Run LLMs on Old Phones: A Practical Guide to Mobile AI Inference

March 6, 2026

Download this guide as PDF

> **Quick Answer:** Yes, old phones can run local LLMs. A 6GB RAM phone from 2021 (Pixel 6, Galaxy S21) handles 1B-3B quantized models at 3-6 tok/s through llama.cpp in Termux or PocketPal AI. An 8GB phone (Pixel 7 Pro, iPhone 15 Pro) runs 3B-4B models usefully. It's slow compared to a GPU, but it's free hardware you already own, it works offline, and the setup takes 10 minutes. For always-on server tasks, a Raspberry Pi 5 is better. For a portable offline chat buddy, the phone wins.

There's a Pixel 6 in my kitchen drawer. It's been there since I upgraded, doing nothing. Turns out it has a better processor for AI inference than a Raspberry Pi 5, 6GB of RAM, and a battery that keeps it running without a power supply.

If you have an old phone sitting around from 2020 or later, you can run a local LLM on it. The models are small, the speed is modest, and you won't be replacing your desktop setup. But for offline questions, voice transcription, or just the satisfaction of seeing an AI run on hardware you were about to recycle, it works better than you'd expect.

What actually runs on a phone

Four options exist right now, and they're not equally good.

llama.cpp in Termux (Android, recommended)

Termux turns your Android phone into a Linux terminal. Install it, build llama.cpp, and you're running GGUF models the same way you would on a desktop. No root required.

This is the most flexible option. You get the full llama.cpp feature set: any GGUF model, adjustable context length, server mode if you want to hit it from another device. The official Android docs walk through the build process.

One important detail: get Termux from F-Droid, not the Play Store. The Play Store version is outdated and will give you problems.

PocketPal AI (Android + iOS)

PocketPal AI is the easiest path if you don't want to touch a terminal. It's a React Native app that runs GGUF models on-device with a chat interface. Download from the App Store or Play Store, pick a model, and go.

It supports pulling models directly from HuggingFace (including gated models with an HF token as of v1.9.0). You can swap between models, customize inference settings, and see real-time performance metrics. The app handles memory management automatically, offloading the model when you switch away.

The tradeoff: less control than Termux, and you're limited to what the app exposes. But for "I want to try this in 2 minutes," PocketPal is hard to beat.

MLC LLM (Android + iOS)

MLC LLM compiles models specifically for mobile hardware and can use the GPU via OpenCL (Android) or Metal (iOS). In theory, GPU acceleration should be faster than CPU. In practice, benchmarking research found that MLC LLM on mobile GPUs actually performs worse at prefill than llama.cpp on CPUs for some workloads. The decode speed is competitive, but the setup is heavier.

Worth trying if you have a phone with a strong GPU (Adreno 750 or newer). Skip it if you want the simplest path.

whisper.cpp for voice transcription

If your use case is speech-to-text rather than chat, whisper.cpp runs in Termux and handles offline transcription on phones with 2-4GB of free RAM. The small and base models work well on older hardware. There's also a dedicated Whisper app on F-Droid if you prefer not to build from source.

Performance by phone generation

I'll be honest: phone LLM inference is slow. But "slow" means different things at different model sizes. Here's what to expect.

Decode speed (tok/s) by chipset

These numbers come from academic benchmarking of Llama 2 7B at 4-bit quantization via llama.cpp on CPU. Smaller models scale roughly 2-3x faster.

Chipset (example phone)	Year	Decode tok/s (7B Q4)	Decode tok/s (3B Q4, est.)
Dimensity 9300 (Vivo Pad3 Pro)	2024	8.2	~18-20
Snapdragon 8 Gen 3 (Xiaomi 14 Pro)	2024	6.6	~14-16
Snapdragon 8+ Gen 1 (Galaxy S22 Ultra)	2022	3.5	~8-10
Snapdragon 870 (older midrange)	2021	1.7	~4-5
Tensor G1 (Pixel 6)	2021	~2-3	~5-7

For context: reading speed is about 4-5 words per second. At 5+ tok/s on a 3B model, the text streams faster than you read it. At 2 tok/s on a 7B model, you're watching paint dry.

What fits in your phone's RAM

Android and iOS keep about 40-50% of RAM for the system. A "6GB RAM phone" gives you roughly 3-3.5GB for the model.

Phone RAM	Usable for model	What fits	Practical?
4GB (2019-2020 budget)	~2GB	0.5B-1B Q4	Barely. One-line answers.
6GB (2021, Pixel 6, Galaxy S21)	~3-3.5GB	1B-3B Q4	Yes. Simple questions, translation, short tasks.
8GB (2022+, Pixel 7 Pro, iPhone 15 Pro)	~4-5GB	3B-4B Q4	Genuinely useful. Phi-3.5 3.8B, Gemma 2B, Qwen 2.5 3B.
12GB (flagships, Galaxy S23 Ultra)	~7-8GB	7B Q4, maybe 9B Q3	Real models. Qwen 3.5 9B at Q2/Q3 might squeeze in.

The sweet spot is 8GB phones running 3B models. You get coherent answers at usable speed, and 3B models in 2026 are far more capable than 3B models from two years ago. Qwen 3.5 4B and Gemma 3 4B are genuinely good at simple tasks.

Phone vs Raspberry Pi 5

This comes up constantly on r/LocalLLaMA. Both are cheap ARM devices with limited RAM. But they're different tools for different jobs.

	Old phone (8GB)	**Raspberry Pi 5 (8GB)**
CPU	Snapdragon 8 Gen 1+ (fast ARM cores)	Cortex-A76 (decent, slower)
NPU/AI accelerator	Yes (Hexagon, ANE, etc.)	No (unless AI HAT+)
RAM bandwidth	LPDDR5, ~50 GB/s	LPDDR4X, ~34 GB/s
3B Q4 decode speed	~8-10 tok/s	~4-5 tok/s
Thermal throttling	Yes, after 5-10 min	Less (passive cooling available)
Power	Battery (portable)	Wall power required
Always-on server	Awkward	Built for it
SSH/headless	Possible but clunky	Native, easy
Screen + chat UI	Built in	Need external monitor
Cost	$0 (you already own it)	$80-100 with case and power supply
OS flexibility	Android/iOS (limited)	Full Linux

The Pi wins when you want a headless always-on server. SSH in, run Ollama or llama.cpp as a daemon, hit it from your laptop over the network. The Pi doesn't overheat in a drawer and gives you full Linux flexibility. It's the better home lab device.

The phone wins when you want portability, offline use, or don't want to spend money. The faster ARM cores and higher RAM bandwidth mean a 2022 flagship phone runs 3B models about twice as fast as a Pi 5. You already have a screen and a chat UI. And the phone costs nothing because it was going to sit in a drawer anyway.

If you're buying hardware specifically for this, get the Pi. If you're raiding the junk drawer, use the phone.

The 10-minute setup (Termux + llama.cpp)

Here's the fastest path to running a model on an old Android phone.

1. Install Termux from F-Droid (not the Play Store).

2. Install build tools:

```
apt update && apt upgrade -y
apt install git cmake clang
```

3. Build llama.cpp:

```
git clone https://github.com/ggml-org/llama.cpp
cd llama.cpp
cmake -B build
cmake --build build --config Release -j4
```

This takes 3-5 minutes on a modern phone. Older phones, maybe 10.

4. Download a model. Start small. For a 6GB phone:

```
curl -L "https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct-GGUF/resolve/main/qwen2.5-1.5b-instru
  -o ~/qwen-1.5b-q4.gguf
```

For an 8GB phone, grab a 3B model instead:

```
curl -L "https://huggingface.co/bartowski/gemma-2-2b-it-GGUF/resolve/main/gemma-2-2b-it-Q4_K_M.g
  -o ~/gemma-2b-q4.gguf
```

5. Run it:

```
./build/bin/llama-cli -m ~/qwen-1.5b-q4.gguf -c 2048 \
  -p "Translate this to French: 'Where is the nearest train station?'"
```

Or start the server for a web UI:

```
./build/bin/llama-server -m ~/qwen-1.5b-q4.gguf -c 2048 --host 0.0.0.0
```

Then open `http://localhost:8080` in your phone's browser.

Tips:

- Keep models in `~/` (Termux home directory) for best file access speed
- Start with `-c 2048` context. Going higher eats RAM fast.
- If Termux gets killed by Android's memory manager, go to Settings > Apps > Termux > Battery > Unrestricted

---

Use cases that actually make sense

I'm going to be direct about what works and what doesn't on phone hardware.

Works well:

- Offline translation (1.5B-3B models handle common language pairs)
- Quick factual questions when you have no internet
- Voice transcription via whisper.cpp (the small model runs on anything)
- Personal journaling with AI prompts
- Learning and experimenting with local AI

Works, but barely:

- Summarizing short text (keep input under 500 words)
- Simple code snippets (3B+ models only)
- Roleplay and creative writing (slow but functional on 8GB+ phones)

Don't bother:

- RAG or document search (not enough RAM for embeddings + model)
- Coding assistants (too slow, context too limited)
- Long multi-turn conversations (context fills up fast at 2K)
- Image generation (not happening on phone hardware)
- Anything that needs 7B+ quality answers reliably

The honest pitch: a phone LLM is a novelty that occasionally proves useful. It's not replacing your desktop setup or your ChatGPT subscription. But there's something satisfying about asking a question and getting an answer from a model running on hardware that was collecting dust. And when you're on a plane with no WiFi, that 3B model in your pocket feels like a minor superpower.

Bottom line

Raid the junk drawer. If you have an old phone with 6GB+ RAM from 2021 or later, you can have a working local LLM in 10 minutes. PocketPal if you want the easy path, Termux + llama.cpp if you want control.

Set your expectations: 3B models at 5-10 tok/s, useful for quick questions and offline translation, not for replacing your daily AI workflow. That's the deal, and for $0 and a Saturday afternoon, it's a good one.

Related guides

- Best Models Under 3B Parameters
- What Can You Run on 8GB VRAM?
- Running LLMs on Mac M-Series
- CPU-Only LLMs: What Actually Works
- Voice Chat with Local LLMs: Whisper + TTS

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/run-llms-old-phones-mobile-inference/