Open access • Proceedings Article • DOI:10.1109/ISCAS.2019.8702290

Real-Time Speech Recognition for IoT Purpose using a Delta Recurrent Neural Network Accelerator — Source link

Chang Gao, Stefan Braun, Ilya Kiselev, Jithendar Anumula ...+2 more authors

Institutions: University of Zurich

Published on: 26 May 2019 - International Symposium on Circuits and Systems

Topics: Recurrent neural network, Latency (engineering), Throughput (business) and Field-programmable gate array

Real-Time Speech Recognition for IoT Purpose using a Delta Recurrent Neural Network Accelerator

Gao, Chang ; Braun, Stefan ; Kiselev, Ilya ; Anumula, Jithendar ; Delbruck, Tobi ; Liu, Shih-Chii

Abstract: This paper describes a continuous speech recognition hardware system that uses a delta recurrent neural network accelerator (DeltaRNN) implemented on a Xilinx Zynq-7100 FPGA to enable low latency recurrent neural network (RNN) computation. The implemented network consists of a single-layer RNN with 256 gated recurrent unit (GRU) neurons and is driven by input features generated either from the output of a filter bank running on the ARM core of the FPGA in a PmodMic3 microphone setup or from the asynchronous outputs of a spiking silicon cochlea circuit. The microphone setup achieves 7.1 ms minimum latency and 177 frames-per-second (FPS) maximum throughput while the cochlea setup achieves 2.9 ms minimum latency and 345 FPS maximum throughput. The low latency and 70 mW power consumption of the DeltaRNN makes it suitable as an IoT computing platform.

DOI: https://doi.org/10.1109/iscas.2019.8702290

Accepted for publication at the 2019 IEEE International Symposium on Circuits and Systems (ISCAS)

Real-time Speech Recognition for IoT Purpose using a Delta Recurrent Neural Network Accelerator

Chang Gao, Stefan Braun, Ilya Kiselev, Jithendar Anumula, Tobi Delbruck and Shih-Chii Liu
Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland
Email: [chang, sbraun, kiselev, anumula, tobi, shih]@ini.uzh.ch

***Abstract*—This paper describes a continuous speech recognition hardware system that uses a delta recurrent neural network accelerator (DeltaRNN) implemented on a Xilinx Zynq-7100 FPGA to enable low latency recurrent neural network (RNN) computation. The implemented network consists of a single-layer RNN with 256 gated recurrent unit (GRU) neurons and is driven by input features generated either from the output of a filter bank running on the ARM core of the FPGA in a PmodMic3 microphone setup or from the asynchronous outputs of a spiking silicon cochlea circuit. The microphone setup achieves 7.1 ms minimum latency and 177 frames-per-second (FPS) maximum throughput while the cochlea setup achieves 2.9 ms minimum latency and 345 FPS maximum throughput. The low latency and 70 mW power consumption of the DeltaRNN makes it suitable as an IoT computing platform.**

***Index Terms*—deep learning, speech recognition, Internet-of-Things, edge computing, cochlea, RNN, FPGA**

I. Introduction

Deep neural networks are currently the state-of-the-art algorithms for solving many machine learning tasks. Recurrent Neural Networks (RNNs) with gated units, such as the Long Short-Term Memory (LSTM) [1] and Gated Recurrent Unit (GRU) [2], are useful for solving tasks with long sequences such as speech recognition. Deploying these networks on mobile platforms require hardware implementations that are energy efficient. Hardware accelerators for deep neural networks (DNNs) show better energy efficiency numbers than their implementations on CPUs and GPUs [3]. Network architectures on mobile platforms and embedded systems, also face the constraint of reduced memory, memory bandwidth, and computing resources. A reduced memory footprint is critical for the intended network running on these platforms.

Proposed methods to reduce memory footprint of networks include quantization, pruning, and the use of special weight matrices. Quantization of weights or activations help to reduce external memory bandwidth requirement and area of arithmetic units [3]–[6]. Special hardware modules such as multipliers that are based on Look-Up Tables (LUTs) [7] or multiplexers [8] can be used for networks with low bit precision parameters thereby reducing the hardware area. Various training methods have been proposed for reducing the bit precision of the parameters while maintaining close to the accuracy of the full precision network [9]–[12]. Training methods to prune connections with small weight values also help to create networks with small number of parameters, e.g. [3], [13]. The use of Toeplitz-like weight matrices, such as circulant matrices, also helps to reduce the on-board memory requirements [14], [15].

Other methods of reducing computation cost and therefore memory accesses include the delta network where neurons of an RNN layer are updated only when their activations change by a threshold across two timesteps [16], [17]. This is particularly useful with natural signals such as speech where the input does not change quickly over time. Reducing neuron updates also leads to reduced memory fetches of the weight parameters and increased energy efficiency of the hardware.

This paper describes a real-time continuous speech recognition system that uses the DeltaRNN accelerator to reduce updates of recurrent network activations [18]. The accelerator can be interfaced to either the output of a PmodMic3 microphone or the output of a spiking silicon cochlea system. The features are obtained by either applying a log filter bank on the microphone outputs or by creating spike count features from a silicon spiking cochlea [19]. Unlike previous studies that generated offline cochlea features [20] this work uses online generated features [21]. The implementation of this automatic speech recognition (ASR) system using a spiking sensor is an extension to the deep network studies described in [22], [23].

Section II describes the input audio feature extraction methods and the training algorithms. Section III describes the system architecture and Section IV presents the results of the recognition task and system measurements.

II. Continuous Speech Recognition

The speech recognition system maps input speech to output digit labels. It is based on a recurrent network model trained in an end-to-end fashion using the Connectionist Temporal Classification (CTC) cost function [24]. The input speech signal is binned into $t = 1, ..., T$ frames and then converted into D-dimensional audio features $x_t \in \mathcal{R}^D$. For each frame t, a neural network with parameters θ, computes the probability distribution $P_t(y|x_{1:t}, \theta)$ of the output labels $y \in \mathcal{R}^{V+1}$. The output labels consist of 12 classes: $V = 11$ digit labels ('zero', 'oh', '1', ..., '9') and the CTC blank label. For decoding, we select the most probable label $\hat{y}_t$ for every frame t as the output label.

This work was partially supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 644732, the Swiss National Science Foundation, HEAR-EAR, 200021_172553, and the Samsung Institute of Advanced Technology.

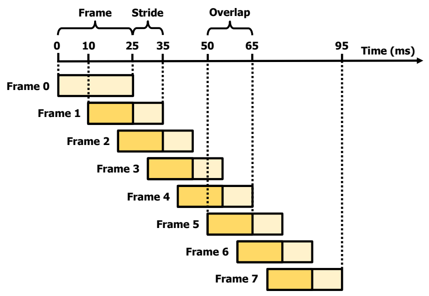


Fig. 1: Frame size, stride and overlap.

A. *Processing Microphone Audio*

1) Real-Time Normalization & Feature Extraction: Audio frames are created using a frame size of 25 ms and a stride of 10 ms with a 15 ms overlap between adjacent frames.

As shown in Fig. 1, every 8 frames of an audio sequence are normalized by the mean and standard deviation computed over these frames. Log filter bank features are then obtained from this normalized sequence by applying the Hamming window function, computing the Fourier transform, calculating the power spectrum and finally using 40 triangular filters.

2) Training: The end-to-end model consists of a recurrent layer with 256 unidirectional delta-GRUs [16] with 0.25 delta threshold. The recurrent layer is followed by a 200-dimensional fully connected (FC) layer with a ReLU non-linearity and a 12-dimensional FC output layer. During training, dropout with probability 0.5 is applied on the 200-dimensional FC layer. The `Adam` optimizer is used with a learning rate of 1e-4. The L1 cost factor [16] is set to 0.01. Before each forward pass, weights, biases, inputs and activations of the delta-GRU layer are quantized into 16-bit Q8.8 fixed-point numbers with `Pow-2` rounding [12] following Eq. 1 which describes how a parameter P in floating-point format is quantized to P_q in Qm.f format.

$$P_q = \frac{\mathrm{rnd}[\max(\min(P \times 2^{\mathrm{f}}, 2^{\mathrm{m+f-1}} - 1), -2^{\mathrm{m+f-1}} + 1)]}{2^{\mathrm{f}}} \tag{1}$$

where the rnd function rounds the argument to the nearest real integer.

The model is trained on the original `TIDIGITS` dataset [25] and an augmented dataset. The original dataset has 7h training data and 7h test data. For augmentation, both speed perturbations and noise injection are applied. Each utterance is scaled at 90%, 100% and 110% of the original speed; and then mixed with a randomly selected noise segment from the CHiME-4 [26] background recordings (bus, cafe, pedestrian area, street junction) at a randomly selected signal-to-noise ratio (SNR) of $[0:5:50]$dB. The augmented dataset has 80h noisy audio data and is split into a 72h training set and a 8h test set. Data augmentation is useful to obtain high accuracy during real-world operation of the system which is evaluated by testing the model on a new test dataset **NT** that is created by the authors. The **NT** dataset has 220 utterances from 11 subjects in the authors' department and is recorded by a Trust Mico mono USB microphone. Moreover, the model is trained on both normalized and unnormalized frames so that we can compare the effectiveness of the real-time normalization method during test.

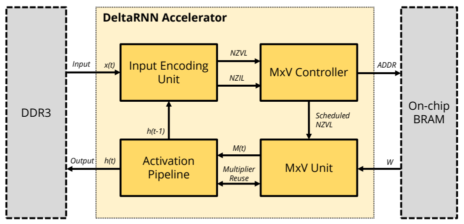


Fig. 2: DeltaRNN architecture.

B. *Spiking Cochlea*

1) Silicon Cochlea: The Dynamic Audio Sensor (DAS) spiking cochlea [19] is a binaural silicon cochlea with 64 channels tuned for frequencies on a log spacing from 100 Hz to about 20 kHz. The sensor models the functionality of the basilar membrane, inner hair cells and spiral ganglion cells of the biological cochlea. The asynchronous output spikes can be streamed to a computer using the USB port on the DAS PCB or to an external processing device via 40-pin parallel AER CAVIAR connector.

2) Dataset: The recordings were done using a subset of the augmented `TIDIGITS` dataset without speed scaling and utterances are mixed with a randomly selected SNR from [0, 5, 10] dB. We used only a subset because of the extensive time needed to record from the cochlea. The files are played through loud speakers placed about 3 m from the DAS PCB and in a normal office room with reverberation T60 = 0.2. The recorded dataset is called `TIDIGITS_das`.

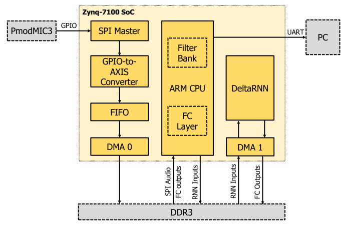
3) Feature Extraction: We generate spike count features by binning spikes within a time window of 5 ms, with no overlap between consecutive windows [21]. Only spikes from one ear and from one out of the 4 neurons of each frequency channel of the DAS cochlea are used for the feature generation.C The time bin length of the spike counter is programmable for a range of 1 – 16383 µs with 1 µs resolution. 8-bit counters are used to count the spikes of each channel and formatted in 16-bit Q8.8 number to be transferred to DeltaRNN by an AXIS interface.

4) Training: The same network architecture for the microphone audio is used on the spike features except that the input vector dimension becomes 64, the delta threshold of the GRU layer is set to 0 and there is no quantization in the forward pass. Recurrent layer weights are quantized after training on the GPU and before inference on the FPGA.

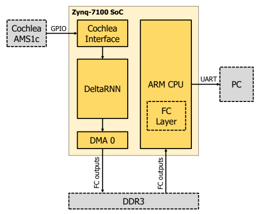
III. System Architecture

A. *Delta Recurrent Neural Network Accelerator*

The DeltaRNN accelerator consists of 4 main modules, Input Encoding Unit (IEU, Matrix-Vector Multiplication Unit (MxV), MxV Controller and Activation Pipeline (AP) as shown in Fig. 2. IEU computes the absolute difference between



(a)



(b)

Fig. 3: System architecture of the microphone setup (a) and of the DAS cochlea setup (b).

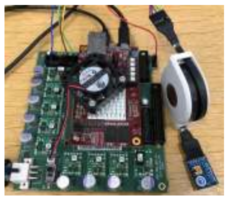
input vectors of the current and the previous timesteps and stores results in a delta vector (DV). Below-threshold elements of the DV are then zero-out. MxV with its controller executes MxV on the DVs and weight matrices (W) fetched from the on-chip BRAM. MxV accumulation results are finally sent to AP to compute non-linear functions including LUT-based sigmoid and tanh, element-wise multiplication and addition. RNN activations are transferred to both IEU and DRAM. The IEU uses current activations to calculate DVs of the next timestep. The ARM core fetches activations from DRAM to calculate FC layers and the softmax function [18].

B. System Integration

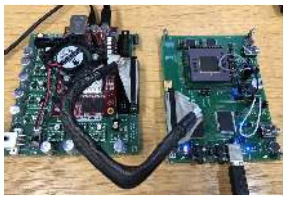
1) Microphone Setup: In the microphone setup architecture shown in Fig. 3a, an SPI master core is used to receive 20 kHz audio samples from the PmodMic3. Audio samples are transferred to the ARM core through DMA0 to compute filter bank features, which are then fed to the DeltaRNN accelerator through DMA1. The ARM core computes FC layers and softmax function using DeltaRNN outputs to generate final classification results that are then sent to a PC through the serial port. There are two clock domains in this setup: **CLK0** for the DeltaRNN is set at either 1 MHz or 100 MHz depending on the required latency or throughput while the other modules on the FPGA are driven by **CLK1** that is fixed at 100 MHz to avoid high latency of data transfer among modules. As shown in Fig. 4a, the setup is implemented on a Zynq Mini-Module Plus (MMP) board and the PmodMic3 is connected to GPIO pins on the baseboard.

It takes 4 steps (normalization, feature extraction, GRU, FC) for the microphone setup to generate classification results. The setup supports the following two modes. **Low Latency Mode (LL):** After normalization of a frame batch (8 frames), remaining steps are executed in sequence for every single frame. A new classification result is generated per frame to achieve relatively low latency; however, the RNN computations on the frames are not pipelined. **High Throughput Mode (HT):** After normalization, the 8 frame vectors in a batch are concatenated into a single vector to be processed by the remaining steps. In this way, computation of a batch is pipelined in DeltaRNN so that relatively higher throughput can be achieved but with increased latency.

2) Cochlea Setup: Figure 3b shows the architecture of cochlea setup, in which the cochlea interface is implemented to



(a)



(b)

Fig. 4: Demo setup with PmodMIC3 (a) and with DAS cochlea (board on the right) (b).

TABLE I: PER and LER of the microphone setup models (**NORM** and **AUG** denote the normalization and augmentation methods respectively). The results of each model are from the epoch that gives the best LER_NT.

	PER_T	LER_T	PER_AT	LER_AT	PER_NT	LER_NT
None	13.17%	4.42%	-	-	14.29%	4.49%
NORM	6.99%	2.35%	-	-	17.86%	5.12%
AUG	-	-	13.31%	5.18%	17.24%	4.00%
Both	-	-	6.19%	2.31%	3.45%	0.67%

count the asynchronous events from the DAS. Spike counter features are transferred directly to DeltaRNN and the GRU layer activations are sent to the ARM CPU through DMA0 to carry out the same classification steps as in the microphone setup. There is only one clock domain at 100 MHz for all modules in the cochlea setup. Figure 4b shows the setup is implemented on the same board which is interfaced to the cochlea board through a AER CAVIAR connector.

IV. Experimental Results

A. Accuracy

Accuracy of 4 models for the microphone setup are evaluated on the original `TIDIGITS` testset (**T**), augmented testset (**AT**) and the new testset (**NT**). Each of the 4 models is trained for 100 epochs with or without real-time normalization or augmentation. Results in Table I show that the real-time normalization method helped to reduce the Phrase Error Rate (PER) and the Label Error Rate (LER) on the testsets of both the original dataset and the augmented dataset. When training the model with the combination of the normalization method and augmentation, the best error rates are achieved on both **T** and **NT** testsets. This model also shows the best performance over the other models in real-world demonstrations. Error rates of the model trained on the cochlea dataset `TIDIGITS_das` are shown in Table II. The higher error rates could be attributed to two reasons. One, the difficulty in using spikes from cochlea [22]. Two, the error rates in the case of cochlea are evaluated on a noisy test set rather than the clean test set (T) as in the case of the microphone setup. A previous work [22] reports 18% LER on recognizing single digits trained on N-TIDIGITS, a spike dataset recorded by a silicon cochlea, suggesting that it is more difficult to get similar error rates for continuous speech recognition using these features.

TABLE II: PER and LER of the model trained on the `TIDIGITS_das` dataset. The results are from the epoch that gives the best LER.

	PER	LER
TIDIGITS_das	47.13%	21.28%

TABLE III: End-to-end latency breakdown and throughput in frame-per-second (FPS) of the microphone setup.

	Platform	Mic_LL	Mic_LL	Mic_HT	Mic_HT
CLK0 (MHz)	-	1	100	1	100
Batch Size	-	1	1	8	8
NORM (ms)	ARM	1.56	1.57	1.38	1.38
FE (ms)	ARM	2.60	2.64	20.91	20.91
GRU (ms)	DeltaRNN	0.43	0.01	0.47	0.01
FC (ms)	ARM	2.88	2.88	23.00	23.00
Hardware (ms)	ARM+DeltaRNN	7.47	7.10	45.76	45.30
Hardware FPS	ARM+DeltaRNN	133.9	140.8	174.8	176.6
Framing (ms)	-	95	95	95	95
Total (ms)	ARM+DeltaRNN	102.47	102.10	140.76	140.30

TABLE IV: End-to-end latency breakdown of the cochlea setup with DeltaRNN running at 100 MHz.

	Platform	Cochlea
CLK0 (MHz)	-	100
Batch Size	-	1
GRU (ms)	DeltaRNN	0.02
FC (ms)	ARM	2.88
Hardware (ms)	ARM+DeltaRNN	2.90
Hardware FPS	ARM+DeltaRNN	344.8
Framing (ms)	-	5
Total (ms)	ARM+DeltaRNN	7.90

B. Latency and Throughput

Table III shows the latencies of the 4 steps of the microphone pipeline and the effective throughput of the hardware. The total system latency is given by the sum of the hardware latency and the framing latency. To run this system in real-time, the hardware throughput should be higher than 100 FPS given the 10 ms frame stride. Since the system steps are not pipelined from end to end, the hardware FPS is the reciprocal of the hardware latency. The maximum hardware throughput of 176.6 FPS is achieved in **HT** mode with CLK0 = 100 MHz and the processing by the DeltaRNN of the 8 frames in 0.01 ms [1]. In real-world demonstrations, the **LL** mode at 1 MHz is used to reduce power consumption. Table IV shows the latency and throughput numbers of the cochlea setup. Lower latency is achieved by the cochlea setup without feature extraction on ARM. Spike features generated by the counters of the cochlea interface are fed to the DeltaRNN through an AXIS port and the latency of the interface is negligible. The cochlea setup achieves hardware latency of 2.90 ms and hardware throughput of 344.8 FPS.

[1]DeltaRNN throughput is throttled by the throughput of the DMA at 100 MHz

TABLE V: System power breakdown (*measured by a wall plug power meter)

Setup	Mic_LL	Mic_LL	Cochlea
CLK0 (MHz)	1 MHz	100 MHz	100 MHz
Baseboard + Fan + Mic*	8.1 W	8.1 W	8.1 W
ARM	1.58 W	1.59 W	1.58 W
FPGA	0.07 W	2.86 W	2.78 W
DAS ASIC	-	-	22 mW
Static Power	0.25 W	0.29 W	0.29 W
Wall Plug Power*	11.6 W	13.7 W	13.7 W

C. System Power Measurements

Power measurements are shown in Table V. The total power consumption of the Zynq-7100 SoC and the baseboard is measured by using a wall plug power meter. The power of the cochlea ASIC is extracted from [19]. Other power numbers are estimated using the Xilinx Power Estimator with 0.5 switching rate. The minimum wall plug power of 11.6 W is achieved by the **Mic_LL** setup running at 1 MHz. At this clock frequency, the FPGA consumes only 70 mW, which includes 20 mW from the DeltaRNN, 20 mW from BRAM blocks and 30 mW from other modules. The hardware power efficiency of the **Mic_LL** setup is 11.5 FPS/W while the **cochlea** setup achieves 25.2 FPS/W. The main bottleneck of the power efficiency is the relatively low throughput of ARM core and the excessive power of the baseboard. Considering only the FPGA in **Mic_LL** setup at 1 MHz, the power efficiency is 33 kFPS/W at 0.07 W; while an Nvidia GTX 1080 achieves 2 kFPS/W at 48 W measured by running the same demo in PyTorch 1.0.1 with CUDA 10 and CuDNN 7.4.2. An Intel Core i7-5820K CPU is used for feature extraction in the measurement.

V. Conclusion

This paper presents a real-time continuous ASR delta RNN-based hardware system that can be interfaced to either PMOD microphones or the DAS spiking cochlea. The DeltaRNN accelerator on FPGA consumes only 70 mW and can process each feature frame in microseconds therefore, this system is suitable for IoT applications. For better ASR accuracy of the system in real-world conditions, the training dataset was augmented with noise samples of different SNRs. The paper also describes the first online ASR hardware DNN system that is interfaced to the DAS. The system allows one to prototype ASR networks which can then be made into a low-power ASIC with either a front-end filter bank for voice activity detection (VAD) or word recognition as in [27], [28] or an ultra low-power spiking cochlea [29] for VAD [8]. The DeltaRNN can also be added to the real-time portable wireless acoustic multi-microphone platform [30] useful for source separation and ASR.

VI. Acknowledgement

We gratefully acknowledge the Robotics and Technology of Computers Lab, University of Seville for providing the baseboard for Zynq MMP.

References

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[2] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN Encoder–Decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Oct. 2014, pp. 1724–1734. [Online]. Available: http://www.aclweb.org/anthology/D14-1179

[3] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang *et al.*, “ESE: Efficient speech recognition engine with sparse LSTM on FPGA,” in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 75–84.

[4] A. X. M. Chang and E. Culurciello, “Hardware accelerators for recurrent neural networks on FPGA,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, ser. ISCAS ’17, 2017.

[5] M. Lee, K. Hwang, J. Park, S. Choi, S. Shin, and W. Sung, “FPGA-based low-power speech recognition with recurrent neural networks,” *CoRR*, vol. abs/1610.00552, 2016. [Online]. Available: http://arxiv.org/abs/1610.00552

[6] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, “FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates,” in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, April 2017, pp. 152–159.

[7] D. Shin, J. Lee, J. Lee, and H. Yoo, “14.2 dnpu: An 8.1tops/w reconfigurable cnn-rnn processor for general-purpose deep neural networks,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 240–241.

[8] M. Yang, C. Yeh, Y. Zhou, J. P. Cerqueira, A. A. Lazar, and M. Seok, “A 1μW voice activity detector using analog feature extraction and digital deep neural network,” in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 346–348.

[9] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, “WRPN: Wide reduced-precision networks,” in *International Conference on Learning Representations*, 2018.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4107–4115.

[11] J. Ott, Z. Lin, Y. Zhang, S. Liu, and Y. Bengio, “Recurrent neural networks with limited numerical precision,” *CoRR*, vol. abs/1608.06902, 2016.

[12] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, “Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms,” *Frontiers in neuroscience*, vol. 9, p. 222, 2015.

[13] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding,” *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[14] Z. Wang, J. Lin, and Z. Wang, “Accelerating recurrent neural networks: A memory-efficient approach,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2763–2775, Oct 2017.

[15] S. Wang, Z. Li, C. Ding, B. Yuan, Q. Qiu, Y. Wang, and Y. Liang, “C-lstm: Enabling efficient lstm using structured compression techniques on fpgas,” in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA ’18. New York, NY, USA: ACM, 2018, pp. 11–20. [Online]. Available: http://doi.acm.org/10.1145/3174243.3174253

[16] D. Neil, J. Lee, T. Delbrück, and S. Liu, “Delta networks for optimized recurrent network computation,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 2584–2593. [Online]. Available: http://proceedings.mlr.press/v70/neil17a.html

[17] P. O’Connor and M. Welling, “Sigma delta quantized networks,” *arXiv preprint arXiv:1611.02024*, 2016.

[18] C. Gao, D. Neil, E. Ceolini, S.-C. Liu, and T. Delbruck, “DeltaRNN: A power-efficient recurrent neural network accelerator,” in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA ’18. New York, NY, USA: ACM, 2018, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/3174243.3174261

[19] S. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, “Asynchronous binaural spatial audition sensor with $2 \times 64 \times 4$ channel output,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 453–464, Aug 2014.

[20] M. Abdollahi and S.-C. Liu, “Speaker-independent isolated digit recognition using an AER silicon cochlea,” in *Proceedings of the Biomedical Circuits and Systems Conference (BIOCAS)*, 2011, pp. 269–272.

[21] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, “Feature representations for neuromorphic audio spike streams,” *Frontiers of Neuroscience*, 2018.

[22] D. Neil and S. Liu, “Effective sensor fusion with event-based sensors and deep network architectures,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 2282–2285.

[23] I. Kiselev, D. Neil, and S.-C. Liu, “Event-driven deep neural network hardware system for sensor fusion,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 2495–2498.

[24] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[25] R. G. Leonard and G. Doddington, “Tidigits speech corpus,” *Texas Instruments, Inc*, 1993.

[26] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, “An analysis of environment, microphone and data simulation mismatches in robust speech recognition,” *Computer Speech & Language*, 2016.

[27] K. Badami, S. Lauwereins, W. Meert, and M. Verhelst, “Context-aware hierarchical information-sensing in a 6μW 90nm CMOS voice activity detector,” in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb 2015, pp. 1–3.

[28] M. Price, J. Glass, and A. P. Chandrakasan, “A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 244–245.

[29] M. Yang, C. Chien, T. Delbruck, and S. Liu, “A 0.5v 55 μW 64 $\times$ 2 channel binaural silicon cochlea for event-driven stereo-audio sensing,” *IEEE Journal of Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov 2016.

[30] I. Kiselev, E. Ceolini, D. Wong, A. d. Cheveigne, and S.-C. Liu, “WHISPER: Wirelessly synchronized distributed audio sensor platform,” in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, Oct 2017, pp. 35–43.