IF3270 Pembelajaran Mesin

Recurrent Neural Network (ANN)

Tim Pengajar IF3270

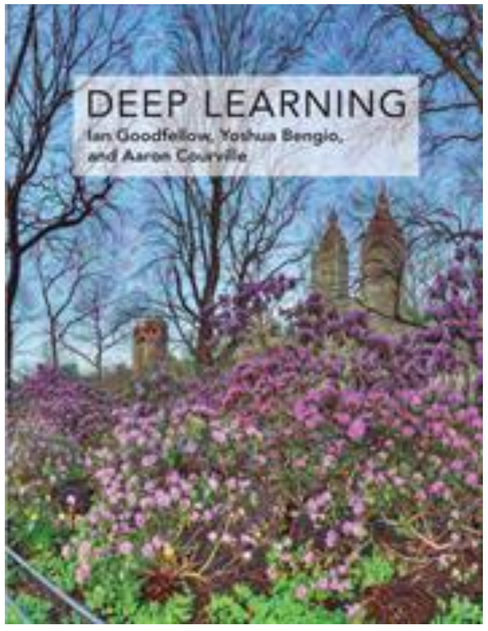
Review

- Machine learning (ML) overview

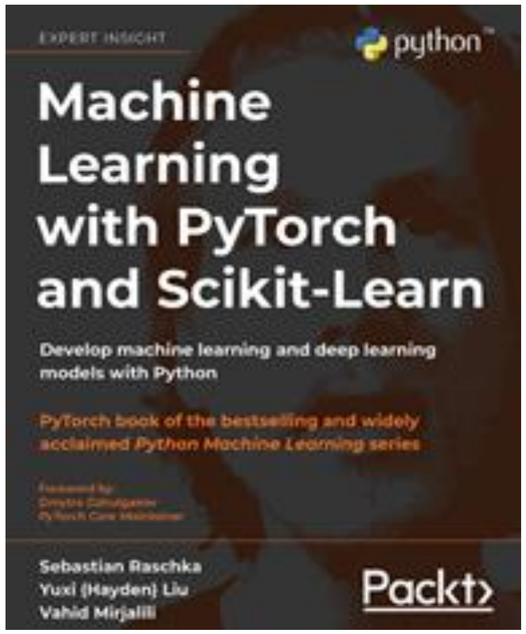
- Ensemble Methods → Supervised Learning

- Supervised Learning: Perceptron
- Supervised Learning: ANN: Feed Forward Neural Network
- Supervised Learning: ANN: Convolutional Neural Network
- Supervised Learning: ANN: Recurrent Neural Network → Today

References

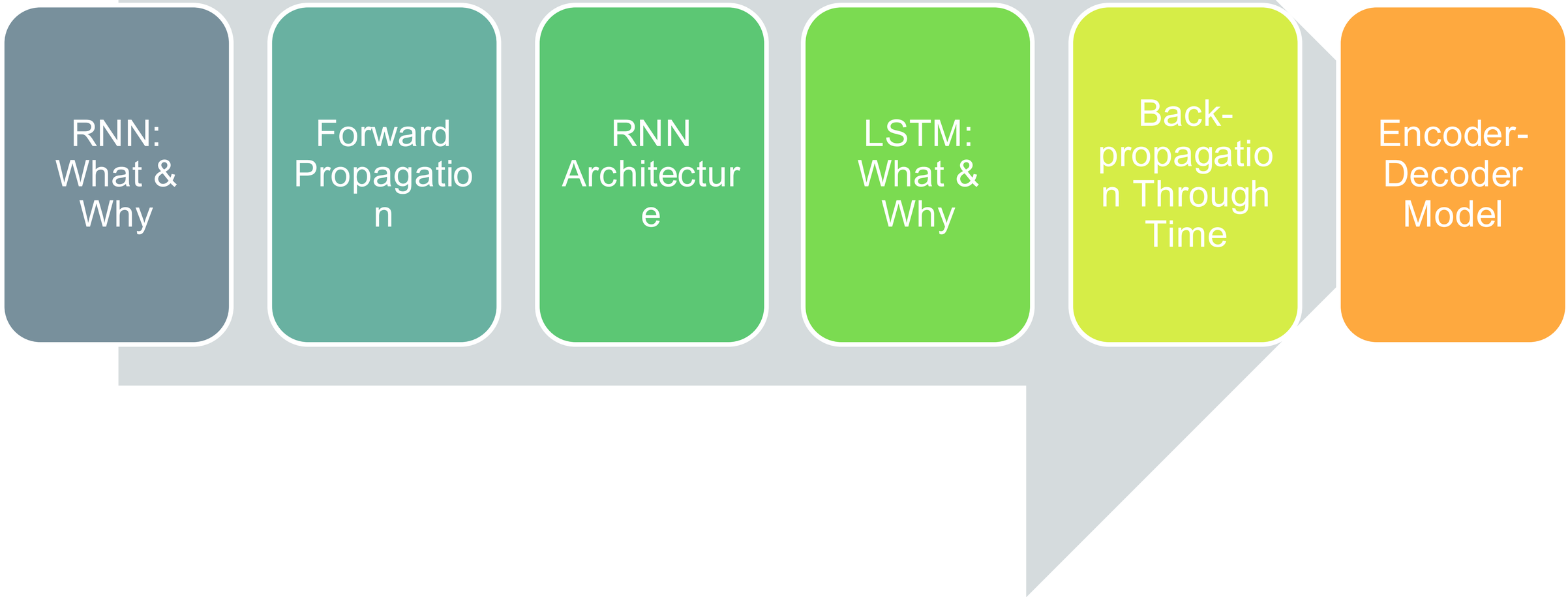


Deep learning. I Goodfellow, Y Bengio, A Courville, Y Bengio. MIT press 1 (2), 2016 (Chapter 10)



Raschka, et.al., Machine Learning with Pytorch and Scikit-Learn, Packt Publishing Ltd., 2022 (Chapter 15)

Outline


RNN: What & Why
Forward Propagation
RNN Architecture
LSTM: What & Why
Back-propagation Through Time
Encoder-Decoder Model

LSTM: What & Why

menangani problem RNN "long term dependency"

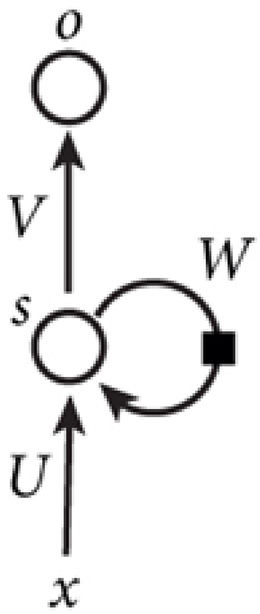
masalah :

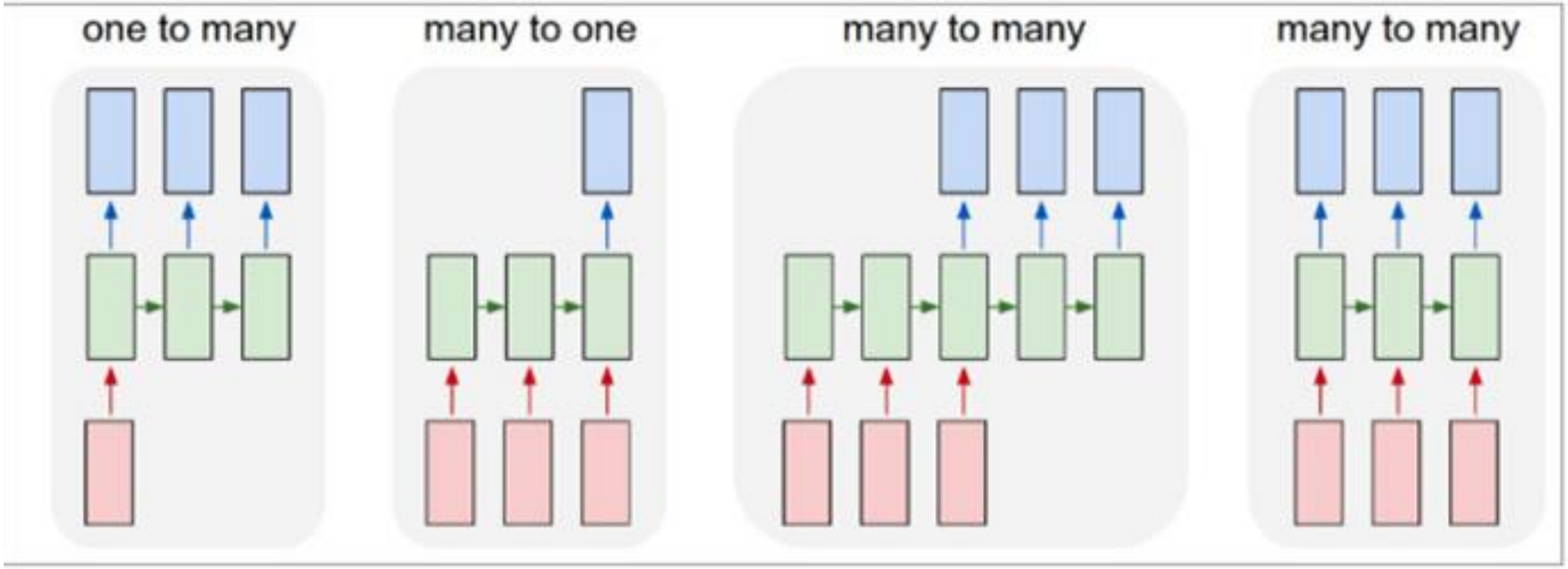
- short term memory (fp)
- long term dependency
- vanishing gradient (bp)

Vanishing Gradient Problem

- Each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training.
- In some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value.
  - In the worst case, this may completely stop the neural network from further training.
- Example:
  - activation functions such as the hyperbolic tangent function have gradients in the range (0, 1), and backpropagation computes gradients by the chain rule.
  - This has the effect of multiplying n of these small numbers to compute gradients of the "front" layers in an n-layer network
    - the gradient (error signal) decreases exponentially with n

Overview RNN



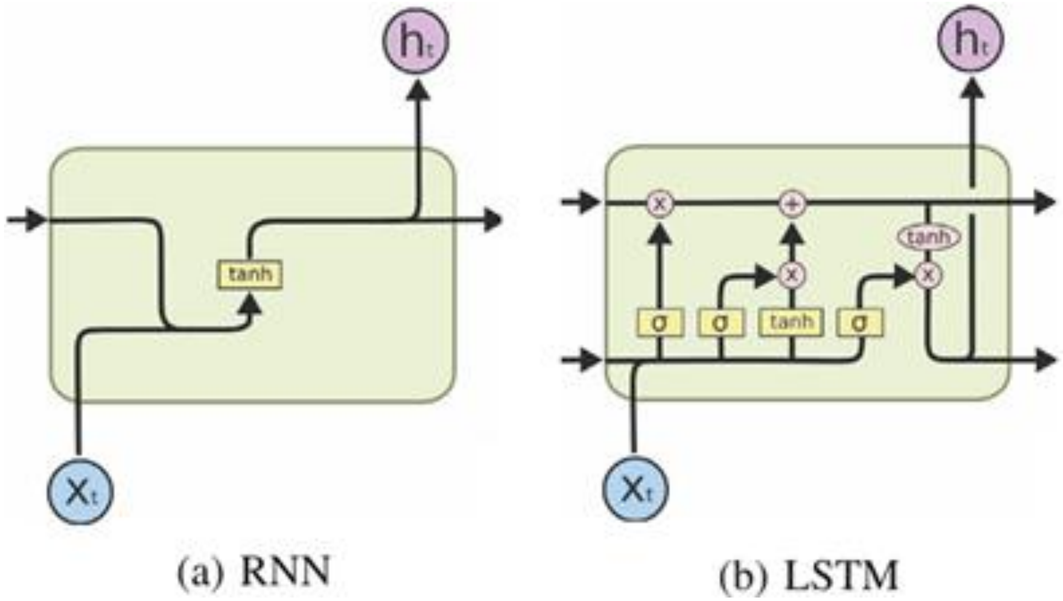


$$s_t = f(Ux_t + Ws_{t-1})$$

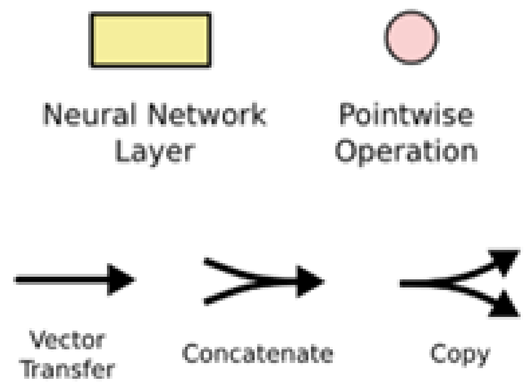
$$o_t = \mathrm{softmax}(Vs_t)$$

Long Short Term Memory (LSTM) Networks

- Introduced by Hochreiter & Schmidhuber (1997), LSTMs are explicitly designed to avoid the long-term dependency problem.
- Special kind of RNN. In standard RNNs, repeating module will have a very simple structure, such as a single tanh layer. Repeating module in an LSTM contains four interacting layers.

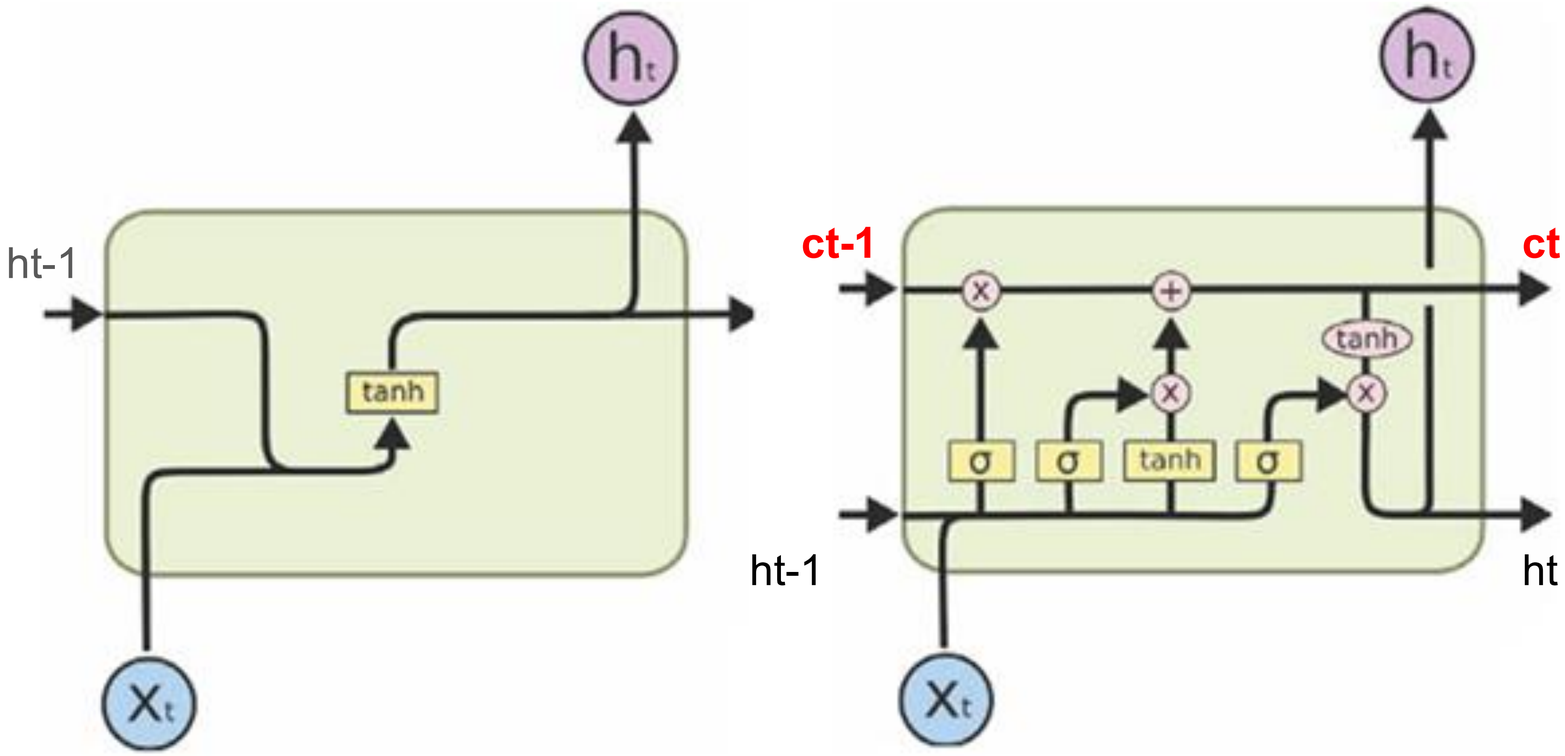


(a) RNN (b) LSTM



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

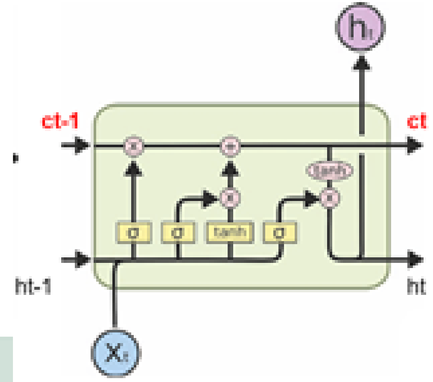
RNN vs LSTM



(a) RNN (b) LSTM

$h_t=\tanh(W\_xh*x_t+(W\_hh*h_{t-1}+bias))$

LSTM: Cell State & Gates



Cell State (Ct)

- as the “memory” of the network
- act as a transport highway that transfers relevant information throughout the processing of the sequence.

Forget Gate

- decides what information should be thrown away or kept.
- Values closer to 0 means to forget, and closer to 1 means to keep.

Input Gate

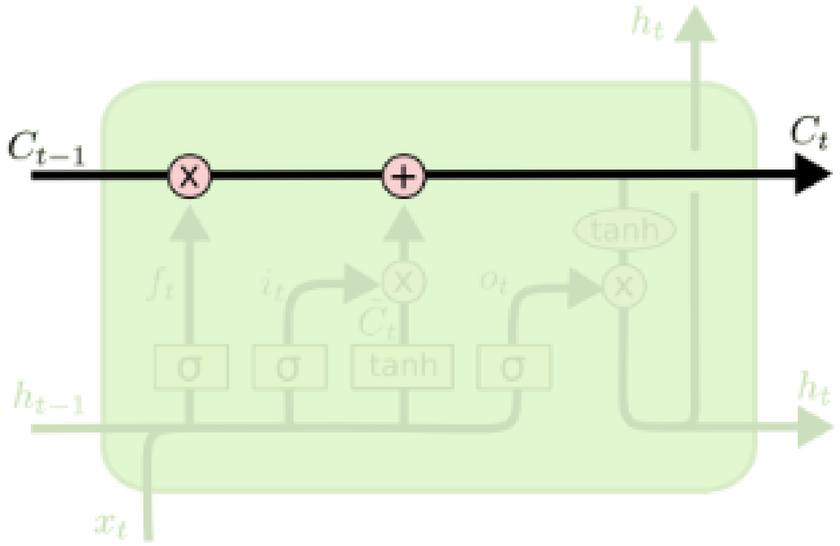
- Decides what information is relevant to add from the current step
- update the cell state by hidden state and current input

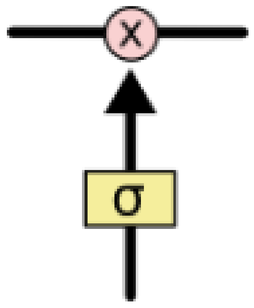
Output Gate

- decides what the next hidden state should be.
- Hidden state contains information on previous inputs. The hidden state is also used for predictions.

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

LSTM Unit





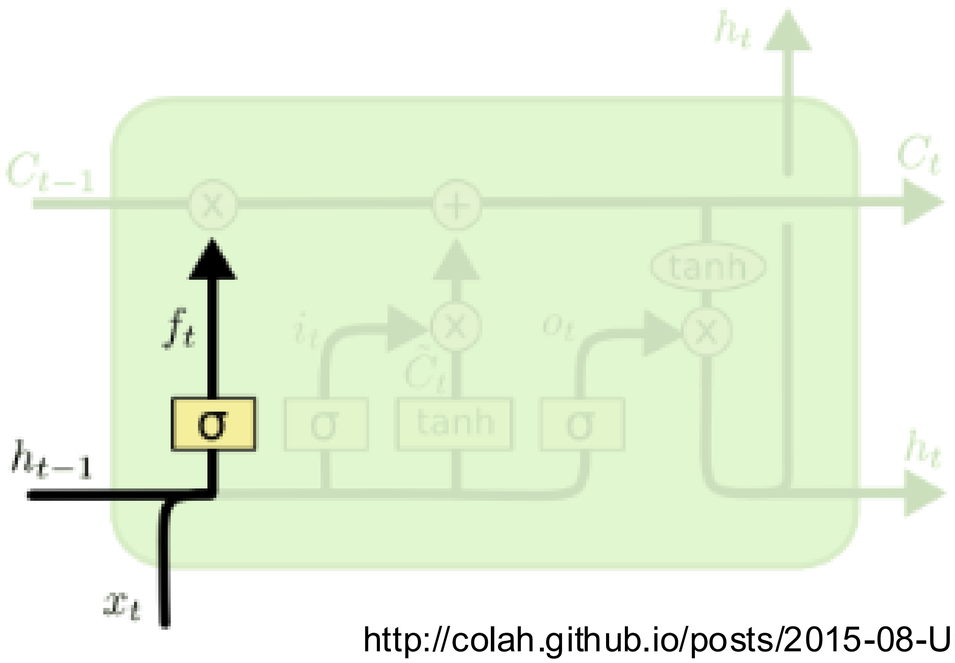
- Cell state ct is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.
- Gates are a way to optionally let information through. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Unit: Forget Gate Layer

menentukan informasi mana yg akan dibuang dr cell state

Forget gate layer: to decide what information we're going to throw away from the cell state. It looks at ht−1 and xt, and outputs a number between 0 and 1 for each number in the cell state Ct−1. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."



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

↳ sama kaya di RNN

$f_t = \sigma\left(u_f \cdot x_t + (w_f \cdot h_{t-1} + b_f)\right)$

σ → fungsi sigmoid ($0 \leq \ldots \leq 1$)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

$f_t \approx 1$ → informasi disimpan

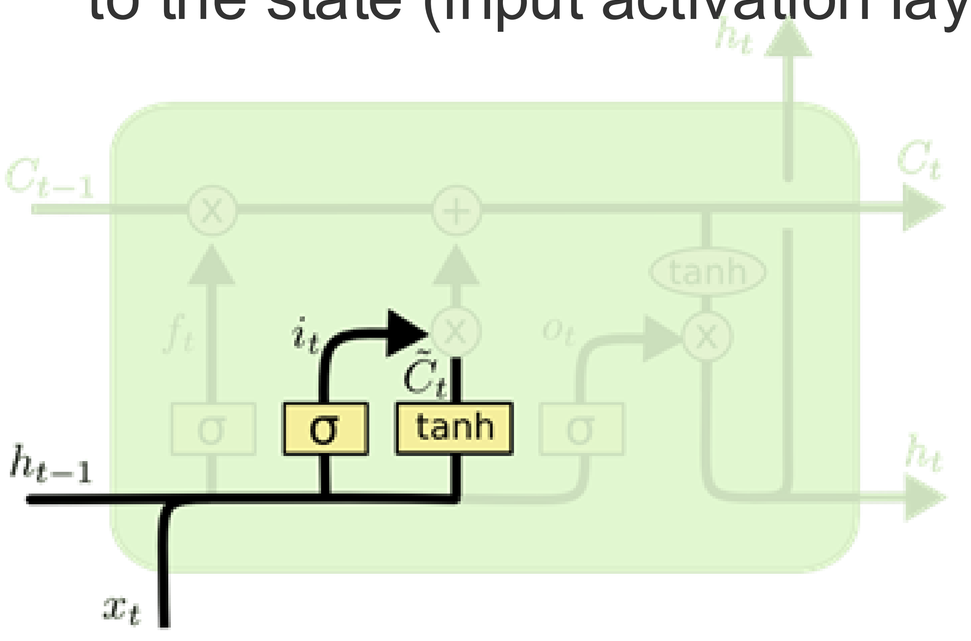
$f_t \approx 0$ → informasi dibuang

menentukan informasi baru apa yg akan disimpan ke dlm cell state

LSTM Unit: Input Gate & Tanh Layer

To decide what new information we're going to store in the cell state.

- Input gate layer decides which values we'll update.
- Tanh layer creates a vector of new candidate values, Ct, that could be added to the state (input activation layer)



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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

0 ≤ ... ≤ 1

-1 ≤ ... ≤ 1

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

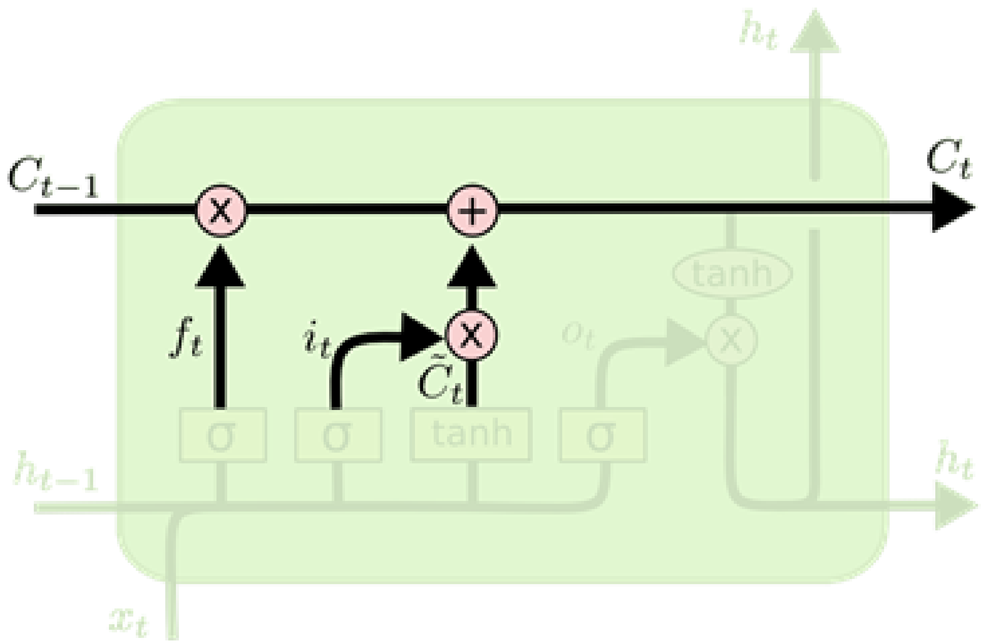
$i_t = \sigma(u_i \cdot x_t + (w_i \cdot h_{t-1} + b_i))$

$\tilde{C}_t = \tanh(u_c \cdot x_t + (w_c \cdot h_{t-1} + b_c))$

CELL STATE

LSTM Unit: Update Ct−1 into Ct

Multiply Ct−1* ft, forgetting the things we decided to forget earlier.
Add it∗Ct, the new candidate values, scaled by how much we decided to update each state value.



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

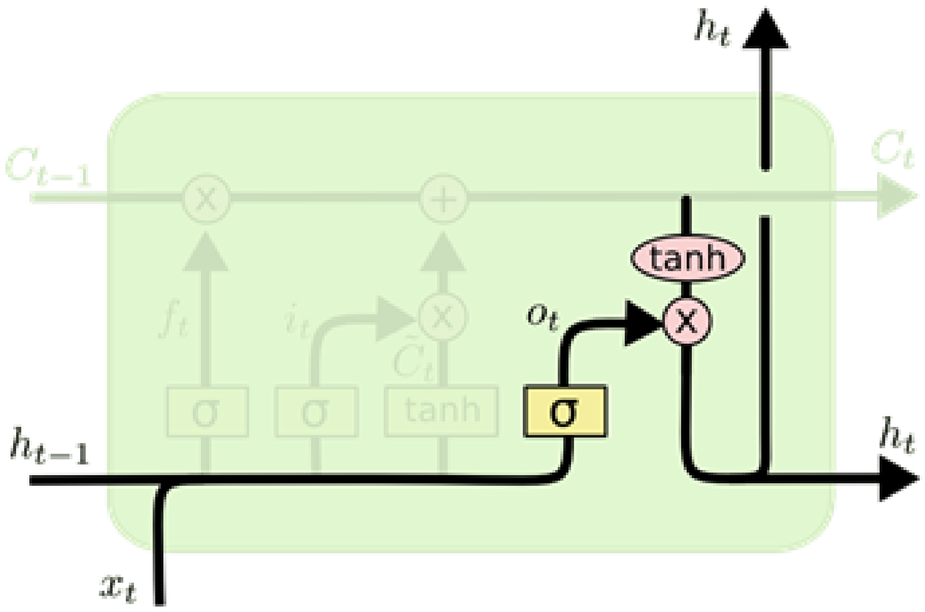
$0 \le \dots \le 1$ (forget gate)
$-1 \le \dots \le 1$ (prev CS)
$0 \le \dots \le 1$ (input gate)
$-1 \le \dots \le 1$ (candidate)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

→ output gate (menentukan nilai ht)

LSTM Unit: Output ht

Output ht will be based on Ct, but will be a filtered version. Run a sigmoid layer which decides what parts of the cell state going to output. Then, put Ct through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



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

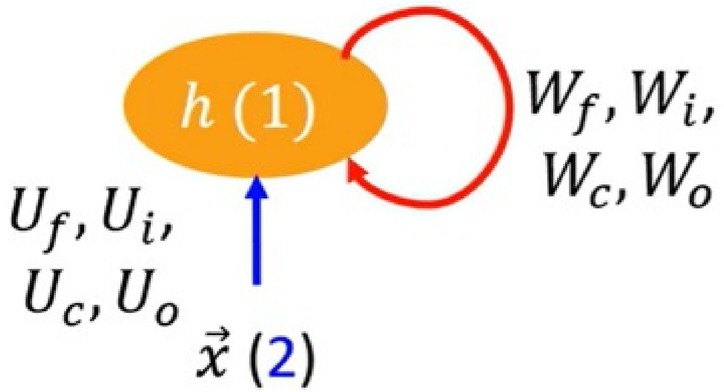
$$h_t = o_t * \tanh\left(C_t\right)$$

$o_t = \sigma(U_o \cdot x_t + (W_o \cdot h_{t-1} + b_o))$

$h_t = o_t * \tanh(C_t)$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Forward Propagation: Example



A1	A2	Target
1	2	0.5
0.5	3	1
…		

Uf	
0.700	0.450

Ui	
0.950	0.800

Uc	
0.450	0.250

Uo	
0.600	0.400

Wf	bf
0.100	0.150

Wi	bi
0.800	0.650

Wc	bc
0.150	0.200

Wo	bo
0.250	0.100

ht-1	Ct-1
0	0

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$
$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$
$$\tilde{C}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

urutan hitung $f_t \rightarrow i_t \rightarrow \tilde{C}_t \rightarrow o_t \rightarrow C_t \rightarrow h_t$

Computing h_t and c_t : Timestep t1

t1=<$\begin{bmatrix}1\\2\end{bmatrix}$, 0.5>

ht-1	Ct-1
0	0

Uf.xt	Wf.ht-1+bf	net_ft	ft
1.600	0.150	1.750	0.852

Ui.xt	Wi.ht-1+bi	net_it	it
2.550	0.650	3.200	0.961

Uc.xt	Wc.ht-1+bc	net_~ct	~ct
0.950	0.200	1.150	0.818

Uo.xt	Wo.ht-1+bo	net_ot	ot
1.400	0.100	1.500	0.818

Ct	ht
0.786	0.536

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

$$\tilde{C}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c)$$

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

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Computing h_t and c_t : Timestep t2

t2=<$\begin{bmatrix}0.5\\3\end{bmatrix}$, 1>

Ct-1	Ht-1
0.786	0.536

Uf.xt	Wf.ht-1+bf	net_ft	ft
1.700	0.204	1.904	0.870

Ui.xt	Wi.ht-1+bi	net_it	it
2.875	1.079	3.954	0.981

Uc.xt	Wc.ht-1+bc	net_~ct	~ct
0.975	0.280	1.255	0.850

Uo.xt	Wo.ht-1+bo	net_ot	ot
1.500	0.234	1.734	0.850

Ct	ht
1.518	0.772

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

$$\tilde{C}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c)$$

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

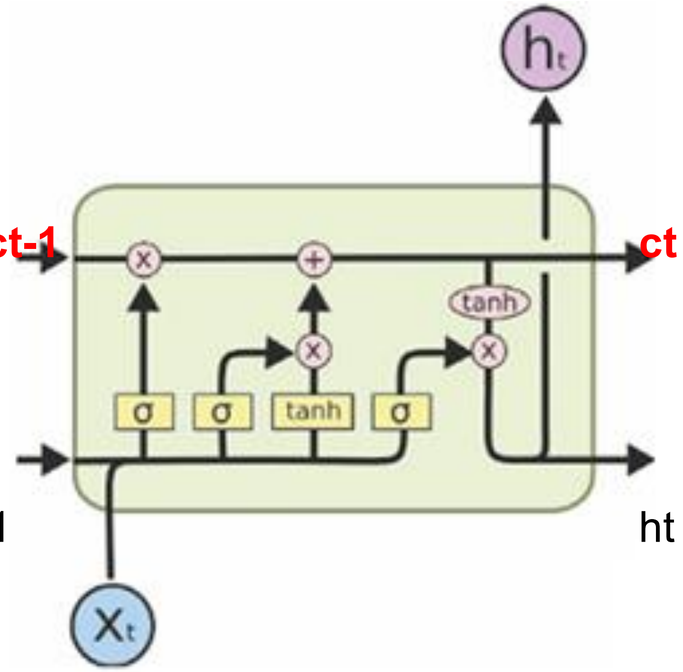
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

Figma

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Contoh Perhitungan: Persamaan Simbol



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

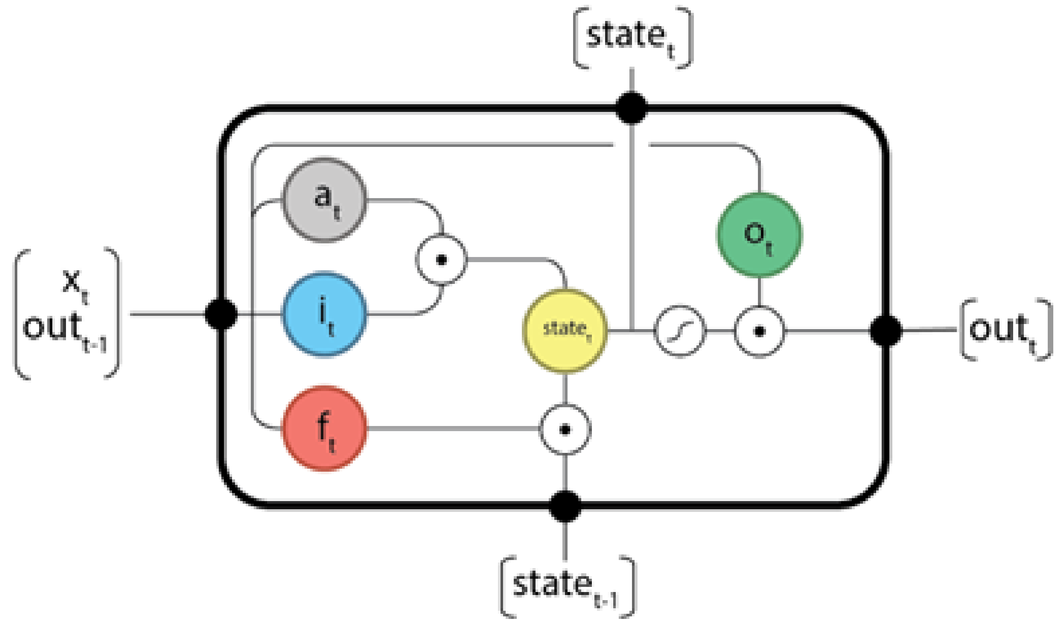
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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

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

$$h_t = o_t * \tanh\left(C_t\right)$$



Input activation:

$$a_t = \tanh(W_a \cdot x_t + U_a \cdot out_{t-1} + b_a)$$

Input gate:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot out_{t-1} + b_i)$$

Forget gate:

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot out_{t-1} + b_f)$$

Output gate:

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot out_{t-1} + b_o)$$

Internal state:

$$state_t = a_t \odot i_t + f_t \odot state_{t-1}$$

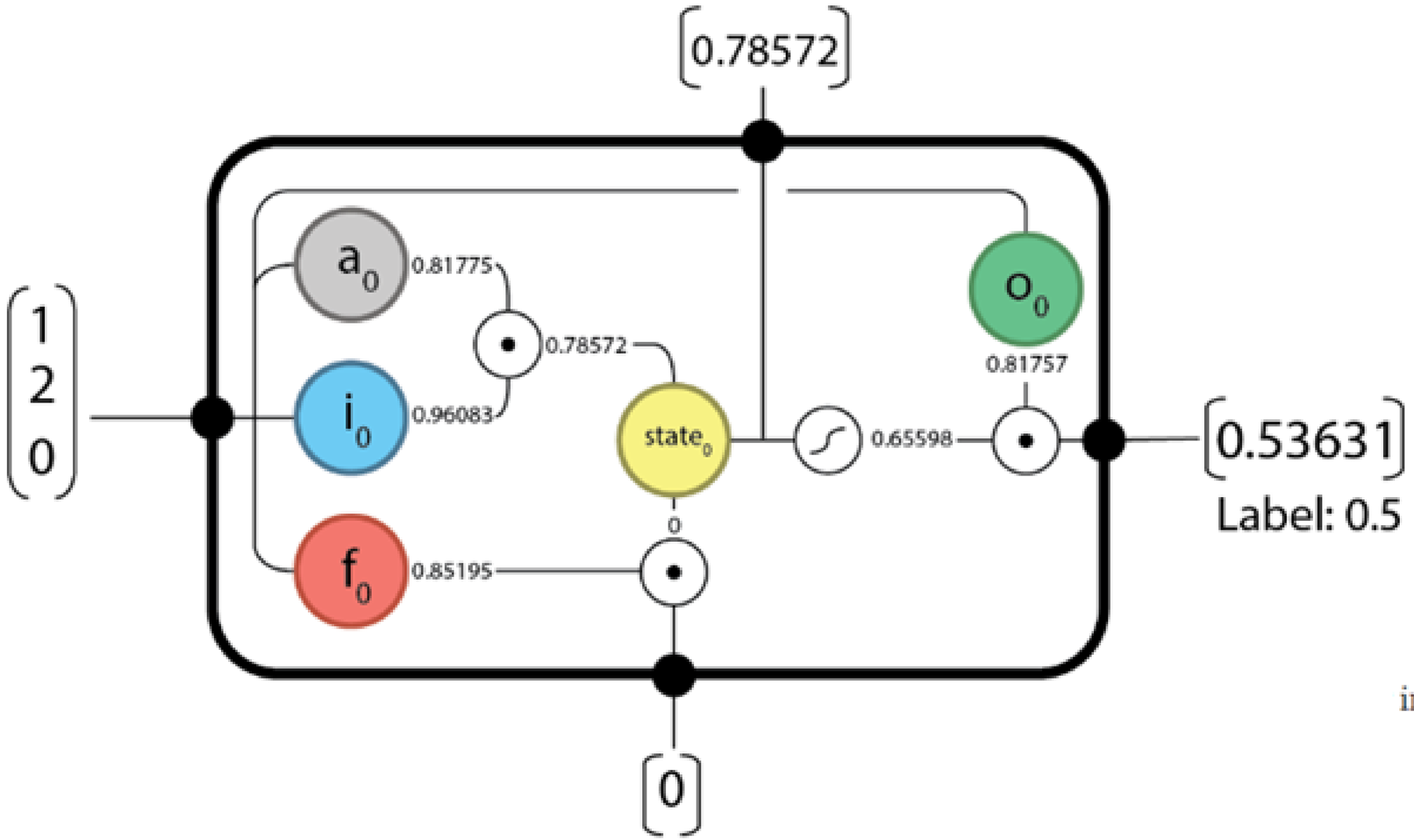
Output:

$$out_t = \tanh(state_t) \odot o_t$$

h_t=out_t

c_t=state_t

Forward phase t=0: data <1,2> dan label 0.5



$$W_a = \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}, U_a = [0.15], b_a = [0.2]$$

$$W_i = \begin{bmatrix} 0.95 \\ 0.8 \end{bmatrix}, U_i = [0.8], b_i = [0.65]$$

$$W_f = \begin{bmatrix} 0.7 \\ 0.45 \end{bmatrix}, U_f = [0.1], b_f = [0.15]$$

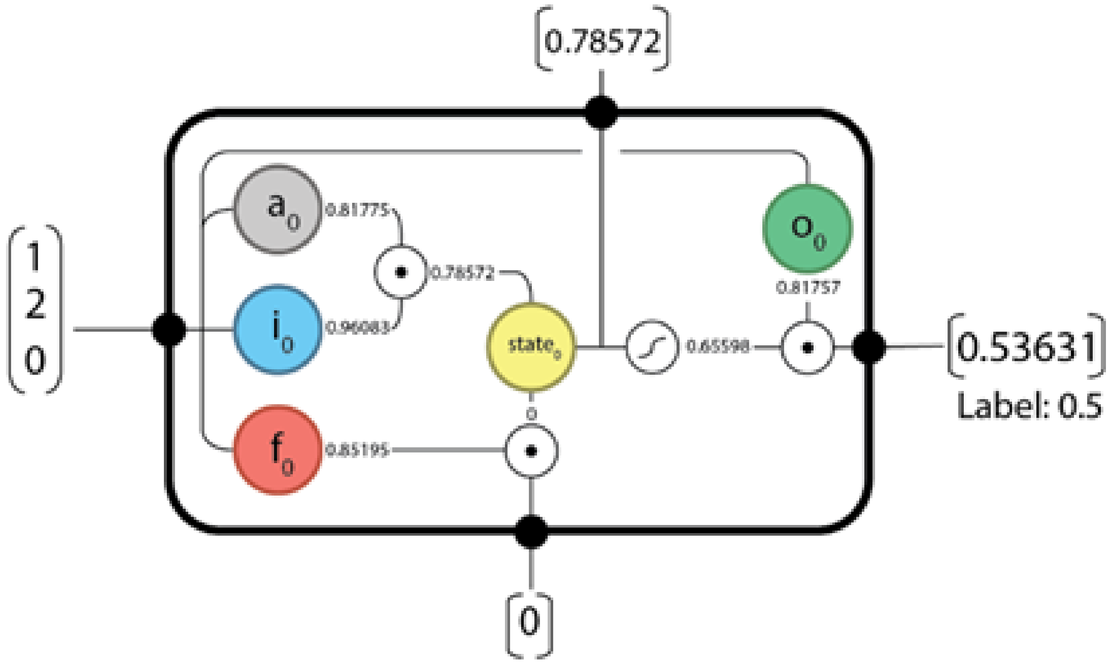
$$W_o = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, U_o = [0.25], b_o = [0.1]$$

input data:

$x_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ with label: 0.5

$x_1 = \begin{bmatrix} 0.5 \\ 3 \end{bmatrix}$ with label: 1.25

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9



https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

$$W_a = \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}, U_a = \begin{bmatrix} 0.15 \end{bmatrix}, b_a = \begin{bmatrix} 0.2 \end{bmatrix}$$

$$W_i = \begin{bmatrix} 0.95 \\ 0.8 \end{bmatrix}, U_i = \begin{bmatrix} 0.8 \end{bmatrix}, b_i = \begin{bmatrix} 0.65 \end{bmatrix}$$

$$W_f = \begin{bmatrix} 0.7 \\ 0.45 \end{bmatrix}, U_f = \begin{bmatrix} 0.1 \end{bmatrix}, b_f = \begin{bmatrix} 0.15 \end{bmatrix}$$

$$W_o = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, U_o = \begin{bmatrix} 0.25 \end{bmatrix}, b_o = \begin{bmatrix} 0.1 \end{bmatrix}$$

$$a_0 = \tanh(W_a \cdot x_0 + U_a \cdot out_{-1} + b_a) = \tanh(\begin{bmatrix} 0.45 & 0.25 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.15 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} + \begin{bmatrix} 0.2 \end{bmatrix}) = 0.81775$$

$$i_0 = \sigma(W_i \cdot x_0 + U_i \cdot out_{-1} + b_i) = \sigma(\begin{bmatrix} 0.95 & 0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.8 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} + \begin{bmatrix} 0.65 \end{bmatrix}) = 0.96083$$

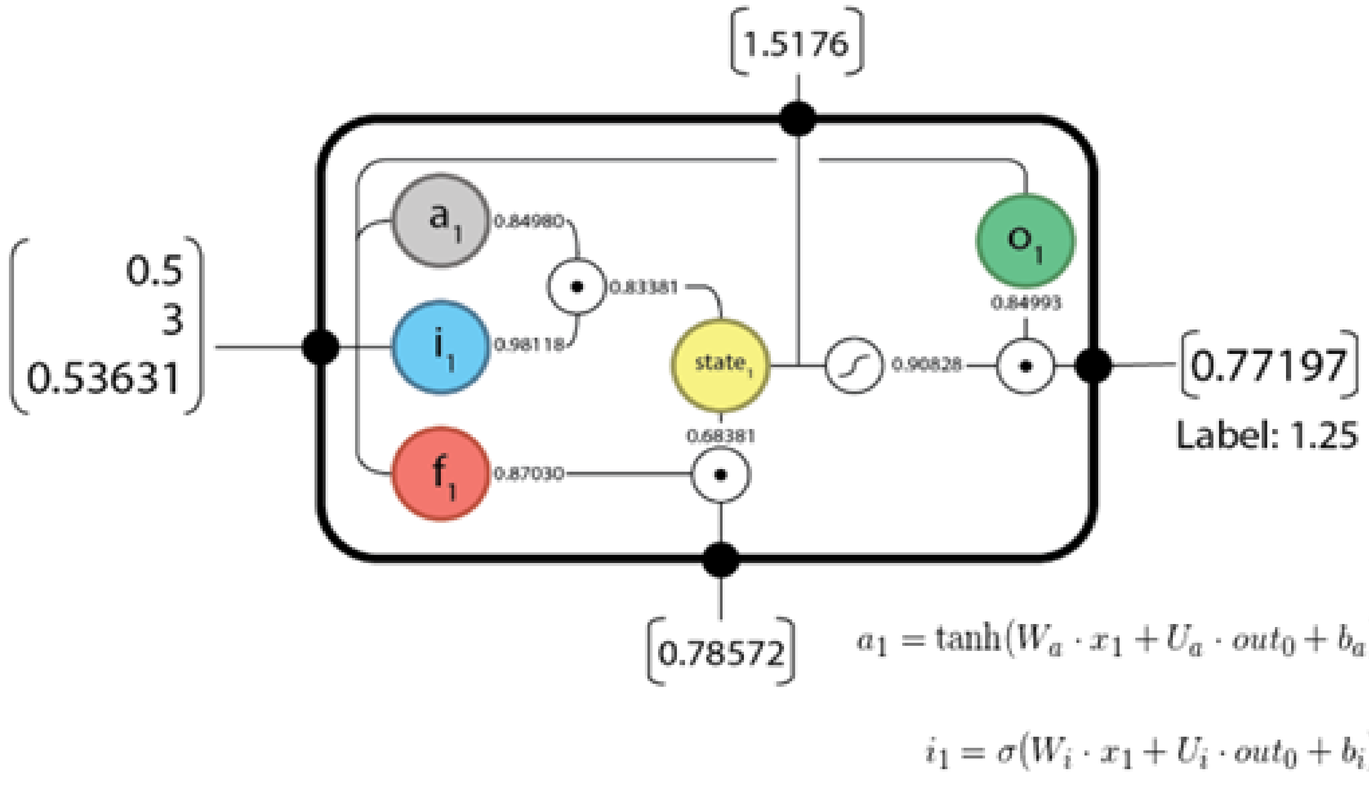
$$f_0 = \sigma(W_f \cdot x_0 + U_f \cdot out_{-1} + b_f) = \sigma(\begin{bmatrix} 0.7 & 0.45 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.1 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} + \begin{bmatrix} 0.15 \end{bmatrix}) = 0.85195$$

$$o_0 = \sigma(W_o \cdot x_0 + U_o \cdot out_{-1} + b_o) = \sigma(\begin{bmatrix} 0.6 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.25 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} + \begin{bmatrix} 0.1 \end{bmatrix}) = 0.81757$$

$$state_0 = a_0 \odot i_0 + f_0 \odot state_{-1} = 0.81775 \times 0.96083 + 0.85195 \times 0 = 0.78572$$

$$out_0 = \tanh(state_0) \odot o_0 = \tanh(0.78572) \times 0.81757 = 0.53631$$

Forward phase t=1: data <0.5,3> dan label 1.25



$$W_a = \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}, U_a = \begin{bmatrix} 0.15 \end{bmatrix}, b_a = \begin{bmatrix} 0.2 \end{bmatrix}$$

$$W_i = \begin{bmatrix} 0.95 \\ 0.8 \end{bmatrix}, U_i = \begin{bmatrix} 0.8 \end{bmatrix}, b_i = \begin{bmatrix} 0.65 \end{bmatrix}$$

$$W_f = \begin{bmatrix} 0.7 \\ 0.45 \end{bmatrix}, U_f = \begin{bmatrix} 0.1 \end{bmatrix}, b_f = \begin{bmatrix} 0.15 \end{bmatrix}$$

$$W_o = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, U_o = \begin{bmatrix} 0.25 \end{bmatrix}, b_o = \begin{bmatrix} 0.1 \end{bmatrix}$$

$$a_1 = \tanh(W_a \cdot x_1 + U_a \cdot out_0 + b_a) = \tanh\left(\begin{bmatrix} 0.45 & 0.25 \end{bmatrix} \begin{bmatrix} 0.5 \\ 3 \end{bmatrix} + \begin{bmatrix} 0.15 \end{bmatrix} \begin{bmatrix} 0.53631 \end{bmatrix} + \begin{bmatrix} 0.2 \end{bmatrix}\right) = 0.84980$$

$$i_1 = \sigma(W_i \cdot x_1 + U_i \cdot out_0 + b_i) = \sigma\left(\begin{bmatrix} 0.95 & 0.8 \end{bmatrix} \begin{bmatrix} 0.5 \\ 3 \end{bmatrix} + \begin{bmatrix} 0.8 \end{bmatrix} \begin{bmatrix} 0.53631 \end{bmatrix} + \begin{bmatrix} 0.65 \end{bmatrix}\right) = 0.98118$$

$$f_1 = \sigma(W_f \cdot x_1 + U_f \cdot out_0 + b_f) = \sigma\left(\begin{bmatrix} 0.7 & 0.45 \end{bmatrix} \begin{bmatrix} 0.5 \\ 3 \end{bmatrix} + \begin{bmatrix} 0.1 \end{bmatrix} \begin{bmatrix} 0.53631 \end{bmatrix} + \begin{bmatrix} 0.15 \end{bmatrix}\right) = 0.87030$$

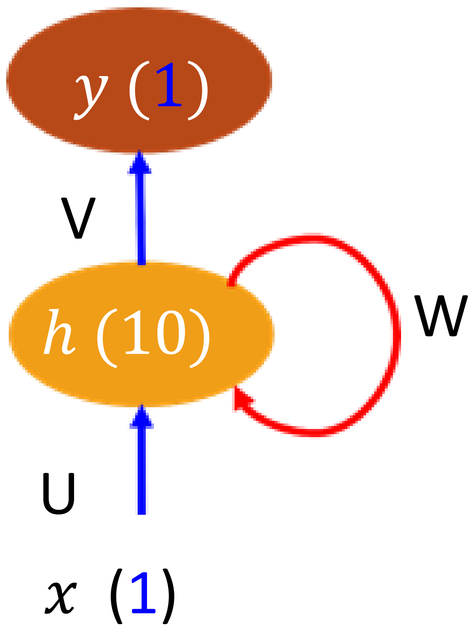
$$o_1 = \sigma(W_o \cdot x_1 + U_o \cdot out_0 + b_o) = \sigma\left(\begin{bmatrix} 0.6 & 0.4 \end{bmatrix} \begin{bmatrix} 0.5 \\ 3 \end{bmatrix} + \begin{bmatrix} 0.25 \end{bmatrix} \begin{bmatrix} 0.53631 \end{bmatrix} + \begin{bmatrix} 0.1 \end{bmatrix}\right) = 0.84993$$

$$state_1 = a_1 \odot i_1 + f_1 \odot state_0 = 0.84980 \times 0.98118 + 0.87030 \times 0.78572 = 1.5176$$

$$out_1 = \tanh(state_1) \odot o_1 = \tanh(1.5176) \times 0.84993 = 0.77197$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Implementing LSTM on Keras: Many to One

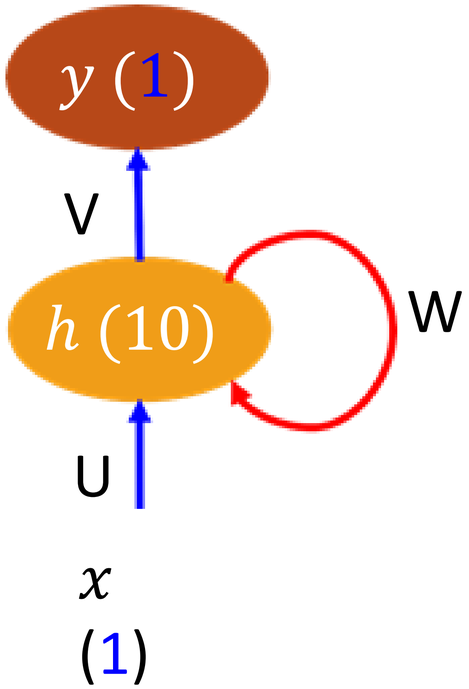



```
# predict amazon stock closing prices, LSTM 50
timestep


from keras import Sequential
from keras.layers import LSTM, Dense
model = Sequential() model.add(LSTM(10,
input_shape=(50,1))) #10 neurons &
process 50x1 sequences
model.add(Dense(1,activation='linear'))
#linear output as regression problem
```

https://towardsdatascience.com/a-comprehensive-guide-to-working-with-recurrent-neural-networks-in-keras-f3b2d5e2fa7f

Number of Parameter




```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10)                480
_________________________________________________________________
dense (Dense)                (None, 1)                 11
=================================================================
Total params: 491
Trainable params: 491
Non-trainable params: 0
```

U = 10 × (1+1) × 4 = 80

W = 10 × 10 × 4 = 400

V = 1 × (10+1) = 11

} 491

Total parameter =
(1+10+1)*4*10+(10+1)*1=491

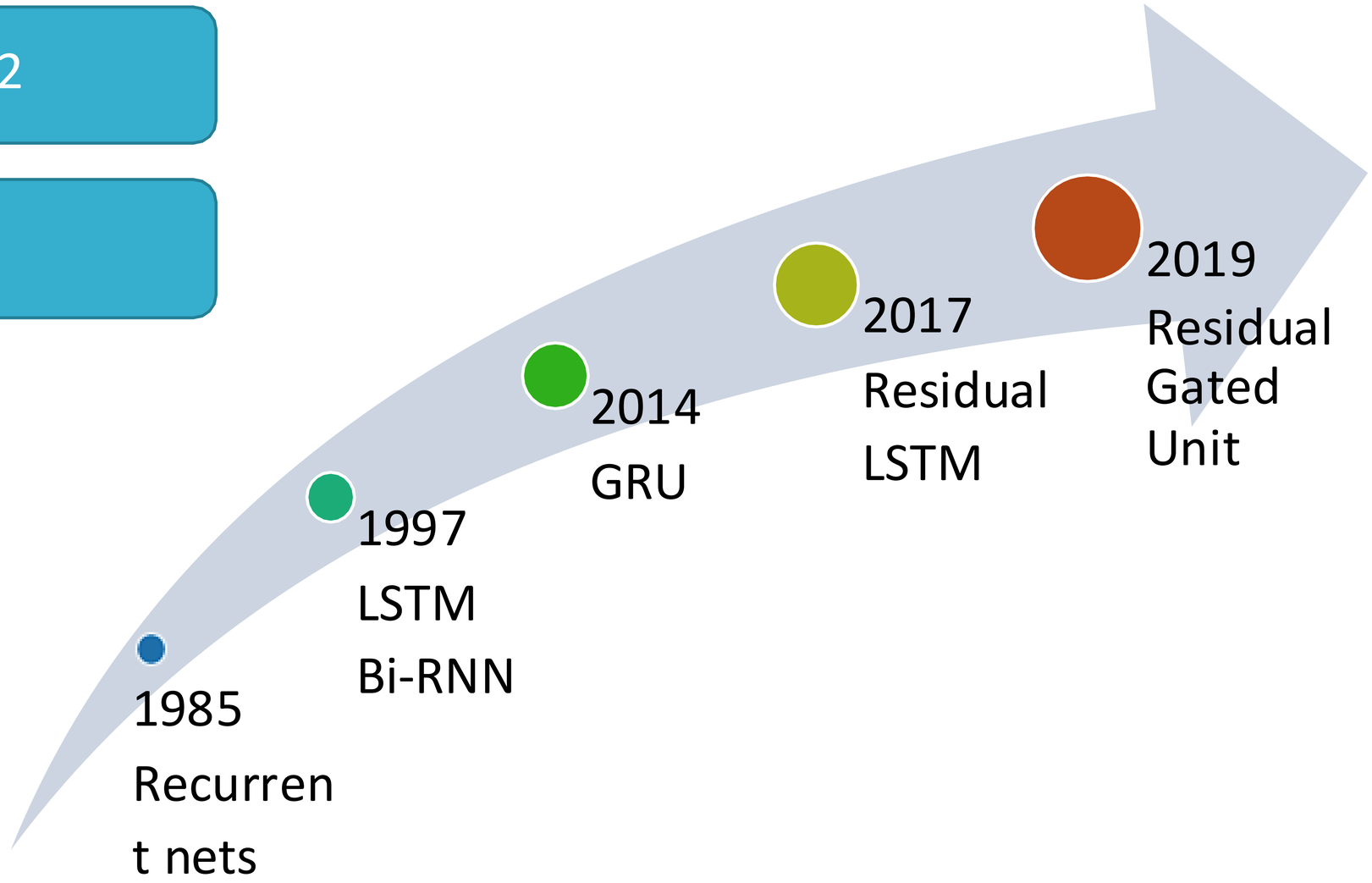
Simple RNN with equal networks: 131 parameter
U: matrix hidden neurons x (input dimension + 1)
W: matrix hidden neurons x hidden neurons
V: matrix output neurons x (hidden neurons+1)

Total parameter for n unit lstm from m- dimension input to k dimension output
= (m+n+1)*4*n+(n+1)*k

RNN → LSTM → GRU → ReGU

GRU: no cell state, 2 gates

ReGU: shortcut connection



Backpropagation Through Time

Backpropagation through time (BPTT)
02

Backpropagation Through Time (BPTT)

BPTT learning algorithm is an extension of standard backpropagation that performs gradients descent on an unfolded network.

Forward Pass

get sequence current output

Backward Pass

compute $\delta gates_t$, δx_t, Δout_{t-1}, δU, δW, δb

Update Weights

$w^{new} = w^{old} - \eta . \delta w^{old}$

dari δU, δW dan δb

1:03 / 9:57

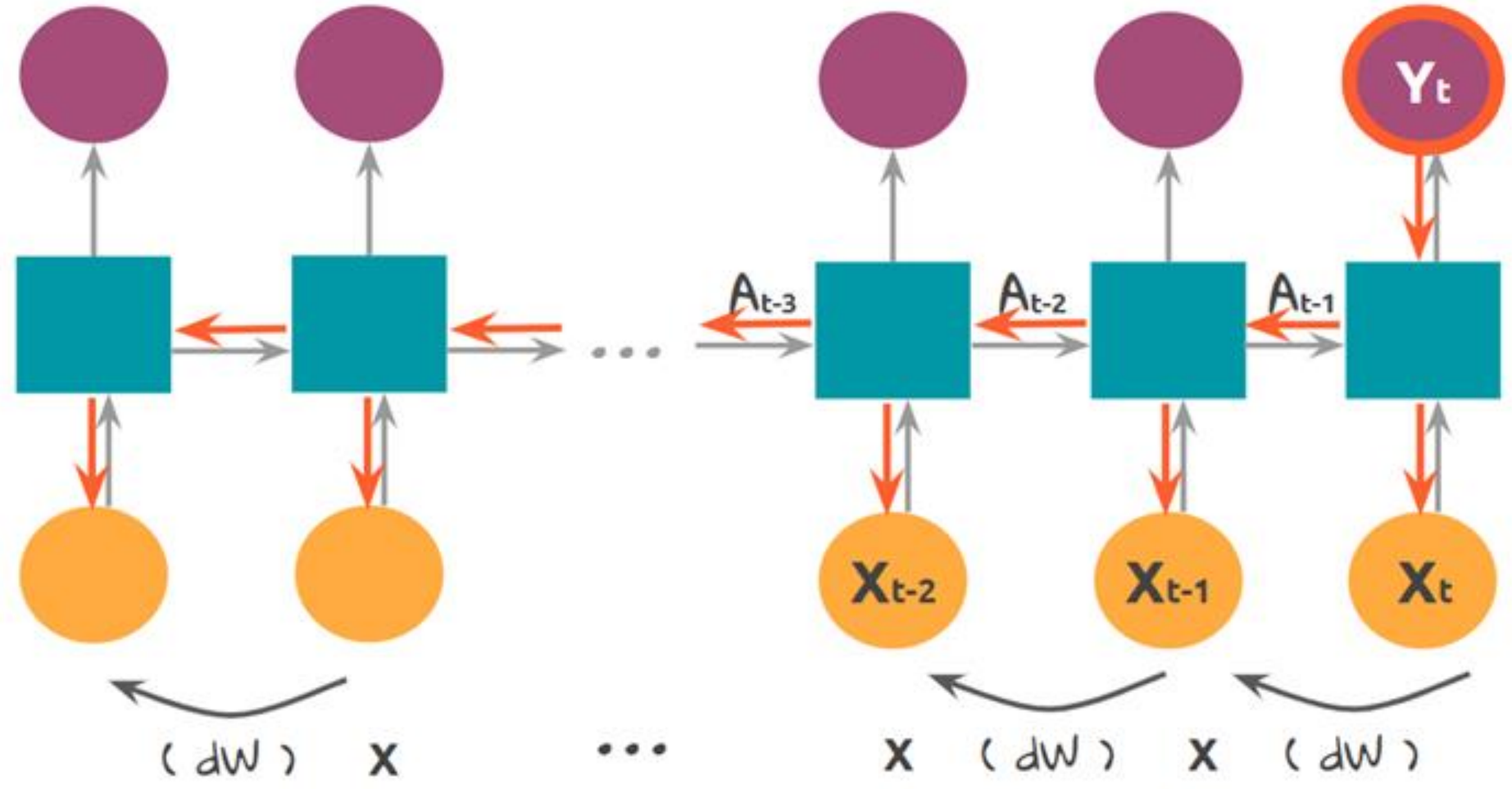
EDUNEX ITB

Backpropagation Through Time (BPTT)

BPTT learning algorithm is an extension of standard backpropagation that performs gradients descent on an unfolded network.

1. Forward pass: get sequence current output
2. Backward pass:
    a. Compute cross entropy error (negative log-likelihood of y) using the sequence current output ($\bar{y}=\bar{y}1..\bar{y}n$) and the sequence actual output ($y=y1..yn$):
    $E(\bar{y},y) = -\sum \bar{y}t \log(yt)$ dengan $Et(\bar{y}t,yt) = -\bar{y}t \log(yt)$
    a. For the unrolled network, the gradient is calculated for each time step with respect to the weight parameter
3. Now that the weight is the same for all the time steps the gradients can be combined together for all time steps. The weights are then updated for both recurrent neuron and the dense layers

Backward Pass



https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-recurrent-neural-network-873c29da73c7

BPTT: Update Weight

fo',fh' : turunan fungsi aktivasi di output/hidden layer

δo(t) : error di output layer

δh(t) : error di hidden layer

$\odot$: Hadamard product

s(t) : hasil fh(ah(t))

$$V(t+1) = V(t) + \alpha \frac{\partial E}{\partial V}$$

$$\frac{\partial E}{\partial V} = s(t) \cdot \delta_o(t)$$

$$\frac{\partial E}{\partial a_o(t)} = \frac{\partial E}{\partial o(t)} \frac{\partial o(t)}{\partial a_o\,(t)} = \delta_o(t) = \big(d(t) - o(t)\big) f_o'\big(a_o(t)\big)$$

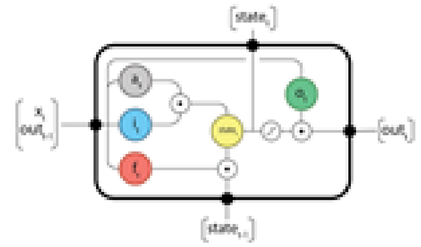
$$U(t+1) = U(t) + \alpha \frac{\partial E}{\partial U}$$

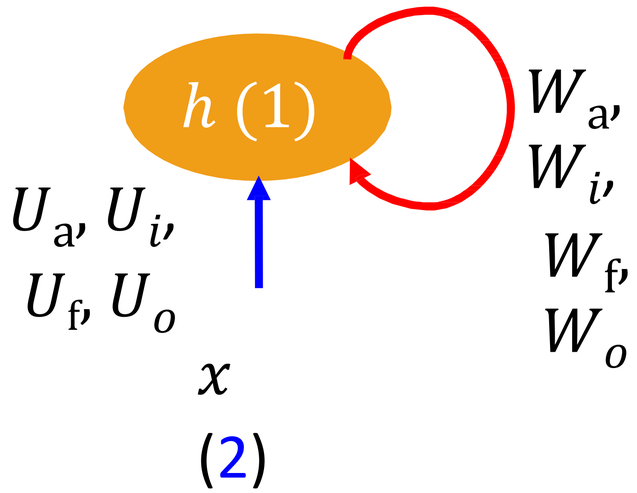
$$\frac{\partial E}{\partial U} = x(t) \cdot \delta_h(t)$$

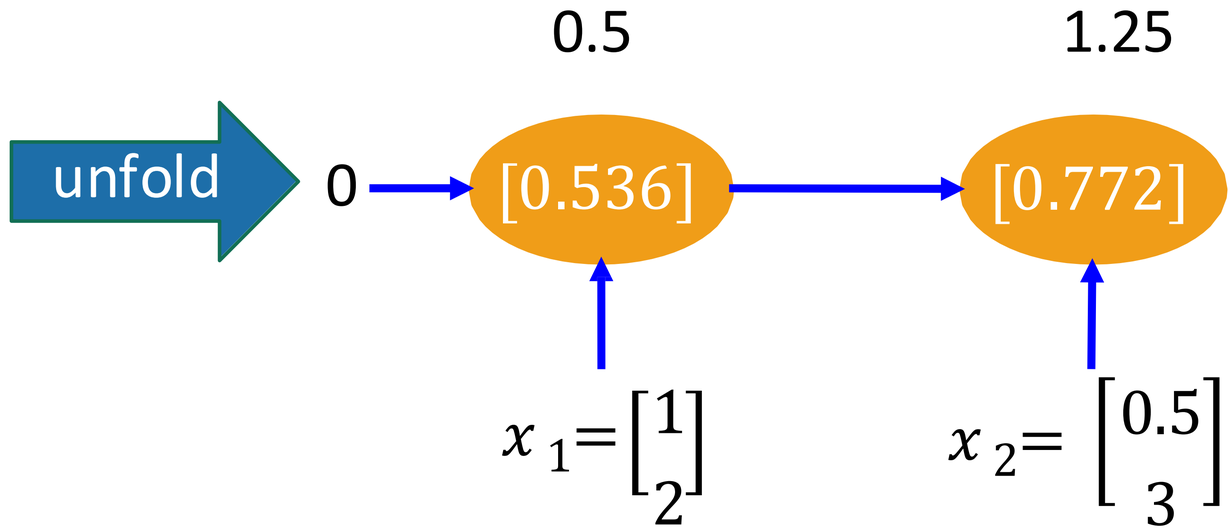
$$\frac{\partial E}{\partial a_h(t)} = \delta_h(t) = f_h'\big(a_h(t)\big) \odot \big(V^T \delta_O(t) + W^T \delta_h(t+1)\big)$$

$$W(t+1) = W(t) + \alpha \frac{\partial E}{\partial W} \qquad \frac{\partial E}{\partial W} = s(t-1) \cdot \delta_h(t)$$

Example: Forward Pass





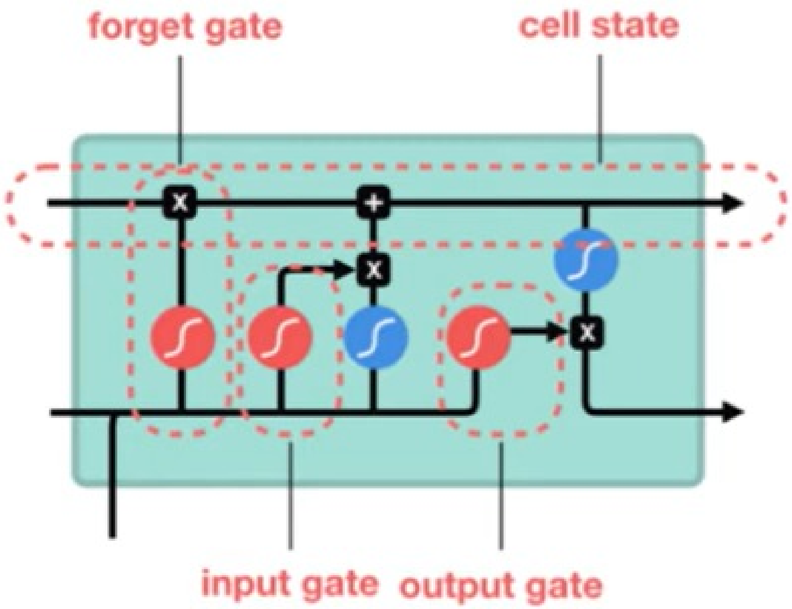


U			
0.45	0.95	0.7	0.6
0.25	0.8	0.45	0.4
W			
0.15	0.8	0.1	0.25

t=1	t=0
a1	**a0**
0.8498	0.81755
i1	**i0**
0.98118	0.96083
f1	**f0**
0.8703	0.85195
o1	**o0**
0.84993	0.81757

t=1	t=0
state1	**state0**
1.5176	0.78752
out1	**out0**
0.77197	0.53631

LSTM: Backward Propagation Timestep t



$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$
$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$
$$\tilde{C}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

FWP

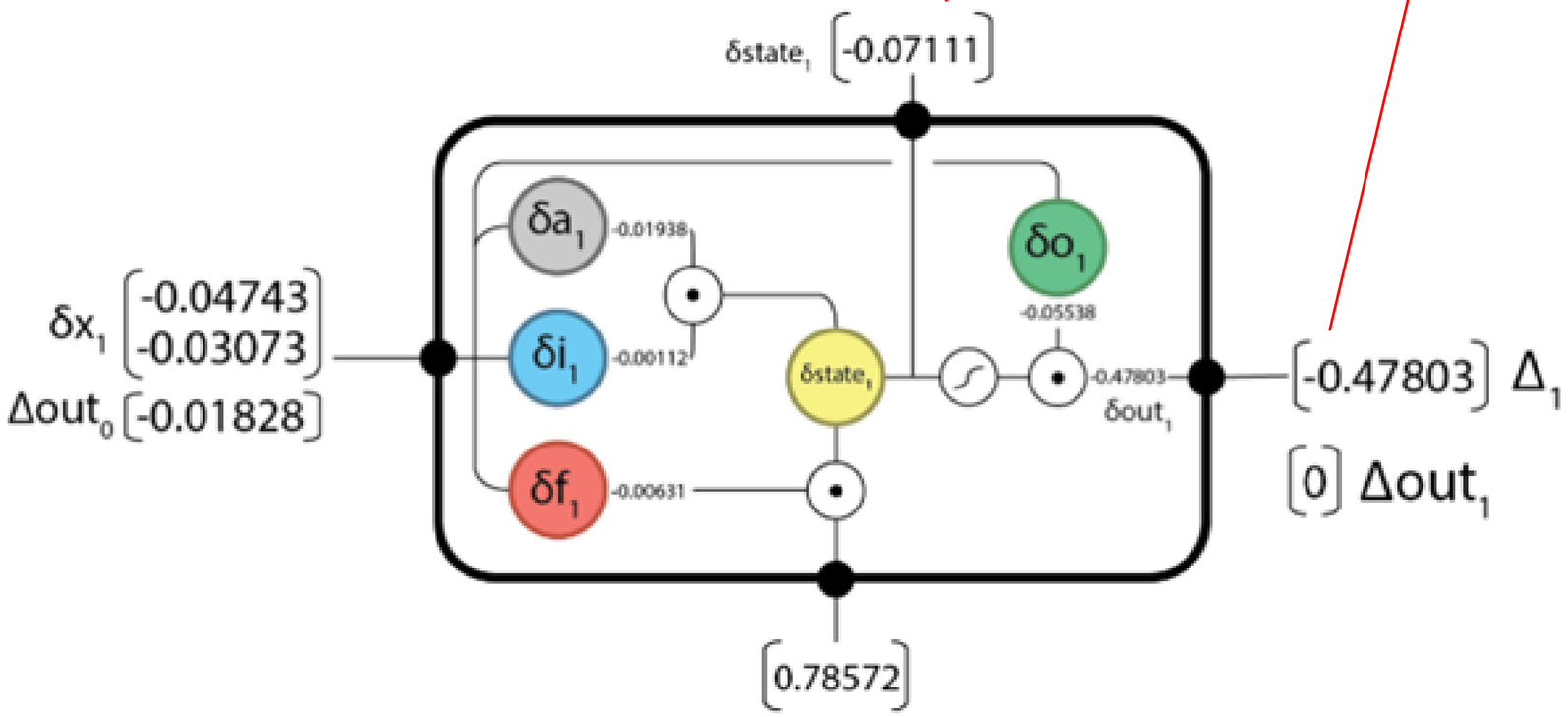
$$\delta out_t = \Delta_t + \Delta out_t$$
$$\delta C_t = \delta out_t \odot o_t \odot \left(1 - tanh^2(C_t)\right) + \delta C_{t+1} \odot f_{t+1}$$
$$\delta \tilde{C}_t = \delta C_t \odot i_t \odot (1 - \tilde{C}_t^{\,2})$$
$$\delta i_t = \delta C_t \odot \tilde{C}_t \odot i_t \odot (1 - i_t)$$
$$\delta f_t = \delta C_t \odot C_{t-1} \odot f_t \odot (1 - f_t)$$
$$\delta o_t = \delta out_t \odot \tanh(C_t) \odot o_t \odot (1 - o_t)$$
$$\delta x_t = U^T . \delta gates_t$$
$$\Delta out_{t-1} = W^T . \delta gates_t$$

BWP

Backward Phase t=1

$$\delta out_t = \Delta_t + \Delta out_t$$

$$\delta state_t = \delta out_t \odot o_t \odot (1 - \tanh^2(state_t)) + \delta state_{t+1} \odot f_{t+1}$$



$$\Delta_1 = \partial_x E = 0.77197 - 1.25 = -0.47803$$

$$\Delta out_1 = 0 \text{ because there are no future time-steps.}$$

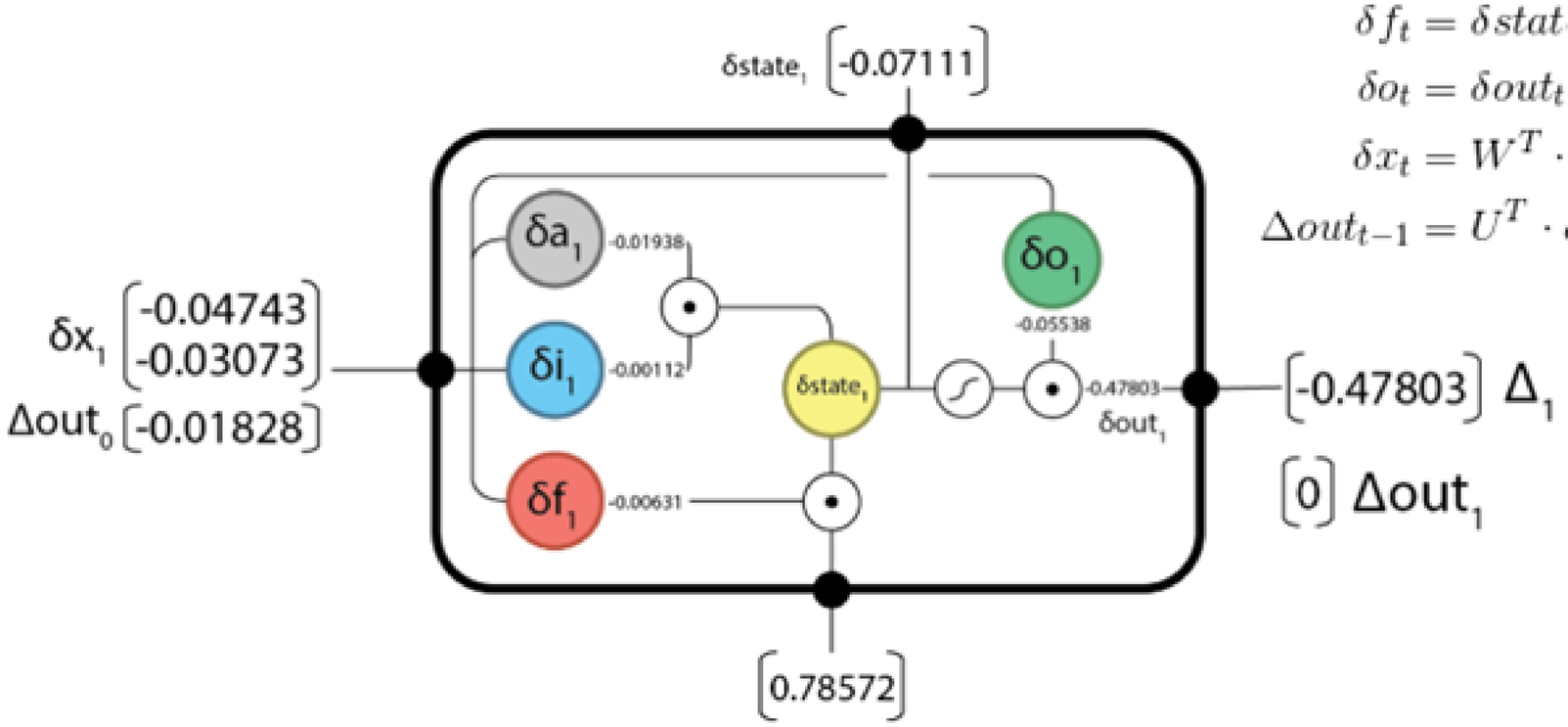
$$\delta out_1 = \Delta_1 + \Delta out_1 = -0.47803 + 0 = -0.47803$$

$$\delta state_1 = \delta out_1 \odot o_1 \odot (1 - \tanh^2(state_1)) + \delta state_2 \odot f_2 = -0.47803 \times 0.84993 \times (1 - \tanh^2(1.5176)) + 0 \times 0 = -0.07111$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Backward Phase t=1 (lanj)

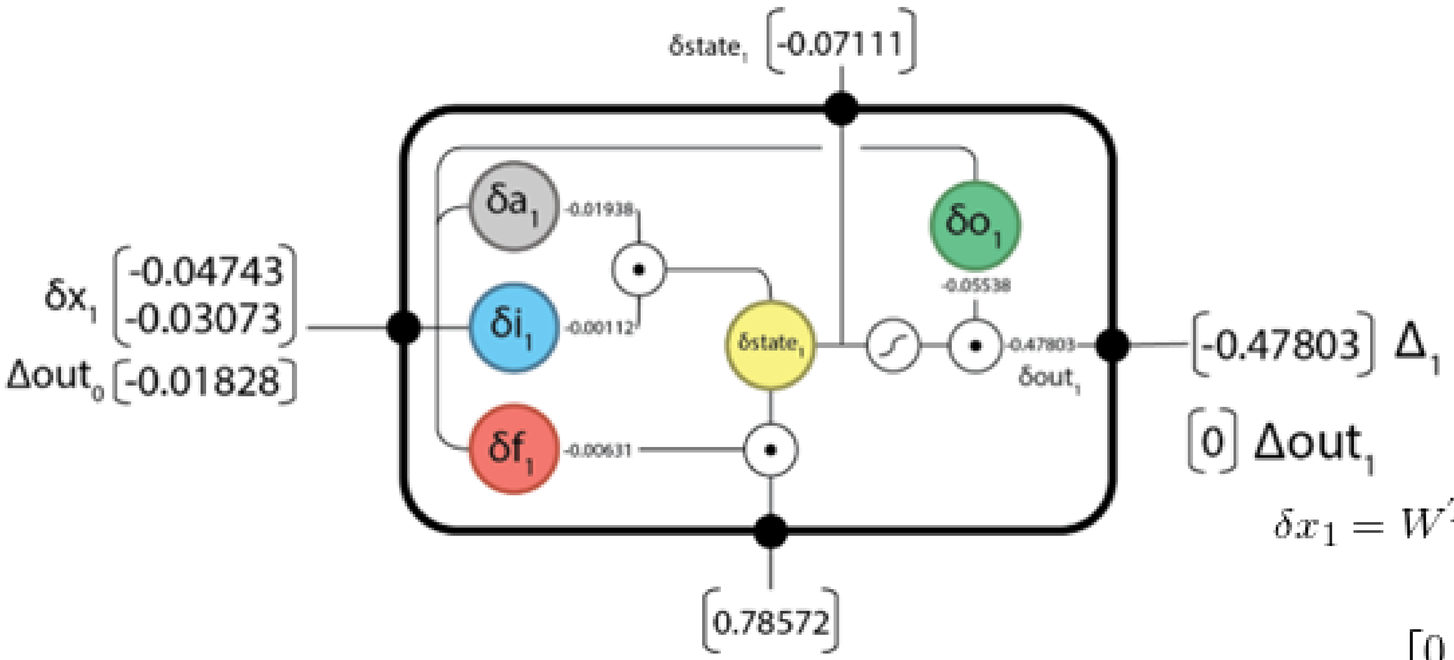
$$\delta a_t = \delta state_t \odot i_t \odot (1 - a_t^2)$$
$$\delta i_t = \delta state_t \odot a_t \odot i_t \odot (1 - i_t)$$
$$\delta f_t = \delta state_t \odot state_{t-1} \odot f_t \odot (1 - f_t)$$
$$\delta o_t = \delta out_t \odot \tanh(state_t) \odot o_t \odot (1 - o_t)$$
$$\delta x_t = W^T \cdot \delta gates_t$$
$$\Delta out_{t-1} = U^T \cdot \delta gates_t$$



$$\delta a_1 = \delta state_1 \odot i_1 \odot (1 - a_1^2) = -0.07111 \times 0.98118 \times (1 - 0.84980^2) = -0.01938$$
$$\delta i_1 = \delta state_1 \odot a_1 \odot i_1 \odot (1 - i_1) = -0.07111 \times 0.84980 \times 0.98118 \times (1 - 0.98118) = -0.00112$$
$$\delta f_1 = \delta state_1 \odot state_0 \odot f_1 \odot (1 - f_1) = -0.07111 \times 0.78572 \times 0.87030 \times (1 - 0.87030) = -0.00631$$
$$\delta o_1 = \delta out_1 \odot \tanh(state_1) \odot o_1 \odot (1 - o_1) = -0.47803 \times \tanh(1.5176) \times 0.84993 \times (1 - 0.84993) = -0.05538$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Backward Phase t=1 (lanj)



$$\delta x_1 = W^T \cdot \delta gates_1$$

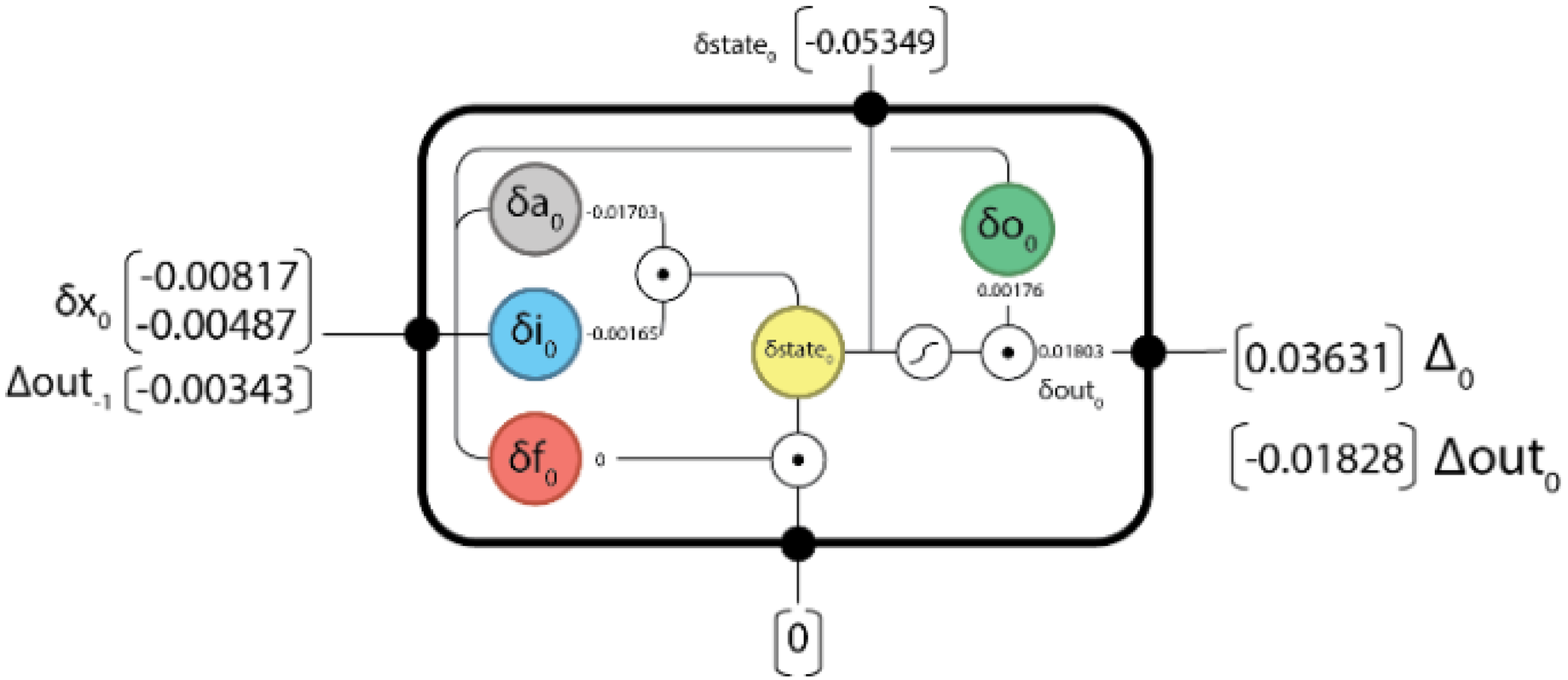
$$= \begin{bmatrix} 0.45 & 0.95 & 0.70 & 0.60 \\ 0.25 & 0.80 & 0.45 & 0.40 \end{bmatrix} \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} = \begin{bmatrix} -0.04743 \\ -0.03073 \end{bmatrix}$$

$$\Delta out_0 = U^T \cdot \delta gates_1$$

$$= \begin{bmatrix} 0.15 & 0.80 & 0.10 & 0.25 \end{bmatrix} \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} = -0.01828$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Backward Phase t=0



https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Backward Phase t=0: Detil

$$\Delta_0 = \partial_x E = 0.53631 - 0.5 = 0.03631$$

$$\Delta out_0 = -0.01828, \text{ passed back from T=1}$$

$$\delta out_0 = \Delta_0 + \Delta out_0 = 0.03631 + -0.01828 = 0.01803$$

$$\delta state_0 = \delta out_0 \odot o_0 \odot (1 - \tanh^2(state_0)) + \delta state_1 \odot f_1 = 0.01803 \times 0.81757 \times (1 - \tanh^2(0.78572)) + -0.07111 \times 0.87030 = -0.05349$$

$$\delta a_0 = \delta state_0 \odot i_0 \odot (1 - a_0^2) = -0.05349 \times 0.96083 \times (1 - 0.81775^2) = -0.01703$$

$$\delta i_0 = \delta state_0 \odot a_0 \odot i_0 \odot (1 - i_0) = -0.05349 \times 0.81775 \times 0.96083 \times (1 - 0.96083) = -0.00165$$

$$\delta f_0 = \delta state_0 \odot state_{-1} \odot f_0 \odot (1 - f_0) = -0.05349 \times 0 \times 0.85195 \times (1 - 0.85195) = 0$$

$$\delta o_0 = \delta out_0 \odot \tanh(state_0) \odot o_0 \odot (1 - o_0) = 0.01803 \times \tanh(0.78572) \times 0.81757 \times (1 - 0.81757) = 0.00176$$

$$\delta x_0 = W^T \cdot \delta gates_0$$

$$= \begin{bmatrix} 0.45 & 0.95 & 0.70 & 0.60 \\ 0.25 & 0.80 & 0.45 & 0.40 \end{bmatrix} \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} = \begin{bmatrix} -0.00817 \\ -0.00487 \end{bmatrix}$$

$$\Delta out_{-1} = U^T \cdot \delta gates_1$$

$$= \begin{bmatrix} 0.15 & 0.80 & 0.10 & 0.25 \end{bmatrix} \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} = -0.00343$$

Update Phase

$$\delta W = \sum_{t=0}^{T} \delta gates_t \otimes x_t$$

$$= \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} \begin{bmatrix} 1.0 & 2.0 \end{bmatrix} + \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} \begin{bmatrix} 0.5 & 3.0 \end{bmatrix} = \begin{bmatrix} -0.02672 & -0.0922 \\ -0.00221 & -0.00666 \\ -0.00316 & -0.01893 \\ -0.02593 & -0.16262 \end{bmatrix}$$

$$\delta U = \sum_{t=0}^{T-1} \delta gates_{t+1} \otimes out_t$$

$$= \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} \begin{bmatrix} 0.53631 \end{bmatrix} = \begin{bmatrix} -0.01039 \\ -0.00060 \\ -0.00338 \\ -0.02970 \end{bmatrix}$$

$$\delta b = \sum_{t=0}^{T} \delta gates_{t+1}$$

$$= \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} + \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} = \begin{bmatrix} -0.03641 \\ -0.00277 \\ -0.00631 \\ -0.05362 \end{bmatrix}$$

Wold

$$W_a = \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}, U_a = \begin{bmatrix} 0.15 \end{bmatrix}, b_a = \begin{bmatrix} 0.2 \end{bmatrix}$$

$$W_i = \begin{bmatrix} 0.95 \\ 0.8 \end{bmatrix}, U_i = \begin{bmatrix} 0.8 \end{bmatrix}, b_i = \begin{bmatrix} 0.65 \end{bmatrix}$$

$$W_f = \begin{bmatrix} 0.7 \\ 0.45 \end{bmatrix}, U_f = \begin{bmatrix} 0.1 \end{bmatrix}, b_f = \begin{bmatrix} 0.15 \end{bmatrix}$$

$$W_o = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, U_o = \begin{bmatrix} 0.25 \end{bmatrix}, b_o = \begin{bmatrix} 0.1 \end{bmatrix}$$

$$W^{new} = W^{old} - \lambda * \delta W^{old}$$

λ=0.1

Wnew

$$W_a = \begin{bmatrix} 0.45267 \\ 0.25922 \end{bmatrix}, U_a = \begin{bmatrix} 0.15104 \end{bmatrix}, b_a = \begin{bmatrix} 0.20364 \end{bmatrix}$$

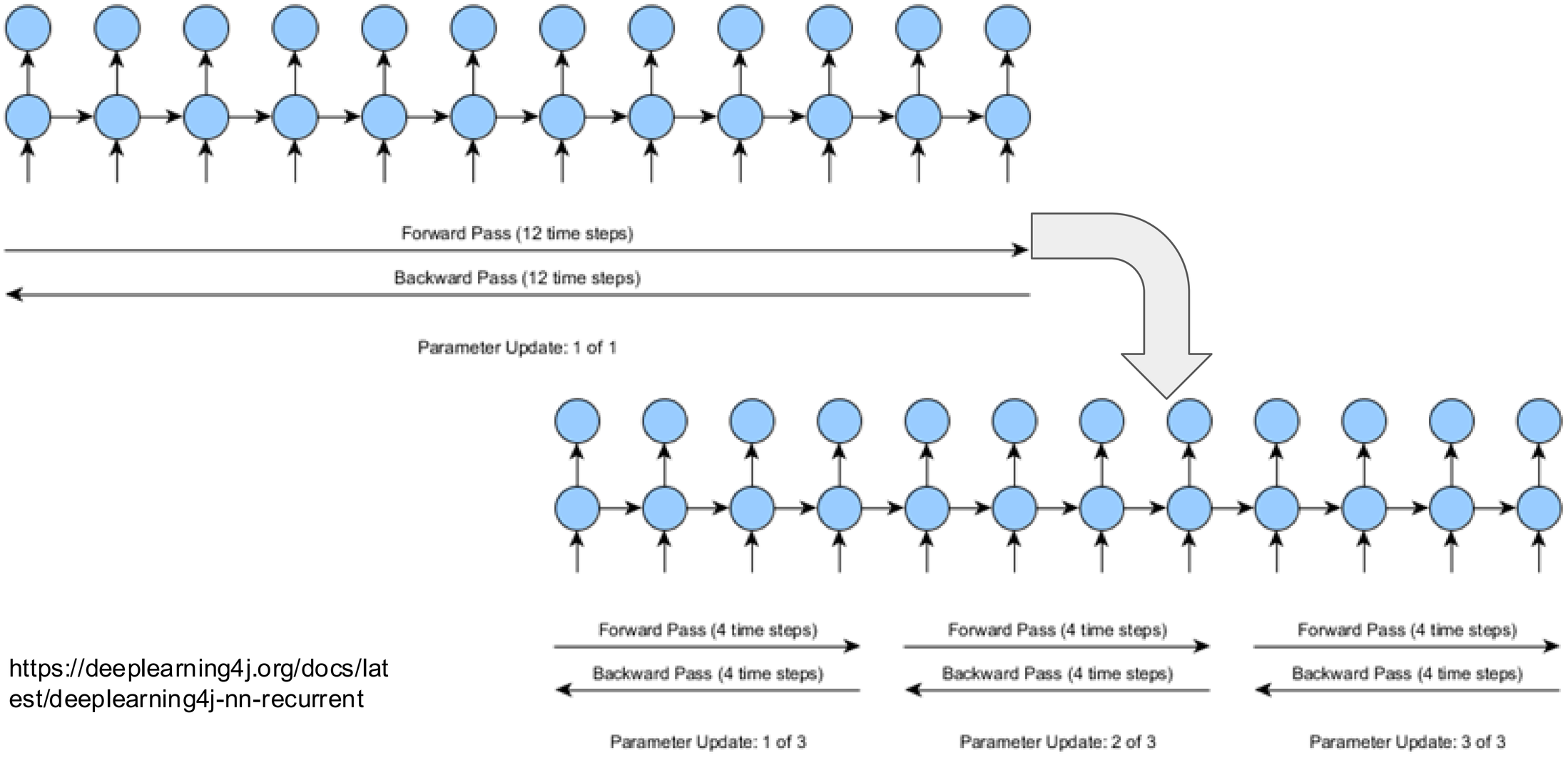
$$W_i = \begin{bmatrix} 0.95022 \\ 0.80067 \end{bmatrix}, U_i = \begin{bmatrix} 0.80006 \end{bmatrix}, b_i = \begin{bmatrix} 0.65028 \end{bmatrix}$$

$$W_f = \begin{bmatrix} 0.70031 \\ 0.45189 \end{bmatrix}, U_f = \begin{bmatrix} 0.10034 \end{bmatrix}, b_f = \begin{bmatrix} 0.15063 \end{bmatrix}$$

$$W_o = \begin{bmatrix} 0.60259 \\ 0.41626 \end{bmatrix}, U_o = \begin{bmatrix} 0.25297 \end{bmatrix}, b_o = \begin{bmatrix} 0.10536 \end{bmatrix}$$

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9

Truncated BPTT: Ilustrasi



https://deeplearning4j.org/docs/latest/deeplearning4j-nn-recurrent

Truncated BPTT

- Truncated BPTT was developed in order to reduce the computational complexity of each parameter update in a recurrent neural network.
- Truncated BPTT splits the forward and backward passes into a set of smaller forward/backward pass operations. The specific length of these forward/backward pass segments is a parameter set by the user.
- With a TBPTT length of 4. Suppose that at time step 10, the network needs to store some information from time step 0 in order to make an accurate prediction. In standard BPTT, this is ok: the gradients can flow backwards all the way along the unrolled network, from time 10 to time 0. In truncated BPTT, this is problematic: the gradients from time step 10 simply don't flow back far enough to cause the required parameter updates that would store the required information. This tradeoff is usually worth it, and (as long as the truncated BPTT lengths are set appropriately), truncated BPTT works well in practice.

Bidirectional RNNs: BPTT

for $t = 1$ to T **do**
  Do forward pass for the forward hidden layer, storing activations at each timestep
for $t = T$ to 1 **do**
  Do forward pass for the backward hidden layer, storing activations at each timestep
for $t = 1$ to T **do**
  Do forward pass for the output layer, using the stored activations from both hidden layers

Algorithm 3.1: BRNN Forward Pass

for $t = T$ to 1 **do**
  Do BPTT backward pass for the output layer only, storing δ terms at each timestep
for $t = T$ to 1 **do**
  Do BPTT backward pass for the forward hidden layer, using the stored δ terms from the output layer
for $t = 1$ to T **do**
  Do BPTT backward pass for the backward hidden layer, using the stored δ terms from the output layer

Algorithm 3.2: BRNN Backward Pass

https://www.cs.toronto.edu/~tingwuwang/rnn_tutorial.pdf

Encoder – Decoder Model

Sequence-to-Sequence Model

- A sequence-to-sequence model aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ.
- Use cases:

1. Machine translation

[English] Mary eats apples. -> [French] Marie mange des pommes.

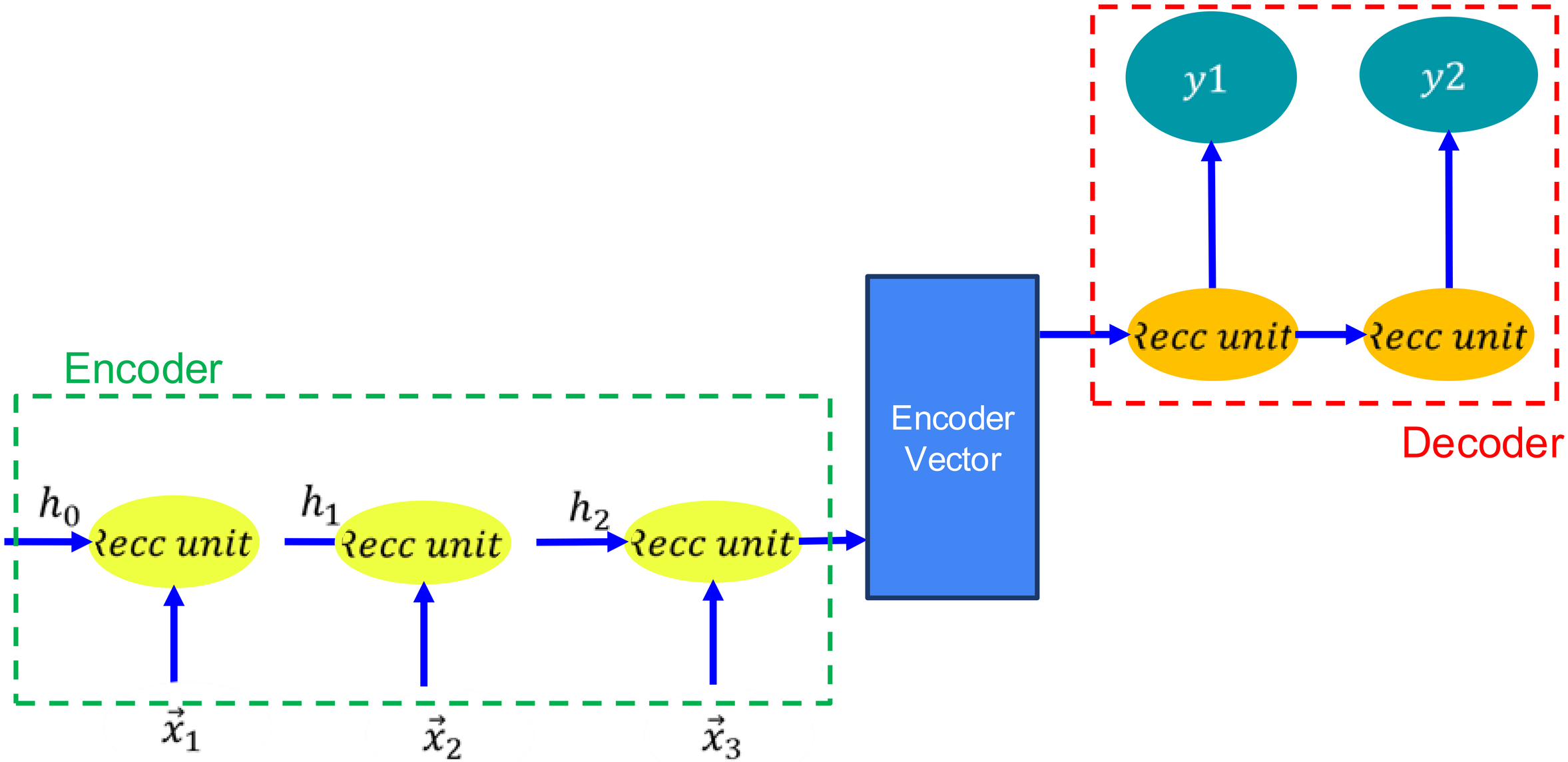
2. Question answering

[Question] Tim is playing in his room.||Where is Tim? -> [Answer] Tim is in his room.

3. Video captioning:

S2VT: A herd of zebras are walking in a field.

RNN Encoder-Decoder



Encoder-Decoder

Encoder

- A stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Encoder Vector

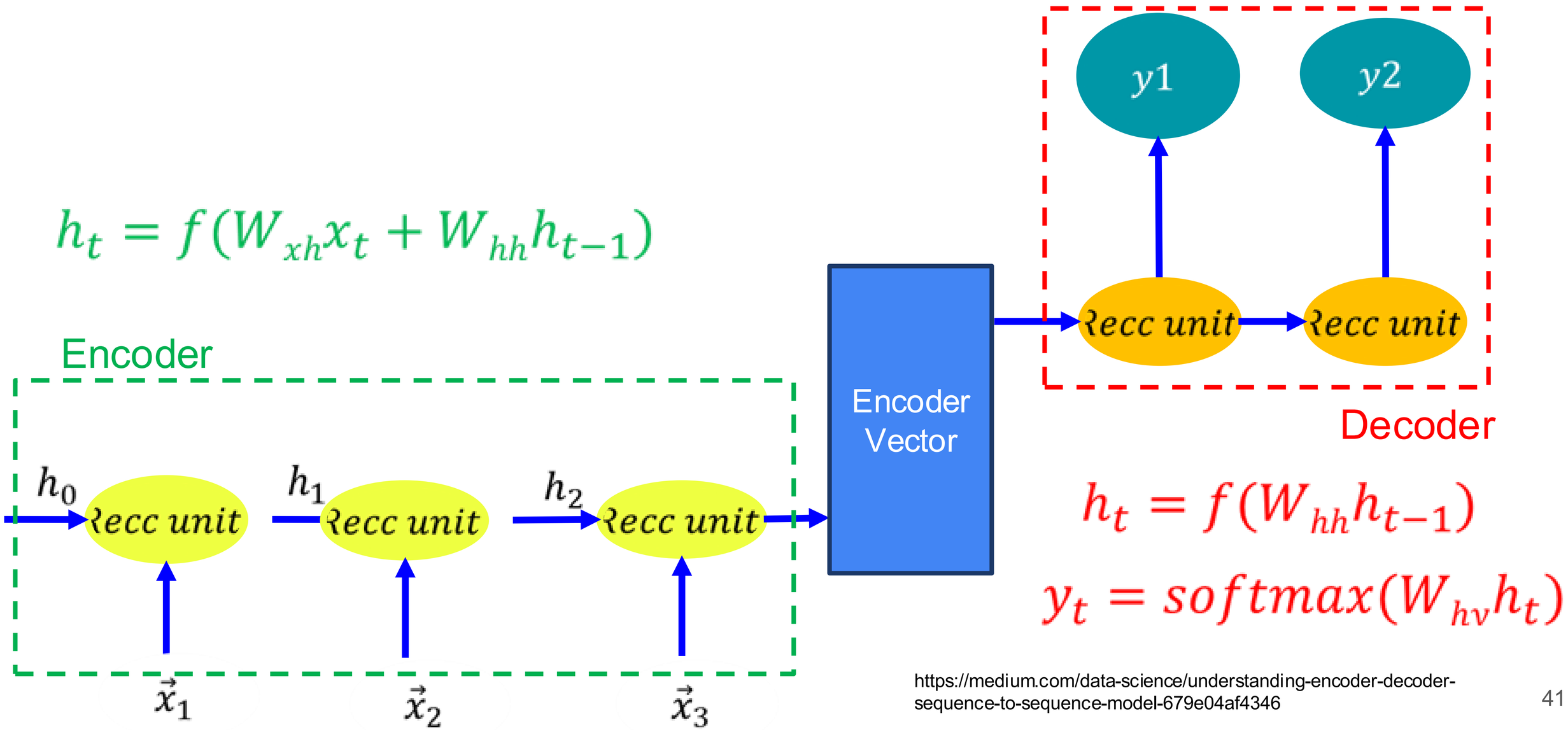
- This is the final hidden state produced from the encoder part of the model. It is calculated using the formula for Encoder.
- This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions.
- It acts as the initial hidden state of the decoder part of the model.

Decoder

- A stack of several recurrent units where each predicts an output yt at a time step t.
- Each recurrent unit accepts a hidden state from the previous unit and produces and output as well as its own hidden state.

RNN Encoder-Decoder

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

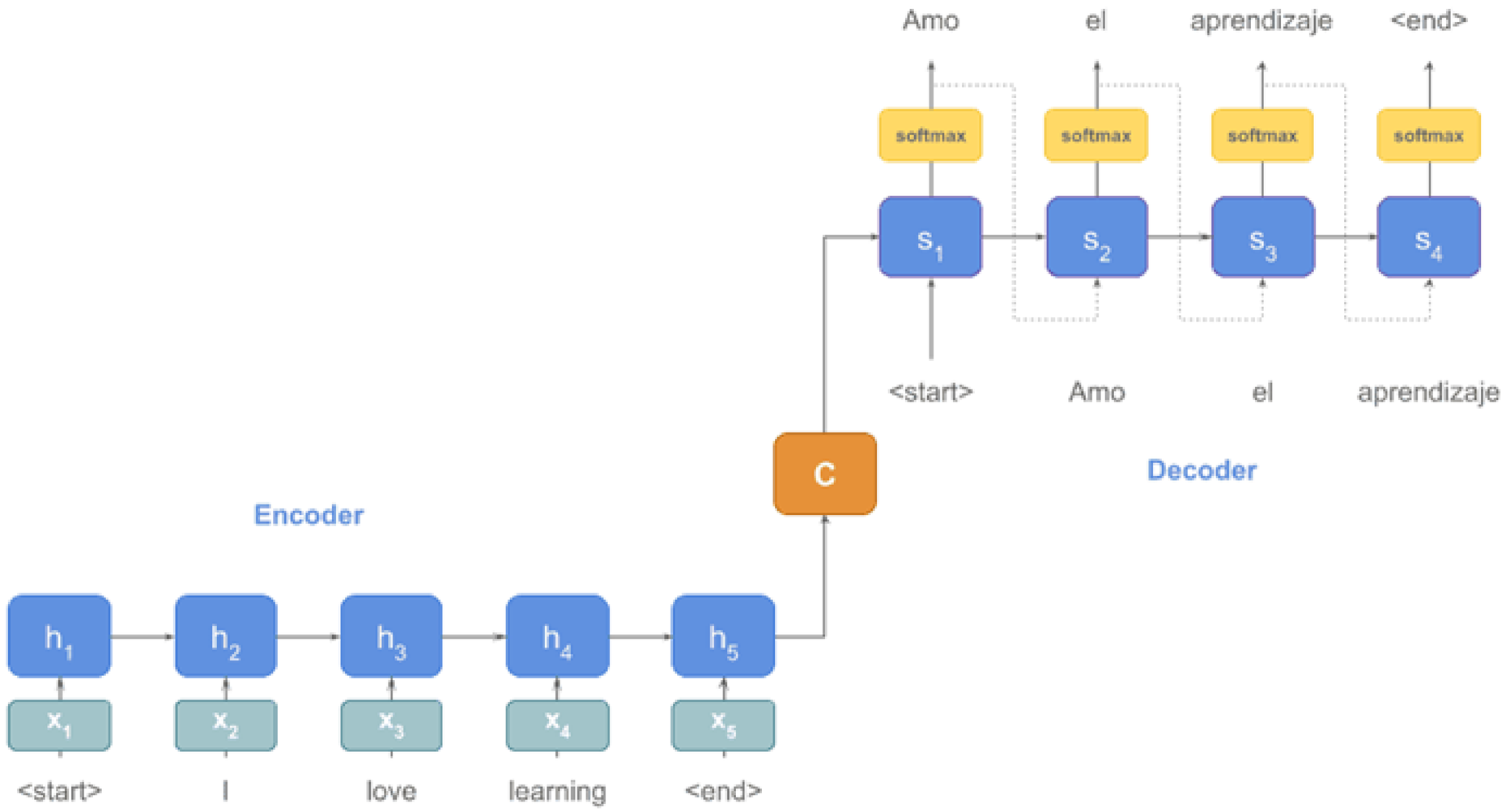


$$h_t = f(W_{hh}h_{t-1})$$

$$y_t = softmax(W_{hv}h_t)$$

https://medium.com/data-science/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346

Example: Machine Translation



Thank you