UNIVERSITY OF CALGARY

Recurrent Neural Networks

Starting Recurrent Neural Networks and LSTMs

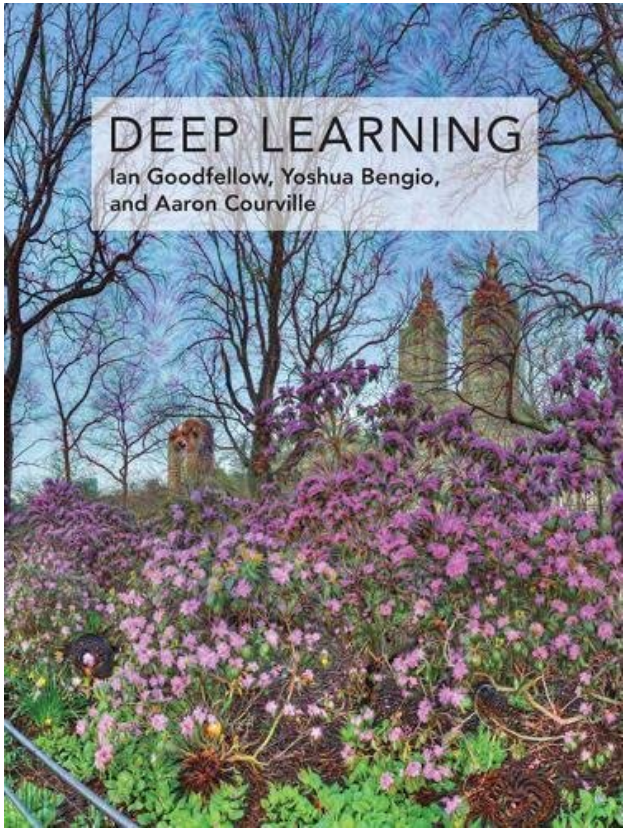
Banafshe Felfeliyan

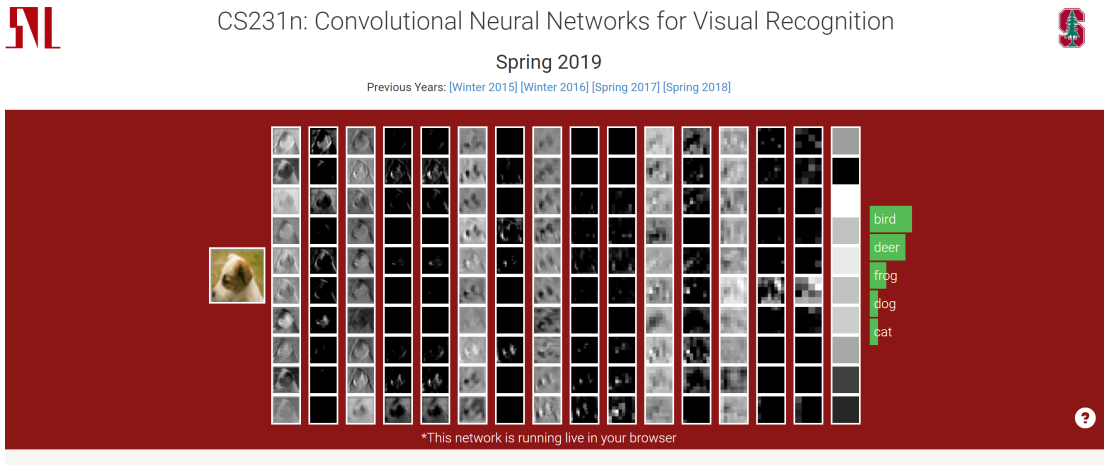
Biomedical Engineering, University of Calgary, Calgary, Canada
McCaig institute for bone & joint health, University of Calgary, Calgary, Canada

June, 2019

Resources

Slides are adopted and inspired from :



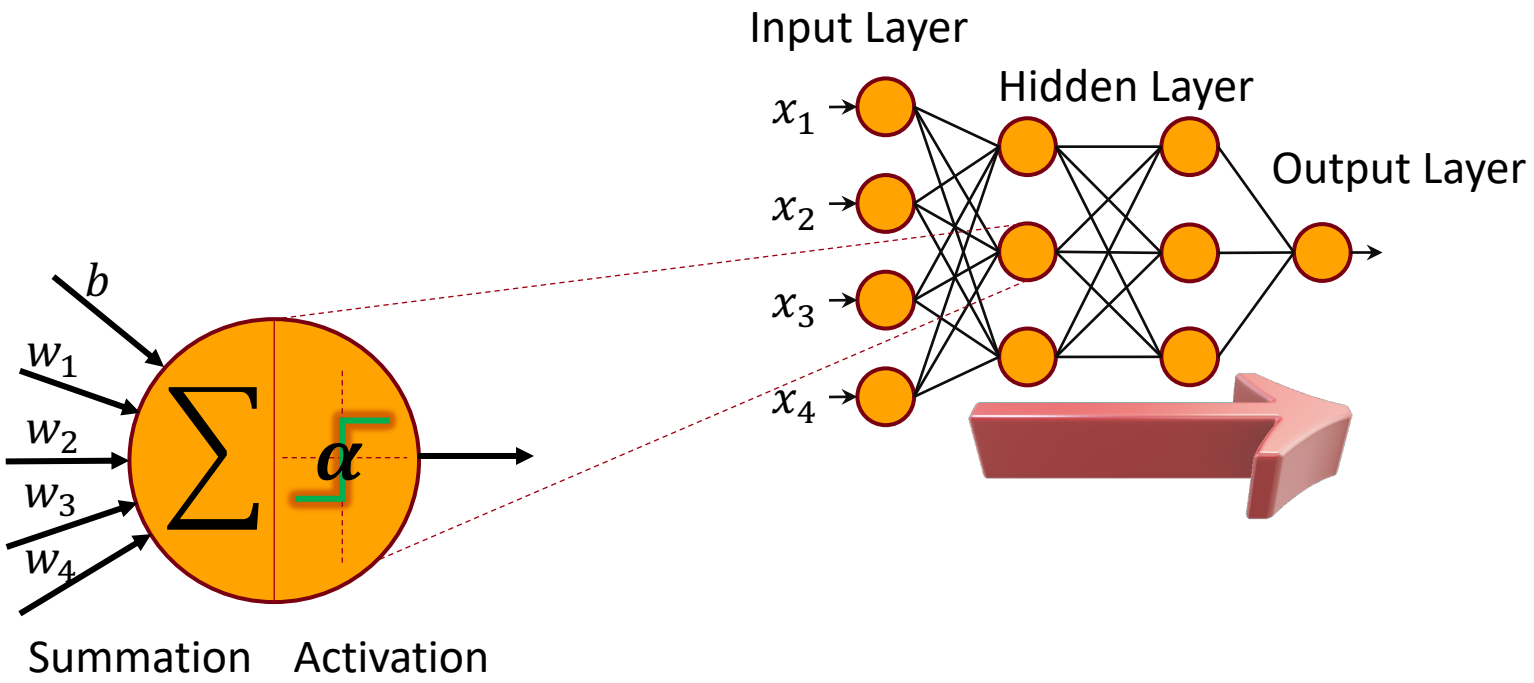


Content

UNIVERSITY OF CALGARY

- Feedforward Networks
- Recurrent Networks
- Backpropagation Through Time
- Vanishing and Exploding Gradients
- Long Short-Term Memory Units (LSTMs)
- …

Feedforward Networks!!

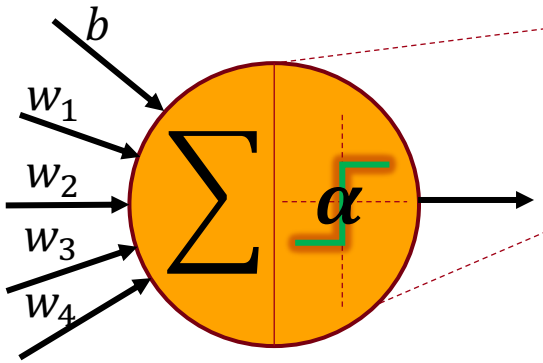


$$S = \sum w_i . x_i \qquad f(S)$$

Unit step
Sigmoid
tanh
Relu

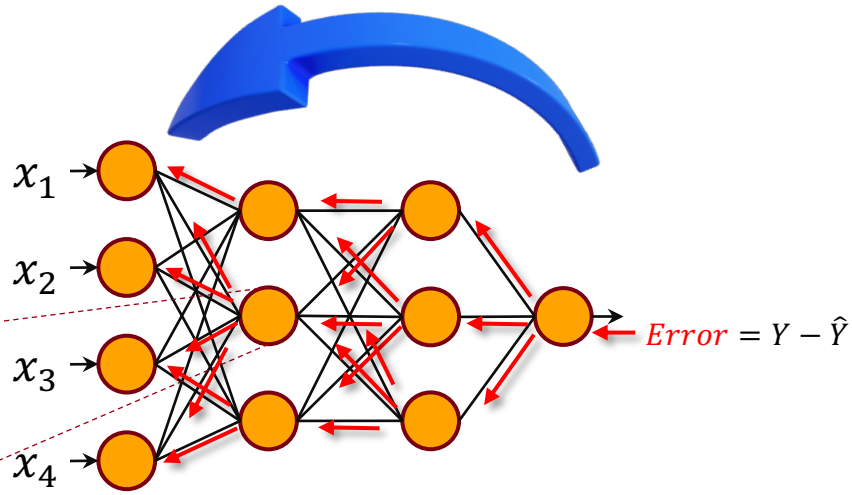
Back-propagation

- We need to minimize the error
- We need to compute partial derivative of each weight
- Chain rule



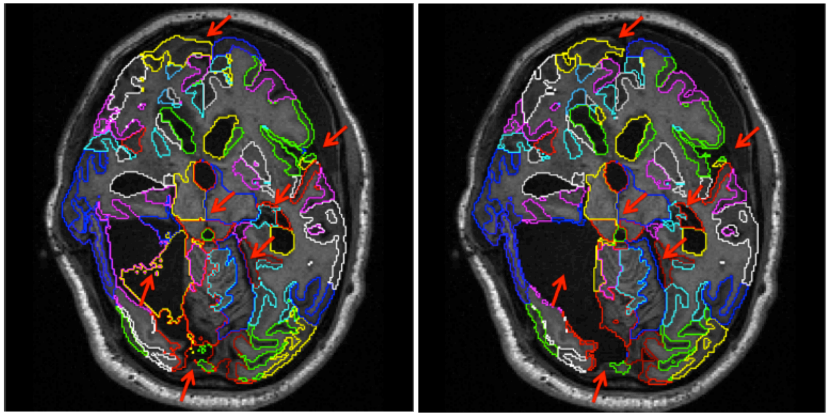
Summation Activation

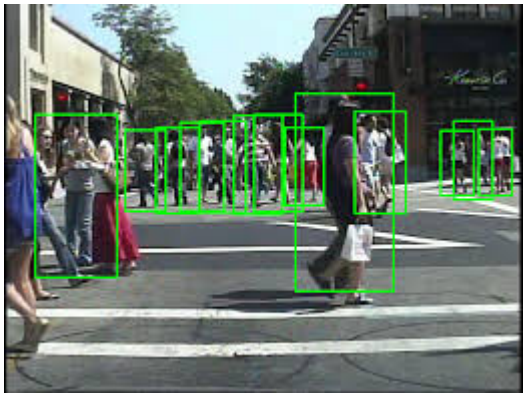
$$S = \sum w_i . x_i \qquad f(S) \begin{cases} Unit\ step \\ Sigmoid \\ \tanh \\ Relu \end{cases}$$

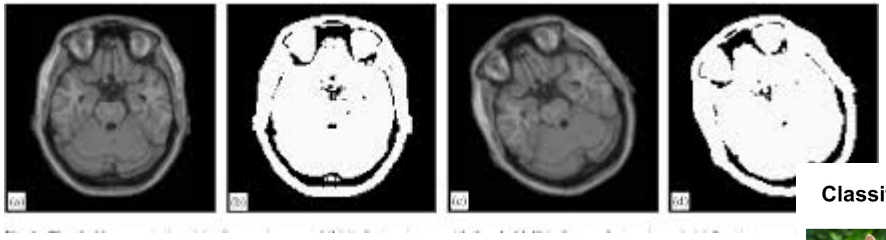


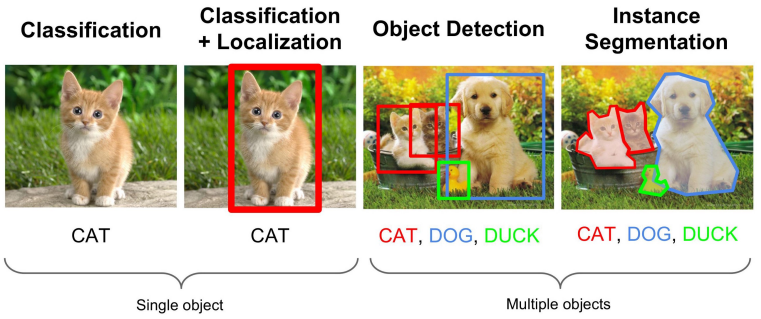
- Back-propagate error signals
- Calculate parameter gradients
- Update weights

Applications of Feedforward networks









Sometimes we need to take into account information from time step $t-1$ to make decision at time step t

Processing Sequence Data

UNIVERSITY OF CALGARY

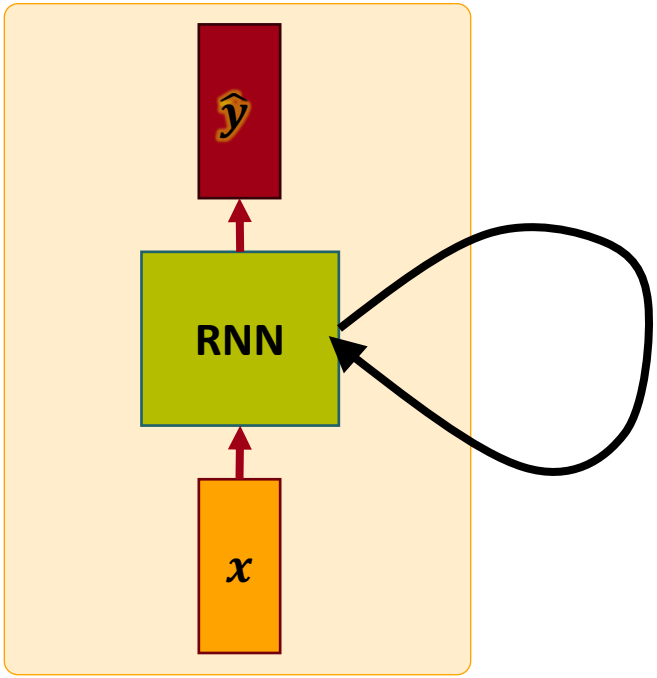
- Electrocardiography
- Ultrasound
- Angiography
- Fluoroscopy
- Longitudinal studies
- . . .

RNN

$$h_t = f_w(\, h_{t-1}, x_t)$$

$$h_t = tanh(\, W_{hh} h_{t-1} + W_{xh} x_t)$$

- f_w and relative parameters like are the same

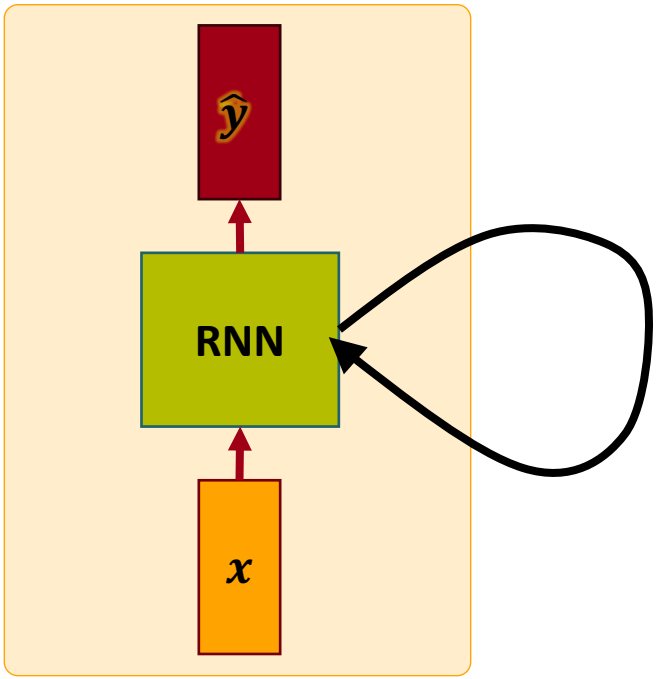


RNN

$$h_t = f_w(h_{t-1}, x_t)$$

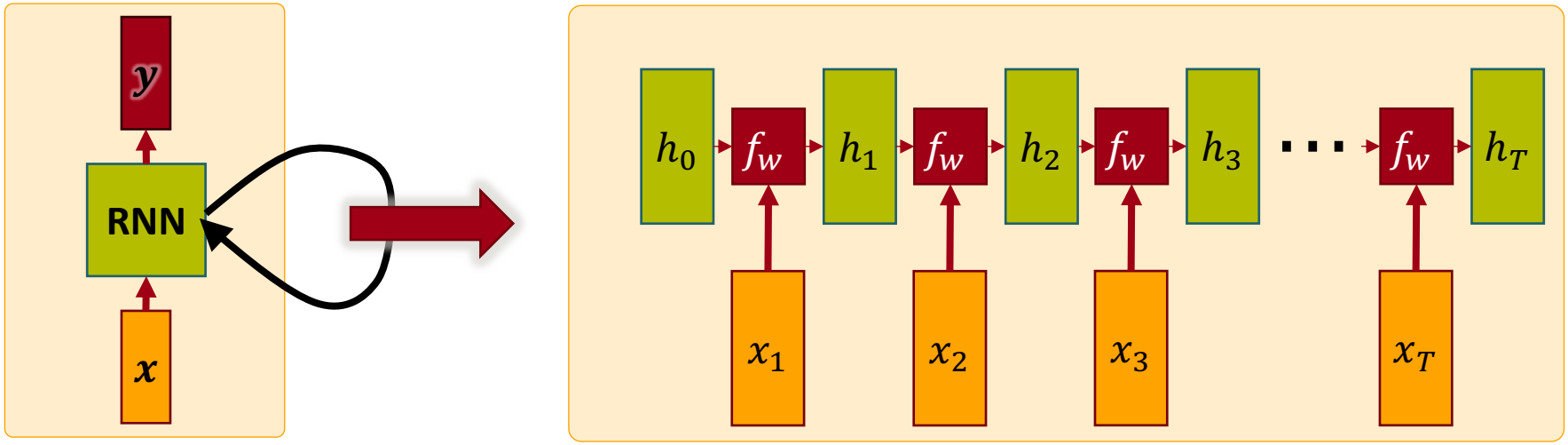
$$h_3 = f_w(h_2, x_3)$$

$$h_3 = f_w(f_w(h_1, x_2), x_3)$$

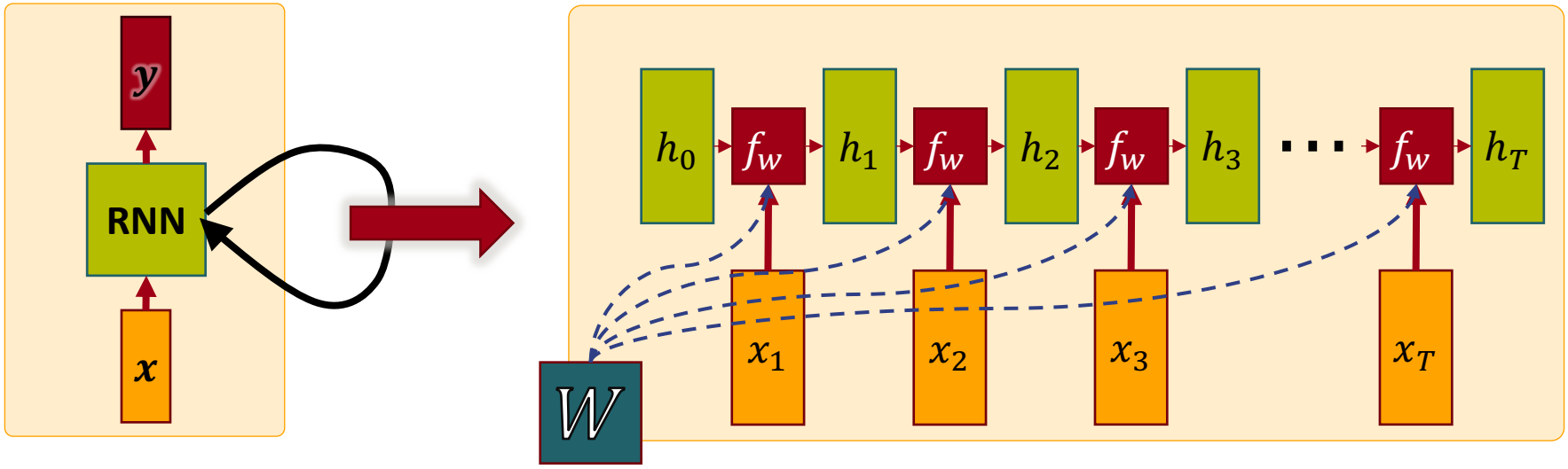


- f_w and relative parameters like are the same

RNN Unfolded Computational Graph



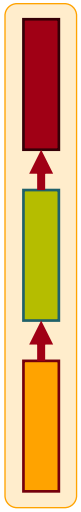
RNN Unfolded Computational Graph



Recurrent Networks

UNIVERSITY OF CALGARY

one to one



one to many

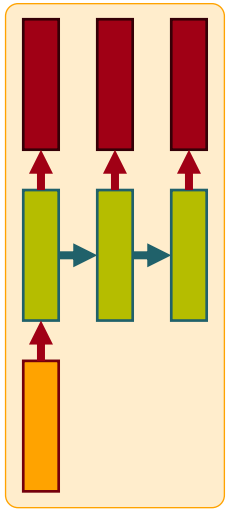
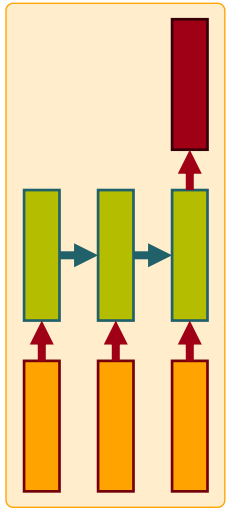


Image Captioning

"girl in pink dress is jumping in air."

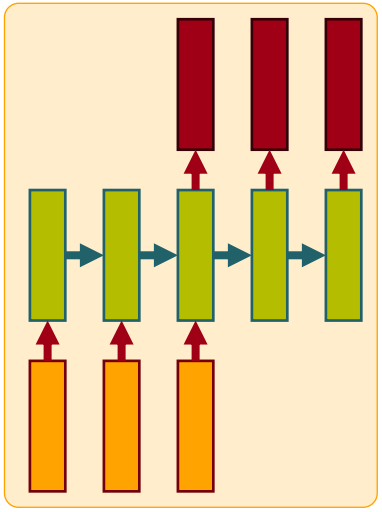
many to one



Sentiment Classification

I love Avengers!!!

many to many

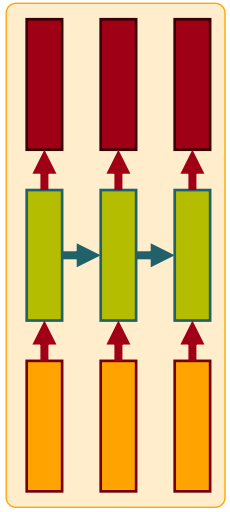


Machine Translation

چشمها را باید شست
جور دیگر باید دید...

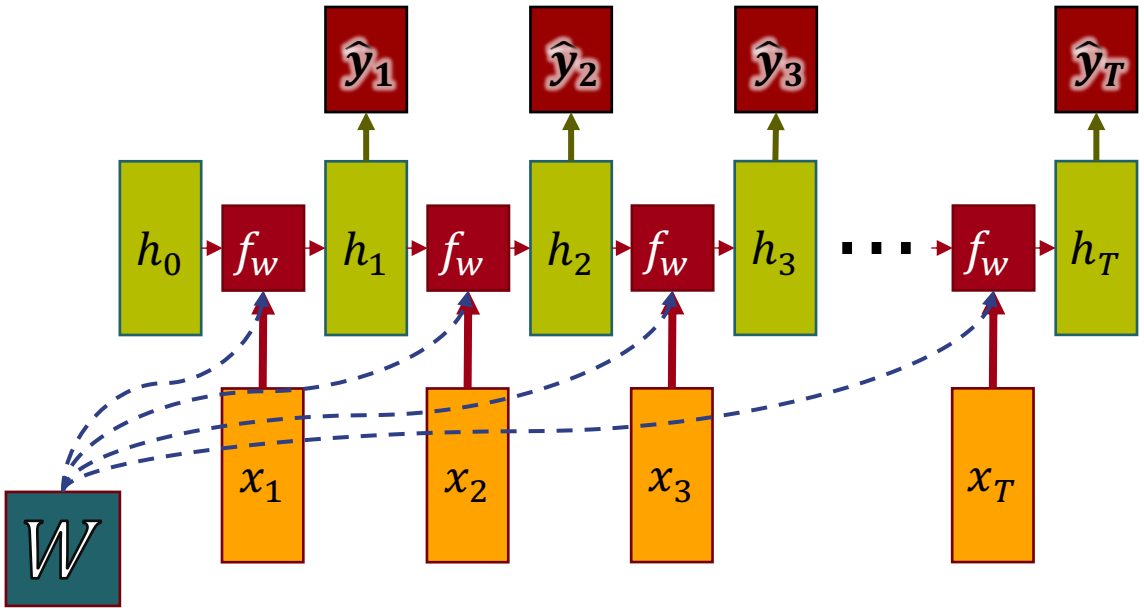
We need to rinse our eyes, and view things differently...

many to many

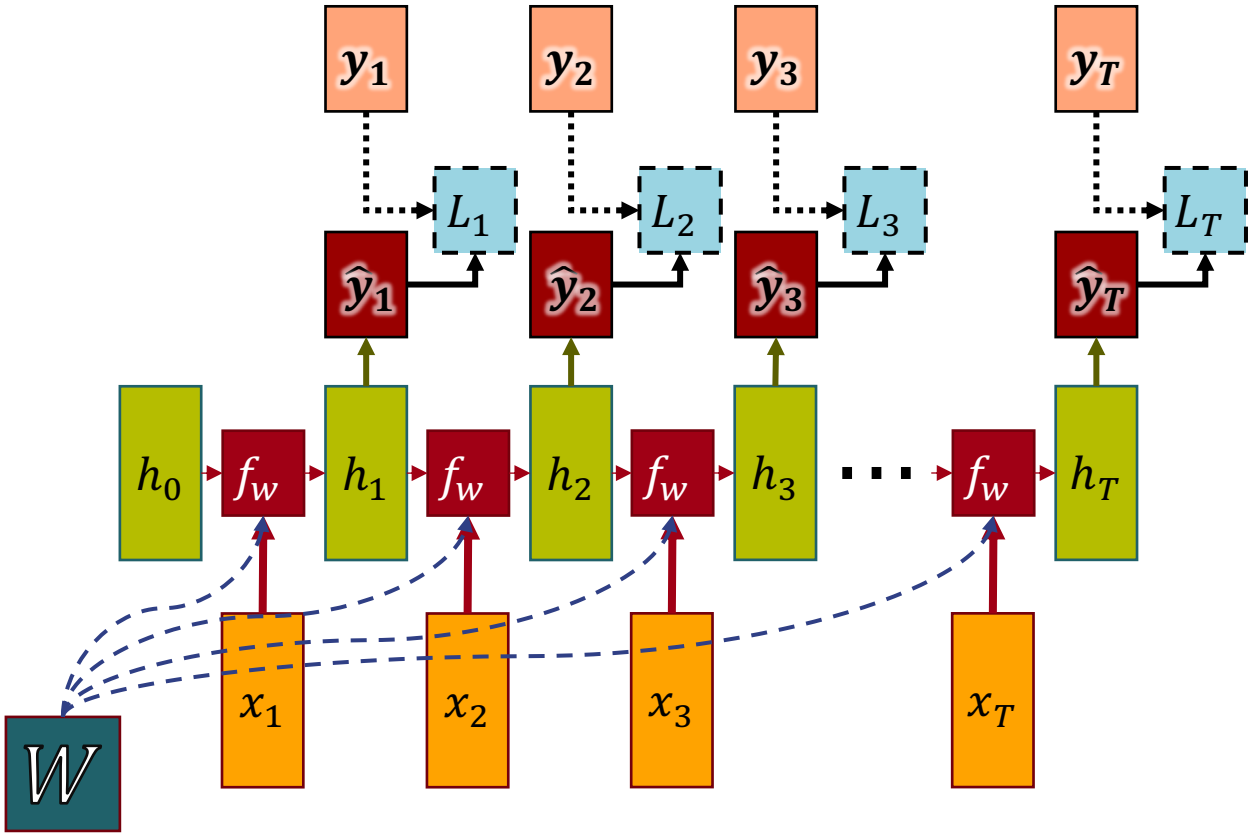


Video classification on frame level

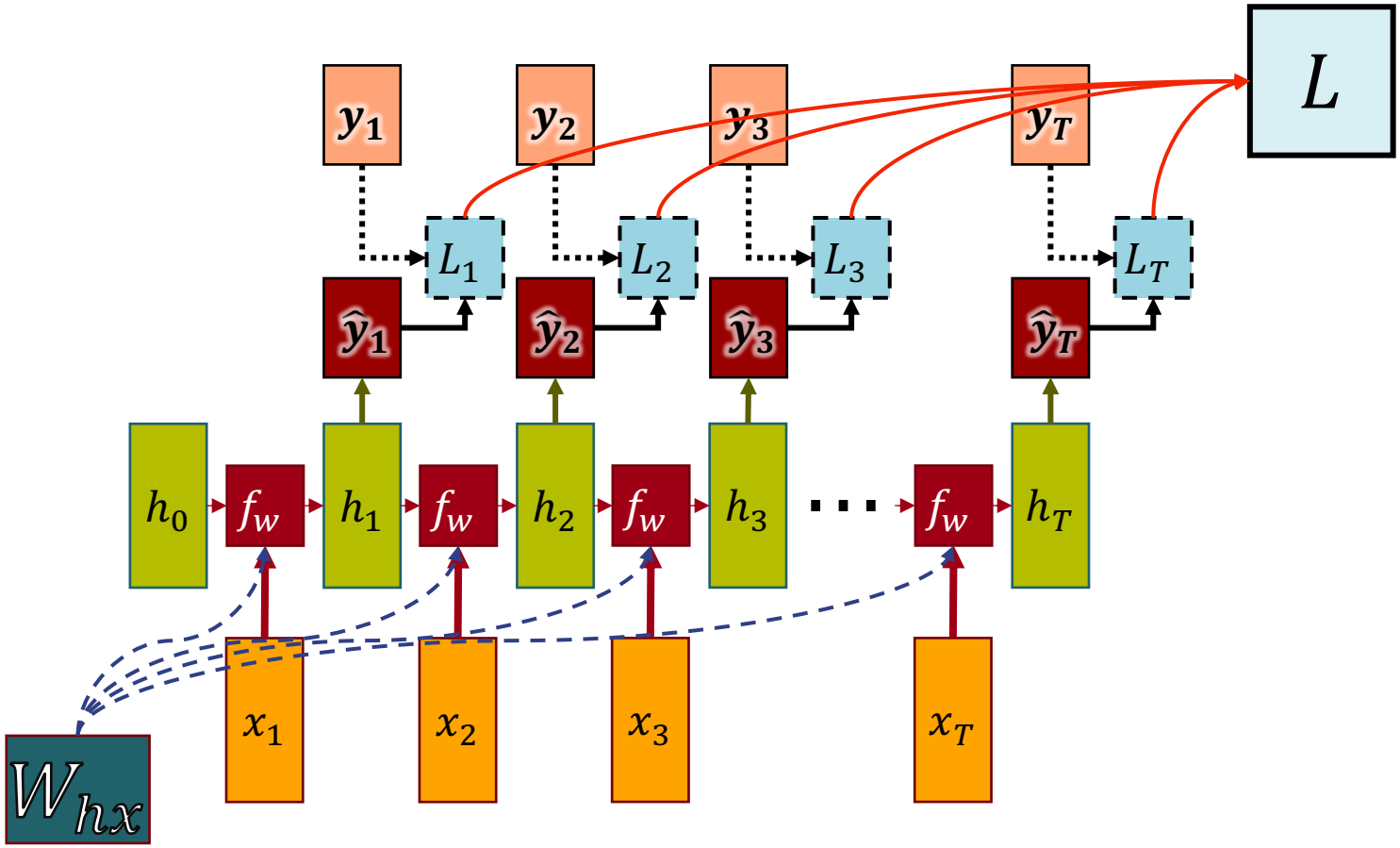
Many to many



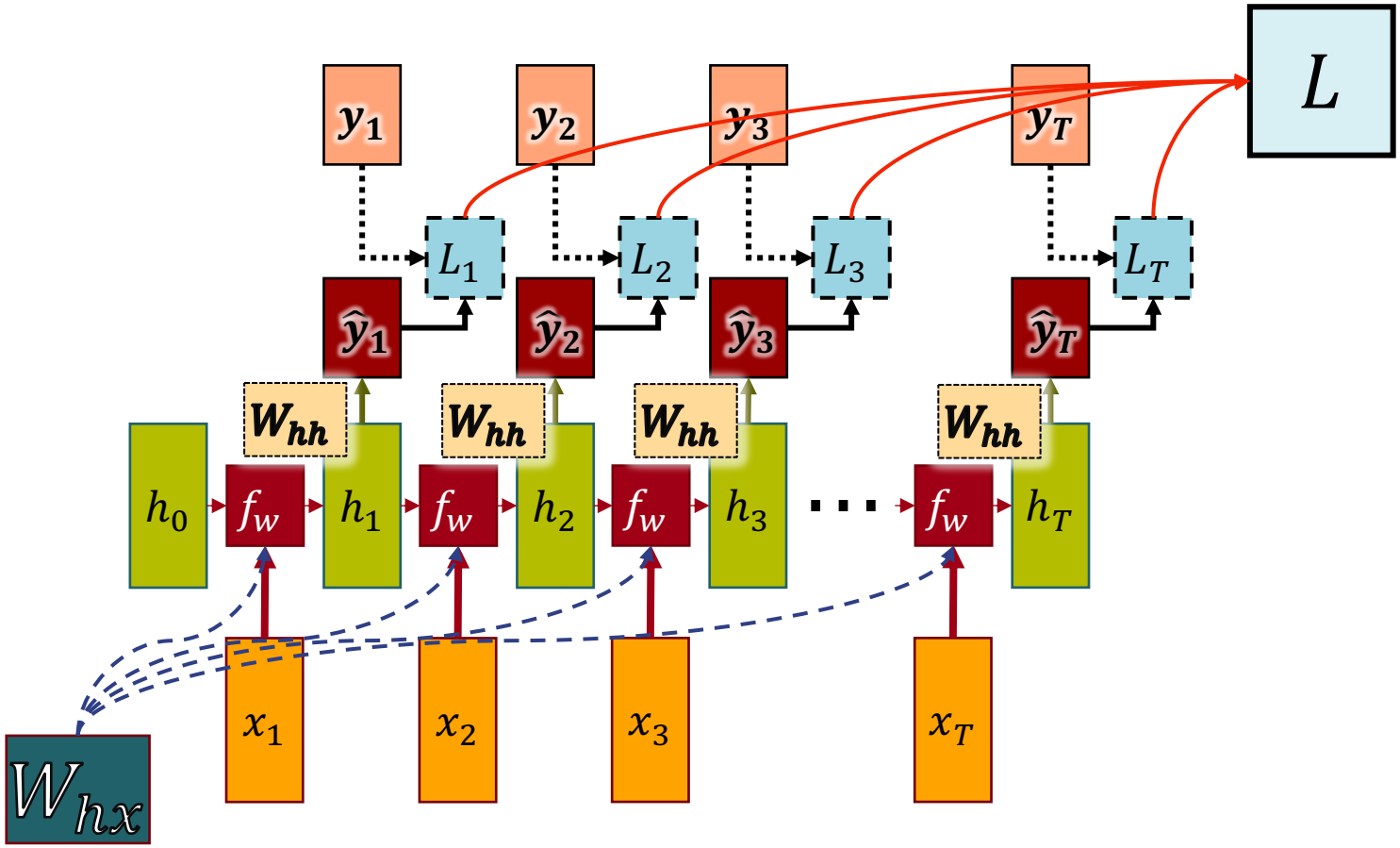
Many to many


y_1
y_2
y_3
y_T
L_1
L_2
L_3
L_T
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\hat{y}_T$
h_0
f_w
h_1
f_w
h_2
f_w
h_3
f_w
h_T
x_1
x_2
x_3
x_T
W

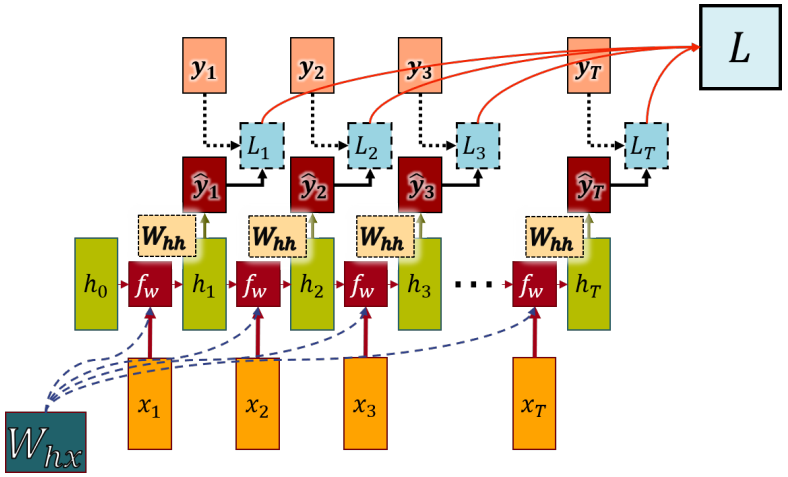
Many to many


L
y_1
y_2
y_3
y_T
L_1
L_2
L_3
L_T
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\hat{y}_T$
h_0
f_w
h_1
f_w
h_2
f_w
h_3
f_w
h_T
x_1
x_2
x_3
x_T
W_{hx}

Many to many


L
y_1
y_2
y_3
y_T
L_1
L_2
L_3
L_T
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\hat{y}_T$
W_{hh}
W_{hh}
W_{hh}
W_{hh}
h_0
f_w
h_1
f_w
h_2
f_w
h_3
f_w
h_T
x_1
x_2
x_3
x_T
W_{hx}

Many to many



- $a_t = b + W_{hh}h_{t-1} + W_{xh}x_t$
- $h_t = tanh(a_t)$
- $o_{ht} = c + W_{hy}h_t$
- $\hat{y}_t = softmax\ (o_t)$
- $L = \sum_t L_t$

Three Blocks and Three W

$$\begin{cases} input\ to\ hidden\ state:\ W_{xh} \\ previous\ hidden\ state\ to\ next\ hidden\ state: W_{hh} \\ hidden\ state\ to\ output: W_{hy} \end{cases}$$

The **back-propagation** algorithm applied to the **unrolled graph with $O(\tau)$** cost is called

Back-propagation through time

Not a specialized algorithm

Gradients obtained by back-propagation might use

UNIVERSITY OF CALGARY

$$L = \sum_{t} L_t$$

Computing the gradient of this **loss function** with respect to the parameters is an **expensive operation**

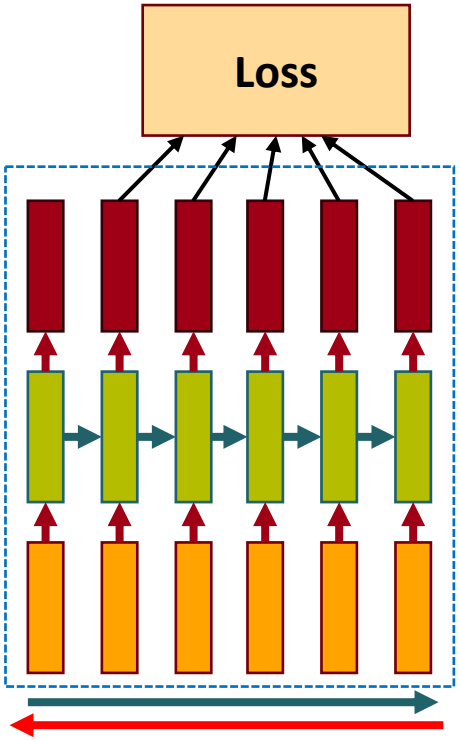
Run time $O(\tau)$

Memory cost $O(\tau)$

Truncated Backpropagation Through Time

UNIVERSITY OF CALGARY

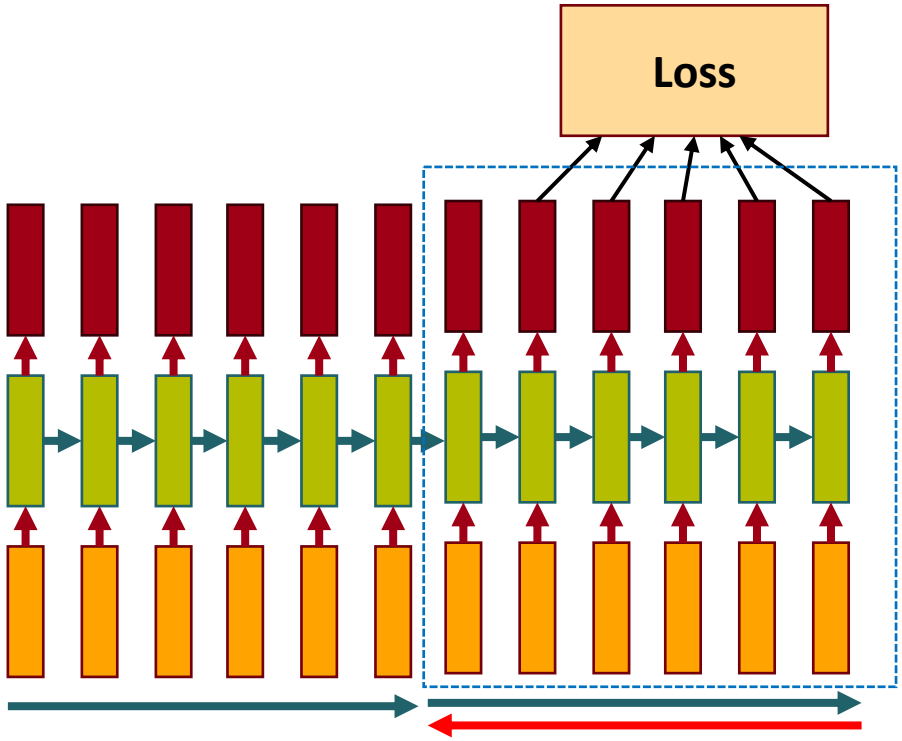
- Run forward and backward through chunks of sequence and not through whole sequence



Truncated Backpropagation Through Time

UNIVERSITY OF CALGARY

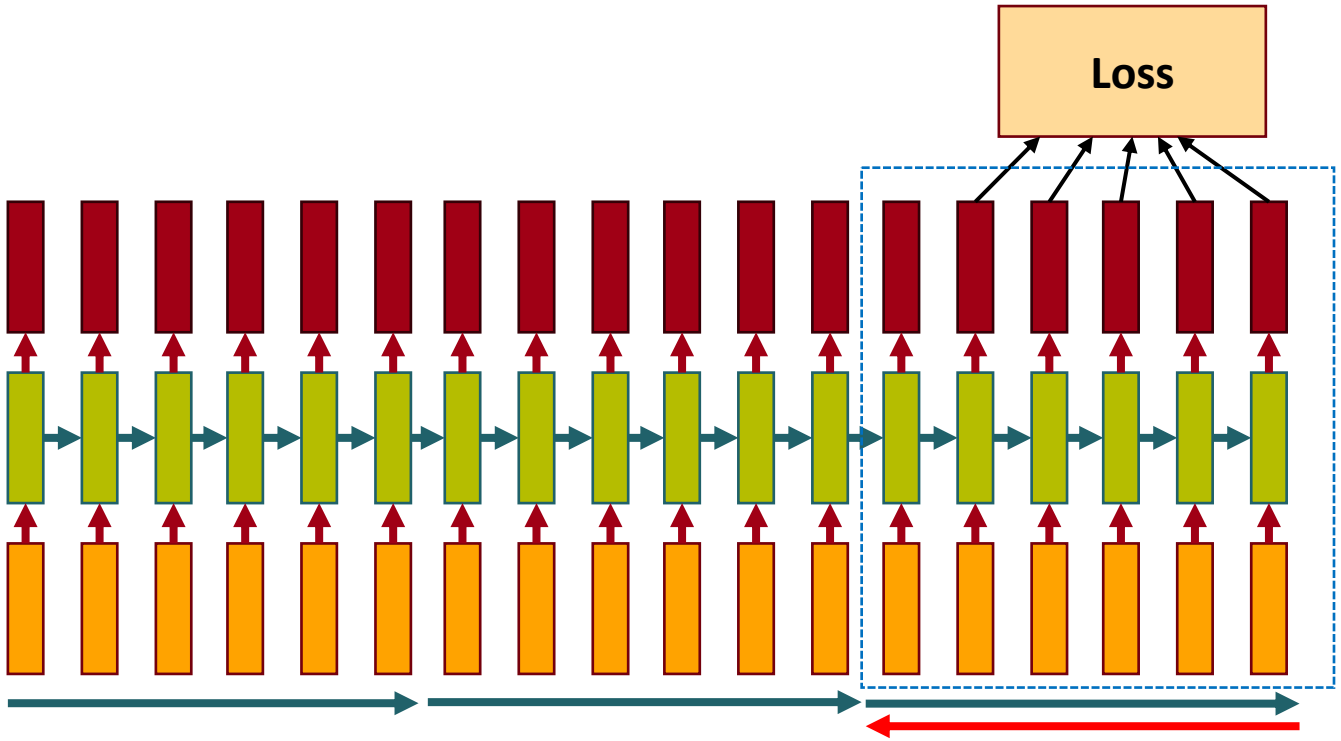
- Run forward and backward through chunks of sequence and not through whole sequence



Truncated Backpropagation Through Time

UNIVERSITY OF CALGARY

- Run forward and backward through chunks of sequence and not through whole sequence



Single Cell RNN Gradient Flow

UNIVERSITY OF CALGARY

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

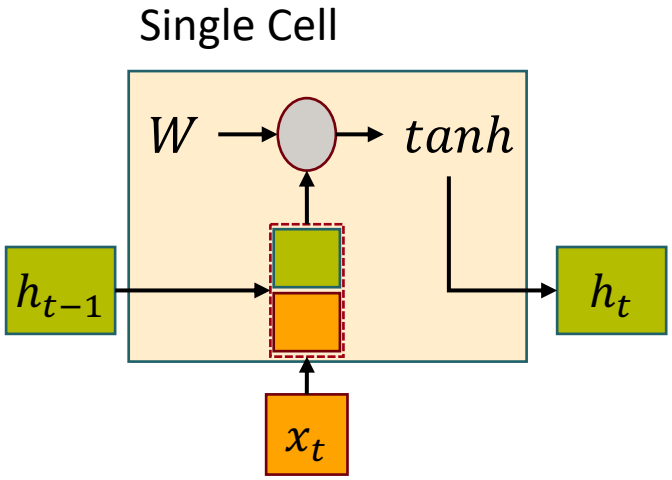
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

Single Cell RNN Gradient Flow

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

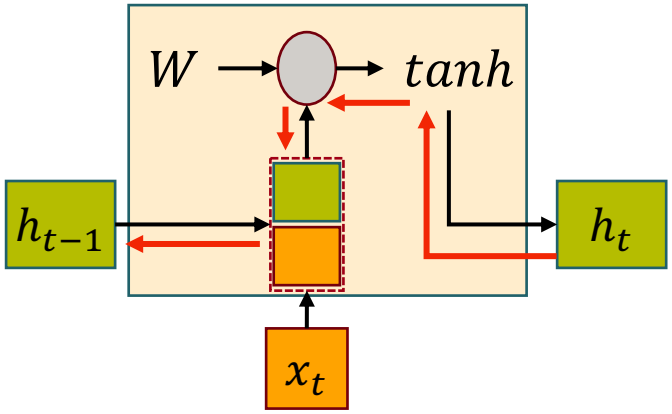


Single Cell RNN Gradient Flow

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$



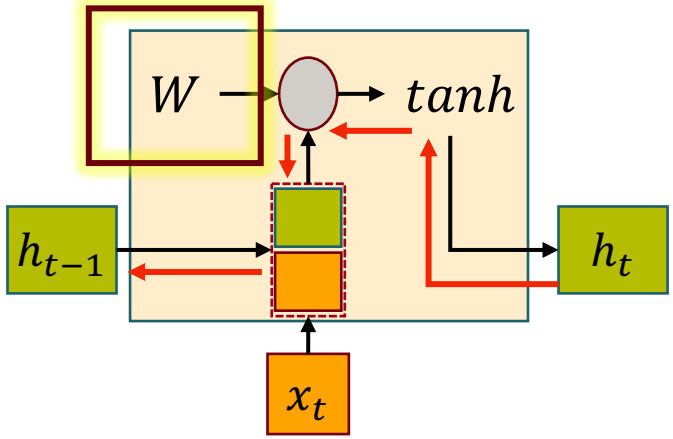
Single Cell RNN Gradient Flow

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

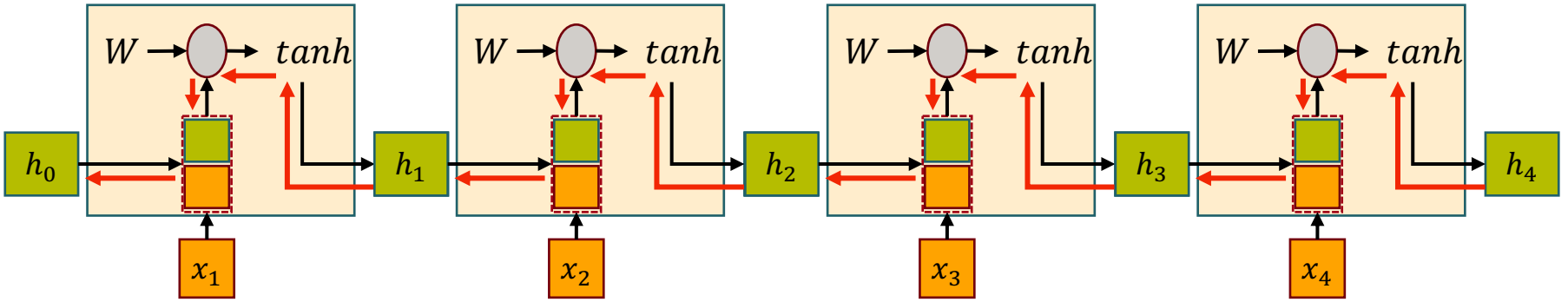
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Backpropagation from h_t to h_{t-1} multiplies by W^T



Single Cell RNN Gradient Flow



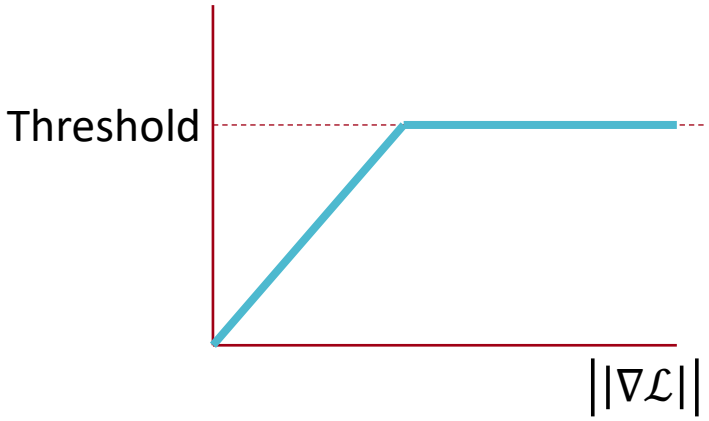
Computing gradients with respect to h_0 involves many W^T and repeated $tanh$ which is computationally expensive

Exploding Gradients

UNIVERSITY OF CALGARY

- Multiplicative gradient that can be exponentially **increasing** with respect to the number of layers
- Largest Singular Value > 1

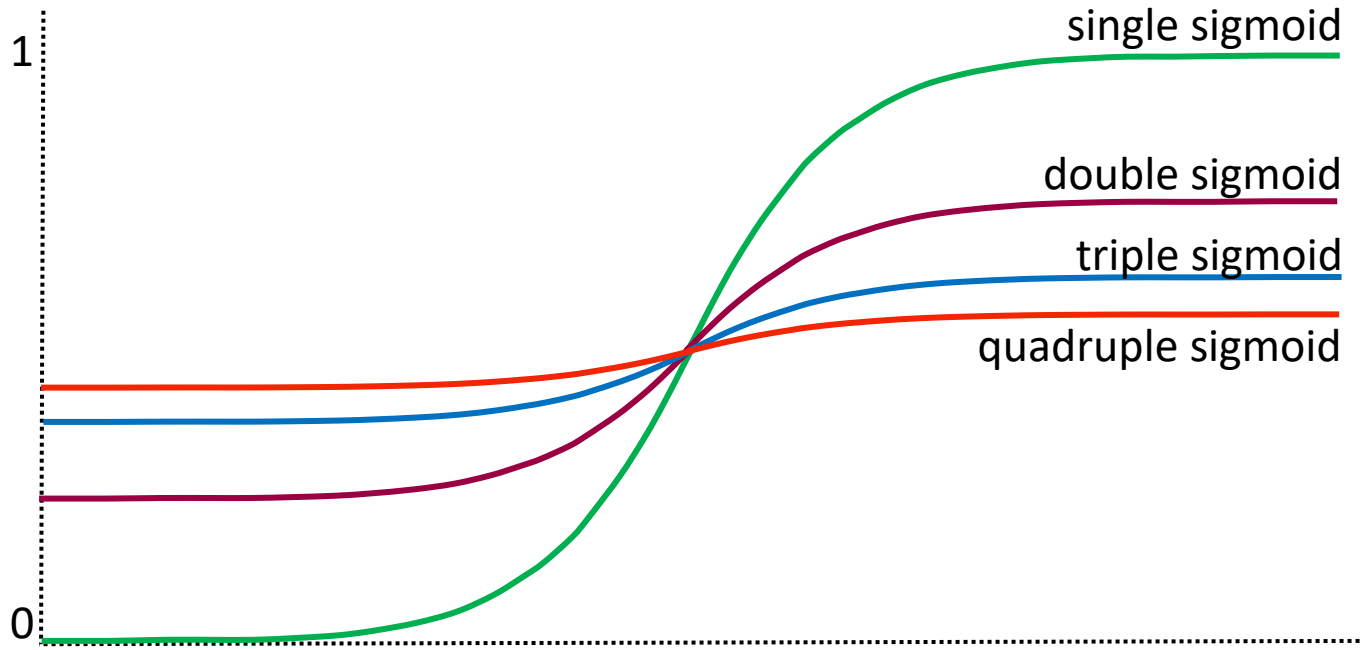
- **Solution**: Gradient Clipping



Vanishing Gradients

UNIVERSITY OF CALGARY

- Multiplicative gradient that can be exponentially **decreasing** with respect to the number of layers
- Largest Singular Value < 1



Long Short-Term Memory Units (LSTMs)

UNIVERSITY OF CALGARY

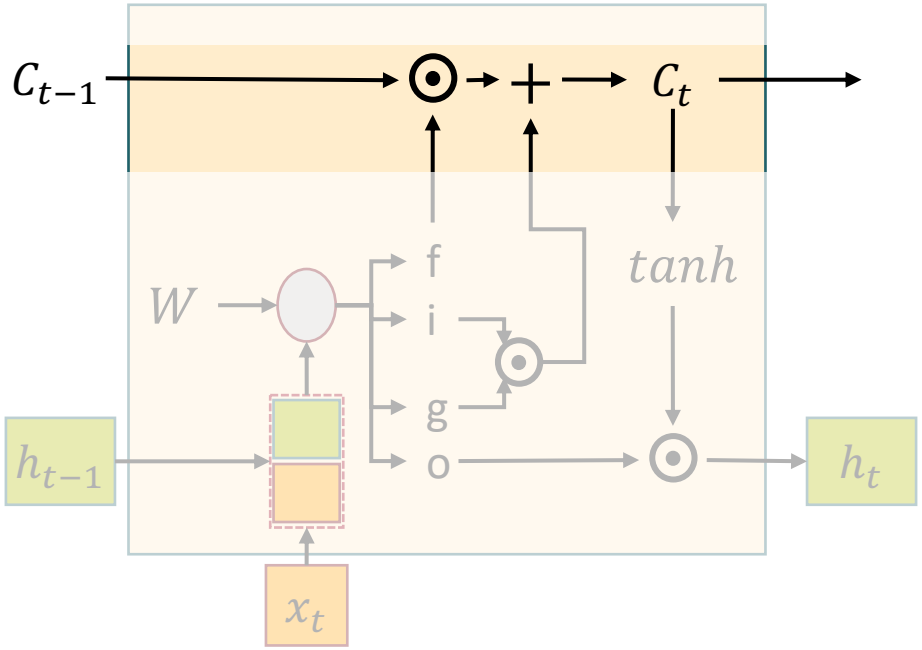
LSTM (Hochreiter and Schmidhuber, 1997)

LSTM cell is a designed unit of logic that will help reduce the **vanishing gradient** problem sufficiently to make recurrent neural networks capable of learning long-term dependencies.

Long Short-Term Memory Units (LSTMs)

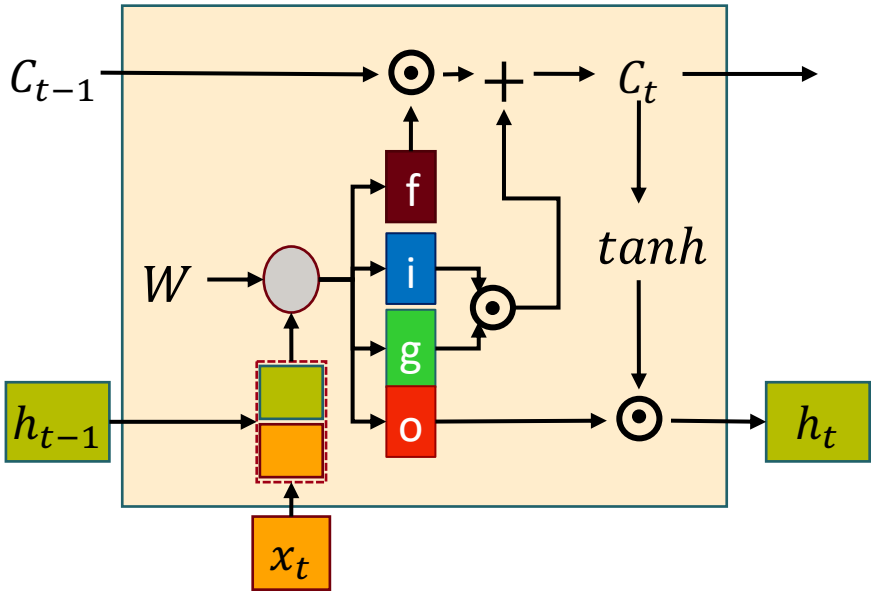
The key to LSTMs is the cell state highway line!

This line has only some **minor linear interactions**



LSTM

- f: Forget gate, Whether to erase cell
- i: Input gate, whether to write to cell
- g: Gate (NoName!) gate, how much to write to cell
- o: Output gate, how much to reveal cell



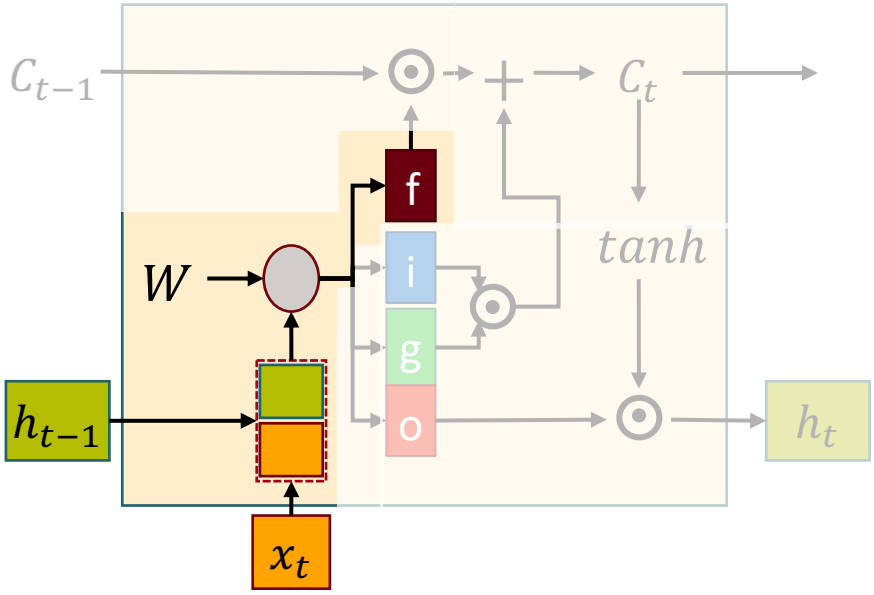
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_t \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = \mathrm{o} \odot tanh(c_t)$$

Step-by-step LSTM Walk Through

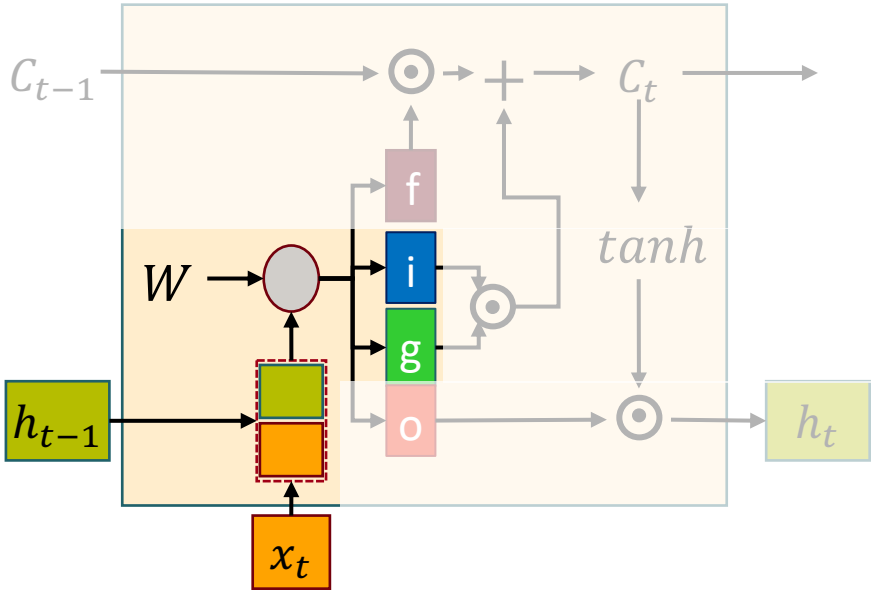
- **1st step :** what information we're going to throw away from the cell state.
- "forget gate layer" -> sigmoid



$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right)$$

Step-by-step LSTM Walk Through

- **2nd step :** what new information we're going to store in the cell state.
- "input gate layer" -> sigmoid
- "NoName gate layer" -> tanh : creates a vector of new candidate values

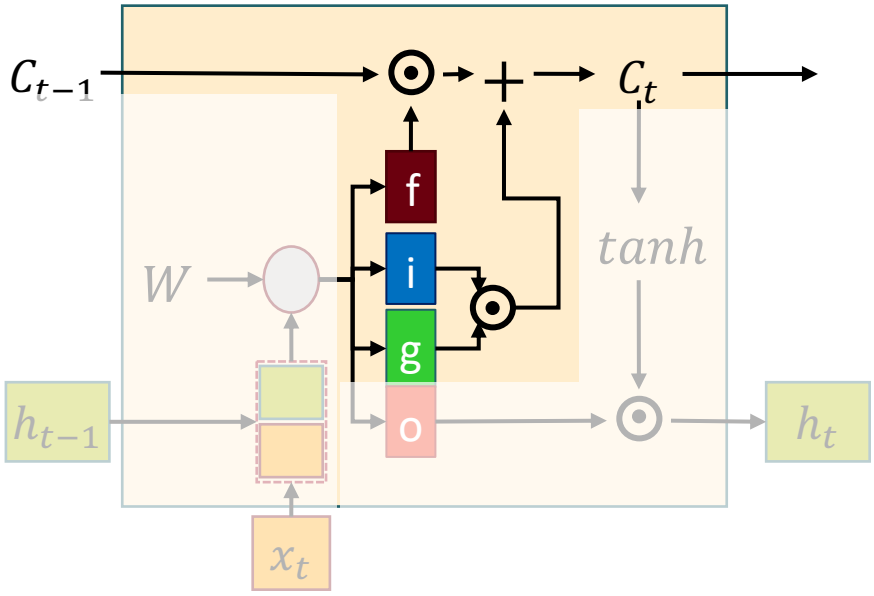


$$i_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_i\right)$$

$$\widetilde{C}_t = tanhs\left(W_f.[h_{t-1}, x_t] + b_C\right)$$

Step-by-step LSTM Walk Through

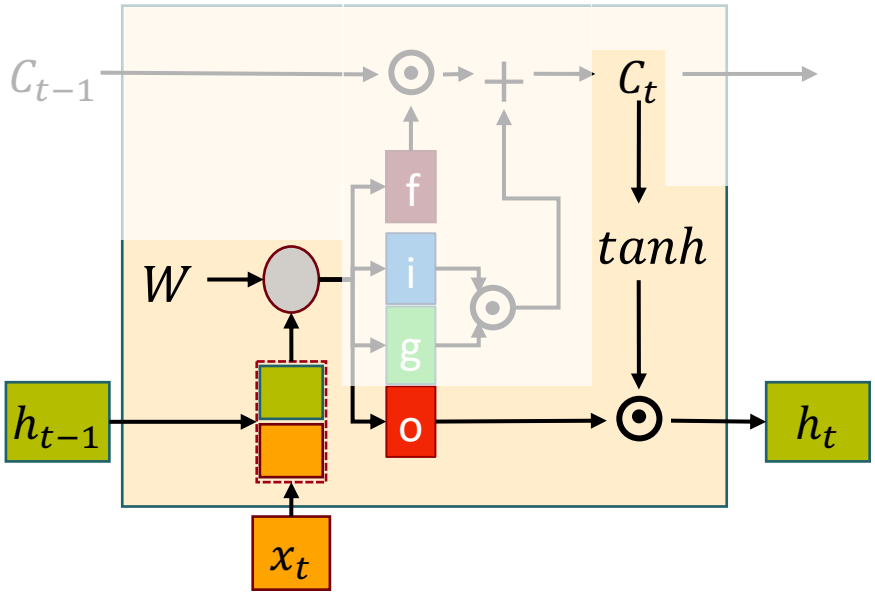
- **3d step :** Update the old cell state C_{t-1}, into new cell state C_t



$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

Step-by-step LSTM Walk Through

- **Final step :** decide what we're going to output.
- The output is a filter version of cell state

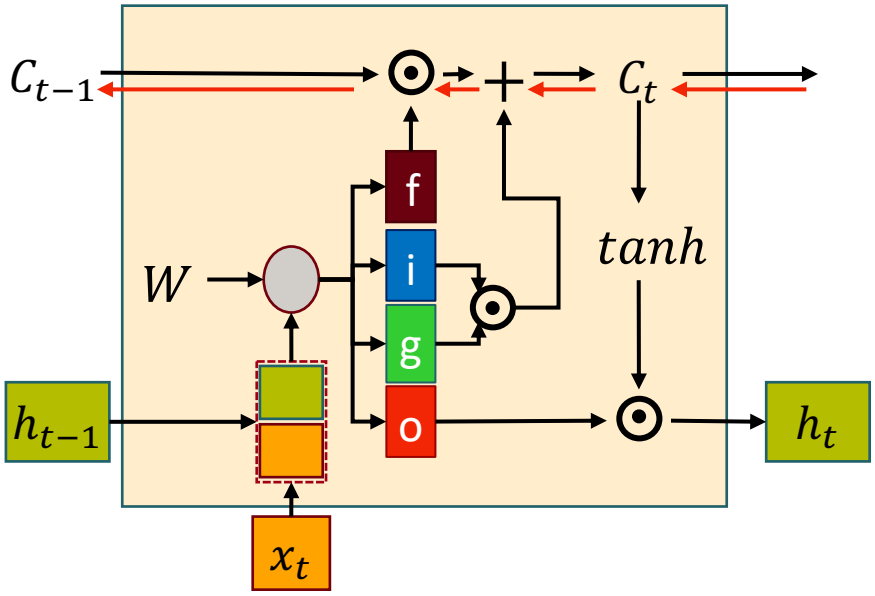


$$o_t = \sigma(\, W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh\,(C_t)$$

LSTM Gradient Flow

- Backpropagation from C_t to C_{t-1} only elementwise multiplication by "forget gate".

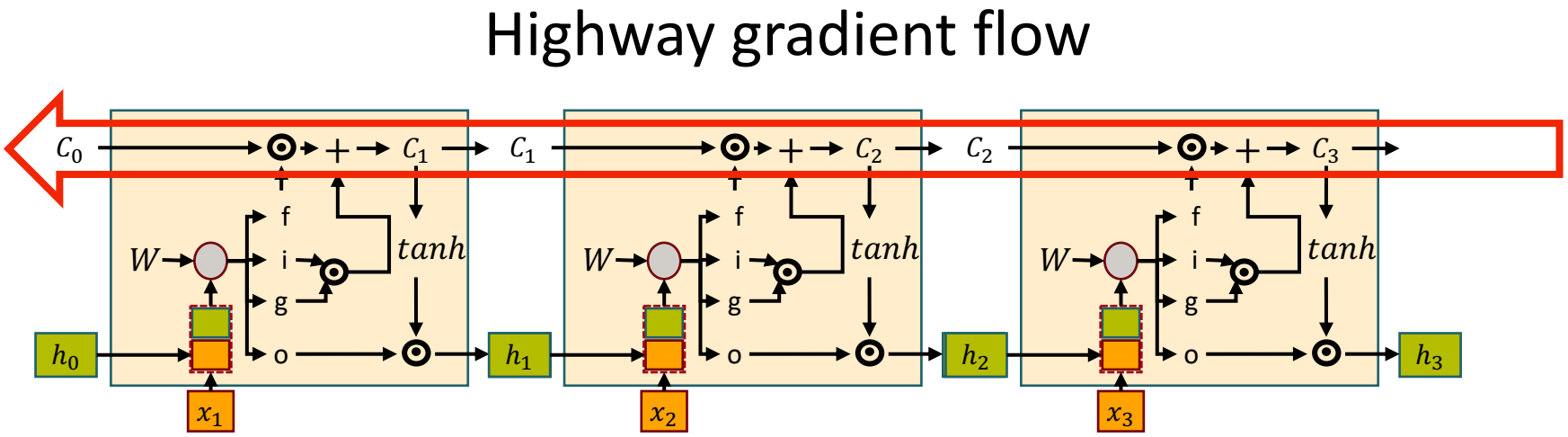


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_t \\ x_t \end{pmatrix}$$

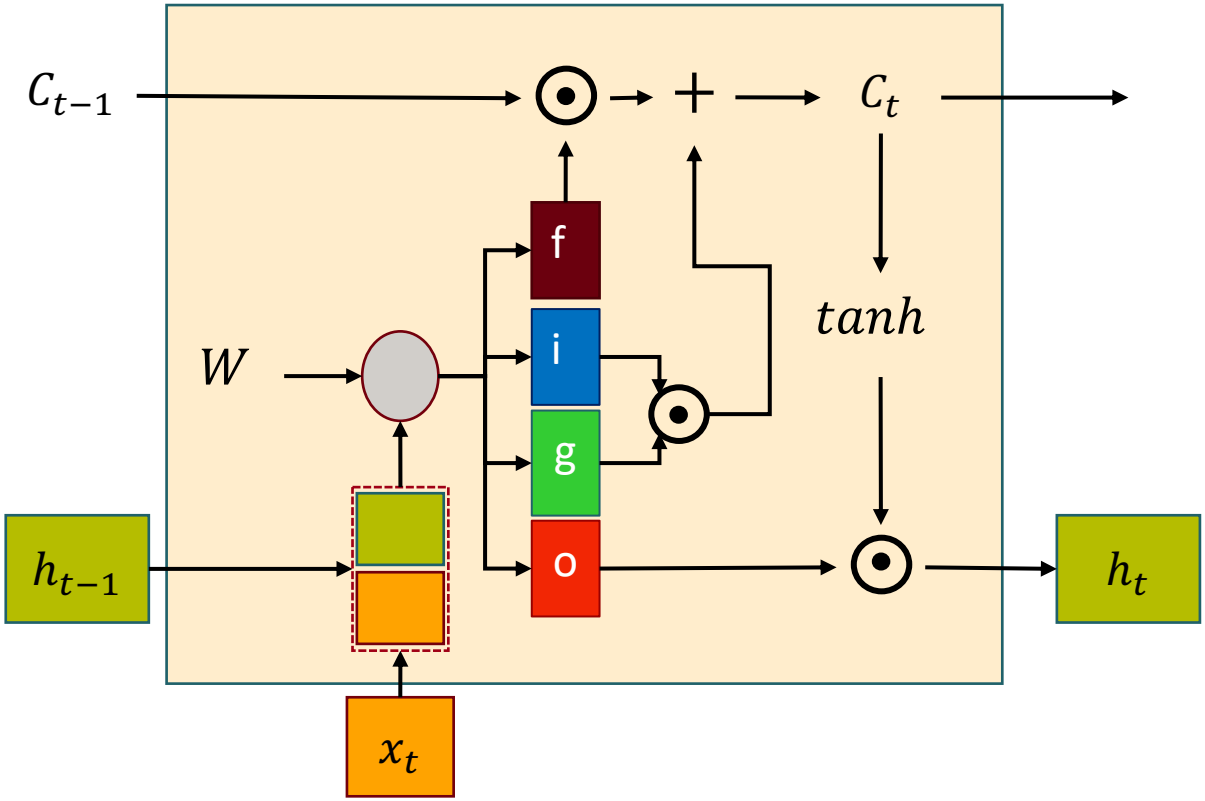
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = \mathrm{o} \odot tanh(c_t)$$

LSTM Gradient Flow



LSTM



UNIVERSITY OF CALGARY

Other Types of RNNs …

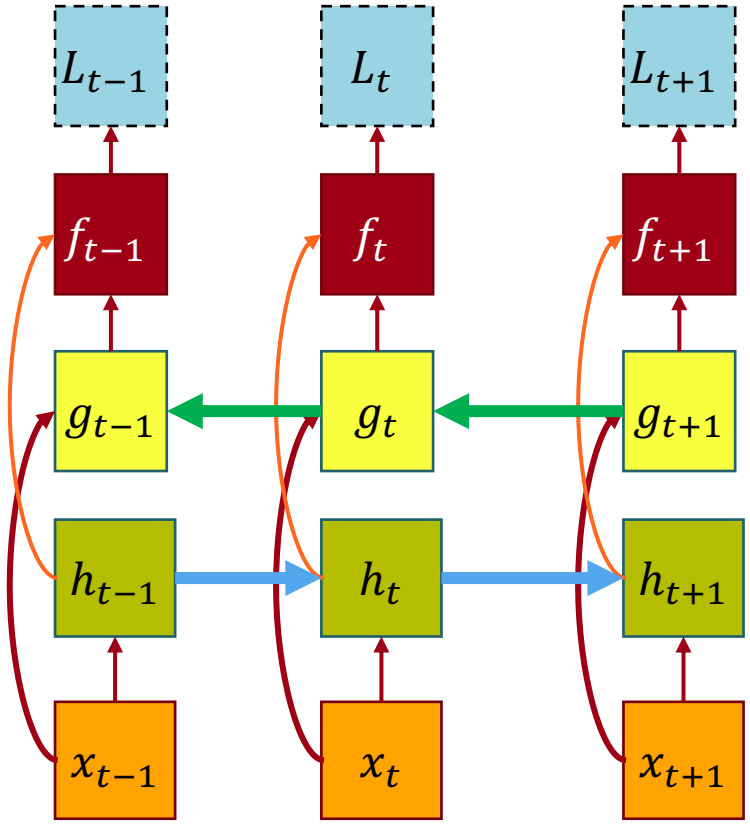
Deep RNN

UNIVERSITY OF CALGARY

- The hidden recurrent state can be broken down into groups organized hierarchically.
- Deeper computation can be introduced in the input-to hidden, hidden-to-hidden and hidden-to-output parts. This may lengthen the shortest path linking different time steps.
- The path-lengthening effect can be mitigated by introducing skip connections.

Bidirectional RNN

- many applications we want to output a prediction of y(t) which may depend on the whole input sequence.



Applications

- Drawing pictures:
  - [1] DRAW: A Recurrent Neural Network For Image Generation
- Computer-composed music
  - [2] Song From PI: A Musically Plausible Network for Pop Music Generation
- Semantic segmentation
  - [3] Conditional random fields as recurrent neural networks

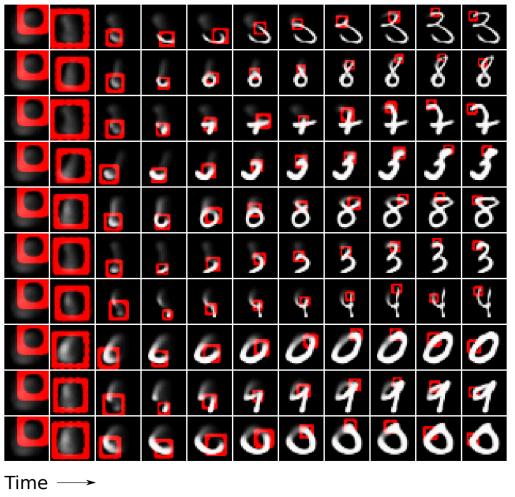


Fig. A trained DRAW network generating MNIST digits.

[1] Gregor, Karol, et al. "Draw: A recurrent neural network for image generation." *arXiv preprint arXiv:1502.04623* (2015). [2] Chu, Hang, Raquel Urtasun, and Sanja Fidler. "Song from pi: A musically plausible network for pop music generation." *arXiv preprint arXiv:1611.03477* (2016). [3] Zheng, Shuai, et al. "Conditional random fields as recurrent neural networks." *Proceedings of the IEEE international conference on computer vision*. 2015.

Medical Applications

UNIVERSITY OF CALGARY

Journal of the American Medical Informatics Association, 24(2), 2017, 361–370
doi: 10.1093/jamia/ocw112
Advance Access Publication Date: 13 August 2016
Research and Applications

Research and Applications

Using recurrent neural network models for early detection of heart failure onset

Edward Choi,[1] Andy Schuetz,[2] Walter F Stewart,[2] and Jimeng Sun[1]

[1]Georgia Institute of Technology, Atlanta and [2]Sutter Health, Walnut Creek, California

Correspondence to Jimeng Sun, School of Computational Science and Engineering, Georgia Institute of Technology, 266 Ferst Drive Atlanta, GA 30313, USA; jsun@cc.gatech.edu; Tel: 404.894.0482

Received 27 February 2016; Revised 30 June 2016; Accepted 5 July 2016

ABSTRACT

Objective: We explored whether use of deep learning to model temporal relations among events in elec health records (EHRs) would improve model performance in predicting initial diagnosis of heart failur compared to conventional methods that ignore temporality.

Materials and Methods: Data were from a health system's EHR on 3884 incident HF cases and 28 903 cor identified as primary care patients, between May 16, 2000, and May 23, 2013. Recurrent neural network models using gated recurrent units (GRUs) were adapted to detect relations among time-stamped event disease diagnosis, medication orders, procedure orders, etc.) with a 12- to 18-month observation wind cases and controls. Model performance metrics were compared to regularized logistic regression, neura work, support vector machine, and K-nearest neighbor classifier approaches.

Results: Using a 12-month observation window, the area under the curve (AUC) for the RNN model was

Published in final edited form as:
JMLR Workshop Conf Proc. 2016 August ; 56: 301–318.

Doctor AI: Predicting Clinical Events via Recurrent Neural Networks

Edward Choi and **Mohammad Taha Bahadori**
College of Computing Georgia Institute of Technology Atlanta, GA, USA

Andy Schuetz and **Walter F. Stewart**
Research Development & Dissemination Sutter Health Walnut Creek, CA, USA

Jimeng Sun
College of Computing Georgia Institute of Technology Atlanta, GA, USA

Abstract

Leveraging large historical data in electronic health record (EHR), we developed Doctor AI, a generic predictive model that covers observed medical conditions and medication uses. Doctor AI is a temporal model using recurrent neural networks (RNN) and was developed and applied to longitudinal time stamped EHR data from 260K patients over 8 years. Encounter records (e.g. diagnosis codes, medication codes or procedure codes) were input to RNN to predict (all) the

ses the history of

Convolutional Recurrent Neural Networks for Dynamic MR Image Reconstruction

Chen Qin, Jo Schlemper, Jose Caballero, Anthony N. Price, Joseph V. Hajnal, and Daniel Rueckert, *Fellow, IEEE*

***Abstract*— Accelerating the data acquisition of dynamic magnetic resonance imaging leads to a challenging ill-posed inverse problem, which has received great interest from both the signal processing and machine learning communities over the last decades. The key ingredient to the problem is how to exploit the temporal correlations of the MR sequence to resolve aliasing artifacts. Traditionally, such observation led to a formulation of an optimization problem, which was solved using iterative algorithms. Recently, however, deep learning-based approaches have gained significant popularity due to their ability to solve general inverse problems. In this paper, we propose a unique, novel convolutional recurrent neural network architecture which reconstructs high quality cardiac MR images from highly undersampled k-space data by jointly exploiting the dependencies of the temporal sequences as well as**

diagnosis and research. Dynamic MRI attempts to reveal both spatial and temporal profiles of the underlying anatomy, which has a variety of applications such as cardiovascular imaging and perfusion imaging. However, the acquisition speed is fundamentally limited due to both hardware and physiological constraints as well as the requirement to satisfy the Nyquist sampling rate. Long acquisition times are not only a burden for patients but also make MRI susceptible to motion artefacts.

In order to accelerate MRI acquisition, most approaches consider undersampling the data in *k*-space (frequency domain). Due to the violation of the Nyquist sampling theorem, undersampling introduces aliasing artefacts in the image

RNN vs Markov

UNIVERSITY OF CALGARY

- Not having Markov assumption
- Long term dependencies
- representational power
- intelligent smoothing by taking into account syntactic and semantic features
- Transfer learning

Recurrent Reinforcement Learning (RRL)

UNIVERSITY OF CALGARY

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 12, NO. 4, JULY 2001, PP 875–889

Learning to Trade via Direct Reinforcement

John Moody, Matthew Saffell

Abstract— **We present methods for optimizing portfolios, asset allocations and trading systems based on Direct Reinforcement. In this approach, investment decision making is viewed as a stochastic control problem, and strategies are discovered directly. We present an adaptive algorithm called Recurrent Reinforcement Learning (RRL) for discovering investment policies. The need to build forecasting models is eliminated, and better trading performance is obtained.**

The Direct Reinforcement approach differs from dynamic programming and reinforcement algorithms such as TD-

decisions about establishing new positions must consider current positions held.

In Moody et al [1], [2], we proposed the RRL algorithm for Direct Reinforcement. RRL is an adaptive *policy search* algorithm that can learn an investment strategy on-line. We demonstrated in those papers that Direct Reinforcement provides a more elegant and effective means for training trading systems and portfolio managers when market frictions are considered than do more standard supervised approaches.

In this paper, we contrast our *Direct Reinforcement* (or "policy search") approach with commonly used value function based approaches. We use the term "Direct Reinforcement" to refer to algorithms that *do not* have to learn a value function in order to derive a policy. Direct Reinforcement methods date back to the pioneering work by Farley

Recurrent Reinforcement Learning: A Hybrid Approach

Xiujun Li[1], Lihong Li[2], Jianfeng Gao[2], Xiaodong He[2], Jianshu Chen[2], Li Deng[2], Ji He[3]
lixiujun@cs.wisc.edu
{lihongli, jfgao, xiaohe, jianshuc, deng}@microsoft.com
jvking@uw.edu
[1]University of Wisconsin - Madison [2]Microsoft Research [3]University of Washington - Seattle

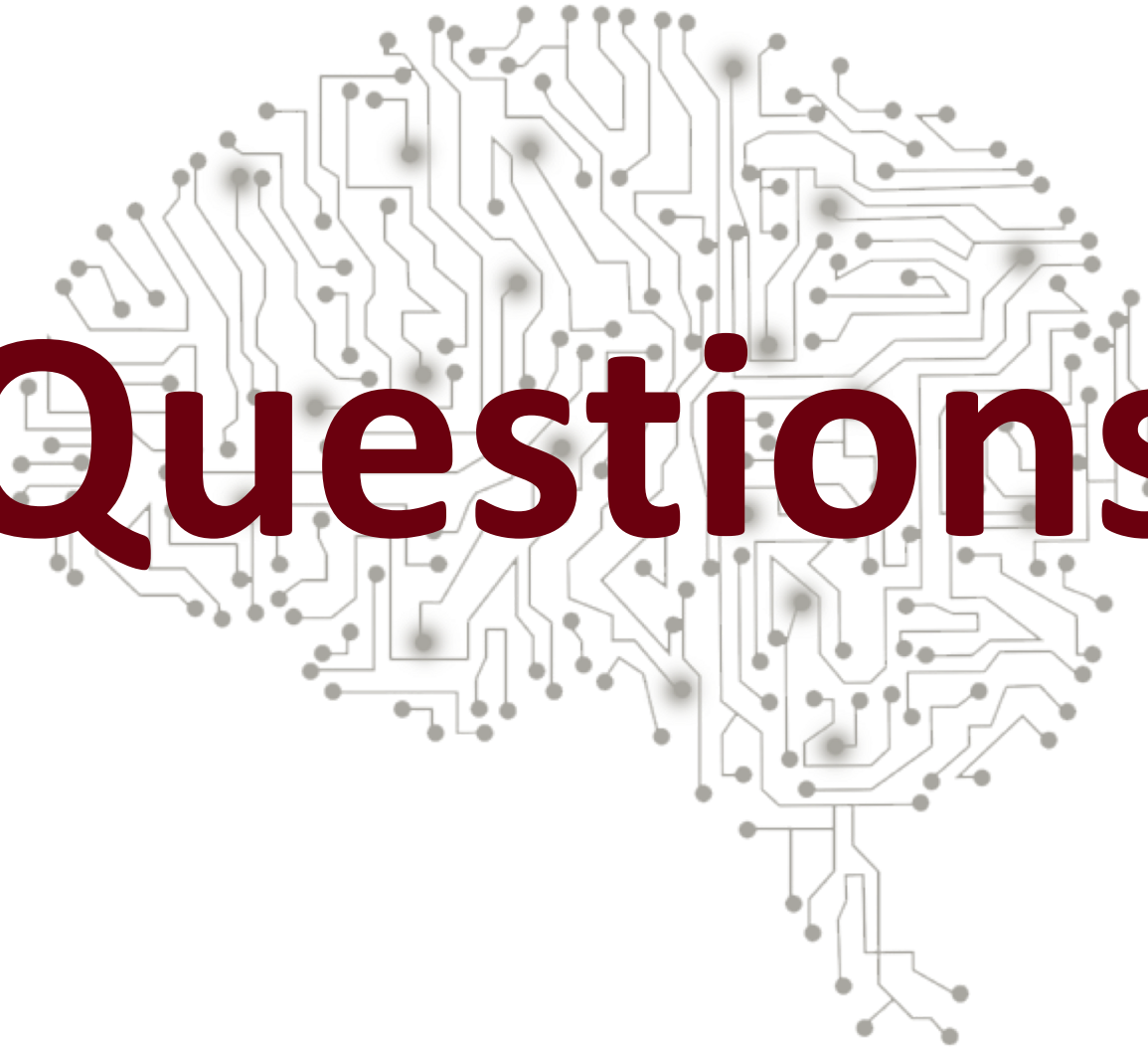
Abstract

Successful applications of reinforcement learning in real-world problems often require dealing with partially observable states. It is in general very challenging to construct and infer hidden states as they often depend on the agent's entire interaction history and may require substantial domain knowledge. In this work, we investigate a deep-learning approach to learning the representation of states in partially observable tasks, with minimal prior knowledge of the domain. In particular, we propose a new family of hybrid models that combines the strength of both supervised learning (SL) and reinforcement learning (RL), trained in a joint fashion: The SL component can be a recurrent neural networks (RNN) or its long short-term memory (LSTM) version, which is equipped with the desired property of being able to capture long-term dependency on history, thus providing an effective way of learning the representation of hidden states. The RL component is a deep Q-network (DQN) that learns to optimize the control for maximizing

Summary

UNIVERSITY OF CALGARY

- RNNs are good for sequential data
- Wide variety of recurrent neural network architectures, based on different application.
- Simple Vanilla RNN does not work well in practice
- LSTM a solution for vanishing gradient
- LSTM are more useful for long-term memory tasks


Questions?