INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Endpoint Detection using Grid Long Short-Term Memory Networks for Streaming Speech Recognition

Shuo-Yiin Chang, Bo Li, Tara N. Sainath, Gabor Simko, Carolina Parada
Google Inc., U.S.A
{shuoyiin, boboli, tsainath, gsimko, carolinap}@google.com

Abstract

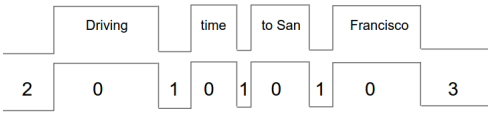
The task of endpointing is to determine when the user has finished speaking. This is important for interactive speech applications such as voice search and Google Home. In this paper, we propose a GLDNN-based (grid long short-term memory deep neural network) endpointer model and show that it provides significant improvements over a state-of-the-art CLDNN (convolutional, long short-term memory, deep neural network) model. Specifically, we replace the convolution layer in the CLDNN with a grid LSTM layer that models both spectral and temporal variations through recurrent connections. Results show that the GLDNN achieves 32% relative improvement in false alarm rate at a fixed false reject rate of 2%, and reduces median latency by 11%. We also include detailed experiments investigating why grid LSTMs offer better performance than convolution layers. Analysis reveals that the recurrent connection along the frequency axis is an important factor that greatly contributes to the performance of grid LSTMs, especially in the presence of background noise. Finally, we also show that multichannel input further increases robustness to background speech. Overall, we achieve 16% (100 ms) endpointer latency improvement relative to our previous best model on a Voice Search Task.

1. Introduction

In many streaming speech recognition applications, the *endpointer* is an essential component that determines when the user of a system has finished speaking. Endpointing is a challenging task because we want to make the endpoint decision as fast as possible so the system feels very responsive (as measured by reduced latency), while not cutting off the user prematurely which would increase WER. Bad endpointing performance has been blamed for low user satisfaction [1, 2]. In a traditional speech recognition system [3, 4, 5, 6] a voice-activity detection (VAD) classifier is used to label each audio frame as either speech or silence (strictly non-speech), and the mic closing decision is made when a fixed amount of silence is detected. One downside of this approach is that it ignores potential acoustic cues such as filler sounds or user speaking rate, which may indicate whether a given pause is temporary or query-final.

Similar to the work in [7], we directly train a model for the endpointing task: we predict when the user is done speaking. Specifically, the *end-of-query* (*EOQ*) classifier can predict one of 4 labels: *speech*, *initial silence*, *intermediate silence*, and *final silence*. Figure 1 shows an example of the targets. The posteriors of final silence are then thresholded to obtain a mic closing decision. We use this system as our baseline since it showed around 100 ms latency improvements compared to a VAD system trained and evaluated on the same data.

The LSTM [8] is a popular architecture for sequential modeling in speech recognition [9, 10, 11] . It has also shown state-of-the-art performance for endpointer/VAD task [12]. In [13], LSTMs were further integrated with DNNs and a convolution layer in a unified framework, known as the CLDNN architecture. CLDNNs have shown improvements over LSTMs for the VAD task [4]. However, convolution layers could degrade performance in highly noisy conditions as observed in [14]. In fact, [14] proposed an alternative to convolution by using a grid LSTM that models time-frequency patterns with LSTMs unrolled across both time and frequency jointly. A grid LSTM models the time/frequency variations through a recurrent state that is passed from one step to another rather than convolving and pooling as CNNs do. In this paper, we explore using a grid LSTM architecture for the endpointer detection task. The architecture that replaces the convolution layer in CLDNN with a grid LSTM is referred as GLDNN.



0: speech, 1: intermediate sil, 2: initial sil, 3: final sil

Figure 1: *End-of-query classifier.*

We compare the GLDNN endpointer against our best system using CLDNN for the EOQ endpointing task [7]. Results show that GLDNN achieves 32% relative improvement in terms of false alarm (FA) when fixing false reject rate (FR) at 2% for the final silence classification and reduced median endpointer latency by 11%.

Furthermore, we perform analysis to investigate the benefit of recurrent modeling that differentiates grid LSTMs from a convolution layer. Analysis reveals that the recurrent connection in the GLDNN along the frequency axis is an important factor that greatly contributes to the performance of grid LSTMs in the presence of background noise. Finally, we further explore the value of using 2-channel audio vs 1-channel audio for this task, and we find that we achieve 16% improvement when using 2-channel models.

The rest of this paper is as follows. In Section 2, we describe the grid LSTM architecture. The experimental setup is described in Section 3 and results and analyses are presented in Section 4. Finally, Section 5 concludes the paper.

2. Neural Network Architecture

This section describes the GLDNN endpointer architecture. Figure 2 shows an overview of the proposed GLDNN (*right*) and baseline CLDNN (*left*) endpointer architectures. In the GLDNN architecture, we use a grid LSTM in the first layer in-

http://dx.doi.org/10.21437/Interspeech.2017-284

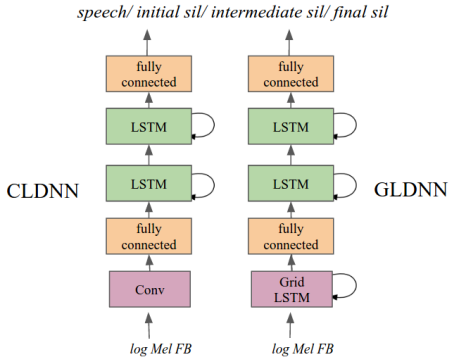


Figure 2: *CLDNN (left) and GLDNN (right).*

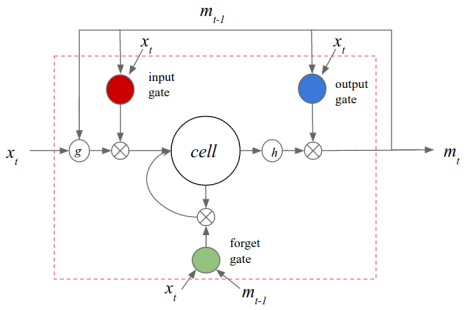


Figure 3: *Memory block in LSTM.*

stead of a convolution layer. The output of grid LSTM is then passed to a dimensionality reduction layer, two basic LSTM layers and finally to a single fully connected layer. The output layer is a softmax distinguishing four classes: speech, initial silence, intermediate silence and final silence.

2.1. LSTM

We used the LSTM architecture described in [8] (see Figure 3) in the third and fourth layer of the proposed architecture. The LSTM consists of a set of recurrently connected subnetworks referred to as *memory blocks*. Each memory block contains *memory cells* to store the temporal state of the network, as well as three multiplicative gate units to control information flow. The *input gate* controls the information passed from the input activations into the memory cells, and the *output gate* controls the information passed from the memory cells to the rest of the network. Finally, the *forget gate* adaptively resets the memory of the cell. The LSTM model is described by the following equations at step t:

$$q_u = W_{um} m_{t-1}, \quad u \in \{i, f, c, o\} \tag{1}$$
$$i_t = \sigma(W_{ix} x_t + q_i + b_i) \tag{2}$$
$$f_t = \sigma(W_{fx} x_t + q_f + b_f) \tag{3}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx} x_t + q_c + b_c) \tag{4}$$
$$o_t = \sigma(W_{ox} x_t + q_o + b_o) \tag{5}$$
$$m_t = o_t \odot h(c_t) \tag{6}$$

where i_t, f_t, c_t and o_t denote the input, forget, memory cell and output gate activations at step t. m_t is the output of the LSTM layer. W are the different weight matrices, for example W_{ix} is the weight matrix from the input gate to the input. $\odot$ is an element-wise dot product. Finally, σ is the logistic sigmoid non-linearity while g and h are the cell input and output activations, which we take to be *tanh*.

2.2. Grid LSTM

A grid LSTM [15] extends basic LSTM to model the sequential process for multidimensional input (time-frequency pattern from spectrogram in our task). A grid LSTM is shown in Figure 4 (*left*). Unlike the standard LSTM architecture presented in section 2.1, in a grid LSTM, each memory cell is connected to a local subset of frequency bands to store current time-frequency state of the network.

A grid LSTM consists of two seperate LSTMs, grid time LSTM (gT-LSTM) and grid frequency LSTM (gF-LSTM), to model correlation in time and frequency independently. A gT-LSTM with memory cells ($c_{t,k}^{(k)}$) forwards output ($m_{t,k}^{(k)}$) in next time step while a gF-LSTM passes output ($m_{t,k}^{(k)}$) from memory cell ($c_{t,k}^{(k)}$) in each frequency step through recurrent connections. The grid LSTM is given by the following equations at each time-frequency step (t, k):

$$q_{u,t,k} = W_{um}^{(t)} m_{t-1,k}^{(t)} + W_{um}^{(k)} m_{t,k-1}^{(k)}, \quad u \in \{i, f, c, o\} \tag{7}$$
$$i_{t,k}^{(s)} = \sigma(W_{ix}^{(s)} x_{t,k} + q_{i,t,k} + b_i^{(s)}), \quad s \in \{t, k\} \tag{8}$$
$$f_{t,k}^{(s)} = \sigma(W_{fx}^{(s)} x_{t,k} + q_{f,t,k} + b_f^{(s)}), \quad s \in \{t, k\} \tag{9}$$
$$c_{t,k}^{(t)} = f_{t,k}^{(t)} \odot c_{t-1,k}^{(t)} + i_{t,k}^{(t)} \odot g(W_{cx}^{(t)} x_{t,k} + q_{c,t,k} + b_c^{(t)}) \tag{10}$$
$$c_{t,k}^{(k)} = f_{t,k}^{(k)} \odot c_{t,k-1}^{(k)} + i_{t,k}^{(k)} \odot g(W_{cx}^{(k)} x_{t,k} + q_{c,t,k} + b_c^{(k)}) \tag{11}$$
$$o_{t,k}^{(s)} = \sigma(W_{ox}^{(s)} x_{t,k} + q_{o,t,k} + b_o^{(s)}), \quad s \in \{t, k\} \tag{12}$$
$$m_{t,k}^{(s)} = o_{t,k}^{(s)} \odot h(c_{t,k}^{(s)}), \quad s \in \{t, k\} \tag{13}$$

In these equations, replacing s with t gives the gT-LSTM and replacing s with k gives the gF-LSTM. Eq. (7) extends (1) by using information of the previous output from both the gF-LSTM ($m_{t,k-1}^{(k)}$) and gT-LSTM ($m_{t-1,k}^{(t)}$) to explore the benefit of seperate LSTMs modeling correlation on time and frequency.

To extract features using the grid LSTM to feed to the LDNN, we follow the same approach used in [14]. Specifically, given input feature vector $v_t \in \Re^N$, we window the first F elements from this feature, denoted by $x_{t,0} = v_t^{0:F} \in \Re^F$ and give this as input to the grid LSTM. At the next step, we stride the window over the input by S, and take the next F features, denoted by $x_{t,1} = v_t^{S:(F+S)} \in \Re^F$, and pass this to the grid LSTM. Hence the input to the grid LSTM (Equations (8) - (12)) at each time step t and frequency step k is given by $x_{t,k} = v_t^{k*S:(F+k*S)}$. In most cases $S < F$ so the chunks have overlapping information. The grid LSTM is thus unrolled over frequency by an amount $L = (N - F)/S + 1$.

At each time step t, the output of the gT-LSTM, denoted by $\{m_{t,0}^{(t)}, \ldots, m_{t,L}^{(t)}\}$ and gF-LSTM, denoted by $\{m_{t,0}^{(k)}, \ldots, m_{t,L}^{(k)}\}$ are concatenated together and given to the linear dimensionality reduction layer, followed by the LSTMs.

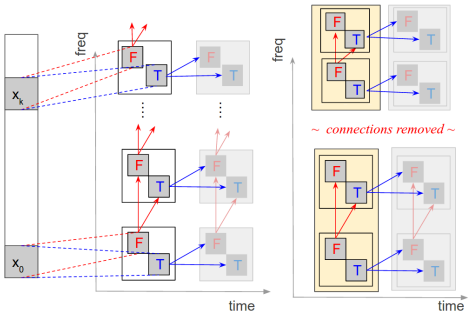


Figure 4: *Left: grid LSTM that consists of gT-LSTM (T) and gF-LSTM (F); Right: frequency block grid LSTM.*

2.3. Frequency Block Grid LSTM

A grid LSTM layer is similar to the convolution layer in that both models look over a small local time-frequency patch and share model parameters as the filter is shifted. One main difference is that the grid LSTM models frequency variations through a recurrent state that is passed along the frequency axis rather than convolving and pooling as CNNs do.

In this section, we modify the grid LSTM architecture to investigate how the recurrent connections contribute to the performance. Specifically, we split the entire frequency space into independent frequency blocks where only the connections within blocks are allowed while connections passing state across blocks is disabled. The modified grid LSTM architecture is referred as *frequency block grid LSTM*. Figure 4 (*right*) illustrates the processing flow of frequency block grid LSTM comparing to original grid LSTM (*left*). We analyze the impact of recurrent states by investigating the performance of frequency block grid LSTMs.

2.4. Output layer

In the output layer, we followed the end-of-query (EOQ) classifier [7] to directly predict when the user has finished speaking. Whereas a probabilistic VAD classifier makes a sequence of framewise binary speech / silence predictions, the EOQ classifier predicts one of 4 classes of speech / initial silence / intermediate silence / final silence and use the final silence for making mic closing decisions. Having 4 target classes helps the system capture any acoustic cues that may help differentiate silence at different points in the utterance, distinguishing between pre-query silence, mid-query pauses, and final silence.

3. Experimental Setup

3.1. Data

The proposed approach was trained and evaluated using farfield data collected from Google Home where the audio is often at 2-3 meters from the device, which makes this problem particularly challenging. Both training and evaluation data were recorded from two microphone channels. The training set contains 2 million (over 1,000 hours) English utterances. The testing set is based on roughly 20k utterances and composed of 6k clean utterances, 4k utterances with background speech at an average SNR of 13 dB, and 10k utterances with other noises at an SNR of 16 dB. The utterances are anonymized and hand-transcribed, and are representative of the Google Home traffic.

3.2. Model configurations

The acoustic features used for all experiments are 40-dimensional log-mel filterbank energies, computed using a 25ms window every 10ms. Table 1 shows the CLDNN and GLDNN configurations used in our experiments. Specifically, CLDNN covers 64 convolutional filters where the filter size is 8 frequency bands by one frame (10 ms) in time, and the stride of the filter is 1. We use a non-overlapping max pooling along frequency axis[16], with a pooling size of 3. For GLDNN, we use 12 grid LSTM units where the filter size was 8 frequency bands with stride 2 (overlap by 6). In both CLDNN and GLDNN topologies, the outputs of convolution layer or grid-LSTM layer are fed into an LDNN model [13]. The LDNN consists of a low-rank layer, followed by 2 time-LSTM layers with 64 cells, then one fully connected layer with 64 hidden units, and finally a softmax layer with 4 output targets.

All networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) [17]. The weights for DNN layers are initialized using the Glorot-Bengio strategy described in [18], while all LSTM parameters are uniformly initialized to lie between -0.02 and 0.02. We use a constant learning rate of 2e-5. Total number of parameters for GLDNN and CLDNN is between 100-120k with roughly 150k of multiplies plus adds.

Table 1: *Model configurations for CLDNN and GLDNN.*

CLDNN vs GLDNN		
frequency processing	convolution	grid LSTM
filter size (freq × time)	8 × 1	8 × 1
filter stride (freq × time)	1 × 1	2 × 1
pooling size (freq × time)	3 × 1	
number of filters	64	
number of grid LSTMs units		12
LDNN layers		
number of units per LSTM layer	64	64
number of units per DNN layer	64	64
computations & parameters		
parameters	116k	100k
MultAdd	~150k	~150k

4. Results

In this section we present a series of experimental results comparing the GLDNN and CLDNN endpointer models.

4.1. Results using GLDNN vs CLDNN

The ROC (receiver operating characteristics) curves are frequently used to describe a binary classification task. Here, we report the ROC curve (false rejection against false accept) for final silence classification. Lower curves are better. The posterior of final silence is thresholded to obtain the mic closing decision. We have a false accept when we predict final silence for a frame that belongs to one of the other 3 classes. A false rejection means we fail to predict final silence for a frame belonging to that class.

As shown in Figure 5, the GLDNN provides a 32% relative improvement in FA over the CLDNN at the operating point of 2% FR. Given that the task is to endpoint as soon as the user is done speaking, we also evaluate median latency relative to the end of the user-speech. In our experiments, we determine the

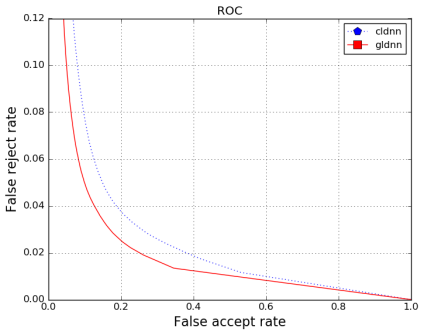


Figure 5: *ROC curve for GLDNN and CLDNN.*

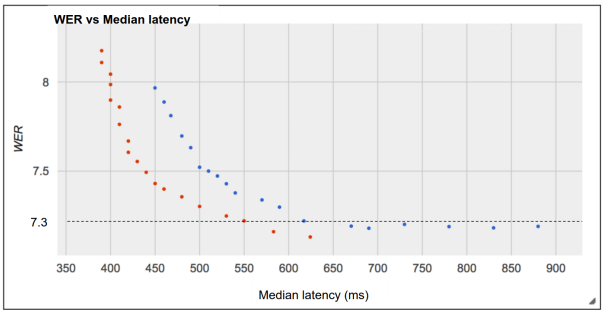


Figure 6: *WER vs median latency for CLDNN (blue) and GLDNN (red).*

"true" end of speech by running forced alignment and taking the time stamp of the last phoneme as true end-of-speech. The system makes a trade off between fast endpointing and avoiding cutting off the user. An aggressive decision threshold provides faster responds at the expense of hurting WERs while a better recognition accuracy increases latency. Figure 6 shows the WER vs latency curve for the GLDNN and CLDNN endpointers by varying the decision threshold. Again, lower curves are better. As shown in Figure 6, the GLDNN is better than the CLDNN and reduces median latency from 620 ms to 550 ms (11% relative improvement) with the same WER of 7.3%.

We also investigate how each system performs in different conditions at a fixed WER (see Table 2). The fixed WERs are selected to be consistent with current production using CLDNN. In Table 3, the GLDNN is consistently better than the CLDNN where we observe 10.5% latency reduction for clean cases, a 12.3% latency reduction in the presence of noise and a 8.7% improvement in the presence of background speech.

Table 2: *Fixed WERs for different conditions.*

	clean	bkg speech	bkg noise	overall
WER	5.6	12.3	7.2	7.3

4.2. Frequency block GLDNN

Given the substantial improvements from GLDNN compared to CLDNN, particularly in the presence of noise, we investigate the impact of modeling frequency variations in grid LSTMs by passing the recurrent state in frequency. One way to assess this is to evaluate using the frequency block gridLSTM (FB-GLDNN), described in Section 2.3.

Table 3: *Comparing CLDNN vs GLDNN latency for different conditions.*

Model	latency (ms)			
	clean	bkg speech	bkg noise	overall
CLDNN	570	690	650	620
GLDNN	510	630	570	550
rel. imp.	10.5%	8.7%	12.3%	11.3%

To train a frequency block GLDNN, we split the entire frequency space into 4 independent non-overlapping blocks (i.e. 10 frequency bands per block) where only connections within blocks are kept. In this sense, the grid LSTM layer behaves a bit more similar to convolution (but without pooling).

The results are in Table 4. We can see that frequency blocks caused 5% latency degradation for noisy case. This implies that passing the recurrent state is much more robust to model frequency variations, particularly in noise, compared to convolution-type methods.

Table 4: *Comparing frequency block GLDNN vs GLDNN latency for different conditions.*

Model	latency (ms)			
	clean	bkg speech	bkg noise	overall
GLDNN	510	630	570	550
FB-GLDNN	520	640	600	580
rel. regression	2.0%	1.6%	5.3%	5.5%

4.3. Two-channel GLDNN

Finally, we explore the performance of two-channel input where the observations are concatenated (i.e. 8 log-mel filterbank $\times$ 2 channels). Table 5 reveals that multi-channel input is particularly helpful for robustness to background speech improving the latency by about 8%.

Table 5: *Comparing single channel vs two channel GLDNN latency for different conditions.*

Model	latency (ms)			
	clean	bkg speech	bkg noise	overall
single-channel	510	630	570	550
two-channel	490	580	540	520
rel. imp.	3.9%	7.9%	5.3%	5.5%

5. Conclusions

In this work, we proposed a GLDNN model for endpointing and found that it can achieve 32% FA relative improvement when fixing FR at 2% and reduce median latency by 11% with respect to CLDNN baseline. By introducing frequency block to grid LSTM, we found that the recurrent connection along frequency axis is an important factor that improves the robustness to background noise. Finally, using two-channel input improves performance further in the presence of background speech. Overall, the proposed architecture reduced 16% (100 ms) latency.

6. References

[1] N. G. Ward, A. G. Rivera, K. Ward, and D. G. Novick, "Root causes of lost time and user stress in a simple dialog system," in *Interspeech*, 2005.

[2] A. Raux, D. Bohus, B. Langner, A. W. Black, and M. Eskenazi, "Doing research on a deployed spoken dialogue system: one year of lets go! experience." in *Interspeech*, 2006.

[3] R. Hariharan, J. Hakkinen, and K. Laurila, "Robust end-of-utterance detection for real-time speech recognition applications," in *ICASSP*, 2001.

[4] R. Zazo, T. N. Sainath, G. Simko, and C. Parada, "Feature learning with raw-waveform cldnns for voice activity detection," in *Proc. Interspeech*, 2016.

[5] S. Thomas, G. Saon, M. V. Segbroeck, and S. Naranyanan, "Improvements to the IBM speech activity detection system for the darpa rats program," in *Proc. ICASSP*, 2015.

[6] M. G. et al., "All for one: feature combination for highly channel-degraded speech activity detection," in *Proc. Interspeech*, 2013.

[7] G. Simko, M. Shannon, S. Chang, and C. Parada, "Alternative loss functions for training endpointers," in *Interspeech*, 2017.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735 – 1780, 1997.

[9] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *Proc. Interspeech*, 2014.

[10] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013.

[11] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. ASRU*, 2013.

[12] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, "Real-life voice activity detection with LSTM recurrent neural networks and an application to hollywood movies," in *Proc. ICASSP*, 2013.

[13] A. S. H. S. Tara N. Sainath, O. Vinyals, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. ICASSP*, 2015.

[14] T. N. Sainath and B. Li, "Modeling time-frequency patterns with LSTM vs. convolutional architectures for LVCSR tasks," in *Proc. Interspeech*, 2016.

[15] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid Long Short-Term Memory," in *in Proc. ICLR*, 2016.

[16] T. N. Sainath, B. Kingsbury, A. rahman Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. ASRU*, 2013.

[17] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Proc. NIPS*, 2012.

[18] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proc. AISTATS*, 2014.