Improving Neural Network Quantization without Retraining using Outlier Channel Splitting

Ritchie Zhao [1] **Yuwei Hu** [1] **Jordan Dotzel** [1] **Christopher De Sa** [1] **Zhiru Zhang** [1]

Abstract

Quantization can improve the execution latency and energy efficiency of neural networks on both commodity GPUs and specialized accelerators. The majority of existing literature focuses on training quantized DNNs, while this work examines the less-studied topic of quantizing a floating-point model without (re)training. DNN weights and activations follow a bell-shaped distribution post-training, while practical hardware uses a linear quantization grid. This leads to challenges in dealing with *outliers* in the distribution. Prior work has addressed this by clipping the outliers or using specialized hardware. In this work, we propose outlier channel splitting (OCS), which duplicates channels containing outliers, then halves the channel values. The network remains functionally identical, but affected outliers are moved toward the center of the distribution. OCS requires no additional training and works on commodity hardware. Experimental evaluation on ImageNet classification and language modeling shows that OCS can outperform state-of-the-art clipping techniques with only minor overhead.

1. Introduction

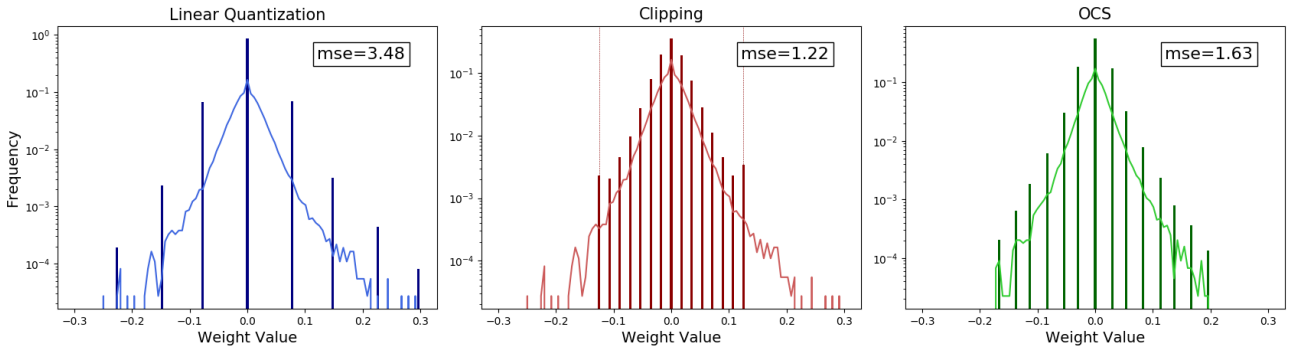
Over the past few years, deep neural networks (DNNs) have become the state-of-the-art approach for many large-scale computer vision and sequence modeling problems. Deep convolutional networks dominate the leaderboards for popular image classification and object detection datasets such as ImageNet (Deng et al., 2009) and Microsoft COCO (Lin et al., 2014). However, the significant compute and memory requirements of running DNNs impedes the adoption of neural nets in application domains such as edge computing or latency-critical services (Xu et al., 2018). One approach to reducing the costs of DNN execution is to quantize the floating-point weights and activations into low-precision fixed-point numbers. This reduces the model size as well as the complexity of multiply-accumulate (MAC) operations in hardware, enabling better throughput and energy efficiency. DNN quantization is an active area of research (Wu et al., 2018; Jacob et al., 2018; Choi et al., 2018b; Banner et al., 2018) and sees deployment in commercial systems such as Google's TPU (Jouppi et al., 2017), NVIDIA's TensorRT (Migacz, 2017), and Microsoft's Brainwave (Chung et al., 2018).

The majority of literature on DNN quantization involves training — either from scratch (Courbariaux et al., 2015; Wu et al., 2018; Jacob et al., 2018) or retraining/fine-tuning from a floating-point model (Han et al., 2016; Zhou et al., 2017). Although such techniques are valuable, there are important real-world scenarios in which (re)training is not applicable. Consider an ML service provider (e.g. Amazon, Microsoft, Google) which wants to run a black-box floating-point client model in low-precision. The service provider does not have the training data, and the client may not be able to train for quantization because: (1) it lacks the expertise or manpower; (2) it is using an off-the-shelf or legacy model for which training data is not available. The importance of *post-training* quantization can be seen from NVIDIA's TensorRT, a product specifically designed to perform 8-bit integer quantization without (re)training. This paper focuses on post-training DNN quantization.

DNN weights and activations follow a bell-shaped distribution after training. However, commodity hardware uses a linear number representation with evenly-spaced grid points. The naïve approach is to linearly map the entire range of the distribution to the range of the quantization grid (Figure 1(a)). Here the grid points extend to the maximum value in the distribution (Hubara et al., 2017). Clearly, this method over-provisions grid points for the rarely-occurring *outliers*. A better approach is to make the grid narrower than the distribution — this is known as *clipping*, as it is equivalent to thresholding the outliers before applying linear quantization (Figure 1(b)). Empirically, clipping can improve the accuracy of quantized DNNs, and many techniques exist to choose the optimal clip threshold (Sung et al., 2015;

[1]Cornell University, Ithaca, New York 14850, USA. Correspondence to: Ritchie Zhao <rz252@cornell.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.



(a) (b) (c)

Figure 1. **Weight histograms for linear, clipping, and OCS quantization techniques**. The floating-point weight histogram is in the light color while the quantized weight histogram is in dark color. Both clipping and OCS reduces mean squared quantization error (MSE) by reducing the dynamic range of the distribution. Clipping reduces the overall MSE but greatly distorts the outliers. OCS avoids this distortion by splitting the outliers instead, moving them towards the center of the distribution at the cost of some model size overhead.

Zhuang et al., 2018; Migacz, 2017). Unfortunately, clipping can only reduce overall quantization error by increasing the distortion on the outliers — it is constrained by this tradeoff.

Another approach to handling outliers is to quantize them separately from the central values. Such outlier-aware quantization (Park et al., 2018a;b) is highly effective, but involves the use of dedicated non-commodity hardware.

In this paper, we propose *outlier channel splitting* (OCS). OCS identifies a small number of channels containing outliers, duplicates them, then halves the values in those channels. This creates a functionally identical network, but moves the affected outliers towards the center of the distribution (Figure 1 (c)). OCS takes inspiration from Net2Net (Chen et al., 2016); it does not require retraining and can be used on commodity CPUs and GPUs. OCS introduces a new tradeoff: it reduces quantization error at the expense of making the neural network larger. Experimental evaluation shows that for practical CNN and RNN models, OCS can significantly improve post-training quantization accuracy over state-of-the-art clipping methods with just a few percent overhead.

To present a comprehensive study of post-training quantization, we also evaluate different techniques for optimizing the clip threshold on both weights and activations. To our best knowledge, we are the first to perform a detailed literature comparison. Code for both OCS and clipping is available in open source [1]. Our specific contributions are as follows:

1. We propose outlier channel splitting, a technique to improve DNN model quantization that does not require retraining and works with commodity hardware.

2. We present a comprehensive evaluation of post-training clipping techniques found in literature. To our best knowledge this is the first such study.

3. We demonstrate that OCS can outperform state-of-the-art clipping techniques on weight quantization, while incurring negligible overheads.

[1] https://github.com/cornell-zhang/dnn-quant-ocs

2. Related Work

2.1. Post-Training Quantization

Clipping is the state-of-the-art for DNN quantization without training. Clipping can be applied to both weights and activations — for the latter the activation distributions are sampled from a small number of inputs. (Sung et al., 2015) and (Shin et al., 2016) examined post-training quantization for CNNs and RNNs, respectively. They adopt a clip threshold that minimizes the L2-norm of the quantization error. ACIQ (Banner et al., 2018) fits a Gaussian and Laplacian to the sampled distribution, then uses the better-fitting curve to analytically compute the optimal clip threshold. In a similar vein, SAWB (Choi et al., 2018a) linearly extrapolates the clip threshold using statistics from fitting six different distributions. (McKinstry et al., 2018) clips the weights to a multiple of its standard deviation, and the activations to a percentile of the sampled values (99.99th for 8-bit and 99.9th for 4-bit). Different from the others, (Settle et al., 2018) tunes the bitwidth and floating-point format, achieving 32-bit accuracy performance with only 8-6 bits.

NVIDIA's TensorRT (Migacz, 2017) is a commercial library that quantizes floating-point models to 8-bit for GPU inference. Clipping is used for the activations to control the effect of outliers. TensorRT profiles the activation distributions using a small number (1000s) of user-provided training samples, then computes a clipping threshold by minimizing the KL divergence between the original and

quantized distributions.

OCS is different from these works as it leverages model expansion to improve quantization.

2.2. Outlier-Aware Quantization

Park et al. propose outlier-aware quantization (Park et al., 2018b;a), which uses a low-precision grid for the center values and a high-precision grid for the outliers. Placing 3% of values on the high-precision grid enabled post-training quantization of many popular CNN models to 4-bit without accuracy loss. This technique requires a specialized outlier-aware DNN accelerator; our approach is very different as it is designed to be applicable on commodity hardware.

2.3. Net2Net

OCS is inspired by Net2Net (Chen et al., 2016), which presents transformations to make a neural network wider or deeper while preserving functional equivalence. The goal of Net2Net was to speed up training by expanding a smaller DNN model into a larger one; the larger model inherits knowledge from the smaller and does not need to be trained from scratch. In this work we apply the `Net2WiderNet` transform to reduce outliers and improve quantization.

2.4. Cell Division

Cell division (Park & Choi, 2019) examines the same idea as OCS, and was published concurrently with our work. Though conceptually very similar, there are some technical differences between their work and ours: (1) they apply OCS on weights only while we examine both weights *and activations*; (2) they first tune the fixed-point bitwidths per layer with 50K training images while we use no data for weight OCS; (3) they do not compare against clipping as a baseline; (4) they evaluate on MNIST, CIFAR-10, and AlexNet while we evaluate on more modern (i.e. post-ResNet) ImageNet CNNs and language models.

3. Outlier Channel Splitting

3.1. Linear Quantization

The simplest form of linear quantization maps the inputs to a set of discrete, evenly-spaced grid points which span the entire dynamic range of the inputs. The maximum quantization error for any single value is one-half of the increment. For symmetric k-bit quantization, we have $2^k - 1$ grid points:

$$\textbf{LinearQuant}(\mathbf{x}) = \text{round}\left(\frac{\mathbf{x}\,(2^{k-1}-1)}{\max(|\mathbf{x}|)}\right)\frac{\max(|\mathbf{x}|)}{2^{k-1}-1} \tag{1}$$

Because each value is scaled by $\max(\mathbf{x})$, **LinearQuant** is sensitive to the largest inputs, i.e. the outliers. Many existing works first clip the range of $\mathbf{x}$ prior to linear quantization; a survey of such clipping techniques can be found in Section 4.

3.2. Improving Quantization with `Net2WiderNet`

The core idea of OCS is to reduce the magnitude of outlier weights and/or activations in a DNN layer by duplicating a neuron, then either (1) halving its output; (2) halving the outgoing weight connections. This leaves the layer functionally equivalent but makes the weight/activation distribution narrower and thus more suitable for linear quantization. Such layer transformations were originally proposed as `Net2WiderNet` in Net2Net (Chen et al., 2016); we leverage them to improve quantization.

More formally, consider a linear layer in a DNN which takes as input the m-channel activation vector $\mathbf{x} = \{\mathbf{x}_i\}_{i=0}^{m}$, where each $\mathbf{x}_i$ can be a single value (FC layer) or a 2D feature map (conv layer). Let $\mathbf{y} = \{\mathbf{y}_j\}_{j=0}^{n}$ be the n-channel output. We can define a linear layer as follows:

$$\mathbf{y}_j = \sum_{i=1}^{m} \mathbf{x}_i * \mathbf{W}_{ij} \tag{2}$$

where $\mathbf{W}_{ij}$ represents the weight(s) connecting $\mathbf{x}_i$ and $\mathbf{y}_j$ and $*$ represents multiplication or 2D convolution over a single channel. Without loss of generality, consider using OCS to split the last channel $\mathbf{x}_m$. This equates to rewriting Equation 2 as follows:

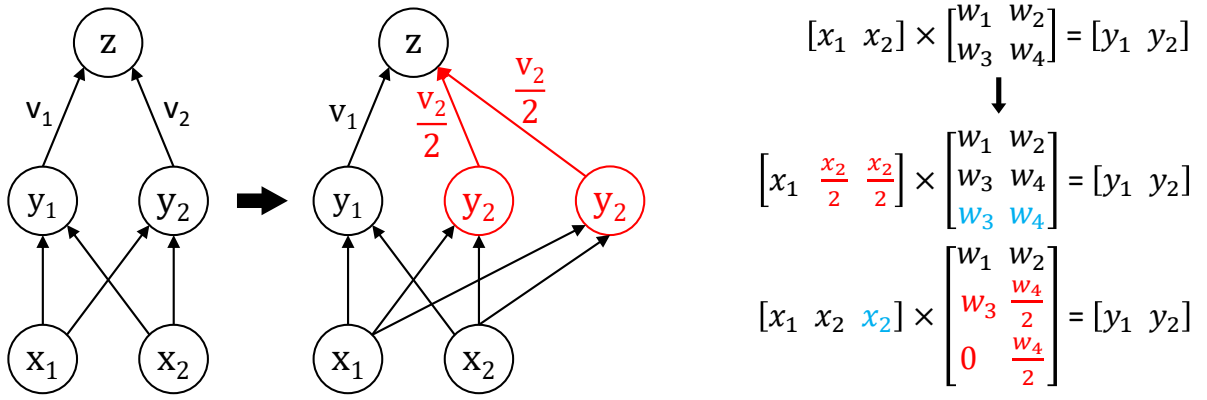
$$\mathbf{y}_j = \sum_{i=1}^{m-1} \mathbf{x}_i * \mathbf{W}_{ij} + (\mathbf{x}_m * \frac{\mathbf{W}_{mj}}{2}) + (\mathbf{x}_m * \frac{\mathbf{W}_{mj}}{2}) \tag{3}$$

$$\mathbf{y}_j = \sum_{i=1}^{m-1} \mathbf{x}_i * \mathbf{W}_{ij} + (\frac{\mathbf{x}_m}{2} * \mathbf{W}_{mj}) + (\frac{\mathbf{x}_m}{2} * \mathbf{W}_{mj}) \tag{4}$$

In both cases, we split channel m into 2 channels. To preserve equivalence, we can halve the weights (Equation 3) or halve the input activations (Equation 4). Figure 2(a) taken from the Net2Net paper illustrates weight OCS visually: by duplicating y_2 we can cut its outgoing weight v_2 in half.

OCS is an alternative to clipping for reducing the dynamic range of DNN values without retraining. Compared to clipping, OCS preserves the outliers but incurs additional network overhead. The outlier values are the largest values in a layer and contribute the most to the outputs. We expect OCS to outperform clipping in neural network accuracy — the question is whether it can do so with low overhead.

Figure 2(b) shows some additional caveats of OCS in a layer with 2 inputs and 2 outputs. The top equation describes the original layer; the next two equations illustrate OCS



(a) (b)

Figure 2. **OCS network transformation –** after duplicating a neuron, we can divide either the neuron's output value or its outgoing weights in half to preserve functional equivalence. (a) figure taken from Net2Net (Chen et al., 2016) where the weight v_2 is halved by duplicating y_2; (b) an example with multiple inputs and multiple outputs, showing how x_2 or w_3 can be halved while maintaining the same outputs. Here, an entire row must be added to the weight matrix to split a single value.

to split the activations and the weights, respectively. One caveat is that to split any weight value, an entire row must be added to the weight matrix. For a conv layer, OCS requires duplicating an entire 2D activation channel and all 2D weight filters connected to that channel. A second caveat is that not all values need to be split. At the bottom of Figure 2(b), w_4 is split in half while w_3 is not split.

3.3. Quantization-Aware Splitting

In this section, we show that duplicating a value and dividing it by two (i.e. `Net2WiderNet`) increases the total quantization error. We then propose an alternative split ratio which preserves quantization error. Without loss of generality, consider the deterministic rounding function $Q(x) = \lfloor x + \frac{1}{2} \rfloor$, which maps each real number to its closest integer, rounding halves toward $-\infty$. The maximum quantization error introduced by $Q(x)$ is 0.5. Next define OCS as a function $f(w) : \mathbb{R} \rightarrow \mathbb{R}^2$ which maps a single value to two values. The naïve split used in `Net2WiderNet` is:

$$\text{OCS}_{\text{naive}}(w) = \begin{pmatrix} w/2 \\ w/2 \end{pmatrix} \tag{5}$$

It is clear that $Q(w) \neq Q(\frac{w}{2}) + Q(\frac{w}{2})$, i.e. naïve OCS does not preserve the quantized value. The maximum total quantization error is doubled as both halves may be rounded in the same direction (e.g., $w = 3$ and each half is 1.5).

To address this, we propose the following *quantization-aware* (QA) splitting function:

$$\text{OCS}_{\text{QA}}(w) = \begin{pmatrix} (w - 0.5)/2 \\ (w + 0.5)/2 \end{pmatrix} \tag{6}$$

Intuitively, this forces $Q(x)$ to round in different directions when w is close to the midpoint between grid points. More formally, we can prove the following:

$$\begin{aligned} & Q(\frac{w - 0.5}{2}) + Q(\frac{w + 0.5}{2}) \\ &= \left\lfloor \frac{w - 0.5}{2} + \frac{1}{2} \right\rfloor + \left\lfloor \frac{w + 0.5}{2} + \frac{1}{2} \right\rfloor \\ &= \left\lfloor \frac{w + 0.5}{2} \right\rfloor + \left\lfloor \frac{w + 0.5}{2} + \frac{1}{2} \right\rfloor \\ &= \lfloor w + 0.5 \rfloor \end{aligned} \tag{7}$$

The last line is simply $Q(w)$, showing that QA OCS preserves the original quantization result. To derive the last line, we apply Hermite's Identity (Savchev & Andreescu, 2003) with $n = 2$:

$$\sum_{k=0}^{n-1} \left\lfloor x + \frac{k}{n} \right\rfloor = \lfloor nx \rfloor \tag{8}$$

We can further show that QA splitting is optimal, i.e. there exists no way to split w which results in lower quantization error. This proof is omitted due to length.

3.4. Channel Selection

As stated earlier, OCS cannot target individual weights or activations and must duplicate entire channels. OCS performs splits one at a time, and always splits the channel containing the largest absolute value in the layer. By prioritizing channels containing the largest values, OCS seeks to minimize distortion caused by any subsequent clipping.

We use a simple method to determine how many splits to perform in each layer (i.e. how many extra channels are created). For a layer containing C channels, OCS splits ceil$(r * C)$ channels, where r is the *expansion ratio*, a hyperparameter that determines approximately the level of

tolerable overhead in the network. This method allocates extra channels without considering each layer's weight or activation distributions. We also tried a more intelligent approach which formulates extra channel allocation as a knapsack problem. The reward function is the percentage reduction in the dynamic range of the distribution, and the cost is the increase in memory size. We optimize the number of extra channels for all layers simultaneously subject to a constraint on the memory overhead. Unfortunately, the knapsack approach is experimentally not better than the simple method described above, and for space reasons we do not show results with knapsack.

Channel selection on DNN weights is straightforward to implement as the weights are known and fixed post-training. For the activations, we take an approach similar to TensorRT (Migacz, 2017): we use a small number of training images to sample the activations in each layer. The sampled distributions in each layer are then used for OCS.

3.5. Implementation on Commodity Hardware

A key strength of OCS is simplicity, allowing it to be used in practical scenarios with either commodity hardware or emerging deep learning accelerators. Figure 2(b) shows the network modifications needed to implement OCS — we need to duplicate and possibly scale certain channels in the weights and activations. The weight modifications can be done off-line prior to serving the model. For the activations, a custom layer can be inserted which simply copies and scales the appropriate channels.

4. Clipping

Clipping represents the state-of-the-art in post-training quantization. This section gives a brief overview of different methods for optimizing the clip threshold in literature; we present an evaluation of these methods in Section 5.

4.1. Mean Squared Error

This method chooses a clip threshold which minimizes the mean squared error (MSE) or L2-norm between the floating-point and quantized values (Sung et al., 2015; Shin et al., 2016). It first constructs a histogram of the floating-point values. Let x_i and $h(x_i)$ be the bin values and frequencies, and $i = 1 \dots n$ denote the n bins. The MSE is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} h(x_i) * (x_i - Q(x_i))^2 \qquad (9)$$

where $Q(x)$ is the quantization function. In our experiments, we generate a large number of candidate clip thresholds evenly spaced between 0 and the max absolute value, and choose the one with minimal MSE.

Table 1. **Quantization-aware (QA) splitting in OCS –** each table entry is formatted as (**QA / non-QA**) where the non-QA split is simply dividing by two. The model is ResNet-20 for CIFAR-10.

Wt. Bits	OCS Expand Ratio			
	0.01	**0.05**	**0.1**	**0.2**
6	**92.0** / 91.9	91.9 / **92.0**	**92.1** / 91.9	92.0 / 92.0
5	**91.6** / 91.4	**91.7** / 91.6	**92.0** / 91.8	91.7 / **91.8**
4	88.0 / **88.3**	88.2 / **88.3**	**88.7** / 86.8	**89.1** / 86.8
3	**49.9** / 44.5	**58.3** / 44.8	**62.7** / 44.6	**76.5** / 52.8

4.2. ACIQ

Proposed by (Banner et al., 2018), ACIQ first determines whether distribution is closer to a Gaussian or a Laplacian. Using statistics from the appropriate distribution, it uses an (approximate) closed-form solution for the clip threshold which minimizes MSE. Compared to the MSE method above, ACIQ avoids sweeping candidate thresholds and is much faster — this allows the clip threshold to be adjusted between input batches for activation quantization.

We used open-source code from the authors [2]. Banner *et al.* assumed that an m-bit fixed-point format contains 2^m grid points; this representation lacks a grid point at zero for signed values. We use $2^m - 1$ grid points instead (i.e. sign-magnitude) as it is the default in our framework, and slightly adjusted the formulas from the paper to suit.

4.3. KL Divergence

This method chooses a clip threshold which (approximately) minimizes the KL divergence between the floating-point and quantized. Similar to the MMSE method, it works on the histogram of values and selects the optimal clip threshold from a set of candidates. The method was first proposed in a set of slides on NVIDIA's TensorRT (Migacz, 2017), which unfortunately does not contain enough technical detail for replication. Instead, we adapted an open-source implementation from Apache MXNet (Chen et al., 2015).

In general, floating-point and quantized distributions do not have the same support and the KL divergence is thus undefined. To get around this, the MXNet implementation smooths the quantized histogram slightly by moving some of the probability mass into zero-frequency bins.

5. Experimental Evaluation on CNNs

This section reports experiments on CNN models for ImageNet classification (Deng et al., 2009) conducted using PyTorch (Paszke et al., 2017) and Intel's open-source Distiller [3] quantization library. Post-training quantization was per-

[2] https://github.com/submission2019/AnalyticalScaleForIntegerQuantization

[3] https://github.com/NervanaSystems/distiller

Table 2. **ImageNet Top-1 validation accuracy with weight quantization –** the float accuracy is displayed under the model name. Results include different **Clip** methods, **OCS** with different expand ratios, and **OCS + the Best Clip method** at each bitwidth. For clipping, the best result is bolded and copied to the **Clip - Best** column. For OCS, the smallest expand ratio that outperforms all clipping methods is bolded, and the smallest expand ratio that achieves +1% accuracy over clipping is highlighted in blue. Best viewed in color.

Network	Wt Bits	Clip				Clip Best	OCS			OCS + Best Clip		
		None	MSE	ACIQ	KL		0.01	0.02	0.05	0.01	0.02	0.05
VGG-16 BN (73.4)	8	**73.0**	72.6	72.8	68.4	73.0	72.6	72.9	72.8	72.7	72.8	72.5
	7	**72.8**	72.5	72.5	60.7	72.8	72.1	**72.8**	72.5	72.4	72.1	72.6
	6	70.8	**71.3**	71.2	63.2	71.3	72.3	72.2	72.3	**71.8**	71.8	72.1
	5	63.1	**66.9**	61.2	62.7	66.9	69.3	70.2	71.0	68.8	69.5	70.0
	4	0.2	53.5	34.2	**59.4**	59.4	10.4	26.3	37.9	63.8	63.8	65.9
ResNet-50 (76.1)	8	75.4	**75.5**	75.4	73.5	75.5	**75.7**	75.7	75.7	**75.7**	75.7	75.4
	7	75.0	**75.2**	75.0	72.8	75.2	**75.5**	75.5	75.6	**75.5**	75.5	75.5
	6	72.9	73.5	**74.3**	71.6	74.3	**74.9**	74.7	75.0	**74.8**	74.8	75.2
	5	14.5	69.1	**69.9**	69.4	69.9	69.4	71.9	72.6	71.0	71.9	73.4
	4	0.1	45.0	33.2	**62.9**	62.9	12.1	36.1	55.2	66.2	67.1	69.3
DenseNet-121 (74.4)	8	**74.1**	73.8	73.7	71.0	74.1	**74.2**	74.2	74.2	**74.2**	74.2	74.2
	7	**73.8**	73.3	73.1	62.3	73.8	**73.9**	74.0	74.0	**74.1**	74.2	74.1
	6	71.0	**71.4**	71.1	60.7	71.4	72.9	73.0	73.2	73.2	73.1	73.1
	5	46.9	**65.4**	61.4	54.6	65.4	**65.5**	69.7	71.3	70.0	70.7	71.6
	4	0.4	33.3	25.2	**42.6**	42.6	13.1	37.5	53.0	52.7	56.5	63.0
Inception-V3 (75.9)	8	**74.8**	74.6	74.0	72.6	74.8	**75.2**	75.4	75.3	74.8	**75.0**	74.9
	7	**73.2**	71.2	69.1	69.4	73.2	74.8	74.7	74.7	71.8	**73.8**	74.2
	6	58.3	**66.2**	62.3	63.0	66.2	71.3	71.8	72.1	70.5	71.7	72.5
	5	0.5	30.4	29.6	**40.5**	40.5	45.2	54.0	60.2	57.0	60.0	62.9
	4	0.1	0.2	0.1	**1.6**	1.6	0.1	0.2	0.6	**2.1**	2.3	4.8

formed using Distiller's symmetric linear quantizer, which scales the quantization grid based on the maximum absolute value following Equation 1. For activation quantization, we first sampled the activation distributions using 512 *training* images (i.e. images not part of the validation/test set) to determine the quantization grid points, then use this grid during testing. This profiling took between 40 and 200 seconds on our machine using an NVIDIA GTX 1080 Ti. Weight clipping and OCS does not require profiling and was performed without any input data.

The chosen CNN benchmarks are four popular ImageNet classification models: VGG16 (Simonyan & Zisserman, 2015) with batch normalization added, ResNet-50 (He et al., 2015), DenseNet-121 (Huang et al., 2017), and Inception-V3 (Szegedy et al., 2015). Pre-trained weights were obtained from the PyTorch model zoo and we ran inference only. The first layer was not quantized as it generally requires more bits than the others, and contains only 3 input channels meaning OCS would incur a large overhead.

5.1. Effect of Quantization-Aware Splitting

The first experiment compares our proposed quantization-aware (QA) splitting (Section 3.3) against simply dividing by two as per Net2Net. Table 1 displays results from ResNet-20 for CIFAR-10 (Krizhevsky & Hinton, 2009). Although the difference is negligible until 4 bits (at which point there is significant accuracy degradation), QA splitting is clearly better than the naive method. This validates our mathematical ideas in Section 3.3, and we use QA splitting in all ensuing experiments.

5.2. Weight Quantization

Table 2 compares different clipping methods and OCS on weight quantization. The weights were quantized to 8-4 bits, while the activations were quantized to 8 bits. Floating-point accuracy is displayed under the model name. Linear quantization without clipping or OCS is shown in the **Clip - None** column. For ease of comparison we copy the best clipping result to the **Clip - Best** column. A range of small expand ratios r was chosen for OCS.

Our results indicate that for large bitwidths, there is no advantage to doing weight clipping. This is in line with (Migacz, 2017), which reported the same at 8 bits. Clipping becomes beneficial at 6 bits or fewer, improving accuracy by up to 55% for Resnet-50 and 40% for Inception-V3. Interestingly, the best-performing clipping technique depends on the bitwidth and appears to follow a consistent pattern across network architectures. As we go from high to low bitwidth, the winning technique goes from no clipping, to MSE/ACIQ, to KL at 4 bits.

Table 3. **ImageNet Top-1 validation accuracy with activation quantization –** formatting is identical to Table 2 except weight bits is kept at 8 while the activation bitwidth is changed. We did not combine OCS with clipping due to ineffectiveness of OCS on activations.

Network	Act. Bits	Clip				Clip	OCS		
		None	MSE	ACIQ	KL	Best	0.01	0.02	0.05
VGG16-BN (73.4)	8	72.5	**73.2**	73.1	**73.2**	73.2	72.7	72.8	72.5
	7	70.8	**72.8**	**72.8**	72.7	72.8	70.5	70.7	70.2
	6	49.0	71.3	**71.4**	70.6	71.4	49.2	46.0	45.9
	5	0.7	**62.0**	58.1	51.6	62.0	1.6	1.0	1.4
	4	0.1	**11.5**	5.0	2.4	11.5	0.1	0.2	0.1
ResNet-50 (76.1)	8	75.5	**75.9**	75.8	75.8	75.9	75.6	75.5	75.7
	7	**75.4**	75.3	75.2	75.3	75.4	74.1	74.5	74.1
	6	62.6	**73.5**	**73.5**	72.8	73.5	63.3	63.3	63.6
	5	5.7	63.7	**65.4**	56.7	65.4	10.0	12.6	6.0
	4	0.1	9.0	**20.6**	7.2	20.6	0.1	0.1	0.1
DenseNet-121 (74.4)	8	74.0	**74.1**	73.8	**74.1**	74.1	74.1	***74.2***	74.1
	7	73.0	**73.7**	72.9	73.7	73.6	73.2	73.2	73.0
	6	67.2	**72.6**	70.9	72.1	72.6	67.9	65.8	66.6
	5	19.9	**66.9**	64.6	64.5	66.9	16.0	18.7	13.2
	4	0.2	**26.9**	20.1	16.5	26.9	0.1	0.1	0.1
Inception-V3 (75.9)	8	74.8	**75.1**	73.4	75.0	75.1	74.8	74.9	74.8
	7	72.6	**74.2**	71.3	73.8	74.2	72.6	72.4	72.6
	6	51.6	**69.6**	60.7	67.9	69.6	54.1	51.5	48.5
	5	1.3	**34.2**	5.8	25.3	34.2	1.0	0.9	1.0
	4	0.1	**0.3**	0.1	0.2	0.3	0.1	0.1	0.2

Table 4. **ImageNet Top-1 accuracy for Oracle OCS on activations –** Oracle OCS splits a different set of channels for each input batch. Results use 6 activation bits and $r = 0.02$.

Batch Size	ResNet 50	Inception V3
1	74.6	71.7
2	74.5	71.7
4	74.0	71.6
8	74.1	70.9
32	73.5	70.7
128	73.3	70.3
No OCS	62.6	51.6
Clip Best	73.5	69.6

Weight OCS with an expansion ratio of only $r = 0.01$ outperforms our benchmarked clipping methods at 8-5 bits. At 8 and 7 bits the difference between OCS and clipping is small and there isn't a clear trend of improvement for higher expand ratios; in this regime OCS is not especially effective and the accuracy differences between expand ratios are mostly noise. At 6 and 5 bits, OCS with $r = 0.02$ outperforms clipping by 1% for all models except ResNet-50, and up to 13% for Inception-V3. This demonstrates that the basic idea of OCS works — by splitting the outliers to preserve their values instead of clipping them, OCS can improve the accuracy of post-training quantization. Another trend is that OCS gets most of its gains from small expansion ratios. The gain from $r = 0$ (no OCS) to $r = 0.01$ is always larger than the gain from moving to higher r values. Note that our benchmark networks have channel widths in the tens to hundreds so $r = 0.01$ equates to a single channel split in many layers. This again makes intuitive sense: the first channel split will target the unique largest outlier, guaranteeing a narrower weight distribution. Further splits target smaller values which occur with higher frequency, making OCS less effective at reducing the distribution width.

Given this intuition, we expect that a combination of OCS (to remove the largest outliers) followed by clipping (to further shrink the quantization grid) might surpass either method alone. The rightmost columns of Table 2 show results for OCS + Best Clip (i.e. applying OCS followed by the best performing clip method at each bitwidth). At 5 and 4 bits, OCS plus clipping cleanly outperforms OCS alone. OCS and clipping both seek to shrink the dynamic range of the quantized values, and thus there is some level of overlap between them. At high precision, OCS with our chosen expand ratios eliminates enough outliers such that additional clipping is unnecessary. At low precision (i.e. 4 bits), we believe that OCS would require huge expand ratios to fully address the outlier problem — in this regime OCS can combine with clipping to produce the best quantization results.

5.3. Activation Quantization

The same benchmarks and setup were used for activation quantization, except weights were kept at 8 bits while the bitwidth was varied for activations. To select the channels to split, we sampled activation distributions and counted the number of extreme values (we used values greater than the 99'th percentile) in each channel. Channels with the highest counts were split.

Table 3 shows activation quantization results. Unlike the weights, clipping is effective at all bitwidths tested. This is again in agreement with (Migacz, 2017), which applied clipping to 8-bit activations. MSE clipping outperforms the other clip threshold techniques in nearly all cases. The gap between MSE and KL divergence is very small for large

Table 5. **Model size overhead for ResNet-50 with OCS** – the overhead is very close to the user-provided expand ratio.

ResNet-50	OCS Expand Ratio			
	0.01	**0.02**	**0.05**	**0.1**
Rel. Weight Size	1.01	1.02	1.05	1.1
Rel. Activation Size	1.02	1.03	1.06	1.11

bitwidths, but at fewer bits MSE is clearly better. ACIQ performs worse than the other two methods with the exception of ResNet-50, where it showed good performance.

Activation OCS provides some improvement over simple linear quantization, but performs worse than clipping. This is likely because OCS relies on being able to identify the exact channel containing the largest outlier. With activations, profiling can only indicate which channels are *likely* to contain outliers, the best channel to split varies from input to input. To test our explanation, we experiment with **Oracle OCS**, which is simply OCS with exact knowledge of the activations generated by the network during testing — Oracle OCS chooses different channels to split in each input batch. Table 4 displays the results for Oracle OCS with different batch size on two models with 6 bit activations. Even at batch size 32, the oracle can already match or surpass the best clipping result. Further reducing the batch size (allowing channel selection at a finer granularity) leads to even better accuracy. These results show that OCS and our channel selection strategy can be effective for activations. However, channel selection must be done *dynamically*, requiring additional run-time analysis which is difficult to implement and likely inefficient in commodity systems.

5.4. OCS Memory Overhead

Because OCS increases the input channels by a factor of r, rounded up, the expand ratio r is a lower bound for the model size overhead. Table 5 shows both weight and activation overhead for ResNet-50 with different values of r, show that the true overhead matches r very closely.

6. Experimental Evaluation on RNNs

This section reports experiments on an RNN model with two stacked LSTM layers for language modeling (Zaremba et al., 2014). The corpus is the WikiText-2 dataset (Merity et al., 2016) with a vocabulary of 33,278 words. Each LSTM layer has a hidden size of 650, and the dimension of the word embedding in the input layer is 650. As the CNN results have shown that activation OCS is not effective, we focused on OCS and clipping on the weights. Activations and the hidden state are kept in floating-point for this experiment.

Table 6 compares the effects of OCS combined with different clipping methods on weight quantization. Lower perplexity is better, and the baseline floating-point model achieves a perplexity of 95.1. The best result on each row (i.e. the best clipping method at each OCS expand ratio) is bolded. Clipping is not effective on this model — none of the clipping techniques achieve any perplexity improvement. OCS achieves a much better result. At 6 bits, OCS begins to outperform the baseline with $r = 0.05$. At 5 bits, OCS sees steady perplexity decrease with successively larger expand ratios, clearly outperforming the best clipping result past $r = 0.02$. This is strong evidence that OCS can effectively improve post-training quantization beyond what can be achieved via clipping.

Table 6. **WikiText-2 perplexity with quantized weight –** lower is better. The floating-point baseline achieves a perplexity of 95.1. The best performing clip method along each row is bolded.

Wt. Bits	Expand Ratio	Clip Method			
		None	MSE	ACIQ	KL
6	0.00	**94.5**	98.1	99.0	97.7
	0.01	**95.0**	97.9	99.0	97.7
	0.02	**94.6**	97.8	96.1	96.3
	0.05	**93.9**	96.6	96.1	95.9
5	0.00	**98.2**	99.4	100.8	98.8
	0.01	**97.3**	99.7	99.9	97.7
	0.02	**95.7**	98.9	99.2	97.0
	0.05	**95.1**	98.2	98.5	96.3

7. Conclusions and Future Work

We propose outlier channel splitting, a method to improve DNN quantization without retraining which can be applied on commodity hardware. OCS splits channels in a layer to reduce the magnitude of outliers. Unlike the existing clip-based methods, OCS introduces a new tradeoff by reducing quantization error at the cost of network size overhead. Experimental results demonstrate that OCS on weights outperforms state-of-the-art clipping techniques with minimal overhead on deep CNN and RNN benchmarks. At very low precision, OCS in conjunction with clipping outperforms either method alone. Because clipping is used in NVIDIA TensorRT — a commercial post-training quantization flow — we believe that OCS has potential applicability in real-life systems.

Future work includes a more in-depth study into different channel selection methods, as well as applying OCS quantization during training. Specifically, we believe that OCS can help shape weight distributions during training to obtain better results than training for quantization alone.

Acknowledgments

This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA. One of the Titan Xp GPUs used for this research was donated by NVIDIA.

References

Banner, R., Nahshan, Y., Hoffer, E., and Soudry, D. ACIQ: Analytical Clipping for Integer Quantization of Neural Networks. *arXiv e-print*, arXiv:1810.05723, Oct 2018.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv preprint*, arXiv:1512.01274, Dec 2015.

Chen, T., Goodfellow, I., and Shlens, J. Net2net: Accelerating Learning via Knowledge Transfer. *Int'l Conf. on Learning Representations (ICLR)*, May 2016.

Choi, J., Chuang, P. I.-J., Wang, Z., Venkataramani, S., Srinivasan, V., and Gopalakrishnan, K. Bridging the Accuracy Gap for 2-bit Quantized Neural Networks (QNN). *arXiv e-print*, arXiv:1807.06964, Jul 2018a.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I.-J., Srinivasan, V., and Gopalakrishnan, K. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *arXiv e-print*, arXiv:1805.0608, May 2018b.

Chung, E., Fowers, J., Ovtcharov, K., Papamichael, M., Caulfield, A., Massengill, T., Liu, M., Lo, D., Alkalay, S., Haselman, M., Abeydeera, M., Adams, L., Angepat, H., Boehn, C., Chiou, D., Firestein, O., Forin, A., Gatlin, K. S., Ghandi, M., Heil, S., Holohan, K., Husseini, A. E., Juhasz, T., Kagi, K., Kovvuri, R. K., Lanka, S., van Megen, F., Mukhortov, D., Patel, P., Perez, B., Rapsang, A. G., Reinhardt, S. K., Rouhani, B. D., Sapek, A., Seera, R., Shekar, S., Sridharan, B., Weisz, G., Woods, L., Xiao, P. Y., Zhang, D., Zhao, R., , and Burger, D. Serving DNNs in Real Time at Datacenter Scale with Project Brainwave . *IEEE Micro*, 38(2):8–20, 2018.

Courbariaux, M., Bengio, Y., and David, J.-P. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *Advances in Neural Information Processing Systems (NIPS)*, pp. 3123–3131, 2015.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Han, S., Mao, H., and Dally, W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *Int'l Conf. on Learning Representations (ICLR)*, Feb 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *arXiv e-print*, arXiv:1512.0338, Dec 2015.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1(2): 3, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *Journal of Machine Learning Research (JMLR)*, 18(187): 1–30, 2017.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2704–2713, Jun 2018.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. *Int'l Symp. on Computer Architecture (ISCA)*, pp. 1–12, 2017.

Krizhevsky, A. and Hinton, G. Learning Multiple Layers of Features from Tiny Images. *Tech report*, 2009.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

McKinstry, J. L., Esser, S. K., Appuswamy, R., Bablani, D., Arthur, J. V., Yildiz, I. B., and Modha, D. S. Discovering low-precision networks close to full-precision networks for efficient embedded inference. *arXiv preprint*, arXiv:1809.04191, Sep 2018.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer Sentinel Mixture Models. *arXiv preprint*, arXiv:1609.07843, Sep 2016.

Migacz, S. 8-bit Inference with TensorRT. *NVIDIA GPU Technology Conference*, May 2017.

Park, E., Kim, D., and Yoo, S. Energy-Efficient Neural Network Accelerator Based on Outlier-Aware Low-Precision Computation. *Int'l Symp. on Computer Architecture (ISCA)*, Jun 2018a.

Park, E., Yoo, S., and Vajda, P. Value-aware Quantization for Training and Inference of Neural Networks. *arXiv e-print*, arXiv:1804.07802, Apr 2018b.

Park, H. and Choi, K. Cell Division: Weight Bit-Width Reduction Technique for Convolutional Neural Network Hardware Accelerators. *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pp. 286–291, Jan 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic Differentiation in PyTorch. *Advances in Neural Information Processing Systems Workshops (NIPS-W)*, 2017.

Savchev, S. and Andreescu, T. *Mathematical Miniatures*, chapter 12. Hermite's Identity, pp. 41–44. Mathematical Association of America, 2003.

Settle, S. O., Bollavaram, M., D'Alberto, P., Delaye, E., Fernandez, O., Fraser, N., Ng, A., Sirasao, A., and Wu, M. Quantizing Convolutional Neural Networks for Low-Power High-Throughput Inference Engines. *arXiv preprint*, arXiv:1805.07941, May 2018.

Shin, S., Hwang, K., and Sung, W. Fixed-Point Performance Analysis of Recurrent Neural Networks. *Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 976–980, 2016.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-print*, arXiv:1409.15568, Apr 2015.

Sung, W., Shin, S., and Hwang, K. Resiliency of Deep Neural Networks Under Quantization. *arXiv preprint arXiv:1511.06488*, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going Deeper with Convolutions. *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Wu, S., Li, G., Chen, F., and Shi, L. Training and Inference with Integers in Deep Neural Networks. *Int'l Conf. on Learning Representations (ICLR)*, May 2018.

Xu, X., Ding, Y., Hu, S. X., Niemier, M., Cong, J., Hu, Y., and Shi, Y. Scaling for Edge Inference of Deep Neural Networks. *Nature Electronics*, 1(4):216, 2018.

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights. *arXiv preprint*, arXiv:1702.03044, 2017.

Zhuang, B., Shen, C., Tan, M., Liu, L., and Reid, I. Towards Effective Low-Bitwidth Convolutional Neural Networks. *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 7920–7928, Jun 2018.