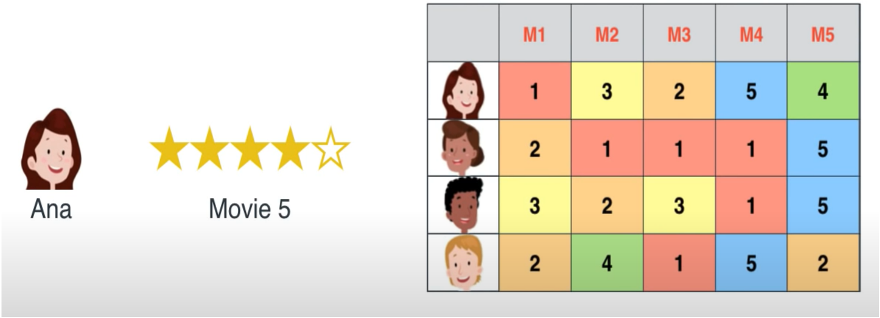
Sequence Modeling: Recurrent and ~~Recursive~~ Nets (10.1-10.5)

Jaehyoung Hong

Recurrent neural network(RNN) is specialized for sequence-like input

✓ CNN → Specialized for grid-like input

Image

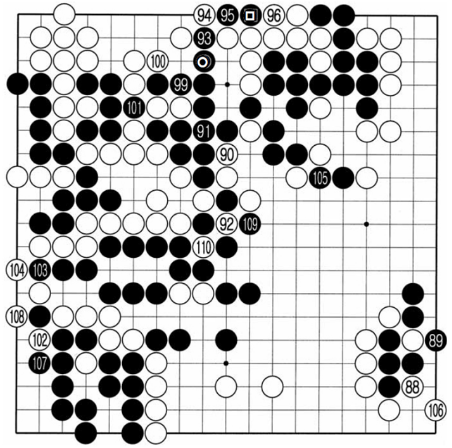


Netflix problem (Rating of movie)

Fixed input (Given for one time-step)

✓ RNN → Specialized for sequence-like input

Language

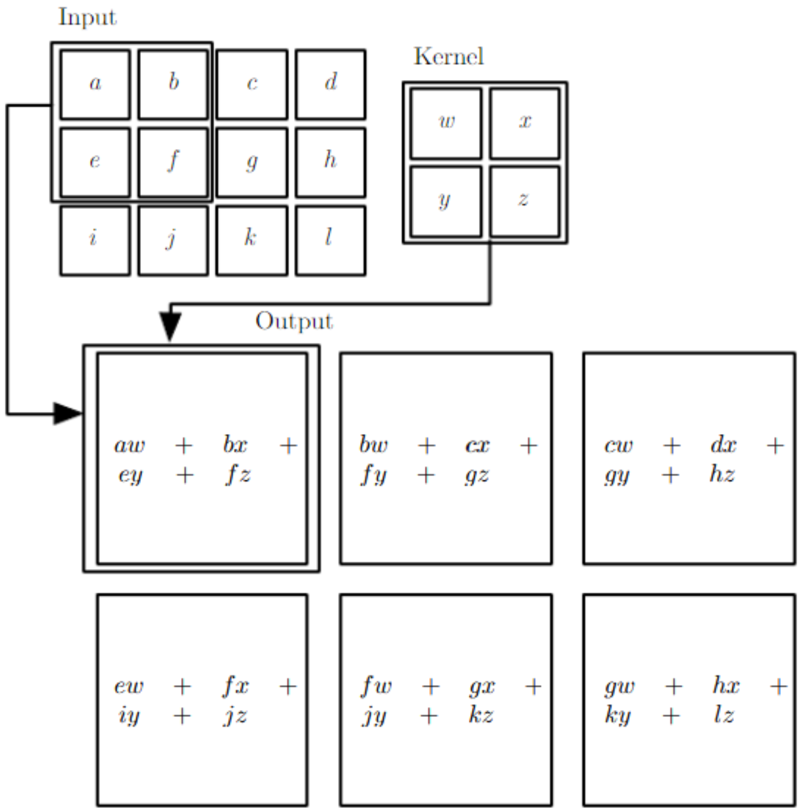


Go

For many case, each time-point of sequence are correlated with each other

$x^{(t)}$ and $x^{(t+1)}$ are correlated

Like CNN, RNN can handle various length of sequence by using parameter sharing

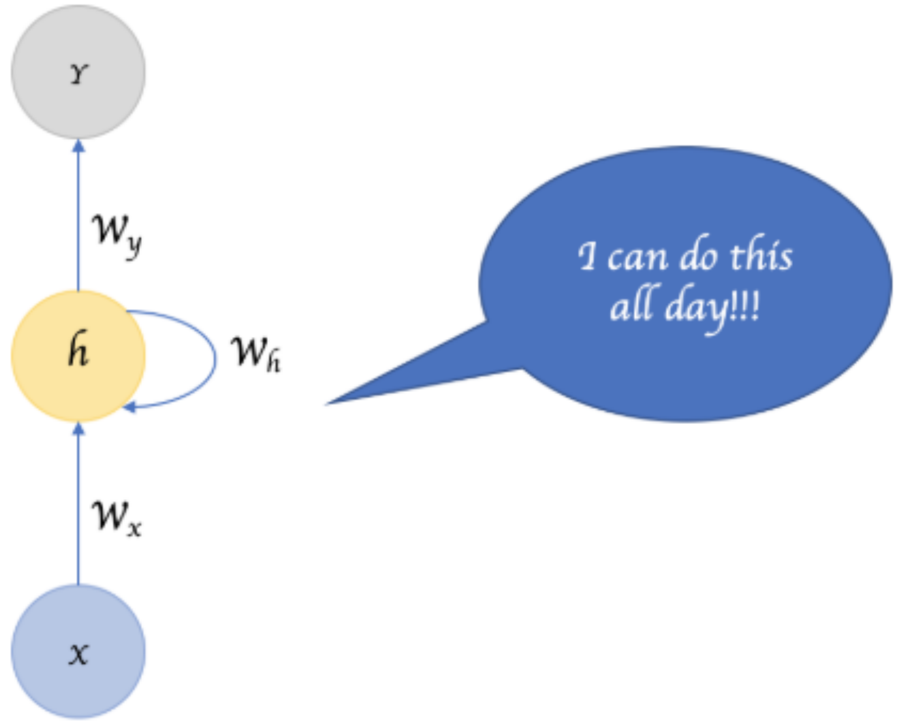


CNN : Using same kernel (parameter sharing)

Handle various size of image

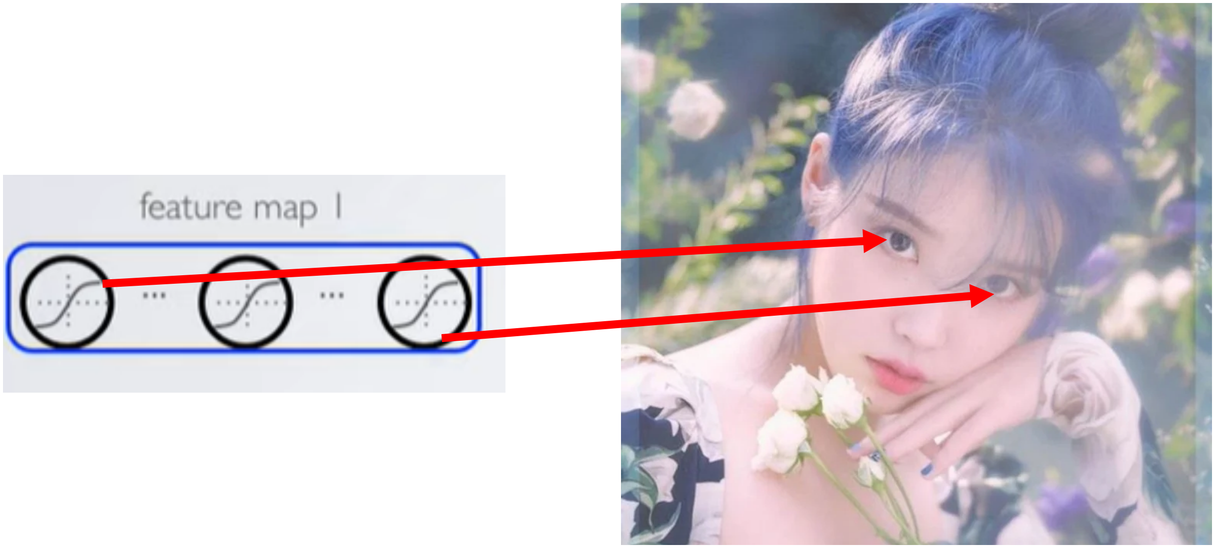
hidden

hidden



Handle various length of image

By parameter sharing, RNN is invariant under position



Eye must be detected as eye
although it has different position

I went to Nepal in 2009

In 2009, I went to Nepal

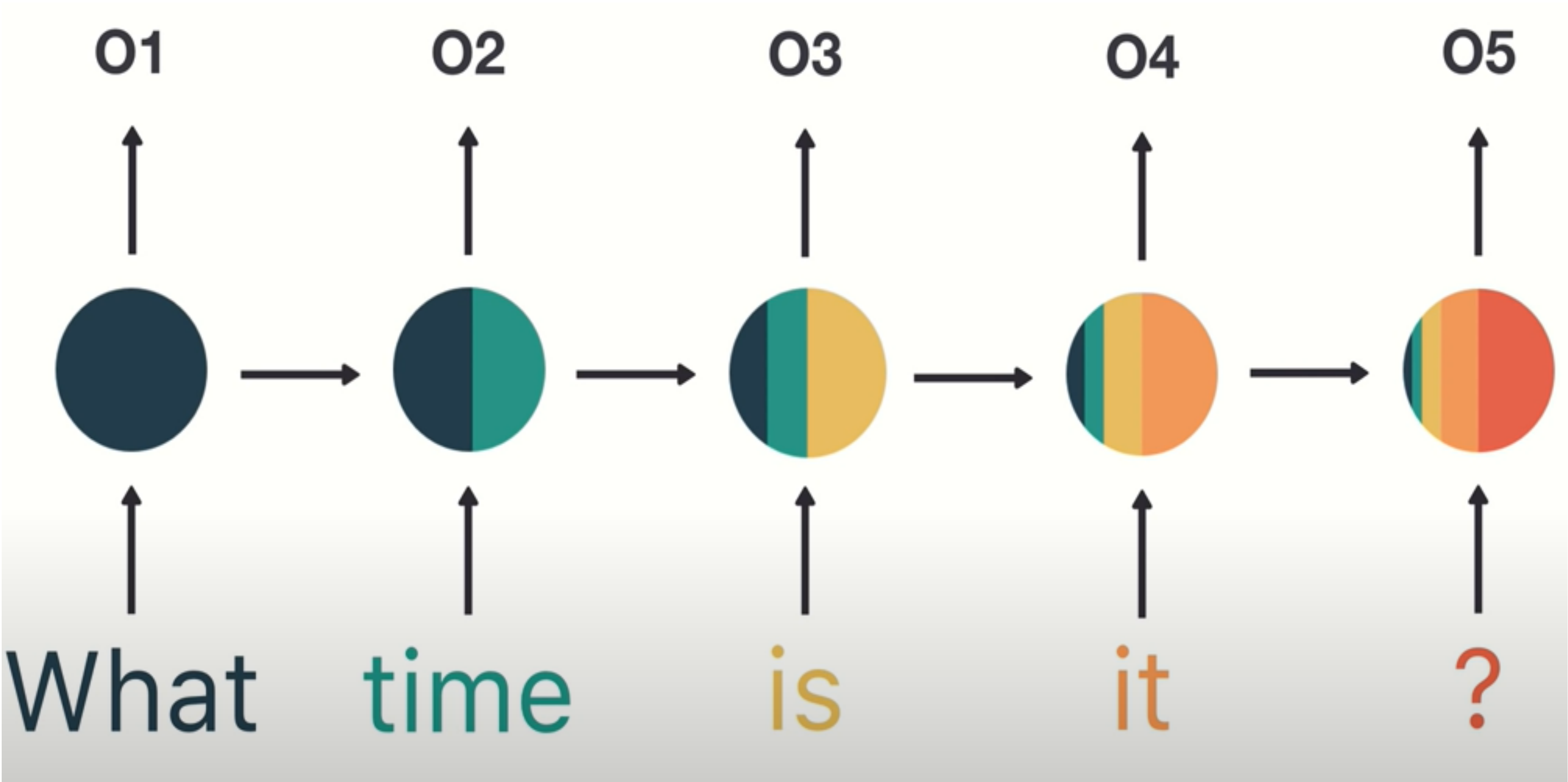
CNN : Using same kernel (parameter sharing)

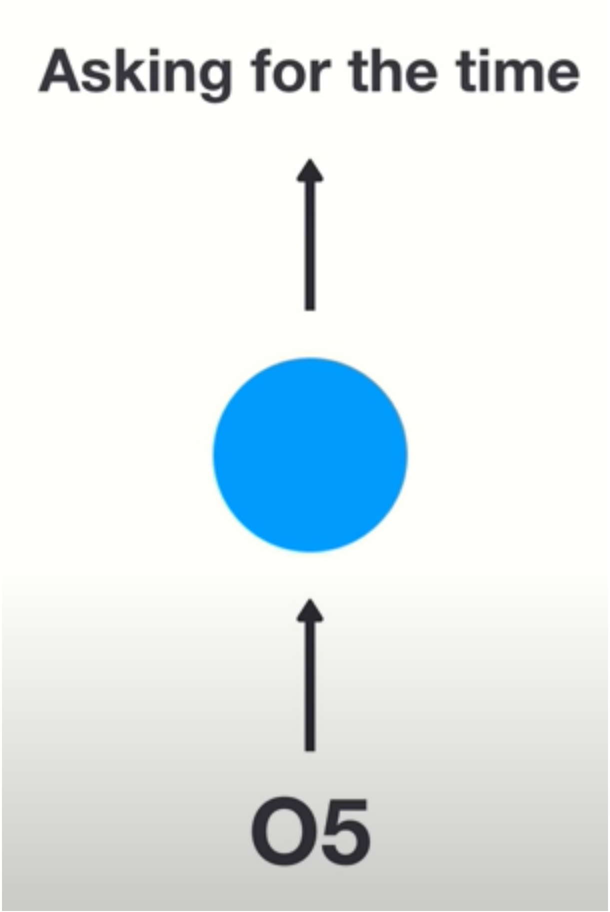
Same input gives same feature

RNN : Using same weight (parameter) for each time-step (parameter sharing)

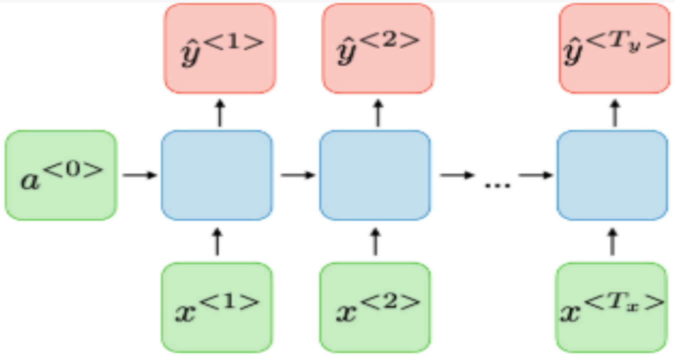
Same word gives same meaning

Description of RNN

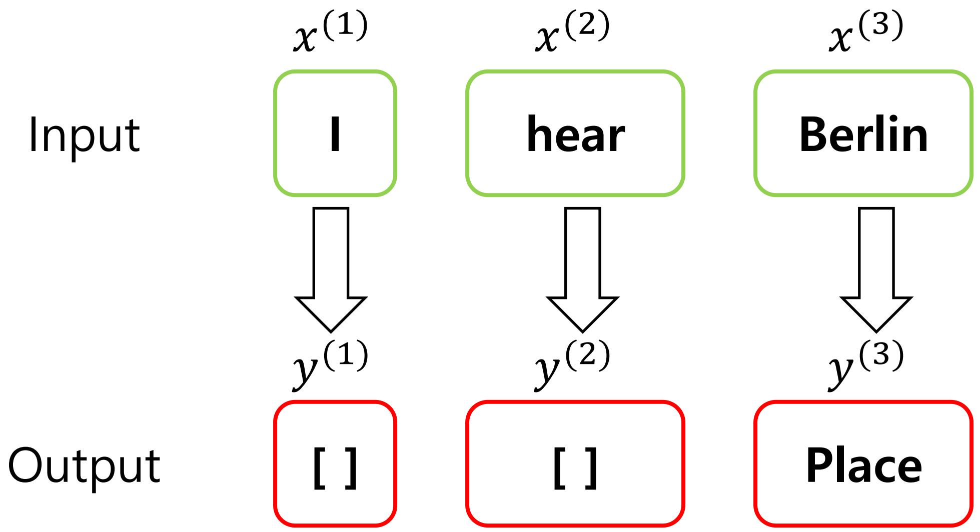




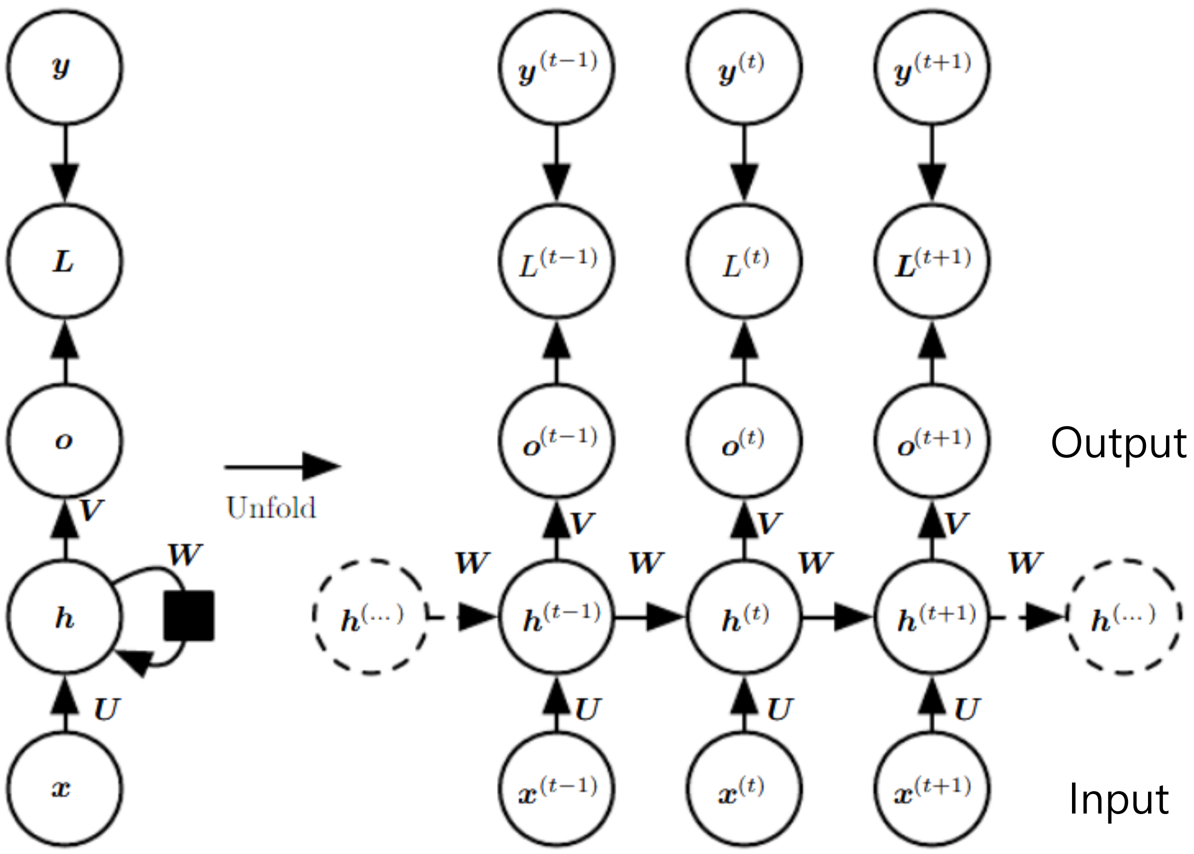
First, we consider sequence-to-sequence RNN

Type of RNN	Illustration	Example
Many-to-many $T_x = T_y$		Name entity recognition

<Name entity recognition>



Such RNN can be defined as recurrent form from $t = 1$ to $t = \tau$



Previous information influenced on current information only through hidden state (h)

$t = 1$ to $t = \tau$

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\ \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}), \end{aligned}$$

Same parameter!!

RNN

$$L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right)$$

$$= \sum_t L^{(t)}$$

Total loss is sum of losses over all the time steps

$$= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right)$$

Loss

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \left.\frac{\partial \mathcal{L}^{(T)}}{\partial W}\right|_{(t)}$$

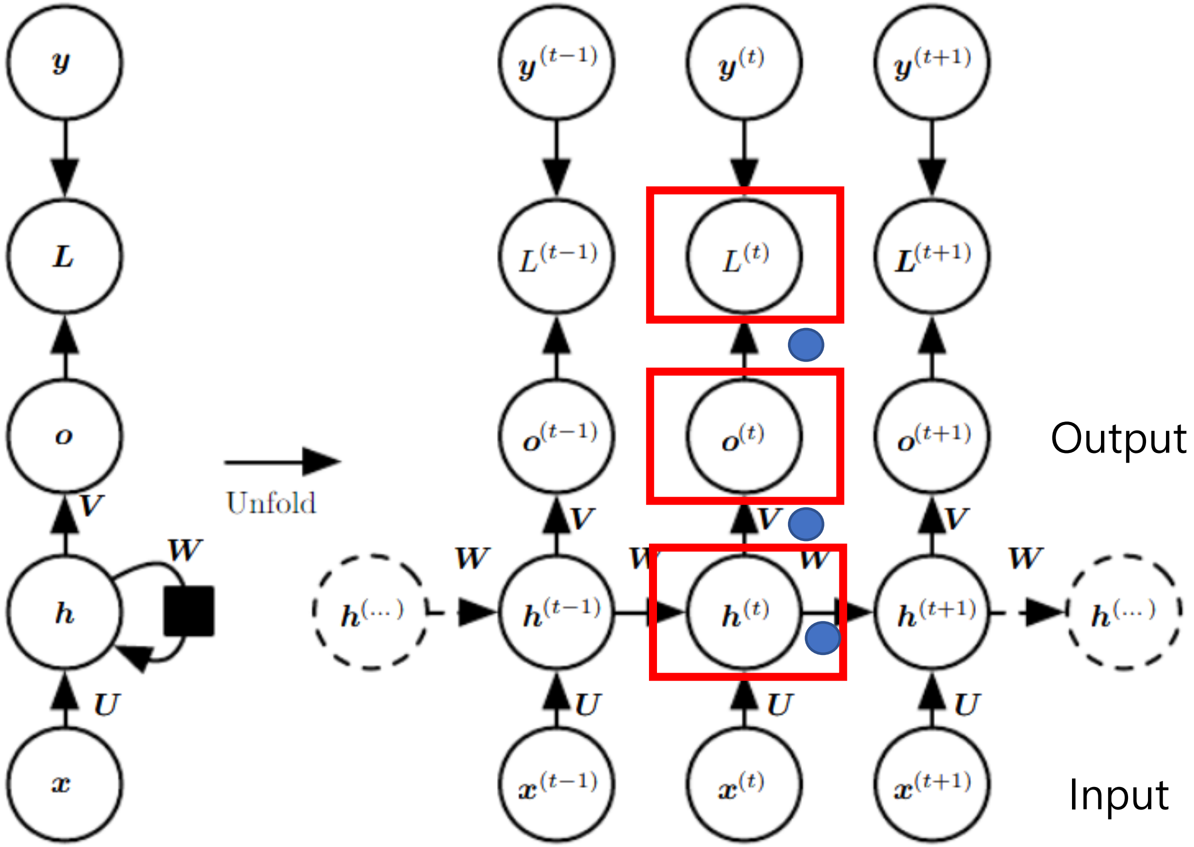
Runtime : $O(\tau)$

Memory : $O(\tau)$

Gradient

(Backpropagation through time : BPTT)

Example of calculation of BPTT



$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}$$

Follow the arrow

(1) $\frac{\partial L}{\partial L^{(t)}} = 1.$

(2) $(\nabla_{\boldsymbol{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i=y^{(t)}}.$

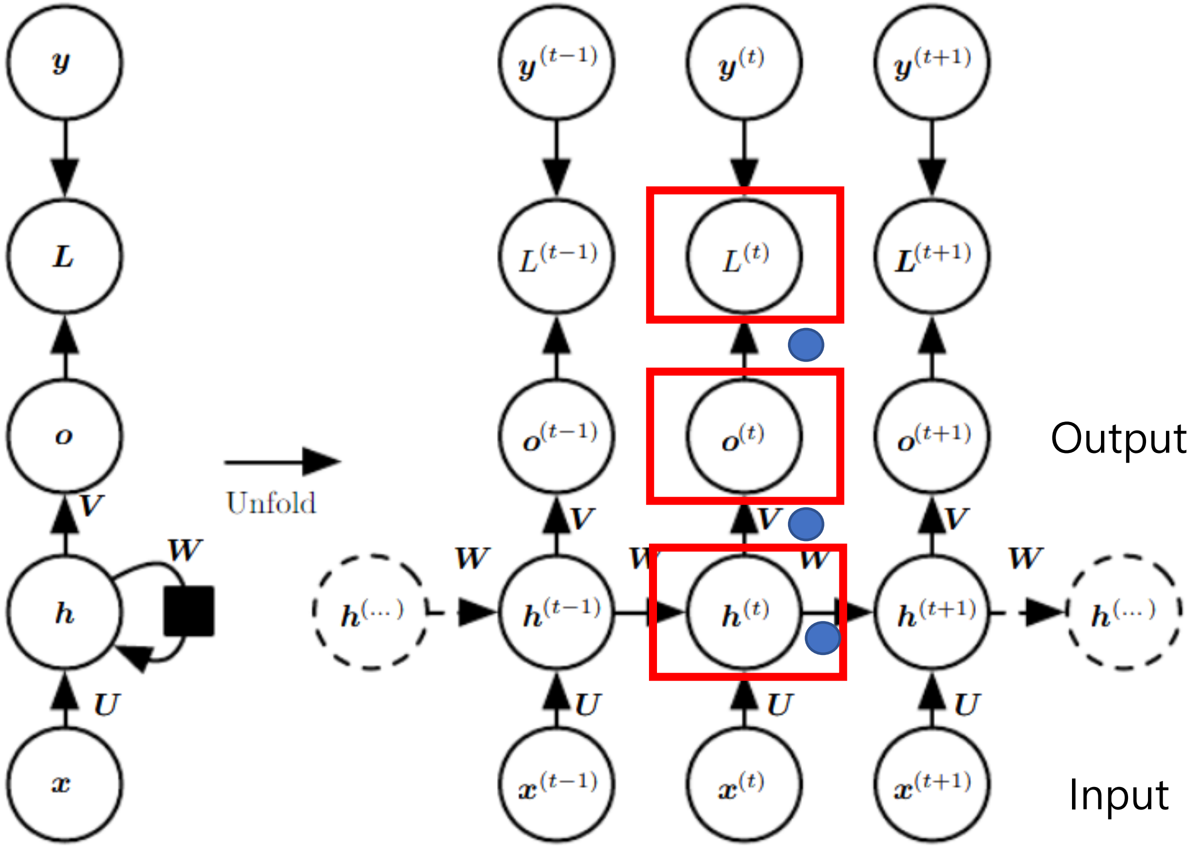
(3) $\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L.$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$= \boldsymbol{W}^\top \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t+1)}\right)^2\right)(\nabla_{\boldsymbol{h}^{(t+1)}} L) + \boldsymbol{V}^\top (\nabla_{\boldsymbol{o}^{(t)}} L)$$

Example of calculation of BPTT

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$



Follow the arrow

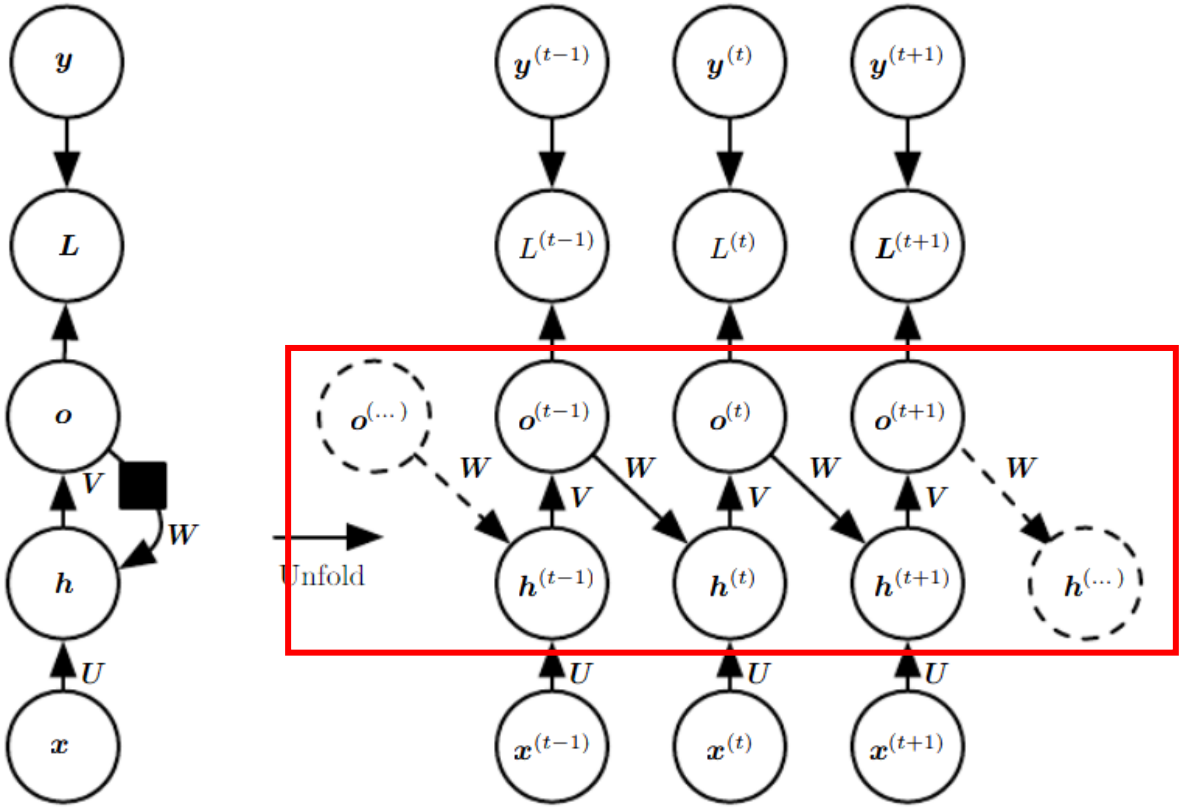
$$
\begin{aligned}
\nabla_{\boldsymbol{c}} L &= \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}}\right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L, \\
\nabla_{\boldsymbol{b}} L &= \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}}\right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L, \\
\nabla_{\boldsymbol{V}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_{\boldsymbol{V}^{(t)}} o_i^{(t)} = \sum_t \left(\nabla_{\boldsymbol{o}^{(t)}} L\right) \boldsymbol{h}^{(t)^\top}, \\
\nabla_{\boldsymbol{W}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} \\
&= \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{h}^{(t-1)^\top}, \\
\nabla_{\boldsymbol{U}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} \\
&= \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{x}^{(t)^\top},
\end{aligned}
$$

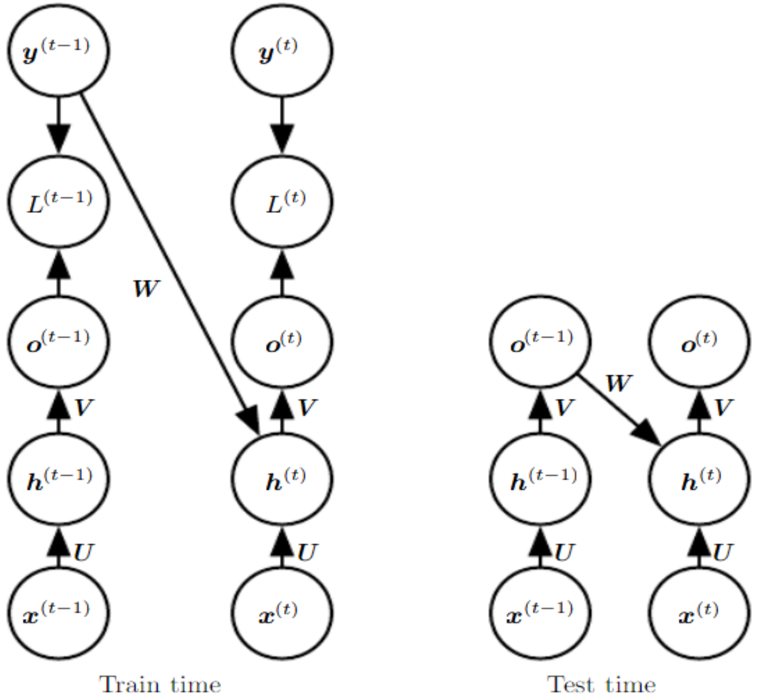
Bad Result

$$W = W^{(1)} = \cdots = W^{(\tau)},\ U = U^{(1)} = \cdots = U^{(\tau)},\ V = V^{(1)} = \cdots = V^{(\tau)}$$

We can use previous information from output to better trained the RNN (Teacher forcing)

$$\log p\left(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right)$$
$$= \log p\left(\boldsymbol{y}^{(2)} \mid \boldsymbol{y}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) + \log p\left(\boldsymbol{y}^{(1)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right).$$



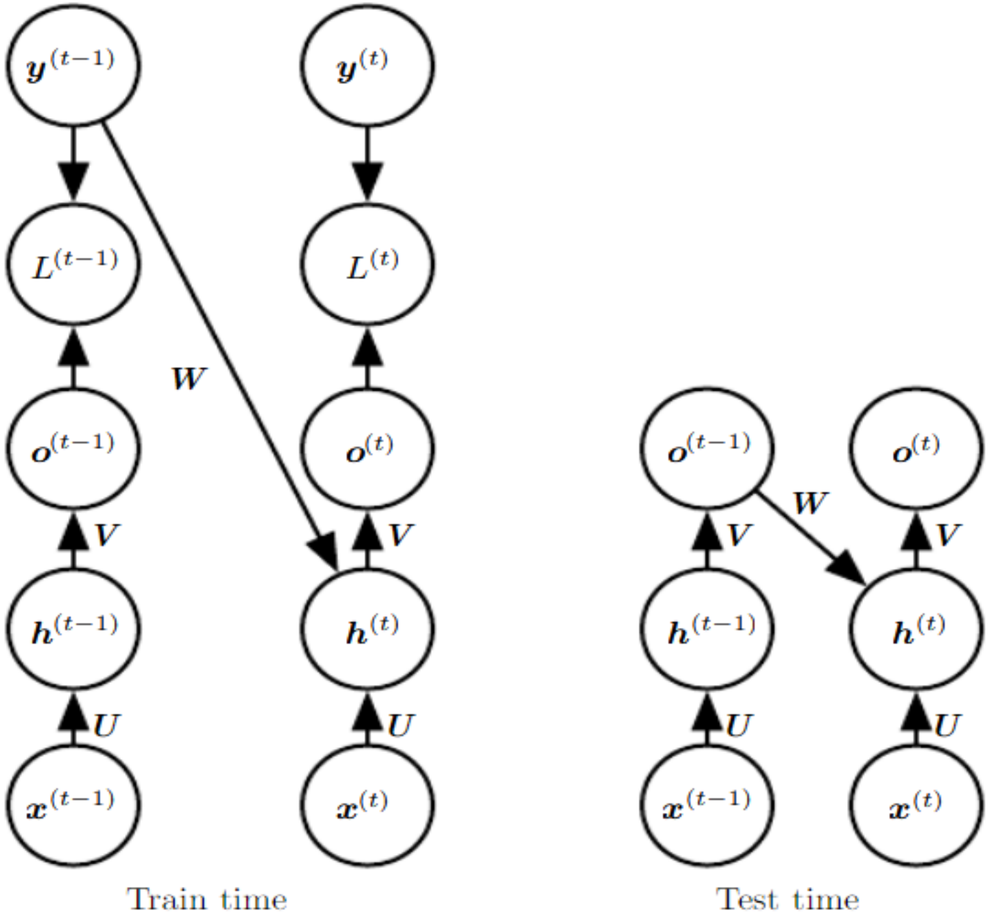


Generally less powerful :
Output has less information than hidden

Teacher forcing : Use output as next input

Avoid BPTT !! (No hidden-to-hidden)

We can use previous information from output to better trained the RNN (Teacher forcing)



Teacher forcing : Use output as next input

Want to train?

"Mary had a little lamb whose fleece was white as snow"

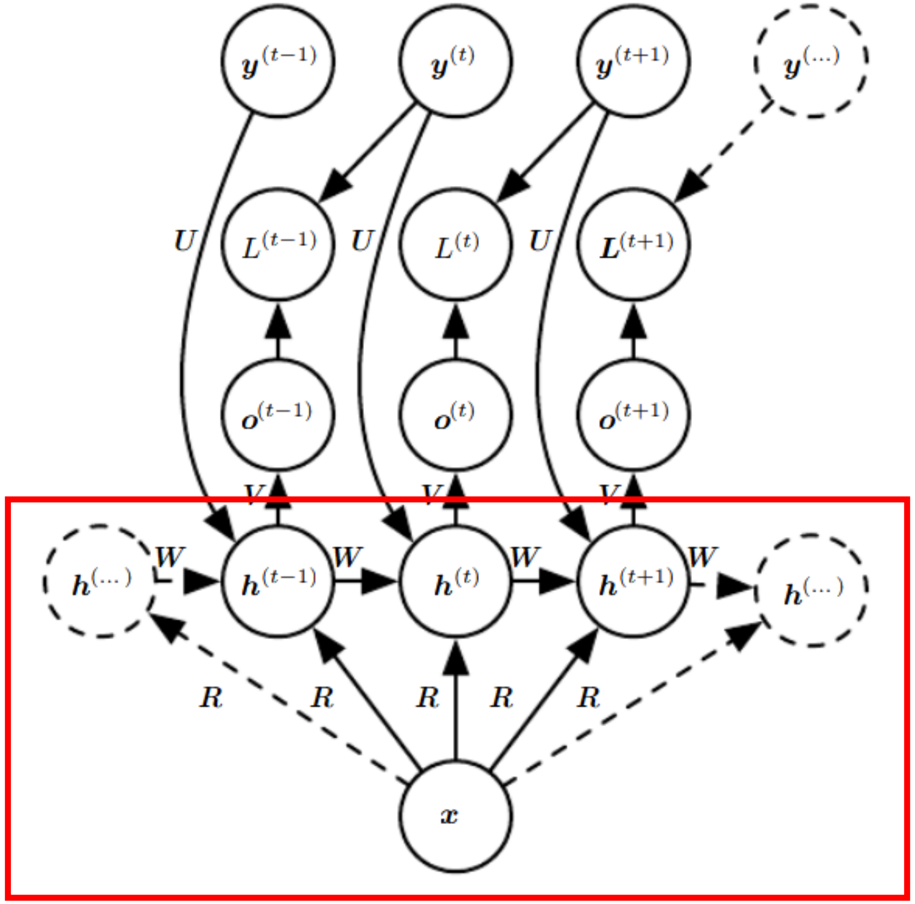
X	$\hat{y}$
[Start]	"a"
[Start], "a"	? (Any word can get punished)

<Without teacher forcing : Slow & incorrect>

X	$\hat{y}$
[Start]	"a"
[Start], "Mary"	?

With teacher forcing

RNN which takes single vector as input



When input is $\boldsymbol{x}$ rather than $\boldsymbol{x}^{(t)}$

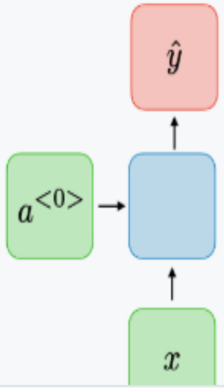
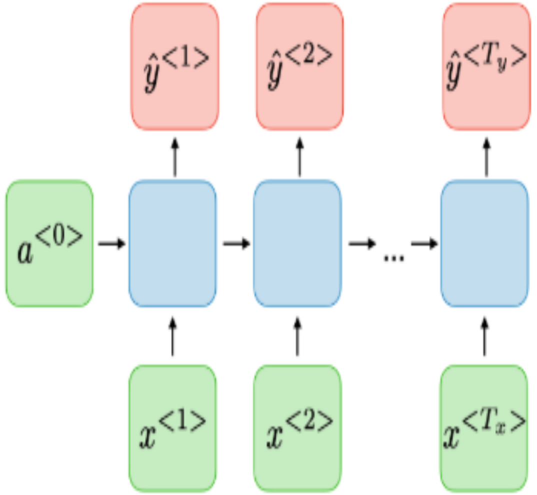
1. as an extra input at each time step, or
2. as the initial state $\boldsymbol{h}^{(0)}$, or
3. both.

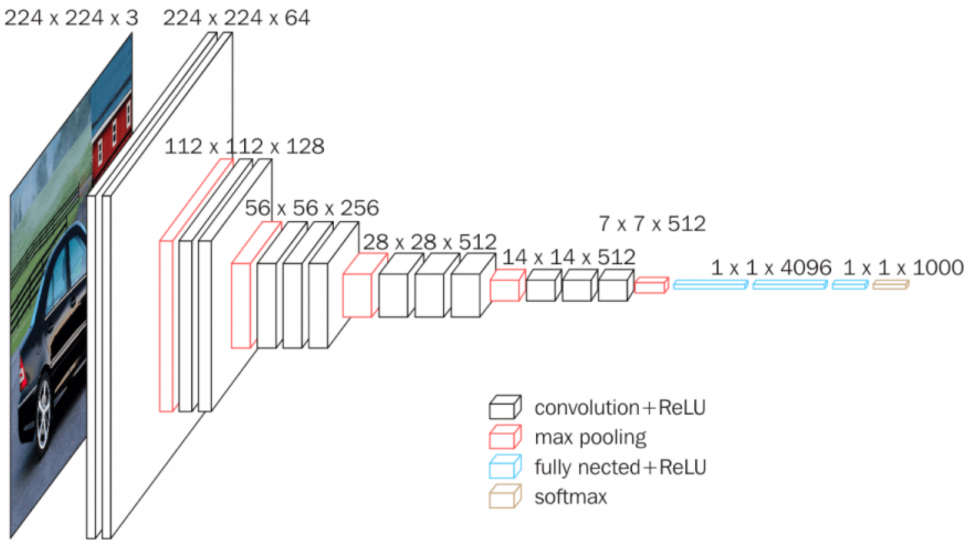
$x^T R$ is added as additional input to the hidden units

$x^T R$ is new bias parameter

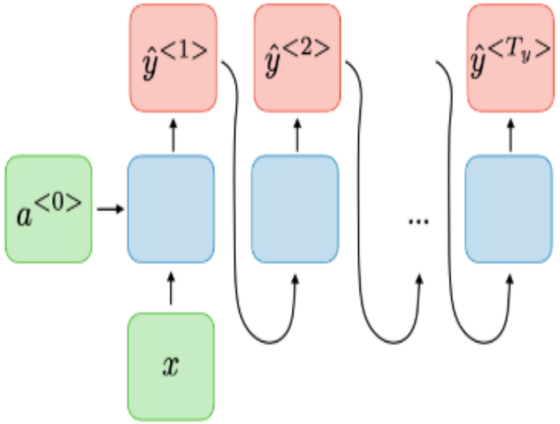
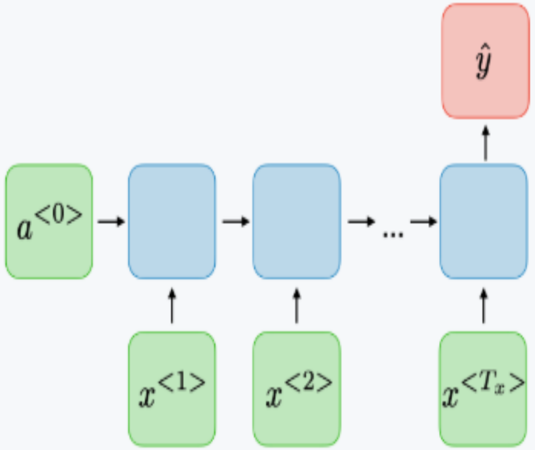
$P(y|x) \rightarrow P(y|w), w = f(x)$

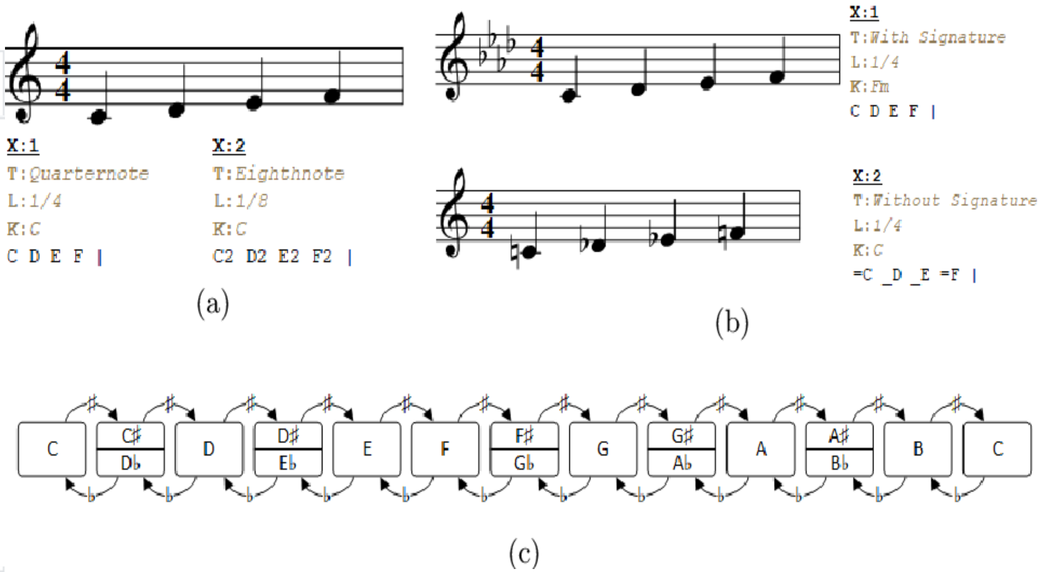
Variant type of RNN : Depending on Input and Output

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
Many-to-many $T_x = T_y$		Name entity recognition



Variant type of RNN : Depending on Input and Output

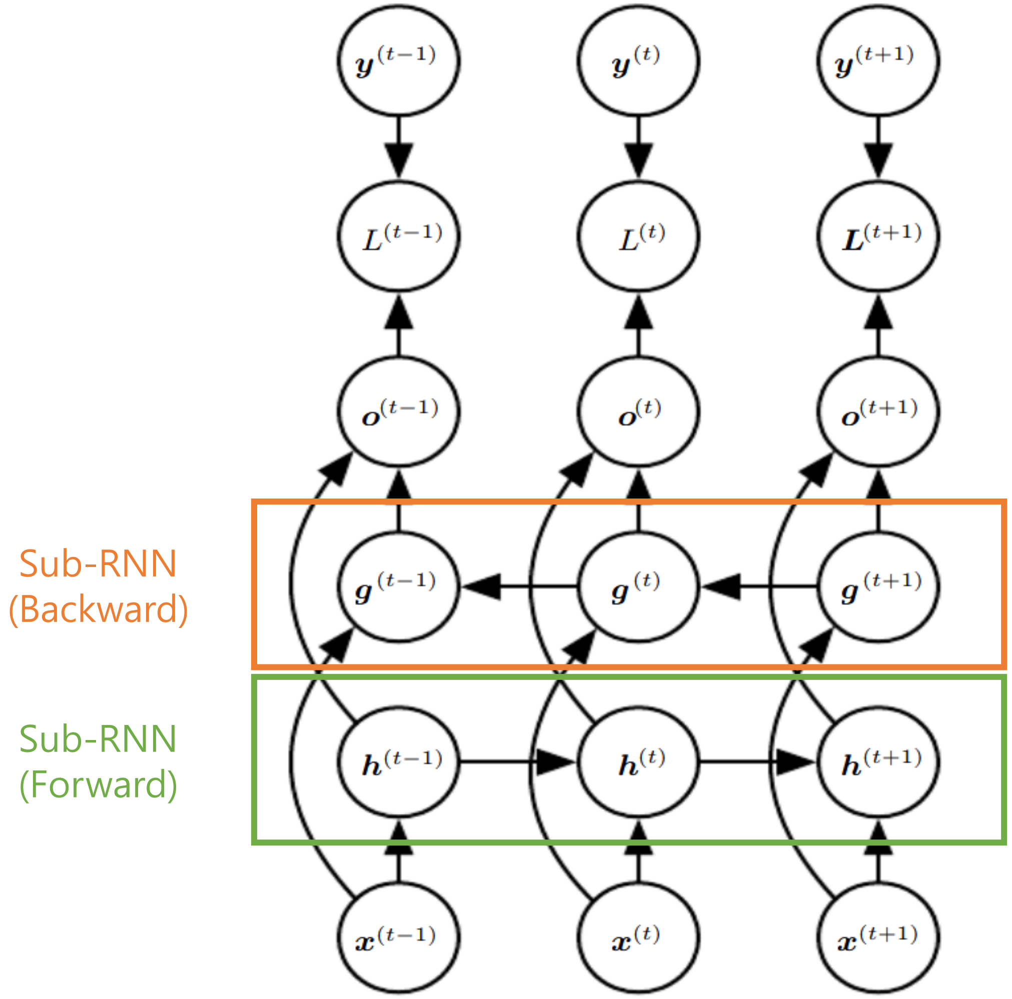
Type of RNN	Illustration	Example
One-to-many $T_x = 1, T_y > 1$		Music generation
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification



"I love this movie. I've seen it many times and it's still awesome." → 

"This movie is bad. I don't like it it all. It's terrible." → 

Various type of RNN : Bidirectional RNNs using future information



Example

- ✓ Speech recognition
- ✓ Handwriting recognition
- ✓ Bioinformatics

Vanilla RNN
(Causal : $x^{(t)} \leftarrow x^{(1)}, \dots, x^{(t-1)}$)

vs

Bidirectional RNN
(Around t : $x^{(t)} \leftarrow x^{(t+1)}, x^{(t-1)}, \dots$)

En
pr

❒ **Attention model** — This model allows an RNN to pay attention to specific parts of the input that is considered as being important, which improves the performance of the resulting model in practice. By noting $\alpha^{<t,t'>}$ the amount of attention that the output $y^{<t>}$ should pay to the activation $a^{<t'>}$ and $c^{<t>}$ the context at time t, we have:

$$\boxed{c^{<t>} = \sum_{t'} \alpha^{<t,t'>} a^{<t'>}} \quad \text{with} \quad \sum_{t'} \alpha^{<t,t'>} = 1$$

Remark: the attention scores are commonly used in image captioning and machine translation.

ontext'

Encoder

:

a Context

e

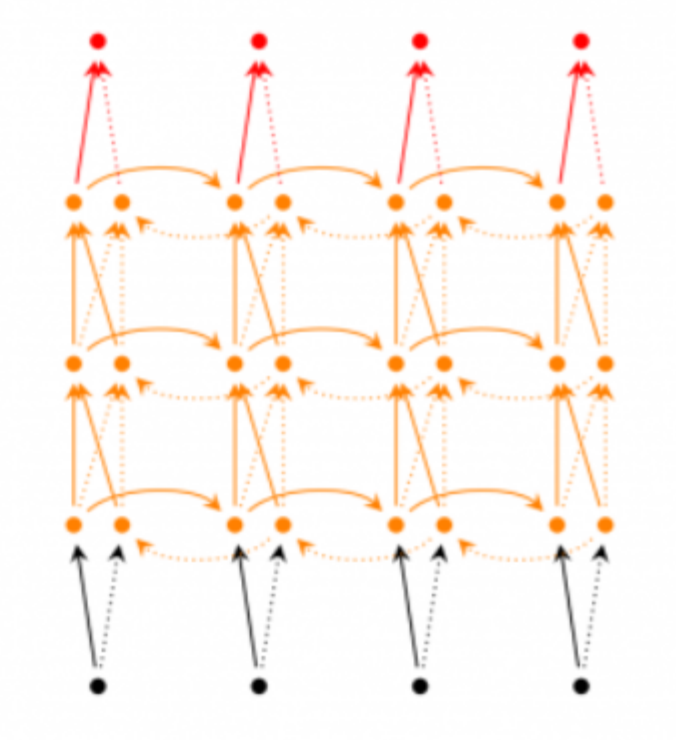
A cute teddy bear is reading Persian literature

A cute teddy bear is reading Persian literature

Deco

$c^{(n_x)})$

RNN has opportunities to make deep RNN



Deep RNN