深度神经网络中的计算和内存带宽

来源

相关知识来源于这里。

原理介绍

> **Memory bandwidth** and **data re-use** in deep neural network computation can be estimated with a few simple simulations and calculations. Deep neural network computation requires the use of **weight data** and **input data**. Weights are neural network parameters, and input data (maps, activations) is the data you want to process from one layer to the next.

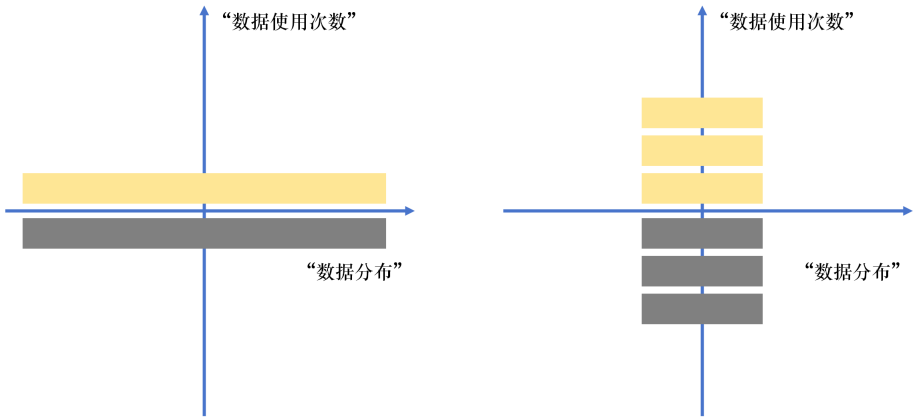
深度神经网络计算中的内存带宽（memory bandwidth）和数据重用（data reuse）可以通过一些简单的模拟计算来估计。深度神经网络计算需要使用**权重数据**和**输入数据。权重数据**是神经网络参数，**输入数据**（线性映射、非线性激活函数）是要从 *一个神经网络层* 传输到 *下一个神经网络层* 的数据。

> In general, if a computation re-uses data, it will require less memory bandwidth. Re-use can be accomplished by:
>
> - sending more inputs to be processed by the same weights;
> - sending more weights to process the same inputs.

如果计算重用数据，则需要较少的内存带宽。可以通过以下方式实现重用：

1. 发送更多输入，以由相同的权重进行处理；
2. 发送更多权重以处理相同的输入；
3. 如果没有输入或权重数据重用，则带宽对于给定应用程序处于最大值。

为什么“计算重用数据可以减小内存带宽”？这是我的理解：如果把 *数据分布* 作为横轴，数据的使用次数作为纵轴，那么就可以画出左边**“高带宽”（*数据分布* 广且数据使用次数低）**以及右边**“低带宽”（*数据分布* 窄但数据使用次数高）**的图。显然，当把数据“拆成”可重复利用的三份时， *数据分布* 变窄且数据的使用次数自然提高，以便保证处理到所有的数据，因此可以右边是“低带宽”。



对于一个深度学习模型而言，可以用到的数据就是**权重数据**和**输入数据**。所以减小内存带宽的策略就是对这两部分数据进行复用。复用**权重数据**就是频繁地把不同数据输入到一组权重数据中；复用**输入数据**就是频繁地把一组输入数据输入到不同权重数据中。最好的情况是同时复用**权重数据**和**输入数据**（似乎有点困难）。最差的情况是同时都不复用**权重数据**和**输入数据**，此时内存带宽达到最大值，类似于上图左半部分。

分析1：线性层

> Here a weight matrix of M by M is used to process a vector of M values with b bits. Total data transferred is: $b(M + M^2)$ or $\approx bM^2$.

这里使用 $M \times M$ 的权重矩阵来处理具有 b 位的 M 值向量。传输的数据总数为：$b(M + M^2)$ 或 $\approx bM^2$ 。

这是在执行线性层操作时，需要移动或处理的数据总量。公式 $b(M + M^2)$ 给出了在这个过程中涉及的总位数（即数据量）。bM 表示向量的位数总和，因为向量有 M 个值，每个值 b 位。bM^2 表示权重矩阵中所有元素的位数总和，因为矩阵有 bM^2 个元素，每个元素 b 位。

也就是说，线性矩阵层的计算包括**矩阵计算**和**计算结果传输**两个步骤。**矩阵计算**本质就是矩阵乘法，必然是对矩阵内的所有元素进行计算；**计算结果传输**即是生成新的向量。此外，当模型比较大的时候， M 值较大，此时 M^2 远远大于 M ，那么就意味着传输的来源主要是**矩阵计算**的**中间变量**。

> If the linear layer is used only for one vector, it will require to send the entire M^2 matrix of weights as computation occurs. If your system has T operations/second of performance, then the time to perform the computation is $\frac{bM^2}{T}$. Given than bandwidth $BW = \text{total data transferred}/\text{time}$, in case of linear layers $BW = T$.

如果线性层仅用于一个向量，则在进行计算时，它将需要发送**整个 M^2 权重矩阵**。如果系统具有 T 次操作/秒的性能，则执行计算的时间为 $\frac{bM^2}{T}$。给定带宽 $BW = \text{传输的总数据}/\text{时间}$ ，如果是线性层，$BW = T$。

这就说明，对于一个网络模型全是线性矩阵的话，系统**每秒能处理的计算操作**越多，那么**内存带宽就会越大**。

> This means that if your system has 128 G-ops/s of performance, you will need a bandwidth of more than 128 GB/s to perform the operation at full system efficiency (provided, of course that the system can do this!).

如果系统具有 128 G-ops/s 的性能，您将需要超过 128 GB/s 的带宽才能以全系统效率执行线性矩阵计算操作（当然，前提是系统可以做到这一点！作者在这里提到“全效率”这个概念，主要是想表达，如果网络模型大部分是线性层、且只操作单个向量的话，一般是比较低效的。

> Of course if you have multiple inputs for the same linear layer (multiple vectors that need to be multiplied by the same matrix) then: $BW = T/B$, where B is the number of vectors or Batch.

如果同一线性层有多个输入（需要乘以同一矩阵的多个矢量），那么 $BW = T/B$，其中 B 是向量数或批次数。

当线性矩阵处理的不是一个向量而是一个矩阵的时候，相当于复用**权重数据**——频繁地把不同数据输入到一组权重数据中，相当于多个向量把内存带宽"拆解"了，从公式 $BW = T/B$ 可以看出， B 越大内存带宽越小。我的理解是，transformer 中对一个序列 $(1 \times N)$ 进行嵌入处理，得到了维度比较高的矩阵 $(N \times d_{emb})$，这也是在降低 transformer 中的带宽。

分析2：卷积层

> For convolution operation, the bandwidth requirements are usually lower, as an input map data can be used in several convolution operation in parallel, and convolution weights are relatively small.

对于卷积运算，内存带宽要求通常较低，因为输入图数据可以并行用于多个卷积运算，并且卷积权值相对较小。

> For example: a 13×13 pixel map in a 3×3 convolution operation from 192 input maps to 192 output maps (as, for example, in Alexnet layer 3) requires: ≈4MB weight data and ≈ 0.1MB input data from memory. This may require about 3.2 GB/s to be performed on a 128 G-ops/s system with ≈99% efficiency (SnowFlake Spring 2017 version). The bandwidth usage is low is because the same input data is used to compute 192 outputs, albeit with different small weight matrices.

从 192 个输入映射到 192 个输出映射（例如在 Alexnet 第 3 层中）的 3×3 卷积运算中的 13×13 像素映射需要：≈4MB 权重数据和 ≈0.1MB 内存输入数据。这可能需要在 128 G-ops/s 系统上执行约 3.2 GB/s，效率约为 99%。带宽使用率较低是因为相同的输入数据用于计算 192 个输出，尽管具有不同的小权重矩阵。

每个输出像素需要进行一次完整的卷积运算，即九次乘加操作（对于 3×3 滤波器）。
因此，对于一个输出特征映射，需要**169（像素点）× 9（操作/像素点）= 1,521次**运算。
对于192个输出特征映射，总计算量为：**1,521 × 192 = 291,840次**运算。
如果系统运行在128 G-ops/s（即**128亿次运算/秒**），那么执行291,840次运算大约需要：**291,840 / (128 x 10^9)秒 ≈ 0.00228毫秒**。可见是很高效的。

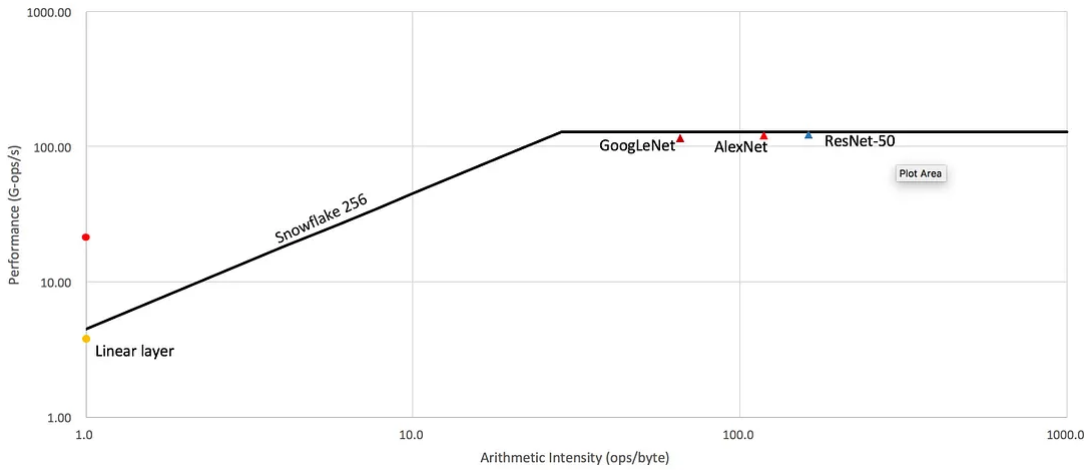
分析3：循环层

> Memory bandwidth for recurrent neural networks is one of the highest. Deep Speech 2 system or similar use 4 RNN layers of 400 size (see here and here). Each layer uses the equivalent of 3 linear-layer-like matrix multiplications in a GRU model. During inference the input batch is only 1 or a small number, and thus running these neural network requires the highest amount of memory bandwidth, so high it usually it is not possible to fully utilize even efficient hardware at full utilization.

循环神经网络的内存带宽是**最高**的。Deep Speech 2 系统或类似系统使用 4 个 400 大小的 RNN 层。每层相当于 GRU 模型中 3 个线性层的矩阵乘法。在推理过程中，输入批次只有 1 个或很小的数字，因此运行这些神经网络需要的内存带宽最高，即使是高效硬件通常也无法充分利用。

我的理解是，RNN 系列的工作是串行的，意味着**输入数据**部分难复用；此外，GRU 或者 LSTM 的模型本身复杂，**权重数据**也很难复用。这就导致推理时硬件效率低下，占用内存带宽太高。

总结



> This is the arithmetic intensity for our accelerator Snowflake. Arithmetic intensity is the number of operations performed on a byte of data. As you can see all neural network models tested perform at the maximum (roofline) efficiency of the device. On the other hand linear layers have very little data re-use and are limited by memory bandwidth constraints.

在加速器 Snowflake 下测试的算术强度。**算术强度是对一个字节的数据进行运算的次数。**可以看到，所有测试过的神经网络模型都达到了设备的最高（顶线、极限情况）效率。线性层的数据重复利用率非常低，并且受到内存带宽的限制。顶端的 GoogleNet 、 AlexNet 和 Resnet-50 都是卷积层为主的网络，效率和复用率都是很高的。