OPTIMIZING QUANTIZED NEURAL NETWORKS IN A WEAK CURVATURE MANIFOLD

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantized Neural Networks (QNNs) have achieved an enormous step in improving computational efficiency, making it possible to deploy large models to mobile and miniaturized devices. In order to narrow the performance gap between low-precision and full-precision models, we introduce the natural gradient to train a low-precision model by viewing the parameter space as a Riemannian manifold. Specifically, we propose a novel Optimized Natural Gradient Descent (ONGD) method defined by the Hyperbolic divergence, which provides a perspective to calculate the optimized natural gradient in weak curvature and updates the parameters with an amount of computation comparable to Stochastic Gradient Descent (SGD). We conduct an ablation study and results show that the 4-bit quantized ResNet-32 trained with ONGD has a better result than SGD, i.e. 2.05% higher in Top-1 accuracy on CIFAR100 dataset. Further comparison experiments illustrate that our method achieves state-of-the-art results in CIFAR and ImageNet datasets, where the 8-bit version of MobileNet achieves 0.25%/0.13% higher in Top-1/Top-5 accuracies than the full-precision version on ImageNet dataset.

1 INTRODUCTION

Neural networks can handle many complex tasks due to their large number of trainable parameters and strong nonlinear capabilities. However, the massive amount of models and calculations hinders the application of neural networks on mobile and miniaturized devices, which naturally comes with constraints on computing power and resources. Neural network quantization is considered to be an efficient solution in the inference that alleviates the number of parameters and optimizes the computation by reducing the bit width of weights and activations.

Neural network quantization can be roughly divided into two categories: forced quantization and relaxed quantization. Most of the quantization methods adopted by the QNNs belong to the former, i.e. enforcing discretization during training. For instance, XNOR-Net (Rastegari et al., 2016) quantized the filters and the inputs of neural network to binary, which can use efficient binary operations. DeepShift (Elhoushi et al., 2019) replaced all multiplications with bitwise shift and sign flipping by representing the weights as 6-bit. Besides, INT8 (Zhu et al., 2020) had achieved the quantized gradients according to the distinctive characteristics of gradients. Moreover, MeliusNet (Bethge et al., 2020) proposed a mixed-precision method between 32-bit and 1-bit by adding dense block and improvement block into the network structure. Other relaxed quantizations maintain partial discretization during training, e.g. INQ (Zhou et al., 2017) divided the weights into two groups until all parameters are quantized, where the first group is directly quantized and fixed; the second group needs to be retrained to make up for the decrease of accuracy caused by quantization of the first group. RQ (Louizos et al., 2019) introduced a differentiable quantizer that can transform the continuous distributions of weights and activations to categorical distributions with gradient-based optimization. Recently, AQE (Chen et al., 2020) proposed a novel asymptotic-quantized estimator to discretize the weights and activations gradually. However, these methods all use general gradients for training, which lacks the model-level information. In this paper, we introduce the natural gradient to train a low-precision model that considers the model's curvature information, as shown the red line in Figure 1.

For neural networks with a scale of one million or more parameters, the time complexity of inverting Fisher Information Matrix (FIM), a component of natural gradients, is $O(n^3)$ (Povey et al., 2014).

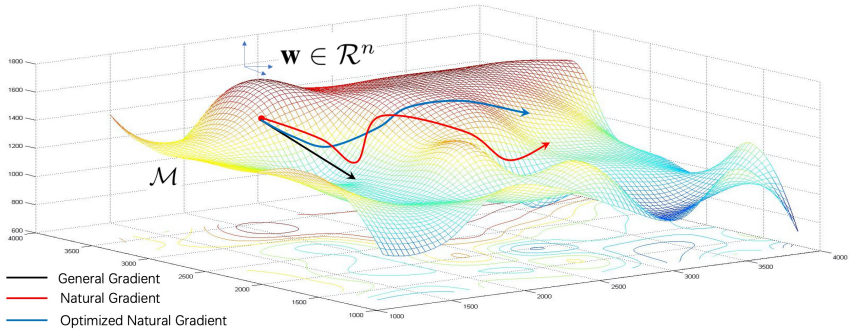


Figure 1: The general gradient can be intuitively expressed by the black line in Euclidean space. Suppose the parameter space is changed to a Riemannian manifold. In that case, the natural gradient can be described by the red line (Amari, 2016) in arbitrary curvature. In weak curvature, we can visualize the optimized natural gradient with the blue line.

Prior to this, there were some works to calculate the natural gradient efficiently, e.g. (Roux et al., 2008) decomposed the FIM into multiple diagonal blocks, where each diagonal block is approximated by a low-rank matrix. (Bastian et al., 2011) also used the idea of diagonal blocks by constructing a diagonal block that corresponds to a weight matrix. (Martens & Grosse, 2015) proposed to approximate FIM through the Kronecker product of two smaller matrices to improve computational efficiency. Even so, the complex decomposition methods are still not suitable for large-scale tasks, and the amount of computation is large compared to general gradient. We propose a novel ONGD method in weak curvature with constraint Hyperbolic divergence, i.e. the blue line in Figure 1, whose calculated amount is close to general gradient.

In this paper, our purpose is to introduce a novel method of training QNNs that bypasses SGD to achieve better performance. The main contributions of this work are three-fold: **First**, we show how to calculate the natural gradient by constructing a Riemannian metric or FIM in a low-precision model. **Second**, we define a novel Hyperbolic divergence by a convex function with geometric structure. With constraint Hyperbolic divergence, we introduce an efficient method ONGD for computing optimized natural gradient in weak curvature. Naturally, forced quantization is transformed into smooth quantization by establishing a geometric constraint in low-precision parameter space. **Third**, we reveal that the essence of two-way distillation between low-precision and full-precision models is a synchronization phenomenon, and fine-tune the low-precision model on this basis.

2 Related Work

2.1 General Gradient

For a convolutional neural network, the full-precision weight tensor for all layers is just marked as $\mathbf{w}$. Let $\mathcal{M} = \{\mathbf{w} \in \mathcal{R}^n\}$ be a parameter space on which a loss function L associated with weight is well-defined. It is relatively easy to express the general gradient in training:

$$\nabla_{\mathbf{w}} L = \frac{\partial L}{\partial \mathbf{w}}, \tag{1}$$

which is the steepest descent method when $\mathcal{M}$ is considered Euclidean space with an orthonormal coordinate. Note that the negative gradient represents the direction of the steepest descent inherently.

When Euclidean space is considered, Euclidean divergence between two sufficiently closed points w and w' is actually defined by default:

$$D_E[w : w'] = \frac{1}{2} \sum (w - w')^2, \tag{2}$$

which is a half of the square of the Euclidean distance identified by Euclidean metric δ_{ij}, so that

$$|dw|^2 = 2D_E[w : w + dw] = \sum (dw^i)^2 = \sum_{i,j} \delta_{ij} dw^i dw^j. \tag{3}$$

2.2 NATURAL GRADIENT

The general gradient descent method only considers the parameter update along the gradient direction that does not involve the model-level information, which may cause uneven neural network update. By treating the parameter space $\mathcal{M}$ as a Riemannian manifold (Amari, 1998), rough optimization is alleviated through the local curvature information distributed in the parameter space.

In that case, the steepest descent direction also depends on the quadratic form introduced by a small incremental row vector dw that connects w and $w + dw$, whose form is given by

$$|dw|^2 = \sum_{i,j} G_{ij}(\mathbf{w}) dw^i dw^j, \tag{4}$$

where $G_{ij}(\mathbf{w})$ is a Riemannian metric, and dw^i is the component of dw. Under the constraint $|dw|^2$, the steepest descent direction is toward the optimization goal of $L(w + dw)$. Intuitively, it measures the Kullback-Leibler (KL) divergence between two distributions $\psi(w), \psi(w + dw)$ of network model, which is equivalent to the interval of two adjacent points on a Riemannian manifold. The KL divergence under the Riemannian metric is well approximated through the FIM with the second-order Taylor expansion of the KL divergence (see Appendix 6.4) (Ba et al., 2016):

$$|dw|^2 = 2D_{KL}[\psi(w) : \psi(w + dw)] \approx -dw\mathbb{E}_{\psi_w}[\nabla^2 \log \psi(w)]dw^\top = dwF_{ij}(\mathbf{w})dw^\top. \tag{5}$$

Amari deduced that the Riemannian metric is given by the Hessian of $\psi(w)$: $G_{ij}(\mathbf{w}) = \nabla^2 \log \psi(w)$ (Amari, 1998). By the Lagrangian form, we have

$$\tilde{\nabla}_{\mathbf{W}} L = F^{-1}(\mathbf{w}) \frac{\partial L}{\partial \mathbf{W}} = -\mathbb{E}_{\psi_\mathbf{w}}[G(\mathbf{w})]^{-1} \frac{\partial L}{\partial \mathbf{W}}, \text{where } F(\mathbf{w}) = \mathbb{E}_{\psi_w}[\nabla \log \psi(w) \nabla \log \psi(w)], \tag{6}$$

where $\tilde{\nabla}_{\mathbf{W}} L$ is the natural gradient that is the steepest descent method in a Riemannian space, which is why the natural gradient can also be called the Riemannian gradient. Note that $\tilde{\nabla}_{\mathbf{W}} L$ will return to $\nabla_{\mathbf{W}} L$ when $G_{ij}(\mathbf{w})$ is equal to the Euclidean metric δ_{ij}.

2.3 NETWORK QUANTIZATION

Considering a quantized model, we notate the low-precision weight tensor using $\hat{\mathbf{w}}$ for differentiating full-precision weight $\mathbf{w}$. During training a QNN, the forced quantizer $Q(\cdot)$ is a many-to-one mapping $Q : \mathbf{w} \in \mathcal{R}^n \mapsto \hat{\mathbf{w}} \in \mathcal{R}^n$ that can be expressed as

$$\hat{w}_i = Q(w_i), \tag{7}$$

where the weight vector w_i or $\hat{w}_i$ straightens the i-th layer of the weight tensor $\mathbf{w}$ or $\hat{\mathbf{w}}$. Quantized models bring new challenges that have caught the disappearance or explosion of the gradients arisen with the staircase character of quantizers during back-propagation. Fortunately, this issue can always be solved by Straight-Through Estimator (STE) (Hinton, 2012; Courbariaux et al., 2016):

$$\nabla_{w_i} L = \nabla_{\hat{w}_i} L \circ \mathbb{I}_{|w_i| \leq 1}, \tag{8}$$

where $\circ$ is the Hadamard product, and $\mathbb{I}$ is the indicator function.

As the parameter space $\mathcal{M}$ has been assumed to be a flat and orthogonal Euclidean space, the parameters are updated along the general gradient described as the black line in Figure 1. This simply discretization of the neural network applied to the rounding quantizer leads us to add STE into SGD. Except for the application of STE and quantizers, there is no difference between low-precision and full-precision models in training.

3 THE GEOMETRY OF QUANTIZED MODELS

In this section, we will describe in detail how to optimize the training of low-precision models using geometric methods. Then we elaborate the principle and process in the subsequent.

3.1 NETWORK QUANTIZATION WITH NATURAL GRADIENT

By involving the practice to train a neural network with low bit-width, the process of the quantization needs to be further designed that plays a vital role in the final performance, as for the other parts, e.g. via mapping from an input pattern $\hat{a}_{i-1}$ to output $\hat{a}_i$ can still be imitated in the same way as full-precision neural networks:

$$s_i = \hat{w}_i \hat{a}_{i-1}, \hat{a}_i = Q(f_i \circ s_i), \tag{9}$$

where f_i is non-linear function acted on element-wise. To distinguish the full-precision model, we mark the notation " $\hat{}$ " to represent the operation through a quantizer in a low-precision model, whether for weight w or activation a.

The concept of the natural gradient is closely related to FIM and KL divergence (see Appendix 6.4). Since KL divergence is intrinsic, the natural gradient is also intrinsic that remains unchanged under the parameter transformation. By viewing FIM as an l by l block matrix where l denotes the number of layers in neural network, the natural gradient formula can be introduced when training a QNN and updating its parameters (Martens & Grosse, 2015):

$$\begin{aligned}
\tilde{\nabla}_{\mathbf{W}} L &\overset{\text{STE}}{=} F_{ij}^{-1}(\hat{\mathbf{w}}) \frac{\partial L}{\partial \hat{\mathbf{W}}} \circ \mathbb{I}_{|\mathbf{W}|\leq 1} = \mathbb{E}\left[\frac{\partial \log \psi(\hat{w})}{\partial \hat{w}_i} \frac{\partial \log \psi(\hat{w})}{\partial \hat{w}_j}\right]^{-1} \frac{\partial L}{\partial \hat{\mathbf{W}}} \circ \mathbb{I}_{|\mathbf{W}|\leq 1} \\
&= \mathbb{E}\left[\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_i}) \operatorname{vec}(\frac{\partial L}{\partial \hat{w}_j})^\top\right]^{-1} \frac{\partial L}{\partial \hat{\mathbf{W}}} \circ \mathbb{I}_{|\mathbf{W}|\leq 1} \\
&= \begin{bmatrix} \mathbb{E}\left[\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_1}) \operatorname{vec}(\frac{\partial L)}{\partial \hat{w}_1})^\top\right] & \cdots & \mathbb{E}\left[\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_1}) \operatorname{vec}(\frac{\partial L}{\partial \hat{w}_l})^\top\right] \\ \vdots & \ddots & \vdots \\ \mathbb{E}\left[\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_l}) \operatorname{vec}(\frac{\partial L}{\partial \hat{w}_1})^\top\right] & \cdots & \mathbb{E}\left[\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_l}) \operatorname{vec}(\frac{\partial L}{\partial \hat{w}_l})^\top\right] \end{bmatrix}^{-1} \frac{\partial L}{\partial \hat{\mathbf{W}}} \circ \mathbb{I}_{|\mathbf{W}|\leq 1},
\end{aligned} \tag{10}$$

where $\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_i})$ can be represented by the gradient error $\frac{\partial L}{\partial \hat{a}_i}$ [1]. Considering that the gradient propagation needs to span over the quantized neurons and graphs, we still need to use STE (see Appendix 6.1) to update the gradient of QNNs in the learning procedure:

$$\operatorname{vec}(\frac{\partial L}{\partial \hat{w}_i}) = \hat{a}_{i-1}^\top \otimes \left(\frac{\partial L}{\partial a_i} \circ f_i'(s_i)\right) \overset{\text{STE}}{=} \hat{a}_{i-1}^\top \otimes \left(\frac{\partial L}{\partial \hat{a}_i} \circ \mathbb{I}_{|a_i|\leq 1} \circ f_i'(s_i)\right), \tag{11}$$

where $\otimes$ denotes the Kronecker product [2]. Relying on the KL divergence, we indicate the parameter space as Riemannian manifold rather than Euclidean space in the quantization procedure.

3.2 WEAK CURVATURE CONDITION

The reason why we can use Euclidean coordinates to calculate the natural gradient on the manifold is the local homeomorphism of a manifold based on **Definition** 1 (Wald, 2010). Note that the homeomorphic mapping ϕ_U satisfies $U \in \mathcal{M} \mapsto \phi_U(U) \in \mathcal{R}^n$. For any point $x \in U$, we can define $\phi_U(x)$ as the Euclidean coordinate absolutely. The natural gradient calculated here is obviously the result of the homeomorphic mapping.

Definition 1 *Let $\mathcal{M}$ be an n-dimensional manifold, for any point $x \in \mathcal{M}$, and then it satisfies that there exists a neighborhood U of x in $\mathcal{M}$ which is homeomorphic to an open set in an n-dimensional Euclidean space $\mathcal{R}^n$.*

The natural gradient poses a significant challenge for computing. Intuitively, the computing $F_{ij}^{-1}(\hat{\mathbf{w}})$ attached to the natural gradient will take on massive computation, which can only be numerically estimated. Mostly, this inversion is entirely unrealistic, when deep neural networks are very redundant, with tens of thousands of the neural connections.

[1] Note that we use $\mathbf{W}$ or $\hat{\mathbf{W}}$ to denote a large matrix of $l \times n$ that is composed by each layer of weight tensor $\mathbf{w}$ or $\hat{\mathbf{w}}$ as a row vector, where n is the largest size of rows and other insufficient rows are filled with zero. The weight vector w_i or $\hat{w}_i$ is obtained by straightening i-th layer of the weight tensor $\mathbf{w}$ or $\hat{\mathbf{w}}$ into a row vector. The weight vector w or $\hat{w}$ is obtained by straightening the weight tensor $\mathbf{w}$ or $\hat{\mathbf{w}}$ into a row vector. The operator $\operatorname{vec}(\cdot)$ means to straighten a matrix or tensor into a row vector.

[2] Since the input $\hat{a}_{i-1}^\top$ and output $\frac{\partial L}{\partial a_i}$ sizes of each layer may be different, Kronecker product is necessary.

In this paper, we consider the optimized natural gradient under the condition of weak curvature in low-precision models. This means that the Riemannian manifold is nearly flat, where the Riemannian metric is an approximation of the Euclidean metric. In practice, we develop a linearized Riemannian metric from the Euclidean metric δ_{ij}, which is systematically defined in general relativity (Wald, 2010):

$$G_{ij} = \delta_{ij} + \epsilon_{ij}, \tag{12}$$

where the metric ϵ_{ij} is smaller than 1 in global Euclidean coordinate system of δ_{ij}. It is an adequate definition of "weakness" in this context and ensures to be a positive-definite metric. Combining eq. 6, FIM can be linearly decomposed in weak curvature [3]:

$$F_{ij}(\hat{\mathbf{w}}) = -\mathbb{E}_{\psi_{\hat{\mathbf{w}}}}[\delta_{ij} + \epsilon_{ij}] = -\delta_{ij} - \mathbb{E}_{\psi_{\hat{\mathbf{w}}}}[\epsilon_{ij}(\mathrm{diag}(\mathrm{vec}(\hat{\mathbf{w}})))]. \tag{13}$$

Since the complexity of FIM is determined by KL divergence in arbitrary curvature, we urgently need to develop a divergence defined in weak curvature to help simplify the calculation.

3.3 Optimized natural gradient in Hyperbolic divergence

In practice, we can determine a unique geodesic through exponential map $\exp_w(w - \hat{w})$ that maps $w - \hat{w}$ back to $\mathcal{M}$, by defining $w - \hat{w} \in T_w\mathcal{M}$ as the tangent vector. Note that the definition of exponential map is developed by (Wald, 2010; Helgason, 2001).

Definition 2 *Let $\mathcal{M}$ be a Riemannian manifold, for the tangent vector $v \in T_x\mathcal{M}$ in a point $x \in \mathcal{M}$ where $T_x\mathcal{M}$ is the tangent space, there is a unique geodesic $\gamma_v(t)$ locally that satisfies $\gamma_v(0) = x$ and $\gamma_v'(0) = v$. The exponential map $\exp_x : T_x\mathcal{M} \mapsto \mathcal{M}$ corresponding to $\gamma_v(t)$ is defined as $\exp_x(v) = \gamma_v(1)$. When constraining $\exp_x$ to a neighbourhood U, this mapping is one-to-one.*

The exponential map $\exp_w(w - \hat{w})$ is to map a tangent vector $(w - \hat{w})$ in the tangent bundle to the point where the arc length from point w is equal to $|w - \hat{w}|$ on geodesic with the initial condition $(w, w - \hat{w})$. In order to make $\exp(w - \hat{w})$ and $\exp(\hat{w} - w)$ have the same effect in the training process, we symmetrize it and derive a convex function

$$\psi(w - \hat{w}) = \log\left[\frac{\exp(w - \hat{w}) + \exp(\hat{w} - w)}{2}\right] = \log(\cosh(w - \hat{w})). \tag{14}$$

Geometrically, the convex function with geometric structure is introduced into Bregman divergence (see Appendix 6.3) (Bregman, 1967), and we obtain a novel Hyperbolic divergence that satisfies the criteria of divergence (see Appendix 6.2).

Definition 3 *For a convex function ψ defined by eq. 14 in a weak curvature space, the Hyperbolic divergence between $w' - \hat{w}$ and $w - \hat{w}$ is*

$$D_H[w' - \hat{w} : w - \hat{w}] = \log\frac{\cosh(w' - \hat{w})}{\cosh(w - \hat{w})} - (w' - w)\tanh(w - \hat{w}). \tag{15}$$

Let $dw \to 0$. The Taylor expansion of Hyperbolic divergence is (see Appendix 6.5)

$$|dw|^2 = 2D_H[w - \hat{w} : w + dw - \hat{w}] \approx dw\,\mathrm{diag}\left(\mathrm{vec}(1 - \tanh^2(\mathbf{w} - \hat{\mathbf{w}}))\right) dw^\top. \tag{16}$$

Comparing eq. 4 and eq. 12, we can deduce the metric $\epsilon_{ij} = -\mathrm{diag}\left(\mathrm{vec}(\tanh^2(\mathbf{w} - \hat{\mathbf{w}}))\right)_{ij}$. Now we can deduce the steepest descent direction while taking into account the weak curvature in Riemannian manifold defined by Hyperbolic divergence. With constraint Hyperbolic divergence in a constant c, we do the minimization of the loss function $L(\mathbf{w})$ in Lagrangian form:

$$\begin{aligned} dw^* &= \underset{s.t. D_H[w-\hat{w}:w+dw-\hat{w}]=c}{\arg\min} L(w + dw) \\ &= \arg\min_{dw} L(w + dw) - \frac{1}{\lambda}\left(D_H[w - \hat{w} : w + dw - \hat{w}] - c\right) \\ &\approx \arg\min_{dw} L(w) + \mathrm{vec}(\frac{\partial L}{\partial \mathbf{w}})dw^\top - \frac{1}{2\lambda}dw\,\mathrm{diag}\left(\mathrm{vec}(1 - \tanh^2(\mathbf{w} - \hat{\mathbf{w}}))\right) dw^\top + \frac{c}{\lambda}. \end{aligned} \tag{17}$$

[3] The operator $\mathrm{diag}(\cdot)$ means to convert a row vector to a diagonal matrix.

To solve the above minimization, we set its derivative with respect to dw to zero:

$$\begin{aligned} 0 &= \mathrm{vec}(\frac{\partial L}{\partial \mathbf{w}})^\top - \frac{1}{2\lambda}\,\mathrm{diag}\left(\mathrm{vec}(1-\tanh^2(\mathbf{w}-\hat{\mathbf{w}}))\right) dw^\top \\ \frac{1}{\lambda}\,\mathrm{diag}&\left(\mathrm{vec}(1-\tanh^2(\mathbf{w}-\hat{\mathbf{w}}))\right) dw^\top = \mathrm{vec}(\frac{\partial L}{\partial \mathbf{w}})^\top \\ dw^\top &\approx \lambda\,\mathrm{diag}\left(\mathrm{vec}(1+\tanh^2(\mathbf{w}-\hat{\mathbf{w}}))\right) \mathrm{vec}(\frac{\partial L}{\partial \mathbf{w}})^\top. \end{aligned} \tag{18}$$

Where a constant factor λ can be absorbed into learning rate. This is the optimized natural gradient defined by Hyperbolic divergence, which is only suitable for weak curvature manifolds. By constructing a geometric structure with weak curvature, this natural gradient is consistent with the derived eq. 13. Although it has certain limitations compared with the natural gradient defined by KL divergence, it is quite computationally friendly and maks full use of the inherent geometric structure of the low-precision model.

3.4 Smooth quantization along the nearly flat manifold

For the process of the quantization, we design the smooth quantization $\widetilde{Q}$ based on the proposed Hyperbolic divergence in weak curvature:

$$\widetilde{Q}(w_i, \hat{w}_i) = \hat{w}_i - \frac{s}{2}\tanh(w_i - \hat{w}_i), \tag{19}$$

where a constant factor s is used to switch between training and test states ($s = 1$ corresponds to the training process, and $s = 0$ corresponds to the test process). Note that keeping s at zero during training will degrade ONGD to SGD. By limiting the value range of weights to $[-1, 1]$ and applying STE, we can obtain the optimized natural gradient through back-propagation:

$$\begin{aligned} \mathrm{vec}(\nabla_{w_i} L)^\top &= \mathrm{vec}(\frac{\partial L}{\partial \widetilde{Q}(w_i, \hat{w}_i)} \circ \frac{\partial \widetilde{Q}(w_i, \hat{w}_i)}{\partial w_i})^\top \\ &\overset{\mathrm{STE}}{=} \mathrm{vec}(\frac{\partial L}{\partial \widetilde{Q}(w_i, \hat{w}_i)})^\top \circ \mathbb{I}_{|w_i|\le 1} - \frac{1}{2}\,\mathrm{diag}\left(\mathrm{vec}(1-\tanh^2(w_i-\hat{w}_i))\right) \mathrm{vec}(\frac{\partial L}{\partial \widetilde{Q}(w_i, \hat{w}_i)})^\top \\ &= \frac{1}{2}\,\mathrm{diag}\left(\mathrm{vec}(1+\tanh^2(w_i-\hat{w}_i))\right) \mathrm{vec}(\nabla_{\widetilde{Q}(w_i,\hat{w}_i)} L)^\top. \end{aligned} \tag{20}$$

Intuitively, we utilize $\nabla_{w_i} L$ to calculate the optimized natural gradient of the loss function with respect to $\widetilde{Q}(w_i, \hat{w}_i)$, by transforming the parameter space from w_i to $\widetilde{Q}(w_i, \hat{w}_i)$.

3.5 Synchronization in progressive training

Given the smooth quantization, it is necessary to state that the form of quantization function is unconstrained. For weights quantized to k-bit, we can design the uniform quantization, non-uniform power quantization or any other quantizations (see Appendix 6.6). Note that the uniform quantization keeps convolutional calculation into low-precision fixed-point. And the quantization constrained to the power of 2 replaces multiplication with shift operation, which can make computation extremely efficient. Of course, we take the same measure to quantize activations, whether it is the quantization function or the smooth method.

According to the model of input-output relations of neural networks, the training of a low-precision model is regarded as the process of approximating the output probability distribution of the full-precision model. To link these two models between full-precision and low-precision, we add the KL divergence into the loss function. The complete training with the distillation can be carried out in two stages [4]. In the first stage, we train the low-precision model based on ONGD-D with the loss function:

$$L(\widetilde{Q}(w_i, \hat{w}_i)) = \alpha\,\mathrm{CE}(y_{\widetilde{Q}(w_i,\hat{w}_i)}, y_{\mathrm{true}}) + (1-\alpha) D_{KL}[y_{\widetilde{Q}(w_i,\hat{w}_i)} : y_{w_i'}], \tag{21}$$

[4]Note ONGD-D represents that the optimized natural gradient and distillation are used in the training, and ONGD represents that the optimized natural gradient is only used in the training, i.e. without the information of the full-precision model.

Table 1: The classification accuracy results on CIFAR10 and CIFAR100 with ResNet-20, ResNet-32 and VGG13-small. Note the accuracy of full-precision baseline is reported by (Chen et al., 2020).

Method	**W**	**A**	**CIFAR10**		**CIFAR100**	
			Top-1	Gap	Top-1	Gap
ResNet-20 (Original)	32	32	92.25%	-	68.14%	-
ONGD (ours)	4	4	89.41%	-2.84%	63.30%	-4.84%
SGD	4	4	89.27%	-2.98%	62.38%	-5.76%
ONGD-D (ours)	1	8	91.16%	-1.09%	65.65%	-2.49%
PQ-B (Bai et al., 2018)	1	32	90.65%	-1.60%	-	-
ResNet-32 (Original)	32	32	93.29%	-	69.74%	-
ONGD (ours)	4	4	91.70%	-1.59%	66.18%	-3.56%
SGD	4	4	91.23%	-2.06%	64.13%	-5.61%
ONGD-D (ours)	1	8	92.36%	-0.93%	67.07%	-2.67%
PQ-B (Bai et al., 2018)	1	32	91.47%	-1.82%	-	-
VGG13-small (Original)	32	32	93.17%	-	72.06%	-
ONGD (ours)	4	4	92.69%	-0.47%	71.19%	-0.87%
SGD	4	4	92.43%	-0.74%	70.44%	-1.62%

where α is a balancing parameter that also serves as a smooth label. Specifically, $\mathrm{CE}(\cdot)$ is the cross-entropy between the softmax output of low-precision models $\widetilde{Q}(w_i, \hat{w}_i)$ and true hard label, and $D_{KL}[\cdot]$ is the KL divergence of the softmax output between low-precision $\widetilde{Q}(w_i, \hat{w}_i)$ and full-precision w_i' models. In the next second stage, following with the works (Hinton et al., 2015; Zhuang et al., 2018), we consider two-way knowledge distillation for low-precision and full-precision models in progressive training. As such, we continue to fine-tune the low-precision model with eq.21, and update the full-precision model with eq.22, alternating between the two pieces of training,

$$L(w_i') = \alpha\,\mathrm{CE}(y_{w_i'}, y_{\mathrm{true}}) + (1-\alpha) D_{KL}[y_{w_i'}, y_{\widetilde{Q}(w_i:\hat{w}_i)}]. \tag{22}$$

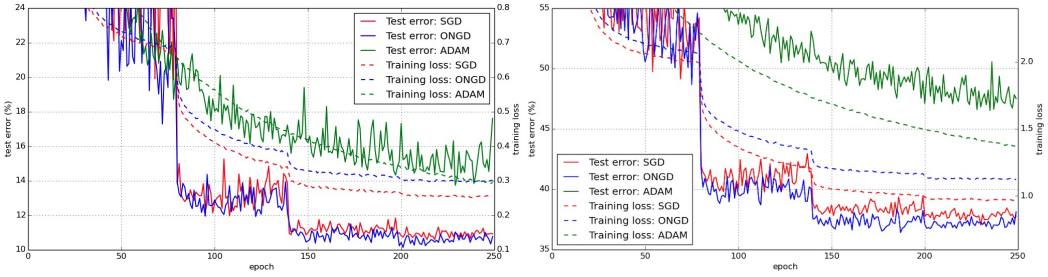
Since we have built a bridge between the two models, this measure to improve the accuracy of the low-precision model can be attributed to a physical phenomenon called synchronization. The two pendulum clocks placed on the same wooden board will have the same frequency θ_1 and phase θ_2 after a while, whether their previous states $\theta = (\theta_1, \theta_2)$ are consistent or not. Here, the essence of two-way distillation is also a synchronization phenomenon, which will synchronize the states $\theta' = (\theta_1', \theta_2', \cdots, \theta_n')$ of the two models (namely the classification probability of the network output). The KL divergence between the two models acts as the wooden board that transmits the kinetic energy of two pendulum clocks to each other. As a result, we have obtained a low-precision model with improved performance.

4 Experiment

In this section, we implement experiments to demonstrate the effectiveness of our proposed methods on benchmark datasets that are CIFAR and ImageNet mainly here. Intuitively, experiments on CIFAR and ImageNet are the ablation study to validate the advantages of ONGD compared to SGD. Comparisons with other quantizers on quantitative indicates will be carried out on ImageNet. All experiments are conducted with TensorFlow (Abadi et al., 2016) and Keras (Chollet et al., 2015).

4.1 Experiments on CIFAR

In order to illustrate the superiority of the optimized natural gradient, we train QNNs with our method ONGD on the CIFAR dataset from scratch. We choose 1-bit, 4-bit or 8-bit quantization with eq.19 and eq.38 to test the ResNet-20 (He et al., 2016) model and VGG13-small (Simonyan & Zisserman, 2014) model with standard data augmentation and pro-precessing. Note thats VGG13-small is the structure of VGG13 that does not contain all fully-connected layers. We use a weight

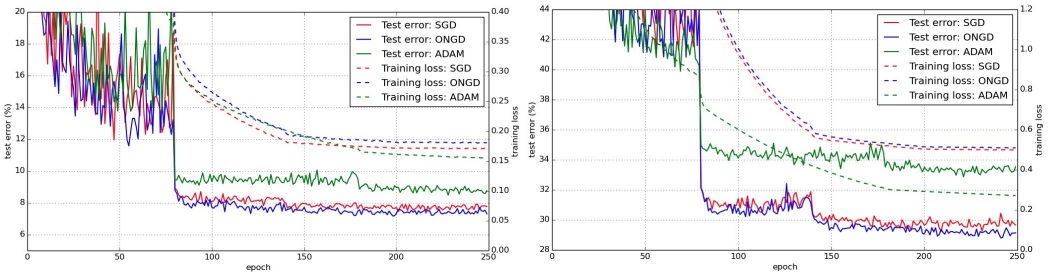


(a) ResNet-20 on CIFAR10 (b) ResNet-20 on CIFAR100

Figure 2: Training and test curves of ResNet-20 compared between ONGD and SGD.

decay of 1e-4, a batch size of 128 and nesterov momentum of 0.9. The learning strategy is lowered by 10 times at epoch 80, 140 and 200, with the initial 0.1. The accuracy results are shown in Table 1, where the most significant results show that the 4-bit quantized ResNet-32 trained using ONGD has better results than using SGD, i.e. 2.05% higher on CIFAR100 dataset.

By observing the curves in Figure 2, we see that our method ONGD can make the test error lower when its training loss is much larger than SGD and ADAM (ADAM with a learning rate of 1e-4). Under the same condition, including quantization function, quantized layers, etc., we evaluate the training error of ResNet-20 model using ONGD that increases much more than using SGD (*i.e.* 1.49% in CIFAR10 and 4.86% in CIFAR100). However, the test error of ResNet-20 model with ONGD can be lower compared to SGD (*i.e.* 0.14% in CIFAR10 and 0.92% in CIFAR100).



(a) VGG13-small on CIFAR10 (b) VGG13-small on CIFAR100

Figure 3: Training and test curves of VGG13-small compared between ONGD and SGD.

Similar to the above experiment, we test our method in Figure 3 using VGG13-small model that is more redundant than ResNet model (The learning strategy of ADAM is lowered by 10 times at epoch 80 and 180, with the initial 1e-4). The graph shows that the performance of ONGD in training continues to be worse, but its test error is lower than SGD (*i.e.* 0.26% in CIFAR10 and 0.75% in CIFAR100).

4.2 Experiments on ImageNet

We next conduct experiments on the more challenging large ImageNet dataset (Krizhevsky et al., 2012). For training, we use standard data augmentation, i.e., resize the images to 256x256 and then randomly crop them to 224x224, and apply random horizontal flips. We use the 224x224 crop of the images on evaluation. In weights or activations, the quantizer applies eq.19 and eq.38 for 1-bit or 8-bit, and we use eq.19 and eq.39 for 4-bit. We show the accuracy results in Table 2.

For ImageNet, the quantized models are trained from the guidance of a pre-trained full-precision model with a weight decay of 1e-4 and nesterov momentum of 0.9 followed with (Elhoushi et al., 2019). The learning strategy is lowered by 10 times every 15 epochs, with the initial 0.1. As a result, the best accuracy for ResNet-50 with 8-bit quantization is less than the full-precision model by

Table 2: The classification accuracy results on ImageNet and comparison with other quantizers for fair, with AlexNet, ResNet-18, ResNet-50 and MobileNet. Note that the accuracy of full-precision baseline is reported by (Elhoushi et al., 2019).

Method	**W**	**A**	**Top-1**		**Top-5**	
			Accuracy	Gap	Accuracy	Gap
AlexNet (Original)	32	32	56.52%	-	79.07%	-
ONGD-D-small (ours)	4	4	55.89%	-0.63%	78.58%	-0.49%
DeepShift-Q (Elhoushi et al., 2019)	6	32	54.97%	-1.55%	78.26%	-0.81%
ResNet-18 (Original)	32	32	69.76%	-	89.08%	-
Bi-Real (Liu et al., 2018)	1	1	56.40%	-13.36%	79.50%	-9.58%
ONGD-D-small (ours)	1	1	58.63%	-11.13%	81.44%	-7.64%
QN (Yang et al., 2019)	1	2	63.40%	-6.36%	84.90%	-4.18%
ONGD-D-small (ours)	1	8	64.05%	-5.71%	85.09%	-3.99%
MetaQuant (Chen et al., 2019)	1	32	60.32%	-9.44%	83.02%	-6.06%
ONGD-D-small (ours)	4	4	67.55%	-2.21%	87.32%	-1.76%
RQ ST (Louizos et al., 2019)	4	4	62.46%	-7.30%	84.78%	-4.30%
ResNet-50 (Original)	32	32	76.13%	-	92.86%	-
ONGD-D-small (ours)	4	4	73.51%	-2.62%	90.76%	-2.10%
LQ-Nets (Zhang et al., 2018)	4	4	75.10%	-1.03%	92.40%	-0.46%
ABC-Net (Lin et al., 2017)	5	5	70.10%	-6.03%	89.70%	-3.16%
ONGD-D-large (ours)	8	8	76.10%	-0.03%	92.88%	+0.02%
INT8 (Zhu et al., 2020)	8	8	75.87%	-0.26%	-	-
MobileNet (Original)	32	32	70.61%	-	89.47%	-
ONGD-D-small (ours)	4	4	61.32%	-9.31%	82.35%	-7.12%
RQ (Louizos et al., 2019)	5	5	61.38%	-9.23%	83.73%	-5.74%
SR+DR (Gysel et al., 2018)	5	5	59.39%	-11.22%	82.35%	-7.12%
RQ ST (Louizos et al., 2019)	5	5	56.85%	-13.76%	80.35%	-9.12%
ONGD-D-large (ours)	8	8	70.86%	+0.25%	89.60%	+0.13%
RQ (Louizos et al., 2019)	8	8	70.43%	-0.18%	89.42%	-0.05%

0.03% in Top-1 accuracy. For MobileNet (Howard et al., 2017), the accuracy of applying ONGD-D is even slightly higher than the original model about 0.25% in Top-1 accuracy. As for 4-bit quantization, our performance is also better than some other quantizers under the same conditions. Note that more ablation studies about the optimized natural gradient and distillation are in Appendix 6.7.

5 Conclusion

By defining a convex function with a geometric structure, we get a novel Hyperbolic divergence that helps us calculate the optimized natural gradient in weak curvature, i.e. differenting from the natural gradient in arbitrary curvature under the definition of KL divergence. Coupled with the two-way distillation between low-precision and full-precision models, we demonstrate that the QNN trained with ONGD can achieve state-of-the-art classification results on ImageNet.

The development of QNNs began with the study of ultra low-precision, e.g. BNN (Courbariaux et al., 2016), BC (Courbariaux et al., 2015), TWN (Li et al., 2016) and TTQ (Zhu et al., 2016), which replace multiplication with XNOR operation. It was later developed into DeepShift (Elhoushi et al., 2019) that replaces multiplication with SHIFT operation. Recently, there are works to modify the network structure (Bethge et al., 2020) and activation function (Choi et al., 2018b;a), which can be said to include all aspects of neural networks. Future work may need to pay more attention to the characteristics of the quantization itself and use mathematical tools to study it.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. *arXiv preprint arXiv:1810.00861*, 2018.

Michael R Bastian, Jacob H Gunther, and Todd K Moon. A simplified natural gradient learning algorithm. *Advances in Artificial Neural Systems*, 2011, 2011.

Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. Meliusnet: Can binary neural networks achieve mobilenet-level accuracy? *arXiv preprint arXiv:2001.05936*, 2020.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1468–1477, 2020.

J. Chen, Y. Liu, H. Zhang, S. Hou, and J. Yang. Propagating asymptotic-estimated gradients for low bitwidth quantized neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 14(4): 848–859, 2020.

Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. In *Advances in Neural Information Processing Systems*, volume 32, pp. 3916–3926. Curran Associates, Inc., 2019.

Jungwook Choi, Pierce I-Jen Chuang, Zhuo Wang, Swagath Venkataramani, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Bridging the accuracy gap for 2-bit quantized neural networks (qnn). *arXiv preprint arXiv:1807.06964*, 2018a.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018b.

François Chollet et al. keras, 2015.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. *arXiv preprint arXiv:1905.13298*, 2019.

Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5784–5789, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sigurdur Helgason. *Differential geometry and symmetric spaces*, volume 341. American Mathematical Soc., 2001.

G Hinton. Neural networks for machine learning. coursera,[video lectures], 2012.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pp. 345–353, 2017.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkxjYoCqKX`.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of dnns with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Nicolas L Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In *Advances in neural information processing systems*, pp. 849–856, 2008.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Robert M Wald. *General relativity*. University of Chicago press, 2010.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xiansheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382, 2018.

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.

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

Feng Zhu, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan. Towards unified int8 training for convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1969–1979, 2020.

Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7920–7928, 2018.

6 APPENDIX

6.1 STRAIGHT THROUGH ESTIMATOR IN N-BIT MODELS

Let us define $L = L(\hat{a}_i, e_i)$ where $\hat{a}_i$ follows eq.9 that is chosen from $(\pm 2^{-0}, \pm 2^{-1}, \cdots, 0)$, then we get a new expression as

$$\mathbb{E}_{e_i}\left[\frac{\partial L}{\partial s_i}\right] = c\frac{\partial L}{\partial \hat{a}_i}, \text{if } |s_i| \leq 1. \tag{23}$$

Where e_i is the noise source that influences s_i, $\mathbb{E}_{e_i}[\cdot]$ means the expectation over s_i, and c is a constant. We have

$$\begin{aligned}
&\mathbb{E}_{e_i}\left[\frac{\partial}{\partial s_i}L\right] = \frac{\partial}{\partial s_i}\mathbb{E}_{e_i}[L] \\
&= \frac{\partial}{\partial s_i}[\sum_i L(\hat{a}_i = +2^j)P(s_i > e_i|s_i) + \sum_j L(\hat{a}_i = -2^j)(1 - P(s_i > e_i|s_i))] \\
&= \frac{\partial P(s_i > e_i|s_i)}{\partial s_i}[\sum_j L(\hat{a}_i = +2^j) - \sum_j L(\hat{a}_i = -2^j)].
\end{aligned} \tag{24}$$

For $L(\hat{a}_i = \pm 2^j)$, we can approximate it using the Taylor expansion:

$$\begin{aligned}
L(\hat{a}_i = +2^j)) &= L(\hat{a}_i = 0) + \frac{\partial L}{\partial \hat{a}_i}\Big|_{\hat{a}_i=0} 2^i + \frac{\partial^2 L}{\partial \hat{a}_i^2}\Big|_{\hat{a}_i=0} 2^{2j} + O\left(\frac{\partial^3 L}{\partial \hat{a}_i^3}\Big|_{\hat{a}_i=0} 2^{3j}\right), \\
L(\hat{a}_i = -2^j)) &= L(\hat{a}_i = 0) - \frac{\partial L}{\partial \hat{a}_i}\Big|_{\hat{a}_i=0} 2^j + \frac{\partial^2 L}{\partial \hat{a}_i^2}\Big|_{\hat{a}_i=0} 2^{2j} + O\left(\frac{\partial^3 L}{\partial \hat{a}_i^3}\Big|_{\hat{a}_i=0} 2^{3j}\right).
\end{aligned} \tag{25}$$

For $\frac{\partial P(s_i > e_i|s_i)}{\partial s_i}$, we split it into two parts:

$$\begin{aligned}
\frac{\partial P(s_i > e_i|s_i)}{\partial s_i} &= \frac{\partial P(s_i > e_i|s_i)}{\underset{|s_i|>1}{\partial s_i}} + \frac{\partial P(s_i > e_i|s_i)}{\underset{|s_i|\leq 1}{\partial s_i}} \\
&= \frac{\partial \int_{-1}^{1} \frac{1}{2}\, de_j}{\partial s_i} + \frac{\partial \int_{-s_i}^{s_i} \frac{1}{2}\, de_i}{\partial s_i} = \mathbf{1}_{|s_i|\leq 1}.
\end{aligned} \tag{26}$$

Combining eq.25 and eq.26, the eq.24 can be derived as

$$\mathbb{E}_{e_i}\left[\frac{\partial L}{\partial s_i}\right] = \mathbb{I}_{|s_i|\leq 1} \circ \left(2\sum_j 2^{2j}\frac{\partial L}{\partial \hat{a}_i}\Big|_{\hat{a}_i=0}\right). \tag{27}$$

Let $2\sum_j 2^{2j} = c$, then

$$\mathbb{E}_{e_i}\left[\frac{\partial L}{\partial s_i}\right] = c\frac{\partial L}{\partial \hat{a}_i} \circ \mathbb{I}_{|s_i|\leq 1}. \tag{28}$$

6.2 DEFINITION OF DIVERGENCE IN A MANIFOLD

$D[P:Q]$ is called a divergence when it satisfies the following criteria:

1) $D[P:Q] \geq 0$.

2) $D[P:Q] = 0$ when and only when $P = Q$.

3) When P and Q are sufficiently close, by denoting their coordinates by ψ_P and $\psi_Q = \psi_P + d\psi$, the Taylor expansion of D is written as

$$D[\psi_P : \psi_P + d\psi] = \frac{1}{2}\sum_{i,j} G_{ij}(\psi_P) d\psi^i d\psi^j + O(|d\psi|^3), \tag{29}$$

and Riemannian metric G_{ij} is positive-definite, depending on ψ_P.

6.3 BREGMAN DIVERGENCE

Bregman divergence $D_B[w:w']$ is defined as the difference between a convex function $\psi(w)$ and its tangent hyperplane $z = \psi(w') + (w - w')\nabla\psi(w')$, depending on the Taylor expansion at the point w':

$$D_B[w:w'] = \psi(w) - \psi(w') - (w - w')\nabla\psi(w'). \tag{30}$$

6.4 KL DIVERGENCE AND FISHER INFORMATION MATRIX

KL divergence can be defined between $\psi(x|w)$ and $\psi(x|w')$:

$$D_{KL}[\psi(x|w) : \psi(x|w')] = \int \psi(x|w)\log\psi(x|w)dx - \int \psi(x|w)\log\psi(x|w')dx. \tag{31}$$

The first derivative is:

$$\begin{aligned}\nabla_{w'} D_{KL}[\psi(x|w) : \psi(x|w')] &= \int \psi(x|w)\nabla_{w'}\log\psi(x|w)dx - \int \psi(x|w)\nabla_{w'}\log\psi(x|w')dx \\ &= -\int \psi(x|w)\nabla_{w'}\log\psi(x|w')dx.\end{aligned} \tag{32}$$

The second derivative is:

$$\begin{aligned}\nabla^2_{w'} D_{KL}[\psi(x|w) : \psi(x|w')] &= \int \psi(x|w)\nabla^2_{w'}\log\psi(x|w)dx - \int \psi(x|w)\nabla^2_{w'}\log\psi(x|w')dx \\ &= -\int \psi(x|w)\nabla^2_{w'}\log\psi(x|w')dx.\end{aligned} \tag{33}$$

We deduce the Taylor expansion of KL divergence at $w = w'$:

$$\begin{aligned}&D_{KL}[\psi(x|w) : \psi(x|w')] \approx D_{KL}[\psi(x|w) : \psi(x|w)] - \\ &\left(\int \psi(x|w)\nabla_{w'}\log\psi(x|w)|_{w=w'}dx\right)dw^\top - \frac{1}{2}dw\left(\int \psi(x|w)\nabla^2_{w'}\log\psi(x|w')|_{w=w'}dx)\right)dw^\top \\ &= 0 - \left(\int \psi(x|w)\frac{\nabla\psi(x|w)}{\psi(x|w)}dx\right)dw^\top - \frac{1}{2}dw\left(\int \psi(x|w)\nabla\left[\frac{\nabla\psi(x|w)}{\psi(x|w)}\right]dx\right)dw^\top \\ &= -\left(\nabla\int \psi(x|w)dx\right)dw^\top - \frac{1}{2}dw\left(\int \psi(x|w)\frac{\nabla^2\psi(x|w)\psi(x|w) - \nabla\psi(x|w)\nabla\psi(x|w)}{\psi(x|w)\psi(x|w)}dx\right)dw^\top \\ &= -(\nabla 1)dw^\top - \frac{1}{2}dw\left(\nabla^2\int \psi(x|w)dx - \int \psi(x|w)\left[\frac{\nabla\psi(x|w)}{\psi(x|w)}\right]\left[\frac{\nabla\psi(x|w)}{\psi(x|w)}\right]dx\right)dw^\top \\ &= \frac{1}{2}dw\mathbb{E}_{\psi_w}[\nabla\log\psi(w)\nabla\log\psi(w)]dw^\top = \frac{1}{2}dwF_{ij}dw^\top.\end{aligned} \tag{34}$$

The Taylor expansion of KL divergence at $w = w'$ is related to Fisher Information Matrix.

Table 3: The ablation studies on ImageNet, with AlexNet, ResNet-18 and ResNet-50. Note that the accuracy of full-precision baseline is reported by (Elhoushi et al., 2019).

Method	**W**	**A**	**Top-1**		**Top-5**	
			Accuracy	Gap	Accuracy	Gap
AlexNet (Original)	32	32	56.52%	-	79.07%	-
ONGD-D (ours)	4	4	55.89%	-0.63%	78.58%	-0.49%
ONGD (ours)	4	4	54.95%	-1.57%	78.38%	-0.69%
SGD	4	4	53.84%	-2.68%	77.57%	-1.5%
ResNet-18 (Original)	32	32	69.76%	-	89.08%	-
ONGD-D (ours)	4	4	67.55%	-2.21%	87.32%	-1.76%
ONGD (ours)	4	4	66.42%	-3.34%	86.20%	-2.88%
SGD	4	4	64.64%	-5.12%	85.10%	-3.98%
ResNet-50 (Original)	32	32	76.13%	-	92.86%	-
ONGD-D (ours)	4	4	73.51%	-2.62%	90.76%	-2.10%
ONGD (ours)	4	4	71.80%	-4.33%	90.45%	-2.41%
SGD	4	4	68.99%	-7.14%	89.39%	-3.47%

6.5 HYPERBOLIC DIVERGENCE

The first derivative of Hyperbolic divergence is:

$$\begin{aligned}\nabla_{w'} D_H[w' - \hat{w} : w - \hat{w}] &= \nabla_{w'} \log\cosh(w' - \hat{w}) - 0 - \nabla_{w'}(w' - w)\tanh(w - \hat{w}) \\ &= \tanh(w' - \hat{w}) - \tanh(w - \hat{w}).\end{aligned} \tag{35}$$

The second derivative of Hyperbolic divergence is:

$$\begin{aligned}\nabla^2_{w'} D_H[w' - \hat{w} : w - \hat{w}] &= \nabla^2_{w'} \log\cosh(w' - \hat{w}) - 0 - \nabla^2_{w'}(w' - w)\tanh(w - \hat{w}) \\ &= 1 - \tanh^2(w' - \hat{w}).\end{aligned} \tag{36}$$

We deduce the Taylor expansion of Hyperbolic divergence at $w = w'$:

$$\begin{aligned}D_H[w' - \hat{w} : w - \hat{w}] \approx\ & D_H[w - \hat{w} : w - \hat{w}] + \\ & \operatorname{vec}\left(\tanh(\mathbf{w'} - \hat{\mathbf{w}})|_{\mathbf{w}=\mathbf{w'}} - \tanh(\mathbf{w} - \hat{\mathbf{w}})\right) dw^\top + \\ & \frac{1}{2} dw \operatorname{diag}\left(\operatorname{vec}(1 - \tanh^2(\mathbf{w'} - \hat{\mathbf{w}}))\right)|_{\mathbf{w}=\mathbf{w'}} dw^\top \\ =\ & \frac{1}{2} dw \operatorname{diag}\left(\operatorname{vec}(1 - \tanh^2(\mathbf{w} - \hat{\mathbf{w}}))\right) dw^\top.\end{aligned} \tag{37}$$

6.6 QUANTIZATION FUNCTION

The uniform quantization:

$$\hat{w}_i = \max(|w_i|)\left[\frac{1}{2^{k-1} - 1}\operatorname{round}\left((2^{k-1} - 1)\frac{w_i}{\max(|w_i|)}\right)\right]. \tag{38}$$

The non-uniform power quantization:

$$\hat{w}_i = \max(|w_i|)\left[\operatorname{sign}(w_i) 2\hat{}\operatorname{round}\left(\operatorname{clip}(\log_2(\frac{|w_i|}{\max(|w_i|)} + \epsilon), -2^{k-1} + 1, 0)\right)\right], \tag{39}$$

where ϵ is a small constant to prevent the log function from exploding. In a convolutional layer, the full-precision convolutional operation can be converted into the low-precision fixed-point convolutional operator and multiplied by a factor $\max(|w_i|)\max(|a_i|)$ [5]:

[5]The operator $\operatorname{Conv}(\cdot, \cdot)$ means the 2D convolutional operation.

$$\begin{aligned}\mathrm{Conv}(w_i, a_i) &\approx \mathrm{Conv}\left(\max(|w_i|)\frac{\hat{w}_i}{\max(|w_i|)}, \max(|a_i|)\frac{\hat{a}_i}{\max(|a_i|)}\right) \\ &= \max(|w_i|)\max(|a_i|)\,\mathrm{Conv}\left(\frac{\hat{w}_i}{\max(|w_i|)}, \frac{\hat{a}_i}{\max(|a_i|)}\right).\end{aligned} \tag{40}$$

6.7 THE ABLATION STUDY ON IMAGENET

The ablation studies on ImageNet are shown in Table 3. We train QNNs with our method on the ImageNet dataset from scratch, with a weight decay of 1e-4 and nesterov momentum of 0.9. The learning strategy is lowered by 10 times every 15 epochs, with the initial 0.1 and a batch size of 128. During training and inference, we strictly ensure that quantization function and other conditions are the same. We use the initialization of Xavier, and quantize the weights and activations of all layers except the first layer where the low-precision model is trained from scratch.