SUB-8-BIT QUANTIZATION FOR ON-DEVICE SPEECH RECOGNITION: A REGULARIZATION-FREE APPROACH

Kai Zhen, Martin Radfar, Hieu Nguyen, Grant P. Strimel, Nathan Susanj, Athanasios Mouchtaris

Amazon Alexa AI

ABSTRACT

For on-device automatic speech recognition (ASR), quantization aware training (QAT) is ubiquitous to achieve the trade-off between model predictive performance and efficiency. Among existing QAT methods, one major drawback is that the quantization centroids have to be predetermined and fixed. To overcome this limitation, we introduce a regularization-free, "soft-to-hard" compression mechanism with self-adjustable centroids in a μ-Law constrained space, resulting in a simpler yet more versatile quantization scheme, called General Quantizer (GQ). We apply GQ to ASR tasks using Recurrent Neural Network Transducer (RNN-T) and Conformer architectures on both LibriSpeech and de-identified far-field datasets. Without accuracy degradation, GQ can compress both RNN-T and Conformer into sub-8-bit, and for some RNN-T layers, to 1-bit for fast and accurate inference. We observe a 30.73% memory footprint saving and 31.75% user-perceived latency reduction compared to 8-bit QAT via physical device benchmarking.

Index Terms— On-device speech recognition, quantization aware training, RNN-T, conformer, model efficiency

1. INTRODUCTION

Improving the efficiency of neural automatic speech recognition (ASR) models via quantization is critical for on-device deployment scenarios. For neural network accelerator (NNA) embedded devices, where memory and bandwidth are at a premium, quantization can reduce the footprint and lower the bandwidth consumption of ASR execution, which will not only afford a faster model inference but also facilitate model deployment to various portable devices where a stable network connection is limited.

Existing quantization methods can be post-training quantization (PTQ) or in-training / quantization aware training (QAT). PTQ is applied after the model training is complete by compressing models into 8-bit representations and is relatively well supported by various libraries [1, 2, 3, 4, 5, 6], such as TensorFlow Lite [7] and AIMET [8] for on-device deployment. However, almost no existing PTQ supports customized quantization configurations to compress machine learning (ML) layers and kernels into sub-8-bit (S8B) regimes [9]. Moreover, the performance drop is inevitable as the model is unaware of the loss of precision when being quantized at test time. In contrast, QAT performs bit-depth reduction of model weights (for example, from 32-bit floating point to 8-bit integer) during training which usually yields superior performance over PTQ [10][11]. The QAT mechanism can be in the forward pass (FP-QAT) or the backward pass (BP-QAT), with the difference being whether regularization is used in the loss function. FP-QAT [9] quantizes the model weights during forward propagation to pre-defined quantization centroids. BP-QAT [12, 13, 14] relies on customized regularizers to gradually force weights to those quantization centroids (i.e., "soft quantization" via gradient) during training before hard compression performs in the late training phase. As model weights are informed by the customized regularizers to move closer to where they are quantized at runtime per training step, the predictive performance is often well preserved. Therefore, the focus of this work is on QAT.

Under both FP- and BP-QAT, it is essential that the quantization centroids are defined and specified before model training. As such, the demerit is the low feasibility when quantizing models in S8B mode because one needs to select the proper quantization centroids and their configurations for each kernel in each layer to ensure minimal runtime performance degradation. Consequently, applying existing QAT methods to Conformer [15] becomes quite challenging, as it usually contains more than hundreds of kernels.

In this work, we propose General Quantizer (GQ), a regularization-free, model-agnostic quantization scheme with a mixed flavor of both FP- and BP-QAT. GQ is "general" in that it does not augment the objective function by introducing any regularizer as in BP-QAT but determines the appropriate quantization centroids during model training for a given bit depth, and it can be simply applied in a plug-and-play manner to an arbitrary ASR model. Unlike FP-QAT, GQ features a soft-to-hard quantization during training, allowing model weights to hop around adjacent partitions more easily. Under GQ, quantization centroids are self-adjustable but in a μ-Law constrained space. As a proof-of-concept, we adopt the ASR task and conduct experiments on both the LibriSpeech and de-identified far-field datasets to evaluate GQ on three major end-to-end ASR architectures, namely conventional Recurrent Neural Network Transducer (RNN-

978-1-6654-7189-3/22/$31.00 ©2023 IEEE

T) [16], Bifocal RNN-T [17], and Conformer [18][19]. Our results show that in all three architectures, GQ yields little to no accuracy loss when compressing models to S8B or even sub-5-bit (5-bit or lower). We also present performance optimization strategies from ablation studies on bit-allocation and quantization frequency. Our contributions are as follows:

- We propose GQ, inspired by both FP- and BP-QAT approaches. GQ enables on-centroid weight aggregation without augmented regularizers. Instead, it leverages Softmax annealing to impose soft-to-hard quantization on centroids from the μ-Law constrained space.
- GQ supports different quantization modes for a wide range of granularity: different bit depths can be specified for different kernels/layers/modules.
- With GQ, we losslessly compress a lightweight streaming Conformer into sub-5-bit with more than 6× model size reduction. To our best knowledge, this is among the first sub-5-bit Conformer models for on-device ASR. Without accuracy degradation, our GQ-compressed 5-bit Bifocal RNN-T reduces the memory footprint by 30.73% and P90 user-perceived latency (UPL) by 31.30%.

We describe the problem in Sec. 2 and GQ in Sec. 3. The experimental settings and results are detailed in Sec. 4. We conclude in Sec. 5 with some final remarks.

2. PRELIMINARIES

2.1. Problem Formulation

Consider a general deep neural network architecture with K layers, $\mathcal{F} = \mathcal{F}_1 \circ \cdots \circ \mathcal{F}_K$, mapping the input from $\mathbb{R}^{d_1}$ to the output in $\mathbb{R}^{d_{K+1}}$ as $\mathcal{F} : \mathbb{R}^{d_1} \longmapsto \mathbb{R}^{d_{K+1}}$, where the input and output of an arbitrary k-th layer are $\boldsymbol{x}^{(k+1)} := \mathcal{F}_k(\boldsymbol{x}^{(k)})$. Under supervised learning, the training data $\mathcal{X} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x}_N\}$ and $\mathcal{Y} = \{\boldsymbol{y_1}, \ldots, \boldsymbol{y}_N\}$ are used for updating model weights $\mathbb{W} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_K]$ for K layers in $\mathcal{F}$. Usually the optimization process is over the training objective function $\mathcal{L}(\mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathbb{W}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\mathcal{F}(\boldsymbol{x_i}), \boldsymbol{y_i}) + \lambda R(\mathbb{W})$, where i is the data batch index, ℓ is the major loss term measuring model accuracy and $R(\mathbb{W})$ is the regularizer blended to the objective function via a coefficient λ.

Network quantization aims at discretizing model weights. For scalar quantization, it is to convert each weight, $w \in \mathbb{W}$, to a quantization centroid, $z \in \boldsymbol{z}$, where $\boldsymbol{z} = [z_1, ...z_m]$ to ensure the network is compressed into $\lceil log_2 m \rceil$-bit. For S8B quantization, centroids are from a subset of `INT8` values.

2.2. Related QAT Approaches

BP-QAT counters model weight continuity via regularization. For example, it introduces weight regularizers on model weights, i.e., $R(\mathbb{W})$, measuring the point-wise distance between each weight and m quantization centroids in the centroid vector $\boldsymbol{z} = [z_1, ...z_m]$. Note that the quantization weight regularizer in the loss function, as $R(\mathbb{W})$, must be gradient descent compatible. Consequently, $R(\mathbb{W})$ cannot enforce each weight to be replaced by the closest centroid in $\boldsymbol{z}$ as $w = \arg\min_i ||w - z_i||$, for $w \in \mathbb{W}$ and $z_i \in \boldsymbol{z}$, because the $\min$ operator is not differentiable. Recent BP-QAT methods force weights to approach the centroid in $\boldsymbol{z}$ using $R(\mathbb{W}) = \sum_{w \in \mathbb{W}} \mathcal{D}(w, \boldsymbol{z})$, where the differentiable dissimilarity function $\mathcal{D}$ is based on a cosine function in [12, 13].

In contrast, FP-QAT can be regularizer free [9, 20]. Usually, the process is to use a "fake quantizer" or equivalent operations during training, hard quantizing weights to a specific range and bit-depth; and then at runtime, converting the model to `INT8` format via TFLite [21]. The study [9] uses native quantization operators with which, during training, the weights are quantized and then converted to the integer type for model deployment. However, FP-QAT is essentially hard compression recurring during training with severely dropped performance when applied to S8B quantization. Consequently, finetuning is usually needed, which prolongs the model training time [22, 23].

Both FP-QAT and BP-QAT require specifying appropriate quantization centroids before model training. While the centroids for `INT8` model compression are pre-defined, for S8B quantization, the optimal set of centroids is usually kernel/layer specific. For models, such as Conformer [15], where there are usually hundreds of kernels, current S8B QAT methods become less tractable.

In this work, we combine the merit from both FP- and BP-QAT and propose General Quantizer (GQ) that navigates weights to the corresponding quantization centroids without introducing augmented regularizers but via feedforward-only operators. Our work is inspired by a continuous relaxation of quantization [24] also used for speech representation learning [25, 26, 27, 28, 29], and μ-Law algorithm for 8-bit pulse-code modulation (PCM) digital telecommunication [30].

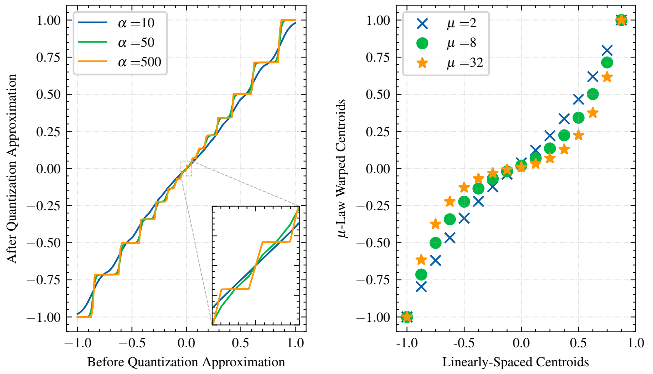
3. METHODS

3.1. Centroid Selection via Softmax-Based Dissimilarity Matrices

For any weight value $w_i \in \boldsymbol{w}$ where $|\boldsymbol{w}| = n$, and the quantization centroid vector $\boldsymbol{z} = [z_1, ...z_m]$, we define the point-wise dissimilarity matrix in Eq. 1

$$\boldsymbol{A}_{\text{soft}} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}, \tag{1}$$

where a_{ij} is the probability of representing w_i by z_j. Each row in $\boldsymbol{A}_{\text{soft}}[i\cdot]$ is summed to 1 with the largest probability



(a) Weight value before and after quantization approximation. Here, the larger the softmax temperature α is, the closer to true quantization that the weight transformation becomes.

(b) Relationship between the wrapped quantization centroids and μ. A large μ means more quantization centroids are allocated near 0 and vice versa.

Fig. 1: Hyperparameters in GQ: α adjusts the quantization temperature which increases gradually during training, and μ modulates the non-linearity of quantization centroids.

going to the closest centroid. This is achieved when the pointwise distance $||w_i - z_j||_1$ is scaled by a negative number $-\alpha$ and wrapped by a Softmax function in Eq. 2

$$a_{ij} = \frac{e^{-\alpha||w_i - z_j||_1}}{\sum\limits_{j=1}^{m} e^{-\alpha||w_i - z_j||_1}}. \tag{2}$$

Here, $\alpha \in [1, \infty)$ serves as the Softmax temperature for quantization annealing. When α is relatively small, w_i will be approximated by all centroids in $\boldsymbol{z}$ (see Eq.3); when $\alpha \to \infty$, $\boldsymbol{A}_{\text{soft}}[i\cdot]$ becomes a one-hot vector $\boldsymbol{A}_{\text{hard}}[i\cdot]$ and the weight will be the closest centroid.

$$\overline{w_i} = \boldsymbol{z} \times \boldsymbol{A}_{\text{soft}}[i\cdot]^T. \tag{3}$$

For simplicity, during training, we set the initial and target scalar values to be α_{start} and α_{end}, and allow α to gradually and linearly increase from s_{start} to s_{end}, as shown in Eq. 4.

$$\alpha = \alpha_{\text{start}} + (s - s_{\text{start}}) \times \frac{\alpha_{\text{end}} - \alpha_{\text{start}}}{s_{\text{end}} - s_{\text{start}}}. \tag{4}$$

As a result, the QAT effect is gradually intensified. At $\alpha = 10$, a rather small value, weights after being approximated by quantization centroids in $\boldsymbol{z}$ roughly preserve their original values; however, as α gradually increased to 500, the near-linear line almost becomes a step function, aggregating weights to just a few centroids (see Fig. 1 (a)). This forms a soft-to-hard QAT and allows model weights to be updated via gradients with barely any extra constraint during the early stage of training before driving weights to a certain centroid.

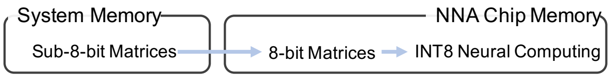


Fig. 2: Loading weights from system memory to chip memory on NNA is faster in S8B format, although the arithmetic operation is still in `INT8` format.

3.2. Adjusting Centroids with μ-Law Expanding

We assume weight distribution symmetry from any kernel in a trained 32-bit neural network in which the absolute values of most weights are small. Consequently, the imposed m quantization centroids in $\boldsymbol{z}$ should also be symmetric ($|z_i| = |z_{m+1-i}|$ where $i \in [1, m]$) with most centroids close to 0. To specify and adjust the level of non-linearity of $\boldsymbol{z}$ per kernel during training, we resort to μ-Law algorithm, mainly used in 8-bit PCM telecommunication (similar to A-Law algorithm standardized in Europe). The motivation for using μ-Law function is that it accents samplings from small (soft) values, reducing the quantization error and increasing signal-to-quantization-noise-error (SQNR) for data transmission. Hence, we employ the μ-Law algorithm in GQ to improve the quantization robustness of ASR models.

In μ-Law expanding function (Eq. 5), $\boldsymbol{z}$, linearly spaced values within the range of -1 and 1, are warped as $\boldsymbol{z}'$ in which the values are driven closer to 0, except for the boundary poles. As shown in Fig. 1 (b), when μ increases, the linearly spaced values are more noticeably warped in the μ-Law transformed space: by adjusting the value of μ that minimizes the quantization error $\arg\min\limits_{\mu} ||\boldsymbol{z}' \times \boldsymbol{A}_{\text{soft}} - \boldsymbol{w}||_2$, quantization centroids in $\boldsymbol{z}$ can be re-distributed to better reflect the dynamic weight range of a specific neural component. A larger μ means the weight distribution is concentrated near 0; therefore, we allocate more quantization centroids near the origin. Smaller μ values indicate the weight distribution is tail-heavy.

$$\boldsymbol{z}' = \text{sng}(\boldsymbol{z})\left(\frac{(1+\mu)^{|\boldsymbol{z}|} - 1}{1+\mu}\right). \tag{5}$$

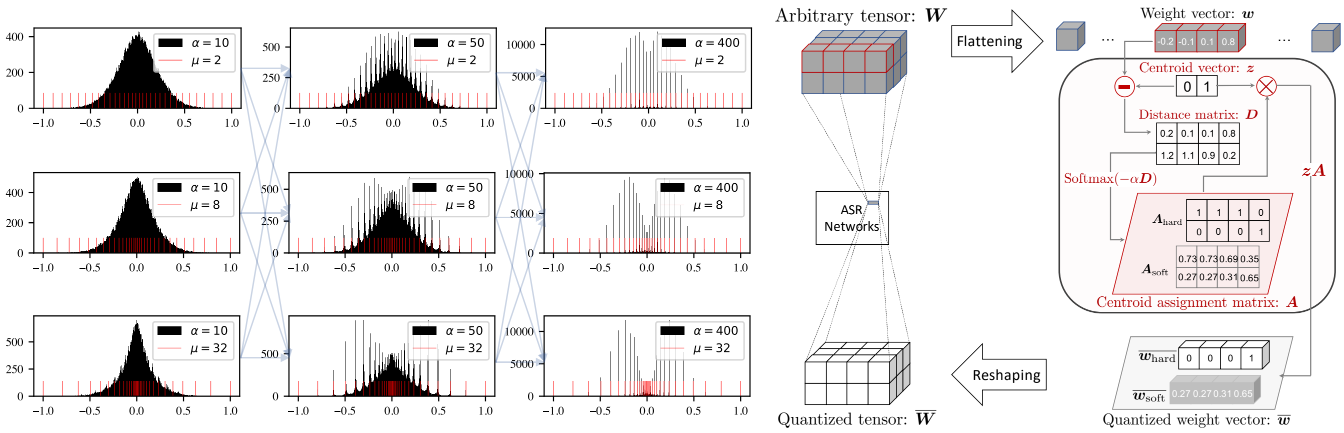
3.3. S8B Model for 8-Bit Computing

Due to limited chip memory size and bandwidth of the NNA, weights are loaded from system memory to the chip memory per matrix, which is time consuming. Hence, compressing the model into S8B can achieve inference speedup, even though NNA uses `INT8` for neural computing (see Fig.2).

Nonetheless, we map $\boldsymbol{z}'$ in Eq. 5 to the closest value in $[\cdots k/128 \cdots]$, where the integer $k \in [-128, 127]$, such that in-training and runtime quantization centroids are consistent.

3.4. Callback “Is All You Need”

A callback is a set of functions to be invoked at certain training stages. Under GQ, the callback is all you need: For any


α = 10
μ = 2
α = 50
μ = 2
α = 400
μ = 2
α = 10
μ = 8
α = 50
μ = 8
α = 400
μ = 8
α = 10
μ = 32
α = 50
μ = 32
α = 400
μ = 32
−1.0
−0.5
0.0
0.5
1.0
400
200
0
500
250
10000
5000
Arbitrary tensor: $\boldsymbol{W}$
Flattening
Weight vector: $\boldsymbol{w}$
-0.2
-0.1
0.1
0.8
Centroid vector: $\boldsymbol{z}$
0
1
Distance matrix: $\boldsymbol{D}$
0.2
0.1
0.1
0.8
1.2
1.1
0.9
0.2
Softmax($-\alpha\boldsymbol{D}$)
$\boldsymbol{zA}$
$\boldsymbol{A}_{\text{hard}}$
1
1
1
0
0
0
0
1
$\boldsymbol{A}_{\text{soft}}$
0.73
0.73
0.69
0.35
0.27
0.27
0.31
0.65
Centroid assignment matrix: $\boldsymbol{A}$
ASR Networks
Reshaping
$\overline{\boldsymbol{w}}_{\text{hard}}$
0
0
0
1
$\overline{\boldsymbol{w}}_{\text{soft}}$
0.27
0.27
0.31
0.65
Quantized tensor: $\overline{\boldsymbol{W}}$
Quantized weight vector: $\overline{\boldsymbol{w}}$

(a) During model training, the approximation scalar α increases, gradually and more effectively aggregating adjacent weights toward the corresponding centroid. μ is adaptable as model weights converge.

(b) An illustration of a GQ callback, in which every weight matrix in an arbitrary ASR network is compressed to a certain bit-depth given the annealing factor α and non-linearity factor μ.

Fig. 3: The Softmax temperature α and non-linearity factor μ for quantization centroids are updated during GQ callbacks.

tensor from an ASR model, during the callback, GQ will be applied to every weight vector $\boldsymbol{w} \in \boldsymbol{W}$ compressing it into $\overline{\boldsymbol{w}}$, for any $\boldsymbol{W} \in \mathbb{W}$. Concretely in Fig. 3 (b), we consider a binarized case where the centroid vector contains two values: 0 and 1. As the annealing factor α increases during training, the centroid assignment probabilities in $\boldsymbol{A}$ become more contrastive with the probability on the closest centroid becoming almost 1 and the other nearing 0. When α is sufficiently large, $\boldsymbol{A}$ approximates $\boldsymbol{A}_{\text{hard}}$ where each row is a one-hot vector. In the meantime, μ is adjusted as shown in Fig. 3 (a). Up to this stage, hard-quantizing $\mathbb{W}$ with $\boldsymbol{A}_{\text{hard}}$ does not yield noticeable degradation. In other words, without any regularizer, GQ effectively pushes weights to centroids during training for runtime model compression.

4. EXPERIMENT

4.1. Model

We consider both RNN-T and a lightweight streamable Conformer for experimental validation. We build a conventional RNN-T consisting of 5 LSTM encoding layers and 2 LSTM decoding layers with 1024 hidden units per layer and a fully connected joint layer. Furthermore, we benchmark GQ on an RNN-T variant with a branched encoder, named Bifocal RNN-T [17]. It has 2 encoders of different computational complexity and decides on-the-fly which encoder to use per input frame. Aside from the encoder in conventional RNN-T, bifocal RNN-T has another much smaller encoder with the same amount of layers but only 256 hidden units to process less intentful input frames.

Another end-to-end streaming ASR model is Conformer [15], whose audio encoder consists of stacks of Conformer blocks (see Table.1). Each Conformer block consists of two feedforward layers, a multi-head attention layer, and a convolutional module. We build a Conformer with 14 layers in which we use multi-head attention with 4 heads and each head with a dimension of 64. We make it causal by applying masks to multi-head attention layers to only attend to the left context. For the sub-sampling block of Conformer, we use two layers of 2D CNN with filters of 128 channels, kernel size of 3, and stride of 2. The feedforward hidden unit dimension is 1024. We adopt SpecAug [32] with the following hyper-parameters: maximum ratio of masked time frames=0.04, adaptive multiplicity=0.04, maximum ratio of masked frequencies=0.34, and number of frequency masks=2.

We use a word-piece tokenizer and generate 2500 word-piece tokens as the output vocabulary. We use the Adam optimizer [33] with β_1=0.9, β_2=0.98, and ϵ=1e-9. The learning rate is 0.002 with 10k warm-up steps. The step size is 1k, and 5k for Librispeech and de-identified in-house data, respectively, and the model is trained until no improvement is observed on the dev set. With μ=8, α is gradually increased from 10 to 400 before the hard compression.

4.2. Data

We train conventional RNN-T and Conformer on LibriSpeech data corpus with 960 hours of training data for 120k steps. The 5.4 hours of dev-clean dataset and 5.3 hours of dev-other dataset are used for checkpoint selection. Models are then evaluated on 5.4 hours of test-clean and 5.1 hours of test-other dataset. We train the Bifocal RNN-T on a de-identified far-field dataset consisting of 100k hours of human transcribed data for 700k steps, validate the training via 50k utterances of dev dataset, and evaluate the model with 50k test utterances that are frequently queried in spoken language understanding tasks. We benchmark the UPL metrics from 4 NNA embedded devices on 6k utterances.

Table 1: The Conformer architecture with $\sim$ 28M parameters and 14 blocks. All first 13 blocks share the same topology while the last block does not include the dense kernel labeled by ‡. Those with † form the dot product multi-head attention with relative positional embeddings (causal-relmha) [31]. Biases are not listed.

	Conformer Module	Kernel Shape	Params (M)
Encoder	Subsampling Kernel	(3, 3, 1, 128) (3, 3, 128, 128) (6144, 256)	1.72
	Conformer Block	(256, 1024) (1024, 256) (4, 256, 64)† (4, 256, 64)† (4, 256, 64)† (4, 256, 64)† (4, 64, 256)† (1, 1, 256, 512) (32, 1, 256, 1) (1, 1, 256, 256) (256, 1024) (1024, 256)‡ ×14	21.87
	Encoder Proj Dense Kernel	(256, 100)	0.26
Decoder	Single LSTM Layer	(256, 2560) (640, 2560)	2.62
	Vocab Proj Dense Kernel	(640, 512)	
Joint	Dense Kernel	(512, 2501)	1.28

Table 2: WER performance of Conventional RNN-T on LibriSpeech datasets

	dev-clean	dev-other	test-clean	test-other
32-bit baseline	8.11	21.27	8.68	22.29
8-bit QAT	8.15	21.41	8.70	22.36
6-bit QAT	8.32	21.84	8.90	22.82
6-bit GQ	7.76	20.80	8.39	22.12
5-bit GQ	7.93	21.34	8.33	22.16
4-bit GQ	8.23	21.82	8.78	22.54

4.3. Accuracy Comparison for RNN-T and Conformer

For conventional RNN-T, we compare GQ with its BP-QAT [12] in various S8B settings in Table.2. At 6-bit, GQ achieves a lower word error rate (WER) from all 4 datasets with 5.7% relative WER improvement on test-clean and 2.6% on test-other. It shows no degradation at 5-bit and 6-bit from 32-bit. GQ at 4-bit only shows less than 2% relative degradation compared to QAT at 8-bit.

We also report WERs from a GQ compressed Conformer in Table. 3. While the 32-bit baseline yields the best WER on the training set, both 5-bit and 6-bit quantized Conformers (with all other settings being the same) generalize better on dev-clean and dev-other datasets. Furthermore, we select the best checkpoint based on dev-clean WER and observe no degradation from the test sets with the model size reduced by 6.4×. Although this seems counter-intuitive in that the compressed model outperforms the 32-bit baseline, it is not rare as also shown in [9]. One explanation is that by driving weights towards quantization centroids, the search space is drastically reduced, yielding an arguably easier optimization process.

Table 3: GQ performance on a lightweight and streaming Conformer for LibriSpeech datasets. Here, full x-bit means all weights in the model are compressed to x-bit. If a module is labeled as x-bit, all weights in that module are compressed to x-bit with all other weights in 8-bit.

	train	dev-clean	dev-other	test-clean	test-other	size reduct.
32-bit baseline	0.86	5.62	14.11	5.74	14.21	–
Full 6-bit	1.42	5.06	13.42	5.26	13.38	5.3×
Full 5-bit	2.15	5.38	13.58	5.50	13.85	6.4×
4-bit Conv-Block	1.73	5.97	14.52	6.02	14.87	7.1×
4-bit MHSA	1.02	5.46	13.85	5.76	14.26	4.6×

Although GQ shows little ($< 5\%$ relatively) to no accuracy loss compared to the 32-bit baseline, it is worth mentioning that 4-bit quantization severely impacts the generalizability of Conformer on dev and test datasets, compared to the 5-bit mode. Even when only multi-head self-attention (MHSA) modules (approximately only 15% of total parameters) are quantized in 4-bit, the WER from dev and test datasets declines noticeably while that from the training dataset is much better. It is also observed from the 4-bit conv-block setting where about 77% of weights are 4-bit compressed.

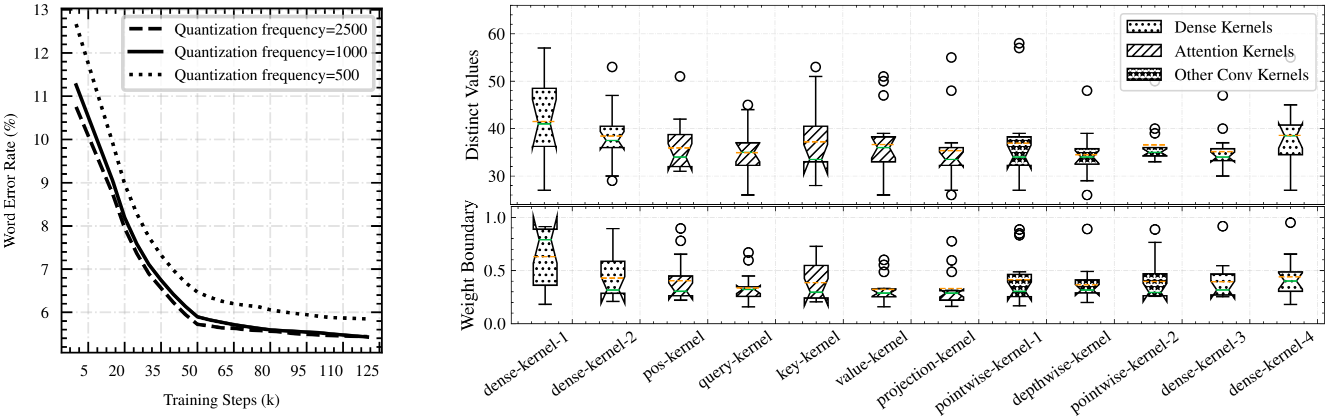
4.4. Accuracy, Memory Footprint and UPL Comparisons for Bifocal RNN-T

To better understand GQ's impact on memory footprint and UPL, we apply both GQ and our previous QAT methods to Bifocal RNN-T trained on a de-identified far-field dataset, where weights are compressed to various bit-depth configurations (see Table.4). Under S8B-QAT, we compress weights in all but the first layer in 5-bit for the left encoder (L-Enc), right encoder (R-Enc), and decoder with all other weights in 8-bit. With GQ, we compress the model to 5-bit (5B-GQ) or lower (S5B-GQ) without damaging the predictive performance on the frequent test set, compared to our previous 8-bit (8B-QAT) and sub-8-bit (S8B-QAT) methods.

We compile the ONNX files [34] of the trained models to hardware executable binary files for memory and UPL benchmarkings. In Table. 4, the memory consumption is reduced to 20.83MB in 5-bit from 30.07MB in 8-bit, which amounts to a 30.73% memory savings and yields to 20% p50 UPL reduction (32.30% and 32.75% UPL reduction for P90 and P99, respectively). Although we binarize the left encoder without degrading the accuracy, the impact on memory and latency is not significant as the left encoder is already small.

Table 4: Memory footprint, UPL and accuracy benchmarks for Bifocal RNN-T under various quantization bit-depth settings. The number of model parameters is the same among four experimental settings for a fair comparison.

	Bit-Depth				Normalized Memory Footprint		Normalized UPL						Normalized Accuracy
	L-Enc	R-Enc	Dec	Joint	Total	Rel. Dgrd.	P50	Rel. Dgrd.	P90	Rel. Dgrd.	P99	Rel. Dgrd.	Frequent Test Set
8B-QAT	8	8	8	8	1.00	–	1.00	–	1.55	–	2.58	–	1.00
S8B-QAT	5/8	5/8	5/8	8	0.83	-17.43	0.87	-12.52	1.22	-21.08	2.04	-20.68	0.99
5B-GQ	5	5	5	5	0.69	-30.73	0.80	-19.75	1.07	-31.30	1.76	-31.75	0.96
S5B-GQ	1	5	5	5	0.68	-32.09	0.80	-20.31	1.05	-32.20	1.70	-34.07	0.97



(a) Quantization frequency effect on training WER. The effect on other datasets is similar.

(b) Kernel-wise bit allocation and its weight boundary for the 6-bit Conformer. Every quantized kernel contains no more than 64 distinct weight values. Some kernels are with a narrow weight boundary and consume less than 32 values, indicating a possibility of 5-bit quantization.

Fig. 4: Without loss of generality, we use Conformer for ablation studies on quantization frequency and bit allocation of GQ.

4.5. Analysis of Quantization Frequency and Bit-Allocation

To understand how frequent should GQ be invoked during training, we alternate different quantization frequencies to train the 6-bit Conformer. Fig. 4 (a) shows that a too frequent in-training quantization setting hurts the predictive performance. One explanation is that it offsets the gradient effect by dragging weights back to the near-centroid zone. In practice, we observe no accuracy degradation even when the quantization frequency is 10k training steps, allowing us to compress a Conformer model to 4-bit in just a few GQ invocations. Additionally, the WER curves in Fig. 4 (a) are relatively smooth as GQ periodically performs, which indicates that the training is quantization aware even without any augmented regularizers.

We perform bitwise analysis of all encoding blocks of a lightweight and streaming Conformer (Fig. 4 (b)). 14 encoding blocks are included in the Conformer, accounting for $\sim 80\%$ of the total parameters. The Conformer is fully quantized to 6-bit requiring that all weight matrices (kernels) must have no more than 64 distinct weight values. Weight matrices are categorized as a dense kernel, MHSA kernel, or other types of convolution kernel, such as a depthwise convolution kernel. Using a box plot, we show the lower and upper quartiles of the 95% confidence interval along with the mean (orange dotted line) and median (green solid line) values for weight matrices from all blocks. It is worth noting that most weight matrices do not consume all 64 distinct values from 6-bit quantization to yield predictive results as good as 32-bit counterparts. Particularly, convolution kernels, whether in MHSA modules or not, are less bit consuming than dense kernels. The number of distinct values is found positively correlated to the weight boundary of the kernel: a smaller weight range indicates a fewer number of distinct values or quantization bit-depth and vice versa. To further reduce the memory footprint and UPL, compressing components with a narrow boundary to a lower bit-depth could be preferred over the dense kernel with a larger weight boundary.

5. CONCLUDING REMARKS

We proposed General Quantizer (GQ) a plug-and-play QAT mechanism, allowing models to be compressed to an arbitrary bit-depth during training without augmented regularizers. We applied GQ to three popular end-to-end Automatic Speech Recognition (ASR) models: conventional RNN-T, Bifocal RNN-T, and Conformer. In various sub-8-bit settings, GQ shows little to no accuracy degradation while noticeably reducing the memory footprint and user-perceived latency. GQ is model-agnostic and can be applied to feature map compression as one future direction.

6. REFERENCES

[1] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu, "Brecq: Pushing the limit of post-training quantization by block reconstruction," *arXiv preprint arXiv:2102.05426*, 2021.

[2] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao, "Post-training quantization for vision transformer," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[3] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson, "Loss aware post-training quantization," *Machine Learning*, vol. 110, no. 11, pp. 3245–3262, 2021.

[4] Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng, "Towards accurate post-training network quantization via bit-split and stitching," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9847–9856.

[5] Andrea Fasoli, Chia-Yu Chen, Mauricio Serrano, Xiao Sun, Naigang Wang, Swagath Venkataramani, George Saon, Xiaodong Cui, Brian Kingsbury, Wei Zhang, et al., "4-Bit Quantization of LSTM-Based Speech Recognition Models," *arXiv preprint arXiv:2108.12074*, 2021.

[6] Sehoon Kim, Amir Gholami, Zhewei Yao, Nicholas Lee, Patrick Wang, Aniruddha Nrusimha, Bohan Zhai, Tianren Gao, Michael W Mahoney, and Kurt Keutzer, "Integer-only zero-shot quantization for efficient speech recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4288–4292.

[7] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "TensorFlow: A System for Large-Scale Machine Learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[8] Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare, "Neural Network Quantization with AI Model Efficiency Toolkit (AIMET)," *arXiv preprint arXiv:2201.08442*, 2022.

[9] Shaojin Ding, Phoenix Meadowlark, Yanzhang He, Lukasz Lew, Shivani Agrawal, and Oleg Rybakov, "4-bit conformer with native quantization aware training for speech recognition," *arXiv preprint arXiv:2203.15952*, 2022.

[10] Junhao Xu, Shoukang Hu, Jianwei Yu, Xunying Liu, and Helen Meng, "Mixed precision quantization of transformer language models for speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7383–7387.

[11] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat, "Q8BERT: Quantized 8bit BERT," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*. IEEE, 2019, pp. 36–39.

[12] Hieu Duy Nguyen, Anastasios Alexandridis, and Athanasios Mouchtaris, "Quantization Aware Training with Absolute-Cosine Regularization for Automatic Speech Recognition.," in *Interspeech*, 2020, pp. 3366–3370.

[13] Kai Zhen, Hieu Duy Nguyen, Raviteja Chinta, Nathan Susanj, Athanasios Mouchtaris, Tariq Afzal, and Ariya Rastrow, "Sub-8-bit quantization aware training for 8-bit neural network accelerator with on-device speech recognition," in *Interspeech*, 2022.

[14] Nikko Strom, Haidar Khan, Wael Hamza, and Amazon Alexa AI, "Squashed weight distribution for low bit quantization of deep models," *Interspeech*, 2022.

[15] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., "Conformer: Convolution-Augmented Transformer for Speech Recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[16] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-Transducer," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 193–199.

[17] Jon Macoskey, Grant P Strimel, and Ariya Rastrow, "Bifocal neural asr: Exploiting keyword spotting for inference optimization," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5999–6003.

[18] Bo Li, Anmol Gulati, Jiahui Yu, Tara N Sainath, Chung-Cheng Chiu, Arun Narayanan, Shuo-Yiin Chang, Ruoming Pang, Yanzhang He, James Qin, et al., "A better and faster end-to-end model for streaming asr," in *ICASSP 2021-2021 IEEE International Conference*

on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 5634–5638.

[19] Tara N Sainath, Yanzhang He, Arun Narayanan, Rami Botros, Weiran Wang, David Qiu, Chung-Cheng Chiu, Rohit Prabhavalkar, Alexander Gruenstein, Anmol Gulati, et al., "Improving the latency and quality of cascaded encoders," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8112–8116.

[20] Qian Lou, Feng Guo, Lantao Liu, Minje Kim, and Lei Jiang, "AutoQ: Automated kernel-wise neural network quantization," *arXiv preprint arXiv:1902.05690*, 2019.

[21] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al., "Tensorflow lite micro: Embedded machine learning for tinyml systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, 2021.

[22] Julieta Martinez, Jashan Shewakramani, Ting Wei Liu, Ioan Andrei Bârsan, Wenyuan Zeng, and Raquel Urtasun, "Permute, quantize, and fine-tune: Efficient compression of neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15699–15708.

[23] Tairen Piao, Ikhyun Cho, and U Kang, "SensiMix: Sensitivity-Aware 8-bit index & 1-bit value mixed precision quantization for BERT compression," *PloS one*, vol. 17, no. 4, pp. e0265621, 2022.

[24] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," *Advances in neural information processing systems*, vol. 30, 2017.

[25] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[26] K. Zhen, J. Sung, M. S. Lee, S. Beack, and M. Kim, "Cascaded cross-module residual learning towards lightweight end-to-end speech coding," in *Interspeech*, 2019.

[27] Xue Jiang, Xiulian Peng, Huaying Xue, Yuan Zhang, and Yan Lu, "Cross-scale vector quantization for scalable neural speech coding," *arXiv preprint arXiv:2207.03067*, 2022.

[28] D. Petermann, S. Beack, and M. Kim, "Harp-net: Hyper-autoencoded reconstruction propagation for scalable neural audio coding," in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2021, pp. 316–320.

[29] Youngju Cheon, Soojoong Hwang, Sangwook Han, Inseon Jang, and Jong Won Shin, "Coded speech enhancement using neural network-based vector-quantized residual features.," in *Interspeech*, 2021, pp. 1664–1668.

[30] Hisashi Kaneko, "A unified formulation of segment companding laws and synthesis of codecs and digital compandors," *Bell System Technical Journal*, vol. 49, no. 7, pp. 1555–1588, 1970.

[31] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[32] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[33] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] Junjie Bai, Fang Lu, Ke Zhang, et al., "ONNX: Open Neural Network Exchange," https://github.com/onnx/onnx, 2019.