Lecture 8: Recurrent Neural Networks

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/VE445/index.html

Motivation

- Not all problems can be converted into one with fixed length inputs and outputs
- Problems such as *Speech Recognition* or *Time-series Prediction* require a system to store and use context information
  - Simple case:
  - Output YES if the number of 1s is even, else NO!

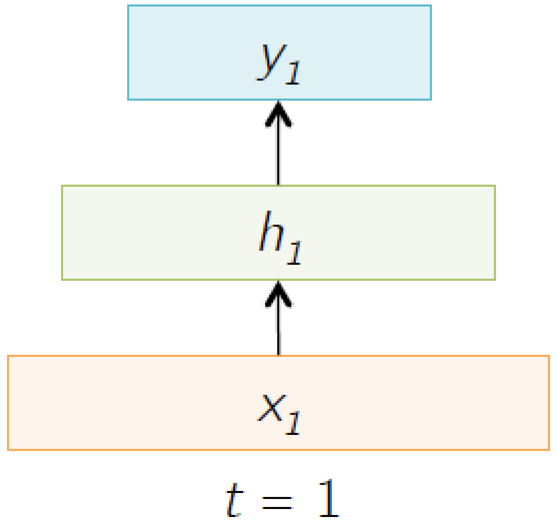
    1000010101 – YES, 100011 – NO, …
- Hard/Impossible to choose a fixed context window
  - There can always be a new sample longer than anything seen

Recurrent neural networks (RNNs)

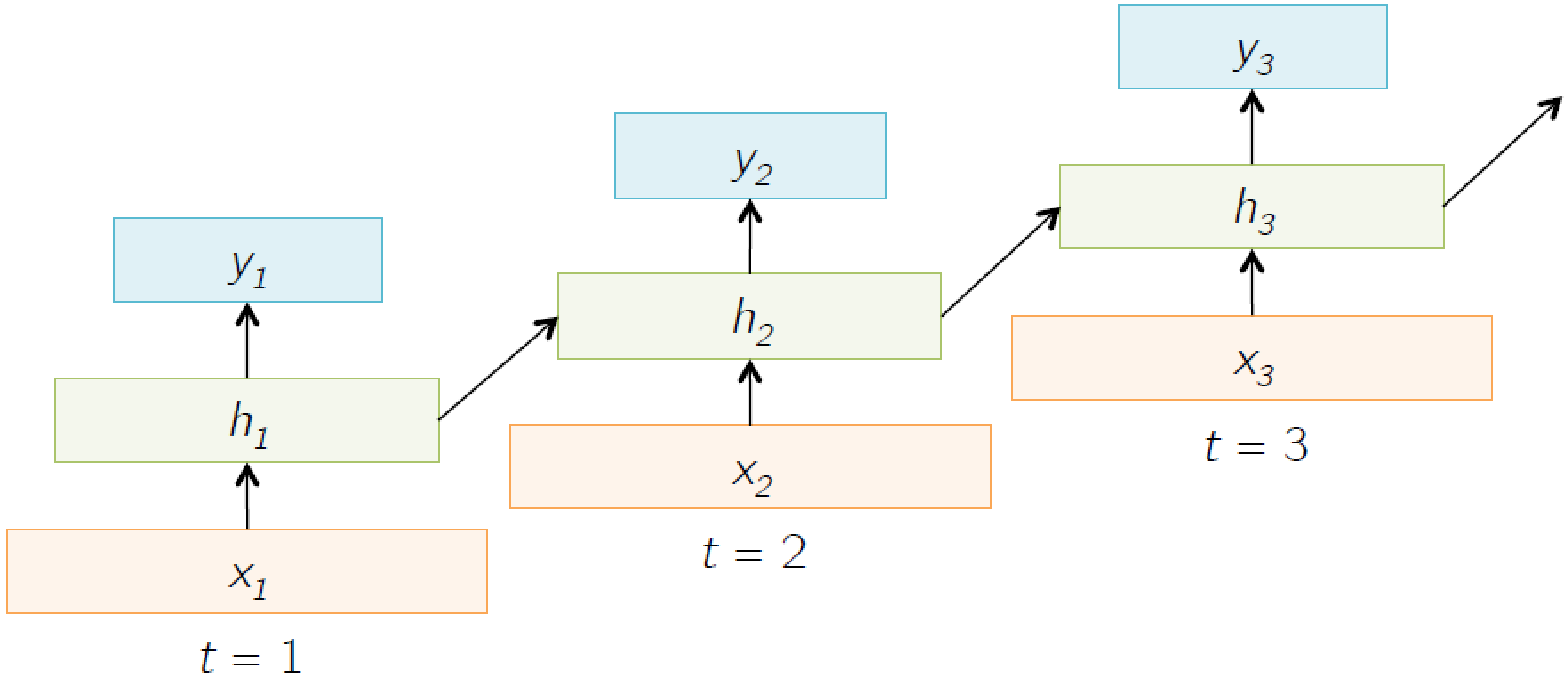
- Recurrent neural networks take the previous output or hidden states as inputs. The composite input at time t has some historical information about the happenings at time T < t

- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori

Feed-forward Network

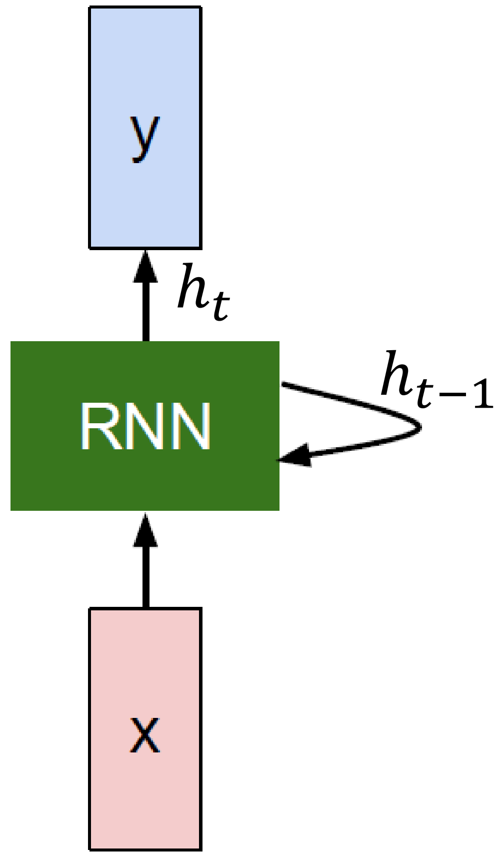


RNN example



The vanilla RNN cell

- The state consists of a single *"hidden"* vector **h**:



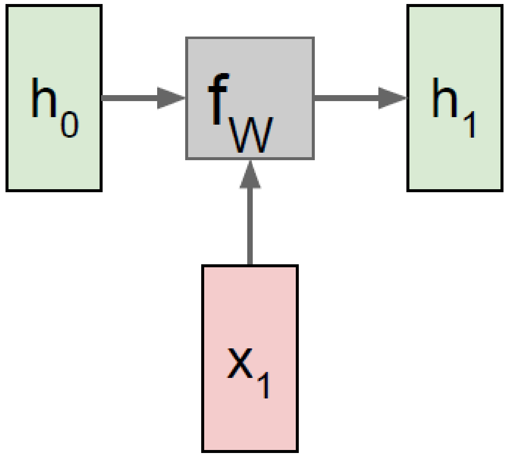
$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

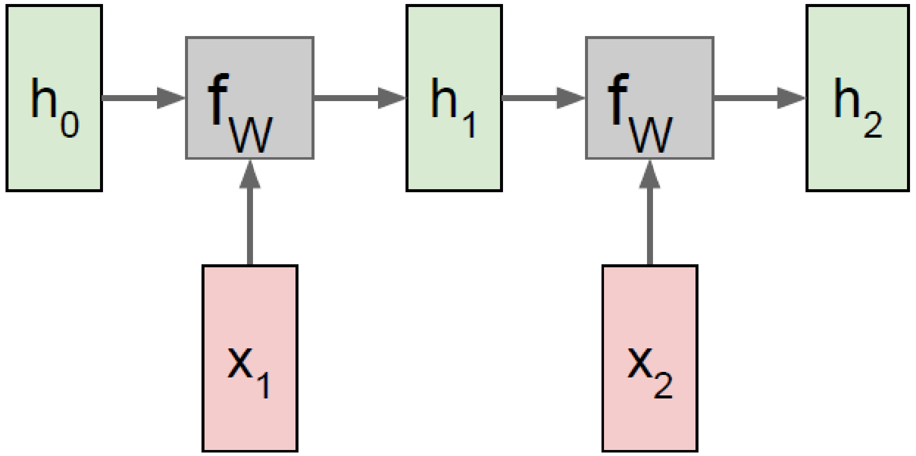
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

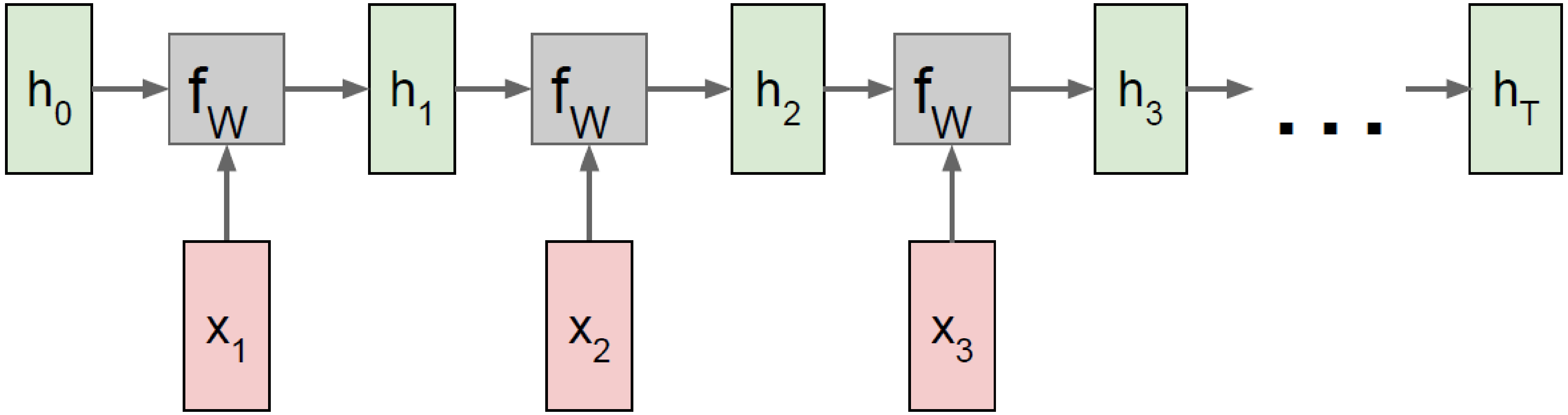
Computational graph of RNN



Computational graph of RNN

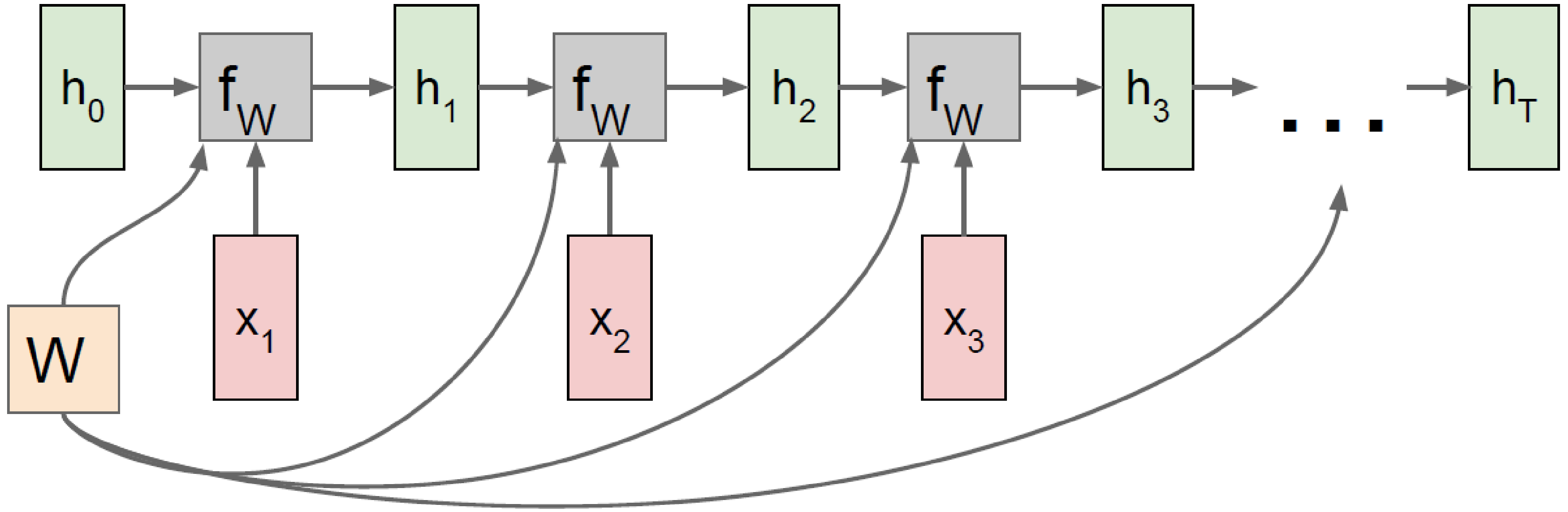


Computational graph of RNN

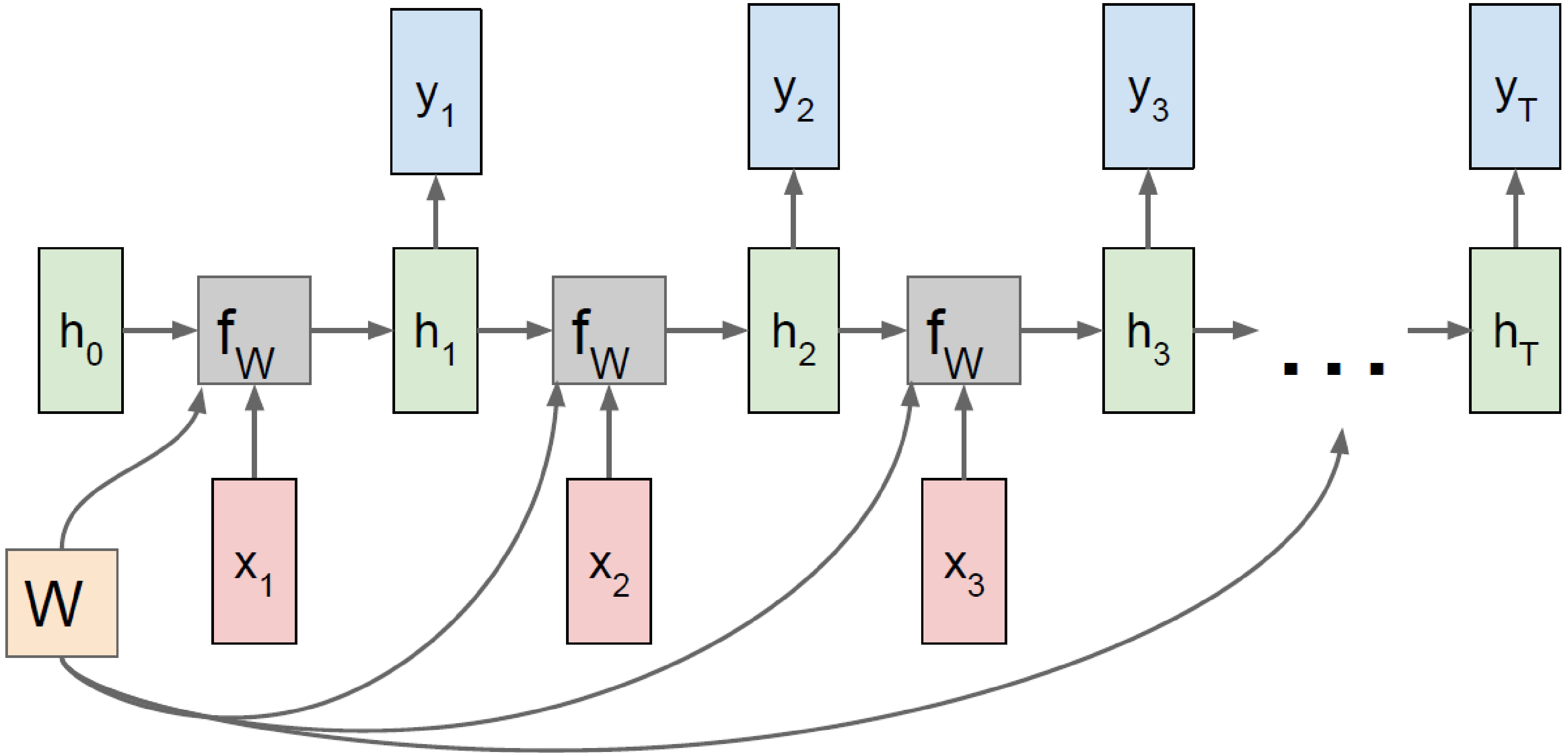


Computational graph of RNN

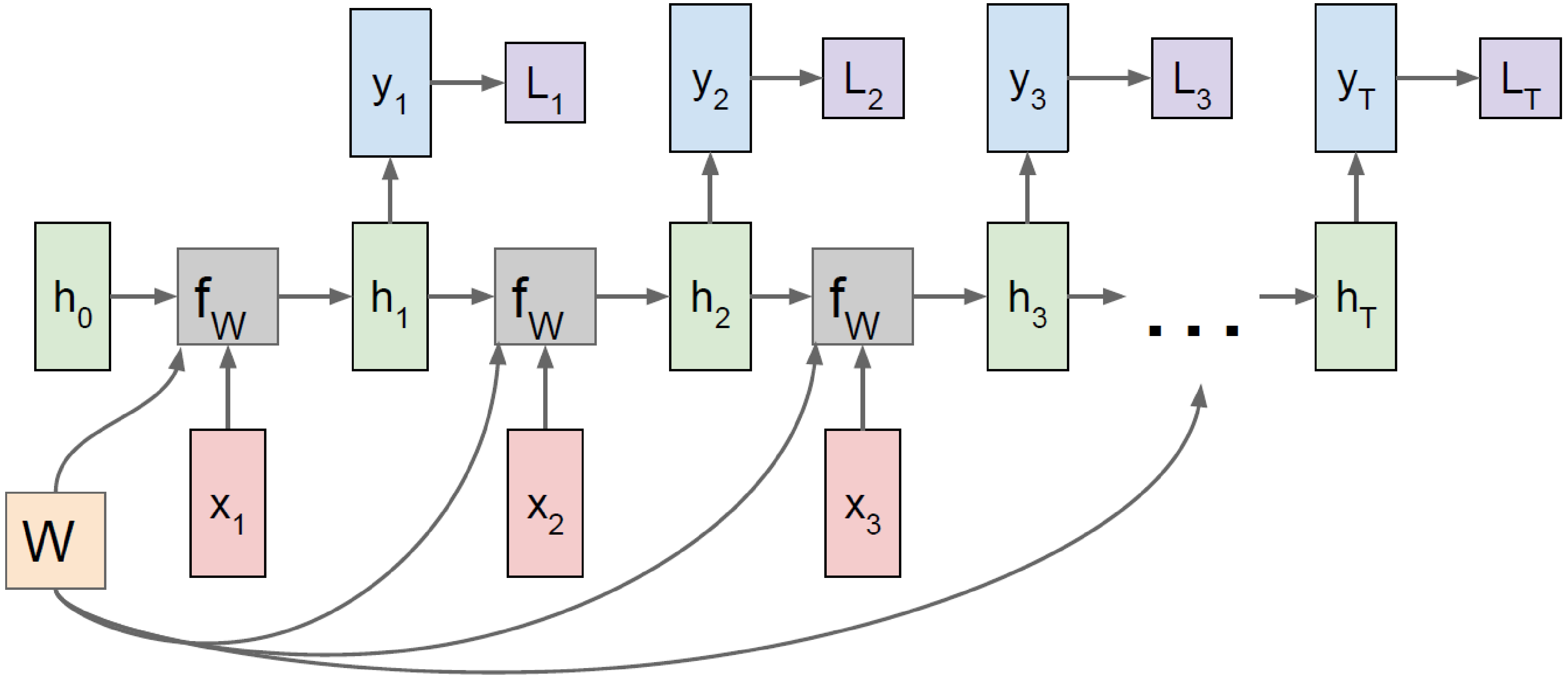
- Re-use the same weight matrix at every time-step



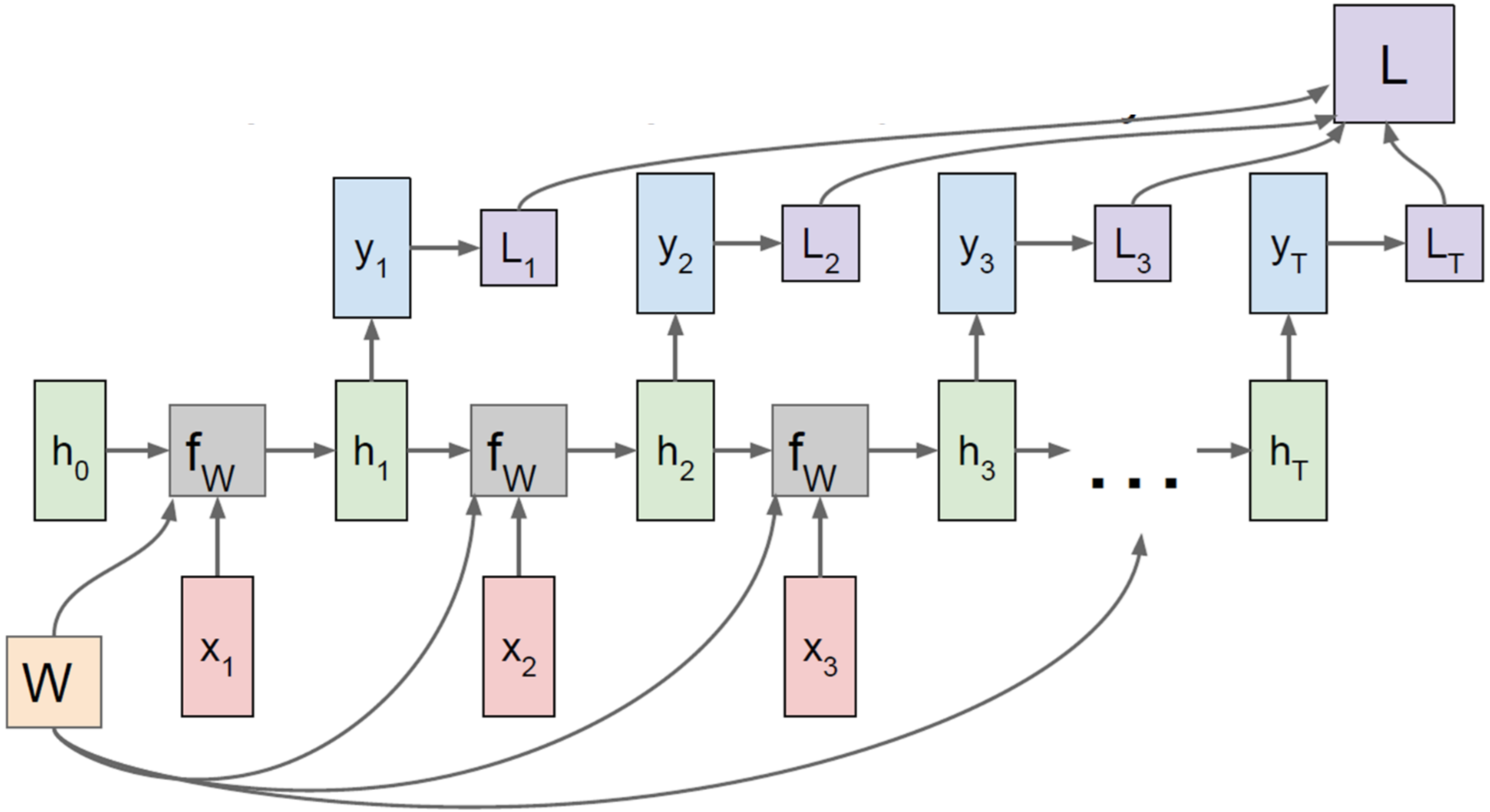
Computational graph of RNN: many to many



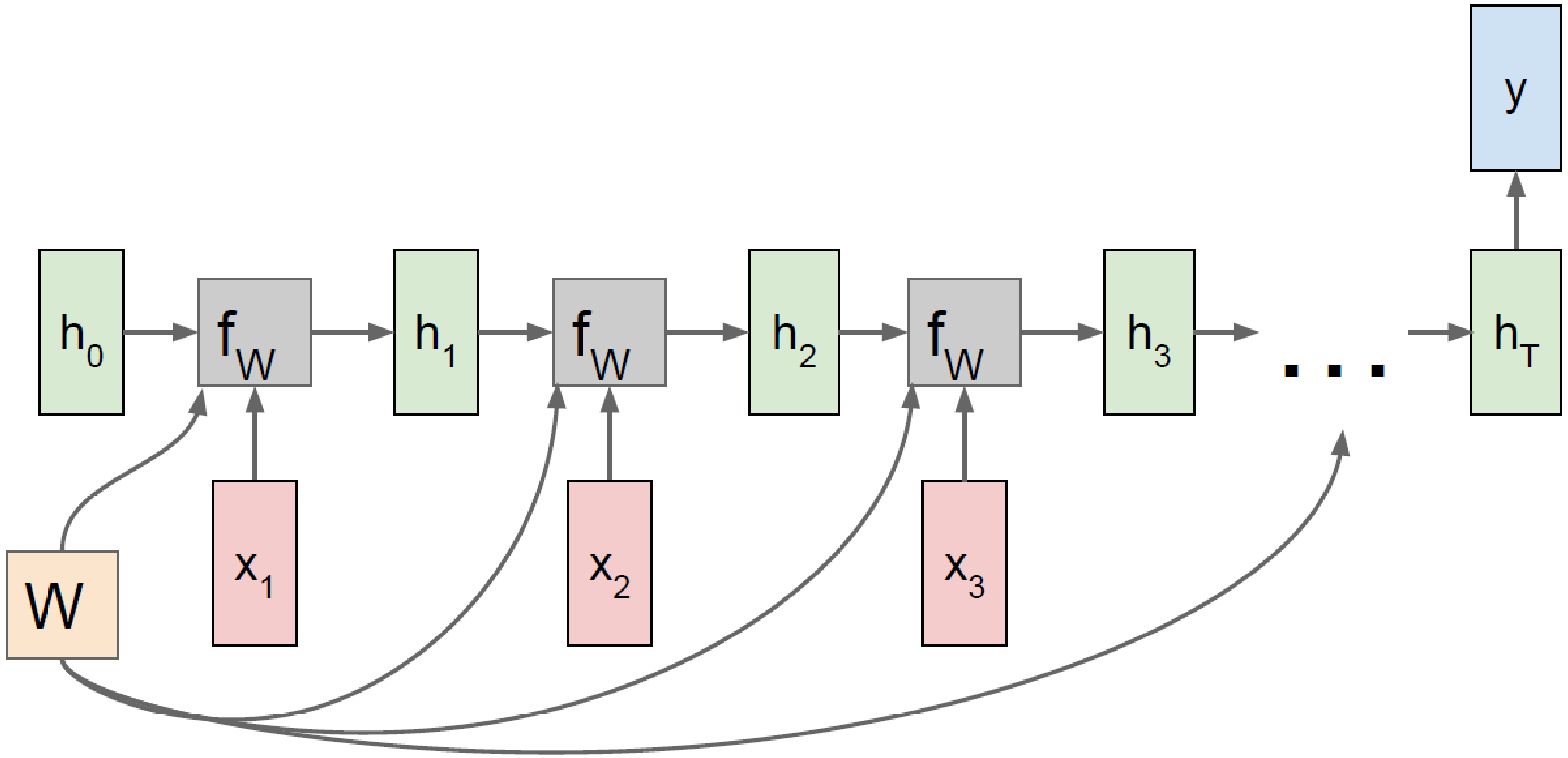
Computational graph of RNN: many to many



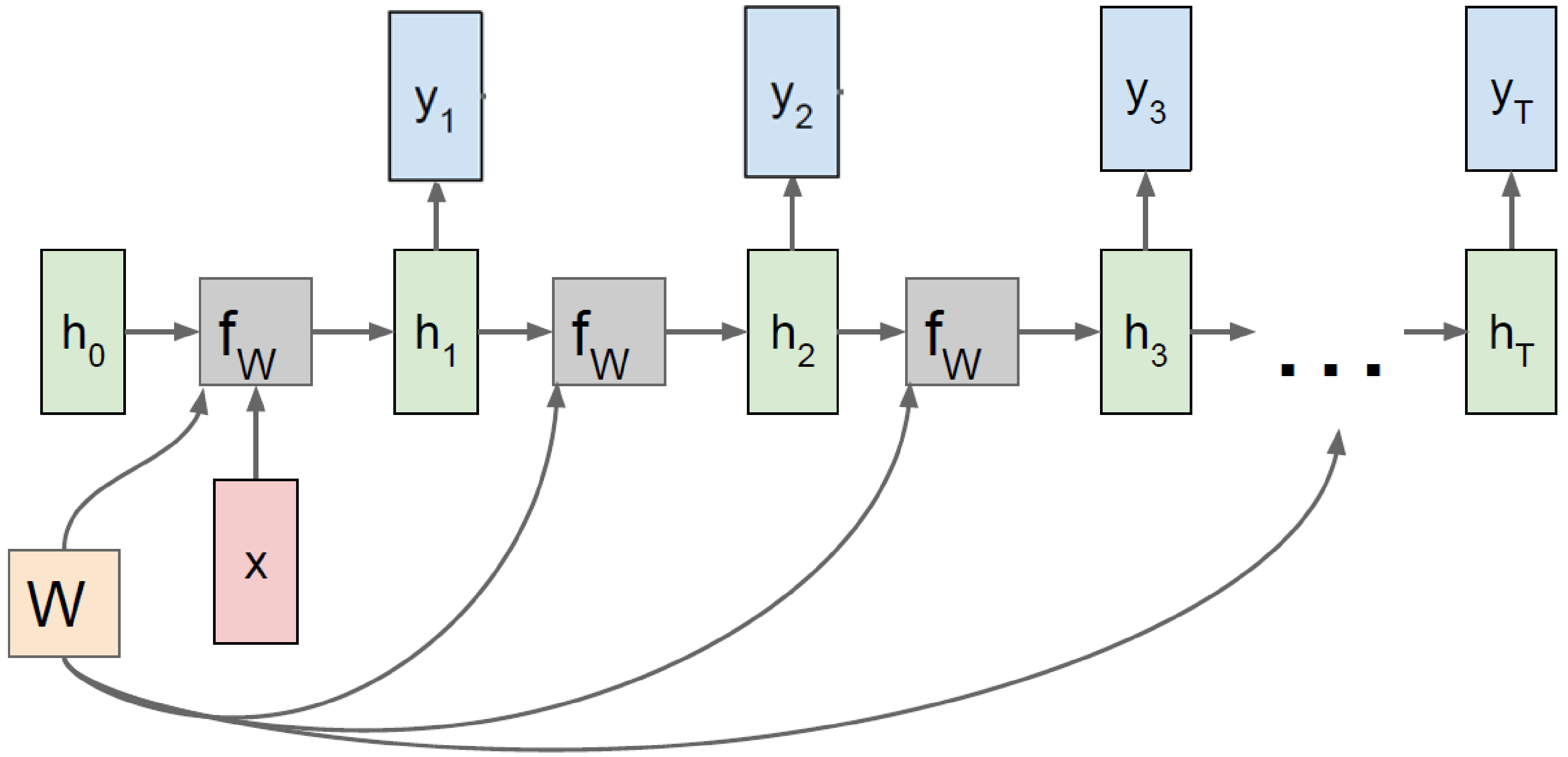
Computational graph of RNN: many to many



Computational graph of RNN: many to one

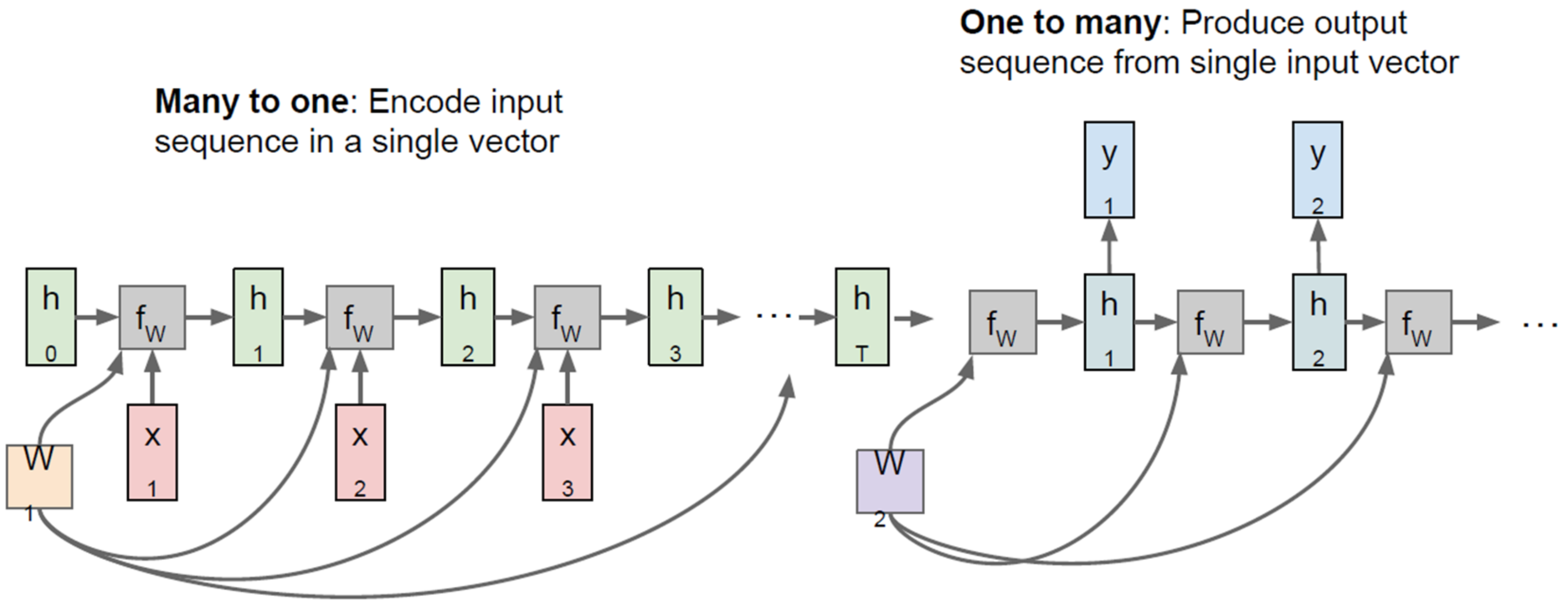


Computational graph of RNN: one to many



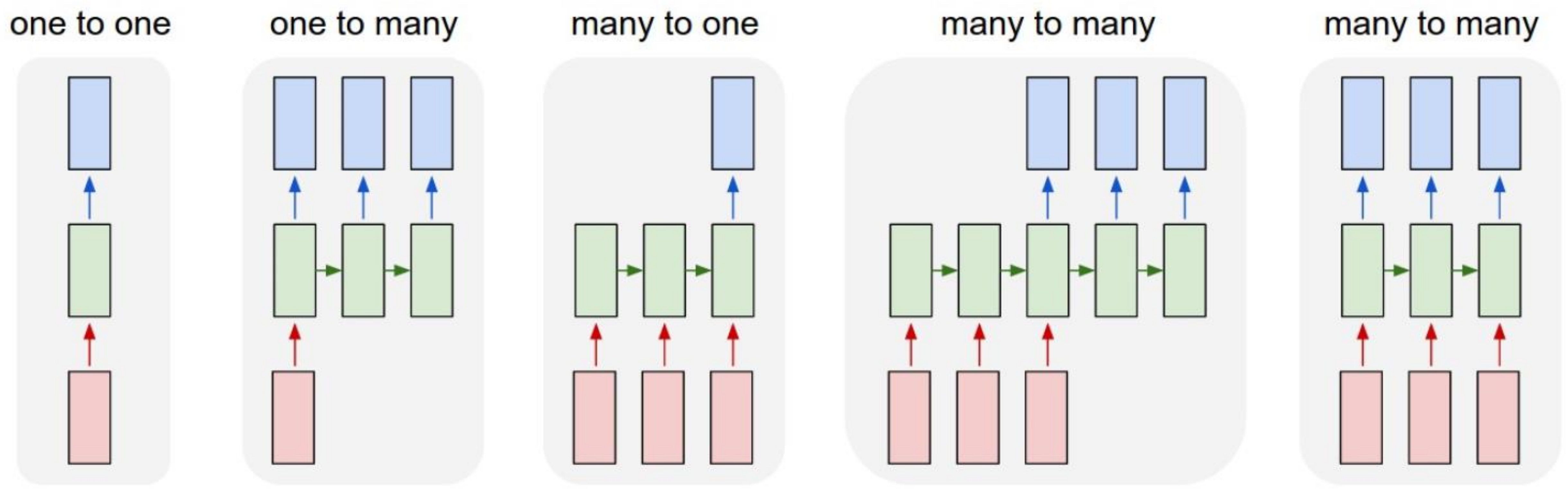
Sequence to sequence

- Equals to man-to-one plus one-to-many



One to many: Produce output sequence from single input vector
Many to one: Encode input sequence in a single vector
h_0
f_W
h_1
h_2
h_3
h_T
x_1
x_2
x_3
W_1
W_2
y_1
y_2
...

Different RNNs



Vanilla NN

Image captioning
Text generation

Text classification
Sentiment analysis

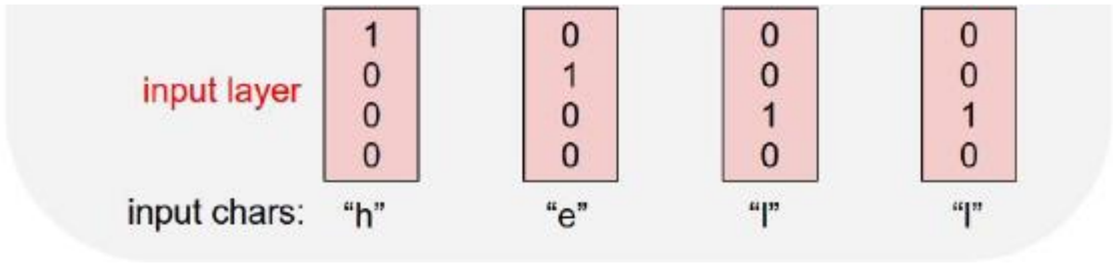
Machine translation
Dialogue system

Stock price estimation
Video frame classification

Example: Character-level language model

- Given previous words/characters, predict the next
- Vocabulary: [h, e, l, o]
- Example of training sequence:

  “Hello”

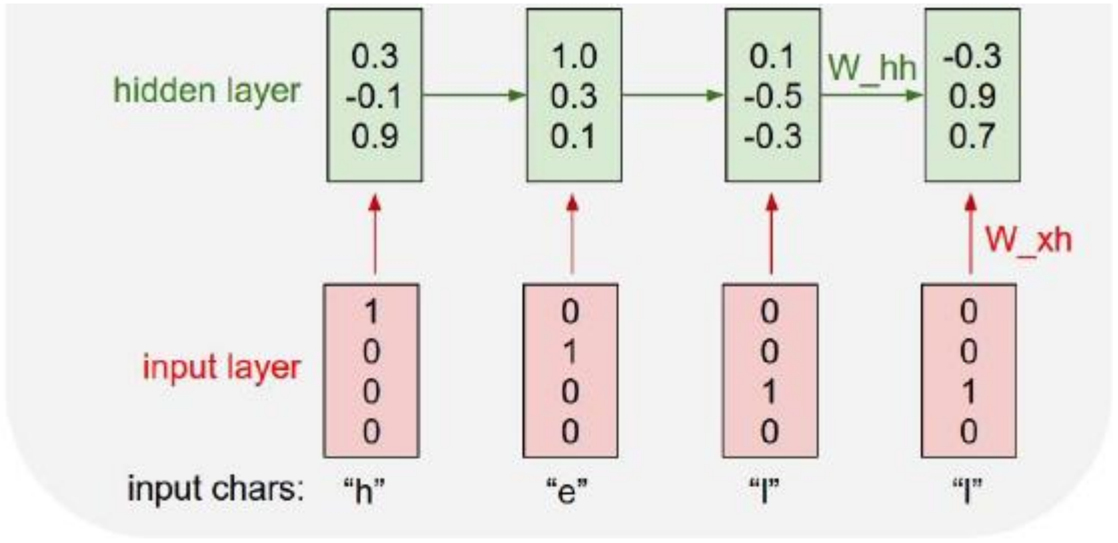


Example: Character-level language model

- Given previous words/characters, predict the next
- Vocabulary: [h, e, l, o]
- Example of training sequence:

  "Hello"

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



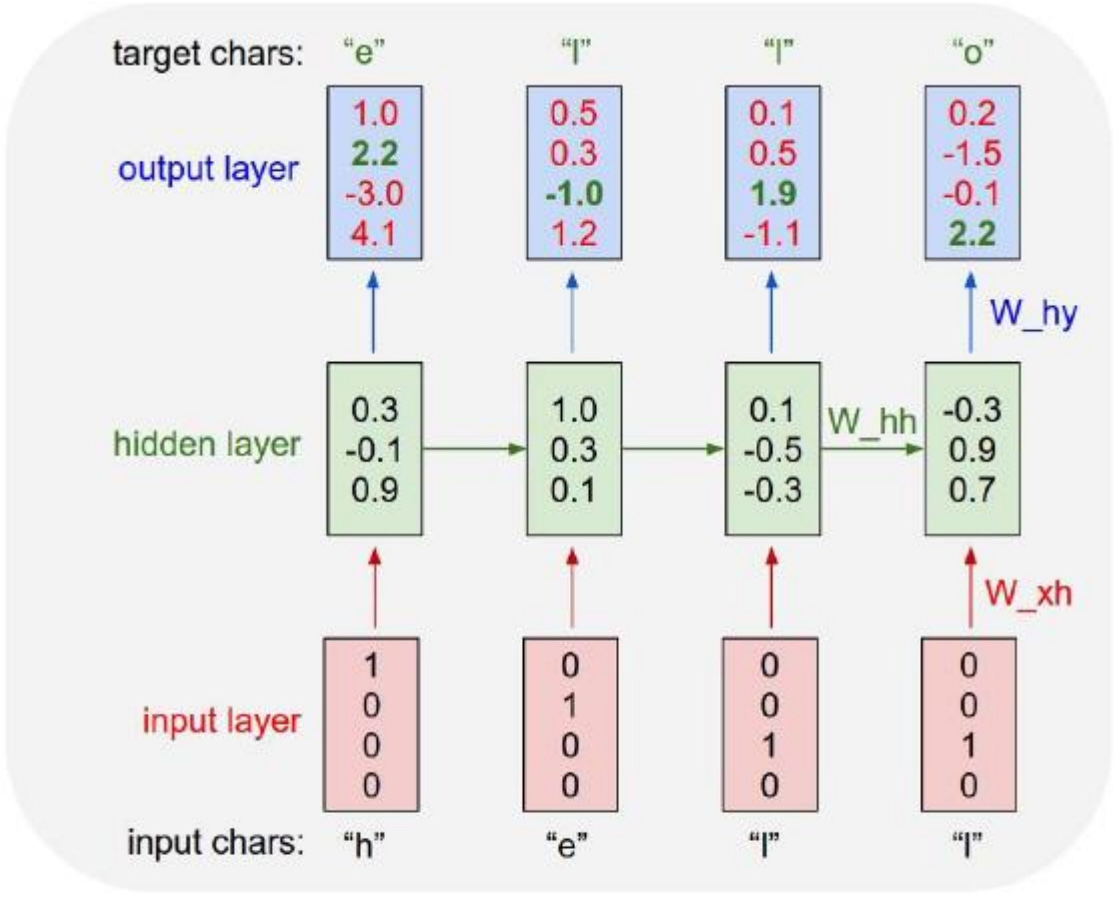
Example: Character-level language model

- Given previous words/characters, predict the next
- Vocabulary: [h, e, l, o]
- Example of training sequence:

  “Hello”

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

$$y_t = W_{hy}h_t$$

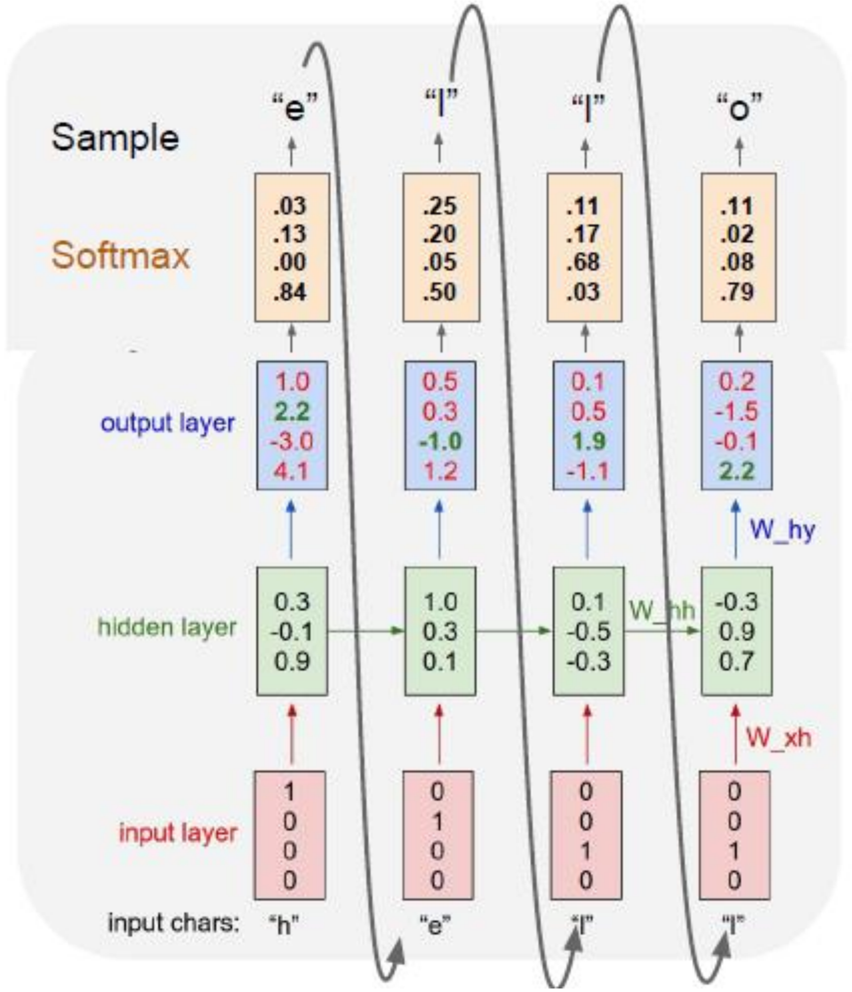


Example: Character-level language model

- Given previous words/characters, predict the next
- Vocabulary: [h, e, l, o]
- Example of training sequence:

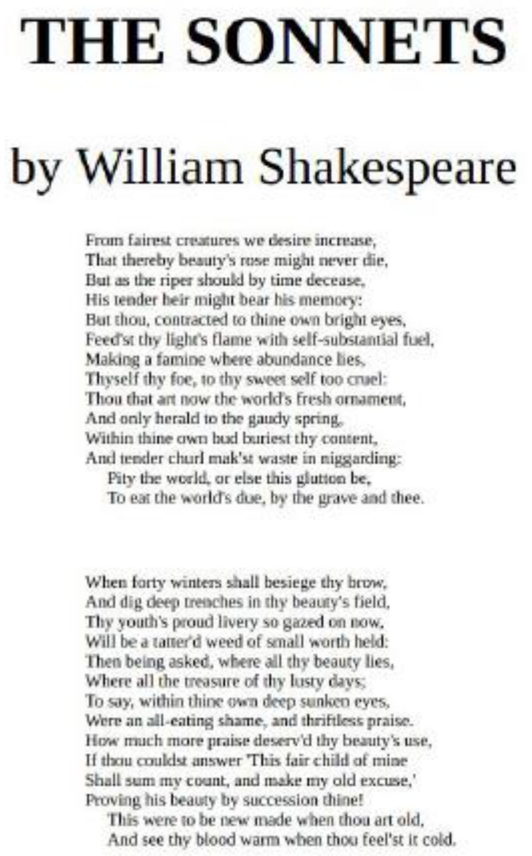
  “Hello”

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



Example of text generating

- Can we generate sonnets like a poet, using RNN?

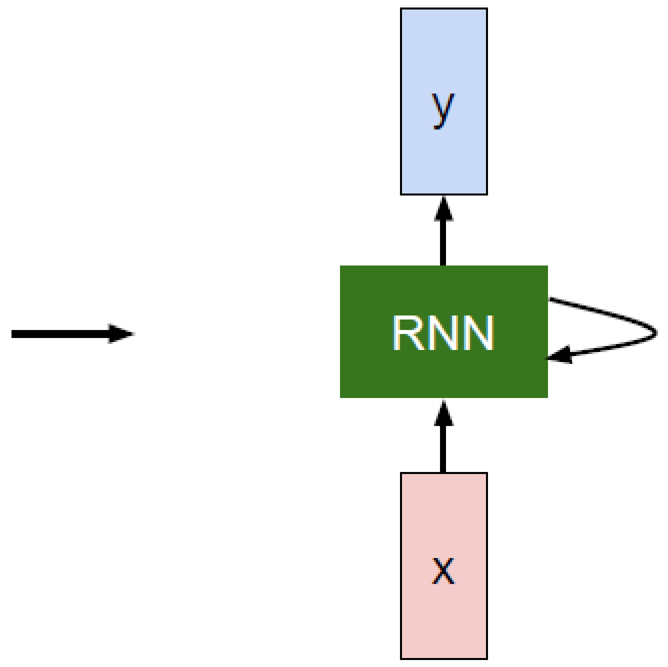


THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.



Example result

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

↓ train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

↓ train more

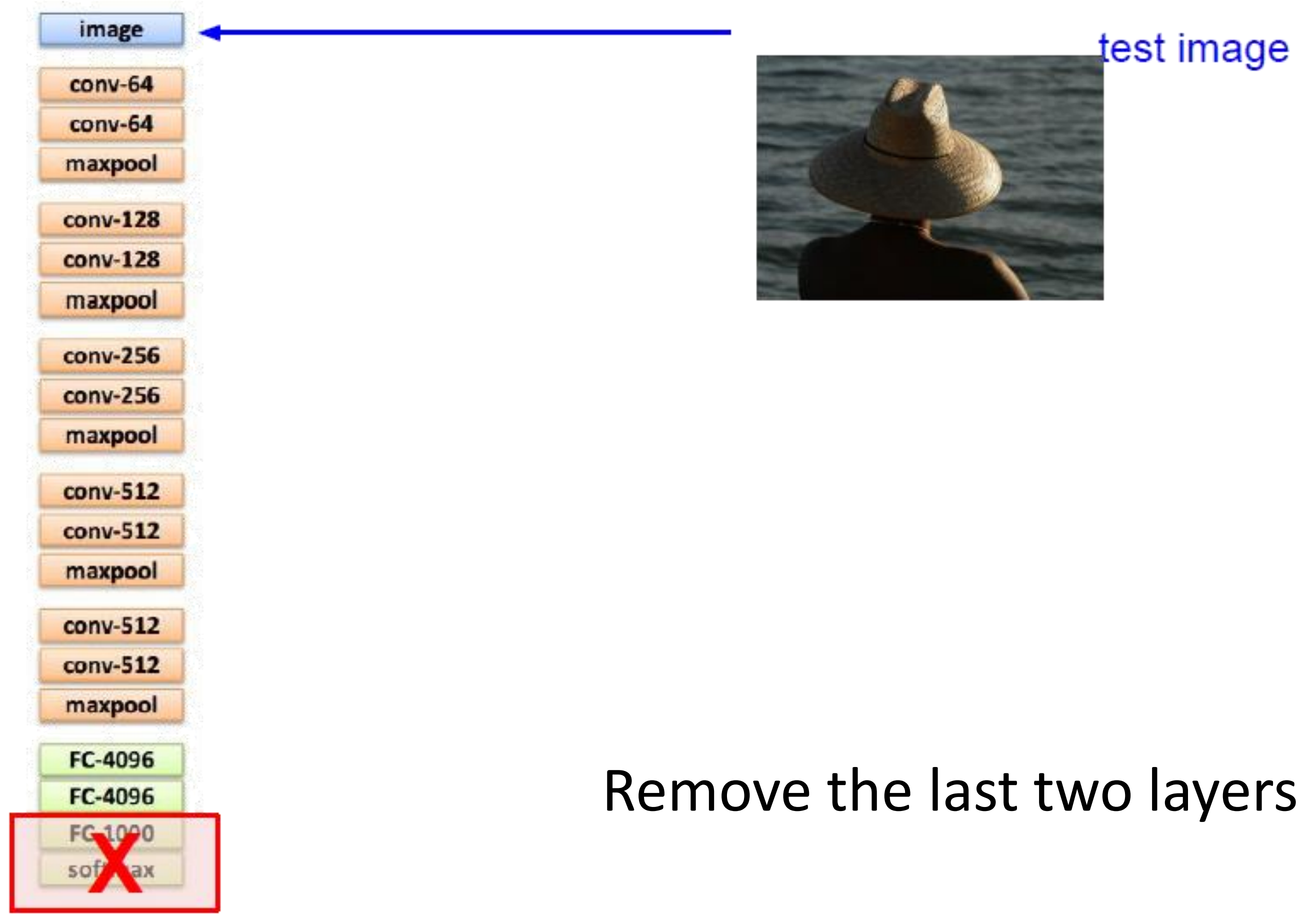
```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

↓ train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

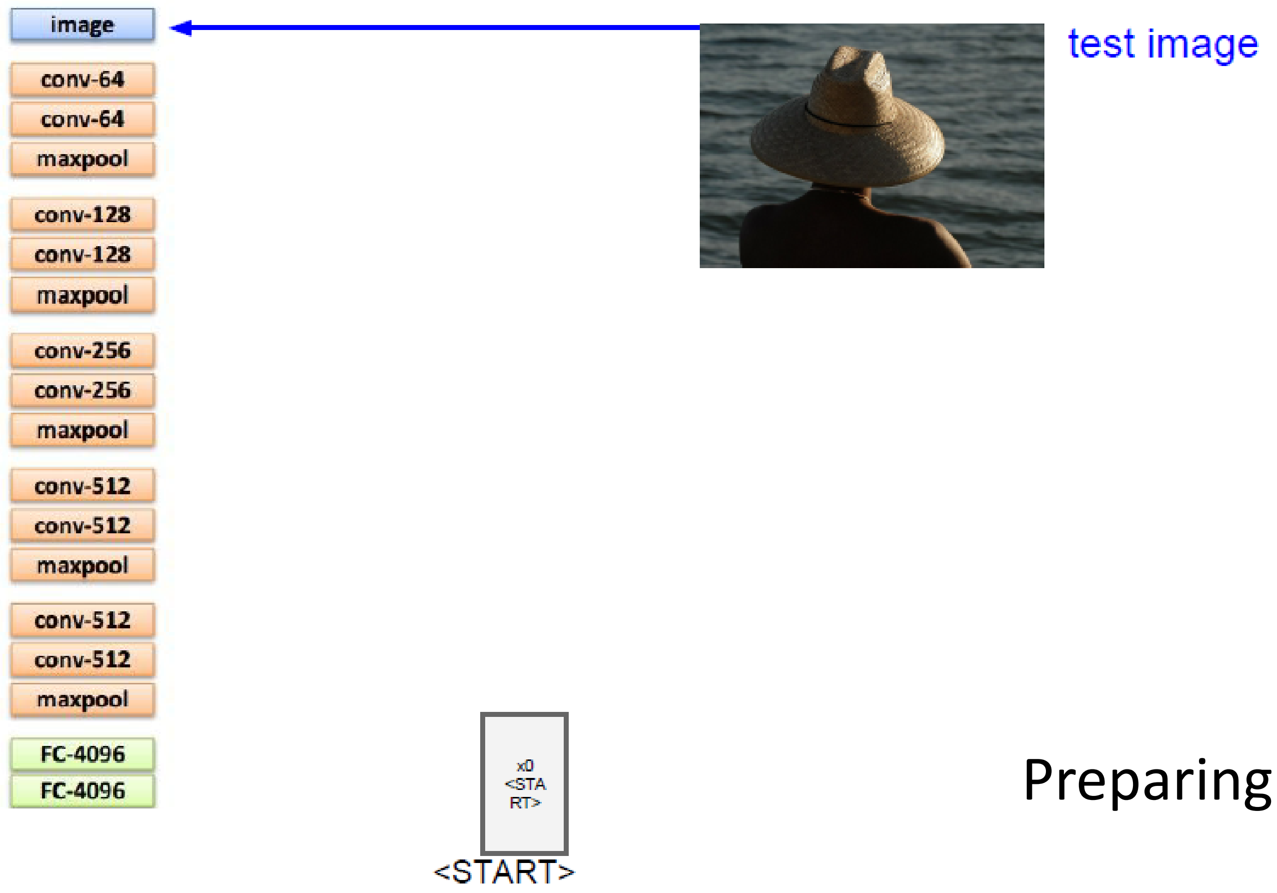
Image captioning

- Using RNN to explain the content in an Image



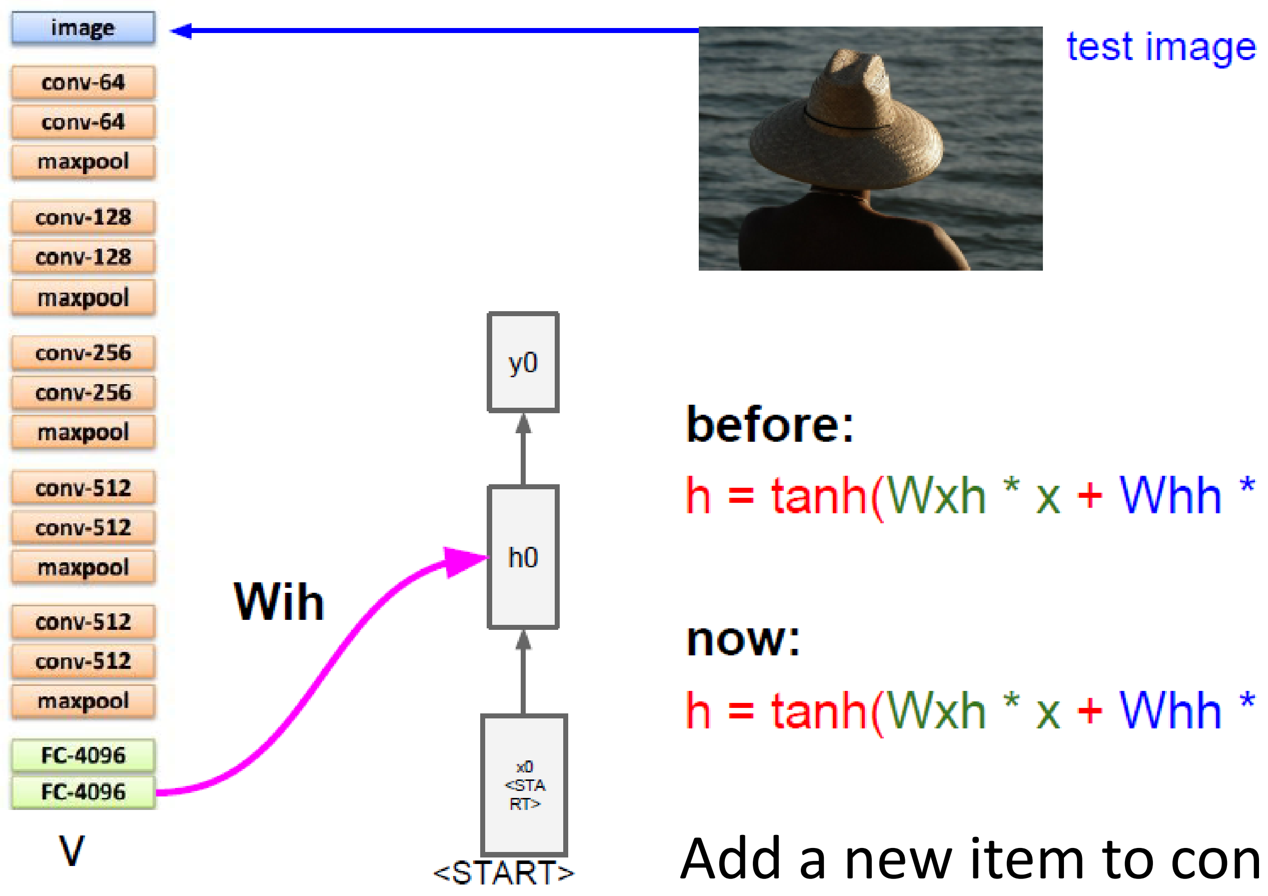
Remove the last two layers in CNN

Image captioning



Preparing the initial input for RNN

Image captioning



before:

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

now:

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

Add a new item to connect CNN to RNN

Image captioning

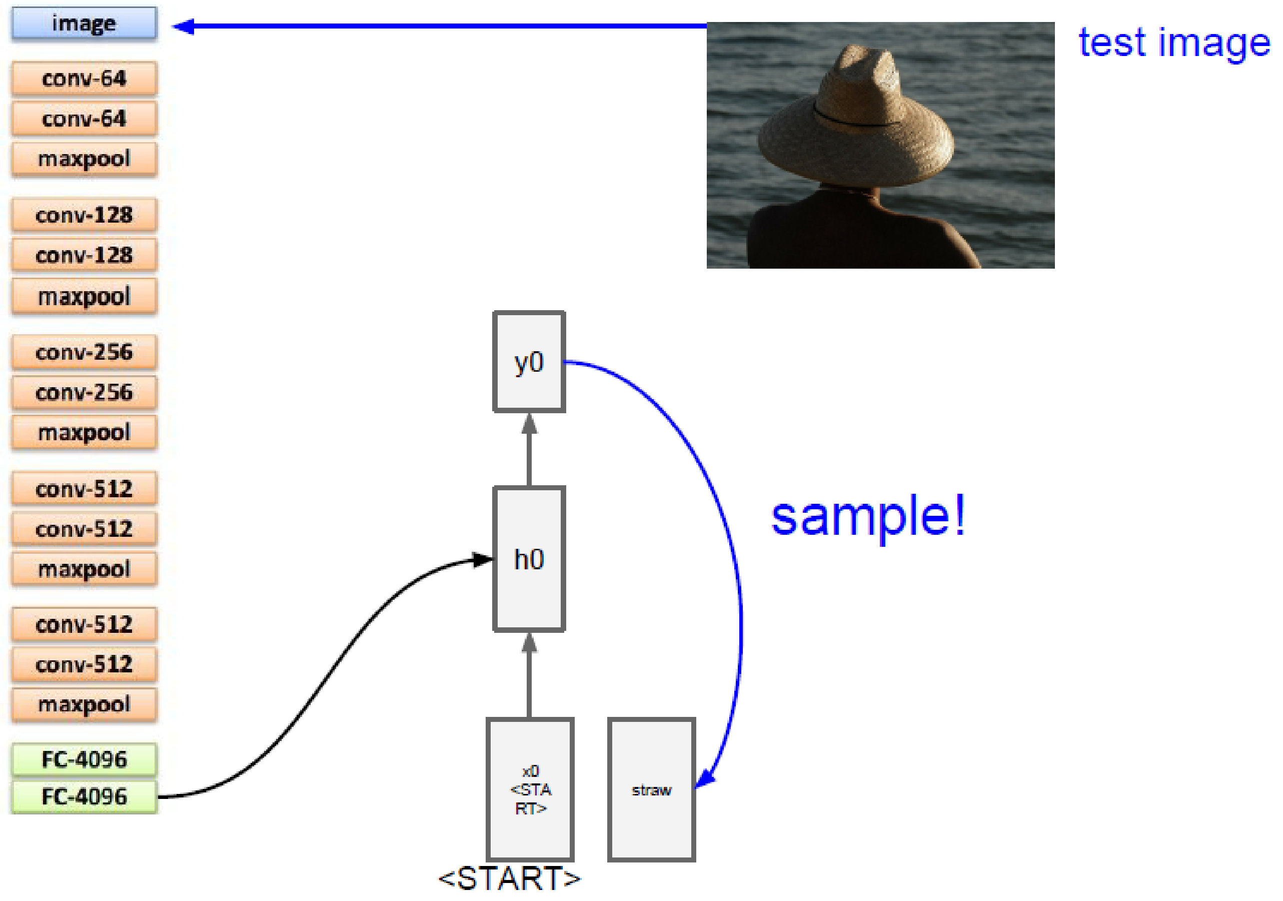


Image captioning

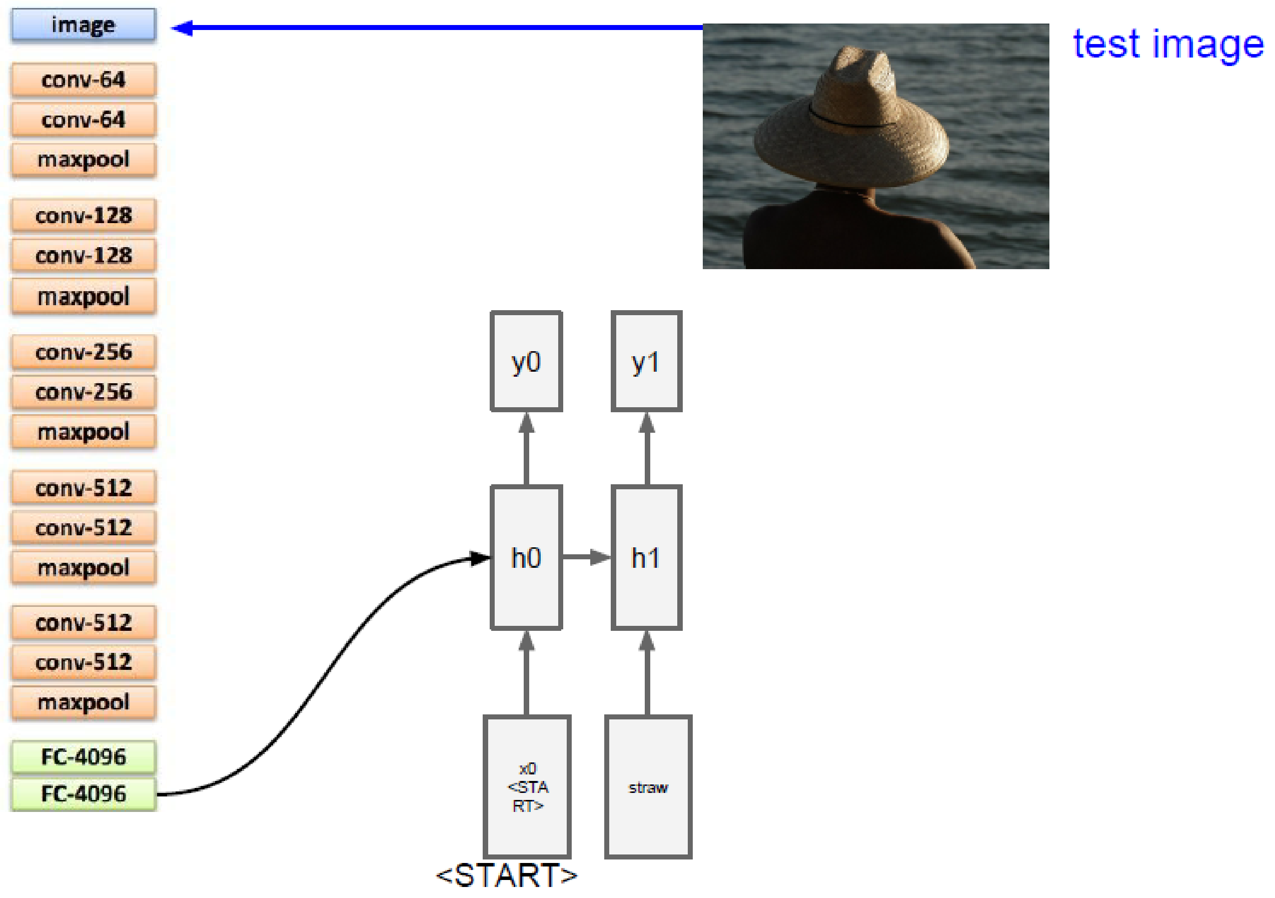


Image captioning

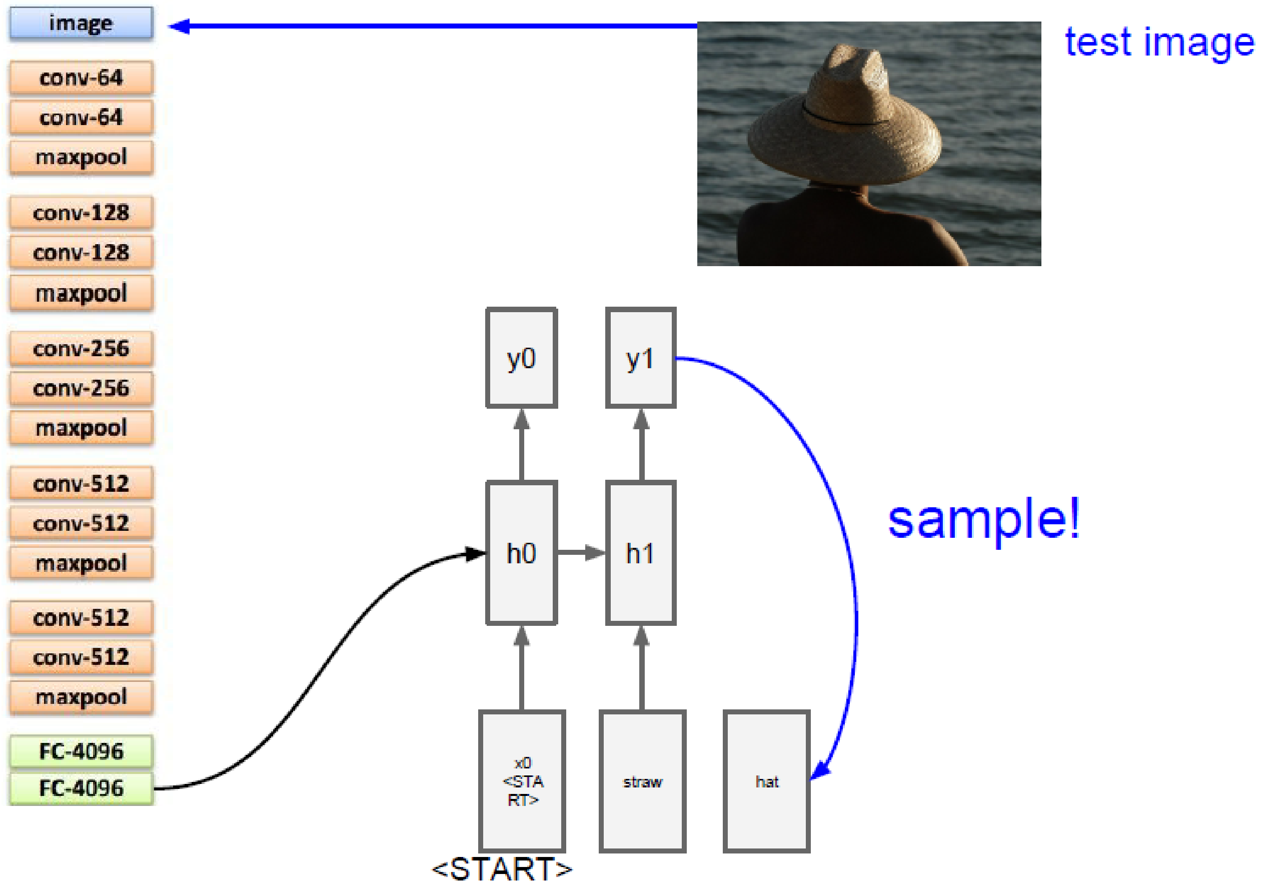


Image captioning

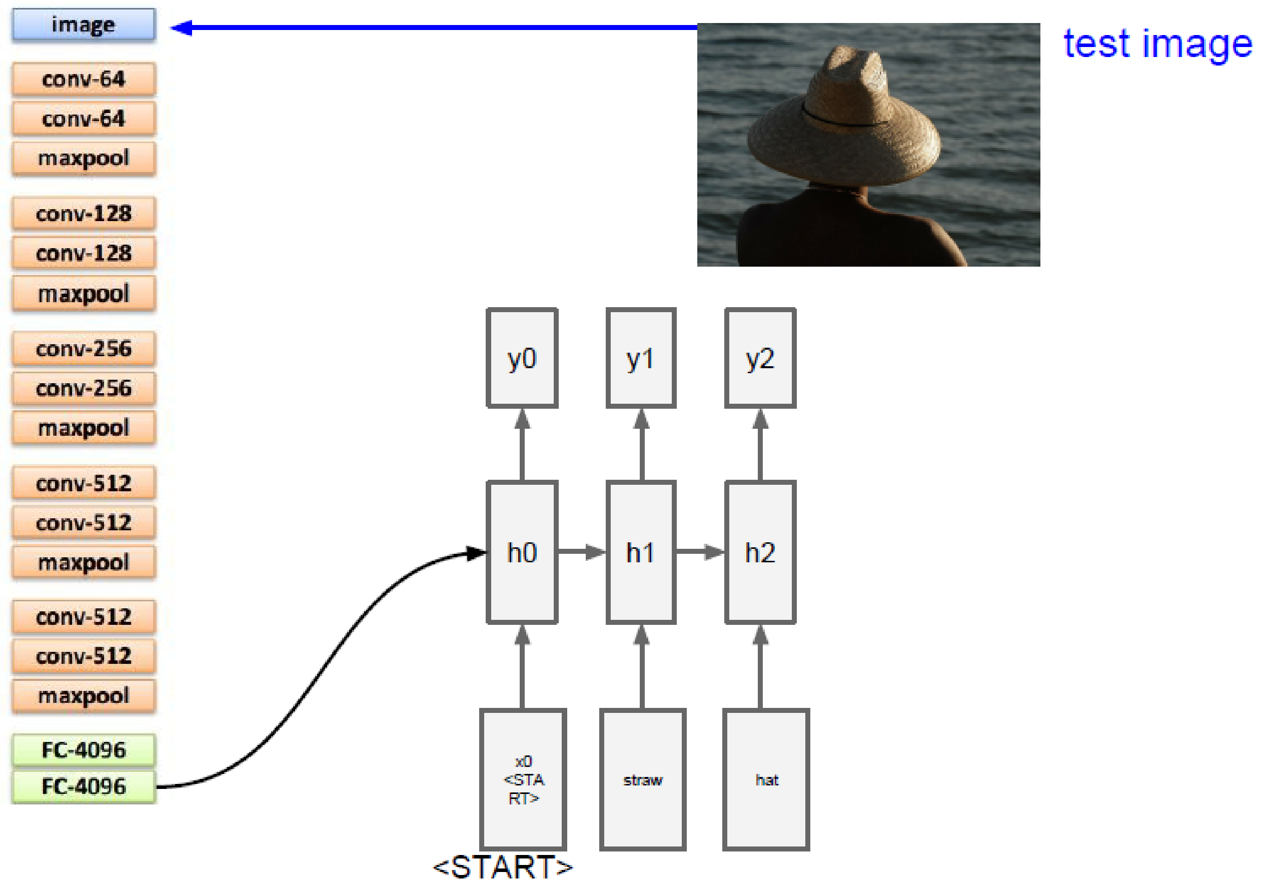


Image captioning

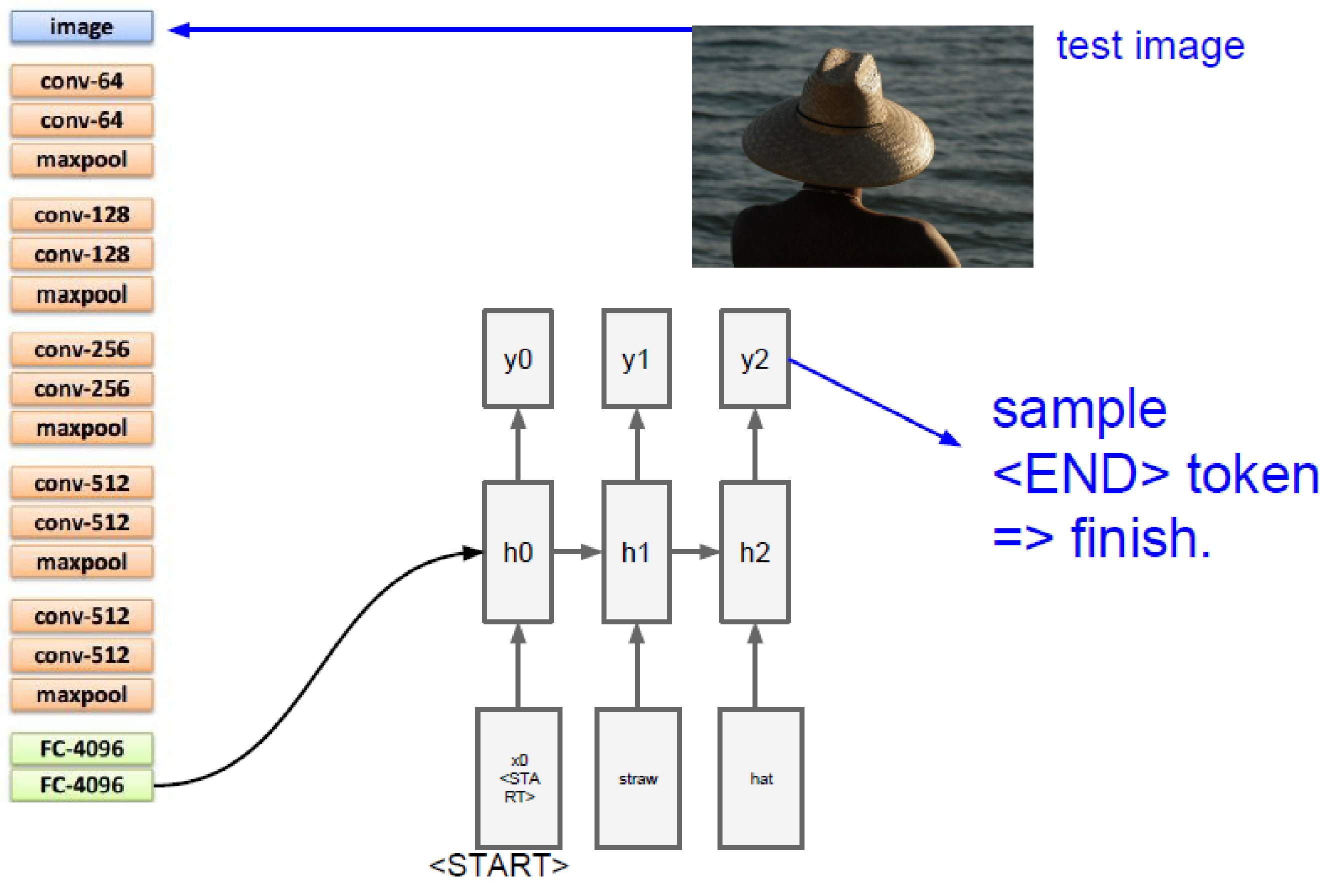


Image captioning: Example results

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

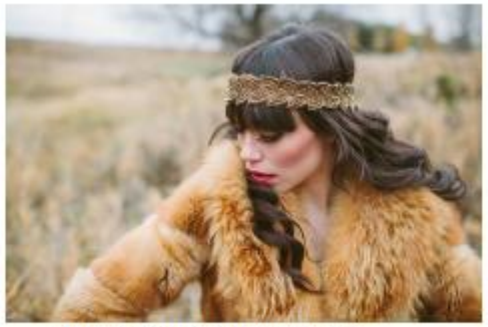
Two people walking on the beach with surfboards

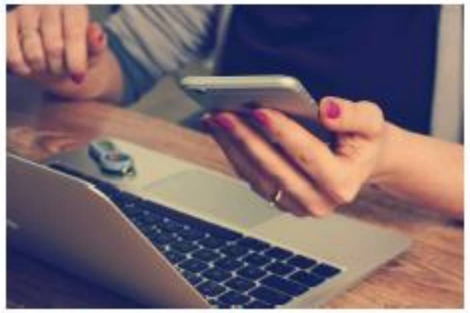
A tennis player in action on the court

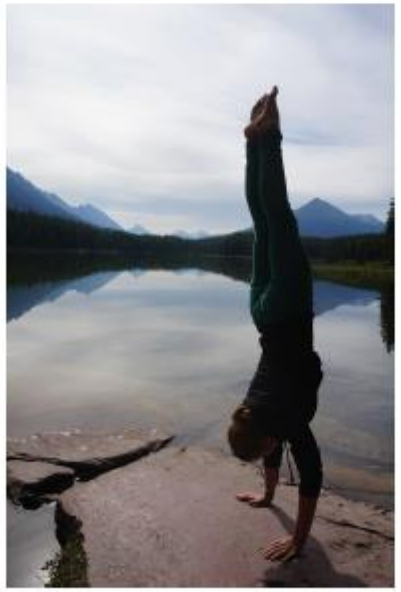
Two giraffes standing in a grassy field

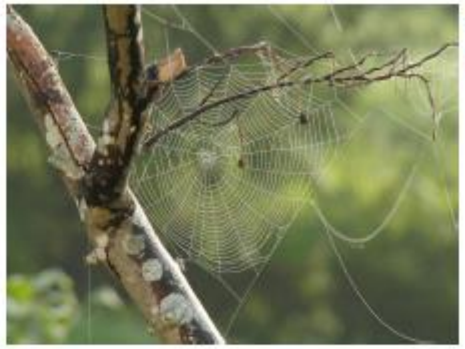
A man riding a dirt bike on a dirt track

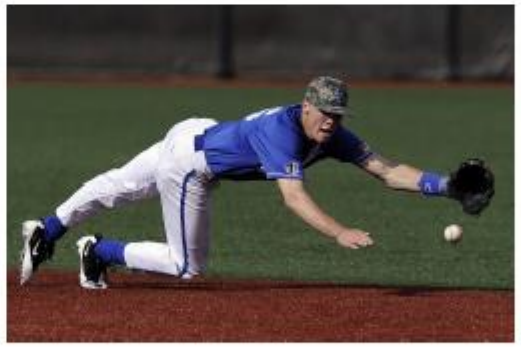
Image captioning: Failure cases


A woman is holding a cat in her hand


A person holding a computer mouse on a desk

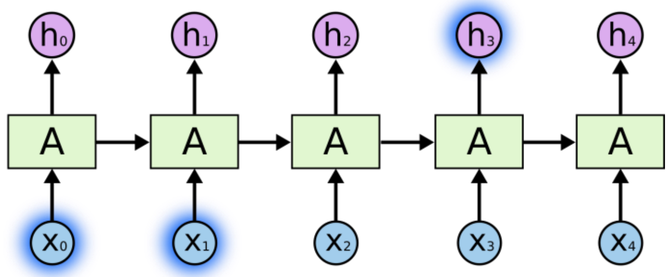

A woman standing on a beach holding a surfboard

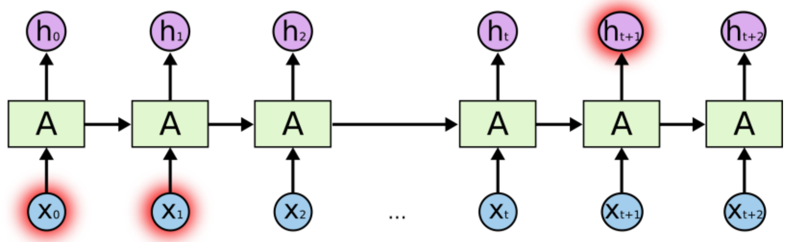

A bird is perched on a tree branch


A man in a baseball uniform throwing a ball

The problem of long-term dependencies

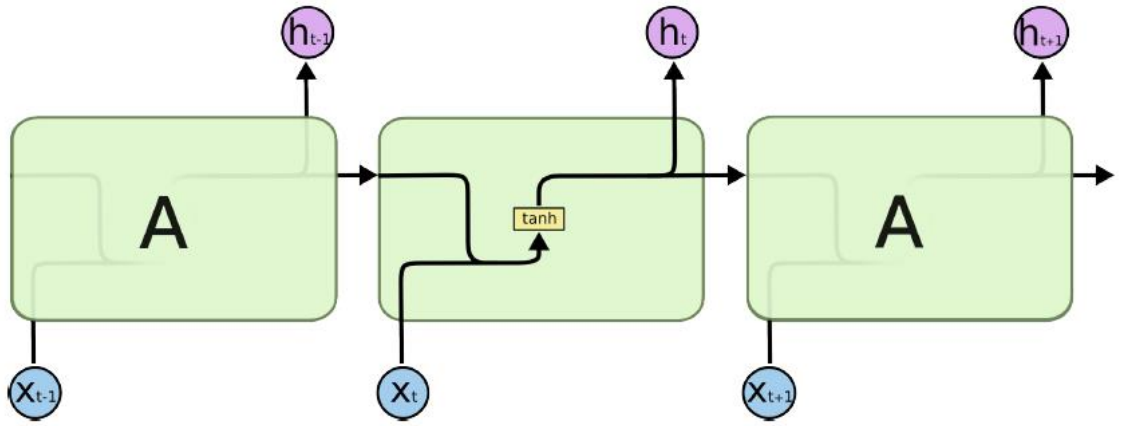
- One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame
- RNN usually works well for recent information
  - E.g. The clouds are in the *sky*.
- But might work worse for further back context
  - E.g. I grew up in France… I speak fluent *French*.





Long short-term memory (LSTM)

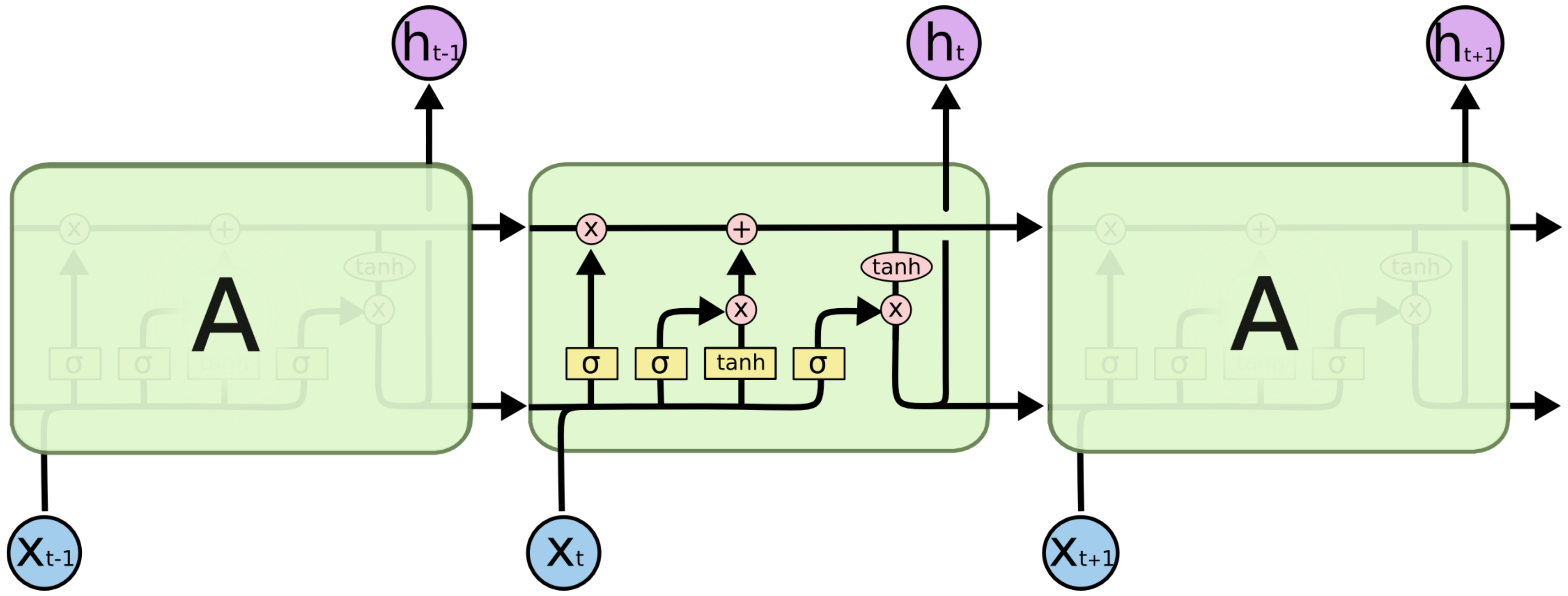
- Hochreiter & Schmidhuber [1997]
- A variant of RNN, capable of learning long-term dependencies
- Can remember information for long periods of time

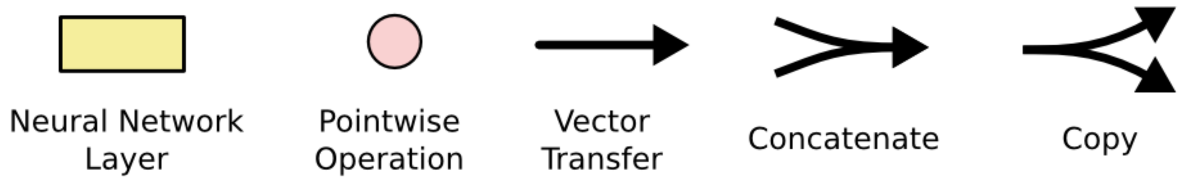


Remember the architecture of RNN cells, we will make many changes on it

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

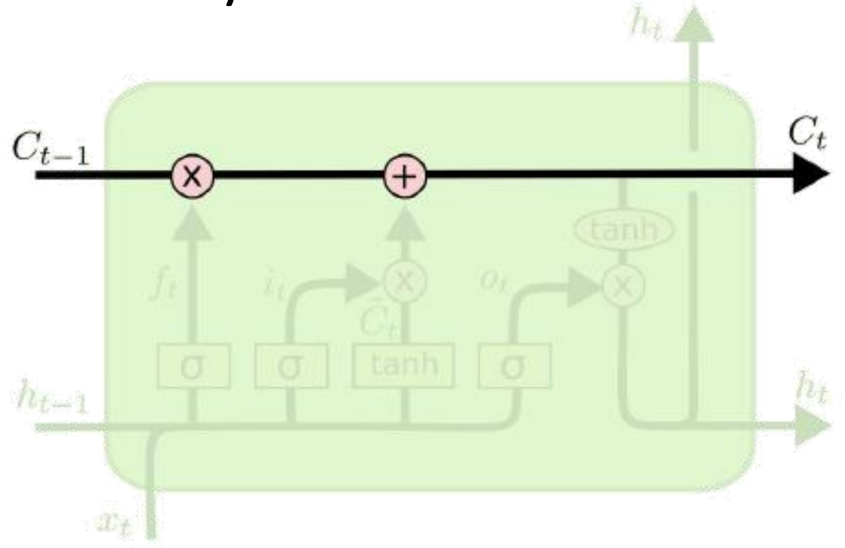
LSTM at a glance





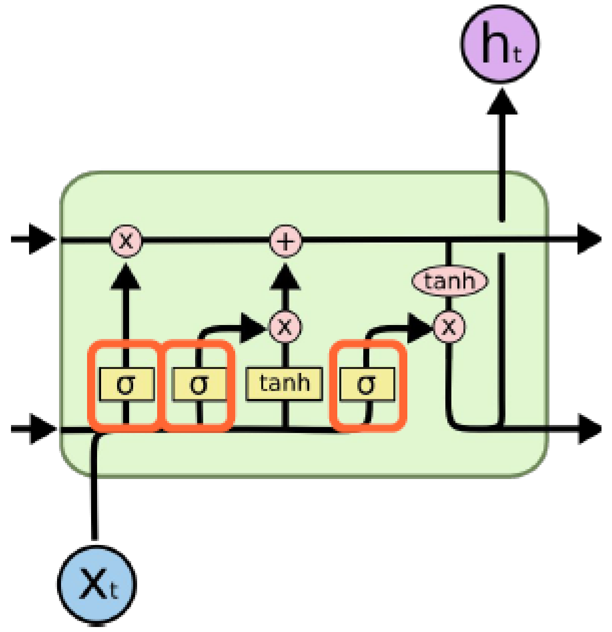
LSTM

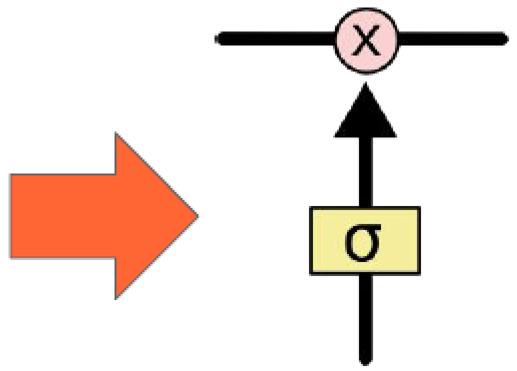
- C_t is the cell state
  - Horizontal line running through the top of the diagram
  - Like a conveyer
  - Run straight down the entire chain, with only some minor linear interactions
  - LSTM uses gates to carefully remove or add information to the cell state

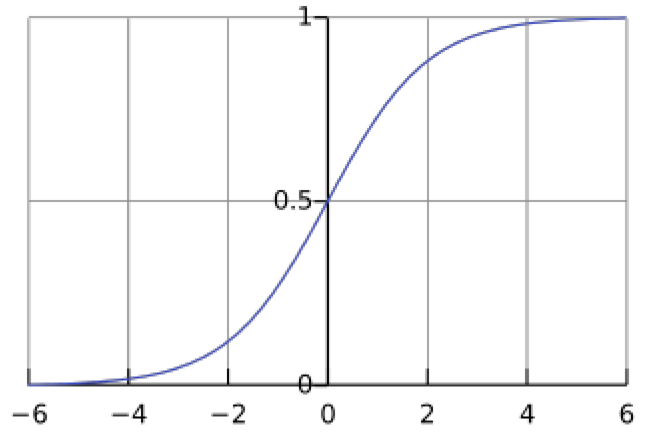


LSTM (cont.)

- Three gates are governed by *sigmoid* units
- Describing how much of each component should be let through
- “let nothing through” (zero value)/“let everything through” (one value)

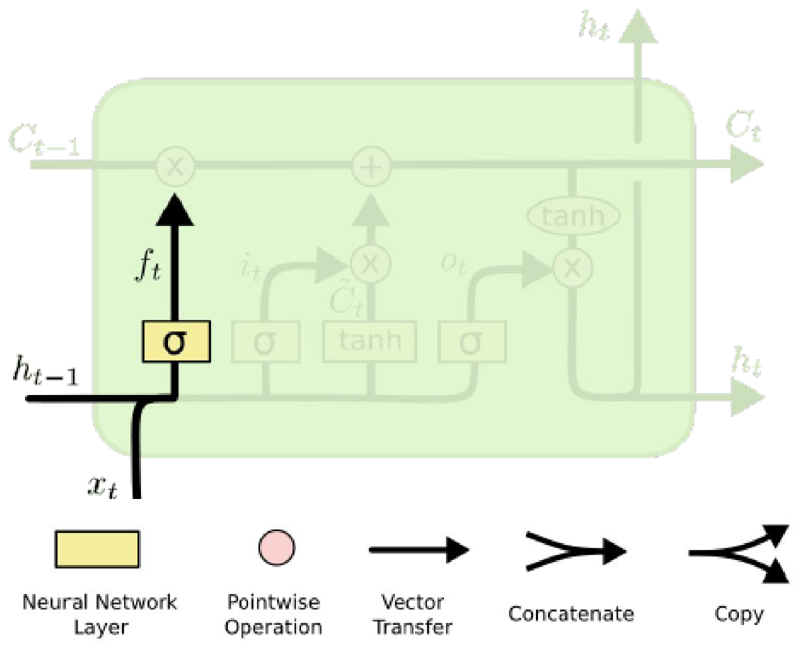






LSTM (cont.)

- The first gate is forget gate layer
  - Decide what information we're going to throw away from the cell state
  - Output a number in $(0,1)$ for each number in the cell state
  - 1/0: completely keep/remove this

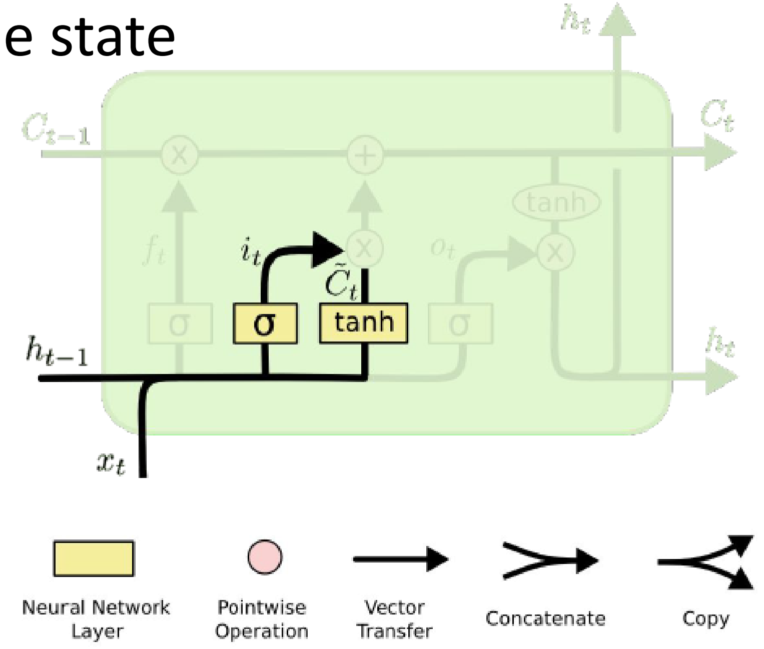


Forget Gate:

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

LSTM (cont.)

- Next is to decide what new information we’re going to store in the cell state
  - The sigmoid layer is the input gate layer, deciding which values we’ll update
  - The tanh layer creates a vector of new candidate values $\tilde{C}_t$ that should be added to the state



Input Gate Layer

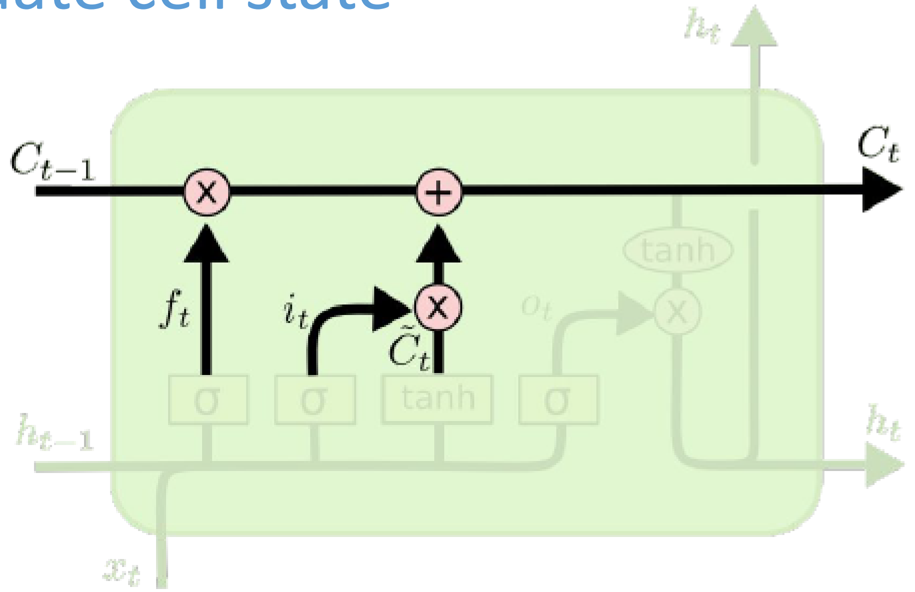
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

New contribution to cell state

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

LSTM (cont.)

- Multiply the old state by f_t
  - Forgetting the things from previous state
- Add new candidate values
- Update cell state

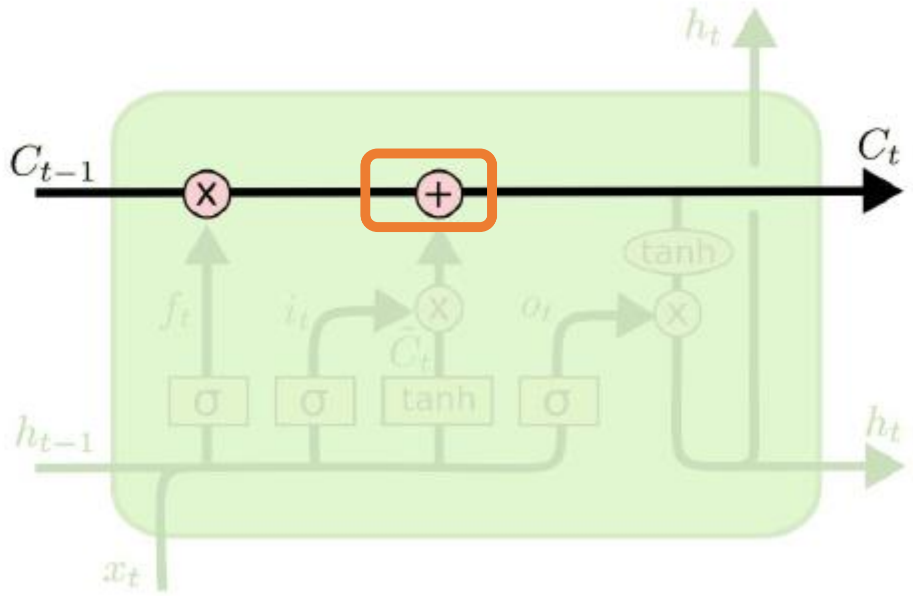


Update Cell State (memory):

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

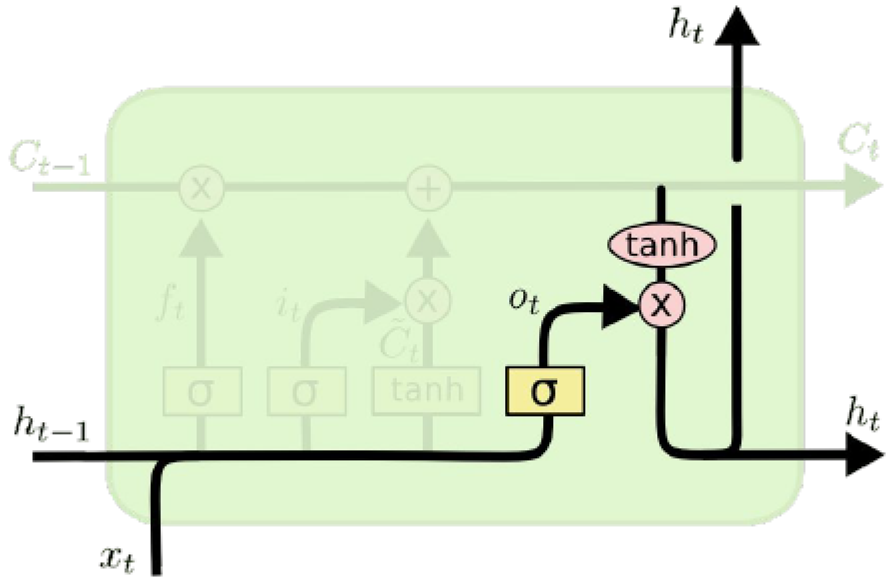
LSTM (cont.)

- The sum operation avoids memory vanishing and exploding/vanishing backpropagation gradients, which are limitations of RNN



LSTM (cont.)

- Last decide the output h_t
  - Filtered (tanh) version of the cell state
  - The sigmoid layer decides what parts of the cell state we're going to output h_{t-1}, x_t are like *context*



Output Gate Layer

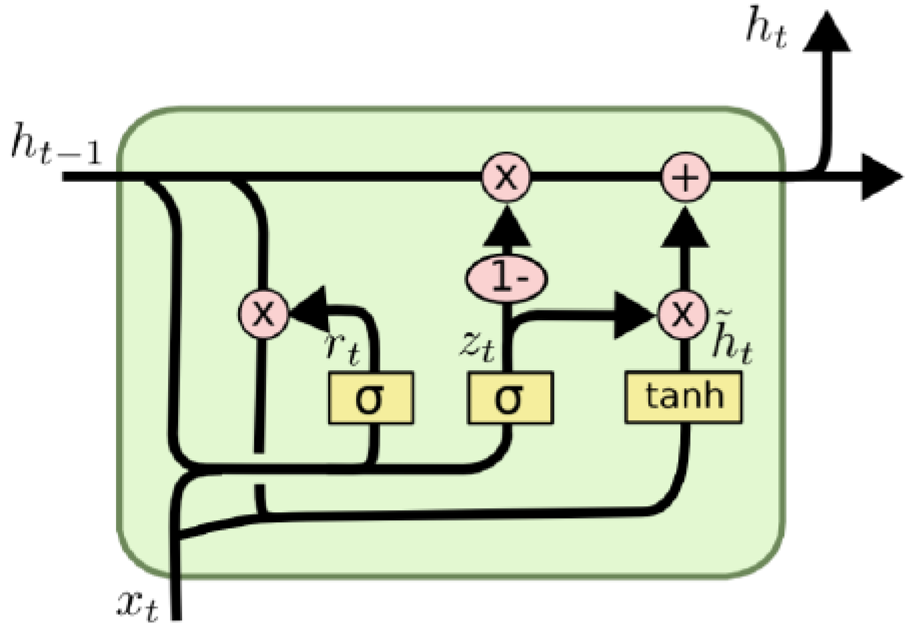
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

Output to next layer

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

Gated recurrent unit (GRU)

- GRU is a variants of LSTM
  - Combines the forget and input gates into a single "update gate"
  - Also merges the cell state and hidden state



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

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

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

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

LSTM hyperparameter tuning

- Watch out for *overfitting*
- Regularization helps: regularization methods include L1, L2, and dropout among others.
- The larger the network, the more powerful, but also easier to overfit. Don’t want to try to learn a million parameters from 10,000 examples
  - parameters > examples = trouble
- More data is usually better
- Train over multiple epochs (complete passes through the dataset)
- The learning rate is the single most important hyper parameter