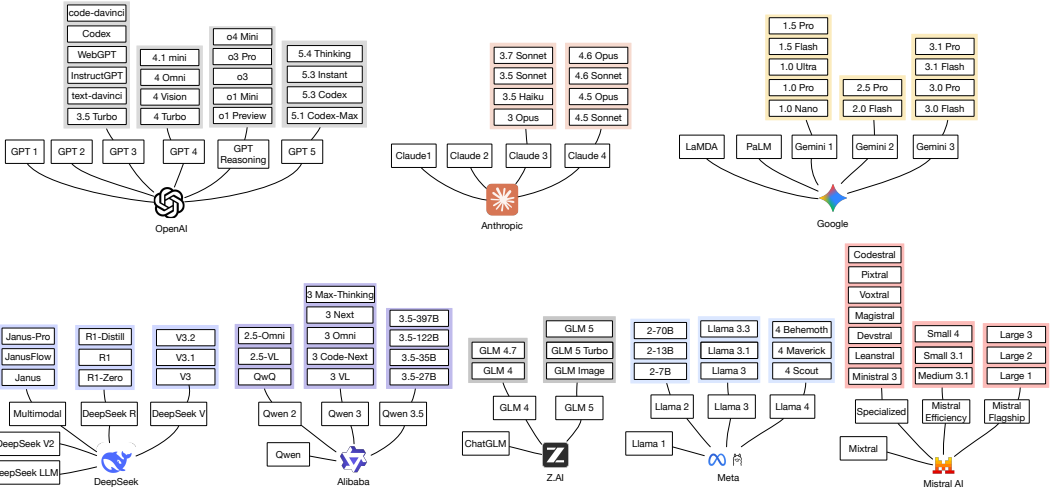
AI时代下智能体演化与应用研讨会

LLM Basics

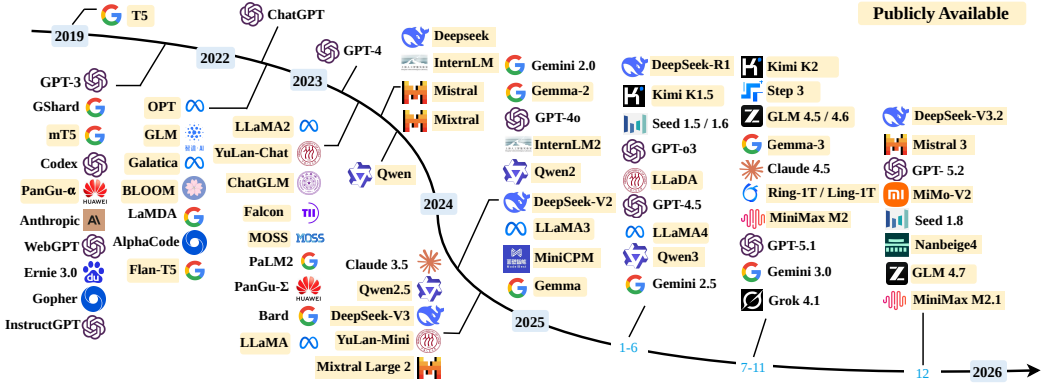
Bei Yu (余备)
CSE Department, CUHK
byu@cse.cuhk.edu.hk

Aug. 29, 2026

Popular LLM Families



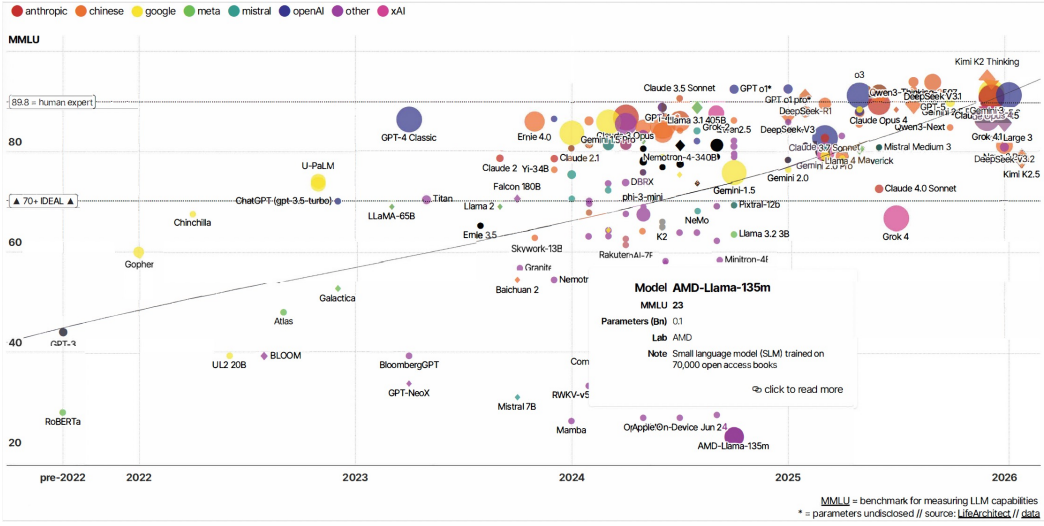
LLM Progress[1]



[1] Wayne Xin Zhao et al. (2023). "A survey of large language models". In: *arXiv preprint arXiv:2303.18223*.

LLM Metrics[2]

Major Large Language Models (LLMs)
ranked by capabilities, sized by billion parameters used for training



[2] https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt

Tokenization（分词）

What Is a Large Language Model?

An **LLM** is an *autoregressive* language model trained on massive text; its core task is simply to **predict the next token**.

- Autoregressive factorization: $p(x_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_{<t})$ — generate one token at a time.
- Every task (chat, code, reasoning) is cast as next-token prediction.
- **"Large"** = params × data × compute: 10^9–10^{12} params, $10^{12}+$ tokens[3].

[3] **Jared Kaplan, Sam McCandlish, Tom Henighan, et al. (2020). "Scaling Laws for Neural Language Models".** In: *arXiv:2001.08361*.

Why Tokenize at All?

- Models consume **discrete token-id sequences**, never raw characters.
- Characters $\Rightarrow$ sequences too long; whole words $\Rightarrow$ vocabulary explodes and out-of-vocabulary words break.
- **Subword** is the compromise: a fixed **vocab size** $\leftrightarrow$ **sequence length** trade-off.

Granularity	**Segmentation of "unbelievable"**	**# tok**	**Issue**
Character	`u n b e l i e v a b l e`	12	sequences too long
Word	`unbelievable`	1	huge vocab / OOV
Subword (BPE)	`un | believ | able`	3	balanced ✓

Byte-Pair Encoding (BPE)

Two phases

Train: start from characters; repeatedly **merge the most frequent adjacent pair** into a new token. Each merge is saved as a *rule*.

Inference: greedily apply the learned rules in order.

Corpus (word × count): `low`×5 `lower`×2 `owl`×3 `lowest`×2

Frequency-driven merges are the original BPE recipe[4].

Round	Best pair	Count	New token
1	`o + w`	12	`ow`
2	`l + ow`	9	`low`
3	`low + e`	4	`lowe`



applying rules to a new word “lower”

[4] **Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). “Neural Machine Translation of Rare Words with Subword Units”.** In: *ACL*.

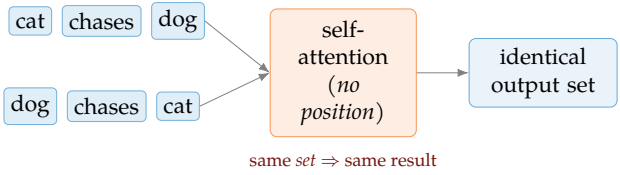
Where Tokenization Bites

- **Numbers:** digits split inconsistently ⇒ weak arithmetic.
- **Multilingual:** Chinese spends *more tokens* per meaning ⇒ costlier, unfair.
- **Code:** indentation / whitespace gets fragmented ⇒ brittle.
- **Counting:** "strawberry" = `[straw, berry]` ⇒ the model *cannot* count the r's.

```
>>> tok("3.14 + 42")
    ['3','.','14',' +',' 4','2']
>>> tok("strawberry")
    ['straw','berry']
>>> tok(" 你好世界")
    [' 你',' 好',' 世',' 界']  # 4 tok / 4
chars
```

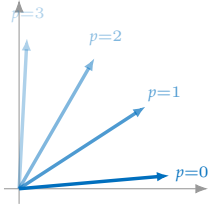
These are properties of the *tokenizer*, not bugs in the model.

Attention Is Order-Blind



- Self-attention is **permutation-invariant**: it sees a *set*, not a sequence.
- So "cat chases dog" = "dog chases cat" without extra signal.
- **Order carries meaning** — position must be **injected explicitly**.

Modern Position Encoding: RoPE & ALiBi



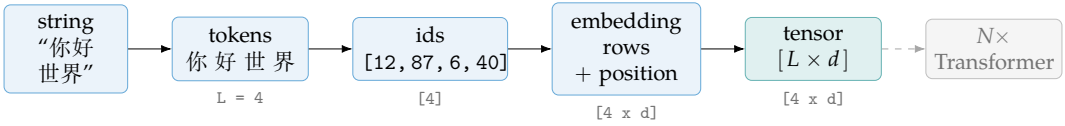
RoPE rotates each 2-D (Q, K) sub-vector by an angle $\propto$ position.

- **RoPE:** rotate Q, K by position $\Rightarrow$ the dot product depends only on the **relative** offset. Today's default.
- **ALiBi:** add a linear *distance penalty* $-m\,|i-j|$ to attention scores.

Errata: bare RoPE does *not* "naturally extrapolate". Because it is *relative*, **interpolation**-style extension (PI / YaRN) becomes feasible[a].

[a] **Jianlin Su et al. (2021). "RoFormer: Enhanced Transformer with Rotary Position Embedding".** In: *arXiv:2104.09864*; **Ofir Press, Noah A. Smith, and Mike Lewis (2022). "Train Short, Test Long: Attention with Linear Biases (ALiBi)".** In: *ICLR*.

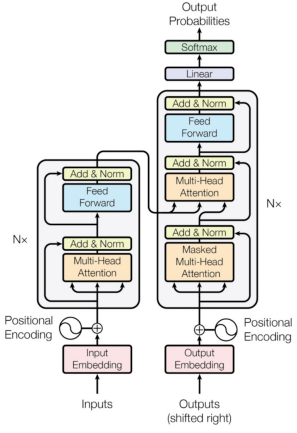
Putting It Together: String → Tensor



Discrete text becomes a dense $[L \times d]$ tensor (L = sequence length): **tokens** → **ids** → **embedding rows + position**, ready for the Transformer.

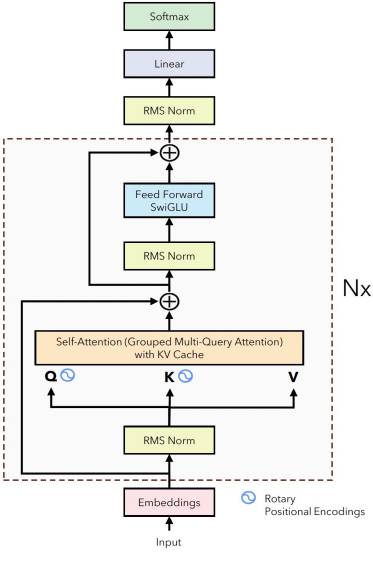
Transformer Layer

Transformer in LLM



Transformer

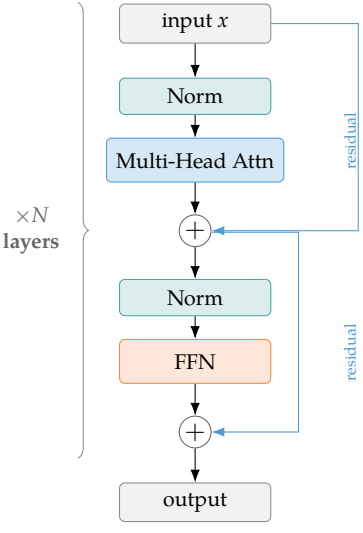
Encoder-Decoder Structure



LLM

Decoder-only Structure

Anatomy of One Transformer Layer

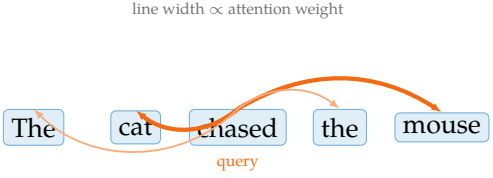


- Two sublayers: **Attention** (mixes across positions) then **FFN** (transforms each position).
- Each sublayer follows one template: **Norm** $\rightarrow$ **transform** $\rightarrow$ **add residual**.
- The whole model is this block **stacked** N **times** (7B: $N = 32$; larger models 80+).

```
def block(x):          # Pre-LN
   x = x + attn(norm(x))
   x = x + ffn(norm(x))
   return x
```

The residual carries x forward unchanged; each sublayer only *adds* a correction.

Self-Attention: Query, Key, Value

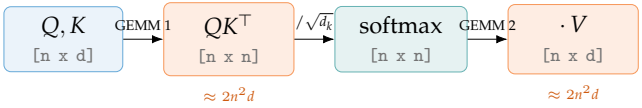


The verb "chased" attends most to its subject and object; every token pools from the *whole* sequence.

- Each token emits three vectors: **Q**uery, **K**ey, **V**alue.
- Retrieval analogy: **Q** = what I look for, **K** = my label, **V** = my content.
- Attention = weighted pooling of all **V**, weighted by how well each **K** matches the **Q**.

Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \underbrace{\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)}_{\text{weights } [n \times n]} V$$

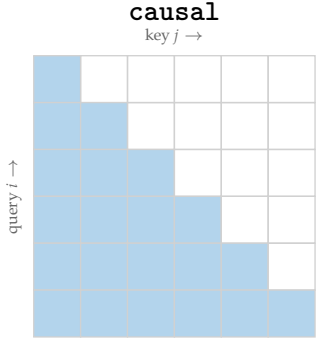


Two big matmuls dominate: score $QK^\top$ and value-mix $\cdot V$, each $\approx 2n^2 d$ FLOPs.

1. **Score:** $QK^\top$ — dot every query with every key.
2. **Scale:** $\div \sqrt{d_k}$.
3. **Normalize:** row-wise softmax.
4. **Aggregate:** weighted sum of V.

$\sqrt{d_k}$ keeps dot products from growing with d_k and flattening softmax gradients.

Causal Mask: Only Look Backward

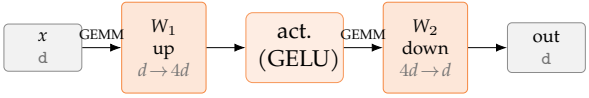


Shaded = allowed. Query i may attend to key j only if $j \leq i$.

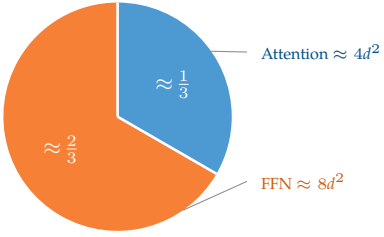
- Autoregressive generation may only use the **past**: future positions are set to $-\infty$ *before* softmax (they get weight 0).
- **Training** stays fully parallel: one forward pass scores every position's next-token prediction at once.
- **Inference** must still run **token by token** — each new token conditions on all previous ones.

This train-parallel / infer-sequential asymmetry is the root of KV-cache and decoding cost

Feed-Forward Network (FFN) Sublayer



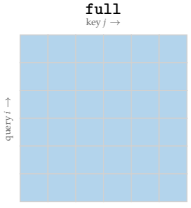
- Two matmuls: **expand** to $4d$, non-linear activation, **project** back to d.
- Applied **independently at every position** — exactly a 1×1 convolution over the sequence.



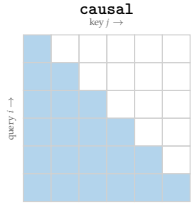
per-layer params (classic FFN $= 4d$)

With classic FFN $= 4d$ (two matrices): Attention $\approx 4d^2$, FFN $\approx 8d^2$ — the FFN is the **biggest parameter block** ($\approx 2/3$) per layer.

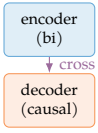
Three Architecture Paradigms



Encoder-only
bidirectional
BERT — understanding



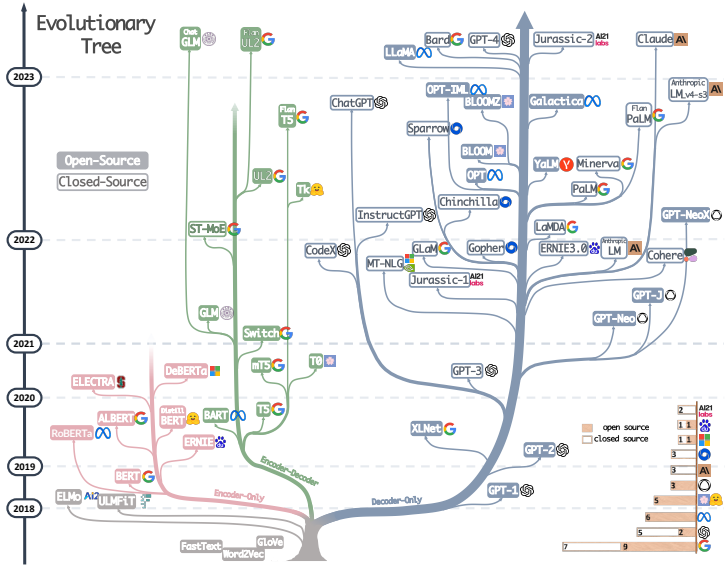
Decoder-only
causal
GPT/LLaMA — generation



Encoder-Decoder
T5 — translation

- **Decoder-only wins today:** one uniform next-token objective, clean scaling, and **KV-cache-friendly** generation.
- Encoders keep bidirectional context but cannot generate; encoder-decoders shine on fixed input→output tasks (translation, summarization).

Evolutionary Tree[5]



[5]Jingfeng Yang et al. (2024). "Harnessing the power of LLM in practice: A survey on ChatGPT and beyond". In: 18.6, pp. 1–32.

Attension

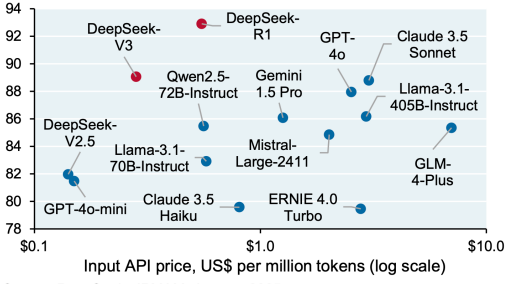
Cost vs. Performance

EYE ON THE MARKET • MICHAEL CEMBALEST • J.P. MORGAN • January 28, 2025

2025 Eye on the Market Outlook

Cost vs performance of select AI models

MMLU Redux ZeroEval Score (multi-subject performance)

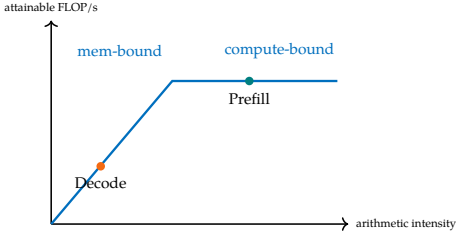


Source: DeepSeek, JPMAM, January 2025

How does a small Chinese company like DeepSeek with less than 200 employees offer strong language model performance at low cost? The short list:

- Mixture of experts
- Multi-head latent attention
- Lower precision parameter storage during model training
- Multi-token prediction
- Reinforcement "chain of thought" learning without human feedback

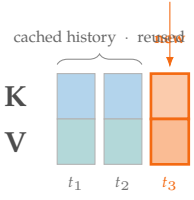
Two Phases of Inference: Prefill & Decode



- **Prefill:** the whole prompt in one parallel pass — large matrix–matrix GEMMs, high arithmetic intensity ⇒ **compute-bound**.
- **Decode:** one token per step, reusing cached history — a thin matrix–vector product, low intensity ⇒ **memory-bound**.

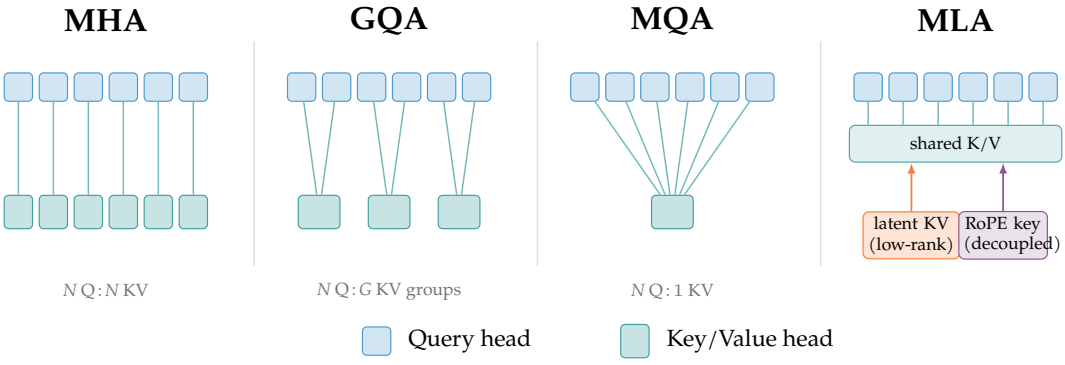
KV Cache — the First Core Concept

- A history token's **K/V never changes** when a new token arrives ⇒ compute them once, cache, and reuse.
- **Per step** the cost drops from $O(n^2)$ (recompute all pairs) to $O(n)$ (new token attends to history) — but **total generation is still** $O(n^2)$.
- Price: the cache grows **linearly** with sequence length.



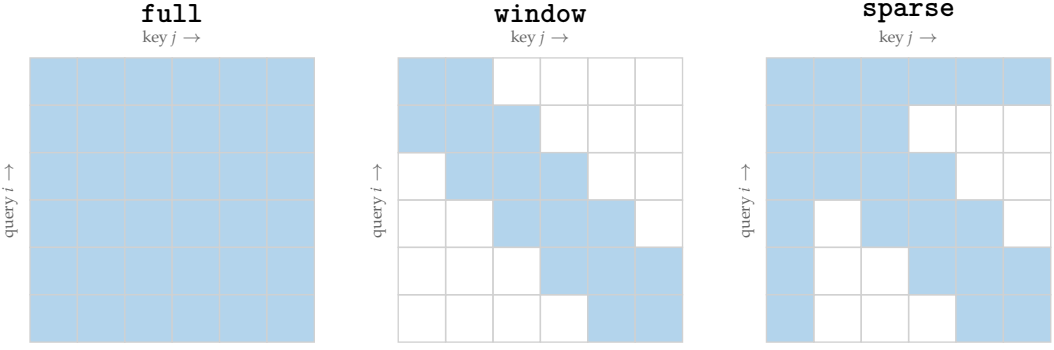
Step K=2: past K/V are **reused** from cache —only the **new** token's K/V column is computed and appended.

Compressing the KV: MQA / GQA / MLA



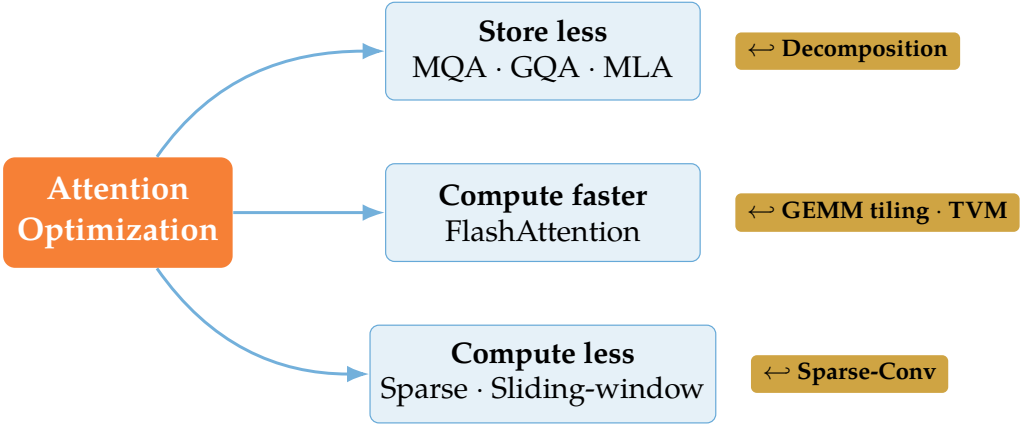
- **MHA:** every query head keeps its own K/V — baseline, largest cache.
- **MQA:** all heads share *one* K/V (max saving, some quality loss); **GQA:** G groups share K/V — the mainstream compromise, $N/G\times$ smaller.
- **MLA** (DeepSeek): a **low-rank compressed latent** KV path *plus* an **independent decoupled-RoPE key** path — not a single low-rank squeeze.

Sparse & Sliding-Window Attention



- **Sliding window:** attend only to the last w tokens $\Rightarrow O(n\,w)$, constant per-step cost.
- **Sparse:** mix local windows with a few *global* tokens (e.g. attention sinks) $\Rightarrow$ cheap long-range links.
- Full is dense $O(n^2)$; both trade a little coverage for big long-context savings.

The Attention-Optimization Landscape



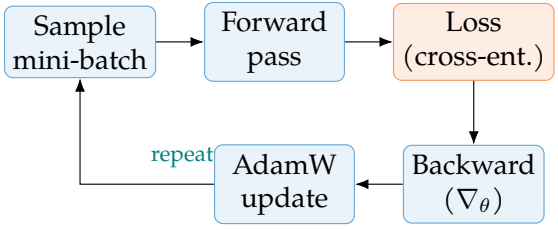
Three routes, **one theme**: every trick is an old systems idea moved to a new battlefield.

Pretrain（预训练）

The Objective & One Training Iteration

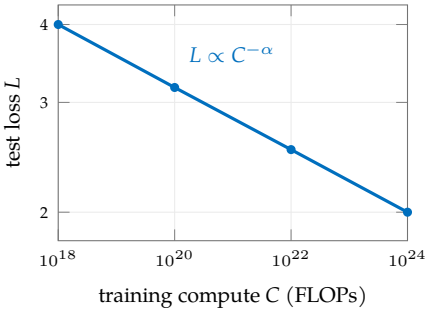
Pre-training **maximizes the likelihood** of the next token = minimizes cross-entropy.

- Loss: $\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t})$
- **Perplexity** $= \exp(\mathcal{L})$ — lower is better.
- Contrast: BERT's *MLM* masks tokens & predicts **bidirectionally**; we predict strictly **left-to-right**.



One optimization step, repeated *millions* of times.

Scaling Laws: Why “Bigger” Works



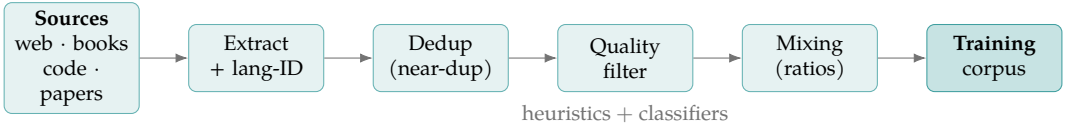
A power law is a *straight line* on log–log axes.

- Loss falls as a **power law** in each of params N, data D, compute C[a].
- **Chinchilla**: at fixed compute, the optimum is $D \approx 20\,N^{b}$.
- Handy identity: training cost $C \approx 6ND$ FLOPs.
- In practice we *over-train* small models — amortized **inference** cost dominates.

[a] **Jared Kaplan, Sam McCandlish, Tom Henighan, et al. (2020). “Scaling Laws for Neural Language Models”.** In: *arXiv:2001.08361*.

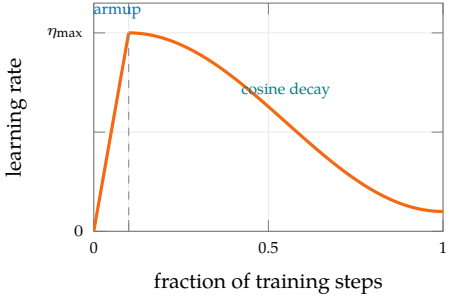
[b] **Jordan Hoffmann, Sarah Borgeaud, Arthur Mensch, et al. (2022). “Training Compute-Optimal Large Language Models”.** In: *arXiv:2203.15556*.

Data Engineering: the Real Ceiling



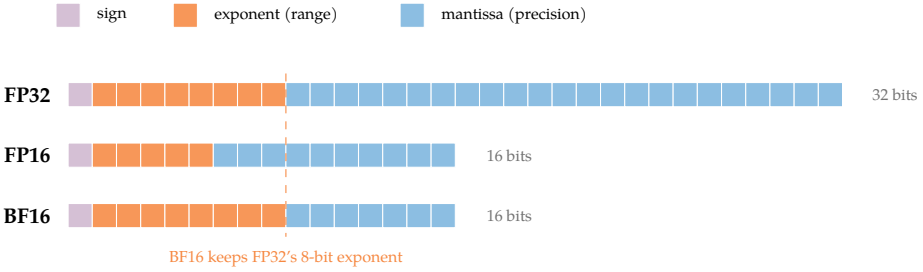
Trillions of raw tokens → a clean, well-mixed corpus. **Data quality often sets the ceiling** more than any modeling trick; the *mixture* (how much code, how much web) is itself a tuned hyper-parameter.

Optimizer & Learning-Rate Schedule



- **AdamW**: per-parameter adaptive step + **decoupled** weight decay — the de-facto standard.
- Schedule: **linear warmup** (stabilize the fragile early phase) $\rightarrow$ **cosine decay** to a small floor.
- **Gradient clipping** (global-norm) caps rare spikes and prevents blow-ups.

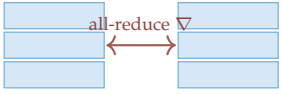
Mixed-Precision Training



- **BF16**: same exponent width as FP32 ⇒ wide dynamic range, hard to overflow — **modern default**. FP8 still exploratory.
- **Loss scaling** is needed for **FP16** (narrow exponent under-flows), *not* for BF16. A FP32 *master weight* copy is kept regardless.

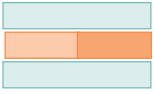
Distributed Training: 3D Parallelism

Data Parallel



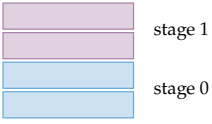
full model $\times 2$,
split the data

Tensor Parallel



slice each layer's
matrix (comm-heavy)

Pipeline Parallel



split by layers;
mind the *bubble*

DP replicates & all-reduces gradients

TP = distributed GEMM, so keep it **within one node**

PP pipelines layers

The Product: a Base Model

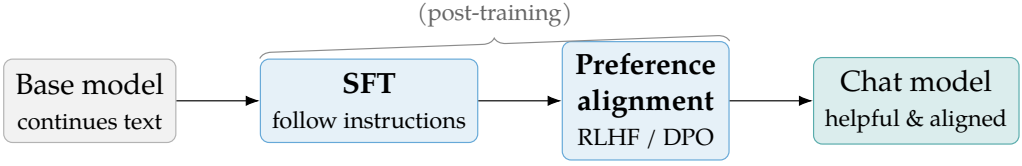
```
 # prompt (an instruction)
Explain photosynthesis to a
five-year-old.
# base model just *continues* the text:
  Explain cellular respiration to a
  ten-year-old. Explain the water
  cycle to a ... [rambles on] ...
```

It pattern-matches a web document, it does *not* answer.

- A base model only **continues text** — it does not *converse* or follow instructions.
- The **capability** is already inside; only the **behavior** is unshaped.
- Turning "continue" into "assist" is **post-training**

Finetune

From Base Model to Assistant



The **framework is stable** (SFT → alignment); the *algorithms* inside iterate fastest in the whole field.

Supervised Fine-Tuning (SFT) / Instruction Tuning

- Train on **(instruction, response)** pairs — supervised, like pretraining but curated.
- Loss is computed **only on the response** tokens; the prompt is masked out.
- Turns a text *continuer* into an **instruction follower**.
- **Data quality** $\gg$ **quantity**: a few thousand clean samples beat millions of noisy ones.

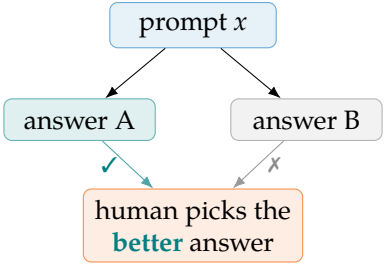
```
 system:   You are a helpful assistant.
user:     List two prime numbers.
assistant: 2 and 3.

_____________ prompt: masked, no loss
_____________ response: loss here
```

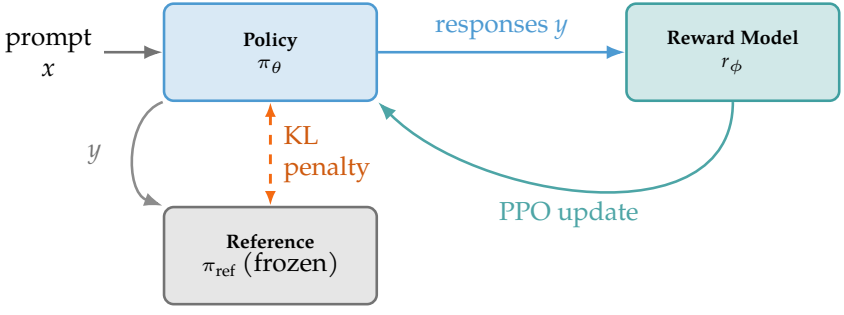
Only the assistant span contributes gradient.

Why SFT Is Not Enough: The Alignment Problem

- SFT only *imitates* demonstrations — but a "good" answer is **hard to enumerate or write out**.
- Humans struggle to *author* the best answer, yet **easily judge which of two is better**.
- Change the signal: from **imitation** (copy demos) to **preference learning** (rank outputs).



Classic RLHF: The Three-Stage Recipe



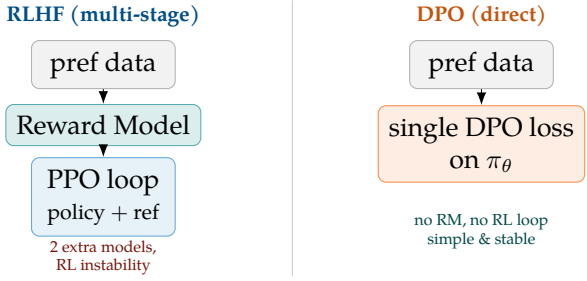
$$\max_{\theta} \quad \mathbb{E}\big[\, r_\phi(y) \,\big] \;-\; \beta\, \mathrm{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)$$

1. Collect response pairs; humans label **preferences**.
2. Train a **Reward Model** r_ϕ to score responses.
3. **PPO** optimizes the policy for reward, with a **KL leash** to the frozen reference[a].

Engineering pain: up to *four* models in memory (policy, reference, RM, critic); RL is **unstable** and costly to tune.

[a] **Long Ouyang, Jeff Wu, Xu Jiang, et al. (2022). "Training Language Models to Follow Instructions with Human Feedback".** In: *NeurIPS*.

DPO: Preference Optimization without a Reward Model



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

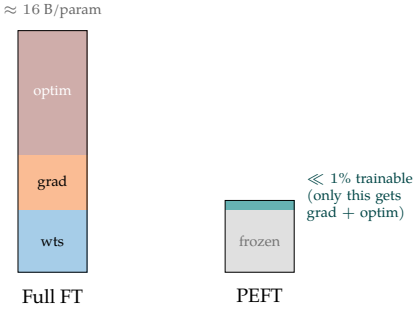
DPO's real cousins: **KTO / ORPO / IPO** — still reward-model-free direct-preference losses[a].

Errata: **GRPO** is *not* a DPO variant. It is a **PPO-family RL** algorithm (group-relative baseline, rule/verifier rewards, used by reasoning models) — it lives on the RL branch.

[a] **Rafael Rafailov et al. (2023). "Direct Preference Optimization: Your Language Model is Secretly a Reward Model".** In: *NeurIPS*.

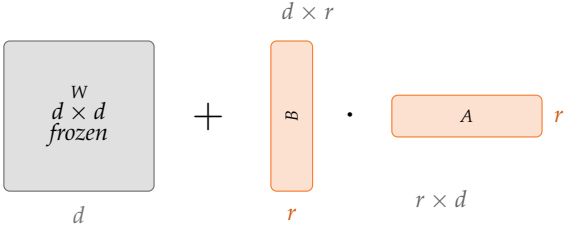
Why Parameter-Efficient Fine-Tuning (PEFT)?

- Full fine-tuning of a 7B model needs **∼100 GB-scale** memory (weights + grads + optimizer states).
- Storing a **full weight copy per task** does not scale to many tasks.
- **Idea:** freeze the backbone, train only a **small increment**.



Only the **trainable** slice needs gradients and optimizer state — and the frozen backbone can even be quantized.

LoRA: Low-Rank Adaptation

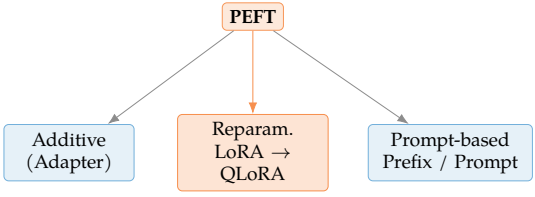


- Freeze W; learn the update as a **low-rank product** $\Delta W = BA$ with $r \ll d$.
- Trainable params drop to **$\sim$1%**: $2dr$ instead of d^2.
- At inference, **merge** $W' = W + BA$ back into the weight — **zero latency overhead**.

This is exactly **low-rank decomposition** from the Decomposition course, reused for fine-tuning[a].

[a] **Edward J. Hu et al. (2022). "LoRA: Low-Rank Adaptation of Large Language Models".** In: *ICLR*.

QLoRA & the PEFT Family

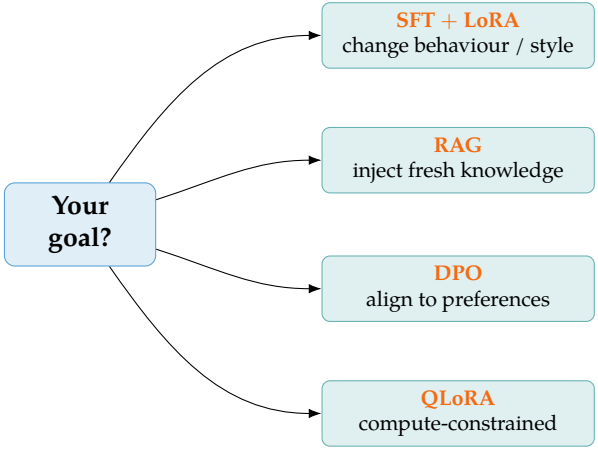


- **QLoRA:** 4-bit *quantized* frozen backbone + LoRA ⇒ fine-tune large models on **one GPU**.
- **Adapters:** small trainable MLP blocks inserted between layers.
- **Prompt family:** learn soft prompts (Prefix / Prompt / P-Tuning), backbone untouched.

QLoRA = **Quant** + **Decomposition** — two prior courses combined[a].

[a] **Tim Dettmers et al. (2023). "QLoRA: Efficient Finetuning of Quantized LLMs".** In: *arXiv preprint*.

Choosing a Fine-tuning Strategy



Knowledge injection is usually better served by **RAG** than by fine-tuning