Hands-on deployment of 671B model inference (2.51-bit quantization)

Features and versions of R1

The DeepSeek-R1 series of models has made significant breakthroughs in reasoning capabilities through reinforcement learning technology based on DeepSeek-V3, while maintaining the characteristics of low cost and open source. The technical advantages are as follows:

1. The training effect of "reinforcement learning" on large models has been demonstrated: DS is trained entirely based on RL (reinforcement learning) without using any supervised training or human feedback. It can improve performance through self-learning and reduce dependence on manual labeling.
2. High performance with low hardware cost: DeepSeek-R1's performance on tasks such as mathematics, programming, and natural language reasoning is comparable to the official version of OpenAI's GPT-4, but the training cost is only 1/30 of that of OpenAI's similar model.
3. Provides distillation models with multiple parameters to adapt to different application scenarios.

In terms of model versions, DeepSeek has released different versions of the R1 series including: R1-Zero, R1 and distilled versions.

1. DeepSeek-R1-Zero: Based entirely on reinforcement learning training, without using supervised fine-tuning data, it demonstrates strong reasoning capabilities, but has limitations such as poor readability and language consistency.
2. DeepSeek-R1: Based on R1-Zero, it introduces cold start data and multi-stage training strategies to improve the readability, stability, and language consistency of the model.
3. Distilled version: The reasoning capability of DeepSeek-R1 is migrated to smaller-scale models such as Qwen and LLaM, and multiple versions with parameter ranges from 1.5B to 70B are launched.

Press Apply first, and the basic model R1-Zero will not be used directly. Industry evaluations have given high praise to the R1 and Distill models, but it is worth noting that existing articles confuse R1 with the Distill model. The practice of deliberately equating the 7B, 14B and other distillation models with R1 can easily mislead readers to underestimate the actual deployment cost of R1. For the sake of clear distinction, the "R1" mentioned below specifically refers to

the 671B version with the largest number of parameters.

Quantization selection and hardware and software requirements

According to official and community discussions, the full-blooded version of R1 (671B, without quantization) requires two 8-card H100s, or one 8-card H20, or one 8-card H200 to achieve memory unloading of all model parameters. If this is the case, only enterprise-level applications with a budget of at least 2 million can use R1 local deployment. Therefore, the quantized version of R1 launched by the Unsloth.AI community can be used as a "trial package" before using the full-blooded version of R1. ——Unsloth: We explored how to get more local users to run it, and managed to quantize DeepSeek's R1 671B parameter model to 131GB, a reduction of 80% from the original 720GB, and it is very practical.
In actual deployment, different dynamic quantization versions have different effects:

MoE Bits	Type	Disk Size	Accuracy
1.58bit	IQ1_S	131GB	normal
1.73bit	IQ1_M	158GB	good
2.22bit	IQ2_XXS	183GB	better
2.51bit	Q2_K_XL	212GB	most

As it happens, our laboratory has an 8-card H20 server (each card has 96GB of video memory), and we will use it to deploy the 2.51-bit version with the best quantization effect.

- Operating system: Ubuntu 22.04
- software:
    - ollama: v0.5.7
    - llama-gguf-split: 4611 (53debe6f)
    - Model: DeepSeek R1 671b 2.51-bit quantized

Installation Steps

Install ollama

1. Download and unzip the software

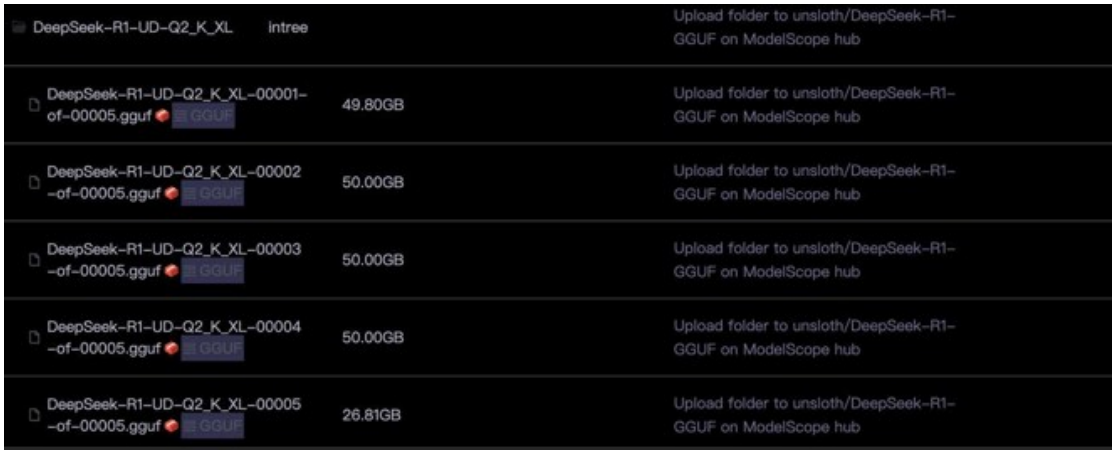
curl -L https://ollama.com/download/ollama-linux-amd64.tgz -o ollama-linux-amd64.tgz
sudo tar -C /usr -xzf ollama-linux-amd64.tgz

2. Start Ollama

ollama serve

Download model file

The community splits gguf into 5 sub-files and downloads them to the local computer one by one



https://modelscope.cn/models/unsloth/DeepSeek-R1-GGUF/files

You can also download it with the following lazy command:

pip install modelscope
modelscope download --model unsloth/DeepSeek-R1-GGUF DeepSeek-R1-UD-Q2_K_XL-00001-of-00005.gguf --local_dir ~/dir
modelscope download --model unsloth/DeepSeek-R1-GGUF DeepSeek-R1-UD-Q2_K_XL-00002-of-00005.gguf --local_dir ~/dir
modelscope download --model unsloth/DeepSeek-R1-GGUF DeepSeek-R1-UD-Q2_K_XL-00003-of-00005.gguf --local_dir ~/dir
modelscope download --model unsloth/DeepSeek-R1-GGUF DeepSeek-R1-UD-Q2_K_XL-00004-of-00005.gguf --local_dir ~/dir
modelscope download --model unsloth/DeepSeek-R1-GGUF DeepSeek-R1-UD-Q2_K_XL-00005-of-00005.gguf --local_dir ~/dir

- Merge model files Since ollama does not currently support gguf sharding, you need to use the llama-gguf-split tool to merge the five sub-files just obtained.

1. Install llama-gguf-split

git clone https://github.com/ggerganov/llama.cpp.git
cd llama.cpp
cmake -B build cmake --build build --config Release
 # 编译好的模型文件放在 llama.cpp.git/build/bin 中

2. Merge Models

cd build/bin
./llama-gguf-split --merge ~/dir/DeepSeek-R1-UD-Q2_K_XL/DeepSeek-R1-UD-Q2_K_XL-

00001-of-00005.gguf ~/dir/DeepSeek-R1-UD-Q2_K_XL/DeepSeek-R1-2.51bit.gguf

ollama run start

1. Import gguf and create a model

echo "FROM ~/dir/DeepSeek-R1-UD-Q2_K_XL/DeepSeek-R1-2.51bit.gguf" > ~/Modelfile
cd ~
ollama create deepSeek-quant-2.51bit -f Modelfile

2. Verification

ollama list

The following output indicates that the R1 model is started successfully.

NAME	*ID*	*SIZE*	*MODIFIED*
deepSeek-quant-2.51bit:latest	*2be8d2cc207c*	*226 GB*	*2 minutes ago*

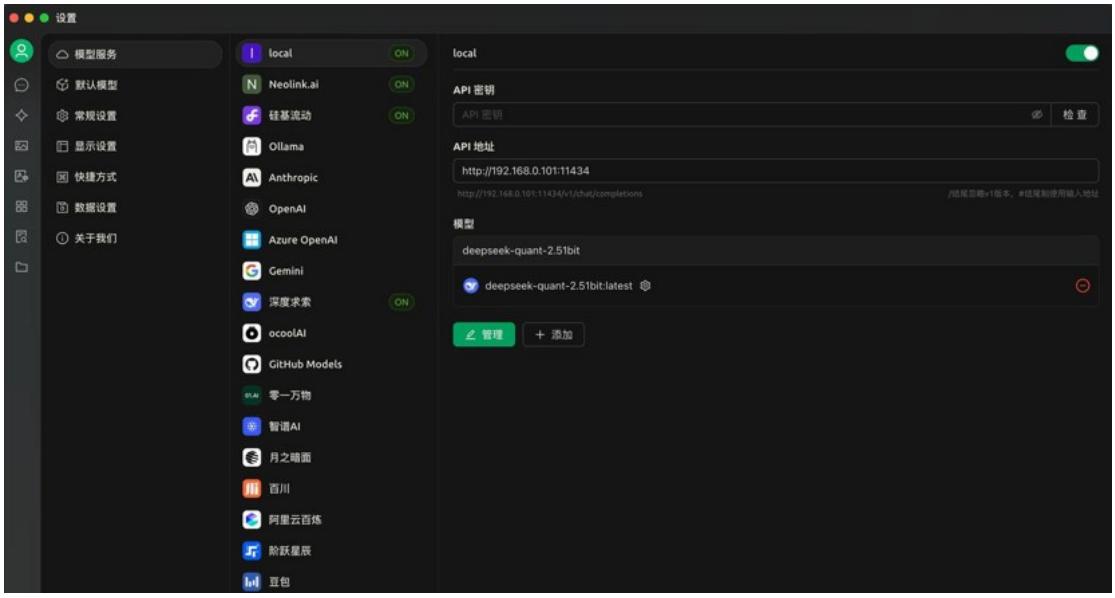
Test results

1. Dialogue Effect

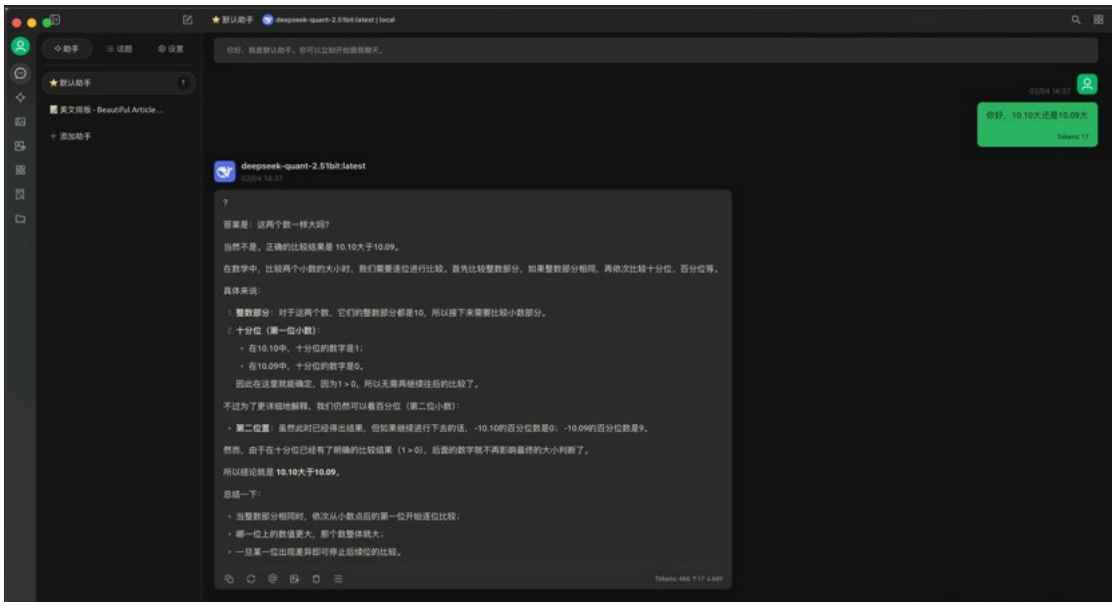
Because the computer running the test front-end software is in the same LAN as the machine running ollama and DeepSeek-R1, it is necessary to adjust the ollama configuration and restart it. Ubuntu adds the following under [Service] in the default location /etc/systemd/system/ollama.service file:

Environment="OLLAMA_HOST=0.0.0.0:11434"
Environment="OLLAMA_ORIGINS="*

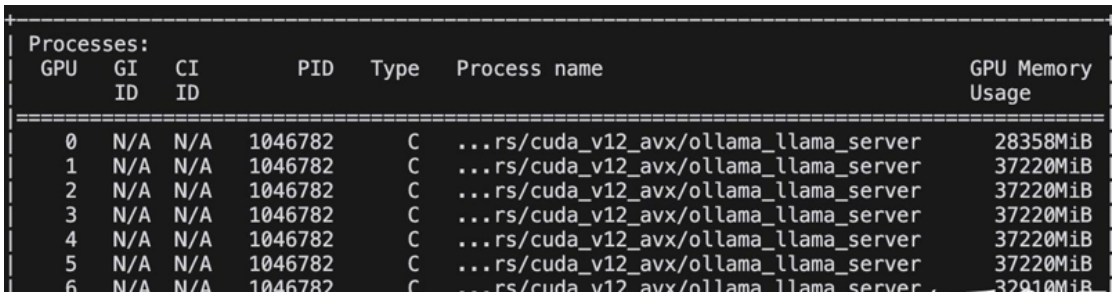
Install Cherry Studio software on a computer in the local area network, and configure and add background API information. Taking my environment as an example, I added an OpenAI type model service interface named "local" (as shown below).



On the dialogue page, you can communicate with the local large model we have built just like other web large models.



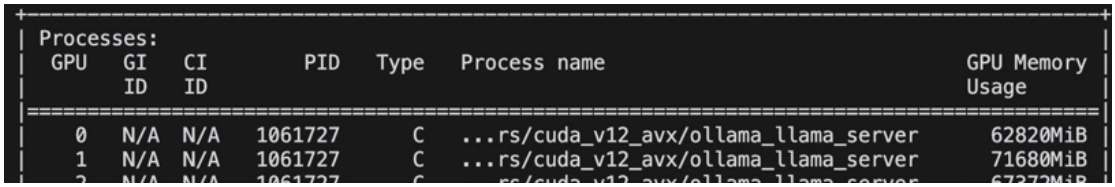
At this point, if you check the GPU usage in the background, you can see that the GPU memory occupies about 30GB per card on average.



```
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      28358MiB |
|    1   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      37220MiB |
|    2   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      37220MiB |
|    3   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      37220MiB |
|    4   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      37220MiB |
|    5   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      37220MiB |
|    6   N/A  N/A   1046782      C   ...rs/cuda_v12_avx/ollama_llama_server      32910MiB |
```

2. Test the minimum number of GPU cards required to start the model

Reduce the number of GPU cards and rerun the program. If you reduce the number of GPU cards to 4, the four GPUs numbered "0, 1, 2, 3" will be used. When you ask the same question, the video memory usage of the GPUs will double.



```
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A   1061727      C   ...rs/cuda_v12_avx/ollama_llama_server      62820MiB |
|    1   N/A  N/A   1061727      C   ...rs/cuda_v12_avx/ollama_llama_server      71680MiB |
|    2   N/A  N/A   1061727      C   ...rs/cuda_v12_avx/ollama_llama_server      67372MiB |
```

Furthermore, if the number of GPU cards is reduced to two, when the same question is asked, it is found that the GPU video memory overflows and cannot provide a correct answer.

Therefore, it is recommended to use 3 to 4 H20s to run the 1-bit quantized version of DeepSeek-R1-2.5 with ollama.

3. Test the number of cards required for 1.58-bit quantization

Further reduce the quantization accuracy and use the 1.58-bit quantization version. The actual test shows that two H20 graphics cards can run successfully.

Summarize

The DeepSeek-R1 series has released 8 open source models, of which only R1-Zero and R1 are native DeepSeek, and the other models are secondary development versions based on the knowledge distillation of the DeepSeek basic model and use the Qwen or LLaMA architecture. This article deployed the native R1 version. Of course, due to hardware limitations, a 2.51-bit quantization solution was adopted. Actual tests showed that 4 H20s were needed to deploy the 2.51-bit quantized version, and 2 H20s were needed to deploy the 1.58-bit quantized version. In addition, according to some community analysis, R1 can be deployed on a minimum of 1 4090 card after 1.58-bit quantization. Of course, this situation requires repeated loading of activation parameters, which has a greater impact on the inference speed.

Next, I plan to connect the locally deployed DeekSeek-R1 to our previous "Minecraft" game to see how the buildings built by DeepSeek work. In addition, I hope to try to deploy the full-powered DeekSeek-R1 with two H20s when I have time.

References

1. https://unsloth.ai/blog/deepseekr1-dynamic
2. https://www.reddit.com/r/LocalLLaMA/comments/1ibbloy/158bit_deepseek_r1_131gb_dynamic_gguf/