

第 24 课

让显卡们别再干等

本课讲 2018 年谷歌的一篇文章。Yanping Huang 和同事发明了 GPipe：把一个超大的模型切成几段，分给几块显卡，再让它们像流水线一样同时干活。

适合 10 岁左右

约 40 分钟

需要：铅笔、12 张小卡片、计时器、3 位同伴

1 热身：洗衣房里的时间问题

学校洗衣房有 1 台洗衣机、1 台烘干机和 1 张叠衣服的桌子。每一步都要 30 分钟。

洗完、烘干、叠好 1 筐衣服，一共要 _____ 分钟。

有 4 筐衣服。前一筐叠好了，下一筐才进洗衣机。一共要 _____ 分钟。

换个办法：第 1 筐一进烘干机，第 2 筐马上进洗衣机。4 筐一共要 _____ 分钟。

第 2 题里，洗衣机洗完第 1 筐，要空等 1 个小时。答案在第 4 节揭晓。科学家训练超大的 AI，用的正是这个洗衣房的办法。

2 模型太大，一块显卡装不下

第 22 课我们发现：网络越大，练得越多，本领就越强。可是网络要放在**显卡**里练（第 8 课）。每块显卡有一块**内存**，就像它的书桌。书桌上要摆下整个网络，还要摆下练习时的草稿。网络太大，书桌就摆不下。

第 8 课的 AlexNet 有大约 6000 万个“旋钮”，就已经要劈成两半，交给 2 块显卡。本课 GPipe 练出的翻译模型有 60 亿个旋钮，是 AlexNet 的 100 倍。

网络是**一层一层**叠起来的，像一座积木塔。题目从第 1 层进去，一层一层往上算出答案。对完答案，电脑再从最后一层往回调旋钮（第 1 课的**训练**）。所以练习分两趟：先**往前算**，再**往回改**。

3 办法 1：切成几段，可是大家在干等

最直接的办法是：按层把网络切成几段，每段放一块显卡。比如一个 40 层的网络，显卡 1 管第 1—10 层，显卡 2 管第 11—20 层，以此类推。

这很像接力赛：显卡 1 算完一大批练习题，才能交给显卡 2。下面是 4 块显卡的**时间表**。一行是一块显卡，一格是一段时间。涂色的格子在干活，**空格就是闲着**。

时间 →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
显卡 1	一大批															
显卡 2					一大批											
显卡 3									一大批							
显卡 4													一大批			

示意图：只画了“往前算”这一趟。“往回改”也一样排队，只是方向倒过来。

数一数空格：这张表一共 64 格，干活的只有 16 格，闲着的有 _____格。任何时候都只有 1 块显卡在干活。买了 4 块显卡，却只用上了大约 1 块的力气。

4 GPipe 的办法：切成小份，排成流水线

热身题答案：90 分钟，360 分钟，180 分钟。GPipe 用的就是洗衣房的办法。它把一大批练习题切成很多**小份**，叫**微批次**（“微”就是很小）。显卡 1 算完第 1 小份，马上交给显卡 2，自己接着算第 2 小份。这就像工厂的**流水线**：每个工位同时做不同的东西。

时间 →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
显卡 1	1	2	3	4				全部做完！ 比不切的时候少用 9 格时间								
显卡 2		1	2	3	4											
显卡 3			1	2	3	4										
显卡 4				1	2	3	4									

示意图：同样 4 块显卡，同样多的练习题，切成 4 小份。格子里的数字是第几小份。

原来要 16 格时间，现在只要 7 格。开头时，后面的显卡在等第 1 小份；结尾时，前面的显卡没活了。这些空格叫**气泡**，像水管里没装满水的地方。

切成几份	不切	4 份	8 份	16 份
4 块显卡闲着的时 间约占	75%	43%	27%	16%

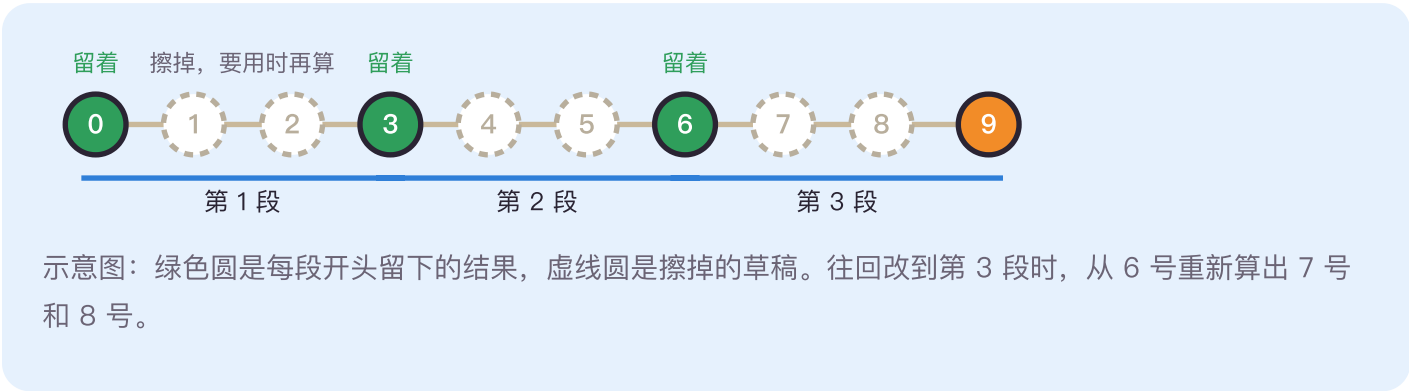
论文发现：小份数量达到显卡数量的 4 倍，气泡就小到几乎可以不管。不过不能切得太碎。每份太小，显卡里大部分“同学”没事干，反而浪费。

最后一起改。每一小份对完答案，先记下要改的地方。所有小份做完，再一起改一次旋钮。就像老师把 4 个小组的错题合起来，统一讲一遍。所以切不切小份，练出来的结果都一样。

每段要差不多重。流水线跑多快，看最慢的那一段。所以切的时候，每块显卡的活儿要差不多一样多。

5 另一个窍门：草稿纸不够，就擦掉重算

往前算时，每一层都会算出**中间结果**。往回改时要用到它们，所以通常全部记在书桌上，就像草稿纸。网络越大，草稿纸越多，书桌很快就堆满了。GPipe 的办法是：每一段只留下**开头那一份**结果，段里面的草稿都擦掉。往回改到这一段时，再从段开头重新算一遍。



这样大约要多算一遍“往前算”。可是书桌上省出一大块地方，能放下更大的网络。这叫**用时间换内存**。

6 结果：练出了当时的大个子

靠流水线和“擦掉重算”，谷歌练出了两个当时非常大的模型。

做什么	有多大	考得怎么样
看图认东西	5.57 亿个旋钮，名叫 AmoebaNet	在第 8 课的 ImageNet 认图考试里，第一次就猜对的有 84.4%。
翻译	60 亿个旋钮，128 层，是一个 Transformer（第 13 课）	一个模型就能翻译 100 多种语言。在大多数语言上，它比只练一种语言的翻译模型翻得更好。

7 这和今天有什么关系

今天训练聊天 AI，科学家会同时用好几种分工办法。**按层切段、排成流水线**就是其中一种。GPipe 是最早把这一招用好的论文之一。

流水线的排法后来改进了。比如“算完一份就马上往回改一份”，草稿更少。2024 年 DeepSeek-V3 的 DualPipe 几乎把气泡挤没了。“擦掉重算”成了训练大模型的常用办法，“最后一起改”也保留了下来。

诚实地说：GPipe 自己的排法已经不是最新的。可是想看懂新排法，得先看懂它。

8 动手：洗衣房流水线

玩法（3 个人干活，1 个人计时，约 15 分钟）

12 张小卡片当作 12 道练习题。3 个人坐成一排：A“洗”，画一个圆；B“烘”，在圆里打勾；C“叠”，对折放进盒子。用 3 种做法各玩一轮：① 一大批：12 张全做完，整叠交给下一个人。② 切成 3 小份：每做完 4 张就交出去，自己接着做下 4 张。③ 一张一张：每做完 1 张就交出去。

计时员喊“开始”，记下 C 等了几秒才拿到第一张，和一共用了几秒：

① 等 _____ 秒，共 _____ 秒 ② 等 _____ 秒，共 _____ 秒 ③ 等 _____ 秒，共 _____ 秒

做完第 ② 种，在时间表里填上第几小份（1、2、3）。第 1 格已经填好。

时间 →	1	2	3	4	5
A 洗	1				
B 烘					
C 叠					

你会发现：切得越碎，C 等得越短，总时间也越短。可是交接也花时间。有一个人特别慢，别人都得等他。显卡也是这样。

9 动手：草稿纸不够

玩法（2 个人，约 10 分钟）

2 → ①加 4 → ②乘 3 → ③减 8 → ④乘 2 → ⑤加 5
→ ⑥除以 5 → ⑦乘 4 → ⑧减 11 → ⑨乘 3

A 从 2 开始算到第 9 步。B 再从后往前问：“第 8 步是几？第 7 步呢？”一直问到第 1 步，就像“往回改”。A 每算一次，B 在旁边画一笔，最后数出一共算了几次。草稿纸格子每轮不同：9 格，每步都能写下；3 格，2 格只写第 3、6 步，1 格临时用；0 格，每问一次都从头心算。

9 格：_____ 次

3 格：_____ 次

0 格：_____ 次

你会发现：格子越少，重算越多。只留每段开头的答案，多算得不多，草稿纸却省了一大半。这就是“用时间换内存”。

10 词语小卡片

内存

显卡的“书桌”。网络和练习时的草稿都要摆在上面，摆不下就练不了。

流水线

几个工位同时开工，每个工位做完一份就往下传，自己马上做下一份。

微批次

把一大批练习题切成的小份。小份越多，闲着的空格（气泡）越少。

擦掉重算

只留少量中间结果，别的擦掉，要用时再算。多花时间，省下内存。

11 想一想

1. 洗衣房里，如果烘干机要 60 分钟，洗和叠还是 30 分钟，会发生什么？

2. 小份是不是切得越多越好？把一大批练习题切成一道一道的，有什么坏处？

3. 生活里也能“用时间换内存”：书包太重时，你会少带哪样东西，等要用时再想办法？举一个例子。

一句话记住：网络太大，就切成几段交给几块显卡；练习题再切成小份排成流水线，大家就不用干等。

给家长和老师：本课讲的是 Yanping Huang、Youlong Cheng、Ankur Bapna 等人（谷歌）2018 年的论文 [GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism](#) (arXiv 2018, NeurIPS 2019)。论文把 L 层网络按层切成 K 段放到 K 个加速器上，把小批次切成 M 个微批次依次流过各段，梯度同步累加后统一更新；并用重计算只保存段边界激活。论文实验主要用谷歌 TPU，讲义统称“显卡”。

洗衣房时间、40 层切 4 段、两张时间表、10 步草稿图和“小组错题”比喻都是示意。时间表只画前向。“闲着的时间”按气泡占比 $(K-1)/(M+K-1)$ 估算， $K=4$ 。草稿纸游戏是重计算的简化类比，参考次数：9 格 9 次，3 格 15 次，0 格 45 次。“结果一样”未计批归一化（训练时只用微批次统计量）。

想深入可看本站的 [第 24 天中文导读](#)，代码见开源项目 [30u30 第 24 天目录](#)（英文）。