Deep Neural Network Quantization via Layer-Wise Optimization using Limited Training Data

Shangyu Chen, Wenya Wang, Sinno Jialin Pan

School of Computer Science and Engineering
NTU

Outline

Deep Neural Network Quantization in Edge Devices

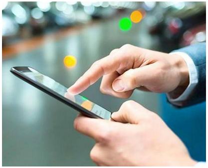


Figure 1: Smartphones

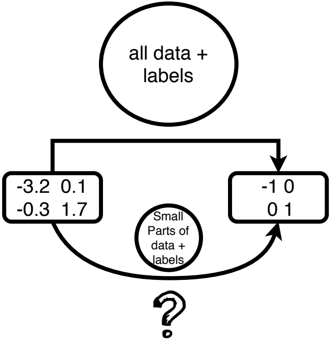
Figure 2: Cameras

- More and more deep learning applications are deployed in edge devices: cellphone, surveillance camera, etc.
- It is impossible to perform inference without optimization:
  - Compress the model: Convert float numbers into limited integers.
  - Accelerate computation: float-float multiplication to float-integer multiplication.

Limitation of Current Quantization

Most existing methods rely access to full training data and labels:

- Data privacy in commercial models with high confidential requirement.
- Impossible to store all data in edge devices for on-device quantization.
- Accuracy is preserved (especially for non training-based quantization).



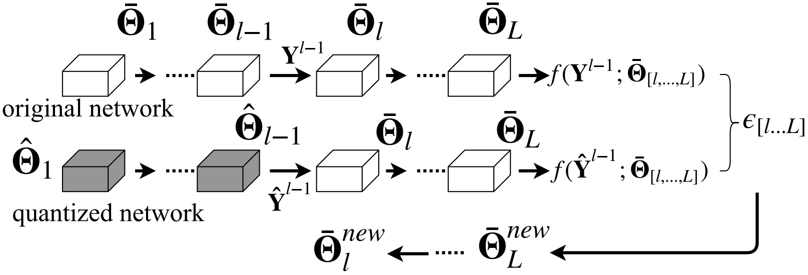
Highlight

- **L**ayer-wise/**L**imited Training Data **D**eep **N**eural Network **Q**uantization (L-DNQ).
- For each layer, parameters are quantized while the layer output is similar to that of the original full-precision parameters.
- Layer-wise quantization is formulated as a discrete quadratic optimization problem, with efficient solution.

NANYANG TECHNOLOGICAL UNIVERSITY

Outline

Workflow of Cascade L-DNQ



- $\hat{\Theta}$: Quantized weight
- $\hat{Y}$: Quantized output
- Cascade Quantization

Objective:

$$\min_{\hat{\Theta}_{[l,\dots,L]}} ||f(\hat{\mathbf{Y}}^{l-1}; \hat{\Theta}_{[l,\dots,L]}) - f(\mathbf{Y}^{l-1}; \bar{\Theta}_{[l,\dots,L]})||_F^2, \quad (1)$$

$$\text{s.t.} \quad \hat{\Theta}_{[l,\dots,L]} \in \mathbf{\Omega}_{[l,\dots,L]}.$$

NANYANG TECHNOLOGICAL UNIVERSITY

Relaxation of Quantization

Directly solving the above problem is difficult as the inputs to quantized network ($\hat{\mathbf{Y}}^{l-1}$) and the reference network ($\mathbf{Y}^{l-1}$) are different:

$$
\begin{aligned}
&||f(\hat{\mathbf{Y}}^{l-1}; \hat{\mathbf{\Theta}}_{[l,\ldots,L]}) - f(\mathbf{Y}^{l-1}; \bar{\mathbf{\Theta}}_{[l,\ldots,L]})||_F^2 \\
&\quad \leq \underbrace{||f(\hat{\mathbf{Y}}^{l-1}; \hat{\mathbf{\Theta}}_{[l,\ldots,L]}) - f(\hat{\mathbf{Y}}^{l-1}; \bar{\mathbf{\Theta}}^{new}_{[l,\ldots,L]})||_F^2}_{\text{Quantization}} \\
&\quad + \underbrace{||f(\hat{\mathbf{Y}}^{l-1}; \bar{\mathbf{\Theta}}^{new}_{[l,\ldots,L]}) - f(\mathbf{Y}^{l-1}; \bar{\mathbf{\Theta}}_{[l,\ldots,L]})||_F^2}_{\text{Weights Update}}, \qquad (2)
\end{aligned}
$$

- Final quantization error is bounded by quantization error between an updated weights and approximation error after the weights are updated.

NANYANG TECHNOLOGICAL UNIVERSITY

Layer-Wise Quantization

- Minimize the discrepancy of layer output before and after quantization:

$$E^l = E(\hat{\mathbf{Z}}^l) = \frac{1}{n}\left\|\hat{\mathbf{Z}}^l - \mathbf{Z}_*^l\right\|_F^2, \tag{3}$$

- Cascade input: $\hat{\mathbf{Y}}^{l-1}$, quantized weight: $\hat{\boldsymbol{\Theta}}_l$, updated weights:$\bar{\boldsymbol{\Theta}}_l^{new}$.
- After quantized output: $\hat{\mathbf{Z}}^l = \hat{\boldsymbol{\Theta}}_l^\top \hat{\mathbf{Y}}^{l-1}$.
- Before quantization output: $\mathbf{Z}_*^l = (\bar{\boldsymbol{\Theta}}_l^{new})^\top \hat{\mathbf{Y}}^{l-1}$.

$$\begin{aligned} E^l &= E(\hat{\mathbf{Z}}^l) - E(\mathbf{Z}_*^l) \\ &= \underbrace{\left(\frac{\partial E^l}{\partial \boldsymbol{\Theta}_l}\right)^\top \delta\boldsymbol{\Theta}_l}_{\frac{\partial E^l}{\partial \boldsymbol{\Theta}_l}\big|_{\boldsymbol{\Theta}_l = \bar{\boldsymbol{\Theta}}_l^{new}} = \mathbf{0}} + \frac{1}{2}\delta\boldsymbol{\Theta}_l^\top \mathbf{H}_l \delta\boldsymbol{\Theta}_l + \underbrace{O(\|\delta\boldsymbol{\Theta}_l\|_2^3)}_{\text{vanish}}, \end{aligned} \tag{4}$$

NANYANG TECHNOLOGICAL UNIVERSITY

Layer-Wise Quantization (Cont.)

By replacing $\delta\boldsymbol{\Theta}_l$ with $\hat{\boldsymbol{\Theta}}_l - \bar{\boldsymbol{\Theta}}_l^{new}$, the final objective becomes:

$$\min_{\hat{\boldsymbol{\Theta}}_l} f(\hat{\boldsymbol{\Theta}}_l) = \tfrac{1}{2}(\hat{\boldsymbol{\Theta}}_l - \bar{\boldsymbol{\Theta}}_l^{new})^\top \mathbf{H}_l(\hat{\boldsymbol{\Theta}}_l - \bar{\boldsymbol{\Theta}}_l^{new}), \tag{5}$$

$$\text{s.t. } \hat{\boldsymbol{\Theta}}_l \in \boldsymbol{\Omega}_{\boldsymbol{l}},$$

- $\boldsymbol{\Omega}_l$ is a discrete set of all possible values of the quantized weights in layer l.
- To solve (5), which is a **discrete optimization problem**, we develop a ADMM-based algorithm:

$$\min_{\hat{\boldsymbol{\Theta}},\mathbf{G}} f(\hat{\boldsymbol{\Theta}}) + I_{\boldsymbol{\Omega}}(\mathbf{G}), \text{ s.t. } \hat{\boldsymbol{\Theta}} = \mathbf{G}, \tag{6}$$

By incorporating $\mathbf{G}$ and introduce $\boldsymbol{\lambda}$, ρ:

$$\begin{aligned} &L_\rho(\hat{\boldsymbol{\Theta}}, \mathbf{G}, \boldsymbol{\lambda}) \\ &= f(\hat{\boldsymbol{\Theta}}) + I_{\boldsymbol{\Omega}}(\mathbf{G}) + \frac{\rho}{2}\|\hat{\boldsymbol{\Theta}} - \mathbf{G} + \boldsymbol{\lambda}\|_2^2 - \frac{\rho}{2}\|\boldsymbol{\lambda}\|_2^2. \end{aligned} \tag{7}$$

Quantization via ADMM – Proximal

At iteration $k+1$, the proximal step involves the update on $\hat{\mathbf{\Theta}}$ via

$$\hat{\mathbf{\Theta}}^{k+1} = \arg\min_{\hat{\mathbf{\Theta}}} L_\rho(\hat{\mathbf{\Theta}}, \mathbf{G}^k, \boldsymbol{\lambda}^k), \tag{8}$$

where

$$L_\rho(\hat{\mathbf{\Theta}}, \mathbf{G}^k, \boldsymbol{\lambda}^k) = f(\hat{\mathbf{\Theta}}) + \frac{\rho}{2}\|\hat{\mathbf{\Theta}} - \mathbf{G}^k + \boldsymbol{\lambda}^k\|_2^2. \tag{9}$$

Since $f(\hat{\mathbf{\Theta}})$ is a quadric function with continuous variable, setting the gradient to $\mathbf{0}$ leads to the optimal solution by solving the following linear equation:

$$\big(\mathbf{H} + \mathrm{diag}(\rho)\big)\hat{\mathbf{\Theta}}^{\boldsymbol{k+1}} = \mathbf{H}\bar{\mathbf{\Theta}}^{new} + \mathrm{diag}(\rho)(\mathbf{G}^k - \boldsymbol{\lambda}^k). \tag{10}$$

Quantization via ADMM – Projection

In projection step, we optimize $\mathbf{G}$ by solving the following optimization problem:

$$\min_{\mathbf{G}} \|\hat{\mathbf{\Theta}}^{k+1} - \mathbf{G} + \boldsymbol{\lambda}^k\|_2^2, \text{ s.t. } \mathbf{G} \in \mathbf{\Omega}. \tag{11}$$

- $\mathbf{V}^k = \hat{\mathbf{\Theta}}^{k+1} + \boldsymbol{\lambda}^k$, $\mathbf{G} = g(\alpha, \mathbf{Q}) = \alpha \cdot \mathbf{Q}$:

$$\min_{\mathbf{G},\alpha} \|\mathbf{V}^k - \alpha \cdot \mathbf{Q}\|_2^2, \text{ s.t. } \mathbf{Q} \in \{-1, 0, 1\}, \tag{12}$$

- Iterative solution:
  - $\alpha = \frac{\mathbf{V}^\top \mathbf{Q}}{\mathbf{Q}^\top \mathbf{Q}}$
  - $\mathbf{Q} = \text{Proj}_{\{-1,0,1\}}\left(\frac{\mathbf{V}^k}{\alpha}\right)$

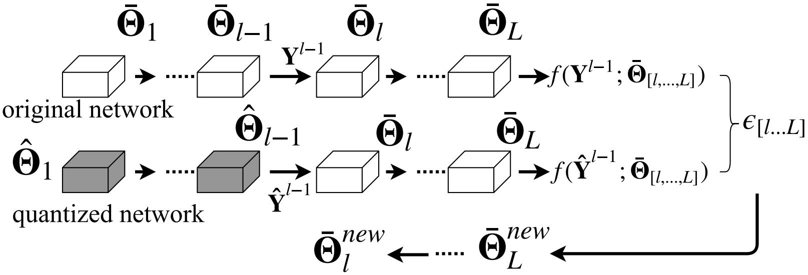
NANYANG TECHNOLOGICAL UNIVERSITY

Quantization via ADMM – Dual Update

After obtaining $\hat{\mathbf{\Theta}}^{\boldsymbol{k+1}}$ and $\mathbf{G}^{k+1}$, the dual variable $\boldsymbol{\lambda}$ is updated using the following rule:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^{k} + \hat{\mathbf{\Theta}}^{k+1} - \mathbf{G}^{k+1}. \tag{13}$$

Cascade Weights Update



- $\epsilon_{[l...L]} = \| f(\hat{\mathbf{Y}}^{l-1}; \bar{\mathbf{\Theta}}^{new}_{[l,...,L]}) - f(\mathbf{Y}^{l-1}; \bar{\mathbf{\Theta}}_{[l,...,L]}) \|_F^2$.
- $f(\mathbf{Y}^{l-1}; \bar{\mathbf{\Theta}}_{[l,...,L]})$ as groundtruth.
- Learn $\bar{\mathbf{\Theta}}^{new}_{[l,...,L]}$.

Outline

Experiments

Our method (L-DNQ) compares with the following baselines using CIFAR10 and ImageNet dataset:

- Training based – with data:
  - Extremely Low Bit Neural Network (ExNN) [4]
  - Trained Ternary Quantization (TTQ) [7]
  - Incremental Network Quantization (INQ) [6]
  - Loss-Aware weight Ternarized network (LAT) [2]
  - Model compression via distillation and quantization (DistilQuant) [5]
- Direct quantization – without data:
  - Compressing Deep Convolutional Networks using Vector Quantization (VQ) [1]
  - Direct Quantization (DQ) [3]

For training-based methods, we reimplemented it using limited training data (1%). For direct quantization, we fine-tune the un-quantized weights using limited training data (1%).

NANYANG TECHNOLOGICAL UNIVERSITY

Comparison in CIFAR10

Network	Method	bits	Imp* (%)	FP Acc**
ResNet20	TTQ	3	-77.25	91.77
	INQ	15	-48.48	90.02
	ExNN	3	-11.15	91.5
	VQ	3	-11.27	
	DQ	3	-19.92	
	L-DNQ	3	**-4.30**	
CIFARNet	LAT	3	-11.62	89.62
	VQ	3	-11.83	92.27
	DQ	3	-21.72	
	L-DNQ	3	**-1.96**	
WRN	DistilQuant	3	-6.57	92.25
	L-DNQ	3	**-2.22**	91.43

Table 1: Comparison on CIFAR-10. All methods use 1% (500 images) of training instances. * indicates improvement. ** represents **F**ull **P**recision (pre-trained model) Accuracy.

Comparison in ImageNet

Network	Method	bits	Improvement(%)	FP Accuracy
ResNet18	TTQ	3	-69.48/-88.49	69.6/89.2
	INQ	15	-61.27/-64.22	68.27/88.69
	ExNN	3	-43.53/-37.82	69.76/89.02
	VQ	3	-35.69/-29.08	
	DQ	3	-61.22/-65.64	
	L-DNQ	3	**-16.43/-10.67**	
	DQ	8	-56.78/-58.92	
		16	-13.81/-8.68	
		32	-2.82/-1.51	
	L-DNQ	9	**-2.73/-0.90**	
ResNet34	DistilQuant	3	-32.03/24.3	56.55/79.09
	L-DNQ	3	**-29.31/18.37**	

Table 2: Comparison on ImageNet. All methods use 1% (12,800 images) training instances.

Effect of Number of Training Data

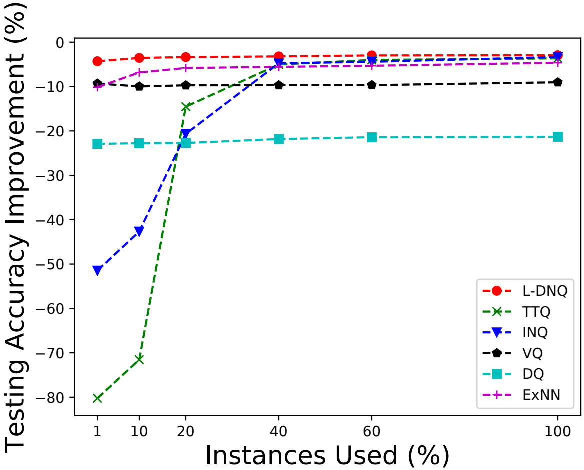


Figure 3: Performance among L-DNQ, ExNN, TTQ, INQ, VQ, DQ using ResNet20 in CIFAR10 with increasing instances. X-axis presents portion of training data used, Y-axis represents performance improvement after quantization (Higher the better).

Layer Output Error V.S. Performance

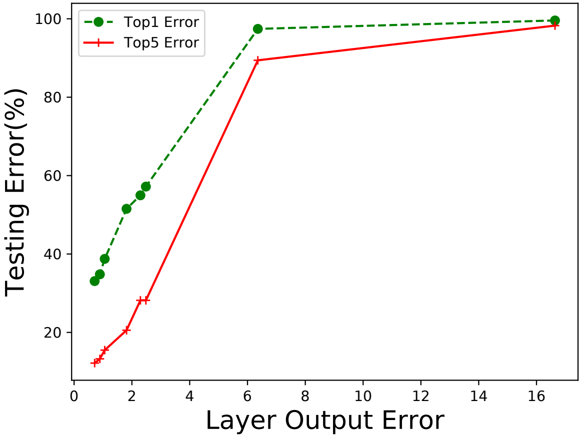


Figure 4: ImageNet in ResNet18: Performance under different layer output error. X-axis presents final layer output error, Y-axis represents testing error after quantization (the lower the better).

NANYANG TECHNOLOGICAL UNIVERSITY

Outline

Conclusion

- A novel layer-wise quantization framework: it is able to quantize deep models without big performance drop using only **limited training data**.
- Layer-wise quantization is formulated as discrete optimization problem.
- Cascade weights update is utilized to minimize discrepancy.

References I

[1] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[2] Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. 2018.

[3] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *arXiv preprint arXiv:1712.05877*, 2017.

[4] Cong Leng, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *AAAI*. 2017.

[5] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. 2018.

NANYANG TECHNOLOGICAL UNIVERSITY

References II

[6] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[7] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

NANYANG TECHNOLOGICAL UNIVERSITY