Deep Learning: Theory and Practice

Recurrent Neural Networks

30-04-2019

Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Introduction

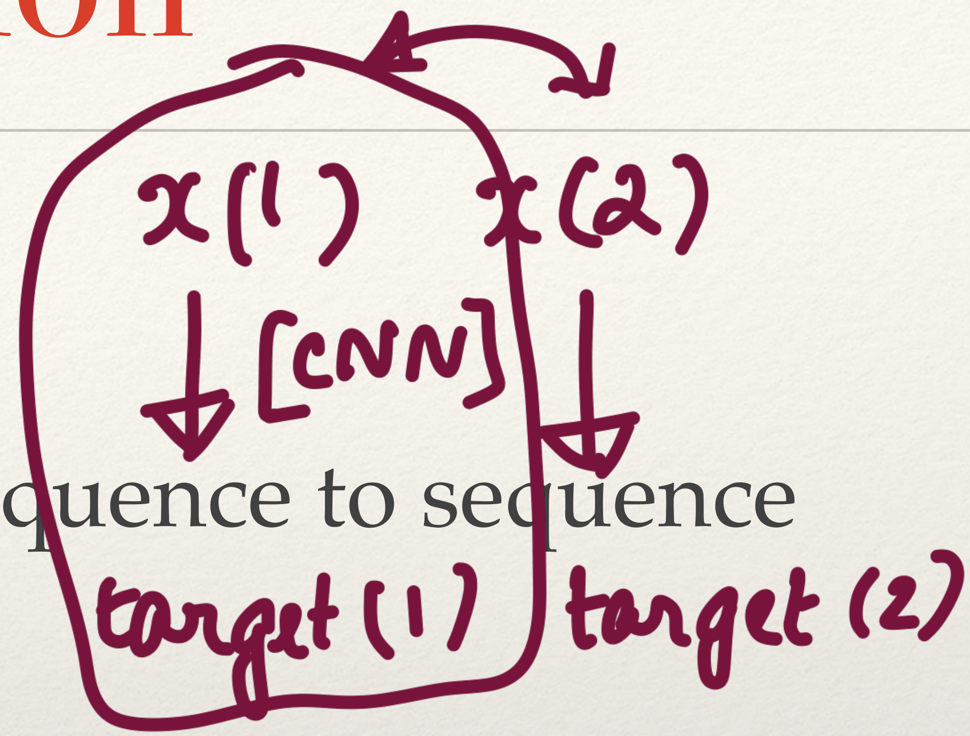
- The standard DNN/CNN paradigms
    - x, t
      (x,y) - ordered pair of data vectors/images (x) and target (~~y~~ t)
- Moving to sequence data

  x(1) x(2) x(3) x(4)

  target(1) (2) (3) (4)
    - (x(t),y(t)) where this could be sequence to sequence mapping task.
    - (x(t),y) where this could be a sequence to vector mapping task.

Introduction

- Difference between CNNs/DNNs
  - (x(t),y(t)) where this could be sequence to sequence mapping task.
    - Input features / output targets are correlated in time.
    - Unlike standard models where each pair is independent.
    - Need to model dependencies in the sequence over time.



Chap 10 of Deep Learning Book

Introduction to Recurrent Networks

Recurrence

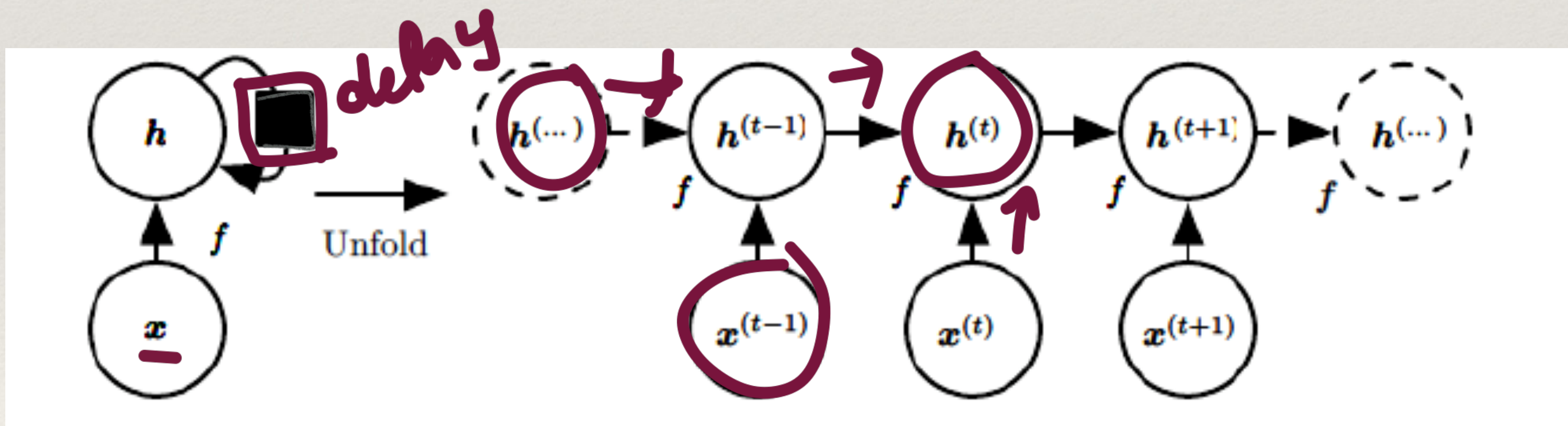
$$s^{(t)} = f(s^{(t-1)}; \theta),$$

$$\begin{aligned} s^{(3)} &= f(s^{(2)}; \theta) \\ &= f(f(s^{(1)}; \theta); \theta) \end{aligned}$$

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta),$$

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta),$$

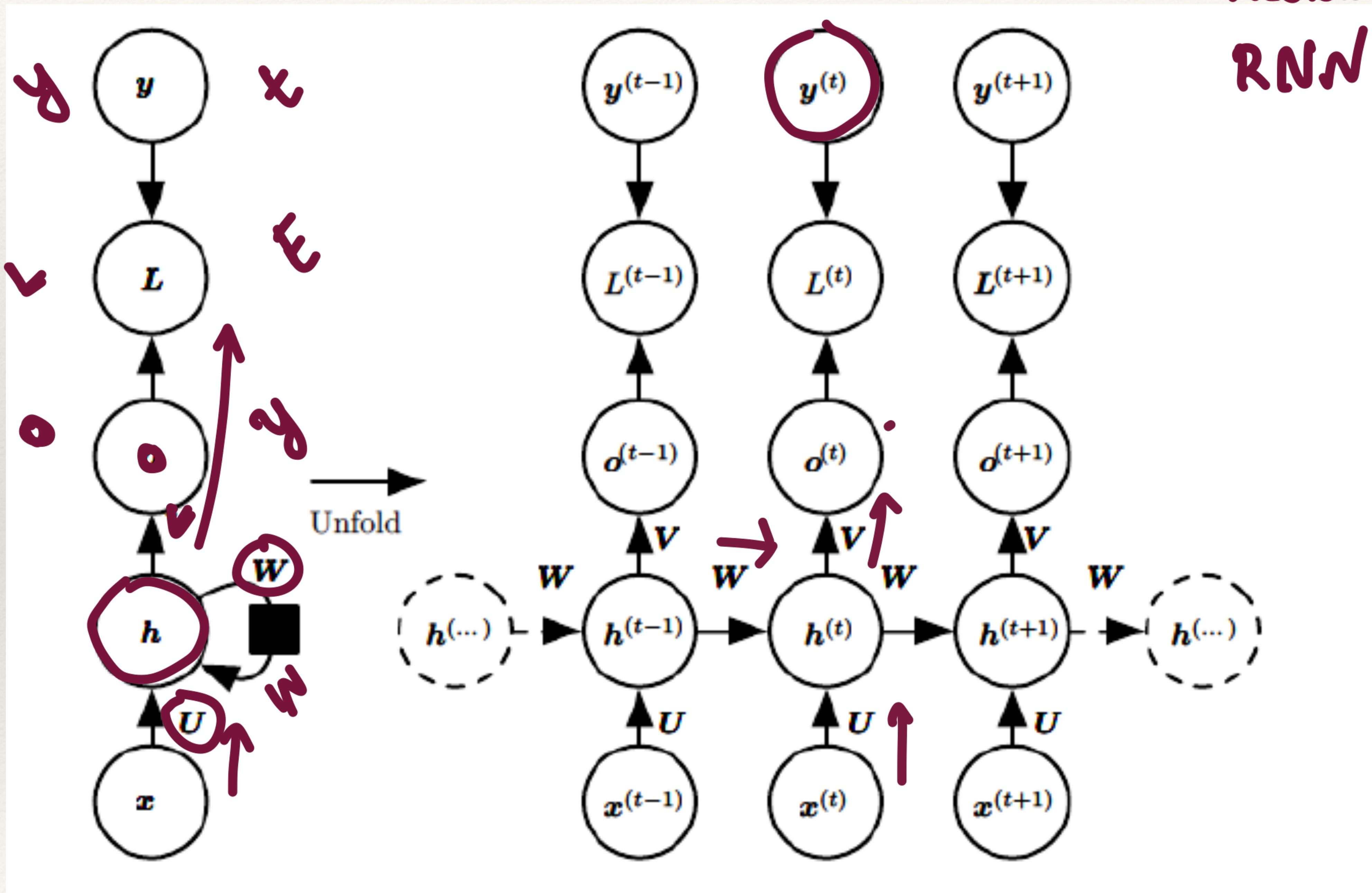
Neural network



"Deep Learning", Ian Goodfellow, Yoshua Bengio, Aaron Courville

Recurrent Networks

1- hidden layer
RNN



"Deep Learning", Ian Goodfellow, Yoshua Bengio, Aaron Courville

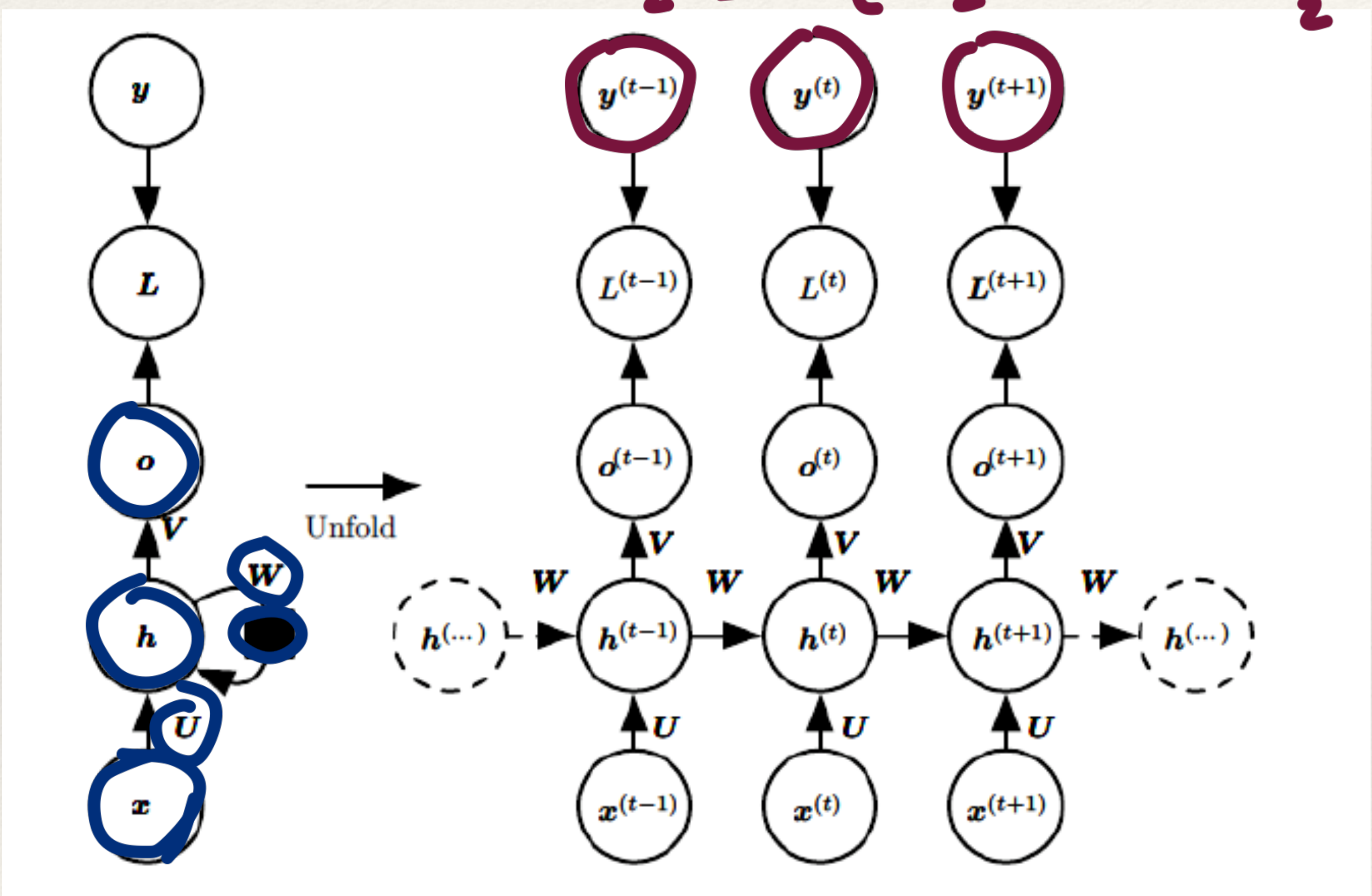
Recurrent Networks

time

$X_1 = \{x_1^{(1)} \ldots\ldots x_1^{(\tau_1)}\}$ ← sequence index

$X_2 = \{x_2^{(1)} \ldots\ldots x_2^{(\tau_2)}\}$

$\vdots$



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

$L(\hat{y}(t), y(t))$

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

"Deep Learning", Ian Goodfellow, Yoshua Bengio, Aaron Courville

Back Propagation in RNNs

$[x^{(1)}_1, x^{(2)}_1, \ldots x^{(30)}]$ April

$[x^{(1)}_2 \ldots x^{(31)}_2]$ May

$$
\begin{aligned}
a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t)} \\
h^{(t)} &= \tanh(a^{(t)}) \\
o^{(t)} &= c + Vh^{(t)} \\
\hat{y}^{(t)} &= \text{softmax}(o^{(t)})
\end{aligned}
$$

$(y^{(t)}$ =

$$
\begin{aligned}
& L\left(\{x^{(1)}, \ldots, x^{(\tau)}\}, \{y^{(1)}, \ldots, y^{(\tau)}\}\right) \\
&= \sum_t L^{(t)} \\
&= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{x^{(1)}, \ldots, x^{(t)}\}\right)
\end{aligned}
$$

Model Parameters

U, V, W, b and c

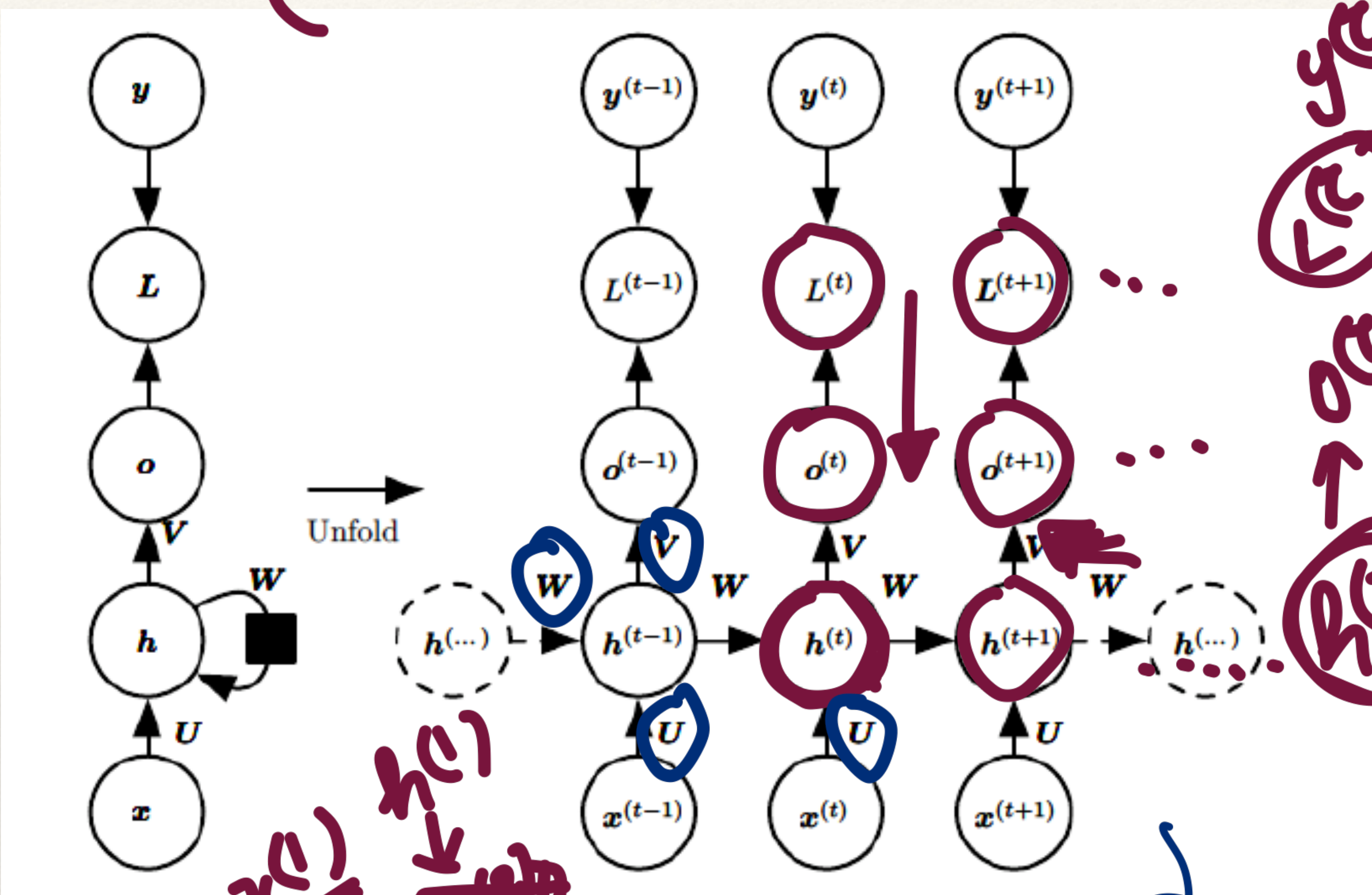
Gradient Descent

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

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

Recurrent Networks

(z)



$h^{(t)} \rightarrow h^{(t)} + \epsilon$

$i = y^{(t)}$

$y^{(t)}$ $L^{(t)}$ $o^{(t)}$ $h^{(\tau)}$

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

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

$h^{(t)}$ $x^{(t)}$ $h^{(t+1)}$

Backward Recurrence

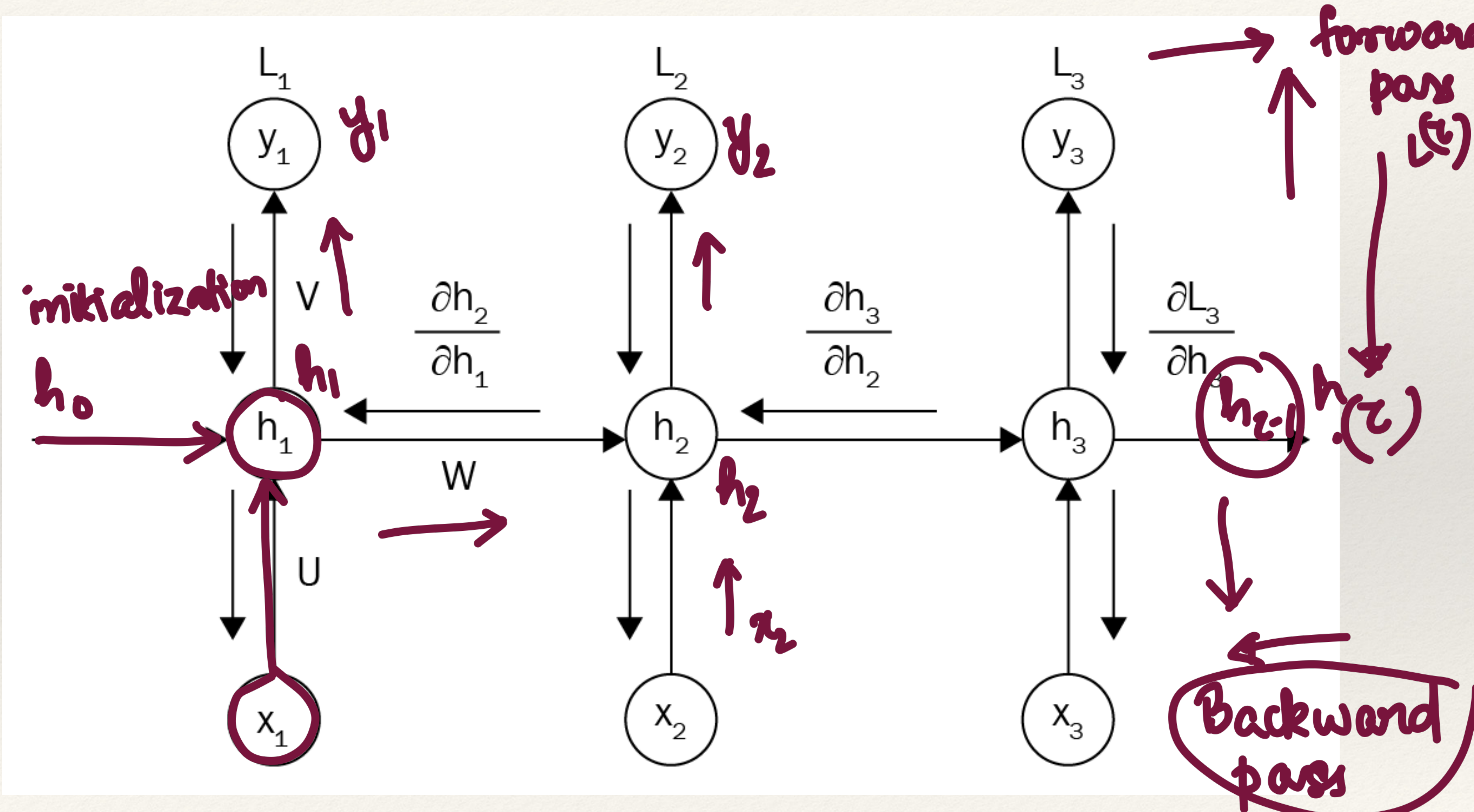
*

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

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

(BPTT)

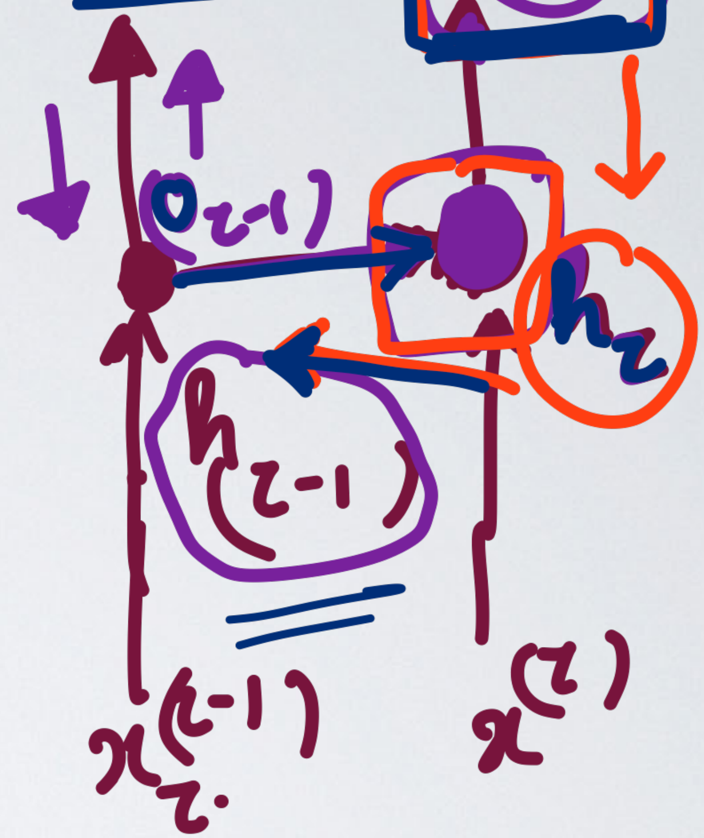
Back Propagation Through Time



$$\left[\frac{\partial L}{\partial h^{(\tau-1)}} \right]$$

$L = \ldots\ldots + L^{(\tau-1)} + L^{(\tau)}$

$\boxed{h_{\tau-1} + c}$



$O_\tau = V h_\tau + c$; $\hat{y}_\tau = \text{softmax}(O_\tau)$

$L^\tau = CE[y_\tau, \hat{y}_\tau]$

$$\frac{\partial L}{\partial h^\tau} = V^T \left(\frac{\partial L}{\partial O_\tau} \right)$$

$L^{(\tau)}$

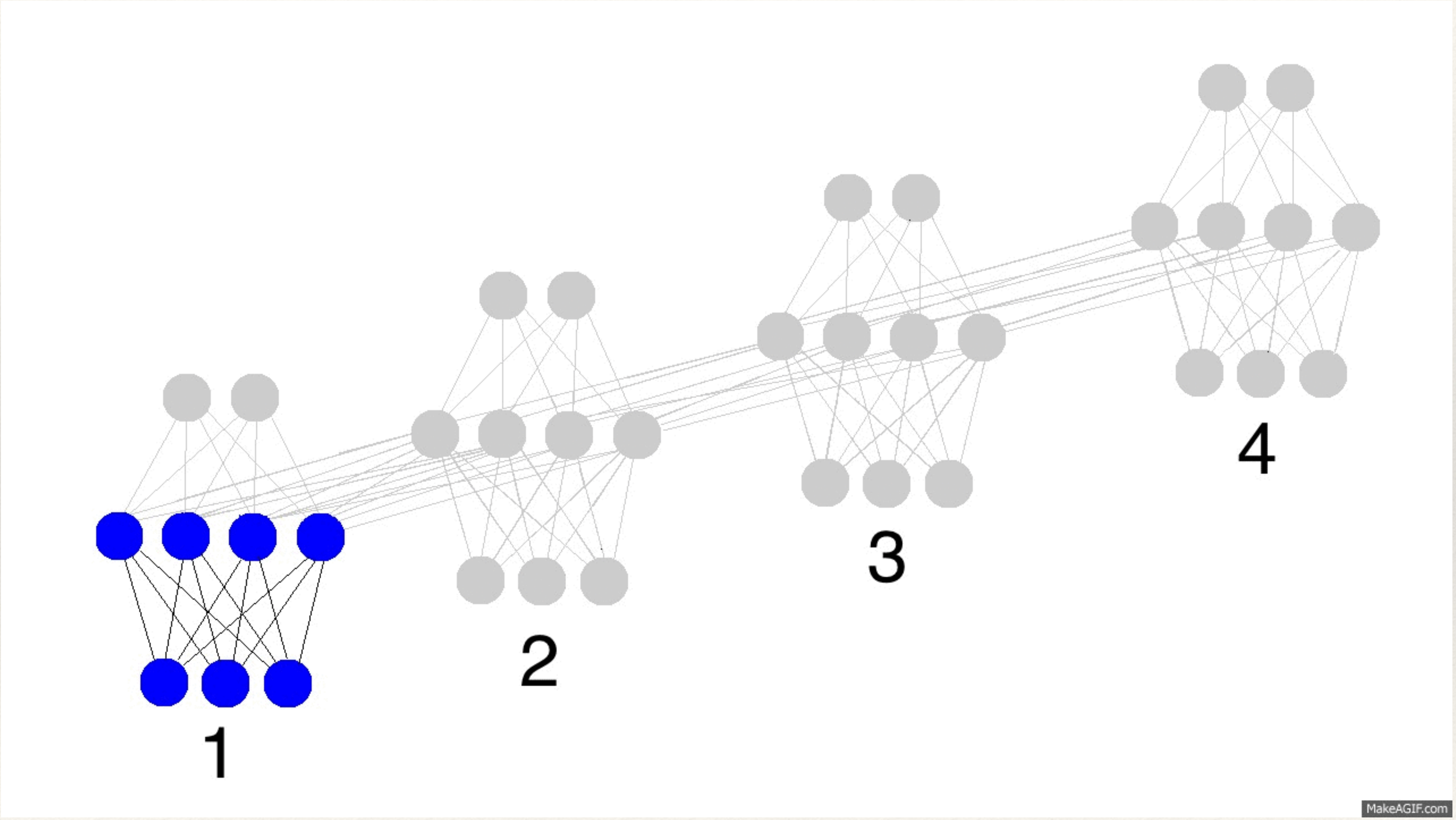
$$\boxed{\frac{\partial L}{\partial h^{(\tau-1)}}} = V^T \frac{\partial L}{\partial O_{\tau-1}} + W^T (\text{deriv})_{\tanh} \frac{\partial L}{\partial h^\tau}$$

$L^{(t)}$ $\quad$ (L)

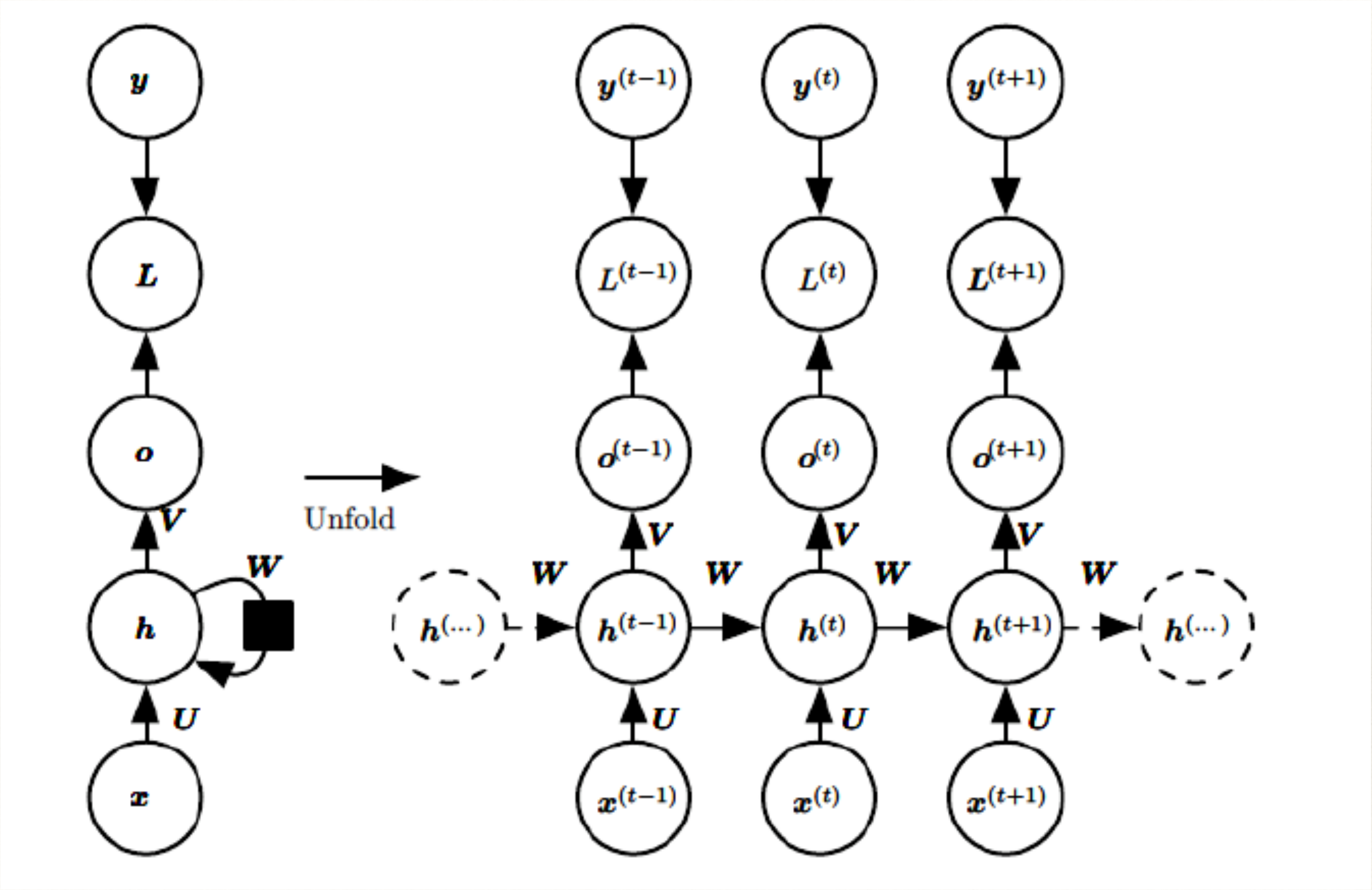
$$h^{(\tau)} = \tanh(W h^{(\tau-1)} + U x^{(\tau)} + b)$$

$h_t \longrightarrow h_{t+1}$

Back Propagation Through Time

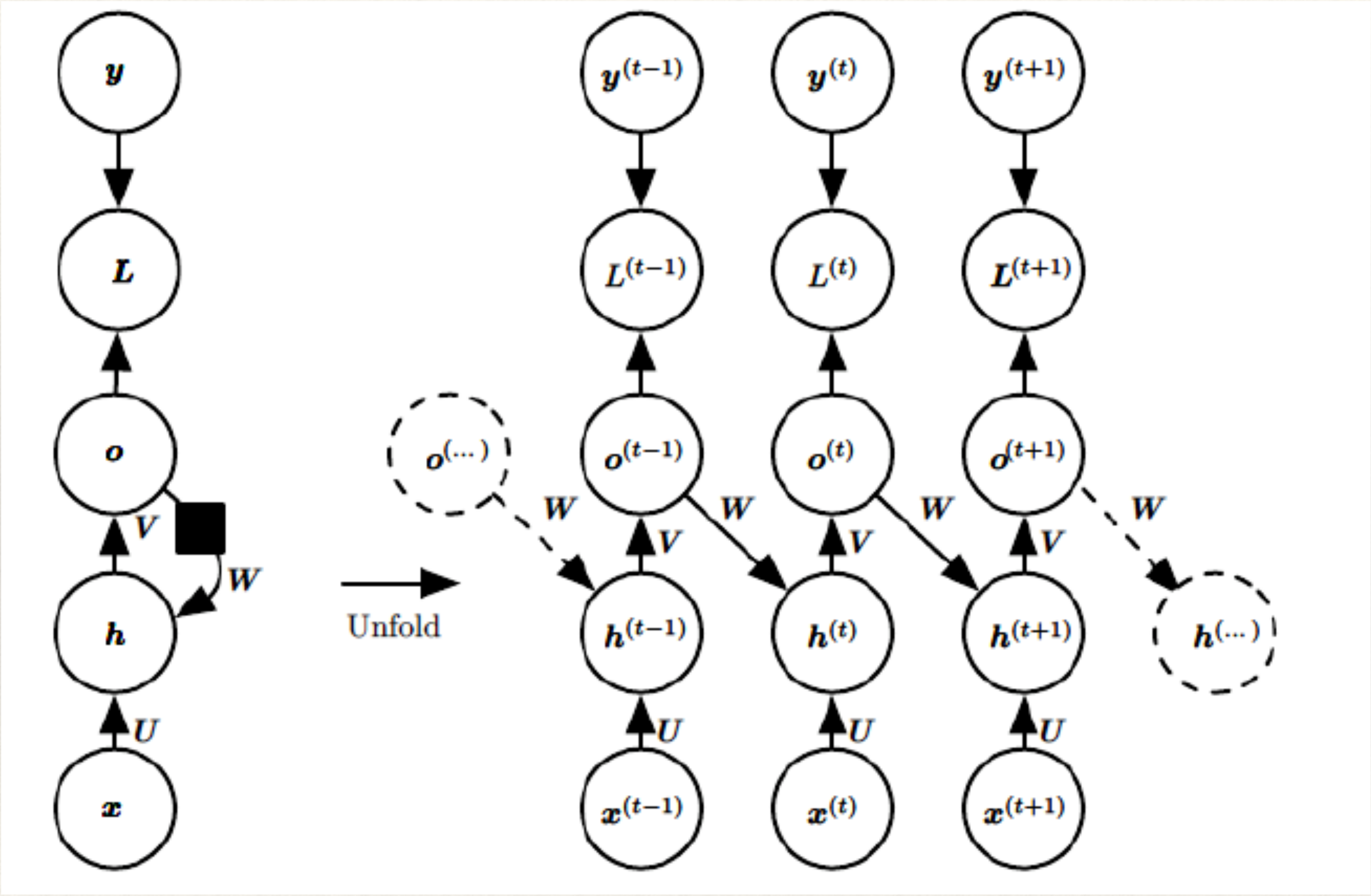


Standard Recurrent Networks



"Deep Learning", Ian Goodfellow, Yoshua Bengio, Aaron Courville

Other Recurrent Networks



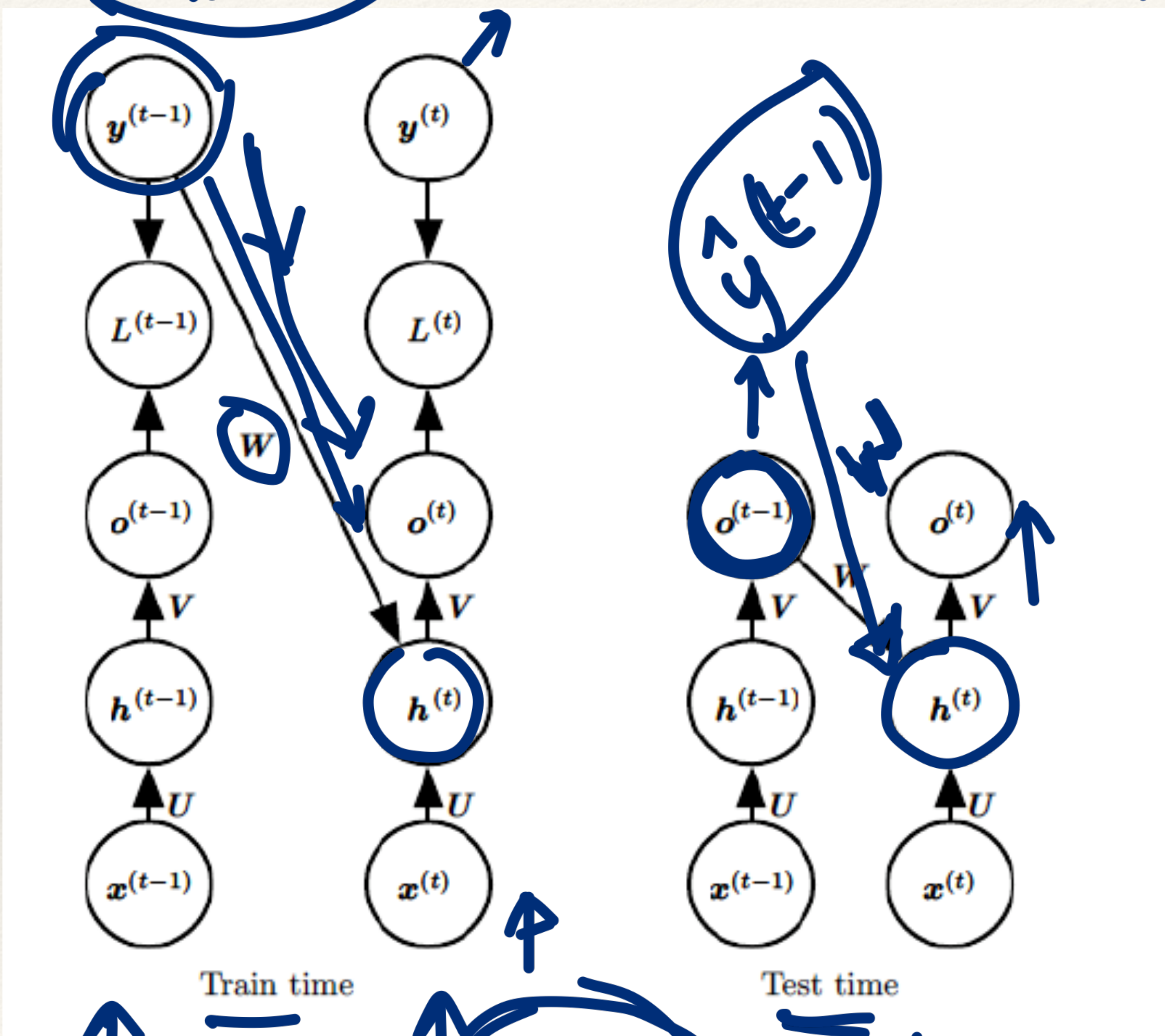
Teacher Forcing Networks

"Deep Learning", Ian Goodfellow, Yoshua Bengio, Aaron Courville

Recurrent Networks

Sentiment

Sentiment + − neutral



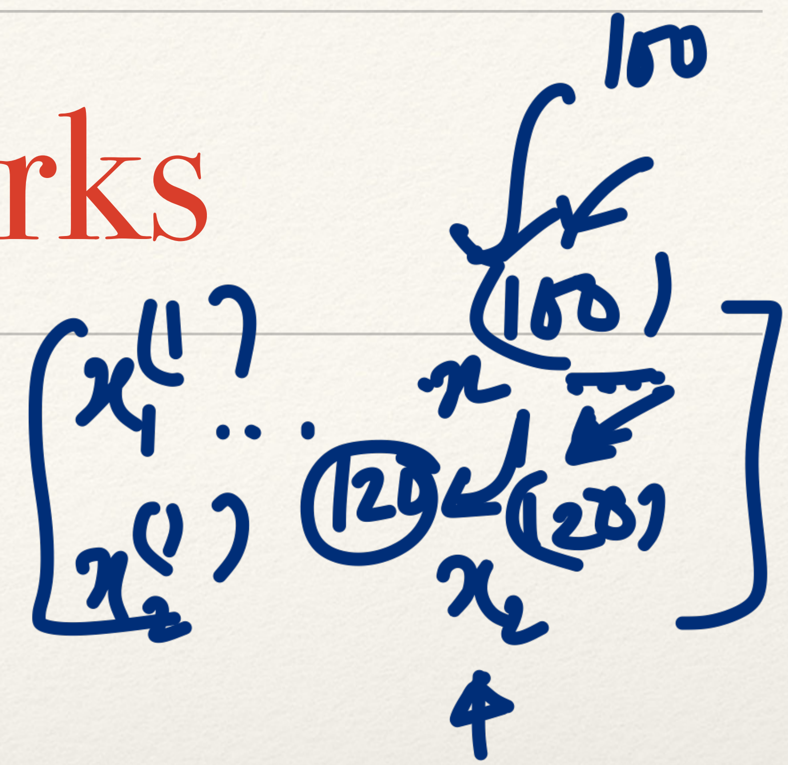
Teacher Forcing Networks

previous embedding

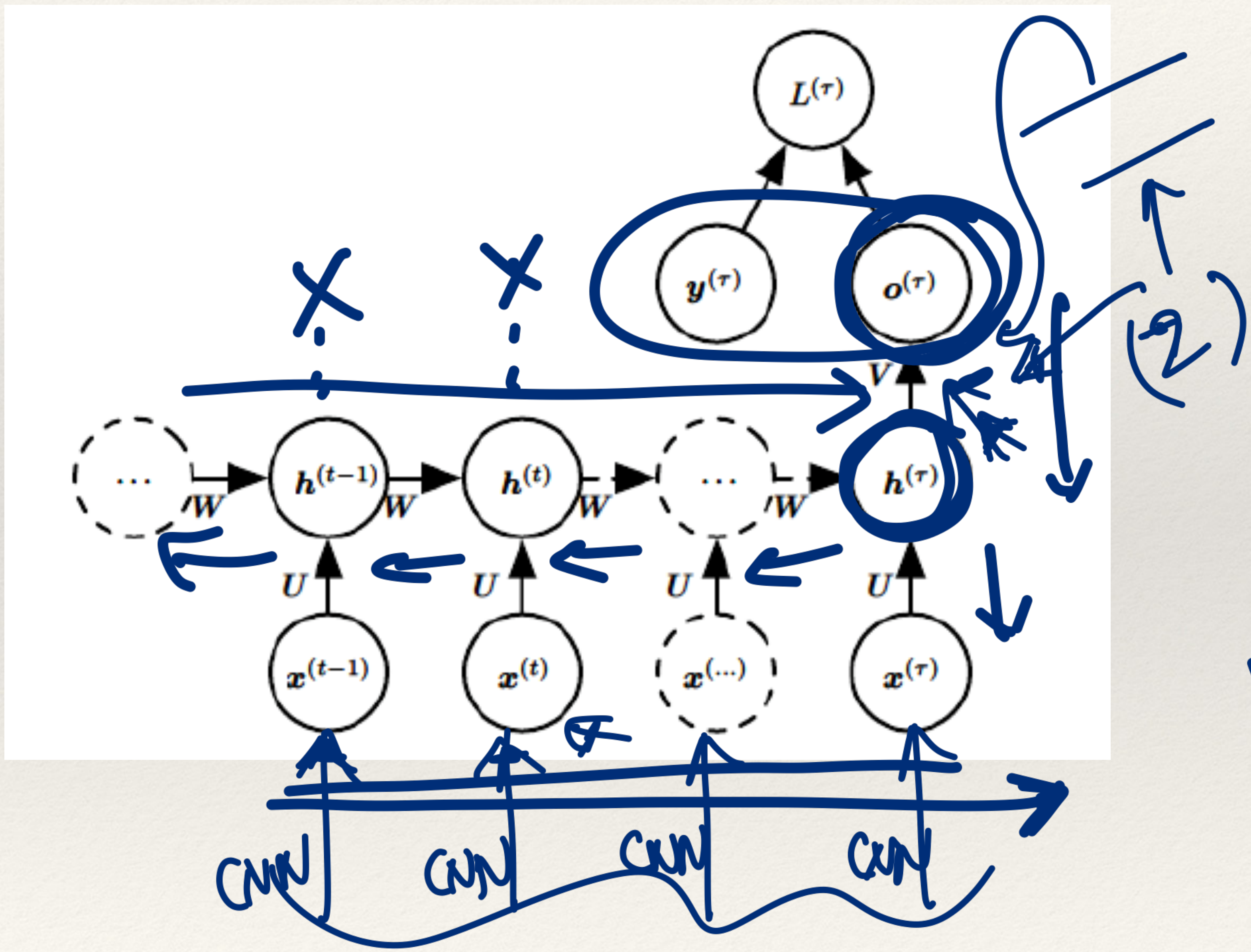
current word embedding

“Deep Learning”, Ian Goodfellow, Yoshua Bengio, Aaron Courville

Recurrent Networks



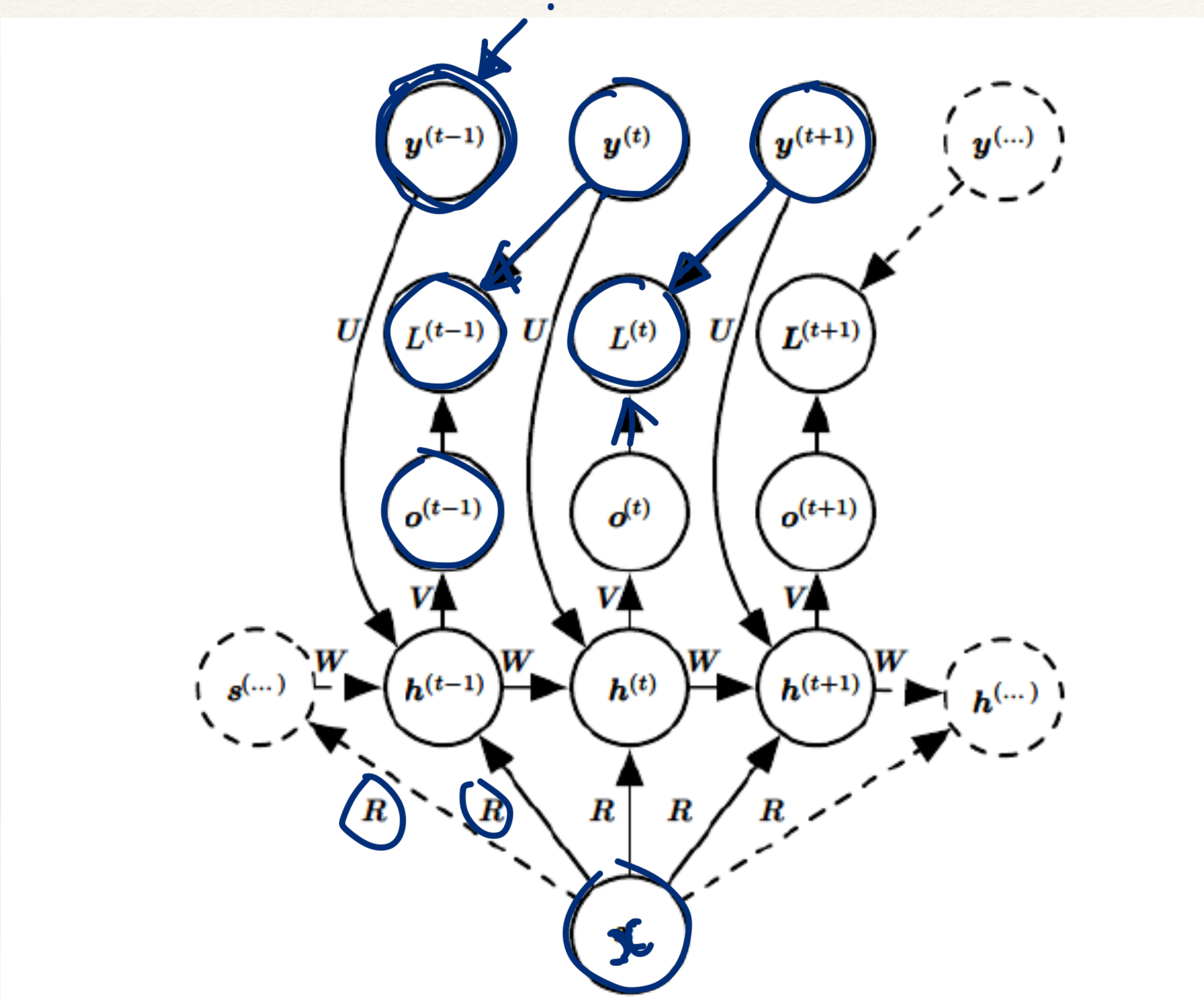
sentiment



Multiple Input Single Output

Variable length inputs.

Recurrent Networks



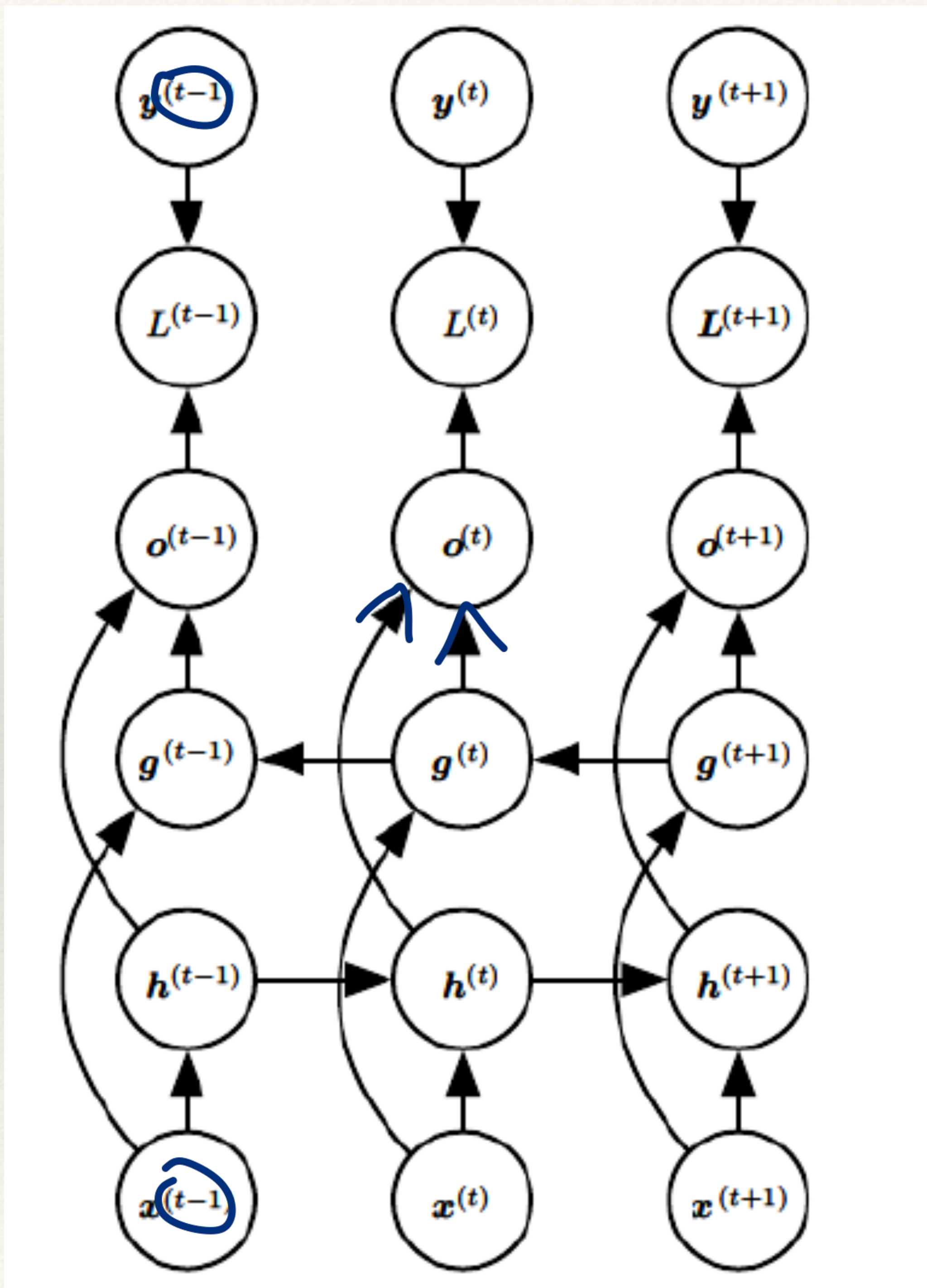
(y_t)

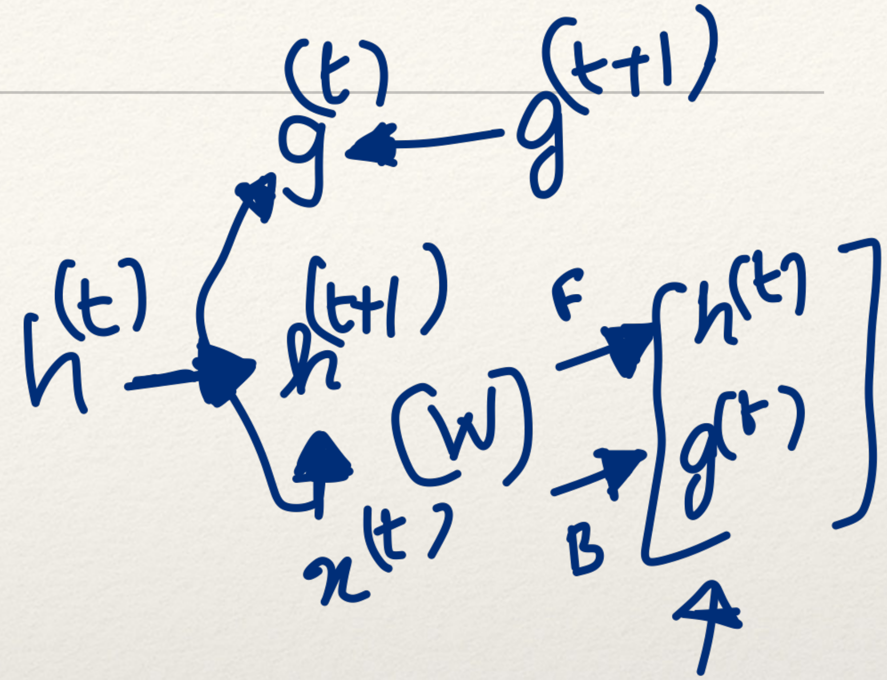
$y_1 \; y_2 \cdots y_{t-1}$ x

Single Input Multiple Output

Autoregressive Models

Recurrent Networks





Bi-directional Networks

$[W_1\ W_2]\begin{bmatrix}h\\g\end{bmatrix}$

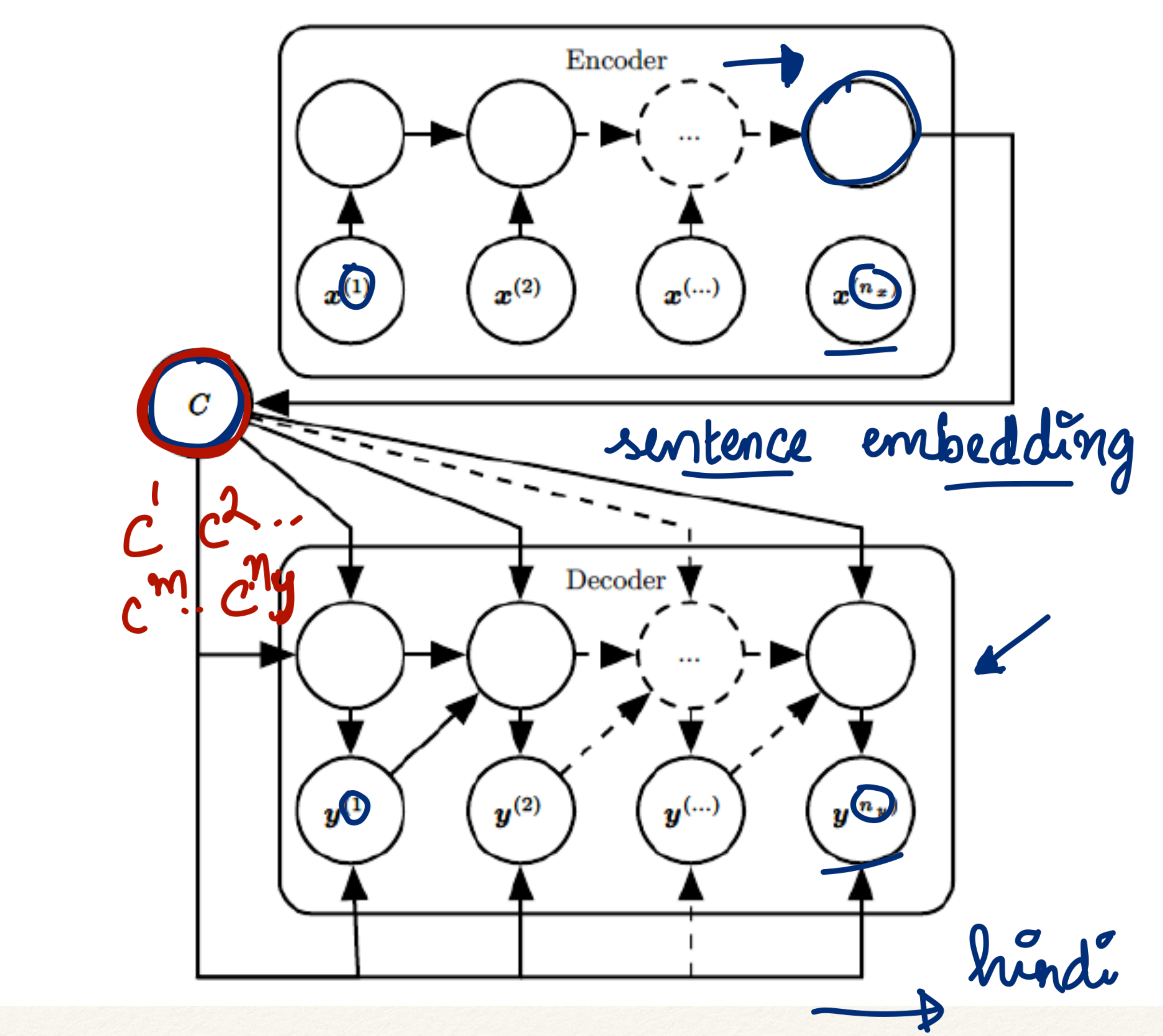
multiple input

single output

C A T

B A R

Recurrent Networks



$x^{(t)} \rightarrow y^{(t)}$

$x^{(t)} \rightarrow (y^m)$

Sequence to Sequence Mapping Networks

Encoder ← sequence to vector

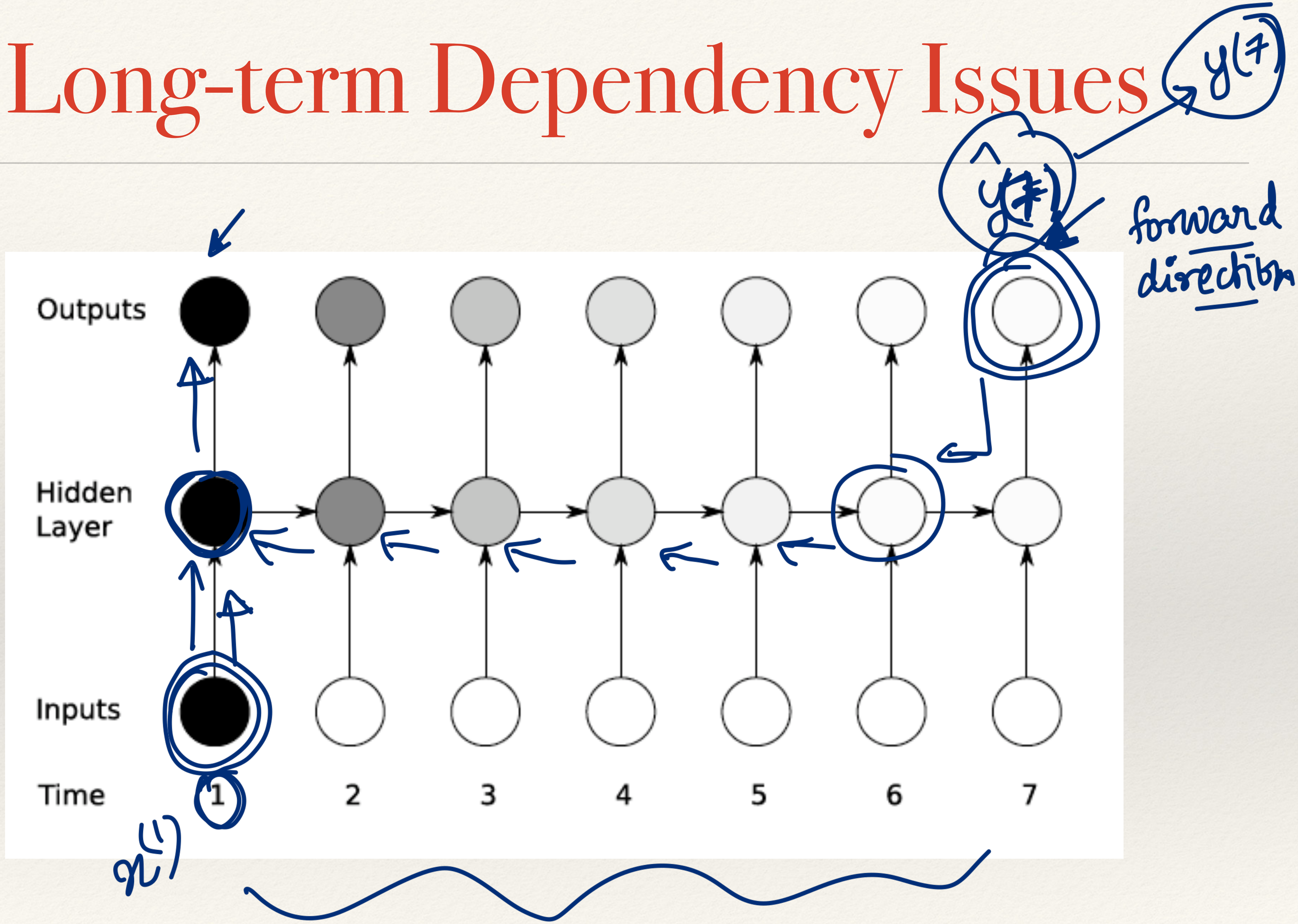
Decoder ← vector to a new sequence

$$X = \{ x^{(1)} \ldots\ldots\ldots x^{(T)} \} \rightarrow \{ \text{Eng Senten} \}$$

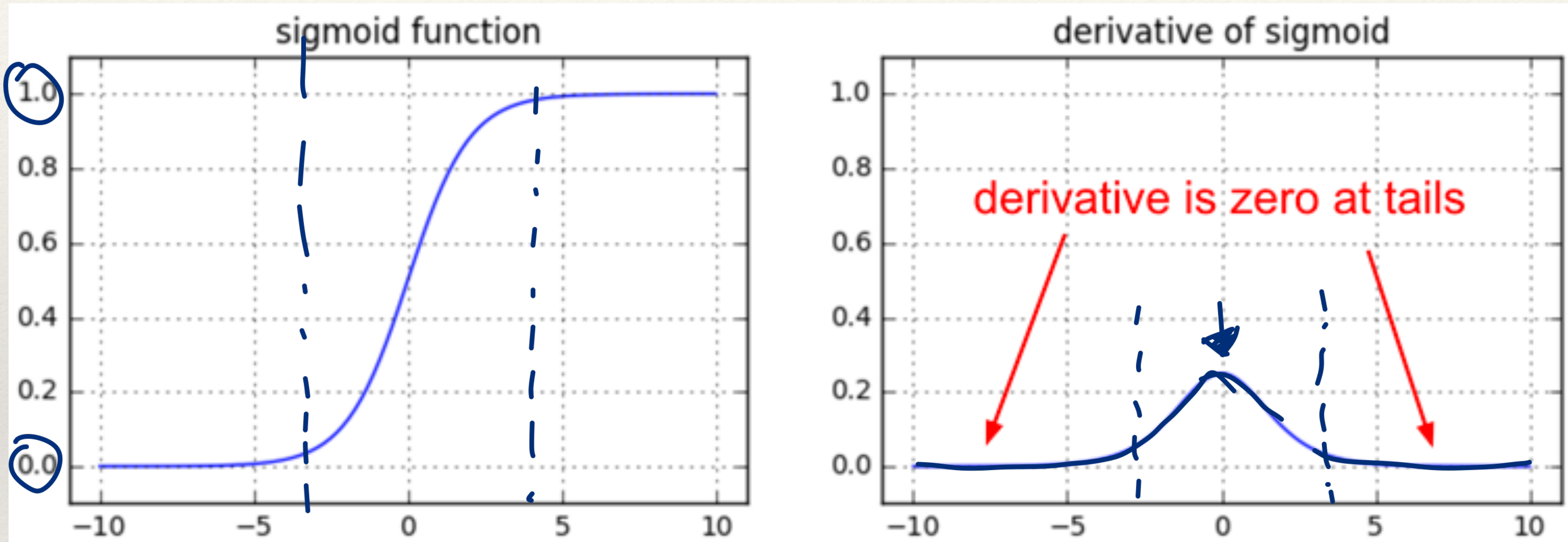
$$Y = \{ y^{(1)} \ldots\ldots y^{(M)} \} \rightarrow \{ \text{Hindi Sentence} \}$$

$$X \longrightarrow \boxed{\begin{array}{l}\text{Encoder}\\ \text{Decoder}\end{array}} \longrightarrow Y$$

Long-term Dependency Issues

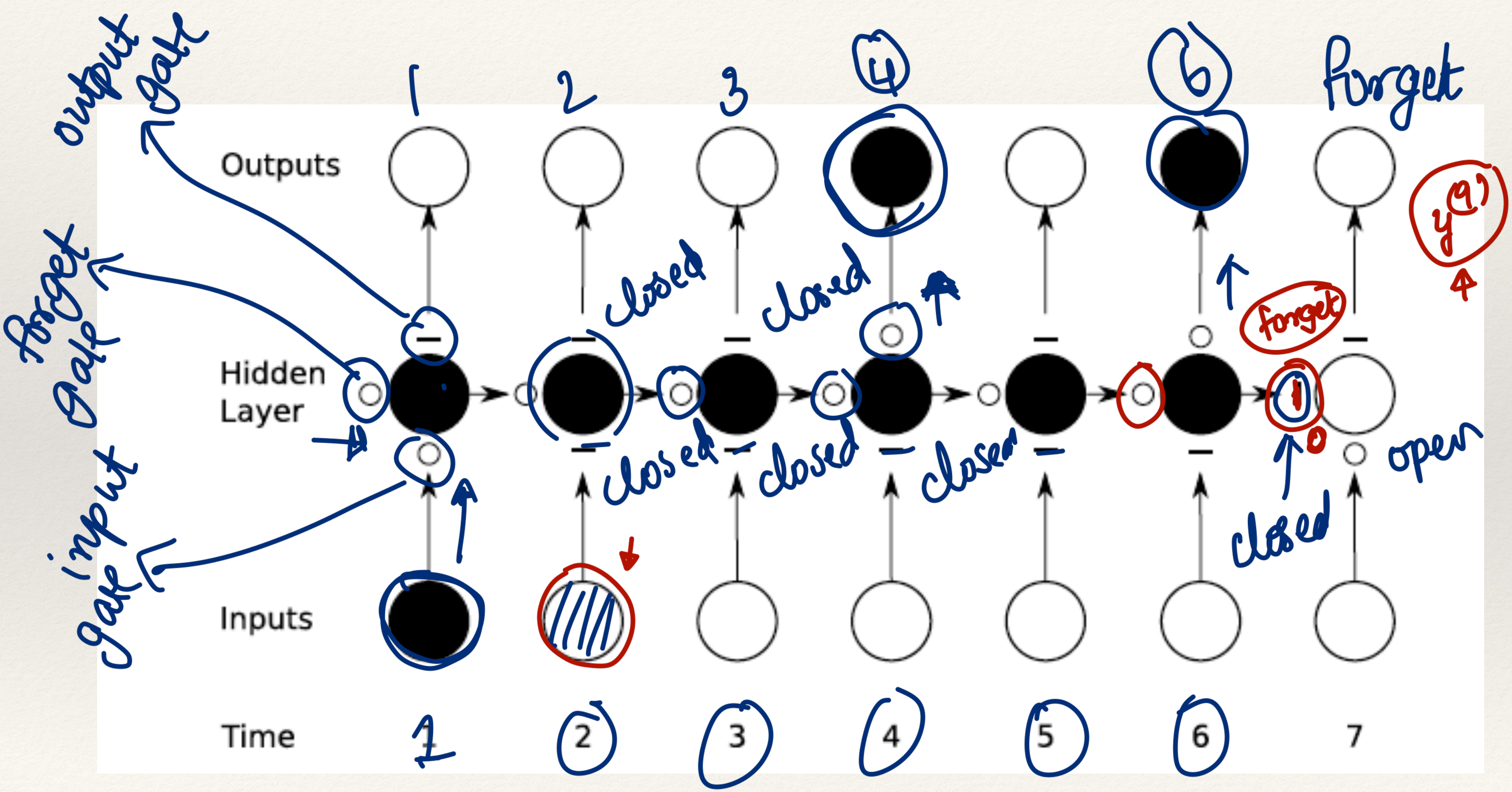


Vanishing/Exploding Gradients



- Gradients either vanish or explode
    - Initial frames may not contribute to gradient computations or may contribute too much.

Long-Short Term Memory



LSTM Cell

f - sigmoid function
g, h - tanh function

Input Gate

$$a_\iota^t = \sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} b_h^{t-1} + \sum_{c=1}^{C} w_{c\iota} s_c^{t-1}$$

$$b_\iota^t = f(a_\iota^t)$$

Forget Gate

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1}$$

$$b_\phi^t = f(a_\phi^t)$$

Cell

$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_\iota^t g(a_c^t)$$

Output Gate

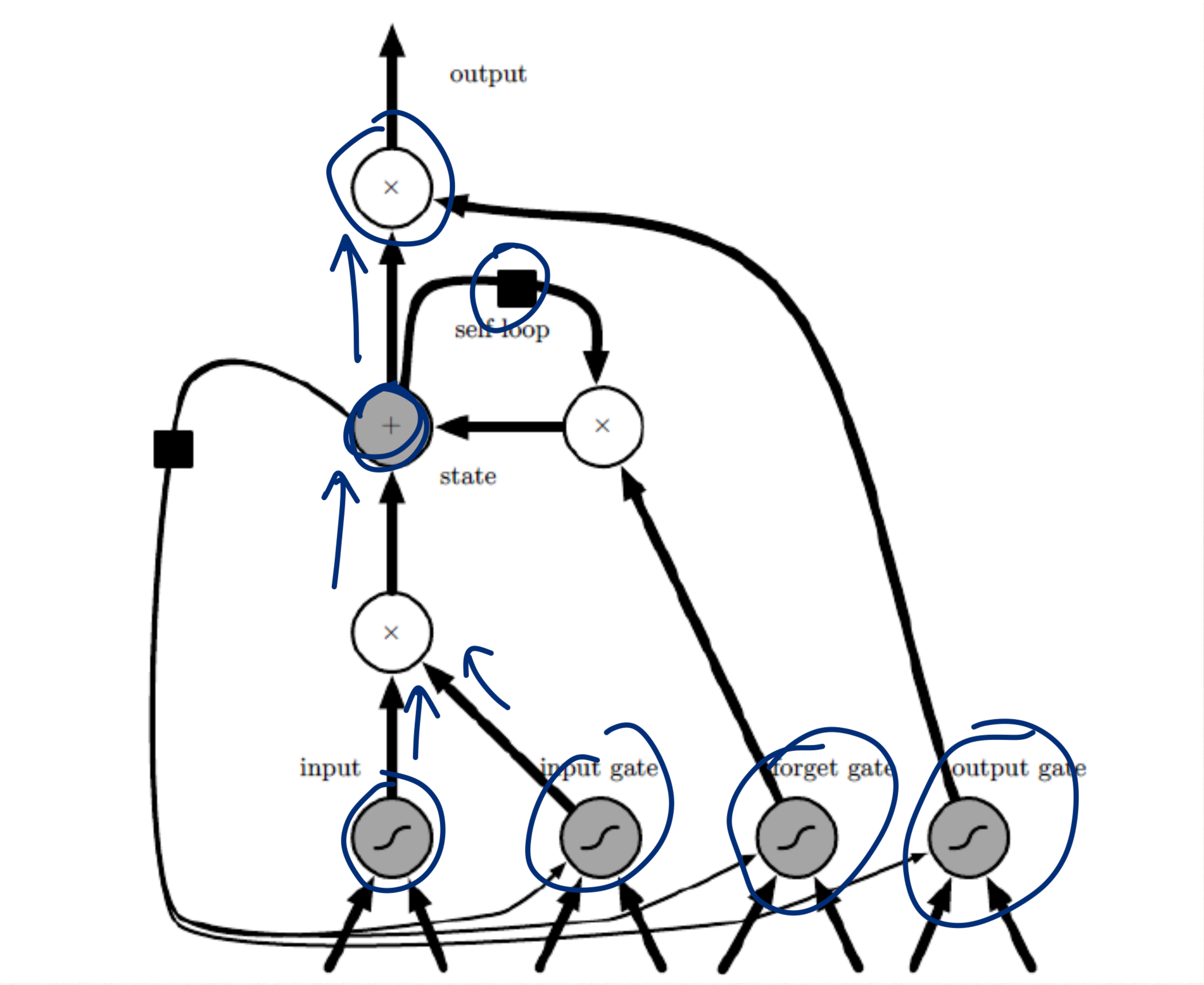
$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega} x_i^t + \sum_{h=1}^{H} w_{h\omega} b_h^{t-1} + \sum_{c=1}^{C} w_{c\omega} s_c^t$$

$$b_\omega^t = f(a_\omega^t)$$

LSTM output

$$\boxed{b_c^t = b_\omega^t h(s_c^t)}$$

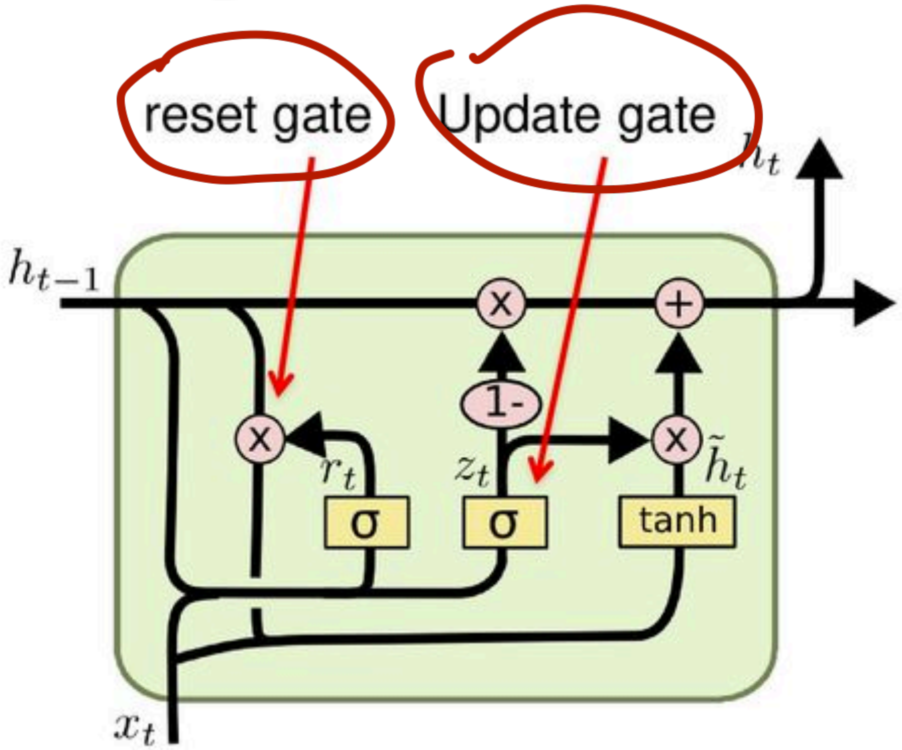
Long Short Term Memory Networks

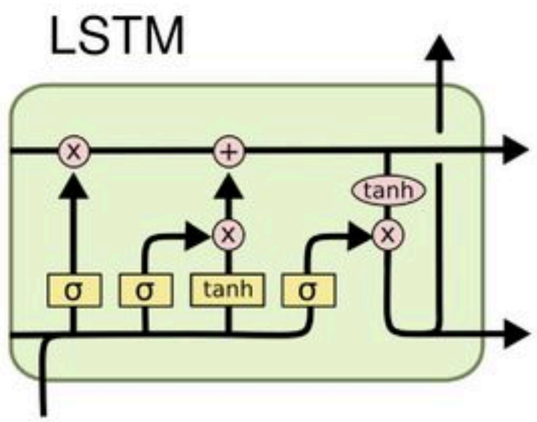


Gated Recurrent Units (GRU)

GRU – gated recurrent unit

(more compression)





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

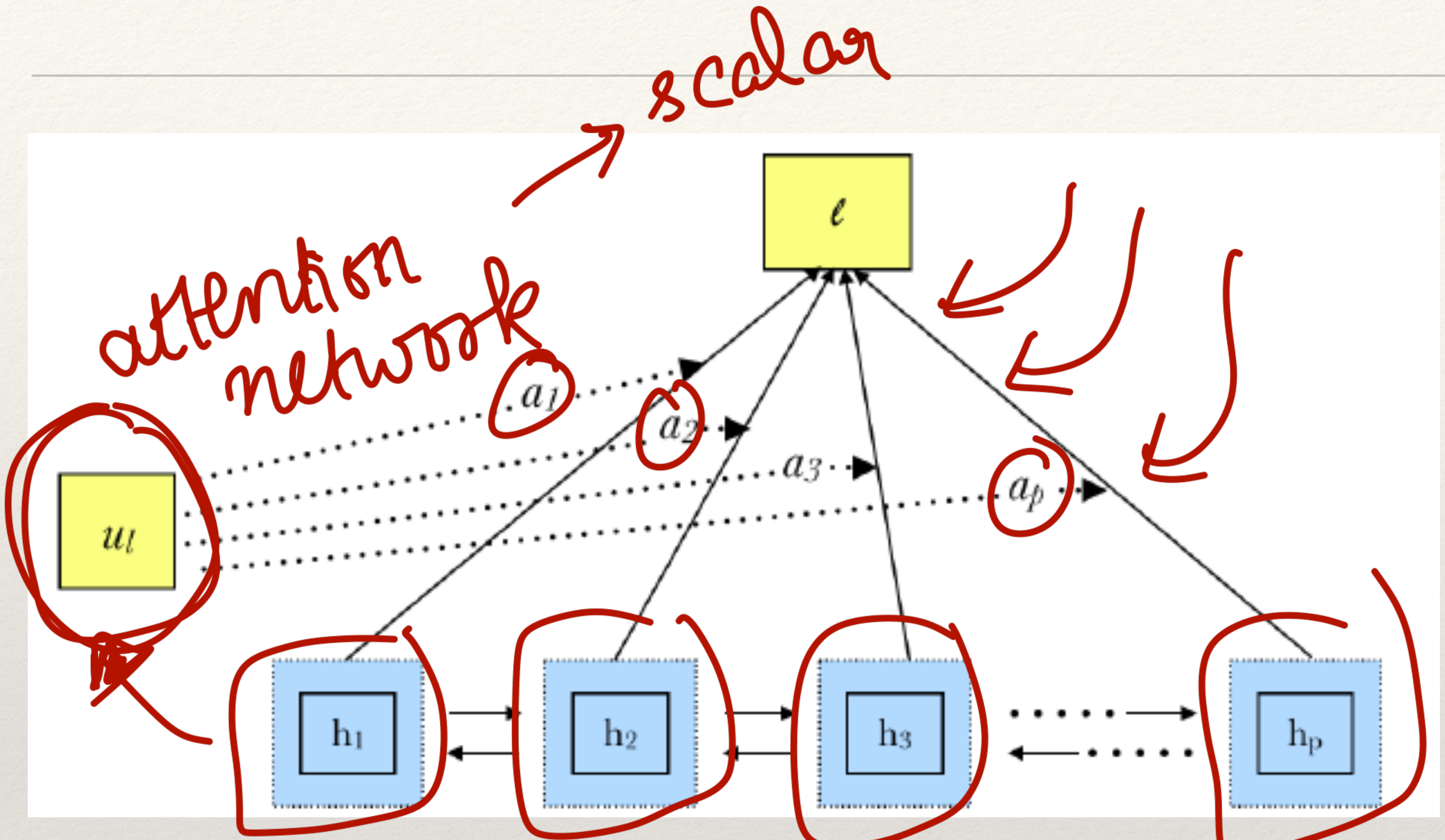
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

It combines the forget and input into a single update gate. It also merges the cell state and hidden state. This is simpler than LSTM. There are many other variants too.

X,*: element-wise multiply

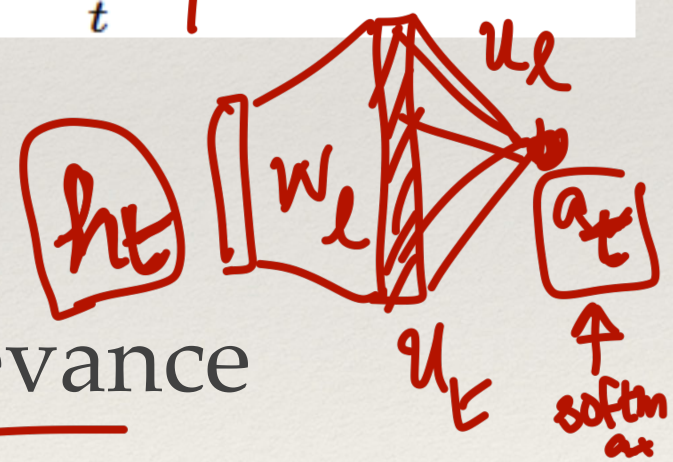
Attention in LSTM Networks



$$\mathbf{u}_t = tanh(\mathbf{W}_l \mathbf{h}_t + b_l)$$

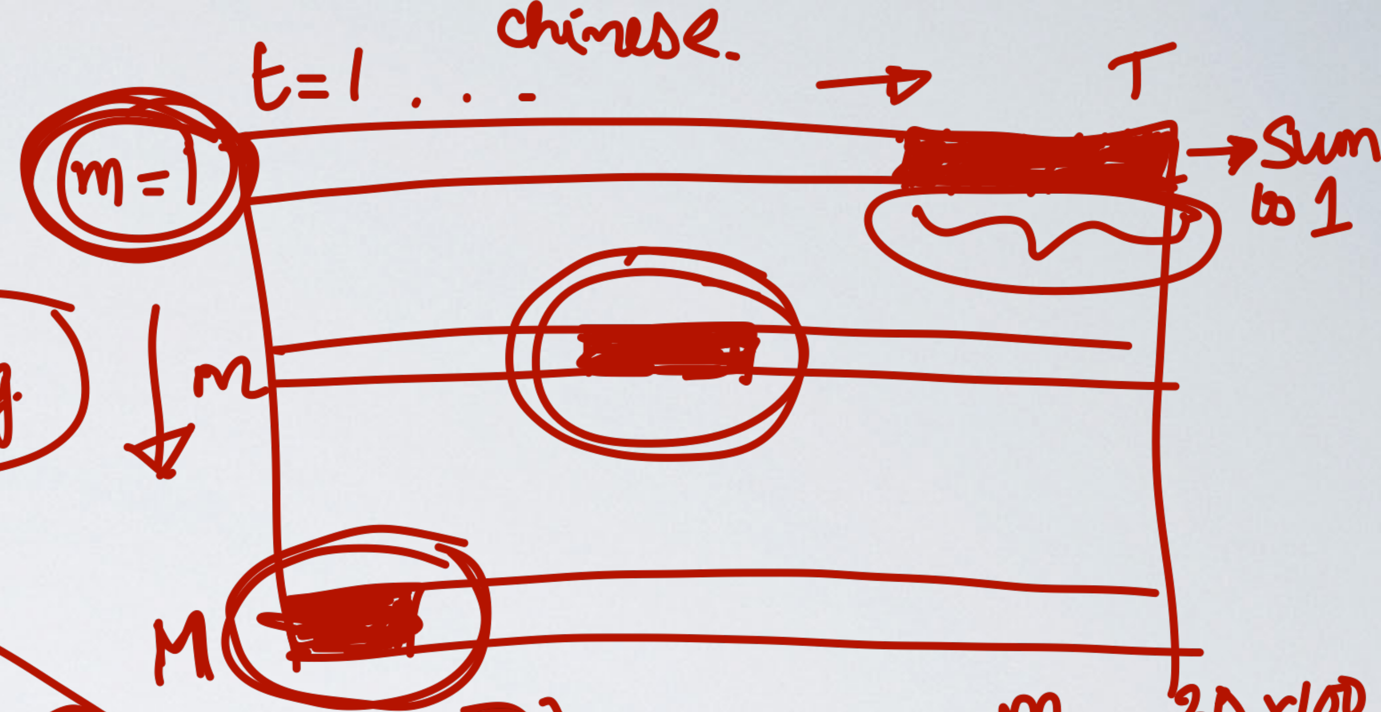
$$a_t = \frac{exp(\mathbf{u}_t^T \mathbf{u}_l)}{\sum_t exp(\mathbf{u}_t^T \mathbf{u}_l)}$$

$$l = \sum_t a_t \mathbf{h}_t$$



- Attentions allows a mechanism to add relevance
  - Certain regions of the audio have more importance than the rest for the task at hand.

Attention



a_t^m

T=100

M=30

$\{x_1 \ldots x_T\}$

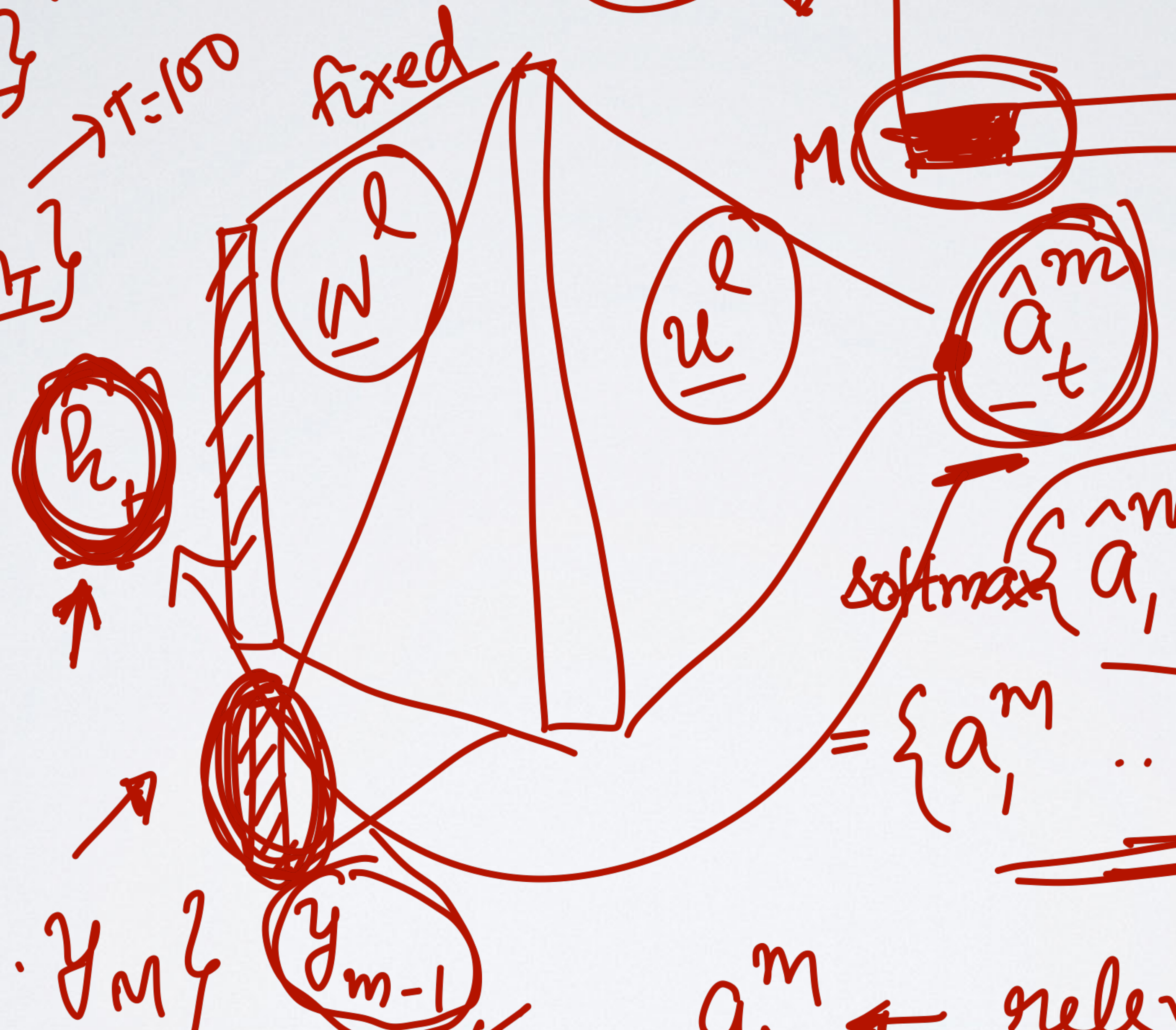
recurrent

$\{h_1 \ldots h_T\} \rightarrow T=100$

h_1

$\vdots$

h_T



$\text{softmax}\{\hat{a}_1^m \ldots \hat{a}_T^m\}$

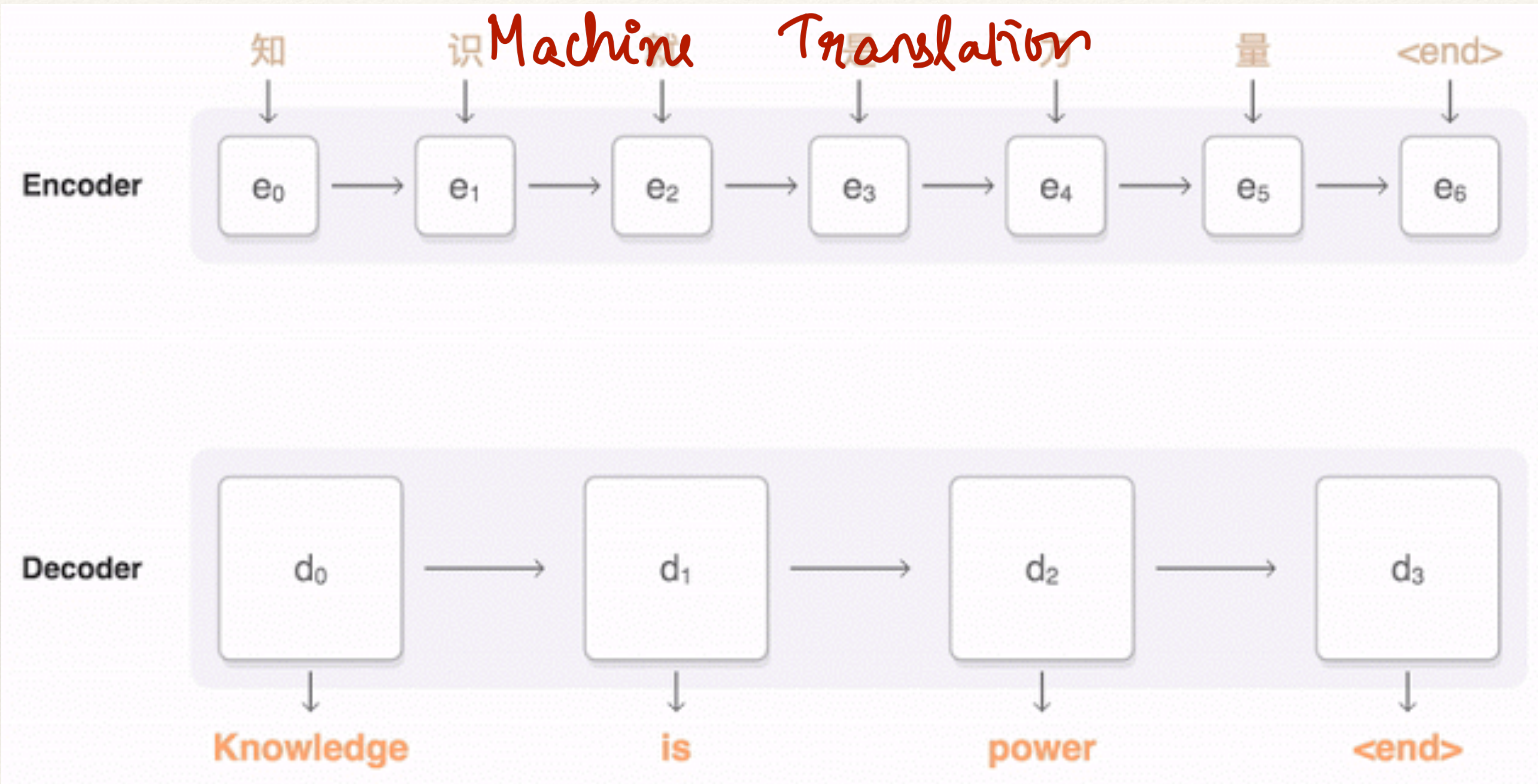
$= \{a_1^m \ldots\ldots a_T^m\}$ 100 $\underline{a}$

$\{y_1 \ldots y_M\}$ y_{m-1}

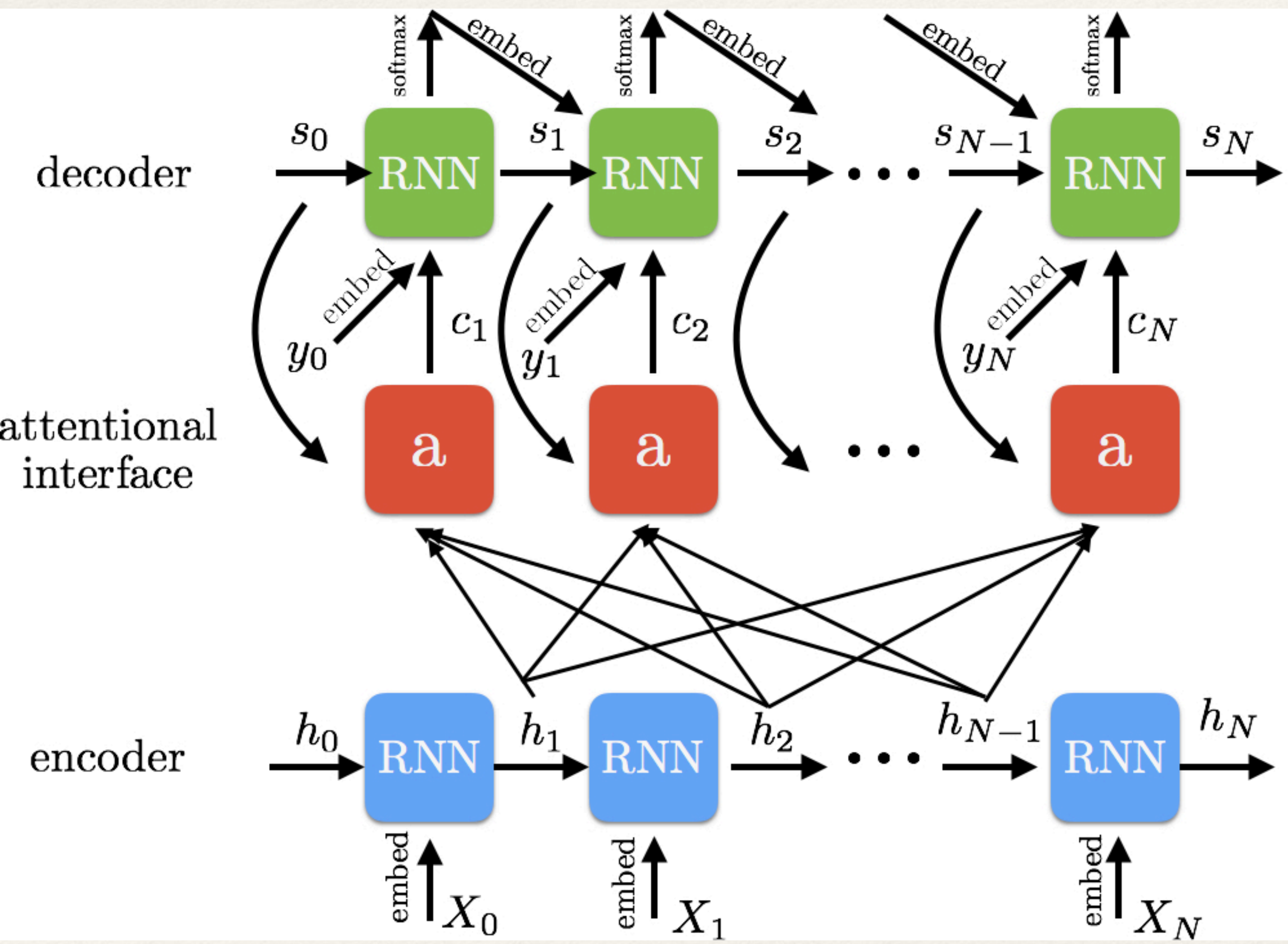
$$c^m = \sum_{t=1}^{T} h_t \, a_t^m$$

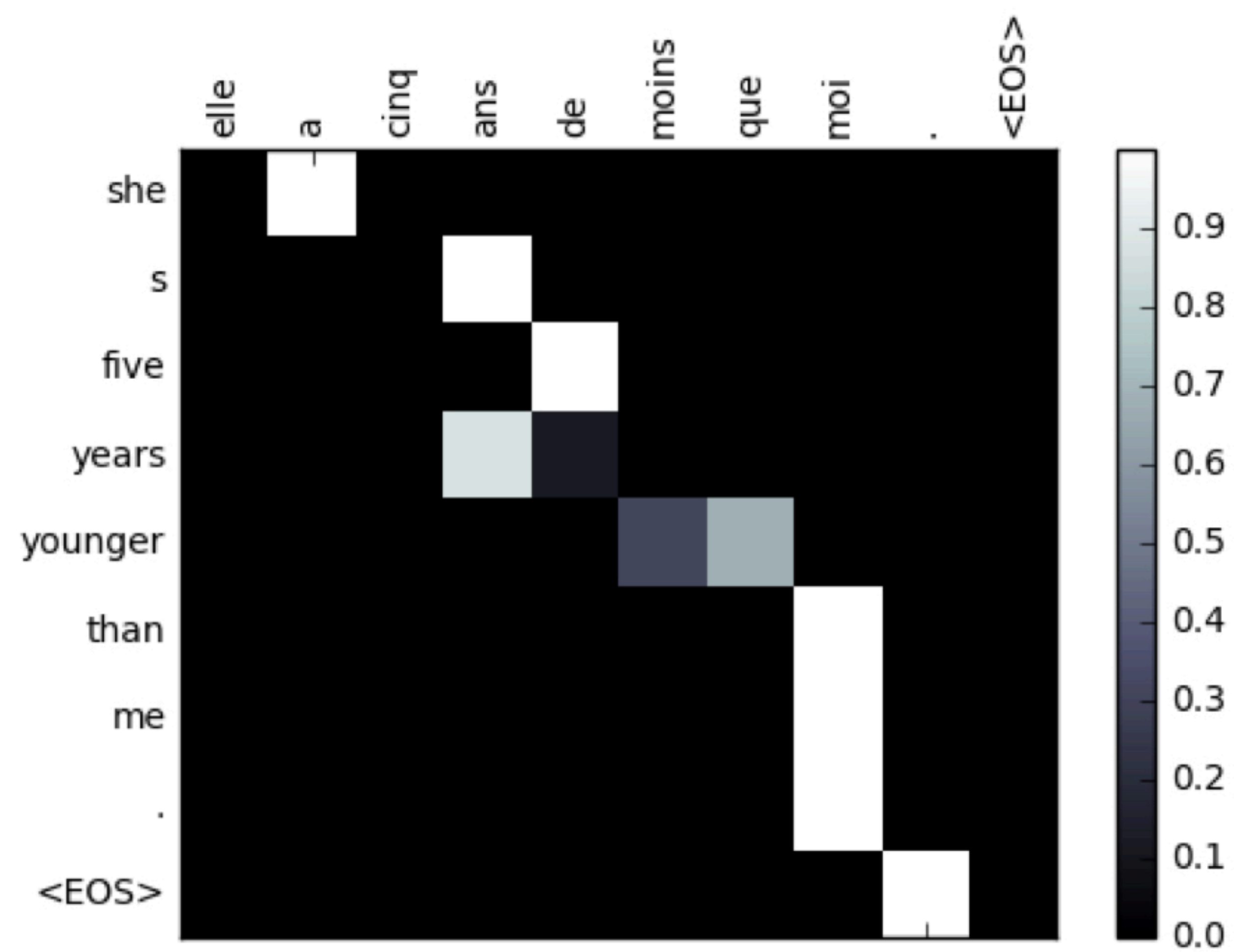
$a_t^m \leftarrow$ relevance of t^{th} input for generating m^{th} output

Encoder - Decoder Networks with Attention



Attention Models





Attention - Speech Example

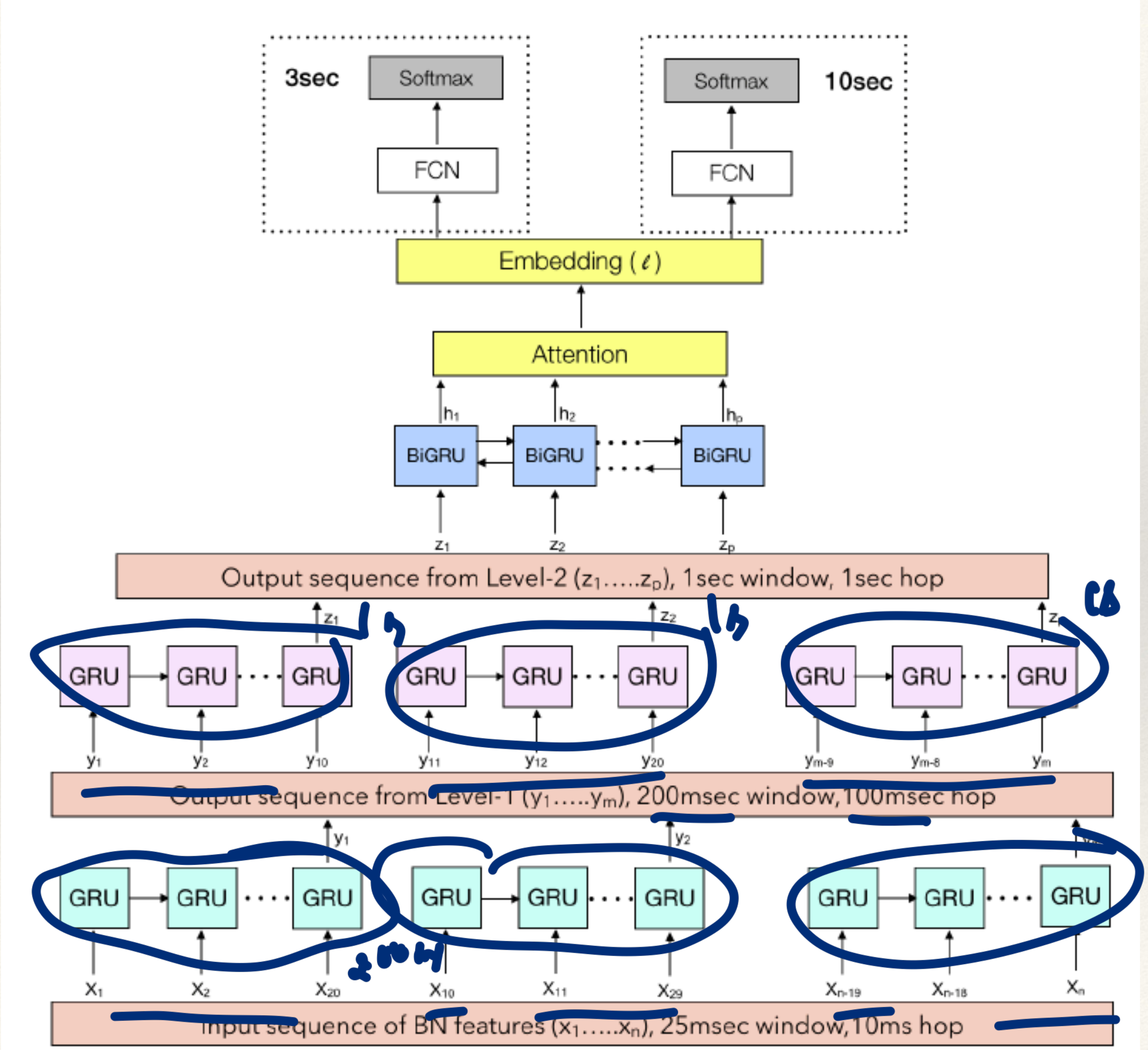
From our lab [part of ICASSP 2019 paper].

Language Recognition Evaluation

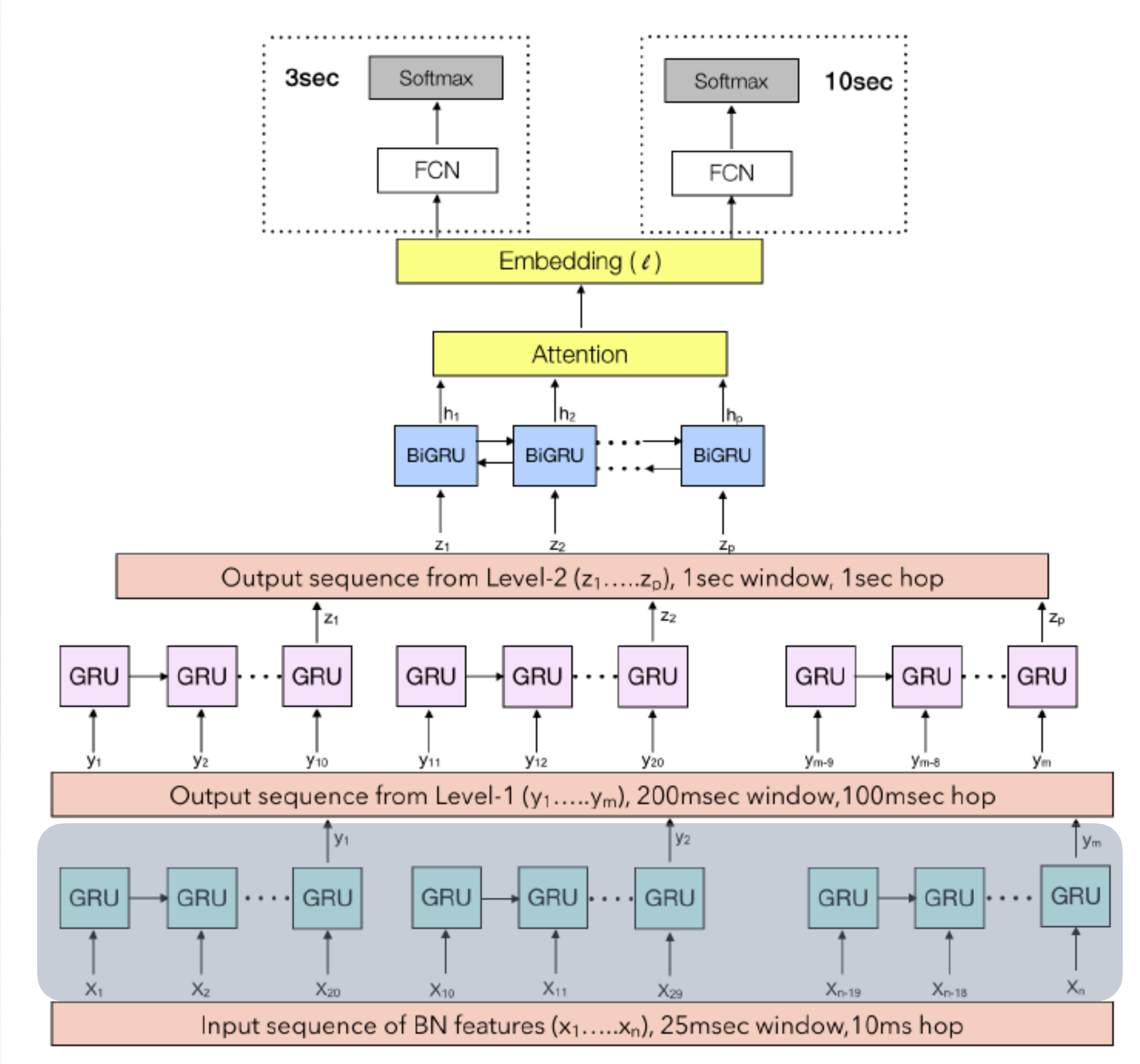
Table 1: LRE17 training set : target languages, language clusters and total number of hours.

Cluster	Target Languages	Hours
Arabic	Egyptian Arabic (ara-arz)	190.9
	Iraqi Arabic (ara-acm)	130.8
	Levantine Arabic (ara-apc)	440.7
	Maghrebi Arabic (ara-ary)	81.8
Chinese	Mandarin (zho-cmn)	379.4
	Min Nan (zho-nan)	13.3
English	British English (eng-gbr)	4.8
	General American English (eng-usg)	327.7
Slavic	Polish (qsl-pol)	59.3
	Russian (qsl-rus)	69.5
Iberian	Caribbean Spanish (spa-car)	166.3
	European Spanish (spa-eur)	24.7
	Latin American Continental Spanish (spa-lac)	175.9
	Brazilian Portuguese (por-brz)	4.1

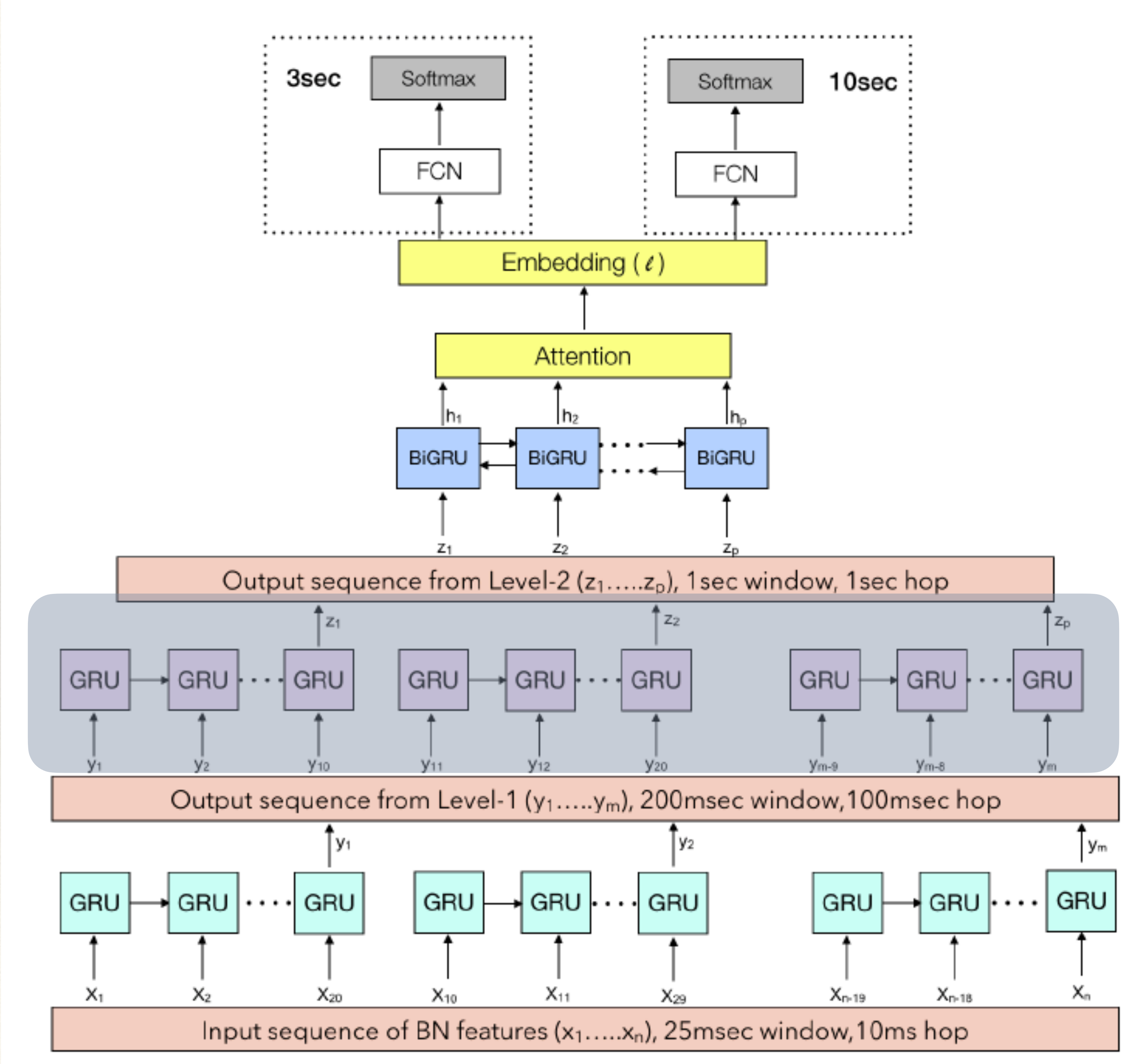
End-to-end model using GRUs and Attention



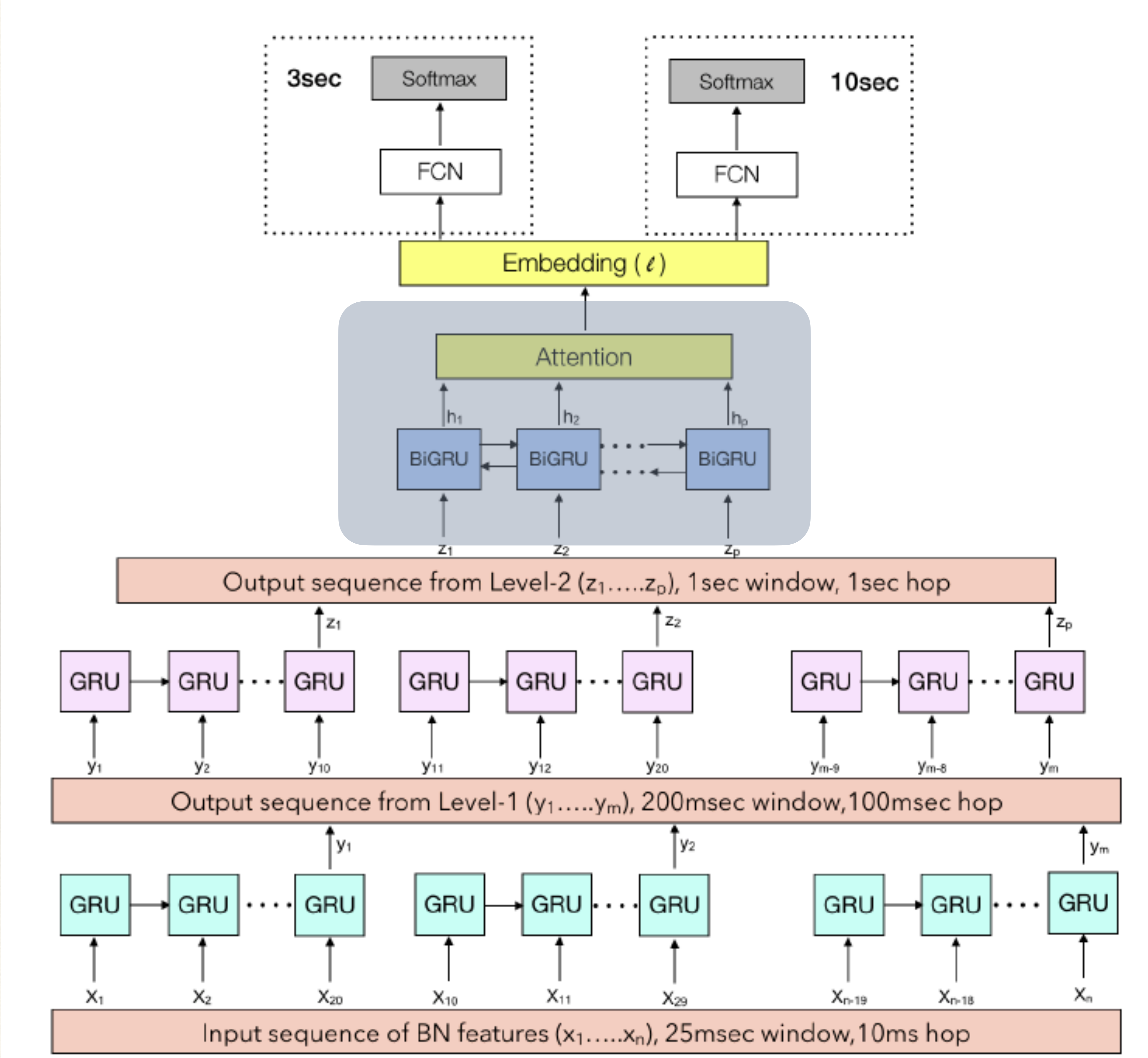
Proposed End-to-End Language Recognition Model



Proposed End-to-End Language Recognition Model



Proposed End-to-End Language Recognition Model



Language Recognition Evaluation

- State-of-art models use the input sequence directly.
- We proposed the attention model - Attention weighs th importance of each short-term segment feature for the task.

Attention Weight

0-3s : O...One muscle at all, it was terrible

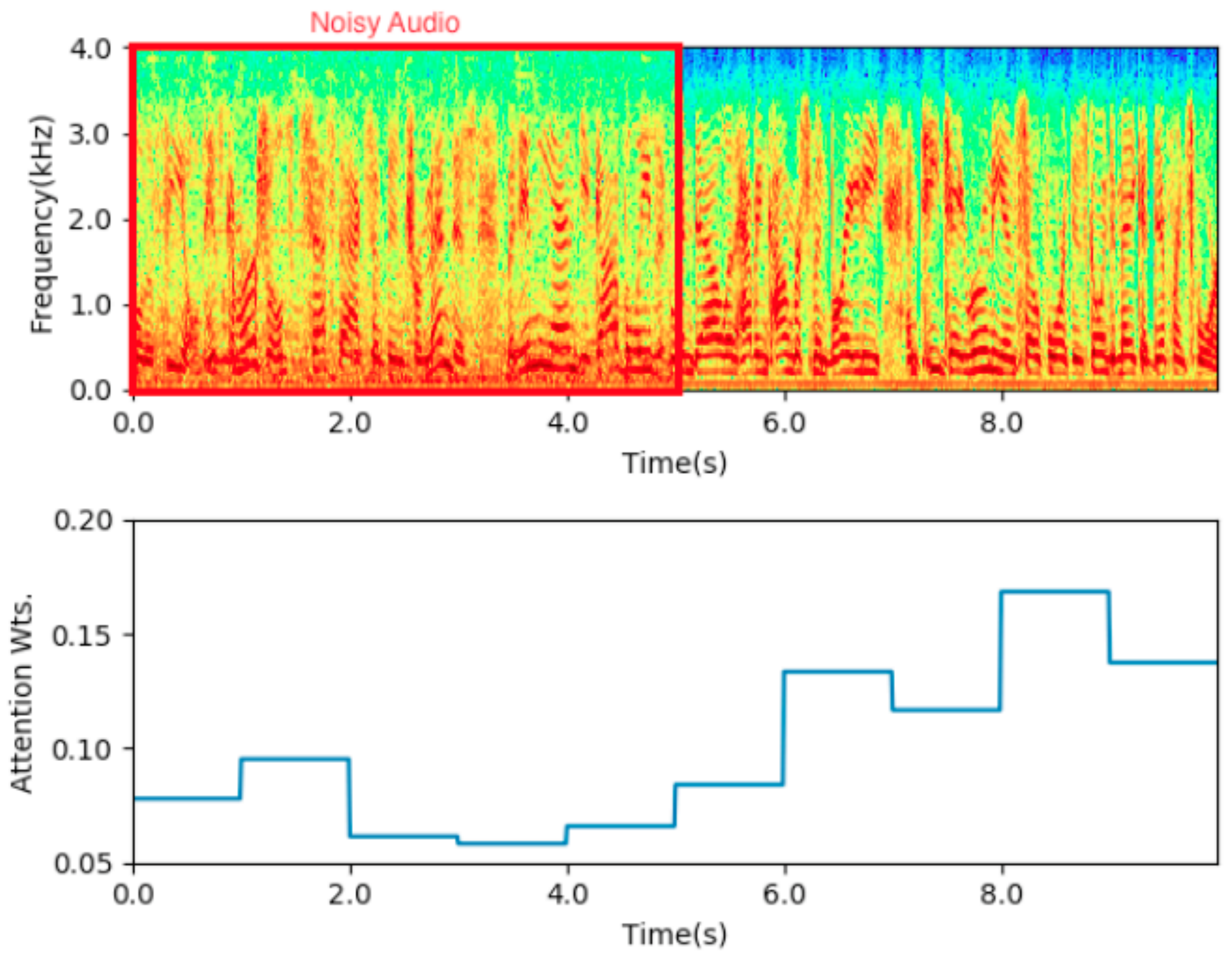
3s-4s : .... ah .... ah ....

4s - 9s : I couldn't scream, I couldn't shout, I couldn't even move my arms up, or my legs

9s -11s : I was trying me hardest, I was really really panicking.

Bharat Padi, et al. “End-to-end language recognition using hierarchical gated recurrent networks”, under review 2018.

Language Recognition Evaluation



Language Recognition Evaluation

Table 3. Approximate computational time in seconds for ten 30sec eval files using a single CPU. Machine Specification: 32 CPU, 8 core, 2 thread Intel x86_64 machine with 16 GB Nvidia Quadro P5000 GPU cards.

	ivec. [19]	LSTM [16]	HGRU
CPU	12	51	8
GPU	12	11.5	1.5

Table 4. LID accuracy in % for additional experiments with multiple speakers speaking the same language and the experiments without any SAD information.

Cond.	i-vec. [19]	HGRU
Multi-Speaker	60.6	**67.7**
Without SAD information	49.7	**52.7**