一.问题简述

在深度学习训练中，Windows系统支持**CUDA**方法———将繁复的前馈和反向传播等运算交由GPU完成，提高了训练效率。然而，由于Windows采用Spawn机制[1]创建子进程，导致数据读入很难使用多进程进行。
与此同时，使用Fork机制[2]的Linux系统可以充分利用多进程的优势，加速数据读取速度；然而，如果在虚拟机（如VMware）上使用Linux，无法使用物理显卡，因而无法运行CUDA指令，往往会导致运算速度反而不如Windows。
此问题可以通过**WSL2**（Windows Subsystem for Linux）得到完美解决。
此外，本文也将讨论对于大数据集（图像等）和小数据集（以Titanic挑战赛为例），dataloader函数中num_workers的数量设置问题。

二.从Windows到Linux

1. Windows

Dataloader是PyTorch中用于加载数据的工具，它可以：

1. 批量加载数据（batch loading）
2. 打乱数据（shuffling）
3. 并行加载数据（多线程）
4. 自定义数据加载方式

这是最基本的Dataloader运行代码：

```
train_loader=DataLoader(
    dataset=dataset,        #数据集
    batch_size=32,          #批处理大小（Minibatch）
    shuffle=True,           #每个epoch是否打乱
    num_workers=0,          #并行线程数
    pin_memory=True         #数据加载器会将张量复制到CUDA固定内存中，加速CPU到GPU的数据传输
)
```

在Windows系统中，一旦将num_workers的数目设置为0以上，就必须将除了导入模块和类定义以外的“执行代码”放入以下代码下：

```
if __name__ == "__main__":
     ...
```

原因是子进程。Spawn模式下子进程会重新运行一遍程序，只有加入主进程才能执行的限制，才能避免套娃等问题的出现（如Dataloader如果放在外面，将反复创建子进程）。
然而容易发现，即便如此，代码运行效果仍然不佳（运行非常缓慢）。因此我们转投Linux测试。

2.Linux(VMware虚拟机)

我的虚拟机没配过Python的环境，因此列一下大致的工作：

Python环境

- 下载 wget：Linux下下载文件的工具

```
sudo apt install wget
```

- 下载 Miniconda（只包含最基本的conda管理器和Python环境，不预装大量库）

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/mini
```

- 初始化Shell

```
~/miniconda3/bin/conda init bash
```

- 重启终端

Pytorch环境

- 创建环境（指定 Python 3.8 或 3.9 比较稳定）：

```
conda create -n pytorch_env python=3.9
```

(遇到 Proceed (y/n)? 输入 y 回车)

- 激活环境：

```
conda activate pytorch_env
```

- 安装 PyTorch 和依赖库：
  由于是在虚拟机里（通常只有 CPU），安装 CPU 版本的 PyTorch 即可（包小下载快）：

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
pip install numpy matplotlib
```

这样，你就完成了环境搭建！

之后，可以先尝试运行原代码，再改变workers数目，都是可以运行的。

三. 一个发现

> 对于你的这个糖尿病数据集（非常小），num_workers=0 实际上可能比多进程更快，因为多进程启动和通信的开销远大于读取这几行数据的时间。——Gemini 3

在增加num_workers数目时，我们惊讶地发现，训练速度不但没有变快，反而变得很慢！
Gemini3 告诉了我们答案。因此，我们要特别小心地衡量是否需要更多workers。
此外，值得注意的是：我们打开多进程加载数据，是为了不让GPU空转——**理想状态下，GPU 计算时，CPU 已在准备下一批数据。** 然而，当计算体量小时，CPU加载的速度可能慢于计算速度，此时多进程就不必要了。

> 记于26年2月6日，杭州到乌鲁木齐火车上。

---

1. Spawn机制下，当 DataLoader 启动一个子进程（Worker）时，它实际上启动了一个全新的 Python 解释器。这个新的解释器**必须重新导入并执行一遍**脚本文件。 ↩
2. Fork机制下，子进程是父进程的精确副本（内存克隆）。子进程“天生”就拥有父进程已经运行过的所有变量、函数定义和状态，因此它**不需要重新运行**代码文件。 ↩