

Autoencoders

Aayush Bajaj

July 10, 2026

Contents

1	the undercomplete autoencoder	1
1.1	the linear case is pca — stated carefully	2
2	regularised autoencoders	2
2.1	denoising autoencoders	3
2.2	sparse autoencoders	3
2.3	contractive autoencoders	3
3	the variational autoencoder	3
3.1	the elbo, by jensen	4
3.2	the reparameterisation trick	4
3.3	the gaussian kl in closed form	4
3.4	beta-vae	4
4	code: a linear autoencoder rediscovers pca	5
5	see also	6

an autoencoder is a network trained to do the one thing that sounds useless: output its own input. the trick is the obstacle course in the middle — a bottleneck, a corruption, a penalty — that makes verbatim copying impossible, so the network is forced to learn *what about the input is worth keeping*. the family tree below runs from the linear special case (which is **pca** wearing a trenchcoat) to the variational autoencoder, which turns the whole construction into a generative model (Goodfellow, Ian, 2016).

supervision without labels: the data is its own target, and the architecture is the teacher

1 the undercomplete autoencoder

two maps and a loss. an encoder f compresses, a decoder g reconstructs, and training minimises the reconstruction error

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n \|x_i - g(f(x_i))\|^2, \quad f : \mathbb{R}^d \rightarrow \mathbb{R}^m, \quad g : \mathbb{R}^m \rightarrow \mathbb{R}^d, \quad (1)$$

with $m < d$: the code $h = f(x)$ is *undercomplete*, so the identity function is unrepresentable and the network must ration its m numbers on the directions of greatest structure. with deep nonlinear f and g , the model learns a curved m -dimensional manifold that hugs the data — nonlinear pca in the honest sense of the phrase.

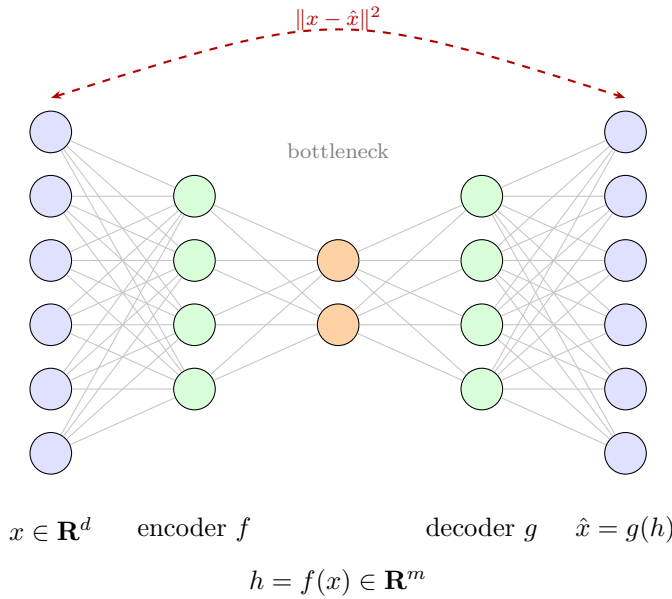


Figure 1: the hourglass: encoder squeezes d dimensions into an m -dimensional code, decoder inflates back. reconstruction error is measured between the two rims.

1.1 the linear case is pca — stated carefully

strip the nonlinearities: $f(x) = W_e x$, $g(h) = W_d h$, squared error, centred data. then the optimal reconstruction subspace is exactly the pca subspace, but the equivalence has fine print worth getting right:

- at any global optimum, the columns of W_d span the subspace of the top m principal components, and the achieved loss equals the sum of the discarded eigenvalues $\sum_{j>m} \lambda_j$ — the same optimum pca attains.
- the individual weight vectors need *not* be the eigenvectors: any invertible $m \times m$ mixing A leaves the model invariant $(W_d A^{-1}, A W_e)$, so the network identifies the principal *subspace*, not the principal *axes*, and codes are not decorrelated or variance-ordered as pca's are.
- the loss surface has no spurious local minima — every additional critical point is a saddle corresponding to picking a non-top subset of eigenvectors (baldi and hornik, 1989).¹
- gradient descent therefore reliably finds the pca solution up to that mixing — which is precisely what the experiment at the bottom of this page verifies numerically.

the corollary: depth and nonlinearity are not decoration. a linear autoencoder of any depth collapses to this same solution; everything genuinely new about autoencoders lives in the nonlinear case.

2 regularised autoencoders

capacity control by bottleneck is blunt. the alternative family keeps the code wide — even *overcomplete*, $m \geq d$ — and prevents copying with a penalty or a corruption instead (Goodfellow, Ian, 2016).

¹p. baldi and k. hornik (1989), *neural networks and principal component analysis: learning from examples without local minima*, neural networks 2(1).

2.1 denoising autoencoders

corrupt the input, demand the clean version back:

$$\mathcal{L} = \mathbb{E}_{x, \tilde{x} \sim C(\tilde{x}|x)} \|x - g(f(\tilde{x}))\|^2, \quad (2)$$

with C a corruption process — gaussian noise, or masking a random subset of coordinates to zero. the identity is no longer a solution even with infinite capacity, because $\tilde{x} \neq x$. what the network learns instead is a map from off-manifold points back *onto* the data manifold; for small gaussian corruption the optimal reconstruction points along the gradient of the log-density, so the dae implicitly estimates the shape of the data distribution.

masking noise is the ancestor of masked-token pretraining: bert is a denoising autoencoder over text

2.2 sparse autoencoders

keep the code big but mostly asleep. with sigmoid hidden units, let $\hat{\rho}_j$ be unit j 's average activation over the training set, pick a small target ρ (say 0.05), and add a kl-divergence penalty between two bernoulli distributions:

$$\mathcal{L}_{\text{sparse}} = \mathcal{L}_{\text{recon}} + \beta \sum_{j=1}^m \left[\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right]. \quad (3)$$

each unit is pushed to fire rarely, so any single input touches only a few of them — a distributed dictionary where codes are sparse combinations of learned features. (the cheaper cousin: an ℓ_1 penalty $\beta \sum_j |h_j|$ directly on the code.)

2.3 contractive autoencoders

penalise the encoder's sensitivity itself, via the frobenius norm of its jacobian:

$$\mathcal{L}_{\text{cae}} = \mathcal{L}_{\text{recon}} + \lambda \left\| \frac{\partial f}{\partial x}(x) \right\|_F^2 = \mathcal{L}_{\text{recon}} + \lambda \sum_{j,k} \left(\frac{\partial h_j}{\partial x_k} \right)^2. \quad (4)$$

the penalty wants f locally constant; reconstruction wants f to preserve information. the compromise: the encoder becomes flat in directions *orthogonal* to the data manifold (noise is contracted away) while staying sensitive along it (structure survives). denoising and contracting are two prices for the same product — local robustness — one paid stochastically, one analytically.

3 the variational autoencoder

everything above gives representations; none of it gives a principled way to *generate*. the vae (kingma and welling, 2013, *auto-encoding variational bayes*) fixes that by making the story probabilistic from the start: a prior $p(z) = \mathcal{N}(0, I)$ over latents, a decoder network defining $p_\theta(x | z)$, and the ambition to maximise the marginal likelihood

$$\log p_\theta(x) = \log \int p_\theta(x | z) p(z) dz, \quad (5)$$

which is intractable — as is the posterior $p_\theta(z | x)$ needed by em. the move: introduce an *encoder* network $q_\phi(z | x) = \mathcal{N}(\mu_\phi(x), \text{diag}(\sigma_\phi^2(x)))$ as a learned approximate posterior.

3.1 the elbo, by jensen

multiply and divide inside the marginal by q_ϕ , then push the (concave) logarithm through the expectation with jensen’s inequality:

$$\begin{aligned} \log p_\theta(x) &= \log \mathbb{E}_{z \sim q_\phi(z|x)} \left[\frac{p_\theta(x | z) p(z)}{q_\phi(z | x)} \right] \\ &\geq \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log \frac{p_\theta(x | z) p(z)}{q_\phi(z | x)} \right] \\ &= \underbrace{\mathbb{E}_{q_\phi} [\log p_\theta(x | z)]}_{\text{reconstruction}} - \underbrace{D_{\text{KL}}(q_\phi(z | x) \| p(z))}_{\text{regulariser}} = \mathcal{L}_{\text{ELBO}}(x; \theta, \phi). \end{aligned}$$

the slack in jensen is exactly $D_{\text{KL}}(q_\phi(z | x) \| p_\theta(z | x)) \geq 0$: the bound is tight precisely when the encoder matches the true posterior. maximising the elbo therefore does two jobs at once — pushes the likelihood up, and squeezes the approximate posterior toward the real one. and squint at the two terms: reconstruction error plus a penalty keeping codes near a fixed prior — the vae is an autoencoder whose regulariser fell out of a probability argument rather than being bolted on.

3.2 the reparameterisation trick

the elbo contains an expectation over $z \sim q_\phi$, and ϕ lives inside the sampling distribution — you cannot backpropagate through a random draw. rewrite the sample as a deterministic function of ϕ and parameter-free noise:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \quad (6)$$

which has exactly the law $q_\phi(z | x)$, but now gradients flow through μ_ϕ and σ_ϕ while the randomness sits harmlessly in ε . a single monte carlo sample per datum gives a low-variance unbiased gradient estimate, and the whole model trains end-to-end by sgd.

3.3 the gaussian kl in closed form

for $q = \mathcal{N}(\mu, \text{diag}(\sigma^2))$ against $p = \mathcal{N}(0, I)$, the kl term needs no sampling at all:

$$D_{\text{KL}}(q \| p) = \frac{1}{2} \sum_{j=1}^m (\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1), \quad (7)$$

each term zero iff $\mu_j = 0, \sigma_j = 1$: unused latent coordinates are shrunk exactly onto the prior. generation is then two cheap steps: draw $z \sim \mathcal{N}(0, I)$, decode.

3.4 beta-vae

scale the kl term by $\beta > 1$,

when the decoder is powerful enough to ignore z entirely, all coordinates collapse onto the prior — posterior collapse, the vae’s characteristic failure mode

$$\mathcal{L}_\beta = \mathbb{E}_{q_\phi} [\log p_\theta(x | z)] - \beta D_{\text{KL}}(q_\phi(z | x) \| p(z)), \quad (8)$$

and the model trades reconstruction fidelity for a code squeezed harder against the isotropic prior — empirically encouraging *disentangled* latents, one factor of variation per coordinate (higgins et al., 2017). β is a dial between the two things an autoencoder can be: at low β a compressor, at high β an explainer.

4 code: a linear autoencoder rediscovers pca

2-d data with one dominant direction, squeezed through a 1-d code. gradient descent on reconstruction error, then compare the learned subspace against the eigendecomposition:

```

1  import numpy as np
2
3  rng = np.random.default_rng(0)
4
5  # --- synthetic 2d data with one dominant direction ---
6  n = 500
7  true_dir = np.array([2.0, 1.0]) / np.sqrt(5.0)          # unit principal
8  ↪ direction
9  z = rng.normal(0, 2.0, n)                                # large variance along
10 ↪ it
11 noise = rng.normal(0, 0.3, (n, 2))                      # small isotropic
12 ↪ residue
13 X = np.outer(z, true_dir) + noise
14 X -= X.mean(axis=0)                                     # centre the data
15
16 # --- pca answer, for reference ---
17 C = X.T @ X / n
18 eigvals, eigvecs = np.linalg.eigh(C)
19 v_pca = eigvecs[:, -1]                                  # top principal
20 ↪ direction
21
22 # --- linear autoencoder: 2 -> 1 -> 2, no biases, untied weights ---
23 W_enc = rng.normal(0, 0.1, (1, 2))                     # encoder
24 W_dec = rng.normal(0, 0.1, (2, 1))                     # decoder
25 lr = 0.05
26 for step in range(2001):
27     h = X @ W_enc.T                                     # codes      (n, 1)
28     Xr = h @ W_dec.T                                    # reconstructions
29     err = Xr - X
30     loss = (err ** 2).sum(axis=1).mean()
31     if step % 400 == 0:
32         print(f"step {step:>4} reconstruction mse {loss:.4f}")
33         dW_dec = 2 * err.T @ h / n                      # (2, 1)
34         dW_enc = 2 * (err @ W_dec).T @ X / n           # (1, 2)
35         W_dec -= lr * dW_dec
36         W_enc -= lr * dW_enc
37
38 # the learned decoder column spans the model's 1-d reconstruction subspace
39 v_ae = W_dec[:, 0] / np.linalg.norm(W_dec[:, 0])
40 align_pca = abs(v_ae @ v_pca)
41 align_true = abs(v_ae @ true_dir)
42 print(f"\npca direction:      {np.round(v_pca, 4)}")
43 print(f"ae direction:        {np.round(v_ae, 4)}")
44 print(f"|cos| ae vs pca:      {align_pca:.6f}")
45 print(f"|cos| ae vs truth:    {align_true:.6f}")
46 print(f"residual variance (pca optimum): {eigvals[0]:.4f}")

```

step 0 reconstruction mse 4.3774

```
step 400 reconstruction mse 0.0877
step 800 reconstruction mse 0.0877
step 1200 reconstruction mse 0.0877
step 1600 reconstruction mse 0.0877
step 2000 reconstruction mse 0.0877

pca direction:      [-0.8919 -0.4522]
ae direction:       [-0.8919 -0.4522]
|cos| ae vs pca:    1.000000
|cos| ae vs truth:  0.999984
residual variance (pca optimum): 0.0877
```

the theory section, verified line by line:

- the learned decoder direction matches the top principal component to six decimal places (cosine similarity 1.000000), and both sit within sampling noise of the true generating direction.
- the converged reconstruction mse, 0.0877, *equals* the discarded eigenvalue λ_2 — the pca optimum — to four decimals. gradient descent found the global minimum of the linear autoencoder, as baldi–hornik promise.
- absolute values are used in the comparison because the subspace, not the sign, is identified: negate encoder and decoder together and nothing changes.

5 see also

- [pca](#) — the linear case, done with eigenvectors instead of gradients
- [generative adversarial networks](#) — generation by game instead of by likelihood bound
- [stable diffusion](#) — a vae compresses the pixels; diffusion happens in its latent space
- [feedforward deep neural networks](#) — the machinery inside f and g

references

Goodfellow, Ian (2016). *Deep Learning*, MIT Press.