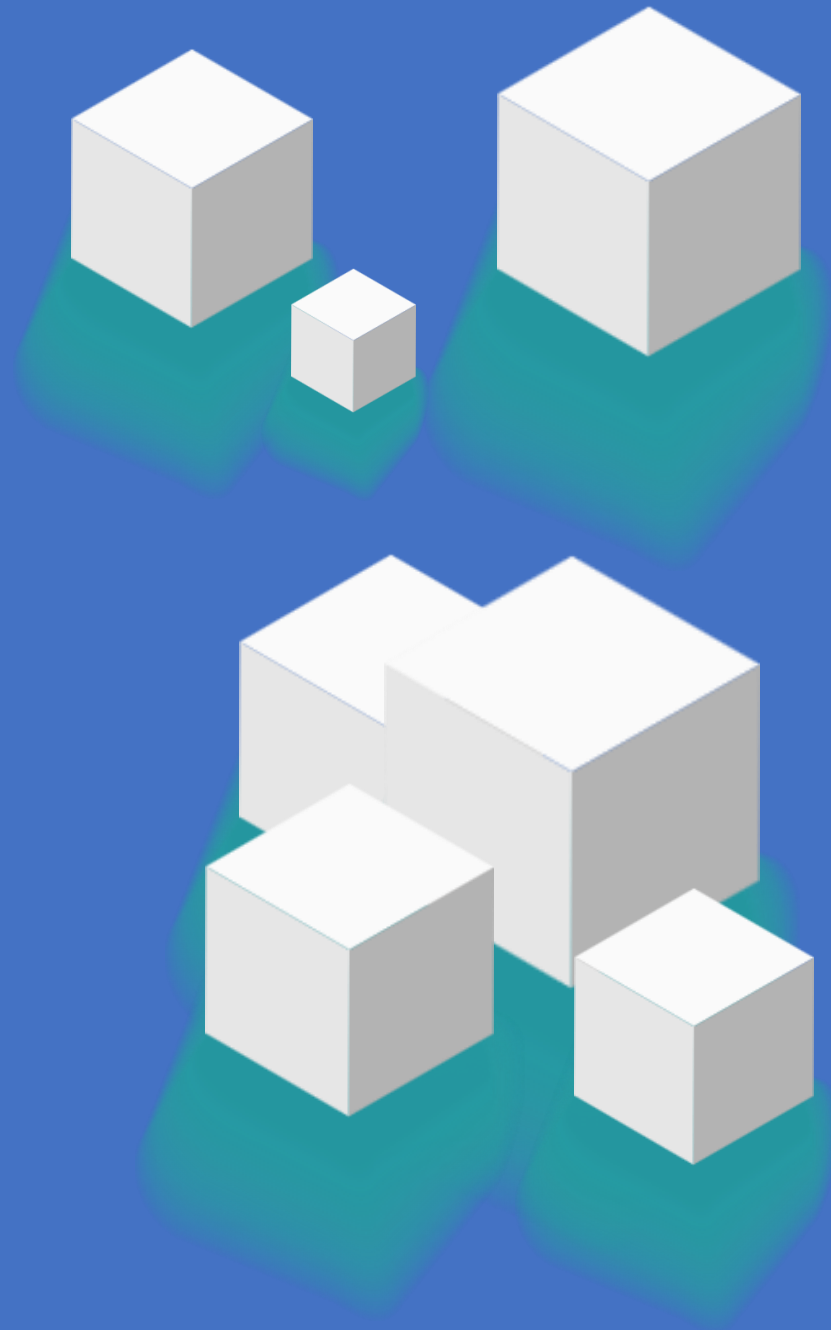
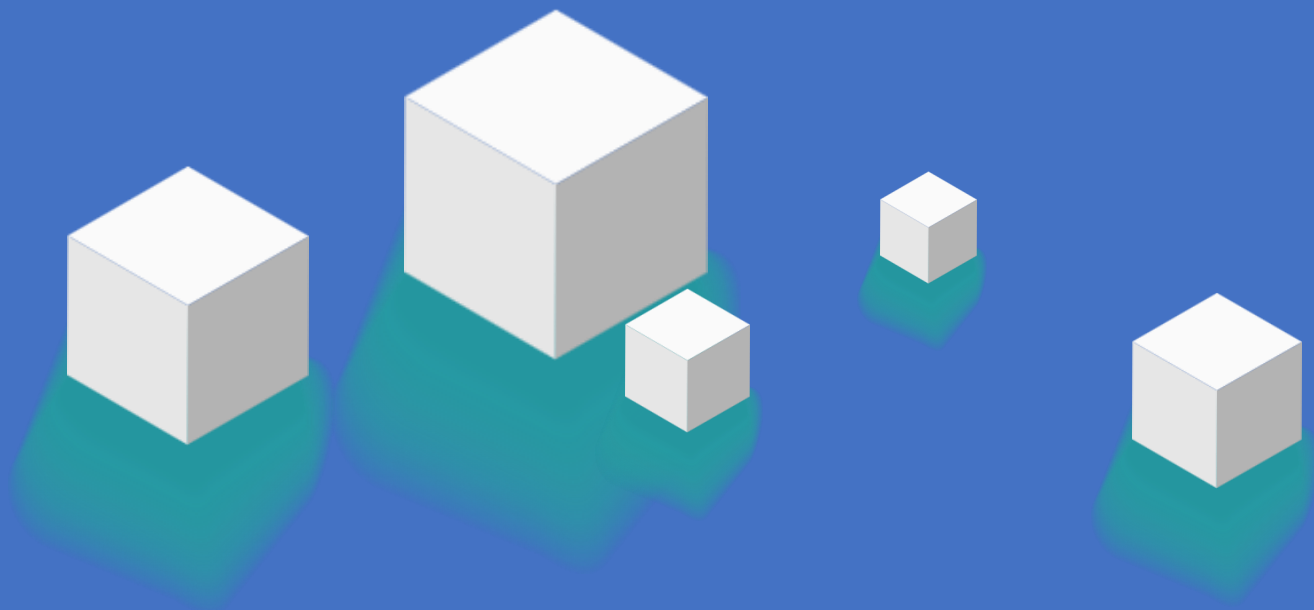


# DeepSeek使用



# >> 今天的学习目标

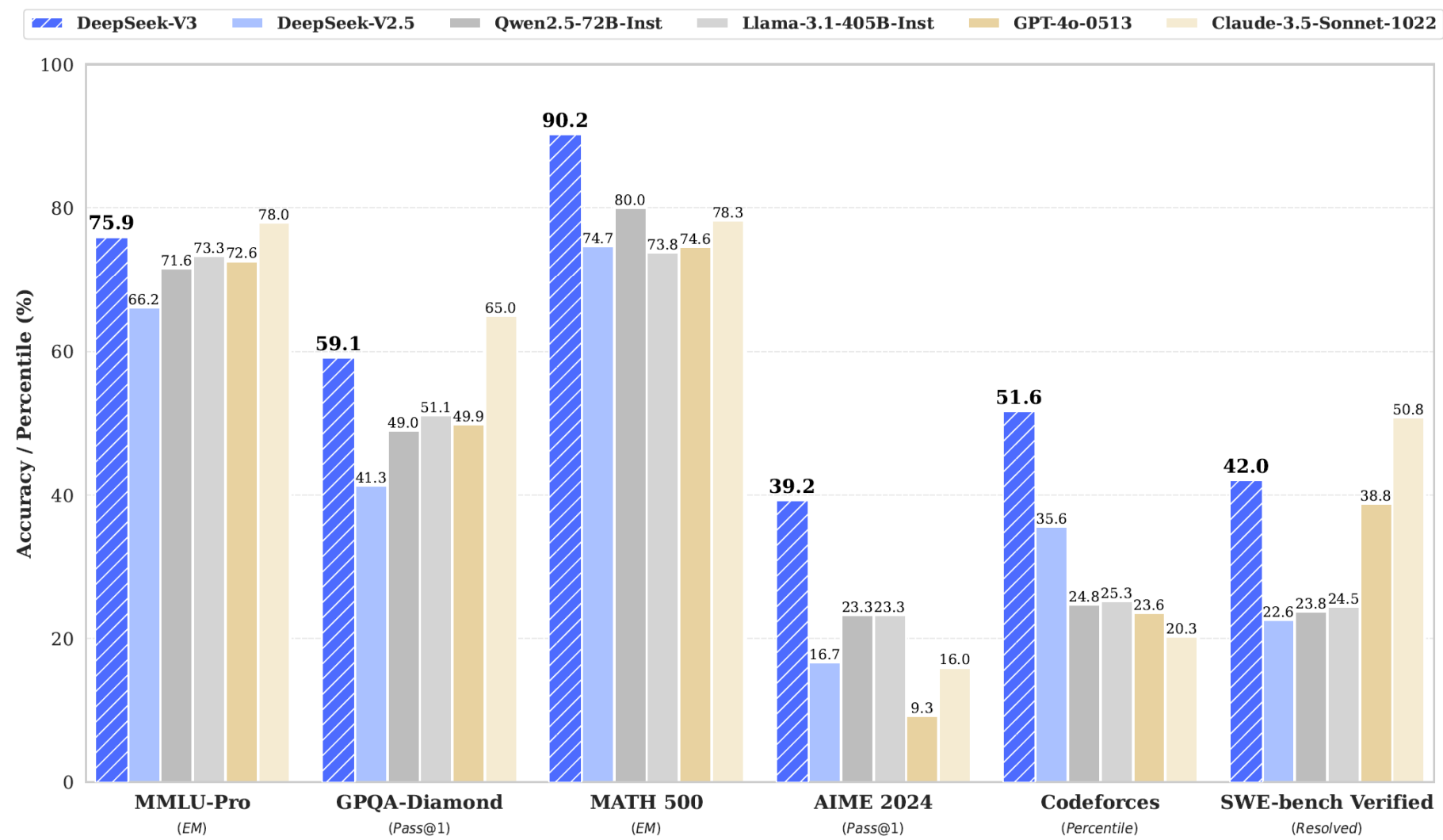
---

## DeepSeek使用

- DeepSeek的创新
- CASE：小球碰撞试验（Cursor + DeepSeek-R1）
- DeepSeek私有化部署选择
- Ollama部署DeepSeek-R1
- API调用DeepSeek

# DeepSeek的创新

# DeepSeek-V3模型



DeepSeek-V3 在推理速度上相较历史模型有了大幅提升。在目前大模型主流榜单中，DeepSeek-V3 在开源模型中位列榜首，与世界上最先进的闭源模型不分伯仲。

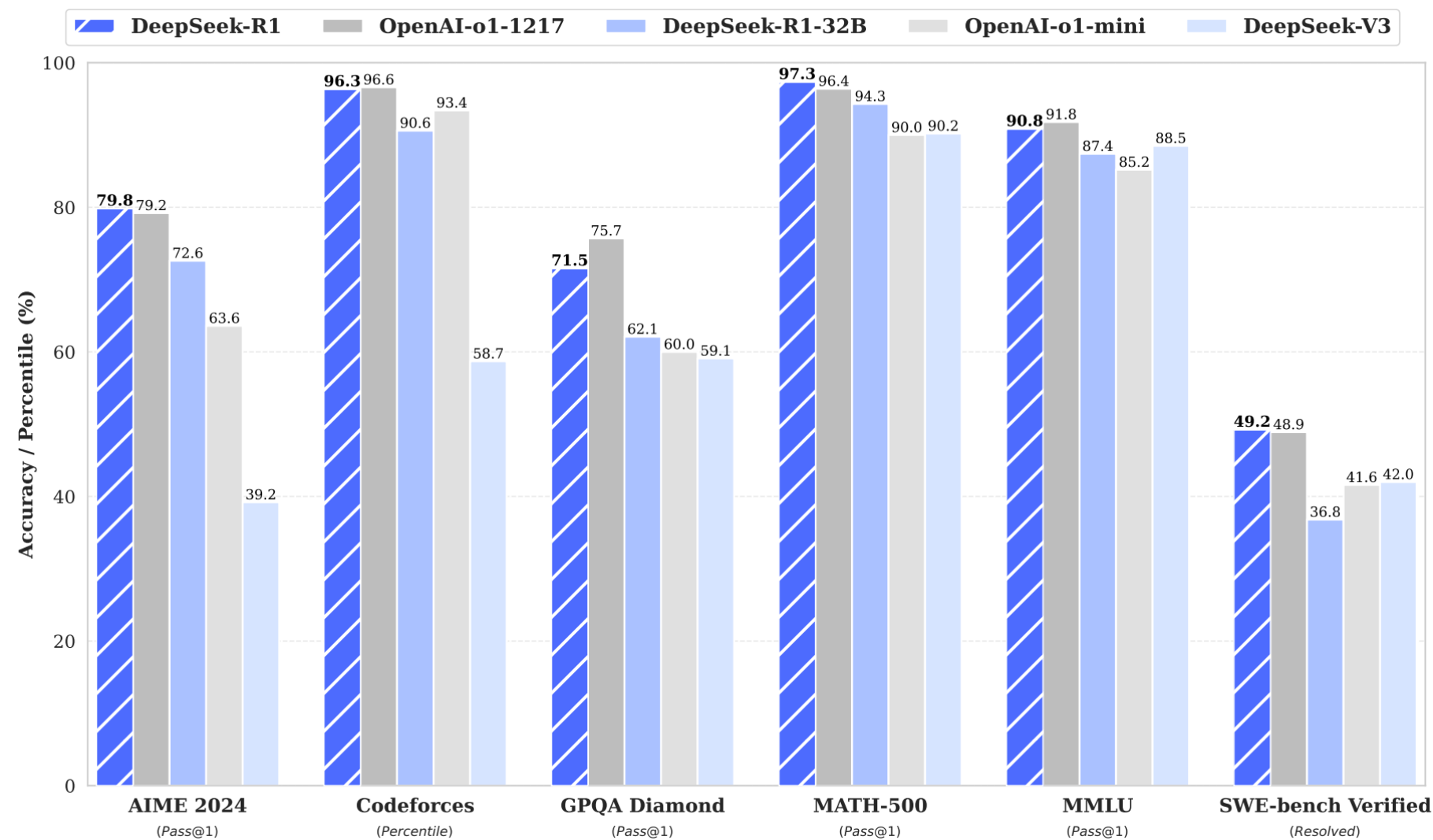
# DeepSeek-V3的训练成本

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	\$5.328M	\$0.238M	\$0.01M	\$5.576M

DeepSeek-V3 的训练成本，假设H800的租用是 \$2/小时

DeepSeek-V3的推出是2024年12月，并没有太大波澜  
DeepSeek-R1火出圈，通过新的奖励机制GRPO (group relative policy optimization)，并使用规则类验证机制自动对输出进行打分。以V3为基础模型，一个多月内训练出了性能堪比GPT-o1的R1模型，成果非常亮眼。

# DeepSeek-R1模型



DeepSeek-R1 完全对标 OpenAI-o1，与之前的DeepSeek-V3有大幅提升

# DeepSeek-R1模型

---

DeepSeek-R1

DeepSeek-R1 遵循 MIT License，允许用户通过蒸馏技术借助 R1 训练其他模型。

DeepSeek-R1 上线 API，对用户开放思维链输出，通过设置 `model='deepseek-reasoner'` 即可调用。

2024-12-26 发布V3

2025-1-15 发布APP

2025-1-20 发布R1

多家企业宣布融合DeepSeek

.....

MIT License是一种非常宽松的开源许可协议，允许用户自由地使用、修改、分发和商业化软件或模型。相比之下，Meta Llama的License相对严格，虽然LLaMA3是开源的，但许可协议限制了商业用途和对模型的修改，比如新的模型如果使用 LLaMA，需要名称上带有LLaMA标识。

# 蒸馏小模型超越 OpenAI o1-mini

在开源 DeepSeek-R1-Zero 和 DeepSeek-R1 两个 660B 模型的同时，通过 DeepSeek-R1 的输出，蒸馏了 6 个小模型，其中 32B 和 70B 模型在多项能力上实现了对标 OpenAI o1-mini 的效果。

	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	759.0
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717.0
o1-mini	63.6	80.0	90.0	60.0	53.8	1820.0
QwQ-32B	44.0	60.0	90.6	54.5	41.9	1316.0
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954.0
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189.0
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481.0
DeepSeek-R1-Distill-Qwen-32B	72.6	83.3	94.3	62.1	57.2	1691.0
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205.0
DeepSeek-R1-Distill-Llama-70B	70.0	86.7	94.5	65.2	57.5	1633.0



# DeepSeek的创新： MLA

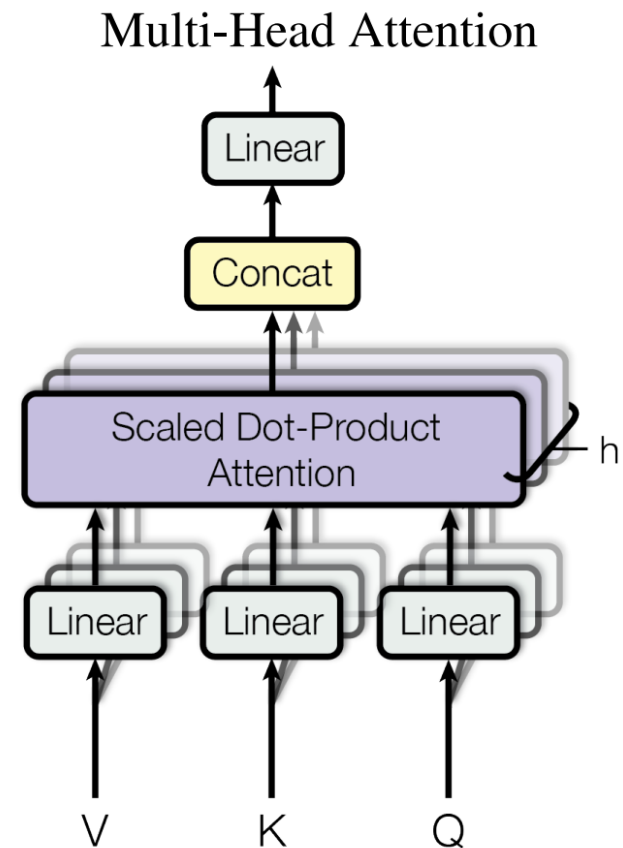
- MLA (Multi-Head Latent Attention)

在“All you need is attention”的背景下，传统的多头注意力（MHA, Multi-Head Attention）的键值（KV）缓存机制事实上对计算效率形成了较大阻碍。缩小KV缓存（KV Cache）大小，并提高性能，在之前的模型架构中并未很好的解决。

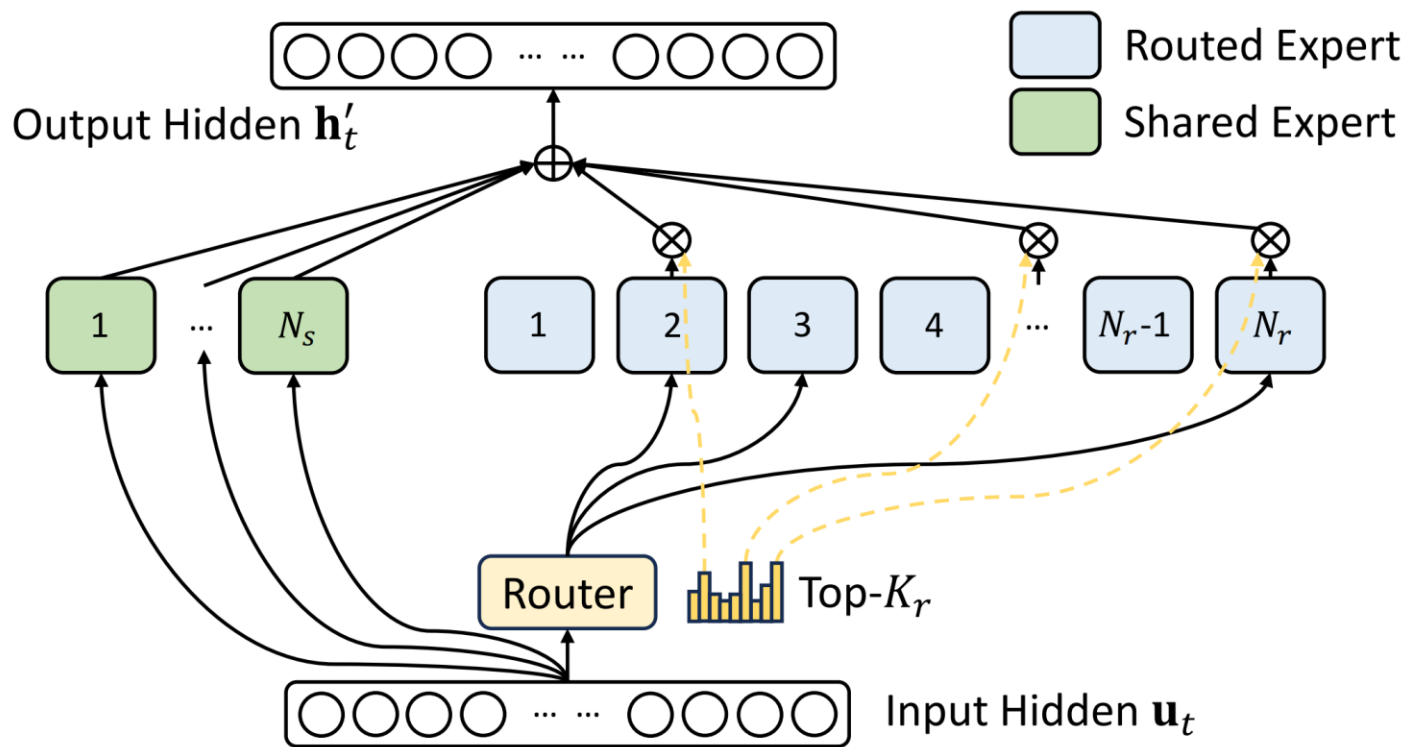
DeepSeek引入了MLA，一种通过低秩键值联合压缩的注意力机制，在显著减小KV缓存的同时提高计算效率。

低秩近似是快速矩阵计算的常用方法，在MLA之前很少用于大模型计算。

从大模型架构的演进情况来看，Prefill和KV Cache容量瓶颈的问题正一步步被新的模型架构攻克，巨大的KV Cache正逐渐成为历史。（实际上在2024年6月发布的DeepSeek-V2就已经很好的降低了KV Cache的大小）



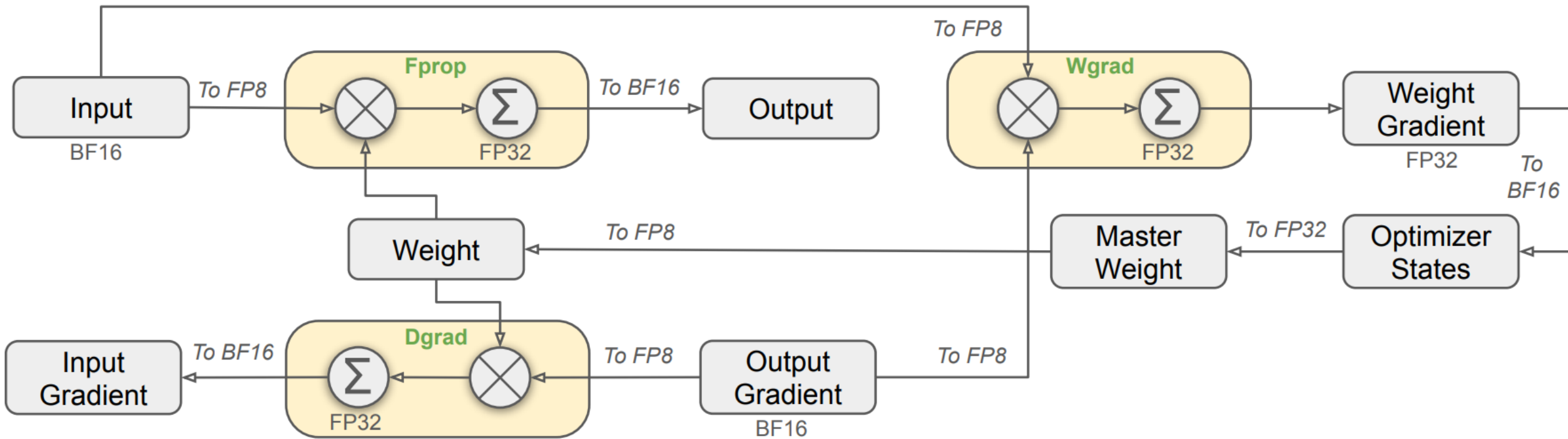
# DeepSeek的创新: DeepSeek-MoE



V3使用了61个MoE (Mix of Expert 混合专家) block, 虽然总参数量很大, 但每次训练或推理时只激活了很少链路, 训练成本大大降低, 推理速度显著提高。

**MoE类比为医院的分诊台**, 在过去所有病人都要找全科医生, 效率很低。但是MoE模型相当于有一个分诊台, 将病人分配到不同的专科医生那里。DeepSeek在这方面也有创新, 之前分诊是完全没有医学知识的保安, 而现在用的是有医学知识的本科生来处理分流任务

# DeepSeek的创新：混合精度框架



整个混合精度框架使用了FP8数据格式，但为了简化说明，只展示了线性算子（Linear Operator）的部分

采用了混合精度框架，即在不同的区块里使用不同的精度来存储数据。我们知道精度越高，内存占用越多，运算复杂度越大。DeepSeek在一些不需要很高精度的模块，使用很低的精度FP8储存数据，极大的降低了训练计算量。

# Summary

---

Thinking: 为什么DeepSeek计算速度快，成本低？

- 架构设计方面

DeepSeek MoE架构：在推理时仅激活部分专家，避免了激活所有参数带来的计算资源浪费。

MLA架构：MLA通过降秩KV矩阵，减少了显存消耗。

- 训练策略方面

多token预测（MTP）目标：在训练过程中采用多token预测目标，即在每个位置上预测多个未来token，增加了训练信号的密度，提高了数据效率。

混合精度训练框架：在训练中，对于占据大量计算量的通用矩阵乘法（GEMM）操作，采用FP8精度执行。同时，通过细粒度量化策略和高精度累积过程，解决了低精度训练中出现的量化误差问题。

# Summary

---

Thinking: 为什么DeepSeek-R1的推理能力强大?

- **强化学习驱动**: DeepSeek-R1通过大规模强化学习技术显著提升了推理能力。在数学、代码和自然语言推理等任务上表现出色, 性能与OpenAI的o1正式版相当。
- **长链推理 (CoT) 技术**: DeepSeek-R1采用长链推理技术, 其思维链长度可达数万字, 能够逐步分解复杂问题, 通过多步骤的逻辑推理来解决问题

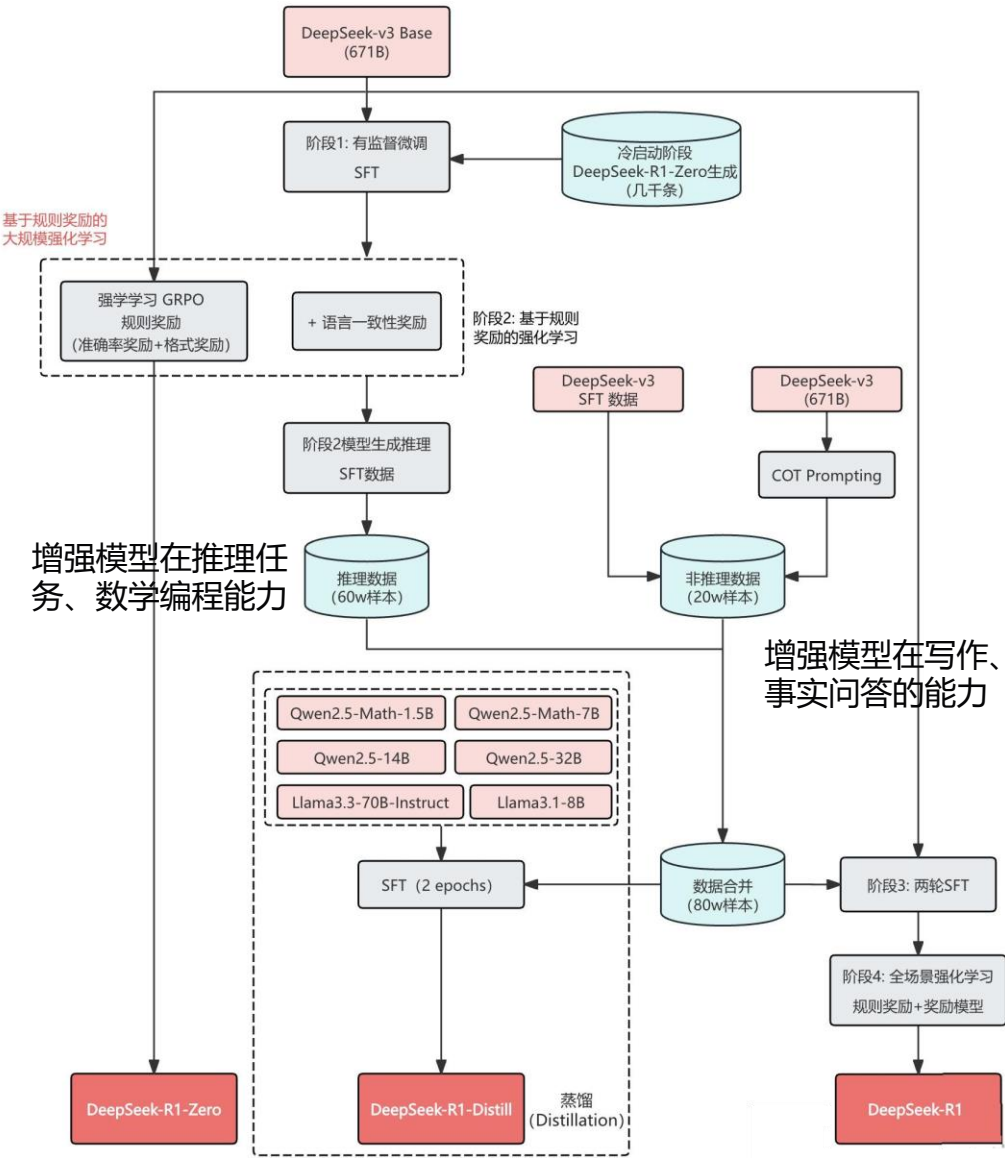
强化学习的作用: 训练大模型, 结合少量SFT, 引入少量高质量监督数据 (如数千个CoT示例) 进行微调, 提升模型初始推理能力, 再通过RL进一步优化, 最终达到与OpenAI o1相当的性能

长链推理CoT: CoT让AI模型逐步分解复杂问题, 比如在智能客服、市场分析报告、AI辅助编程领域

# Summary

模型	方法
DeepSeek-R1-Zero	纯强化学习
DeepSeek-R1	冷启动 SFT -> RL -> COT + 通用数据 SFT (80w) ->全场景 RL
蒸馏小模型	直接用上面的 80w 数据进行SFT

DeepSeek-R1-Zero首次验证了纯强化学习在 LLM 中能显著增强推理能力的可行性，即无需SFT，仅通过 RL 即可激励模型学会长链推理和反思。提出了多阶段训练策略（冷启动->RL->SFT->全场景 RL），有效兼顾准确率与可读性，产出 DeepSeek-R1，性能比肩 OpenAI-o1-1217。展示了知识蒸馏在提升小模型推理能力方面的潜力，并开源多个大小不一的蒸馏模型（1.5B~70B）



# DeepSeek + Cursor使用： 物理世界中的小球碰撞

# DeepSeek + Cursor使用

在 File -> Preferences -> Cursor Settings 中设置

deepseek-r1 和 deepseek-v3模型

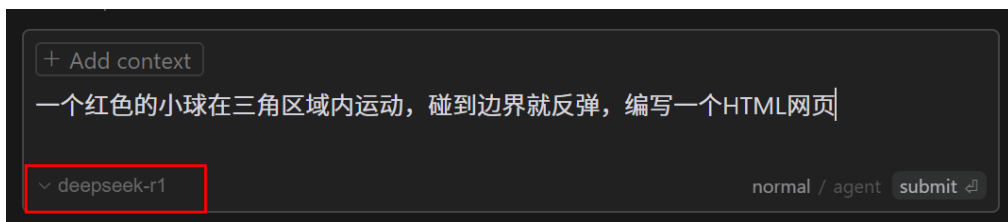
在OpenAI API Key中进行设置，这里是采用OpenAI的协议，可以使用自定义的模型

OpenAI Key = sk-

Q2gN9CgZOz9jrjCCHkijalkUyaXpHS6xssmmkl327kkib0G

OpenAI Base URL = <http://chatapi.littlewheat.com/v1>

设置好 deepseek-r1 和 deepseek-v3模型之后，可以在Cursor中选择该模型进行编程



## Model Names

Add new models to Cursor. Often used to configure the latest OpenAI models or OpenRouter models.

- ☐ claude-3.5-sonnet
- ☐ cursor-small
- ☒ deepseek-r1
- ☒ deepseek-v3

## OpenAI API Key

With your Cursor Pro subscription, you do not need to use your own OpenAI key unless you're using custom models.

Dismiss [Turn Off OpenAI Key](#)

You can put in [your OpenAI key](#) to use Cursor at public API costs. Note: this can cost more than pro and won't work for custom model features.

..... [Verify →](#)

Override OpenAI Base URL ▾

<http://chatapi.littlewheat.com/v1>

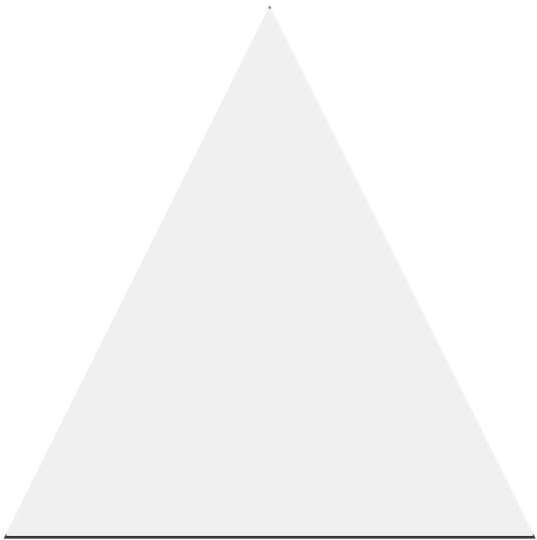
[Save](#)

[Reset to default](#)



# CASE：物理世界中的小球碰撞

一个红色的小球在三角区域内运动，碰到边界就反弹，  
编写一个HTML网页



deepseek-r1 会进行很长时间的思考，然后编写出HTML代码。  
编写的HTML代码运行可能不符合要求，需要进一步调整

<think>

好的，我现在需要帮助用户创建一个在三角区域内反弹的小球的HTML页面。首先，我得考虑如何实现这个效果。用户想要一个红色小球在三角形内运动，碰到边界反弹。我需要用HTML5的Canvas来绘制图形和动画。

首先，我得确定三角形的边界。三角形通常有三个边，但检测碰撞可能会比较复杂。因为Canvas的路径是矢量，可能需要用数学方法来判断小球是否碰到三角形的边。或者，我可以将三角形视为三个线段，分别检测碰撞。

接下来是小球的运动。小球有位置 (x,y) 和速度 (dx,dy)，每次动画帧更新位置。当碰到边界时，速度方向改变。但是，三角形的边界是斜边，所以反弹方向不能简单取反，需要计算反射向量。

这可能会比较复杂。用户可能希望简化处理，比如将三角形视为三个墙壁，每次碰撞后根据碰到的边反转速度的相应分量。或者，可能需要计算小球到各边的距离，判断是否发生碰撞，然后调整速度方向。

考虑到用户可能希望代码简单，我先考虑一个等边三角形，顶点在上方，底边水平。这样三条边分别是左斜边、右斜边和底边。对于左右斜边，碰撞检测可能需要计算点是否在边的附近，并调整速度方向。

但这样做可能数学计算较多。或者，可以使用三个直线方程，检测小球的中心是否接近某条边，并计算反射方向。

另一个思路是使用边界框，将三角形的三个边作为反射面，当小球碰撞时，根据边的方向调整速度。例如，底边是水平的，碰撞时反转dy；左右边是斜的，碰撞时需要计算反射。

这可能涉及到向量运算，比如计算反射向量。假设边的法线向量，然后根据入射速度计算反射速度。

不过，这样计算起来可能有点复杂，特别是对于初学者。有没有更简单的方法？比如，将三角形视为由三个线段组成，每次检测小球与这三个线段的碰撞，然后反弹。

或者，使用边界的斜率和小球的运动方向来判断碰撞。例如，当小球碰到左斜边时，根据边的斜率调整速度分量。

不过，这样可能需要较多的几何计算。为了简化，我可以假设三角形是一个等边三角形，三个边的方程已知，然后检测小球的中心是否接近某条边，并在碰撞时调整速度方向。

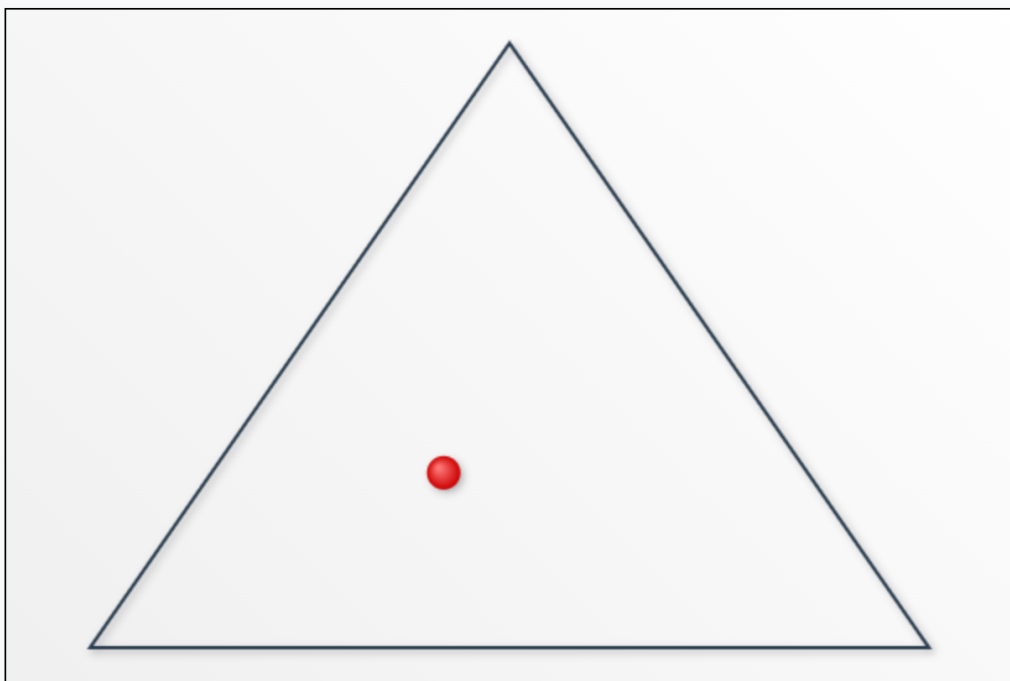
现在，我需要考虑如何表示三角形的边。例如，左斜边可能是从左上角到右下角，右斜边是从右上角到右下角，底边是水平的。假设三角形的顶点在画布的上中点，底边在底部。

接下来，小球的初始位置应该在三角形内部。当小球移动时，检查它是否超出了三角形的任何一边。如果超出，则反弹。

但如何判断点是否在三角形内？或者，当小球的中心碰到边时，视为碰撞。这可能需要进

# CASE：物理世界中的小球碰撞

基于之前的思考，帮我完善HTML



红色小球在三角形区域内持续反弹运动

HTML样式有了调整，但是小球还是会飞出三角区域，需要进一步调整

<think>

好吧，用户希望完善之前的HTML，但当前的代码已经实现了基本功能。首先，我需要仔细检查代码，看看哪些地方可以改进。注意到用户提到之前的功能是正确的，所以不能破坏现有逻辑。

首先，界面方面，原生的canvas边框比较单调，可以考虑添加背景色或者渐变效果，让画面更吸引人。比如给画布加个渐变背景，这样视觉效果更好。

然后，动画流畅度方面，当前的移动速度是固定的，用户可能想要更自然的运动。可以考虑调整速度参数，比如使用小数，让运动更平滑。或者添加随机的速度变化，但要注意不能改变原有反弹逻辑。

界面文字提示也是一个改进点。可以在页面下方添加说明文字，比如使用CSS在body中添加一个div，说明这是三角区域反弹球，这样用户更清楚演示内容。

另外，小球的外观可以优化，比如添加阴影效果或者高光，让它看起来更立体。在drawBall函数里，用ctx.shadow来增加立体感。

代码可读性方面，可以添加更多注释，尤其是碰撞检测的部分，虽然用户可能已经了解，但清晰的注释有助于维护。比如在checkCollisionWithEdge函数中解释参数A和B代表边的两个端点。

检查是否存在中文乱码，确保HTML的meta标签有设置UTF-8。原来的代码没有<meta charset>，需要补充，这样中文注释和可能的文本就不会乱码。

最后，考虑响应式布局，让画布在不同设备上居中显示。可以通过CSS的flex布局让canvas居中，提升用户体验。

综合这些点，逐步修改代码，确保每一处改动都不影响原有功能。比如添加meta标签，修改背景样式，增加文字提示，美化小球外观，并调整速度参数。同时保持代码结构清晰，注释准确。

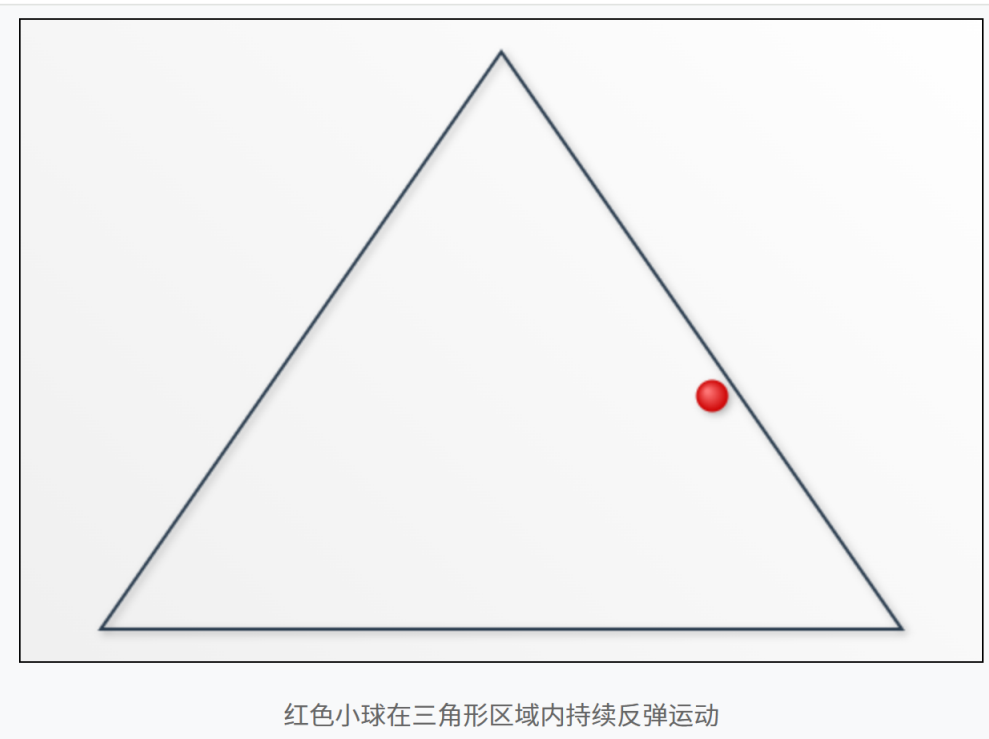
</think>

我将在原有功能不变的基础上做以下改进优化：

```
<?index.html +168 -5 ✓>
<!DOCTYPE html>
<html>
<head>
<style>
```

# CASE：物理世界中的小球碰撞

小球弹了之后，弹出去了啊，帮我检查代码



直接告诉deepseek-r1遇到的错误，修正后，小球会在三角区域内进行运动

<think>

好的，用户反馈说小球碰撞后弹出去了，我需要仔细检查代码中的碰撞检测逻辑。首先，查看checkCollisionWithEdge函数，注意到计算法线向量的方式可能有问题。

原算法线方向的部分，使用的是最近的点的坐标减去起点A的坐标，再除以边长度，但这种计算是否垂直于边呢？假设边的向量是 (B.x - A.x, B.y - A.y)，那么与其垂直的向量应该是 (B.y - A.y, -(B.x - A.x)) 或者相反方向。原代码中计算normal的x分量是(nearest.y - A.y)/edgeLength，这等于边的y分量除以长度，这可能错误地将边的方向当成了法线方向。

正确的法线应该使用边的垂直向量。比如，边向量是 (edgeX, edgeY)，则法线应该是 (edgeY, -edgeX) 方向，并进行单位化。原代码中使用的是从最近点减去A点后的处理，这可能导致了法线方向错误，导致反射后的速度方向不正确，出现弹出问题。

需要修改normal的计算方式，使用边向量取垂直向量，确保法线方向正确，从而正确反射速度，这样小球就不会弹出边界了。

</think>

经过检查，碰撞检测中的法线方向计算存在微小误差。我们需要修正法线向量计算方式：

<> index.html +139 -2 •

```
<!DOCTYPE html>
<html>
<head>
  <style>
    canvas {
      border: 1px solid #000;
      background: linear-gradient(45deg, #f0f0f0, #ffffff);
    }
  </style>
</head>
</html>
```

问题原因与修正说明：

1. 原法线方向计算错误：

- 旧代码使用的计算方式会导致法线方向与实际边向量方向不一致
- 修正后直接通过边向量计算法向量，确保方向准确

2. 反射公式正确性验证：

- $normal.x = edge.y / edgeLength$  和  $normal.y = -edge.x / edgeLength$

# CASE：物理世界中的小球碰撞

考虑小球的重力，以及三角区域的支撑力，小球的弹力 进行完善

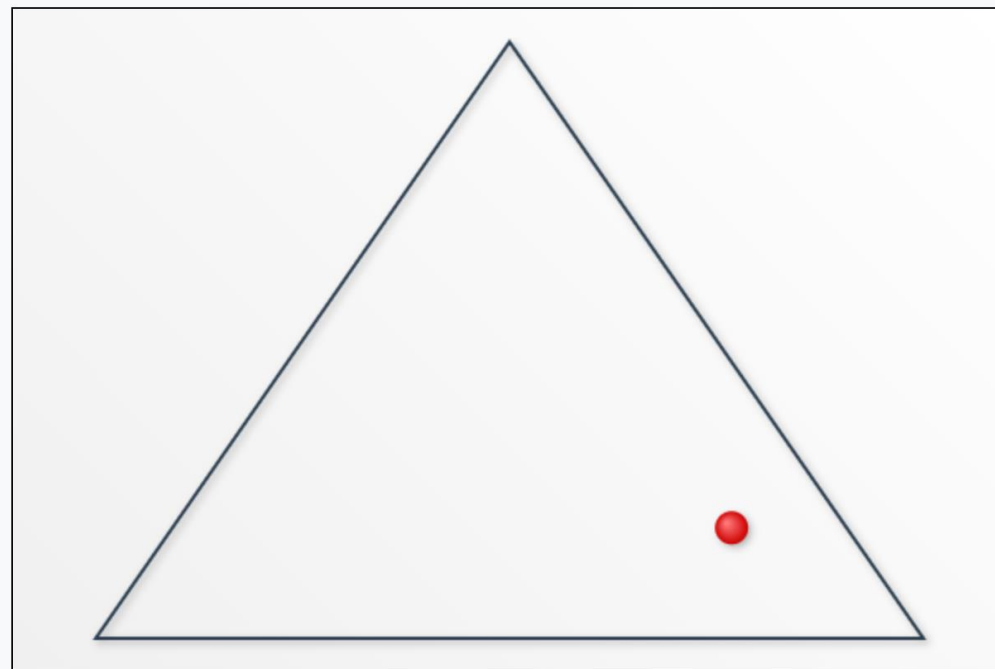
每次运行，小球可以从随机的任意方向抛出

小球的弹力大一些

最后到地面都垂直弹跳了，不是应该还会滚动么，就是有水平的速度

下面增加一个刷新按钮，可以重新执行这个HTML

中文显示是乱码，另外考虑到手机显示，页面需要做自适应



红色小球在三角形区域内持续反弹运动

重新开始

<http://case.wucaai.com/ball/>

# Summary

---

Thinking: 当我们写程序的时候, 什么时候用 DeepSeek-V3-0324, 什么时候用 DeepSeek-R1?

V3-0324: 日常编程、快速开发、前端代码生成、常规脚本任务。

R1: 数学密集型计算、复杂算法、代码逻辑深度优化、需要推理过程的任务。

=> 更擅长复杂算法实现, 能优化逻辑并减少错误

DeepSeek私有化部署

# 私有化部署（DeepSeek模型）

## DeepSeek-R1模型

Model	#Total Params	#Activated Params	Context Length	Download
DeepSeek-R1-Zero	671B	37B	128K	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1</a>
DeepSeek-R1	671B	37B	128K	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Zero">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Zero</a>

## DeepSeek-R1蒸馏模型

Model	Base Model	Download
DeepSeek-R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B</a>
DeepSeek-R1-Distill-Qwen-7B	Qwen2.5-Math-7B	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B</a>
DeepSeek-R1-Distill-Llama-8B	Llama-3.1-8B	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B</a>
DeepSeek-R1-Distill-Qwen-14B	Qwen2.5-14B	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B</a>
DeepSeek-R1-Distill-Qwen-32B	Qwen2.5-32B	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B</a>
DeepSeek-R1-Distill-Llama-70B	Llama-3.3-70B-Instruct	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Llama-70B">https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Llama-70B</a>

# 私有化部署（DeepSeek模型）

目前开放出来的1.5B、7B、14B等模型是Qwen/llama借助R1推理强化调出来的“蒸馏”版本，不是真正的R1。真正的DeepSeek-R1是671B全量版

deepseek-r1:1.5b——1-2G显存

deepseek-r1:7b——6-8G显存

deepseek-r1:8b——8G显存

deepseek-r1:14b——10-12G显存

deepseek-r1:32b——24G-48显存

deepseek-r1:70b——96G-128显存

deepseek-r1:671b——496GB



<https://modelscope.cn/search?search=deepseek>



# 私有化部署（DeepSeek模型）

Category	Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
	Architecture	-	-	MoE	-	-	MoE
	# Activated Params	-	-	37B	-	-	37B
	# Total Params	-	-	671B	-	-	671B
English	MMLU (Pass@1)	88.3	87.2	88.5	85.2	<b>91.8</b>	90.8
	MMLU-Redux (EM)	88.9	88	89.1	86.7	-	<b>92.9</b>
	MMLU-Pro (EM)	78	72.6	75.9	80.3	-	<b>84</b>
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2	<b>92.2</b>
	IF-Eval (Prompt Strict)	86.5	84.3	86.1	84.8	-	83.3
	GPQA-Diamond (Pass@1)	65	49.9	59.1	60	<b>75.7</b>	71.5
	SimpleQA (Correct)	28.4	38.2	24.9	7	<b>47</b>	30.1
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-	<b>82.5</b>
	AlpacaEval2.0 (LC-winrate)	52	51.1	70	57.8	-	<b>87.6</b>
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92	-	<b>92.3</b>

DeepSeekR1 在英文上表现出色

# 私有化部署（DeepSeek模型）

Category	Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
Code	LiveCodeBench (Pass@1-COT)	33.8	34.2	-	53.8	63.4	<b>65.9</b>
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	<b>96.6</b>	96.3
	Codeforces (Rating)	717	759	1134	1820	<b>2061</b>	2029
	SWE Verified (Resolved)	<b>50.8</b>	38.8	42	41.6	48.9	49.2
	Aider-Polyglot (Acc.)	45.3	16	49.6	32.9	<b>61.7</b>	53.3
Math	AIME 2024 (Pass@1)	16	9.3	39.2	63.6	79.2	<b>79.8</b>
	MATH-500 (Pass@1)	78.3	74.6	90.2	90	96.4	<b>97.3</b>
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-	<b>78.8</b>
Chinese	CLUEWSC (EM)	85.4	87.9	90.9	89.9	-	<b>92.8</b>
	C-Eval (EM)	76.7	76	86.5	68.9	-	<b>91.8</b>
	C-SimpleQA (Correct)	55.4	58.7	<b>68</b>	40.3	-	63.7

DeepSeek-R1 在代码、数学、中文上表现出色

# 私有化部署（DeepSeek模型）

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	26.7	78.3	65	38.9	717
o1-mini	63.6	80	90	60	53.8	<b>1820</b>
QwQ-32B-Preview	44	60	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	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	<b>72.6</b>	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80	89.1	49	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70	<b>86.7</b>	<b>94.5</b>	<b>65.2</b>	<b>57.5</b>	1633

DeepSeek-R1蒸馏模型的表现优于GPT-4o及Claude-3.5-Sonnet闭源模型

# 私有化部署（代码模型）

Model	#Total Params	#Active Params	Context Length	Download
DeepSeek-Coder-V2-Lite-Base	16B	2.4B	128k	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Lite-Base">https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Lite-Base</a>
DeepSeek-Coder-V2-Lite-Instruct	16B	2.4B	128k	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Lite-Instruct">https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Lite-Instruct</a>
DeepSeek-Coder-V2-Base	236B	21B	128k	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Base">https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Base</a>
DeepSeek-Coder-V2-Instruct	236B	21B	128k	<a href="https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Instruct">https://modelscope.cn/models/deepseek-ai/DeepSeek-Coder-V2-Instruct</a>

基于DeepSeekMoE框架发布了拥有160亿和2360亿参数的DeepSeek-Coder-V2。

其中，激活参数仅为24亿和210亿，这包括了基础模型和指令模型。

# vllm使用

---

vllm使用：是由伯克利大学 LMSYS 组织开源的LLM高速推理框架，用于提升LLM的吞吐量与内存使用效率。它通过 PagedAttention 技术高效管理注意力键和值的内存，并结合连续批处理技术优化推理性能。vLLM 支持量化技术、分布式推理、与 Hugging Face 模型无缝集成等功能

```
vllm serve deepseek-ai/DeepSeek-R1-Distill-Qwen-32B --tensor-parallel-size 2 --max-model-len 32768 --enforce-eager
```

- vllm serve, 启动 vLLM 推理服务的命令
- deepseek-ai/DeepSeek-R1-Distill-Qwen-32B, Hugging Face 模型库中的模型名称, vLLM 会尝试从 HF 下载模型
- --tensor-parallel-size 2, 启用张量并行, 在 2 个 GPU 上分布式运行模型 (适合 32B 大模型)
- --max-model-len 32768, 设置模型的最大上下文长度 (32K tokens), 确保能处理长文本。
- --enforce-eager, 禁用 CUDA Graph 优化 (可能在某些环境下更稳定, 但性能稍低)

# Vllm使用

Thinking: 如果我在本地的ubuntu下面有 /root/autodl-tmp/models/tcl90/deepseek-r1-distill-qwen-32b-gptq-int4, 如何使用vllm进行推理?

```
vllm serve /root/autodl-tmp/models/tcl90/deepseek-r1-distill-qwen-32b-gptq-int4 --tensor-parallel-size 1 --max-model-len 32768 --enforce-eager --quantization gptq --dtype half
```

关键改动: 指定本地路径: 替换 HF 模型名为你的本地路径。

--quantization gptq: 显式声明使用 GPTQ 量化。

--dtype: 设为 half (FP16) 或 auto (自动选择), 因为 GPTQ 本身是 4-bit, 但计算时需指定中间精度。

```
Loading safetensors checkpoint shards: 0% Completed | 0/4 [00:00<?, ?it/s]
Loading safetensors checkpoint shards: 25% Completed | 1/4 [00:00<00:02, 1.13it/s]
Loading safetensors checkpoint shards: 50% Completed | 2/4 [00:02<00:02, 1.22s/it]
Loading safetensors checkpoint shards: 75% Completed | 3/4 [00:03<00:01, 1.22s/it]
Loading safetensors checkpoint shards: 100% Completed | 4/4 [00:04<00:00, 1.17s/it]
Loading safetensors checkpoint shards: 100% Completed | 4/4 [00:04<00:00, 1.17s/it]
```

```
ERROR 03-25 11:02:05 [core.py:340] RuntimeError: CUDA out of memory occurred when warming up sampler
ase try lowering `max_num_seqs` or `gpu_memory_utilization` when initializing the engine.
ERROR 03-25 11:02:05 [core.py:340]
CRITICAL 03-25 11:02:05 [core_client.py:269] Got fatal signal from worker processes, shutting down. S
cause issue.
```

# Vllm使用

```
vllm serve /root/autodl-tmp/models/tclf90/deepseek-r1-distill-qwen-32b-gptq-int4 --tensor-parallel-size 1 --max-model-len 4096 --quantization gptq --dtype half --gpu-memory-utilization 0.8 --max-num-seqs 8 --enforce-eager
```

```
INFO 03-25 11:06:45 [loader.py:429] Loading weights took 4.84 seconds
INFO 03-25 11:06:45 [gpu_model_runner.py:1176] Model loading took 18.1678 GB and 5.752492 seconds
ERROR 03-25 11:06:47 [core.py:340] EngineCore hit an exception: Traceback (most recent call last):
ERROR 03-25 11:06:47 [core.py:340]   File "/root/miniconda3/lib/python3.10/site-packages/vllm/v1/engine/core.py", line 332, in run
_engine_core
ERROR 03-25 11:06:47 [core.py:340]     engine_core = EngineCoreProc(*args, **kwargs)
ERROR 03-25 11:06:47 [core.py:340]   File "/root/miniconda3/lib/python3.10/site-packages/vllm/v1/engine/core.py", line 287, in __i
nit__
ERROR 03-25 11:06:47 [core.py:340]     super().__init__(vllm_config, executor_class, log_stats)
ERROR 03-25 11:06:47 [core.py:340]   File "/root/miniconda3/lib/python3.10/site-packages/vllm/v1/engine/core.py", line 62, in __in
it__
ERROR 03-25 11:06:47 [core.py:340]     num_gpu_blocks, num_cpu_blocks = self._initialize_kv_caches(
ERROR 03-25 11:06:47 [core.py:340]   File "/root/miniconda3/lib/python3.10/site-packages/vllm/v1/engine/core.py", line 124, in _in
italize_kv_caches
ERROR 03-25 11:06:47 [core.py:340]     kv_cache_configs = get_kv_cache_configs(vllm_config, kv_cache_specs,
ERROR 03-25 11:06:47 [core.py:340]   File "/root/miniconda3/lib/python3.10/site-packages/vllm/v1/core/kv_cache_utils.py", line 576
, in get_kv_cache_configs
ERROR 03-25 11:06:47 [core.py:340]     check_enough_kv_cache_memory(vllm_config, kv_cache_spec,
ERROR 03-25 11:06:47 [core.py:340]   File "/root/miniconda3/lib/python3.10/site-packages/vllm/v1/core/kv_cache_utils.py", line 468
, in check_enough_kv_cache_memory
ERROR 03-25 11:06:47 [core.py:340]     raise ValueError("No available memory for the cache blocks. ")
ERROR 03-25 11:06:47 [core.py:340] ValueError: No available memory for the cache blocks. Try increasing `gpu_memory_utilization` w
hen initializing the engine.
ERROR 03-25 11:06:47 [core.py:340]
CRITICAL 03-25 11:06:47 [core_client.py:269] Got fatal signal from worker processes, shutting down. See stack trace above for root
cause issue.
Killed
```

尽管我们已经将参数调整到非常保守的配置  
(max-model-len=4096、gpu-memory-  
utilization=0.8)，但 32B GPTQ 量化模型仍然  
无法在 24GB GPU 上运行。

# CASE: DeepSeek-R1-7B使用 (GPU部署)



# CASE: DeepSeek-R1-7B使用

# 导入必要的库

```
from modelscope import AutoModelForCausalLM, AutoTokenizer
```

# 设置模型路径

```
model_name = "/root/autodl-tmp/models/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B"
```

# 加载模型

# torch\_dtype="auto" 自动选择合适的数据类型

# device\_map="cuda" 指定使用GPU加速

```
model = AutoModelForCausalLM.from_pretrained(  
    model_name,  
    torch_dtype="auto",  
    device_map="cuda" # 也可以设置为 "auto" 自动选择设备  
)
```

# 加载对应的分词器

```
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

# 设置用户输入的提示词

```
prompt = "帮我写一个二分查找法"
```

# 构建对话消息列表

```
messages = [  
    {"role": "system", "content": "You are a helpful assistant."},  
    {"role": "user", "content": prompt}  
]
```

# 将消息转换为模型可接受的格式

```
text = tokenizer.apply_chat_template(  
    messages,  
    tokenize=False,  
    add_generation_prompt=True  
)
```

# CASE: DeepSeek-R1-7B使用

# 将文本转换为模型输入格式并移动到正确的设备上

```
model_inputs = tokenizer([text],
return_tensors="pt").to(model.device)
```

# 生成回复

```
generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=2000
)
```

# 提取生成的文本部分（去除输入部分）

```
generated_ids = [
    output_ids[len(input_ids):] for input_ids, output_ids in
zip(model_inputs.input_ids, generated_ids)
]
```

# 将生成的token ID解码为文本

# skip\_special\_tokens=True 跳过特殊token

```
response = tokenizer.batch_decode(generated_ids,
skip_special_tokens=True)[0]
```

# 打印生成的回复

```
print(response)
```

嗯，我现在要学习一下二分查找法，也就是二分查找。我对这个算法不是很熟悉，但我知道它是一种高效的查找方法，特别是在有序数组中。让我先理清思路，然后一步步来。

首先，二分查找的基本思想是将一个数组不断对半分，然后在分好的两部分中继续查找，直到找到目标元素。这种方法的时间复杂度是 $O(\log n)$ ，这比线性查找要快很多，特别是当数组很大时。

那具体步骤是什么呢？我记得应该是先确定数组的中点，然后比较中点处的元素和目标元素的大小。如果中点处的元素等于目标，就找到了；如果目标比中点小，那么在左边继续查找；如果目标比中点大，则在右边继续查找。这个过程不断重复，直到找到目标或者数组遍历完毕。

举个例子，假设有一个有序数组：[1,3,5,7,9,11,13,15]，我要找数字7。首先，数组的长度是8，中点应该是第4个元素，也就是7的位置。所以直接找到了。但如果我要找的是数字6，那么中点是7，比6大，所以应该在左边继续查找。然后新的子数组是[1,3,5]，中点是3的位置，元素是5。6比5大，所以继续在右边查找。新的子数组是[5,7]，中点是6的位置，元素是7。但是6比7小，所以继续在左边查找，但左边只有一个元素5，比较后发现6不在数组中。

那么，二分查找的关键点在于数组必须是有序的，并且每次查找都要计算中点，并根据比较结果调整左右边界。那具体怎么在代码中实现呢？

我记得通常会用一个循环，初始时左边界left=0，右边界right=n-1。然后计算中点mid=(left+right)/2。如果数组[mid]等于目标，返回mid；否则，如果目标小于数组[mid]，则right=mid-1，继续在左边查找；否则，left=mid+1，继续在右边查找。直到left>right时，说明没找到。

那我来试着写一个伪代码：

```
function binary_search(array, target):
    left = 0
    right = len(array) - 1
    while left <= right:
        mid = (left + right) // 2
        if array[mid] == target:
            return mid
        elif array[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1 # 表示没找到
```

# Ollama使用

# Ollama使用

---

Ollama 官方主要支持 macOS 和 Linux，但 Windows用户也可以安装

## 方法1: 使用 WSL

Step1, 打开 PowerShell (管理员权限)，运行：

```
wsf --install
```

重启电脑后，WSL 会自动完成安装 (默认安装 Ubuntu)

Step2, 安装 Ollama

在 WSL 终端 (Ubuntu) 中运行：

```
curl -fsSL https://ollama.com/install.sh | sh
```

Step3, 启动 Ollama 服务

```
ollama serve
```

(保持此终端运行，另开一个新终端进行后续操作)

# Ollama使用

方法 2: 直接下载 Windows 版

<https://ollama.com/>

安装后, Ollama 会作为服务运行 (可在任务管理器查看)

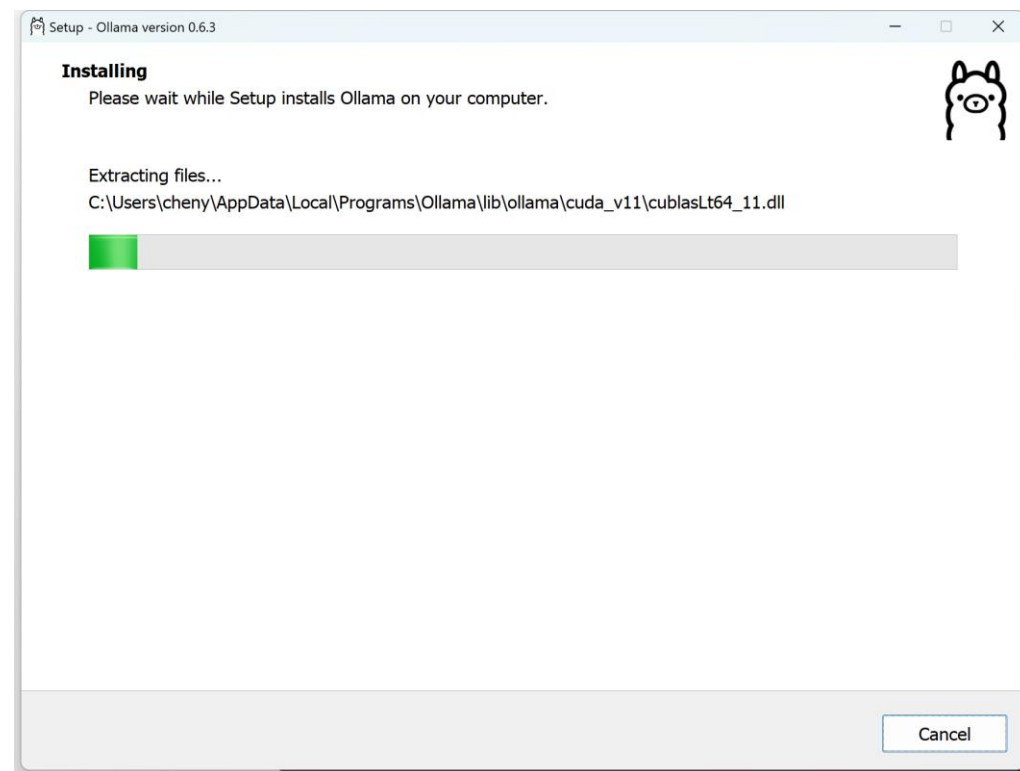


Get up and running with large  
language models.

Run Llama 3.3, DeepSeek-R1, Phi-4, Mistral,  
Gemma 3, and other models, locally.

Download ↓

Available for macOS,  
Linux, and Windows



# Ollama使用

Ollama官方模型库 <https://ollama.com/library>

下载 deepseek-r1:1.5b 模型

```
ollama pull deepseek-r1:1.5b
```

如果要删除该模型，可以使用

```
ollama rm deepseek-r1:1.5b
```

运行该模型，使用

```
ollama run deepseek-r1:1.5b
```

```
PS C:\Windows\System32> ollama run deepseek-r1:1.5b
>>> 你好啊
<think>

</think>
你好！很高兴见到你，有什么我可以帮忙的吗？
```

## qwq

QwQ is the reasoning model of the Qwen series.

tools

32b

↓ 1.1M Pulls    🔖 8 Tags    ⌚ Updated 2 weeks ago

## deepseek-r1

DeepSeek's first-generation of reasoning models with comparable performance to OpenAI-o1, including six dense models distilled from DeepSeek-R1 based on Llama and Qwen.

1.5b

7b

8b

14b

32b

70b

671b

↓ 31.5M Pulls    🔖 29 Tags    ⌚ Updated 7 weeks ago

# CASE: DeepSeek-R1使用 (API调用)

# CASE：DeepSeek-R1使用-阿里代理

## 1、基本设置：

```
import dashscope
from dashscope.api_entities.dashscope_response import Role
# 设置 API key
dashscope.api_key = "your-api-key"
```

## 2、模型调用封装：

```
def get_response(messages):
    response = dashscope.Generation.call(
        model='deepseek-r1', # 使用 deepseek-r1 模型
        messages=messages,
        result_format='message' # 将输出设置为message形式
    )
    return response
```

## 3、情感分析实现：

```
review = '这款音效特别好 给你意想不到的音质。'
messages = [
    {"role": "system", "content": "你是一名舆情分析师，帮我判断产品口碑的正负向，回复请用一个词语：正向 或者 负向"},
    {"role": "user", "content": review}
]
response = get_response(messages)
print(response.output.choices[0].message.content)
```

正向



# Summary

---




- 大模型API是连接AI能力的桥梁，让开发者无需关注底层架构即可调用前沿AI能力，极大拓展了技术应用的边界。
- Prompt工程是激活大模型潜力的钥匙

结构化设计（角色定义/分步指令/示例规范）

业务场景对齐（需求分析→Prompt迭代→效果验证）

性能优化技巧（温度系数/输出限制/上下文管理）

- LLM正在重塑开发范式，通过大模型API接口覆盖NLP/CV/多模态任务



Thank You  
Using data to solve problems