QUANTIZED WINOGRAD ACCELERATION FOR CONV1D EQUIPPED ASR MODELS ON MOBILE DEVICES

Yiwu Yao, Chengyu Wang, Jun Huang

Alibaba Group, Hangzhou, China

ABSTRACT

The intensive computation of Automatic Speech Recognition (ASR) models obstructs them from being deployed on mobile devices. In this work, we present a novel quantized Winograd optimization framework, combining quantization and fast convolution to achieve efficient inference acceleration for ASR models on mobile devices. To avoid the information loss due to the combination of quantization and Winograd convolution, a Range-Scaled Quantization (RSQ) training method is proposed, integrating integer-range scaling and quantization noise minimization. Moreover, the Conv1D equipped DFSMN (ConvDFSMN) model is designed for mobile applications and experimental verification. We conduct extensive experiments on ConvDFSMN and Wav2letter models, demonstrating that the models can be effectively optimized with the proposed optimization framework. Especially, the optimized Wav2letter model achieves 1.48× speedup for end-to-end inference and 1.92× speedup for model backbone inference on ARMv7-based mobile devices, with only an approximate 0.07% decrease in WER on AIShell-1.

***Index Terms*—** Range-Scaled Quantization, INT8 Winograd, ConvDFSMN, mobile deployment

1. INTRODUCTION

Recent years has witnessed the great success of deep neural networks in real-world applications, *e.g.,* image classification [1] and speech recognition [2]. While sophisticated neural networks have adequately advanced the performance, the computational workload and the storage requirement have also been drastically increased, which obstruct their applications in mobile devices [3]. Especially, Automatic Speech Recognition (ASR) applications require real-time voice interaction, leading to a more challenging field of deploying ASR models on mobile devices [4, 5, 6].

In the literature, mainstream ASR models can be classified into feed-forward structures [7, 8, 9, 10, 11] and auto-regressive structures [12, 13, 14, 15]. Auto-regressive models (*e.g.,* the transformer model) exploit beam search to achieve end-to-end ASR, but are difficult to accelerate for mobile applications due to while-loop decoding. In contrast, feed-forward models avoid the auto-regressive property, showing better inference efficiency. To enhance the ability of capturing contextual information, several feed-forward acoustic models (*e.g.,* TDNN [7] and DFSMN [9]) use the one-dimensional convolution (Conv1D) to extract contextual features. However, the ratio of the computation to memory access of Conv1D is relatively low, which forms the inference performance bottleneck of these ASR models.

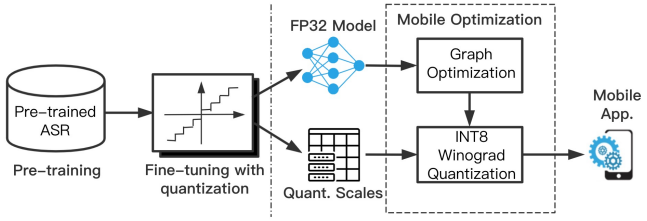

Fig. 1. Proposed optimization framework: i) the pre-trained Conv1D equipped ASR model is fine-tuned with RSQ; ii) the re-trained model with quantization scales is sent to mobile optimization flow, where graph optimization and INT8 Winograd are conducted for mobile deployment.

Quantization [16, 17, 18, 19, 20] is a crucial technique to reduce the computation latency and the model size when deploying deep neural networks on mobile devices. Current mobile inference frameworks (*i.e.,* TFLite [21], MNN [22]) provide 8-bit quantization runtime based on General Matrix Multiplication (GEMM). Compared to matrix multiplication, Winograd's minimal filtering [23] reduces multiplications in convolution, showing higher efficiency. Unfortunately, the simple combination suffers from precision overflow [24, 25], which obstructs applying these two methods simultaneously.

We propose a novel quantized Winograd optimization framework, integrating Range-Scaled Quantization (RSQ) training and low-precision quantized Winograd inference for mobile applications (shown in Figure 1). Firstly, the network is fine-tuned with RSQ including integer range scaling and quantization noise minimization. Weights and activations are then quantized as lower precision to prevent the addition overflow in quantized Winograd. Finally, in the Winograd domain, the full integer Hardmard production is executed to improve the efficiency of convolution for mobile deployment.

Y. Yao and C. Wang contributed equally to this work.

2. RANGE-SCALED WINOGRAD QUANTIZATION

2.1. Analysis on Overflow of Quantized Winograd

Previously, the literature shows that Conv1D operations (with kernel size $k \geq 3$) can be efficiently accelerated with the Winograd algorithm [23]. However, when combining quantization and Winograd, if we quantize the values before Winograd convolution, the addition overflow will occur during Winograd transformation. To verify this argument, we consider the symmetric uniform scheme [16] to quantize the input activation d and weight g as t-bit signed integers. Here, the quantized Winograd of Conv1D can be computed as:

$$S = A^T[(GQ(g)) \odot (B^T Q(d))], \tag{1}$$

where S is the output. A, G and B are Winograd transformation matrices. $\odot$ represents the Hadamard production. $Q(\cdot)$ is the quantization function in the non-Winograd domain.

Assume that the elements of quantized activation $Q(d)$ and weight $Q(g)$ reach the positive boundary values $D_Q = 2^{t-1} - 1$ and $G_Q = 2^{t-1} - 1$ (*e.g.,* $D_Q = 127$ when $t = 8$ for INT8 Winograd) The $F(2, 3)$ transformation [23] is:

$$\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} D_Q \\ D_Q \\ D_Q \\ D_Q \end{bmatrix} = \begin{bmatrix} 0 \\ 2D_Q \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

$$2 \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} G_Q \\ G_Q \\ G_Q \end{bmatrix} = \begin{bmatrix} 2G_Q \\ 3G_Q \\ G_Q \\ 2G_Q \end{bmatrix} \tag{3}$$

where the number 2 multiplied in Eq. 3 is for integer-only computation in the Winograd domain, which will be further divided during de-quantization.

The transformation outputs of Eq. 2 and Eq. 3 reach $2D_Q$ and $3G_Q$, which are both out of the quantized integer-range $[1 - 2^{t-1}, 2^{t-1} - 1]$, leading to the numerical overflow. Therefore, to avoid the addition overflow, $Q(d)$ and $Q(g)$ should be set in smaller ranges, *i.e.,* $[\frac{1-2^{t-1}}{2}, \frac{2^{t-1}-1}{2}]$ and $[\frac{1-2^{t-1}}{3}, \frac{2^{t-1}-1}{3}]$.

2.2. The Range Scaling Mechanism

According to the anti-overflow conditions described previously, when $t = 8$, the range of $Q(g)$ is $[-42, 42]$, which is in the range of 7-bit and 6-bit. It makes the weight g can only be quantized to 6-bit (whose quantized range is $[-31, 31]$) for general quantization tools (*i.e.,* MNN and NCNN), causing a large information loss. In this work, a scaling factor α is specifically introduced to obtain the scaled integer-range $[-T_s, T_s] = [-T/\alpha, T/\alpha]$ where $[-T, T]$ is a wide integer range. As shown in Figure 2, the kernel weight of Conv1D can be quantized in the scaled 7-bit interval $[-42, 42]$ by setting $\alpha = 1.5$, thus the numerical representation becomes richer than the 6-bit integer-range $[-31, 31]$. Additionally, $\alpha = 1.0$ is set for 7-bit quantization of input activation of Conv1D, which is represented as $[-63, 63]$.

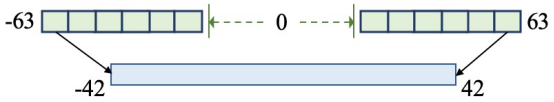

Fig. 2. Integer range scaling: the 7-bit integer-range [-63, 63] is scaled to [-42, 42] for weight quantization of Conv1D with quantized Winograd.

2.3. QAT with Quantization Noise Loss

A common practice for training domain-specific ASR models is to fine-tune existing large-scale pre-trained models. If the Post-training Quantization (PTQ) approach is adopted to quantize the fine-tuned ASR models, two issues should be tackled: i) extra steps including the calibration-set collection and PTQ should be considered; and ii) the accuracy will be decreased with PTQ, even the performance loss is small.

To simplify the entire optimization process and ensure the prediction accuracy after quantization, Quantization-Aware Training (QAT) [18, 19] should be introduced during fine-tuning. Here, Learned Step-size Quantization (LSQ) [19] is employed as the basic QAT approach. Based on LSQ, the tensor v (either weight or activation) is fake quantized with the quantization scale s and the scaled integer-range $[-T_s, T_s]$:

$$Q(v) = s \cdot round(clip(v/s, -T_s, T_s)) \tag{4}$$

The gradients can be formulated as follows:

$$\frac{\partial Q}{\partial v} = \begin{cases} 1, & (-T_s \leq v/s \leq T_s) \\ 0, & (otherwise) \end{cases} \tag{5}$$

$$\frac{\partial Q}{\partial s} = \begin{cases} -T_s, & (v/s < -T_s) \\ -v/s + round(v/s), & (-T_s \leq v/s \leq T_s) \\ T_s, & (v/s > T_s) \end{cases} \tag{6}$$

To enhance the effectiveness of LSQ, we further utilize the Mean Squared Error (MSE) between $q(v)$ and v as an auxiliary loss to distill the knowledge from high-precision values, which makes the quantization values close to original ones:

$$L_q = MSE(Q(v), v) = \frac{1}{N}(Q(v) - v)^2 \tag{7}$$

Overall, the MSE loss is multiplied with a penalty coefficient β, and added to the main task loss for fine-tuning. Hence, the overall loss of our model is $L = L_{asr} + \beta \cdot L_q$ where L_{asr} is the normal loss of the underlying ASR model.

To clarify the benefit of the auxiliary MSE loss, we firstly infer the gradients of L_q w.r.t v and s as follows:

$$\frac{\partial L_Q}{\partial v} = \frac{2}{N}(Q(v) - v)(\frac{\partial Q}{\partial v} - 1) \tag{8}$$

$$\frac{\partial L_Q}{\partial s} = \frac{2}{N}(Q(v) - v)\frac{\partial Q}{\partial s} \tag{9}$$

Combining Eq. 5 and Eq. 8, the gradient is less than 0 when $v/s < -T_s$, and is greater than 0 when $v/s > T_s$. Thus, the values of tensor v outside $[-sT_s, sT_s]$ can be updated towards a more concentrated distribution, friendly to quantization. Additionally, under the consideration of Eq. 6 and Eq. 9, the scale s will be updated to achieve the suitable quantization resolution for values inside or outside $[-sT_s, sT_s]$.

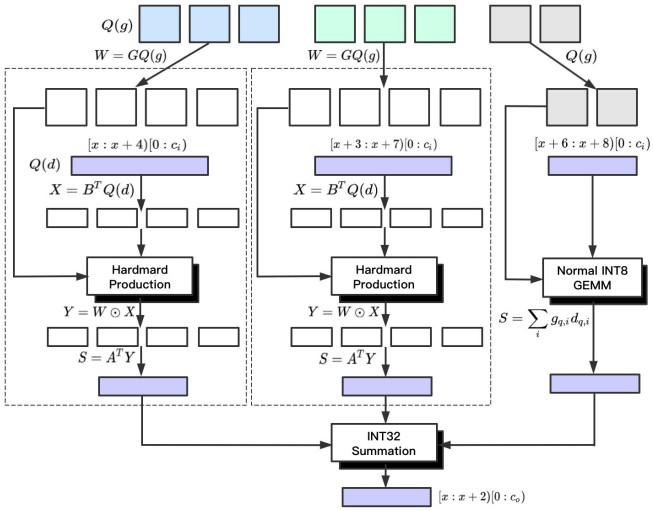

Fig. 3. INT8 Winograd Operator of Conv1D (with kernel size $k = 8$): the Conv1D is split as basic $F(2,3)$ Winograd convolutions and normal INT8 GEMMs.

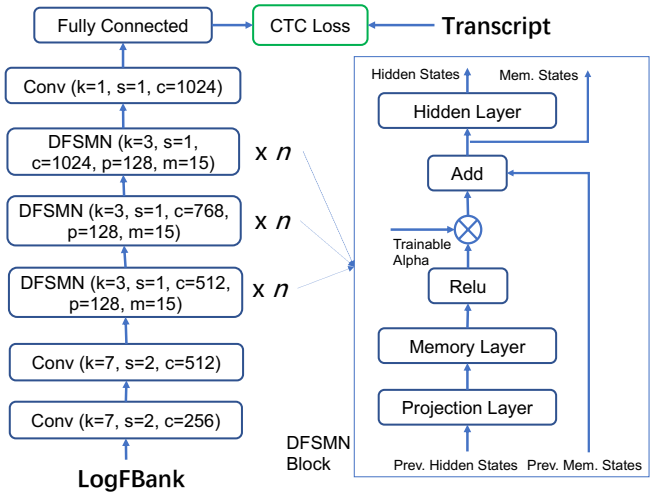

Fig. 4. ConvDFSMN architecture. Notations: k: kernel size, s: stride, c: hidden size, p: project size, m: memory size. The DFSMN block repetition factor is set as: n=2 or 4.

3. QUANTIZED WINOGRAD OPTIMIZATION

In our work, we deploy ASR models on mobile devices based on the open-source mobile inference framework MNN [22]. To further speed up the inference of quantized ASR models, an efficient quantized Winograd operator is designed.

When the Conv1D operation with kernel size ($k \geq 3$) and stride ($s = 1$) is quantized as INT8 Winograd representation. The activation and weights are expressed as 7-bit integers in the non-Winograd domain as depicted in Sect. 2.2. As shown in Figure 3, the designed INT8 Winograd operator for Conv1D with kernel size ($k \geq 3$) is split into several basic $F(2,3)$ formations, with the remaining operations as normal INT8 GEMMs (General Matrix Multiplication). These sub-flows are computed in parallel. Hence, the inference time is decreased with parallel computation. In each $F(2,3)$ flow, the sub-sampled 7-bit weight (size: 3×1) and the activation (size: 4×1) are firstly transformed to the Winograd domain. Next, the fully INT8 Hardmard production is conducted as a highly efficient execution method of Conv1D. Finally, the vector results of each $F(2,3)$ flow and GEMM flows are element-wisely summed as the final INT32 outputs.

Next, we give a theoretical analysis on speedup. As seen, the split numbers of $F(2,3)$ and normal INT8 GEMM are $\lfloor k/3 \rfloor$ and $k\%3$. To generate 2 outputs, each $F(2,3)$ flow consumes 4 multiplies. The normal INT8 GEMM consumes the number of multiplies proportional to the kernel size. Therefore, compared with the fully INT8 GEMM realization, the theoretical speedup of INT8 Winograd Conv1D with kernel size ($k \geq 3$) can be derived as follows:

$$speedup = \frac{2k}{4\lfloor k/3 \rfloor + 2(k\%3)} \tag{10}$$

The maximum speedup is 1.5 when k is divisible by 3.

Model	Parameters (M)	FLOPs (G)	Latency (ms)
ConvDFSMN-b	16.20 (6.52)	2.45 (0.98)	63.5 (28.1)
ConvDFSMN-s	12.37 (6.52)	1.87 (0.98)	53.4 (28.5)
Wav2letter	17.32 (6.52)	2.62 (0.98)	87.2 (28.5)

Table 1. Description of the three FP32 models. The input sequence length equals to 600. Contents in () are attributed to output layers, dominating the model size and computation.

4. EXPERIMENTS

4.1. Experimental Settings

Our framework is evaluated over three model architectures. The ConvDFSMN model is illustrated in Figure 4. After stem layers, a few Conv1D equipped DFSMN blocks [9] (with $n =$ 2 or 4 in Figure 4) are employed to capture the local contexts. We test two ConvDFSMN models in the experiments, namely ConvDFSMN-b (-base) with $n = 4$ and ConvDFSMN-s (-small) with $n = 2$. In addition, the simplified Wave2letter model [8] with almost the equivalent size to ConvDFSMN-b is designed for comparison. Details of these models are presented in Table 1. All these models are pre-trained on the EasyASR platform [26], and fine-tuned and tested on Aishell-1 [27]. The raw input features are the log of FBank features ($dimension = 80$). Since the vocabulary size of Aishell-1 is about $6K$, the output layer dominates the size and computation, which is sensitive to quantization. Hence, we keep the output layer as float32 (FP32) realization. We fine-tune all ASR models for 3000 steps with the mini-batch size as 128 and use the polynomial decay of the learning rate initialized with $lr = 0.005$. During fine-tuning, RSQ is applied with $\beta = 0.25$ set in the MSE loss function in Eq. 7.

For comparison, we also realize the KL algorithm in [28] for PTQ. The fine-tuned ASR models without RSQ are quan-

Model	Op. Type	Quant. Method	WER
ConvDFSMN-b*	FP32 Wino.	-	8.80%
	INT8 GEMM	PTQ	8.89%
	INT8 Wino.	PTQ	8.94%
ConvDFSMN-b	INT8 Wino.	RSQ†	8.76%
	INT8 Wino.	RSQ	8.73%
ConvDFSMN-s*	FP32 Wino.	-	10.54%
	INT8 GEMM	PTQ	10.62%
	INT8 Wino.	PTQ	10.68%
ConvDFSMN-s	INT8 Wino.	RSQ†	10.53%
	INT8 Wino.	RSQ	10.52%
Wav2letter*	FP32 Wino.	-	13.78%
	INT8 GEMM	PTQ	13.84%
	INT8 Wino.	PTQ	13.87%
Wav2letter	INT8 Wino.	RSQ†	13.73%
	INT8 Wino.	RSQ	13.71%

Table 2. Accuracy on Aishell-1 test set. Models with (*) are fine-tuned without RSQ. RSQ with (†) means the MSE loss is not used. Output layers are kept non-quantized. Conv1D with $k = 1$ or $s > 1$ is realized as normal INT8 GEMM.

Model	Op. Type	Latency	SR	SR‡
ConvDFSMN-b	FP32 Wino.	63.5	-	-
	INT8 GEMM	59.1	1.07×	1.14×
	INT8 Wino.	56.9	1.12×	1.23×
ConvDFSMN-s	FP32 Wino.	53.4	-	-
	INT8 GEMM	50.7	1.05×	1.12×
	INT8 Wino.	48.6	1.10×	1.24×
Wav2letter	FP32 Wino.	87.2	-	-
	INT8 GEMM	63.2	1.38×	1.69×
	INT8 Wino.	59.1	1.48×	1.92×

Table 3. Evaluation of end-to-end inference latency (unit: ms). Speedup ratio (SR) with (‡) means the one without the time cost of the output layer.

tized as normal INT8 GEMM or INT8 Winograd by PTQ. Meanwhile, the KL algorithm is applied for initializing the quantization scales of RSQ.

The inference latency of proposed models is profiled on the K20 Pro mobile phone with ARMv7 ISA (4 threads) and the batch size equals to 1. The input sequence length of acoustic features is fixed as 600, which equals to the audio signal of 6 seconds for real-time factor (RTF) measurement.

4.2. Experimental Results

We present the experimental results in detail. The three ASR models are also experimented with different operator realizations and quantization methods for ablation study.

Prediction Accuracy. We take the Word Error Rate (WER) as the evaluation metric. As demonstrated in Table 2, the WER of ConvDFSMN is lower than Wav2letter, which indicates the superiority of the ConvDFSMN model for mobile application. Even the ConvDFSMN-s model (with $12.37M$ parameters) achieves 10.5% WER on Aishell, which is lower than the result of Wav2letter (with $17.32M$ parameters).

In addition, the quantized ASR models fine-tuned with RSQ achieve better prediction performance than models quantized with PTQ. Thus, the advantages of integrating the RSQ method into fine-tuning are two-folds: i) the quantization flow is simplified by omitting the extra PTQ with the calibration-set; and ii) RSQ can act as the regularizer to improve the prediction performance after quantization.

Model	Kernel Shape	INT8 GEMM	INT8 Wino.
ConvDFSMN	(3, 128, 512)	0.601	0.536 (1.12×)
	(3, 128, 768)	0.898	0.788 (1.14×)
	(3, 128, 1024)	0.881	0.710 (1.24×)
	(15, 128, 128)	0.540	0.491 (1.10×)
Wav2letter	(9, 256, 512)	3.42	2.75 (1.24×)
	(13, 512, 512)	9.70	8.27 (1.17×)
	(15, 512, 512)	9.29	7.13 (1.30×)

Table 4. Latency (unit: ms) of Conv1D operators with INT8 GEMM or INT8 Winograd realization. The input sequence length of listed Conv1D operations is 150.

Inference Latency. In terms of inference latency, as shown in Table 3, compared with the normal INT8 GEMM, the neural networks gain further speedup with INT8 Winograd. To quantify the acceleration, we can see that the maximum speedup of end-to-end inference of Wav2letter is about $1.48\times = 87.2/59.1$, while the speedup ignoring the output layer is about $1.92\times = (87.2 - 28.5)/(59.1 - 28.5)$, indicating the valuable benefit of INT8 Winograd acceleration. We further analyze model accuracy and inference latency at the same time. As shown in Table 2, PTQ results with INT8 Winograd are less accurate but faster than INT8 GEMM, thus fine-tuning with RSQ is necessary to keep prediction performance while achieving acceleration with INT8 Winograd.

Analysis of INT8 Winograd Operators. To further verify the acceleration of INT8 Winograd operators in detail, Table 4 gives the runtime cost of Conv1D operators used in ConvDFSMN and Wav2letter. From the results, we can see that the INT8 Winograd has positive effects on the inference acceleration for all used kernel sizes. However, the actual speedup of INT8 Winograd is constrained by the ratio of computation to memory access. Hence, Conv1D with 15×1 kernel size in ConvDFSMN achieves $1.1\times$ speedup, of which the theoretical speedup is $1.5\times$. In the future, we will further study how to approach the theoretical speedup by better implementations of the inference process of Conv1D blocks.

5. CONCLUSION

To facilitate real-time voice interaction on mobile devices, an quantized Winograd optimization framework for ASR models is presented for better inference speed, which consists of range-scaled quantization (RSQ) training and INT8 Winograd realization. Based on the experimental results and theoretical analysis, we show that the designed Conv1D equipped ASR models can be effectively quantized and efficiently accelerated with the proposed method, indicating a significant potential for industrial applications.

6. REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[2] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Speech Audio Process.*, vol. 20, no. 1, pp. 14–22, 2012.

[3] J. Wang, B. Cao, P. Yu, L. Sun, W. Bao, and X. Zhu, "Deep learning towards mobile applications," in *ICDCS*, 2018.

[4] J. Cohen, "Embedded speech recognition applications in mobile phones: Status, trends, and challenges," in *ICASSP*, 2008.

[5] I. López-Espejo, A. M. Peinado, A. M. Gomez, and J. A. González, "Dual-channel VTS feature compensation for noise-robust speech recognition on mobile devices," *IET Signal Process.*, vol. 11, no. 1, pp. 17–25, 2017.

[6] J. Park, Y. Boo, I. Choi, S. Shin, and W. Sung, "Fully neural network based speech recognition on mobile and embedded devices," in *NeurIPS*, 2018, pp. 10 642–10 653.

[7] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *ISCA*, 2015.

[8] R. C., C. P., and G. S., "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint 1609.03193*, 2016.

[9] S. Zhang, M. Lei, Z. Yan, and L. Dai, "Deep-fsmn for large vocabulary continuous speech recognition," in *ICASSP*, 2018, pp. 5869–5873.

[10] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *ICASSP*, 2020, pp. 6124–6128.

[11] J. Rownicka, P. Bell, and S. Renals, "Multi-scale octave convolutions for robust speech recognition," in *ICASSP*, 2020, pp. 7019–7023.

[12] Z. Gao, S. Zhang, M. Lei1, and I. McLoughlin2, "San-m: Memory equipped self-attention for end-to-end speech recognition," *arXiv preprint 2006.01713*, 2020.

[13] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *ICASSP*, 2018.

[14] X. Chen, S. Zhang, D. Song, P. Ouyang, and S. Yin, "Transformer with bidirectional decoder for speech recognition," in *Interspeech*, 2020, pp. 1773–1777.

[15] S. Li, L. Li, Q. Hong, and L. Liu, "Improving transformer-based speech recognition with unsupervised pre-training and multi-task semantic knowledge learning," in *Interspeech*, 2020, pp. 5006–5010.

[16] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint 1806.08342*, 2018.

[17] M. Nagel, M. van Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *ICCV*, 2019.

[18] J. C. et al., "Pact: Parameterized clipping activation for quantized neural networks," in *ICLR*, 2018.

[19] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," in *ICLR*, 2020.

[20] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-BERT: hessian based ultra low precision quantization of BERT," in *AAAI*, 2020, pp. 8815–8821.

[21] G. inc., "Tensorflow lite: an open source deep learning framework for on-device inference," https://tensorflow.google.cn/lite, 2017.

[22] X. Jiang, H. Wang, Y. Chen, Z. Wu, L. Wang, B. Zou, Y. Yang, Z. Cui, Y. Cai, T. Yu, C. Lv, and Z. Wu, "Mnn: A universal and efficient inference engine," in *MLSys*, 2020.

[23] L. A. and G. S., "Fast algorithms for convolutional neural networks," in *CVPR*, 2016.

[24] J. Fernandez-Marques, P. N. Whatmough, A. Mundy, and M. Mattina, "Searching for winograd-aware quantized networks," in *MLSys*, 2020.

[25] G. Li, L. Liu, X. Wang, X. Ma, and X. Feng, "Lance: efficient low-precision quantized winograd convolution for neural networks based on graphics processing units," in *ICASSP*, 2020.

[26] M. Cheng, C. Wang, J. Huang, and X. Wang, "Weakly supervised construction of asr systems from massive video data," in *Interspeech*, 2021, p. 4533–4537.

[27] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *O-COCOSDA*, 2017, pp. 1–5.

[28] N. inc., "Tensorrt: programmable inference accelerator," https://developer.nvidia.com/tensorrt, 2020.