This is a sample of the report, but applicable for all homework.

[112062542] [賴琮翰] This is for double verification.

Don't copy the problem statement, just write the answer.

Please write down the question number in unit of sub-question.

Please write down the sub-question number even if you don't know how to solve it.

(1.3.1)

Batch size = 1

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net                                      [1, 10]                   --
├─Sequential: 1-1                        [1, 6, 28, 28]            --
│    └─Conv2d: 2-1                       [1, 6, 28, 28]            150
│    └─ReLU: 2-2                         [1, 6, 28, 28]            --
├─Sequential: 1-2                        [1, 6, 14, 14]            --
│    └─MaxPool2d: 2-3                    [1, 6, 14, 14]            --
├─Sequential: 1-3                        [1, 16, 10, 10]           --
│    └─Conv2d: 2-4                       [1, 16, 10, 10]           2,400
│    └─ReLU: 2-5                         [1, 16, 10, 10]           --
├─Sequential: 1-4                        [1, 16, 5, 5]             --
│    └─MaxPool2d: 2-6                    [1, 16, 5, 5]             --
├─Sequential: 1-5                        [1, 120, 1, 1]            --
│    └─Conv2d: 2-7                       [1, 120, 1, 1]            48,000
│    └─ReLU: 2-8                         [1, 120, 1, 1]            --
├─Sequential: 1-6                        [1, 84]                   --
│    └─Linear: 2-9                       [1, 84]                   10,080
│    └─ReLU: 2-10                        [1, 84]                   --
├─Sequential: 1-7                        [1, 10]                   --
│    └─Linear: 2-11                      [1, 10]                   840
==========================================================================================
Total params: 61,470
Trainable params: 61,470
Non-trainable params: 0
Total mult-adds (M): 0.42
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.05
Params size (MB): 0.25
Estimated Total Size (MB): 0.30
==========================================================================================
```

(1.3.2)

Batch size = 1

	Type	Input Activation Size	Output Activation Size	Activation Function
conv1	Convolution	1*32*32 = 1024	6*28*28 = 4704	ReLU
maxpool2	Pooling	6*28*28 = 4704	6*14*14 = 1176	
conv3	Convolution	6*14*14 = 1176	16*10*10 = 1600	ReLU
maxpool4	Pooling	16*10*10 = 1600	16*5*5 = 400	
conv5	Convolution	16*5*5 = 400	120*1*1 = 120	ReLU
fc6	Fully connected	120*1*1 = 120	84	ReLU
output	Fully connected	84	10	

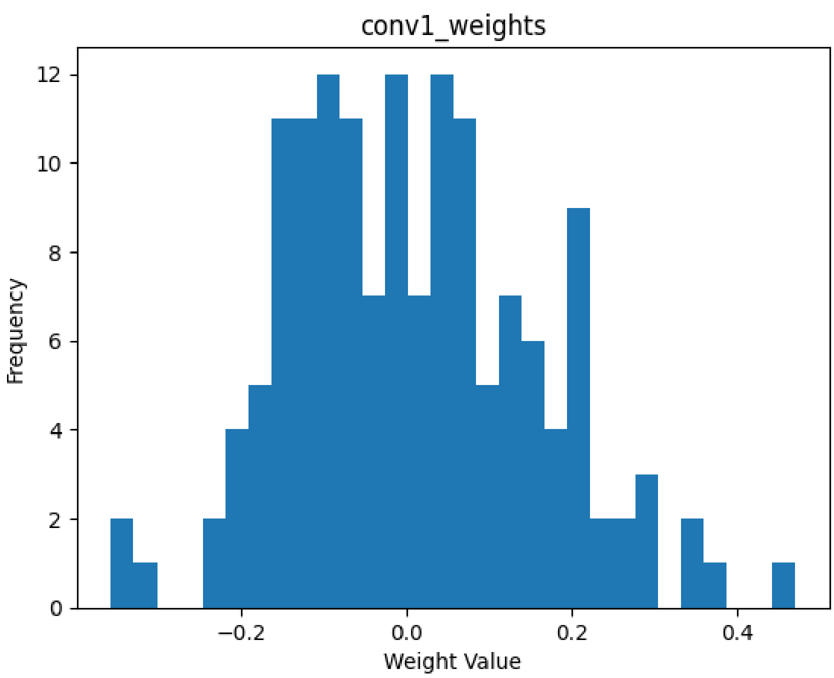
(1.3.3)

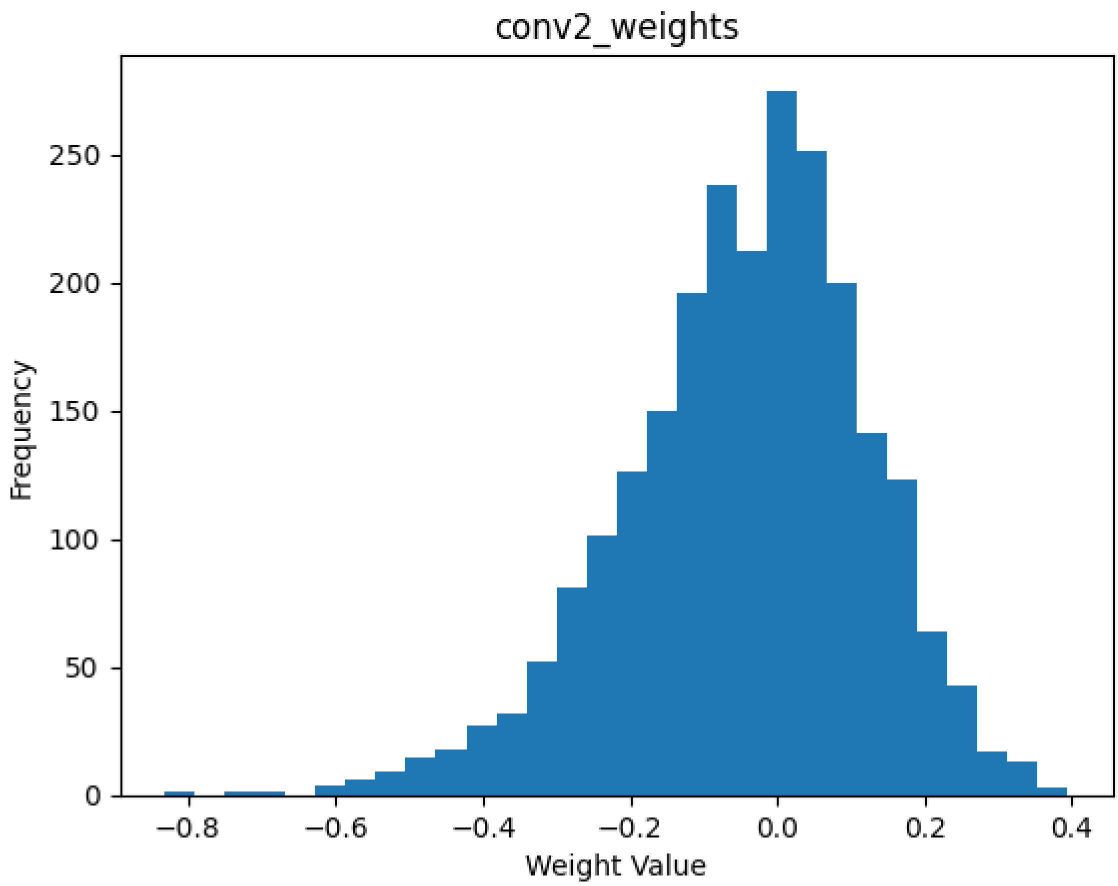
Paper內的Lenet-5架構和本次作業提供的code差異之處在於Activation Function的選擇。
論文中使用sigmoid squashing function，而作業的code使用ReLU function來加速訓練過程和提高效能。

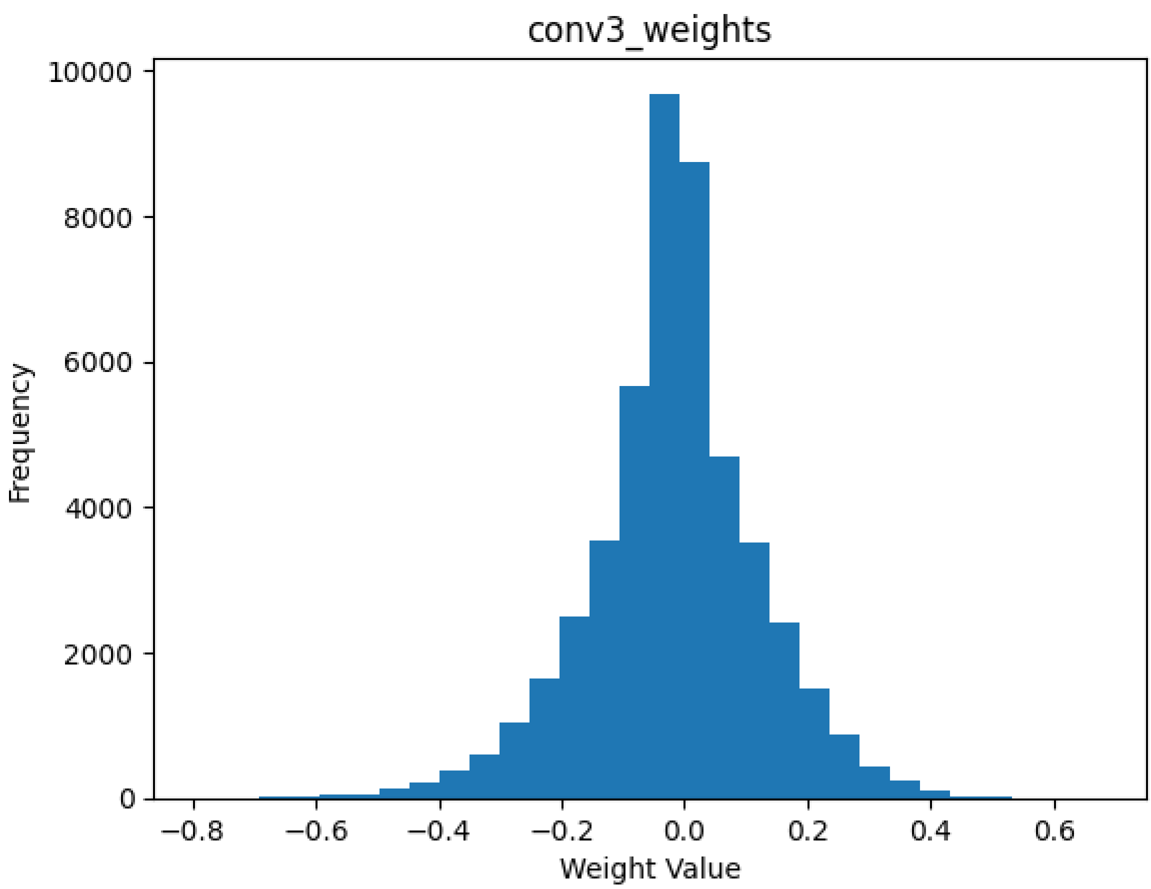
(1.3.4)

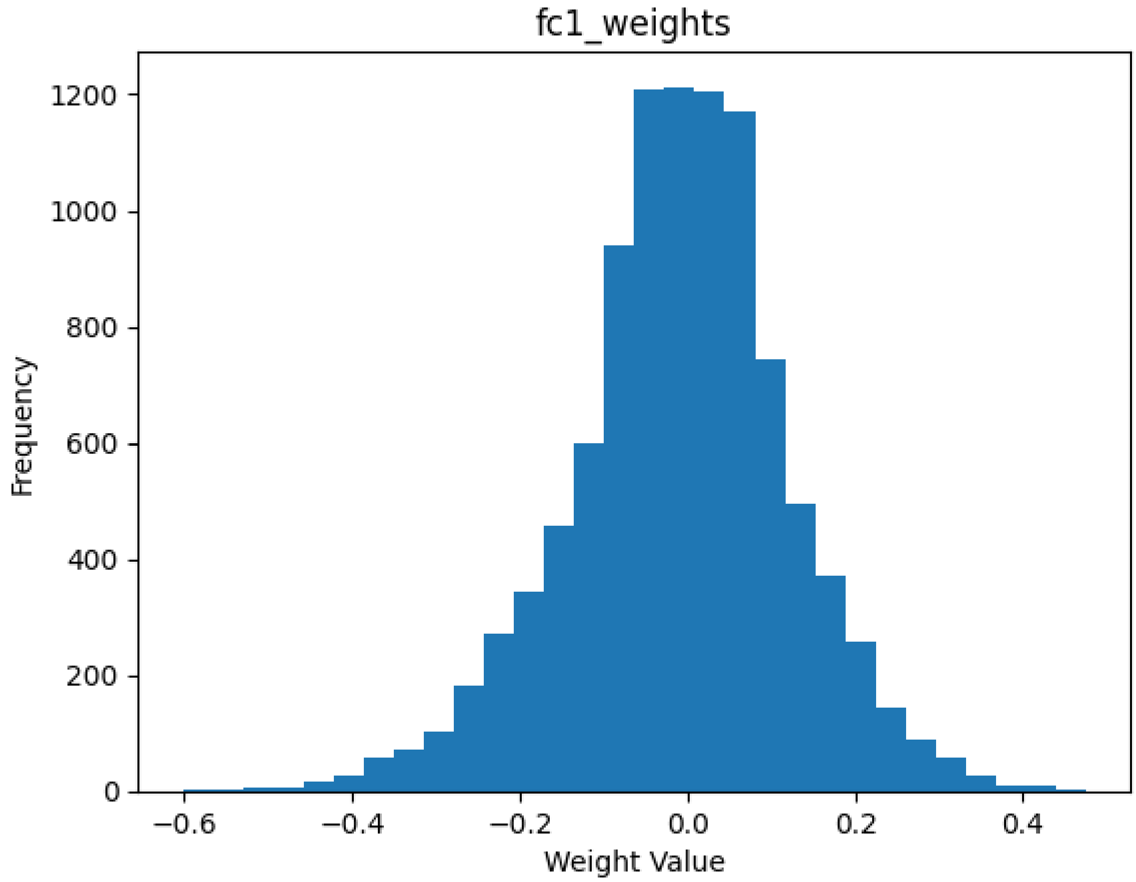
單就模型能否正常運作這點來看，我認為是可以將conv5替換為fully connected layer的，因為卷積層和全連接層本質上都是在做矩陣乘法，所以只要考慮到input/output activation size，替換後是可以正常使用的。然而，全連接層的參數量遠大於卷積層，會使模型效能降低。

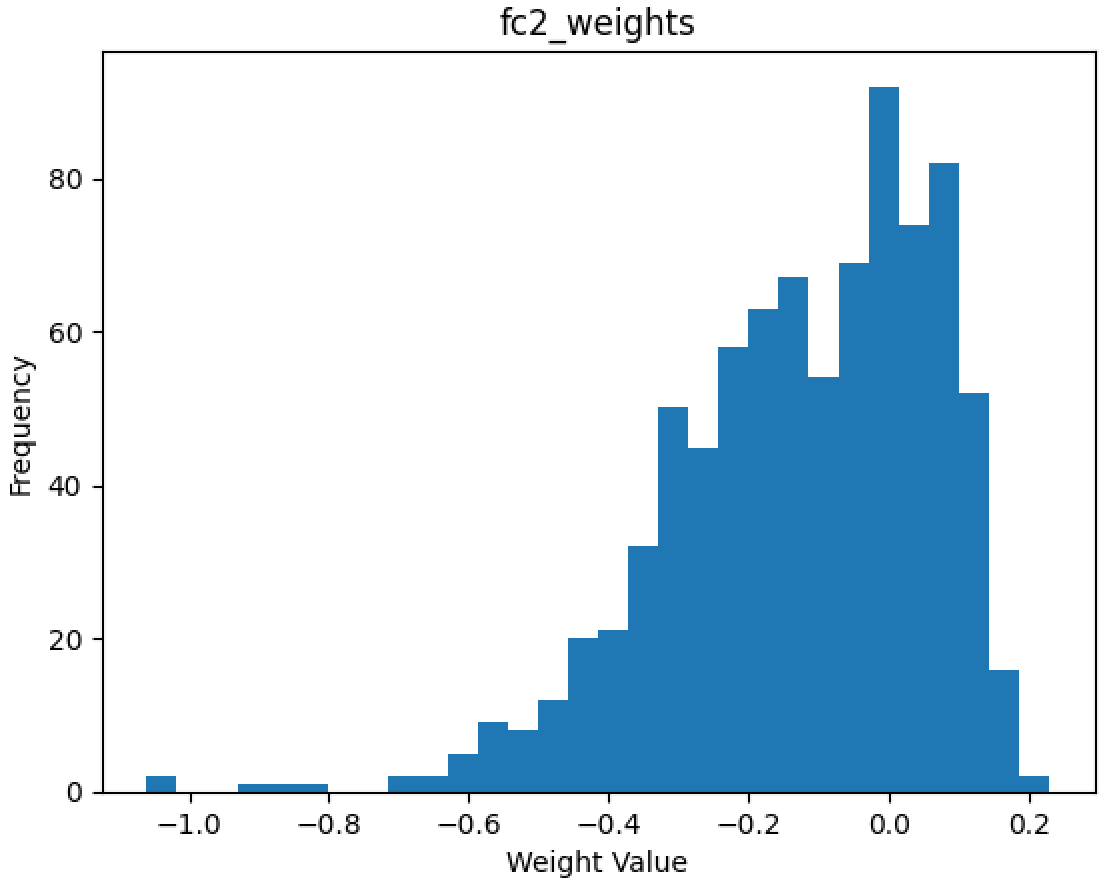
(2.1.1)




conv2_weights
Frequency
Weight Value
0
50
100
150
200
250
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4


conv3_weights
Frequency
Weight Value
0
2000
4000
6000
8000
10000
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


fc1_weights
Frequency
Weight Value
0
200
400
600
800
1000
1200
−0.6
−0.4
−0.2
0.0
0.2
0.4


fc2_weights
Frequency
Weight Value
0
20
40
60
80
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2

(2.1.2)

	Range		3-sigma range		兩Range間大小關係
	Min	Max	μ-3σ	μ+3σ	
Conv1	-0.356631	0.470222	-0.440169	0.459173	B > A
Conv2	-0.831554	0.393874	-0.540313	0.451378	A > B
Conv3	-0.789741	0.677149	-0.428571	0.395550	A > B
Fc1	-0.600528	0.474918	-0.401008	0.381691	A > B
Fc2	-1.060360	0.229388	-0.694463	0.443968	A > B

Note: A = Range, B = 3-sigma range

(2.1.3)

我會傾向於使用 3-sigma range 來 quantize weight，原因為觀察上述 weight 直方圖可發現 weight 基本上是呈現類似常態分布，因此用 3-sigma range 即可以涵蓋絕大多數資料範圍，並且避免了使用 min-max range 可能會有 outlier 導致精度下降的問題。至於 3-sigma range 無法包含的數值，則 map 到欲 quantize 值閾的最大與最小值。

(2.2.1)

scaling factor 是$(max(abs(sigma_range_max), abs(sigma_range_min)))/127$，sigma_range_max 和 sigma_range_min 取 3-sigma range 的兩端點，127 是因為要 map 到 8 bits 有號整數，此處為 symmetric quantization。

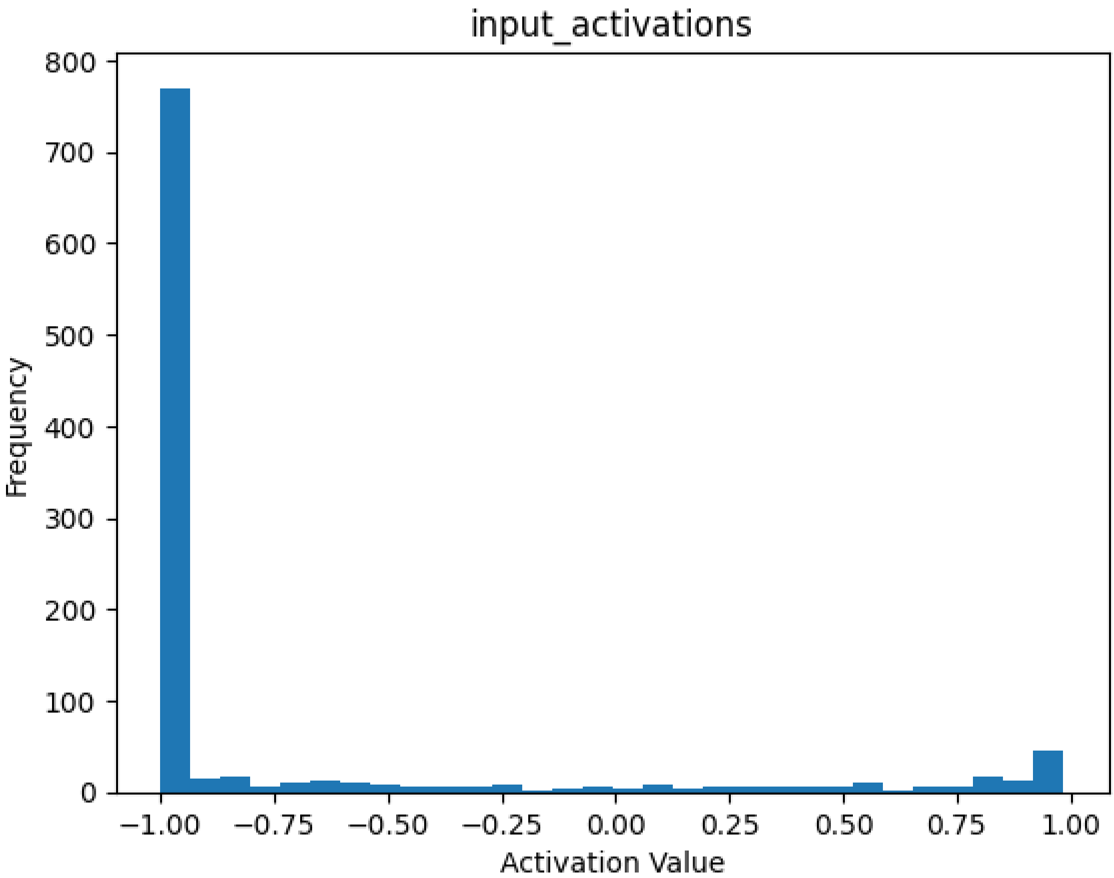
(2.2.2)

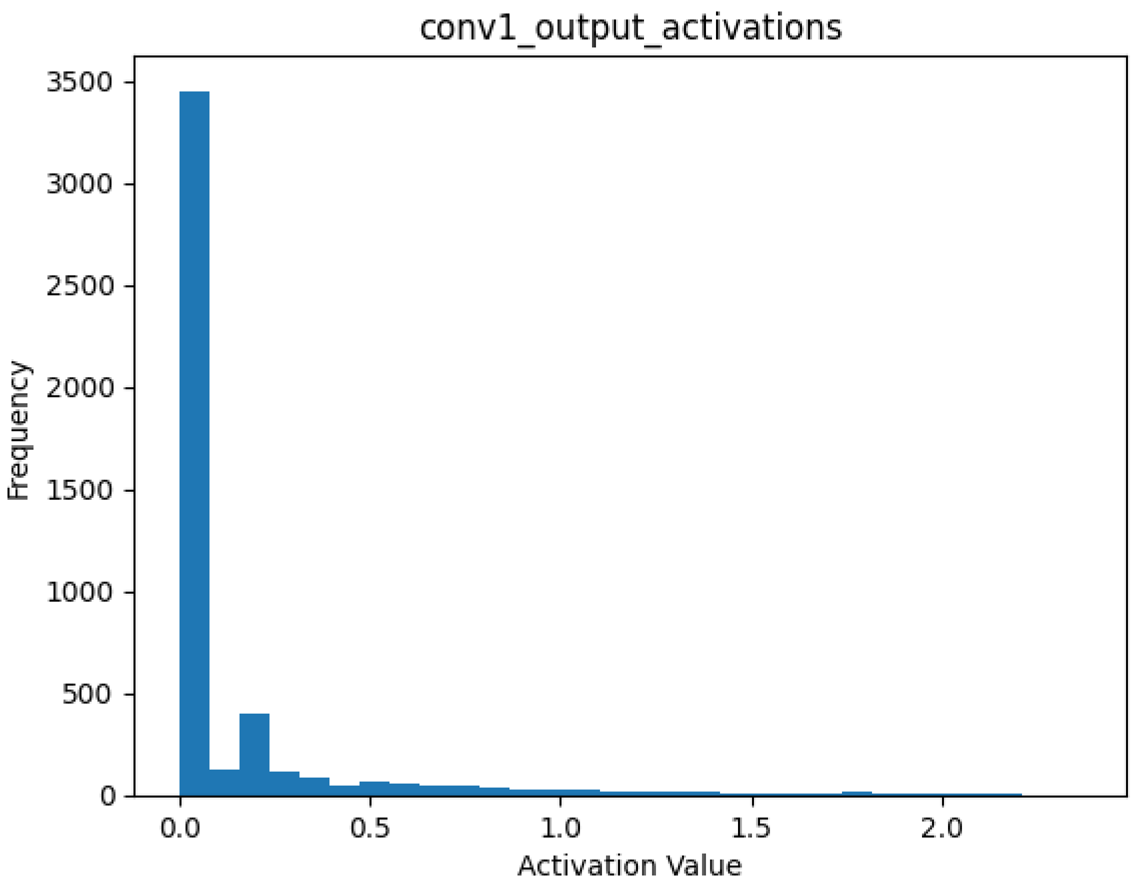
很奇怪的是，做完quantization後準確率反而提升，因此accuracy degradation為 -0.01%

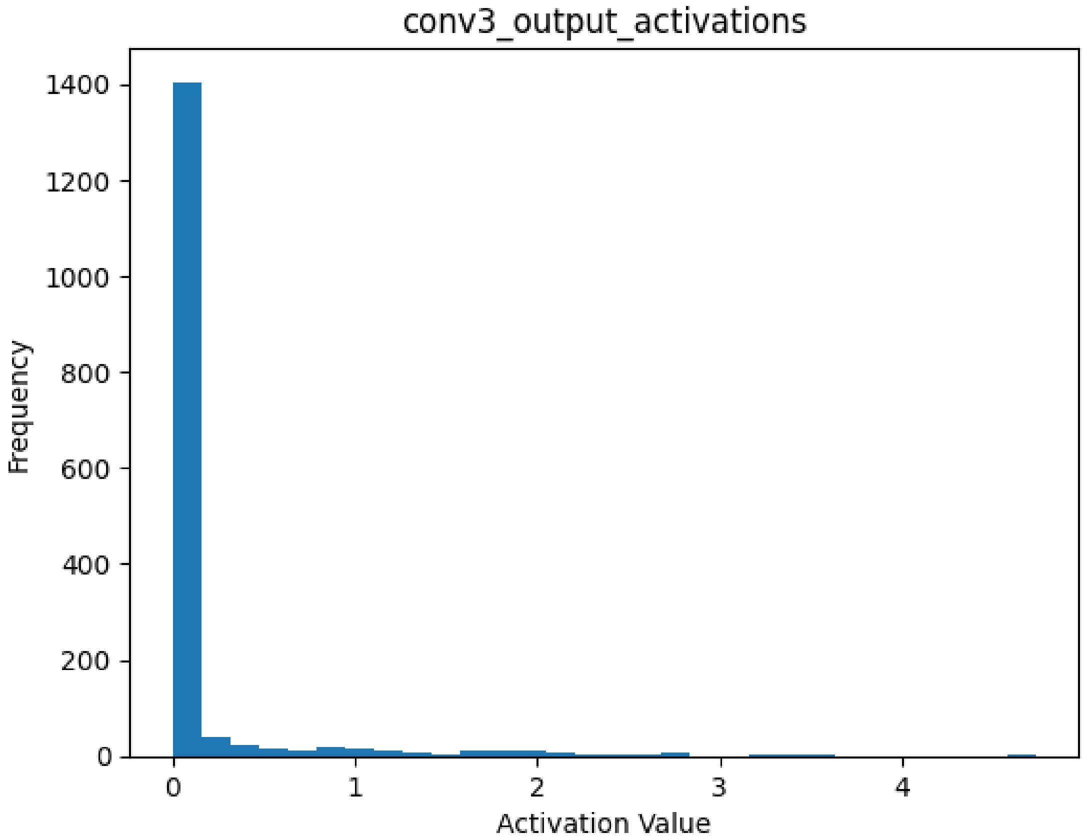
```
Accuracy of the network on the test images: 98.67%
```

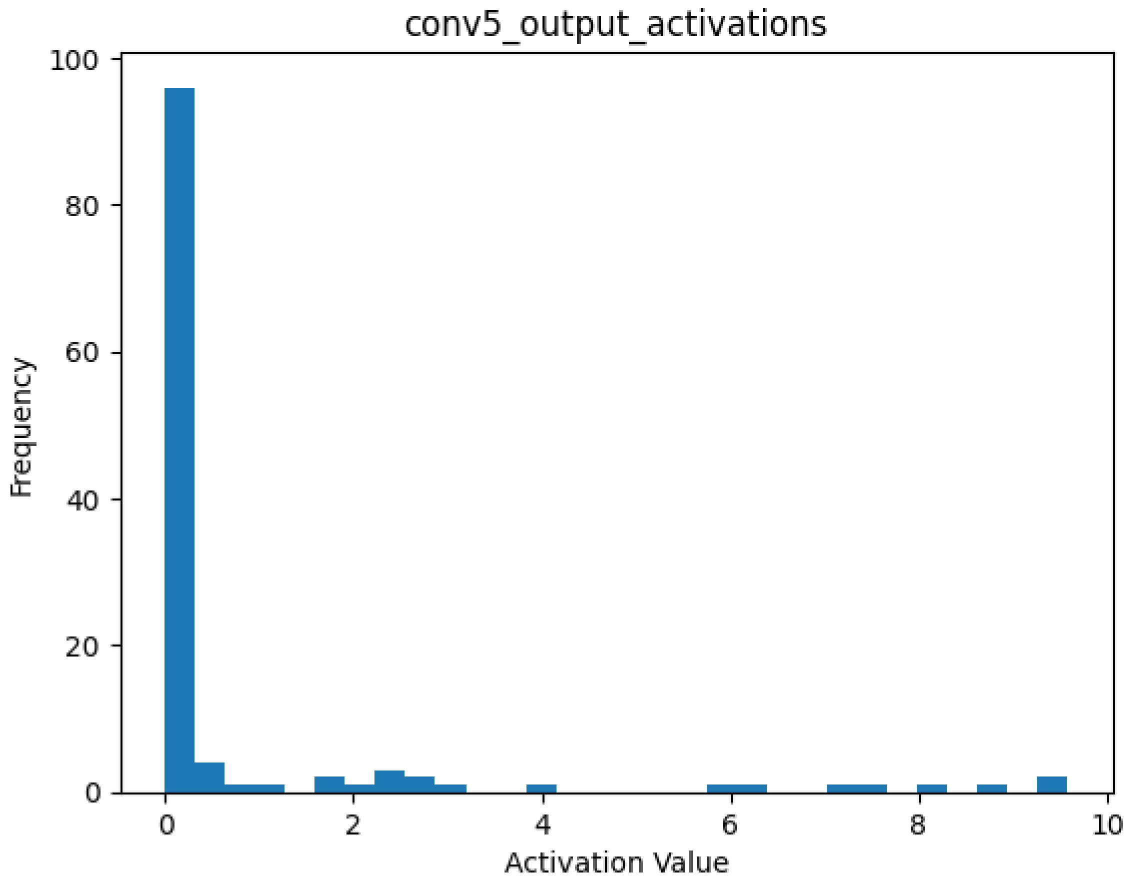
```
Accuracy of the network after quantizing all weights: 98.68%
```

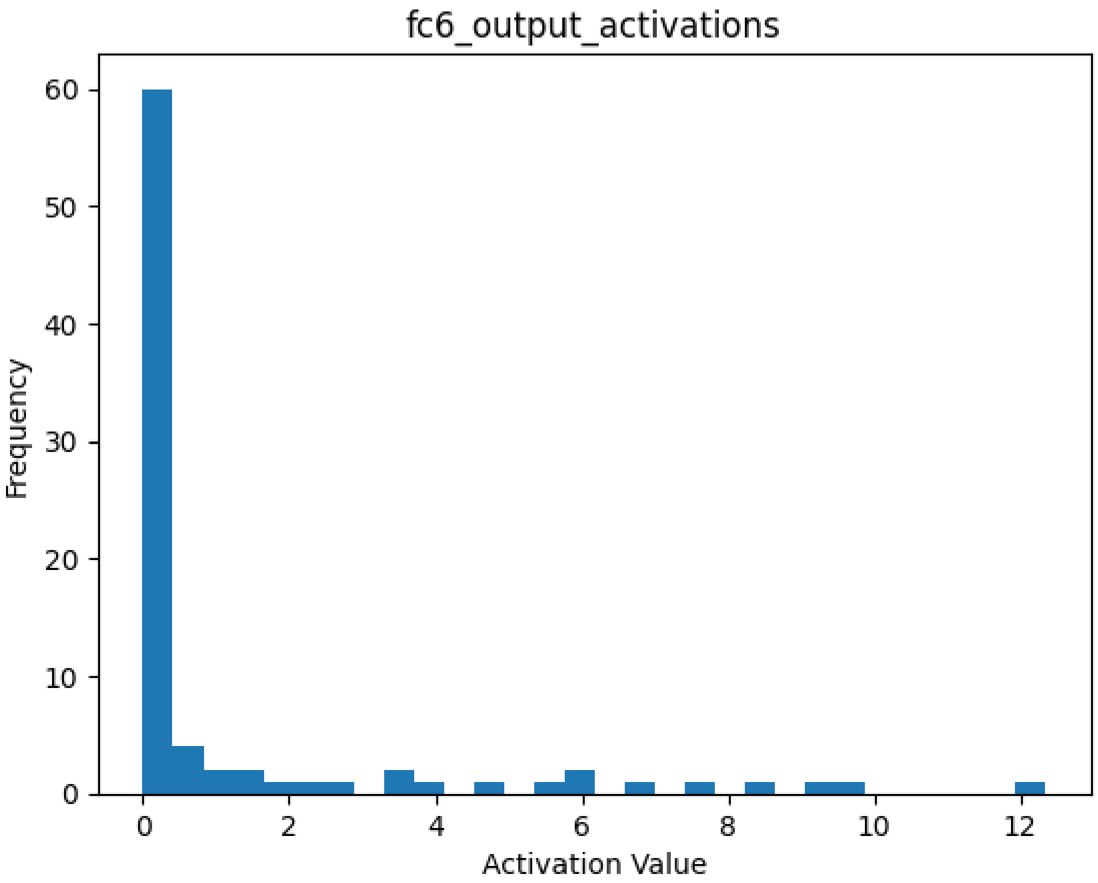
(2.3.1)

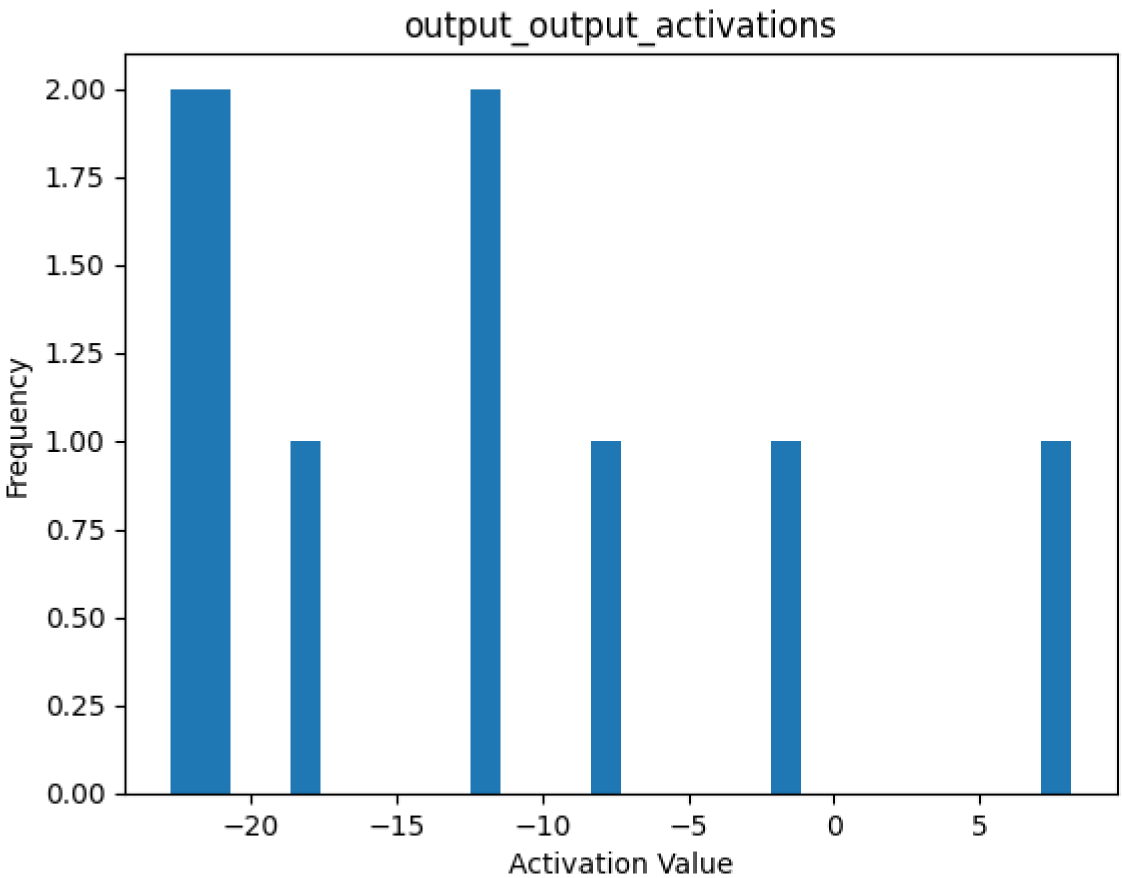

input_activations
Frequency
800
700
600
500
400
300
200
100
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Activation Value


conv1_output_activations
Frequency
3500
3000
2500
2000
1500
1000
500
0
0.0
0.5
1.0
1.5
2.0
Activation Value


conv3_output_activations
Frequency
1400
1200
1000
800
600
400
200
0
0
1
2
3
4
Activation Value


conv5_output_activations
Frequency
100
80
60
40
20
0
0
2
4
6
8
10
Activation Value


fc6_output_activations
Frequency
60
50
40
30
20
10
0
0
2
4
6
8
10
12
Activation Value


output_output_activations
Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−20
−15
−10
−5
0
5
Activation Value

(2.3.2)

	Range		3-sigma range		兩Range間大小關係
	Min	Max	μ-3σ	μ+3σ	
Input	-1.000000	0.984314	-2.483544	1.037082	B > A
Conv1	0.000000	2.368435	-0.857875	1.140249	B > A
Conv3	0.000000	4.735325	-1.236447	1.507193	B > A
Conv5	0.000000	9.576344	-5.380785	6.906925	B > A
Fc6	0.000000	12.349260	-6.600145	9.011153	B > A
Output	-22.772041	8.168824	-42.317790	16.034135	B > A

Note: A = Range, B = 3-sigma range

(2.3.3)

我會使用 min-max range 來 quantize each layer's output activations，原因是 activations 的資料分布非常不平均，資料平均值位於 0，但其餘資料都位於正值。

(2.4.1)

S_I, SW_{conv1}, 與 SO_{conv1} 皆是透過 symmetric quantization 的方式來計算，以S_I為例，具體公式為 Max(abs(Input Activation$_i$)) / 127，除上 127 是因為要映射到 8bits 有號數區間，其餘兩個算法以此類推。

(2.4.2)

浮點數卷積運算 I * W = O，Quantize前後運算式要相等，所以

$$(SI * Iq) * (SWconv1 * Wconv1q) = (SOconv1 * Oconv1q)$$

經過移項後可得，

$$Oconv1q = \frac{SWconv1 * SI}{SOconv1} * (Iq * Wconv1q)$$

$$M1 = \frac{SWconv1 * SI}{SOconv1}$$

(2.4.3)

Conv3 運算同樣可寫為

$$(SIconv3 * Iconv3q) * (SWconv3 * Wconv3q) = (SOconv3 * Oconv3q)$$

對於 Conv3 來說，其 input 為 Conv1 的 Output，因此等式可改寫為

$$(SOconv1 * Oconv1q) * (SWconv3 * Wconv3q) = (SOconv3 * Oconv3q)$$

經過移項後可得，

$$Oconv3q = \frac{SWconv3 * SOconv1}{SOconv3} * (Oconv1q * Wconv3q)$$

(2.4.4)

For Layer n

$$Mn = SWn * SI / SOn, if\ n = 1$$
$$= SWn * SOn - 1 / SOn, else$$

(2.4.5) accuracy degradation: 0.01%

```
Accuracy of the network on the test images: 98.67%
```

```
Accuracy of the network after quantizing both weights and activations: 98.66%
```

(2.4.6)

使用 floor 可以簡化硬體層面的實現難度，假設用 round 則需要另外實作進位的硬體，並且因為統一為捨去，能確保誤差的一致性。

(2.4.7)

```
print("input_scale:\n",  net_quantized.input_scale.item())
print("output_scale:\n  {}\n  {}\n  {}\n  {}\n  {}".format(
        net_quantized.conv1.output_scale.item(),
        net_quantized.conv3.output_scale.item(),
        net_quantized.conv5.output_scale.item(),
        net_quantized.fc6.output_scale.item(),
        net_quantized.output.output_scale.item()
))
```

```
input_scale:
 127.0
output_scale:
 0.0015265502734109766
 0.0021279123611748222
 0.0016686655809974677
 0.0024485469330102205
 0.0029654093086719513
input_scale:
 127.0
```

```
print("input_scale:\n",  net_quantized.input_scale.item())
print("output_scale:\n  {}\n  {}\n  {}\n  {}\n  {}".format(
        round(1/net_quantized.conv1.output_scale.item()),
        round(1/net_quantized.conv3.output_scale.item()),
        round(1/net_quantized.conv5.output_scale.item()),
        round(1/net_quantized.fc6.output_scale.item()),
        round(1/net_quantized.output.output_scale.item())
))
```

```
input_scale:
 127.0
output_scale:
 655
 470
 599
 408
 337
```

round(1/output_scale)所產生的數值為整數，x/round(1/output_scale)為整數除法運算，硬體實現比浮點數乘法運算 x*output_scale 簡單。

(2.5.1)

從題目給的方程式開始推導

$$M \times (W_q \times I_q + \beta_q) = O_q$$

因為是在最後一層(非第一層)加上 bias，M 展開來可寫為

$$\frac{SW \ * \ SOn - 1}{SOn} \times (W_q \times On - 1_q + \beta_q) = O_q$$

移項整理

$$(SW\ * W_q) * (\ SOn-1\ \times On-1_q) + (SW\ *\ SOn-1) * \beta_q = SOn * O_q$$

所以最後一層的 bias 的 scaling factor 應是 $(SW\ *\ SOn-1)$ ，即最後一層 weight 的 scaling factor 乘上倒數第二層的 output scaling factor。

```
Accuracy of the network on the test images after all the weights are quantized but the bias isn't: 98.22%

Accuracy of the network on the test images after all the weights and the bias are quantized: 98.25%
```

(3.1.1)
QAT 因為在模型訓練過程中就加入了 quantization，因此訓練出來的 weight 能比較好的適應 quantization error，故 accuracy 高於直接做 quantization 的 PTQ。

(3.1.2)
Quant 用來將 weight quantization 成低精度的數值(ex. int8，但仍用 float23 儲存)，經過一層卷積運算後透過 dequant 還原成高精度數值(ex. float32)，這是為了反映出 quantization error，並且浮點數的輸出可以藉由 backward propagation 調整 weight，最後找出最佳的 weight。