

Recurrent Neural Network (RNN)

Computer Vision Group, IIT Patna

Slides prepared by: [Kumari Priya](#)

Presented by: [Dr. Chandranath Adak](#)
chandranath@iitp.ac.in

Examples of sequence data

Speech recognition



The quick brown fox jumps
over the lazy dog

Music generation

Φ



Sentiment classification

“There is nothing to like
in this movie.”



DNA sequence analysis

AGCCCCTGTAGGAAGTAG



AG**CCCCTGTAGGAAGTAG**

Machine translation

你想和我一起唱歌吗?



do you want to sing
with me?

Video activity recognition



Running

Name entity recognition

Yesterday, Harry Potter
met Hermoine Granger.

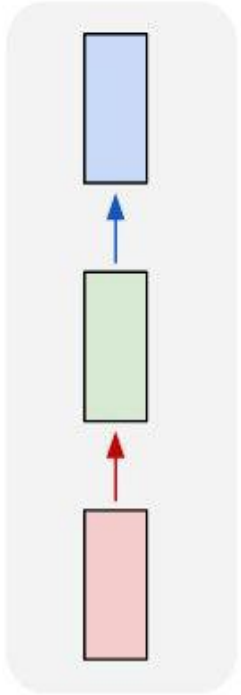


Yesterday, **Harry Potter**
met **Hermione Granger**.

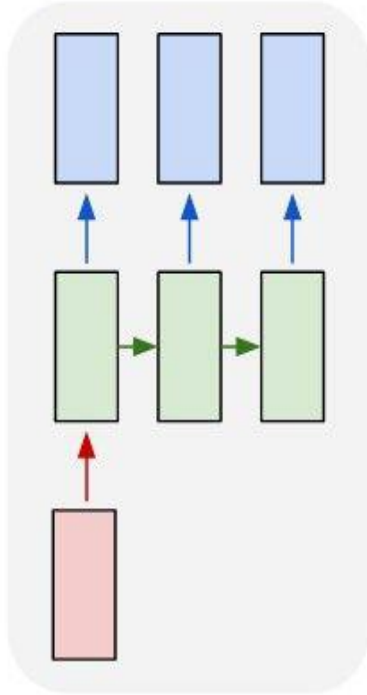
- A recurrent neural network (RNN) is an artificial neural network that uses sequential or time series data.
- These deep learning algorithms are commonly used for ordinal or temporal problems, such as
 - Natural language translation,
 - Natural language processing (NLP),
 - Speech recognition,
 - Image captioning;
- Incorporated into popular applications such as Siri, voice search, and Google Translate.

Process Sequences

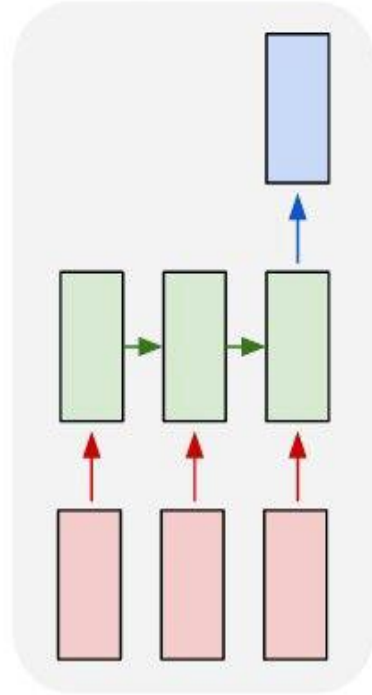
one to one



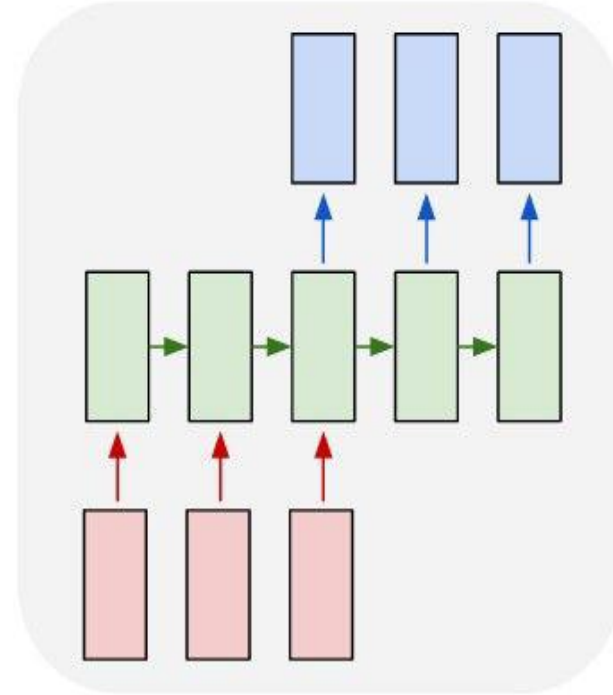
one to many



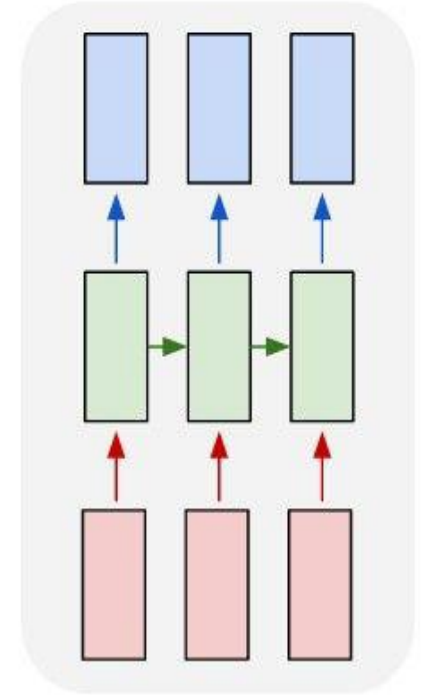
many to one



many to many



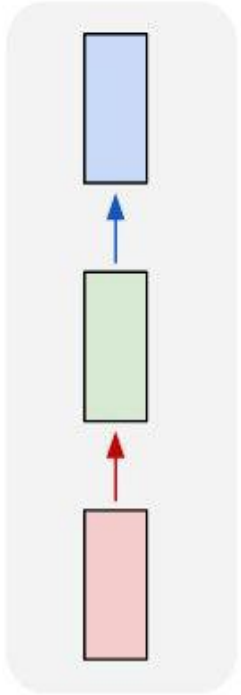
many to many



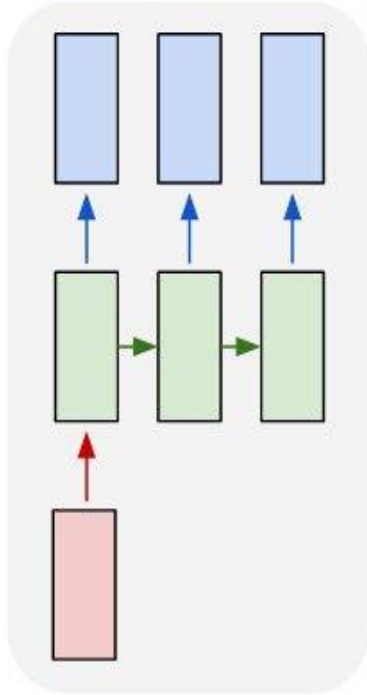
Vanilla Neural Networks

Process Sequences

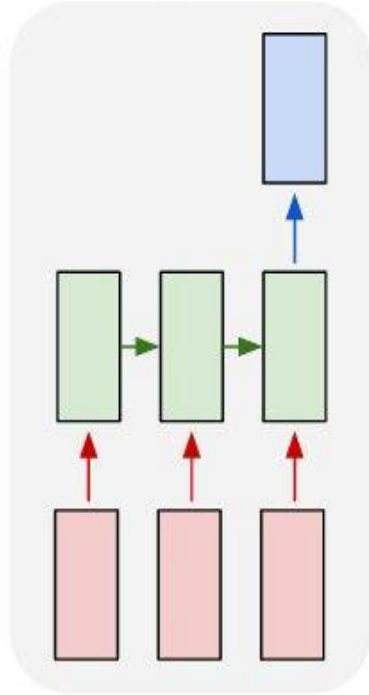
one to one



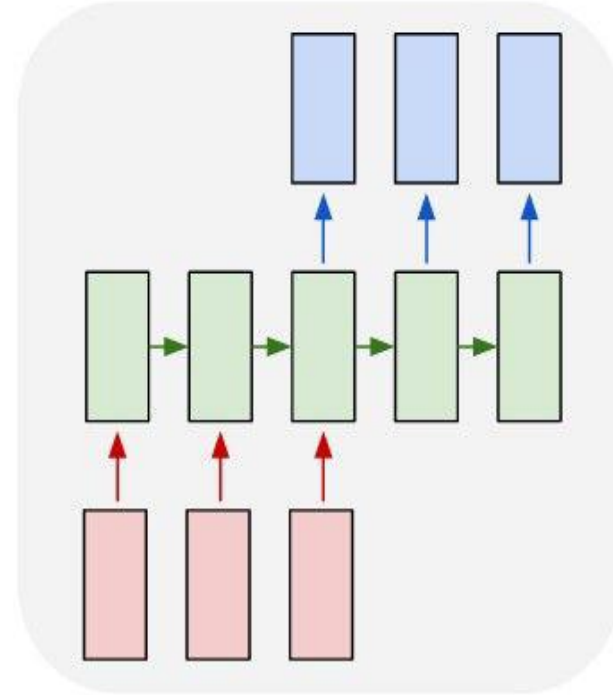
one to many



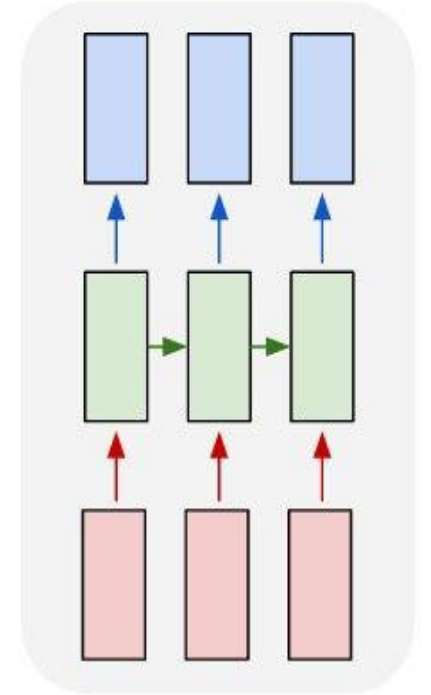
many to one



many to many



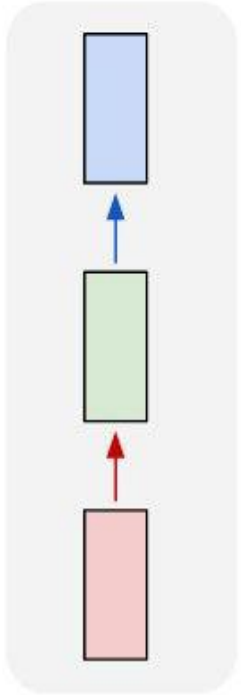
many to many



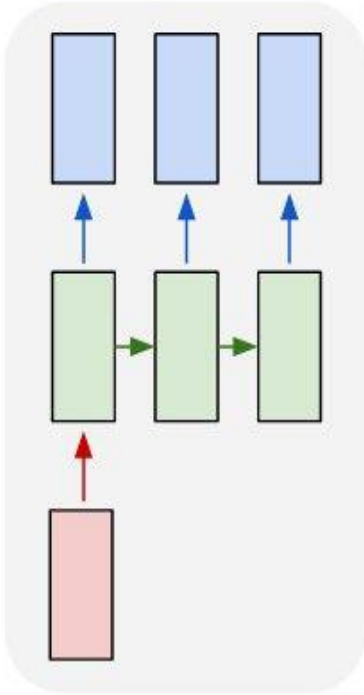
e.g. Image Captioning
image -> sequence of words

Process Sequences

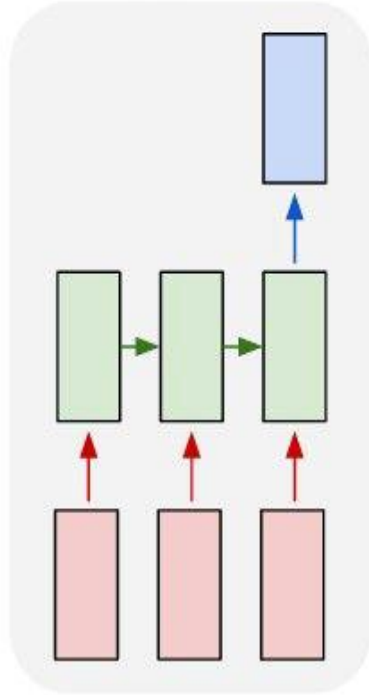
one to one



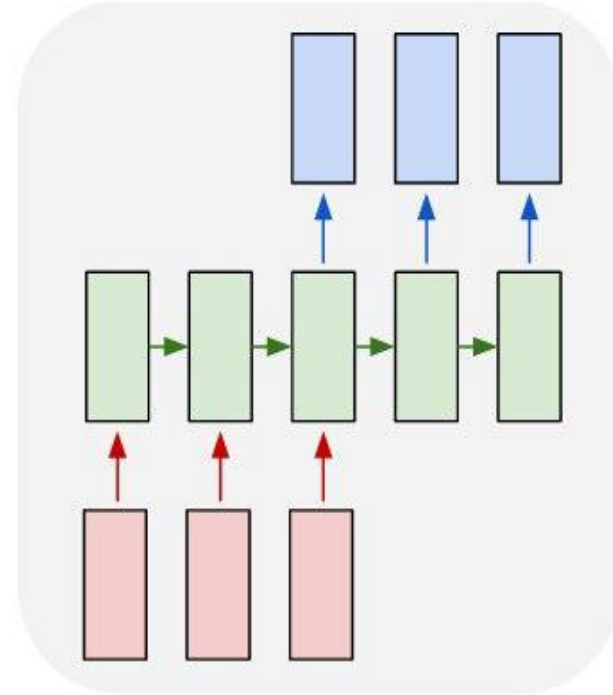
one to many



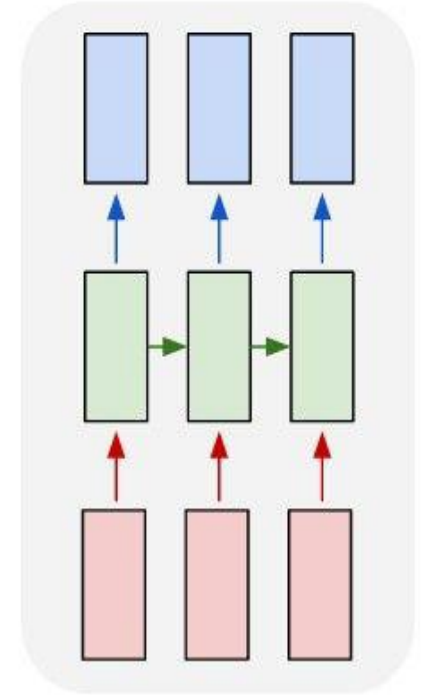
many to one



many to many



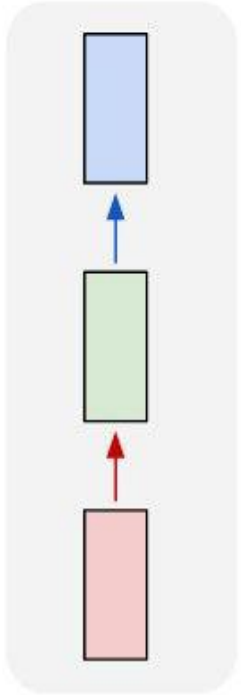
many to many



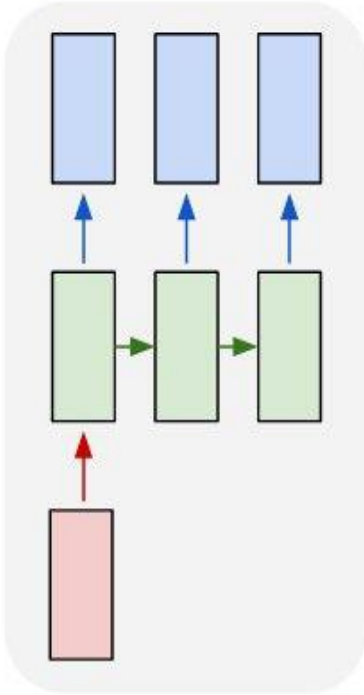
e.g. Sentiment Classification
sequence of words -> sentiment

Process Sequences

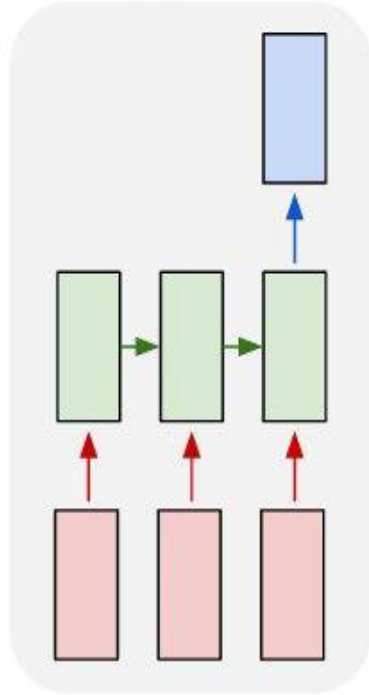
one to one



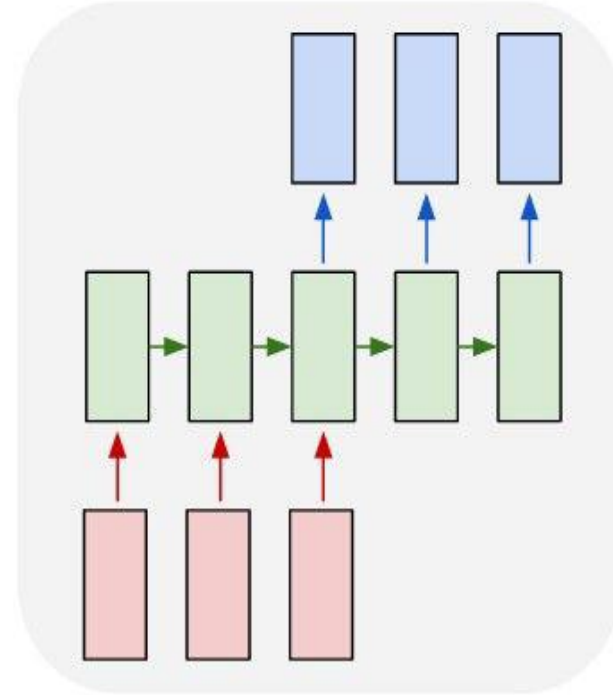
one to many



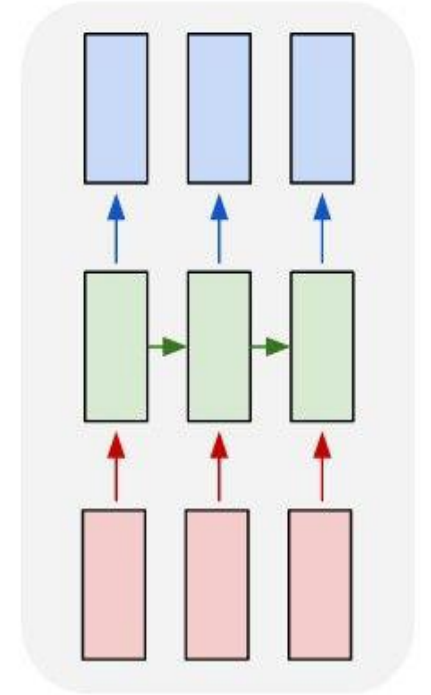
many to one



many to many



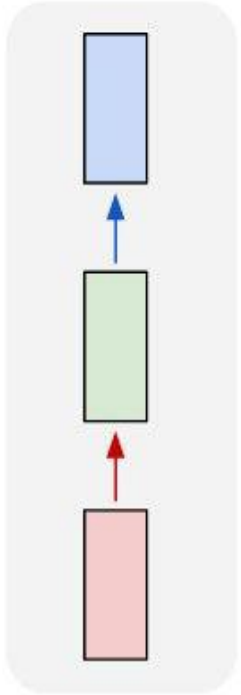
many to many



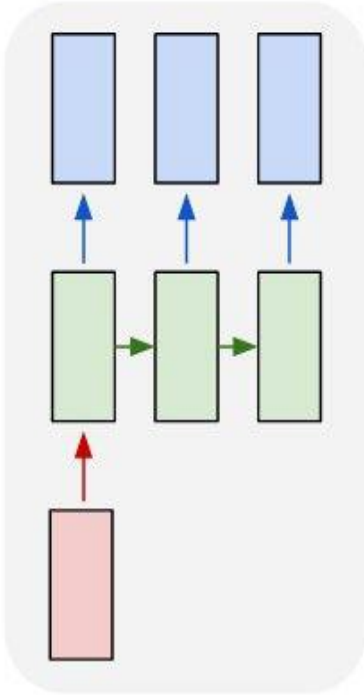
e.g. Machine Translation
seq of words -> seq of words

Process Sequences

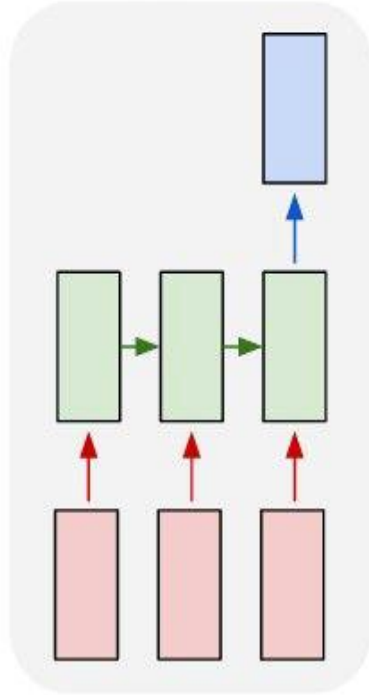
one to one



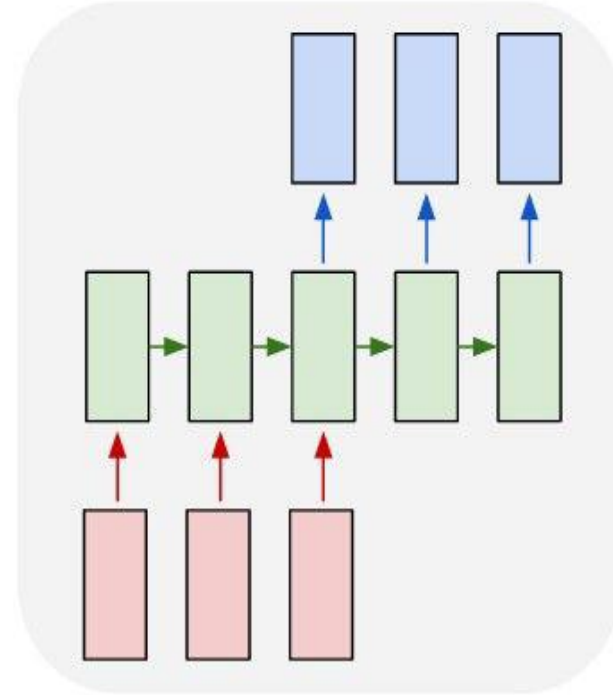
one to many



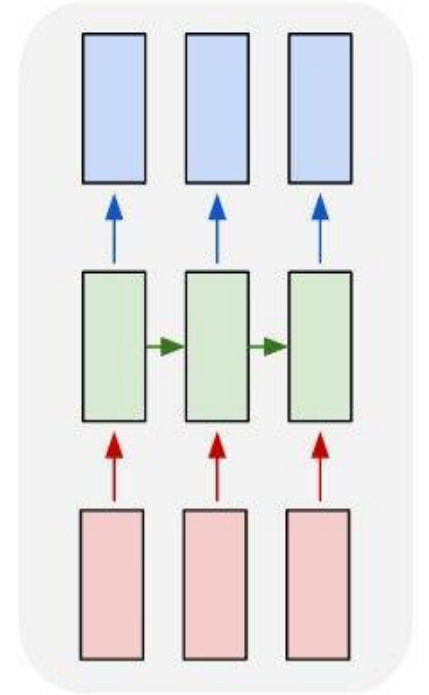
many to one



many to many



many to many



e.g. Video classification on frame level

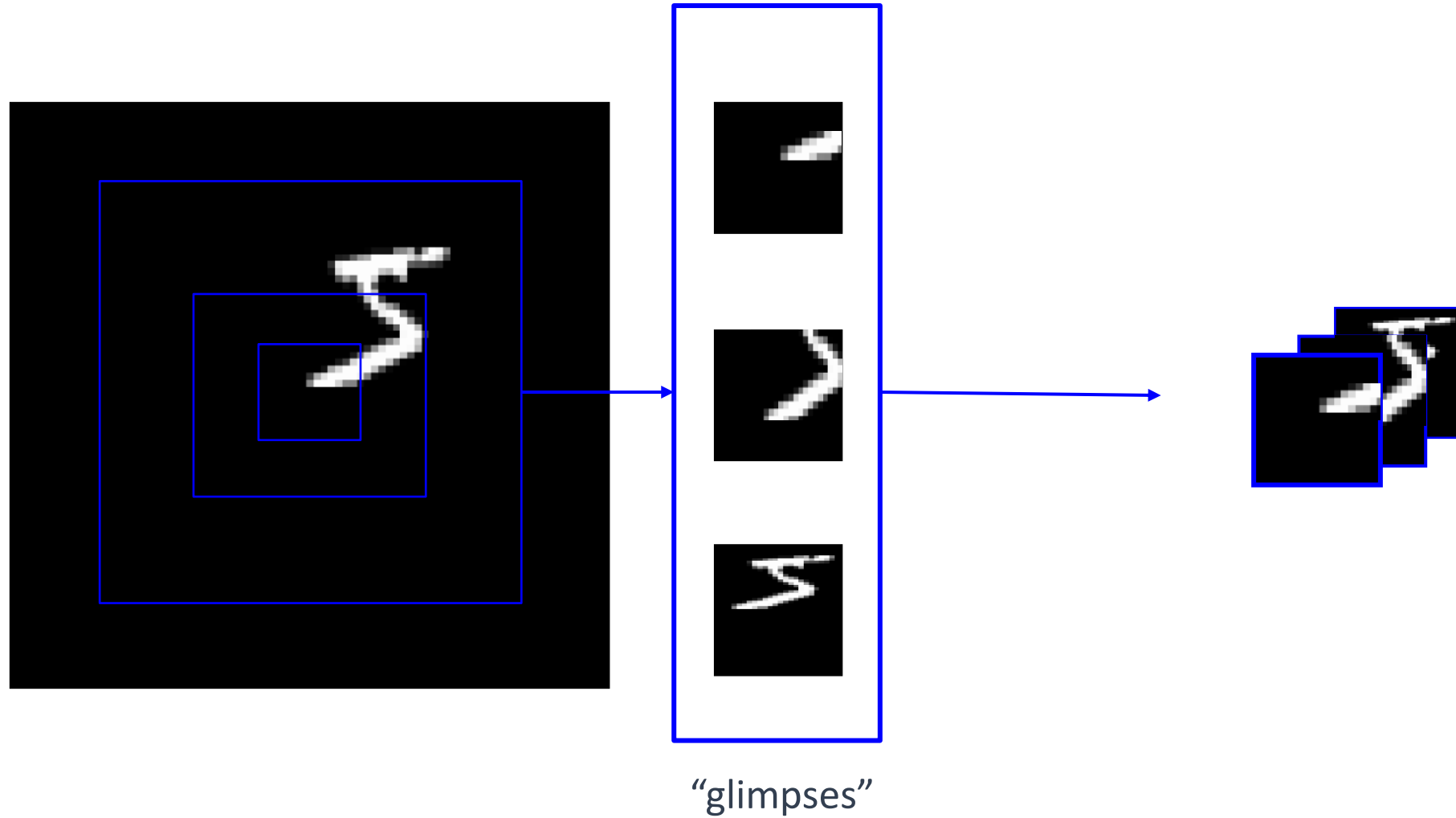


Sequential Processing of Non-Sequence Data

Classify images by taking a series of “glimpses”

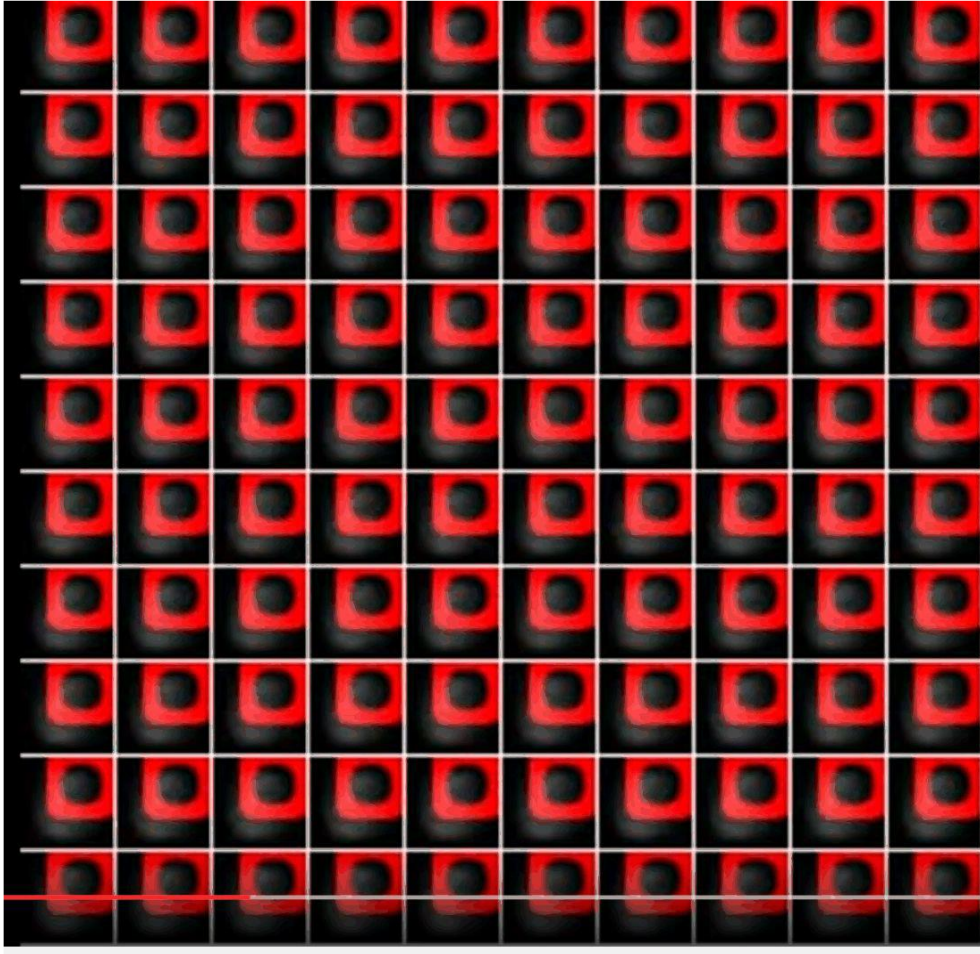


Sequential Processing of Non-Sequence Data

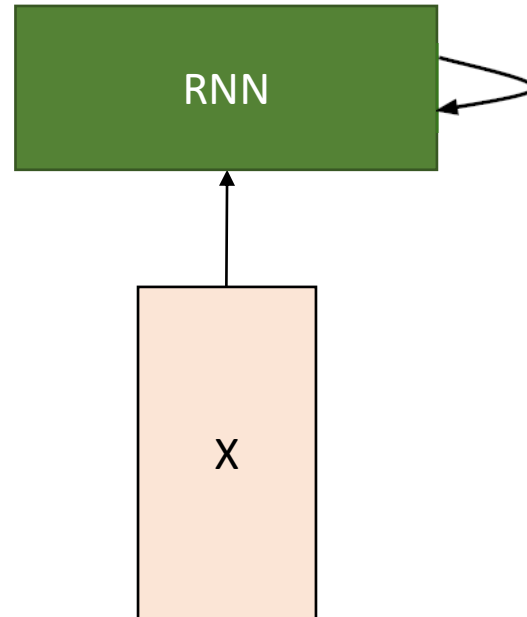


Sequential Processing of Non-Sequence Data

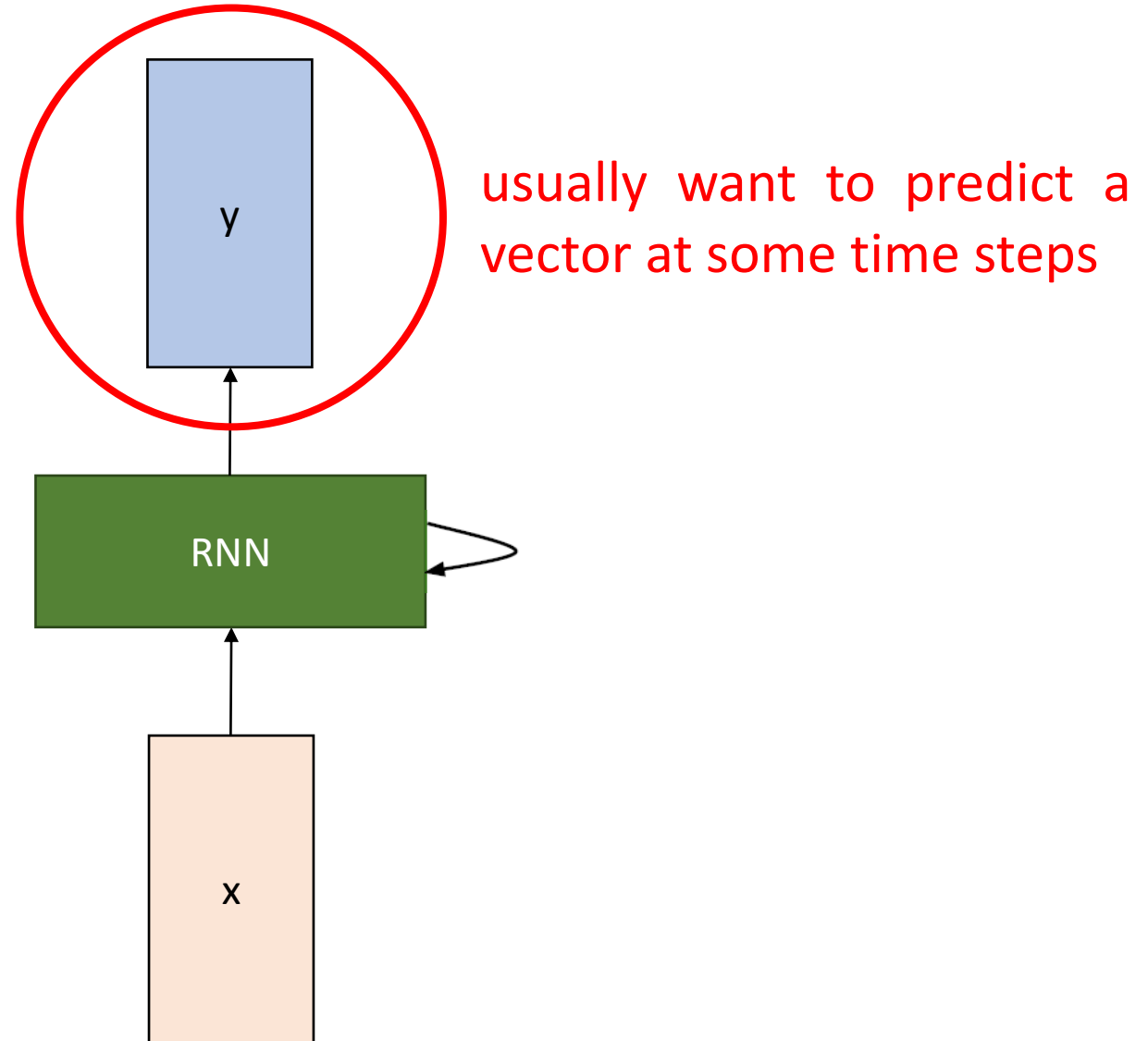
Generate images one piece at a time!



Recurrent Neural Networks



Recurrent Neural Networks



Recurrent Neural Networks

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

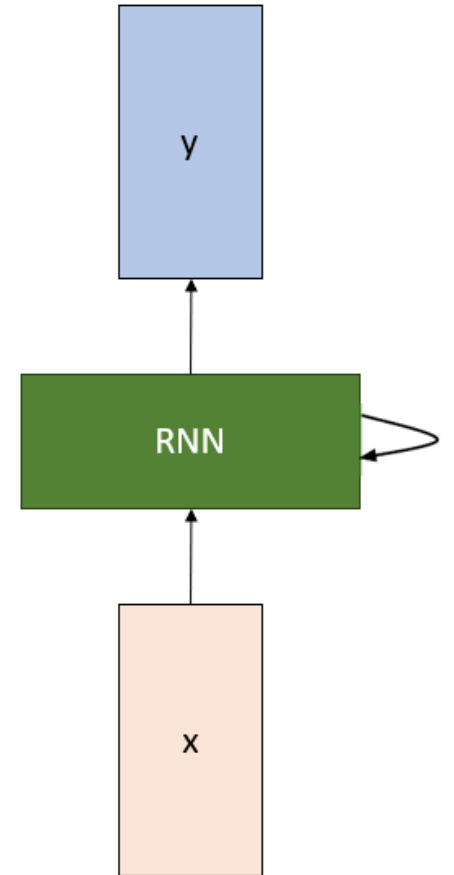
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

old state

input vector at some time step

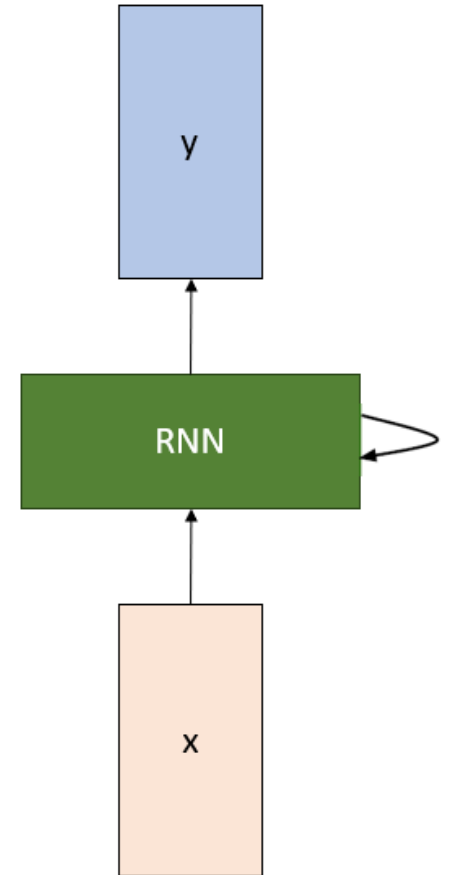


Recurrent Neural Networks

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.



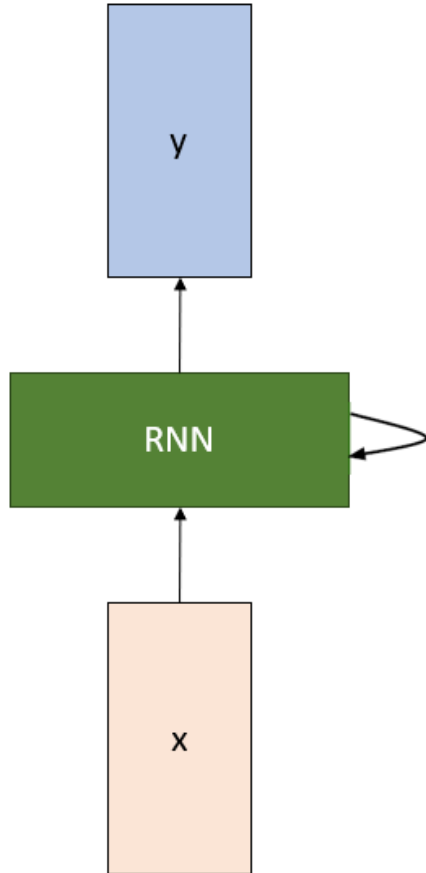
(Vanilla) Recurrent Neural Networks

The state consists of a single “*hidden*” vector \mathbf{h} :

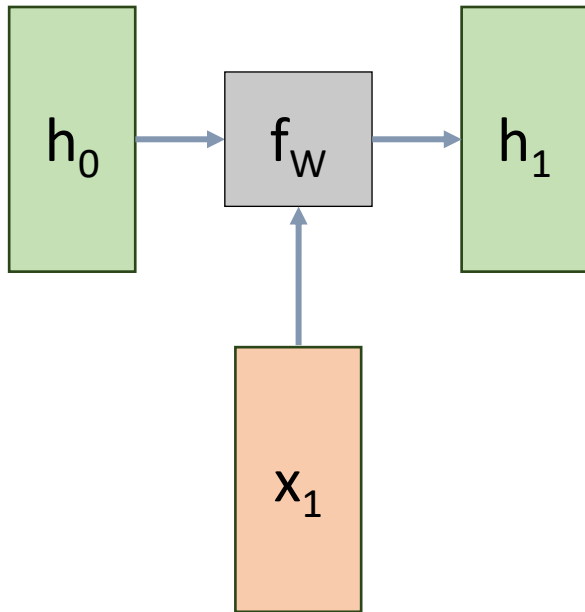
$$h_t = f_W(h_{t-1}, x_t)$$

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

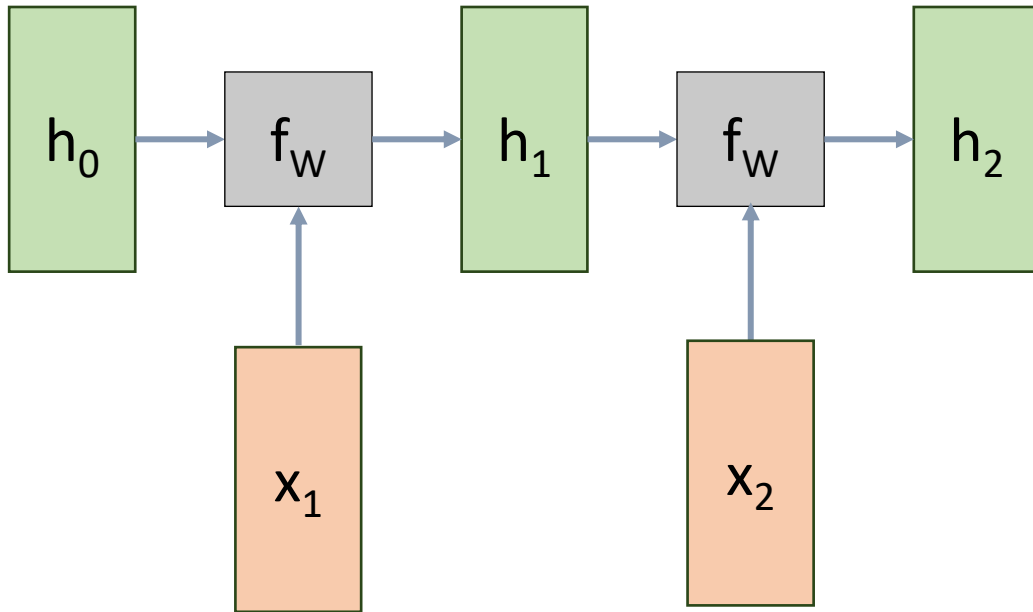
$$y_t = W_{hy}h_t$$



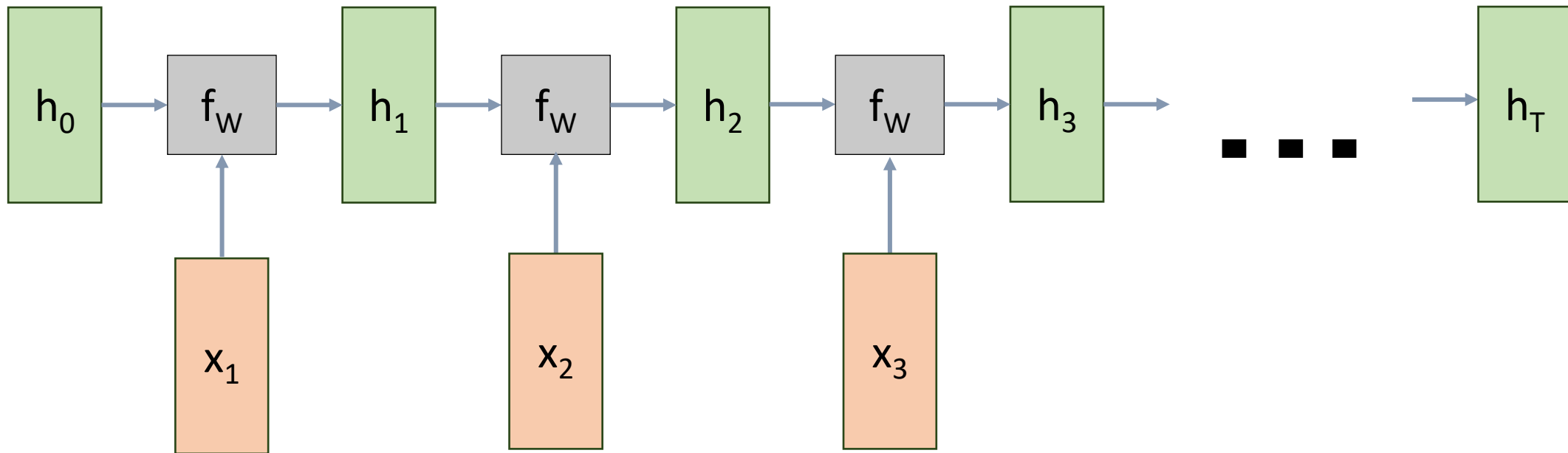
RNN: Computational Graph



RNN: Computational Graph

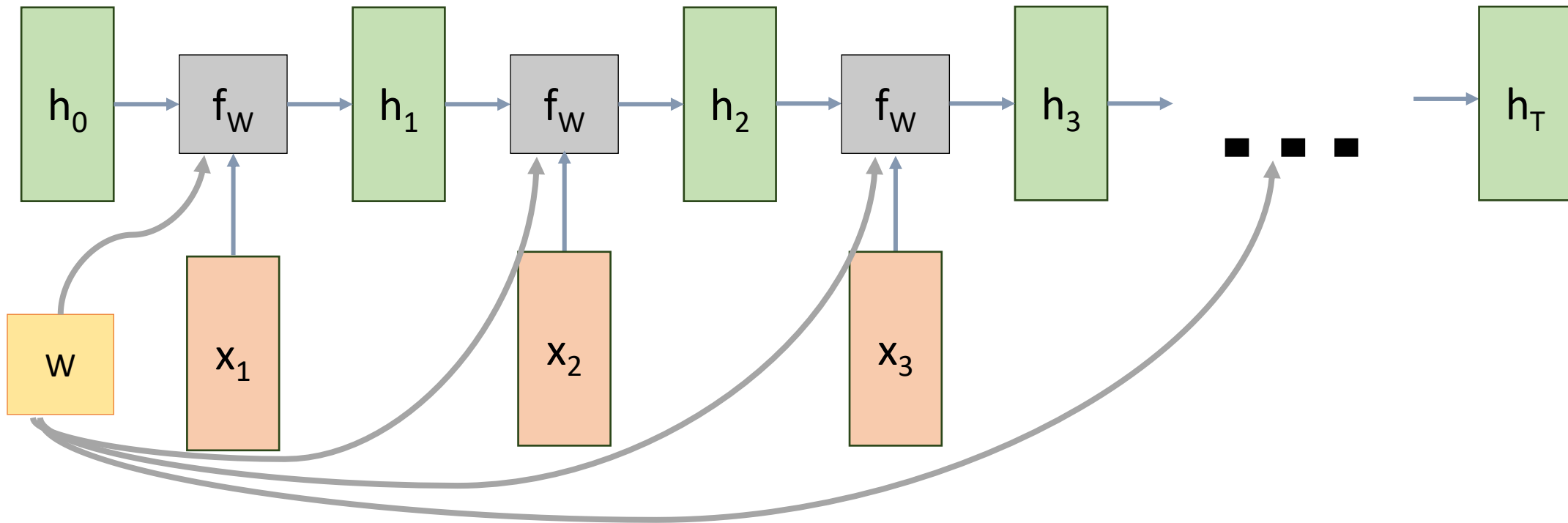


RNN: Computational Graph

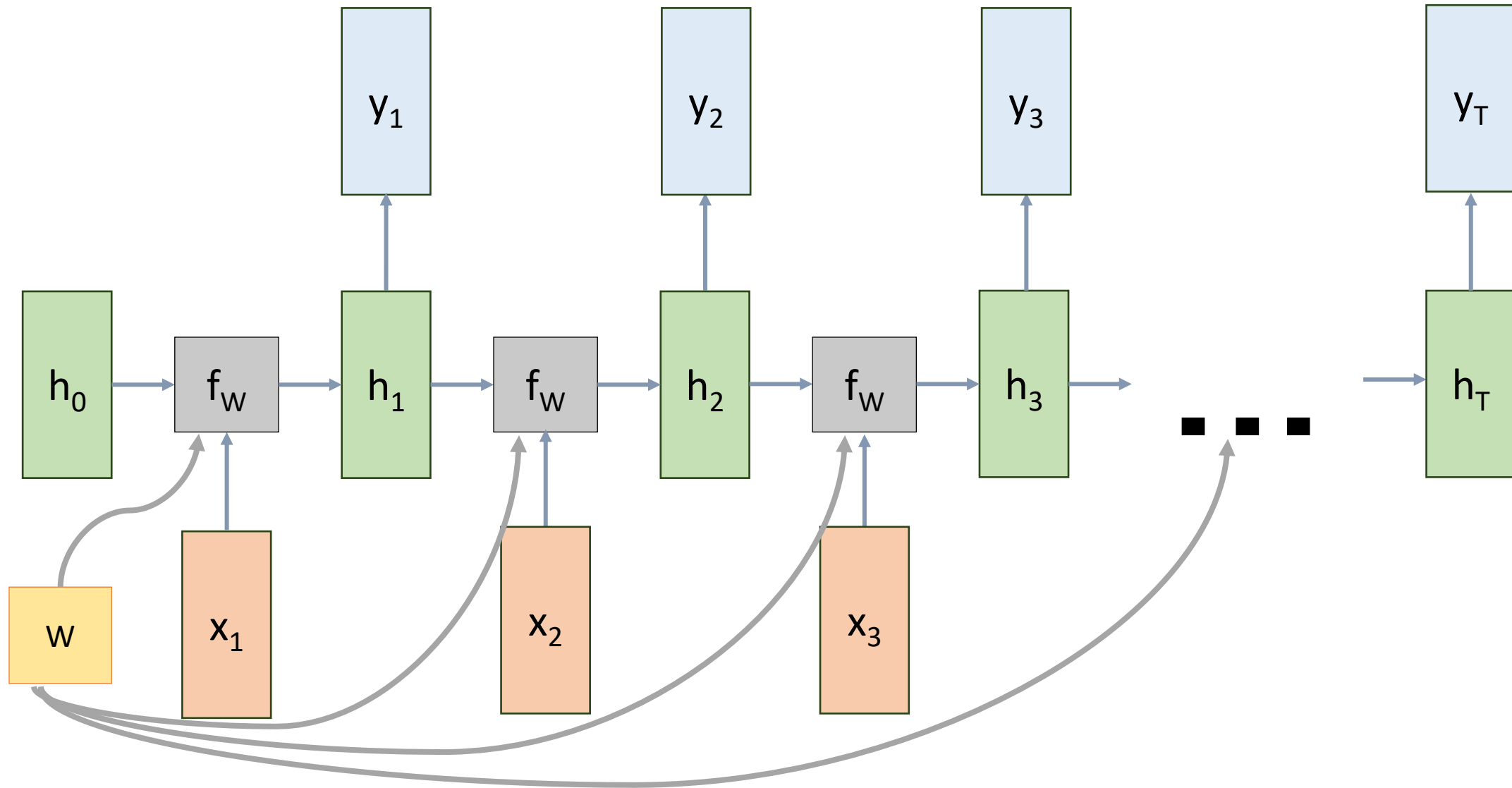


RNN: Computational Graph

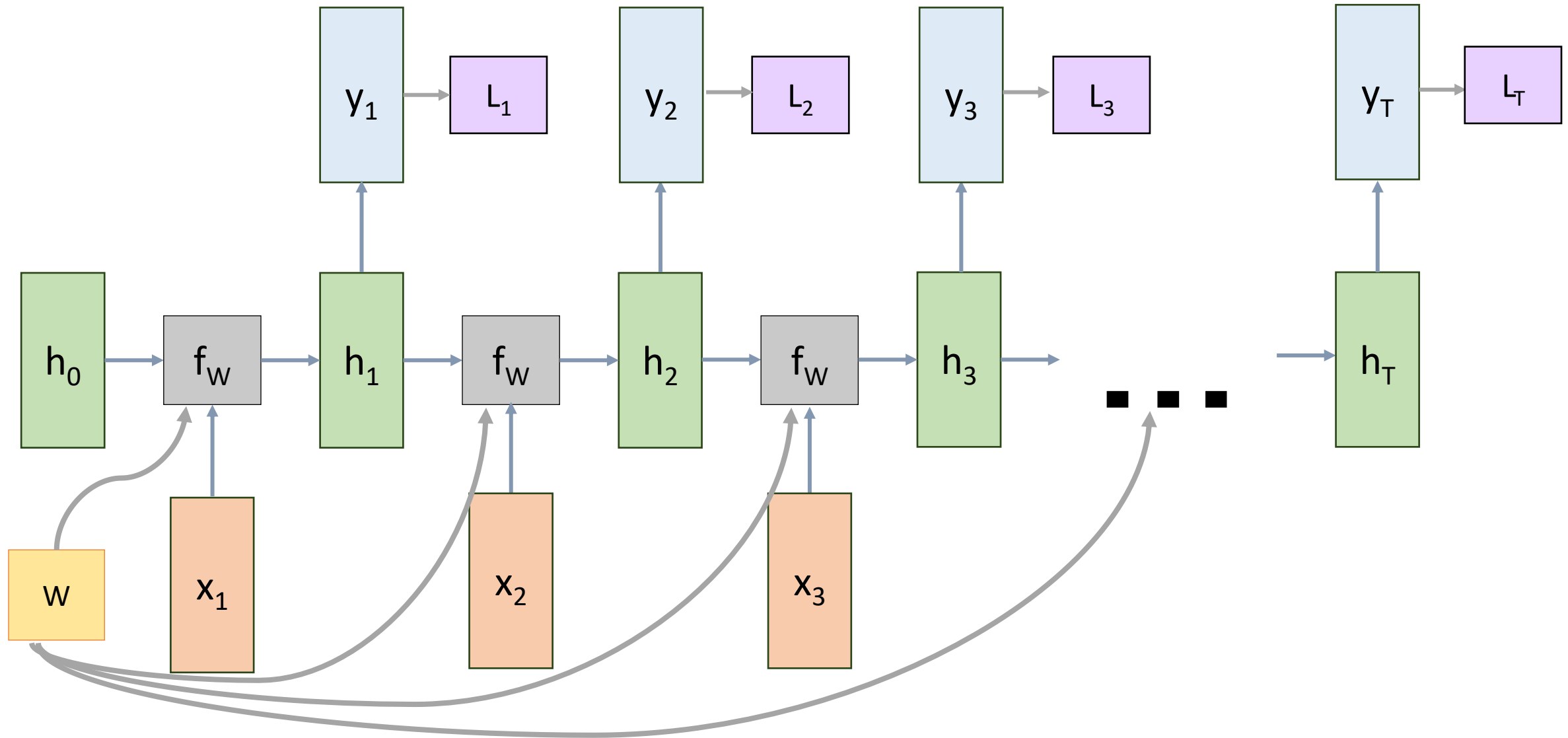
Re-use the same weight matrix at every time-step

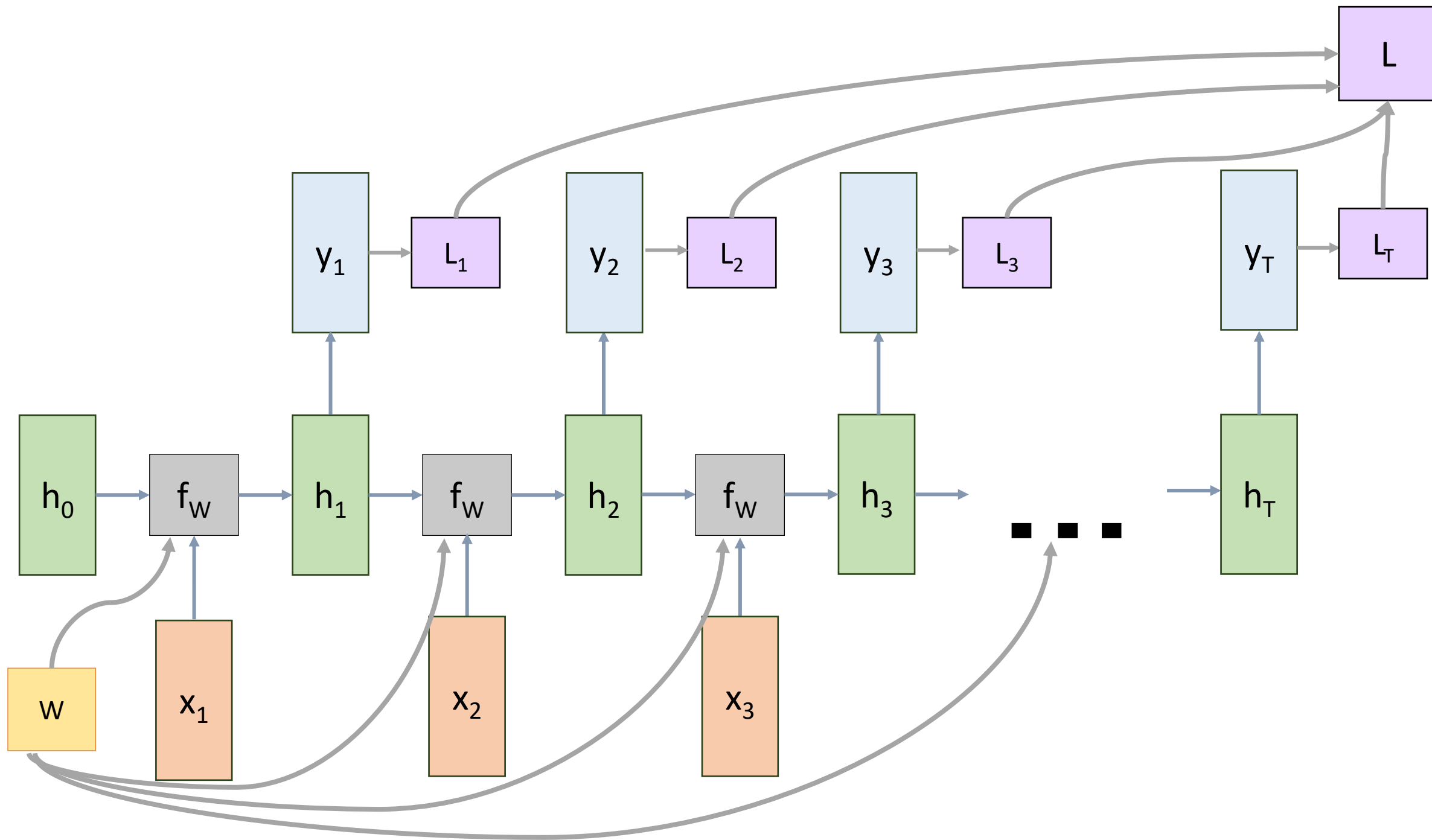


RNN: Computational Graph: Many to Many

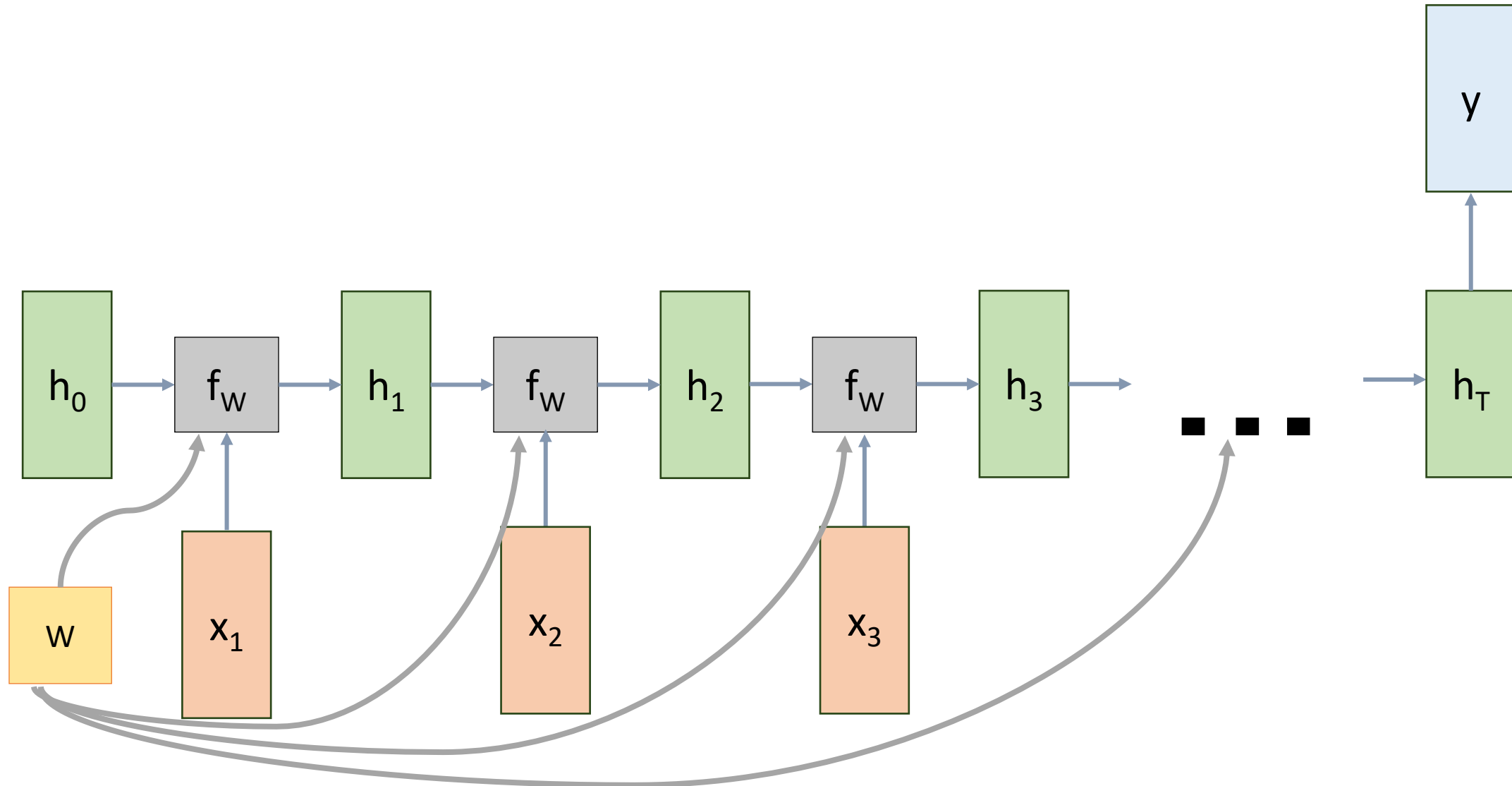


RNN: Computational Graph: Many to Many

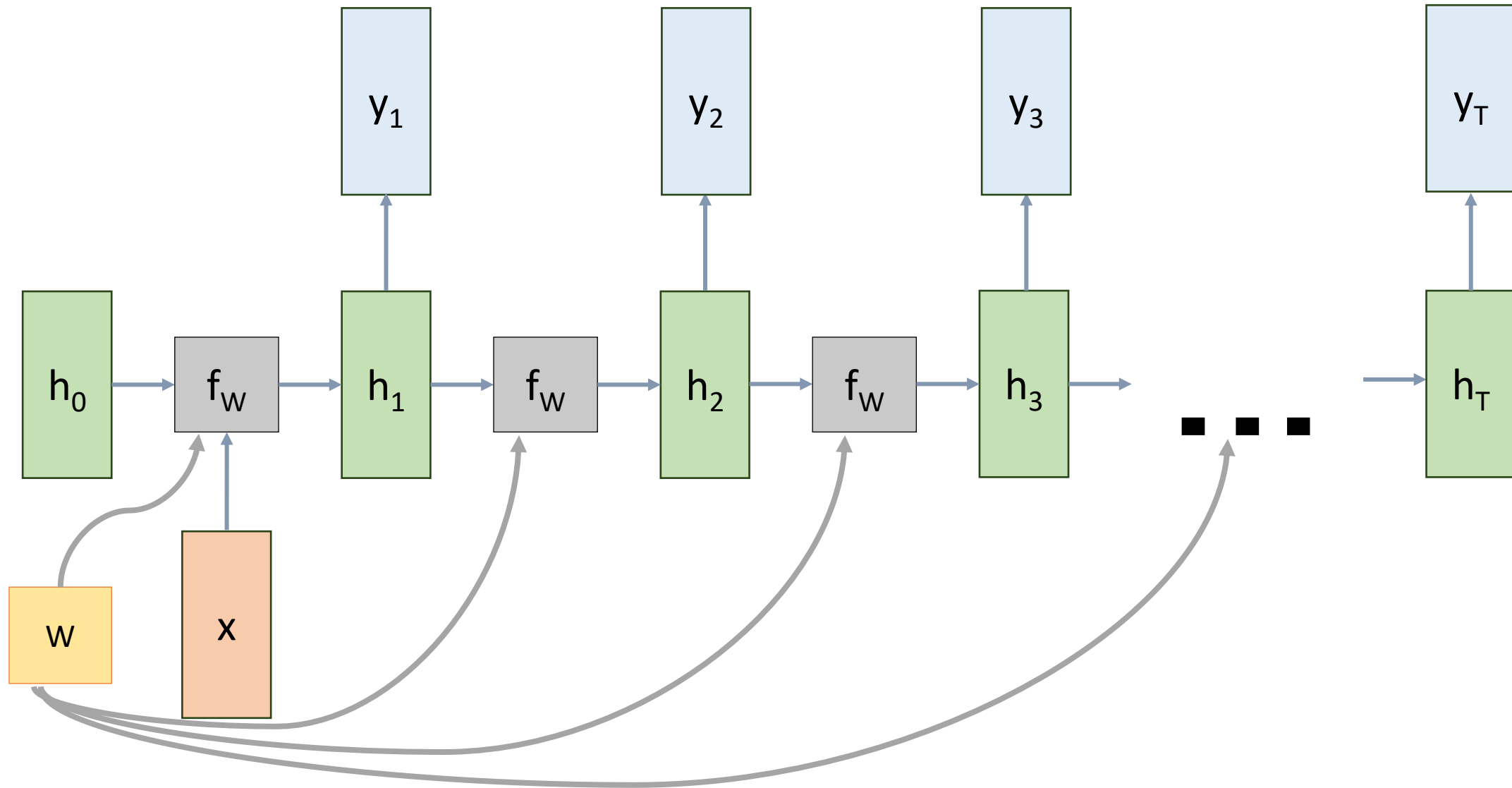




RNN: Computational Graph: Many to One

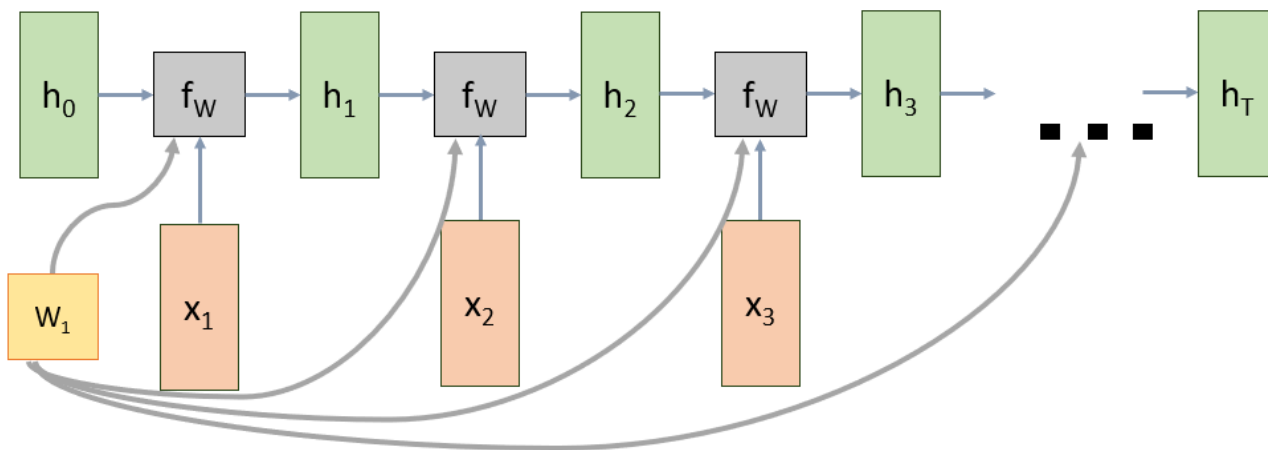


RNN: Computational Graph: One to Many



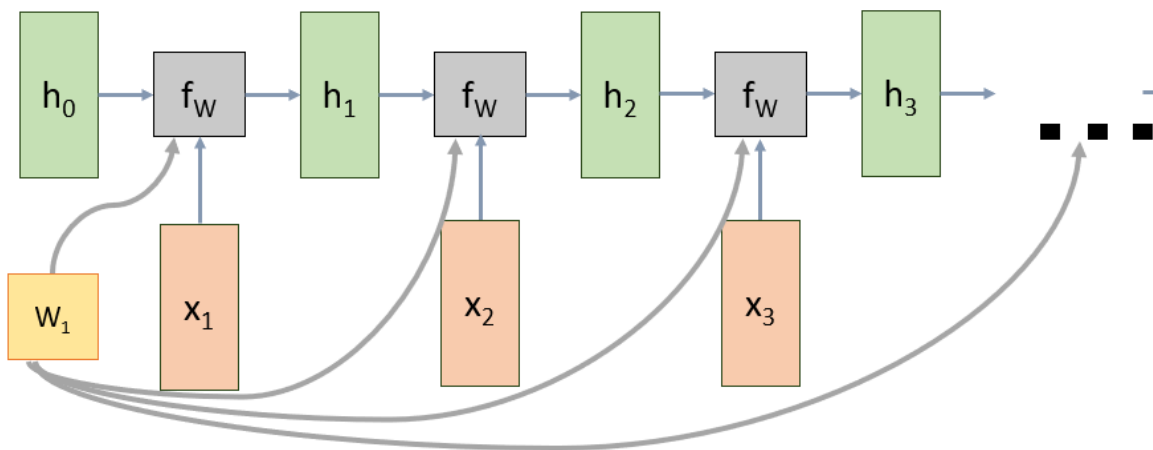
Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

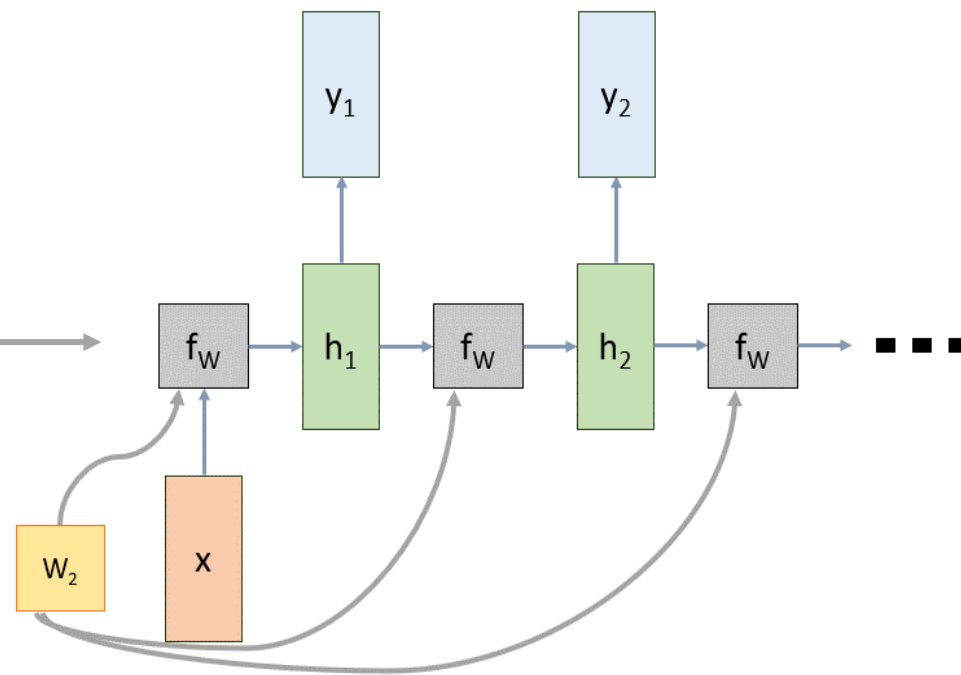


Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

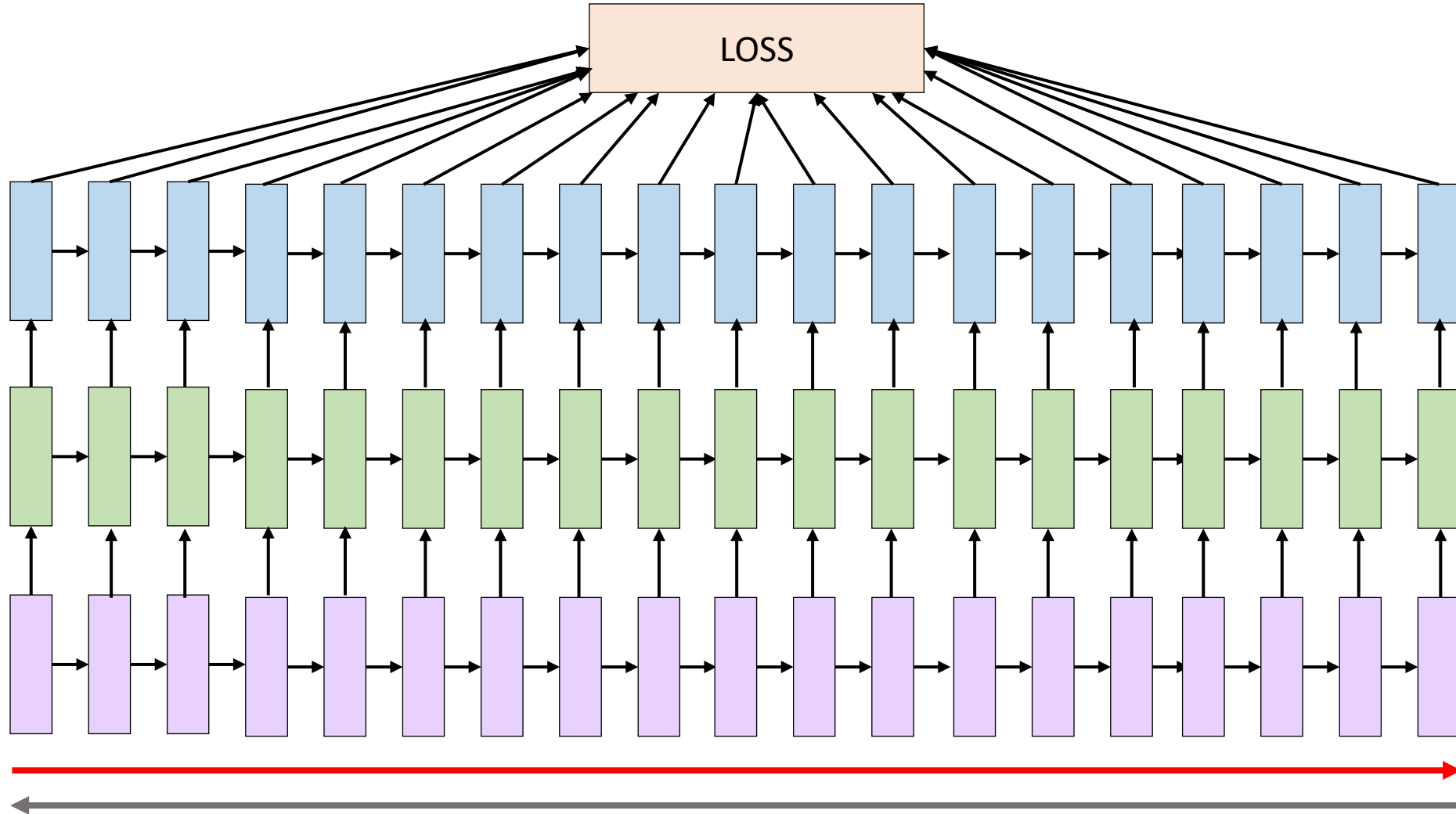


One to many: Produce output sequence from single input vector

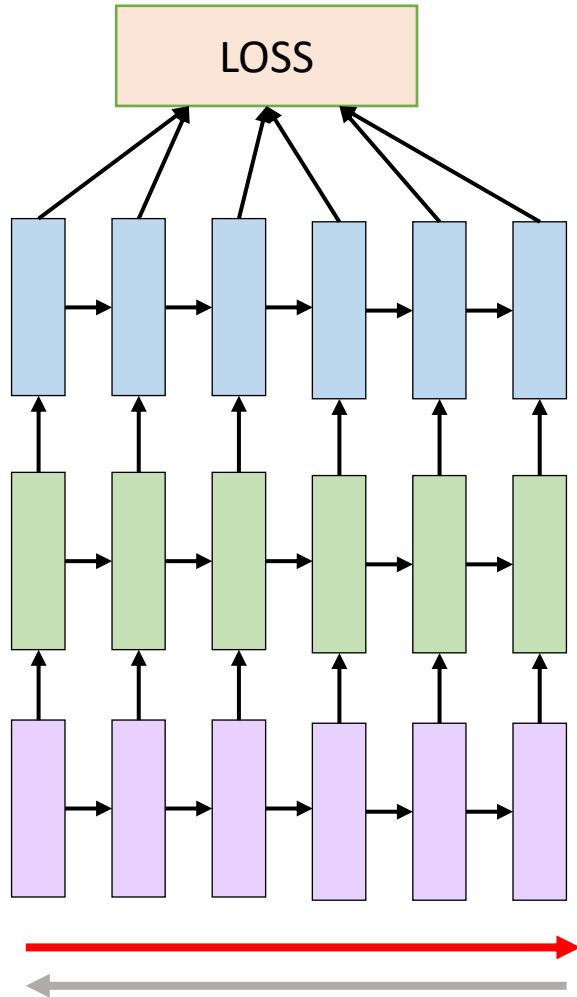


Backpropagation through time

Forward through an entire sequence to compute loss, then backward through entire sequence to compute the gradient.

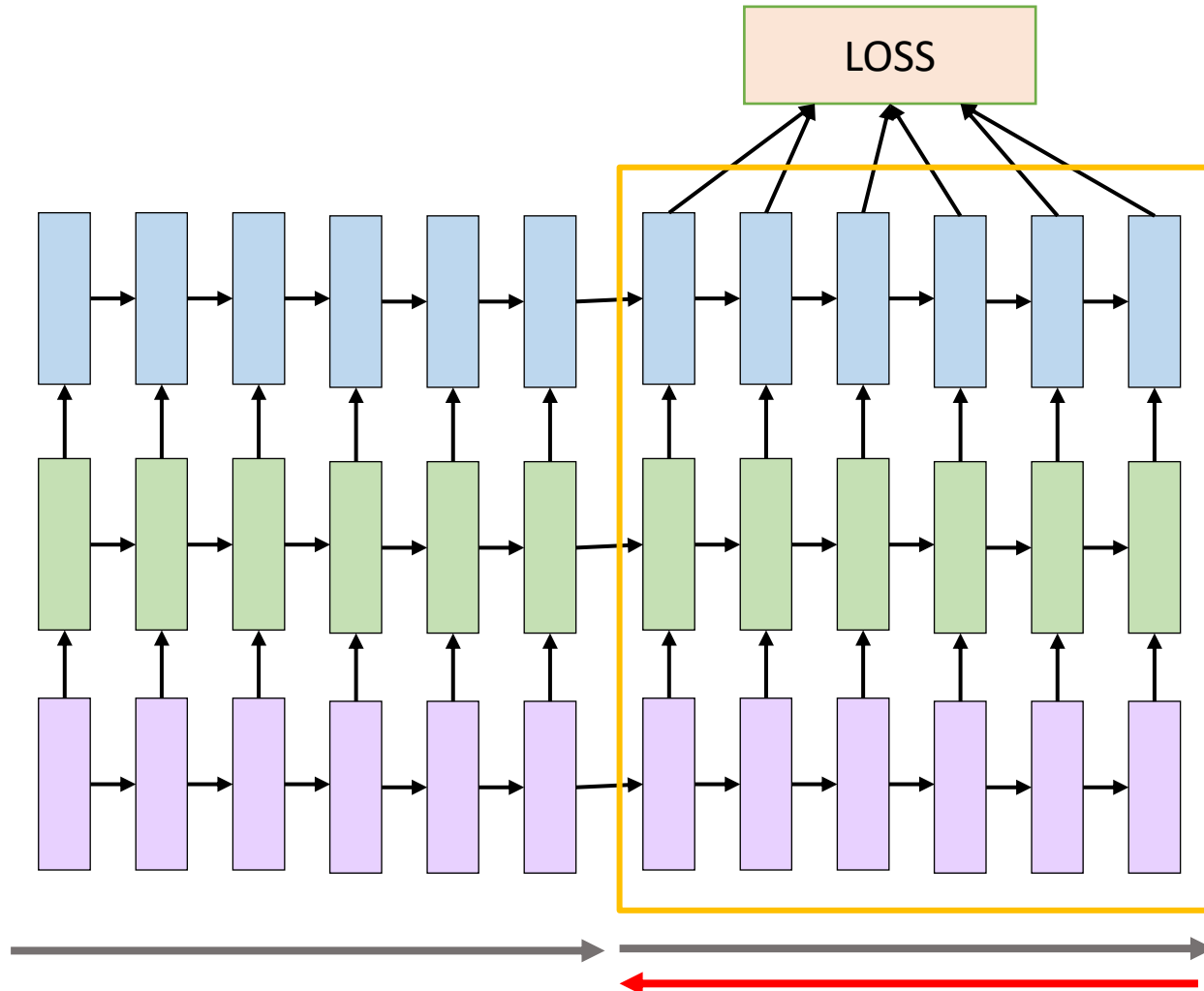


Truncated Backpropagation through time



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

Truncated Backpropagation through time



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

Truncated Backpropagation through time

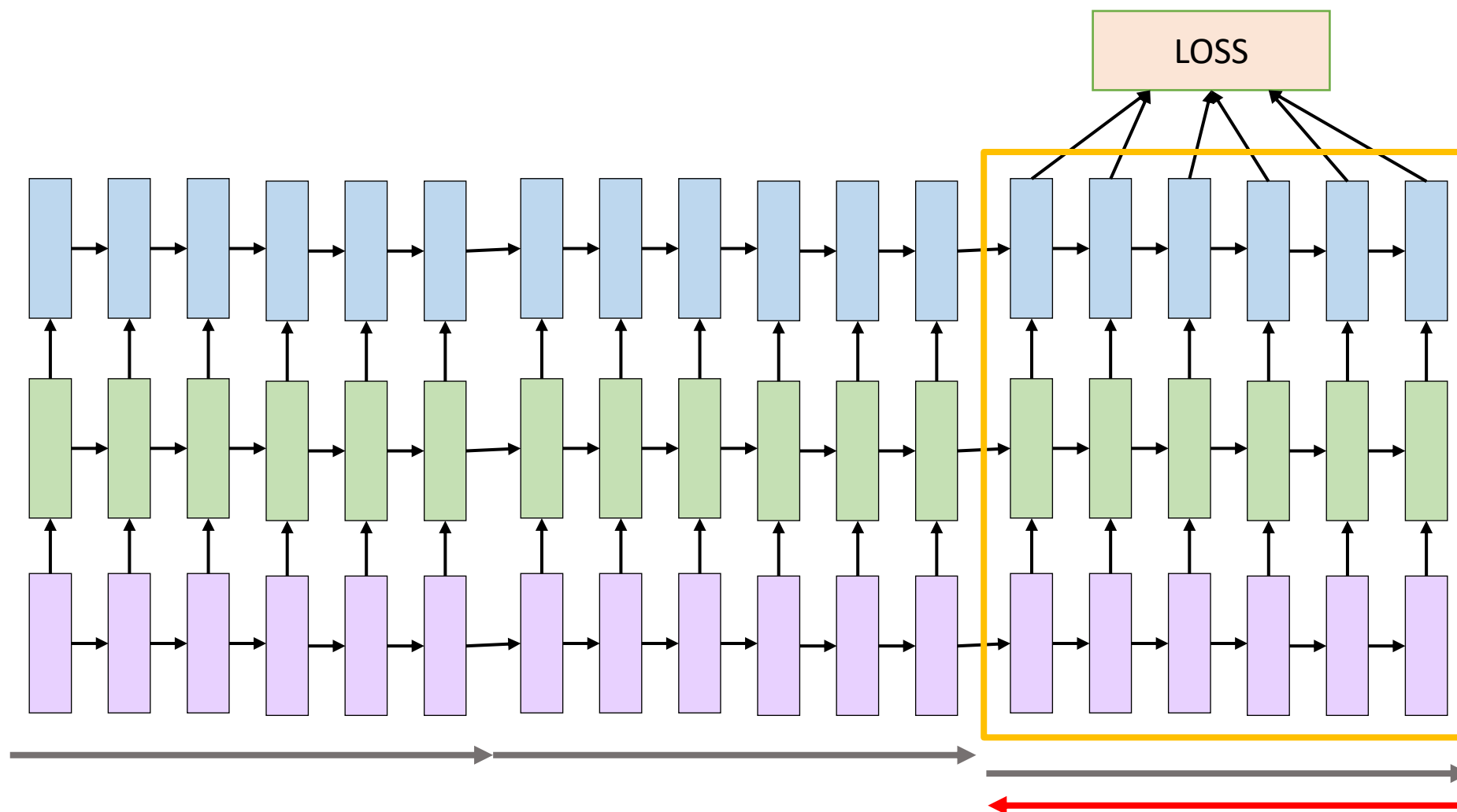
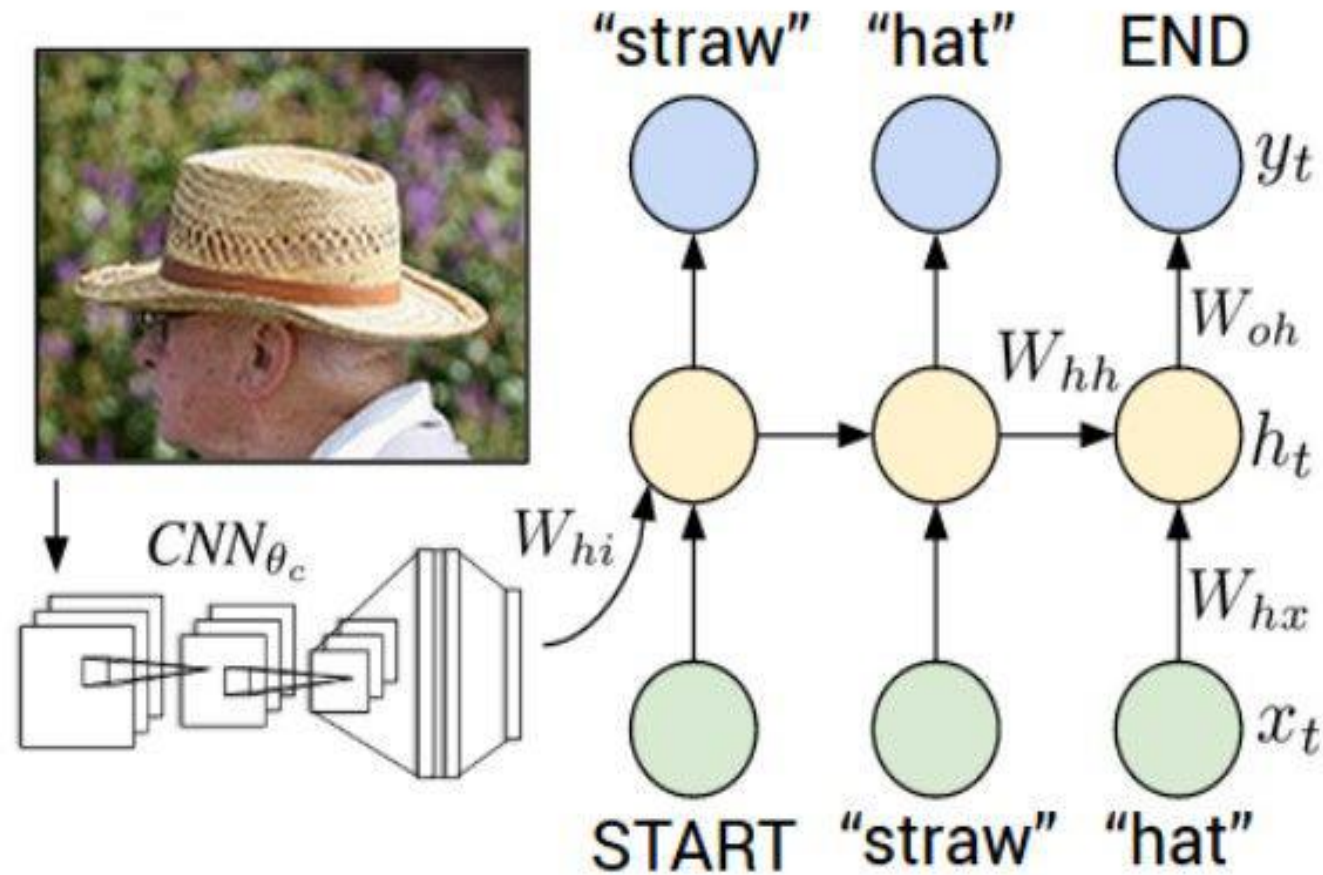
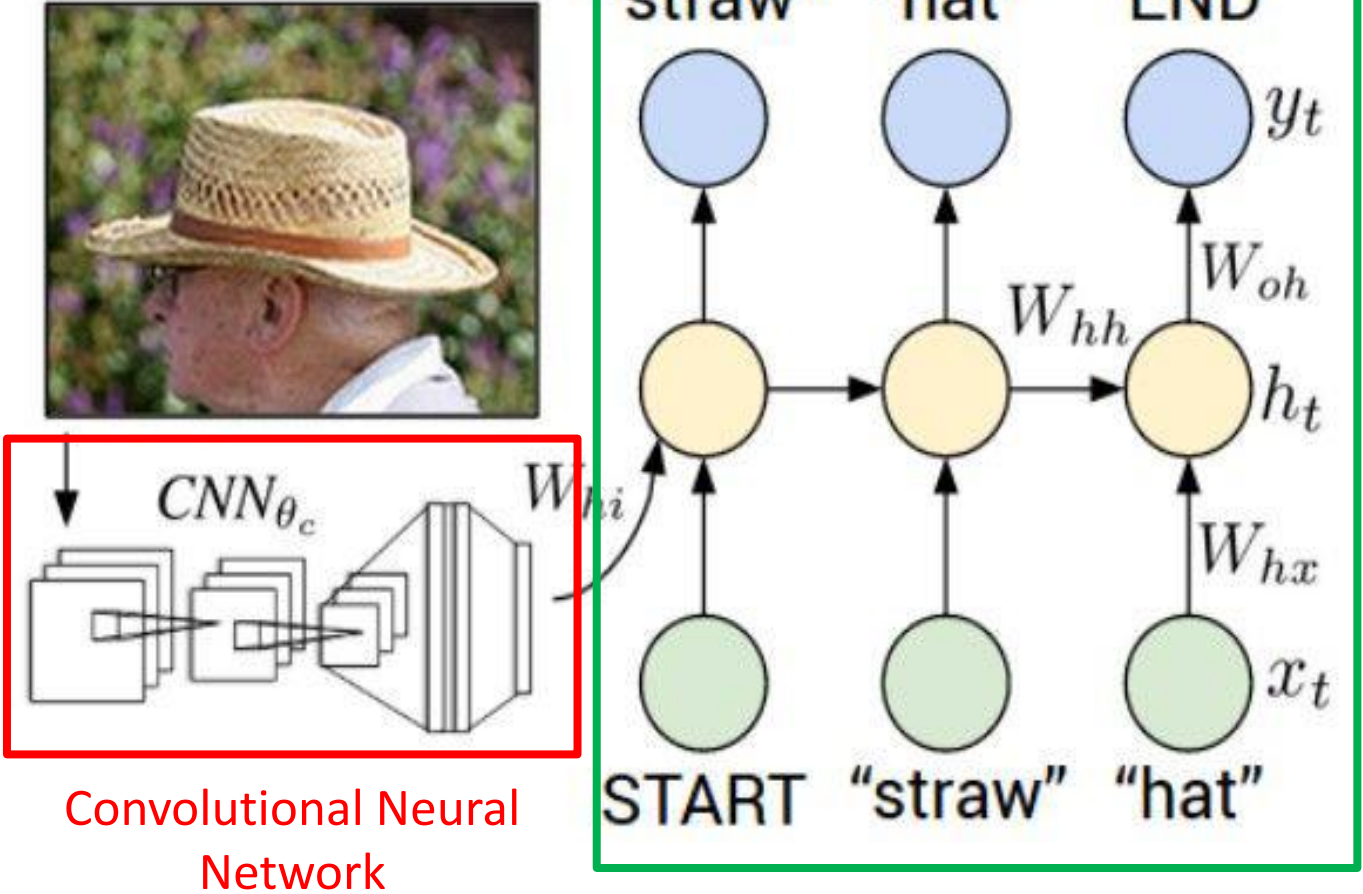


Image Captioning



Recurrent Neural Network



Convolutional Neural Network

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

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

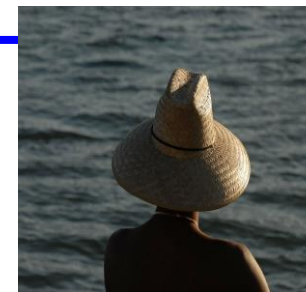
conv-512

conv-512

maxpool

FC-4096

FC-4096



This image is CC0 public domain

test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

v

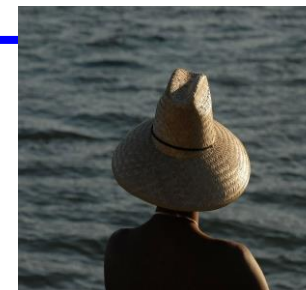
W_{ih}

y_0

h_0

x_0

<START>



This image is CC0 public domain

test image

before:

$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

now:

$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$$

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

v

<START>

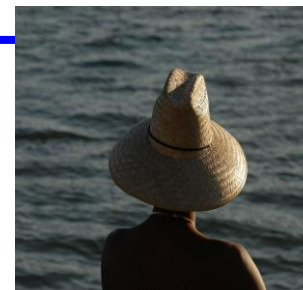
y_0

h_0

x_0

straw

sample!



This image is CC0 public domain

test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

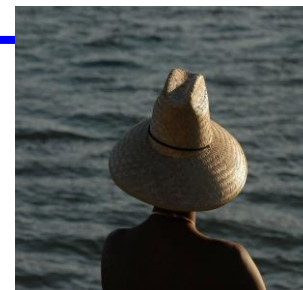
conv-512

maxpool

FC-4096

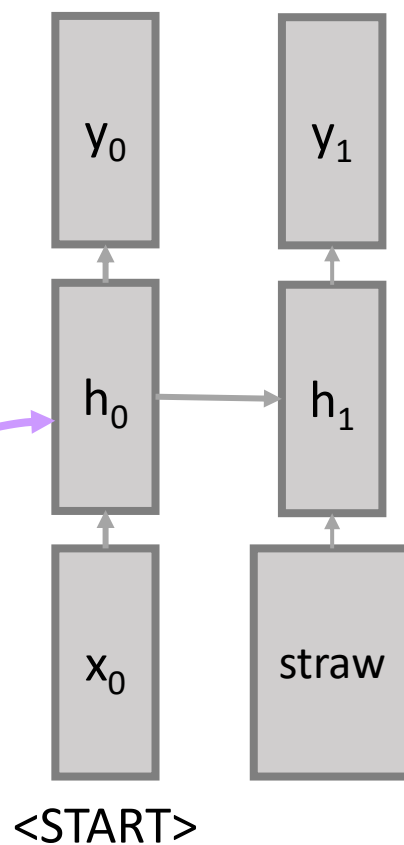
FC-4096

v



test image

This image is CC0 public domain



image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

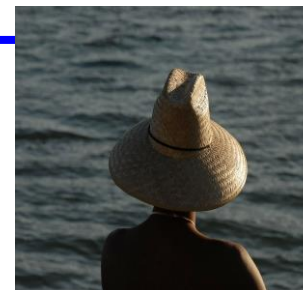
conv-512

maxpool

FC-4096

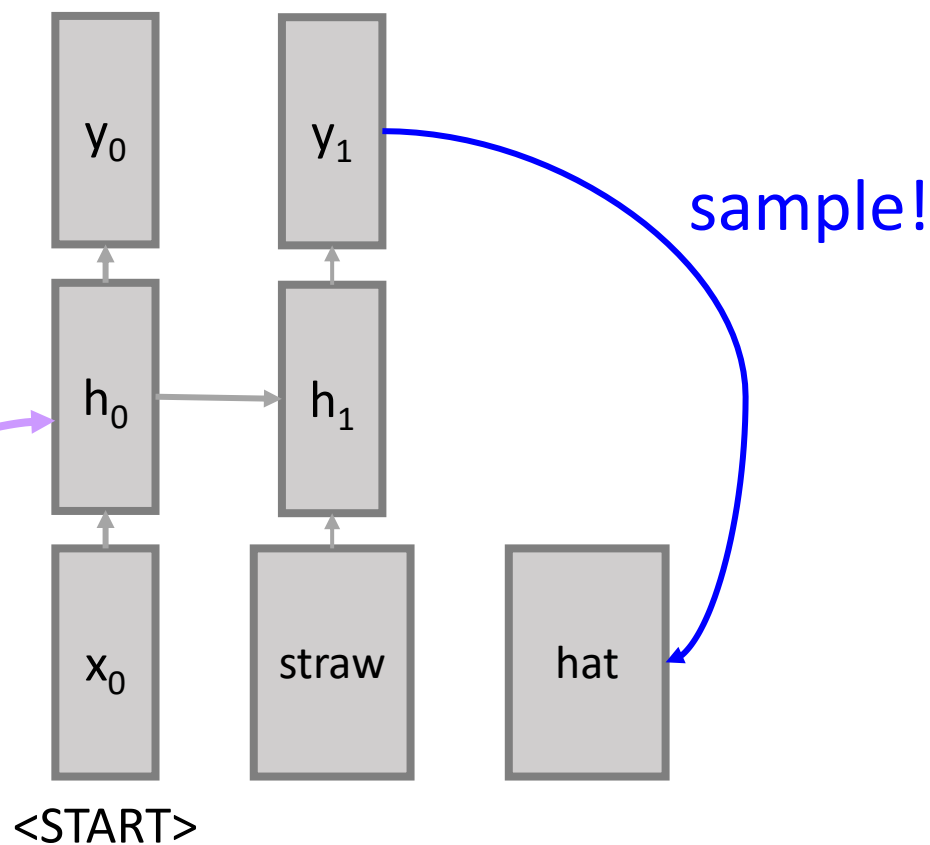
FC-4096

v



test image

This image is CC0 public domain



image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

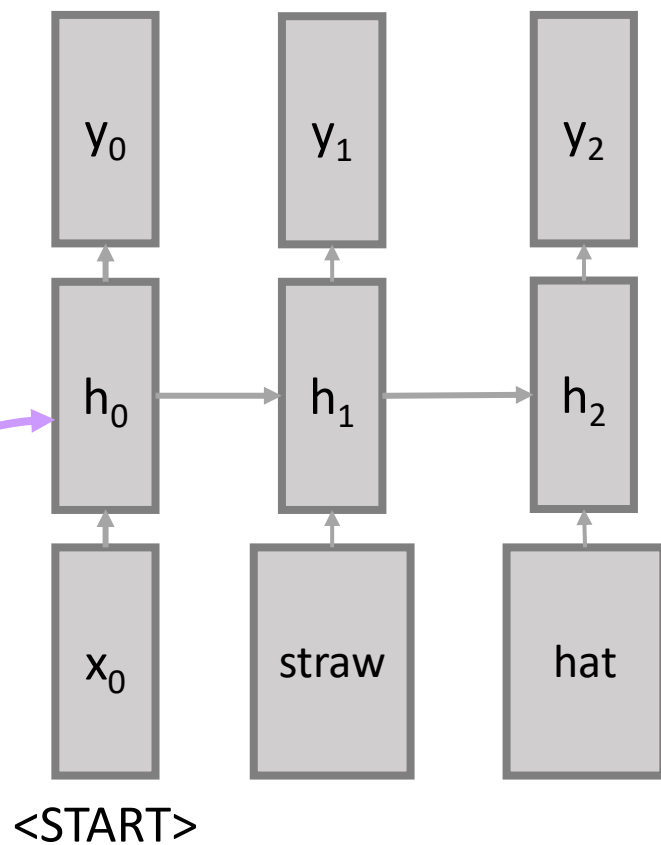
FC-4096
FC-4096

v



test image

This image is CC0 public domain



image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

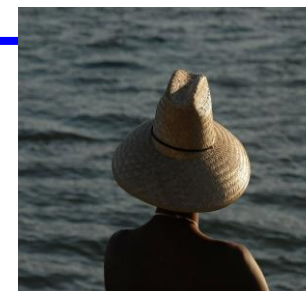
conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096

v



test image

This image is CC0 public domain

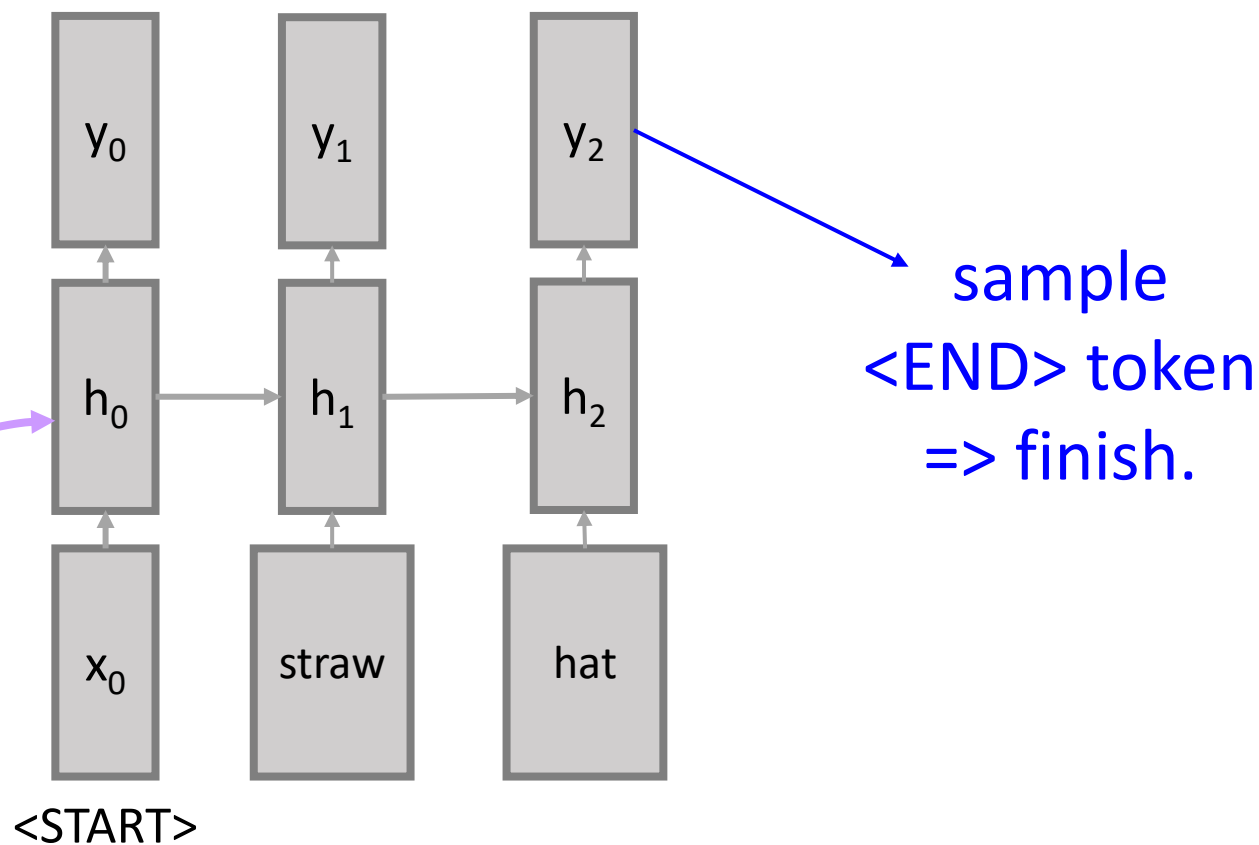


Image Captioning: Example Results



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

Image Captioning with Attention

RNN focuses its attention on a different spatial location when generating each word

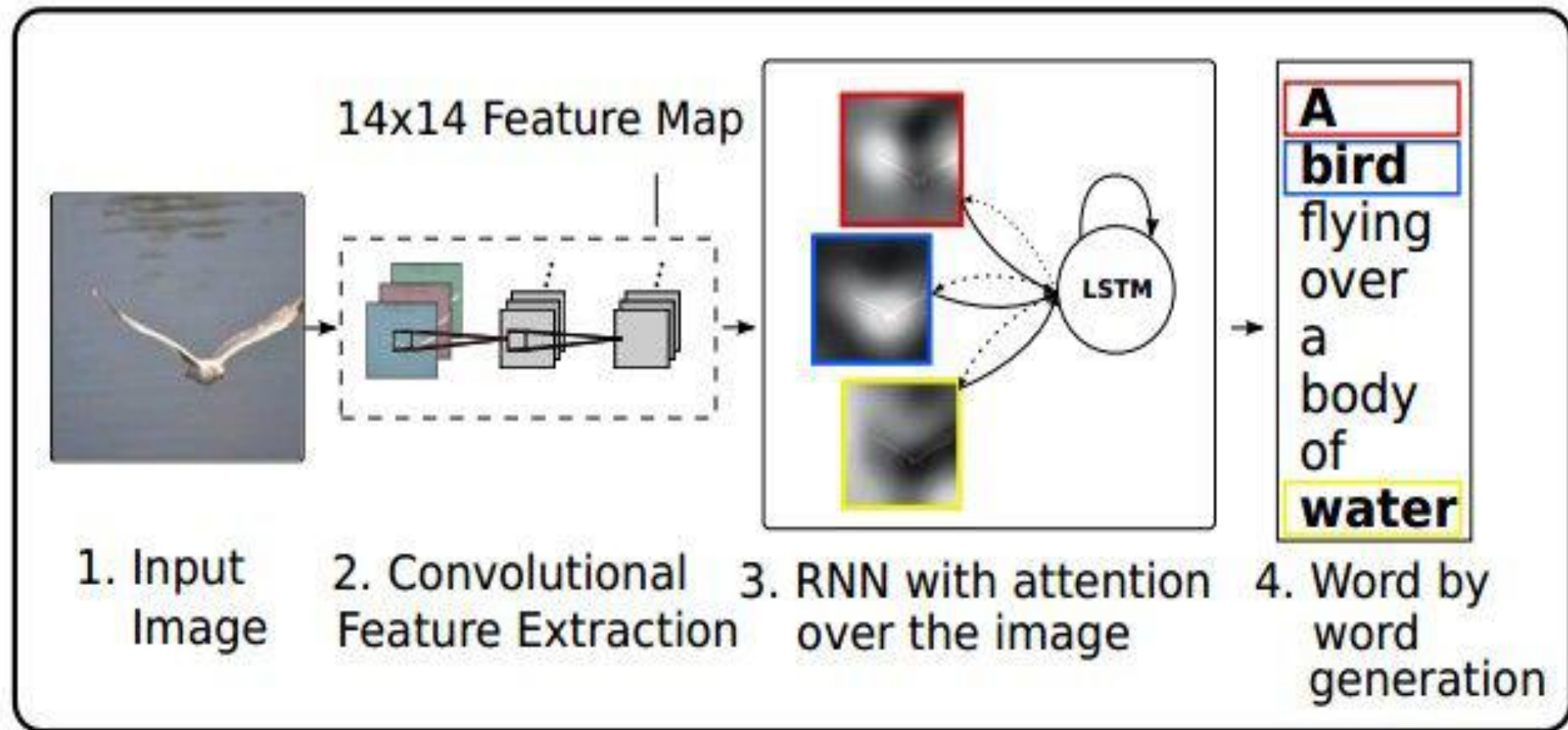


Image Captioning with Attention

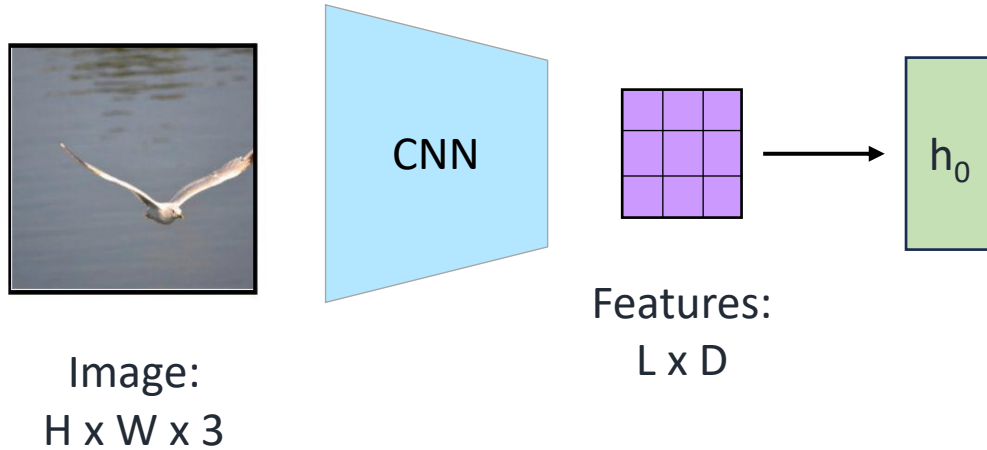


Image Captioning with Attention

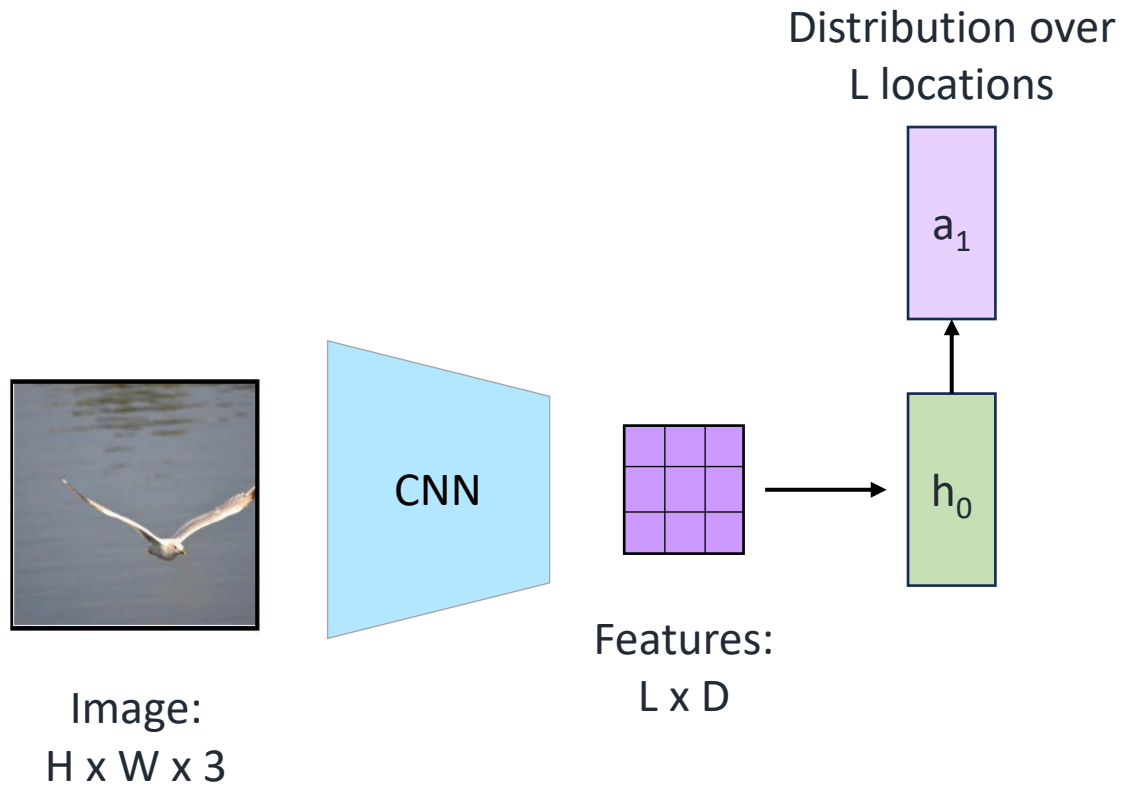


Image Captioning with Attention

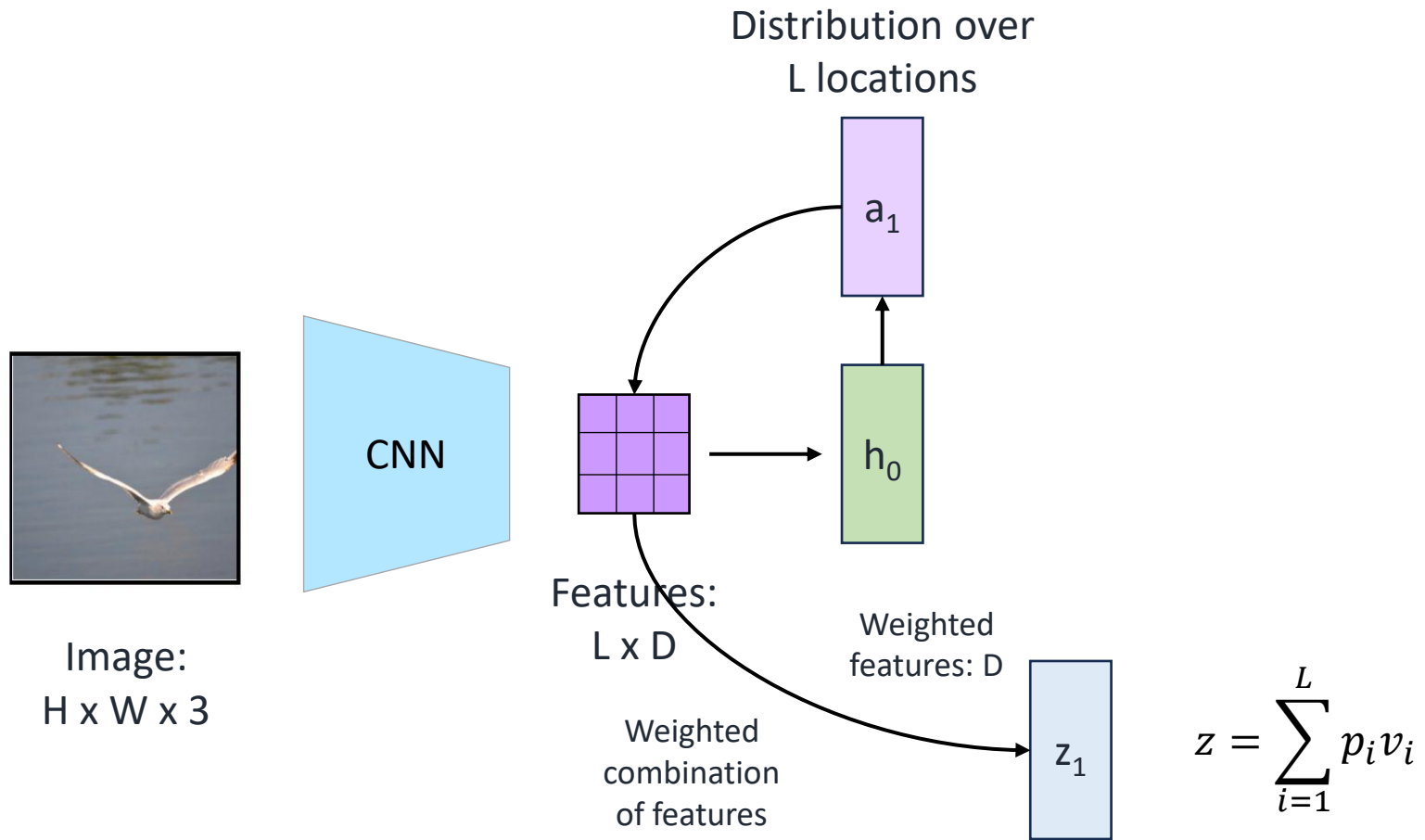


Image Captioning with Attention

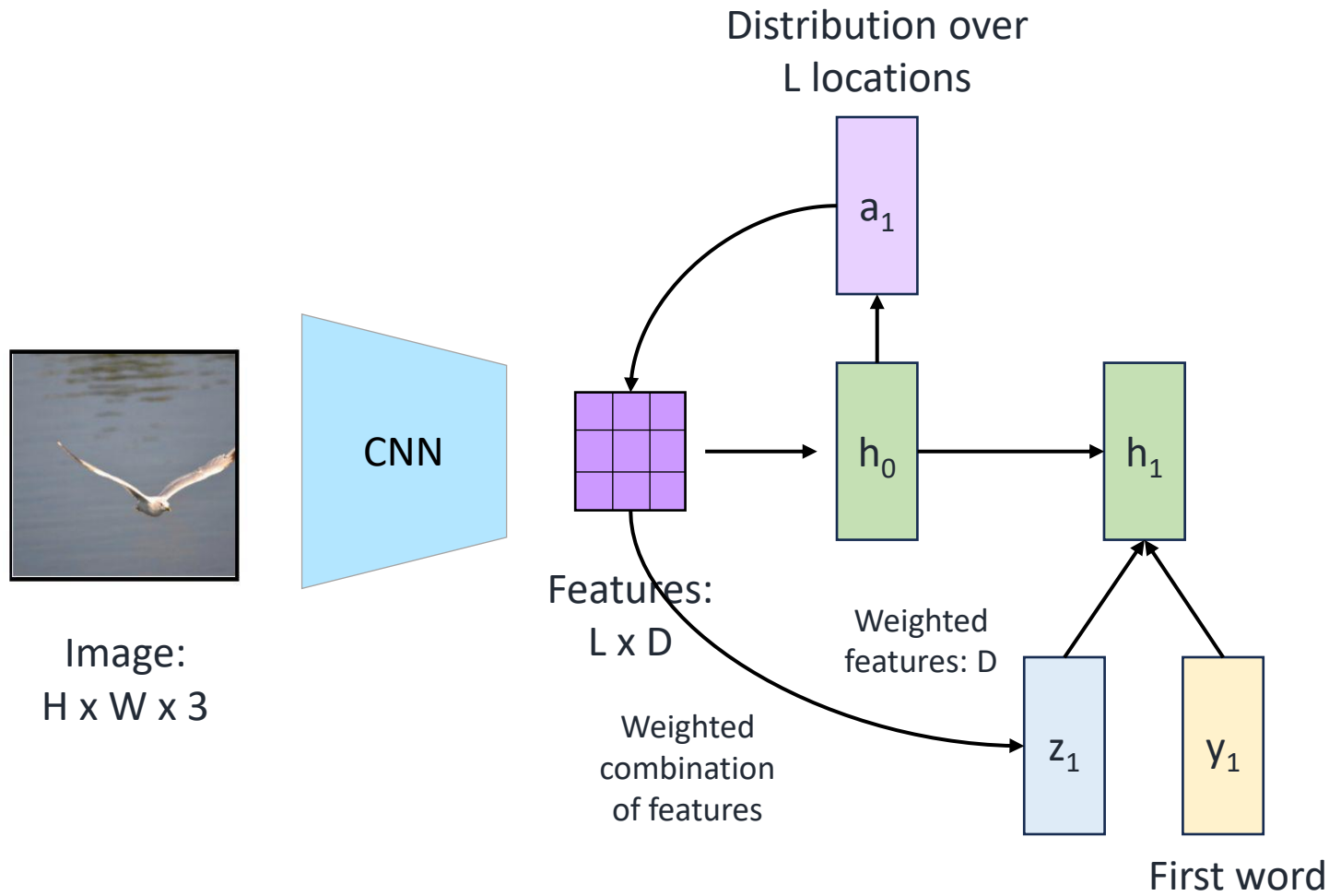


Image Captioning with Attention

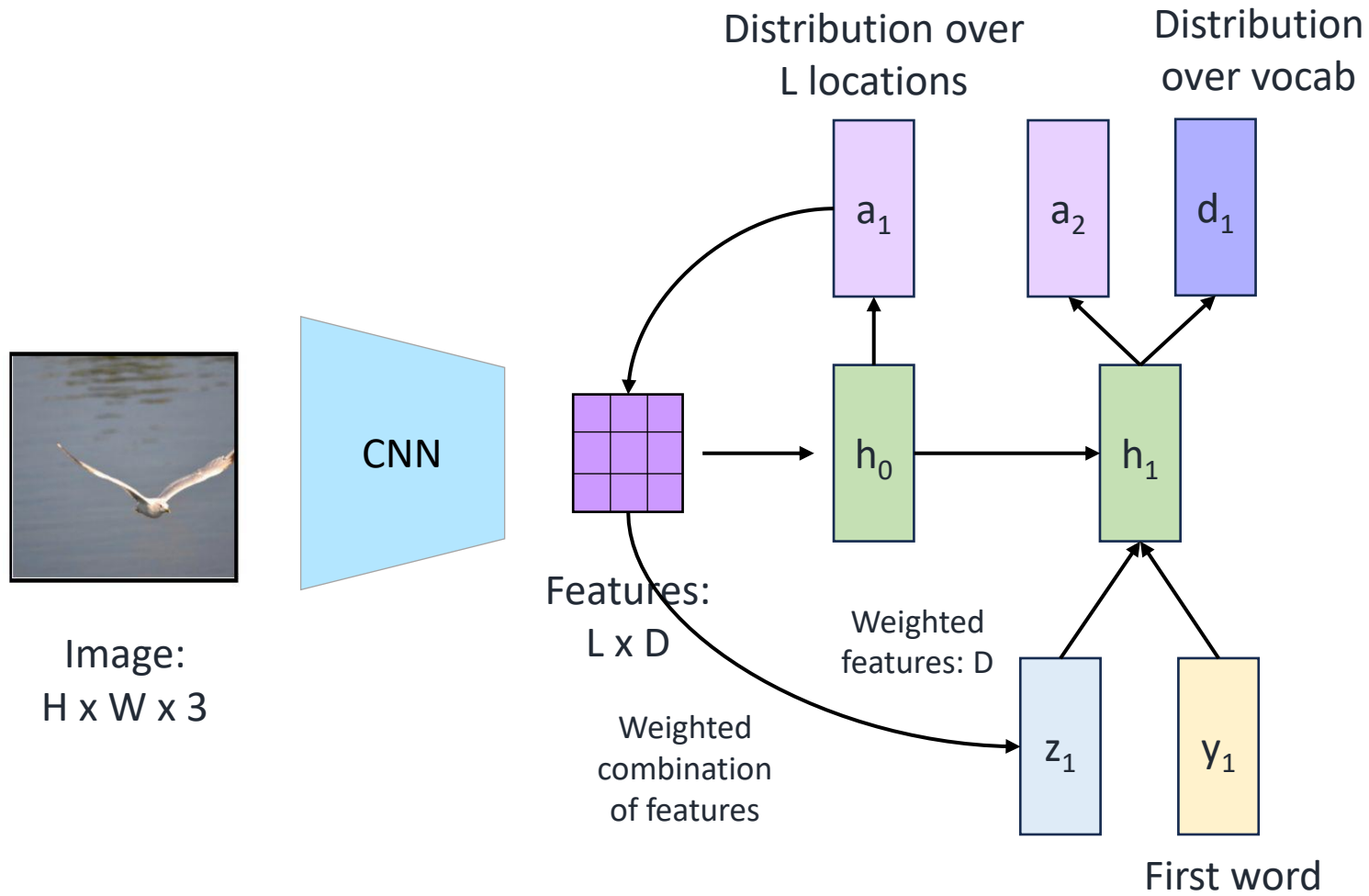


Image Captioning with Attention

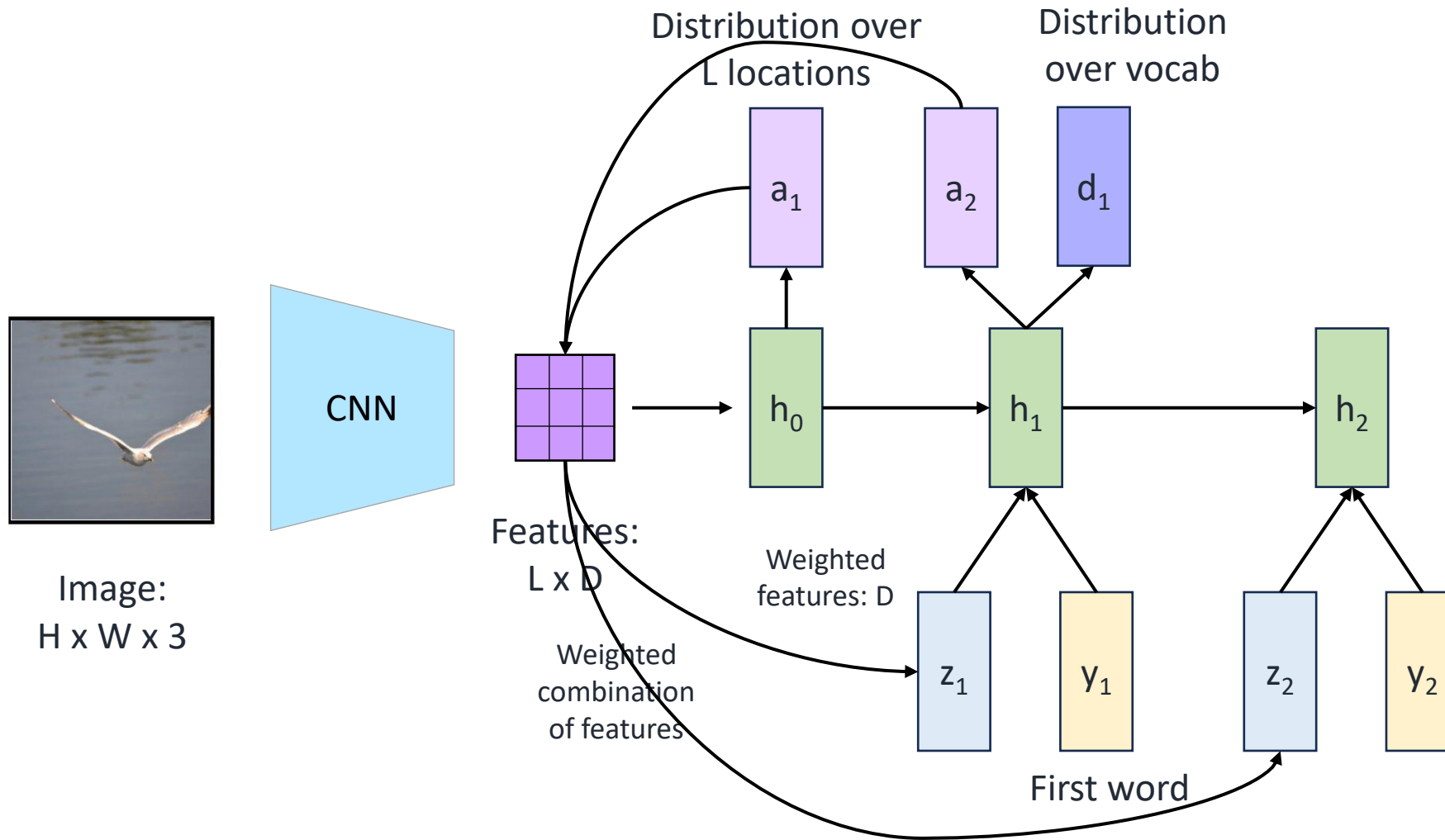


Image Captioning with Attention

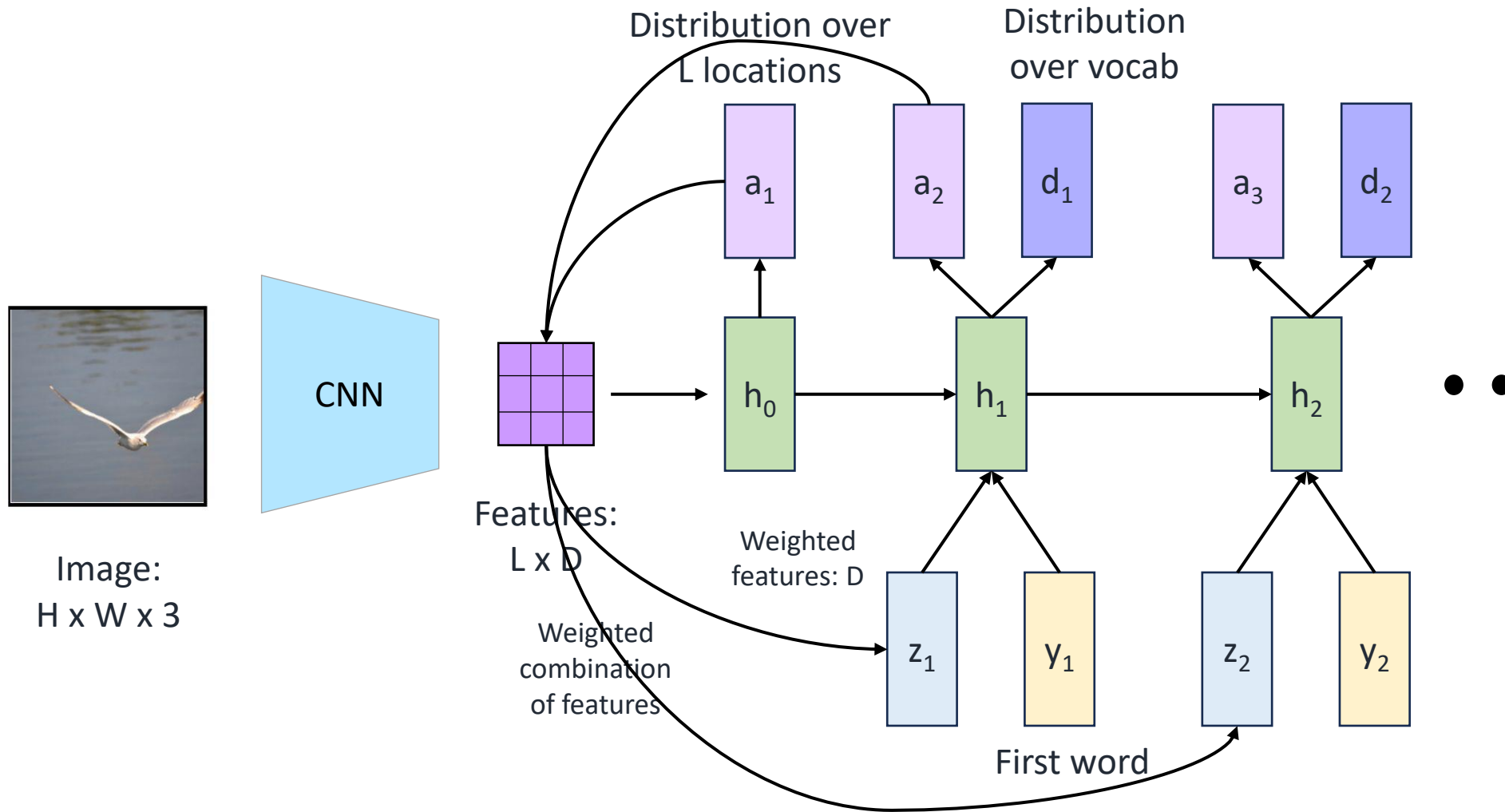


Image Captioning with Attention

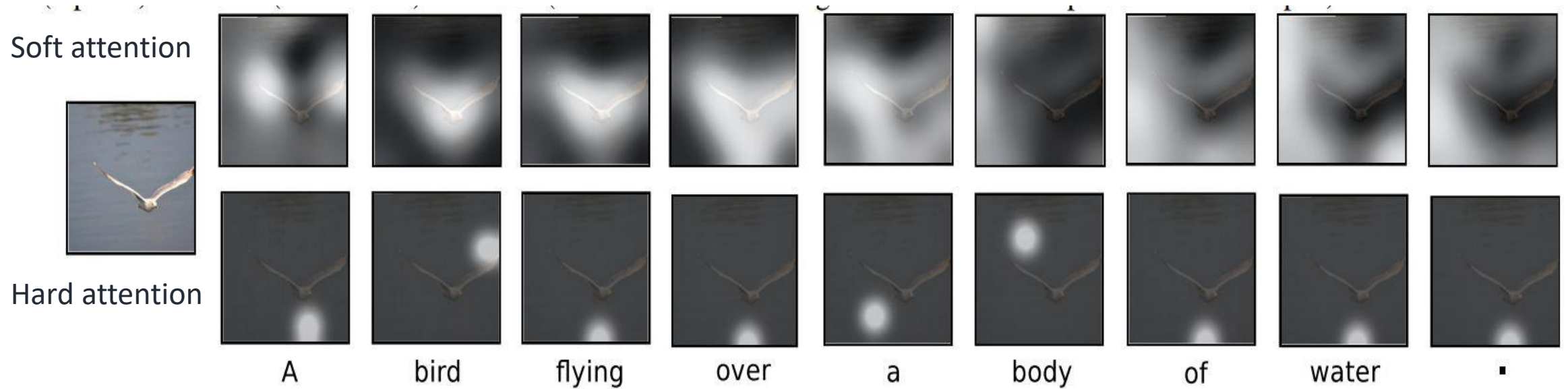


Image Captioning with Attention



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Multilayer RNNs

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$h \in \mathbb{R}^n$

$W^l [n \times 2n]$

LSTM:

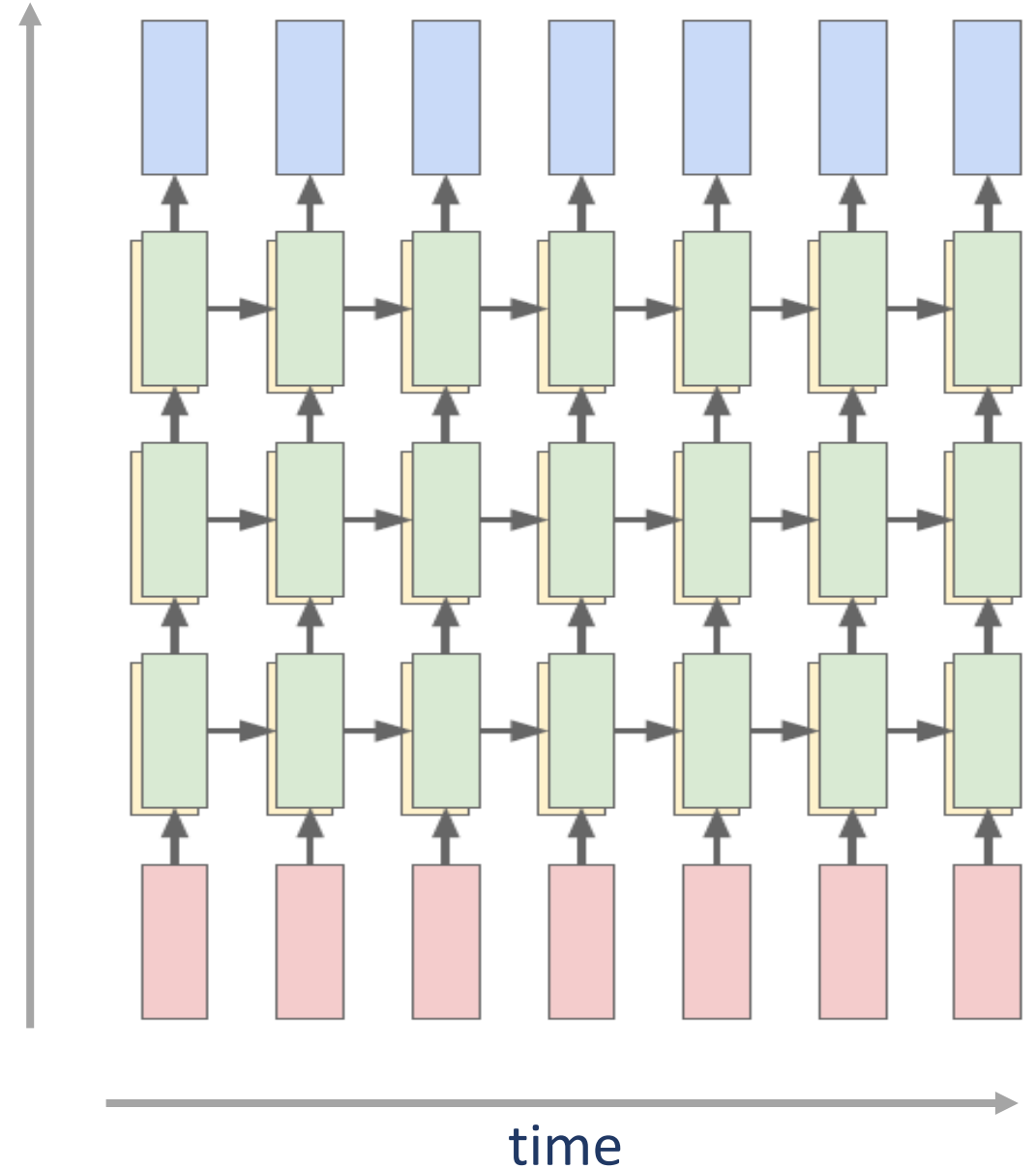
$W^l [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

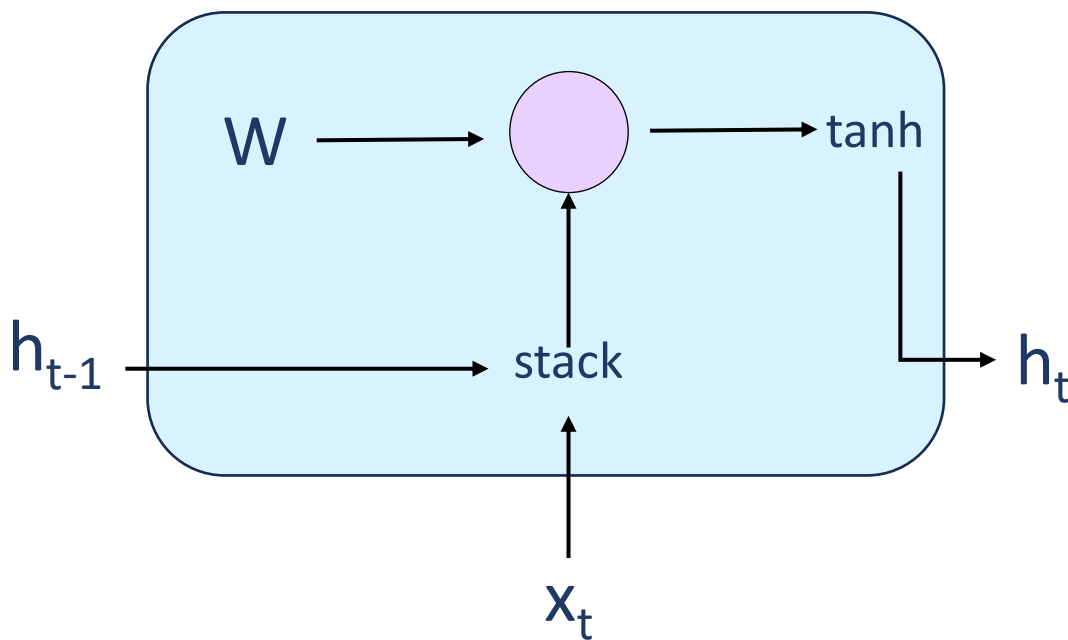
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

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

depth



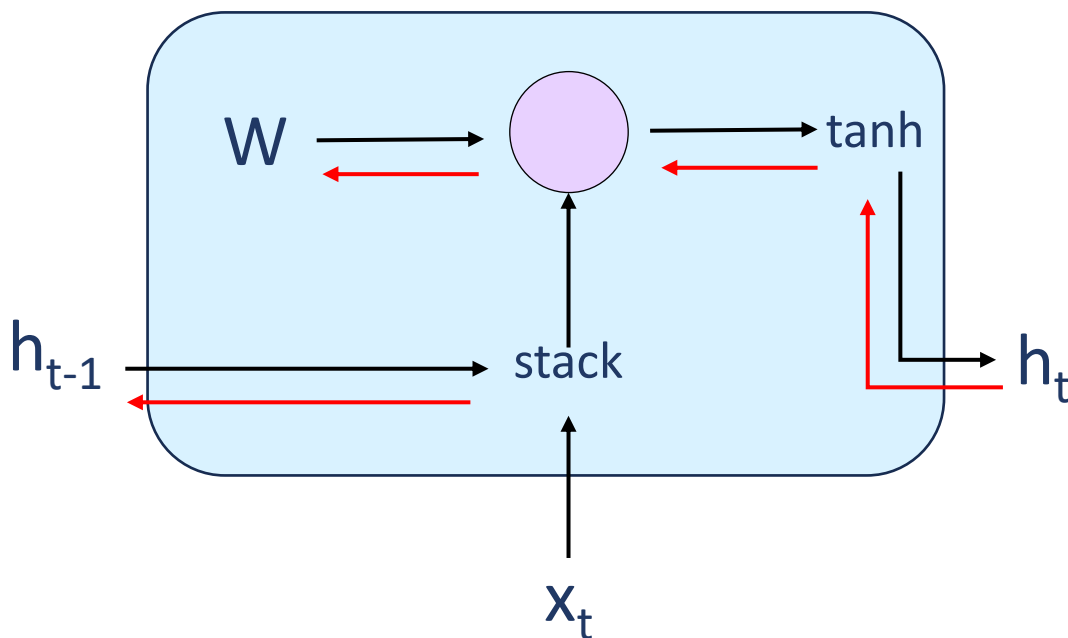
Vanilla RNN gradient flow



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

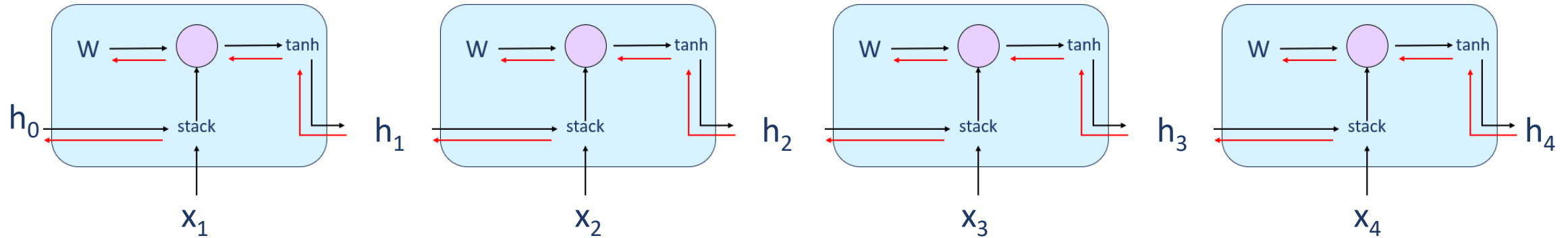
Vanilla RNN gradient flow

Backpropagation from h_t
to h_{t-1} multiplies by
 W (actually W_{hh}^T)



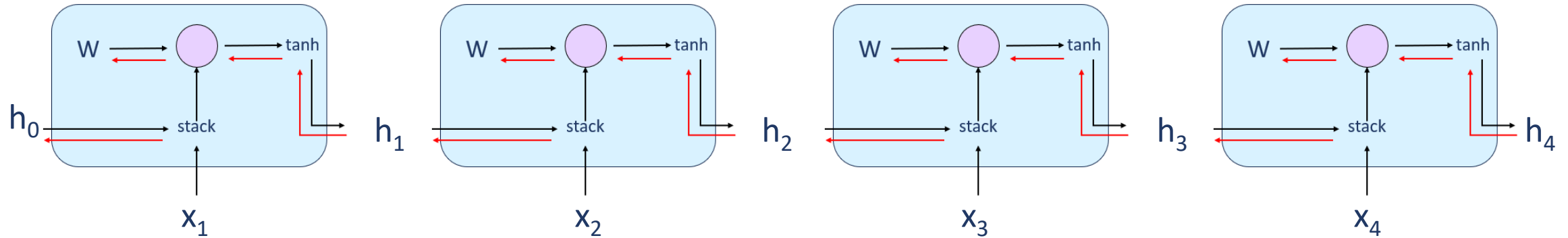
$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \\ &= \tanh\left((W_{hh} \quad W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\ &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \end{aligned}$$

Vanilla RNN gradient flow



Computing gradient of h_0 involves many factors of W (and repeated \tanh)

Vanilla RNN gradient flow

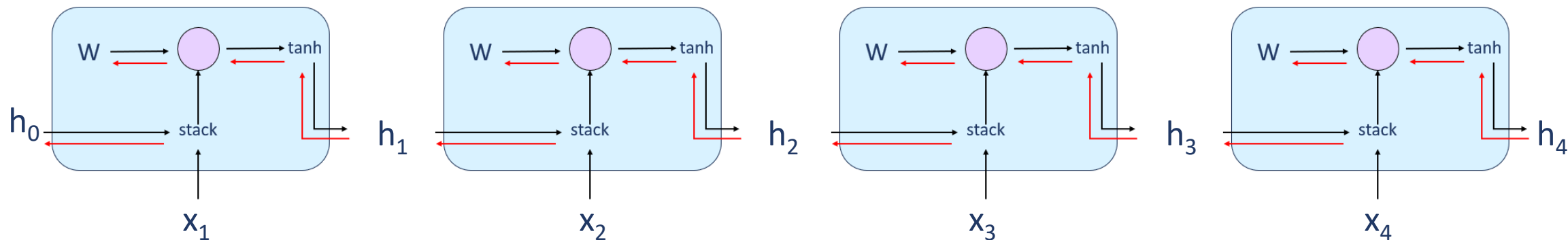


Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1 : **Exploding gradients**

Largest singular value < 1 : **Vanishing gradients**

Vanilla RNN gradient flow



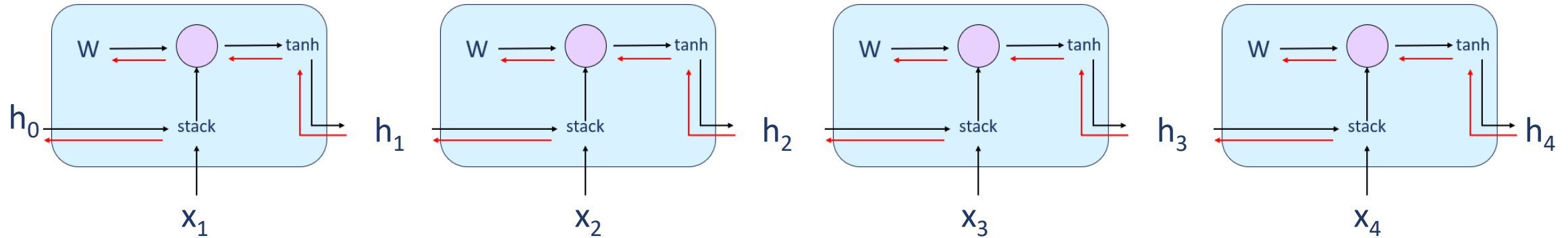
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1 : **Exploding gradients**

Gradient clipping: Scale gradient if its norm is too big

Largest singular value < 1 : **Vanishing gradients**

Vanilla RNN gradient flow



Computing gradient of h_0 involves many factors of W (and repeated \tanh)

Largest singular value > 1 : **Exploding gradients**

Largest singular value < 1 : **Vanishing gradients**



Change RNN architecture

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM:

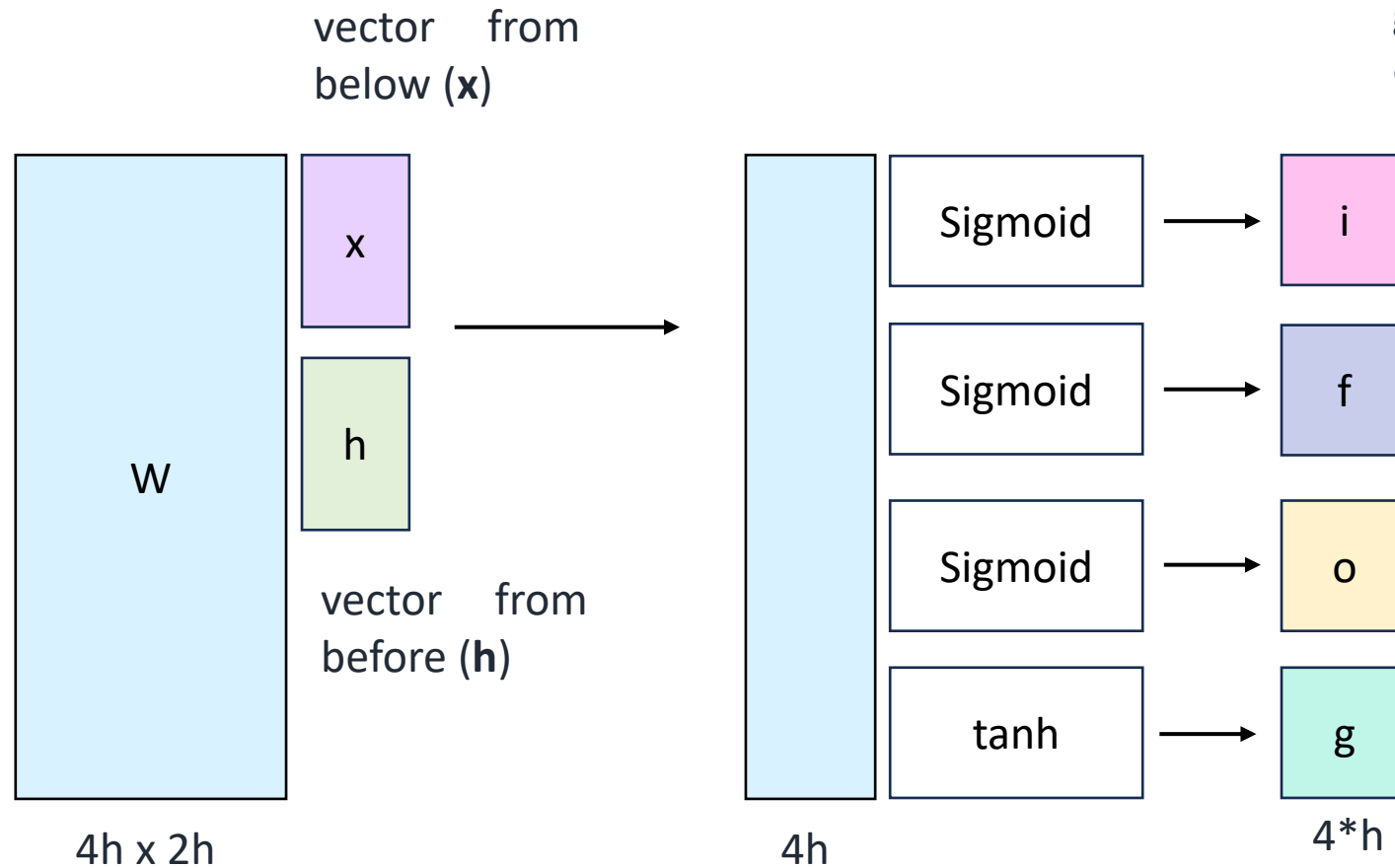
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Long Short Term Memory (LSTM)

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell

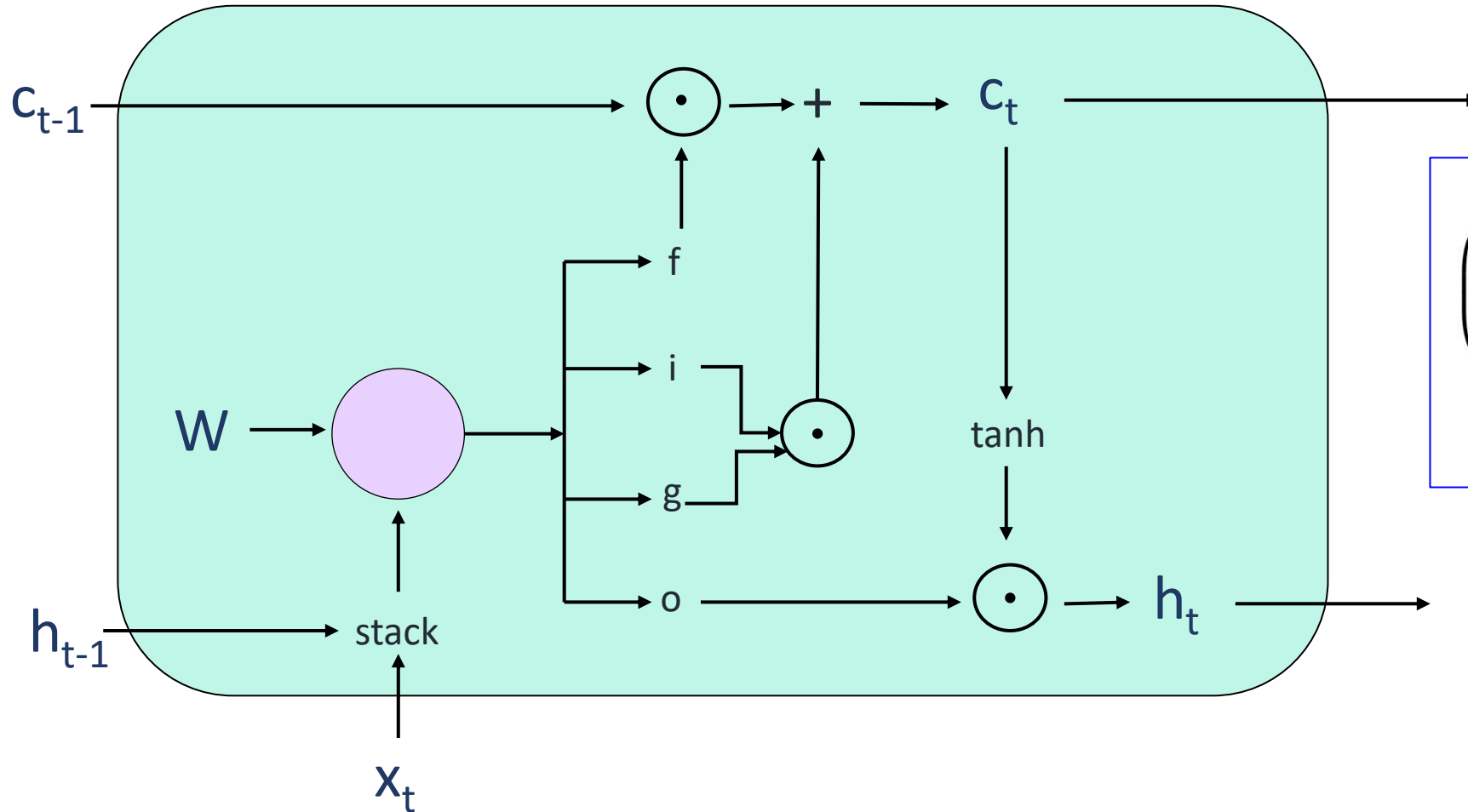


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

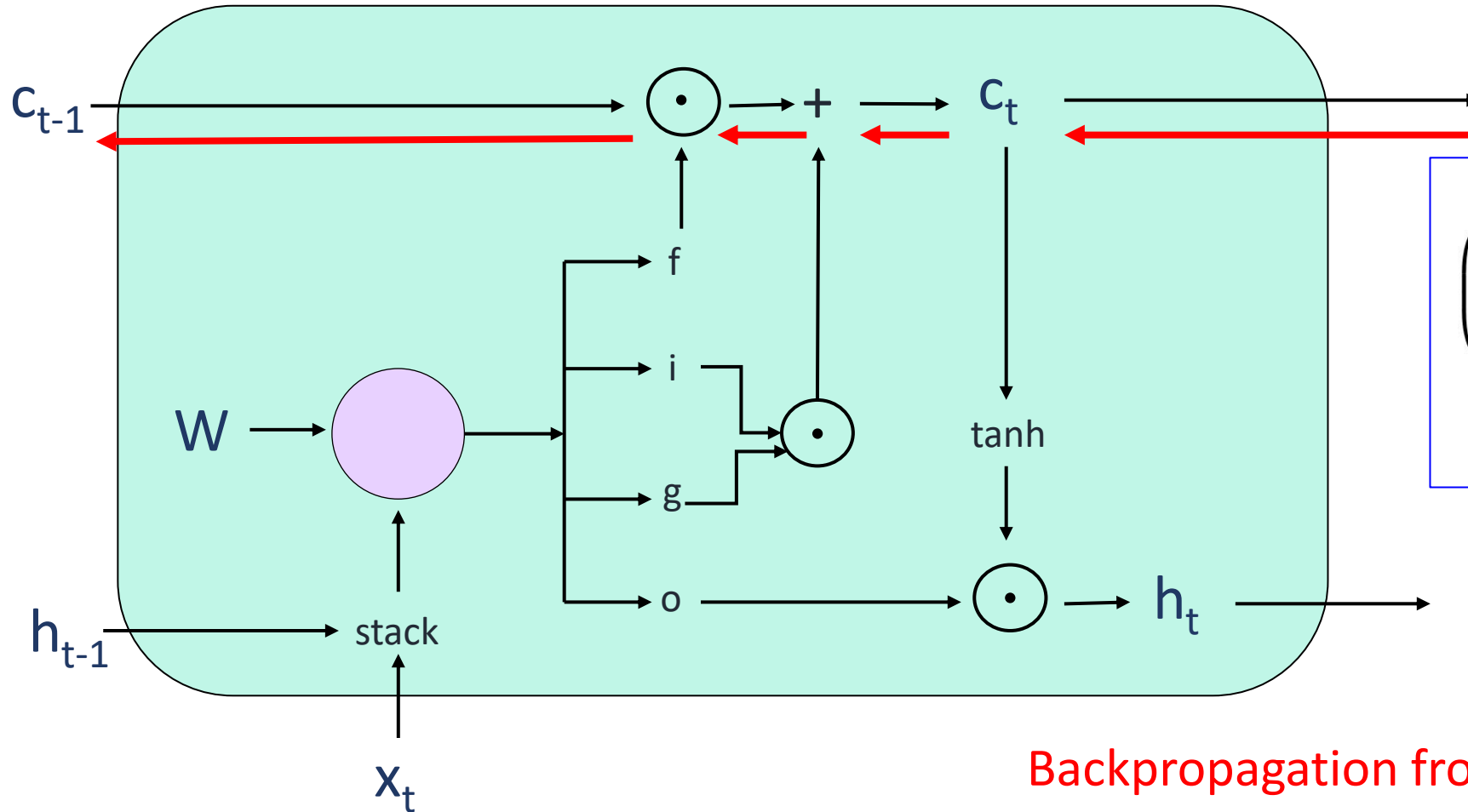
$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM)



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM)

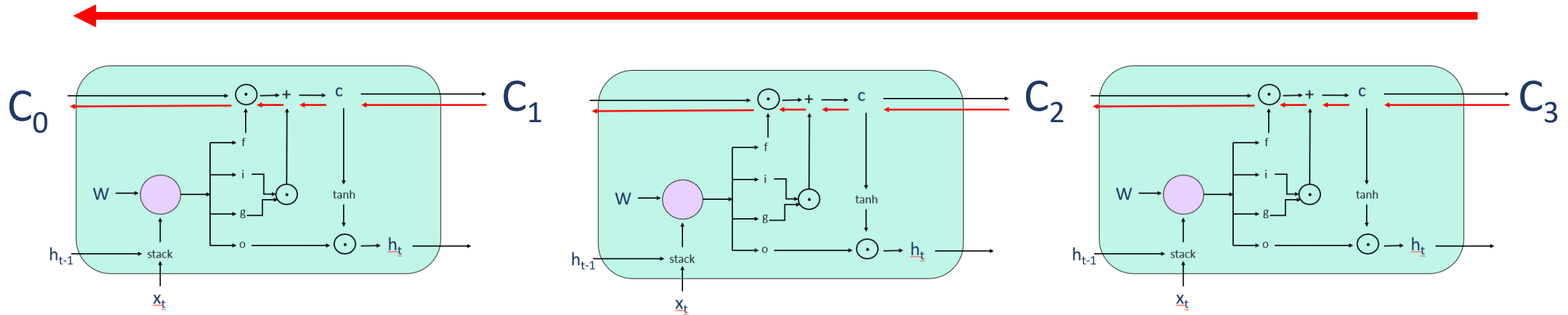


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

Backpropagation from c_t to c_{t-1} only
elementwise multiplication by f , no
matrix multiply by W

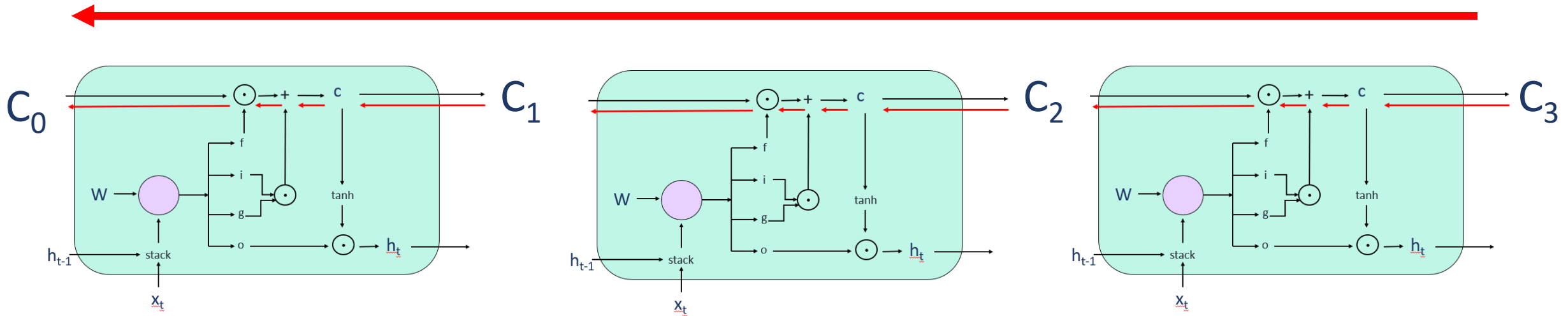
Long Short Term Memory (LSTM)

Uninterrupted gradient flow!

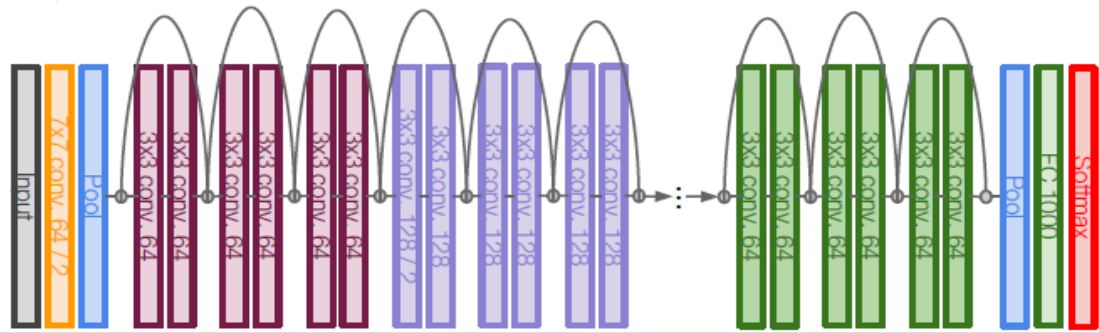


Long Short Term Memory (LSTM)

Uninterrupted gradient flow!



Similar to
ResNet!



In between:
Highway Networks

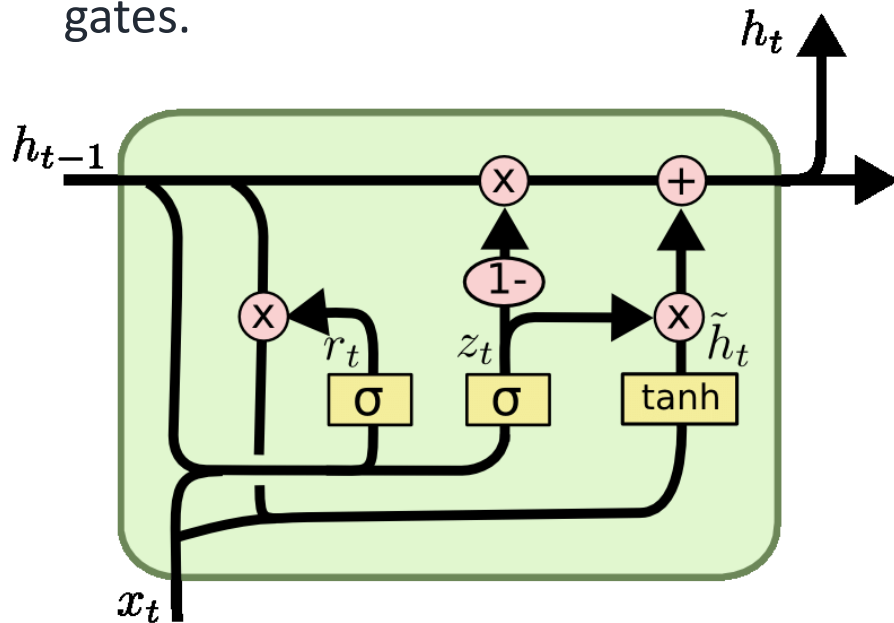
$$g = T(x, W_T)$$

$$y = g \odot H(x, W_H) + (1 - g) \odot x$$

Other RNN Variants

Gated Recurrent Unit (GRU)

Alternative RNN to LSTM that uses fewer gates.



Combines forget and input gates to “update” gate

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$

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

Eliminates cell state vector

Summary

- RNNs allow a lot of **flexibility** in architectural design.
- Vanilla RNNs are simple but don't work very well.
- Common to use LSTM or GRU: their additive interactions improve gradient flow.
- Backward flow of gradients in RNN can explode or vanish.
Exploding is controlled with **gradient clipping**. **Vanishing** is controlled with **additive interactions (LSTM)**.
- **Better/simpler architectures** are a hot topic of current research.
- Better understanding (both theoretical and empirical) is needed.

T

H

A

N

K

Y

O

U