Title: Llama.cpp

URL: https://en.wikipedia.org/wiki/Llama.cpp

PageID: 76900589

Categories: Category:Free and open-source software, Category:Large language models, Category:Open-source artificial intelligence

Source: Wikipedia (CC BY-SA 4.0).

---

llama.cpp is an open source software library that performs inference on various large language models such as Llama . [3] It is co-developed alongside the GGML project, a general-purpose tensor library. [4]

Command-line tools are included with the library, [5] alongside a server with a simple web interface . [6] [7]

Background

Towards the end of September 2022, Georgi Gerganov started work on the GGML library, a C library implementing tensor algebra . Gerganov developed the library with the intention of strict memory management and multi-threading. The creation of GGML was inspired by Fabrice Bellard 's work on LibNC. [8]

Before llama.cpp, Gerganov worked on a similar library called whisper.cpp which implemented Whisper , a speech to text model by OpenAI . [9]

Development

llama.cpp began development in March 2023 by Georgi Gerganov as an implementation of the Llama inference code in pure C/C++ with no dependencies. This improved performance on computers without GPU or other dedicated hardware, which was a goal of the project. [3] [10] [11] llama.cpp gained traction with users who lacked specialized hardware as it could run on just a CPU including on Android devices. [10] [12] [13] While initially designed for CPUs, GPU inference support was later added. [14] As of August 2025 it has more than 85,000 stars on GitHub. [15]

In March 2024 Justine Tunney introduced new optimized matrix multiplication kernels for x86 and ARM CPUs, improving prompt evaluation performance for FP16 and 8-bit quantized data types. [16] These improvements were committed upstream to llama.cpp. [16] Tunney also created a tool called llamafile that bundles models and llama.cpp into a single file that runs on multiple operating systems via the Cosmopolitan Libc library also created by Tunney which allows C/C++ to be more portable across operating systems. [16]

On Apr 30, 2024 FlashAttention was introduced.

On Apr 10, 2025 libmtmd was introduced, which reinvigorated support for multimodal models that has been stagnant previously.

Architecture

Llama.cpp supports multiple hardware targets including x86 , ARM , Metal , BLAS , BLIS , SYCL , MUSA , CUDA , HIP , CANN , OpenCL , RPC and Vulkan (version 1.2 or greater). [17] [18] [19] [20] These back-ends make up the GGML tensor library which is used by the front-end model-specific llama.cpp code. [21] llama.cpp makes use of several CPU extensions for optimization: AVX , AVX2 and AVX-512 for X86-64 , and Neon on ARM. Apple silicon is an important target for the project. [15] [22]

Llama.cpp supports a variety of features aimed at inference on edge devices such as:

ahead of time model quantization and on-the-fly kv-cache quantization. [23]

speculative decoding . [7]

partial offloading of model layers to system RAM , allowing devices to load models that would be too large to fit solely in GPU VRAM .

In addition, llama.cpp supports a variety of features and APIs for frontend communication such as:

OpenAI-compatible endpoints like v1/chat/completions .

grammar-based output formatting as JSON . [11]

GGUF file format

The GGUF (GGML Universal File) [26] file format is a binary format that stores both tensors and metadata in a single file, and is designed for fast saving, and loading of model data. [27] It was introduced in August 2023 by the llama.cpp project to better maintain backwards compatibility as support was added for other model architectures. [14] [28] It superseded previous formats used by the project such as GGML.

GGUF files are typically created by converting models developed with a different machine learning library such as PyTorch . [27]

Design

The format focuses on quantization, the act of reducing precision in the model weights. This can lead to reduced memory usage, and increased speed at the expense of lower model accuracy. [29] [28]

GGUF supports 2-bit to 8-bit quantized integer types; [30] common floating-point data formats such as float32 , float16 , and bfloat16 ; and 1.56 bit quantization. [5]

This file format contains information necessary for running a GPT-like language model such as the tokenizer vocabulary, context length, tensor info and other attributes. [31]

Byte-level structure (little-endian)

Metadata block

Tensors info block

References