The Top Local LLMs You Can Actually Run at Home

6 min read · 1 day ago

Civil Learning

Running large AI models at home used to seem like a wild idea. Now, it's just… normal.

You don't need a data center or a fancy company setup. Just a few GPUs and a weekend to tinker, and you can run your own "mini ChatGPT" right on your desk.

Over the past few months, I've tested a variety of models. Some performed well, while others damaged my GPUs. Here are the ones that really worked — **my top local LLMs.**

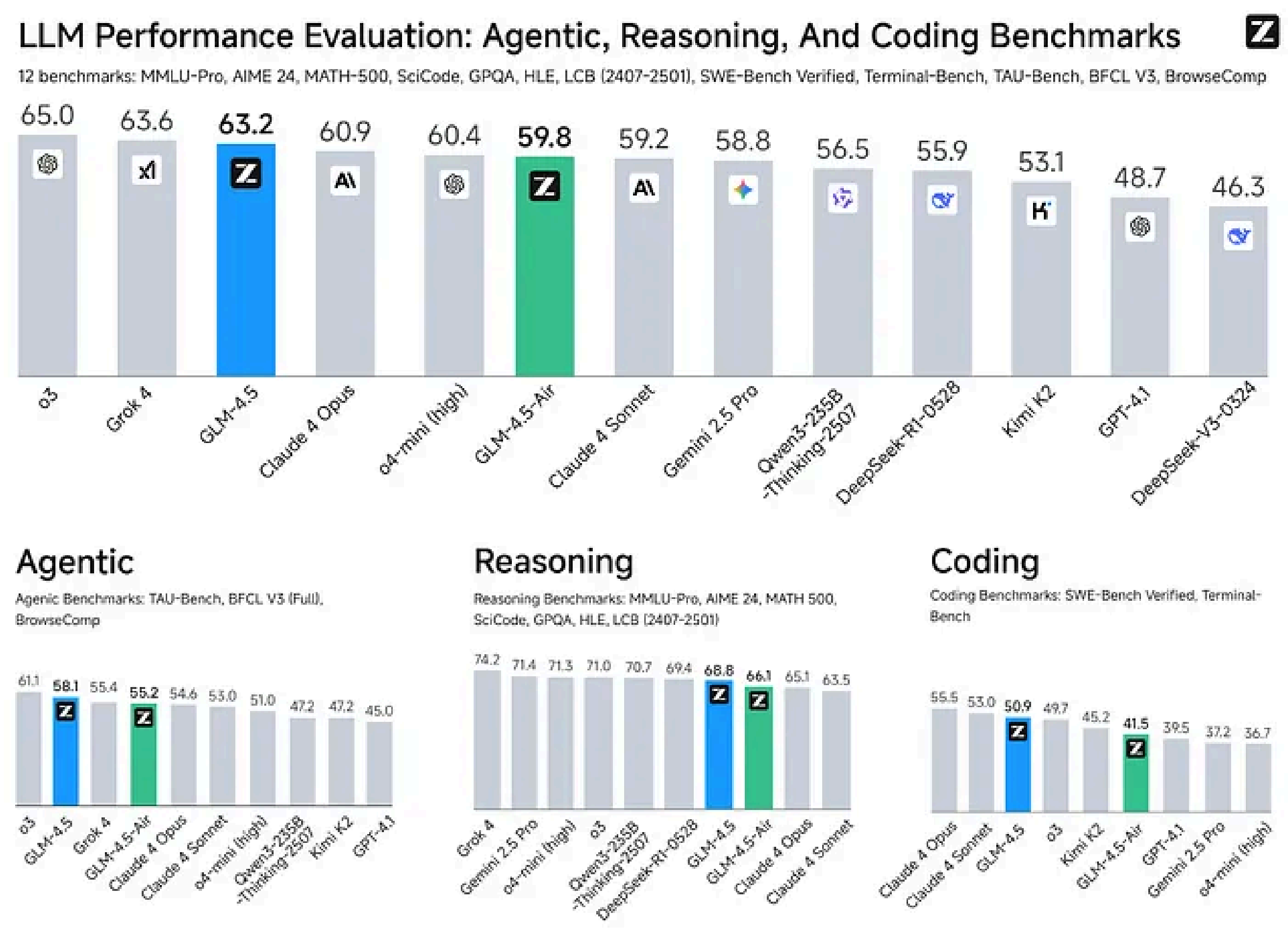
· · ·

GLM-4.5-Air — The Everyday Legend

If I had to choose a model to live with, this would be it. **GLM-4.5-Air** works.

It fits on four RTX 3090s and runs smoothly. Handles coding, planning, and daily tasks without hassle. Feels intelligent, sounds natural, and won't spike your power bill.

It's not the largest or most flashy model, but you find yourself using it more than anything else. That says a lot.



First, install the necessary packages according to `requirements.txt`.

```
pip install -r requirements.txt
```

transformers

Please refer to the `trans_infer_cli.py` code in the `inference` folder.

vLLM

- Both BF16 and FP8 can be initiated using the following code:

```
vllm serve zai-org/GLM-4.5-Air \
    --tensor-parallel-size 8 \
    --tool-call-parser glm45 \
    --reasoning-parser glm45 \
    --enable-auto-tool-choice \
    --served-model-name glm-4.5-air
```

If you're using 8x H100 GPUs and experience insufficient memory when running GLM-4.5 or GLM-4.6 model, you'll need `--cpu-offload-gb 16` (only applicable to vLLM).

If you encounter `flash infer` issues, use `VLLM_ATTENTION_BACKEND=XFORMERS` as a temporary replacement. You can also specify `TORCH_CUDA_ARCH_LIST='9.0+PTX'` to use `flash infer` (different gPUs have different TORCH_CUDA_ARCH_LIST values; please verify the specific value accordingly).

SGLang

- BF16

```
python3 -m sglang.launch_server \
  --model-path zai-org/GLM-4.5-Air \
  --tp-size 8 \
  --tool-call-parser glm45  \
  --reasoning-parser glm45 \
  --speculative-algorithm EAGLE \
  --speculative-num-steps 3 \
  --speculative-eagle-topk 1 \
  --speculative-num-draft-tokens 4 \
  --mem-fraction-static 0.7 \
  --served-model-name glm-4.5-air \
  --host 0.0.0.0 \
  --port 8000
```

- FP8

```
python3 -m sglang.launch_server \
  --model-path zai-org/GLM-4.5-Air-FP8 \
  --tp-size 4 \
  --tool-call-parser glm45  \
  --reasoning-parser glm45  \
  --speculative-algorithm EAGLE \
  --speculative-num-steps 3  \
  --speculative-eagle-topk 1  \
  --speculative-num-draft-tokens 4 \
  --mem-fraction-static 0.7 \
  --disable-shared-experts-fusion \
  --served-model-name glm-4.5-air-fp8 \
  --host 0.0.0.0 \
  --port 8000
```

- PD-Disaggregation

The following is a straightforward method to implement PD-Disaggregation using a single machine equipped with multiple GPUs, where P and D each utilize 4 GPUs.

```
python -m sglang.launch_server --model-path zai-org/GLM-4.5-Air  --disaggregati
python -m sglang.launch_server --model-path zai-org/GLM-4.5-Air  --disaggregati
python -m sglang_router.launch_router --pd-disaggregation --prefill http://127.
```

. . .

GPT-OSS-120B — The Heavyweight Brain

This one's massive. Like, *genuinely* massive. However, once it's operational, it feels nearly as powerful as GPT-5.

It's sharp, logical, and consistent — ideal for reasoning and coding. However, it's not as effective for creative writing. Quite dry.

You'll need high-end hardware to run it — like 8x A6000s or a small server setup. But if you have the equipment, this model feels like having a private AI lab at home.

```
ollama run gpt-oss:120b
```

. . .

GPT-OSS-20B — The Speed Monster

If speed is more important than raw power, choose **GPT-OSS-20B**.

It’s incredibly fast. Ideal for quick bug fixes, short scripts, or just experimenting. Doesn’t hang, doesn’t overcomplicate.

You don’t get very deep reasoning, but for quick tasks, it’s unbeatable. It feels responsive and lightweight, making it enjoyable to use.

. . .

Qwen3–30B-A3B — The Smart One

This model is similar to that friend who knows random facts about everything. **Qwen3–30B-A3B** isn't the fastest, but it's packed with knowledge.

Ask it about old programming tricks or trivia, and it'll tell you more than you expected. It's calm, clear, and dependable — ideal for writing or research.

Feels somewhat like a walking encyclopedia, but in a good way.

The code of Qwen3-MoE has been included in the latest Hugging Face, and we recommend using the most recent version.

With `transformers<4.51.0`, you will encounter the following error:

```
KeyError: 'qwen3_moe'
```

This code snippet shows how to use the model to generate content from given inputs.

```python
from transformers import AutoModelForCausalLM, AutoTokenizer

model_name = "Qwen/Qwen3-30B-A3B"
# load the tokenizer and the model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",
    device_map="auto"
)
# prepare the model input
prompt = "Give me a short introduction to large language model."
messages = [
    {"role": "user", "content": prompt}
]
text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True,
    enable_thinking=True # Switches between thinking and non-thinking modes. De
)
model_inputs = tokenizer([text], return_tensors="pt").to(model.device)
# conduct text completion
generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=32768
)
output_ids = generated_ids[0][len(model_inputs.input_ids[0]):].tolist()
# parsing thinking content
try:
    # rindex finding 151668 (</think>)
    index = len(output_ids) - output_ids[::-1].index(151668)
except ValueError:
    index = 0
thinking_content = tokenizer.decode(output_ids[:index], skip_special_tokens=Tru
content = tokenizer.decode(output_ids[index:], skip_special_tokens=True).strip(
print("thinking content:", thinking_content)
print("content:", content)
```

For deployment, you can use `sglang>=0.4.6.post1` or `vllm>=0.8.5` or to develop an API endpoint compatible with OpenAI:

- SGLang:

```
    python -m sglang.launch_server --model-path Qwen/Qwen3-30B-A3B --reasoning-pars
```

- vLLM:

```
    vllm serve Qwen/Qwen3-30B-A3B --enable-reasoning --reasoning-parser deepseek_r1
```

For local use, applications such as Ollama, LMStudio, MLX-LM, llama.cpp, and KTransformers also support Qwen3.

. . .

Qwen3-Coder-30B — The Chill Coder

You don’t need a big rig for this. **Qwen3-Coder-30B** runs on a single 16GB GPU (quantized) and still codes like a pro.

It’s ideal for small projects, debugging, or quick explanations. Simple, reliable, and handles large files without choking.

If you enjoy coding but want to avoid building a server farm, this is the ideal option for you.

. . .

GLM-4.6 — The Beast

Alright, this one’s in a league of its own.

GLM-4.6 is for users with serious gear — like 4x Pro 6000 Max-Qs. It handles complex workflows and toolchains effortlessly.

If you run agents, automation, or multi-tool setups, this thing flies. It’s not a hobby model — it’s a “clear your weekend and prepare to be amazed” model.

. . .

The Wrap-Up

Running local LLMs in 2025 is no longer just a geeky thing; it’s now practical.

Choose something that matches your setup, work, and patience level.

```
| Model            | What It’s Good At  | Hardware Needed | Vibe           |
|------------------|--------------------|-----------------|----------------|
| GLM-4.5-Air      | Balanced and fast  | 4x 3090s        | All-rounder    |
| GPT-OSS-120B     | Deep reasoning     | 8x A6000s       | Power brain    |
| GPT-OSS-20B      | Super quick        | 1-2 GPUs        | Snappy worker  |
| Qwen3-30B-A3B    | Knowledge-rich     | 2-3 GPUs        | Trivia master  |
| Qwen3-Coder-30B | Coding helper       | 16GB VRAM        | Light & handy  |
| GLM-4.6         | Advanced workflows | 4x Pro 6000s     | Absolute beast |
```

Start small if you're new. **GPT-OSS-20B** or **Qwen3-Coder-30B** are perfect for getting your feet wet. Once you're hooked (and you *will* be), you'll probably start dreaming about upgrading your rig.

Once you run your own model locally, there's no turning back. It just feels right — fast, private, and yours.

Llm · Llm Applications · Llm Agent · AI · Ai Agent

Written by Civil Learning

3.2K followers · 6 following

We share what you need to know. Shared only for information.

No responses yet