Lecture 17:
Recurrent Neural Networks (RNN)

COURSE: BIOMEDICAL DATA SCIENCE

Parisa Rashidi

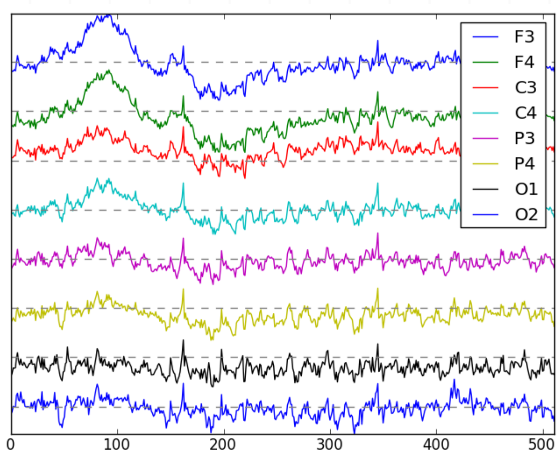
FALL 2019

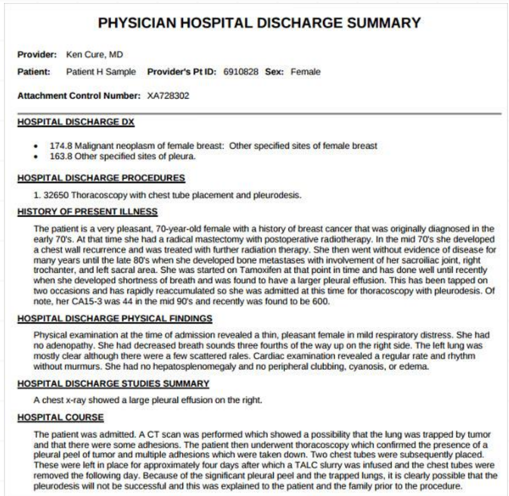
Resources

- The following slides are partially based on
  - CS231n, Stanford, 2018 lecture notes
  - CS6501, Univ. of Virginia, lecture notes
  - www.wildml.com

Applications

- Useful for modelling sequential information.
- Natural language data
  - E.g. clinical notes, each sentence is an ordered set of words.
- Time Series
  - Many datasets consist of time series information
    - Physiological signals, sensor data over time, EEG, ECG, …





PHYSICIAN HOSPITAL DISCHARGE SUMMARY

Provider: Ken Cure, MD

Patient: Patient H Sample **Provider's Pt ID:** 6910828 **Sex:** Female

Attachment Control Number: XA728302

HOSPITAL DISCHARGE DX

- 174.8 Malignant neoplasm of female breast: Other specified sites of female breast
- 163.8 Other specified sites of pleura.

HOSPITAL DISCHARGE PROCEDURES

1. 32650 Thoracoscopy with chest tube placement and pleurodesis.

HISTORY OF PRESENT ILLNESS

The patient is a very pleasant, 70-year-old female with a history of breast cancer that was originally diagnosed in the early 70's. At that time she had a radical mastectomy with postoperative radiotherapy. In the mid 70's she developed a chest wall recurrence and was treated with further radiation therapy. She then went without evidence of disease for many years until the late 80's when she developed bone metastases with involvement of her sacroiliac joint, right trochanter, and left sacral area. She was started on Tamoxifen at that point in time and has done well until recently when she developed shortness of breath and was found to have a larger pleural effusion. This has been tapped on two occasions and has rapidly reaccumulated so she was admitted at this time for thoracoscopy with pleurodesis. Of note, her CA15-3 was 44 in the mid 90's and recently was found to be 600.

HOSPITAL DISCHARGE PHYSICAL FINDINGS

Physical examination at the time of admission revealed a thin, pleasant female in mild respiratory distress. She had no adenopathy. She had decreased breath sounds three fourths of the way up on the right side. The left lung was mostly clear although there were a few scattered rales. Cardiac examination revealed a regular rate and rhythm without murmurs. She had no hepatosplenomegaly and no peripheral clubbing, cyanosis, or edema.

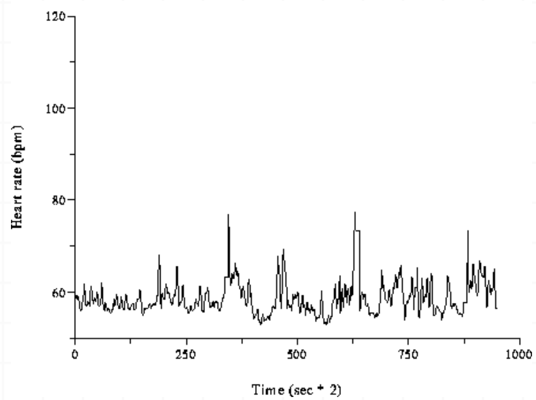
HOSPITAL DISCHARGE STUDIES SUMMARY

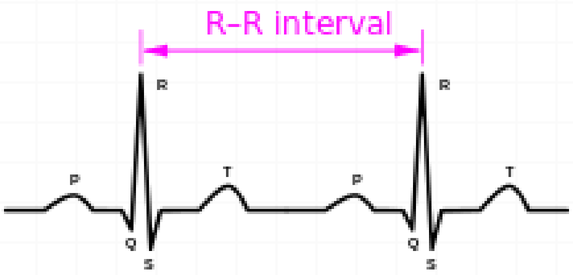
A chest x-ray showed a large pleural effusion on the right.

HOSPITAL COURSE

The patient was admitted. A CT scan was performed which showed a possibility that the lung was trapped by tumor and that there were some adhesions. The patient then underwent thoracoscopy which confirmed the presence of a pleural peel of tumor and multiple adhesions which were taken down. Two chest tubes were subsequently placed. These were left in place for approximately four days after which a TALC slurry was infused and the chest tubes were removed the following day. Because of the significant pleural peel and the trapped lungs, it is clearly possible that the pleurodesis will not be successful and this was explained to the patient and the family prior to the procedure.

Example: Heart Rate

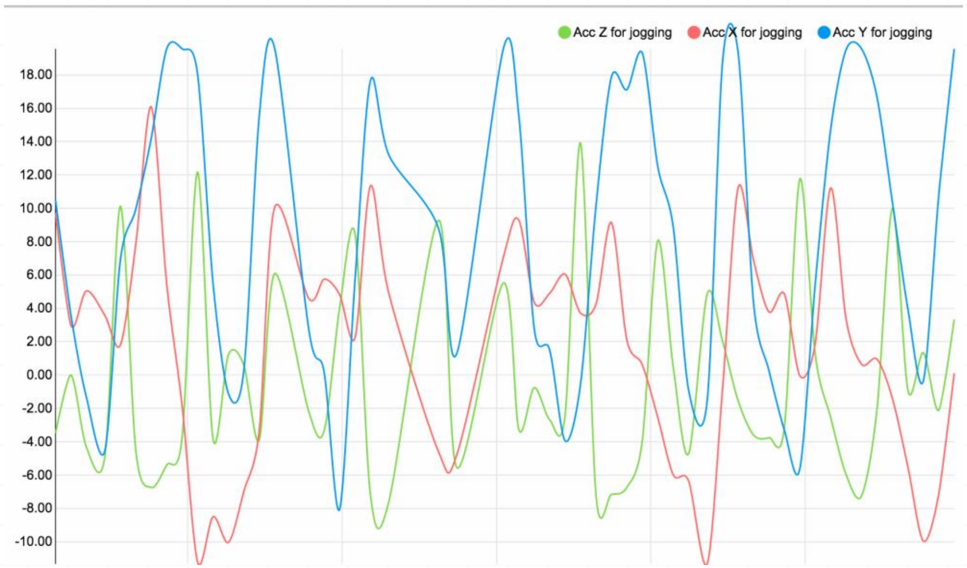




- A time series has sequence of
  - Values and
  - Their corresponding timestamps.

Example: Accelerometer

- Notice the periodic patterns



http://blog.cityzendata.com/2015/03/25/recognize_activity_accelerometer/

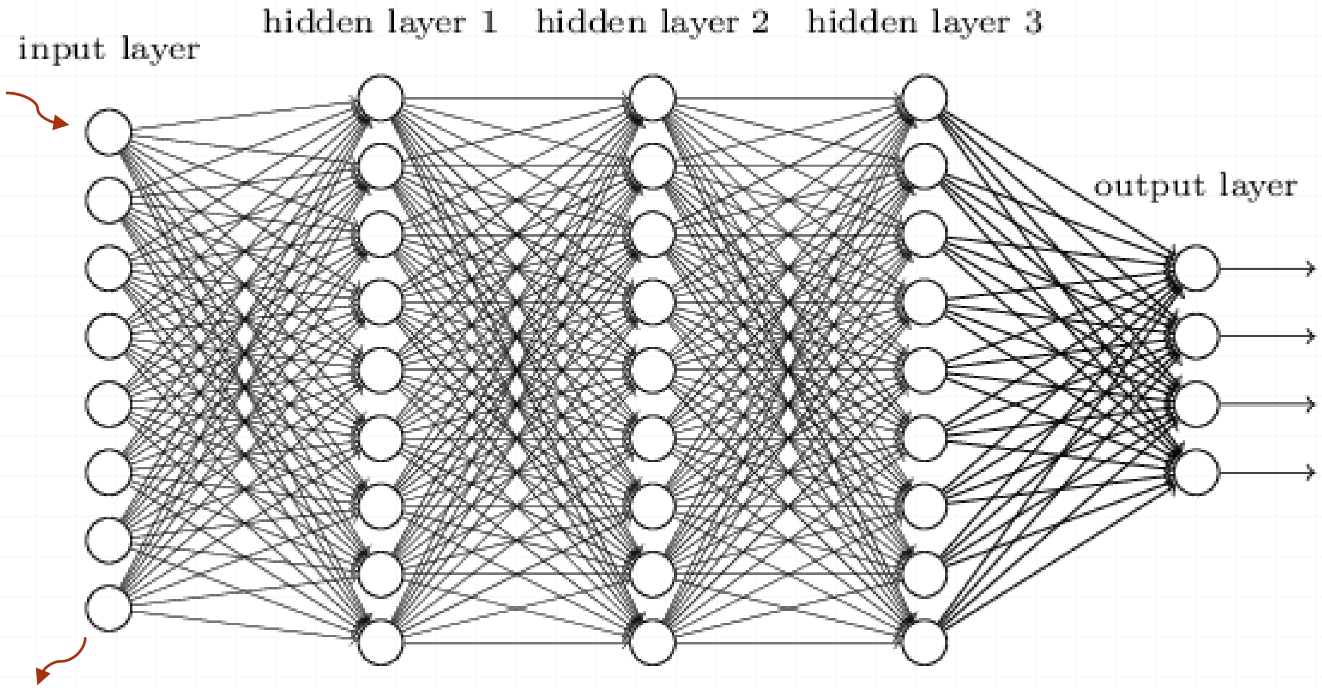
Physiological Datasets

- PhysioBank contains over 90,000 recordings, or over 4 terabytes of digitized physiologic signals and time series, organized in over 80 databases.

Competitions

- Early Prediction of Sepsis from Clinical Data -- the PhysioNet Computing in Cardiology Challenge 2019
  - Given a table of clinical measurements (columns) over time (rows), your entry must report the risk of sepsis (a real number) and a binary sepsis prediction (0 or 1) at each hour of a patient's clinical record using the current and past (but not future) data for the patient.

Sequence Prediction

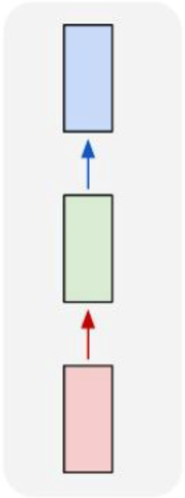


How you could represent sequential/time series data?

Regular Networks

"Vanilla" Neural Network

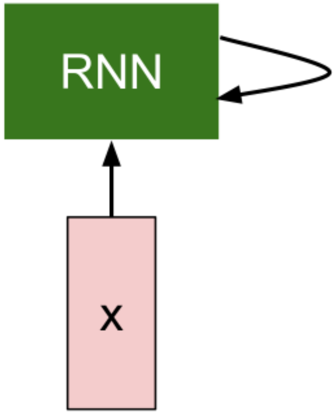
one to one



Vanilla Neural Networks

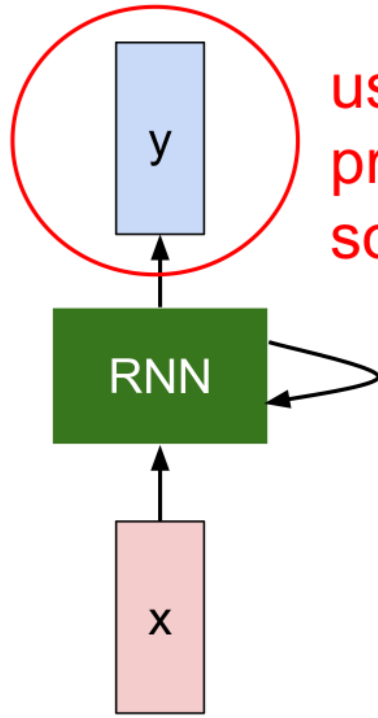
Does it work for sequence data?

Recurrent Neural Network



With sequential data, our model depends on prior data as well.

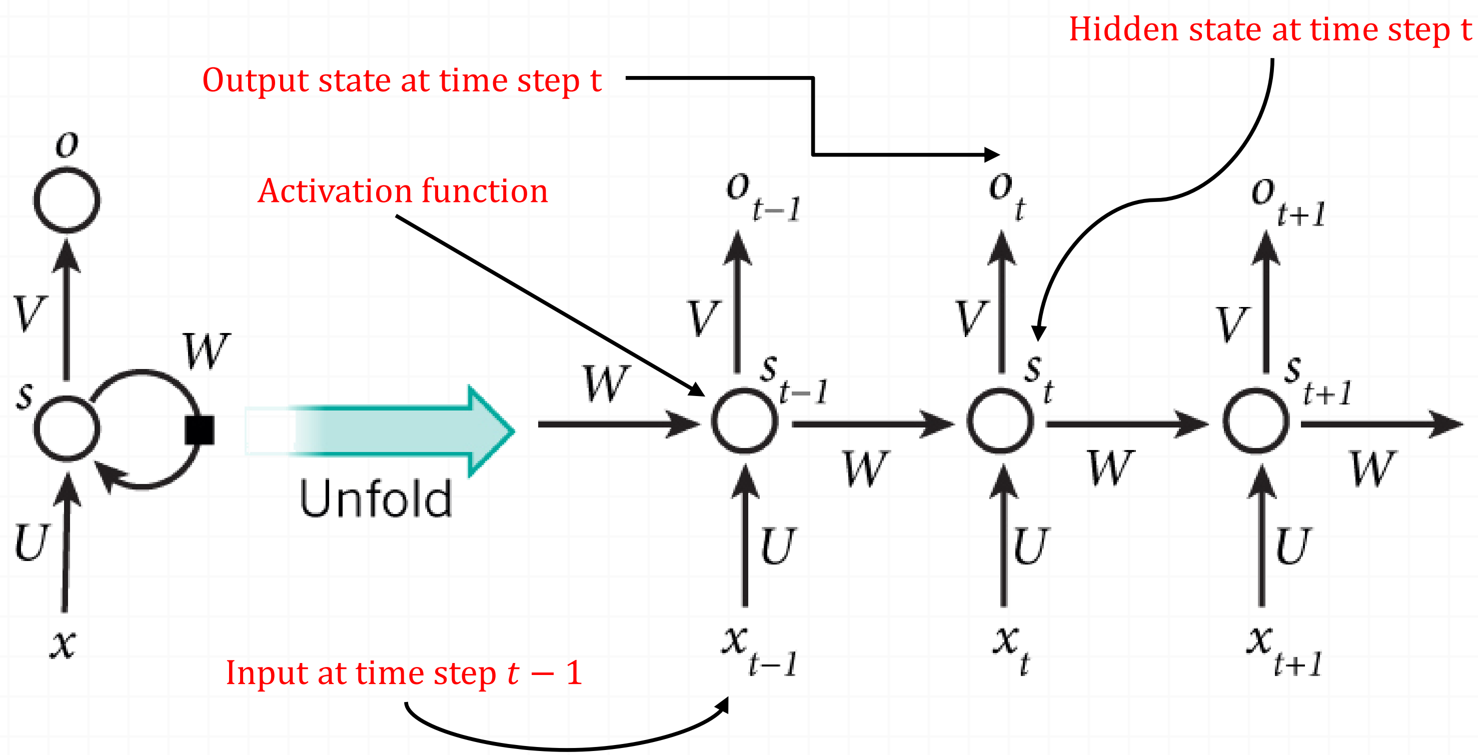
Recurrent Neural Network



usually want to predict a vector at some time steps

e.g. Patient outcome based on physiological signals

Unfolding RNNs

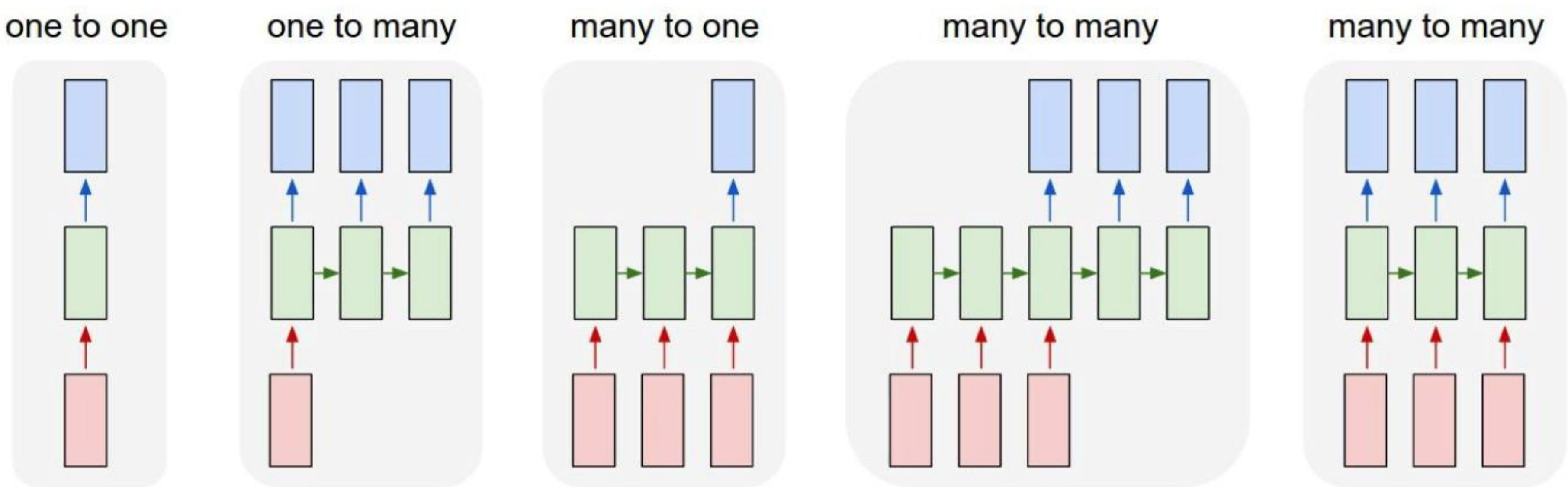


RNN vs. NN

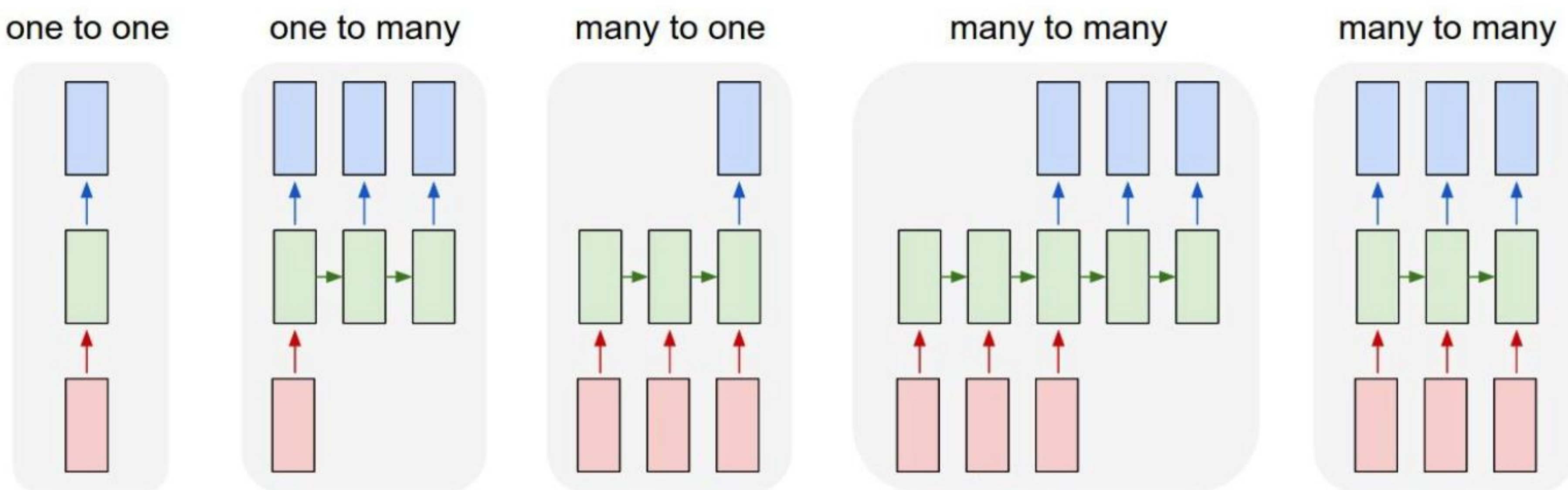
- How RNN is different from neural network?
  - Vanilla neural networks assume all inputs and outputs are independent of each other.
  - But for many tasks, that's a very bad idea.
- What RNN does?
  - Perform the same task for every element of a sequence (that's what recurrent stands for)
  - Output depends on the previous computations!
- Another way of interpretation – RNNs have a "memory"
  - To store previous computations

Think about examples for each case ....

Recurrent Neural Networks: Process Sequences

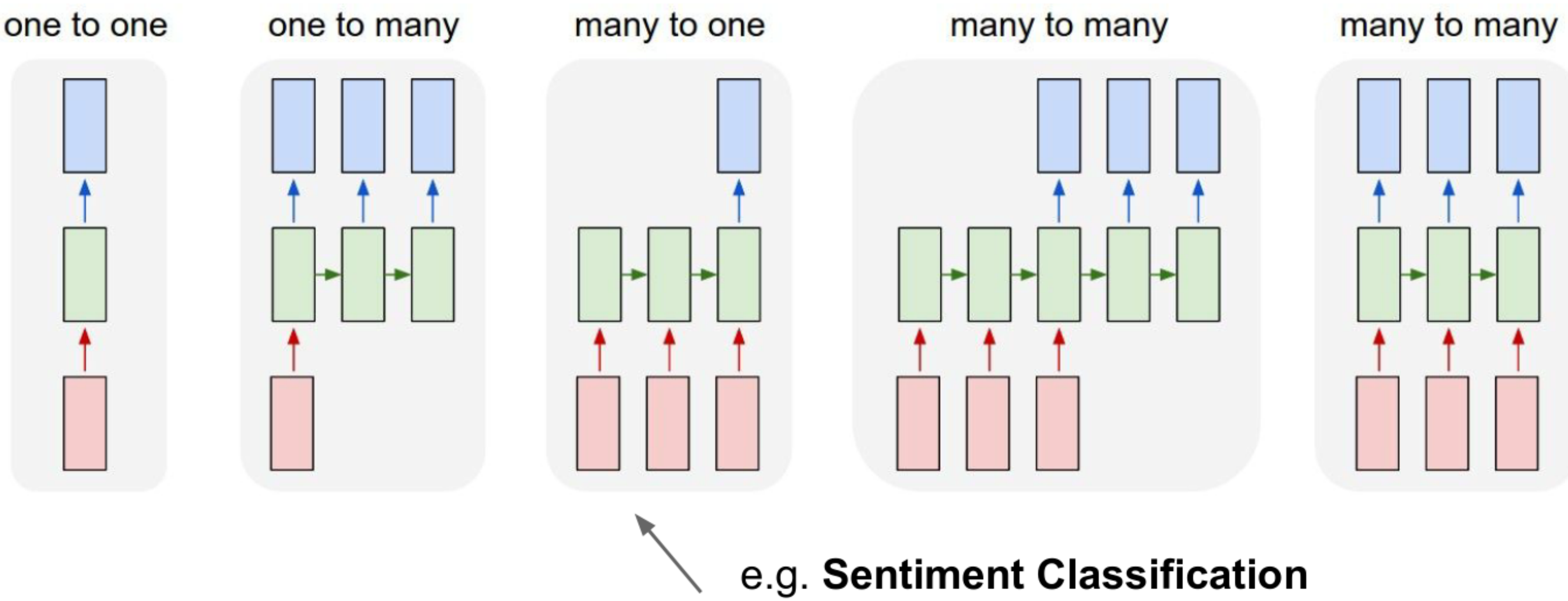


Recurrent Neural Networks: Process Sequences



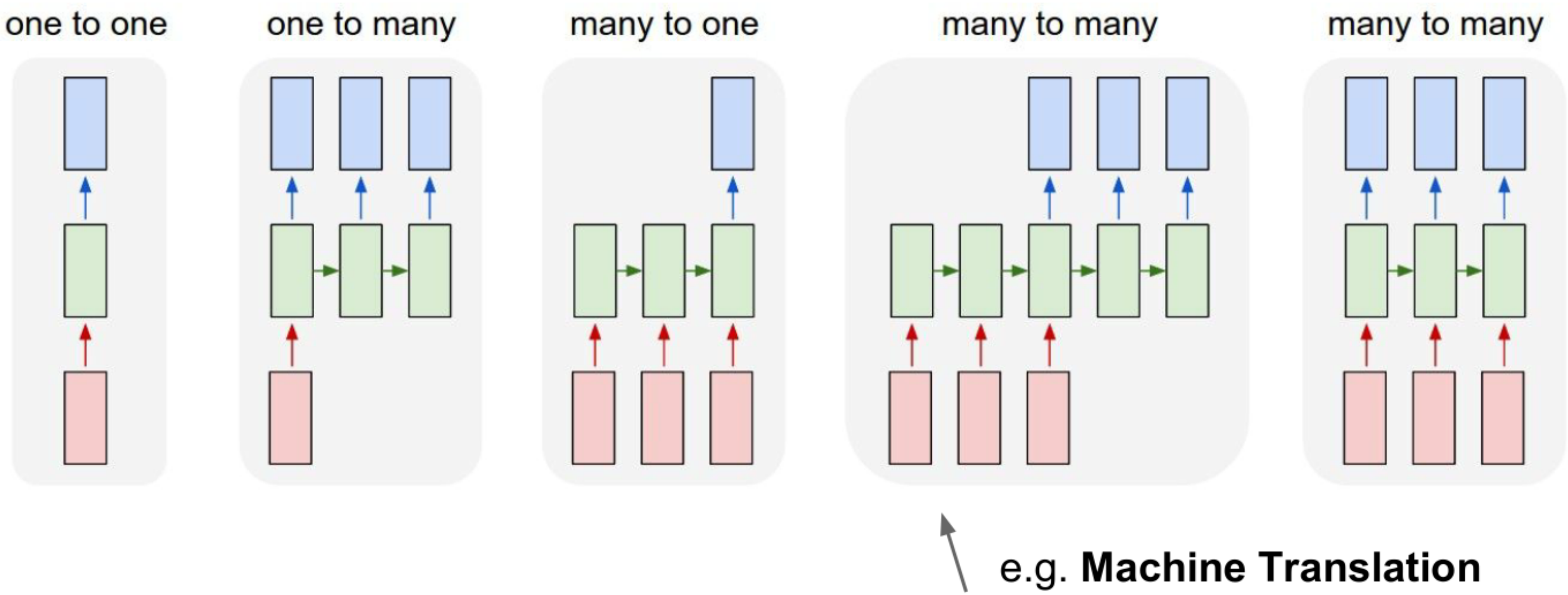
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



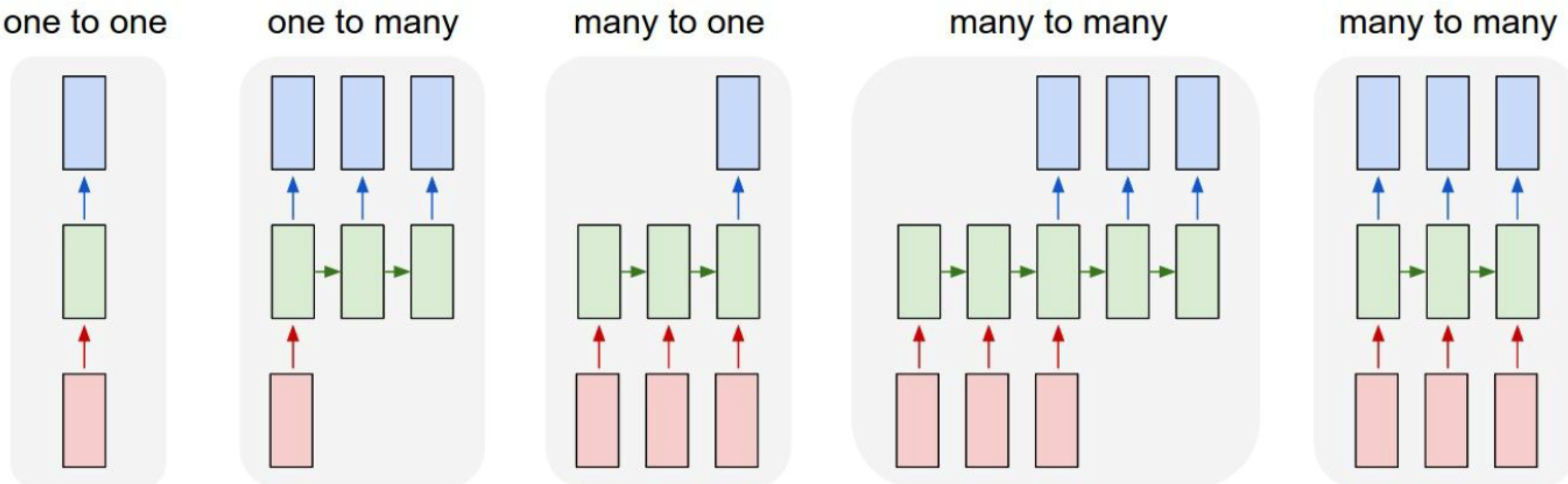
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences



e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks: Process Sequences

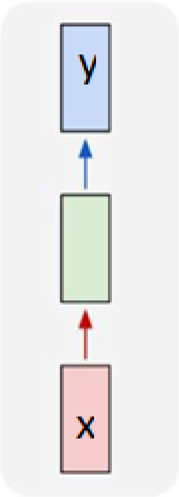


e.g. **Video classification on frame level**

Neural Network

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

$$\boxed{h_t} = \boxed{f_W}(\boxed{x_t})$$



Recurrent Neural Network

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

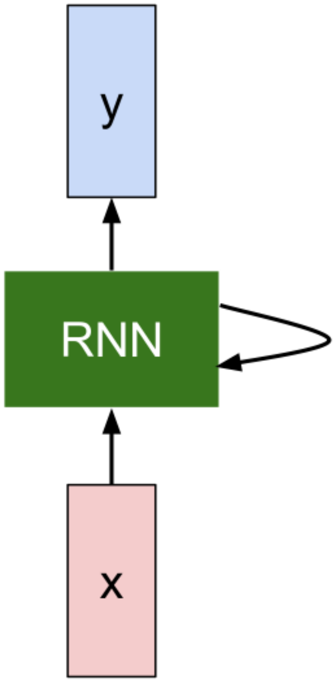
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W

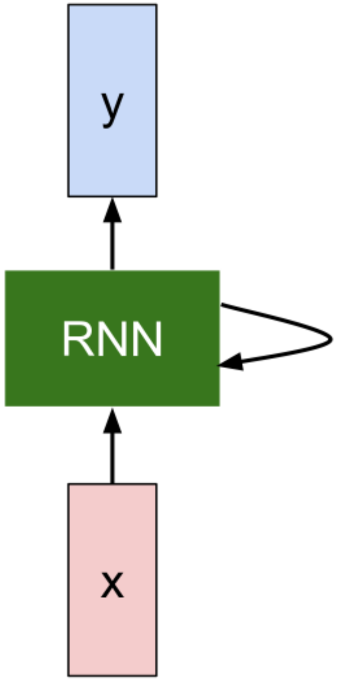


Recurrent Neural Network

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

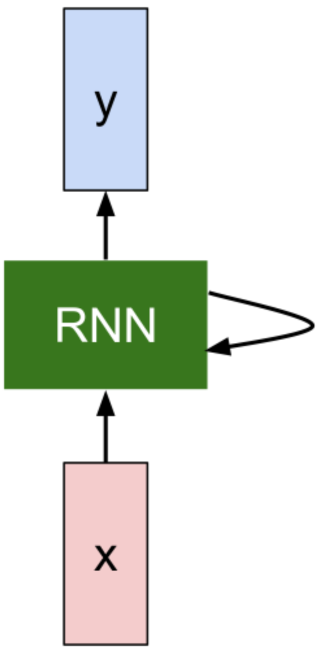
$$h_t = f_W(h_{t-1}, x_t)$$

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



(Simple) Recurrent Neural Network

The state consists of a single *“hidden”* vector **h**:



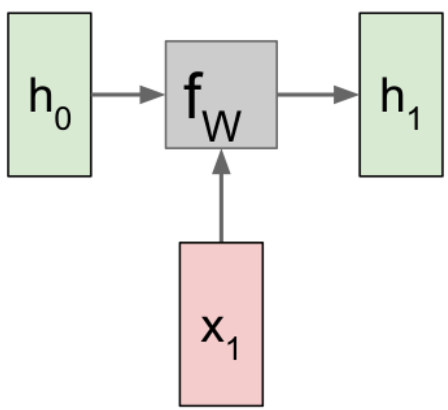
$$h_t = f_W(h_{t-1}, x_t)$$

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

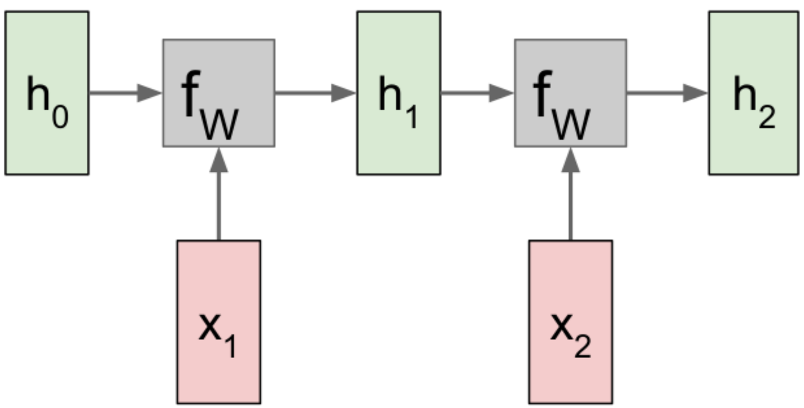
$$y_t = W_{hy}h_t$$

Sometimes called a “Vanilla RNN” or an “Elman RNN” after Prof. Jeffrey Elman

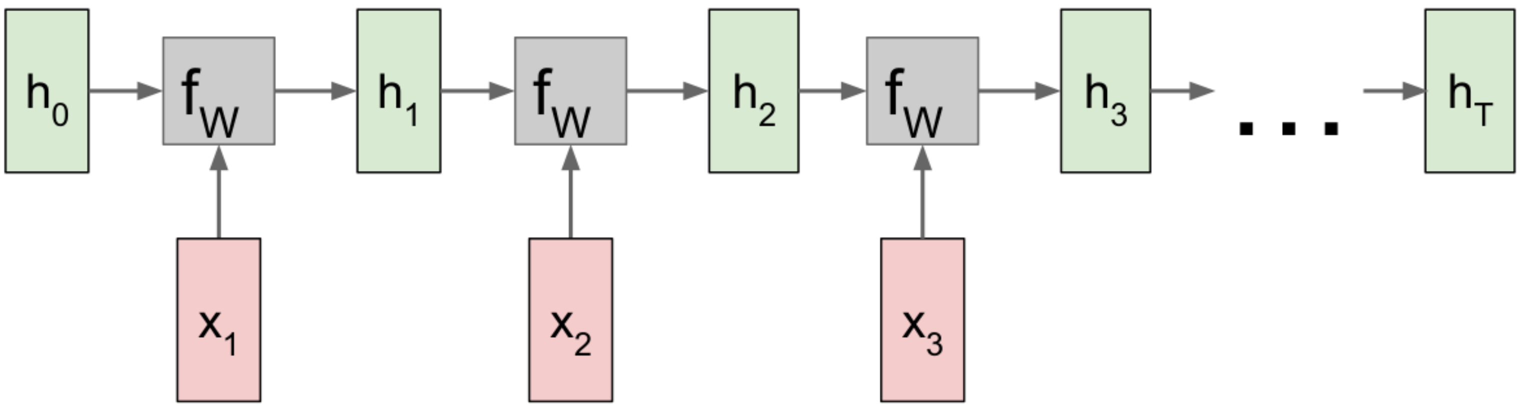
RNN: Computational Graph



RNN: Computational Graph

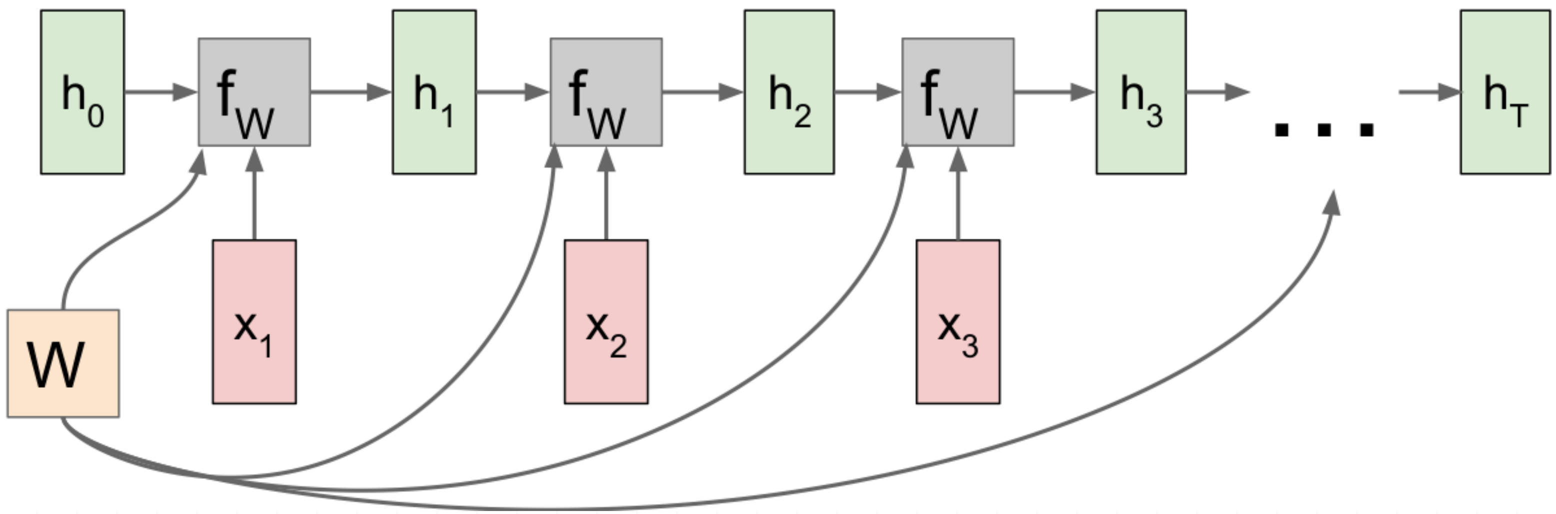


RNN: Computational Graph

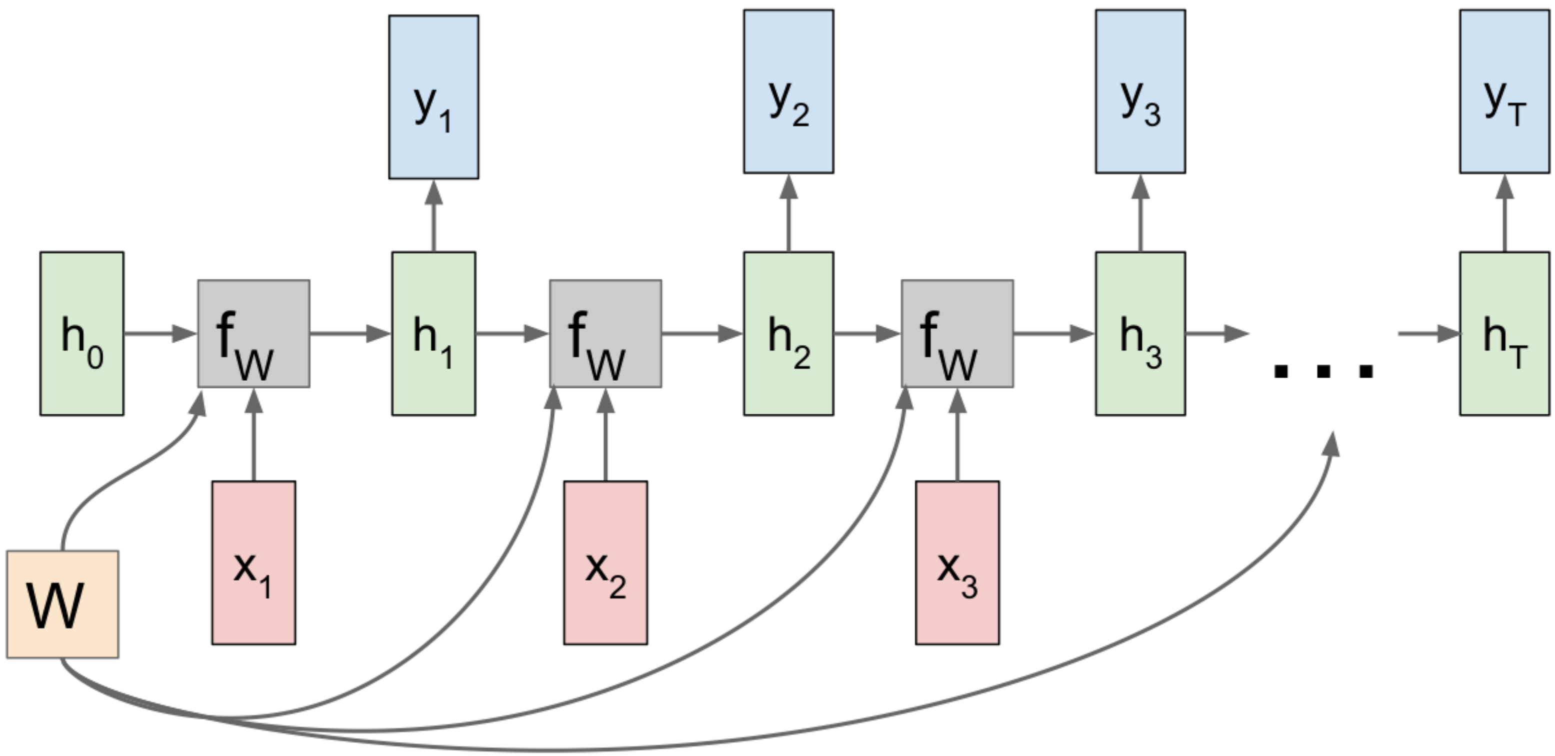


RNN: Computational Graph

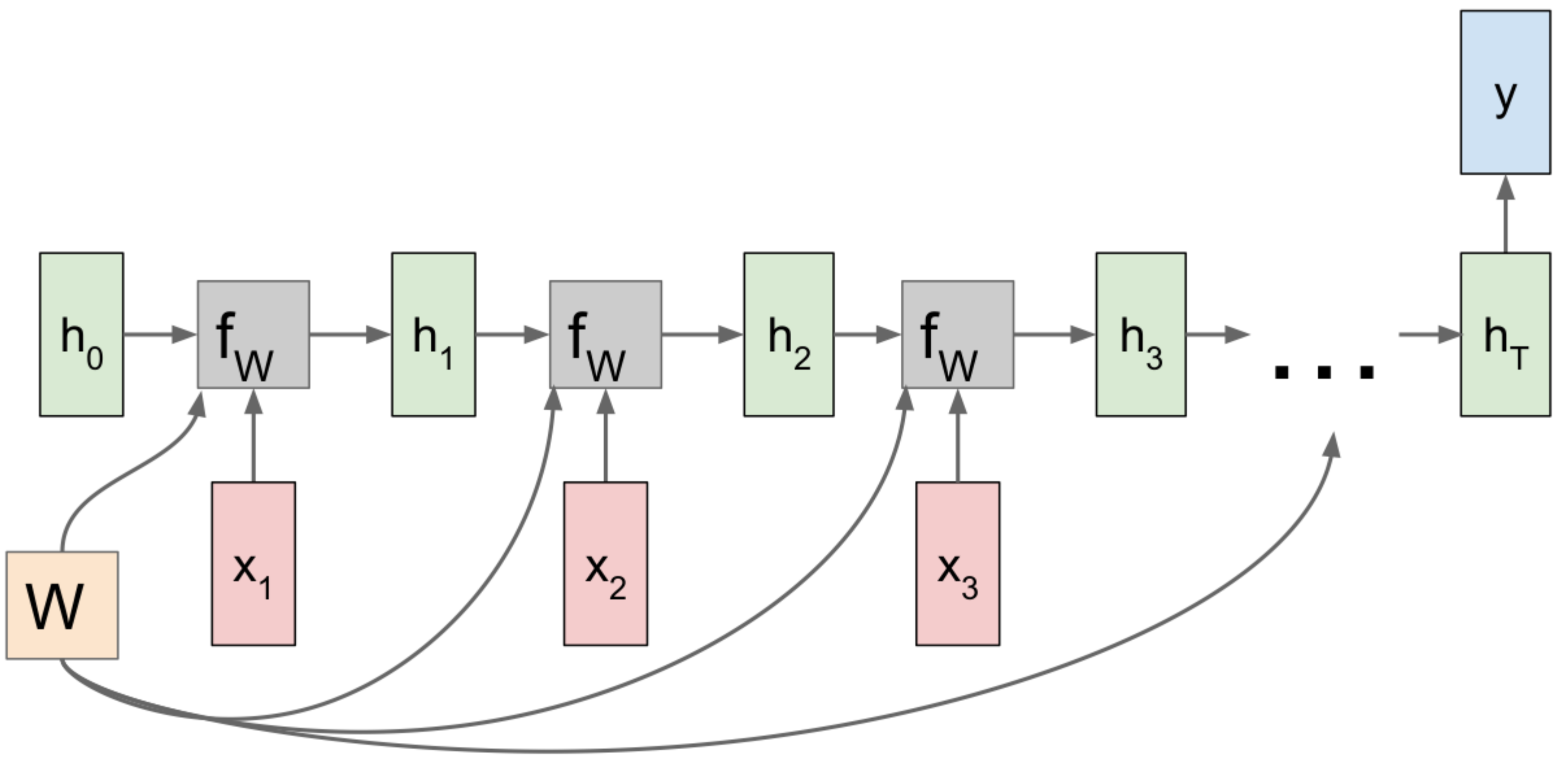
Re-use the same weight matrix at every time-step



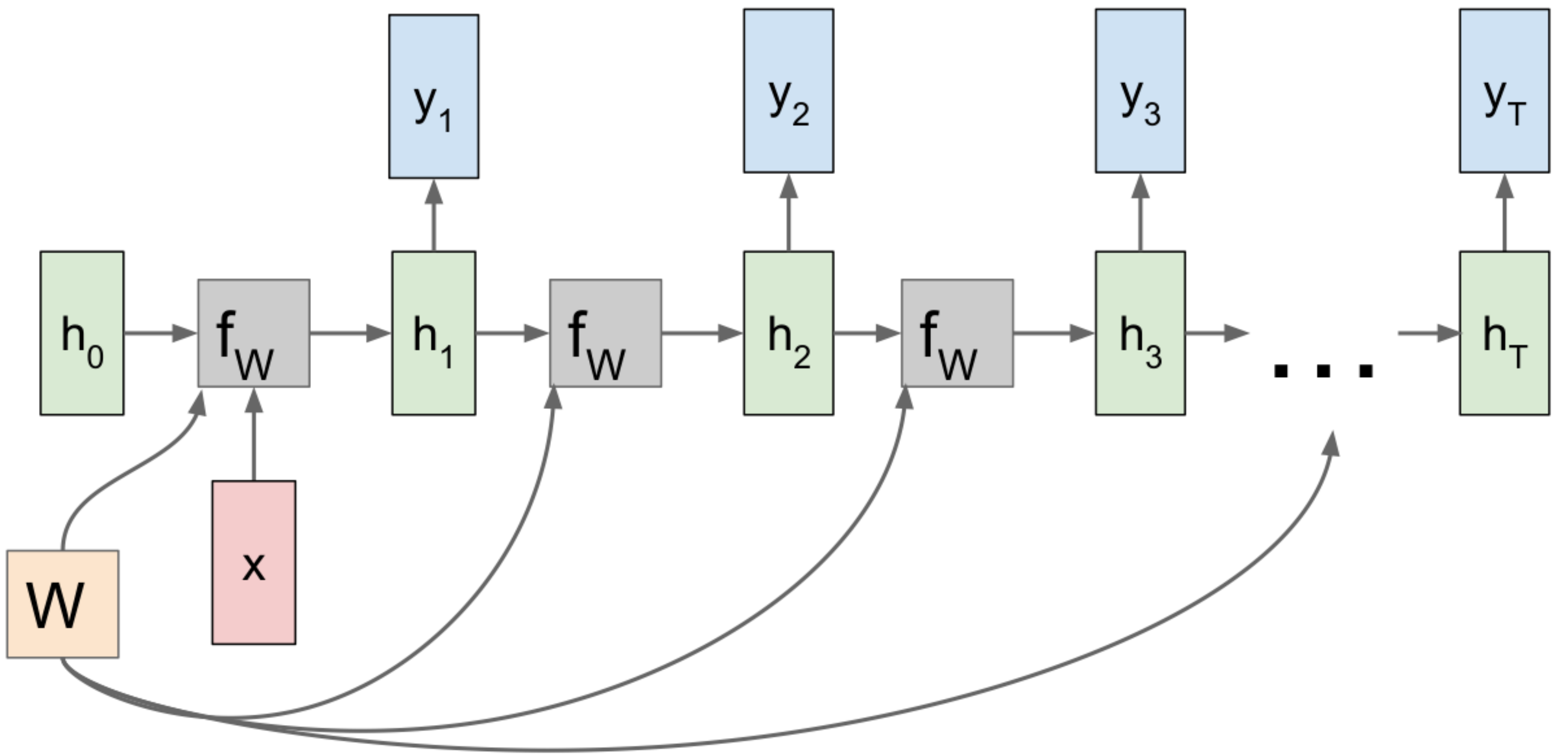
RNN: Computational Graph: Many to Many



RNN: Computational Graph: Many to One



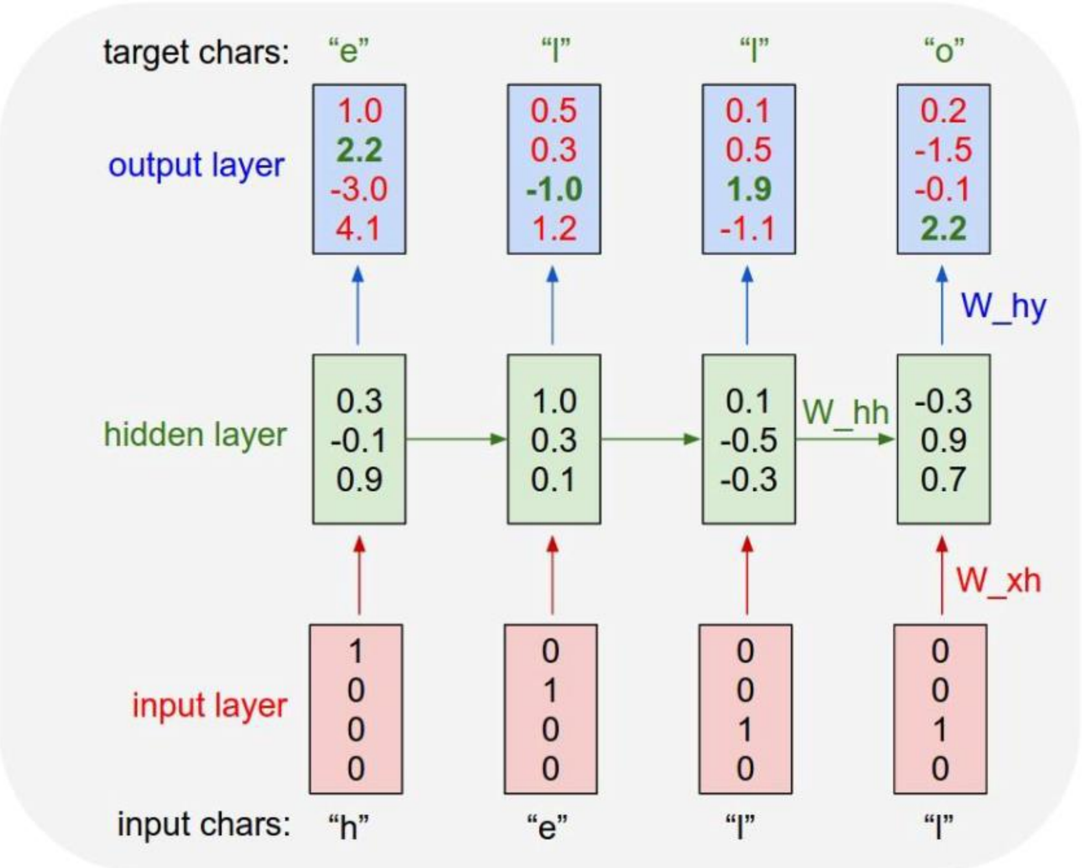
RNN: Computational Graph: One to Many



Example: Character-level Language Model

Vocabulary: [h,e,l,o]

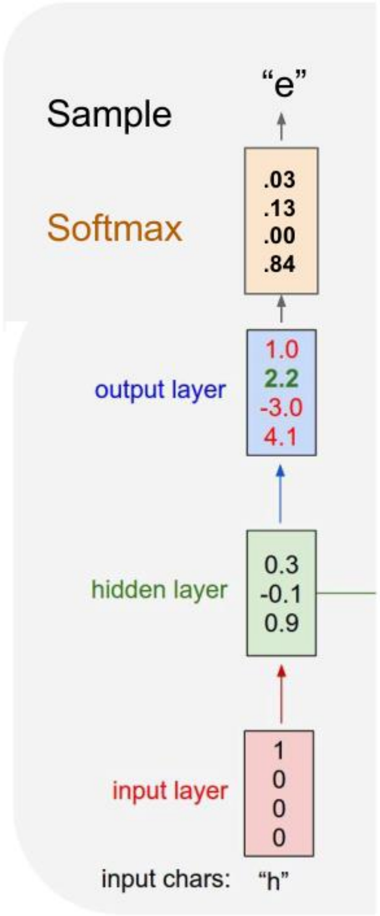
Example training sequence: **“hello”**



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

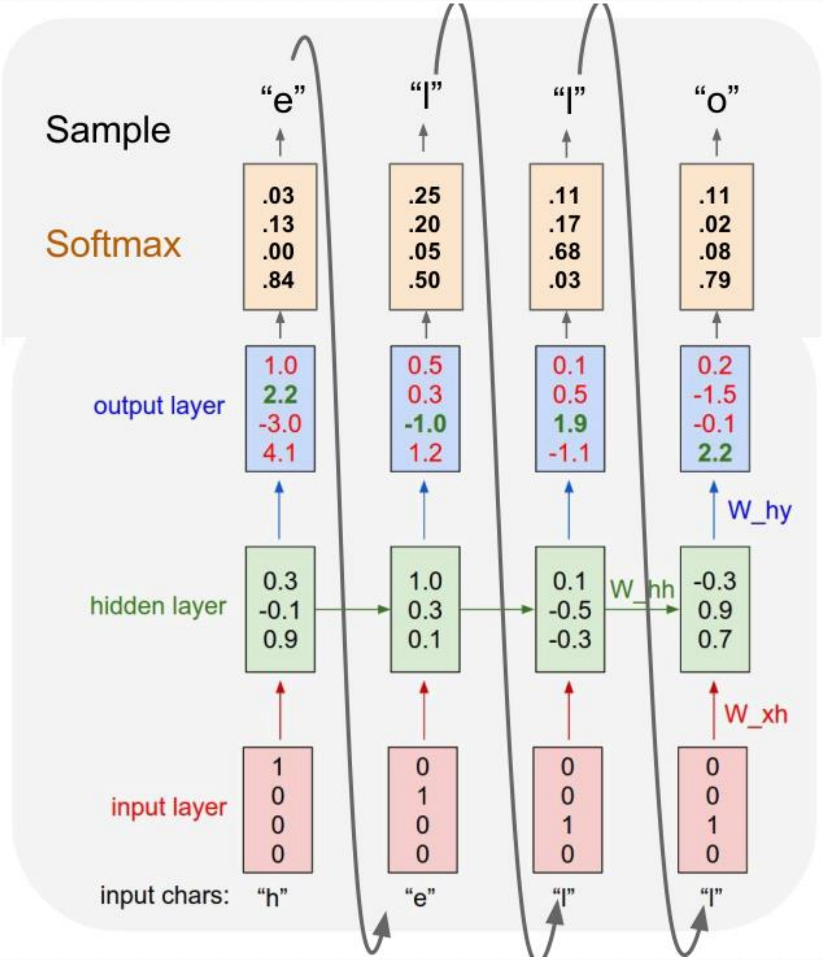
At test-time sample characters one at a time, feed back to model



Example: Character-level Language Model Sampling

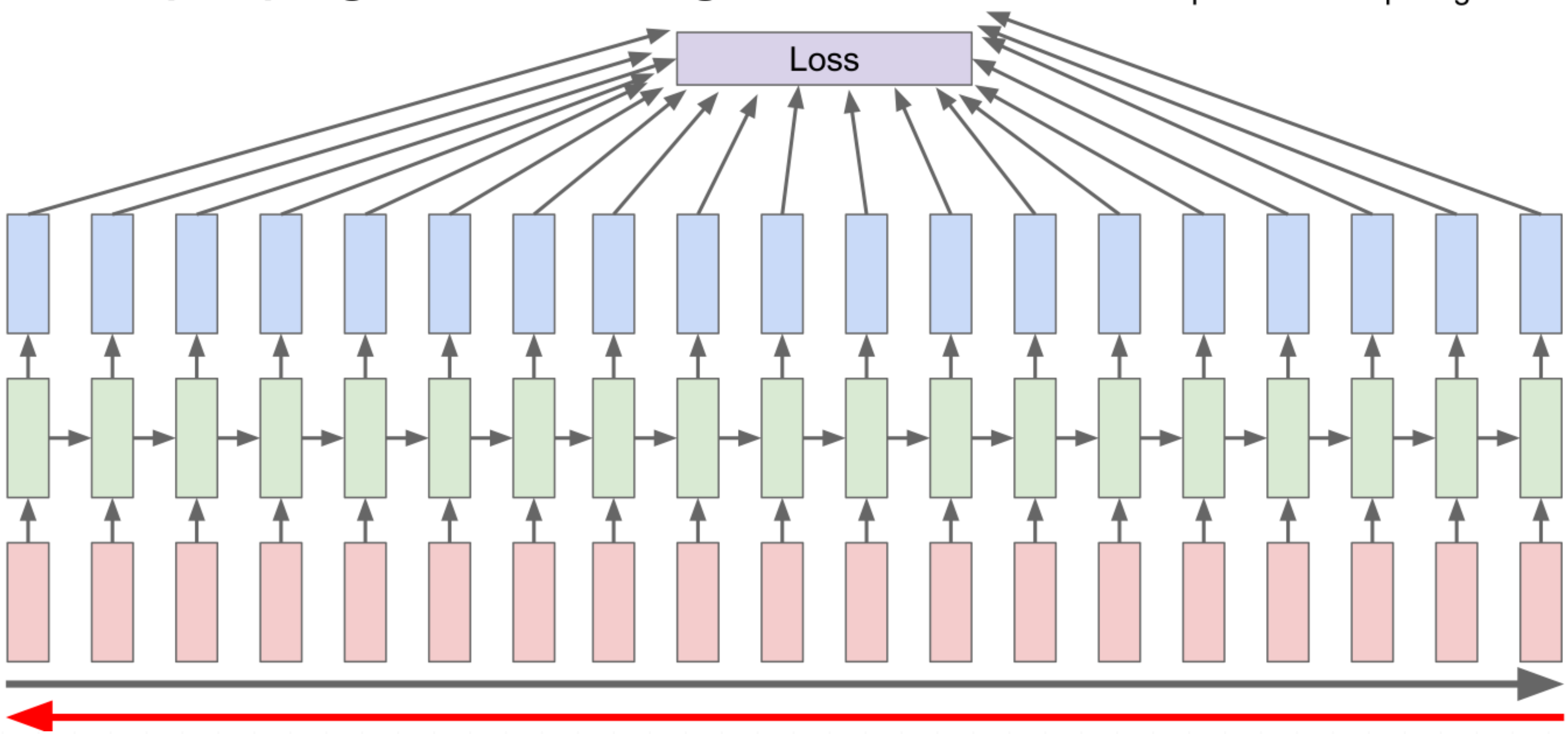
Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

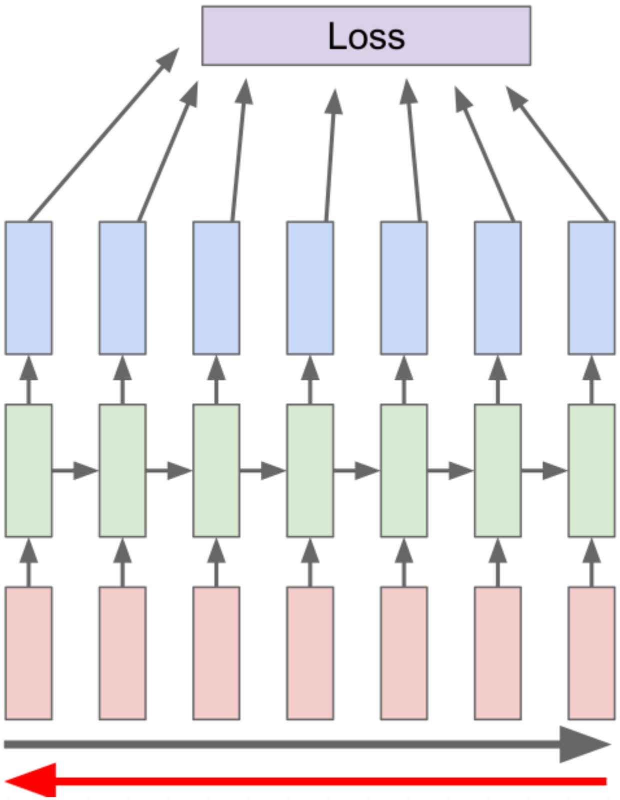


Backpropagation through time

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

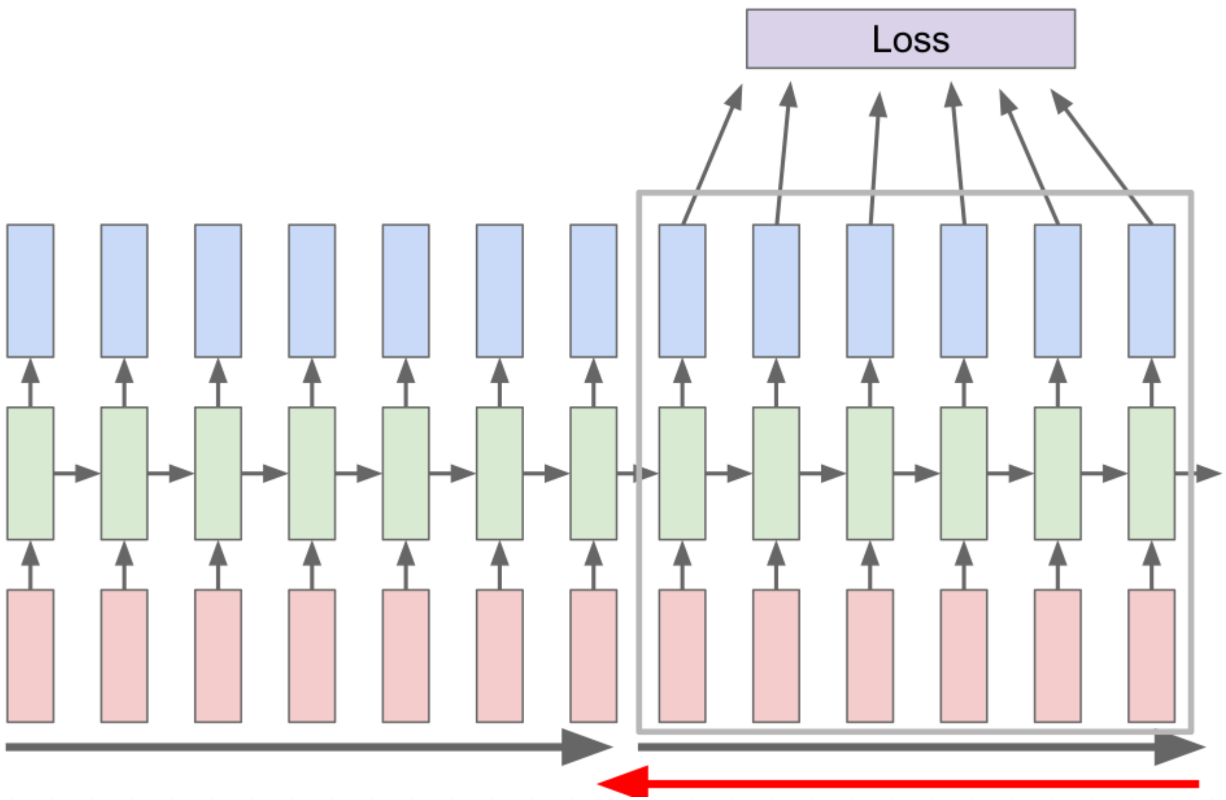


Truncated Backpropagation through time



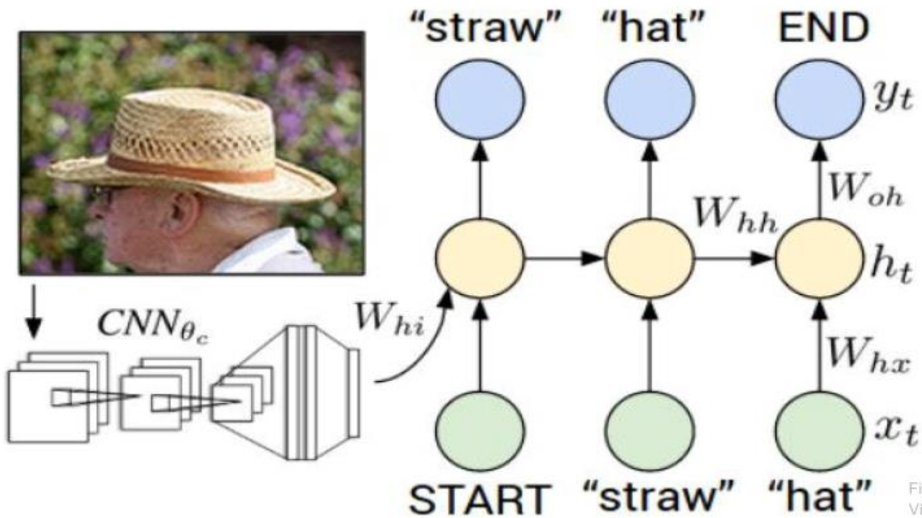
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time



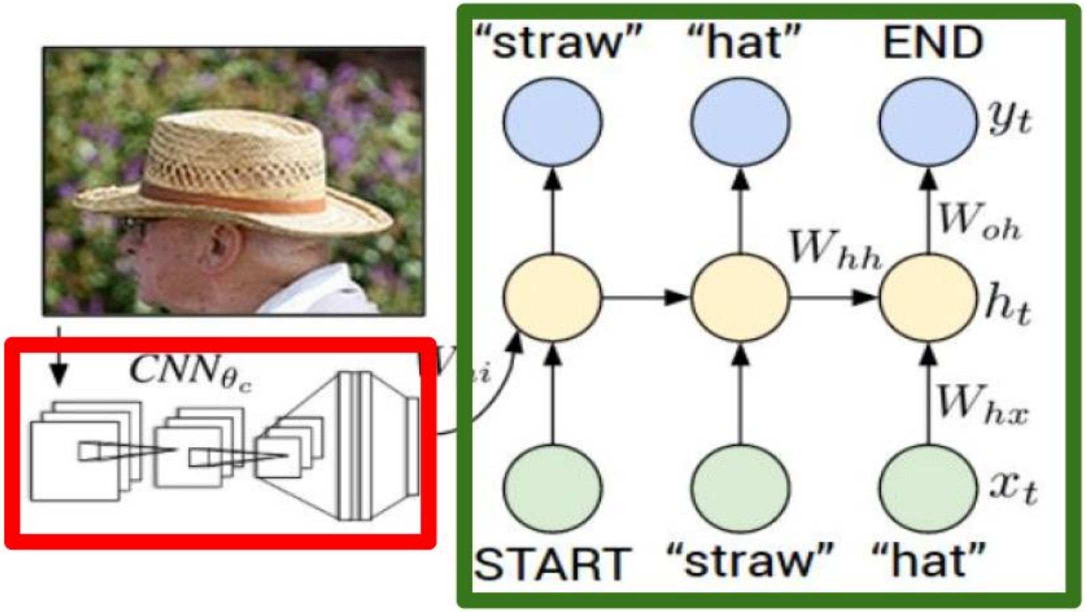
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Image Captioning



Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

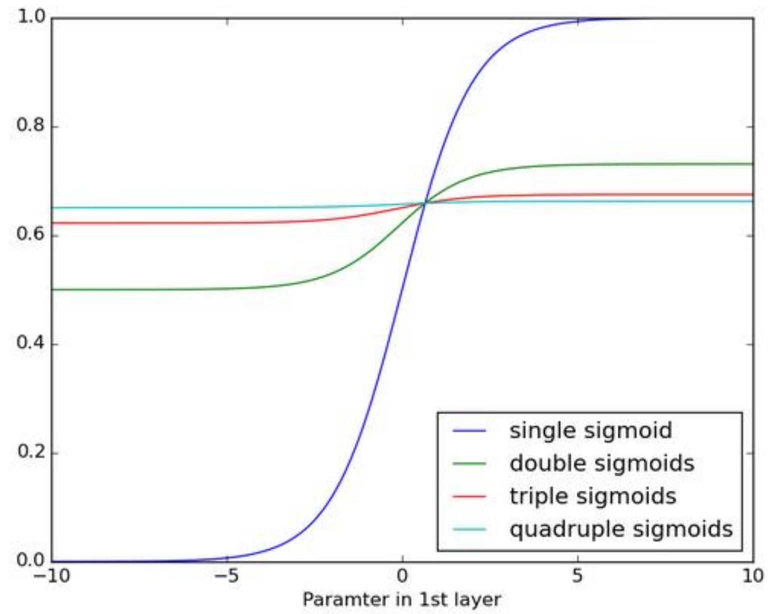
Recurrent Neural Network



Convolutional Neural Network

Vanishing or Exploding Gradients

- Because the layers and time steps of deep neural networks relate to each other through multiplication, derivatives are susceptible to vanishing or exploding.



https://skymind.ai/wiki/lstm

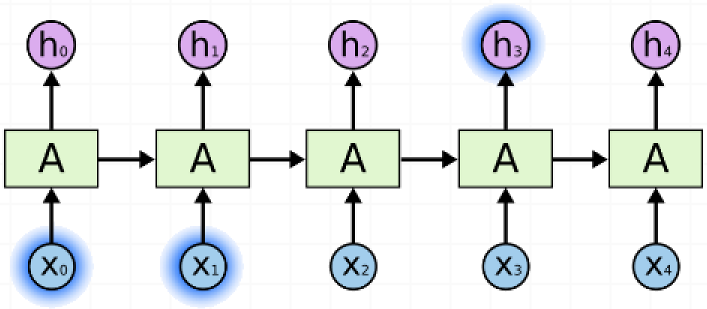
Long Short Term Memory (LSTM)

- LSTM networks
  - Not fundamentally different from RNN
  - Use different functions to compute hidden state
  - Memory of LSTMs are called cells
  - Cells decide what to keep in memory
- Very effective in capturing long-term dependencies

Long Term Dependencies

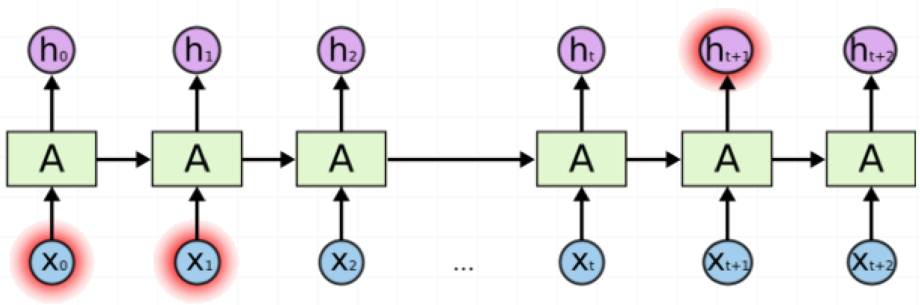
- Is RNN capable of capturing long-term dependencies?
- Why long-term dependencies?
  - Sometimes we only need to look at recent information to perform present task
- Consider an example
  - Predict next word based on the previous words

The clouds are in the sky



RNN & Long Term Dependency

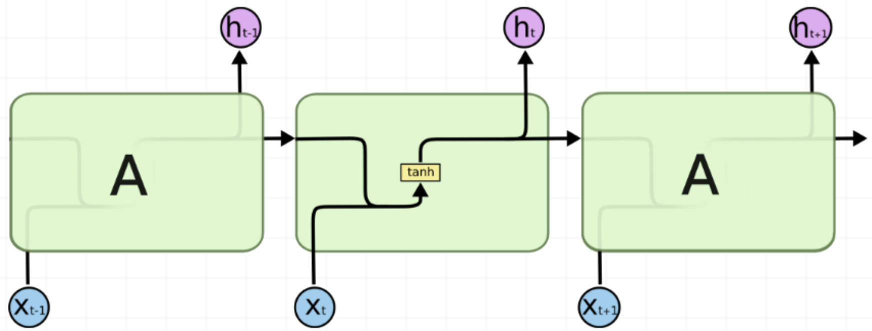
- What if we want to predict the next word in a long sentence?
- Do we know which past information is helpful to predict the next word?
- In theory, RNNs are capable of handling long-term dependencies.
- But in practice, they are not!



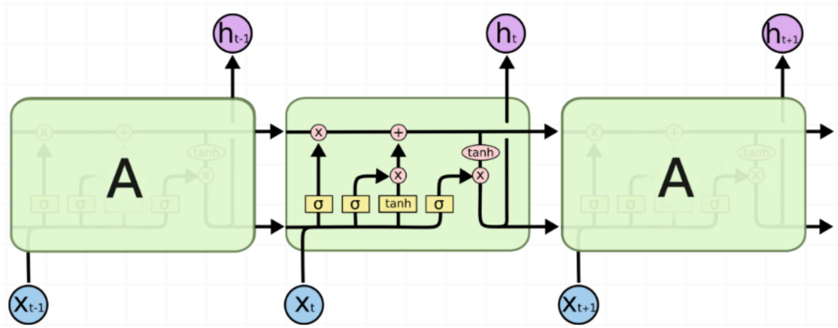
LSTM

- Special kind of recurrent neural network
- Works well in many problems and now widely used
- Explicitly designed to avoid the long-term dependency problem
- Remembering information for long periods of time is their default behavior
  - Not something they struggle to learn
- So, what is the structural difference between RNN and LSTM?

Difference between RNN and LSTM



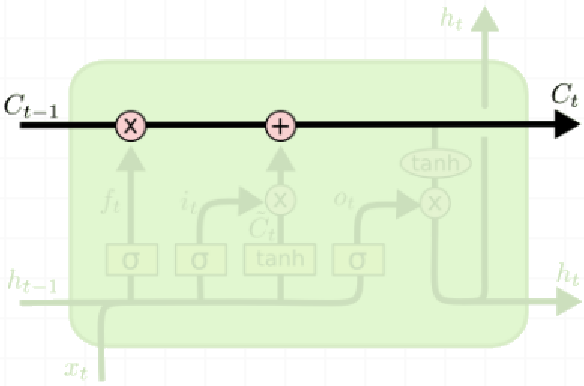
The repeating module in a standard RNN contains a single layer.

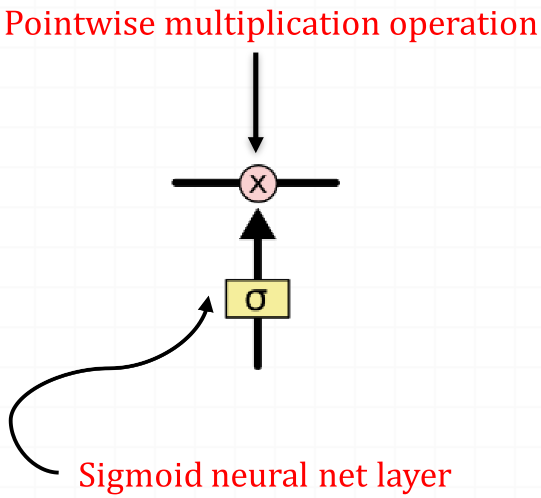


The repeating module in an LSTM contains four interacting layers.

Core Idea Behind LSTM

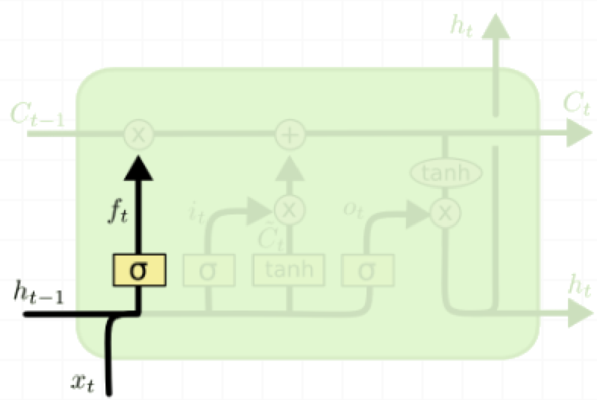
- Key to LSTMs is the cell state
    - The horizontal line running through the top of the diagram
- LSTM can add or remove information to the cell state
- How? Through regulated structures called gates.
- LSTM has three gates to protect and control cell state





Step-by-Step LSTM Walkthrough

- Forget gate layer decides what information will be thrown away
- Looks at h_{t-1} and x_t and outputs a number between 0 and 1
- 1 represents completely keep this, 0 represents completely get rid of this
- Example: forget the gender of the old subject, when we see a new subject

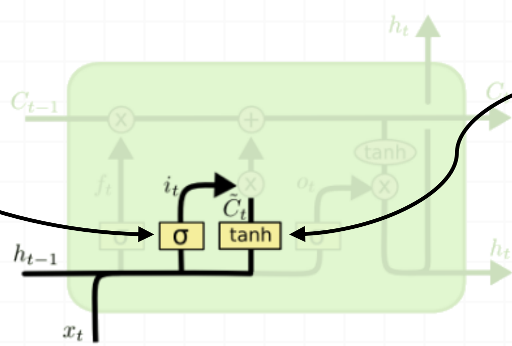


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Step-by-Step LSTM Walkthrough

- Next step: decides what new information will be stored in the cell state
- Two parts –
  - A sigmoid layer (input gate layer): decides what values we'll update
  - A tanh layer: creates a vector of new candidate values, $\tilde{C}_t$
- Example: add the gender of the new subject to the cell state
  - Replace the old one we're forgetting

Input gate layer



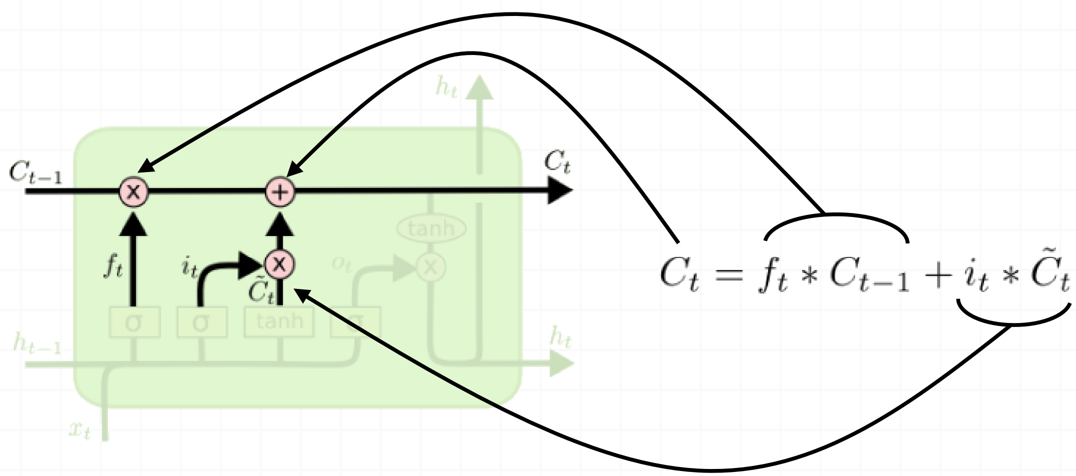
tanh layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

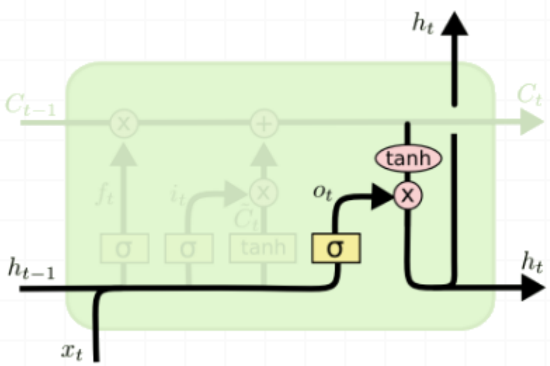
Step-by-Step LSTM Walkthrough

- Next step: update old state by C_{t-1} into the new cell state C_t
- Multiply old state by f_t
    - Forgetting the things we decided to forget earlier
- Then we add $i_t * \tilde{C}_t$



Step-by-Step LSTM Walkthrough

- Final step: decide what we're going to output
- First, we run a sigmoid layer
  - Which decides what parts of the cell state we're going to output
- Then, we put the cell state through tanh and multiply it by the output of the sigmoid gate



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ \tilde{C} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

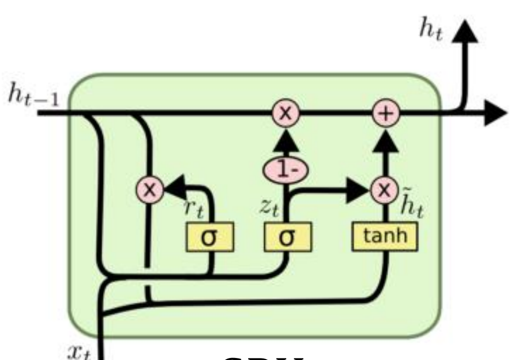
$$c_t = f \odot c_{t-1} + i \odot \tilde{C}$$

$$h_t = o \odot \tanh(c_t)$$

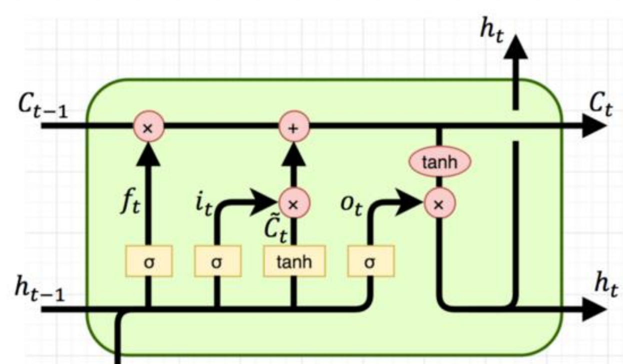
i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell

Gated Recurrent Unit (GRU)

- Similar to LSTMs, except that they have two gates:
  - Reset gate and Update gate.
- Reset gate determines how to combine new input with previous memory
- Update gate determines how much of the previous state to keep.



GRU



LSTM