Incremental Network Quantization: Towards Lossless CNNs With Low-Precision Weights

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, Yurong Chen

Presented by Zhuangwei Zhuang

South China University of Technology

June 6, 2017

Outline

- Background
- Motivation
- Proposed Methods
  - Variable-length enconding
  - Incremental quantization strategy
- Experimental Results
- Conclusions

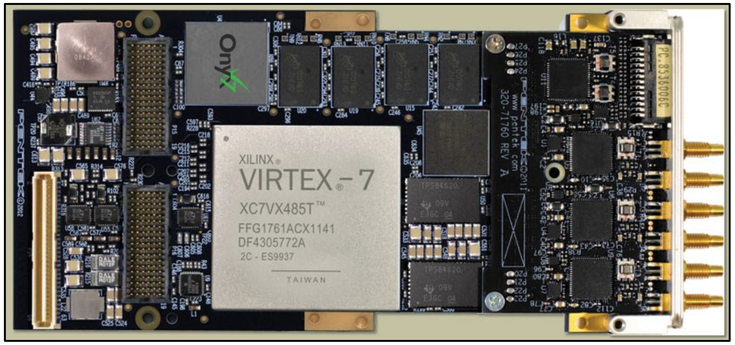
Background

Background

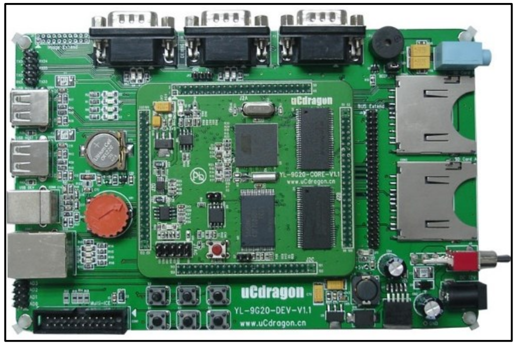
Huge networks lead to heavy consumption on memory and computational resources.

- ResNet-152 has model size of **230 MB**, and needs about **11.3 billion FLOPs** for a 224×224 image

Difficult to implement deep CNNs on **hardware** with the limitation of computation and power.



FPGA

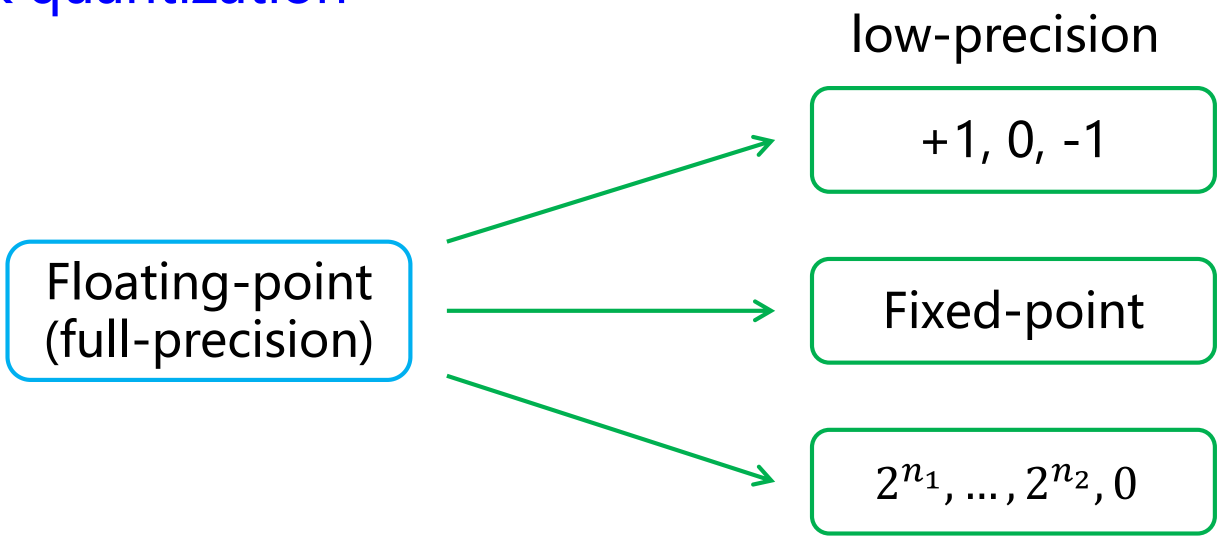


ARM

Motivation

Motivation

- Network quantization



CNN quantization still an open question due to two critical issues:

- Non-negligible accuracy loss for CNN quantization methods
- Increased number of training iterations for ensuring convergence

Proposed Methods

Proposed Methods

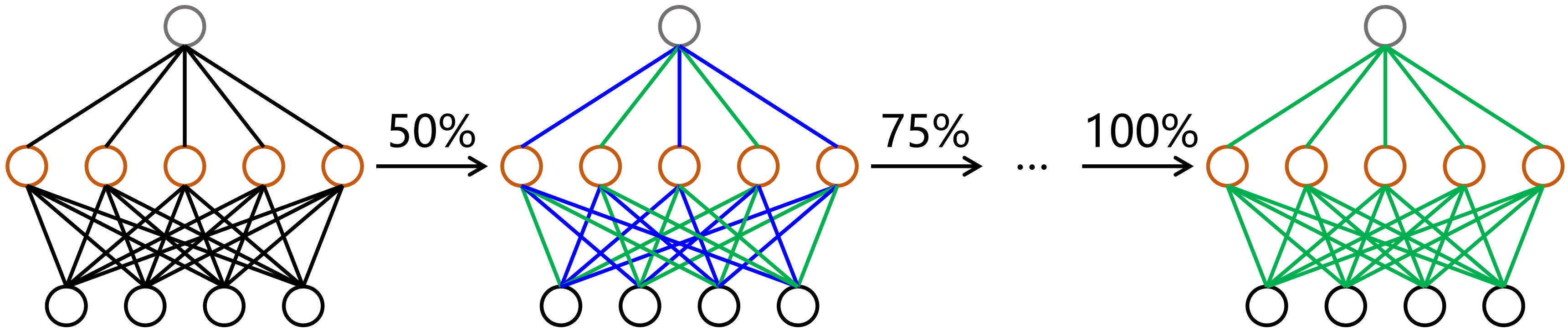


Figure. Overview of INQ

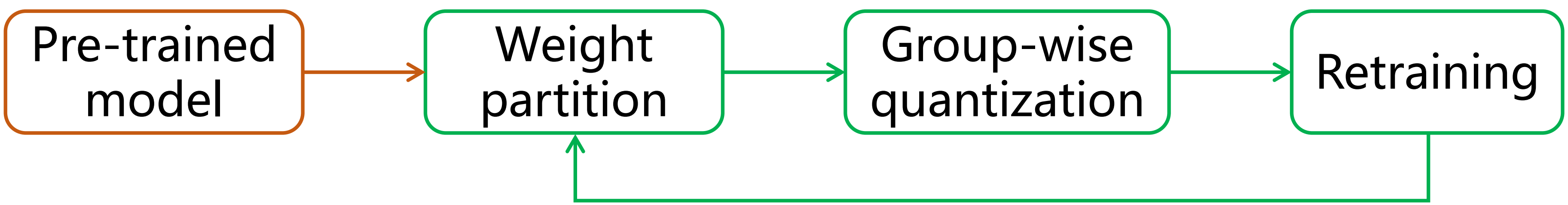


Figure. Quantization strategy of INQ

Variable-Length Encoding

Suppose a pre-trained full precision CNN model can be represented by $\{\mathrm{W}_l: 1 \leq l \leq L\}$.

W_l: weight set of l^{th} layer

L: number of layers

Goal of INQ: Convert 32 floating-point W_l to be low-precision $\widehat{\mathrm{W}_l}$, each entry of $\widehat{\mathrm{W}_l}$ is chosen from

$$\mathrm{P}_l = \{\pm 2^{n_1}, \cdots, \pm 2^{n_2}, 0\},$$

where n_1 and n_2 are two integer numbers, and $n_2 \leq n_1$.

Variable-Length Encoding

$$\mathrm{P}_l = \{\pm 2^{n_1}, \cdots, \pm 2^{n_2}, 0\}$$

- $\widehat{W}_l$ is computed by:

$$\widehat{\mathrm{W}}_l(i,j) = \begin{cases} \beta \operatorname{sgn}\left(\widehat{W}_l(i,j)\right) & \text{if } (\alpha+\beta) \le abs\left(\mathrm{W}_l(i,j)\right) < 3\beta/2 \\ 0 & \text{otherwise,} \end{cases}$$

Where α and β are two adjacent elements in the sorted P_l, and $0 \le \alpha < \beta$.

Variable-Length Encoding

$$\mathrm{P}_l = \{\pm 2^{n_1}, \cdots, \pm 2^{n_2}, 0\}$$

- Define bit-width b

  1 bit to represent 0, and the remaining bits to represent $\pm 2^n$

- n_1 and n_2 are computed by

$$n_1 = \mathrm{floor}(\log_2(4s/3))$$

$$n_2 = n_1 + 1 - 2^{b-1}/2$$

- s is calculated by

$$s = \max(\mathrm{abs}(\mathrm{W}_l))$$

Incremental Quantization Strategy

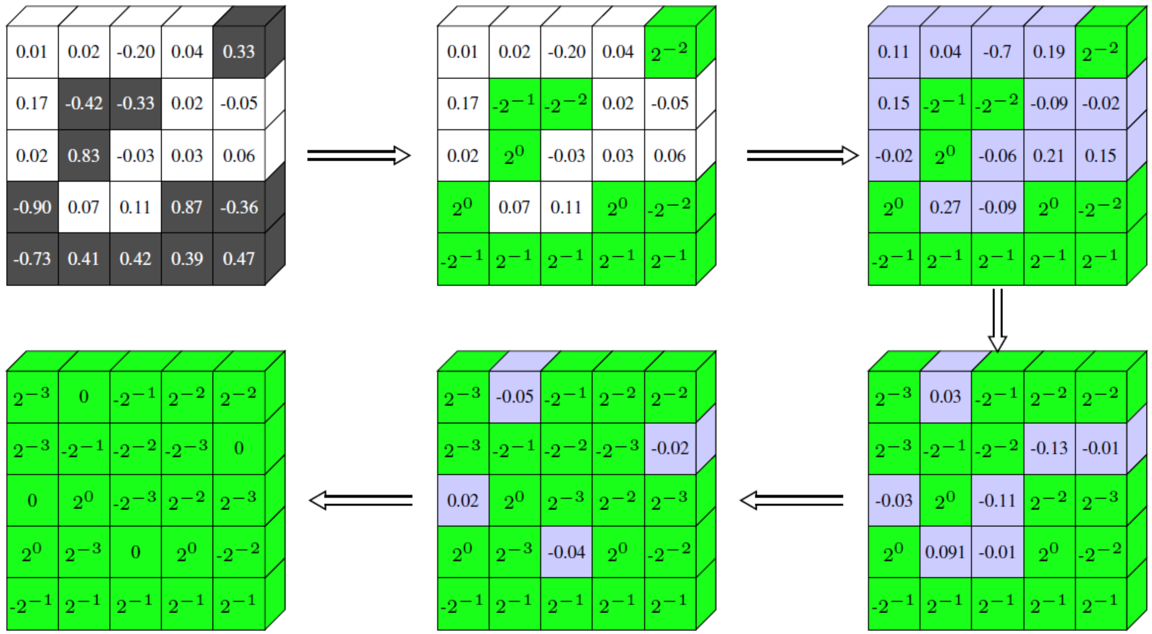


Figure. Result illustrations

- Quantization strategy:
  - **Weight partition:** divide weights in each layers into two disjoint groups
  - **Group-wise quantization:** quantize weights in first group
  - **Retraining:** retrain whole network and update weights in second group

Incremental Quantization Strategy

For the l^{th}, weight partition can be defined as

$$\mathrm{A}_l^{(1)} \cup \mathrm{A}_l^{(2)} = \{\mathrm{W}_l(i,j)\}, \text{and } \mathrm{A}_l^{(1)} \cap \mathrm{A}_l^{(2)} = \emptyset$$

$\mathrm{A}_l^{(1)}$: first weight group that needs to be quantized

$\mathrm{A}_l^{(2)}$: second weight group that needs to be retrained

- Define binary matrix T_l

$$\mathrm{T}_l(i,j) = \begin{cases} 0, \mathrm{W}_l(i,j) \in \mathrm{A}_l^{(1)} \\ 1, \mathrm{W}_l(i,j) \in \mathrm{A}_l^{(2)} \end{cases}$$

- Update W_l

$$\mathrm{W}_l(i,j) \leftarrow \mathrm{W}_l(i,j) - \gamma \frac{\partial E}{\partial\big(\mathrm{W}_l(i,j)\big)} \mathrm{T}_l(i,j)$$

Incremental Quantization Strategy

Algorithm. Pseudo Code of INQ

Algorithm 1 Incremental network quantization for lossless CNNs with low-precision weights.

Input: X: the training data, $\{\mathbf{W}_l : 1 \leq l \leq L\}$: the pre-trained full-precision CNN model, $\{\sigma_1, \sigma_2, \cdots, \sigma_N\}$: the accumulated portions of weights quantized at iterative steps

Output: $\{\widehat{\mathbf{W}}_l : 1 \leq l \leq L\}$: the final low-precision model with the weights constrained to be either powers of two or zero

1: Initialize $\mathbf{A}_l^{(1)} \leftarrow \emptyset$, $\mathbf{A}_l^{(2)} \leftarrow \{\mathbf{W}_l(i,j)\}$, $\mathbf{T}_l \leftarrow \mathbf{1}$, for $1 \leq l \leq L$
2: **for** $n = 1, 2, \ldots, N$ **do**
3: Reset the base learning rate and the learning policy
4: According to σ_n, perform layer-wise weight partition and update $\mathbf{A}_l^{(1)}$, $\mathbf{A}_l^{(2)}$ and $\mathbf{T}_l$
5: Based on $\mathbf{A}_l^{(1)}$, determine $\mathbf{P}_l$ layer-wisely
6: Quantize the weights in $\mathbf{A}_l^{(1)}$ by Equation (4) layer-wisely
7: Calculate feed-forward loss, and update weights in $\{\mathbf{A}_l^{(2)} : 1 \leq l \leq L\}$ by Equation (8)
8: **end for**

Experimental Results

Results on ImageNet

Table. Converting full-precision models to 5-bit versions

Network	Bit-width	Top-1 error	Top-5 error	Decrease in top-1/top-5 error
AlexNet ref	32	42.76%	19.77%	
AlexNet	5	**42.61%**	**19.54%**	0.15%/0.23%
VGG-16 ref	32	31.46%	11.35%	
VGG-16	5	**29.18%**	**9.70%**	2.28%/1.65%
GoogleNet ref	32	31.11%	10.97%	
GoogleNet	5	**30.98%**	**10.72%**	0.13%/0.25%
ResNet-18 ref	32	31.73%	11.31%	
ResNet-18	5	**31.02%**	**10.90%**	0.71%/0.41%
ResNet-50 ref	32	26.78%	8.76%	
ResNet-50	5	**25.19%**	**7.55%**	1.59%/1.21%

Analysis of Weight Partition Strategies

- **Random partition**: all weights have equal probability to fall into the two groups
- **Pruning-inspired partition**: weights with larger absolute values have more probability to be quantized

Table. Comparison of different weight partition strategies on ResNet-18

Strategy	Bit-width	Top-1 error	Top-5 error
Random partition	5	32.11%	11.73%
Pruning-inspired partition	5	**31.02%**	**10.90%**

Trade-Off Between Bit-Width and Accuracy

Table. Exploration on bit-width on ResNet-18

Model	Bit-width	Top-1 error	Top-5 error
ResNet-18 ref	32	31.73%	11.31%
INQ	5	31.02%	10.90%
INQ	4	31.11%	10.99%
INQ	3	31.92%	11.64%
INQ	2 (ternary)	33.98%	12.87%

Table. Comparison of the proposed ternary model and the baselines on ResNet-18

Method	Bit-width	Top-1 error	Top-5 error
BWN(Rastegari et al., 2016)	1	39.20%	17.00%
TWN(Li & Liu, 2016)	2 (ternary)	38.20%	15.80%
INQ (ours)	2 (ternary)	**33.98%**	**12.87%**

Low-Bit Deep Compression

Table. Comparison of INQ+DNS, and deep compression method on AlexNet. Conv: Convolutional layer, FC: Fully connected layer, **P:** Pruning, **Q:** Quantization, **H:** Huffman coding

Method	Bit-width(Conv/FC)	Compression ratio	Decrease in top-1/top5 error
Han et al. (2016) (P+Q)	8/5	27×	0.00%/0.03%
Han et al. (2016) (P+Q+H)	8/5	35×	0.00%/0.03%
Han et al. (2016) (P+Q+H)	8/4	-	-0.01%/0.00%
Our method (P+Q)	5/5	**53**×	**0.08%/0.03%**
Han et al. (2016) (P+Q+H)	4/2	-	-1.99%/-2.60%
Our method (P+Q)	4/4	**71**×	**-0.52%/-0.20%**
Our method (P+Q)	3/3	**89**×	**-1.47%/-0.96%**

Conclusions

Conclusions

- Contributions
  - Present INQ to convert any pre-trained full-precision CNN model into a **lossless** low-precision version
  - The quantized models with **5/4/3/2** bits achieve comparable accuracy against their full-precision baselines
- Future work
  - Extend incremental idea from low-precision **weights** to low-precision **activations** and low-precision **gradients**.
  - Implement the proposed low-precision models on **hardware** platforms

Q & A

References

[1] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *In ICLR*, 2017.

[2] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *In NIPS*, 2016.

[3] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *In NIPS*, 2015.

[4] Song Han, Jeff Pool, John Tran, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *In ICLR*, 2016.

[5] Fengfu Li and Bin Liu. Ternary weight networks. *arXiv preprint arXiv: 1605.04711v1*, 2016

[6] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv: 1603.05279v4*, 2016.