I/O 高效稀疏神经网络推理理论

Niels Gleinig、Tal Ben-Nun、Torsten Hoefler {niels.gleinig, talbn, torsten.hoefler}@inf.ethz.ch 苏黎世联邦理工学院

arXiv:2301.01048v1 [cs.DC] 3 Jan 2023

摘要——**随着机器学习模型的准确率快速提升，其对能量和计算资源的需求也随之增加。从底层来看，这些资源中的主要部分消耗在不同存储单元之间的数据移动上。现代硬件架构包含一种容量较小的快速存储器（例如缓存、寄存器），以及一种容量较大但访问成本高昂的慢速存储器（例如动态随机存取存储器）。我们只能处理存储在快速存储器中的数据，因此需要在这两个存储单元之间进行数据移动（输入/输出操作，或*I/Os*）。本文对稀疏前馈神经网络（FFNN）推理所需的 I/Os 进行了严格的理论分析。我们建立了能够将最优 I/Os 数量确定到 2 倍因子以内的界，并提出了一种使用处于该范围内的 I/Os 数量的方法。*I/O* 复杂度很大程度上由 FFNN 的少数几个高层属性决定（输入、输出、神经元和连接的数量），但如果希望更接近精确下界，则需要考虑具体实例的稀疏模式。在偏离 2-最优计算策略的基础上，我们展示了如何通过模拟退火进一步减少 I/Os 数量。作为这一结果的补充，我们提供了一种能够构造性地生成具有最大推理 I/O 效率的网络的算法。我们对这些算法进行了测试，并通过实验证实了我们的理论和算法贡献。在真实硬件上的实验中，相较于执行推理的标准方式，我们观察到最高可达 45× 的加速比。**

索引词—**神经网络推理、稀疏神经网络、I/O 复杂度、模拟退火。**

一、引言

几乎所有现代计算系统都采用不同的存储单元：有些速度快但容量小，有些容量大但速度慢。数据只能在快速存储器中处理。由于快速存储器的容量非常有限，在计算过程中，数据需要在快速存储器和慢速存储器之间移动。这些数据移动称为**输入/输出操作**（简称输入/输出操作）。它们在时间和能量方面的开销都很大。例如，在 NVIDIA V100 GPU 上，从全局内存加载数值所需的时钟周期最多可达 *514* 倍*5*× 于一次融合乘加操作（该操作对两个 32 位浮点值进行运算）14×（如果这些值存储在 L1 缓存中，则 [1]）。

在深度学习以及一般的科学计算中，输入/输出操作占据了大部分时间和能量开销 [2]。因此，重要的是以高效的方式执行输入/输出操作，让计算能够利用“数据局部性”和“数据重用”。我们需要以这样的方式安排计算步骤，使对相同数据的访问彼此接近，从而避免将数值从快速存储器（例如缓存）中逐出后又重新读取。因此，优化深度神经网络推理 [3]和训练 [4]的实现，往往需要针对现有内存架构进行定制——化实现，以实现高利用率。

本文采用一种理论模型来分析并优化输入/输出操作，该模型适用于此类现象所涉及的各种硬件。我们仅考虑不含共享连接的稀疏前馈神经网络（FFNN）。该模型涵盖但不限于经过剪枝的多层感知机（MLP）。我们假设 FFNN 表示为有向无环图（DAG）中的加权边列表，并且每个顶点还附带一个额外的值（对于输入神经元，该值为输入值；对于非输入神经元，该值为偏置）。在这种情况下，实现高 I/O 效率的主要方法是以高效的顺序“使用”连接。也就是说，使用某条连接时，输入神经元的值以及输出神经元的部分和理想情况下都应已位于快速存储器中。因此，我们需要找到一种连接顺序，使具有共同神经元的连接彼此相邻。

寻找一个关于该属性最优的顺序是一个组合问题。我们将证明，如果采用传统的计算顺序——这对应于对前馈神经网络推理的一种自然理解（例如，将其视为一系列矩阵向量乘法）——那么结果可能远非最优：读取 I/O 的次数最多可能是最优次数的两倍，而写入 I/O 的次数与最优次数之间的差距可能达到任意大的倍数。

机器学习中的计算具有一些特殊特征，在追求 I/O 效率时可以加以利用。与机器学习领域之外关于 I/O 高效计算的研究不同，我们有更多可以利用的自由度。例如，对于给定问题，通常存在多个可接受的解：两个不同的前馈神经网络可能产生不同的输出，但都能达到令人满意的准确率。因此，根据硬件调整神经网络架构是合理的。另一方面，硬件方面也存在额外的自由度，因为针对特定机器学习应用来寻找甚至构建硬件并不罕见 [5], [6], [3]。

因此，我们的研究旨在回答以下问题：（1）给定一个前馈神经网络和指定大小的快速存储器，执行该前馈神经网络推理所需的最少输入/输出操作次数是多少？（2）对于给定的前馈神经网络，执行推理并达到最少输入/输出操作次数所需的最小内存大小是多少（即，我们可以避免因内存不足而读写中间结果）？以及（3）对于指定大小的快速存储器，

哪些前馈神经网络架构可以实现输入/输出操作次数最少的推理？

A. 背景与相关工作

a) I/O 复杂度：机器学习领域之外，关于输入/输出高效算法的理论研究由来已久。Hong & Kung 的开创性工作 [7] 是最早正式研究输入/输出高效算法的工作之一。该工作引入了一种名为红蓝 *pebble* 游戏的理论模型，并利用该模型证明，标准矩阵乘法或快速傅里叶变换的调度方式应如何安排，才能使输入/输出操作次数在渐近意义上达到最小。此后，这一模型以多种方式得到扩展和改进。这些扩展被用于为排序、置换和矩阵转置等问题 [8]，以及稀疏矩阵与稠密向量乘法 [9], 和各种图算法 [10]，建立下界和相匹配的上界。不过，这些解通常只能在常数乘法因子的意义下达到最优，而且这些因子往往很大或无法精确确定。事实上，已有研究表明，寻找红蓝 pebble 游戏的精确最优解（对应于数据移动次数恰好达到最小值的计算调度）是 PSPACE 完全问题 [11]。此外，有人推测该问题甚至难以近似 [11], ，因为已知相关 pebble 游戏具有不可近似性 [12], [13]。

对于包含较大比例输入的常规计算，也存在关于输入/输出复杂度的下界 [14]。这些下界适用于本文所研究的问题。然而，与本文建立的下界相比，它们的粒度较粗（没有像我们这样区分读 I/O 和写入 I/O），并且通常也不够紧。

在机器学习专用算法中，I/O 效率在实践中已被确定为一个关键的性能因素。深度神经网络的性能建模通常根据运算次数 [15]以及对各个层进行基准测试所得固定值之和 [16], [17]推导得出。就深度神经网络评估的输入/输出复杂度而言，据我们所知，除本文外唯一的理论研究是 Demmel & Dinh [18], 的工作，他们为卷积层和池化层提供了通信下界。

b) 网络剪枝与稀疏度：深度神经网络剪枝（即在保持准确率的同时移除网络中的某些边或节点）在过去几十年中得到了广泛研究 [19], [20], [21]。对于边缘计算设备而言，剪枝具有重要的实际意义，因为它可以大幅减少内存占用和浮点运算次数。稠密（全连接）层一直是剪枝的主要目标，因为它们往往占据内存大小的很大一部分。例如，在 AlexNet [22], 中，最后的稠密层包含了 90% 的参数。此外，目前非常流行且在自然语言处理任务中取得最先进结果的 Transformer [23],还包含前馈神经网络，这些网络占据了整体规模的很大一部分。例如，BERT [23] 包含多个深度为 2、权重矩阵维度为1024×4096 和 4096×1024 的前馈神经网络，而 BERT 的大部分参数和推理计算时间都来自这些前馈神经网络 [24]。

剪枝网络有多种方法，例如，根据给定的权重阈值 [25], [26], ，剪除排名前-k 的值 [27], [28], ，或者通过逐步权重消除 [19]。无论采用哪种方法，对稠密层进行剪枝都会产生稀疏且非结构化网络，因此更难实现数据局部性。

c) 硬件架构：针对推理的硬件优化已有不少研究，尤其是针对稀疏前馈神经网络这一类。Sze 等人 [3] 对此进行了详细综述。具体方法包括：EDEN [29],使用近似 DRAM 来提高能效；Bhardwaj 等人 [30], 研究分布式环境中的推理通信；以及 EIE [5] 和 Eyeriss [31] 分别通过加速稀疏矩阵-向量乘法和游程编码，直接在压缩卷积神经网络上执行推理。尽管这些方法改善了深度神经网络处理的某些方面，但随着网络规模增大，这类架构最终也需要采用多级存储层次结构，而这正是我们的方法所要解决的问题。

B. 贡献

我们的主要理论贡献是一系列分析前馈神经网络推理中输入/输出操作的定理和命题。定理1给出了在给定的前馈神经网络架构上执行推理所需输入/输出操作数量的界限。命题 1 表明，这些界限都是最优的，即无法通过乘以任何非 1 的因子来进一步收紧。输入/输出操作总数的下界和上界相差至多一个乘法因子 2。定理1的证明具有构造性，展示了如何执行计算，使输入/输出操作数量落在这个 2-最优范围内。为了超越 2-最优性，我们的目标是通过有利地重排连接，进一步逼近精确的 最优值。我们采用模拟退火来解决这一问题。在第五节中，我们提出一种生成前馈神经网络的方法。根据定理 2，该方法能够完全刻画在给定内存大小下可用最少输入/输出操作执行推理的架构（即恰好达到下界）。因此，该方法可作为协同设计神经网络与硬件的有力工具。作为该定理的推论，我们得到一个关于最小可能内存大小的上界；达到该内存大小后，即可实现最高 I/O 效率的推理（也就是说，无需写入或读取任何临时值）。我们在涵盖多种参数设置（密度、深度、宽度、内存大小）的随机前馈神经网络上测试了这些算法。

II. 模型与符号

我们假设，初始时所有前馈神经网络参数和所有输入值都存放在慢速存储器中。这些数据恰好由以下三类数据组成：

- 带权连接。
- 每个非输入神经元对应一个偏置值。
- 每个输入神经元对应一个输入值。

每个连接由一个独立参数三元组描述（i，j，w_{ij}），其中 i 是输入神经元，j 是输出神经元，w_{ij} 是权重（权重不会重复使用或共享）。我们将连接数记为 W，神经元数记为 N，输入神经元数记为 I，输出神经元数记为 S。在理论分析中，我们

假设连接（描述它们的完整三元组）、神经元上的数值（偏置、部分和、激活函数的输出）以及计算中涉及的所有其他数值，都是大小相同的数据类型。因此，整个问题规模由权重、偏置和输入神经元的总数决定： $W+(N-I)+I=W+N$。我们有一个快速存储器，在任意时刻可以容纳一定数量的（此数据类型的）值，该数量由参数 M 给出，并假设 $M \geq 3$。在这个模型中，单个参数M 定义了整个架构：大小为 M的快速存储器，以及容量无限的慢速存储器。我们可以对存储在快速存储器中的数据执行任意计算，且不计代价，包括对这些值应用激活函数。具体而言，如果一个连接、其输入神经元的值以及其输出神经元的部分和（我们将其初始值视为偏置）都存储在快速存储器中，那么我们就可以将输入值与连接权重的乘积加到该输出神经元的部分和上（并假设此操作不需要额外内存）。如果要执行的计算需要某个当前仅存储在慢速存储器中的值，就必须先将该值从慢速存储器移至快速存储器。此移动计为 1 次**读入/输出**。此外，如果存储器已满，我们必须先腾出空间，才能读取新值。为此，可以删除值。但是，如果今后的计算步骤还需要某个值，或者该值是输出神经元的值，那么在删除它之前，必须先将其写入慢速存储器。删除不计代价，而每次写入操作计为 1 次**写入/输出**。初始时，所有 $N+W$ 参数都存储在慢速存储器中。一次计算的输入/输出操作次数等于读入/输出操作次数与写入/输出操作次数之和。目标是执行推理计算，即计算所有输出神经元的值并将其存储（也就是写回慢速存储器），同时尽可能减少输入/输出操作。我们将目标最优值形式化定义如下：

定义 1。 对于给定的 *FFNN* $\mathcal{N}$ 和快速存储器大小M，我们令 输入/输出操作$(\mathcal{N}, M) \in \mathbb{N}$表示在内存大小为 M 的情况下对$\mathcal{N}$ 执行推理所需的最少总体输入/输出操作次数，其中最小值取遍所有可能的计算策略（即能够解决该问题的计算、读取和写入步骤序列）。令 读取输入/输出操作$(\mathcal{N}, M) \in \mathbb{N}$ 表示所需的最少读取输入/输出操作次数，并令 写入/输出$(\mathcal{N}, M) \in \mathbb{N}$表示所需的最少写入/输出操作次数。

A. 在该模型中，我们如何描述推理计算？

我们引入驱逐策略这一概念，并指出，计算对应于驱逐策略与连接的拓扑序的组合。**驱逐策略** 是一组规则或指令，用于指定如何从快速存储器中逐出数据。也就是说，当快速存储器已满，而我们需要逐出一个值以腾出空间时，驱逐策略决定逐出哪个值。**最近最少使用** （LRU）是一种驱逐策略，其规则是在快速存储器中的所有值里，总是逐出最近最少使用的值。**轮转** （RR）是一种驱逐策略，它通过一个指针指定下一个要逐出的值；每次逐出一个值后，指针便向右移动一位到达末尾时，指针会移回内存的起始位置。**MIN** 驱逐策略（也称为 Belady 最优置换算法）规定，总是逐出未来最远才会被引用的值（如果某些值不会再被使用，则可逐出其中任意一个）。已有研究表明，对于给定的计算步骤序列 [32]，**MIN**所需的输入/输出操作次数最少。值得注意的是，尽管对于一般计算而言，实现**MIN** 驱逐策略较为困难，但对于前馈神经网络推理，只要确定处理权重的拓扑序，便可轻松离线实现该策略（始终逐出与该拓扑序中最靠后的权重相邻的激活值）。

我们假设提供了一种高效的驱逐策略。也就是说，当我们驱逐一个值，而该值属于以下两种情况之一时：（1）已经存储在慢速存储器中（例如，当我们读取某个已计算神经元的值，并将其用于其出边，但不对其进行更改时）；或（2）将来不再需要使用的值（某个已计算的非输出神经元已经被用于计算所有依赖于它的神经元），我们只需将其从快速存储器中删除，而无需执行写入/输出操作。

现在，令 $e_1, e_2, \ldots, e_W$ 为神经网络连接的一个拓扑序（也就是说，对于任意满足 e_i 和 e_j 为连接，且 e_i 的输出神经元是 e_j 的输入神经元的情况，都有 $i<j$）。结合驱逐策略，连接的拓扑序以自然的方式导出一种推理计算，如算法1所示。

算法1 推理算法

输入：连接的一个拓扑序 $e_1, \ldots, e_W$

输出：输出神经元的值
对于i = 1 **到** W，**执行**
  读取连接 $e_i = (a, b, w)$;
  如果输入神经元的值 n_a 不在快速存储器中 **then**
    读取 n_a （如果快速存储器已满，则可能先驱逐一个值内存已满）；
  结束 if
  如果输出神经元 n_b 的部分和不在快速存储器中**则**
    读取 n_b （如果快速存储器已满，则可能先逐出一个值）；
  结束 if
  更新 $n_b = n_b + w \cdot n_a$
  如果 在 e_i 之后没有与输出神经元 n_b 的连接 **then**
    应用激活函数： $n_b = f(n_b)$;
  结束 if
结束 for

请注意，这段伪代码概括了执行推理的所有“标准方式”。例如，基于矩阵-向量乘法的推理，对应的顺序是先处理来自第一层的所有连接，再处理第二层的连接，依此类推。然而，它还

允许我们对任意有向无环图所表示的前馈神经网络架构执行推理（包括那些具有非常“混乱”的跳跃连接的架构），而不仅限于分层架构。

至于优化输入/输出操作，它为我们提供了更大的灵活性，因为我们可以采用连接的任意拓扑序，包括不对应于逐层计算的拓扑序。例如，即使尚未计算完 i层的所有神经元，也可以开始计算 $i+1$、 $i+2$、.....层的神经元。这可以减少输入/输出操作，因为当我们完成对i层某个神经元的计算后，可以直接复用它来开始计算$(i+1)$层的神经元，而不必先存储起来，之后再重新读取。由于对每个连接进行计算时，都需要其输入神经元的值以及其输出神经元的部分和，因此我们希望找到一种顺序，使具有共同神经元的连接聚集在一起。

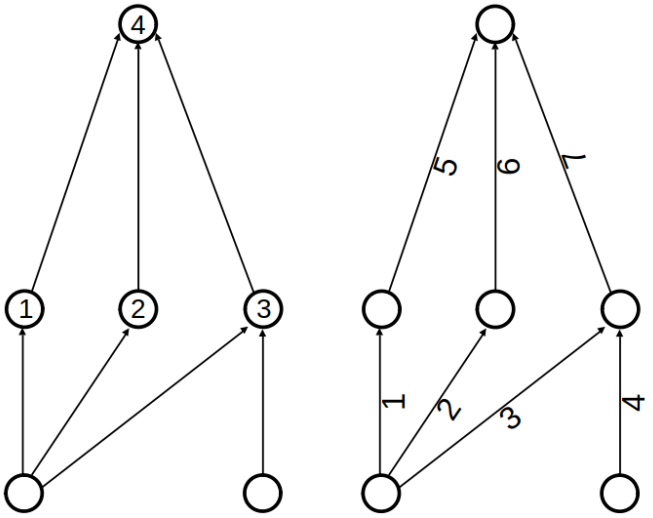


图 1：神经元与连接对应的拓扑序。

III. 推理的界

本节给出推理的输入/输出操作复杂度的一般界。这些界可作为指导，帮助我们进行后文讨论的、更针对具体实例的优化。

定理1。 设 $\mathcal{N}$ 是一个连通的前馈神经网络，并假设 $M \geq 3$。则推理所需的最优输入/输出次数满足

$$W + N + S \leq \textbf{\textit{I/Os}}(\mathcal{N}, M) \leq 2 \cdot (W + N - I). \quad (1)$$

该问题所需的最优读取 ***I/O*** 次数满足

$$W + N \leq \textbf{\textit{rI/Os}}(\mathcal{N}, M) \leq 2 \cdot W + N - I. \quad (2)$$

该问题所需的最优写入 ***I/O*** 次数满足

$$S \leq \textbf{\textit{wI/Os}}(\mathcal{N}, M) \leq N - I. \quad (3)$$

证明。 权重、偏置和输入值的总大小为 $N+W$。由于至少要读取一次所有这些数据才能进行推理，因此可得读取 I/O 次数的下界。同样，推理所需的写入 I/O 次数不能少于 S，因为（根据推理问题的定义）我们需要写入所有 S 输出神经元的值。因此，将读取和写入 I/O 次数的下界相加，即可得出总 I/O 次数的下界。

现在只需证明，我们可以使用不超过 $N + 2 \cdot W - I$ 次读取 I/O、 $N - I$ 次写入 I/O，以及总计 $2 \cdot ($ $N + W - I$）次 I/O 来完成推理。为此，我们固定非输入神经元的一个拓扑序： n_1、.....、 n_{N-I}。接着，我们重新排列连接，使其输出神经元按照该拓扑序排列（注意，这也是连接的一个拓扑序；图 1 展示了这种拓扑序之间的对应关系）。注意，该顺序自然划分为若干区间：首先是一个以n_1结尾的连接区间，接着是以 n_2 结尾的连接，依此类推。

现在我们证明，使用我们的推理算法（主论文中的算法1）并采用 MIN 驱逐策略，按此顺序执行推理的开销至多为 $2W + N - I$ 次读取 I/O 和 $N - I$ 次写入 I/O。在开始读取连接时，我们花费 1 次读取 I/O 来读取 n_1 的偏置，然后对每条以 n_1 结尾的连接，至多花费 2 次读取 I/O（1 次用于读取输入神经元，1 次用于读取连接本身）。处理完以 n_1 结尾的这段连接区间后，我们应用激活函数，完成神经元 n_1 的计算。然后继续处理以 n_2 结尾的连接区间。在此连接区间以及所有后续连接区间中，我们也会花费 1 次读取 I/O 来读取输出神经元的偏置，并对每条连接至多花费 2 次。由于每个 $N - I$ 非输入神经元恰好对应一个连接区间，因此读取偏置至多花费 $N - I$ 次读取 I/O，读取连接及其输入神经元则额外至多花费 $2W$ 次，合计为 $2W + N - I$ 次读取 I/O。

现在我们来统计这项计算中执行的写入 I/O。由于所有以同一神经元为终点的连接彼此相邻，我们总能在无需写入临时值的情况下计算神经元（开始计算某个神经元后，接下来的所有计算步骤也都针对该神经元；只有在前一个神经元计算完成后，我们才开始计算另一个神经元）。由此可知，如果我们对某个神经元执行一次写入 I/O，那么写入慢速存储器的就是一个已完全计算出的神经元值。因此，即使我们再次读取并驱逐该值，也不会再为它执行一次写入 I/O（由于该值已完整存储在慢速存储器中，高效的驱逐策略会通过删除该值将其驱逐）。因此，每个非输入神经元最多对应一次写入 I/O。由于有 $N - I$ 个非输入神经元，总计最多执行 $N - I$ 次写入 I/O。将读取和写入 I/O 相加后，我们得出 I/O 总数至多为 $N+2 \cdot W-I+N-I=2 \cdot (N+W-I)$。

□

注意，在不等式 1 和 2 中，右侧的项最多是左侧项的两倍。接下来的命题证明，无法通过乘以一个非 1 的常数来进一步收紧该定理给出的任何通用界限

通过乘以不等于 1 的常数进一步缩放它。

命题 1. 作为仅取决于 W、N、I 和 S 的通用界，定理 *1* 中的界是紧的：对于每个界以及任意 $\epsilon > 0$，都存在一些实例，使真实值与该界相差一个乘法因子，该因子属于 $[1-\epsilon, 1+\epsilon]$。

为证明这一命题，我们需要证明，对于定理1中的每个界，都存在与该界任意接近的实例（这里的接近是指乘法因子意义上的接近）。

第一个引理证明，存在一些实例能够精确达到命题 1 的所有下界（因此显然也能任意接近这些下界）。

引理 1. 对于任意前馈神经网络 $\mathcal{N} = [L_1, L_2, \ldots, L_d]$，若任意两个相邻层 L_i、L_{i+1} 合计至多有 $M-1$个神经元，则可以进行推理，且读取、写入和总输入/输出次数均达到定理*1*给出的下界。

证明。 我们通过描述一种能够达到这一目标的计算策略来证明。

将 $|L_1|$ 输入值读入快速存储器。同时，在快速存储器中用 L_2中神经元的偏置初始化 $|L_2|$ 部分和。由于 $|L_1| + |L_2| \leq M - 1$，快速存储器中至少还有足够的空闲空间容纳一个值。我们利用这块空闲空间遍历 L_1 与 L_2 之间的权重，并对每个权重执行以下步骤：

1) 读取该权重。2) 将其与对应输入神经元的值相乘（该值已在快速存储器中）。3) 将乘积加到对应输出神经元的部分和中。4) 将该权重从快速存储器中删除。

遍历完成后，我们对 L_2的部分和应用激活函数，从而完成该层神经元的计算。现在我们不再需要 L_1中的值，可以将其从快速存储器中删除。这样就有足够的空闲空间来读取 L_3的偏置，并以相同的方式计算这一层。我们依此逐层进行计算。完成最后一层神经元值的计算后，我们将这些 S 值写回慢速存储器。

请注意，我们对每个 $N-I$ 偏置、I 输入值和 W 权重都恰好读取一次。因此，我们使用了 $N+W$ 读入/输出。我们只在写入 S 输出值时使用写入/输出。因此，总共使用了 $N+W+S$ 输入/输出操作。 □

另外，我们也可以通过验证具有所述性质的前馈神经网络能够用我们稍后介绍的紧凑增长方法构造出来，来证明前一个引理。下一个引理表明，存在一些实例，其读取 I/O 次数和总 I/O 次数可以任意接近各自的上界。

引理 2。 对于每个 $\epsilon > 0$，都存在具有 $I/Os(\mathcal{N}, M) > (1-\epsilon) \cdot 2 \cdot (W + N - I)$ 以及$rI/Os(\mathcal{N}, M) > (1-\epsilon) \cdot (2 \cdot W + N - I)$ 的前馈神经网络 $\mathcal{N}$（大小任意）。

证明。对于输入神经元占比很大的架构，情况正是如此。例如，在一棵有 I 个输入神经元的树中，所有输入神经元都连接到同一个输出神经元，我们有 $I/Os(\mathcal{N}, M) = \cdot 2 \cdot (W + N - I)$ 和 $rI/Os(\mathcal{N}, M) = 2 \cdot W + N - I$。 □

下一个引理表明，存在一些实例，其所需的写入 I/O 数可以任意接近上界。

引理 3. 对于每个 $\epsilon > 0$，都存在规模任意大的前馈神经网络 $\mathcal{N}$，满足 $wI/Os(\mathcal{N}, M) > (1-\epsilon) \cdot (N - I)$。

证明。 注意，所有神经元均为输入神经元或输出神经元的任何前馈神经网络都满足这个不等式。不过，为了给出更不平凡的例子，我们将构造满足该不等式且具有隐藏神经元的前馈神经网络。

对于 M，h，$w \in \mathbb{N}^+$，考虑一个具有 I 个输入神经元、S 个输出神经元以及一个包含 h 个神经元的隐藏层的前馈神经网络。显然，该前馈神经网络至少需要 S 次写入 I/O。因此，对于任何满足 $S > h \cdot (1-\epsilon)/\epsilon$ 的参数配置 I，h，S，都有$wI/Os(\mathcal{N}, M) > (1-\epsilon) \cdot (N - I)$。 □

前面三个引理给出了紧确的极端情形，使我们得以证明命题 1。尽管这些引理证明中构造的架构是“人为的”，但它们足以表明，仅取决于 $I, S, N,$和 W 的通用界，不可能比我们的界更紧（紧到相差一个不为 1 的常数乘法因子）。不过请注意，尽管这些极端情形是“人为的”，但它们远不只是“少数几个边缘情况”。事实上，每个证明中的构造都可以任意大。

命题 *1* 的证明。 引理 1 表明，读入、写入和总输入/输出次数的下界是紧确的。引理 2 表明，读取 I/O 次数和总输入/输出次数的上界是紧确的。引理 3 表明，写入 I/O 次数的上界是紧确的。 □

请注意，所有界限都不依赖于 M。这令人惊讶，因为对于大多数计算问题，I/O 复杂度都强烈依赖于内存大小[1]。随着 M 增大，我们可以获得更大的灵活性，从而避免执行输入/输出操作。另一个令人惊讶的事实是，这些界限不依赖于网络的具体属性，除其规模之外。这并不意味着总体 *I/O* 复杂度不依赖于这些量（内存大小和前馈神经网络架构），但从界限的紧致性可以得出结论，这种依赖程度有限。

对于写入 I/O，情况截然不同。定理1给出的写入 I/O 界不够紧。事实上，这些界可能任意宽松（尽管按照命题 1 的意义，它们是最优的）。写入 I/O 的问题在于，根本不可能给出只依赖于W、N、I和 S，而不依赖连接顺序、内存大小 M以及前馈神经网络架构等其他因素的有意义的界（命题 1 蕴含了这种不可能性）。下面的命题表明，按逐层 方式进行推理，其开销可能任意大于采用最优顺序进行推理的开销。

[1]例如，对于矩阵乘法，其对 M 的依赖程度为 $O(1/\sqrt{M})$ [7]。

命题 2。 对于每个 $c \in \mathbb{N}$ 和每种内存大小 $M \in \mathbb{N}$，都存在一个前馈神经网络 $\mathcal{N}_{M,\ c}$，使得按逐层 方式进行推理所需的写入 *I/O* 至少是最优方式的 c 倍。

证明。 考虑一个具有 $c+2$ 层的稀疏前馈神经网络：输入层有 1 个神经元，每个 c 个隐藏层中各有 $2M$ 个神经元，输出层有 1 个神经元。设每个隐藏神经元恰好有一个输入连接和一个输出连接，输入神经元与第一隐藏层中的每个神经元相连，输出神经元则与最后一个隐藏层中的所有神经元相连（换言之，该前馈神经网络包含 $2M$ 条长度为 $c+2$ 的神经元链，它们汇聚于输入神经元和输出神经元）。如果我们按层依次对该前馈神经网络进行推理，就需要所有 $2M$ 个隐藏层神经元的值来计算下一层。由于快速存储器只能容纳 M 个神经元，我们需要存储其余 M 个神经元，因此每个隐藏层至少需要 M 次写入操作。因此，如果我们按层依次对该前馈神经网络进行推理，就至少需要 $M \times c$ 次写入 I/O。然而，如果我们逐条计算神经元链，只需一次写入 I/O 即可。

□

IV. 连接重排序：调整顺序以适应前馈神经网络和硬件

在本节中，我们介绍连接重排序，这是一种针对给定前馈神经网络架构和内存大小 M 优化连接拓扑序的方法。该方法依赖以下超参数：**窗口大小** $ws \in \mathbb{N}$、**冷却率** $\sigma \in \mathbb{R}$以及**迭代次数** $T \in \mathbb{N}$。该方法的总体思路基于模拟退火 [33]: 在 T 次迭代中，我们随机改变拓扑序（称为生成邻居），并根据新旧顺序的质量，以一定概率保留或舍弃这些改变（称为更新）。接下来，我们将通过描述生成邻居和更新的过程，对该方法进行完整说明。

A. 生成邻居

我们从一个连接的拓扑序开始e_1，e_2，……，e_W并令**旧输入/输出操作数** $\in \mathbb{N}$ 表示该拓扑序所需的输入/输出操作次数。输入/输出操作次数显然取决于内存大小和驱逐策略，但在算法执行期间二者保持不变，因此输入/输出操作次数仅取决于连接的拓扑序。

首先，我们均匀随机地选择一条连接 e_i。令 w 为我们从$\{0, 1, \ldots, ws-1\}$中均匀随机选取的一个整数。我们考虑连接窗口e_i，e_{i+1}，……，$e_{\min(i+w,W)}$。我们以 0.5 的概率选择移动此窗口中连接的方向：向左或向右。

a) 情况 *1*：向左移动：如果我们将连接向左移动，就从窗口中最靠左的连接 e_i 开始。我们将 e_i 向左移动，直到遇到另一个连接 e_s，该连接的输入神经元与 e_i 相同，或者其输出神经元等于 e_i 的输入神经元（如果始终没有遇到这样的连接，我们就将 e_i 移到顺序的最前面）。我们将 e_i 插入到 e_s 紧邻的位置，使 $\ldots, e_s, e_i, e_{s+1}, \ldots$ 成为新的顺序。请注意，这仍然是一个拓扑顺序。然后，我们以相同的方式继续移动窗口中第二靠左的连接。如此继续，直到移动完这个窗口中的所有连接。

b) 情况 *2*：向右移动：如果我们将连接向右移动，就从窗口中最右边的连接开始移动。将它移动，直到遇到另一条连接e_z；该连接的输出神经元与 e_i 相同，或者其输入神经元等于 e_i的输出神经元。我们将 e_i 插入到 e_z 之前，使得……，e_{z-1}，e_i，e_z，……成为新的顺序。然后，我们继续移动窗口中第二靠右的连接。依此操作，直到移动完此窗口中的所有连接。

B. 更新

将窗口中的所有连接移出后，我们便得到一个新的拓扑序。记为 $\tilde{e_1}, \tilde{e_2}, \ldots, \tilde{e_W}$。我们统计按此新顺序进行推理所需的输入/输出操作数，并将其记为 **newI/Os**。如果 **newI/Os< oldI/Os**，我们必定进行更新：$e_1 = \tilde{e_1}, e_2 = \tilde{e_2}, \ldots, e_W = e_{\tilde{W}}$并**oldI/Os=newI/Os**。

如果 **newI/Os≥oldI/Os**，我们可以选择更新，也可以恢复旧顺序。我们随机决定，选择更新的概率为 $2^{-(\text{newI/Os}-\text{oldI/Os})\cdot t^{\sigma}}$, 其中 t 表示迭代次数。

V. 紧凑增长：使前馈神经网络适配硬件

本节提出一种构造方案，能够完整刻画在给定内存大小 M 下允许以最少输入/输出操作进行推理的前馈神经网络架构。这回答了以下问题：对于给定的内存大小 M，哪些前馈神经网络架构能够以与定理1给出的下界相符的输入/输出操作数完成推理？或者等价地说：哪些前馈神经网络能够在不受内存不足限制、无需写入或读取中间值的情况下完成推理？核心思路是将前馈神经网络的构造与推理计算中的各个步骤紧密结合。更具体地说，我们将通过一系列步骤构建前馈神经网络，每个步骤分为四种不同类型。

我们引入卵石和袋子的概念，以便对计算中的输入/输出操作进行推理。每颗卵石对应一个神经元。卵石可以是灰色的（如果尚未完全计算），也可以是黑色的（如果已完全计算，且其出边可以使用它）。袋子代表快速存储器。

A. 构造规则

我们从一个“空前馈神经网络”（其中没有神经元和连接）以及一个表示快速存储器内容的空袋子开始构造。我们通过一系列以下四种类型的构造步骤来逐步构建前馈神经网络（对于每种类型，我们会在括号中写出它可转换成的对应计算步骤）。

1) 只要我们的袋中卵石数量不超过 $M-2$，就可以添加一颗灰色或黑色卵石，并同时向前馈神经网络中添加一个孤立的神经元，以此将该神经元读入快速存储器。

袋中，并同时向前馈神经网络中添加一个孤立的神经元（将一个神经元读入快速存储器）。
2) 当袋中有一颗黑色卵石和一颗灰色卵石时，我们可以在前馈神经网络中画一条连接，从对应于黑色卵石的神经元出发，连接到对应于灰色卵石的神经元。（将输入神经元的值乘以权重，并加到输出神经元的部分和中。）

3) 我们可以将一颗灰色卵石变成黑色卵石（应用激活函数，完成该神经元的计算）。**4)** 我们可以从袋中移除一颗黑色卵石（从快速存储器中删除）。

以下定理概述了通过上述步骤依次构建的前馈神经网络的性质。

定理 2 (紧凑增长**)**。 设 $\mathcal{N}$ 是一个具有 W 个权重和 N 个神经元的连通前馈神经网络。给定大小为$M \geq 3$的内存，当且仅当 $\mathcal{N}$ 可以通过我们刚才描述的紧凑增长方案构建时，我们才能对 $\mathcal{N}$ 进行推理，且需要 $N + W$ 次读取 *I/O* 和 S 次写入 *I/O*。

定理 *2* 的证明。 如果我们可以通过一系列这样的构造步骤构建 $\mathcal{N}$，那么括号中对应的步骤就描述了一个有效的推理计算步骤序列。通过只允许在袋中的卵石数量严格少于 $M-1$ 时插入卵石，我们可以确保内存中的卵石数量始终不超过 $M-1$，因此连接始终有一个空闲位置。这样，这一计算步骤序列就可以在大小为 M 的快速存储器中执行，而无需对临时值进行任何输入/输出操作。这就证明了该陈述的“若”部分。

反过来，假设对于给定的 $\mathcal{N}$ 和 $M \geq 3$，我们可以对连接进行排序，使得推理所需的输入/输出操作数最少。将这一顺序记为 e_1，e_2，.....，e_W。我们将说明如何把计算步骤转换为一系列卵石移动，从而构造出前馈神经网络。根据定理1，输入/输出操作数最少意味着我们执行 $W + N$ 次读取 I/O 和 S 次写入 I/O。由于这是读取所有必要参数一次、并写出输出神经元的值一次所需的读取 I/O 数量，这些 I/O 仅足以完成这些必要的读取和写入。尤其是，我们不能再花费额外的 I/O 来写入或读取临时值。因此，对于每个神经元，我们会在某个时刻读取一个值（偏置或输入值），然后在一段时间内将该值保存在快速存储器中，并根据计算的进展对其进行更新；之后，在神经元的值计算完成且不再需要时，将其移出。此后，我们将无法再次使用该值。我们将此过程转化为卵石构造：将一颗灰色卵石放入袋中并保留一段时间，在神经元完全计算完成后将其变为黑色卵石，然后在神经元的值不再需要并被移出时将其移除。

在此区间内，这种最优计算策略必须使用该神经元的所有输入连接和出边：区间的前半段使用输入连接（此时神经元尚未完全计算，因此卵石为灰色），后半段使用出边。此外，每当使用一条连接时，该连接的“另一端”也必须位于快速存储器中，因此相应的卵石必须在袋中（请记住，每当读取偏置/输入时，我们都会放入一个卵石；当逐出该值时，则移除这个卵石）。此外，每当使用一条连接时，输入神经元的卵石已经是黑色（因为神经元一旦完全计算完成，我们就会立即将其卵石变为黑色），而输出神经元的卵石仍为灰色。因此，当我们在计算侧使用这些连接时，也可以在卵石模型侧画出这些连接。

由于内存大小受限，我们不妨假设快速存储器中的神经元值始终不超过 $M-1$ 个（如果读取第 M 个值，就没有空间存放连接，计算便会停滞，直到我们移除一个神经元值；但我们完全可以在读取另一个值之前先移除这个神经元值）。因此，我们不妨假设，只有在快速存储器中的值不超过 $M-2$ 个时才读取值。在卵石构造中，这确保了只有当袋中的卵石不超过 $M-2$ 个时，我们才添加卵石。

□

我们可以将此定理应用于针对给定前馈神经网络的最优硬件问题。为此，我们考虑前馈神经网络的**带宽**，其定义为满足以下条件的最小 $k \in \mathbb{N}$：存在一种神经元的拓扑序，使得任意两个相连的神经元在该拓扑序中的间隔至多为 k 步。我们现在证明，当前馈神经网络的带宽为 k 时，可以使用 $M = k + 2$ 以紧凑增长的方式构建它们。

推论1. 设 $\mathcal{N}$ 的带宽为 $k \in \mathbb{N}$。如果内存大小为 $M \geq k + 2$，那么我们可以在$\mathcal{N}$ 上执行推理，而无需读取或写入任何临时值。

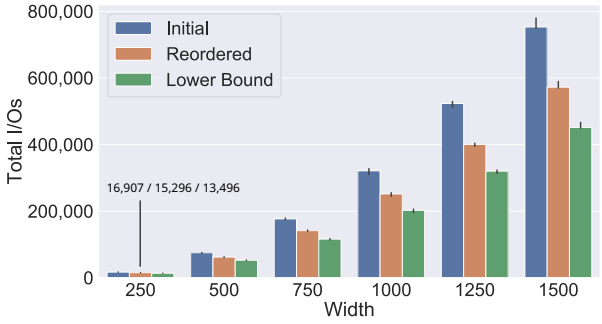
证明。设 $\mathcal{N}$ 的带宽为 k。这意味着神经元 n_1、n_2、.....、n_N存在一个拓扑序，使得对于任意 $i \in \{1, \ldots, N\}$，所有指向 n_i 的连接都来自此前的 k 个神经元 $\{n_{max(1,i-k)}, n_{max(1,i-k)+1}, \ldots, n_{i-1}\}$。因此，我们可以通过按照神经元的拓扑序添加卵石来构建具有紧凑增长的 $\mathcal{N}$；当袋装满时，移除 $k+1$ 步之前添加的卵石。这样，当我们将神经元 n 的卵石放入袋中时，所有 n 直接依赖的其他神经元的卵石也都在袋中，因此我们可以添加所有指向n的连接。因此，使用大小为 $k+2$的内存，我们可以构建 $\mathcal{N}$。

□

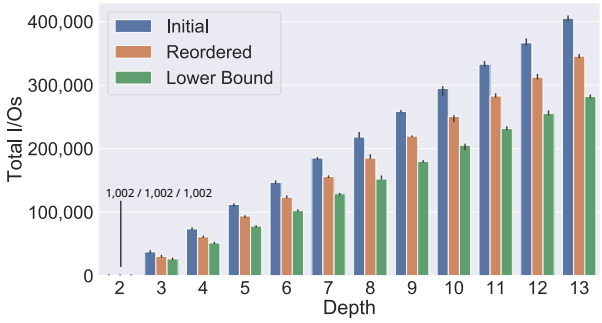
VI. 实验评估

为了展示我们结果在理论上的准确性及其实用价值，我们既进行了统计输入/输出操作次数的模拟实验，也测量了重新排序后的稀疏神经网络相较于高性能库的 CPU 性能。

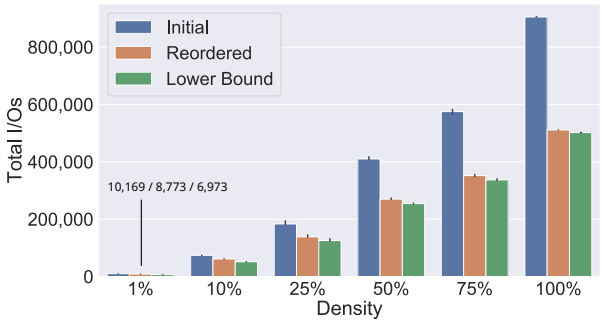
作为输入，我们使用两种类型的网络：深度、广度和密度各异的多层感知机；以及来自



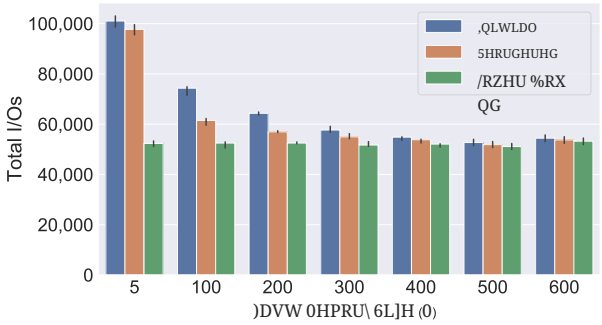
(a) 每层神经元数



(b) 深度



(c) 连接密度



(d) 快速存储器大小

图 2：针对不同稀疏神经网络属性的连接重排序。除非另有说明，网络为一个宽度为 500 个神经元、包含 4 层的多层感知机，边密度为 10%，且为 $M = 100$。下界根据定理1得出。当规模过小时，会标注多层感知机的输入/输出操作。

广受欢迎的 BERT [34] Transformer [23] 神经网络中的全连接层。对于前者，我们生成五个随机稀疏的前馈神经网络（通过随机边采样或使用紧凑增长得到）。对于后者中，我们使用预训练的 $\text{BERT}_{\text{LARGE}}$ 神经网络，并进行幅值剪枝 [21]。

A. 模拟实验

为了测试连接重排序（CR）和紧凑增长（CG），我们实现了算法1和缓存模拟，以及 LRU、RR 和 MIN 淘汰策略。每种实验配置都使用五个随机多层感知机和 BERT 运行，并报告中位数以及以误差条表示的 95% 非参数置信区间。

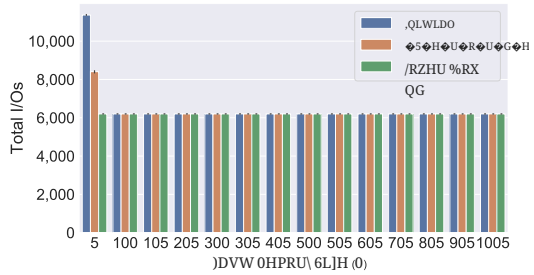
1) 连接重排序：我们通过优化随机稀疏前馈神经网络的输入/输出操作，评估连接重排序（CR）在不同结构特性和稀疏度特性下的表现。图2展示了我们改变基准前馈神经网络的四个维度：一个密度为 10%、包含 4 层且每层有 500 个神经元、另有一个输出神经元的多层感知机；推理时快速存储器大小为 100 个元素。我们分别改变密度、宽度、深度或快速存储器大小，同时将其他参数保持在其基准值。我们采用 MIN 淘汰策略。首先，我们按照定理1证明中的方式对连接排序，以保证输入/输出操作次数不超过最优次数的两倍。然后，我们应用 CR，设置为 $T = 1{,}000{,}000$，$\sigma = 0.2$，并将 ws 设为网络平均入度的四倍。从这些示例可以看出，CR 将输入/输出操作总数进一步减少了最多 43.5%，使结果最多比理论下界近 97.4%。此外，CR 在所有参数配置下都能持续带来显著提升（初始 I/O 复杂度已接近下界的情况除外）。

2) 紧凑增长：我们按照附录B中的描述，生成了三个采用紧凑增长的前馈神经网络。这些前馈神经网络分别对应于数值 $M_g = 100$，300 和 500。然后，我们统计了这些前馈神经网络在不同快速存储器大小 M 下所需的输入/输出操作次数。这些数字显示在图3中。正如定理 2 所预测的，我们可以看到，对于这三个前馈神经网络，当 $M \geq M_g$ 时，所需的输入/输出操作次数最少。在这些情况下，我们通过紧凑增长生成的连接顺序直接达到了这一最小次数，无需应用连接重排序。

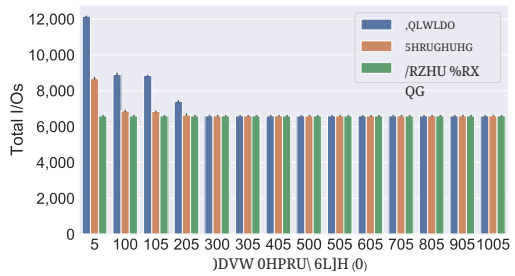
此外，在某些内存大小范围 M 且满足 $M < M_g$ 时，应用连接重排序也能使输入/输出操作次数达到最少。当然，这并不与定理 2 矛盾：该定理指出，当且仅当前馈神经网络能够使用 $M_g = M$ 构建时，我们才能在内存大小为 M 的情况下达到最少的输入/输出操作次数。但这并不意味着同一个前馈神经网络不能使用 $M_g < M$ 构建。

3) 缓存淘汰策略：图4展示了随机前馈神经网络在 RR、最近最少使用和 MIN 缓存淘汰策略下，输入/输出复杂度随模拟退火迭代次数的变化情况。由于采用了该更新方案，我们观察到收敛速率逐渐下降并趋近于最小值，其中大部分输入/输出缩减发生在前 10,000 次迭代中。值得注意的是，经过优化后，RR 和最近最少使用的输入/输出操作数趋于相似，这表明连接重排序能够针对采用特定缓存机制的硬件架构调整前馈神经网络。

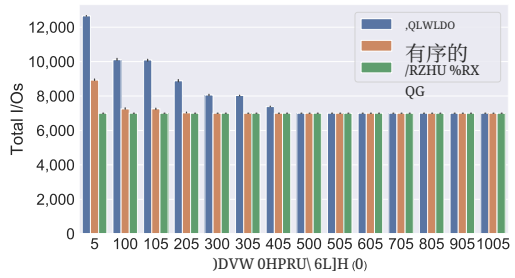
4) 快速存储器大小：在图5中，我们展示了增加快速存储器大小时，输入/输出操作的数量如何变化



(a) $M_g = 100$



(b) $M_g = 300$



(c) $M_g = 500$

图 3：研究针对不同快速存储器大小设计的紧凑增长生成的前馈神经网络 M_g（1,000 个入度为 4 的神经元和一个入度为 M_g的输出神经元）。

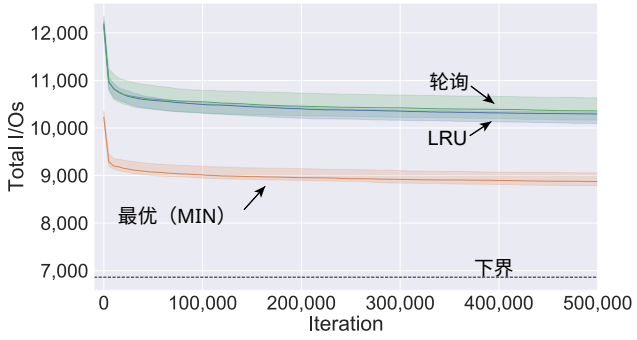


图 4：不同缓存淘汰策略下的输入/输出操作。

，比较连接重排序（CR）前后的情况。可以看到，当快速存储器足够大时，无论是否进行连接重排序，我们使用的输入/输出操作数量都达到定理1给出的理论下界。但当内存大小不足时，可以看到，采用连接重排序能够减少输入/输出操作，并更快地趋近下界。

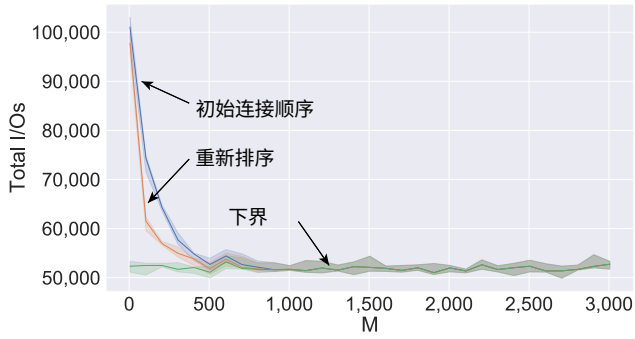


图 5：研究不同快速存储器大小（M）对随机稀疏前馈神经网络的影响（3 层，每层 500 个神经元，后接一个输出神经元，密度为 1%）。

5）在 *Transformer* 模型 *BERT* 中的应用：Transformer [23] 是一种由所谓的编码器和解码器构成的深度神经网络架构。Transformer 的一个典型组成部分是注意力头，这使其适用于自然语言处理及相关任务。Transformer 的一个著名实例是 BERT [34]，，它于2019年推出。BERT 的突破性成果使 Transformer 可以说成为近年来最受欢迎、研究最广泛的深度神经网络架构。部署 Transformer 时面临的主要挑战是模型规模庞大，因此对计算资源的需求也很高 [24]。BERT 包含多个深度为 2、权重矩阵维度为1024×4096 和 4096×1024的前馈神经网络。BERT 的大部分参数以及推理所需的计算时间都来自这些前馈神经网络 [24]。因此，它们是剪枝的理想目标。我们选取其中一个前馈神经网络，通过移除权重绝对值最小的连接对其进行剪枝。图 6 展示了该前馈神经网络推理时的输入/输出计数和输入/输出下界。我们考察了不同的稀疏度和不同的缓存淘汰策略。此外，我们假设快速存储器的容量为 100。

B. 性能实验

为验证输入/输出缩减带来的性能优势，我们在英特尔 CPU 上执行批量推理并测量性能。在所有实验中，我们使用模拟的最优（MIN）驱逐策略对前馈神经网络进行重排序，然后分别运行重排序前后的前馈神经网络，并使用 MKL 执行传统的基于层的方法进行稀疏-稠密矩阵乘法（压缩稀疏行矩阵乘法）。批量推理（生产环境中采用的方式）能够使用 SIMD 指令，从而更充分地利用内存带宽。每项实验均运行 10 次，批量大小为 128，并报告中位数以及最短和最长执行时间的误差条。注释显示了我们的方法（不进行和进行重排序）相对于基于层的推理方法所获得的加速比。

实验使用的硬件是一台 32 核英特尔至强金牌 6130 处理器（运行频率为 2.10 GHz），配备 1.5 TB

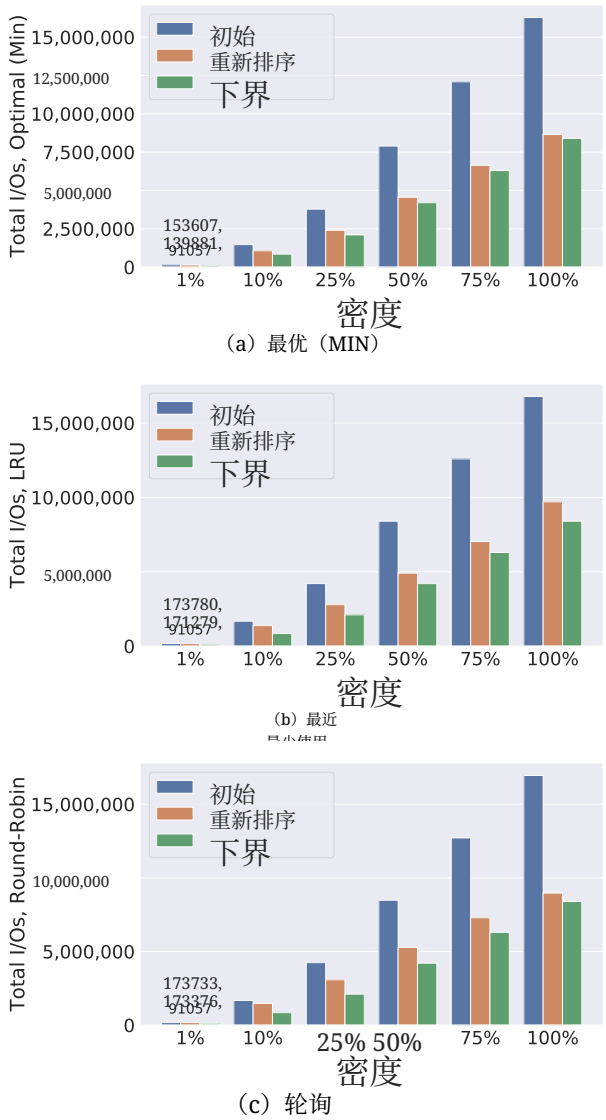


图6：对来自 $\text{BERT}_{\text{LARGE}}$ 的 MLP 进行连接重排序，采用不同的淘汰策略、不同的边密度和$M = 100$。下界根据定理1得出。

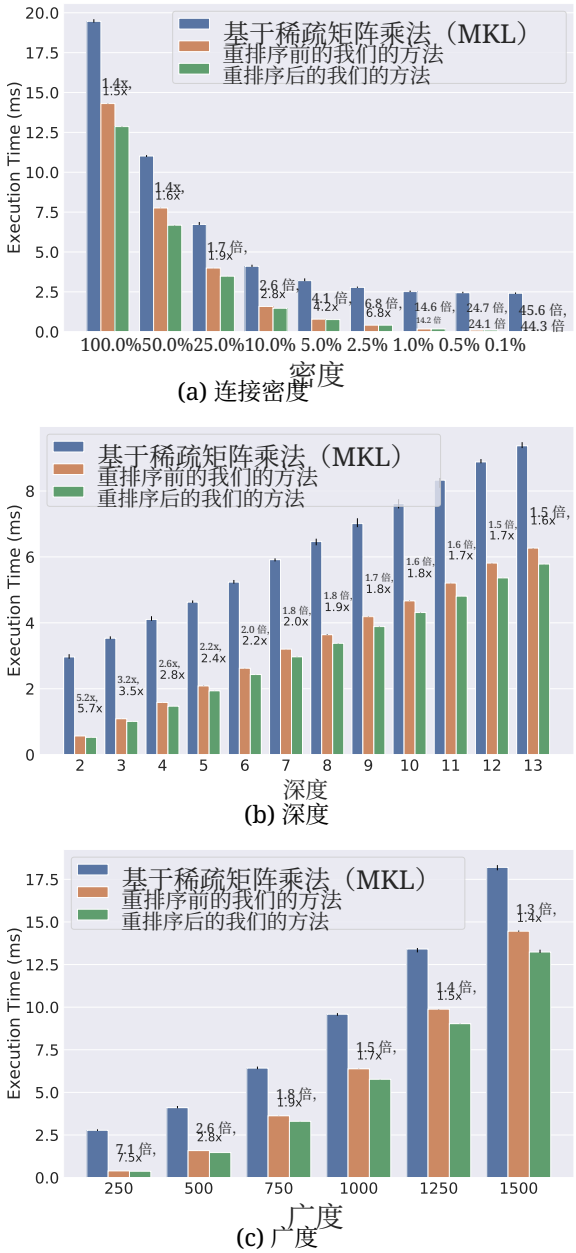


图7：采用不同方法的随机稀疏前馈神经网络的执行时间。

随机存取存储器。软件方面，CSRMM 使用 Intel oneAPI MKL 2021.3.0，重新排序后的网络使用 GCC 10.2.0。

1) 随机稀疏前馈神经网络：我们以模拟实验中使用的同一基准前馈神经网络为起点（密度为 10%、包含 4 层且每层有 500 个神经元、另有 1 个输出神经元的多层感知机），在将其他参数保持为基准值的同时，分别改变密度、宽度或深度，以测量我们的方法的执行时间。

图 7a 展示了不同密度下的执行时间。可以看出，前馈神经网络越稀疏，与标准的逐层推理方法相比，我们的方法所获得的加速比就越大。例如，在密度为 0.1% 时，我们的加速比约为 45 倍。另一方面，前馈神经网络越稀疏，通过重排序获得的额外加速比就越小。事实上，有些出乎意料的是，当密度低于 1% 时，不进行重排序的执行时间略短于进行重排序的执行时间，我们认为这有两个原因：(a) 输入/输出操作的减少幅度相对较小（见第 VI-A1 节）；以及 (b) 网络规模较小，由于前馈神经网络能够装入 L2 缓存，性能收益微乎其微。

图 7a 还展示了与 MKL 相比，在 100% 稠密前馈神经网络上的加速比。出现这一现象的原因是，为了保证测量的一致性，我们在这种情况下使用了压缩稀疏行矩阵乘法，而不是稠密矩阵与稠密矩阵相乘（通用矩阵乘法），这也解释了观察到的性能下降。

在图 7b–7c 中，我们展示了执行时间随前馈神经网络的深度（图 7b）和广度（图 7c）的变化（同时将其他结构参数保持在其基准值不变）。在这两种情况下，我们观察到加速比随深度和广度线性增长（因此也随实例规模增长），

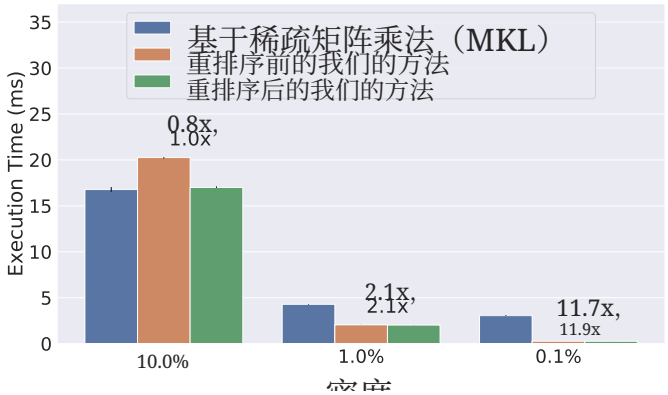


图 8：采用连接重排序时，$\text{BERT}_{\text{LARGE}}$ 在不同密度下的执行时间。

不过，当网络只在一个维度上增大、其他维度保持不变时，收益会逐渐递减。

总之，我们的多层感知机性能实验与模拟结果一致，表明重排序在实际应用中有益于稀疏神经网络，尤其是在密度较低的情况下。

2）*BERT*：图 8 展示了此前介绍的 $\text{BERT}_{\text{LARGE}}$ 编码器多层感知机在不同密度下的推理执行时间。对于密度为 10% 时的 MKL 推理，我们发现一个明显的异常值（根据 Tukey 方法 [35] 的定义；一次运行耗时 106 毫秒，而其余 9 次运行耗时约 17 毫秒），因此在本图所示的数据中将其剔除。图中显示，重排序后的运行时间始终低于重排序前，并证实即便对于仅有两层的实际网络模块，采用我们的方法也有益处，尤其是在密度较低时。

VII. 讨论

A. 预取

现代系统支持预取以加速输入/输出操作，而本文提出的模型未考虑这一点。不过请注意，预取的效果与本文考虑的优化“正交”，因为预取并不会减少输入/输出操作的总量，只会影响这些操作所需的时间。

B. 使用给定拓扑序的成本

“良好”（即输入/输出高效）拓扑序的计算是在“离线”状态下完成的；一旦确定了顺序并进行推理，就无需再显式查看该顺序，因为它已编码在连接的布局方式中。因此，在推理过程中，按照任何给定的拓扑序处理连接都不会产生额外成本。

C. *“I/O”*一词

论文中的理论不仅适用于缓存，也适用于任何类型的本地内存（包括寄存器等）。I/O 指的是对“慢速”内存进行加载/存储，这种内存可以是片外存储器、与另一个节点通信的内存，甚至是暂存存储器。这项工作既不假设也不依赖缓存，还涵盖了如何将内存高效地流式传输到芯片等方面。

VIII. 结论

我们为前馈神经网络推理的 I/O 复杂度建立了紧确界。我们提出了一种 2-最优计算策略，并展示了对连接顺序进行组合优化如何进一步减少输入/输出操作。给定固定的内存大小，紧凑增长使我们能够构建可实现 I/O 最优推理的前馈神经网络：无需读取或写入任何临时值。我们的工作为将现有神经网络映射到硬件这一问题提供了理论见解，从而使部署期间的能耗和延迟降至最低。最重要的是，这些见解表明，只有摒弃以简单的逐层方式处理深度神经网络，才能实现这些改进。为此，我们提出的连接重排序打破了这一误解，显著降低了 I/O 开销。在真实硬件上的实验中，相较于标准的（基于 CSRMM 的）推理方式，我们观察到最高达 45× 的加速比。此外，在我们的 2-最优计算策略与经过连接重排序进一步改进的策略之间，我们观察到最高达 1.17× 的加速比。

致谢

本研究得到欧盟地平线项目 GLACIATION（资助协议编号 101070141）的支持。T.B.N. 得到瑞士国家科学基金会（Ambizione 项目 #185778）的支持。我们还感谢瑞士国家超级计算中心（CSCS）为本研究提供支持并开放计算资源。

参考文献

[1] Zhe Jia、Marco Maggioni、Benjamin Staiger 和 Daniele P. Scarpazza。通过微基准测试剖析 NVIDIA Volta GPU 架构，2018年。[2] D. Unat、A. Dubey、T. Hoefler、J. Shalf、M. Abraham、M. Bianco、B. L. Chamberlain、R. Cledat、H. C. Edwards、H. Finkel、K. Fuerlinger、F. Hannig、E. Jeannot、A. Kamil、J. Keasler、P. H. J. Kelly、V. Leung、H. Ltaief、N. Maruyama、C. J. Newburn 和 M. Pericás。高性能计算系统中数据局部性抽象的发展趋势。*IEEE* 并行与分布式系统汇刊，28(10):3007–3020，2017年。[3] V. Sze、Y. Chen、T. Yang 和 J. S. Emer。深度神经网络的高效处理：教程与综述。*IEEE* 会刊，105(12):2295–2329，2017年。[4] Tal Ben-Nun 和 Torsten Hoefler。揭示并行与分布式深度学习：深入的并发分析。*ACM Comput. Surv.*，52(4)，2019年8月。[5] Song Han、Xingyu Liu、Huizi Mao、Jing Pu、Ardavan Pedram、Mark A. Horowitz 和 William J. Dally。EIE：基于压缩深度神经网络的高效推理引擎。载于 第*43*届国际计算机体系结构研讨会论文集，ISCA ’16，第243–254页。IEEE Press，2016年。

[6] Norman P. Jouppi、Cliff Young、Nishant Patil、David Patterson、Gaurav Agrawal、Raminder Bajwa、Sarah Bates、S uresh Bhatia、Nan Boden、Al Borchers、Rick Boyle、Pierre-luc Cantin、Clifford Chao、Chris Clark、Jeremy Coriell、Mike Daley、Matt Dau、Jeffrey Dean、Ben Gelb、Tara Vazir Ghaemmaghami、Rajendra Gottipati、William Gulland、Robert Hagmann、C. Richard Ho、Doug Hogberg、John Hu、Robert Hundt、Dan Hurt、Julian Ibarz、Aaron Jaffey、Alek Jaworski、Alexander Kaplan、Harshit Khaitan、Daniel Killebrew、Andy Koch、Naveen Kumar、Steve Lacy、James Laudon、James Law、Diemthu Le、Chris Leary、Zhuyuan Liu、Kyle Lucke、Alan Lundin、Gordon MacKean、Adriana Maggiore、Maire Mahony、Kieran Miller、Rahul Nagarajan、Ravi Narayanaswami、Ray Ni、Kathy Nix、Thomas Norrie、Mark Omernick、Narayana Penukonda、Andy Phelps、Jonathan Ross、Matt Ross、Amir Salek、Emad Samadiani、Chris Severn、Gregory Sizikov、Matthew Snelham、Jed Souter、Dan Steinberg、Andy Swing、Mercedes Tan、Gregory Thorson、Bo Tian、Horia Toma、Erick Tuttle、Vijay Vasudevan、Richard Walter、Walter Wang、Eric Wilcox 和 Doe Hyun Yoon。数据中心内张量处理单元的性能分析。载于 第*44*届年度国际计算机体系结构研讨会论文集，ISCA '17，第1–12页，美国纽约州纽约，2017年。计算机协会。 [7] Hong Jia-Wei 和 H. T. Kung。输入/输出复杂度：红蓝 pebble 游戏。载于 第十三届年度 *ACM* 计算理论研讨会论文集，STOC '81，第326–333页，美国纽约州纽约，1981年。计算机协会。 [8] Alok Aggarwal 和 S. Vitter, Jeffrey。排序及相关问题的输入/输出复杂度。*Commun. ACM*，31(9):1116–1127，1988年9月。[9] Michael Bender、Gerth Brodal、Rolf Fagerberg、Riko Jacob 和 Elias Vicari。输入/输出模型中的最优稀疏矩阵与稠密向量乘法。第47卷，第61–70页，2007年6月。 [10] Kameshwar Munagala 和 Abhiram Ranade。图算法的输入/输出复杂度。载于 第十届年度 *ACM-SIAM* 离散算法研讨会论文集，SODA '99，第687–694页，美国，1999年。工业与应用数学学会。 [11] Erik D. Demaine 和 Quanquan C. Liu。红蓝 pebble 游戏：计算缓存大小与内存传输之间权衡的复杂度。载于 第*30*届算法与体系结构并行性研讨会论文集，SPAA '18，第195–204页，美国纽约州纽约，2018年。计算机协会。 [12] Erik D. Demaine 和 Quanquan C. Liu。标准 pebble 游戏的不可近似性与难以进行 pebble 操作的图。载于 算法与数据结构，第313–324页，Cham，2017年。施普林格国际出版社。 [13] Quanquan Liu。红蓝与标准 pebble 游戏：顺序与并行模型中的复杂度及应用。硕士论文，电气工程与计算机科学系，麻省理工学院，马萨诸塞州，2018年。[14] Niels Gleinig 和 Torsten Hoefler。大输入下树与有向无环图上的红蓝 pebble 游戏。载于 结构信息与通信复杂性：第*29*届国际研讨会，*SIROCCO 2022*，帕德博恩，德国，*2022*年*6*月*27*日至*29*日，论文集，第135–153页，柏林、海德堡，2022年。施普林格出版社。 [15] H. Qi、E. R. Sparks 和 A. Talwalkar。Paleo：深度神经网络的性能模型。载于 国际学习表征会议（*ICLR*）论文集，2017年。 [16] A. Viebke、S. Memeti、S. Pllana 和 A. Abraham。Chaos：在英特尔 Xeon Phi 上训练卷积神经网络的并行化方案。《超级计算杂志》，2017年。 [17] Y. Oyama 等。预测 GPU 超级计算机上大规模分布式深度学习系统的异步随机梯度下降参数统计信息。载于 *IEEE* 国际大数据会议（*Big Data*），第66 75页，2016年 – 。[18] James Demmel 和 Grace Dinh。通信最优卷积神经网络。*CoRR*，abs/1802.06905，2018年。[19] Yann LeCun、John S. Denker 和 Sara A. Solla。最优脑损伤。载于 D. S. Touretzky（编），神经信息处理系统进展 *2*，第598–605页。Morgan-Kaufmann，1990年。 [20] Jonathan Frankle 和 Michael Carbin。彩票假设：寻找稀疏可训练神经网络。2018年。[21] Torsten Hoefler 、 Dan Alistarh 、 Tal Ben-Nun 、 Nikoli Dryden 和 Alexandra Peste。深度学习中的稀疏性：用于神经网络高效推理与训练的剪枝和增长。《机器学习研究杂志》，22(241):1–124，2021年。 [22] Alex Krizhevsky、Ilya Sutskever 和 Geoffrey E Hinton。使用深度卷积神经网络进行 ImageNet 分类。载于 神经信息处理系统进展，第1097–1105页，2012年。

[23] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Ł ukasz Kaiser 和 Illia Polosukhin。《注意力就是你所需要的一切》。载于 I. Guyon、U. V. Luxburg、S. Bengio、H. Wallach、R. Fergus、S. Vishwanathan 和 R. Garnett 编，《神经信息处理系统进展》，第30卷，第5998–6008页。Curran Associates, Inc.，2017年。[24] Andrei Ivanov、Nikoli Dryden、Tal Ben-Nun、Shigang Li 和 Torsten Hoefler。《数据移动就是你所需要的一切：一项优化 Transformer 的案例研究》，2020年。[25] Nikko Strom。《使用商用GPU云计算实现可扩展的分布式深度神经网络训练》。载于《国际语音通信协会第十六届年会》，2015年。[26] Song Han、Huizi Mao 和 William J. Dally。《深度压缩：通过剪枝、训练后量化和霍夫曼编码压缩深度神经网络》，2015年。[27] Alham Fikri Aji 和 Kenneth Heafield。《用于分布式梯度下降的稀疏通信》。《*2017*年自然语言处理实证方法会议论文集》，2017年。[28] Cedric Renggli、Saleh Ashkboos、Mehdi Aghagolzadeh、Dan Alistarh 和 Torsten Hoefler。《Sparcml：面向机器学习的高性能稀疏通信》。载于《国际高性能计算、网络、存储与分析会议论文集》，SC '19，美国纽约州纽约市，2019年。计算机协会。[29] Skanda Koppula、Lois Orosa、A. Giray Ya˘glıkçı、Roknoddin Azizi、Taha Shahroodi、Konstantinos Kanellopoulos 和 Onur Mutlu。《EDEN：利用近似 DRAM 实现节能、高性能的深度神经网络推理》。《第*52*届*IEEE/AC M*国际微体系结构年会论文集》，2019年10月。[30] Kartikeya Bhardwaj、Ching-Yi Lin、Anderson Sartor 和 Radu Mar- culescu。《面向物联网分布式深度学习推理的内存与通信感知模型压缩》。《*ACM*嵌入式计算系统汇刊》，18(5s)：1–22，2019年10月。[31] Y. Chen、T. Yang、J. Emer 和 V. Sze。《Eyeriss v2：面向移动设备中新兴深度神经网络的灵活加速器》。《*IEEE*新兴与精选电路及系统主题期刊》，9 308，2019 2 292( ): – 。[32] Laszlo A. Belady。《虚拟存储计算机置换算法研究》。《*IBM Syst. J.*》，5：78–101，1966年。[33] C Koulamas、SR Antony 和 R Jaen。《模拟退火在运筹学问题中的应用综述》。《*Omega*》，22 56，1994 1 41 ( ): – 。[34] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。《BERT：用于语言理解的深度双向Transformer预训练》。《*CoRR*》，abs/1810.04805，2018年。[35] Torsten Hoefler 和 Roberto Belli。《并行计算系统的科学基准测试：报告性能结果时向大众说明情况的十二种方法》。载于《国际高性能计算、网络、存储与分析会议论文集》，SC '15，美国纽约州纽约市，2015年。计算机协会。

附录

A. 如何生成随机前馈神经网络

我们描述如何生成宽度为w、深度为 d、稀疏度为 p 的随机前馈神经网络：对于每个非输出神经元，我们通过从 1 到 max（1，$\lceil 2 \cdot p \cdot$#下一层的神经元数量 $-1\rceil$）中均匀随机抽取一个整数 k，来确定它拥有多少条出边。然后，将该神经元连接到下一层中随机选出的 k 个神经元（$k \geq 1$确保前馈神经网络是连通的，且输出神经元与最后一个隐藏层中的所有神经元相连）。确定所有连接后，我们按照输出神经元逐层排列这些连接（根据定理1的证明，这确保连接布局所需的输入/输出操作次数不超过最优次数的两倍）。

B. 如何生成具有紧凑增长的前馈神经网络

我们从袋中已有的 $M_g - 2$ 个可直接计算的输入神经元开始（在我们的实验中，使用了 $M_g = 100$，300、 和 500 个）。

然后，我们从 1 迭代到 1000，每次迭代中：1）添加一个新神经元；2）从袋中随机选取另外 5 个神经元，并绘制它们到该神经元的输入连接；3）从袋中移除这 5 个随机选取的神经元中的最后一个。完成 1000 次迭代后，我们再添加一个神经元（输出神经元），并绘制袋中所有剩余神经元到该输出神经元的连接。