Deep Learning: cheat sheet

Chapters 1 to 23 of 2026.2, one thread per page: 1 data, 2 metrics, 3 learning and optimization, 4 regularization, depth and convolutions, 5 attention and Transformers, 6 adapting models and LLMs, 7 and 8 generative models, 9 chapter by chapter. The numbers are the ones measured in the classes; the small number on each bar is the chapter.



The one rule 2

Nothing that learns a number from the data may be fitted before the split.

fitted, so inside the `Pipeline`	not fitted: before is fine
imputer, scaler, category vocabulary, target encoding, k-best selection, PCA, oversampling (training fold only)	plots, `describe()`, dropping exact duplicates, fixing a unit or a sentinel

Trap: a hand-written `scaler.fit(X_train)` is fitted once while the folds change five times. Only a `Pipeline` survives cross-validation.

Train, validation, test 2

set	for	look
train	fitting the weights	always
validation	every choice: architecture, η, when to stop, features	freely, it is consumed
test	one number no choice was based on	**once**

Split by what the model must generalize to

`stratify=y`	classification, always; essential with a rare class
`GroupKFold`	one patient, user or device makes several rows
`TimeSeriesSplit`	ordered rows: the test set is the future

- An interval narrows like $1/\sqrt{n}$: 200 test rows cannot tell 0.90 from 0.94 (Wilson: 0.851–0.934 against 0.898–0.965).
- Sizes: <10k rows, CV plus a held-out test; 10k–1M, 80/10/10; >1M, 98/1/1.
- One 80/20 split of the same model ranged 0.717–0.875 over 120 seeds; 5 folds cut the sd from 0.032 to 0.006.
- Best of 40 equivalent models on 200 rows: +1.4 points of pure noise. Nested CV, or a final holdout.

Data leakage 2

Would I have had *exactly this* at the moment of the prediction?

it enters through	measured in the lab	honest
a column	a consequence of the answer; one feature with $\lvert r\rvert > 0.95$ is a leak until proven otherwise	
selection before the split	5 000 noise columns, keep 20: 0.870	0.500
the split (groups)	`StratifiedKFold` 0.986	0.708
time	`rolling(3, center=True)`: 0.937	0.716
the pipeline	scaling early $+0.004$ AUC; oversampling early $+0.173$	

Trap: the cheap leak is the famous one. Selection, groups and resampling are invisible in the code and raise no error.

Feature types: an encoding is a claim 2

type	order	distance	safe encoding
continuous, count	yes	yes	scale (counts: `log1p` first)
ordinal (S/M/L)	yes	no	integers, with `categories=` explicit
cyclical (hour)	yes	circular	$\sin(2\pi h/24)$ and $\cos(2\pi h/24)$
nominal (city)	no	no	one-hot, or an embedding
identifier	no	no	**not a feature**: it defines the groups

Trap: label-encoding a nominal column: R^2 0.163 at 40 categories, against 0.94 with one-hot.

Scaling 2

scaler	formula	fitted from, behaviour
Standard	$(x-\mu)/\sigma$	mean, sd; unbounded
MinMax	$(x-\min)/(\max-\min)$	extremes; one typo crushes the rest
Robust	$(x-\text{med})/\text{IQR}$	quartiles; ignores the typo
MaxAbs	$x/\max\lvert x\rvert$	keeps zeros: sparse data
Quantile	rank, then map	only the order survives

- Scale every numeric input for anything gradient-based: unscaled 0.605, any scaler 0.89–0.91. Trees do not need it.
- Sigmoid or tanh in the first layer: MinMax to $[0,1]$ or $[-1,1]$. Sparse: never centre.

Trap: `fit_transform` on the test set. The test set is transformed, never fitted.

Data quality 2

missing	who disappears, and the cost
MCAR	nobody in particular: only sample size
MAR	a group other columns identify: fixable with them
MNAR	defined by the missing value itself: a bias the data cannot reveal

- Mean imputation cannot fix a bias and shrinks the spread (to 0.30 of the truth under MNAR). Keep a `was_missing` column.
- Sentinels pose as data: `-999`, `0`, `"N/A"`, `1970-01-01`.
- Outliers: the z-score is computed with the statistics the outliers corrupted (at 10% contamination it found 0 of 100); IQR and the modified z-score (median, MAD) hold.
- Duplicates on both sides of the split are an answer key: de-duplicate before splitting.
- A network fits 40% wrong labels to training accuracy 1.000 while the honest score falls from 0.951 to 0.639; early stopping brings it back to 0.825.

Class imbalance 2, 8

- **Fix the metric**: from 30% to 0.5% positives, the trivial accuracy rises 0.700→0.995, ROC-AUC stays 0.968→0.967, average precision falls 0.934→0.457.
- **Fix the threshold**: from the costs, $p^\star = C_{FP}/(C_{FP}+C_{FN})$, chosen on validation.
- Class weights, oversampling, SMOTE and undersampling did not move the ranking (AP 0.758–0.761) and broke the probabilities (Brier 0.0088→0.0360).
- Deep learning: `pos_weight` $\approx N_{neg}/N_{pos}$ or focal loss when negatives dominate the gradient; balanced batches only when batches come up empty.

Trap: resampling before the split; and a resampled model's probabilities, which need Elkan's correction of the prior.

High dimension 2

shell	a fraction $1-(1-\varepsilon)^d$ of the volume; median radius $(1/2)^{1/d}$
ball/cube	$\pi^{d/2}/(\Gamma(d/2{+}1)\,2^d) \to 0$
distances	concentrate when the extra coordinates are i.i.d. noise: k-NN, k-means, t-SNE lose contrast
angles	random vectors become orthogonal, cosine sd $\approx 1/\sqrt{d}$

What governs is the **intrinsic** dimension, not the column count. Select features, or reduce while keeping the signal.

Dimensionality reduction 2

PCA: $\max_{\|w\|=1} w^\top \Sigma w \;=\; \min_{\|w\|=1} \frac{1}{n}\sum_i \|x_i - (x_i^\top w)w\|^2$

	PCA	t-SNE	UMAP
preserves	global variance	neighbourhoods	local, some global
`transform` new rows	yes	no	approximate
parameter	k	perplexity	`n_neighbors`
use	compress, preprocess	look	look at scale

- `PCA(n_components=0.95)` keeps 95% of the variance; it is refitted in every fold.
- Hundreds of columns: PCA to about 50 before t-SNE or UMAP.

Trap: reading a t-SNE map: cluster size and the distance between clusters mean nothing, the axes mean nothing, and t-SNE coordinates are not features.

Reading a distribution 2

- Two classes along one feature: $d' = \lvert\mu_B - \mu_A\rvert / \sqrt{(\sigma_A^2+\sigma_B^2)/2}$, and the best possible accuracy is $\Phi(d'/2)$.
- Skewed (incomes, counts, durations): `log1p`. A ceiling in the histogram is a property of the data (784 districts at 500 001).
- One feature at a time is not two at once: salmon and sea bass separate in the plane, not on either axis.

Before you believe a score 2

- What does the trivial model get? Report it next to yours.
- Is there a group (patient, user, session) with more than one row?
- Was anything fitted, selected or resampled before the split?
- Would every feature exist, with this value, at prediction time?
- A sudden jump is an incident to investigate: *what does this model know that it should not?*

Metrics and evaluation

Chapter 8: classification, regression, generative models and LLMs. A metric is a decision about how to add up the wrongness.

The confusion matrix 8.1

	actually positive	actually negative
predicted positive	TP: hit	FP: false alarm (type I)
predicted negative	FN: miss (type II)	TN: correct rejection

$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{Recall} = \frac{TP}{TP+FN} \quad \text{Specificity} = \frac{TN}{TN+FP}$$

$$F_1 = \frac{2PR}{P+R} \qquad F_\beta = (1+\beta^2)\frac{PR}{\beta^2 P+R}$$

$$\text{MCC} = \frac{TP\cdot TN - FP\cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

- Read the denominator: precision divides by what the *model* claimed, recall by what *reality* holds.
- Balanced accuracy = (recall + specificity)/2. Prevalence = $(TP+FN)/N$.
- F_2: recall weighs twice (screening). $F_{0.5}$: precision weighs twice (search). F1 never sees TN; MCC uses all four cells.

Trap: 99% accuracy on a disease that hits 1 in 100 is the score of `return 0`. Ask first what the trivial classifier gets.

Scores, thresholds and curves 8.1

$$\text{AUC} = P\big(s(x^+) > s(x^-)\big) \qquad \text{AP} = \textstyle\sum_n (R_n - R_{n-1})P_n$$

- ROC plots recall against FPR $= FP/(FP+TN)$: blind to prevalence. With rare positives ROC flatters, PR tells the truth; AP's baseline is the prevalence.
- The threshold is not part of the model. Expected costs give $p > p^\star = C_{FP}/(C_{FP}+C_{FN})$: a miss that costs $9\times$ a false alarm means flag above 10%.

Trap: tuning the threshold on the test set. Choose it on validation, report it on test.

Which metric 8.1

situation	use
balanced, errors cost the same	accuracy
a miss is expensive (disease, fraud)	recall, F_2
a false alarm is expensive (spam)	precision, $F_{0.5}$
rare positives, comparing models	PR-AUC (AP)
comparing rankers, roughly balanced	ROC-AUC
the probability is used as a number	log loss, Brier, calibration
imbalanced multi-class	macro-F1, balanced accuracy

Multi-class: the averaging *is* the metric

Per-class F1 of 0.970, 0.675 and 0.261 (support 900, 90, 10): micro 0.938 (= accuracy), weighted 0.937, **macro 0.636**, the only one that reports the rare class failing.

Probabilities 8.1

$$\text{LogLoss} = -\tfrac{1}{N}\textstyle\sum\big[y\log\hat{p} + (1-y)\log(1-\hat{p})\big]$$

$$\text{Brier} = \tfrac{1}{N}\textstyle\sum(\hat{p}-y)^2, \text{ bounded in } [0,1]$$

$$\text{ECE} = \textstyle\sum_m \frac{|B_m|}{N}\big|\text{acc}(B_m) - \text{conf}(B_m)\big|$$

- One confident error: $-\log 0.01 = 4.6$ against 0.92 for predicting 0.4.
- Networks are usually overconfident. **Temperature scaling** divides the logits by one T fitted on validation: AUC and accuracy do not move.

Trap: metrics and losses are not the same thing: a loss must also be differentiable.

Regression 8.2

$$\text{MAE} = \tfrac{1}{N}\textstyle\sum|e_i| \quad \text{MSE} = \tfrac{1}{N}\textstyle\sum e_i^2 \quad \text{RMSE} = \sqrt{\text{MSE}} \;\geq\; \text{MAE}$$

$$R^2 = 1 - \frac{\sum(y_i-\hat{y}_i)^2}{\sum(y_i-\bar{y})^2} \qquad R^2_{adj} = 1-(1-R^2)\frac{N-1}{N-k-1}$$

- MSE is minimized by the **mean**, MAE by the **median**: the metric you optimize is the model you get. MedAE ignores any single outlier.
- RMSE/MAE ≈ 1.25 for Gaussian errors; far above it, a few huge misses. The cheapest smoke alarm.
- R^2 compares with predicting $\bar{y}$: it is negative when you do worse, common on test data, and not comparable across datasets.
- Huber: $\frac{1}{2}e^2$ for $|e| \leq \delta$, $\delta(|e| - \frac{1}{2}\delta)$ beyond. Quantile (pinball) loss: τe if $e \geq 0$, $(\tau-1)e$ if not; $\tau = 0.1, 0.5, 0.9$ give an interval.

Trap: MAPE: undefined at $y=0$, asymmetric, so it biases models to predict low. Prefer MASE (<1 beats the naive forecast) or WAPE $= \sum|e| / \sum|y|$.

Trap: no baseline. Report the mean, the median or, with time, the last value next to the model.

Generative models 8.3

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\big)$$

$$\text{CLIPScore} = w\cdot\max\big(0, \cos(e_{\text{img}}, e_{\text{text}})\big)$$

- BLEU (n-gram precision, brevity penalty, geometric mean) and ROUGE (n-gram recall) compare strings: a perfect paraphrase scores BLEU 0.00, adding a "not" keeps ROUGE-1 at 1.00.
- BERTScore compares contextual embeddings, and imports the encoder's biases.
- FID fits **one Gaussian** per cloud: a one-mode model matching a two-mode reality scored 0.009. It falls with the sample count and depends on the feature extractor.
- Precision and recall of distributions split FID's number: **mode dropping** lowers recall (GANs), **blur** lowers precision (VAEs).
- FID for realism, CLIP score for prompt alignment; CLIP is weak at composition. Then pairwise human preference.

Trap: reporting only a number. Look at twenty samples.

Language models 8.4

$$\text{PPL} = \exp\Big(-\tfrac{1}{N}\textstyle\sum_i \log p_\theta(w_i \mid w_{<i})\Big)$$

$$\text{pass@}k = \mathbb{E}_{\text{problems}}\Big[1 - \tbinom{n-c}{k}\Big/\tbinom{n}{k}\Big], \; c \text{ of } n \text{ samples pass}$$

$$P(A \text{ beats } B) = \frac{1}{1+10^{(R_B-R_A)/400}} \qquad R_A \leftarrow R_A + K\,(S_A - \mathbb{E}[S_A])$$

- A uniform guess over V tokens has perplexity V. Lower is better, but only with the **same tokenizer**; otherwise compare bits per byte.
- Benchmarks (MMLU, HumanEval, GSM8K…) measure tasks; a score is evidence only if the benchmark is newer than the training cutoff (**contamination**).
- Codex: 28.8% at pass@1, 70.2% at pass@100. pass@k without its k means nothing.
- LLM judges agree with people about 80% of the time. Biases: position (judge both orders), verbosity, self-preference, scores bunched at 7–9 (prefer pairwise).
- RAG: retrieval (context precision and recall) and generation (faithfulness, answer relevancy), measured separately.

Trap: a single safety rate. Refusal needs over-refusal next to it: a model that refuses everything scores perfectly.

Choosing, by question 8.2–8.4

question	use
errors symmetric, no wild outliers	RMSE
a real tail you must not chase	MAE, MedAE
beat a do-nothing baseline?	R^2, MASE
a range, not a point	quantile loss
heavy-tailed target, training a network	Huber / SmoothL1
translation	chrF or COMET; BLEU for continuity
summary	ROUGE plus a factuality check
text-to-image	FID + CLIP score + people
unconditional images	FID + precision/recall, n stated
is the corpus being learned?	perplexity, same tokenizer
code that works	pass@1; pass@k only with a verifier
is answer A better than B?	pairwise judge, both orders
do users prefer it?	A/B with users, Arena Elo

What every metric is 8

- A **proxy**: know what it stands in for and you know where it breaks.
- A **specification**: minimizing MSE asks for the conditional mean; F1 says precision and recall matter equally. If you can name the cost ratio, use it.
- Reported **with its baseline** and, for anything sampled, with its spread over seeds or folds.

Neurons, backpropagation and optimization

Chapters 1 and 3 to 6: from one neuron to the optimizer that trains a network of millions.

Where it sits 1, 3

- AI paradigms: **symbolic** (rules and logic), **connectionist** (neural networks: learn from data), **neuro-symbolic** (both).
- AI ⊃ machine learning ⊃ neural networks ⊃ deep learning. Supervised: inputs with labels. Unsupervised: structure without labels. Semi- and self-supervised in between.
- 1943 McCulloch–Pitts neuron; 1958 perceptron; 1986 backpropagation; 1989 universal approximation (Cybenko: one hidden layer approximates any continuous function on a compact set).

Perceptron 4

$\hat{y} = H(w \cdot x + b)$, with $H(z) = 1$ if $z \geq 0$, else 0

$w \leftarrow w + \eta\,(y - \hat{y})\,x \qquad b \leftarrow b + \eta\,(y - \hat{y})$

- Updates only on mistakes. One pass over the data is an **epoch**.
- Converges in a finite number of steps **if and only if** the data are linearly separable; otherwise it oscillates: cap the epochs.
- No probabilities, binary only (one-vs-rest for more), sensitive to η.

Trap: XOR. No line separates it, so no perceptron learns it: it takes a hidden layer.

Multi-layer perceptron 5

$h = \sigma(Wx + b^h) \qquad \hat{y} = \sigma(Vh + b^y)$

Without the activation, two layers collapse into one: $V(Wx) = (VW)x$. The non-linearity is what buys depth.

Activations and their derivatives

name	$f(z)$	$f'(z)$
sigmoid	$1/(1+e^{-z})$, in $(0,1)$	$\sigma(1-\sigma) \leq 0.25$
tanh	$(e^{2z}-1)/(e^{2z}+1)$, in $(-1,1)$	$1 - \tanh^2$
ReLU	$\max(0,z)$	1 if $z > 0$, else 0
Leaky ReLU	$\max(\alpha z, z)$	1 or α
GELU	$\frac{1}{2}z(1+\tanh(0.798(z+0.0447z^3)))$	smooth ReLU
SiLU / Swish	$z\,\sigma(z)$	used in SwiGLU

Each sigmoid a gradient crosses shrinks it by at least 4; ReLU passes it unchanged where the unit is active (and a dead ReLU passes nothing).

Losses 5, 8, 15

regression	MSE $\frac{1}{N}\sum(y-\hat{y})^2$; MAE; Huber
binary	BCE $-[y \log \hat{p} + (1-y)\log(1-\hat{p})]$ on a sigmoid
K classes	cross-entropy $-\log p_y$, $p = \text{softmax}(z)$; $\partial L/\partial z = p - y$

In PyTorch pass **logits**: `BCEWithLogitsLoss`, `CrossEntropyLoss` (they apply the sigmoid or softmax themselves, stably).

Backpropagation 5

The chain rule, from the loss back to every weight ($u = Vh + b^y$, $z = Wx + b^h$):

$$\delta_y = \frac{\partial L}{\partial u} = \frac{\partial L}{\partial \hat{y}}\,\sigma'(u) \qquad \frac{\partial L}{\partial V} = \delta_y\, h^\top \qquad \frac{\partial L}{\partial b^y} = \delta_y$$

$$\delta_h = (V^\top \delta_y) \odot \sigma'(z) \qquad \frac{\partial L}{\partial W} = \delta_h\, x^\top \qquad \frac{\partial L}{\partial b^h} = \delta_h$$

$$p \leftarrow p - \eta\, \partial L/\partial p$$

One step, by hand (chapter 5)

$W = \left[\begin{smallmatrix} 0.2 & 0.4 \\ 0.6 & 0.8 \end{smallmatrix}\right]$, $b^h = [0.1, 0.2]$, $V = [0.3,\ 0.5]$, $b^y = 0.4$, $x = [0.5, 0.8]$, $y = 0$, $\eta = 0.7$, MSE: $z = [0.52,\ 1.14]$, $h = [0.627,\ 0.758]$, $u = 0.967$, $\hat{y} = 0.725$, $L = 0.5249$; $\delta_y = 0.289$, $\delta_h = [0.020,\ 0.027]$; the first row of W becomes $[0.193,\ 0.389]$.

Gradient descent 6

$\theta_{t+1} = \theta_t - \eta\,\nabla J(\theta_t)$ (∇J points uphill)

- On a quadratic with curvature L it converges only if $\eta < 2/L$: monotone below $1/L$, zig-zag between $1/L$ and $2/L$, explodes above. Divergence is arithmetic, not bad luck.
- Ravines: $\kappa = L/\mu$ (condition number) sets the cost, $O(\kappa \log 1/\varepsilon)$ steps; with $\kappa = 100$, 115 steps per order of magnitude.
- The enemy is flatness (saddles, plateaus), not bad local minima.

How much data per step

Batch: all N (exact, one update per epoch). Stochastic: one sample. Mini-batch: B = 32 to 512. The gradient noise falls like $\sigma/\sqrt{B}$, so SGD ends in a **cloud** around the minimum whose radius grows with $\sqrt{\eta}$ and shrinks with $\sqrt{B}$: decay η at the end. Batch $\times k$: $\eta \times k$ (linear rule) or $\times\sqrt{k}$.

Momentum 6

$V_{t+1} = \beta V_t + (1-\beta)\nabla J(\theta_t) \qquad \theta_{t+1} = \theta_t - \eta V_{t+1}$

- Across a ravine the gradient flips sign and cancels; along it, it adds up. $1/(1-\beta)$ is the memory: 10 steps at $\beta = 0.9$.
- Nesterov: the gradient at the look-ahead point $\theta_t - \eta\beta V_t$.

Trap: PyTorch's `SGD(momentum=0.9)` uses $v \leftarrow \beta v + g$, without the $(1-\beta)$: the same direction, a step $10\times$ larger.

Adaptive methods 6

AdaGrad: $G_t = G_{t-1} + g_t^2$, step $\eta\, g_t/(\sqrt{G_t} + \epsilon)$

RMSProp: $V_{t+1} = \beta V_t + (1-\beta) g_t^2$, step $\eta\, g_t/(\sqrt{V_{t+1}} + \epsilon)$

Adam: $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

$\hat{m}_t = m_t/(1-\beta_1^t)$, $\hat{v}_t = v_t/(1-\beta_2^t)$, $\theta \leftarrow \theta - \eta\,\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$

AdamW: $\theta \leftarrow \theta - \eta\left(\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon) + \lambda\theta\right)$

- AdaGrad's sum never decreases, so its step dies; RMSProp averages instead.
- Defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. Without bias correction the first step would be $3.16\,\eta$.
- AdamW decays the weights directly; in Adam an L2 term is divided by $\sqrt{\hat{v}}$ too. Do not decay biases or norm gains.

Schedules and clipping 6

$\eta_t = \eta_{\max}\, s(t)$: **warmup** from about 0 for the first steps, then **cosine** decay to 0. **Clipping**: $g \leftarrow g \cdot \min(1, c/\|g\|)$, with $c = 1$: the direction stays, the length is capped.

In practice 6

situation	optimizer	start at
Transformer from scratch	AdamW	3e-4, $\beta = (0.9, 0.95)$, wd 0.1, warmup + cosine
fine-tuning	AdamW	2e-5 to 5e-5, wd 0.01
CNN, long run	SGD + Nesterov	0.1 (batch 256), 0.9, wd 5e-4, cosine
MLP, tabular; first try	Adam	1e-3

Sweep η over 1e-1, 1e-2, 1e-3, 1e-4 for a few hundred steps; keep the largest that does not blow up, then halve it.

symptom	try
loss `NaN`/`inf`	$\eta \div 10$, clipping, check for $\log 0$
flat from step 0	$\eta \times 10$; can it overfit 10 samples?
drops, then plateaus high	momentum, decay η, check the init
jitters violently	bigger batch, smaller η
diverges at epoch k	schedule, warmup
train fine, validation rising	not the optimizer: regularize (page 4)

The training loop 6

```
opt = torch.optim.AdamW(params, lr=3e-4, weight_decay=0.1)
for x, y in loader:
    opt.zero_grad()                 # gradients accumulate
    loss = loss_fn(model(x), y)     # logits, not probabilities
    loss.backward()
    nn.utils.clip_grad_norm_(params, 1.0)
    opt.step(); sched.step()
```

```
def adam(th, g, m, v, t, lr=1e-3, b1=.9, b2=.999, e=1e-8):
    m = b1 * m + (1 - b1) * g           # direction
    v = b2 * v + (1 - b2) * g ** 2      # scale
    mh, vh = m / (1 - b1 ** t), v / (1 - b2 ** t)
    return th - lr * mh / (np.sqrt(vh) + e), m, v
```

Regularization, depth and convolutions

Chapters 7, 9 and 10: fitting worse on purpose, keeping the signal alive through many layers, and the first architecture that is a prior.

Generalization 7

gap $= R(\theta) - \hat{R}(\theta)$, true risk minus training risk

$$\mathbb{E}\left[(y-\hat{f})^2\right] = \underbrace{(\mathbb{E}\hat{f}-f)^2}_{\text{bias}^2} + \underbrace{\mathbb{E}\left[(\hat{f}-\mathbb{E}\hat{f})^2\right]}_{\text{variance}} + \underbrace{\sigma^2}_{\text{noise}}$$

- **Underfitting**: train and validation errors both high and close. **Overfitting**: train low, validation much higher; the gap is the symptom.
- At a fixed amount of data, huge weights of alternating sign are the signature of memorizing ($\|w\|^2$: 6 at degree 3, 7×10^6 at degree 14).
- **Double descent**: past the interpolation point, test error can fall again. "Make it smaller" is bad advice for big networks.

L2 and L1 7

L2: $\tilde{J} = J + \frac{\lambda}{2}\sum w_i^2 \;\Rightarrow\; w \leftarrow (1-\eta\lambda)\,w - \eta\nabla J$ (weight decay)

L1: $\tilde{J} = J + \lambda \sum |w_i|$, gradient $\lambda\,\text{sign}(w)$

- L2 pushes proportionally and never reaches zero (a circle); L1 pushes with constant force to exactly zero (a diamond, all corners): feature selection. Both: elastic net.
- Penalize weight matrices only, not biases or norm gains. λ: 10^{-5} to 10^{-2} with SGD, 0.01–0.1 with AdamW.

Trap: with Adam, an L2 term is not weight decay. Use AdamW.

Dropout 7

$m_j \sim \text{Bernoulli}(1-p)$, $\quad \tilde{h}_j = \dfrac{m_j}{1-p}\,h_j$ (inverted: inference is a plain pass)

- Breaks co-adaptation; trains an ensemble of 2^n subnetworks sharing weights.
- Dense layers $p = 0.2$–0.5; Transformers $p = 0.1$; rarely in convolutions; never on the output.
- Put it after BatchNorm, not before: they disagree about the variance.

Trap: forgetting `model.eval()`: dropout stays on, BatchNorm keeps updating, and nothing crashes.

Normalization 7, 9, 12

$$\hat{x} = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}, \qquad y = \gamma\hat{x} + \beta \quad (\epsilon \text{ inside the root})$$

	BatchNorm	LayerNorm
statistics over	the batch, per feature	one sample's features
training	batch mean and variance	the same at inference
inference	running averages, frozen	
regularizes?	yes, through batch noise	no
where	CNNs	Transformers, any batch size

BatchNorm's backward: nudging one x_i moves μ and σ^2, so every sample's output moves; $\sum_i \partial L/\partial x_i = 0$. Its benefit is mostly optimization, not "internal covariate shift".

Early stopping and more data 7

- Evaluate every epoch; **patience**; **restore the best checkpoint**. For a linear model it is ridge with $\lambda \approx 1/(\eta t)$.
- Label smoothing: $y_k^{LS} = (1-\varepsilon)y_k + \varepsilon/K$, $\varepsilon = 0.1$.
- Mixup: $\tilde{x} = \lambda x_i + (1-\lambda)x_j$, $\tilde{y} = \lambda y_i + (1-\lambda)y_j$, $\lambda \sim \text{Beta}(\alpha,\alpha)$, $\alpha = 0.1$–0.4.
- More labelled data beats every technique here; augmentation is the next best.

the two curves	means
train ↓, val. flat then ↑	overfitting: stop early, regularize, get data
both high, close	underfitting: capacity, longer, less regularization
validation *better* than train	dropout/BN still on, or a leak
perfect validation	a leak, always

Trap: regularizing a model that cannot overfit 10 samples. That is a bug, not a variance problem.

Deep networks: signal and gradient 9

$$\mathbb{E}[a_{l+1}^2] = \underbrace{n\,\sigma^2\,\rho}_{\text{factor}}\,\mathbb{E}[a_l^2], \text{ after } L \text{ layers raised to } L$$

n = fan-in, σ^2 = weight variance, ρ = the share the activation lets through ($\frac{1}{2}$ for ReLU). A factor of 0.95 is harmless at 5 layers and 10^{-7} at 300.

activation	init	why
tanh, sigmoid	Xavier $\sigma^2 = 1/n$	$(2/(n_{in}+n_{out})$ for both passes)
ReLU	He $\sigma^2 = 2/n$	ReLU zeroes half; the 2 pays for it
last layer of a residual block	zero	the block starts as the identity

ReLU at 30 layers: naive $\mathcal{N}(0,1)$ grows $10^{39}\times$ (`inf`); Xavier shrinks to 10^{-9}; He stays near 1. Visible at initialization, before any step.

residual: $a_{l+1} = a_l + f(a_l)$ $\qquad \dfrac{\partial a_{l+1}}{\partial a_l} = 1 + \dfrac{\partial f}{\partial a_l}$

The 1 is a floor, not a ceiling: a skip alone makes signal and gradient explode, so it always comes with a normalization. Post-LN $\text{Norm}(a+f(a))$ pins the signal; pre-LN $a + f(\text{Norm}(a))$ keeps the clean identity path.

Trap: `nn.Linear`'s default init has variance $1/(3n)$, a sixth of He: a deep ReLU stack built from defaults fades.

Convolution: the arithmetic 10

$$H_{out} = \left\lfloor \frac{H_{in} + 2p - k}{s} \right\rfloor + 1 \qquad \#\text{params} = C_{out}\,(C_{in}k^2 + 1)$$

- Same padding: $p = (k-1)/2$ with odd k. Stride 2 halves the resolution.
- No H or W in the count: `Conv2d(3, 64, 3)` has 1 792 weights; a dense layer between the same tensors, ≈201 million.
- A convolution is a dense layer with **locality** and **weight sharing** imposed: a prior, not capacity.

Receptive field

Stride 1: each $k \times k$ layer adds $k-1$ (linear growth). Each stride-2 layer doubles the step of all later layers (geometric). Dilation reaches further without weights. Two 3×3 reach 5 pixels like one 5×5, with 18 weights against 25 and a non-linearity in between (VGG).

Equivariance and invariance

Weight sharing gives translation **equivariance**. A classifier wants **invariance**: global average pooling and augmentation buy it. Rotation and scale are bought only with augmentation and data.

Convolution: the backward pass 10

$\partial L/\partial b$	sum of the output gradient over batch and space
$\partial L/\partial W$	cross-correlation of the input with the output gradient
$\partial L/\partial X$	full convolution of the output gradient with the kernel ($k-1$ padding on each side)

Max pooling is a **router**: only the window's winner gets gradient (27 of 36 cells get none at 2×2, stride 2); average pooling gives each $1/k^2$. Used forwards, $\partial L/\partial X$ is `ConvTranspose2d`.

Trap: transposed convolution with a stride that does not divide the kernel: checkerboard artifacts. Upsample, then convolve.

A whole CNN 10

layer	output	params	RF
`Conv2d(1,16,3,p=1)+BN+ReLU`	$16 \times 28 \times 28$	192	3
`MaxPool2d(2)`	$16 \times 14 \times 14$	0	4
`Conv2d(16,32,3,p=1)+BN+ReLU`	$32 \times 14 \times 14$	4 704	8
`MaxPool2d(2)`	$32 \times 7 \times 7$	0	10
`Conv2d(32,64,3,p=1)+BN+ReLU`	$64 \times 7 \times 7$	18 624	18
`AdaptiveAvgPool2d(1)+Flatten`	64	0	all
`Linear(64,10)`	10	650	

Resolution down, channels up; the parameters live in the late layers. With `Flatten` instead of global pooling the head alone would cost 31 370.

What survived

3×3 stacks, residual blocks, normalization, global average pooling, 1×1 convolutions and depthwise separable ones (cost $\approx 1/C_{out} + 1/k^2$). Gone: big dense heads, LRN, large stems. ConvNeXt: much of ViT's edge was the training recipe. Convolutions still run the latent autoencoders, medical segmentation (nnU-Net), on-device vision and the ViT's patch stem.

Regularization, put together 7

mechanism	technique	where it earns its place
constrain the weights	L2, weight decay	almost always: 1e-4 (SGD), 0.01–0.1 (AdamW)
inject noise	dropout; BN; augmentation, mixup	dense and Transformer blocks; vision and audio
limit the search	early stopping	free, always
soften the target	label smoothing	many classes, $\varepsilon = 0.1$
add data	more labels	beats every row above

Order: get the model to overfit first, then add regularization one piece at a time, measuring each.

Attention, Transformers and Vision Transformers

Chapters 11 to 13: "the cat climbed the" → " roof", one step at a time, then the same block on image patches.

From text to the next word 11, 12

1 tokens	text → pieces → integer ids
2 embeddings	one vector per id, plus its position
3 attention	each token re-reads the others
4 mask, heads	no peeking at the future; several heads side by side
5 block	attention + MLP + residuals, $n \times d \to n \times d$
6 output	last vector → one logit per vocabulary token → softmax
7 generation	pick a token, append it, repeat

	toy of the classes	GPT-2 small
vocabulary V	12	50 257
width d; heads $h \times d_k$	4; 2×2	768; 12×64
blocks L; MLP width	1; 8	12; 3 072
weights	256	124 M

Tokens 11

- **BPE**: start from bytes, merge the most frequent pair, again and again; tokenize new text by replaying the merges in order. GPT-2: 50 000 merges + 256 bytes + 1 special token.
- The space belongs to the next word: `the` and `␣the` are different tokens.
- Portuguese costs more: 1.92 GPT-2 tokens per word against 1.06 in English (more latency, less context, a bigger bill).

Embeddings and position 11

$y_t = E[i_t] = \text{onehot}(i_t)^\top E,\ E \in \mathbb{R}^{V \times d}$ $\quad x_t = E[i_t] + P_t$ (GPT-2)

- GPT-2's table: 38.6 M numbers, 31% of the model. Its gradient is sparse: only the rows of the ids in the batch move, so rare tokens learn slowly.
- Attention without positions is permutation-equivariant: it cannot tell "dog bites man" from "man bites dog".

sinusoidal	fixed sin/cos added to the input; historical
learned	a vector per position (BERT, GPT-2); no longer contexts
ALiBi	a distance penalty on the scores
RoPE	rotates q and k by an angle $\propto$ position: $\langle R_m q, R_n k\rangle = \langle q, R_{n-m} k\rangle$, only the offset counts. Universal.

Scaled dot-product attention 11

$$\text{Attention}(Q,K,V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)V$$

$Q = XW_Q,\quad K = XW_K,\quad V = XW_V$: three projections of the same X

A **soft dictionary**: the query is compared with every key, and the output is the weighted average of the values. A dense layer whose weights are computed from the input.

X	$n \times d$	one vector per token
Q, K	$n \times d_k$	questions and labels
V	$n \times d_v$	contents
$S = QK^\top/\sqrt{d_k}$	$n \times n$	every pair: the $O(n^2)$
$A = \text{softmax}(S)$	$n \times n$	each row sums to 1
$O = AV$	$n \times d_v$	one vector with context per token

By hand: "bank" next to "river"

$q = [1,2]$, keys $[1,0], [1,1], [0,2]$: scores $1, 3, 4$; $\div\sqrt{2}$: $0.71, 2.12, 2.83$; softmax $[0.07, 0.31, 0.62]$; with values $[1,0], [0.5,0.5], [1,0]$ the output is $[0.85, 0.15]$: the *river* bank.

Why $\sqrt{d_k}$

$\text{Var}(q \cdot k) = d_k$ for unit-variance components. At $d_k = 128$ the scores differ by $\sim$20, the softmax becomes a hard argmax and its gradient vanishes. Dividing by d_k instead flattens it into a plain average.

▲ **Trap:** attention entropy collapsing early looks like a training bug. Log it.

Masks and heads 11

$M_{ij} = 0$ if $j \le i$, $-\infty$ if $j > i$, added to S before the softmax

$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)\, W^O$

- The causal mask gives n training examples per forward pass. Others: padding, sliding window, document boundaries when packing.
- h heads of width d/h cost what one head of width d costs: free parallelism. Parameters: W_Q, W_K, W_V, W^O, four $d \times d$ matrices, whatever h is.
- Shapes: $B \times n \times d \to B \times h \times n \times d/h$; scores $B \times h \times n \times n$.

The KV cache and the $O(n^2)$ 11

cache bytes $= 2 \times L \times h_{kv} \times d_{head} \times n \times B \times$ bytes per number

- Llama 3 8B ($L = 32$, $h_{kv} = 8$, $d_{head} = 128$, 16 bits): 128 KiB per token, 16 GiB at 128k tokens, as much as the weights.
- h_{kv} is the free term: MHA (h), MQA (1, worse), **GQA** (8, the default), MLA (a cached low-rank latent).
- Decoding reads the whole cache per token: bandwidth-bound, not FLOP-bound.
- FlashAttention: tiles, never stores the $n \times n$ matrix, exact. Sliding windows, linear attention and SSMs trade recall; hybrids win.

What came before

An RNN multiplies the gradient by a matrix at each step: after T steps, a product of T factors ($0.9^{64} \approx 10^{-3}$). The LSTM's gated cell slows the decay; attention reaches any token in one step, and in parallel.

The Transformer block 12

$h = x + \text{Attn}(\text{Norm}(x))$ $\quad y = h + \text{FFN}(\text{Norm}(h))$

pre-LN stack: $x_L = x_0 + \sum_\ell f_\ell(\cdot) \Rightarrow \partial x_L/\partial x_0 = I + \dots$

- Attention **talks** (the only exchange between positions); the MLP **thinks** (each token alone, $d \to 4d \to d$). Residuals add corrections to one **residual stream**.
- The MLP holds two-thirds to three-quarters of a block's parameters.

piece	2017	2026
norm placement	post-LN (needs warmup)	**pre-LN**
normalizer	LayerNorm	RMSNorm $x/\sqrt{\frac{1}{d}\sum x_i^2 + \epsilon} \odot g$
MLP	ReLU, $4d$, 2 matrices	SwiGLU $(\text{Swish}(xW_1) \odot xW_3)W_2$, $\frac{8}{3}d$
attention	MHA	GQA
position	sinusoidal, at the input	RoPE, on q and k
bias	everywhere	none

Output and decoding 12

logits $z = f\, W_U$ (weight tying: $W_U = E^\top$) $\quad p_j = \frac{e^{z_j/T}}{\sum_k e^{z_k/T}}$

loss: $\mathcal{L} = -\frac{1}{n}\sum_t \ln p(\text{true next token}_t)$, the cross-entropy

- Toy: mean loss 0.979 nats against $\ln 12 = 2.485$ for a uniform guess. Adding 10 to every logit changes nothing.
- Temperature never changes the argmax: logits $[2,1,0]$ give $[0.67, 0.24, 0.09]$ at $T = 1$, $[0.87, 0.12, 0.02]$ at $T = 0.5$. $T \to 0$ is greedy.
- Top-k keeps k tokens; top-p the smallest set reaching mass p; min-p the tokens above a fraction of the top one.

Families and scaling 12

- Encoder-only (BERT, bidirectional, masked tokens): embeddings, retrieval, rerankers. Encoder–decoder (T5): where the modalities differ. **Decoder-only won**: every token is a training signal, one objective covers every task, and the KV cache works.
- Training compute $C \approx 6ND$ (N parameters, D tokens). Chinchilla: about 20 tokens per parameter. Deployed models train far past it (Llama 3 8B: about 1 900), because serving costs scale with N.

Vision Transformer 13

$x \in \mathbb{R}^{H \times W \times C} \to N = HW/P^2$ patches of P^2C numbers $\xrightarrow{E} \mathbb{R}^{N \times d}$

cost per image: $12LNd^2 + 2LN^2d$ multiply-adds

- 224 px, $P = 16$: 196 tokens (+`[CLS]`). ViT-B/16: $d = 768$, 12 blocks, 12 heads, ≈86.6 M parameters. The patch embedding *is* `Conv2d(C, d, P, stride=P)`.
- Halving P quadruples N; attention dominates only when $N > 6d$ (4 608 tokens for ViT-B).
- The convolutional prior is a floor at small scale and a ceiling at large scale: ImageNet-1k only, ViT loses; 14 M images, level; 300 M, ViT wins. DeiT showed most of that gap was the recipe.
- Today: pooling instead of `[CLS]`, 2-D RoPE, register tokens. A ViT wins because it emits **tokens**: it is the image encoder of VLMs, CLIP/SigLIP, DINOv2 and the DiT backbone.

Adapting models and large language models

Chapters 14 and 15: whether to adapt a pretrained model at all, how to do it cheaply, and what turns a text-continuer into an assistant.

Before you fine-tune: don't 14

try first	fixes	cannot fix
a better prompt	format, tone, framing	missing knowledge
retrieval (RAG)	facts it lacks, facts that change, citations	behaviour, reasoning
tools, function calls	arithmetic, live data, actions	anything without an API
a bigger model	almost everything, now	your cost and latency
fine-tuning	consistent behaviour cheaply; a small model on *your* task	facts that change

Trap: fine-tuning to teach facts. They are smeared over the weights, cannot be cited or updated. Fine-tune for **behaviour**, retrieve for **knowledge**.

How much to train 14

regime	trains	DistilBERT
linear probe	the head only, on cached features	1 538
partial	the head and the last blocks	7.1 M/block
full	everything, small η	66.4 M

Full fine-tuning is expensive because of the **optimizer state**: AdamW keeps an fp32 copy and two moments, 12 bytes per trainable parameter (six times the bf16 weights); with gradients, about 16.

LoRA and QLoRA 14

$$W = W_0 + \frac{\alpha}{r} BA, \qquad B \in \mathbb{R}^{d\times r},\ A \in \mathbb{R}^{r\times k},\ r \ll \min(d,k)$$

trainable: $r(d+k)$ instead of dk $\quad(4096\times 4096,\ r=8$: $256\times$ fewer)

- The hypothesis: the change a task needs has low rank. W_0 frozen; A random, $B=0$, so training starts from the pretrained model ($\partial L/\partial A = 0$ on step one). Both at zero would never move.
- Apply it to **every** linear layer (the MLP included), $r=16$, $\alpha=32$, $\eta \approx$ 1e-4 ($10\times$ full fine-tuning). Merge for deployment: no extra inference cost.
- **QLoRA**: the frozen base in 4-bit NF4, the adapter trains around the quantization error. **DoRA**: magnitude and direction. Adapters, prefix tuning, BitFit: historical.

Trap: teaching a new language with LoRA: that update is not low-rank, and no affordable r makes it so.

Post-training 14, 15

SFT on demonstrations (format) → preference optimization (which answer is better) → RL from a verifier (being right).

$$\mathcal{L}_{\text{DPO}} = -\log\sigma\Big(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\Big)$$

GRPO: $\hat{A}_i = \dfrac{r_i - \text{mean}(r_{1..G})}{\text{std}(r_{1..G})}$ over G attempts at one prompt

- RLHF: a reward model trained on human comparisons, then PPO. DPO trains on the comparisons directly: two models instead of four, no sampling loop.
- GRPO drops PPO's value network; with a checker (unit tests, a verifiable answer) it produced the reasoning models (DeepSeek-R1).

Trap: catastrophic forgetting: keep η low, mix about 10% general data, and evaluate on tasks you are not training on.

When to use what 14

wrong format or tone	prompt
missing or changing facts	RAG
big-model quality at 1% of the cost	distil: SFT a small model on its outputs
1k–100k examples, one GPU	LoRA, all linear layers, $r=16$
memory-bound	QLoRA
many customers, one base	unmerged LoRA adapters
"prefer this answer"	SFT → DPO
answers a program can check	SFT → GRPO

Pretraining 15

$$\mathcal{L} = -\sum_{t=1}^{T}\log p_\theta(x_t \mid x_{<t}) \quad \text{(nats; 1 nat} \approx 1.44 \text{ bits)}$$

$$\text{perplexity} = e^{\mathcal{L}/T}$$

- Self-supervised: the labels are the next tokens, so any text is training data. Compressing text well requires modelling what produced it.
- A loss of 3 nats per token is a perplexity of $e^3 \approx 20$. Pythia: $L \propto N^{-0.10}$, a straight line on log–log axes.

symptom	tokenization cause
r's in "strawberry"	the model never sees letters
brittle arithmetic	`1234` as `12|34`, `12345` as `123|45`
Portuguese costs more	merges fitted on mostly English text
trailing space ruins it	`" Paris"` and `"Paris"` are different tokens

Trap: expecting answers from a base model: it continues text, and may reply with three more exam questions.

Test-time compute 15

pass@$k = 1-(1-p)^k$ for a problem solved with probability p

best-of-n with a verifier, as n grows: $\dfrac{p\,\text{TPR}}{p\,\text{TPR} + (1-p)\,\text{FPR}}$

- pass@k is an upper bound; majority voting plateaus; a verifier cashes in the gap, as far as its false positives allow ($p=0.25$, TPR 0.80, FPR 0.35: 0.43 at any n; FPR 0: 1).
- That is why reasoning got good first at code and maths: cheap verifiers with almost no false positives.

Trap: reward hacking: optimizing against a checker optimizes the checker, not correctness.

Mixture of experts 15

$$\text{MoE}(x) = \sum_{i\in\text{Top-}k(g(x))} g(x)_i\, E_i(x), \qquad g(x) = \text{softmax}(W_g x)$$

Memory for the **total** parameters, compute for the **active** ones (DeepSeek-V3: 671 B total, 37 B active). The hard part is load balancing, not routing; fine-grained experts plus a shared one.

Decoding and serving 15

- Defaults: facts, code, tool calls at $T = 0$–0.2; prose at 0.7–1.0 with top-p 0.9 or min-p 0.05.
- **Prefill** reads the prompt in parallel (compute-bound); **decode** makes one token per pass (memory-bound). The KV cache turns $O(n^2)$ token passes into $O(n)$.
- Quantization (8 or 4 bits) reads fewer bytes per token. Speculative decoding: a draft proposes, the big model accepts each token with probability $\min(1, p/q)$: faster, same distribution.

Where the failures come from 15

- "Emergence" is often a threshold metric; the loss improves smoothly.
- Hallucination is trained in: accuracy-only grading rewards a guess over "I don't know".
- Prompt injection has no robust fix once the model has tools.
- Agents are loops, and errors compound: a 95%-reliable step is 60% reliable after ten.

From chat to agents 15

A chat returns text; an **agent** runs a loop: read the goal, call a tool, read the result, decide, repeat, stop. Most of the work is the tools, how errors surface, how much history is kept and when to stop. The context window is a budget: every tool result competes with the goal. MCP standardizes the tool-server side.

The landscape, and choosing 15

frontier, closed	GPT, Claude, Gemini: reasoning modes, multimodal, the best tool use; rented
open weights	DeepSeek, Qwen, Kimi, GLM, Llama, Mistral: months behind, a fraction of the price
small, open	1B–30B, one GPU or a phone: targets for distillation
specialists	code, embeddings, rerankers, safety, OCR

Prices fall by an order of magnitude a year. Choose not from a leaderboard: write 20–50 examples of *your* task, run three or four models, read the failures.

LoRA in code 14

```python
from peft import LoraConfig, get_peft_model
cfg = LoraConfig(r=16, lora_alpha=32,
                 target_modules="all-linear")
model = get_peft_model(model, cfg)   # r(d+k) per matrix
merged = model.merge_and_unload()    # W0 + (alpha/r) BA
```

Generative models: the map, VAEs, GANs and CLIP

Chapters 16 to 19: turn a distribution you can sample from into one you cannot, and learn what a sentence means for an image.

Four ways to cut the problem 16

family	decomposition	sampling	likelihood
autoregressive	$p(x)=\prod_i p(x_i \mid x_{<i})$	one pass per dimension	exact
latent (VAE)	$p(x)=\int p(x\|z)p(z)\,dz$	one pass	a bound
adversarial (GAN)	none: a generator and a critic	one pass	none
diffusion, flow	a path in time from noise to data	10s to 100s of passes	via the ODE

forward $\mathrm{KL}(p_{\text{data}}\|p_\theta)$: covers every mode, blurs (maximum likelihood, VAE)
reverse $\mathrm{KL}(p_\theta\|p_{\text{data}})$: picks a mode, drops the rest (GAN-like)

Blur and mode collapse are the two ends of one asymmetry. The trilemma: quality, coverage, speed; speed is the one you can buy back later (distillation).

Autoencoder to VAE 17

- An autoencoder $o=D(E(x))$, $\mathcal{L}=\|x-o\|^2$, compresses; it cannot generate: its codes are a set of points with holes, not a distribution.
- A VAE encodes a **distribution** per input: $q_\phi(z|x)=\mathcal{N}(\mu_\phi(x),\mathrm{diag}\,\sigma^2_\phi(x))$, pulled towards the prior $p(z)=\mathcal{N}(0,I)$.

$$\log p_\theta(x)=\text{ELBO}+\mathrm{KL}\big(q_\phi(z|x)\,\|\,p(z|x)\big)\ \geq\ \text{ELBO}$$
$$\text{ELBO}=\mathbb{E}_q\big[\log p_\theta(x|z)\big]-\mathrm{KL}\big(q_\phi(z|x)\,\|\,p(z)\big)$$
$$\mathcal{L}=\|x-\hat{x}\|^2+\beta\cdot\tfrac{1}{2}\textstyle\sum_i\left(\mu_i^2+\sigma_i^2-1-\log\sigma_i^2\right)$$
reparameterization: $z=\mu+\sigma\odot\epsilon,\quad \epsilon\sim\mathcal{N}(0,I)$
$\partial z/\partial\mu=1,\qquad \partial z/\partial\log\sigma^2=\tfrac{1}{2}\sigma\epsilon$: the noise enters as data

- The gap is exactly how wrong the encoder's guess of the posterior is; one objective trains both networks.
- By hand: $p(z)=\mathcal{N}(0,1)$, $p(x|z)=\mathcal{N}(2z,1)$, $x=3$: $\log p(x)=-2.6237$, posterior $\mathcal{N}(1.2,0.2)$. The best ELBO is the posterior's, not the best reconstruction's.
- The encoder outputs $s=\log\sigma^2$: $\sigma=e^{s/2}$.
- $\mathrm{KL}(\mathcal{N}(\mu_q,\sigma_q^2)\|\mathcal{N}(\mu_p,\sigma_p^2))=\log\frac{\sigma_p}{\sigma_q}+\frac{\sigma_q^2+(\mu_q-\mu_p)^2}{2\sigma_p^2}-\frac{1}{2}$; never negative, not symmetric.

β is the whole story

$\beta\to 0$: an autoencoder, samples from the prior decode to nothing. β too large: **posterior collapse**, KL ≈ 0 and the code ignores the input. Remedies: KL annealing, free bits.

Trap: watching only the total loss. Log reconstruction and KL separately: KL near 0 with reconstruction near the data variance is collapse.

What a VAE is for

Blur is the objective (forward KL, and ℓ_2 returns the conditional mean). Production autoencoders add a perceptual (LPIPS) and an adversarial term, $\beta\approx 10^{-6}$, and stay frozen under a diffusion model: SD 1.5 latent $64\times64\times4$ ($48\times$ fewer numbers), SD3/FLUX $64\times64\times16$ ($12\times$). **VQ-VAE**: $z_q=\text{codebook}[\arg\min_k\|z_e-e_k\|]$, integers for autoregressive models.

GANs 18

$$\min_G\max_D\ \mathbb{E}_x\big[\log D(x)\big]+\mathbb{E}_z\big[\log(1-D(G(z)))\big]$$
$$D^*(x)=\frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}\qquad V(D^*,G)=-\log 4+2\,\mathrm{JS}(p_{\text{data}}\|p_G)$$

- A loss you cannot write, learned: the discriminator is a density-ratio estimator.
- The generator uses the **non-saturating** loss $-\log D(G(z))$: at $D(G(z))=0.01$ its gradient is -0.99, the minimax one -0.01.
- $z\in\mathbb{R}^k$; k is the latent dimension, a size you choose.

mode collapse	a few outputs whatever z: nothing rewards diversity
vanishing gradient	D wins too completely
oscillation	the players cycle; a game need not converge
D overfitting	it memorizes the training set

What stuck: WGAN-GP, spectral normalization, R1, ADA, all doing one thing: **stop the discriminator from becoming too sharp**.

Trap: reading GAN losses: they sit at $2\log 2\approx 1.39$ and $\log 2\approx 0.69$ whether it works or not. Look at samples and at a metric.

The architecture lost (diffusion has a regression loss that goes down); the **adversarial loss** lives on in latent-autoencoder decoders, super-resolution, vocoders and 1–4-step distillation: wherever the target is one of many valid answers.

CLIP 19

Two encoders (a ViT and a Transformer over $\leq$77 tokens), a linear projection and L2 normalization each, into one d-dimensional space.

$$\mathcal{L}=\tfrac{1}{2}\Big(\mathrm{CE}\big(\tfrac{IT^\top}{\tau},\mathrm{diag}\big)+\mathrm{CE}\big(\tfrac{TI^\top}{\tau},\mathrm{diag}\big)\Big)\qquad\text{(symmetric InfoNCE)}$$
SigLIP: $\frac{1}{N}\sum_{i,j}\log\left(1+e^{-y_{ij}(t\langle I_i,T_j\rangle+b)}\right)$, $y_{ij}=\pm1$, b starts at -10

- Row i of $IT^\top/\tau$ is an N-class problem: which caption is image i's? Every other caption in the batch is a negative (32 767 of them at $N=32\,768$); a blind guess pays $\log N$.
- τ is learned: starts at 0.07, clipped at $1/\tau\leq 100$, ends at 0.01. Right and wrong captions differ by a few hundredths of cosine (0.26 against 0.20): only $\times100$ makes the softmax confident. τ changes the loss, not the ranking.
- Zero-shot: logits $=Wx/\tau$ with rows w_c = encoded prompts, a linear classifier written in sentences; "zero-shot" is retrieval from 400 M training pairs.
- Unit vectors: $\|a-b\|^2=2-2\cos(a,b)$. Random cosines in d dimensions have sd $\approx 1/\sqrt{d}$ (0.044 at 512).

Trap: CLIP is close to a bag of words: counting, spatial relations and composition are weak; and there is a modality gap (images and texts in separate cones).

Diffusion 20

$$q(x_t|x_{t-1})=\mathcal{N}\big(\sqrt{1-\beta_t}\,x_{t-1},\,\beta_t I\big),\ T=1000$$
$$x_t=\sqrt{\bar\alpha_t}\,x_0+\sqrt{1-\bar\alpha_t}\,\epsilon,\qquad \bar\alpha_t=\textstyle\prod_{s\leq t}(1-\beta_s)$$
$$\mathcal{L}=\mathbb{E}_{x_0,\epsilon,t}\big\|\epsilon-\epsilon_\theta(x_t,t,c)\big\|^2$$
$$\hat{x}_0=\big(x_t-\sqrt{1-\bar\alpha_t}\,\epsilon_\theta\big)/\sqrt{\bar\alpha_t}\qquad \text{score}\approx-\epsilon_\theta/\sqrt{1-\bar\alpha_t}$$

- Any noise level in closed form: draw t, jump there, regress the noise you drew yourself. Ordinary supervised regression, stable, covers the modes.
- ϵ-, x_0- and v-prediction ($v=\sqrt{\bar\alpha_t}\epsilon-\sqrt{1-\bar\alpha_t}x_0$) are interchangeable; they weight the noise levels differently.
- The schedule decides what is learned: SD 1.x's $\bar\alpha_T\approx 0.0047\neq 0$ is why it cannot make a truly black image.

Stable Diffusion 20

Diffusion + a VAE latent ($48\times$ smaller) + a CLIP text encoder + classifier-free guidance.

stage	shape	params
tokenizer, CLIP text encoder	77×768	123 M
initial noise z_T (the seed)	$4\times64\times64$	
U-Net, twice per step	$2\times4\times64\times64$	860 M
VAE decoder, once	$3\times512\times512$	50 M

The prompt enters by **cross-attention**: queries from the image latent, keys and values from the text.

CFG: $\tilde\epsilon=\epsilon_\theta(x_t,\varnothing)+w\,\big(\epsilon_\theta(x_t,c)-\epsilon_\theta(x_t,\varnothing)\big)$

- Train with the caption dropped 10% of the time; $w=0$ unconditional, $w=1$ conditional, $w>1$ extrapolates. It doubles the cost and trades diversity for adherence; too much leaves the data manifold.
- Samplers are separate from the model: DDPM 1000 steps; DDIM deterministic and invertible; DPM-Solver++ 15–25. Below ~10 steps you need distillation (LCM, SDXL-Turbo with a discriminator: 1–4).
- ControlNet, IP-Adapter and LoRA all freeze the base and add a small path beside it, initialized to do nothing.
- Since SD 1.5: a Transformer backbone (MMDiT), T5 + CLIP text, 16-channel latents, rectified flow.

Trap: the seed fixes only the initial noise; with a stochastic sampler the same seed still changes the image.

Generative models: flow matching, diffusion Transformers and tokens

Chapters 21 to 23: straighter paths, a scalable backbone, and images as sequences.

Flow matching 21

$$\frac{dx}{dt} = v_\theta(x,t), \quad x_0 \sim \mathcal{N}(0,I) \;\Rightarrow\; x_1 \sim p_{\text{data}}$$
$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,x_0,x_1}\left\|v_\theta(x_t,t) - (x_1 - x_0)\right\|^2, \qquad x_t = (1-t)x_0 + t x_1$$

- The marginal field $u_t(x) = \mathbb{E}[x_1 - x_0 \mid x_t = x]$ is intractable; regressing the **conditional** target $x_1 - x_0$ has the same gradient. The same move as DDPM regressing ϵ instead of the score.
- So the loss has a floor above zero, and the learned paths cannot cross.
- Here $t = 0$ is noise, $t = 1$ data: the opposite of chapter 20.

method	$a(t)$	$b(t)$	path
DDPM	$\sqrt{\bar\alpha_t}$	$\sqrt{1-\bar\alpha_t}$	curved, quarter circle
flow matching	t	$1-t$	straight per pair
rectified flow	t	$1-t$	straight, re-coupled

With $x_t = a(t)x_1 + b(t)x_0$: diffusion is flow matching with a curved interpolant. The conditional path is straight; the **marginal** one, which the model learns, is not.

- The real advantage is the objective: one regression, no schedule, no ϵ-vs-v choice.
- **Reflow**: integrate the trained ODE, retrain on its own (z, x) pairs; straighter paths, one Euler step gets closer.
- One step: learn jumps, not velocities (consistency, shortcut, MeanFlow) or distil (FLUX.1 schnell).
- SD3 samples t from a logit-normal (the middle); shift t with the resolution.
- Used for image, video, speech and robot actions (π_0: 10 Euler steps per chunk of 50 actions).

Trap: three conventions differ between codebases: is $t = 0$ noise or data, does the model predict velocity, x_1 or ϵ, and is t in $[0,1]$ or an integer index.

Diffusion Transformers 22

patchify a $C \times H \times W$ latent: $N = \frac{H}{p} \cdot \frac{W}{p}$ tokens of p^2C numbers $\to d$

$$c = \text{MLP}(\text{emb}(t)) + \text{emb}(y); \quad (\beta_1,\gamma_1,\alpha_1,\beta_2,\gamma_2,\alpha_2) = W\,\text{SiLU}(c) + b$$
$$h = x + \alpha_1 \odot \text{Attn}\big((1+\gamma_1)\odot \text{LN}(x) + \beta_1\big)$$
$$x' = h + \alpha_2 \odot \text{MLP}\big((1+\gamma_2)\odot \text{LN}(h) + \beta_2\big)$$

- One resolution throughout, every block sees every token. DiT-XL/2 at 256 px: latent $4 \times 32 \times 32$, $p = 2$, 256 tokens.
- **adaLN-Zero**: $W = 0$, $b = 0$ at the start, so every gate $\alpha = 0$ and each block is the identity (like LoRA's $B = 0$). It beat in-context and cross-attention conditioning at the lowest cost; $6d^2$ parameters per block.
- Patch size is the compute dial: halving p quadruples N at nearly the same parameters, and FID improves.
- The text gets in by cross-attention (read-only), joint attention with two streams (MMDiT: SD3) or one stream (FLUX's later blocks).
- RoPE per axis (row, column, time): resolution is just a sequence length.
- Cost per block $12Nd^2 + 2N^2d$; attention's share $N/(N+6d)$: 4% for DiT-XL/2, 20% for FLUX at 1024^2, ≈87% for a 720p video. Images are MLP-bound, video attention-bound.
- The contribution is a **scaling curve**: FID falls steadily with Gflops.

Images as tokens 23

$$\mathcal{L}_{VQ} = \|x - D(z_q)\|^2 + \|\text{sg}[z_e] - e\|^2 + \beta\|z_e - \text{sg}[e]\|^2$$
$$z_q = z_e + \text{sg}[e - z_e] \quad \text{(sg: stop-gradient)}$$
$$p(t_1,\ldots,t_N) = \textstyle\prod_i p(t_i \mid t_{<i}, c)\text{; CFG on logits: } \ell = \ell_\varnothing + w(\ell_c - \ell_\varnothing)$$

- VQ-GAN: 256×256, downsampling 16 $\to$ 256 tokens; with $K = 16\,384$ codes, $\le 256 \times 14 = 3\,584$ bits.
- The last term of z_q is the **straight-through estimator**. Only the selected code gets a gradient, so codes die (codebook collapse). FSQ rounds a few bounded dimensions (levels $8 \cdot 6 \cdot 5 = 240$ codes, no loss); LFQ uses signs (2^{18}).
- Cost: one sequential pass per token (256 for a 16×16 grid). MaskGIT: all masked tokens at once, keep the confident ones, cosine schedule, 8–12 rounds. VAR: the next *scale*, 10 passes. MAR: continuous tokens with a small diffusion head.
- The real axes: generation order (raster, parallel rounds, coarse to fine, all at once) and per-token distribution (categorical or continuous).
- Exact likelihood, but of the tokens. It came back for **unification**: one sequence, one loss for text and images (Chameleon, Emu3, Janus-Pro; Transfusion and BAGEL with a diffusion loss inside).

Which generator 16–23

text-to-image, quality first	diffusion / flow, DiT
interleaved text and images	autoregressive, or hybrid
fast sampling, discrete tokens	MaskGIT, VAR
exact likelihood, compression	autoregressive
speech and audio	autoregressive over codec tokens
one step at inference	GAN, or a distilled diffusion model
a compressor under a diffusion model	VAE (frozen)

A history of bottlenecks 16

when	what	the bottleneck it removed
2013	VAE	backpropagating through a random node
2014	GAN	needing a tractable likelihood
2015–20	diffusion, DDPM	quality without an adversary
2021	CLIP, latent diffusion	conditioning on language; a 48× smaller space
2022–23	SD, DALL·E 2	access: open weights
2023–24	flow matching, DiT	straighter paths, a scalable backbone, 1000 $\to$ <10 steps
2024–26	video, any-to-any	coherence in time; one model for every modality

Trap: evaluation and provenance are unsolved: FID is a weak proxy, and the training images are other people's work.

Flow matching in code 21, 22

```
# training: t = 0 is noise, t = 1 is data
x1 = batch                              # data
x0 = torch.randn_like(x1)               # noise
t = torch.rand(len(x1), 1, 1, 1)
xt = (1 - t) * x0 + t * x1
y = torch.where(torch.rand(len(y)) < 0.1, null_id, y)  # CFG
loss = ((model(xt, t, y) - (x1 - x0)) ** 2).mean()

# sampling: Euler steps along the learned field
x = torch.randn(n, C, H, W)
for i in range(steps):
    t = torch.full((n, 1, 1, 1), i / steps)
    x = x + model(x, t, y) / steps
```

Steering a diffusion model 20

img2img	encode an image, noise it to some t (the strength), denoise from there
inpainting	at each step, put back the correctly noised original outside the mask
ControlNet	a trainable copy of the encoder, joined by zero convolutions: pose, depth, edges
IP-Adapter	extra cross-attention on CLIP image embeddings: style or identity
LoRA	a style, a character, a concept, on the attention layers

sampler or distillation	steps
DDPM (stochastic)	1000
DDIM (deterministic, invertible)	50
Euler, Heun	20–30
DPM-Solver++ (the practical default)	15–25
UniPC, DEIS	10–20
progressive distillation	8192 $\to$ 4
LCM / LCM-LoRA (consistency)	2–8
SDXL-Turbo, ADD (with a discriminator)	1–4

Chapter by chapter

The question each chapter answers, what to take from it and the trap that catches most people.

1 · Concepts — learning

Where do neural networks sit among the ways of building AI?
Symbolic, connectionist and neuro-symbolic AI; AI ⊃ ML ⊃ neural networks ⊃ deep learning; supervised, unsupervised, self-supervised.
⚠ **Trap:** calling every learned model "deep learning".

2 · Data — data

What can break before any weight exists?
The split is the experiment. Fit nothing before it; split by the unit you generalize to; encode by the comparisons that are real; scale inside the `Pipeline`.
⚠ **Trap:** a filter, a scaler or oversampling run before the split.

3 · Neural networks — learning

How did we get from a neuron model to deep learning?
1943 McCulloch–Pitts, 1958 perceptron, 1969 the XOR winter, 1986 backpropagation, 1989 universal approximation, 2012 AlexNet, 2017 Transformer.
⚠ **Trap:** "one hidden layer can approximate anything" does not say it is easy to learn.

4 · Perceptron — learning

What can one neuron learn by correcting its mistakes?
$w \leftarrow w + \eta(y - \hat{y})x$ converges iff the classes are linearly separable.
⚠ **Trap:** expecting it to solve XOR.

5 · Multi-layer perceptron — learning

How does the error at the output reach the first weight?
Forward, loss, backward by the chain rule ($\delta_h = (V^\top \delta_y) \odot \sigma'$), update. A non-linearity is what makes two layers more than one.
⚠ **Trap:** a sigmoid in every layer: each one shrinks the gradient by 4.

6 · Optimization — learning

How big a step, in which direction, with how much data?
$\eta < 2/L$; noise $\propto 1/\sqrt{B}$; momentum averages, adaptive methods rescale; AdamW + warmup + cosine; tune η first.
⚠ **Trap:** PyTorch momentum is $10\times$ the textbook step at $\beta = 0.9$.

7 · Regularization — learning

How do you make a network stop memorizing?
Fit worse on purpose: weight decay, dropout ($1/(1-p)$ in training), normalization, early stopping, augmentation; more data beats them all.
⚠ **Trap:** forgetting `model.eval()`.

8 · Metrics and evaluation — data

Which number decides, and what does a mistake cost?
Precision over claims, recall over truth; PR-AUC under imbalance; threshold from costs; RMSE with a baseline; FID + CLIP score; pass@k with its k.
⚠ **Trap:** accuracy on rare positives; perplexity across tokenizers.

9 · Deep networks — architectures

What breaks when you stack many layers, and what fixed it?
Signal and gradient are products of L factors $n\sigma^2\rho$: He or Xavier init, residuals $(1 + \partial f/\partial a)$, normalization.
⚠ **Trap:** a skip connection without normalization: everything explodes.

10 · Convolutional networks — architectures

What do you gain by assuming locality and weight sharing?
$C_{out}(C_{in}k^2 + 1)$ weights, independent of the image; receptive field is the budget; equivariance from sharing, invariance from pooling and augmentation.
⚠ **Trap:** shape errors: compute $\lfloor (H + 2p - k)/s \rfloor + 1$.

11 · Attention — language

How does each token re-read all the others?
softmax$(QK^\top/\sqrt{d_k})V$: a dense layer whose weights come from the input. Tokens, embeddings, RoPE, masks, heads, KV cache.
⚠ **Trap:** dropping $\sqrt{d_k}$: the softmax saturates and the gradient dies.

12 · Transformers — language

From the last vector to the next word, and what changed since 2017?
Embedding → L blocks (attention talks, MLP thinks, residual stream) → logits → sampling. Pre-LN, RMSNorm, SwiGLU, GQA, RoPE; $C \approx 6ND$.
⚠ **Trap:** thinking temperature changes the most likely token.

13 · Vision Transformers — architectures

What if an image is just a sequence of patches?
$N = HW/P^2$ tokens; a weak prior bought with data or a recipe; ViTs won because they emit tokens.
⚠ **Trap:** "ViTs need 300 M images": most of that gap was the training recipe.

14 · Transfer learning — language

Should you adapt a pretrained model at all, and how cheaply?
Prompt, RAG and tools first; LoRA $W_0 + \frac{\alpha}{r}BA$ on every linear layer; QLoRA when memory-bound; SFT → DPO → GRPO.
⚠ **Trap:** fine-tuning to teach facts.

15 · Large language models — language

What turns a next-token predictor into an assistant, and what does a token cost?
Pretraining compresses text; post-training makes the assistant; test-time compute and verifiers; MoE; decoding is memory-bound.
⚠ **Trap:** trusting pass@k without a verifier, or a single safety rate.

16 · Generative models — generative

How do you turn noise you can sample into data you cannot?
Four decompositions (AR, latent, adversarial, time); forward KL blurs, reverse KL collapses.
⚠ **Trap:** treating blur and mode collapse as unrelated bugs.

17 · VAEs — generative

How do you encode a distribution and sample from it?
ELBO = reconstruction − KL to the prior; $z = \mu + \sigma\epsilon$; β decides; today, the frozen compressor under diffusion.
⚠ **Trap:** posterior collapse hidden in a falling total loss.

18 · GANs — generative

What if the loss itself is learned?
$D^* = p_{\text{data}}/(p_{\text{data}} + p_G)$; non-saturating loss; keep D from becoming too sharp; the adversarial loss lives on.
⚠ **Trap:** reading the losses as progress.

19 · CLIP — generative

How do images and sentences end up in one space?
Symmetric InfoNCE over the batch, learned τ, unit vectors; zero-shot is a linear classifier made of sentences; SigLIP replaced it.
⚠ **Trap:** trusting it on counting and composition.

20 · Stable Diffusion — generative

How does noise become the image a sentence asked for?
Regress the noise; run it in a VAE latent; condition by cross-attention; CFG with w; samplers and distillation set the steps.
⚠ **Trap:** w too high: off-manifold, oversaturated images.

21 · Flow matching — generative

Can generation be one regression onto a straight line?
Regress $x_1 - x_0$ along $x_t = (1-t)x_0 + tx_1$; diffusion is the curved case; few steps need reflow or distillation.
⚠ **Trap:** mixing up the $t = 0$ convention between codebases.

22 · Diffusion Transformers — generative

What do you gain by replacing the U-Net with a Transformer?
Patchify the latent, adaLN-Zero conditioning, joint attention for text, RoPE per axis, and a scaling curve.
⚠ **Trap:** forgetting that video is attention-bound: share $N/(N + 6d)$.

23 · Autoregressive generation — generative

What if an image is generated like text, token by token?
A tokenizer (VQ, FSQ, LFQ) plus next-token prediction; MaskGIT, VAR and MAR change the order or the token; the route to unified models.
⚠ **Trap:** a dead codebook: most codes never used.