Lab5

113062573 余伽璇

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

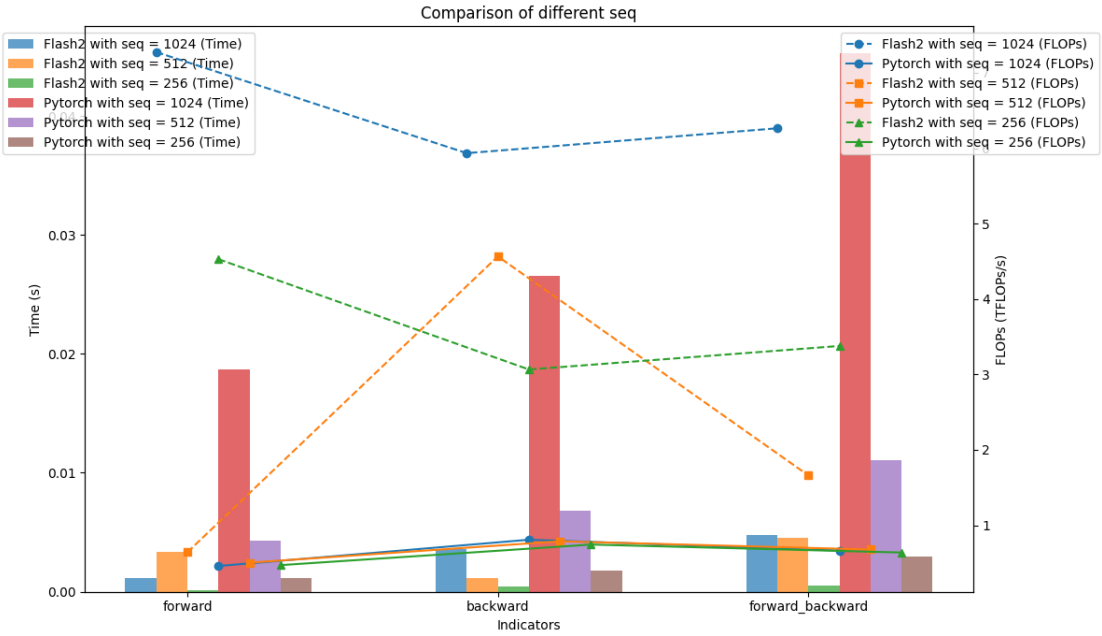
--causal

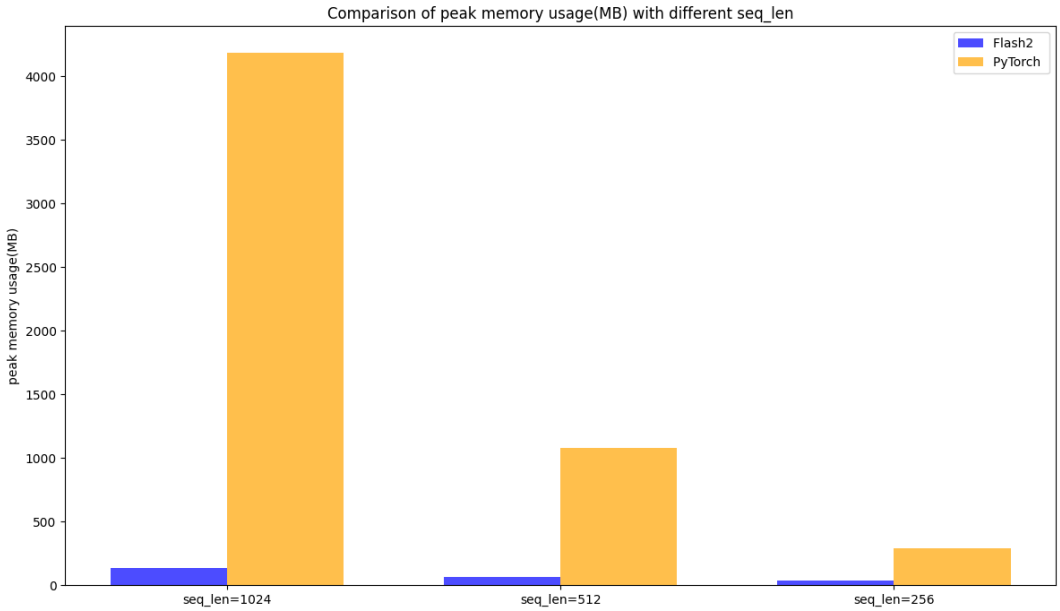
--repeat 30

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

- seq

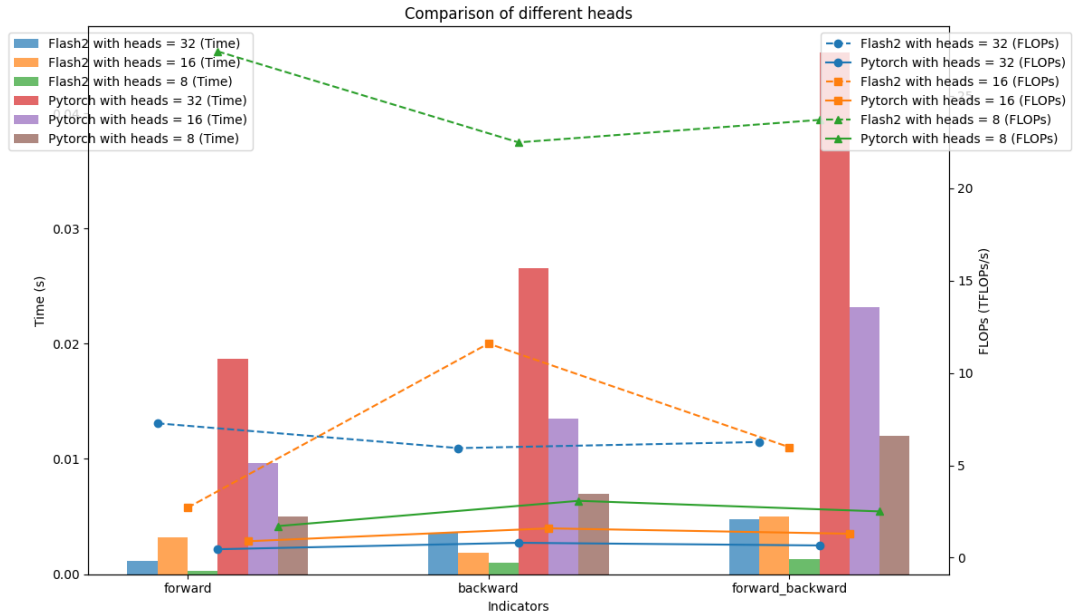
  首先是 seq 長度分別設成 1024、512、256，可以發現 flash2 所花的 execution time 遠低於 pytorch，隨著需要處理的 seq 長度變小，時間大致上也越小。不過 Flash2 的 FLOPs 高於 pytorch，可能是因為 tiling 等方式，會多次計算中間值與加權和，增加了不少計算量，但由於減少了對記憶體的 access 次數，故整體的 execuation time 仍然低於 pytorch。

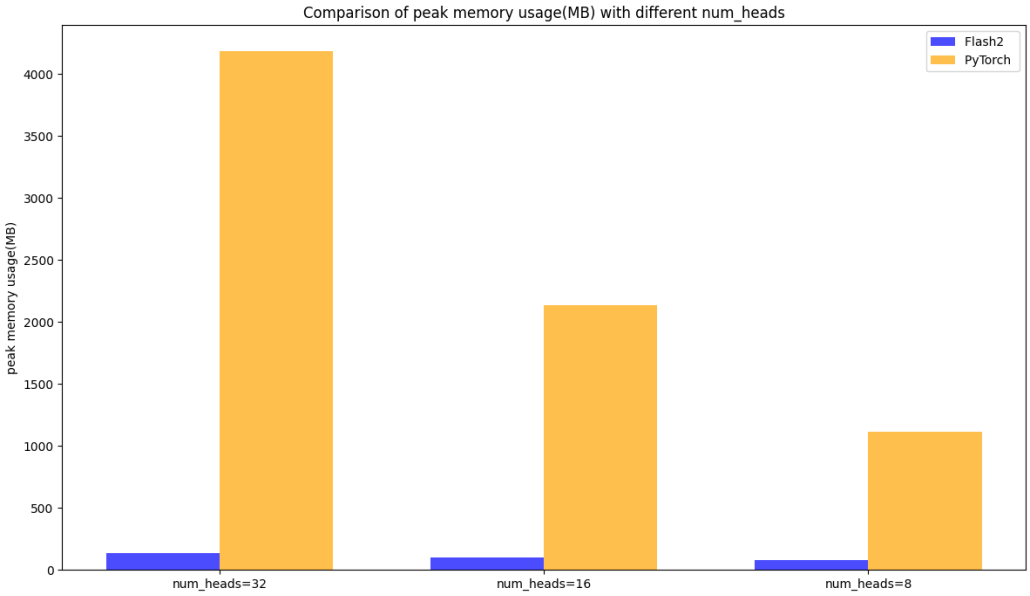




- heads

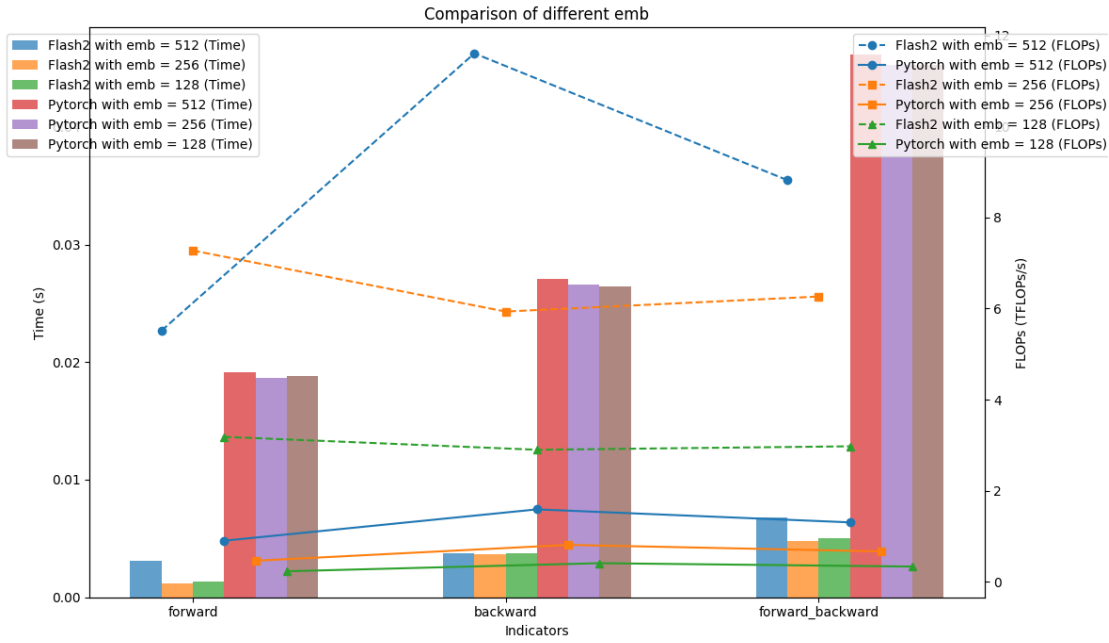
number of heads 分別設成 32、16、8。Execution time 很明顯仍是 flash2 遠低於 pytorch，不管 head 數量。至於 FLOPs 則是 heads 數量越少會變越大，可能是因為 heads 變少導致維度增加，矩陣計算的 FLOPs 也隨之成長。如果不想要太多的計算量，取得適中的 heads 數量非常重要。可以發現 Flash2 的 memory usage 隨著 head 數量變化並不大，幾乎持平，可能是因為每個 block 處理的數據量沒有改變，故 memory usage 差異不明顯。

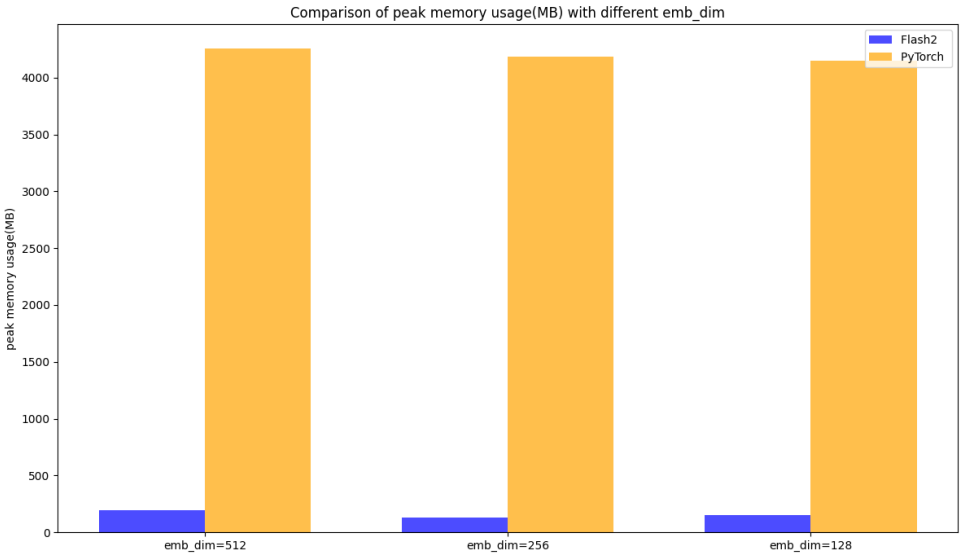




- emb

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。