Lecture: Variable Precision

- Topics: approaches to support variable precision (Stripes, Bit-Pragmatic, Laconic, BitFusion), compression during training, pruning
Wasted Work

Four types of unnecessary work in DNN execution:

• Data re-fetch: when the same value is fetched repeatedly from storage structures; can be addressed with reuse/tiling

• High data width: when the bits exceed the information content; addressed with quantization/approximation

• Sparsity: computations involving zeros; already seen pruning, Cnvlutin, SCNN

• Temporal/spatial redundancy: value similarity in time/space; can be addressed with early diffs
Low Precision

High potential for low-res and variable low-res computations

<table>
<thead>
<tr>
<th>Network</th>
<th>Per Layer Neuron Precision in Bits</th>
<th>100% Ideal Speedup</th>
<th>Per Layer Neuron Precision in Bits</th>
<th>99% Ideal Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>3-3</td>
<td>5.33</td>
<td>2-3</td>
<td>7.33</td>
</tr>
<tr>
<td>ConvNet</td>
<td>4-8-8</td>
<td>2.89</td>
<td>4-5-7</td>
<td>3.53</td>
</tr>
<tr>
<td>AlexNet</td>
<td>9-8-5-5-7</td>
<td>2.38</td>
<td>9-7-4-5-7</td>
<td>2.58</td>
</tr>
<tr>
<td>NiN</td>
<td>8-8-8-9-7-8-8-9-9-8-8-8</td>
<td>1.91</td>
<td>8-8-7-9-7-8-8-9-9-8-7-8</td>
<td>1.93</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>10-8-10-9-8-10-9-8-9-10-7</td>
<td>1.76</td>
<td>10-8-9-8-8-9-10-8-9-10-8</td>
<td>1.80</td>
</tr>
<tr>
<td>VGG_M</td>
<td>7-7-7-8-7</td>
<td>2.23</td>
<td>6-8-7-7-7</td>
<td>2.34</td>
</tr>
<tr>
<td>VGG_S</td>
<td>7-8-9-7-9</td>
<td>2.04</td>
<td>7-8-9-7-9</td>
<td>2.04</td>
</tr>
<tr>
<td>VGG_19</td>
<td>12-12-12-12-12-10-11-11-13-12-13-13-13-13-13</td>
<td>1.35</td>
<td>9-9-9-8-12-10-10-12-13-13-11-12-13-13-13-13-13</td>
<td>1.57</td>
</tr>
</tbody>
</table>

TABLE I: Per Convolutional layer neuron precision profiles needed to maintain the same accuracy as in the baseline (100%) and to reduce it within 1% of the baseline (99%). Ideal: Potential speedup with Stripes over a 16-bit baseline.
Stripes

- Philosophy: spread the computation over several cycles; perform one bit at a time; compensate with parallel multiplications of other activations that contribute to the same neuron; computation ends early for low-res values

- Increases area by 32% (more weights, more shift registers, more accumulators)

- Reduces energy by 57%, improves performance by 1.92x on average (fewer bits moving through the memory hierarchy and through compute)
Bit Pragmatic

- Starts with a Stripes-like design, then exploits the fact that 0’s in the activation are not doing anything effectual; by skipping those cycles, the multiplication finishes even sooner
- Requires a pre-processing step where the input value is encoded to only keep the 1 bits (results in operand expansion \( \rightarrow \) more energy, area, complexity)
- The position indicates the shift amount before adding
Effectuals in Activations

<table>
<thead>
<tr>
<th></th>
<th>Alexnet</th>
<th>NiN</th>
<th>Google</th>
<th>VGGM</th>
<th>VGGS</th>
<th>VGG19</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>16-bit Fixed-Point</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>7.8%</td>
<td>10.4%</td>
<td>6.4%</td>
<td>5.1%</td>
<td>5.7%</td>
<td>12.7%</td>
</tr>
<tr>
<td>NZ</td>
<td>18.1%</td>
<td>22.1%</td>
<td>19.0%</td>
<td>16.5%</td>
<td>16.7%</td>
<td>24.2%</td>
</tr>
<tr>
<td><strong>8-bit Quantized</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>31.4%</td>
<td>27.1%</td>
<td>26.8%</td>
<td>38.4%</td>
<td>34.3%</td>
<td>16.5%</td>
</tr>
<tr>
<td>NZ</td>
<td>44.3%</td>
<td>37.4%</td>
<td>42.6%</td>
<td>47.4%</td>
<td>46.0%</td>
<td>29.1%</td>
</tr>
</tbody>
</table>

**TABLE I**

Average fraction of non-zero bits per neuron for two fixed-length representations: 16-bit fixed-point, and 8-bit quantized. **All**: over all neurons. **NZ**: over non-zero neurons only.

They employ a few techniques that lower performance, but also reduce area (the size of the shifters/accum). Note that the buffers have to supply data at a higher rate.
Laconic

- An 8b multiplication is like adding 64 single bits, most are 0
- Not very area-effective for 8b baseline, but effective for 16b
BitFusion

• Not trying to exploit dynamic opportunities (like Bit-Pragmatic and Laconic), but exploiting narrow ops (like Stripes)

• Designs hardware that is more compact than bit-serial architectures like Stripes

• Organizes an 8bx8b multiplier (a Fusion Unit) as 16 2bx2b multipliers (Bit Bricks); smaller precision computations are then mapped to a set of Bit Bricks
Example

(a) A 4-bit multiplication
\[ 6_{10} \times 11_{10} = 66_{10} \]

(b) Decomposing the 4-bit multiplication to four 2-bit multiplications.
Fusion Unit

(a) Fusion Unit with 16 BitBricks

(b) 16x Parallelism, Binary (1-bit) or Ternary (2-bit)

(c) 4x Parallelism, Mixed-Bitwidth (2-bit weights, 8-bit inputs)

(d) No Parallelism, 8-bits
Comparison to Stripes

Key difference is that Stripes needs large shifters and accumulators for each “bit brick”, while BitFusion does not.

Fig. 8: Temporal design. Operands $a−h$ are 2-bit.

Fig. 9: Spatial fusion. Operands $a−h$ are 2-bit.
GIST

- Competing approach to vDNN
- While activations are consumed at full precision in forward pass, they are stored in compressed format for the backward pass; compression can be lossy or lossless
- ReLu followed by Conv: Sparse encoding (lossless)
- ReLu followed by pooling: Binarize (lossless)
- Others: Low (8b) precision storage (lossy)

Figure 3: Breakdown of memory within stashed feature maps. ReLU layer consumes a major portion of the footprint.
Compressed vDNN

- Observes high sparsity in activations
- Data is compressed before the offload
- Average compression ratio of 2.6x and improves vDNN performance by 32%
Pruning

• Outlier aware pruning: 97% of weights can use 4-bit precision, while 3% of weights need full precision

• Dark side of pruning: prior work only focuses on output, but many DNNs also produce a confidence that is fed to other downstream algorithms (Viterbi); pruning can mess up the confidence even if it doesn’t affect the top-1 or top-5 classification
References

- “Stripes: Bit Serial Deep Neural Network Computing”, P. Judd et al., MICRO 2016
- “Bit Pragmatic Deep Neural Network Computing”, J. Albericio et al., MICRO 2017
- “Laconic Deep Learning Inference Acceleration”, S. Sharify et al., ISCA 2019
- “Bit Fusion: Bit-level Dynamically Composable Architecture for Accelerating Deep Neural Networks”, H. Sharma et al., ISCA 2018
- “GIST: Efficient Data Encoding for Deep Neural Network Training”, A. Jain et al., ISCA 2018
- “Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks”, M. Rhu et al., HPCA 2018
- “The Dark Side of DNN Pruning”, R. Yazdani et al., ISCA 2018
- “Energy Efficient Neural Network Accelerator based on Outlier-Aware Low-Precision Computation”, E. Park et al., ISCA 2018