VSD HW1

[111061642] [王煒翔]

(1.3.1)

```
==========================================================================================
Layer (type:depth-idx)                   Input Shape               Kernel Shape              Output Shape              Param #
==========================================================================================
Net                                      [1, 1, 32, 32]            --                        [1, 10]                   --
├─Sequential: 1-1                        [1, 1, 32, 32]            --                        [1, 6, 28, 28]            --
│    └─Conv2d: 2-1                       [1, 1, 32, 32]            [5, 5]                    [1, 6, 28, 28]            150
│    └─ReLU: 2-2                         [1, 6, 28, 28]            --                        [1, 6, 28, 28]            --
├─Sequential: 1-2                        [1, 6, 28, 28]            --                        [1, 6, 14, 14]            --
│    └─MaxPool2d: 2-3                    [1, 6, 28, 28]            [2, 2]                    [1, 6, 14, 14]            --
├─Sequential: 1-3                        [1, 6, 14, 14]            --                        [1, 16, 10, 10]           --
│    └─Conv2d: 2-4                       [1, 6, 14, 14]            [5, 5]                    [1, 16, 10, 10]           2,400
│    └─ReLU: 2-5                         [1, 16, 10, 10]           --                        [1, 16, 10, 10]           --
├─Sequential: 1-4                        [1, 16, 10, 10]           --                        [1, 16, 5, 5]             --
│    └─MaxPool2d: 2-6                    [1, 16, 10, 10]           [2, 2]                    [1, 16, 5, 5]             --
├─Sequential: 1-5                        [1, 16, 5, 5]             --                        [1, 120, 1, 1]            --
│    └─Conv2d: 2-7                       [1, 16, 5, 5]             [5, 5]                    [1, 120, 1, 1]            48,000
│    └─ReLU: 2-8                         [1, 120, 1, 1]            --                        [1, 120, 1, 1]            --
├─Sequential: 1-6                        [1, 120]                  --                        [1, 84]                   --
│    └─Linear: 2-9                       [1, 120]                  --                        [1, 84]                   10,080
│    └─ReLU: 2-10                        [1, 84]                   --                        [1, 84]                   --
├─Sequential: 1-7                        [1, 84]                   --                        [1, 10]                   --
│    └─Linear: 2-11                      [1, 84]                   --                        [1, 10]                   840
==========================================================================================
Total params: 61,470
Trainable params: 61,470
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 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)

LeNet-5

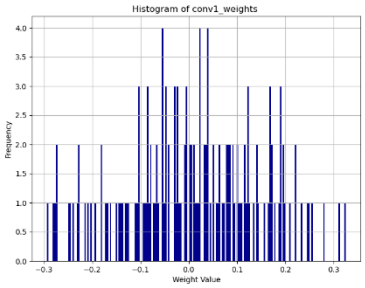
Layer	Input shape	Output shape	Activation function
Conv1	(1,1,32,32)	(1,6,28,28)	tanh
Avgpool2	(1,6,28,28)	(1,6,14,14)	X
Conv3	(1,6,14,14)	(1,16,10,10)	tanh
Avgpool4	(1,16,10,10)	(1,16,5,5)	X
Conv5	(1,16,5,5)	(1,120,1,1)	Sigmoid
Fc6	(1,1,120)	(1,1,84)	Sigmoid
Output	(1,1,84)	(1,1,10)	softmax

(1.3.3)

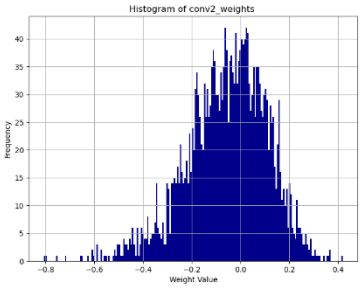
model	Activation function	pooling
本次作業	Convolution+Fully-connected layer : Relu Output : X	Max pooling
Lenet-5	Convolution+Fully-connected layer : Sigmoid Output : softmax	Average pooling

(1.3.4) 可以，不過在經過maxpool4之後，要先拉成向量，所以fc的input變為 400，output改成120，這樣可以發現accuracy會從原本的98.7% -> 97.83，掉了快1%的準確率，不推薦這樣做

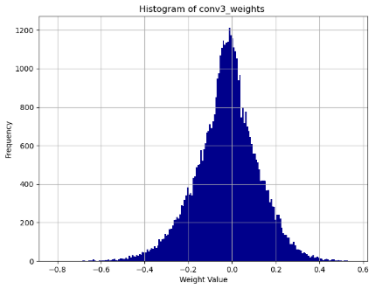
(2.1.1)



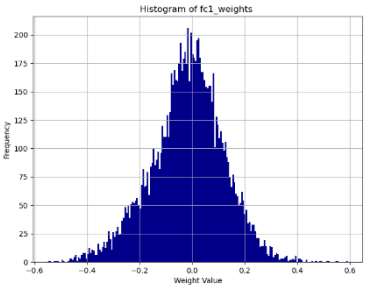
Conv1



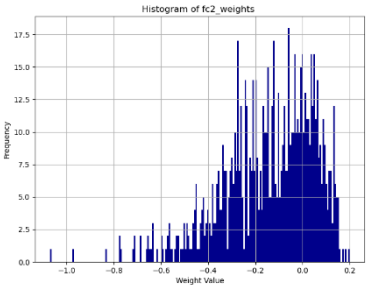
Conv2



Conv3



Fc1



Fc2

(2.1.2)

```
conv1_weights - Actual Range: [-0.2922130227088928 0.32676854729652405], total :0.6189815998077393
              - 3-sigma Range: [-0.4041466573253274 0.4223885675892234], total :0.8265352249145508
              - (3-sigma Range larger than actual range)

conv2_weights - Actual Range: [-0.8035275936126709 0.4228428304195404], total :1.2263704538345337
              - 3-sigma Range: [-0.546493899077177 0.44276853278279305], total :0.9892624318599701
              - (3-sigma Range smaller than actual range)

conv3_weights - Actual Range: [-0.8119214177131653 0.5552406907081604], total :1.3671621084213257
              - 3-sigma Range: [-0.45236088894307613 0.41117371059954166], total :0.8635345995426178
              - (3-sigma Range smaller than actual range)

fc1_weights - Actual Range: [-0.5456373691558838 0.5944367051124573], total :1.1400740146636963
              - 3-sigma Range: [-0.4288624729961157 0.4003728423267603], total :0.829235315322876
              - (3-sigma Range smaller than actual range)

fc2_weights - Actual Range: [-1.0672738552093506 0.20243659615516663], total :1.2697104215621948
              - 3-sigma Range: [-0.722946971654892 0.4459252953529358], total :1.1688722670078278
              - (3-sigma Range smaller than actual range)
```

(2.1.3)

我會選擇3-sigma，因為範圍小於actual range，意味著大部分的資料都在3-sigma range裡面，去除掉一些極端值，對於後面的量化比較好

(2.2.1)

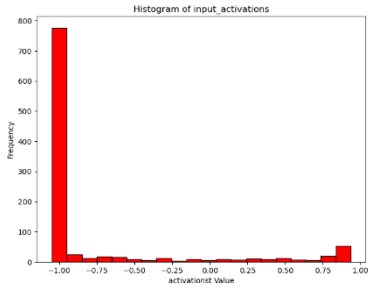
S_w就是該層 weight 的 scaling factor，根據講義 Lec04 第 38 頁可得知

$S_w = \frac{r_{max} - r_{min}}{2^8 - 1}$不過這次的作業 range 屬於 symmetric，所以公式要改成

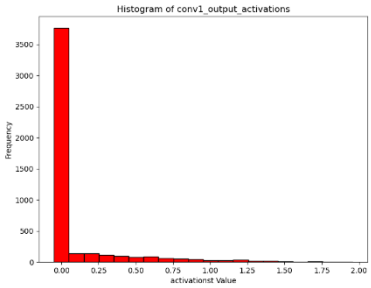
：$S_w = \frac{2 * \max(abs(weights))}{2^8 - 1}$

(2.2.2) 從原本的 98.7% ->98.69%幾乎一模一樣

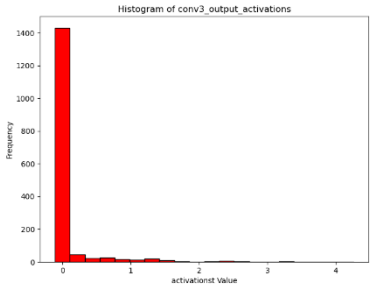
(2.3.1)



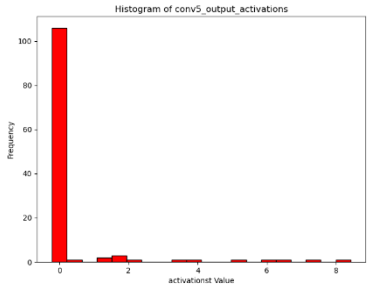
Input



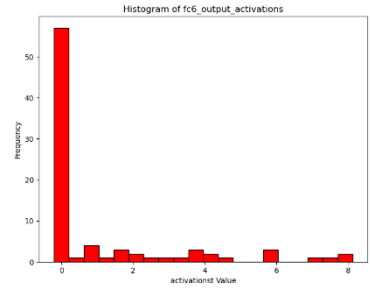
Conv1



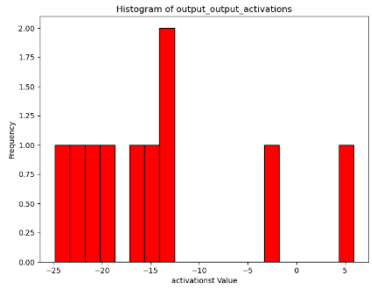
Conv3



Conv5



Fc6



Output

(2.3.2)

```
input_activations - Actual Range: [-1.0 0.9843137264251709], total :1.984313726425171
              - 3-sigma Range: [-2.483543574810028 1.0370821356773376], total :3.5206257104873657
              - (3-sigma Range larger than actual range)

conv1_output_activations - Actual Range: [0.0 2.007612943649292], total :2.007612943649292
              - 3-sigma Range: [-0.7455417737364769 0.9826797023415565], total :1.7282214760780334
              - (3-sigma Range smaller than actual range)

conv3_output_activations - Actual Range: [0.0 4.369547367095947], total :4.369547367095947
              - 3-sigma Range: [-1.0958806201815605 1.3178673461079597], total :2.4137479662895203
              - (3-sigma Range smaller than actual range)

conv5_output_activations - Actual Range: [0.0 8.63629150390625], total :8.63629150390625
              - 3-sigma Range: [-4.058525711297989 4.970582336187363], total :9.029108047485352
              - (3-sigma Range larger than actual range)

fc6_output_activations - Actual Range: [0.0 8.32901382446289], total :8.32901382446289
              - 3-sigma Range: [-5.389976263046265 7.8150036334991455], total :13.20497989654541
              - (3-sigma Range larger than actual range)

output_output_activations - Actual Range: [-24.065505981445312 6.648786544799805], total :30.714292526245117
              - 3-sigma Range: [-39.918185234069824 12.858126640319824], total :52.77631187438965
              - (3-sigma Range larger than actual range)
```

(2.3.3)

我會選擇 actual range，因為大部分的 layer 都小於 sigma range，代表權部的資料都很集中於平均值附近，幾乎不會有極端值的出現

(2.4.1)

$$S_I = \frac{2*\max(abs(pixels))}{2^8-1} \quad S_{W_{conv1}} = \frac{2*\max(abs(W_{conv1}))}{2^8-1} \quad S_{O_{conv1}} = \frac{2*\max(abs(activations_{conv1}))}{2^8-1}$$

(2.4.2)

根據講義的公式，$M = \frac{S_W * S_{IA}}{S_{OA}}$ 由公式即可推得$M_1 = \frac{S_{W_{conv1}} * S_I}{S_{O_{conv1}}}$

就可以得到$O_{q1} = M_1 * (I_q * W_{conv1q})$

(2.4.3)

$$M_3 = \frac{S_{W_{conv3}} * S_{O_{conv1}}}{S_{O_{conv3}}}, \quad O_{q3} = M_3 * (O_{conv1} * W_{conv3q})$$

(2.4.4)

$$M_l = \frac{S_{W_{convl}} * S_{O_{convl-1}}}{S_{O_{convl}}} \qquad s.t \qquad l \geq 1$$

$$M_l = \frac{S_{W_{convl}} * S_{input}}{S_{O_{convl}}} \qquad s.t \qquad l = 0$$

(2.4.6)

使用 floor，跟 round 相比，可以省下硬體的數量，因為 floor 是無條件捨去，不用去做判斷的一些行為，但 round 多出需要判斷是否要進位的電路

(2.4.7)

由於 x*output_scale 屬於浮點數乘法，不過先 round(1/output_scale)把浮點數轉成整數，再用 x 去除它，雖然這樣多做了兩次除法，不過整數除法的硬體數量遠小於浮點數的乘除法，所以好處就是省下硬體面積

(2.5.1)

由於必須使用 2.4 的 M，所以β要直接做量化，量化公式跟 2.4.1 一樣，所以

$$s_\beta = \frac{\max(abs(bias))}{2^8 - 1}$$

(3.1.1)

QAT 可以比後訓練量化（PTQ）實現更高的準確性，這是因為在 QAT 過程中，模型在訓練過程中考慮了量化的影響，並且通過引入 QAT 來最小化量化對模型準確性的影響。相比之下，PTQ 是在訓練完成後對模型進行量化，模型在訓練過程中沒有考慮到量化的影響，因此可能會丟失一些細節信息，導致準確性下降。

(3.1.2)

在使用 PyTorch QAT 方法對模型進行量化後，會出現兩個額外的層，即量化層（quant）和反量化層（dequant）。這兩個層的作用如下：

量化層（quant）：將浮點數表示的模型權重和輸入量化為整數表示。這個過程使得模型的權重和輸入可以以更低的精度表示，從而減少存儲空間和計算成本。
反量化層（dequant）：將量化後的模型輸出反量化為浮點數表示，以便後續的計算或輸出。這個過程將整數表示轉換回原始的浮點數表示，以保留模型的準確性。