# Propagating Asymptotic-Estimated Gradients for Low Bitwidth Quantized Neural Networks

Jun Chen, Yong Liu, *Member, IEEE*, Hao Zhang, Shengnan Hou, Jian Yang

***Abstract*—The quantized neural networks (QNNs) can be useful for neural network acceleration and compression, but during the training process they pose a challenge: how to propagate the gradient of loss function through the graph flow with a derivative of 0 almost everywhere. In response to this non-differentiable situation, we propose a novel Asymptotic-Quantized Estimator (AQE) to estimate the gradient. In particular, during back-propagation, the graph that relates inputs to output remains smoothness and differentiability. At the end of training, the weights and activations have been quantized to low-precision because of the asymptotic behaviour of AQE. Meanwhile, we propose a M-bit Inputs and N-bit Weights Network (MINW-Net) trained by AQE, a quantized neural network with 1-3 bits weights and activations. In the inference phase, we can use XNOR or SHIFT operations instead of convolution operations to accelerate the MINW-Net. Our experiments on CIFAR datasets demonstrate that our AQE is well defined, and the QNNs with AQE perform better than that with Straight-Through Estimator (STE). For example, in the case of the same ConvNet that has 1-bit weights and activations, our MINW-Net with AQE can achieve a prediction accuracy 1.5% higher than the Binarized Neural Network (BNN) with STE. The MINW-Net, which is trained from scratch by AQE, can achieve comparable classification accuracy as 32-bit counterparts on CIFAR test sets. Extensive experimental results on ImageNet dataset show great superiority of the proposed AQE and our MINW-Net achieves comparable results with other state-of-the-art QNNs.**

***Index Terms*—Deep learning, quantized neural network (QNN), back-propagation, estimators**

## I. Introduction

RECENTLY, deep convolutional neural networks (DCNNs) have substantially dominated a variety of computer vision fields such as image classification [1]–[3], face recognition [4], [5], semantic segmentation [6], [7] and object detection [8], [9]. However, the dazzling performance of DCNNs is at the expense of a large number of parameters and high computational complexity, which place high demands on the storage unit and greatly drag down the computational efficiency when considering to apply them to embedded devices and hardware platforms [10]–[12].

In order to deal with this problem, some studies [13]–[24] have specifically considered to reduce the parameters and improve the computational efficiency by using quantized neural networks (QNNs). In general, QNNs can be roughly divided into three categories: 1) simultaneously quantizing weights and activations whose model is the simplest such as [13]. 2) only quantizing weights such as [14], [17]. 3) introducing scale factors in addition to the quantized values to fit floating-point numbers [15], [16], [19].

In principle, we can train neural networks by using gradient-based learning algorithm which is a well-known back-propagation algorithm [25]. In this algorithm, the gradients propagate through the graph flow related between inputs and outputs. Although some researchers have extended the smoothness condition of the graph flow to non-smoothness condition [26], [27], the graph with a derivative of 0 almost everywhere is still not well processed when considering QNNs with low-precision weights and activations. For the graph flow like Dirac Delta Function, Bengio *et al*. [28] give an unbiased estimator of gradient, where some components of the model allow a binary decision with stochastic perturbations [29]. This estimator only requires broadcasting the loss function, instead of using back-propagation algorithm. Another estimator is Straight-Through Estimator (STE) proposed by Hinton *et al*. [30], which back-propagates by replacing the sign function with identity function. However, why can't we revert to the original smoothness condition to consider the problem of quantization?

In this paper, according to the smoothness condition of the graph, we propose the Asymptotic-Quantized Estimator that propagates the gradients through continuously differentiable graph, no need to worry about the derivative of 0. In our AQE, the neuron output function connected with the graph is related to the full-precision value and the quantized value. Under the guidance of the asymptotic behaviour of AQE, the output function gradually approaches the quantized value. When the training is completed, the output function is completely converted into the quantized value, and the graph flow at this time is "non-smooth" non-linearities. By using AQE, we introduce the MINW-Net with low-precision weights and activations. The main contributions of this article are summarized as follows:

1) We propose a novel Asymptotic-Quantized Estimator based on the smoothness condition of the graph that can be used to train quantized neural networks. This estimator maintains the smoothness and differentiability of the graph flow during the training process and restores the non-smooth connection of the graph at the end of training. In particular, we demonstrate that our AQE will degenerate to STE and unbiased estimator when $\alpha = \frac{1}{2}$.
2) We propose a highly efficient MINW-Net with low-

Jun Chen, Yong Liu, Hao Zhang and Shengnan Hou are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China, 310027, e-mail: yongliu@iipc.zju.edu.cn. Jian Yang is with China Research and Development Academy of Machinery Equipment, Beijing, China, 100089.

Yong Liu is the corresponding author.

precision weights and activations. When choosing 1-3 bits according to the actual situation, we can use XNOR or SHIFT operations instead of convolution operations to accelerate the inference, and the model has a small parameter capacity.

The rest of this paper is organized as follows: Section II summarizes related prior works on estimators and QNNs. Our AQE and MINW-Net are presented in Section III. In Section IV, we demonstrate the validities of our AQE and MINW-Net via comparable experiments. The conclusion is given in Section V.

## II. Related work

### A. Estimating or Propagating Gradients

In general, the output $h_i$ of a stochastic neuron $i$ depends on the noise source $z_i$ and the input $a_i$, where the input $a_i$ is typically the affine transformation $a_i = b_i + \sum_j W_{ij} x_{ij}$ (the internal parameters are typically the bias and weights of the neuron) and $x_i$ is the input of neuron $i$ in neural networks,

$$h_i = f(a_i, z_i) \tag{1}$$

Only the above continuous equation has a non-zero gradient with respect to $a_i$, then the process of calculating gradients using the back-propagation algorithm can be performed. However, considering that the input and output of the neuron are assigned to quantized values in quantized neural networks, the above continuous equation is transformed into the discrete function. At this point, this equation has a derivative of 0 with respect to $a_i$ almost everywhere, the back-propagation algorithm cannot deal with this equation. Therefore, many scholars have carried out researches on this issue, and part of the important work is to construct effective estimators [28]–[31].

1) *Unbiased Estimator:* Bengio *et al.* [28] present the unbiased estimator for stochastic binary neuron based on the framework of **Eq** (1), which can be applied to binary neural networks. In this work, they consider the output $h_i$:

$$h_i = f(a_i, z_i) = \mathbf{1}_{z_i > \mathrm{sigm}(a_i)} \tag{2}$$

Where $z_i \sim U[0,1]$ is the uniform distribution and $\mathrm{sigm}(a_i) = \frac{1}{1+\exp(-a_i)}$ is the sigmoid function.

The gradient can be described as $g_i = \frac{\partial \mathbb{E}_{z_i, c_{-i}}[L|c_i]}{\partial a_i}$ by defining the $L = L(h_i, c_i, c_{-i})$, where the loss function $L$ depends on $h_i$, $c_i$ the noise source that influences $a_i$, and $c_{-i}$ the noise source that does not influence $a_i$.

Considering $h_i \in \{0, 1\}$ based on **Eq** (2), so

$$\begin{aligned} g_i = &\mathbb{E}_{c_i, c_{-i}} [\mathrm{sigm}(a_i)(1 - \mathrm{sigm}(a_i))(L(1, c_i, c_{-i}) - \\ &L(0, c_i, c_{-i}))] \end{aligned} \tag{3}$$

Since the gradient $g_i$ cannot be computed by its derivative, they have constructed another gradient $\hat{g}_i = (h_i - \mathrm{sigm}(a_i)) \times L$ that satisfies $\mathbb{E}[\hat{g}_i] = g_i$. Thus, an unbiased estimation of the gradient can be used to update all parameters in the entire neural networks without the need for a derivative with respect to $a_i$.

2) *Straight-Through Estimator:* Another useful method of training low-precision neural networks is the "Straight-Through Estimator (STE)" introduced in Hinton's lectures [30]. This method simply back-propagates through the sign function (1 if the input is positive, $-1$ otherwise) as if it has been the identity function. Considering the output of activation function, that is sign function here, from a previous layer, $a_i$ in the framework of **Eq** (1),

$$h_i = \mathrm{sign}(a_i). \tag{4}$$

Since an estimator $g_{h_i}$ of the gradient $\frac{\partial L}{\partial h_i}$ has been obtained by back-propagation algorithm where $L$ is the loss function, the STE of $\frac{\partial L}{\partial a_i}$ with respect to the activation value $a_i$ is simply

$$g_{a_i} = g_{h_i} \mathbf{1}_{|a_i| \leq 1}. \tag{5}$$

The binary neural networks can be trained by back-propagation algorithm with STE.

### B. Low Precision Quantized Neural Networks

In inference, considering the acceleration of neural networks with low-precision weights and activations, many studies have been devoted to achieve low-precision neural networks such as Binary Neural Network (BNN), Ternary Neural Network (TNN) and Quantized Neural Network (QNN) [13]–[18], [32].

BinaryConnect [14] is the first article that summarizes the complete quantization process, whose weights in the forward and backward of DNNs are 1-bit fixed-point rather than 32-bit floating-point. The process they proposed allows hardware calculation to simplify multiplication operations into accumulation operations. Simultaneously, a large amount of storage space is reduced, and the performance of classification accuracy has not decreased on MNIST, CIFAR-10 and SVHN. BNN [13] further converts the activation values to 1-bit based on the BinaryConnect, and simplifies many multiply-accumulate operations into bitwise operations. Since both the weights and the activations are quantized into $\{-1, 1\}$, the gradient of the discrete neurons needs to be solved by STE. XNOR-Net [15] proposes a method of fitting floating-point numbers, which adds a scale factor based on binary value instead of simply taking the sign function. By combining binary value weights with scale factor, XNOR-Net can almost achieve the same performance as the full-precision model on AlexNet. The above works are either directly or indirectly related to the 1-bit weights, and Ternary Weight Network (TWN) [17] proposes to quantize the weights into 2-bit, taking only three numbers $\{-1, 0, 1\}$ instead of four numbers, which can perform better than 1-bit weights on MNIST and CIFAR-10.

There is also another series of works on quantized neural networks, DoReFa-Net [16] sets the weights, the activations and the gradients to low-precision, whose advantage is that it can accelerate the calculation not only in the inference, but also in the training because the gradients are quantized. So DoReFa-Net is very suitable to train directly on the hardware platform. INQ [32] proposes an incremental network quantization method, which quantizes the full-precision network to low-precision network by weight grouping, group quantization

and retraining. By restoring a part of the floating-point weights to restore the performance of the model, the precision loss is mitigated. Since the existing methods with binary weights and activations can cause performance degradation, ABC-Net [18] fits floating-point weights through a linear combination of multiple binary weights, which will reduce information loss.

Above all, these quantized neural networks, once involved in quantizing the activations, are likely to use STE to estimate the gradient of the loss with respect to the activations. In other words, the performance of quantized neural networks based on STE is limited by the performance of the STE.

## III. MINW-NET

In this section, we detail our MINW-Net, a QNN with low-precision weights and activations trained by AQE, because we note that STE may not be the best choice for quantized neural networks. First, we elaborate how to exploit Asymptotic-Quantized Estimator and then quantize the weights and the activations to low-precision through AQE.

### A. Quantized Estimator for Binary Neurons

The Straight-Through Estimator above can deal with the Dirac Delta Function $\delta(\cdot)$ with a derivative of 0 almost everywhere, but it can only propagate the gradient by the identity function. Now, let us consider this case, where the probability is a continuous function through stochastic neurons. The stochastic binary neurons of our model correspond to several binary decisions, and sign function is no longer used directly in the forward pass of neural networks. In the back-propagation algorithm, we assume that another continuous function is used to propagate the gradient instead of the identity function. In order to discuss this issue without loss of generality, in the framework of **Eq** (1), we propose our AQE where the output $\hat{h}_i$ of a neuron $i$ satisfies

$$\hat{h}_i(\alpha, h_i(a_i), a_i) = \alpha h_i(a_i) + (1-\alpha)a_i \tag{6}$$

where $\alpha$ is an adjustable variable in the range of $(0, 1)$ and $h_i(a_i) = \text{sign}(a_i)$ as **Eq** (4). According to **Eq** (6), the output $\hat{h}_i(1, h_i(a_i), a_i) = \text{sign}(a_i)$ degenerates into binary neuron if $\alpha = 1$ and the output $\hat{h}_i(0, h_i(a_i), a_i) = a_i$ degenerates into full-precision neuron if $\alpha = 0$.

The following discussion needs to be included in the framework of probability, so we must assume that there is an independent noise source $z_i$ that drives the stochastic samples first. In addition, considering that the value of $\text{sign}(a_i - z_i)$ is $-1$ or 1, there is no reason for the range of $z_i$ to exceed the output of function. So we follow the work [28], and we assume that $z_i \sim U[-1, 1]$ is the uniform distribution.

With the above assumptions, we rewrite **Eq** (6) as

$$\hat{h}_i(\alpha, h_i, a_i, z_i) = \alpha \,\text{sign}(a_i - z_i) + (1-\alpha)a_i \tag{7}$$

During the back-propagation algorithm, we use following equation to calculate the gradient of weights ($W_{ij}^l$ is the connecting between neuron $j$ at layer $l-1$ and neuron $i$ at layer $l$),

$$\frac{\partial L}{\partial W_{ij}^l} = \frac{\partial L}{\partial a_i^l}\hat{h}_j^{l-1}. \tag{8}$$

Considering the expectation of the above equation, this equation can be re-expressed as

$$\begin{aligned}\frac{\partial}{\partial W_{ij}^l}\mathbb{E}_{z_i,c_{-i}}[L|c_i] =& \mathbb{E}_{z_i,c_{-i}}\left[\frac{\partial L}{\partial a_i^l}\hat{h}_j^{l-1}\middle| c_i\right] \\ =& \mathbb{E}_{c_{-i}}\left[\mathbb{E}_{z_i}\left[\frac{\partial L}{\partial a_i^l}\right]\hat{h}_j^{l-1}\middle| c_i\right]\end{aligned} \tag{9}$$

where $c_i$ is the noise source that influences $a_i$, $c_{-i}$ is the noise source that does not influence $a_i$, $\mathbb{E}_{c_{-i}}[\cdot]$ means the expectation over $c_{-i}$, and $\mathbb{E}[\cdot|c_i]$ means the expectation over everything besides $c_i$.

**Theorem 1:** Let us define $L = L(\hat{h}_i, c_i, c_{-i})$ where $\hat{h}_i$ follows **Eq** (7), then our AQE is equivalent to the STE when $\alpha = \frac{1}{2}$ (for brevity, we drop the indices of $l$).

*Proof.*

$$\begin{aligned}&\mathbb{E}_{z_i}\left[\frac{\partial}{\partial a_i}L\right] = \frac{\partial}{\partial a_i}\mathbb{E}_{z_i}[L] \\ &= \frac{\partial}{\partial a_i}[L(\hat{h}_i(\alpha, 1, a_i))P(a_i > z_i|a_i) + \\ &L(\hat{h}_i(\alpha, -1, a_i))(1 - P(a_i > z_i|a_i))] \\ &= \frac{\partial P(a_i > z_i|a_i)}{\partial a_i}[L(\hat{h}_i(\alpha, 1, a_i)) - L(\hat{h}_i(\alpha, -1, a_i))]\end{aligned} \tag{10}$$

For $L(\hat{h}_i(\alpha, h_i, a_i))$, we can approximate it using the Taylor expansion. The output of a single neuron ($h_i = \pm 1$) generally has only a small impact on the loss function [28], thus, $\left.\frac{\partial L}{\partial h_i}\right|_{h_i=0} \gg \left.\frac{\partial^2 L}{\partial h_i^2}\right|_{h_i=0} \gg \left.\frac{\partial^3 L}{\partial h_i^3}\right|_{h_i=0}$. In other words, the $O(\cdot)$ is usually negligible when considering its impact on the last equation.

$$\begin{aligned}&L(\hat{h}_i(\alpha, 1, a_i)) = L(\hat{h}_i(\alpha, 0, a_i)) + \left.\frac{\partial L}{\partial h_i}\right|_{h_i=0} + \\ &\left.\frac{\partial^2 L}{\partial h_i^2}\right|_{h_i=0} + O\left(\left.\frac{\partial^3 L}{\partial h_i^3}\right|_{h_i=0}\right) \\ &L(\hat{h}_i(\alpha, -1, a_i)) = L(\hat{h}_i(\alpha, 0, a_i)) - \left.\frac{\partial L}{\partial h_i}\right|_{h_i=0} + \\ &\left.\frac{\partial^2 L}{\partial h_i^2}\right|_{h_i=0} + O\left(\left.\frac{\partial^3 L}{\partial h_i^3}\right|_{h_i=0}\right)\end{aligned} \tag{11}$$

For $\frac{\partial P(a_i > z_i|a_i)}{\partial a_i}$, we split it into two parts $|a_i| \le 1$ and $|a_i| > 1$.

$$\begin{aligned}&\frac{\partial P(a_i > z_i|a_i)}{\partial a_i} = \frac{\partial P(a_i > z_i|a_i)}{\partial a_i} + \frac{\partial P(a_i > z_i|a_i)}{\partial a_i} \\ &= \left.\frac{\partial \int_{-1}^{1}\frac{1}{2}\,dz_i}{\partial a_i}\right|_{|a_i|>1} + \left.\frac{\partial \int_{-a_i}^{a_i}\frac{1}{2}\,dz_i}{\partial a_i}\right|_{|a_i|\le 1} = \mathbf{1}_{|a_i|\le 1}\end{aligned} \tag{12}$$

Combining **Eq** (11) and **Eq** (12), the **Eq** (10) can be derived as

$$\begin{aligned}&\mathbb{E}_{z_i}\left[\frac{\partial L}{\partial a_i}\right] = \mathbf{1}_{|a_i|\le 1}\left(2\left.\frac{\partial L(\hat{h}_i)}{\partial h_i}\right|_{h_i=0}\right) \\ &= \left(2\frac{\partial L}{\partial h_i}\left.\frac{\partial \hat{h}_i}{\partial h_i}\right|_{h_i=0}\right)\mathbf{1}_{|a_i|\le 1} = 2\alpha\frac{\partial L}{\partial h_i}\mathbf{1}_{|a_i|\le 1}\end{aligned} \tag{13}$$

Let $\alpha = \frac{1}{2}$, then

$$\mathbb{E}_{z_i}\left[\frac{\partial L}{\partial a_i}\right] = \frac{\partial L}{\partial h_i}\mathbf{1}_{|a_i|\leq 1} \tag{14}$$

The **Eq** (14) is equivalent to **Eq** (5) that is the STE. □

**Theorem 2:** On the basis of **Theorem** 1, our AQE is also equivalent to the unbiased estimator when $\alpha = \frac{1}{2}$.

*Proof.*

$$\begin{aligned}\frac{\partial L(\hat{h}_i)}{\partial a_i} &= \frac{\partial L}{\partial h_i}\frac{\partial h_i}{\partial \hat{h}_i(\alpha, h_i, a_i)}\frac{\partial \hat{h}_i(\alpha, h_i, a_i)}{\partial a_i} \\ &= \frac{\partial L}{\partial h_i}\frac{1}{\alpha}(1-\alpha) \underset{\alpha=\frac{1}{2}}{=} \frac{\partial L}{\partial h_i}\end{aligned} \tag{15}$$

We have $\mathbb{E}\left[\frac{\partial L}{\partial a_i}\right] = \frac{\partial L}{\partial a_i}$ that is an unbiased estimator, because $\mathbb{E}\left[\frac{\partial L}{\partial a_i}\right] \underset{\alpha=\frac{1}{2}}{=} \frac{\partial L}{\partial h_i}$ based on **Eq** (13). □

### B. The Asymptotic Behaviour of the AQE

We have demonstrated that our AQE is equivalent to STE and unbiased estimator when $\alpha = \frac{1}{2}$, whereas the output $\hat{h}_i(\alpha, h_i, a_i) = \alpha\,\mathrm{sign}(a_i) + (1-\alpha)a_i$ of stochastic neuron $i$ is still the continuous function, not the discrete function we need. We will use **Theorem** 3 to explain that the continuous function can approach to the discrete function, when the training process is completed.

**Theorem 3:** In the framework of the series, $\hat{h}_i(\alpha, h_i, a_i, z_i)$ will approach to $h_i = \mathrm{sign}(a_i - z_i)$ when the number of iterations is sufficient. An important assumption is that noise source $z_i$ during all iterations obeys the same uniform distribution.

*Proof.* We consider a back-propagation as an iterative process. Thus, we define $a_i(n)$ as the $n$-th iteration of $a_i$, then $\hat{h}_i(\alpha, h_i, a_i(n))$ can be considered the next iteration of $a_i(n)$ which is equivalent to $a_i(n+1)$. We divide the input of stochastic neurons $a_i$ into two parts, which are $a_i^{>z_i}$ and $a_i^{\leq z_i}$. First consider the part of the neurons $a_i > z_i$ ($h_i = 1$ at this time). Thus, the general terms of series $a_i^{>z_i}$ from 1 to $n$ are written as follows based on **Eq** (6),

$$\begin{aligned} a_i^{>z_i}(2) - (1-\alpha)a_i^{>z_i}(1) &= \alpha \quad (1) \\ \vdots \qquad\qquad &\quad \vdots \\ a_i^{>z_i}(n) - (1-\alpha)a_i^{>z_i}(n-1) &= \alpha \quad (n-1) \\ a_i^{>z_i}(n+1) - (1-\alpha)a_i^{>z_i}(n) &= \alpha \quad (n)\end{aligned} \tag{16}$$

Let $(n) + (1-\alpha)\times(n-1) + \cdots + (1-\alpha)^{n-1}\times(1)$, then we get the equation as follows,

$$\begin{aligned} &a_i^{>z_i}(n+1) - (1-\alpha)^{n-1}a_i^{>z_i}(1) = \alpha[1 + (1-\alpha) + \\ &(1-\alpha)^2 + \cdots + (1-\alpha)^{n-1}] \\ &= \alpha\frac{1-(1-\alpha)^n}{1-(1-\alpha)} = 1-(1-\alpha)^n\end{aligned} \tag{17}$$

As the number of iterations increases, $a_i^{>z_i}(n+1)$ will asymptotically approach to 1 (Similarly, $a_i^{\leq z_i}(n+1)$ will asymptotically approach to $-1$). Since the number of iterations is enough ($n \to \infty$) and $1-\alpha$ is in the range of $(0, 1)$, the input of stochastic neurons $a_i(n+1)$ ($a_i^{>z_i}(n+1)$ and $a_i^{\leq z_i}(n+1)$) can be rewritten as $a_i(n+1) = h_i$ (in other words, $\hat{h}_i(\alpha, h_i, a_i(n)) = h_i = \mathrm{sign}(a_i - z_i)$). With the guarantee of **Theorem** 3, we can use our AQE to train the binary neural networks. □

### C. Low-precision Quantization of Weights and Activations

From **Theorem** 3, we know the fact that both weights and activations are continuous values with full-precision in the training process; both weights and activations are discrete values with low-precision in the inference process. In particular, the weights and activations involved in the calculation of loss function are full-precision, and those participated in the calculation of accuracy are low-precision. We don't care if the weights and activations in the training process are quantized, as long as the trained model is quantized in the inference. From the two parts of training and inference, we detail how to execute our MINW-Net with low-precision weights and activations using AQE.

In QNNs, the activations are updated by our AQE in the training:

$$\begin{aligned}&\textbf{Forward: } \hat{h}_i^l = \alpha h_i(a_i^l) + (1-\alpha)a_i^l \\ &\textbf{Backward: } \frac{\partial L}{\partial a_i^l} = 2\alpha\frac{\partial L}{\partial h_i^l}\mathbf{1}_{|a_i|\leq 1},\end{aligned} \tag{18}$$

the weights are updated by our AQE in the training:

$$\begin{aligned}&\textbf{Forward: } \hat{W}_{ij}^l = \alpha h_i(W_{ij}^l) + (1-\alpha)W_{ij}^l \\ &\textbf{Backward: } \frac{\partial L}{\partial W_{ij}^l} = 2\alpha\frac{\partial L}{\partial h_i^l}\mathbf{1}_{|W_{ij}^l|\leq 1}.\end{aligned} \tag{19}$$

In inference, the weights and activations are quantized as

$$\begin{aligned}\hat{h}_i^l &= h_i(a_i^l) \\ \hat{W}_{ij}^l &= h_i(W_{ij}^l).\end{aligned} \tag{20}$$

For example in BNN, the output $h_i$ is sign function as **Eq** (4), which returns a value from $\{-1, 1\}$ that is consistent with [13]–[15]. In TNN, the output $h_i$ can be written as below:

$$h_i = h_i\left(a_i|\Delta\right) = \begin{cases} 0 & \text{if } |a_i| \leq \Delta \\ \mathrm{sign}(a_i) & \text{otherwise}\end{cases} \tag{21}$$

where $\Delta$ is a positive threshold parameter, which returns a value from $\{-1, 0, 1\}$ based on [17].

The inference of the above two networks can be computed by bitwise operations without multiply-accumulate operations. When considering the bit width of weights and activations exceeds 2-bit, we no longer use the round-off integers scheme like [16] because its inference will require multiply operations. In our QNN scheme, the function output $h_i$ is defined as below:

$$h_i = h_i\left(a_i|\Delta_j\right) = \begin{cases} 0 & \text{if } |a_i| \leq \Delta_q \\ \mathrm{sign}(a_i)2^{-p} & \text{if } |a_i| \leq \Delta_p\end{cases} \tag{22}$$

where $p$ is taken from $q-1$ to 0 in turn, $\Delta_j$ represents $q$ positive threshold parameters, and $q = 2^{n-1} - 1$ for $n$-bit QNN. In this scheme, we can use shift operations instead of multiply operations to infer the entire network, because the weights and activations are powers of two.

### D. The Training and Inference Algorithm for MINW-Net

We present the training and inference algorithm for our MINW-Net as **Algorithm** 1, which is equally applicable to both convolutional layers and fully-connected layers, ignoring the details like Batch Normalization and Pooling layers (of course, they will be used in practice). According to $h_i$ in **Algorithm** 1 which can be chosen from **Eq** (4), **Eq** (21) and **Eq** (22), we can execute BNN, TNN and QNN respectively. As a result, the forward function in this algorithm can obtain the acceleration by bitwise or shift operations.

**Algorithm 1:** Training our MINW-Net with M-bit Inputs (Activations) and N-bit Weights using AQE. The Activations and Weights are quantized based on **Eq** (18) and **Eq** (19) respectively.

**Require:** a minibatch of inputs and targerts $(a_0, a^*)$, learning rate $\eta$, and previous weights $W_l$
**Ensure :** the updated weights $W^{t+1}$
{1. Computing the gradients:}
{1.1 Forward propagation:}
**for** $l = 1$ *to* $n$ **do**
  **if** *'Training'* **then**
    $\hat{W}_l^q \leftarrow \alpha h_i(W_l) + (1-\alpha)W_l$
    $\tilde{a}_l \leftarrow a_{l-1}^q \hat{W}_l^q$
    **if** $l < n$ **then**
      $a_l^q \leftarrow \alpha h_i(\tilde{a}_l) + (1-\alpha)\tilde{a}_l$
    **end**
  **else if** *'Inference'* **then**
    $\hat{W}_l^q \leftarrow h_i(W_l)$
    $\tilde{a}_l \leftarrow a_{l-1}^q \hat{W}_l^q$
    **if** $l < n$ **then**
      $a_l^q \leftarrow h_i(\tilde{a}_l)$
    **end**
  **end**
**end**
{1.2 Backward propagation:}
Computing $g_{a_n} = \frac{\partial L}{\partial a_n}$ based on $a_n$ and $a^*$.
**for** $l = n$ *to* $1$ **do**
  **if** $l < n$ **then**
    $\tilde{a}_l \leftarrow \max(-1, \min(1, \tilde{a}_l))$
    $g_{\tilde{a}_l} \leftarrow g_{a_l^q}$
  **end**
  $g_{a_{l-1}^q} \leftarrow g_{\tilde{a}_l} \hat{W}_l^q$
  $g_{\hat{W}_l^q} \leftarrow g_{\tilde{a}_l}^T a_{l-1}^q$
**end**
{2. Updating the gradients:}
**for** $l = 1$ *to* $n$ **do**
  $W_l^{t+1} \leftarrow \text{update}\left(W_l, g_{\hat{W}_l^q}, \eta\right)$
  $W_l^{t+1} \leftarrow \max\left(-1, \min\left(1, W_l^{t+1}\right)\right)$
**end**

### E. Description of the Special Layers

According to the related work of Courbariaux *et al*. [13], they do not quantize the input of the first convolutional layer, as they find that this behaviour will cause a significant degradation in classification accuracy compared to quantize other convolutional layers. On the other hand, the input of the first convolutional layer is the image itself, and quantizing the image recklessly will bring information loss. For these two reasons and observations, we don't deal with the input of the first convolutional layer. Of course, there is no reason not to quantize the input of other convolutional layers. Unlike the work of Zhou *et al*. [16], they are not quantizing the output of the final fully-connected layer. In our opinion, no other layer is special (including all the input and output of convolutional and fully-connected layers) except the input of the first convolutional layer, so we have quantized them all.

In addition, in the related works of Zhou *et al*. [16] and Han *et al*. [33], they are not quantizing the weights of the first convolutional layer because of the influence on classification accuracy. Nevertheless, from the consideration of computational complexity and parameter capacity, we quantize all the weights across the entire network.

Following the related works of Zhou *et al*. [16] and Courbariaux *et al*. [13], we also add the Batch Normalization (BN) [34] in convolutional and fully-connected layers, as it is conducive to reducing the overall impact of the weight scale and accelerating the training. There are two convolutional layer constructs in MINW-Net, which are {QuantizedConv-pooling-BN-HardTanh} with Pooling layer and {QuantizedConv-BN-HardTanh} without Pooling layer respectively, where HardTanh is the "hard tanh" function: $\max(-1, \min(1, x))$ whose role is reflected on $\mathbf{1}_{|a_i|\leq 1}$ in **Eq** (14).

## IV. Experiment

### A. Histogram of the Asymptotic Behaviour

We have deduced our AQE by **Theorem** 1 and **Theorem** 2 in the previous section. However, the weights and activations used in these two theorems are still continuous values with full-precision. Then, we use **Theorem** 3 to prove that there is an asymptotic behaviour in the training process, that is, these weights and activations will asymptotically approach the quantized value. After the training, they will all be quantized.

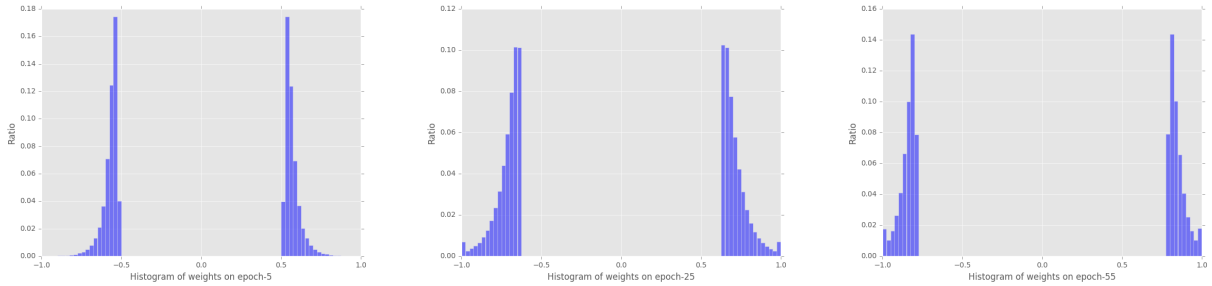
Taking BNN and TNN for example, we observe the evolution of their distribution by extracting the weights and activations of different epochs in a convolutional layer during training. In BNN whose weights and activations will be constrained to $\{-1, 1\}$, the **Fig** 1 2 detail the distribution of weights and activations at epoch 5, 25 and 55. In TNN whose weights and activations will be constrained to $\{-1, 0, 1\}$, the **Fig** 3 4 detail the distribution of weights and activations at epoch 5, 25 and 55. From the figures above, as the number of epochs increases, the weights and activations approach the quantized values gradually, and the proportion of the quantized values is increasing. This experiment also validates the asymptotic behaviour of **Theorem** 3, which means that our asymptotic-quantized estimator is well defined.

### B. Comparison of performance between AQE and STE

We use CIFAR10 and CIFAR100 datesets to explore the performance of AQE, where the network structure is the same BNN.

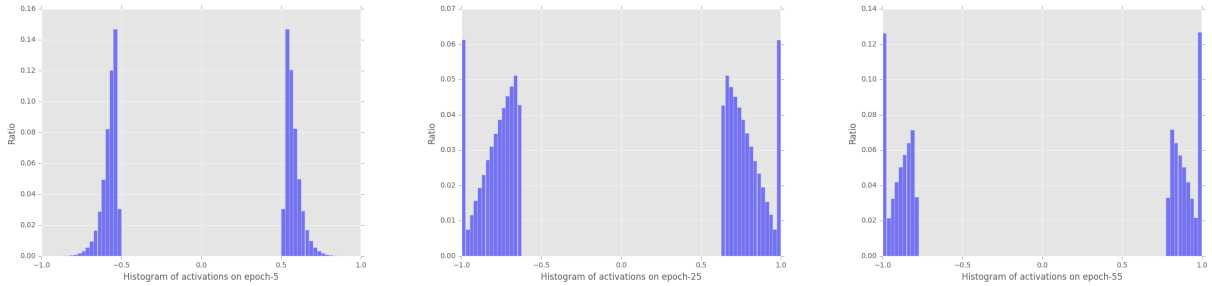
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2966327, IEEE Journal of Selected Topics in Signal Processing





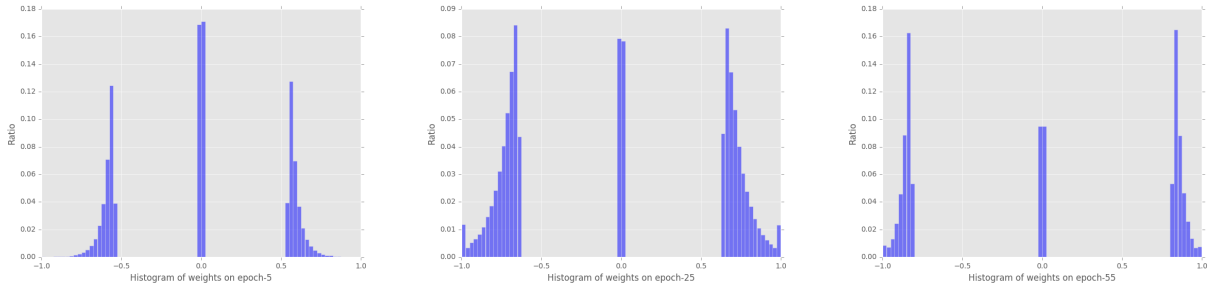

(a) (b) (c)

Fig. 1: Histogram of weights of layer “conv2” of BNN model at epoch 5, 25 and 55, where the y-axis is in logarithmic scale.



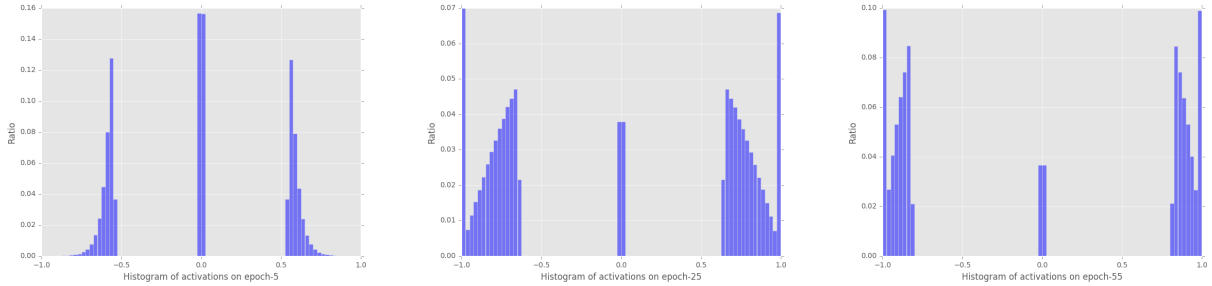

(a) (b) (c)

Fig. 2: Histogram of activations of layer “conv2” of BNN model at epoch 5, 25 and 55, where the y-axis is in logarithmic scale.




(a) (b) (c)

Fig. 3: Histogram of weights of layer “conv2” of TNN model at epoch 5, 25 and 55, where the y-axis is in logarithmic scale.

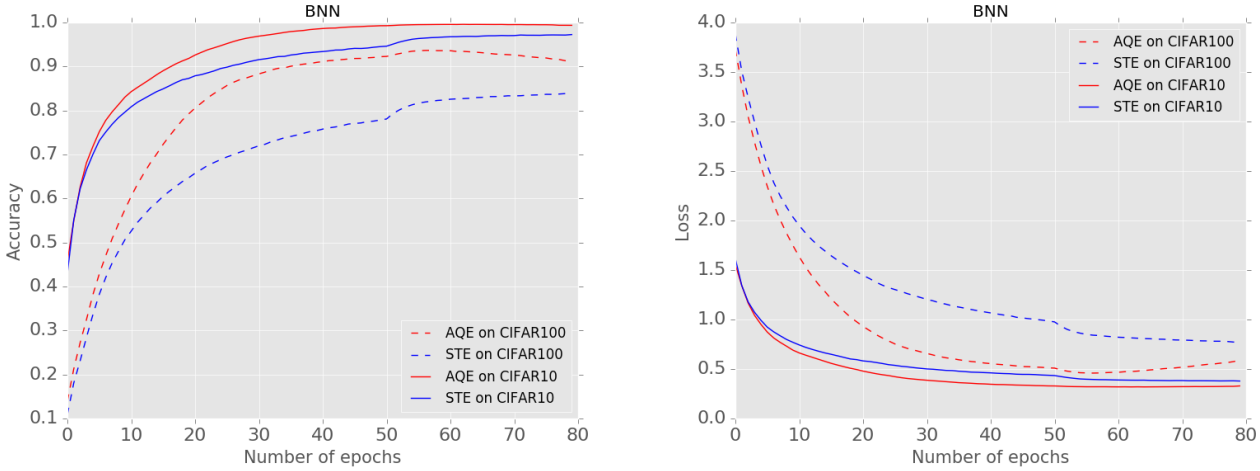



(a) (b) (c)

Fig. 4: Histogram of activations of layer “conv2” of TNN model at epoch 5, 25 and 55, where the y-axis is in logarithmic scale.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2966327, IEEE Journal of Selected Topics in Signal Processing






(a) (b)

Fig. 5: Left: the accuracy curves for STE and AQE approaches applied to the CIFAR10 and CIFAR100 datesets on the training set. Right: the loss curves for STE and AQE approaches applied to the CIFAR10 and CIFAR100 datesets on the training set.

TABLE I: Comparison of classification accuracy on the test set for CIFAR10 with different bitwidth in MINW-Net. We just remove the quantization layers when the precision is 32.

| M-bit Inputs | N-bit Weights | Inference operation | Inference computational precision | Model A Accuracy | Model B Accuracy | Model C Accuracy | Model D Accuracy |
|---|---|---|---|---|---|---|---|
| 1 | 1 | XNOR | 1-bit fixed | 0.884 | 0.851 | 0.847 | 0.844 |
| 1 | 2 | XNOR | 1-bit fixed | 0.889 | 0.866 | 0.859 | 0.855 |
| 1 | 3 | XNOR | 2-bit fixed | 0.897 | 0.873 | 0.868 | 0.867 |
| 1 | 16 | XNOR ADDER | 16-bit floating | 0.901 | 0.876 | 0.870 | 0.871 |
| 2 | 1 | XNOR | 1-bit fixed | 0.891 | 0.875 | 0.868 | 0.867 |
| 2 | 2 | XNOR | 1-bit fixed | 0.898 | 0.879 | 0.873 | 0.871 |
| 2 | 3 | XNOR | 2-bit fixed | 0.903 | 0.888 | 0.888 | 0.881 |
| 2 | 16 | XNOR ADDER | 16-bit floating | 0.909 | 0.890 | 0.891 | 0.887 |
| 3 | 1 | XNOR | 2-bit fixed | 0.890 | 0.889 | 0.880 | 0.880 |
| 3 | 2 | XNOR | 2-bit fixed | 0.895 | 0.892 | 0.887 | 0.888 |
| 3 | 3 | SHIFT | 2-bit fixed | **0.906** | **0.898** | **0.891** | **0.893** |
| 3 | 16 | SHIFT ADDER | 16-bit floating | 0.911 | 0.901 | 0.896 | 0.895 |
| 16 | 1 | XNOR ADDER | 16-bit floating | 0.908 | 0.890 | 0.886 | 0.887 |
| 16 | 2 | XNOR ADDER | 16-bit floating | 0.908 | 0.894 | 0.892 | 0.891 |
| 16 | 3 | SHIFT ADDER | 16-bit floating | 0.911 | 0.901 | 0.904 | 0.903 |
| 32 | 32 | MAC | 32-bit floating | 0.918 | 0.910 | 0.912 | 0.910 |

TABLE II: Comparison of classification accuracy on the test set for CIFAR100 with different bitwidth in MINW-Net. We just remove the quantization layers when the precision is 32.

| M-bit Inputs | N-bit Weights | Inference operation | Inference computational precision | Model A Accuracy | Model B Accuracy | Model C Accuracy | Model D Accuracy |
|---|---|---|---|---|---|---|---|
| 1 | 1 | XNOR | 1-bit fixed | 0.596 | 0.554 | 0.546 | 0.530 |
| 1 | 1 | XNOR | 1-bit fixed | 0.596 | 0.552 | 0.535 | 0.515 |
| 1 | 2 | XNOR | 1-bit fixed | 0.602 | 0.559 | 0.551 | 0.542 |
| 1 | 3 | XNOR | 2-bit fixed | 0.611 | 0.564 | 0.557 | 0.550 |
| 1 | 16 | XNOR ADDER | 16-bit floating | 0.620 | 0.569 | 0.560 | 0.555 |
| 2 | 1 | XNOR | 1-bit fixed | 0.618 | 0.586 | 0.580 | 0.564 |
| 2 | 2 | XNOR | 1-bit fixed | 0.622 | 0.590 | 0.588 | 0.569 |
| 2 | 3 | XNOR | 2-bit fixed | **0.631** | **0.603** | **0.592** | **0.582** |
| 2 | 16 | XNOR ADDER | 16-bit floating | 0.633 | 0.607 | 0.595 | 0.585 |
| 3 | 1 | XNOR | 2-bit fixed | 0.593 | 0.565 | 0.548 | 0.544 |
| 3 | 2 | XNOR | 2-bit fixed | 0.606 | 0.582 | 0.565 | 0.560 |
| 3 | 3 | SHIFT | 2-bit fixed | 0.618 | 0.601 | 0.581 | 0.573 |
| 3 | 16 | SHIFT ADDER | 16-bit floating | 0.621 | 0.610 | 0.585 | 0.579 |
| 16 | 1 | XNOR ADDER | 16-bit floating | 0.631 | 0.618 | 0.597 | 0.584 |
| 16 | 2 | XNOR ADDER | 16-bit floating | 0.635 | 0.622 | 0.608 | 0.598 |
| 16 | 3 | SHIFT ADDER | 16-bit floating | 0.640 | 0.630 | 0.619 | 0.614 |
| 32 | 32 | MAC | 32-bit floating | 0.646 | 0.634 | 0.630 | 0.623 |

The two CIFAR datasets [35] consist of color natural images with 32×32 pixels, respectively 50,000 training images and 10,000 test images, and we hold out 5,000 training images as a validation set from training set. CIFAR10 consists of images organized into 10 classes, and CIFAR100 into 100 classes. We adopt a standard data augmentation scheme (random corner cropping and random flipping) that is widely used for these two datasets [36]–[39]. We normalize the images using the channel means and standard deviations in preprocessing.

In this experiment, we perform the classification accuracy by comparing the AQE with STE on CIFAR10 and CIFAR100. The evaluation of the experiment is based on ConvNet, which is used by Courbariaux *et al*. [13], where the stochastic neuron output is **Eq** (4) in STE and **Eq** (6) in AQE respectively. On the **Fig** 5, we test the two estimators using the same conditions[1] (including learning rate, network structure and number of epochs), and conclude that our AQE performs better in both accuracy and loss than the STE.

[1]We need to correct an ambiguity. In the previous section, we have mentioned that our AQE uses the weights and activations with full-precision on inference during the training process, as they will participate in the loss calculation of the updated parameters. But in this experiment, in order to be consistent with STE, we will use the binarized weights and activations to obtain accuracy and loss.

### C. Low-bitwidth Exploration

In this section, we use AQE to explore the best configuration for different combinations of weights and activations precisions on CIFAR datasets.

**ConvNet:** The basic structure of ConvNet [13] consists of two layers, one of which is feature extraction layer. The input of each neuron is connected to the local receptive field of the previous layer, and the local features are extracted. The second is the feature mapping layer, Sigmoid or ReLU function is used as the activation function of ConvNet in feature mapping structure, which makes feature mapping have displacement invariance. Each convolutional layer in ConvNet is followed by a pooling layer used to calculate local average and quadratic extraction.

**ResNet:** The main idea of ResNet [40] is to add a direct connection channel to the network, namely Highway Network [41]. The previous network structure is a non-linear transformation of the input, while Highway Network retains a certain proportion of the output of the previous layer. Similarly, ResNet allows the original input information to be passed directly to the later layers, whose idea paves the way for a very deep network (more than 100 layers) to train.

**DenseNet:** It is inspired by ResNet and Highway Network, and proposes to transform the input of each convolutional layer into the splicing of the output of all previous layers [42]. Such

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2966327, IEEE Journal of Selected Topics in Signal Processing



a dense connection makes it possible for each layer to take advantage of all the features previously learned, without the need for repetitive learning. At the same time, it imitates the structure of ResNet to make the gradient spread better, which makes it more convenient to train deep networks.

TABLE III: Comparison of test error on CIFAR10 (100) between 3-bit weights and 32-bit float weights, where the results are given based on ResNet and DenseNet.

| Network | Depth | Dataset | Bitwidth | Test error (%) |
|---|---|---|---|---|
| **ResNet** [40] | 110 | CIFAR10 | 32 (float) | 6.61 |
| | | CIFAR10 | 3 | 7.05 (+0.44) |
| | | CIFAR100 | 32 (float) | 35.87 |
| | | CIFAR100 | 3 | 37.16 (+1.29) |
| **DenseNet** [42] | 100 | CIFAR10 | 32 (float) | 4.51 |
| | | CIFAR10 | 3 | 5.21 (+0.70) |
| | | CIFAR100 | 32 (float) | 22.27 |
| | | CIFAR100 | 3 | 23.70 (+1.43) |

For the convolutional and fully-connected layers in our MINW-Net, we have listed M-bit Inputs and N-bit Weights, inference operation, inference computational precision and classification accuracy of different models in **Table** I II. When the bitwidth of inputs and weights are both 1, our MINW-Net is degenerated to BNN, and its inference operation is XNOR and computational precision is 1-bit fixed-point number. When the bitwidth of inputs and weights are both 2, our MINW-Net is degenerated to TNN, and the inference operation and computational precision are consistent with BNN. In this experiment, **Eq** (4) is used for 1-bit bitwidth where the quantized values are constrained to $\{-1, 1\}$, **Eq** (21) is used for 2-bit bitwidth where the quantized values are constrained to $\{-1, 0, 1\}$ and **Eq** (22) is used for 3-bit bitwidth where the quantized values are constrained to $\{-1, -2^{-1}, -2^{-2}, 0, 2^{-2}, 2^{-1}, 1\}$.

We use several CNN models on CIFAR datasets to evaluate the performance of MINW-Net. Model A is the ConvNet that costs about 1.23 M parameter capacity for 1-bit weights, and it consists of six convolutional layers and three full-connected layers. Model B is derived from Model A by reducing the number of channels by half for all six convolutional layers, which costs about 0.54 M parameter capacity. Model C continues to reduce the number of channels by half for all three fully-connected layers based on Model B, which costs 0.30 M parameter capacity. Model D has the least parameter capacity here that costs 0.21 M by reducing the number of channels by half for all three fully-connected layers based on Model C. These models are trained with a batch size of 256 and a learning rate of 0.01 whose learning rule is ADAM [43] with the exponential decay. The classification accuracy on the listed test set is the results of the model training over 200 epochs.

The first two lines listed in **Table** II are the same network structure with 1-bit weights and 1-bit activations, where the first line is the prediction accuracy trained from scratch with our AQE and the second line is the prediction accuracy trained from scratch with the STE. We evaluate the performance of AQE by 1-bit weights and activations MINW-Net on CIFAR100 dataset. As the size of the model decreases, the advantages of using AQE gradually emerge. Due to the reduction in network redundancy, the QNNs using AQE have a higher accuracy than using STE. We achieve the prediction accuracy improvement of 0.0%, 0.2%, 1.1% and 1.5% on Model A, B, C and D respectively.

The trends in **Table** I II show that the number of channels affects the prediction accuracy. Although MINW-Net with low-precision weights and activations can cause degradation in prediction accuracy, the tiny degradation in accuracy can be ignored compared to the much reduced resource requirement and increased computational efficiency. For CIFAR10, the best performance is MINW-Net with 3-bit weights and 3-bit activations. We achieve the prediction accuracy degradation of 1.2%, 1.2%, 2.1% and 1.7% on Model A, B, C and D respectively compared with 32-bit counterparts. For CIFAR100, the best performance is MINW-Net with 3-bit weights and 2-bit activations. We achieve the prediction accuracy degradation of 1.5%, 3.1%, 4.0% and 4.1% on Model A, B, C and D respectively compared with 32-bit counterparts. When running inference, the inference operation is determined as XNOR, SHIFT or MAC according to the smaller bitwidth between weights and activations, and the inference computational precision is based on the larger bitwidth between weights and activations, where MAC represents multiply accumulate operation.

For ResNet, we use a weight decay of 0.0001, a momentum of 0.9 and BN without dropout. The model is trained with a mini-batch size of 128 and a learning rate of 0.1, divided by 10 at 32k and 38k iterations, and terminates at 64k iterations. Seen from the experiments of ResNet in **Table** III, our MINW-Net achieves the test error rates of 7.05% on CIFAR10, 37.16% and on CIFAR100, just rises 0.44% on CIFAR10 and 1.29% on CIFAR100 compared with 32-bit float ResNet.

For DenseNet, we use a weight decay of 1e-4, a momentum of 0.9 and BN without dropout. The initial learning rate for this model is 0.1, and the model is divided by 10 at 50% and 75% of the total number of training periods. And we use a batch size of 64 for a total of 300 periods on CIFAR. Compared with 32-bit float DenseNet, the test error of our MINW-Net on CIFAR10 increases by 0.70% (from 4.51% to 5.21%) and on CIFAR100 increases by 1.43% (from 22.27% to 23.70%), as shown in **Table** III.

### *D. ImageNet*

ImageNet data set is a large image data set organized by professor Fei Fei Li of Stanford university covering all aspects of computer vision. The dataset used for image classification is the ILSVRC2012 [55] image classification dataset, which identifies the main objects in the image. To further evaluate the effect of our MINW-Net on ILSVRC2012 image classification dataset which contains 1.2 million high-resolution natural images from 1000 categories, we resize these images to 224×224 pixels before input them to the network. In the following experiments, we use top-1 and top-5 accuracy to measure our single-crop evaluation results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2966327, IEEE Journal of Selected Topics in Signal Processing



TABLE IV: Comparison of classification accuracy on ImageNet test set with different bitwidths of weights and activations. Top-1 and top-5 accuracy for single-crop evaluation results are given based on AlexNet. Note that the "Ours" results are implemented by our MINW-Net. Other results are reported by [44]. We quantize the same layers of AlexNet to low-precision, just like BNN [13], BC [14], TWN [17], TNN [45] and DoReFa-Net [16]. Top-1 accuracy for full-precision AlexNet is 56.6% and top-5 accuracy is 80.2%.

| **Reduce Precision Method** | | **Bitwidth** | | **Inference Operation** | **Top-1 Accuracy loss vs. 32-bit float (%)** | **Top-5 Accuracy loss vs. 32-bit float (%)** |
|---|---|---|---|---|---|---|
| | | **Weights** | **Inputs** | | | |
| **Reduce Weights** | **BinaryConnect** [14] | 1 | 32 (float) | XNOR ADDER | 19.8 | 18.2 |
| | Ours | 1 | 16 (float) | XNOR ADDER | 8.0 | 6.8 |
| | **Ternary Weight Network** [17] | 2 | 32 (float) | XNOR ADDER | 3.7 | 3.6 |
| | Ours | 2 | 16 (float) | XNOR ADDER | 3.0 | 2.5 |
| **Reduce Weights and Inputs** | **Binary Neural Network** [13] | 1 | 1 | XNOR | 28.7 | 29.8 |
| | **DoReFa-Net** [16] | 1 | 1 | XNOR | 17.1 | 19.1 |
| | Ours | 1 | 1 | XNOR | 21.8 | 20.1 |
| | **DoReFa-Net** [16] | 1 | 2 | XNOR | 10.5 | 11.4 |
| | Ours | 1 | 2 | XNOR | 9.8 | 10.1 |
| | **Ternary Neural Network** [45] | 2 | 8 (float) | XNOR ADDER | 7.6 | 7.4 |
| | Ours | 2 | 8 (float) | XNOR ADDER | 3.2 | 2.9 |
| | **DoReFa-Net** [16] | 8 | 8 | MAC | 3.6 | 3.4 |
| | Ours | 3 | 8 (float) | SHIFT ADDER | **0.6** | **0.7** |

**AlexNet:** It is the first CNN structure to show success and wide attention on the ImageNet classification task. The architecture consists of 61 million parameters and 65,000 neurons, and is composed of 5 convolutional layers and 2 fully connected layers [1]. The output layer is a 1000 channel softmax. We use AlexNet in combination with BN.

In the training, the input image is cropped with a random size, the clipping is adjusted to 256×256 pixels, and then 224×224 images are extracted randomly for training. We train MINW-Net for 50 epochs based on AlexNet, with a batch size of 128. In our AlexNet implementation, we use an ADAM optimizer with faster convergence and the learning rate of 1e-4. We replace the "Local Contrast Renormalization" layer with the "Batch Normalization" layer. At inference, we use 224×224 center crop for forward propagation.

The ablation experiments are listed in **Table** IV. It is reported in [15] that the accuracy score of baseline AlexNet model is 56.6% for top-1 and 80.2% for top-5. In the ablation study, in addition to the difference of quantization methods, we strictly control the consistency of variables such as network structure, bit width and quantization layers. In experiments of "1-1" v.s. "1-1" for BNN, "1-16" v.s. "1-32" for BC, "2-16" v.s. "2-32" for TWN, "2-8" v.s. "2-8" for TNN and "3-8" v.s. "8-8" for DoReFa-Net, our MINW-Net top-1 accuracy improved by 6.9%, 11.8%, 0.7%, 4.4% and 3.0% respectively. For "3-8" v.s. "32-32", our MINW-Net only reduces the top-1 accuracy by 0.6%.

In **Table** V, we list our MINW-Net results for top-1 and top-5 and compare against multiple recent works on ResNet-18. The ablation experiments employ all fixed-point weights quantization and part fixed-point activations quantization: SR+DR [49], [50], TWN [17], BWN [15], XNOR-Net [15], DoReFa-Net [16], INQ [32], HWGQ [54], LR-Net [51], SYQ [53], ELQ [52] and rounding. The instructions of quantization are mentioned explicitly by footnotes.

## V. Conclusion and Future Work

In this paper, we have introduced AQE, a novel estimator to propagate asymptotic-estimated gradient through stochastic neuron output, which is correlated with both the original full-precision value and the quantized value, in neural networks involving noise sources. Since our AQE has the asymptotic behaviour, the neuron output will gradually approach the quantized value as the training epochs increase (the actual performance is that the ratio of the original full-precision value is decreased and the ratio of the quantized value is increased). After the training, the output will completely convert to the quantized value. With the hyper-parameter $\alpha = \frac{1}{2}$, we have proven that our AQE will degenerate into STE and unbiased estimator. We have proposed MINW-Net, a quantized neural network with M-bit Inputs and N-bit Weights, which is trained by AQE. All of the activations and weights during forward passes are quantized, because there is no layer is special in MINW-Net. When the smaller bitwidth between weights and activations are 1 or 2, all the convolution operations in inference can be replaced by XNOE operations. Similarly, all

TABLE V: Top-1 and top-5 error (%) with ResNet18 on ImageNet. Note that the "Ours" results are implemented by our MINW-Net. Other results are reported by [46]–[48].

| ResNet18 | | | | | | |
|---|---|---|---|---|---|---|
| **Method** | **# Bits weights/inputs** | [1] | [2] | [3] | **Top-1** | **Top-5** |
| **Original** | 32/32 | - | - | - | 30.46 | 10.81 |
| **Rounding** | 8/8 | - | - | - | 30.22 | 10.60 |
| | 6/6 | - | - | - | 31.61 | 11.32 |
| | 5/5 | - | - | - | 36.97 | 14.95 |
| | 4/4 | - | - | - | 78.79 | 57.10 |
| **LQ-Nets** [48] | 2/32 | ✓ | ✓ | - | 32.00 | 11.80 |
| | 3/32 | ✓ | ✓ | - | 30.70 | 11.10 |
| | 1/2 | ✓ | ✓ | - | 37.40 | 15.50 |
| **SR+DR** [49], [50] | 8/8 | - | - | - | 31.83 | 11.48 |
| | 6/6 | - | - | - | 40.75 | 16.90 |
| | 5/5 | - | - | - | 45.48 | 20.16 |
| **LR Net** [51] | 1/32 | ✓ | ✓ | - | 40.10 | 17.70 |
| | 2/32 | ✓ | - | - | 36.50 | 15.20 |
| **ELQ** [52] | 1/32 | - | - | - | 35.28 | 13.96 |
| | 2/32 | - | - | - | 32.48 | 11.95 |
| **SYQ** [53] | 1/8 | ✓ | ✓ | - | 37.10 | 15.40 |
| | 2/8 | ✓ | ✓ | - | 32.30 | 12.20 |
| **TWN** [17] | 2/32 | - | - | - | 38.20 | 15.80 |
| **INQ** [32] | 5/32 | - | - | - | 31.02 | 10.90 |
| **BWN** [15] | 1/32 | - | - | - | 39.20 | 17.00 |
| **XNOR-Net** [15] | 1/1 | - | - | - | 48.80 | 26.80 |
| **HWGQ** [54] | 1/2 | ✓ | ✓ | ✓ | 40.40 | 17.80 |
| **DoReFa-Net** [16] | 1/4 | ✓ | ✓ | - | 40.80 | 18.50 |
| **MINW-Net** (Ours) | 2/32 | - | - | - | 36.36 | 15.10 |
| | 1/32 | - | - | - | 37.60 | 15.81 |
| | 1/2 | ✓ | ✓ | - | 42.16 | 18.90 |
| | 1/1 | ✓ | - | - | 47.87 | 26.04 |

the convolution operations are replaced by SHIFT operations when the smaller bitwidth is 3.

By experiments, we have verified the asymptotic behaviour of our AQE. And the BNN with AQE can achieve a prediction accuracy 1.5% higher than the BNN with STE. The MINW-Net, which is trained from scratch by AQE, can achieve comparable classification accuracy as 32-bit counterparts on CIFAR test sets. Extensive experimental results on ImageNet dataset show great superiority of the proposed AQE and our MINW-Net achieves comparable results with other state-of-the-art QNNs.

Our future work should explore how to achieve the inference of MINW-Net on FPGAs where the inference operation of XNOR or SHIFT will be applied on hardware platform.

[1] First layer not quantized
[2] Last layer not quantized
[3] Modified architecture

## REFERENCES


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, pp. 1097–1105, 2012.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," pp. 1–9, 2014.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[4] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

[5] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1891–1898, 2014.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[8] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[10] A. Tulloch and Y. Jia, "High performance ultra-low-precision convolutions on mobile devices," *arXiv preprint arXiv:1712.02427*, 2017.

[11] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 764–775. IEEE, 2018.

[12] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 688–698. IEEE, 2018.

[13] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[14] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, pp. 3123–3131, 2015.

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

[16] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[17] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[18] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems*, pp. 345–353, 2017.

[19] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[20] E. Park, D. Kim, S. Yoo, and P. Vajda, "Precision highway for ultra low-precision quantization," *arXiv preprint arXiv:1812.09818*, 2018.

[21] J. Choi, P. I.-J. Chuang, Z. Wang, S. Venkataramani, V. Srinivasan, and K. Gopalakrishnan, "Bridging the accuracy gap for 2-bit quantized neural networks (qnn)," *arXiv preprint arXiv:1807.06964*, 2018.

[22] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[23] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. 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)*, pp. 722–737, 2018.

[24] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Towards effective low-bitwidth convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7920–7928, 2018.

[25] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[27] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[28] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[29] J. C. Spall *et al.*, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE transactions on automatic control*, vol. 37, no. 3, pp. 332–341, 1992.

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

[31] I. R. Fiete and H. S. Seung, "Gradient learning in spiking neural networks by dynamic perturbation of conductances," *Physical review letters*, vol. 97, no. 4, p. 048104, 2006.

[32] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[33] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[36] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[37] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, pp. 562–570, 2015.

[38] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[39] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, pp. 2377–2385, 2015.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[41] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[42] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, p. 3, 2017.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[44] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[45] N. Mellempudi, A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey, "Ternary neural networks with fine-grained quantization," *arXiv preprint arXiv:1705.01462*, 2017.

[46] C. Louizos, M. Reisser, T. Blankevoort, E. Gavves, and M. Welling, "Relaxed quantization for discretized neural networks," *arXiv preprint arXiv:1810.01875*, 2018.

[47] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

[48] D. Zhang, J. Yang, D. Ye, and G. 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.

[49] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, pp. 1737–1746, 2015.

[50] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5784–5789, 2018.

[51] O. Shayer, D. Levi, and E. Fetaya, "Learning discrete weights using the local reparameterization trick," *arXiv preprint arXiv:1710.07739*, 2017.

[52] A. Zhou, A. Yao, K. Wang, and Y. Chen, "Explicit loss-error-aware quantization for low-bit deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9426–9435, 2018.

[53] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4300–4309, 2018.

[54] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5918–5926, 2017.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.