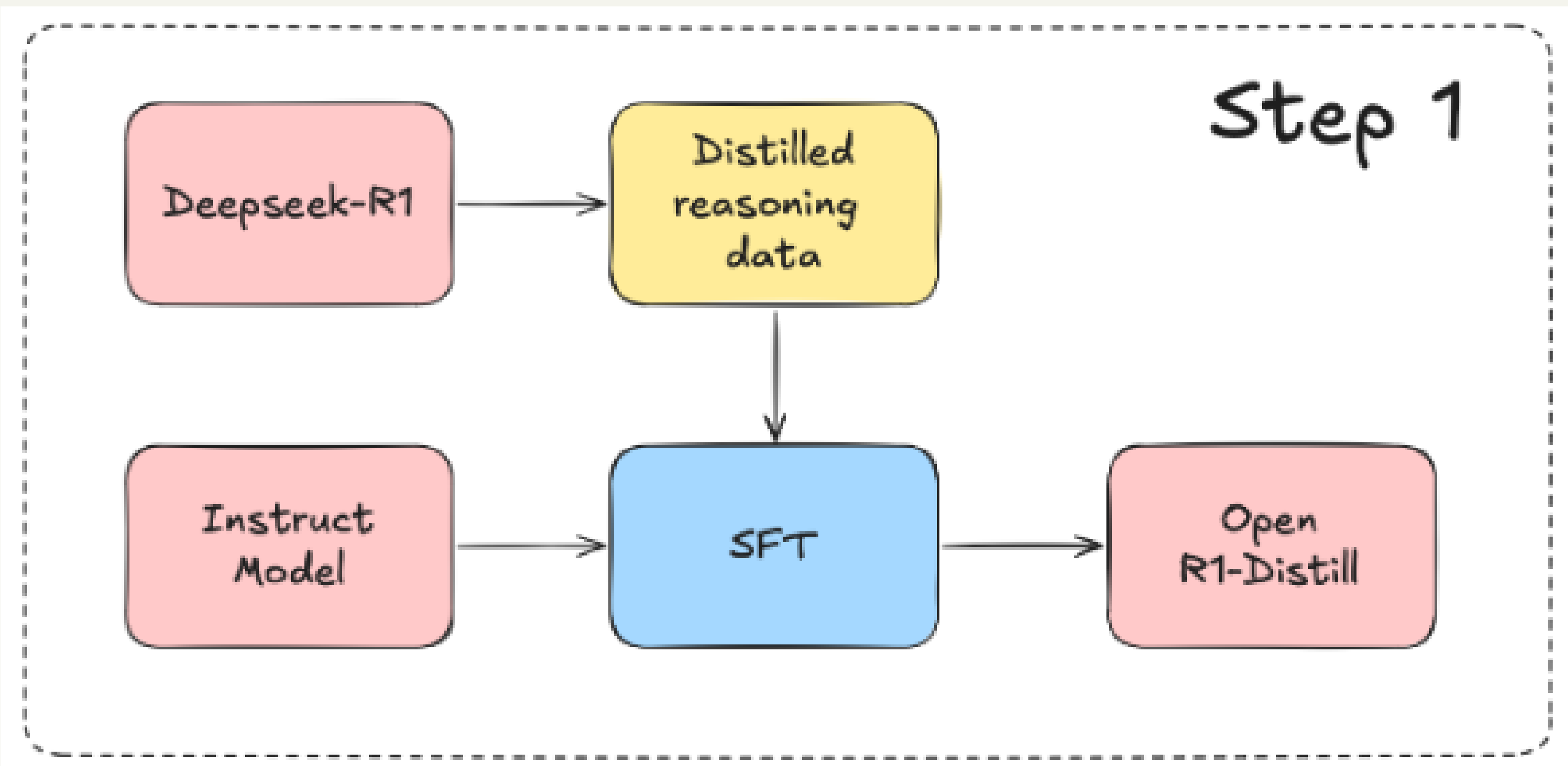
Analytics Vidhya

Report on DeepSeek-R1

deepseek R1 vs OpenAI o1

How to access deepseek using Ollama



Dipanjan (DJ)

DeepSeek R1

The DeepSeek R1 model boasts a **671 billion parameters** architecture and has been trained on the DeepSeek V3 Base model. Its focus on Chain of Thought (CoT) reasoning makes it a strong contender for tasks requiring advanced comprehension and reasoning. Interestingly, despite its large parameter count, only 37 billion parameters are activated during most operations, similar to DeepSeek V3.

DeepSeek R1 isn't just a monolithic model; the ecosystem includes six distilled models fine-tuned on synthetic data derived from DeepSeek R1 itself. These smaller models vary in size and target specific use cases, offering solutions for developers who need lighter, faster models while maintaining impressive performance.

Distilled Model Lineup

Model	Base Model	Download
DeepSeek-R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	🤗 HuggingFace
DeepSeek-R1-Distill-Qwen-7B	Qwen2.5-Math-7B	🤗 HuggingFace
DeepSeek-R1-Distill-Llama-8B	Llama-3.1-8B	🤗 HuggingFace
DeepSeek-R1-Distill-Qwen-14B	Qwen2.5-14B	🤗 HuggingFace
DeepSeek-R1-Distill-Qwen-32B	Qwen2.5-32B	🤗 HuggingFace
DeepSeek-R1-Distill-Llama-70B	Llama-3.3-70B-Instruct	🤗 HuggingFace

These distilled models enable flexibility, catering to both local deployment and API usage. Notably, the Llama 33.7B model outperforms the o1 Mini in several benchmarks, underlining the strength of the distilled variants.

Model	#Total Params	#Activated Params	Context Length	Download
DeepSeek-R1-Zero	671B	37B	128K	🤗 HuggingFace
DeepSeek-R1	671B	37B	128K	🤗 HuggingFace

Source: DeepSeek R1 vs OpenAI o1

Analytics Vidhya

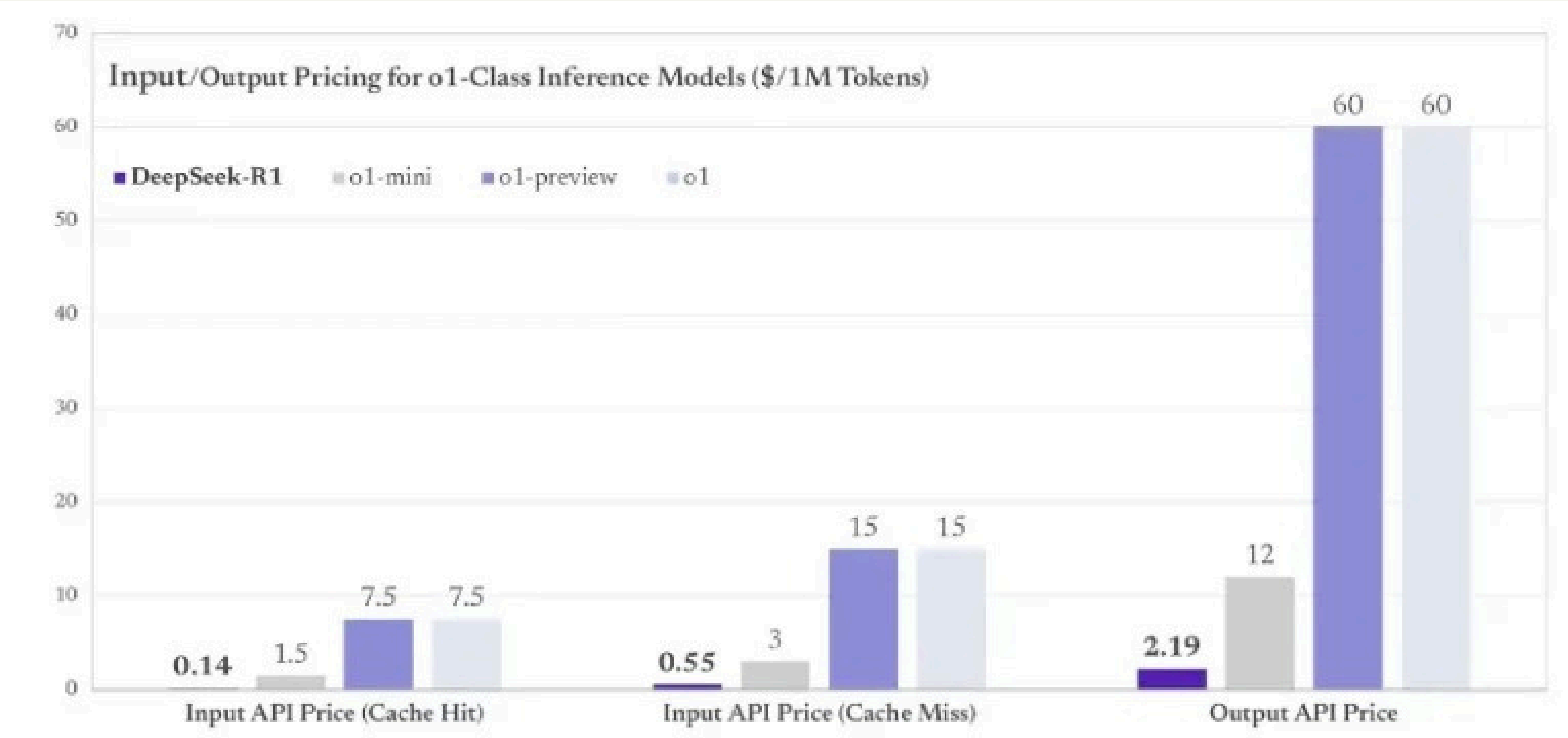
High Performance at Low Cost

How did they do it?

- **Reinforcement Learning First** – Instead of relying heavily on human-labeled data, DeepSeek R1 used **self-evolution** to boost reasoning, cutting annotation costs. A small supervised dataset provided a solid "cold start."
- **Smart Distillation** – They transferred powerful reasoning from large models into smaller, efficient ones (e.g., 14B), achieving top-tier performance with lower costs.
- **Benchmark Focus** – R1 shines in key areas like math & coding, ranking impressively on AIME (79.8%) and MMLU (90.8%).
- **Efficient Architecture** – Techniques like **Sparse Attention** and **Mixture of Experts (MoE)** keep it fast & cost-effective.
- **Strategic Design** – Prioritizing reasoning over general NLP ensures optimized performance where it matters most.

Source: DeepSeek R1 vs OpenAI o1

Price Comparison



DeepSeek R1 scores comparably to OpenAI o1 in most evaluations and even outshines it in specific cases. This high level of performance is complemented by accessibility; DeepSeek R1 is free to use on the DeepSeek chat platform and offers affordable API pricing. Here's a cost comparison:

- DeepSeek R1 API: 55 Cents for input, $2.19 for output (1 million tokens)
- OpenAI o1 API: $15 for input, $60 for output (1 million tokens)

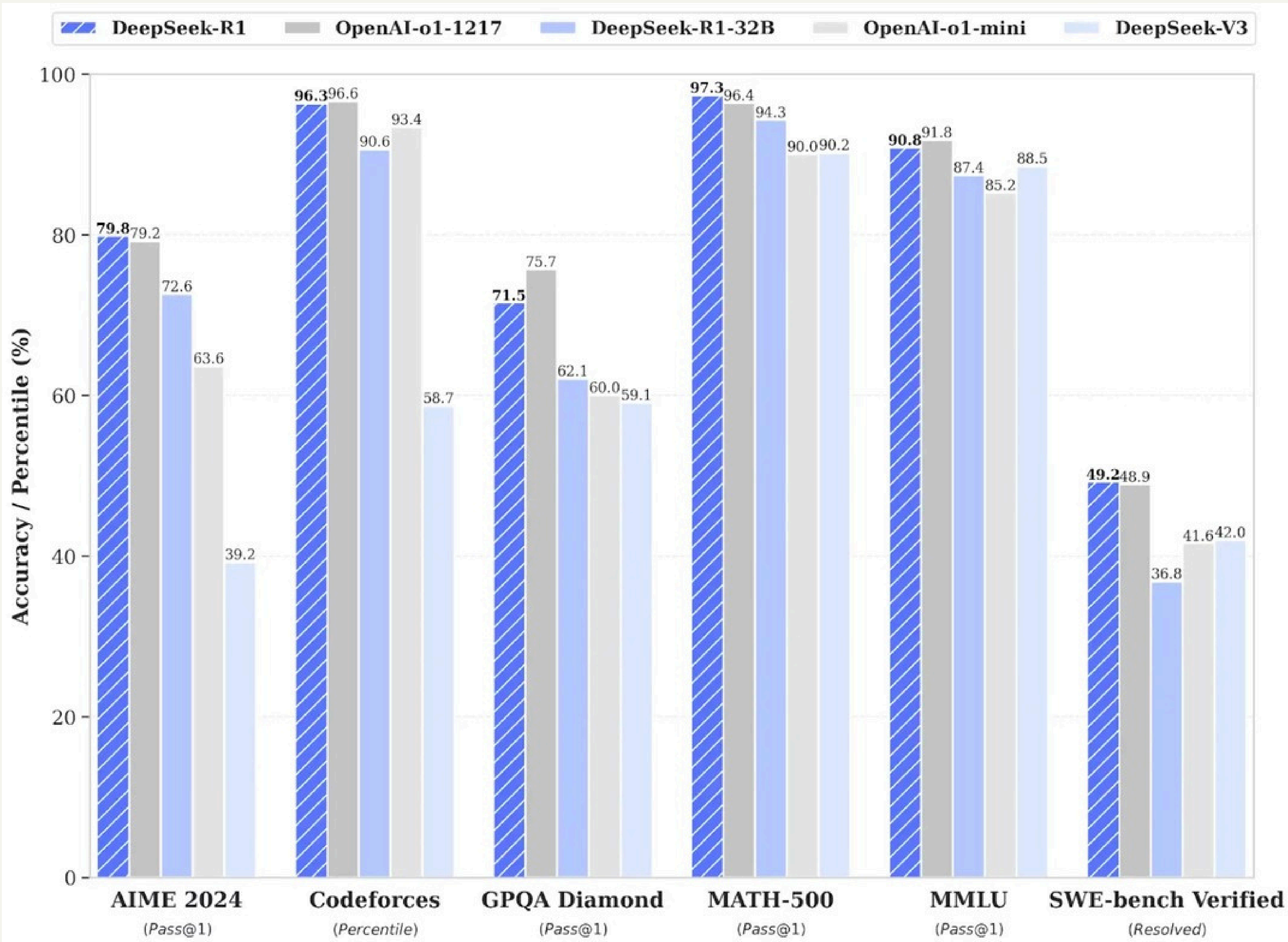
API is **96.4% cheaper than chatgpt**.

DeepSeek R1's lower costs and free chat platform access make it an attractive option for budget-conscious developers and enterprises looking for scalable AI solutions.

Source: DeepSeek R1 vs OpenAI o1

Analytics Vidhya

Comparison of Different Benchmarks



Source: DeepSeek R1 vs OpenAI o1

Visit the Ollama website to download the tool.

For Linux users:

- Execute the following command in your terminal:

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

Then run the model.
Here's the Ollama like for DeepSeek R1: ollama run deepseek-r1
Copy the command: ollama run deepseek-r1

```
ollama run deepseek-r1
```

Source: DeepSeek R1 vs OpenAI o1

We are running Ollama run deepseek-r1:1.5b in local and it will take few minutes to download the model.

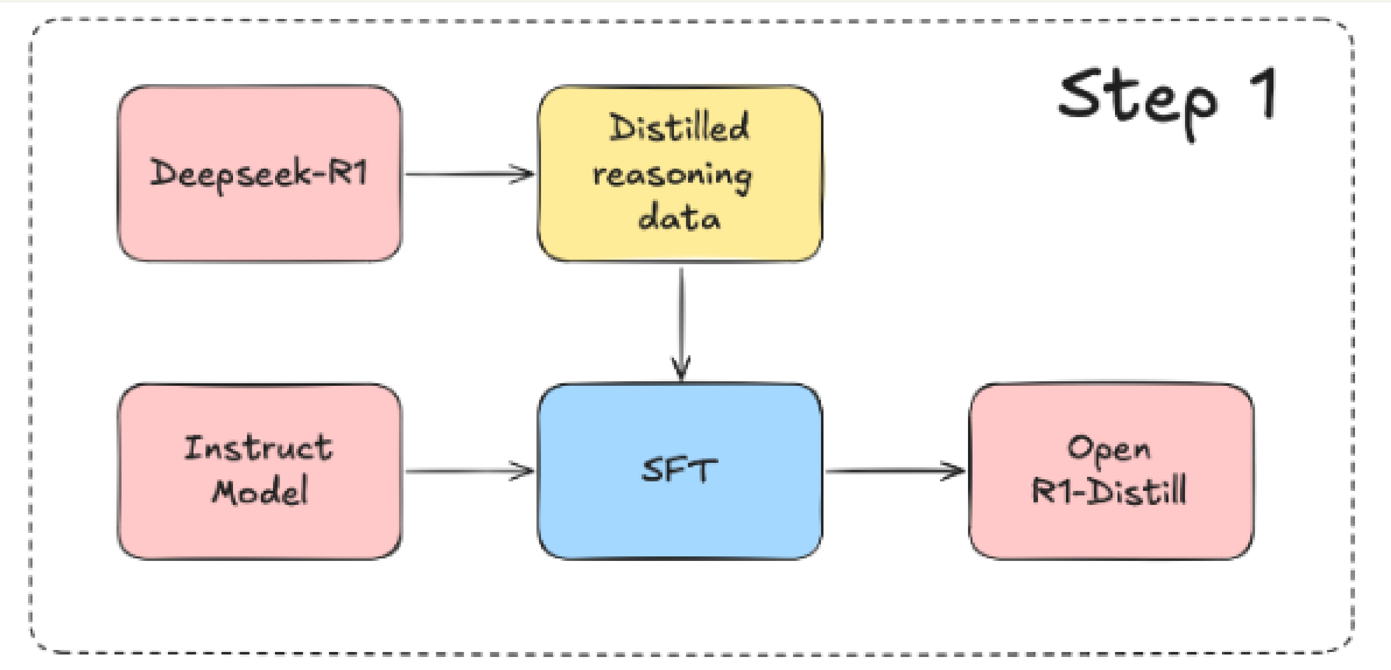
```
pankaj@AVLT092:~$ ollama run deepseek-r1:1.5b
>>> Send a message (/? for help)
```

Prompt: Give me code for the Fibonacci nth series

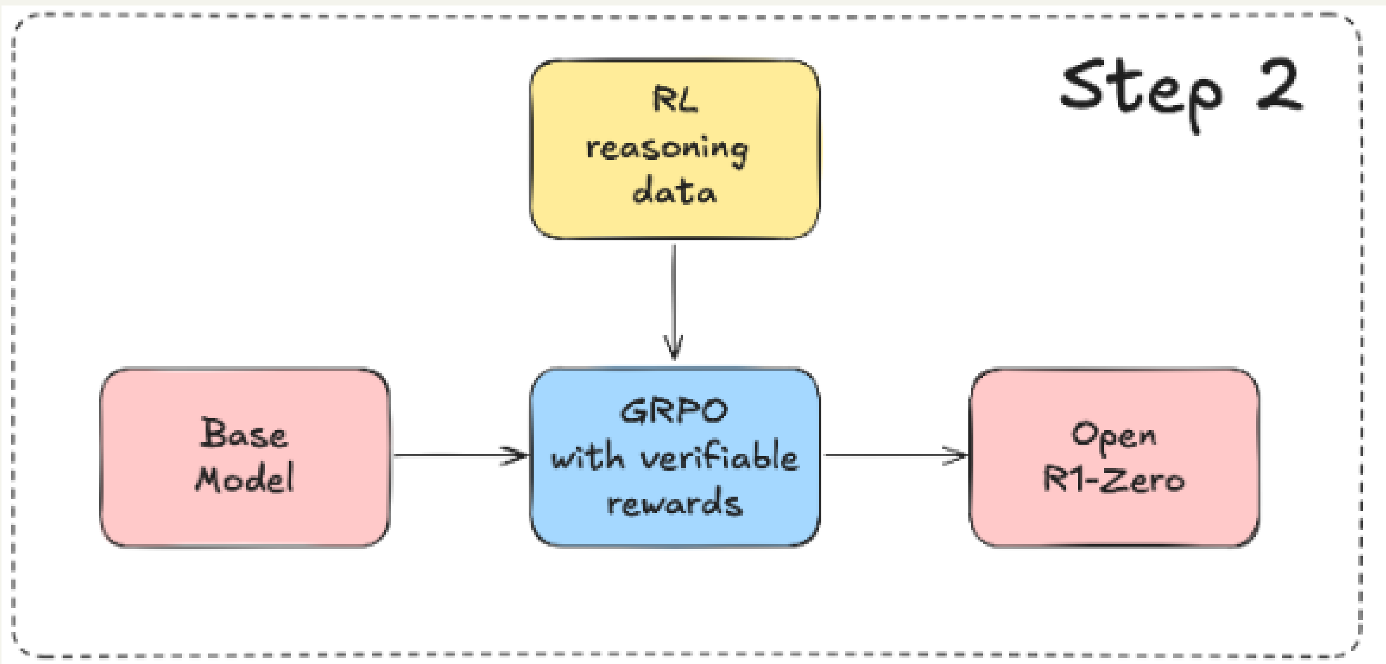
```
pankaj@AVLT092:~$ ollama run deepseek-r1:1.5b
>>> Give me code for fibonacci nth series
<think>
Okay, the user is asking for a Fibonacci sequence calculator. I remember
that the Fibonacci sequence starts with 0 and 1, and each subsequent
number is the sum of the previous two.

I should write a function called fib(n) where n is the term number we want
to find. If n is 0 or 1, it returns 0 since those are the starting points.
For other values, I'll use recursion because each term depends on the two
before it, which seems straightforward.

Wait, but recursion can be inefficient for large n because it recalculates
many terms repeatedly. Maybe the user wants an iterative approach instead
to make it faster and more efficient. That way, even if n is big, it
doesn't take too long.

I'll structure my function with a loop that starts from 2 up to n. I'll
keep track of the two previous numbers, prev1 and prev2. Then for each
step beyond 2, calculate current as the sum of prev1 and prev2. Update
prev2 to be prev1, and update prev1 to current.

Wait again, what if n is 0? The function should return 0, same with n=1.
So I'll add a condition at the beginning of the loop where if it's the
first term, we handle it correctly.

Let me test some examples. For fib(5), it should be 5. Let's see: prev1
starts as 0, prev2 as 1. Then for i=3 to 5:
i=3: current = 1 (0+1)
i=4: current = 2 (1+1)
i=5: current = 3 (1+2) Wait, that's not right. Oh no, I think I messed up the calculation steps.

Wait, for fib(5), let me recount:
n=0: 0
n=1: 1
n=2: 1
n=3: 2
n=4: 3
n=5: 5. So my previous step-by-step was wrong. I need to make sure each iteration is correct.

I think in the loop, when i reaches n-2, which for fib(5) is index 5 (since starting at 0), but wait, let me adjust the code correctly.

Let me write down:

def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    prev1 = 0
    prev2 = 1
    for i in range(2, n+1):
        current = prev1 + prev2
        prev1 = prev2
        prev2 = current
    return prev2
```

Source: DeepSeek R1 vs OpenAI o1

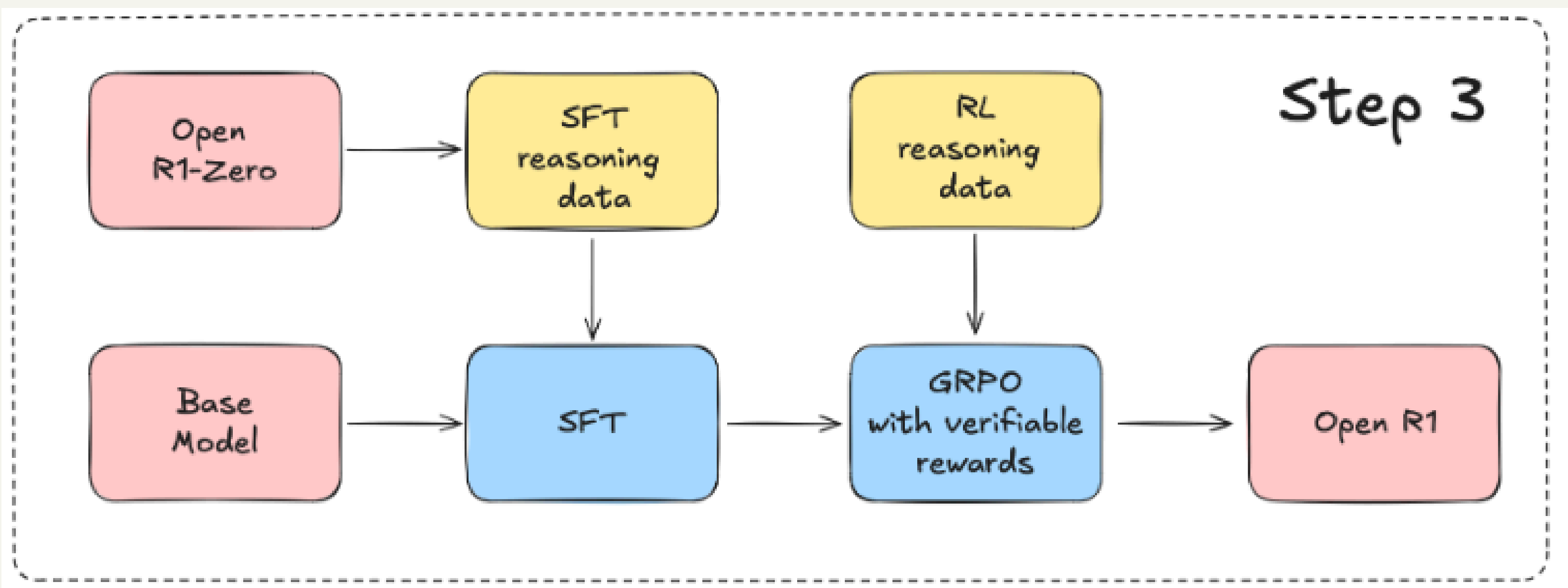
Hugging Face's Work in Progress to Reproduce DeepSeek-R1



Step 1: Replicate the R1-Distill models by distilling a high-quality corpus from DeepSeek-R1.



Step 2: Replicate the pure RL pipeline that DeepSeek used to create R1-Zero
This will likely involve curating new, large-scale datasets for math, reasoning, and code.



Step 3: Show we can go from base model to RL-tuned via multi-stage training.

Source: Open R1

Analytics Vidhya

How DeepSeek Trained AI 30 Times Cheaper?

1. No Fancy Chips, Just Smart Optimizations

Many assumed that export restrictions from the US on advanced AI chips would limit DeepSeek's capabilities. However, they proved that great software can compensate for hardware limitations. Instead of relying on the latest high-end GPUs like the NVIDIA H100, they optimized the hardware they had—likely the NVIDIA H800, which has lower chip-to-chip bandwidth.

DeepSeek engineers focused on **low-level code optimizations** to make memory usage as efficient as possible. Their improvements ensured that **performance was not hindered by chip limitations.** In essence, they maximized what they had instead of waiting for better hardware.

Key takeaway: They didn't bypass restrictions; they simply made their existing resources work smarter.

2. Training Only the Important Parts

Training AI models usually involves updating everything, even parts that don't contribute much. This leads to a massive waste of resources. DeepSeek tackled this problem head-on by training **only the necessary parts of the model.**

Using a technique called **Auxiliary-Loss-Free Load Balancing,** they ensured that only the most relevant parts (experts) of the model were activated and updated. Instead of depending on additional loss functions to balance workload, they introduced a **bias term** that helps dynamically distribute tasks to the right parts of the model.

How it Works?

- Each token (piece of text) is sent to a **small set of experts**, instead of engaging the entire model.
- The system monitors workload and adjusts the **bias term** to prevent some experts from being overloaded while others remain underutilized.
- This dynamic adjustment allows for **efficient resource usage without extra computational overhead.**

Results

- Only **5% of the model's parameters** were trained per token.
- This led to a **95% reduction in GPU usage** compared to companies like Meta.
- Faster training at **significantly lower costs,** without losing accuracy.

In short: Train only what's needed, save big on costs.

Source: How DeepSeek Trained AI 30 Times Cheaper?

How DeepSeek Trained AI 30 Times Cheaper?

3. Faster and Cheaper AI with Compression

Running AI models, especially inference (when generating outputs), is memory-intensive and costly. DeepSeek overcame this by using an innovative technique called **Low-Rank Key-Value (KV) Joint Compression.**

The KV cache stores key-value pairs crucial for attention mechanisms, but storing them at full capacity takes up a lot of memory. DeepSeek found a way to **compress these key-value pairs efficiently,** reducing storage without sacrificing performance.

How it Works?

- The model compresses key and value vectors using a **down-projection matrix,** reducing their size while preserving essential information.
- During inference, only the **compressed version** is stored, significantly reducing memory requirements.
- When needed, the compressed data is expanded back with minimal loss of accuracy.

Benefits

- **Lower memory usage:** DeepSeek stores a much smaller amount of data without losing performance.
- **Faster inference:** Less data to process means quicker responses.
- **Reduced costs:** Less hardware is required to run the model efficiently.

In short: Smaller memory, faster results, lower costs.

How DeepSeek Trained AI 30 Times Cheaper?

4. Smarter Learning with Reinforcement Learning

DeepSeek also improved model learning efficiency through **reinforcement learning.** Instead of relying solely on traditional training methods, they focused on tasks that have **clear, verifiable answers,** such as math and coding problems.

How it Works?

- The AI is given complex, easily verifiable tasks (e.g., coding challenges).
- If the model produces the correct result, it is rewarded and learns to reinforce those patterns.
- If it makes mistakes, adjustments are made to improve performance in future iterations.

This method allowed DeepSeek to **improve accuracy with fewer resources** by focusing only on challenges that provided immediate, measurable feedback.

Why is DeepSeek a Big Deal?

DeepSeek's success comes down to three powerful yet straightforward ideas:

- **Training only what matters:** Focusing on the most important parts of the model to reduce computation.
- **Smart memory compression:** Using less storage without losing performance.
- **Efficient hardware use:** Getting the most out of available resources instead of relying on cutting-edge chips.

These strategies didn't just cut costs—they gave DeepSeek the ability to test, experiment, and innovate faster than their competitors.

What makes their story so compelling is that it's not about having unlimited resources. It's about **making the best use of what's available**. DeepSeek has proven that groundbreaking AI doesn't have to come with an outrageous price tag. Their approach is a blueprint for how companies can think smarter, not harder, when it comes to AI. By focusing on efficiency, they've opened the door for others to rethink how AI models are trained and deployed.

Analytics Vidhya

Useful Resources

1. How DeepSeek Trained AI 30 Times Cheaper?
2. DeepSeek R1 vs OpenAI o1: Which One is Faster, Cheaper and Smarter?
3. Kimi k1.5 vs DeepSeek R1: Battle of the Best Chinese LLMs
4. Building AI Application with DeepSeek-V3
5. How is DeepSeek Making Money?
6. DeepSeek V3 vs GPT-4o: Can Open-Source AI Compete with GPT-4o's Power?
7. https://github.com/deepseek-ai/DeepSeek-R1
8. https://github.com/huggingface/open-r1