Generative Modeling — A Background Crash Course

Preparation for Lecture 15 (Normalizing Flows)

Prof. Young Woo Choi

Department of Physics, Sogang University

PHYG004 / PHY5006 — 2026 Spring

Goal of this deck: build up the language and the ELBO so the L15 hands-on makes sense.

5/26/2026

What is a generative model?

A **generative model** learns a distribution $p_\theta(x)$ from samples $x \sim p_{\text{data}}$ so that we can:

Sample

Draw new examples $x' \sim p_\theta(x)$.

Examples:

- new molecular conformations
- new images of cats
- new text completions
- new lattice gauge configurations

Evaluate the density

Compute $p_\theta(x)$ at a given x.

Useful for:

- anomaly detection
- importance reweighting
- free-energy estimation
- model comparison

Different model families specialise in different subsets of these.

Generative vs Discriminative

So far in this course (L01–L14) we've mostly built **discriminative** models.

	Discriminative	Generative
Learns	$p_\theta(y \mid x)$	$p_\theta(x)$
Answers	"Given x, what is y?"	"What kinds of x exist?"
Examples	classifier, regressor, force field	LLM, diffusion image model, AlphaFold-style structure generator
Loss	NLL on labels	NLL on x

> Today's bridge: a generator can be *seen* as a discriminator with $y = x$ — but the inductive biases are completely different.

The big picture in one equation

We want a parametric distribution $p_\theta(x)$ such that

$$p_\theta \;\approx\; p_{\text{data}}.$$

The natural distance is the **Kullback–Leibler divergence**:

$$D_{\text{KL}}(p_{\text{data}} \,\|\, p_\theta) \;=\; \mathbb{E}_{x\sim p_{\text{data}}}\left[\log p_{\text{data}}(x) - \log p_\theta(x)\right].$$

Since $\log p_{\text{data}}$ does not depend on θ, **minimising forward KL** is equivalent to **maximising the log-likelihood**:

$$\arg\min_\theta \; D_{\text{KL}}(p_{\text{data}} \,\|\, p_\theta) \;=\; \arg\max_\theta \; \mathbb{E}_{x\sim p_{\text{data}}}\left[\log p_\theta(x)\right].$$

All five generative model families differ in **how they parametrise** $\log p_\theta(x)$.

Why naive density estimation fails

Why can't we just fit a histogram or KDE to $p_\theta(x)$?

Histogram

Partition $\mathbb{R}^d$ into K bins per axis.

Need K^d samples per bin. For $d = 784$ (MNIST), K^{784} is hopeless.

Kernel density estimation

$\hat{p}(x) = \frac{1}{N} \sum_i K_h(x - x_i)$

Needs all data at inference. No generative mechanism.

Mixture of Gaussians

$p(x) = \sum_k \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$

Tractable but cannot capture image-like manifolds.

Energy-based

$p_\theta(x) = \frac{1}{Z(\theta)} e^{-E_\theta(x)}$

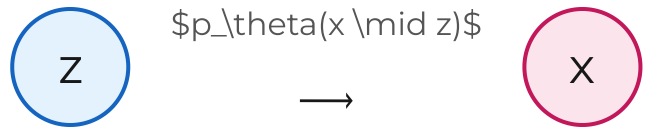
Partition function $Z(\theta)$ is intractable.

The way out: introduce auxiliary **latent variables** z.

Latent variable models

Postulate a low-dimensional **latent** $z \in \mathbb{R}^{d_z}$ and a generative process:

$$z \sim p(z), \qquad x \sim p_\theta(x \mid z).$$



The **marginal likelihood** of the data is an integral:

$$\boxed{p_\theta(x) \;=\; \int p_\theta(x \mid z)\, p(z)\, dz.}$$

Physics intuition. Think of z as collective coordinates / order parameters / reaction coordinates — slow degrees of freedom that explain the data.

The integral is intractable

The integral

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

is **not analytically tractable** for general neural-network $p_\theta(x \mid z)$.

Two classical workarounds:

MCMC

Sample $z \sim p_\theta(z \mid x)$ via a Markov chain.

✓ Asymptotically exact ✕ Slow, hard to scale, no gradient signal

Variational inference

Replace $p_\theta(z \mid x)$ with a **learned, tractable** approximation $q_\phi(z \mid x)$.

✓ Fast, gradient-friendly ✕ Approximate (introduces a "gap")

Variational inference — the setup

Introduce a parametric **approximate posterior** $q_\phi(z \mid x)$ — the "encoder" or "inference network".

For now, treat it as any distribution. We will derive a lower bound on $\log p_\theta(x)$ that:

1. Uses only $q_\phi(z \mid x)$ (sampleable), $p(z)$ (known), $p_\theta(x \mid z)$ (the decoder).
2. Avoids the intractable integral.
3. Becomes **exact** when $q_\phi(z \mid x) = p_\theta(z \mid x)$.

The bound is called the **Evidence Lower Bound (ELBO)**. It is the central object of variational inference — and the most important formula you must understand for Lecture 15.

ELBO — derivation 1 (Jensen)

Start from the marginal $p_\theta(x) = \int p_\theta(x, z)\, dz$. Multiply and divide by $q_\phi(z \mid x)$:

$$p_\theta(x) \;=\; \int q_\phi(z \mid x) \cdot \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\, dz \;=\; \mathbb{E}_{z \sim q_\phi(z|x)}\left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right].$$

Take the log of both sides. Since $\log$ is **concave**, **Jensen's inequality** gives

$$\log p_\theta(x) \;=\; \log \mathbb{E}_{q_\phi}\left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right] \;\geq\; \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right].$$

$$\boxed{\mathcal{L}(\theta, \phi;\, x) \;\equiv\; \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \;\leq\; \log p_\theta(x).}$$

The lower-bound term on the left is the **ELBO**.

ELBO — derivation 2 (KL identity)

Same identity, different angle. Bayes: $p_\theta(x, z) = p_\theta(z \mid x)\, p_\theta(x)$. Take the expectation under q_ϕ:

$$\log p_\theta(x) \;=\; \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right] \;+\; \underbrace{\mathbb{E}_{q_\phi}\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right]}_{D_{\mathrm{KL}}(q_\phi \,\|\, p_\theta(\cdot|x)) \,\geq\, 0}$$

$$\boxed{\log p_\theta(x) \;=\; \mathcal{L}(\theta, \phi; x) \;+\; D_{\mathrm{KL}}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x))\,.}$$

Two things at once: maximising the ELBO simultaneously

- pushes $\log p_\theta(x)$ **up** (better generative model)
- pulls $q_\phi(z \mid x)$ **towards** $p_\theta(z \mid x)$ (better inference network)

Equality iff the approximate posterior equals the true posterior.

ELBO — the form we optimise

Decompose the joint $p_\theta(x, z) = p_\theta(x \mid z)\, p(z)$:

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

Reconstruction term

Encourages z to retain enough information about x that the decoder can re-create it.

(For binarized MNIST: per-pixel BCE.)

KL term

Pulls $q_\phi(z \mid x)$ towards the **prior** $p(z) = \mathcal{N}(0, I)$ so that we can sample at test time.

For Gaussian q and p, this has a **closed form**.

Training = maximise ELBO = simultaneously fit data and stay close to the prior. Two competing forces.

Why we cannot just sample inside the network

The reconstruction term

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

requires sampling z from a distribution whose parameters depend on ϕ. Naively, that breaks differentiability:

$$\frac{\partial}{\partial \phi} \mathbb{E}_{z \sim q_\phi}[f(z)] \neq \mathbb{E}_{z \sim q_\phi}\left[\frac{\partial f}{\partial \phi}\right].$$

The expectation operator and the gradient operator **do not commute** when ϕ appears inside the sampler.

A REINFORCE-style score-function estimator works but has very high variance.

Solution: the **reparameterization trick**.

The reparameterization trick

For a Gaussian $q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \operatorname{diag} \sigma_\phi^2(x))$, rewrite a sample as

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

Now ϕ is **outside** the sampler. Gradients flow through μ_ϕ and σ_ϕ via standard backprop:

$$\nabla_\phi \, \mathbb{E}_{q_\phi}[f(z)] \;=\; \mathbb{E}_\epsilon\big[\nabla_\phi \, f\big(\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon\big)\big] \,.$$

> The randomness (ϵ) is shunted to a **parameter-free auxiliary variable**. The learned distribution is recovered by the deterministic affine map.

Putting it together — the VAE (L14)

$$-\mathcal{L}(\theta, \phi;\, x) \;=\; \underbrace{-\log p_\theta(x \mid z)}_{\text{BCE / Gaussian NLL}} \;+\; \underbrace{\tfrac{1}{2}\sum_i \left(\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1\right)}_{\text{closed-form KL to } \mathcal{N}(0,I)},$$

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

Training loop

1. Encode: $(\mu, \log \sigma) = \mathrm{Enc}_\phi(x)$.
2. Sample noise: $\epsilon \sim \mathcal{N}(0, I)$; form z.
3. Decode: $\hat{x} = \mathrm{Dec}_\theta(z)$.
4. Compute $-\mathcal{L}$; backprop on (θ, ϕ).

Generation

$z \sim \mathcal{N}(0, I), \quad x \sim p_\theta(x \mid z).$

Forward vs Reverse KL — a critical distinction

Two ways to "measure distance" between p and q. They are **not symmetric**.

Forward KL $D_{\mathrm{KL}}(p \,\|\, q)$

$$\mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$$

- Penalised wherever $p > 0$ but $q \approx 0$
- **Mode-covering** — model puts mass on **all** modes
- Used in **MLE training with data** (Section 7 of notebook)

Reverse KL $D_{\mathrm{KL}}(q \,\|\, p)$

$$\mathbb{E}_{x\sim q}\left[\log \frac{q(x)}{p(x)}\right]$$

- Penalised wherever $q > 0$ but $p \approx 0$
- **Mode-seeking** — model may concentrate on one mode
- Used in **Boltzmann generators** (data-free training)

The same KL definition; different argument order; very different behaviour.

The five families

Family	$\log p_\theta(x)$?	Sampling	Killer property
VAE	lower bound (ELBO)	fast (1 pass)	fast inference, interpretable latent
Normalizing flow	**exact**	fast (1 pass)	exact density + exact sampling
Diffusion	bound	slow (T steps)	SOTA images, molecules, materials
GAN	none (implicit)	fast	sharp samples, no likelihood
Autoregressive	exact	slow (sequential)	text

Three of these — VAE, flow, diffusion — are deeply related.

> A flow can be viewed as a deterministic, invertible latent-variable model with no variational gap. A diffusion model is closely related to a deep hierarchical VAE in the $T \to \infty$ limit.

Where flows fit (preview of L15)

A normalizing flow chooses $p_\theta(x \mid z)$ to be a **deterministic, invertible map** $x = f_\theta(z)$. The change-of-variables formula gives:

$$\log p_\theta(x) \;=\; \log p_Z\big(f_\theta^{-1}(x)\big) \;+\; \log \left|\det J_{f_\theta^{-1}}(x)\right|.$$

No ELBO. The integral disappears — there is no marginalisation over z because z is a deterministic function of x.

The price: f_θ must be a **diffeomorphism** with a **tractable Jacobian determinant**. The architecture design of flows (NICE, RealNVP, Glow, neural splines…) is one long answer to: *"how do we build such an f for high-dim x?"*

Physics motivations — why we care

Sample equilibrium states

$p(x) \propto e^{-U(x)/k_B T}$

Boltzmann generators (Noé *Science* 2019), lattice QCD (Albergo–Kanwar–Shanahan PRD 2019).

Inverse design

Generate crystals / molecules conditioned on a property.

CDVAE (ICLR 2022), MatterGen (Nature 2025).

Surrogate simulators

Replace expensive Monte Carlo or PDE solvers.

CaloGAN, CaloDiffusion (HEP detector simulation).

Order parameters

Latent axes that align with magnetisation, reaction coordinates, *etc.*

Wetzel PRE 2017, RAVE for MD.

Pop quiz — check your understanding

1. **What is the integral that defines $p_\theta(x)$ in a latent-variable model? Why is it hard?**
2. **State the ELBO. Why is it a lower bound on $\log p_\theta(x)$?**
3. **What is the variational gap, and when does it close?**
4. **Why do we need the reparameterization trick?**
5. **Why is forward KL "mode-covering" and reverse KL "mode-seeking"? Which one does maximum-likelihood training use?**
6. **In what sense is a normalizing flow "a VAE with the variational gap closed by construction"?**

If you can answer 3+ of these comfortably, you're ready for L15.

Today's lecture (L15)

We will:

1. Derive the **change of variables** for densities (the physicist's Jacobian, but for probability).
2. Build a **RealNVP** flow in JAX with affine coupling layers.
3. Train it on 2D toy data (two moons, rings) — **exact log-likelihoods**, no ELBO.
4. Convert it into a **Boltzmann generator** for a double-well potential — training **without data**, using only the energy.
5. Compare with MCMC (which gets trapped) and with the VAE (which can compress, but only has a lower bound).

Open the notebook: `handson.ipynb`.

> The whole point of these slides: when we write $\log p_\theta(x)$, you should now know that for a flow it is *exact*, for a VAE it is a *bound*, and that the bound has a name (ELBO) and a derivation (Jensen).

References (background)

Foundational

- Kingma & Welling, *Auto-Encoding Variational Bayes*, ICLR 2014. arXiv:1312.6114
- Rezende, Mohamed & Wierstra, *Stochastic Backpropagation and Approximate Inference*, ICML 2014. arXiv:1401.4082
- Kingma & Welling, *An Introduction to Variational Autoencoders*, Found. Trends ML 2019.

Pedagogical

- Doersch, *Tutorial on Variational Autoencoders*, arXiv:1606.05908 (2016).
- Tomczak, *Deep Generative Modeling*, Springer (2022).

Physics applications

- Noé *et al.*, *Boltzmann Generators*, Science 365, eaaw1147 (2019).
- Wetzel, *Unsupervised learning of phase transitions...*, Phys. Rev. E 96, 022140 (2017).
- Cranmer, Brehmer & Louppe, *The frontier of simulation-based inference*, PNAS 117, 30055 (2020).

Questions?

Now open `handson.ipynb` and let's build a flow.

5/26/2026