Q-ASR: Integer-only Zero-shot Quantization for Efficient Speech Recognition

Sehoon Kim, Amir Gholami[†], Zhewei Yao, Anirudda Nrusimha,
Bohan Zhai, Tianren Gao, Michael W. Mahoney, Kurt Keutzer
University of California, Berkeley
{sehoonkim, amirgh, zheweiy, aninrusimha,
zhaibohan, terrygao87, mahoneymw, keutzer}@berkeley.edu

Abstract—End-to-end neural network models achieve improved performance on various automatic speech recognition (ASR) tasks. However, these models perform poorly on edge hardware due to large memory and computation requirements. While quantizing model weights and/or activations to low-precision can be a promising solution, previous research on quantizing ASR models is limited. Most quantization approaches use floating-point arithmetic during inference; and thus they cannot fully exploit integer processing units, which use less power than their floating-point counterparts. Moreover, they require training/validation data during quantization for finetuning or calibration; however, this data may not be available due to security/privacy concerns. To address these limitations, we propose Q-ASR, an integer-only, zero-shot quantization scheme for ASR models. In particular, we generate synthetic data whose runtime statistics resemble the real data, and we use it to calibrate models during quantization. We then apply Q-ASR to quantize QuartzNet-15x5 and JasperDR-10x5 without any training data, and we show negligible WER change as compared to the full-precision baseline models. For INT8-only quantization, we observe a very modest WER degradation of up to 0.29%, while we achieve up to 2.44× speedup on a T4 GPU. Furthermore, Q-ASR exhibits a large compression rate of more than 4× with small WER degradation.

I. INTRODUCTION

Recent end-to-end neural network (NN) models have achieved state-of-the-art results in various automatic speech recognition (ASR) tasks [12, 13, 20, 21, 34]. However, these accuracy improvements come with increasingly large model sizes. For instance, the highest performing versions of Jasper [21], Conformer [12], ContextNet [13], and Transformer-Transducer [34] contain 333M, 118.8M, 112.7M, and 139M parameters, respectively. As a result, memory and compute resources have limited real-time edge deployment.

One promising solution to tackle this challenge is to quantize model weights and/or activations to low-precision. This provides multiple benefits. First, quantization reduces the memory footprint by storing weights/activations in low-precision, e.g., INT8 uses 4× less memory relative to FP32. Second, quantization accelerates execution and decreases power consumption by using specialized hardware for low-precision arithmetic [17, 18, 31]. Following its success on many computer vision [8, 17, 31, 33] and natural language processing tasks [18, 27, 32], there have been attempts to apply quantization to ASR models [4, 22, 26, 30]. However, prior quantization work lacks two important features: integer-only quantization and zero-shot quantization.

Integer-only quantization [17, 18, 31] is a quantization scheme where *all* operations (e.g., convolution and matrix multiplication) are performed using low-precision integer arithmetic. To the best of our knowledge, all the previous quantization works for ASR models use *simulated quantization*, where all or part of operations are performed with floating point arithmetic. For instance, [26] only performs convolution with integer arithmetic and leaves ReLU and Batch Normalization (BatchNorm) as floating point operations. Compared to simulated quantization, integer-only quantization can decrease latency and increase power efficiency by fully utilizing efficient integer processing units. More importantly, it also allows deployment on popular and highly optimized edge processors designed for embedded, mobile, or IoT devices that do not support floating point arithmetic. ARM Cortex-M [3], GreenWaves GAP-9 [10], and Google's Edge TPU [1] are some examples of edge processors that do not contain dedicated floating point units.

Zero-shot quantization [5, 6, 9, 14, 15, 29] is a quantization scheme that does not require any training or validation data. This is important because datasets are not always available, especially for ASR use cases

such as smart speakers or healthcare, where privacy and security are concerns. However, prior work is based on either quantization-aware training [4, 22, 26, 30] or post-training quantization [4, 26], both of which require access to training/validation data to finetune or calibrate the quantized models. As such, these methods cannot be applied if no data is available.

We propose Q-ASR to address the aforementioned limitations of the previous quantization work for ASR models. In particular, we make the following contributions:

- We apply an integer-only, zero-shot quantization scheme to ASR models. This allows efficient deployment on integer processing units, without any access to training/validation data. See Section II-B and II-C for details.
- For zero-shot quantization, we apply a synthetic data generation scheme where input data (i.e., mel spectrogram) is generated to match runtime statistics of the real training data. See Section II-C and Figure 3 for details.
- We apply Q-ASR to QuartzNet-15x5 [20] and JasperDR-10x5 [21] and evaluate their word-to-error-rate (WER) on the Librispeech benchmark [24]. Despite the fact that we only use static quantization, we achieve negligible accuracy degradation. In particular, with 8-bit quantization for weights and activations, Q-ASR exhibits WER degradation of only up to 0.29%. See Section III-A for details.
- We deploy INT8-only QuartzNet-15x5 on a T4 GPU using TensorRT [23], and we show that it achieves up to $2.44\times$ speedup as compared to its FP32 counterpart. See Section III-B for details.

II. METHODOLOGY

A. Basic Quantization Method

In this work, we use *uniform symmetric quantization*. This method uniformly maps a real number x to an integer value $q \in [-2^{b-1}, 2^{b-1} - 1]$, where b is the quantization bit precision:

$$q = \mathrm{Q}(x, b, S) = \mathrm{Int}\left(\frac{\mathrm{clip}(x, -\alpha, \alpha)}{S}\right), \qquad (1)$$

where Q is the quantization operator, Int is the round operation, clip is the truncation function with the clipping parameter α, and S is the scaling factor defined as $\alpha/(2^{b-1} - 1)$. The reverse mapping from the quantized values q to the real values (aka dequantization) is:

$$\tilde{x} = \mathrm{DQ}(q, S) = Sq \approx x, \qquad (2)$$

where DQ denotes the dequantization operator.

Determining the clipping range $[-\alpha, \alpha]$ that best represents the range of the input x is a primary challenge of quantization. One popular choice in practice is to use the minimum and maximum values of the input x, i.e., $\alpha = \max(|x_{max}|, |x_{min}|)$. However, this approach is susceptible to outliers. A unnecessarily large clipping range due to a few outliers increases the rounding error of quantized values within the range. One way to alleviate this is to use percentile values, e.g., 99% smallest/largest, instead of the min/max values.

In *dynamic quantization*, clipping ranges are computed during inference. However, input statistics (e.g., min, max, and percentile) are costly to compute in real-time. Furthermore, calculating these statistics requires floating point operations that would prevent us from doing integer-only quantization. Therefore, in this work, we only use *static quantization* where we pre-compute the clipping ranges and fix them during inference as in [17, 18, 31]. It is straightforward to pre-compute the ranges for weights as they are fixed during inference. However, activations vary across different inputs, and therefore their ranges vary. One popular method to address this is *calibration*, which runs a series of training data to compute the typical range of activations. Later in Section II-C, we extend this idea and show how to calibrate *without* training data. We refer the interested readers to [11] for more details in quantization methods.

B. Integer-only Quantization

Integer-only quantization [17, 18, 31] not only represents the model weights and activations with low-precision integer values, but it also carries out the entire inference with integer arithmetic. Broadly speaking, the core of integer-only quantization is the linear property of the operations. For instance, let q_W and q_x denote the quantized values for weight $\tilde{W}$ and activation $\tilde{x}$, respectively, i.e., $\tilde{W} = S_W q_W$ and $\tilde{x} = S_x q_x$. Then, $\mathrm{Conv}(\tilde{W}, \tilde{x}) = \mathrm{Conv}(S_W q_W, S_x q_x)$ is equivalent to $S_W S_x \mathrm{Conv}(q_W, q_x)$ due to the linear property. Therefore, we can apply *integer* convolution directly to the quantized values q_W and q_x without dequantizing them into $\tilde{W}$ and $\tilde{x}$ and performing floating point convolution. This property holds true for convolution, matrix multiplication, and ReLU, which are the basic building blocks for CNN-based NN architectures.

C. Zero-shot Quantization

Quantization generally requires training data (1) to pre-compute the clipping ranges for Eq. 1 (aka calibration)
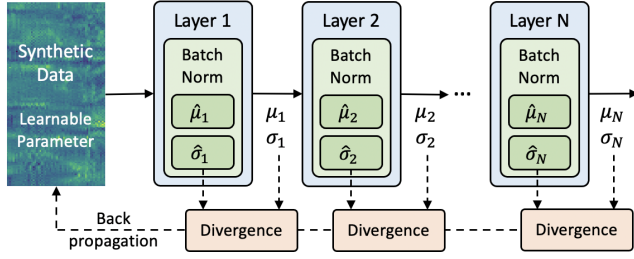
Fig. 1: *End-to-end process of synthetic data (i.e., mel spectrogram) generation for zero-shot quantization. Input data is trained with standard backpropagation in a direction that matches the hidden activation statistics of the real data.*

and/or (2) to finetune the quantized model to recover accuracy degradation from quantization. However, in many cases, pre-trained models are distributed without the data they were trained on due to proprietary, privacy, and security concerns. As such, multiple zero-shot quantization schemes have been proposed, mostly in the field of computer vision, to allow quantization in such cases. A widely adopted method is to generate synthetic image data that can be used in place of the real training data for calibration and/or finetuning [5, 6, 9, 14, 15, 29].

Similar to [9, 14], we generate synthetic data with a similar data distribution to the training data. In particular, we generate the data to match the runtime statistics (i.e., distributions of the hidden activations) of the training data using the running means and variances stored in BatchNorm layers. Note that the running means and variances capture the runtime statistics of the training data. Let x denote the synthetic data, h_i the activation of i-th BatchNorm layer with x as an input, and μ_i and σ_i the mean and variance of h_i. Additionally, let $\hat{\mu}_i$ and $\hat{\sigma}_i$ denote the running mean and variance of the BatchNorm. Then, we aim to minimize the loss defined by the Kullback-Leibler (KL) divergence between $N(\hat{\mu}_i, \hat{\sigma}_i)$ and $N(\mu_i, \sigma_i)$, assuming that the hidden activations follow the Gaussian distribution:

$$
\begin{aligned}
x^* &= \arg\min_x \sum_{i=1}^{N} KL(N(\hat{\mu}_i, \hat{\sigma}_i) || N(\mu_i, \sigma_i)) \\
&= \arg\min_x \sum_{i=1}^{N} \log \frac{\sigma_i}{\hat{\sigma}_i} - \frac{1}{2}\left(1 - \frac{\hat{\sigma}_i^2 + (\hat{\mu}_i - \mu_i)^2}{\sigma_i^2}\right)
\end{aligned}
\tag{3}
$$

In computer vision tasks, input x is regarded as a learnable parameter and is trained with standard backpropagation to reduce the defined loss [5, 14]. However, it is not straightforward to apply this directly to ASR, as in ASR tasks the inputs are continuous audio signals, and

they often involve non-differentiable preprocessing steps. We instead propose to generate the synthetic data for the mel spectogram. The latter is a widely used preprocessed representation of audio signals [12, 13, 20, 21, 25]. Also, its data representation is more suitable for synthetic data generation as it can be regarded as a 1-dimensional image. Specifically, our zero-shot quantization scheme consists of two steps.

- **Synthetic data generation:** A batch of 1-dimensional data is randomly initialized and set as a learnable parameter. It is then fed into the full-precision (i.e., non-quantized) model to compute the loss in Eq. 3. This minimizes the KL divergence between two distributions, one from the real training data and the other from the synthetic data, for each BatchNorm activation. Using backpropagation, the input is iteratively trained using gradient descent to minimize the loss. Figure 1 illustrates the end-to-end process of synthetic data generation.
- **Calibration:** The synthetic data is fed into the target model in place of the real mel spectrograms to determine the clipping ranges for all activations. Then the model is quantized according to Eq. 1.

III. RESULTS

In this section, we first demonstrate in Section III-A that Q-ASR can match the accuracy of full-precision baselines. We then evaluate in Section III-B the latency speedup of Q-ASR on real hardware. We select QuartzNet15x5 [20] and JasperDR-10x5 [21] as baselines because: (1) They are CNN-based architectures that are compatible with the integer-only quantization scheme described in Section II-B; (2) They contain BatchNorm layers, enabling the zero-shot quantization method described in Section II-C; (3) QuartzNet is already a highly optimized architecture designed with a small number of parameters, making it a competitive baseline. However, Q-ASR is not a model-specific solution, and it can be applied to many other ASR models. For example, our zero-shot quantization method can be applied to any model that uses BatchNorm [16].

A. Accuracy Evaluation

For accuracy evaluation, we use the NeMo implementation of the pre-trained QuartzNet-15x5 and JasperDR-10x5 [2] for the full-precision baseline models. The synthetic data is initialized with the uniform distribution from $[-0.3, 0.3]$, and is trained using the Adam optimizer [19] with batch size 8, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of {0.03, 0.04, 0.05, 0.06} for {200,

3

Table I: *WER of the quantized QuartzNet15x5 and JasperDR-10x5 with different bit-width settings, evaluated on the LibriSpeech dev-clean/other and test-clean/other datasets. W and A are the bit-width for weights and activations. We denote the baseline accuracy of the full-precision models at the top row. We also include the model size and BOPs for each model. BOPs are computed using a 1-second input with a sampling rate of 16k.*

(a) *QuartzNet-15x5*

Method	W/A	dev		test		Size	BOPs
		clean	other	clean	other	(MB)	(G)
Baseline	32/32	3.80	10.05	3.82	10.08	73.81	1890
Q-ASR	8/8	3.92	10.28	4.04	10.37	18.45	118
	6/8	4.16	10.83	4.32	11.06	13.84	88.8
	6/6	5.09	12.31	5.33	12.74	13.84	66.6

(b) *JasperDr-10x5*

Method	W/A	dev		test		Size	BOPs
		clean	other	clean	other	(MB)	(T)
Baseline	32/32	3.47	10.40	3.68	10.49	1208	33.3
Q-ASR	8/8	3.47	10.49	3.68	10.57	302.0	2.08
	6/8	3.52	10.59	3.73	10.68	226.5	1.56
	6/6	3.69	11.13	3.93	11.07	226.5	1.17

250} iterations. We generate 20 such batches and use them for calibration. Figure 3 contains some examples of the generated synthetic data. For calibration, we find it effective to use percentile values when computing the clipping ranges. We searched for the optimal percentile in the search space of {100, 99.9999, 99.999, 99.998, 99.997, 99.996}. After determining the clipping ranges, we quantize the baseline model with three different bit-width settings: W8A8 (i.e., 8-bit for both weights and activations), W6A8, and W6A6. We do not finetune the model after quantization. All the reported numbers are averaged over 4 different runs (i.e., synthetic data generation, calibration, and evaluation).

Both the baseline and quantized models are evaluated on the widely used LibriSpeech [24] benchmark. Table I compares the WER of the baseline and quantized models with different bit-width settings on dev-clean/other and test-clean/other datasets, where hyperparameters are tuned on the dev sets. We also include the model size and BOPs (bit-operations) of each model for comparison. BOP is the total number of bit operations and is a hardware-agnostic proxy to model complexity [28]. For QuartzNet-15x5, Q-ASR with W8A8 results in a negligible WER degradation of 0.22% and 0.29% on the test-clean/other datasets as compared to the full-precision baseline. While W8A8 can have up to 4× reduction of the model size, we can further

reduce the size by allocating a lower bit-width for the weights. W6A8 can achieve more than 5× reduction of the model size, but it results in 0.50% and 0.97% higher WER on the test-clean/other datasets. More aggressive quantization using W6A6 results in 3% higher WER on the test-other dataset.

We observe a similar trend for JasperDr-10x5, where W8A8 quantization does not show any WER degradation for the clean sets, and only exhibits 0.08% degradation on the test-other dataset. Q-ASR also exhibits negligible WER increase of up to 0.19% and 0.58% on the test sets for more aggressive quantization using W6A8 and W6A8, respectively. It can be interpreted that JasperDr-10x5 is more resilient to quantization than QuartzNet15x5 due to its larger model size.

B. Latency Evaluation

We evaluate the latency speedup of Q-ASR by direct deployment of the quantized QuartzNet models on a Tesla T4 GPU with the Turing Tensor Cores that support accelerated INT8 execution. We use NVIDIA's TensorRT library [23] for model deployment. Specifically, all the experiments were conducted on Google Cloud Platform virtual machine with a single Tesla T4 GPU, CUDA 11.1, cuDNN 8.0, and TensorRT 7.2.1. Although we select T4 GPU as our target device due to its extensive software support [7, 23], including TensorRT, we should highlight that our approach is not specific to GPUs, and one could also deploy the quantized models to other processors.

Table II shows the inference latency speedup of the quantized QuartzNet15x5 as compared to the FP32 baseline. As the Turing Tensor Cores only support INT8 (and INT4) computations, we only test W8A8 for the quantized model. We use two audio inputs, which were 10 and 20 seconds long, with a sampling rate of 16K. The preprocessing is excluded from latency measurement; only the encoding and decoding are measured. As one can see, INT8 inference of Q-ASR achieves up to 2.44× speedup as compared to FP32.

Figure 2 (Left) further compares the latency of the baseline and quantized QuartzNet models with different configurations. QuartzNet-5x5, 10x5, and 15x5 are small, medium, and large variants of the QuartzNet family, with different numbers of layers [20]. In Figure 2 (Right), we additionally plot the size of each model, measured through the corresponding ONNX-exported version of each model to account for the entire memory footprint. As one can see, the quantized QuartzNet15x5 is comparable to the smallest QuartzNet5x5 in terms of latency (20.4% slower) and model size (29.8% smaller), but at the same

Table II: *Latency speedup of INT8 inference of the quantized QuartzNet15x5 (Q-ASR) with respect to FP32 inference of the full-precision baseline. W and A are the bit-width for weights and activations, respectively. Latency is measured using 10- and 20-second audio inputs with a sampling rate of 16k.*

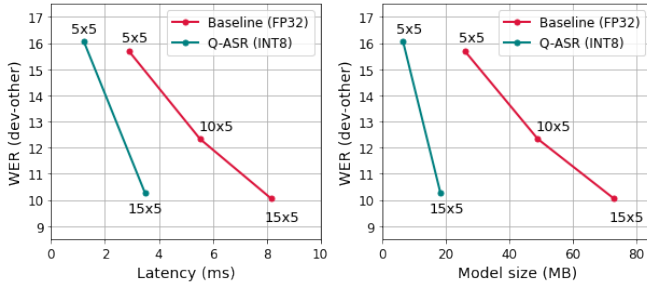| Method	W	A	Latency (ms)		Speedup	
			10s	20s	10s	20s
Baseline	32	32	8.17	14.43	1×	1×
Q-ASR	8	8	3.48	5.91	2.35×	2.44×

Fig. 2: *Latency (Left) and model size (Right) of the quantized (Q-ASR) and full-precision (Baseline) QuartzNets with different sizes. WER of QuartzNet-10x5 is the reported value in [20], and latency is measured with 10-second audio input with a 16k of sampling rate.*

Table III: *WER of the quantized QuartzNet15x5 calibrated with random data (Random) and the synthetic data (Q-ASR). The models are evaluated on the LibriSpeech dev-clean/other and test-clean/other datasets, and W and A are the bit-width for weights and activations, respectively. Calibration with the synthetic data consistently outperforms calibration with random data for all bit-width settings.*

Method	W/A	dev		test		Size	BOPs
		clean	other	clean	other	(MB)	(G)
Baseline	32/32	3.80	10.05	3.82	10.08	73.81	1890
Random	8/8	5.39	12.20	5.56	12.55	18.45	118
Q-ASR	8/8	3.92	10.28	4.04	10.37	18.45	118
Random	6/8	6.26	13.57	6.56	14.10	13.84	88.8
Q-ASR	6/8	4.16	10.83	4.32	11.06	13.84	88.8
Random	6/6	9.87	20.96	10.10	21.75	13.84	66.6
Q-ASR	6/6	5.09	12.31	5.33	12.74	13.84	66.6

time, it achieves similar WER as compared to the largest QuartzNet15x5. Interestingly, Q-ASR achieves the WER of the largest model, while only needing the computation and memory requirements of the smallest model. This is an important observation which suggests why the small accuracy degradation after quantization can be negligible, when a practitioner wants to reduce the latency. The alternative approach of using a full precision but shallower model is clearly sub-optimal to quantization here.

IV. DISCUSSION

A. Ablation studies

Here, we show the effectiveness of the zero-shot quantization scheme. To do so, we additionally quantize QuartzNet15x5 using a set of random calibration data from the uniform distribution in $[-3, 3]$ (denoted as "Random" for the Method column in Table III). We observe significant WER degradation when we calibrate the model with random data instead of the synthetic data of Q-ASR. With random calibration data, the test-other WER degrades by 2.47%, 4.02%, and 11.67% for W8A8, W6A8, and W6A6, respectively, which is consistently higher than the corresponding values with Q-ASR as

shown in Table III. The results confirm the need to generate synthetic data that well represents the actual data used in training.

B. Analysis of Synthetic Data Generation

Here, we briefly analyze the synthetic data generated according to Section II-C. In the synthetic data generation, we start with a randomly initialized mel spectogram as shown in the second row of Figure 3. We then solve Eq. 3 to generate synthetic mel spectrograms. We show three such generated data in the last three rows of Figure 3. First, note that the synthetic data can capture the local patterns observed in real mel spectrograms. Second, we observe that if the synthetic data is fed into QuartzNet followed by a 3-gram language model, it can be decoded into meaningful words and sentences, instead of a sequence of random characters. Examples of the decoded sentences are captioned below the corresponding synthetic data in Figure 3. This is a further evidence that the synthetic data is close to the actual training data, and it explains why it enables good quantization results.

V. CONCLUSION

In this work, we proposed Q-ASR, a zero-shot quantization scheme for ASR models that uses integer-only computation. Our proposed method entirely removes floating point operations from inference, thereby allowing efficient model deployment on power-efficient integer processing units. In addition, it does not require any training data, as it calibrates models with synthetic data that resembles the real data. Q-ASR exhibits a large
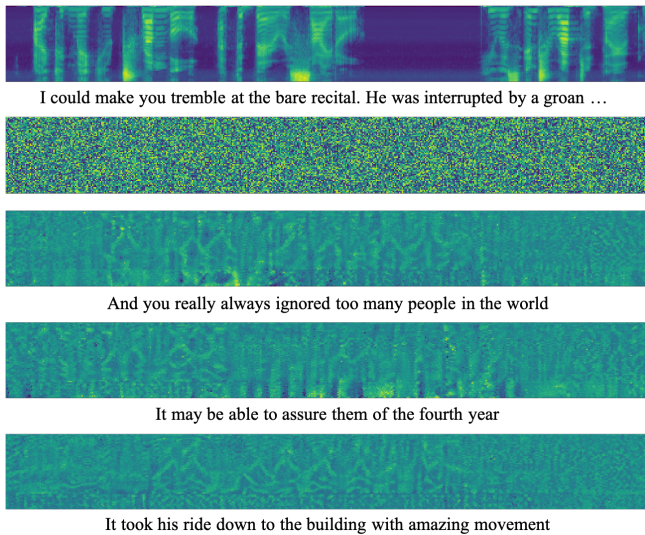
I could make you tremble at the bare recital. He was interrupted by a groan …

And you really always ignored too many people in the world

It may be able to assure them of the fourth year

It took his ride down to the building with amazing movement

Fig. 3: *Comparison of the real (1st), random (2nd), and synthetic (3rd-5th) mel spectrograms. Below each synthetic mel spectrogram is the text that it is decoded.*

compression rate of more than $4\times$ with small WER degradation. In particular, quantization of QuartzNet-15x5 and JasperDr-10x5 models shows only 0.29% and 0.08% WER degradation for INT8-only quantization. Furthermore, we achieve $2.44\times$ speedup by directly deployment of INT8-only QuartzNet-15x5 on a T4 GPU.

VI. Acknowledgements

References

[1] Google Edge TPU, https://cloud.google.com/edge-tpu.

[2] NeMo, https://github.com/nvidia/nemo.

[3] ARM. Cortex-M, https://developer.arm.com/ip-products/processors/cortex-m.

[4] Alex Bie, Bharat Venkitesh, Joao Monteiro, Md Haidar, and Mehdi Rezagholizadeh. A simplified fully quantized transformer for end-to-end speech recognition. *arXiv preprint arXiv:1911.03604*, 2019.

[5] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. ZeroQ: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

[6] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3514–3522, 2019.

[7] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, , Carlos Guestrin, and Arvind Krishnamurthy. TVM: An automated end-to-end optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, 2018.

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

[9] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 710–711, 2020.

[10] Eric Flamand, Davide Rossi, Francesco Conti, Igor Loi, Antonio Pullini, Florent Rotenberg, and Luca Benini. GAP-8: A RISC-V SoC for AI at the edge of the IoT. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–4. IEEE, 2018.

[11] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[12] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[13] Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. ContextNet: Improving convolutional neural networks for automatic speech recognition with global context. *arXiv preprint arXiv:2005.03191*, 2020.

[14] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2020.

[15] Xiangyu He, Qinghao Hu, Peisong Wang, and Jian Cheng. Generative zero-shot network quantization. *arXiv preprint arXiv:2101.08430*, 2021.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

[18] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-BERT: Integer-only BERT quantization. *arXiv preprint arXiv:2101.01321*, 2021.

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

[20] Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. QuartzNet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6124–6128. IEEE, 2020.

[21] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An end-to-end convolutional neural acoustic model. *arXiv preprint arXiv:1904.03288*, 2019.

[22] Hieu Duy Nguyen, Anastasios Alexandridis, and Athanasios Mouchtaris. Quantization aware training with absolute-cosine regularization for automatic speech recognition. *Proc. Interspeech 2020*, pages 3366–3370, 2020.

[23] NVIDIA. TensorRT: https://developer.nvidia.com/tensorrt.

[24] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[25] 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.

[26] Amrutha Prasad, Petr Motlicek, and Srikanth Madikeri. Quantization of acoustic model parameters in automatic speech recognition framework. *arXiv preprint arXiv:2006.09054*, 2020.

[27] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-BERT: Hessian based ultra low precision quantization of BERT. In *AAAI*, pages 8815–8821, 2020.

[28] Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. *arXiv preprint arXiv:2005.07093*, 2020.

[29] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In *European Conference on Computer Vision*, pages 1–17. Springer, 2020.

[30] Yiwu Yao, Yuchao Li, Chengyu Wang, Tianhang Yu, Houjiang Chen, Xiaotang Jiang, Jun Yang, Jun Huang, Wei Lin, Hui Shu, and Chengfei Lv. Int8 winograd acceleration for conv1d equipped asr models deployed on mobile devices. *arXiv preprint arXiv:2010.14841*, 2020.

[31] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael W Mahoney, and Kurt Keutzer. HAWQV3: Dyadic neural network quantization. *arXiv preprint arXiv:2011.10680*, 2020.

[32] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8BERT: Quantized 8bit BERT. *arXiv preprint arXiv:1910.06188*, 2019.

[33] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[34] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. Transformer Transducer: A streamable speech recognition model with transformer encoders and RNN-T loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833. IEEE, 2020.