LEARNING BILATERAL CLIPPING PARAMETRIC ACTIVATION FUNCTION FOR LOW-BIT NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Rectified Linear Unit (ReLU) is a widely used activation function in deep neural networks, and several works are devoted to designing its variants to improve performance. However, the output is unbounded for most of such functions, which brings severe accuracy degeneration when the full-precision model is quantized. To tackle the problem of unboundedness, Choi et al. (2019) introduce an activation clipping parameter for the standard ReLU. In this paper, we propose a Bilateral Clipping Parametric Rectified Linear Unit (BCPReLU) as a generalized version of ReLU and some variants of ReLU. Specifically, the trainable slopes and thresholds for both positive and negative input are introduced in BCPReLU. We theoretically prove that BCPReLU has almost the same expressive ability as the corresponding unbounded one, and establish its convergence in low-bit quantization training. Numerical experiments on a range of popular models and datasets verify its effectiveness, which outperforms the state-of-the-art methods.

1 INTRODUCTION

Deep Neural Networks (DNNs) have achieved great success in various computer vision tasks, such as image classification (Simonyan & Zisserman, 2014; He et al., 2016), object detection (Ren et al., 2015; Cai et al., 2016), and image segmentation (Yu et al., 2018; Takikawa et al., 2019) etc. However, the high computational cost and large memory storage make DNNs difficult to be deployed on resource-constrained devices. Thus, the compression methods by quantization have gained attention, which map the full-precision floating-point values to low-bit fixed-point ones.

The non-saturated activation function plays an important role in both the full-precision and quantized network, whose non-saturated feature settles the problem of exploding/vanishing gradient, thereby accelerating the speed of model convergence. In all of these activation functions, the most popular one is ReLU (Glorot et al., 2011), which outputs zero for the negative input, and retains the same value for the positive input. To alleviate the problem of zero gradients in ReLU, leaky ReLU (LReLU) (Maas et al., 2013) assigns a small and fixed slope for the negative input. PReLU (He et al., 2015) introduces a trainable slope for the negative input rather than fixed one in LReLU. The PReLU is a key factor for machine to surpass human-level classification performance on the ImageNet 2012 dataset for the first time. To reduce overfitting, Xu et al. (2015) propose a randomized version of LReLU, where the slope for the negative input during training is randomized in a given range, and then fixed in the testing. Different from the above static activation functions, SE (Hu et al., 2018) and DY-ReLU (Chen et al., 2020) adopt a trainable slope function of input. The former is a special case of the latter, and the latter achieves solid improvement with only an extra increase (5%) of computational cost. We illustrate the above functions in Figure 1 for comparisons.

Although different types of rectified activation functions have achieved huge success for float-point network, the output of these activation functions is unbounded. This makes its quantization difficult because of the high dynamic range. So far, the problem has been partially addressed by some different clipping activation functions. Jacob et al. (2018) show that the quantization error can be reduced by setting a fixed upper-bound on the output to limit the range of output. PACT (Choi et al., 2019) places a learnable upper-bound on the output of standard ReLU. HWGQ (Cai et al., 2017) exploits the statistics of activation and proposes a variant of ReLU which constrains the unbounded value. As far as we know, these methods mainly focus on the standard ReLU, other ReLU variants

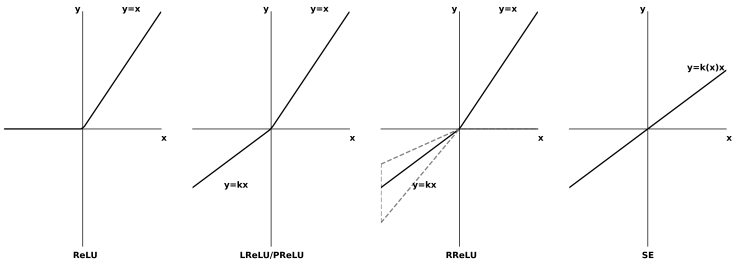

Figure 1: ReLU, LReLU/PReLU, RReLU, and SE. For PReLU, k is learned and for LReLU k is fixed. For RReLU, k is randomized during training in a given range. For SE, the slope is decided by a trainable dynamic function.

have not been studied for quantization.

In this paper, we propose a Bilateral Clipping Parametric Rectified Linear Unit(BCPReLU) as a generalization of ReLU and the existing variants. The learnable two sided thresholds for activation value and trainable non-zero slopes for activation input are introduced in BCPReLU. we prove that the expressive ability of BCPReLU is almost the same as corresponding unbounded one, and establish the convergence property of the quantization error for BCPReLU. Experiments on a set of popular models and datasets demonstrate the effectiveness of BCPReLU in both full-precision and quantized network. Our contributions are summarized below:

- We propose a new clipping activation function as a generalized version of ReLU and its variants. The trainable slopes and thresholds for both positive and negative input are introduced in BCPReLU.
- We theoretically prove that BCPReLU has almost the same expressive ability as the corresponding unbounded function in full-precision network, and establish the convergence of BCPReLU in quantized network.
- Extensive experiments on CIFAR-10 and SVHN datasets demonstrate the effectiveness of BCPReLU.

The rest of the paper is organized as follows: Section 2 provides a summary of prior works on quantization. Section 3 proposes a novel clipping activation function, of which the expressive ability in full-precision network is analyzed, and the convergence in quantized network is established. In Section 4, we demonstrate the effectiveness of our quantization schemes by experiments.

2 Related Work

Currently, quantization has become an efficient way to compress model with limited accuracy degeneration. Approaches for quantizing full-precision network into low-bit one can be roughly divided into two categories: weight quantization and activation quantization, the former reduces computational cost, and the latter saves memory storage. Early quantization works (Li et al., 2016; Zhu et al., 2016; Rastegari et al., 2016) are mainly concerned with weight quantization, which quantize weight to 1-bit (binary) or 2-bits (ternary). However, the quality of activation quantization is also a key factor affecting the low-bit network performance. Thus, activation quantization has gained much attention. Recently, several works design new activation quantization schemes according to the activation distribution or the range of activation output. Sung et al. (2015) examine post-training quantization for DNNs. They adopt a clip threshold that minimizes the L2-norm of the quantization error. In the work of Jacob et al. (2018), quantization is performed by applying point-wise the quantization function during training, in which the quantization factor is parameterized by quantization level and clamping range. Besides, ACIQ (Banner et al., 2019) proposes a 4-bit post training quantization approach, in which the fixed-point values are obtained by rounding the full-precision ones to the midpoint of every quantization region. These works show that a bounded range is beneficial to improve the quantization performance. However, the fixed bounded range on the output is suboptimal due to layers/channels difference. PACT (Choi et al., 2019) places learnable threshold α on

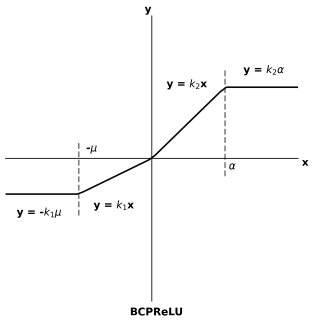

Figure 2: BCPReLU: the k_1 and k_2 control the slopes of the negative input and the positive input, respectively. The $-k_1\mu$ and $k_2\alpha$ control the lower bound and upper bound of output, respectively. The k_1, k_2, μ, α are adaptively learnedn in training, and then fixed in the testing.

the output of ReLU to find the optimal range of output, and then the values in the optimized range are linearly quantized to M bits.

3 BCPReLU and Quantization Analysis

In this section, we first propose a new clipping activation function, then prove that the proposed activation function has almost the same representation ability as the corresponding unbounded one in the full-precision network. According to the reasonable assumption of activation distribution, we finally establish the convergence of the quantization error.

3.1 BCPReLU: Bilateral Clipping Parametric Rectified Linear Unit

As stated in the previous section, we propose a new clipping trainable activation function (BCPReLU):

$$y = f(x; \mu, k_1, \alpha, k_2) = \begin{cases} -k_1\mu, & x \in (-\infty, -\mu) \\ k_1 x, & x \in [-\mu, 0) \\ k_2 x, & x \in [0, \alpha) \\ k_2\alpha, & x \in [\alpha, +\infty) \end{cases} \tag{1}$$

Here x is the input of the nonlinear activation y, the parameters k_1 and k_2 are trainable and control the slopes of the negative input and the positive input, respectively. The parameters $-\mu$ and α are the clipping values of the negative input and the positive input, respectively. When $k_1 = 0$ and $k_2 = 1$, the clipping activation function becomes PACT (the trainable clipping ReLU); when k_1 is a small and fixed value and $k_2 = 1$, it becomes a clipping form of LReLU; and when k_1 is a trainable variable and $k_2 = 1$, it becomes a clipping form of PReLU.

As shown in Figure 2, when $-\mu \to -\infty$ and $\alpha \to +\infty$, the output of BCPReLU is unbounded, which incurs large quantization error. To limit the range of output, it is necessary to place reasonable clipping values. The clipping values $-k_1\mu$ and $k_2\alpha$ in BCPReLU limit the range of output, k_1 and μ also enhance the expressive power in negative input to avoid the case of dead neurons.

3.2 BCPReLU has almost the same as expressive ability as the corresponding unbounded one

In this subsection, we consider the effect of clipping values, and compare BCPReLU with the corresponding unbounded one in full-precision network. Following Choi et al. (2019), we summary as follows:

Theorem 1 *Assume that x is a activation input, $y = f(x; k_1, \mu, k_2, \alpha)$ represents corresponding output. The network with BCPReLU can be trained to find the same output as the corresponding unbounded function and converge faster than the corresponding unbounded function.*

proof 1 *Assume that y^* is the corresponding label of x, the cost function is defined as following by the mean-square-error (MSE) :*

$$L(y) = 0.5 \cdot (y - y^*)^2.$$

We define the unbounded function corresponding to BCPReLU as g:

$$g = g(x; k_1, k_2) = \begin{cases} k_1 x, & x \in (-\infty, 0) \\ k_2 x, & x \in [0, +\infty) \end{cases} \tag{2}$$

If $x \in [-\mu, \alpha]$, the network with BCPReLU (Eq. 1) behaves approximately the same as the network with the unbounded function g (Eq. 2).

If $x < -\mu$, then $y = -k_1\mu, g = k_1 x$, k_2 and α are not updated due to

$$\frac{\partial y}{\partial k_2} = \frac{\partial y}{\partial \alpha} = 0.$$

Updating k_1 and μ :

$$k_1^{new} = k_1 - \eta \frac{\partial L}{\partial k_1} = k_1 - \eta \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial k_1} = k_1 + \eta\mu \frac{\partial L}{\partial y},$$

$$\mu^{new} = \mu - \eta \frac{\partial L}{\partial \mu} = \mu - \eta \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \mu} = \mu + \eta k_1 \frac{\partial L}{\partial y}.$$

The update of k_1 and u depend on $\frac{\partial L}{\partial y}$, so we need consider the different cases of $\frac{\partial L}{\partial y} = y - y^*$:
Case 1: *If $y^* < k_1 x$, then $y^* < y$, $\frac{\partial L}{\partial y} > 0$, k_1 increase, μ increase, thus $y = -k_1\mu$ dcreases until $-\mu < x$, i.e., $y = g$. The network with BCPReLU (Eq. 1) behaves the same as the network with the unbounded function g (Eq. 2) in this case.*

Case 2: *If $k_1 x < y^* < y$, $\frac{\partial L}{\partial y} > 0$, k_1 increase, μ increase, thus $y = -k_1\mu$ decrease and converge to y^*.*

Case 3: *If $y < y^*$, $\frac{\partial L}{\partial y} < 0$, k_1 decrease, μ decrease, thus $y = -k_1\mu$ increase and converge to y^*.*

*Note that in **case2** and **case3**, the output of BCPReLU converges to the same target y^* faster than the corresponding unbounded function g.*

If $x > \alpha$, then $y = k_2\alpha$, k_1 and μ are not updated. The analysis way is similar as that when $x < -\mu$.

This completes the proof of Theorem 1. Q.E.D.

From Theorem 1, we know that BCPReLU has almost the same expressive power with the corresponding unbounded function.

3.3 Quantization method and the convergence of quantization error

The BCPReLU limits the range of activation to $[-k_1\mu, k_2\alpha]$. According to the quantization method from (Choi et al., 2018), the bounded range of output is linearly quantized to M bits as follows

$$y_q = round(y \cdot \frac{2^M - 1}{k_1\mu + k_2\alpha}) \cdot \frac{k_1\mu + k_2\alpha}{2^M - 1}.$$

During the training stage, k_1, μ, k_2, α, are trainable variables. The gradient with respect to the involved parameters $\frac{\partial y_q}{\partial \mu}$, $\frac{\partial y_q}{\partial k_1}$, $\frac{\partial y_q}{\partial \alpha}$ and $\frac{\partial y_q}{\partial k_2}$ can be computed by the Straight-Through Estimator (STE) (Bengio et al., 2013). Thus,

$$\frac{\partial y_q}{\partial \mu} = \frac{\partial y_q}{\partial y} \cdot \frac{\partial y}{\partial \mu} = \begin{cases} -k_1, & x \in (-\infty, -\mu) \\ 0, & x \in [-\mu, +\infty), \end{cases}$$

$$\frac{\partial y_q}{\partial k_1} = \frac{\partial y_q}{\partial y} \cdot \frac{\partial y}{\partial k_1} = \begin{cases} -\mu, & x \in (-\infty, -\mu) \\ x, & x \in [-\mu, 0) \\ 0, & x \in [0, +\infty), \end{cases}$$

$$\frac{\partial y_q}{\partial k_2} = \frac{\partial y_q}{\partial y} \cdot \frac{\partial y}{\partial k_2} = \begin{cases} 0, & x \in (-\infty, 0) \\ x, & x \in [0, \alpha) \\ \alpha, & x \in [\alpha, +\infty), \end{cases}$$

$$\frac{\partial y_q}{\partial \alpha} = \frac{\partial y_q}{\partial y} \cdot \frac{\partial y}{\partial \alpha} = \begin{cases} 0, & x \in (-\infty, \alpha) \\ k_2, & x \in [\alpha, +\infty). \end{cases}$$

Next, we consider the quantization error of BCPRLU. Under a reasonable activation distribution assumption, the convergence of quantization error is established as follows:

Theorem 2 *Assume that activation input* x *is a random variable, with a probability density function* $f(x) = Laplace(0, b)$*, the quantization error of BCPReLU converges to zero when bit-width* $M \rightarrow \infty$.

A proof of the Theorem 2 will be given in Appendix A. This theorem shows that our proposed activation function using common quantization method satisfys the property of quantization error i.e. the quantization error converges to zero when the bit-width M approaches infinity ($\rightarrow \infty$).

4 EXPERIMENTS

To demonstrate the effectiveness of BCPReLU, we evaluate it on several well-known models: ResNet20/32 (He et al., 2016) for the CIFAR10 and SVHN datasets. CIFAR-10 dataset contains 10 different classes images, and each image is an RGB image in size 32×32. There are 50,000 training images and 10,000 test images. SVHN consists of a training set of size 76K examples and a test set of size 26K, where each image is 32×32 color images. In all experiments on SVHN, we follow the same procedure used for the CIFAR-10 experiments.

4.1 THE EQUIVALENT FORM OF BCPRELU

With loss of generality, we consider the case where convolutions and BCPReLU are fused to derive the equivalent form of BCPReLU. Considering a single-neuron network with BCPReLU, where (b, y^*) is a sample of training data and w is weight, then $x = wb$,

$$y = \begin{cases} -k_1\mu, & wb \in (-\infty, -\mu) \\ k_1 wb, & wb \in [-\mu, 0) \\ k_2 wb, & wb \in [0, +\alpha) \\ k_2\alpha, & wb \in [\alpha, +\infty) \end{cases}$$

The $k_2 w$, $k_2\mu$ and $k_2\alpha$ are denoted by w^*, μ^* and α^*, respectively.

$$y = \begin{cases} -\frac{k_1}{k2}\mu^*, & w^*b \in (-\infty, -\mu^*) \\ \frac{k_1}{k2} w^* b, & w^*b \in [-\mu^*, 0) \\ w^*b, & w^*b \in [0, +\alpha^*) \\ \alpha^*, & w^*b \in [\alpha^*, +\infty) \end{cases}$$

So the slope k_2 of the positive input can be integrated into the training of k_1, μ, α and weight. The proposed clipping activation function is equivalent to the bounded from of PReLU (BCPReLU) as follow

$$y = f(x; \mu, k, \alpha) = \begin{cases} -k\mu, & x \in (-\infty, -\mu) \\ kx, & x \in [-\mu, 0) \\ x, & x \in [0, \alpha) \\ \alpha, & x \in [\alpha, +\infty) \end{cases} \tag{3}$$

Here k is a coefficient controlling the slope of the negative input, $-k\mu$ and α control the lower bound and upper bound of y, respectively. Note that we use the equivalent function from Eq.3 in the following experiments.

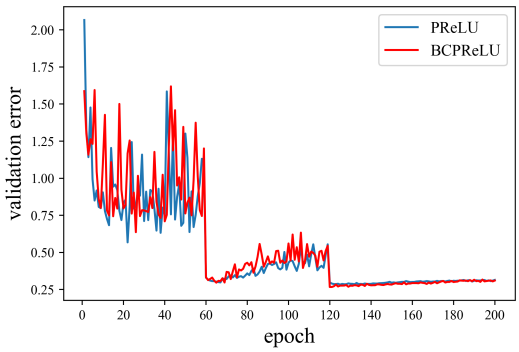

Figure 3: The validation error of PReLU and BCPReLU on CIFAR10-ResNet20.

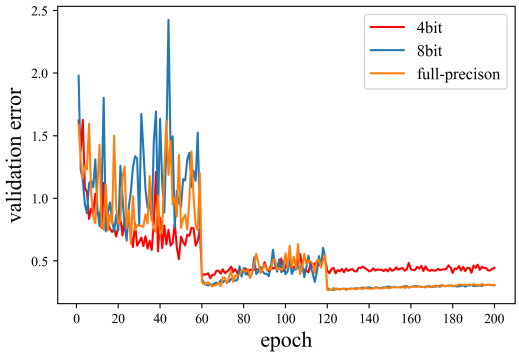

Figure 4: The validation error of different bit-width BCPReLU on CIFAR10-ResNet20.

4.2 Comparison of PReLU and BCPReLU in full precision network

For BCPReLU experiments, we only replace PReLU with BCPReLU but keep the other hyper-parameters the same in full precision network. Figure 3 shows the validation error of PReLU and BCPReLU on CIFAR10-ResNet20. The curves show that the BCPReLU has almost the same expressive ability as PReLU in full-precision network. The performance of BCPReLU is consistent with the theoretical analysis from the Theorem 1.

4.3 Quantization performance for BCPReLU in different bit-width

In this subsection, we give the comparisons between the full-precision network and the low-bit (4,8-bit) network. The curves of validation error with different bit-width are shown in Figure 4, we observe that the validation error is gradually reduced as the bit-width increases. The result is in line with the theoretical analysis in Theorem 2, i.e., when the bit-width M increases, the quantization error of BCPReLU decreases.

4.4 Comparison of Quantization Accuracy with Other methods

To evaluate our method, we compare with PACT Choi et al. (2018) on several well-known CNNs: ResNet20/32 (He et al., 2016) for the CIFAR10 and SVHN datasets. We used stochastic gradient descent (SGD) with momentum of 0.9. The learning rate starts from 0.1 and scaled by 0.1 at epoch 60, 120. The mini-batch size of 128 is used, and the maximum number of epochs is 200. Table 1 and Table 2 summarize quantization accuracy. As can be seen from Table 1 and 2, BCPReLU achieves high accuracy consistently across the networks tested, and outperforms PACT for all the cases. This is reasonable because the parameters of BCPReLU are more flexible to adapt to training than PACT.

Table 1: The Comparison of top-1 accuracy between PACT and BCPReLU on CIFAR-10.

Network	**PACT**		**BCPReLU**	
	4-bit	8-bit	4-bit	8-bit
ResNet20	0.913	0.915	0.914	0.919
ResNet32	0.919	0.923	0.922	0.926

Table 2: The Comparison of top-1 accuracy between PACT and BCPReLU on SVHN.

Network	**PACT**		**BCPReLU**	
	4-bit	8-bit	4-bit	8-bit
ResNet20	0.956	0.961	0.962	0.965
ResNet32	0.96	0.963	0.964	0.967

5 CONCLUSION

In this paper, we propose a novel clipping activation function (BCPReLU) as a generalization of ReLU and its variants, which introduces trainable clipping values and learnable slopes for the output of activation function. The theoretical analysis of the representation ability on the full-precision network and the convergence of the low-bit network has also been established. Extensive experiments on CIFAR10 and SVHN datasets show that the network with BCPReLU maintains almost the same accuracy as corresponding unbounded activation function in full-precision network. In comparison to PACT, the networks with BCPReLU achieve higher accuracy in the low-bit network.

REFERENCES

Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, pp. 7950–7958, 2019.

Yoshua Bengio, Nicholas Leonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *Computer Science*, 2013.

Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pp. 354–370. Springer, 2016.

Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno 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.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. *arXiv preprint arXiv:2003.10027*, 2020.

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

Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang. Accurate and efficient 2-bit quantized neural networks. *Proceedings of Machine Learning and Systems*, 1, 2019.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

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

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

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

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

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

Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *Computer Science*, pp. 229–233, 2015.

Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5229–5238, 2019.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *Computer ence*, 2015.

Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 325–341, 2018.

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

A APPENDIX

In the quantized network, we need notations as follows:

$$\eta_1 = round(-k_1\mu\frac{2^M-1}{k_1\mu+k_2\alpha}), \quad \eta_2 = round(k_2\alpha\frac{2^M-1}{k_1\mu+k_2\alpha}), \quad \beta = \frac{k_1\mu+k_2\alpha}{2^M-1}.$$

Then $y_q = round(y.\frac{1}{\beta})\beta$.

Assume that the activation input x is a random variable, with a probability density function $f(x) = \text{Laplace}(0,b) = \frac{1}{2b}e^{\frac{-|x|}{b}}$, then kx has a density function $h(x) = \text{Laplace}(0,kb) = \frac{1}{2kb}e^{\frac{-|x|}{kb}}$.

Thus, the activation output y has the probability density function

$$h(y) = \begin{cases} \frac{1}{2k_1b}e^{\frac{y}{k_1b}}, & y \in (-\infty, 0) \\ \frac{1}{2k_2b}e^{\frac{-y}{k_2b}}, & y \in [0, \infty) \end{cases}$$

The quantization error between the activation output y and its quantized version y_q is denoted by $error(y)$, which can be written as follows:

$$\begin{aligned} error(y) &= \int_{-k_1\mu}^{k_2\alpha} h(y)(y-y_q)^2 dy \\ &= \int_{-k_1\mu}^{(\eta_1+0.5)\beta} h(y)(y-\eta_1\beta)^2 dy + \int_{(\eta_2-0.5)\beta}^{\alpha} h(y)(y-\eta_2\beta)^2 dy + \\ &\quad \sum_{i=1}^{2^M-2} \int_{(\eta_1+i-0.5)\beta}^{(\eta_1+i+0.5)\beta} h(y)[y-(\eta_1+i)\beta]^2 dy \end{aligned}$$

Because the probability density on the positive input is different from that of the negative input, we need to consider the integral of different intervals when calculating the quantization error. Obviously, $y_q = 0$ when $y = 0$. Assume that $\eta_1 + i^* = 0$, then $0 \in [(\eta_1+i^*-0.5)\beta, (\eta_1+i^*+0.5)\beta]$.

Thus,

$$\begin{aligned} error(y) = &\int_{-k_1\mu}^{(\eta_1+0.5)\beta} h(y)(y-\eta_1\beta)^2 dy + \sum_{i=1}^{i^*-1} \int_{(\eta_1+i-0.5)\beta}^{(\eta_1+i+0.5)\beta} h(y)[y-(\eta_1+i)\beta]^2 dy + \\ &\int_{-0.5\beta}^{0} h(y)y^2 dy + \int_{0}^{0.5\beta} h(y)y^2 dy + \sum_{i=i^*+1}^{2^M-2} \int_{(\eta_1+i-0.5)\beta}^{(\eta_1+i+0.5)\beta} h(y)[y-(\eta_1+i)\beta]^2 dy \\ &+ \int_{(\eta_2-0.5)\beta}^{\alpha} h(y)(y-\eta_2\beta)^2 dy \end{aligned}$$

Defining

$$\begin{aligned} h_1(y) &= \frac{1}{2} e^{\frac{y}{k_1 b}} (y-\eta_1\beta)^2 - (k_1 b) e^{\frac{y}{k_1 b}} (y-\eta_1\beta) + (k_1 b)^2 e^{\frac{y}{k_1 b}} \\ h_2(y) &= -\frac{1}{2} e^{\frac{-y}{k_2 b}} (y-\eta_2\beta)^2 - (k_2 b) e^{\frac{-y}{k_2 b}} (y-\eta_2\beta) - (k_2 b)^2 e^{\frac{-y}{k_2 b}} \end{aligned}$$

The quantization error is calculated by integration by parts as follows:

$$\begin{aligned} error(y) = &h_1(-k_1\mu) + (k_1 b)^2 + (k_2 b)^2 + h_2(k_2\alpha) \\ &- \sum_{i=1}^{i^*-1} (k_1 b\beta) e^{\frac{(\eta_1+i-0.5)\beta}{k_1 b}} - \sum_{i=i^*+1}^{2^M-2} (k_2 b\beta) e^{\frac{-(\eta_1+i-0.5)\beta}{k_2 b}} \end{aligned} \tag{4}$$

If $M \to \infty$, then

$$\begin{aligned} &h_1(-k_1\mu) \to -(k_1 b)^2 e^{\frac{-k_1\mu}{k_1 b}},\ h_2(k_2\alpha) \to -(k_2 b)^2 e^{-\frac{k_2\alpha}{k_2 b}} \\ &\sum_{i=1}^{i^*-1} (k_1 b\beta) e^{\frac{(\eta_1+i-0.5)\beta}{k_1 b}} \to -(k_1 b)^2 (e^{\frac{-k_1\mu}{k_1 b}} - 1) \\ &\sum_{i=i^*+1}^{2^M-2} (k_2 b\beta) e^{\frac{-(\eta_1+i-0.5)\beta}{k_2 b}} \to (k_2 b)^2 e^{-\frac{k_2\alpha}{k_2 b}} \end{aligned} \tag{5}$$

According to the Eq. (5), the quantization error $error(y)$ converge to zero. The proof is completed.