VLSI System Design HW1

[112061533] [潘金生]

(1.3.1)

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net                                      [4, 10]                   --
├─Sequential: 1-1                        [4, 6, 28, 28]            --
│    └─Conv2d: 2-1                       [4, 6, 28, 28]            150
│    └─ReLU: 2-2                         [4, 6, 28, 28]            --
├─Sequential: 1-2                        [4, 6, 14, 14]            --
│    └─MaxPool2d: 2-3                    [4, 6, 14, 14]            --
├─Sequential: 1-3                        [4, 16, 10, 10]           --
│    └─Conv2d: 2-4                       [4, 16, 10, 10]           2,400
│    └─ReLU: 2-5                         [4, 16, 10, 10]           --
├─Sequential: 1-4                        [4, 16, 5, 5]             --
│    └─MaxPool2d: 2-6                    [4, 16, 5, 5]             --
├─Sequential: 1-5                        [4, 120, 1, 1]            --
│    └─Conv2d: 2-7                       [4, 120, 1, 1]            48,000
│    └─ReLU: 2-8                         [4, 120, 1, 1]            --
├─Sequential: 1-6                        [4, 84]                   --
│    └─Linear: 2-9                       [4, 84]                   10,080
│    └─ReLU: 2-10                        [4, 84]                   --
├─Sequential: 1-7                        [4, 10]                   --
│    └─Linear: 2-11                      [4, 10]                   840
==========================================================================================
Total params: 61,470
Trainable params: 61,470
Non-trainable params: 0
Total mult-adds (M): 1.67
==========================================================================================
Input size (MB): 0.02
Forward/backward pass size (MB): 0.21
Params size (MB): 0.25
Estimated Total Size (MB): 0.47
==========================================================================================
```

(1.3.2)

Type	Input activation size (Channel, width, height)	Output activation size (Channel, width, height)	Activation function
Convolution 1	(1, 32, 32) = 6144	(6, 28, 28) = 4704	ReLu
Maxpooling 2	(6, 28, 28) = 4704	(6, 14, 14) = 1176	
Convolution 3	(6, 14, 14) = 1176	(16, 10, 10) = 1600	ReLu
Maxpooling 4	(16, 10, 10) = 1600	(16, 5, 5) = 400	
Convolution 5	(16, 5, 5) = 400	(120, 1, 1) = 120	ReLu
Fully-connected layer 6	(120, 1, 1) = 120	(84)	ReLu
Fully-connected Output	84	10	

(1.3.3)

本次作業架構把 Lenet-5 中的 subsampling layer 換成 maxpooling layer，並且把所有 activation function 換成 ReLu function，且 param 的部分有些差異，因為我們沒有考慮到 bias 的情況。

(1.3.4)

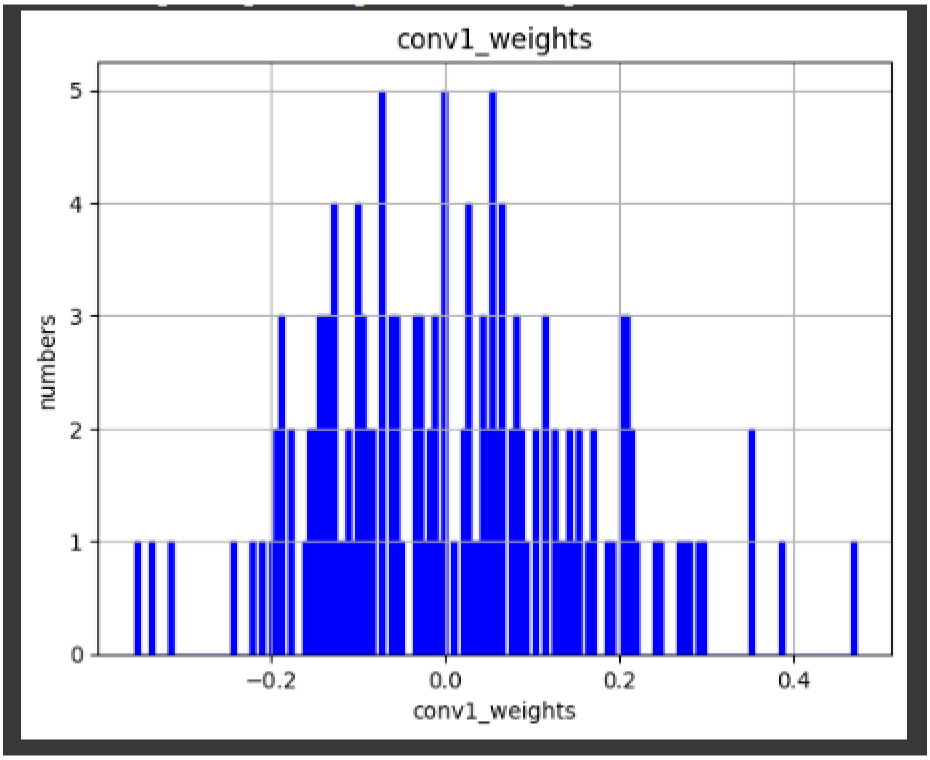
要將 convolution layer 換成 fully connected layer，只要注意向量維度即可，將特徵圖攤平成向量(16*5*5,120)，但這樣可能會遇到一些問題，由於 convolution layer 擁有局部特徵萃取的能力，換成 fully connected layer 會失去局部特徵萃取，再者，這樣會造成參數量大幅增加，提高運算量，造成較大的計算成本，由於參數量增加，也更容易有過擬合狀況產生。

Accuracy:

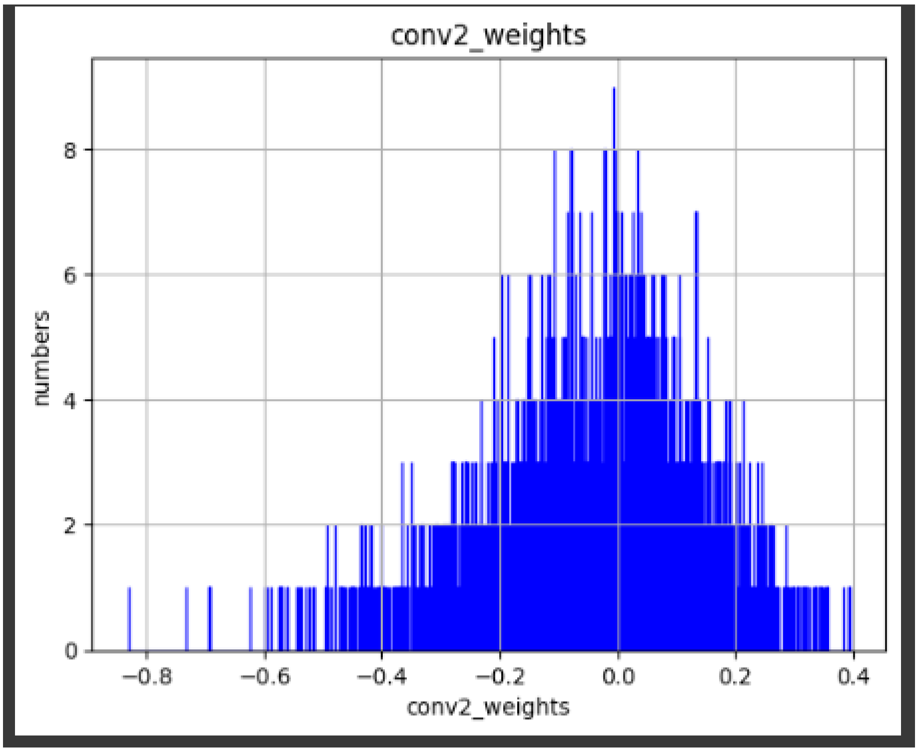
```
[1,  2000] loss: 0.379
[1,  4000] loss: 0.134
[1,  6000] loss: 0.114
[1,  8000] loss: 0.095
[1, 10000] loss: 0.087
[1, 12000] loss: 0.090
[1, 14000] loss: 0.074
98.38
[2,  2000] loss: 0.059
[2,  4000] loss: 0.066
[2,  6000] loss: 0.060
[2,  8000] loss: 0.070
[2, 10000] loss: 0.055
[2, 12000] loss: 0.062
[2, 14000] loss: 0.065
98.67
Finished Training
Accuracy of the network on the test images: 98.67%
```

(2.1.1)

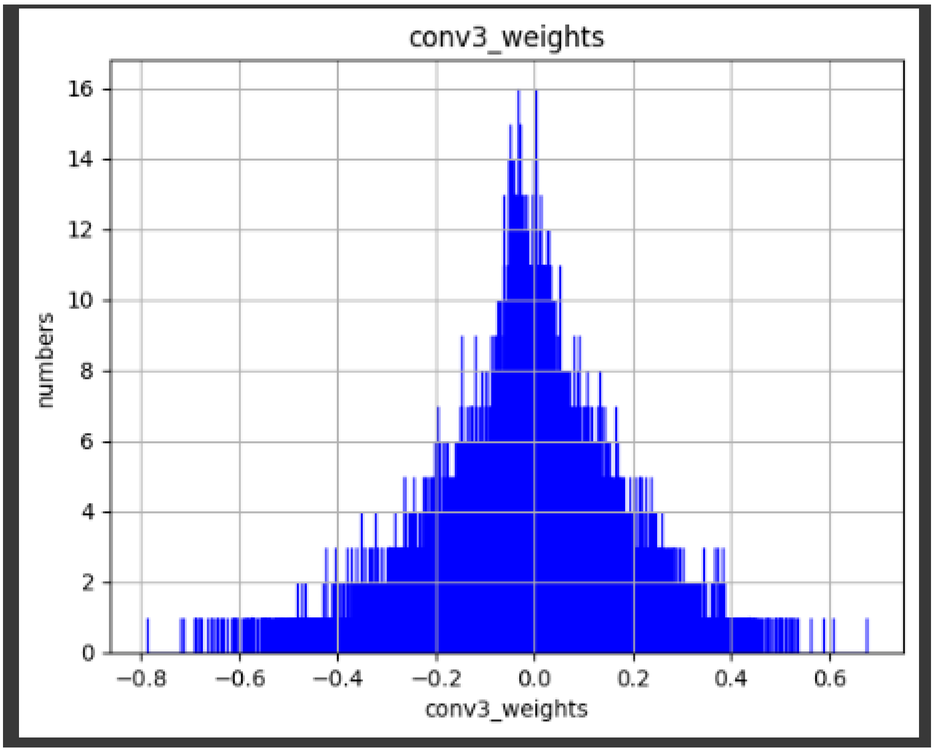
Conv1_weights:



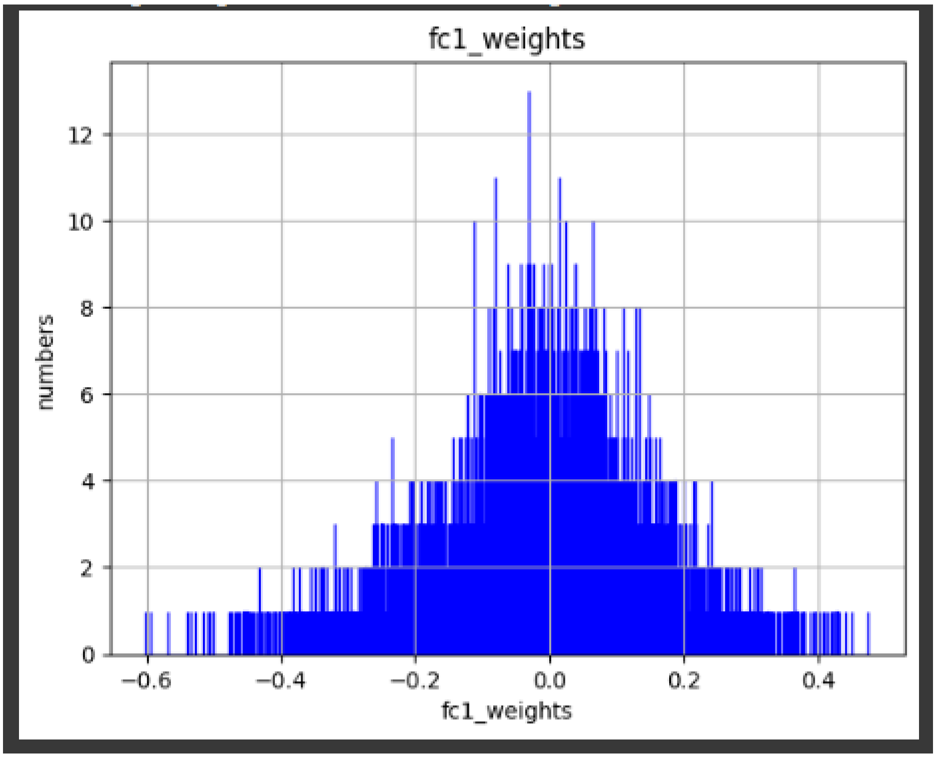
Conv2_weights:



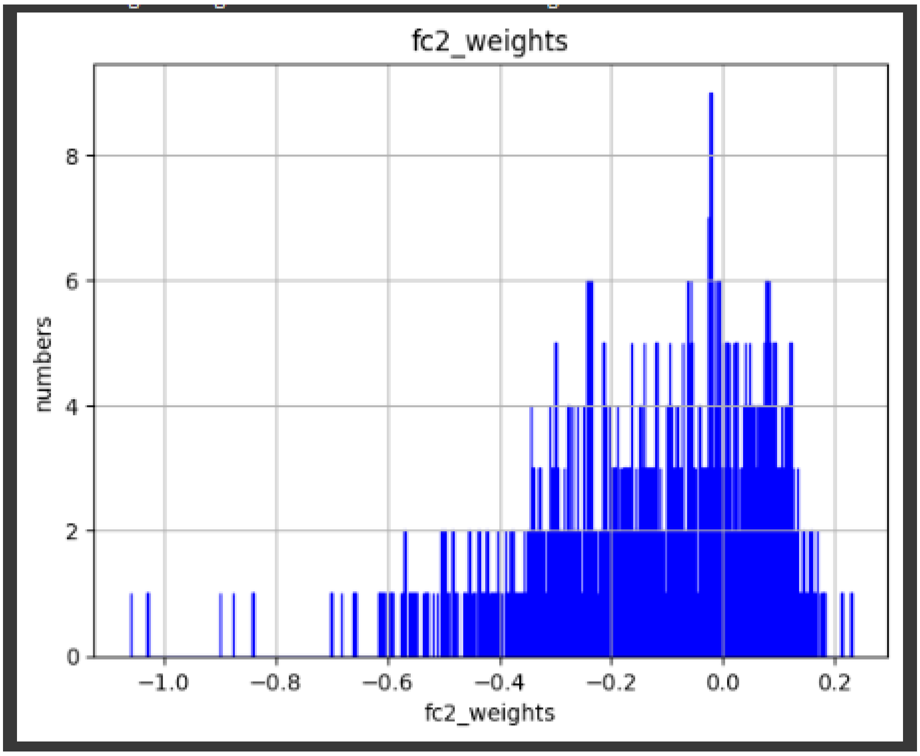
Conv3_weights:



Fc1_weights:



Fc2_weights:



(2.1.2)

Conv1 3-sigma min max range = [-0.4416758120059967, 0.4606795012950897]

Conv1 3-sigma range = [0.9023553133010864]

Conv1 real_value min max range = [-0.35663148760795593, 0.47022220492362976]

Conv1 real_value range = [0.8268536925315857]

Conv1: 3-sigma range is larger than real range

Conv2 3-sigma min max range = [-0.5404166579246521, 0.45148125290870667]

Conv2 3-sigma range = [0.9918979406356812]

Conv2 real_value min max range = [-0.8315535187721252, 0.3938743770122528]

Conv2 real_value range = [1.2254278659820557]

Conv2: 3-sigma range is smaller than real range

Conv3 3-sigma min max range = [-0.4285754859447479 , 0.3955545723438263]

Conv3 3-sigma range = [0.8241300582885742]

Conv3 real_value min max range = [-0.7897406220436096 , 0.6771494150161743]

Conv3 real_value range = [1.4668900966644287]

Conv3: 3-sigma range is smaller than real range

Fc1 3-sigma min max range = [-0.4010276794433594 , 0.3817105293273926]
Fc1 3-sigma range = [0.782738208770752]
Fc1 real_value min max range = [-0.600527822971344 , 0.47491809725761414]
Fc1 real_value range = [1.0754458904266357]
Fc1: 3-sigma range is smaller than real range

Fc2 3-sigma min max range = [-0.6948025822639465 , 0.4443071484565735]
Fc2 3-sigma range = [1.13910973072052]
Fc2 real_value min max range = [-1.0603604316711426 , 0.22938837110996246]
Fc2 real_value range = [1.2897487878799438]
Fc2: 3-sigma range is smaller than real range

(2.1.3)

在 3-sigma range 和真實值的 range 比較中，我會選擇真實值的 range，原因在於它的範圍較廣，較能涵蓋全部的值，且能 clamp 掉極值。

(2.2.1)

Sw 是用來量化權重的 scaling factor，在本次題目中，要實作的是 symmetric quantization，因此我們需要找到|Weight|max，接著題目需要我們 fix-point 在 8-bits，帶入下面公式，即可求得 Sw。

$$Sw = \frac{2^n - 1}{2|Weight|max}\text{，}n = 8$$

(2.2.2)

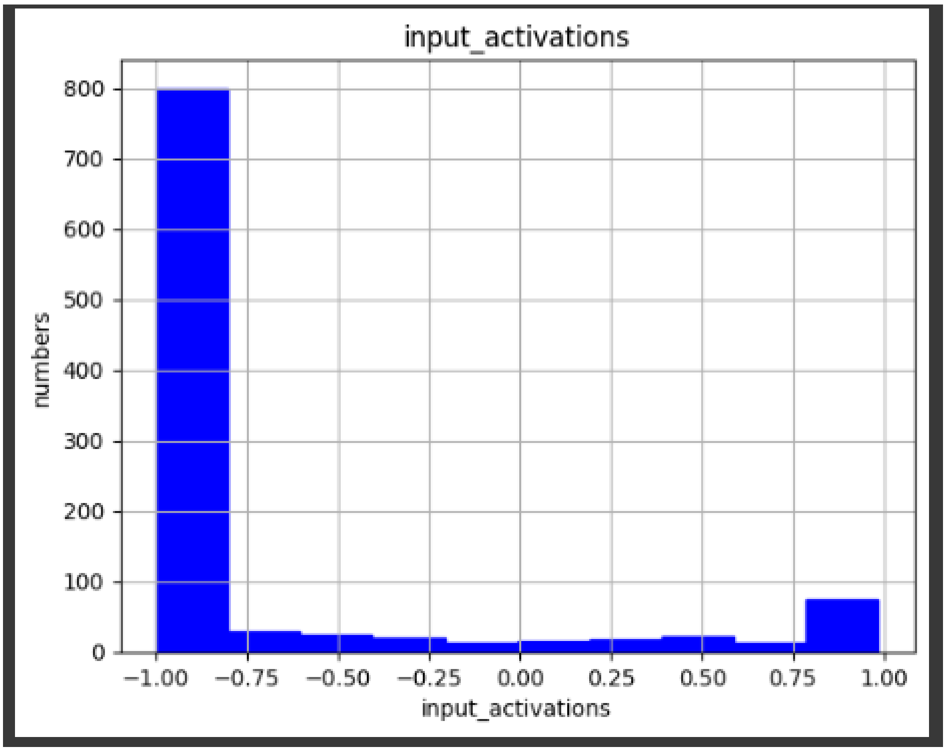
在量化過程中，會從 32-bits floating point 轉成 8-bits fixed point，雖然可以有效的減少運算量和資源的消耗，但會犧牲一點準確度，造成準確度下降

Accuracy:

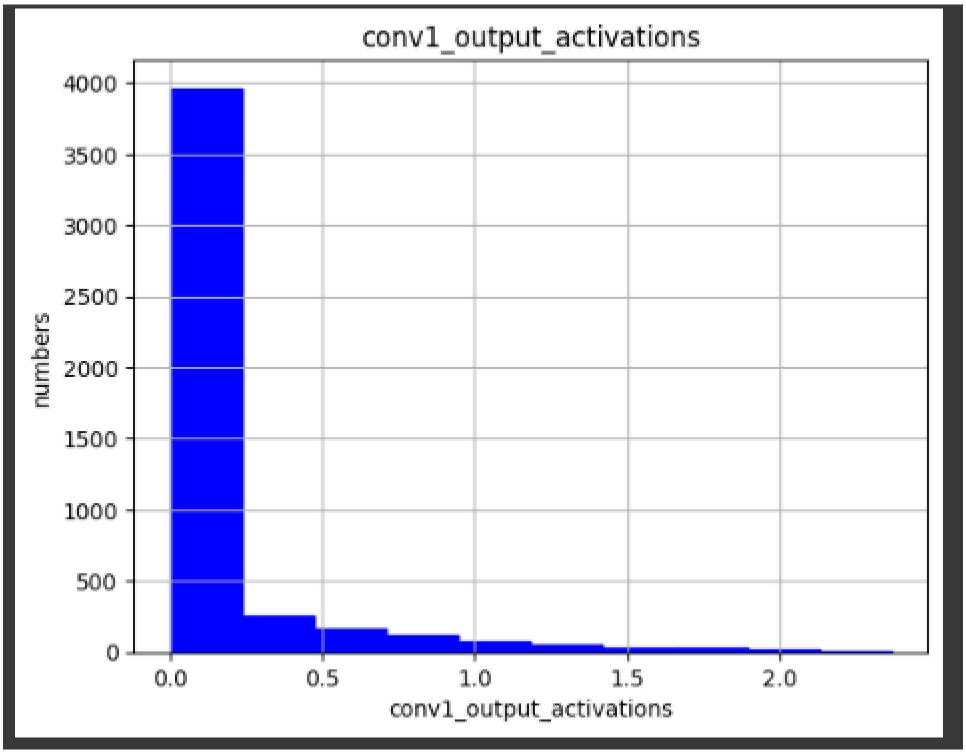
```
Accuracy of the network after quantizing all weights: 98.67%
```

(2.3.1)

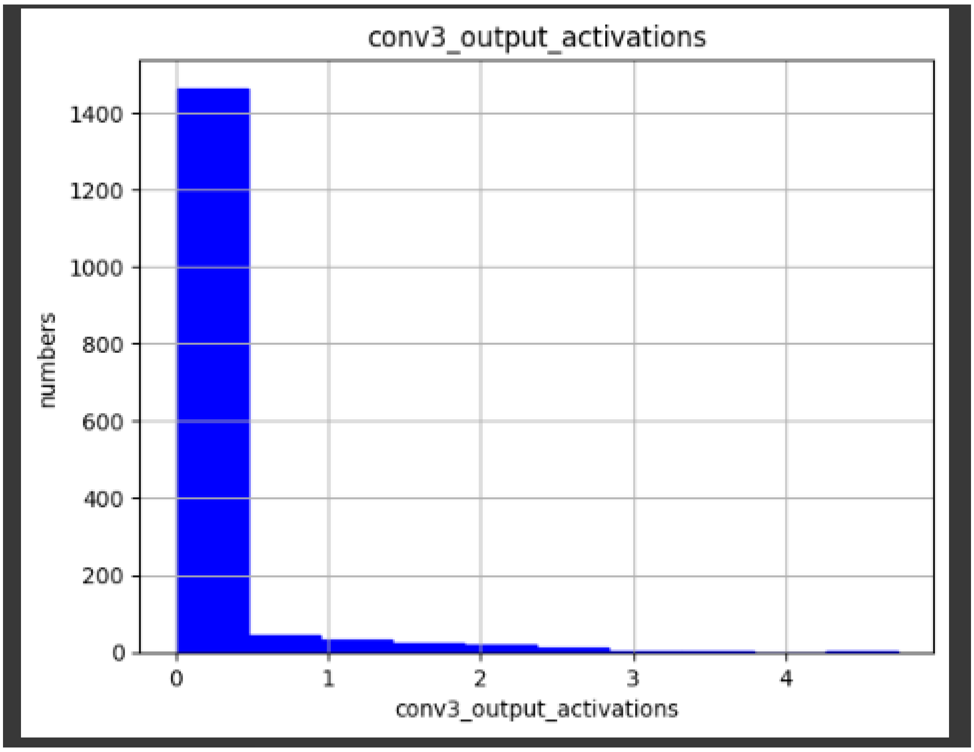
Input activations:



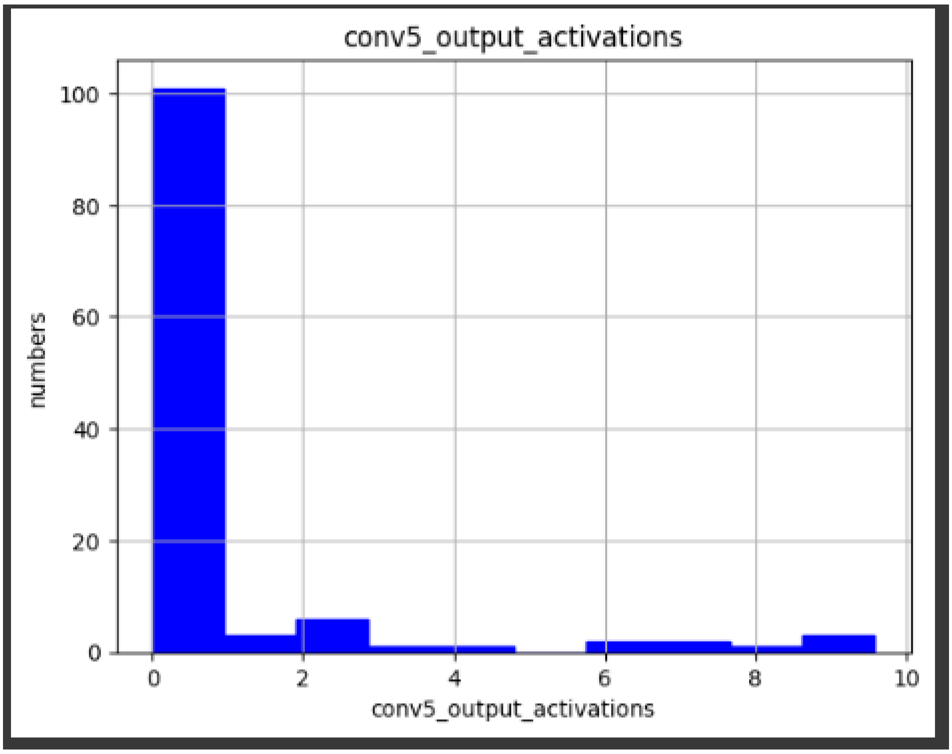
Conv1_output_activations:



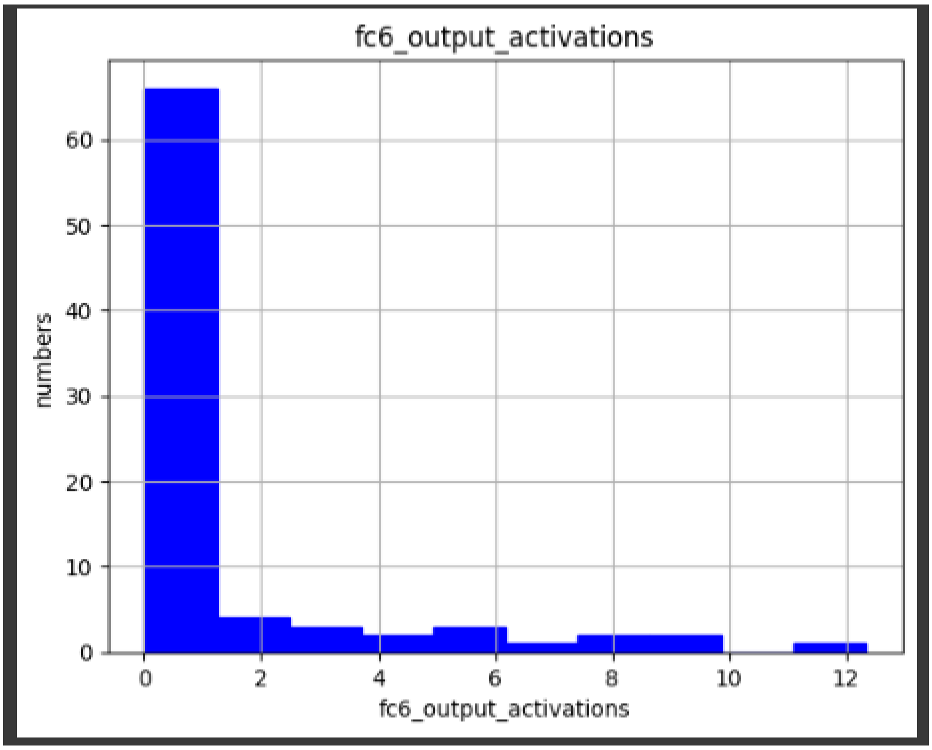
Conv3_output_activations:



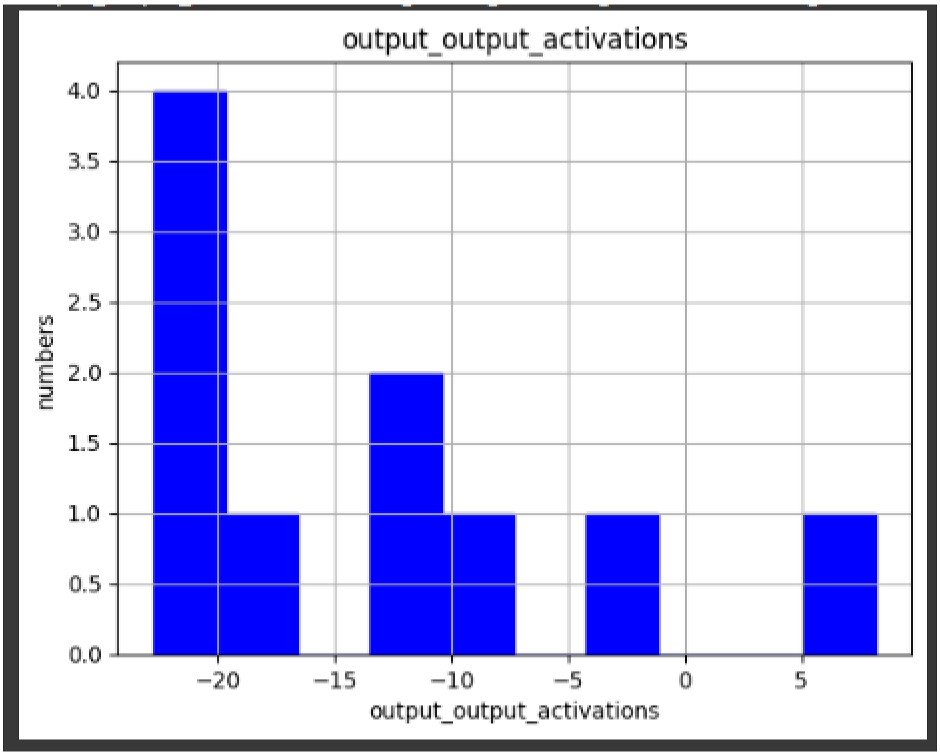
Conv5_output_activations:



Fc6_output_activations:



Output_activations:



(2.3.2)

input_activations 3-sigma range = [-2.484403610229492 , 1.0379424095153809]
input_activations 3-sigma range = [3.522346019744873]
input_activations real_value range = [-1.0 , 0.9843137264251709]
input_activations real_value range = [1.984313726425171]
input_activations: 3-sigma range is larger than real range

conv1_output_activations 3-sigma range = [-0.8579810857772827, 1.1403552293777466]
conv1_output_activations 3-sigma range = [1.9983363151550293]
conv1_output_activations real_value range = [0.0 , 2.3684353828430176]
conv1_output_activations real_value range = [2.3684353828430176]
conv1_output_activations: 3-sigma range is smaller than real range

conv3_output_activations 3-sigma range = [-1.2368762493133545 , 1.5076217651367188]
conv3_output_activations 3-sigma range = [2.7444980144500732]
conv3_output_activations real_value range = [0.0 , 4.735324859619141]
conv3_output_activations real_value range = [4.735324859619141]
conv3_output_activations: 3-sigma range is smaller than real range

conv5_output_activations 3-sigma range = [-5.406546115875244 , 6.932685375213623]
conv5_output_activations 3-sigma range = [12.339231491088867]
conv5_output_activations real_value range = [0.0 , 9.57634449005127]
conv5_output_activations real_value range = [9.57634449005127]
conv5_output_activations: 3-sigma range is larger than real range

fc6_output_activations 3-sigma range = [-6.647026538848877 , 9.058034896850586]
fc6_output_activations 3-sigma range = [15.705060958862305]
fc6_output_activations real_value range = [0.0 , 12.349260330200195]
fc6_output_activations real_value range = [12.349260330200195]
fc6_output_activations: 3-sigma range is larger than real range

output_activations 3-sigma range = [-43.895992279052734 , 17.61233901977539]
output_activations 3-sigma range = [61.508331298828125]
output_activations real_value range = [-22.77204132080078 , 8.168824195861816]
output_activations real_value range = [30.94086456298828]
output_activations: 3-sigma range is larger than real range

(2.3.3)

在 3-sigma range 和真實值的 range 比較中，這裡我會選擇 3-sigma range，原因在於它的範圍較廣，較能涵蓋全部的值，且能 clamp 掉極值。

(2.4.1)

求得 S_I, S_{wconv1}, S_{oconv1} 的值是透過課堂上所教的方法，透過找出 Input, $Weight_{conv1}$, $Output_{conv1}$ 絕對值後的最大值，乘上兩倍後，透過想要求得的 bits 數 (這裡是 8-bits)，透過下列方程式得到 scaling_factor。

$$\mathrm{SI} = \frac{255}{2 * |Input|max}$$

$$\mathrm{Swconv1} = \frac{255}{2 * |Weightconv1|max}$$

$$\mathrm{SI} = \frac{255}{2 * |Outputconv1|max}$$

(2.4.2)

$$\mathrm{Swconv1} * \mathrm{Wconv1q} * \mathrm{SI} * \mathrm{Iq} = Soconv1 * Oconv1q$$

$$\frac{Swconv1 * SI}{Soconv1} * Wconv1q * \ Iq = Oconv1q$$

$$M1 = \frac{Swconv1 * SI}{Soconv1}$$

where Wconv1q is the quantized 8-bit signed integer weight tensor,
Iq is the quantized 8-bit signed integer input activation tensor,
and Oconv1q is the quantized 8-bit signed integer output activation tensor.

(2.4.3)

$$\mathrm{Swconv3} * \mathrm{Wconv3} * \mathrm{Soconv1} * \mathrm{Oconv1} = Soconv3 * Oconv3q$$

$$\mathrm{M3} = \frac{Swconv3 * Soconv1}{Soconv3}$$

(2.4.4)

$$M = \frac{Sw(\text{當前層}) * SI(\text{前一層})}{So(\text{當前層})}$$

Accuracy:

```
Accuracy of the network after quantizing both weights and activations: 98.69%
```

(2.4.6)

使用 floor 函式能夠將小數向下取整到最接近的整數，使量化值始終小於或等於原始值，它的好處是保持量化的一致性，確保他們在某個範圍中，也使數值不會分布偏向較大的大小。

(2.4.7)

能夠有效的減少浮點數的運算，讓數值不會過度縮放

(2.5.1)

$$Sw * Wq * SI * Iq + SB * Bq = So * Oq$$

$$\frac{Sw * SI}{So} * \left(Wq * Iq + \frac{Sb}{Sw * SI} * Bq\right) = Oq$$

$$M = \frac{Sw * SI}{So}$$

Accuracy:

```
[1,  2000] loss: 0.420
[1,  4000] loss: 0.144
[1,  6000] loss: 0.110
[1,  8000] loss: 0.104
[1, 10000] loss: 0.084
[1, 12000] loss: 0.091
[1, 14000] loss: 0.079
98.23
[2,  2000] loss: 0.071
[2,  4000] loss: 0.056
[2,  6000] loss: 0.067
[2,  8000] loss: 0.073
[2, 10000] loss: 0.057
[2, 12000] loss: 0.058
[2, 14000] loss: 0.063
97.77
Finished Training
Accuracy of the network (with a bias) on the test images: 97.77%
```

```
Accuracy of the network on the test images after all the weights are quantized but the bias isn't: 97.81%
```

```
Accuracy of the network on the test images after all the weights and the bias are quantized: 97.77%
```

(3.1.1)

QAT 能夠達到較高的準確率是因為它在訓練期間加入了量化操作，讓量化能夠更適應訓練的過程，模型在訓練時，就已經可以在低精度上進行運算，因此，模型就可以最大程度的減少準確率的損失，而 PTQ 相反，它是在訓練完成後對模型進行量化，模型是在高精度上訓練，因此，在訓練完成後，會導致一定程度的準確率損失，然而，QAT 通常會比 PTQ 來的更耗時且複雜。

(3.1.2)

Quant layer: 用於將浮點數轉換成 fixed point
Dequant Layer: 量化後的整數值轉為浮點數值

Accuracy:

```
[1,  2000] loss: 0.541
[1,  4000] loss: 0.125
[1,  6000] loss: 0.111
[1,  8000] loss: 0.088
[1, 10000] loss: 0.084
[1, 12000] loss: 0.079
[1, 14000] loss: 0.074
98.16
[2,  2000] loss: 0.058
[2,  4000] loss: 0.058
[2,  6000] loss: 0.067
[2,  8000] loss: 0.063
[2, 10000] loss: 0.055
[2, 12000] loss: 0.067
[2, 14000] loss: 0.065
98.08
Finished Training
Accuracy of the MODEL_FP32: 98.09%
Accuracy of the quantized LeNet-5 model on the test images: 98.11%
```

```
Accuracy of the network with fixed point scale: 98.14%
```