112062576 Lab5 report

前言 :

對每一個參數設定不同的值一個一個去比較情況，所以總共會有
3*3*3*3*2*2=324 種不同的 benchmark_outputs，並且統一用 csv 紀錄。

Define parameter ranges

batch_sizes=(16 32 64)

seq_lengths=(512 1024 2048)

num_heads=(8 16 32)

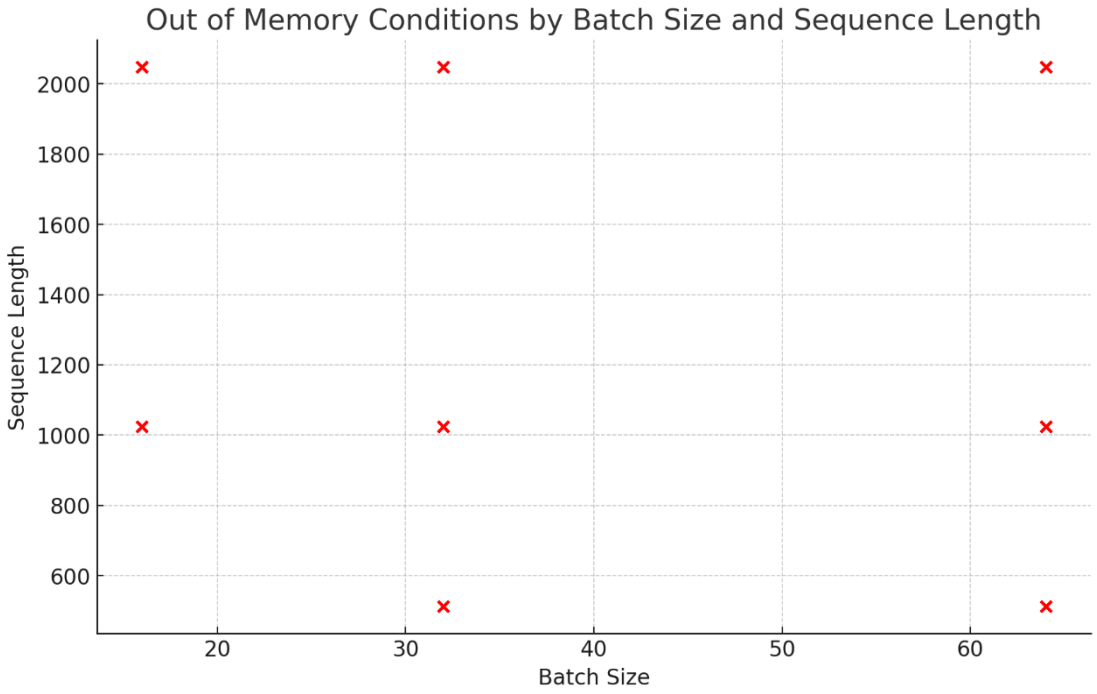
emb_dims=(512 1024 2048)

implementations=("Pytorch" "Flash2")

causal_flags=(true false)

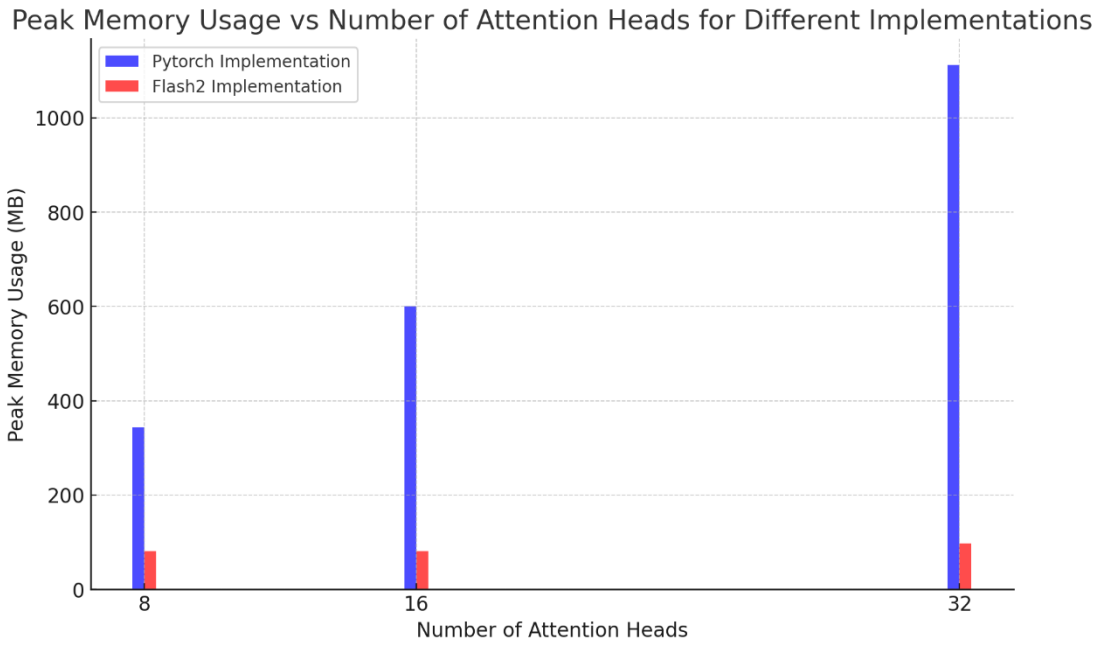
Analyze :

1. **Cuda out of memory**
   觀測何種狀況會導致 Cuda out of memory。基本上當 seq_lengths >= 1024 && batch_size >= 32 時就會發生了



2. **Peak Memory Usage vs Number of Attention Heads**

   隨著 Attention head 數量增加，記憶體使用量也顯著增加。尤其當注意力頭數從 8 增加到 16 和 32 時，記憶體使用量有大幅上升。在相同的注意力頭數量下，Flash2 相比於 PyTorch 實作，記憶體使用量低。PyTorch 在 Attention heads 數增加時，由於計算的並行性和矩陣操作變多，其內部實現可能無法像 Flash2 一樣進行高效優化，因此導致記憶體使用量更高。

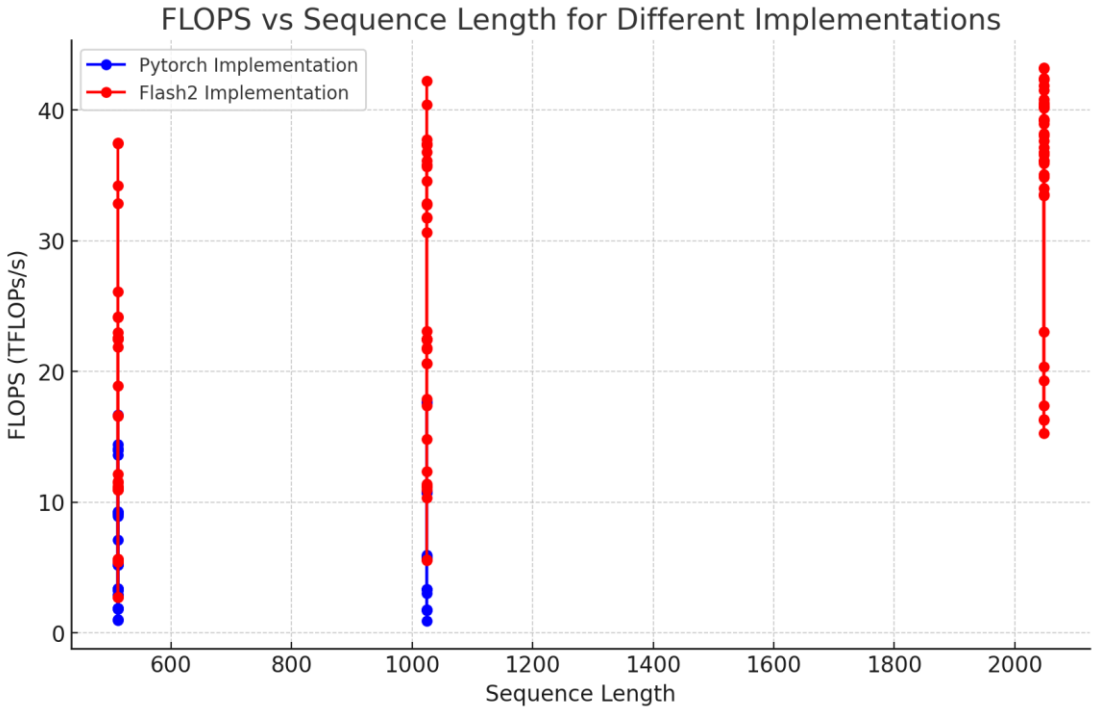


3. **FLOPS vs Sequence Length**

PyTorch 實作使用藍色，Flash2 使用紅色，每條線上的不同點是由於在不同的實驗中考慮了其他參數（例如 batch、heads 等）的變化而導致的。

從圖中可以看到，無論 Flash2 Pytorch 方式，FLOPS 雙雙都有提升，但可以看的出來 Flash2 明顯優於 PyTorch 成長幅度，甚至在序列長度增加到 1024 及以上時，Flash2 的 FLOPS 顯示出更高的計算效率，達到 40 TFLOPs/s 左右。

說明 Flash2 更適合應用於計算密集型且需要處理大量長序列的場景。



4. **Forward Time Vs Batch Size**

在固定參數的情況下（seq_lengths = 1024，num_heads = 16，emb_dims = 1024），不同 batch_sizes 下的 forward time 對比。

可以先發現第一點，當 batch_size 為 64 時，PyTorch 遇到了「記憶體不足（Out of Memory）」的情況，特別是在 seq_lengths、num_heads 和 emb_dims 都相對較高的情況下，模型需要使用大量的 Cuda memory 來進行矩陣計算和存儲中間結果。

隨著批量大小增加，每次前向傳播需要處理的數據量增多，這導致了計算負荷和內存需求的增加。因此，前向傳播時間隨著批量大小的增加而增加是符合預期的。