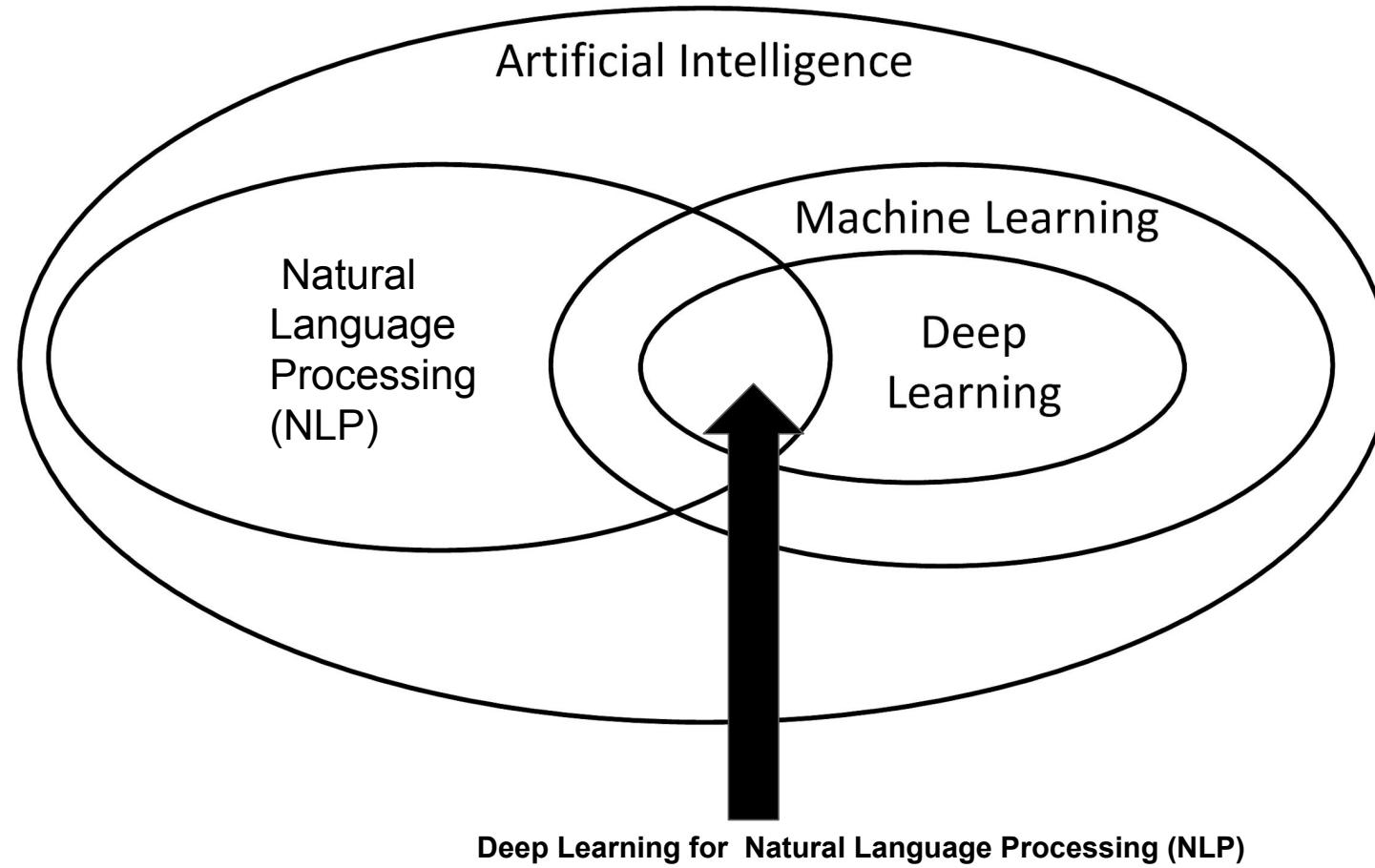


# Deep Learning for Natural Language Processing (NLP)

Artificial Intelligence

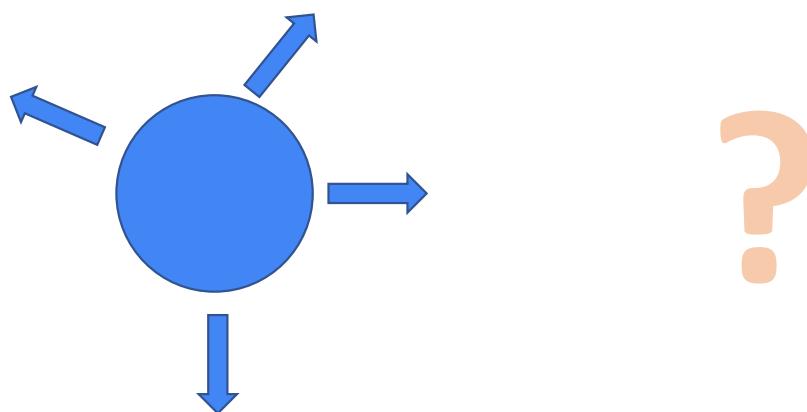
Machine Learning

Natural  
Language  
Processing  
(NLP)

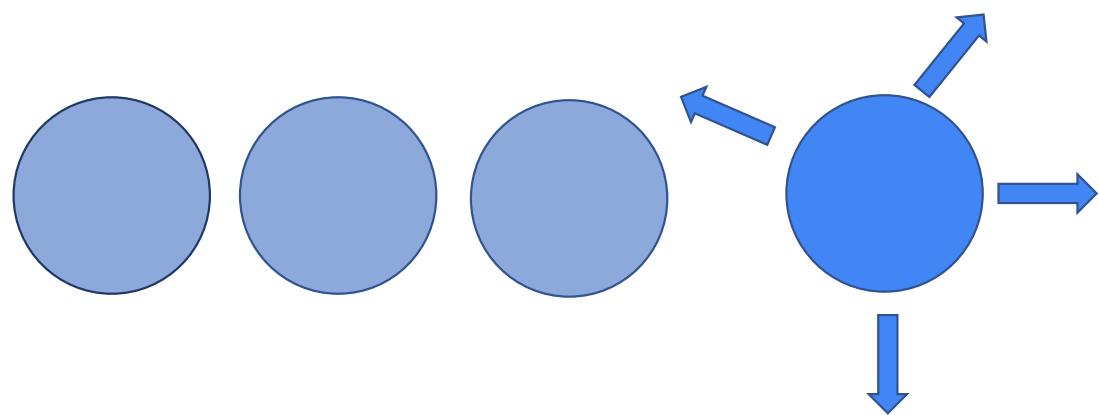


# Sequence Modelling

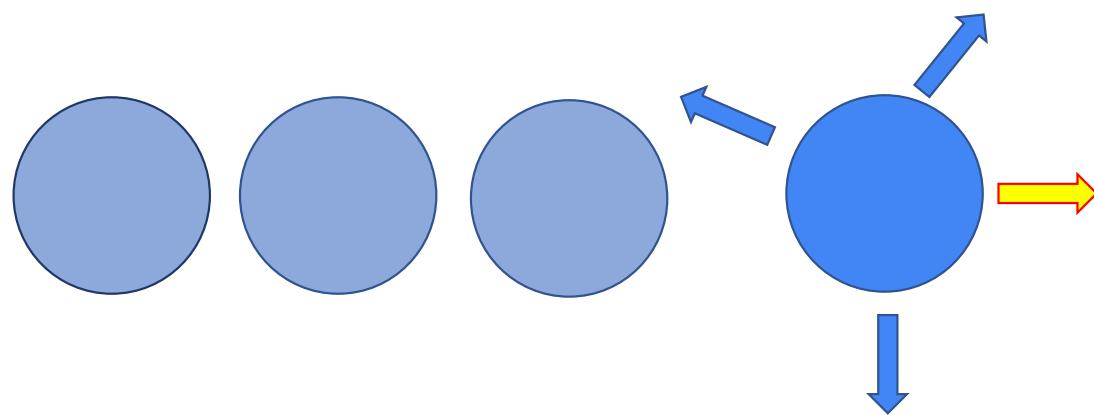
# Sequence modeling



# Sequence modeling



# Sequence modeling



# Sequences are Everywhere

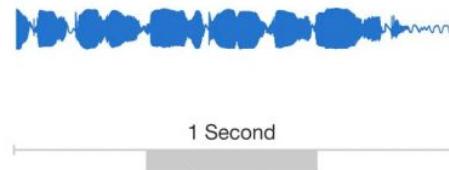
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

**Words, letters**

# Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters

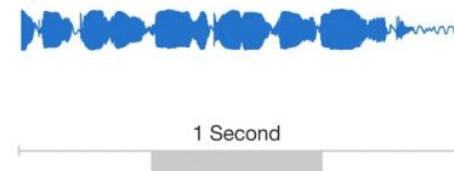


Speech

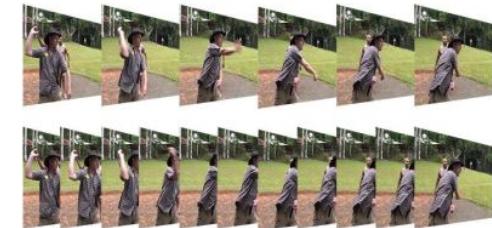
# Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



Speech

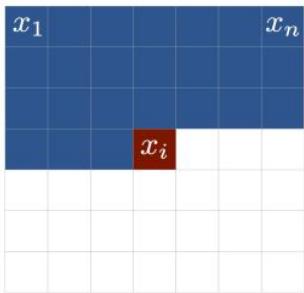


Videos

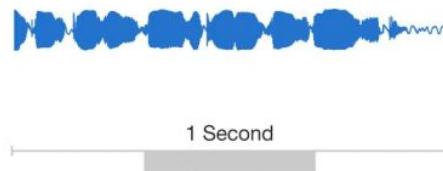
# Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



Images



Speech

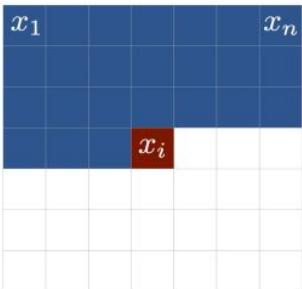


Videos

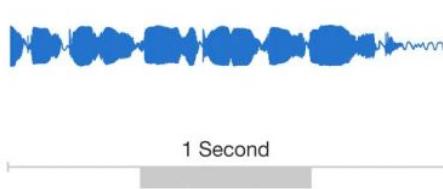
# Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



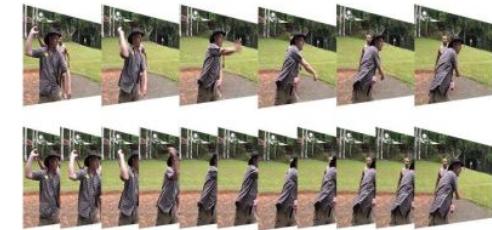
Images



Speech

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def forward_backward_prop(w, T):
5     hs = [0.5]
6     for i in range(T):
7         hs.append(np.tanh(w*hs[-1]))
8
9     dh = 1
10    for t in range(T):
11        dh = (1-hs[-t-1]**2) * w * dh
12
13    return hs[-1], dh
14
15 T = 10 # sequence length
16 wlim = 4 # limit of interval over weights w
17
18 results = []
19 ws = np.linspace(-wlim, wlim, 1000)
20 for w in ws:
21     results.append(forward_backward_prop(w, T))
22
23 plt.plot(ws, [r[0] for r in results], label='RNN state')
24 plt.plot(ws, [r[1] for r in results], label='Gradients')
```

Programs

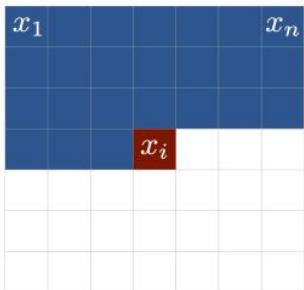


Videos

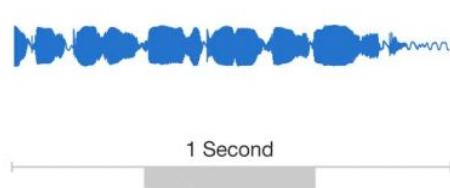
# Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



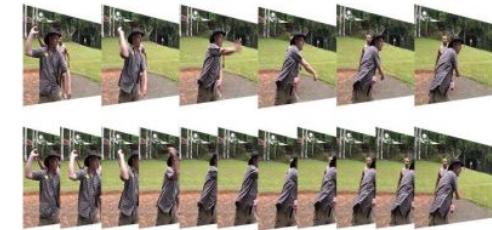
Images



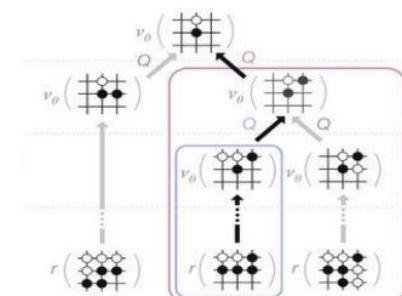
Speech

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def forward_backward_prop(w, T):
5     hs = [0.5]
6     for t in range(T):
7         hs.append(np.tanh(w*hs[-1]))
8
9     dh = 1
10    for t in range(T):
11        dh = (1-hs[-t-1]**2) * w * dh
12
13    return hs[-1], dh
14
15 T = 10 # sequence length
16 wlim = 4 # limit of interval over weights w
17
18 results = []
19 ws = np.linspace(-wlim, wlim, 1000)
20 for w in ws:
21     results.append(forward_backward_prop(w, T))
22
23 plt.plot(ws, [r[0] for r in results], label='RNN state')
24 plt.plot(ws, [r[1] for r in results], label='Gradients')
```

Programs



Videos



Decision making

# NLP Tasks Overview

# Language Translation

The image shows a Google search results page for "Language Translation". The search bar contains the query. Below it, a navigation bar includes "All" (selected), Books, News, Images, Videos, More, and Tools. The search results indicate approximately 12.77 billion results found in 0.59 seconds. The main content area displays a translation pair: "good morning" in English is translated to "शुभ प्रभात" (shubh prabhaat) in Hindi. Both terms have pronunciation guides: "good mōrning" and "shubh prabhaat". At the bottom, there are microphone and speaker icons for audio playback, and a "Verified" checkmark icon.

Language Translation

All Books News Images Videos More Tools

About 12,77,00,00,000 results (0.59 seconds)

English ▾ Hindi ▾

good morning × शुभ प्रभात  
good mōrning shubh prabhaat

Verified

# Query Recommendations

The screenshot shows a search interface with a search bar at the top containing the partially typed query "The sma". To the right of the search bar are a close button (an 'X') and a microphone icon for voice search. Below the search bar is a list of ten query suggestions, each preceded by a magnifying glass icon:

- the smallest whole number is
- the smart shop
- the smallest composite number is
- the smallest prime number is
- the smallest natural number is
- the smallest
- the smallest 4 digit number
- the smallest unit of memory is
- the smallest counting number is
- the smallest 5 digit number

# Spelling and Grammar Corrections

The screenshot shows a Google Docs interface. In the top left corner, there's a green circular icon with a white letter 'G' and a three-dot menu icon. To the right, the title 'Untitled document' is displayed. The main content area contains two paragraphs of text:

This is an exempl document.  
This is test statement.

The word 'exempl' in the first sentence and 'test' in the second sentence are underlined with red, indicating they are misspelled. The word 'document.' in the first sentence and 'statement.' in the second sentence are underlined with blue, indicating they are grammatically incorrect or flagged by the spell checker.

# Sentiment Analysis

★★★★★ **Excellent taste**

Reviewed in India on 30 March 2020

Size: 504 g (Pack of 36) | **Verified Purchase**

I'm hooked on to Girnar teas. They are excellent and taste great. Easy to dissolve and hassle free. If you are the kind of person who wants a consistent taste for their chai, then switch to Girnar and carry the sachets wherever you go. A tad expensive though

★★★★★ **Poorest in quality**

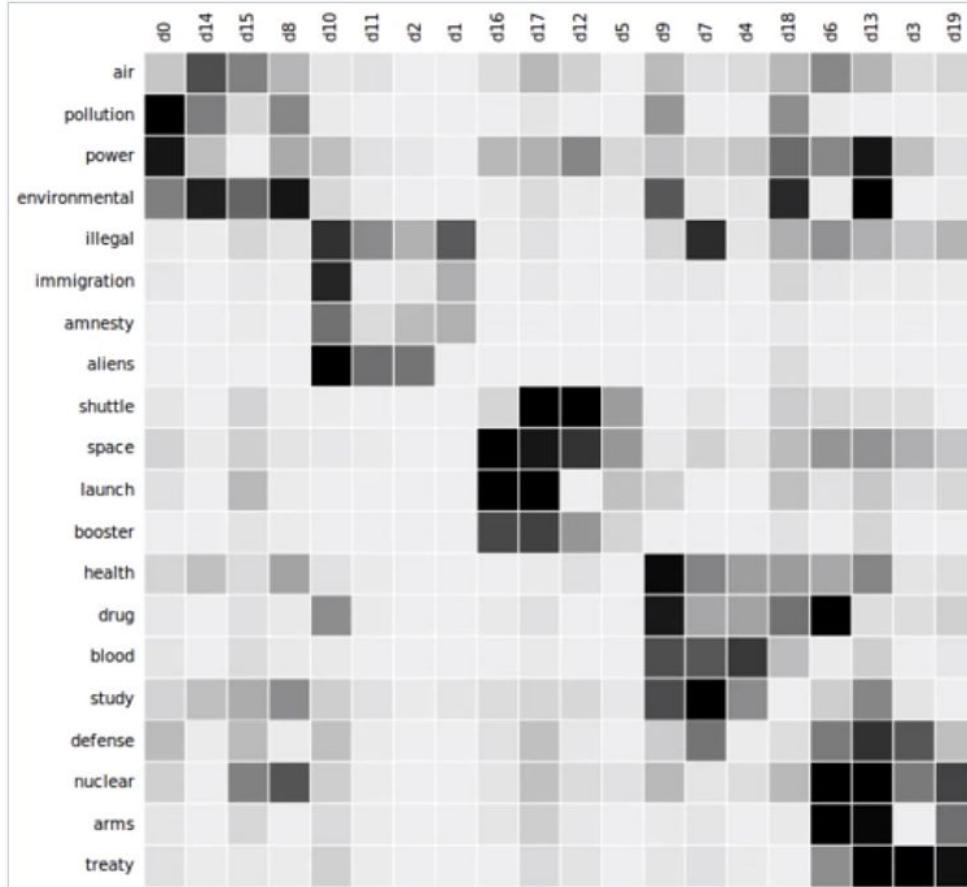
Reviewed in India on 26 December 2020

Size: 10 Sachets (Low Sugar) | **Verified Purchase**

I have been old customer of this product. But this is worst in taste. I dont know whether it is original or not? or company has deteriorated the quality badly.

One person found this helpful

# Topic modeling



Document-Word Matrix

# NLP Tasks challenge

## NLP Tasks challenge

- Variable length sequence

The food is great

Jaipur city is famous as a pink city

- variable length input
- Long term dependencies
- Difference in sequence orders.

# NLP Tasks challenge

- Variable length sequence

The food is great

Jaipur city is famous as a pink city

- Long-term dependency

France is where he grew up, but now he is in India. He speaks fluent \_\_\_\_\_.

# NLP Tasks challenge

- Variable length sequence

The food is great

Jaipur city is famous as a pink city

- Long-term dependency

France is where he grew up, but now he is in India. He speaks fluent \_\_\_\_\_. French

# NLP Tasks challenge

- Variable length sequence

The food is great

Jaipur city is famous as a pink city

- Long-term dependency

France is where he grew up, but now he is in India. He speaks fluent \_\_\_\_\_. French

- Differences in sequence order.

The food was good, not bad at all

The food was bad, not good at all.

- Variable input
  - long term dependency
  - Differences in seq. order .

# NLP Tasks challenge

- Variable length sequence

The food is great

Jaipur city is famous as a pink city

- Long-term dependency

France is where he grew up, but now he is in India. He speaks fluent \_\_\_\_\_. French

- Different seen in sequence order.

The food was good, not bad at all

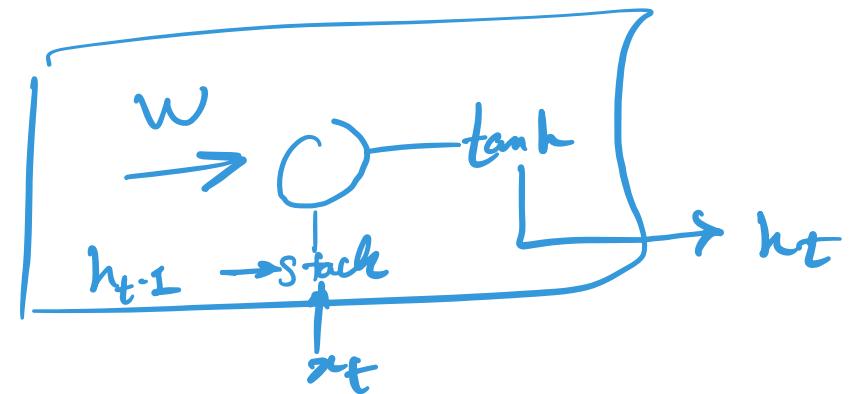
The food was bad, not good at all.

We can model NLP Tasks as Sequence to Sequence Problem

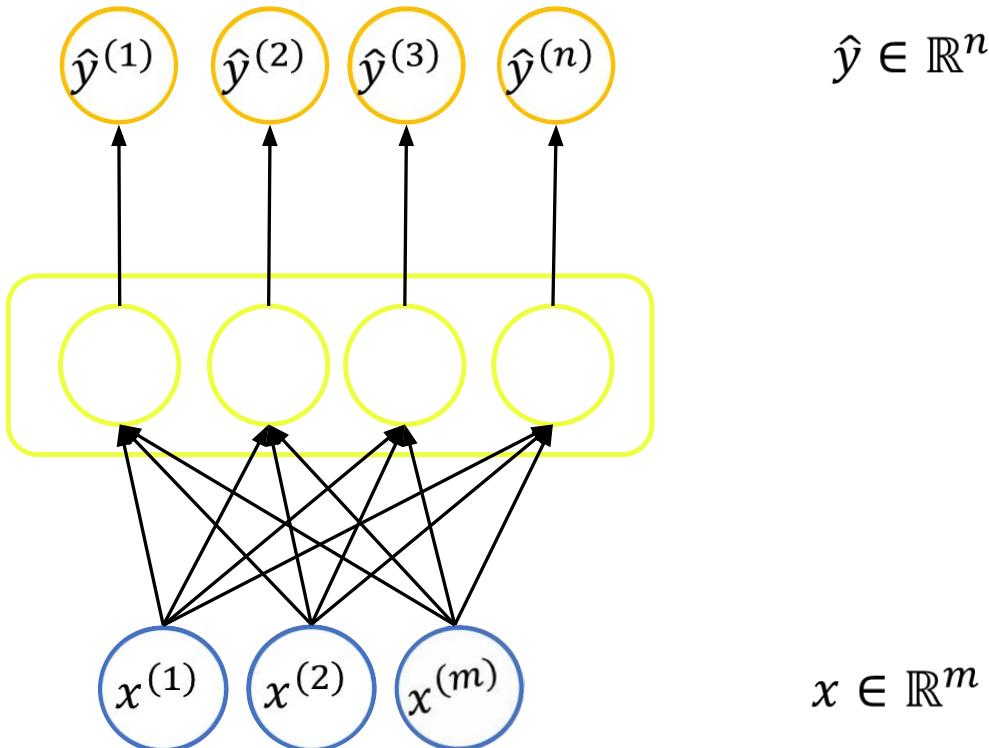
seq 2 seq

① RNN

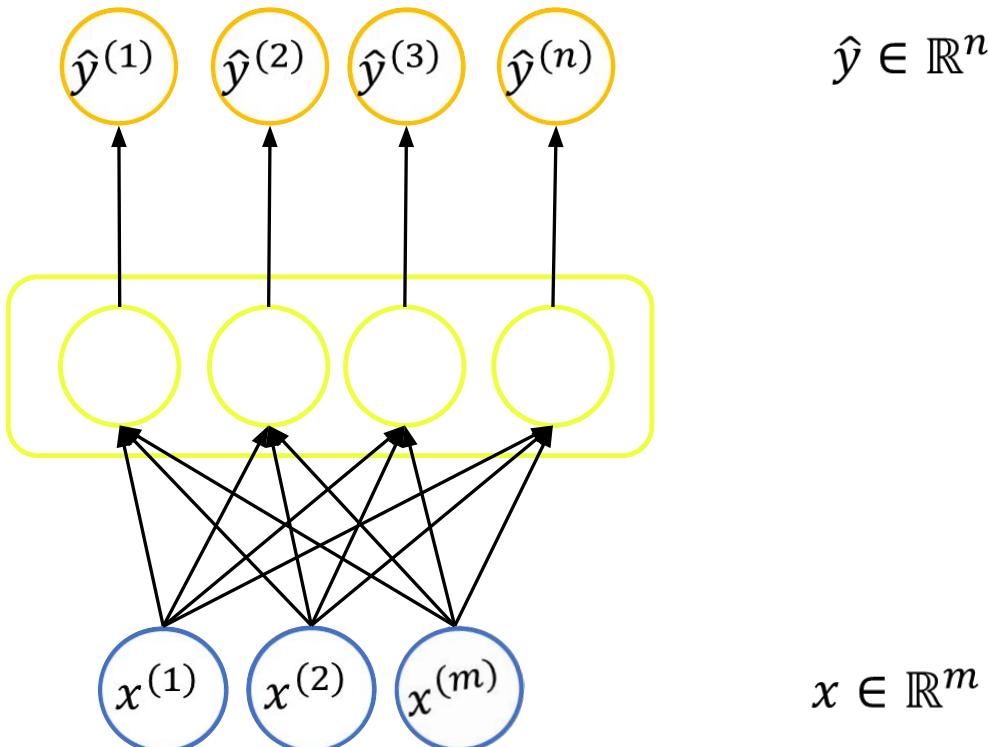
# Sequence to Sequence Learning with Neural Networks



# Feed-Forward Neural Network

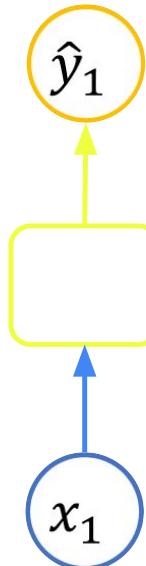


# Feed-Forward Neural Network



Difficult to model speech recognition,  
question answering, and machine  
translation (Sequence to Sequence) etc.

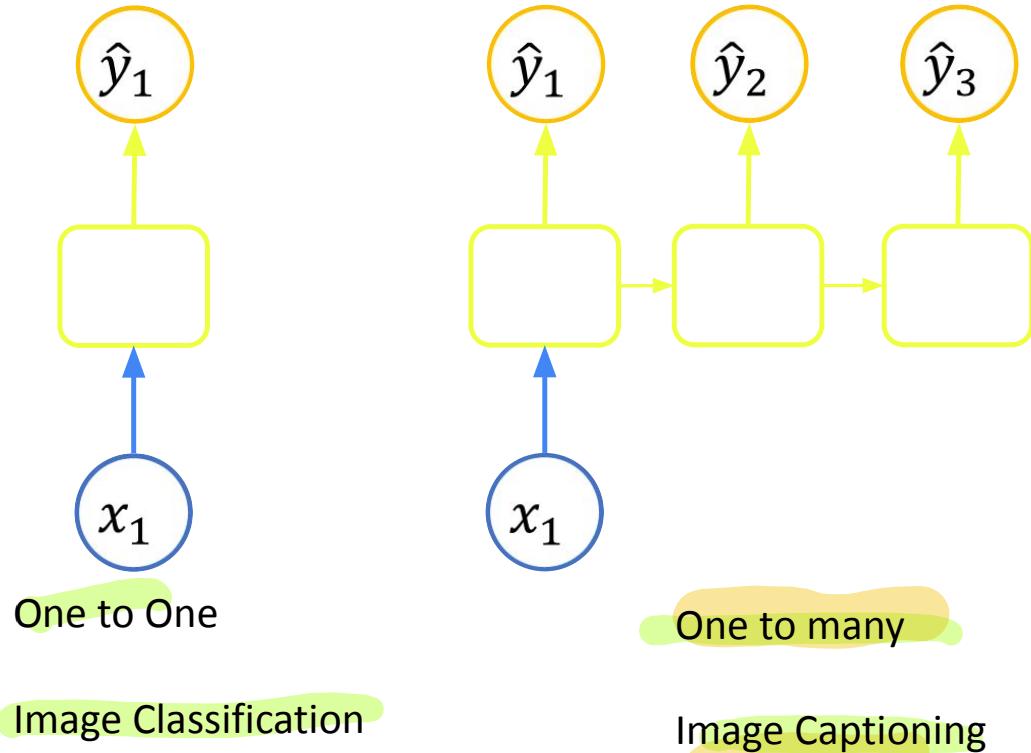
# Sequence to Sequence Problem



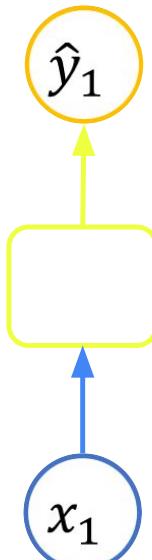
One to One

Image Classification

# Sequence to Sequence Problem

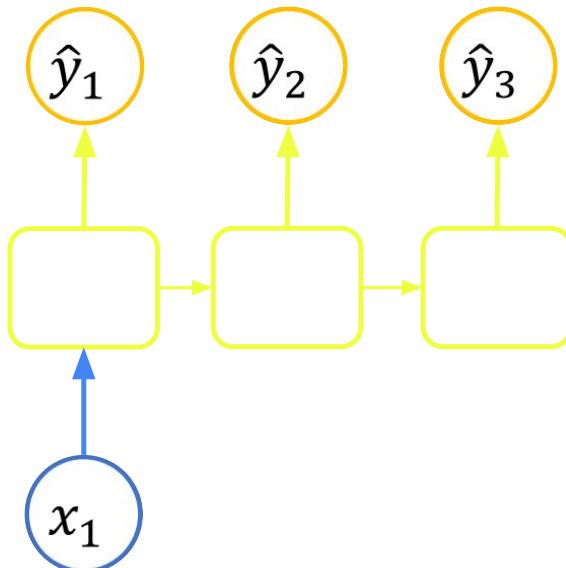


# Sequence to Sequence Problem



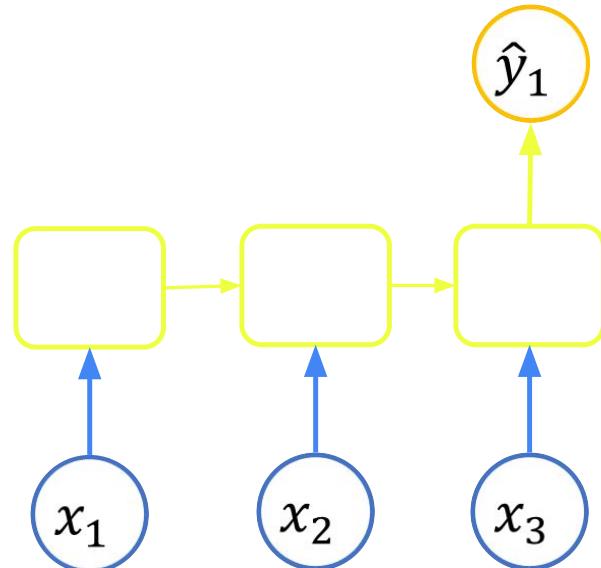
One to One

Image Classification



One to many

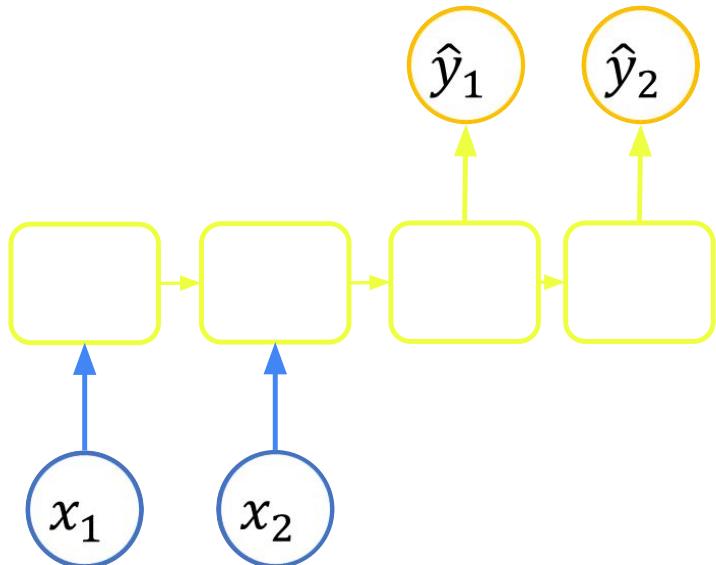
Image Captioning



many to one

Language Recognition

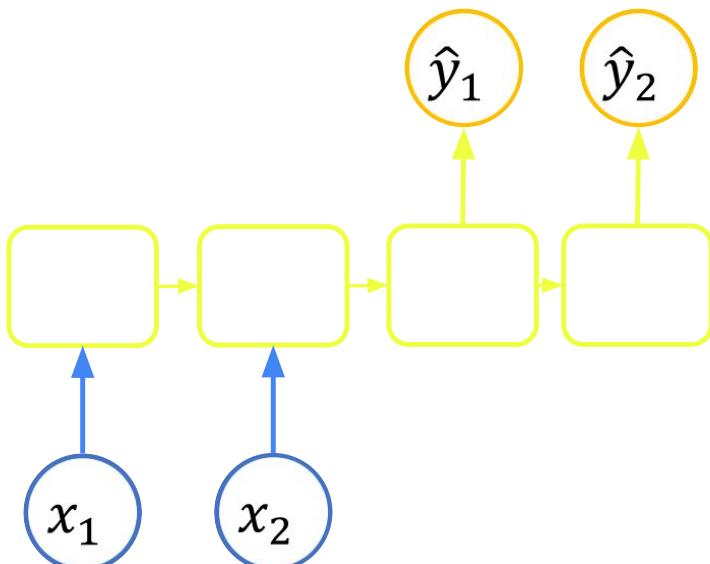
# Sequence to Sequence Problem



Many to Many

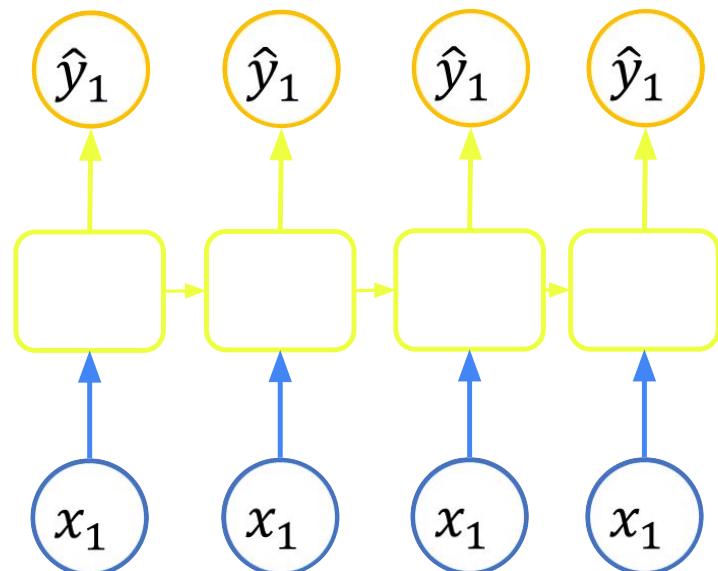
Machine Translation

# Sequence to Sequence Problem



Many to Many

Machine Translation



Many to many

Video Activity Recognition

# seq2seq Learning

- Seq2seq models are neural network architecture used to transform input sequences into output sequences of **variable length**.

# seq2seq Learning

- Seq2seq models are neural network architecture used to transform input sequences into output sequences of **variable length**.
- Seq2seq does the sequence transformation using a simple RNN or using LSTM or GRU (to avoid problem of vanishing gradients)

# Basics of RNN

RNN

- process seq. data
- seq. input of variable length
- Apply same weights

Recurrent Neural Network is a family of neural networks that:

- Process sequence data
- Take sequential **input** of variable length
- Apply the same weights on each step
- Can produce **output** of variable length

Apply some  
weights  $w$ .

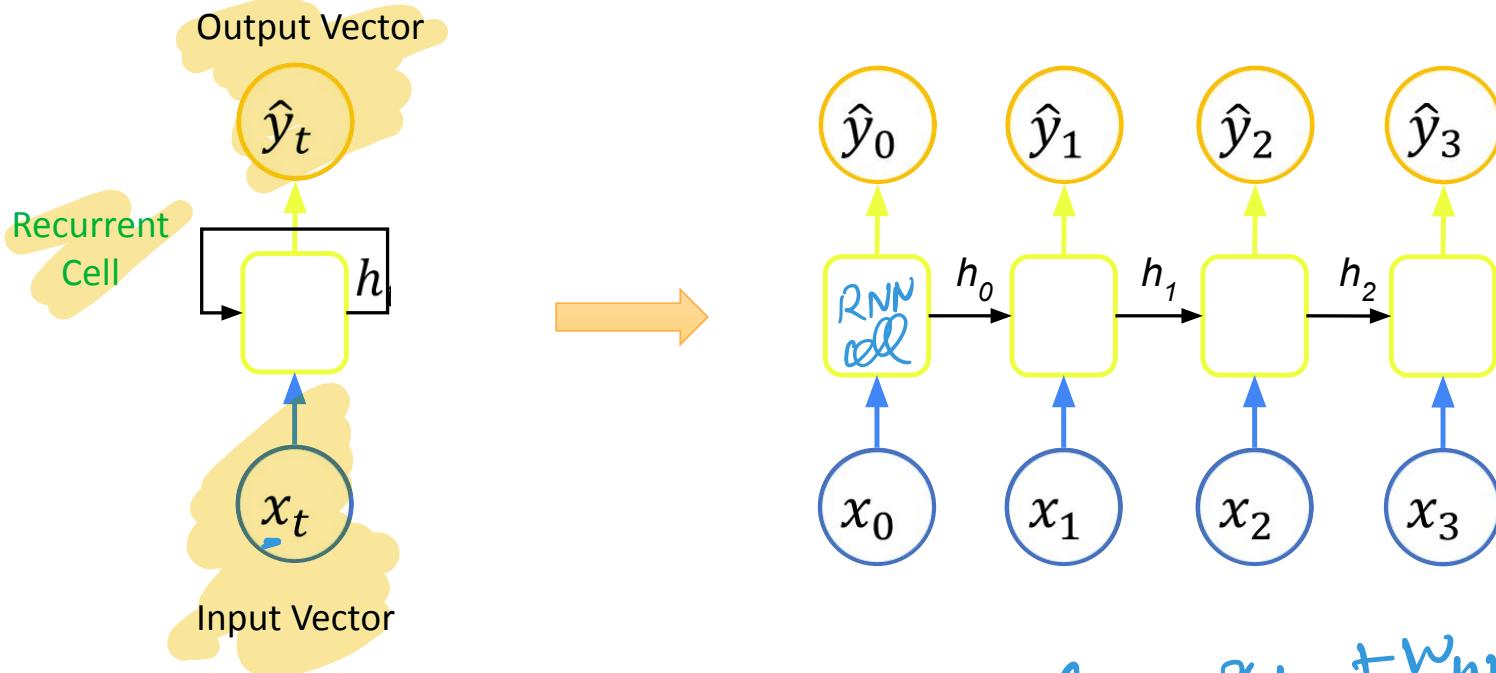
• Apply some weights

Output Vector



Input Vector

# Recurrent Neural Networks



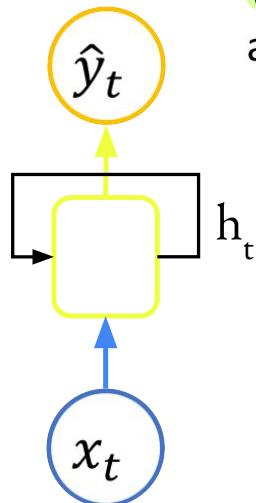
$$h_t = \tanh(w_{nx} \underline{x_t} + w_{nh} \underline{h_{t-1}} + b_n)$$

$$\hat{y} = \underline{w_{ny} h_t + b_{ny}}$$

## Recurrent Neural Networks

$$h_t = f_w(h_{t-1}, x_t)$$

Output



We can process a sequence of vectors  $x$  by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

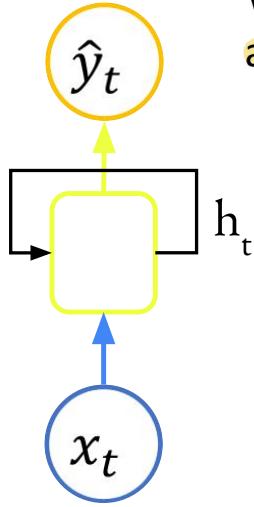
new state      old state      input vector at some time step  
some function with parameters W      recurrence formula.

Input

$$h_t = \tanh(w_{xn} x_t + w_{hn} h_{t-1} + b_{hn} + b_{xn})$$

# Recurrent Neural Networks

Output



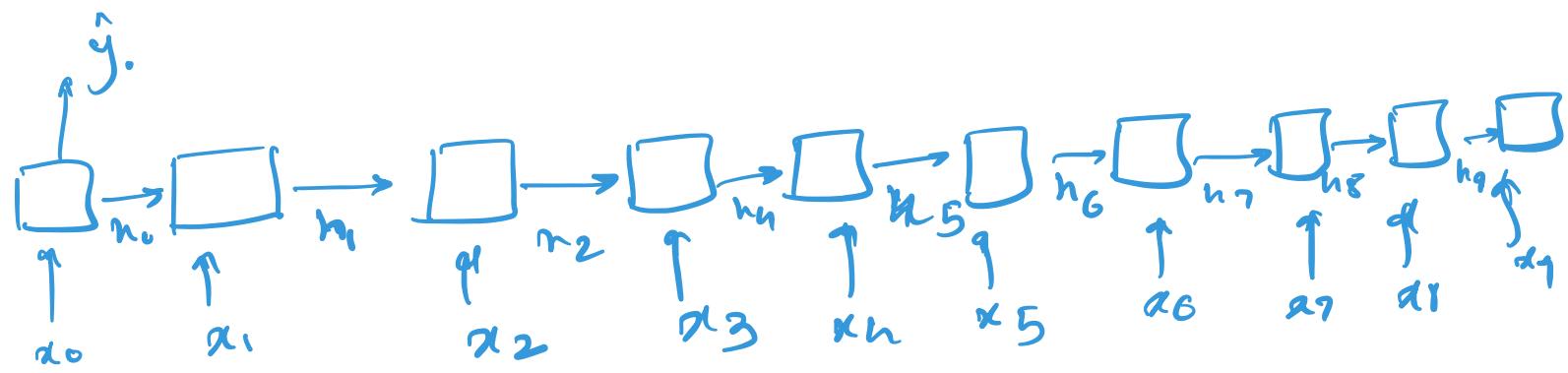
We can process a sequence of vectors  $x$  by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state                      old state              input vector at some time step  
some function with parameters W

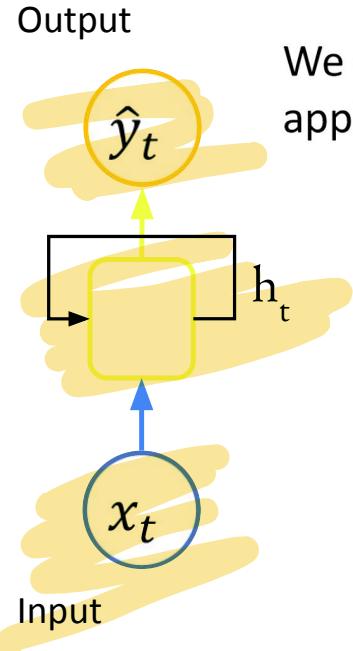
Input

Notice: the same function and the same set of parameters are used at every time step.



# Recurrent Neural Networks

Notice: the same function and the same set of parameters are used at every time step.

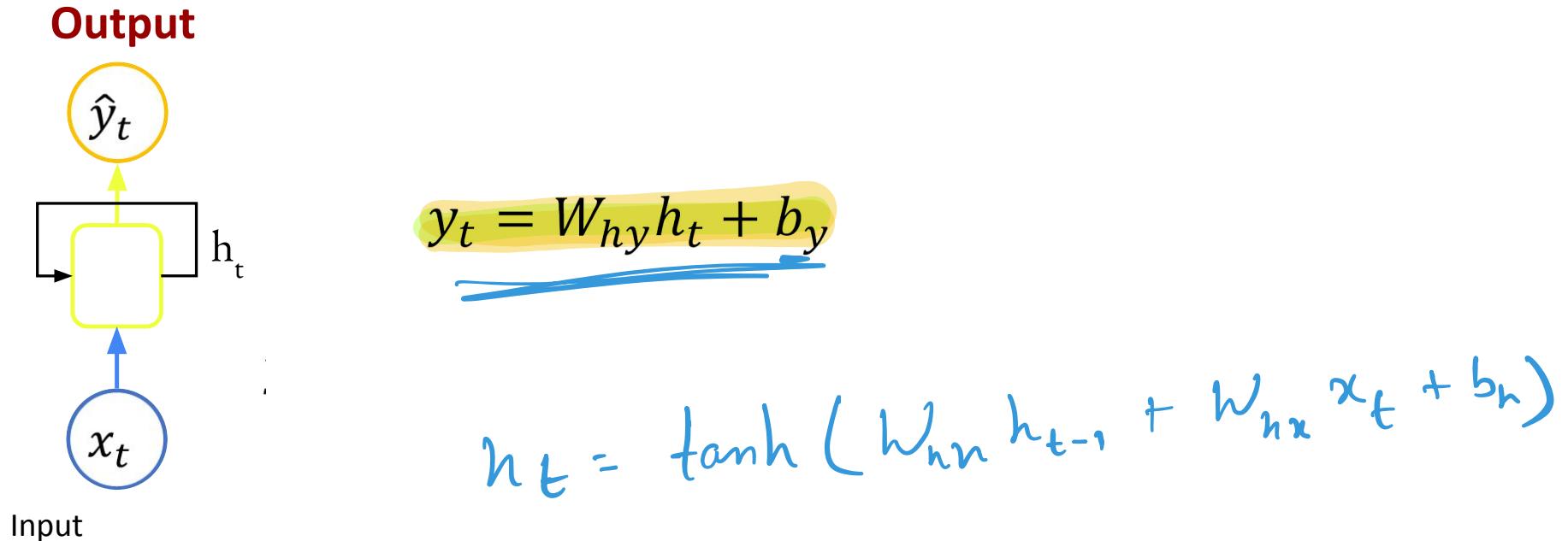


We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

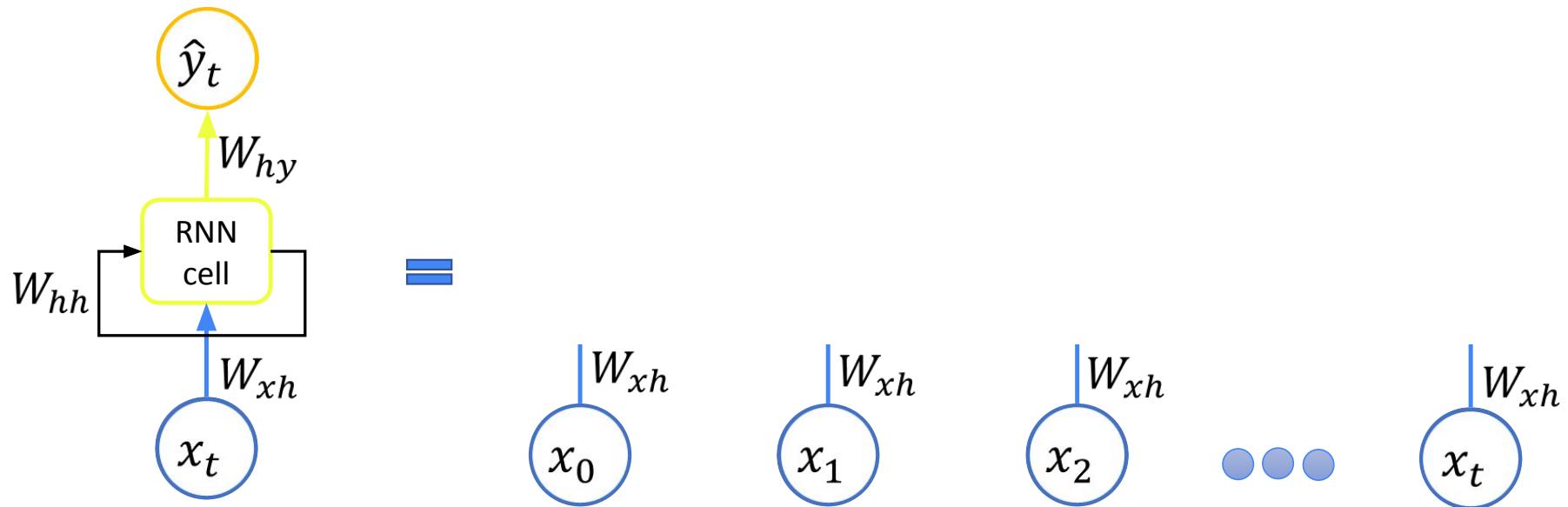
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

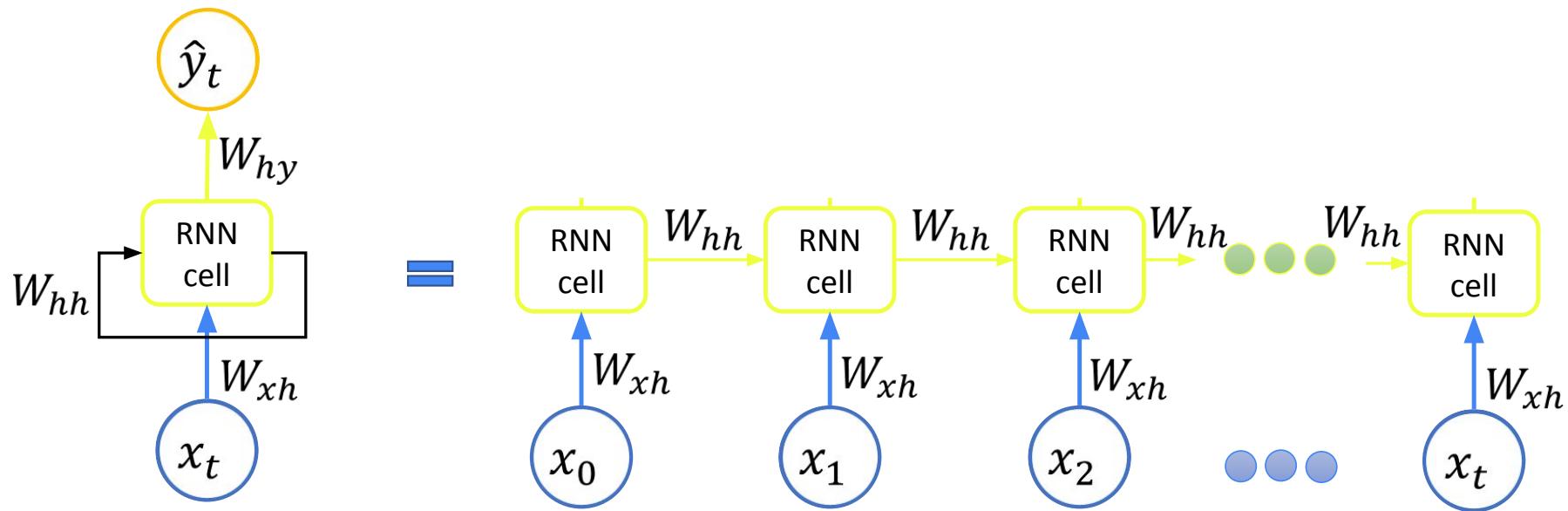
# Recurrent Neural Networks



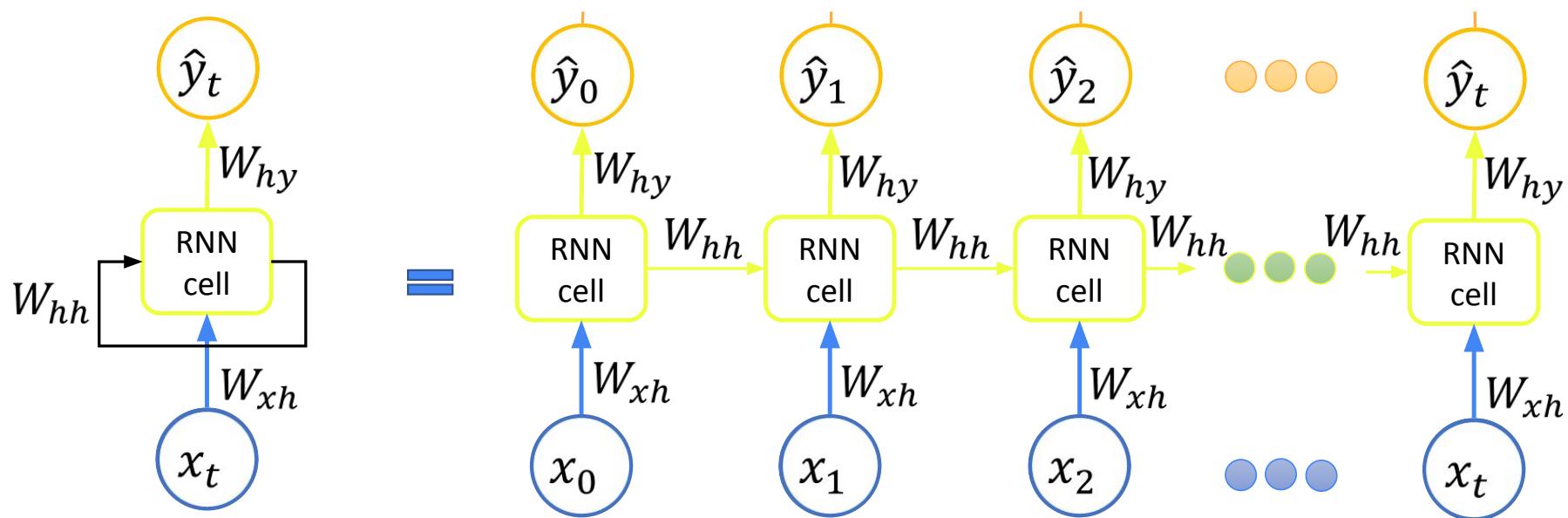
# RNN Parameters



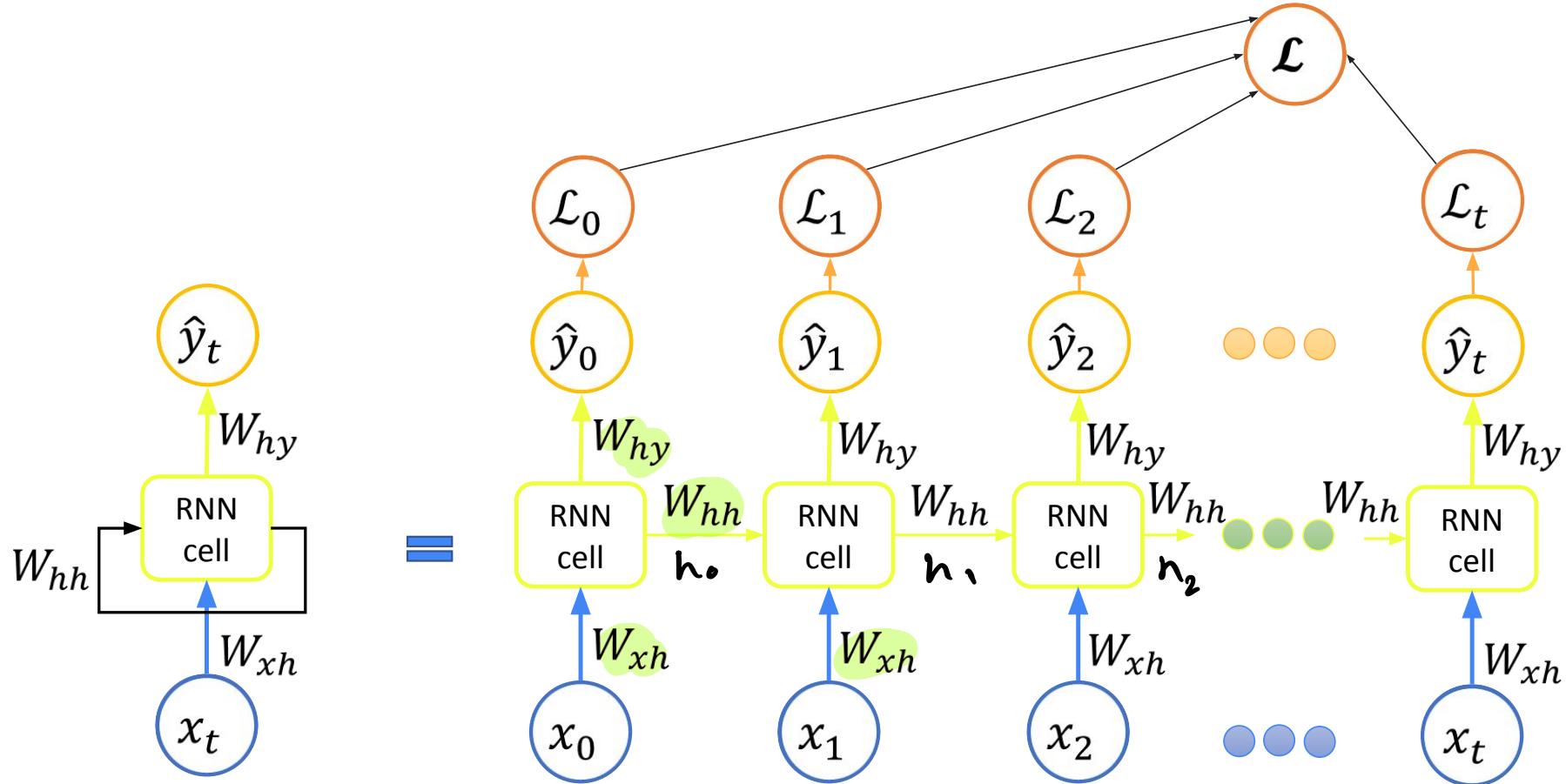
# RNN Parameters



# RNN Parameters



# RNN Computational Graph Across Time

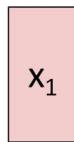
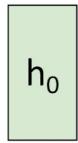


# RNN Computational Graph

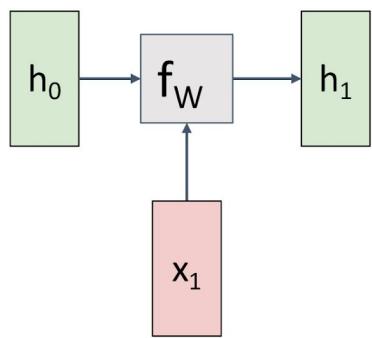
# RNN Computational Graph

Initial hidden state

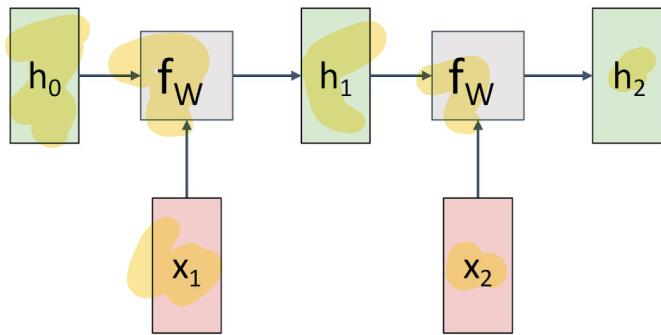
Either set to all 0,  
Or learn it



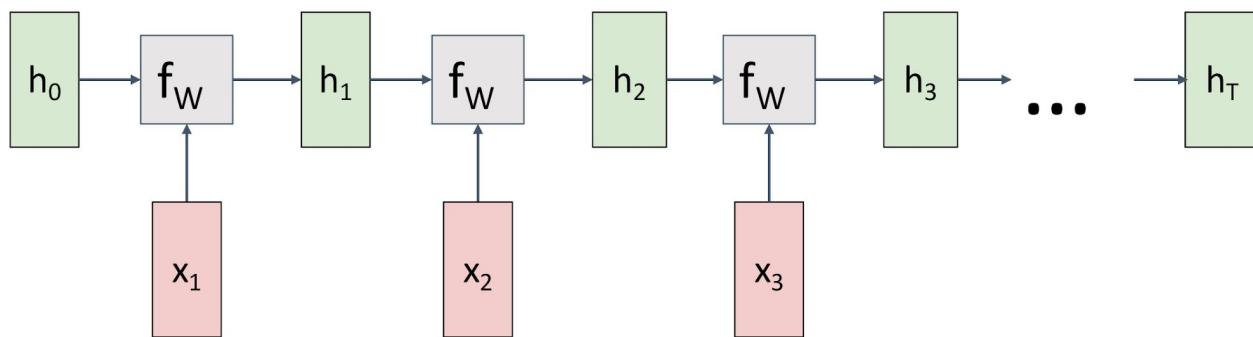
# RNN Computational Graph



# RNN Computational Graph

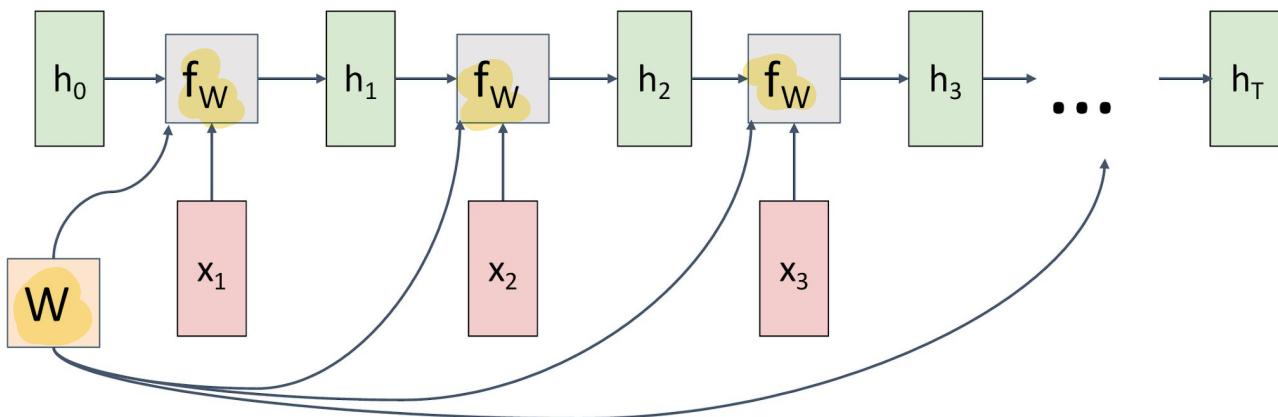


# RNN Computational Graph

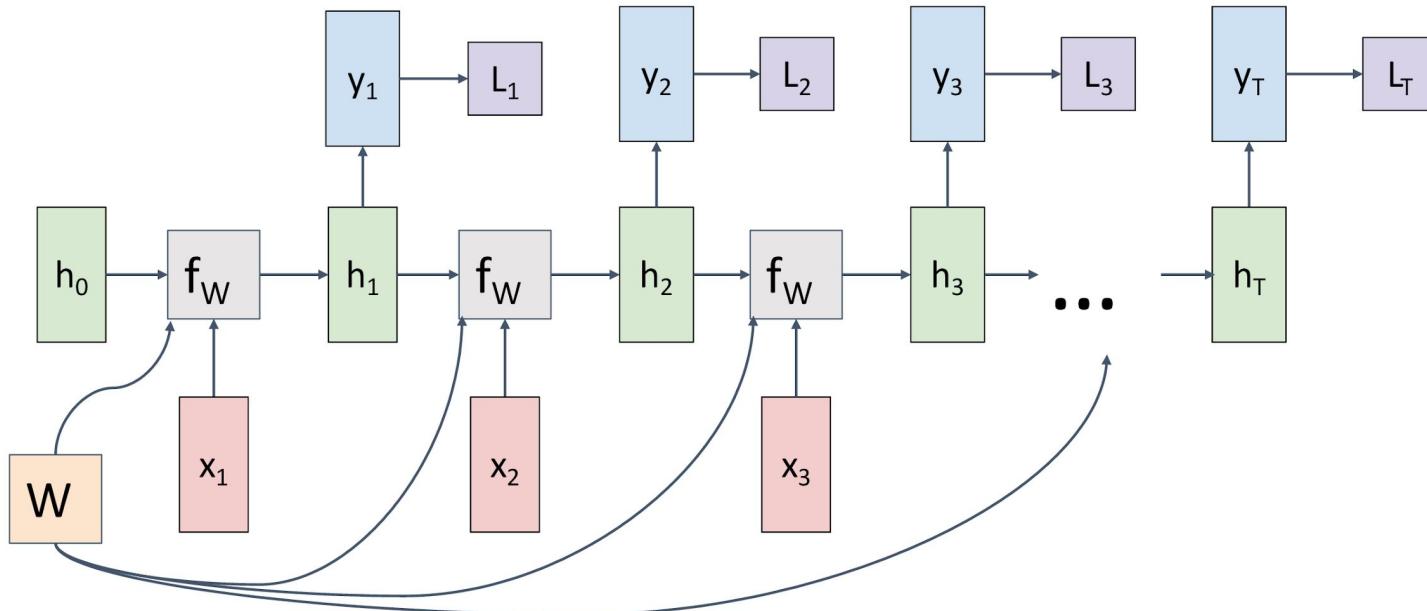


# RNN Computational Graph

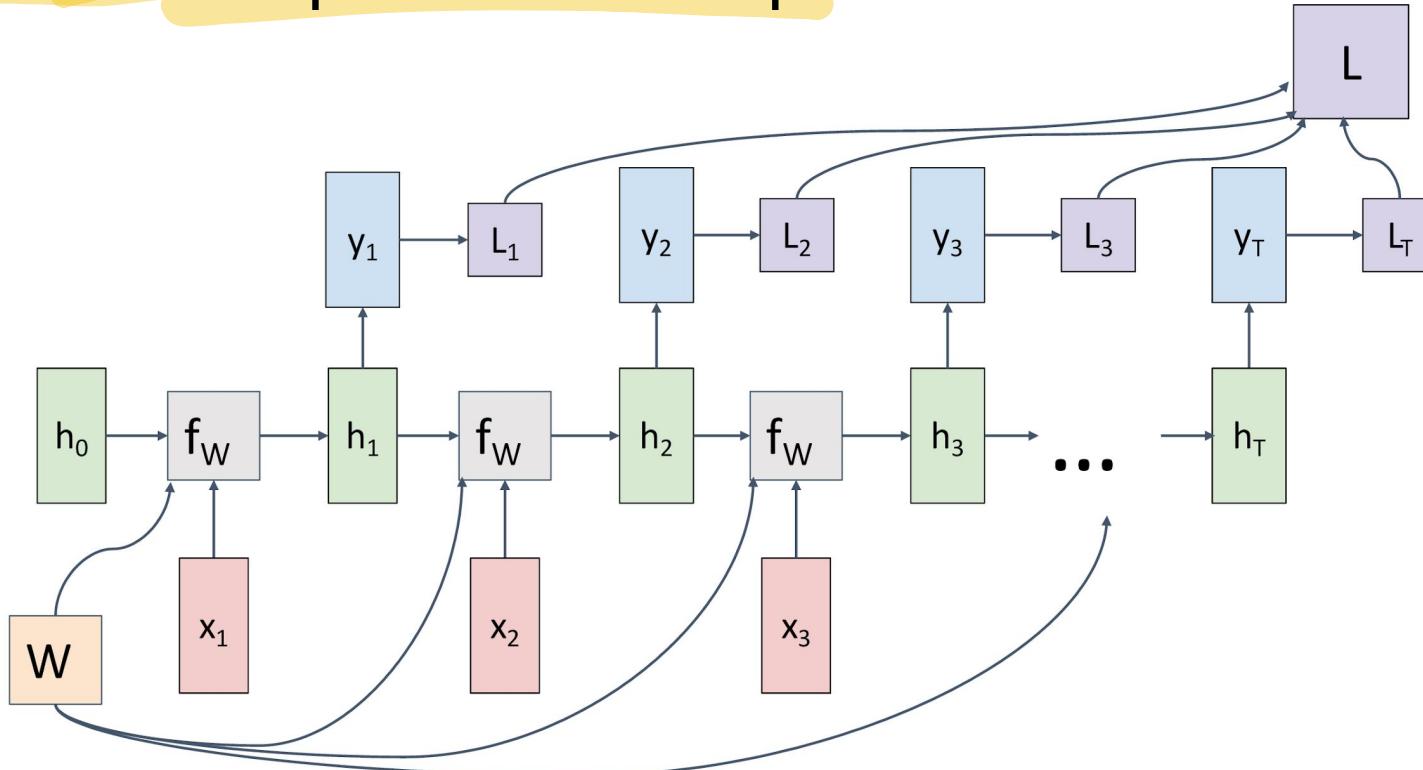
Re-use the same weight matrix at every time-step



# RNN Computational Graph

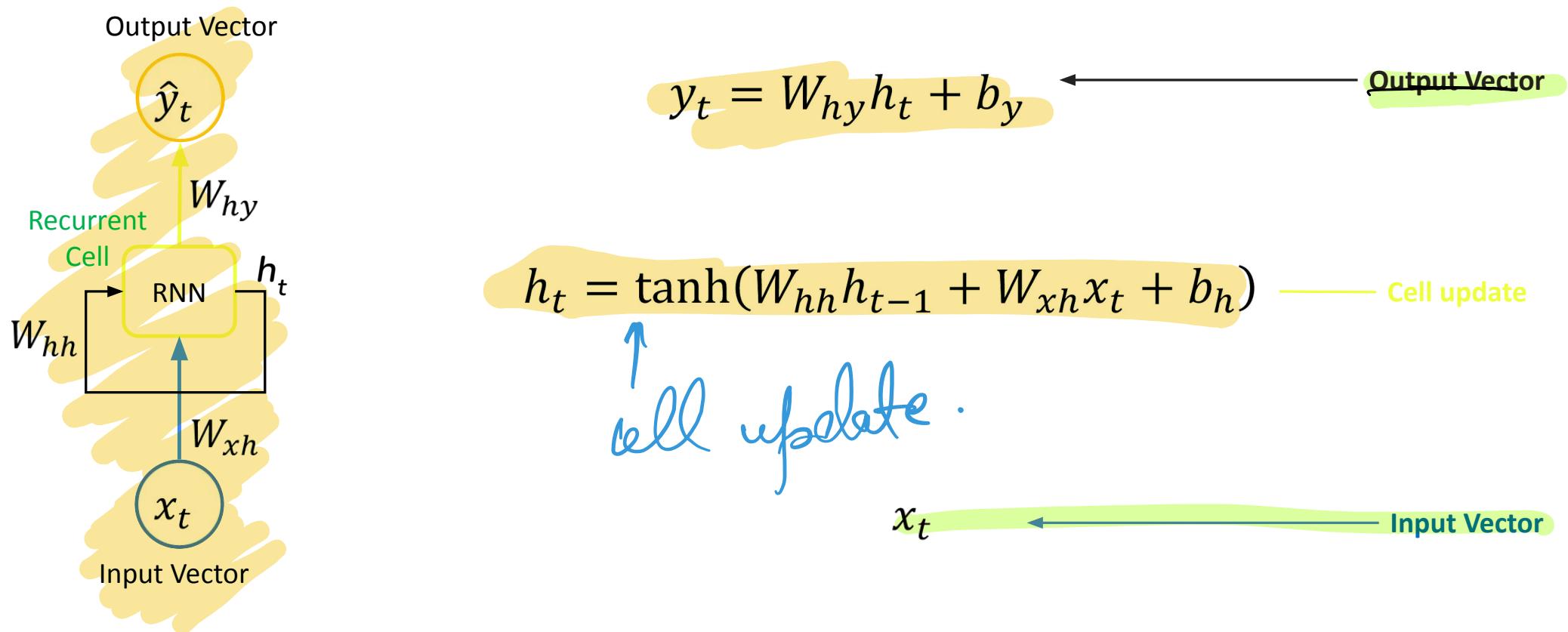


# RNN Computational Graph

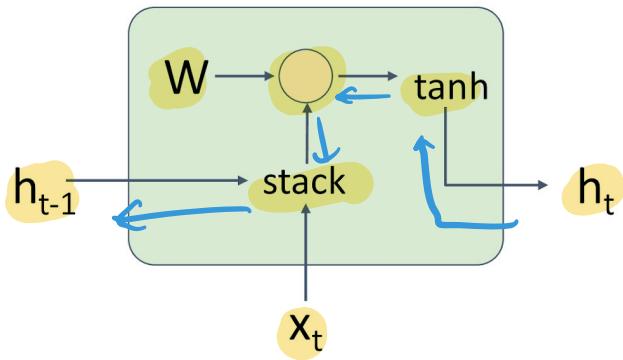
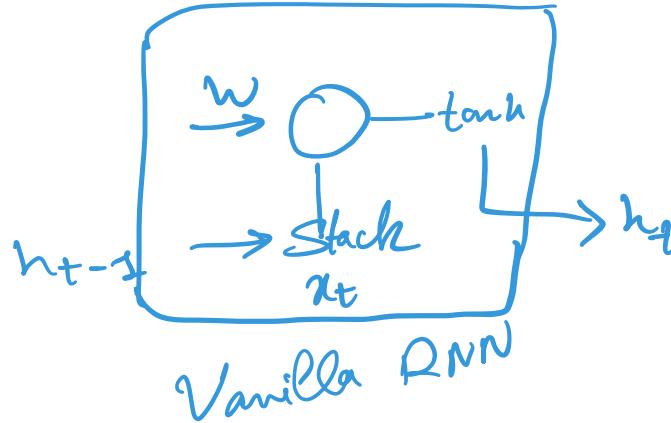


# Training RNNs

# RNN State Update

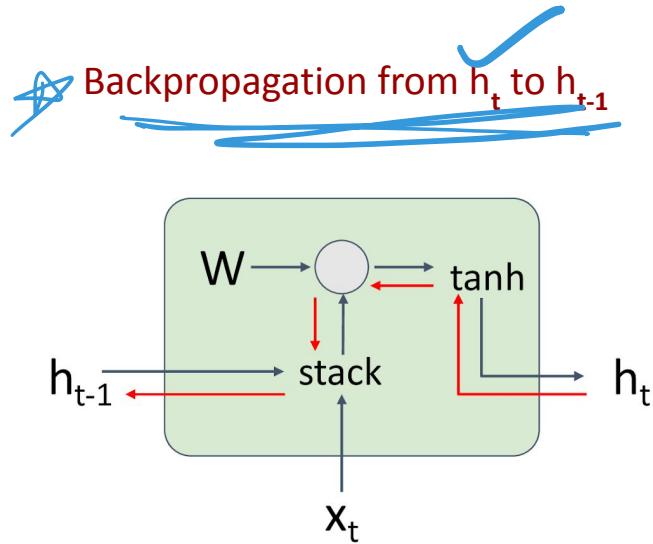


# Vanilla RNN



$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\ &= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right) \\ &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right) \end{aligned}$$

# Gradient Flow



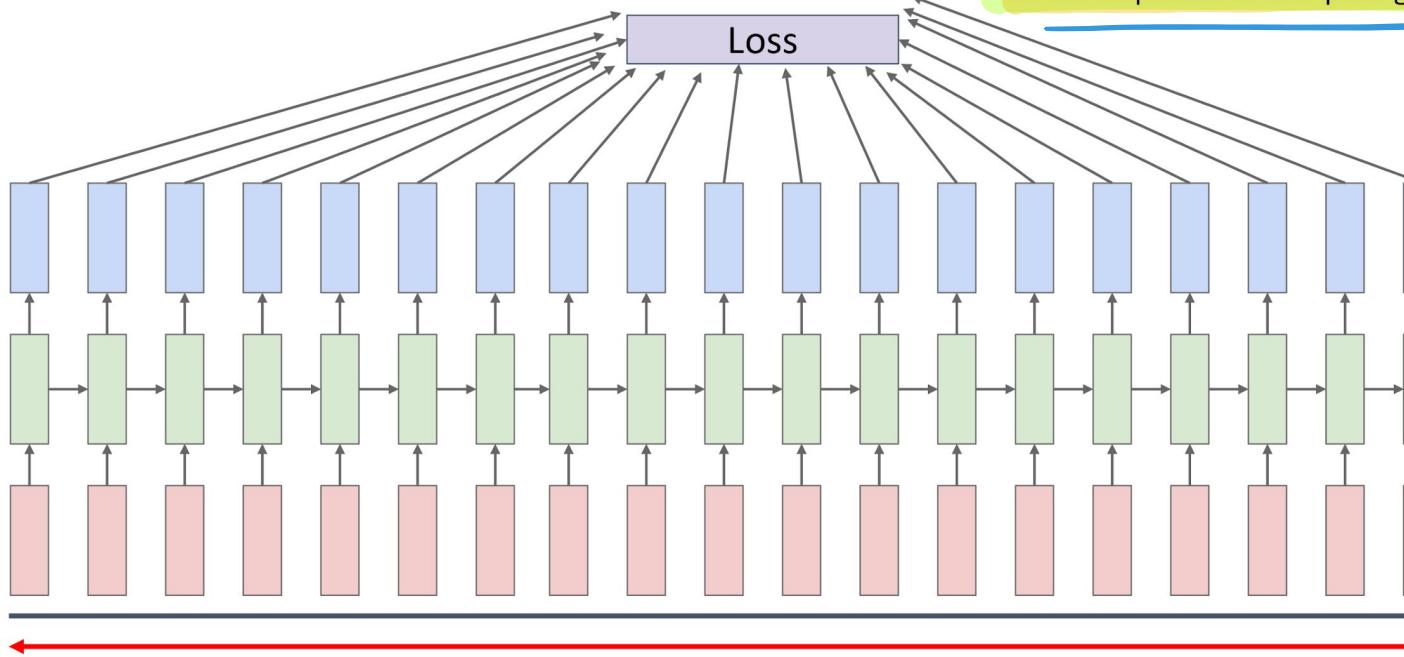
$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\ &= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right) \\ &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right) \end{aligned}$$

# Backpropagation steps

- Input feedforward in the network
- Compute the Loss
- Take the derivative of the Loss with respect to each parameter
- Update parameters to minimize the Loss

# Backpropagation Through Time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



# Backpropagation Through Time



Problem: Takes a lot of  
memory for long sequences!

Forward through entire sequence to  
compute loss, then backward through  
entire sequence to compute gradient

