Adding a Custom CPU or GPU Backend to LM Studio

Overview

This document describes how to add a custom `llama.cpp` backend to LM Studio. It applies to both CPU-only builds (with AVX or AVX1) and GPU builds (Vulkan). The procedure allows LM Studio to run on older CPUs, use a custom Vulkan GPU backend, or, analogously, a CUDA backend (requires modern GPU or bfloat16 patching).

Steps Taken

1. **Build `llama.cpp` locally.**

   - CPU build example:

     ```
     ggml-base.dll
     ggml-cpu.dll
     ggml_llamacpp.dll
     ```

   - GPU/Vulkan build example:

     ```
     ggml-base.dll
     ggml-cpu.dll # optional CPU support
     ggml-vulkan.dll
     ggml_llamacpp.dll
     liblmstudio_bindings_vulkan.node
     llm_engine_vulkan.node
     ```

2. **Locate LM Studio backend directory.** Example:

   ```
   C:\Users\Admin\.lmstudio\extensions\backends\llama.cpp-win-x86_64-vulkan-avx2
      -1.48.0
   ```

3. **Duplicate and rename the backend folder.**

   - Use a descriptive name for your build flags. Example:

     ```
     llama.cpp-win-x86_64-vulkan-avx-1.48.0
     ```

   - This prevents overwriting existing backends.

4. **Edit backend metadata files.**
   - `backend-manifest.json`: update `"required_features"` to match your build (e.g., `"AVX"` instead of `"AVX2"`).
   - `display-data.json`: update the display name for clarity:

     ```
     "displayName": "Vulkan llama.cpp (Windows, AVX)"
     ```

   - Optional: For CPU-only builds, change display name to:

     ```
     "displayName": "CPU llama.cpp (Windows, AVX)"
     ```

5. **Copy build outputs to backend folder.**
   - Example PowerShell snippet (CPU or Vulkan):

     ```
     $backend = "$env:USERPROFILE\.lmstudio\extensions\backends\llama.cpp-win-
         x86_64-vulkan-avx-1.48.0"
     $build = "C:\Users\Admin\source\llama.cpp\build-vulkan\bin"

     Copy-Item "$build\ggml-base.dll" -Destination $backend -Force
     Copy-Item "$build\ggml-cpu.dll" -Destination $backend -Force
     Copy-Item "$build\ggml-vulkan.dll" -Destination $backend -Force
     # Optional: only overwrite if needed
     # Copy-Item "$build\ggml_llamacpp.dll" -Destination $backend -Force
     # Copy-Item "$build\liblmstudio_bindings_vulkan.node" -Destination $backend -
         Force
     # Copy-Item "$build\llm_engine_vulkan.node" -Destination $backend -Force
     ```

   - For CPU-only backends, skip Vulkan files.

6. **Refresh LM Studio.** The new backend will appear with your custom display name and use the locally-built binaries.

7. **Optional: CUDA backend.** Repeat the process with a CUDA-configured `llama.cpp` build. Note: Ensure GPU supports bfloat16 or patch the backend for older cards.

Usage Notes

- **For simplicity, just paste the pre built version to backends**
- General idea: copy DLLs from local build to a uniquely-named LM Studio backend folder, adjust metadata, and refresh.
- CPU builds: AVX, AVX1, or no SIMD support can be configured in CMake.
- GPU/Vulkan builds: requires Vulkan SDK; allows running larger models on GPU.
- CUDA builds: requires modern GPU; may need bfloat16 patching for older hardware.
- This method preserves LM Studio stability while using optimized builds.