Interspeech 2022
18-22 September 2022, Incheon, Korea

Sub-8-Bit Quantization Aware Training for 8-Bit Neural Network Accelerator with On-Device Speech Recognition

Kai Zhen[†] *Hieu Duy Nguyen*[†] *Raviteja Chinta*[⋆]
Nathan Susanj[†] *Athanasios Mouchtaris*[†] *Tariq Afzal*[⋆] *Ariya Rastrow*[†]

Alexa AI, Amazon, USA[†]
Hardware Compute Group, Amazon, USA[⋆]
{zhenk, hieng, ravitec}@amazon.com
{nsusanj, mouchta, tafzal, arastrow}@amazon.com

Abstract

We present a novel sub-8-bit quantization-aware training (S8BQAT) scheme for 8-bit neural network accelerators. Our method is inspired from Lloyd-Max compression theory with practical adaptations for a feasible computational overhead during training. With the quantization centroids derived from a 32-bit baseline, we augment training loss with a Multi-Regional Absolute Cosine (MRACos) regularizer that aggregates weights towards their nearest centroid, effectively acting as a pseudo compressor. Additionally, a periodically invoked hard compressor is introduced to improve the convergence rate by emulating runtime model weight quantization. We apply S8BQAT on speech recognition tasks using Recurrent Neural Network-Transducer (RNN-T) architecture. With S8BQAT, we are able to increase the model parameter size to reduce the word error rate by 4-16% relatively, while still improving latency by 5%.
Index Terms: on-device speech recognition, sub-8-bit quantization, INT8 neural network accelerator, Lloyd-Max quantizer

1. Introduction

Latency reduction without introducing noticeable accuracy degradation is critical to on-device automatic speech recognition (ASR) systems in various scenarios, such as in in-car units and on portable devices, where Internet connectivity can be intermittent. Ubiquitous as end-to-end ASR models are [1, 2, 3, 4], deploying them on edge can be challenging due to the strict limit of bandwidth and memory of edge devices [5, 6]. Hence, it is highly necessary to improve model efficiency, i.e. model memory size vs. model accuracy, through applying model optimization and compression techniques when running real-time ASR on-device.

Efficient neural network inference can be approached by various ways. The teacher-student training paradigm for knowledge distillation simplifies the network topology, although the student may fail to mimic the teacher's behavior if the model capacity is too low [7]. Furthermore, it usually requires additional network compression techniques prior to the hardware deployment [8]. Alternatively, enforcing model sparsity can significantly reduce memory footprint [9, 10]. However, one can only realize the inference speedup if the sparsity pattern matches the specific memory design of the hardware [11].

Neural network quantization can be effectively employed to compress 32-bit weights down to 8-bit, or activations to 5-bit [12, 13], via applying a simple post-training quantization step [14, 15] or a more involved QAT mechanism [16, 17]. Several quantization methods have been proposed to lower the bit-depth to 4 [18, 19, 20], or even 1 for proof-of-concept bit-wise neural networks [21]. Nevertheless, these methods can be heuristic which rely on an extensive hyper-parameter tuning to maintain the accuracy level. Furthermore, such sub-8-bit quantization methods require sub-8-bit operators on neural network accelerators (NNAs), which often have inferior performance compared to their 8-bit counterpart due to the reduced numerical accuracy. Consequently, sub-8-bit NNAs are less adopted and thus there is no real latency measurement for existing sub-8-bit approaches [16, 18, 20, 21]. The most prevalent type of NNAs are rather based on 8-bit arithmetic operators, i.e. addition/multiplication/etc, accepting 8-bit inputs and computing the outputs via bitwise operations. For the rest of the paper, we will refer to this type as INT8 NNA.

In this work, we propose a novel sub-8-bit quantization aware training (S8BQAT) that integrates with INT8-based runtime NNAs. It differs from our previous linear-QAT method [17] in twofold. Firstly, S8BQAT distills quantization centroids from a pre-trained 32-bit baseline via a mechanism derived from Lloyd-Max scalar quantization theory. We introduce Multi-Regional Absolute Cosine (MRACos) regularizer, which is INT8 compatible and computationally efficient. The MRACos regularizer penalizes off-the-centroid weights and aggregates them towards their nearest quantization centroids. Additionally, the MRACos regularizer is accompanied by a periodic compressor that assigns each model weight to that nearest quantization centroid, ensuring quantization convergence and therefore minimizing runtime quantization-induced performance degradation. Throughout the paper, we also refer to the MRACos regularizer and periodic compressors as soft and hard compressors, respectively. The soft compressor affects the gradient calculation only through the use of the regularization term, while the hard compressor quantizes each model weight to the exact centroid that are also used at runtime inference. We apply our S8BQAT into the ASR task and measure the performance in terms of word error rate (WER) and on-device runtime user perceived latency (UPL) on various runtime settings. Results show that the proposed S8BQAT achieves superior WER-UPL tradeoff compared to an 8-bit baseline. In particular, we increase the number of model parameters by 10.3%, thus reducing WER by 4-16% relative while reducing UPL by 5% with S8BQAT.

The rest of the paper is structured as follows. The proposed S8BQAT and the associated data loading mechanism are introduced in Sec.2. In Sec.3, we conduct runtime validation including the measurement of model accuracy and on-device UPL from general-purposed RNN-T models for speech recognition, along with an ablation study. Sec. 4 concludes the paper.

10.21437/Interspeech.2022-874

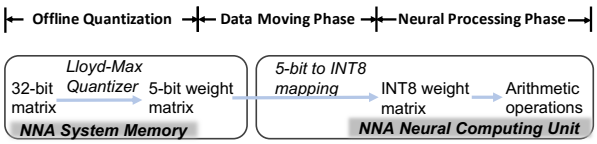


Figure 1: *NNA loads 32-bit model weights in the sub-8-bit form and decompresses them for integer arithmetic operations.*

2. Algorithm description

2.1. Sub-8-bit data loading mechanism in NNA

The proposed algorithm compresses deep learning models that are hosted on an NNA. At runtime, the NNA loads the model weights from the system memory into the neural computing unit's local memory buffer (data moving phase) to perform bit-wise arithmetic operations. This data moving phase needs to be accounted for each model inference call as often NNAs have limited on-chip memory to fully cache model weights locally.

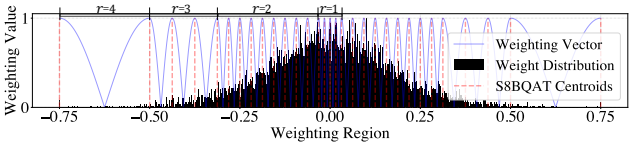
High efficiency can be achieved by accelerating matrix related operations, such as matrix multiplication on NNAs. Yet, the data moving phase is time consuming due to the constrained available memory bandwidth on-device. To reduce the bandwidth and consequently the latency, we quantize model weights into sub-8-bit (offline quantization), then transfer them to the NNA's neural computing unit where sub-8-bit weights are decompressed into INT8 format for the neural processing phase (see Figure 1). Consider a weight matrix with the shape of $(1024, 4096)$. With Lloyd-Max quantizer representing all weights by 32 distinct values or 5-bit, the compressed matrix requires 2.5MB, instead of 4MB for 8-bit, thus reducing the in-memory size and data transfer latency by 37.5%.

2.2. Lloyd-Max scalar quantization theory

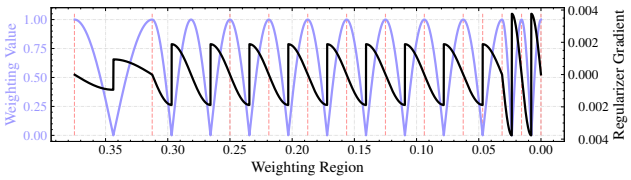
The problem of compressing a set of weights into another set with a smaller cardinality is solved by S. Lloyd and J. Max, which is often referred to as Lloyd-Max scalar quantization theorem [22, 23]. Let $\{c_1, ..., c_k\}$ be k partitions of the model weights $\boldsymbol{w} = \{w_1, ..., w_n\}$, and $\{m_1, ..., m_k\}$ be the prototypes (or quantization centroids) for the corresponding k partitions. Lloyd Max algorithm minimizes the mean squared error between the model weights and corresponding centroids: $\mathcal{H} = \sum_{k=1}^{K} \sum_{w_i \in c_k} ||w_i - m_k||^2$. The solutions of $\{m_1, ..., m_k\}$ can be derived in closed-form [22, 23], or via an iterative method which is also used for K-Means clustering [24]: for each iteration, we 1) assign each w_i to the nearest m_k: $c_k^{\text{new}} = \{w_i : \arg\min_{k'} ||w_i - m_{k'}||^2 = k\}$; and 2) update prototypes by setting $m_k^{\text{new}} = \frac{1}{|c_k^{\text{new}}|} \sum_{w_i \in c_k^{\text{new}}} w_i$, where both steps decrease $\mathcal{H}$ unless the algorithm has converged. With all weights $w \in c_i$ represented by $\{m_1, ..., m_k\}$, they are quantized into $\lceil \log_2 k \rceil$ bits.

2.3. Lloyd-Max quantizer-like regularization

Directly applying Lloyd-Max scalar quantizer to sub-8-bit QAT can be problematic. Firstly, the quantization centroids are not guaranteed to conform to INT8 format. Consequently, at runtime, when they are updated to $k/128$, where the integer $k \in [-128, 127]$, the model performance is subject to degradation. Secondly, executing the Lloyd-Max algorithm per training step is computationally expensive and memory consuming.



(a) Multi-regional absolute cosine regularizer with 31 peaks



(b) Gradient decays when the cosine frequency gets smaller

Figure 2: *Proposed soft compressor and its gradient*

To mitigate this issue, we propose multi-regional absolute cosine (MRACos) regularizer

$$\mathcal{L}_{\text{MRACos-reg}}(\boldsymbol{w}) = \sum_{w_i \in \boldsymbol{w}} \sum_{r=1}^{R} \delta_r \lambda_r (1 - |cos(\pi \theta_r w_i)|), \quad (1)$$

where $|cos(\pi\theta_r w_i)|$, λ_r and θ_r define the weighting vector, regularization weight coefficient and frequency of the cosine function in region r, while $\delta_r(w_i)$=1 for all $w_i \in \boldsymbol{w}$ that are inside the range of the r-th region, and $\delta_r(w_i)$=0 otherwise. It approximates the Lloyd-Max quantization centroids per region, e.g. r=3 in Figure 2 (a). Note that the frequency of the cosine function for all regions in Eq.1 is chosen such that regularizer maxima have INT8 format $k/128$, in which $k \in [-128, 127]$, and closest to the Lloyd-Max quantizer. The weighting vector penalizes model weights by how far they are off the nearest centroid: as shown in Eq.1, the further the model weight is off, the larger penalty is applied; the weight receives no penalty when it's on the centroid, as it leads to no extra degradation when being quantized. Weights outside all regions are clipped.

The gradient of MRACos regularizer can be calculated as,

$$\frac{\partial \mathcal{L}_{\text{MRACos-reg}}}{\partial w_i} = \sum_{r=1}^{R} \theta_r \lambda_r (-1)^{\xi} \pi \delta_r sin(\pi \theta_r w_i), \quad (2)$$

in which ξ=0 if $w \in [\frac{t-2}{\theta_r}, \frac{t}{\theta_r}]$ with t=$\{\cdots, -3, -1, 1, 3, \cdots\}$ and ξ=1, otherwise. Note that the cosine frequency θ_r in Eq. 2 becomes a decay factor in the gradient of the regularizer. For a region with a relatively small cosine frequency, the gradient there will be comparably small (see Figure 2 (b)). Admittedly, with a relatively large λ_r, weights will be more aggressively aggregated toward quantization centroids for faster quantization convergence. However, setting a high regularization weight will negatively affect the model performance, as discussed in Sec.3.

2.4. Periodic hard compressor

Instead of increasing the regularization weight to address the gradient decay issue, we introduce a periodic hard compressor that performs runtime quantization during model training per τ epochs (Figure 3). We measure the quantization convergence rate of weights for the k-th partition c_k, and its centroid m_k as

$$\gamma_k = \frac{|||w_i - m_k||^2 < \epsilon|}{|w_i \in c_k|}, \quad (3)$$

where the threshold ϵ is relatively small. As discussed in Sec.3, γ_k can be boosted effectively with the hard compressor to reduce the quantization induced degradation at runtime.

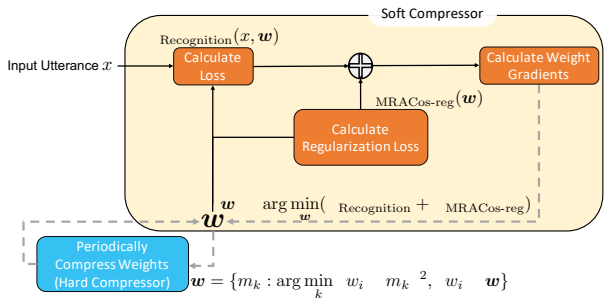


Figure 3: *Data flow during training with S8BQAT: the soft compressor operates at the gradient level while the hard compressor mimics runtime/post-training quantization.*

Table 1: *Architecture for M-II and M-III. S8BQAT can be enabled layer-wise: layers of M-II in gray are with 5-bit S8BQAT while others in 8-bit. M-III applies 5-bit S8BQAT to all layers.*

	Layer	Kernel	Recurrent Kernel	#Params
Encoder	L0	(192, 4096)	(1024, 4096)	47.8
	L1	(1024, 4096)	(1024, 4096)	
	L2	(2048, 4480)	(1120, 4480)	
	L3, L4	(1120, 4480)	(1120, 4480)	
	Projection	(1120, 512)	–	
Decoder	L0	(512, 4352)	(1088, 4352)	17.0
	L1	(1088, 4352)	(1088, 4352)	
	Projection	(1088, 512)	–	
Jointer	L0	(512, 2501)	–	2.6
	Decoder Embedding	(2501, 512)	–	

3. Experiment

3.1. Experimental setup

To validate the effectiveness of the proposed S8BQAT method, we apply it into the ASR task using the Recurrent Neural Network - Transducer (RNN-T) [25] architecture and measure two performance metrics: the word error rate (WER) for accuracy and user perceived latency (UPL). We consider several RNN-T variants, and compare the performance against a linear quantization-aware training baseline method [17]. Essentially, the proposed method differs from its baseline in two aspects: the soft compressor approximates Lloyd-Max scalar quantization to regularize model weights in a non-linear space while the hard compressor is introduced and invoked periodically to emulate runtime quantization.

We consider 3 RNN-T model settings, all with 5 encoding layers, 2 decoding layers, 1 joint layer and 2.5k word pieces: M-I is the baseline including 61.0M parameters, and trained with the 8-bit linear QAT mechanism [17]. M-II and M-III adopt the topology of M-I but with a slightly larger number of hidden units: specifically, the number of hidden units are increased from 1024 to 1120 and 1088 for Encoder L2-L4/Projection layer and Decoder, respectively (see Table 1). Consequently, the number of parameters for M-II and M-III is 67.3M, or 10.3% larger than M-I. Under S8BQAT, the process is as follows. We first train M-II and M-III for a pre-determined duration, e.g. 10k out of 850k steps; then extract the weight distribution and sub-8-bit configurations for all layers of M-III, and all but L0 layers of M-II; then apply the proposed S8BQAT mechanism (see Figure 3) to train M-II and M-III for the rest of the epochs; and finally, compress all sub-8-bit layers of M-II and M-III after the training finishes.

M-I, II and III are trained with a de-identified far-field dataset including 100k hours of human transcribed utterances and 40k hours of utterances generated by self-supervised learning speech model. All models are trained for 850k steps. We consider 4 far-field test sets for the WER evaluation: Frequent, Appliances, Entertainment, Rare, each with 50K utterances. Frequent and Rare datasets include tasks that are frequently or rarely queried, while Appliances and Entertainment contain queries from specific supported Appliances and Entertainment tasks. We benchmark latency metrics on 6k test utterances.

For reproducible purposes, we also train RNN-T models with S8BQAT on LibriSpeech corpus [26] with 960k hours of training data for 120k steps, and measure WERs with 5.4k, 5.3k, 5.4k, and 5.1k hours of dev-clean, dev-other, test-clean and test-other data, respectively.

3.2. Experimental results

3.2.1. Accuracy and latency analysis

We compare the accuracy and latency performance among M-I, M-II and M-III in Table 2. We normalize the WER numbers by the WER of M-I for Frequent test set. The latency numbers are normalized by the latency of M-I for latency test set. The latency is observed to be reduced even when the number of parameters increases thanks to the employment of S8BQAT: concretely, compared to the 8-bit baseline (M-I), M-II lowers the UPL-P50 by 5% and UPL-P90 by 3%; a more aggressive quantization configuration in M-III further brings down the latency with 6% UPL-P50 reduction and 5% UPL-P90 reduction, respectively. Moreover, M-II and M-III consistently show the accuracy improvement over the 8-bit baseline from all 4 test sets, thanks to their slightly increased number of parameters.

The mechanism of S8BQAT is depicted in Figure 4, where we extract 4 snapshots of M-II during training and plot their weight distributions. With the MRACos regularizer, weights are effectively driven to quantization centroids even with only 50k training steps (Figure 4 (a)). However, peaks in the tail of the distribution (grey circle in Figure 4 (a)) are far less spiky (indicating a lower quantization convergence rate, γ). This is mitigated at step 500k (see Figure 4 (c)), as all 5-bit regularized weights are hard quantized via the periodic hard compressor at step 350k (see Figure 4 (b)). Had the hard compressor not been invoked, the convergence rate at the tail of the weight distribution would have still been low, due to the decayed gradient issue as discussed in Sec.2. On the other hand, utilizing only the hard compressor results in large performance fluctuation and fundamentally worse WERs [17]. Thanks to the quantization-aware training with MRACos regularizer, the weights are gradually pushed towards the quantization centroids, leading to smaller fluctuations in the validation loss (see Figure 4 (d)).

3.2.2. Ablation analysis

We also conduct the ablation analysis (Table 3) by alternating the regularization weight (λ) for the soft regularizer and the hard compression period (τ). The baseline models have M-I architecture and linear-QAT with 8, 5, and 4-bit weights, i.e. the model is QAT trained and post-training quantized with x bits in which x=$\{8, 5, 4\}$. For S8BQAT models, we consider 4-bit and 5-bit, training/compressing and post-training quantizing all layers. All models are trained on LibriSpeech dataset with all other training hyperparameters being the same.

As shown in Table 3, the linear-QAT method [17] performs well in 8-bit, but fails to preserve the accuracy level in sub-8-bit modes. In contrast, the proposed soft compressor leverages