

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)

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.

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*.”

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 → the bridge to VAE (Ch. 3)
- why the AutoEncoder **fails as a generative model** — the starting point of the VAE (Ch. 3)

Cryptographic origins

The AutoEncoder: definition and components

A history of mathematical modelling

PART 1 • pre-VAE

Cryptographic origins

1 of 60

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.

⇒ 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

$$E : \text{plaintext} \rightarrow \text{ciphertext}, \quad D : \text{ciphertext} \rightarrow \text{plaintext}, \quad D \circ E = \text{id}.$$

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 ($D \circ E = \text{id}$) 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

2 of 60

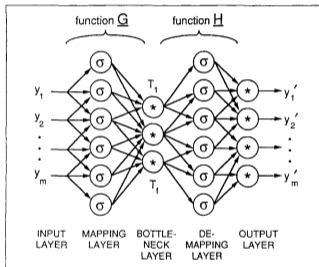


Figure 2. Network architecture for simultaneous determination of f nonlinear factors using an autoassociative network.

σ indicates sigmoidal nodes, $*$ indicates sigmoidal or linear nodes.

Fig 1. Autoassociative network: G = recognition, H = generation, the bottleneck T_1, \dots, 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, $\mathcal{X} \rightarrow \mathcal{Z}$): “recognises” the latent cause of the data.

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

Training objective (preview)

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

Definition 2.1 • AutoEncoder

A neural network of an encoder $f_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^d$ ($d < D$) and a decoder $g_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^D$. 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

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

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

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

resulting in:

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

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"):

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

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

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

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

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

Fig 2. A narrow passage compresses information \rightarrow 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

$$D_{\text{KL}}(p \parallel q) = \sum_w p(w) \log \frac{p(w)}{q(w)} \geq 0, \quad = 0 \iff p = q.$$

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

Mutual information = a KL divergence

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

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

Entropy form

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

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 $X = \text{data}$, $T = z$ 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)	$z = f_\phi(x) = F(W_1x)$	$q_\phi(z x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi^2(x)I)$
generation (decoder)	$\hat{x} = g_\theta(z) = W_2z$	$p_\theta(x z)$ (Gaussian / Bernoulli)
activation F	identity \Rightarrow Linear AE sigmoid/tanh/ReLU \Rightarrow Nonlinear AE	
latent vector	$z \in \mathbb{R}^p, z_j = F(\sum_i W_1(j, i)x_i)$	$z \sim q_\phi(z x)$ (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	$z = W_e x$	$p(z x)$ derived (Bayes)	$z = G_\phi(x)$	$q_\phi(z x)$ learned
generation	$\hat{x} = W_d z$	$\mathcal{N}(Wz + \mu, \sigma^2 I)$	$\hat{x} = H_\theta(z)$	$p_\theta(x z)$
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.

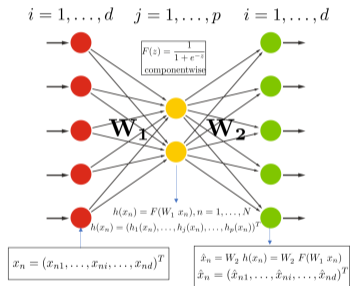


Fig. 1 Shallow undercomplete autoencoder

- 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, \dots, T_f form one latent vector.
- a latent vector is both a code and a set of coefficients over the decoder's basis vectors.

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

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

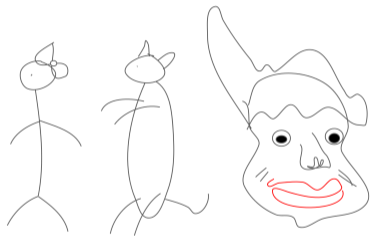
$$\|v\|_2 = \sqrt{\sum_j v_j^2}, \quad \|x - \hat{x}\|_2^2 = \sum_d (x_d - \hat{x}_d)^2$$

matrix norms (clash of notation)

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

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

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



$P(x=\text{human}) = ?$

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

AutoEncoder reconstruction loss

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

- “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 $p(x | \hat{x})$

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

$$p(x | \hat{x}) = \prod_{d=1}^D \mathcal{N}(x_d | \hat{x}_d, \sigma^2), \quad \hat{x} = g_\theta(f_\phi(x))$$
$$\log p(x | \hat{x}) = -\frac{D}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_d (x_d - \hat{x}_d)^2 = -\frac{1}{2\sigma^2} \|x - \hat{x}\|_2^2 + \text{const}$$

Conclusion

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

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

$$p(x | \hat{x}) = \prod_i \hat{\pi}_i^{x_i} (1 - \hat{\pi}_i)^{1-x_i}, \quad \hat{\pi} = \text{sigm}(g_\theta(f_\phi(x)))$$
$$-\log p(x | \hat{x}) = -\sum_i [x_i \log \hat{\pi}_i + (1 - x_i) \log(1 - \hat{\pi}_i)]$$

\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, $\log p(x | \hat{x})$ 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** $p(x | \hat{x}) = \delta(x - \hat{x})$ — 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

3 of 60

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	$P = W_d W_e, \hat{x} = P x$

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.

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

P is always rank-deficient

$\text{rank } W_e \leq d$ and $\text{rank } W_d \leq d$, so $\text{rank } P \leq d < D$:
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 $X = [x_1, \dots, x_N]$ and $\|M\|_F^2 = \text{tr}(MM^\top)$,

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

Trace form

$$\mathcal{L}(W_e, W_d) = \text{tr} \left((I - W_d W_e) \Sigma_x (I - W_d W_e)^\top \right)$$

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

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

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

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:

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

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

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

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

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

Global optimum

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

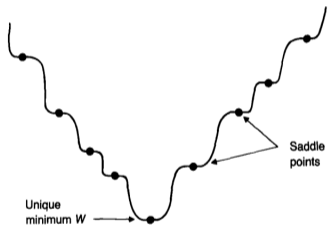
FIGURE 2. The landscape of E .

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, $P = U_{\mathcal{J}}U_{\mathcal{J}}^T$ 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 \rightarrow 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* $\text{col}(U_{1:d})$, 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

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

$p(x)$ by integrating out z (3.2.4); the encoder $p(z | x)$ 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**.

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

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

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

Derived (everything else)

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

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

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

Only the *right-going* arrow is a modelling choice. The *left-going* encoder $p(z | x)$ 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)

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

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}

$p(x | z) = \mathcal{N}(x | Wz + \mu, \sigma^2 I)$, so $f_{\text{dec}}(z) = Wz + \mu = \mathbb{E}[x | z]$ 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 $z \perp \epsilon$:

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

A linear combination of Gaussians is Gaussian:

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

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

39

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

$$p(z | x) \propto p(x | z)p(z) \propto \exp\left(-\frac{1}{2\sigma^2}\|y - Wz\|^2 - \frac{1}{2}z^\top z\right).$$

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

$$-\frac{1}{2}z^\top\left(\frac{1}{\sigma^2}W^\top W + I\right)z + \frac{1}{\sigma^2}z^\top W^\top y, \quad \frac{1}{\sigma^2}W^\top W + I = \frac{1}{\sigma^2}\underbrace{(W^\top W + \sigma^2 I)}_{=M}.$$

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

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

Encoder (posterior) — a result, nothing assumed

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

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

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

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

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

$$\sum_n (x_n - \mu)^\top C^{-1} (x_n - \mu) = \text{tr} \left(C^{-1} \underbrace{\sum_n (x_n - \mu)(x_n - \mu)^\top}_{=NS} \right) = N \text{tr}(C^{-1}S).$$

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

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 :

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

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

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 $y_n = x_n - \mu$)

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

M-step = retrain decoder

$$W_{\text{new}} = \left[\sum_n y_n \mathbb{E}[z_n]^\top \right] \left[\sum_n \mathbb{E}[z_n z_n^\top] \right]^{-1}$$

$$\sigma_{\text{new}}^2 = \frac{1}{ND} \sum_n \left\{ \|y_n\|^2 - 2 \mathbb{E}[z_n]^\top W_{\text{new}}^\top y_n + \text{tr}(\mathbb{E}[z_n z_n^\top] W_{\text{new}}^\top W_{\text{new}}) \right\}$$

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

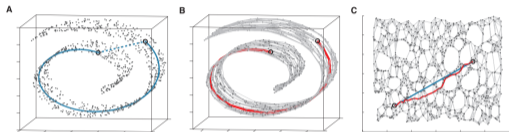


Fig. 3. The “Swiss roll” data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (A) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their 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).

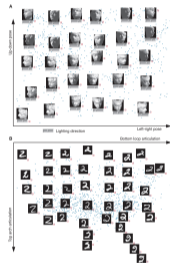


Fig 7. Isomap: geodesic distance “unfolds” the manifold.

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

- PCA / Linear AE see only **2nd-order (covariance)** structure. But *uncorrelated* \neq *independent*.
- e.g. $t \sim \text{Unif}[0, 2\pi)$, $x = (\cos t, \sin t)$: 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.

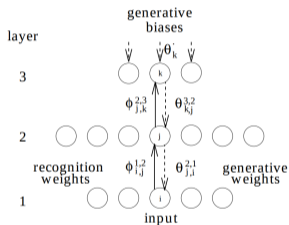


Figure 3: A Helmholtz Machine.

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

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

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

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

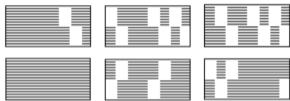


Figure 1: Shift Patterns. In each of these six patterns the bottom row of square pixels is a random binary vector, the top row is a copy shifted left or right by one pixel with wrap-around, and the middle two rows are copies of the outer rows. The patterns were generated by a two-stage process. First 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:

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

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

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

$$p_{\theta}(z_j^l = 1 | z^{l+1}) = \text{sigm}\left(b_j^l + \sum_k W_{jk}^l z_k^{l+1}\right).$$

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 $p_{\theta}(z | x)$, are intractable.
- EM (which needs the exact posterior) cannot be used.

⇒ 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*:

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

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

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

— \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.

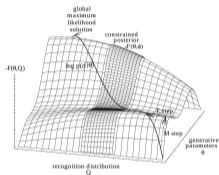


Figure 2: Graphical view of our approximation. The surface shows a simplified example of $-F(\theta, Q)$ as a function of the generative parameters θ and the recognition distribution Q . As discussed by Neal and Hinton (1998), the Expectation-Maximization algorithm ascends this surface by optimizing 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 $Q_t = P_t$, 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 (labelled “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.

This EM gets stuck in a local maximum here is largely for graphical convenience, although neither θ , 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.

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

$$E_\alpha = -\log [p(\alpha | \theta) p(d | \alpha, \theta)]$$

- the true posterior P_α is Boltzmann; its free energy equals $-\log p(d | \theta)$ 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).

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

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**).

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

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

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 $\nabla_\phi \log q_\phi(z | x)$

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 $D_{\text{KL}}(p_\theta \| q_\phi)$ on dreamed data, not the $D_{\text{KL}}(q_\phi \| p_\theta)$ 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:

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

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.

⇒ Variational AutoEncoder (VAE)

AutoEncoder — point

$$x \mapsto z = f_\phi(x) \text{ (one point)}$$



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

VAE — distribution

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

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*.

deterministic \rightarrow probabilistic, point \rightarrow 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 <i>derived</i> 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.

Everything here precedes the VAE

- the same **six components**, filled deterministically or probabilistically, drive the whole development;
- Linear AE = PCA \rightarrow PPCA (encoder *derived* by Bayes) \rightarrow Deep AE (nonlinear) \rightarrow 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**:

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

\rightarrow VAE, β -VAE, VQ-VAE.

Thank you.