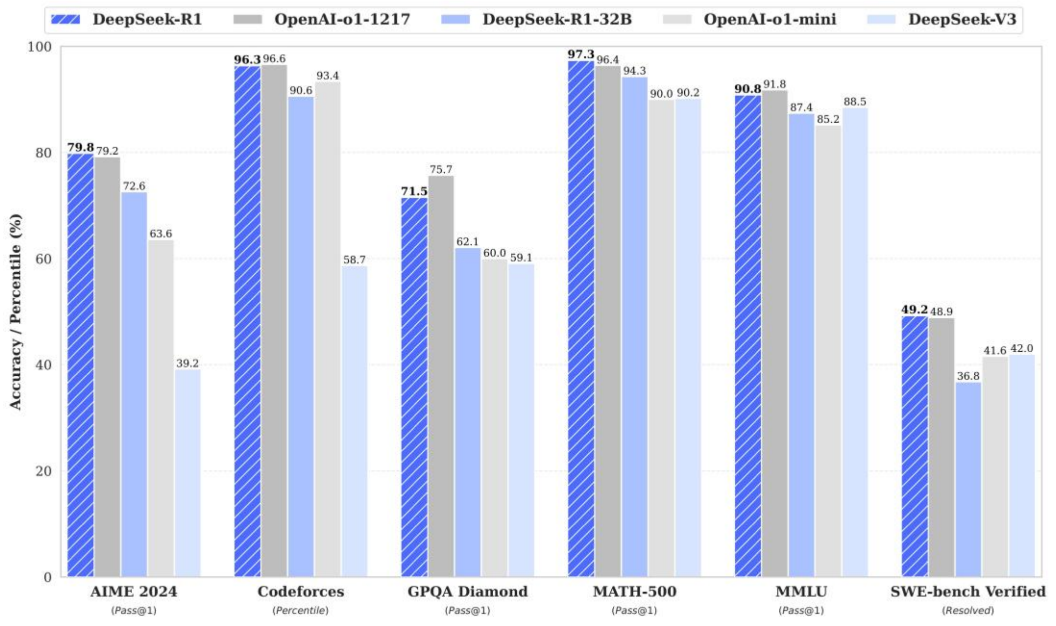
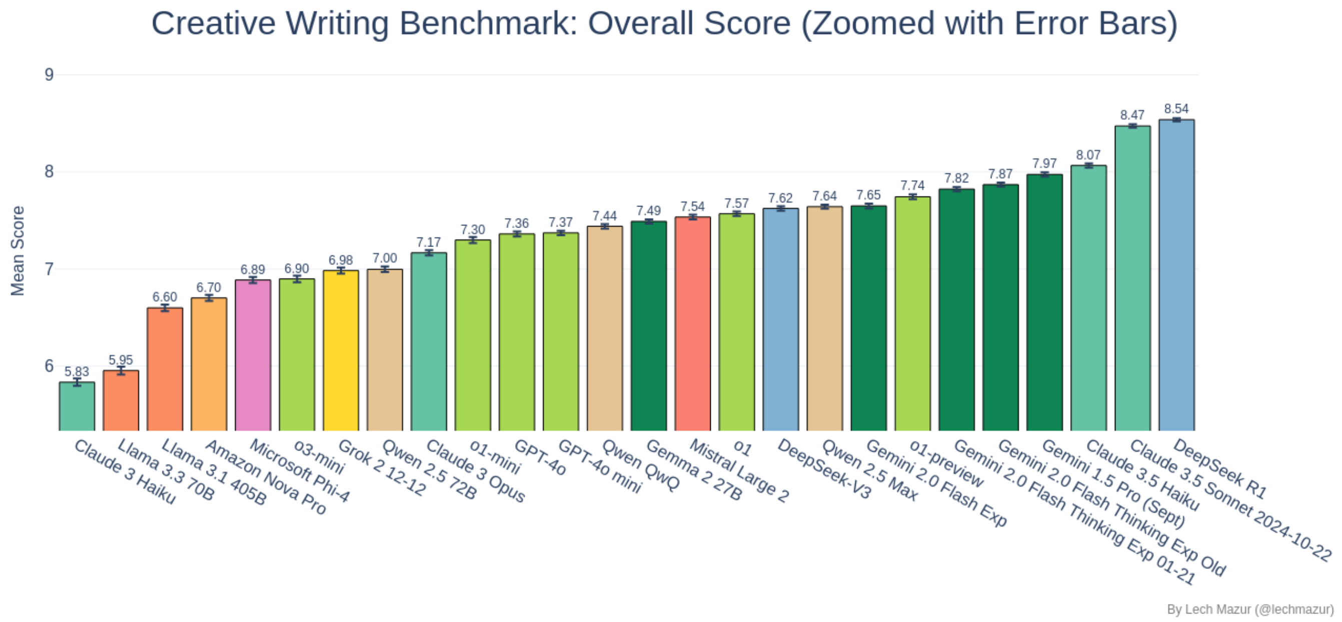
Local Deployment of DeepSeek R1 671B

Xihan Li
University College London
Google Developers Expert in Machine Learning
https://snowkylin.github.io

DeepSeek R1 671B: one of the first-tier LLMs with reasoning



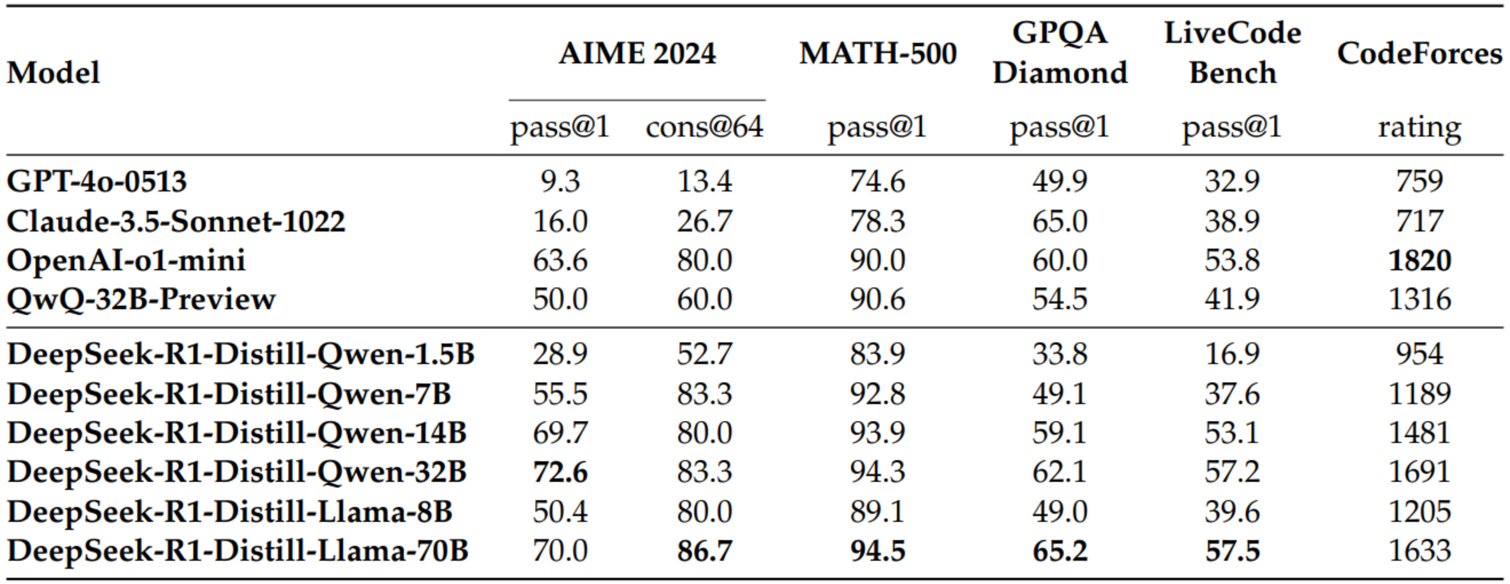


https://huggingface.co/deepseek-ai/DeepSeek-R1
https://github.com/deepseek-ai/DeepSeek-R1
https://arxiv.org/abs/2501.12948

https://github.com/lechmazur/writing

Distilled Model vs Original Model

- Distilled DeepSeek R1 = fine-tuned Qwen (Alibaba) or Llama (Meta) model
  - Use the 671B original model to “teach” other open-sourced models
    - Qwen 1.5B (mobile), 7B (8G+ VRAM), 14B (16G+ VRAM) , 32B (24G+ VRAM)
    - Llama: 8B (8G+ VRAM) / 70B (2 x 24G+ VRAM)
  - Easy deployment (LM Studio)

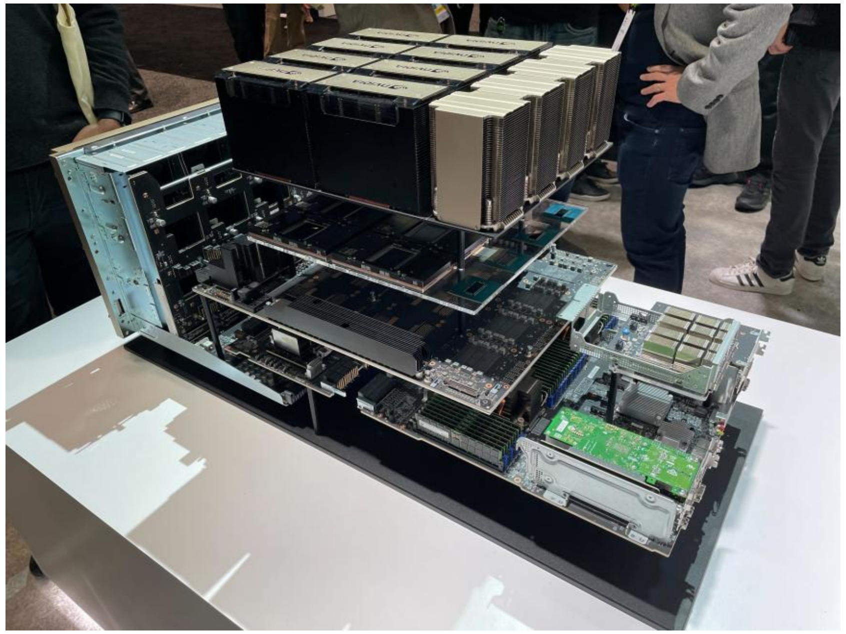


Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	**1820**
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	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-R1-671B	**79.8**		**97.3**	**71.5**	**65.9**	**2029**

Is it possible to deploy a 671B model locally on our own machine?

- 671B means 671,000,000,000 parameters
- ~720GB in its original form
- Typically requires a monster-like AI server like NVIDIA DGX H100
  - 8 x H100 80G
  - ~$200,000 USD
  - 130.45kg
  - 10.2kW max
- Seems far from customer-grade hardware

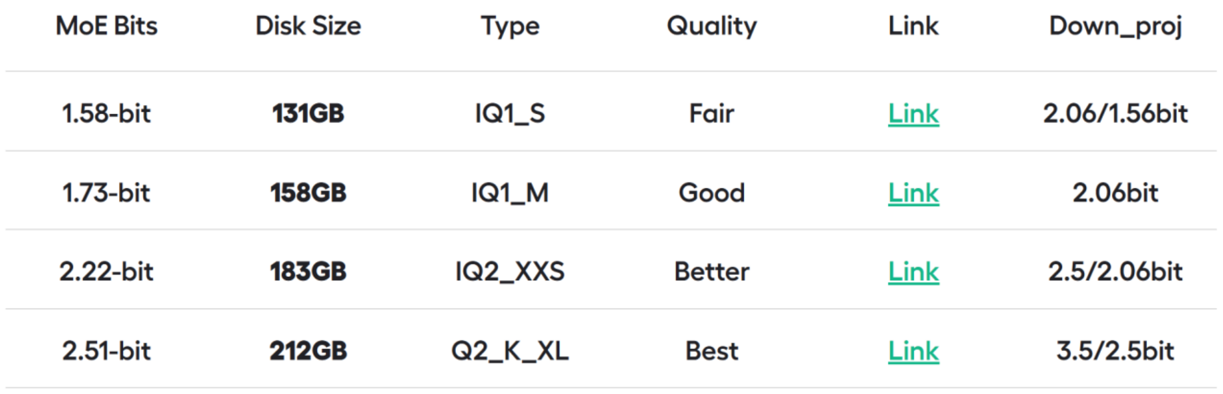
Building Your AI Center of Excellence
The NVIDIA DGX H100 is the foundation of modern data centers, with eight NVIDIA H100 GPUs to tackle any AI workload.
NVIDIA DGX H100



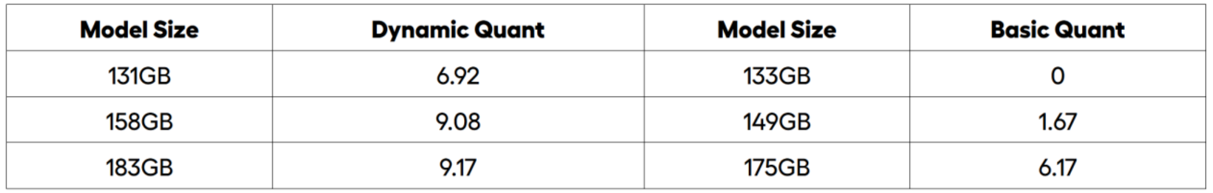
https://www.nvidia.com/en-gb/data-center/dgx-h100/
https://x.com/netris_io/status/1770593406395515207

Dynamic Quantization

- DeepSeek R1 is trained on FP8
- Standard Quantization: quantize all parameters from 8-bit to x-bit
  - Standard 4-bit quantized version is still 404GB
- Dynamic Quantization:
  - Selectively quantize a few important layers to higher bits (4-6 bits)
  - Leave most of the MoE layers (~88% parameters) to be quantized to lower bits (1.5-2.5 bits)



MoE Bits	Disk Size	Type	Quality	Link	Down_proj
1.58-bit	**131GB**	IQ1_S	Fair	Link	2.06/1.56bit
1.73-bit	**158GB**	IQ1_M	Good	Link	2.06bit
2.22-bit	**183GB**	IQ2_XXS	Better	Link	2.5/2.06bit
2.51-bit	**212GB**	Q2_K_XL	Best	Link	3.5/2.5bit

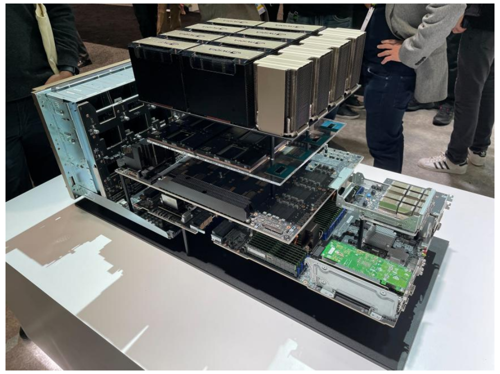


Model Size	Dynamic Quant	Model Size	Basic Quant
131GB	6.92	133GB	0
158GB	9.08	149GB	1.67
183GB	9.17	175GB	6.17

https://unsloth.ai/blog/deepseekr1-dynamic

Many "impossible" missions can be accomplished if we can do some trade-offs…

- Precision
  - 8-bit (720GB) – 4-bit (405GB) – **2.51~1.53-bit (212GB-131GB)**
- Target speed / context window
  - 50-100 token/s, max 128k tokens – **2-10 token/s, max 4096 tokens**
- Budget
  - $500-3000 (typical customer-grade PC) – **$5600+ (high-end PC)**

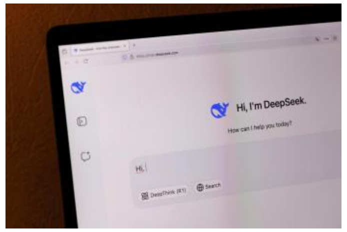


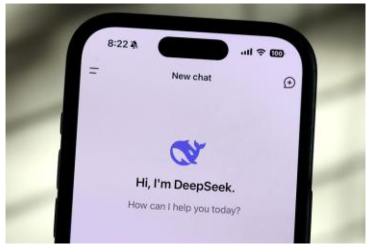
→

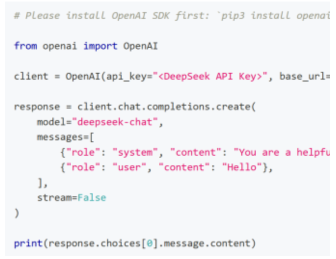
/

Is it worth it?

Cloud Service (Web/API)



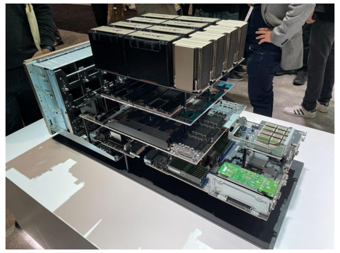




```
# Please install OpenAI SDK first: `pip3 install openai

from openai import OpenAI

client = OpenAI(api_key="<DeepSeek API Key>", base_url=

response = client.chat.completions.create(
    model="deepseek-chat",
    messages=[
        {"role": "system", "content": "You are a helpfu
        {"role": "user", "content": "Hello"},
    ],
    stream=False
)

print(response.choices[0].message.content)
```

- Cheap
- Convenient
- Fast
- Larger context window
- (should be) more stable
- Less restrictive when using third-party hosted model (e.g., in Cursor)

Local Deployment


/

- Fully offline
  - Run without stable Internet (robot, car, embedded/industrial devices, etc.)
  - Less privacy concerns
- Possibility of full customization
  - Less restrictive, can be fine-tuned
- Not very costly if existing hardware can be utilized
- It is so cool, isn't it?

Hardware Requirement

- Memory (RAM + VRAM) size and bandwidth is the main constraint
  - 1.73-bit dynamic quantization: RAM + VRAM ≥ 200 GB
  - 4-bit standard quantization: RAM + VRAM ≥ 500 GB
- Mac Studio: 192GB unified memory ($5600)
- Maximal memory that can be installed on a 4-slot customer-grade motherboard: 4x48GB = 192GB (VENGEANCE® RGB 192GB (4x48GB) DDR5 DRAM 5200MT/s, ~$750)
  - Buy a kit with four memory sticks for stablibility!
- Maximal memory that can be installed on a 4-slot workstation motherboard (TRX50 for ThreadRipper): 4x96GB = 384GB (Micron 96GB 5600MHz ECC Registered 2Rx4 DDR5 Server Memory, ~$2400)
  - 128GB is possible but super expensive
  - 8-slot workstation motherboard (WRX90) is also available but expensive
- VRAM (GPU) is optional
  - Offload some layers to VRAM for acceleration

Deploy with Ollama

- Popular choice: llama.cpp, ollama, vLLM
  - Llama.cpp requires manual compilation for CUDA support (failed for me)
    - If you don't use GPU, llama.cpp is a good choice, see https://unsloth.ai/blog/deepseekr1-dynamic for instruction
  - vLLM's GGUF format support is not good
    - Required by the dynamic quantization version
    - If you don't use GGUF, vLLM is a good choice

Deployment process

1. Download .gguf files from https://huggingface.co/unsloth/DeepSeek-R1-GGUF and merge with llama.cpp's "llama-gguf-split" command
2. Install Ollama

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

3. Create a `modelfile` that guide ollama to create a model
4. Create the model in ollama

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

5. Run the model

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

```
FROM /home/snowkylin/DeepSeek-R1-UD-IQ1_M.gguf
PARAMETER num_gpu XXX
PARAMETER num_ctx 2048
PARAMETER temperature 0.6
TEMPLATE "<|User|>{{ .System }} {{ .Prompt }}<|Assistant|>"
```

Thank You!

Xihan Li
University College London
Google Developers Expert in Machine Learning
https://snowkylin.github.io