HIGH ORDER RECURRENT NEURAL NETWORKS FOR ACOUSTIC MODELLING

Chao Zhang & Phil Woodland

UNIVERSITY OF CAMBRIDGE

Machine Intelligence Laboratory, Cambridge University Engineering Department, Cambridge, United Kingdom

Abstract

- Vanishing long-term gradients is a major issue in RNN training.
- LSTM solves vanishing gradients with memory cells, but 4× as many parameters as an RNN.
- We propose to use high order (Markovian) RNNs (HORNN) with extra connections from multiple previous time steps.
- ASR experiments on English multi-genre broadcast (MGB3) data showed sigmoid and ReLU HORNNs reduced WERs by 6.3% and 4.2% over RNNs.
- HORNNs gave similar WERs to projected LSTMs (LSTMP) by using 20%–50% of the RNN parameters and computation.

Markov Conditional Independence

- A standard RNN layer is defined as

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}),$$

where
  - $\mathbf{h}_t$ and $\mathbf{x}_t$ are the output and input vectors at time t;
  - $\mathbf{W}$ and $\mathbf{U}$ are weight matrices, $\mathbf{b}$ is bias vector;
  - $f(\cdot)$ is the activation function.
- RNNs have the **1st-order Markov conditional independence property** (detailed proof given in paper).
  - $\mathbf{h}_t$ is continuous-valued hidden state;
  - $\mathbf{h}_t$ depends on only $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$;
  - Property also applies to bidirectional RNNs by viewing $\mathbf{h}_t^{\text{bid}} = \{\mathbf{h}_t^{\text{fwd}}, \mathbf{h}_t^{\text{bwd}}\}$.
- RNN language models (LM) have std. 1st-order Markov property.

HORNNs for Sigmoid and ReLU Functions

- Vanishing graidents can be solved by relaxing the 1st-order Markov conditional independence constraint.
- Use n-order Markov property by including $\mathbf{h}_{t-n}$ in $\mathbf{h}_t$ calculation.
  - To directly access to long-term information in testing.
  - To create shortcuts to allow additional long-term information to flow more easily in training.
- The gradients of an objective function $\mathcal{F}$ w.r.t. $\mathbf{h}_{t-1}$ is

$$\frac{\partial \mathcal{F}}{\partial \mathbf{h}_{t-1}} = \sum_{i=1}^{n} \frac{\partial \mathcal{F}}{\partial \mathbf{h}_{t-i-1}} \frac{\partial \mathbf{h}_{t-i-1}}{\partial \mathbf{h}_{t-1}}$$

that alleviates long-term information vanishing by integrating it explicitly at each time step.

- Many ways of using $\mathbf{h}_{t-n}$ in calculating $\mathbf{h}_t$ (e.g. pooling, gating *etc.*). We focus on using $\mathbf{h}_{t-n}$ as input to a HORNN layer.
- Standard HORNN model structures are
  - ReLU: $\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}_1\mathbf{h}_{t-1} + \mathbf{U}_n\mathbf{h}_{t-n} + \mathbf{b})$
  - Sigmoid: $\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}_1\mathbf{h}_{t-1} + \mathbf{U}_n\mathbf{h}_{t-n} + \mathbf{h}_{t-m} + \mathbf{b})$
  - Sigmoid HORNN requires to add extra $\mathbf{h}_{t-m}$ since it suffers more from vanishing gradients.

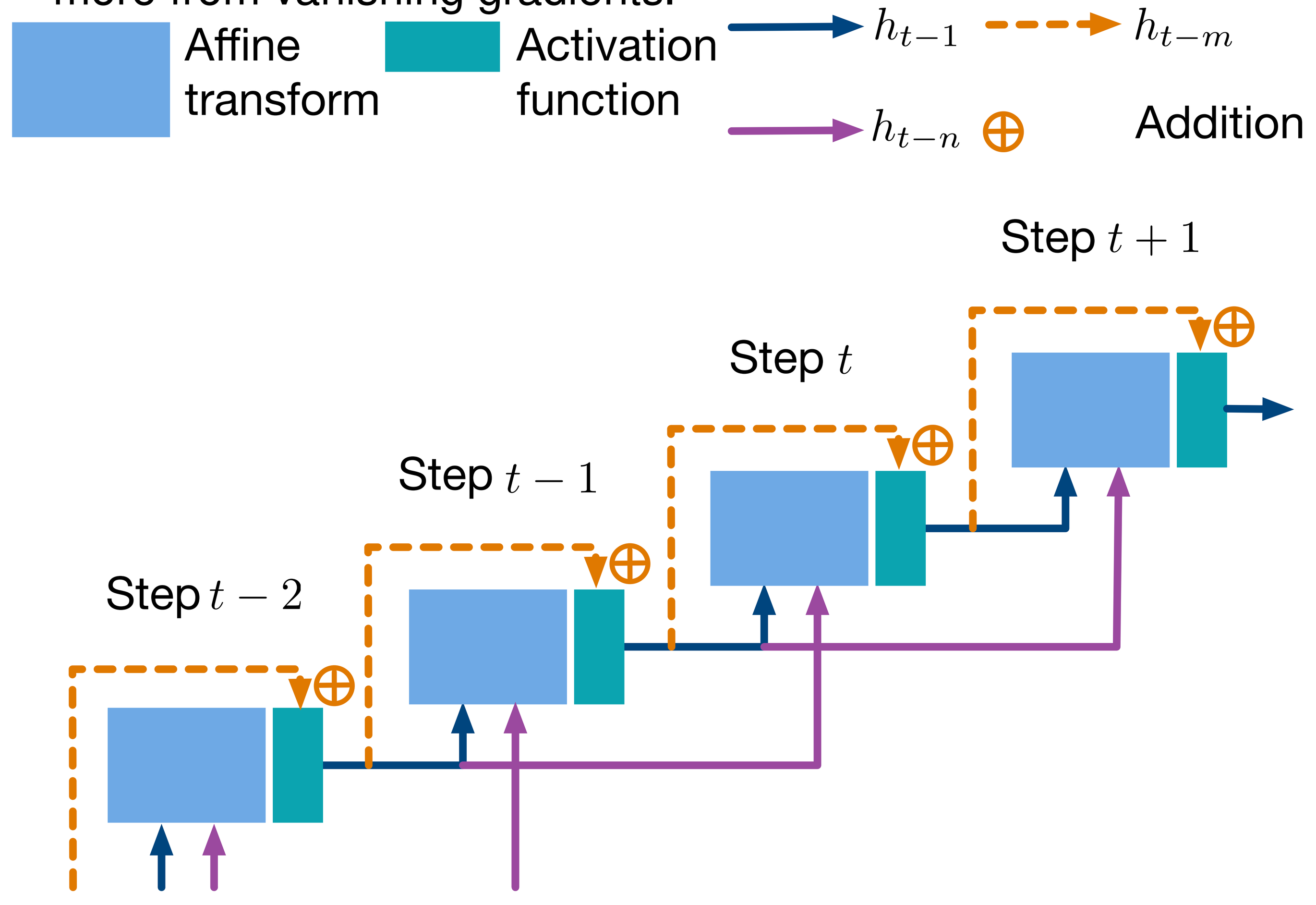

Figure 1: The HORNN structure unfolded through time steps (t).

- Like LSTMP, $\mathbf{U}$ matrices can be factorised with a shared linear projection $\mathbf{P}$ that results in projected HORNN (HORNNP).
  - ReLU: $\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}_{p1}\mathbf{P}\mathbf{h}_{t-1} + \mathbf{U}_{pn}\mathbf{P}\mathbf{h}_{t-n} + \mathbf{b})$
  - Sigmoid: $\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}_{p1}\mathbf{P}\mathbf{h}_{t-1} + \mathbf{U}_{pn}\mathbf{P}\mathbf{h}_{t-n} + \mathbf{h}_{t-m} + \mathbf{b})$

Computation and Storage Complexities

- Comparing to LSTM/LSTMP with the same $\mathbf{h}_t$ and $\mathbf{P}\mathbf{h}_{t-1}$ sizes, HORNN/HORNNP often uses less than 50% #params.
  - An LSTM layer uses 4 times more parameters than an RNN layer to implement the memory cells.
  - A HORNN layer has lower cost than 2 RNN layers.
  - It saves more than 50% #calculations & storage space in both training and testing.
- Residual RNNs (ResRNN) were also evaluated as baselines.
  - $\mathbf{h}_t = f(\mathbf{U}_{d2}f(\mathbf{W}\mathbf{x}_t + \mathbf{U}_{d1}\mathbf{h}_{t-1} + \mathbf{b}) + \mathbf{h}_{t-m})$
  - ResRNN is slower than HORNN since matrices cannot be multiplied in parallel.

Experimental Setup

- Experiments used 55 hour and 275 hour MGB3 challenge data for training and a 5.6 hour test set, **dev17b**.
- 63k word dictionary + trigram LM (**tg**) and confusion network decoding (**cn**).
- All models used cross-entropy training with extended HTK 3.5.

55 Hour Single Layer HORNN Experiments

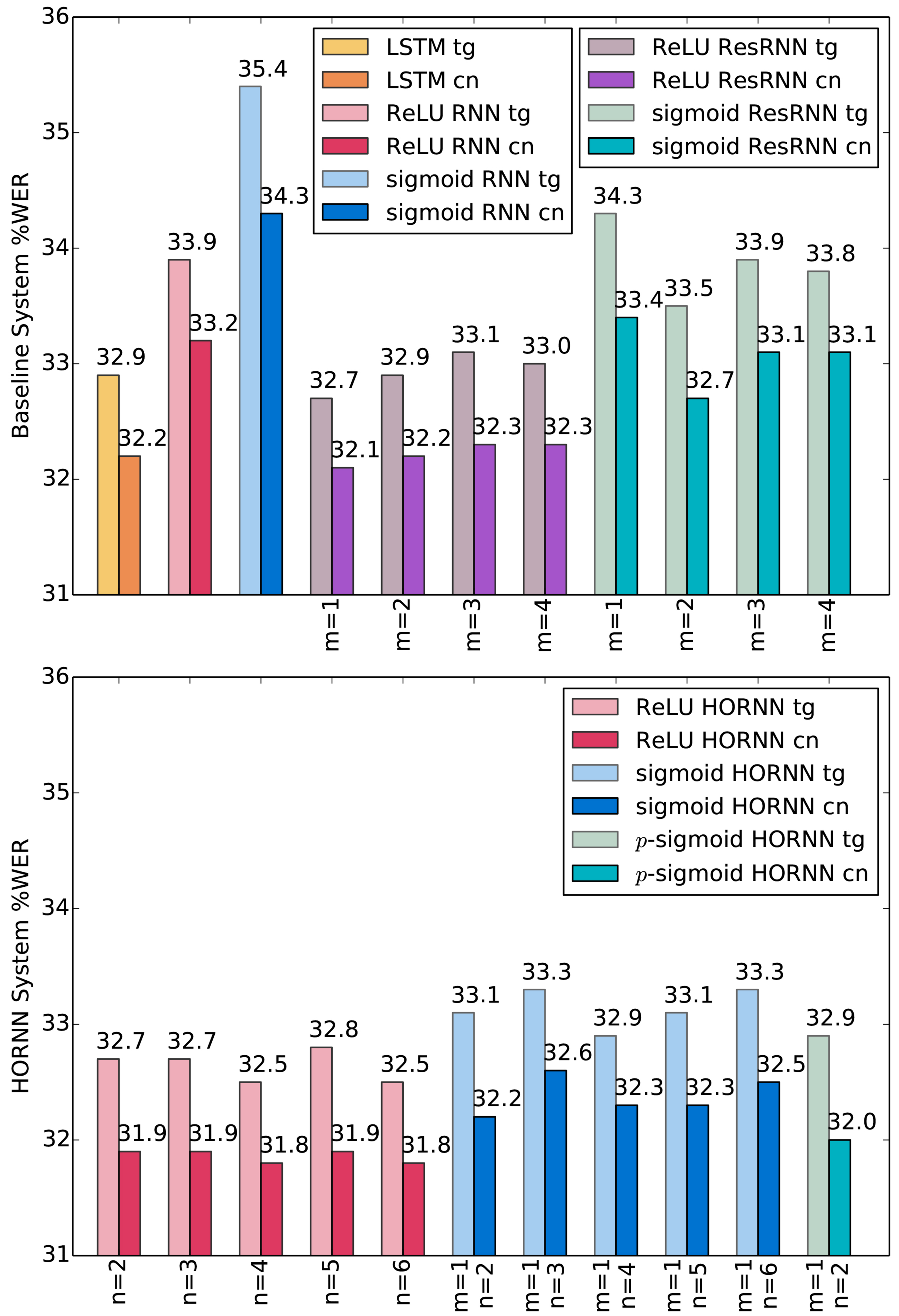

Figure 2: %WERs for systems with $\mathbf{h}_t$ size 500.

- HORNNs (0.5M) gave similar WERs to LSTMs (1.2M).
- Sigmoid HORNNs had lower WERs than ResRNNs (0.5M).
- Use $n = 4$ for ReLU and $m = 1$, $n = 2$ for sigmoid HORNNs.

55 Hour Multi-Layered HORNNP Experiments

System	#Layers	#Params	tg	cn
LSTMP	1L	1.1M	32.7	32.0
	2L	1.9M	31.3	30.6
ReLU HORNNP	1L	0.2M	32.5	31.8
	1L	0.4M	32.0	31.4
	1L	1.0M	31.4	30.7
	2L	0.9M	31.4	30.7
sigmoid HORNNP	1L	0.2M	33.0	32.1
	1L	0.4M	32.8	31.9
	1L	1.0M	31.6	30.9
	2L	0.9M	31.4	30.7

Table 1: LSTMP & HORNNP %WERs with different depths, $\mathbf{h}_t$ and $\mathbf{P}\mathbf{h}_{t-1}$ sizes.

- For 1L/2L HORNNs to LSTMPs,
  - projection $\mathbf{P}$ reduced $\mathbf{h}_t$ size by 2 or by 4;
  - HORNNPs used far fewer RNN layer parameters (#params) without increasing WER;
  - fewer parameters improve speed and save space.
- By going wider or deeper with similar #params, HORNNs outperformed LSTMPs by ~4% relative WER reduction.

275 Hour Validation Experiments

System	#Layers	tg	cn	#Layers	tg	cn
LSTMP	1L	26.5	26.0	2L	25.7	25.2
ReLU HORNNP	1L	26.4	25.9	2L	25.3	25.0
sigmoid HORNNP	1L	26.4	25.8	2L	25.6	25.2

Table 2: LSTMP & HORNNP %WERs with $\mathbf{h}_t$ size 1000 and $\mathbf{P}\mathbf{h}_{t-1}$ size 500.

- ~3% relative WER reduction by going from 1L to 2L recurrent layers for both LSTMP and HORNNP.
- HORNNPs still produced similar WERs to LSTMPs by using far fewer #params.
- 7L sigmoid DNN produced WERs of 28.4 (tg) and 27.5 (cn).

Conclusions

- Proposed to use high order (Markov) connections to address the RNN long-term vanishing gradient issue.
- Proposed two HORNNs structures for sigmoid and ReLU.
- HORNNs yielded 4%–6% WER reductions over std. RNNs.
- Comparing to LSTMs/LSTMPs, HORNNs/HORNNPs
  - gave similar WERs, and used far less computation & space;
  - had 4% relative WER reduction with similar #params.

E-Mail: {cz277,pcw}@eng.cam.ac.uk