BitX: Empower Versatile Inference with Hardware Runtime Pruning

Hongyan Li\textsuperscript{12}, Hang Lu\textsuperscript{12}, Jiawen Huang\textsuperscript{1}, Wenxu Wang\textsuperscript{12}, Mingzhe Zhang\textsuperscript{1}, Wei Chen\textsuperscript{1}, Liang Chang\textsuperscript{3}, Xiaowei Li\textsuperscript{12}

\textsuperscript{1}State Key Laboratory of Computer Architecture, Institute of Computing Technology, CAS, Beijing, China
\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China
\textsuperscript{3}University of Electronic Science and Technology of China, Chengdu, China
Introduction

DNN networks

<table>
<thead>
<tr>
<th>Model</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet50</td>
<td>$22.5\times10^6$</td>
</tr>
<tr>
<td>3D-ConvNet</td>
<td>$79\times10^6$</td>
</tr>
<tr>
<td>Bert_large</td>
<td>$340\times10^6$</td>
</tr>
</tbody>
</table>

Pruning

Demand of large computation

2~3 days sparse training

3~4 days retraining / fine-tuning

YoloV3

network slimming

6 TITAN Xp
2 Intel Xeon E5-2650 v4

Introduction

The contribution of this work

Propose a novel hardware runtime pruning method -- BitX, to empower versatile DNN inference

① Software effortless
No retraining! No fine-tuning!

② Orthogonal to the existing software pruning methodologies
Obtain additional speedup

③ Multi-precision support
Floating point & fixed point DNNs
Introduction

The contribution of this work

Propose a deep learning accelerator capable of unprecedented hardware runtime pruning to mine the maximum potential of BitX.

Speedup

2.61x~4.82x (fp32), 2.00x (16 fixed point), 4.98x and 14.76x higher over the baseline based on software pruning

Accuracy

Negligible accuracy under fp32, about 1% accuracy improvement under 16-bit fixed point

Accelerator Performance

2.00x and 3.79x performance improvement compared with other SOTA accelerators

*BitX* is designed for *flexible* and *versatile* DNN inference for the most tasks
## Motivation

### Weight sparsity versus bit sparsity

<table>
<thead>
<tr>
<th>Model</th>
<th>Weight Sparsity</th>
<th>Bit Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet121</td>
<td>4.84%</td>
<td>48.64%</td>
</tr>
<tr>
<td>ResNet50</td>
<td>0.33%</td>
<td>48.64%</td>
</tr>
<tr>
<td>ResNet152</td>
<td>0.75%</td>
<td>48.64%</td>
</tr>
<tr>
<td>ResNext50_32x4d</td>
<td>0.37%</td>
<td>48.64%</td>
</tr>
<tr>
<td>ResNext101_32x8d</td>
<td>3.43%</td>
<td>48.65%</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>0.05%</td>
<td>48.64%</td>
</tr>
<tr>
<td>MNASNet0.5</td>
<td>0.00%</td>
<td>48.60%</td>
</tr>
<tr>
<td>MNASNet1.0</td>
<td>8.07%</td>
<td>48.98%</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>0.01%</td>
<td>48.67%</td>
</tr>
<tr>
<td>ShuffleNetV2_x0_5</td>
<td>0.00%</td>
<td>48.36%</td>
</tr>
<tr>
<td>ShuffleNetV2_x1_0</td>
<td>1.53%</td>
<td>48.63%</td>
</tr>
<tr>
<td>SqueezeNet1_0</td>
<td>0.05%</td>
<td>48.64%</td>
</tr>
<tr>
<td>SqueezeNet1_1</td>
<td>0.02%</td>
<td>48.64%</td>
</tr>
</tbody>
</table>

**Weight sparsity**: the values below $10^{-5}$ over the total parameter size

**Bit sparsity**: total bit 0s over the total “bit count” of the mantissas

- **Very limited headroom**
- **Significantly abundant**
Motivation

Trivial bits

X-axis: the bit slice of the binary represented weight (in fp32)
Y-axis: the fraction of bit ‘1’
All the evaluated DNNs exhibit an “arched” shape.

Distribution analysis of bit 1s

The central bit slices own most of the bit 1s (~40%). While all these bits are tiny. Taking bit significance $2^{-21} \sim 2^{-30}$ as the representative, the equivalent decimals are in range: $0.0000000477$ ($\sim10^{-8}$) to $0.000000000931$ ($\sim10^{-11}$)
Problem tackled in this work

Goal: pinpoint the essential bits and prune away the useless bits

Genetic 0 bits
Zero-skipping mechanism to avoid the ineffectual computations caused by the zero bits -- Easy to implement ^_^

Trivial 1 bits
The impact of a single bit to the whole network is not that easy to be determined -- How to tackle this problem? -_-!

We intend to solve the two issues in BitX!
Methodology – core concept

Weights represented in floating-point 32 mode.
Different colors indicate the bit significance from $2^{-1}$ to $2^{-9}$ after the binary point.
Methodology – bit pruning

Approximating Matrix Multiplication

- Given an \( m \times n \) matrix \( A \) and an \( n \times p \) matrix \( B \),
- The product \( AB \), is equivalent to the sum of \( n \) rank-one matrices

\[
AB = \sum_{i=1}^{n} (A(i)) (B(i))
\]

- \( A(i) \) the \( i \)-th column of \( A \)
- \( B(i) \) the \( i \)-th row of \( B \)
- Each term in the summation is a rank-one matrix

Metric of selecting rank-one matrices

\[
p_i = \frac{|A(i)| |W(i)|}{\sum_{i'=1}^{l} |A(i')||W(i')|}
\]

\[
= \frac{|A(i)| \sqrt{\sum_{j=1}^{n} \left(2^{E_i^j} \times v_j\right)^2}}{\sqrt{(2^{E_i^j})^2 \times \text{BitCnt}(i)}}
\]

Two determining factors: \( E_i \) and \( \text{BitCnt}(i) \)

Row vector of bit weight matrix

Column vector

\[
[A_1, A_2, \ldots, A_n]^T
\]

of activation matrix

Exponent at position \( j \) of \( i \)-th row

.. j-th bit of the \( i \)-th row vector in \( W \)

\( E_i \) and \( \text{BitCnt}(i) \)

A constant

\[
\sum_{i'=1}^{l} \sqrt{(2^{E_i^j})^2 \times \text{BitCnt}(i')}
\]
Methodology – BitX accelerator

Data shifter Zero-padding

Memory
DMA

Bypassed for fixed-point DNN

Perform bit pruning algorithm

Sort the shifted BitCnt(i) and select the top n largest rows

Methodology – BitX accelerator

Data shifter Zero-padding

Memory
DMA

Bypassed for fixed-point DNN

Perform bit pruning algorithm

Sort the shifted BitCnt(i) and select the top n largest rows

Methodology – BitX accelerator

Data shifter Zero-padding

Memory
DMA

Bypassed for fixed-point DNN

Perform bit pruning algorithm

Sort the shifted BitCnt(i) and select the top n largest rows

Methodology – BitX accelerator

Data shifter Zero-padding

Memory
DMA

Bypassed for fixed-point DNN

Perform bit pruning algorithm

Sort the shifted BitCnt(i) and select the top n largest rows

Methodology – BitX accelerator

Data shifter Zero-padding

Memory
DMA

Bypassed for fixed-point DNN

Perform bit pruning algorithm

Sort the shifted BitCnt(i) and select the top n largest rows
Methodology – computing units

Extract the significance of each essential bit $k$

Perform the final partial-sum accumulation

Computing Unit (CU)

Extract $s_0$ $K_0$ $K_6$ $K_7$ $K_{l-1}$ $E_{\text{max}}$

$A_0$ $A_1$ $\ldots$ $A_6$ $A_7$

adder tree

$R$ $>>$ out

Shift according to $k$'s significance
### Evaluation

#### • Accuracy & Sparsity

Parameter to control the granularity of pruning

<table>
<thead>
<tr>
<th>Model</th>
<th>Original</th>
<th>N=10</th>
<th>N=8</th>
<th>N=6</th>
<th>N=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet121</td>
<td>71.96/1x</td>
<td>71.95/1.34x</td>
<td>71.00/1.47x</td>
<td>71.00/1.62x</td>
<td>65.00/1.76x</td>
</tr>
<tr>
<td>DenseNet161</td>
<td>75.28/1x</td>
<td>75.20/1.32x</td>
<td>75.14/1.46x</td>
<td>74.79/1.61x</td>
<td>72.00/1.76x</td>
</tr>
<tr>
<td>DenseNet169</td>
<td>73.75/1x</td>
<td>73.56/1.31x</td>
<td>73.55/1.45x</td>
<td>73.55/1.60x</td>
<td>68.62/1.75x</td>
</tr>
<tr>
<td>Densenet201</td>
<td>74.56/1x</td>
<td>74.46/1.30x</td>
<td>74.40/1.44x</td>
<td>74.24/1.59x</td>
<td>69.00/1.74x</td>
</tr>
<tr>
<td>ResNet18</td>
<td>67.28/1x</td>
<td>67.09/1.64x</td>
<td>67.00/1.73x</td>
<td>66.72/1.81x</td>
<td>62.52/1.90x</td>
</tr>
<tr>
<td>ResNet34</td>
<td>71.32/1x</td>
<td>71.11/1.65x</td>
<td>71.10/1.73x</td>
<td>70.92/1.82x</td>
<td>68.00/1.90x</td>
</tr>
<tr>
<td>ResNet50</td>
<td>74.50/1x</td>
<td>74.50/1.41x</td>
<td>74.51/1.54x</td>
<td>74.10/1.67x</td>
<td>67.00/1.80x</td>
</tr>
<tr>
<td>ResNet101</td>
<td>76.00/1x</td>
<td>76.06/1.43x</td>
<td>76.05/1.55x</td>
<td>75.76/1.68x</td>
<td>69.02/1.81x</td>
</tr>
<tr>
<td>ResNet152</td>
<td>77.02/1x</td>
<td>76.56/1.44x</td>
<td>76.55/1.56x</td>
<td>76.46/1.69x</td>
<td>72.30/1.81x</td>
</tr>
<tr>
<td>ResNext50_32x4d</td>
<td>76.29/1x</td>
<td>75.99/1.24x</td>
<td>75.96/1.39x</td>
<td>75.67/1.56x</td>
<td>65.01/1.72x</td>
</tr>
<tr>
<td>ResNext101_32x8d</td>
<td>78.24/1x</td>
<td>78.20/1.27x</td>
<td>78.30/1.42x</td>
<td>78.10/1.58x</td>
<td>73.00/1.74x</td>
</tr>
<tr>
<td>SqueezeNet1_1</td>
<td>54.84/1x</td>
<td>54.86/1.42x</td>
<td>54.70/1.54x</td>
<td>54.40/1.67x</td>
<td>47.30/1.80x</td>
</tr>
<tr>
<td>Avg. loss / sparsity</td>
<td>0.000/1x</td>
<td>0.131/1.40x</td>
<td>0.242/1.52x</td>
<td>0.444/1.66x</td>
<td>6.023/1.79x</td>
</tr>
</tbody>
</table>

1. less than 0.5% average accuracy loss at N = 10, 8, 6.
2. Accuracy **improvement** on some models.

#### • Speedup

BitX exhibits promising speedup of $\sim 2.6x$ at $N = 10$, and $\sim 4.8x$ at $N = 6$. 
### Evaluation

- **Design Space Exploration**

<table>
<thead>
<tr>
<th>ResNet50</th>
<th>Original Accuracy: 74.50</th>
<th>DenseNet121</th>
<th>Original Accuracy: 71.96</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=10</td>
<td>74.50</td>
<td>M=8</td>
<td>71.95</td>
</tr>
<tr>
<td>N=8</td>
<td>74.51</td>
<td>M=16</td>
<td>71.97</td>
</tr>
<tr>
<td>N=6</td>
<td>74.10</td>
<td>M=32</td>
<td>72.03</td>
</tr>
<tr>
<td>N=4</td>
<td>67.00</td>
<td>M=64</td>
<td>71.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=128</td>
<td>71.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=256</td>
<td>71.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=512</td>
<td>71.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ResNet101_32x8d</th>
<th>Original Accuracy: 78.24</th>
<th>SqueezeNet1_1</th>
<th>Original Accuracy: 54.84</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=10</td>
<td>78.20</td>
<td>M=8</td>
<td>54.86</td>
</tr>
<tr>
<td>N=8</td>
<td>78.30</td>
<td>M=16</td>
<td>54.80</td>
</tr>
<tr>
<td>N=6</td>
<td>78.10</td>
<td>M=32</td>
<td>54.00</td>
</tr>
<tr>
<td>N=4</td>
<td>73.00</td>
<td>M=64</td>
<td>54.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=128</td>
<td>54.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=256</td>
<td>54.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=512</td>
<td>54.81</td>
</tr>
</tbody>
</table>

- **M** number of input weights that the accelerator could simultaneously prune

- **Two BitX instances:**
  - BitX-mild (N=10, M=8)
  - BitX-wild (N=6, M=8)

1. M barely influences the overall accuracy scaling from 8~512 for all 4 DNNs.
2. Accuracy improvement in some models.
• Performance of the Fixed-point DNN

<table>
<thead>
<tr>
<th>Model</th>
<th>Baseline(16b)</th>
<th>BitX-mild</th>
<th>BitX-wild</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet50</td>
<td>74.50</td>
<td>74.50</td>
<td>74.10</td>
</tr>
<tr>
<td></td>
<td>(0.00)</td>
<td>(-0.40)</td>
<td></td>
</tr>
<tr>
<td>SqueezeNet1_1</td>
<td>54.86</td>
<td>54.80</td>
<td>54.40</td>
</tr>
<tr>
<td></td>
<td>(-0.06)</td>
<td>(-0.46)</td>
<td></td>
</tr>
<tr>
<td>DenseNet121</td>
<td>71.00</td>
<td>71.90</td>
<td>71.80</td>
</tr>
<tr>
<td></td>
<td>(+0.90)</td>
<td>(+0.80)</td>
<td></td>
</tr>
<tr>
<td>ResNext101_32x8d</td>
<td>78.00</td>
<td>78.20</td>
<td>78.10</td>
</tr>
<tr>
<td></td>
<td>(+0.20)</td>
<td>(+0.10)</td>
<td></td>
</tr>
</tbody>
</table>

BitX-mild and BitX-wild both exhibit higher accuracy than most of non-pruned models.

1. ~2x speedup in BitX-wild.
2. ~10% but abundant speedup in BitX-mild.
### Evaluation

- **Working with Software-based Pruning**

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP(%)</th>
<th>Speedup(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YoloV3 (baseline)</td>
<td>50.36</td>
<td>1</td>
</tr>
<tr>
<td>YoloV3 + <strong>BitX-mild</strong></td>
<td>(50.42)</td>
<td>(+0.06)</td>
</tr>
<tr>
<td>YoloV3 + <strong>BitX-wild</strong></td>
<td>50.05</td>
<td>4.98</td>
</tr>
<tr>
<td>YoloV3 + Slimming (baseline)</td>
<td>50.23</td>
<td>2.35</td>
</tr>
<tr>
<td>YoloV3 + Slimming + <strong>BitX-mild</strong></td>
<td>50.30</td>
<td>(+0.07)</td>
</tr>
<tr>
<td>YoloV3 + Slimming + <strong>BitX-wild</strong></td>
<td>48.72</td>
<td>14.76</td>
</tr>
</tbody>
</table>

**BitX is orthogonal to any software-based pruning schemes**

- Higher speedup than software based pruning
- Considerable speedup than genetic model
Evaluation

- **Comparison with SOTA Accelerators**

1. The speedup shows better result over other SOTA accelerator both in BitX-wild and BitX-mild.

2. The floating-point results are higher than the fixed-point results.

Energy efficiency of BitX also outperforms other accelerators.
**Evaluation**

- **Energy breakdown**
  
  ![Energy Breakdown Chart]
  
  - Memory accesses dominate
  - CU dominates

- **Area and Power breakdown**

<table>
<thead>
<tr>
<th>Precision</th>
<th>BitX (floating-point 32)</th>
<th>BitX (16b fixed point)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Area (mm²)</td>
<td>Power (mW)</td>
</tr>
<tr>
<td>E-alignment</td>
<td>0.017 (43.60%)</td>
<td>11.15 (16.20%)</td>
</tr>
<tr>
<td>Bit Extraction</td>
<td>0.008 (20.10%)</td>
<td>0.04 (0.05%)</td>
</tr>
<tr>
<td>16 CUs</td>
<td>0.003 (7.70%)</td>
<td>53.71 (78.30%)</td>
</tr>
<tr>
<td>Misc&amp;Control</td>
<td>0.011 (28.20%)</td>
<td>3.72 (5.40%)</td>
</tr>
<tr>
<td>Total</td>
<td><strong>0.039</strong></td>
<td><strong>68.62</strong></td>
</tr>
</tbody>
</table>

1. **Area**: only 0.039 mm²
2. **36.41 mW**: high speedup, high power consumption
3. **68.62 mW**: low speedup, low power consumption
Recap

The contribution of this work

1. Propose a novel hardware runtime pruning method -- BitX, to empower versatile DNN inference

   - Software effortless
   - No retraining! No fine-tuning!

   - Orthogonal to the existing software pruning methodologies
   - Obtain additional speedup

   - Multi-precision support
   - Floating point & fixed point DNNs

2. Propose a deep learning accelerator capable of unprecedented hardware runtime pruning to mine the maximum potential of BitX.

Applications, and what’s more?
Thanks for listening!

Questions?

Hongyan Li\textsuperscript{12}, Hang Lu\textsuperscript{12}, Jiawen Huang\textsuperscript{1}, Wenxu Wang\textsuperscript{12}, Mingzhe Zhang\textsuperscript{1}, Wei Chen\textsuperscript{1}, Liang Chang\textsuperscript{3}, Xiaowei Li\textsuperscript{12}

\textsuperscript{1}State Key Laboratory of Computer Architecture, Institute of Computing Technology, CAS, Beijing, China

\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China

\textsuperscript{3}University of Electronic Science and Technology of China, Chengdu, China