Lab5

113062573 余侞璇

我總共實驗了三種不同的參數變化，分別是seq、heads與emb。基礎指令為

--batch 16

--seq 1024

--heads 32

--emb 256

--impl Flash2 / Pytorch

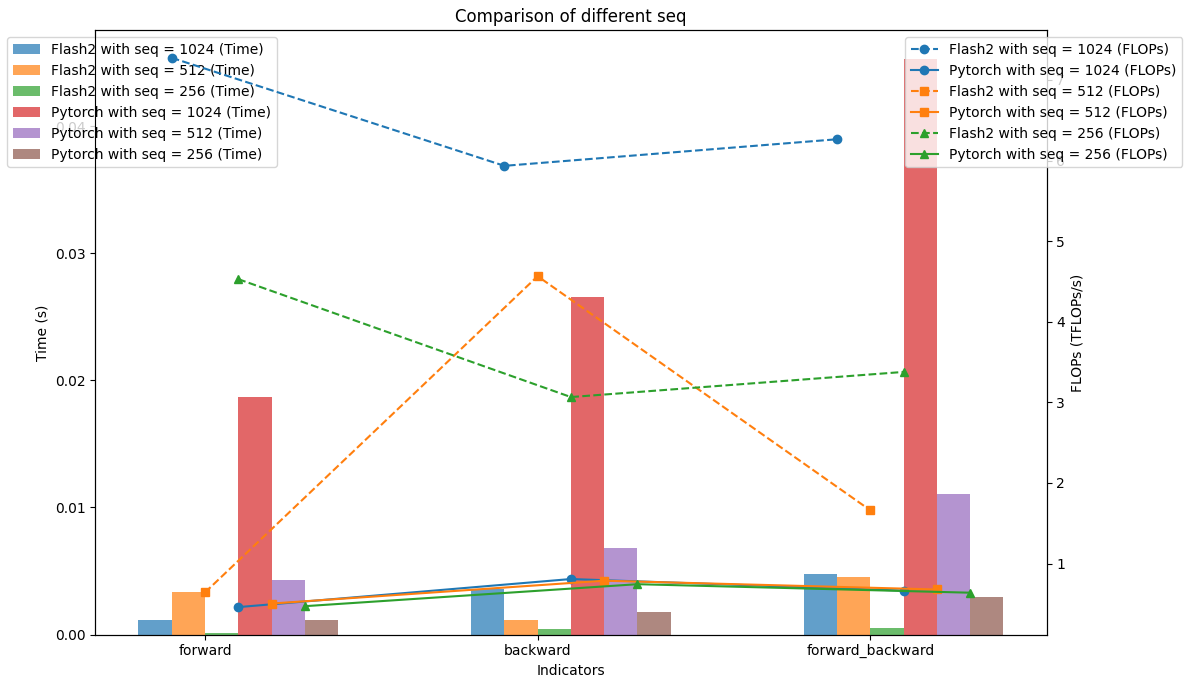
--causal

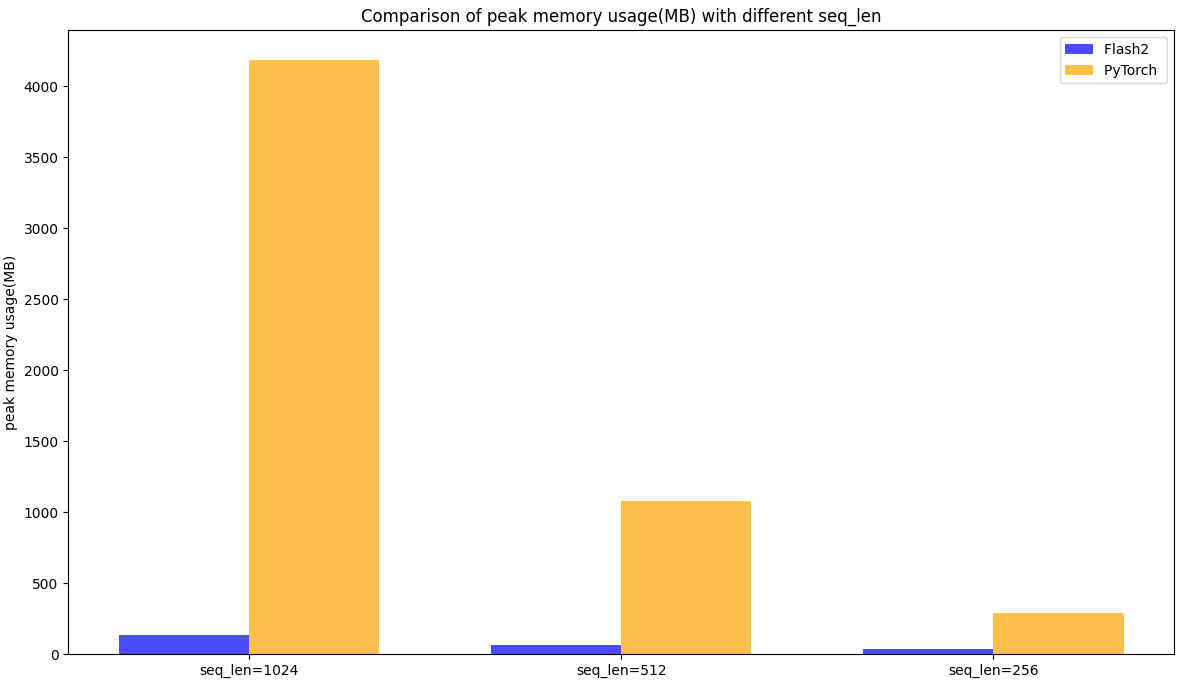
--repeat 30

基於上述指令去修改seq, heads與emb，觀察Flash2和Pytorch的變化。

* seq

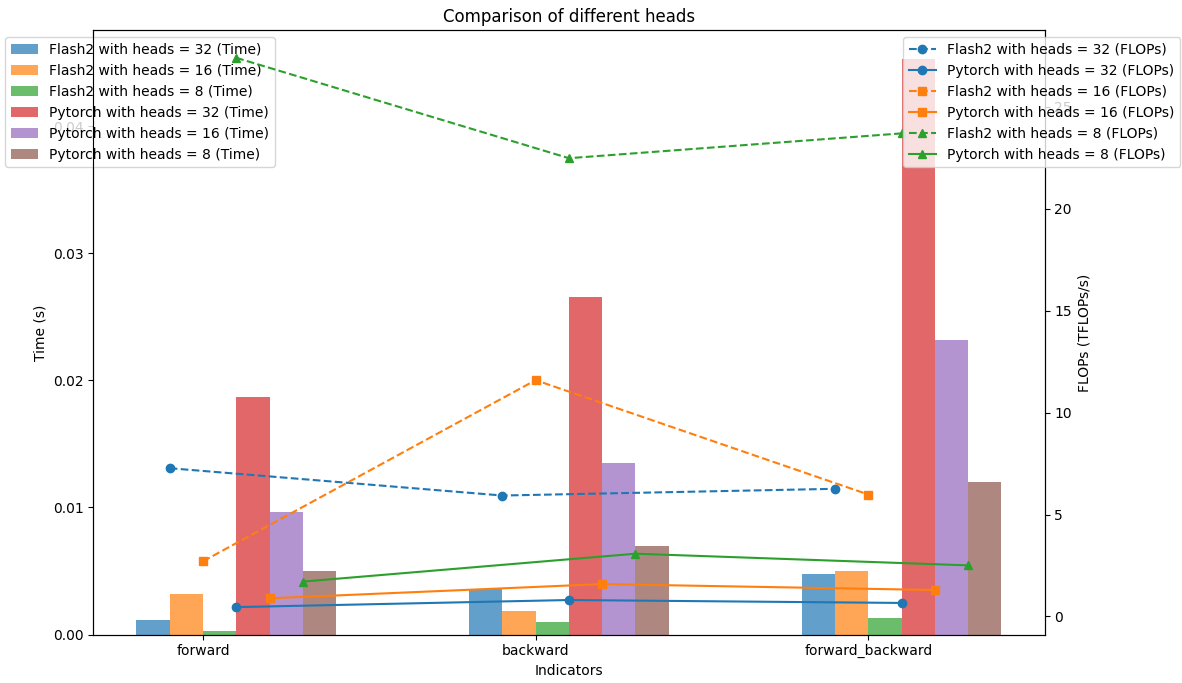
首先是seq長度分別設成1024、512、256，可以發現flash2所花的execution time遠低於pytorch，隨著需要處理的seq長度變小，時間大致上也越小。不過Flash2 的 FLOPs高於pytorch，可能是因為tiling等方式，會多次計算中間值與加權和，增加了不少計算量，但由於減少了對記憶體的access次數，故整體的execuation time仍然低於pytorch。

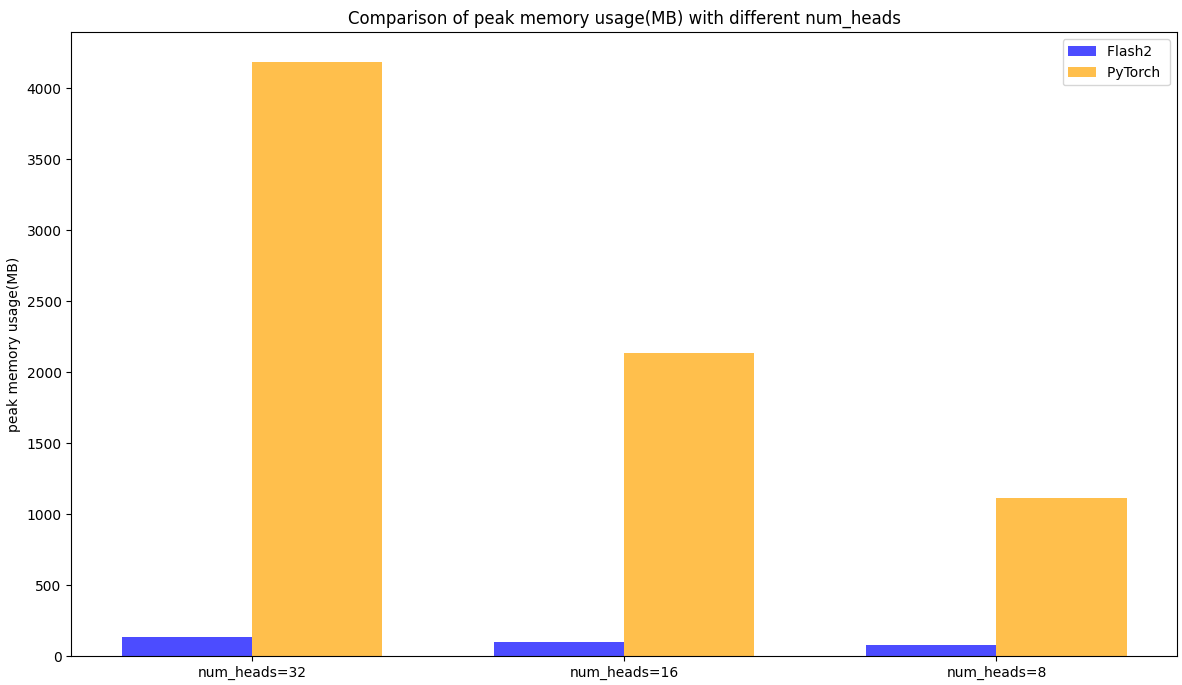




* heads

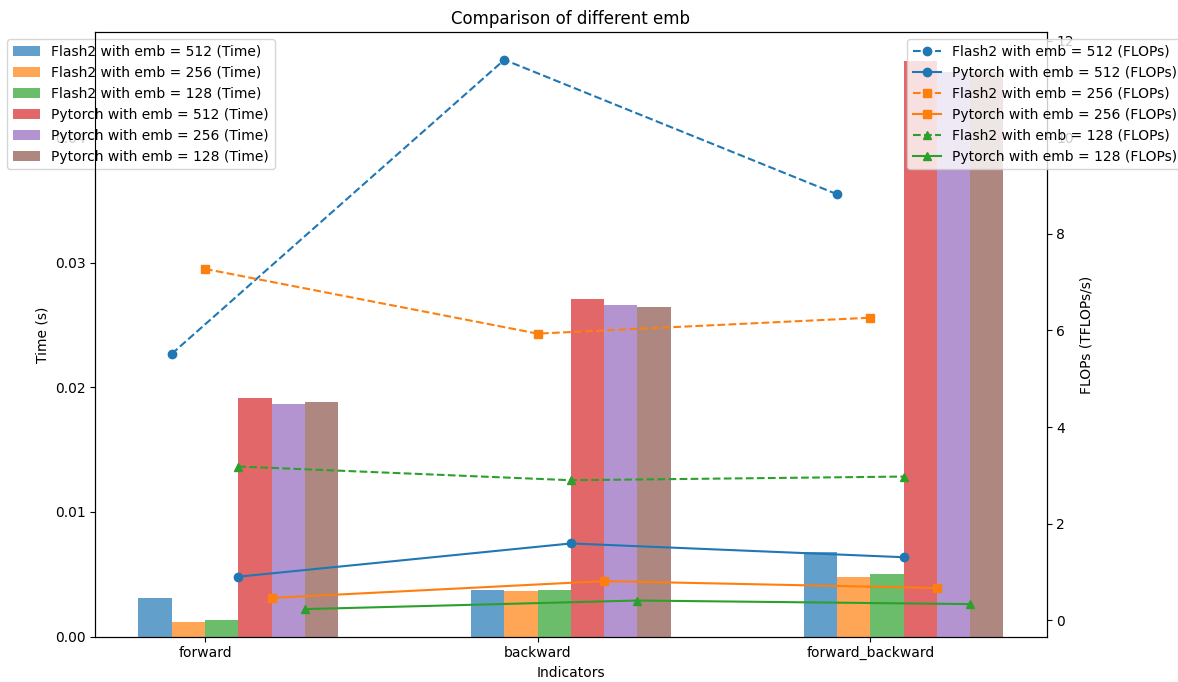
number of heads分別設成32、16、8。Execution time很明顯仍是flash2遠低於pytorch，不管head數量。至於FLOPs則是heads數量越少會變越大，可能是因為heads變少導致維度增加，矩陣計算的FLOPs也隨之成長。如果不想要太多的計算量，取得適中的heads數量非常重要。可以發現Flash2的memory usage隨著head數量變化並不大，幾乎持平，可能是因為每個block處理的數據量沒有改變，故memory usage差異不明顯。

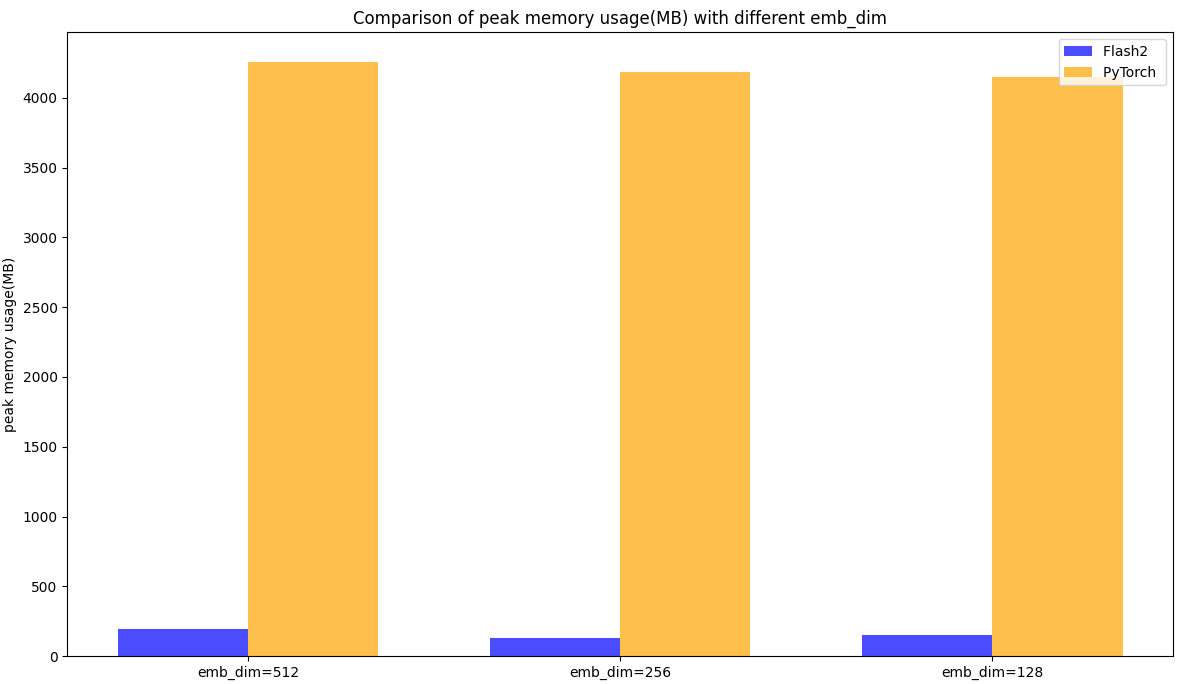




* emb

emb分別設成512、256、128。Execution time很明顯仍是flash2遠低於pytorch，FLOPs亦然，因為矩陣運算的複雜度就是直接依賴emb\_dim。這裡的memory usage就有隨著emb\_dim變小而跟著變化了。





一開始使用spec提供的範例指令：python lab5.py \ --batch\_size 32 \ --seq\_len 1024 \ --num\_heads 32 \ --emb\_dim 2048 \ --impl Flash2 \ --causal \ --repeats 30 \ --output benchmark\_result.json，但遇到了記憶體不夠用的情況。後來嘗試了調小batch\_size、seq\_len與num\_heads等組合，才終於找到現在使用的基本指令，避免記憶體不夠用的問題。

綜合上述，Flash2使用了tiling分塊處理matrix，故可以在seq\_len很長的情況下顯著縮短執行時間，同時也降低內存的壓力；另外， Flash2可以有效地將每個head的計算平行化，所以head數量增加時，Flash2的 execution time 增長速度也比 Pytorch慢；最後是emb\_dim，Flash2會將大matrix分塊處理以減少memory overhead，並且對softmax等步驟進行了inlined optimization。

不過，Flash2的 FLOPs都比 Pytorch高，但因其memory access的效率比Pytorch來的好，故整體執行時間仍然更短。

Flash2在sequence length長、multi-heads與 emb\_dim大的情況下，可以顯著降低執行時間與記憶體需求。