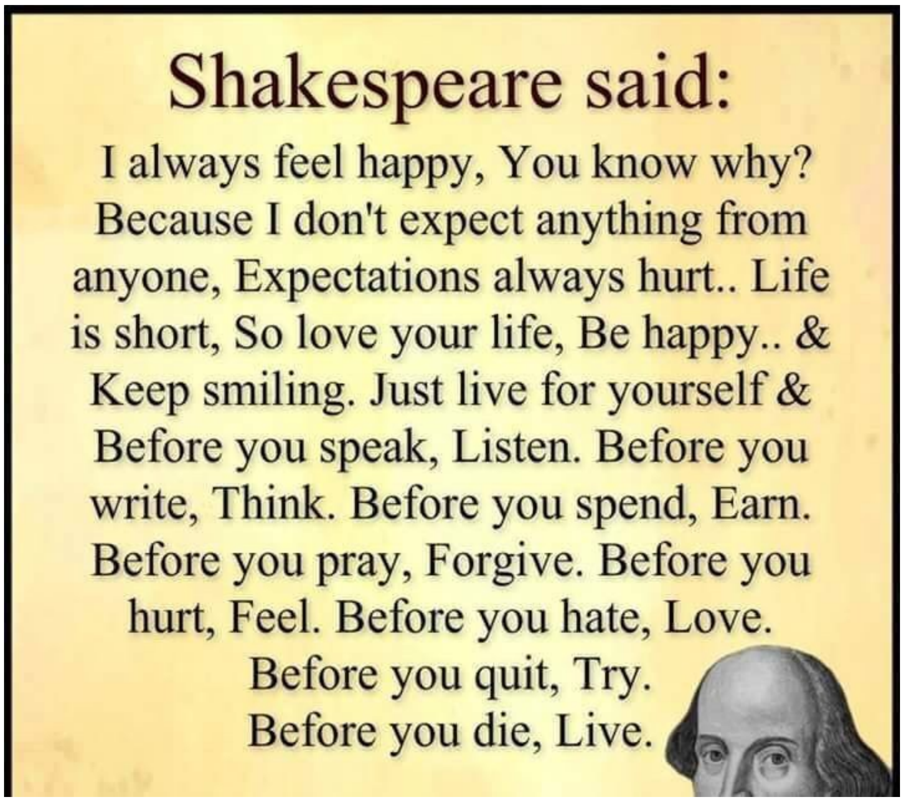
Machine Learning Using Tensorflow

Week 7:
Recurrent Neural Network

Shu-Ting Pi, PhD
UC Davis

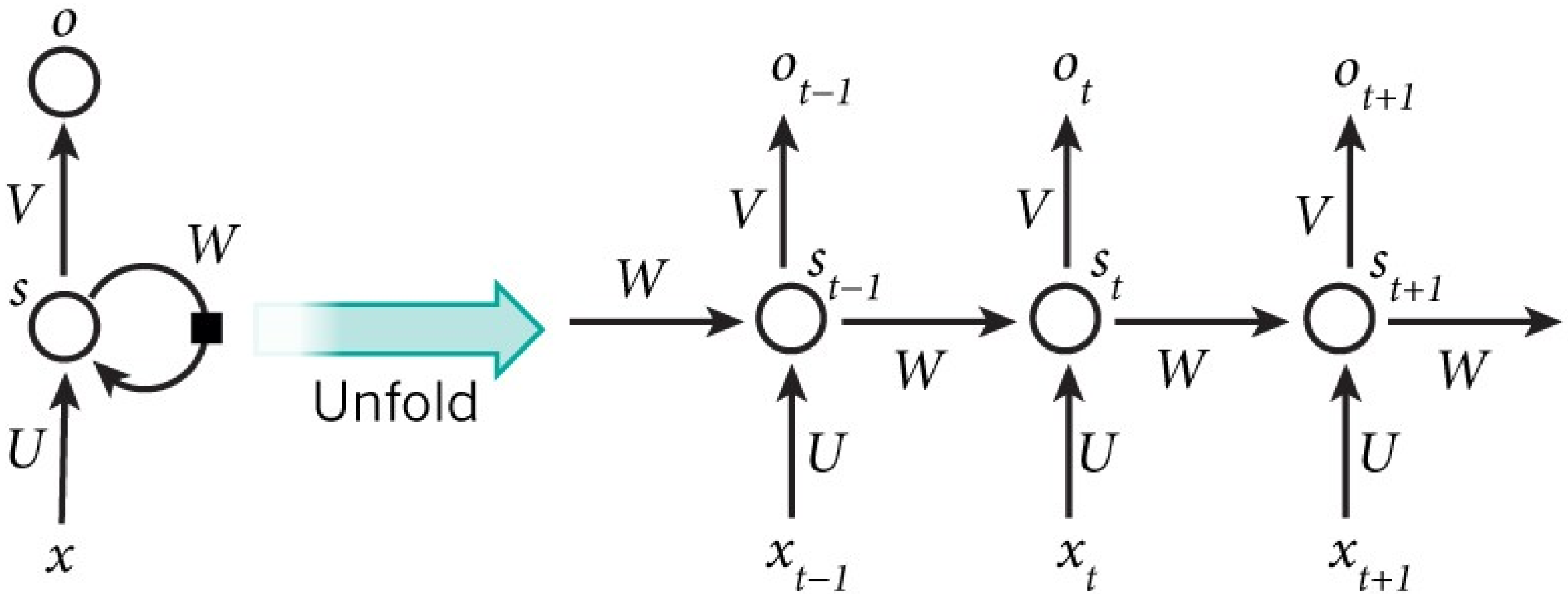
TensorFlow ™

Time Series Problem



Data sequence are not considered in MLP or CNN !

Recurrent Neural Network (RNN)



$$
\begin{aligned}
o_t &= g(Vs_t) \\
s_t &= f(Ux_t + Ws_{t-1})
\end{aligned}
$$

$$
\begin{aligned}
o_t &= g(Vs_t) \\
&= Vf(Ux_t + Ws_{t-1}) \\
&= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2})) \\
&= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3}))) \\
&= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \ldots))))
\end{aligned}
$$

Details of Cell

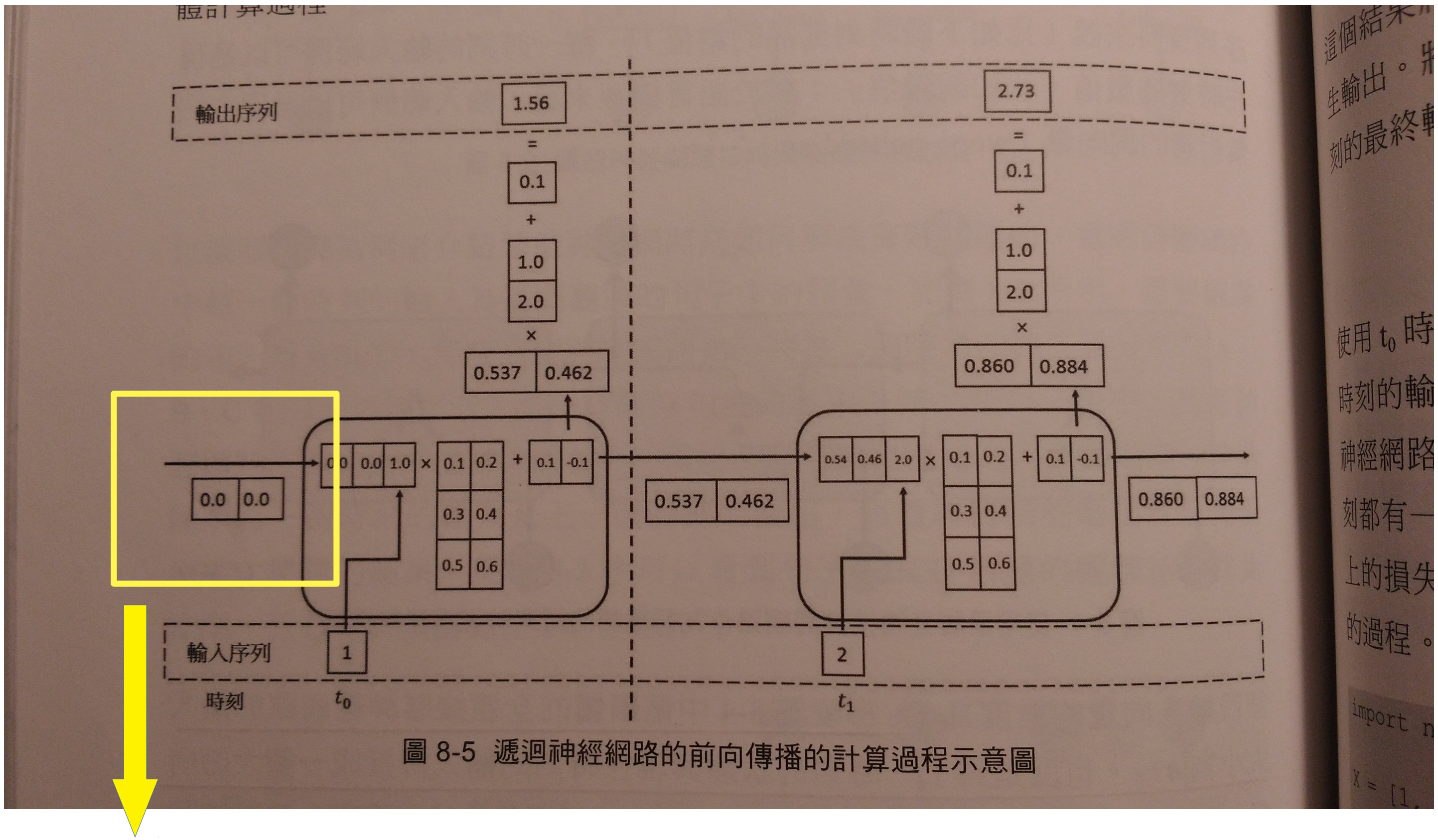
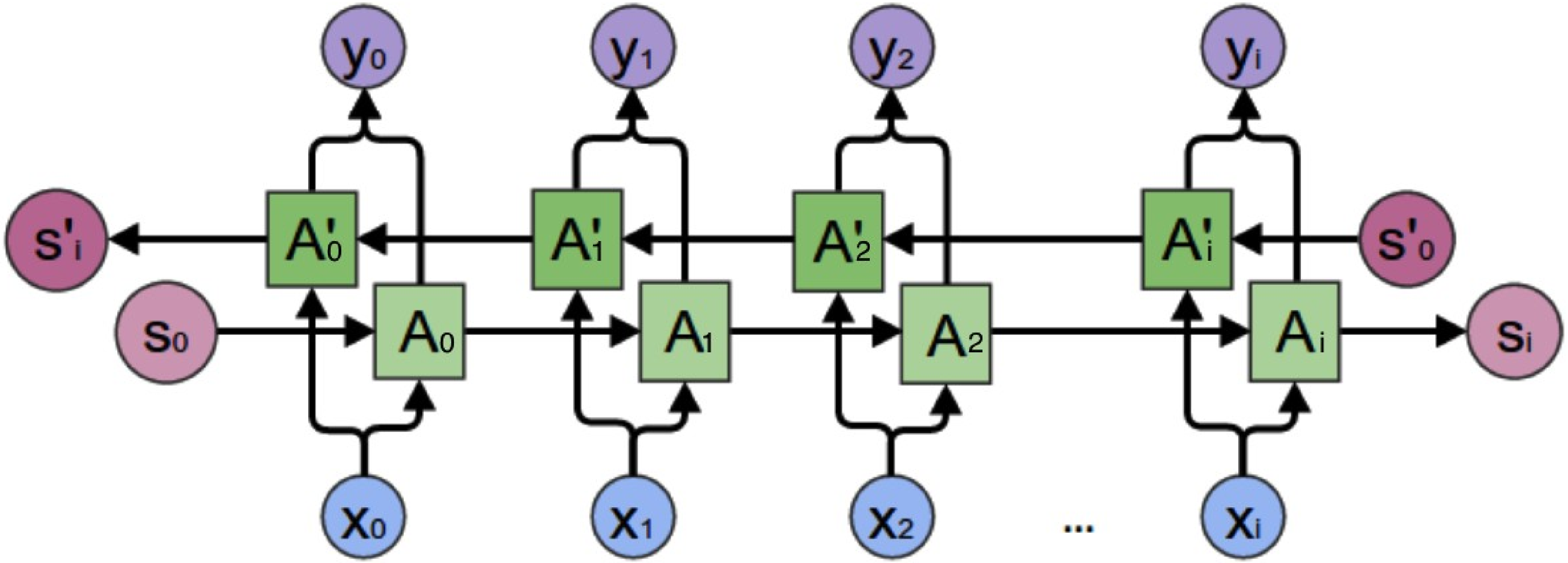


圖 8-5 遞迴神經網路的前向傳播的計算過程示意圖

Size of the “state” (or size of hidden unit, or size of the cell) is something you need to specify !

Bidirectional RNN

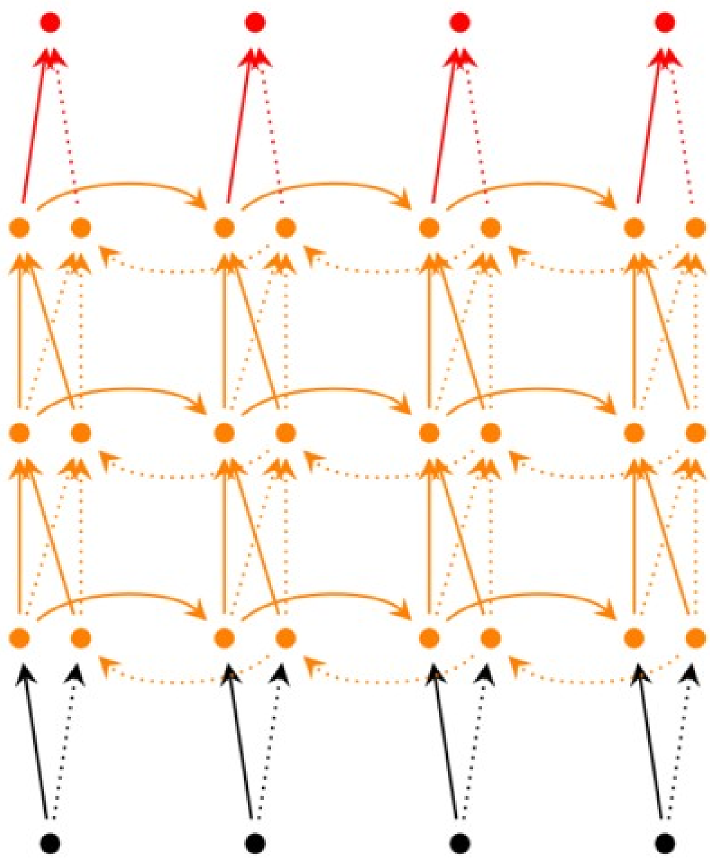


$$y_2 = g(VA_2 + V'A_2')$$

$$A_2 = f(WA_1 + Ux_2)$$
$$A_2' = f(W'A_3' + U'x_2)$$

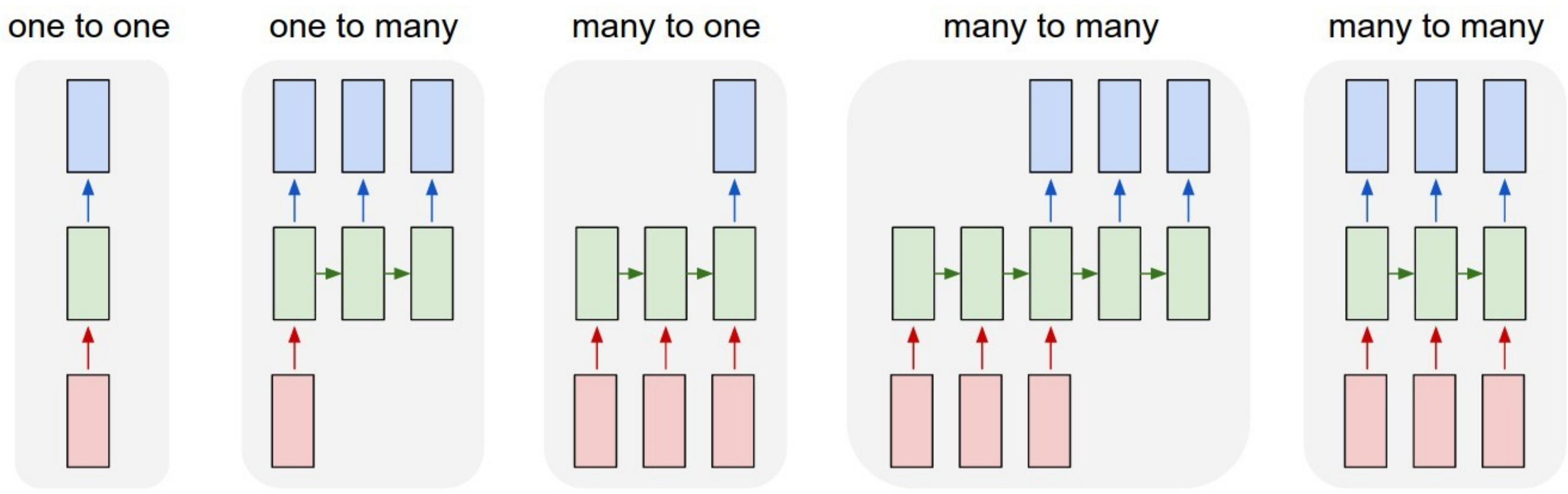
$$o_t = g(Vs_t + V's_t')$$
$$s_t = f(Ux_t + Ws_{t-1})$$
$$s_t' = f(U'x_t + W's_{t+1}')$$

Deep RNN

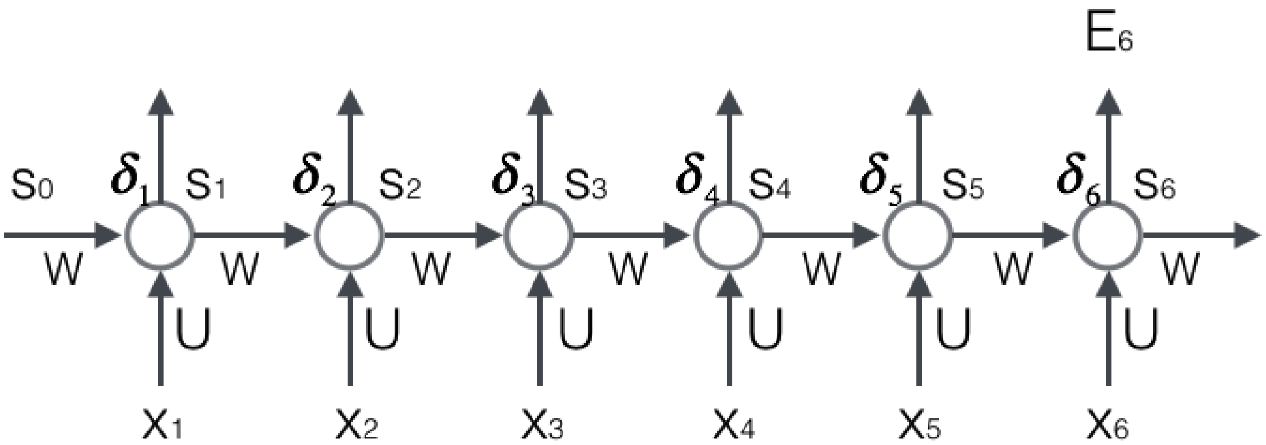


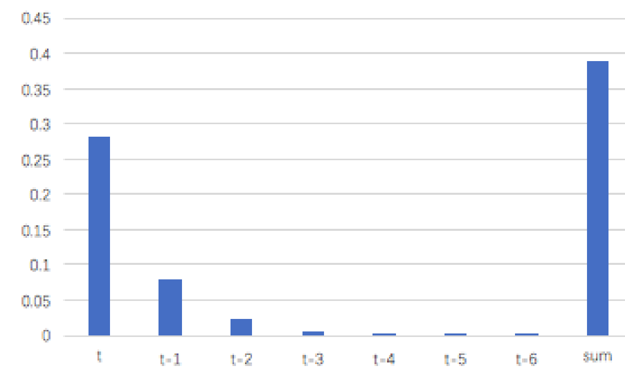
$$
\begin{aligned}
o_t &= g(V^{(i)} s_t^{(i)} + V'^{(i)} s_t'^{(i)}) \\
s_t^{(i)} &= f(U^{(i)} s_t^{(i-1)} + W^{(i)} s_{t-1}) \\
s_t'^{(i)} &= f(U'^{(i)} s_t'^{(i-1)} + W'^{(i)} s'_{t+1}) \\
&\cdots \\
s_t^{(1)} &= f(U^{(1)} x_t + W^{(1)} s_{t-1}) \\
s_t'^{(1)} &= f(U'^{(1)} x_t + W'^{(1)} s'_{t+1})
\end{aligned}
$$

Correspondence of RNN



Gradient Problem





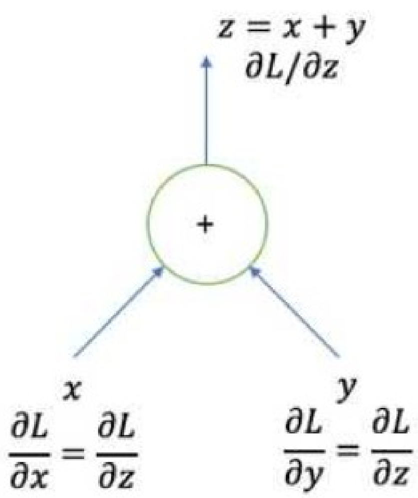
$$\delta_k^T = \delta_t^T \prod_{i=k}^{t-1} W diag[f'(\mathrm{net}_i)]$$

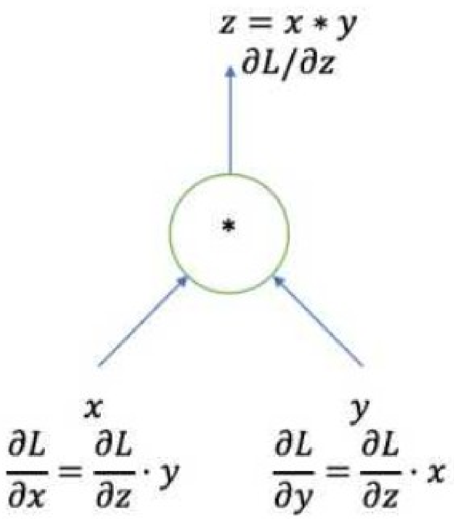
$$\|\delta_k^T\| \leqslant \|\delta_t^T\| \prod_{i=k}^{t-1} \|W\| \|diag[f'(\mathrm{net}_i)]\| \leqslant \|\delta_t^T\| (\beta_W \beta_f)^{t-k}$$

B < 1: Gradient vanishing

B > 1: Gradient explosion

The gates

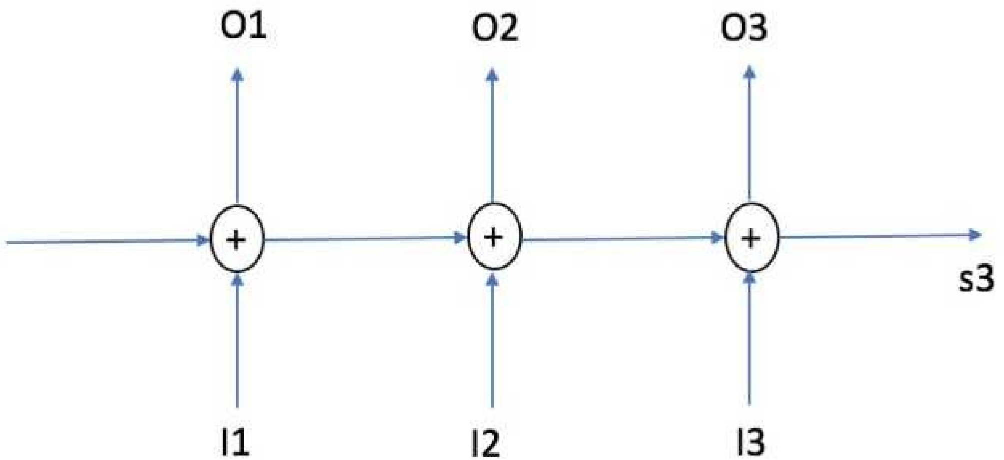




Forward: equal
Backward: equal

Forward: filter
Backward: scale

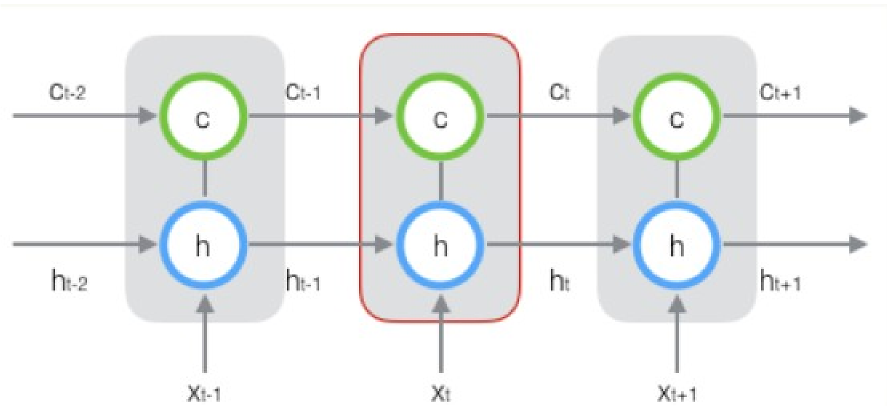
Add the network

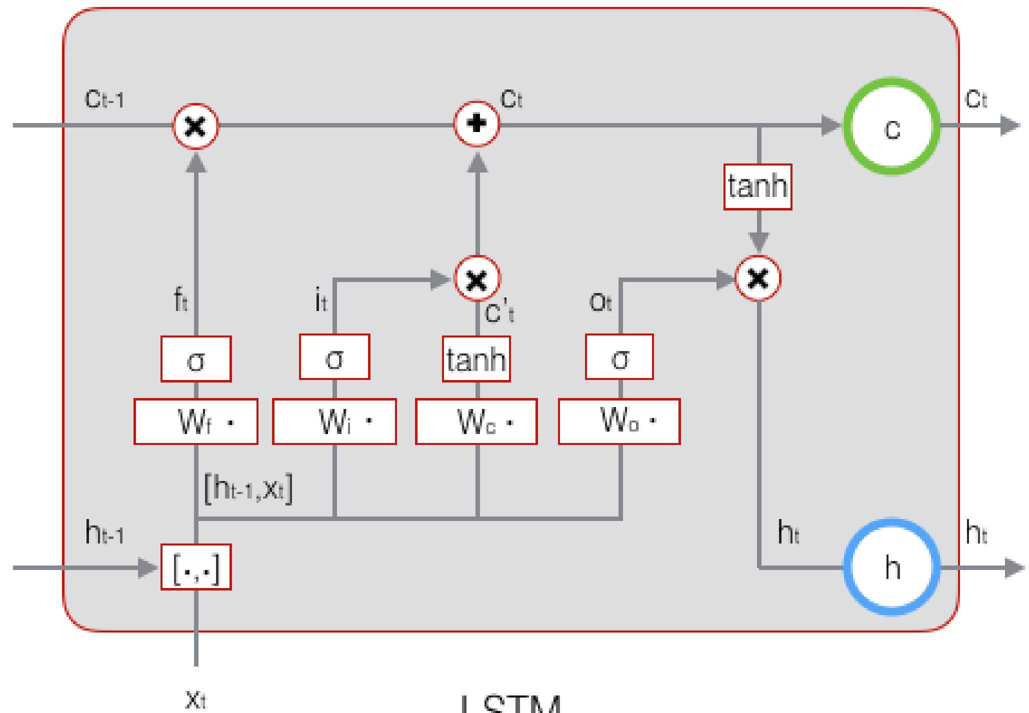


O(1) O(1) O(1)

No gradient vanishing

LSTM

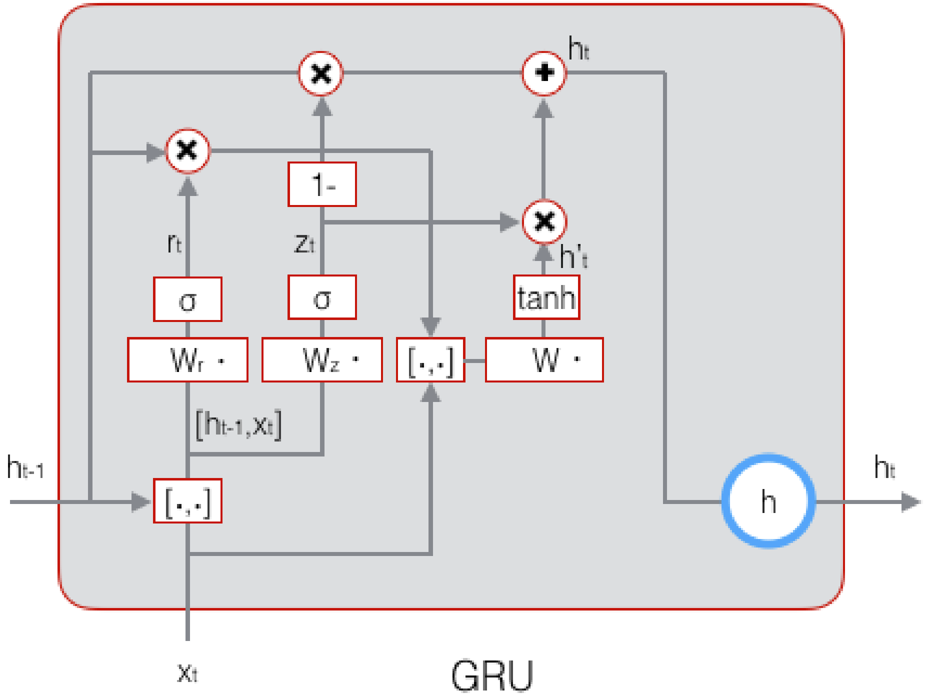




LSTM

f: forget gate, i:input state, o: output gate
c: state, h: output

GRU



Z: update gate, r: reset gate, h: output state

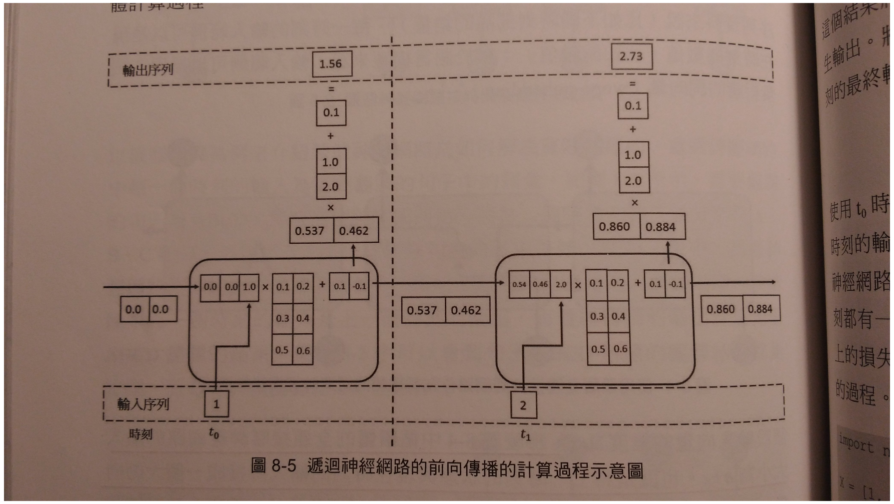
IRNN

A Simple Way to Initialize Recurrent Networks of Rectified Linear Units

Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton
Google

Abstract

Learning long term dependencies in recurrent networks is difficult due to vanishing and exploding gradients. To overcome this difficulty, researchers have developed sophisticated optimization techniques and network architectures. In this paper, we propose a simpler solution that use recurrent neural networks composed of rectified linear units. Key to our solution is the use of the identity matrix or its scaled version to initialize the recurrent weight matrix. We find that our solution is comparable to a standard implementation of LSTMs on our four benchmarks: two toy problems involving long-range temporal structures, a large language modeling problem and a benchmark speech recognition problem.



Use: 1). Relu 2). identity matrix to initialize