基于 TorchProf 的深度学习模型性能分析

背景简介

TorchProf 是一个轻量级的 PyTorch 模型分析工具，用于逐层分析神经网络的性能指标，包括 CPU 和 CUDA 的执行时间、内存占用和调用次数等。通过 TorchProf，研究者可以高效定位性能瓶颈，优化模型结构。本次作业将引导学生使用 TorchProf 进行模型性能分析，并探索如何提高模型计算效率。

作业任务

任务 1：项目环境搭建

克隆 `TorchProf` 项目到本地开发环境。

任务 2：理解工具功能，实现模型逐层性能分析

1. **模型选择**：
   - 使用 PyTorch 提供的 AlexNet 模型。
2. **分析内容**：
   - 使用 TorchProf 对 AlexNet 模型进行逐层分析，记录以下指标：
     - 每层的 CPU 总时间和 CUDA 总时间。
     - 每层的内存占用（CPU 和 CUDA）。
     - 每层的调用次数。

任务 3：性能瓶颈定位与优化

1. **性能瓶颈定位**：
   - 根据任务 2 的分析结果，找出 AlexNet 中计算时间最长或内存占用最高的一层。
2. **优化方案**：
   - 尝试通过以下方法优化性能瓶颈：
     - 减少卷积核数量或调整卷积核尺寸。
     - 替换为更高效的激活函数（如 ReLU）。
3. **优化后分析**：
   - 使用 TorchProf 对优化后的模型重新进行性能分析，比较优化前后的结果。

任务 4：扩展与应用

完成以下扩展任务，并进行对比实验：

1. **比较不同模型的性能**：
    - 使用 TorchProf 分别分析以下模型的性能（CPU 时间、CUDA 时间、内存占用）：
        - ResNet-18
        - MobileNetV2
2. **批量输入分析**：
    - 修改 TorchProf 的输入形状，模拟不同批量大小（如 batch_size=1 和 batch_size=32）对模型性能的影响。
    - 比较批量大小对内存占用和执行时间的影响。
3. **多输入模型分析**：
    - 构建一个多输入模型（如包含两个输入分支的自定义网络）。
    - 使用 TorchProf 分析该模型的性能，包括每个输入分支的计算时间和内存占用。
4. **动态输入形状分析**：
    - 使用 TorchProf 分析同一模型在不同输入分辨率（如 224x224 和 512x512）下的性能差异。
    - 比较输入分辨率对内存使用和计算时间的影响。

任务 5：实战模拟

本部分要求学生使用 `torchprof` 对一系列具有代表性但结构迥异的深度学习模型进行深入的性能剖析和比较分析：

1. **选择并准备模型与数据**:
    - 选择**至少三种**不同类型的、具有一定复杂度的预训练模型进行分析。推荐组合：
        - **经典 CNN**: ResNet-50 (研究残差连接和标准卷积的开销)
        - **高效 CNN**: MobileNetV3-Large (研究深度可分离卷积、SE 模块和瓶颈结构的效率)
        - **Transformer**: Vision Transformer (ViT-B/16) (研究自注意力机制和 MLP 块的性能特征)
    - 使用标准的图像数据集，例如 **Tiny ImageNet** 或 标准 **ImageNet 验证集的前 N 张图片。**
    - 准备数据加载器，能够提供指定批次大小 (Batch Size) 和输入分辨率 (Input Resolution) 的数据。
2. **实现细粒度的性能剖析**:
    - **核心挑战**: 编写脚本，使用 `torchprof` 对所选的每个模型进行详细的 **forward pass** 剖析。
    - **剖析要求**:
        - 记录**每个命名层 (named layer/module)** 的以下信息：
            - CPU 执行时间
            - GPU 执行时间 (如果 GPU 可用)

        - 参数数量
        - 浮点运算数 (FLOPs) / MACs (Multiply-Accumulate operations)
        - (如果 `torchprof` 支持且可靠) 峰值内存占用 或 内存增量 (需要仔细验证 `torchprof` 的内存报告能力，或结合 `torch.cuda.memory_allocated()` 等进行补充分析)
    - **系统性实验**: 对每个模型，在**不同条件下**进行剖析：
        - **不同硬件**: 至少在 CPU 和 GPU (若可用) 上分别运行剖析。
        - **不同输入分辨率**: 例如，使用 Tiny ImageNet 的标准 64x64，以及上采样到 224x224 (模型通常期望的尺寸)。
        - **不同批次大小**: 例如，Batch Size = 1, 8, 32 (需要考虑 GPU 显存限制)。
3. **数据聚合与可视化**:
    - 将 `torchprof` 输出的原始数据进行聚合和处理。
    - 为每个模型和每种剖析条件，生成清晰的可视化图表：
        - **资源分布饼图/条形图。**
        - **Top-K 耗时层**: 列出 CPU 和 GPU 上耗时最长的 Top-K (e.g., K=10) 层。

作业提交要求

1. **代码文件**:
    - 提交完整的代码文件，包括模型定义、性能分析和扩展任务的实现。
    - 确保代码注释完整，便于阅读和理解。
2. **实验结果**:
    - 提交实验日志文件和性能对比结果，以表格或图表形式展示。
    - 可视化结果图片整理到 `results/` 文件夹中。
3. **实验报告**:
    - 报告内容应包括任务描述、实现过程、实验结果和对比分析。
    - 报告格式不限。

附加说明

- **参考资料**:
    - TorchProf 官方文档：GitHub - awwong1/torchprof
    - PyTorch 官方文档：[Page Redirection]