

Variational Autoencoders

ELG 5218 - Uncertainty Evaluation in Engineering Measurements and Machine Learning

Adapted from Philip Schulz and Wilker Aziz <https://github.com/vitutorial/VITutorial>

University of Ottawa

February 25, 2026

Roadmap

- 1 Parametrizing Distributions with Neural Networks
- 2 Autoencoders as deterministic representation learning
- 3 Variational Autoencoders
 - VAE Introduction
 - After training
- 4 Discrete Variational Encoders

Why Neural Networks for Probabilistic Models?

Neural Networks alone

- Excel at supervised tasks
- Struggle with *lack of supervision*
- No built-in uncertainty

Probabilistic Models alone

- Principled uncertainty
- Limited expressiveness
- Hand-crafted features

The marriage: Deep Generative Models

Let neural networks **parametrize the distributions** of a probabilistic model – we get the best of both worlds.

Distributions Parametrized by NNs

Classical approach – fixed parametric family, hand-picked parameters:

$$X \sim \text{Cat}(\pi_1, \dots, \pi_K) \quad \text{or} \quad X \sim \mathcal{N}(\mu, \sigma^2)$$

Deep approach – a neural network maps side information ϕ to parameters:

$$X|\phi \sim \text{Cat}(\pi_\theta(\phi)) \quad \text{or} \quad X|\phi \sim \mathcal{N}(\mu_\theta(\phi), \sigma_\theta(\phi)^2)$$

Concrete example: image captioning

ϕ = image (raw pixels); x = caption token

A CNN maps $\phi \rightarrow h_\phi$, then an RNN maps $h_\phi \rightarrow \pi_\theta(\phi)$ (probability over vocabulary).

Key point

From a statistical perspective, NNs *parametrize distributions* – they do **not** generate data themselves.

Autoencoder (AE): architecture

Autoencoders are deterministic reconstruction models (not probabilistic generative models).

- Encoder $f_\phi : x \mapsto z$
- Decoder $g_\theta : z \mapsto \hat{x}$
- Train to reconstruct: $\hat{x} \approx x$
- ℓ is a generic loss function

Typical objective

$$\min_{\theta, \phi} \sum_{n=1}^N \ell\left(x^{(n)}, g_\theta(f_\phi(x^{(n)}))\right)$$



Regularized Autoencoders: A Brief Taxonomy

Many variants of autoencoders introduce explicit regularization on the encoder or latent code to improve the structure of the learned representation:

- **Denoising Autoencoder (DAE):** Corrupt input $\tilde{x} \sim q(\tilde{x} | x)$ and reconstruct x . Encourages learning of *stable, local manifolds*.
- **Sparse Autoencoder:** Add penalties encouraging sparsity of activations in z (e.g. $\|z\|_1$ or KL divergence constraints). Promotes *parts-based, interpretable* representations.
- **Contractive Autoencoder:** Penalize the encoder Jacobian norm $\|\partial f_\phi(x)/\partial x\|$ to enforce local invariance.

Why This Matters for Bayesian Views

These regularizers can be interpreted as imposing *implicit priors* on representations. However, they still lack a *generative latent-variable model*—there is no explicit $p(z)$ and no uncertainty over latent codes. This motivates probabilistic models such as **VAEs**, which supply both a prior and an inference model.

What Autoencoders Can and Cannot Do

Strengths (What AEs Do Well)

- **Reconstruction:** learn a compact code z that preserves information needed to recover x .
- **Compression:** dimensionality reduction via a learned encoder.
- **Useful representations:** good features for downstream tasks (classification, clustering).
- **Anomaly detection:** large reconstruction error flags unusual data.

For a broader perspective, see [1].

Limitations (What AEs Cannot Do)

- **Uncertainty:** decoder is deterministic; cannot express calibrated uncertainty about reconstructions.
- **Disentanglement:** latent factors rarely separate meaningfully unless explicit constraints or regularization are added.

Variational Autoencoder (VAE) — Overview

Key idea: Combine a *generative model* (decoder) with an *inference model* (encoder) and train them jointly on a single lower-bound objective.

Generative model

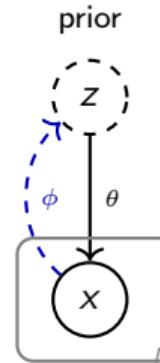
$$p_{\theta}(x, z) = \underbrace{p(z)}_{\text{prior}} \underbrace{p_{\theta}(x|z)}_{\text{NN decoder}}$$

$z \sim \mathcal{N}(0, I)$; NN maps $z \rightarrow$ distribution params

Inference model (encoder)

$$q_{\phi}(z|x) = \mathcal{N}(\mu_{\phi}(x), \text{diag}(\sigma_{\phi}(x))^2)$$

NN maps $x \rightarrow (\mu, \sigma)$; approximates true posterior $p_{\theta}(z|x)$



Solid: generative (θ)

Dashed (blue): inference (ϕ)

Core insight

Maximising a single objective (ELBO) jointly trains both models.

ELBO: Two Equivalent Forms

Form 1 — Joint & entropy

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)]$$

Useful for Monte-Carlo estimation (single z sample).

Form 2 — Reconstruction + Regularisation

$$\mathcal{L}_{\theta,\phi}(x) = \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{expected reconstruction}} - \underbrace{D_{\text{KL}}(q_\phi(z|x) \parallel p(z))}_{\text{regularisation}}$$

Follows from $\log p_\theta(x, z) = \log p(z) + \log p_\theta(x|z)$.

Interpretation

Reconstruction: decode latent z to recover x . **Regularisation:** keep posterior $q_\phi(z|x)$ close to prior $p(z)$.

These two terms are in tension — the balance is automatic.

Intractable Marginal Likelihood

For unsupervised learning we need the **marginal**:

$$p(x|\theta) = \int p(z) p(x|z, \theta) dz$$

Why is this hard?

When $p(x|z, \theta)$ is a *non-linear* neural network, the integral has no closed form – we cannot enumerate all z (continuous space) or it has exponentially many terms (discrete z).

Analytical examples of the two intractable cases:

Discrete z , ($z = c$), continuous x

$$p(x|\theta) = \sum_{c=1}^K \pi_c \mathcal{N}(x|\mu_\theta(c), \sigma_\theta(c)^2)$$

Too many NN forward passes for large K . To compute the marginal likelihood exactly, we must evaluate $p(x | z = c, \theta)$ for every discrete latent $c \in \{1, 2, \dots, K\}$, so run the NN *once for each mixture component*.

Continuous z , discrete x

$$p(x|\theta) = \int \mathcal{N}(z|0, I) \text{Cat}(x|\pi_\theta(z)) dz$$

To compute the marginal likelihood exactly, we need $p(x | z, \theta)$ to be evaluated at *every* real value of z that has nonzero probability under the prior, which would require *infinitely* many neural network forward passes.

Solution: Variational Inference

Evidence Lower Bound (ELBO)

$$\log p(x \mid \theta) \geq \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)] - \text{KL}(q_\phi(z \mid x) \| p(z)).$$

$$(\theta^*, \phi^*) = \arg \max_{\theta, \phi} [\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)] - \text{KL}(q_\phi(z \mid x) \| p(z))].$$

- $\text{KL}(q_\phi(z \mid x) \| p(z))$ is analytic for Gaussian q_ϕ, p .
- $\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)]$ is approximated via Monte Carlo.

Decoder (Generator) Network Gradient

$$\nabla_{\theta} \left[\mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x | z)] - \text{KL}(q_{\phi}(z | x) \| p(z)) \right].$$

KL does not depend on θ :

$$\nabla_{\theta} \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x | z)] = \mathbb{E}_{q_{\phi}(z|x)} [\nabla_{\theta} \log p_{\theta}(x | z)].$$

Monte Carlo approximation:

$$\nabla_{\theta} \approx \frac{1}{S} \sum_{i=1}^S \nabla_{\theta} \log p_{\theta}(x | z^{(i)}), \quad z^{(i)} \sim q_{\phi}(z | x).$$

Encoder (Inference) Network Gradient: Difficulty

We need:

$$\nabla_{\phi} \left(\mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x | z)] - \text{KL}(q_{\phi}(z | x) \| p(z)) \right).$$

KL is analytic. The difficult term is:

$$\nabla_{\phi} \int q_{\phi}(z | x) \log p_{\theta}(x | z) dz = \int (\nabla_{\phi} q_{\phi}(z | x)) \log p_{\theta}(x | z) dz.$$

But $\nabla_{\phi} q_{\phi}(z | x)$ is *not* a probability distribution:

- cannot sample from it,
- cannot build a Monte Carlo estimator.

Reparameterization Trick (Pathwise Estimator)

Rewrite sampling as a differentiable transformation:

$$z = g(\phi, \epsilon), \quad \epsilon \sim p(\epsilon) \text{ independent of } \phi.$$

Then the expectation becomes:

$$\mathbb{E}_{q_\phi(z|x)}[\cdot] = \mathbb{E}_{\epsilon \sim p(\epsilon)}[\cdot].$$

The encoder gradient becomes:

$$\nabla_\phi \mathbb{E}_\epsilon[f(g(\phi, \epsilon))] = \mathbb{E}_\epsilon[\nabla_z f(z) \nabla_\phi g(\phi, \epsilon)].$$

This is the *pathwise* (low-variance) estimator.

Gaussian Reparameterization

$$q_\phi(z | x) = \mathcal{N}(z; \mu_\phi(x), \sigma_\phi(x)^2)$$

Define noise:

$$\epsilon \sim \mathcal{N}(0, I), \quad z = g(\phi, \epsilon) = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon.$$

Gradients flow via:

$$\nabla_\phi z = \nabla_\phi \mu_\phi(x) + \epsilon \odot \nabla_\phi \sigma_\phi(x).$$

This makes sampling differentiable.

Inference Gradient: Final Pathwise Derivation

Using the reparameterization:

$$z_i = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, I).$$

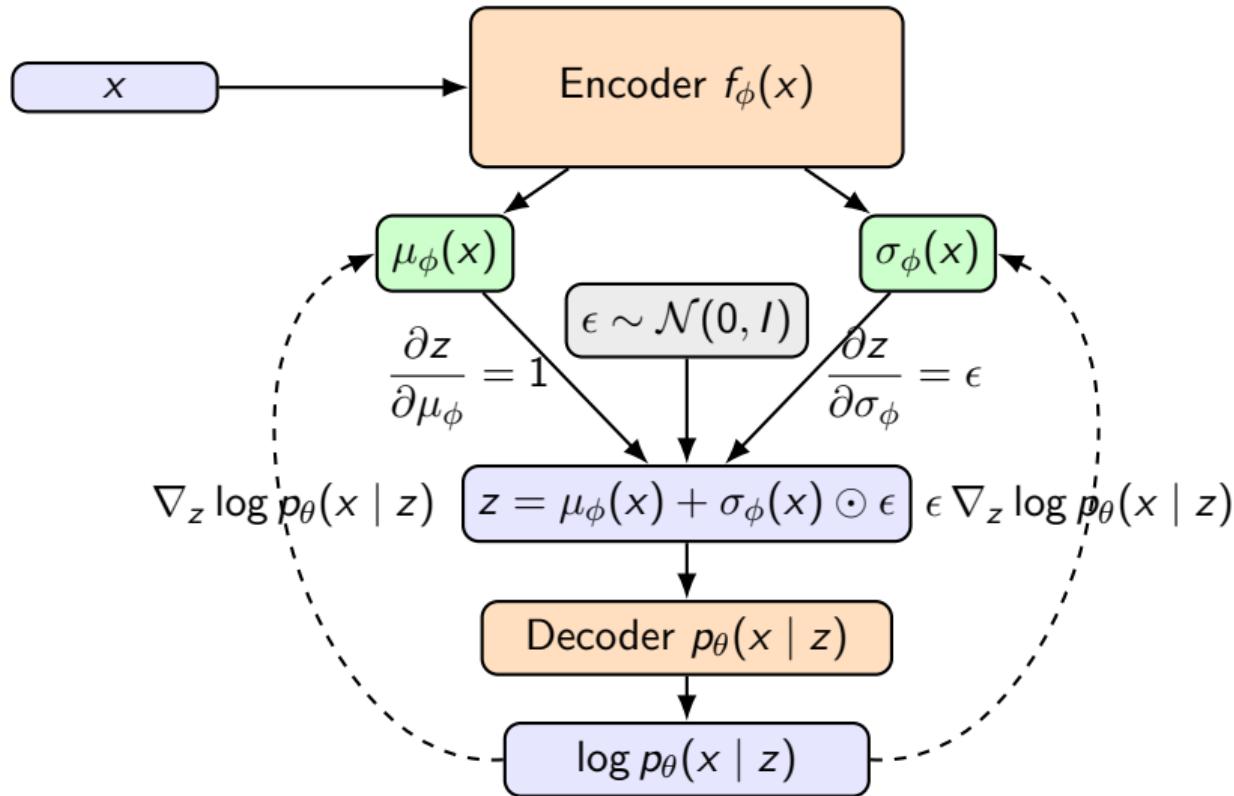
Two gradient paths:

$$\frac{\partial z}{\partial \mu_\phi} = 1, \quad \frac{\partial z}{\partial \sigma_\phi} = \epsilon.$$

Final Monte Carlo estimator:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x | z)] \approx \frac{1}{S} \sum_{i=1}^S \left[\nabla_z \log p_\theta(x | z_i) \left(\nabla_\phi \mu_\phi(x) + \epsilon_i \odot \nabla_\phi \sigma_\phi(x) \right) \right].$$

Forward vs. Backward Flow in the VAE (Reparameterization Trick)



Analytical KL for Gaussian Prior & Posterior

When $q_\phi(z|x) = \mathcal{N}(\mu, \text{diag}(\sigma^2))$ and $p(z) = \mathcal{N}(0, I)$, the KL is available in closed form — **no sampling needed for this term.**

Derivation (scalar case $d = 1$)

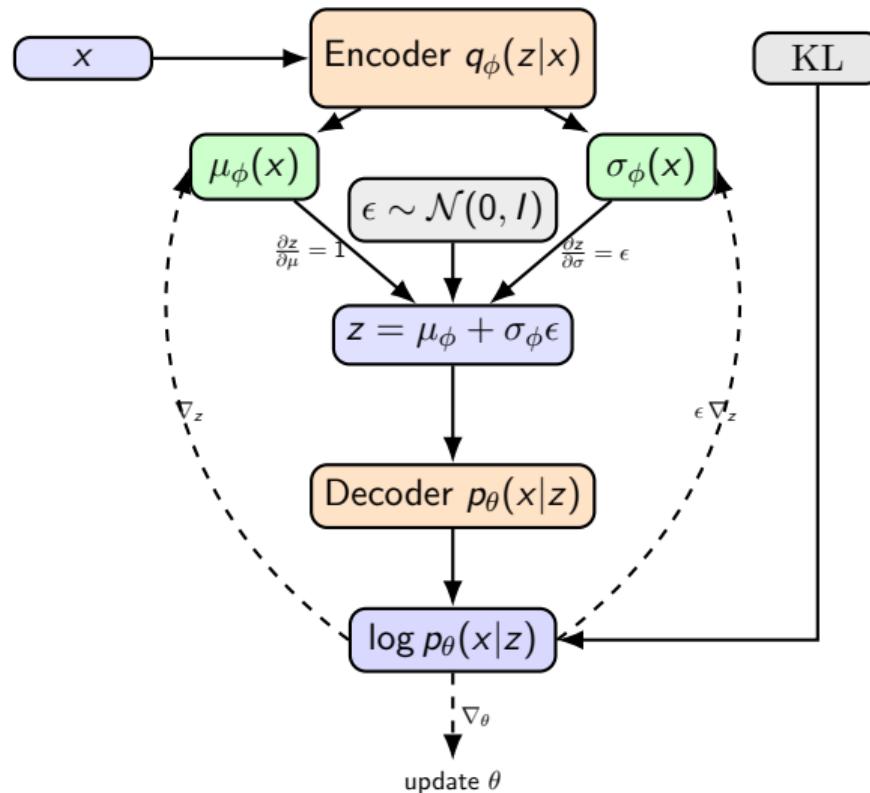
$$\begin{aligned} D_{\text{KL}}(q\|p) &= \int q(z) \log \frac{q(z)}{p(z)} dz = \mathbb{E}_q \left[\log \frac{\mathcal{N}(\mu, \sigma^2)}{\mathcal{N}(0, 1)} \right] \\ &= \mathbb{E}_q \left[-\frac{(z-\mu)^2}{2\sigma^2} + \frac{z^2}{2} + \frac{1}{2} \log \sigma^{-2} \right] = \frac{1}{2} (\sigma^2 + \mu^2 - 1 - \log \sigma^2) \end{aligned}$$

General d -dimensional diagonal Gaussian

$$D_{\text{KL}}(q_\phi\|p) = \frac{1}{2} \sum_{j=1}^d \left(\sigma_j^2 + \mu_j^2 - 1 - \log \sigma_j^2 \right) \geq 0$$

Sanity checks: $\mu=0, \sigma=1$: $D_{\text{KL}} = \frac{1}{2}(1+0-1-0) = 0$. ✓ $\mu=1, \sigma=1$: $D_{\text{KL}} = \frac{1}{2}(1+1-1-0) = \frac{1}{2}$. ✓

Compact Full VAE: Forward and Backward Pass



Training the VAE: AEVB / Reparameterization Algorithm

Algorithm (Kingma & Welling, 2013)

Inputs: dataset \mathcal{D} ; encoder $q_\phi(z|x)$; decoder $p_\theta(x|z)$

Initialize θ, ϕ randomly.

Repeat until convergence:

- ① Sample mini-batch $\mathcal{M} \subset \mathcal{D}$
- ② For each $x \in \mathcal{M}$:
 - ① Compute encoder outputs $(\mu_\phi(x), \sigma_\phi(x))$
 - ② Sample $\epsilon \sim \mathcal{N}(0, I)$
 - ③ Reparameterize $z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$
 - ④ Compute reconstruction log-likelihood $\log p_\theta(x|z)$ via the decoder
 - ⑤ Compute analytic KL: $\text{KL}(q_\phi(z|x) \| p(z))$
- ③ Form Monte Carlo ELBO estimate:

$$\tilde{\mathcal{L}}(x) = \log p_\theta(x|z) - \text{KL}(q_\phi(z|x) \| p(z))$$

- ④ Update parameters via stochastic gradient:

$$\theta \leftarrow \theta + \eta \nabla_\theta \tilde{\mathcal{L}}, \quad \phi \leftarrow \phi + \eta \nabla_\phi \tilde{\mathcal{L}}.$$

After Training: Inference with the Encoder

Goal: Map a new observation x^* into a latent representation.

Given a trained encoder $q_\phi(z | x)$:

$$q_\phi(z | x^*) = \mathcal{N}(z; \mu_\phi(x^*), \sigma_\phi^2(x^*)),$$

the encoder produces:

- **Mean** $\mu_\phi(x^*)$ (best estimate of latent location)
- **Standard deviation** $\sigma_\phi(x^*)$ (uncertainty)

Deterministic encoding (common in downstream tasks):

$$z^* = \mu_\phi(x^*) \quad (\text{no sampling})$$

Sampling from the posterior:

$$z^* = \mu_\phi(x^*) + \sigma_\phi(x^*) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

Interpretation: The encoder provides a distribution over latent codes for x^* , not just a point—capturing uncertainty about the underlying factors.

After Training: Inference with the Decoder

Goal: Generate new samples or reconstruct from latent codes.

Given a trained decoder $p_\theta(x | z)$:

Reconstruction of an input x^*

$$z^* = \mu_\phi(x^*) \quad \Rightarrow \quad \hat{x} = \mathbb{E}_{p_\theta(x|z^*)}[x] \approx g_\theta(z^*).$$

Used for denoising, compression, anomaly detection, etc.

Generation of new samples

Sample a latent point from the prior:

$$z \sim p(z) = \mathcal{N}(0, I)$$

and pass through the decoder:

$$x_{\text{new}} \sim p_\theta(x | z).$$

This produces *new* data consistent with the training distribution.

Interpretation: The decoder acts as a *learned generative model* mapping latent variables to data space according to $p_\theta(x | z)$.

Deterministic vs Probabilistic Inference After Training

Two different ways to use the encoder & decoder:

1. Deterministic Inference (Typical for downstream ML)

$$z^* = \mu_\phi(x^*), \quad \hat{x} = g_\theta(z^*).$$

Use cases: clustering, classification features, embeddings, dimensionality reduction.

2. Probabilistic Inference (True generative usage)

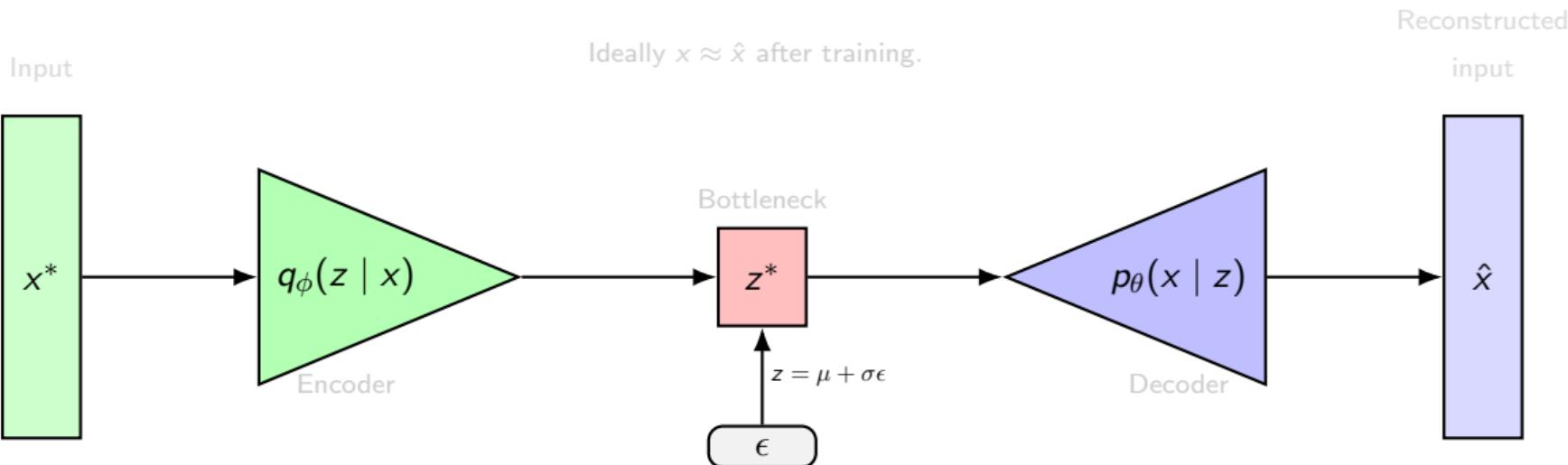
$$z^* = \mu_\phi(x^*) + \sigma_\phi(x^*) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

$$x_{\text{sample}} \sim p_\theta(x \mid z^*).$$

Use cases: uncertainty estimation, stochastic reconstructions, data augmentation.

Key idea: The VAE provides a *distribution* over latent codes, not a single point. You choose which mode fits the task.

Encoder–Decoder Flow After Training



Encoder: inference (features, uncertainty). Decoder: generation (reconstruction, new samples).

VAE for Time Series: Window-Based Approach

Many engineering signals are **multivariate time series**. We use a *window-based VAE*: segment the signal into fixed-length windows, treat each as a data point.

Generative model

- ① Sample operating state $z \sim \mathcal{N}(0, I)$
- ② Decode to window: $x | z \sim \mathcal{N}(\mu_\theta(z), \sigma^2 I)$
 μ_θ : CNN or MLP decoder

Inference model

$$(\mu_\phi, \log \sigma_\phi) = \text{Encoder}(x; \phi)$$
$$q_\phi(z|x) = \mathcal{N}(\mu_\phi, \text{diag}(\sigma_\phi)^2)$$

Applications

- **Anomaly detection**: train on healthy data; flag windows with low ELBO
- **Condition monitoring**: latent z encodes operating regime
- **Sensor fusion**: multivariate x compressed to low- d z
- **Missing-data imputation**: decode from z given partial x

Practical Concern: Posterior Collapse

Symptom

The encoder *ignores* the data: $q_\phi(z|x) \approx p(z)$ for all x .

The KL term vanishes (≈ 0) but reconstruction quality is poor.

Why it happens: at the start of training, the decoder is weak. The only way to minimise the KL is to make $q_\phi(z|x)$ match the prior — the model gets stuck in this local optimum.

β -VAE / KL Annealing

$$\mathcal{L}_\beta = \mathbb{E}_q[\log p_\theta(x|z)] - \beta \cdot D_{\text{KL}}(q_\phi \| p)$$

Start $\beta \approx 0$, anneal to $\beta = 1$ over training.

Lets the decoder learn first.

Free Bits

Ensure $\geq \lambda$ nats per latent dimension:

$$\mathcal{L}^{\text{fb}} = \mathbb{E}_q[\log p_\theta(x|z)] - \sum_j \max(\lambda, D_{\text{KL},j})$$

Forces the encoder to use each latent unit.

Practical Recommendations

Architecture

- **Encoder:** MLP or CNN; output $(\mu_\phi, \log \sigma_\phi)$ — use log-parameterisation to ensure $\sigma > 0$
- **Decoder:** match to data type — Gaussian MSE for continuous signals, Bernoulli BCE for binary
- **Latent dim:** start small (8–32); too large \Rightarrow posterior collapse

Training

- **Optimiser:** Adam ($\text{lr} \approx 10^{-3}$, decay to 10^{-4})
- **Batch size:** ≥ 64 ; larger batches \Rightarrow more stable KL estimates
- **KL annealing:** linear or cyclical schedule; avoid switching too fast
- **Warm-up:** 10–50 epochs of reconstruction-only training before enabling KL

Monitoring during training

Track **both** reconstruction loss and KL separately. Healthy training: KL rises gradually, reconstruction falls. Collapse sign: KL $\rightarrow 0$ and reconstruction stagnates.

Full Gaussian-VAE ELBO

Gaussian decoder assumption:

$$p_\theta(x | z) = \mathcal{N}(x; \hat{x}_\theta(z), \sigma^2 I).$$

ELBO:

$$\mathcal{L}(x; \theta, \phi) = \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x | z)] - \text{KL}(q_\phi(z | x) \| p(z)).$$

Gaussian log-likelihood:

$$\mathbb{E}_{q_\phi} [\log p_\theta(x | z)] = -\frac{1}{2\sigma^2} \mathbb{E}_{q_\phi} \|x - \hat{x}_\theta(z)\|^2 + \text{const.}$$

KL term:

$$\text{KL}(q_\phi(z | x) \| p(z)) = \frac{1}{2} \sum_i (\mu_i^2 + \sigma_i^2 - 1 - \log \sigma_i^2).$$

Final loss (minimized in practice):

$$\boxed{\mathcal{L}(x) = \|x - \hat{x}_\theta(z)\|^2 + \text{KL}(q_\phi(z | x) \| p(z))}$$

or more generally (-VAE):

$$\boxed{\mathcal{L}(x) = \|x - \hat{x}_\theta(z)\|^2 + \beta \text{KL}(q_\phi(z | x) \| p(z))}.$$

VAE: Advantages & Limitations

Advantages

- End-to-end backprop (single ELBO objective)
- Fast amortised inference at test time
- Structured, continuous latent space
- Works for any data type with a suitable decoder
- Principled anomaly scoring via ELBO

Limitations

- ELBO is only a *lower bound* on $\log p(x)$
- Posterior collapse (decoder too powerful)
- Gaussian posterior may be too simple (use normalising flows for richer q)
- Generated samples often blurry (mode averaging)

Rule of thumb for engineering applications

If you need **anomaly detection** or **representation learning** on sensor data: VAE is an excellent baseline. If you need **photo-realistic generation**: consider more advanced models.

Why Discrete Latent Variables?

Some phenomena are *inherently discrete*:

Applications

- **Topic models:** document → topic assignment (integer)
- **Machine translation:** alignment between words (index)
- **Morphological inflection:** morphological tag (category)
- **Vision:** object category, part segmentation (label)

The problem with reparametrisation

Reparametrisation requires computing a Jacobian $J_{ij} = \partial h_i / \partial z_j$.

For discrete variables the outcome space is **non-continuous** ⇒ derivatives are not defined.

Score Function Estimator (REINFORCE)

For discrete latent variables z , start from the ELBO gradient:

$$\frac{\partial \mathcal{L}}{\partial \phi} = \sum_z \frac{\partial}{\partial \phi} q(z|\phi) \cdot \log p_\theta(x|z).$$

Log-derivative trick:

$$\frac{\partial}{\partial \phi} q(z|\phi) = q(z|\phi) \frac{\partial}{\partial \phi} \log q(z|\phi).$$

Score-function (REINFORCE) estimator

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \phi} &= \mathbb{E}_{q(z|\phi)} \left[\frac{\partial}{\partial \phi} \log q(z|\phi) \cdot \log p_\theta(x|z) \right]. \\ &\approx \frac{1}{S} \sum_{i=1}^S \left[\frac{\partial}{\partial \phi} \log q(z_i|\phi) \cdot \log p_\theta(x|z_i) \right], \quad z_i \sim q(\cdot|\phi).\end{aligned}$$

Key issue

The score-function estimator is **unbiased** but has **very high variance**. Variance reduction is essential.

Comparison: Two Gradient Estimators

Reparameterization Estimator

$$z = g(\phi, \epsilon), \quad \epsilon \sim p(\epsilon)$$

$$\nabla_\phi \mathcal{L} = \mathbb{E}_\epsilon [\nabla_z \log p_\theta(x|z) \nabla_\phi g(\phi, \epsilon)].$$

Requires continuous differentiable z . Variance:
low.

Score Function Estimator

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{q(z|\phi)} [\nabla_\phi \log q(z|\phi) \log p_\theta(x|z)].$$

Works for discrete or continuous z . Variance:
high.

Variance Reduction: Baselines

Key identity:

$$\mathbb{E}_{q(z|\phi)} [\nabla_{\phi} \log q(z|\phi)] = 0.$$

Baseline subtraction

For any constant or learned function $C(x)$:

$$\mathbb{E}_q [\nabla_{\phi} \log q(z|\phi) (\log p_{\theta}(x|z) - C(x))] = \mathbb{E}_q [\nabla_{\phi} \log q(z|\phi) \log p_{\theta}(x|z)].$$

Expectation unchanged, variance reduced.

Learning the baseline

Train a regression network $C(x; \omega)$:

$$\min_{\omega} (C(x; \omega) - \log p_{\theta}(x|z))^2.$$

Acts as a *value function* (as in REINFORCE).

Example: Discrete Latent Factor Model

Binary latent factors:

$$z_j \sim \text{Ber}(\phi), \quad j = 1, \dots, k,$$

$$x_i \mid z \sim \text{Cat}(g(z)), \quad g(\cdot) \text{ a neural network.}$$

Mean-field posterior

$$q(z|x, \phi) = \prod_{j=1}^k \text{Ber}(z_j \mid \psi_j(x, \phi)).$$

Independent Bernoulli factors for tractability.

Score-function ELBO gradient

Sample $z \sim q(z|x, \phi)$

Compute $\ell = \log p_\theta(x|z)$

Update:

$$\nabla_\phi \mathcal{L} \approx \nabla_\phi \log q(z|\phi) \cdot (\ell - C(x)).$$

Summary

What we covered

- ① **NN as distribution parametriser** – NNs map inputs to parameters of probabilistic models.
- ② **Deep Generative Models** – combine prior, NN likelihood, latent variables; marginalisation is intractable.
- ③ **MLE & SGD** – tractable likelihoods admit backprop + SGD.
- ④ **ELBO** – variational lower bound; reconstruct data while regularising the posterior.
- ⑤ **VAE** – reparametrisation trick enables end-to-end backprop for continuous latent variables.
- ⑥ **Discrete VAEs** – score function estimator works universally but needs variance reduction (baselines).

Further reading: Kingma & Welling (2013); Rezende et al. (2014); Williams (1992); Tucker et al. (2017)

References I



Bank, Dor, Noam Koenigstein, and Raja Giryes (2020). *Autoencoders*. arXiv:2003.05991.