DSE 3121 DEEP LEARNING

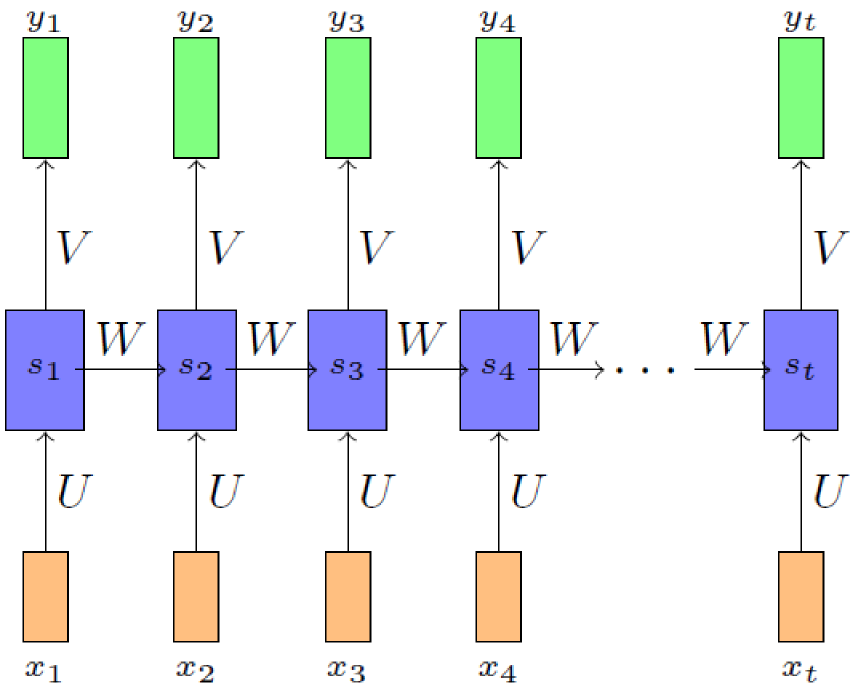
Advanced RNNs – LSTM, GRU, BRNN

Dr. Rohini Rao & Dr. Abhilash K Pai

Dept. of Data Science and Computer Applications

MIT Manipal

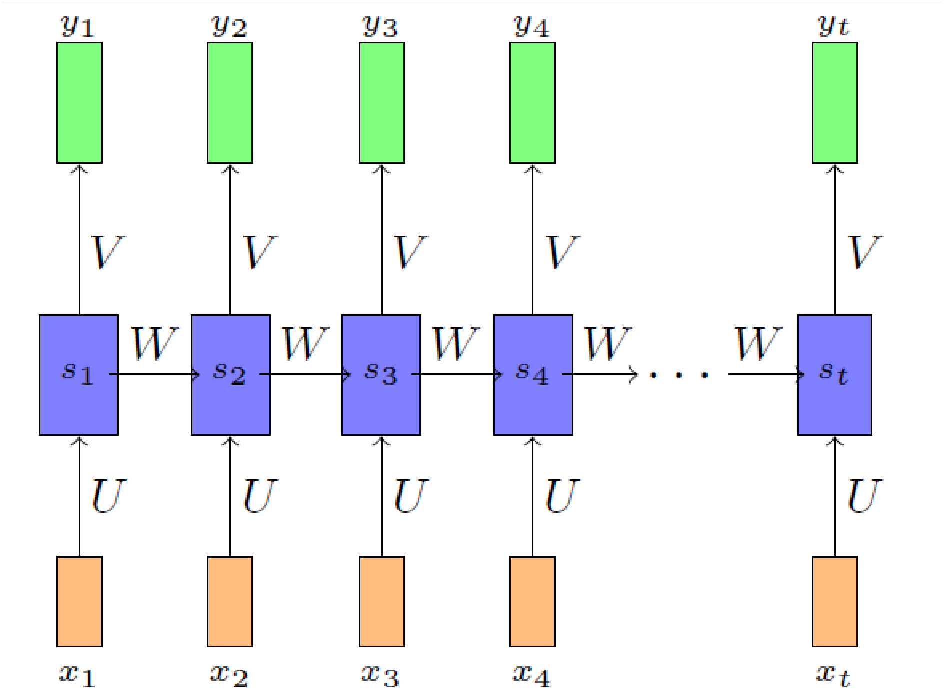
LSTM and GRU : Introduction



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- The state (s_i) of an RNN records information from all previous time steps.
- At each new timestep the old information gets morphed by the current input.
- After 't' steps the information stored at time step t-k (for some k < t) gets completely morphed.
- It would be impossible to extract the original information stored at time step t – k.

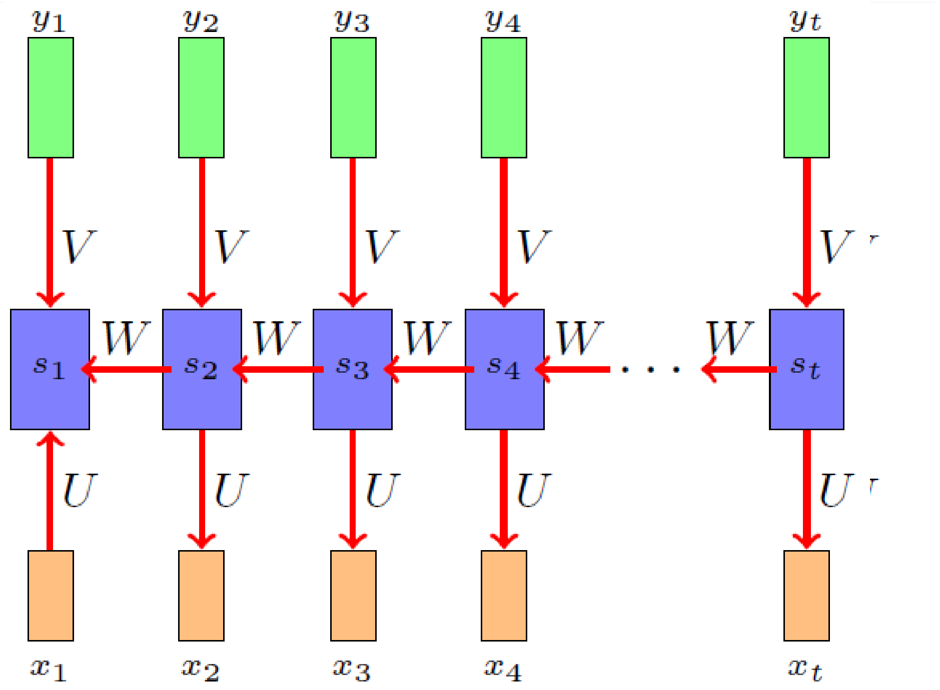
LSTM and GRU : Introduction



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- The state (s_i) of an RNN records information from all previous time steps.
- At each new timestep the old information gets morphed by the current input.
- After 't' steps the information stored at time step t-k (for some $k < t$) gets completely morphed.
- It would be impossible to extract the original information stored at time step $t - k$.
- Also, there is the Vanishing gradients problem!

LSTM and GRU : Introduction

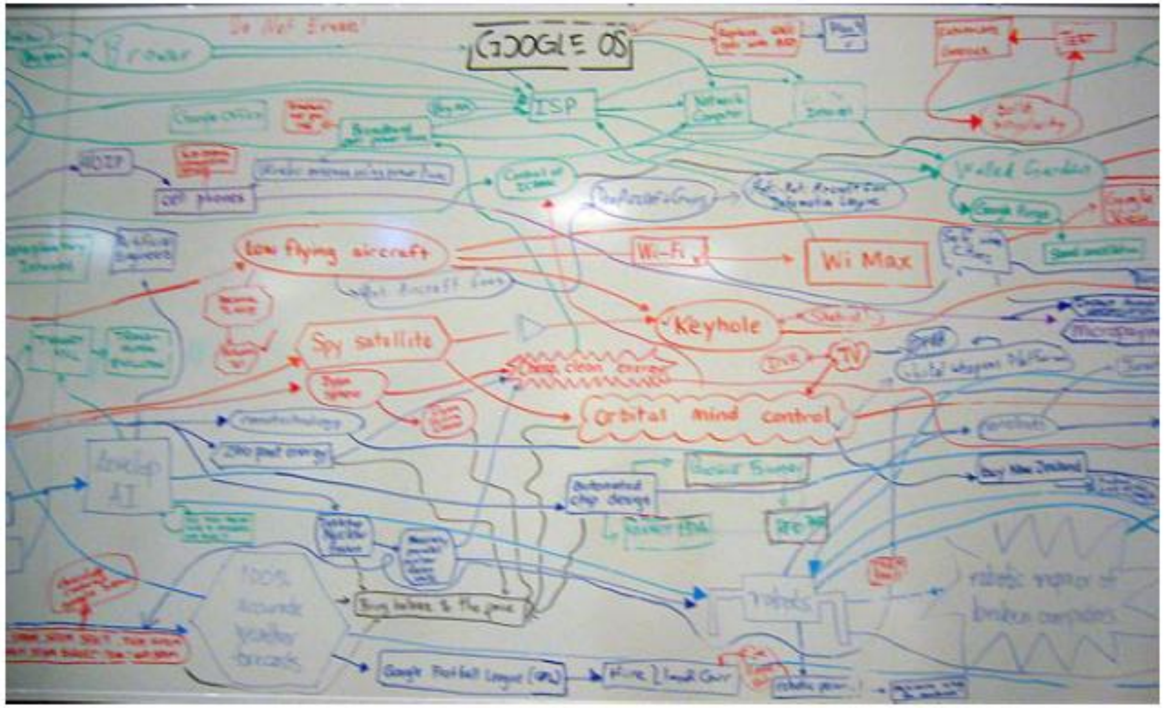


Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- The state (s_i) of an RNN records information from all previous time steps.
- At each new timestep the old information gets morphed by the current input.
- After 't' steps the information stored at time step t-k (for some k < t) gets completely morphed.
- It would be impossible to extract the original information stored at time step t – k.
- Also, there is the Vanishing gradients problem!

LSTM and GRU : Introduction

The white board analogy:



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- Consider a scenario where we have to evaluate the expression on a whiteboard:

  Evaluate "**ac(bd+a) + ad**"
  given that **a= 1, b= 3, c= 5, d=11**

- Normally, the evaluation in white board would look like:

$$\begin{aligned}
&\mathbf{ac = 5}\\
&\mathbf{bd = 33}\\
&\mathbf{bd + a = 34}\\
&\mathbf{ac(bd + a) = 170}\\
&\mathbf{ad = 11}\\
&\mathbf{ac(bd + a) + ad = 181}
\end{aligned}$$

- Now, if the white board has space to accommodate only 3 steps, the above evaluation cannot fit in the required space and would lead to loss of information.

LSTM and GRU : Introduction

Evaluate "$\mathbf{ac(bd+a) + ad}$"
given that $\mathbf{a=1, b=3, c=5, d=11}$

$$\mathbf{ac = 5}$$

$$\mathbf{bd = 33}$$

$$\mathbf{bd + a = 34}$$

$$\mathbf{ac(bd + a) = 170}$$

$$\mathbf{ad = 11}$$

$$\mathbf{ac(bd + a) + ad = 181}$$

- A solution is to do the following:

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

$\mathbf{ac = 5}$

$\mathbf{bd = 33}$

$\mathbf{bd + a = 34}$

$\mathbf{ac(bd + a) = 170}$

$\mathbf{ad = 11}$

$\mathbf{ac(bd + a) + ad = 181}$

- A solution is to do the following:
  - Selectively write:

    $\mathbf{ac = 5}$
    $\mathbf{bd = 33}$

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

ac = 5

bd = 33

bd + a = 34

ac(bd + a) = 170

ad = 11

ac(bd + a) + ad = 181

- A solution is to do the following:
  - Selectively write:

    ac = 5
    bd = 33
  - Selectively read:

    ac = 5
    bd = 33
    bd + a = 34

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

$\mathbf{ac = 5}$

$\mathbf{bd = 33}$

$\mathbf{bd + a = 34}$

$\mathbf{ac(bd + a) = 170}$

$\mathbf{ad = 11}$

$\mathbf{ac(bd + a) + ad = 181}$

- A solution is to do the following:
  - Selectively write:

    $\mathbf{ac = 5}$
    $\mathbf{bd = 33}$
  - Selectively read:

    $\mathbf{ac = 5}$
    $\mathbf{bd = 33}$
    $\mathbf{bd + a = 34}$

Now the board is full

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

ac = 5

bd = 33

bd + a = 34

ac(bd + a) = 170

ad = 11

ac(bd + a) + ad = 181

- A solution is to do the following:
  - Selectively write:
    ac = 5
    bd = 33
  - Selectively read:
    ac = 5
    bd = 33
    bd + a = 34

    Now the board is full
  - So, Selectively forget:
    ac = 5

    bd + a = 34

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

ac = 5

bd = 33

bd + a = 34

ac(bd + a) = 170

ad = 11

ac(bd + a) + ad = 181

- A solution is to do the following:
  - Selectively write:

    ac = 5
    bd = 33
  - Selectively read:

    ac = 5
    bd = 33
    bd + a = 34

    Now the board is full
  - So, Selectively forget:

    ac = 5
    ac(bd + a) = 170
    bd + a = 34

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

ac = 5

bd = 33

bd + a = 34

ac(bd + a) = 170

ad = 11

ac(bd + a) + ad = 181

- A solution is to do the following:
  - Selectively write:

    ac = 5
    bd = 33
  - Selectively read:

    ac = 5
    bd = 33
    bd + a = 34

    Now the board is full
  - So, Selectively forget:

    ac = 5
    ac(bd + a) = 170
    ad = 11

LSTM and GRU : Introduction

Evaluate "**ac(bd+a) + ad**"
given that **a= 1, b= 3, c= 5, d=11**

ac = 5

bd = 33

bd + a = 34

ac(bd + a) = 170

ad = 11

ac(bd + a) + ad = 181

- A solution is to do the following:
  - Selectively write:

    ac = 5
    bd = 33
  - Selectively read:

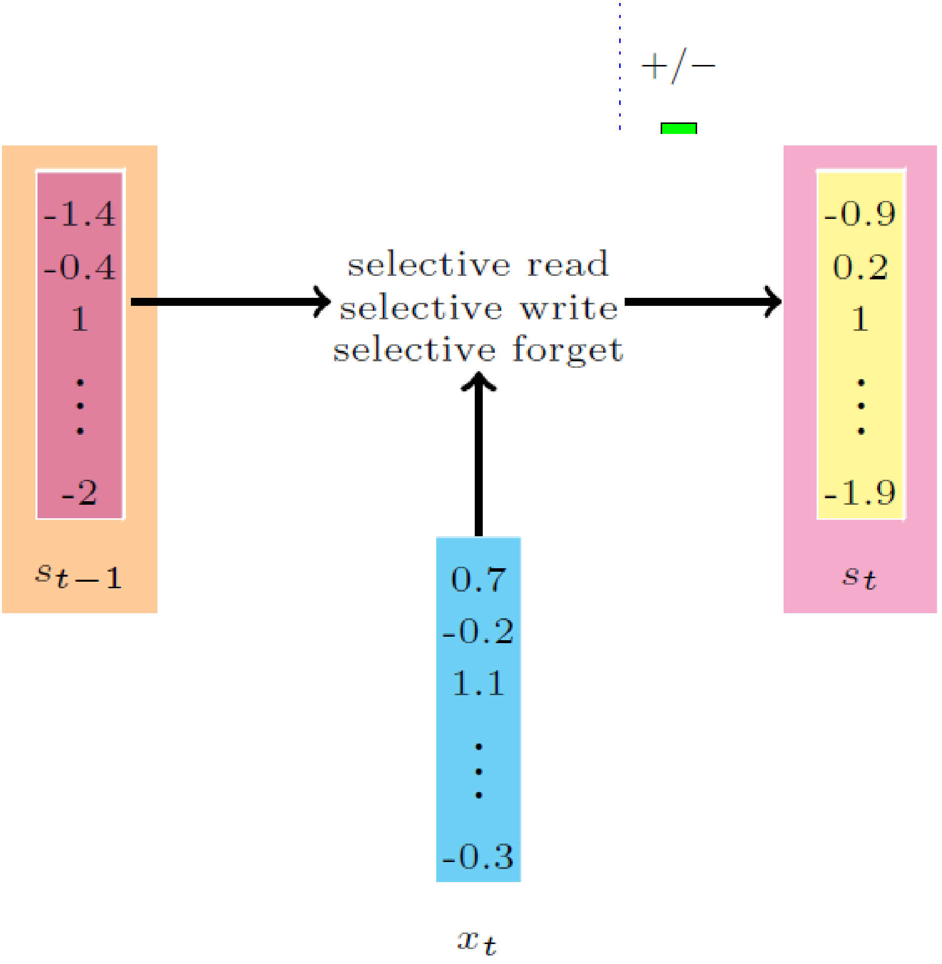
    ac = 5
    bd = 33
    bd + a = 34

    Now the board is full
  - So, Selectively forget:

    ac(bd + a) + ad = 181
    ac(bd + a) = 170
    ad = 11

Since the RNN also has a finite state size, we need to figure out a way to allow it to selectively read, write and forget

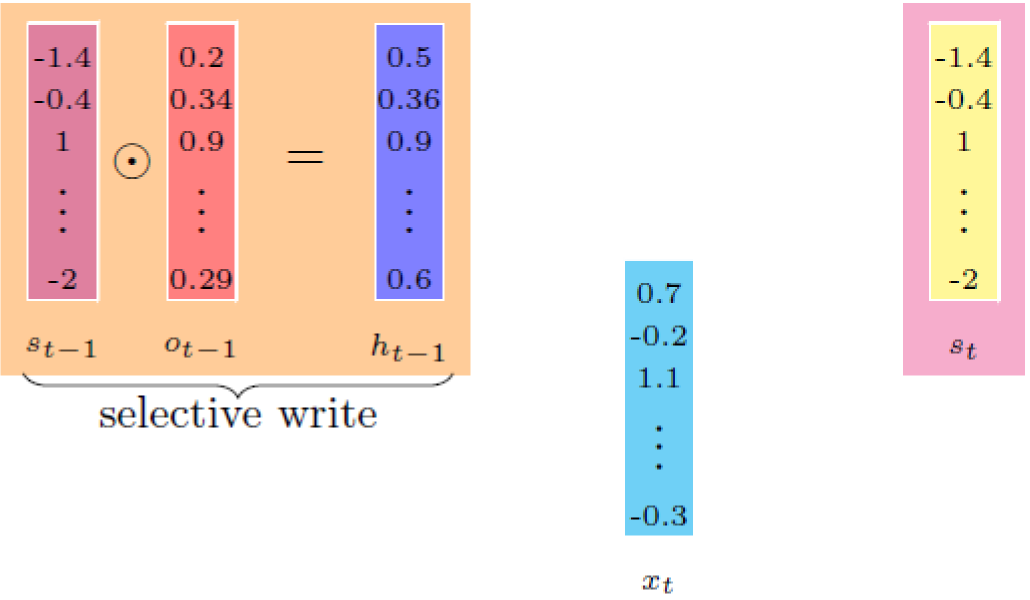
LSTM and GRU : Introduction



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- RNN reads the document from left to right and after every word updates the state.
- By the time we reach the end of the document the information obtained from the first few words is completely lost.
- In our improvised network, ideally, we would like to:
  - **Forget** the information added by stop words (a, the, etc.)
  - **Selectively read** the information added by previous sentiment bearing words (awesome, amazing, etc.)
  - **Selectively write** new information from the current word to the state.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras



The RNN has to learn o_{t-1} along with other parameters (W,U,V)

$$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$$
$$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

New parameters to be learned are: **W_o, U_o, b_o**

O_t is called the output gate as it decides how much to pass (write) to the next time step.

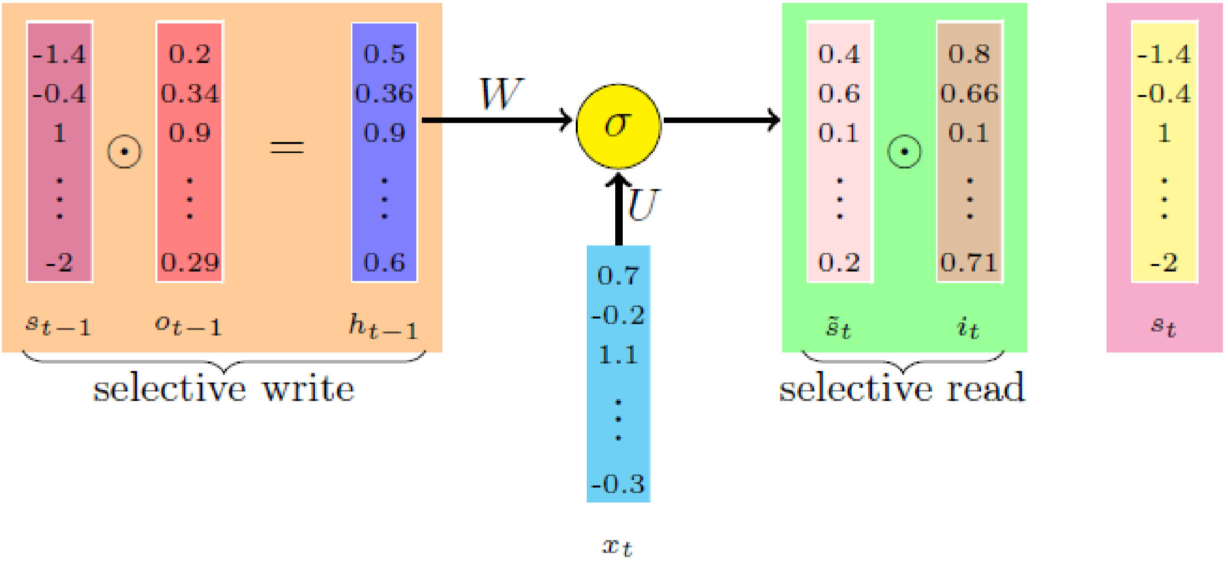
Selectively write:

- In an RNN, the state s_t is defined as follows:

$$s_t = \sigma(W s_{t-1} + U x_t) \ (ignoring\ bias)$$

- Instead of passing s_{t-1} as it is, we need to pass (**write**) only some portions of it.
- To do this, we introduce a vector $\mathbf{o_{t-1}}$ which decides what fraction of each element of s_{t-1} should be passed to the next state.
- Each element of o_{t-1} (restricted to be between 0 and 1) gets multiplied with s_{t-1}
- How does RNN know what fraction of the state to pass on?

LSTM and GRU : Introduction



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Selectively read:

- We will now use h_{t-1} and x_t to compute the new state at the time step t :

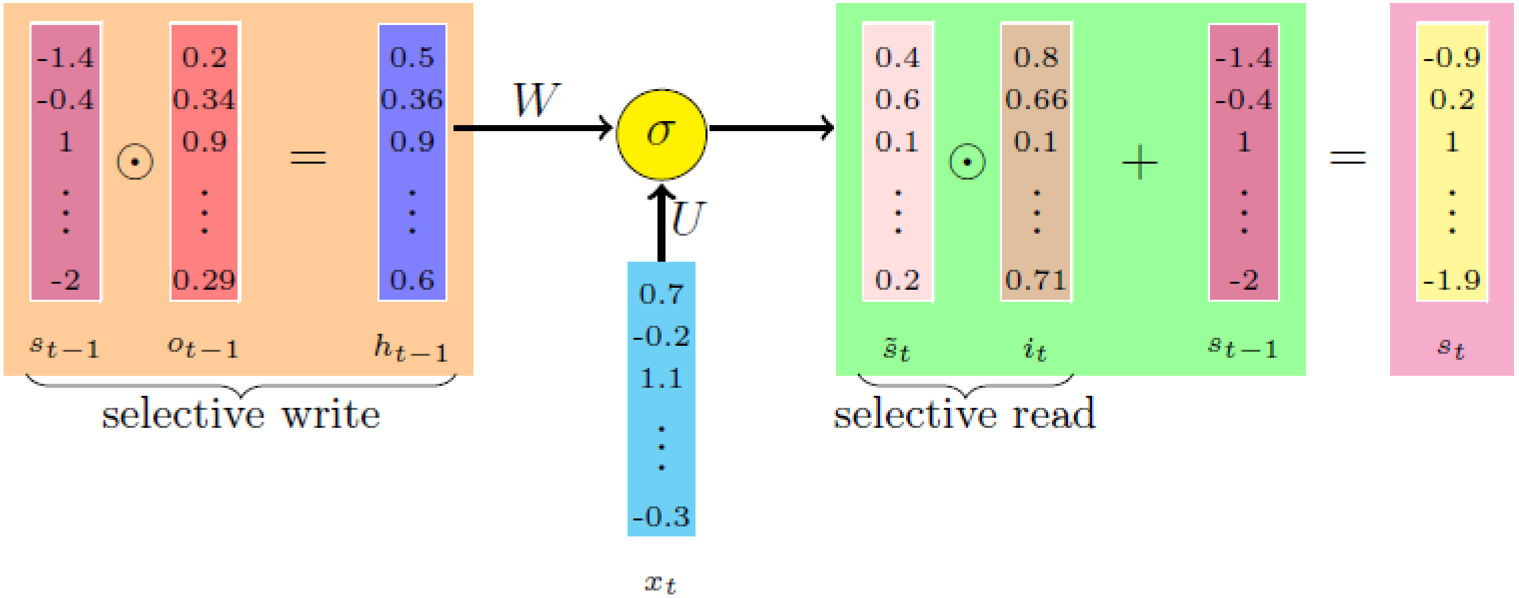
$$\tilde{s}_t = \sigma(Wh_{t-1} + Ux_t + b)$$

- Again, to pass only useful information from $\tilde{s}_t$ to s_t, we selectively read from it before constructing the new cell state.

- To do this we introduce another gate called as the input gate:

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

- And use $i_t \odot \tilde{s}_t$ to selectively read the information.



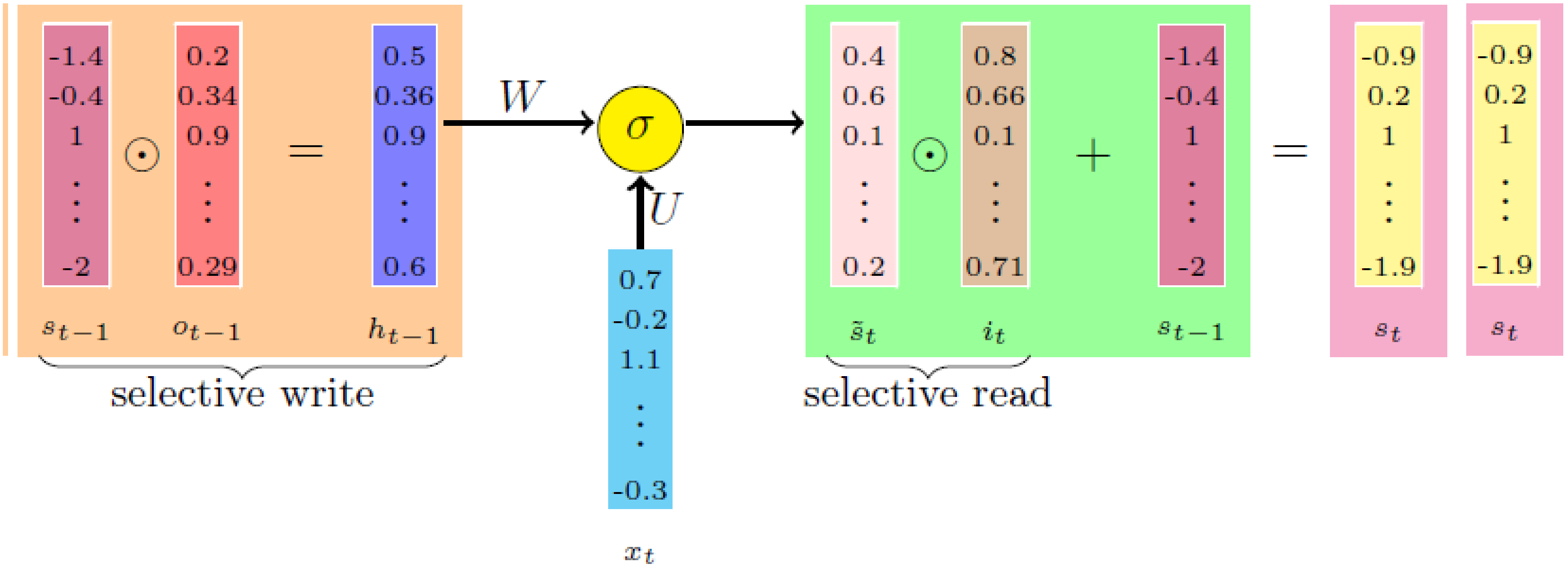
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Selectively forget

- How do we combine s_{t-1} and $\tilde{s}_t$ to get the new state?

$$s_t = s_{t-1} + i_t \odot \tilde{s}_t$$

LSTM and GRU : Introduction



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Selectively forget

- How do we combine s_{t-1} and $\tilde{s}_t$ to get the new state?

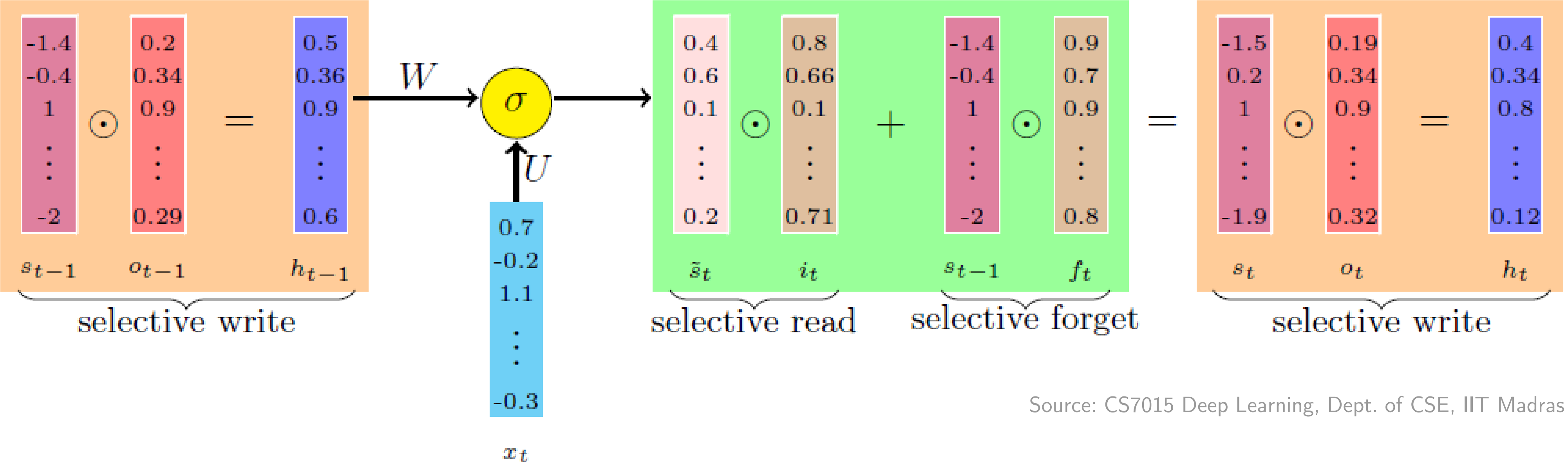
$$s_t = s_{t-1} + i_t \odot \tilde{s}_t$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

- But we may not want to use the whole of s_{t-1} but forget some parts of it.
- To do this a forget gate is introduced:

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

LSTM (Long Short-Term Memory)



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Gates:

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

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

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

States:

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$

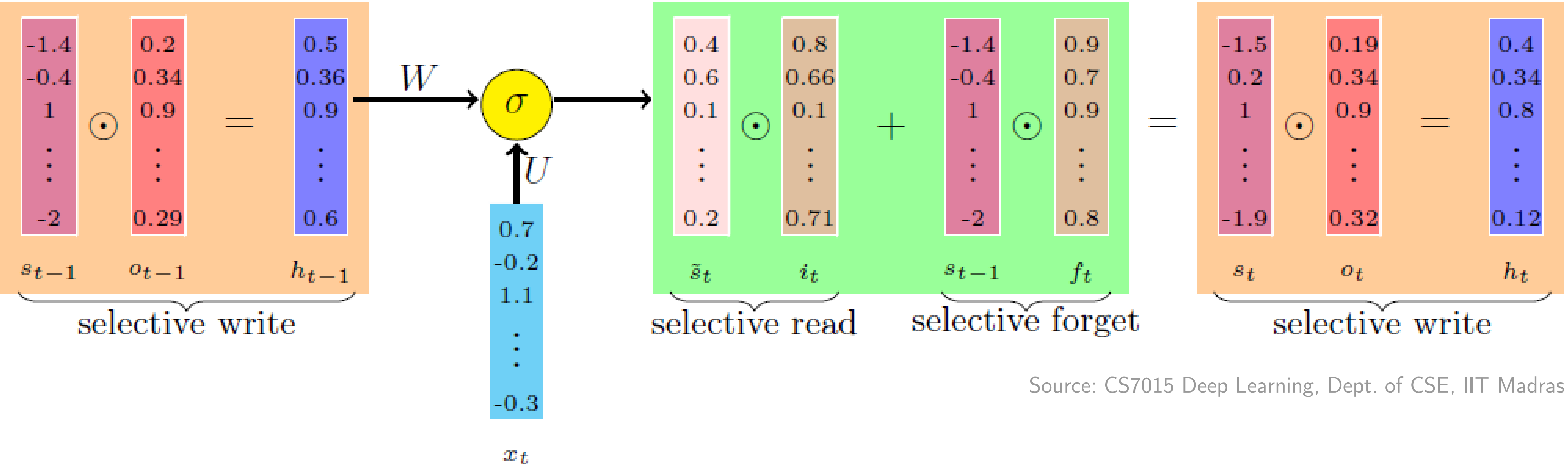
Long-term memory

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

$$h_t = o_t \odot \sigma(s_t) \text{ and } rnn_{out} = h_t$$

Short-term memory

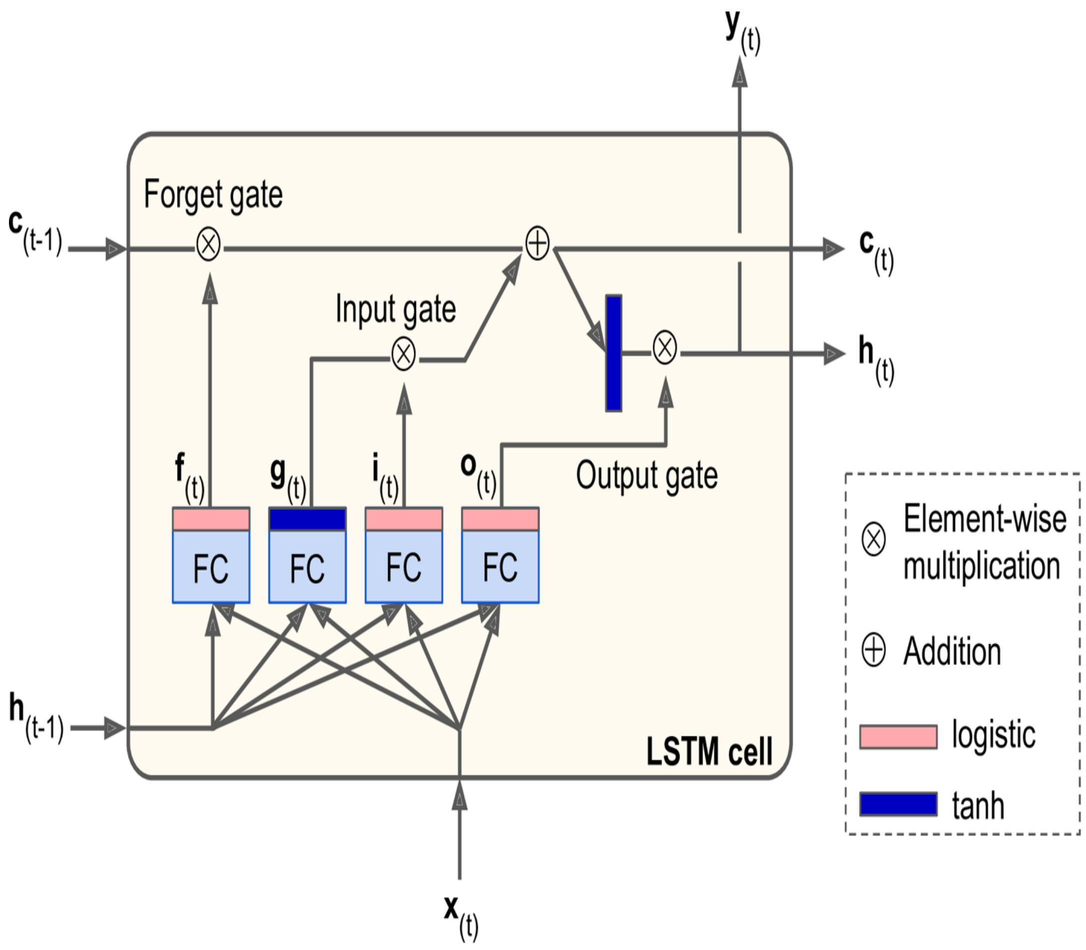
LSTM (Long Short-Term Memory)



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- LSTM has many variants which include different number of gates and also different arrangement of gates.
- A popular variant of LSTM is the Gated Recurrent Unit (GRU).

LSTM Cell

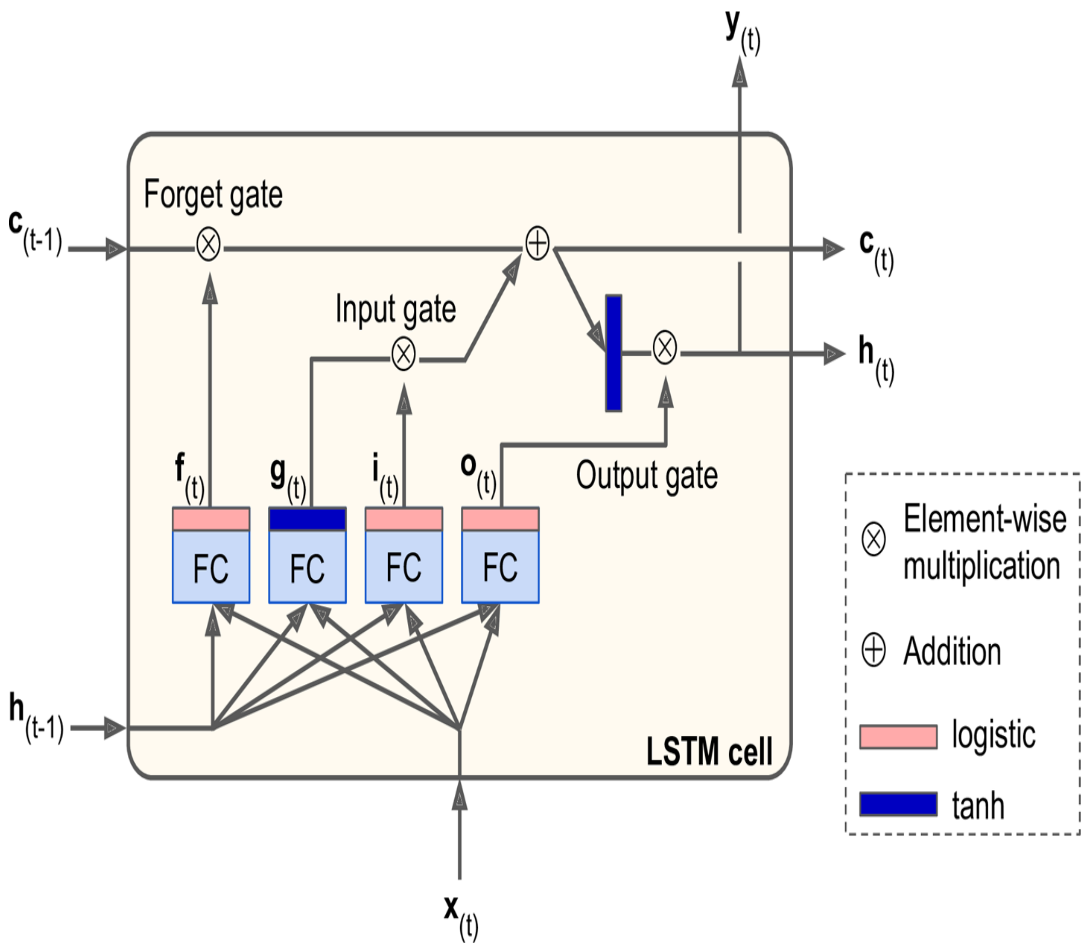


- Note: c_t is same as s_t

- Neuron called a Cell
- FC are fully connected layers
- Long Term state $c_{(t-1)}$ traverses through forget gate forgetting some memories and adding some new memories
- Long term state $c_{(t-1)}$ is passed through tanh and then filtered by an output gate, which produces short term state $h_{(t)}$
- Update gate- $g_{(t)}$ takes current input $x_{(t)}$ and previous short term state $h_{(t-1)}$
- Important parts of output $g_{(t)}$ goes to long term state

Source: Aurélien Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2019.

LSTM Cell



- **Gating Mechanism-** regulates information that the network stores

Other 3 layers are gate controllers

- **Forget gate $f_{(t)}$** controls which part of the long-term state should be erased
- **Input gate $i_{(t)}$** controls which part of $g_{(t)}$ should be added to long term state
- **Output gate $o_{(t)}$** controls which parts of long term state should be read and output at this time state
    - both to $h_{(t)}$ and to $y_{(t)}$

Source: Aurélien Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2019.

LSTM computations

$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma({\mathbf{W}_{xi}}^T \cdot \mathbf{x}_{(t)} + {\mathbf{W}_{hi}}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
\mathbf{f}_{(t)} &= \sigma({\mathbf{W}_{xf}}^T \cdot \mathbf{x}_{(t)} + {\mathbf{W}_{hf}}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
\mathbf{o}_{(t)} &= \sigma({\mathbf{W}_{xo}}^T \cdot \mathbf{x}_{(t)} + {\mathbf{W}_{ho}}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
\mathbf{g}_{(t)} &= \tanh({\mathbf{W}_{xg}}^T \cdot \mathbf{x}_{(t)} + {\mathbf{W}_{hg}}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
\end{aligned}
$$

An LSTM cell can learn to recognize an important input (role of the input gate), store it in long term state, preserve it for as long as possible it is needed (role of forget gate), and extract it whenever it is needed.

$\mathbf{W}_{xi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{xo}$, $\mathbf{W}_{xg}$ are the weight matrices of each of the four layers for their connection to the input vector $\mathbf{x}_{(t)}$.

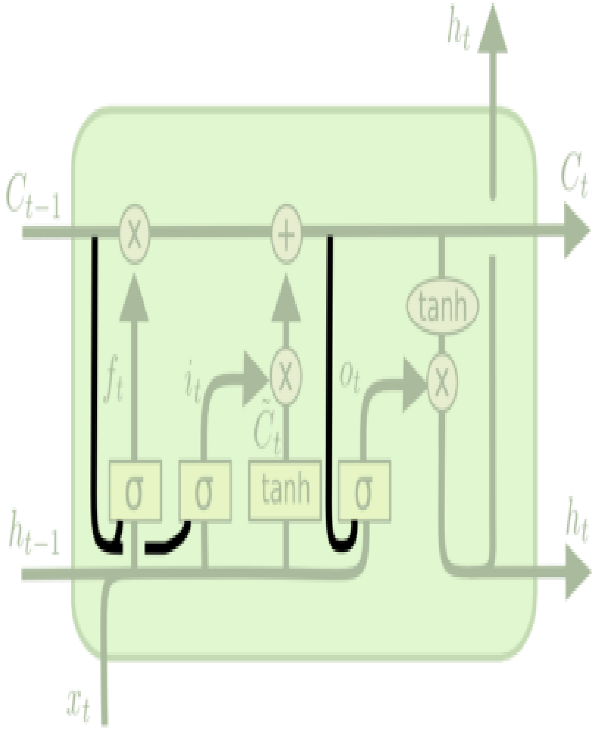
$\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho}$, and $\mathbf{W}_{hg}$ are the weight matrices of each of the four layers for their connection to the previous short-term state $\mathbf{h}_{(t-1)}$.

$\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, and $\mathbf{b}_g$ are the bias terms for each of the four layers.

Source: Aurélien Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2019.

LSTM with Peephole connections (Felix Gers and Jürgen Schmidhuber in 2000)

- In LSTM cell, the gate controllers get input $\mathbf{x}_{(t)}$ and $\mathbf{h}_{(t-1)}$.
- Can be given more context by letting them peek at the long-term state as well
- LSTM variant with extra connections called ***peephole connections***
  - previous long-term state $\mathbf{c}_{(t-1)}$ is added as an input to the controllers of the forget gate and the input gate
  - current long-term state $\mathbf{c}_{(t)}$ is added as input to the controller of the output gate



$$f_t = \sigma\left(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [C_t, h_{t-1}, x_t] + b_o\right)$$

Gated Recurrent Unit (GRU)

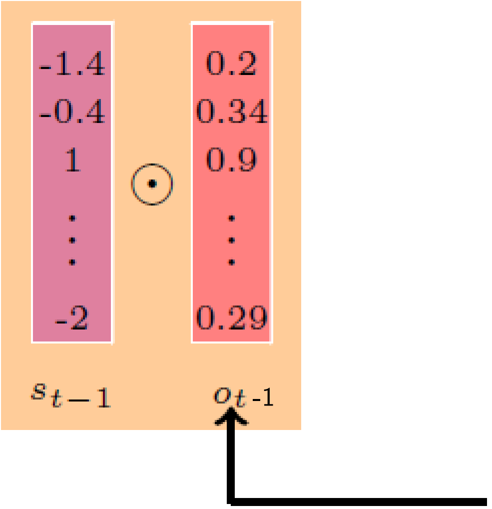
$$\begin{bmatrix} -1.4 \\ -0.4 \\ 1 \\ \vdots \\ -2 \end{bmatrix}$$

s_{t-1}

Gates:

States:

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras



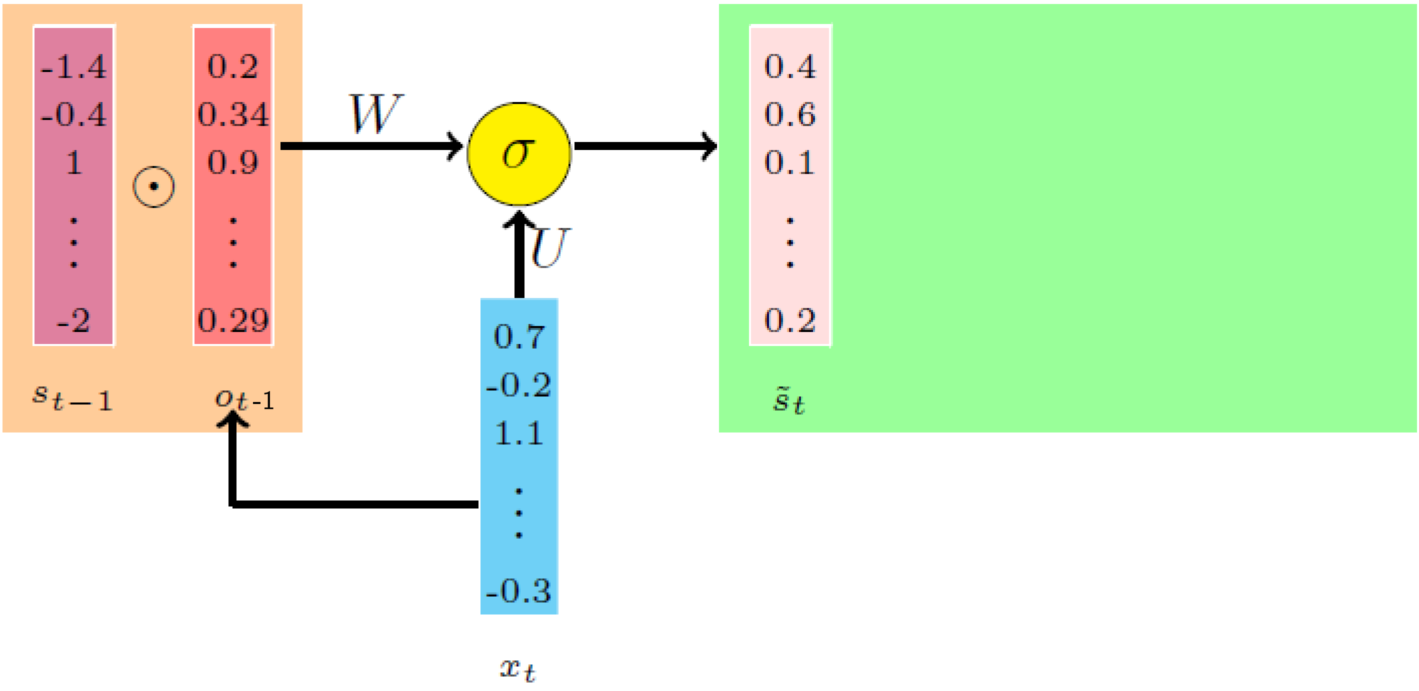
Gates:

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

States:

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Gated Recurrent Unit (GRU)



Gates:

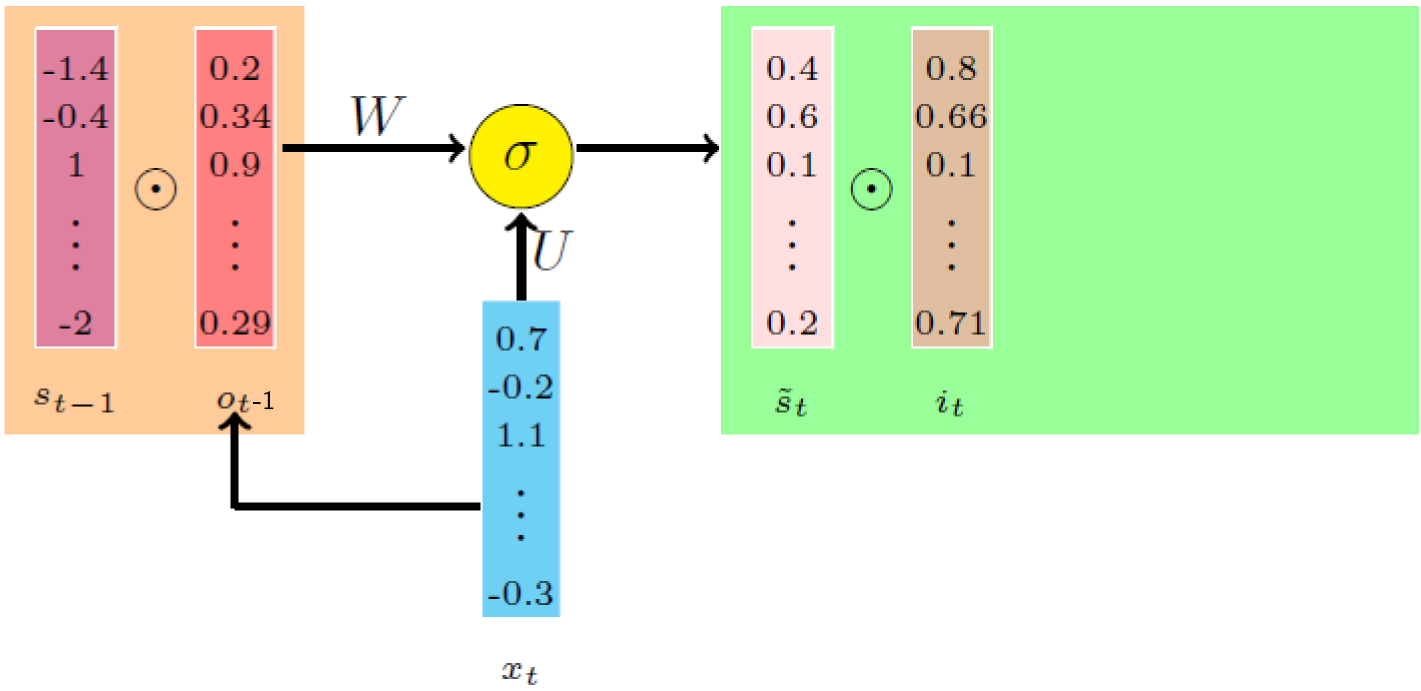
$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Gated Recurrent Unit (GRU)



Gates:

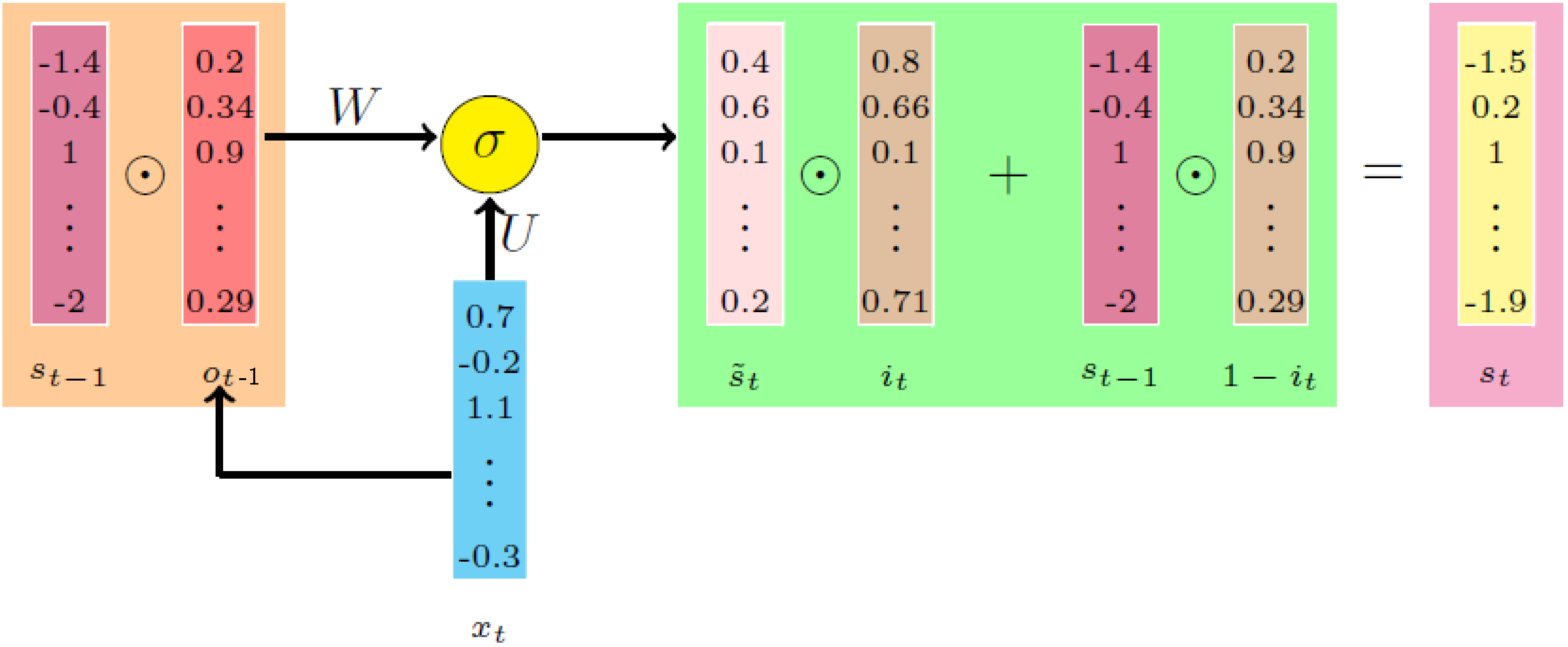
$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + Ux_t + b)$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Gated Recurrent Unit (GRU)



Gates:

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

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

States:

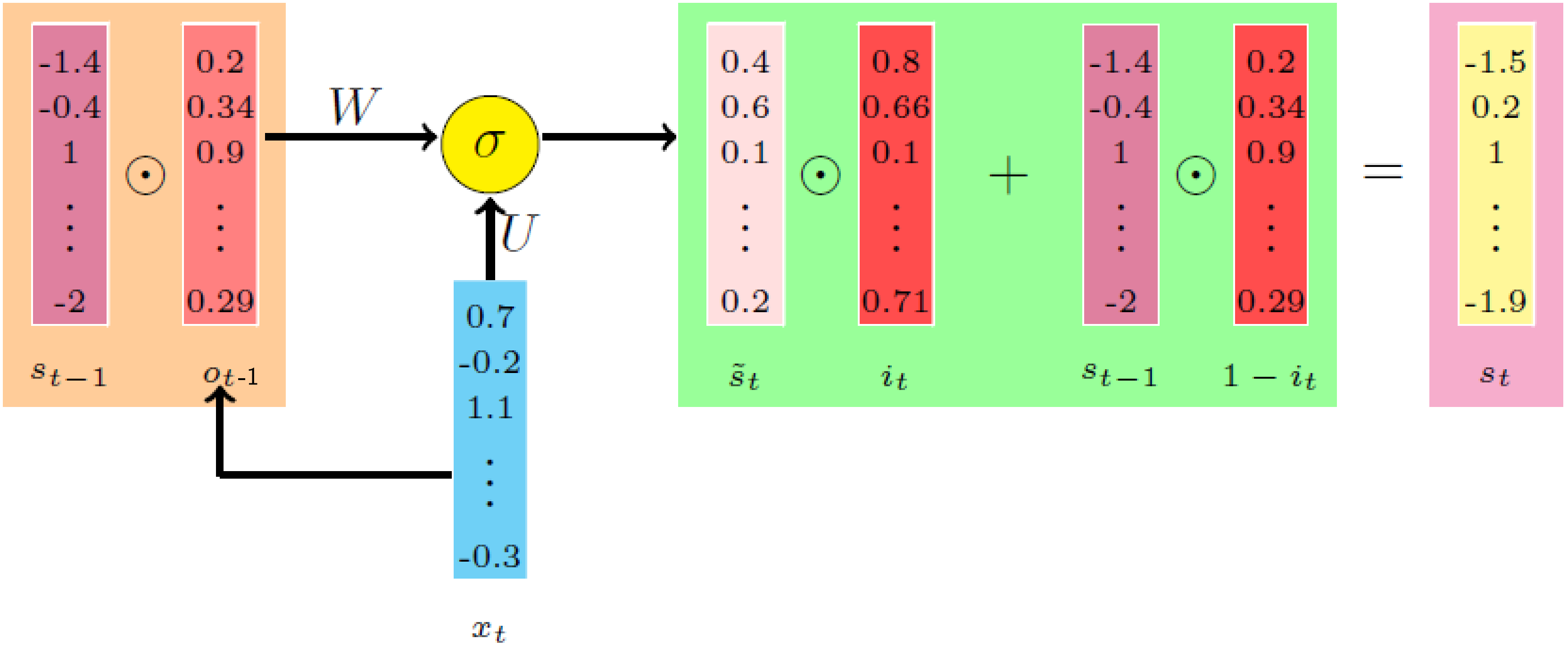
$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$

$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Gated Recurrent Unit (GRU)

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras



Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

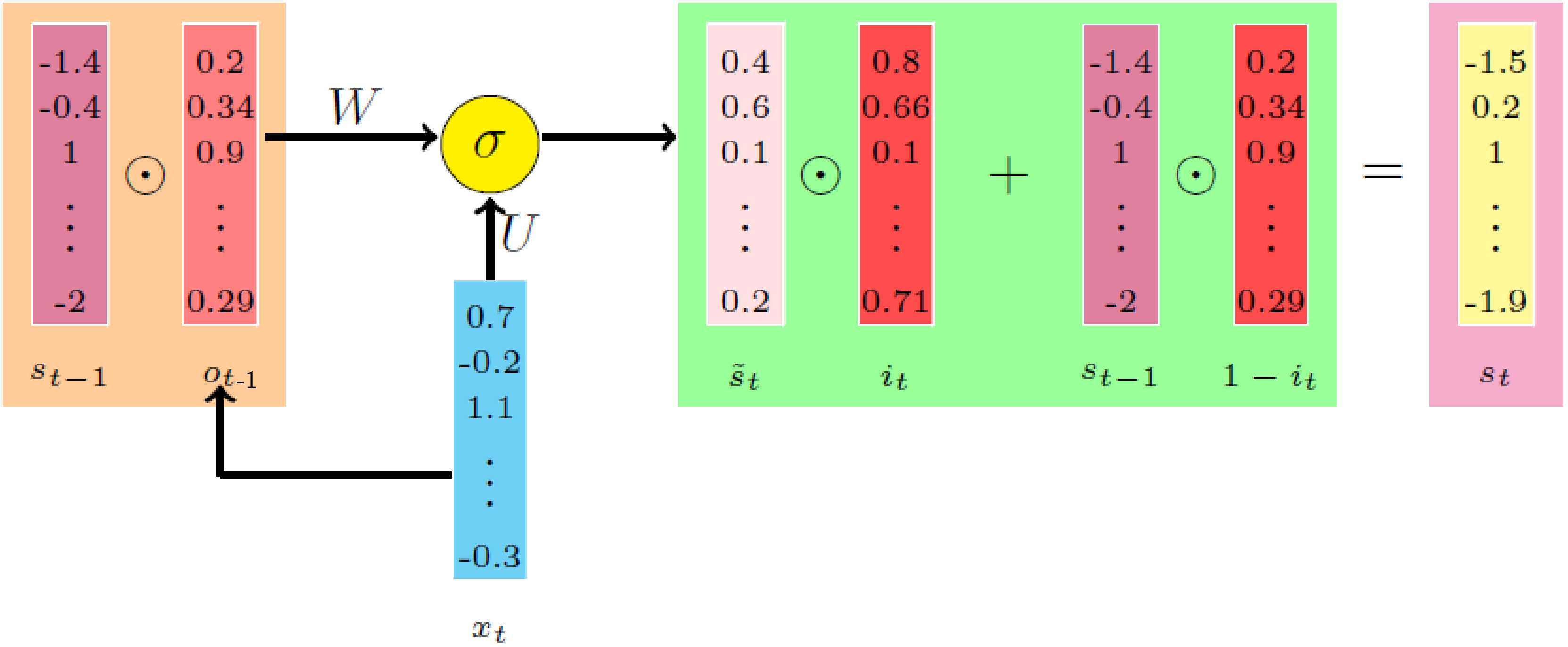
States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = \boxed{(1 - i_t)} \odot s_{t-1} + \boxed{i_t} \odot \tilde{s}_t$$

No explicit forget gate (the forget gate and input gates are tied)

Gated Recurrent Unit (GRU)

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras



Gates:

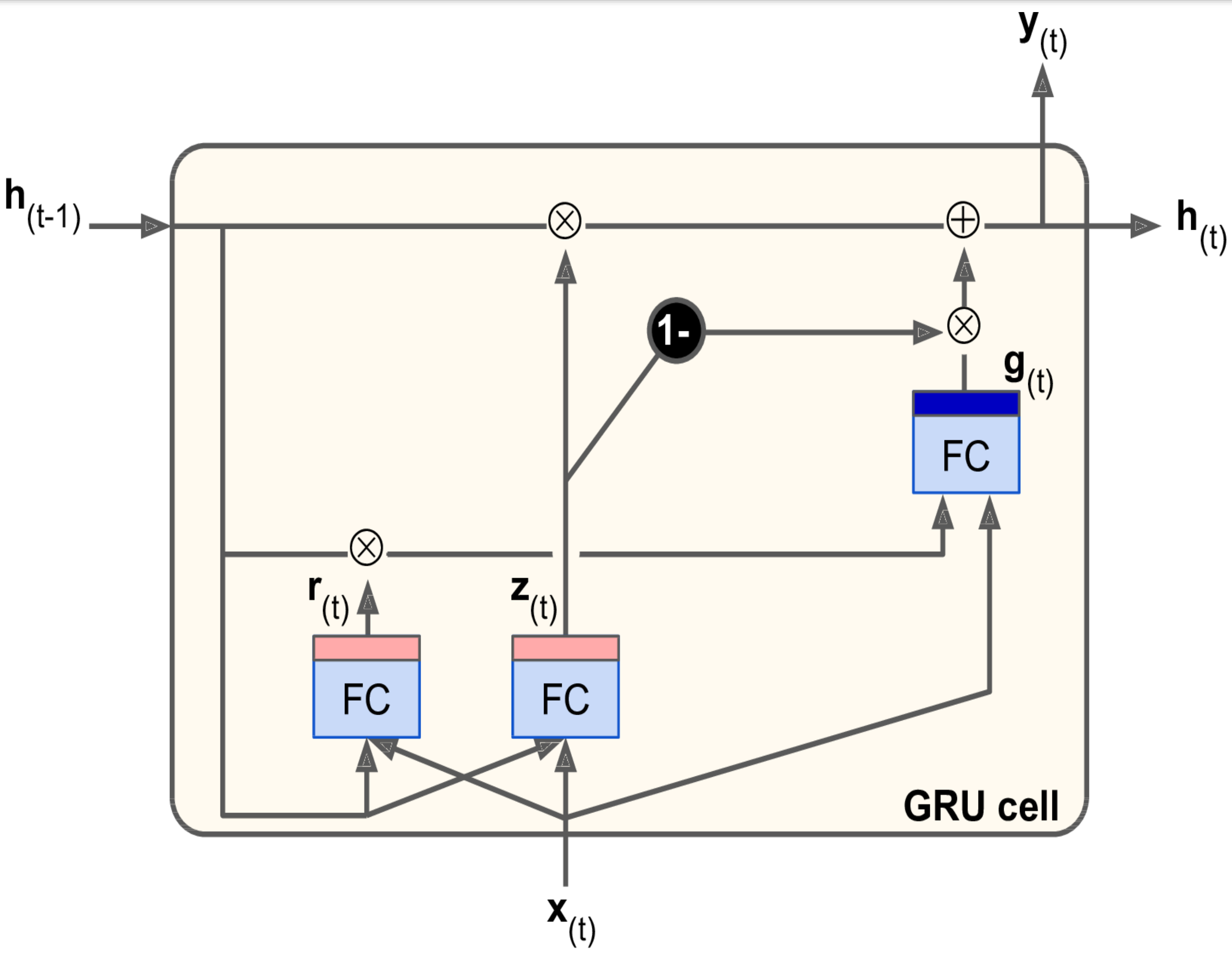
$$o_t = \sigma(W_o \boxed{s_{t-1}} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i \boxed{s_{t-1}} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = \boxed{(1 - i_t)} \odot s_{t-1} + \boxed{i_t} \odot \tilde{s}_t$$

The gates depend directly on s_{t-1} and not the intermediate h_{t-1} as in the case of LSTMs

Gated Recurrent Unit CELL (Kyunghyun Cho et al, 2014)



The main simplifications of LSTM are:

- Both state vectors (short and long term) are merged into a single vector $\mathbf{h}_{(t)}$.
- **Gate controller $\mathbf{z}_{(t)}$** controller controls both the forget gate and the input gate.
- If the gate controller outputs
  - 1, the forget gate is open and the input gate is closed.
  - 0, the opposite happens
  - whenever a memory must be written, the location where it will be stored is erased first.
- No output gate, the full state vector is output at every time step.
- **Reset gate controller $\mathbf{r}_{(t)}$** that controls which part of the previous state will be shown to the main layer $\mathbf{g}_{(t)}$.

Source: Aurélien Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2019.

LSTM vs GRU computation

$$\begin{aligned}
\mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{i}) \\
\mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{f}) \\
\mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{o}) \\
\mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{g}) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
\end{aligned}$$

$$\begin{aligned}
\mathbf{z}_{(t)} &= \sigma(\mathbf{W}_{xz}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hz}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{z}) \\
\mathbf{r}_{(t)} &= \sigma(\mathbf{W}_{xr}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hr}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{r}) \\
\mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot (\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}) + \mathbf{b}_{g}) \\
\mathbf{h}_{(t)} &= \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + (1 - \mathbf{z}_{(t)}) \otimes \mathbf{g}_{(t)}
\end{aligned}$$

- GRU Performance is good but may have a slight dip in the accuracy
- But lesser number of trainable parameters which makes it advantageous to use

Avoiding vanishing gradients with LSTMs: Intuition

- During forward propagation the gates control the flow of information. They prevent any irrelevant information from being written to the state. Similarly during backward propagation, they control the flow of gradients.

- It is easy to see that during backward pass the gradients will get multiplied by the gate.
- If the state at time t-1 did not contribute much to the state at time t, then during backpropagation the gradients flowing into s_{t-1} will vanish, which is fine. $||f_t|| \to 0$ and $||o_{t-1}|| \to 0$

- The key difference from vanilla RNNs is that the flow of information and gradients is controlled by the gates which ensure that the gradients vanish only when they should.

Avoiding vanishing gradients with LSTMs: Intuition

- Also, we can argue that, in the case of LSTMs there exists at least one path through which the gradients can flow effectively (and hence no vanishing gradients)

LSTM Equations

$$o_k = \sigma(W_o h_{k-1} + U_o x_k + b_o)$$

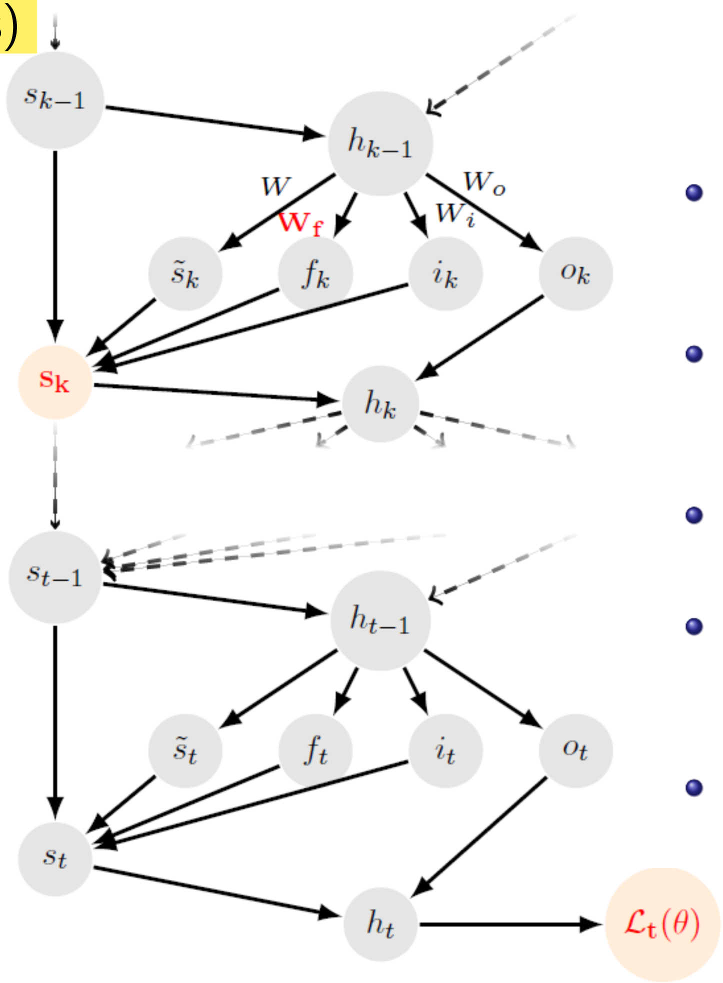
$$i_k = \sigma(W_i h_{k-1} + U_i x_k + b_i)$$

$$f_k = \sigma(W_f h_{k-1} + U_f x_k + b_f)$$

$$\tilde{s_k} = \sigma(W h_{k-1} + U x_k + b)$$

$$s_k = f_k \odot s_{k-1} + i_k \odot \tilde{s_k}$$

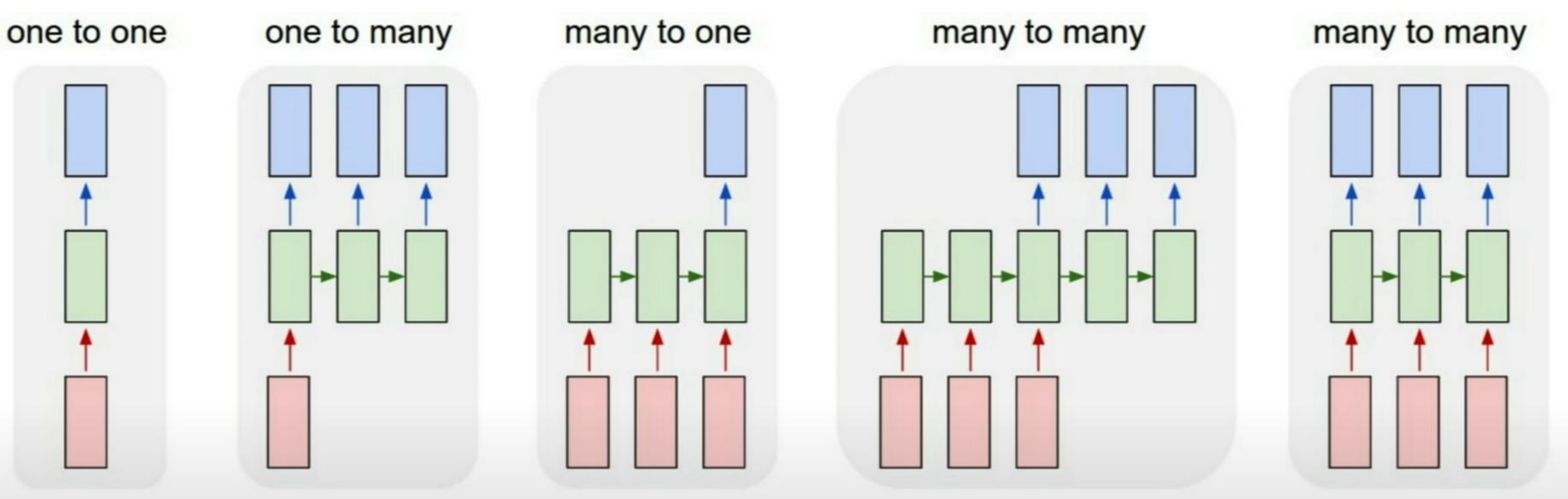
$$h_k = o_k \odot \sigma(s_k)$$



- It is sufficient to show that $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_k}$ does not vanish (because if this does not vanish we can reach W_f through s_k)
- First, we observe that there are multiple paths from $\mathcal{L}_t(\theta)$ to s_k (you just need to reverse the direction of the arrows for backpropagation)
- For example, there is one path through s_{k+1}, another through h_k
- Further, there are multiple paths to reach to h_k itself (as should be obvious from the number of outgoing arrows from h_k)
- So at this point just convince yourself that there are many paths from $\mathcal{L}_t(\theta)$ to s_k

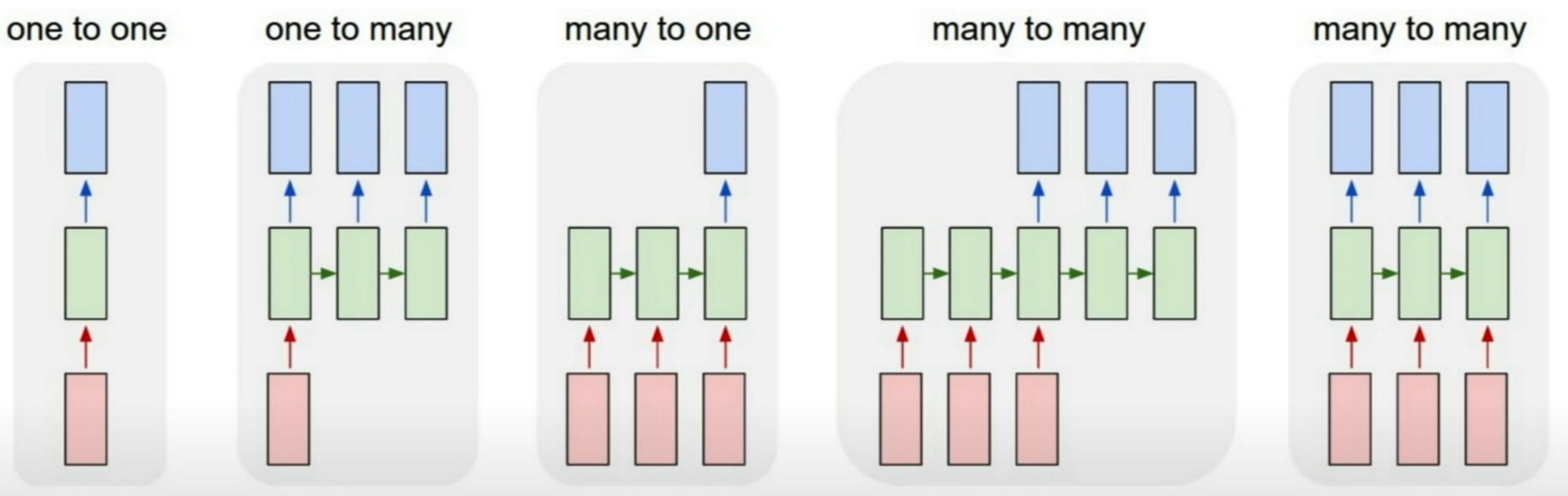
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Different RNNs



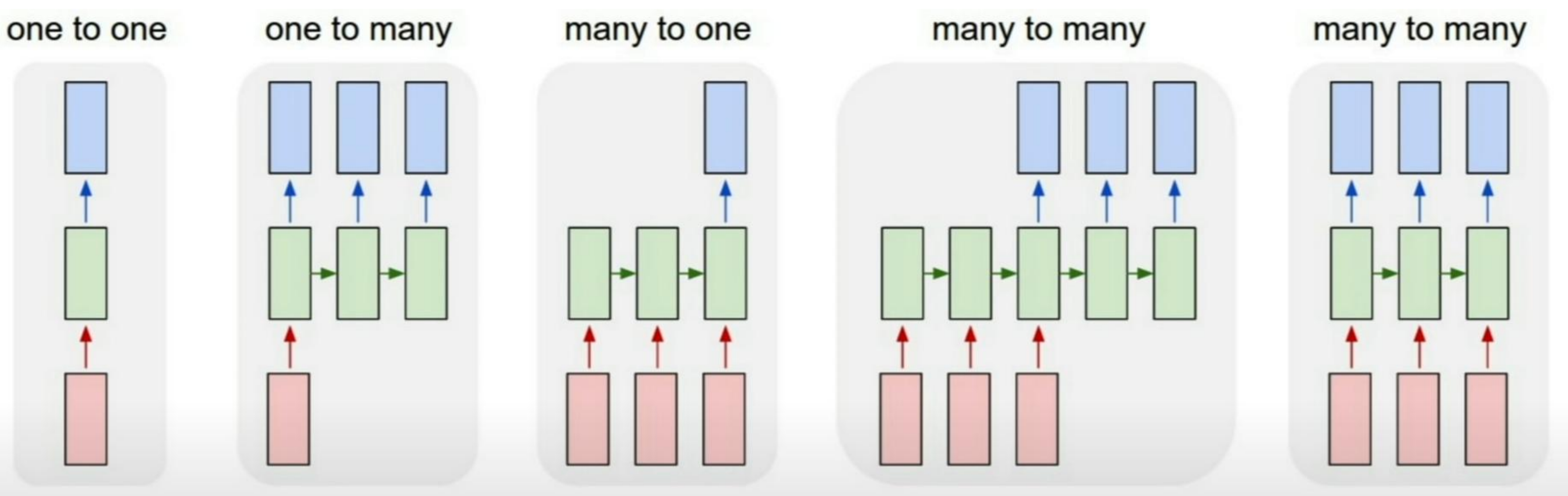
Vanilla RNNs

Different RNNs



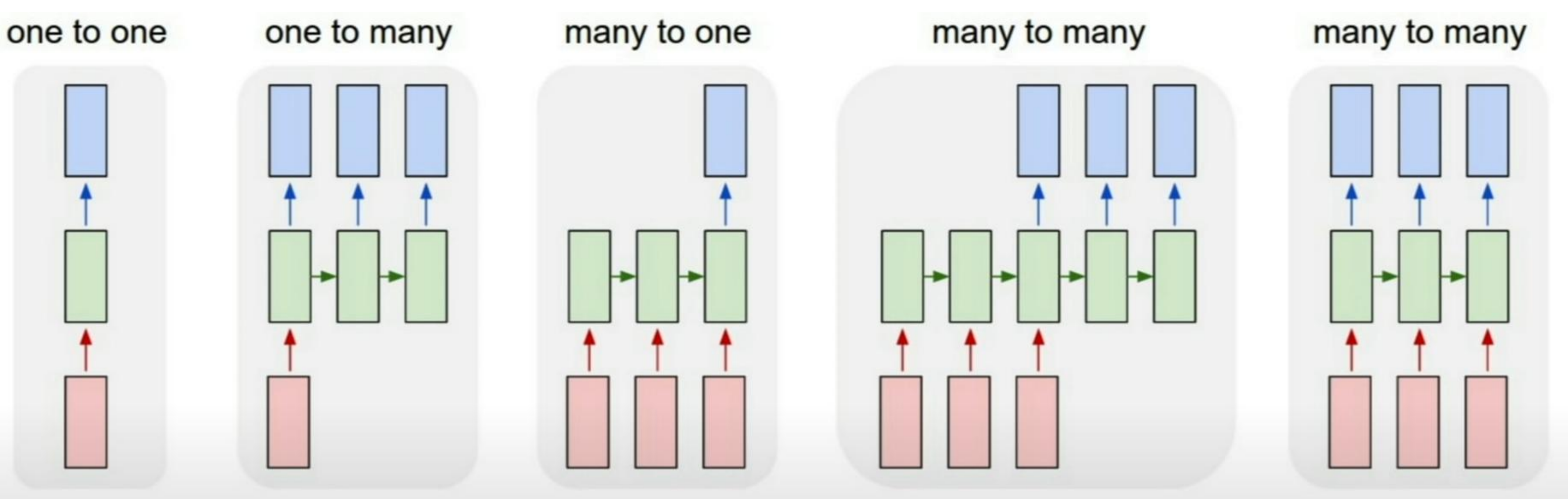
Eg: Image Captioning
Image -> Sequence of words

Different RNNs



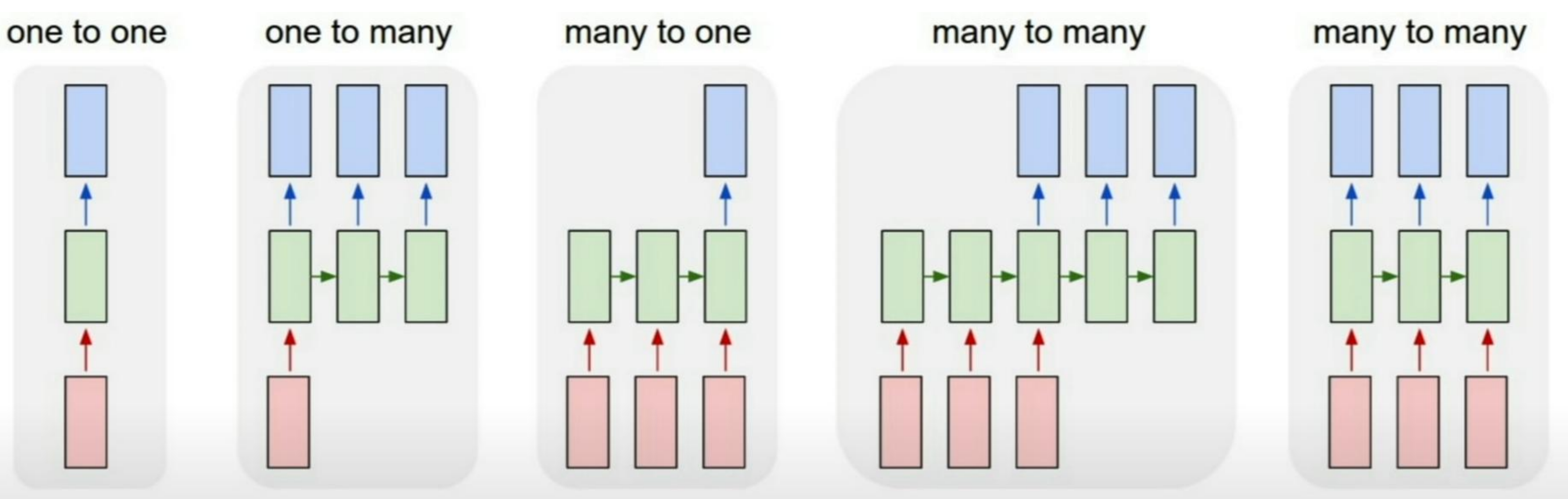
Eg: Sentiment classification
Sequence of words -> Sentiment

Different RNNs



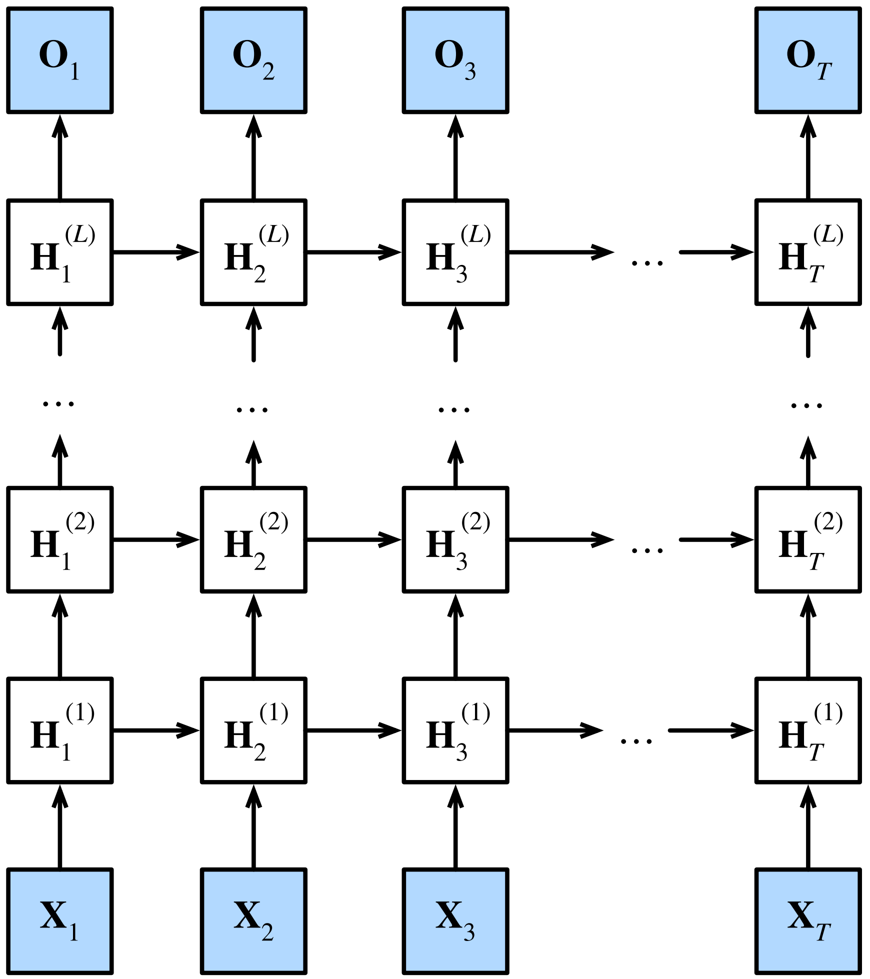
Eg: Machine Translation
Sequence of words -> Sequence of words

Different RNNs



Eg: Video Classification on frame level

Deep RNNs



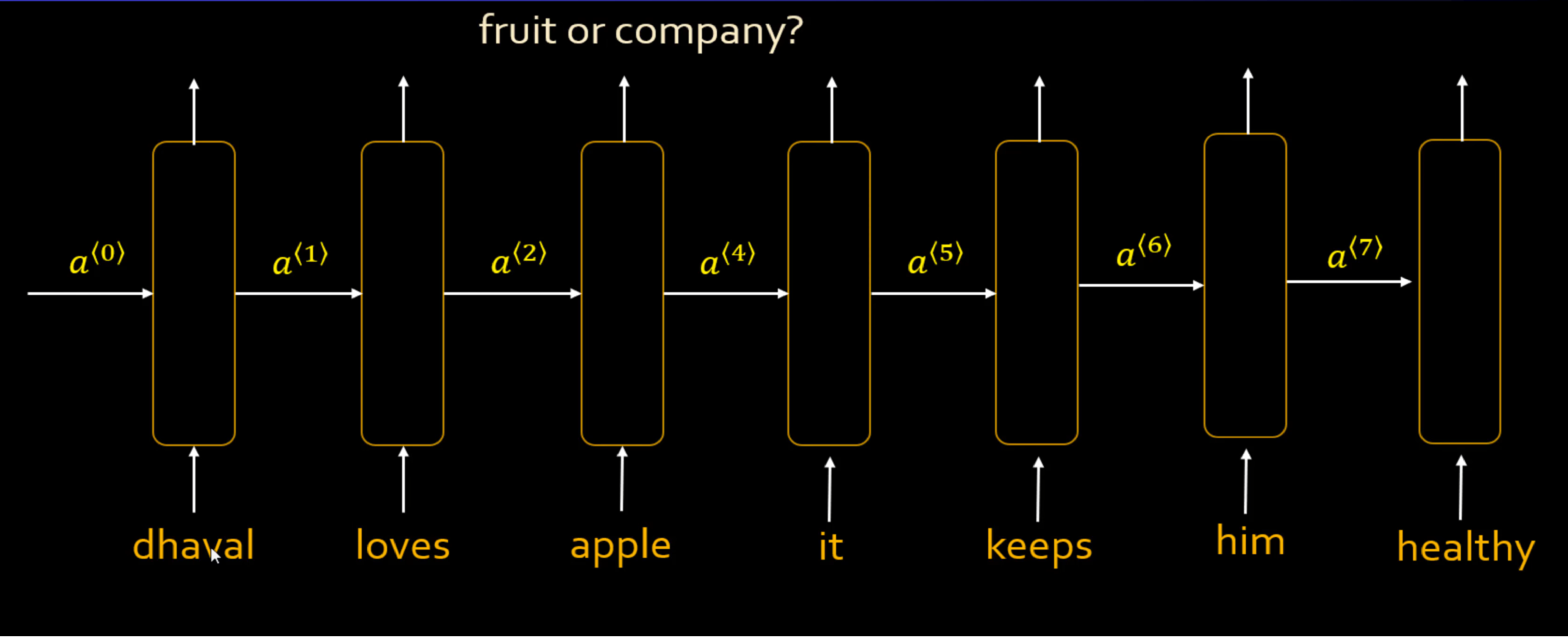
RNNs that are deep not only in the time direction but also in the input-to-output direction.

$$\mathbf{H}_t^{(l)} = \phi_l(\mathbf{H}_t^{(l-1)}\mathbf{W}_{xh}^{(l)} + \mathbf{H}_{t-1}^{(l)}\mathbf{W}_{hh}^{(l)} + \mathbf{b}_h^{(l)})$$

$$\mathbf{O}_t = \mathbf{H}_t^{(L)}\mathbf{W}_{hq} + \mathbf{b}_q$$

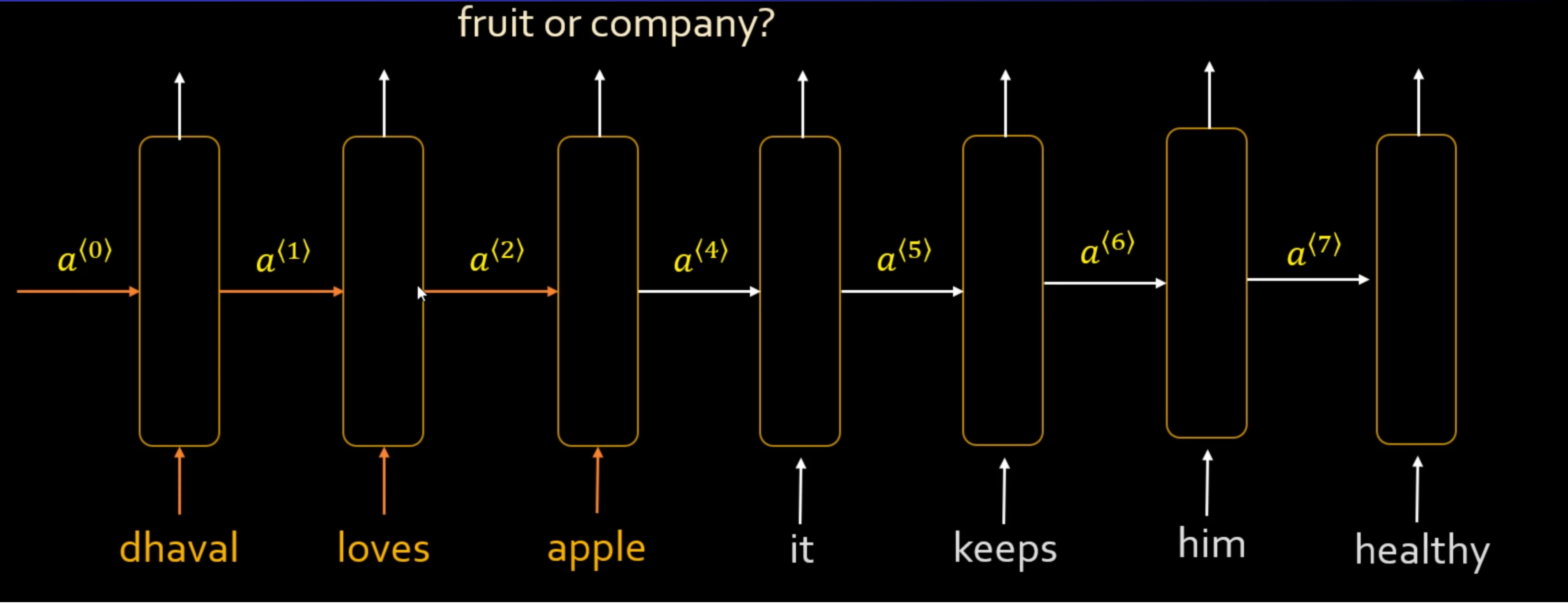
Source: Deep Recurrent Neural Networks — Dive into Deep Learning 1.0.0-alpha1.post0 documentation (d2l.ai)

Bi-Directional RNNs: Intuition



Source: codebasics - YouTube

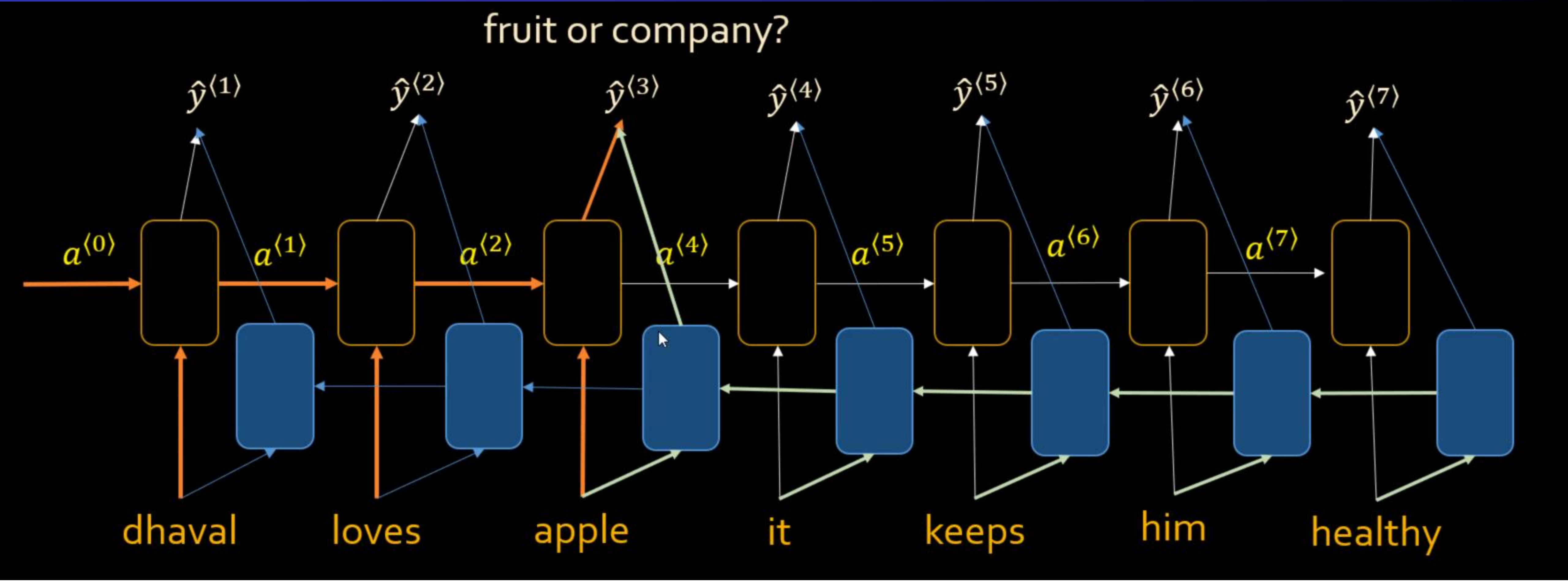
Bi-Directional RNNs: Intuition



- The o/p at the third time step (where input is the string "apple") depends on only previous two i/ps

Source: codebasics - YouTube

Bi-Directional RNNs



- Adding an additional backward layer with connections as shown above makes the o/p at a time step depend on both previous as well as future i/ps.

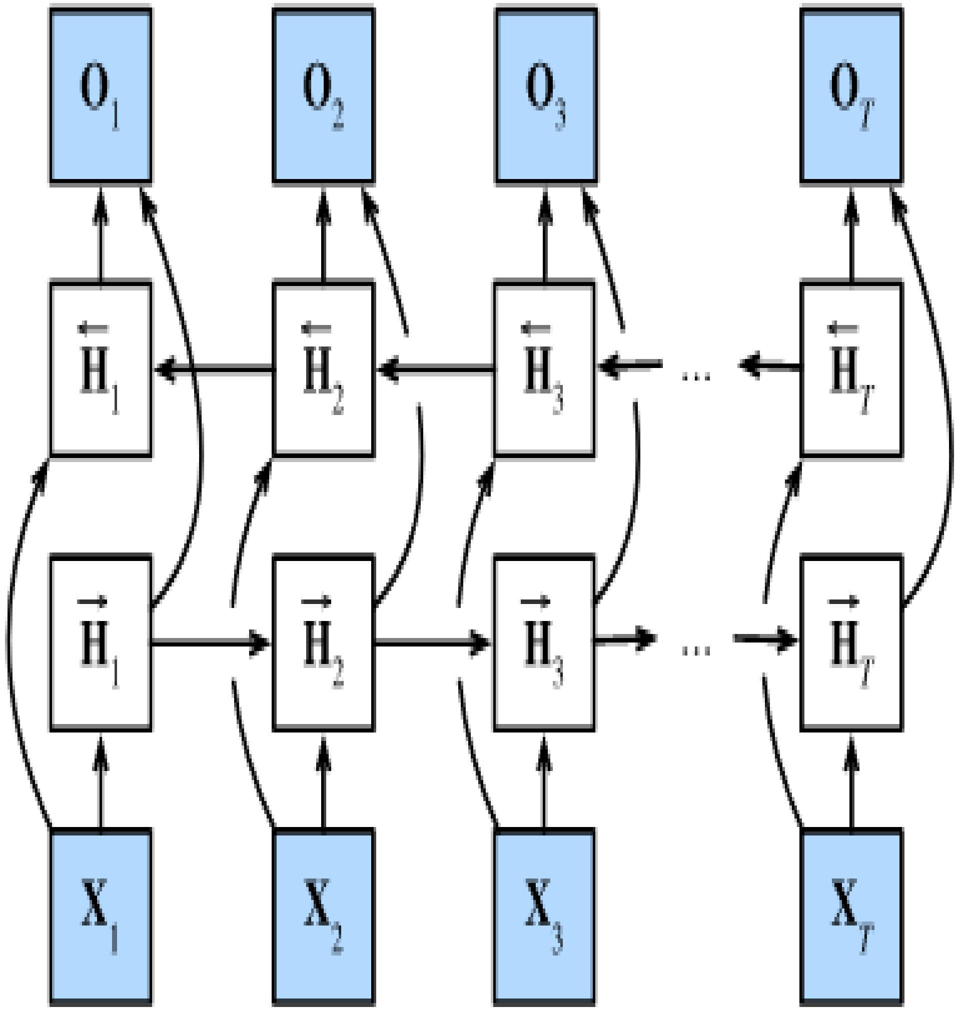
Source: codebasics - YouTube

Bi-directional RNNs

Example - speech detection

- I am ___.
- I am ___ hungry.
- I am ___ hungry, and I can eat half a cake.

- Regular RNNs are causal
  - look at past and present inputs to generate output.
- Use 2 recurrent layers on the same inputs
  - One reading words from left to right
  - Another reading words from right to left
- Combine their outputs at each time step



Source: Bidirectional Recurrent Neural Networks — Dive into Deep Learning 1.0.0-alpha1.post0 documentation (d2l.ai)

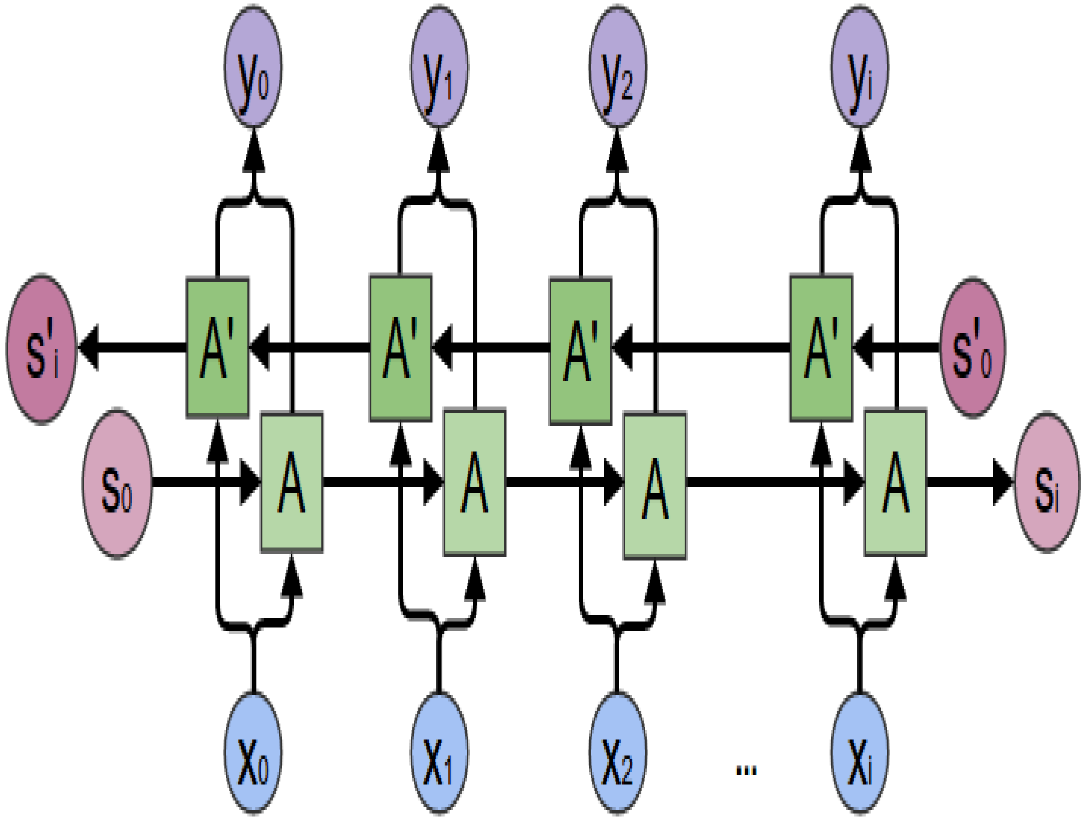
Bi-directional RNN computation

$$A_t(Forward) = \phi(X_t * W_{XA}^{forward} + A_{t-1}(Forward) * W_{AA}^{forward} + b_A^{forward})$$

$$A_t(Backward) = \phi(X_t * W_{XA}^{backward} + A_{t+1}(Backward) * W_{AA}^{backward} + b_A^{backward})$$

- ϕ is the activation function
- W the weight matrix
- b the bias.
- The hidden state at time t is given by a combination of At(Forward) and At(Backward).
- The output at any given hidden state is:

$$O_t = H_t * W_{AY} + b_Y$$



Bi-directional RNN computation

$$A_t(Forward) = \phi(X_t * W_{XA}^{forward} + A_{t-1}(Forward) * W_{AA}^{forward} + b_A^{forward})$$

$$A_t(Backward) = \phi(X_t * W_{XA}^{backward} + A_{t+1}(Backward) * W_{AA}^{backward} + b_A^{backward})$$

- ϕ is the activation function
- W the weight matrix
- b the bias.
- The hidden state at time t is given by a combination of At(Forward) and At(Backward).
- The output at any given hidden state is:

$$O_t = H_t * W_{AY} + b_Y$$