LOW-LATENCY LIGHTWEIGHT STREAMING SPEECH RECOGNITION WITH 8-BIT QUANTIZED SIMPLE GATED CONVOLUTIONAL NEURAL NETWORKS

Jinhwan Park, Xue Qian, Youngmin Jo and Wonyong Sung

Department of Electrical and Computer Engineering
Seoul National University
1 Gwanak-ro, Gwanak-gu, Seoul, 08826 Korea
{bnoo,cherieq_tx,edc8563,wysung}@snu.ac.kr

ABSTRACT

Automatic speech recognition (ASR) is very important for mobile devices. However, deep neural network-based ASR demands a large number of computations, while the memory bandwidth and battery capacity of mobile devices are limited. Server-based implementations are mostly employed, but this increases latency or privacy concerns. Efficient on-device ASR is the solution for these issues. In this paper, we propose a low-latency on-device speech recognition system with a simple gated convolutional network (SGCN). The SGCN shows a competitive recognition accuracy even with 1M parameters. In addition, SGCN is advantageous for parallelization which enables efficient cache utilization. 8-bit quantization is applied to reduce the memory size and computation time. The proposed system features online recognition fulfilling the 0.4s latency limit and operates with the real-time factor of 0.2 using only a single 900MHz CPU core. The system occupying 1.2MB memory footprint shows 19.75% word error rate (WER) with greedy decoding.

Index Terms— On-device speech recognition, Convolutional neural networks

1. INTRODUCTION

Deep neural network-based acoustic models have greatly improved the accuracy of automatic speech recognition (ASR) [1, 2, 3]. Neural network-based ASR requires a lot of computations, which demands high memory bandwidth and power consumption for the real-time operation. Server-based implementations are mostly employed to address these issues. However, data transfer between server and edge devices can result in high latency and privacy issues. Also, the connectivity to the network must be guaranteed to use the server-based service. Therefore, it is important to develop an on-device ASR system that can operate in real-time even with limited computing resources [4].

Recurrent neural networks (RNNs), such as an long short-term memory (LSTM) [5] or gated recurrent unit (GRU) [6], have been widely used for acoustic modeling. RNNs employ feedback from the output of the previous time steps, which is advantageous for sequence learning, such as speech recognition. However, the computation of a single sequence RNN is difficult to parallelize because of the temporal dependency due to the feedback structure. This results in ineffective cache utilization since the weights cannot be reused for multiple time-steps. This is especially serious for on-device applications where the cache capacity is often less than the parameter size of neural network models.

Recently, parallelizable models such as convolutional neural networks (CNNs) [7] or quasi-RNNs (QRNNs) [8] are actively studied for the application to sequential tasks. For the speech recognition, fully convolutional networks were trained with the connectionist temporal classification (CTC) and showed a lower word error rate (WER) than RNN-based models [9, 10]. However, these structures demand a very large parameter size because good acoustic modeling needs to observe a long time span. Depthwise convolutions can be used to reduce the complexity of a CNN or QRNN. The models with depthwise convolutions are successfully applied to CTC- [11, 12] and attention-based models [13].

In this paper, we propose the implementation of an on-device streaming end-to-end speech recognition system with the simple gated convolutional neural network (SGCN) [12]. This design is intended for low-cost ARM CPU based implementations supporting SIMD (Single Instruction Multiple Data) instructions. SGCN is advantageous compared to other architectures when the number of parameters is extremely low. The model is trained with CTC [14], which is suitable for online decoding. We applied a training method with symmetrical noise injection to weights for good regularization. 8-bit quantization is employed to further reduce the memory bandwidth and use SIMD instructions. The proposed system fea-

This work was in part supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2018R1A2A1A05079504). This work was also supported by Samsung Research. This work was also supported in part by AIR Lab (AI Research Lab) in Hyundai Motor Company through HMC-SNU AI Consortium Fund. This work was also partly supported by the Google AI Focused Research Awards Program awarded to Wonyong Sung.

tures streaming inference, where the intermediate outputs are available before the end of input utterance is given. We have developed a 1.2MB end-to-end ASR model that operates in 0.2 real-time factor (RTF) (five times the speed of real-time) with only a single 900MHz CPU core. The proposed system shows the WER lower than 20% on WSJ eval92 without any language model.

2. SIMPLE GATED CONVOLUTIONAL NEURAL NETWORKS

2.1. Structure of simple gated convolutional neural networks

The gated convolutional neural network (GCN) is a non-recurrent network architecture which has been successfully applied to sequential tasks, such as language modeling [15], translation [7] and speech recognition [9]. GCN employs a gating mechanism similar to the output gate of LSTM RNN. When the input $\mathbf{x}_t \in \mathbb{R}^N$ is given, the output of GCN $\mathbf{h}_t$ is computed as follows:

$$\mathbf{h}_t = f(\sum_{i=t-W+1+d}^{t+d} \mathbf{V}_i\mathbf{x}_i + \mathbf{b}) \odot \sigma(\sum_{i=t-W+1+d}^{t+d} \mathbf{U}_i\mathbf{x}_i + \mathbf{c}), \tag{1}$$

where $\mathbf{V}_i, \mathbf{U}_i \in \mathbb{R}^{N\times N}$ and $\mathbf{b}, \mathbf{c} \in \mathbb{R}^N$ are trainable variables. σ is a sigmoid function for gating. We used ReLU for the activation function f. W and d denote the width and delay of GCN. Computation of $\mathbf{h}_t$ does not contain any dependencies on $\mathbf{h}_{t-1}$, and the outputs for multiple time-steps can be computed simultaneously.

GCNs can only consider a finite range of context that is limited by the filter width. To apply GCNs to speech recognition, a filter wider than 15 time-steps is often used. The number of parameters and computations for GCN is approximately $2WN^2$ and increases proportionally to the width W. This increases the parameter size of GCNs too much for on-device speech recognition.

Simple gated convolutional neural networks (SGCNs) employ depthwise convolutions to reduce the parameter size [12]. In SGCN, the width of 1-D convolutions in GCN is reduced to 1. Instead, depthwise convolutions are applied to increase the length of the input context. The depthwise convolution in SGCN consults K neighboring features to consider the correlation between them. The output of SGCN is computed as follows:

$$\begin{aligned} x'_{t,k} &= \sum_{j=0}^{K-1} \sum_{i=t-W+1+d}^{t+d} w_{i,j} \cdot x_{i,k+j}, \\ \mathbf{h}_t &= f(\mathbf{V}\mathbf{x}_t' + \mathbf{b}) \odot \sigma(\mathbf{U}\mathbf{x}_t{}' + \mathbf{c}). \end{aligned} \tag{2}$$

The number of parameters for the convolution and depthwise convolution are $2N^2$ and KWN, respectively. In general, N is a few hundreds and K is about 5. Therefore, increasing the width of SGCN has a negligible impact on the total number of parameters.

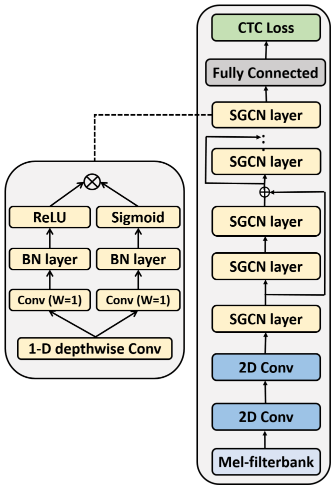

Fig. 1. SGCN architecture for acoustic modeling.

Figure 1 describes the whole structure of SGCN for acoustic modeling (AM). We applied two levels of 2-D convolutions to the input features in frequency- and time-axis. The output of the first layer was max-pooled with the factor of 2 for down-sampling. We used a residual connection for every two SGCN layers.

2.2. Multi-time-step parallelization

The multi-time-step parallelization technique computes multiple output samples of a DNN, by which the number of memory accesses can be dramatically reduced. The multi-time-step parallelization merges matrix-vector multiplications for the multiple inputs into a single matrix-matrix multiplication. For example, $\mathbf{V}\mathbf{x}_t'$ in Eq. (2) can be computed for T time steps as follows.

$$[\mathbf{x}_1', \mathbf{x}_2', ..., \mathbf{x}_T'] = \mathbf{V}[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T] \tag{3}$$

With a single fetch from DRAM, the weights can be reused for the T time-step inputs. The multi-time-step parallelization cannot be applied to the recurrent path in RNN, where the computation of the previous time-step output must be completed before the current time-step computation. If the cache size is smaller than the parameter size, the number of DRAM accesses can be reduced to 1/T by applying the multi-time-step parallelization.

3. TRAINING CTC AM WITH SGCN

3.1. Symmetrical weight noise injection

We used the CTC (Connectionist Temporal Classification) loss [14] for end-to-end SGCN model training. To obtain

Table 1. WER (%) of SGCN trained with the symmetrical weight noise injection. SN denotes the symmetrical noise injection. The models are trained on WSJ si-284.

Model	**Params.**	**WER**
12x190 K=5, W=11	1.09M	21.66
12x190 K=5, W=11 + SN	1.09M	19.90
12x300 K=5, W=11	2.24M	18.30
12x300 K=5, W=11 + SN	2.24M	16.87
Jasper 10x3 [10]	201M	13.3

Table 2. WER (%) of SGCN trained with the symmetrical weight noise injection. Trained on WSJ si-all (147 hours).

Model	**Params.**	**WER**
12x190 K=5, W=11	1.09M	18.18
12x190 K=5, W=11 + SN	1.09M	16.76
12x300 K=5, W=11	2.24M	15.30
12x300 K=5, W=11 + SN	2.24M	12.74

higher recognition accuracy, we applied the symmetrical weight noise injection during training. Training with noise injection helps to find wide local minima in the loss surface and avoid overfitting [16]. Conventional stochastic gradient descent updates the parameters of the model $\mathbf{w}$ as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{B}\sum_{i=1}^{B} \nabla L_i(\mathbf{w}_t), \tag{4}$$

where B is the batch size. For the symmetrical weight noise injection, we used $\widetilde{\mathbf{w}}_{t+} = \mathbf{w}_t + \alpha \mathbf{n}_t$, and $\widetilde{\mathbf{w}}_{t-} = \mathbf{w}_t - \alpha \mathbf{n}_t$ for training as Equation 5. Weight noise $\mathbf{n}_t$ is a uniformly distributed random vector. The magnitude of the noise is determined from the standard deviation of the weight. We used 0.05 for the scale factor α.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{2B}\sum_{i=1}^{B} \nabla (L_i(\widetilde{\mathbf{w}}_{t+}) + L_i(\widetilde{\mathbf{w}}_{t-})) \tag{5}$$

3.2. 8-bit quantization

To use 8-bit SIMD instructions and lower the memory bandwidth, we quantized the weights and activations of the SGCN model. We used TensorFlow Lite [17] for quantization.

We applied retraining-based quantization to reduce the accuracy drop of the quantized model. First, we trained SGCN models in the floating-point domain. After the floating-point model is converged, the model is retrained with quantized weights and activations. The minimum and maximum values for quantization are obtained during retraining. For retraining, we used the learning rate schedule identical to that used for training the floating-point model.

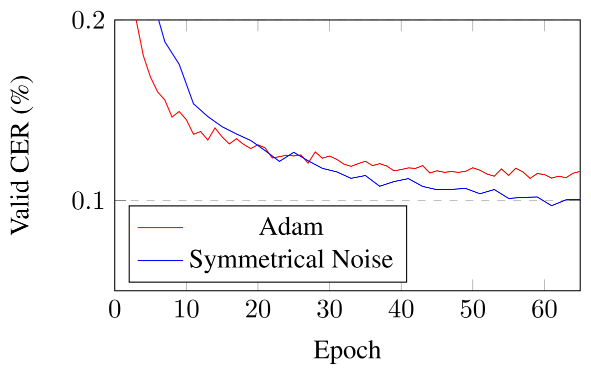

Fig. 2. Valid character error rate (CER) curve of the SGCN trained on WSJ si-284.

Table 3. WER (%) evaluated on the WSJ eval92 test set. D.w. conv. denotes a depthwise convolution.

Model	**Params.**	**WER**
12x190 K=5, W=11, 1200ms latency	1.09M	16.76
12x190 K=5, W=11, 200ms latency	1.09M	18.46
12x190 K=3, W=11, 200ms latency	1.04M	19.28
12x190 K=5, W=9, 200ms latency	1.07M	19.60
12x180 K=5, W=11, 200ms latency	1.00M	19.77
4x160 uni. LSTM [12]	0.96M	31.2
4x160 uni. LSTM + D.w. conv. [12]	0.97M	24.1
12x300 K=5, W=11, 1200ms latency	2.24M	12.74

4. EXPERIMENTAL RESULTS

4.1. Experimental setting

The Wall Street Journal (WSJ) Corpus [18] was used to train the models. We used two subsets of WSJ Corpus for training, one is si-284, which is the standard subset for training, and the other is si-all which includes all the speaker-independent data in the corpus. The amount of data in si-284 and si-all are 81 hours and 147 hours, respectively. The utterances with verbalized punctuations are excluded from training. We used a 40-dimensional log Mel frequency filter-bank with delta and delta-delta for the input features.

All the SGCN models in this paper were trained using the following hyperparameters. For training, a learning rate of 3e-3 was chosen. If the validation error did not decrease for 8 epochs, the learning rate was decayed by 0.2. After the learning rate decayed six times, the training was ended. The Adam optimizer [19] was employed for training.

Table 4. Computation time measured on 900MHz ARM CPU. Only a single core is used for evaluation.

Model	Weight	Activation	Model Size	RTF
12x190 SGCN K=5, W=11	32-bit float	32-bit float	4.30MB	0.469
12x190 SGCN K=5, W=11	8-bit int	32-bit float	1.16MB	0.520
12x190 SGCN K=5, W=11	8-bit int	8-bit int	1.16MB	0.194
12x190 SGCN K=3, W=11	8-bit int	8-bit int	1.08MB	0.191
12x190 SGCN K=5, W=9	8-bit int	8-bit int	1.12MB	0.183
12x180 SGCN K=5, W=11	8-bit int	8-bit int	1.03MB	0.174

Table 5. WER (%) of the SGCN after quantization.

Model	WER
12x190 K=5, W=11, 200ms latency	18.46
12x190 K=5, W=11, 200ms latency, quantized	19.75

4.2. Results on WSJ eval92

The results of training with and without the symmetrical noise injection are shown in Table 1 and 2. The word error rate (WER) is evaluated on WSJ eval92. 12x190 model denotes that 12 SGCN layers with $D = 190$ are stacked. The symmetrical noise injection consistently improved the performance regardless of the size of the model or training data. For comparison, we also showed the WER of recent convolutional CTC model, Jasper [10]. Considering the number of parameters, the SGCN model shows competitive accuracy.

Table 3 shows the WERs of SGCN models with various configurations. The models are trained on WSJ si-all. The symmetrical noise injection is applied for training all the models. For the 200ms latency model, we used $d = 5$ for the last 2 SGCN layers and $d = 0$ for the others. All the SGCN layers in the 1,200ms use $d = 5$. When we decrease K, W, or dimension of layers from the 12x190 K=5, W=11 SGCN model, WER drops accordingly. For comparison, WERs of LSTMs with a comparable model size are shown. If we increase the model size to 2.24M, the WER decreases from 16.76 % to 12.74%. This suggests that the SGCN model can scale-up to a large one.

4.3. Implementation on an embedded system

For a streaming implementation, we used the 12x190 K=5, W=11, 200ms latency model. The WER of the model after quantization is reported in Table 5. Table 4 reports the computation time and memory size of the SGCN AM. The computation time is measured on Raspberry Pi 2 model B. The system has 900MHz ARM Cortex-A7 CPU and 1GB DRAM. The CPU has 256KB L2 cache. For the results in Table 4, 20 frames, which corresponds to 200 ms, of the inputs are computed at a time. RTF can be reduced if the number of time steps for parallelization increases because of memory access reduction. RTF with different parallelization time-steps is shown in Table 6.

Table 6. RTF with the different number of computation steps.

Time steps	200 ms	400 ms	3000 ms
RTF	0.194	0.188	0.179

Table 7. Percentage of computation time for each type of operation.

Operation Type	Percentage	Cumulative Sum
Conv.	70.89%	70.89%
Depthwise Conv.	21.06%	91.95%
Dense	0.46%	92.41%
Etc. (Activation, ...)	7.59%	100%

Table 7 shows the computation time for each block. Tensorflow Lite profiler is used to measure the computation time. As we analyzed in Section 2.1, the most time-consuming operation is the convolutional operation. 71% of the time is dedicated to the convolutions, while the depthwise convolutions occupy 21% of the total computation time.

5. CONCLUDING REMARKS

We have developed an on-device streaming ASR system using the Simple Gated ConvNet, instead of widely used RNN models. We reduce the memory bandwidth requirements by parameter quantization and also employing the multi-time-step parallelization technique. The developed model operates in 0.2 RTF with a 900MHz CPU using only 1.2MB memory footprint for parameters. The system can be used for many applications where the computation and power budgets are very limited, including keyword spotting and always-on speech recognition.

6. REFERENCES

[1] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "RWTH ASR systems for librispeech: Hybrid vs attention," 2019.

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.

[3] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018, pp. 4774–4778.

[4] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[5] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Syntax, Semantics and Structure in Statistical Translation*, p. 103, 2014.

[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin, "Convolutional sequence to sequence learning," in *International Conference on Machine Learning (ICML)*, 2017, pp. 1243–1252.

[8] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher, "Quasi-recurrent neural networks," *International Conference on Learning Representations (ICLR)*, 2017.

[9] Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert, "Letter-based speech recognition with gated ConvNets," *arXiv preprint arXiv:1712.09444*, 2017.

[10] 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," in *Proceedings of Interspeech*, 2019.

[11] Jinhwan Park, Yoonho Boo, Iksoo Choi, Sungho Shin, and Wonyong Sung, "Fully neural network based speech recognition on mobile and embedded devices," in *Advances in Neural Information Processing Systems*, 2018, pp. 10620–10630.

[12] Lukas Lee, Jinhwan Park, and Wonyong Sung, "Simple gated convnet for small footprint acoustic modeling," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019.

[13] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," in *Proceedings of Interspeech*, 2019.

[14] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 369–376.

[15] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier, "Language modeling with gated convolutional networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 933–941.

[16] Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li, "SmoothOut: Smoothing out sharp minima to improve generalization in deep learning," *arXiv preprint arXiv:1805.07898*, 2018.

[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*, 2018, pp. 2704–2713.

[18] Douglas B Paul and Janet M Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

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