项目报告 PT.1

**如何在本地部署大模型？**

"本地部署大模型" 指的是将大语言模型 (LLM) 的运行环境搭建在自己的电脑、服务器或局域网内的设备上，而不是依赖第三方的云服务（如 OpenAI 的 API、Google Cloud、AWS 等）。这样做的好处包括数据隐私、无按量付费、离线可用性以及对模型运行环境的完全控制。

本地部署面临的主要挑战通常是 硬件要求，尤其是显卡 (GPU) 和显存 (VRAM)。大模型参数量巨大，加载和运行它们需要大量的计算资源和显存来存储模型的权重和激活值。

如下是在本地部署大模型的一般方法和流程：

评估硬件能力： 这是第一步也是最关键的一步。

GPU (显卡): 大部分大模型的计算都依赖 GPU 加速。你需要一块性能较好、显存 (VRAM) 足够大的独立显卡。显存是决定你能运行多大模型的关键。例如，一个 7B (70亿参数) 模型可能需要 6GB-8GB 以上的显存，一个 13B 模型可能需要 10GB-12GB 以上，更大的模型需要更多。即使模型被量化，显存依然是主要瓶颈。

内存 (RAM): 运行模型也需要一定的系统内存来加载模型的部分权重和处理数据。通常建议至少 16GB 或 32GB 以上。

存储空间: 模型文件本身很大，可能需要几十GB甚至上百GB的硬盘空间。

选择合适的模型：

模型大小: 根据硬件能力选择。有各种参数量的开源模型，如DeepSeek、 Llama 系列、Mistral、Qwen、Yi 等。小参数量（如 7B）更容易在消费级显卡上运行，大参数量（如 70B）则需要专业级硬件或更强的消费级硬件。

模型格式： 原始模型通常是 PyTorch 或 TensorFlow 格式。为了方便本地部署，出现了很多优化格式。

利用量化模型： 这是目前最常用的在消费级硬件上运行大模型的方法。

什么是量化？ 量化是将模型的权重和/或激活值从高精度（如 FP16 浮点数）转换为低精度（如 8-bit 或 4-bit 整数）。这可以显著减小模型文件大小和显存占用，同时尽量保持模型性能。

常见的量化格式：

GGML/GGUF: 这是 llama.cpp 项目及其衍生工具使用的格式，非常流行，支持 CPU 和 GPU 混合运行，对显存要求相对较低。

GPTQ/AWQ: 也是常见的量化算法和格式，通常需要较好的 GPU 支持。

选择运行框架/软件： 有很多工具可以帮助我们在本地加载和运行这些模型。

llama.cpp 及其衍生: 是一个非常强大的 C++ 库，最初用于运行 Llama 模型，现在支持很多其他模型。它特别擅长利用 CPU 和 GPU 混合计算，非常适合在各种硬件上运行 GGUF 格式的模型。很多用户友好的工具都基于它开发，例如：

LM Studio, Ollama, LiteLLM GUI: 提供图形用户界面 (GUI)，下载模型、启动服务、与模型交互都非常简单。

Text Generation Web UI (oobabooga): 一个功能强大的 Web UI，支持加载各种格式的模型 (包括 GGUF, GPTQ, AWQ, PyTorch 等)，提供聊天、文本生成、参数调整等多种功能。

KoboldCpp: 另一个基于 llama.cpp 的工具，提供易用的界面。

Hugging Face transformers 库: 如果有足够的硬件（特别是运行非量化模型），可以直接使用 Hugging Face 的 transformers 库加载 PyTorch/TensorFlow 格式的模型。但这通常需要更多显存。

具体部署步骤 (以使用 GGUF 模型 + GUI 工具为例):

安装运行软件: 下载并安装选择的工具，如 LM Studio, Ollama 或 Text Generation Web UI。

下载模型文件: 在软件内置的模型库或从 Hugging Face 等网站下载选择的模型的 GGUF 格式文件。

加载模型: 在软件中选择下载的模型文件进行加载。软件会自动将模型加载到显存（如果足够）或内存中。

开始交互: 通过软件提供的聊天界面、API 或 Web UI 与模型进行对话或生成文本。

本地部署大模型需要合适的硬件，选择适合硬件的模型（通常是量化版本），并使用方便的运行工具。对于大多数普通用户而言，使用 Ollama, LM Studio 或 Text Generation Web UI 运行 GGUF 格式的量化模型是目前最便捷有效的方式。