Efficient Deep Convolutional Neural Networks Accelerator without Multiplication and Retraining

Weihong Xu[1,2], **Xiaohu You**[2], **Chuan Zhang**[1,2]

[1]Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
[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

Motivation

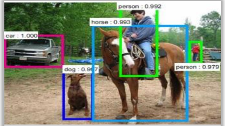
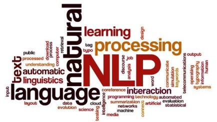


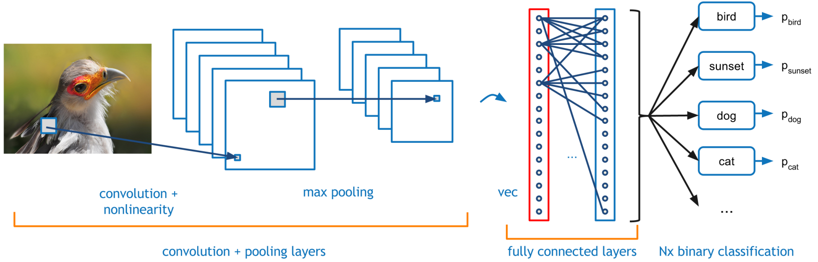
Image Segmentation

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

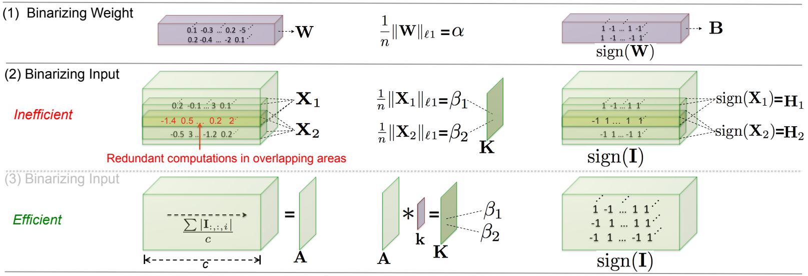
Major Challenges

- **Computation-intensive**: convolution takes up over 95% of ovarall complexity
  - $\mathcal{O}(N^2K^2)$ complexity per image $\longrightarrow$ Prohibitive complexity
  - Floating point MAC is expensive $\longrightarrow$ Low energy efficiency
- **Memory-intensive**: FC layers contribute 90% parameters
  - Densely connected networks $\longrightarrow$ Millions of weights
  - Massive data movement $\longrightarrow$ Bandwidth limitation

Outline

Low-precision Neural Networks

- **Binarized Neural Networks**
  - Binary weights $\{-1, +1\}$ with scaling factor α
  - Activation: 32-bit float
  - α 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

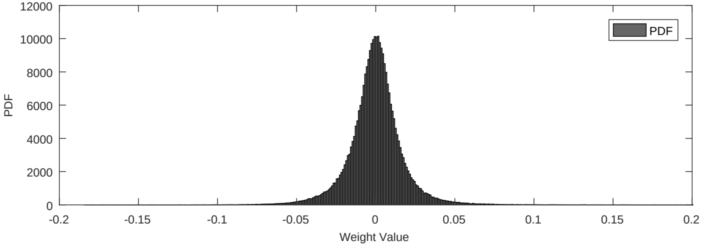
- **Ternary Weight Nets**
  - Ternary weights $\{-1, 0, +1\}$ with scaling factor α
  - Activation: 32-bit float
  - Adding zero value increases expressive abilities of weights
  - Accuracy degradation: 3.7% on AlexNet
- **Objective of BNNs and TWNs**
  - Minimize distance between full precision weights $\mathbf{W}$ and the ternary weights $\mathbf{W}^t$ using scaling factor α:

$$\alpha^*, \mathbf{W}^{t*} = \underset{\alpha, \mathbf{W}^t}{\arg\min} \, ||\mathbf{W} - \alpha \mathbf{W}^t||^2$$

[1] [Li, Zhang, and Liu, *arXiv* 2016]

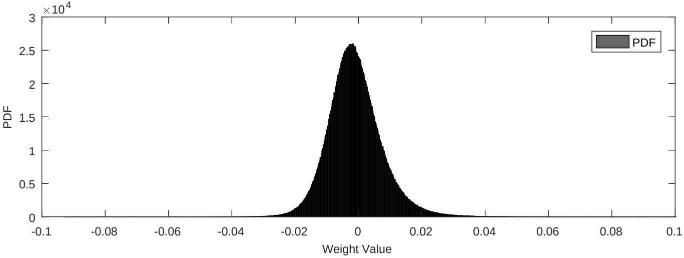
Non-Linear Quantization

- **Distribution of weights in 5th layer of VGGNet**



Non-Linear Quantization

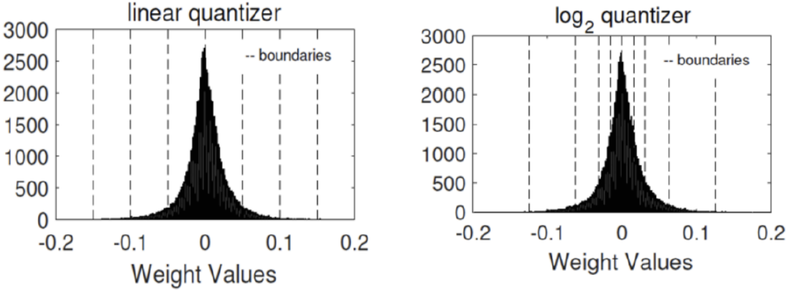
- **Distribution of weights in 15th layer of VGGNet**



  - Near normal distribution
  - Deeper layers tend to have smaller weights

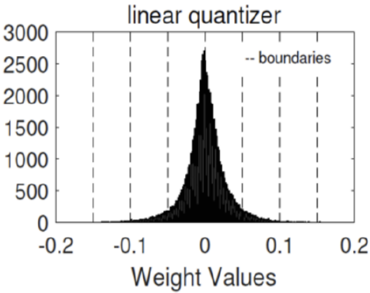
Non-Linear Quantization

- **An intuitive perspective**

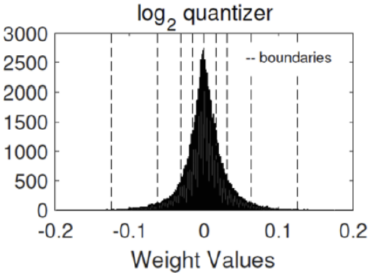


Non-Linear Quantization

- **An intuitive perspective**



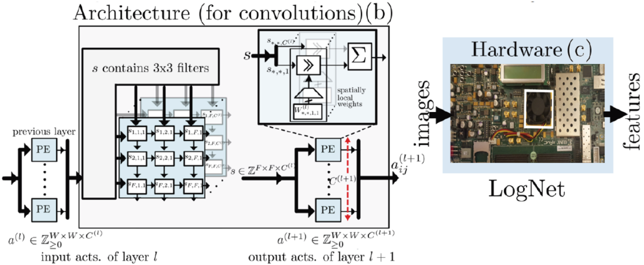
Inefficient



More efficient

Non-Linear Quantization

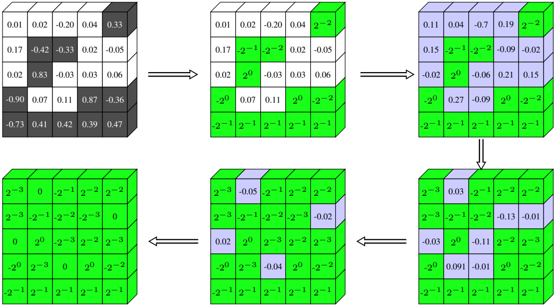
- **LogNet**
  - Weights: 4-bit, Activation: 32-bit
  - No scaling factor $\alpha \longrightarrow$ Hardware friendly
  - Substitute MAC with Shift and Add
  - Accuracy degradation: 4.9% on AlexNet without Retraining
  - Accuracy degradation: 4.6% on VGG16 with Retraining



[1][Lee, Miyashita, Chai, *et al.*, *ICASSP* 2017]

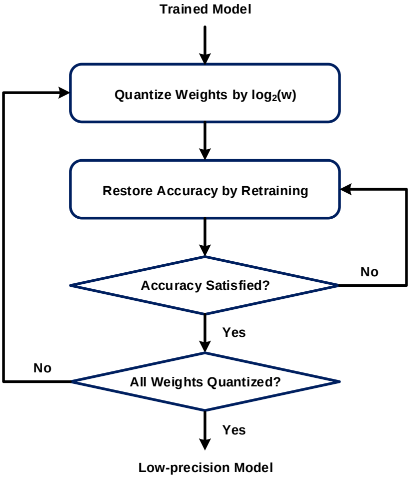
Non-Linear Quantization

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

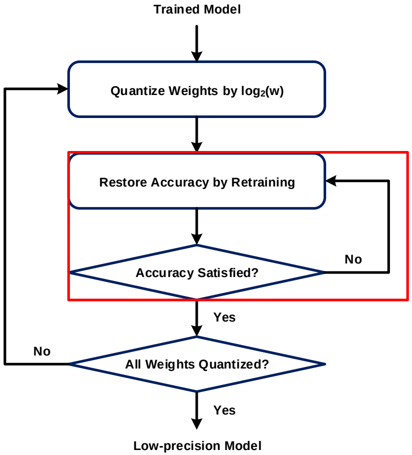


[1] [Zhou, Yao, Guo, *et al.*, *ICLR* 2017]

Problem Formulation

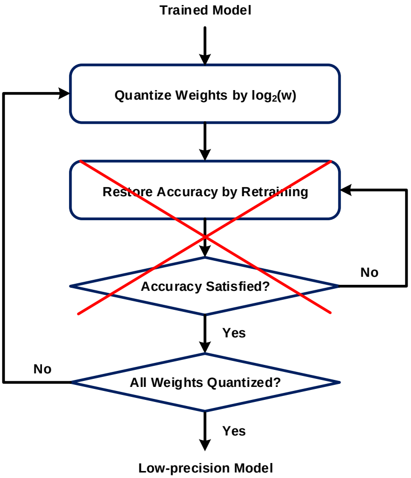


Problem Formulation



- Retraining is expensive!

Problem Formulation



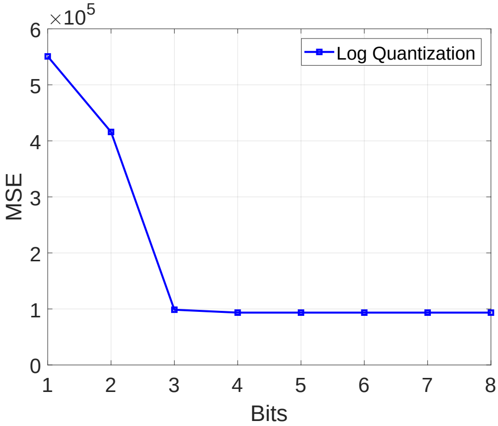
- How to skip retraining?

Outline

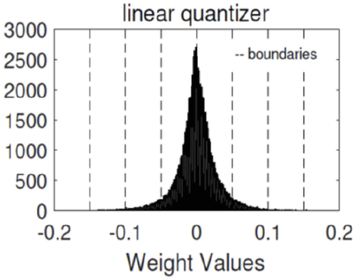
1. Motivation
2. Related Work and Problem Formulation
3. Proposed Quantization and Hardware Co-design
4. Results and Analysis
5. Conclusion

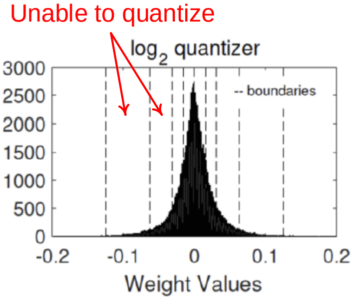
Non-uniform Quantization

- **More Log Bits $\neq$ Less Quantization Error**



Non-uniform Quantization





Proposed Non-uniform Quantization

- **Non-linear Quantization with Codebook**

$$\hat{w}_i = \sum_{n=1}^{N} \phi_n \left[\mathrm{idx_{i,n}}\right]$$

  - $\mathsf{idx}_{i,n}$: ith segment of $\hat{w}_i$
  - N codebooks

- **Codebook Structure**

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

- **Quantize weights to codebook index** idx
- **Only process codebook index during inference**

Proposed Non-uniform Quantization

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

Proposed Non-uniform Quantization

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

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

$$\phi_1 = \{0,\ 2^{-1}\},\ \ \phi_2 = \{0,\ 2^{-1},\ 2^{-2},\ 2^{-3}\},$$

$$1,\ 10$$

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

Proposed Non-uniform Quantization

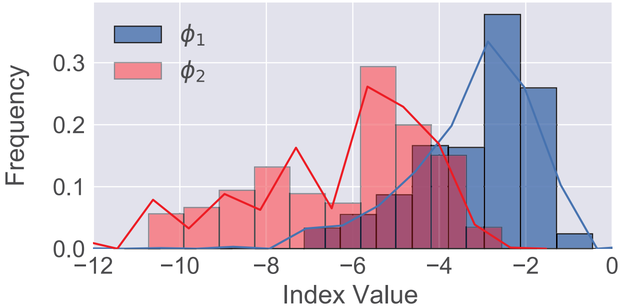


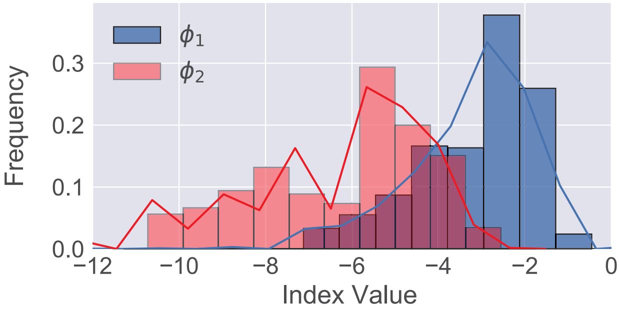
Figure: Index value distribution of FC layer in VGGNet16

- **Codebook index values tend to be centered within a range**
- **More bits are required without optimization**
  - 3 bits for ϕ_1, 4 bits for ϕ_2 for this case

Proposed Non-uniform Quantization

- **Offset β_n to cover wider range**

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

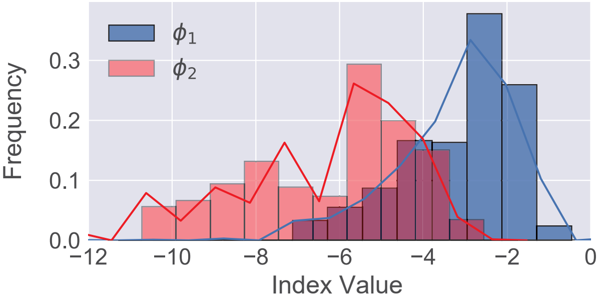


$$\beta_1 = 0 \qquad \beta_2 = 2$$

Proposed Non-uniform Quantization

- **Offset β_n to cover wider range**

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



  – Reduce to 3 bits for ϕ_1, 3 bits for ϕ_2

Proposed Non-uniform Quantization

- **MSE criterion to determine optimal offset β_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**

$$y = \hat{w}_i * x_i + b = \sum_{n=1}^{N} \phi_n \left[\mathrm{idx_{i,n}}\right] * x_i + b.$$

- **Codebook elements are all power of 2 or zero**
- **Shift and add instead of bulky multiplier**
- **One multiplication = N shift and $N-1$ addition**

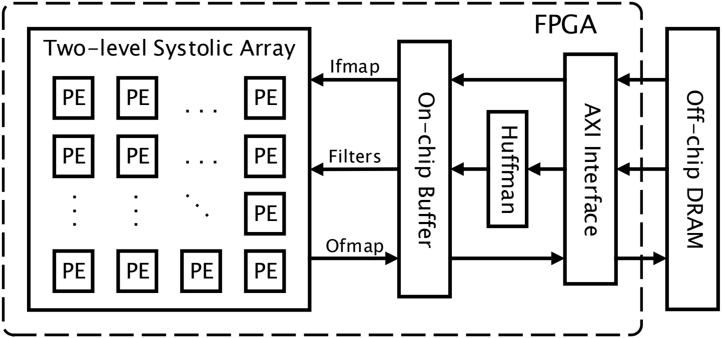
Efficient MAC Operation

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

	Power	Area
Shift-add MAC	$1\times$	$1\times$
Fixed-point MAC	$7.3\times$	$14.5\times$

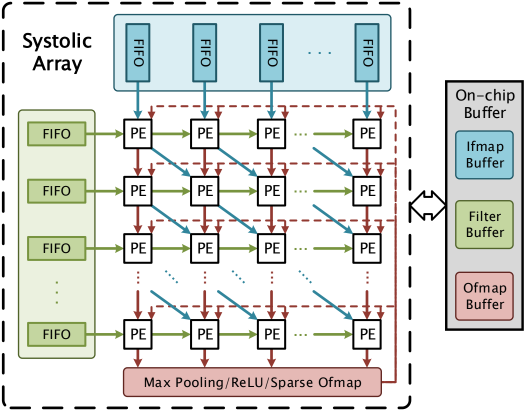
Hardware Architecture

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



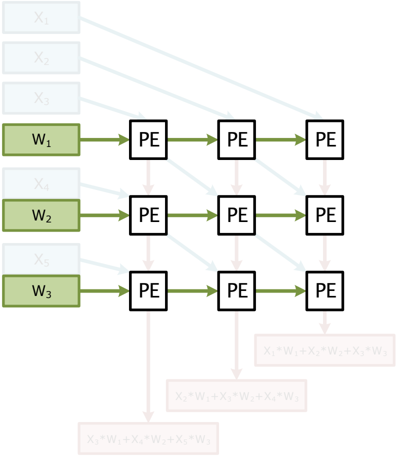
Two-level Systolic Array

- 14× 14 PE array
- Row Stationary (RS) dataflow ⟶ Minimize data movement



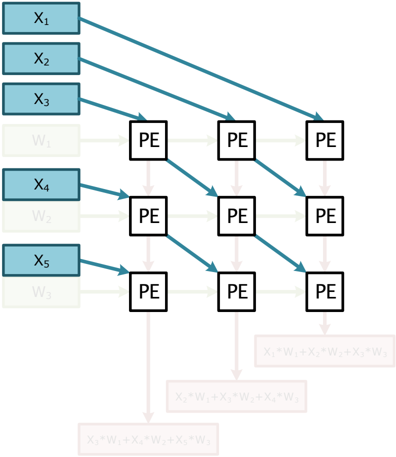
Dataflow of Systolic Array

1. Weights Broadcast



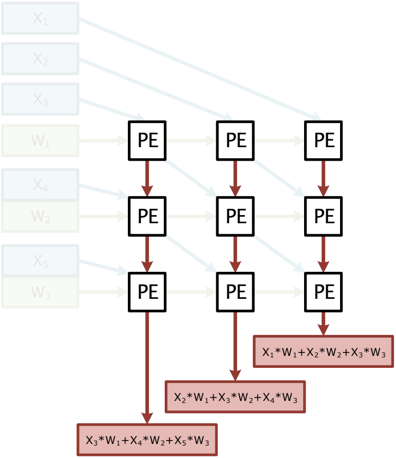
Dataflow of Systolic Array

2. Data Input (16-bit fixed)

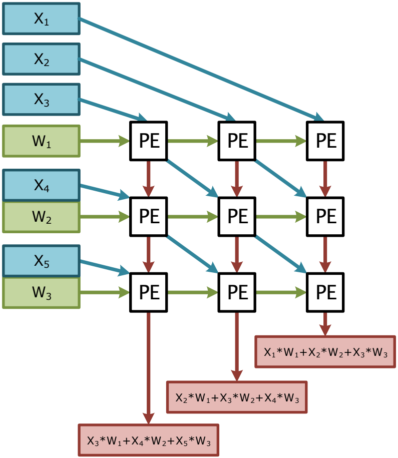


Dataflow of Systolic Array

3. Data Output (Activation: 16-bit fixed)

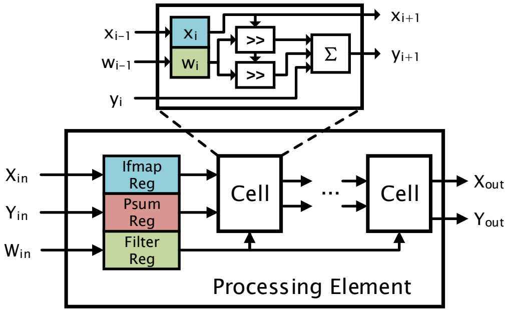


Dataflow of Systolic Array



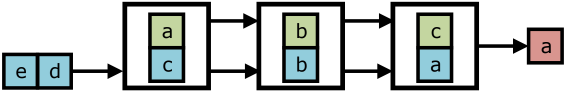
Processing Element

- Each PE contains 5 Cells
- Cell implements shift-add MAC operation
- 1-D systolic convolution $\longrightarrow$ Higher throughput



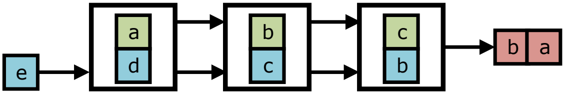
Dataflow of PE

- Weights stay
- Input data move systolically



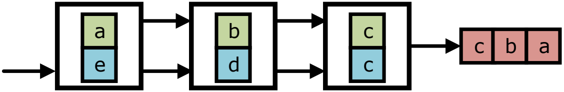
Dataflow of PE

- Weights stay
- Input data move systolically



Dataflow of PE

- Weights stay
- Input data move systolically



Outline

Test on AlexNet

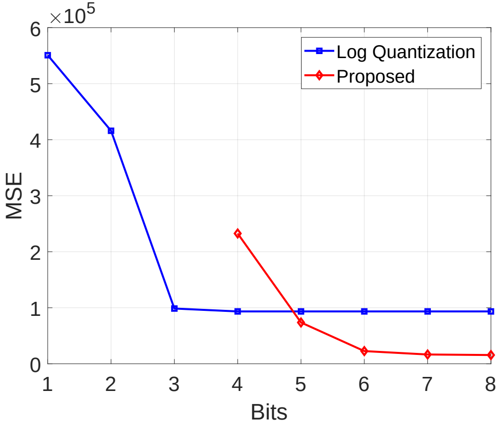
- Codebook size $N = 2$ without **Retraining**

Model	Codebook	Top-1/top-5 Accuracy	Degradation
AlexNet	Baseline	$56.55\%/79.09\%$	$-/-$
	$(\mathbf{3}, \mathbf{2})$	$41.76\%/66.22\%$	$\mathbf{-14.79}\%/-\mathbf{12.87}\%$
	$(\mathbf{4}, \mathbf{2})$	$48.36\%/72.33\%$	$\mathbf{-8.19}\%/-\mathbf{6.76}\%$
	$(\mathbf{3}, \mathbf{3})$	$54.98\%/77.89\%$	$\mathbf{-1.57}\%/-\mathbf{1.20}\%$
	$(\mathbf{4}, \mathbf{4})$	$55.45\%/78.64\%$	$\mathbf{-1.10}\%/-\mathbf{0.45}\%$

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

Test on AlexNet

- Quantization MSE comparison



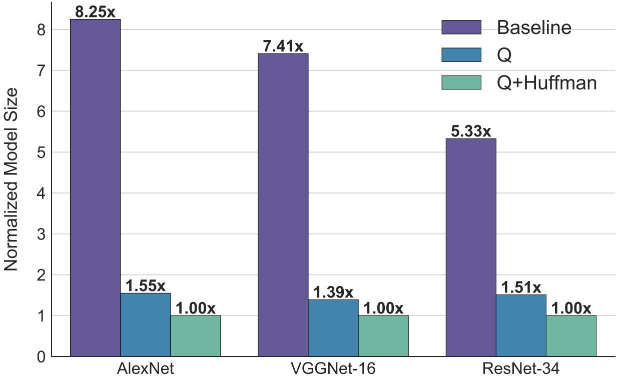
Validation on ImageNet

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

Model	Method	Bit-width	Degradation	Retraining
AlexNet	Baseline	32	$-/-$	No
	Proposed	$(\mathbf{3}, \mathbf{3})$	$-\mathbf{1.57}\%/-\mathbf{1.20}\%$	**No**
	LogNet	5	$-/-3.70\%$	**No**
VGGNet-16	Baseline	32	$-/-$	No
	Proposed	$(\mathbf{3}, \mathbf{3})$	$-\mathbf{2.23}\%/-\mathbf{1.95}\%$	**No**
	Fixed-point	16	$-3.58\%/-2.49\%$	No
ResNet-18	Baseline	32	$-/-$	No
	Proposed	$(\mathbf{3}, \mathbf{3})$	$-\mathbf{1.97}\%/-\mathbf{1.17}\%$	**No**
	ShiftCNN	$(4, 4)$	$-3.21\%/-2.05\%$	No
	TWNs	2	$-2.56\%/-1.80\%$	Yes

[†] Top-1/top-5 error are tested with single center crop.
[$*$] Degradation is taken from original papers.

Model Compression



Implementation Results

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

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

Outline

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

1. M. Rastegari, V. Ordonez, J. Redmon, *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016
2. F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016
3. E. H. Lee, D. Miyashita, E. Chai, *et al.*, "Lognet: Energy-efficient neural networks using logarithmic computation," in *ICASSP*, 2017
4. A. Zhou, A. Yao, Y. Guo, *et al.*, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *ICLR*, 2017
5. J. Qiu, J. Wang, S. Yao, *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *ISFPGA*, 2016
6. C. Zhang, Z. Fang, P. Zhou, *et al.*, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," in *ICCAD*, 2016
7. D. A. Gudovskiy and L. Rigazio, "Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks," *arXiv preprint arXiv:1706.02393*, 2017

Thanks for Your Attention!

Q & A