AI on Chip 2024

LAB II Quantization
REPORT

Student name: __胡家豪_____
Student ID: __N26122246_____

目錄

一 Question List 3

1. Question. 1 3

2. Question. 2 3

3. Question. 3 3

二.Problem 4

1.Problem 1 4

2.Problem 2 4

三.Paper reading questions 4

四.My opinion 4

一.Question List

1. Question. 1

```python
def get_scale_and_zero_point(fp32_tensor, bitwidth=8):
    q_min, q_max = -2**(bitwidth-1), 2**(bitwidth-1) - 1
    fp_min = fp32_tensor.min().item()
    fp_max = fp32_tensor.max().item()

    ########################################################

    scale = (fp_max-fp_min) / (q_max-q_min)
    zero_point = q_min-fp_min /scale

    ########################################################


    zero_point = round(zero_point)          #round
    zero_point = max(q_min, min(zero_point, q_max)) #clip

    return scale, int(zero_point)
```

2. Question. 2

```python
def linear_quantize(fp32_tensor, bitwidth=8):
    q_min, q_max = -2**(bitwidth-1), 2**(bitwidth-1) - 1

    scale, zero_point = get_scale_and_zero_point(fp32_tensor)

    ########################################################

    q_tensor = torch.round( fp32_tensor/scale ) +zero_point

    ########################################################

    #clamp
    q_tensor = torch.clamp(q_tensor, q_min, q_max)
    return q_tensor, scale, zero_point
```

3. Question. 3

```python
def quantized_linear(input, weights, input_scale, weight_scale, output_scale, input_zero_point, weight_zero_point, output_zero_point, device, bitwidth=8, activation_bitwidth=8):
    input, weights = input.to(device), weights.to(device)

    ################################################

    M = input_scale * weight_scale / output_scale
    output = torch.nn.functional.linear((input - input_zero_point ), (weights - weight_zero_point))
    output *= M
    output += output_zero_point

    ################################################

    #clamp and round
    output = output.round().clamp(-2**(activation_bitwidth-1), 2**(activation_bitwidth-1)-1)

    return output
```

Restart Visual Studio Code to apply the latest update.

二. Problem

1. Problem 1 - What is the size of the model after int8 quantization if its original size is 50MB?

$$50\text{MB} \times \frac{8\text{bits}}{32\text{bits}} = 12.5\text{MB}$$

2. Problem 2- If M = 0.2, determine values for M0 and n such that the equation on page 11 is true

$$M0 = 0.8、\text{n} = 2$$

三. Paper reading questions

軟體：由於在推論時參數已經固定，所以可以將 batch norm 的參數 fold 進 linear layer，如此一來可增加效率。具體的計算是利用下列公式將權重折疊

$$w_{\text{fold}} := \frac{\gamma w}{\sqrt{\sigma_B^2 + \varepsilon}}.$$

另外，作者也提到，一般的 fuse layer 中，他會選擇將 bias 使用 32 uint 進行運算。這樣做的原因是因為 32uint 可以降低模型的偏差，所以不會捨棄 bias 而且還使用 32bits。但是如此一來，如果要使用 int8 的話，必須將其在進入 activation layer 之前再次轉換成 uint8。

硬體：在硬體上，量化的式子因為可以寫成

$$q_3^{(i,k)} = Z_3 + M \sum_{j=1}^{N} (q_1^{(i,j)} - Z_1)(q_2^{(j,k)} - Z_2)$$

這條式子只有 M 的部分是浮點數(而且通常小於 1 大於 0)，所以我們可以將其拆解成

$$M = 2^{-n} M_0$$

這部分我覺得論文說明有些奇怪，但是我覺得他想表達的是將 M 使用整數(M0)做乘上 2 的負冪次，如此一來，上面的式子除了 2 的負冪次都為整數，而 2 的負冪次則使用位移進行運算。如此可以均使用整數的乘法器。

四. My opinion

本次作業讓我更加理解量化的過程：分成不須額外進行訓練的 Post-training quantization (PTQ)與利用訓練尋求更佳效果的 Qunatiztion-aware training (QAT)。

透過量化，可以降低硬體的功率與面積(因為不需要浮點運算)。但是在理解量化的過程還是令人有些吃力。因為量化不只分為上述兩種，而且還有針對不同 layer 而要進行不同的量化方式，而每一種量化方式又各有不同。
不過本次作業給的範例 code 算是清晰明瞭，只要好好研究還是可以一窺量化的奧妙。所以即使流程繁瑣，還是能夠從中學到不少東西。