Large Vocabulary Continuous Speech Recognition with Long Short-Term Recurrent Networks

Haşim Sak, Andrew Senior,
Oriol Vinyals, Georg Heigold, Erik McDermott,
Rajat Monga, Mark Mao, Françoise Beaufays
andrewsenior/hasim@google.com

Google

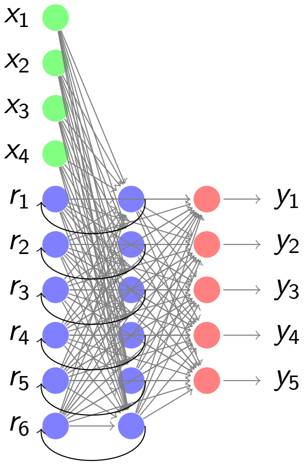
See Sak et al. [2014b,a]

Overview

- Recurrent neural networks
- Training RNNs
- Long short-term memory recurrent neural networks
- Distributed training of LSTM RNNs
- Acoustic modeling experiments
- Sequence training LSTM RNNs

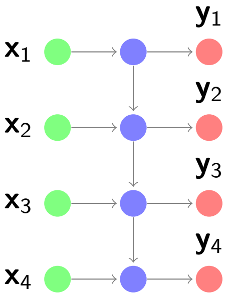
Recurrent neural networks

- An extension of feed-forward neural networks
- Output fed back as input with time delay.
- A dynamic time-varying neural network
- Recurrent layer activations encode a “state”.
- Sequence labelling, classification, prediction, mapping.
- Speech recognition [Robinson et al., 1993]



Back propagation through time

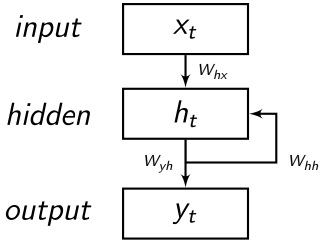
Unroll the recurrent network through time.



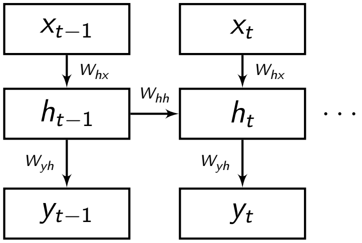
- Truncating at some limit “bptt_steps” it looks like a DNN.
- External gradients provided at the outputs
  - e.g. gradient of cross entropy loss
- Internal gradients computed with the chain rule (backpropagation).

Simple RNN

Simple RNN architecture in two alternative representations:



(a) RNN



(b) RNN unrolled in time

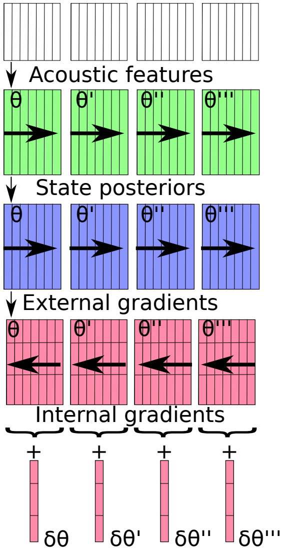
RNN hidden and output layer activations:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = \phi(W_{yh}h_t + b_y)$$

Training RNNs

- Forward pass: calculate activations for each input sequentially and update network state
- Backward pass: calculate error and back propagate through network and time (back-propagation through time (BPTT))
- Update weights with the gradients summed over all time steps for each weight
- Truncated BPTT: error is truncated after a specified back-propagation time steps

Backpropagation through time

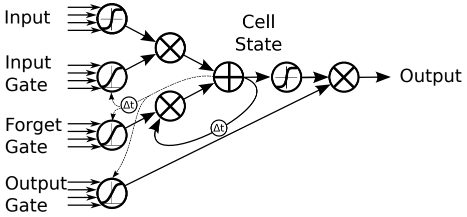


Long Short-Term Memory (LSTM) RNN

- Learning long-term dependencies is difficult with simple RNNs, unstable training due to vanishing gradients problem [Hochreiter, 1991]
- Limited capability (5-10 time steps) to model long-term dependencies
- LSTM RNN architecture designed to address these problems [Hochreiter and Schmidhuber, 1997]
- LSTM memory block: memory cell storing temporal state of network and 3 multiplicative units (gates) controlling the flow of information

Long Short-Term Memory Recurrent Neural Networks

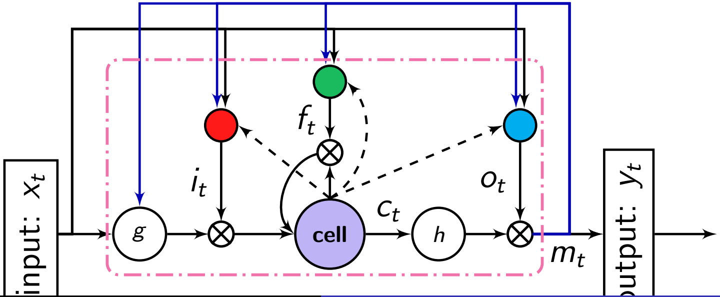
- Replace the units of an RNN with memory cells with sigmoid
  - Input gate
  - Forget gate
  - Output gate



- Enables long-term dependency learning
- Reduces the vanishing/exploding gradient problems
- 4× more parameters than RNN

LSTM RNN Architecture

Input gate: controls flow of input activations into cell
Output gate: controls output flow of cell activations
Forget gate: process continuous input streams [Gers et al., 2000]
"Peephole" connections added from cells to gates to learn precise timing of outputs [Gers et al., 2003]



LSTM RNN Related Work

- LSTM performs better than RNN for learning context-free and context-sensitive languages [Gers and Schmidhuber, 2001]
- Bidirectional LSTM for phonetic labeling of acoustic frames on the TIMIT [Graves and Schmidhuber, 2005]
- Online and offline handwriting recognition with bidirectional LSTM better than HMM-based system [Graves et al., 2009]
- Deep LSTM - stack of multiple LSTM layers - combined with CTC and RNN transducer predicting phone sequences gets state of the art results on TIMIT [Graves et al., 2013]

LSTM RNN Activation Equations

An LSTM network computes a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$ by calculating the network unit activations using the following equations iteratively from $t = 1$ to T:

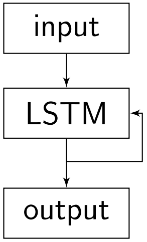
$$
\begin{align}
i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \tag{1}\\
f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \tag{2}\\
c_t &= f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{3}\\
o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{4}\\
m_t &= o_t \odot h(c_t) \tag{5}\\
y_t &= \phi(W_{ym}m_t + b_y) \tag{6}
\end{align}
$$

Proposed LSTM Projected (LSTMP) RNN

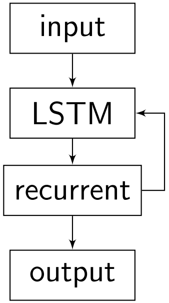
- $O(N)$ learning computational complexity with stochastic gradient descent (SGD) per time step
- Recurrent connections from cell output units (n_c) to cell input units, input gates, output gates and forget gates
- Cell output units connected to network output units
- Learning computational complexity dominated by $n_c \times (4 \times n_c + n_o)$ parameters
- For more effective use of parameters, add a recurrent projection layer with n_r linear projections ($n_r < n_c$) after LSTM layer.
- Now $n_r \times (4 \times n_c + n_o)$ parameters

LSTM RNN Architectures

LSTM RNN architectures



(a) LSTM



(b) LSTMP

LSTMP RNN Activation Equations

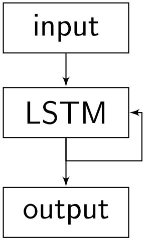
With the proposed LSTMP architecture, the equations for the activations of network units change slightly, the m_{t-1} activation vector is replaced with r_{t-1} and the following is added:

$$\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{im}r_{t-1} + W_{ic}c_{t-1} + b_i) && (7)\\
f_t &= \sigma(W_{fx}x_t + W_{fm}r_{t-1} + W_{fc}c_{t-1} + b_f) && (8)\\
c_t &= f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}r_{t-1} + b_c) && (9)\\
o_t &= \sigma(W_{ox}x_t + W_{om}r_{t-1} + W_{oc}c_t + b_o) && (10)\\
m_t &= o_t \odot h(c_t) && (11)\\
r_t &= W_{rm}m_t && (12)\\
y_t &= \phi(W_{yr}r_t + b_y) && (13)
\end{aligned}$$

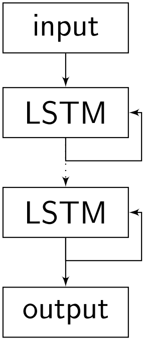
where the r denote the recurrent unit activations.

Deep LSTM RNN Architectures

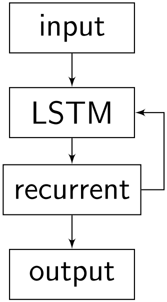
LSTM RNN architectures



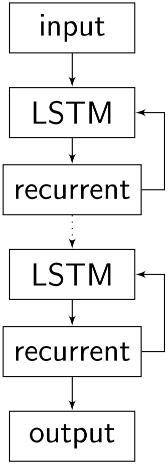
(a) LSTM



(b) DLSTM



(c) LSTMP

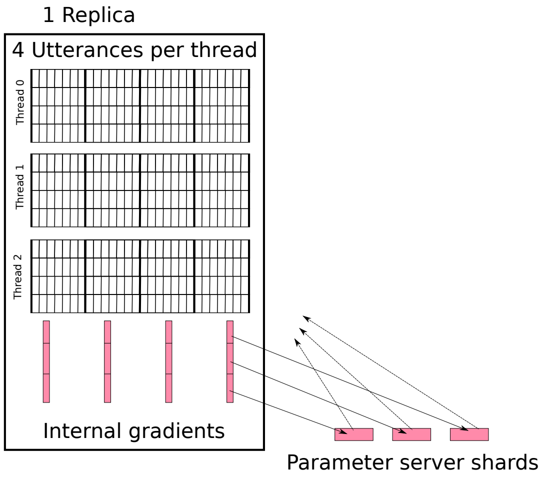


(d) DLSTMP

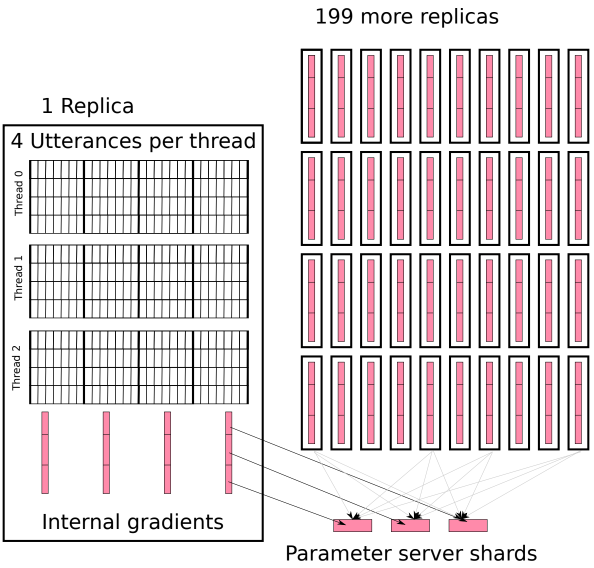
Distributed Training of LSTM RNNs

- Asynchronous stochastic gradient descent (ASGD) to optimize network parameters
- Google Brain's distributed parameter server: store, read and update the model parameters (50 shards)
- Training replicas on 200 machines (data parallelism)
- 3 synchronized threads in each machine (data parallelism)
- Each thread operating on mini batch of 4 sequences simultaneously
- TBPTT: 20 time steps of forward and backward pass
- Training: read fresh parameters, process $3 \times 4 \times 20$ time steps of input, send gradients to parameter server
- Clip cell activations to [-50, 50] range for long utterances

Asynchronous Stochastic Gradient Descent



Asynchronous Stochastic Gradient Descent



Asynchrony

Three forms of asynchrony:

- Within a replica every *bptt_steps* frame chunk is computed with different parameters.
    - State is carried over from one chunk to the next.
- Each replica is updating independently.
- Each shard of the parameter server is updated independently.

System

- Google Voice Search in US English
- 3M (1900hours) 8kHz anonymized training utterances
- 600M 25ms frames (10ms offset)
- Normalized 40-dimensional log-filterbank energy features
- 3-state HMMs with $14,000$ context-dependent states
- Cross-entropy loss
- Targets from DNN Viterbi forced-alignment
- 5 frame output delay
- Hybrid Unidirectional "DLSMTP"
- 2 layers of 800 cells with 512 linear projection layer.
- 13M parameters

Evaluation

- Scale posteriors by priors for inference.
- Deweight silence prior.
- Evaluate ASR on a test set of $22,500$ utterances
- First pass LM of 23 million *n*-grams, lattice rescoring with an LM of 1 billion 5-grams

Results for LSTM RNN Acoustic Models

WERs and frame accuracies on development and training sets: L number of layers, for shallow (1L) and deep (2,4,5,7L) networks C number of memory cells and N total number of parameters

C	Depth	N	Dev (%)	Train (%)	WER (%)
840	5L	37M	67.7	70.7	10.9
440	5L	13M	67.6	70.1	10.8
600	2L	13M	66.4	68.5	11.3
385	7L	13M	66.2	68.5	11.2
750	1L	13M	63.3	65.5	12.4

Results for LSTMP RNN Acoustic Models

WERs and frame accuracies on development and training sets: L number of layers, for shallow (1L) and deep (2,4,5,7L) networks C number of memory cells, P number of recurrent projection units, and N total number of parameters

C	P	Depth	N	Dev (%)	Train (%)	WER (%)
6000	800	1L	36M	67.3	74.9	11.8
2048	512	2L	22M	68.8	72.0	10.8
1024	512	3L	20M	69.3	72.5	10.7
1024	512	2L	15M	69.0	74.0	10.7
800	512	2L	13M	69.0	72.7	10.7
2048	512	1L	13M	67.3	71.8	11.3

LSTMP RNN models with various depths and sizes

C	P	Depth	N	WER (%)
1024	512	3L	20M	10.7
1024	512	2L	15M	10.7
800	512	2L	13M	10.7
700	400	2L	10M	10.8
600	350	2L	8M	10.9

Sequence training

- Conventional training minimizes the frame-level cross entropy between the output and the target distribution given by forced-alignment.
- Alternative criteria come closer to approximating the Word Error Rate and take into account the language model:
- Instead of driving the output probabilities closer to the targets, adjust the parameters to correct mistakes that we see in decoding actual utterances.
- Since these critera are computed on whole sequences we have sequence discriminative training [Kingsbury, 2009].
- e.g. Maximum Mutual Information or state-level Minimum Bayes Risk.

Sequence training criteria

Maximum mutual information is defined as:

$$F_{MMI}(\theta) = \frac{1}{T} \sum_u \log \frac{p_\theta(X_u|W_u)^\kappa p(W_u)}{\sum_W p_\theta(X_u|W)^\kappa p(W)}. \qquad (14)$$

State-level Minimum Bayes Risk (sMBR) is the expected frame state accuracy:

$$F_{sMBR}(\theta) = \frac{1}{T} \sum_u \sum_W \frac{p_\theta(X_u|W)^\kappa p(W)}{\sum_{W'} p_\theta(X_u|W')^\kappa p(W')} \delta(s, s_{ut}). \qquad (15)$$

Sequence training details

Discard frames with state occupancy close to zero, [Veselý et al., 2013]

Use a weak language model $p(W_u)$ and attach the reciprocal of the language model weight, κ, to the acoustic model.

No regularization. (Such as ℓ_2-regularization around the initial network) or smoothing such as the H-criterion [Su et al., 2013]

Computing gradients

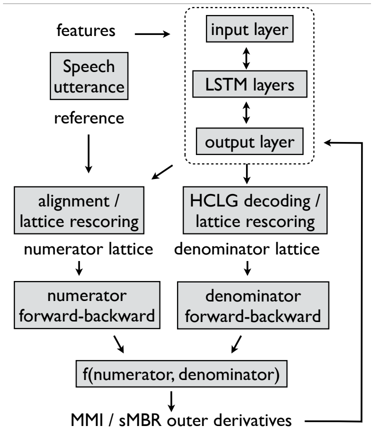
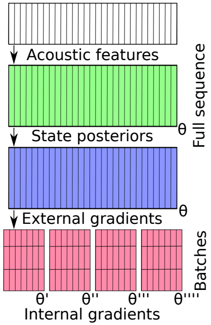

features
input layer
Speech utterance
LSTM layers
reference
output layer
alignment / lattice rescoring
HCLG decoding / lattice rescoring
numerator lattice
denominator lattice
numerator forward-backward
denominator forward-backward
f(numerator, denominator)
MMI / sMBR outer derivatives

Figure: Pipeline to compute outer derivatives.

Algorithm

```
$\mathcal{U} \leftarrow$ the data set of utterances with transcripts
$\mathcal{U} \leftarrow randomize(\mathcal{U})$
$\theta$ is the model parameters
for all $u \in \mathcal{U}$ do
    $\theta \leftarrow$ read from the parameter server
    calculate $\kappa l_{\theta,ut}^{(MMI/sMBR)}(s)$ for $u$
    for all $s \in$ subsequences($u$, bptt_steps) do
        $\theta \leftarrow$ read from the parameter server
        forward_pass($s$, $\theta$, bptt_steps)
        $\vec{\Delta}\theta \leftarrow$ backward_pass($s$, $\theta$, bptt_steps)
        $\Delta\theta \leftarrow$ sum_gradients($\vec{\Delta}\theta$, bptt_steps)
        send $\Delta\theta$ to the parameter server
    end for
```



Asynchronous sequence training system

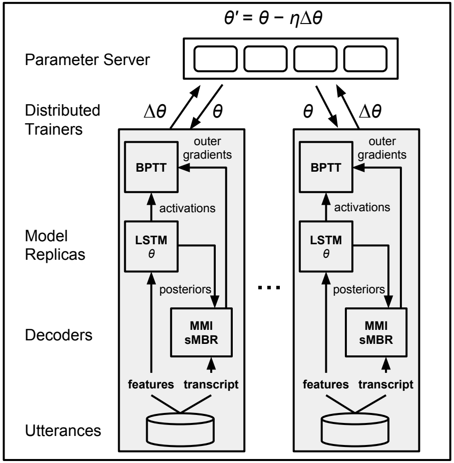


Figure: Asynchronous SGD: Model replicas asynchronously fetch parameters θ and push gradients $\Delta\theta$ to the parameter server.

Results: Choice of Language model

How powerful should the sequence training language model be?

Table: *WERs for sMBR training with LMs of various n-gram orders.*

CE	1-gram	2-gram	3-gram
10.7	10.9	10.0	10.1

Results

Table: *WERs for sMBR training of LSTM RNN bootstrapped with CE training on DNN versus LSTM RNN alignments.*

Alignment	CE	sMBR
DNN Alignment	10.7	10.1
LSTM RNN Alignment	10.7	10.0

Switching from CE to sequence training

Table: *WERs achieved by MMI/sMBR training for around 3 days when we switch from CE training at different times before convergence. * indicates the best WER achieved after 2 weeks of sMBR training.*

CE WER at switch	MMI	sMBR
15.9	13.8	-
14.9	12.0	-
12.0	10.8	10.7
11.2	10.8	10.3
10.7	10.5	10.0 (9.8*)

An 85M parameter DNN achieves 11.3% WER (CE) and 10.4% WER (sMBR).

Conclusions

- LSTMs for LVCSR outperform much larger DNNs, both CE (5%) and sequence-trained (6%).
- Distributed sequence training for LSTMs was a straight forward extension of DNN sequence training.
- LSTMs for LVCSR improved (8% relative) by sequence training
- sMBR gives better results than MMI.
- Sequence training needs to start from a converged model.

Ongoing work

- Alternative architectures
- Bidirectional
- Modelling units
- Other tasks
    - Noise robustness
    - Speaker ID
    - Language ID
    - Pronunciation modelling
    - Language modelling
    - Keyword spotting

Bibliography I

Felix A. Gers and Jürgen Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12 (10):2451–2471, 2000.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, March 2003. ISSN 1532-4435. doi: 10.1162/153244303768966139.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 12:5–6, 2005.

Alex Graves, Marcus Liwicki, Santiago Fernandez, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.137.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, 2013.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Master's thesis, T.U.München, 1991.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

B. Kingsbury. Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3761–3764, Taipei, Taiwan, April 2009.

A. J. Robinson, L. Almeida, J. m. Boite, H. Bourlard, F. Fallside, M. Hochberg, D. Kershaw, P. Kohn, Y. Konig, N. Morgan, J. P. Neto, S. Renals, M. Saerens, and C. Wooters. A neural network based, speaker independent, large vocabulary, continuous speech recognition system: The Wernicke project. In *PROC. EUROSPEECH'93*, pages 1941–1944, 1993.

H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao. Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *Proc. Interspeech 2014*, 2014a.

Hasim Sak, Andrew Senior, and Francoise Beaufays. Long Short-Term Memory Recurrent Neural Network architectures for large scale acoustic modeling. In *Proc. Interspeech 2014*, 2014b.

H. Su, G. Li, D. Yu, and F. Seide. Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6664–6668, 2013.

K. Veselý, A. Ghoshal, L. Burget, and D. Povey. Sequence-discriminative training of deep neural networks. In *INTERSPEECH*, 2013.