



第 24 課

讓顯卡們別再乾等

本課講 2018 年穀歌的一篇文章。Yanping Huang 和同事發明瞭 GPipe：把一個超大的模型切成幾段，分給幾塊顯卡，再讓它們像流水線一樣同時幹活。

適合 10 歲左右

約 40 分鐘

需要：鉛筆、12 張小卡片、計時器、3 位同伴

1 熱身：洗衣房裡的時間題

學校洗衣房有 1 臺洗衣機、1 臺烘乾機和 1 張疊衣服的桌子。每一步都要 30 分鐘。

洗完、烘乾、疊好 1 筐衣服，一共要 _____ 分鐘。

有 4 筐衣服。前一筐疊好了，下一筐才進洗衣機。一共要 _____ 分鐘。

換個辦法：第 1 筐一進烘乾機，第 2 筐馬上進洗衣機。4 筐一共要 _____ 分鐘。

第 2 題裡，洗衣機洗完第 1 筐，要空等 1 個小時。答案在第 4 節揭曉。科學家訓練超大的 AI，用的正是這個洗衣房的辦法。

2 模型太大，一塊顯卡裝不下

第 22 課我們發現：網絡越大，練得越多，本領就越強。可是網絡要放在**顯卡**裡練（第 8 課）。每塊顯卡有一塊**內存**，就像它的書桌。書桌上要擺下整個網絡，還要擺下練習時的草稿。網絡太大，書桌就擺不下。

第 8 課的 AlexNet 有大約 6000 萬個「旋鈕」，就已經要劈成兩半，交給 2 塊顯卡。本課 GPipe 練出的翻譯模型有 60 億個旋鈕，是 AlexNet 的 100 倍。

網絡是**一層一層**疊起來的，像一座積木塔。題目從第 1 層進去，一層一層往上算出答案。對完答案，電腦再從最後一層往回調旋鈕（第 1 課的**訓練**）。所以練習分兩趟：先**往前算**，再**往回改**。

3 辦法 1：切成幾段，可是大家在乾等

最直接的辦法是：按層把網絡切成幾段，每段放一塊顯卡。比如一個 40 層的網絡，顯卡 1 管第 1—10 層，顯卡 2 管第 11—20 層，以此類推。

這很像接力賽：顯卡 1 算完一大批練習題，才能交給顯卡 2。下面是 4 塊顯卡的**時間表**。一行是一塊顯卡，一格是一段時間。塗色的格子在幹活，**空格就是閒著**。

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

示意圖：只畫了「往前算」這一趟。「往回改」也一樣排隊，只是方向倒過來。

數一數空格：這張表一共 64 格，幹活的只有 16 格，閒著的有 _____ 格。任何時候都只有 1 塊顯卡在幹活。買了 4 塊顯卡，卻只用上了大約 1 塊的力氣。

4 GPipe 的辦法：切成小份，排成流水線

熱身題答案：90 分鐘，360 分鐘，180 分鐘。GPipe 用的就是洗衣房的辦法。它把一大批練習題切成很多**小份**，叫**微批次**（「微」就是很小）。顯卡 1 算完第 1 小份，馬上交給顯卡 2，自己接著算第 2 小份。這就像工廠的**流水線**：每個工位同時做不同的東西。

時間 →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
顯卡 1	1	2	3	4				全部做完！ 比不切的時候少用 9 格時間								
顯卡 2		1	2	3	4											
顯卡 3			1	2	3	4										
顯卡 4				1	2	3	4									

示意圖：同樣 4 塊顯卡，同樣多的練習題，切成 4 小份。格子裡的數字是第幾小份。

原來要 16 格時間，現在只要 7 格。開頭時，後面的顯卡在等第 1 小份；結尾時，前面的顯卡沒活了。這些空格叫**氣泡**，像水管裡沒裝滿水的地方。

切成幾份	不切	4 份	8 份	16 份
4 塊顯卡閒著的時間約佔	75% 	43% 	27% 	16% 

論文發現：小份數量達到顯卡數量的 4 倍，氣泡就小到幾乎可以不管。不過不能切得太碎。每份太小，顯卡里大部分「同學」沒事幹，反而浪費。

最後一起改。每一小份對完答案，先記下要改的地方。所有小份做完，再一起改一次旋鈕。就像老師把 4 個小組的錯題合起來，統一講一遍。所以切不切小份，練出來的結果都一樣。

每段要差不多重。流水線跑多快，看最慢的那一段。所以切的時候，每塊顯卡的活兒要差不多一樣多。

5 另一個竅門：草稿紙不夠，就擦掉重算

往前算時，每一層都會算出**中間結果**。往回改時要用到它們，所以通常全部記在書桌上，就像草稿紙。網絡越大，草稿紙越多，書桌很快就堆滿了。GPipe 的辦法是：每一段只留下**開頭那一份**結果，段裡面的草稿都擦掉。往回改到這一段時，再從段開頭重新算一遍。



這樣大約要多算一遍「往前算」。可是書桌上省出一大塊地方，能放下更大的網絡。這叫**用時間換內存**。

6 結果：練出了當時的大個子

靠流水線和「擦掉重算」，谷歌練出了兩個當時非常大的模型。

做什麼	有多大	考得怎麼樣
看圖認東西	5.57 億個旋鈕，名叫 AmoebaNet	在第 8 課的 ImageNet 認圖考試裡，第一次就猜對的有 84.4%。
翻譯	60 億個旋鈕，128 層，是一個 Transformer（第 13 課）	一個模型就能翻譯 100 多種語言。在大多數語言上，它比只練一種語言的翻譯模型翻得更好。

7 這和今天有什麼關係

今天訓練聊天 AI，科學家會同時用好幾種分工辦法。**按層切段、排成流水線**就是其中一種。GPipe 是最早把這一招用好的論文之一。

流水線的排法後來改進了。比如「算完一份就馬上往回改一份」，草稿更少。2024 年 DeepSeek-V3 的 DualPipe 幾乎把氣泡擠沒了。「擦掉重算」成了訓練大模型的常用辦法，「最後一起改」也保留了下來。

誠實地說：GPipe 自己的排法已經不是最新的。可是想看懂新排法，得先看懂它。

8 動手：洗衣房流水線

玩法（3 個人幹活，1 個人計時，約 15 分鐘）

12 張小卡片當作 12 道練習題。3 個人坐成一排：A「洗」，畫一個圓；B「烘」，在圓裡打勾；C「疊」，對摺放進盒子。用 3 種做法各玩一輪：① 一大批：12 張全做完，整疊交給下一個人。② 切成 3 小份：每做完 4 張就交出去，自己接著做下 4 張。③ 一張一張：每做完 1 張就交出去。

計時員喊「開始」，記下 C 等了幾秒才拿到第一張，和一共用了幾秒：

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

做完第 ② 種，在時間表裡填上第幾小份（1、2、3）。第 1 格已經填好。

時間 →	1	2	3	4	5
A 洗	1				
B 烘					
C 疊					

你會發現：切得越碎，C 等得越短，總時間也越短。可是交接也花時間。有一個人特別慢，別人都得等他。顯卡也是這樣。

9 動手：草稿紙不夠

玩法（2 個人，約 10 分鐘）

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

A 從 2 開始算到第 9 步。B 再從後往前問：「第 8 步是幾？第 7 步呢？」一直問到第 1 步，就像「往回改」。A 每算一次，B 在旁邊畫一筆，最後數出一共算了幾次。草稿紙格子每輪不同：9 格，每步都能寫下；3 格，2 格只寫第 3、6 步，1 格臨時用；0 格，每問一次都從頭心算。

9 格：_____次

3 格：_____次

0 格：_____次

你會發現：格子越少，重算越多。只留每段開頭的答案，多算得不多，草稿紙卻省了一大半。這就是「用時間換內存」。

10 詞語小卡片

內存

顯卡的「書桌」。網絡和練習時的草稿都要擺在上面，擺不下就練不了。

流水線

幾個工位同時開工，每個工位做完一份就往下傳，自己馬上做下一份。

微批次

把一大批練習題切成的小份。小份越多，閒著的空格（氣泡）越少。

擦掉重算

只留少量中間結果，別的擦掉，要用時再算。多花時間，省下內存。

11 想一想

1. 洗衣房裡，如果烘乾機要 60 分鐘，洗和疊還是 30 分鐘，會發生什麼？

2. 小份是不是切得越多越好？把一大批練習題切成一道一道的，有什麼壞處？

3. 生活裡也能「用時間換內存」：書包太重時，你會少帶哪樣東西，等要用時再想辦法？舉一個例子。

一句話記住：網絡太大，就切成幾段交給幾塊顯卡；練習題再切成小份排成流水線，大家就不用幹等。

給家長和老師：本課講的是 Yanping Huang、Youlong Cheng、Ankur Bapna 等人（谷歌）2018 年的論文 [GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism](#) (arXiv 2018, NeurIPS 2019)。論文把 L 層網絡按層切成 K 段放到 K 個加速器上，把小批次切成 M 個微批次依次流過各段，梯度同步累加後統一更新；並用重計算只保存段邊界激活。論文實驗主要用谷歌 TPU，講義統稱「顯卡」。

洗衣房時間、40 層切 4 段、兩張時間表、10 步草稿圖和「小組錯題」比喻都是示意。時間表只畫前向。「閒著的時間」按氣泡佔比 $(K-1)/(M+K-1)$ 估算， $K=4$ 。草稿紙遊戲是重計算的簡化類比，參考次數：9 格 9 次，3 格 15 次，0 格 45 次。「結果一樣」未計批歸一化（訓練時只用微批次統計量）。

想深入可看本站的 [第 24 天中文導讀](#)，代碼見開源項目 [30u30 第 24 天目錄](#)（英文）。