Lab 5

112062706 林泳成

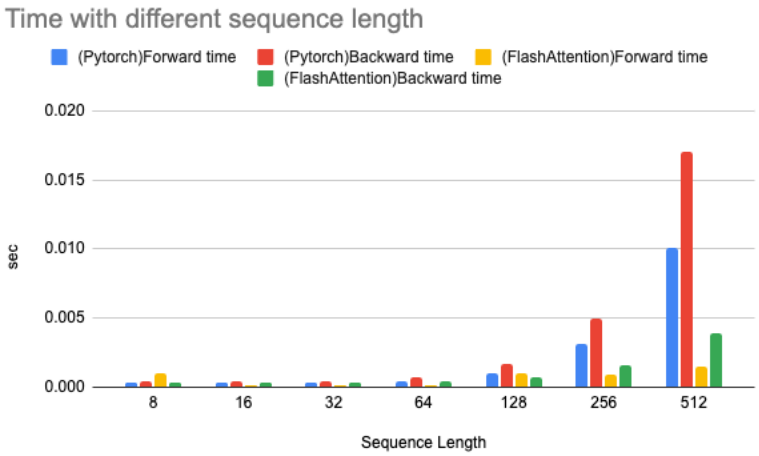
可修改變數：

```
--batch_size BATCH_SIZE
                      Batch size for training. Default is 32.
--seq_len SEQ_LEN     Sequence length. Default is 1024.
--num_heads NUM_HEADS
                      Number of attention heads. Must be divisible by emb_dim.
                      Default is 32.
--emb_dim EMB_DIM     Embedding dimension. Default is 2048.
--impl {Pytorch,Flash2}
                      Implementation type. Must be one of ['Pytorch', 'Flash2'].
                      Default is 'Flash2'.
--causal              If set, enables causal attention. Default is False.
--repeats REPEATS     Number of repeats for evaluation. Default is 30.
--output OUTPUT       The JSON filename for the benchmark result output file. Default
                      is benchmark_result.json.
```

實驗結果分析：

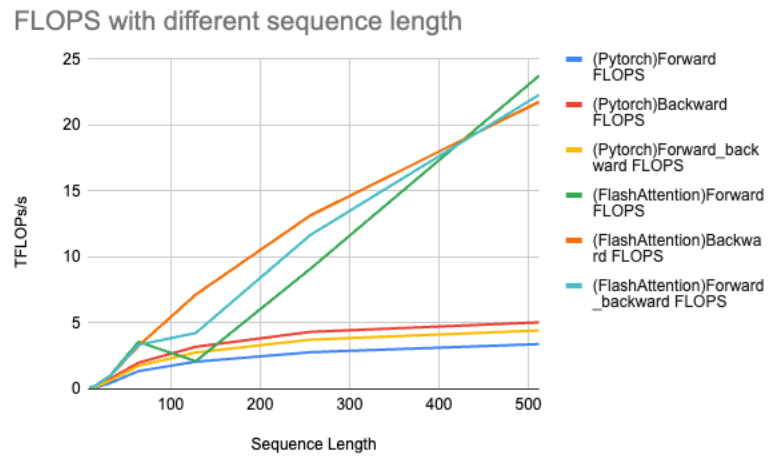
Sequence Length	8	16	32	64	128	256	512
(Pytorch)Forward time	0.0003096981347	0.000305258669	0.000310226343	0.000395425148	0.001039602794	0.003085763417	0.0101037588
(Pytorch)Forward FLOPS	0.02708640144	0.1099213074	0.4326445216	1.357705534	2.065677065	2.783730776	3.4006887
(Pytorch)Backward time	0.0004559183866	0.000434067919	0.000441029556	0.000674577429	0.001678634435	0.00496415969	0.01700838372
(Pytorch)Backward FLOPS	0.04599840809	0.1932556546	0.7608204833	1.989656369	3.198259852	4.325976161	5.05041204
(Pytorch)Forward_backward time	0.0007656165212	0.000739326588	0.000751255899	0.001070002578	0.002718237229	0.008049923107	0.02711214252
(Pytorch)Forward_backward FLOPS	0.03834834697	0.1588479486	0.6253023081	1.756115574	2.765098163	3.734789845	4.435617149
(Pytorch)peak_memory_usage	31.125	44.75	74.25	136.25	288.25	816.25	2640.25
(FlashAttention)Forward time	0.0009916025773	0.000130176544	0.000134394504	0.00015055269	0.001021877453	0.000944127701	0.001447060704
(FlashAttention)Forward FLOPS	0.008459647234	0.2577609677	0.9986846476	3.566000129	2.101508006	9.098276198	23.74450378
(FlashAttention)Backward time	0.0003163449466	0.000327986478	0.000357863083	0.000410993918	0.000750561505	0.001633745581	0.003946750835
(FlashAttention)Backward FLOPS	0.06629320374	0.2557607872	0.9376332332	3.26568647	7.152923618	13.14454143	21.76457281
(FlashAttention)Forward_backward time	0.001307947524	0.000458163023	0.000492257587	0.000561546608	0.001772438958	0.002577873282	0.005393811539
(FlashAttention)Forward_backward FLOPS	0.02244748162	0.2563290927	0.9543012842	3.346201655	4.240593298	11.66262565	22.29575198
(FlashAttention)peak_memory_usage	40.53173828	48.56298828	64.62548828	96.75048828	161.0004883	322.0004883	644.0004883

在 batch_size = 32, num_heads = 32, emb_dim = 2048, causal = true 的情況下改變 seq_len，並且測 100 次取平均。

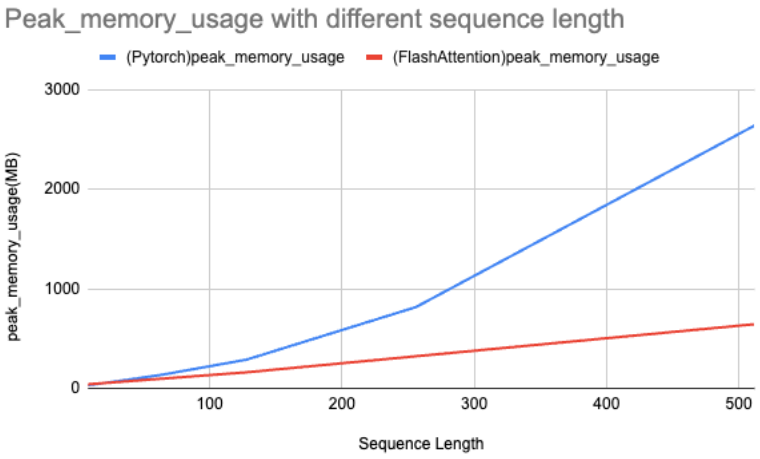


Pytorch 的總運算時間隨著 sequence 長度逐漸增加而大幅增加，相較之下 FlashAttension 的總運算時間則隨著 sequence 長度增加而微幅增加。可以看得出來 SRAM 的讀取速度相當快，且方法上也不像 Pytorch 的傳統實作要分成多個

步驟分別平行化。



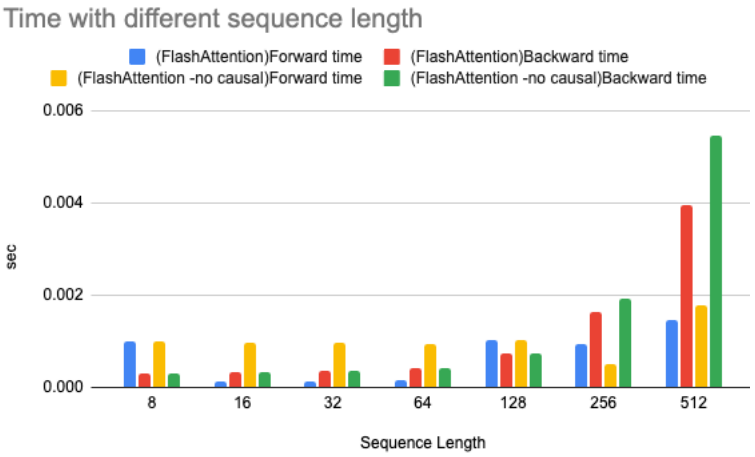
由於 FlashAttension 能更有效利用到 SRAM 的資源，因此每秒能做的浮點數運算會比傳統方法多上許多。



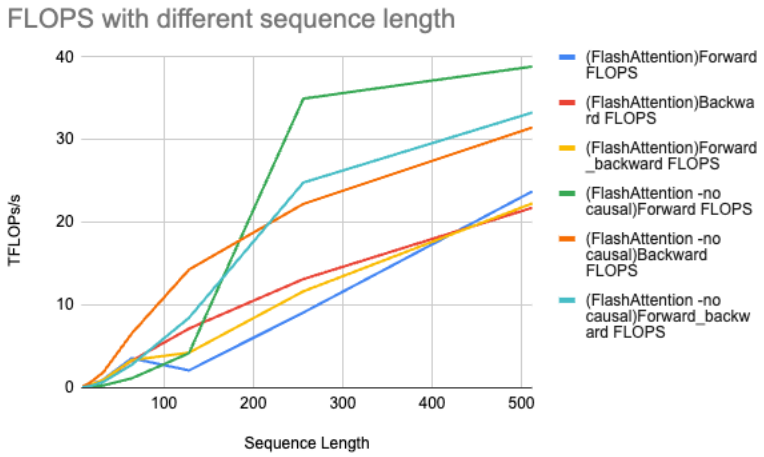
由於 FlashAttension 使用了 tiling 的技巧，所以不會同時佔用大量 memory，因此 peak memory 佔用量比傳統方法低很多。

Sequence Length	8	16	32	64	128	256	512
(FlashAttention -no causal)Forward time	0.000987135097	0.000981998182	0.000979668572	0.000941639877	0.001026387587	0.000491354987	0.001768871732
(FlashAttention -no causal)Forward FLOPS	0.01699586616	0.06833909184	0.2740063972	1.140289244	4.184547192	34.96427153	38.84932722
(FlashAttention -no causal)Backward time	0.000315039008	0.000331130027	0.000361614935	0.000408907793	0.000751366801	0.001931555156	0.005458685942
(FlashAttention -no causal)Backward FLOPS	0.1331360207	0.5066654967	1.855810075	6.564694053	14.29051459	22.23579935	31.47253637
(FlashAttention -no causal)Forward_backwarc	0.04509401294	0.1788713563	0.7004664492	2.782646228	8.455828112	24.81707472	33.27793142
(FlashAttention -no causal)peak_memory_usa	40.53173828	48.56298828	64.62548828	96.75048828	161.0004883	322.0004883	644.0004883

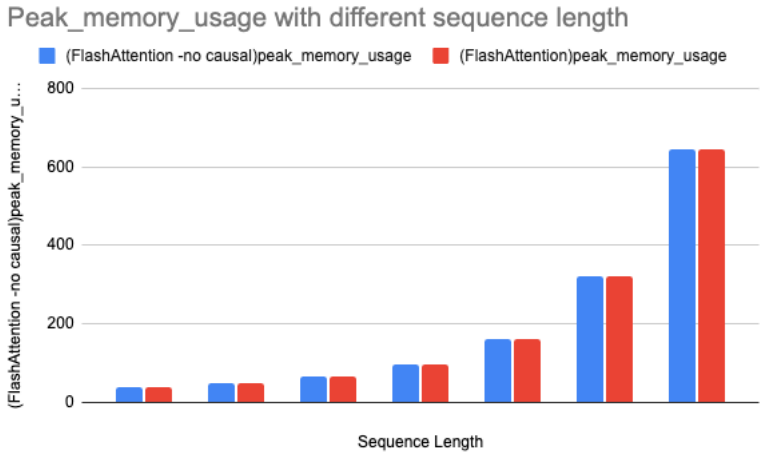
接下來的實驗設定和前一個相同，差別只在是否使用 causal attention



當 sequence 長度很短時，誤差會一定程度影響實驗結果。當 sequence 長度提升時，causal attention 方法能小幅度減少計算時間，因為整個 S 矩陣會有接近一半不需要做運算。



使用 causal attention 因為執行時間下降了，FLOPS 也相對下降了。

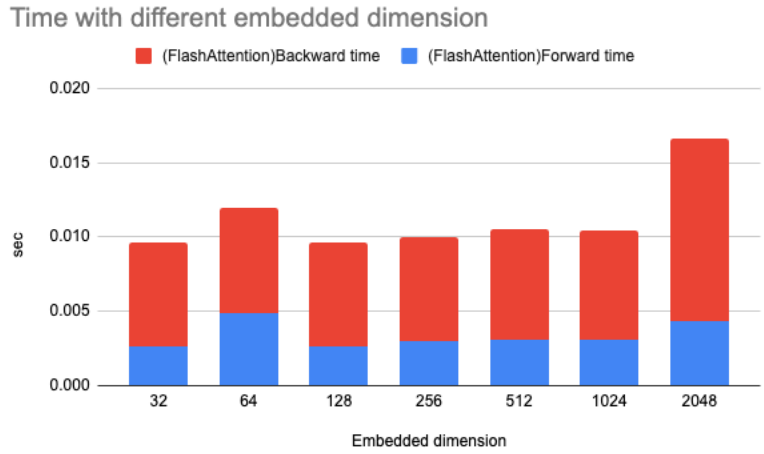


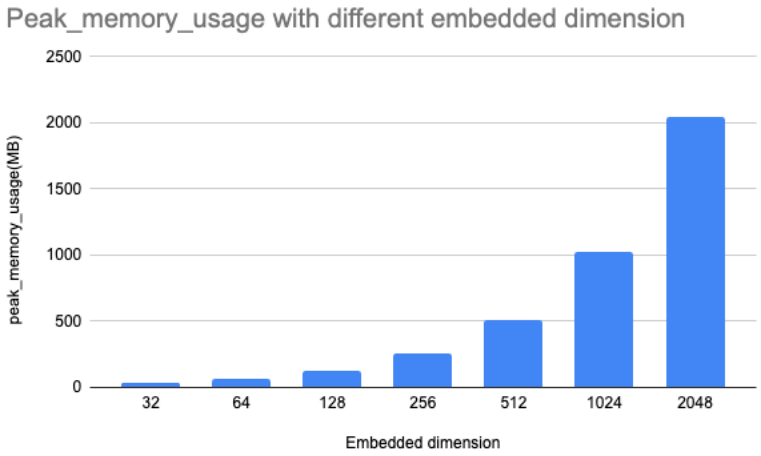
至於因為只是少計算一半的 S 矩陣，本質上還是 FlashAttention，因此 peak memory 的使用量不會改變。

Embedded dimension	32	64	128	256	512	1024	2048
(FlashAttention)Forward time	0.00261640735	0.00483633416	0.00266716033	0.00295583618	0.00305686978	0.00306344690	0.00431343192
(FlashAttention)Forward FLOPS	0.8207757283	0.8880625595	3.220629254	5.812185834	11.24017078	22.43207697	31.86301668
(FlashAttention)Backward time	0.00698102253	0.00715861481	0.00696194482	0.00700998666	0.00742843213	0.00738322424	0.01229439452
(FlashAttention)Backward FLOPS	0.7690433737	1.499929597	3.084603085	6.126926486	11.56359032	23.26878965	27.94748314
(FlashAttention)Forward_backward FLOPS	0.7831464112	1.2532263	3.122280896	6.033575258	11.46930106	23.02342681	28.96443665
(FlashAttention)peak_memory_usage	272.0004883	280.0004883	296.0004883	264.0004883	392.0004883	648.0004883	1288.000488

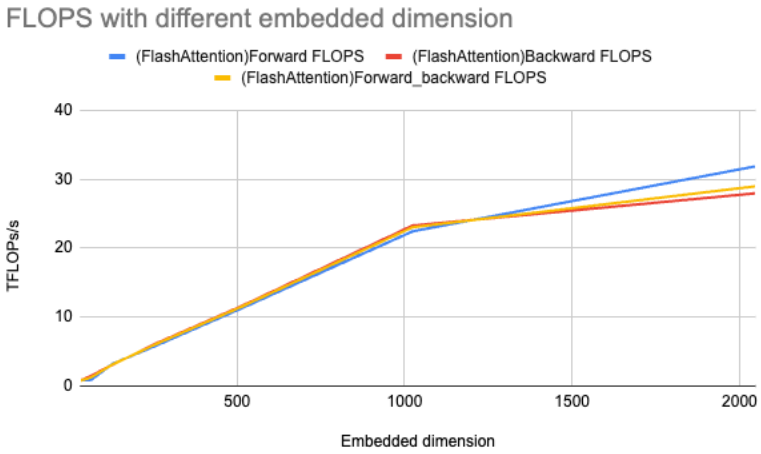
接下來的實驗設定是在 batch_size = 32, num_heads = 32, seq_len = 1024, causal = true 的情況下改變 emb_dim，並且測 100 次取平均。

由於 Pytorch 在這個條件的 memory 使用率會超過 8GB，因此沒有實驗數據。





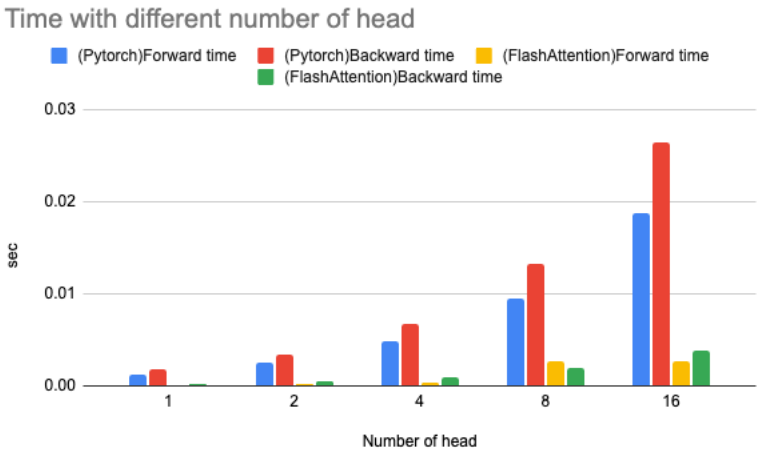
當 num_head 不變的情況下提升 emb_dim，相當於 head_size 提升，因此計算量也會上升，需要用到的 memory 也會上升。



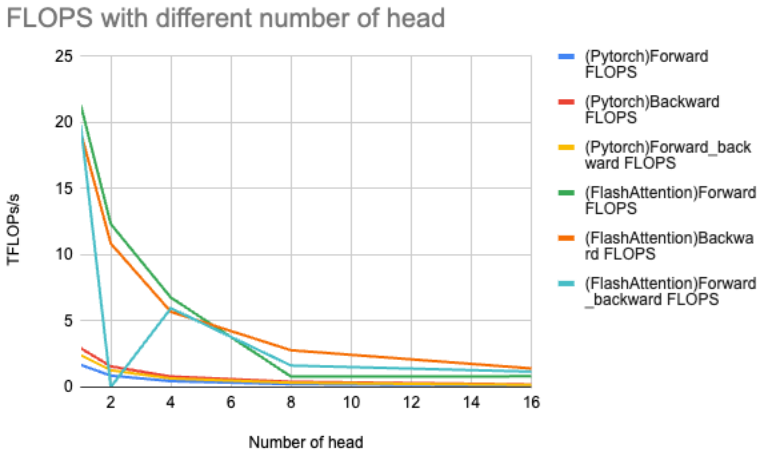
由於有額外使用到更多 SRAM，因此 FLOPS 有稍微提升。

Number of head	1	2	4	8	16
(Pytorch)Forward time	0.001278542231	0.002480797345	0.004800598944	0.009548310811	0.01871855209
(Pytorch)Forward FLOPS	1.679634506	0.8656425129	0.447336608	0.2249071789	0.1147248803
(Pytorch)Backward time	0.001829066997	0.003423504531	0.006694805125	0.01326279044	0.02635066062
(Pytorch)Backward FLOPS	2.935217315	1.568191037	0.8019216421	0.4047948389	0.2037409687
(Pytorch)Forward_backward FLOPS	2.41864154	1.273002791	0.6538432858	0.3294971464	0.1667700036
(Pytorch)peak_memory_usage	284.25	546.25	1058.25	2082.25	4130.25
(FlashAttention)Forward time	0.000100745136	0.000174195816	0.000317436953	0.002684945241	0.002628099297
(FlashAttention)Forward FLOPS	21.31600315	12.32798638	6.765071379	0.7998240022	0.8171242426
(FlashAttention)Backward time	0.000278528903	0.000494670247	0.000943343838	0.001928465068	0.003803887839
(FlashAttention)Backward FLOPS	19.27523156	10.853107	5.691147706	2.783928632	1.411374191
(FlashAttention)Forward_backward	19.81731404	0.000668866063	5.961538135	1.629205352	1.168564645
(FlashAttention)peak_memory_usag	20.25048828	24.50048828	33.00048828	74.00048828	140.0004883

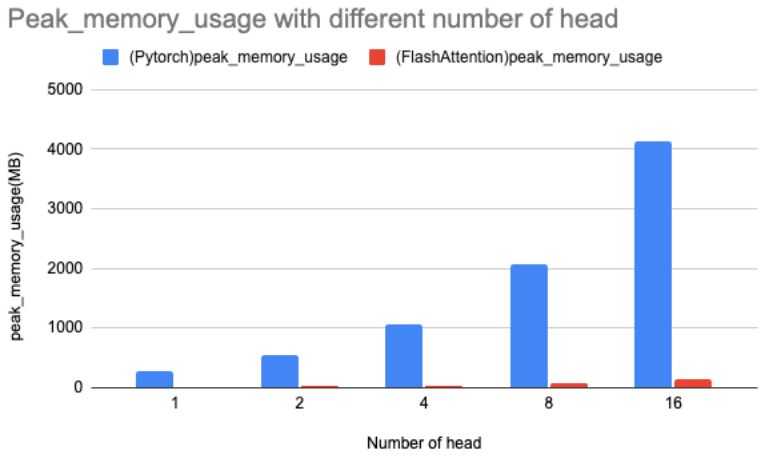
接下來的實驗設定是在 batch_size = 32, emb_dim = 32, seq_len = 1024, causal = true 的情況下改變 num_heads，並且測 100 次取平均。



不論是傳統方法或是 FlashAttetion 都是一個一個 head 去做計算的，所以基本上 head 每多一倍，計算時間就上升一倍。對於 FlashAttention 來說，因為算一個 head 的時間很短，因此計算時間不會成長太快。



不論是 Pytorch 還是 FlashAttetion 都因為 memory 逐漸被佔滿，開始無法好好使用 SRAM，FLOPS 因此降低。



Pytorch 基本上 memory 使用量是隨著 head 增加一倍，memory 也跟著增加一倍的。而 FlashAttention 的 memory 使用量則是根據 tiling 切出來的 block 而定，所以前面上升幅度不大，但當 head 上量越大，基本上 memory 使用量也是幾乎翻倍。

結論：

FlashAttetion 相較於 Pytorch 實作的傳統方法能更好地利用 tiling 來利用 SRAM，因此 FLOPS 都能維持在一個不錯的值，計算時間也比 Pytorch 快上許多。