Scale-Clip Net: Shape Distribution For Low-Bit Quantization

ID: 6210

Abstract

Quantization, as an effective strategy for reducing computational cost of deep neural network, has promoted the deployment of the network into computation source-limited devices. Most conventional quantization methods just focus on minimizing the loss between the float point model and quantized model, without considering the relationship between distribution and quantization. In this paper, we turn to exploring what kind of distribution of float point model is good for quantization. We first introduced a quantitative metric to measure the quantized loss ratio (QLR) and experimentally found that small QLR is especially suited for low-bits quantization, which can recover the lost accuracy easily by fine-tuning. Theoretically we proved that the distribution with small ratio of *abs-max* to *abs-mean* leads to small QLR. Based on above theory, we proposed a Scale-Clip method, which clips the outliers of weight and activation with the dynamic threshold, shaping weight and activation into a distribution suited for quantization. With incorporating Scale-Clip method into quantization framework, we perform experiments on the image classification, object detection and semantic segmentation tasks and demonstrate that the proposed method achieves much better performance than the state-of-the-art quantization methods.

Introduction

In recent years, convolutional neural network(CNN) has helped Computer Vision get significant breakthroughs in variety of tasks, such as image classification, object detection, semantic segmentation, etc. However, these deep networks contain millions of learnable weights, and leads to the drastically increasing in computation source cost, which critically restricts the deployment of models in limited computation resources devices with ARM or FPGA. This spawns the research area of model compression to reduce the computational cost. There are many ways to achieve model compression, such as desgining light weight networks like MobileNet and ShuffleNet, pruning the network with less channels and sparser weights, distilling the small network from big network with co-training, quantizing the weight and activation of full precision(i.e., 32-bit floating-point) into low-bit fixed point. Among the existing approaches, quantization has attracted great attention in recent years for its capability of saving computation resources and facilitate the deployment of complex model into computation source-limited devices. In this paper, we focus on the CNN quantization.

Quantization mainly means discretizing weights or activations of full precision(i.e., 32-bit floating-point) into low-bit fixed point number without modifying the model architecture, so that each weight(or activation) can be represented by fewer bits. For example, if weights only take on 16, then each weight could be encoded in 4 bits. In order to obtain low-bit models, previous works have largely been carried out in two aspects. One aspect is to improve the method of training quantized model. The other aspect is to explore the proper quantization strategy.

For the training method, a network training framework for arbitrary bit length is established in the previous work, such as DoReFa-Net and Ristretto []. TWN [] is the first method of training framework for very low bit network weights and achieves good performance on ImageNet. INQ [] and ELQ use an incremental strategy to train the network for low bits. XNOR-Net [] and BNN [] propose training method that simultaneously quantize the network weights and activations with extremely low bits. [] introduces the method of distillation learning into quantization training, using full precision model to guide the process of low-bit training. TSQ decouples the weight and activation quantization and proposes a two-stage training method.

For quantization strategy, one way is to optimize the quantization levels. Some work specify the quantization levels according to the distribution of network weights and activations. [] selects the quantization bins from the weight distributions, ensuring the balanced distributions of quantized values. [] uses weighted entropy to determine the distribution of quantization levels in different ranges of weights and activations. In the low bit quantization scenario, some work use the low rank matrix approximation [], or cast weights into optimal quantization levels to minimize the reconstruction error []. Other work attempt to change the distribution of activation functions. [] utilizes Half Wave Gaussian Quantizer to quantize the activations and introduces some alternative ReLU versions to remove outliers. [] squeezes the dynamic range of the activations by a learnable upper bound, resulting in a more robust quantization performance.

Despite of tremendous advances with above methods,

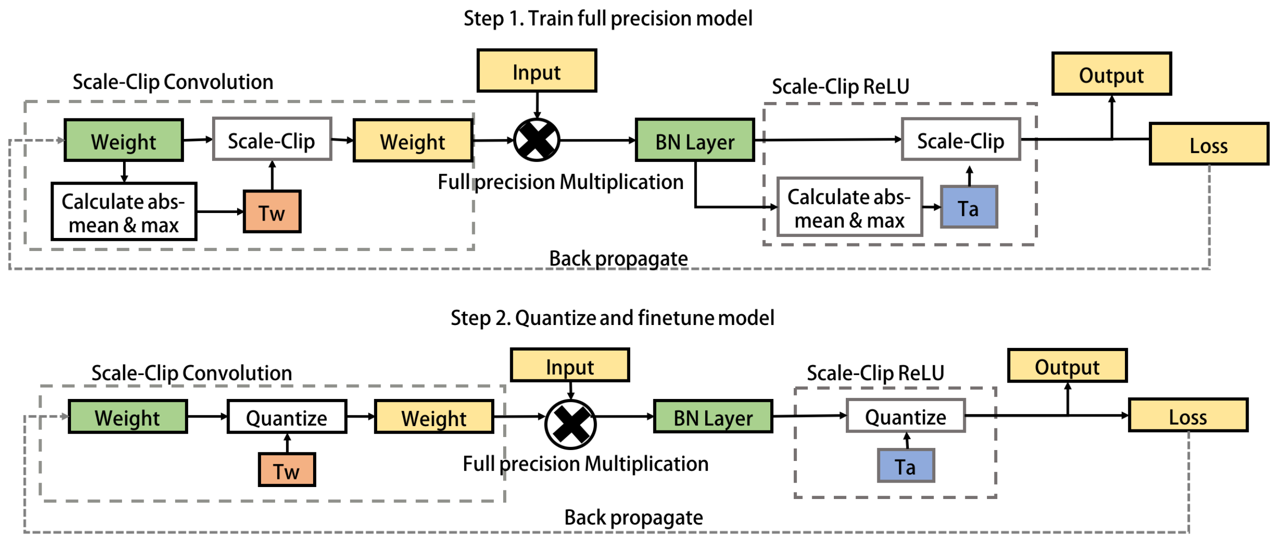

Figure 1: In the Step 1, we train the full-precision model. We substitute the all convolution and ReLU layers with our Scale-Clip version, and adopt Algorithm 1 to update the network. In the Step 2, we analyze the statics of weight and activation, then quantize the full-precision model into low-bit fixed-point model. We adopt the uniform quantization scheme and quantize the whole network simultaneously. In the Step 3, we finetune the quantized model. We adopt *fake-quantization* framework mentioned in [], which forward propagate with fixed-point number, but back propagate the gradient on the full-precision number.

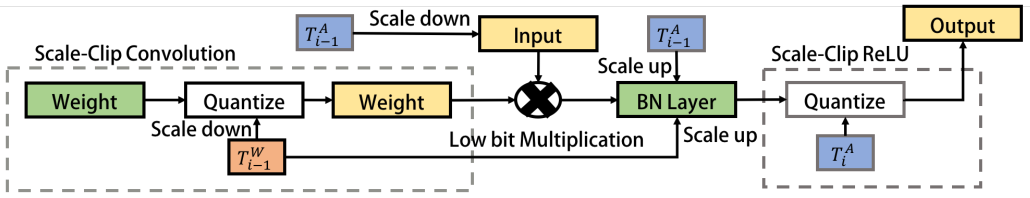

Figure 2: Inference framework. Actually, we often map the input and weight into integer and then quantize them, simultaneously, merge the factor into BN layer.

there is still a key point people may overlook: what distribution is proper for quantization? In this paper, inspired by the phenomenon we observed in engineering that different distribution of full precision models lead to different quantization performance, we will explore what distribution of the full precision model is easy to quantize, and furthermore attempt to shape the distribution into the specific distribution.

We start with the definition of quantization loss ratio(QLR), which is similar to signal noise ratio(SNR) and could measure the signal loss after quantizing. To simplify the discussion, we first only consider quantizing weights without quantizing activations. The empirical idea is that the smaller the QLR is, the more easily can quantized model recover the lost accuracy after finetuning. To get smaller QLR, we find that the weight distribution with small ratio of abs-max to abs-mean holds smaller QLR. The theory is proved by computing the quantized weight loss expectation as an intermediate proof.

Based on above theory, we attempt to explore how to train

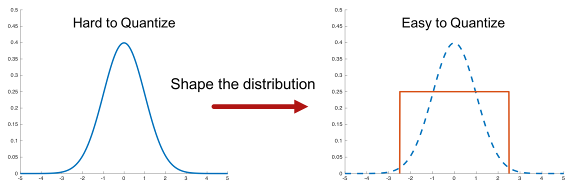

Figure 3: Original distribution of weight is hard to quantize, since the outliers impair the quantization precision. We shape the distribution of weights and make it easier to quantize the network.

the full precision model to the specific distribution while ensuring high performance. We propose a simple Scale-Clip method, which determines the threshold by the weight's average energy and clips the outliers. Afterwards, we generalize the Scale-Clip method to quantizing activation by transforming the vanilla ReLU into Scale-Clip ReLU. We use an illustrating example to show our theory and how the Scale-

Clip method works.

In the end, we apply the Scale-Clip method for weights and activations to our quantization framework to get the final low-bit fixed point model. We evaluate our method on many experiments such as ImageNet classification task and Cityscape segmentation task. The results show that our framework outperforms than other state-of-the-art methods.

The main contributions of this paper lie in the followings:

1. We find and prove that the distribution with small ratio of abs-max to abs-mean is more suitable for quantization.
2. We propose an effective Scale-Clip method to shape the distribution of full precision model into flatter distribution while ensuring high performance.
3. We apply the Scale-Clip method into the quantization framework, and validate that our method outperforms than state-of-art methods in variety of tasks such as ImageNet Classification task.

Scale-Clip Net

In this section, we propose a Scale-Clip method to reshape the distribution of full precision model's weights and activation for better quantization performance. We first state what distribution is good for quantization; then we design an algorithm to implement the Scale-Clip method.

Before all, we give our quantization training and inference framework to get low-bit fixed-point model without explanation.

Training Framework Our training framework is composed of three parts: trainging a full-precision model with Scale-Clip method for weight and activation, quantizing the full-precision model into low-bit fixed-point model, and recover the lost accuracy with fine-tuning. For fine-tuning, we adopt *fake-quantization* framework mentioned in [], which forward propagate with fixed-point number, but back propagate the gradient on the full-precision number. The framework is also displayed in Fig ,

Inference Framework The core of our quantization inference framework is composed of: quantizing the weight and input, operating in full-precision number, then quantizing the output. Since the output is also another layer's input, we often ignore quantizing the output. The inference framework is also performed in Fig 2.

Distribution Exploration

If we regard trained CNN as a nonliner regressor, then it could be denoted as

$$f^* = f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2 \dots, \mathbf{W}_m) \tag{1}$$

where $\mathbf{x}$ is the input of CNN, $\mathbf{W}_i$ is the ith convolution layer's weight of trained model, and $\mathbf{W}_i = \{w_{ij}, j = 1 \cdots, n_i\}$. We denote $\mathbf{I}_i$ as the input of ith convolution layer, $\mathbf{O}_i$ as the output of ith convolution layer, then $\mathbf{O}_i = \mathbf{I}_i \otimes \mathbf{W}_i$.

To simplify the discussion, we consider a quantized model obtained by quantizing weights $\mathbf{W}_i$ with following form

$$\tilde{f}^* = f(\mathbf{x}; \tilde{\mathbf{W}}_1, \tilde{\mathbf{W}}_2 \dots, \tilde{\mathbf{W}}_m) \tag{2}$$

, where $\tilde{\mathbf{W}}_i$ as quantized $\mathbf{W}_i$, then $\tilde{\mathbf{O}}_i = \mathbf{I}_i \otimes \tilde{\mathbf{W}}_i$. According to the linear property of convolution, we have

$$\begin{aligned} \Delta\mathbf{O}_i &= \mathbf{O}_i - \tilde{\mathbf{O}}_i \\ &= \mathbf{I}_i \otimes \mathbf{W}_i - \mathbf{I}_i \otimes \tilde{\mathbf{W}}_i \\ &= \mathbf{I}_i \otimes (\mathbf{W}_i - \tilde{\mathbf{W}}_i) \\ &= \mathbf{I}_i \otimes \Delta\mathbf{W}_i \end{aligned} \tag{3}$$

Since the signal propagates through layer by layer, similar to $\mathbf{SNR}$(signal noise ratio), we have

$$\mathbf{SNR}_i = \frac{||\Delta\mathbf{O}_i||_F^2}{||\mathbf{O}_i||_F^2} \tag{4}$$

Due to the powerful capacity of CNN, there are numerous regressors with different weight distribution that can achieve high performance. So here comes up an intuitive idea: a model with small quantization $\mathbf{SNR}$ will largely maintain its performance after quantization, and can also easily retrieve its performance by finetuning.

Suppose the input data can be modeled as a Gaussian Distribution $\mathcal{N}(0, \sigma^2)$. According to *Parseval* Theorem,

$$\begin{aligned} \mathbf{SNR}_i &= \frac{||\mathbf{I_i} \otimes \Delta\mathbf{W}_i||_F^2}{|\mathbf{I_i} \otimes \mathbf{W_i}||_F^2} \\ &= \frac{||\mathcal{F}(\mathbf{I_i})\mathcal{F}(\Delta\mathbf{W}_i)||_F^2}{||\mathcal{F}(\mathbf{I_i})\mathcal{F}(\mathbf{W_i})||_F^2} \\ &= \frac{||\Delta\mathbf{W}_i||_F^2}{||\mathbf{W_i}||_F^2} \\ &= \frac{\mathbb{E}(\Delta\mathbf{W}_i^2)}{\mathbb{E}|\mathbf{W_i}^2|} \\ &\le \frac{\mathbb{E}(\Delta\mathbf{W}_i^2)}{\mathbb{E}^2|\mathbf{W_i}|} = \mathbf{QLR}_i^2 \end{aligned} \tag{5}$$

. where we denote **Q**uantization **L**oss **R**atio ($\mathbf{QLR}_i$) as:

$$\mathbf{QLR}_i = \frac{\mathbb{E}^{\frac{1}{2}}(\Delta\mathbf{W}_i)}{\mathbb{E}|\mathbf{W_i}|} \tag{6}$$

Hence we convert optimizing the $\mathbf{SNR}_i$ into optimizing its upper bound $\mathbf{QLR}_i^2$, which is more simple for calculation and analysis. We firstly propose Theorem 1 to optimize $\mathbf{QLR}_i$.

Theorem 1 *Suppose that $\mathbf{W}_i$ are quantized into n bits, $\mathbf{QLR}_i \propto \frac{\max|\mathbf{W}_i|}{\mathbb{E}|\mathbf{W}_i|}$.*

Proof Denote $\max|\mathbf{W}_i|$ as α. Using the uniform quantization scheme,

$$\tilde{\mathbf{W}}_i = round(\tilde{\mathbf{W}}_i \cdot \frac{2^{n-1}-1}{\alpha}) \cdot \frac{\alpha}{2^{n-1}-1} \tag{7}$$

then $\mathbf{W}_i$ can be rewritten as the sum of quantized value and quantization error $\Delta\mathbf{W}_i$.

$$\mathbf{W}_i = \tilde{\mathbf{W}}_i + \Delta\mathbf{W}_i \tag{8}$$

where $\Delta\mathbf{W}_i = \mathbf{n}_i \frac{\alpha}{2^{n-1}-1}, \mathbf{n}_i \in [-\frac{1}{2}, \frac{1}{2})$. We denote $\mathbb{E}(\mathbf{n}_i^2)$ as n_q^2, which is exactly the variance when n_q has zero mean. Then $\mathbb{E}^{\frac{1}{2}}(\Delta\mathbf{W}_i)$ can be further written as

$$\begin{aligned}\mathbb{E}(\Delta\mathbf{W}_i) &= \frac{\alpha}{2^{n-1}-1}\mathbb{E}^{\frac{1}{2}}(\mathbf{n}_i^2) \\ &= \frac{\alpha n_q}{2^{n-1}-1}\end{aligned} \tag{9}$$

So it can be drawn out that

$$\mathbf{QLR}_i = \frac{\mathbb{E}^{\frac{1}{2}}(\Delta\mathbf{W}_i)}{\mathbb{E}|\mathbf{W}_i|} = \frac{\alpha}{\mathbb{E}|\mathbf{W}_i|} \cdot \frac{n_q}{2^{n-1}-1} \tag{10}$$

Furthermore, the quantization variance n_q^2 is independent to weights, hence we have that **QLR** is proportion to the $\frac{\max(|\mathbf{W}_i|)}{\mathbb{E}(|\mathbf{W}_i|)}$. ■

Scale-Clip Method

To get a better quantized model, we should restrict the weight distribution to get ratio as small as possible:

$$\begin{aligned}&\min_{\mathbf{W}_i} \mathbf{QLR} \\ &s.t. \\ &\quad \mathbf{y} = f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2 \ldots, \mathbf{W}_m)\end{aligned} \tag{11}$$

Based on Theorem 1, the (11) is equivalent to the following form:

$$\begin{aligned}&\min_{\mathbf{W}_i} \frac{\max|\mathbf{W}_i|}{\mathbb{E}|\mathbf{W}_i|}, i = 1, \cdots, m \\ &s.t. \\ &\quad \mathbf{y} = f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2 \ldots, \mathbf{W}_m)\end{aligned} \tag{12}$$

However, it is difficult to directly solve (12). Instead, we propose an effective Scale-Clip method to restrict the distribution of weights and activations of the full precision model. For weights, Scale-Clip method is to clip weights with the dynamic threshold relied on the average energy of weights. One simple form is defined as:

$$Q(w) = \begin{cases} T_i^w, & w \geq T_i^w \\ w, & w \in (-T_i^w, T_i^w) \\ -T_i^w, & w \leq -T_i^w \end{cases} \tag{13}$$

$$T^w = k \cdot \mathbb{E}|\mathbf{W}_i| \tag{14}$$

where k is called Scale-Clip factor which needs to be set specifically.

Usually k cannot be trivially obtained since the $\frac{\max|\mathbf{W}_i|}{\mathbb{E}(|\mathbf{W}_i|)}$ is difficult to foreknow before training. To better estimate the reasonable range of Scale Clip factor k, we assume the initial distribution of $\mathbf{W}_i$ follows $\mathcal{N}(0, \sigma)$. Then $|\mathbf{W}_i|$ follows a half Gaussian Distribution. The proportion of clipped values can be calculated by cumulative distribution function $\text{cdf}(x) = \text{erf}(\frac{x}{\sigma\sqrt{2}})$,

$$k(\frac{\sqrt{2}}{\sigma\sqrt{\pi}}\int_0^{T^w} xe^{-\frac{x^2}{2\sigma^2}}dx + T^w(1-\text{cdf}(T^w)) = T^w \tag{15}$$

where erf$(\cdot)$ is the error function. Fig 4 depicts the relationship between Scale-Clip factor k and the ratio of numbers clipped by thresholds. It can be observed that k monotonically converges to 1 when the ratio approaches 1. Smaller k indicates more information of weights is clipped, increasing the risk of degrading performance of the model. We set $k \in (2, 3)$ corresponding to the clipped value proportion about in $(0.01, 0.15)$, in order to balance between minimizing QLR and performance of the network.

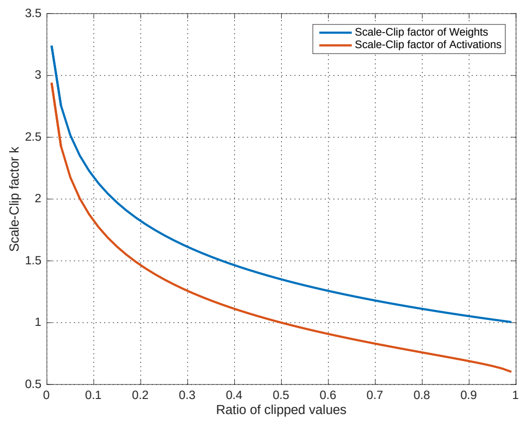

Figure 4: Relationship between Proportion of clipped values and Scale Clip factor k.

For activation, we replace all vanilla ReLU layers with our Scale-Clip ReLU layers. Different from determining the threshold directly as (14), the mean of activations depends on the input data. We use **E**xponential **M**oving **A**verage (EMA) to estimate the mean of activation similar as [Google].

$$\begin{aligned}T^a(t) &= k \cdot \text{EMA}^t(|\mathbf{A}_i|)) \\ &= k \cdot (\mu\mathbb{E}(\mathbf{A}_i) + (1-\mu)\text{EMA}^{t-1}(|\mathbf{A_i}|)) \\ &= \mu k\mathbb{E}(\mathbf{A}_i) + (1-\mu)T^a(t-1)\end{aligned} \tag{16}$$

where $\mu \in (0, 1)$. The EMA algorithm introduces a fluctuation of threshold when we train the model with Scale-Clip ReLU layer. So a small μ is used in our implementation.

Different from weights, the vanilla ReLU layer clips input values by zero. We also assume the original input data follow Gaussian Distribution, then the relationship between proportion of clipped numbers and k can be obtained in the same way. Here $\text{cdf}(x) = \frac{1}{2}\left[1 + \text{erf}(\frac{x}{\sigma\sqrt{2}})\right]$.

$$k(\frac{1}{\sqrt{2\pi}\sigma}\int_0^{T^a} xe^{-\frac{x^2}{2\sigma^2}}dx + T^a(1-\text{cdf}(T^a)) = T^a \tag{17}$$

It can be seen that similar relationship are shared with the Scale-Clip factor of weights. So a same scheme is utilized for select the Scale-Clip factor of activations. Our training process with Scale-Clip is shown in following Algorithm 1.

Algorithm 1 Process of Training Float Point Model with Scale-Clip

```
Require: input data, CNN architecture,$k$
Ensure: trained float point model
 1: initiate the CNN model
 2: for $epoch$ do
 3:     for each Layer do
 4:         calculate the threshold of Scale-Clip with $T_i^w = k \cdot \mathbb{E}|\mathbf{W}_i|$ and $T_i^a = k \cdot \mathbb{E}|\mathbf{A}_i|$
 5:         clip the $W_i$ with $T_i^w$
 6:         clip the $A_i$ with $T_i^a$
 7:     end for
 8:     forward & backward
 9:     for each Layer $W_i$ do
10:         update the weight
11:     end for
12: end for
13: return float point model
```

Experimental Results

In this section, we evaluate our method and framework by conducting extensive quantization experiments. We first demonstrate that Scale-Clip method helps to shape the weight distribution of full precision network into a distribution suitable for quantization. Then we report our quantization result on a ImageNet classification task and compare it with other low bit quantization work. We also demonstrate that our Scale-Clip method can be generalized to other tasks, such as detection and semantic segmentation. We implement our Scale-Clip Net by PyTorch framework.

Analysis of Scale-Clip Method

To analysis the versatility of Scale-Clip Method, we examine the effect of Scale-Clip algorithm in different quantization settings.

Experiment Settings We choose a simple network, ResNet18, and train it on CIFAR-100. We set mini batch size as 64, and our learning rate is 0.1. The number of epochs is set as 150, except for 5 warm-up epochs. SGD is used for optimizing the parameters, with the momentum as 0.9 and weight decay as 0.0005. Besides, we apply Scale-Clip method on all regular convolution layers. And Scale-Clip factor ks are tested to show the its effect to the model. In our experiments, we set k as 2, 2.5, 3, 4 and ∞ according to the recommended range mentioned before. To be mentioned, as the Scale-Clip factor k goes infinity, the Scale-Clip Net comes down to a regular network.

Result We compare the distributions of weights trained in different Scale-Clip factors. Fig 4 shows the how different ks shape the distribution of trained weights. The distribution of regular convolution weights produce a distribution similar to a Gaussian, with most of data lying around central part but some resting in its tail. When k gets smaller, outliers of distribution are removed, resulting a flat distribution. When k is 2.5 or 2, however, the range narrows, and two peaks start emerging at the clipping boundaries, forming a distribution similar to Ternary Distribution.

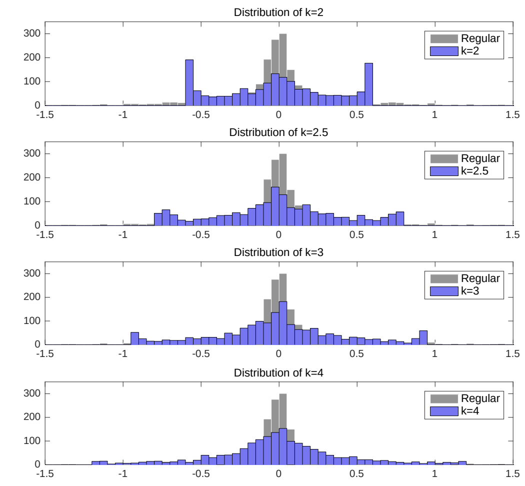

Figure 5: Weight distribution of different Scale-Clip factor. For comparison, the background is the distribution of weights trained without Scale-Clip

Fig 6 plots the curves of quantization bit number vs. QLR in different Scale-Clip k. It can be observed that the curve of regular convolution settings($k = \infty$) is above other curves and with smaller Scale-Clip factor, the QLR is consistently lower than the lager Scale-Clip factor in all quantization bit numbers. This indicates that training with Scale-Clip method can alleviate the QLR to a great extent, especially in low bit, so that a higher quantization performance can be expected.

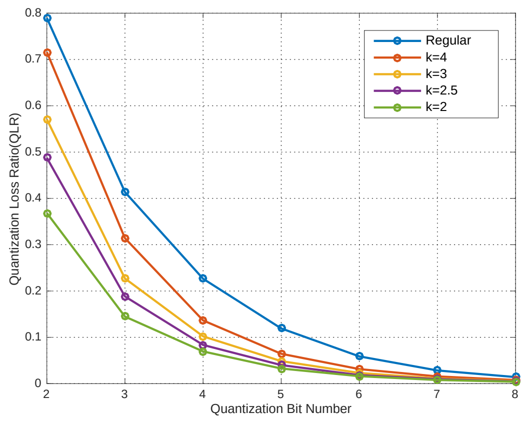

Figure 6: Loss Ratio After Quantizing First Layer's weights. The Regular curve is the QLR trained without Scale-Clip.

We show the accuracy of different k settings in Fig 7. First, the results of full precision model show that if we decrease k, the performance of the model will gets worse. This is because Scale-Clip restricts the distribution of weights and impairs the network performance. However, only a fraction of weights are clipped as our ks lie in the recommended range, the accuracy merely drops from the best result within 1%. As for quantization performance, We quantize the weights of trained model from 2 to 8 bits, with activations as full precision. Results shows the model quantized in relatively large bits only suffer a little performance degradation. As the Scale-Clip factor decreases, the degradation becomes negligible, which is in accordance with the relationship between QLR and k. Moreover, lower Scale-Clip factors show their advantages in extremely low bit quantization scenario. We can see that the performance of 2/3bit quantization in $k = \{2, 2.5\}$ surpasses larger k's counterpart with over 4%. The results reflect the effectiveness of Scale-Clip for its capability of make the distribution of weight more suitable for low bit quantization.

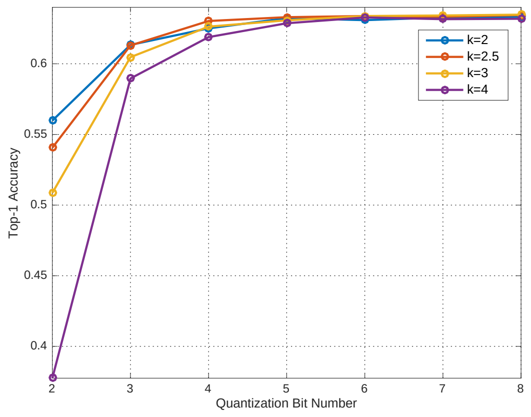

Figure 7: Top-1 Accuracy of different Scale-Clip factors.

Results On ImageNet

ImageNet ImageNet dataset has about 1.2 million training images and 50 thousand validation images. Each image is annotated as one of 1000 object classes. The size of each image is 256×256, and we crop the training image by 224×224 as input image, and use the center crops of validation image to validate the model.

ResNet50 ResNets50 has batch normalization layers and relief the vanishing gradient problem by using shortcut connections. Compared to ResNet18, ResNet50 has 1×1 convolution layers. In the training process of full precision model, we substitute the ReLU layers with our Scale-Clip ReLU layers, and set the scale clip factor as 6. We also use the Scale-Clip for weights and set the scale clip factor as 2. In the training settings, the learning rate is 0.1, batch size is 64, maximal epoch number is 150, the weight decay is 0.0005, and the momentum is 0.9. We use the SGD method to optimize the model.

Table 1: Accuracy Comparison with [4,4]

Accu \ Bits / Model	Top-1	full prec	Top-5	full prec
DoReFa-Net	71.4	-	86.6	-
Ours(k=2,6)	72.7	74.7	90.9	92.1

Table 2: Accuracy Comparison with [2,8]

Accu \ Bits / Model	Top-1	full prec	Top-5	full prec
SYQ	72.6	76	90.6	93
FGQ	70.8	-	-	-
DoReFa-Net	68.6	-	90.6	-
Ours(k=2,6)	73.6	74.7	91.6	92.1

Results We quantize the convolutional layers' weight and input activation with the combination of [2 bit, 8 bit] and [4 bit, 4 bit]. In the fine-tuning step, the learning rate starts at 0.001 and is divided by 10 every 20 epochs, the batch size is 64, the weight decay is 0.0005, and the momentum is 0.9. In the inference process, we fix the threshold of the Scale-Clip ReLU layers and remove all Scale-Clip operations.

We choose the best accuracy in fine tuning process to compare with some open benchmarks reported from SYQ, FGQ and DoreFa-Net, which are the sate-of-the-art methods. The results are illustrated in Table From Table 1 and Table 2, although the performance of full precision model which adds our Scale-Clip method is not good as other full precision model, the final low-bit model outperforms that these methods more than 1%.

Versatility of Scale Clip

To demonstrate the versatility, we validate ScaleClip method on other more complex computer vision task. We perform object detection on PASCAL-VOC dataset and Semantic Segmentation on Cityscape dataset. Both quantization result and finetuning result based on quantization model will be reported.

Detection on PASCAL-VOC We choose Faster-RCNN with ResNet-50 backbone, a popular detection architecture as the baseline model. Different from classification models above, Faster-RCNN is composed of Region Proposal Network, RoI-pooling, Bounding Box Regression and other modules, where more complex operations are involved so that quantization cannot be well defined every where. Also, we keep Bounding Box Regression as full precision for a

Table 3: mAP of PASCAL-VOC

Model	float	(8,8)	(4,8)	(4,4)	(2,4)
Ours(k=3)	79.0	78.7	78.8	78.1	73.2
Ours(k=3)+f	79.0	-	-	79.0	78.3
Weighted Q	77.61	73	66	-	-

Table 4: mIoU of Cityscape

Model	float	(8,8)	(4,8)	(4,4)	(2,4)
Ours(k=3)	75.6	75.31	72.48	68.5	38.24
Ours(k=3)+f	75.6	75.66	75.29	75.62	74.7

more precise regression result. So in our experiment, only convolution layers are quantized and all ReLU layers are replaced as Scale-Clip ReLU. As for the training settings, a ImageNet pretrained model is used to initialize the ResNet-50 backbone. For PASCAL-VOC, We set the learning rate as 0.1, batch size as 8, epoch number as 50 except for 5 warm-up epochs, the weight decay as 0.0005, the momentum as 0.9. As for fine-tuning step, we set the learning rate as 0.0001, epoch number as 10, and other hyper parameters as the same as their training settings.

Segmentation on Cityscape We test Scale-Clip method on the image segmentation network, PSPNet with ResNet-50 backbone. We replace all regular convolution layers and ReLU layers in PSPNet as the Scale-Clip version. As for the training settings, we train from a ImageNet pretrained model. We set the learning rate as 0.01, batch size as 8, epoch number as 500 except for 5 warm-up epochs, the weight decay as 0.0005, the momentum as 0.9. As for fine-tuning step, we set the learning rate as 0.0001, epoch number as 50, and other hyper parameters as the same as their training settings.

Result Like results in classification, different quantization bits and k for Scale-Clip are explored. We keep the input image as 8bit throughout the experiment. Even through the detection task is more difficult than classification, we can observe a good quantization performance of Scale-Clip method. Notice that without finetuning, there is only a little decrease in mAP. Except for $(2, 4)$, there is only a slight loss within 1% in other quantization settings, indicating that our Scale-Clip method can well maintain the performance after quantization. We compare our result with Weighted Entropy Loss. It can be observed that after finetuning, our mAP results is consistently better than other work at different quantization bits, demonstrating that our quantization method can be generalized to detection task and better than other quantization methods. Like results shown above, we test our Scale-Clip on different quantization bits for Scale-Clip. Input images are set as 8 bit throughout the experiment. From the results shown in Table , it can be found that after finetuning, the mIoU results in all quantization bit settings only suffer a negligible decrease, even in the extremely low bits. However, for the quantization performance without finetuning, we observe a drop from full precision model (around 3-10%). We believe that the reason is the difficulty of fitting segmentation results on Cityscape causing the trained network more sensitive to quantization noise. In summary, our Scale-Clip method can also achieve a good performance in segmentation task.

Conclusion

In this paper, we start from SNR and analyze the relationship between the distribution of weights and activation. We present a ratio called QLR to indicate the performance decrease, and prove that it is proportional to the ratio of abs-max to abs-mean. To minimize the QLR and improve quantization performance, we propose a novel approach called Scale-Clip to force the weights and activations converge to a distribution suitable for quantization. We further discuss the trade-off between minimizing the QLR and the loss of information caused by Scale-Clip method, and offer a reasonable range of Scale Clip factor k. And then we incorporate Scale Clip method into regular convolution and ReLU layers and design the quantization training framework. We demonstrate our method on several kinds of tasks, comparing the quantization performances with and without fine-tuning. We quantize models of different architectures and achieve only with 1% loss in performance with low bit weights (2bit/4bit) and activations (4bit). We show our approach can not only outperform than other state-of-the-art low bit quantization works, but can also be generalized to many common computer vision tasks. Further research includes the how robust a network is to quantization error and how to adaptively set the value of k without specifically tuning by hand.