IF3270 Pembelajaran Mesin

Recurrent Neural Network (RNN)

Tim Pengajar IF3270

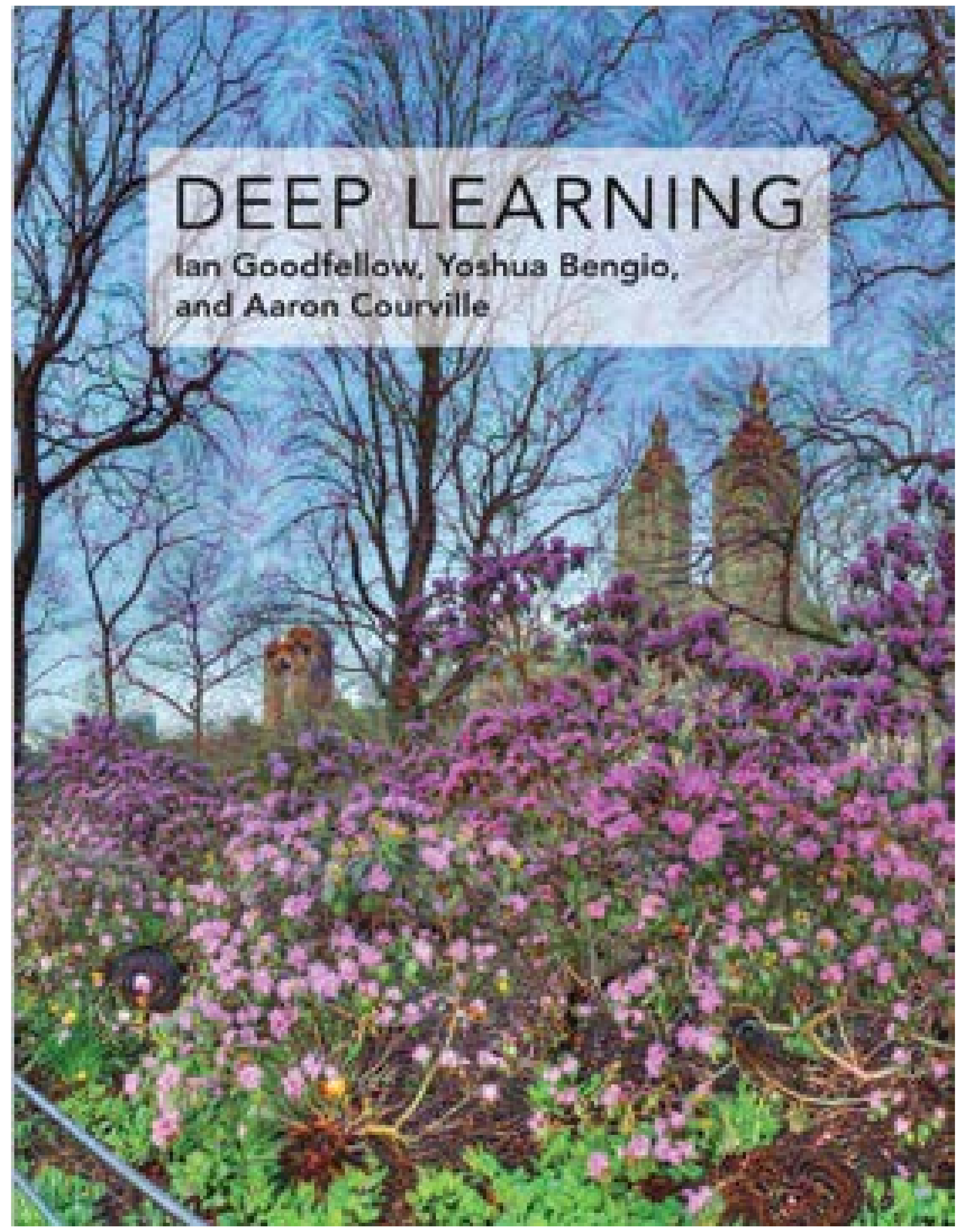
Review

- Machine learning (ML) overview

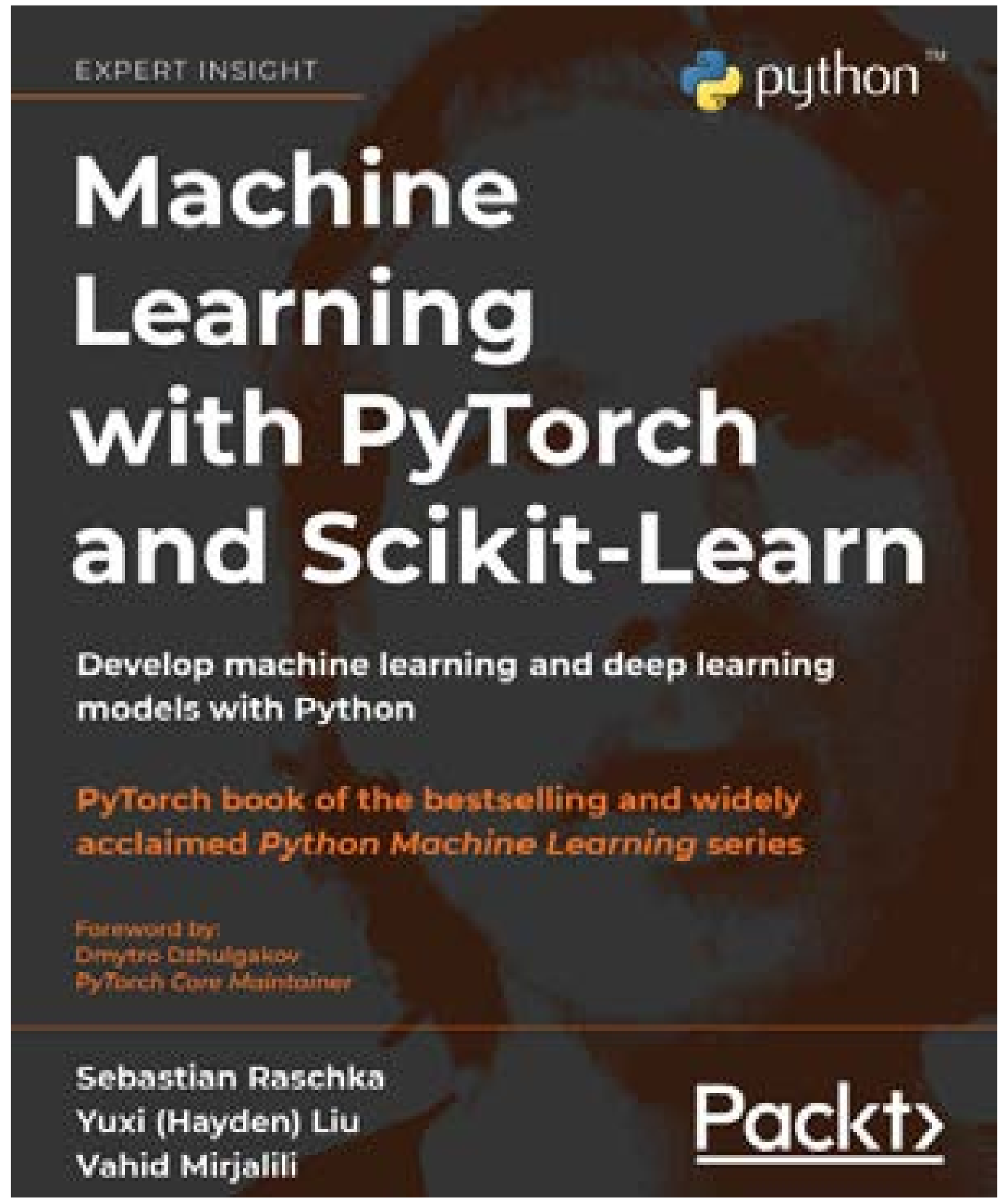
- Ensemble Methods → Supervised Learning

- 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)

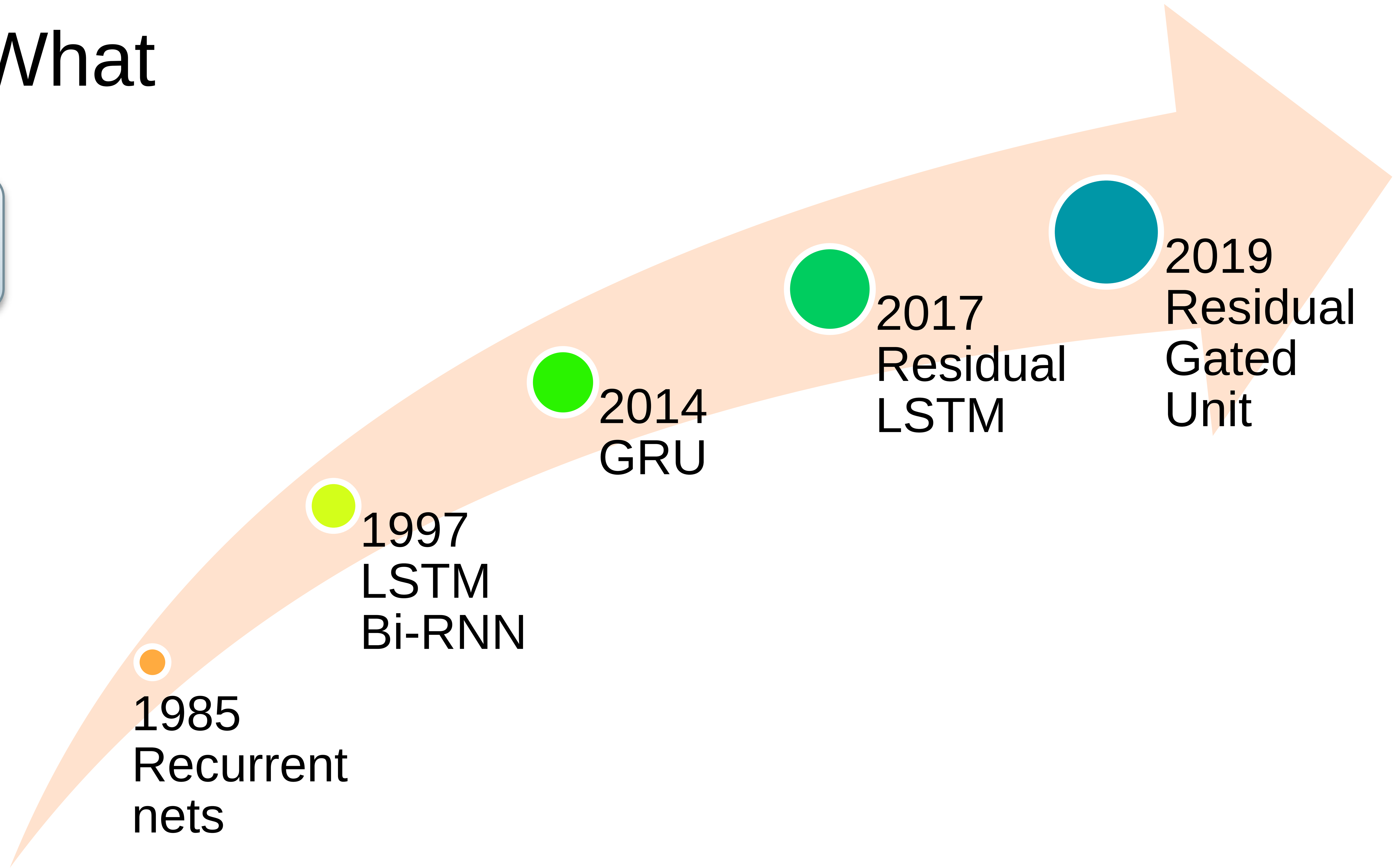
Outline

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

RNN: What & Why

Recurrent NN: What

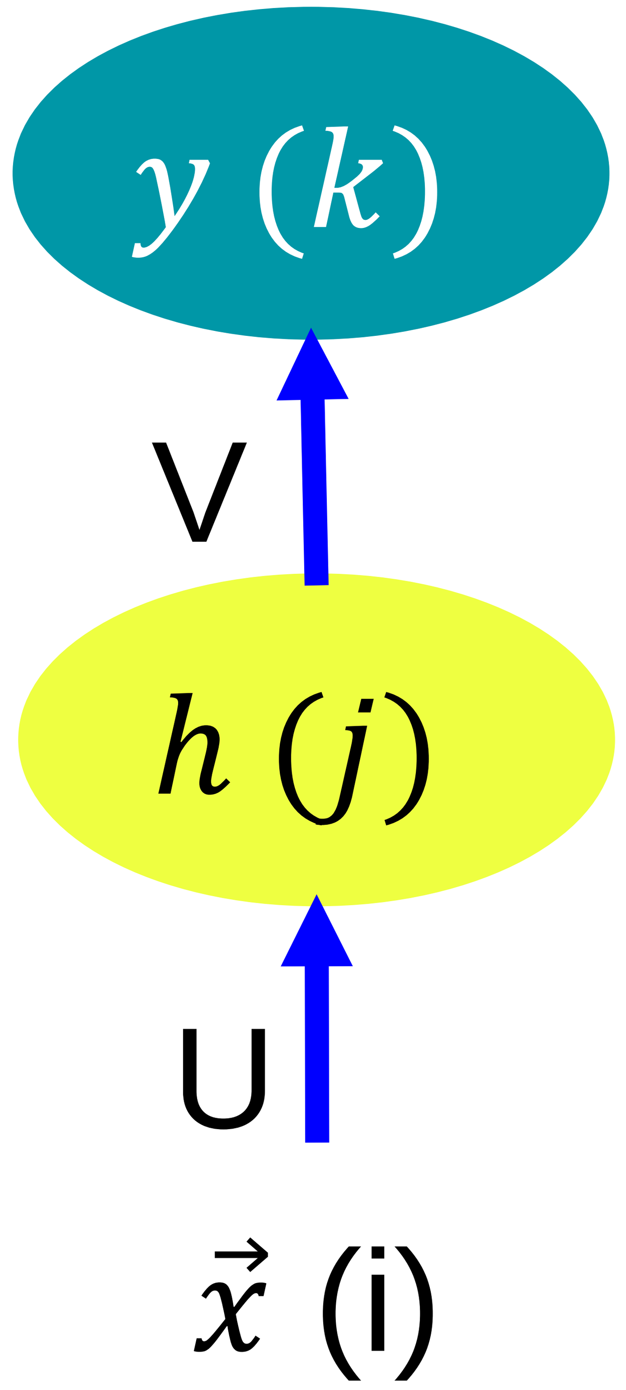
ANN with forward and backward link



Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.
Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*(11), 2673-2681.
Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).
Kim, J., El-Khamy, M., & Lee, J. (2017). Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*.
Luo, H., Li, T., Liu, B., & Zhang, J. (2019). DOER: Dual cross-shared RNN for aspect term-polarity co-extraction. *arXiv preprint arXiv:1906.01794*.

Feedforward vs Feedback Neural Network

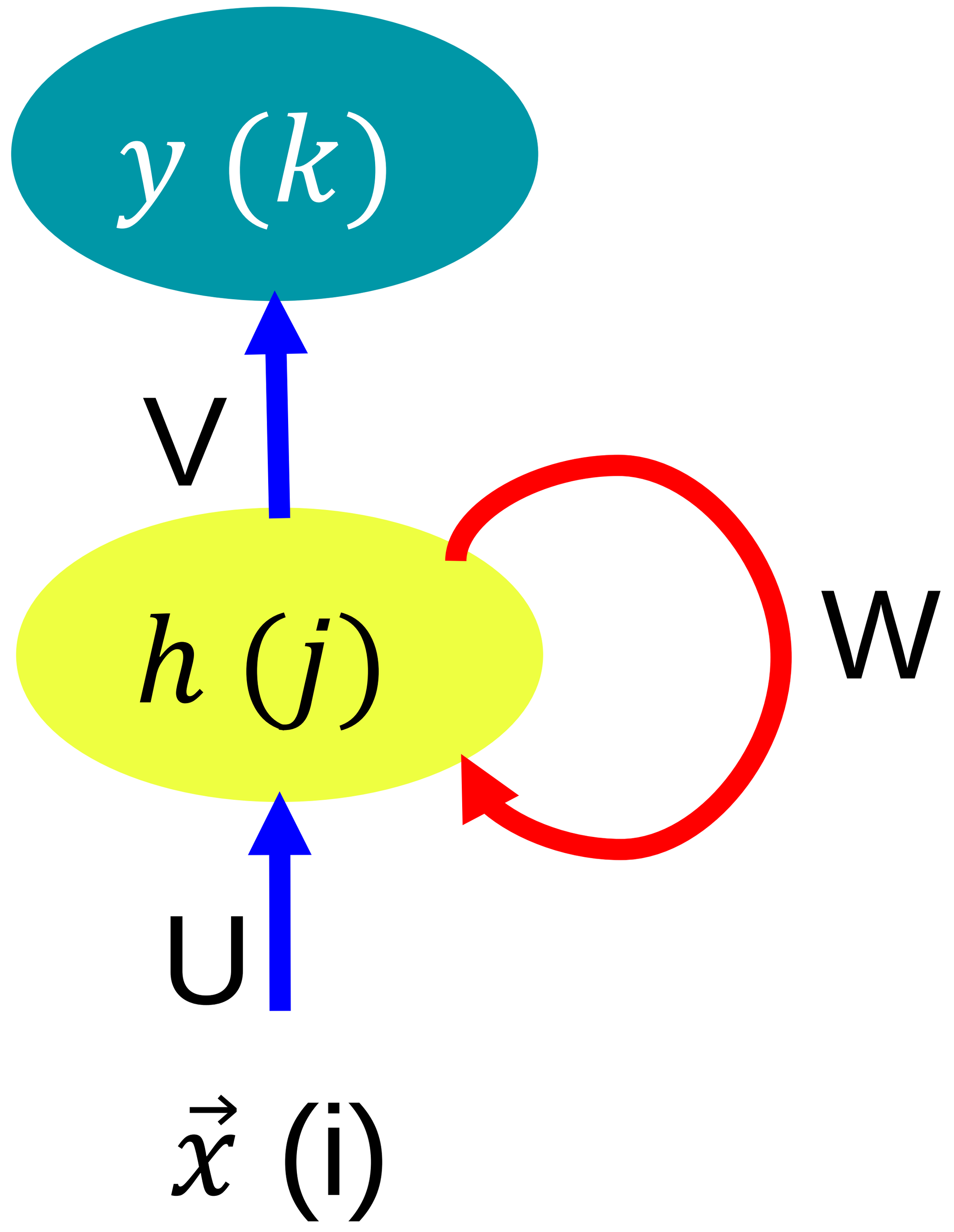
Feedforward NN



Forward link only

there isn't any concept of order in time between the data

Feedback (Recurrent) NN



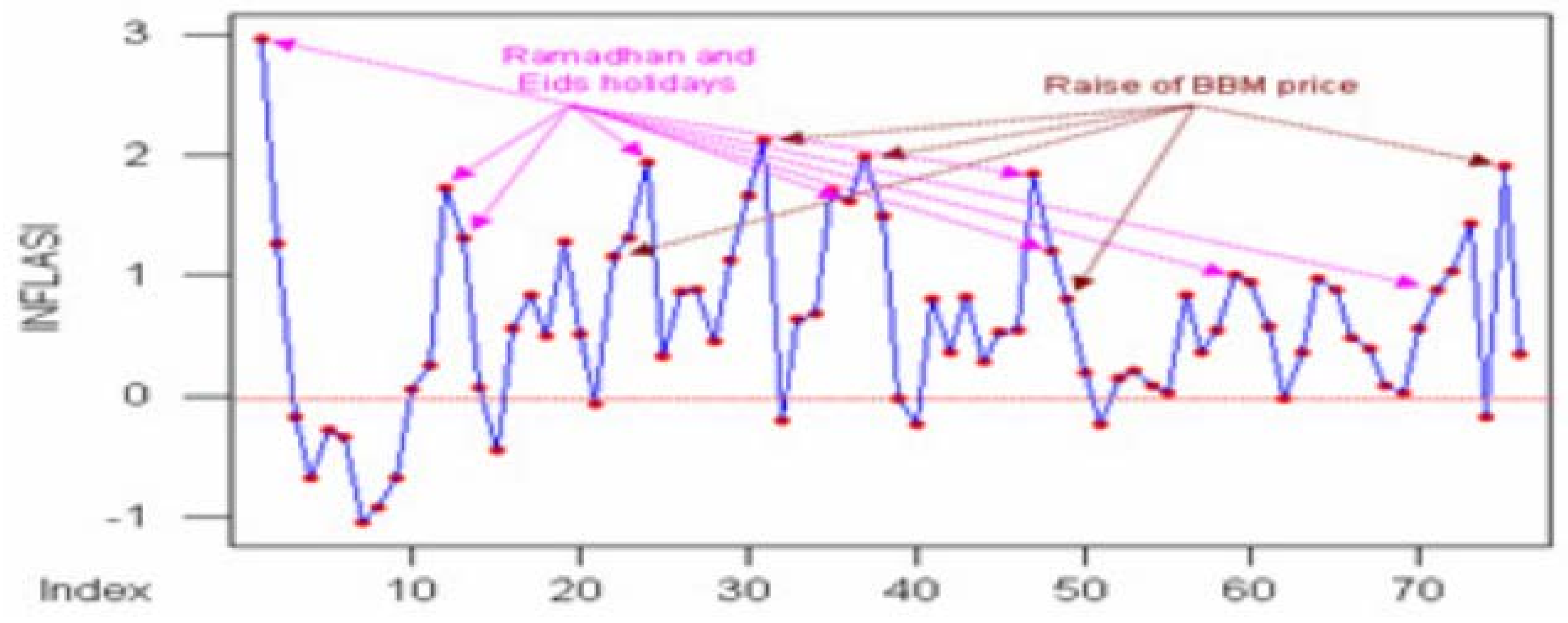
Forward and backward link

Useful topology if order of data matters (sequential data)

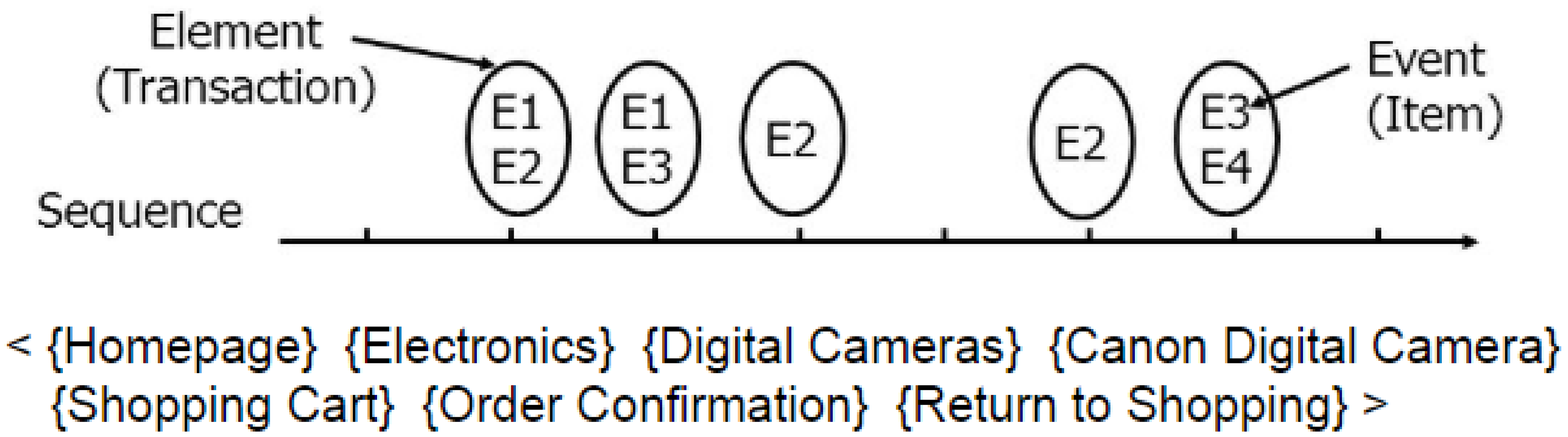
Sharing parameter through sequence / timestep

Why RNN ? Order of data matters

Time-series data
consist of long sequences of data, recorded at equal time intervals



Symbolic sequence data
consist of long sequences of event or nominal data, which typically are not observed at equal time intervals.



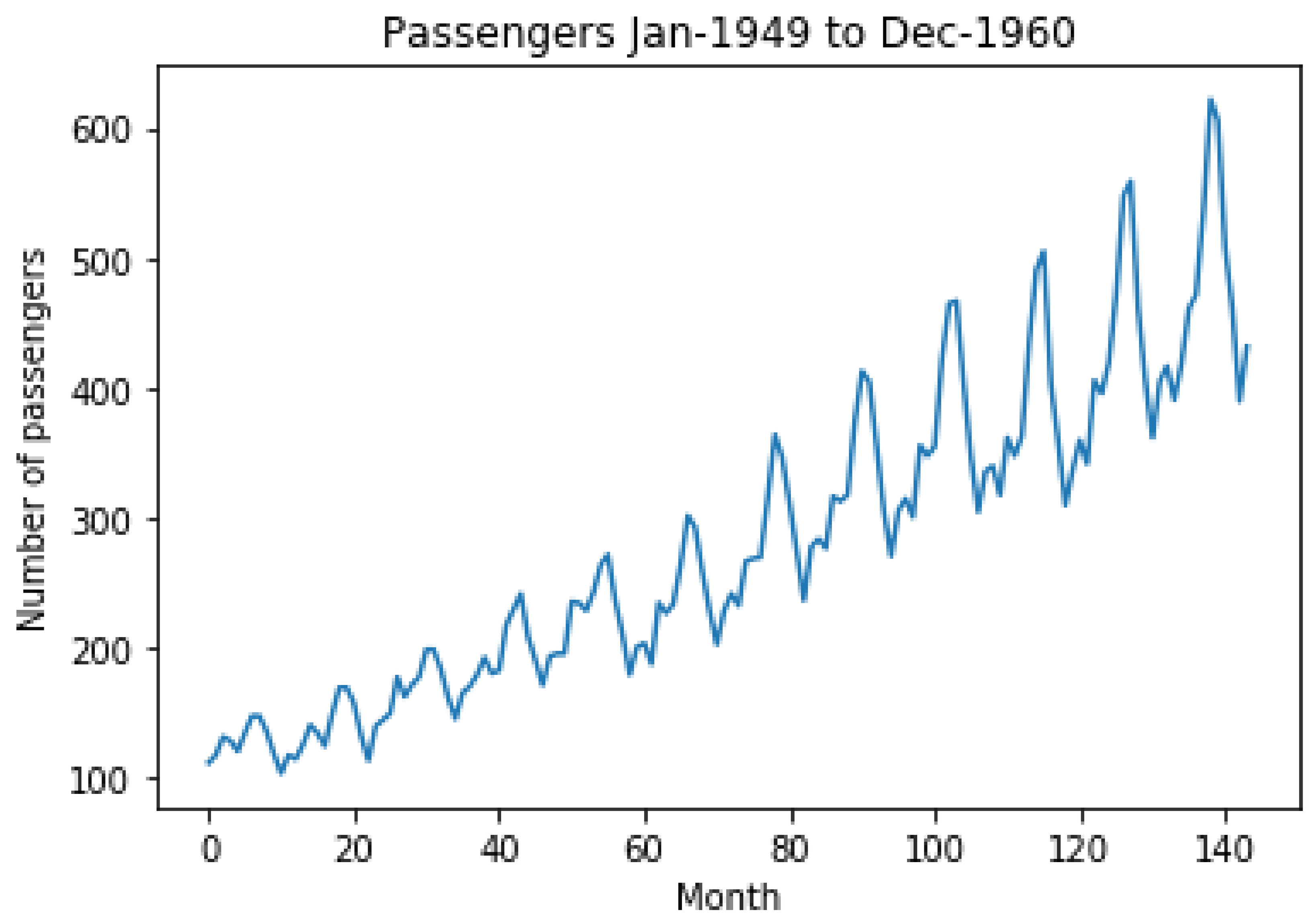
Natural language data
text, speech

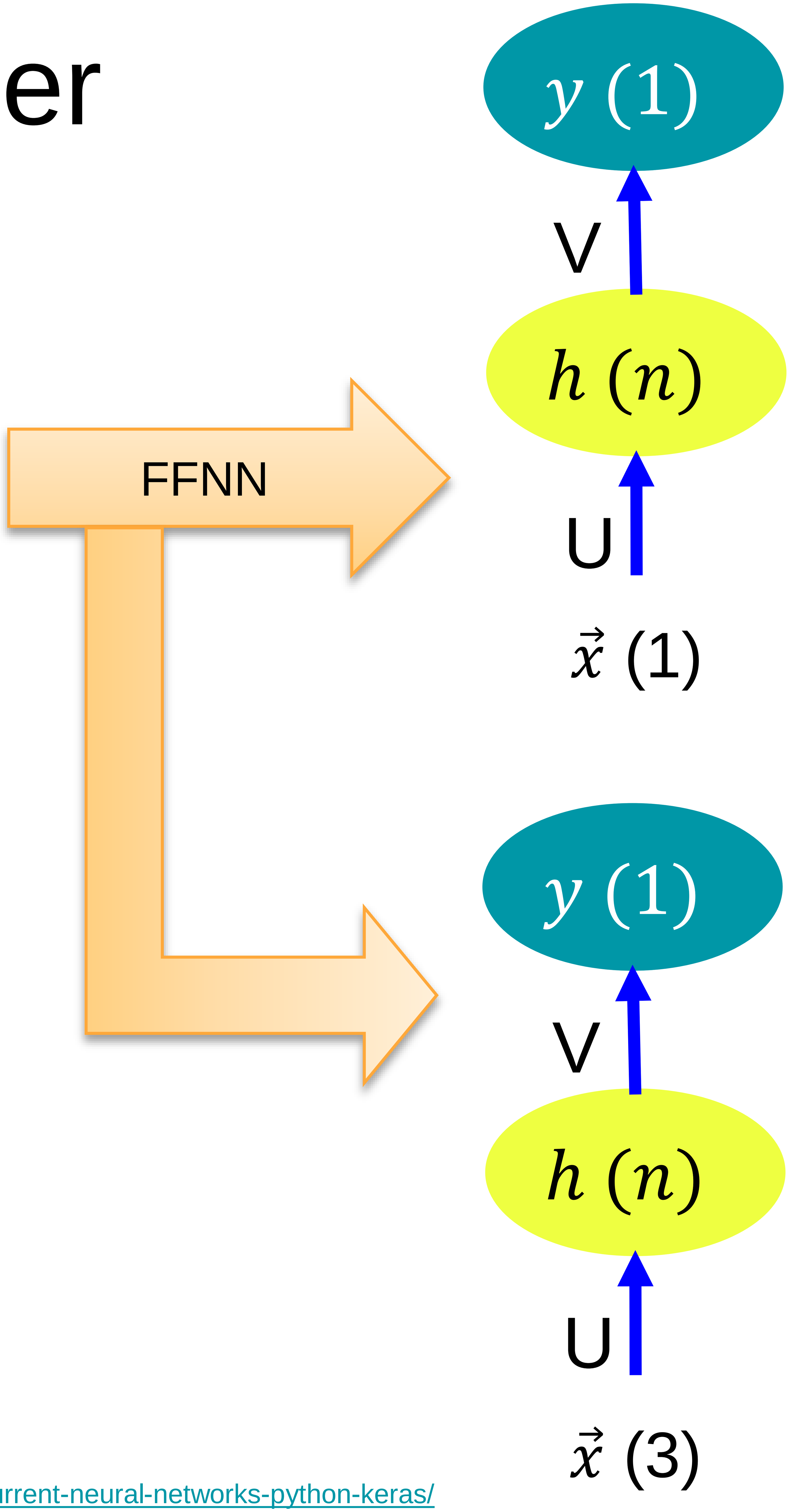
Other sequence data
video

Hidayat, Y., Sutijo, B., Bon, A. T., & Supian, S. (2016). Indonesian financial data modeling and forecasting by using econometrics time series and neural network. *Global Journal of Pure and Applied Mathematics*, *12*(4), 3745-3757.
Tan dkk. (2004): https://slideplayer.com/slide/778153/

Airline Passenger

"Month","Passengers"
"1949-01",112
"1949-02",118
"1949-03",132
"1949-04",129
"1949-05",121





<u>1 Feature Dataset:</u>

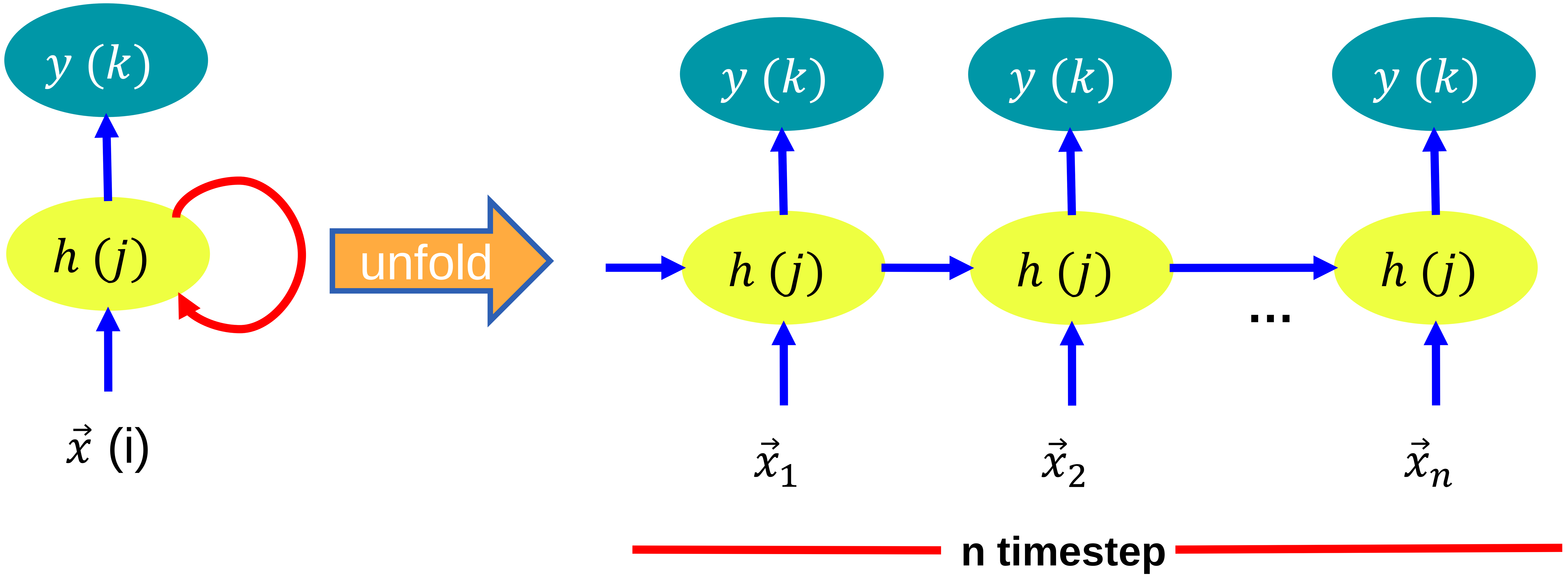
X=t	Y=(t+1)
112	118
118	132
132	129
129	121
121	135

<u>3 Feature Dataset:</u>

X1=t-2	X2=t-1	X3=t	Y=(t+1)
112	118	132	129
118	132	129	121
132	129	121	135
129	121	135	148
121	135	148	148

https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

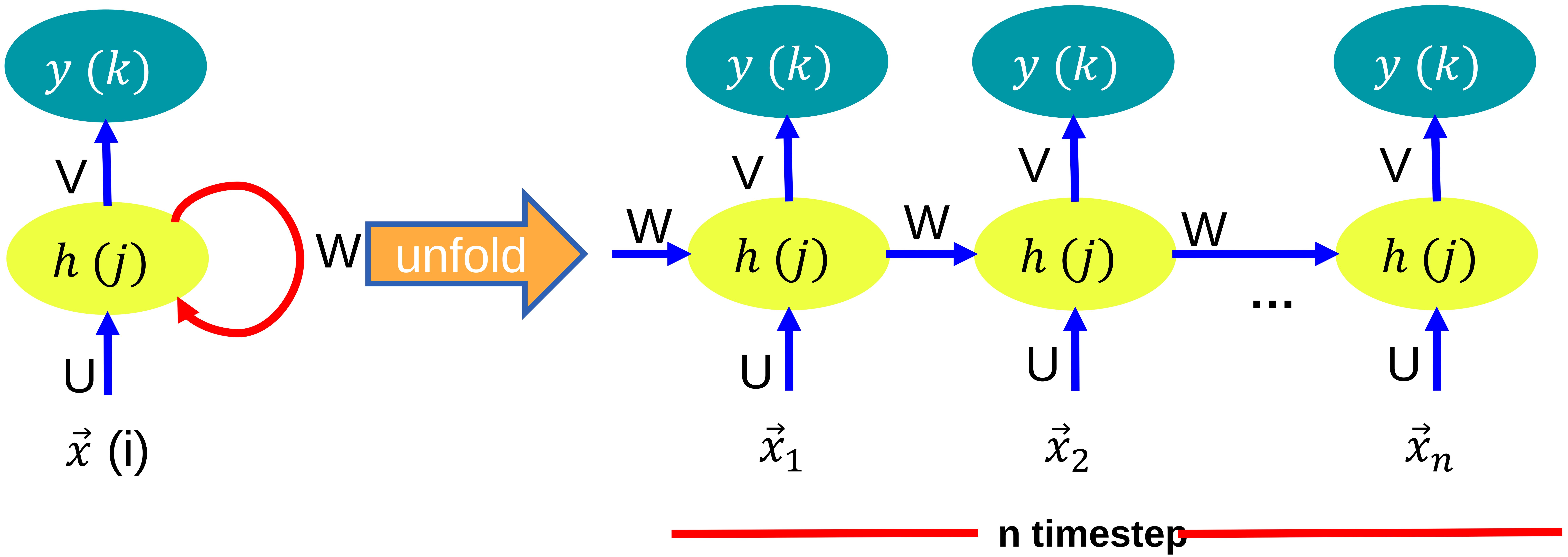
RNN: Neuron Dependent



A RNN unit has loops in them that allow information to be carried across neurons while reading in input. RNN cannot rely on the input alone and must use its recurrent connection to keep track of the context to achieve this task.

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

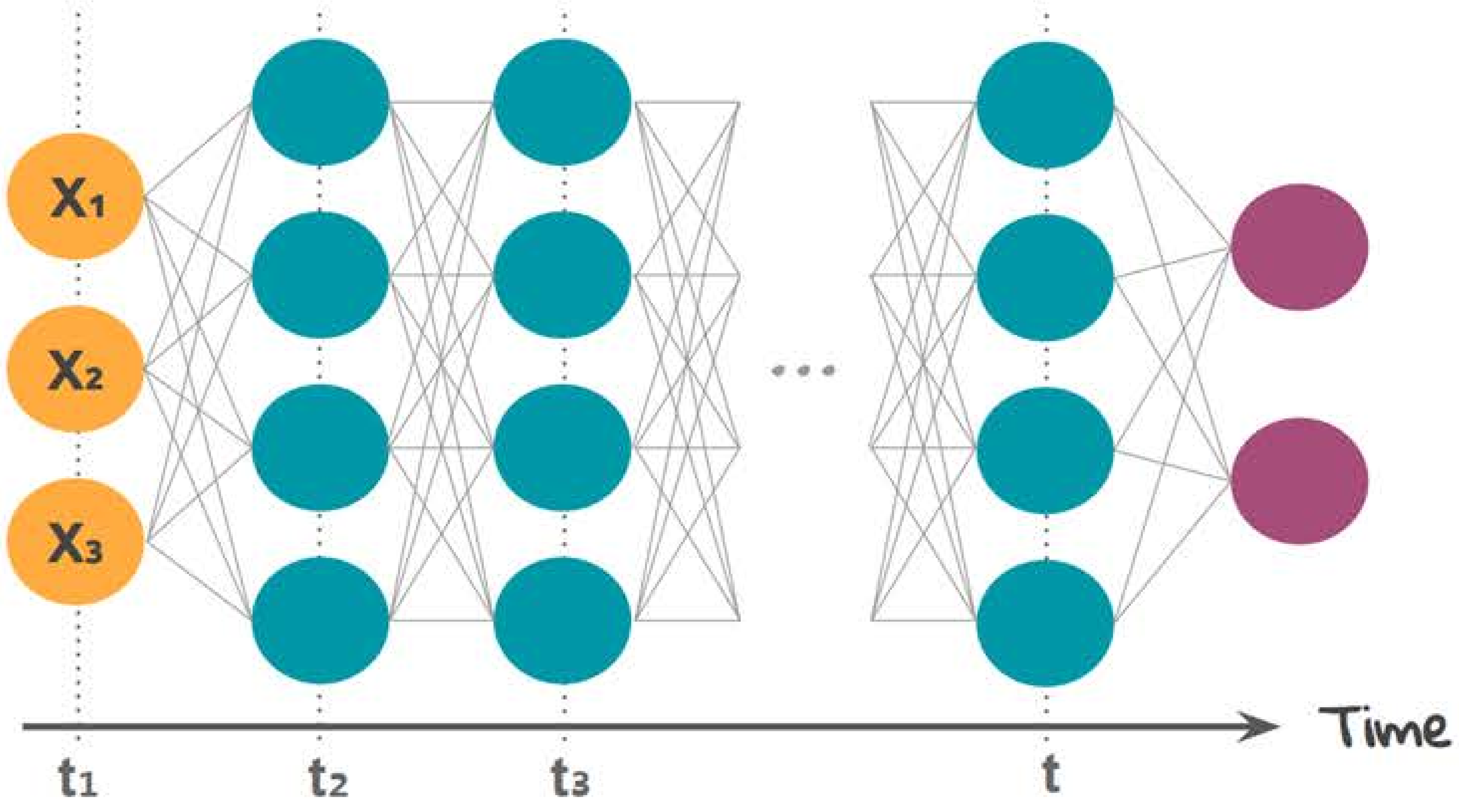
RNN: Parameter Sharing

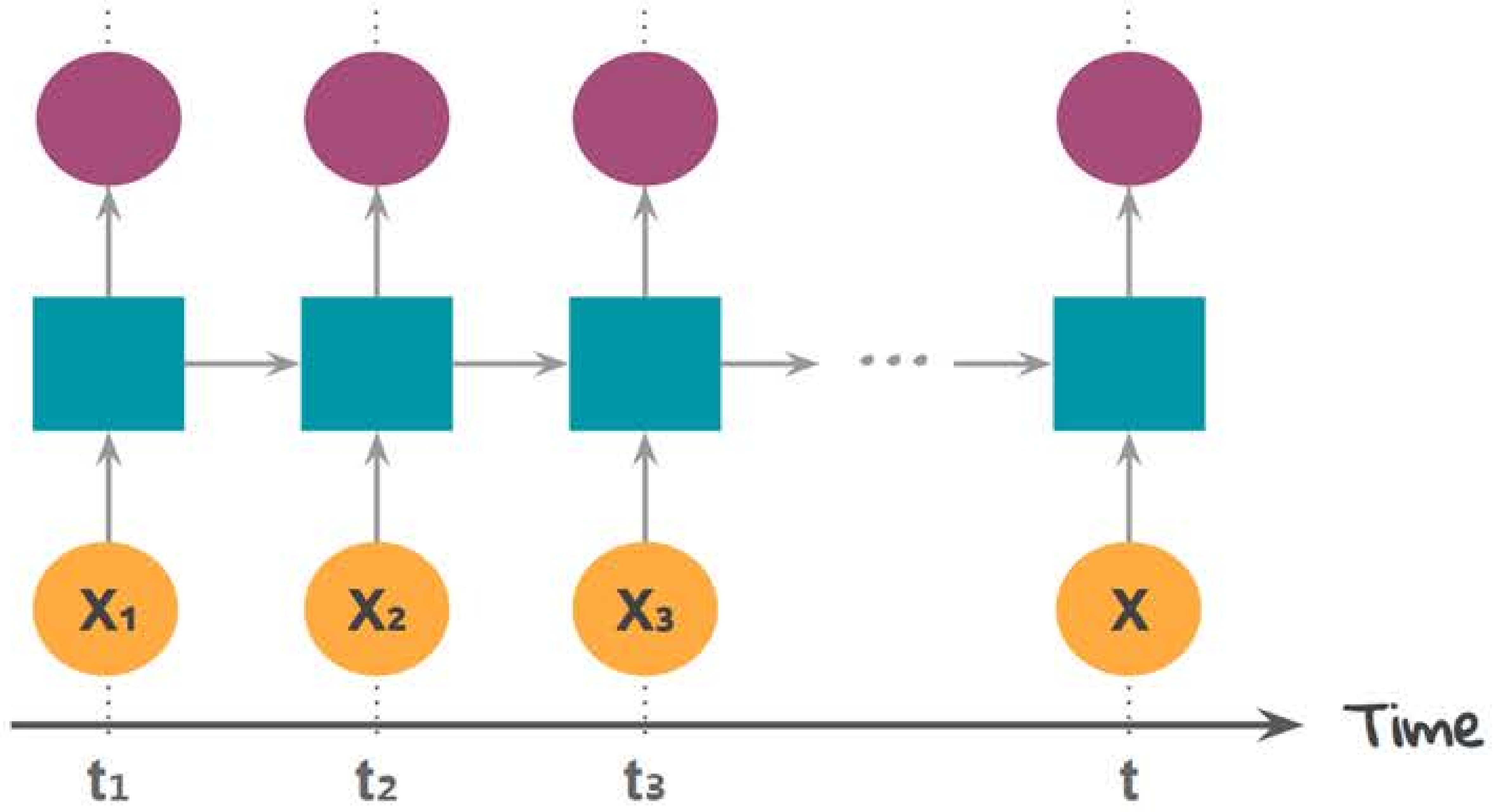


$$h_t = f(Ux_t + (Wh_{t-1} + b_{\text{xh}}))$$
$$y_t = f(Vh_t + b_{\text{hy}})$$

- x_t : feature vector (i features) at step t
- h_t : hidden state.
- y_t : output at step t.
- f: activation function

FFNN vs RNN: Sequential Data

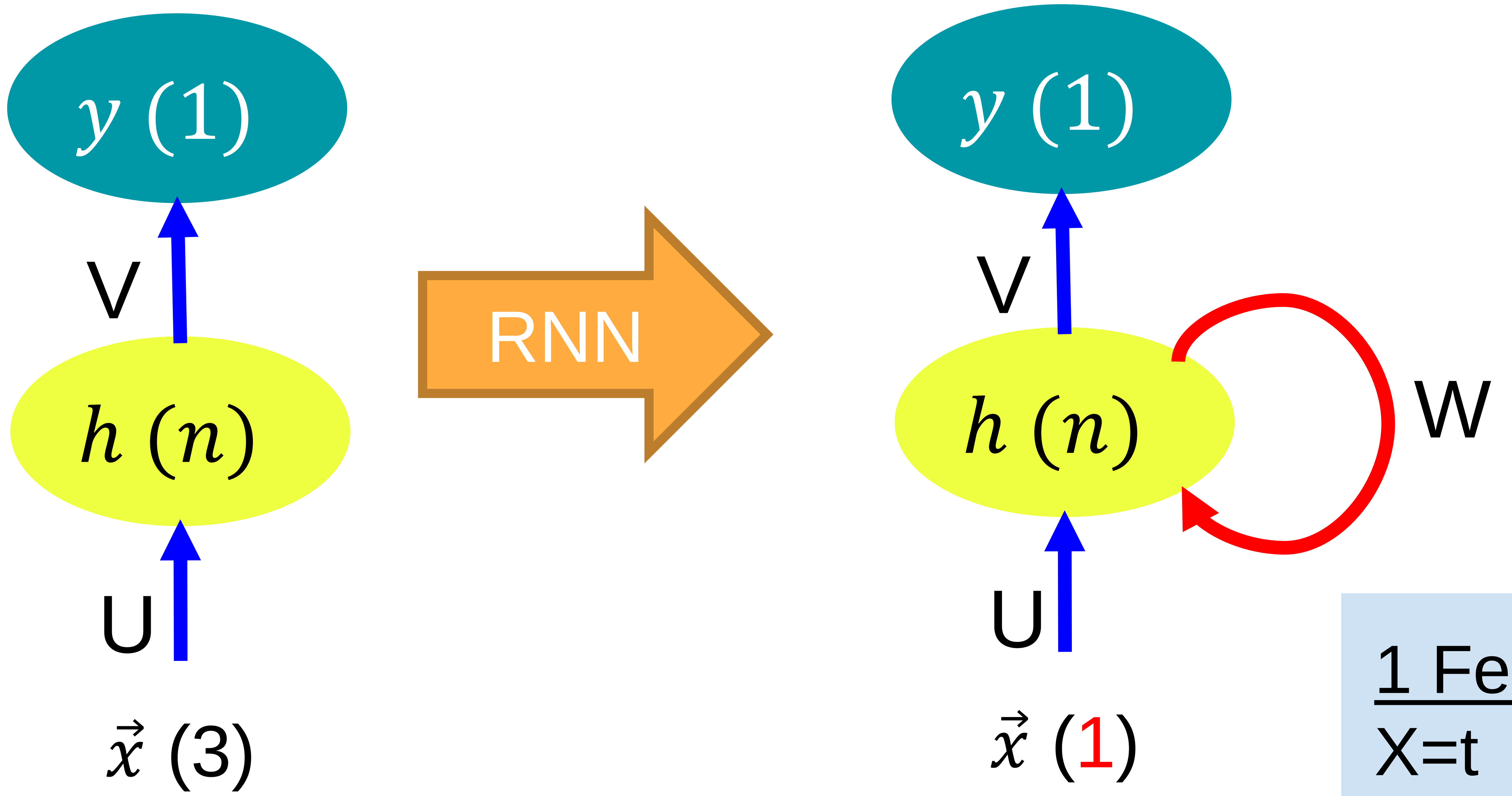




- FFNN: there isn't any concept of order in time between the data
- RNN: there is order in time between the data. We will input **X1** first and then input **X2** to the result of **X1** computation. So in the same way, **X3** is computed with the result from **X2** computation stage.

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

FFNN vs RNN



3 timestep

<u>3 Feature Dataset:</u>

X1	X2	X3	Y=(t+1)
112	118	132	129
118	132	129	121
132	129	121	135
129	121	135	148
121	135	148	148

<u>1 Feature Dataset:</u>

X=t	Y=(t+1)
112	118
118	132
132	129
129	121
121	135

Summary

RNN: Feedback NN

Why RNN ? Order of data matters (Sequence data)

Neuron dependent

Parameter sharing

Forward Propagation

Forward Propagation

Classification for Sequence Data

Sequence: ABCCD…

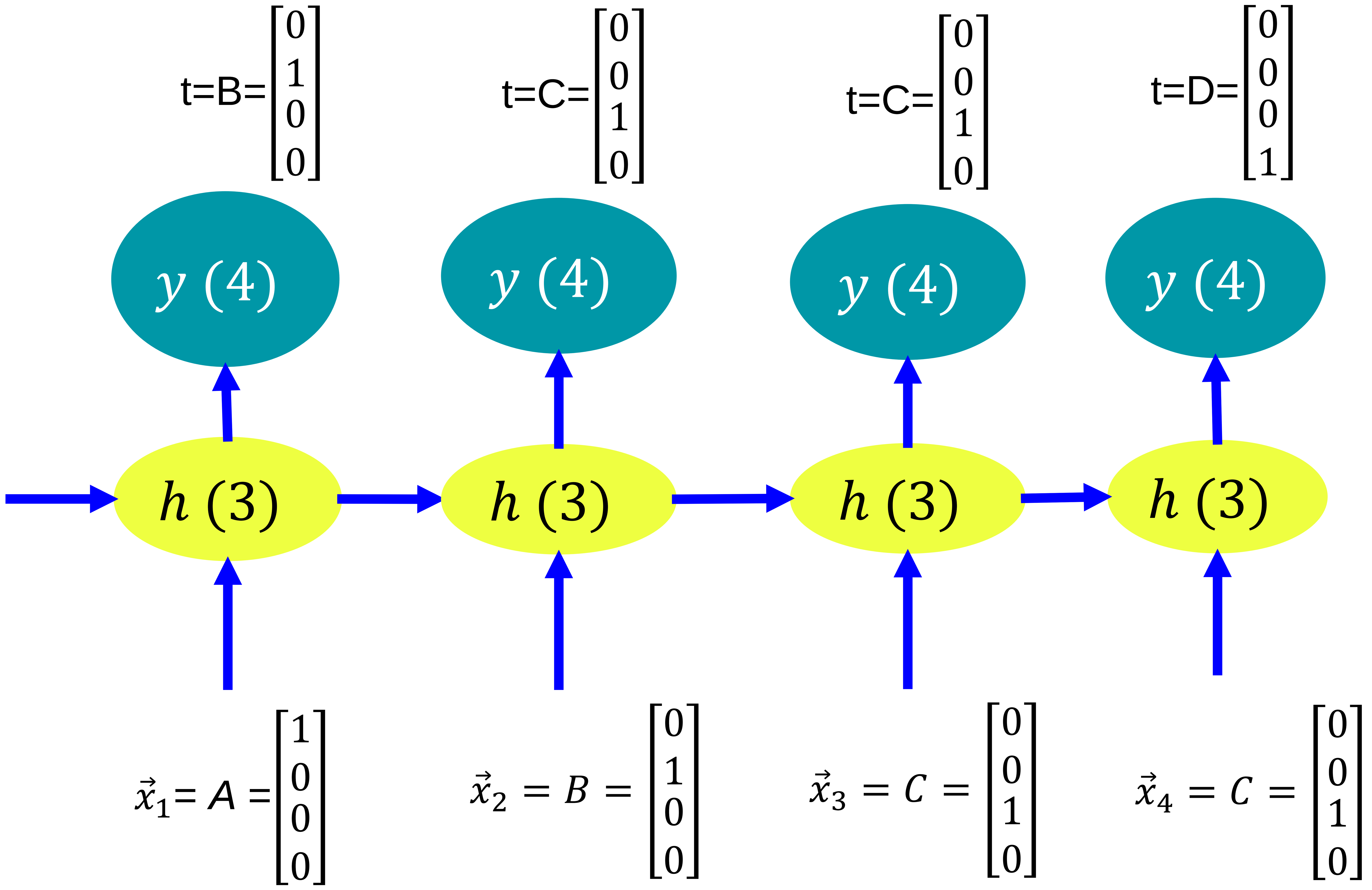
Browsing history: {Homepage}{Electronics}{Camera}{Camera}{ShoppingCart}…
Language model of character: hello…
Language model of words: aku sedang pura pura tertawa …

Vector representation:

$$A=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, B=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}, C=\begin{bmatrix}0\\0\\1\\0\end{bmatrix}, D=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$\text{ABCCD} \ldots = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\begin{bmatrix}0\\0\\1\\0\end{bmatrix}\begin{bmatrix}0\\0\\1\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\cdots$$

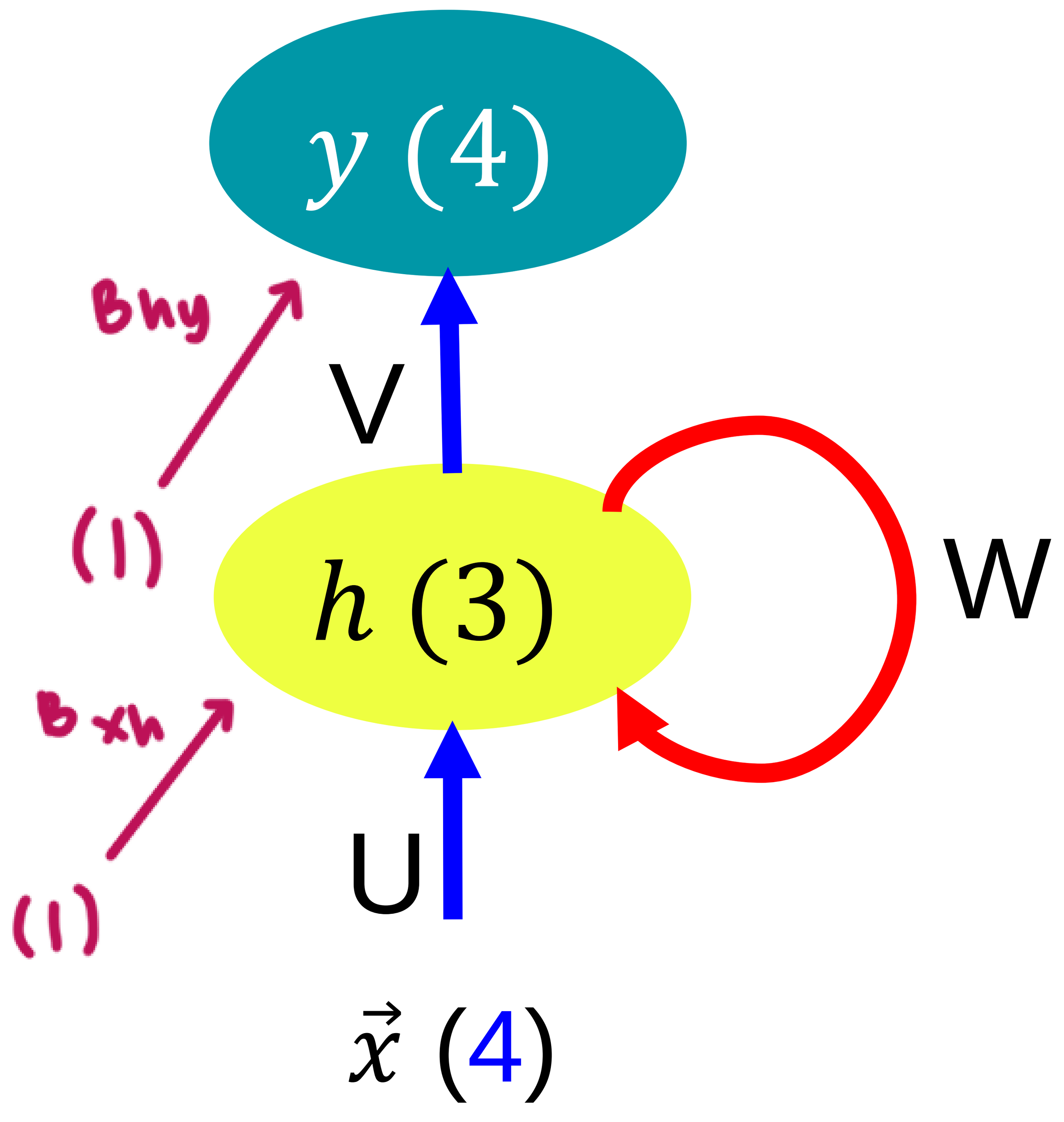
Dataset Construction: 4 timestep



Sequence: ABCCD…

A1	A2	A3	A4	Class
1	0	0	0	B
0	1	0	0	C
0	0	1	0	C
0	0	1	0	D
…				

Sequence Classification: RNN



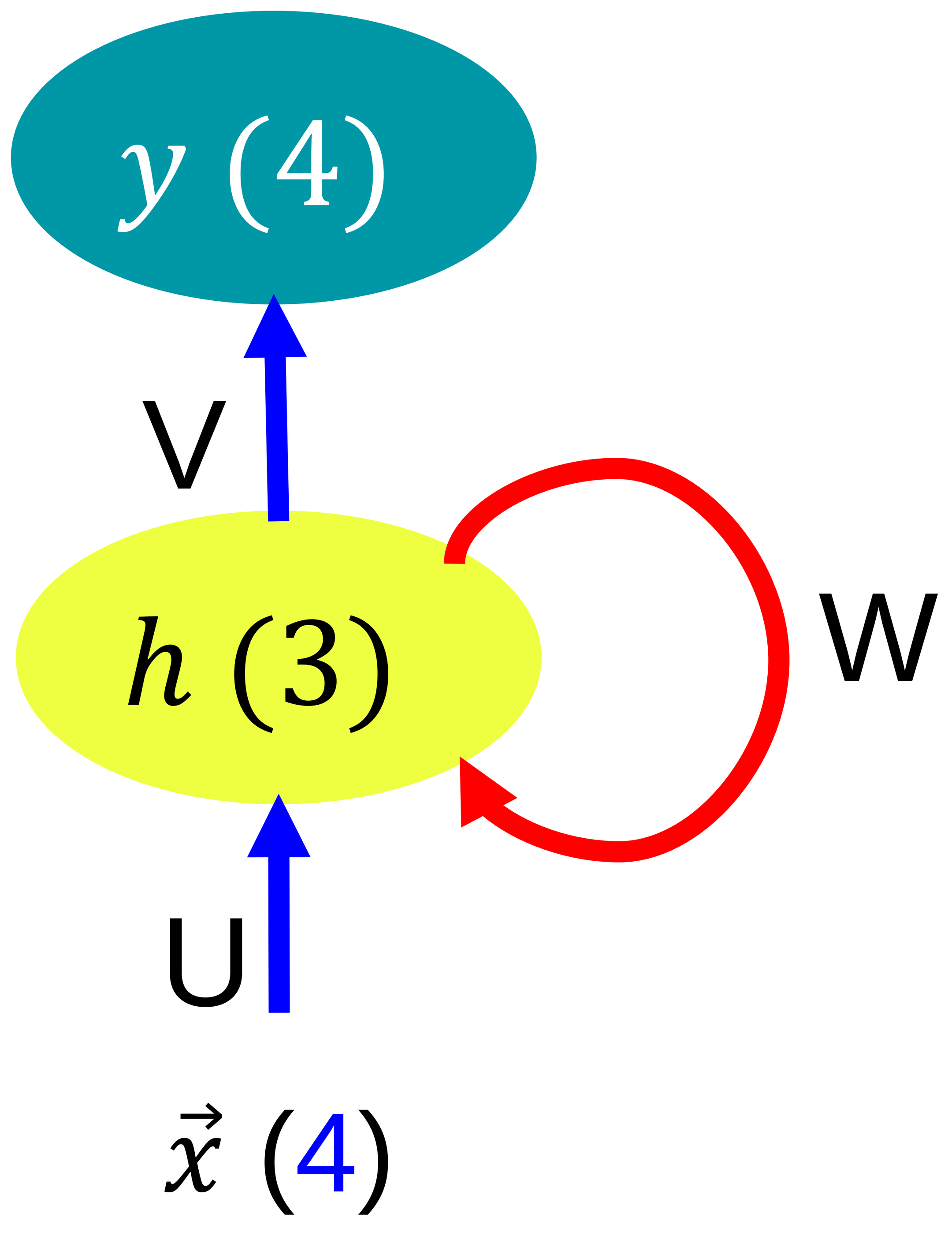
- U: matrix 3x4 (hidden neurons x input dimension)
- V: matrix 4x3 (output neurons x hidden neurons)
- W: matrix 3x3 (hidden neurons x hidden neurons)
- Bias_{xh}: matrix 3x1
- Bias_{hy}: matrix 4x1

$$h_t = tanh(Ux_t + (Wh_{t-1} + b_{\text{xh}}))$$
$$y_t = softmax(Vh_t + b_{\text{hy}})$$

$$h_t = \tanh\left(U.x_t + \left(W.h_{t-1} + b_{xh}\right)\right)$$

$$y_t = \mathrm{softmax}\left(V.h_t + b_{hy}\right)$$

Weight Initialization: Example (Random)



U			
0.100	0.150	0.200	0.300
0.150	0.200	0.300	0.100
0.200	0.300	0.100	0.150

W		
0.500	0.500	0.500
0.500	0.500	0.500
0.500	0.500	0.500

b_xh
0.100
0.100
0.100

ht-1
0
0
0

V		
0.100	0.200	0.300
0.200	0.300	0.100
0.300	0.100	0.200
0.100	0.100	0.100

b_hy
0.100
0.100
0.100
0.100

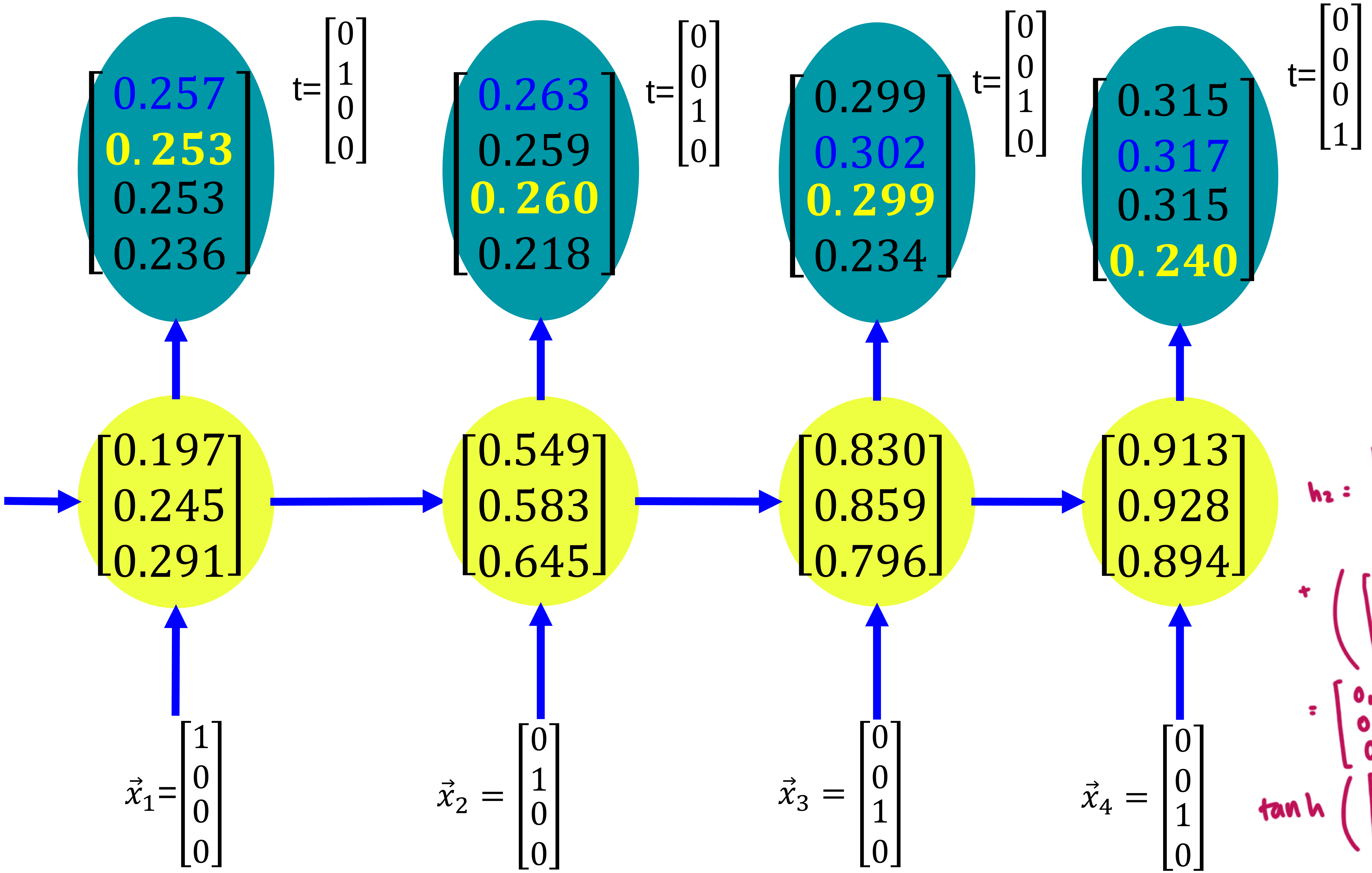
$$h_1 = \begin{bmatrix} 0,1 & 0,15 & 0,2 & 0,3 \\ 0,15 & 0,2 & 0,3 & 0,1 \\ 0,2 & 0,3 & 0,1 & 0,15 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \left(\begin{bmatrix} 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0,1 \\ 0,1 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0,1 \\ 0,15 \\ 0,2 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0,1 \\ 0,1 \end{bmatrix} = \begin{bmatrix} 0,2 \\ 0,25 \\ 0,3 \end{bmatrix}$$

$$\tanh\left(\begin{bmatrix} 0,2 \\ 0,25 \\ 0,3 \end{bmatrix}\right) = \begin{bmatrix} 0,197 \\ 0,245 \\ 0,291 \end{bmatrix}$$

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Forward Propagation: 1 instance, 4 timestep



$$y_1 = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,1 \\ 0,3 & 0,1 & 0,2 \\ 0,1 & 0,1 & 0,1 \end{bmatrix} \begin{bmatrix} 0,197 \\ 0,245 \\ 0,291 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0,1 \\ 0,1 \\ 0,1 \end{bmatrix} = \begin{bmatrix} 0,256 \\ 0,242 \\ 0,2418 \\ 0,1733 \end{bmatrix}$$

$$y_1 = \mathrm{softmax}\left(\begin{bmatrix} 0,256 \\ 0,242 \\ 0,2418 \\ 0,1733 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 1,29175 / 5,0283 \\ 1,27379 / 5,0283 \\ 1,27354 / 5,0283 \\ 1,18922 / 5,0283 \end{bmatrix} = \begin{bmatrix} 0,257 \\ 0,253 \\ 0,253 \\ 0,236 \end{bmatrix}$$

$$h_2 = \begin{bmatrix} 0,1 & 0.15 & 0.2 & 0,3 \\ 0,15 & 0,2 & 0,3 & 0,1 \\ 0,2 & 0,3 & 0,1 & 0,15 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \left(\begin{bmatrix} 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \end{bmatrix} \begin{bmatrix} 0,197 \\ 0,245 \\ 0,291 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0,1 \\ 0,1 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0,15 \\ 0,2 \\ 0,3 \end{bmatrix} + \begin{bmatrix} 0,3665 \\ 0,3665 \\ 0,3665 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0,1 \\ 0,1 \end{bmatrix} = \begin{bmatrix} 0,6165 \\ 0,6665 \\ 0,7665 \end{bmatrix}$$

$$\tanh\left(\begin{bmatrix} 0,6165 \\ 0,6665 \\ 0,7665 \end{bmatrix} \right) = \begin{bmatrix} 0,549 \\ 0.583 \\ 0,645 \end{bmatrix}$$

Computing h_t and y_t : Timestep t1 and t2

$$h_t = tanh(Ux_t + (Wh_{t-1} + b_{\text{xh}}))$$
$$y_t = softmax(Vh_t + b_{\text{hy}})$$

t1=< $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ >

Uxt	Wht-1+bxh	net_ht	ht
0.100	0.100	0.200	0.197
0.150	0.100	0.250	0.245
0.200	0.100	0.300	0.291

Vht+bhy	exp(Vht+bhy)	yt
0.256	1.292	0.257
0.242	1.274	0.253
0.242	1.274	0.253
0.173	1.189	0.236

t2=< $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ >

Uxt	Wht-1+bxh	net_ht	ht
0.150	0.467	0.617	0.549
0.200	0.467	0.667	0.583
0.300	0.467	0.767	0.645

Vht+bhy	exp(Vht+bhy)	yt
0.465	1.592	0.263
0.449	1.567	0.259
0.452	1.571	0.260
0.278	1.320	0.218

Computing h_t and y_t : Timestep t3 and t4

$$h_t = tanh(Ux_t + (Wh_{t-1} + b_{\mathrm{xh}}))$$
$$y_t = softmax(Vh_t + b_{\mathrm{hy}})$$

t3=< $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ >

Uxt	Wht-1+bxh	net_ht	ht
0.200	0.988	1.188	0.830
0.300	0.988	1.288	0.859
0.100	0.988	1.088	0.796

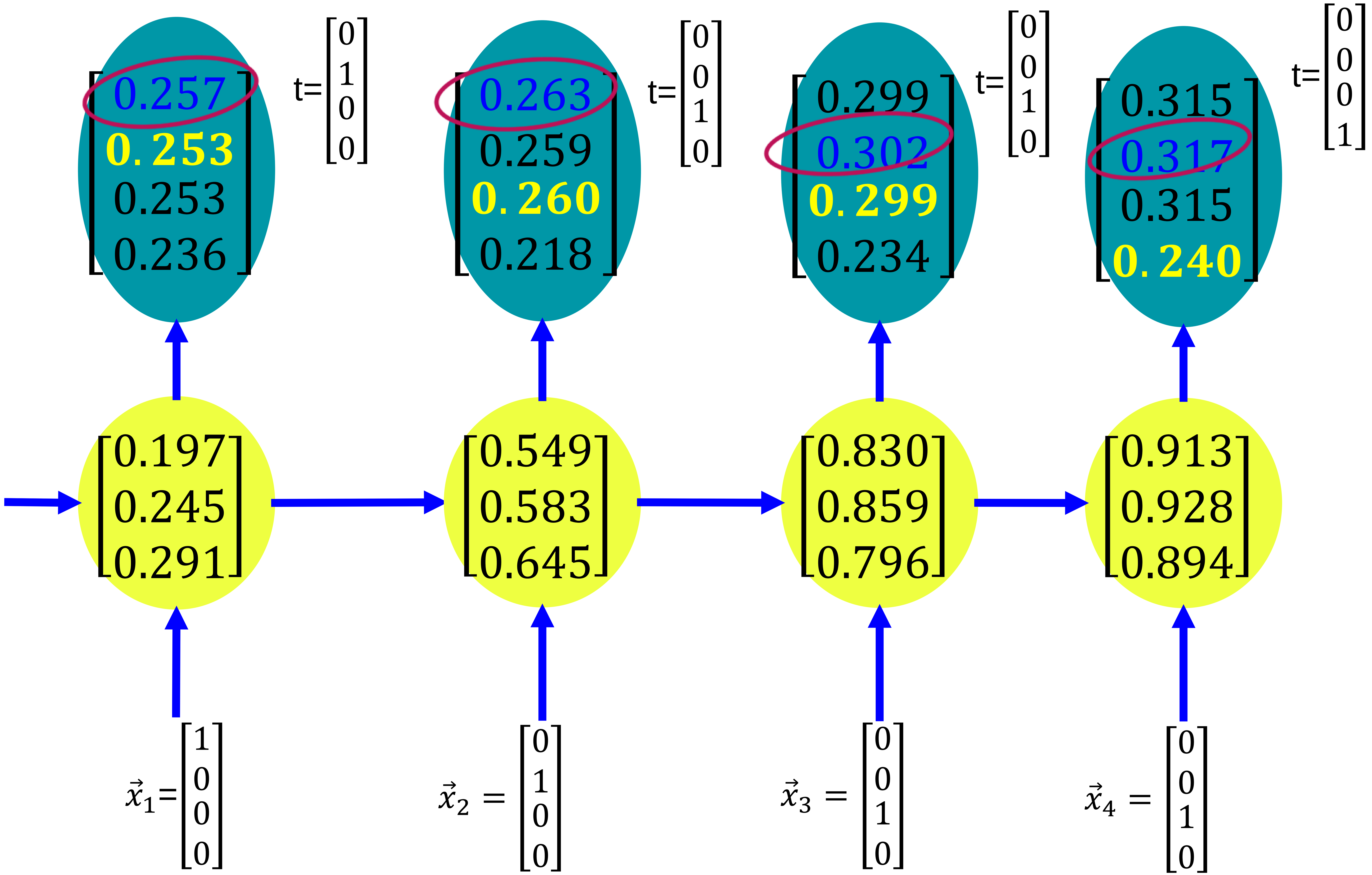
Vht+bhy	exp(Vht+bhy)	yt
0.594	1.811	0.299
0.603	1.828	0.302
0.594	1.812	0.299
0.349	1.417	0.234

t4=< $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$ >

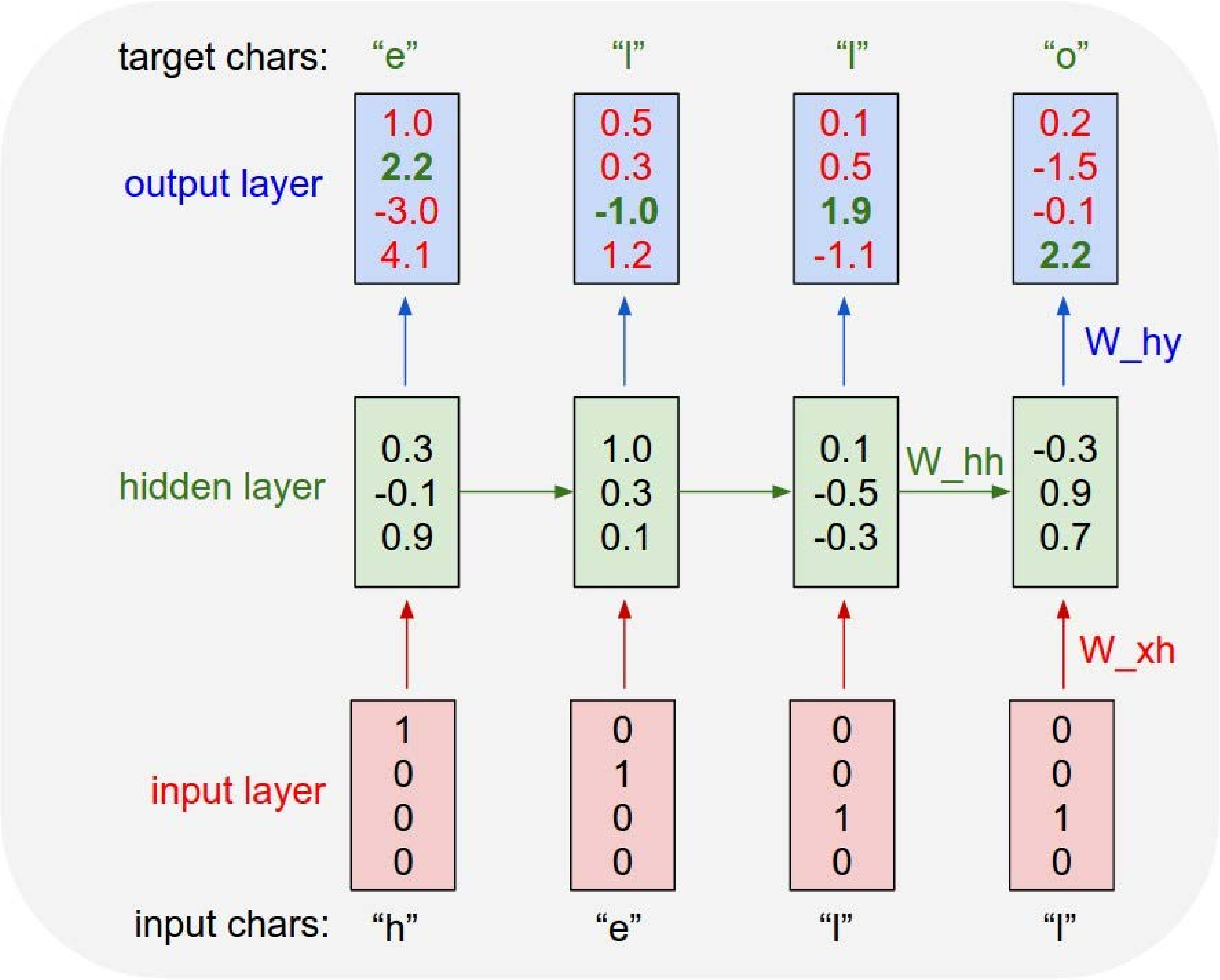
Uxt	Wht-1+bxh	net_ht	ht
0.200	1.343	1.543	0.913
0.300	1.343	1.643	0.928
0.100	1.343	1.443	0.894

Vht+bhy	exp(Vht+bhy)	yt
0.645	1.906	0.315
0.650	1.916	0.317
0.645	1.907	0.315
0.373	1.453	0.240

Forward Propagation: 1 instance, 4 timestep



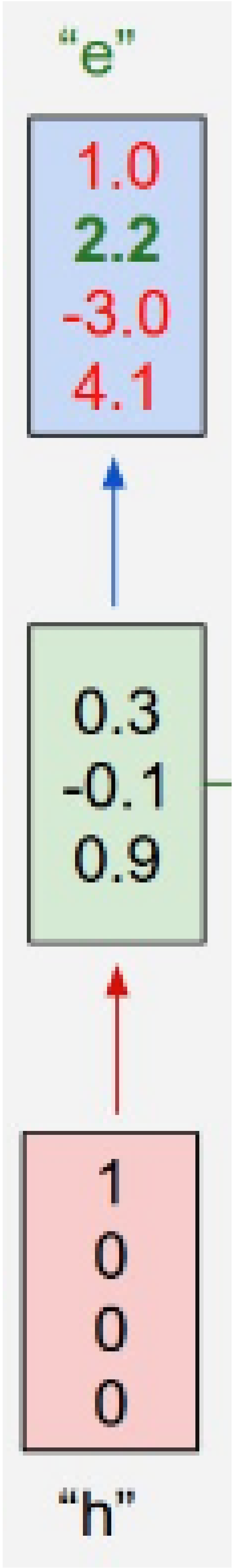
Language Model of Character



- t=1 (input “h”): output=”o”, target=”e”
- t=2 (input “e”): output=”o”, target=”l”
- t=3 (input “l”): output=”l”, target=”l”
- t=4 (input “l”): output=”o”, target=”o”

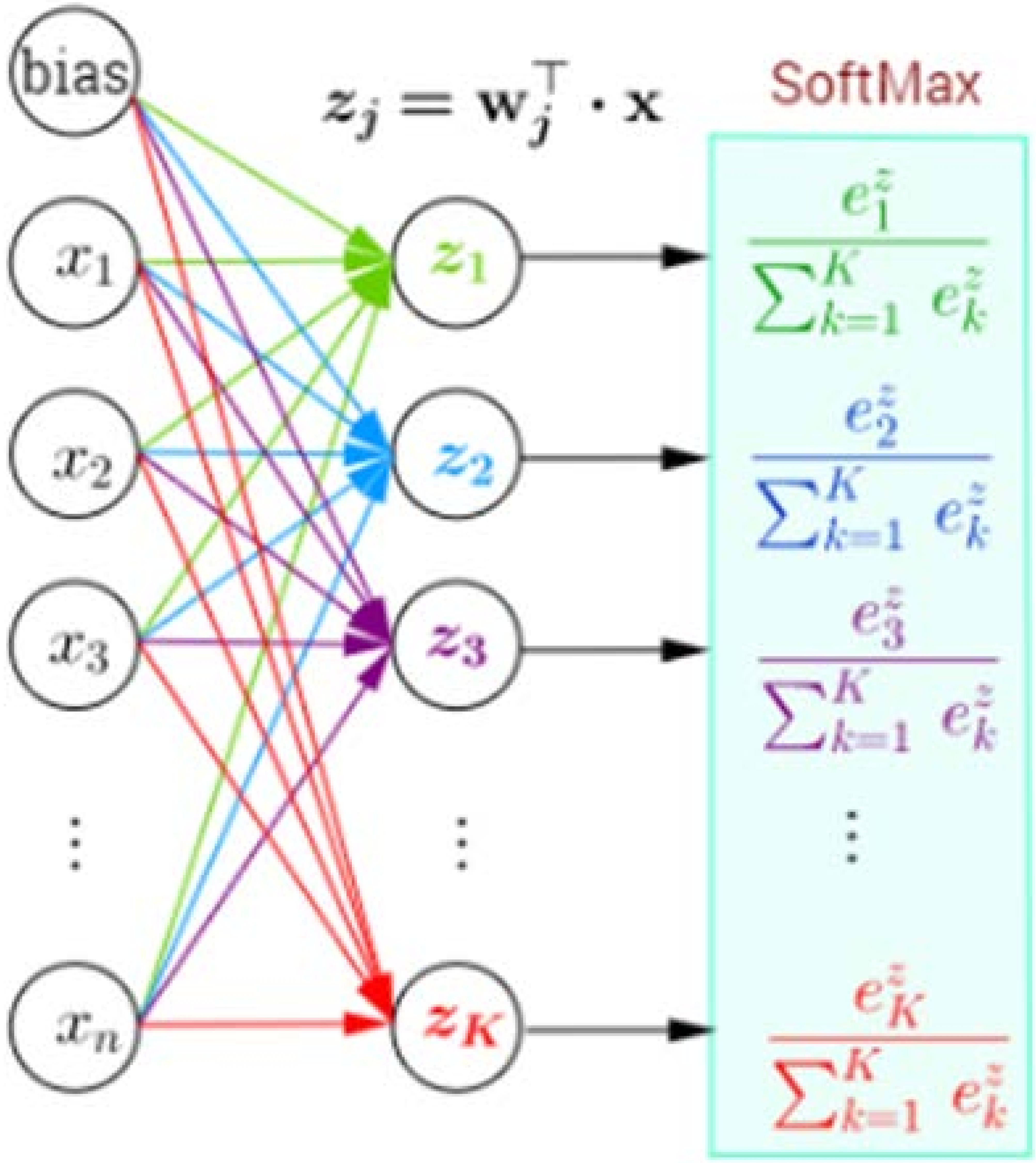
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Output with Softmax



Target: ‘e’
Output: ‘o’

z	exp(z)	P(z)
1	2,718	0
2,2	9,025	0,1
-3	0,05	0
4,1	60,34	0,8
	72,13	1



Summary

Sequence Classification

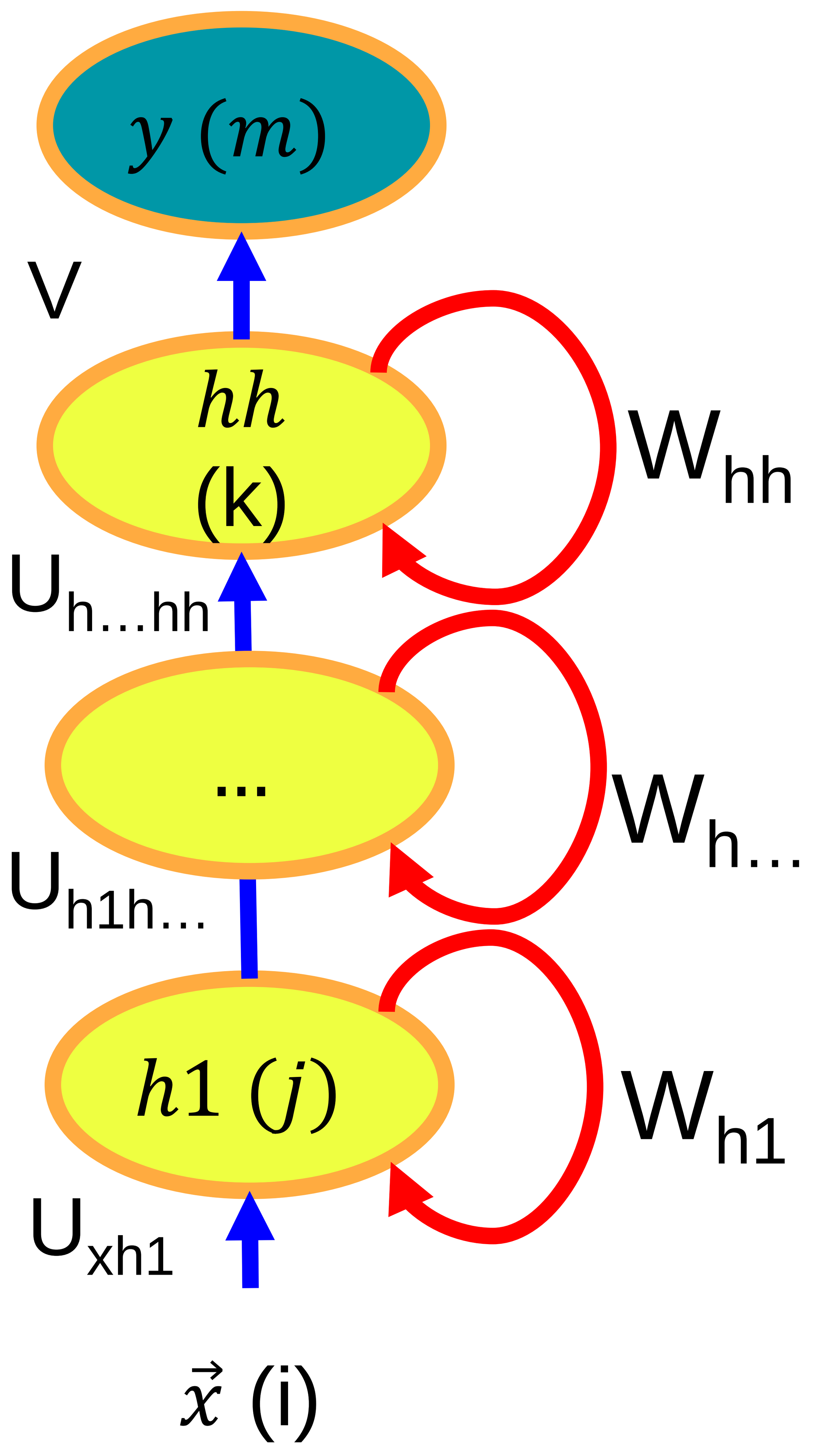
Dataset Construction

Computing h_t and y_t

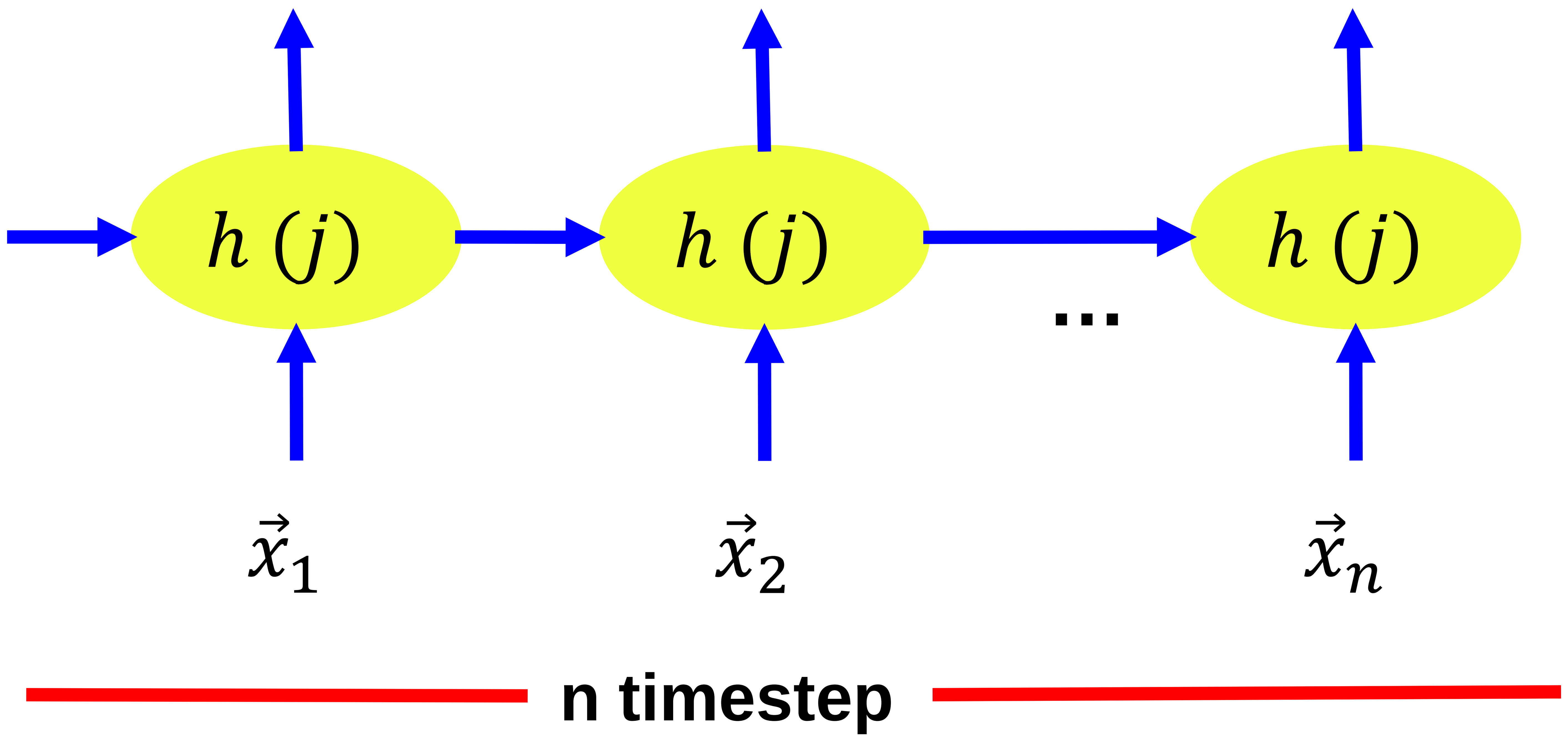
RNN Architecture

RNN Architecture

General Architecture



Return sequence = True/False



Architecture

fixed-sized input vector xt

RNN state st

fixed-sized output vector ot

One to many: image captioning
Many to one: text classification
Many to many: machine translation, video frame classification, POS tagging

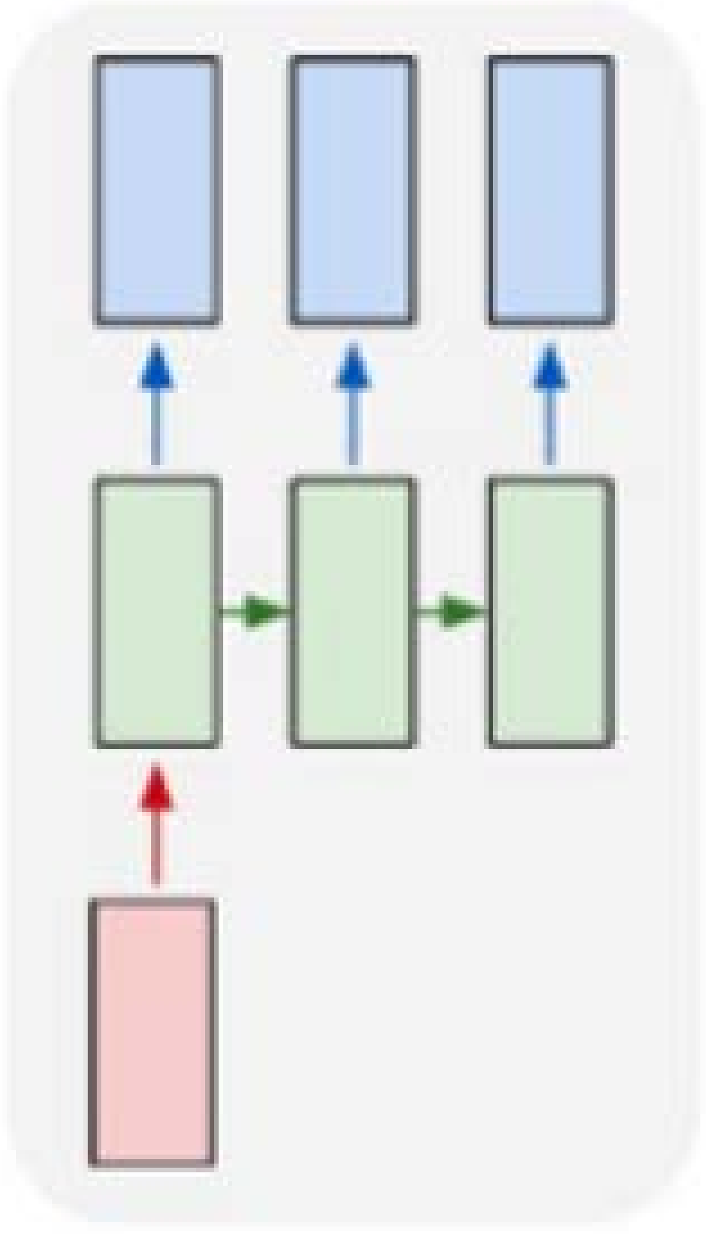
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

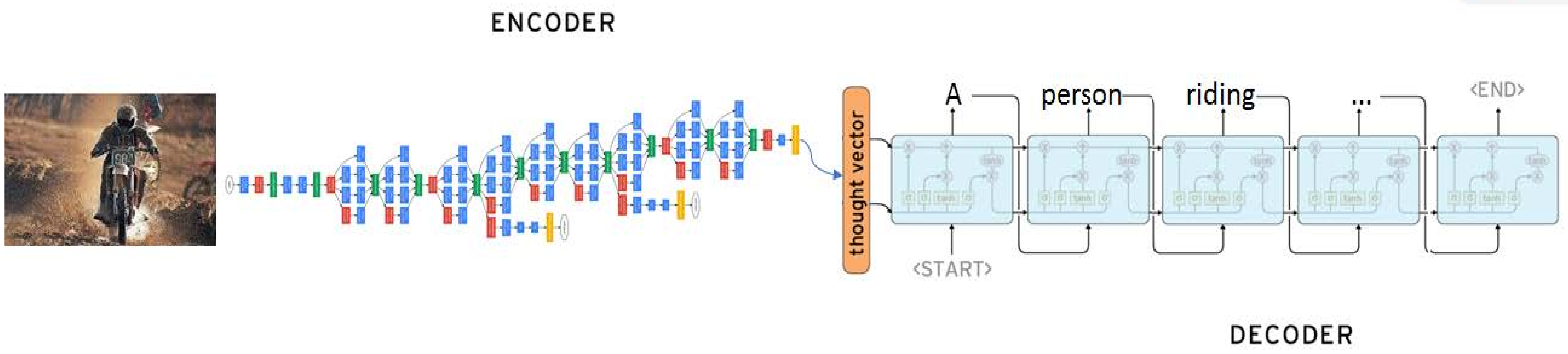
One to Many: Image Captioning

image → text → sequence of token

feature map (vektor)
image → CNN →(one) RNN → text

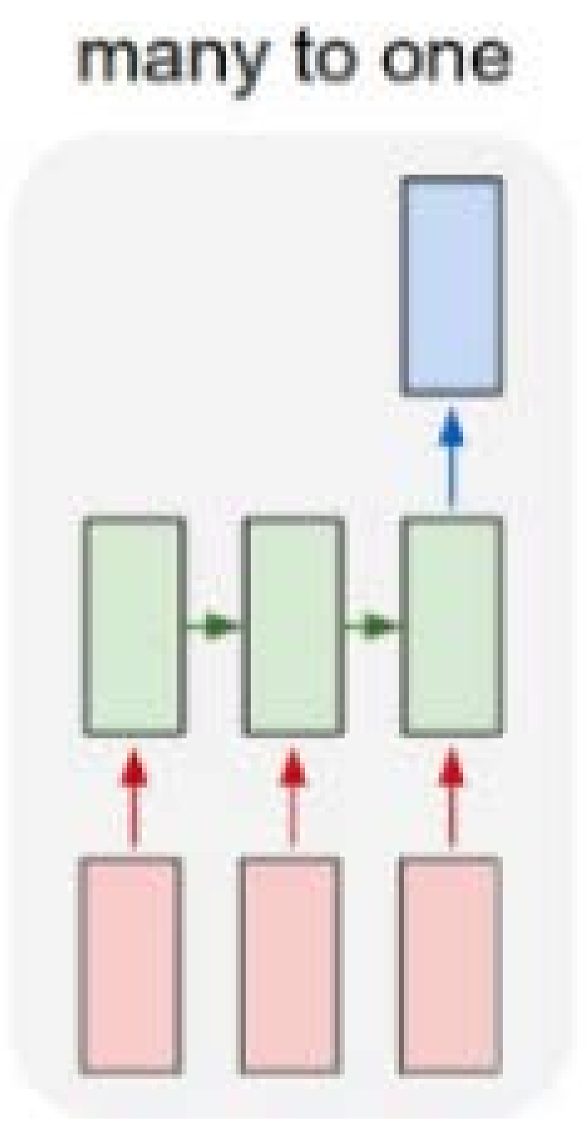
one to many





CNN *Encoder* (Inception) - RNN *Decoder* (LSTM) (Vinyals dkk., 2014)

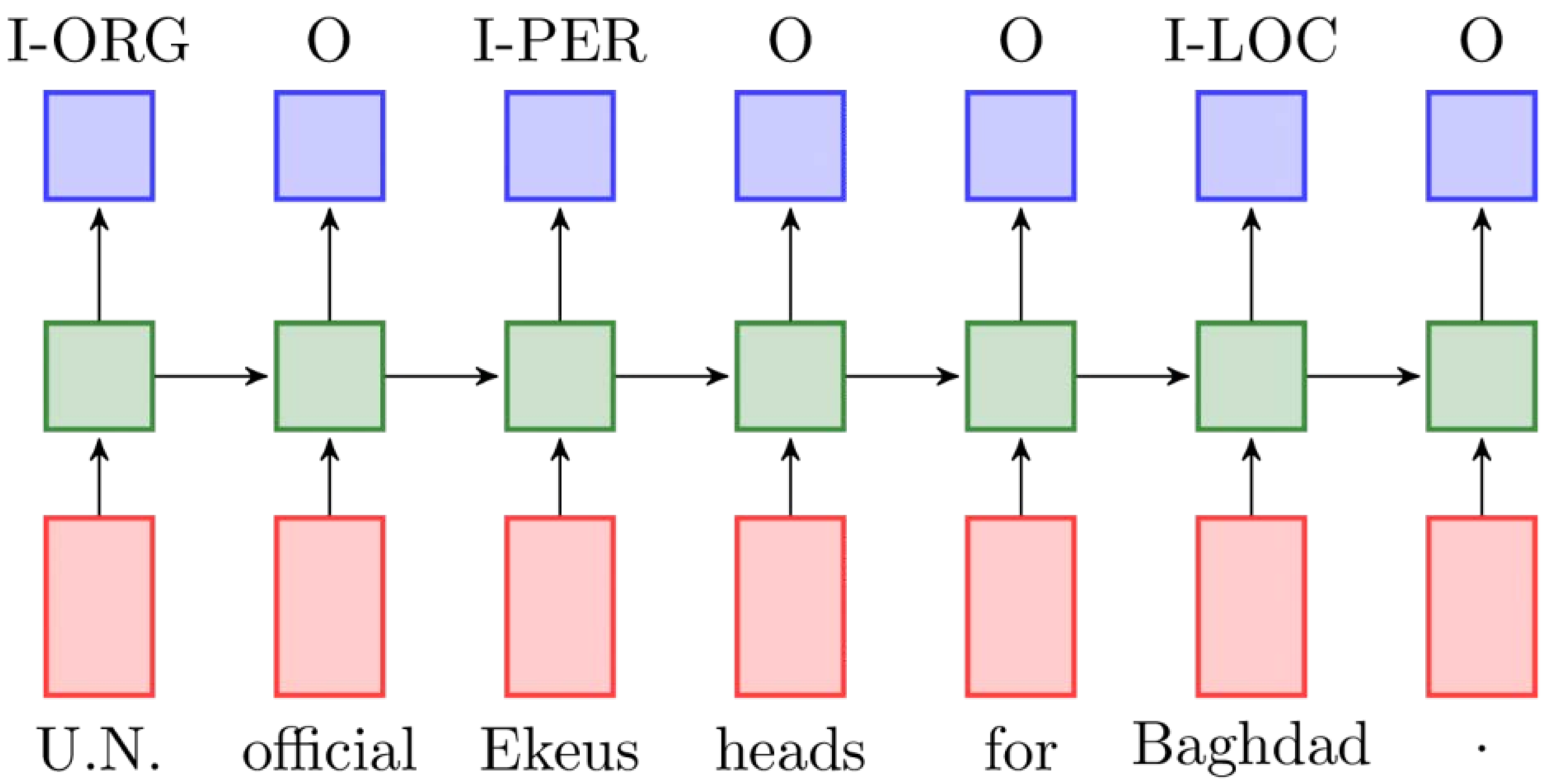
Many to One: Text Classification



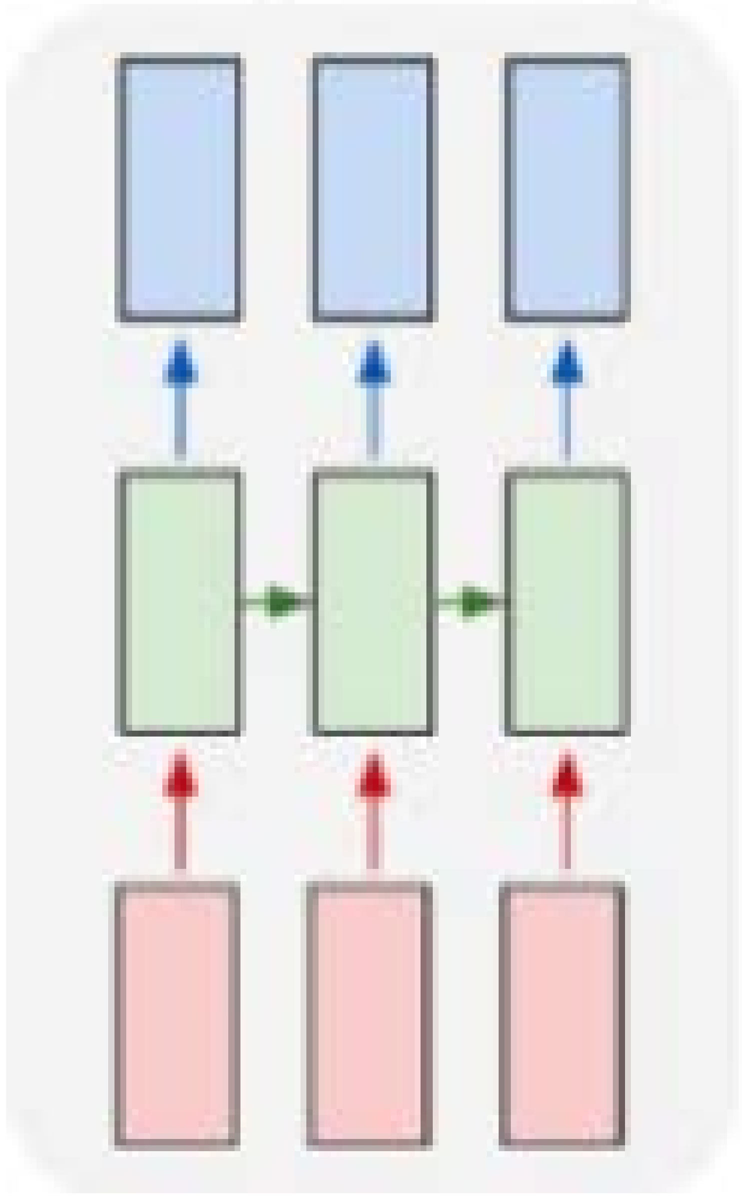
https://www.oreilly.com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow udah jelas indexnya

Many to Many: Sequence Tagging

1 input menghasilkan 1 output



many to many

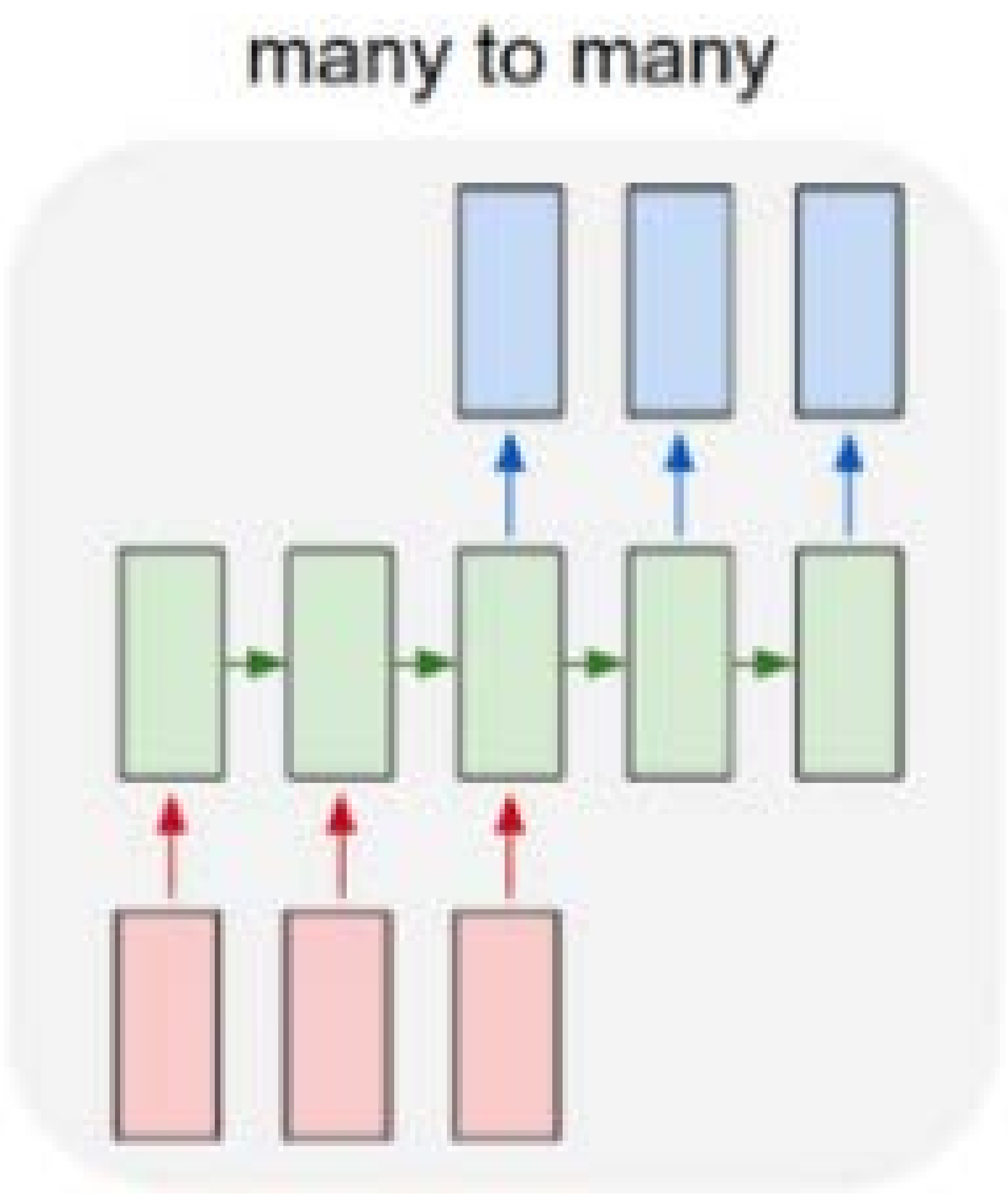


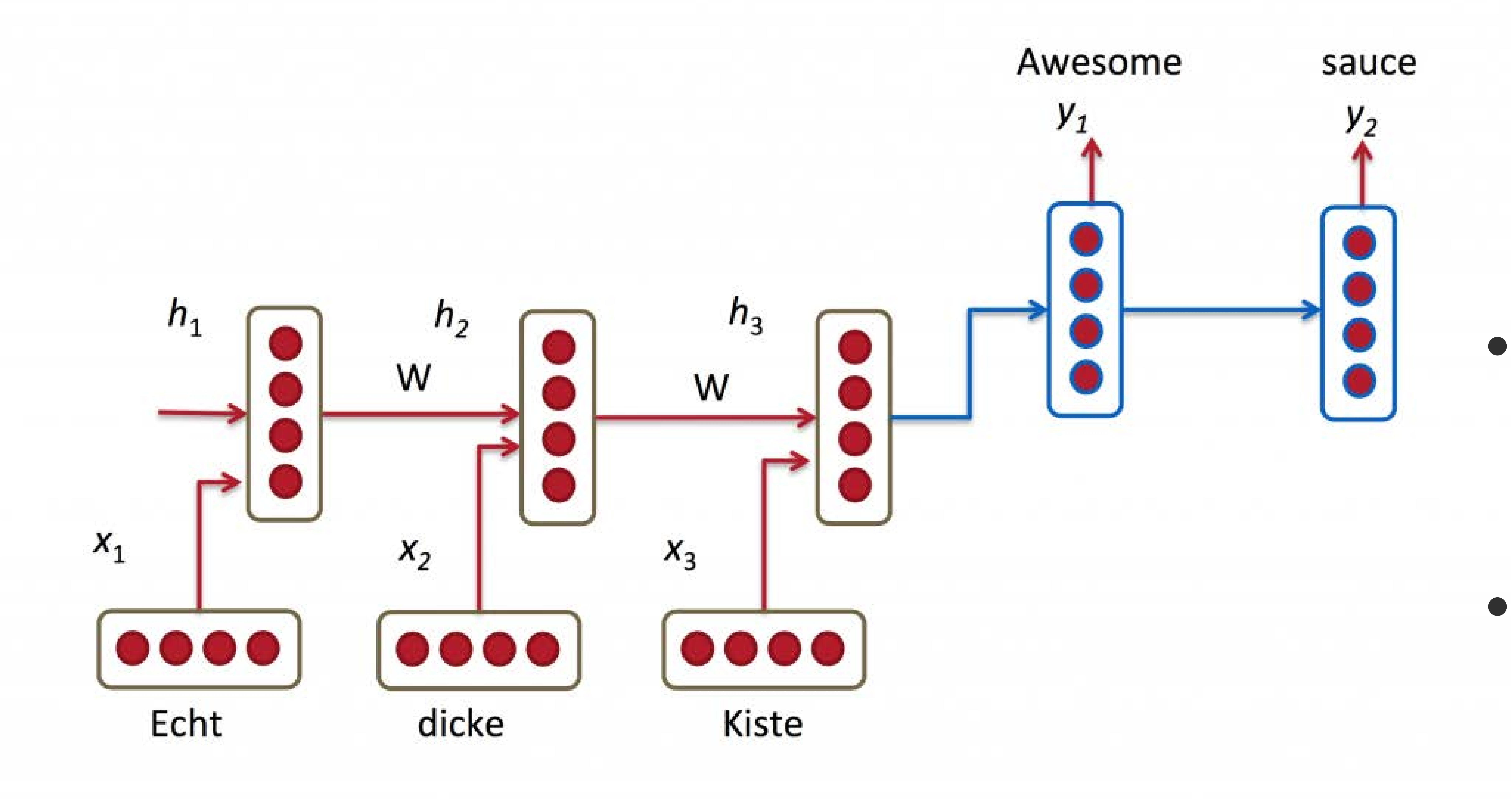
Input is a sequence of words, and output is the sequence of POS tag for each word.

https://www.depends-on-the-definition.com/guide-sequence-tagging-neural-networks-python/

selesaikan semua sampai indeks terakhir
baru masuk lg ke RNN utk hasilin output

Many to Many: Machine Translation

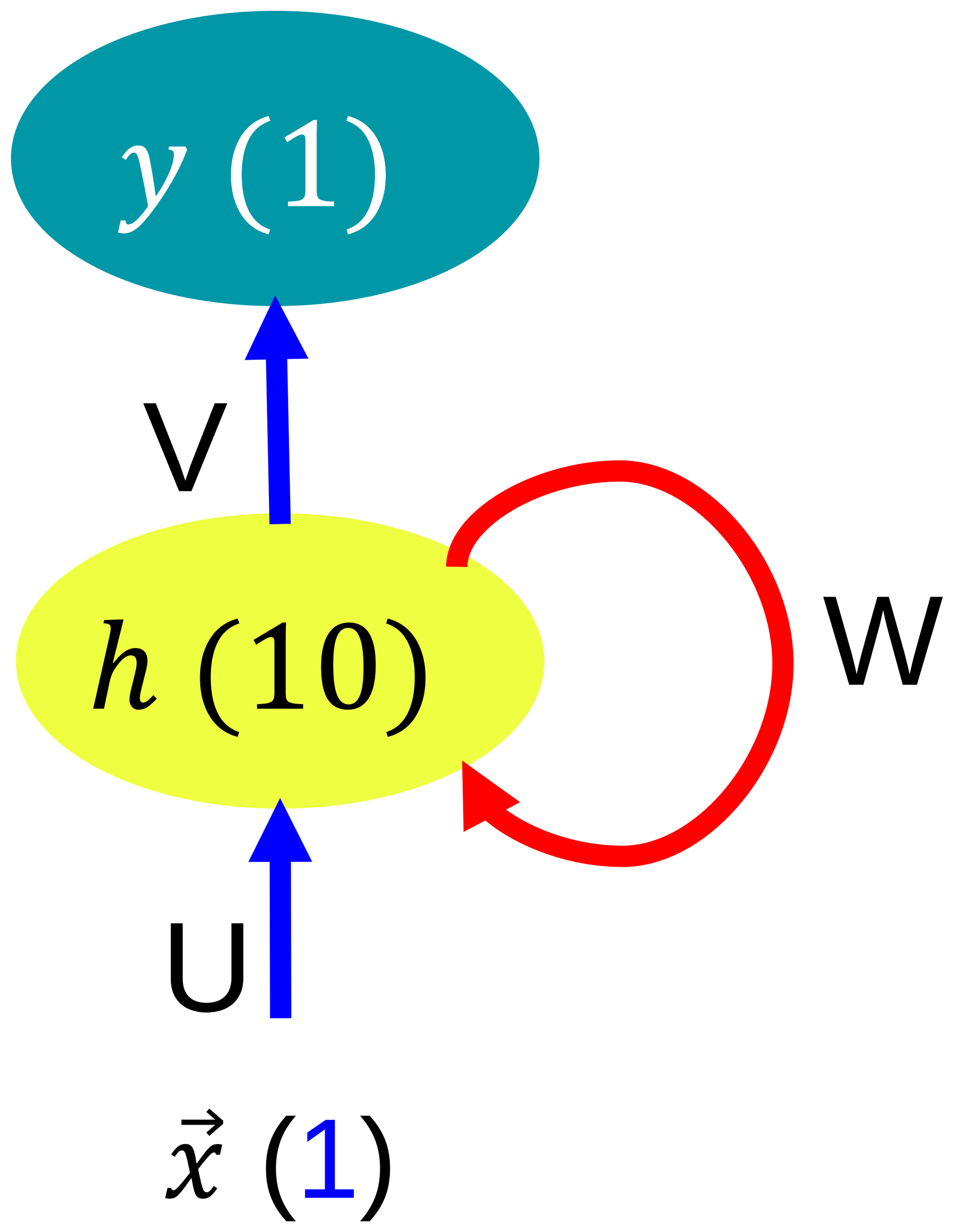




- Machine Translation: input is a sequence of words in source language (e.g. German). Output is a sequence of words in target language (e.g. English).
- A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Implementing RNN on Keras: Many to One

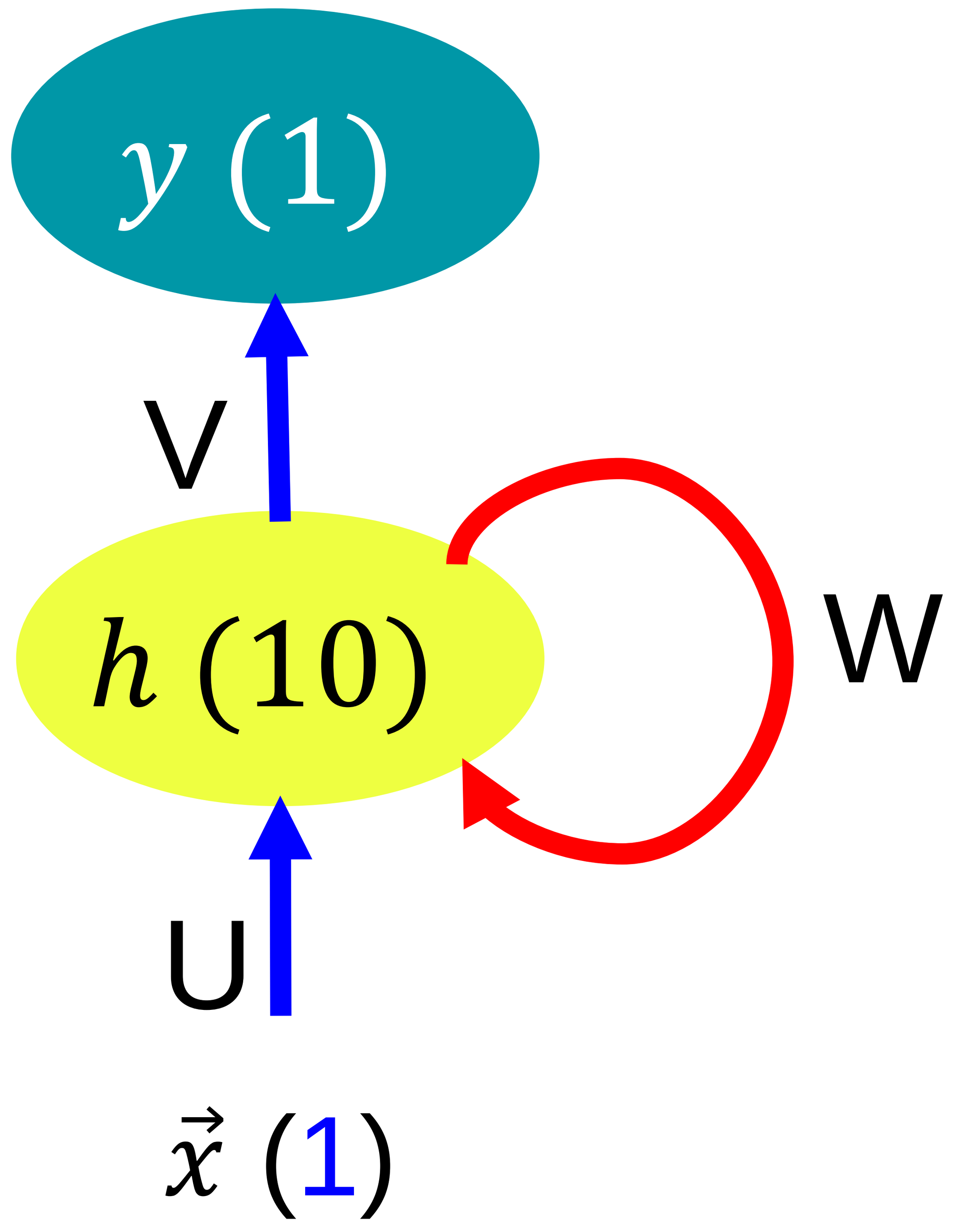


predict amazon stock closing prices, RNN 50 timestep

```
from keras import Sequential
from keras.layers import SimpleRNN, Dense
model = Sequential()
model.add(SimpleRNN(10, input_shape=(50,1)))
#simple recurrent layer, 10 neurons & process
50x1 sequences
model.add(Dense(1,activation='linear’)) #linear
output because this is a regression problem
```

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

Number of Parameter



U = 10 × (1+1) = 20
W = 10 × 10 = 100
V = 1 × (10+1) = 11
131

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, 10)                120
_________________________________________________________________
dense (Dense)                (None, 1)                 11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

Total parameter = (1+10+1)*10+(10+1)*1=131

Simple RNN:
U: matrix hidden neurons x (input dimension + 1)
W: matrix hidden neurons x hidden neurons
V: matrix output neurons x (hidden neurons+1)

Number of Parameter: Example 2

```
model = Sequential() #initialize model
model.add(SimpleRNN(64, input_shape=(50,1), return_sequences=True))#64 neurons
model.add(SimpleRNN(32, return_sequences=True))#32 neurons
model.add(SimpleRNN(16)) #16 neurons
model.add(Dense(8,activation='tanh'))
model.add(Dense(1,activation='linear'))
```

1) $\left.\begin{aligned} U &= 64 \times (1+1) = 128 \\ W &= 64 \times 64 = 4096 \end{aligned}\right\} 4224$

2) $\left.\begin{aligned} U &= 32 \times (64+1) = 2080 \\ W &= 32 \times 32 = 1024 \end{aligned}\right\} 3104$

3) $\left.\begin{aligned} U &= 16 \times (32+1) = 528 \\ W &= 16 \times 16 = 256 \end{aligned}\right\} 784$

4) $V = 8 \times (16+1) = 136$

5) $V = 1 \times (8+1) = 9$

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 50, 64)            4224
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 50, 32)            3104
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, 16)                784
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 8,257
Trainable params: 8,257
Non-trainable params: 0
_________________________________________________________________
```

= (1+64+1)*64=4224

= (64+32+1)*32=3104

= (32+16+1)*16=784

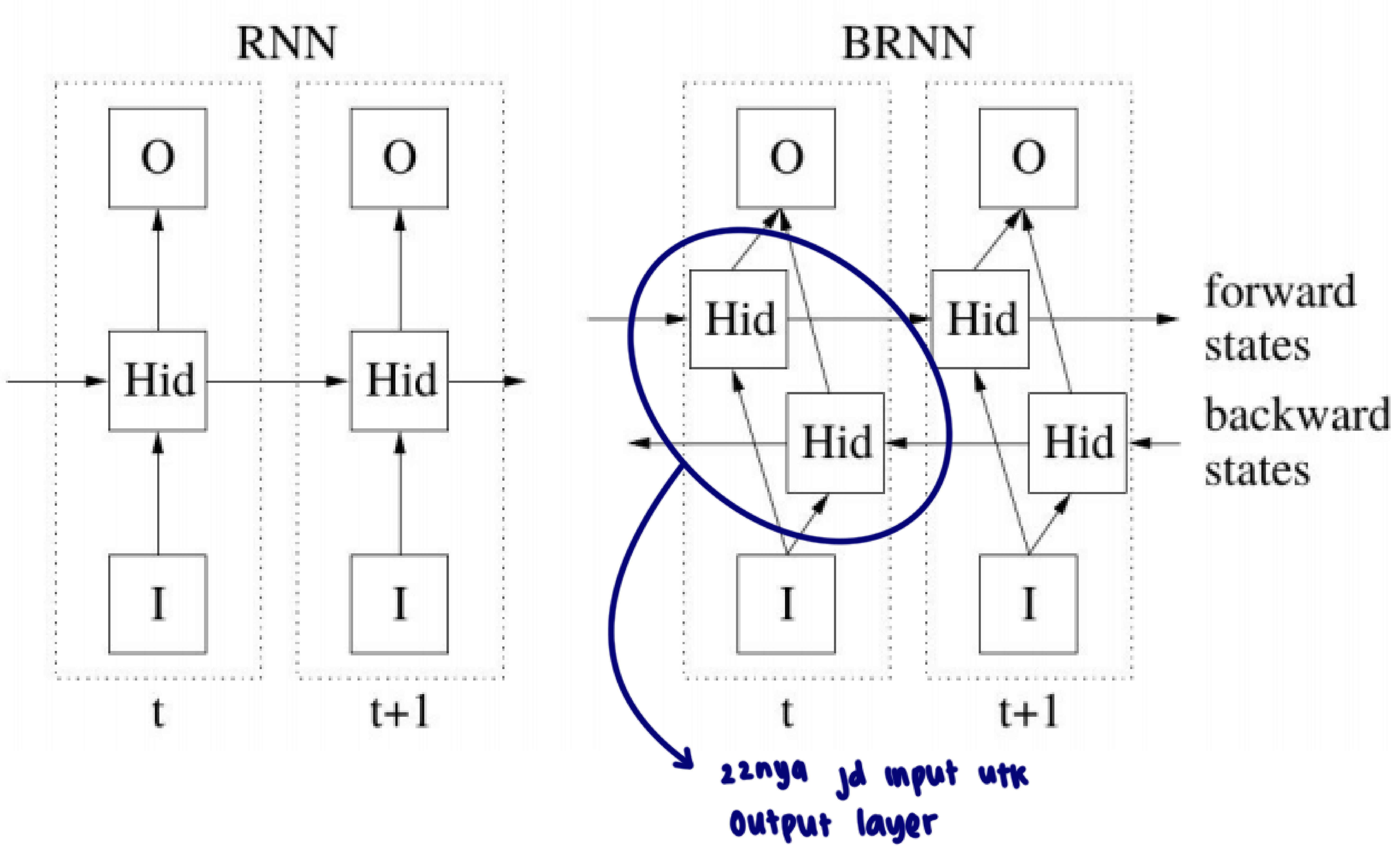
= (16+1)*8=136

= (8+1)*1=9

Total parameter = 8257

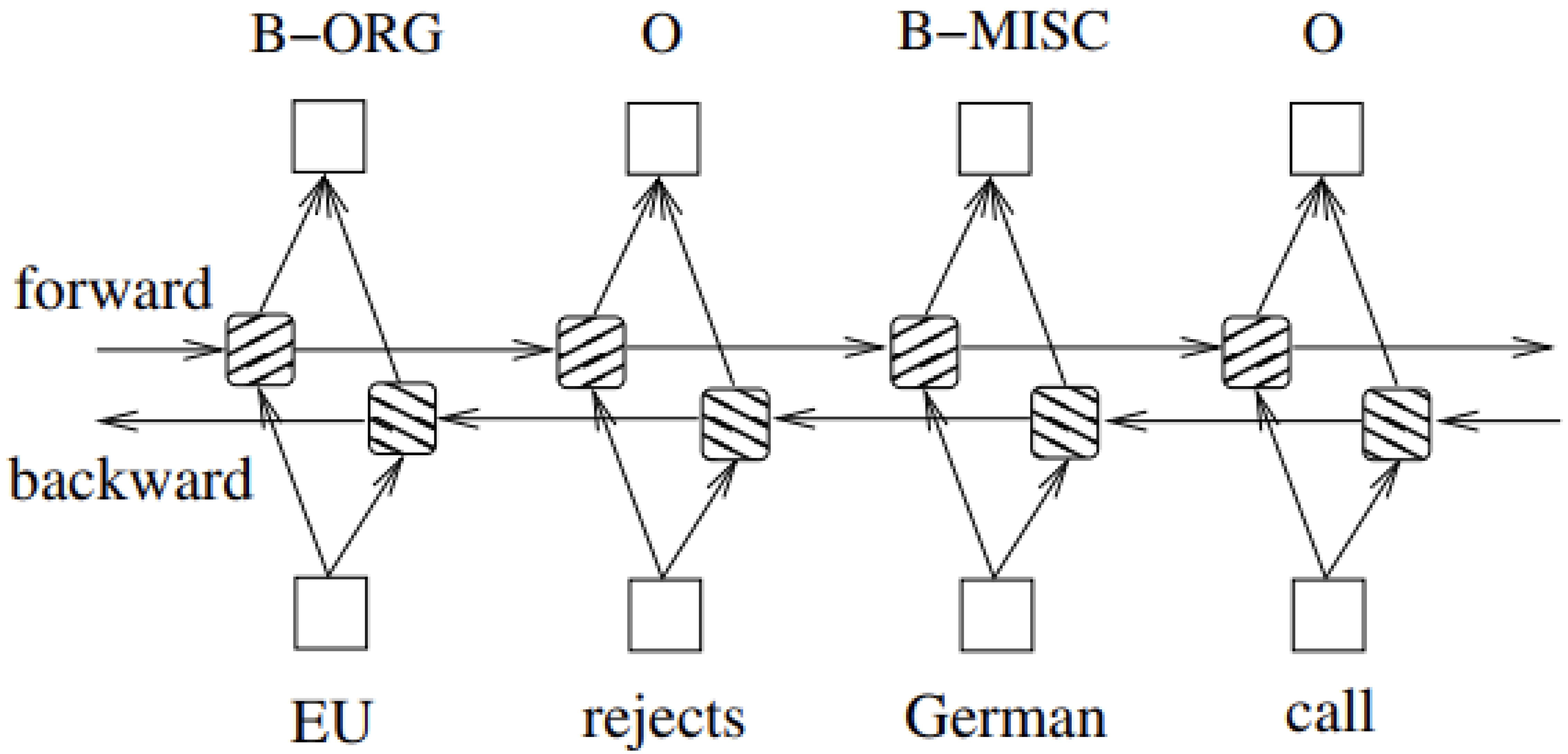
Bidirectional RNNs

In many applications we want to output a prediction of y (t) which may depend on the whole input sequence. E.g. co-articulation in speech recognition, right neighbors in POS tagging, etc. **Bidirectional RNNs** combine an RNN that moves forward through time beginning from the start of the sequence with another RNN that moves backward through time beginning from the end of the sequence.



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

Bidirectional RNNs for Information Extraction



https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/

Summary

Architecture: 1-to-n, n-to-1, n-to-n

Number of parameter RNN

Bidirectional RNN

LSTM

Thank you