Efficient Deep Convolutional Neural Networks Accelerator without Multiplication and Retraining

Weihong Xu\textsuperscript{1,2}, Xiaohu You\textsuperscript{2}, Chuan Zhang\textsuperscript{1,2}

\textsuperscript{1}Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
\textsuperscript{2}National Mobile Communications Research Laboratory, Southeast University, Nanjing, China

wh.xu@seu.edu.cn
April 17, 2018
Outline

1. Motivation
2. Related Work and Problem Formulation
3. Proposed Quantization and Hardware Co-design
4. Results and Analysis
5. Conclusion
Outline

1. Motivation

2. Related Work and Problem Formulation

3. Proposed Quantization and Hardware Co-design

4. Results and Analysis

5. Conclusion
Motivation

Image Segmentation

Game

NLP
Convolutional Neural Networks

- **Convolution**: feature extraction by convolving various filters over input image
- **Fully-connected**: linear transform over input features
- **Pooling and Non-linear**: perform down sampling and non-linear function
Major Challenges

- **Computation-intensive**: convolution takes up over 95% of overall complexity
  - $O(N^2 K^2)$ complexity per image $\rightarrow$ **Prohibitive complexity**
  - Floating point MAC is expensive $\rightarrow$ **Low energy efficiency**

- **Memory-intensive**: FC layers contribute 90% parameters
  - Densely connected networks $\rightarrow$ **Millions of weights**
  - Massive data movement $\rightarrow$ **Bandwidth limitation**
Outline

1. Motivation

2. Related Work and Problem Formulation

3. Proposed Quantization and Hardware Co-design

4. Results and Analysis

5. Conclusion
Low-precision Neural Networks

- Binarized Neural Networks
  - Binary weights \([-1, +1]\) with scaling factor \(\alpha\)
  - Activation: 32-bit float
  - \(\alpha\) is determined by \(L_1\)-norm of weights
  - Accuracy degradation: 19\% on AlexNet

1[Rastegari, Ordonez, Redmon, et al., ECCV 2016]
Low-precision Neural Networks

- **Ternary Weight Nets**
  - Ternary weights $\{-1, 0, +1\}$ with scaling factor $\alpha$
  - Activation: 32-bit float
  - Adding zero value increases expressive abilities of weights
  - Accuracy degradation: 3.7% on AlexNet

- **Objective of BNNs and TWNs**
  - Minimize distance between full precision weights $\mathbf{W}$ and the ternary weights $\mathbf{W}^t$ using scaling factor $\alpha$:
    \[
    \alpha^*, \mathbf{W}^{t*} = \arg\min_{\alpha, \mathbf{W}^t} ||\mathbf{W} - \alpha \mathbf{W}^t||^2
    \]

\[1\text{[Li, Zhang, and Liu, arXiv 2016]}\]
Non-Linear Quantization

- Distribution of weights in 5th layer of VGGNet
Non-Linear Quantization

- Distribution of weights in 15th layer of VGGNet

- Near normal distribution
- Deeper layers tend to have smaller weights
Non-Linear Quantization

- An intuitive perspective
Non-Linear Quantization

- An intuitive perspective

Inefficient  More efficient
Non-Linear Quantization

- **LogNet**
  - Weights: 4-bit, Activation: 32-bit
  - No scaling factor \( \alpha \rightarrow \text{Hardware friendly} \)
  - Substitute MAC with Shift and Add
  - Accuracy degradation: 4.9% on AlexNet without Retraining
  - Accuracy degradation: 4.6% on VGG16 with Retraining

\[ \text{Architecture (for convolutions)(b)} \]

\[ \text{Hardware (c)} \]

\[ \text{LogNet} \]

\[ s \text{ contains } 3 \times 3 \text{ filters} \]

\[ a^{(l)} \in \mathbb{Z}_{\geq 0}^{W \times W \times C^{(l)}} \]

\[ a^{(l-1)} \in \mathbb{Z}_{\geq 0}^{W \times W \times C^{(l-1)}} \]

\[ \text{input acts. of layer } l \]

\[ \text{output acts. of layer } l + 1 \]

\[ \left[ \text{Lee, Miyashita, Chai, et al., ICASSP 2017} \right] \]
Non-Linear Quantization

- **Incremental Network Quantization**
  - Incremental retraining on Log domain
  - Weights: 5-bit, Activation: 4-bit
  - Accuracy degradation: 1.16% on VGG16

---

1[Zhou, Yao, Guo, et al., ICLR 2017]
Problem Formulation

1. Trained Model
2. Quantize Weights by \( \log_2(w) \)
3. Restore Accuracy by Retraining
4. Accuracy Satisfied?
   - Yes
   - No
5. All Weights Quantized?
   - Yes
   - No
6. Low-precision Model
Problem Formulation

- Retraining is expensive!
Problem Formulation

- How to skip retraining?
Outline

1. Motivation

2. Related Work and Problem Formulation

3. Proposed Quantization and Hardware Co-design

4. Results and Analysis

5. Conclusion
Non-uniform Quantization

- More Log Bits $\neq$ Less Quantization Error

![Graph showing MSE vs. Bits for Log Quantization](image-url)
Non-uniform Quantization

Unable to quantize

Unable to quantize
Proposed Non-uniform Quantization

• Non-linear Quantization with Codebook

\[ \hat{w}_i = \sum_{n=1}^{N} \phi_n [\text{id}x_{i,n}] \]

  – \( \text{id}x_{i,n} \): \( i \)th segment of \( \hat{w}_i \)
  – \( N \) codebooks

• Codebook Structure

\[ \phi_n = \left[ 0, 2^{-1}, 2^{-2}, \ldots, 2^{-(2^{B_n}-1)} \right] \]

• Quantize weights to codebook index \( \text{id}x \)

• Only process codebook index during inference
Proposed Non-uniform Quantization

- **Example:** To quantize value 0.75
  - Log domain quantization: \(2^{\text{round}(\log_2(0.75))} = 2^{-1} = 0.5\)
  - Increasing bits don’t help!
Proposed Non-uniform Quantization

- **Example:** To quantize value 0.75
  - Log domain quantization: $2^{\text{round}(\log_2(0.75))} = 2^{-1} = 0.5$
  - Increasing bits doesn’t help!

- Reduce quantization error with $N = 2, B_1 = 1, B_2 = 2$
  - Codebook $\phi_1 = \{0, 2^{-1}\}, \phi_2 = \{0, 2^{-1}, 2^{-2}, 2^{-3}\}$

\[ \phi_1 = \{0, 2^{-1}\}, \quad \phi_2 = \{0, 2^{-1}, 2^{-2}, 2^{-3}\}, \]

\[ 1, 10 \]

- Quantized value: $\hat{w}_i = 1, 10 = 2^{-1} + 2^{-2} = 0.75$
Proposed Non-uniform Quantization

- Codebook index values tend to be centered within a range
- More bits are required without optimization
  - 3 bits for $\phi_1$, 4 bits for $\phi_2$ for this case

Figure: Index value distribution of FC layer in VGGNet16
Proposed Non-uniform Quantization

- Offset \( \beta_n \) to cover wider range

\[
\phi_n = \left[ 0, 2^{-1-\beta_n}, 2^{-2-\beta_n}, \ldots, 2^{-(2^{B_n}-1)-\beta_n} \right],
\]

\( \beta_1 = 0 \quad \beta_2 = 2 \)
Proposed Non-uniform Quantization

- Offset $\beta_n$ to cover wider range

$$\phi_n = \left[ 0, 2^{-1-\beta_n}, 2^{-2-\beta_n}, \ldots, 2^{-(2^{B_n}-1)-\beta_n} \right],$$

- Reduce to 3 bits for $\phi_1$, 3 bits for $\phi_2$
Proposed Non-uniform Quantization

• MSE criterion to determine optimal offset $\beta_n$:

$$
\beta_n = \arg\min_{\beta_n} \frac{1}{I} \sum_{i=0}^{I-1} ||\hat{w}_i - w_i||^2,
$$

• Weights in the same layer share the same offsets
• Only require $N$ offset values for a layer
• Increase quantization resolution
Efficient MAC Operation

- MAC based on shift and add
  \[ y = \hat{w}_i \cdot x_i + b = \sum_{n=1}^{N} \phi_n [\text{id}x_{i,n}] \cdot x_i + b. \]

- Codebook elements are all power of 2 or zero

- Shift and add instead of bulky multiplier

- One multiplication = \( N \) shift and \( N - 1 \) addition
Efficient MAC Operation

- Normalized energy and area cost comparison for single MAC unit for $N = 2, B_1 = B_2 = 3 \rightarrow (3, 3)$

<table>
<thead>
<tr>
<th></th>
<th>Power</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift-add MAC</td>
<td>1×</td>
<td>1×</td>
</tr>
<tr>
<td>Fixed-point MAC</td>
<td>7.3×</td>
<td>14.5×</td>
</tr>
</tbody>
</table>
Hardware Architecture

- Huffman Coding $\rightarrow$ Lossless compression
- Two-level Systolic Array $\rightarrow$ Maximize data reuse
Two-level Systolic Array

- 14 × 14 PE array
- Row Stationary (RS) dataflow → Minimize data movement
Dataflow of Systolic Array

1. Weights Broadcast

X₁
X₂
X₃
PE
PE
PE
W₁
PE

X₄
PE
PE
PE
W₂
PE

X₅
PE
PE
PE
W₃

X₁ * W₁ + X₂ * W₂ + X₃ * W₃

X₂ * W₁ + X₃ * W₂ + X₄ * W₃

X₃ * W₁ + X₄ * W₂ + X₅ * W₃
Dataflow of Systolic Array

2. Data Input (16-bit fixed)
Dataflow of Systolic Array

3. Data Output (Activation: 16-bit fixed)
Dataflow of Systolic Array

\[ X_1 \ast W_1 + X_2 \ast W_2 + X_3 \ast W_3 \]

\[ X_2 \ast W_1 + X_3 \ast W_2 + X_4 \ast W_3 \]

\[ X_3 \ast W_1 + X_4 \ast W_2 + X_5 \ast W_3 \]
Processing Element

- Each PE contains 5 Cells
- Cell implements shift-add MAC operation
- 1-D systolic convolution → **Higher throughput**
Dataflow of PE

- Weights stay
- Input data move systolically
Dataflow of PE

- Weights stay
- Input data move systolically
Dataflow of PE

- Weights stay
- Input data move systolically
Outline

1. Motivation

2. Related Work and Problem Formulation

3. Proposed Quantization and Hardware Co-design

4. Results and Analysis

5. Conclusion
Test on AlexNet

- Codebook size $N = 2$ without **Retraining**

<table>
<thead>
<tr>
<th>Model</th>
<th>Codebook</th>
<th>Top-1/top-5 Accuracy</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>Baseline</td>
<td>56.55%/79.09%</td>
<td>--/--</td>
</tr>
<tr>
<td></td>
<td>(3, 2)</td>
<td>41.76%/66.22%</td>
<td>-14.79%/-12.87%</td>
</tr>
<tr>
<td></td>
<td>(4, 2)</td>
<td>48.36%/72.33%</td>
<td>-8.19%/-6.76%</td>
</tr>
<tr>
<td></td>
<td>(3, 3)</td>
<td>54.98%/77.89%</td>
<td>-1.57%/-1.20%</td>
</tr>
<tr>
<td></td>
<td>(4, 4)</td>
<td>55.45%/78.64%</td>
<td>-1.10%/-0.45%</td>
</tr>
</tbody>
</table>

[†] Top-1/top-5 error are tested with single center crop.
Test on AlexNet

- Quantization MSE comparison

![Graph showing MSE comparison between Log Quantization and Proposed methods. The x-axis represents Bits and the y-axis represents MSE. The graph shows a significant decrease in MSE as the number of bits increases. Log Quantization starts higher than Proposed and then remains relatively flat, while Proposed shows a more pronounced decrease.]
Validation on ImageNet

- Quantize pretrained AlexNet, VGGNet16, ResNet34 model from Pytorch
- Codebook size $N = 2$ with $B_1 = B_2 = 3$

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Bit-width</th>
<th>Degradation</th>
<th>Retraining</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>Baseline</td>
<td>32</td>
<td>−/−</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>(3, 3)</td>
<td>−1.57% / −1.20%</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>LogNet</td>
<td>5</td>
<td>−/ −3.70%</td>
<td>No</td>
</tr>
<tr>
<td>VGGNet-16</td>
<td>Baseline</td>
<td>32</td>
<td>−/−</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>(3, 3)</td>
<td>−2.23% / −1.95%</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Fixed-point</td>
<td>16</td>
<td>−3.58% / −2.49%</td>
<td>No</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>Baseline</td>
<td>32</td>
<td>−/−</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>(3, 3)</td>
<td>−1.97% / −1.17%</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ShiftCNN</td>
<td>(4, 4)</td>
<td>−3.21% / −2.05%</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>TWNs</td>
<td>2</td>
<td>−2.56% / −1.80%</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[*] Top-1/top-5 error are tested with single center crop.

[†] Degradation is taken from original papers.
Model Compression

![Bar chart showing normalized model size for different models and compression methods.]

- **Baseline**: 8.25x for AlexNet, 7.41x for VGGNet-16, 5.33x for ResNet-34.
- **Q**: 1.55x for AlexNet, 1.39x for VGGNet-16, 1.51x for ResNet-34.
- **Q+Huffman**: 1.00x for all models.
## Implementation Results

<table>
<thead>
<tr>
<th>Design</th>
<th>Qiu2016</th>
<th>Zhang2016</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Zynq XC7Z045</td>
<td>Virtex-7 VX690t</td>
<td>Virtex-7 VX690t</td>
</tr>
<tr>
<td>Clock(MHz)</td>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>Quantization</td>
<td>16-bit fixed</td>
<td>16-bit fixed</td>
<td>(3, 3)</td>
</tr>
<tr>
<td>LUT</td>
<td>186,251</td>
<td>$\approx$ 300,000</td>
<td>107995</td>
</tr>
<tr>
<td>FF</td>
<td>127,653</td>
<td>$\approx$ 300,000</td>
<td>117795</td>
</tr>
<tr>
<td>DSP</td>
<td>2240</td>
<td>2833</td>
<td>0</td>
</tr>
<tr>
<td>BRAM</td>
<td>1024</td>
<td>1248</td>
<td>1279</td>
</tr>
<tr>
<td>Throughput (GOP/s)</td>
<td>187.8</td>
<td>636.0</td>
<td>238.2</td>
</tr>
</tbody>
</table>

1 [Qiu, Wang, Yao, et al., ISFPGA 2016]
2 [Zhang, Fang, Zhou, et al., ICCAD 2016]
Outline

1. Motivation
2. Related Work and Problem Formulation
3. Proposed Quantization and Hardware Co-design
4. Results and Analysis
5. Conclusion
Conclusion

- **A framework to implement low-precision CNNs**
  - Non-uniform quantization with multiple codebooks and offset
  - Retraining-free quantization approaches
  - Multiplier-free shift-add convolution

- **Efficient hardware architecture**
  - Two-level systolic to maximize data reuse
  - Huffman compression to reduce memory bandwidth
  - 1-D systolic PEs to obtain high throughput
Reference


Thanks for Your Attention!