深度学习模型训练的批量大小与GPU内存之间的关系

来源

这个网站。

批量大小介绍

批量大小是在更新可训练模型变量（权重 w 和偏差 b ）之前用于训练模型的样本（例如图像、文本和视频）数量。也就是说，在每个训练步骤中，一批样本都会通过模型向前传播，然后向后传播以计算每个样本的梯度。然后，所有样本的梯度将被**平均**或**求和**，并且**平均值**或**求和值**将用作优化器公式的输入，优化器公式计算可训练模型变量的更新量。只有更新参数后，下一批样品才会经历相同的过程。

批量大小对模型的影响

1. 泛化：大批量可能会导致泛化不良（甚至陷入局部最小值）。

   泛化指的是神经网络在训练集之外的样本上表现得很好。糟糕的泛化（几乎是过度拟合）意味着神经网络在训练集之外的样本上表现不佳。

   （这句话应该是默认训练轮数是一定的，大批量数据集喂给模型，可以减小训练轮数）

2. 收敛速度：小批量可能会导致学习算法收敛缓慢。

   每个步骤中使用的权重更新量（使用一批样本计算得出）将确定下一批样本的起点。

   训练样本是每一步从训练集中随机抽取的，因此得到的梯度是基于部分数据的噪声估计。

   在单个批次中使用的样本越少，梯度估计的噪声越大且越不准确。也就是说，批次越小，单个样本对所使用的权重更新量影响就越大。换句话说，较小的批量可能会使学习过程变得更加嘈杂和波动，从本质上延长了算法收敛所需的时间。

如何确定最适合的批量大小?

调参。

模型训练时GPU到底存了什么?

1. Parameters — The weights and biases of the network.

参数——网络的权重和偏差。

2. Optimizer's variables — Per-algorithm intermediate variables (e.g. momentums).

   优化器的变量——每个算法的中间变量（例如动量）。

3. Intermediate calculations — Values from the forward pass that are temporarily stored in GPU memory and then used in the backward pass. (e.g. the activation outputs of every layer are used in the backward pass to calculate the gradients)

   中间计算——来自前向传递的值暂时存储在 GPU 内存中，然后在后向传递中使用。

   例如，每一层的激活输出都用于向后传递来计算梯度。

4. Workspace — Temporary memory for local variables of kernel implementations.

   工作空间——内核实现的局部变量的临时内存。

批量大小越大，在前向传播中通过神经网络传播的样本就越多。这会导致需要存储在 GPU 内存中的较大中间计算（例如层激活输出）。从技术上讲，激活的大小与批量大小线性相关。

如何在 GPU 内存较小时运行批次大小较大的数据集?

对一个批次的数据继续拆分成迷你批次

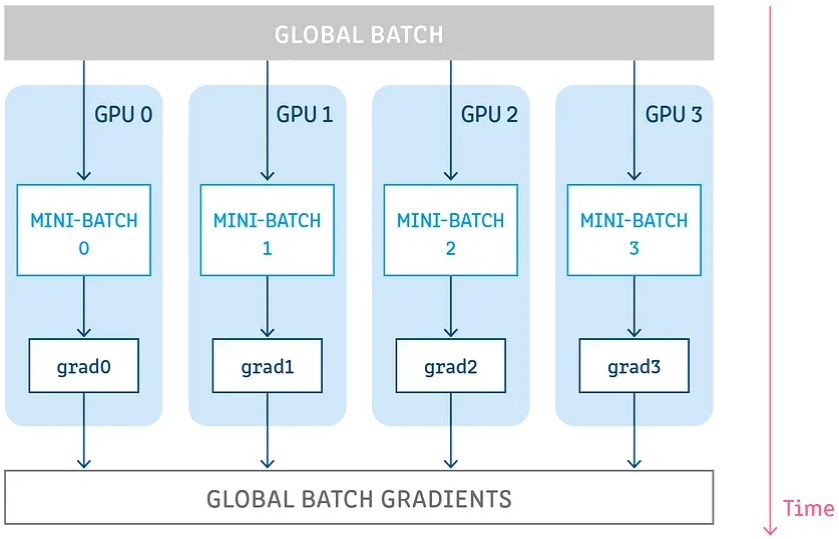
将样本批次分成较小的迷你批次（mini batch），每个迷你批次所需的 GPU 内存量可以满足要求。

这些迷你批次可以独立运行，在计算模型变量更新之前，应该对它们的梯度进行平均或求和。

Method 1: 数据并行

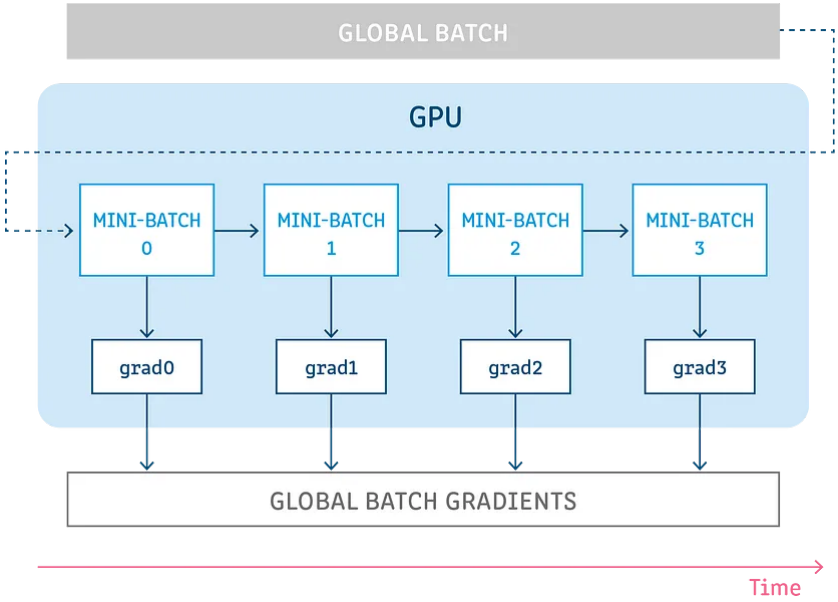
使用多个 GPU 并行训练所有迷你批次，每个迷你批次都在单个 GPU 上。

所有迷你批次的梯度都会累积，结果用于在每个反向传播结束时更新模型参数。



Method 2: 梯度累积

在累积梯度的同时，按顺序运行迷你批次。累积的结果用于在最后一个小批量结束时更新模型变量。



相比于前面的数据并行，梯度累积更适合单个GPU使用，因为每次顺序经过每个迷你批次都会产生梯度，而这个梯度是可以增量式更新的。

二者的对比

- 它们都不支持运行需要可用内存超过的 GPU 当前内存的模型（即使使用单个样本也是如此）。
- 批量规范化是在每个迷你批次上单独完成的，而不是在全局批次上完成的，这导致它们不完全等同于使用全局批处理大小运行相同的模型（我的理解是，分组完对每个小组求平均与全局求平均在有限样本情况下是存在差异的，这就好比“速度平均值”与“平均速度值”是不一样的）。

  注意：尽管全局批处理上的批处理规范化可以在数据并行中实现，但通常情况并非如此，而是单独完成的。
- 它们都允许我们增加全局批处理大小，同时仍然受到 GPU 内存的限制。

  （我的理解是，增加全局批处理大小，合理分配好迷你批次就没事）

- 虽然这两个选项非常相似，但梯度累积可以使用单个 GPU 按顺序完成，这使得它对无法访问多个 GPU 的用户或希望最大程度地减少资源使用的用户更具吸引力。
- 数据并行和梯度累积两者可以一起使用。使用多个 GPU，运行几个步骤并在每个 GPU 上先累积梯度，并在步骤结束时计算平均所有 GPU 的累积结果。