Sequence Modeling by RNNs

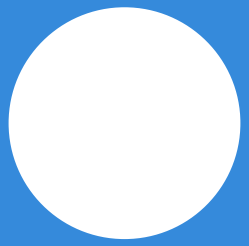
textbook: Deep Learning, An MIT Press book

http://www.deeplearningbook.org/

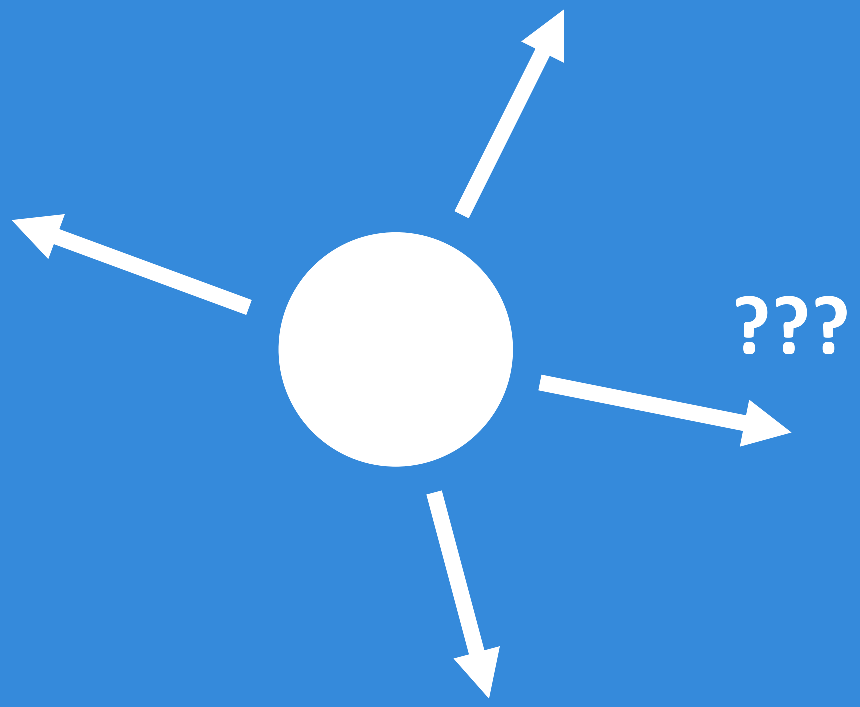
slides by: Ava Soleimany, MIT

MIT EECS

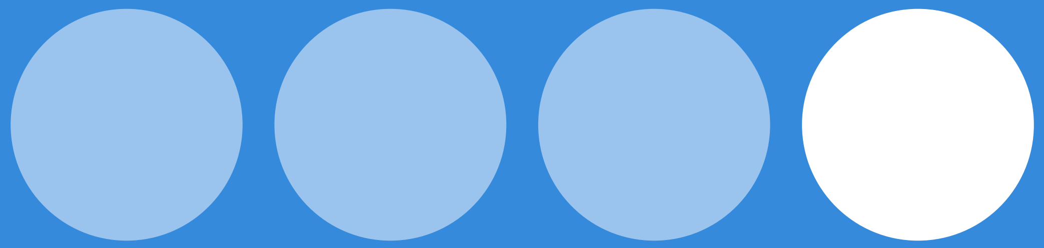
Given an image of a ball,
can you predict where it will go next?



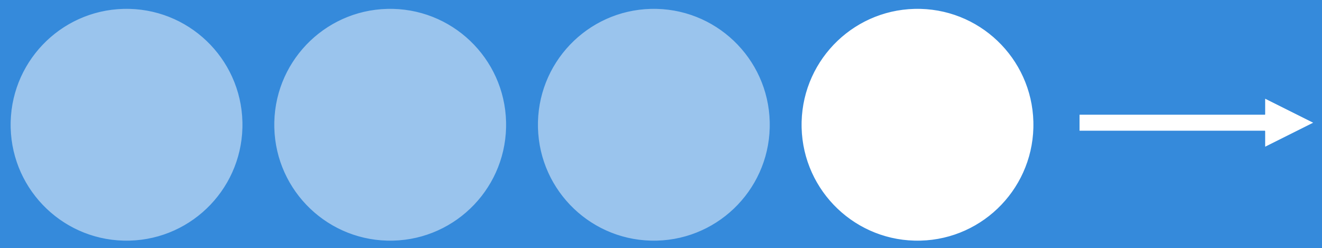
Given an image of a ball,
can you predict where it will go next?



Given an image of a ball,
can you predict where it will go next?



Given an image of a ball,
can you predict where it will go next?



Sequences in the wild

Audio

Sequences in the wild

Introduction to Deep Learning

Text

A Sequence Modeling Problem: Predict the Next Word

A sequence modeling problem: predict the next word

"This morning I took my cat for a walk."

Adapted from H. Suresh, 6.S191 2018

A sequence modeling problem: predict the next word

"This morning I took my cat for a walk."

given these words

Adapted from H. Suresh, 6.S191 2018

A sequence modeling problem: predict the next word

"This morning I took my cat for a walk."

given these words

predict the next word

Adapted from H. Suresh, 6.S191 2018

Idea #1: use a fixed window

"This morning I took my cat for a walk."

given these two words

predict the next word

Adapted from H. Suresh, 6.S191 2018

Idea #1: use a fixed window

"This morning I took my cat for a walk."

given these two words

predict the next word

One-hot feature encoding: tells us what each word is

[1 0 0 0 0 0 1 0 0 0]

for a

prediction

Adapted from H. Suresh, 6.S191 2018

Problem #1: can't model long-term dependencies

"**Sweden** is where I grew up, but I now live in **Berlin**. I speak fluent ___."

We need information from **the distant past** to accurately predict the correct word.

Adapted from H. Suresh, 6.S191 2018

Idea #2: use entire sequence as set of counts

"This morning I took my cat for a"

↓

"bag of words"

[0 1 0 0 1 0 0 ... 0 0 1 1 0 0 0 1]

↓

prediction

Adapted from H. Suresh, 6.S191 2018

Problem #2: counts don't preserve order

The food was good, not bad at all.

vs.

The food was bad, not good at all.

Adapted from H. Suresh, 6.S191 2018

Idea #3: use a really big fixed window

"This morning I took my cat for a walk."

given these words

predict the next word

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

morning I took this cat

↓

prediction

Adapted from H. Suresh, 6.S191 2018

Problem #3: no parameter sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter:**

Adapted from H. Suresh, 6.S191 2018

Problem #3: no parameter sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter:**

[0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 ...]

this morning

Adapted from H. Suresh, 6.S191 2018

Problem #3: no parameter sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter:**

[0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 ...]

this morning

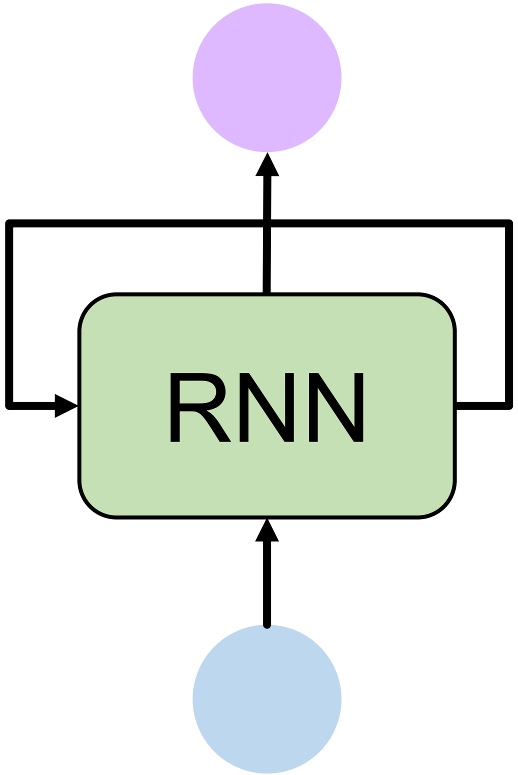
Things we learn about the sequence **won't transfer** if they appear **elsewhere** in the sequence.

Adapted from H. Suresh, 6.S191 2018

Sequence modeling: design criteria

To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence

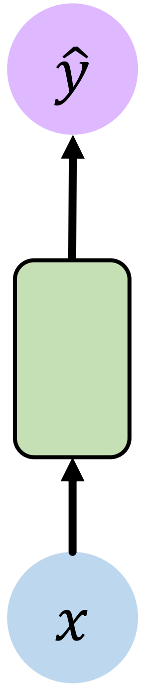


Today: **Recurrent Neural Networks (RNNs)** as an approach to sequence modeling problems

Adapted from H. Suresh, 6.S191 2018

Recurrent Neural Networks (RNNs)

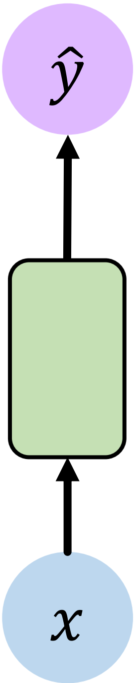
Standard feed-forward neural network



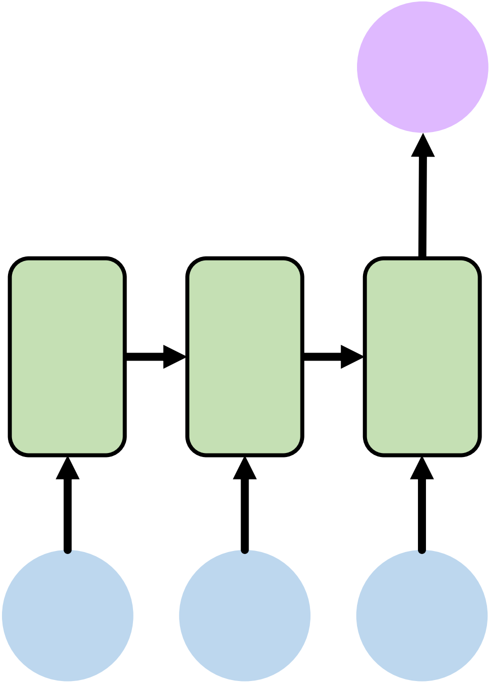
One to One
"Vanilla" neural network

[1]

Recurrent neural networks: sequence modeling



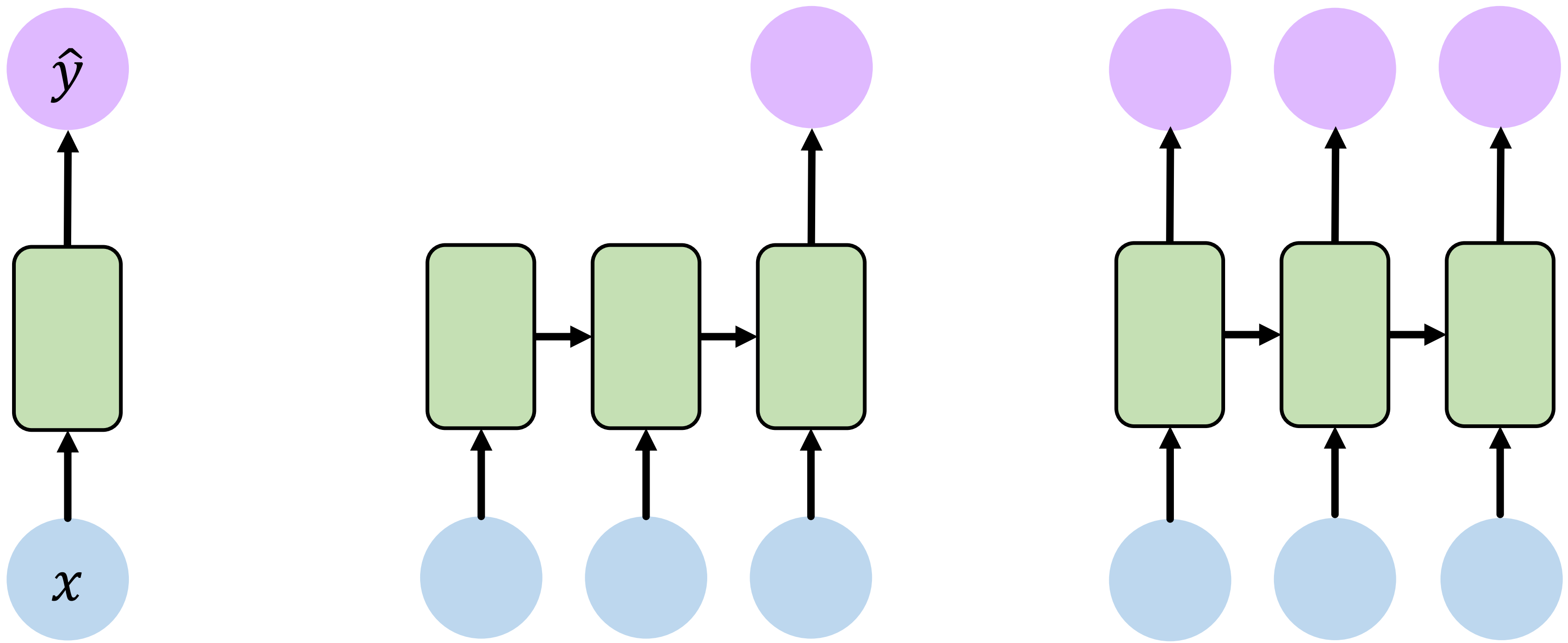
One to One
"Vanilla" neural network



Many to One
Sentiment Classification

[1]

Recurrent neural networks: sequence modeling



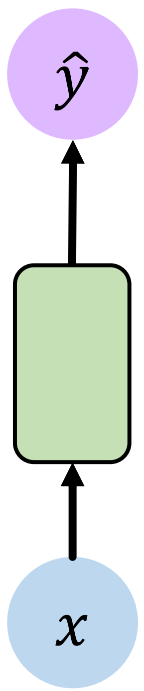
One to One
"Vanilla" neural network

Many to One
Sentiment Classification

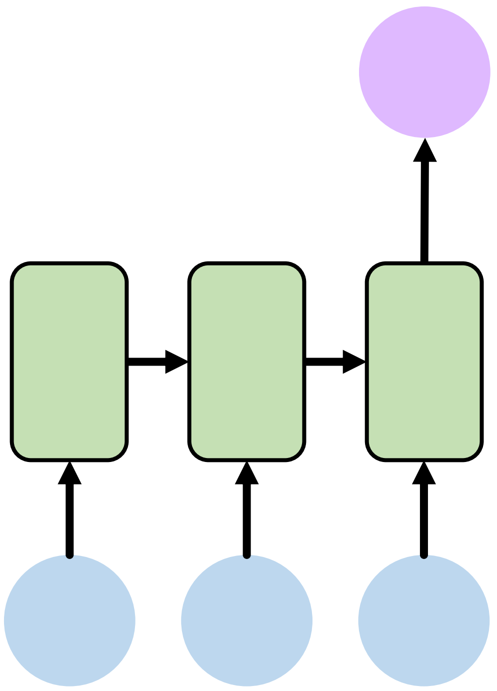
Many to Many
Music Generation

[1]

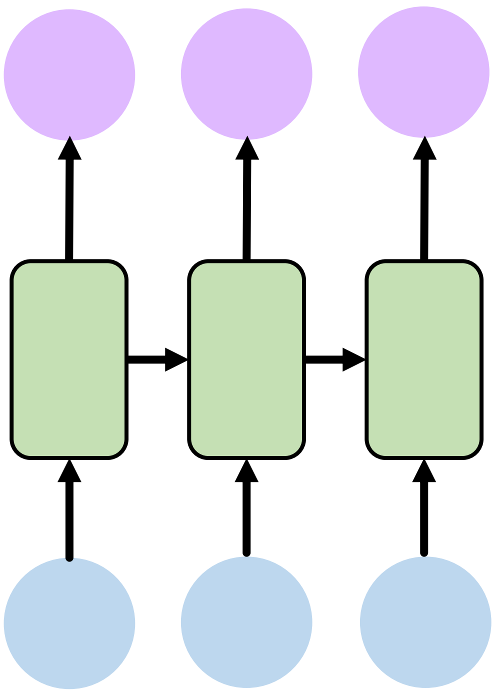
Recurrent neural networks: sequence modeling



One to One
"Vanilla" neural network



Many to One
Sentiment Classification

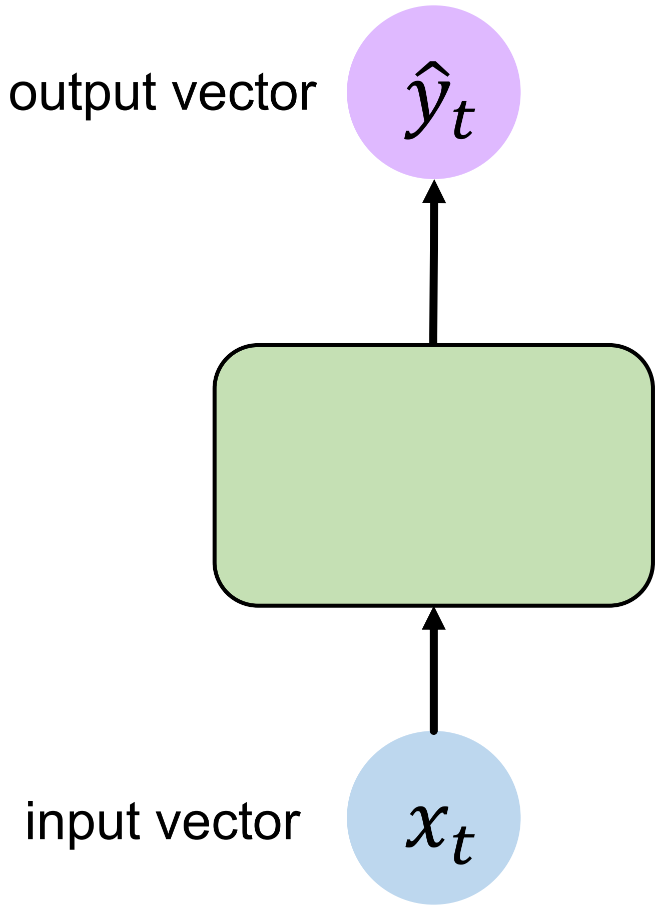


Many to Many
Music Generation

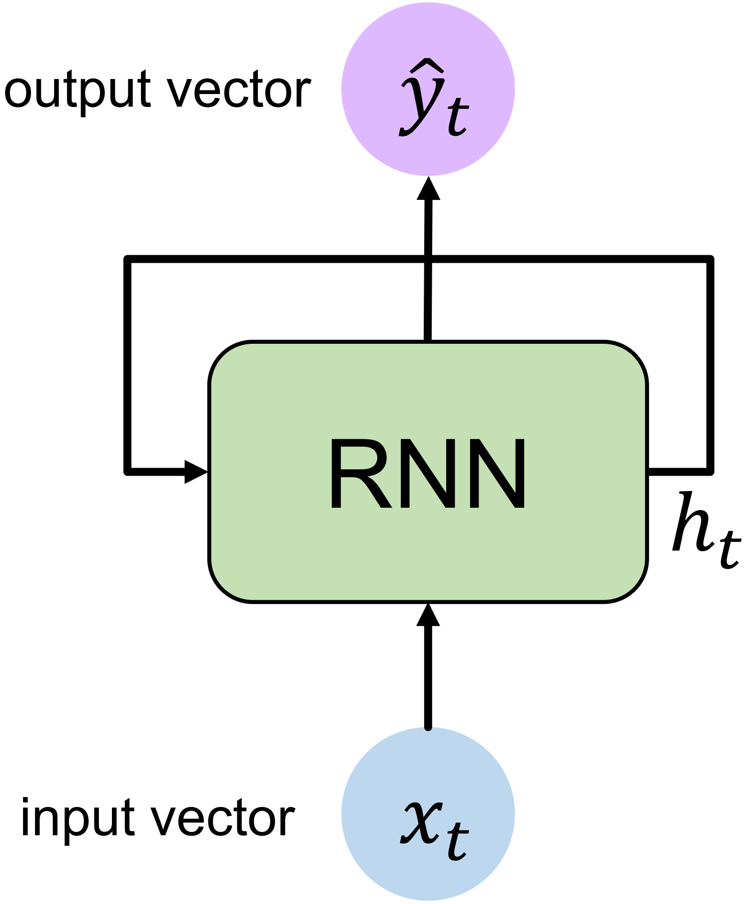
… and many other architectures and applications

[1]

A standard "vanilla" neural network

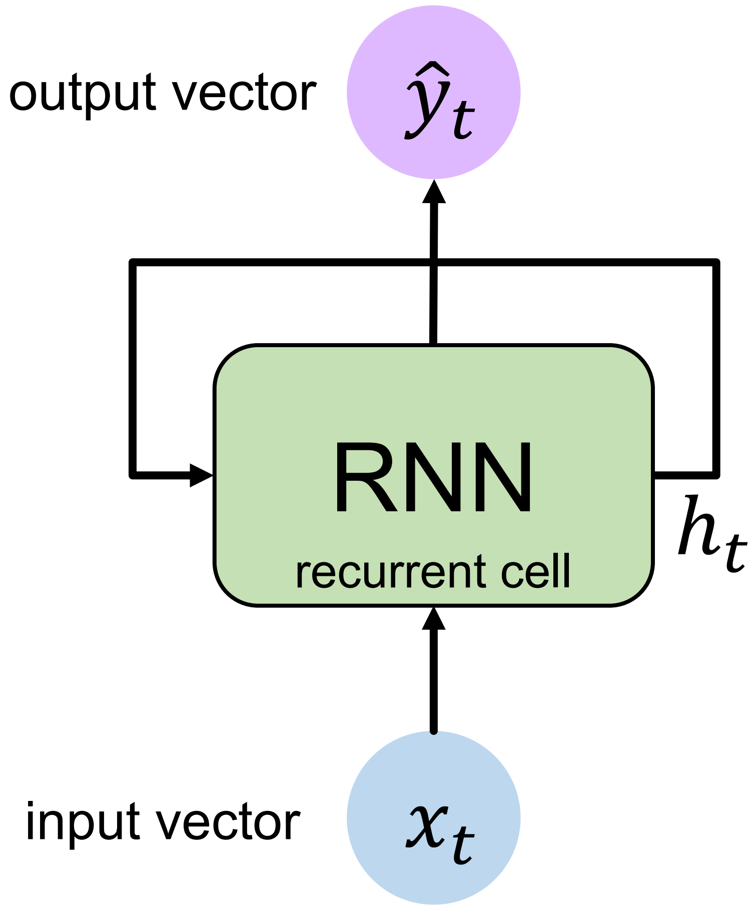


A recurrent neural network (RNN)

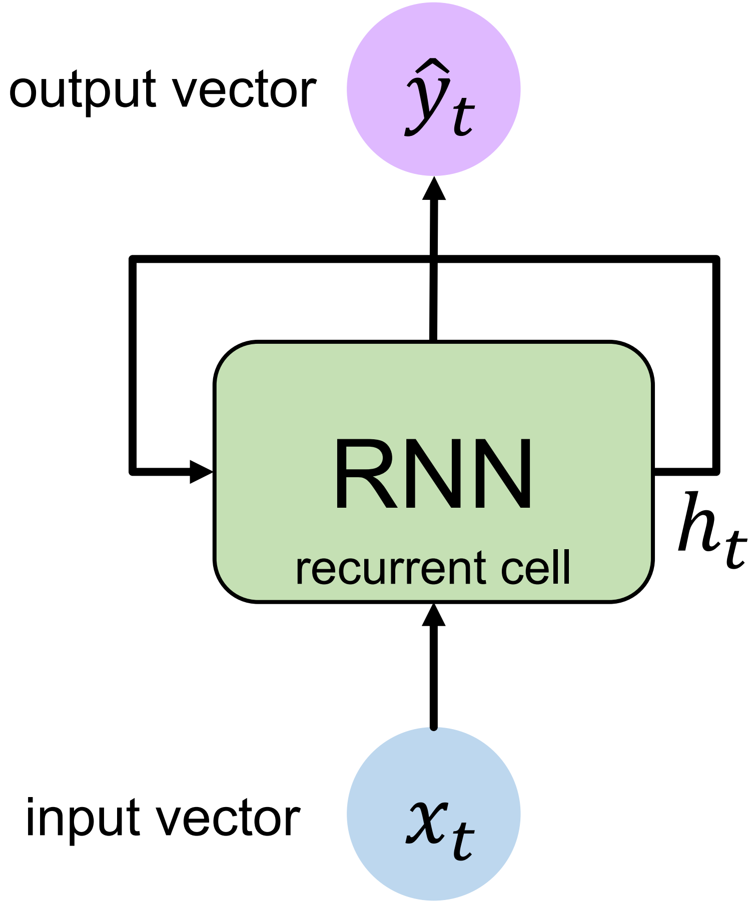


[2]

A recurrent neural network (RNN)

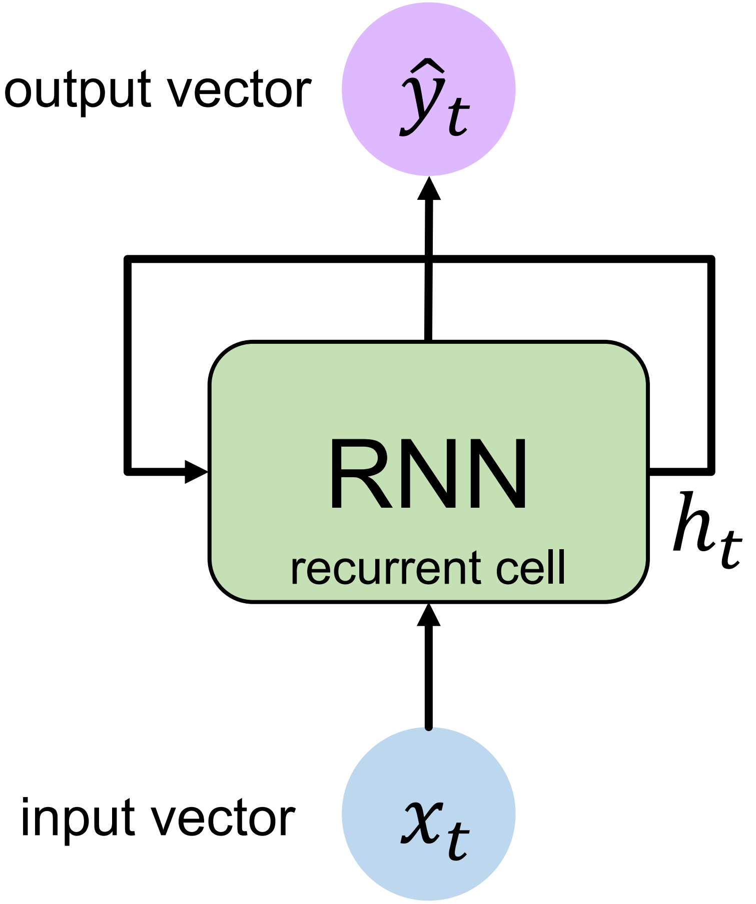


A recurrent neural network (RNN)



Apply a **recurrence relation** at every time step to process a sequence:

A recurrent neural network (RNN)



Apply a **recurrence relation** at every time step to process a sequence:

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

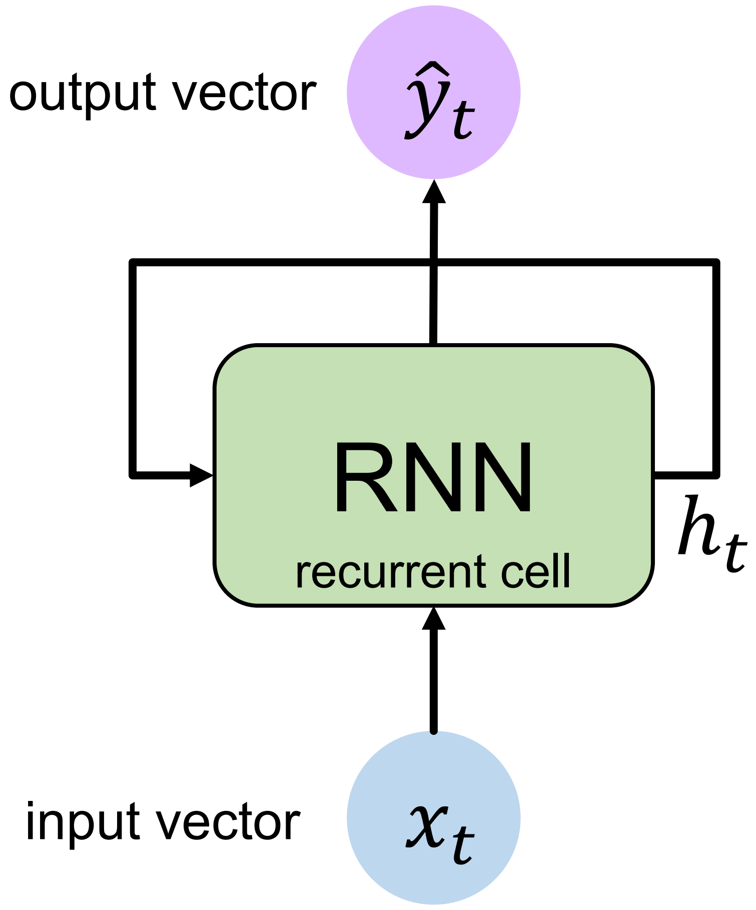
cell state

function parameterized by W

old state

input vector at time step *t*

A recurrent neural network (RNN)



Apply a **recurrence relation** at every time step to process a sequence:

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

new state

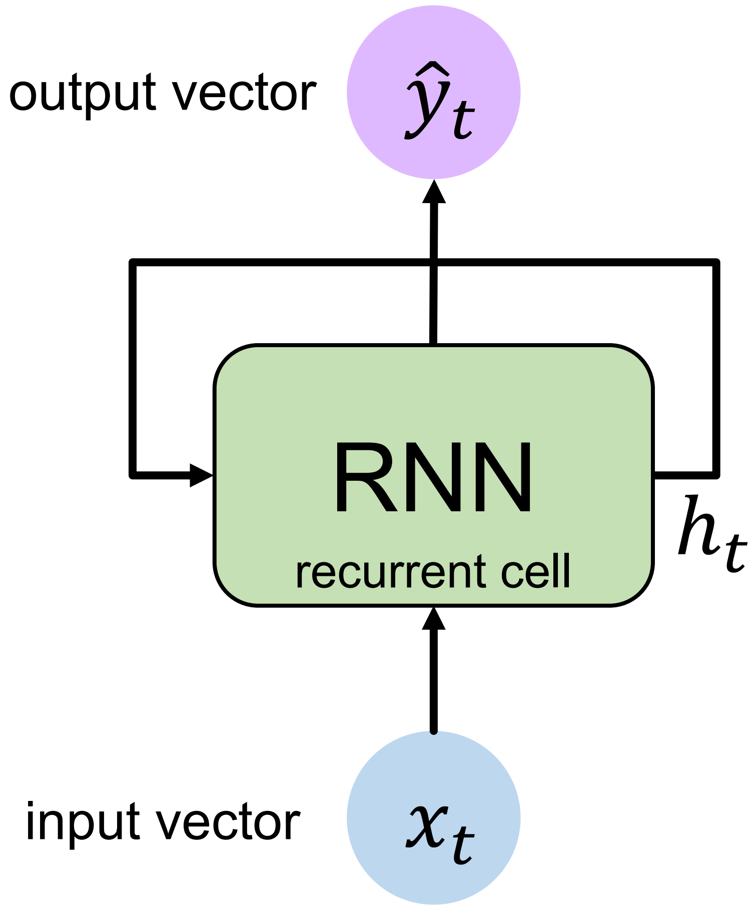
function parameterized by W

old state

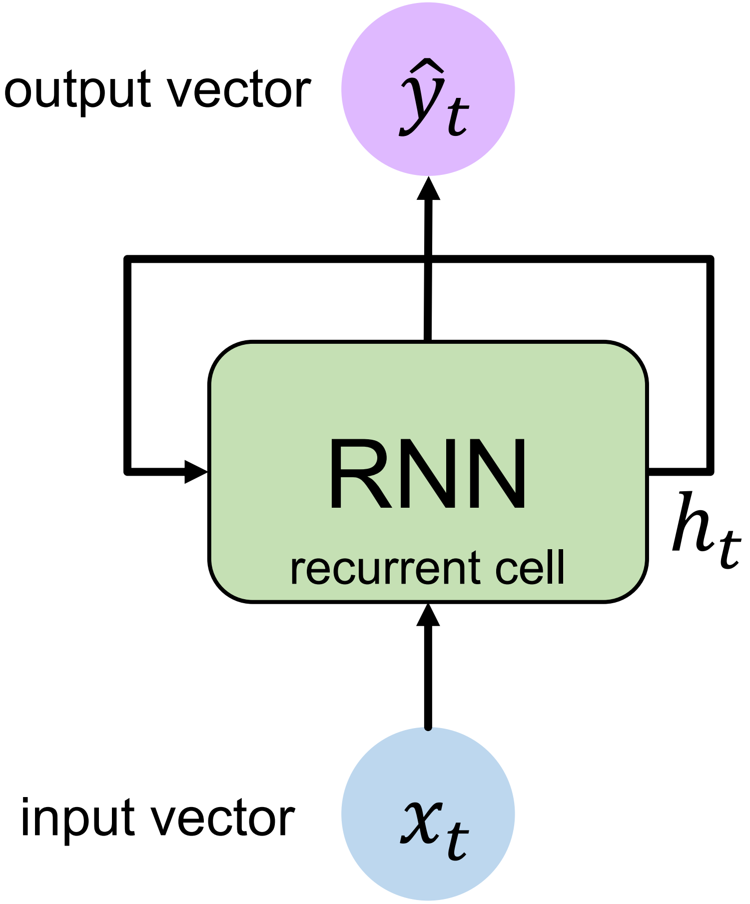
input vector at time step *t*

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

RNN state update and output

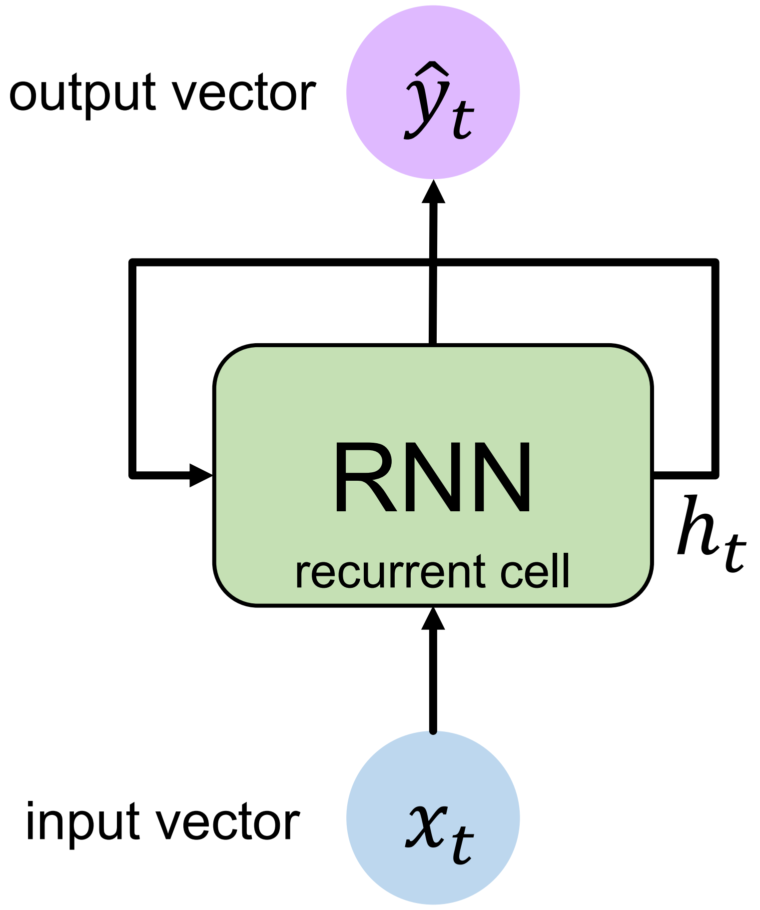


RNN state update and output



Input Vector

RNN state update and output

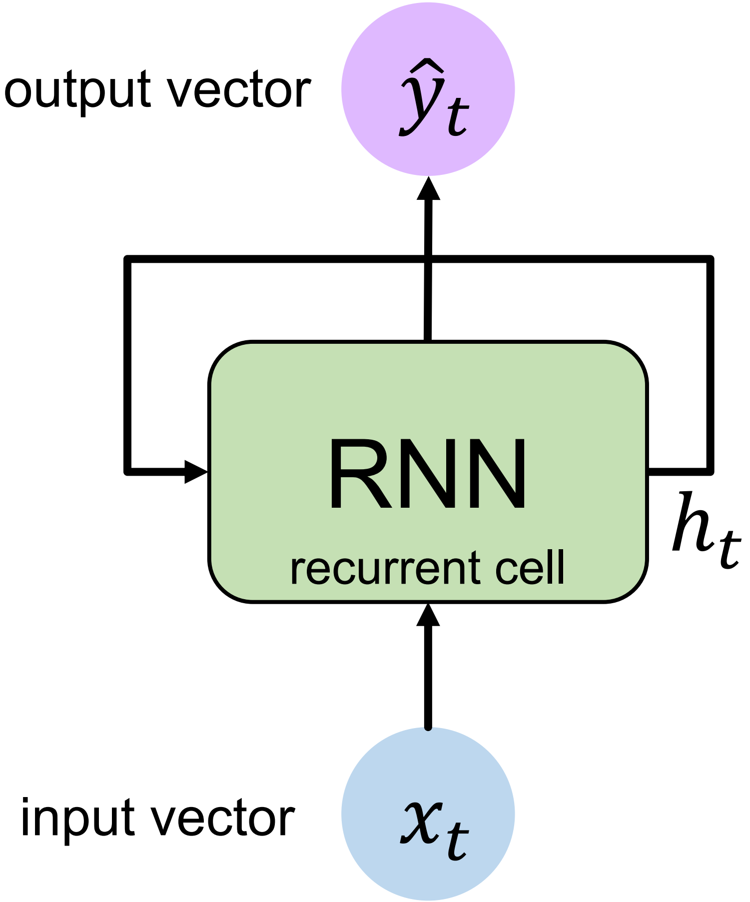


Update Hidden State

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

Input Vector

RNN state update and output



Output Vector

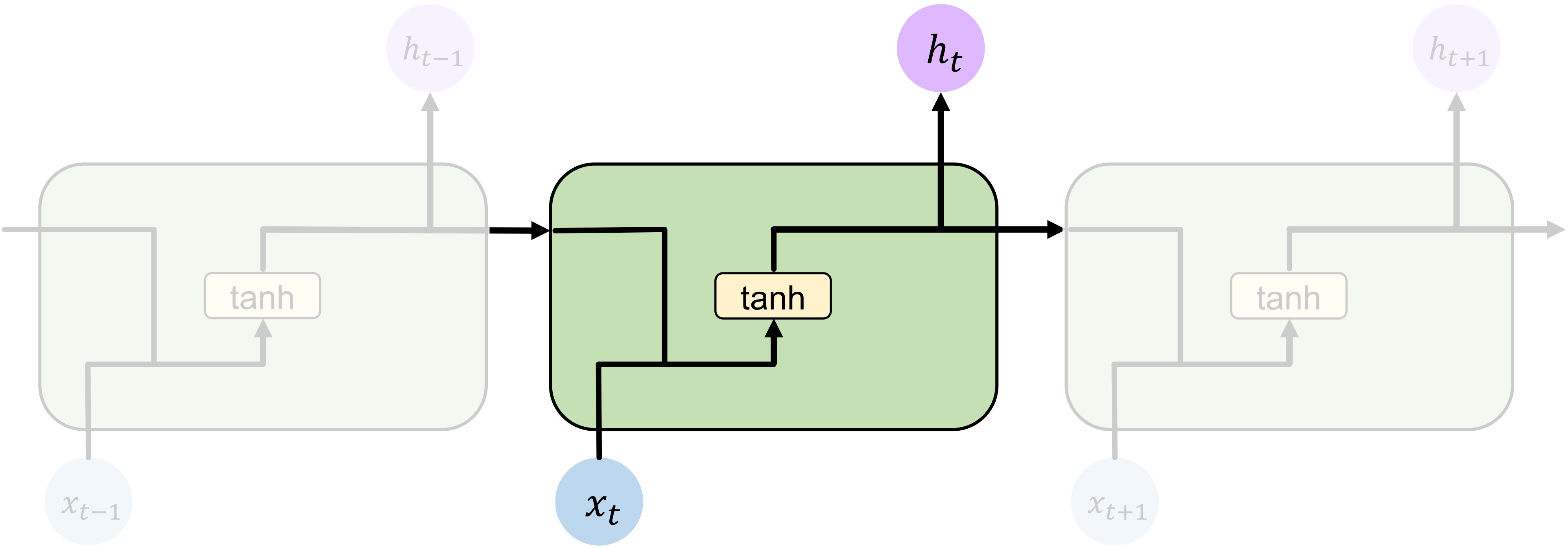
$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}} h_t$$

Update Hidden State

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

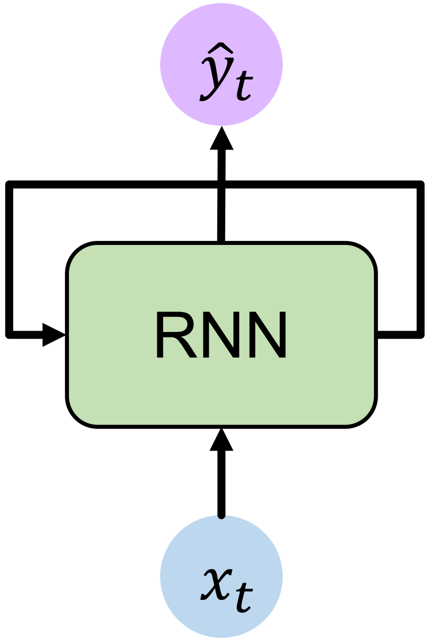
Input Vector

RNN state update and output



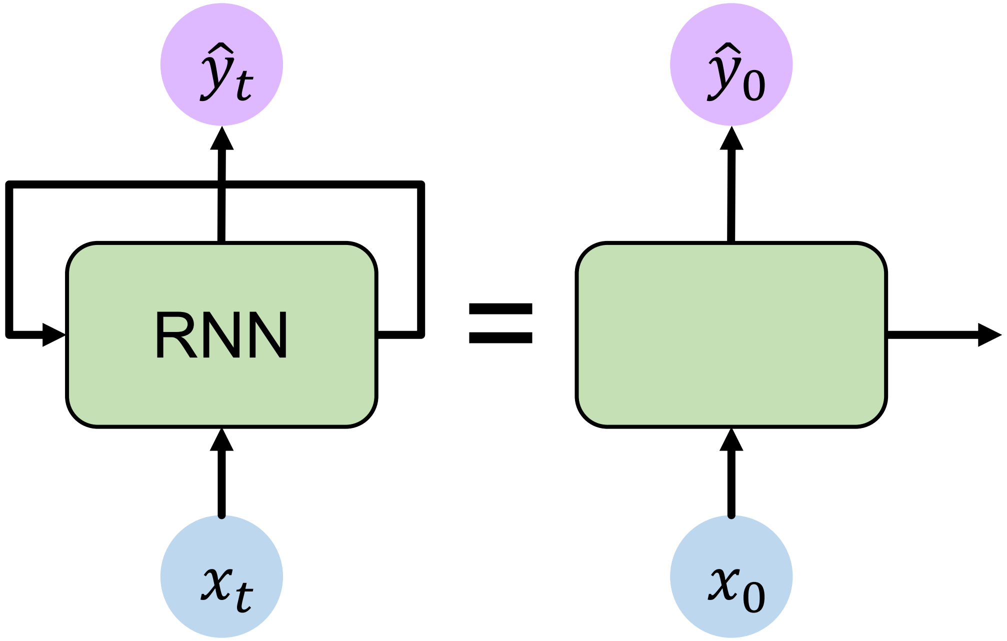
[2]

RNNs: computational graph across time

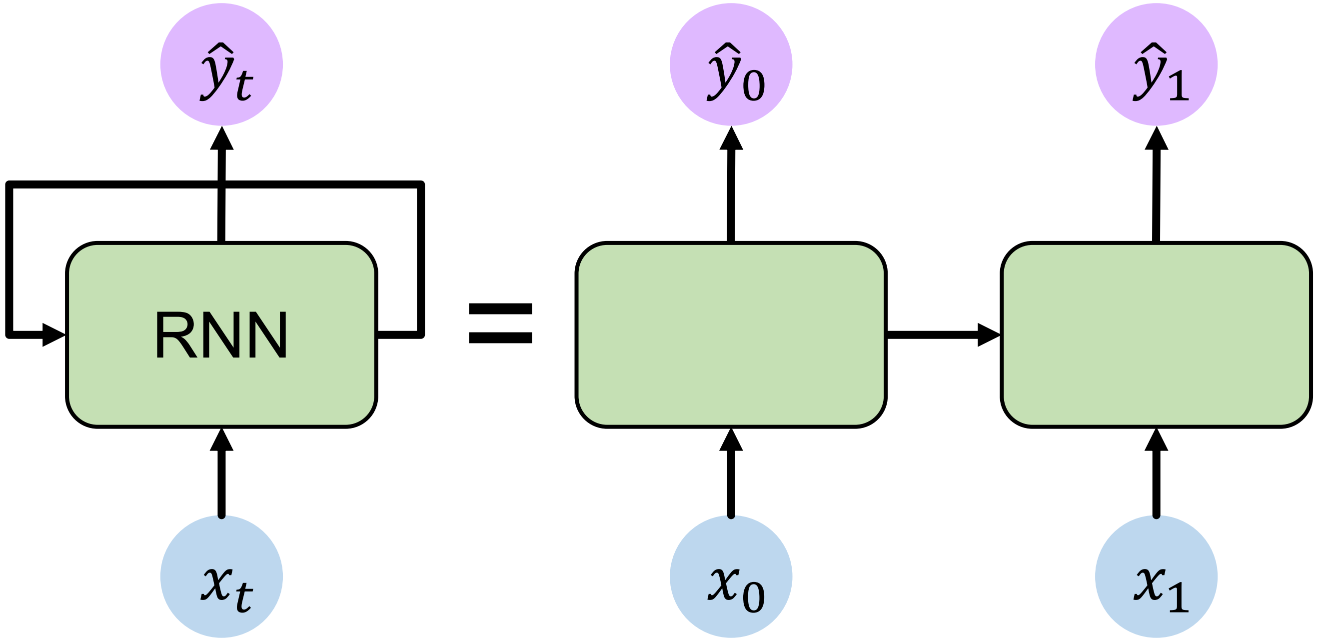


$=$ Represent as computational graph unrolled across time

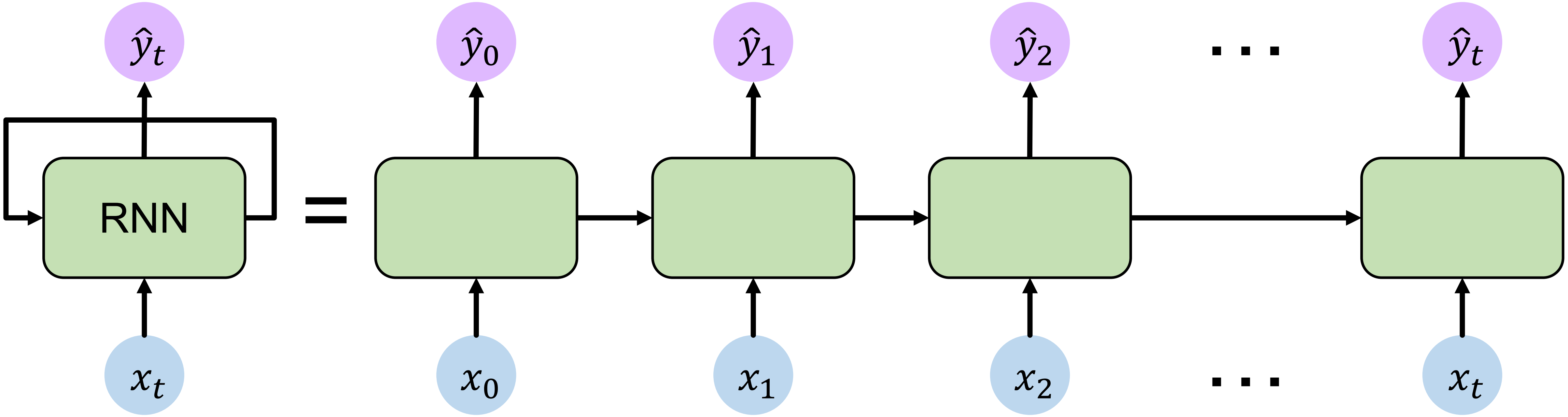
RNNs: computational graph across time



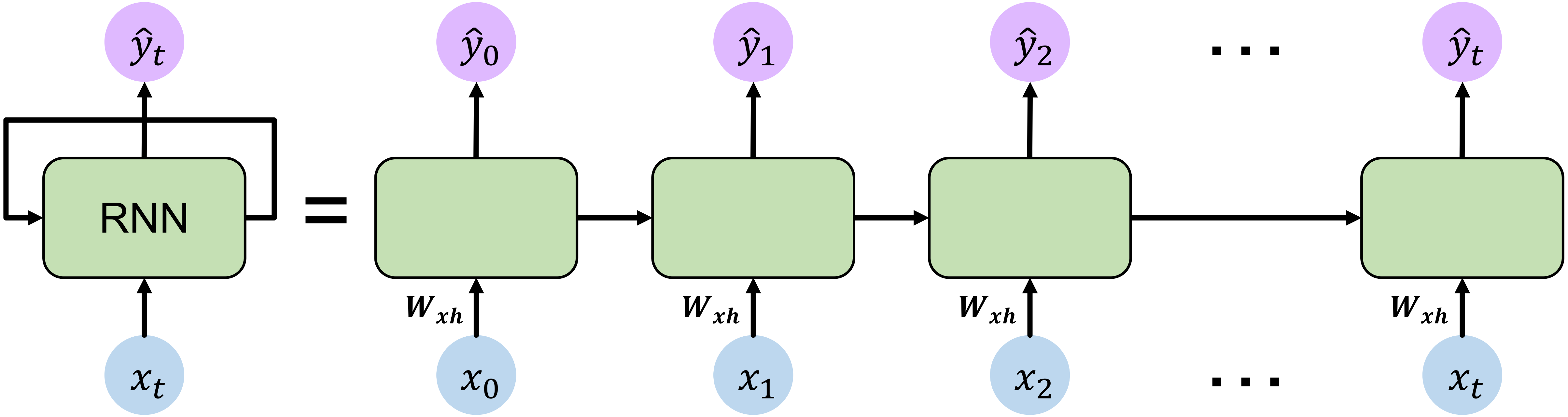
RNNs: computational graph across time



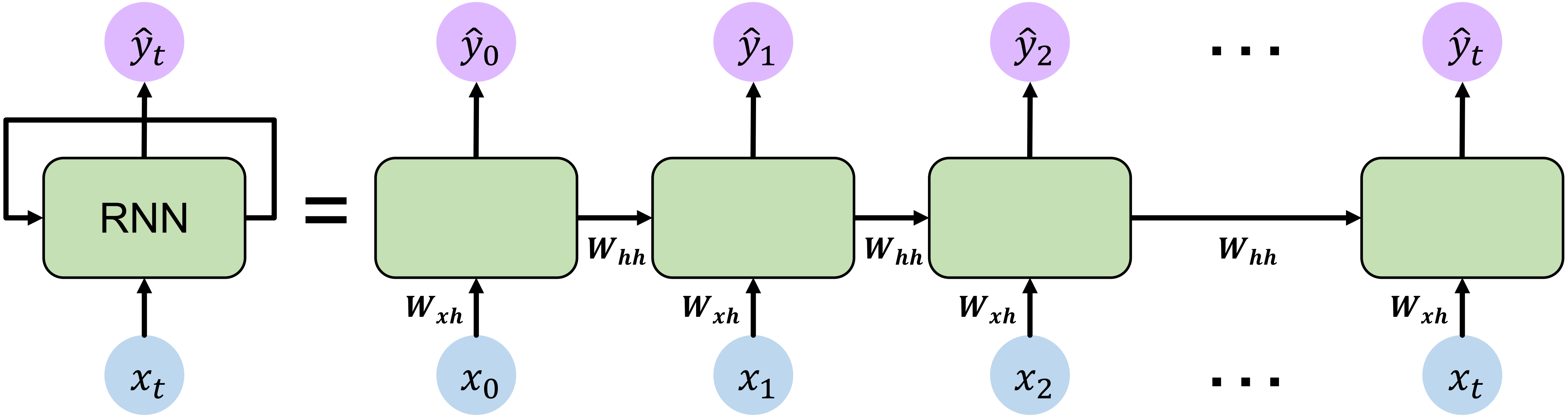
RNNs: computational graph across time



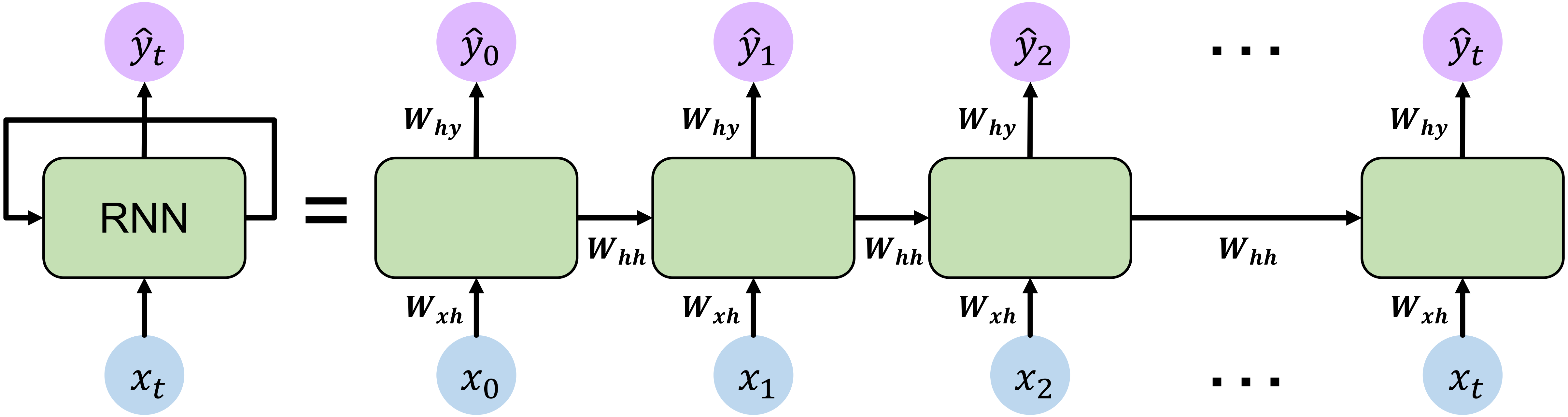
RNNs: computational graph across time



RNNs: computational graph across time

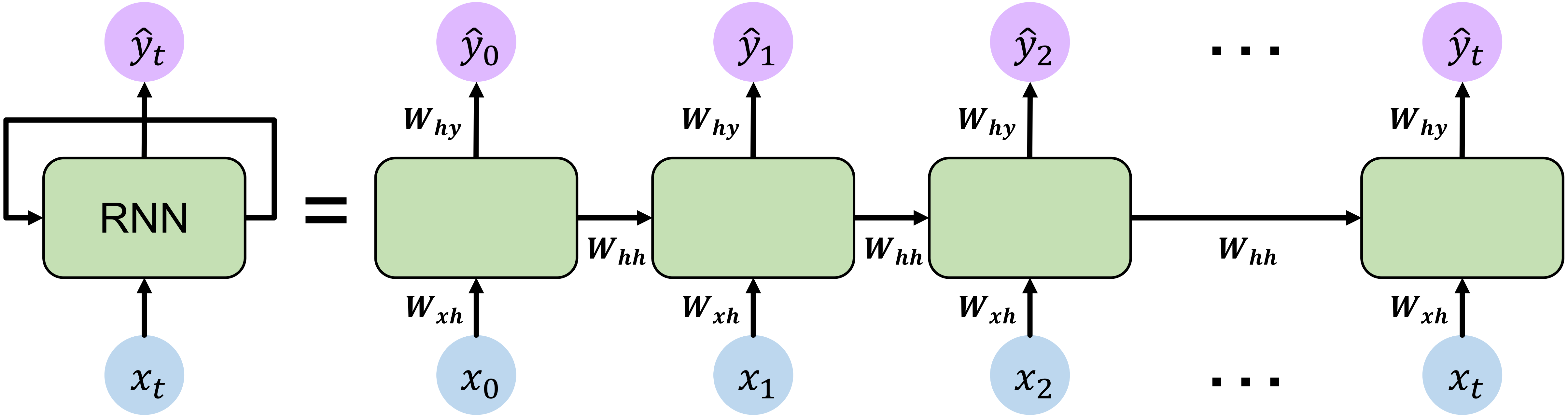


RNNs: computational graph across time



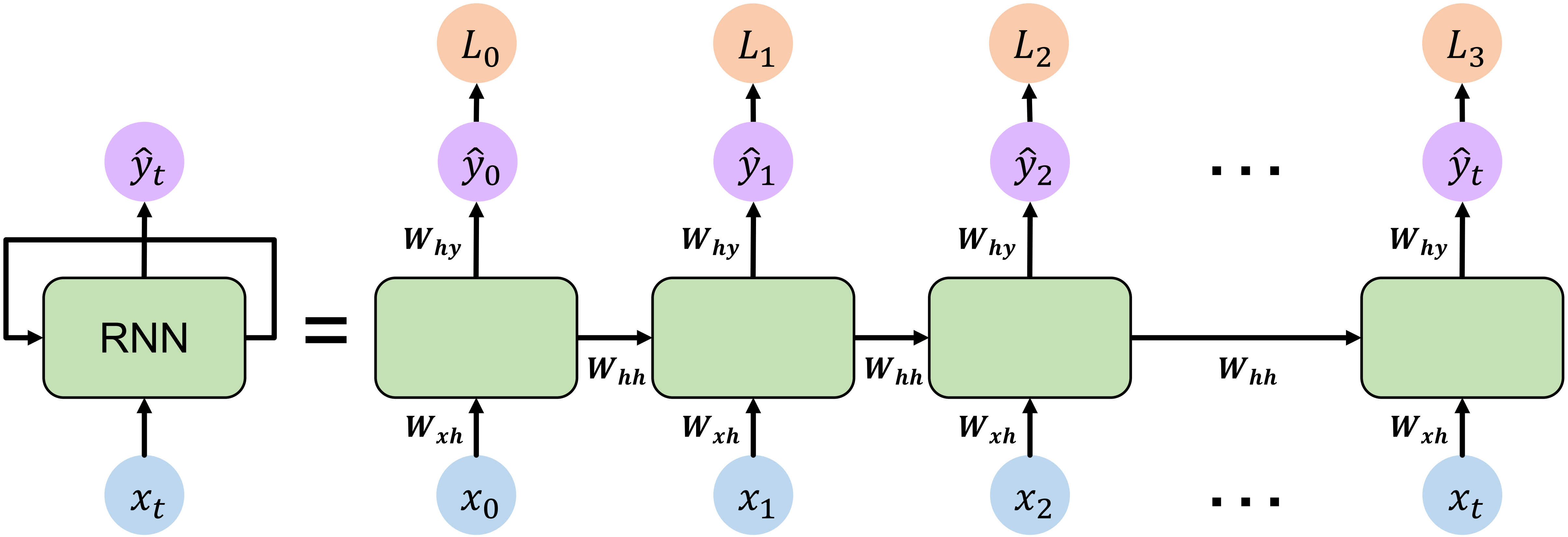
RNNs: computational graph across time

Re-use the **same weight matrices** at every time step

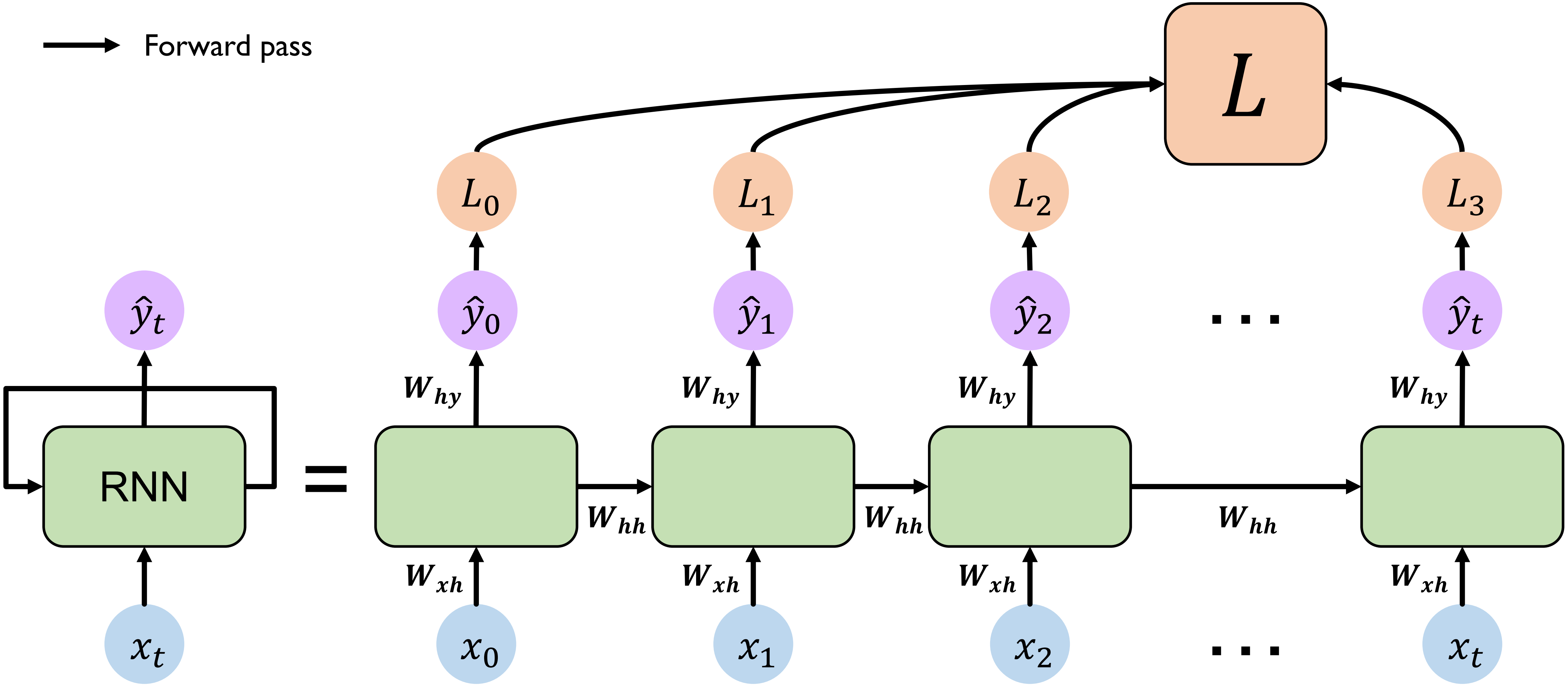


RNNs: computational graph across time

Forward pass

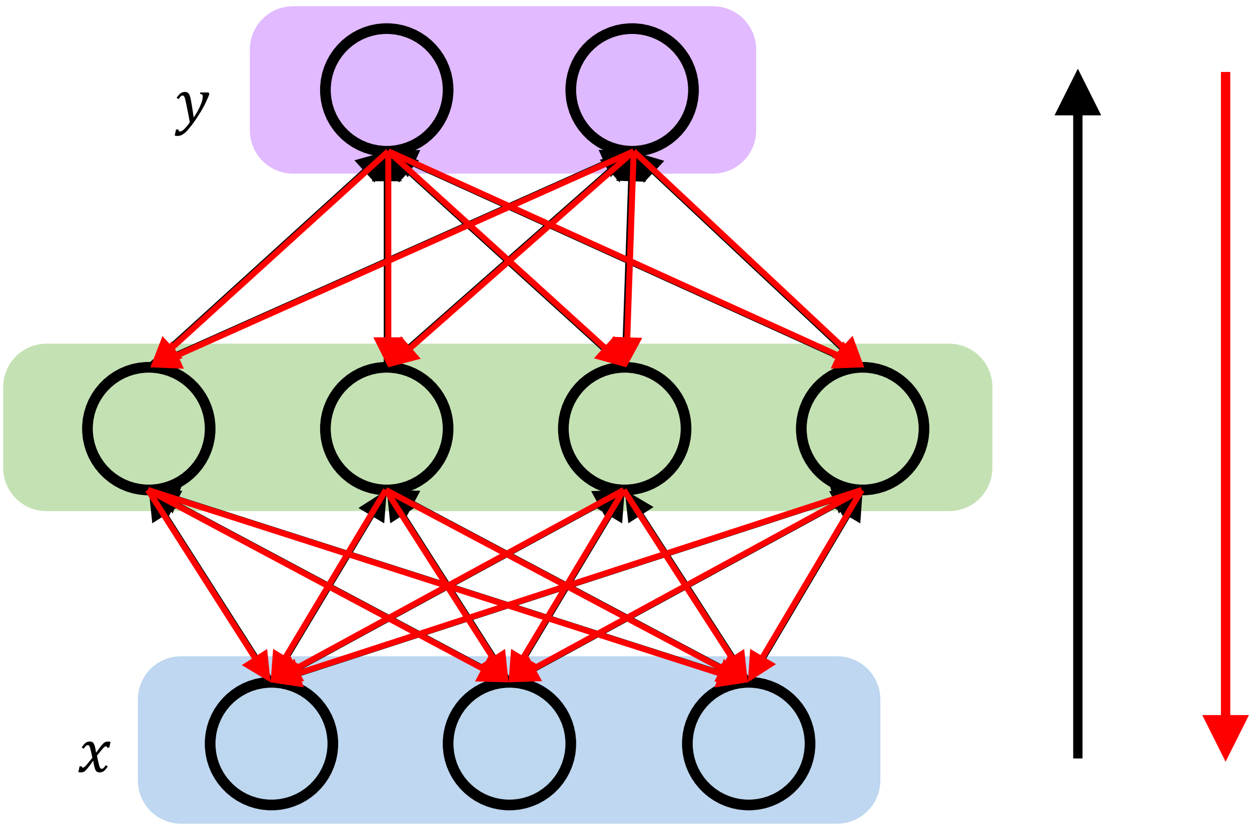


RNNs: computational graph across time



Backpropagation Through Time (BPTT)

Recall: backpropagation in feed forward models

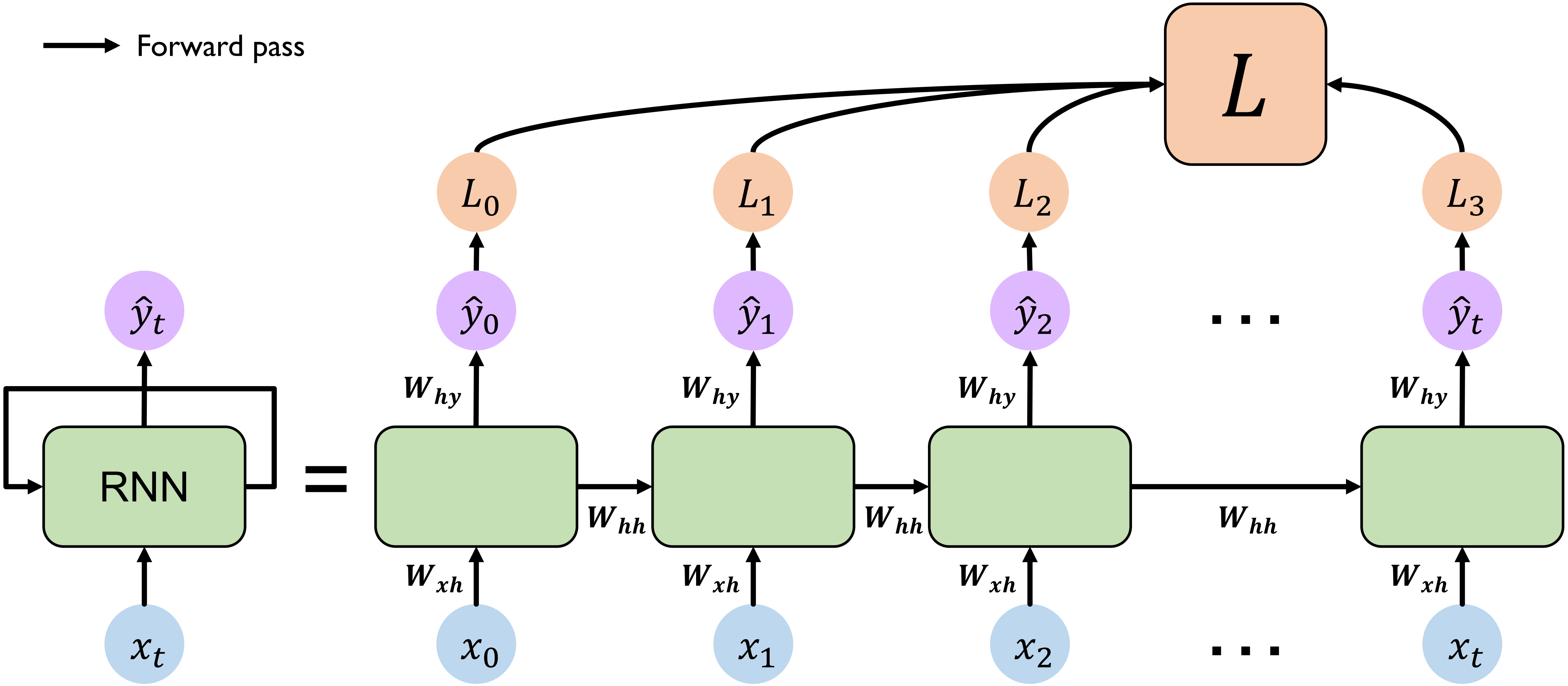


Backpropagation algorithm:

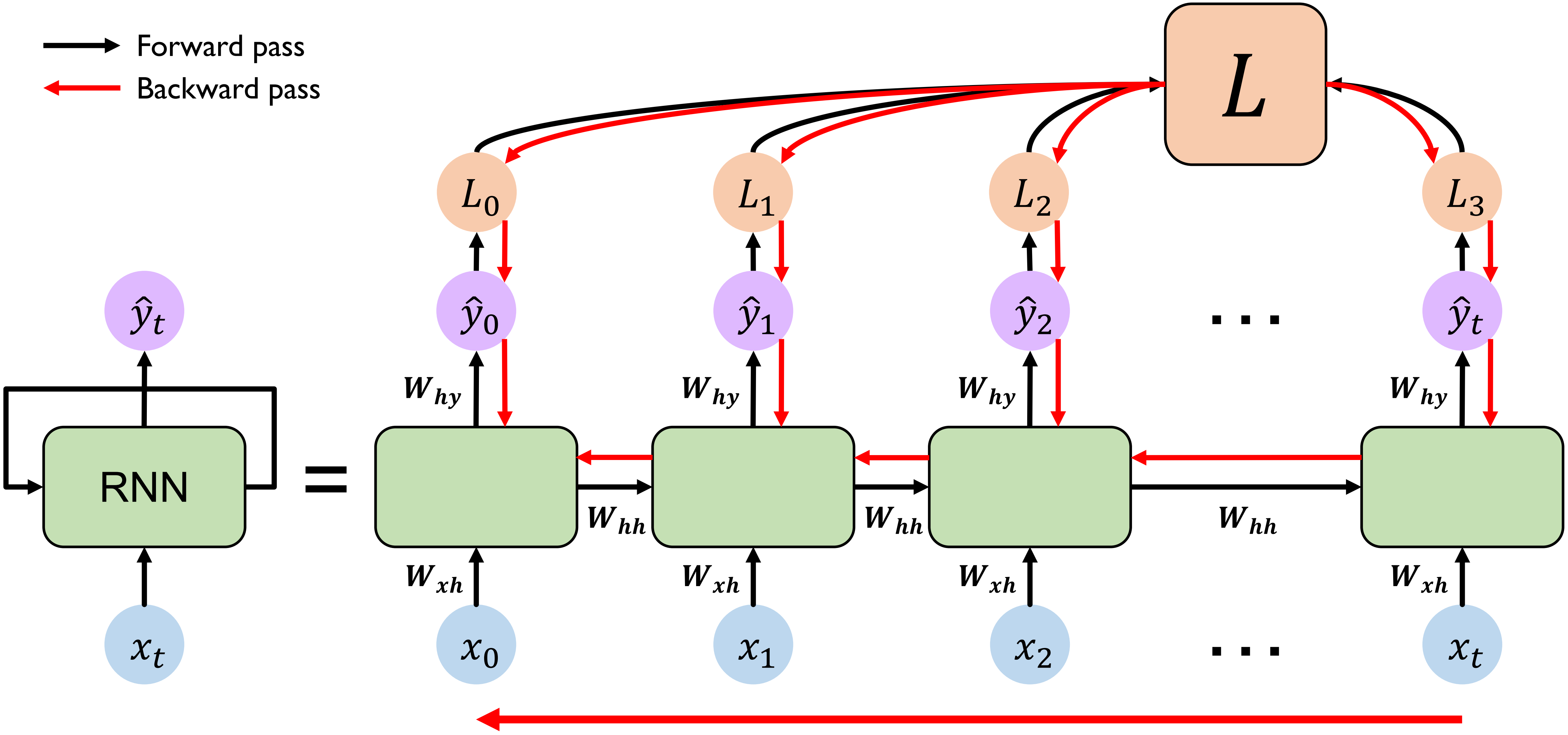
1. Take the derivative (gradient) of the loss with respect to each parameter
2. Shift parameters in order to minimize loss

[3]

RNNs: backpropagation through time

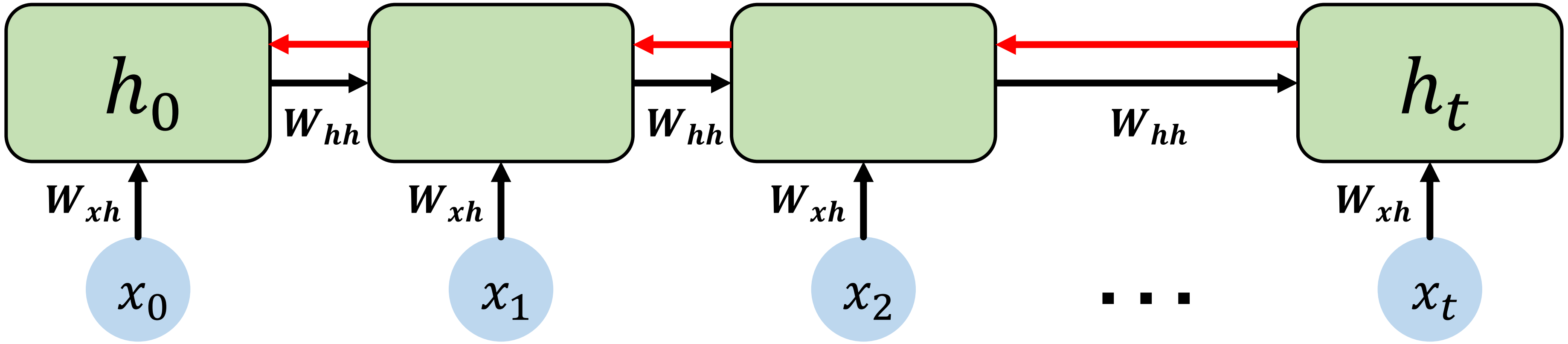


RNNs: backpropagation through time



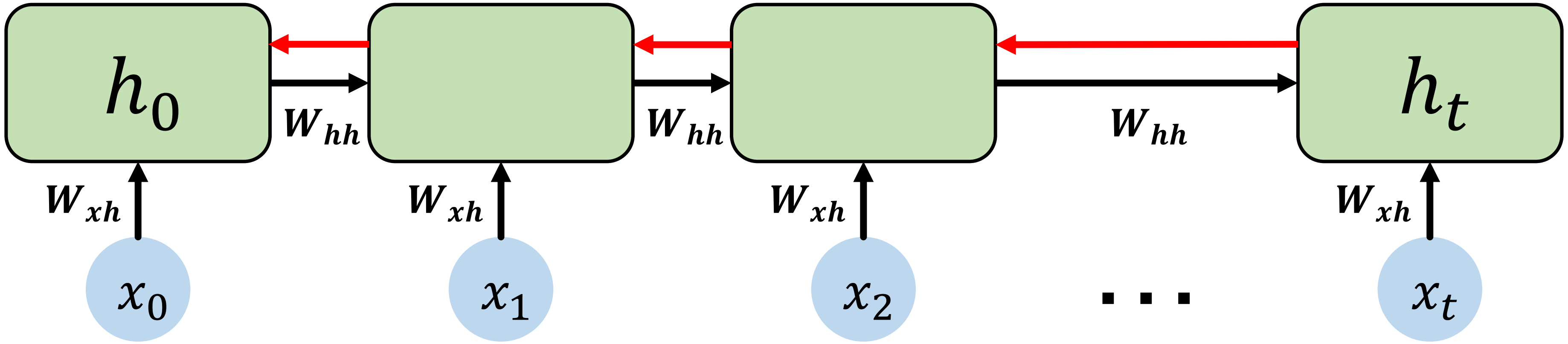
[4]

Standard RNN gradient flow



[1]

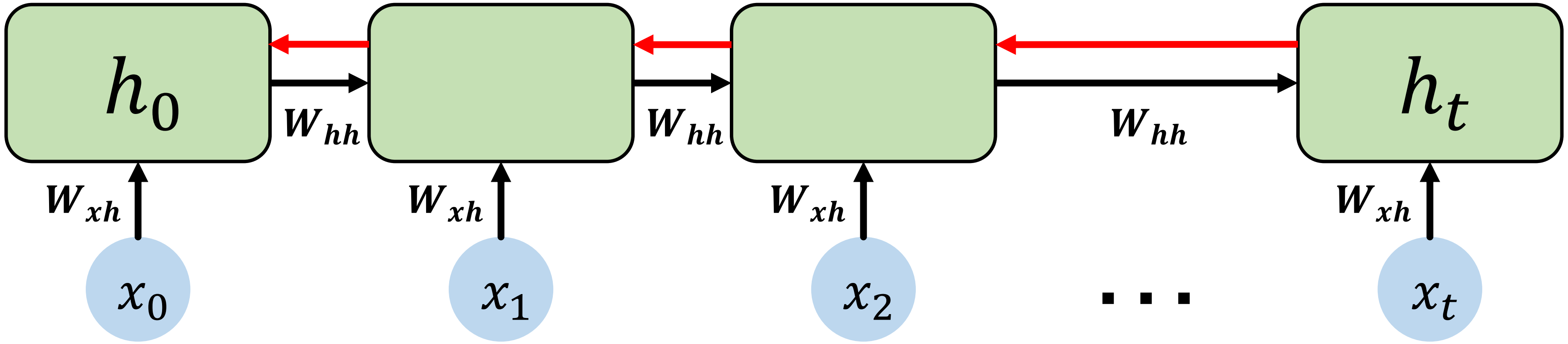
Standard RNN gradient flow



Computing the gradient wrt h_0 involves **many factors of** W_{hh} (and repeated f'!)

[1]

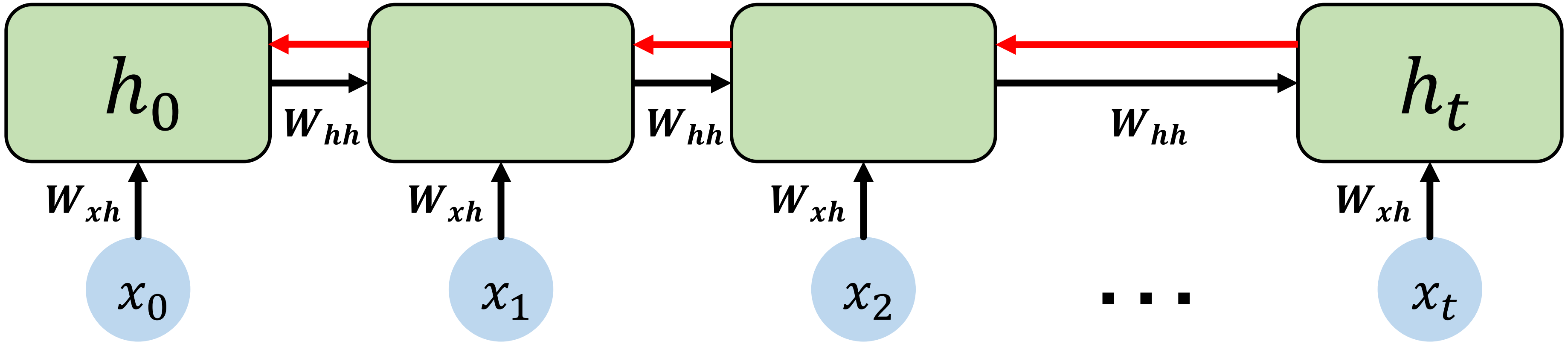
Standard RNN gradient flow: exploding gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh}** (and repeated f'!)

Many values > 1:
exploding gradients

Standard RNN gradient flow: exploding gradients



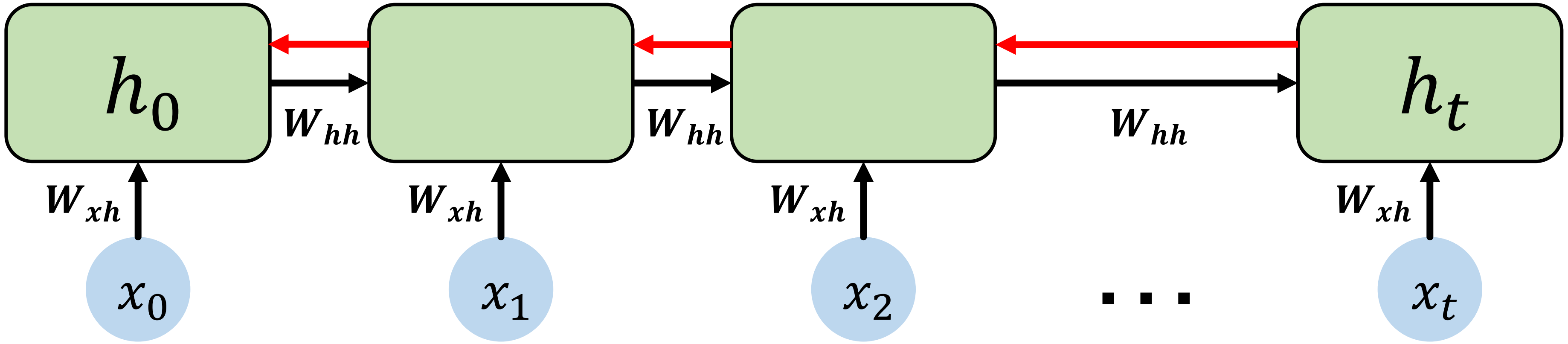
Computing the gradient wrt h_0 involves **many factors of W_{hh}** (and repeated f'!)

Many values > 1:
exploding gradients

Gradient clipping to
scale big gradients

[1]

Standard RNN gradient flow: vanishing gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh}** (and repeated f'!)

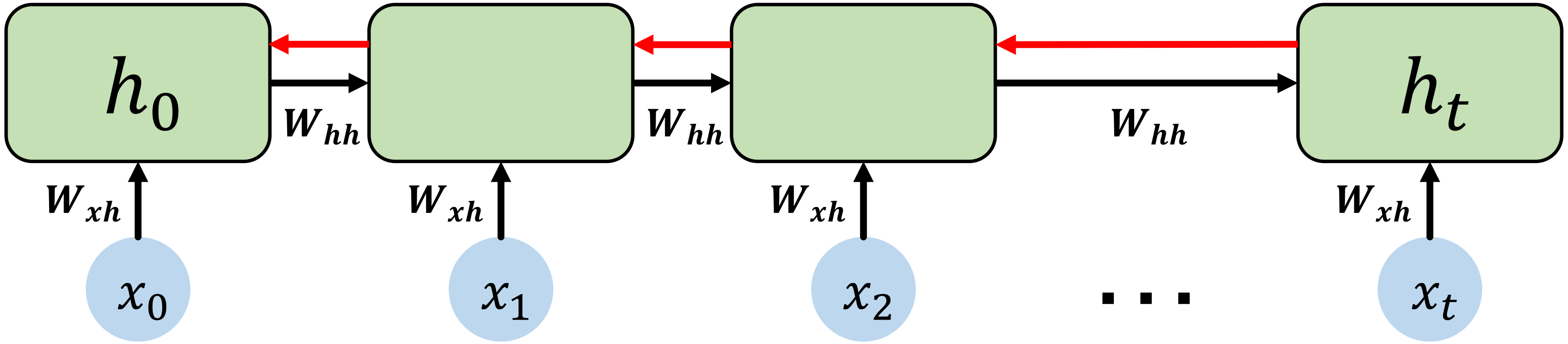
Many values > 1:
exploding gradients

Gradient clipping to
scale big gradients

Many values < 1:
vanishing gradients

[1]

Standard RNN gradient flow: vanishing gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh}** (and repeated f'!)

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

Many values < 1:
vanishing gradients

1. Activation function
2. Weight initialization
3. Network architecture

[1]

The problem of long-term dependencies

Why are vanishing gradients a problem?

The problem of long-term dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

The problem of long-term dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps
have smaller and smaller gradients

The problem of long-term dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias network to capture short-term dependencies

The problem of long-term dependencies

"The clouds are in the ___"

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias network to capture short-term dependencies

The problem of long-term dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

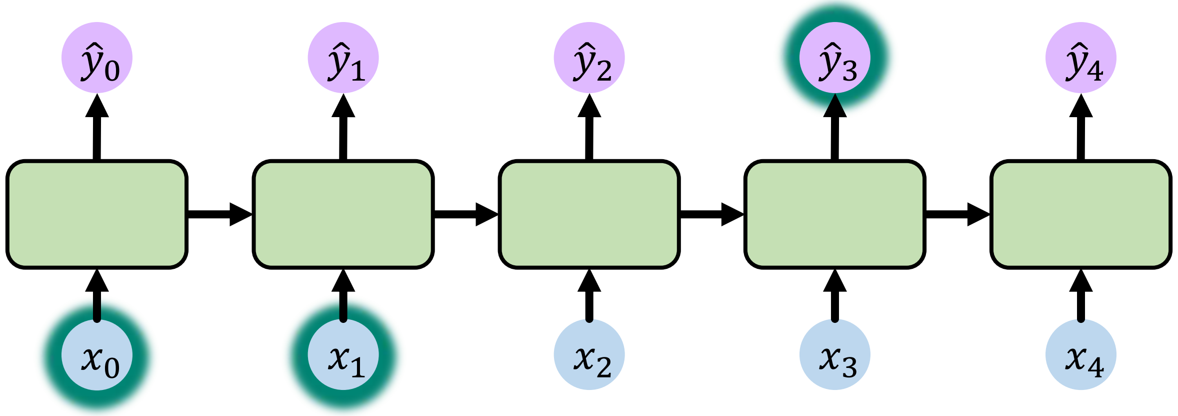
↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"



The problem of long-term dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

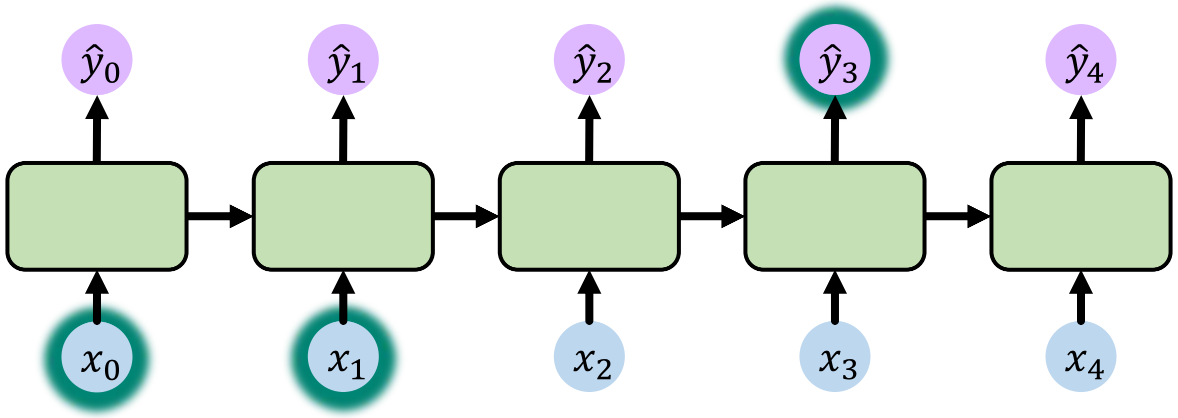
↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"



"I grew up in France, … and I I speak fluent___ "

The problem of long-term dependencies

Why are vanishing gradients a problem?

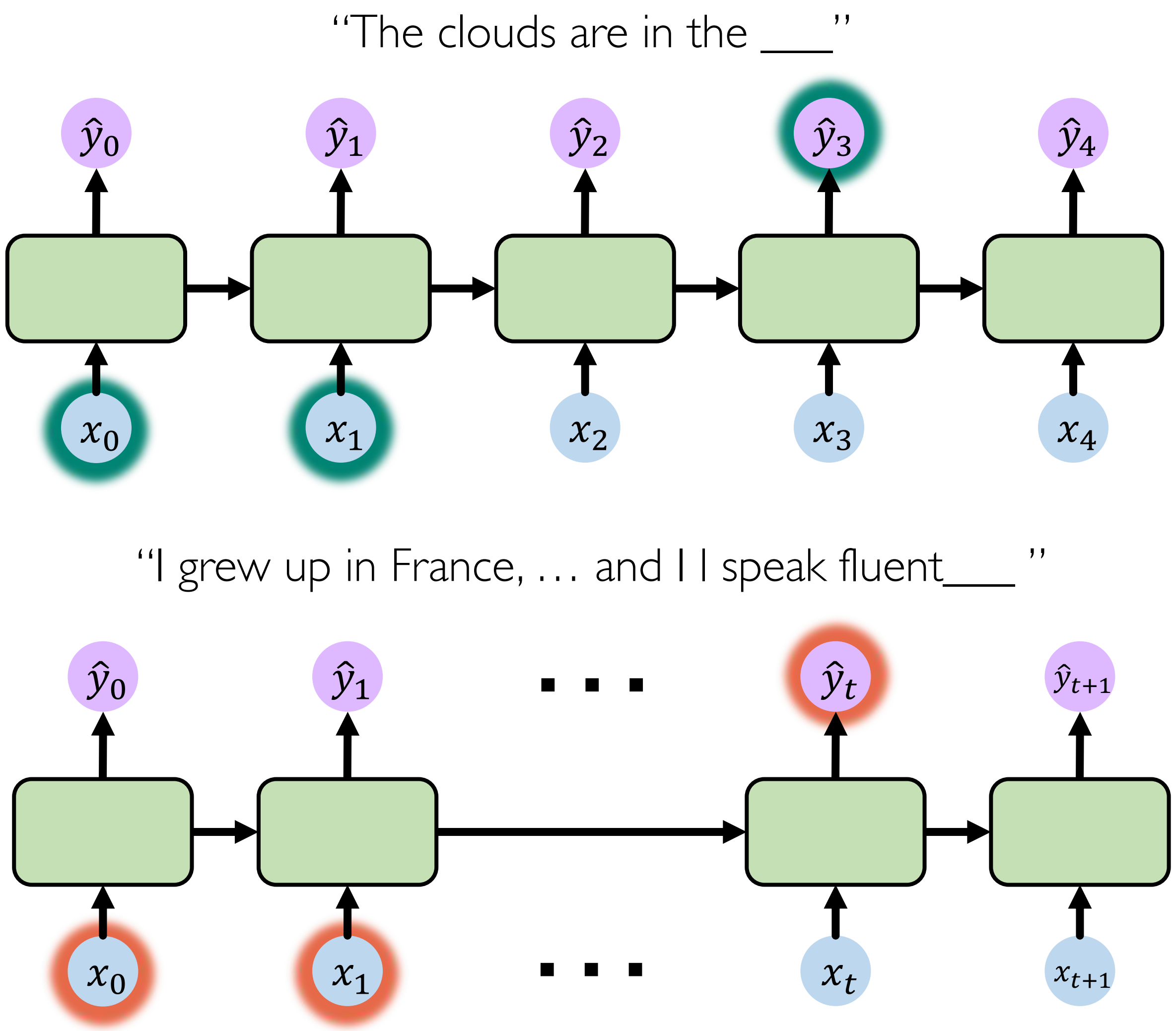
Multiply many **small numbers** together

↓

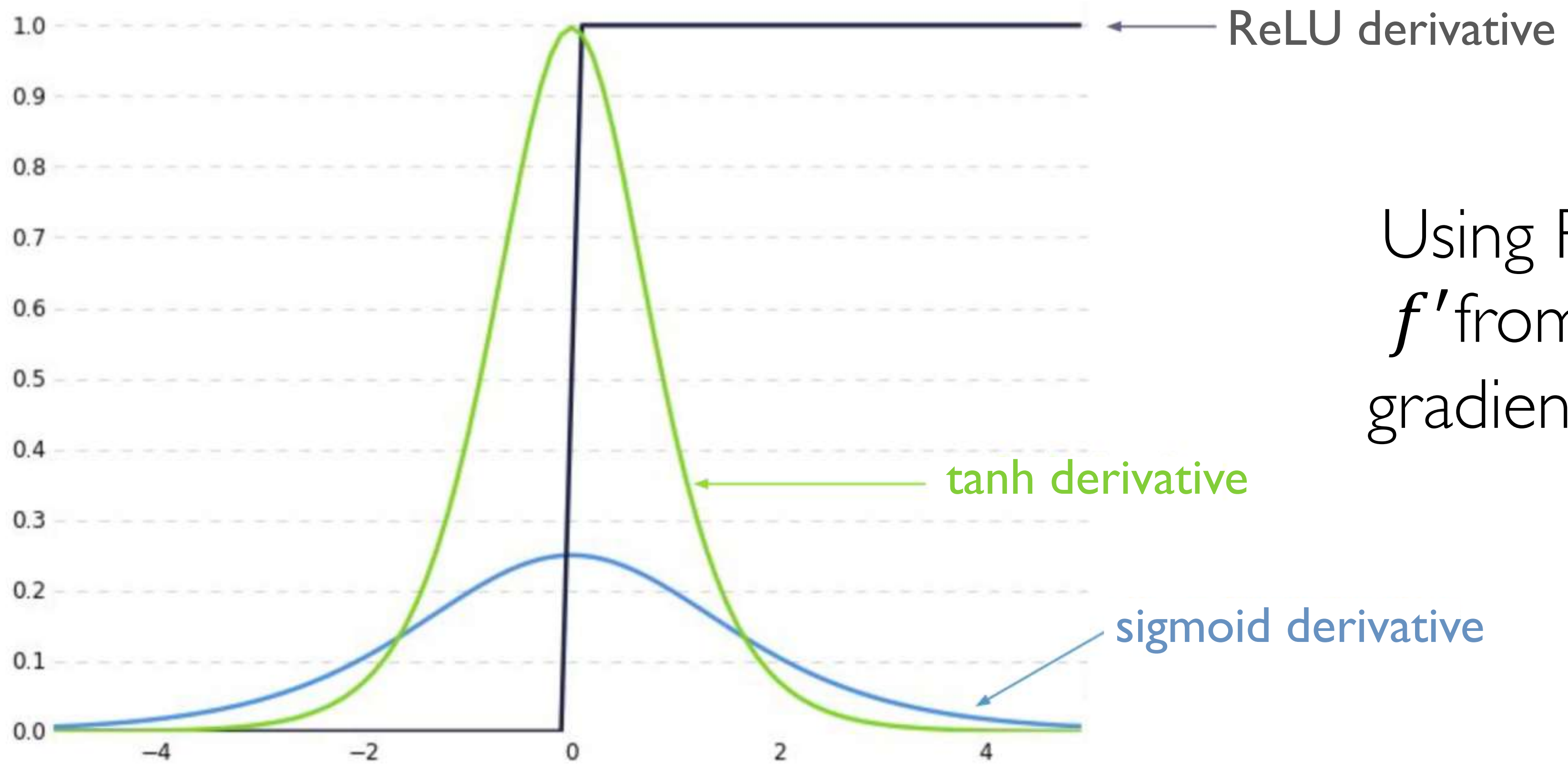
Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies



Trick #1: activation functions



Using ReLU prevents f' from shrinking the gradients when $x > 0$

Adapted from H. Suresh, 6.S191 2018

Trick #2: parameter initialization

Initialize **weights** to identity matrix

Initialize **biases** to zero

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

This helps prevent the weights from shrinking to zero.

Adapted from H. Suresh, 6.S191 2018

Solution #3: gated cells

Idea: use a more **complex recurrent unit with gates** to control what information is passed through

gated cell

LSTM, GRU, etc.

Adapted from H. Suresh, 6.S191 2018

Solution #3: gated cells

Idea: use a more **complex recurrent unit with gates** to control what information is passed through

gated cell

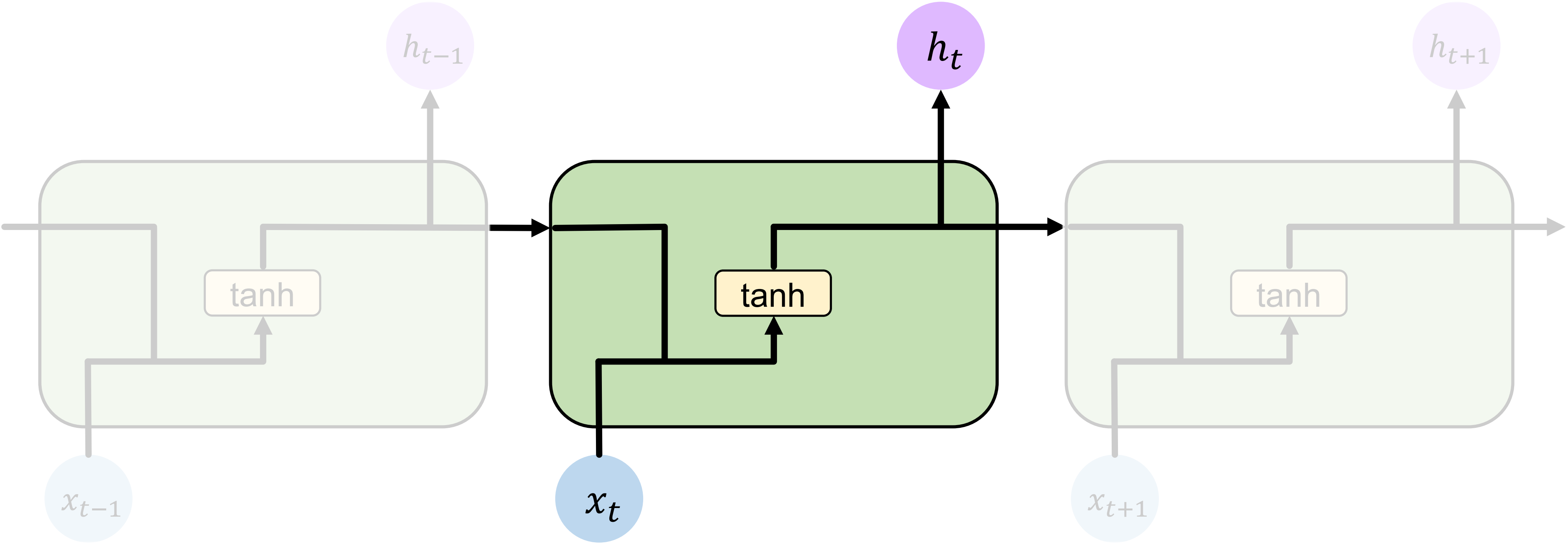
LSTM, GRU, etc.

Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

Adapted from H. Suresh, 6.S191 2018

Standard RNN

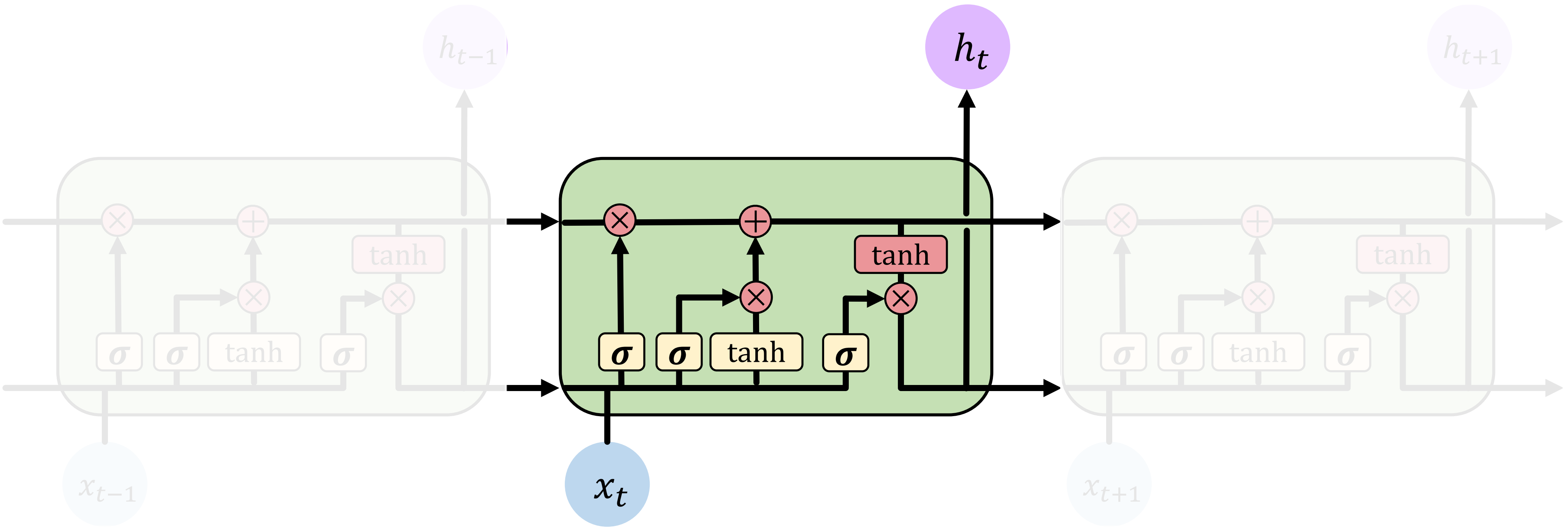
In a standard RNN, repeating modules contain a **simple computation node**



[2]

Long Short Term Memory (LSTMs)

LSTM repeating modules contain **interacting layers** that **control information flow**

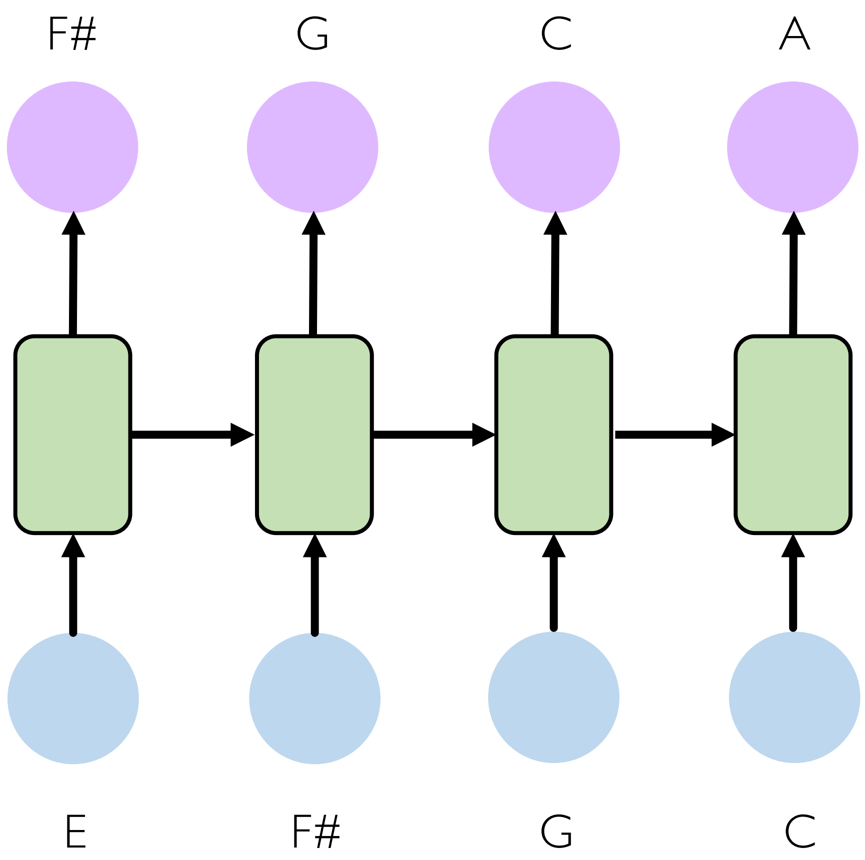


LSTM cells are able to track information throughout many timesteps

Hochreiter & Schmidhuber, 1997 [2, 5]

RNN Applications

Example task: music generation



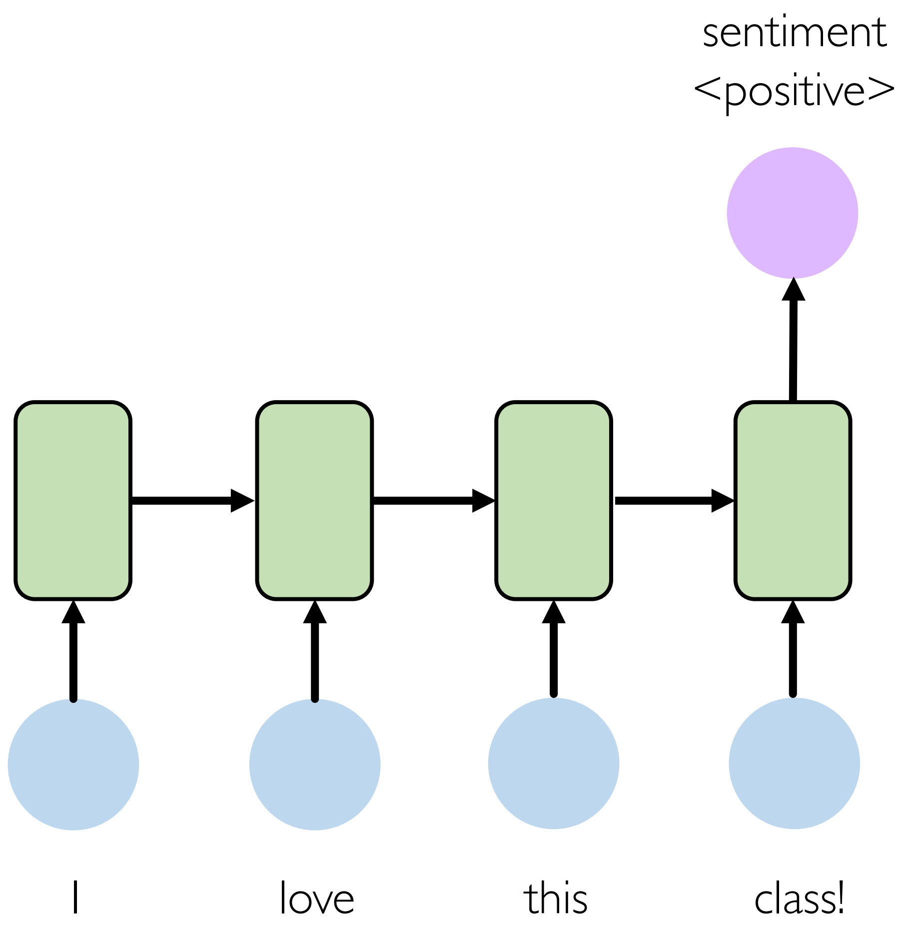
Input: sheet music

Output: next character in sheet music

Adapted from H. Suresh, 6.S191 2018

[6]

Example task: sentiment classification



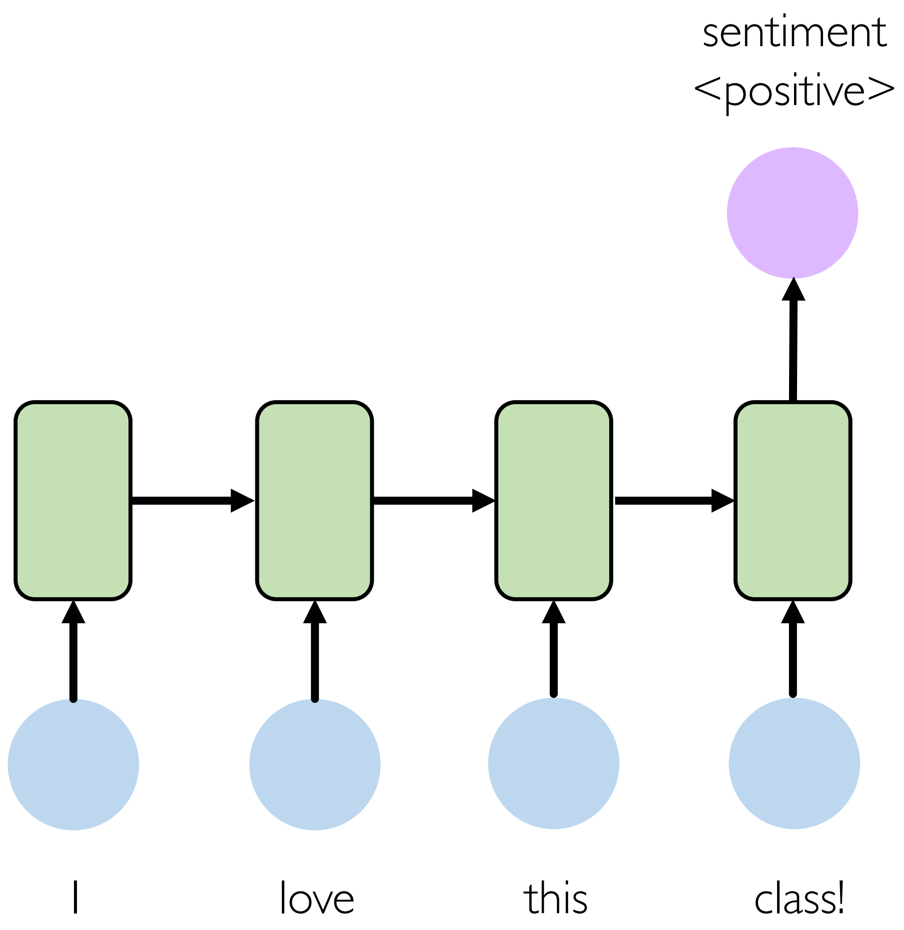
Input: sequence of words

Output: probability of having positive sentiment

Adapted from H. Suresh, 6.S191 2018

[7]

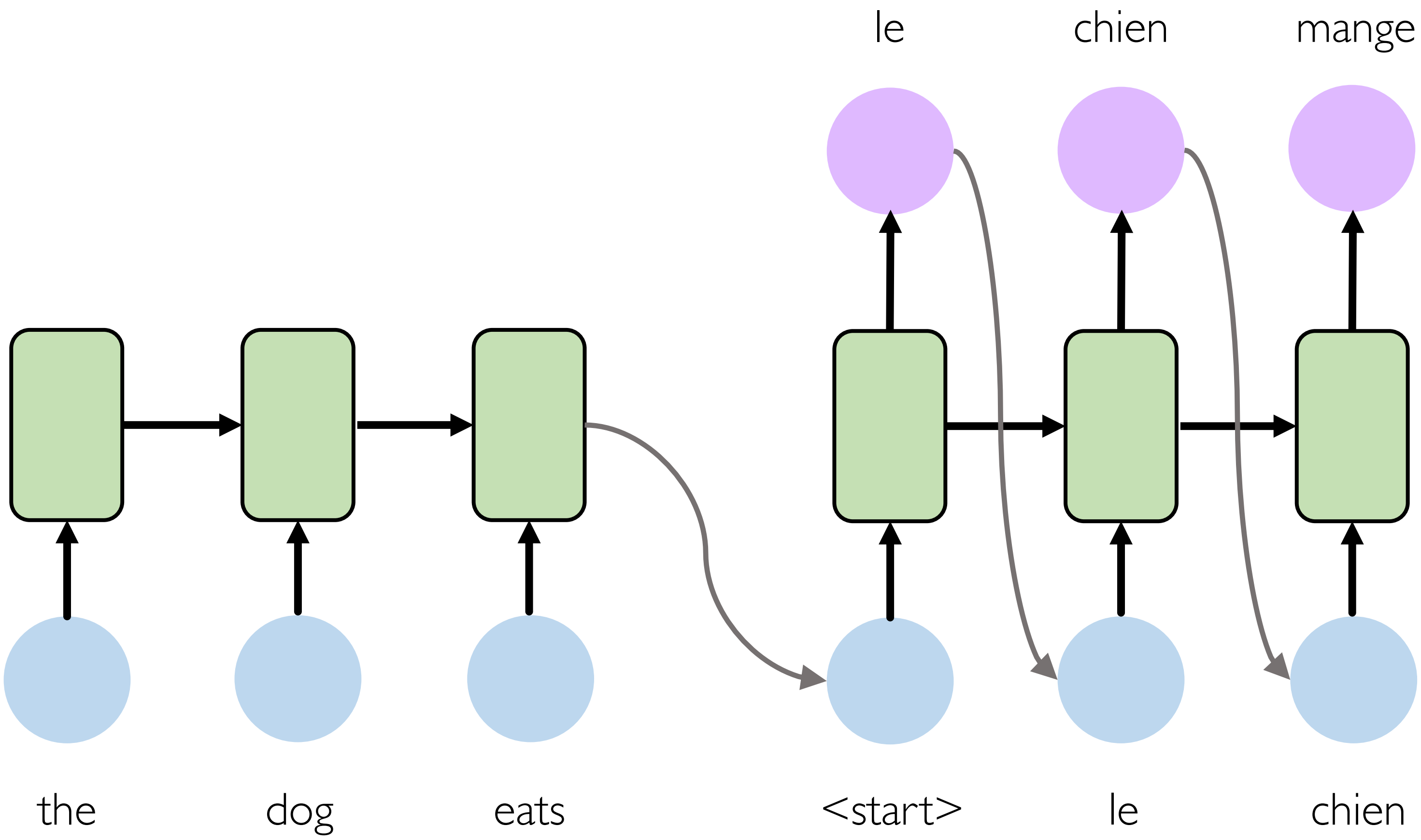
Example task: sentiment classification



Tweet sentiment classification

Adapted from H. Suresh, 6.S191 2018

Example task: machine translation

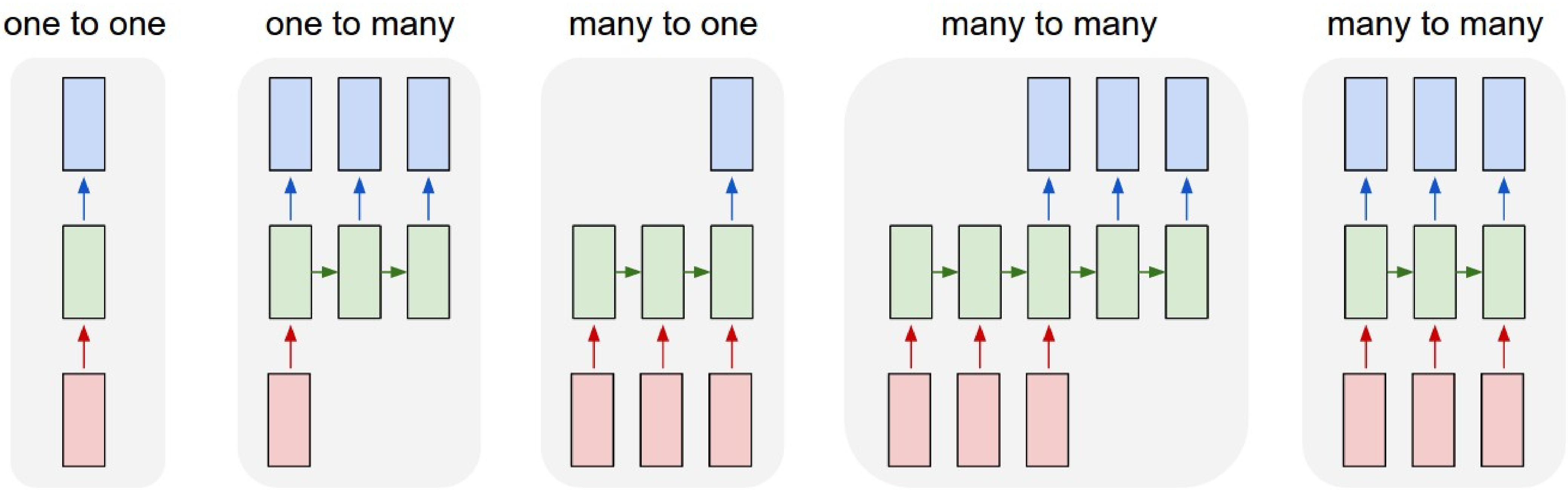


Encoder (English) **Decoder (French)**

Adapted from H. Suresh, 6.S191 2018

[8,9]

Different designs



res: Andrej Karpathy

Different designs

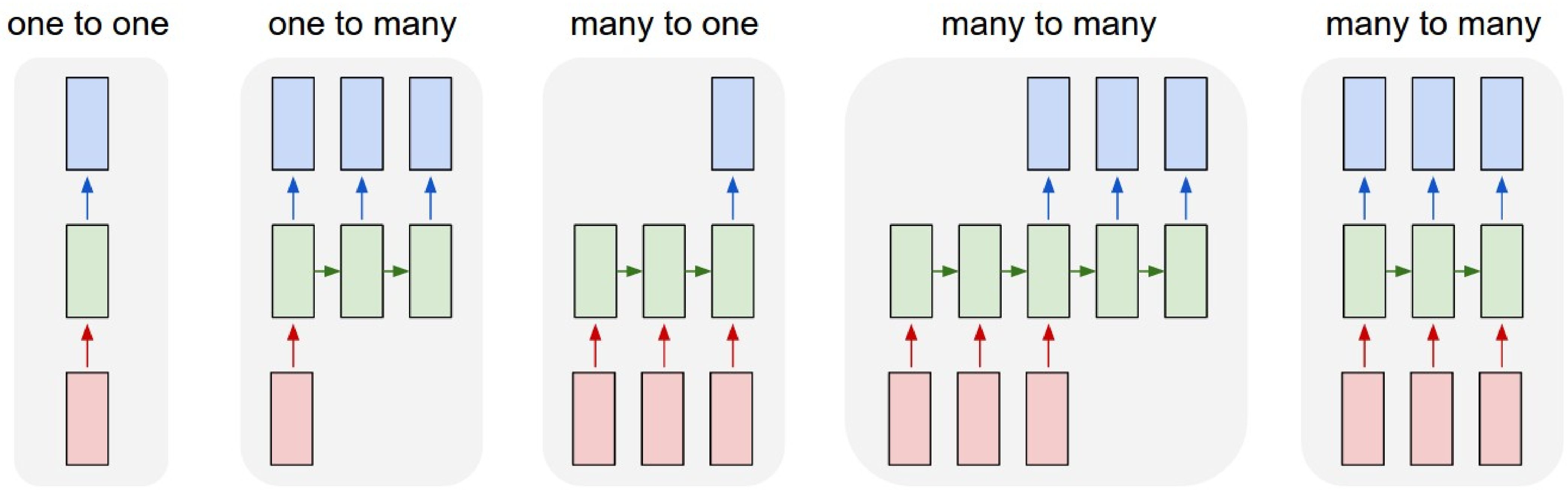


image captioning

sentiment analysis

machine translation

video classification

res: Andrej Karpathy

Recurrent neural networks (RNNs)

1. RNNs are well suited for sequence modeling tasks
2. Model sequences via a recurrence relation
3. Training RNNs with backpropagation through time
4. Models for caption generation, classification, machine translation