RNN Variants

Cao Ying

Outlines

1. *Motivations and background*
    - Challenges of RNNs from the ML perspective
    - Challenges of RNNs from the system perspective
2. *RNN variants*
    - Dilated LSTM (NIPS 2017)
    - ClockworkRNN (2014)
    - Multi-dimensional RNN (2007)
    - Grid LSTM (2015)
    - Hierarchical Multiscale Recurrent Neural Networks (ICLR 2017)
    - ON-LSTM (ICLR 2019's best paper award)
3. *Summary & Disscussion*

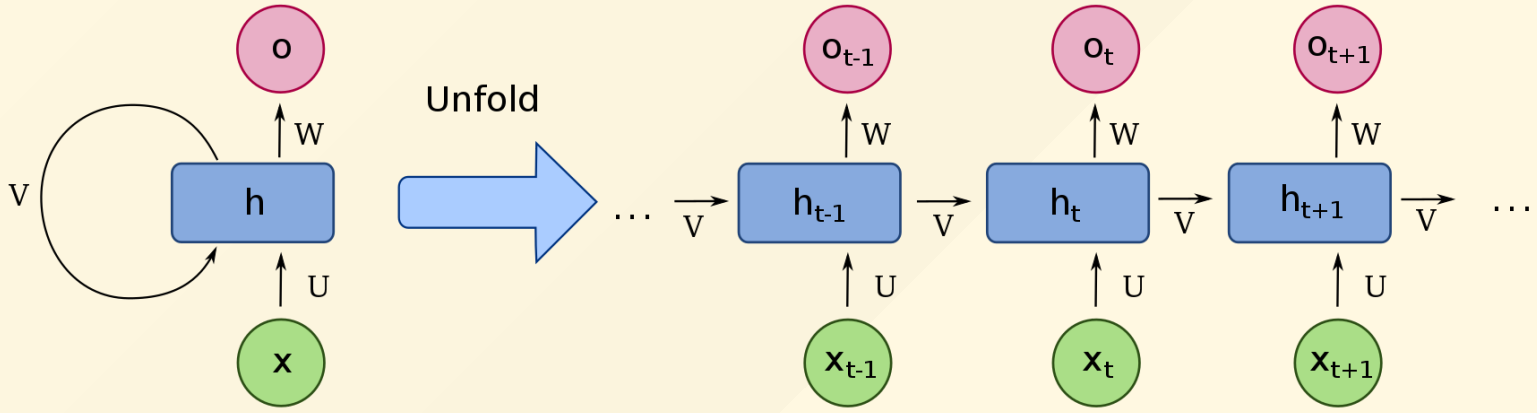
Motivations

1. RNNs are notorious for their sequential execution natures. Nevertheless, is there no parallelism at all in recurrent models?
2. RNN is a wide spectrum of models more than LSTM/GRU. Is there common semantics to describe general-purpose RNN computation, and the description is able to be optimized?

Background: Challenges of sequence modeling

1. The vanishing gradient problems is originated from BP algorithm.
    - Existing solutions
        1. Truncated BPTT
        2. gate
        3. skip connections
        4. normalization
2. How to design RNNs to adaptively learn patterns span over multiple and variable time step?
3. How to design RNNs to adaptively learn underlying hierarchically structured features?

Recap RNN computation



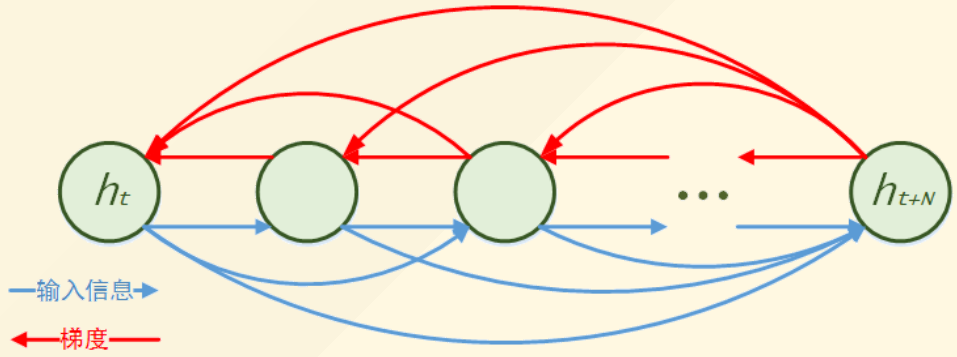
1. All RNNs are made up of three parts:
    1. *input-to-hidden* projection U
    2. *hidden-to-hidden* projection V
    3. *hidden-to-output* projection W
2. hidden-to-hidden projection U is intrinsically sequential.
3. the computation of RNN is dominated by MMs, and GEMM works poorly for small MMs.

Related work for RNN optimizations

1. Investigating performance of GPU BLAS Libraries
2. Persistent RNNs: 30 times faster RNN layers at small mini-batch sizes
3. DeepCpu

Gradient vanishing/explosion problem and LSTM

RNNs form a long chain over which gradient flows.



References

1. waybackprop

Recap the standard LSTM cell

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o)$$

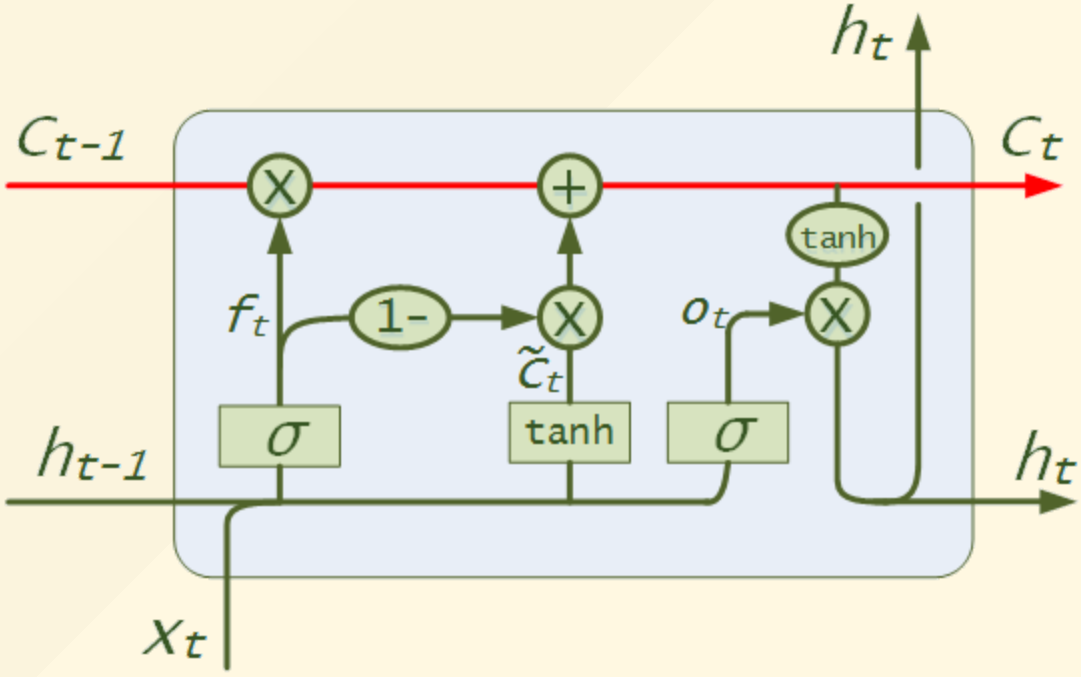
$$\hat{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \hat{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

Summary: csore of LSTM

1. Introduce the cell memory (GRU has no cell).
2. In LSTM, history information flow along with the redline, and no non-linear squashing function is applied.



Question: are there no parallelism at all for RNNs?

1. Stacked multiple RNNs
2. Bidirection

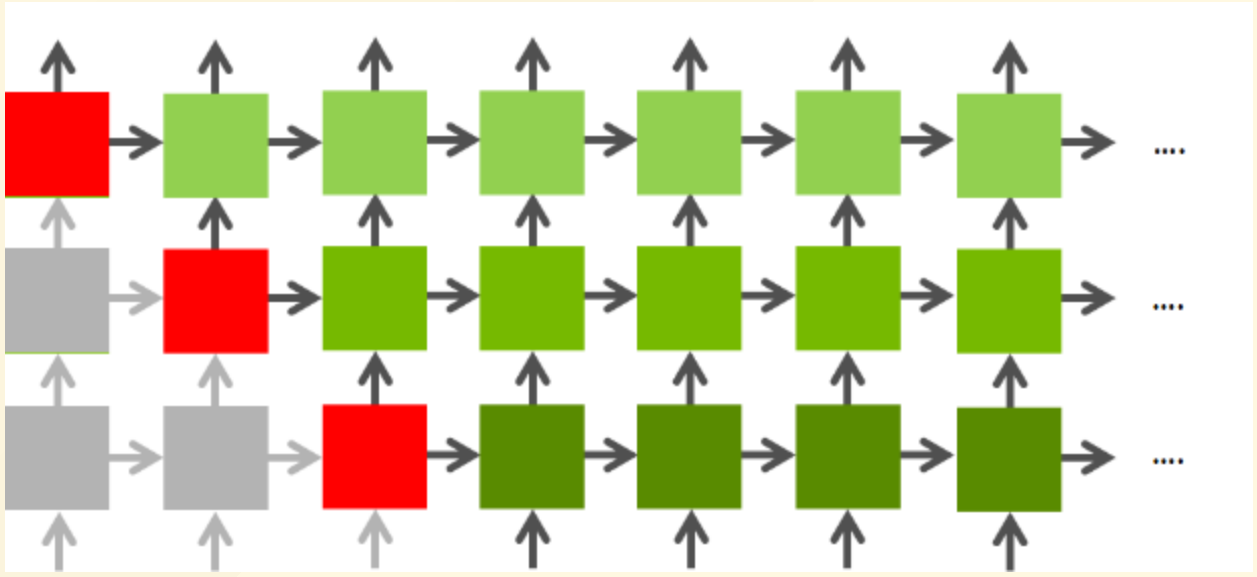


Fig. As dependencies are resolved a wavefront of operations moves through the network.

- *This picture is from the blog [Optimizing Recurrent Neural Networks in cuDNN 5]*

RNN Variants

Dilated RNN (NIPS 2017)

Problem to address

1. complex dependencies in long sequence
2. vanishing and exploding gradients
3. efficient parallelization.

Approach

- multi-resolution dilated recurrent skip connections that can be applied to any RNN cell

Dilated RNN (II)

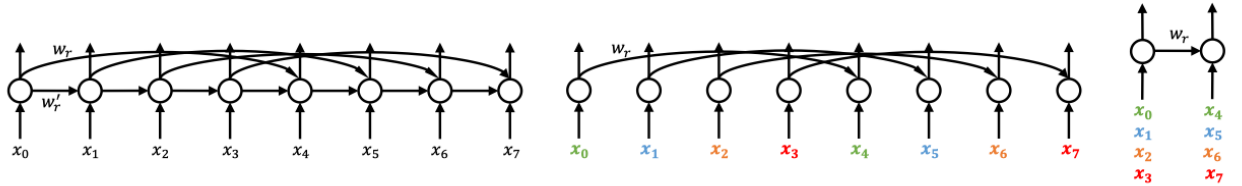


Figure 1: (left) A single-layer RNN with recurrent skip connections. (mid) A single-layer RNN with dilated recurrent skip connections. (right) A computation structure equivalent to the second graph, which reduces the sequence length by four times.

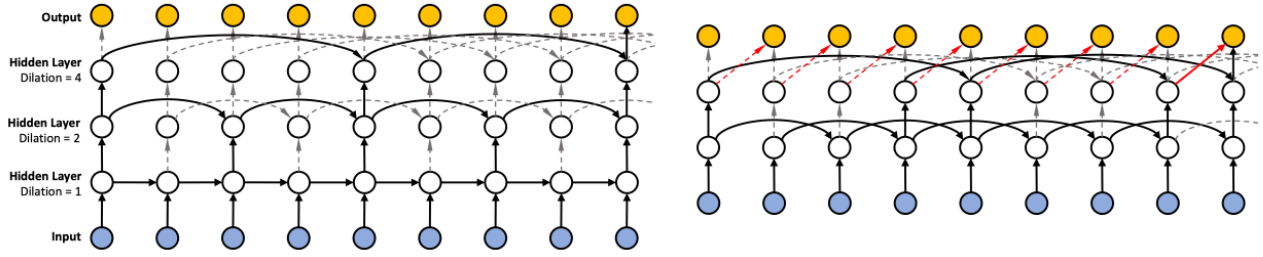


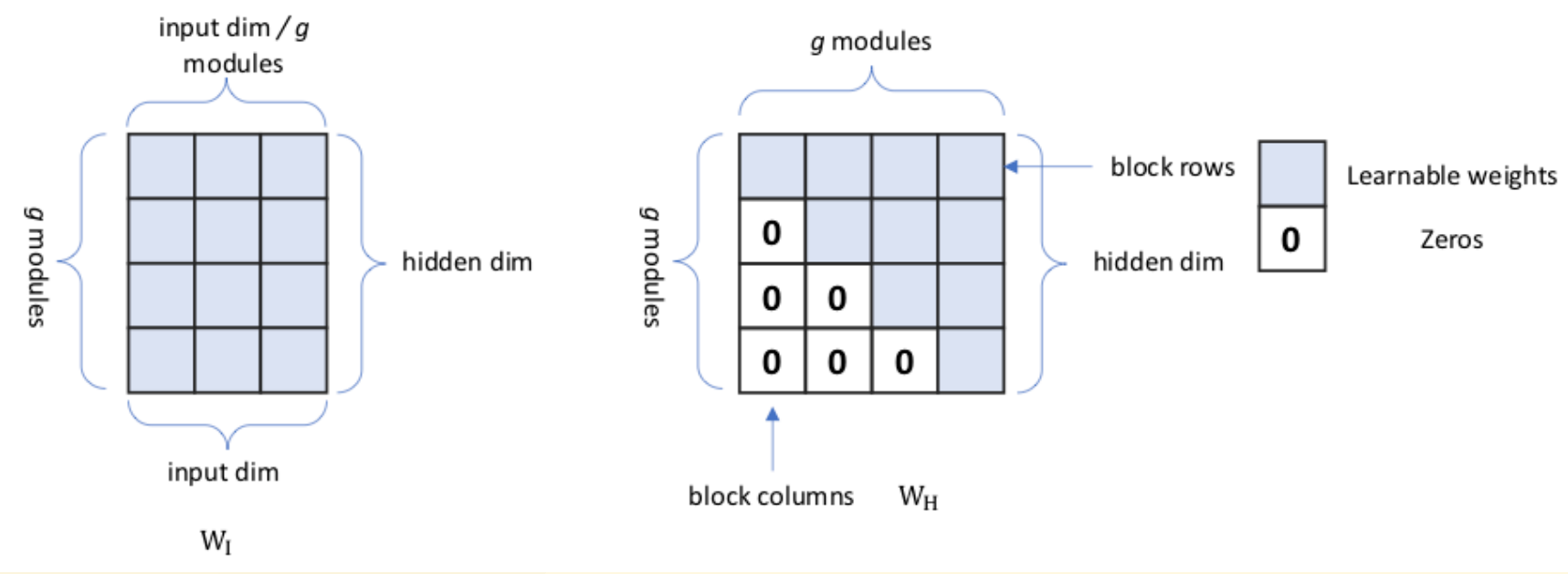
Figure 2: (left) An example of a three-layer DILATEDRNN with dilation 1, 2, and 4. (right) An example of a two-layer DILATEDRNN, with dilation 2 in the first layer. In such a case, extra embedding connections are required (red arrows) to compensate missing data dependencies.

Clockwork RNN (CW-RNN, 2014)

- Problem to address
    1. vanishing/gradient.
    2. learn patterns that span at different time scale.
    3. less computation but learning comparable performance as LSTM.

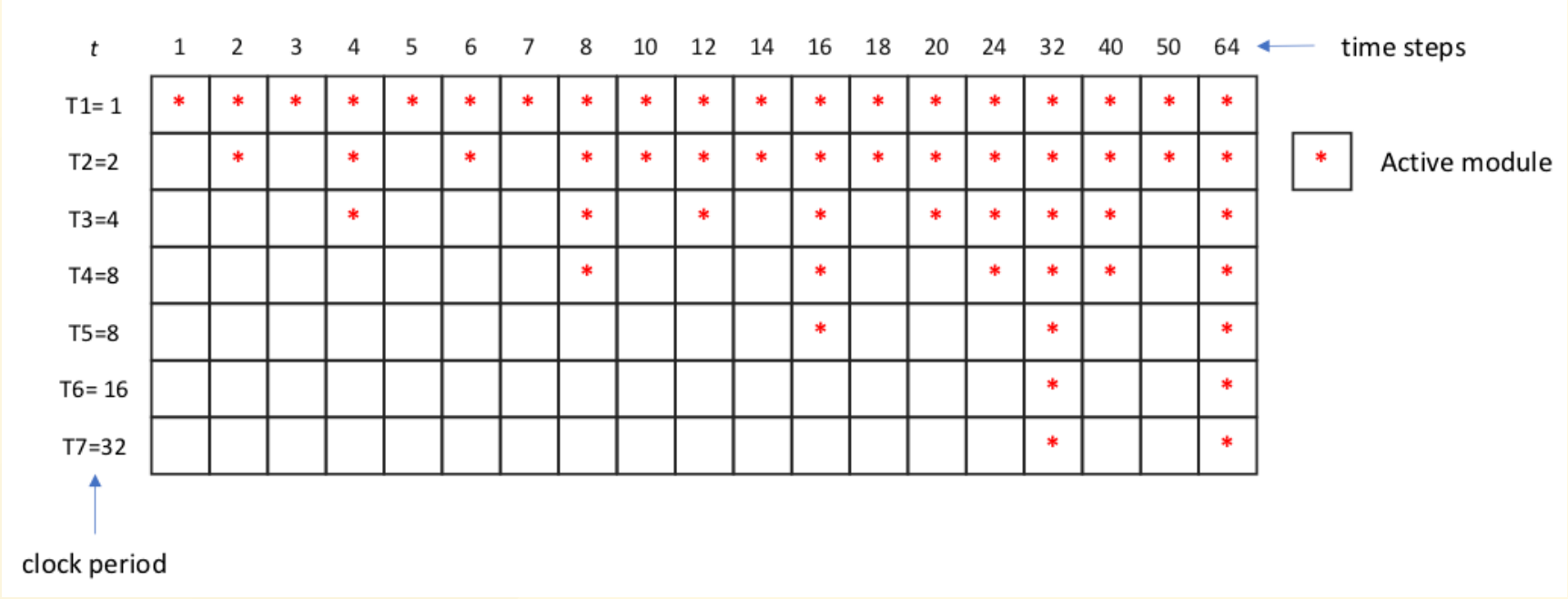
Clockwork RNN (I)

1. Partition hidden layer into sub-modules.



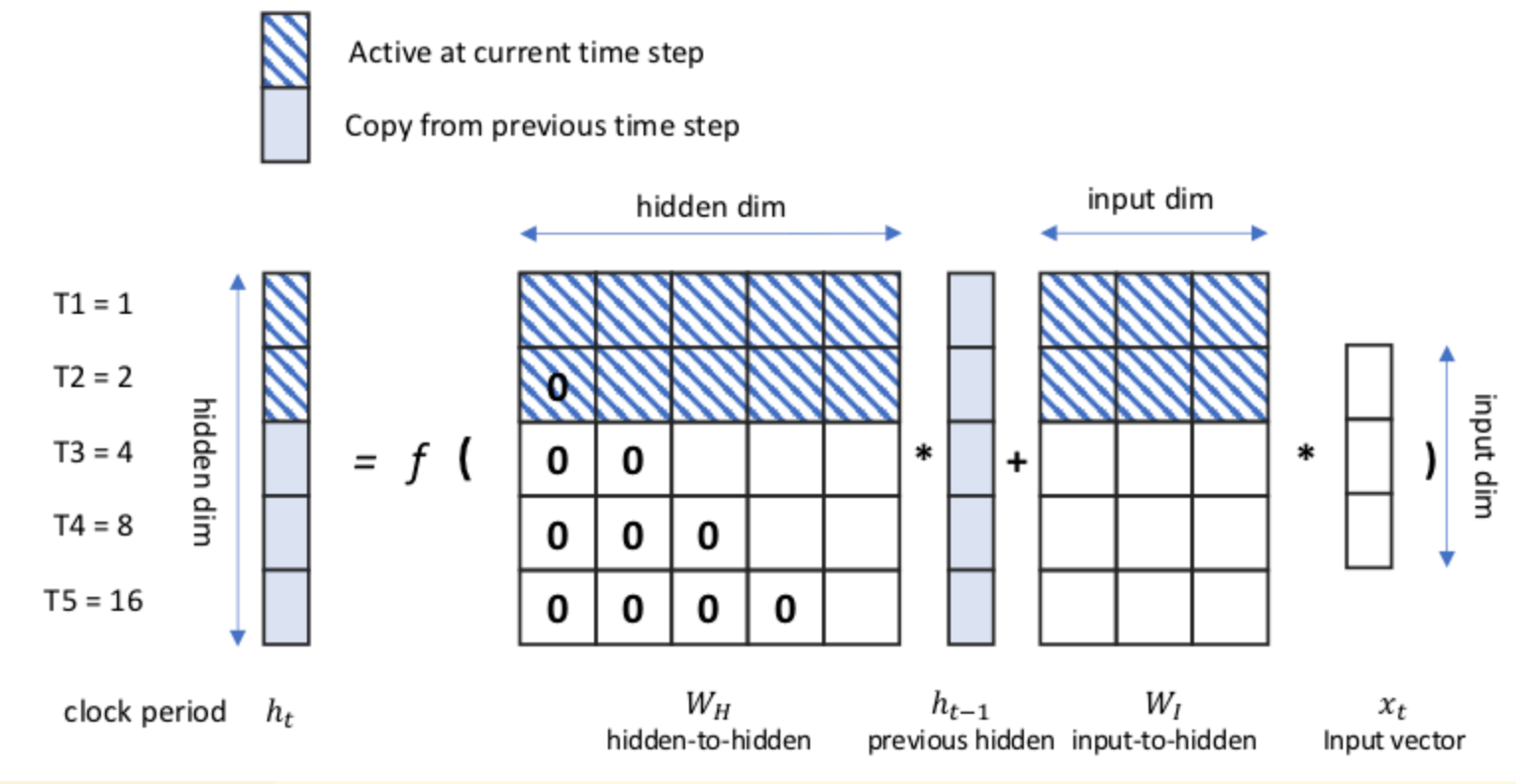
Clockwork RNN (II)

2. Assign different clock period to different modules.



Clockwork RNN (III)

3. For unactive modules, directly copy their values in previous time step to current time steps.



Downside of Clockwork RNN

How to choose clock period significantly affect clockwork RNN's learning performance and is non-trivial.

Multi-dimensional LSTM (MD-LSTM)

- Problem to solve: Applied RNN to multi-dimensional data rather than sequence.
- Models
  1. Replace the single recurrent connection found in standard RNNs with ***as many recurrent connections*** as there are dimensions in the data.
  2. During the forward pass, at each point in the data sequence, the hidden layer of the network receives both an external input and its activations from one step back along all dimensions.

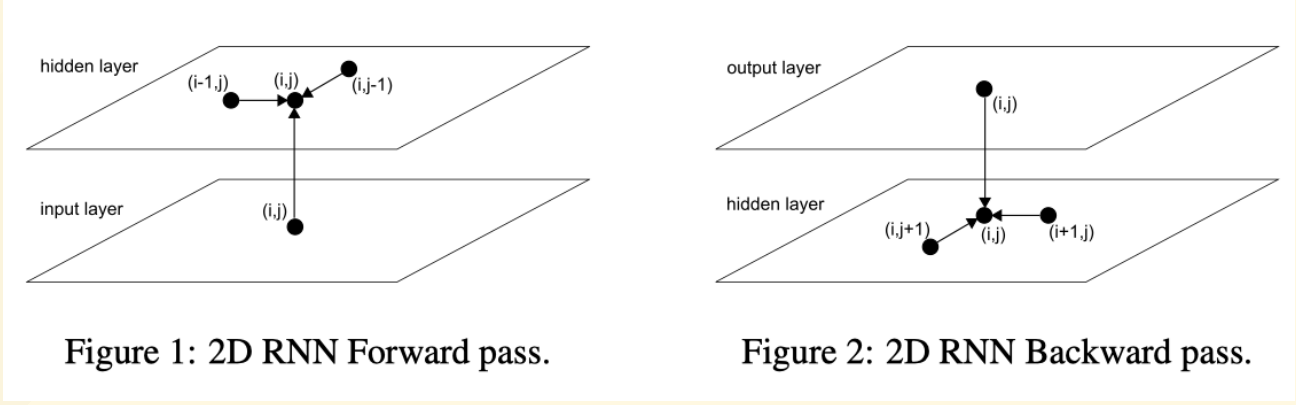


Figure 1: 2D RNN Forward pass.

Figure 2: 2D RNN Backward pass.

Fig1. Figure 1 and 2 from the paper.

Multi-dimensional LSTM (MD-LSTM)

3. MD-LSTM can also be equipped with directional information.

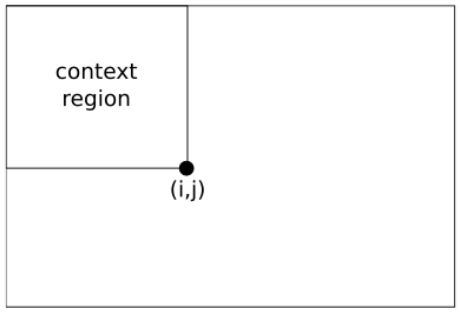


Figure 4: Context available at (i,j) to a 2D RNN with a single hidden layer.

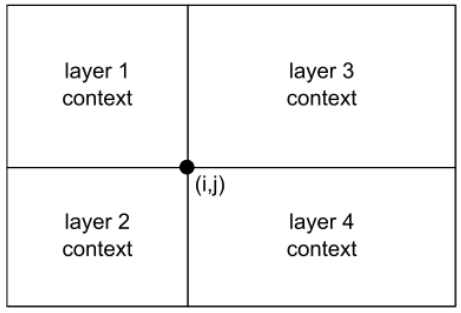
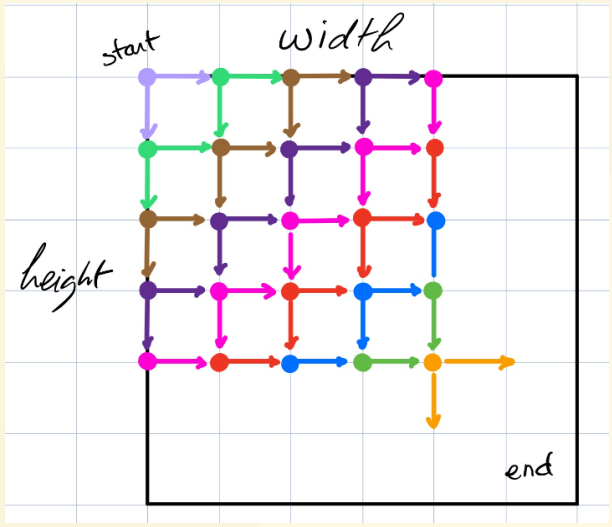


Figure 5: Context available at (i,j) to a multi-directional 2D RNN.

Fig2. Figure 4 and 5 from the paper.

MD-LSTM on 2D



LSTM on 2D

1. N: the dimension number
2. Input $\mathbf{x}$ that is arranged in an N-dimensional grid
    - such as 2-D grid of pixels in an image.
3. N hidden vectors $\mathbf{h}_1, \ldots, \mathbf{h}_N$
4. N memory vectors $\mathbf{c}_1, \ldots, \mathbf{c}_N$

$$\mathbf{c}_t = \sum_i^N \mathbf{f}_t^i \circ \mathbf{c}_t^i + \mathbf{i}_t \circ \hat{\mathbf{c}}_t$$

Grid LSTM

- Extend LSTM cell to deep networks within a unified architecture.
- Propose a novel robust way for modulating N-way communication across the LSTM cells.

GridLSTM

Inputs:

1. a N-dimensioanl block receives N hidden vectors: $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N$ and,
2. N memory vectors $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_N$

Compute:

1. deploys cells along ***any*** or ***all*** of the dimensions including the depth of the network;
2. ***concatenate*** all input hiddens to form $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]'$. ***This is the difference from HM-LSTM***.
3. compute N LSTM transforms: $(\mathbf{h}_i, \mathbf{m}_i) = \mathrm{LSTM}(\mathbf{H}, \mathbf{m}_i, \mathbf{W}_i)$ where $i = [1, \ldots, N]$, W cancatenates $\mathbf{W}_i^i$, $\mathbf{W}_f^i$, $\mathbf{W}_o^i$, $\mathbf{W}_c^i$ in $\mathbb{R}^{d \times Nd}$.

GridLSTM

Priority Dimensions

1. in general case, a N-dimensional block computes the transforms for all dimensions are ***in parallel.***
2. prioritize the dimension of the network. For dimensions other than prioritized dimensions, their output hidden vectors are computed first, and finally, the prioritized.
    - for example, to prioritize the first dimension of the network, the block first computes the $N - 1$ transforms for the other dimensions obtaining the output hidden vectors $\mathbf{h}'_2, \ldots, \mathbf{h}_N$.

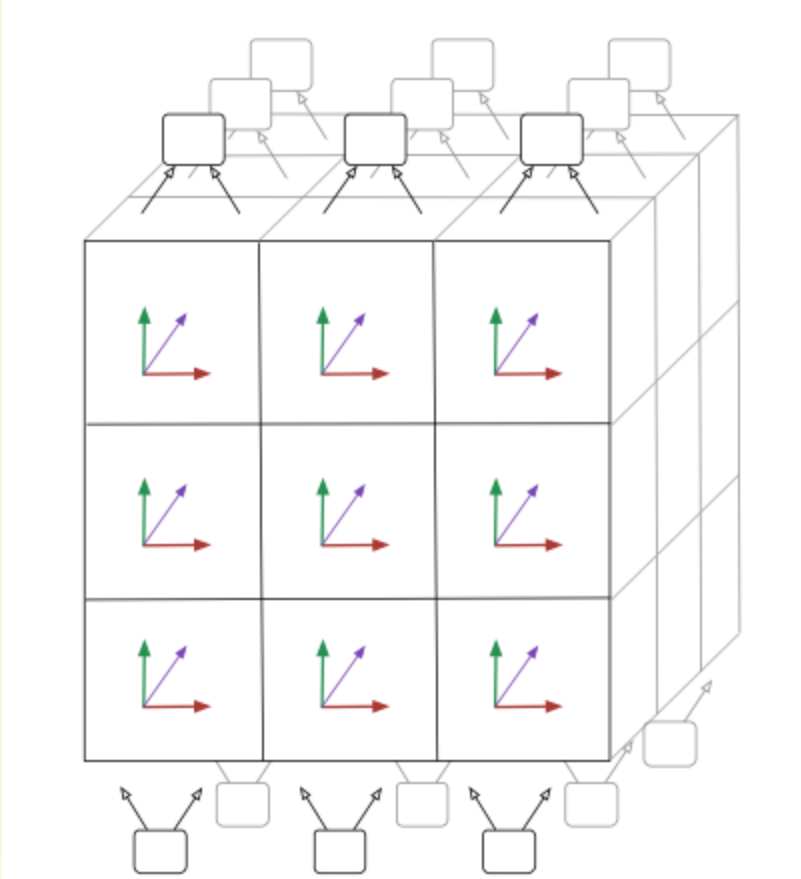
Non-LSTM dimensions

Along some dimension, regular connection instead of LSTM is used.

$$\mathbf{h}'_1 = \alpha(\mathbf{V} * \mathbf{H})$$

α above is a standard nonlinear transfer function or identity mapping.

An example: a 3D GridLSTM



3d Grid LSTM

3D GridLSTM for NMT

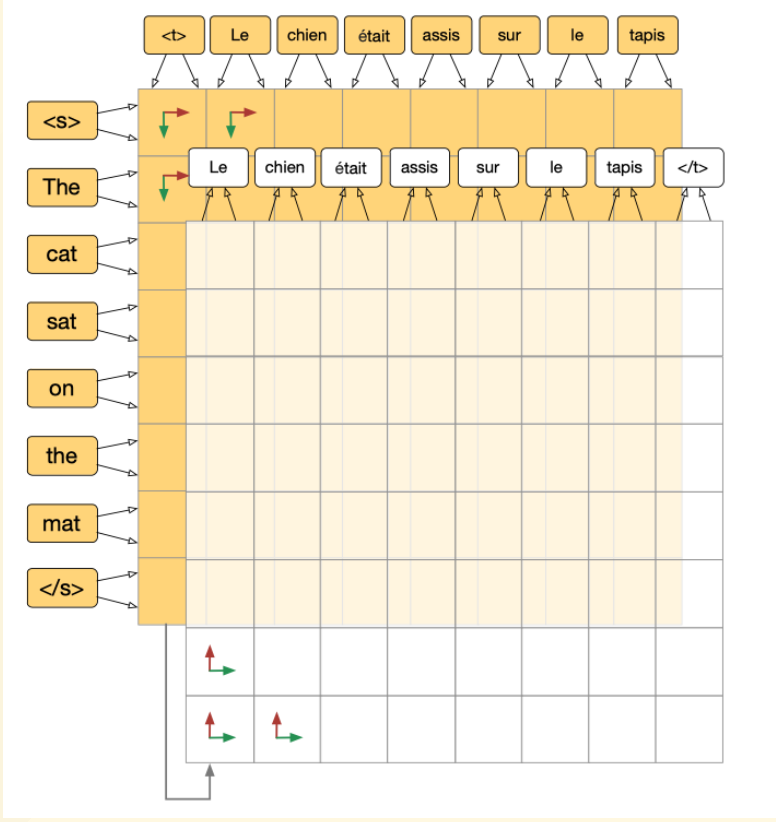


Fig. GridLSTM for NMT.

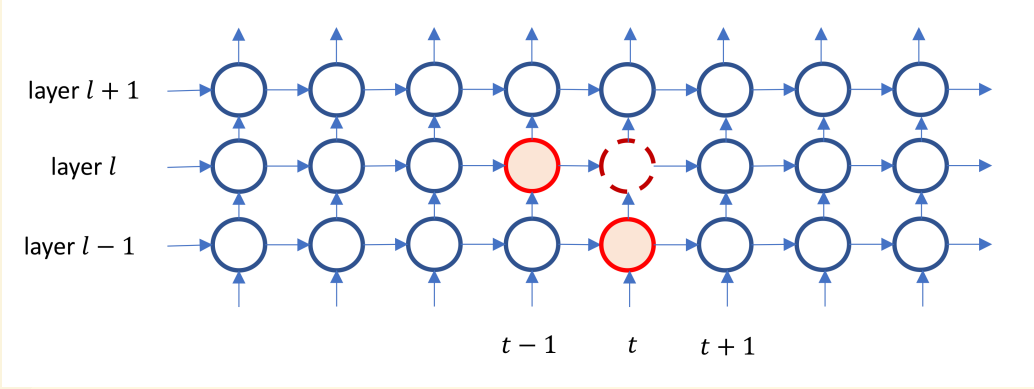
Hierarchical Multiscale Recurrent Neural Networks (HM-RNN)

- Problem to address:
  Learn the hierarchical multiscale structure from temporal data ***without explicit boundary information***.

HM-RNN (I)

Key points of the model

1. Introduce ***a parametrized binary boundary detector*** z_t^i at **each layer**.
    - turned on only at the time steps where a segment of the corresponding abstraction level is completely processed.
    - boundary state is a discrete variable.
    - ***straight-through estimator*** is used to calculate gradient of the boundary detector.
2. One of three operations: **UPDATE**, **COPY**, and **FLUSH** is choosen according to z_t^{l-1} and z_{t-1}^l



HM-RNN: boundary detector and action selection

z_{t-1}^{l} **left**	z_{t}^{l-1} **buttom**	**The Selected Operation**	
0	0	**COPY**	buttom and left states both do not reach to a boundary.
0	1	**UPDATE**	left state does not reaches to a boundary but buttom does. **UPDATE** is executed sparsely
1	0	**FLUSH**	left state reaches to a boundary.
1	1	**FLUSH**	left state reaches to a boundary.

- **UPDATE**: similar to update rule of the LSTM, but executed ***SPARSELY***.
- **COPY**: ***simply copies***. Upper layer keeps its state unchanged until it receives the summarized input from the lower layer.
- **FLUSH**: executed when a boundary is detected
  - it first ejects the summarized representation of the current segment to the upper layer
  - then reinitializes the states to start processing the next segment.

HM-LSTM: equations

1. pre-activation

$$\begin{pmatrix} \mathbf{f}_t^\ell \\ \mathbf{i}_t^\ell \\ \mathbf{o}_t^\ell \\ \mathbf{g}_t^\ell \\ \tilde{z}_t^\ell \end{pmatrix} = \begin{pmatrix} \texttt{sigm} \\ \texttt{sigm} \\ \texttt{sigm} \\ \texttt{tanh} \\ \texttt{hard sigm} \end{pmatrix} f_{\texttt{slice}} \left(\mathbf{s}_t^{\text{recurrent}(\ell)} + \mathbf{s}_t^{\text{top-down}(\ell)} + \mathbf{s}_t^{\text{bottom-up}(\ell)} + \mathbf{b}^{(\ell)} \right),$$

where

$$\begin{aligned} \mathbf{s}_t^{\text{recurrent}(\ell)} &= U_\ell^\ell \mathbf{h}_{t-1}^\ell, \\ \mathbf{s}_t^{\text{top-down}(\ell)} &= z_{t-1}^\ell U_{\ell+1}^\ell \mathbf{h}_{t-1}^{\ell+1}, \\ \mathbf{s}_t^{\text{bottom-up}(\ell)} &= z_t^{\ell-1} W_{\ell-1}^\ell \mathbf{h}_t^{\ell-1}. \end{aligned}$$

2. cell update

$$\mathbf{c}_t^\ell = \begin{cases} \mathbf{f}_t^\ell \odot \mathbf{c}_{t-1}^\ell + \mathbf{i}_t^\ell \odot \mathbf{g}_t^\ell & \text{if } z_{t-1}^\ell = 0 \text{ and } z_t^{\ell-1} = 1 \text{ (UPDATE)} \\ \mathbf{c}_{t-1}^\ell & \text{if } z_{t-1}^\ell = 0 \text{ and } z_t^{\ell-1} = 0 \text{ (COPY)} \\ \mathbf{i}_t^\ell \odot \mathbf{g}_t^\ell & \text{if } z_{t-1}^\ell = 1 \text{ (FLUSH)}, \end{cases}$$

3. output hidden

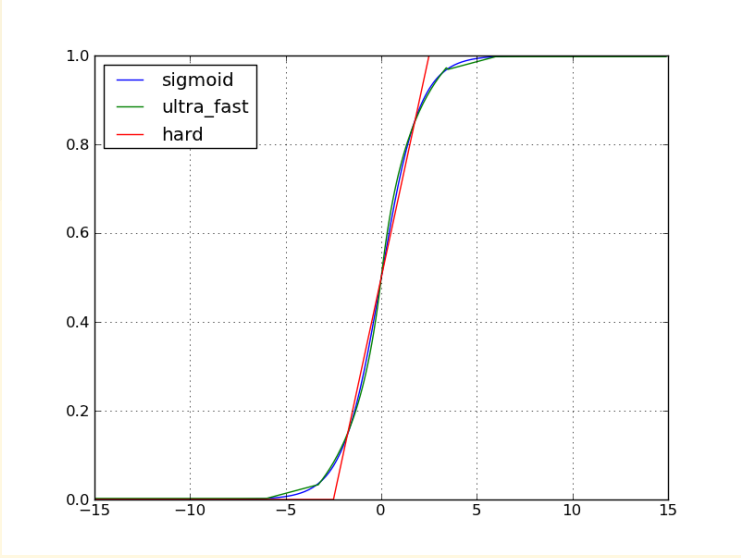
$$\mathbf{h}_t^\ell = \begin{cases} \mathbf{h}_{t-1}^\ell & \text{if COPY,} \\ \mathbf{o}_t^\ell \odot \texttt{tanh}(\mathbf{c}_t^\ell) & \text{otherwise.} \end{cases}$$

STE (Straight-through estimator)

1. Straight-through estimator.
    - forward pass uses the step function to activate z_t^l
    - backward pass uses [hardsigmoid] function as the biased estimator of the outgoing gradient.

$$\sigma(x) = \max(0, \min(1, (\alpha x + 1)/2))$$

2. Slope annealing.
    - start from slope $\alpha = 1$.
    - slowly increase the slope until it reaches a threshold. In the paper, the annealing function task-specific.



ON-LSTM (ICLR 2019 the best paper award)

The problem to address in this paper

1. natural language is hierarchically structured.
2. design a better architecture ***equipped with an inductive bias towards learning such latent tree structures***.

ON-LSTM: Key ideas

Neurons are split into high-ranking and low-ranking neurons.

low-ranking neurons	high-ranking neurons
contains long-term or global information that lasts for several time steps to the entire sentence	encodes local information that lasts only one or a few time steps.

The differentiation between high-ranking and low-ranking neurons is learned in a completely data-driven fashion by controlling the update frequency of single neurons.

1. to erase (or update) high-ranking neurons, the model ***should first erase (or update) all lower-ranking neurons***.
2. low-ranking neurons are ***always*** updated more frequently than high-ranking neurons others, and order is pre-determined as part of the model architecture.

Recap LSTM

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\hat{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

Modificatioins made to standard LSTM

1. make input and forget gate for each neuron dependent on others
2. replace the update function for the cell state $\mathbf{c}_t$.

ON-LSTM: cumax as gate activation

The `cumax` activation function: $\hat{g}$ which is the cumulative sum of softmax.

$$\hat{\mathbf{g}} = \text{cumsum}(\text{softmax}(\ldots))$$

Note for `cumax`

1. $\mathbf{g} = [0, \ldots, 0, 1, \ldots, 1]$ is a binary gate that splits the cell into two segments: the 0-segment and the 1-segment. As a result, the model can apply different update rules on the two segments.
2. $\hat{\mathbf{g}}$ is the expectation of the binary gate $\hat{\mathbf{g}} = \mathbb{E}[g]$.
    - ideally, $\mathbf{g}$ should take the form of a ***discrete*** variable, but computing gradients when a discrete variable is included in the computation graph is not trivial.
    - $\hat{\mathbf{g}}$ here is a continuous relaxation.

ON-LSTM: structured gating

Master forget and input gate

$$\tilde{f}_t = \text{cumax}(W_{\tilde{f}} x_t + U_{\tilde{f}} h_{t-1} + b_{\tilde{f}})$$

$$\tilde{i}_t = 1 - \text{cumax}(W_{\tilde{i}} x_t + U_{\tilde{i}} h_{t-1} + b_{\tilde{i}})$$

$\tilde{f}_t$	$\tilde{i}_t$
controls the erasing behavior	controls the writing behavior
monotonically increasing	monotonically decreasing

NOTES for master gates

1. master gates only focus on coarse-grained control.
2. it is computationally expensive and unnecessary to model them with the same dimensions as the hidden states.
3. the paper sets $\tilde{f}_t$ and $\tilde{t}_t$ to bo $D_m = \frac{D}{C}$ where C is the chunk size factor.
4. each dimension of the master gates are repeated C times before the element-wise multiplication with LSTM's original forget gates f_t and input gates i_t.

ON-LSTM: cell update

update rules for c_t based on master gates

$$\omega_t = \tilde{f}_t \circ \tilde{i}_t$$

$$\hat{f}_t = f_t \circ \omega_t + (\tilde{f}_t - \omega_t) = \tilde{f}_t \circ (f_t \circ \tilde{i}_t + 1 - \tilde{i}_t)$$

$$\hat{i}_t = i_t \circ \omega_t + (\tilde{i}_t - \omega_t) = \tilde{i}_t \circ (i_t \circ \tilde{f}_t + 1 - \tilde{f}_t)$$

$$c_t = \hat{f}_t \circ c_{t-1} + \hat{i}_t \circ \hat{c}_t$$

ω_t represents the overlap of $\tilde{f}_t$ and $\tilde{i}_t$

Summary

1. RNNs can model data produced by context-free grammars and context-sensitive grammars.
2. Gate is necessary to modulate gradient flow.
3. In practice, RNN models are all stacked to form deep RNN network. The depth seldom accesses 10 layers.

RNN Variants	Key Lessons
Dilated LSTM	skip connection
CW-RNN	split hidden into modules, and modules are updated sparsely
MM RNN/Grid LSTM	Recurrent computation to high order tensor
Hierarchical Multiscale Recurrent Neural Networks	sparsely updated cell
ON-LSTM	split cell into low and high rank parts whose updating frequency are differentiated