Coding Nexus · Follow publication

Member-only story

Local LLMs 101: What Really Happens When You Run an AI Model on Your Own Machine

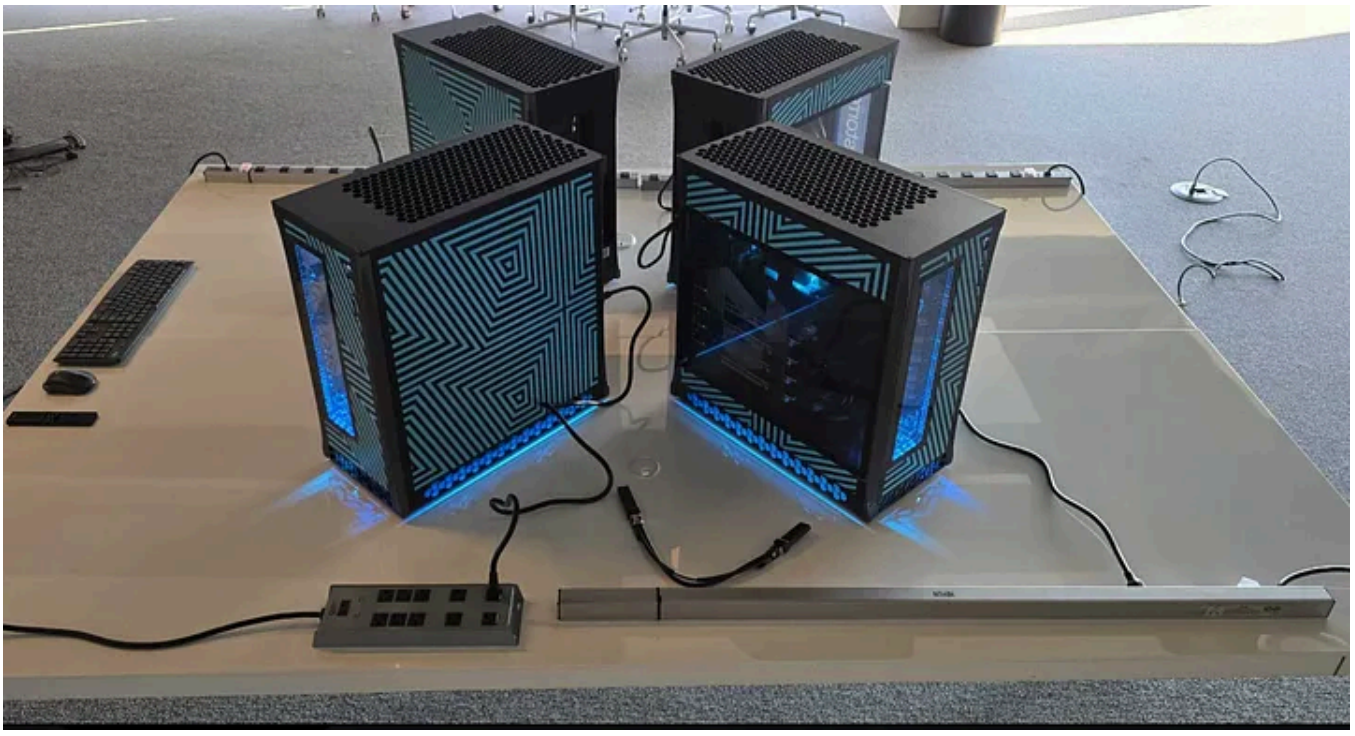
5 min read · 5 days ago

Algo Insights Following

Listen Share More

If you've ever tried running a large language model locally — and your GPU fans started screaming — you've probably wondered: *what's really happening behind the scenes?*

Let's explain. No hype, no academic jargon. How these models actually work when you click "Run".



. . .

The Basics: What “Running a Model” Actually Means

When you run a model, you’re performing **inference**. That is simply the technical term for *using the model to make predictions*.

In practice, inference involves the model predicting the **next token** (a segment of text) based on your input and everything it has generated so far.

You can think of it like this small loop:

```
sequence = ["Hello"]
while True:
    next_token = model.predict(sequence)
    sequence.append(next_token)
```

That’s it. The model keeps asking, *“Given all this, what comes next?”* Then it spits out one token at a time. Repeatedly.

. . .

Tokens Aren’t Words

This part confuses many people. Tokens are not the same as words. They are simply the segments of text that the model actually processes.

For example:

- `"hello"` might be one token, or maybe two.
- `"internationalization"` could be five, six, maybe eight tokens.

It depends on the **tokenizer**, which is the tool that breaks up text. Common examples include **BPE (Byte Pair Encoding)** and **SentencePiece**.

So when you hear about a model's **context window** — say 8K or 32K — that's the number of tokens it can "see" at once, not words.
Longer context = more memory, more compute, and slower generation.

. . .

Behind the scenes: It's All Math

At its core, every step is simply this:

```
next_token = f(weights, sequence)
```

- The **weights** are the model's learned knowledge — billions of tiny numbers.
- The **sequence** is what you've typed and what it's generated so far.
- The function `f` is the **transformer architecture**, the brain doing the math.

Every time you produce a new token, the model reanalyzes the full sequence.
That's why it's not blazing fast — it's literally recalculating context at every step.

. . .

What's Inside a Model

When you download a model like *Mistral 7B* or *LLaMA 3*, you're actually getting a little ecosystem:

1. **The architecture** — the “skeleton” (layers, heads, attention, all that jazz)
2. **The weights** — the learned parameters (the actual knowledge)
3. **The tokenizer** — how text gets split into tokens
4. **The config file** — shapes, limits, and metadata
5. **Chat template** — the hidden formatting that makes chat models behave properly

If you ever loaded a base model and received pure nonsense, you probably missed the chat template.
It's like trying to communicate with a robot in the wrong language.

. . .

Model Sizes and the VRAM Reality Check

You’ll see models labeled like “7B” or “13B”.
That “B” stands for **billion parameters** — basically, how many numbers the model learned.

Each parameter consumes space on your GPU's memory (VRAM).
Here’s the rough math:

```
| Precision | Bytes per Parameter | 7B Model Size |
| --------- | ------------------- | ------------- |
| FP16      | 2 bytes             | ~14 GB        |
| INT8      | 1 byte              | ~7 GB         |
| INT4      | 0.5 byte            | ~3.5 GB       |
```

Add a few gigabytes for overhead and **KV cache** — the memory that stores your conversation context.

So if you have, say, an 8 GB GPU, you’ll want a **4-bit quantized model**. Otherwise, it’s game over (or worse: the model starts swapping to CPU — pure pain).

. . .

Transformers: The Real Engine

At the core, there's the **transformer** — the architecture that powers these models. It's designed for sequences, like sentences or code.

Each layer does a few things:

- **Self-attention**: decides which previous tokens matter most
- **MLPs (neural subnets)**: refine the representation
- **LayerNorms + residuals**: keep everything stable
- **RoPE (rotary position embeddings)**: tell the model where each token sits

It's like a multi-layered thought process — each pass gradually enhances understanding.
Stack enough layers, and you develop reasoning, memory, and even personality.

. . .

How the Model Chooses Words

Once the model calculates probabilities for all possible following tokens, it must *select one*.

That's where **decoding** strategies come in:

- **Greedy**: always choose the highest probability (predictable, robotic)
- **Temperature**: adds randomness; higher = more creative
- **Top-k / Top-p**: limit the pool of choices to make text more natural
- **Repetition penalty**: prevents the "I love cats. I love cats. I love cats." loop

Example:

```
token_probs = model(sequence)
next_token = sample(token_probs, temperature=0.8, top_p=0.9)
```

That little sampling trick is the difference between “meh” and “this actually sounds human.”

. . .

Quantization: The Memory Cheat Code

Quantisation means “store numbers with fewer bits.” It is the most effective trick for fitting large models into smaller GPUs.

```
Type Memory Quality     FP16 High Excellent   INT8 Medium Good   INT4 (NF4/GPTQ
```

For most people, **4-bit** is the ideal choice — low memory use and a barely noticeable drop in quality.
Even the **KV cache** can now be quantized, which greatly benefits long chats.

. . .

Running Locally: The Real Options

Depending on your configuration, you can select how to deploy your model.

- **vLLM** → crazy fast, handles multiple users
- **llama.cpp** → simple and portable, great for quantized models
- **ExLlama V2/V3** → optimized for speed
- **PyTorch / Transformers** → flexible, great for experimentation
- **FastAPI / Flask** → if you want to build your own mini API

Local doesn’t always mean “offline.” You can still serve your model to apps — it just s*tays* under *your* control.

. . .

When Things Go Wrong

Let's be honest. Running models locally isn't always smooth. Here are the typical "oh no" moments:

```
| Issue               | Why It Happens                                        |
| ------------------- | ----------------------------------------------------- |
| OOM (Out of Memory) | Model or context too big; try 4-bit or smaller model |
| Gibberish output    | Wrong chat template or high temperature               |
| Sluggish speed      | CPU offloading, no FlashAttention, bad drivers        |
| Suspicious files    | Avoid `.bin`; use `safetensors` for safety            |
```

. . .

Why Bother Running Locally?

Because once you get it working, it's *yours*.

No API limits.
No cloud fees.
No data is leaving your machine.

You can tweak temperature, context size, quantization — everything — until it behaves exactly how you want.
It's like finally driving the car instead of just being a passenger.

. . .

Quick Setup Checklist

Before you hit "Run," double-check:

1. Model fits in VRAM
2. Precision (4-bit if needed)
3. Runtime (vLLM, llama.cpp, etc.)
4. Correct chat template

5. Decoding tuned (temp, top-p)
6. Tested on your use case
7. Served locally via API or script

That's it. You're officially running your own AI model.

. . .

Final Thoughts

Running LLMs locally isn't some mysterious art. It's mainly about **memory, formatting, and patience**.

Once you understand how tokens, weights, and VRAM work together, you can run almost any modern model — from tiny 3B ones to massive 70B beasts (if your hardware supports it).

It's somewhat like building your own spaceship. You don't need to understand every bolt — just enough to keep it flying.

And once it's running?
You'll see: the "magic" everyone mentions is just math, memory, and one token at a time.

Llm | Llm Applications | Llm Agent | AI | Ai Agent

Following

Written by Algo Insights

3.5K followers · 6 following

Algo Insights: Unlocking the Future of Trading with Code, Data, and Intelligence.
https://linktr.ee/Algo_Insights

Responses (2)

Bgerby

What are your thoughts?

David Bui
2 days ago

Now that's (Algo) insights 😊 . Easy to understand.

7 1 reply Reply

Andrey Shevel
Nov 1

Nice explanation!

Reply

More from Algo Insights and Coding Nexus

In Coding Nexus by Algo Insights

How to Build Your Own AI Rig for Running Local LLMs (Gemma, Mistral, Qwen, GPT-OSS and Llama)

About three months ago, I realised I was utterly dependent on companies that didn’t care about anything except power, Money, and control.

Oct 8 358 14

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You'll Install This Year

I didn't expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 382 15

In Coding Nexus by Civil Learning

The Guy Who Let ChatGPT Trade for Him—and Somehow It Worked

You know how everyone says, "Don't let AI touch your Money"? Well, someone on Reddit decided to ignore that.

Oct 8 188 8

In Coding Nexus by Algo Insights

4 Open-Source Tools That Made Me Rethink My Dev Setup

I've been coding for a while now. Most of the tools we use every day... they've been the same for years. Editors, browsers, frameworks. Just...

Sep 3 450 9

See all from Algo Insights

See all from Coding Nexus

Recommended from Medium

In Towards AI by Teja Kusireddy

We Spent $47,000 Running AI Agents in Production. Here’s What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic’s Model Context Protocol (MCP) are revolutionary. But...

Oct 16 1.8K 49

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You’ll Install This Year

I didn't expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 382 15

In AI Software Engineer by Joe Njenga

Cursor 2.0 Has Arrived — And Agentic AI Coding Just Got Wild

Cursor has released version 2.0 , bringing the most powerful agentic AI we have seen yet, more autonomous than ever before,here's what's...

3d ago 325 6

In Data Science Collective by Ida Silfverskiöld

Agentic AI: Single vs Multi-Agent Systems

Building with a structured data source in LangGraph

4d ago 450 9

In Artificial Intelligence in Plain English by Surendra Pandar

Every Paid AI—Now FREE & UNLIMITED (100% Legal)

The founder raised $5.7M to make this possible

Oct 21 75 4

Agen.cy

20+ Genius Ways Power Users Are Using Claude Code Right Now

Here are 10+ ways power users are using Claude Code🧵

Oct 23 33 1