Lab2 Reference Solution

Quantize weights only

Question 2.1

```
def quantized_weights(weights: torch.Tensor)
```

$$scale = \frac{max(abs(weights))}{127}$$

$$result = clamp(round(\frac{weights}{scale}), -128, 127)$$

Quantize activations

Let's look at the expression given in the problem statement. The product of the first two terms on the left side of the equal sign is the result of calculations using weights and activations that have been quantized to int8. The resulting integer value represented with 8 bits might overflow. Additionally, since the output of this layer serves as the input activation for the next layer, an additional quantization is required. Therefore, the left side of the equation needs to be divided by the scale of the output once more. It's important to note that the numerators of the fractions are all floating-point values, while the overall fraction represents the quantized integer value.

$$\frac{W}{s_W} * \frac{In}{s_{In}} * \frac{1}{s_{Out}} = \frac{Out}{s_W s_{In} s_{Out}}$$

Question 4.1

Notes: It's important to note that after eliminating the denominators from both sides of the equation, the resulting equation should remain equal to the original equation given in the problem. Otherwise, you will receive half score.

Solution:

(a)

$$\frac{W_{conv1}}{s_{W_{conv1}}} * \frac{In}{s_{In}} * \frac{1}{s_{Out_{conv1}}} = \frac{Out_{conv1}}{s_{W_{conv1}} s_{In} s_{Out_{conv1}}} \tag{1.1}$$

Incorrect Answer Example:

$$\frac{W_{conv1}}{s_{W_{conv1}}} * \frac{In}{s_{In}} * s_{Out_{conv1}} = Out_{conv1}$$

(b)

$$W_{conv2} * In_{conv2} = Out_{conv2} \tag{2.1}$$

$$W_{conv2} * Out_{conv1} = Out_{conv2} \tag{2.2}$$

$$W_{conv2} * (W_{conv1} * In) = Out_{conv2} \tag{2.3}$$

$$\frac{W_{conv2}}{s_{W_{conv2}}} * (\frac{W_{conv1}}{s_{W_{conv1}}} * \frac{In}{s_{In}} * \frac{1}{s_{Out_{conv1}}}) * \frac{1}{s_{Out_{conv2}}} = \frac{Out_{conv2}}{s_{W_{conv2}} s_{W_{conv1}} s_{In} s_{Out_{conv1}} s_{Out_{conv2}}} \tag{2.4}$$

Question 4.2

```
def quantize_initial_input(pixels: np.ndarray)
```

$$scale = \frac{max(abs(pixels))}{127}$$

```
def quantize_activations(activations: np.ndarray, s_w: float, s_initial_input:
float, ss: List[Tuple[float, float]])
```

Note: Some of us experience significant precision loss when quantizing activations, the reason being the use of only activations to calculate the scale. Let's look at the right side of Equation 2.4.

Here, we use $s_{Out_{conv2}}$ not just to quantize Out_{conv2}, but rather to quantize the entire term:

$$\frac{Out_{conv2}}{s_{W_{conv2}} s_{W_{conv1}} s_{In} s_{Out_{conv1}}}$$

Now, we will match the variables in the formula one-to-one with the input variables in the quantization function. `activations` represents the floating-point output of the current layer, corresponding to the Out_{conv2} in the formula. `s_w` represents the scale of the current layer's weight, corresponding to the $s_{W_{conv2}}$ in the formula. `s_initial_input` represents the snetwork input scale, corresponding to s_{In} in the formula. Additionally, if we derive the output scale formulas for subsequent layers of the network, we can find that calculating the output scale of a certain layer always involves the product of the scaling factors of the weights and outputs of all previous network layers. This corresponds to the `ss` list in the function.

$$scale = \frac{max(abs(activations / s_w / s_{initial_{input}} / \prod_{i=0}^{len(ss)}(ss[i][0] * ss[i][1])))}{127}$$

Question 4.3

After the computation of each network layer is completed, divide by the corresponding scale, and then perform round and clamp operations.

For example:

```
x = x / self.input_scale
x = torch.round(x)
x = torch.clamp(x, -128, 127)

x = self.conv1(x)
x = x / self.conv1.output_scale
x = torch.round(x)
x = torch.clamp(x, -128, 127)
x = F.relu(x)
```

Quantize Biases

Question 5.1

$$(\frac{W}{s_W} * \frac{In}{s_{In}} + \frac{\beta}{s_W s_{In}}) * \frac{1}{s_{Out}} = \frac{Out}{s_W s_{In} s_{Out}}$$

Question 5.3

The quantization method for bias is exactly the same as that for activation because they share all the scales.

$$scale = \frac{max(abs(bias / s_w / s_{initial_{input}} / \prod_{i=0}^{len(ss)}(ss[i][0] * ss[i][1])))}{127}$$