1}^{M} -\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}^{(m)})}_{\text{Monte Carlo reconstruction estimate}} + \beta \underbrace{D_{\mathrm{KL}}(q_\phi(\cdot \mid \boldsymbol{x}) \| p(\cdot))}_{\text{closed-form Gaussian KL}}.$$

- Backpropagate $\nabla_{\theta,\phi}\widehat{\mathcal{J}}_\beta$; at $\beta = 1$, it estimates $-\nabla_{\theta,\phi}\mathcal{L}$.
- **Usually $M = 1$ per image**; average over a minibatch and draw fresh noise each update.
- Reparameterization supplies the gradient path; Monte Carlo approximates the integral.

Each minibatch trains the encoder and decoder together

For a batch $\{\boldsymbol{x}_i\}_{i=1}^{B}$, compute all B examples in parallel:

Step	Batched computation (each $i = 1, \dots, B$)
1. Encode	$(\boldsymbol{\mu}_i, \boldsymbol{s}_i) = f_\phi(\boldsymbol{x}_i)$; $\boldsymbol{s}_i$ is log variance
2. Sample codes	$\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, I)$; $\boldsymbol{z}_i = \boldsymbol{\mu}_i + \exp(\boldsymbol{s}_i/2) \odot \boldsymbol{\epsilon}_i$
3. Decode and score	$r_i = -\log p_\theta(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$; $k_i = \frac{1}{2}\sum_j(\mu_{ij}^2 + e^{s_{ij}} - 1 - s_{ij})$
4. Average the loss	$\widehat{\mathcal{J}}_\beta = \frac{1}{B}\sum_{i=1}^{B}(r_i + \beta k_i)$
5. Update both networks	$\theta \leftarrow \theta - \eta\nabla_\theta\widehat{\mathcal{J}}_\beta$; $\phi \leftarrow \phi - \eta\nabla_\phi\widehat{\mathcal{J}}_\beta$

One fresh code per image. At $\beta = 1$, the batch loss estimates the average negative ELBO.

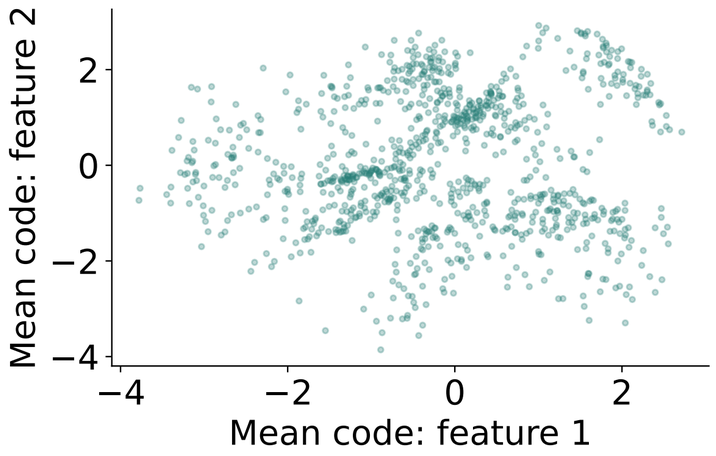
What happens if we train using only the KL term?

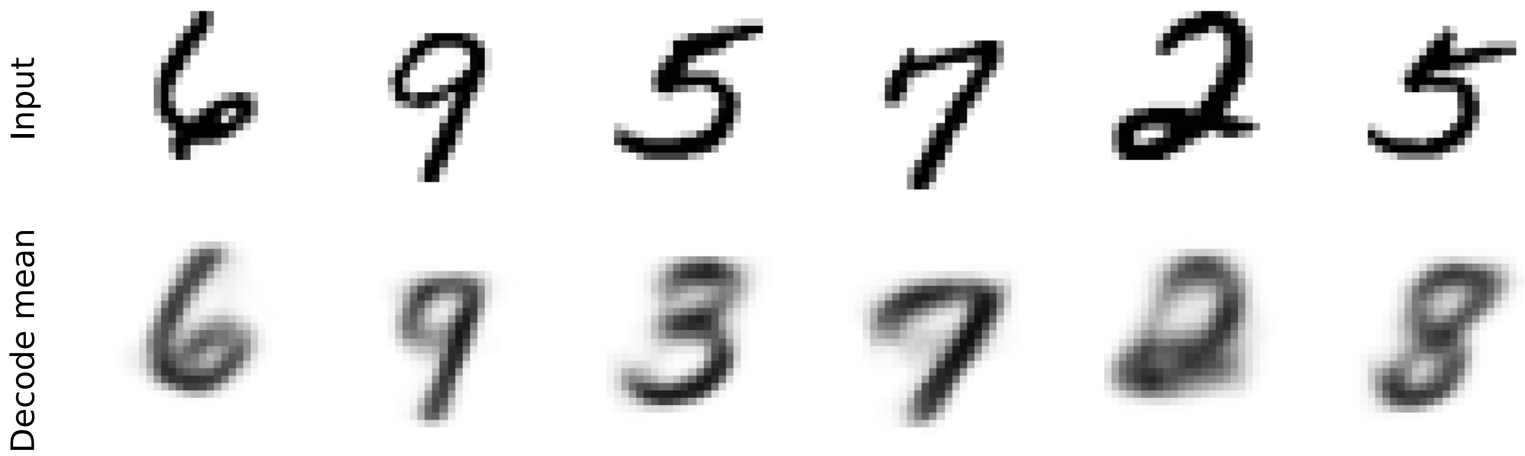
The prior is $\mathcal{N}(0, I)$ and the encoder can represent it exactly.
Which encoder minimizes KL for every image?
A. A different narrow distribution for each image.
B. A point concentrated at zero for every image.
C. The same standard Gaussian for every image.

C. Matching both center and spread gives zero KL; that encoder distribution carries no image-specific information.

After training, the encoder provides representations and reconstructions

- **Represent:** $x \mapsto \mu_\phi(x)$ gives a feature vector for downstream tasks.
- **Reconstruct:** encode an image, then decode its mean code (or a sampled code).

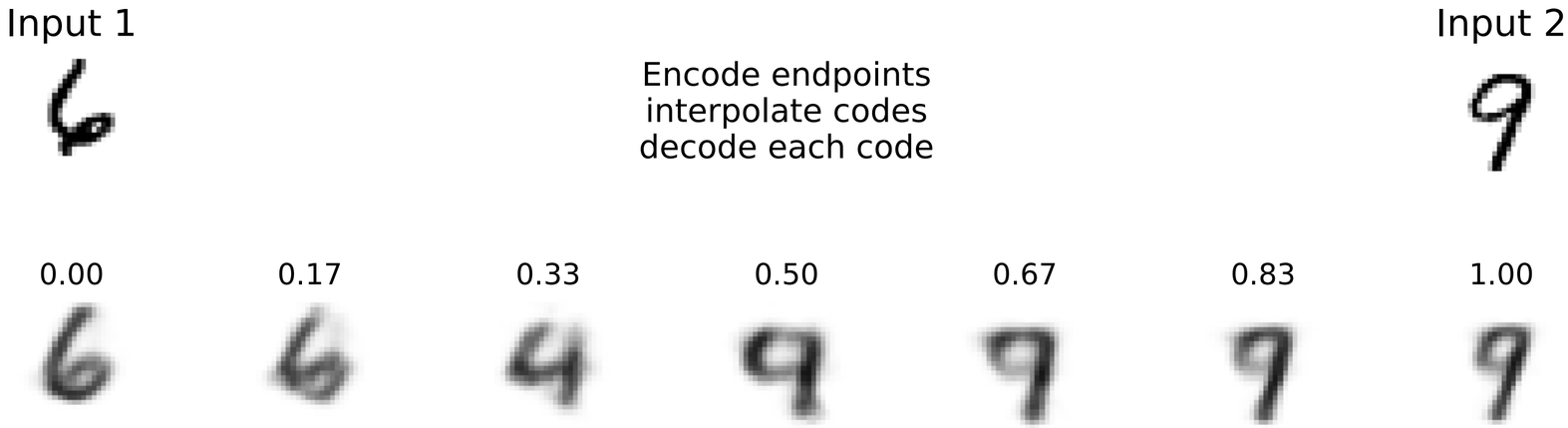




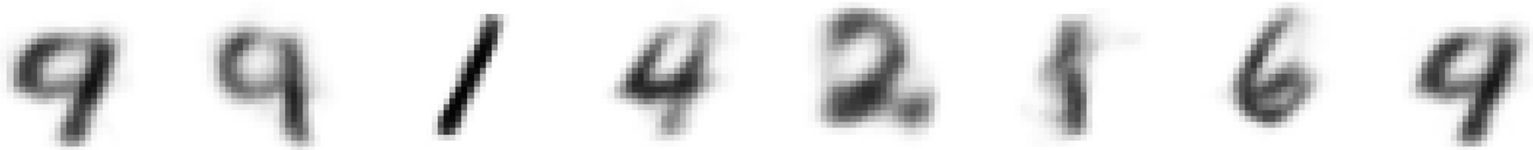
Selected held-out MNIST examples; this demo uses a **Bernoulli decoder**. Reconstructions can lose detail.

Interpolation uses encoded endpoints; generation needs only the decoder

Interpolate: encode two images, then decode $z(a) = (1-a)\mu_1 + a\mu_2$.



Generate: draw $z \sim p(z)$ and decode; **no input image or encoder** is needed.



Decoder means shown; smooth interpolation does not guarantee good prior samples.

A trained VAE still needs evidence for its intended use

Intended use	Evidence to inspect
Reconstruct an observed image	Held-out reconstruction under the stated loss
Generate new images	Fresh prior samples: quality, diversity and failure cases
Use the learned code for another task	Held-out performance on that downstream task

Audit the assumptions: chosen prior, decoder likelihood and approximate-posterior family.

A good training objective gives a learning procedure; it does not guarantee all three uses succeed.

Diffusion returns to denoising at many noise levels

Two ways to generate an image:

1. **VAE:** sample a latent code, then decode an image.
2. **Diffusion:** start from noise and form an image through repeated denoising steps.

The next model learns how to denoise at different noise levels, using the noise level as an input.

Remember the denoising AE: corrupted input, clean-image information as the training signal.