DSE 3121 DEEP LEARNING

Sequence Models - RNN

Dr. Rohini Rao & Dr. Abhilash K Pai

Dept. of Data Science and Computer Applications

MIT Manipal

Examples of Sequence Data

- Speech Recognition

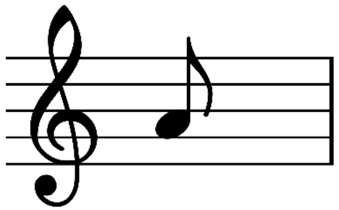
Mary had a little lamb

- Music Generation
- Sentiment Classification
- DNA Sequence Analysis
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition

Examples of Sequence Data

- Speech Recognition
- Music Generation
- Sentiment Classification
- DNA Sequence Analysis
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition

La



Examples of Sequence Data

- Speech Recognition
- Music Generation
- Sentiment Classification

  "Its an average movie" → 

- DNA Sequence Analysis
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition

Examples of Sequence Data

- Speech Recognition
- Music Generation
- Sentiment Classification
- DNA Sequence Analysis   AGCCCCTGTGAGGAACTAG → AGCCCCTGTGAGGAACTAG
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition

Examples of Sequence Data

- Speech Recognition
- Music Generation
- Sentiment Classification
- DNA Sequence Analysis
- Machine Translation — ARE YOU FEELING SLEEPY → क्या आपको नींद आ रही है
- Video Activity Recognition
- Name Entity Recognition

Examples of Sequence Data

- Speech Recognition
- Music Generation
- Sentiment Classification
- DNA Sequence Analysis
- Machine Translation
- Video Activity Recognition

WAVING

- Name Entity Recognition

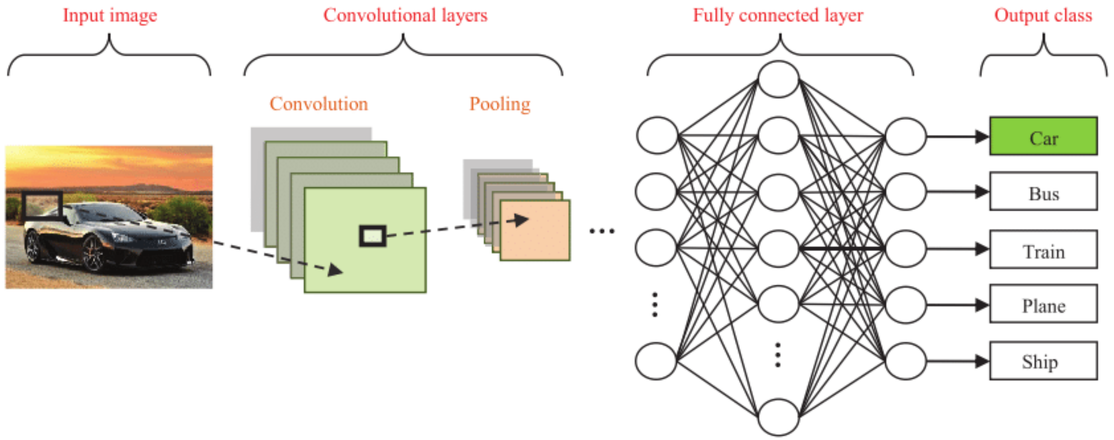
Examples of Sequence Data

- Speech Recognition
- Music Generation
- Sentiment Classification
- DNA Sequence Analysis
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition

"Alice wants to discuss about Deep Learning with Bob" → "Alice wants to discuss about Deep Learning with Bob"

Issues with using ANN/CNN on sequential data

- In feedforward and convolutional neural networks, the size of the input was always fixed.
    - In many applications with sequence data, the input is not of a fixed size.



Input 1: "Its an average movie"

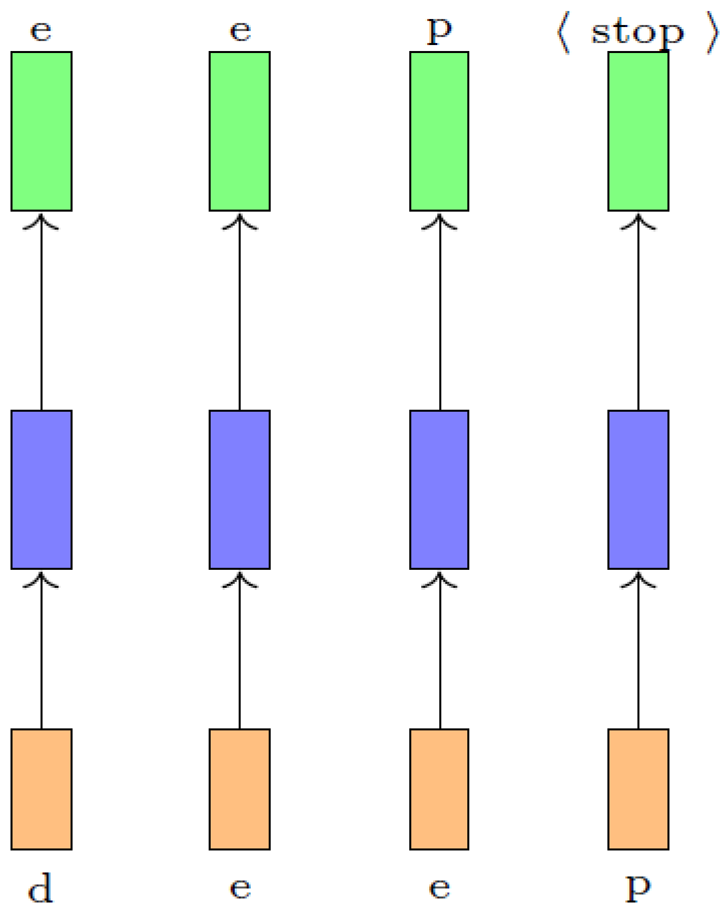
Input 2: " The direction was pathetic but the cinematography was fantastic"

Issues with using ANN/CNN on sequential data

- In feedforward and convolutional neural networks, the size of the input was always fixed.
  - In many applications with sequence data, the input is not of a fixed size.

- Further, each input to the ANN/CNN network was independent of the previous or future inputs.
  - With sequence data, successive inputs may not be independent of each other.

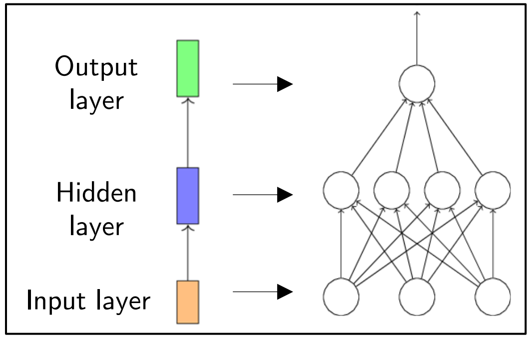
Modelling Sequence Learning Problems: Introduction



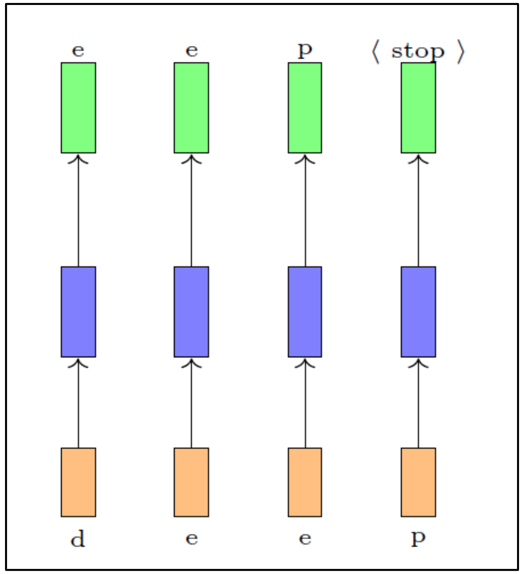
Task: Auto-complete

- Successive inputs are no longer dependent
- The length of the inputs and the no. of predictions you need to make is not fixed
- Each network is performing the same task

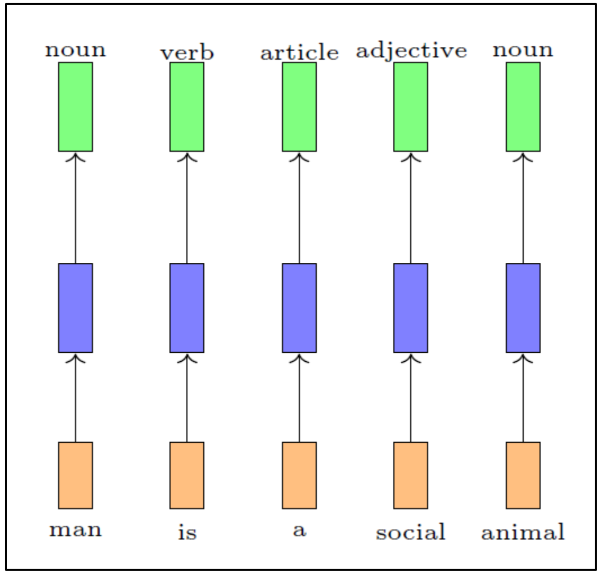
Legend



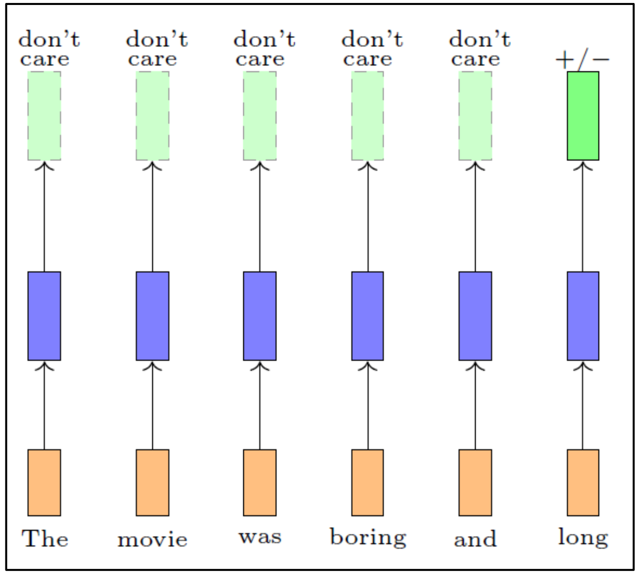
Modelling Sequence Learning Problems: Examples



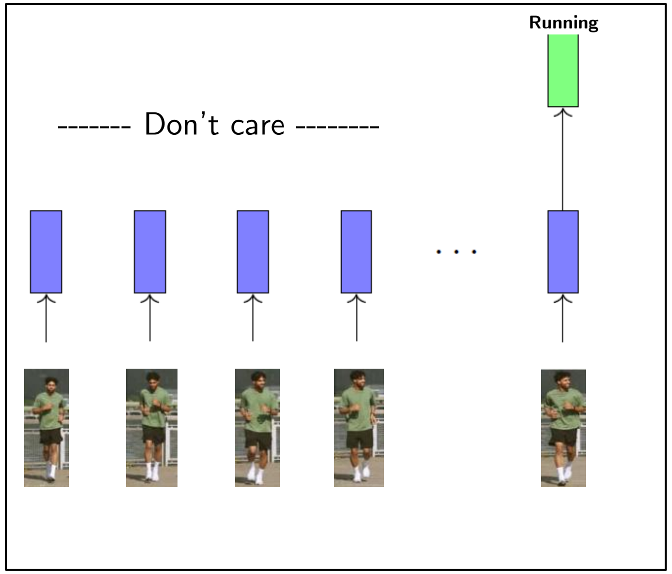
Task: Auto-complete



Task: P-o-S tagging

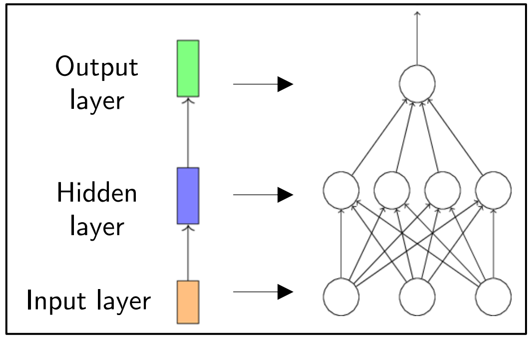


Task: Movie Review



Task: Action Recognition

Legend



- The model needs to look at a sequence of inputs and produce an output (or outputs).
- For this purpose, lets consider each input to be corresponding to one time step.
- Next, build a network for each time step/input, where each network performs the same task (eg: Auto complete: input=character, output=character)

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

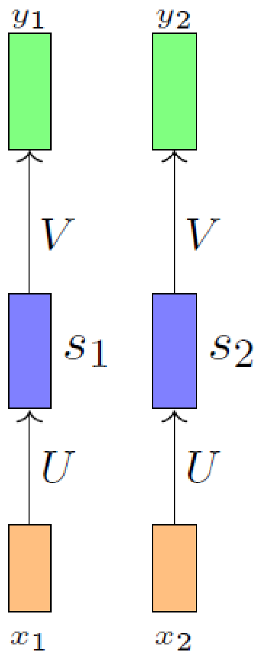
How to Model Sequence Learning Problems?

1. Model the dependence between inputs.
   - Eg: The next word after an 'adjective' is most probably a 'noun'.

2. Account for variable number of inputs.
   - A sentence can have arbitrary no. of words.
   - A video can have arbitrary no. of frames.

3. Make sure that the function executed at each time step is the same.
   - Because at each time step we are doing the same task.

Introduction

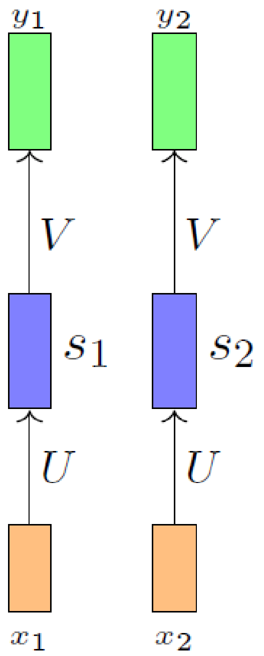


Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Considering the network at each time step to be a fully connected network, the general equation for the network at each time step is:

$$s_i = \sigma(Ux_i + b)$$
$$y_i = \mathcal{O}(Vs_i + c)$$
$$i = \text{timestep}$$

Introduction



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

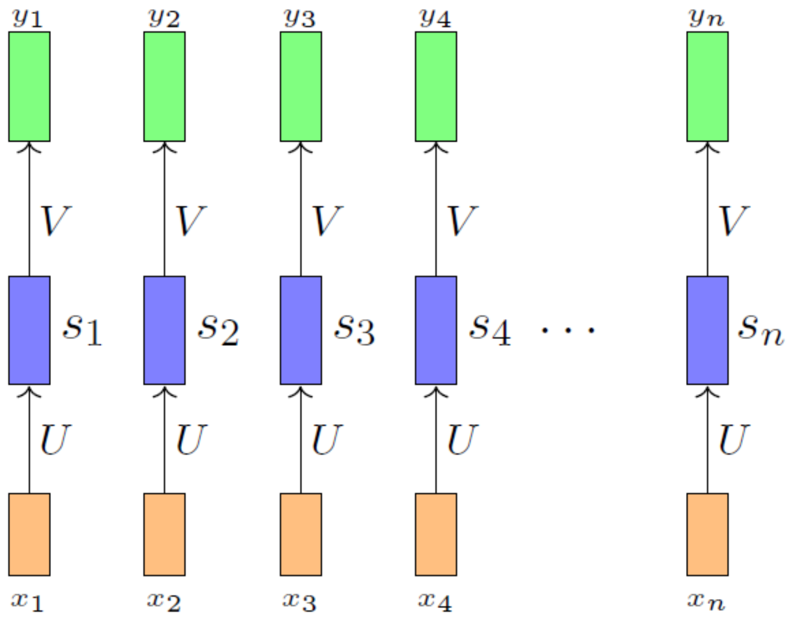
Considering the network at each time step to be a fully connected network, the general equation for the network at each time step is:

$$s_i = \sigma(Ux_i + b)$$
$$y_i = \mathcal{O}(Vs_i + c)$$
$$i = \text{timestep}$$

Since we want the same function to be executed at each timestep we should share the same network (i.e., same parameters at each timestep)

Recurrent Neural Networks (RNN): Introduction

- If the input sequence is of length 'n', we would create 'n' networks for each input, as seen previously.

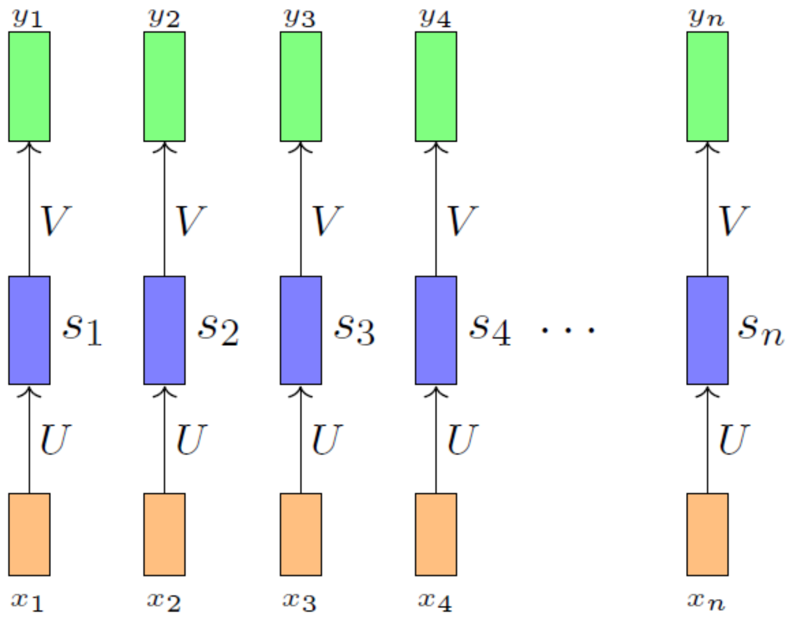


Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

By doing so, we have addressed the issue of variable input size!!

Recurrent Neural Networks (RNN): Introduction

- If the input sequence is of length 'n', we would create 'n' networks for each input, as seen previously.

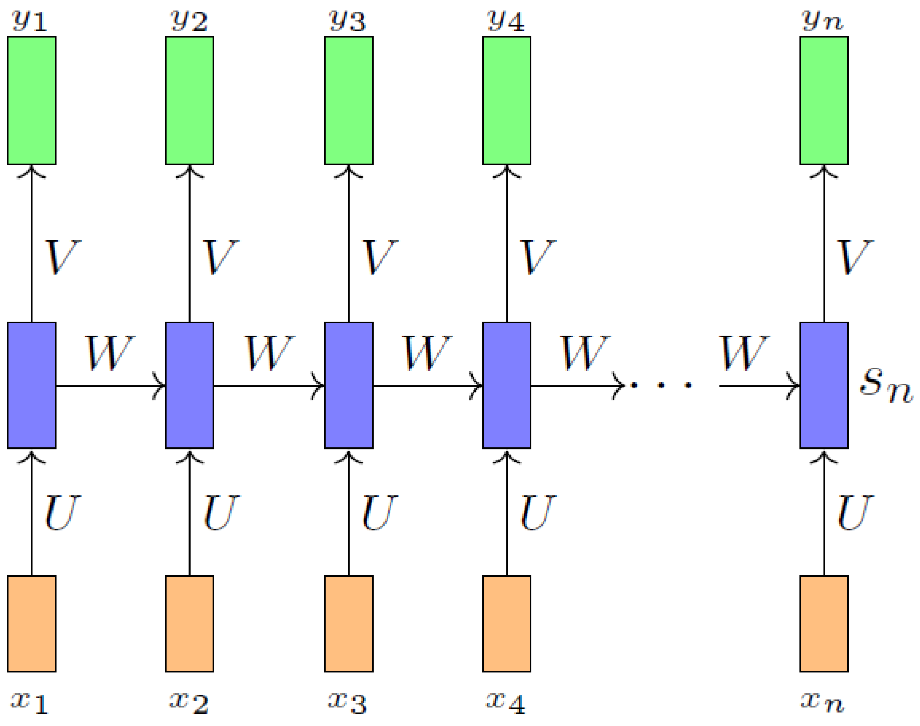


Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

But, how to model the dependencies between the inputs ?

Recurrent Neural Networks (RNN)

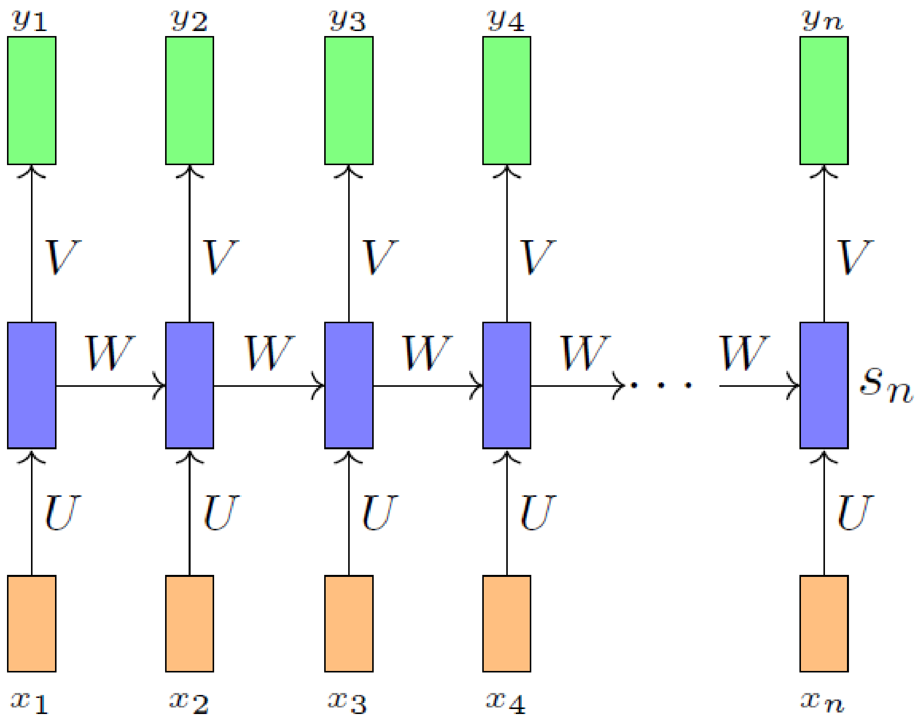
Solution: Add recurrent connection in the network.



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

Recurrent Neural Networks (RNN)

Solution: Add recurrent connection in the network.



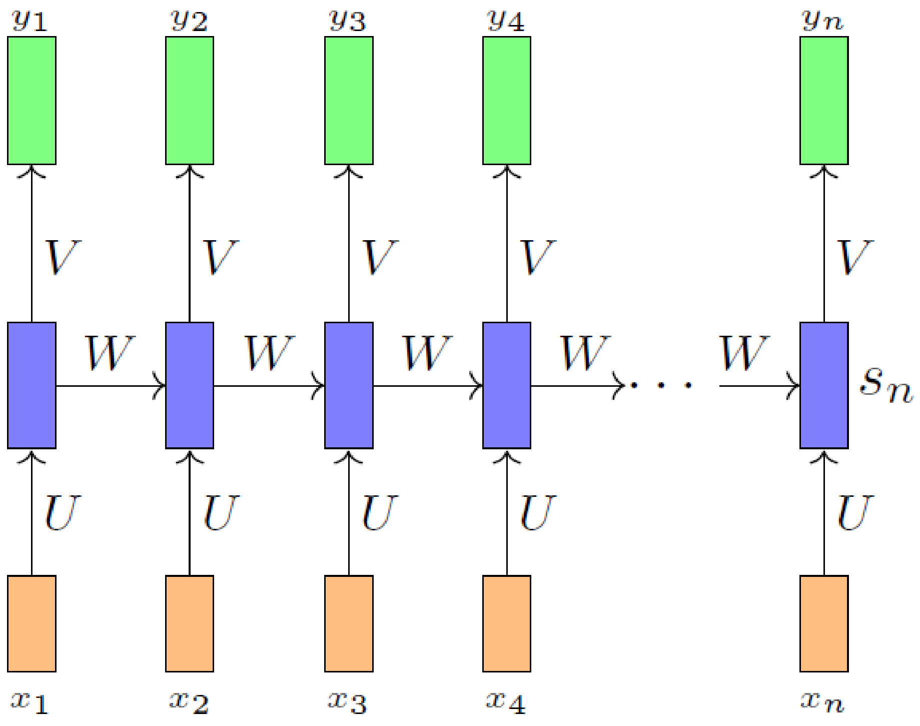
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- So, the RNN equation:

$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$y_i = \mathcal{O}(Vs_i + c)$$

Recurrent Neural Networks (RNN)

Solution: Add recurrent connection in the network.



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

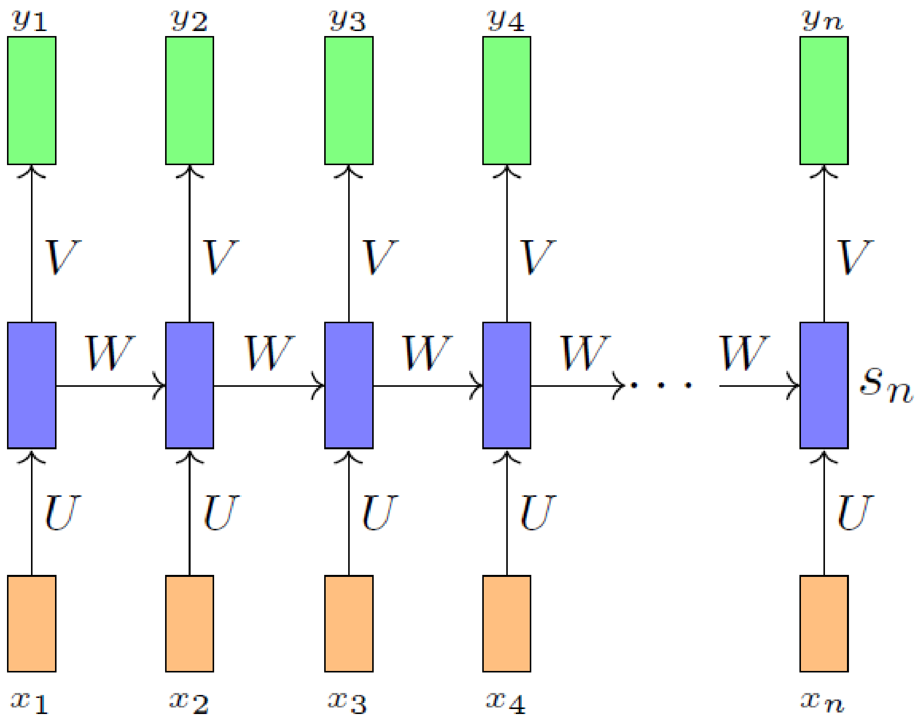
- So, the RNN equation:

$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$y_i = \mathcal{O}(Vs_i + c)$$

U, W, V, b, c are parameters of the network

Recurrent Neural Networks (RNN)

Solution: Add recurrent connection in the network.



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

see my notes for better understanding of architecture

- So, the RNN equation:

$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$y_i = \mathcal{O}(Vs_i + c)$$

U, W, V, b, c are parameters of the network

The dimensions of each term is as follows:

X_i -- [1 x no. of i/p neurons]

s_i -- [1 x no. of neurons in the hidden state]

W -- [no. of neurons in the hidden state x no. of neurons in the hidden state]

U -- [no. of i/p neurons x no. of neurons in the hidden state]

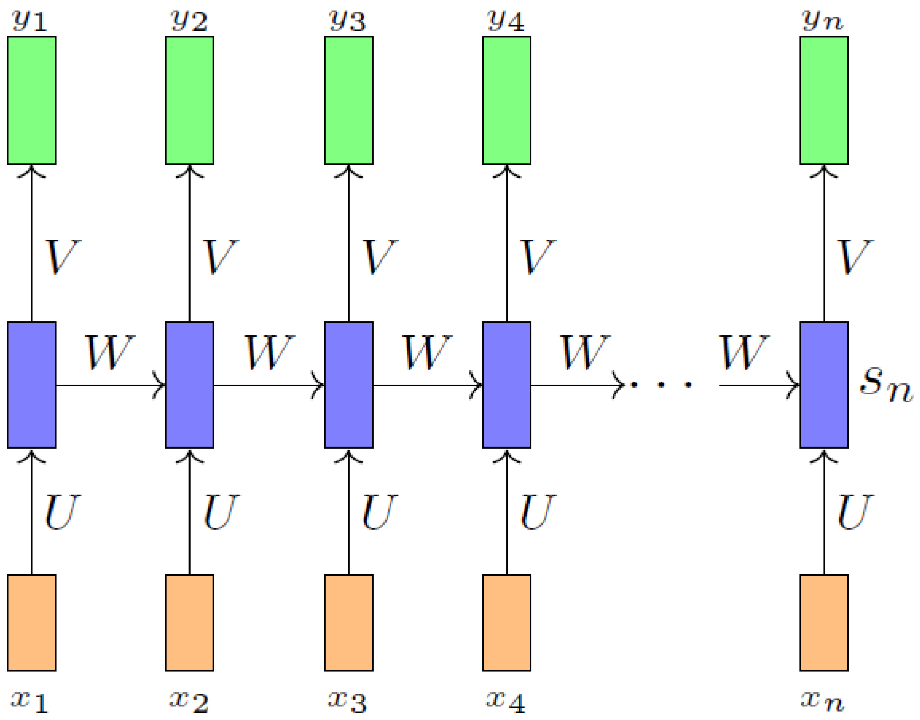
V -- [no. of neurons in the hidden state x no. of neurons in the o/p state]

b -- [1 x no. of neurons in the hidden state]

c – [1 x no. of neurons in the o/p state]

Recurrent Neural Networks (RNN)

Solution: Add recurrent connection in the network.



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

- So, the RNN equation:

$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$y_i = \mathcal{O}(Vs_i + c)$$

U, W, V, b, c are parameters of the network

The dimensions of each term is as follows:

X_i -- [1 x no. of i/p neurons]

s_i -- [1 x no. of neurons in the hidden state]

W -- [no. of neurons in the hidden state x no. of neurons in the hidden state]

U -- [no. of i/p neurons x no. of neurons in the hidden state]

V -- [no. of neurons in the hidden state x no. of neurons in the o/p state]

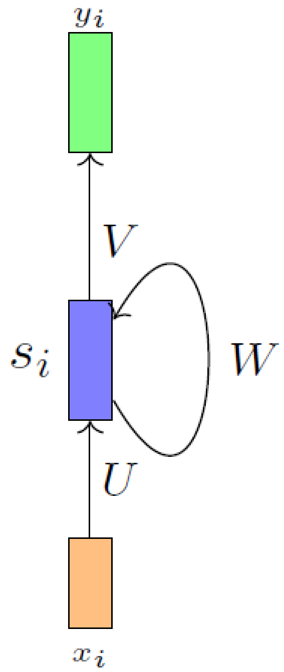
b -- [1 x no. of neurons in the hidden state]

c – [1 x no. of neurons in the o/p state]

- At time step i=0 there are no previous inputs, so they are typically assumed to be all zeros.
- Since, the output of s_i at time step i is a function of all the inputs from previous time steps, we could say it has a form of **memory.**
- A part of a neural network that preserves some state across time steps is called a **memory cell** (or simply a **cell**)

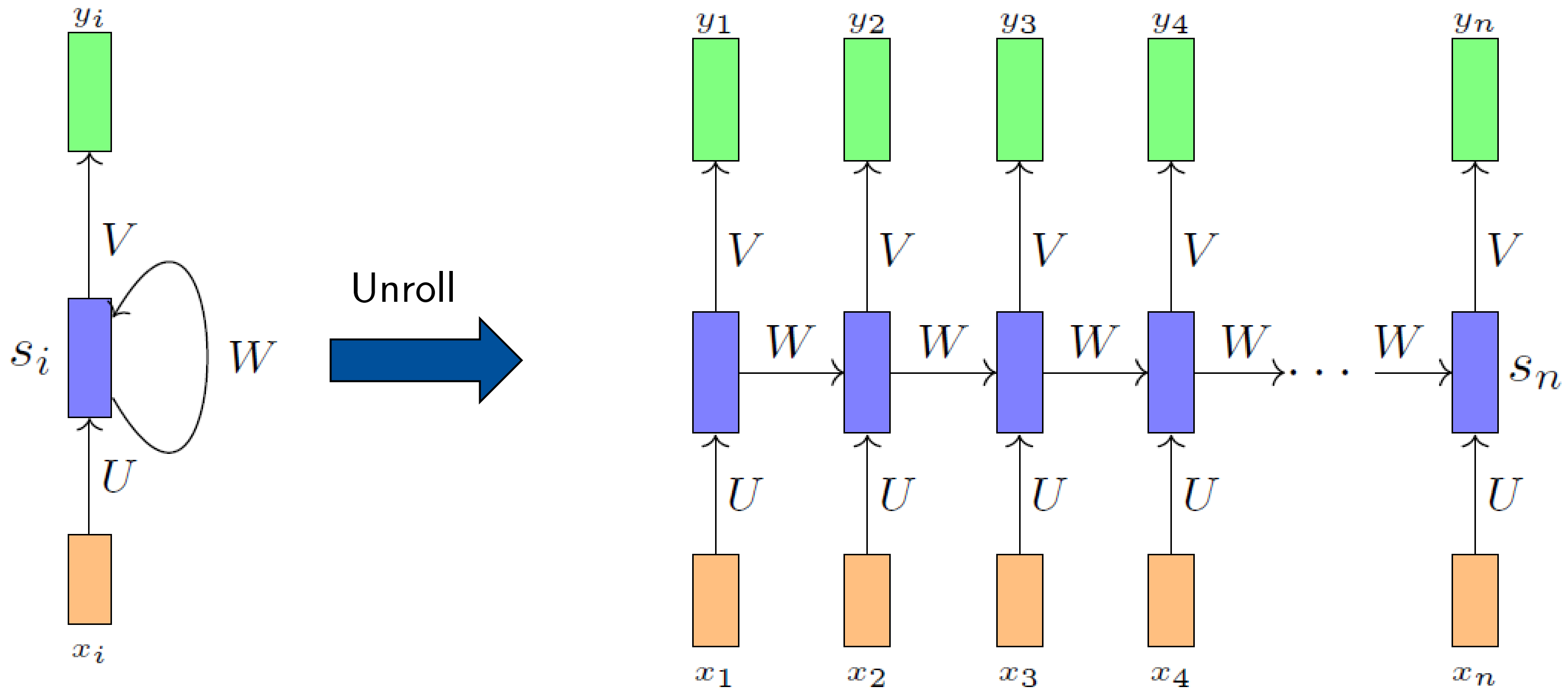
Recurrent Neural Networks (RNN)

Compact representation of a RNN:



Recurrent Neural Networks (RNN)

best architecture to understand this link go to end and see basically how(my prompt for better): https://chatgpt.com/c/66e7989d-0a38-8001-adfd-4a04a8fa74dd



Same representation as seen previously

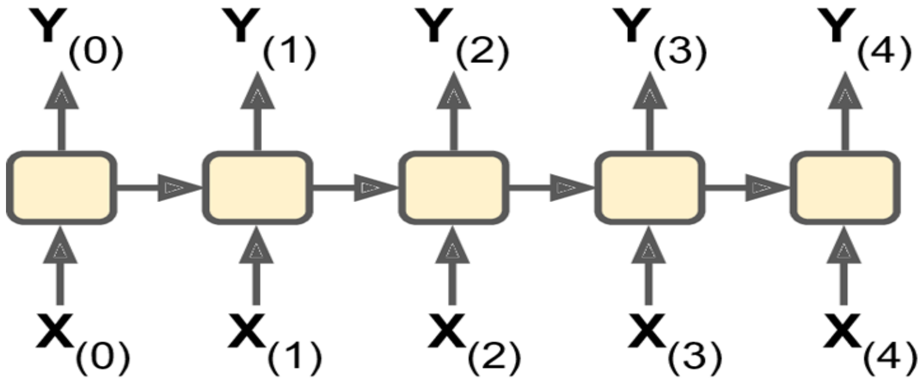
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

not yi-1 it shd have been hidden state ka output si-1(mostly shd check also)

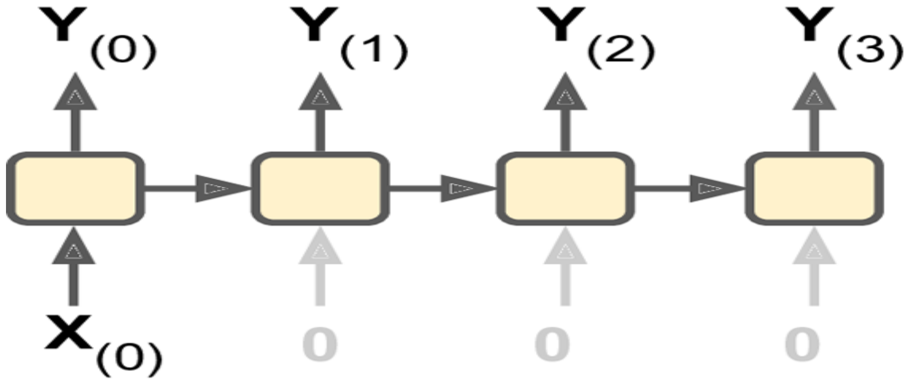
- Unrolling the network through time = representing network against time axis.
- At each time step t (also called a frame) RNN receives inputs x_i as well as output from previous step y_{i-1}

Input and Output Sequences

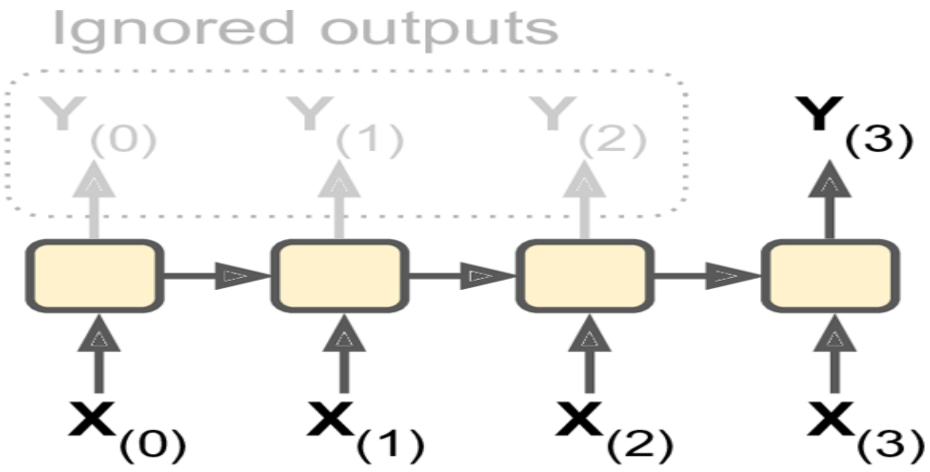
Seq-to-Seq

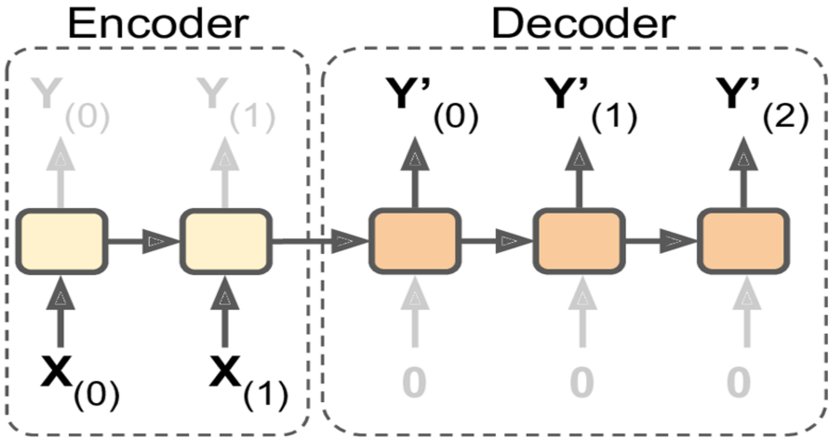


Vector-to-Seq

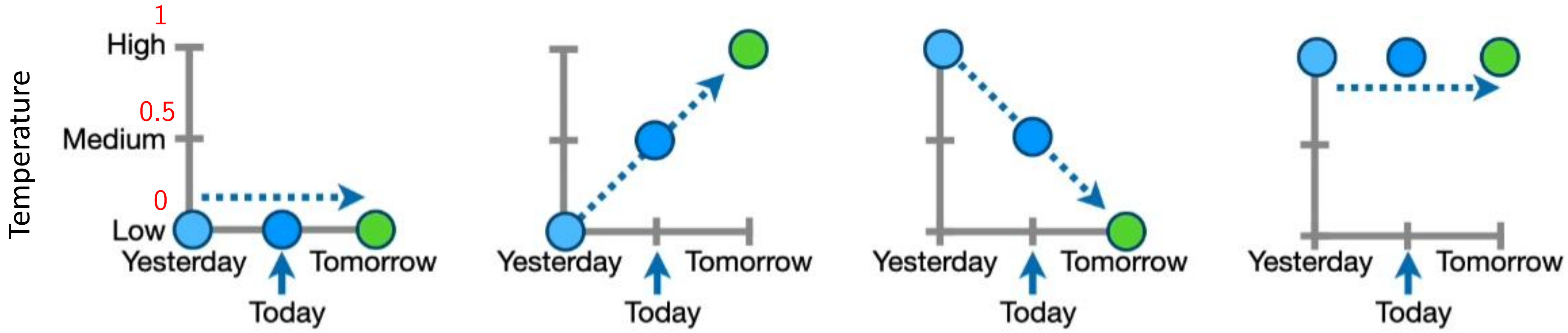


Seq-to-Vector



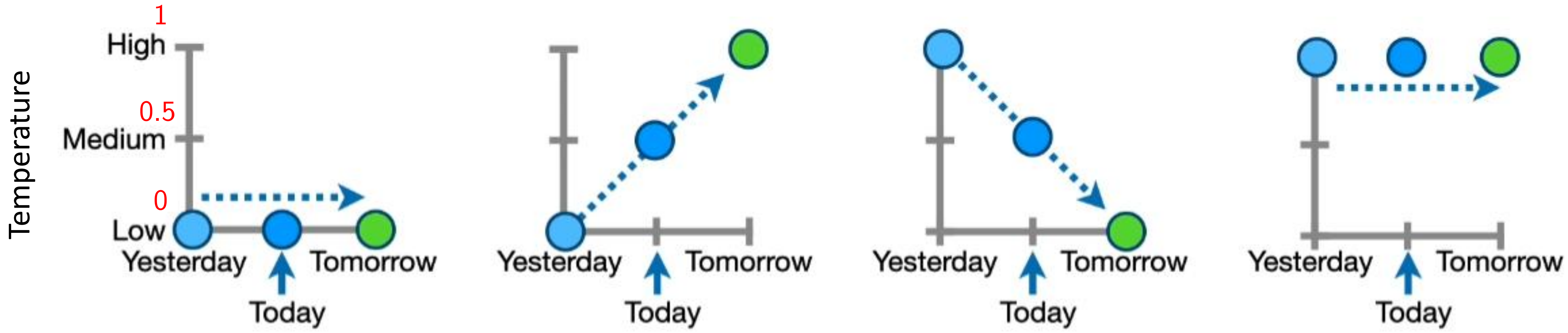


Recurrent Neural Networks (RNN) : Example



Source: https://www.youtube.com/c/joshstarmer

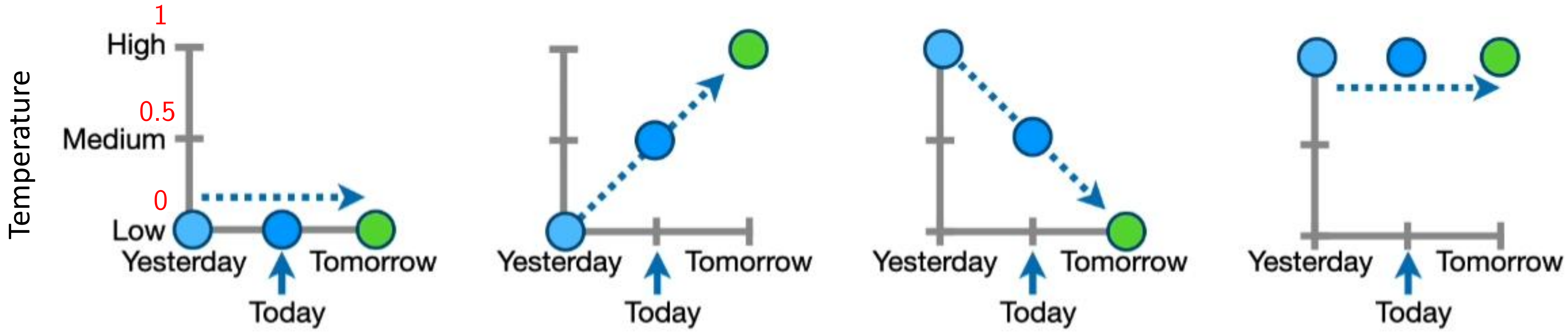
Recurrent Neural Networks (RNN) : Example

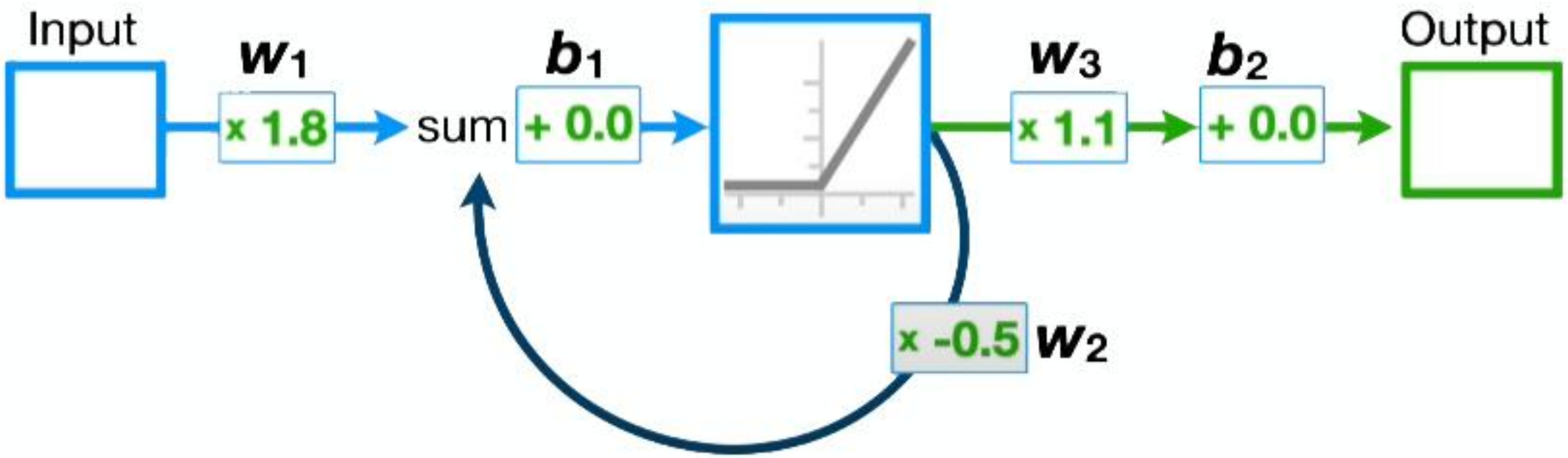


Problem :Given the temperatures of yesterday and today predict tomorrow's temperature.

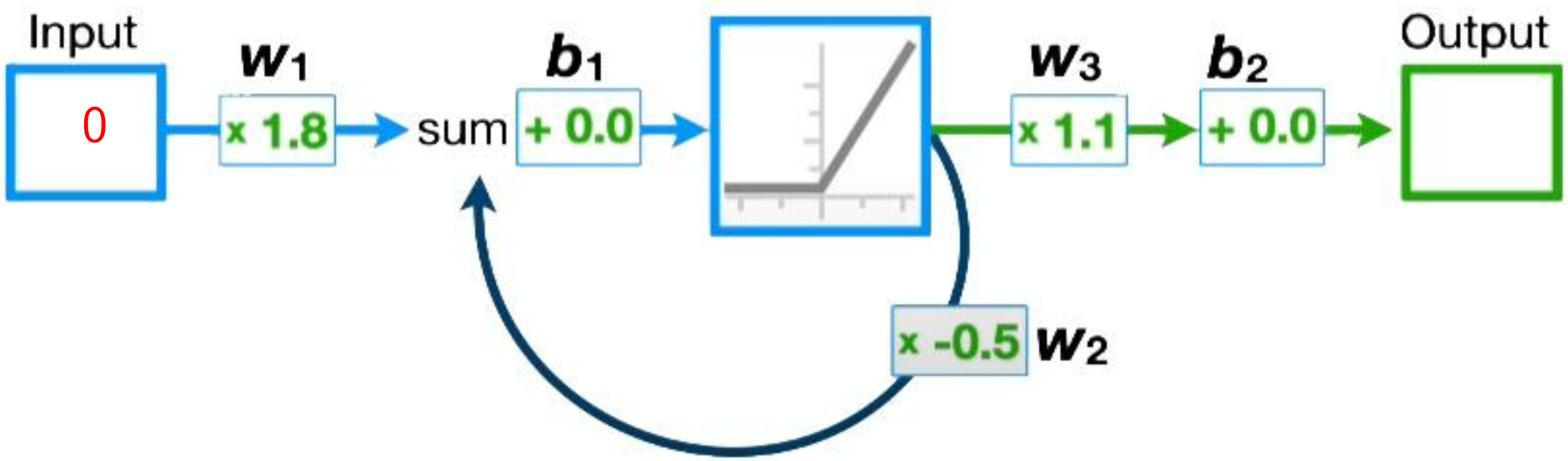
Source: https://www.youtube.com/c/joshstarmer

Recurrent Neural Networks (RNN) : Example



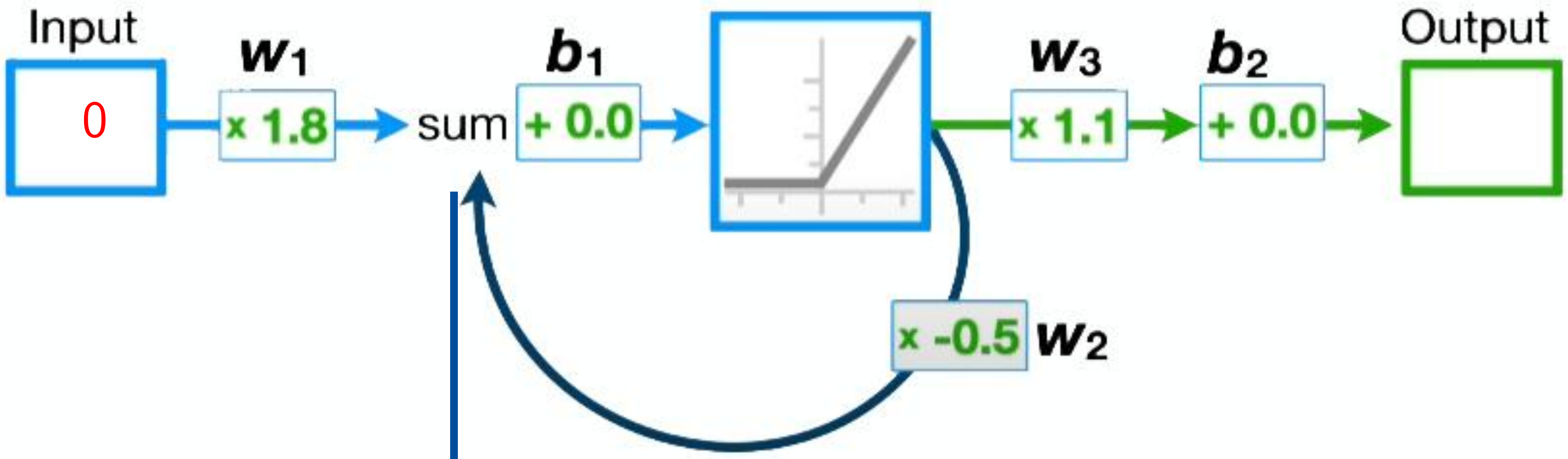


Source: https://www.youtube.com/c/joshstarmer



Source: https://www.youtube.com/c/joshstarmer

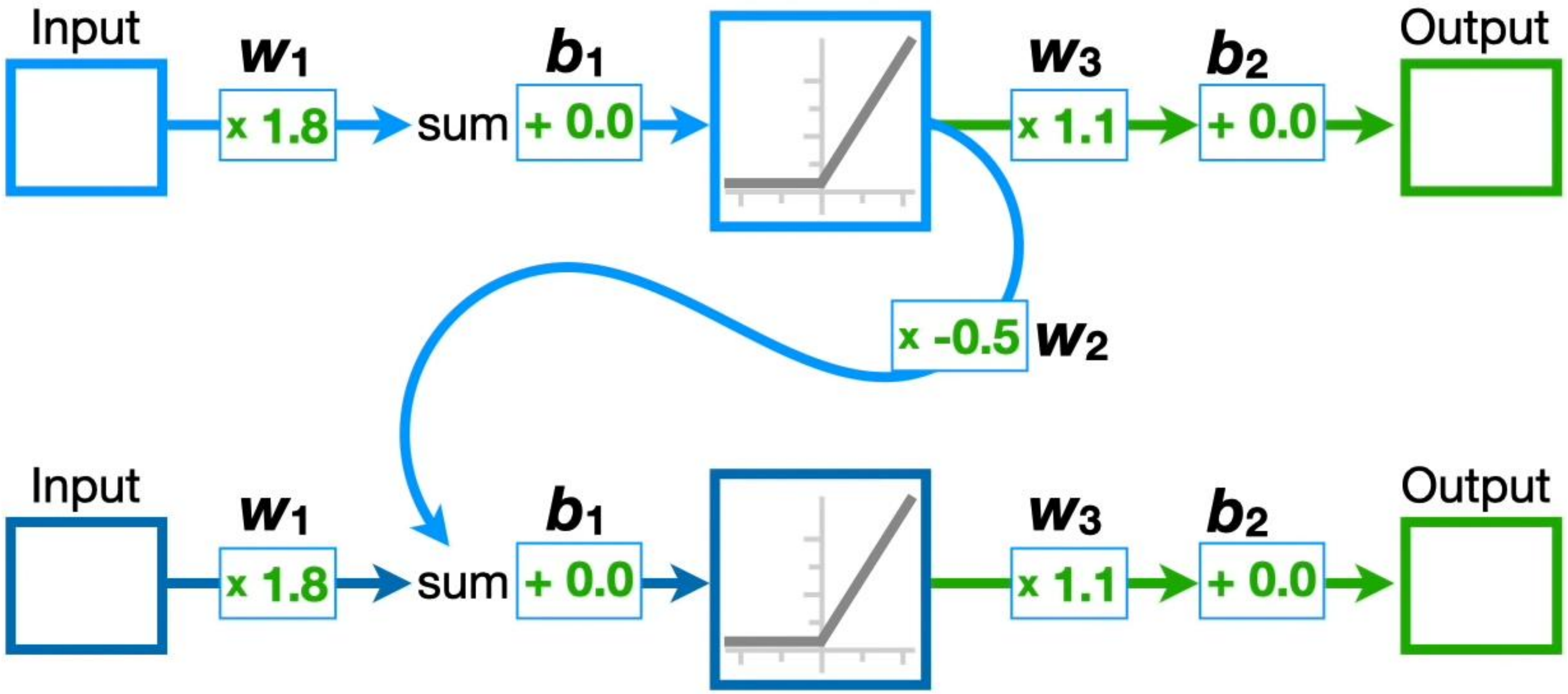
Recurrent Neural Networks (RNN) : Example



$$(\textbf{Today} \times w_1) + (y_1 \times w_2)$$

Source: https://www.youtube.com/c/joshstarmer

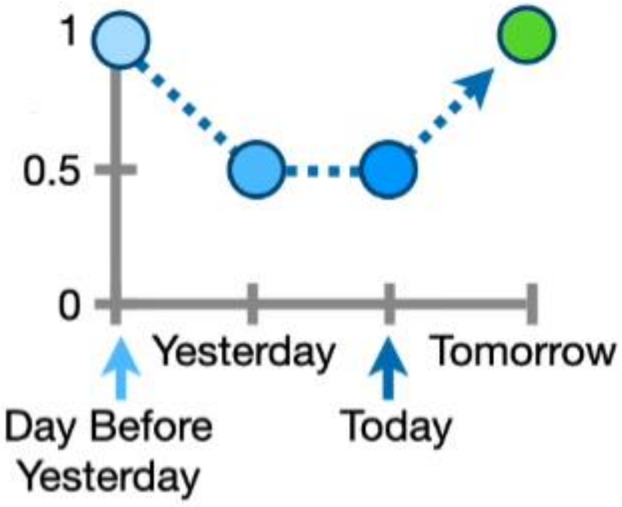
Recurrent Neural Networks (RNN) : Example



Unrolling the feedback loop by making a copy of NN for each input value

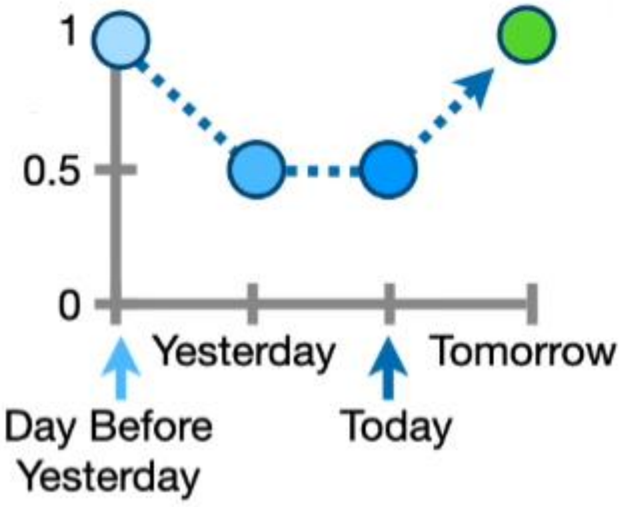
Source: https://www.youtube.com/c/joshstarmer

Recurrent Neural Networks (RNN) : Example



Source: https://www.youtube.com/c/joshstarmer

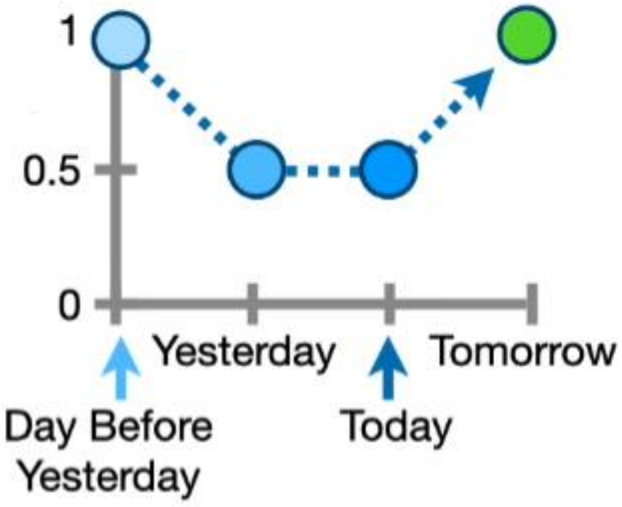
Recurrent Neural Networks (RNN) : Example

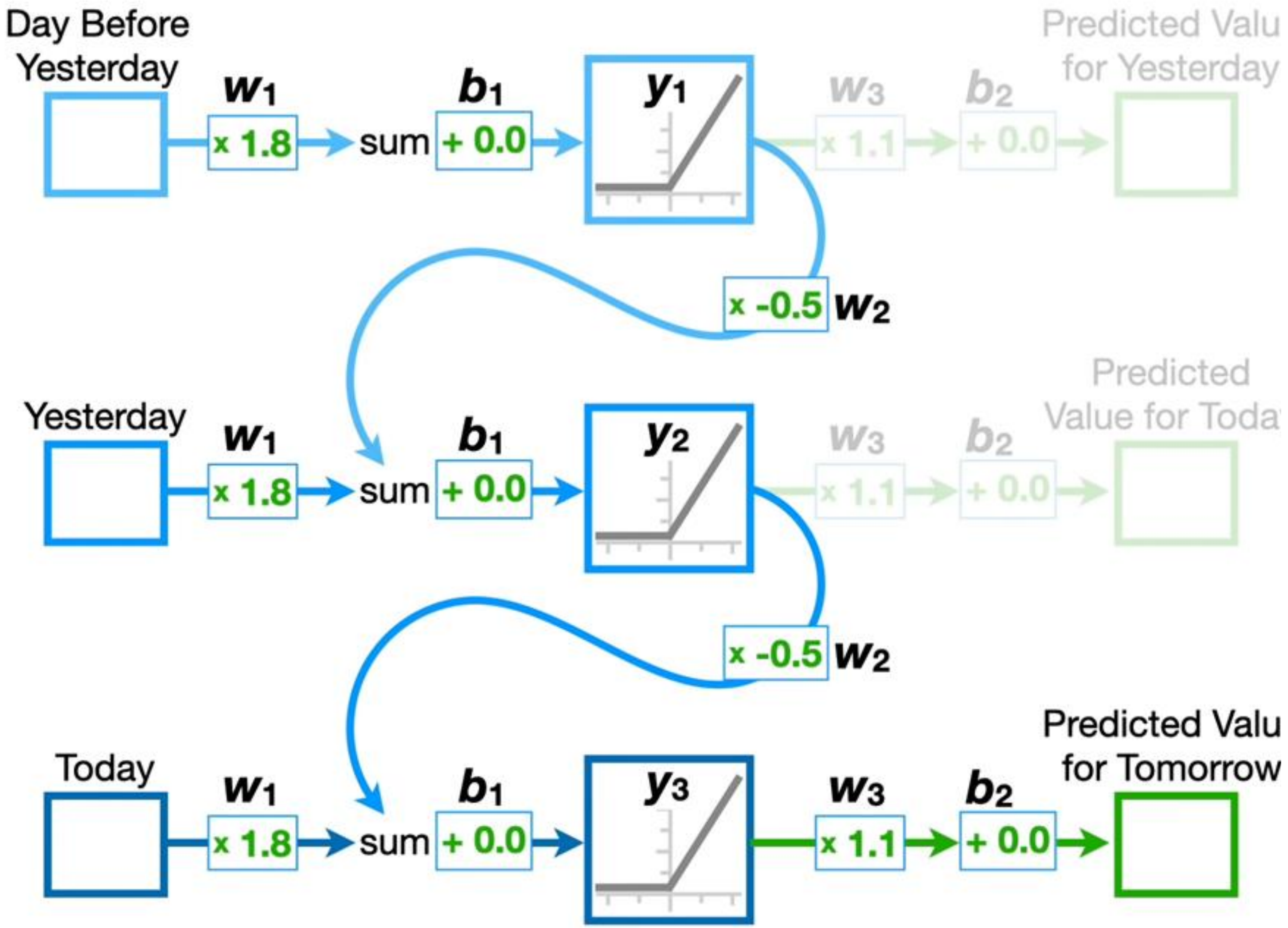


Problem : Given the temperature of 3 days (today, yesterday and day before yesterday), Predict tomorrow's temperature?

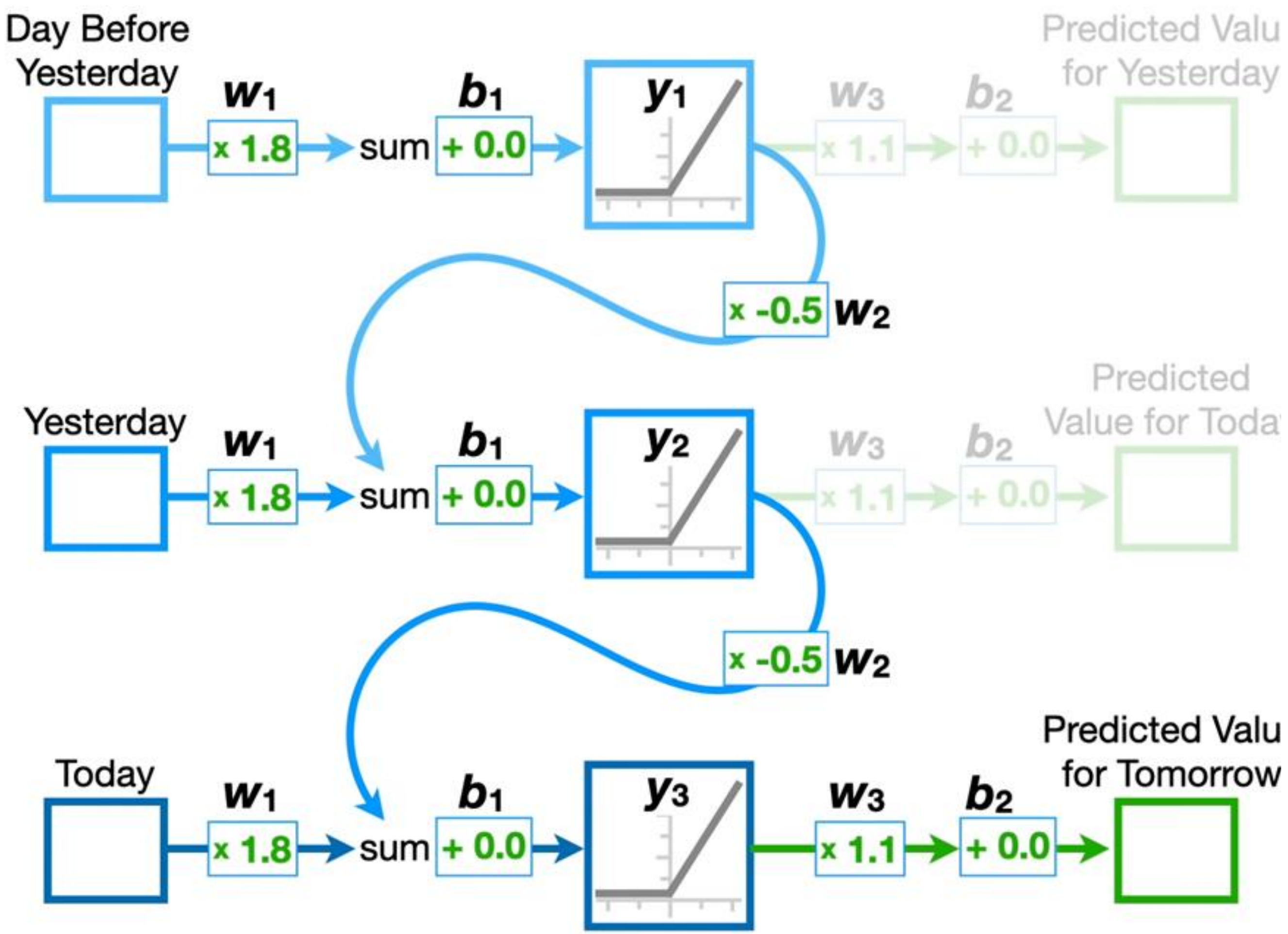
Source: https://www.youtube.com/c/joshstarmer

Recurrent Neural Networks (RNN) : Example





Source: https://www.youtube.com/c/joshstarmer



So, the no. of networks = no. of inputs

Source: https://www.youtube.com/c/joshstarmer

Recurrent Neural Networks (RNN) : Example

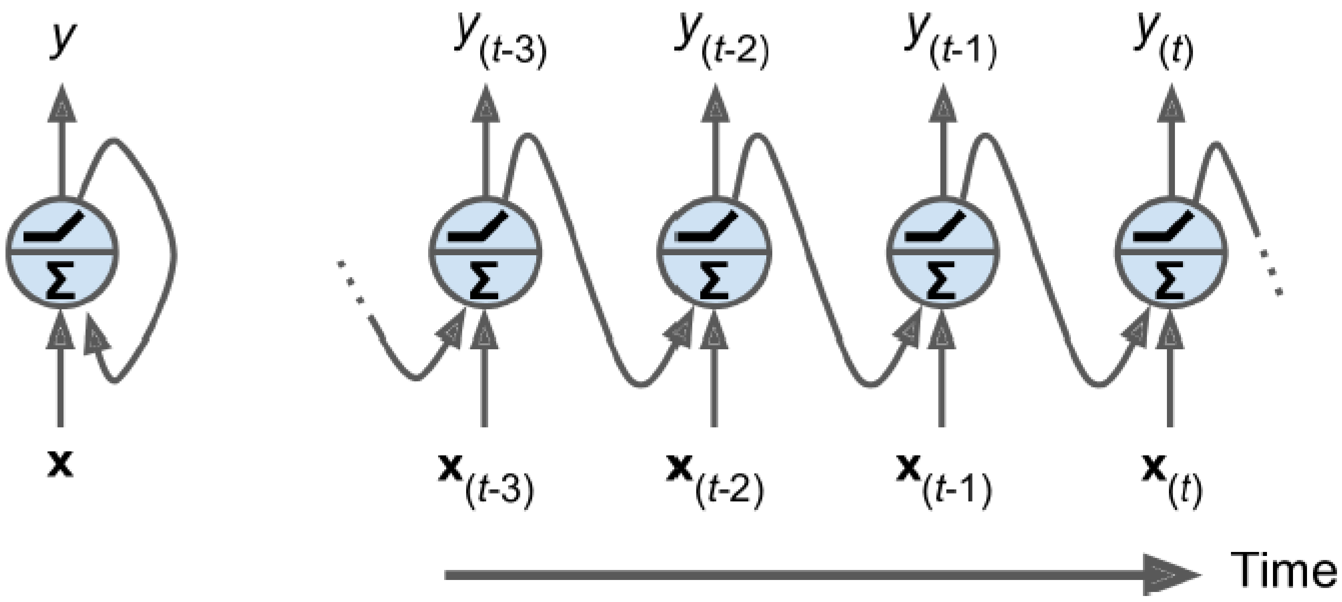


Figure 15-1. A recurrent neuron (left) unrolled through time (right)

```
model = keras.models.Sequential([
  keras.layers.SimpleRNN(1, input_shape=[None, 1])
])
```

Recurrent Neural Networks (RNN) : Example

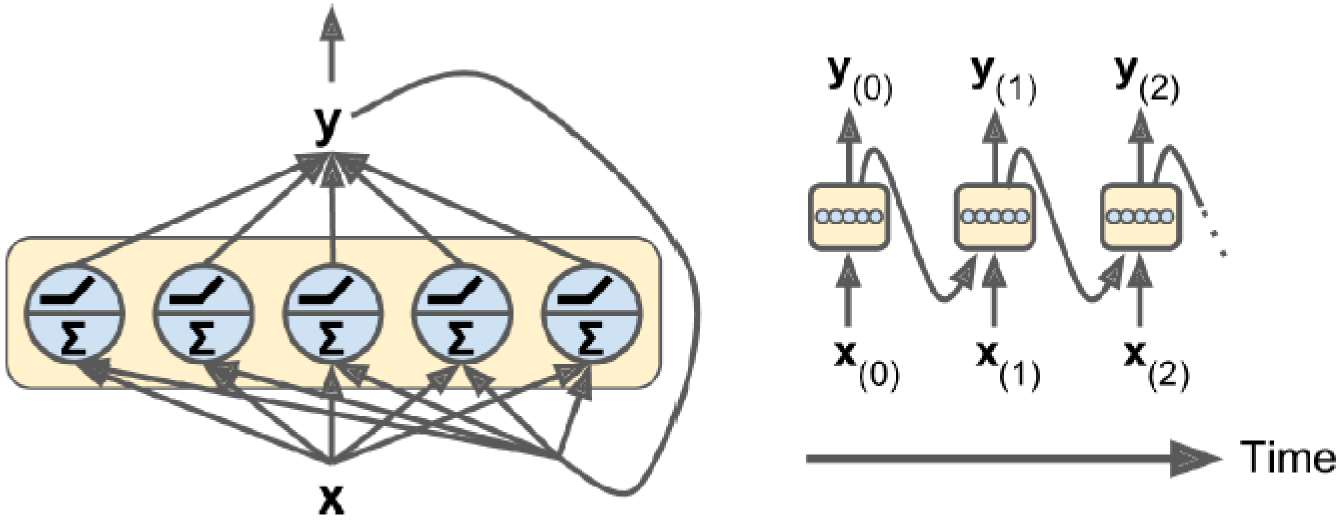


Figure 15-2. A layer of recurrent neurons (left) unrolled through time (right)

```
model = keras.models.Sequential([
  keras.layers.SimpleRNN(5, input_shape=[None, 1])
])
```

Recurrent Neural Networks (RNN) : Example

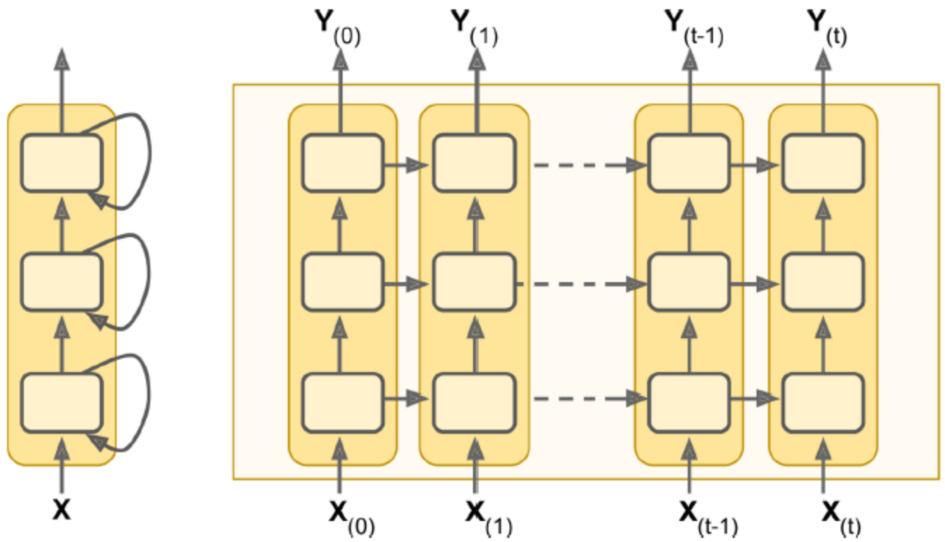


Figure 15-7. Deep RNN (left) unrolled through time (right)

```
model = keras.models.Sequential([
    keras.layers.SimpleRNN(20, return_sequences=True, input_shape=[None,1]),

    keras.layers.SimpleRNN(20, return_sequences=True),
    keras.layers.SimpleRNN(1)
])
```

Recurrent Neural Networks (RNN) : Example

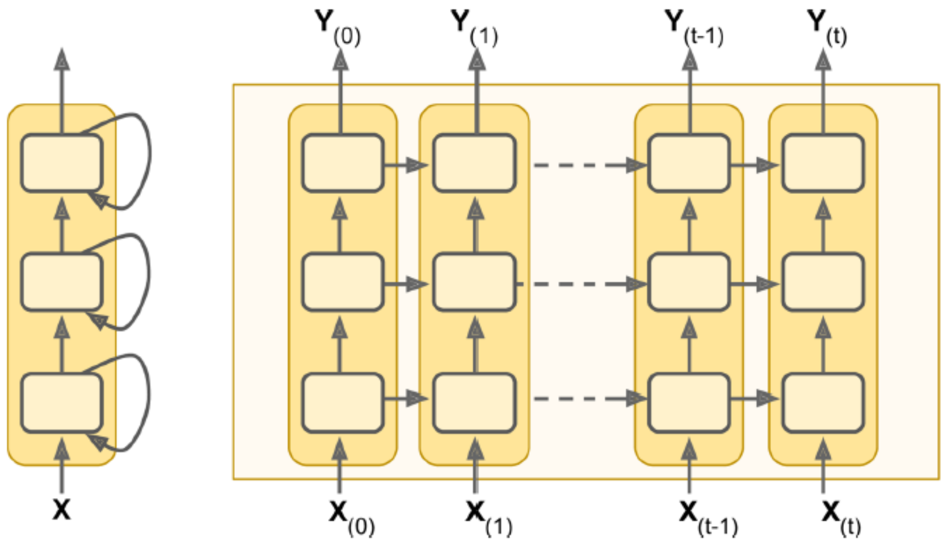
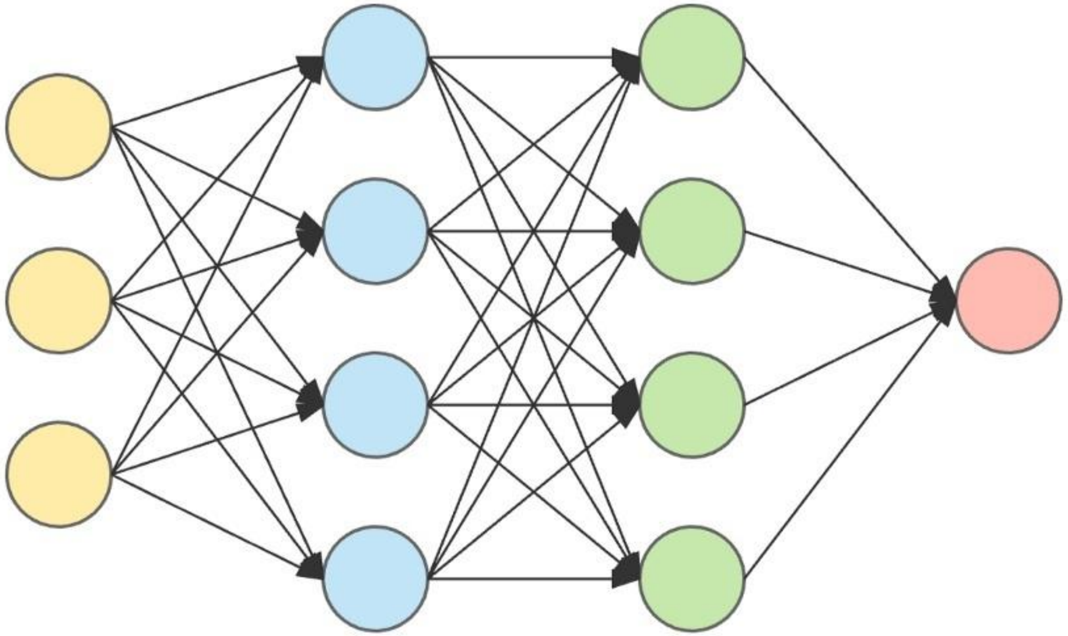


Figure 15-7. Deep RNN (left) unrolled through time (right)

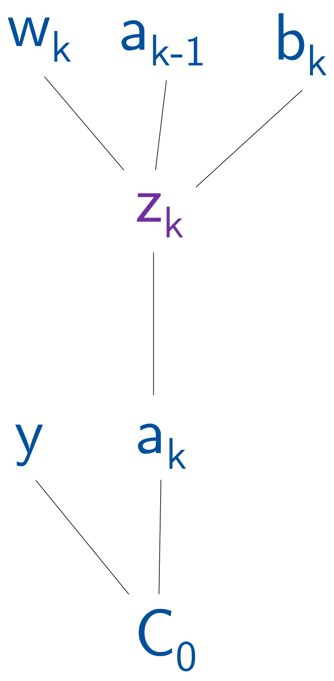
```
model = keras.models.Sequential([
    keras.layers.SimpleRNN(20, return_sequences=True, input_shape=[None, 1]),

    keras.layers.SimpleRNN(20),
    keras.layers.Dense(1)
])
```

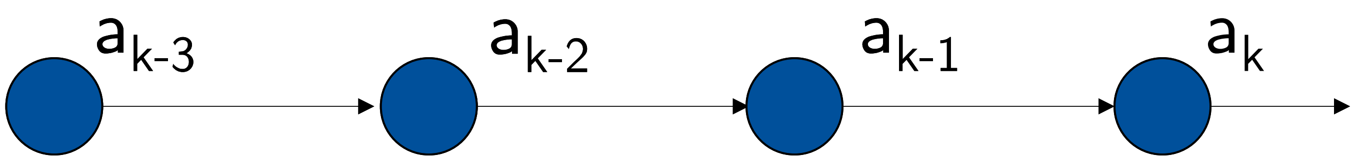
Backpropagation in ANN : Recap



$$\frac{\partial C_0}{\partial w_k} = \frac{\partial z_k}{\partial w_k}\frac{\partial a_k}{\partial z_k}\frac{\partial C_0}{\partial a_k} = a_{k-1}\,\sigma'(z_k) * 2*(a_k - y)$$



Dependency graph



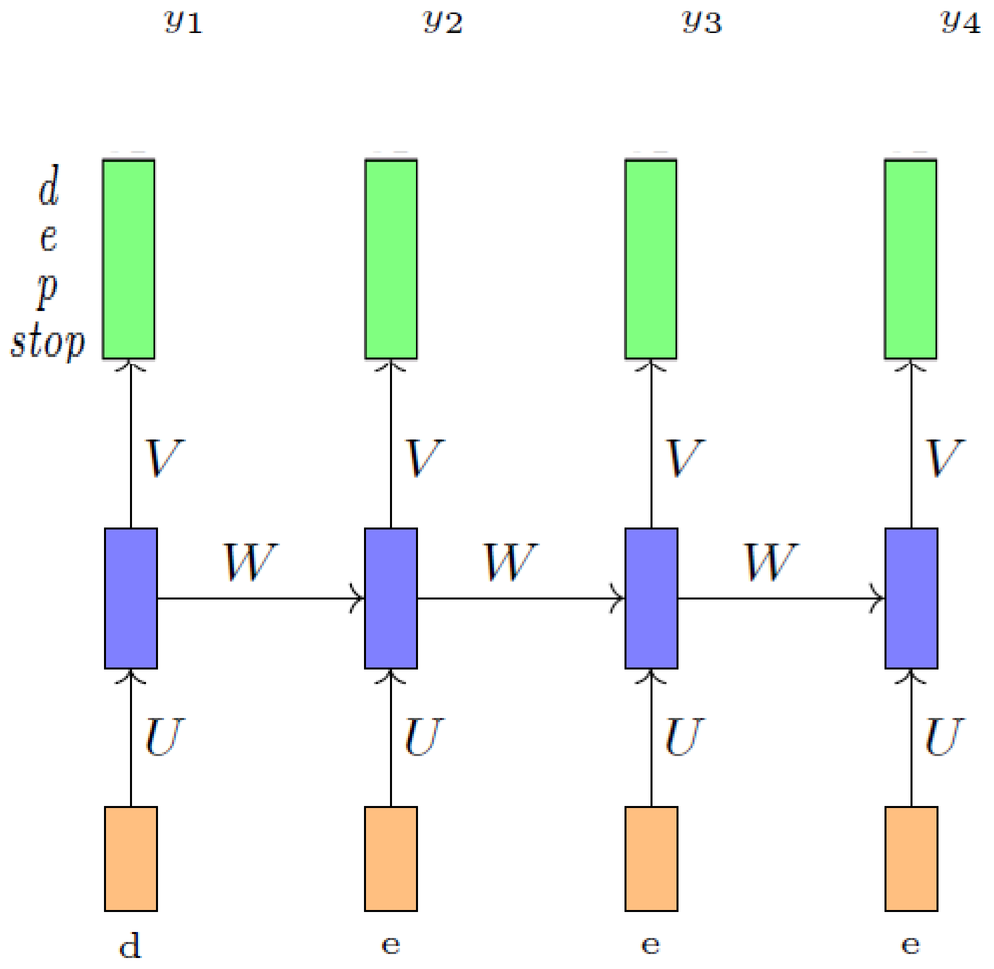
$$C_0 = (a_k - y)^2$$

$$a_k = \sigma(w_k\, a_{k-1} + b_k)$$

Now, if $z_k = w_k\, a_{k-1} + b_k$

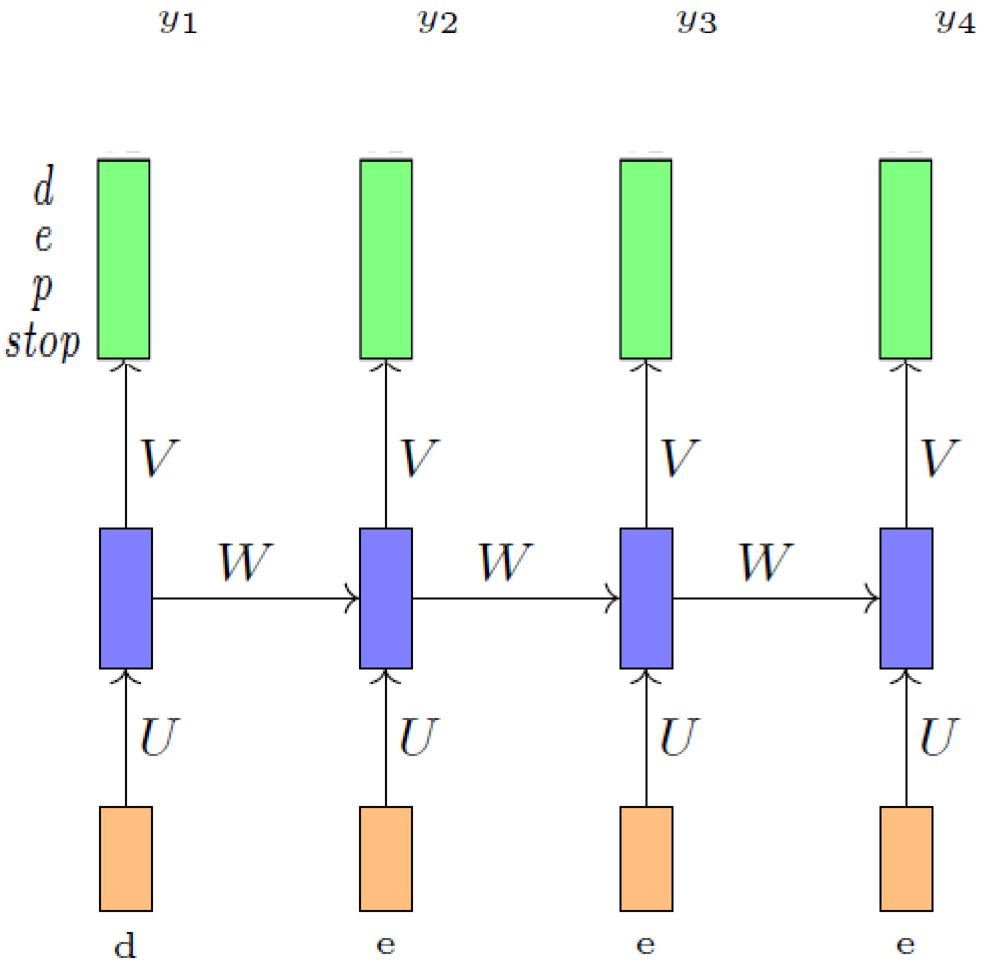
Then, $a_k = \sigma(z_k)$

Training RNNs : Backpropagation through time (BPTT)



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

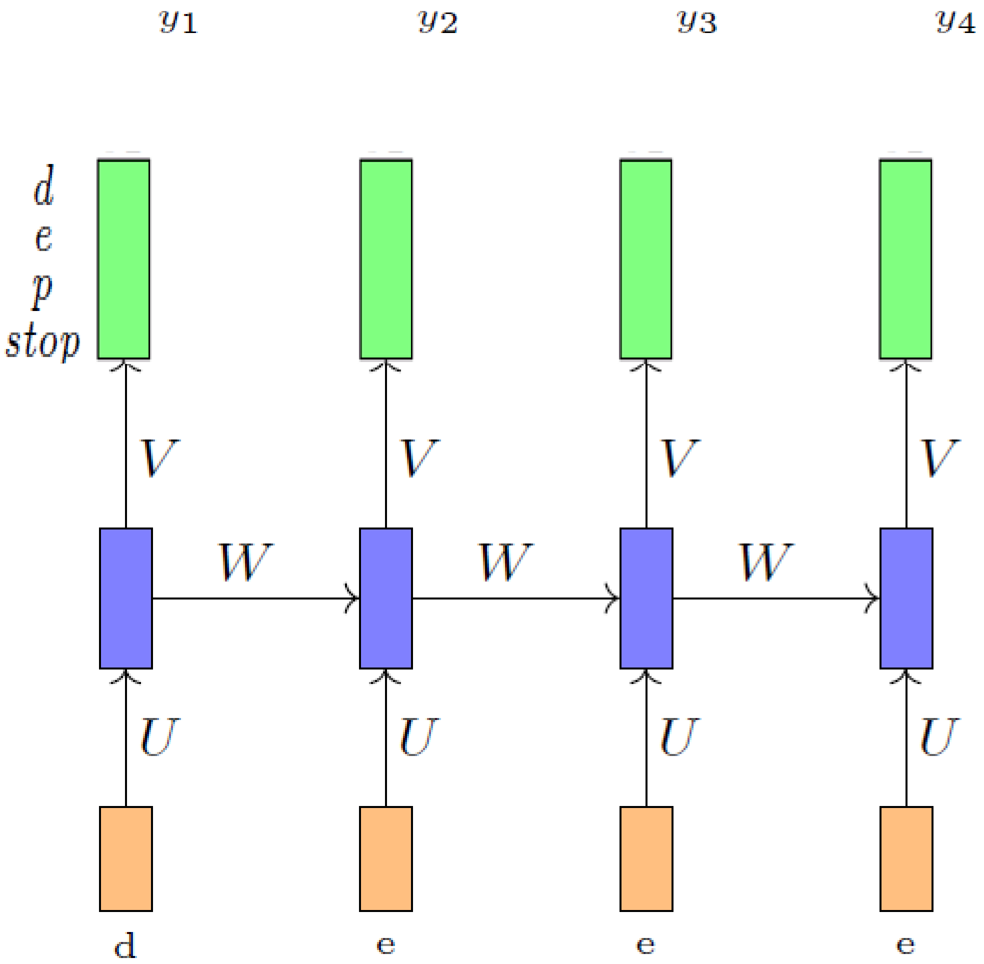
Training RNNs : Backpropagation through time (BPTT)



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

- For instance, consider the task of auto-completion (predicting the next character).

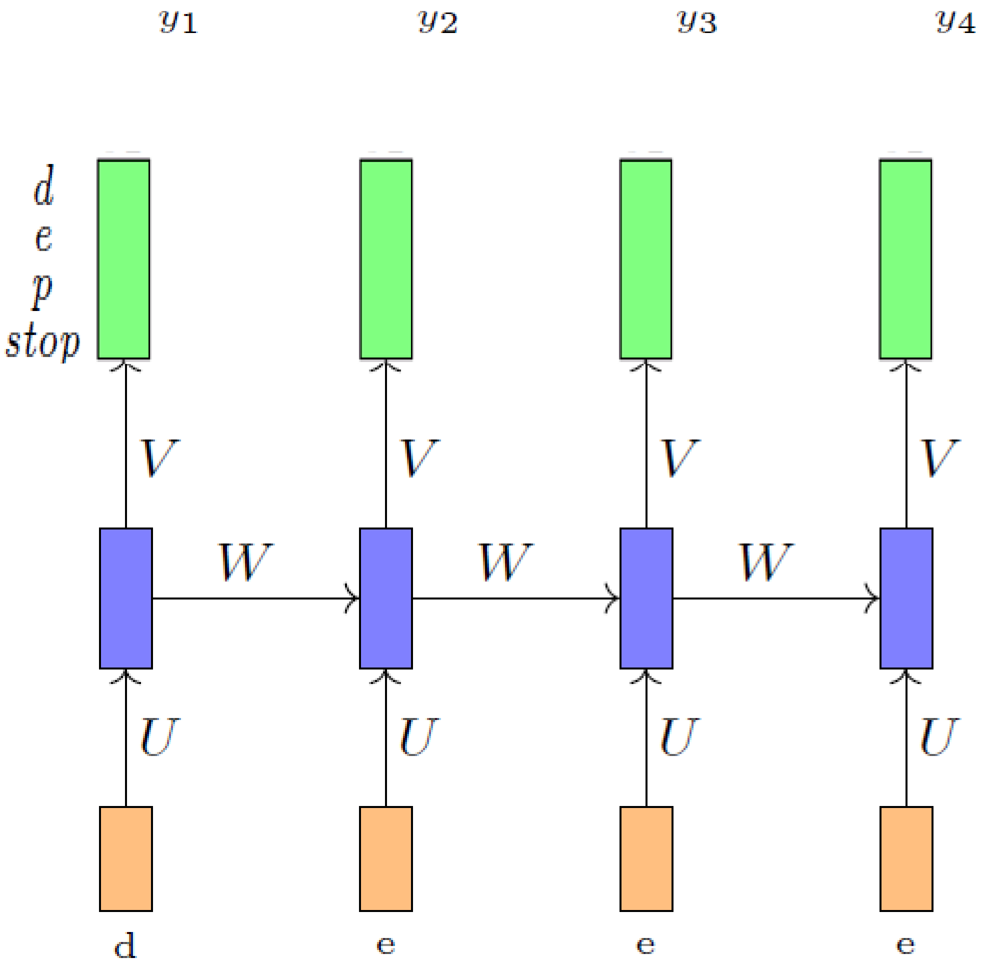
Training RNNs : Backpropagation through time (BPTT)



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

- For instance, consider the task of auto-completion (predicting the next character).
- For simplicity we assume that there are only 4 characters in our vocabulary (d, e, p, <stop>).

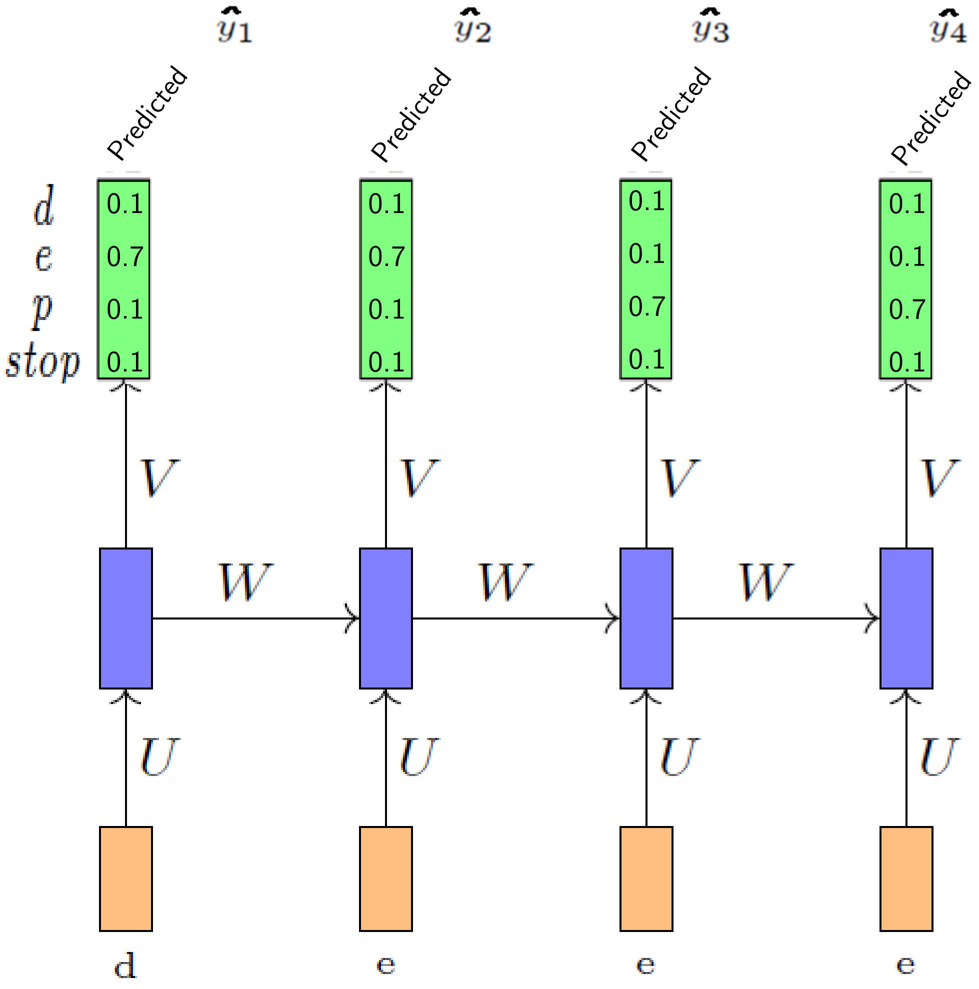
Training RNNs : Backpropagation through time (BPTT)



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

- For instance, consider the task of auto-completion (predicting the next character).
- For simplicity we assume that there are only 4 characters in our vocabulary (d, e, p, <stop>).
- At each timestep we want to predict one of these 4 characters.

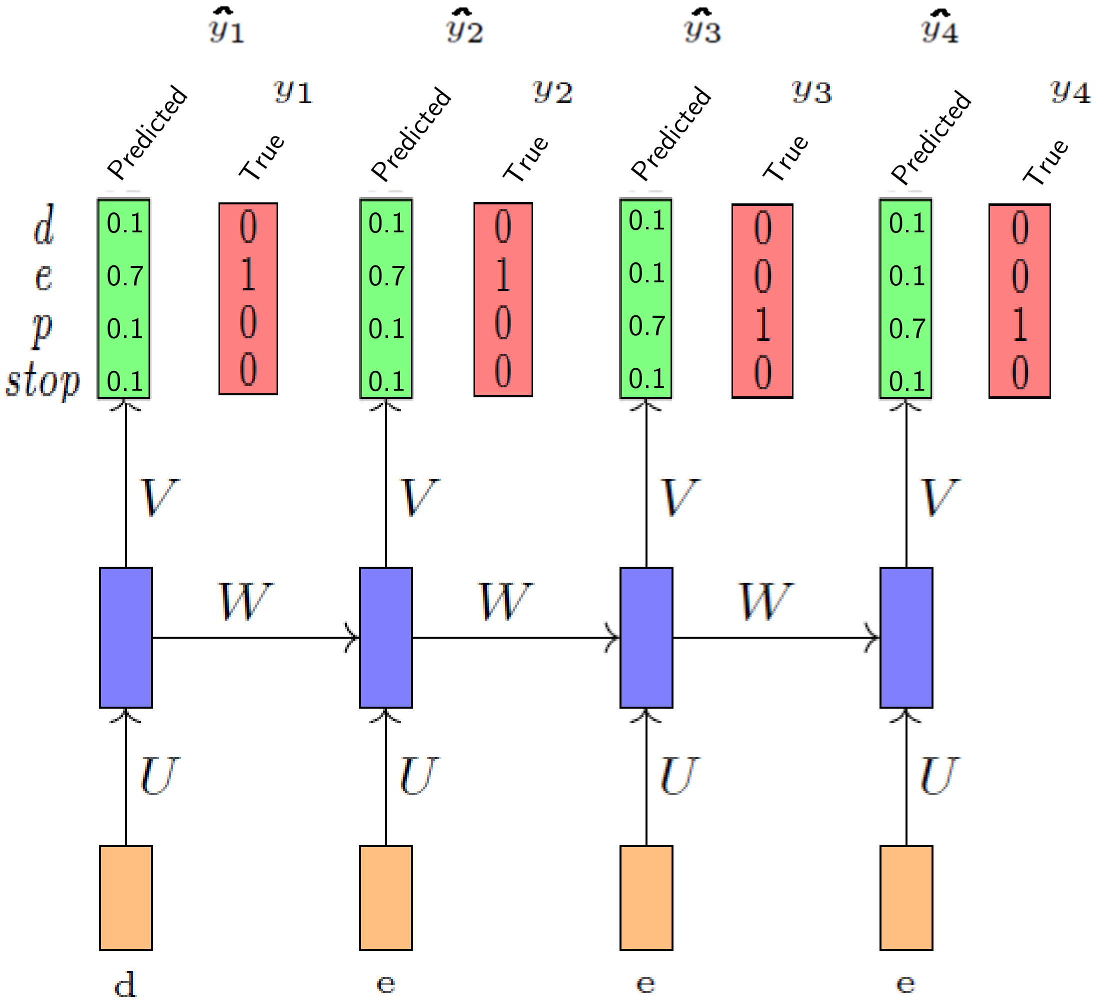
Training RNNs : Backpropagation through time (BPTT)



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

- For instance, consider the task of auto-completion (predicting the next character).
- For simplicity we assume that there are only 4 characters in our vocabulary (d, e, p, <stop>).
- At each timestep we want to predict one of these 4 characters.
- Suppose we initialize U, V, W randomly and the network predicts the probabilities (green block)

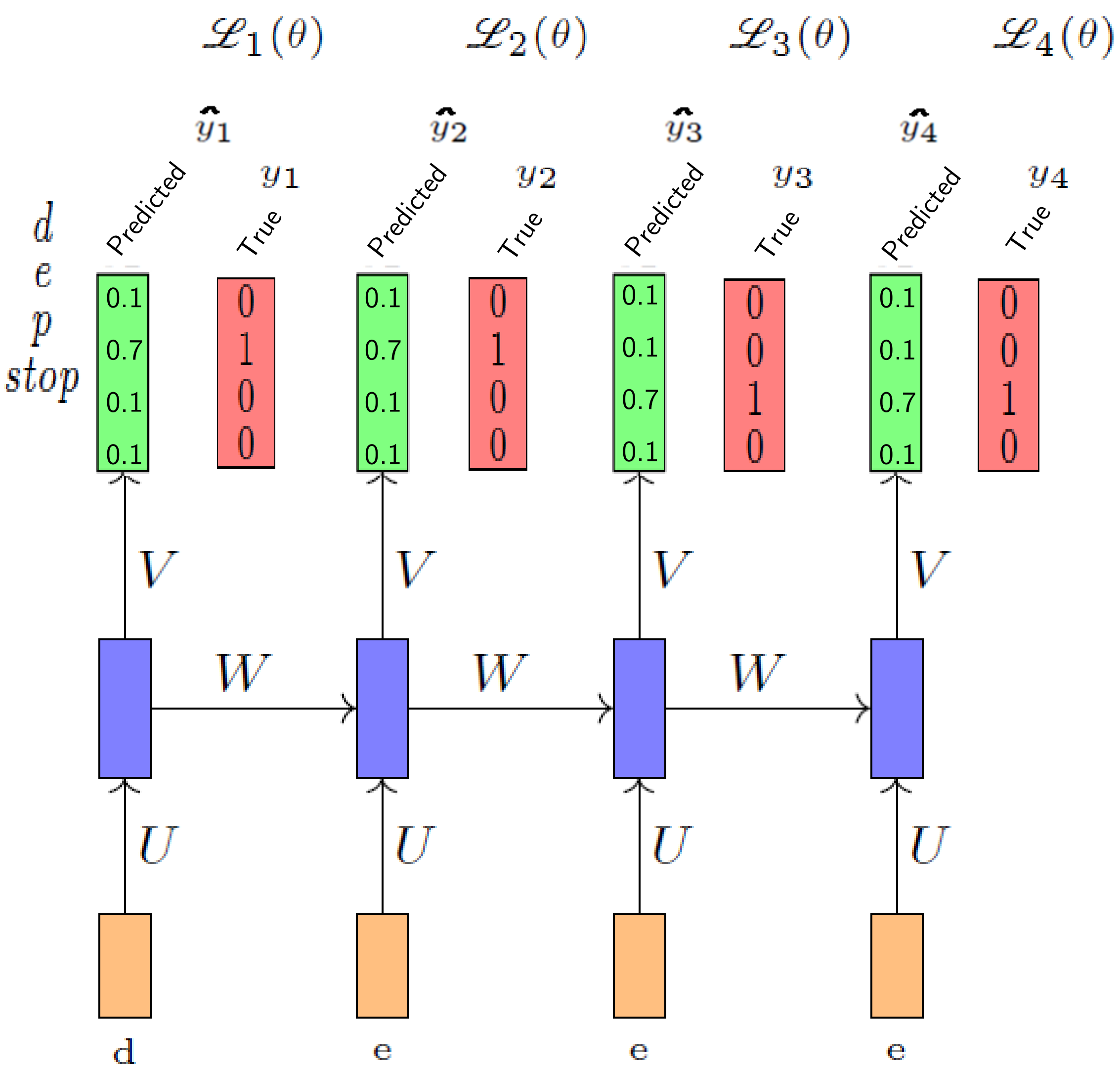
Training RNNs : Backpropagation through time (BPTT)



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

- For instance, consider the task of auto-completion (predicting the next character).
- For simplicity we assume that there are only 4 characters in our vocabulary (d, e, p, <stop>).
- At each timestep we want to predict one of these 4 characters.
- Suppose we initialize U, V, W randomly and the network predicts the probabilities (green block)
- And the true probabilities are as shown (red block).

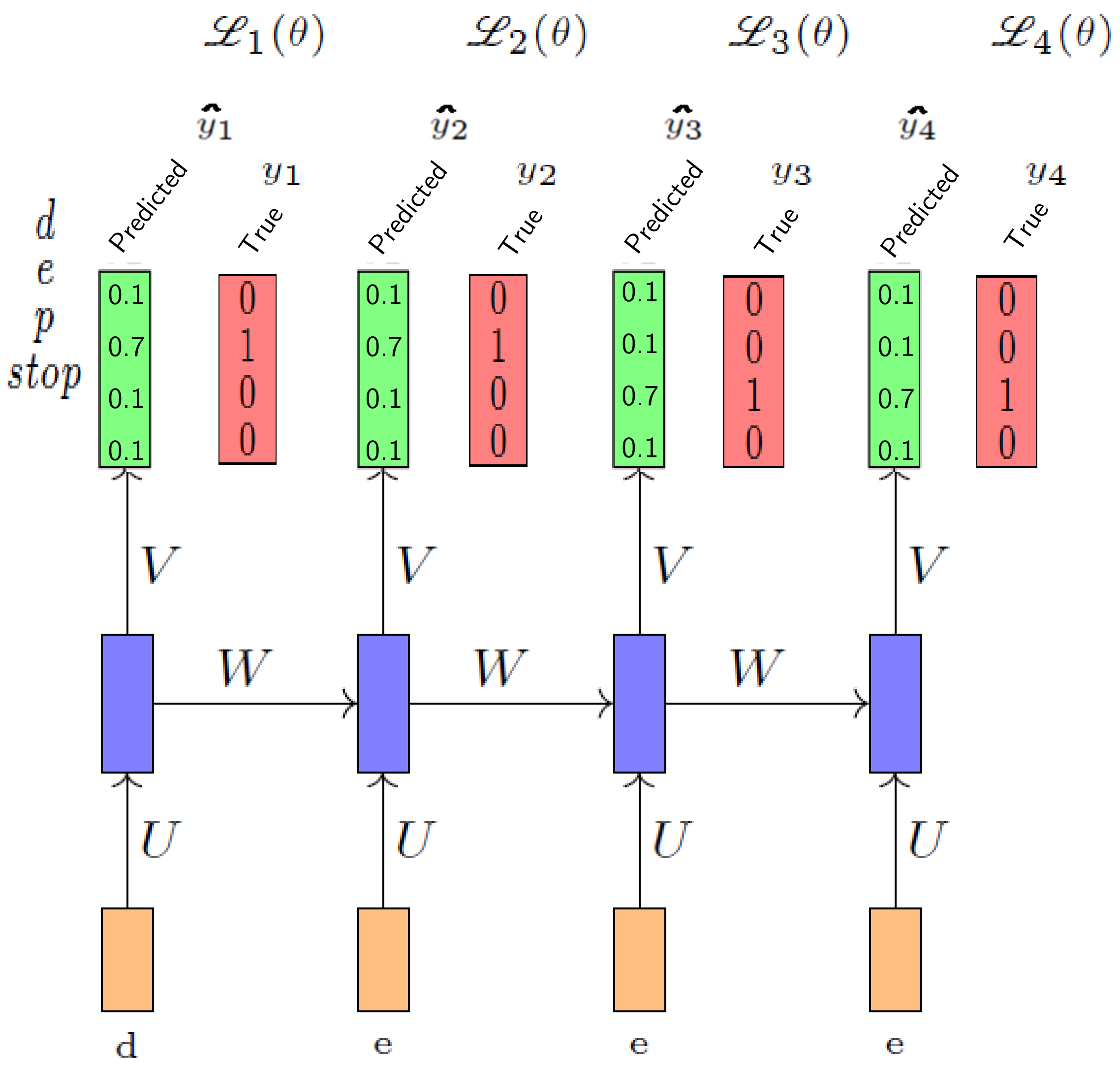
Training RNNs : Backpropagation through time (BPTT)



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

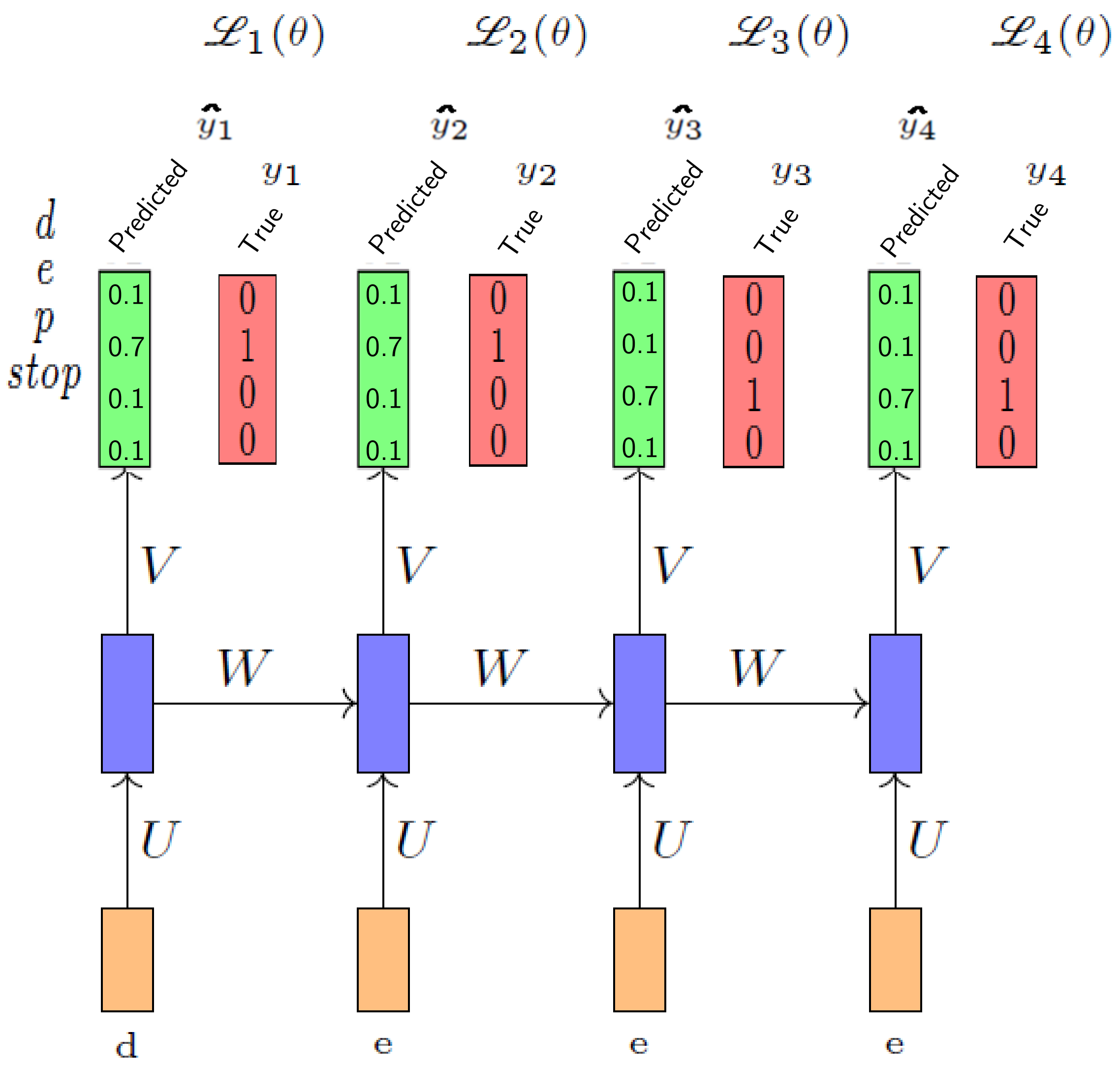
- For instance, consider the task of auto-completion (predicting the next character).
- For simplicity we assume that there are only 4 characters in our vocabulary (d, e, p, <stop>).
- At each timestep we want to predict one of these 4 characters.
- Suppose we initialize U, V, W randomly and the network predicts the probabilities (green block)
- And the true probabilities are as shown (red block).
- At each time step, the loss $L_i(\theta)$ is calculated, where $\theta = \{U,V,W,b,c\}$ is the set of parameters.

Training RNNs : Backpropagation through time (BPTT)



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

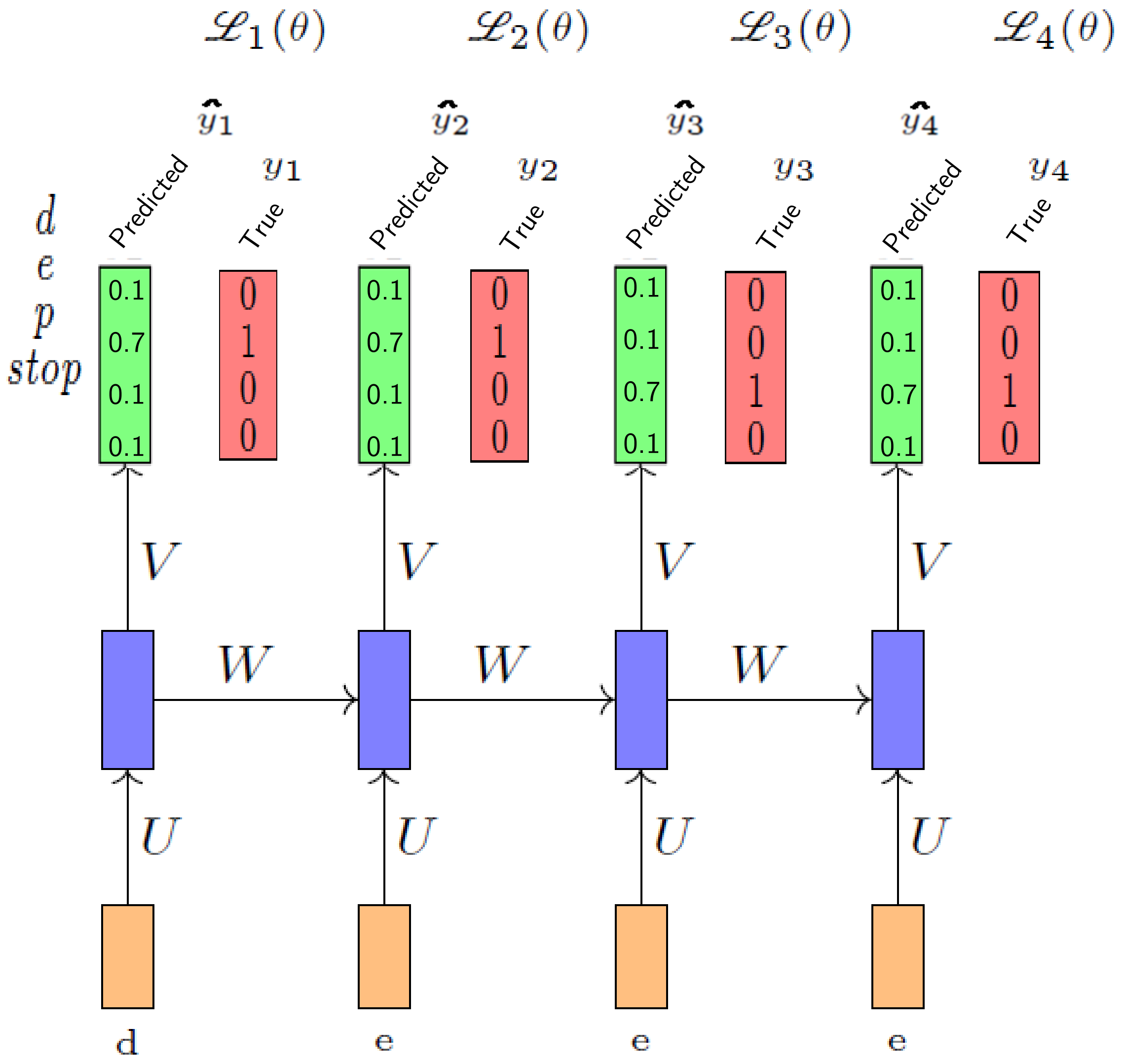
Training RNNs : Backpropagation through time (BPTT)



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

To train the RNNs we need to answer two questions:

Training RNNs : Backpropagation through time (BPTT)

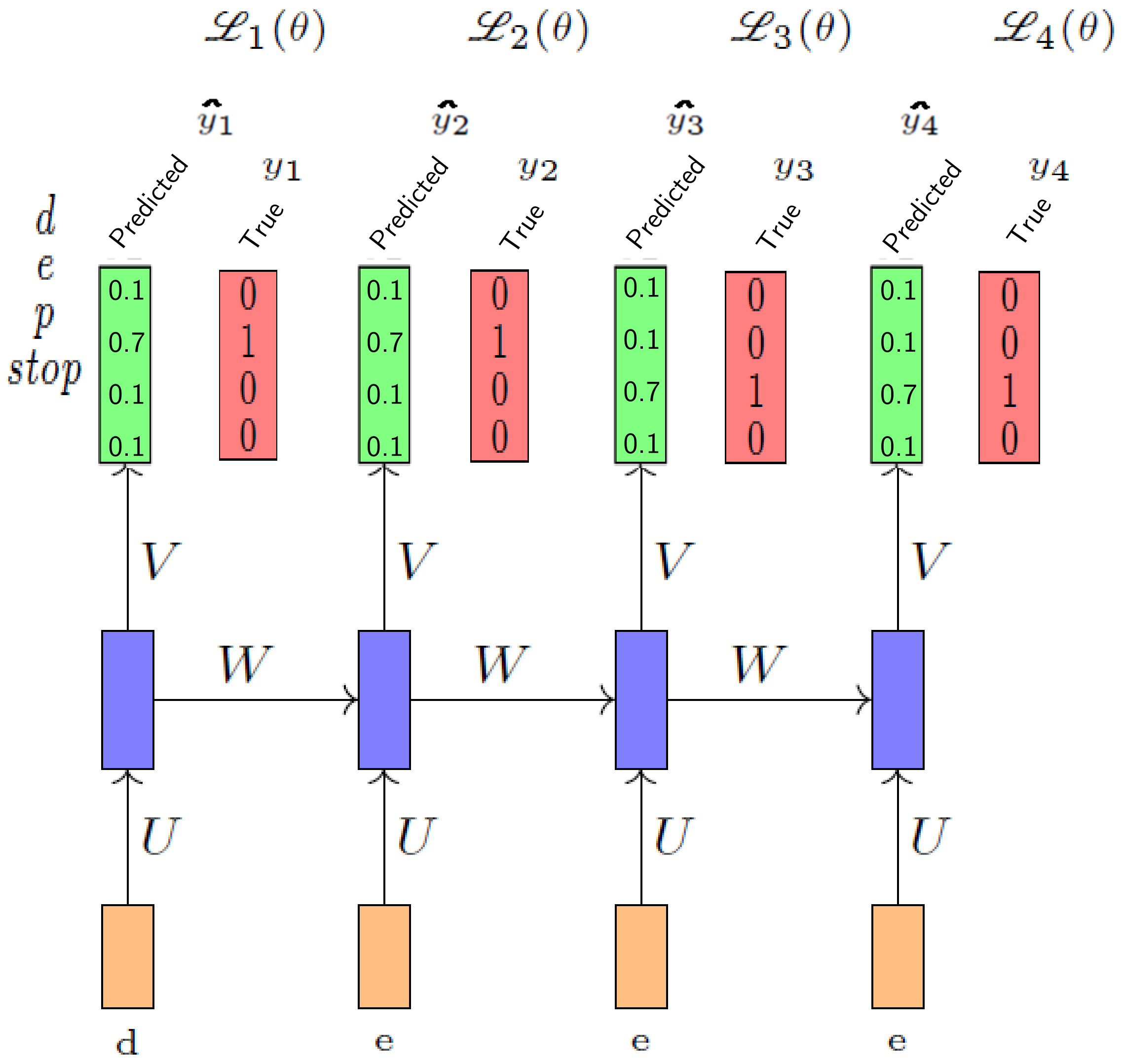


Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

To train the RNNs we need to answer two questions:

1) What is the total loss made by the model ?

Training RNNs : Backpropagation through time (BPTT)



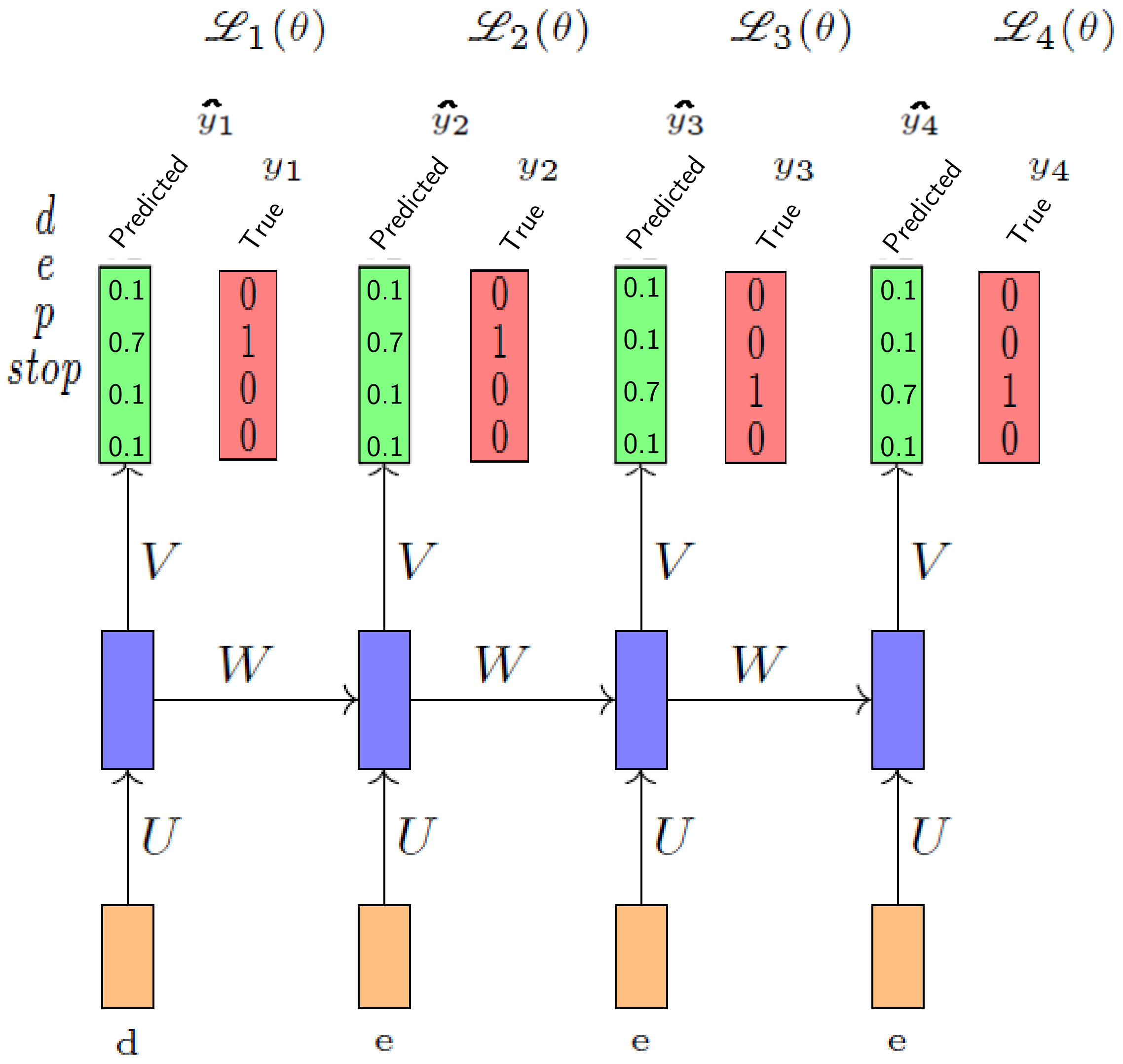
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

To train the RNNs we need to answer two questions:

1) What is the total loss made by the model ?

2) How do we backpropagate this loss and update the parameters of the network ?

Training RNNs : Backpropagation through time (BPTT)



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

To train the RNNs we need to answer two questions:

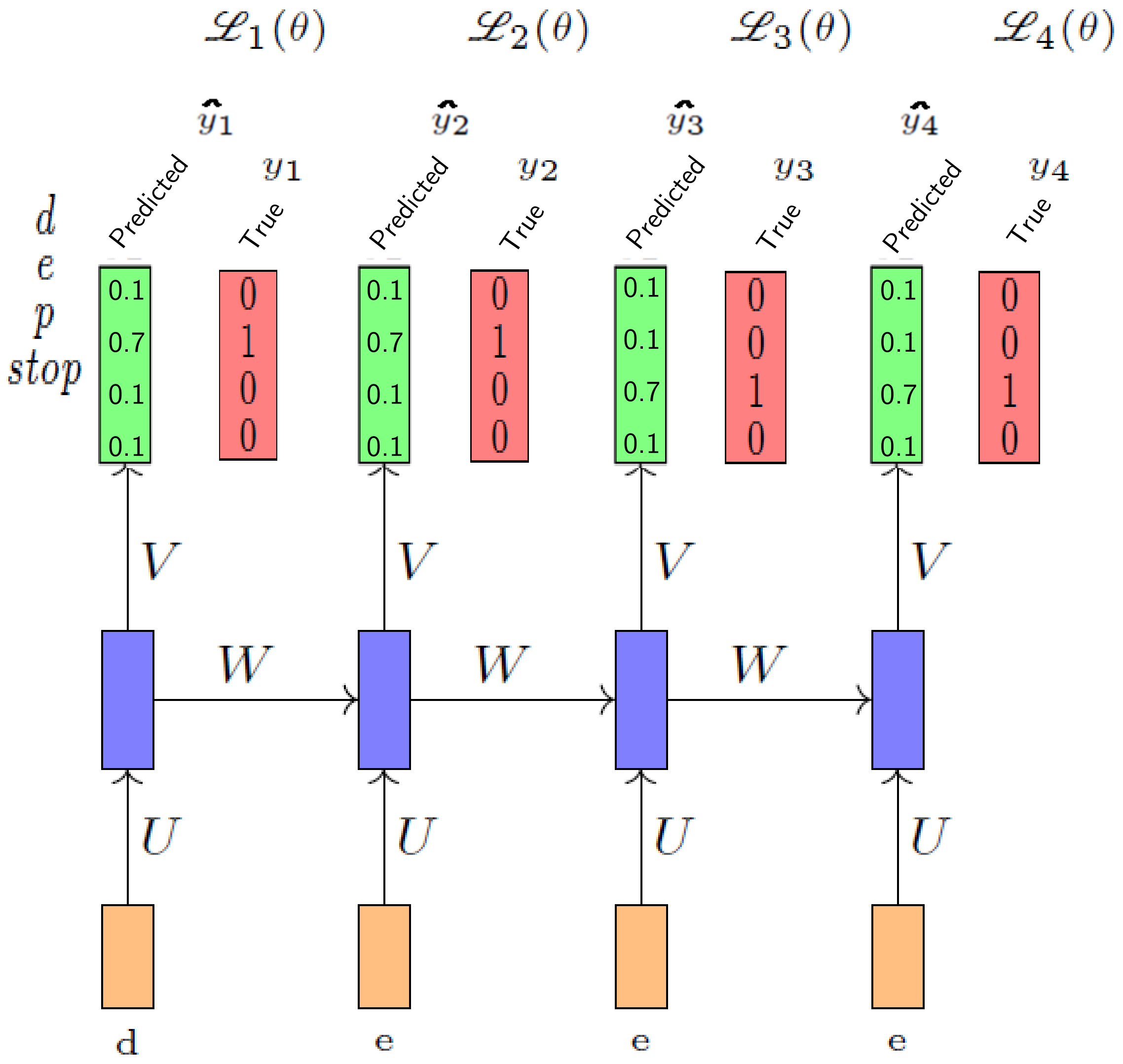
1) What is the total loss made by the model ?

Ans: the Sum of individual losses

$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta)$$

2) How do we backpropagate this loss and update the parameters of the network ?

Training RNNs : Backpropagation through time (BPTT)



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

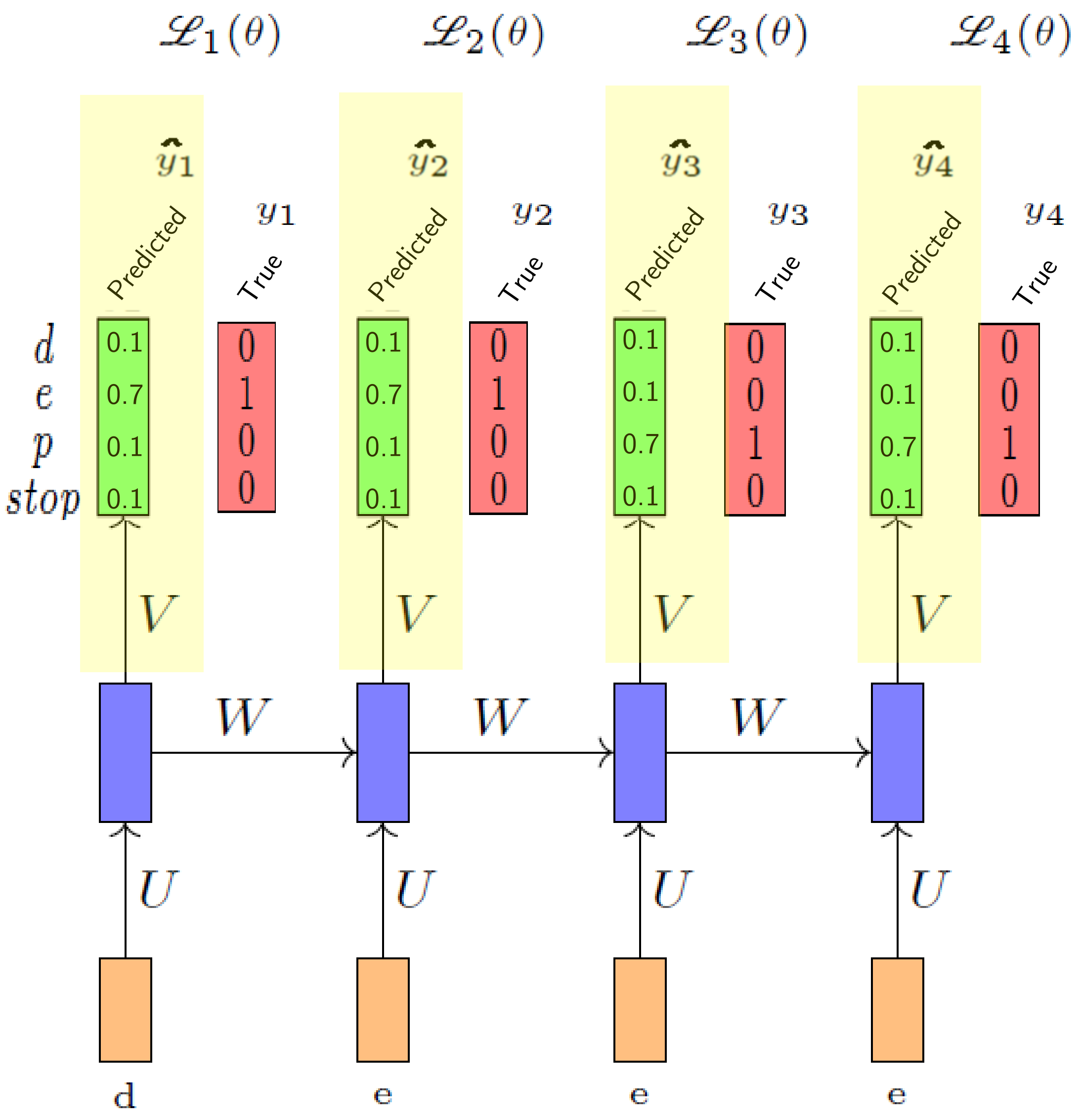
To train the RNNs we need to answer two questions:

1) What is the total loss made by the model ?
Ans: the Sum of individual losses

$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta)$$

2) How do we backpropagate this loss and update the parameters of the network ?
Ans: BPTT by computing the partial derivative of L w.r.t U, V, W, b, c

Training RNNs : Backpropagation through time (BPTT)



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

Let us consider $\frac{\partial \mathscr{L}(\theta)}{\partial V}$ (V is a matrix so ideally we should write $\nabla_v \mathscr{L}(\theta)$)

$$\frac{\partial \mathscr{L}(\theta)}{\partial V} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial V}$$

For example, if:

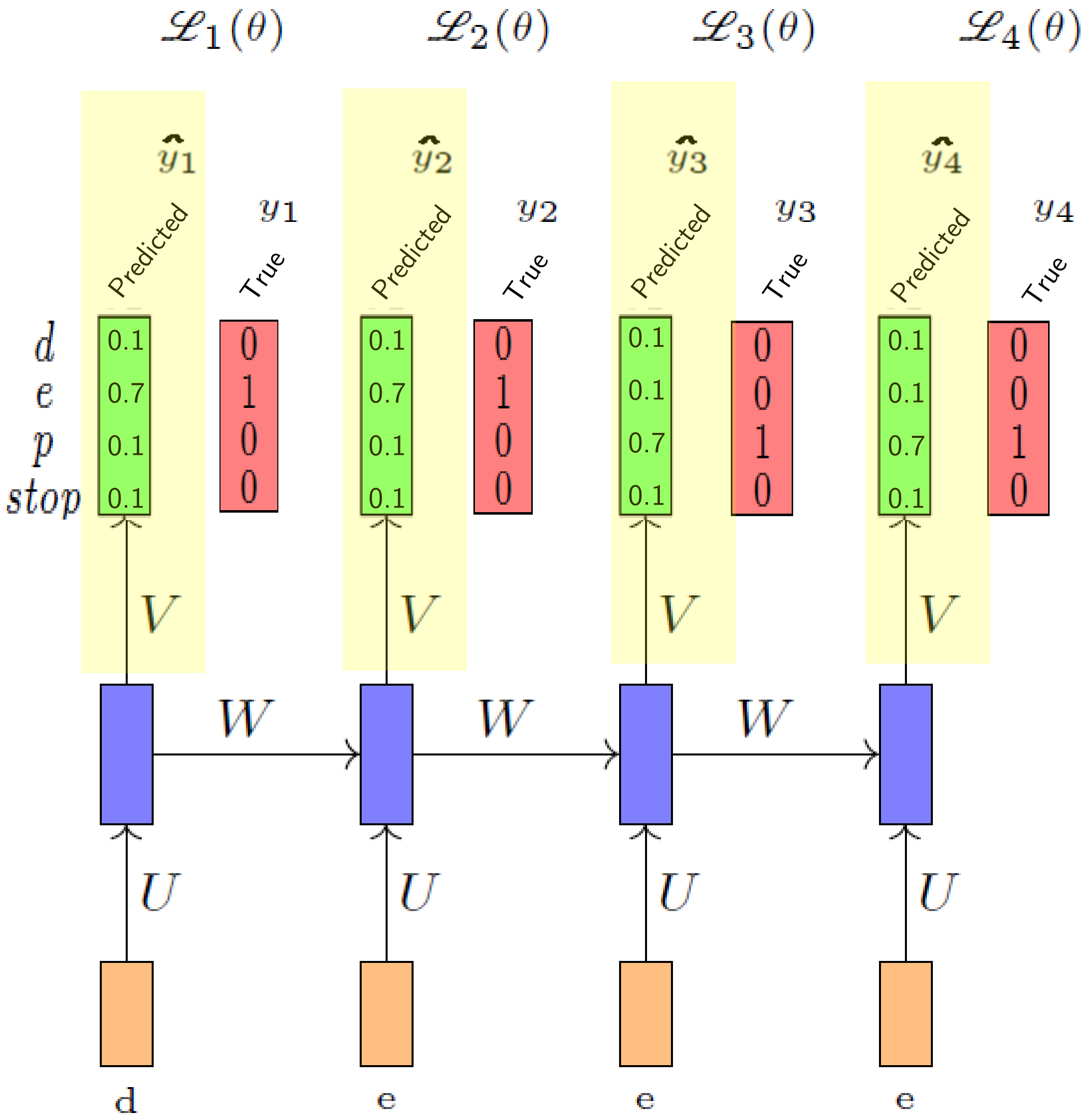
$\hat{y}_4 = O(VS_4 + c)$ and $L_4 = \frac{1}{2}(y_4 - \hat{y}_4)^2$

Ignoring bias and considering O as linear:

$$\frac{\partial L_4}{\partial V} = \frac{\partial L_4}{\partial \hat{y}_4} \frac{\partial \hat{y}_4}{\partial V}$$

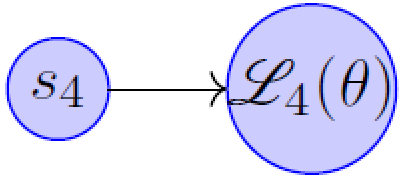
$$\frac{\partial L_4}{\partial V} = -(y_4 - \hat{y}_4) . s_4$$

Training RNNs : Backpropagation through time (BPTT)

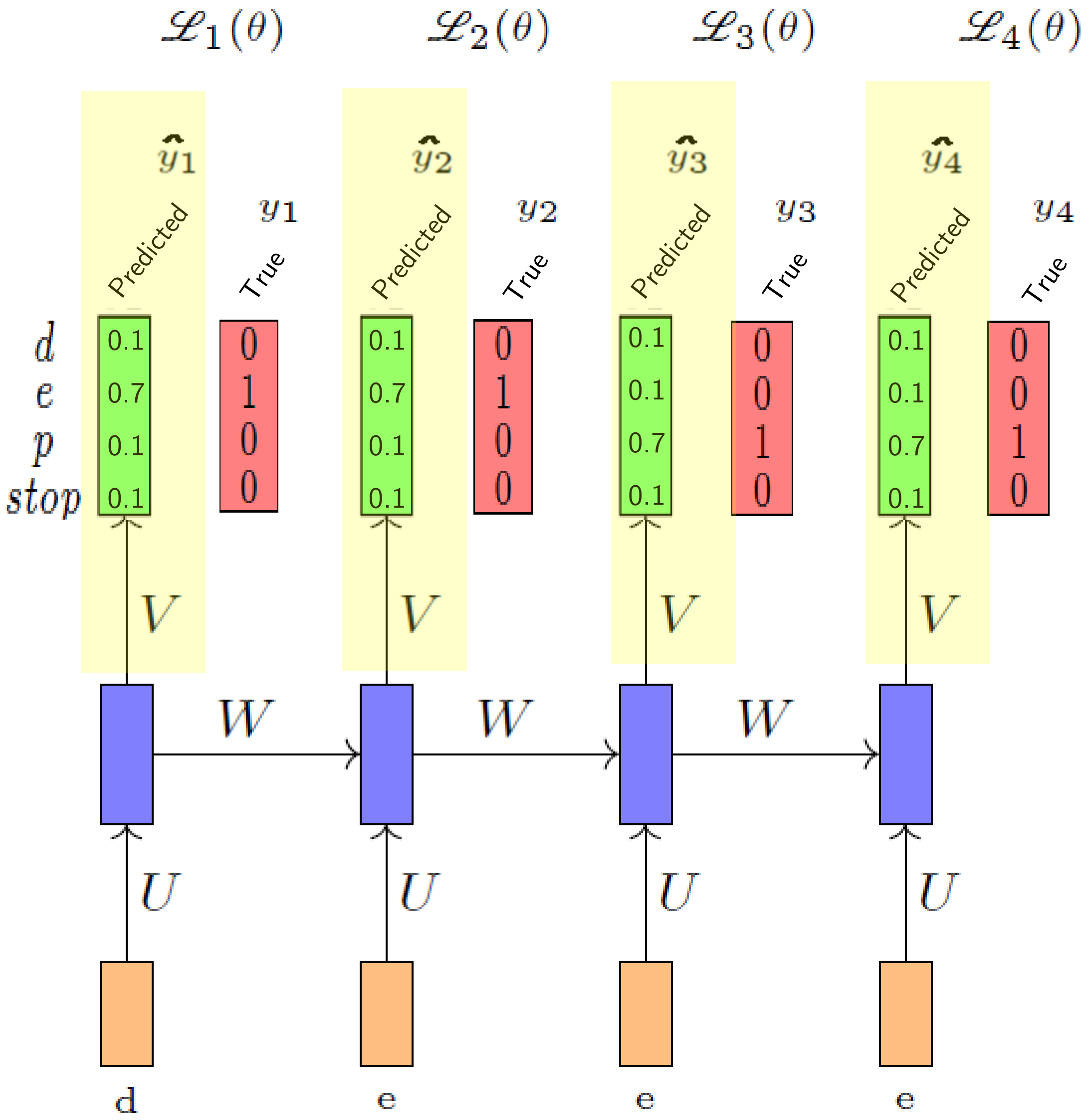


Let us consider the derivative $\frac{\partial \mathscr{L}(\theta)}{\partial W}$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial W}$$



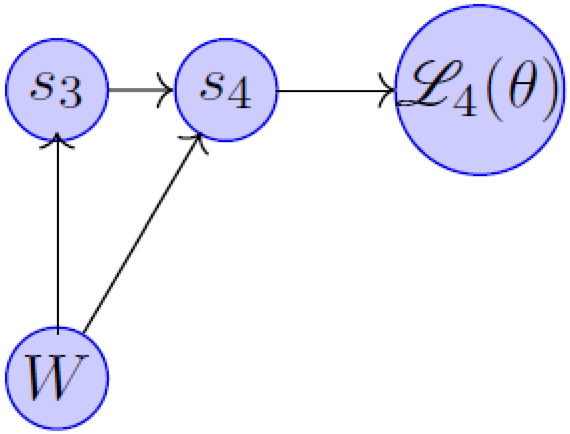
Training RNNs : Backpropagation through time (BPTT)



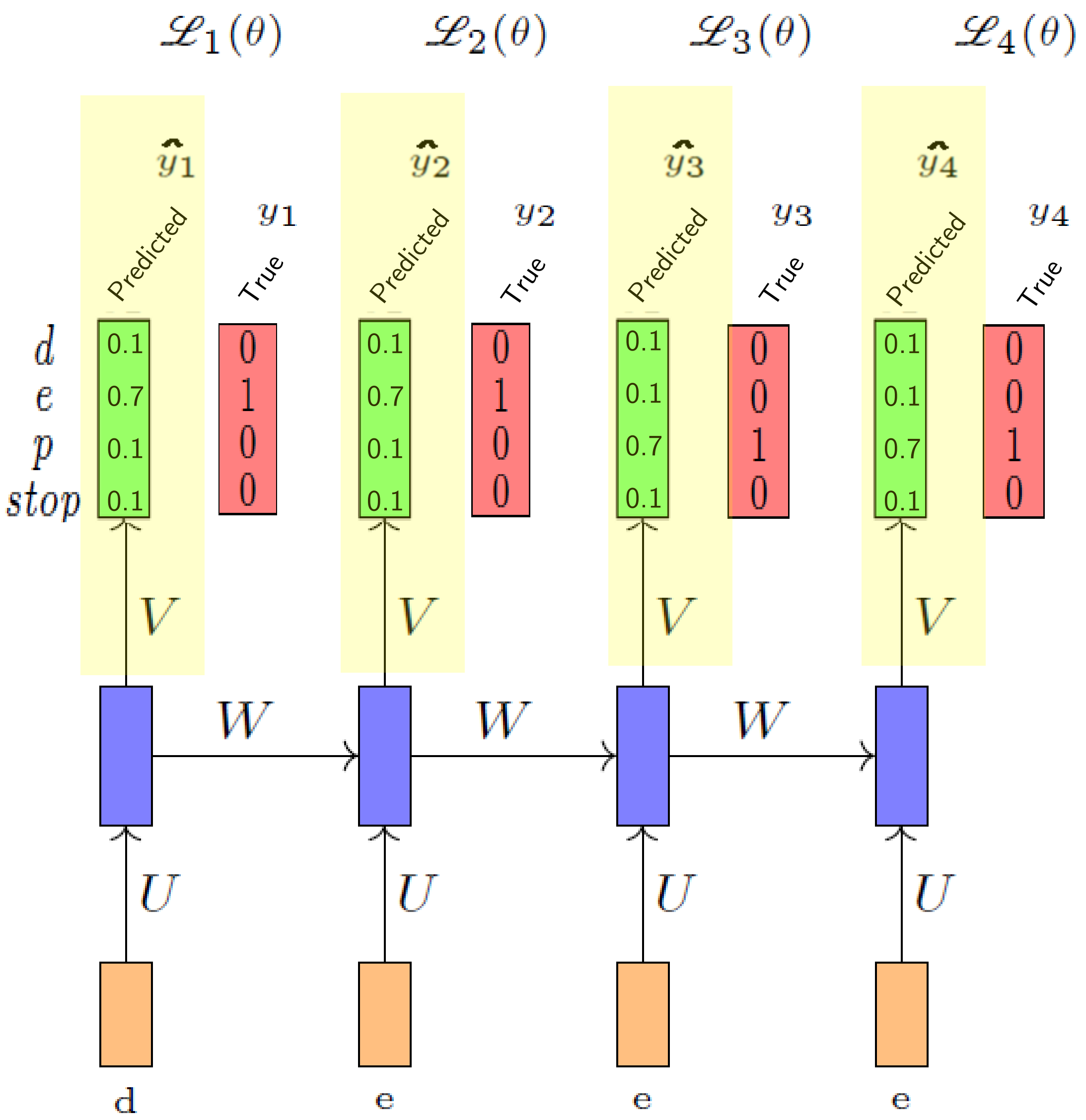
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Let us consider the derivative $\frac{\partial \mathscr{L}(\theta)}{\partial W}$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial W}$$



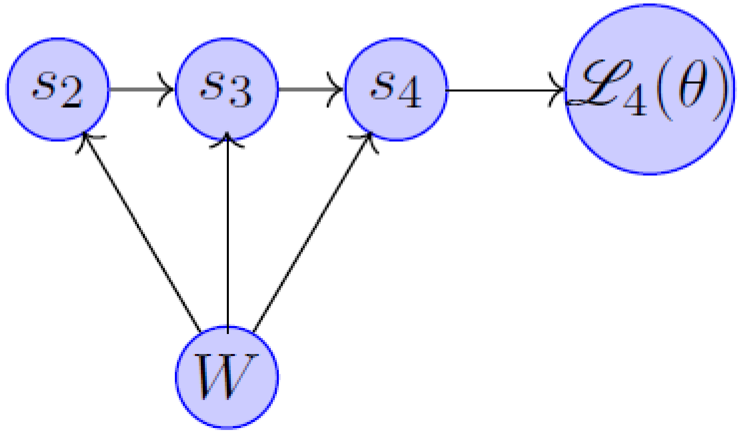
Training RNNs : Backpropagation through time (BPTT)



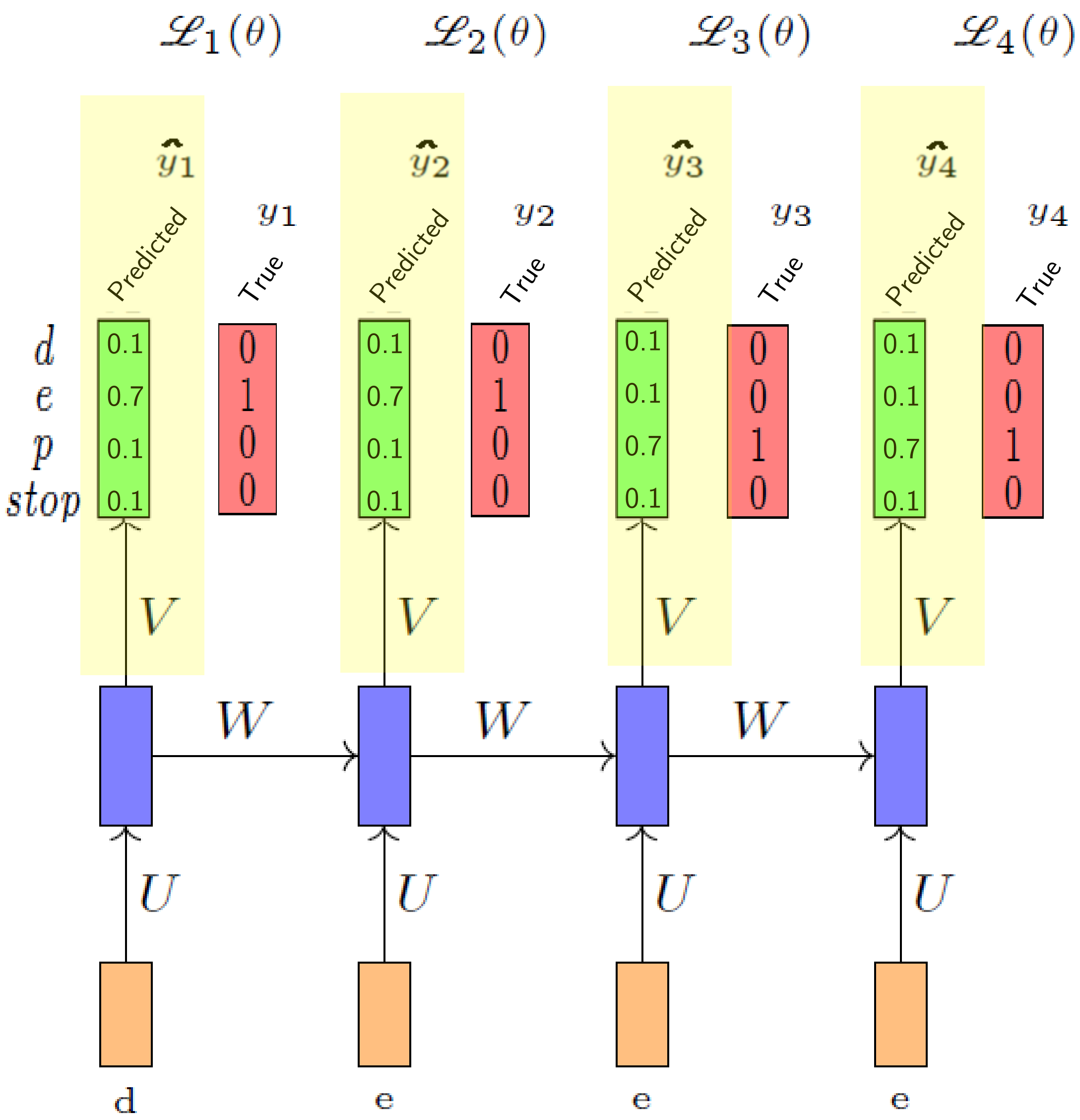
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Let us consider the derivative $\frac{\partial \mathscr{L}(\theta)}{\partial W}$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial W}$$



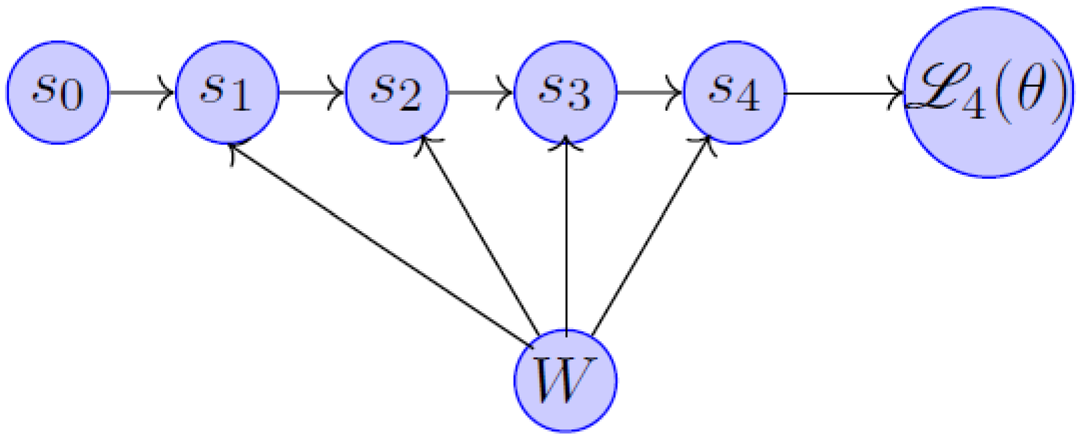
Training RNNs : Backpropagation through time (BPTT)



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

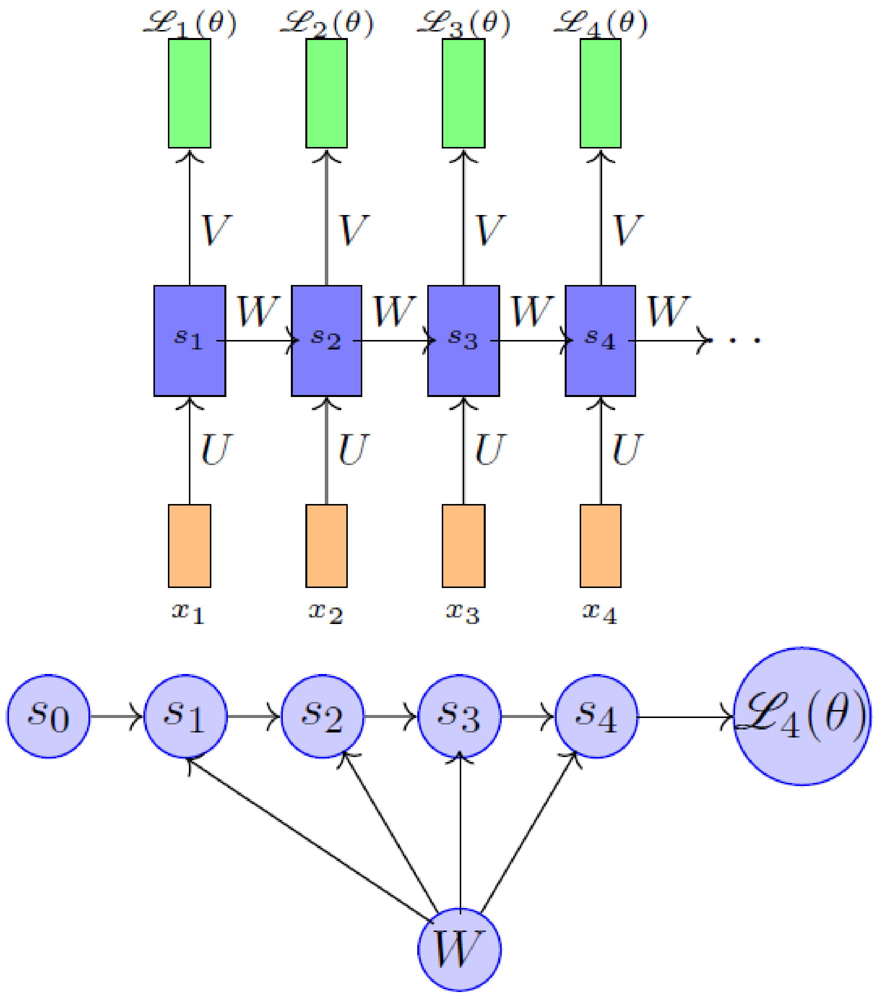
Let us consider the derivative $\frac{\partial \mathscr{L}(\theta)}{\partial W}$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial W}$$



Ordered network

Training RNNs : Backpropagation through time (BPTT)



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

$$\frac{\partial \mathscr{L}_4(\theta)}{\partial W} = \frac{\partial \mathscr{L}_4(\theta)}{\partial s_4}\frac{\partial s_4}{\partial W}$$

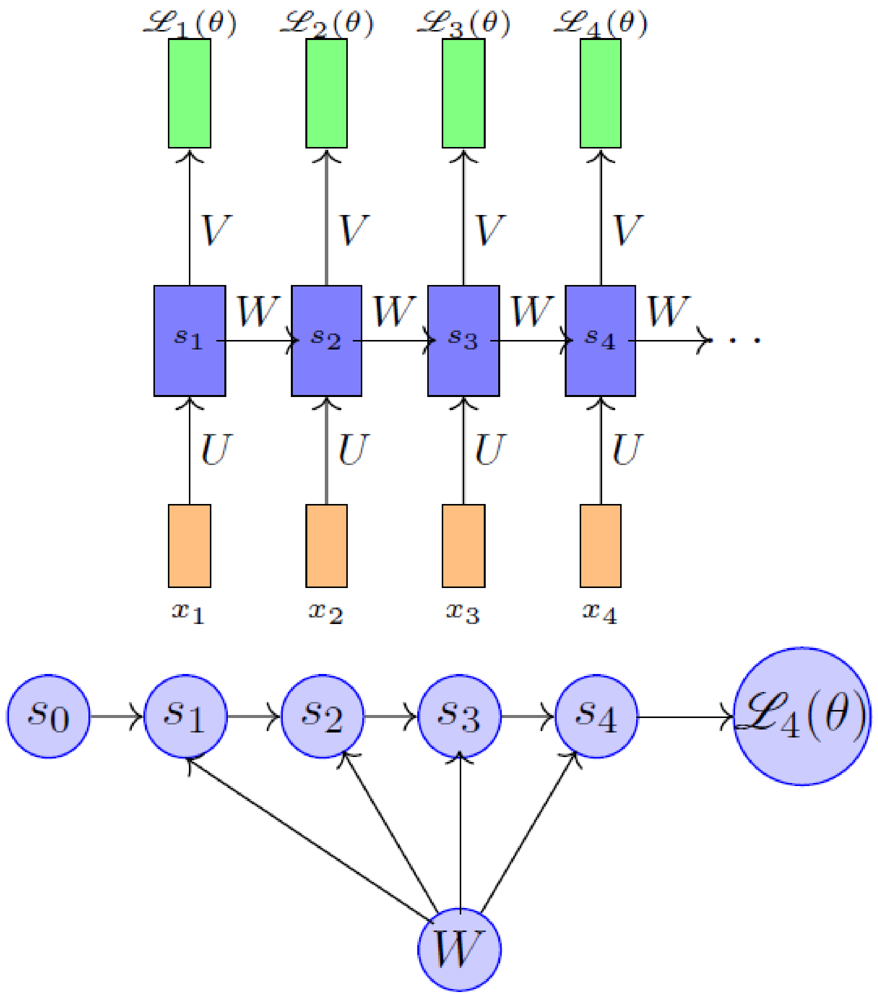
$\frac{\partial \mathscr{L}_4(\theta)}{\partial s_4}$ computation is straight forward

But how do we compute $\frac{\partial s_4}{\partial W}$

$$s_4 = \sigma(W s_3 + b)$$

In such an ordered network, we can't compute $\frac{\partial s_4}{\partial W}$ by simply treating s_3 as a constant (because it also depends on W)

Training RNNs : Backpropagation through time (BPTT)



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

But how do we compute $\frac{\partial s_4}{\partial W}$

In such networks the total derivative $\frac{\partial s_4}{\partial W}$ has two parts

Explicit : $\frac{\partial^+ s_4}{\partial W}$, treating all other inputs as constant

Implicit : Summing over all indirect paths from s_4 to W

Training RNNs : Backpropagation through time (BPTT)

$$\frac{\partial s_4}{\partial W} = \underbrace{\frac{\partial^+ s_4}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial W}}_{\text{implicit}}$$

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

Training RNNs : Backpropagation through time (BPTT)

$$\frac{\partial s_4}{\partial W} = \underbrace{\frac{\partial^+ s_4}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial W}}_{\text{implicit}}$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\left[\underbrace{\frac{\partial^+ s_3}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W}}_{\text{implicit}}\right]$$

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

Training RNNs : Backpropagation through time (BPTT)

$$\frac{\partial s_4}{\partial W} = \underbrace{\frac{\partial^+ s_4}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial W}}_{\text{implicit}}$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\left[\underbrace{\frac{\partial^+ s_3}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W}}_{\text{implicit}}\right]$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\left[\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W}\right]$$

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

Training RNNs : Backpropagation through time (BPTT)

$$\frac{\partial s_4}{\partial W} = \underbrace{\frac{\partial^+ s_4}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial W}}_{\text{implicit}}$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\left[\underbrace{\frac{\partial^+ s_3}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W}}_{\text{implicit}}\right]$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\left[\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W}\right]$$

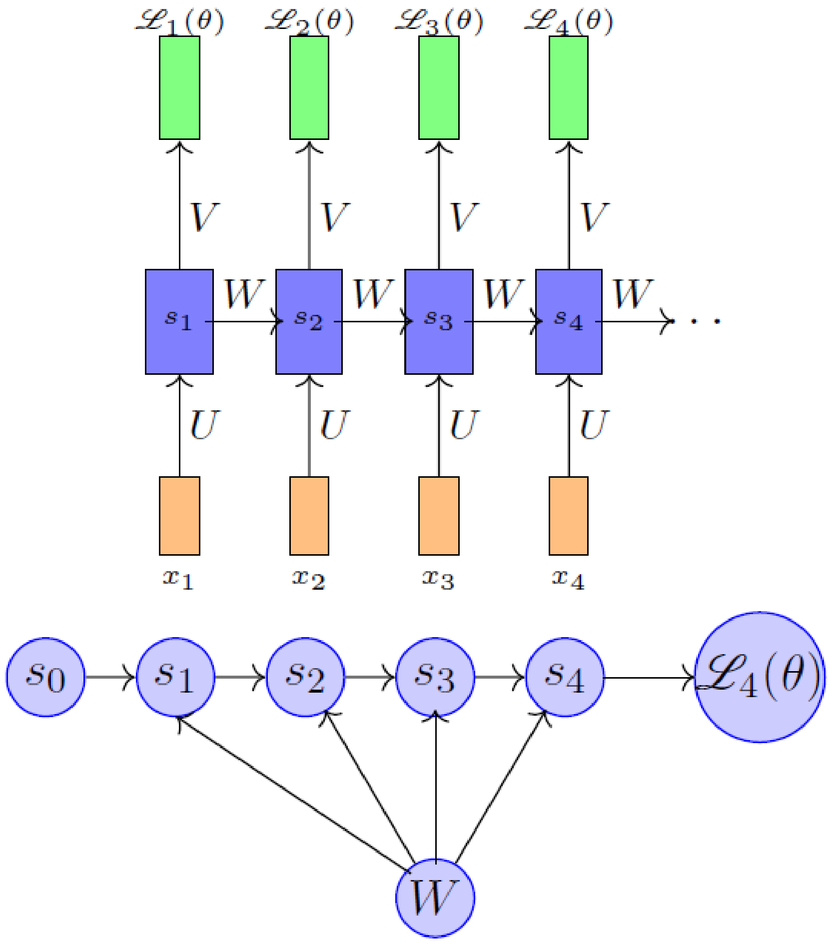
$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}\left[\frac{\partial^+ s_1}{\partial W}\right]$$

For simplicity we will short-circuit some of the paths

$$\frac{\partial s_4}{\partial W} = \frac{\partial s_4}{\partial s_4}\frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_2}\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_4}{\partial s_1}\frac{\partial^+ s_1}{\partial W} = \sum_{k=1}^{4}\frac{\partial s_4}{\partial s_k}\frac{\partial^+ s_k}{\partial W}$$

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

Training RNNs : Backpropagation through time (BPTT)



Finally we have

$$\frac{\partial \mathscr{L}_4(\theta)}{\partial W} = \frac{\partial \mathscr{L}_4(\theta)}{\partial s_4} \frac{\partial s_4}{\partial W}$$

$$\frac{\partial s_4}{\partial W} = \sum_{k=1}^{4} \frac{\partial s_4}{\partial s_k} \frac{\partial^+ s_k}{\partial W}$$

$$\therefore \frac{\partial \mathscr{L}_t(\theta)}{\partial W} = \frac{\partial \mathscr{L}_t(\theta)}{\partial s_t} \sum_{k=1}^{t} \frac{\partial s_t}{\partial s_k} \frac{\partial^+ s_k}{\partial W}$$

This algorithm is called backpropagation through time (BPTT) as we backpropagate over all previous time steps

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

We will now focus on $\frac{\partial s_t}{\partial s_k}$ and highlight an important problem in training RNN's using BPTT

$$\frac{\partial s_t}{\partial s_k} = \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \cdots \frac{\partial s_{k+1}}{\partial s_k} = \prod_{j=k}^{t-1} \frac{\partial s_{j+1}}{\partial s_j}$$

Let us look at one such term in the product (i.e., $\frac{\partial s_{j+1}}{\partial s_j}$)

Recall that:

$$a_j = W s_{j-1} + b$$
$$s_j = \sigma(a_j)$$

Therefore:

$$\frac{\partial s_{j+1}}{\partial s_j} = \frac{\partial s_j}{\partial s_{j-1}} = \frac{\partial s_j}{\partial a_j} \frac{\partial a_j}{\partial s_{j-1}}$$

$$a_j = [a_{j1}, a_{j2}, a_{j3}, \ldots a_{jd},]$$
$$s_j = [\sigma(a_{j1}), \sigma(a_{j2}), \ldots \sigma(a_{jd})]$$

$$\frac{\partial s_j}{\partial a_j} = \left[\quad \right]$$

Back Propagation through time – Vanishing & Exploding gradient

We will now focus on $\frac{\partial s_t}{\partial s_k}$ and highlight an important problem in training RNN's using BPTT

$$\frac{\partial s_t}{\partial s_k} = \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \dots \frac{\partial s_{k+1}}{\partial s_k}$$

Let us look at one such term in the product (i.e., $\frac{\partial s_{j+1}}{\partial s_j}$)

$$a_j = W s_{j-1} + b$$
$$s_j = \sigma(a_j)$$

$$\frac{\partial s_j}{\partial s_{j-1}} = \frac{\partial s_j}{\partial a_j} \frac{\partial a_j}{\partial s_{j-1}}$$

$$a_j = [a_{j1}, a_{j2}, a_{j3}, \dots a_{jd},]$$
$$s_j = [\sigma(a_{j1}), \sigma(a_{j2}), \dots \sigma(a_{jd})]$$

$$\frac{\partial s_j}{\partial a_j} = \begin{bmatrix} \frac{\partial s_{j1}}{\partial a_{j1}} & \frac{\partial s_{j2}}{\partial a_{j1}} & \frac{\partial s_{j3}}{\partial a_{j1}} & \dots \\ \frac{\partial s_{j1}}{\partial a_{j2}} & \frac{\partial s_{j2}}{\partial a_{j2}} & \ddots & \\ \vdots & \vdots & \vdots & \frac{\partial s_{jd}}{\partial a_{jd}} \end{bmatrix}$$

$$= \begin{bmatrix} \sigma'(a_{j1}) & 0 & 0 & 0 \\ 0 & \sigma'(a_{j2}) & 0 & 0 \\ 0 & 0 & \ddots & \\ 0 & 0 & \dots & \sigma'(a_{jd}) \end{bmatrix}$$

Back Propagation through time – Vanishing & Exploding gradient

We will now focus on $\frac{\partial s_t}{\partial s_k}$ and highlight an important problem in training RNN's using BPTT

$$\frac{\partial s_t}{\partial s_k} = \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \cdots \frac{\partial s_{k+1}}{\partial s_k}$$

Let us look at one such term in the product (i.e., $\frac{\partial s_{j+1}}{\partial s_j}$)

$$a_j = W s_{j-1} + b$$
$$s_j = \sigma(a_j)$$

$$\frac{\partial s_j}{\partial s_{j-1}} = \frac{\partial s_j}{\partial a_j} \frac{\partial a_j}{\partial s_{j-1}}$$

$$a_j = [a_{j1}, a_{j2}, a_{j3}, \ldots a_{jd},]$$
$$s_j = [\sigma(a_{j1}), \sigma(a_{j2}), \ldots \sigma(a_{jd})]$$

$$\frac{\partial s_j}{\partial a_j} = \begin{bmatrix} \frac{\partial s_{j1}}{\partial a_{j1}} & \frac{\partial s_{j2}}{\partial a_{j1}} & \frac{\partial s_{j3}}{\partial a_{j1}} & \cdots \\ \frac{\partial s_{j1}}{\partial a_{j2}} & \frac{\partial s_{j2}}{\partial a_{j2}} & \ddots & \\ \vdots & \vdots & \vdots & \frac{\partial s_{jd}}{\partial a_{jd}} \end{bmatrix}$$

$$= \begin{bmatrix} \sigma'(a_{j1}) & 0 & 0 & 0 \\ 0 & \sigma'(a_{j2}) & 0 & 0 \\ 0 & 0 & \ddots & \\ 0 & 0 & \ldots & \sigma'(a_{jd}) \end{bmatrix}$$

$$= diag(\sigma'(a_j))$$

Back Propagation through time – Vanishing & Exploding gradient

We will now focus on $\frac{\partial s_t}{\partial s_k}$ and highlight an important problem in training RNN's using BPTT

$$\frac{\partial s_t}{\partial s_k} = \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \cdots \frac{\partial s_{k+1}}{\partial s_k}$$

Let us look at one such term in the product (i.e., $\frac{\partial s_{j+1}}{\partial s_j}$)

$$a_j = W s_{j-1} + b$$
$$s_j = \sigma(a_j)$$

$$\frac{\partial s_j}{\partial s_{j-1}} = \frac{\partial s_j}{\partial a_j} \frac{\partial a_j}{\partial s_{j-1}}$$

$$\boxed{\text{We are interested in the magnitude of } \frac{\partial s_j}{\partial s_{j-1}} \leftarrow \text{if it is small (large) } \frac{\partial s_t}{\partial s_k} \text{ and hence } \frac{\partial \mathscr{L}_t}{\partial W} \text{ will vanish (explode)}}$$

$$a_j = [a_{j1}, a_{j2}, a_{j3}, \ldots a_{jd},]$$
$$s_j = [\sigma(a_{j1}), \sigma(a_{j2}), \ldots \sigma(a_{jd})]$$

$$\frac{\partial s_j}{\partial a_j} = \begin{bmatrix} \frac{\partial s_{j1}}{\partial a_{j1}} & \frac{\partial s_{j2}}{\partial a_{j1}} & \frac{\partial s_{j3}}{\partial a_{j1}} & \cdots \\ \frac{\partial s_{j1}}{\partial a_{j2}} & \frac{\partial s_{j2}}{\partial a_{j2}} & \ddots & \\ \vdots & \vdots & \vdots & \frac{\partial s_{jd}}{\partial a_{jd}} \end{bmatrix}$$

$$= \begin{bmatrix} \sigma'(a_{j1}) & 0 & 0 & 0 \\ 0 & \sigma'(a_{j2}) & 0 & 0 \\ 0 & 0 & \ddots & \\ 0 & 0 & \ldots & \sigma'(a_{jd}) \end{bmatrix}$$

$$= diag(\sigma'(a_j))$$

Back Propagation through time – Vanishing & Exploding gradient

$$\left\|\frac{\partial s_j}{\partial s_{j-1}}\right\| = \left\|diag(\sigma'(a_j))W\right\|$$
$$\leq \left\|diag(\sigma'(a_j))\right\| \|W\|$$

$\because \sigma(a_j)$ is a bounded function (sigmoid, tanh) $\sigma'(a_j)$ is bounded

$$\sigma'(a_j) \leq \frac{1}{4} = \gamma \text{ [if } \sigma \text{ is logistic]}$$
$$\leq 1 = \gamma \text{ [if } \sigma \text{ is tanh]}$$

$$\left\|\frac{\partial s_j}{\partial s_{j-1}}\right\| \leq \gamma \|W\|$$
$$\leq \gamma\lambda$$

$$\left\|\frac{\partial s_t}{\partial s_k}\right\| = \left\|\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}}\right\|$$
$$\leq \prod_{j=k+1}^{t} \gamma\lambda$$
$$\leq (\gamma\lambda)^{t-k}$$

If $\gamma\lambda < 1$ the gradient will vanish

If $\gamma\lambda > 1$ the gradient could explode

Back Propagation through time – Vanishing & Exploding gradient

And that means the first input value is amplified **16** times before it gets to the final copy of the network.

$\mathbf{Input_1} \times 2 \times 2 \times 2 \times 2$

$= \mathbf{Input_1} \times 2^4$

$= \mathbf{Input_1} \times \boldsymbol{w_2}^{\mathbf{Num.\ Unroll}}$

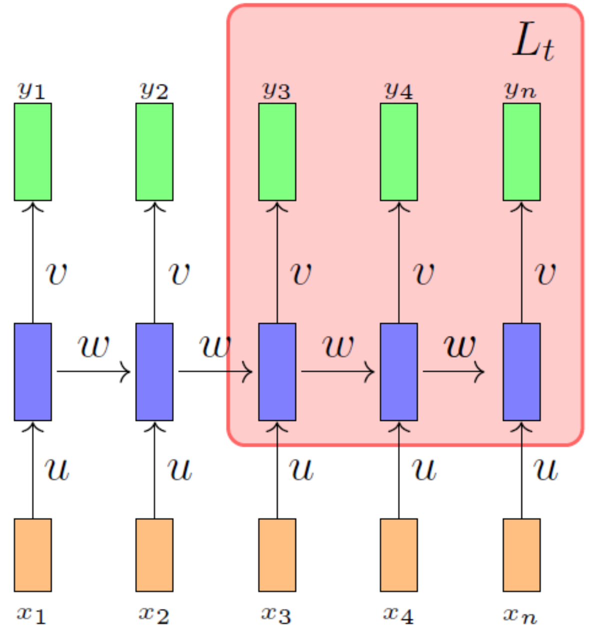
Back Propagation through time in RNNs : Issues & Solutions

1. Gradient calculations are expensive (slow training for long sequences)
   - Solution: Truncated BPTT

2. Exploding gradients

3. Vanishing gradients

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



- Instead of looking at all 'n' time steps, we would look at lesser time steps allowing us to estimate rather than calculate the gradient used to update the weights.

Back Propagation through time in RNNs : Issues & Solutions

1. Gradient calculations are expensive (slow training for long sequences)
    - Solution: Truncated BPTT

2. Exploding gradients
    - Solution: Gradient Clipping

3. Vanishing gradients

Let $g = \frac{\partial L}{\partial W}$

I. Clipping by value:

if $\|\mathbf{g}\| \geq$ **max_threshold** then:

$\mathbf{g} \leftarrow$ **threshold**

end if

II. Clipping by norm:

if $\|\mathbf{g}\| \geq$ **threshold** then:

$\mathbf{g} \leftarrow$ **threshold** * $\mathbf{g}/\|\mathbf{g}\|$

end if

Back Propagation through time in RNNs : Issues & Solutions

1. Gradient calculations are expensive (slow training for long sequences)
   - Solution: Truncated BPTT

2. Exploding gradients
   - Solution: Gradient Clipping

3. Vanishing gradients
   - Solution: Use alternate RNN architectures such as LSTM and GRU.