Wider and Deeper, Cheaper and Faster: Tensorized LSTMs for Sequence Learning

Zhen He[1,2], Shaoging Gao[3], Liang Xiao[1], Daxue Liu[1], Hangen He[1], David Barber[2,4]

[1]National University of Defense Technology [2]University College London [3]Sichuan University [4]Alan Turing Institute

1. Introduction

Motivation:
Increasing the capacity of RNNs

How to increase their capacity?
Widening the network | Deepening the network

Common solutions:
Increasing the size of the hidden layer | Stacking multiple hidden layers

Drawbacks:
Parameter number scales quadratically | Runtime scales linearly | Vanishing/Exploding gradients

Our solutions:
Tensorizing the hidden layer and sharing parameters | Performing deep computations through time | Extending to LSTMs

Advantages:
Parameter number is unchanged | Runtime is almost unchanged | Vanishing/Exploding gradients are alleviated

2. Method

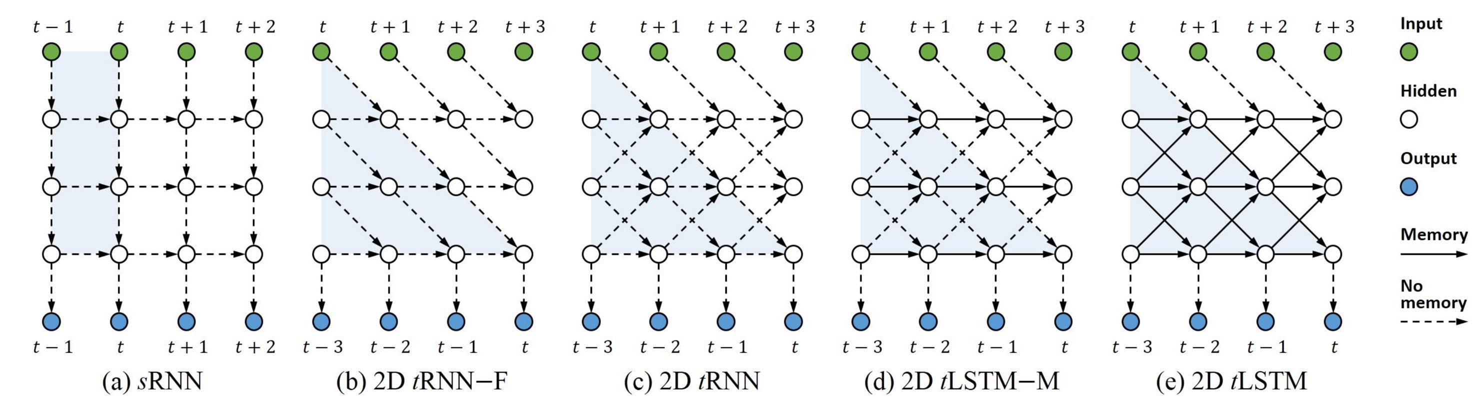

Figure 1: Examples of sRNN, tRNNs and tLSTMs. (a) A 3-layer sRNN. (b) A 2D tRNN without (–) feedback (F) connections, which can be thought as a *skewed* version of (a). (c) A 2D tRNN. (d) A 2D tLSTM without (–) memory (M) cell convolutions. (e) A 2D tLSTM. In each model, the blank circles in column 1 to 4 denote the hidden state at timestep $t-1$ to $t+2$, respectively, and the blue region denotes the receptive field of the current output $\boldsymbol{y}_t$. In (b)-(e), the outputs are delayed by $L-1=2$ timesteps, where $L=3$ is the depth.

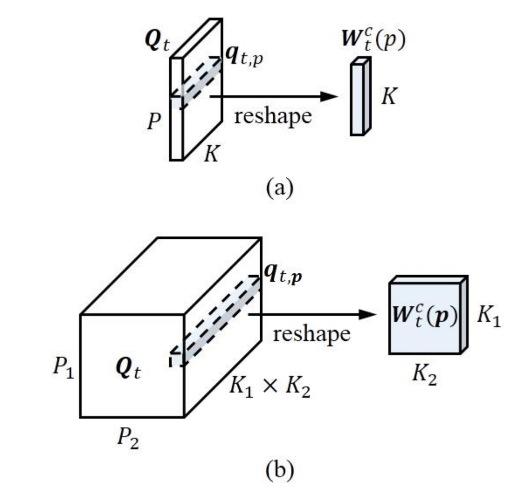

Figure 2: Illustration of generating the memory cell convolution kernel, where (a) is for 2D tensors and (b) for 3D tensors.

Concatenating the input:

For vector hidden states

$$\boldsymbol{h}^{cat}_{t-1} = [\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]$$

For 2-D hidden states

$$\boldsymbol{h}^{cat}_{t-1,p} = \begin{cases} \boldsymbol{x}_t\boldsymbol{W}^x + \boldsymbol{b}^x & \text{if } p=1 \\ \boldsymbol{h}_{t-1,p-1} & \text{if } p>1 \end{cases}$$

For N-D hidden states

$$\boldsymbol{h}^{cat}_{t-1,\boldsymbol{p}} = \begin{cases} \boldsymbol{x}_t\boldsymbol{W}^x + \boldsymbol{b}^x & \text{if } p_d=1 \text{ for } d=1,2,\ldots,D-1 \\ \boldsymbol{h}_{t-1,\boldsymbol{p-1}} & \text{if } p_d>1 \text{ for } d=1,2,\ldots,D-1 \\ \boldsymbol{0} & \text{otherwise} \end{cases}$$

Updating the hidden state:

Vanilla RNN

$$\boldsymbol{a}_t = \boldsymbol{h}^{cat}_{t-1}\boldsymbol{W}^h + \boldsymbol{b}^h$$
$$\boldsymbol{h}_t = \phi(\boldsymbol{a}_t)$$

tRNN:

$$\boldsymbol{A}_t = \boldsymbol{H}^{cat}_{t-1} \circledast \{\boldsymbol{W}^h, \boldsymbol{b}^h\}$$
$$\boldsymbol{H}_t = \phi(\boldsymbol{A}_t)$$

tLSTM-M:

$$[\boldsymbol{A}^g_t, \boldsymbol{A}^i_t, \boldsymbol{A}^f_t, \boldsymbol{A}^o_t] = \boldsymbol{H}^{cat}_{t-1} \circledast \{\boldsymbol{W}^h, \boldsymbol{b}^h\}$$
$$[\boldsymbol{G}_t, \boldsymbol{I}_t, \boldsymbol{F}_t, \boldsymbol{O}_t] = [\phi(\boldsymbol{A}^g_t), \sigma(\boldsymbol{A}^i_t), \sigma(\boldsymbol{A}^f_t), \sigma(\boldsymbol{A}^o_t)]$$
$$\boldsymbol{C}_t = \boldsymbol{G}_t \odot \boldsymbol{I}_t + \boldsymbol{C}_{t-1} \odot \boldsymbol{F}_t$$
$$\boldsymbol{H}_t = \phi(\boldsymbol{C}_t) \odot \boldsymbol{O}_t$$

tLSTM:

$$[\boldsymbol{A}^g_t, \boldsymbol{A}^i_t, \boldsymbol{A}^f_t, \boldsymbol{A}^o_t, \boldsymbol{A}^q_t] = \boldsymbol{H}^{cat}_{t-1} \circledast \{\boldsymbol{W}^h, \boldsymbol{b}^h\}$$
$$[\boldsymbol{G}_t, \boldsymbol{I}_t, \boldsymbol{F}_t, \boldsymbol{O}_t, \boldsymbol{Q}_t] = [\phi(\boldsymbol{A}^g_t), \sigma(\boldsymbol{A}^i_t), \sigma(\boldsymbol{A}^f_t), \sigma(\boldsymbol{A}^o_t), \varsigma(\boldsymbol{A}^q_t)]$$
$$\boldsymbol{W}^c_t(p) = \text{reshape}\left(\boldsymbol{q}_{t,p}, [K,1,1]\right)$$
$$\boldsymbol{C}^{conv}_{t-1} = \boldsymbol{C}_{t-1} \circledast \boldsymbol{W}^c_t(p)$$
$$\boldsymbol{C}_t = \boldsymbol{G}_t \odot \boldsymbol{I}_t + \boldsymbol{C}^{conv}_{t-1} \odot \boldsymbol{F}_t$$
$$\boldsymbol{H}_t = \phi(\boldsymbol{C}_t) \odot \boldsymbol{O}_t$$

Generating the output:

For vector hidden states

$$\boldsymbol{y}_t = \varphi(\boldsymbol{h}_t\boldsymbol{W}^y + \boldsymbol{b}^y)$$

For 2-D hidden states

$$\boldsymbol{y}_t = \varphi(\boldsymbol{h}_{t+L-1,P}\boldsymbol{W}^y + \boldsymbol{b}^y)$$

For N-D hidden states

$$\boldsymbol{y}_t = \varphi(\boldsymbol{h}_{t+L-1,\boldsymbol{P}}\boldsymbol{W}^y + \boldsymbol{b}^y)$$

3. Experiments

Comparison of different configurations

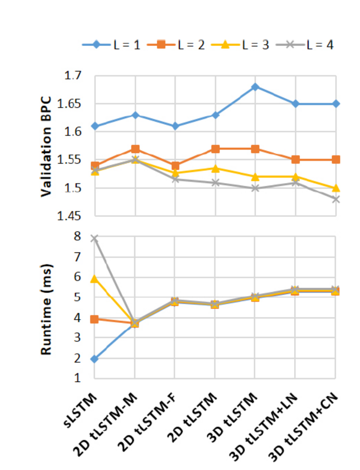

Figure 3: Performance and runtime of different configurations on Wikipedia.

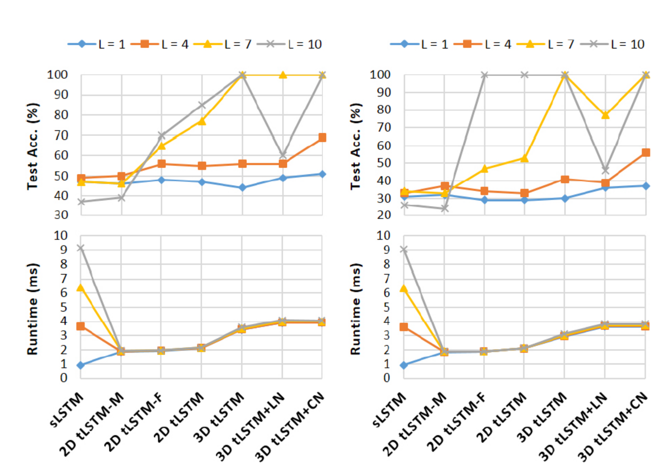

Figure 4: Performance and runtime of different configurations on the addition (left) and memorization (right) tasks.

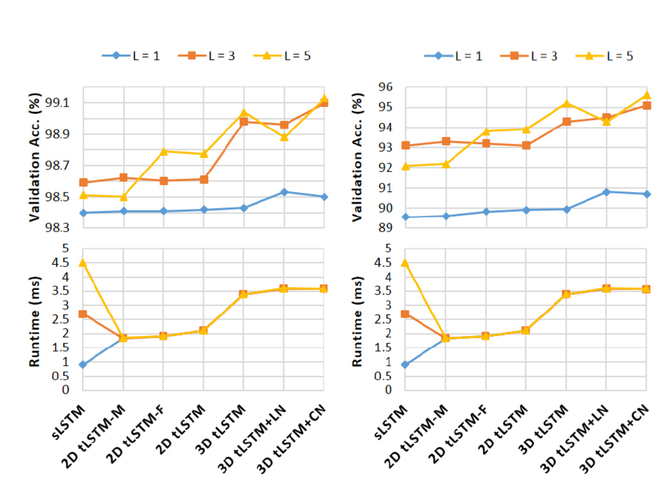

Figure 5: Performance and runtime of different configurations on sequential MNIST (left) and sequential pMNIST (right).

- Wider and deeper networks perform better.
- Parameter number and runtime are invariant.
- Memory cell convolutions are crucial to maintain improvement.
- Feedback/tensorization/CN is useful.

Comparison to the state-of-the-art methods

Table 1: Test BPC on Wikipedia.

	BPC	# Param.
MI-LSTM [51]	1.44	≈17M
mLSTM [32]	1.42	≈20M
HyperLSTM+LN [23]	1.34	26.5M
HM-LSTM+LN [11]	1.32	≈35M
Large RHN [54]	1.27	≈46M
Large FS-LSTM-4 [38]	1.245	≈47M
2 × Large FS-LSTM-4 [38]	**1.198**	≈94M
3D tLSTM+CN ($L=6$, $M=1200$)	1.264	50.1M

Table 2: Test accuracies on two algorithmic tasks.

	Addition		Memorization	
	Acc.	# Samp.	Acc.	# Samp.
Stacked LSTM [21]	51%	5M	>50%	900K
Grid LSTM [30]	>99%	550K	>99%	150K
3D tLSTM+CN ($L=7$)	>99%	**298K**	>99%	115K
3D tLSTM+CN ($L=10$)	>99%	317K	>99%	**54K**

Table 3: Test accuracies (%) on sequential MNIST/pMNIST.

	MNIST	pMNIST
iRNN [33]	97.0	82.0
LSTM [2]	98.2	88.0
uRNN [2]	95.1	91.4
Full-capacity uRNN [49]	96.9	94.1
sTANH [53]	98.1	94.0
BN-LSTM [13]	99.0	95.4
Dilated GRU [8]	**99.2**	94.6
Dilated CNN [40] in [8]	98.3	**96.7**
3D tLSTM+CN ($L=3$)	**99.2**	94.9
3D tLSTM+CN ($L=5$)	99.0	95.7

Visualization of the tLSTM memory cells

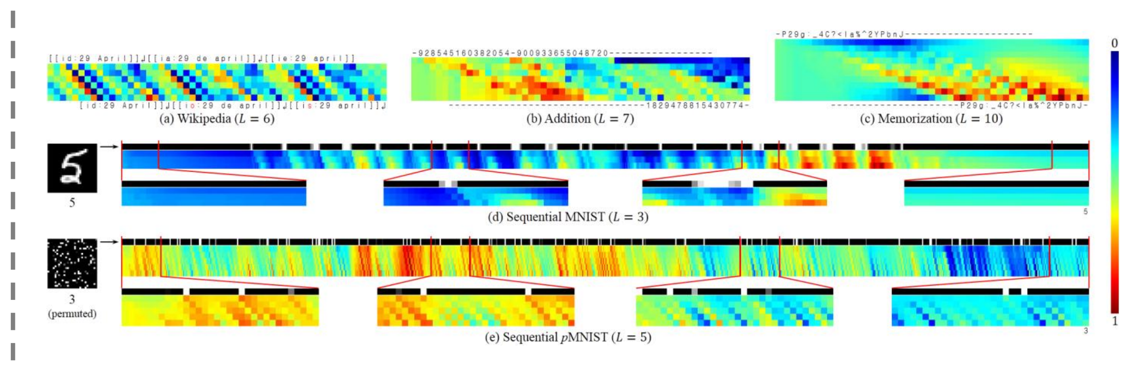

Figure 6: Visualization of the diagonal channel means of the tLSTM memory cells for each task. In each horizontal bar, the rows from top to bottom correspond to the diagonal locations from $\boldsymbol{p}^{in}$ to $\boldsymbol{p}^{out}$, the columns from left to right correspond to different timesteps (from 1 to $T+L-1$ for the full sequence, where $L-1$ is the time delay), and the values are normalized to be in range $[0, 1]$ for better visualization. Both full sequences in (d) and (e) are zoomed out horizontally.

- Wider (larger) tensors can encode more information, with less effort to compress it.
- Deep computations are indeed performed together with temporal computations, with long-range dependencies carried by memory cells.

4. Related Work

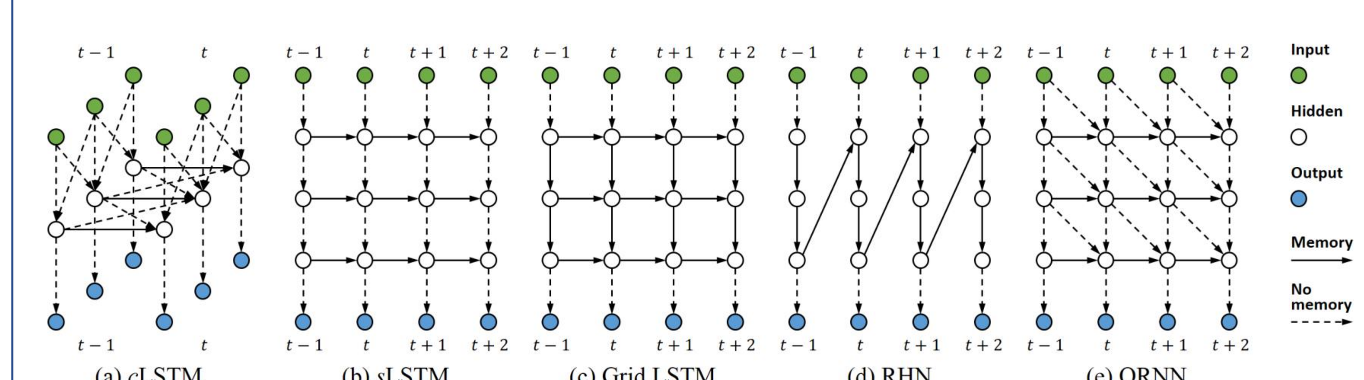

Figure 7: Examples of models related to tLSTMs. (a) A single layer cLSTM [48] with vector array input. (b) A 3-layer sLSTM [21]. (c) A 3-layer Grid LSTM [30]. (d) A 3-layer RHN [54]. (e) A 3-layer QRNN [7] with kernel size 2, where costly computations are done by temporal convolution.

- Convolutional LSTMs (a) are for structured input.
- Stacked/Deep LSTMs (b, c, and d) typically multiply the runtime.
- Temporal parallelization (e) is potentially unsuitable for real-time online inference.