Recurrent Neural Networks

Gemma Roig
Computer Vision
Goethe University

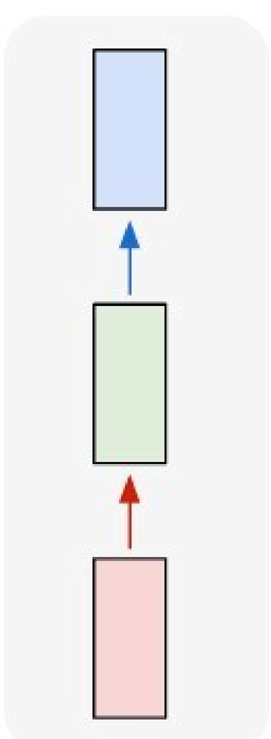
Today's class objectives

- What are Recurrent Neural Networks
- When to use RNN
- Training RNN
- Long-Short Term Memory Networks (LSTM)

Recurrent Neural Networks

ar…
nave seen neural networks to model one to one
endencies

one to one



Input: No sequence - example: image

Output: No sequence - examples: label of object class

Example:

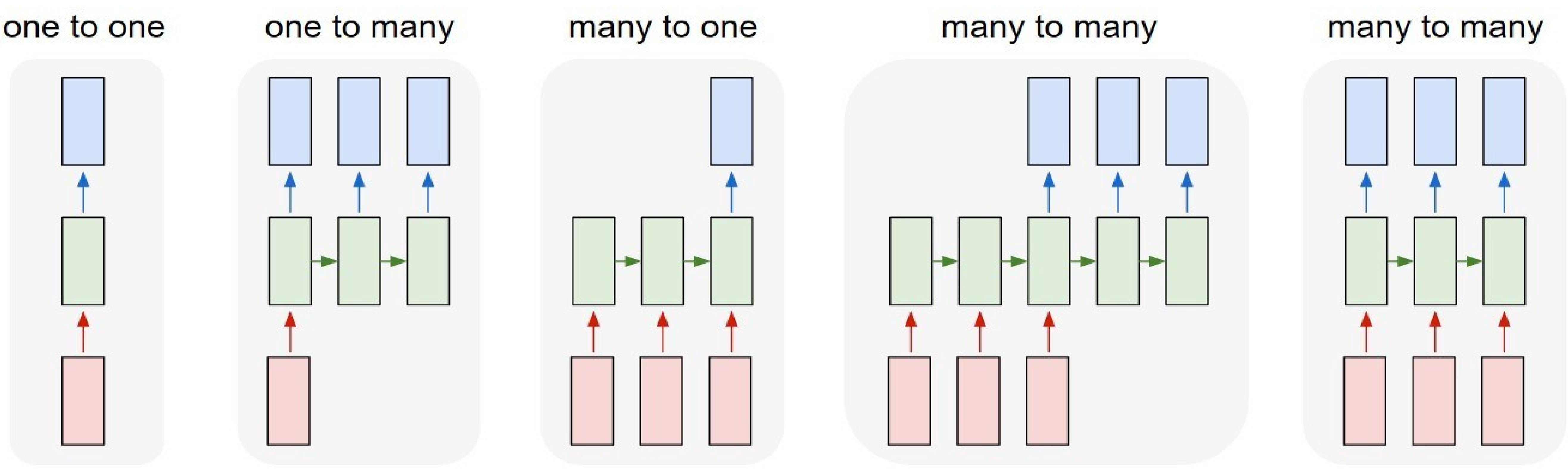
- “standard” classification
- regression problems

Source: Dhruv Batra, Andrej Karpathy

Recurrent Neural Networks

How do we model sequences?

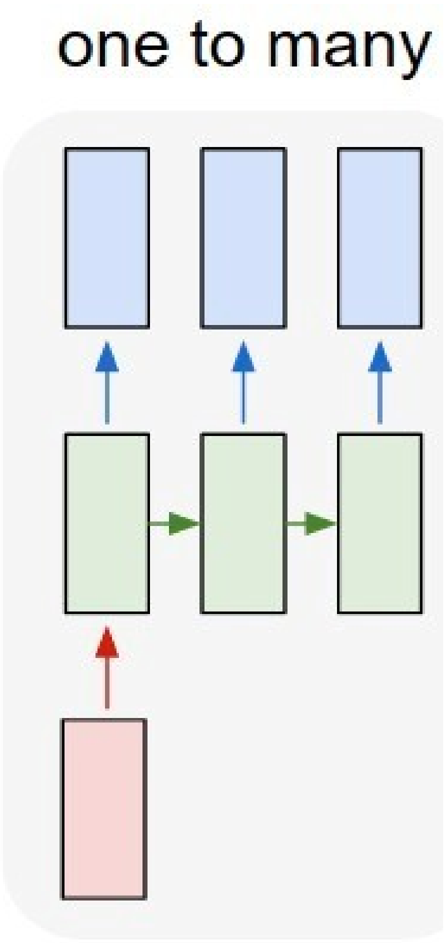
Input -> Output



Source: Dhruv Batra, Andrej Karpathy

Recurrent Neural Networks

How do we model sequences?



Input: No sequence

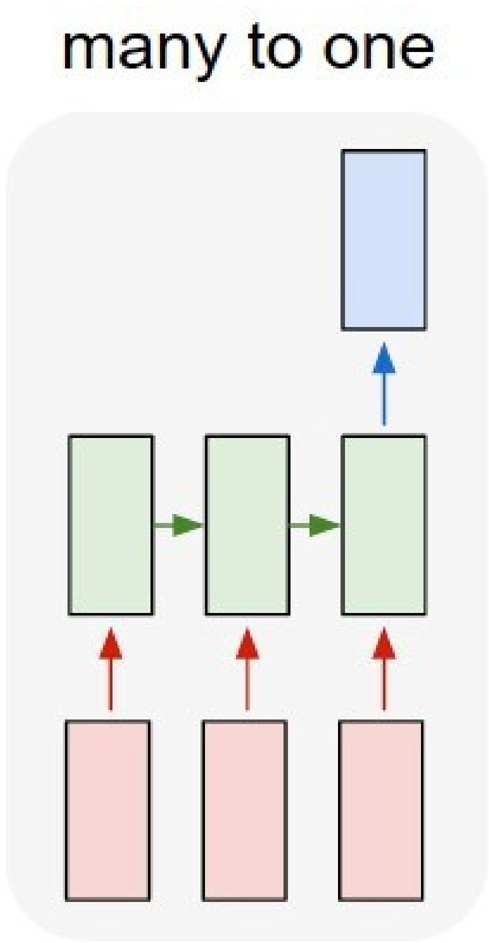
Output: Sequence

Example:
- Image captioning (image -> words)

Source: Dhruv Batra, Andrej Karpathy

Recurrent Neural Networks

How do we model sequences?



Input: Sequence

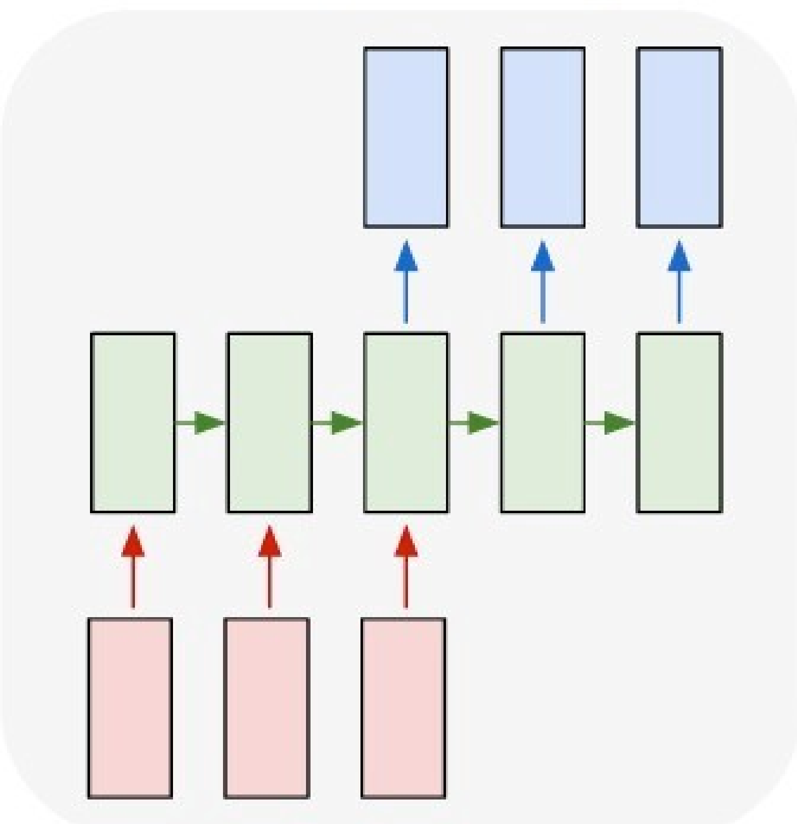
Output: No sequence

Example:
- sentence classification
- sentiment classification
  (words seq.->sentiment)

Recurrent Neural Networks

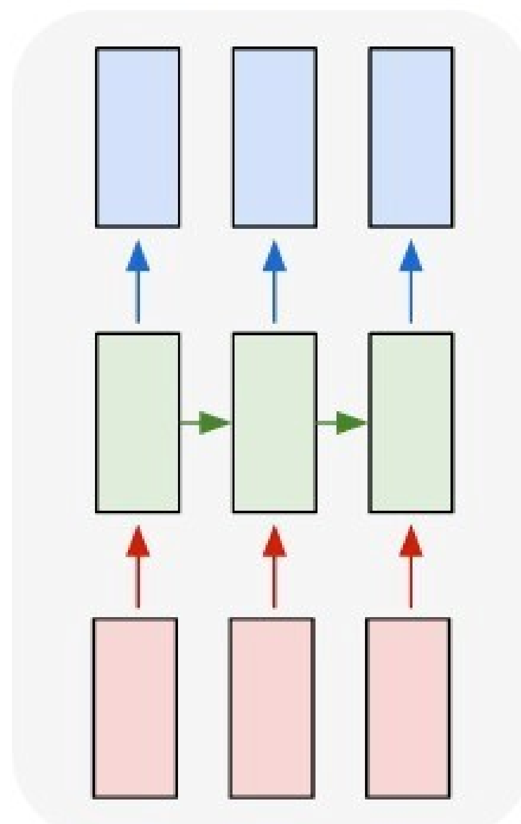
How do we model sequences?

many to many



Example:
- machine translation, (words seq-> words seq)

many to many



Example:
- video captioning

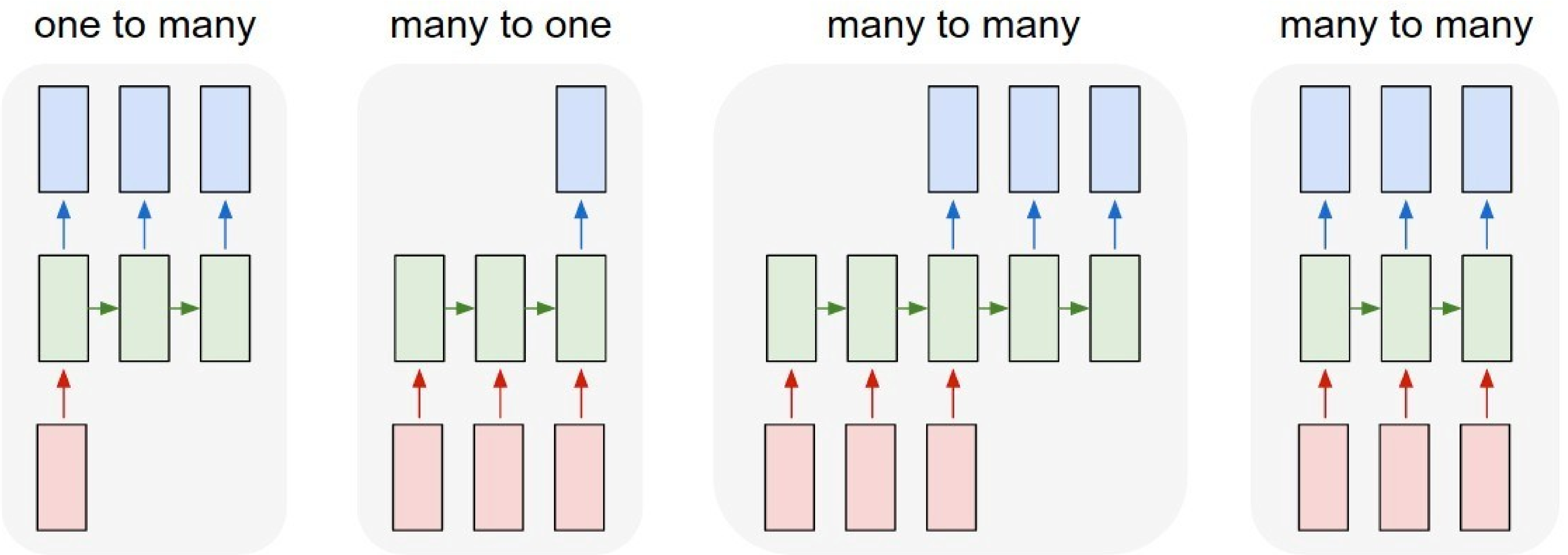
Input: Sequence

Output: Sequence

Source: Dhruv Batra, Andrej Karpathy

Recurrent Neural Networks

How do we model sequences?



Input: No sequence

Output: Sequence

Example:

- Image captioning (image -> words)

Input: Sequence

Output: No sequence

Example:

- sentence classification
- sentiment classification (words seq.->sentiment)

Input: Sequence

Output: Sequence

Example:

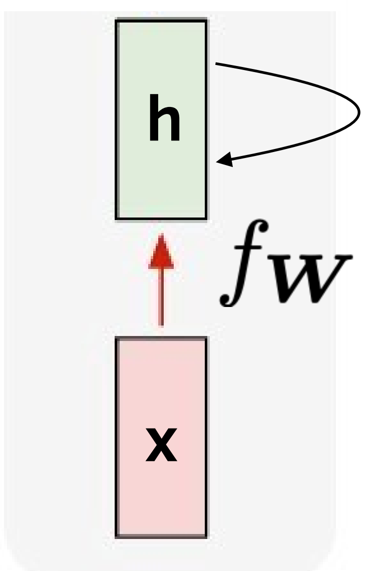
- machine translation, (words seq-> words seq)

Example:

- video captioning

Source: Dhruv Batra, Andrej Karpathy

Recurrent Neural Networks



Recurrent formula at each time step:

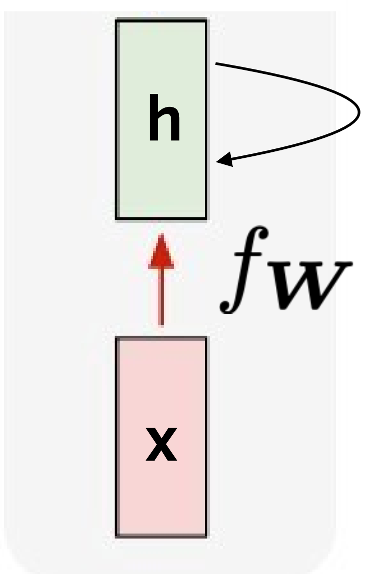
$$\boldsymbol{h}_t = f_{\boldsymbol{W}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

x: input sequence of vectors

h: hidden units, representing the state of the network

f_W : function with parameters W

Recurrent Neural Networks



Recurrent formula at each time step:

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

new state **old state** **input vector at time t**

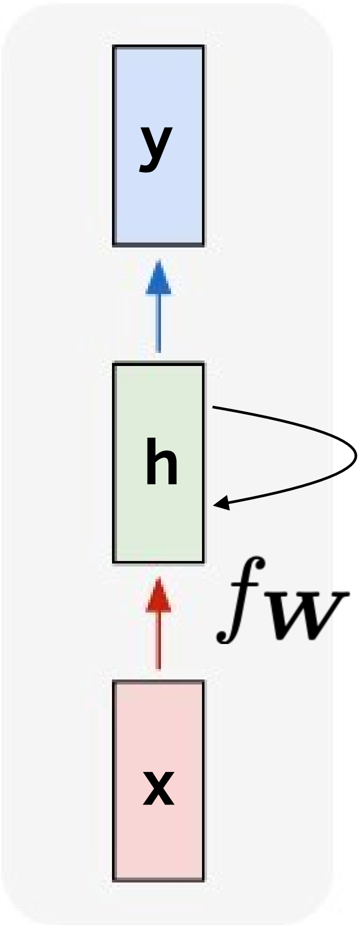
x: input sequence of vectors

h: hidden units, representing the state of the network

f_W: function with parameters W.
The same function and same W are used at every time step

Recurrent Neural Networks

Example - RNN with one hidden vector h:



$$\boldsymbol{h}_t = f_{\boldsymbol{W}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

$$f_{\boldsymbol{W}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) = tanh(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t)$$

$$\boldsymbol{y}_t = \boldsymbol{W}_{hy}\boldsymbol{h}_t$$

x: input sequence of vectors

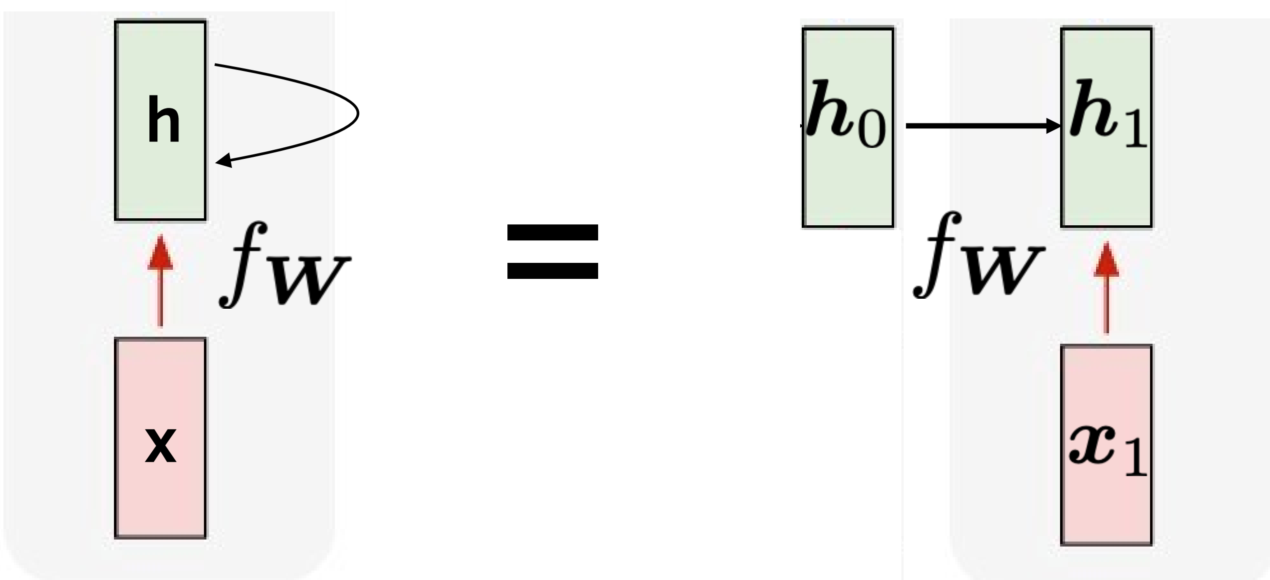
h: hidden units, representing the state of the network

f_W: function with parameters W

y: output sequence

Recurrent Neural Networks

Computational Graph- Unrolled recurrent neural network:

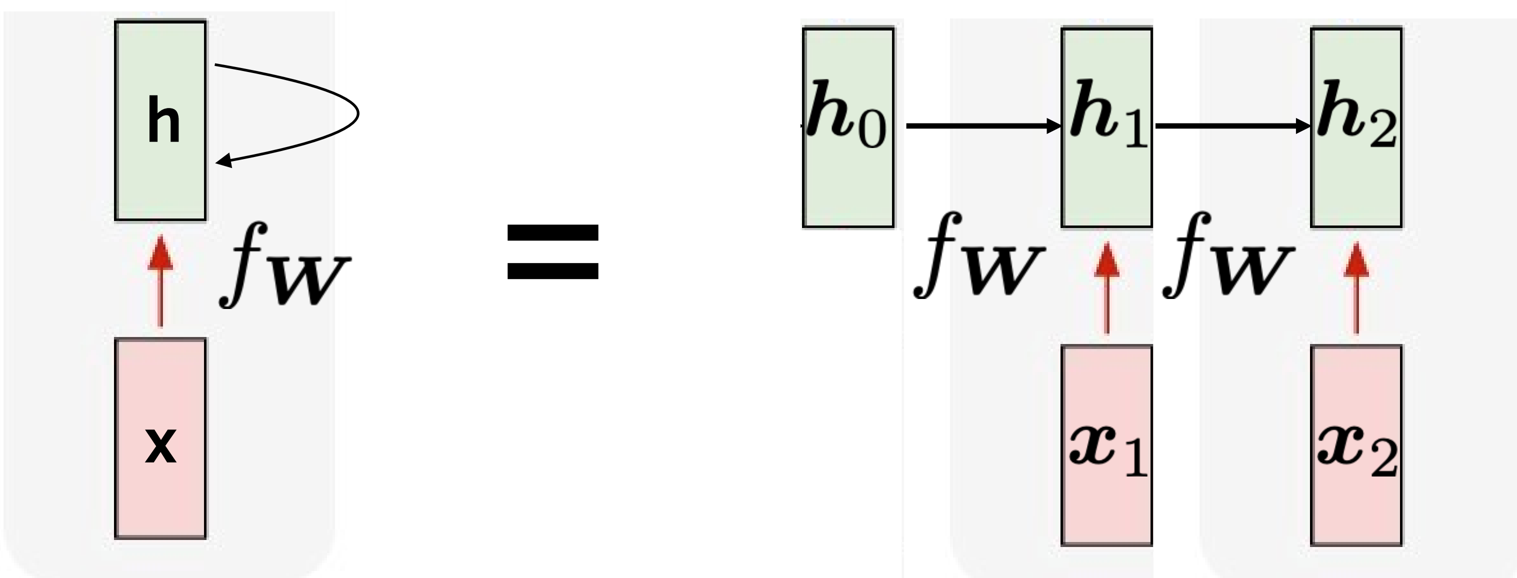


$$h_t = f_W(h_{t-1}, x_t)$$

- A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor

Recurrent Neural Networks

Computational Graph-
Unrolled recurrent neural network:

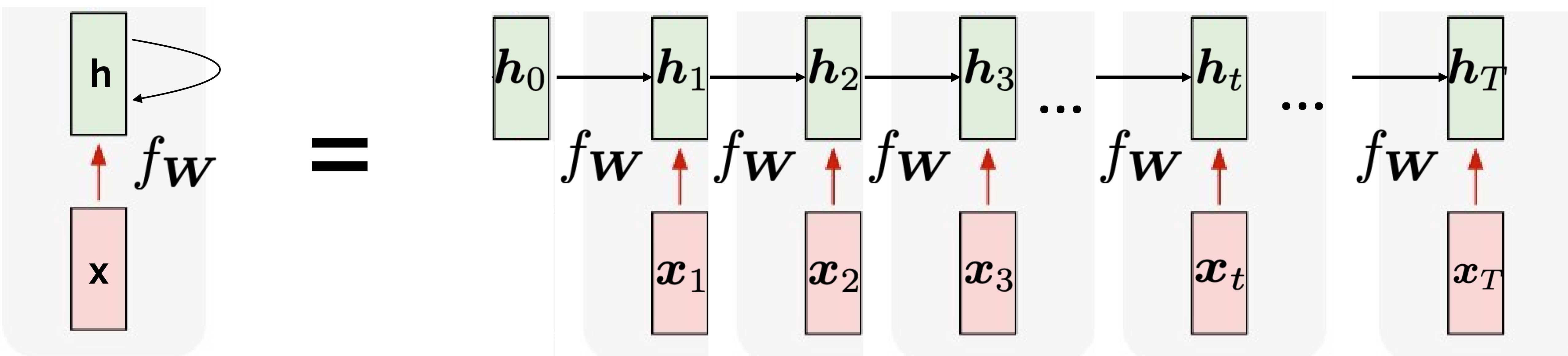


$$h_t = f_W(h_{t-1}, x_t)$$

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

Recurrent Neural Networks

Computational Graph-
Unrolled recurrent neural network:

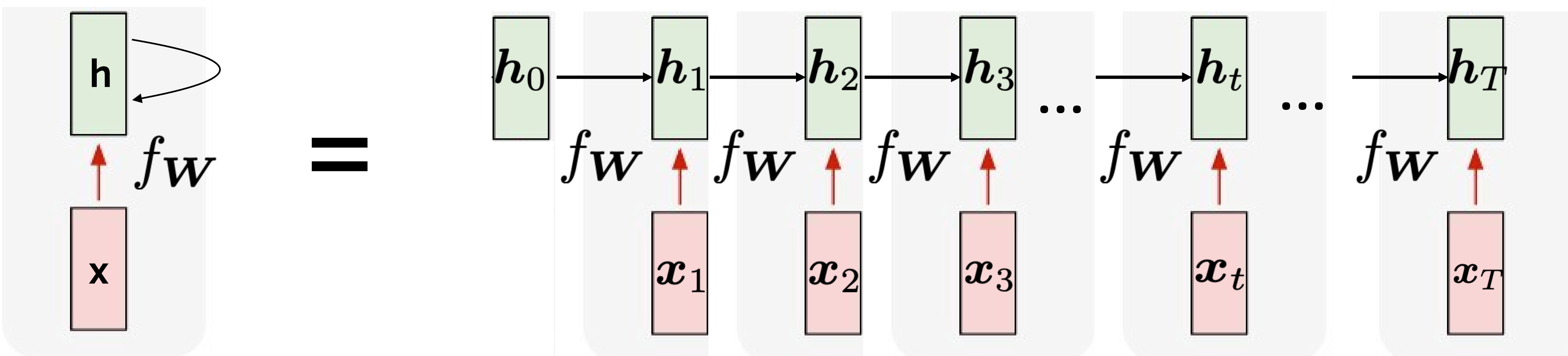


$$h_t = f_W(h_{t-1}, x_t)$$

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

Recurrent Neural Networks

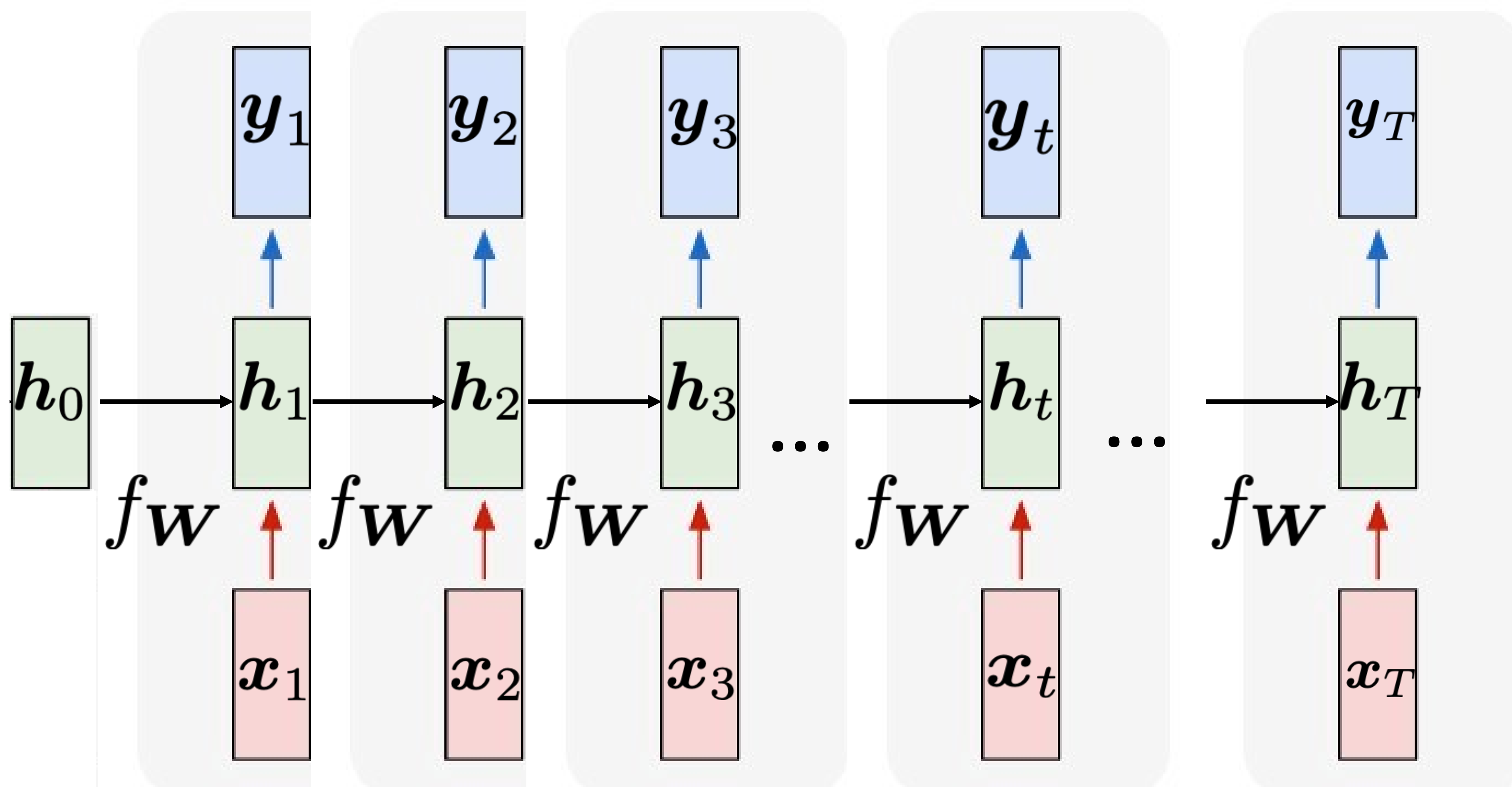
Computational Graph-
Unrolled recurrent neural network:



If the relevant input information is close to where is needed, the RNNs can learn to use the past information

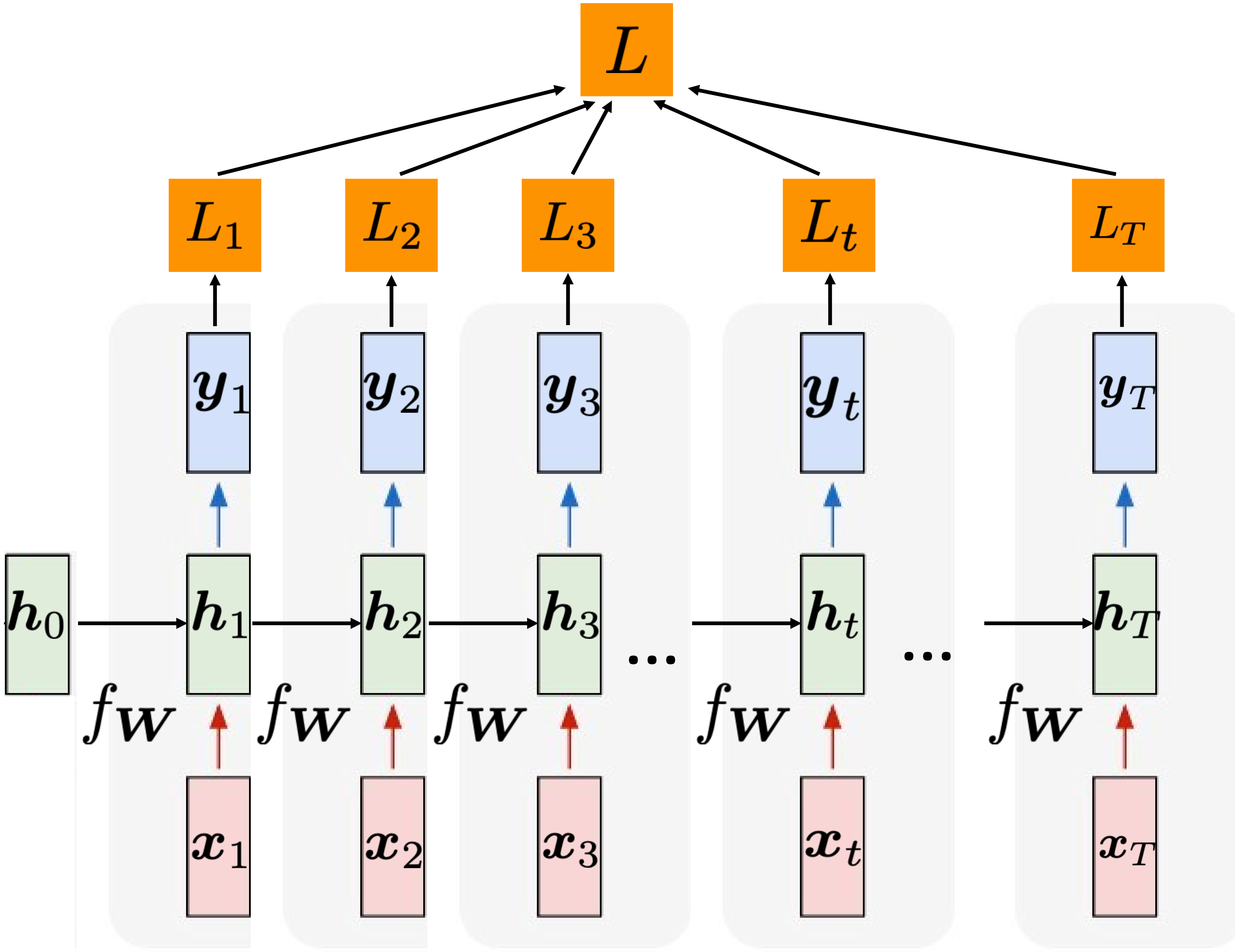
Recurrent Neural Networks

Computational Graph - Many to many



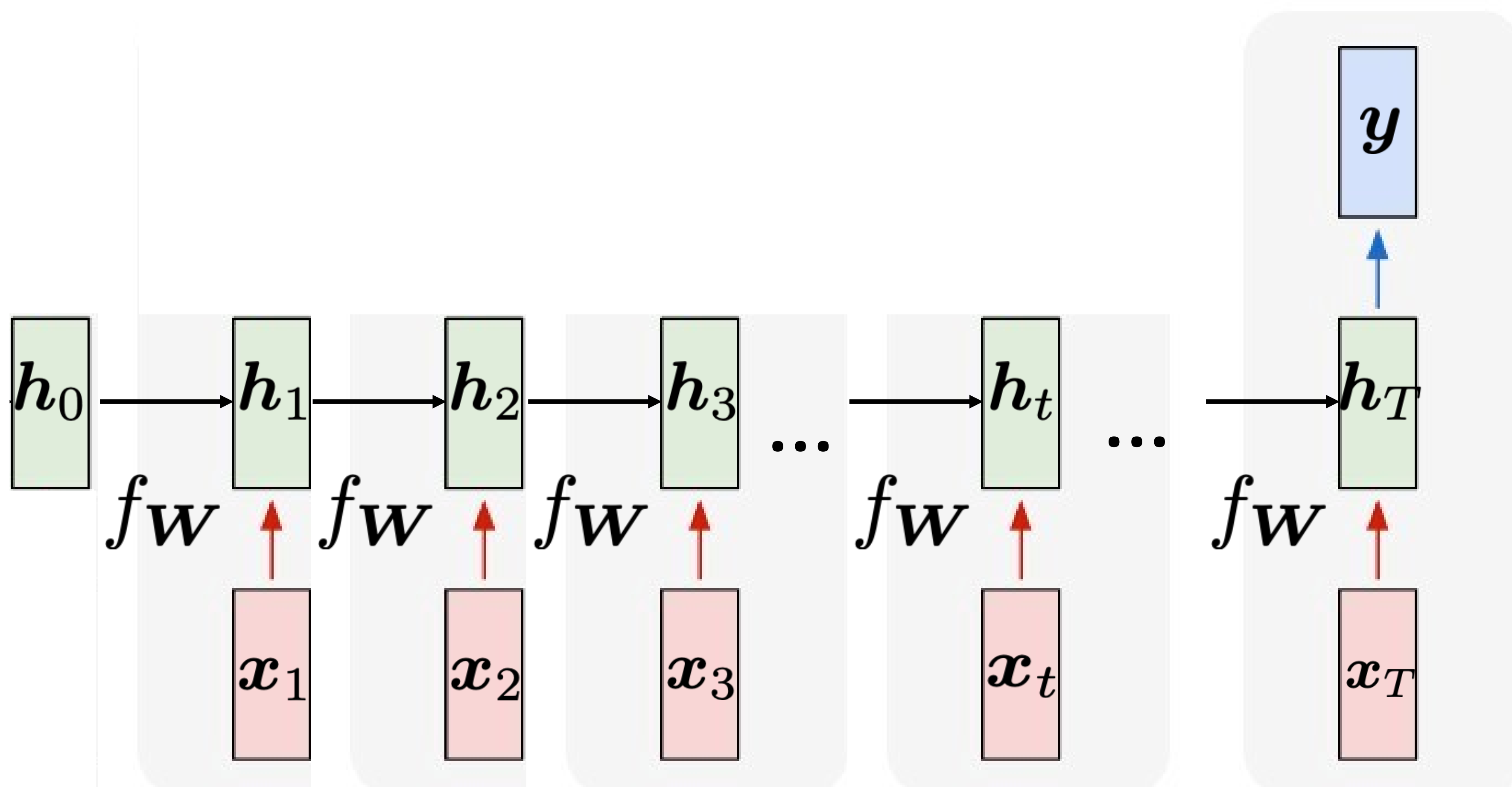
Recurrent Neural Networks

Computational Graph - Many to many



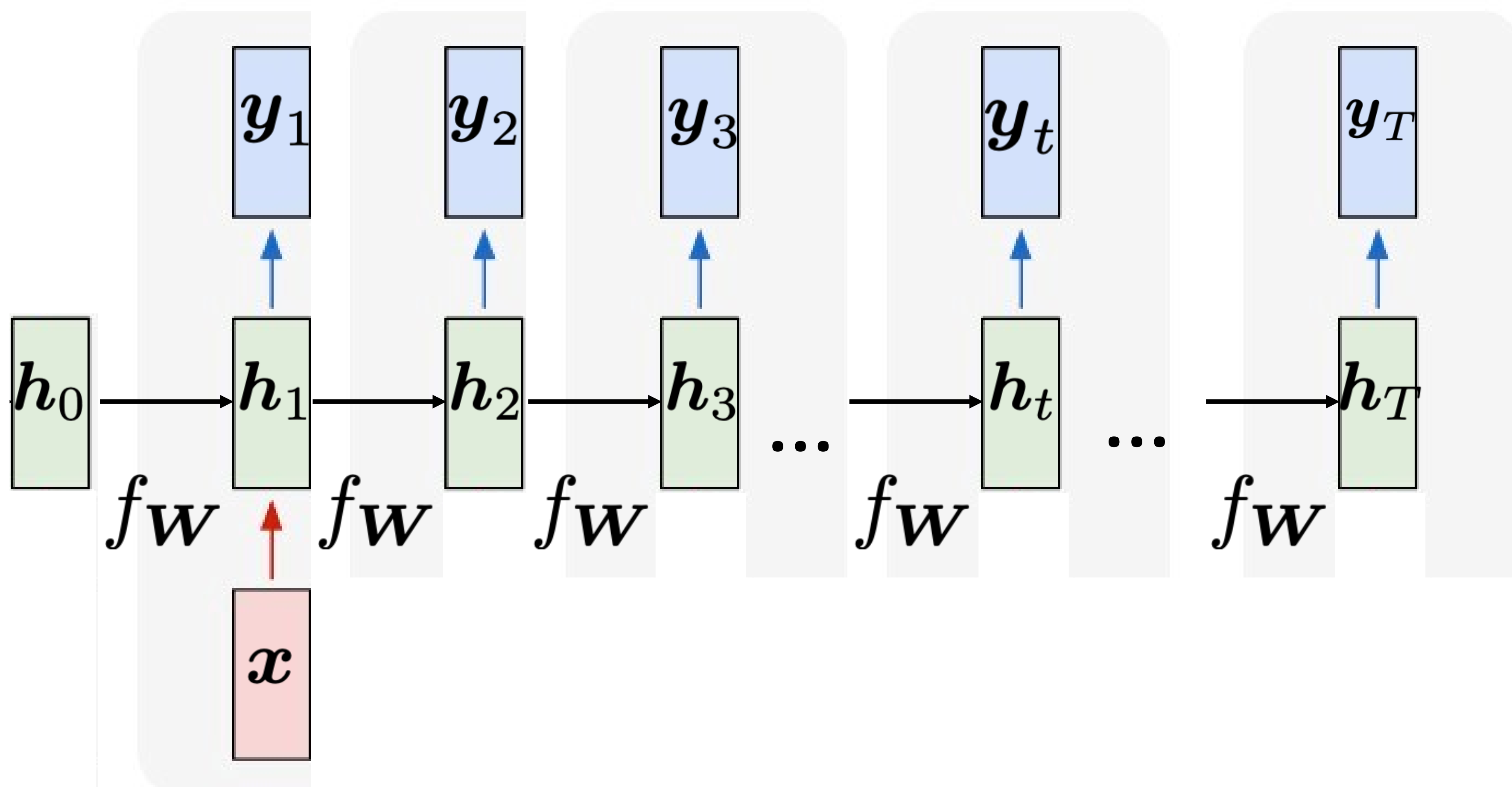
Recurrent Neural Networks

Computational Graph - Many to one



Recurrent Neural Networks

Computational Graph - One to many

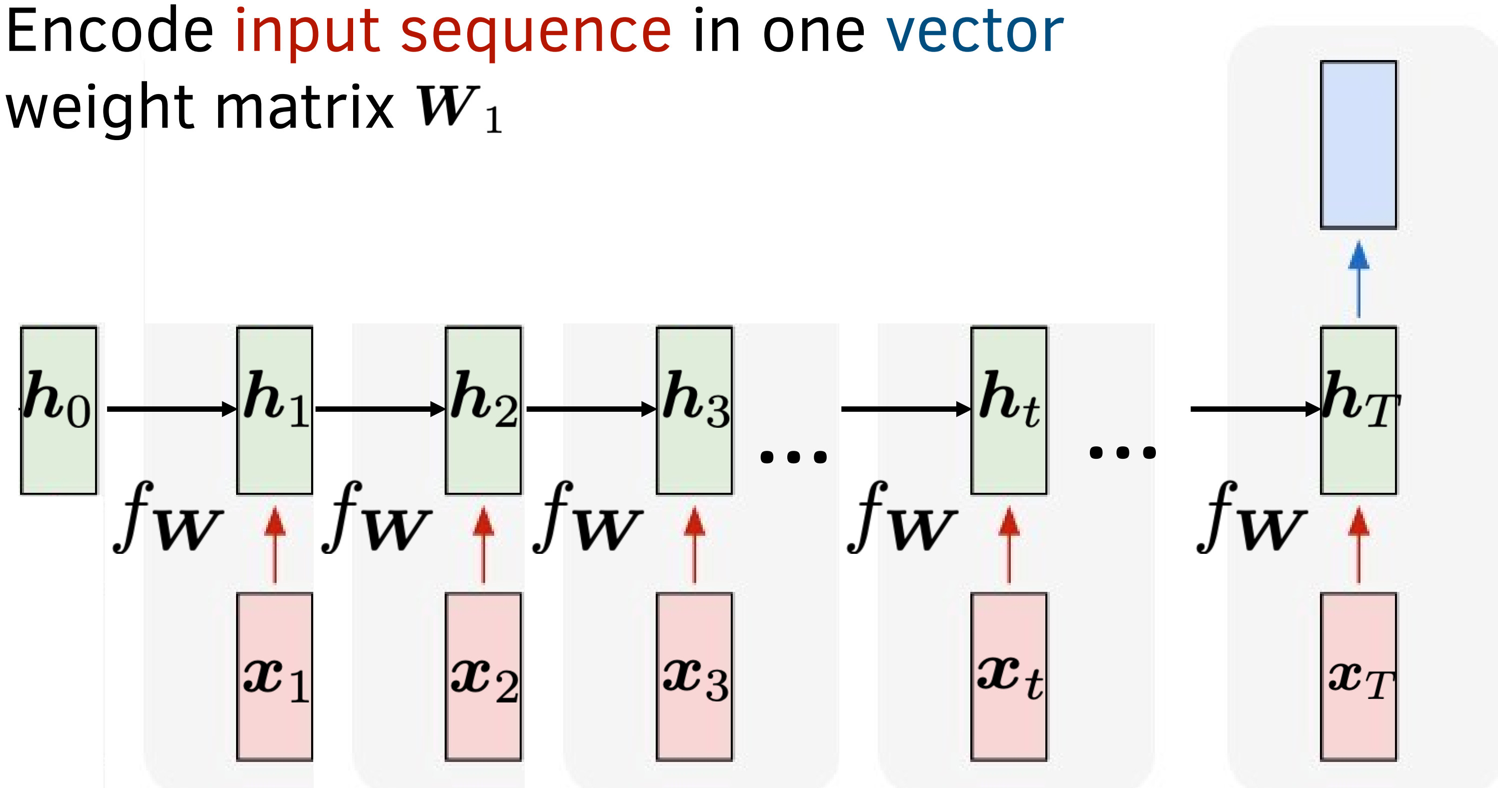


Recurrent Neural Networks

Computational Graph - Sequence to sequence: many to one + one to many

Many to one:

- Encode input sequence in one vector
- weight matrix $\boldsymbol{W}_1$

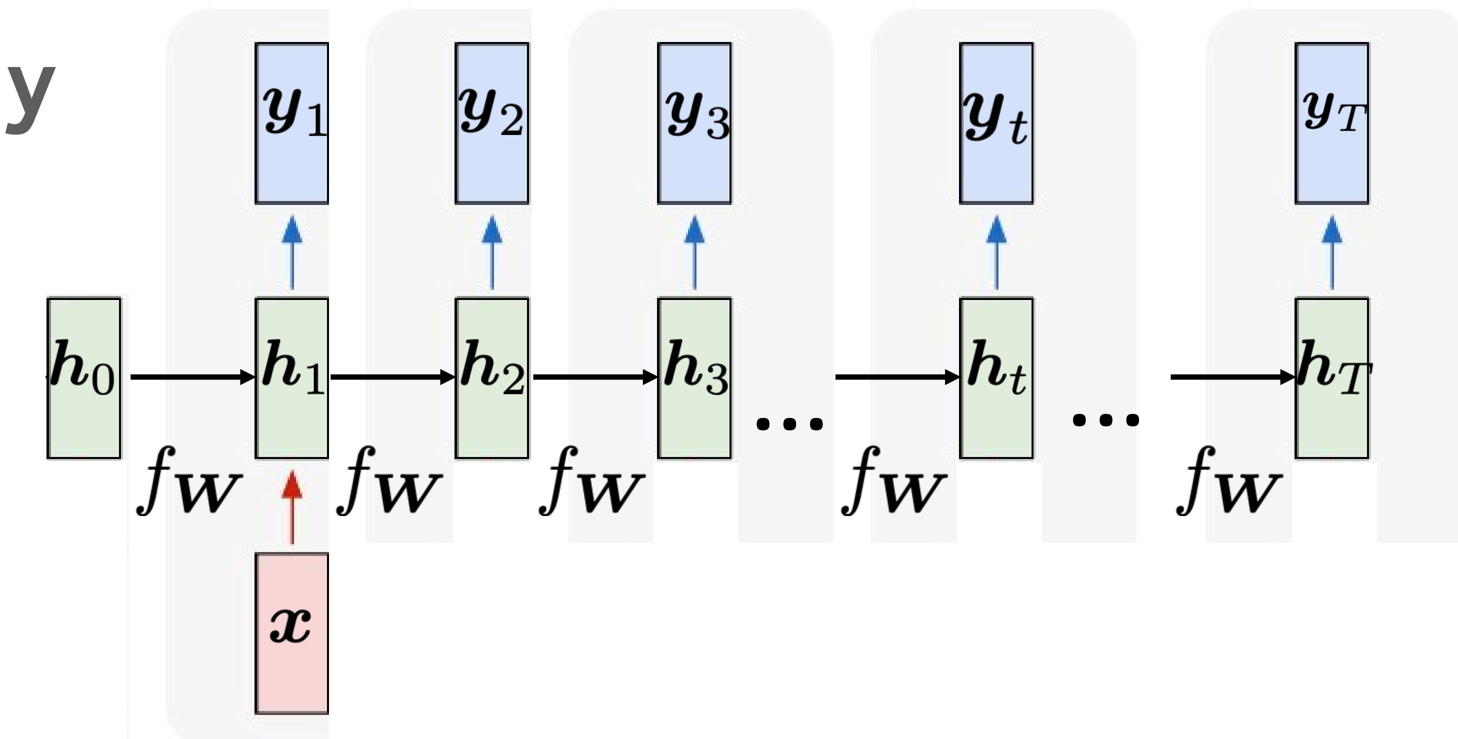


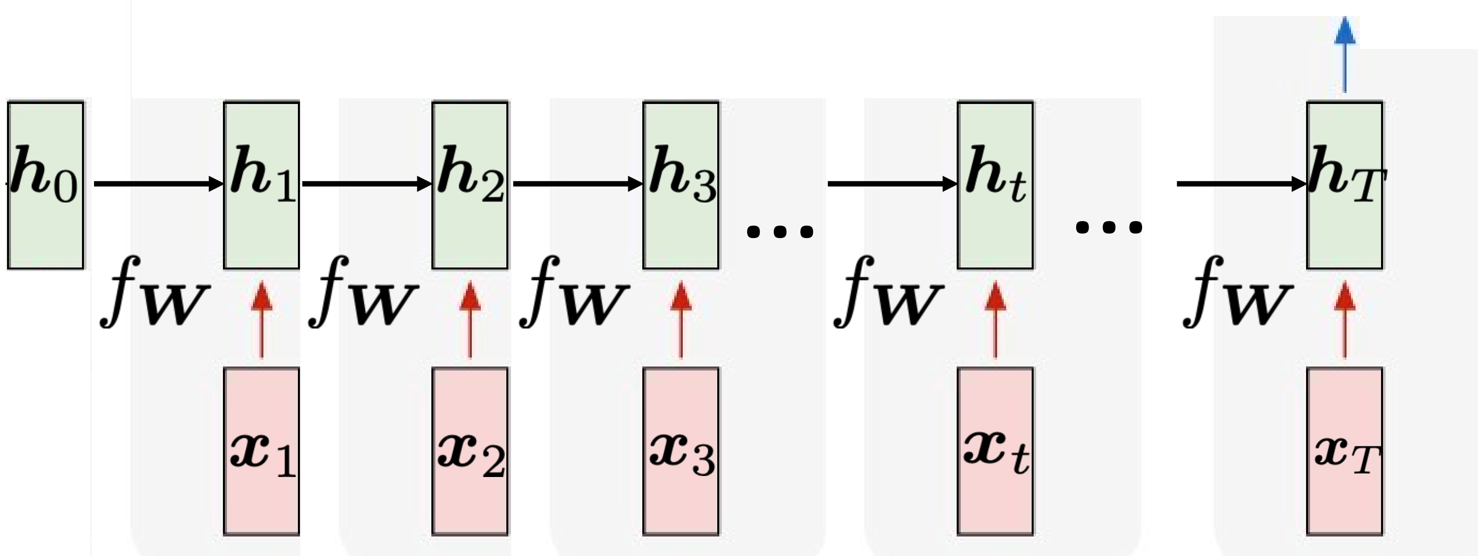
Recurrent Neural Networks

Computational Graph - Sequence to sequence: many to one + one to many

One to many:

- output sequence from input vector
- weight matrix W_2



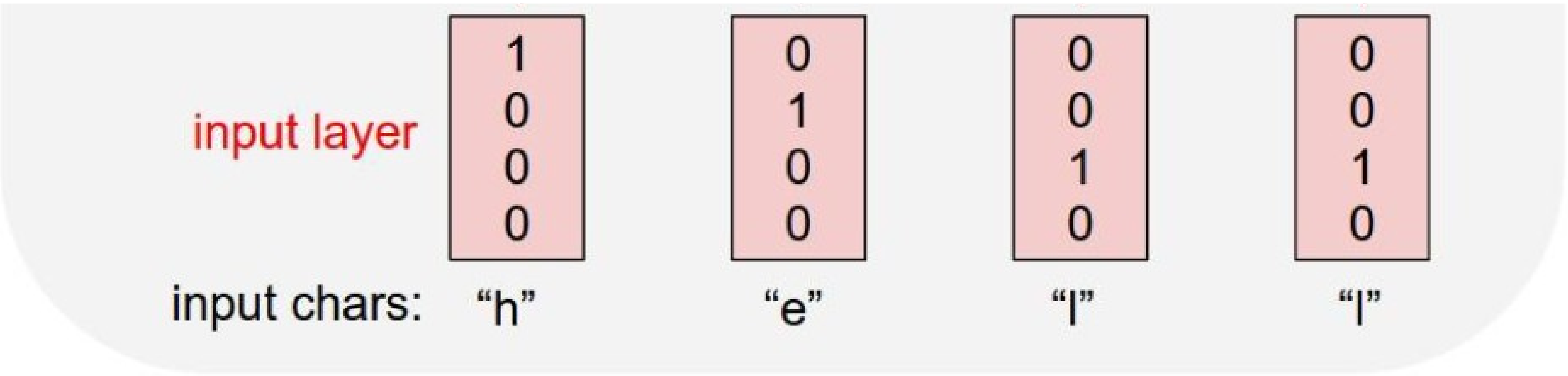


Recurrent Neural Networks

Example: Character-level Language model

- **Vocabulary:**
  [h,e,l,o]

- **Example training:**
  “hello”



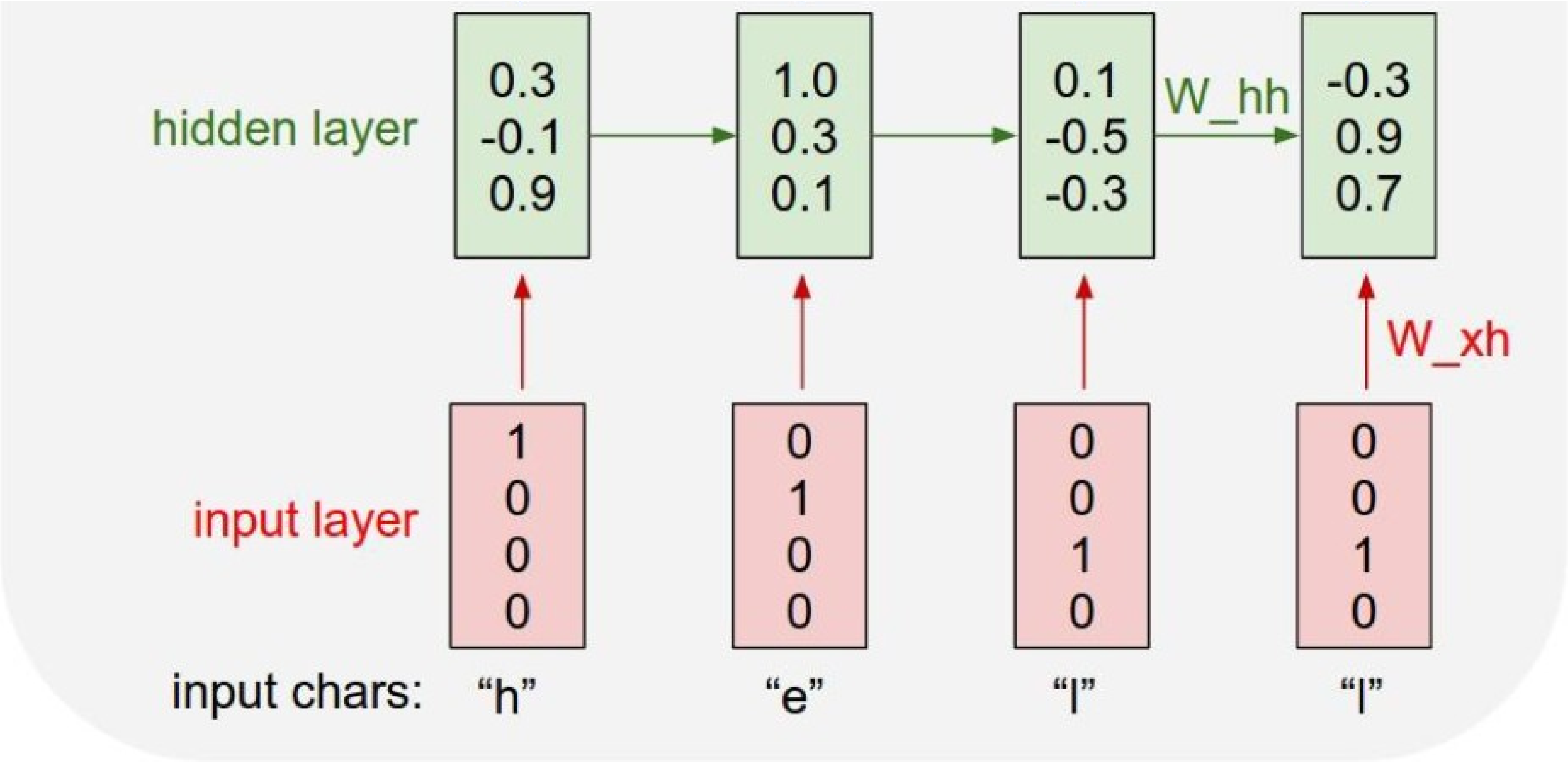
Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example: Character-level Language model

- **Vocabulary: [h,e,l,o]**
- **Example training: "hello"**

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



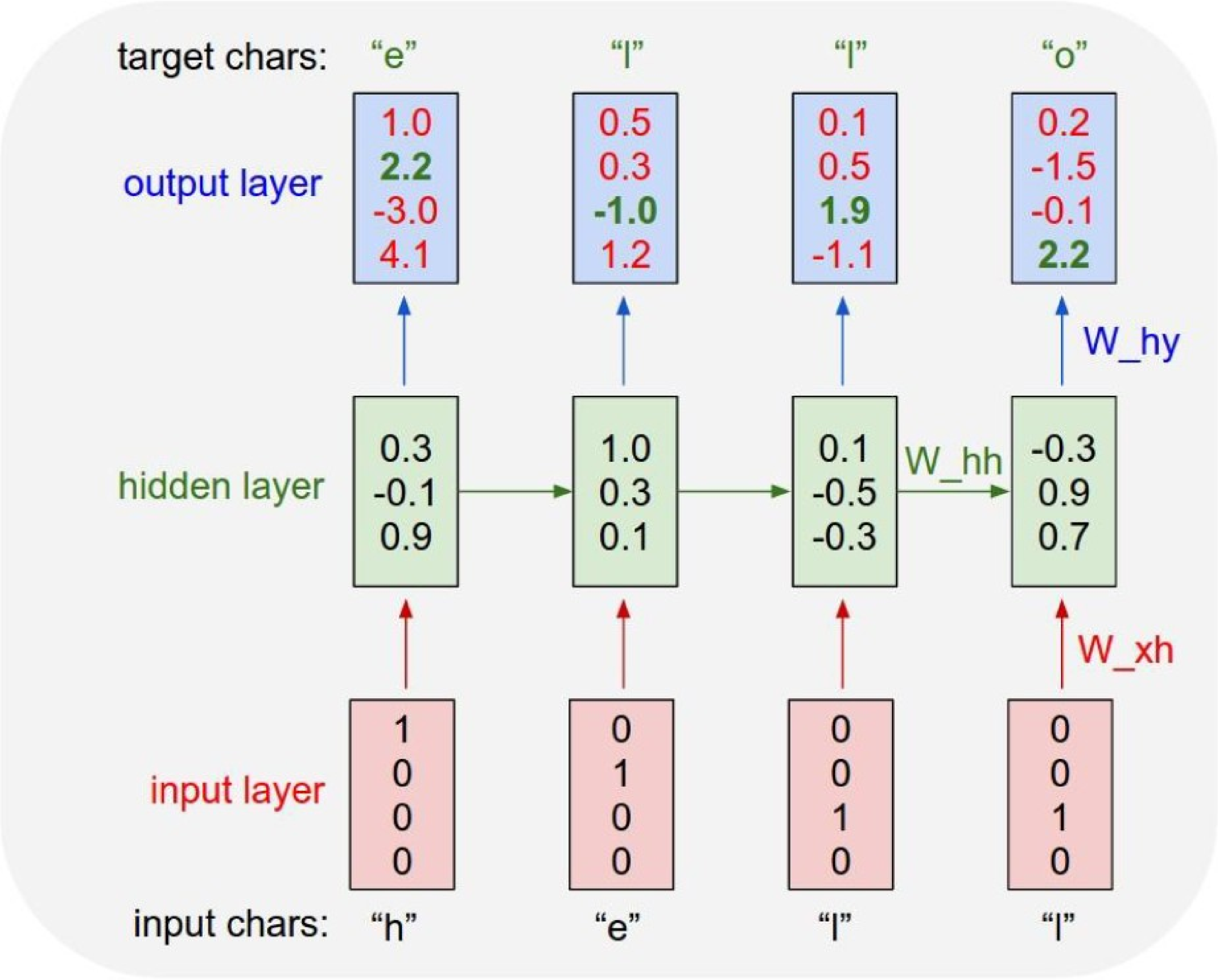
Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example: Character-level Language model

- **Vocabulary: [h,e,l,o]**
- **Example training: "hello"**

$$\boldsymbol{y}_t = \boldsymbol{W}_{hy}\boldsymbol{h}_t$$



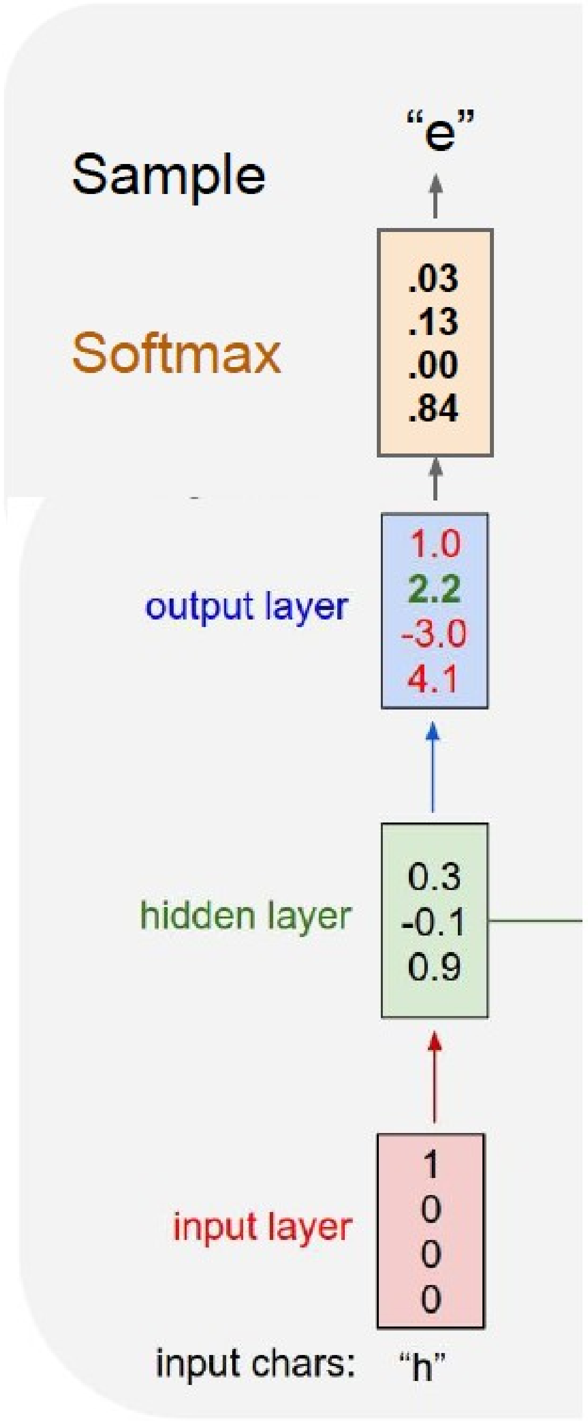
Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example: Character-level Language model

- **Vocabulary:**
  [h,e,l,o]

- **Test:**
  one character at a time

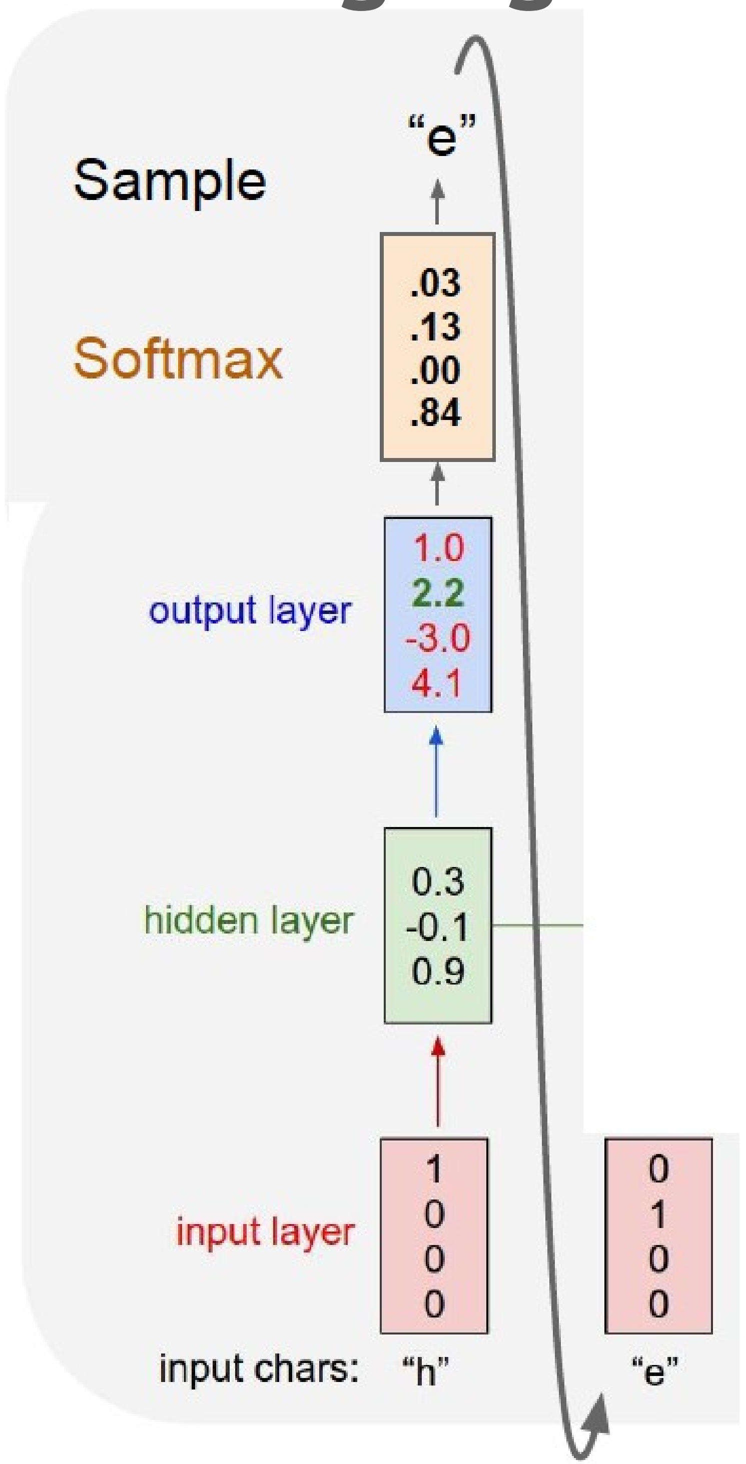


Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example: Character-level Language model

- **Vocabulary: [h,e,l,o]**
- **Test: one character at a time**



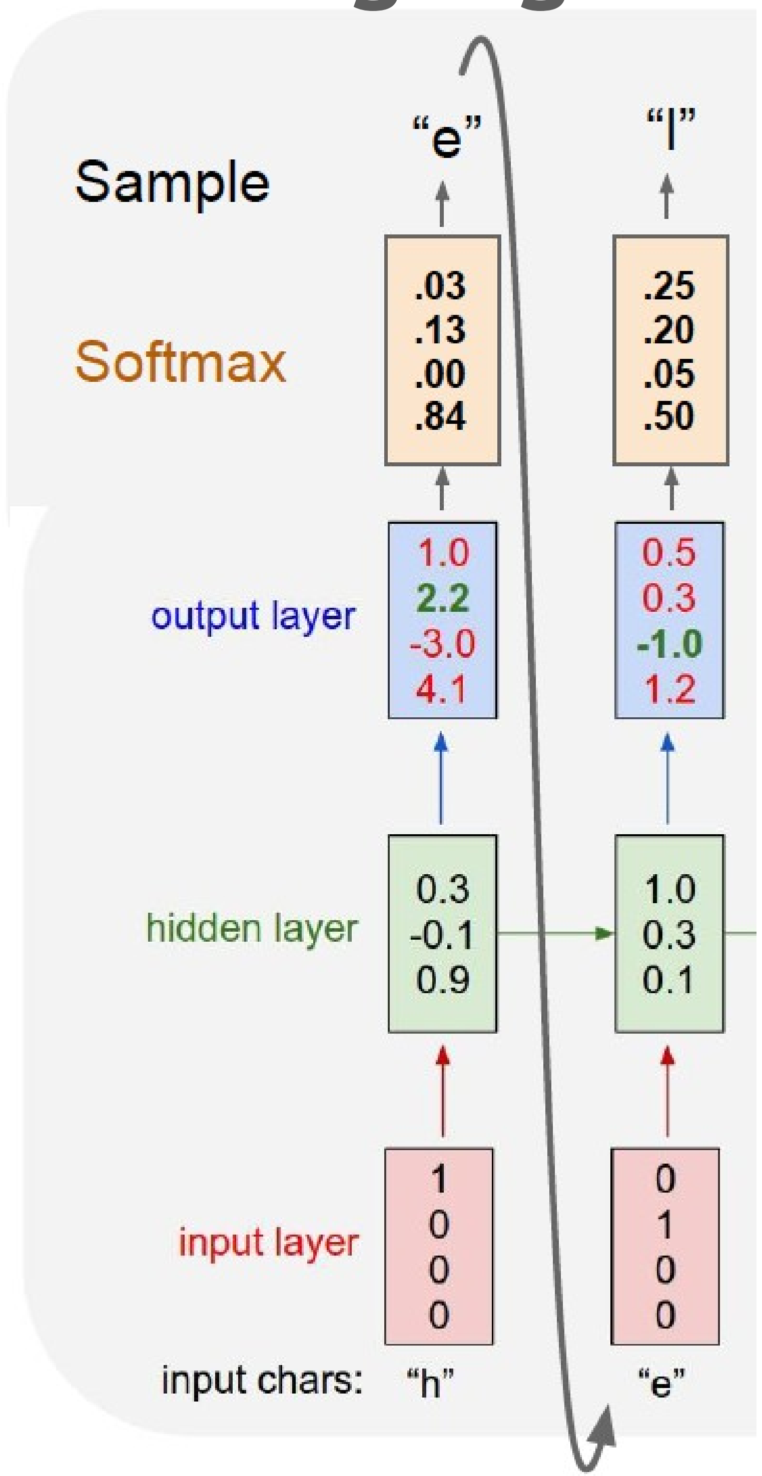
Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example: Character-level Language model

- **Vocabulary:**
  [h,e,l,o]

- **Test:**
  one character at a time

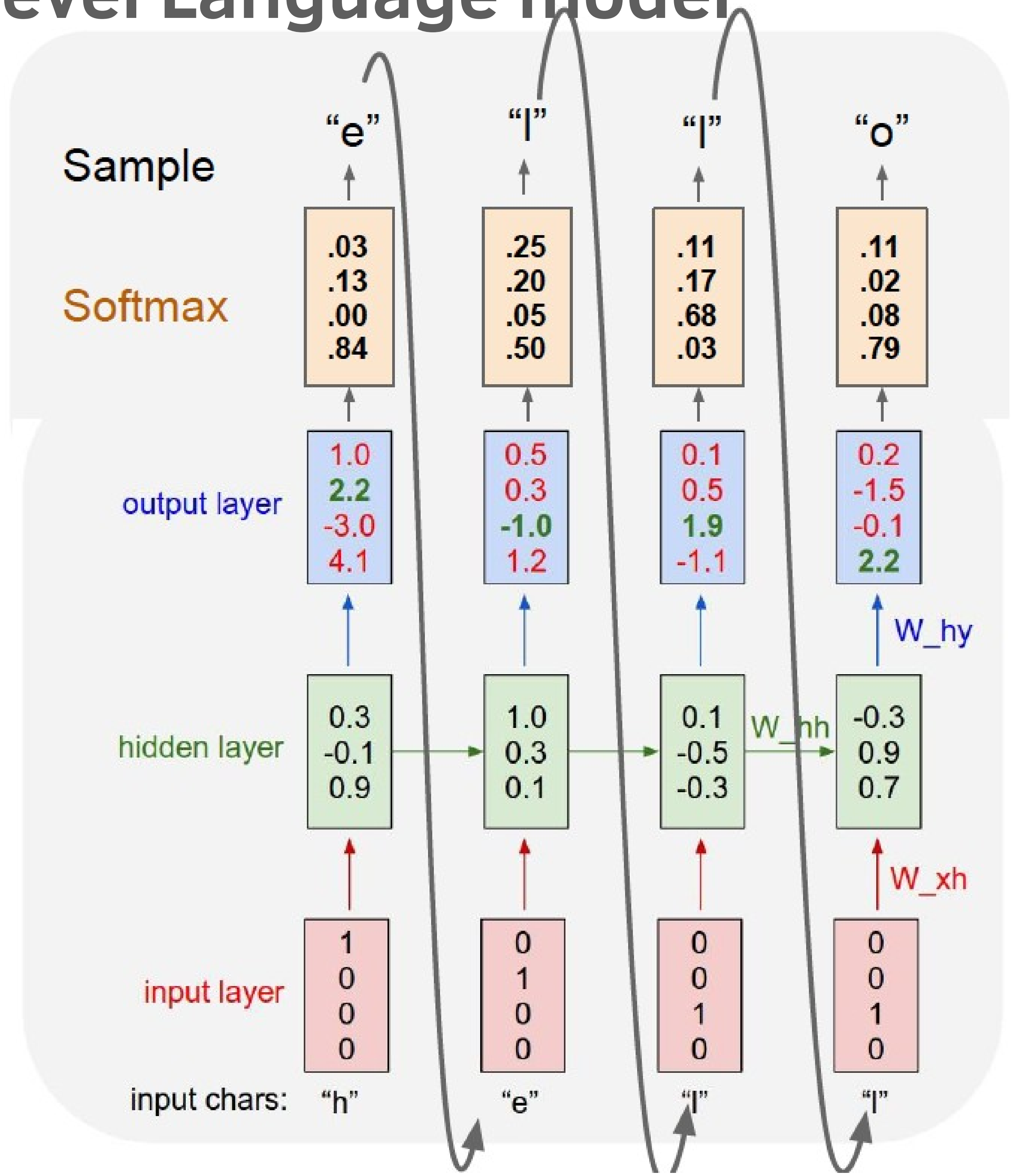


Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example: Character-level Language model

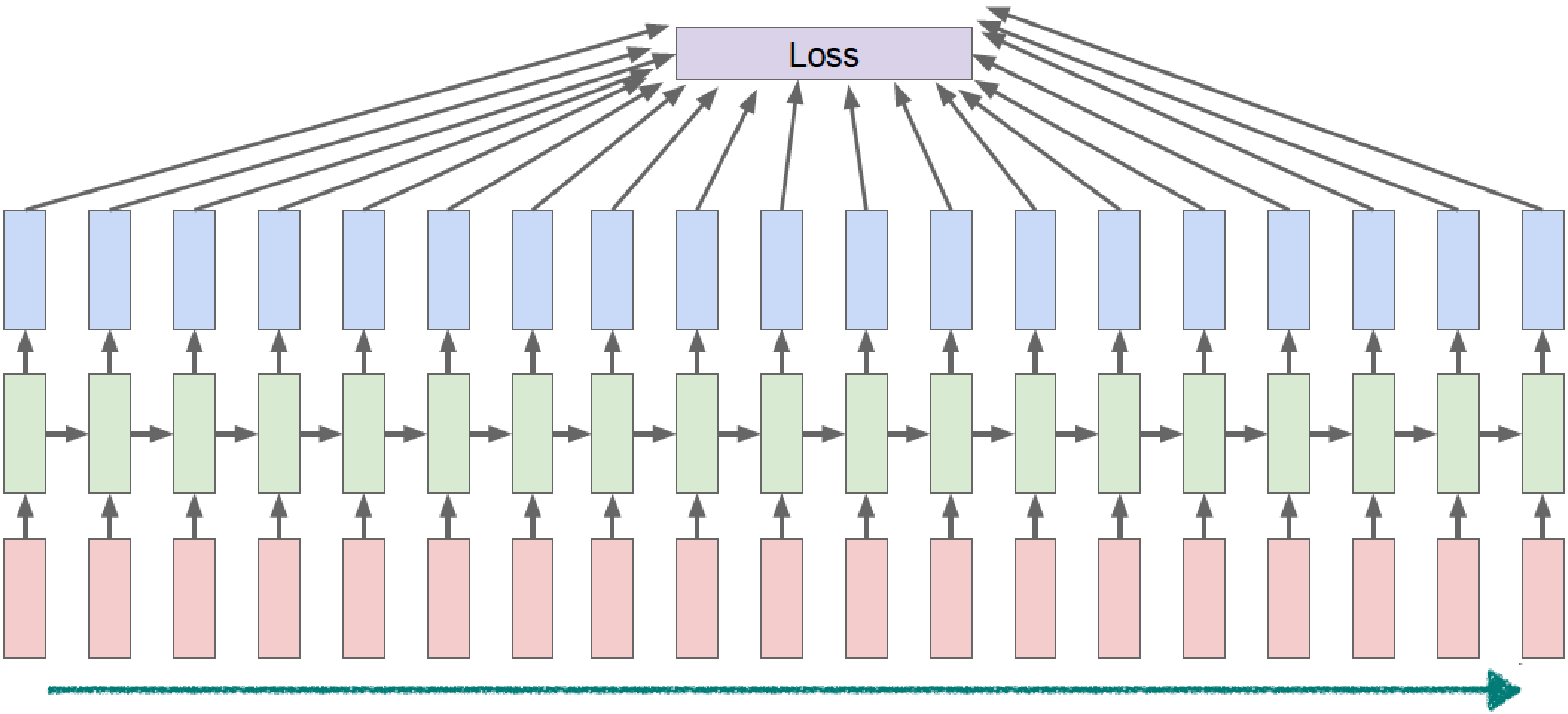
- **Vocabulary:**
  [h,e,l,o]
- **Test:**
  one character at a time



Source: Li,Johnson, Yeung

Recurrent Neural Networks

Learning of the weights - back-propagation through time

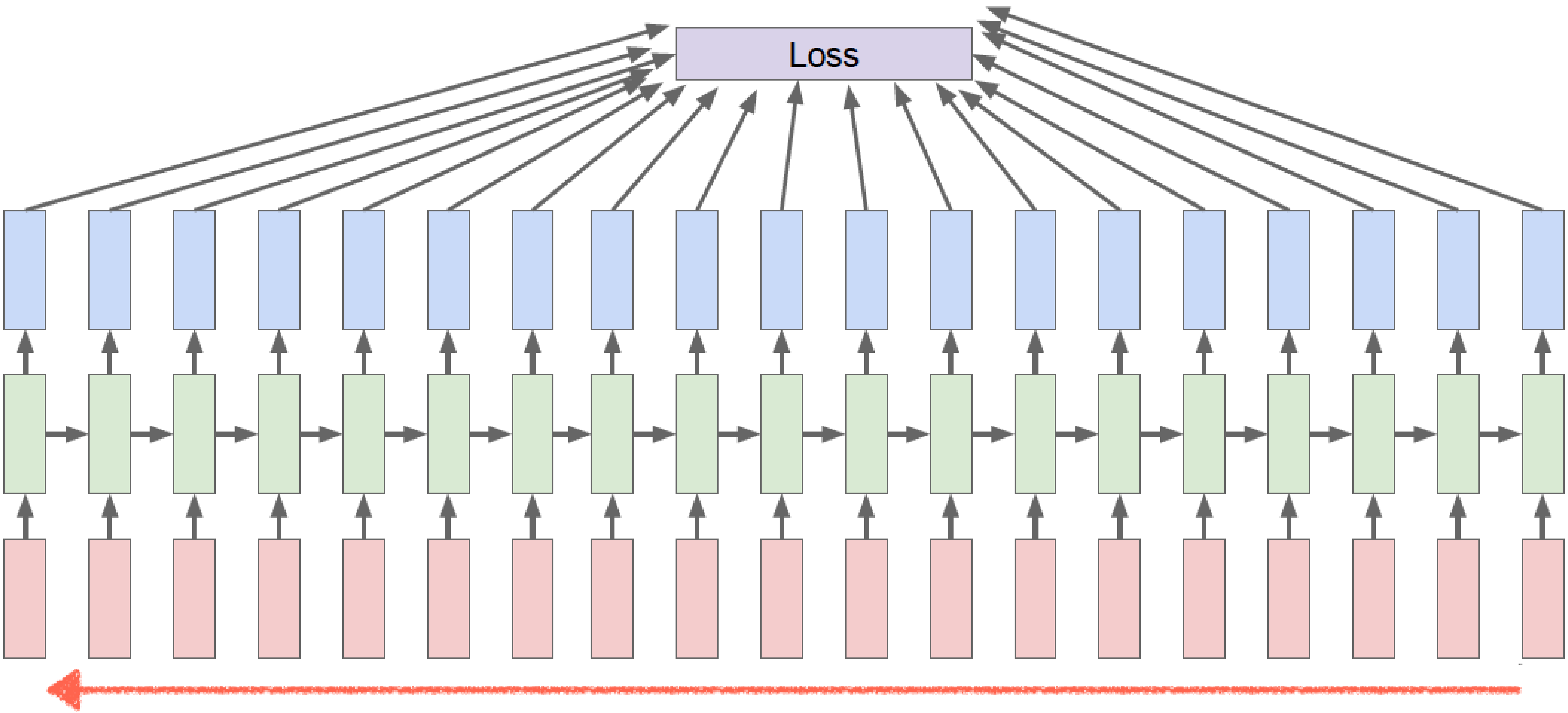


Forward through all the sequence to compute the loss

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Learning of the weights - back-propagation through time

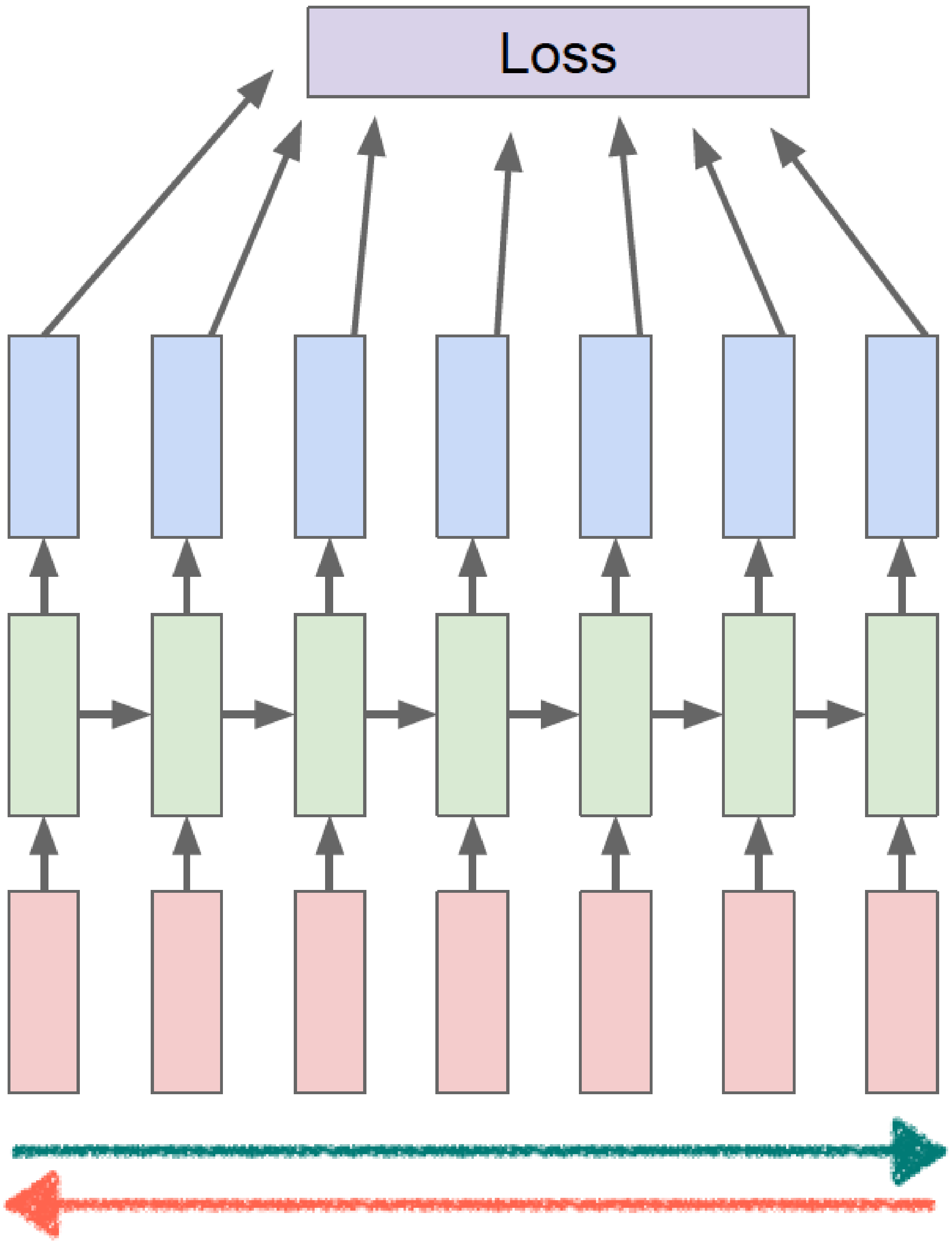


Backward through all the sequence to compute the gradient

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Truncated back-propagation through time

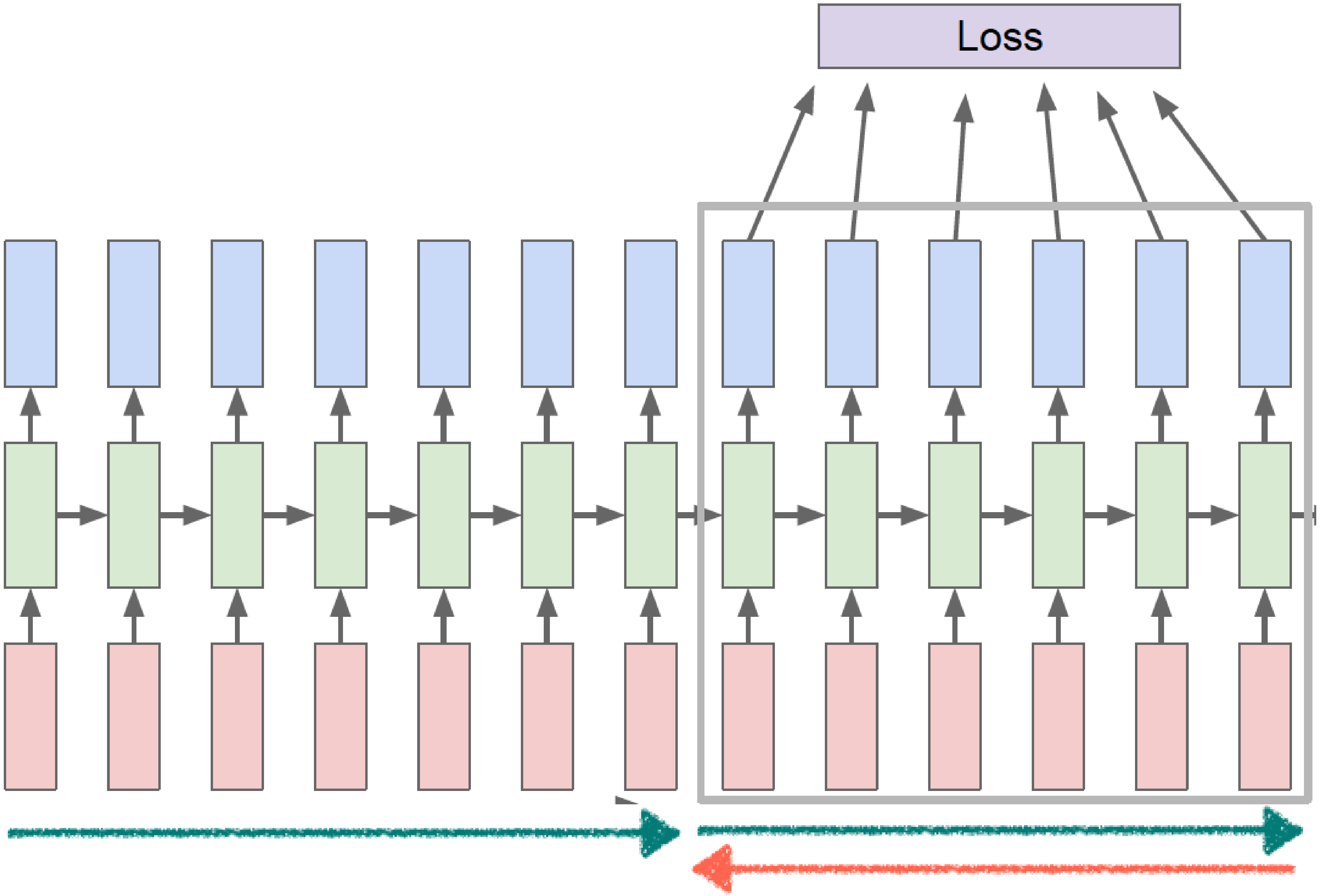


Run forward and backward through chunks of the sequence

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Truncated back-propagation through time

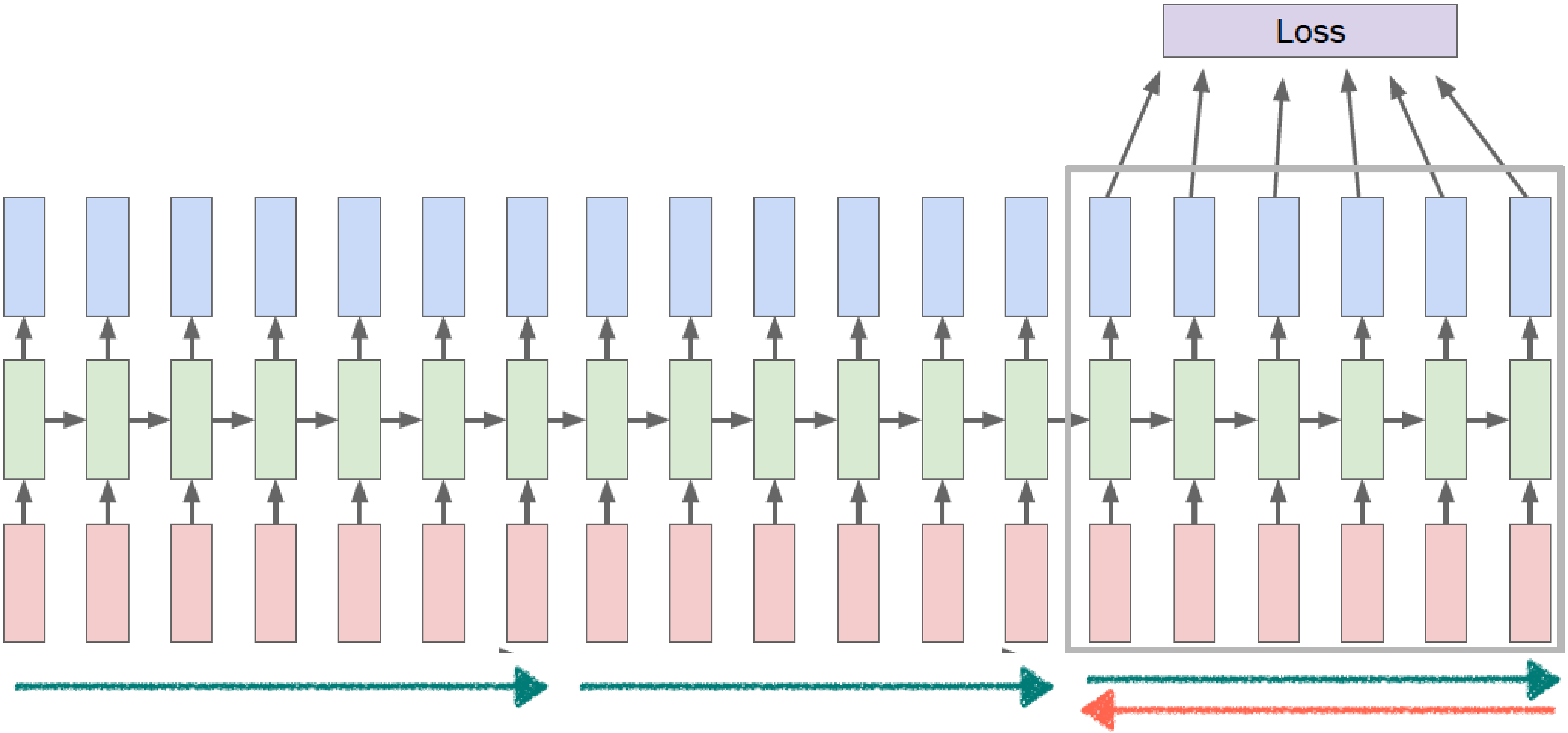


Carry hidden states forward in time, but only back-propagate for some smaller number of steps

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Truncated back-propagation through time



Carry hidden states forward in time, but only back-propagate for some smaller number of steps

Source: Li,Johnson, Yeung

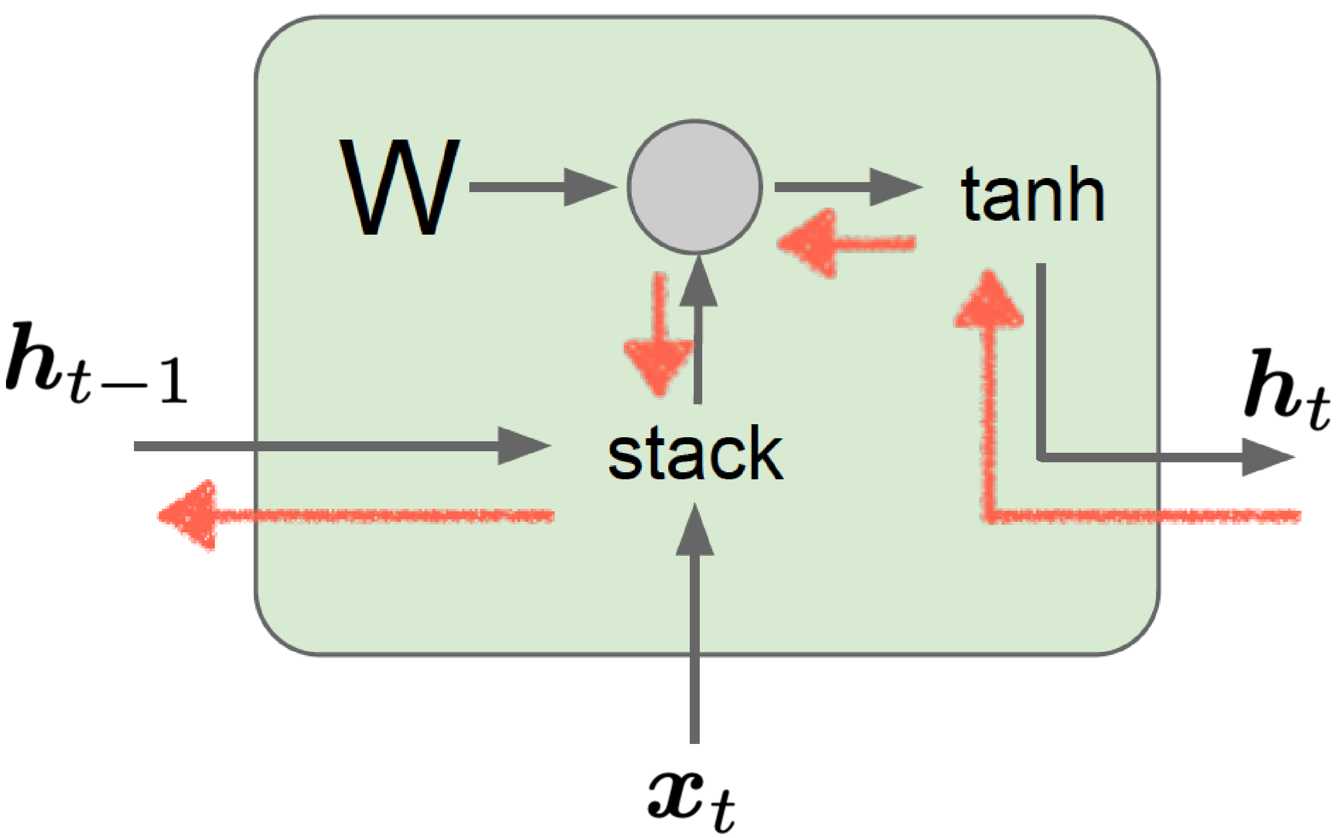
Recurrent Neural Networks

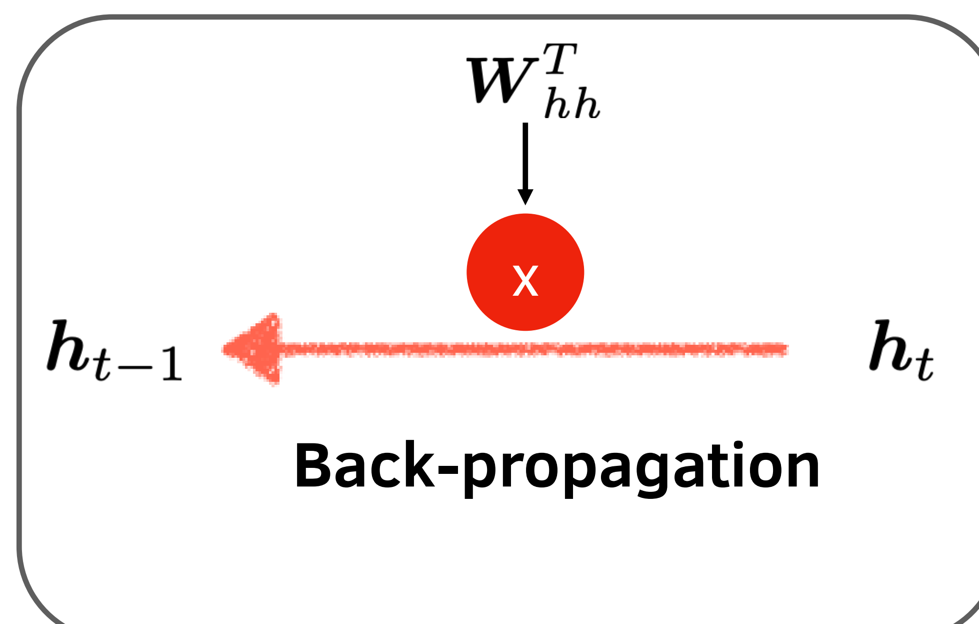
Backpropagation - Gradient flow

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

$$h_t = tanh\left(\begin{bmatrix} W_{hh} & W_{hx} \end{bmatrix} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)$$

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

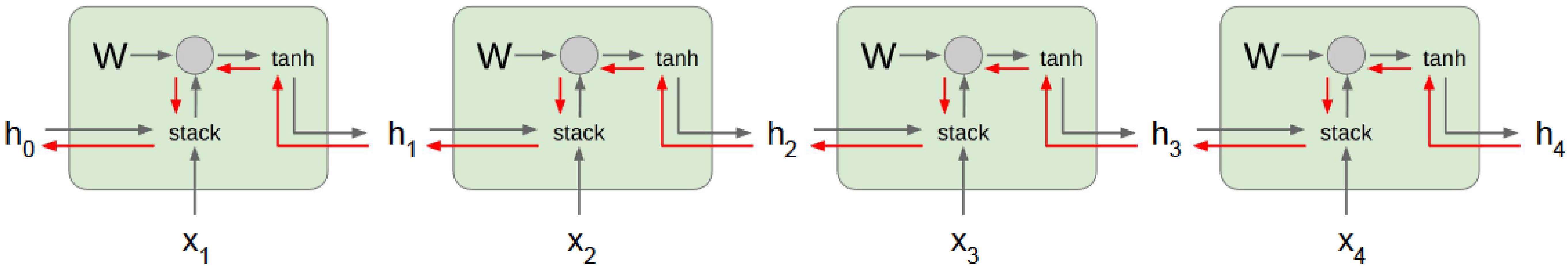




Source: Li,Johnson, Yeung

Recurrent Neural Networks

Backpropagation - Gradient flow



Computing gradient of h0 involves many factors of W and repeated tanh

✳ **Exploding gradients** if largest singular value > 1

- **SOLUTION:** gradient clipping with a threshold

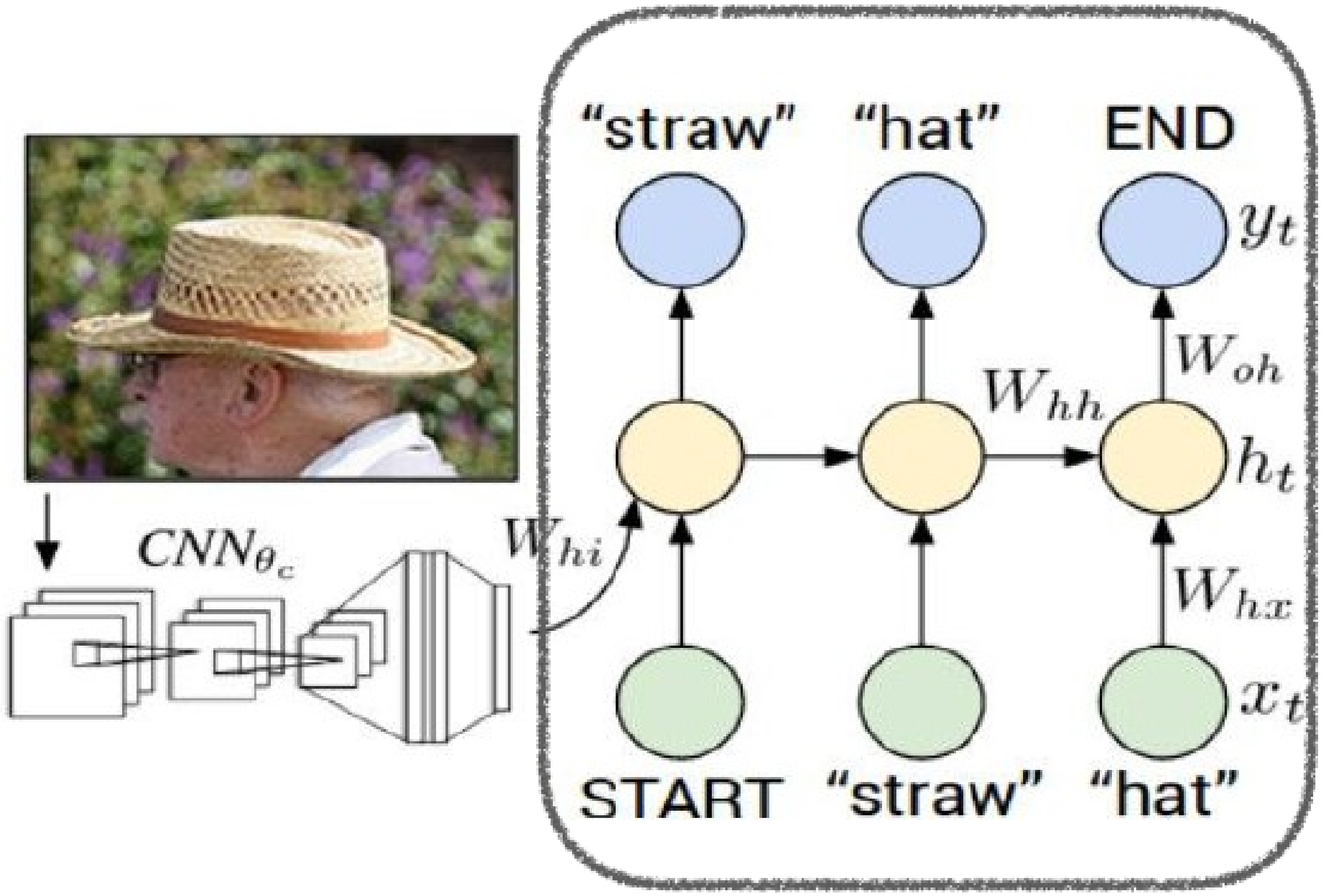
✳ **Vanishing gradient** if largest singular value <1

- **SOLUTION:** change RNN architecture

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example - Image Captioning

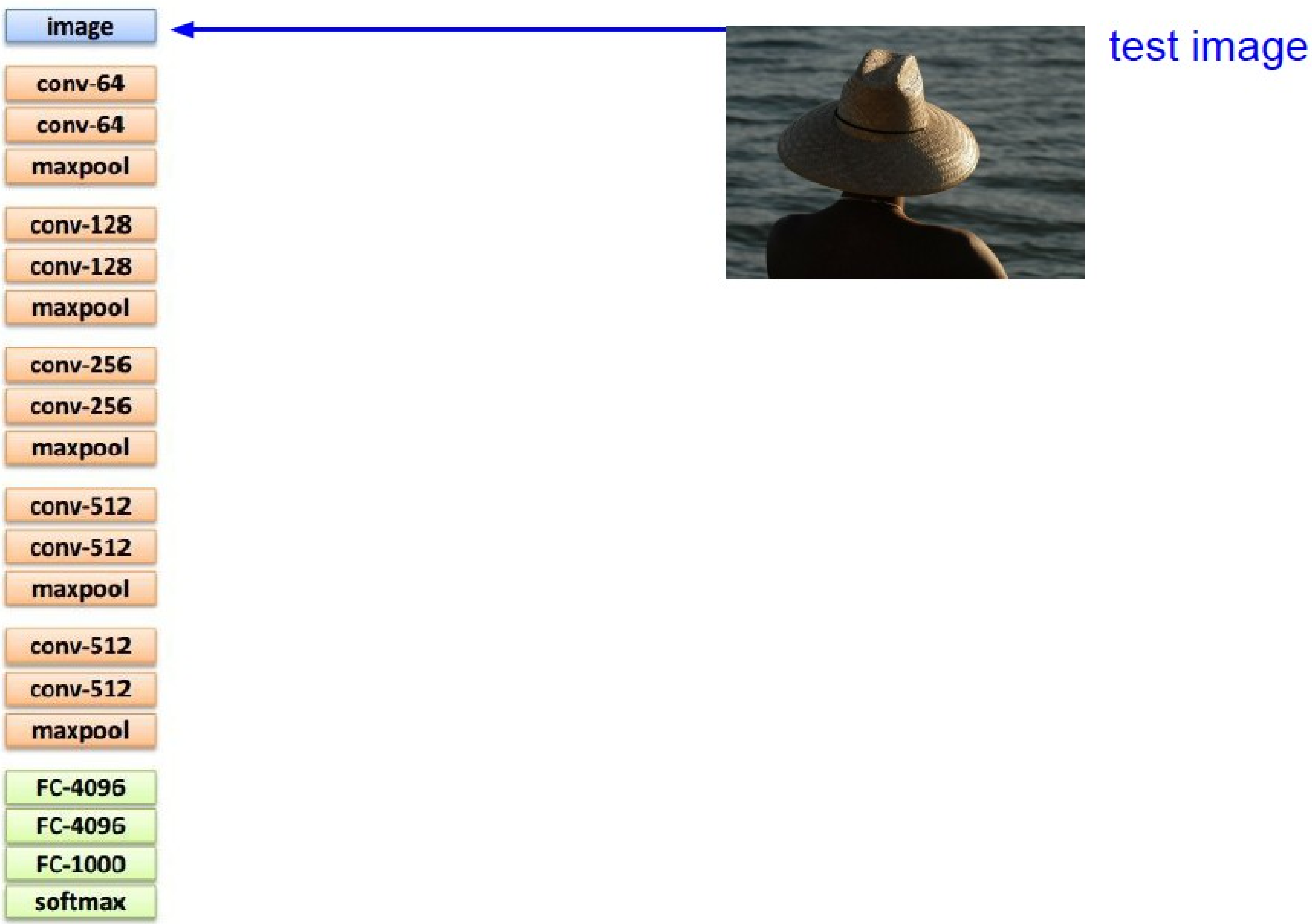


RNN

Figure: Karpathy et al.
Deep Visual Semiantic Alignments for Generating Image Descriptions.
CVPR, 2015.

Recurrent Neural Networks

Example - Image Captioning

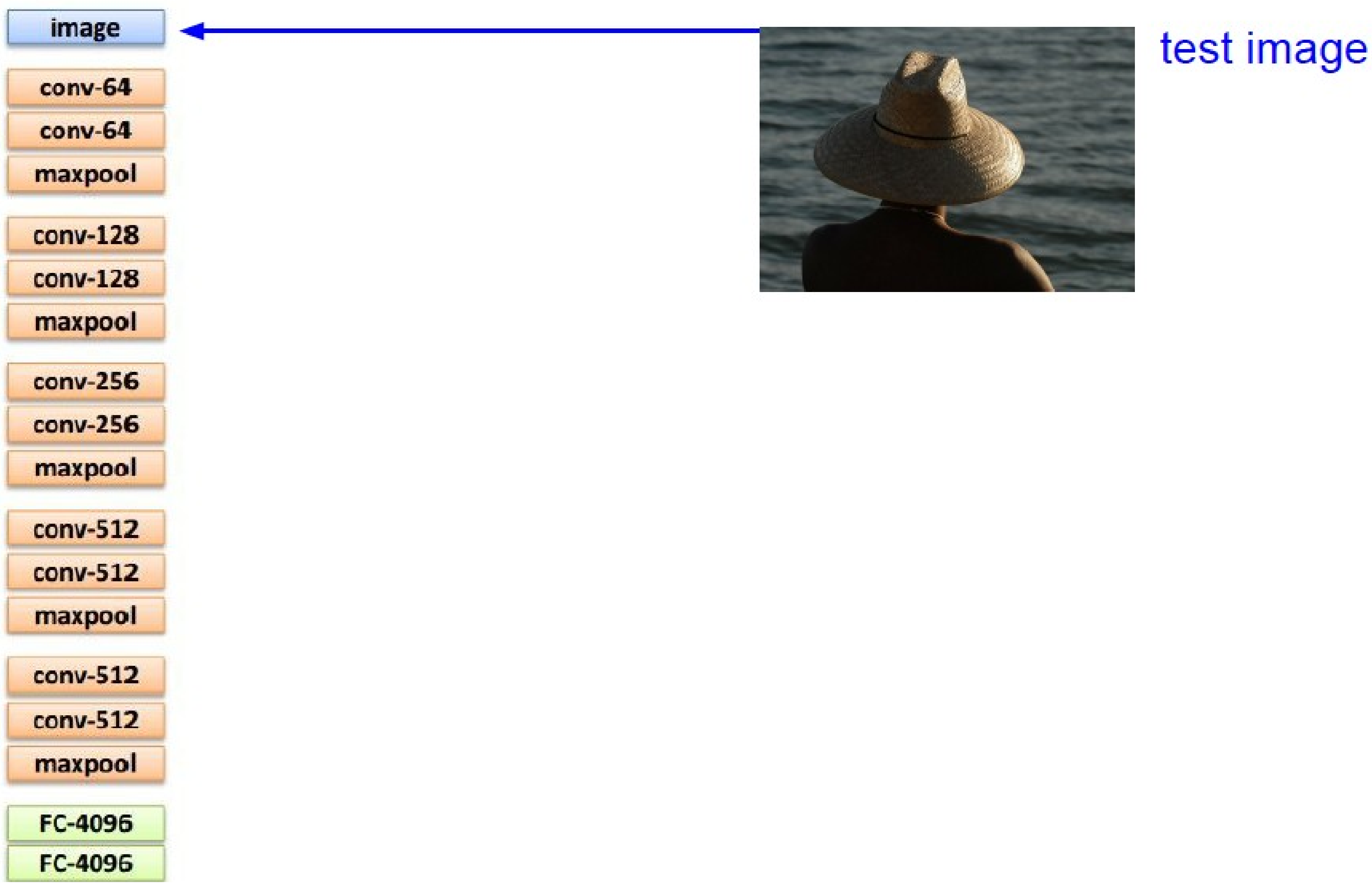


CNN trained on ImageNet

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example - Image Captioning

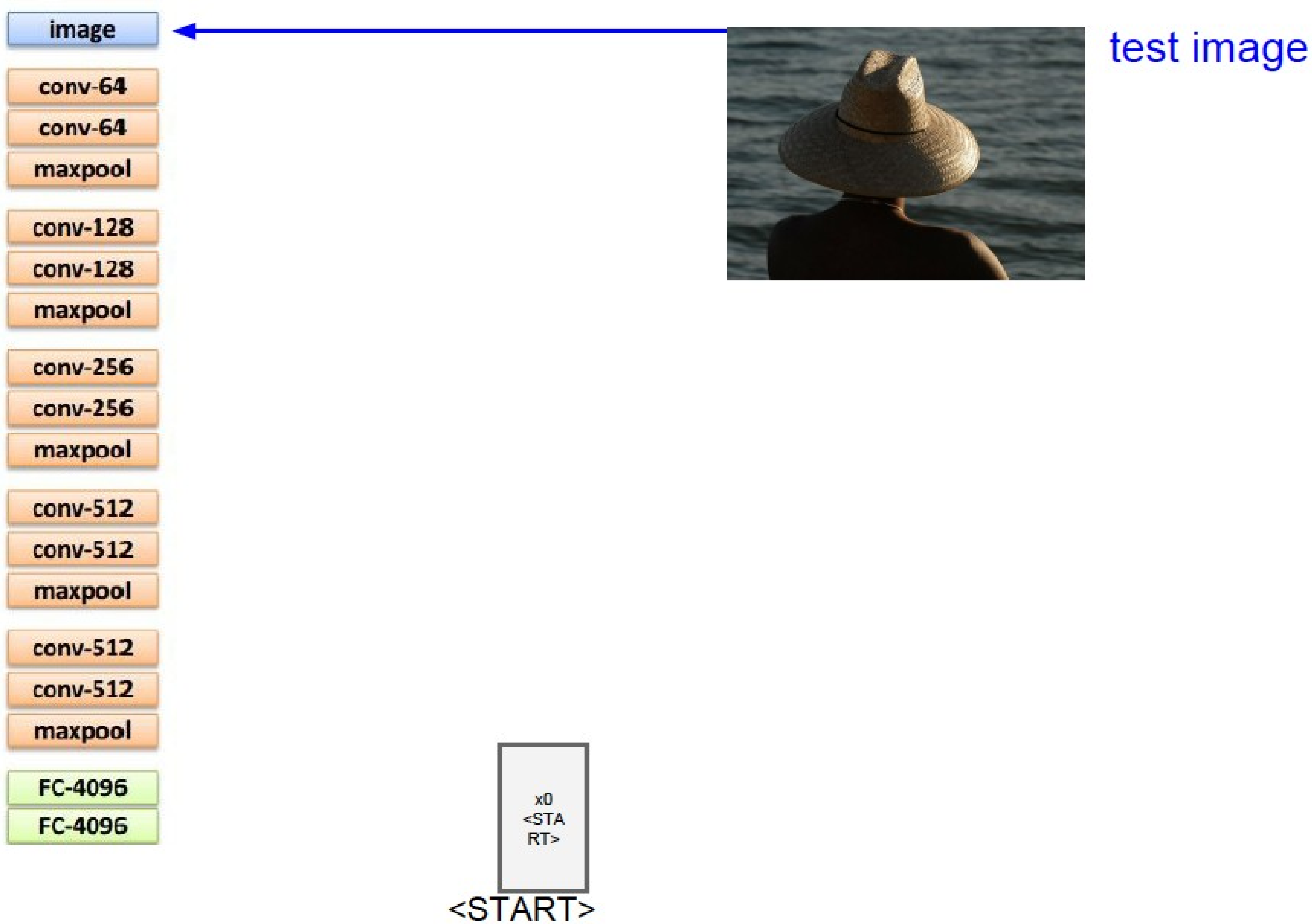


Take features before the lass FC layer

Source: Li,Johnson, Yeung

Recurrent Neural Networks

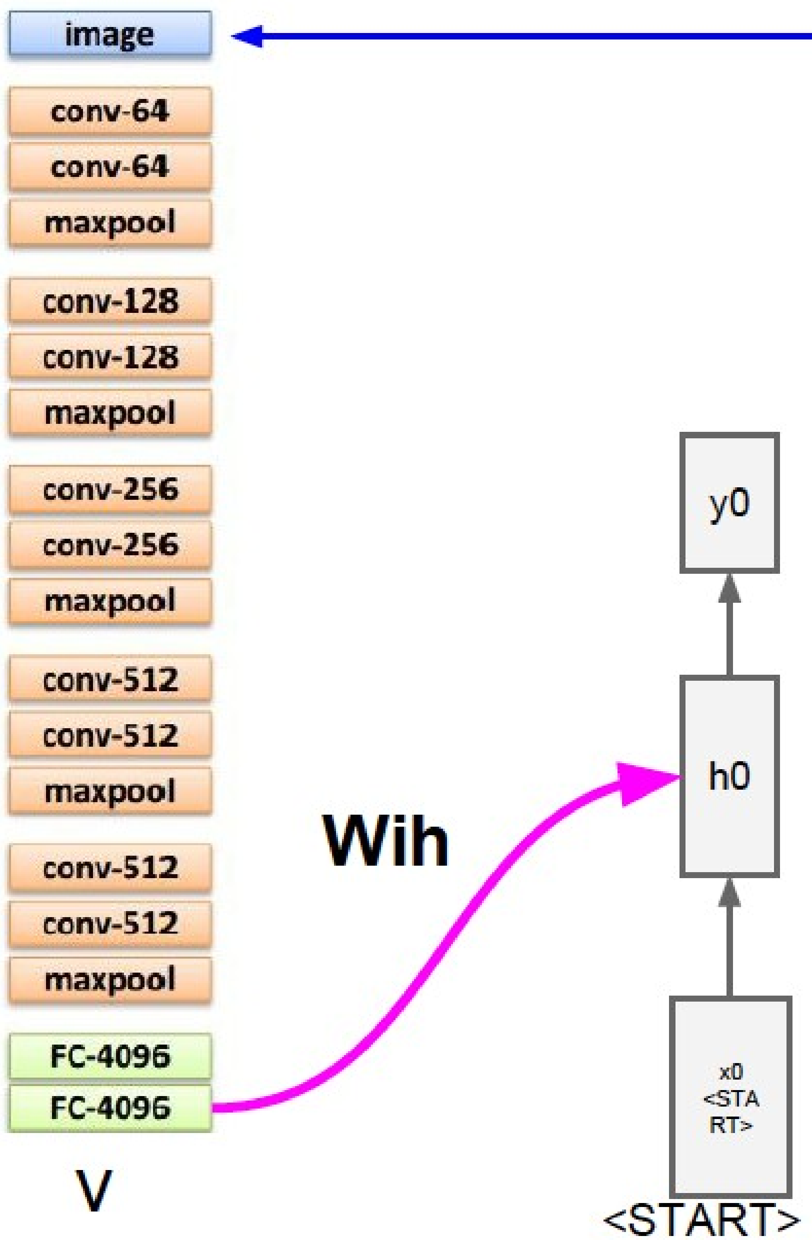
Example - Image Captioning



Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example - Image Captioning



test image

before:

$h = \tanh(Wxh * x + Whh * h)$

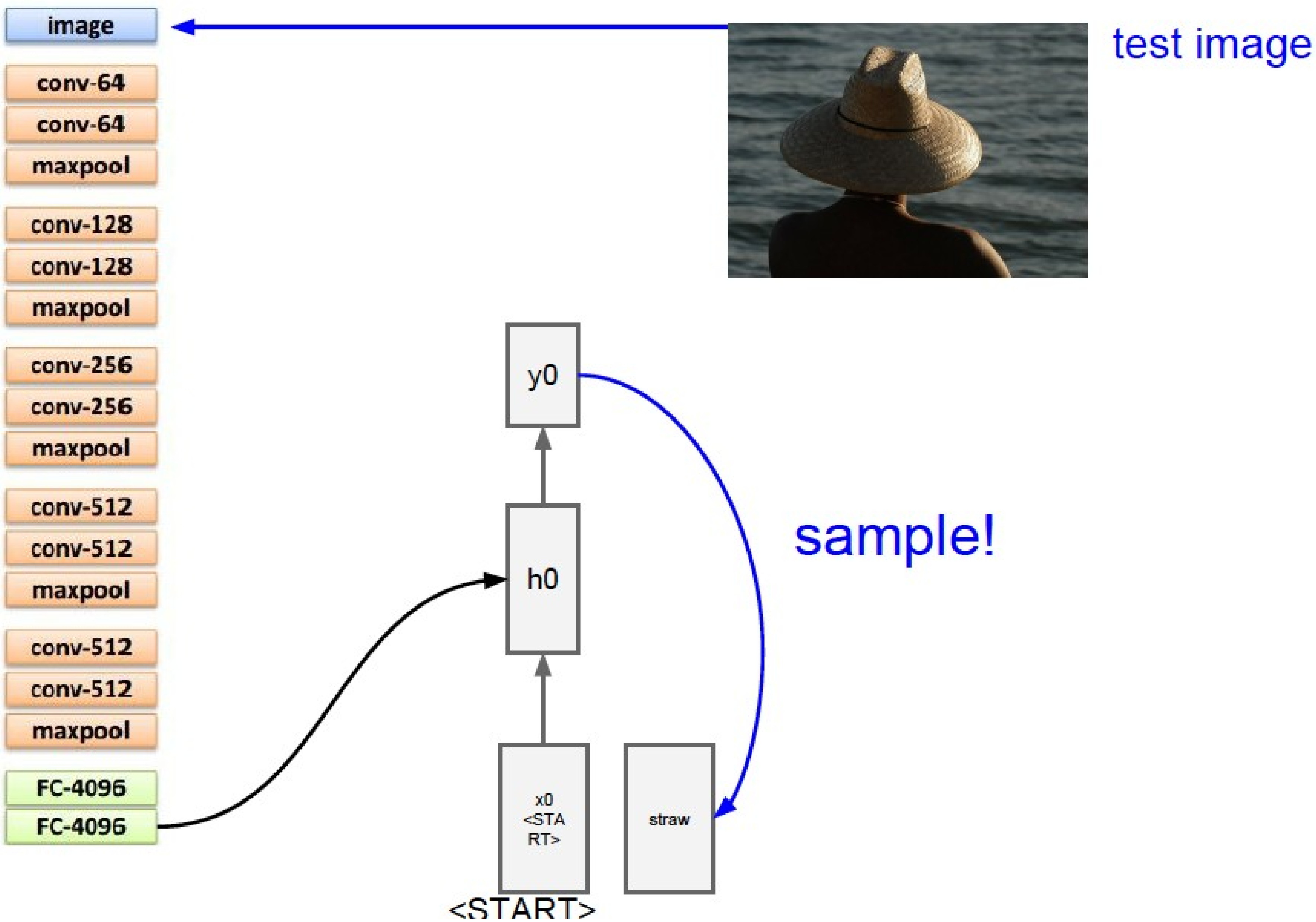
now:

$h = \tanh(Wxh * x + Whh * h + Wih * v)$

Source: Li,Johnson, Yeung

Recurrent Neural Networks

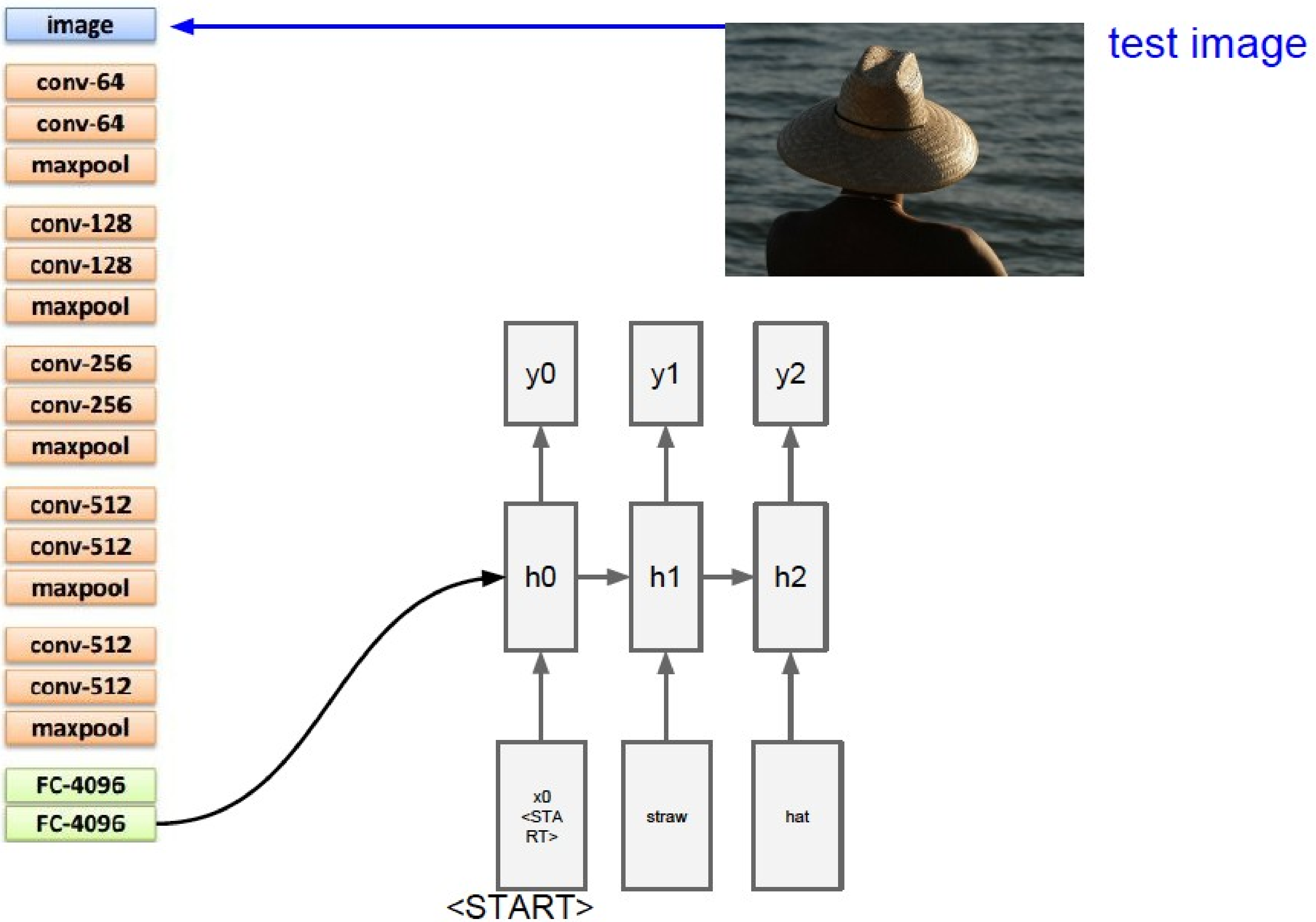
Example - Image Captioning



Source: Li,Johnson, Yeung

Recurrent Neural Networks

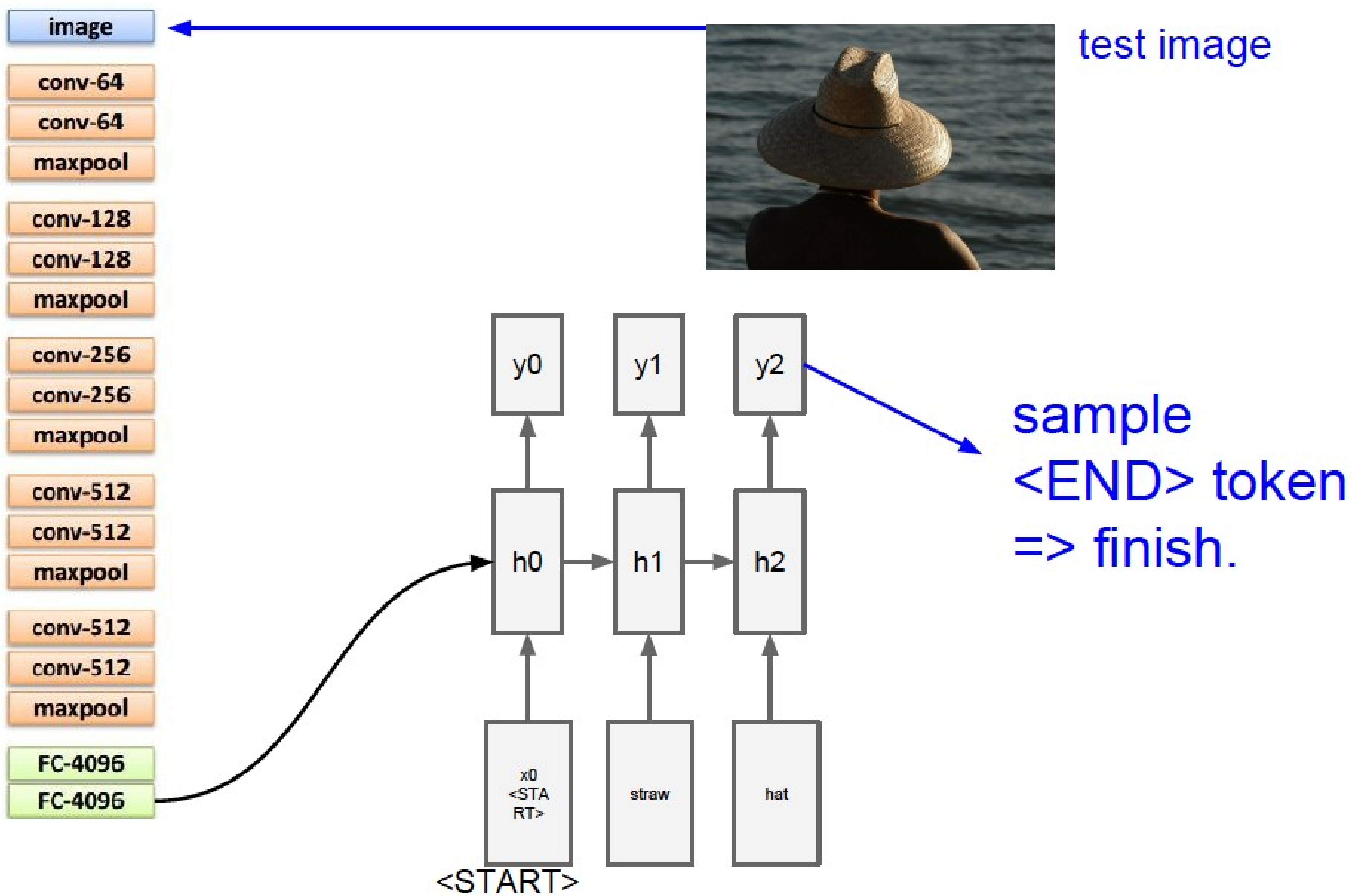
Example - Image Captioning



Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example - Image Captioning



Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example - Image Captioning

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

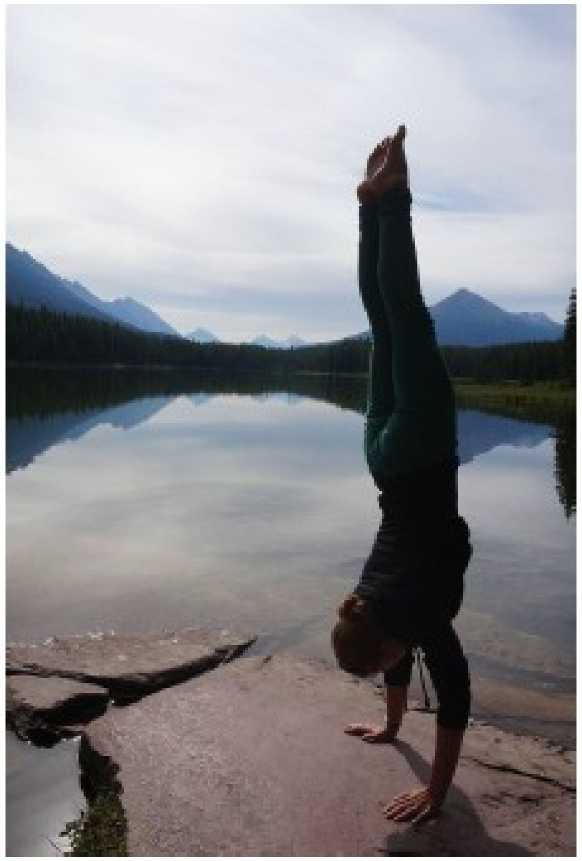
Source: Li,Johnson, Yeung

Recurrent Neural Networks

Example - Image Captioning (one to many)

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Failure results

Source: Li,Johnson, Yeung

Recurrent Neural Networks

Deep RNN

- Multiple layer RNN

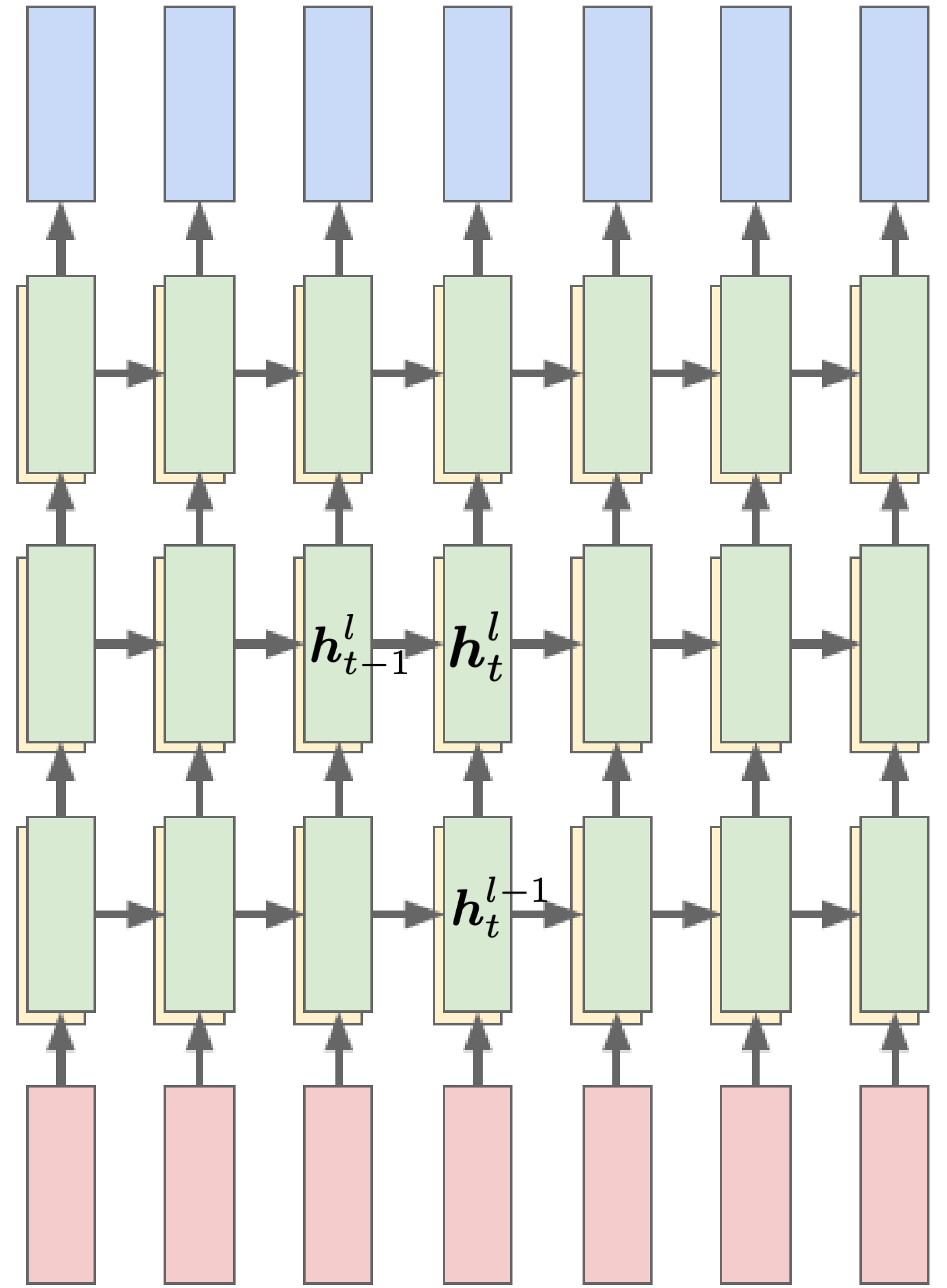
$$\boldsymbol{h}_t^l = tanh \boldsymbol{W}^l [\boldsymbol{h}_t^{l-1} \ \boldsymbol{h}_{t-1}^l]^T$$

$$\boldsymbol{h} \in \mathcal{R}^n$$

$$\boldsymbol{W}^l \in \mathcal{R}^{n \times 2n}$$

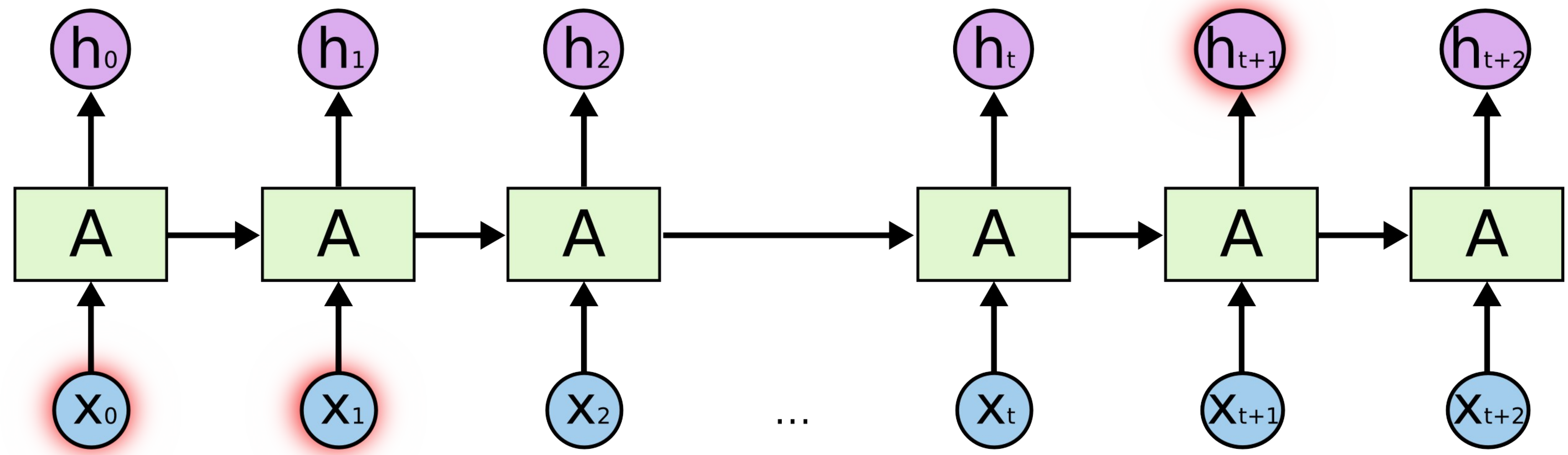
- Recall for one layer RNN:

$$\boldsymbol{h}_t = tanh(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t)$$



Source: Li,Johnson, Yeung

Recurrent Neural Networks

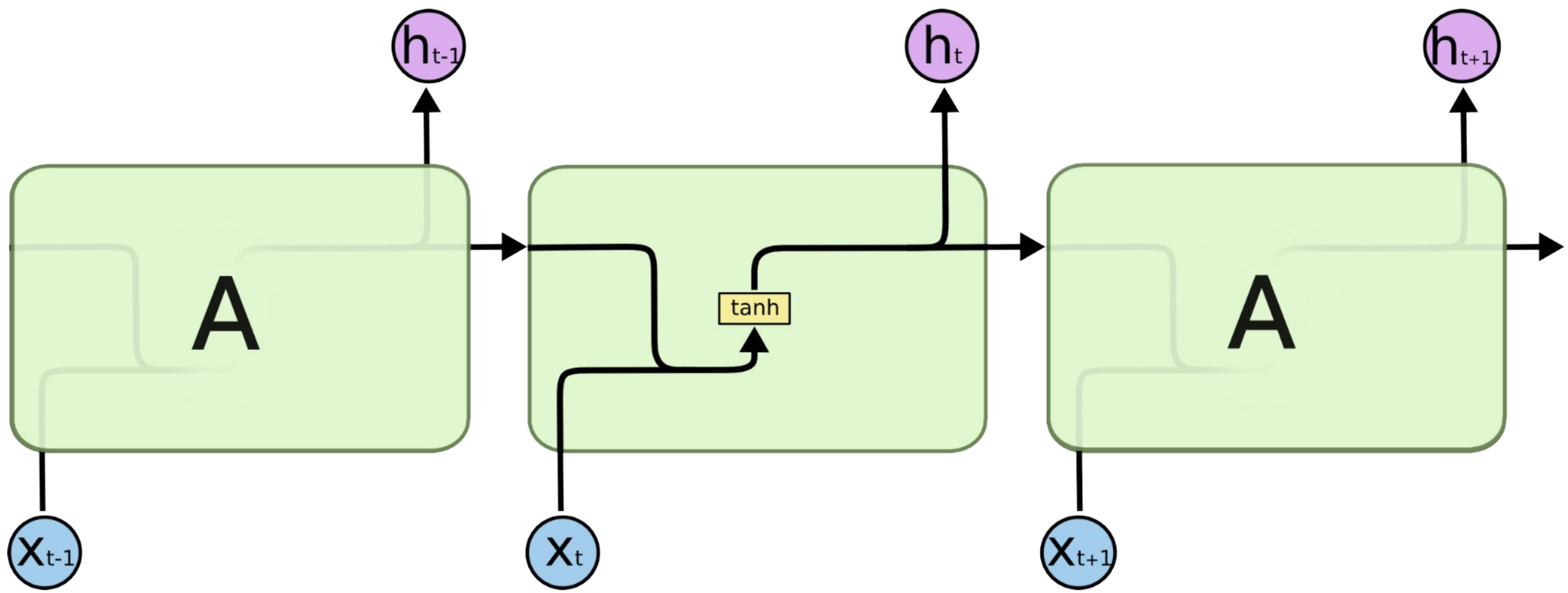


A -> RNN
h -> state
x -> input sequence

Long-term dependencies are hard to model!

Credit: Christopher Olah

Recurrent Neural Networks



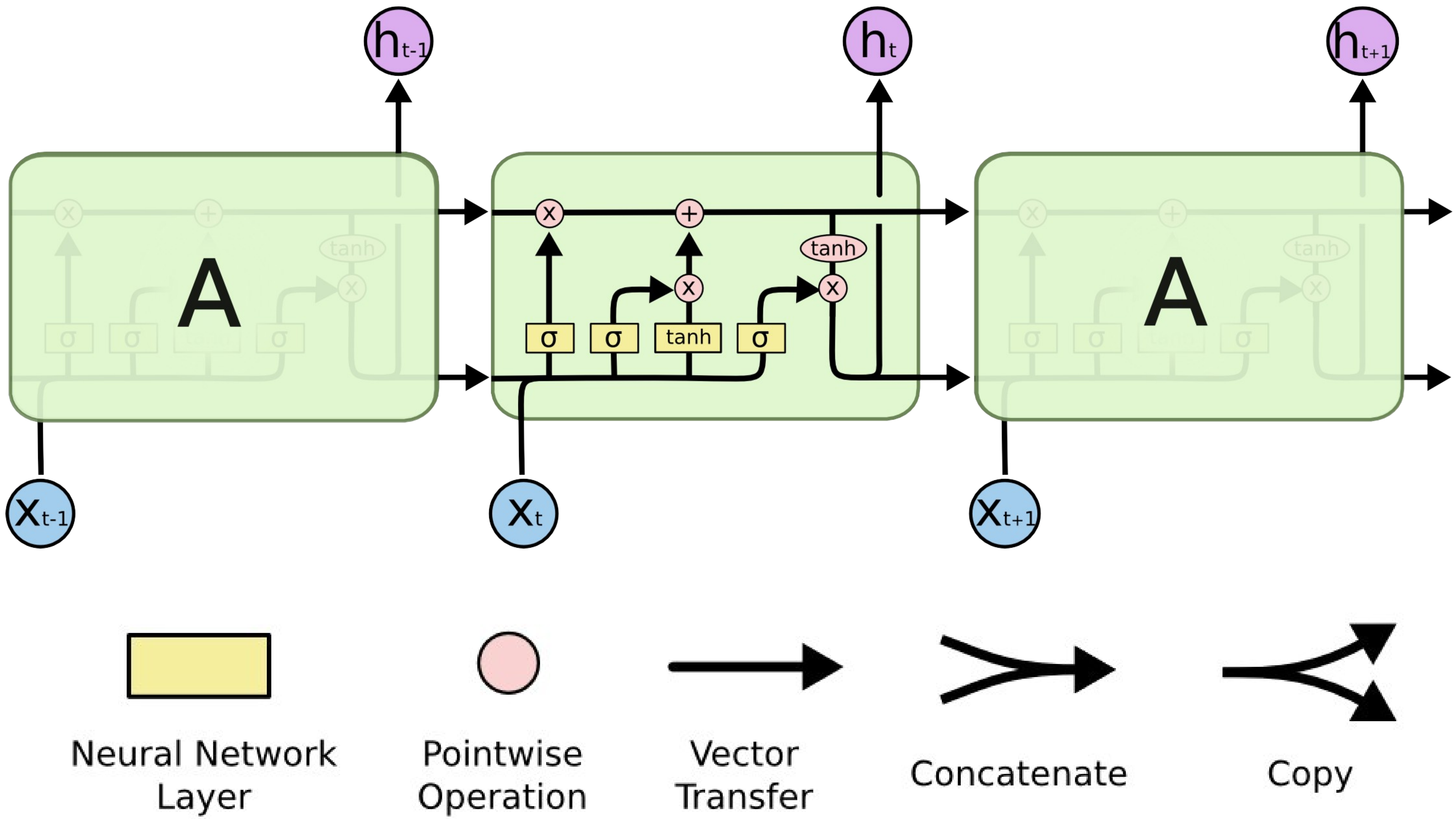
A -> RNN
h -> state
x -> input sequence

Plain RNN - what we have seen so far

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Credit: Christopher Olah

From plain RNN to LSTM

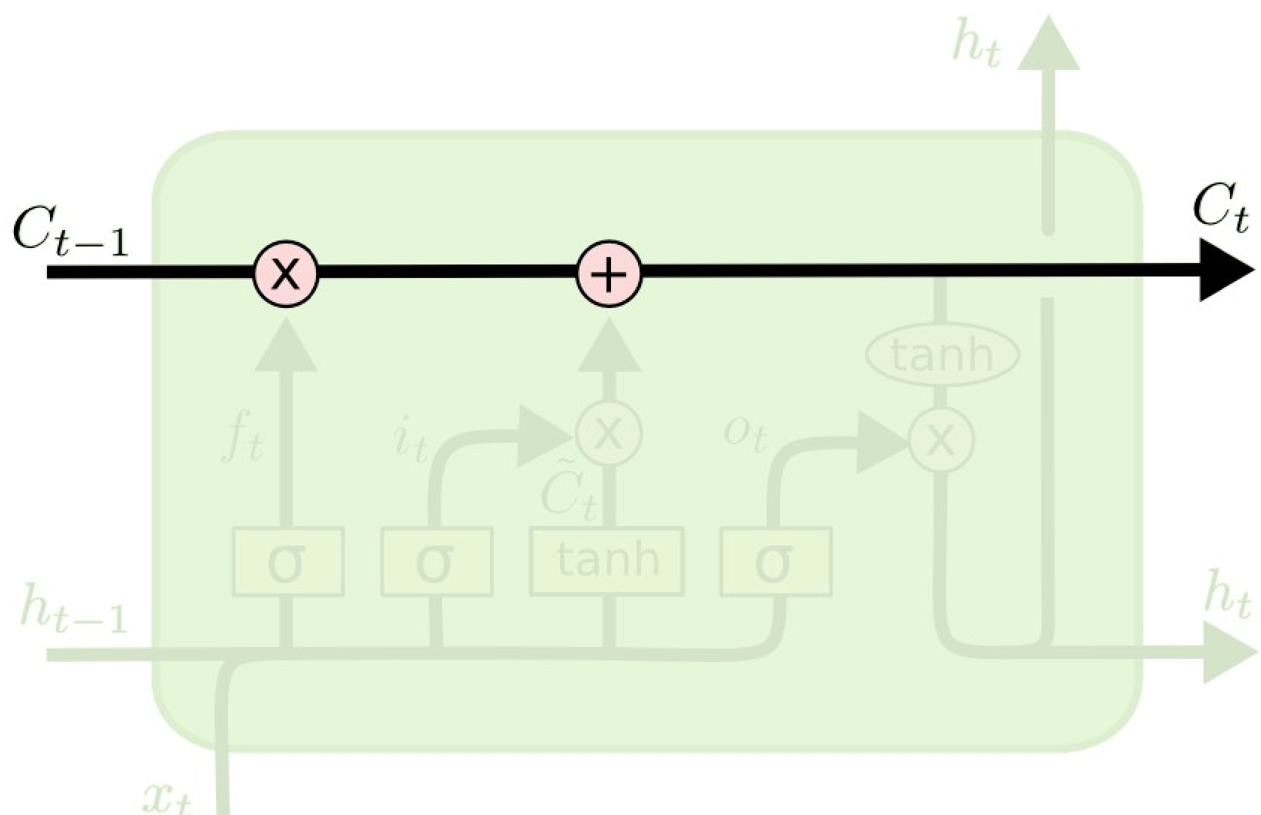


LSTM: Long Short Term Memory Networks

Credit: Christopher Olah

LSTM

1. Cell State / Memory

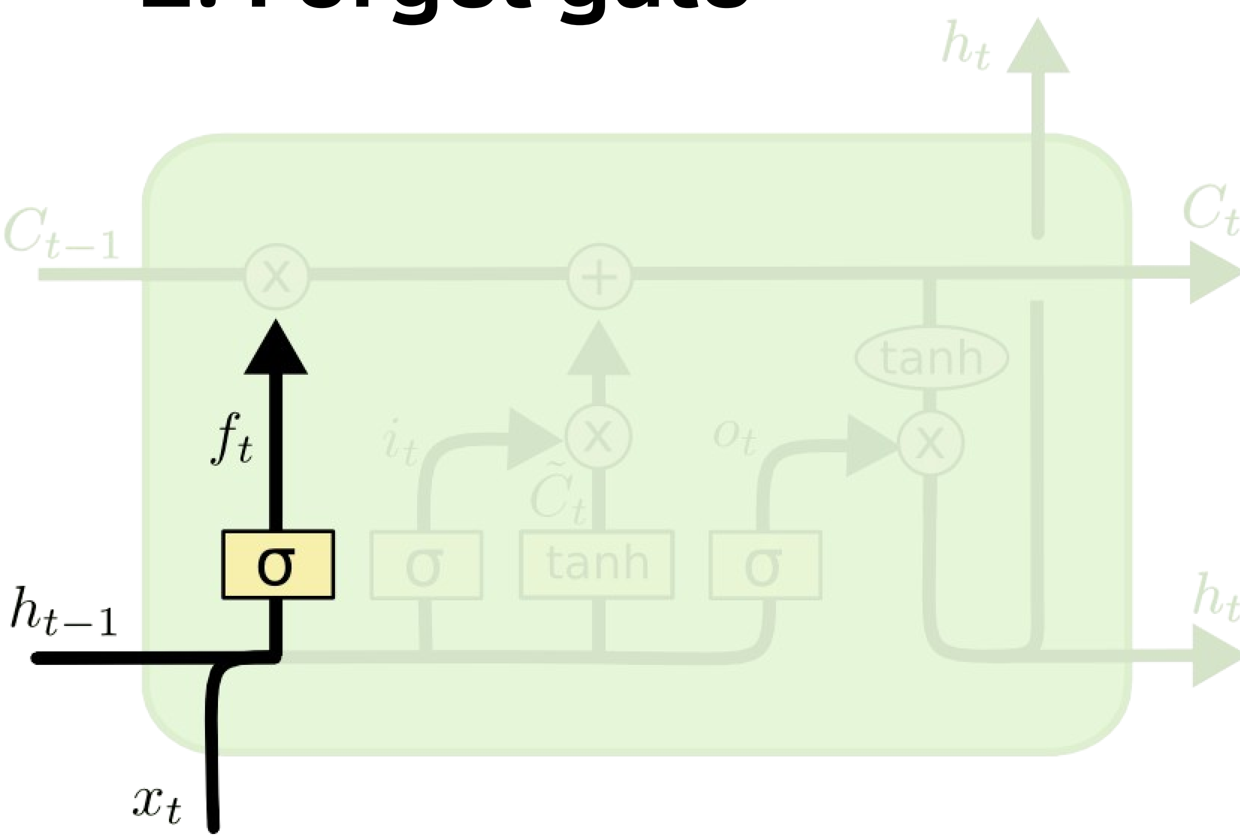


The LSTM have the ability to remove or add information to the cell state, carefully regulated by structures called gates

Credit: Christopher Olah

LSTM

2. Forget gate



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

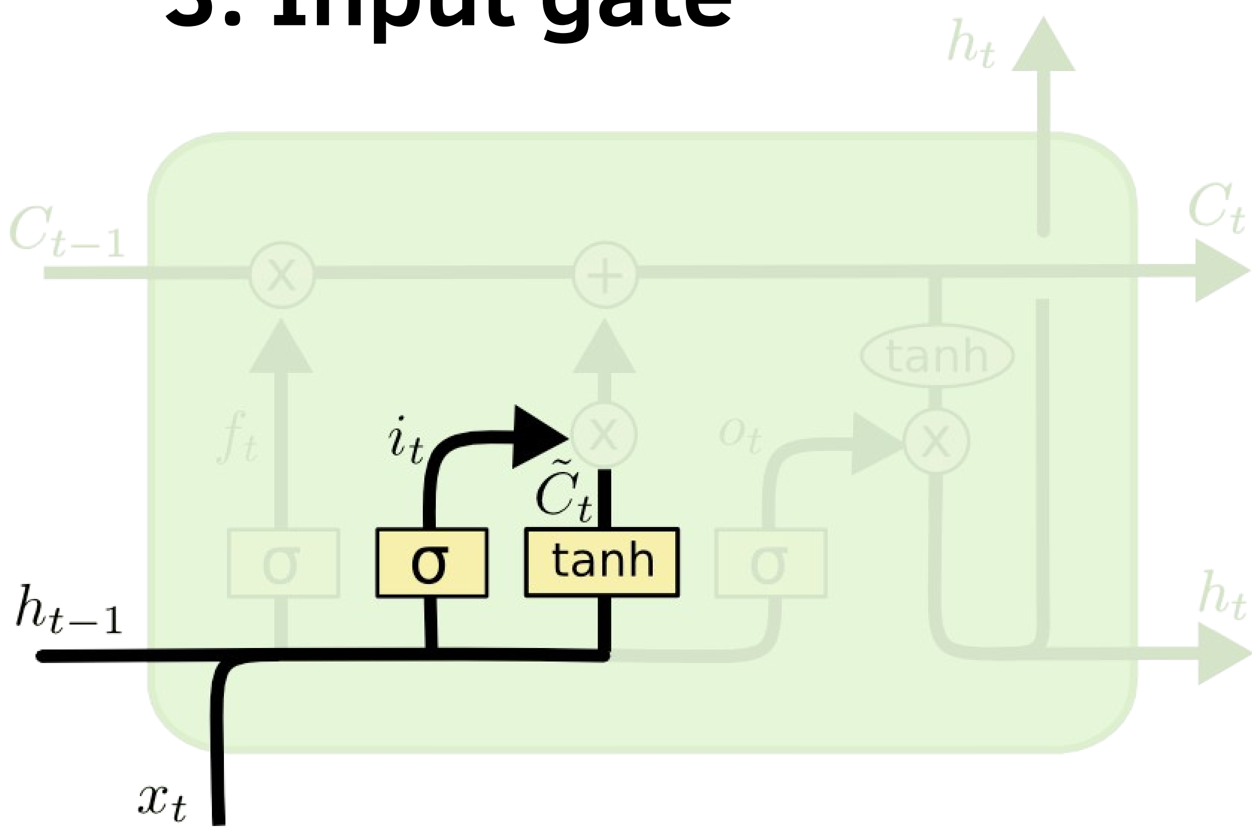
Should we continue to remember this "bit" of information or not?

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer."

Credit: Christopher Olah

LSTM

3. Input gate



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

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

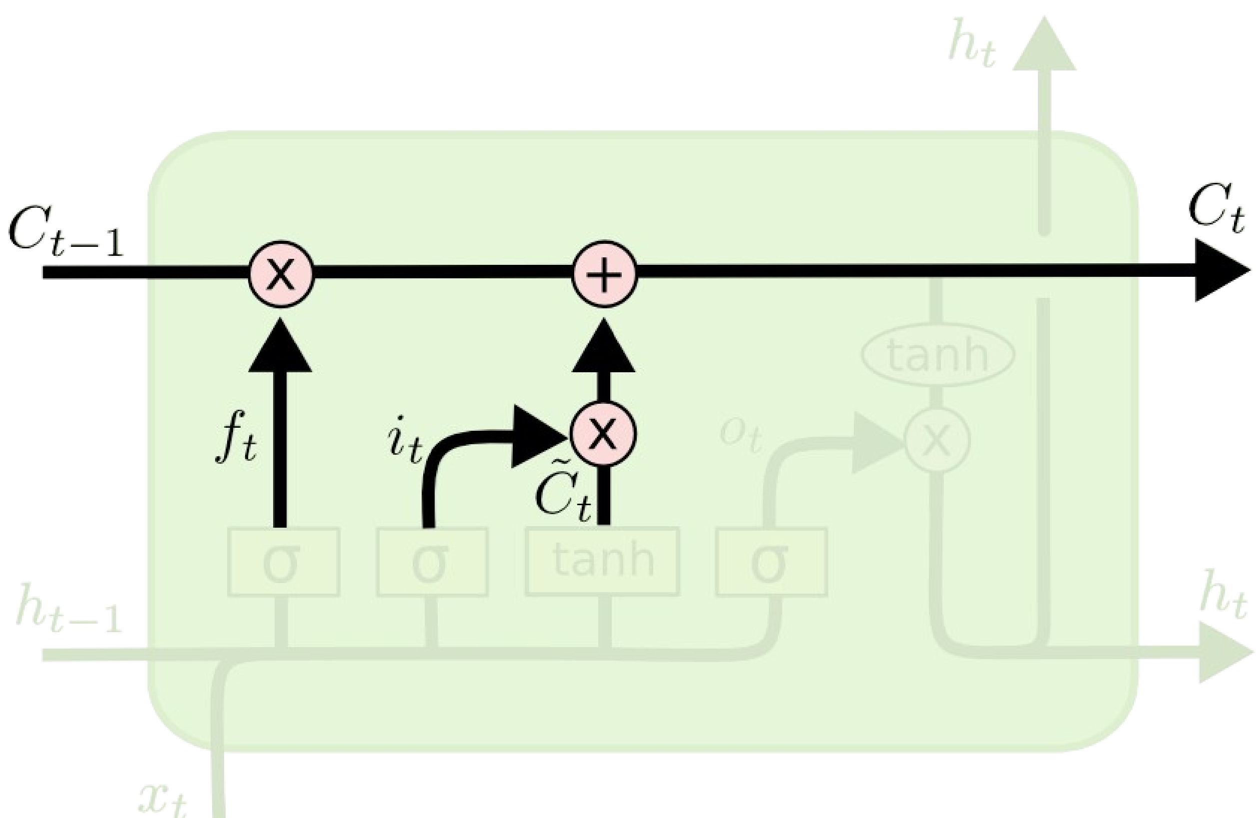
Should we update this “bit” of information or not? If so, with what?

The next step is to decide what new information we’re going to store in the cell state. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.

Credit: Christopher Olah

LSTM

4. Memory Update



Forget that — Memorize this

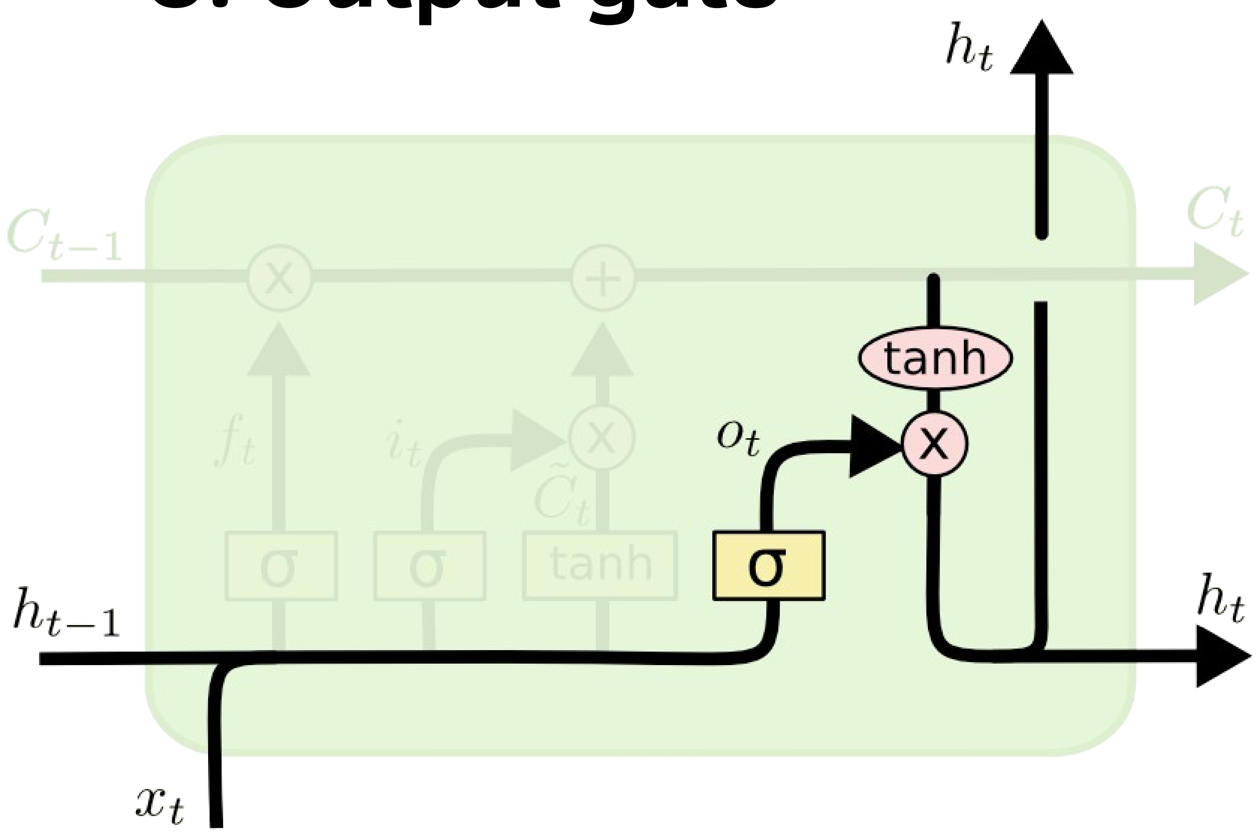
$$C_t = \underbrace{f_t * C_{t-1}}_{} + \underbrace{i_t * \tilde{C}_t}_{}$$

Decide what will be kept in the cell state/memory

Credit: Christopher Olah

LSTM

5. Output gate



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

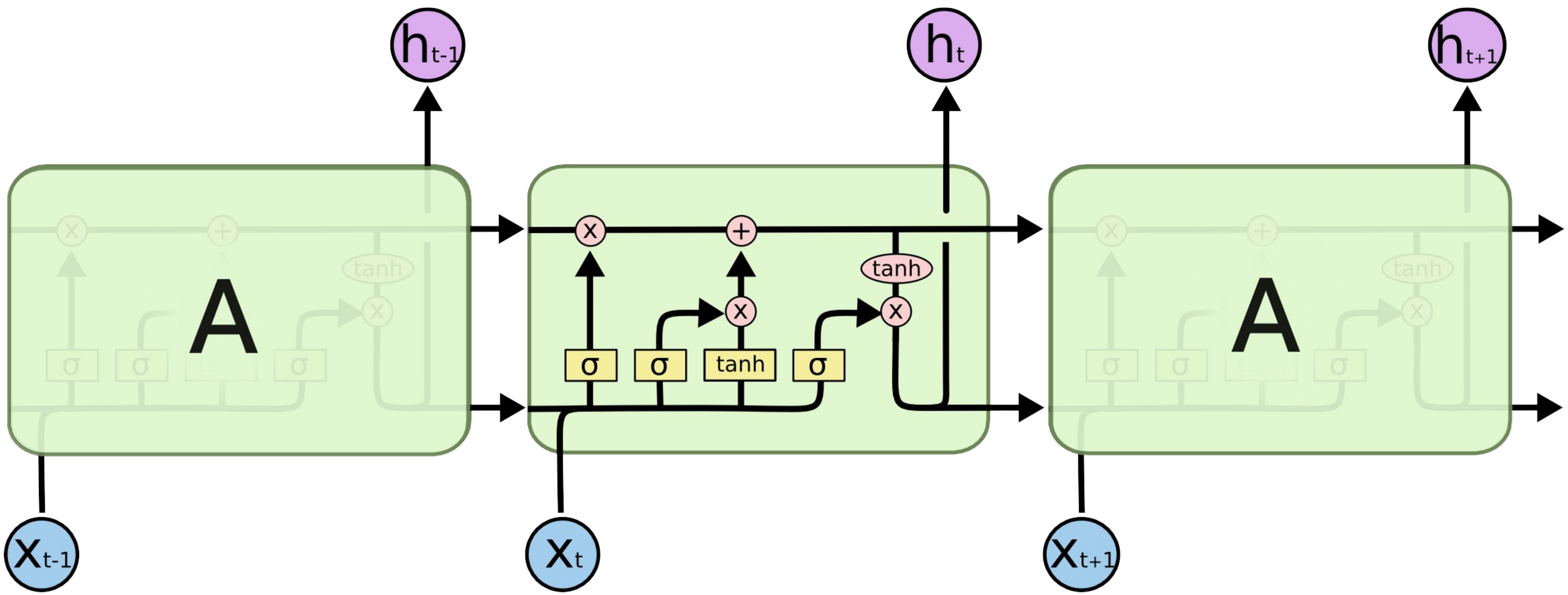
Should we output this “bit” of information?

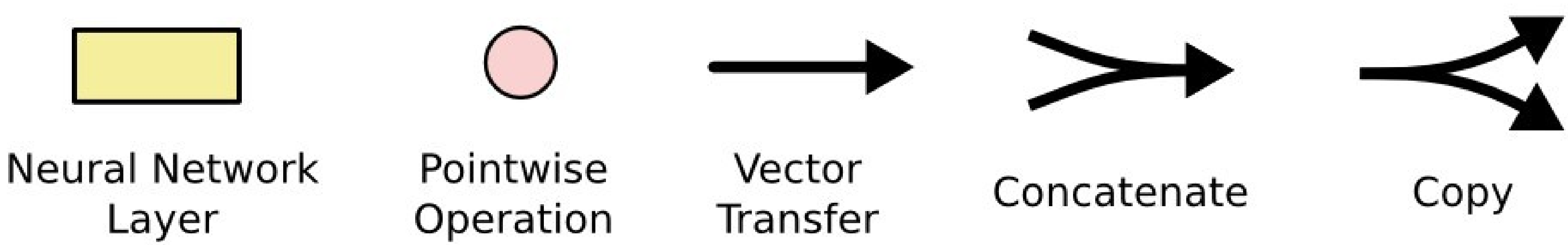
This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we’re going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

Credit: Christopher Olah

LSTM

Complete LSTM - A pretty sophisticated cell

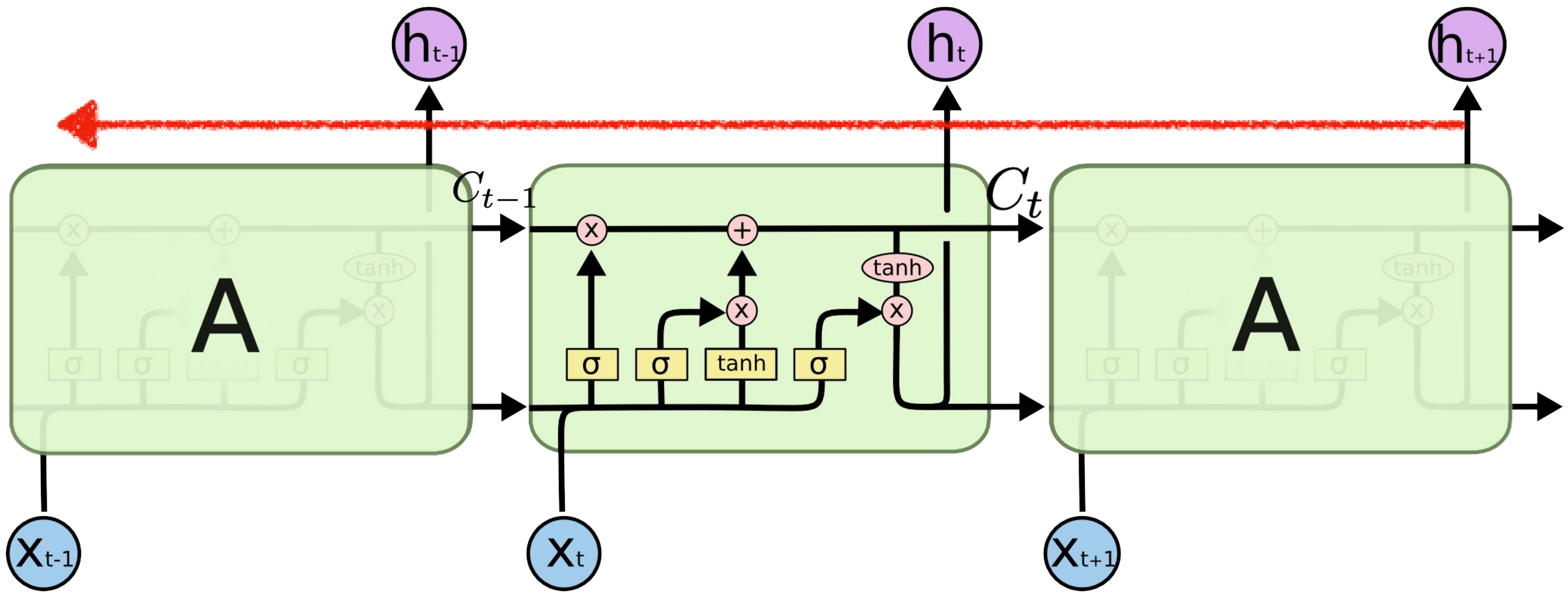




Hochreiter and Schmidhuber.
Long Short Term Memory.
Neural Computation. 1997

Credit: Christopher Olah

LSTM

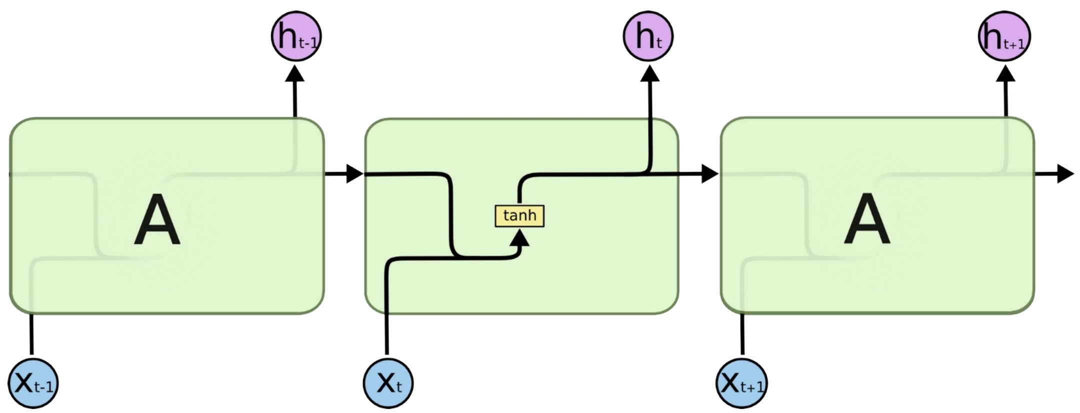


Back-propagation from C_t to C_{t-1} only has element wise multiplication by f_t
and no by weight matrix
There is an uninterrupted gradient flow

Credit: Christopher Olah

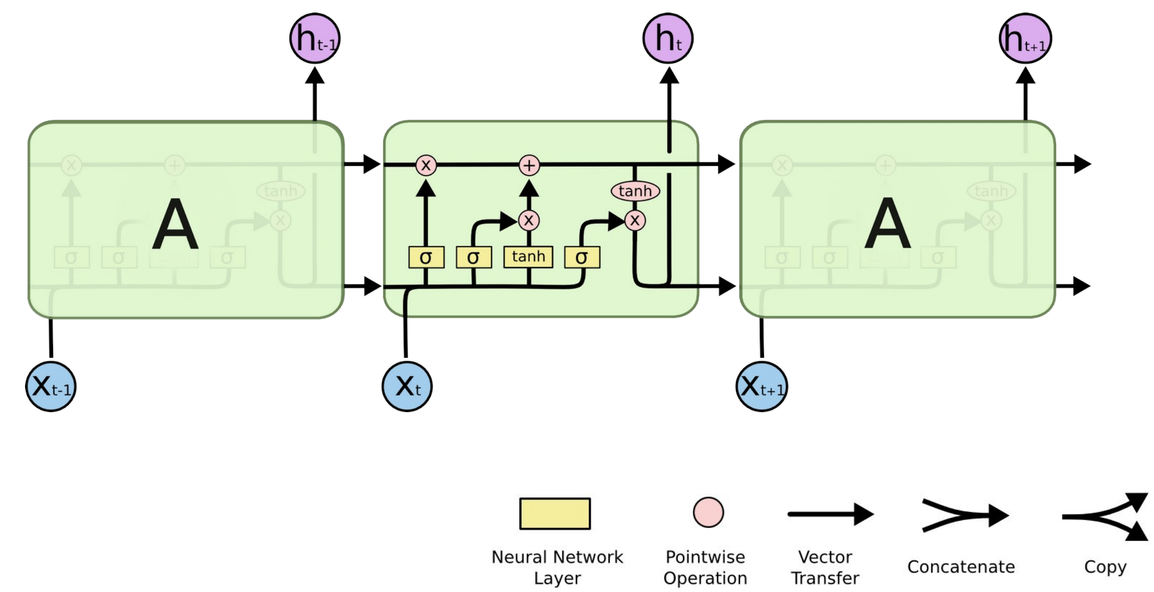
Plain RNN vs LSTM

Plain RNN



$$\boldsymbol{h}_t = tanh(\boldsymbol{W} \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{bmatrix})$$

LSTM



$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \tilde{C}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \boldsymbol{W} \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{bmatrix} + \boldsymbol{b}$$

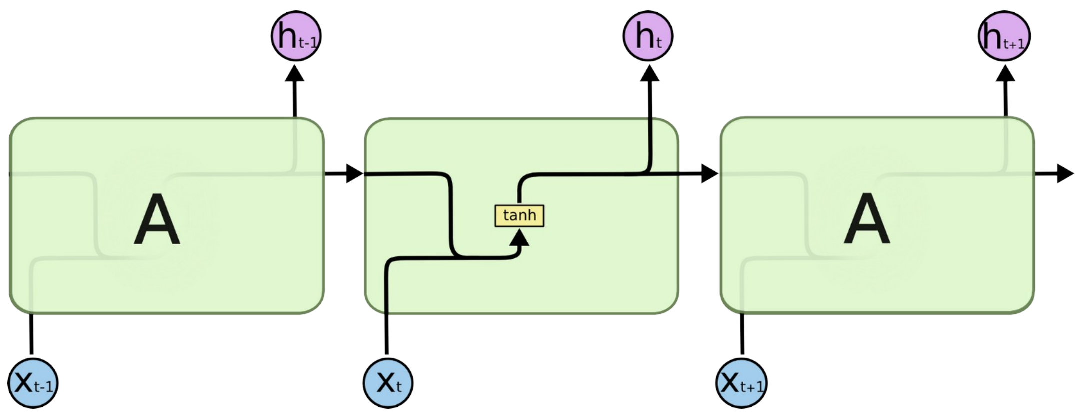
$$\boldsymbol{C}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{C}}_t$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * tanh(\boldsymbol{C}_t)$$

Hochreiter and Schmidhuber. Long Short Term Memory. Neural Computation. 1997

Plain RNN vs LSTM

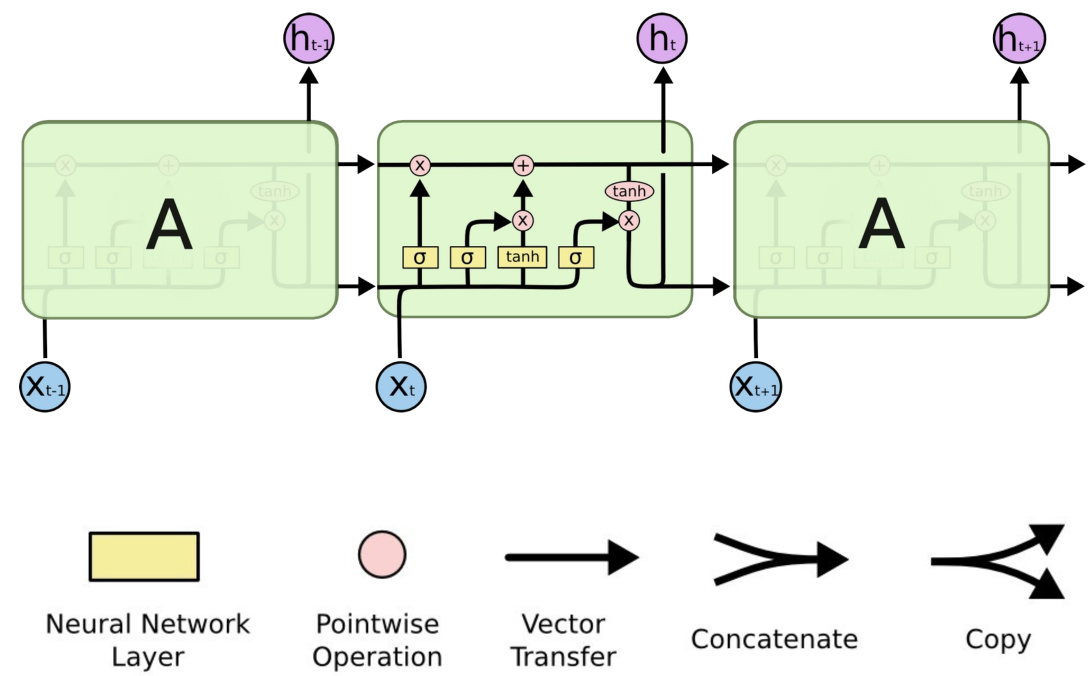
Plain RNN



Backward flow of gradient in RNN can:

- explode -> gradient clipping
- vanish -> use LSTM

LSTM



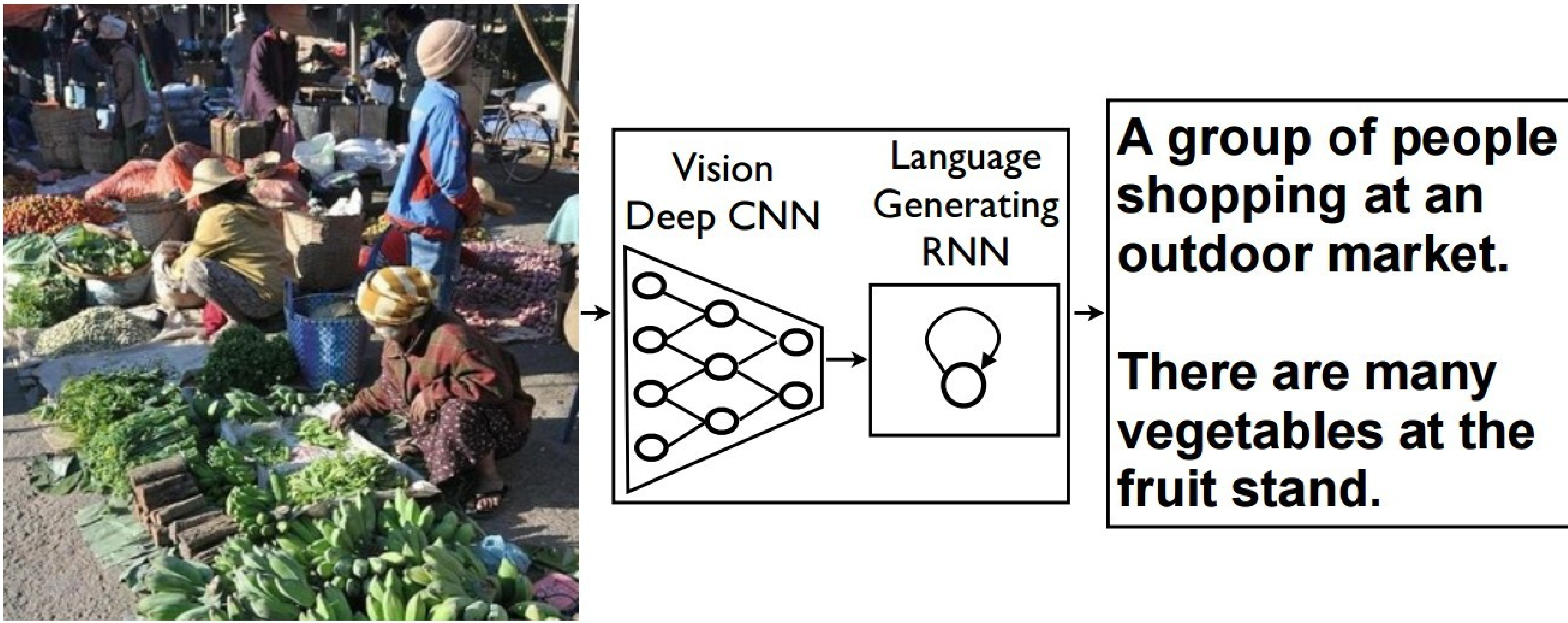
Backward flow of gradient in LSTM:

- their additive interactions improve the gradient flow

Hochreiter and Schmidhuber. Long Short Term Memory. Neural Computation. 1997

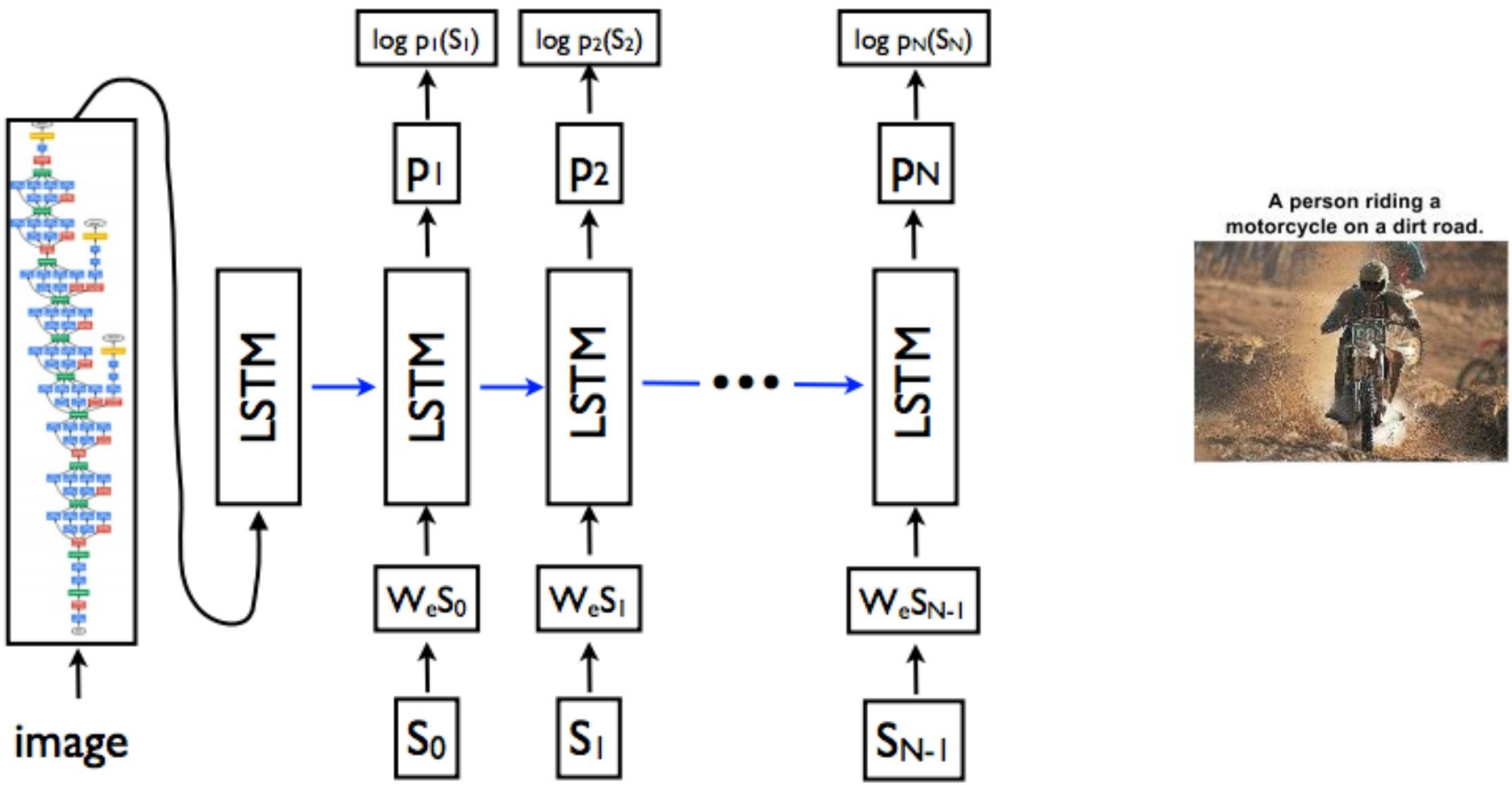
Credit: Christopher Olah

Show and Tell: A neural Image Caption Generator



Show and Tell: A Neural Image Caption Generator, Vinyals et. al., CVPR 2015

Show and Tell: A neural Image Caption Generator



Show and Tell: A Neural Image Caption Generator, Vinyals et. al., CVPR 2015

Show and Tell: A neural Image Caption Generator

https://pdollar.wordpress.com/2015/01/21/image-captioning/

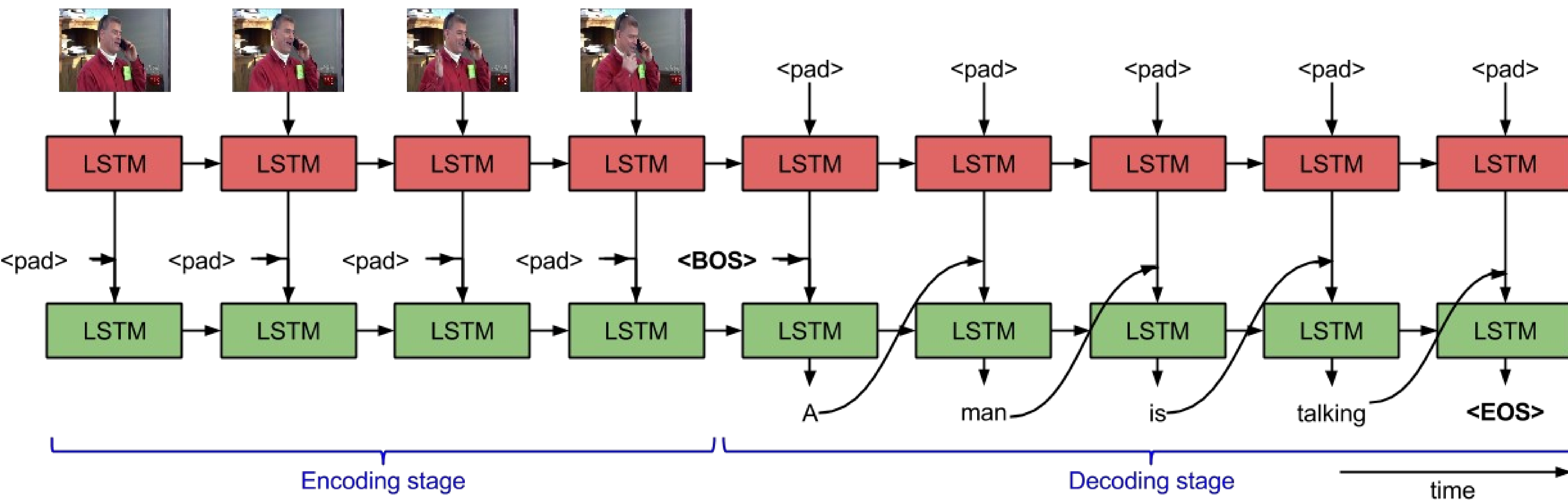
Show and Tell: A Neural Image Caption Generator, Vinyals et. al., CVPR 2015

Show and Tell: A neural Image Caption Generator

https://pdollar.wordpress.com/2015/01/21/image-captioning/

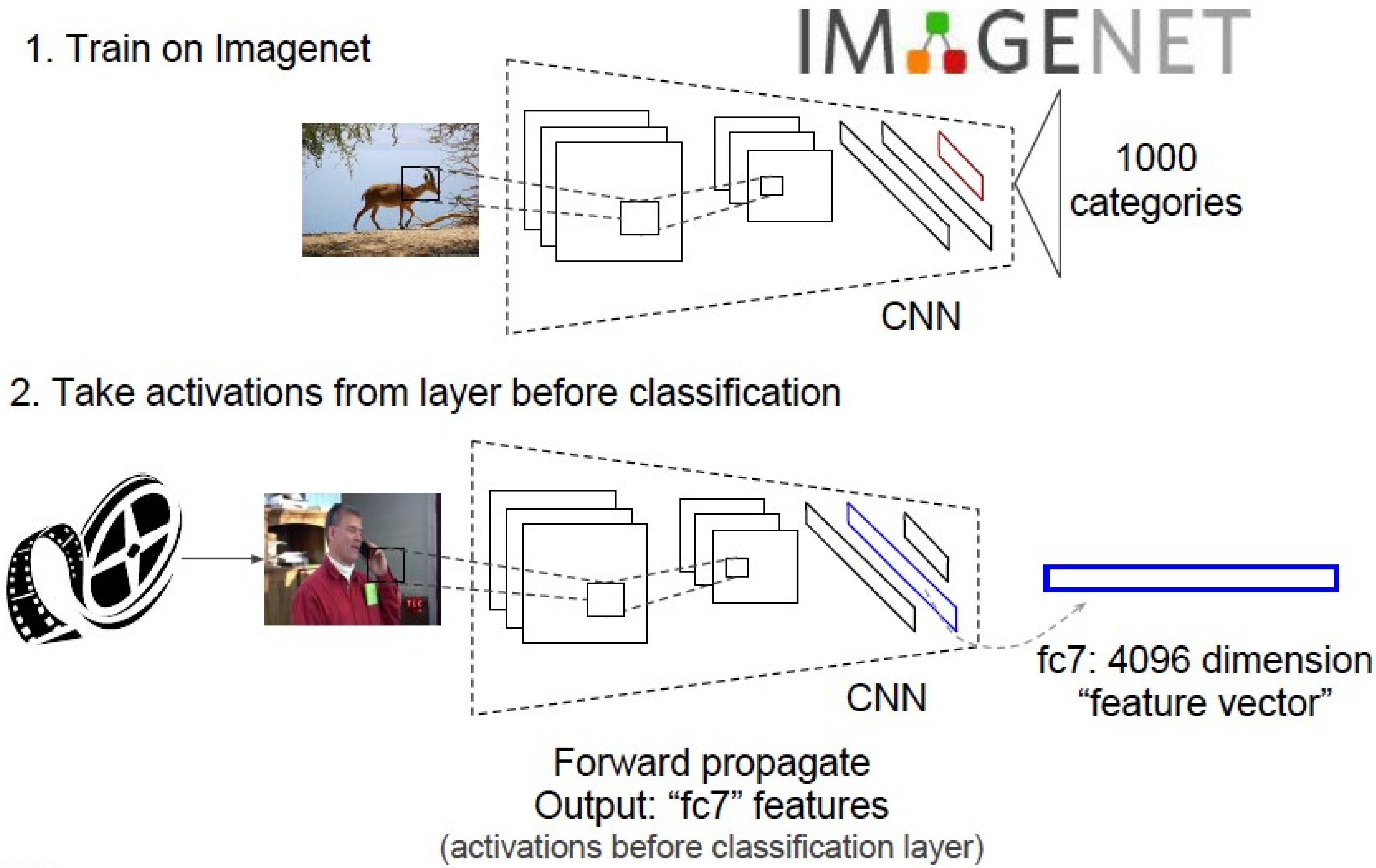
Show and Tell: A Neural Image Caption Generator, Vinyals et. al., CVPR 2015

Sequence to Sequence
Video to Text



Venugopalan, et. al. Sequence to Sequence – Video to Text. ICCV 2015

Sequence to Sequence
Video to Text

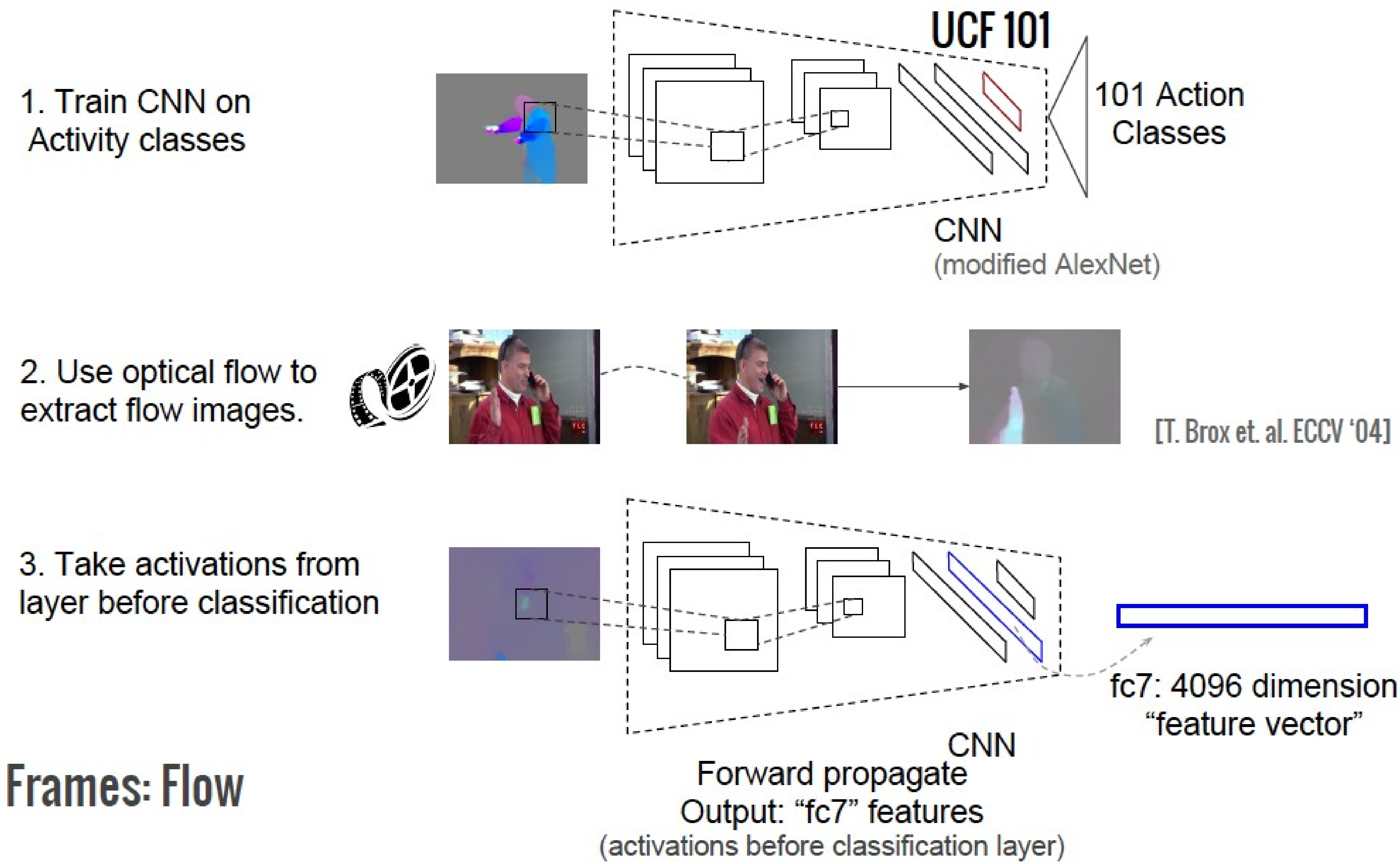


Frames: RGB

source: Venugopalan

Venugopalan, et. al. Sequence to Sequence – Video to Text, ICCV 2015

Sequence to Sequence
Video to Text



source: Venugopalan

Venugopalan, et. al. Sequence to Sequence – Video to Text, ICCV 2015

Sequence to Sequence
Video to Text

Movie Corpus - DVS

CC: Queen: "Which estate?"

DVS: Looking troubled, the Queen descends the stairs.

The Queen rushes into the courtyard. She then puts a head scarf on . . .

. . . and gets into the driver's side of a nearby Land Rover.

The Land Rover pulls away.

Three bodyguards quickly jump into a nearby car and follow her.

Processed: Looking troubled, someone descends the stairs.

Someone rushes into the courtyard. She then puts a head scarf on ...

source: Venugopalan

Venugopalan, et. al, Sequence to Sequence – Video to Text, ICCV 2015

Sequence to Sequence
Video to Text

Summary and examples of results:

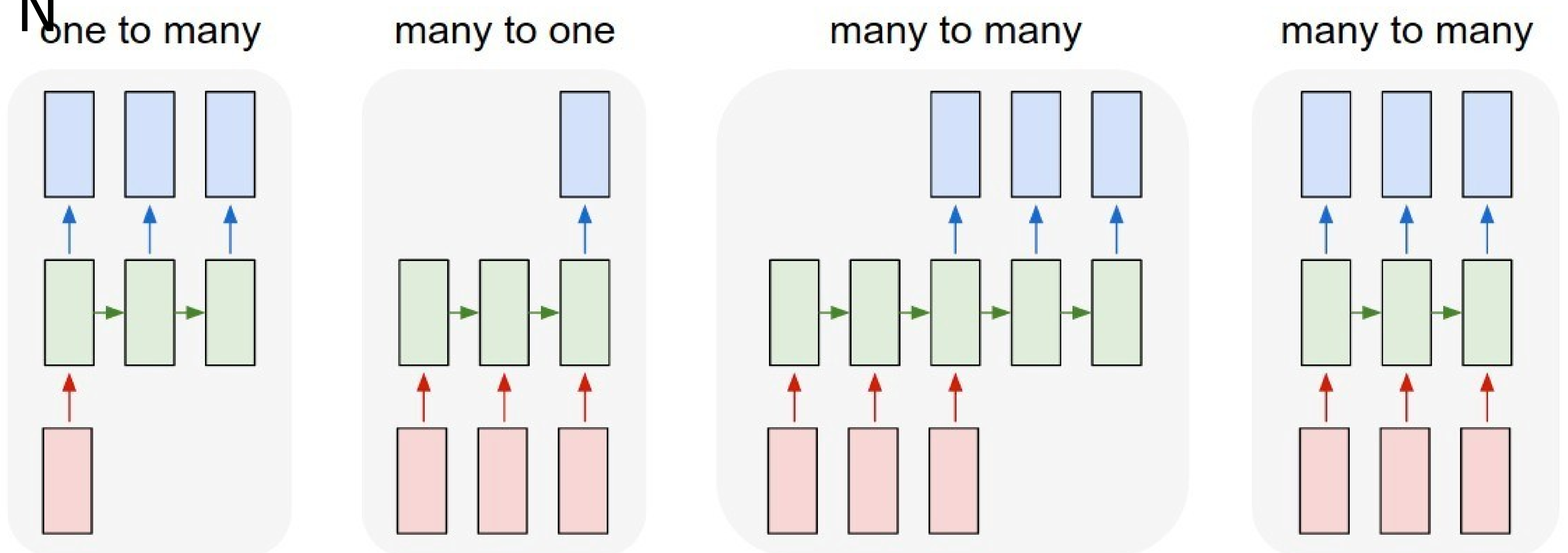
https://youtu.be/-xNI7e7YgDk

source: Venugopalan

Venugopalan, et. al. Sequence to Sequence – Video to Text. ICCV 2015

Summary of Today's class

RN
N



Input: No sequence

Output: Sequence

Example:
- Image captioning (image -> words)

Input: Sequence

Output: No sequence

Example:
- sentence classification
- sentiment classification (words seq.->sentiment)

Input: Sequence

Output: Sequence

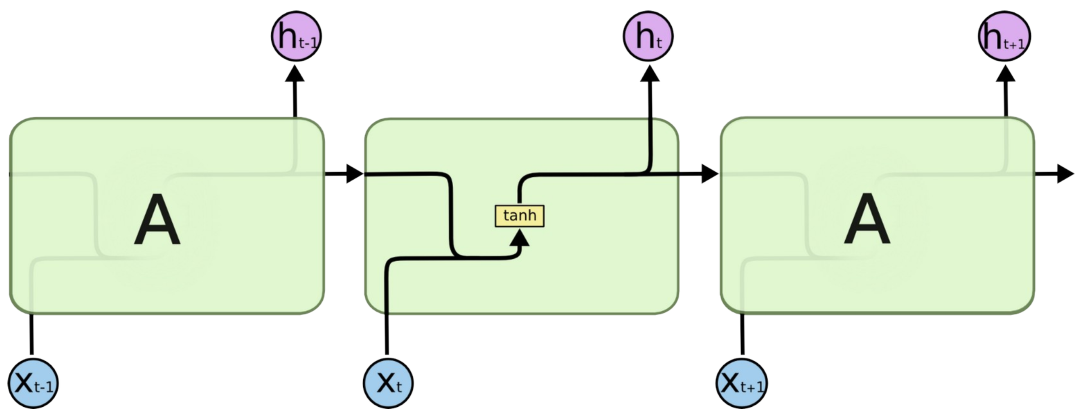
Example:
- machine translation, (words seq-> words seq)

Example:
- video captioning

Source: Dhruv Batra, Andrej Karpathy

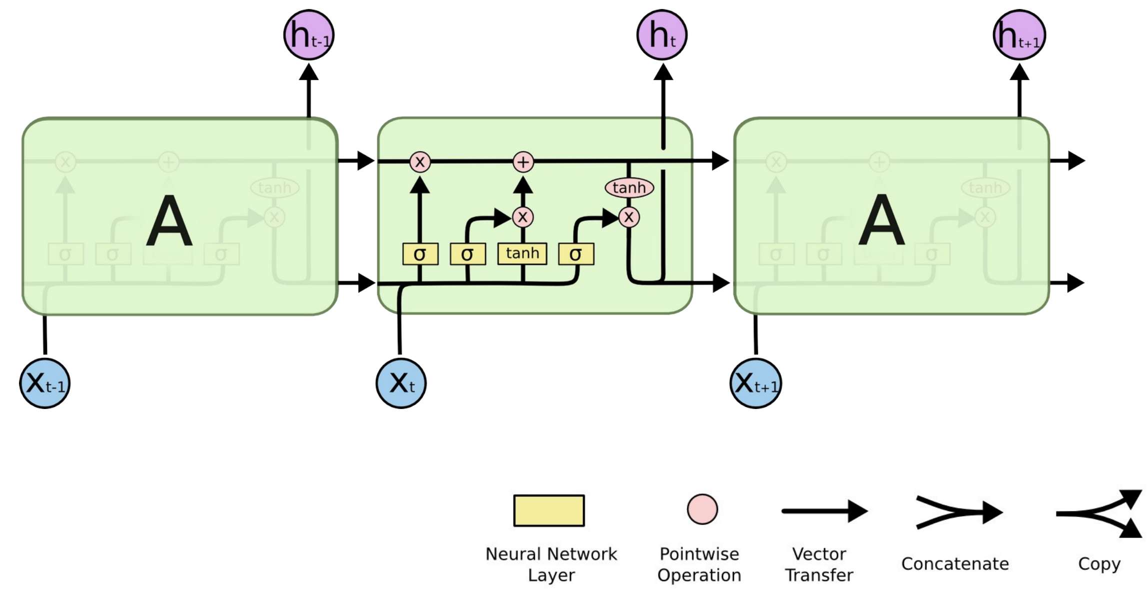
RNN vs LSTM

Plain RNN



$$\boldsymbol{h}_t = tanh(\boldsymbol{W} \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{bmatrix})$$

LSTM



$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \tilde{C}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \boldsymbol{W} \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{bmatrix} + \boldsymbol{b}$$

$$\boldsymbol{C}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{C}}_t$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * tanh(\boldsymbol{C}_t)$$

Hochreiter and Schmidhuber.
Long Short Term Memory.
Neural Computation. 1997