北京大学
PEKING UNIVERSITY

DeepSeek内部研讨系列

DeepSeek 私有化部署和一体机

AI肖睿团队

（韩露、王春辉、顾跃、李娜、谢安明、陈钟）

20250224@北京

- 北大青鸟人工智能研究院
- 北大计算机学院元宇宙技术研究所

摘要

1. 本次讲座为DeepSeek原理和应用系列研讨的讲座之一，让大家可以决策是否需要自己部署DeepSeek系列模型，并了解自己本地化部署DeepSeek的基本方法，同时了解更专业的企业级部署方法，有助于选择DeepSeek一体机型号，并能理解DeepSeek云服务的工作机制和原理，用好DeepSeek云服务的API调用方法。
2. 本讲座的内容分为四个主要部分：
   ① 首先，除了DeepSeek满血版之外，还有各种各样的蒸馏版和量化版，以及DeepSeek的不同专业模型。我们将介绍人工智能大模型的基本概念，以及DeepSeek各个模型的特点与适用场景，对比不同规模模型的性能表现，帮助大家选择最适合自己需求的版本。
   ② 其次，对于普通用户在自己的电脑上部署和体验DeepSeek（蒸馏版）的情况，我们会评估和建议硬件要求，演示如何通过Ollama命令行高效部署DeepSeek模型，解决下载过程中可能遇到的常见问题。为了提升在自己的电脑上安装DeepSeek模型后，大家与DeepSeek模型的交互体验，我们还将介绍Open WebUI和Chatbox等前端展示工具的配置与使用方法。
   ③ 然后，对于专业级的企业部署DeepSeek，或把DeepSeek（蒸馏版和满血版）部署在专业的昂贵的推理机上，本讲座将探讨基于Transformers快速验证和vLLM的高性能部署方案，并提供真实企业基于vLLM的部署DeepSeek-70b的相关数据和经验。
   ④ 最后，作为补充内容，针对计算资源受限的场景，我们专门设计了"低成本部署"环节，详细讲解Unsloth R1动态量化部署的三种实现路径：基于llama.cpp、KTransformers以及Ollama框架动态量化部署。
3. 在技术学习的道路上，优质学习资源至关重要。推荐大家参考《人工智能通识教程（微课版）》这本系统全面的入门教材，结合B站“思睿观通”栏目的配套视频进行学习。此外，欢迎加入ai.kgc.cn社区，以及“AI肖睿团队”的视频号和微信号，与志同道合的AI爱好者交流经验、分享心得。

目 录

CONTENTS

PART 01 ▸

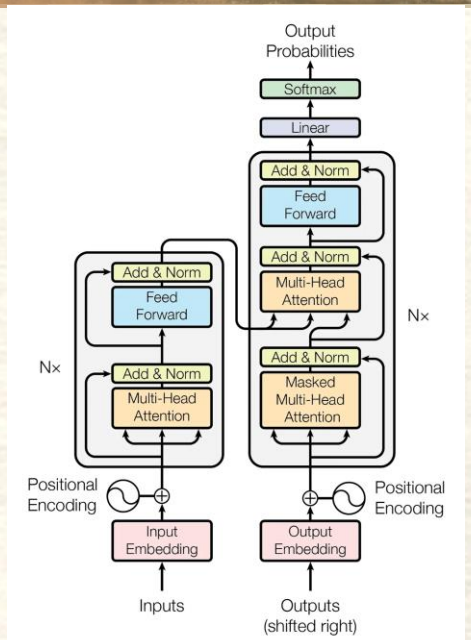
人工智能与DeepSeek

大模型相关术语

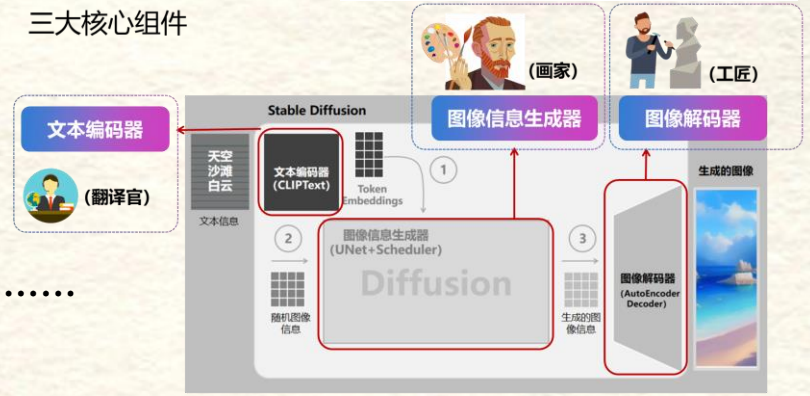
- 多模态
  - 文本、图片、音频、视频
- AI工具（国内）
  - DeepSeek、 豆包、Kimi、腾讯元宝、智谱清言、通义千问、秘塔搜索、微信搜索...
- 通用模型
  - 大语言模型（LLM，**Large Language Model**）
    - 生成模型
    - 推理模型
  - 视觉模型（图片、视频）
  - 音频模型
  - 多模态模型
  - ……
- 行业模型（垂直模型、垂类模型）
  - 教育、医疗、金融、办公、安全等

大模型的前世今生

- **人工智能**：让机器具备动物智能，人类智能，非人类智能（超人类智能）
  - 运算推理：规则核心；自动化
  - 知识工程：知识核心；知识库+推理机
  - **机器学习**：学习核心；数据智能（统计学习方法，数据建模）
    - 常规机器学习方法：逻辑回归，决策森林，支持向量机，马尔科夫链，…..
    - **人工神经网络**：与人脑最大的共同点是名字，机制和架构并不一样
      - 传统神经网络：霍普菲尔德网络，玻尔兹曼机，…..
      - 深度神经网络：**深度学习**
        - 传统网络架构：DBN，CNN，RNN，ResNet，Inception，RWKV，……
        - **Transformer架构**：可以并行矩阵计算（GPU），核心是注意力机制（Attention）
          - 编码器（BERT）：多数embedding模型，Ernie早期版本，…….
          - 混合网络：T5、GLM
          - 解码器（GPT）：**大语言模型（LLM）**，也是传统的多模态模型的核心
            - **生成式人工智能（GenAI）：AIGC**
            - DeepSeek、Qwen、GLM、Step、MiniMax、hunyuan、kimi、火山、……
            - OpenAI GPT（ChatGPT）、Claude、Llama、Grok、……
        - **Diffusion架构**：主要用于视觉模型（比如Stable Diffusion、DALL.E），现在也开始尝试用于语言模型
          - Diffusion+Transformer架构：例如Sora的DiT（加入Diffusion的视觉模型），部分新的多模态模型架构



三大核心组件



大模型的发展阶段

生成模型 → **推理模型**

ref:中文大模型基准测评2024年度报告(2025.01.08)

生成模型与推理大模型的对比

北京大学 PEKING UNIVERSITY

比较项	OpenAI GPT-4o（生成模型）	OpenAI o1（推理模型）
模型定位	专注于通用自然语言处理和多模态能力，适合日常对话、内容生成、翻译以及图文、音频、视频等信息处理、生成、对话等。	侧重于复杂推理与逻辑能力，擅长数学、编程和自然语言推理任务，适合高难度问题求解和专业领域应用。一般是在生成模型的基础上通过RL方法强化CoT能力而来
推理能力	在日常语言任务中表现均衡，但在复杂逻辑推理（如数学题求解）上准确率较低。	在复杂推理任务表现卓越，尤其擅长数学和代码等推理任务。
多模态支持	支持文本、图像、音频乃至视频输入，可处理多种模态信息。	当前主要支持文本输入，不具备图像处理等多模态能力。
应用场景	适合广泛通用任务，如对话、内容生成、多模态信息处理以及多种语言相互翻译和交流；面向大众市场和商业应用。	适合需要高精度推理和逻辑分析的专业任务，如数学竞赛、编程问题和科学研究；在思路清晰度要求高的场景具有明显优势，比如采访大纲、方案梳理。
用户交互体验	提供流畅的实时对话体验，支持多种输入模态；用户界面友好，适合大众使用。	可自主链式思考，不需要太多的过程指令，整体交互节奏较慢。

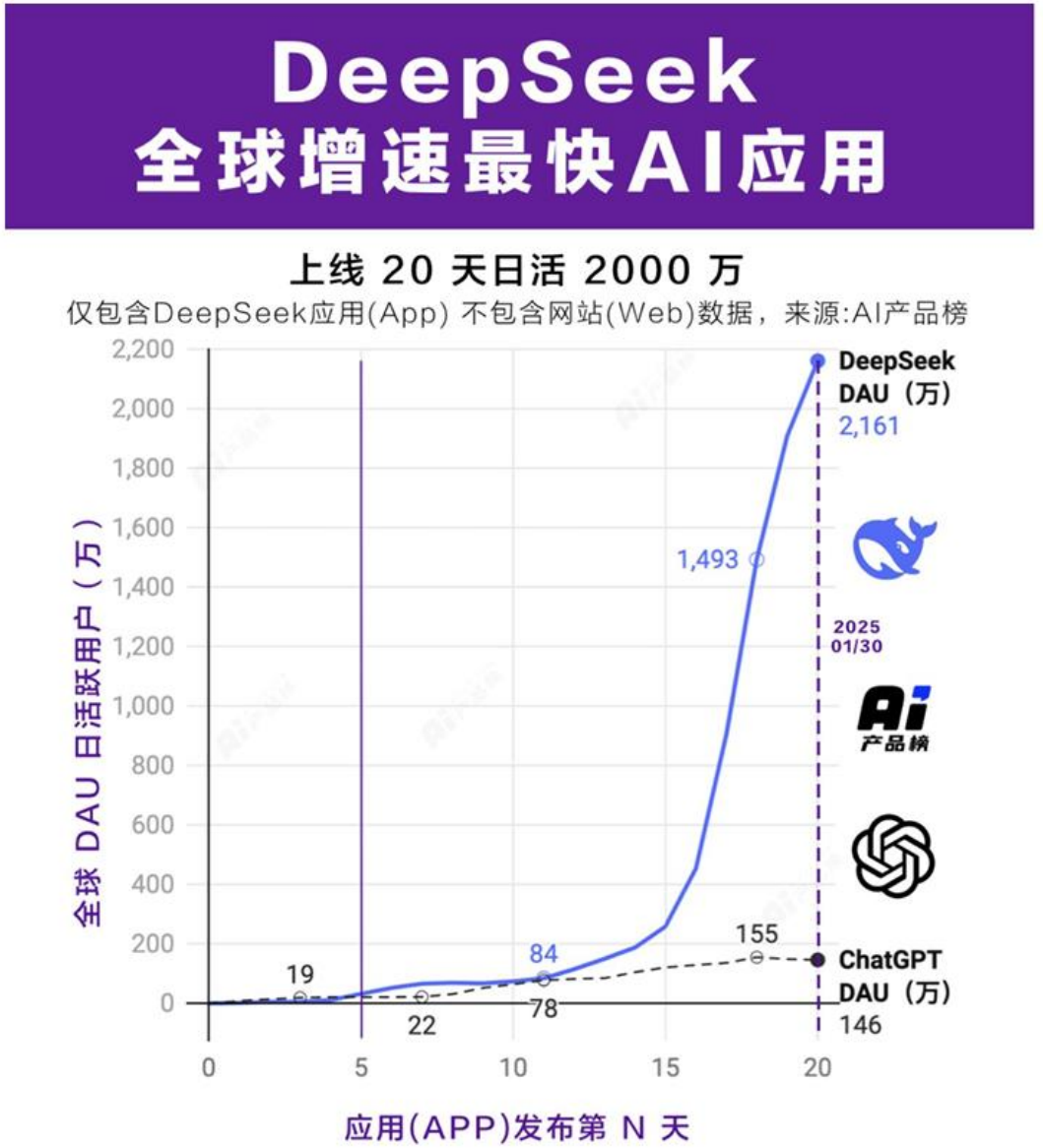
- 普通大模型是玩知识和文字的，推理大模型是玩逻辑的，至于计算问题，还是找计算器吧
- 推理模型也不是万能的，其幻觉通常比生成模型大，很多不需要强推里的场合还是传统的生成模型比较适合

自 2024 年起，AI肖睿团队便接入DeepSeek V2，持续应用 DeepSeek 技术体系，历经 V2（MoE 架构）、V3（MTP）探索，现已在生产项目中接入 DeepSeek R1（满血版）。其中，V2和V3都是生成模型，R1为推理模型。

下面，基于我们团队对DeepSeek的技术研究和实战经验，为大家系统梳理这三大模型技术特性，剖析知识蒸馏在各版本提升效率的逻辑，并结合边缘计算、数学编程、中文等场景实例，对比DeepSeek R1各版本模型的计算效率与推理精度的演进，同时说明标准化 API/SDK 对技术落地的适配机制。

DeepSeek快速出圈

2025年1月20日下午，中共中央政治局常委、国务院总理李强主持召开专家、企业家和教科文卫体等领域代表座谈会，听取对《政府工作报告（征求意见稿）》的意见建议。DeepSeek公司创始人梁文峰作为企业家代表之一参加了此次座谈会。



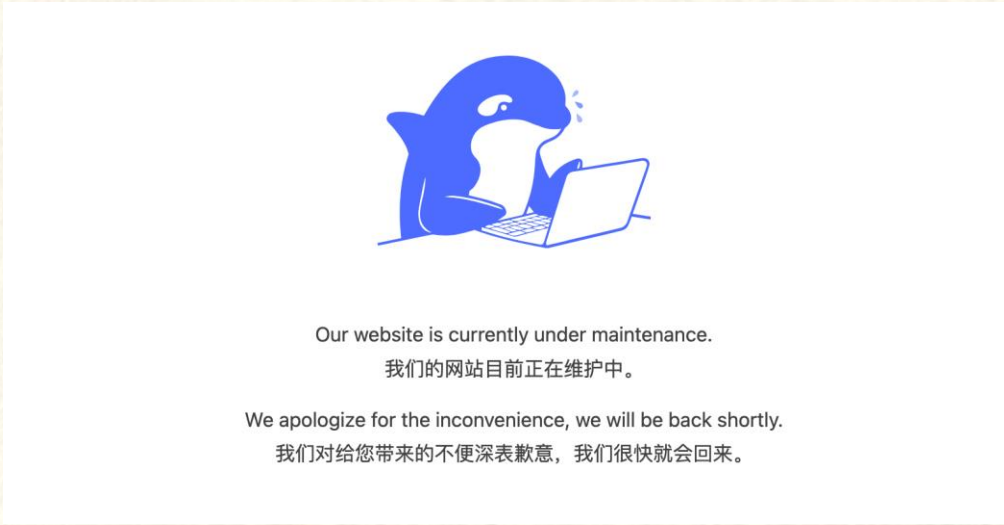
DeepSeek-“服务器繁忙”

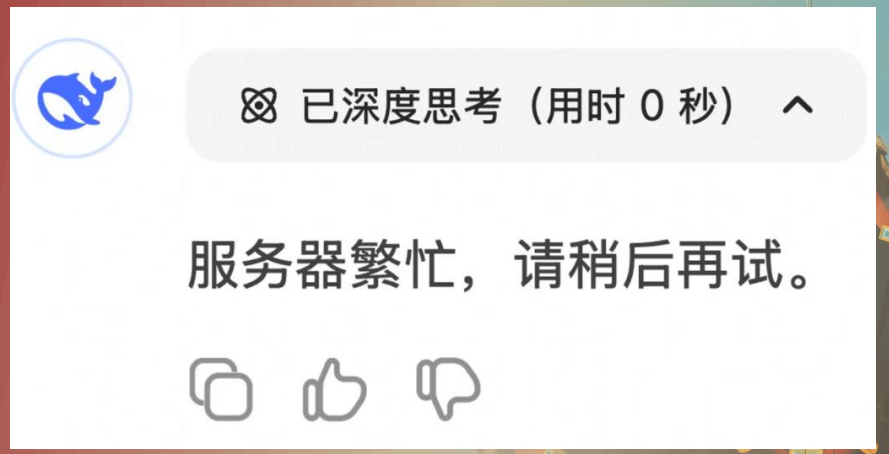
2024年12月26日，DeepSeek因推出对标GPT 4o的语言模型DeepSeek V3，首先在美国的人工智能行业内部一起轰动。

2025年1月20日，DeepSeek继续发布对标OpenAI o1的语言模型DeepSeek R1。由于“深度思考”模式生成的答案的过程可视化，完全开源模型参数和技术方案，采用极致的模型架构优化和系统优化手段降低了模型的训练和推理成本，加之完全由中国本土团队制造，让DeepSeek公司和DeepSeek R1模型在蛇年春节前后彻底出圈。

春节之后，随着大量用户的快速涌入，DeepSeek 官方提供的模型服务就一直在经历拥堵，它的联网搜索功能间歇性瘫痪，深度思考模式则高频率提示“服务器繁忙”，此类现象让大量用户倍感困扰。

这也揭示了AI时代和互联网时代的底层逻辑的不同：在互联网时代，用户使用一个系统的成本很低，边际成本接近于零，但在AI时代，用户使用一个系统的成本比较高，后天会有大量的算力需求和token消耗，边际成本下降并不明显。





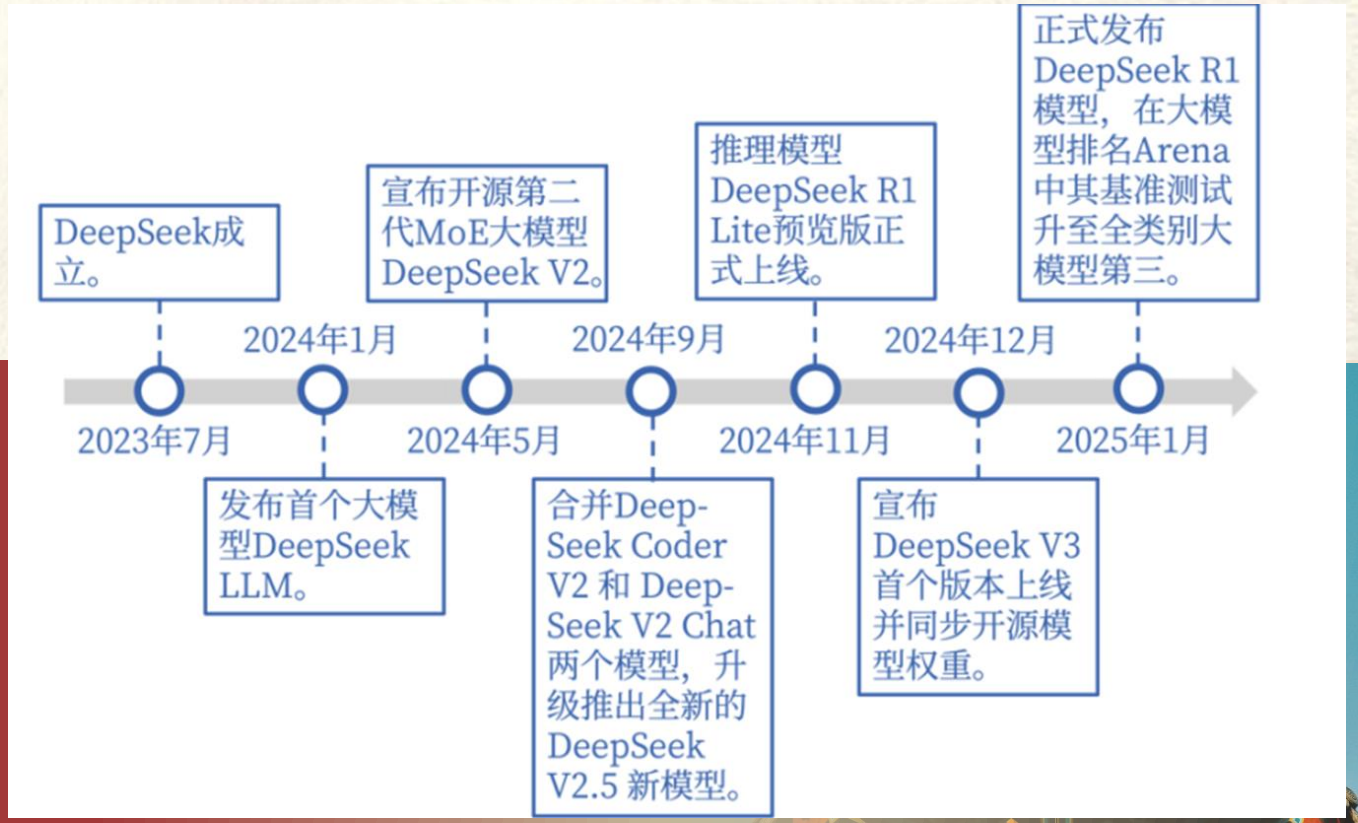
公司与模型介绍

01

公司简介

DeepSeek于2023年7月成立，是幻方量化孵化出的一家大模型研究机构，团队分布在中国杭州和北京，是中国大模型七小虎之一。

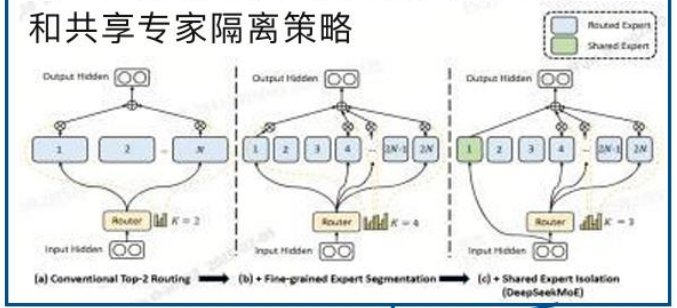
除了DeepSeek之外，其它六家也被投资界称为中国大模型企业六小龙（智谱AI、百川智能、月之暗面、零一万物、阶跃星辰、MiniMax）。



模型的演进历史和特点

北京大学 PEKING UNIVERSITY

在MoE架构中提出细粒度专家分割和共享专家隔离策略

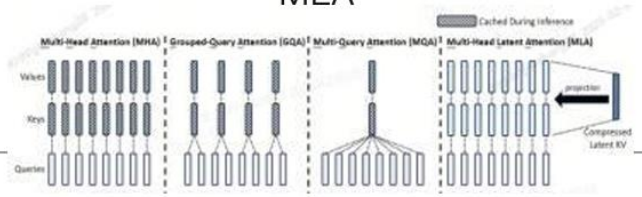


- 负载优化： MoE中采用无辅助损失的负载均衡策略
- 通信优化： DualPipe算法，精细控制分配给计算和通信的GPU SM数量；高效的跨节点通信内核。
- 内存优化：重计算、参数共享等。
- 计算优化： FP8混合精度、细粒度量化、低精度存储、在线量化等。
- 推理优化：多Token预测 （MTP）

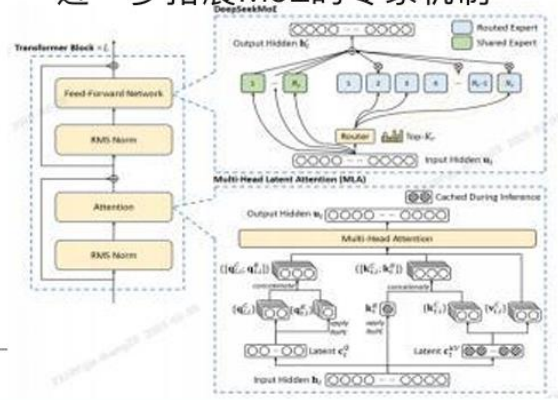
成立	DeepSeekLLM	DeepSeekMoE/ Coder/Math/VL	DeepSeek–V2	DeepSeek–V3	DeepSeek–R1–Zero DeepSeek–R1
2023.07	2024.01		2024.05	2024.12	2025.01

- 7B， 67B
- 2T tokens
- 分组查询注意力
- SFT， DPO

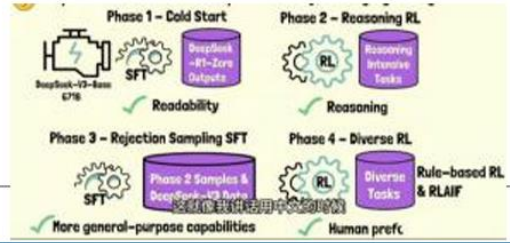
MLA



- 引入多头潜在注意力
- 进一步拓展MoE的专家机制



- 纯强化学习：仅通过RL、无SFT ，展现了自我进化能力。
- 四阶段循环的训练方式：基础→ RL →微调→ RL →微调。
- 蒸馏：比小型模型直接RL更有效、更通用。



三个主要的DeepSeek模型的区别

对比维度	DeepSeek V2	DeepSeek V3	DeepSeek R1
核心架构	• DeepSeekMoE（混合专家架构改进版） • 总参数量 2360 亿 • 单次激活 210 亿参数	• 升级版 DeepSeekMoE 架构 • 总参数量 6710 亿 • 单次激活 370 亿参数	• 与V3模型相同
训练方法	• 传统预训练 + 监督微调（SFT）+ 强化学习（RL） • 数据量 8.1 万亿 tokens	• 预训练 + SFT + MTP+RL • 引入 GRPO 算法提升RL效率和效果 • 数据量14.8万亿tokens	• 跳过 SFT，直接通过RL激发推理能力 • 采用两阶段 RL 和冷启动技术
部分关键特性	• 首次引入 MoE 架构，并进行了改进你	• 无辅助损失的负载均衡 • 代码任务生成速度提升至 60 TPS	• RL驱动推理优化 • 模型蒸馏实验（可迁移至小模型） • Zero版验证了自我进化能力
性能表现举例	• 生成速度20TPS，适合通用生成任务	• 综合 NLP 任务接近 GPT-4o • MMLU 知识理解 88.5% • API 成本大幅降低	• 数学推理（MATH-500 97.3%） • 代码生成（Codeforces）与openAI-o1-1217相当

模型简介DeepSeek-V2

北京大学 PEKING UNIVERSITY

模型简介

DeepSeek-V2模型与DeepSeek LLM 67B相比，DeepSeek-V2实现了更强的性能，同时节省了42.5%的训练成本，减少了93.3%的KV缓存，并将最大生成吞吐量提升至5.76倍。

设计初衷

DeepSeek-V2旨在解决现有大语言模型训练成本高、推理效率低的问题。通过引入MoE架构，它在保持高性能的同时，大幅降低训练成本和推理时间，为广泛的应用场景提供支持。

核心原理

DeepSeek-V2基于混合专家（Mixture-of-Experts，MoE)架构，将任务分配给多个专家模型，每个专家专注于特定子任务，从而提升性能和效率。模型总参数量为236B，其中21B参数在每个token上被激活，使其在处理复杂任务时更灵活高效。

模型简介DeepSeek-V3

北京大学 PEKING UNIVERSITY

模型简介

DeepSeek V3模型采用的模型架构与V2模型差不多，都采用MLA和MoE。V3在V2的基础上主要是增加了多令牌预测（Multi-Token Prediction, MTP）的训练目标。

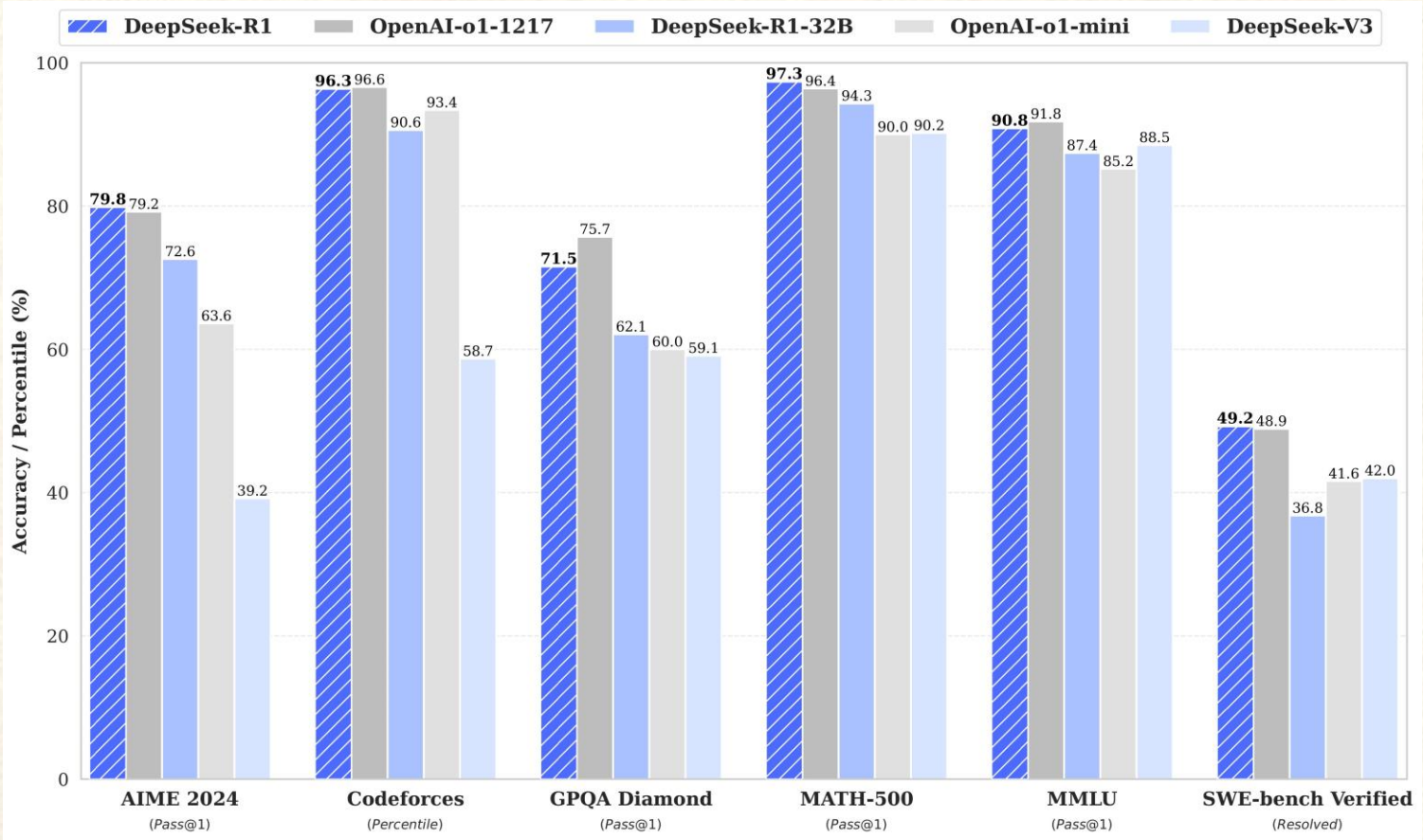
我们都知道大模型是自回归模型，在回答问题时，本质上是一个字一个字的预测出来的，而MTP实现了类似同时预测多个字的效果。

- 准确地讲，这里不是“字”，是“token”。这里用“字”，主要是便于理解。

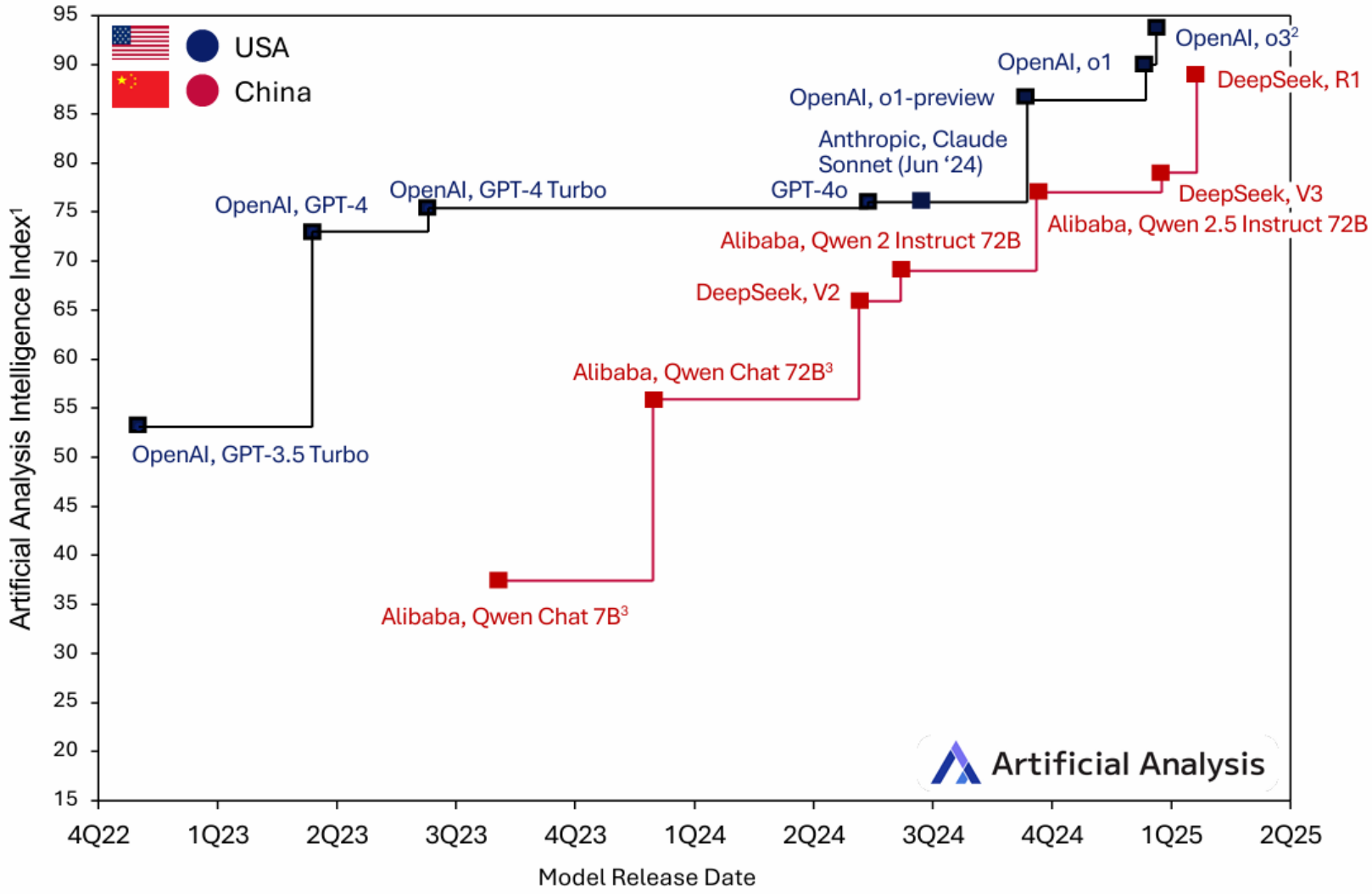
模型简介DeepSeek-R1

与以往的研究不同，R1模型通过强化学习而非监督学习的方式显著提升了大模型的在数学和逻辑推理任务中的表现，验证了强化学习在提升模型推理能力上的作用。

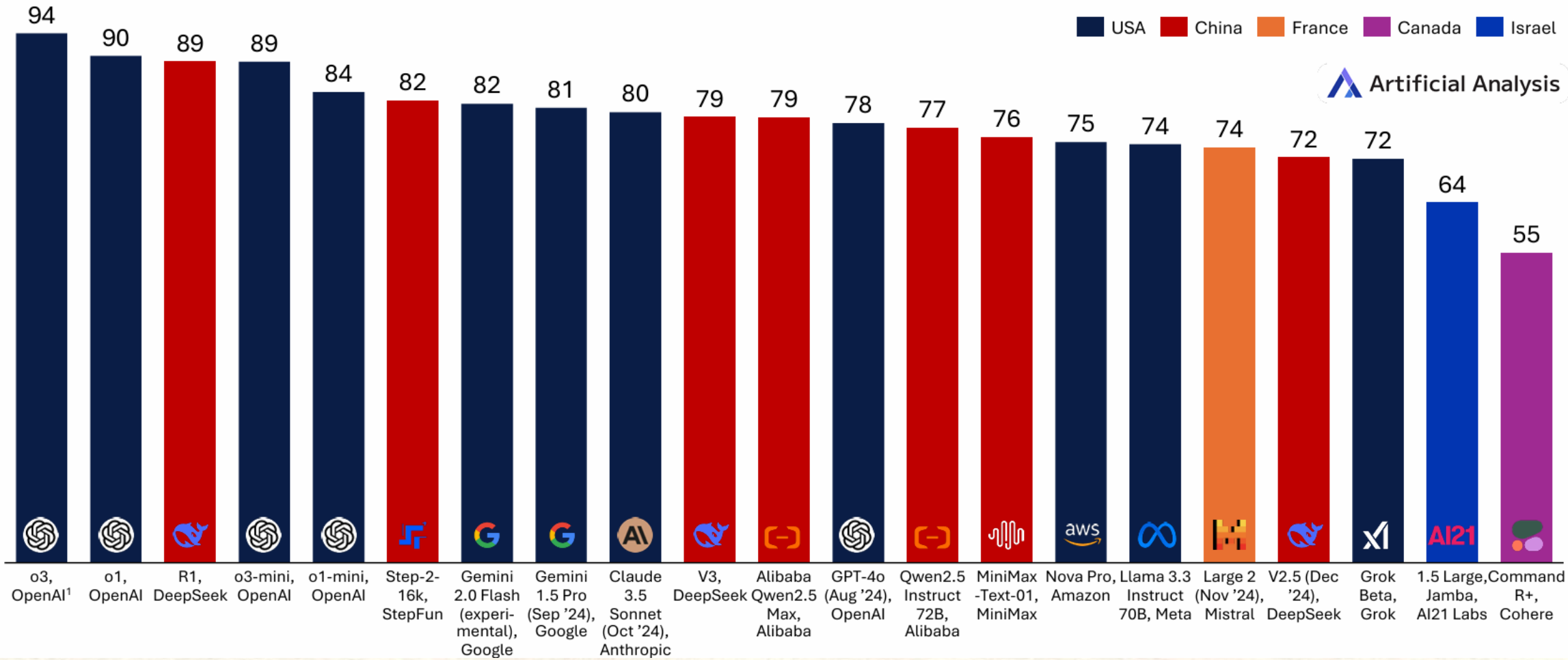
通过强化学习自动学习复杂的推理行为（自我验证与反思），然后随着训练的深入，模型逐步提升了对复杂任务的解答能力，并显著提高了模型推理能力。在数学和编程基准测试集上，与open AI-o1模型的表现相当并大幅超越其它现有大模型。



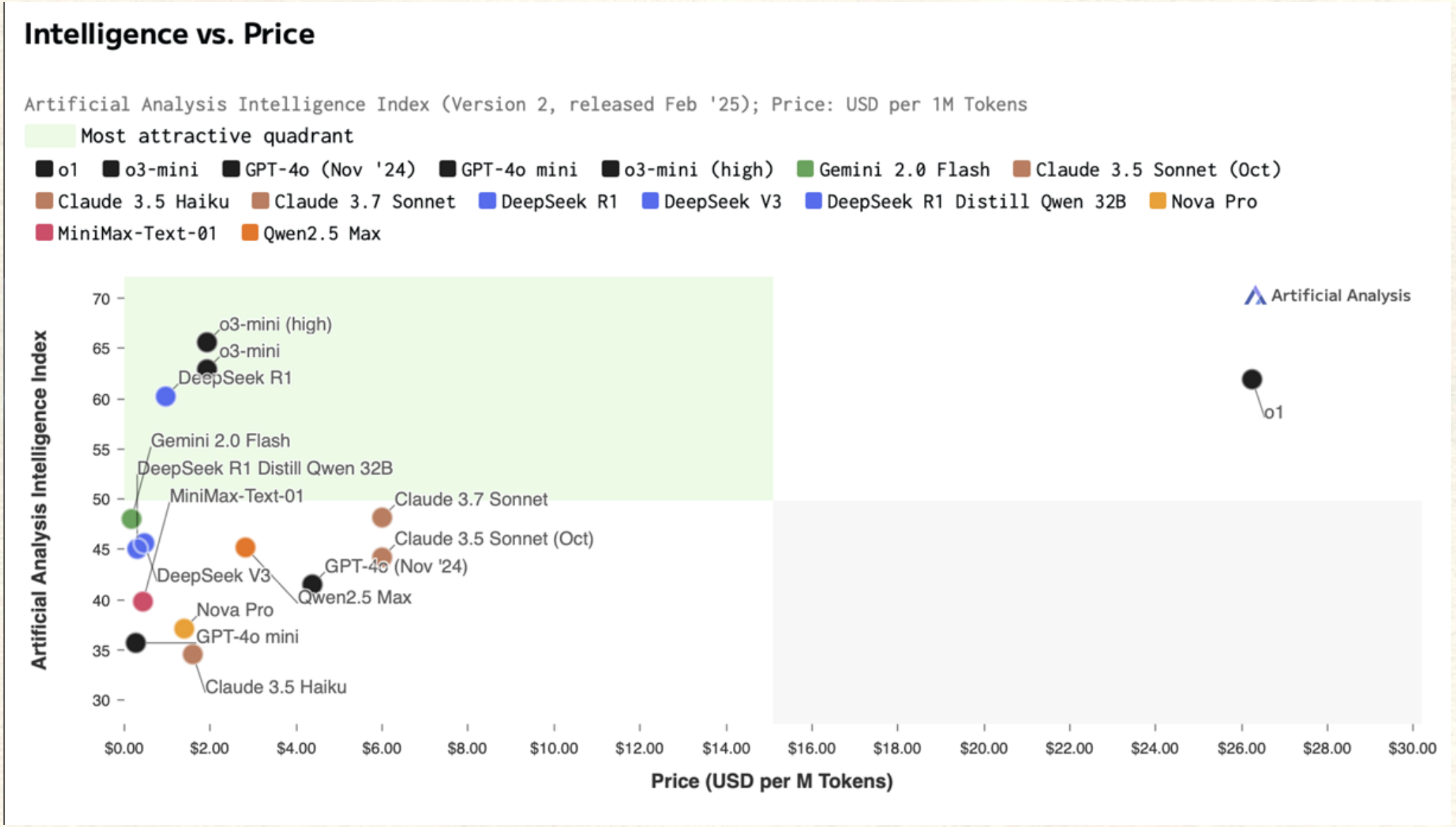
DeepSeek模型优势



DeepSeek模型优势



DeepSeek模型优势



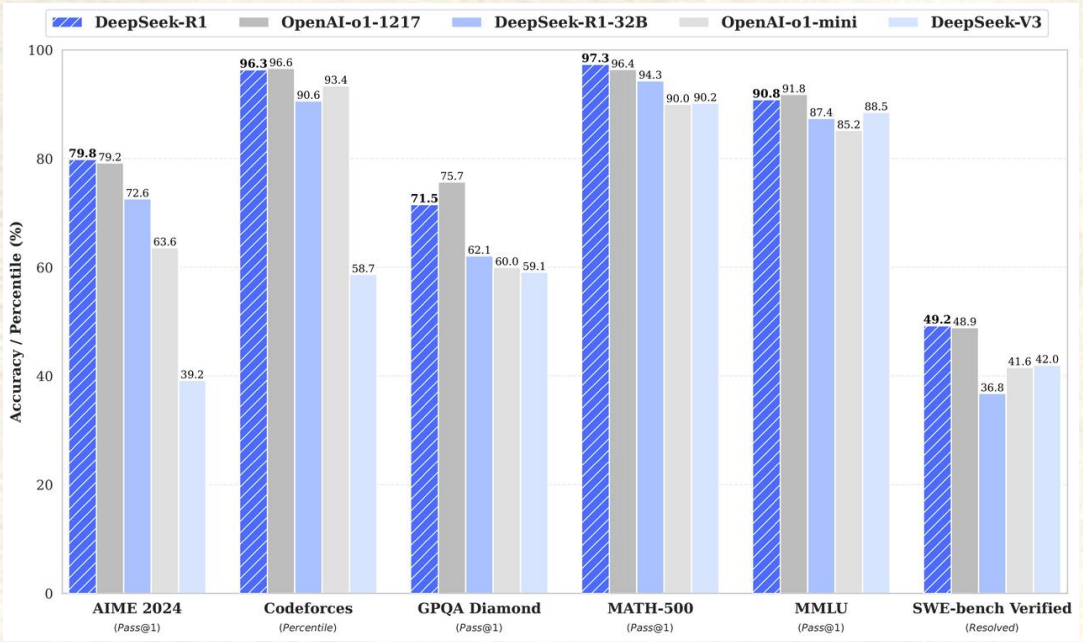
DeepSeek的算法和算力突破

北京大学 PEKING UNIVERSITY

- DeepSeek R1达到了跟o1相当、或者至少接近的推理能力，且将推理过程可视化
- 它做到这个水平只用到少得多的资源，所以价格十分便宜
- 它是完全开源的并且还发布论文，详细介绍了训练中所有的步骤和窍门
- DeepSeek深度求索公司是一家纯粹的中国公司

Deepseek官网地址：
http://ai.com
https://chat.deepseek.com

DeepSeek-R1 训练技术全部公开，论文链接：
https://github.com/deepseek-ai/DeepSeek-R1/blob/main/DeepSeek_R1.pdf



混合专家 MOE

直接硬件编程 PTX

通讯优化 DualPipe

多头潜在注意力 MLA

混合精度训练 FP8

并行训练框架 HAI

强化学习 GRPO

多Token预测 MTP

测试时计算 TTC

DeepSeek模型特点总结

北京大学 PEKING UNIVERSITY

- **基础架构：**
  - **混合专家模型（MoE）**：DeepSeek采用MoE架构，通过动态选择最适合输入数据的专家模块进行处理，提升推理能力和效率。
  - **无辅助损失的负载均衡策略（EP）**：该策略使DeepSeekMoE在不对优化目标产生干扰的前提下，实现各个专家的负载均衡，避免了某些专家可能会被过度使用，而其他专家则被闲置的现象。
  - **多头潜在注意力机制（MLA）**：MLA通过低秩压缩减少Key-Value缓存，显著提升推理效率。
  - **强化学习驱动（RL）**：DeepSeek-R1在训练中大规模应用强化学习，将传统的PPO替换为GRPO训练算法，显著提升推理能力。
  - **多Token预测（MTP）**：通过多Token预测，Deepseek不仅提高了推理速度，还降低了训练成本。

DeepSeek模型特点总结

- **训练及框架：**
  - **FP8混合精度训练**：在关键计算步骤使用高精度，其他模型层使用FP8低精度进一步降低训练成本。这一点，是DeepSeek团队在Infra工程上的非常有价值的突破。
  - **长链推理技术（TTC）**：模型支持数万字的长链推理，可逐步分解复杂问题并进行多步骤逻辑推理。
  - **并行训练策略（HAI）：** 16 路流水线并行(Pipeline Parallelism, PP)、跨 8 个节点的 64 路专家并行(Expert Parallelism, EP)，以及数据并行(Data Parallelism, DP)，大幅提升模型训练速度。
  - **通讯优化DualPipe**： 高效的跨节点通信内核，利用 IB 和 NVLink 带宽，减少通信开销，提高模型推理性能。
  - **混合机器编程（PTX）**：部分代码直接进行使用PTX编程提高执行效率，并优化了一部分算子库。
  - **低成本训练**：DeepSeek-V3的训练成本为557.6万美元，仅为OpenAI的GPT-4o等领先闭源模型的3%-5%。

DeepSeek模型特点总结

- **社会价值：**
  - **开源生态**：DeepSeek采用开源策略，使用最为开放的MIT开源协议，吸引了大量开发者和研究人员，推动了AI技术的发展。
  - **模型蒸馏支持**：DeepSeek-R1同时发布了多个模型蒸馏。虽然这些蒸馏模型的生产初衷是为了验证蒸馏效果，但在实质上帮助用户可以训练更小型的模型以满足不同应用场景需求，也给用户提供更多的抑制了DeepSeek R1满血版模型的能力的小模型选择（虽然也给市场和用户造成了很多困扰）。
  - **AI产品和技术的普及教育：**对于社会，认识到AI是一个趋势，不是昙花一现；对于市场，用户开始主动引入AI，不用教育了；对于大模型企业， 越发开始重视infra工程的价值了。

DeepSeek-R1全家桶

北京大学 PEKING UNIVERSITY

Model	Base Model	Model Download		
DeepSeek-R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	🤗 HuggingFace	ModelScope	蒸馏模型，能力稍弱 实际上是增加了推理能力的Qwen模型和Llama模型，严格来讲不能称为DeepSeek模型（市场上有误解，厂商有误导，Ollama工具的模型选项中也有误导）
DeepSeek-R1-Distill-Qwen-7B	Qwen2.5-Math-7B	🤗 HuggingFace	ModelScope	
DeepSeek-R1-Distill-Llama-8B	Llama-3.1-8B	🤗 HuggingFace	ModelScope	
DeepSeek-R1-Distill-Qwen-14B	Qwen2.5-14B	🤗 HuggingFace	ModelScope	
DeepSeek-R1-Distill-Qwen-32B	Qwen2.5-32B	🤗 HuggingFace	ModelScope	
DeepSeek-R1-Distill-Llama-70B	Llama-3.3-70B-Instruct	🤗 HuggingFace	ModelScope	
DeepSeek-R1-671B	DeepSeek-V3-Base	🤗 HuggingFace	ModelScope	满血版，能力最强

蒸馏模型-概念

北京大学 PEKING UNIVERSITY

模型蒸馏的定义

- 通俗解释：模型蒸馏就像是让一个“老师”（大模型）把知识传授给一个“学生”（小模型），让“学生”变成“学霸”。
- 正式定义：模型蒸馏是一种将大型复杂模型（教师模型）的知识迁移到小型高效模型（学生模型）的技术。

模型蒸馏的原理

- 教师模型的训练：先训练一个性能强大但计算成本高的教师模型。
- 生成软标签：教师模型对数据进行预测，得到每个样本的概率分布，这些就是软标签。
- 训练学生模型：用软标签和硬标签共同训练学生模型。
- 优化与调整：通过调整超参数，优化学生模型的性能。

蒸馏技术的优势

- 模型压缩：学生模型参数少，计算成本低，更适合在资源受限的环境中部署。
- 性能提升：学生模型通过学习教师模型的输出概率分布，能够更好地理解数据的模式和特征。
- 效率提高：学生模型训练所需的样本数量可能更少，训练成本降低。

DeepSeek-R1蒸馏模型-能力对比

北京大学 PEKING UNIVERSITY

Model	AIME 2024 pass@1	AIME 2024 cons@64	MATH-500 pass@1	GPQA Diamond pass@1	LiveCodeBench pass@1	CodeForces rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
o1-mini	63.6	80.0	90.0	60.0	53.8	**1820**
QwQ-32B-Preview	44.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	1633

DeepSeek蒸馏版的选择经验

- 千万别用1.5B和8B做正经业务，会翻车！
- 做自然语言对话7B就很好用了
- 预算有限又想搞事情就选14B
- 要做知识问答选32B，对代码支持也不错
- 70B性价比最低，与32B性能类似，成本翻倍，没有什么理由存在

DeepSeek模型应用场景

边缘计算场景的低延迟部署

DeepSeek结合天翼云智能边缘云ECX，能够在靠近用户的边缘节点部署模型，显著降低数据传输延迟，适用于对实时性要求极高的场景

复杂数学与编程任务

DeepSeek-R1在数学推理和代码生成领域展现了超越同类模型的独特能力

中文场景的深度优化

理解文化背景和习惯用语（如生成春节习俗对比文章），优于GPT-4的中文处理能力

使用方式

直接使用官方服务

- 访问官网(ai.com 或 https://chat.deepseek.com/)，登录后使用，适合电脑端快速使用，但存在服务不稳定问题。
- 下载APP，适合手机平板等移动设备，但同样存在服务不稳定问题。

使用第三方服务与API调用

- 第三方服务：秘塔AI、微信搜索、Molly R1、问小白等。
- API调用：DeepSeek、硅基流动、火山引擎等。获取API密钥调用，适合开发者集成，同样存在服务不稳定问题。

本地部署

- 个人部署：个人在本地设备运行应用，依赖自身计算资源，灵活便捷。
- 企业部署：企业内网搭建私有化系统，支持多用户协作，数据可控。
- 一体机：直接购买配置了DeepSeek模型的具备一定算力的一体机。

北京大学
PEKING UNIVERSITY

个人部署DeepSeek

Personal deployment

PART 02

对于个人玩家如何在自己的电脑上部署和体验DeepSeek（一般是蒸馏版），我们会详细分享具体的模型评估数据和软硬件要求，通过实操环节来详细讲解Ollama命令行高效部署全流程，并构建多形态用户接入方案，包括浏览器插件PageAssist、桌面端Chatbox和团队协作型OpenWebUI，实现从个人到企业的全场景覆盖。其中分享过程中常见问题和经验，帮助大家可完整体验、并成功实现本地化部署DeepSeek。

01

02

03

环境准备

明确模型部署的软硬件要求，分析参数量对推理效果和硬件配置的影响。提供最低与推荐配置清单，详解Ollama框架优势及环境变量配置，建立标准化部署环境。

模型部署

前端展示

模型参数量意义

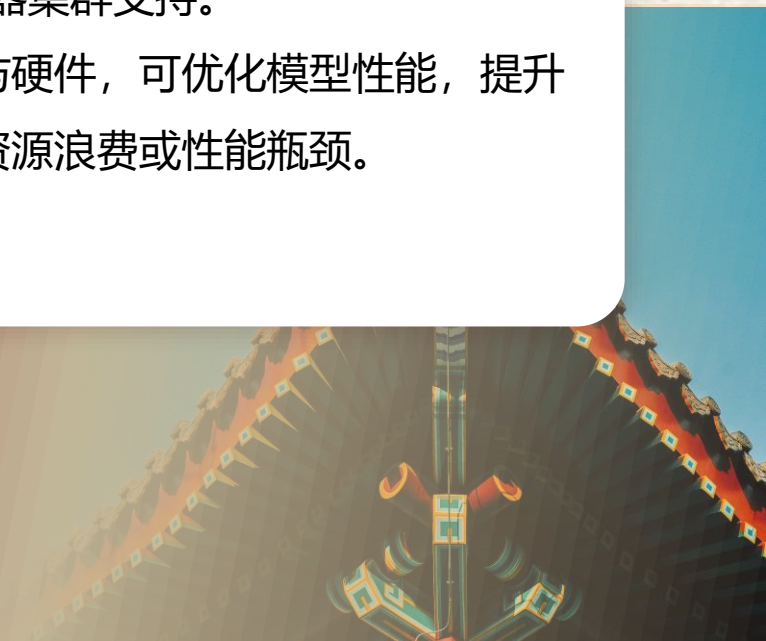
北京大学 PEKING UNIVERSITY

参数量影响模型能力

1. 模型能力：通常来说，参数量越大，模型就有更强的理解和生成能力，但是需要更多计算资源。
2. 硬件需求：参数越多，对内存（RAM）和显存（VRAM）的需求就越高。
3. 运行速度：参数量大的模型，推理速度更慢，尤其是资源不足的时候。

参数量与硬件配置匹配

1. 本地部署DeepSeek时，需根据硬件配置选择合适模型版本。如1.5B模型适合资源受限设备，671B模型需服务器集群支持。
2. 合理匹配参数量与硬件，可优化模型性能，提升运行效率，避免资源浪费或性能瓶颈。



硬件配置-推荐

北京大学 PEKING UNIVERSITY

模型参数	CPU要求	内存要求	显存要求（GPU）	硬盘空间	适用场景
1.5B	6核（现代多核）	16GB	4GB（如:GTX 1650）	5GB+	实时聊天机器人、物联网设备
7B	8核（现代多核）	32GB	8GB（如:RTX 3070）	10GB+	文本摘要、多轮对话系统
8B	10核（多线程）	32GB	10GB	12GB+	高精度轻量级任务
14B	12核	64GB	16GB（如:RTX 4090）	20GB+	合同分析、论文辅助写作
32B	16核（如i9/Ryzen 9）	128GB	24GB（如:RTX 4090）	30GB+	法律/医疗咨询、多模态预处理
70B	32核（服务器级）	256GB	40GB（如:双A100）	100GB+	金融预测、大规模数据分析
671B	64核（服务器集群）	512GB	160GB（8x A100）	500GB+	国家级AI研究、气候建模

Ollama中提供的deepseek-r1模型均为4位量化模型, 所需资源较正常少一些, 如果要运行非量化模型, 则需要更大的显存（比如7B需要至少16G显存）

硬件配置-最低

北京大学 PEKING UNIVERSITY

模型参数	CPU要求	内存要求	显存要求（GPU）	硬盘空间	适用场景
1.5B	4核（Intel/AMD）	8GB	无（纯CPU）或2GB（GPU加速）	3GB+	树莓派、旧款笔记本、简单文本生成
7B	4核（多线程支持）	16GB	4GB	8GB+	本地开发测试、轻量级NLP任务
8B	6核（多线程）	16GB	6GB	8GB+	代码生成、逻辑推理
14B	8核	32GB	8GB	15GB+	企业级文本分析、长文本生成
32B	12核	48GB	16GB	19GB+	复杂场景对话、深度思考任务
70B	16核（服务器级）	64GB	24GB（多卡）	70GB+	创意写作、算法设计
671B	32核（服务器集群）	128GB	80GB（多卡）	300GB+	科研级任务、AGI探索

上页推荐配置中, 是较为流畅的运行模型, 而最低配置是指可以运行模型, 但流畅度会稍差一些

硬件选择建议

01

根据需求选择

- 若仅需简单任务，如实时聊天或轻量级文本生成，可选择1.5B或7B模型，搭配较低配置硬件，如普通笔记本或台式机。
- 对于复杂任务，如合同分析、论文写作或大规模数据分析，需选择14B及以上模型，并配备高性能硬件，如高端显卡和大容量内存。

02

考虑预算与性能平衡

- 在预算有限的情况下，可优先选择较低参数量的模型，以满足基本需求，同时降低硬件成本。如1.5B模型可在资源受限设备上运行，适合预算紧张的用户。
- 若预算充足且对性能要求较高，可选择高参数量模型，如32B或70B，搭配高端硬件，以获得更强的处理能力和更高的运行效率。

03

硬件升级与扩展

- 随着任务需求的增加和预算的提升，可逐步升级硬件配置，如增加内存、更换高性能显卡或升级CPU。
- 对于企业用户或科研机构，可根据实际需求构建服务器集群，以支持大规模模型的运行和复杂任务的处理。

Ollama工具介绍

北京大学 PEKING UNIVERSITY

Ollama简介

- Ollama是一个开源的大型语言模型服务工具，旨在帮助用户快速在本地运行大模型。通过简单的安装指令，用户可以在消费级PC上体验LLM的强大功能。
- Ollama会自动监测本地计算资源，优先使用GPU资源以提升推理速度，若无GPU则直接使用CPU资源。
- Ollama官方链接: https://ollama.com/

Ollama功能特点

- 开源免费
- 简单易用
- 模型丰富
- 功能齐全
- 隐私保护
- 社区活跃
- 支持多平台
- 支持工具调用
- 资源占用低

软件安装步骤

北京大学 PEKING UNIVERSITY

安装Docker（可选）

- Windows/Mac系统：访问Docker官网，下载Docker Desktop安装程序，运行安装并启动Docker Desktop。
- Linux系统：访问Docker官网，根据Linux发行版选择安装方式，如基于Debian系统使用sudo apt-get install docker-ce docker-ce-cli containerd.io命令安装，安装后启动Docker服务。

安装Ollama客户端

- Windows/Mac系统：访问Ollama官网或GitHub页面，下载安装包并运行安装程序。
- Linux系统：一键安装命令**curl -fsSL https://ollama.com/install.sh | sh**，或手动下载并解压安装。
- Docker安装：拉取Ollama镜像，如CPU版使用**docker pull ollama/ollama**，运行镜像时可使用**docker run -d -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama**命令。

环境变量配置

北京大学 PEKING UNIVERSITY

参数	标识与配置
OLLAMA_MODELS	表示模型文件的存放目录，默认目录为当前用户目录即 C:\Users%username%.ollama\models Windows 系统 建议不要放在C盘，可放在其他盘（如 E:\ollama\models）
OLLAMA_HOST	表示ollama 服务监听的网络地址，默认为127.0.0.1 如果想要允许其他电脑访问 Ollama（如局域网中的其他电脑），建议设置成 0.0.0.0
OLLAMA_PORT	表示ollama 服务监听的默认端口，默认为11434 如果端口有冲突，可以修改设置成其他端口（如8080等）
OLLAMA_ORIGINS	表示HTTP 客户端的请求来源，使用半角逗号分隔列表 如果本地使用不受限制，可以设置成星号 *
OLLAMA_KEEP_ALIVE	表示大模型加载到内存中后的存活时间，默认为5m即 5 分钟 （如纯数字300 代表 300 秒，0 代表处理请求响应后立即卸载模型，任何负数则表示一直存活） 建议设置成 24h ，即模型在内存中保持 24 小时，提高访问速度
OLLAMA_NUM_PARALLEL	表示请求处理的并发数量，默认为1 （即单并发串行处理请求） 建议按照实际需求进行调整
OLLAMA_MAX_QUEUE	表示请求队列长度，默认值为512 建议按照实际需求进行调整，超过队列长度的请求会被抛弃
OLLAMA_DEBUG	表示输出 Debug 日志，应用研发阶段可以设置成1 （即输出详细日志信息，便于排查问题）
OLLAMA_MAX_LOADED_MODELS	表示最多同时加载到内存中模型的数量，默认为1 （即只能有 1 个模型在内存中）

建议修改：
将模型路径迁移至非系统盘（如 OLLAMA_MODELS=E:\models），按需提高 OLLAMA_KEEP_ALIVE 和 OLLAMA_NUM_PARALLEL 优化性能，生产环境限制请求来源并关闭调试日志。

网络配置

访问Ollama服务

- 启动Ollama服务后，通过浏览器访问 **http://localhost:11434**
  可看到：“ ollama is running “
- 若无法访问，需检查Ollama服务是否启动，是否有防火墙或安全软件阻止端口访问，以及是否更改了默认端口。

前端界面网络访问

- 本地部署：前端界面运行在本地计算机上，可通过 **http://localhost:11434**与Ollama服务通信。
- 远程部署：需确保前端界面所在计算机可访问Ollama服务所在计算机的IP地址和端口。可在Ollama服务所在计算机上运行**ollama serve --host 0.0.0.0**命令允许远程访问或者修改环境变量，最后通过IP地址访问Ollama服务。

网络配置注意事项

- 端口冲突：确保11434端口未被占用，若被占用，可通过：
  ollama serve --port <new_port>命令或者通过修改环境变量指定其他端口。
- 代理设置：在代理网络环境中，需确保代理设置允许访问本地服务。
- 防火墙设置：确保防火墙规则允许对11434端口的访问，可添加防火墙规则允许该端口流量。

01

环境准备

02

模型部署

演示Ollama完整部署流程：从DeepSeek 7B模型选择到加载，详细讲解命令行指令集，重点说明模型加载与启动命令的正确使用方法。

03

前端展示

Ollama常用命令

北京大学 PEKING UNIVERSITY

命令	描述
ollama serve	启动 Ollama
ollama create	从 Modelfile 创建模型
ollama show	显示模型信息
ollama run	运行模型
ollama pull	从注册表中拉取模型
ollama push	将模型推送到注册表
ollama list	列出所有模型
ollama ps	列出正在运行的模型
ollama cp	复制模型
ollama rm	删除模型
ollama help	显示任意命令的帮助信息

这两个命令是个人部署最主要的两个命令，主要用来下载和启动模型

使用Ollama命令下载模型

北京大学 PEKING UNIVERSITY

下载命令格式

- 使用**ollama pull**命令下载模型，格式为**ollama pull <model_name>**。
- 案例：某用户在终端输入**ollama pull deepseek- r1:7b**，从Ollama模型库中成功下载7B模型，耗时约10分钟。

下载过程监控

- 下载过程中，终端会显示下载进度条，实时显示下载速度和剩余时间。
- 可通过**ollama list**命令查看已下载的模型列表，确认模型是否下载完成。
- 案例：某用户在下载14B模型时，通过**ollama list**命令发现下载进度为50%，预计还需20分钟完成。

下载失败处理

- 若下载失败，可能是网络问题或模型库地址错误。
- 可尝试重新运行下载命令，或检查网络连接和模型库地址。
- 案例：某用户下载7B模型时因网络中断失败，重新运行下载命令后成功完成下载。

使用Ollama命令运行模型

北京大学 PEKING UNIVERSITY

运行命令格式

- 使用**ollama run**命令运行模型，格式为**ollama run <model_name>**。
- 案例：某用户在终端输入**ollama run deepseek-r1:7b**，模型成功启动并进入运行状态。

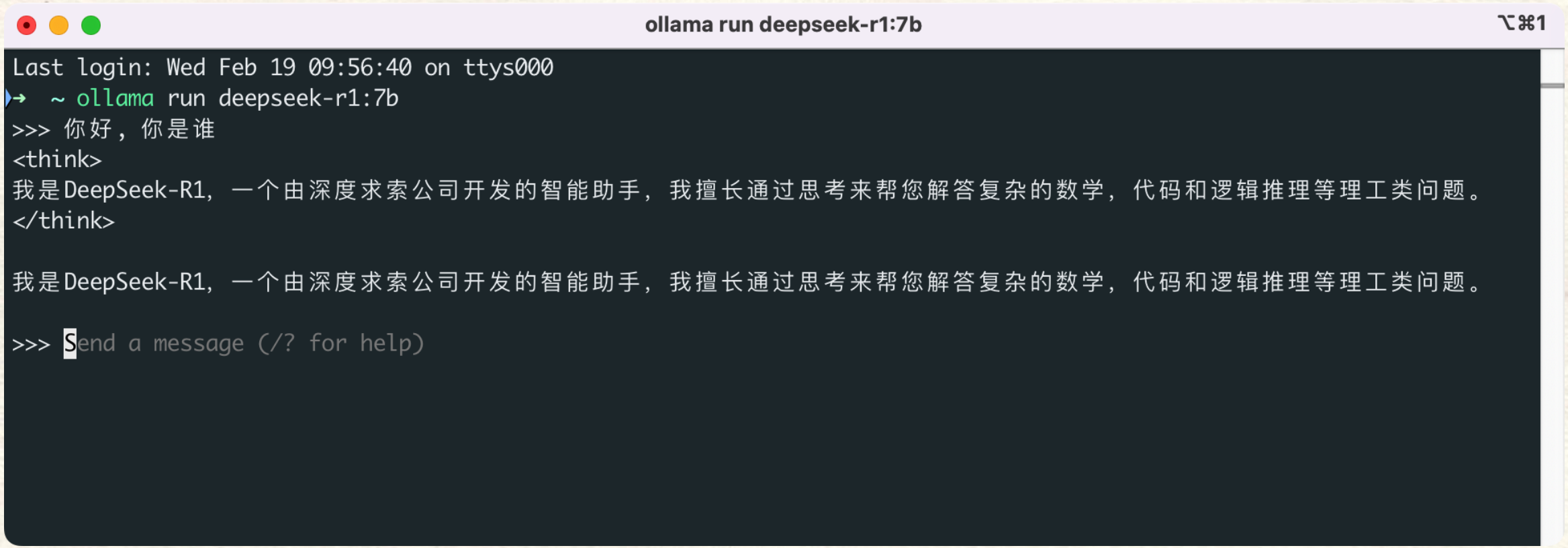
运行状态检查

- 模型运行后，可通过**ollama ps**命令查看正在运行的模型列表，确认模型是否正常运行。
- 若模型未正常运行，可检查硬件资源是否充足，或是否有其他模型占用资源。
- 案例：某用户运行14B模型后，响应速度较慢，经检查发现内存占用过高。

停止运行模型

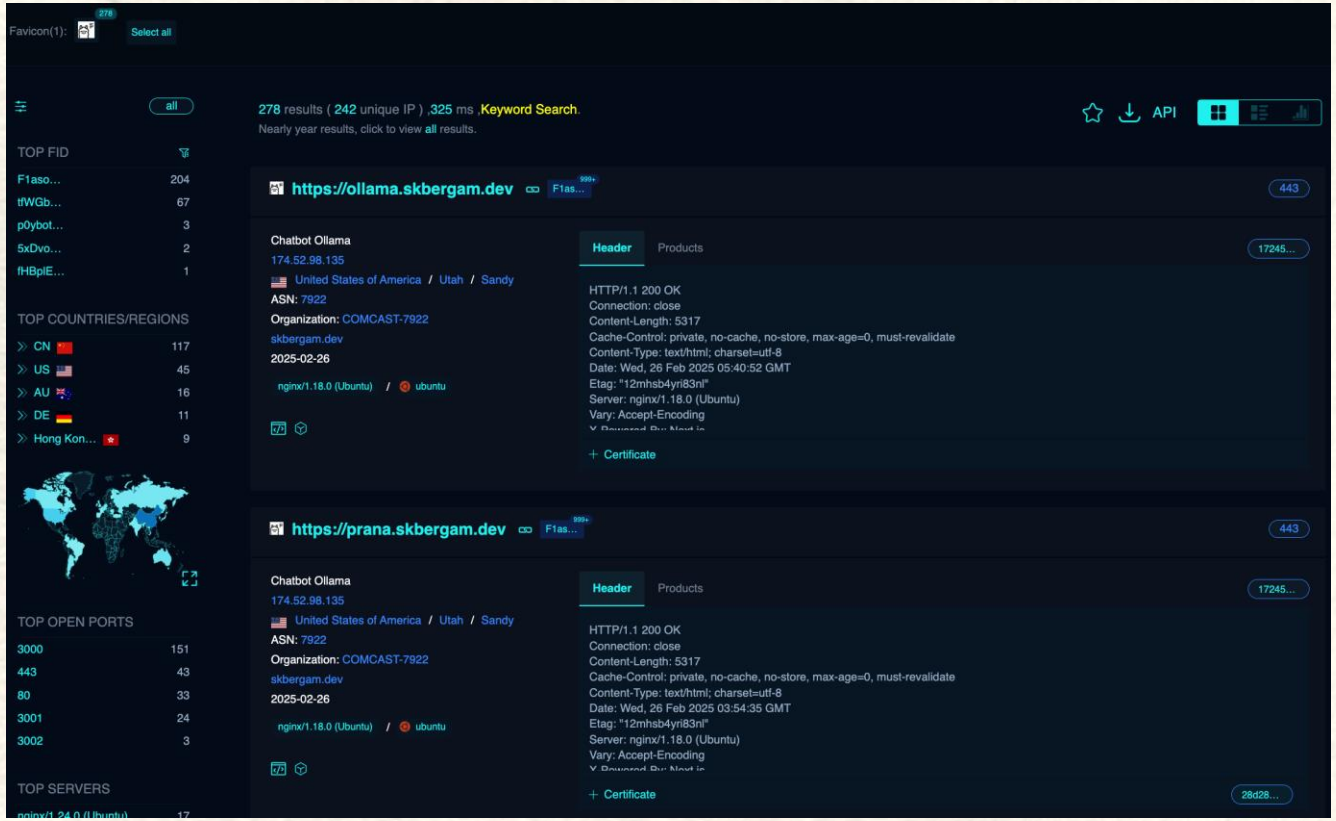
- 使用**ollama stop**命令停止运行的模型，格式为**ollama stop <model_name>**。
- 案例：某用户在测试完成后，通过**ollama stop deepseek-r1:7b**命令停止了7B模型的运行，释放了系统资源。

模型运行效果


ollama run deepseek-r1:7b
⌥⌘1
Last login: Wed Feb 19 09:56:40 on ttys000
➜ ~ ollama run deepseek-r1:7b
>>> 你好，你是谁
<think>
我是DeepSeek-R1，一个由深度求索公司开发的智能助手，我擅长通过思考来帮您解答复杂的数学，代码和逻辑推理等理工类问题。
</think>
我是DeepSeek-R1，一个由深度求索公司开发的智能助手，我擅长通过思考来帮您解答复杂的数学，代码和逻辑推理等理工类问题。
>>> Send a message (/? for help)

Ollama常见问题

安全漏洞与未经授权访问风险



- **问题描述**
  - 默认配置下Ollama服务暴露在公网，易被自动化脚本扫描攻击，导致算力盗用或模型文件删除。
- **解决方案**
  - 强制身份认证：通过反向代理（如Nginx）添加Basic Auth认证，或在Ollama启动时绑定内网IP（OLLAMA_HOST=192.168.x.x）。
  - 网络层隔离：使用Docker的--network host模式部署，配合iptables限制访问来源IP段。
  - 日志监控：启用Ollama的--verbose日志模式，结合Prometheus监控API调用频率，识别异常流量 。

Ollama常见问题

模型升级后性能退化问题

- **问题描述**
  - Ollama版本升级后模型仅运行在CPU，或量化精度丢失导致回答质量下降。
- **解决方案**
  - 锁定依赖版本：通过Docker镜像固定Ollama版本（如ollama/ollama:0.5.1-cuda），避免自动升级引入兼容性问题。
  - 显存分配验证：使用nvidia-smi监控GPU利用率，若发现异常回退至CPU，检查CUDA驱动版本与Ollama编译环境的兼容性。

01

环境准备

02

模型部署

03

前端展示

构建多形态接入方案：PageAssist 实现网页即时问答，Chatbox支持 Markdown渲染，Open WebUI 提供企业权限管理。

浏览器插件

Orian (Ollama WebUI)

- 特点：多功能聊天系统，支持Gmail集成、谷歌搜索增强、上下文网站交互。
- 优点：功能丰富，AI交互体验全面，适用于多种场景。
- 缺点：部分功能（如实时搜索）可能处于维护状态，对不熟悉Gmail和谷歌搜索的用户实用性降低。
- 适用场景：适合需要邮件处理、搜索增强功能的用户。
-

ollama-ui

- 特点：实时对话交互，多模型切换，聊天历史管理，网页内容交互，支持多种文档格式。
- 优点：功能实用，隐私保护，所有交互在本地完成。
- 缺点：功能较为基础，可能无法满足复杂需求。
- 适用场景：适合需要与网页内容交互、管理聊天历史的用户。
-

Page Assist

- 特点：浏览器插件，支持PDF对话、网络搜索集成，与对象存储、Cloud Studio无缝集成。
- 优点：轻量级，安装方便，与网页内容紧密结合。
- 缺点：功能相对基础，对于不常使用网页交互的用户优势不明显。
- 适用场景：适合需要与网页内容紧密结合、进行针对性AI交互的用户。
-

桌面应用

- **Enchanted LLM**
  - 特点：MacOS原生应用，支持多种私人托管模型，界面简洁直观。
  - 适用场景：适合MacOS用户，需要本地运行多种模型。
- **Chatbox**
  - 特点：跨平台开源客户端应用，支持Windows、MacOS、Linux、iOS和Android，支持多种大语言模型。
  - 适用场景：适合跨平台使用，需要多种模型支持的用户。
- **LLocal.in**
  - 特点：跨平台完全开源的客户端，用于利用本地运行的大型语言模型，Electron桌面客户端，易于使用。
  - 适用场景：适合需要本地运行模型、跨平台使用的用户。
- Ollama App
  - 特点：现代且易于使用的多平台客户端，支持Ollama。
  - 适用场景：适合需要简洁界面、多平台支持的用户。

Web服务

Open WebUI

- 特点：基于Web的界面，方便用户与Ollama模型进行交互。
- 适用场景：适合需要通过Web界面与模型交互的用户。
- 项目地址：https://github.com/open-webui/open-webui

NextJS Ollama LLM UI

- 特点：专为Ollama设计的极简主义用户界面，界面美观。
- 适用场景：适合追求简约风格的用户。
- 项目地址：https://github.com/jakobhoeg/nextjs-ollama-llm-ui

Ollama Basic Chat

- 特点：使用HyperDiv反应式UI的Ollama基本聊天界面。
- 适用场景：适合需要简单聊天界面的用户。

其他

北京大学 PEKING UNIVERSITY

01

终端工具

- 特点：Ollama提供了多种终端工具，如oterm、ollama.nvim等，方便开发者在终端中使用Ollama。
- 适用场景：适合开发者，需要在终端中快速使用Ollama。

02

云服务

- 特点：Ollama支持在Google Cloud、Fly.io、Koyeb等云平台上部署。
- 适用场景：适合需要在云端部署模型的用户。

Chatbox-安装

北京大学 PEKING UNIVERSITY

下载与安装

- 访问Chatbox AI官网，根据设备选择版本下载并安装，安装完成后启动应用。
- 适用场景：适合跨平台使用，需要多种模型支持的用户。

使用功能

- 与文档和图片聊天，代码生成与预览，实时联网搜索与查询，AI生成的图表可视化，AI驱动的图像生成，LaTeX和Markdown渲染

配置模型

- 使用本地模型：打开Chatbox软件，点击左下角齿轮图标进入设置，选择「模型提供方」为「Ollama」，「API域名」自动填充为**http://localhost:11434**，选择对应模型，点击「保存」。
- 使用在线API：以接入硅基流动的DeepSeek- R1为例，选择「SiliconFlow API」，粘贴API密钥，选择DeepSeek- R1模型。
- 适用场景：适合需要快速切换本地和在线模型的用户。

其他设置

- 设置API代理：在设置菜单中，配置API代理，连接到不同的AI模型和服务提供者。
- 管理和调试Prompt：使用Chatbox提供的工具设计和调整Prompt，以获得最佳的AI交互体验。
- 保存聊天记录：所有聊天记录和Prompt会自动保存在本地。
- 多平台同步：通过登录账号，可在不同设备上同步数据。
- 适用场景：适合需要多平台使用、管理聊天记录的用户。

Chatbox-界面

温度设置建议:
代码生成/数学解题 -> 0.0
数据抽取/分析 -> 1.0
通用对话 -> 1.3
翻译 -> 1.3
创意类写作/诗歌创作 -> 1.5

场景	温度
代码生成/数学解题	0.0
数据抽取/分析	1.0
通用对话	1.3
翻译	1.3
创意类写作/诗歌创作	1.5

Page Assist-安装

01

安装方法

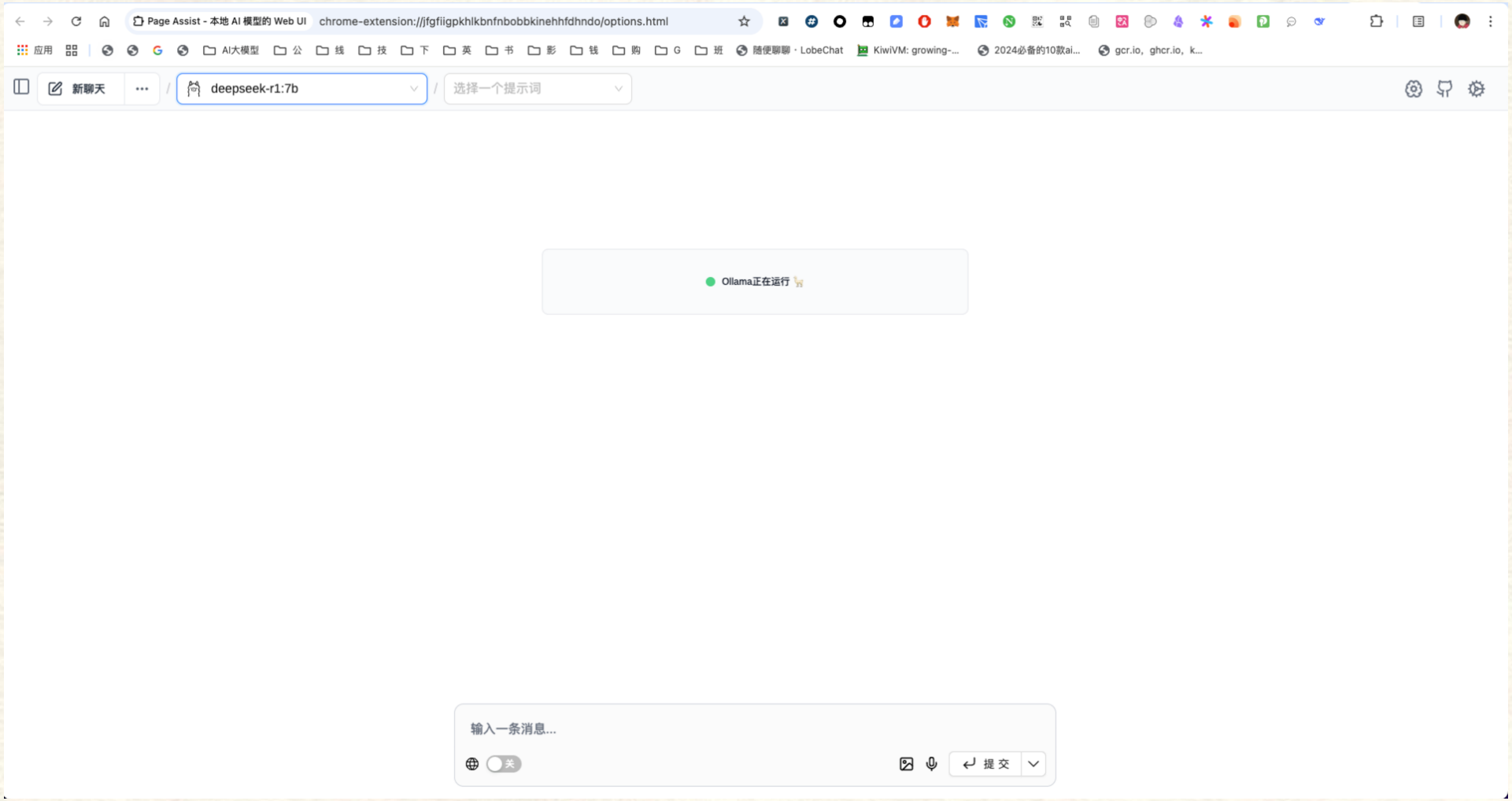
- 打开Chrome Web Store
- 搜索“Page Assist”
- 点击“添加到Chrome”按钮

02

使用方法

- 打开侧边栏：安装完成后，通过右键菜单或快捷键（默认为Ctrl+Shift+P）打开侧边栏。在侧边栏中，可以与本地AI模型进行对话，获取网页内容的相关信息。
- 使用Web UI：点击扩展图标，会在新标签页中打开Web UI。在Web UI中，可以像使用ChatGPT一样与AI模型进行对话。
- 配置本地AI模型：首次使用时，需要配置本地AI模型，目前支持Ollama和Chrome AI (Gemini Nano)等本地AI提供商。选择指定的模型后，即可开始与模型进行交互。
- 其他功能：网页内容对话、文档解析、联网搜索、语言设置。
- 适用场景：适合需要与网页内容紧密结合、进行针对性AI交互的用户。

Page Assist-界面



OpenWebui-安装

北京大学 PEKING UNIVERSITY

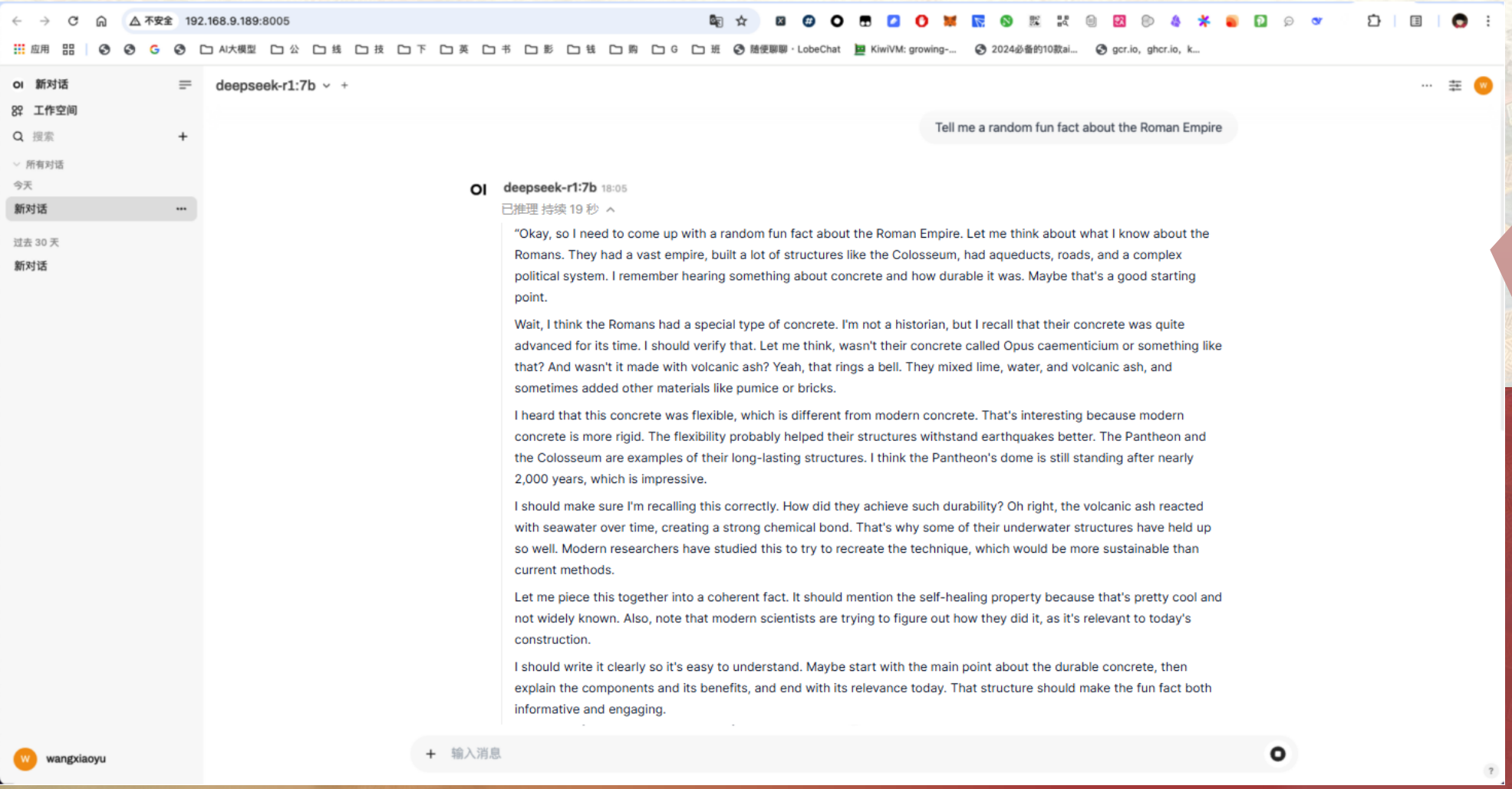
安装方法

- 运行Open WebUI，使用以下命令：
- 无 GPU 加速：**docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:main**
- 使用 GPU 加速：**docker run -d -p 3000:8080 --gpus all --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:cuda**

使用方法

- 访问Open WebUI：安装完成后，打开浏览器访问http://localhost:3000。
- 配置AI模型：选择支持的LLM运行器（如Ollama或OpenAI API），可以配置自定义OpenAI API URL，连接到其他兼容的API服务。

OpenWebui-界面



OpenWebUI 支持多模态交互、本地模型与云端 API 混合部署，并深度集成 RAG 知识库和外部工具，相较于 Page Assist 等单一功能插件，其开源可定制、支持企业级多用户管理及全栈 AI 工作流扩展的特性，更适合开发者构建私有化复杂应用。

三款界面总结

- **PageAssist**以浏览器插件形态实现本地AI与浏览场景的无缝融合，强调隐私优先和轻量交互，通过侧边栏对话、文档解析等能力将模型能力嵌入用户日常操作，适合注重数据安全的高频轻需求场景。
- **Chatbox**定位灵活的中台调度工具，以模块化设计兼容多模型API和插件扩展，平衡开发者的自定义需求与普通用户的易用性，适用于需要多模型协同或快速验证AI能力的场景。
- **Open WebUI**聚焦企业级AI服务全生命周期管理，从模型部署、权限控制到知识库集成提供闭环方案，通过负载均衡、协作聊天等特性满足规模化团队的技术管控需求。

PART 03

企业部署DeepSeek

Enterprise deployment

1. Ollma框架适合个人用户私有化本地部署，但在多用户并发场景下性能衰减明显。这一部分我们将尽可能简单地介绍企业级私有化部署的方案和探索实践，普通用户可以了解即可。
2. 企业级生产环境推荐使用Transformers来快速验证模型能力，使用vLLM框架借助PagedAttention技术实现24倍于Transformers的吞吐量实现大模型的高效推理，针对不同企业场景，则提供不同的企业级部署方案，我们也会分享服务器配置、性能数据及报价参考等实战经验，且深度分析业务场景的适配性，给予参考帮助。
3. 同时，针对近期出现的KTransformers、Unsloth等多套低成本动态量化模型的DeepSeek部署解决方案。虽然不够成熟也无法投入实际生产使用，但我们也会用一定的篇幅分享我们的研究实践和经验。

Transformers简介

- Transformers 提供了可以轻松地下载并且训练先进的预训练模型的 API 和工具。使用预训练模型可以减少计算消耗和碳排放，并且节省从头训练所需要的时间和资源。这些模型支持不同模态中的常见任务，比如：
  - 自然语言处理：文本分类、命名实体识别、问答、语言建模、摘要、翻译、多项选择和文本生成。
  - 机器视觉：图像分类、目标检测和语义分割。
  - 音频：自动语音识别和音频分类。
  - 多模态：表格问答、光学字符识别、从扫描文档提取信息、视频分类和视觉问答。

- Transformers 支持在 PyTorch、TensorFlow 和 JAX 上的互操作性。这给在模型的每个阶段使用不同的框架带来了灵活性；在一个框架中使用几行代码训练一个模型，然后在另一个框架中加载它并进行推理。模型也可以被导出为 ONNX 和 TorchScript 格式，用于在生产环境中部署。

Transformers部署模型 3-1

北京大学 PEKING UNIVERSITY

- 步骤一：安装相关依赖包

```
pip install torch
pip install transformers accelerate bitsandbytes
```

- 步骤二：加载模型

```
from transformers import AutoModelForCausalLM, AutoTokenizer
model_path = "deepseek-ai/deepseek-r1-distill-qwen-7b"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModelForCausalLM.from_pretrained(
    model_path,
    device_map="auto",
    torch_dtype="auto"
)
```

load_in_8bit=True, # 可以添加参数来启用8bit量化

Transformers部署模型 3-2

北京大学 PEKING UNIVERSITY

- 步骤三：运行模型

```
prompt = "<|system|>你是一个人工智能助手<|user|>解释量子计算<|assistant|>"
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
outputs = model.generate(
    **inputs,
    max_new_tokens=256,
    do_sample=True,
    temperature=0.7,
    repetition_penalty=1.05
)
print(tokenizer.decode(outputs[0]))
```

```
from transformers import AutoModelForCausalLM, AutoTokenizer
model_path = "deepseek-ai/deepseek-r1-distill-qwen-7b"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModelForCausalLM.from_pretrained(
    model_path,
    device_map="auto",
    torch_dtype="auto"
)
prompt = "<|system|>你是一个人工智能助手<|user|>解释量子计算<|assistant|>"
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
outputs = model.generate(
    **inputs,
    max_new_tokens=256,
    do_sample=True,
    temperature=0.7,
    repetition_penalty=1.05
)
print(tokenizer.decode(outputs[0]))
```

```
<｜begin_of_sentence｜><|system|>你是一个人工智能助手<|user|>解释量子计算<|assistant|>量子计算是一种利用量
子力学现象来进行信息处理的计算方式。
它基于量子位（qubit）的性质，能够进行并行计算和量子叠加。
量子位不仅可以在0和1之间变化，还能处于两者之间的叠加态，
同时，在特定条件下，可以实现量子的相干性。

量子计算的原理包括叠加原理和量子纠缠。
叠加原理使得一个量子系统可以同时处于多个状态，
而量子纠缠则允许不同位置的量子系统之间产生关联，
从而增强计算能力。

量子计算机使用量子位来构建算法，
通过特定的量子逻辑门进行操作，
最终读取结果。

量子计算在某些领域具有显著优势，
比如密码学、材料科学、优化问题等。
```

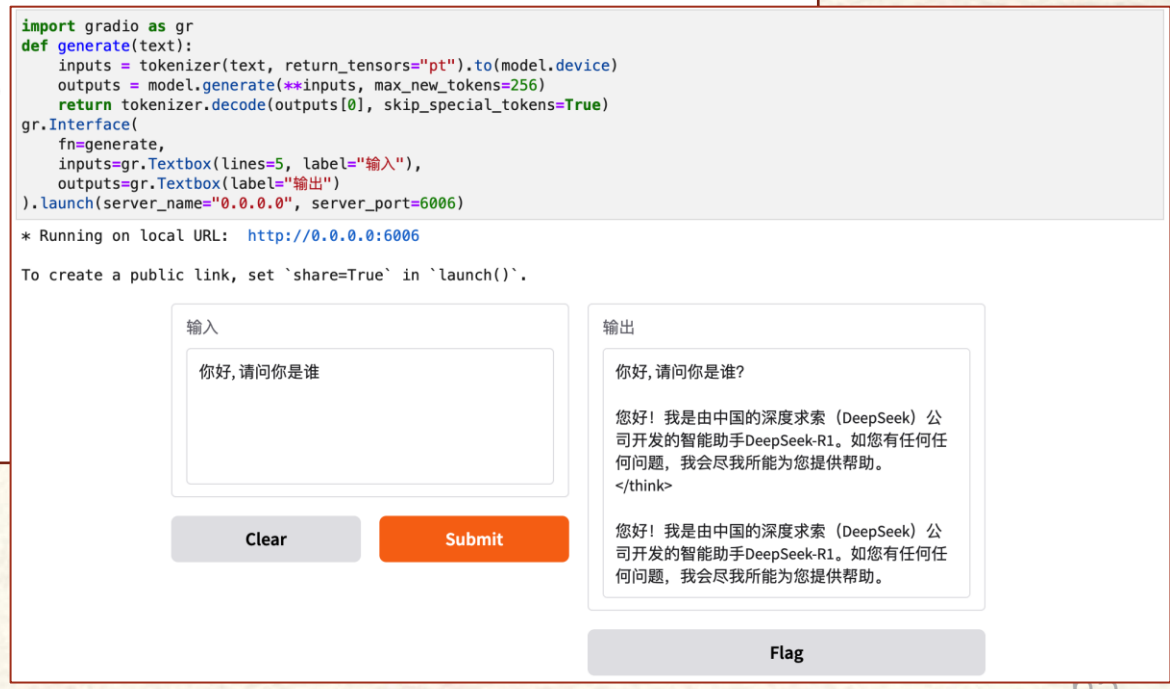
Transformers部署模型 3-3

北京大学 PEKING UNIVERSITY

- 步骤四：其他框架调用

这里选择了gradio框架，可以快速启动一个前端界面，实战中可以选择配合业务逻辑进行调用

```
# pip install gradio
import gradio as gr
def generate(text):
    inputs = tokenizer(text, return_tensors="pt").to(model.device)
    outputs = model.generate(**inputs, max_new_tokens=256)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)
gr.Interface(
    fn=generate,
    inputs=gr.Textbox(lines=5, label="输入"),
    outputs=gr.Textbox(label="输出")
).launch(server_name="0.0.0.0", server_port=6006)
```



生成时常用参数 2-1

参数名称	含义	注意事项
inputs	输入的文本或编码后的 input_ids，用于生成文本的起始部分	如果传入 inputs_embeds，则不能同时传入 inputs
input_ids	与 inputs 类似，但通常用于直接传入编码后的输入	如果传入 inputs_embeds，则不能同时传入 input_ids
inputs_embeds	输入的嵌入向量，通常用于 encoder-decoder 模型	不能同时传入 input_ids 或 inputs
max_length	生成文本的最大长度（包括输入部分）	如果同时设置 max_new_tokens，max_new_tokens 优先
max_new_tokens	生成的新 token 数量（不包括输入部分）	如果同时设置 max_length，max_new_tokens 优先
min_length	生成文本的最小长度（包括输入部分）	如果同时设置 min_new_tokens，min_new_tokens 优先
min_new_tokens	生成的新 token 的最小数量（不包括输入部分）	如果同时设置 min_length，min_new_tokens 优先
num_beams	Beam search 中的 beam 数量，用于控制生成的多样性	值越大，生成结果越多样，但计算成本越高
do_sample	是否启用随机采样生成文本	如果为 True，则生成结果会更随机；如果为 False，则使用贪婪解码或 beam search
temperature	控制生成文本的随机性	值越高，生成结果越随机；值越低，生成结果越确定
top_k	在随机采样中，只从概率最高的 k 个 token 中采样	值越大，生成结果越多样，但可能引入噪声
top_p	在随机采样中，只从累积概率大于 p 的 token 中采样	值越小，生成结果越集中于高概率 token

生成时常用参数 2-2

参数名称	含义	注意事项
early_stopping	是否在达到 max_length 或 max_new_tokens 之前提前停止生成	如果为 True，则可能生成较短的文本
eos_token_id	结束生成的 token ID	如果生成到该 token，生成过程会停止
pad_token_id	填充 token 的 ID	用于处理输入和输出的填充部分
bos_token_id	开始生成的 token ID	如果未指定输入，会用此 token 开始生成
use_cache	是否使用缓存机制（如 kv-cache）加速生成	如果为 True，可以显著提高生成速度，但需要更多的内存
output_scores	是否输出生成 token 的概率分数	如果为 True，会返回每个生成 token 的概率分数
return_dict_in_generate	是否以字典形式返回生成结果	如果为 True，返回值会包含更多详细信息，如生成的 token IDs 和概率分数

vLLM简介

北京大学 PEKING UNIVERSITY

- vLLM 是由加州大学伯克利分校 LMSYS 组织开源的大语言模型（LLM）高速推理框架。它旨在显著提升实时场景下语言模型服务的吞吐量和内存使用效率。vLLM 的主要特点包括：
  1. PagedAttention 技术：vLLM 引入了 PagedAttention 算法，通过分页管理注意力机制中的键（keys）和值（values），有效解决了传统方法中显存碎片化和过度预留的问题。
  2. 显著的性能提升：与 Hugging Face 的 Transformers 相比，vLLM 的吞吐量最高可达 24 倍。
  3. 与 Hugging Face 的无缝集成：vLLM 可以与 Hugging Face 的 Transformers 库无缝对接，用户可以轻松地在现有模型基础上进行推理加速。
  4. 支持多种推理场景：vLLM 支持离线推理、在线 API 服务，以及与 OpenAI API 兼容的接口。
- vLLM 的这些特性使其成为大语言模型推理加速的重要工具，尤其适用于需要高效资源利用和快速响应的场景。

vLLM部署简单, 更适合中小型企业做大模型推理部署, 对于大型企业, 可以使用配置较为复杂的Tensor RT框架

vLLM部署模型 2-1

北京大学 PEKING UNIVERSITY

- 步骤一：安装相关依赖包

```
pip install vllm
```

- 步骤二：加载并启动模型

```
python -m vllm.entrypoints.openai.api_server \
    --model '/root/autodl-tmp/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B' \
    --served-model-name 'deepseek-r1-7b' \
    --host 0.0.0.0 \
    --port 6006 \
    --tensor-parallel-size 1 \
    --gpu-memory-utilization 0.9 \
    --dtype=half
```

vLLM部署模型 2-2

北京大学 PEKING UNIVERSITY

```
[*]: !python -m vllm.entrypoints.openai.api_server \
        --model '/root/autodl-tmp/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B' \
        --served-model-name 'deepseek-r1-7b' \
        --host 0.0.0.0 \
        --port 6006 \
        --tensor-parallel-size 1 \
        --gpu-memory-utilization 0.9 \
        --dtype=half
```

这个参数的作用控制是模型在 GPU 上的显存占用量。如果你的显存不足，可以通过降低 --gpu-memory-utilization 的值来减少显存占用，从而避免出现 CUDA out of memory 的错误

```
INFO 02-24 14:59:54 llm_engine.py:436] init engine (profile, create kv cache, warmup model) took
24.72 seconds
INFO 02-24 14:59:54 api_server.py:958] Starting vLLM API server on http://0.0.0.0:6006
INFO 02-24 14:59:54 launcher.py:23] Available routes are:
INFO 02-24 14:59:54 launcher.py:31] Route: /openapi.json, Methods: HEAD, GET
INFO 02-24 14:59:54 launcher.py:31] Route: /docs, Methods: HEAD, GET
INFO 02-24 14:59:54 launcher.py:31] Route: /docs/oauth2-redirect, Methods: HEAD, GET
INFO 02-24 14:59:54 launcher.py:31] Route: /redoc, Methods: HEAD, GET
INFO 02-24 14:59:54 launcher.py:31] Route: /health, Methods: GET
INFO 02-24 14:59:54 launcher.py:31] Route: /ping, Methods: POST, GET
INFO 02-24 14:59:54 launcher.py:31] Route: /tokenize, Methods: POST
INFO 02-24 14:59:54 launcher.py:31] Route: /detokenize, Methods: POST
INFO 02-24 14:59:54 launcher.py:31] Route: /v1/models, Methods: GET
INFO 02-24 14:59:54 launcher.py:31] Route: /version, Methods: GET
INFO 02-24 14:59:54 launcher.py:31] Route: /v1/chat/completions, Methods: POST
INFO 02-24 14:59:54 launcher.py:31] Route: /v1/completions, Methods: POST
INFO 02-24 14:59:54 launcher.py:31] Route: /v1/embeddings, Methods: POST
INFO 02-24 14:59:54 launcher.py:31] Route: /pooling, Methods: POST
INFO 02-24 14:59:54 launcher.py:31] Route: /score, Methods: POST
```

vLLM启动后, 提供了一个API调用URL, 可以通过访问h ttp://0.0.0.0:6006/docs 来查看 API 文档, 进而通过API来调用内部的大模型

vLLM参数说明 2-1

参数名称	含义	常见值/范围	注意事项
--model	指定模型的路径或名称	模型文件路径或预训练模型名称	确保路径正确且模型文件完整
--served-model-name	指定服务中暴露的模型名称	自定义名称	用于客户端请求时指定模型
--host	服务绑定的主机地址	0.0.0.0 或 127.0.0.1 等	0.0.0.0 表示监听所有网络接口，127.0.0.1 仅监听本地接口
--port	服务监听的端口号	1024-65535 之间的整数	确保端口未被占用
--tensor-parallel-size	模型的张量并行大小	正整数（通常为 GPU 数量）	用于分布式推理，需确保 GPU 资源充足
--gpu-memory-utilization	GPU 内存利用率。	0 到 1 之间的小数（如 0.9 表示 90%）	调整显存占用比例以避免显存不足错误
--dtype	模型数据类型	auto、half、float16、bfloat16、float32 等	half 或 float16 可减少显存占用，但可能影响精度
--max-model-len	模型支持的最大输入长度	正整数（如 2048）	根据模型能力和硬件资源调整
--swap-space	用于交换空间的显存大小（单位：GB）	正整数	用于缓解显存不足，但可能降低推理速度
--cpu-offload-gb	CPU 卸载的显存大小（单位：GB）	正整数	将部分模型数据卸载到 CPU 内存，需确保 CPU 内存充足

vLLM参数说明 2-2

参数名称	含义	常见值/范围	注意事项
--max-num-batched-tokens	每批次最大 token 数量	正整数	调整以优化推理速度和资源利用率
--max-num-seqs	每批次最大序列数量	正整数	调整以优化推理速度和资源利用率
--quantization	模型量化方法	None、fp8、bfloat16、gptq 等	量化可减少显存占用，但可能影响精度
--tokenizer	指定分词器	分词器路径或名称	确保与模型兼容
--tokenizer-mode	分词器模式	auto、slow、mistral、custom 等	根据需求选择分词器模式
--load-format	模型加载格式	auto、pt、safetensors、gguf 等	根据模型文件格式选择加载方式
--revision	模型版本	版本号或分支名称	仅在使用 Hugging Face 模型时适用
--trust-remote-code	是否信任远程代码	True 或 False	启用时需注意安全性
--enable-lora	是否启用 LoRA（Low-Rank Adaptation）	True 或 False	用于模型微调，需指定 LoRA 模块
--enable-prompt-adapter	是否启用提示适配器	True 或 False	用于自定义提示词，需指定适配器配置

vLLM实际并发性能测试

北京大学 PEKING UNIVERSITY

7B并发测试

设备	模型	上下文	并发	循环次数	速率（tokens/s）	显存(GB)	请求超时个数
V100（32GB） * 1	DeepSeek-R1-Distill-Qwen-7B	2048	16	2	482.3	30.4	0
V100（32GB） * 1	DeepSeek-R1-Distill-Qwen-7B	4096	16	2	435.7	30.4	1
V100（32GB） * 1	DeepSeek-R1-Distill-Qwen-7B	8192	16	2	402.1	30.4	12
V100（32GB） * 1	DeepSeek-R1-Distill-Qwen-7B	8192	1	1	42.5	30.4	0

14B并发测试

设备	模型	上下文	并发	循环次数	速率（tokens/s）	显存(GB)	请求超时个数
V100（32GB） * 2	DeepSeek-R1-Distill-Qwen-14B	2048	16	2	462.3	60.1	0
V100（32GB） * 2	DeepSeek-R1-Distill-Qwen-14B	4096	16	2	372.4	60.1	12
V100（32GB） * 2	DeepSeek-R1-Distill-Qwen-14B	8192	16	2	293.2	60.1	23
V100（32GB） * 2	DeepSeek-R1-Distill-Qwen-14B	8192	1	1	37.5	60.1	0

vLLM常见问题及解决方案

注意力机制与PagedAttention兼容性问题

- 问题场景：
  - 模型使用相对位置编码（如RoPE）或稀疏注意力模式时，vLLM的PagedAttention出现位置偏移错误
  - 长序列推理时缓存命中率低于预期
- 优化策略：
  - 自定义Attention Kernel：

```
from vllm._C import ops
ops.paged_attention_v2(
    query, key, value,
    position_offsets,  # 注入自定义位置编码
    block_tables,
    context_len,
    rotary_dim=64
)
```

缓存策略调优：

```
# 启动参数调整
--block_size 32  # 根据序列长度分布调整块大小
--gpu_memory_utilization 0.95
```

企业级70B模型部署案例分享-1

北京大学 PEKING UNIVERSITY

企业场景

企业（**向量智能**）内部应用案例，为各个部门提供Ai能力，为各部门建立知识库和应用集成，提高企业内部的工作效率等。

服务器配置

- CPU：双路64核心，共128线程)
- RAM： 512G内存
- DISK：2T【RAID5磁盘阵列】
- GPU： 8卡4090【192G显存】
- Network：万兆双网卡（公司内部为千兆带宽）

生产环境

- Python Env：Anaconda3
- Python Version：3.11.11
- Torch Version：pytorch2.6.0
- 推理框架：vLLM

部署方案

- 采用vLLM的高性能部署方案
- 采用半精度的方式运行

DeepSeek-R1-Distill-Llama-70B

vLLM启动命令

```
vllm serve /ai/models/DeepSeek-R1-Distill-Llama-70B --max-model-len 8192 --port 5000 --tensor-parallel-size 8 --dtype auto --gpu-memory-utilization 0.8 --served-model-name DeepSeek-R1-Distill-Llama-70B
```

企业级70B模型部署案例分享-2

北京大学 PEKING UNIVERSITY

初始化算力（8卡4090）的GPU状态

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.78                 Driver Version: 550.78         CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4090        Off |   00000000:1A:00.0 Off |                  Off |
| 30%   25C    P8             23W /  450W |   19766MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA GeForce RTX 4090        Off |   00000000:1B:00.0 Off |                  Off |
| 30%   26C    P8             13W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA GeForce RTX 4090        Off |   00000000:1C:00.0 Off |                  Off |
| 30%   25C    P8             14W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA GeForce RTX 4090        Off |   00000000:1D:00.0 Off |                  Off |
| 30%   25C    P8             16W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   4  NVIDIA GeForce RTX 4090        Off |   00000000:9A:00.0 Off |                  Off |
| 30%   27C    P8             24W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   5  NVIDIA GeForce RTX 4090        Off |   00000000:9B:00.0 Off |                  Off |
| 30%   25C    P8             14W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   6  NVIDIA GeForce RTX 4090        Off |   00000000:9C:00.0 Off |                  Off |
| 30%   26C    P8             20W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   7  NVIDIA GeForce RTX 4090        Off |   00000000:9D:00.0 Off |                  Off |
| 30%   24C    P8             16W /  450W |   19702MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

经验分享

- 8卡4090 不足以支持float32精度运行
- 可支持float16和bfloat16精度
- 10并发下4090功率达到300左右
- 10并发GPU显存使用量稳定在21-22G
- 10并发下每秒15-40 Tokens

总结

此配置的服务器支持百人团队的日常使用

随着 DeepSeek 的爆火，低成本实现更好模型性能的部署方式受到广泛关注，目前出现了KTransformers、Unsloth等多套解决方案。我们通过实践发现，现有的低成本部署方案尚不足以满足企业级应用需求，仍处于研究阶段，更适合个人或小团队用于研究参考，其重点在于探索在成本效益下实现部署，最大优势是显著节约成本。下面我们将对于这些解决方案分别进行数据和经验分享。

注意：后面这一部分低成本部署DeepSeek的内容（约30页），也是在快速严谨和变化的技术内容，目标是让大家简单了解这种低成本部署方案，不是了解DeepSeek本地化部署必须掌握的内容。

部署方案简介

北京大学 PEKING UNIVERSITY

R1满血版模型部署方案

DeepSeek R1模型就成了很多应用场景下的当务之急。受限于DeepSeekR1 671B(6710亿参数)的模型规模，通常情况下部署Deepseek R1满血版模型需要1200G左右显存(考虑百人内并发情况)，需要双节点8卡H100服务器才能运行(总成本约在260万-320万左右)，即便是INT4半精度下，也至少需要490G显存，需要单节点8卡H100服务器才能运行。

为了实现低成本DeepSeek R1模型的高性能部署，目前大多数方案都采用了牺牲模型推理速度的策略。使用CPU+GPU混合推理的方式，将一部分推理计算转移到CPU上，降低GPU的负载。由于CPU并不适合深度学习计算，导致模型整体推理速度较慢。

部署方案简介

北京大学 PEKING UNIVERSITY

方案1、llama.cpp

2023年3月，即Llama第一代模型开源不久，GeorgiGerganov在GitHub上发起了一个名为llama.cpp的项目，该项目用C语言编写深度学习底层张量计算库，极大程度降低了大模型等深度学习算法的计算门槛，并最终使得大模型可以在消费级CPU上运行。目前该项目已成为大模型量化的标准解决方案，DeepSeek R1模型的Q2、Q4、Q8等模型量化都是借助llama.cpp完成。

项目地址：https://github.com/ggml-org/llama.cpp

方案特点

借助llama.cpp，可以使用纯CPU模式来运行DeepSeek R1模型，但需要大量的内存来加载模型权重，并且运行速度非常慢，即使是志强4代这种较强性能的CPU，DeepSeek R1 Q4_K_M模型推理速度也只有4tokens/s左右。而且并发性能较差，一个400字的小作文，就得写个2、3分钟。

部署方案简介

北京大学 PEKING UNIVERSITY

方案2、KTransformers

KTransformers(Quick Transformers)项目是清华大学发起的，可以借助R1模型的MoE架构特性，将专家模型的权重加载到内存上，并分配CPU完成相关计算工作，同时将MLA/KVCache加载到GPU上，进而实现CPU+GPU混合推理，这样就达到了最大化降低显存占用的目的。

项目地址：https://github.com/kvcache-ai/ktransformers

方案特点

与llama.cpp的问题类似，模型推理速度也会因为CPU的特性而变慢，需要大量的内存来加载模型权重。经过测评，4代志强芯片能达到10token/s，并且KTransformersduiGPU性能挖掘不够充分，高并发场景表现较为乏力，适合小团队及个人学习使用。但其优势在于内存价格便宜，整体上部署成本较低。

部署方案简介

北京大学 PEKING UNIVERSITY

方案3、Unsloth动态量化

Unsloth团队提出了动态量化方案，所谓动态量化的技术，指的是可以围绕模型的不同层，进行不同程度的量化，关键层量化精度较高，非关键层量化精度较低。该团队最终得到了一组比Q2量化程度更深的模型组，分别是1.58-bit、1.73-bit和2.22-bit模型组。尽管量化精度较低，但实际性能其实并不弱。并且，Unsloth也实现了把模型权重分别加载到CPU和GPU上的方法，用户可以根据自己实际硬件情况，自行选择加载到CPU内存上计算的模型权重数量。

项目地址：https://github.com/kvcache-ai/ktransformers

方案特点

和llama.cpp深度融合，直接通过参数设置即可自由调度CPU和GPU计算资源，灵活高效。并且能够直接和ollama、vLLM、Open-WebUI等框架兼容。深度挖掘GPU性能，相较于KTransformers和llama.cpp，有着较高的并发能力。但为了保障一定的并发量，该方案对硬件基础要求高一些。

性能测评总结

北京大学 PEKING UNIVERSITY

方案	测试场景	硬件配置	资源占用	对话效果(tokens/s)
KTransformer	单并发测试	志强3代CPU+4090GPU+DDR4内存	——	4
KTransformer	单并发测试	9654CPU+4090GPU+12x4800MHz DDR5内存	——	14
Llama.cpp	纯CPU推理	480G内存+4卡4090服务器	仅CPU推理	3.23
Llama.cpp	CPU+GPU推理(单卡4090)	单卡4090	占用显存约23G	3.65
Llama.cpp	CPU+GPU推理(4卡4090)	4卡4090	占用显存约92G	5.78
Llama.cpp	纯GPU推理(4卡H800)	4卡H800,320G总显存	占用显存约140G	20.93
Ollama	纯GPU推理(单卡4090)	单卡4090	——	5.97
Ollama	纯GPU推理(双卡A100)	双卡A100	——	20

部署方案总结

北京大学 PEKING UNIVERSITY

方案总结

- 针对之前提到的三种DeepSeek满血版低成本部署方案，北大AI肖睿团队评价结论是：这些方案主要聚焦于探索如何在保持成本效益的同时实现模型部署。它们的最大亮点在于显著的成本节约。然而，从刚刚的方案性能测评总结中可以看出，就当前的技术水平而言，这三种方案在考虑并发处理能力、推理速度以及推理准确性等方面，尚不足以满足企业级应用的要求。因此，它们更适合用于个人或小团队对低成本部署策略的研究与参考。此外，量化模型的推理效果与DeepSeek官方发布的模型相比仍存在一定差距。

- 在实际的企业级部署中，建议选用如vLLM、Tensor RT等更为成熟的推理框架。特别是对于方案二中提到的KTransformers框架，实际部署时对环境要求极为严格。从该项目的更新日志来看，项目仍处于初始阶段，相较于vLLM等成熟的推理框架，其发展速度较慢，更像是一个年久失修的项目。

- 尽管如此，我们对低成本部署大模型的未来发展持乐观态度，相信在不久的将来，这一领域将取得显著进展，达到企业实际应用的要求标准。

方案实现

硬件环境配置

- 深度学习环境：PyTorch 2.5.0、Python 3.12(ubuntu22.04)、Cuda 12.4
- 硬件环境:
  - GPU:RTX4090(24GB) * 4 (实际只使用一张GPU)
  - CPU: 64 vCPU Intel(R) Xeon(R) Gold 6430
  - 内存:480G(至少需要382G)
  - 硬盘:550G(实际使用需要380G左右)

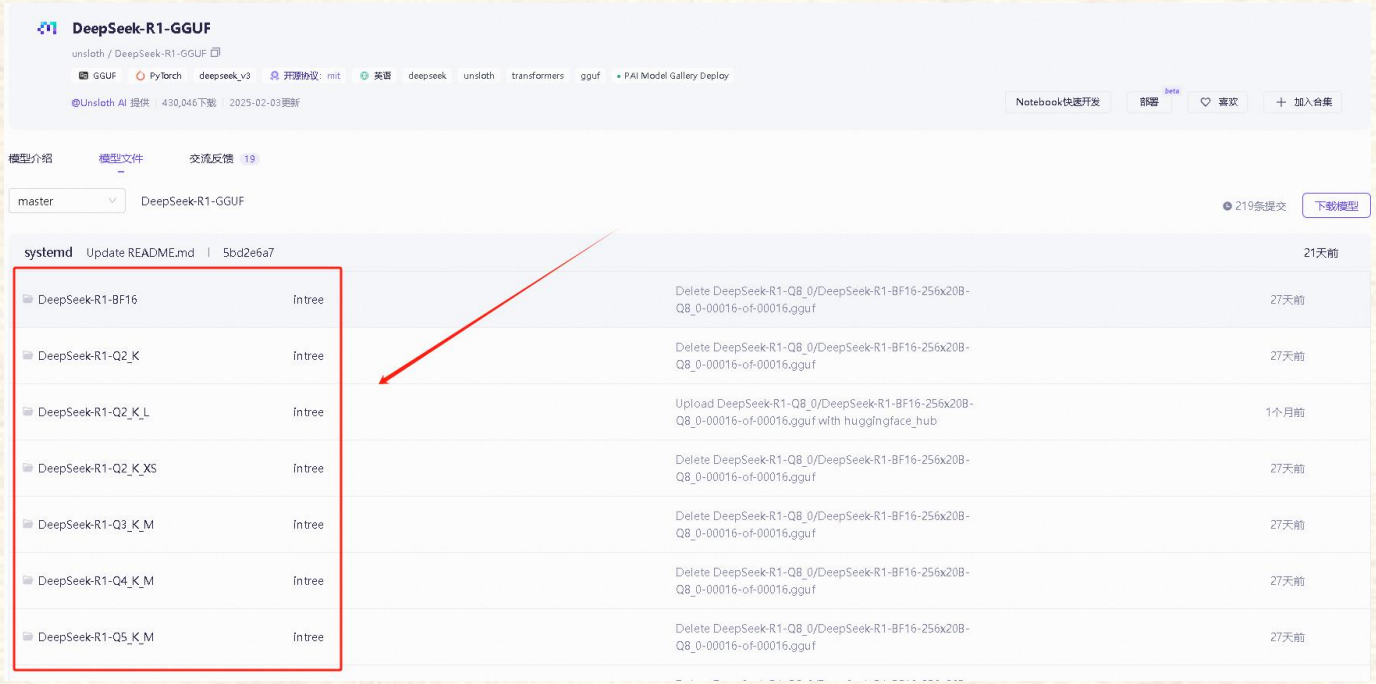
硬件说明

- 本部分所介绍的所有方案均在AutoDL服务器上实现和运行
- 由于AutoDL为虚拟化环境，性能方面会受影响。
- AutoDL地址：https://www.autodl.com/

方案实现

模型下载

- 三种方案都需要下载模型权重文件，这里采用ModelScope社区下载方式。Unsloth团队在ModelScope上传了DeepSeek R1模型的多个版本，除了Unsloth提供的4 个动态量化版本，还包含了Q2、Q4、Q8等多个版本的量化模型文件，大家可以根据需要自行加载。
- Unsloth团队DeepSeek R1模型的ModelScope地址：https://modelscope.cn/models/unsloth/DeepSeek-R1-GGUF/files



方案实现

模型下载

ModelScope提供了多种下载方式这里演示SDK下载。首先，需要在本地环境中安装ModelScope的SDK包，之后运行下列的python代码。如需下载不同版本的R1模型，修改allow_patterns参数即可。下载过程较为缓慢，需要耐心等待。

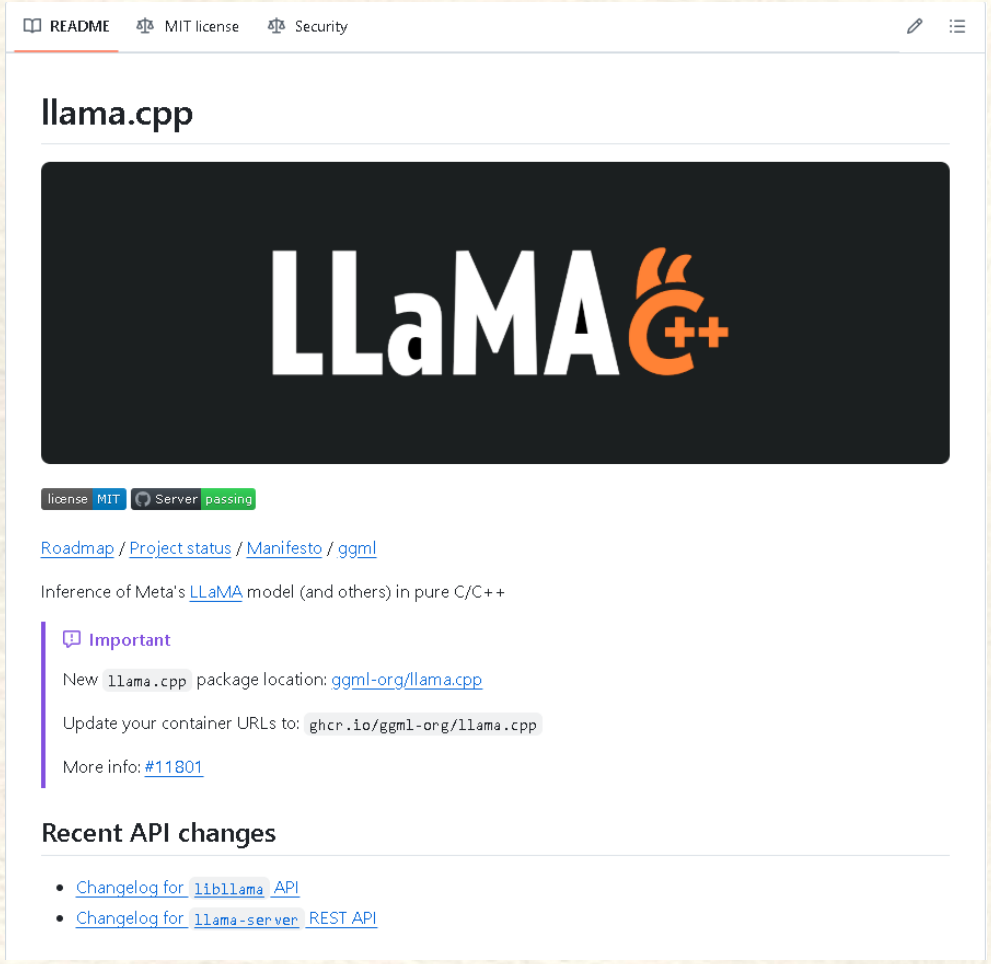
安装modelscope的SDK

```
# 命令行运行
pip install modelscope
```

运行python代码

```
from modelscope import snapshot_download
snapshot_download(
  repo_id = "unsloth/DeepSeek-R1-GGUF",
  local_dir = "DeepSeek-R1-GGUF",
  allow_patterns = ["*UD-IQ1_S*"]
  # Select   quant type UD-IQ1_S for 1.58bit
)
```

方案一、 llama.cpp



由于llama.cpp是个C语言项目，实际调用过程需要先构建项目，再设置参数进行编译，然后创建可执行文件(类似于脚本)，再运行本地大模型。借助llama.cpp和Unsloth的模型权重，可以实现纯CPU推理、纯GPU推理和CPU+GPU混合推理。这里我们尝试纯CPU运行模式。

项目主页：https://github.com/ggml-org/llama.cpp

方案一、 llama.cpp

北京大学 PEKING UNIVERSITY

安装步骤

步骤一：安装依赖

apt-get update

apt-get install build-essential cmake curl libcur14-openss1-dev -y

注：C语言项目在运行前需要对项目进行代码编译，这里安装的了项目创建和代码编译的相关依赖。其中，cmake为跨平台构建工具，用于管理项目的编译过程。

步骤二：llama.cpp源码下载

git clone https://github.com/ggml-org/llama.cpp -i https://pypi.tuna.tsinghua.edu.cn/simple

注： 从清华源拉取llama.cpp的项目源码。

方案一、 llama.cpp

北京大学 PEKING UNIVERSITY

安装步骤

```
# 步骤三：项目构建与编译
cmake llama.cpp -B llama.cpp/build \
-DBUILD SHARED LIBS=OFF -DGGML_CUDA=ON -
DLLAMA_CURL=ON

cmake --build llama.cpp/build --config Release -j --clean-first -
-target llama-quantize llama-cli llama-gguf-split
```

```
# 步骤四：将可执行文件复制都到项目根目录
cp llama.cpp/build/bin/llama-* llama.cpp
```

这里将可执行文件复制到根目录是为了更方便地在根目录下执行

步骤三核心参数说明：

- **cmake**：运行 CMake 工具，用于配置和生成构建文件。
- **llama.cpp**：指定项目的源代码所在的目录。在这个例子中，llama.cpp 是项目的根目录。
- **-B llama.cpp/build**：指定生成构建文件的目录。 -B 参数表示构建目录，这是 CMake 将生成的文件存放的地方（例如 Makefile 或 Ninja 构建文件）。
- **--config**：指定构建的配置为 Release 配置，目的是启用优化配置，加快构建后程序的运行速度。
- **--clean-first**：表示在构建之前先清理掉之前的构建结果。这可以确保每次构建时都是从一个干净的状态开始，避免由于缓存或中间文件引起的编译错误。
- **--target**：指定构建的目标（target）。通常，一个项目会定义多个目标（比如库、可执行文件等），通过这个参数可以告诉 CMake 只编译特定的目标。

方案一、 llama.cpp

北京大学 PEKING UNIVERSITY

推理步骤

```
cd llama.cpp

./llama-cli \
--model /root/autodl-tmp/Deepseek-R1-GGUF/Deepseek-R1-UD-IQ1_S/DeepSeek-R1-UD-IQ1_S-00001-0f-00003.gguf \
--cache-type-k q4_0 \
--threads 64 \
--prio 2 \
--temp 0.6 \
--ctx-size 512 \
--seed 3407 \
--n-gpu-layers 0 \
-no-cnv \
--prompt "<| User |>你好，好久不见，请介绍下你自己。
<|Assistant|>"
```

核心参数说明：

- **--cache-type-k**： K缓存量化为4bit
- **--threads**： CPU核心数
- **--temp**： 模型温度参数，控制生成随机性
- **--ctx-size**： 输出上下文长度
- **--seed**： 随机数种子
- **--n-gpu-layers**： 控制GPU中的模型层数，为0时则代表完全用CPU推理
- **-no-cnv**： 不进行多轮对话

方案一、 llama.cpp

--n-gpu-layers参数设置

为了尝试CPU+GPU混合推理，只需要合理的设置--n-gpu-layers 参数，参数大小可以根据下图的公式确定，即可灵活的将模型的部分层加载到GPU上进行运行。并且无需手动设置，llama.cpp会自动识别当前GPU数量以及可以分配的显存，自动将模型权重加载到各个不同的GPU上。详细参数可以参考下表。

$$n_{\text{offload}} = \frac{\text{VRAM}(GB)}{\text{Filesize}(GB)} \times n_{\text{layers}} - 4$$

Quant	文件大小	24GB GPU	80GBGPU	2x80GB GPU
1.58bit	131	7	33	61
1.73bit	158	5	26	57
2.22bit	183	4	22	49
2.51bit	212	2	19	32

方案一、 llama.cpp

三种推理方式测试结果

1. 纯CPU推理测试：在这种模式下，系统完全依赖内存和CPU进行计算，不使用GPU加速。本次测试使用的服务器配置为480GB内存和4卡4090显卡，但GPU不参与计算。在高并发场景下，内存占用最高可达到约180GB。在这种配置下，生成token的速度为每秒3.23个token。

2. CPU+GPU混合推理:只需要合理设置 --n-gpu-layers 参数，就可以灵活地将模型的部分层加载到GPU上运行。这个过程不需要手动配置，因为 llama.cpp 会自动识别当前GPU的数量以及可用的显存，并将模型权重自动分配到不同的GPU上。在单卡4090设备上运行时，GPU能够容纳大约7层的模型权重，生成速度为每秒3.65个token，总共占用显存23GB。

3. 将更多的模型权重加载到GPU进行推理：以4卡4090服务器为例，总显存为96GB。根据计算公式，这时每个GPU可以容纳大约39层的模型权重。与单卡24GB显存相比，生成token的速度提高到了每秒5.78个token，同时占用的显存约为92GB。

方案二、KTransformers

配置文件下载

在开始部署KTransformer之前，需要注意的是，Unsloth团队提供的只有模型权重，进行模型推理还需要下载DeepSeek官方提供的分词器等模型配置文件。

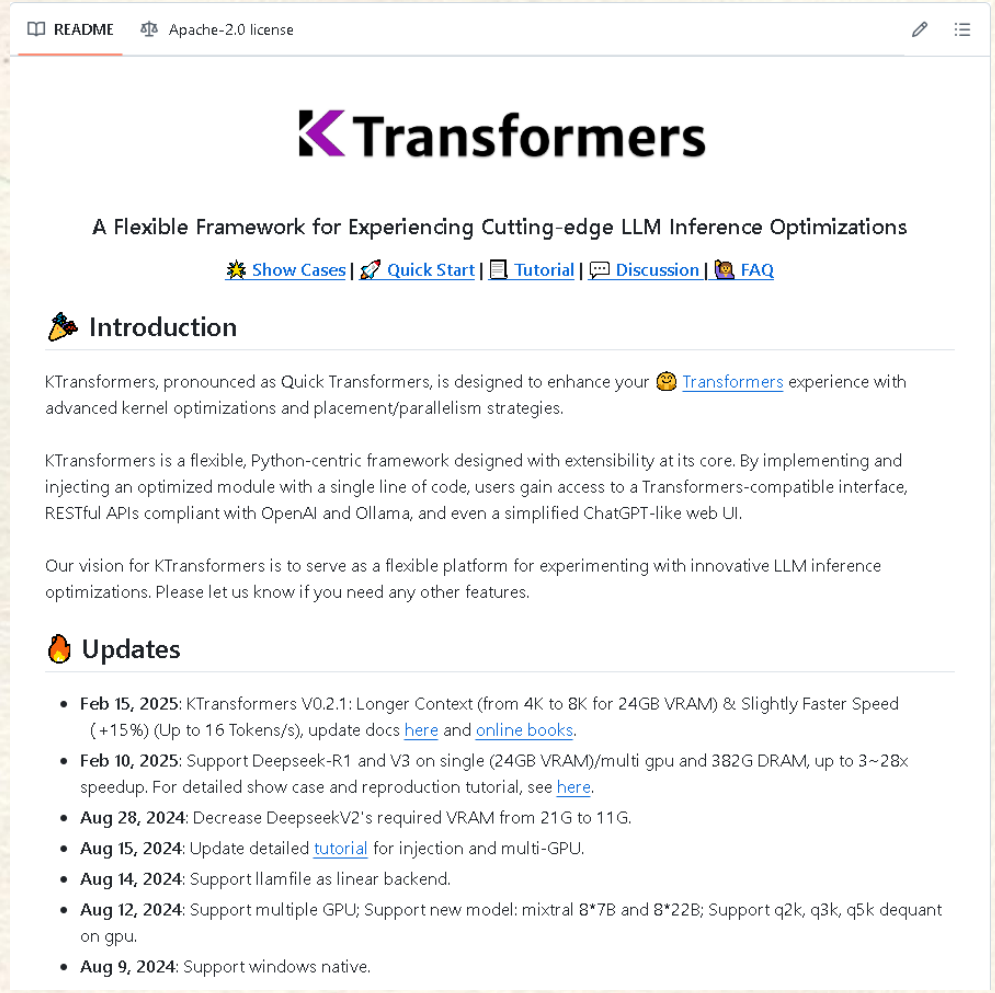
配置文件地址：https://modelscope.cn/models/deepseek-ai/DeepSeek-R1/files

```
# 配置文件下载
modelscope download --model deepseek-ai/DeepSeek-R1 README.md .gitattributes config.json configuration_deepseek.py generation_config.json LICENSE model.safetensors.index.json modeling_deepseek.py tokenizer.json tokenizer_config.json  --local_dir /root/autodl-tmp/DeepSeek-R1-GGUF/R1_config
```

核心参数说明：

- **--model：指定modelscope的下载文件路径下及下载内容，这里我们下载了deepseek-ai/DeepSeek-R1路径下除模型权重外的模型配置文件**
- **--local_dir：设置下载文件的存放位置**

方案二、KTransformers



README Apache-2.0 license

KTransformers

A Flexible Framework for Experiencing Cutting-edge LLM Inference Optimizations

Show Cases | Quick Start | Tutorial | Discussion | FAQ

Introduction

KTransformers, pronounced as Quick Transformers, is designed to enhance your Transformers experience with advanced kernel optimizations and placement/parallelism strategies.

KTransformers is a flexible, Python-centric framework designed with extensibility at its core. By implementing and injecting an optimized module with a single line of code, users gain access to a Transformers-compatible interface, RESTful APIs compliant with OpenAI and Ollama, and even a simplified ChatGPT-like web UI.

Our vision for KTransformers is to serve as a flexible platform for experimenting with innovative LLM inference optimizations. Please let us know if you need any other features.

Updates

- **Feb 15, 2025**: KTransformers V0.2.1: Longer Context (from 4K to 8K for 24GB VRAM) & Slightly Faster Speed (+15%) (Up to 16 Tokens/s), update docs here and online books.
- **Feb 10, 2025**: Support Deepseek-R1 and V3 on single (24GB VRAM)/multi gpu and 382G DRAM, up to 3~28x speedup. For detailed show case and reproduction tutorial, see here.
- **Aug 28, 2024**: Decrease DeepseekV2's required VRAM from 21G to 11G.
- **Aug 15, 2024**: Update detailed tutorial for injection and multi-GPU.
- **Aug 14, 2024**: Support llamfile as linear backend.
- **Aug 12, 2024**: Support multiple GPU; Support new model: mixtral 8*7B and 8*22B; Support q2k, q3k, q5k dequant on gpu.
- **Aug 9, 2024**: Support windows native.

- 准备好模型权重和DeepSeek R1模型配置文件之后，即可开始部署KTransformer，本次部署DeepSeek R1 Q4_K_M模型。该项目部署流程非常复杂，请务必每一步都顺利完成后，再执行下一步。
- 项目主页：**https://github.com/kvcache-ai/ktransformers**
- 项目说明：KTransformer目前开放了V2.0、V2.1和V3.0三个版本（V3.0目前只有预览版，只支持二进制文件下载和安装），其中V2.0和V2.1支持各类CPU，但从V3.0开始，只支持最新几代的Intel CPU。但版本间实际部署流程和调用指令没有区别，本方案以适配性较好的V2.0版本进行演示。

方案二、KTransformers

安装步骤一：安装依赖

一、安装gcc、cmake等基础库包：

```
apt-get update
apt-get install gcc g++ cmake ninja-build
```

二、安装PyTorch、flash-attn等库包

```
pip install torch==2.5.0 packaging ninja cpufeature
numpy
pip install flash-attn
```

三、安装libstdc：

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install --only-upgrade libstdc++6
conda install -c conda-forge libstdcxx-ng
```

核心安装步骤说明：

- **apt-get update**：更新软件包列表，确保获取最新的软件包信息。
- **apt-get install gcc g++ cmake ninja-build**：安装编译器（gcc和g++），CMake构建系统，以及Ninja构建工具。这些都是开发过程中常用的工具。
- **sudo apt-get install --only-upgrade libstdc++6**：仅升级libstdc++6库，它是GNU标准C++库的一部分，提供了C++程序运行时支持。

方案二、KTransformers

安装步骤二：拉取代码及编译

如果因为网络问题拉取缓慢，可以直接打开链接将项目下载下来，再上传服务器。

四、拉取KTransformers项目代码：

git clone https://github.com/kvcache-ai/ktransformers.git

cd ktransformers

git submodule init

git submodule update

五、根据CPU类型，如果是64核双槽版本，则需要运行命令，且该命令只需要在编译时运行一次即可：

export USE_NUMA=1

例如，CPU：64 vCPU Interl(R) Xeon(R) Gold 6430 代表的就是64核双槽CPU。

如果CPU类型是双槽版本而未执行，后续代码步骤可能会报错，此时再次执行该命令即可运行后续命令。

此处编译过程耗费时间较长，需要耐心等待。。。

六、开始编译：

sh ./install.sh 或者 bash install.sh

七、查看安装情况：

pip show ktransformers

方案二、KTransformers

安装步骤三：运行模型

八、运行模型：

```
python ./ktransformers/local_chat.py --model_path
/root/autodl-tmp/DeepSeek-R1-GGUF/R1_config --
gguf_path /root/autodl-tmp/DeepSeek-R1-GGUF  --
cpu_infer 65 --max_new_tokens 1000 --force_think true
```

启动过程需要加载61层模型权重参数，耐心等待。
官方提供的对话脚本默认输出响应速度。

核心参数说明：

- **./ktransformers/local_chat.py**：调用官方提供的最简单的对话脚本。
- **--model_path**：设置为前文下载好的配置文件路径，也可以是来自 Hugging Face 的在线路径（如 deepseek-ai/DeepSeek-V3 ）。
- **--gguf_path**：模型路径地址，建议下载并量化模型以满足需求（注意，这是目录路径）。
- **--max_new_tokens**：1000 是最大输出token长度。如果发现答案被截断，可以增加该值以获得更长的答案，但设置过大会导致爆显存（OOM）问题，并且可能减慢生成速度。
- **--force_think true**：输出R1模型的推理思维链。
- **--cpu_infer 65**：若是单槽版本CPU，则不用输入参数。

方案二、KTransformers

单并发实测效果

在配置： 64 vCPU Interl(R) Xeon(R) Gold 6430 ； 4x4090(24G)下，生成token速度为8.15tokens/s，推理时占用显存11G

```
Chat: 你好，介绍一下你自己
<think>

您好！我是由中国的深度求索（DeepSeek）公司开发的智能助手
。
</think>

您好！我是由中国的深度求索（DeepSeek）公司开发的智能助手
。
prompt eval count:    11 token(s)
prompt eval duration: 0.8419985771179199s
prompt eval rate:     13.064155093529898 tokens/s
eval count:           77 token(s)
eval duration:        9.447279214859009s
eval rate:            8.15049478784238 tokens/s
```

```
| NVIDIA-SMI 550.120                Driver Version: 550.120        CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4090        On  |   00000000:16:00.0 Off |                  Off |
| 30%   32C    P2            104W /  450W |   11461MiB /  24564MiB |     21%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA GeForce RTX 4090        On  |   00000000:27:00.0 Off |                  Off |
| 30%   28C    P8             26W /  450W |       4MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA GeForce RTX 4090        On  |   00000000:5A:00.0 Off |                  Off |
| 30%   29C    P8             25W /  450W |       4MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA GeForce RTX 4090        On  |   00000000:D8:00.0 Off |                  Off |
| 30%   31C    P8             25W /  450W |       4MiB /  24564MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

方案二、KTransformers

单并发实测效果

在配置： 64 vCPU Interl(R) Xeon(R) Gold 6430 ； 4x4090(24G)下，生成token速度为8.73tokens/s，推理时占用显存11G

```
Chat: 3.11和3.80谁大
<think>

嗯，用户问的是3.11和3.80哪个大。首先，我需要确认这两个数的数值大小。先看整数部分，两个数的整数部分都是3，所以整数部分一样大。接下来比较小数部分。3.11的小数部分是0.11，而3.80的小数部分是0.80。这时候需要确定0.11和0.80哪个大。显然，0.80比0.11大很多，因为0.8相当于80分，而0.11只有11分。所以整体来看，3.80比3.11大。不过，可能用户对小数比较不太熟悉，或者容易混淆小数点后的位数。比如，有人可能会误以为0.11比0.8大，因为11比8大，但实际上小数部分的比较是从左到右逐位进行的。0.11的第一位是1，而0.80的第一位是8，所以0.80更大。因此，综合整数和小数部分，3.80大于3.11。为了确保正确，可以再检查一遍：3.11加上0.69等于3.80，所以3.80确实更大。或者用分数来比较，3.11可以写成311/100，3.80是380/100，显然380比311大。所以无论哪种方法比较，结果都是3.80更大。用户可能是在做作业或者实际应用中遇到这个问题，需要明确的答案和解释。所以最终结论是3.80比3.11大。\n
</think>

3.80比3.11大。

**步骤解析：**
1. **比较整数部分：** 两个数的整数部分都是3，因此整数部分相等。
2. **比较小数部分：**
   - 3.11的小数部分是0.11（即11/100）
   - 3.80的小数部分是0.80（即80/100）
   - 由于80/100 > 11/100，所以3.80的小数部分更大。

**结论：** 3.80 > 3.11
（可用数轴辅助理解：3.11位于3.1和3.2之间，而3.80位于3.8的位置，显然更靠右）
prompt eval count:    16 token(s)
prompt eval duration: 1.022550106048584s
prompt eval rate:     15.647154995493002 tokens/s
eval count:           468 token(s)
eval duration:        53.56120944023132s
eval rate:            8.737666772111238 tokens/s
```

```
Chat:  一个农夫要带着一只狼、一只羊和一棵卷心菜过河。他只有一条小船，每次只能带一样东西过河。如果狼和羊单独在一起，狼会吃羊;如果羊和卷心菜单独在一起，羊会吃卷心菜。 问:农夫如何才能安全地将狼羊和卷心菜都运过河?
<think>

嗯，这个问题看起来有点挑战性，不过让我仔细想想。首先，农夫需要把狼、羊和卷心菜都安全地运过河，但每次只能带一样东西，而且如果狼和羊单独在一起，狼会吃羊；同样，羊和卷心菜单独在一起时，羊会吃卷心菜。所以关键是要确保任何时候农夫不在的时候，狼和羊不能单独在一起，羊和卷心菜也不能单独在一起。

首先，我需要明确
。每次农夫过河只

可能的步骤应该是

1. 农夫先带羊过河
在一起不会互相伤

2. 接下来，农夫需
候右岸有狼和羊，

那换另一个选择，
羊会吃卷心菜，所

这时候可能需要农
来。然后第二步，

比如，农夫先带羊
否则狼和羊会在一

1. 农夫带羊到右岸，回来左岸。

2. 农夫带狼到右岸，然后把羊带回来左岸。这时候右岸有狼，左岸有羊和卷心菜。

3. 农夫把卷心菜带到右岸，这时候右岸有狼和卷心菜，左岸有羊。农夫再回来接羊。

不过这样可能可行。让我再检

步骤1：带羊过河到右岸，农夫

步骤2：农夫带狼到右岸，然后

步骤3：农夫带卷心菜到右岸，

步骤4：农夫带羊到右岸。完成

这样整个过程都是安全的，因
有狼和卷心菜，它们不会互相

或者另一种方式：

1. 农夫带羊过河，回来。

2. 带卷心菜过河，把卷心菜留

3. 带狼过河，回来接羊。

不过需要详细检查每一步的状态。

比如：

步骤1：带羊到右岸，回来。左岸：狼、卷心菜；右岸：羊。

步骤2：带卷心菜到右岸，此时右岸有羊和卷心菜，所以必须把羊带回去。所以农夫带卷心菜到右岸，然后带羊回左岸。此时左岸：狼、羊；右岸：卷心菜。

但这时候左岸有狼和羊，如果农夫离开，狼会吃羊，所以这不行。因此，步骤2不能直接带卷心菜过去，而是需要调整。

所以正确的步骤应该是：

1. 农夫带羊到右岸，回来。

2. 农夫带狼到右岸，然后带羊回来。

3. 农夫带卷心菜到右岸，回来。

4. 农夫带羊到右岸。

这样每一步的状态：

1. 初始：左岸（农夫、狼、羊、卷心菜
prompt eval count:    84 token(s)
prompt eval duration: 2.549567461013794s
prompt eval rate:     32.9467650040524 tokens/s
eval count:           1000 token(s)
eval duration:        113.8695900440216s
eval rate:            8.78197593943566 tokens/s
```

方案二、KTransformers

KT部署需要注意的问题

- 深度学习环境严格要求：该项目对环境要求苛刻，例如，torch版本不一致的话可能会导致后续依赖库安装出现问题，我们的环境版本为：Pytorch 2.5.0、Python 3.12、CUDA12.4。
- 硬件要求：该实测是在RTX 4090(24GB) * 4（实际只用1张），64 vCPU Intel(R) Xeon(R) Gold 6430下进行的，在使用Q4量化版本的R1时，最少需要保证有20G以上的显存和382G的CPU内存。
- Ktransformer目前有多个版本：V2.0，V2.1，V3.0。本实验采用的是V2.1版本，不同版本差异较大，请注意辨别。
- 在安装依赖的过程中要注意安装依赖库的先后的顺序，否则会导致其他依赖库无法安装的问题，例如flash-attn在安装过程中遇到的（“**Building wheel for flash-attn (setup.py) ... error error: subprocess-exited-with-error**”）
- 在git初始化的时候：**git submodule init** 若出现初始化失败报错的情况，尝试git clone KT官网的链接。
- 在运行sh ./install.sh 安装运行脚本时，可能会需要等待较长时间（**Building wheels for collected packages: ktransformers Building wheel for ktransformers (pyproject.toml) ...** |）

方案三、Unsloth动态量化+Ollama

本部分内容详细介绍如何通过llama.cpp工具合并Unsloth动态量化模型的权重文件，并借助Ollama工具进行模型注册与调用。通过合并权重文件，解决了Ollama对单文件支持的限制；通过Ollama的高效管理与推理接口，实现了模型的快速部署与性能验证。这一流程为Unsloth动态量化模型的本地部署提供了完整的解决方案。在实际应用中，用户可以根据硬件配置和需求，灵活调整模型参数，以优化推理速度和资源利用率。

方案三、Unsloth动态量化+Ollama

权重文件合并

在部署Unsloth动态量化模型时，模型权重通常以分片形式存储，例如，DeepSeek-R1-UD-IQ1_S模型的权重可能分为多个文件，每个文件包含模型的部分权重。然而，Ollama仅支持单个GGUF格式的模型权重文件，而Unsloth动态量化模型的权重通常以分片形式存储。为了使Ollama能够加载和管理这些模型，必须将分片的权重文件合并为一个完整的GGUF文件。这一过程不仅确保了模型的完整性，还为后续的推理任务提供了基础支持。

这一过程通过llama.cpp工具完成，确保模型能够高效地被Ollama管理并用于推理任务。合并步骤包括准备权重文件、执行合并命令以及验证合并结果。

ollama的下载与llama.cpp的下载可以参考前文。

方案三、Unsloth动态量化+Ollama

权重文件合并

- 使用llama.cpp提供的llama-gguf-split工具执行权重合并操作。该工具能够将分片的权重文件合并为一个完整的GGUF文件。具体命令如下：

```
mkdir DeepSeek-R1-UD-IQ1_S-merge
cd ./llama.cpp
./llama-gguf-split --merge /root/autodl-tmp/DeepSeek-R1-GGUF/DeepSeek-R1-UD_x0002_IQ1_S/DeepSeek-R1-UD-IQ1_S-00001-of-00003.gguf merged_file.gguf
```

- 该命令将指定路径下的分片权重文件合并为merged_file.gguf，并保存至当前目录。在执行命令时，需要注意以下几点：
  1. 确保路径正确无误，避免因路径错误导致文件无法找到。
  2. 如果在合并过程中遇到权限问题，可以尝试使用sudo命令提升权限。
  3. 合并过程可能需要一定时间，具体取决于文件大小和系统性能。

方案三、Unsloth动态量化+Ollama

北京大学 PEKING UNIVERSITY

借助Ollama调用动态量化模型

在完成模型权重合并后，下一步是将合并后的模型注册到Ollama中，并通过Ollama进行调用。Ollama提供了便捷的模型管理与推理接口，能够高效地加载和运行动态量化模型。Ollama支持多种模型格式，并提供了丰富的配置选项，用于优化模型性能。通过Ollama，用户可以轻松地管理多个模型，并在本地环境中进行高效的推理任务。

方案三、Unsloth动态量化+Ollama

创建Ollama模型配置文件

- 为了将合并后的模型注册到Ollama中，需要创建一个模型配置文件（如DeepSeekQ1_Modelfile）。该文件包含模型的基本参数和运行配置，用于指导Ollama如何加载和运行模型。配置文件的格式如下：

```
FROM ./merged_file.gguf
PARAMETER num_gpu 7
PARAMETER num_ctx 2048
PARAMETER temperature 0.6
TEMPLATE "< | User | >{{ .System }} {{ .Prompt }}< | Assistant | >"
```

- 其中：
  - FROM：指定合并后的模型文件路径。
  - PARAMETER num_gpu：指定加载到GPU的层数。该参数根据硬件配置调整，例如单卡4090 GPU可以设置为7层。
  - PARAMETER num_ctx：指定生成的最大token数。该参数决定了模型推理时的最大上下文长度。
  - PARAMETER temperature：指定模型温度参数，用于控制生成结果的随机性。
  - TEMPLATE：指定模型提示词模板，用于定义用户输入和模型输出的格式。

方案三、Unsloth动态量化+Ollama

注册模型到Ollama

- 使用以下命令将合并后的模型注册到Ollama中：

```
ollama create DeepSeek-R1-UD-IQ1_M -f DeepSeekQ1_Modelfile
```

- 该命令将模型DeepSeek-R1-UD-IQ1_M注册到Ollama，并加载配置文件DeepSeekQ1_Modelfile。
- 注册完成后，可以通过以下命令查看模型是否成功注册：

```
ollama list
```

- 如果模型注册成功，Ollama将显示模型名称、路径和相关参数。注册过程中需要注意以下几点：
  1. 确保配置文件路径正确无误。如果路径错误，Ollama将无法加载模型。
  2. 如果模型已存在，可以使用**--overwrite**选项覆盖旧模型：

```
ollama create DeepSeek-R1-UD-IQ1_M -f DeepSeekQ1_Modelfile --overwrite
```

- 确认无误后即可运行模型

```
ollama run DeepSeek-R1-UD-IQ1_M --verbose
```

方案三、Unsloth动态量化+Ollama

运行模型并验证性能

- 模型注册完成后，即可通过Ollama运行模型并验证其性能。运行命令如下：

```
ollama run DeepSeek-R1-UD-IQ1_M --verbose
```

- 在运行过程中，可以观察模型的推理速度、吞吐量以及资源占用情况。
  - 例如，在单卡4090 GPU上，推理速度可达6 tokens/s，而在双卡A100服务器上，纯GPU推理速度可达20 tokens/s。
- 性能验证不仅包括速度指标，还应关注模型的准确性和稳定性。可以通过以下方式验证模型性能：
  - 推理速度测试：通过输入简单的提示词，测试模型的响应时间。
  - 吞吐量测试：在多并发场景下，测试模型的吞吐量和资源占用情况。
  - 准确性测试：通过预定义的测试集，验证模型生成结果的准确性。
- 如果性能未达到预期，可以调整模型配置文件中的参数，例如增加GPU层数或调整温度参数。

PART 04 ▸

DeepSeek一体机

Enterprise deployment

- DeepSeek 一体机是融合 “算力 + 大模型 + 应用” 的创新产品。它的硬件配置强劲，配置高性能CPU和GPU、海量高速内存与固态硬盘，可高效处理复杂计算与大规模数据，进行大模型的推理甚至微调和训练。
- 对一般企业而言，它提供一站式服务，“软硬协同、本地化部署”，降低智能化转型门槛，保障数据安全。对个人用户，其低成本让大模型使用更亲民，有效地降低了AI技术的使用门槛。推动 AI 技术普及，助力各行业利用 AI 提升效率。
- 这部分将以北大青鸟用于高校AI通识课教育的AI实验室中的DeepSeek一体机为例，展示DeepSeek一体机的配置、性能数据及报价参考等，且深度分析业务场景的适配性，给予参考帮助。

国产DeepSeek一体机厂商一览表

北京大学 PEKING UNIVERSITY

截止至2025年2月（不包含北大青鸟DeepSeek一体机，后续会详细介绍）

- 尽管一体机普遍宣传支持 “满血版”DeepSeek，但实际效果受算力卡性能限制，V3/R1 模型推荐 FP8 和 BF16 推理精度。
- 第三方运营方为降低成本普遍采用：BF16 权重转换（占比约 65%）和INT8 量化方案（占比约 30%），导致用户实测效果比DeepSeek官方原版低 15-25%。
- 选购建议： 要求提供官方 FP8 兼容性认证，实测复杂场景下的响应速度与准确率

- 三种精度方案对比
  1. 最佳方案：原生支持 FP8 精度的 GPU，实现 100% 满血推理效果
  2. 次优方案：BF16 精度需自行转换模型权重，精度接近无损但系统开销增加，推理效率降低约 20-30%
  3. 较差方案（残血版）：量化为 INT8/INT4 模型，推理效率提升 3-5 倍，模型精度损失达 40-60%

新华三	旗舰版（671B满血）、标准版（32B、70B）、经济版（32B及以下），可选全套AI工具链及灵犀使能平台、内置多种智慧应用等
联想	预装DeepSeek不同尺寸模型的AI工作站和AI一体机，面向智能体开发和公文写作
浪潮信息	基础版、标准版、高级版、沐曦版、集群版多种规格，适配不同尺寸DeepSeek模型，提供全套AI工具链、数据安全保障
中科曙光	多硬件规格，支持从10亿级参数模型推理到1000亿级参数模型训练，内置AI工具链和模型管理平台（并支持对外贴牌OEM）
超聚变	旗舰版（671B满血）、标准版（32B、70B）、轻量版（14B以下），可选全套AI工具链
华为	FusionCube A3000训推超融合满血Ultra版、蒸馏Pro版、蒸馏Lite版。并支持生态伙伴推出各种昇腾版DeepSeek一体机
宁畅	旗舰版（671B）、专业版（70B）、标准版（14B/32B）、轻量版(14B及以下)，集成常用数据集及AI工具链
超云	高级版（DS671B、Llama405B）、企业版（70B）、基础版（32B及以下），集成算法、调度、运维等组件
中兴通讯	AiCube训推一体机，支持DeepSeek R1全系列蒸馏模型，提供大模型开发工具链
京东云	vGPU智算一体机，包括满血版、极致性价比版、轻量版，内置智能体+知识库双引擎、方案模板和插件

大华	支持多规格，从7B到满血版（满血需4台组集群）
神州鲲泰	推出多种规格一体机，适配满血版及蒸馏版推理，并联合焱融存储推出“训推一体”方案
百度智能云	百舸一体机，支持昆仑芯P800单机8卡满血版DeepSeek；千帆一体机，四种规格，并提供全栈工具链
浪潮云	预置满血版DeepSeek V3和R1的海若一体机，包括启航版、进阶版、旗舰版，并提供全套工具链
威努特	满血版（4台集群）、蒸馏版（单台多种规格），鲲鹏、海光、英特尔全覆盖
天融信	DeepSeek安全智算一体机
中国移动	移动云智算一体机DeepSeek版，基于移动云边缘智能小站打造，全栈国产化，一云多芯，安全可信，预置DeepSeek模型镜像、工具链
中国电信	天翼云息壤智算一体机DeepSeek版，全栈国产化（昇腾+鲲鹏），适配DeepSeek系列模型，提供工具链
中国联通	联通云DeepSeek一体机，适配国产算力芯片，预置包含满血版DeepSeek在内的多尺寸模型，内置安全体系
无问芯穹	满血DeepSeek-R1多并发一体机，支持联网搜索，支持国产和定制化硬件

DeepSeek国产一体机671B推荐配置

北京大学 PEKING UNIVERSITY

基础模型精度：FP8

配置1	配置2	配置3
平台: 昇腾910B 800I A2 整机*2	平台: H20整机*1	平台: 海光K100-A1整机*2
NPU：64G显存NPU模组	GPU：NVIDIA 141GB H20-8GPU模组	GPU：海光DCU K100-A1 (64GB)

基础模型精度：FP16

配置1	配置2	配置3
平台: 昇腾910B 800I A2 整机*4	平台: H20整机*2	平台: AMD-MI300X整机*1
NPU：64G显存NPU模组	GPU：NVIDIA 141GB H20-8GPU模组	GPU：AMD MI300X GPU模组 (192GB)

DeepSeek一体机数据分享

北京大学 PEKING UNIVERSITY

DeepSeek 32B与70B-性能看板

1个并发按照10个用户计算（假设访问的时候只有十分之一的时间在使用模型生成）

32B

- 8卡4090（24G显存）
- DeepSeek R1 32B-4K，支持24并发，240名用户常规访问
- DeepSeek R1 32B-8K，支持12并发，120名用户常规访问

- 8卡5090（32G显存）
- DeepSeek R1 32B-4K，支持32并发，320名用户常规访问
- DeepSeek R1 32B-8K，支持16并发，160名用户常规访问

- 8卡L40S（48G显存）
- DeepSeek R1 32B-4K，支持48并发，480名用户常规访问
- DeepSeek R1 32B-8K，支持24并发，240名用户常规访问

70B

- 8卡4090（24G显存）
- DeepSeek R1 70B-4K，支持10并发，100名用户常规访问
- DeepSeek R1 70B-8K，支持5并发，50名用户常规访问

- 8卡5090（32G显存）
- DeepSeek R1 70B-4K，支持12并发，120名用户常规访问
- DeepSeek R1 70B-8K，支持6并发，60名用户常规访问

- 8卡L40S（48G显存）
- DeepSeek R1 70B-4K，支持20并发，200名用户常规访问
- DeepSeek R1 70B-8K，支持10并发，100名用户常规访问

北大青鸟AI实验室案例分享

北京大学 PEKING UNIVERSITY

企业场景

当前院校在大力推动人工智能专业建设和相关科研课题，但是在具体实施操作，会面临着很多问题，如教学资源不足，科研效率低，学生学习体验不佳，没有AI实训平台以及基础算力设施作为支撑等等，北大青鸟为院校提供AI实验室建设方案（DeepSeek一体机），以支撑各专业方向的教学和科研方面的创新工作。

北大青鸟AI实验室建设方案

北京大学 PEKING UNIVERSITY

应用层	模型训推 自然语言 图像处理 语言识别 文字识别 深度学习
监控层	动态监控 资源监控 集群监控 全景大屏 状态信息 资源使用 任务运行
业务层	管理功能：任务、资源、镜像、存储、数据、用户、运维 TensorFlow PyTorch mxnet
容器层	动态拓展、镜像存储、隔离策略、高可用、单机多卡、多机多卡 docker kubernetes
系统层	操作系统：Centos/ Ubuntu 分布式/并行文件系统
硬件层	计算节点 管理节点 存储节点 网络节点 

在实际应用中，传统部署方案（如vllm、Tensor RT等工具）常面临硬件适配复杂、资源利用率低等问题。为此，AI实验室提供开箱即用的DeepSeek一体机全栈解决方案：深度调优，集成预训练模型与动态调度引擎，兼容多场景推理与训练任务，以更低部署成本、更高响应效率及数据本地化安全保障，助力企业快速实现DeepSeek大模型规模化应用。

01

一键部署

- 内置主流模型管理工具，实现快速部署模型
- 采用容器化，自动化脚本，使模型部署高效可重复
- 一键运行DeepSeek-R1大模型，提供多版本模型（1.5B-671B）

02

Deepseek

- 快速发布服务，调用API实现应用对话
- 支持用户基于DeepSeek大模型构建本地数据库
- 优化算法结构，高效处理数；满足不同用户场景

03

一体机

- 模型训练推理场景集中式一体化
- 任务管理，系统资源，监控运维的可视化
- 多种资源结构集中管理，GPU调度切分

北大青鸟AI实验室（DeepSeek一体机）

DeepSeek基础版 7B

基础模型精度：FP16

类别	规格描述	部件数
平台	1、4U4卡机架式服务器 2、支持2颗Intel第4代CPU 3、16根DDR5内存，最高4800MHz，最大4TB 4、8块3.5/2.5，不支持NVME，支持2块NVME协议M.2	x1
CPU	INTEL 5418Y Sapphire Rapids/24C/48T/2.0GHz/45MB/185W/4400MHz	x2
内存	32GB/RECC/DDR5/4800MHz	x4
系统盘	960G/2.5寸/SATA/1DWPD	x2
数据盘	1.92T/2.5寸/U.2 NVME/1DWPD	x1
GPU	RTX4090 24GB/GDDR6X/PCIE/450W/双宽/16PIN/主动	x2

北大青鸟AI实验室（DeepSeek一体机）

DeepSeek标准版 32B

基础模型精度：FP16

类别	规格描述	部件数
平台	1、4U4卡机架式服务器 2、支持2颗Intel第4代CPU 3、16根DDR5内存，最高4800MHz，最大4TB 4、8块3.5/2.5，不支持NVME，支持2块NVME协议M.2	x1
CPU	INTEL 5418Y Sapphire Rapids/24C/48T/2.0GHz/45MB/185W/4400MHz	x2
内存	32GB/RECC/DDR5/4800MHz	x8
系统盘	960G/2.5寸/SATA/1DWPD	x2
数据盘	3.84T/2.5寸/U.2 NVME/1DWPD	x1
GPU	RTX4090 24GB/GDDR6X/PCIE/450W/双宽/16PIN/主动	x4

北大青鸟AI实验室（DeepSeek一体机）

DeepSeek旗舰版 671B

基础模型精度：FP16

类别	规格描述	部件数
平台	NF5688M7 1.LSI 9560-8i(4G)Raid卡不带电容 *1 2.Mellanox CX7400G单光口HCA卡(不带模块)*8 3.自研X710 10G双光口网卡(带模块)*1 4.3200W铂金电源 *2 5.2700w铂金电源 *6/ 导轨 / 国标16A电源线/3年NBD	x1
CPU	Intel 8480+(56C，2.0GHz)*2	x2
内存	64G 4800MHz DDR5	x24
系统盘	960GB SATA SSD	x2
数据盘	3.84T/2.5寸/U.2 NVME/1DWPD	x4
GPU	GPU Nvidia HGX-H20-8GPU	x1
	整机台数	x2

北大青鸟AI实验室（DeepSeek一体机）

北京大学 PEKING UNIVERSITY

类别	适用场景	性能	报价
DeepSeek基础版（7B）	文本摘要、多轮对话系统（智能客服）、高精度轻量级任务	并发约15-30；吞吐约10-20请求/秒	8.8万
DeepSeek标准版（32B）	科研与学术的研究支持，专业领域问答和复杂逻辑推理，软件工程与高质量代码生成，企业战略分析与决策等	并发约15-30；吞吐约10-15请求/秒	16.8万
DeepSeek旗舰版（671B）（满血版）	国家级大型AI项目研究、气候建模等；院校算力中心建设	并发约90-190；吞吐约30-60请求/秒	199万

建议报价，有一定的时效性

总结

本次研讨，系统梳理了DeepSeek模型私有化部署的全场景解决方案，从模型选型到落地实践，覆盖个人用户与企业级需求的核心要点。通过对比不同版本模型的性能特点，结合Ollama、vLLM等部署框架的实操演示，帮助大家根据自身算力资源、业务场景和技术能力选择最优部署路径。无论是个人开发者通过轻量化工具快速体验模型能力，还是企业基于高性能推理引擎构建专业服务，抑或在有限资源下实现动态量化部署，或者个人、企业、学校、政府机关选购DeepSeek一体机的参考建议，本次分享均提供了可落地的技术方案和已验证的经验数据，为DeepSeek模型从"能用"到"用好"的跨越提供了完整方法论支持。

参考文档

1. https://github.com/deepseek-ai/DeepSeek-R1
2. https://github.com/kvcache-ai/ktransformers
3. https://github.com/vllm-project/vllm
4. https://github.com/ggml-org/llama.cpp
5. https://github.com/ollama/ollama
6. https://arxiv.org/pdf/2412.19437
7. https://arxiv.org/pdf/2501.12948
8. https://arxiv.org/pdf/2405.04434
9. https://arxiv.org/pdf/2401.02954