MetaQuant: Learning to Quantize by Learning to Penetrate Non-differentiable Quantization

Shangyu Chen (schen025@e.ntu.edu.sg), Wenya Wang (wangwy@ntu.edu.sg), Sinno Jialin Pan(sinnopan@ntu.edu.sg)
School of Computer Science and Engineering, Nanyang Technological University, Singapore

Motivation

Existing training-based quantization methods reply on Straight-Through-Estimator (STE) to enable **non-differentiable** quantization training:

$$\ell = \text{Loss}(f(Q(\mathbf{W}); \mathbf{x}), y), \qquad \frac{\partial Q(r)}{\partial r} = \begin{cases} 1 & \text{if} \quad |r| \le 1 \\ 0 & \text{else.} \end{cases} \tag{1}$$

However, it inevitably brings the problem:

- Gradient Mismatch: the gradients of the weights are not generated using the value of weights, but rather its quantized value.
- Poor Gradient: STE fails at investigating better gradients for quantization training.

We propose to learn $\frac{\partial Q(r)}{\partial r}$ by a neural network ($\mathcal{M}$) during quantization training. Such neural network is called *meta quantizer* and is trained together with the base quantized model.

Contribution

- Proper gradients is learned in quantization training without any manually designed knowledge.
- Generated gradient is loss-aware, contributing to the training of base model.
- Meta quantizer is removed and consumes no space in deployment.

Problem Statement

- Pre-trained full-precision base model f with L-layer parameterized as $\mathbf{W} = [\mathbf{W}_1, ..., \mathbf{W}_L]$.
- Pre-processing function $\mathcal{A}(\cdot)$, pre-quantized weights: $\tilde{\mathbf{W}}$.
- Quantization function $Q(\cdot)$, quantized weights: $\hat{\mathbf{W}}$.
- Forward quantization:

$$\textbf{dorefa}: \ \tilde{\mathbf{W}} = \mathcal{A}(\mathbf{W}) = \frac{\tanh(\mathbf{W})}{2\max(|\tanh(\mathbf{W})|)} + \frac{1}{2}, \ \hat{\mathbf{W}} = Q(\tilde{\mathbf{W}}) = 2\frac{\text{round}\left[(2^k-1)\tilde{\mathbf{W}}\right]}{2^k-1} - 1.$$

$$\textbf{BWN}: \ \tilde{\mathbf{W}} = \mathcal{A}(\mathbf{W}) = \mathbf{W}, \qquad \hat{\mathbf{W}} = Q(\tilde{\mathbf{W}}) = \frac{1}{n}||\tilde{\mathbf{W}}||_{l_1} \times \text{sign}(\tilde{\mathbf{W}}).$$

Generation of Meta Gradient

- Gradient of quantized weights: $g_{\hat{\mathbf{W}}} = \frac{\partial \ell}{\partial \hat{\mathbf{W}}}$.
- Meta Gradient / Gradient of pre-quantized weights:

$$g_{\tilde{\mathbf{W}}} = \mathcal{M}_\phi(g_{\hat{\mathbf{W}}}, \tilde{\mathbf{W}}) \tag{2}$$

- Gradient of full-precision weights:

$$g_{\mathbf{W}} = \frac{\partial \ell}{\partial \tilde{\mathbf{W}}}\frac{\partial \tilde{\mathbf{W}}}{\partial \mathbf{W}} = g_{\tilde{\mathbf{W}}}\frac{\partial \tilde{\mathbf{W}}}{\partial \mathbf{W}} = \mathcal{M}_\phi(g_{\hat{\mathbf{W}}}, \tilde{\mathbf{W}})\frac{\partial \tilde{\mathbf{W}}}{\partial \mathbf{W}} \tag{3}$$

- Calibration: **dorefa**: $\frac{\partial \tilde{\mathbf{W}}}{\partial \mathbf{W}} = \frac{1-\tanh^2(\mathbf{W})}{\max(|\tanh(\mathbf{W})|)}$, **BWN**: $\frac{\partial \tilde{\mathbf{W}}}{\partial \mathbf{W}} = \mathbf{1}$
- Gradient Refinement: **SGD**: $\pi(g_{\mathbf{W}}) = g_{\mathbf{W}}$, **Adam**: $\pi(g_{\mathbf{W}}) = g_{\mathbf{W}} + \text{residual}$
- Update of full-precision:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \alpha\pi(g^t_{\mathbf{W}}) \tag{4}$$

Overflow of MetaQuant

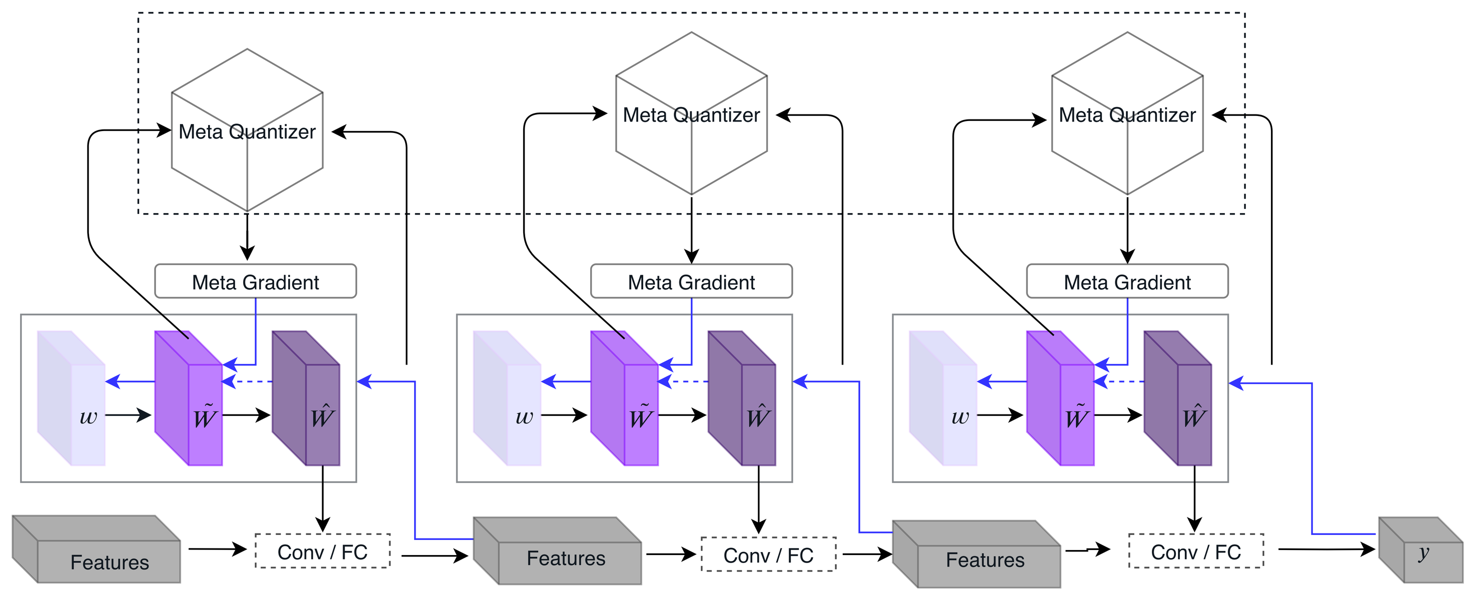


Fig. 1: Overflow of MetaQuant.

Training of Meta Quantizer

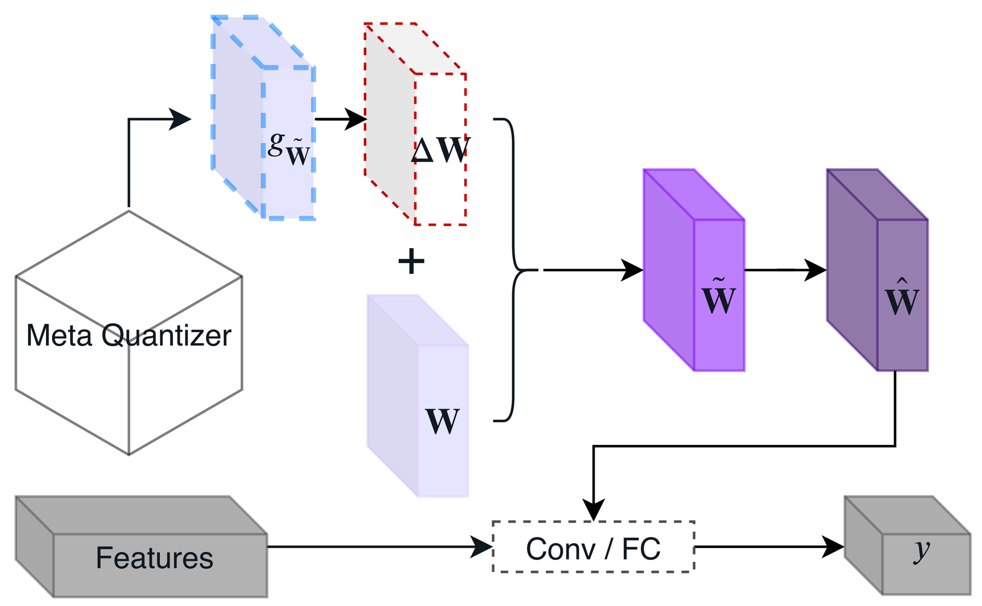


Fig. 2: Incorporation of meta quantizer into quantization training. Red dash box is composed of calibration, gradient refinement and multiplication of learning rate α. Output of meta quantizer is involved in W's update and contributes to final loss, constructing a differential path from loss to ϕ-parameterized meta quantizer.

- Meta quantizer is a **coordinate-wise** neural network: each weight in the base model is processed independently.
- During each inference, the inputs in (4) are arranged as batches with size 1: $\mathbf{W}$ comes from a convolution layer with shape $\mathbb{R}^{o \times i \times k \times k} \rightarrow \mathbb{R}^{(o \times i \times k^2) \times 1}$.

$$\text{Forward:} \quad \tilde{\mathbf{W}}^t = \mathcal{A}(\mathbf{W}^t) = \mathcal{A}\left[\mathbf{W}^{t-1} - \alpha \times \pi(\mathcal{M}_\phi(g^{t-1}_{\hat{\mathbf{W}}}, \tilde{\mathbf{W}}^{t-1})\frac{\partial \tilde{\mathbf{W}}^{t-1}}{\partial \mathbf{W}^{t-1}})\right],$$

$$\ell = \text{Loss}\left\{f\left[Q(\tilde{\mathbf{W}}^t); \mathbf{x}\right], y\right\}, \tag{5}$$

$$\text{Backward:} \quad \frac{\partial L}{\partial \phi^t} = \frac{\partial L}{\partial \tilde{\mathbf{W}}^t}\frac{\partial \tilde{\mathbf{W}}^t}{\partial \phi^t} = \mathcal{M}_\phi(g_{\hat{\mathbf{W}}^t}, \tilde{\mathbf{W}}^t)\frac{\partial \tilde{\mathbf{W}}^t}{\partial \phi^t}. \tag{6}$$

Design of Meta Quantizer

$$\begin{aligned} \texttt{FCGrad}: & \quad \mathcal{M}_\phi(g_{\hat{\mathbf{W}}}) = \text{FCs}(\phi, \sigma, g_{\hat{\mathbf{W}}}), \\ \texttt{MultiFC}: & \quad \mathcal{M}_\phi(g_{\hat{\mathbf{W}}}, \tilde{\mathbf{W}}) = g_{\hat{\mathbf{W}}} \cdot \text{FCs}(\phi, \sigma, \tilde{\mathbf{W}}). \\ \texttt{LSTMFC}: & \quad \mathcal{M}_\phi(g_{\hat{\mathbf{W}}}, \tilde{\mathbf{W}}) = g_{\hat{\mathbf{W}}} \cdot \text{FCs}(\phi_{FCs}, \sigma, (\text{LSTM}(\phi_{LSTM}, \tilde{\mathbf{W}}))). \end{aligned}$$

ϕ: parameters in meta quantizer, σ: nonlinear activation.

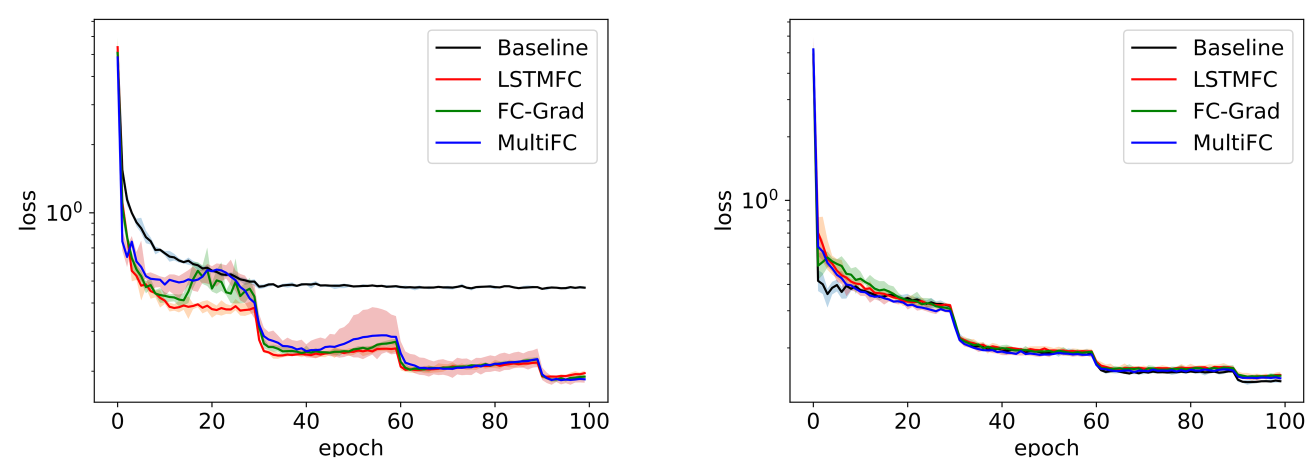
Experiment

Network-Dataset	Forward	Backward	Optimization	Test Acc (%)	FP Acc (%)
ResNet20-CIFAR10	dorefa	STE	SGD	80.745(2.113)	91.5
		MultiFC		**88.942(0.466)**	
		LSTMFC		88.305(0.810)	
		FCGrad		88.840(0.291)	
		STE	Adam	89.782(0.172)	
		MultiFC		89.941(0.068)	
		LSTMFC		**89.979(0.103)**	
		FCGrad		89.962(0.068)	
	BWN	STE	SGD	75.913(3.495)	
		LSTMFC		**89.289(0.212)**	
		FCGrad		88.949(0.231)	
		STE	Adam	89.896(0.182)	
		LSTMFC		90.036(0.109)	
		FCGrad		**90.042(0.098)**	
ResNet56-CIFAR100	dorefa	STE	SGD	42.265(8.143)	71.22
		MultiFC		**65.791(0.415)**	
		LSTMFC		63.645(2.183)	
		FCGrad		64.351(0.935)	
		STE	Adam	66.419(0.533)	
		MultiFC		**66.588(0.375)**	
		LSTMFC		66.483(0.793)	
		FCGrad		66.564(0.351)	
	BWN	STE	SGD	34.479(11.737)	
		LSTMFC		63.346(2.253)	
		FCGrad		**64.402(1.434)**	
		STE	Adam	64.297(1.309)	
		LSTMFC		66.584(0.349)	
		FCGrad		**67.018(0.329)**	
ResNet18-ImageNet	dorefa	STE	Adam	58.349(2.072)/81.477(1.567)	69.76/89.08
		MultiFC		59.472(0.025)/82.410(0.010)	
		FCGrad		**59.835(0.359)/82.671(0.232)**	
	BWN	STE		59.503(0.835)/82.549(0.506)	
		FCGrad		**60.328(0.391)/83.025(0.234)**	

Fig. 3: Overall experiments of MetaQuant.

Network	Method	Acc Drop (%)	Network	Method	Acc Drop (%)
ResNet20	ProxQuant	1.29	ResNet32	ProxQuant	1.28
	MetaQuant	**0.7**		MetaQuant	**0.39**
ResNet44	ProxQuant	0.99	LABNet	LAB	1.4
	MetaQuant	**0.08**		MetaQuant	**-0.2**
ResNet18	ELQ	**3.55/2.65**	ResNet18-2bits	TTQ	**3.00/2.00**
	MetaQuant	6.32/4.31		MetaQuant	5.17/3.59

Fig. 4: Experimental result of MetaQuant V.S Non-STE training-based quantization: ProxQuant, LAB, ELQ, TTQ.



(a) SGD (b) Adam

Fig. 5: Convergence analysis of MetaQuant using ResNet20, CIFAR10, dorefa, SGD/Adam.

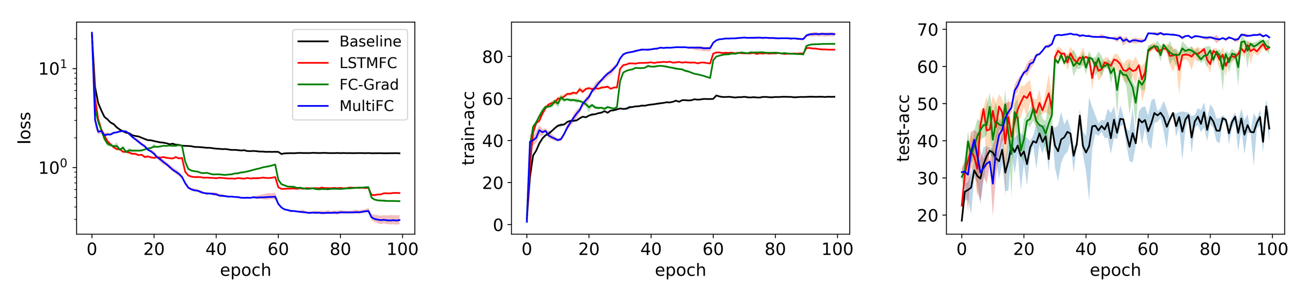


Fig. 6: Overall training of MetaQuant using ResNet110, CIFAR100, dorefa, SGD.