GSoC Proposal

PROJECT #15: GGUF Reader in OpenVINO for Direct GGUF Execution

About Me

Full Name
Karnav Jayeshkumar Shah

University / Current Enrollment
Vellore Institute of Technology – Andhra Pradesh

Degree Program
Bachelor of Technology (B.Tech) in Computer Science and Engineering with specialization in Artificial Intelligence and Machine Learning

Timezone
UTC +05:30 (India Standard Time)

Short Bio

I am a second-year B.Tech student with a strong interest in machine learning systems, model optimization, and efficient inference runtimes. I began contributing to open source in December 2025, primarily focusing on the DNN module of OpenCV.

Since starting my open-source journey, I have made **93** contributions, including 43 pull requests, with multiple merged contributions improving robustness, error handling, and model parsing in OpenCV's deep learning pipeline.

Through this work I became particularly interested in how deep learning frameworks translate model representations (such as ONNX graphs) into optimized runtime execution graphs. This interest naturally led me toward OpenVINO, which focuses on optimized inference and model execution across hardware platforms.

When I learned about Google Summer of Code, I started exploring projects related to model inference pipelines and discovered OpenVINO's work on supporting new model formats. The GGUF reader project stood out because it directly connects model format parsing, graph translation, and runtime optimization — areas that closely match my interests and previous contributions.

Programming Experience

I have experience programming in:

- C
- C++
- Java
- Python

My primary focus is C++ development in performance-critical systems, particularly within machine learning inference pipelines. I regularly work with Git-based collaborative workflows and large open-source codebases.

Relevant Technical Experience

My contributions to OpenCV focused mainly on improving components of the DNN inference pipeline, including:

- Improvements to the Mean-Variance Normalization (MVN) layer implementation (Pull Request #28308)
- Error handling improvements for tensor conversions (Pull Request #28265)
- Enable LayerNorm with more than one output (Pull Request #28431)
- Improvements to CUDA-based DNN execution error handling (Pull Request #28231)
- Experimental work enabling support for quantized ONNX layers in the dynamic execution engine (Pull Request #28543)

These contributions helped me understand how model formats, computation graphs, and runtime backends interact in real-world inference frameworks.

About the Project

Project Choice:

PROJECT#15: GGUF Reader in OpenVINO for Direct GGUF Execution

Mentors: Mustafa Cavus, Ravi Panchumarthy

Why I Choose This Idea

My work in OpenCV introduced me to how inference frameworks load and execute machine learning models. While working with the ONNX importer and DNN module internals, I became interested in how different model formats are translated into runtime execution graphs.

While exploring OpenVINO I discovered that OpenCV can already use OpenVINO as a backend for optimized inference. This connection motivated me to explore OpenVINO's architecture more deeply.

When reviewing the GSoC project ideas, the GGUF reader project particularly stood out. My previous experience working with ONNX models made me curious about how newer model formats such as GGUF are translated into executable computation graphs.

To better understand this pipeline, I began studying llama.cpp and the GGML computation graph representation used internally by the runtime. I explored how the OpenVINO backend inside llama.cpp translates GGML operations into OpenVINO models.

This exploration helped me understand the importance of building a generic GGUF reader that leverages the GGML computation graph instead of manually reconstructing model architectures. The project aligns closely with my interest in machine learning systems and efficient runtime execution.

Abstract of the Proposed Solution

The current GGUF reader in OpenVINO GenAI reconstructs model architectures manually by parsing metadata from GGUF files and building the OpenVINO model layer by layer.

While functional, this approach requires explicit C++ implementations for each supported model architecture, making it difficult to automatically support new GGUF models.

The proposed GGUFReaderV2 introduces a different approach based on dynamic graph translation.

Instead of reconstructing model architectures manually, GGUFReaderV2 will:

1. Use llama.cpp APIs to load the GGUF model and produce the GGML computation graph (`ggml_cgraph`).
2. Pass the resulting computation graph to GgmlOvDecoder, which converts the GGML graph into an OpenVINO **`ov::Model`** using the OpenVINO frontend infrastructure.
3. Integrate the resulting model into the OpenVINO GenAI **`read_model()`** pipeline for compilation and inference.This approach allows OpenVINO to automatically support a wide range of GGUF models without requiring manual architecture implementations for each new model topology.

Prototype Investigation and Findings

Before writing the proposal, I implemented an experimental prototype to validate the proposed architecture which is implied in [Pull Request #3449](#) The prototype attempted to:

1. Trigger a dummy decode operation inside llama.cpp.
2. Capture the generated GGML computation graph (`ggml_cgraph`).
3. Pass the captured graph to GgmlOvDecoder to generate an OpenVINO model.

During this process I discovered several important architectural challenges.

1) Tensor Naming Issue

- OpenVINO's decoder internally maps graph nodes using string identifiers. However, llama.cpp intentionally leaves many intermediate tensors unnamed to reduce overhead.
- As a result, multiple nodes appeared with empty identifiers, which caused collisions inside the decoder's internal mapping structures.
- As a temporary experiment, I used **`ggml_set_name`** to assign unique identifiers to unnamed nodes before translation, which improved graph translation stability.

2) Fragmented Graph Construction

- Further investigation revealed that **`llama_decode`** generates multiple small computation graphs during inference rather than a single monolithic model graph.
- This meant that the graph captured by the prototype only represented a partial portion of the model, causing missing layers during translation.
- These observations highlight important architectural considerations for implementing **GGUFReaderV2** and helped guide the proposed implementation plan.

Detailed Timeline (350 Hours)

Community Bonding (May 1 – May 24)

- Finalize development environment
- Study current GGUF reader implementation
- Review OpenVINO frontend components
- Review mentor feedback on prototype

Deliverable: finalized design plan for GGUFReaderV2.

May 25 – June 14: Monolithic Graph Extraction & Prototype Skeleton

- Implement the base **`GGUFReaderV2`** class.

- Develop a low-level tracing mechanism to capture the complete **`ggml_cgraph`** directly from **`llama.cpp`** APIs, bypassing the **`llama_decode`** scheduler to solve the graph fragmentation issue.
- Integrate the captured monolithic graph with **`GgmlOvDecoder`**.

Deliverable: A functional backend capable of intercepting a complete, un-split GGML graph.

June 15 – July 5: Memory Mapping Refactor & Translation Core

- Overhaul **`TranslateSession`** to utilize absolute memory pointers (**`const struct ggml_tensor*`**) instead of **`std::string`** for node mapping.
- This completely eliminates the **`std::out_of_range`** tensor naming collisions caused by anonymous **`llama.cpp`** intermediate nodes.
- Ensure standard feed-forward and attention layers translate completely without dummy-name hacks.

Deliverable: 100% stable, collision-free GGML to OpenVINO graph translation.

July 6 – July 26: Stateful Execution & GenAI Integration

- Map **`llama.cpp`**'s stateful KV-cache operations into OpenVINO's stateful memory formats (**`ov::pass::MakeStateful`**).
- Ensure the generated **`ov::Model`** connects cleanly to OpenVINO GenAI's Continuous Batching (CB) pipeline.
- Integrate **`GGUFReaderV2`** securely into the public **`read_model()`** API.

Deliverable: Users can load GGUF models directly through GenAI for dynamic inference.

July 27 – August 9: Quantization, Testing, and Hardware Validation

- Implement native OpenVINO translation support for standard GGUF quantization formats (e.g., Q4_K, Q6_K).
- Implement a C++ regression test suite comparing **`GGUFReaderV2`** logits against the original **`llama.cpp`** backend to assert mathematical precision.
- Audit the generated OpenVINO IR for compatibility with Intel NPU plugins.

Deliverable: Validated, highly accurate inference with robust error handling.

August 10 - August 16: — Documentation and Finalization

- Draft comprehensive architectural documentation for **`GGUFReaderV2`**.
- Create a developer guide detailing how to extend OpenVINO frontend support for future GGML operators.
- Finalize code cleanup and merge remaining PRs.

Deliverable: Production-ready GGUF translation pipeline merged into the main branch.

Final Submission & Evaluation (August 17 – August 24)

- Final Deliverable: Production-ready, validated GGUF translation pipeline merged into the OpenVINO main branch, ready for final mentor evaluation.

How I Know OpenVINO

I first encountered OpenVINO while working with the DNN module of OpenCV, where OpenVINO can be used as an optimized backend for inference execution. This motivated me to explore OpenVINO's architecture and its role in efficient model execution across hardware platforms.

Professional Development

My long-term goal is to work on **machine learning systems** and inference infrastructure. Participating in GSoC through the OpenVINO project would allow me to gain hands-on experience working on production-grade inference systems, contributing to open-source software used in real-world AI applications.

Other Summer Career Development Plans:

I do not have any competing internships, university classes, or major commitments this summer. Successfully completing this GSoC project and integrating **`GGUFReaderV2`** into OpenVINO is my absolute priority and full-time focus for the duration of the program.

Why You Should Pick Me

I believe I am a strong candidate for this project because:

- I already have experience contributing to large C++ machine learning codebases such as **OpenCV**.
- My contributions focused specifically on inference pipeline components.
- I have already explored the **GGUF / GGML** pipeline and implemented a prototype investigating this integration.

- Finally, I plan to continue contributing to OpenVINO long after GSoC. ML system engineering and inference optimization have always fascinated me, and I am deeply passionate about doing this work.

Prerequisite

Relevant work and prototype exploration related to this project can be found in my contributions and experimental pull requests exploring GGUF integration.

- PR Link (core-work): Pull Request #3449
- PR Link (submodule):Pull Request #55
- GitHUB Link: shahkarnav115-beep (Karnav Shah)