From AutoEncoders to VQ-VAE

Part 1 • Cryptographic origins → AutoEncoder → Helmholtz Machine

— A history of the mathematical modelling that precedes the VAE

cryptography → encoder/decoder → AutoEncoder → Helmholtz Machine → (VAE → VQ-VAE)

The single question behind every model in this course

Overview

The question

"How do we design a model that can create a new picture of a cat?"

- Forty years ago, when Mario and Tetris were cutting-edge graphics, mathematicians, statisticians and engineers had to build the theory to answer this — largely by imagination.
- They borrowed ideas from **cryptography**, **statistical physics** and **biology**; at the time ML was not yet a separate, clean discipline.
- The AutoEncoder, whose input equals its target (**auto-association**), became one of the most deeply analysed models, because that structure is mathematically clean.
- Knowing *which purpose and philosophy* a technique came from is what tells you *where* to apply it correctly — "latent vector" is not a magic, all-purpose object.

The flow of the course — two research traditions that merge

Historical genealogy

Common root (1989)

- Linear AE = PCA (Baldi–Hornik)

Statistics lineage

- make PCA probabilistic → **PPCA** (Tipping–Bishop)

Neural-network lineage

- Hinton–Zemel (autoencoders, MDL, free energy)
- **Helmholtz Machine** (multilayer + wake–sleep)
- Deep AE (nonlinear expressive power)

Confluence (2014 → 2017)

- **VAE** (Kingma–Welling): unifies Helmholtz wake–sleep into a single **ELBO** + reparameterization
- **VQ-VAE** (van den Oord): discrete codebook

One-line summary

The same *recognition–generation pair*, advanced by turning the encoder from "*derived*" into "*learned*."

What Part 1 covers (everything before the VAE)

Learning objectives

- the encoder–decoder structure and **auto-association**, and how ML reshaped these classical terms (Ch. 1)
- the definition of the AutoEncoder and its **six components** — the frame of reference for the whole course (Ch. 2)
- tools for modelling: the **norm**, and Gaussian / Bernoulli losses (Ch. 2)
- **Linear AE**: a full proof that its global optimum is the PCA subspace (Baldi–Hornik) (Ch. 3)
- **PPCA**: the probabilistic model of the Linear AE — the encoder is *derived* by Bayes (Ch. 3)
- **Nonlinear / Deep AE**: why nonlinearity (manifolds), staged training (Ch. 3)
- **Helmholtz Machine**: free energy, the variational bound, wake–sleep $\rightarrow$ the bridge to VAE (Ch. 3)
- why the AutoEncoder **fails as a generative model** — the starting point of the VAE (Ch. 3)

Outline

Cryptographic origins

The AutoEncoder: definition and components

A history of mathematical modelling

VAE: concept and structure

VAE: the variational lower bound

VAE: optimization

VAE: architecture, characteristics, code

PART 1 • pre-VAE

Cryptographic origins

The ML words “encoder” and “decoder” are pre-neural-network terms.

Problem setting — communication and ciphers

Wartime radio: the enemy intercepts the signal. Send plaintext and they get the same information.
$\Rightarrow$ the sender transforms the message by an agreed rule; the receiver applies the inverse rule (e.g. the Caesar cipher).

Definition 1.1 • encoder and decoder

$$E:\ \text{plaintext} \to \text{ciphertext}, \qquad D:\ \text{ciphertext} \to \text{plaintext}, \qquad D \circ E = \mathrm{id}.$$

Generalising encoder–decoder

Classical cryptography / JPEG

- a *hand-designed* function
- Caesar cipher; DCT, quantization tables, Huffman coding
- explicit, invertible rules

Machine learning

- **learn** the function *itself* from data
- a neural network is a learnable function family (universal approximator)
- for complex natural data (a cat photo) humans cannot hand-design a good rule

Two things ML changed

(1) the function is **learned**, not hand-designed. (2) exact recovery ($D \circ E = \mathrm{id}$) is dropped in favour of **lossy** coding — designed so that what is lost is the "meaningless" part.

PART 1 • pre-VAE

The AutoEncoder: definition and components

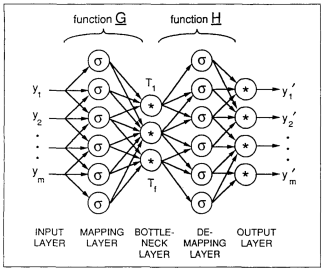


Figure 2. Network architecture for simultaneous determination of f nonlinear factors using an autoassociative network.
σ indicates sigmoidal nodes, $*$ indicates sigmoidal or linear nodes.

Fig 1. Autoassociative network: G = recognition, H = generation, the bottleneck $T_1, \ldots, T_f$ is the latent vector.

latent vector $z \in \mathbb{R}^p$ $(p < d)$: the low-dimensional code of the input.

latent space $\mathcal{Z} \subseteq \mathbb{R}^p$: where the z live.

recognition model (encoder, $\mathcal{X} \to \mathcal{Z}$): "recognises" the latent cause of the data.

generation model (decoder, $\mathcal{Z} \to \mathcal{X}$): "generates" data from the latent.

Training objective (preview)

$$\log p_\theta(x) \;\geq\; \mathbb{E}_{q_\phi}[\log p_\theta(x \mid z)] - D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big)$$

Definition 2.1 • AutoEncoder

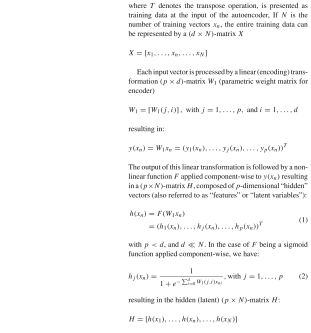
A neural network of an encoder $f_\phi : \mathbb{R}^D \to \mathbb{R}^d$ $(d < D)$ and a decoder $g_\theta : \mathbb{R}^d \to \mathbb{R}^D$. It encodes $z = f_\phi(x)$ and reconstructs $\hat{x} = g_\theta(z)$, minimising the input–reconstruction gap.

(1) auto-association

- the input x is its own target $(y = x)$
- input and target share one distribution $\Rightarrow$ the loss is easy to analyse

(2) information bottleneck

- force a middle layer with $d < D$
- $\text{rank}(W_d W_e) \leq d \Rightarrow$ cannot be the identity
- the model *must* exploit statistical dependence among pixels



where T denotes the transpose operation, is presented as training data at the input of the autoencoder. If N is the number of training vectors x_n, the entire training data can be represented by a $(d \times N)$-matrix X

$$X = [x_1, \ldots, x_n, \ldots, x_N]$$

Each input vector is processed by a linear (encoding) transformation ($p \times d$)-matrix W_1 (parametric weight matrix for encoder)

$$W_1 = [W_1(j,i)], \text{ with } j = 1, \ldots, p, \text{ and } i = 1, \ldots, d$$

resulting in:

$$y(x_n) = W_1 x_n = (y_1(x_n), \ldots, y_j(x_n), \ldots, y_p(x_n))^T$$

The output of this linear transformation is followed by a non-linear function F applied component-wise to $y(x_n)$ resulting in a $(p \times N)$-matrix H, composed of p-dimensional "hidden" vectors (also referred to as "features" or "latent variables"):

$$h(x_n) = F(W_1 x_n) = (h_1(x_n), \ldots, h_j(x_n), \ldots, h_p(x_n))^T \quad (1)$$

with $p < d$, and $d \ll N$. In the case of F being a sigmoid function applied component-wise, we have:

$$h_j(x_n) = \frac{1}{1 + e^{-\sum_{i=0}^{d} W_1(j,i) x_{ni}}}, \text{ with } j = 1, \ldots, p \quad (2)$$

resulting in the hidden (latent) $(p \times N)$-matrix H:

$$H = [h(x_1), \ldots, h(x_n), \ldots, h(x_N)]$$

Fig 2. A narrow passage compresses information → latent vector / latent space.

$$d < D$$

- to write 784 pixels with 32 reals, you cannot store pixels independently — you must use their **statistical dependence**.
- this constraint forces the model to learn "how the data is generated."
- **latent code** (information theory) = **latent vector** (linear algebra): one object, two viewpoints.

Information-theoretic view

Shannon's source-coding bound $H(X)$ is the lossless floor; representation learning does *lossy* coding below it, discarding "meaningless" bits.

Why the bottleneck has an information-theoretic meaning

KL divergence — how far p is from a reference q

$$D_{\mathrm{KL}}(p \,\|\, q) = \sum_{w} p(w) \log \frac{p(w)}{q(w)} \;\geq 0, \qquad = 0 \iff p = q.$$

Asymmetric ($D_{\mathrm{KL}}(p\|q) \neq D_{\mathrm{KL}}(q\|p)$): the outer average is over p only. *Not* a norm — a norm sizes one vector; KL compares two distributions.

Mutual information = a KL divergence

$$I(X;T) = \sum_{x,t} p(x,t) \log \frac{p(x,t)}{p(x)p(t)} = D_{\mathrm{KL}}\big(p(x,t) \,\|\, p(x)p(t)\big)$$

distance between the *true joint* and the *independence* assumption; $I = 0 \iff X \perp T$.

Entropy form

$$I(X;T) = \sum_{x,t} p(x,t) \log \frac{p(x \mid t)}{p(x)} = H(X) - H(X \mid T)$$

The information-bottleneck objective

Find a compressed representation T of X that *reduces* $I(X;T)$ (compression) while *keeping* the relevant information $I(T;Y)$.

- In an AutoEncoder we let $X =$ data, $T = z$ the latent, and implement the bottleneck simply by $\dim z = d < D$.
- The compression forces the representation to capture the data's essential structure.

Definition 2.2 — every later model re-fills these six cells

component	deterministic form	probabilistic form
auto-association	the input is its own target $y = x$ — input and target share one distribution	
recognition (encoder)	$z = f_\phi(x) = F(W_1 x)$	$q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x) I)$
generation (decoder)	$\hat{x} = g_\theta(z) = W_2 z$	$p_\theta(x \mid z)$ (Gaussian / Bernoulli)
activation F	identity $\Rightarrow$ Linear AE sigmoid/tanh/ReLU $\Rightarrow$ Nonlinear AE	
latent vector	$z \in \mathbb{R}^p$, $z_j = F(\sum_i W_1(j,i) x_i)$	$z \sim q_\phi(z \mid x)$ (a random variable, not a point)
latent space	$\mathcal{Z} \subseteq \mathbb{R}^p$ $(p < d)$	a prior $p(z) = \mathcal{N}(0, I)$ on $\mathcal{Z}$

Modelling splits into the **deterministic** (functions/matrices) vs. the **probabilistic** (distributions) column.

Read the whole course as filling this one map

	Linear AE	**PPCA**	**Nonlinear AE**	**Helmholtz**
recognition	$z = W_e x$	$p(z \mid x)$ *derived* (Bayes)	$z = G_\phi(x)$	$q_\phi(z \mid x)$ *learned*
generation	$\hat{x} = W_d z$	$\mathcal{N}(Wz + \mu, \sigma^2 I)$	$\hat{x} = H_\theta(z)$	$p_\theta(x \mid z)$
activation	identity	identity	sigmoid/tanh/ReLU	factorial sigmoid
latent	point z	Gaussian posterior	point z	binary stochastic $z^{1..L}$
how solved	closed form (PCA)	closed form / EM	staged training	wake–sleep / free energy

The decisive move across the course: turn the **recognition** cell from a fixed/derived map into a *learned* network — the rest follows.

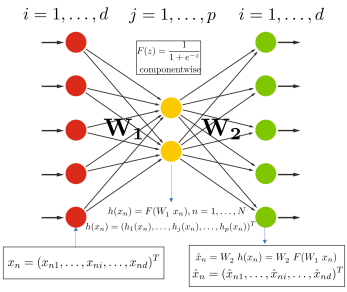


Fig. 1 Shallow undercomplete autoencoder

Fig 3. Shallow undercomplete AE: W_1 = recognition f, W_2 = generation g; the narrow middle layer is the latent vector.

- 784 input pixels are compressed to 32 and reconstructed back to 784.
- the narrow middle (**information bottleneck**) forces the model to learn statistical dependence.
- each hidden node may carry a sigmoid or a linear activation; the activated nodes $T_1, \ldots, T_f$ form one latent vector.
- a latent vector is both a code and a set of coefficients over the decoder's basis vectors.

A standard AutoEncoder is deterministic

The encoder gives $z = f_\phi(x)$ and the decoder $\hat{x} = g_\theta(z)$; feeding the same x twice yields the same $\hat{x}$. The whole map $x \mapsto \hat{x}$ is a fixed function.

- training only pushes $\hat{x} \approx x$, measured by the Frobenius norm.
- a closed-form solution exists (next chapter); otherwise gradients flow back through decoder $\rightarrow$ latent $\rightarrow$ encoder by backpropagation.

Foreshadowing

This deterministic property is exactly what makes the AutoEncoder *fail* as a generative model (§3.5): a single point per input, nothing defined between the points.

Modelling tool (1): norm

- A norm *measures error*: the size of the difference between two vectors/matrices.

vector L^2 norm

$$\|v\|_2 = \sqrt{\textstyle\sum_j v_j^2}, \quad \|x - \hat{x}\|_2^2 = \sum_d (x_d - \hat{x}_d)^2$$

matrix norms (clash of notation)

$$\|A\|_F^2 = \sum_{ij} a_{ij}^2 = \operatorname{tr}(A^\top A)$$

$$\|A\|_2 = \sigma_{\max}(A) \text{ (spectral)}$$

Frobenius = total element energy; spectral = worst-direction stretch — different quantities. Reconstruction loss is element-wise, so **Frobenius** fits. Also $\|A\|_F = \|\operatorname{vec}(A)\|_2$ (isometric isomorphism), which is why L^2 and $\|\cdot\|_F$ are used interchangeably.

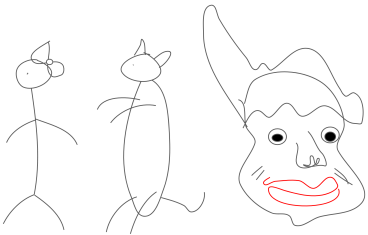


Fig 4. Trained on image 1, which of 2 / 3 does a Frobenius-trained model call "closer" to 1?

AutoEncoder reconstruction loss

$$\mathcal{L}_{\mathrm{AE}}(\phi,\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| x_i - g_\theta(f_\phi(x_i)) \right\|_2^2$$

- "error between two images" is not a uniquely-defined notion; modelling means linking a *plausible* structure, not a perfect one.
- a per-pixel comparison can judge a sketch of a cat "closer" to a person than another person.
- the trained map is **deterministic**: same $x \Rightarrow$ same $\hat{x}$. (This is exactly what breaks generation, §3.5.)

Modelling tool (2): read $\hat{x}$ as the parameter of $p(x \mid \hat{x})$

Continuous data: assume a Gaussian, $\hat{x}$ = its mean

$$p(x \mid \hat{x}) = \prod_{d=1}^{D} \mathcal{N}(x_d \mid \hat{x}_d, \sigma^2), \qquad \hat{x} = g_\theta(f_\phi(x))$$

$$\log p(x \mid \hat{x}) = -\frac{D}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_d (x_d - \hat{x}_d)^2 = -\frac{1}{2\sigma^2}\|x - \hat{x}\|_2^2 + \text{const}$$

Conclusion

Minimising the Frobenius-squared (MSE) loss *is* maximum-likelihood under a Gaussian $p(x \mid \hat{x})$.
"Using MSE" silently assumes "$\hat{x}$ is a Gaussian mean."

Binary data (e.g. binarized MNIST): assume a Bernoulli, $\hat{\pi} = \mathrm{sigm}(\cdot)$

$$p(x \mid \hat{x}) = \prod_i \hat{\pi}_i^{x_i} (1 - \hat{\pi}_i)^{1 - x_i}, \qquad \hat{\pi} = \mathrm{sigm}(g_\theta(f_\phi(x)))$$

$$-\log p(x \mid \hat{x}) = -\sum_i \left[x_i \log \hat{\pi}_i + (1 - x_i) \log(1 - \hat{\pi}_i) \right]$$

$\Rightarrow$ the loss is *not* arbitrary: it is derived from the assumed **output distribution**.
Members of the exponential family (Gaussian, Bernoulli) give clean gradients after taking log.

Taking log of an exponential-family density is well-behaved

For Gaussian / Bernoulli, $\log p(x \mid \hat{x})$ is *linear* in the natural parameter, so its gradient is clean and finite — which is what backpropagation needs.

- A purely deterministic decoder corresponds to a **Dirac delta** $p(x \mid \hat{x}) = \delta(x - \hat{x})$ — a degenerate distribution outside the exponential family, with no finite statistics (mean/variance) and an ill-defined log/gradient.
- This is the precise reason a hard, point-valued output cannot be trained by likelihood; we instead model $\hat{x}$ as the *parameter* of a proper distribution.
- (Being "outside the exponential family" is the larger-picture restatement; many trainable distributions also lie outside it.)

PART 1 • pre-VAE

A history of mathematical modelling

The same six components, filled with different mathematics, give different latent spaces.

1. **Linear AutoEncoder** — global optimum = the PCA subspace (deterministic, linear)
2. **Probabilistic PCA** — the same structure as a probabilistic model (the encoder is *derived*)
3. **Nonlinear / Deep AutoEncoder** — nonlinear, multilayer, no closed-form solution
4. **Helmholtz Machine** — free energy, the variational bound, wake–sleep
5. **Limits of the AutoEncoder** — failure as a generative model → VAE

Example

In a 3-layer network with a bottleneck middle layer ($d < D$), show that on convergence the global solution is (partially) equivalent to the **PCA subspace**, and derive the mathematical solution for each component.

define model $\rightarrow$ define loss $\rightarrow$ closed form of encoder/decoder $\rightarrow$ analysis $\rightarrow$ PCA subspace

Result (Baldi–Hornik, 1989) — stated first

If Σ_x is invertible with distinct eigenvalues, every critical point is $P^* = U_{\mathcal{J}} U_{\mathcal{J}}^\top$. Only the top-$d$ choice is the **global minimum**; all others are **saddle points**. *There are no spurious local minima.*

component	symbol
input / output	$x, \hat{x} \in \mathbb{R}^D$ (auto-assoc. $y = x$)
hidden (latent)	$z \in \mathbb{R}^d$, $d < D$
encoder	$W_e \in \mathbb{R}^{d \times D}$, $z = W_e x$
decoder	$W_d \in \mathbb{R}^{D \times d}$, $\hat{x} = W_d z$
global map	$P = W_d W_e$, $\hat{x} = Px$

Assume **centered** data ($\bar{x} = 0$).

$$\Sigma_x = \frac{1}{N} \sum_i x_i x_i^\top$$

$$(\Sigma_x)_{jk} = \frac{1}{N} \langle \mathbf{c}_j, \mathbf{c}_k \rangle$$

each entry of the covariance is an **inner product** of coordinate vectors; diagonal = variance, off-diagonal 0 = orthogonal (uncorrelated). Σ_x is symmetric and PSD — the start of the spectral analysis.

$$\mathcal{L}(W_e, W_d) = \frac{1}{N}\sum_{i=1}^{N} \|x_i - W_d W_e x_i\|_2^2, \qquad P := W_d W_e \in \mathbb{R}^{D\times D}.$$

P is always rank-deficient

rank $W_e \le d$ and rank $W_d \le d$, so rank $P \le d < D$: a $D \times D$ map that cannot be the identity.

Factorisation is ambiguous

For any invertible $d \times d$ C, $W_d W_e = (W_d C)(C^{-1} W_e)$. Only the *product* P affects the loss; W_e, W_d are not individually determined.

Assume Σ_x positive definite and (for simplicity) distinct eigenvalues. Conclusion "Linear AE learns the PCA principal subspace" survives even with repeated eigenvalues.

From per-sample error to the covariance

With $X = [x_1, \ldots, x_N]$ and $\|M\|_F^2 = \mathrm{tr}(MM^\top)$,

$$\mathcal{L} = \frac{1}{N}\left\|(I - W_d W_e)X\right\|_F^2 = \mathrm{tr}\Big((I - W_d W_e)\underbrace{\tfrac{1}{N}XX^\top}_{=\Sigma_x}(I - W_d W_e)^\top\Big).$$

Trace form

$$\boxed{\mathcal{L}(W_e, W_d) = \mathrm{tr}\big((I - W_d W_e)\,\Sigma_x\,(I - W_d W_e)^\top\big)}$$

The loss depends on data only through $\Sigma_x \Rightarrow$ "minimise error" becomes "preserve high-variance directions."

Fix W_d and differentiate $\mathcal{L} = \operatorname{tr}(\Sigma_x) - 2\operatorname{tr}(W_d W_e \Sigma_x) + \operatorname{tr}(W_d W_e \Sigma_x W_e^\top W_d^\top)$:

$$\frac{\partial \mathcal{L}}{\partial W_e} = -2W_d^\top \Sigma_x + 2W_d^\top W_d W_e \Sigma_x \stackrel{!}{=} 0 \;\Rightarrow\; W_d^\top W_d W_e \Sigma_x = W_d^\top \Sigma_x.$$

Since Σ_x is invertible, right-multiply by Σ_x^{-1}; if W_d has full column rank, $W_d^\top W_d$ is invertible:

$$\boxed{W_e^* = (W_d^\top W_d)^{-1} W_d^\top} \quad \text{(Moore–Penrose left inverse of } W_d\text{)}.$$

Intuition: once the decoder subspace is fixed, the encoder maps the input to its best coordinates in that subspace.

($W_d^\top W_d$ is invertible because $v^\top W_d^\top W_d v = \|W_d v\|_2^2 > 0$ for $v \neq 0$ — positive definite, not merely symmetric.)

Substituting W_e^*:

$$P^* = W_d W_e^* = W_d (W_d^\top W_d)^{-1} W_d^\top .$$

Symmetric: $(P^*)^\top = P^*$ (inverse of symmetric is symmetric).

Idempotent: with $M = W_d^\top W_d$, $(P^*)^2 = W_d M^{-1} M M^{-1} W_d^\top = P^*$.

Consequence

A symmetric idempotent matrix is an **orthogonal projection** onto its image. So P^* projects onto the column space of W_d. The remaining question is no longer "choose all entries" but "which d-dimensional subspace?"

With P an orthogonal projection ($P^\top = P,\ P^2 = P$), the cyclic trace property gives

$$\mathcal{L}(P) = \operatorname{tr}\big((I - P)\Sigma_x\big) = \operatorname{tr}(\Sigma_x) - \operatorname{tr}(P\Sigma_x).$$

$\operatorname{tr}(\Sigma_x)$ is constant, so minimising error = **maximising preserved variance** $\operatorname{tr}(P\Sigma_x)$. With $\Sigma_x = U\Lambda U^\top$, $\lambda_1 \geq \cdots \geq \lambda_D$, and $\rho_j := u_j^\top P u_j \in [0,1]$, $\sum_j \rho_j = \operatorname{tr}(P) = d$:

$$\operatorname{tr}(P\Sigma_x) = \sum_{j=1}^{D} \lambda_j \rho_j \ \Rightarrow\ \text{maximised by } \rho_{1..d} = 1.$$

Global optimum

$$P^* = U_{1:d} U_{1:d}^\top, \qquad \mathcal{L}_{\min} = \operatorname{tr}(\Sigma_x) - \sum_{j=1}^{d} \lambda_j = \sum_{j=d+1}^{D} \lambda_j.$$

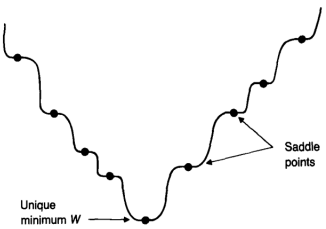


Fig 5. Loss landscape: one global min (PCA), all other critical points are saddles.

Stationarity $\Leftrightarrow P\Sigma_x = \Sigma_x P$. With distinct eigenvalues, $P = U_{\mathcal{J}} U_{\mathcal{J}}^\top$ for some index set $\mathcal{J}$, $|\mathcal{J}| = d$, with $\mathcal{L} = \sum_{j \notin \mathcal{J}} \lambda_j$.

Key fact

Only $\mathcal{J} = \{1, \dots, d\}$ is the global minimum; every other choice can be improved by rotating toward a larger eigenvalue $\Rightarrow$ a **saddle**. *No spurious local minima.*

Conclusion: Linear AE = PCA principal subspace. But the latent *basis* is not pinned to the PCA basis (rotation freedom = latent ambiguity).

What the model learns

- compresses D dimensions to d (MNIST: $784 \to 32$ still reconstructs meaningfully);
- same-class data tend to cluster in latent space;
- the learned z is useful for downstream tasks (classification, clustering).

Latent ambiguity, made precise

The global optimum fixes the *subspace* $\mathrm{col}(U_{1:d})$, but for any orthogonal R, $(W_d R,\ R^\top W_e)$ gives the same projection P^*. So:

subspace: unique coordinates/basis: not unique.

The seed of §3.5

The loss fixes *where* the z_i sit, not *what distribution* they follow — so a random z need not decode to anything meaningful.

Example

Rewrite PCA as a **probabilistic latent-variable model**: put a Gaussian prior on z and an explicit linear-Gaussian decoder. What are the marginal $p(x)$ and the encoder (inference direction)?

Roadmap — assume only the generative direction

$$p(z) = \mathcal{N}(0, I_q), \qquad p_\theta(x \mid z) = \mathcal{N}(Wz + \mu,\ \sigma^2 I_D).$$

$p(x)$ by integrating out z (3.2.4); the encoder $p(z \mid x)$ by **Bayes** (3.2.5); the loss from $-\log p(x)$ (3.2.6); the closed-form ML (3.2.7).

Key: the generative direction is *assumed*; the inference direction (encoder) is derived.

3.2 Probabilistic PCA — what is assumed vs. what is derived

Assumed (generative, $\mathcal{Z} \to \mathcal{X}$)

$$p(z) = \mathcal{N}(0, I) \quad \text{(prior)}$$

$$p(x \mid z) = \mathcal{N}(Wz + \mu, \sigma^2 I) \quad \text{(decoder)}$$

Derived (everything else)

$$p(x) = \mathcal{N}(\mu,\, WW^\top + \sigma^2 I)$$

$$p(z \mid x) \text{ (encoder, by Bayes)}$$

$$\text{loss} = -\log p(x); \quad \text{ML} = \text{PCA}$$

Only the *right-going* arrow is a modelling choice. The *left-going* encoder $p(z \mid x)$ is a **consequence** of that choice, obtained by Bayes' rule — not a separate network. This is the single structural difference from the VAE, where the encoder *is* a learned network.

Model assumptions (only these two)

$$z \sim \mathcal{N}(0, I_q), \qquad x \mid z \sim \mathcal{N}(Wz + \mu,\ \sigma^2 I_D) \iff x = Wz + \mu + \epsilon,\ \ \epsilon \sim \mathcal{N}(0, \sigma^2 I),\ z \perp \epsilon.$$

symbol	shape	role
W / $W^\top$	$D \times q$ / $q \times D$	decoder / encoder linear map
μ	$\mathbb{R}^D$	mean (PPCA does not force centering)
σ^2	scalar	isotropic noise ($\Psi = \sigma^2 I$)
$M = W^\top W + \sigma^2 I_q$	$q \times q$	appears in the posterior — **cheap to invert**
$C = WW^\top + \sigma^2 I_D$	$D \times D$	governs marginal & loss — **large**

Generative direction = assumed; inference direction = derived. (M is $q \times q$, C is $D \times D$ — this matters for EM, 3.2.8.)

Decoder, and the meaning of f_{dec}

$p(x \mid z) = \mathcal{N}(x \mid Wz + \mu, \sigma^2 I)$, so $f_{\text{dec}}(z) = Wz + \mu = \mathbb{E}[x \mid z]$ is *not* a new assumption — just the **mean** of the assumed Gaussian (used for point reconstruction).

Marginal $p(x)$ — integrate out z

Linearity of expectation and $z \perp \epsilon$:

$$\mathbb{E}[x] = W \cdot 0 + \mu + 0 = \mu, \qquad \text{Cov}(x) = W \,\text{Cov}(z)\, W^\top + \sigma^2 I = WW^\top + \sigma^2 I.$$

A linear combination of Gaussians is Gaussian:

$$\boxed{x \sim \mathcal{N}(\mu,\, C), \qquad C = WW^\top + \sigma^2 I_D.}$$

3.2 Probabilistic PCA — the encoder is *derived* by Bayes (completing the square)

The encoder $p(z \mid x)$ was never assumed; with $y \equiv x - \mu$,

$$p(z \mid x) \propto p(x \mid z)p(z) \propto \exp\Big(-\tfrac{1}{2\sigma^2}\|y - Wz\|^2 - \tfrac{1}{2}z^\top z \Big).$$

Expand $\|y - Wz\|^2 = y^\top y - 2z^\top W^\top y + z^\top W^\top W z$ and keep z-terms:

$$-\tfrac{1}{2}z^\top\Big(\tfrac{1}{\sigma^2}W^\top W + I\Big)z + \tfrac{1}{\sigma^2}z^\top W^\top y, \qquad \tfrac{1}{\sigma^2}W^\top W + I = \tfrac{1}{\sigma^2}\underbrace{(W^\top W + \sigma^2 I)}_{=M}.$$

Matching to $\mathcal{N}(z \mid m, \Sigma) \propto \exp(-\tfrac{1}{2}z^\top\Sigma^{-1}z + z^\top\Sigma^{-1}m)$:

$$\Sigma^{-1} = \tfrac{1}{\sigma^2}M \Rightarrow \Sigma = \sigma^2 M^{-1}, \qquad \Sigma^{-1}m = \tfrac{1}{\sigma^2}W^\top y \Rightarrow m = M^{-1}W^\top y.$$

Encoder (posterior) — a result, nothing assumed

$$p(z \mid x) = \mathcal{N}\big(z \mid M^{-1}W^\top(x - \mu),\ \sigma^2 M^{-1}\big), \qquad M = W^\top W + \sigma^2 I_q.$$

3.2 Probabilistic PCA — the loss is the marginal log-likelihood

Conditional log-density (no sum yet): $\log p(x \mid z) = -\frac{D}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|x-\mu-Wz\|^2$. Two changes turn this into the training loss:

(a) integrate out z (b) sum over N points (c) trace trick

$$\log p(x) = -\tfrac{1}{2}\left[D\log 2\pi + \log|C| + (x-\mu)^\top C^{-1}(x-\mu)\right]$$

the reconstruction term becomes a **Mahalanobis** term (no z). Summing and using $a^\top M a = \operatorname{tr}(Maa^\top)$:

$$\sum_n (x_n-\mu)^\top C^{-1}(x_n-\mu) = \operatorname{tr}\Big(C^{-1}\underbrace{\textstyle\sum_n (x_n-\mu)(x_n-\mu)^\top}_{=NS}\Big) = N\operatorname{tr}(C^{-1}S).$$

$$\boxed{\log p(X) = -\frac{N}{2}\left[D\log 2\pi + \log|C| + \operatorname{tr}(C^{-1}S)\right].}$$

Unlike PCA's $\sum_n \|x_n - \hat{x}_n\|^2$, the PPCA loss is Mahalanobis + a $\log|C|$ penalty against an over-spread model.

Closed-form ML = PCA

With $S = U\Lambda U^\top$ and top-q block U_q, Λ_q:

$$\mu_{\mathrm{ML}} = \bar{x}, \quad \sigma^2_{\mathrm{ML}} = \frac{1}{D-q}\sum_{j>q}\lambda_j,$$

$$W_{\mathrm{ML}} = U_q(\Lambda_q - \sigma^2 I)^{1/2}R.$$

R is an arbitrary rotation $\Rightarrow$ the *subspace* (not the coordinates) is essential — same ambiguity as $P = W_d W_e$. U_q is exactly PCA's top eigenvectors.

Why EM? — cost

Closed form needs $O(ND^2)$ to form S and $O(D^3)$ to eigendecompose. EM never forms a $D \times D$ matrix: it inverts M ($q \times q$, $O(q^3)$), each step $O(NDq)$.

- **E-step = encode**: posterior moments $\mathbb{E}[z_n], \mathbb{E}[z_n z_n^\top]$.
- **M-step = retrain decoder**: update W, σ^2.
- PPCA posterior is exact, so its EM is exact (not variational).

Bridge: when the posterior is *not* available in closed form — the encoder must be **learned** by a network — we move to Helmholtz Machine & VAE.

E-step = encoder (posterior moments, with $y_n = x_n - \mu$)

$$\mathbb{E}[z_n] = M^{-1} W^\top (x_n - \mu), \qquad \mathbb{E}[z_n z_n^\top] = \sigma^2 M^{-1} + \mathbb{E}[z_n]\mathbb{E}[z_n]^\top.$$

M-step = retrain decoder

$$W_{\text{new}} = \Big[\textstyle\sum_n y_n\, \mathbb{E}[z_n]^\top\Big] \Big[\textstyle\sum_n \mathbb{E}[z_n z_n^\top]\Big]^{-1}$$

$$\sigma^2_{\text{new}} = \frac{1}{ND} \sum_n \Big\{ \|y_n\|^2 - 2\,\mathbb{E}[z_n]^\top W_{\text{new}}^\top y_n + \operatorname{tr}\big(\mathbb{E}[z_n z_n^\top]\, W_{\text{new}}^\top W_{\text{new}}\big) \Big\}$$

Only the $q \times q$ matrix $M = W^\top W + \sigma^2 I$ is inverted ($O(q^3)$); a step costs $O(NDq)$, replacing the $O(D^3)$ eigendecomposition. E-step = encode, M-step = decode — the same pair, ready to be amortised by a network.

Example

A multilayer AutoEncoder with nonlinear activations everywhere except the code layer has *no* closed-form solution and *no* PPCA-style probabilistic reading. How do we **train** the encoder G_ϕ and decoder H_θ?

Answer — stated first

Do not optimise everything at once. **Stage** it: (1) unsupervised **pretraining** per layer for a good initialisation; (2) **fine-tune** the whole network by backpropagation.

The six components are almost identical to Linear AE — only the activation cell turns nonlinear, which makes the objective *non-convex*.

Example — three questions built from the Linear-AE assumptions

1. Dataset A clearly has low intrinsic dimension, yet PCA / Linear AE reduce it poorly *and* its coordinates are uncorrelated. Why?
2. Given that diagnosis, what method actually works?
3. A nonlinear encoder still builds each code coordinate independently. If the latent coordinates must *depend* on one another, does method 2 still apply?

Answers

(1) PCA sees only covariance; uncorrelated $\neq$ independent, and curved-manifold structure hides from covariance. **(2)** make encoder/decoder nonlinear — (a) explicit geometry (Isomap) or (b) stacked nonlinear layers (this route). **(3)** no — intra-layer dependence needs energy-based models (Boltzmann); restrict them $\rightarrow$ RBM.

Background

After 1988 (Linear AE = PCA), people built PCA-style generative models for MNIST / eigenfaces — performance was disappointing. Stacking nonlinear layers did markedly better.

The change

Linear AE collapsed to a single matrix (PCA/SVD, closed form); PPCA moved it to a probabilistic model (EM). Nonlinear composition makes the objective **non-convex** — no closed form, no PPCA-style reading. The problem becomes "how to train," not "how to solve."

component	Linear AE	Nonlinear AE
recognition G	$z = W_e x$	$z = G_\phi(x)$, composed nonlinear layers
generation H	$\hat{x} = W_d z$	$\hat{x} = H_\theta(z)$, composed nonlinear layers
solution	closed form (PCA)	none — staged training

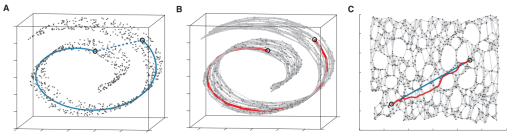


Fig. 3. The "Swiss roll" data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (**A**) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (**B**) The neighborhood graph G constructed in step one of Isomap (with $K = 7$ and $N =$ 1000 data points) allows an approximation (red segments) to the true geodesic path to be computed efficiently in step two, as the shortest path in G. (**C**) The two-dimensional embedding recovered by Isomap in step three, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding (blue) now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).

Fig 6. Swiss roll: uncorrelated, yet intrinsic dimension 2.

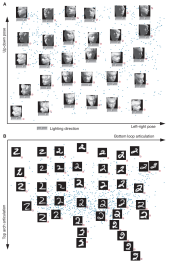


Fig 7. Isomap: geodesic distance "unfolds" the manifold.

- PCA / Linear AE see only **2nd-order (covariance)** structure. But *uncorrelated* $\neq$ *independent*.
- e.g. $t \sim \mathrm{Unif}[0, 2\pi)$, $x = (\cos t, \sin t)$: covariance $\frac{1}{2}I$ (no preferred direction), yet intrinsic dimension is 1.
- On a curved manifold the low-dimensional structure hides from the covariance $\Rightarrow$ need a nonlinear encoder/decoder. This section follows the "stack nonlinear layers" route, not explicit geometry (Isomap).

Layer-wise pretraining

Each layer is pretrained unsupervised as an **RBM** (restricted Boltzmann machine) — "restricted" removes intra-layer connections so units within a layer are conditionally independent, giving a tractable building block.

Then fine-tune

Unfold the stack into a deep autoencoder and refine end-to-end by backpropagation.

Why a third question matters

A feed-forward encoder makes each code coordinate *independently*. If coordinates must **depend** on one another (units within a layer interacting), we enter energy-based models. A fully-connected Boltzmann machine has $O(n^2)$ pairwise interactions $\Rightarrow$ restrict to RBM.

Performance

Deep nonlinear codes beat PCA (Hinton–Salakhutdinov, 2006).

- Nonlinear composition $\Rightarrow$ no closed form and no PPCA-style reading; the model cannot be solved in one shot.
- Instead: **layer-wise unsupervised pretraining** for a good initialisation, then **backpropagation fine-tuning**, yielding deep nonlinear codes that *beat* PCA.
- But greater expressive power does *not* make it generative: the definition is still **deterministic** reconstruction.

Two threads carried forward

"if you can't solve it at once, build a good init and refine" and "model the latent representation as a *distribution*" — the foundations of the next chapter (Helmholtz) and the VAE.

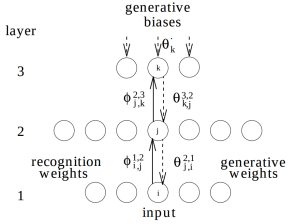


Figure 3: A Helmholtz Machine.
A simple three layer Helmholtz machine modelling the activity of 5 binary inputs (layer 1) using a two stage hierarchical model. Generative weights (θ) are shown as dashed lines, including the generative biases, the only such input to the units in the top layer. Recognition weights (ϕ) are shown with solid lines. Recognition and generative activation functions are described in the text.

Fig 9. bottom-up recognition & top-down generative networks.

Perception = inferring the hidden **causes** of sensory input ("unconscious inference").

Two networks

$$p_\theta(x, z) = p_\theta(z)\, p_\theta(x \mid z) \quad \text{(generative, top-down)}$$

$$q_\phi(z \mid x) \quad \text{(recognition, bottom-up)}$$

- latent is layered $z = (z^1, \ldots, z^L)$, each unit binary stochastic, factorial sigmoid conditionals (the Bernoulli model, repeated per layer).
- recognition = encoder, generative = decoder. The pair structure the VAE inherits.

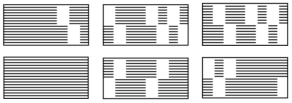


Figure 1: Shift Patterns. In each of these six patterns the bottom row of square pixels is a random binary vector, the top row is a copy shifted left or right by one pixel with wraparound, and the middle two rows are copies of the outer rows. The patterns were generated by a two-stage process. First the direction of the shift was chosen, with left and right being equiprobable. Then each pixel in the bottom row was turned on (white) with a probability of 0.2, and the corresponding shifted pixel in the top row and the copies of these in the middle rows were made to follow suit. If we treat the top two rows as a left retina and the bottom two rows as a right retina, detecting the direction of the shift resembles the task of extracting depth from simple stereo images of short vertical line segments. Copying the top and bottom rows introduces extra redundancy into the images which facilitates the search for the correct generative model.

Fig 8. Data and the patterns the generative model produces from latent causes.

Latent is layered; generative goes top-down, recognition bottom-up, one conditional per layer:

$$p_\theta(x,z) = p_\theta(z^L)\prod_{l=1}^{L-1} p_\theta(z^l \mid z^{l+1})\, p_\theta(x \mid z^1)$$

$$q_\phi(z \mid x) = q_\phi(z^1 \mid x)\prod_{l=2}^{L} q_\phi(z^l \mid z^{l-1})$$

Each unit is binary stochastic; within a layer, units are conditionally independent given the neighbouring layer — a **factorial sigmoid** conditional:

$$p_\theta(z_j^l = 1 \mid z^{l+1}) = \operatorname{sigm}\Big(b_j^l + \textstyle\sum_k W_{jk}^l z_k^{l+1}\Big).$$

This is the Bernoulli model of §2, repeated per layer.

Maximum likelihood is a sum over all explanations

$$\log p_\theta(x) = \log \sum_z p_\theta(x, z).$$

Each z is one *explanation* of x; with L layers of h binary units there are 2^{hL} of them.

- the sum, and the posterior $p_\theta(z \mid x)$, are intractable.
- EM (which needs the exact posterior) cannot be used.

$\Rightarrow$ we need a tractable surrogate for $\log p_\theta(x)$: the **variational free energy**.

Put a tractable Q where the true posterior P sits. This is an *equality*:

$$\log p(d \mid \theta) = \underbrace{-\sum_{\alpha} Q_\alpha E_\alpha - \sum_{\alpha} Q_\alpha \log Q_\alpha}_{=-\mathcal{F}(d;\theta,Q)} + \underbrace{\sum_{\alpha} Q_\alpha \log \frac{Q_\alpha}{P_\alpha}}_{=D_{\mathrm{KL}}(Q\|P)\geq 0}.$$

Since $D_{\mathrm{KL}} \geq 0$, dropping it gives a lower bound; with $E_\alpha = -\log p(\alpha, d \mid \theta)$,

$$\log p(d \mid \theta) \;\geq\; -\mathcal{F}, \qquad \mathcal{F} = \underbrace{\langle E \rangle_Q}_{\text{expected energy}} - \underbrace{H[Q]}_{\text{recognition entropy}}.$$

$-\mathcal{F}$ *is* the **ELBO**. Maximise over θ (fit generative) and over ϕ ($Q = Q_\phi$; tightens the bound).

The Helmholtz-Machine paper predates the VAE — this is why the ELBO is still called "free-energy" optimisation.

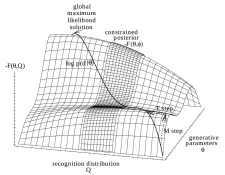


Figure 2: Graphical view of our approximation. The surface shows a simplified example of $-F(\theta, Q)$ as a function of the generative parameters θ and the recognition distribution Q. As discussed by Neal and Hinton (1994), the Expectation-Maximization algorithm ascends this surface by optimising alternately with respect to θ (the M-step) and Q (the E-step). After each E-step, the point on the surface lies on the line defined by $Q_\alpha = P_\alpha$, and on this line, $-F = \log p(d|\theta)$. Using a factorial recognition distribution parameterised by ϕ restricts the surface over which the system optimises (labeled "constrained posterior"). We ascend the restricted surface using a conjugate gradient optimisation method. For a given θ, the difference between $\log p(d|\theta) = \max_Q \{-F(\theta, Q)\}$ and $-F(\theta, Q)$ is the Kullback-Leibler penalty in equation 5.

That EM gets stuck in a local maximum here is largely for graphical convenience, although neither it, nor our conjugate gradient procedure, is guaranteed to find their respective global optima. Showing the factorial recognition as a connected region is an arbitrary convention; the actual structure of the recognition distributions cannot be preserved in one dimension.

Fig 10. Free-energy landscape: lowering energy clusters the latent explanations so they become distinguishable.

Hinton's move: read $\log p(d \mid \theta)$ as a Boltzmann distribution and identify its form with a temperature-1 free energy.

$$E_\alpha = -\log\left[p(\alpha \mid \theta)\, p(d \mid \alpha, \theta)\right]$$

- the true posterior P_α is Boltzmann; its free energy equals $-\log p(d \mid \theta)$ exactly.
- lowering energy makes latent vectors form a **clustered** distribution around a mean $\Rightarrow$ the model learns to tell explanations apart.
- this is the first time an AutoEncoder's training was cast in the language of statistical physics (energy / entropy).

Why "free energy"

Thermodynamic Helmholtz free energy is $A = U - TS$. Here $\mathcal{F} = \langle E \rangle_Q - H[Q]$ is exactly that form (temperature $T = 1$): it trades model fit (low energy) against the spread of Q (high entropy). This formal match names the model.

Factorial Q and explaining-away

Q_ϕ is assumed factorial (units independent) for tractability. But the true posterior is *not*: observing x couples its causes (**explaining-away**).

$$\min_\phi D_{\mathrm{KL}}\big(q_\phi \| p_\theta(z \mid x)\big) > 0 \;\Rightarrow\; \mathcal{F} > -\log p_\theta(x)$$

a *structural* gap (the VAE's diagonal Gaussian inherits it).

3.4 Helmholtz Machine — wake–sleep, and the bridge to VAE

Wake–sleep learning

wake	clamp data x, draw $z \sim q_\phi$; update θ via $\nabla_\theta \log p_\theta(x, z)$
sleep	"dream" $(x, z) \sim p_\theta$; update ϕ via $\nabla_\phi \log q_\phi(z \mid x)$

Local delta rules — no backprop of error needed (biologically plausible).

The limitation → VAE

There is **no single objective** both phases descend: sleep minimises the *reversed* KL $D_{\mathrm{KL}}(p_\theta \| q_\phi)$ on dreamed data, not the $D_{\mathrm{KL}}(q_\phi \| p_\theta)$ inside $\mathcal{F} \Rightarrow$ may oscillate.

The VAE fixes exactly this: one **ELBO** optimised jointly in θ, ϕ via the **reparameterization trick**. (EM is coordinate descent on the same $\mathcal{F}$; Helmholtz amortises its E-step with a network.)

Definition • generative model

A model $p_\theta(x)$ approximating $p_{\text{data}}(x)$ from which we can draw *new* samples $\tilde{x} \sim p_\theta(x)$.

- Take a well-trained AutoEncoder on MNIST and feed the decoder a random $z \sim \mathcal{N}(0, I)$: the output is almost always meaningless noise.
- Cause — the latent space has two regions: near the encoded data $z_i = f_\phi(x_i)$ (decoder works) and everywhere else (no guarantee). A random z lands in the second region almost surely.
- As the Linear-AE analysis showed, the loss only fixes the *positions* of the z_i; it never forces their distribution to be $\mathcal{N}(0, I)$.

A latent space usable for generation needs:

- **Continuity**: nearby z decode to nearby x.
- **Completeness**: any point in the latent region (and interpolations) decodes to something meaningful.
- **Samplability**: a known distribution to sample new z from.

The standard AutoEncoder satisfies *none* of these.

The fix (start of the VAE)

Make the encoder map an input to a **distribution**, not a point:

$$z \sim q_\phi(z \mid x), \qquad p(z) = \mathcal{N}(0, I) \text{ on the space.}$$

Each step samples a different z, so the decoder gets a training signal over a whole *region*, not a single point — the space fills in.

$\Rightarrow$ Variational AutoEncoder (VAE)

AutoEncoder — point

$x \longmapsto z = f_\phi(x)$ (one point)

points are isolated; the gaps (?) are undefined.

VAE — distribution

$x \longmapsto q_\phi(z \mid x) = \mathcal{N}(\mu(x), \sigma^2(x))$

overlapping Gaussians tile the space; *between* data points the decoder still receives signal.

$p(z) = \mathcal{N}(0, I)$ makes the space **samplable**: draw $z \sim p(z)$, decode, get a plausible $\tilde{x}$.

Continuity, completeness, samplability — all three follow from replacing the *point* with a *distribution*.

Recap — the whole arc on one slide

deterministic → probabilistic, point → distribution

model	latent	how trained	key result / limit
Linear AE	point	closed form (PCA)	global opt = PCA subspace; no spurious minima
PPCA	Gaussian posterior	closed form / EM	encoder *derived* by Bayes; ML = PCA
Nonlinear AE	point	pretrain + fine-tune	beats PCA; still deterministic (not generative)
Helmholtz	binary stochastic	wake–sleep / free energy	ELBO appears; but no single joint objective
(VAE)	**Gaussian sample**	**single ELBO**	**generative; Part 2**

Every row keeps the recognition–generation pair; the differences live entirely in the latent and the training.

Part 1 — summary, and a preview of Part 2

Everything here precedes the VAE

- the same **six components**, filled deterministically or probabilistically, drive the whole development;
- Linear AE = PCA → PPCA (encoder *derived* by Bayes) → Deep AE (nonlinear) → Helmholtz (distributions, the bound, wake–sleep);
- the AutoEncoder is **not** a generative model — because it puts the latent at a *point*.

In Part 2

Put the latent in a distribution and unify the scattered training into a single **ELBO** + the **reparameterization trick**:

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

→ VAE, β-VAE, VQ-VAE.

PART 2 • VAE

VAE: concept and structure

Original slide: Contents (course outline)

Deep learning splits into three parts: **(1) Introduction**, **(2) Supervised Learning** (NN, CNN), **(3) Unsupervised Learning**. The **VAE** is an *unsupervised generative model*. Order of the unsupervised track:

label=1) Introduction to Unsupervised Learning

lbbel=2) Autoencoder & Anomaly Detection

lcbel=3) **Variational AutoEncoder (VAE)** ← this topic

ldbel=4) Generative Adversarial Network (GAN)

lebel=5) Advanced GANs

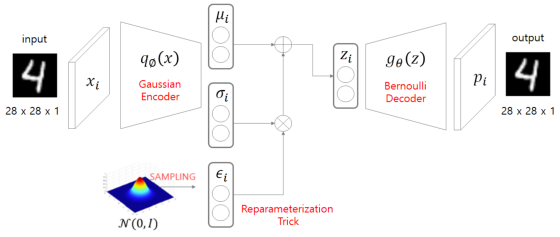
Locating the VAE

The VAE starts from the **AE** and *extends it probabilistically*; it is taught right after the AE and before the GAN. The AE is a deterministic "compress then reconstruct" structure; the VAE adds *distributions* and *sampling* to the same structure, enabling *new-data generation*.

Slide: "Variational Autoencoders (VAE) – How to work"

A pipeline **encoder** → **(reparameterization)** → **decoder**. Input $28 \times 28 \times 1$ reconstructs to the same size. Data flow:

$$x_i \xrightarrow{q_\phi(x)} \mu_i,\ \sigma_i \qquad \mu_i,\ \sigma_i,\ \epsilon_i \longrightarrow z_i \qquad z_i \xrightarrow{g_\theta(z)} p_i$$



Source: Taeu Kim — taeu.github.io/paper/deeplearning-paper-vae

- **Gaussian Encoder** $q_\phi(x)$: outputs the latent *mean* μ_i and *std* σ_i — a *distribution*, not a point.
- **Reparameterization**: combine $\epsilon_i \sim \mathcal{N}(0, I)$ with μ_i, σ_i to form z_i (Ch. 14).
- **Bernoulli Decoder** $g_\theta(z)$: generates p_i from z_i, reconstructing the input.

AE vs VAE — opposite purposes

The **AE was built for its Encoder** (dimensionality reduction / feature extraction); the **VAE for its Decoder** (data generation). Same structure, different goal.

Slide: "VAE – How to work" (p.5)

The loss has two parts: **reconstruction error** and **regularization**.

(1) Reconstruction error (Bernoulli → cross-entropy):

$$-\sum_{j=1}^{D}\left[\, x_{i,j}\log p_{i,j} + (1-x_{i,j})\log(1-p_{i,j})\,\right]$$

as an expectation:

$$-\,\mathbb{E}_{q_\phi(z|x_i)}\left[\,\log\left(p(x_i \mid g_\theta(z))\right)\right]$$

(2) Regularization (pull the latent toward the prior):

$$\tfrac{1}{2}\sum_{j=1}^{J}\left(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1\right)$$

Reconstruction = how well the decoder revives the input; regularization = how close q_ϕ is to $\mathcal{N}(0, I)$. Both are minimized together. (Derivations: Ch. 7–16.)

Slide: "VAE – Loss Function" (p.6)

A *probabilistic spin* on the AE lets us **sample to generate**. Data $\{x^{(i)}\}_{i=1}^{N}$ are generated from an unobserved latent z:

- sample from the true prior: $z \sim p_{\theta^*}(z)$
- sample from the true conditional: $x \sim p_{\theta^*}\left(x \mid z^{(i)}\right)$

Intuition

If x is an image, z holds its latent factors (attributes, orientation). Learn their distribution, draw a new z, and generate a new image.

PART 2 • VAE

VAE: the variational lower bound

Slide: "VAE – Loss Function" (p.8)

Estimate the true θ^* by maximizing the data likelihood; with the latent z it is a marginal:

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

Intractable

The integral sums $p_\theta(x|z)$ over *all* continuous, high-dimensional z — not computable exactly. Direct likelihood maximization is impossible.

Slide: "VAE – Loss Function" (p.9)

$$\underbrace{p_\theta(x)}_{\text{intractable likelihood}} = \int \underbrace{p_\theta(z)}_{\text{Gaussian prior}} \underbrace{p_\theta(x \mid z)}_{\text{decoder net}} \, dz$$

The posterior is also intractable:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)\, p_\theta(z)}{p_\theta(x)}$$

(the denominator is the intractable $p_\theta(x)$).

Fix — add an encoder

$$q_\phi(z \mid x) \approx p_\theta(z \mid x)$$

This yields a *tractable lower bound* to optimize.

Amortized inference

Classical VI optimizes (μ_i, σ_i) *per datum*. The VAE learns one shared encoder q_ϕ that outputs μ_i, σ_i for any x in a single forward pass (the *recognition network*). Cost is amortized over the dataset; the small per-datum suboptimality is the *amortization gap*.

Slide: "VAE – Loss Function" (p.10)

Steps: (a) wrap in $\mathbb{E}_{z\sim q_\phi}$ since $p_\theta(x^{(i)})$ is independent of z; (b) Bayes' rule; (c) multiply by $\frac{q_\phi}{q_\phi} = 1$; (d) split the log; (e) recognize two KL divergences.

$$
\begin{aligned}
\log p_\theta\big(x^{(i)}\big) &= \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\big[\log p_\theta(x^{(i)})\big] \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)}|z)\,p_\theta(z)}{p_\theta(z|x^{(i)})}\right] \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)}|z)\,p_\theta(z)}{p_\theta(z|x^{(i)})}\cdot\frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right] \\
&= \mathbb{E}_z\big[\log p_\theta(x^{(i)}|z)\big] - \mathbb{E}_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right] \\
&= \mathbb{E}_z\big[\log p_\theta(x^{(i)}|z)\big] - D_{\mathrm{KL}}\big(q_\phi(z|x^{(i)})\,\|\,p_\theta(z)\big) + D_{\mathrm{KL}}\big(q_\phi(z|x^{(i)})\,\|\,p_\theta(z|x^{(i)})\big)
\end{aligned}
$$

The last two terms become KL divergences because the expectation is taken w.r.t. q_ϕ.

Note — definition of KL

$$\mathbb{E}_{z\sim q_\phi}\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] = \int_z \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\, q_\phi(z|x^{(i)})\, dz, \qquad D_{\mathrm{KL}}(P \,\|\, Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Slide: “VAE – Loss Function” (p.11)

$$\log p_\theta(x^{(i)}) = \underbrace{\mathbb{E}_z\big[\log p_\theta(x^{(i)}|z)\big]}_{(1)} \underbrace{- D_{\mathrm{KL}}\big(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z)\big)}_{(2)} \underbrace{+ D_{\mathrm{KL}}\big(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z|x^{(i)})\big)}_{(3)}$$

label=**(1)** **decoder term** $\mathbb{E}_z[\log p_\theta(x^{(i)}|z)]$: estimate by sampling; made differentiable by reparameterization.

lbbel=**(2)** **encoder–prior KL** $-D_{\mathrm{KL}}(q_\phi \| p_\theta(z))$: has a closed form (Ch. 12).

lcbel=**(3)** **encoder–posterior KL** $+D_{\mathrm{KL}}(q_\phi \| p_\theta(z|x^{(i)}))$: contains the intractable $p_\theta(z|x)$, *not computable*; but always ≥ 0.

Strategy

(1) and (2) are computable; (3) is not, but is ≥ 0. Use this to build the *lower bound (ELBO)* next.

Slide: "VAE – Loss Function" (p.13)

Group the two computable terms:

$$\mathcal{L}\big(x^{(i)}, \theta, \phi\big) = \mathbb{E}_z\big[\log p_\theta(x^{(i)}|z)\big] - D_{\mathrm{KL}}\big(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z)\big)$$

Since the third term ≥ 0:

$$\boxed{\log p_\theta\big(x^{(i)}\big) \;\geq\; \mathcal{L}\big(x^{(i)}, \theta, \phi\big)}$$

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}\big(x^{(i)}, \theta, \phi\big)$$

Why raising the bound works

$\log p_\theta(x)$ is hard; $\mathcal{L} \leq \log p_\theta(x)$ is computable; the gap (3rd KL) is ≥ 0, so raising $\mathcal{L}$ pushes the log-likelihood up. **Training = maximize the lower bound.**

Slide: "VAE – Loss Function" (p.14)

Maximizing the ELBO = minimizing its negative:

$$\arg\min_{\theta,\phi} \sum_i \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p(x_i|g_\theta(z))\right)\right]}_{\text{reconstruction error}} + \underbrace{D_{\mathrm{KL}}\left(q_\phi(z|x_i)\,\|\,p(z)\right)}_{\text{regularization}}$$

Note

$p(x\,|\,g_\theta(z)) = p_\theta(x\,|\,z)$: the likelihood of the decoder output is $p_\theta(x|z)$.

- **reconstruction error**: the negative log-likelihood under the encoder distribution (the AE reconstruction).
- **regularization**: an extra condition — the latent must be close to the prior, for easy sampling and controllable generation.

PART 2 • VAE

VAE: optimization

Slide: "VAE – Optimization" (p.15)

To actually compute

$$\arg\min_{\theta,\phi} \sum_i -\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p(x_i|g_\theta(z))\right)\right] + \underbrace{D_{\mathrm{KL}}\left(q_\phi(z|x_i)\,\|\,p(z)\right)}_{\text{regularization}}$$

we need distributional assumptions.

Assumption 1 — encoder (approximation class)

A multivariate Gaussian with diagonal covariance:

$$q_\phi(z\,|\,x_i) \sim \mathcal{N}\left(\mu_i,\ \sigma_i^2 I\right)$$

μ_i, σ_i are *outputs* of the shared encoder (amortized), not per-datum free parameters; new-input inference is one forward pass.

Assumption 2 — prior

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

Diagonal covariance $\Rightarrow$ each latent dim is independent $\Rightarrow$ the KL becomes a per-dimension sum (closed form, next).

Slide: "VAE – Optimization / KLD" (p.16)

General KL between two normals:

$$D_{\mathrm{KL}}\left(\mathcal{N}_0 \,\|\, \mathcal{N}_1\right) = \tfrac{1}{2}\Big(\operatorname{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1-\mu_0)^\top \Sigma_1^{-1}(\mu_1-\mu_0) - k + \ln \tfrac{\det \Sigma_1}{\det \Sigma_0}\Big)$$

Substitute $\mathcal{N}_0 = q_\phi(z|x_i) = \mathcal{N}(\mu_i, \sigma_i^2 I)$, $\mathcal{N}_1 = p(z) = \mathcal{N}(0, I)$:

$$\begin{aligned} D_{\mathrm{KL}}\left(q_\phi(z|x_i) \,\|\, p(z)\right) &= \tfrac{1}{2}\Big\{ \operatorname{tr}\left(\sigma_i^2 I\right) + \mu_i^\top \mu_i - J + \ln \frac{1}{\prod_{j=1}^{J} \sigma_{i,j}^2} \Big\} \\ &= \tfrac{1}{2}\Big\{ \sum_{j=1}^{J} \sigma_{i,j}^2 + \sum_{j=1}^{J} \mu_{i,j}^2 - J - \sum_{j=1}^{J} \ln(\sigma_{i,j}^2) \Big\} \\ &= \tfrac{1}{2} \sum_{j=1}^{J} \left(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1\right) \end{aligned}$$

Here $\mathrm{tr}(\sigma_i^2 I) = \sum_j \sigma_{i,j}^2$, $\mu_i^\top \mu_i = \sum_j \mu_{i,j}^2$, $k = J$, $\ln \frac{1}{\prod \sigma^2} = -\sum_j \ln(\sigma_{i,j}^2)$, and $-J = \sum_j (-1)$.

Interpretation

Pushes each dimension's $\mu \to 0$ and $\sigma^2 \to 1$ — pulling q_ϕ toward $\mathcal{N}(0, I)$.

Slide: "VAE – Optimization / Sampling" (p.17)

The reconstruction term is an expectation; approximate it by Monte Carlo:

$$\mathbb{E}_{q_\phi(z|x_i)}\left[\log(p_\theta(x_i|z))\right] = \int \log(p_\theta(x_i|z))\, q_\phi(z|x_i)\, dz \approx \frac{1}{L}\sum_{z^{i,l}} \log\left(p_\theta(x_i|z^{i,l})\right)$$

Draw $z^{i,1}, \ldots, z^{i,L} \sim q_\phi$ and average their log-likelihoods.

- L is the number of latent samples.
- usually $L = 1$ for convenience.

Slide: "VAE – Optimization / Reparameterization" (p.18)

The sampling step

$$z^{i,l} \sim \mathcal{N}(\mu_i,\ \sigma_i^2 I)$$

is a *stochastic* operation $\Rightarrow$ we cannot backpropagate through it (the red-X path on the slide).

Reparameterize

$$z^{i,l} = \mu_i + \sigma_i \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0,\ I)$$

Key

$\mu_i + \sigma_i \odot \epsilon$ has the *same distribution* as $\mathcal{N}(\mu_i, \sigma_i^2 I)$, but z is now differentiable in $\mu_i, \sigma_i \Rightarrow$ *backpropagation works*. The randomness lives only in ϵ, which is not a learning target.

Slide: "VAE – Optimization / Assumption 3-1" (p.19)

With $L = 1$: $\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))] \approx \log\left(p_\theta(x_i|z^i)\right)$.

Assumption 3-1 — decoder likelihood is multivariate Bernoulli, $p_\theta(x_i|z^i) \sim \text{Bernoulli}(p_i)$:

$$\begin{aligned}
\log\left(p_\theta(x_i|z^i)\right) &= \log\prod_{j=1}^{D} p_\theta(x_{i,j}|z^i) = \sum_{j=1}^{D} \log p_\theta(x_{i,j}|z^i) \\
&= \sum_{j=1}^{D} \log\left(p_{i,j}^{x_{i,j}}(1-p_{i,j})^{1-x_{i,j}}\right) && (p_{i,j}:\ \text{net output}) \\
&= \sum_{j=1}^{D} \left[\, x_{i,j}\log p_{i,j} + (1-x_{i,j})\log(1-p_{i,j}) \,\right] && (\textbf{cross-entropy})
\end{aligned}$$

Conclusion

Bernoulli decoder $\Rightarrow$ reconstruction error $=$ (binary) cross-entropy (good for $[0,1]$ pixels, e.g. MNIST). The error is the *negative* of this log-likelihood.

Slide: "VAE – Optimization / Assumption 3-2" (p.20)

Same start: $\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))] \approx \log\left(p_\theta(x_i|z^i)\right)$.

Assumption 3-2 — decoder likelihood is Gaussian:

$$\log\left(p_\theta(x_i|z^i)\right) = \log\left(\mathcal{N}(x_i;\mu_i,\sigma_i^2 I)\right) = -\sum_{j=1}^{D}\left[\tfrac{1}{2}\log(\sigma_{i,j}^2) + \frac{(x_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}\right]$$

With identity covariance the variance term is constant, so it is proportional to

$$\log\left(p_\theta(x_i|z^i)\right) \propto -\sum_{j=1}^{D}\left(x_{i,j}-\mu_{i,j}\right)^2 \qquad (\textbf{squared error})$$

Conclusion

Gaussian decoder ⇒ reconstruction error = squared error (MSE) — for continuous data; identity covariance simplifies to plain MSE.

PART 2 • VAE

VAE: architecture, characteristics, code

Slide: "VAE – Structure" (p.21)

The default structure:

$$x_i\,(\dim D) \xrightarrow{q_\phi \text{ Gaussian Enc.}} \mu_i, \sigma_i \xrightarrow{\oplus,\otimes\ (\epsilon_i \sim \mathcal{N}(0,I))} z^i\,(\dim J) \xrightarrow{g_\theta \text{ Bernoulli Dec.}} p_i$$

μ_i is added to ϵ_i ($\oplus$) and σ_i multiplied ($\otimes$) to form z^i. Two loss terms:

Reconstruction error (cross-entropy):

$$-\sum_{j=1}^{D} \left[\, x_{i,j} \log p_{i,j} + (1 - x_{i,j}) \log(1 - p_{i,j}) \,\right]$$

Regularization:

$$\tfrac{1}{2} \sum_{j=1}^{J} \left(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1\right)$$

Input dim D, latent dim J. The "default" form = Gaussian encoder + Bernoulli decoder.

Slide: "VAE – Structure" (p.22)

The encoder and the regularization are unchanged; only the **reconstruction error** changes.

(A) Gaussian decoder (outputs mean μ' and std σ'):

$$\text{Reconstruction Error} = \sum_{j=1}^{D} \left[\tfrac{1}{2} \log\left(\sigma_{i,j}'^2\right) + \frac{(x_{i,j} - \mu_{i,j}')^2}{2\,\sigma_{i,j}'^2} \right]$$

(B) Gaussian decoder, identity covariance (outputs mean μ_i only):

$$\text{Reconstruction Error} = \sum_{j=1}^{D} \left(x_{i,j} - \mu_{i,j}\right)^2 \qquad (\textbf{squared error})$$

Both keep the same **Regularization**:

$$\tfrac{1}{2} \sum_{j=1}^{J} \left(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1\right)$$

Summary

Bernoulli → cross-entropy (binary/normalized images); Gaussian → squared error (continuous). Identity covariance collapses the Gaussian decoder to plain MSE.

Slide: "VAE – Characteristics" (p.24)

$$\arg\min_{\theta,\phi} \sum_i \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}\left[\log(p_\theta(x_i|z))\right]}_{\text{reconstruction error}} + \underbrace{D_{\mathrm{KL}}\left(q_\phi(z|x_i) \,\|\, p(z)\right)}_{\text{constraint}}$$

Reconstruction = cross-entropy between input and output; constraint = divergence from the prior (KL).

Reconstructions as the latent dim grows ($|z| = 2, 5, 20$ vs. input). Source: Taeu Kim — taeu.github.io/paper/deeplearning-paper-vae

- a probabilistic spin on the AE $\Rightarrow$ can generate data;
- defines an intractable density $\Rightarrow$ optimize a variational lower bound.

Characteristics

1. The **Decoder** can generate (at least) the training data $\Rightarrow$ samples resemble the training data.
2. The **Encoder** represents the training data well as latent vectors $\Rightarrow$ widely used for *abstraction*.

VAE vs GAN: the GAN is sharper, but the VAE trains stably and excels at representation learning.

Slide: "VAE coding" (p.25)

Per-sample loss; minimizing it = maximizing the ELBO:

$$L_i(\phi,\theta,x_i) = \underbrace{-\,\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p(x_i|g_\theta(z))\right)\right]}_{\text{Reconstruction Error}} + \underbrace{D_{\mathrm{KL}}\left(q_\phi(z|x_i)\,\|\,p(z)\right)}_{\text{Regularization}} \;\Rightarrow\; \arg\max\ \mathrm{ELBO}(\phi)$$

[Regularization: Kullback–Leibler divergence]

$$D_{\mathrm{KL}}\left(q_\phi(z|x_i)\,\|\,p(z)\right) = \tfrac{1}{2}\sum_{j=1}^{J}\left(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1\right)$$

[Reconstruction Error] — Monte Carlo ($L = 1$):

$$-\,\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p(x_i|g_\theta(z))\right)\right] \approx \frac{1}{L}\sum_{z^{i,l}}\log\left(p_\theta(x_i|z^{i,l})\right) \approx \log\left(p_\theta(x_i|z^{i,l})\right)$$

For Bernoulli (cross-entropy): $= \sum_{j=1}^{D} \left[x_{i,j} \log p_{i,j} + (1 - x_{i,j}) \log(1 - p_{i,j}) \right]$

For Gaussian: $=$ mean squared error

Implementation

Loss $=$ reconstruction (Bernoulli: cross-entropy / Gaussian: MSE) $+$ regularization (closed-form KL). Encoder outputs μ, σ; sample $z = \mu + \sigma \odot \epsilon$; decoder reconstructs.

the 20 chapters at a glance

1. **Goal:** maximize $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)\,dz$; both the integral and the posterior are *intractable*.
2. **Idea:** introduce $q_\phi(z|x) \approx p_\theta(z|x) \to$ derive the *ELBO*.
3. **Loss:** $\arg\min_{\theta,\phi} \sum_i -\mathbb{E}_{q_\phi}[\log p(x_i|g_\theta(z))] + D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$.
4. **Four devices:** encoder $\mathcal{N}(\mu, \sigma^2 I)$; prior $\mathcal{N}(0, I)$; Monte Carlo $L = 1$; reparameterization $z = \mu + \sigma \odot \epsilon$.
5. **KL closed form:** $\frac{1}{2}\sum_j(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1)$.
6. **Reconstruction:** Bernoulli $\to$ cross-entropy; Gaussian $\to$ MSE.
7. **Characteristics:** generates data + strong latent representations; stabler than the GAN, less sharp.

Thank you.

From AutoEncoders to VQ-VAE — Part 1 (pre-VAE) + Part 2 (VAE).