Efficient Quantization for Neural Networks with Binary Weights and Low Bitwidth Activations

Kun Huang,[1] **Bingbing Ni,**[1,2,3*] **Xiaokang Yang**[1,3]
[1]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China
[2]SJTU-UCLA Joint Research Center on Machine Perception and Inference, Shanghai Jiao Tong University, China
[3]Shanghai Institute for Advanced Communication and Data Science
hunterkun94@gmail.com, {nibingbing, xkyang}@sjtu.edu.cn

Abstract

Quantization has shown stunning efficiency on deep neural network, especially for portable devices with limited resources. Most existing works uncritically extend weight quantization methods to activations. However, we take the view that best performance can be obtained by applying different quantization methods to weights and activations respectively. In this paper, we design a new activation function dubbed CReLU from the quantization perspective and further complement this design with appropriate initialization method and training procedure. Moreover, we develop a specific quantization strategy in which we formulate the forward and backward approximation of weights with binary values and quantize the activations to low bitwdth using linear or logarithmic quantizer. We show, for the first time, our final quantized model with binary weights and ultra low bitwidth activations outperforms the previous best models by large margins on ImageNet as well as achieving nearly a $10.85\times$ theoretical speedup with ResNet-18. Furthermore, ablation experiments and theoretical analysis demonstrate the effectiveness and robustness of CReLU in comparison with other activation functions.

Introduction

Deep neural networks have made unparalleled progress in a variety of computer vision tasks such as image classification (Krizhevsky, Sutskever, and Hinton 2012), object detection (Girshick 2015) and semantic segmentation (Long, Shelhamer, and Darrell 2015). The promising results DNNs have achieved are often built on a vast number of parameters (e.g., AlexNet comes along with nearly 61 million real-valued parameters and 1.5 billion floating-point operations to classify an image (Guo et al. 2017)). However, their complexity is an impediment to widespread deployment on portable devices with limited memory or computational resources. Recently, substantial research efforts are invested to alleviate this problem. One common method proven effective is to quantize the full-precision network directly, which can greatly save memory requirement and speed up models using specialized hardware implementation (e.g. FPGA and Google's TPU (Jouppi et al. 2017)).

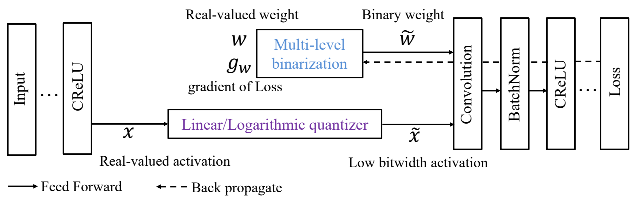


Figure 1: A graphical illustration of the training process of our quantized neural networks with binarized weights and low bitwidth activations.

Some prior works first focus on quantizing weights to reduce the model size. In particular, (Courbariaux, Bengio, and David 2015) introduce BinaryConnect, a method which forces the weights to be binary during the forward and backward propagations. They obtained near state-of-the-art results on the small datasets like MNIST, CIFAR-10. Later, (Li, Zhang, and Liu 2016) proposed ternary weight networks (TWNs) with weights quantized to $\{-\alpha, 0, +\alpha\}$, which achieved better performance on large dataset due to the increased weight precision and scaling factors. However, it is less effective from the computation saving perspective due to the activations are still left to be full-precision (i.e., the input to convolution layers). Recently, BNN (Courbariaux et al. 2016) and XNOR-Net (Rastegari et al. 2016) explored both weights and activations binarization, which can not only reduce model size 32 times compared to full-precision version but cut down execution time since expensive dot-products can be replaced by bitwise operations. However, they directly quantize both weights and activations in the same manner without considering different characteristics of them. As a result, both works suffer from nontrivial losses in large-scale classification tasks on ImageNet (from 56.6% to 27.9% and 44.2% respectively). In this paper, we argue that best performance can be obtained by applying different quantization methods to weights and activations respectively.

For the issue of activations quantization, we observe that the unbounded output of ReLU function is disadvantageous for upcoming quantization. Hence, it is necessary to limit the dynamic range of activations to minimize quantization error. Motivated by this observation, we propose a novel ac-

[*]Corresponding author: Bingbing Ni

tivation function CReLU from the quantization view, which adaptively bound the ReLU output by a learnable clamping parameter to keep the balance between clamping error and quantization error. In addition, we further quantize the activations to low bitwidth representations so as to allow the trade-offs in different combinations of bitwidth vs. model accuracy.

For the issue of weights quantization, XNOR-Net proposes to introduce a scale factor for each element of the weight filter during binarization. As the binarized weights cannot be used to accumulate the high precision gradients, the real-valued weights need to be updated using SGD. However, we find that the impact of this scale factor is completely omitted in the calculation of the gradient with respect to the real weight in XNOR-Net. To eliminate the gradient mismatch, we derive a more effective gradient in the back-propagation. Moreover, we further propose multi-level binarization to approximate the pre-trained weights with binary values in the forward pass.

The major contributions of this work are summarized as follows:

- we design a new activation function called clamping rectified linear unit (CReLU) from the quantization perspective and complement this design with appropriate initialization method and training procedure.
- we propose a multi-level binarization method and linear or logarithmic quantizer for simultaneously binarizing the weights and quantizing the activations to low bitwdth. The overall quantization process is illustrated in Figure 1.
- our final quantized model with binary weights and ultra low bitwidth activations outperforms the previous best models on ImageNet, with nearly $10.85\times$ theoretical speedup with ResNet-18.
- ablation experiments and theoretical analysis further demonstrate the effectiveness and robustness of CReLU in comparison with other activation functions.

Related Work

Network Pruning Pruning is a straightforward method and has been widely studied to compress the networks by pruning parameters or channels. Parameter pruning removes the unimportant connections or neurons which value under a threshold. (Han et al. 2015) presented an simple method to prune the parameters and reduced the model size by $9\times$ and $16\times$ for AlexNet and VGG-16. Although it is effective in model compression, such unstructured pruning may not translate directly to faster inference since specialized hardwares and softwares are needed for a network with intra-kernel sparsity. Channel pruning instead operates at the level of feature maps and channel to avoid the limitations of non-structured pruning above. The key of this method is how to measure the importance of each filters. (Li et al. 2016) pruned unimportant filters by calculating their absolute weight sum. (Molchanov et al. 2016) adopted Taylor expansion to approximate the influence of each filter to loss function. However, this kind of methods all need to change the structure of network. By contrast, any network can automatically benefit from our quantization schemes without modifying the network.

Compact Model The main point of this methods is to design more compact architecture for model compression and acceleration. Xception (Chollet 2016) first introduce depthwise separable convolution to factorize a standard convolution into depthwise convolution and pointwise convolution. Depthwise convolution is responsible to extract features while pointwise convolution merge features. Such architecture is facilitated to meet the resource constraints for an on-device or embedded applications. Beyond these works, (Zhang et al. 2017a) propose a general group convolution algorithm and show that Xception is the special case of their method. Group operation will block the information flow between different group convolutions. Recently, (Zhang et al. 2017b) present ShuffleNet, which introduce channel shuffle operations to maintain the connections between groups to solve that problem. Our quantization method is orthogonal to this line of approach. Hence, it can be jointly applied to achieve better compression results.

Network Quantization There are two meanings about quantization term in the neural networks. On one hand, it refers to a many-to-few mapping, which groups weights with similar values to reduce the number of free parameters. For example, (Chen et al. 2015) hashed weights into different groups before training. The weights are shared within each group and only the shared weights and hash indexes need to be stored. (Han et al. 2015) proposed weight sharing method to compress the network by reducing the number of bits required to represent each weights. The model parameters they store are the index of cluster centroids instead of the real values. Hence, they can quantize model to 5-bits or 8-bits (32 or 256 cluster centroids respectively) compared to 32-bits full-precision counterpart. On the other hand, it means transferring a full-precision model into its low-bitwidth representation, our method is in line with this meaning since a low-precision model not only reduces the memory storage, but also is more computational efficient. (Courbariaux et al. 2016) first propose BNN, a purely BinaryNet which constrains both weights and activations to $\{-1, +1\}$. They obtain near state-of-art results on small datasets while are not very successful on large-scale datasets. Therefore, (Rastegari et al. 2016) propose XNOR-net, which incorporates a real coefficient to compensate for the pure binarization error in BNN. They apply the method to the large-scale ImageNet dataset and achieve better performance than previous works. However, it's straightforward extension of weight quantization approach to activation makes the binarization procedure complex and inefficient in that it needs to calculate more scale factors for activation compared to weight. In order to mitigate this issue, we propose an alternative that removes the scale factor in activations, and replace the ReLU activation function with a new one called CReLU, which does not incur accuracy degradation but is simpler and more efficient than XNOR-net.

Our Approach

In the section, we present in detail how we binarize the weights and quantize the activations to low bitwidth. Let us start with the notion in this paper. For a convolutional layer, we define the output $\mathbf{y} = \mathbf{x} * \mathbf{w}$, where the input $\mathbf{x} \in \mathbb{R}^{c_{in} \times w_{in} \times h_{in}}$, weight filter $\mathbf{w} \in \mathbb{R}^{c_{out} \times c_{in} \times w \times h}$, the output $\mathbf{y} \in \mathbb{R}^{c_{out} \times w_{out} \times h_{out}}$, respectively.

Low Bitwidth Activations

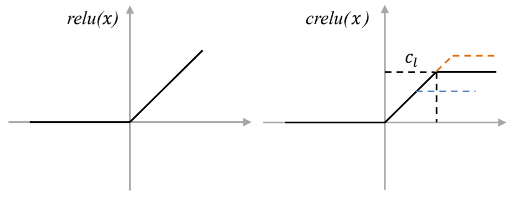
To implement convolution efficiently and speed up CNNs in the run time, it is indispensable to quantize the activations. Some previous works such as BinaryNet, XNOR-Net binarize the activations in the same way as weights by using the sign function in the forward pass. However, (Zhou et al. 2016) claims that they fail to reproduce results and find severe prediction accuracy degradation when applied on ImageNet models like AlexNet. We argue that two reasons can account for this failure. On one hand, the non-smooth and non-convex characteristic of the sign function makes it hard for gradient information to flow from one layer to the next layer. On the other hand, it is computationally intractable. Specifically, in order to convolve weight filter $\mathbf{w}$ with the input tensor $\mathbf{x}$, XNOR-Net has to compute the scaling factor β for all possible sub-tensors with same size as $\mathbf{w}$ in $\mathbf{x}$, which makes the convolution operation inefficient and hard to implementation in most deep learning frameworks. To alleviate the issues above, we resort to an elegant alternative for activation function.

Clamping Rectified Linear Unit As the first problem we discussed, sign function is not suitable for activation quantization. The experimental results in (Cai et al. 2017) further verify our analysis. Thus, it seems more sensible to rely on ReLU when it comes to activation function. However, its unbounded characteristic is disadvantageous for quantization since some outliers on the tail of distribution can lead to large quantization errors. This problem has been alleviated by some recent works. (Zhou et al. 2016) simply clamp activations to [0, 1] for all layers. (Cai et al. 2017) adopt K-means algorithm to obtain the optimal clamping parameters for all layers offline, which is based on the observation that dot-products through batchnorm layer are close to a standard Gaussian distribution of zero mean and unit variance. However, in both works, the values of clamping parameters are deterministic and remain fixed, which could not adapt to the change of activations during the training process. Hence, we redesign a new activation function called clamping rectified linear unit (CReLU) to improve flexibility of the clamping threshold. It is illustrated in the right part of Figure 2.

Let variable x represent the input, and the forward pass function of CReLU is defined as:

$$crelu(x) = \begin{cases} c_l, & x \in (c_l, +\infty) \\ x, & x \in (0,\ c_l] \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where c_l is the l^{th} layer-wise learnable parameter, which places an upper-bound on the outputut.



(a) ReLU (b) CReLU

Figure 2: Illustration of (a) ReLU and (b) CReLU activation function

The backward pass function of CReLU is given by:

$$\begin{aligned} \frac{\partial crelu(x)}{\partial x} &= \begin{cases} 1, & x \in (0, c_l] \\ 0, & \text{otherwise} \end{cases} \\ \frac{\partial crelu(x)}{\partial c_l} &= \begin{cases} 1, & x \in (c_l, +\infty) \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

Therefore, CReLU could utilize the strength of backpropagation to adjust itself dynamically throughout the training process and achieve the optimal clamping parameter in the inference time. For interested readers, we briefly provide theoretical analysis on the effectivenss of CReLU in the supplementary material.

Parameter Training with CReLU In this subsection, we provide a brief analysis on the initialization method and appropriate training procedure for CReLU to favor upcoming quantization.

1) From the back propagation view, it is clear that, if c_l is initialized to a very small value, more input activations fall into the range (c_l, $+\infty$) with the nonzero gradient contributing to c_l, leading to unstable c_l parameter update in the early training, potentially causing large accuracy degradation. On the other hand, if c_l is initialized to a very large value, more input activations fall into the range (-∞, c_l) with the zero gradient contributing to c_l, the gradient becomes too small to update c_l parameter in the early training, which may be stuck at a large value, potentially suffering large quantization error.

2) From the forward propagation view, as the clamping value c_l increases, CReLU is closer to ReLU activation function. Therefore, larger range of activations can be passed to the next layer causing less clamping error, which is determined by $\max(x - c_l, 0)$. On the other hand, the increased c_l means a wider dynamic range thus incurring larger quantization error, which can be determined by $\frac{c_l}{2(2^k-1)}$ when the truncated activation output is linearly quantized to k-bits. This imposes the challenge of finding a proper clamping parameter to keep the balance between clamping error and quantization error.

In conclusion, it is intuitive to initialize c_l with a relatively large value to cover a wide dynamic range and avoid unstable parameter update at the beginning of each layer. Then we jointly optimize the network weights and these clamping parameters with enforcing regularization into the latter, which enables us to reduce the value of c_l so as to alleviate quantization error and lead to fast convergence. In practice, we start with an initial value of 8 and the training objective of our approach is given by

$$\ell(w, b) = \mathcal{C}(w, b) + \lambda \sum_{l=1}^{L} g(c_l) \tag{3}$$

where $\mathcal{C}(w, b)$ is the task-related loss term such as a cross-entropy loss in the classification task. The second sum-term corresponds to a regularization term and the parameter λ balance the relative importance of the two terms. The regularization term $g(\cdot)$ is a penalty on the clamping parameter, which can be chosen as L1 or L2-norm.

In our experiment, we find L2 regularization term $g(c_l) = c_l^2$ is better than L1-norm $g(c_l) = |c_l|$. It is reasonable because L2 regularization term tends to decrease the magnitude while L1-norm is widely used to help sparsity. We observe that c_l will decrease gradually and converges to values much smaller than the initial value in the training process, thus limiting the dynamic range of activations and minimizing quantization errors.

A somewhat surprising fact is that although the value of activations is clamped to c_l, we empirically observe that CReLU is more effective and robust than other activation functions with and without quantization. The experimental details can be found in Section 4.2.

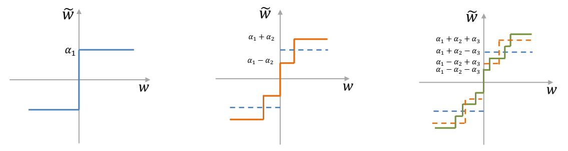
Activations Quantization In this part, we detail our element-wise quantization approach to getting low bitwidth activations. We first uniformly quantize a real number input $\mathbf{x} \in [0, c_l]$ to a k-bitwidth output $\widetilde{\mathbf{x}} \in [0, c_l]$. The linear quantizer is defined as below:

$$\widetilde{\mathbf{x}} = \text{round}(\mathbf{x} \cdot \frac{2^k - 1}{c_l}) \cdot \frac{c_l}{2^k - 1} \tag{4}$$

Note that in the case of $c_l = 1$, it is the quantizer claimed by (Zhou et al. 2016), in which they need to limit the activation of previous layers to [0,1] by passing through a unknown bounded function h and then use the linear quantizer. In addition, sometimes we need the activations to be some special type. For instance, if all the operands are power of two, the intrinsic benefit is that it can employ cheap bit shift operations to replace costly MAC operations, which is convenient to be deployed in FPGA devices. Therefore, we provide another logarithmic quantizer to discrete the activations to be either powers of two or zero, which is defined as follows:

$$\widetilde{\mathbf{x}} = 2^{\,\text{clamp}\left(\lfloor \log_2(\mathbf{x}) \rfloor,\ n - 2^k,\ n\right)}, \quad n = \log_2(c_l) \tag{5}$$

where $\lfloor x \rfloor$ is the greatest integer less than or equal to x, and clamp(x, a, b) := max(a,min(b,x)). The quantization procedure can be described as: we take logs to the base 2, round down to the nearest integer, then restrict the value to the range $[2^{n-2^k}, 2^n]$ according to the bit-width k and the maximum value c_l and finally invert the log operation by raising 2 to that power. In short, the core idea is to simply find the closest neighbor, either power of two or zero, of the input $\mathbf{x}$. In addition, $\lfloor \log_2(\mathbf{x}) \rfloor$ is just the exponent component for a floating-point number and multiplications can be transformed into inexpensive bitwise shift operations (i.e., $w \times 2^x = w << x$). Therefore, this quantizer is not only easy to implement in practical programming and dedicated hardware but also beneficial to the convolutional operation in comparison to linear quantization.



(a) Level1 (b) Level2 (c) Level3

Figure 3: Illustration of multi-level binarization

For the backward pass, we directly use STE to estimate the gradient: $\frac{\partial \ell}{\partial \mathbf{x}} = \frac{\partial \ell}{\partial \widetilde{\mathbf{x}}}$.

Binary Weights

Forward Approximation To approximate the weights precisely, an effective strategy to binarize the weights in the forward pass, which we adopt in this work, has been proposed by (Guo et al. 2017). They use multiple binarizations for weights by tackling the following problem:

$$\min_{\{\alpha_i, \mathbf{b}_i\}_{i=0}^{m-1}} \left\| \mathbf{w} - \sum_{i=0}^{m-1} \alpha_i \mathbf{b}_i \right\|^2, \text{with } \mathbf{b}_i \in \{-1, +1\}^n. \tag{6}$$

However, directly minimizing above problem seems NP-hard and in particular, they proposes a heuristic (or greedy) algorithm, in which $\mathbf{b_i}$ and α_i are selected to be the current optimum by sequentially minimizing the residue $\mathbf{r}_i$. That is,

$$\min_{\alpha_i, \mathbf{b}_i} \|\mathbf{r}_i - \alpha_{i+1}\mathbf{b}_{i+1}\|^2, \text{with } \mathbf{r_i} = \mathbf{w} - \sum_{k=0}^{i-1} \alpha_k \mathbf{b}_k. \tag{7}$$

Through derivative calculations, the optimal solution is given as

$$\alpha_i^* = \frac{1}{n}\|\mathbf{r}_i\|_{\ell 1} \quad \text{and} \quad \mathbf{b}_i^* = \text{sign}(\mathbf{r}_i). \tag{8}$$

where $\mathbf{r}_i$ indicates the approximation residue after combining all the previously generated tensor(s). More details can be found in (Guo et al. 2017).

Backward Approximation While it is easy to binarize the weights through sign function in the forward pass, we have to maintain full-precision values before the discretization since binary weights cannot be used to accumulate the high precision gradients. In backward pass, we need the gradients from binarized weights flow to the original real-value weights.

However, the quantization function used to binarize weights $\mathbf{w}$ to $\widetilde{\mathbf{w}}$ contains the sign function, which is incompatible with backpropagation, since its derivative is zero for all non-zero inputs. Generally, incorporating quantization steps in feed-forward prevents direct training of neural network with the BP algorithm, as mathematically any quantization function will have zero derivatives. To remedy this, (Courbariaux, Bengio, and David 2015) propose to use the "straight-through estimator (STE)" to assign non-zero gradients for quantization functions. Assume ℓ as the cost function, (Rastegari et al. 2016) adopts the following formula to calculate the gradient with respect to $\mathbf{w}$. That is,

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{w}_i} &= \frac{\partial \ell}{\partial \widetilde{\mathbf{w}}_j} \frac{\partial \widetilde{\mathbf{w}}_j}{\partial \mathbf{w}_i} \\ &= \frac{\partial \ell}{\widetilde{\mathbf{w}}_i} (\frac{1}{n} + \alpha \mathbf{w}_i \mathbf{1}_{|\mathbf{w}_i| \leq 1}) \end{aligned} \tag{9}$$

Equation (9) does allow the gradient to flow in backward pass during training. Nevertheless, we find it is still not precise enough since the deduction process ignores the impact of other components in the weight filter when introducing the scale factor α. To eliminate the gradient mismatch, we derive a more effective gradient with respect to real-valued weights in the backpropagation.

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{w}_i} &= \sum_{j=1}^{n} (\frac{\partial \ell}{\partial \widetilde{\mathbf{w}}_j} \frac{\partial \widetilde{\mathbf{w}}_j}{\partial \mathbf{w}_i}) \\ &= \frac{\mathrm{s}(\mathbf{w}_i)}{n} \sum_{j=1}^{n} \left(\frac{\partial \ell}{\partial \widetilde{\mathbf{w}}_j} \mathrm{s}(\mathbf{w}_j) \right) + \frac{\partial \ell}{\partial \widetilde{\mathbf{w}}_i} \mathbf{1}_{|\mathbf{w}_i| \leq 1} \alpha \end{aligned} \tag{10}$$

where $s(\mathbf{w}_i)$ denotes the weight's sign. The detailed deduction is summarized in the supplementary material. In practice, we show that training neural network with Equation (10) can achieve marginally better accuracy.

We call this method multi-level binarization, where m is the number of level that weights are binarized and Fig. 3 show its forward approximation while the backward pass is omitted in the graph.

In summary, our overall training procedure is summarized in Algorithm 1.

Discussion:

Some recent works also proposed similar idea of multi-level binarization. (Guo et al. 2017), (Li et al. 2017) apply this strategy to the activations instead of weights, but our choice is based on such intuition that weights are saved and fixed after the training, we approximate those real-valued weights with binary values as much as possible to minimize accuracy degradation. Activations, by contrast, change all the time and no final optimal values to be stored, thus making them less suitable to be quantized in this way. In comparison with (Guo et al. 2017), we formulate more precise gradient of the loss with respect to real-valued weights to alleviate the gradient mismatch during training.

Algorithm 1: Training a L-layers DNN with binary weights and low-bitwidth activations. Note that, convolutional operation $*$ becomes the fixed-point accumulations or xnor bitwise operation.

```
Input: Minibatch of inputs & targets ($\mathbf{x}$ , $\mathbf{y}^*$); current
       weights $\mathbf{w}^t$; CReLU parameters $c_l^t$; multi-level
       $m$; task-related loss term $\mathcal{C}$, regulation term
       coefficient $\lambda$; learning rate $\eta^t$.
Output: Updated $\mathbf{w}^{t+1}$, $c_l^{t+1}$ and $\eta^{t+1}$.
Forward propagation:
for $l := 1$ to $L$ do
    Compute multi-level binary weights:
    for $i := 0$ to $m-1$ do
        Calculate $\mathbf{b}_i$ and $a_i$ using Eq. (8)
        Update $\mathbf{r}_i$ using Eq. (7)
    end
    $\widetilde{\mathbf{w}}_l = \sum_{i=0}^{m-1} \alpha_i \mathbf{b}_i$
    $\mathbf{y}_l = \widetilde{\mathbf{w}}_l * \widetilde{\mathbf{x}}_{l-1} = \sum_{i=0}^{m-1} (\alpha_i (\mathbf{b}_i * \widetilde{\mathbf{x}}_{l-1}))$
    $\mathbf{x}_l \leftarrow$ CReLU($\mathbf{y}_l$, $c_l$)
    Apply activations quantization method:
    $\mathbf{x}_l \leftarrow \widetilde{\mathbf{x}}_l$ using Eq. (4) or Eq. (5)
end
Compute total loss $\ell$ with $\mathbf{y}_L$ and target $\mathbf{y}^*$:
$\ell \leftarrow \mathcal{C}(\mathbf{y}_L, \mathbf{y}^*) + \lambda \sum_{l=1}^{L} c_l^2$
Backward propagation:
Compute gardients of the final layer $\frac{\partial \ell}{\partial \widetilde{\mathbf{x}}_L}$
for $l := L$ to $1$ do
    $(\frac{\partial \ell}{\partial \mathbf{c}_l}, \frac{\partial \ell}{\partial \mathbf{y}_l}) \leftarrow$ BackCReLU($\mathbf{y}_l$, $c_l$) using Eq. (2)
    $\frac{\partial \ell}{\partial \widetilde{\mathbf{w}}_l} = \frac{\partial \ell}{\partial \mathbf{y}_l} \widetilde{\mathbf{x}}_{l-1}^{\mathrm{T}}$
    $\frac{\partial \ell}{\partial \widetilde{\mathbf{x}}_{l-1}} \leftarrow \widetilde{\mathbf{w}}_l^{\mathrm{T}} \frac{\partial \ell}{\partial \mathbf{y}_l}$
    Flow the gradients to original real-value weights:
    $\frac{\partial \ell}{\partial \mathbf{w}_l} \leftarrow \frac{\partial \ell}{\partial \widetilde{\mathbf{w}}_l}$ using Eq. (10)
end
Update the parameters:
$\mathbf{w}^{t+1} \leftarrow$ UpdateWeights($\mathbf{w}^t$, $\frac{\partial \ell}{\partial \mathbf{w}_l}$)
$c_l^{t+1} \leftarrow$ UpdateCReLU($c_l^t$, $\frac{\partial \ell}{\partial \mathbf{c}_l}$, $\lambda$)
$\eta^{t+1} \leftarrow$ UpdateLearningRate($\eta^t$, t)
```

Experiments

In this section, we empirically validate the superiority of the proposed quantization scheme for CNNs with binary wights and low bitwidth activations on the CIFAR-10 and ImageNet (ILSVRC2012) datasets. We report the results in terms of the top-1 and top-5 errors. We test on several representative CNNs including: VGGNet (Simonyan and Zisserman 2014), AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and ResNet (He et al. 2016). For all experiments, we use the SGD solver with the momentum of 0.9 and set the weight-decay to 0 since we no longer encourage the weights to be close to 0. The initial learning rate is set to 0.1 and is divided by 10 once the validation error exceeds the best record. We remove the standard dropout layer in AlexNet due to the quantization itself can be considered as a kind of regularization. Our implementation is based on Pytorch.

Datasets

CIFAR. The CIFAR-10 dataset (Krizhevsky and Hinton 2009) is image classification benchmark dataset containing 32×32RGB images in a training set of 50000 and a test set of 10000. It consists of 10 categories, like airplanes,trucks, birds, frogs, cats, horses, deer, dogs, ships and automobiles. Each category contains 5000 training images and 1000 testing images. For each image in the CIFAR-10 dataset, we

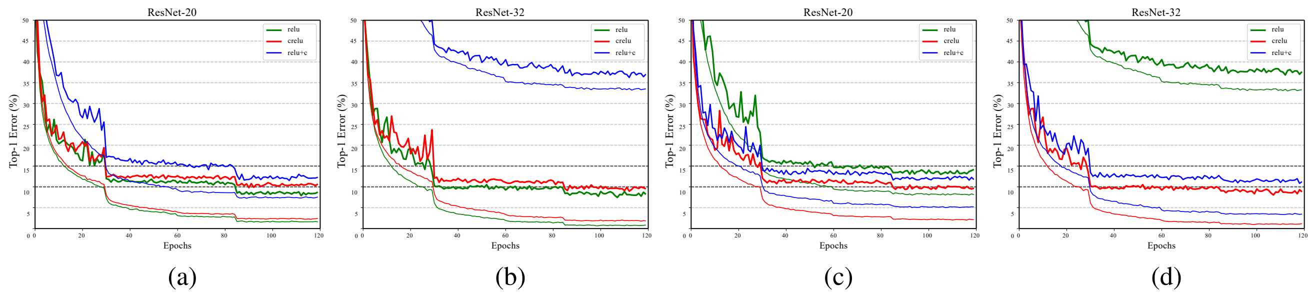


Figure 4: The error curves of training (thin) and test (thick) for relu, crelu and relu with fix clamping values (denoted as relu+c) activation functions on CIFAR-10 dataset. (a)(b) and (c)(d) are trained without and with quantization, respectively.

adopt the standard data augmentation scheme: we pad 2 pixels in each side of images and randomly crop 32×32 size from padded images during training. Original 32×32 images are used to test the networks during inference.

ImageNet. ImageNet is one of the most challenging image classification benchmarks, which has about 1.2million training images and 50 thousand validation images, and these images cover 1000 object classes. In all ImagNet experiments, we resize all the images to 256×256. The images are then randomly clipped to 224×224 patches with mean subtraction and randomly flipping. No other data augmentation tricks are used in the training.

Results for Binary Weights

We firstly compare the performance of our binarization method with that of other binary-weight models: Binary-Connect (Courbariaux, Bengio, and David 2015), BWN-net (Rastegari et al. 2016) and a recent work Sketch (Guo et al. 2017). In this section, all models only binarize the weights and keep the activations to be of full-precision. We start training with pre-trained model rather than from scratch. We use a variant of VGG-Net with only 3 convolutional layers rather than 5. Table 1 shows that the accuracy of our binary model is improved as the number of levels is increased and even outperforms its full-precision counterpart.

We also report our results with ResNet-18 in Table 2. It can be observed that our results have better performance than a recent state-of-the-art work Sketch by a little margin though we adopt the same weights binarization method as them in the forward pass except the backward pass. We conjecture that it is possible (and even likely) that our refined backpropagation maybe better suited for gradient to flow. The results of BWN-net and Ours (m = 1) in Table 1 validate this point again.

Although, the better results can be achieved by increasing the number of level, it also increases some extra computational overhead during training and inference, which is a trade-off between accuracy and complexity. Therefore, we adopt two-level (m = 2) binarization to keep the balance in the following experiments.

Comparison with the state-of-the-art

We then integrate the proposed weights and activations quantization into the training of neural networks. We present

Network		Top1 error	Top5 error	Top1 gap
VGG-variant ref		7.24%	0.90%	-
Binary-Connect		8.12%	1.23%	0.88%
BWN-net		7.84%	1.09%	0.60%
Ours	m = 1	7.60%	0.78%	0.36%
	m = 2	6.86%	0.73%	**-0.38%**
	m = 3	6.42%	0.67%	**-0.82%**

Table 1: This table compares top1 and top5 error of binary weight models on Cifar-10. m is the number of level that weights are binarized.

Network	Top1 error	Top5 error	Top1 gap
ResNet-18 ref	31.2%	11.0%	-
BWN-net	39.2%	17.0%	8.0%
Sketch (dir.)	32.7%	11.8%	1.5%
Ours (m = 2)	32.5%	11.4%	1.3%

Table 2: This table compares top1 and top5 error of binary weight models on ImageNet. The test error of BWN-net & Sketch are directly cited from (Guo et al. 2017).

a comparison between our quantized model with other state-of-the-art models, including XNOR-Net, DoReFa-Net (Zhou et al. 2016), HWGQ-Net (Cai et al. 2017) on the ImageNet classification task. Note that we keep the gradients being full-precision in DoReFa-Net and follow the practice that leaving the first and last layer in full precision during training since quantizing those layers have been reported to significantly degrade accuracy. The results are shown in Table 3. It can be seen that with binary weights and 2-bit activations, the proposed approach achieves lower accuracy degradation that demonstrates the superior performance relative to previous state-of-the-art works.

Ablation study

In this subsection, we try to empirically analyze the impact of CReLU activation function and bitwidth of activations, which are tightly correlated with our proposed quantization approach.

The effect of activation function While we have provided theoretical analysis on why our CReLU works, it is neces-

Network		Bit-width	Top1 error	Top5 error	Top1 gap	Top5 gap
AlexNet	reference	32 + 32	42.8%	19.7%	-	-
	XNOR-Net	1 + 1	55.8%	30.8%	13.0%	11.1%
	DoReFa-Net	2 + 2	53.6%	23.2%	10.8%	3.5%
	HWGQ-Net	1 + 2	47.3%	23.7%	4.5%	4.0%
	Ours	1 + 2	**46.2%**	**22.4%**	**3.4%**	**2.7%**
ResNet-18	reference	32 + 32	31.2%	12.4%	-	-
	XNOR-Net	1 + 1	46.8%	28.8%	15.6%	16.4%
	DoReFa-Net	2 + 2	39.1%	17.6%	7.9%	5.2%
	HWGQ-Net	1 + 2	40.4%	17.8%	9.2%	5.4%
	Ours	1 + 2	**38.4%**	**16.6%**	**7.2%**	**4.2%**

Table 3: Comparison with the state-of-the-art models with binary weights and low-bitwidth activations using AlexNet and ResNet-18. The bit-width before and after "+" is for weights and activations respectively. Note that all the comparing results are reported in terms of accuracy in the original papers, which are converted to corresponding error herein.

sary to empirically demonstrate its effectiveness relative to other activation functions. To this end, we carry out a comparative study with ReLU and ReLU with fixed clamping parameters. Both of them are widely used in previous methods of quantizing activations for CNNs and achieve state-of-the-art results. Note that, for a fair comparison, the experiment setting is identical differing only the activation function.

The results are summarized in Fig. 4, in which we present the training and test errors for the ResNet-20 and ResNet-32 with appropriate changes for the 10-class Cifar-10 dataset. Fig. 4(ab) show that CReLU achieves comparable performance with ReLU without quantization, which is also verified by our theoretical analysis. In Fig. 4(cd), when quantization is applied during training, it can be observed that the significant accuracy degradation when ReLU is used as activation function.

In short, this comparative study illustrates the effectiveness and robustness of CReLU relative to other activation functions with and without quantization.

The effect of activation bitwidth This set of experiment is performed to explore the influence of different combinations of bitwidth for the final accuracy. We adopt linear activation quantizer with ResNet-50 on ImageNet for analysis. The results are provided in Table 4. Generally, with binary weights and increasing bitwidth of activations, we can find the accuracy degradation is steadily reduced. For example, with binary weights and 5-bit activations, the top-1 accuracy drop is only 0.3% and the top-5 accuracy even increases 0.2% relative to its full-precision counterpart.

Bit-width	Top1 error	Top5 error	Top1 / Top5 gap
32 + 32	23.1%	6.9%	- / -
1 + 2	27.8%	9.5%	4.7% / 2.6%
1 + 3	26.0%	8.8%	2.9% / 1.9%
1 + 4	24.7%	7.4%	1.6% / 0.5%
1 + 5	23.4%	6.7%	0.3% / -0.2%

Table 4: Validation accuracy for ResNet50 on ImageNet with different bitwidths of activations.

Computation Complexity Analysis

In this section, we measure computational complexity by FLOPs, the number of floating-point operations, which is widely used in previous papers. For a standard convolution, we have $c_{out}c_{in}whw_{out}h_{out}$ floating point operations (the multiplication and addition are considered as a single cycle operations herein). On the modern CPUs, we can perform 64 binary operations in one clock of CPU (Rastegari et al. 2016), which means one floating-point operation roughly equals to 64 binary operations within one clock cycle. When we adopt m-level binarization for weights and 1-bitwidth activations, we have $mc_{out}c_{in}whw_{out}h_{out}$ binary operations. In addition, we need $mw_{out}h_{out}$ scaling floating-point operations. Therefore, the speedup ratio can be computed as:

$$s = \frac{c_{out}c_{in}whw_{out}h_{out}}{\frac{m}{64}c_{out}c_{in}whw_{out}h_{out} + mw_{out}h_{out}} \tag{11}$$

Considering ResNet-18 for example, the full precision baseline has 1.81×10^9 FLOPs and our corresponding low precision network has only 1.67×10^8 FLOPs, thus we gain $10.86\times$ theoretical speedup. Note that the speedup ratio will relatively reduce for some small non-binary bitwidth of activations (e.g. k = 2, 4), but fixed-point addition or bit shift operation are still enough efficient to reduce inference time for specialized hardwares.

Conclusions

In this paper, we firstly design a new activation function, namely CReLU, to keep the balance between clamping error and quantization error. We further complement this design with experimental results and theoretical analysis to verify its tremendous advantage over conventional ReLU function. Moreover, we propose multi-level binarization method and linear or logarithmic quantizer for simultaneously binarizing the weights and quantizing the activations to low bitwidth. Extensive experimental results demonstrate that the proposed approach show superiority over the state-of-the-art methods.

In future, we will explore the possibility of generating the work to other computer vision tasks such as object detection and semantic segmentation.

Acknowledgements

The work was partly supported by State Key Research and Development Program (2016YFB1001003), National Science Foundation of China (U1611461, 61502301, 61521062, 61527804), China's Thousand Youth Talents Plan, STCSM (18DZ1112300, 17511105401, 18DZ2270700).

References

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. *arXiv preprint arXiv:1702.00953*.

Chen, W.; Wilson, J.; Tyree, S.; Weinberger, K.; and Chen, Y. 2015. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, 2285–2294.

Chollet, F. 2016. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*.

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

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

Girshick, R. 2015. Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Guo, Y.; Yao, A.; Zhao, H.; and Chen, Y. 2017. Network sketching: Exploiting binary structure in deep cnns. *arXiv preprint arXiv:1706.02021*.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jouppi, N. P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12. ACM.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Li, Z.; Ni, B.; Zhang, W.; Yang, X.; and Gao, W. 2017. Performance guaranteed network acceleration via high-order residual quantization. *arXiv preprint arXiv:1708.08687*.

Li, F.; Zhang, B.; and Liu, B. 2016. Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

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

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

Zhang, T.; Qi, G.-J.; Xiao, B.; and Wang, J. 2017a. Interleaved group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4373–4382.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2017b. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*.

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