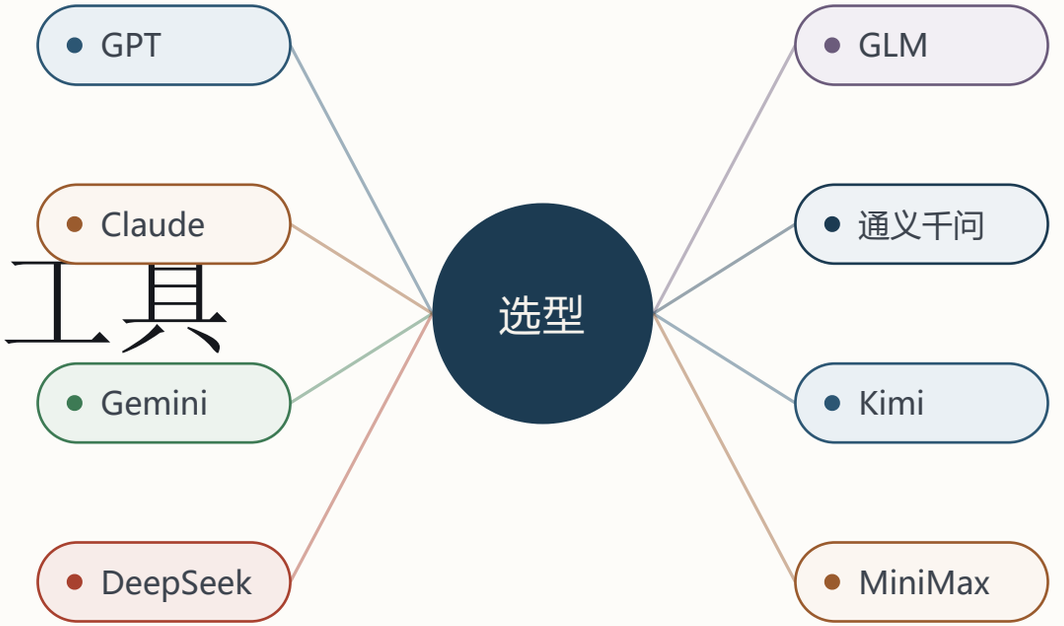
COURSE 02 · AI MODELS

认识主流 AI 大模型与开发工具

选型不靠榜单，靠场景、显存和一次成型的概率



AI 教学课件工厂 · 系列课程 02

面向技术选型与工具使用者 / 90 分钟

两类模型，干的不是一件事

语言模型在写字，视频模型在画帧，混着比没有意义。

01 语言模型

把文字接成文字，产出是可以改、可以发的段落

02 视频模型

把提示词或图片变成逐帧画面，产出是能播的像素

03 计算方式

一个算下一个词的概率，一个算帧与帧之间怎么接

04 厂商重合

腾讯、MiniMax 两类都做，但产品线和价格不通用

先说清要文字还是要画面，再去比模型；混着比，榜单上的分数没有参考价值。

它其实在算下一个词

理解了这一页，它为什么会在没把握时也说得像真的，就解释得通了。

01 输入：你说的话被切成 token，排成一个序列

02 上下文：历史对话和贴进来的资料一起占这个窗口

03 生成：每次算一个 token 的概率，挑一个接上去，再算下一个

04 输出：整段接完一次性还给你，中途没有查库这个动作



它不是检索，是续写。想让它少编，办法是把资料真的放进去，不是提醒它别编。

上下文窗口：一次能看多少

窗口大小分三档，但窗口大不等于读得准。

01 窗口：一次能装进去的 token 上限，输入和输出一起算

02 百万档：Kimi、DeepSeek、GLM 的长文版，整本手册能塞进去

03 256K 档：Claude、通义千问的主力配置，百页合同提取够用

04 128K 档：GPT、Llama 的常见配置，日常长文够，超了要分段

百万档 Kimi / DeepSeek / GLM	100 万 token
256K 档 Claude / 通义千问	256K
128K 档 GPT / Llama	128K
共同问题 中间段落容易被漏掉	长文

输入越长，中间段落越容易被漏掉；重要的信息放开头或结尾。

云模型和开源模型，差在哪

一个按次付费，一个搬回自己家，分界是数据和调用量。

01 云模型

按 token 付费，能力最强的那批只在这里

02 开源模型

权重能下载，装在自己机器上，数据不出门

03 能力差

同代开源模型通常落后云端旗舰半年到一年

04 成本差

高频重复调用自己部署更省，低频杂活云端更省

先问数据能不能出门，再问一个月调用多少次；这两个答案基本就把方向定了。

海外三家：GPT、CLAUDE、GEMINI

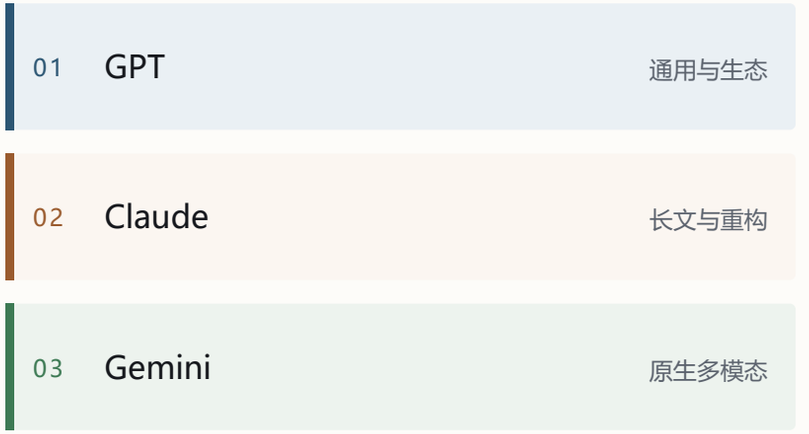
差距不在谁更聪明，在三条不同的长板。

01 GPT：通用能力和生态最成熟，复杂链路开发默认从它开始

02 Claude：长文档写作、跨文件重构最稳，幻觉率低

03 Gemini：原生多模态，图文视频混着处理，定价对开发者友好

04 共同点：国内直连都要额外解决网络和合规问题



先问你的任务卡在哪：长文、多模态，还是生态和工具链。

国内主力：四个常用选择

内部工具看 DeepSeek，接工具看 GLM，上设备看通义，读长文看 Kimi。

01 DeepSeek

中文、代码、数学扎实，接口便宜，国内编程最强一档

02 智谱 GLM

Function Calling 成熟，多 Agent 协作和工具调用稳

03 通义千问

0.5B 到几十 B 全覆盖，小参数版适合边缘设备

04 Kimi

超长无损上下文，几十万字文档的深度阅读是主场

内部工具和知识库先看 DeepSeek；要接一堆工具、跑多步任务，先看 GLM。

全模态、微信生态、合规落地

选它们的理由通常不是分数，是生态。

01 MiniMax：文本语音图像视频音乐五模态都有，语音合成是王牌

02 腾讯混元：Hy4 是 770B 总参数、49B 激活，百万上下文

03 百度文心：合规落地案例最多，政务和教育行业铺得广

04 选择理由：要语音、要微信、要政企资质，模型分数解决不了

MiniMax
五模态 + 语音合成
全模态

腾讯混元
Hy4 · 770B / 49B 激活
百万上下文

百度文心
政务教育落地最多
合规

需要语音或微信生态时，模型分数往往不是决定因素。

差距在哪：架构和窗口

三类架构对应三种成本，参数总量会误导人。

01 密集架构：全参数参与推理，精度上限高，成本也最高

02 MoE 架构：每次只激活少量专家，成本能压到密集模型的三到五分之一

03 原生多模态：端到端处理图文音视频，不用先转成文字再喂

04 看激活参数：770B 总参数、49B 激活，跑起来按 49B 算

成本结构

- 密集：全参数推理
- MoE：只激活一部分
- 成本差三到五倍

模态结构

- 原生多模态：端到端
- 不用先转成文字
- 信息损耗更小

看参数总量会被误导；MoE 模型真正决定速度和成本的是激活参数。

够不够用，看四个维度

一次成型率、复杂工程、性价比、场景匹配，比榜单直接。

01 一次成型率：第一稿能不能直接用，决定你要不要来回改五轮

02 复杂工程能力：跨文件、多步骤的任务，差距在这一项拉得最开

03 性价比：清 C 盘、改格式这种活，用最贵的模型纯属浪费

04 场景匹配：查开源项目做本地部署，不涉及开发，各家差别不大



做 DEMO 先用便宜的出思路，要提效和一次成型再换 DeepSeek、GPT 这类。

想自己跑，先看显卡

能不能跑不看 CPU 多强，只看显存装不装得下。

01 CPU：四到八个强核心，擅长逻辑判断和顺序执行

02 GPU：几千个精简核心，擅长海量重复计算，也就是矩阵运算

03 显存：模型权重得整个装进去，装不下就是跑不起来

04 分水岭：2006 年 CUDA 出来，GPU 才从画图变成算力

CPU	GPU
四到八个强核心	几千个精简核心
擅长逻辑与顺序	擅长矩阵运算
单任务延迟低	吞吐高几十倍

本地部署的第一道门是显存，不是预算，也不是技术难度。

按显存选模型

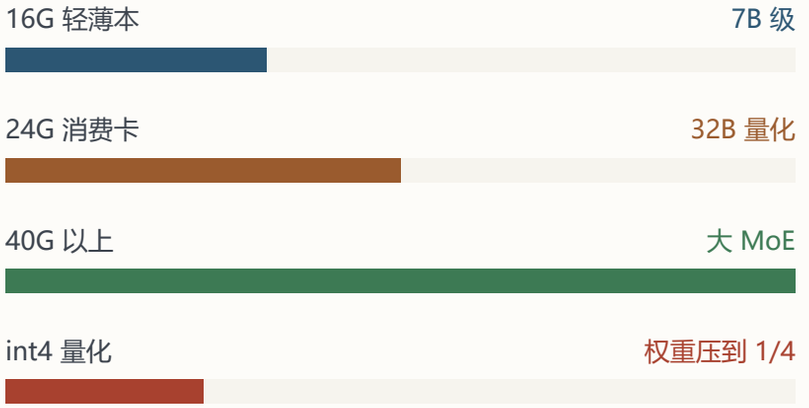
三档显存，对应的模型是固定的，不用猜。

01 16G 及以下：Phi-3-mini 量化后 2.3G，Qwen2-7B 约 5.1G

02 24G 消费卡：DeepSeek R1 蒸馏 32B，量化后单卡能流畅跑

03 40G 以上：DeepSeek V4、Kimi K3 这类大 MoE，要多卡集群

04 量化：int4 把权重压到四分之一，代价是精度略降



先看显存再挑模型；反过来挑，装不下就是白忙一场。

没有显卡：租算力

按分钟计费，关机不扣费，用法就三个动作。

01 AutoDL

体量大、老牌，镜像多，但 API 限制多、开机要手动

02 优云智算

API 能直接开关机，闲置 7 天释放，比 AutoDL 短

03 价格

3090/4090 按分钟计，每小时约 1.18 到 2.22 元

04 关键

关机不扣费，跑完不关，账单会替你记住

租卡的正确用法是开、用、关三个动作，不是一直挂着。

本地部署换来什么，失去什么

这是拿运维成本换数据边界，不是一条省钱的路。

01 换来：数据不出门，断网能用，调用不计费

02 失去：能力落后云端半年到一年，长任务容易断

03 要自己管：显存、并发、模型更新、故障排查都归你

04 适合：数据敏感、调用量大、有明确离线要求的场景



三条不占就别折腾本地部署：数据敏感、调用量大、必须离线。

开发工具：真正在用的五个

选择标准不是补全快不快，是能不能读到整个项目。

01 WorkBuddy

腾讯云全栈开发助手，代码、调试、部署一条线

02 Codex

GitHub Copilot 的前身，IDE 覆盖最全，装机量最高

03 Claude Code

能一次加载整个代码库，跨文件重构最稳

04 OpenCode / Cursor

开源可私有化，和 AI 原生 IDE 两条路

读不到整个项目的工具，只能靠猜，补全再快也补不出结构。

六个场景，六个工具

办公工具的价值在不用换软件，装在工作流里才有人用。

01 WPS AI：嵌在 Word、Excel、PPT 里，零学习成本

02 Kimi：几十份合同一次上传，做跨文档对比

03 飞书妙记：会议实时转写，会后自动出纪要和待办

04 Gamma / Perplexity：一个出 PPT，一个带引用来源的搜索



调研类工作先看来源能不能点开核对，这比答案本身重要。

先两个，再三个

别一次学五个工具，先把一个用到能替你干活。

01 主线

WorkBuddy 和 Codex，
先把日常开发跑通

02 次选

DeepSeek harness、
MiniMax Code、OpenCode

03 顺序

先用现成产品，再学换模
型，最后才碰本地部署

04 判断

能把一件真实工作
交出去，才算学会

一上来就研究部署的人，往往连一个真实任务都没交出去过。

选型是排除法，不是排名

今天带走三样：两类模型的分界、显存决定能不能本地跑、工具先精一个。

课后作业：把你最常做的一件事，用两个不同档位的模型各跑一遍，记下差在哪、多花了多少钱。

