Quantization for TVM

Ziheng Jiang
TVM Conference, Dec 12th 2018
Quantization for TVM

What is Quantization?

Converting weight value to low-bit integer like 8bit precision from float-point without significant accuracy drop.

source: Han et al
Quantization for TVM

Gain Compression & Acceleration:
- Less storage space
- Faster arithmetic operation
- Friendly to accelerator and ultra low-power embedded devices
Quantization for TVM

Choice Spaces for Quantization

- number of bit
  - 4bit, 8bit, 16bit
- quantization scheme:
  - symmetric, asymmetric, etc.
- hardware constraint:
  - e.g. prefer integer shift instead of float multiplication

Goal

Instead of proposing “the only right way to achieve quantization in TVM”, we would like to build a quantization workflow which can be customized flexibly.
SimQ simulates the rounding error and saturating error during quantizing. Its argument will get tuned during `calibrate`.

\[
SimQ(nbit, range, sign) = \frac{\text{Clip}(\text{Round}(\frac{x}{r} \times 2^{nbit-sign})) \times r}{2^{nbit-sign}}
\]
Quantization for TVM

Code Sample

```python
# user can override the annotate function
@register_annotate_function("nn.conv2d", override=True)
def annotate_conv2d(ref_call, new_args, ctx):
    lhs, rhs = new_args
    lhs = attach_simulated_quantize(lhs, sign=False, rounding='round')
    rhs = attach_simulated_quantize(rhs, sign=False, rounding='stochastic_round')
    return expr.Call(ref_call.op, [lhs, rhs], ref_call.attrs)

# assuming we have an existed mxnet model, convert it to relay graph
graph, params = relay.frontend.from_mxnet(mxnet_model)

# quantize the relay graph with all kinds of configure
with qconfig(nbit_dict={QFieldKind.ACTIVATION: 24}, global_scale=8.0, skip_k_conv=1):
    qgraph, qparams = quantize(graph, params)

# ...build and deploy it locally or remotely with tvm
```

Code Sample
## Quantization for TVM

### Demonstration with 8bit Symmetric Quantization

<table>
<thead>
<tr>
<th>Global Scale</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>64.1%</td>
</tr>
<tr>
<td>4.0</td>
<td>68.1%</td>
</tr>
<tr>
<td>8.0</td>
<td>69.5%</td>
</tr>
<tr>
<td>16.0</td>
<td>69.6%</td>
</tr>
</tbody>
</table>

Accuracy Drop with ResNet18 (original 70.8%)

<table>
<thead>
<tr>
<th>Time/ms</th>
<th>Cortex A53</th>
<th>VTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet18</td>
<td>307.09</td>
<td>64.87</td>
</tr>
<tr>
<td>MobileNet</td>
<td>131.14</td>
<td>51.96</td>
</tr>
</tbody>
</table>

End to End Performance