

Exploration of Efficient End-to-End Acoustic Models for On-device Speech Recognition

Wonyong Sung
School of Electrical and Computer Engineering
Seoul National University



Contents

1. On-device speech recognition and DRAM bandwidth bottleneck problem
2. Muti-time step parallel and linear recurrent RNNs
3. QRNN+1-d time-depth convolution based models
4. Gated ConvNet and Simple Gated ConvNet based models

Fully Neural Network Based Speech Recognition on Mobile and Embedded Devices

Jinhwan Park
Seoul National University
bnoo@snu.ac.kr

Yoonho Boo
Seoul National University
dngsh@snu.ac.kr

Iksoo Choi
Seoul National University
akacis@snu.ac.kr

Sungho Shin
Seoul National University
ssh9919@snu.ac.kr

Wonyong Sung
Seoul National University
vysung@snu.ac.kr

Park, Jinhwan, et al.
"Fully neural network based speech recognition on mobile and embedded devices." *Advances in neural information processing systems*. 2018.

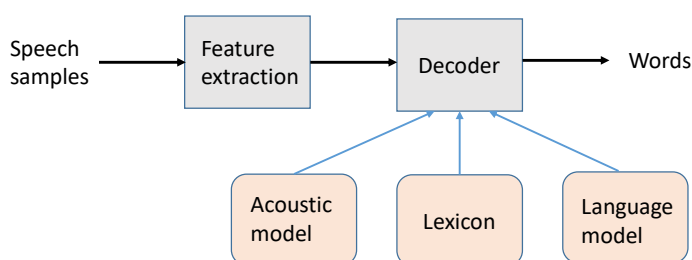


Why on-device speech recognition?

- Most ASR today is server-based
 - Speech (or feature extraction on device) → communication channel -> Remote ASR server, exploiting computation efficient GPUs
 - Problems:
 - Delay of response, privacy issues
 - Large cost of server operation (Edge Computing is needed)
- On-device speech recognition can be a solution for these problems.
 - Processing power and battery life issues at devices



Speech Recognition



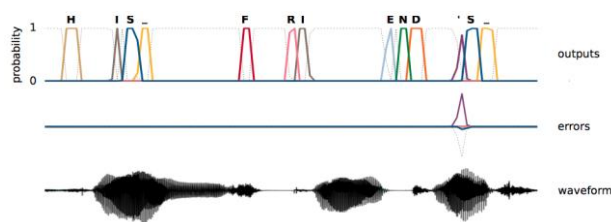
- Frame-level triphone state + WFST based method
- End-to-end approaches – easy decoding and large corpus
 - CTC + RNN LM
 - Neural transducers
 - LAS (Listen Attend & Spell)



CTC algorithm

■ Connectionist Temporal Classification

- Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks, 2006 ICML.
- Feature : frame sequence e.g. every 10 msec
- Label : text sequence: {a, b, c, : : : , z, space, apostrophe, blank}
- To fill the length difference gap between the input and output, employ the blank labels



(Graves and Jaitly, ICML 2014)

CTC algorithm and decoding

- CTC is a sequence modeling algorithm
 - It needs to observe fairly long frames (not a single frame)
 - LSTM RNN is usually used.
- The CTC output can be decoded simply by removing blank and repetition labels, which is so called the greed decoding. But, the output is usually post-processed using a language model.
- The language model:
 - Character-level: the input and output are simple but the performance is lower.
 - Word-level: the input and output (softmax layer) complexity is proportional to the vocabulary size.
 - Word-piece model: no OOV, limited complexity

Contents

- 1. On-device speech recognition and DRAM bandwidth bottleneck problem**
2. Multi-time step parallel and linear recurrent RNNs
3. QRNN+1-d time-depth convolution based models
4. Gated ConvNet and Simple Gated ConvNet based models

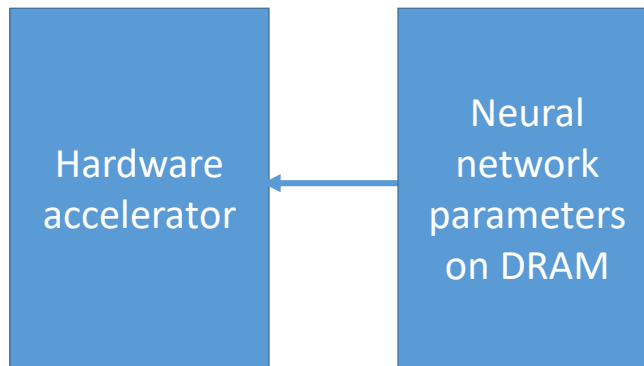


On-device implementation of RNN acoustic models: Memory bound or Arithmetic bound?

- RNN parameter size is 10 million ~ 100 million
- The frame rate is usually about 20 msec
- How many arithmetic operations?
 - $10M \times 50 \times 2$ (for add, mult) ≈ 1 Giga op./sec (~ 10 Giga)
 - 1GHZ, SIMD arithmetic (128 bit data-path for ARM): 4~16 operations (float, half-precision, 8bit) for one instruction (4~16Gops/sec)
- How about DRAM accesses? (Energy consumption?)
 - Parameters are used only once and larger than cache size (2MB for an ARM CPU)
 - 0.5G ~ 5GByte/sec memory accesses
 - 1DRAM access takes about 50ns for fetching 256bits (32Bytes)
 - The effective BW is less than 1GB/sec



Memory access bottleneck is not solved by quipping many multiply-adders



How GPUs solve the memory bottleneck?

- Batch processing in GPU – popular batch size of 32~256, and one memory fetch is reused in the batch
- On-device speech recognition – the batch size is 1.

Memory bandwidth reduction for on-device neural networks

1. Model compression

- Fixed-point optimization
 - 32bit floating-point -> 8bit fixed-point -> (future 4bit ?) (x4 ~ x8)
- Pruning, SVD transformation, and etc.
 - Reducing the number of non-zero parameters.
 - It may need decompression HW for non-structured pruning

2. Parallelization – computing multiple output samples at a time (multi-time-step parallelization)

- Usually, parallelization technique was of interests in GPU based computing to utilize large scale hardware
- We employ this technique to reduce the memory bandwidth



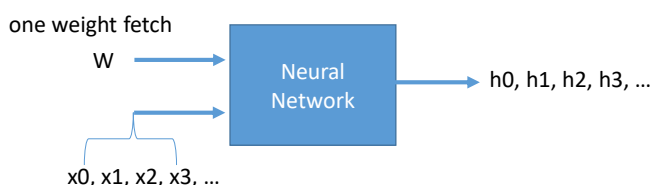
Contents

1. On-device speech recognition and DRAM bandwidth bottleneck problem
- 2. Multi-time step parallel and linear recurrent RNNs**
3. QRNN+1-d time-depth convolution based models
4. Gated ConvNet and Simple Gated ConvNet based models



Multi time-step parallel processing

- We fetch one parameter and use it for multiple (B) inferences (time-steps), and the number of DRAM access becomes inversely proportional to B
- Multi-time-step parallel processing is obvious for feed-forward neural networks (fully connected DNNs, CNNs)



Multi time-step parallel processing for LSTM RNN

- LSTM RNN is most widely used for end-to-end acoustic modeling
- Single thread LSTM RNN and GRU do not permit multi time-step processing because of the dependency problem. We need to compute the output one by one, sequentially.

Non-linear dependency

$$\begin{aligned}
 f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\
 i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\
 o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\
 \tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c), \\
 c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\
 h_t &= o_t \odot \tanh(c_t).
 \end{aligned}$$

Neural networks supporting multi-time-step parallelization

- Linear recurrent RNNs (no U matrices)
 - QRNNs (Quasi-RNN)
 - SRU (Simple Recurrent Unit)

$$\hat{\mathbf{x}}_t = \mathbf{W} \mathbf{x}_t,$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{b}_f),$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{b}_r),$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \hat{\mathbf{x}}_t,$$

$$\mathbf{h}_t = \mathbf{r}_t \odot \tanh(\mathbf{c}_t) + (1 - \mathbf{r}_t) \odot \mathbf{x}_t.$$

linear recurrent

No dependency on \mathbf{h}_{t-1}



The quasi RNN (QRNN)

- Bradbury, James, et al. "Quasi-Recurrent Neural Networks." (2016).
- Simplification of the LSTM RNN without output feedback
 - QRNN just uses memory cell feedback
 - "quasi" RNN
 - The algorithm can consult k consecutive input samples
 - $\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-k+1}$
 - The simplest form of QRNN

$$\hat{\mathbf{x}}_t = \tanh(\mathbf{W}_z \mathbf{x}_t + \mathbf{b}_z),$$

$$[\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t] = \sigma([\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o] \mathbf{x}_t + [\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o]),$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{x}}_t,$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t + (1 - \mathbf{o}_t) \odot \mathbf{x}_t.$$



Linear recurrent neural networks

▪ QRNN:

$$\begin{aligned}\hat{x}_t &= W x_t, \\ f_t &= \sigma(W_f x_t + b_f), \\ r_t &= \sigma(W_r x_t + b_r), \\ c_t &= f_t \odot c_{t-1} + (1 - f_t) \odot \hat{x}_t, \\ h_t &= r_t \odot \tanh(c_t) + (1 - r_t) \odot x_t.\end{aligned}$$

▪ SRU

SRU

$$\begin{aligned}\hat{x}_t &= W_z x_t + b_z, \\ [f_t, o_t] &= \sigma([W_f, W_o]x_t + [b_f, b_o]), \\ c_t &= f_t \odot c_{t-1} + (1 - f_t) \odot \hat{x}_t, \\ h_t &= o_t \odot \tanh(c_t) + (1 - o_t) \odot x_t\end{aligned}$$

i-SRU

$$\begin{aligned}\hat{x}_t &= \tanh(W_z x_t + b_z), \\ [f_t, i_t, o_t] &= \sigma([W_f, W_i, W_o]x_t + [b_f, b_i, b_o]), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \hat{x}_t, \\ h_t &= o_t \odot c_t + (1 - o_t) \odot x_t\end{aligned}\quad (1)$$

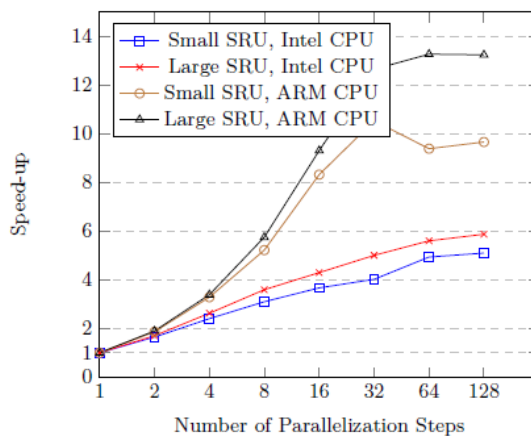
SRU computation with BLAS

$$\begin{aligned}\hat{x}_t &= \tanh(W_z x_t + b_z), \\ [f_t, i_t, o_t] &= \sigma([W_f, W_i, W_o]x_t + [b_f, b_i, b_o]), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \hat{x}_t, \\ h_t &= o_t \odot c_t + (1 - o_t) \odot x_t\end{aligned}\quad \left. \begin{array}{l} \text{time consuming part} \\ \text{element-wise operations} \end{array} \right\}$$

$$\begin{bmatrix} x_0 & x_1 & \dots & x_{T'} \\ i_0 & i_1 & \dots & i_{T'} \\ f_0 & f_1 & \dots & f_{T'} \\ o_0 & o_1 & \dots & o_{T'} \end{bmatrix} = \begin{pmatrix} W_z \\ W_i \\ W_f \\ W_o \end{pmatrix} (x_0 \ x_1 \ \dots \ x_{T'})$$

- The matrix-vector multiplication becomes matrix-matrix multiplications (BLAS can be used)

Speed-up on Intel and ARM CPU



- Larger speed-up is achieved in ARM CPU (poorer memory system)
- Large SRU width 1024, small SRU with 512



Contents

1. On-device speech recognition and DRAM bandwidth bottleneck problem
2. Multi-time step parallel and linear recurrent RNNs
3. QRNN+1-d time-depth convolution based models
4. Gated ConvNet and Simple Gated ConvNet based models



Performance (accuracy) issues

- CTC is a sequence recognition algorithm, and it needs to observe a long span of time
 - RNN is favored
- LSTM and QRNN
 - LSTM RNN provides both long-term (c_{t-1}) and short-term or immediate (h_{t-1}) feedback
 - But, QRNN has only long-term feedback (no immediate feed-back)
- In experiments, QRNN shows significantly lower performance than LSTM RNN.



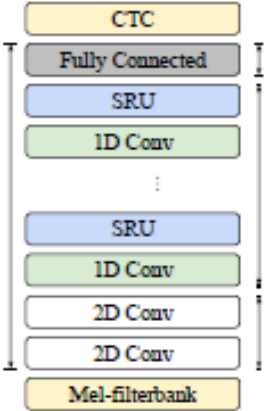
1-dimensional convolution


- Increasing the depth of time-context improves the accuracy even for phoneme recognition based acoustic models
- Linear RNN: how can we augment more recent information to compensate for the lack of immediate output feedback?
 - Increasing the input context length by providing 1-d time domain convolution
 - The number of parameters is very small
 - Past observation of up to 15 steps was very useful.
 - Future observation is good for performance but increases the delay



Acoustic modeling

- DeepSpeech2 like structure
 - CNN layers (with time-domain pooling)
 - RNN layers
 - Fully connected layer
- We add 1-D convolution at the SRU input
 - The convolution spans in time.
 - Each convolution does not share the weights
 - The optimum length is found to be 15
- CTC Grapheme (Character) based model
- WSJ 284 for fast evaluation
- Compare SRU, i-SRU, and LSTM






RNN performances with WSJ Si-284

Table 1: WER and CER in percentage on WSJ eval92 test set. Decoding is conducted with RNN CLM and HCLM.

Model	Params.	Greedy		CLM		HCLM	
		CER	WER	CER	WER	CER	WER
6x800 SRU	10.62M	26.94	82.56	13.24	29.68	7.94	15.41
6x700 i-SRU	10.92M	12.70	45.22	7.04	18.90	4.90	12.27
6x800 SRU, 1-D conv	10.69M	6.06	22.16	3.48	9.53	1.97	4.90
6x700 i-SRU, 1-D conv	10.98M	5.26	19.07	2.70	7.30	2.01	4.90
6x1000 i-SRU, proj, 1-D conv	14.14M	5.85	21.60	3.00	7.80	2.27	5.17
4x600 LSTM	10.85M	7.29	24.88	5.35	14.27	3.70	8.75
4x600 LSTM, 1-D conv	10.88M	6.95	23.57	5.80	15.22	3.10	7.01
4x840 LSTM, proj, 1-D conv	12.01M	7.78	26.80	4.88	12.26	3.36	7.60
6x300 Gated ConvNet	16.38M	8.02	28.65	5.13	13.82	2.98	6.74
4x550 GILR-LSTM	11.34M	8.60	31.99	4.86	13.60	2.66	6.35
4x550 GILR-LSTM, 1-D conv	11.37M	7.15	26.06	4.44	11.92	2.38	5.45
<i>bidirectional models</i>							
6x400 i-SRU, 1-D conv	11.52M	4.90	17.30	2.94	7.90	1.97	4.87
4x350 LSTM	10.70M	5.88	20.17	3.46	9.41	2.57	5.89



Past and future context

- SRU (or i-SRU) augmented with 1-d conv reduce the WER a lot (greedy decoding: 45% -> 19%)
- Augmenting 1-d conv to LSTM does not give much benefit (WER 24.88-> 23.57%)
- The effect of observation window for 1-d convolution

Table 2: Comparison of the model with non-causal and causal 1-D convolutions. 1-D conv $(-a, b)$ uses a past and b future time-steps to compute the output of the current time step.

Model	Greedy		CLM		HCLM	
	CER	WER	CER	WER	CER	WER
6x700 i-SRU, 1-D conv $(-7, 7)$	5.26	19.07	2.70	7.30	2.01	4.90
6x700 i-SRU, 1-D conv $(-14, 0)$	5.70	20.18	3.12	8.47	2.30	5.32
6x700 i-SRU, 1-D conv $(-7, 0)$	6.10	21.96	2.99	7.69	2.35	5.55
6x700 i-SRU, 1-D conv $(-7, 1)$	6.02	21.40	3.09	8.04	2.39	5.57
6x700 i-SRU, 1-D conv $(-7, 2)$	6.32	22.80	3.08	7.80	2.26	5.28



Word-piece model

- Word-piece model employs the (frequently occurring) words, sub-words, and characters as the labels
- The non-blank label generation frequency drops compared to grapheme based AM.
- Can operate the AM at a reduced frequency, directly lowering the complexity of AM and LM
- No OOV (out of vocabulary problem)
- Softmax layer is not as complex as that of word model
- Word piece 500, word piece 1000
- Training of word piece models needs a lot more acoustic data

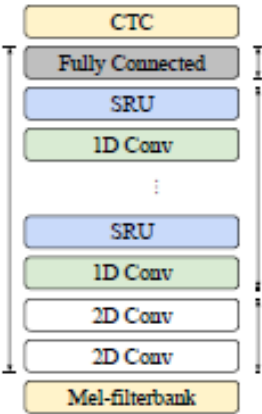


Down sampled word piece models

- We can reduce the frame rates by inserting time-domain pooling at the CNN layers or RNN layers

Table 4: Comparison of WER and CER according to downsampling in the wordpiece

Model	Greedy		CLM	
	CER	WER	CER	WER
x 2 in conv. layer	7.02	18.95	6.05	10.0
x 4 in conv. layer	8.05	20.24	6.55	11.0
x 2 in conv. layer, x 2 in recurrent layer	7.37	17.95	6.00	10.0
x 4 in conv. layer, x 2 in recurrent layer	10.30	25.58	7.83	13.0



Word piece model performance

Libri speech

Table 7: WER on Librispeech *test-clean* and *test-other*. The models are trained on all the LibriSpeech train set (960 hours).

Model	Params.	test-clean	test-other	LM type
6x700 i-SRU, 1-D conv	12M	9.02	23.60	RNN LM
12x1000 i-SRU, 1-D conv	36M	5.73	15.96	RNN LM
Gated ConvNet [21]	208M	4.8	14.5	4-gram LM
5-conv + 4x1024 bidirectional GRU [31]	75M	5.4	14.7	4-gram LM
Encoder-decoder [32]	150M	3.82	12.76	RNN LM

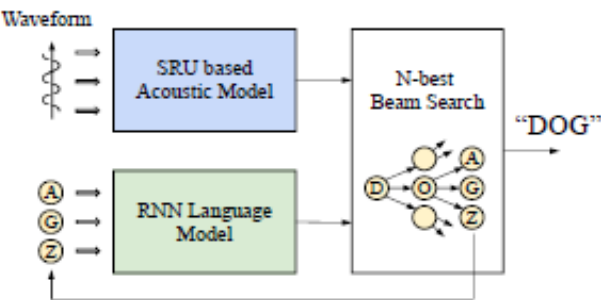
Table 8: Execution time of SRU-AM for 1 second of speech according to the number of parallelization steps.

Parallelization Step	1	2	4	8	16	32
Computation time	1.2129	0.6098	0.3065	0.2064	0.1524	0.1174

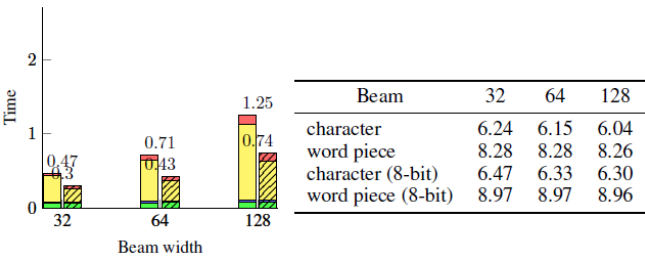
Decoding

- Decoding with RNN based CLM (Character-level LM)
- Decoding with RNN based hierarchical CLM

$$Q(y) = \log(P_{\text{TRC}}(y|x_{1:T})) + \alpha \log(P_{\text{LM}}(y)) + \beta|y|$$



Processing time with LM based beam-search decoding



(b) Word piece-level model.

(c) WER with different beam width.

Contents

1. On-device speech recognition and DRAM bandwidth bottleneck problem
2. Multi-time step parallel and linear recurrent RNNs
3. QRNN+1-d time-depth convolution based models
- 4. Gated ConvNet and Simple Gated ConvNet based models**



Gated ConvNet

- Dauphin, Yann N., et al. "Language Modeling with Gated Convolutional Networks." *International Conference on Machine Learning*. 2017.
- Multi-layered convolution with Gating mechanism
 - Gating mechanism controls gradient flows
- CNN does not have recurrent loops
 - It is suitable for multi-time step parallelization

$$\mathbf{h}(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \odot \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$



CTC-based models without 1-D convolution

- The greedy decoding results are shown below
- Gated ConvNet does not show good performance considering the parameter size.

Table 1: WER and CER (%) evaluated on WSJ eval92. The models are trained on SI-284.

Models	Params.	CER	WER
4x600 LSTM	11.5M	7.29	24.88
6x800 QRNN ($k = 1$)	11.5M	12.70	45.22
6x700 Diagonal LSTM	11.5M	8.97	30.40
6x300 Gated ConvNet ($T = 15$)	16.2M	8.02	28.65
6x300 Gated ConvNet ($T = 7$)	7.7M	8.52	30.56



Element-wise 1-D convolution

- Instead of using shorter 2-D convolution, we add 1-D time-depth convolution to increase the context length
- The size of parameters is very light

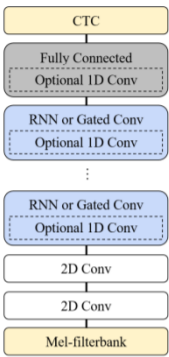


Figure 2: Acoustic modeling architecture for end to end speech recognition.



CTC models with 1-D Convolution

- With 1-D convolution, on-device friendly models catch up and surpasses the LSTM RNN
 - Deeper architecture in Gated ConvNet contributes to performance improvements
- The greedy decoding results are shown below:

Table 2: WER and CER (%) evaluated on WSJ eval92 with 1-D convolution.

Models	Params.	CER(%)	WER(%)
4x600 LSTM, 1-D conv	11.5M	6.95	23.57
6x700 QRNN ($k = 1$), 1-D conv	11.5M	5.26	19.07
6x700 Diagonal LSTM, 1-D conv	11.5M	7.57	23.90
6x300 Gated ConvNet ($T = 15$), 1-D conv	16.2M	7.58	27.00
6x300 Gated ConvNet ($T = 7$), 1-D conv	7.7M	6.57	24.20
20x300 Gated ConvNet ($T = 2$), 1-D conv	7.5M	5.55	19.70
30x300 Gated ConvNet ($T = 2$), 1-D conv	11M	4.73	17.00



Performance comparison with WSJ SI-All datasets

- QRNN and Gated ConvNet show higher performance than LSTM RNN
- In Greedy decoding Gated ConvNet with 1-D conv is best, but as the beam-width increases QRNN based one performs better.

Table 3: WER and CER (%) on WSJ eval92. The models are trained on WSJ SI-ALL.

Models	Params.	Greedy		HCLM	
		CER	WER	CER	WER
4x600 LSTM, 1-D conv	11.5M	5.91	20.14	2.71	6.56
6x700 QRNN ($k = 1$), 1-D conv	11.5M	4.13	18.02	1.51	3.73
30x300 Gated ConvNet ($T = 2$), 1-D conv	11M	3.30	11.60	1.53	3.86
Deep Speech 2 (Amodei et al., 2016)		WER 3.60 with 5-gram LM			



Implementation Results on Embedded Systems

- Multi-time step parallelization is performed in on-device friendly models
 - The QRNN and Gated ConvNet are 7.30 and 5.03 times faster than LSTM when the output for eight timesteps are computed at a time

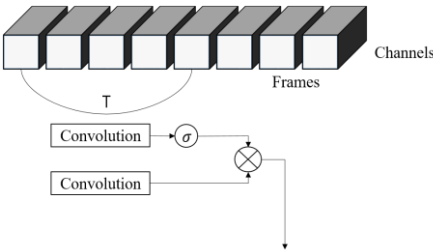
Table 6: Execution time (sec) for models for 1 sec speech, function of T_P

Models	$T_P = 1$	$T_P = 4$	$T_P = 8$	$T_P = 16$
4x600 LSTM	1.46			
6x700 QRNN ($k = 1$), 1-D conv	1.21	0.39	0.20	0.15
6x700 Diagonal LSTM	1.55	0.47	0.25	0.18
30x300 Gated ConvNet, 1-D conv	1.85	0.47	0.29	0.20



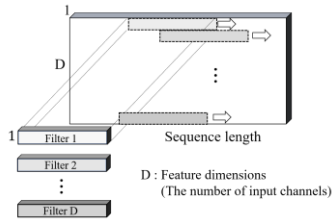
Simple Gated ConvNet for small foot-print end-to-end acoustic modeling

- The complexity of Gated ConvNet increases in proportional to T. What if T=1?
 - The model lacks sequence modeling capability
 - But, what if we add 1-d convolution



Simple Gated ConvNet: The minimum Gated ConvNet (T=1) + 1-D depthwise convolution

- Increasing the time-depth with 1-D convolution
 - The number of parameters needed : TD
 - TD is much smaller than TD^2 of Gated ConvNet
 - Now The number of parameters needed : $2D^2 + TD$
- 1 Matrix operation is needed for type matching (rotating axis)



Balduzzi, David, and Muhammad Ghifary.
"Strongly-typed recurrent neural networks."
arXiv preprint arXiv:1602.02218(2016).

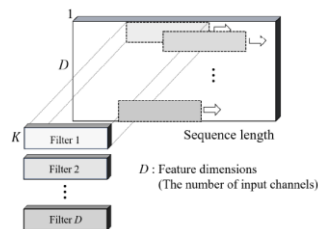


1-D depthwise convolution with the width of K

- Neighboring features may have correlations.
 - Let's consult neighboring channels.

- Consulting multiple K channels

$$h_{t,1,d}^l = \sum_{w=-K/2}^{K/2} \sum_{i=-T/2}^{T/2} \mathbf{F}_{i,1,d,w}^{l,l} \mathbf{X}_{t+i-1,1,d+w}^l$$



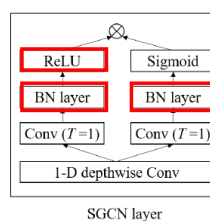
- The model sizes increases from TD to TDK
- Now the number of parameters needed : $2D^2 + TDK$



Batch normalization

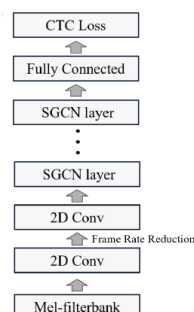
- We employ batch normalization in Simple Gated ConvNet
 - Batch Normalization was not used in previous Gated ConvNet
 - But how? Motivated by ResNet (Kaiming He, 2016)
 - Right after Convolution operations before activations
 - Just add ReLU to the output of

$$X^l * W^l + b^l$$



Whole structure of SGCN for end-to-end acoustic modeling

- Two – layered Convolution Neural networks is adopted
 - Decimate the sequence length by two.
 - Reducing the arithmetic complexity.
- Parameter size Comparison with Gated ConvNet
 - $2TD^2$ of Gated ConvNet vs $2D^2 + TDK$ of Simple Gated
 - Size of D is at least a few hundred.
 - K ranges from 1 to 21.



Experimental results: overall performance comparison on WSJ Si-284

- SCGN outperforms LSTM RNN
 - SGCN with 2.24 M > LSTM with 3M
 - SGCN with 2.24 M > LSTM with 10 M
 - SGCN > Bidirectional LSTM
- SGCN outperforms GCN – Increasing the depth helps

Table 1. WER and CER in percentage trained in WSJ Si-284 and evaluated on WSJ eval92 test set. SGCN is Simple Gated ConvNet. GCN is short of Gated ConvNet.

Model	Params.	Greedy	
		CER	WER
4x300 LSTM	2.95M	8.18	27.6
4x300 LSTM, 1-D depthwise	2.95M	7.13	24.7
4x575 LSTM, 1-D depthwise	10.01M	6.31	21.9
4x180 LSTM, Bidirectional	3.08M	5.61	19.3
6x300 GCN ($T = 15$)	16.38M	8.01	28.6
6x300 GCN ($T = 4$), 1-D depthwise	4.43M	6.37	23.1
6x300 GCN ($T = 2$), 1-D depthwise	2.25M	6.69	24.1
12x220 GCN ($T = 2$), 1-D depthwise	2.47M	5.67	20.1
6x300 SGCN	1.16M	7.03	25.5
12x300 SGCN	2.24M	5.32	18.8

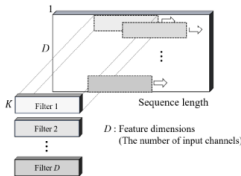


Effect of consulting neighboring channels (K)

Increasing K contributes to performance increase, but shows diminishing return (maybe $K=3$ would be optimum)

Table 2. WER and CER in percentage with various width of 1-D depthwise convolution, trained in WSJ Si-284 and evaluated on WSJ eval92 test set.

Model	Params.	Greedy	
		CER	WER
12x300 SGCN ($K = 1$)	2.24M	5.32	18.8
12x300 SGCN ($K = 3$)	2.33M	5.09	17.8
12x300 SGCN ($K = 5$)	2.42M	5.14	18.3
12x300 SGCN ($K = 7$)	2.50M	5.10	17.8
12x300 SGCN ($K = 11$)	2.68M	4.93	17.5
12x300 SGCN ($K = 15$)	2.85M	4.96	17.7
12x300 SGCN ($K = 21$)	3.10M	4.75	16.8



SGCN under 1 M

- SGCN shows more Robust performance than LSTM RNN

Table 4. WER and CER in models with 1 M parameters, evalutated on WSJ eval92.

(a) Trained in WSJ Si-284.

Model	Params.	Greedy	
		CER	WER
4x160 LSTM	0.96M	10.78	37.1
4x160 LSTM, 1-D depthwise	0.97M	9.64	33.6
12x190 SGCN ($K = 5$)	0.99M	6.11	22.0

(b) Trained in larger dataset(WSJ Si-ALL).

Model	Params.	Greedy	
		CER	WER
4x160 LSTM	0.96M	8.64	31.2
4x160 LSTM, 1-D depthwise	0.97M	6.72	24.1
12x190 SGCN ($K = 5$)	0.99M	5.11	18.8



SGCN for low-latency tasks

- When the filter length T of 1-D depthwise convolution is 11, and the filter consults the input from $t-5$, $t+5$, where $t+5$ is 5-step futures
- One SGCN layer operates with the 20 ms time-steps, and one non-causal layer incurs 100 ms delay (Considering 5-step futures), the latency hinders real-time speech recognition
- In order to eliminate latency, experiments with causal convolution filters were conducted



Performance of SGCN under low latency conditions

- 12-layered Non-causal convolution leads to 1200 ms delay.
- Non-causal convolution of only one layer leads to 100 ms
- Non-causal convolution of only two layers leads 200 ms
- Even with 200 ms latency, it shows affordable performance.

Table 5. WER and CER in percentage with models for low latency tasks, trained in WSJ Si-284 and evalutated on WSJ eval92.

Model	Params.	Greedy	
		CER	WER
12x300 SGCN (100 ms latency)	2.68M	6.20	22.4
12x300 SGCN (200 ms latency)	2.68M	5.01	18.1
12x300 SGCN (1200 ms latency)	2.68M	4.93	17.5



Execution time of multi-time-step SGCN

- The implementation results of SGCN on the ARM Cortex-A57.
- T_p : the number of output samples computed at a time (x10 speed-up compared to the LSTM model)

Table 6. Execution time in second for processing 1 sec speech, function of T_p .

Model	T_p			
	1	4	8	16
4x275 LSTM	0.459	-	-	-
12x300 SGCN	0.363	0.095	0.061	0.043



Other current and future works

- Fine-tuning of the models under latency and other decoding condition, especially AutoML will help.
 - We find that reducing the number of 1-D convolution yields higher accuracy
- Experiments on large and real datasets for on-device SR
- Application of SRU+1D model to LAS
 - LAS would be benefit much by using linear RNNs and GCNs
 - We have not yet obtained very good results
- Application of SGCN to keyword spotting
- Combination with existing speed-up methods, such as compression and fixed-point optimization.
 - RNN model compression to very low bits (2bit)



Summary

- We explored linear RNN (QRNN, SRU) and Gated ConvNet based models for efficient on-device acoustic modeling. These models support multi-time-step parallel execution and show 3~8 times of speedup on embedded devices
- We can increase the performance of QRNN and Gated ConvNet based models by augmenting 1-d convolution at each layer
- Especially, SGCN showed good performance in low footprint setting (1Million or around).



Publications related to today's talk

- Fully Neural Network Based Speech Recognition on Mobile and Embedded Devices, J Park, Y Boo, I Choi, S Shin, W Sung, Advances in Neural Information Processing Systems, 2018
- SIMPLE GATED CONVNET FOR SMALL FOOTPRINT ACOUSTIC MODELING' submitted to the ICASSP 2019

