

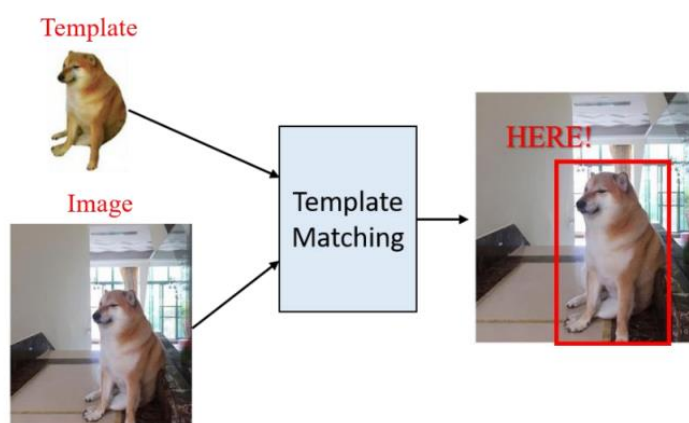
# Computer Architecture Final Project

312510239 王則惟

312510144 張理為

## 題目

這次的期末專題，我們參考 ICLAB 的題目，題目是 Template Matching with Image Processing，Template Matching 可以被視為一種非常基本的物體檢測形式。它是一種高層次的機器視覺技術，能夠識別圖像中與預定模板匹配的部分，這意味著我們可以使用包含我們想要檢測的物體的“模板”來檢測輸入圖像中的物體。例如，計算圖像相似度以找到匹配部分的方法有很多種，比如 Sum of Square Differences、Cross Correlation、Cross Coefficient 等等。而這次的題目，我們分別使用 8 種 operation，包括 Cross correlation、Max pooling、Right-Diagonal Flip、Left-Diagonal Flip、Vertical flip、Horizontal flip、Zoom-in、Shortcut + Brightness Adjustment，並利用 8 個 thread 及 16 個 thread 來做平行化的處理。



## 環境

本次 project 的環境是使用助教的 PPT 所提供的 virtual box，並匯入 Linux 虛擬機檔案來執行程式。

## 檔案說明

cuda1	2024/6/13 下午 10:01	CU 檔案
cuda2	2024/6/13 下午 10:01	CU 檔案
input	2024/6/13 下午 09:04	文字文件
main	2024/6/13 下午 11:56	CU 檔案
Makefile	2024/6/13 下午 10:00	檔案
ref	2024/6/13 下午 09:04	文字文件

本次的 project 包含了 6 個檔案，其中 main.cu 會先透過吃檔案的方式讀取 input.txt 以及 ref.txt 來計算每一個 operation 的結果，並且呼叫 cuda1.cu 以及 cuda2.cu 中的 function，然後驗證 CPU 計算的結果與 GPU 計算的結果是否相同，而 cuda1.cu

及 cuda2.cu 分別代表的是 thread 有 8 個或 16 個，最後再透過 Makefile 來 compile，且執行程式時，最後會產生一個 output.txt 檔案，內容會是矩陣經過每種 operation 後的結果會是如何。

## 演算法內容

- **Operation 1 - Cross Correlation**

Cross correlation 是透過先將 image 做各種 padding，再對 ref image 做 convolution，而這次的題目我們分別使用了三種 padding 的方式，包含 zero padding、edge padding，wrap padding，我們利用這三種 padding 的方式，將 8x8 的矩陣 padding 成一個 10x10 的矩陣，最後再透過對一個 3x3 的 ref image 做 convolution 得到我們想要的結果。

## 1. Zero Padding

[illegible]

## 2. Edge Padding

edge padding

1	1	2	3	4	5	6	7	8	8
1	1	2	3	4	5	6	7	8	8
9	9	10	11	12	13	14	15	16	16
17	17	18	19	20	21	22	23	24	24
25	25	26	27	28	29	30	31	32	32
33	33	34	35	36	37	38	39	40	40
41	41	42	43	44	45	46	47	48	48
49	49	50	51	52	53	54	55	56	56
57	57	58	59	60	61	62	63	64	64
57	57	58	59	60	61	62	63	64	64

## 3. Wrap Padding

wrap padding

64	57	58	59	60	61	62	63	64	57
8	1	2	3	4	5	6	7	8	1
16	9	10	11	12	13	14	15	16	9
24	17	18	19	20	21	22	23	24	17
32	25	26	27	28	29	30	31	32	25
40	33	34	35	36	37	38	39	40	33
48	41	42	43	44	45	46	47	48	41
55	49	50	51	52	53	54	55	56	49
63	57	58	59	60	61	62	63	64	56
8	1	2	3	4	5	6	7	8	1

\* Top row: 使用原圖的 last row  
 Bottom row: 使用原圖的 first row  
 Left col: 使用原圖的 right col  
 Right col: 使用原圖的 left col  
 Corner:  
 左上角: 使用原圖的右下角  
 右上角: 使用原圖的左下角  
 左下角: 使用原圖的右上角  
 右下角: 使用原圖的左上角

#### 4. Cross Correlation

Cross Correlation

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	0
0	9	10	11	12	13	14	15	16	0
0	17	18	19	20	21	22	23	24	0
0	25	26	27	28	29	30	31	32	0
0	33	34	35	36	37	38	39	40	0
0	41	42	43	44	45	46	47	48	0
0	49	50	51	52	53	54	55	56	0
0	57	58	59	60	61	62	63	64	0
0	0	0	0	0	0	0	0	0	0

 $\times$ 

1	0	-1
2	-1	3
0	-1	0

 $=$ 

-4	-1	2	5	8	11	14	-10
2	21	24	27	30	33	36	-3
2	45	48	51	54	57	60	5
2	69	72	75	78	81	84	13
2	93	96	99	102	105	108	21
2	117	120	123	126	129	132	29
2	141	144	147	150	153	156	37
67	231	235	239	243	247	251	117

#### ● Operation 2 - Max Pooling

Max Pooling 是卷積神經網絡 CNN 中的一種下采樣（或降維）技術，用於減少特徵圖的尺寸，同時保留最重要的特徵。最大池化通常應用於卷積層之後的特徵圖，以減少計算量並防止過擬合。而這次題目我們對一個 8x8 的 image 做 max pooling，結果會是一個 4x4 的矩陣，如下圖所示，我們會將矩陣四個一組，並取最大值來填入。

max pooling

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

 $=$ 

10	12	14	16
26	28	30	32
42	44	46	48
58	60	62	64

### ● Operation 3 – Right Diagonal Flip

右對角線翻轉(Right Diagonal Flip)，也稱為對角線對稱翻轉，是一種圖像變換操作。這種翻轉沿著圖像的主要對角線（從左上角到右下角）進行，使得圖像中的每個像素點  $(i,j)$  翻轉到  $(j,i)$  的位置。

right Diagonal flip

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63
8	16	24	32	40	48	56	64

### ● Operation 4 – Left Diagonal flip

左對角線翻轉(Left Diagonal Flip)，也稱為反對角線翻轉，是一種圖像變換操作。這種翻轉沿著圖像的次要對角線（從左下角到右上角）進行，使得圖像中的每個像素點  $(i,j)$  翻轉到  $(n-1-j, n-1-i)$  的位置，其中  $n$  是矩陣的維度。

Left-Diagonal Flip

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

⇒

64	56	48	40	32	24	16	8
63	55	47	39	31	23	15	7
62	54	46	38	30	22	14	6
61	53	45	37	29	21	13	5
60	52	44	36	28	20	12	4
59	51	43	35	27	19	11	3
58	50	42	34	26	18	10	2
57	49	41	33	25	17	9	1

### ● Operation 5 - Vertical Flip

垂直翻轉 (Vertical Flip) 是圖像處理中的一種基本變換操作。這種操作將圖像沿垂直軸 (從上到下) 進行翻轉, 使得圖像中的每個像素點  $(i,j)$  翻轉到  $(n-1-i,j)$  的位置, 其中  $n$  是圖像的高度。

Vertical flip

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

⇒

57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

### ● Operation 6 - Horizontal Flip

水平翻轉 (Horizontal Flip) 是圖像處理中的一種基本變換操作。這種操作將圖像沿水平軸 (從左到右) 進行翻轉, 使得圖像中的每個像素點  $(i,j)$  翻轉到  $(i,n-1-j)$  的位置, 其中  $n$  是圖像的寬度。

Horizontal flip

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

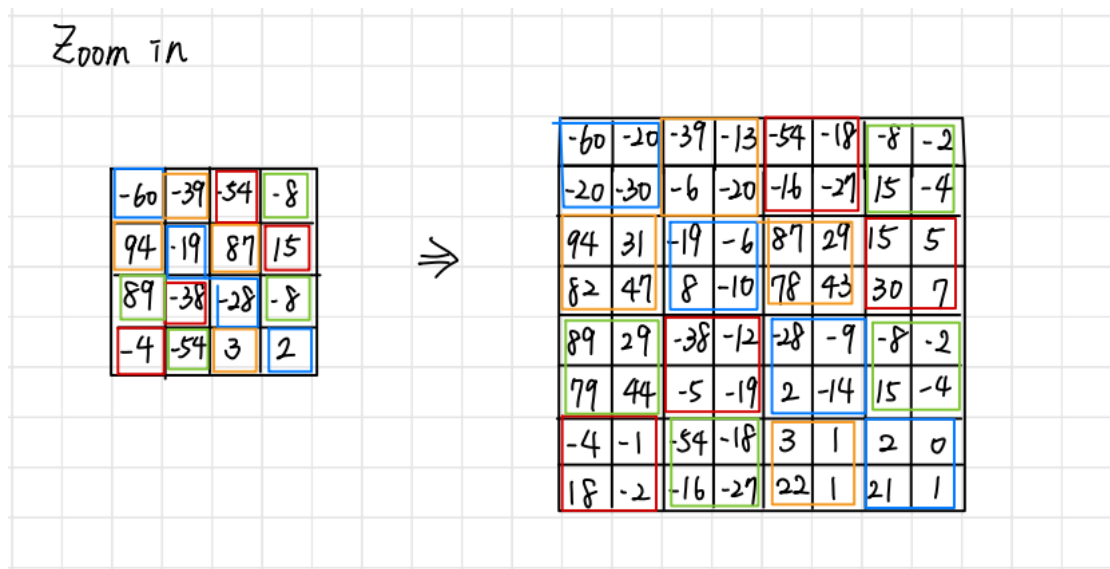
⇒

8	7	6	5	4	3	2	1
16	15	14	13	12	11	10	9
24	23	22	21	20	19	18	17
32	31	30	29	28	27	26	25
40	39	38	37	36	35	34	33
48	47	46	45	44	43	42	41
56	55	54	53	52	51	50	49
64	63	62	61	60	59	58	57

縮放 (Zoom In) 是圖像處理中的一種操作，用於放大圖像的一部分，使其顯示得更清晰和詳細。放大圖像通常涉及插值技術，以在圖像被放大後生成新的像素值。而本題使用的是一種基於 linear equation 的放大操作，該操作會將圖像放大為原來的兩倍，並且每個原始像素會對應放大圖像的左上角像素。給定 alpha 為 0.5，adjust 為 20，並遵循以下規則：

$$\text{Image}'(x' + 1, y' + 1) = \text{alpha} \times \text{Image}(x, y)$$

這些規則將用於從原始圖像生成新的放大圖像。而本題的原始圖像大小為  $8 \times 8$ ，因此經過 zoom-in 後的結果圖像將會是原始圖像的兩倍大小，即為  $16 \times 16$ ，下圖的原始圖像以  $8 \times 8$  為例。





## ● Operation 8 – Shortcut & Brightness Adjustment

Shortcut & Brightness Adjustment 為一種經典的圖像處理操作，其中選擇圖像的中心並進行亮度調整。具體操作包括使用線性函數調整圖像亮度，並根據圖像大小進行縮放。具體內容如下：

### 1. 亮度調整公式：

- 使用線性函數來調整圖像的亮度：

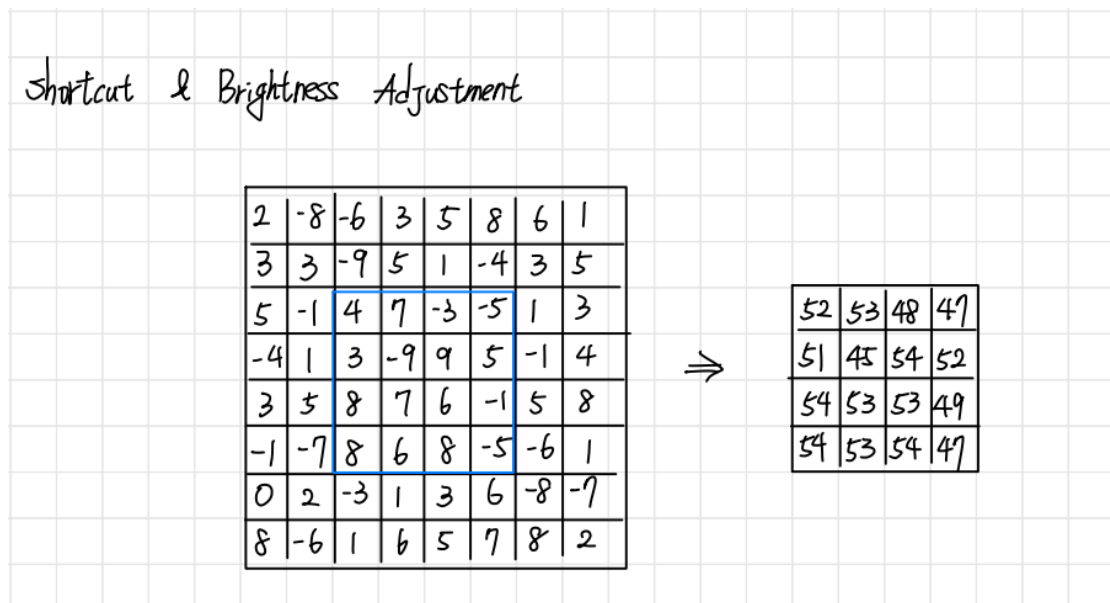
$$\text{Image}'(x,y) = \alpha \times \text{Image}(x,y) + \text{bias}$$

- 其中， $\alpha$  設為 0.5，bias 設為 50。

### 2. 縮放規則：

- 如果圖像的形狀為 8x8 或 16x16，則進行縮小操作，將圖像大小減半。
- 如果圖像的形狀為 4x4，則不進行縮放，保持原來大小，但依然進行亮度調整。

因本題所使用的 image 為 8x8，因此縮放效果如下圖所示





## 切割方式

本次 project 分別使用 8 個 thread 及 16 個 thread 來做平行化的處理，下方的圖為本次 project 的切割方式，其中紅色框框表示一個 thread 的運算範圍，黑色框框代表一個 block 的範圍。

### 1. Cross Correlation

8 個 thread 的 thread\_num=8，block\_num=1。

16 個 thread 的 thread\_num=2，block\_num=8。

Cross Correlation

8 thread

64	63	62	61	60	59	58	57
56	55	54	53	52	51	50	49
48	47	46	45	44	43	42	41
40	39	38	37	36	35	34	33
32	31	30	29	28	27	26	25
24	23	22	21	20	19	18	17
16	15	14	13	12	11	10	9
8	7	6	5	4	3	2	1

16 thread

64	63	62	61	60	59	58	57
56	55	54	53	52	51	50	49
48	47	46	45	44	43	42	41
40	39	38	37	36	35	34	33
32	31	30	29	28	27	26	25
24	23	22	21	20	19	18	17
16	15	14	13	12	11	10	9
8	7	6	5	4	3	2	1

### 2. Max Pooling

8 個 thread 的 thread\_num=2，block\_num=4。

16 個 thread 的 thread\_num=4，block\_num=4。

Max pooling

8 thread

64	62	60	58
48	46	44	42
32	30	28	26
16	14	12	10

16 thread

64	62	60	58
48	46	44	42
32	30	28	26
16	14	12	10

### 3. Right Diagonal Flip

8 個 thread 的 thread\_num=8, block\_num=1。

16 個 thread 的 thread\_num=2, block\_num=8。

Right-Diagonal Flip

8 thread

64	56	48	40	32	24	16	8
63	55	47	39	31	23	15	7
62	54	46	38	30	22	14	6
61	53	45	37	29	21	13	5
60	52	44	36	28	20	12	4
59	51	43	35	27	19	11	3
58	50	42	34	26	18	10	2
57	49	41	33	25	17	9	1

16 thread

64	56	48	40	32	24	16	8
63	55	47	39	31	23	15	7
62	54	46	38	30	22	14	6
61	53	45	37	29	21	13	5
60	52	44	36	28	20	12	4
59	51	43	35	27	19	11	3
58	50	42	34	26	18	10	2
57	49	41	33	25	17	9	1

### 4. Left Diagonal Flip

8 個 thread 的 thread\_num=8, block\_num=1。

16 個 thread 的 thread\_num=2, block\_num=8。

Left-Diagonal Flip

8 thread

8	7	6	5	4	3	2	1
16	15	14	13	12	11	10	9
24	23	22	21	20	19	18	17
32	31	30	29	28	27	26	25
40	39	38	37	36	35	34	33
48	47	46	45	44	43	42	41
56	55	54	53	52	51	50	49
64	63	62	61	60	59	58	57

16 thread

8	7	6	5	4	3	2	1
16	15	14	13	12	11	10	9
24	23	22	21	20	19	18	17
32	31	30	29	28	27	26	25
40	39	38	37	36	35	34	33
48	47	46	45	44	43	42	41
56	55	54	53	52	51	50	49
64	63	62	61	60	59	58	57

## 5. Vertical Flip

8 個 thread 的 thread\_num=8, block\_num=1。

16 個 thread 的 thread\_num=2, block\_num=8。

Vertical flip  
8 thread

8	7	6	5	4	3	2	1
16	15	14	13	12	11	10	9
24	23	22	21	20	19	18	17
32	31	30	29	28	27	26	25
40	39	38	37	36	35	34	33
48	47	46	45	44	43	42	41
56	55	54	53	52	51	50	49
64	63	62	61	60	59	58	57

16 thread

8	7	6	5	4	3	2	1
16	15	14	13	12	11	10	9
24	23	22	21	20	19	18	17
32	31	30	29	28	27	26	25
40	39	38	37	36	35	34	33
48	47	46	45	44	43	42	41
56	55	54	53	52	51	50	49
64	63	62	61	60	59	58	57

## 6. Horizontal Flip

8 個 thread 的 thread\_num=8, block\_num=1。

16 個 thread 的 thread\_num=2, block\_num=8。

Horizontal flip:  
8 thread

57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

16 thread

57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

## 7. Zoom in

8 個 thread 的 thread\_num=8, block\_num=1。

16 個 thread 的 thread\_num=1, block\_num=16。

Zoom in

8 thread

64	21	63	21	62	20	61	20	60	20	59	19	58	19	57	19
62	32	62	31	61	31	60	30	60	30	59	29	58	29	58	28
56	18	55	18	54	18	53	17	52	17	51	17	50	16	49	16
57	28	56	27	56	27	55	26	54	26	54	25	53	25	52	24
48	16	47	15	46	15	45	15	44	14	43	14	42	14	41	13
52	24	51	23	50	23	50	22	49	22	48	21	48	21	47	20
40	13	39	13	38	12	37	12	36	12	35	11	34	11	33	11
46	20	46	19	45	19	44	18	44	18	43	17	42	17	42	16
32	10	31	10	30	10	29	9	28	9	27	9	26	8	25	8
41	16	40	15	40	15	39	14	38	14	38	13	37	13	36	12
24	8	23	7	22	7	21	7	20	6	19	6	18	6	17	5
36	12	35	11	34	11	34	10	33	10	32	9	32	9	31	8
16	5	15	5	14	4	13	4	12	4	11	3	10	3	9	3
30	8	30	7	29	7	28	6	28	6	27	5	26	5	26	4
8	2	7	2	6	2	5	1	4	1	3	1	2	0	1	0
25	4	24	3	24	3	23	2	22	2	22	1	21	1	20	0

16 thread

64	21	63	21	62	20	61	20	60	20	59	19	58	19	57	19
62	32	62	31	61	31	60	30	60	30	59	29	58	29	58	28
56	18	55	18	54	18	53	17	52	17	51	17	50	16	49	16
57	28	56	27	56	27	55	26	54	26	54	25	53	25	52	24
48	16	47	15	46	15	45	15	44	14	43	14	42	14	41	13
52	24	51	23	50	23	50	22	49	22	48	21	48	21	47	20
40	13	39	13	38	12	37	12	36	12	35	11	34	11	33	11
46	20	46	19	45	19	44	18	44	18	43	17	42	17	42	16
32	10	31	10	30	10	29	9	28	9	27	9	26	8	25	8
41	16	40	15	40	15	39	14	38	14	38	13	37	13	36	12
24	8	23	7	22	7	21	7	20	6	19	6	18	6	17	5
36	12	35	11	34	11	34	10	33	10	32	9	32	9	31	8
16	5	15	5	14	4	13	4	12	4	11	3	10	3	9	3
30	8	30	7	29	7	28	6	28	6	27	5	26	5	26	4
8	2	7	2	6	2	5	1	4	1	3	1	2	0	1	0
25	4	24	3	24	3	23	2	22	2	22	1	21	1	20	0

## 8. Shortcut & Brightness Adjustment

8 個 thread 的 thread\_num=2, block\_num=4。

16 個 thread 的 thread\_num=4, block\_num=4。

Shortcut + Brightness Adjustment

8 thread

73	72	72	71
69	68	68	67
65	64	64	63
61	60	60	59

16 thread

73	72	72	71
69	68	68	67
65	64	64	63
61	60	60	59



## 模擬結果

```

Operation 1 - Cross Correlation
CPU Execution time = 0.026011 ms .
Thread number = 8 , Execution time = 73.44 ms .
Thread number = 16 , Execution time = 0.76 ms .
Operation 2 - Max Pooling
CPU Execution time = 0.001270 ms .
Thread number = 8 , Execution time = 0.48 ms .
Thread number = 16 , Execution time = 0.35 ms .
Operation 3 - Right Diagonal Flip
CPU Execution time = 0.004228 ms .
Thread number = 8 , Execution time = 0.37 ms .
Thread number = 16 , Execution time = 0.73 ms .
Operation 4 - Left Diagonal Flip
CPU Execution time = 0.001424 ms .
Thread number = 8 , Execution time = 0.37 ms .
Thread number = 16 , Execution time = 0.62 ms .
Operation 5 - Vertical Flip
CPU Execution time = 0.001216 ms .
Thread number = 8 , Execution time = 0.31 ms .
Thread number = 16 , Execution time = 1.47 ms .
Operation 6 - Horizontal Flip
CPU Execution time = 0.194840 ms .
Thread number = 8 , Execution time = 0.44 ms .
Thread number = 16 , Execution time = 0.60 ms .
Operation 7 - Zoom in
CPU Execution time = 0.014297 ms .
Thread number = 8 , Execution time = 1.23 ms .
Thread number = 16 , Execution time = 1.06 ms .
Operation 8 - Shortcut & Brightness Adjustment
CPU Execution time = 0.001411 ms .
Thread number = 8 , Execution time = 0.51 ms .
Thread number = 16 , Execution time = 0.30 ms .

```

	Op1	Op2	Op3	Op4	Op5	Op6	Op7	Op8
8	73.44ms	0.48ms	0.37ms	0.37ms	0.31ms	0.44ms	1.23ms	0.51ms
16	0.76ms	0.35ms	0.73ms	0.62ms	1.47ms	0.6ms	1.06ms	0.3ms

由上面表格我們可以看到，當我們使用 thread\_num=8 時，他在做一些較複雜的運算時的 runtime 明顯的比 thread\_num=16 時大很多，不過再做一些對角化或是一些 flip 的時候，兩者的 runtime 卻無明顯差別，甚至 thread\_num=16 時還比 thread\_num=8 時大，我們認為這是因為這幾個 operation 因為沒有做甚麼複雜的運算，只是單純的移動 image 的內容，因此這些 operation 用來做平行化的處理無法得到我們預期的效果，不過當我們運算量變大時，例如 cross correlation 中要實作 convolution，很明顯的我們的平行化處理就很成功，在 max pooling 時，因為要四個比大小，因此 thread\_num=16 時也較佳，而在 zoom in 以及 brightness adjustment，因為要根據一些 linear equation 來計算結果，因此當 thread\_num 變大時也有得到相應的好處。

## 結論

根據上述的實驗結果，我們可以得出以下結論：

- **平行化處理的有效性：**在進行較為複雜的運算時，增加執行緒數目可以顯著提高計算效率。例如，在 cross correlation 中實作卷積運算時，使用 16 個執行緒的運行時間顯著少於使用 8 個執行緒，這表明平行化處理在這些計算密集型操作中能夠充分發揮其優勢。
- **簡單操作的平行化限制：**對於一些簡單的圖像操作（如對角化和翻轉），平行化處理的優勢並不明顯，甚至在某些情況下，使用更多的執行緒反而會增加運行時間。這是因為這些操作主要涉及數據移動，並不涉及大量的計算，平行化處理在這些場景下無法顯著減少運行時間，反而可能因為多執行緒管理和調度的開銷而增加總運行時間。
- **適用場景的不同：**平行化處理在需要大量計算的場景中，也能夠取得不錯的效果。這些操作涉及基於線性方程的計算，增加執行緒數目能夠有效分攤計算負荷，從而縮短運行時間。

總結來說，平行化處理在計算密集型操作中能夠顯著提高運行效率，但在僅涉及簡單數據移動的操作中，其效果有限。選擇適當的執行緒數目應根據具體操作的計算特性來決定，以平衡平行化處理的優勢與執行緒管理的開銷。

## 分工

312510239 王則惟	312510144 張理為
cuda1.cu	main.cu
cuda2.cu	Makefile
書面報告	書面報告