Review: RNN

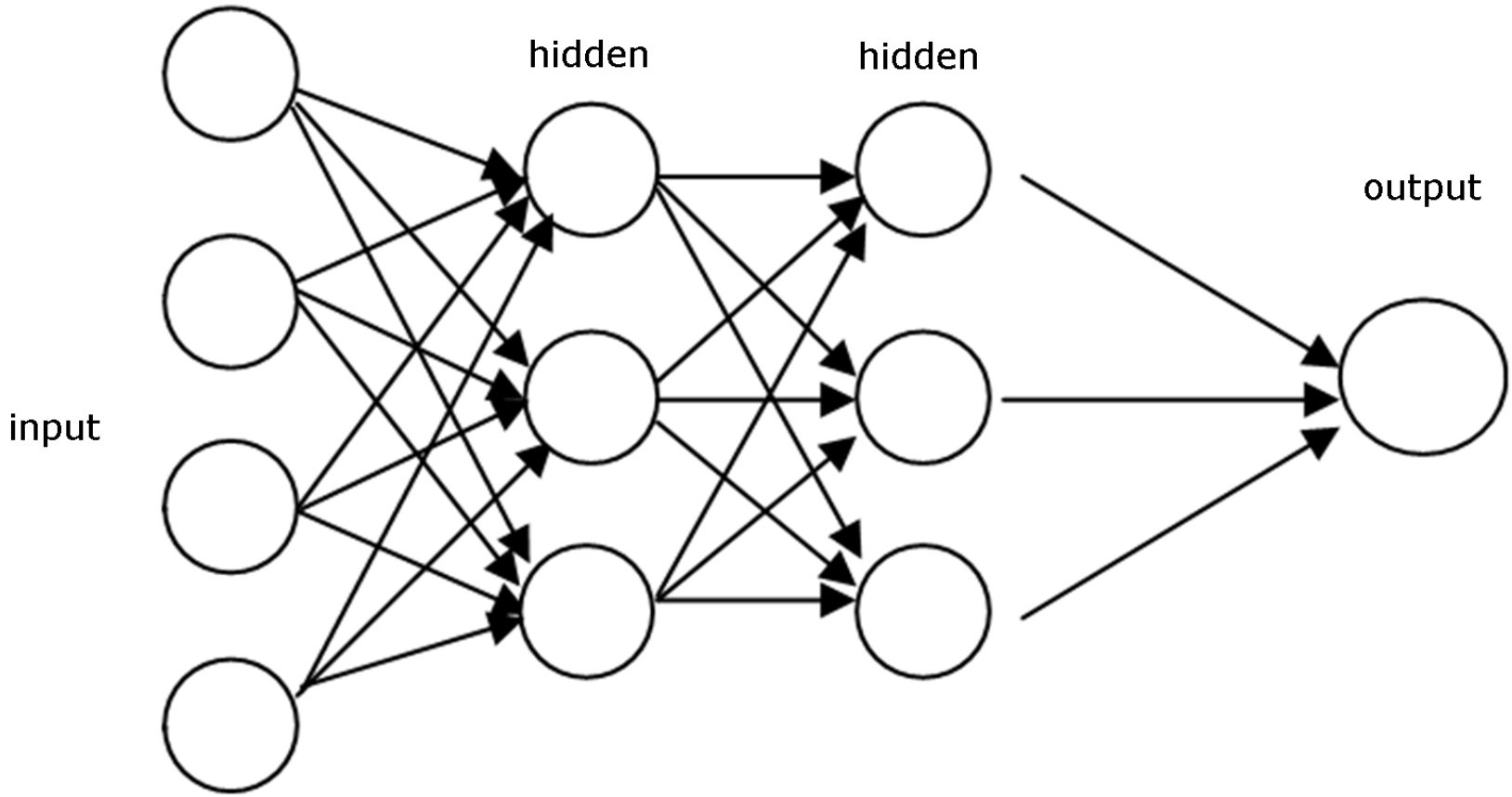
Artificial Intelligence @ Allegheny College

Janyl Jumadinova

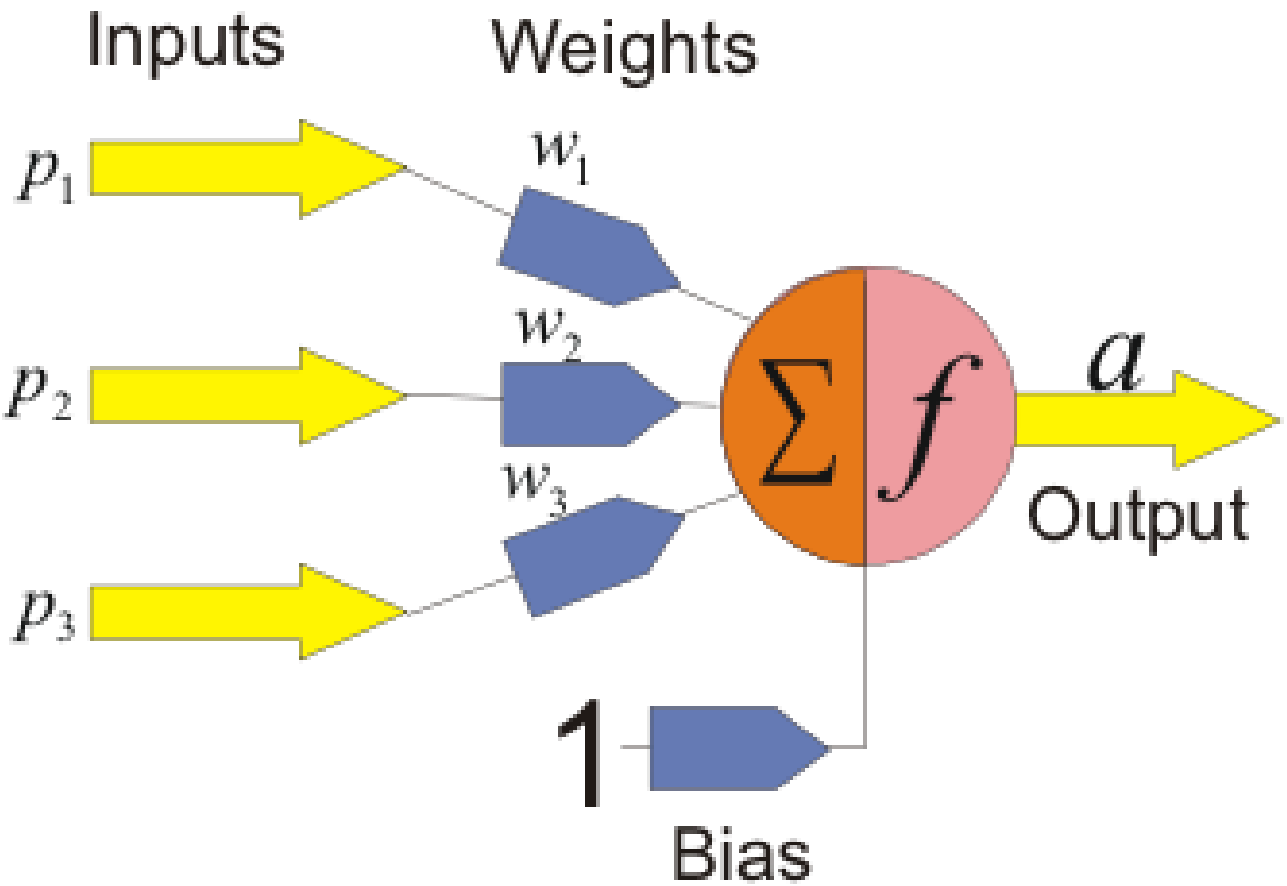
April 7, 2023

Credit: Stanford's RNN Notes

Feed-Forward Neural Network

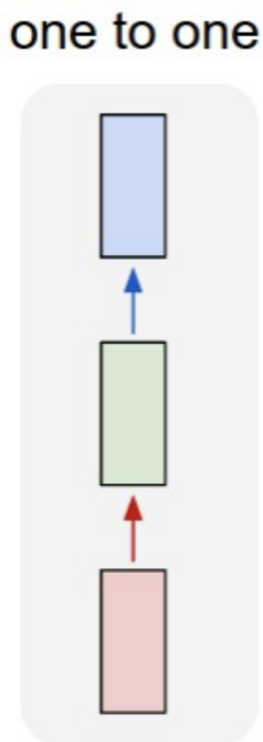


Feed-Forward Neural Network



$$a = f\left(p_1 w_1 + p_2 w_2 + p_3 w_3 + b\right) = f\left(\sum p_i w_i + b\right)$$

Feed-Forward Neural Network



e.g. **Image classification**
Image -> Label

Sequence-to-Sequence Modeling

	Input	Output
Machine Translation	"The cat is black"	["Le", "chat", "est", "noir"] or "Le chat est noir"
Named Entity Recognition	"Sonja works at Apple"	[<PERSON>, <NA>, <NA>, <ORG>]
POS Tagging	"Sonja works at Apple"	[Noun, Verb, Preposition, Noun]
Speech Recognition		"I am the batman"
Video Captioning		"[fire crackles] Not today."
DNA Analysis	"cacgcaaacccttttcaggcttcggtcgtcct caagctgtccgatgcgcagactactttgtcg"	{promoter:[0,67], exon:[74,103]}

Other Sequence Data

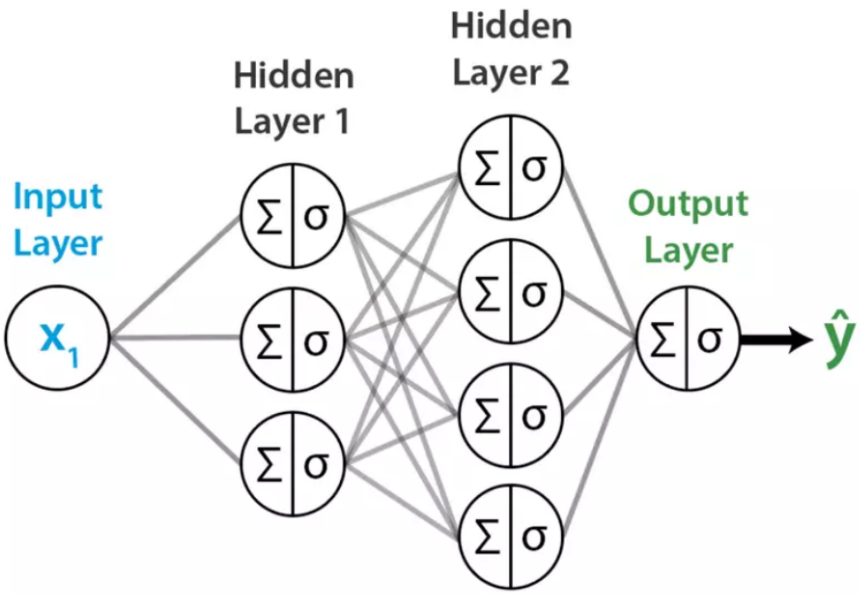
- **Website activity**: [button click, cursor movement, click subscribe, close]

Other Sequence Data

- **Website activity**: [button click, cursor movement, click subscribe, close]
- **User history**: [inactive, open account, close account, request sent, payment made]
- **Code**: constrained language, can learn the structure

Why Feed-Forward is not enough

- *Example*: language translation (English to French)
- Could use feed-forward to translate word by word



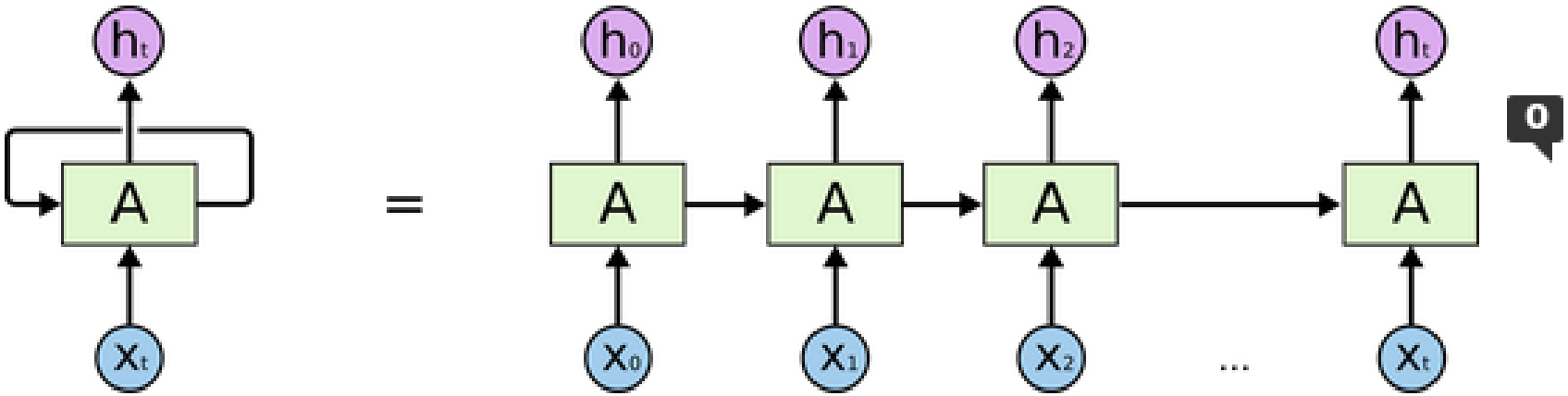
Could build the training set:

Input	Output
The	Le
cat	chat
is	est
black	noir

Why Feed-Forward is not enough!

- Feed-forward treats time steps as completely independent
- **Problem**: correct translation requires context
- **Solution**: need a way to remember information from previous steps

Recurrent Neural Networks

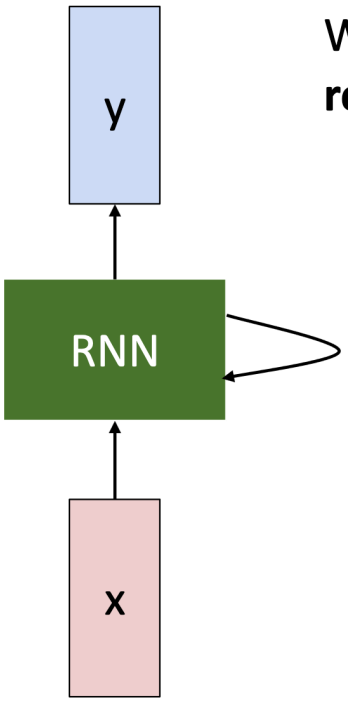


An unrolled recurrent neural network.

Popular way of modeling sequential data.

Key Idea: Process sequence data one step at a time while updating a running internal hidden state.

Recurrent Neural Networks



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

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

new state

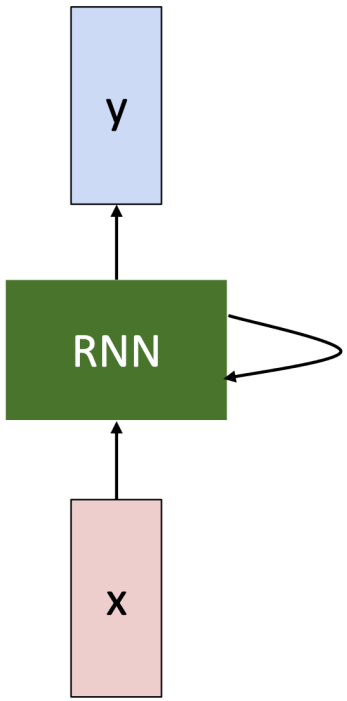
some function with parameters W

old state

input vector at some time step

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

Recurrent Neural Networks



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

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

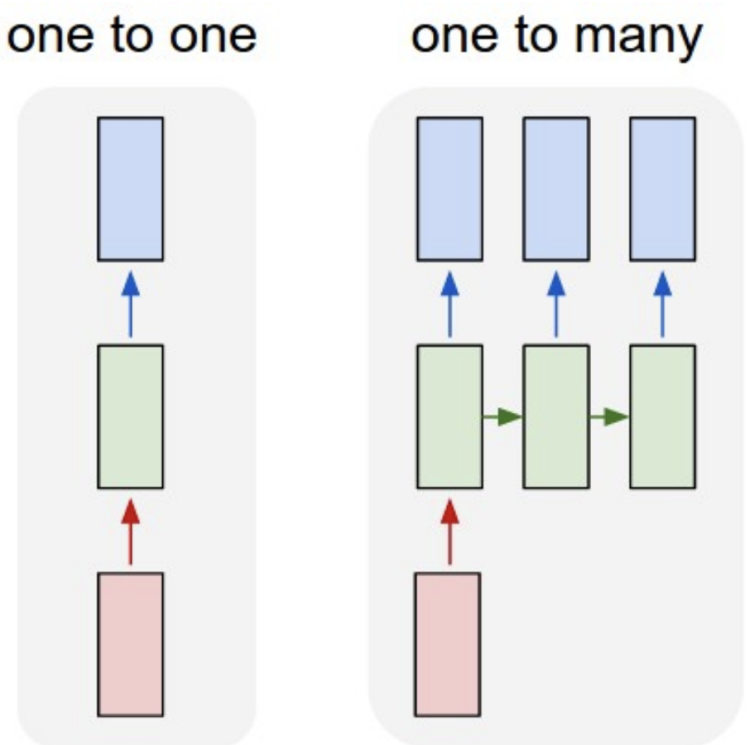
(also bias term)

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

$$y_t = W_{hy}h_t$$

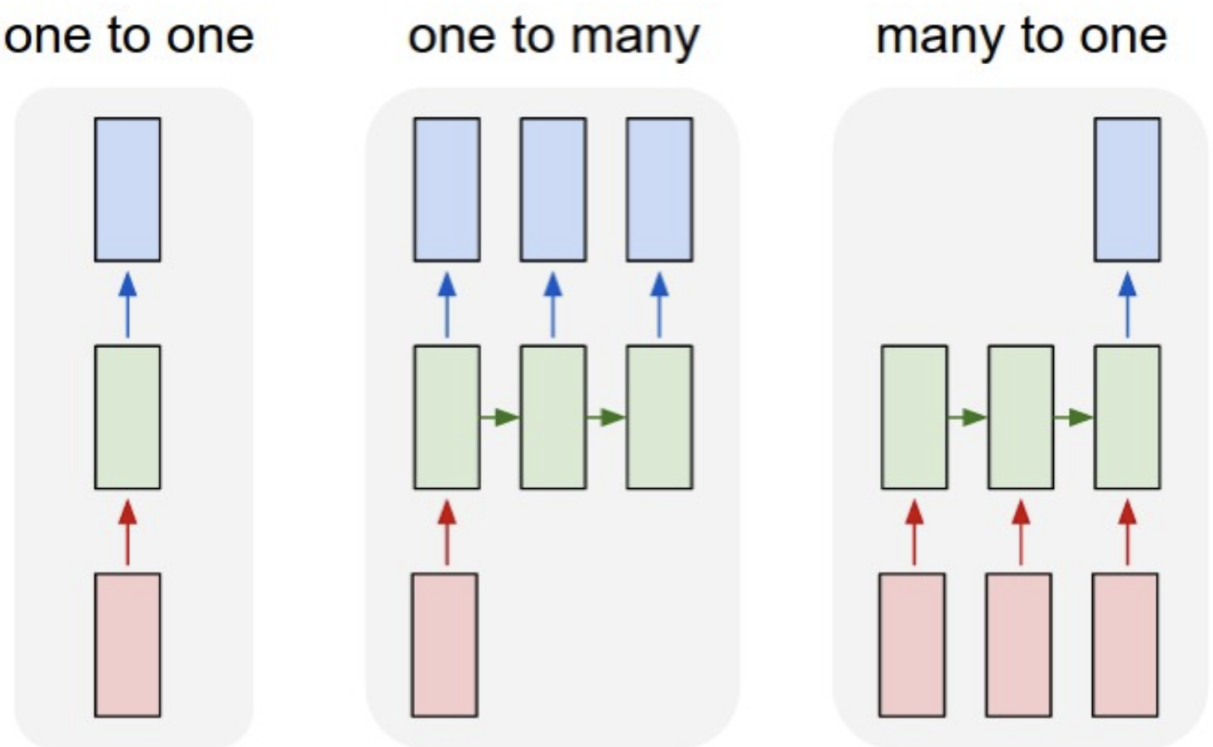
Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

Recurrent Neural Network



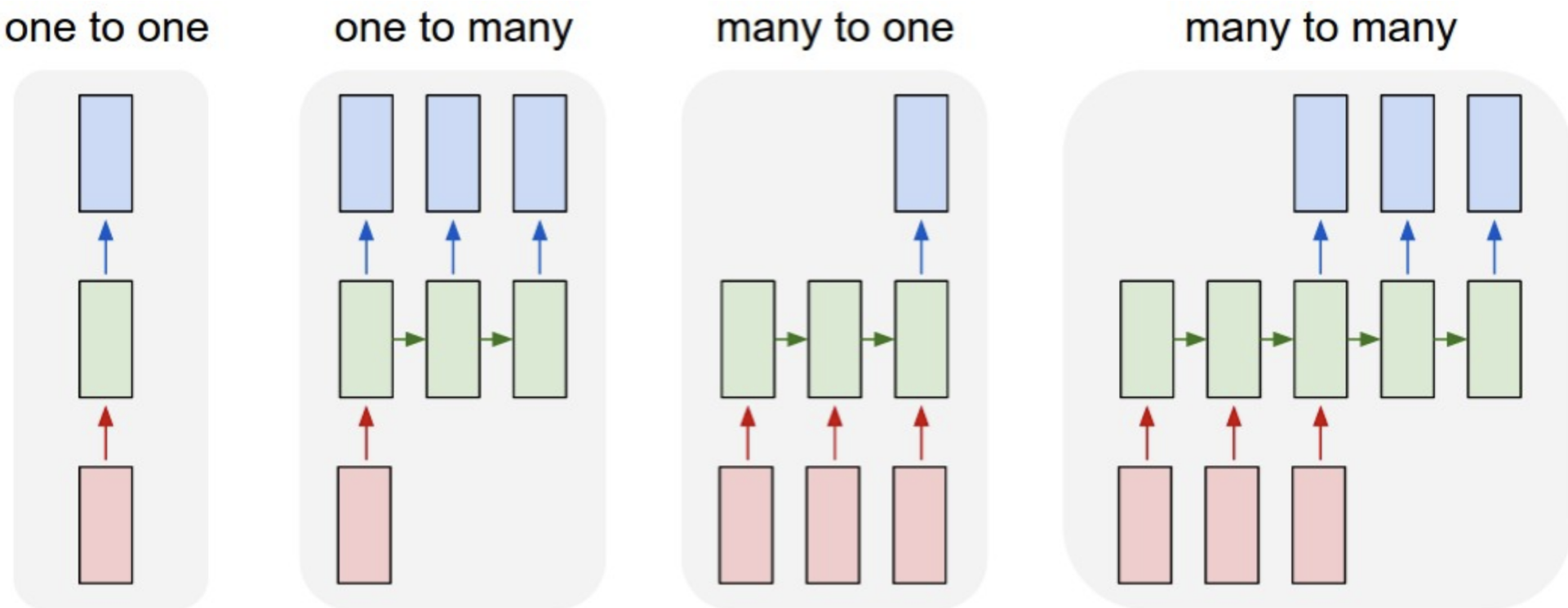
e.g. **Image Captioning**:
Image -> sequence of words

Recurrent Neural Network



e.g. **Video classification:**
Sequence of images -> label

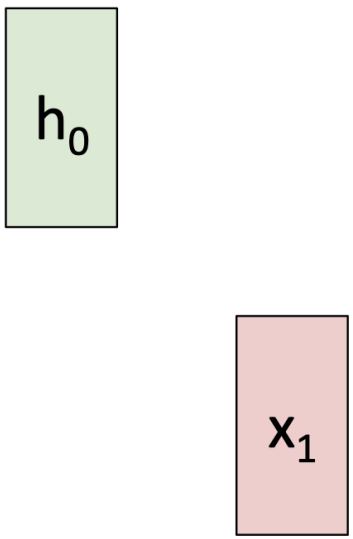
Recurrent Neural Network



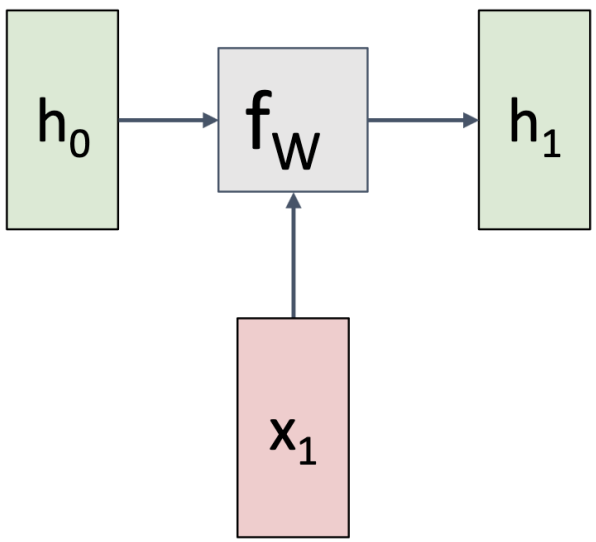
e.g. **Machine Translation:**
Sequence of words -> Sequence of words

RNN Computational Graph

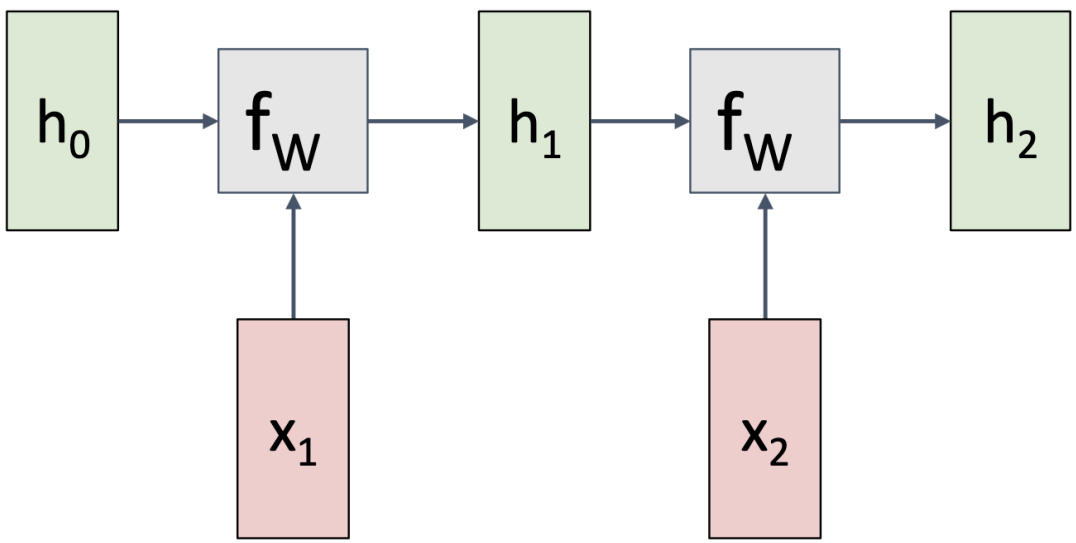
Initial hidden state
Either set to all 0,
Or learn it



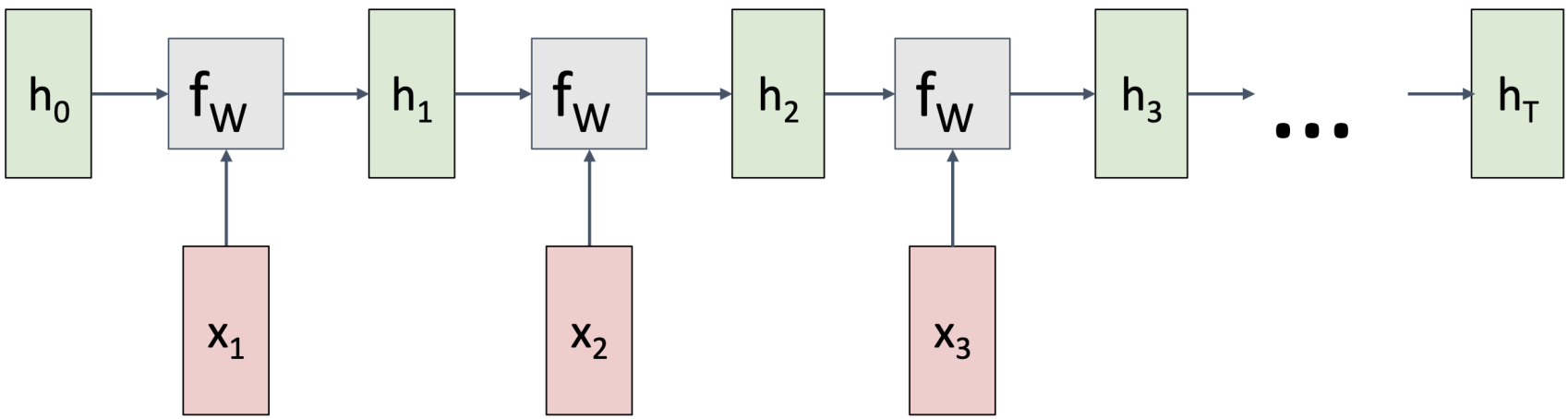
RNN Computational Graph



RNN Computational Graph

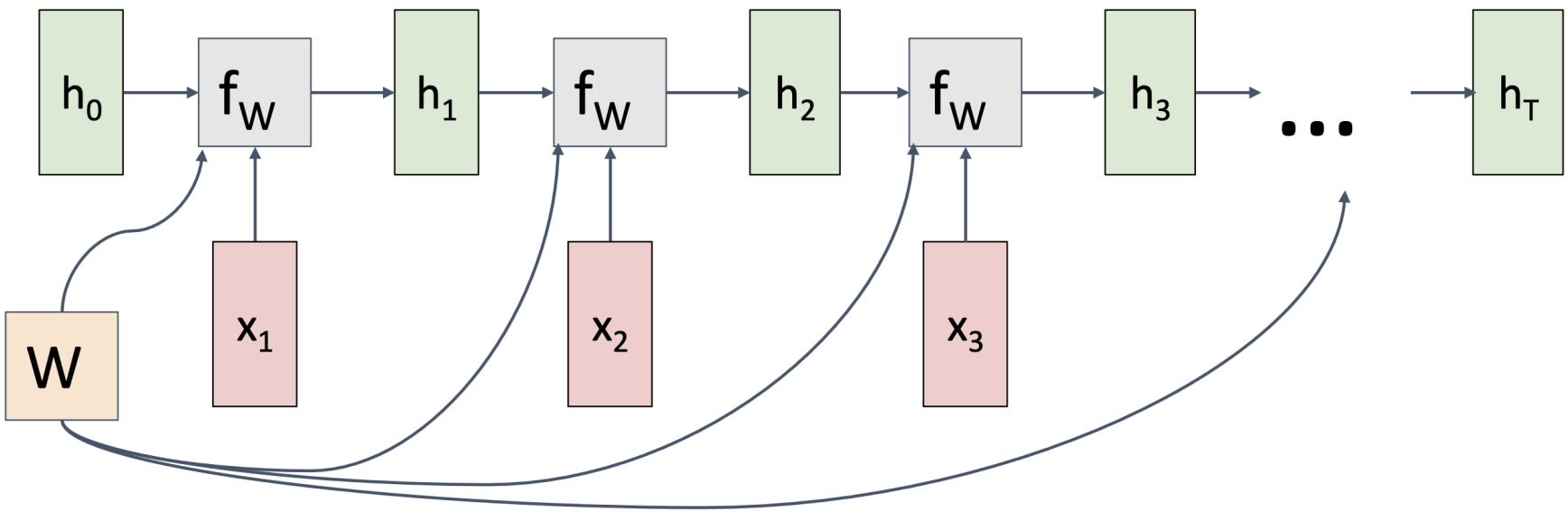


RNN Computational Graph

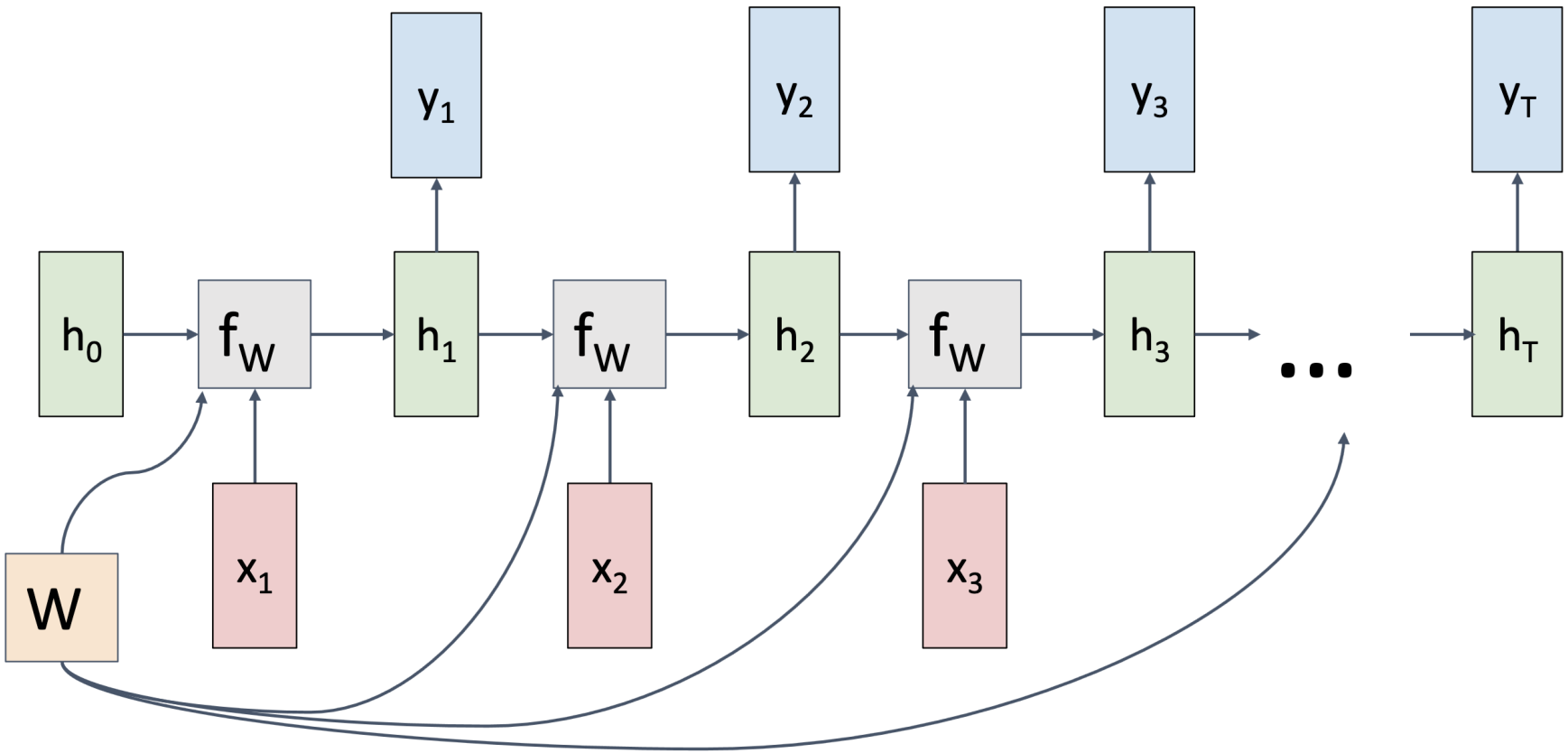


RNN Computational Graph

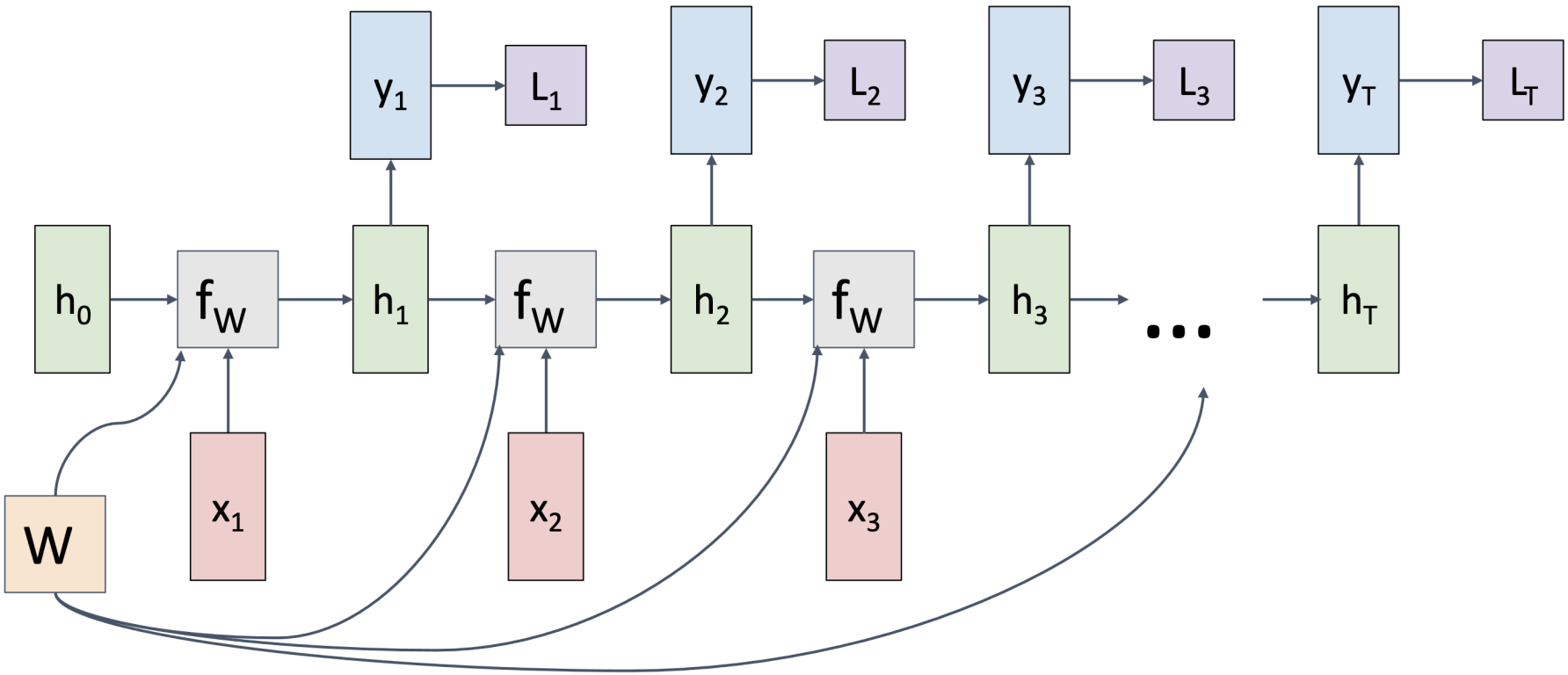
Re-use the same weight matrix at every time-step



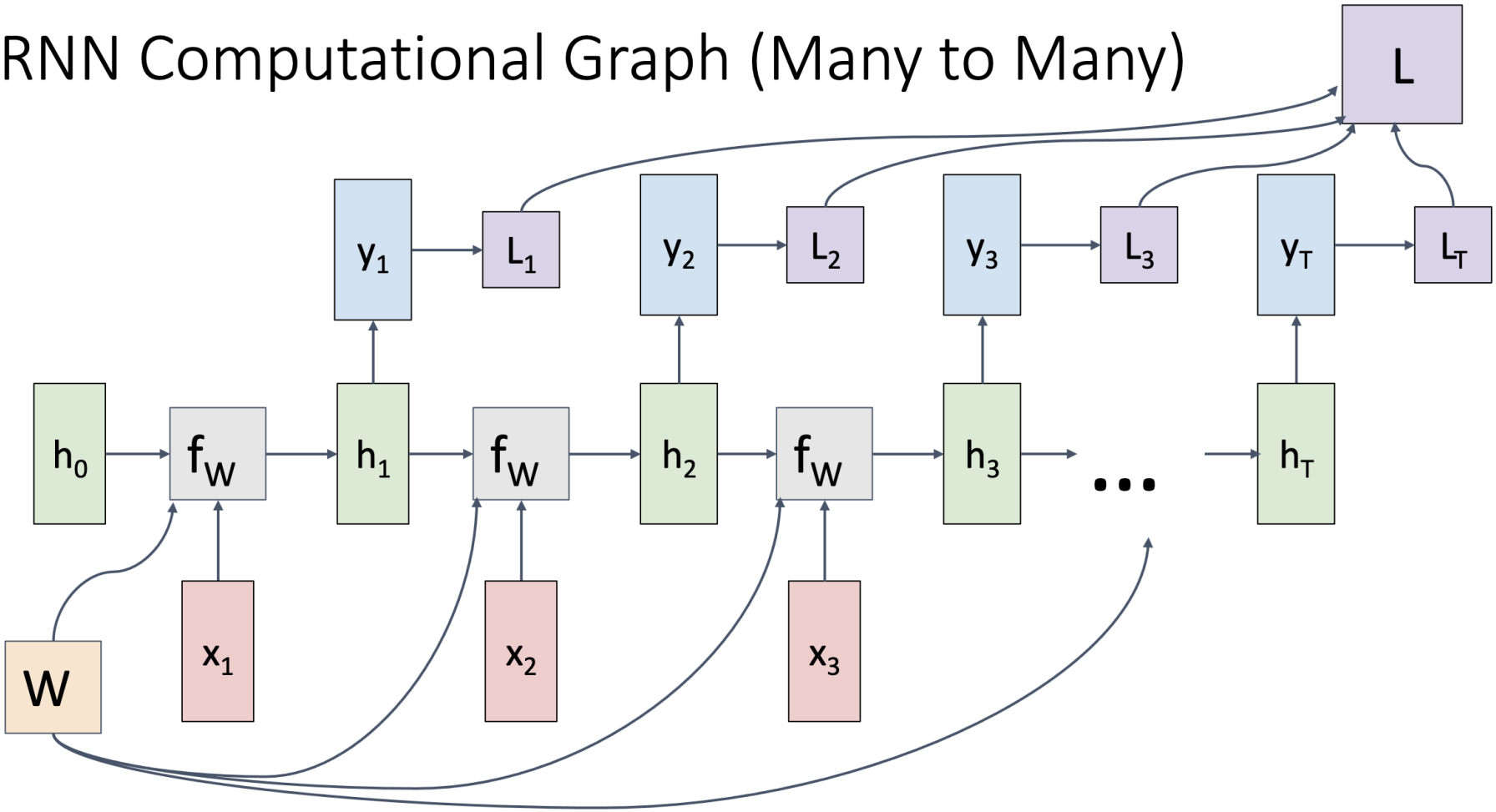
RNN Computational Graph (Many to Many)



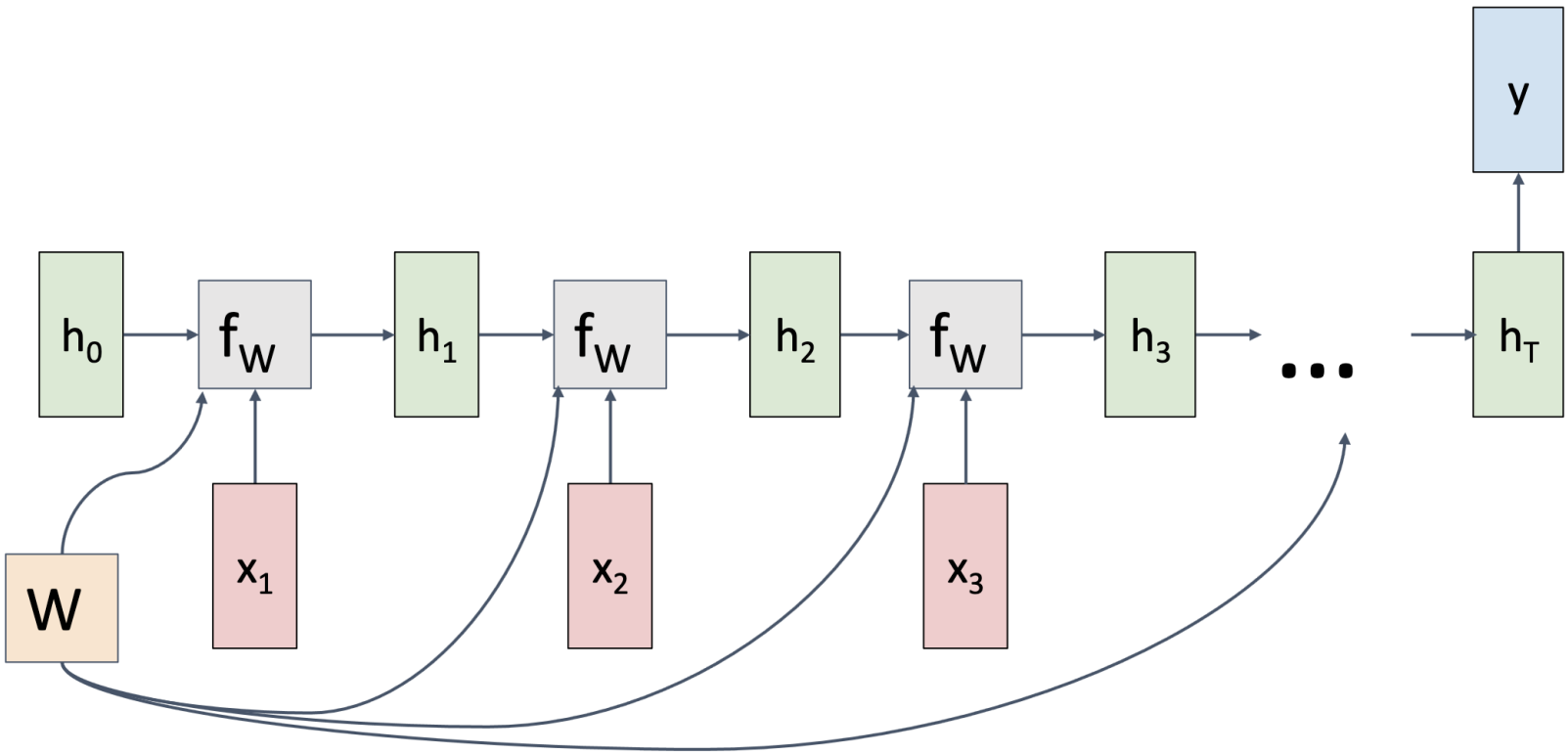
RNN Computational Graph (Many to Many)



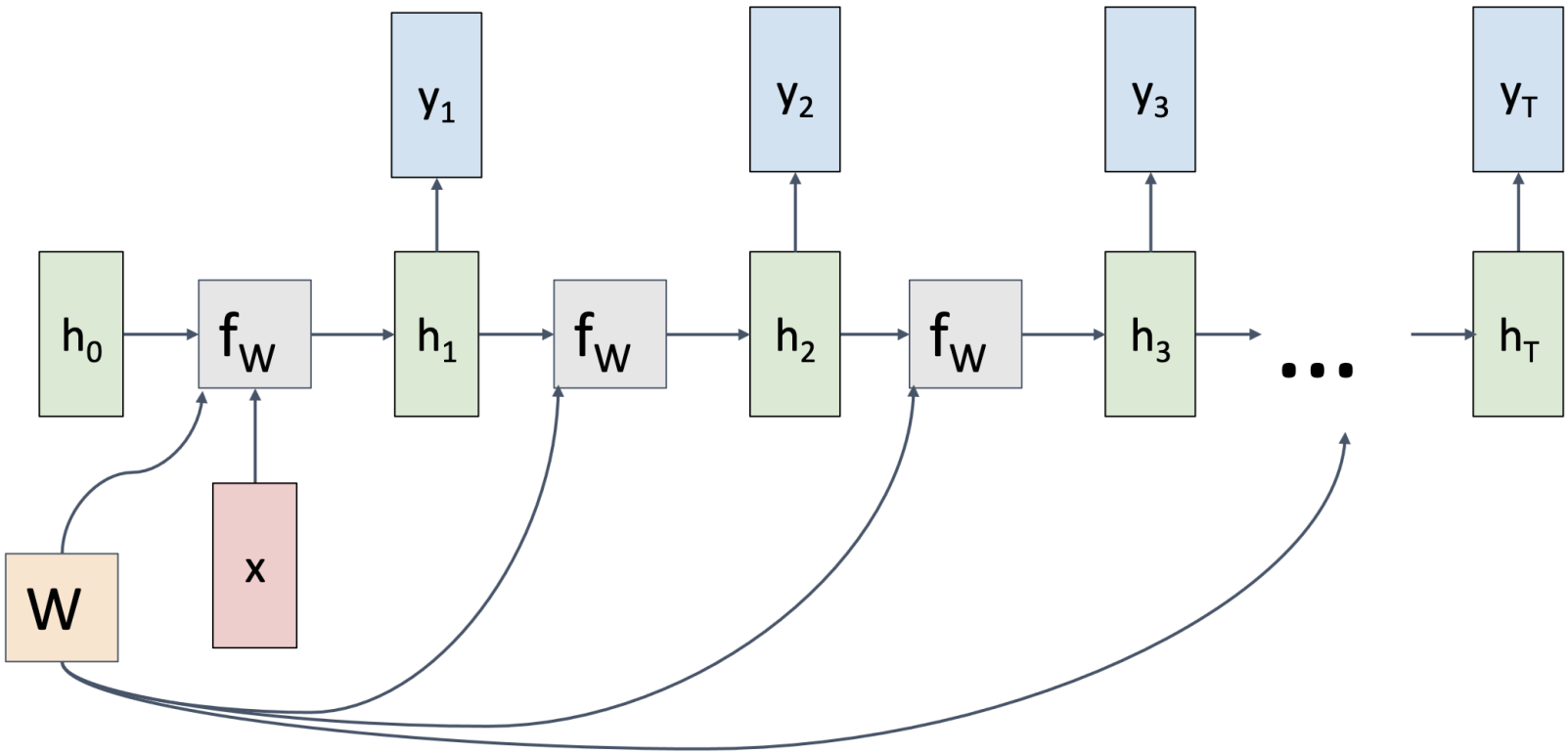
RNN Computational Graph (Many to Many)



RNN Computational Graph (Many to One)



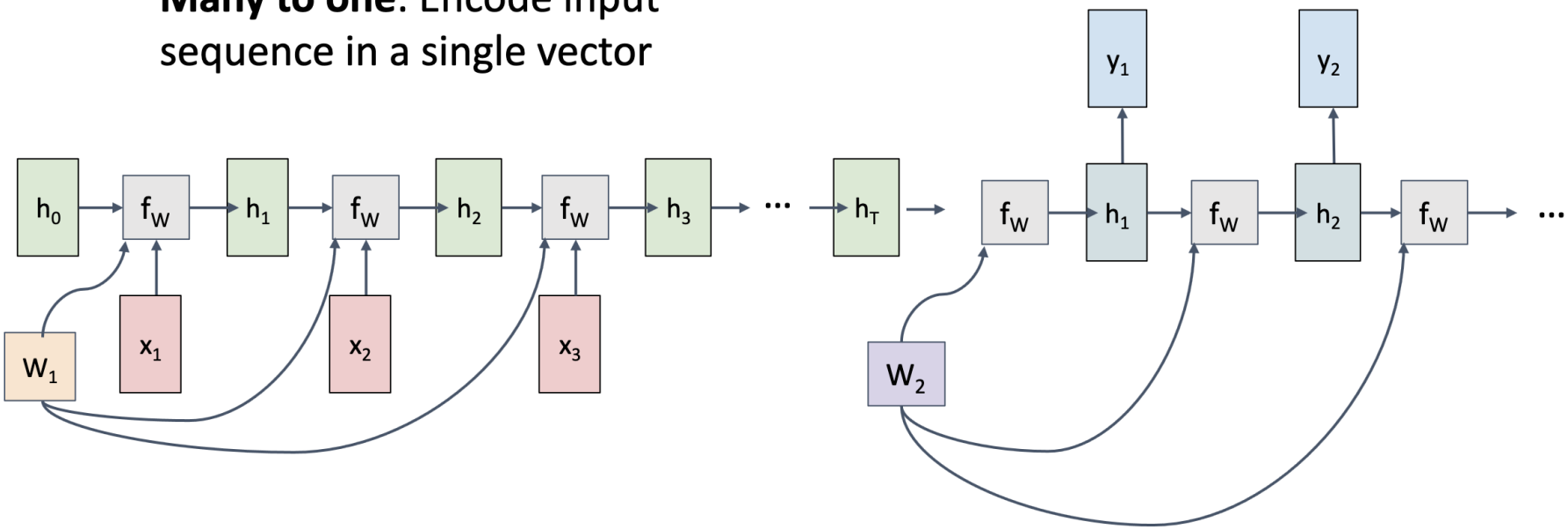
RNN Computational Graph (One to Many)



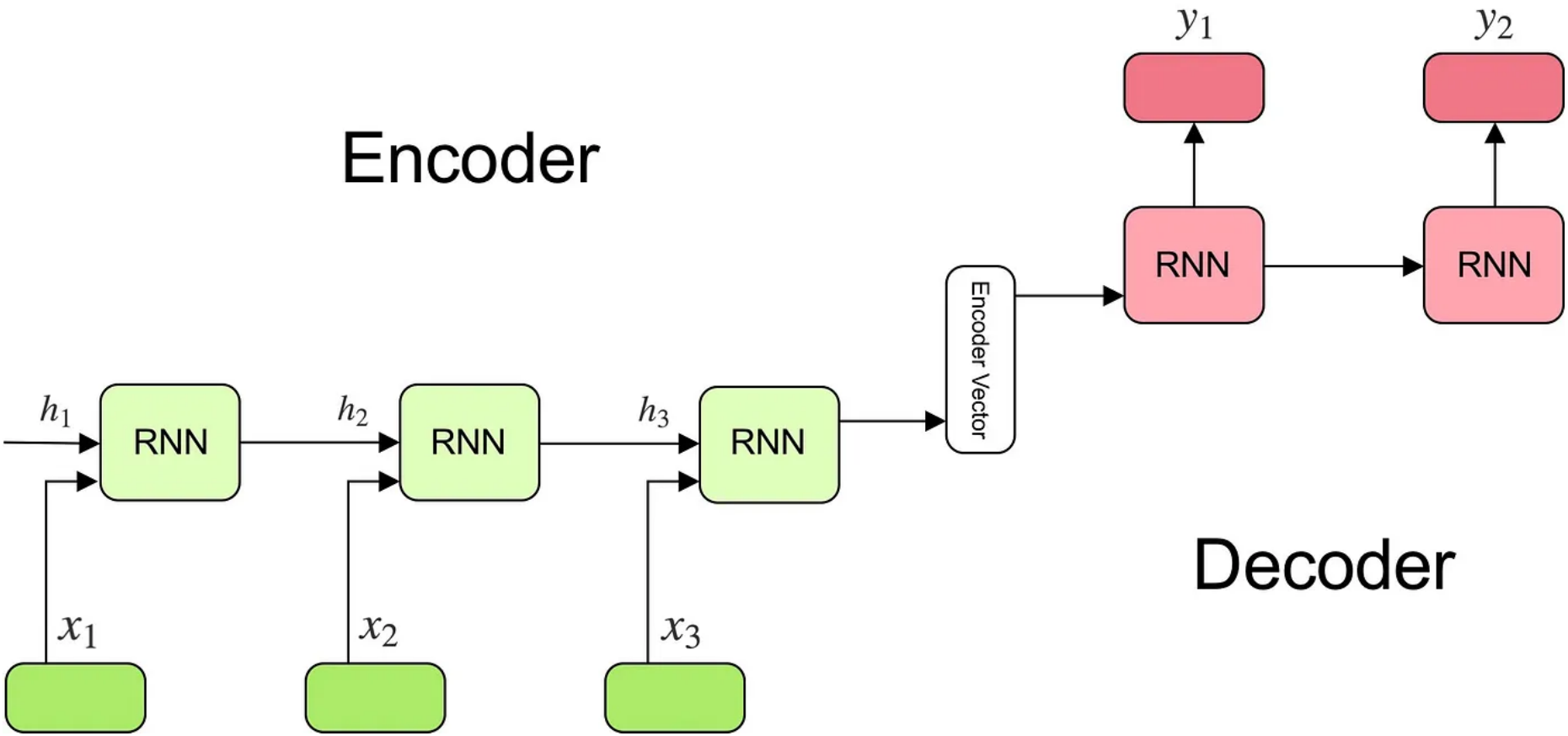
Sequence to Sequence (seq2seq)
(Many to one) + (One to many)

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector



Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014


Encoder
Decoder
RNN
RNN
RNN
RNN
RNN
Encoder Vector
h_1
h_2
h_3
x_1
x_2
x_3
y_1
y_2

Recurrent Neural Networks

- **RNNs problem**: ineffective when the relevant information is further away.
- Due to chain-like structure, as information is passed at each step, the longer the chain is, the more probable the information is lost along the chain.
- LSTMs try to solve this problem by learning this long-term dependencies.

Long Short-Term Memory (LSTM)

Vanilla RNN

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

LSTM

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

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

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

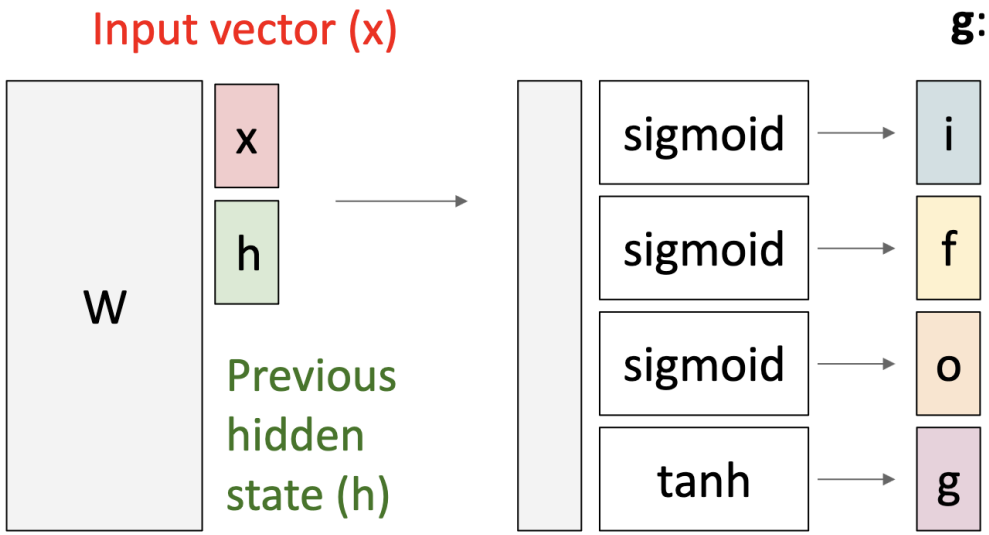
Two vectors at each timestep:

Cell state

Hidden state

Long Short-Term Memory (LSTM)

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



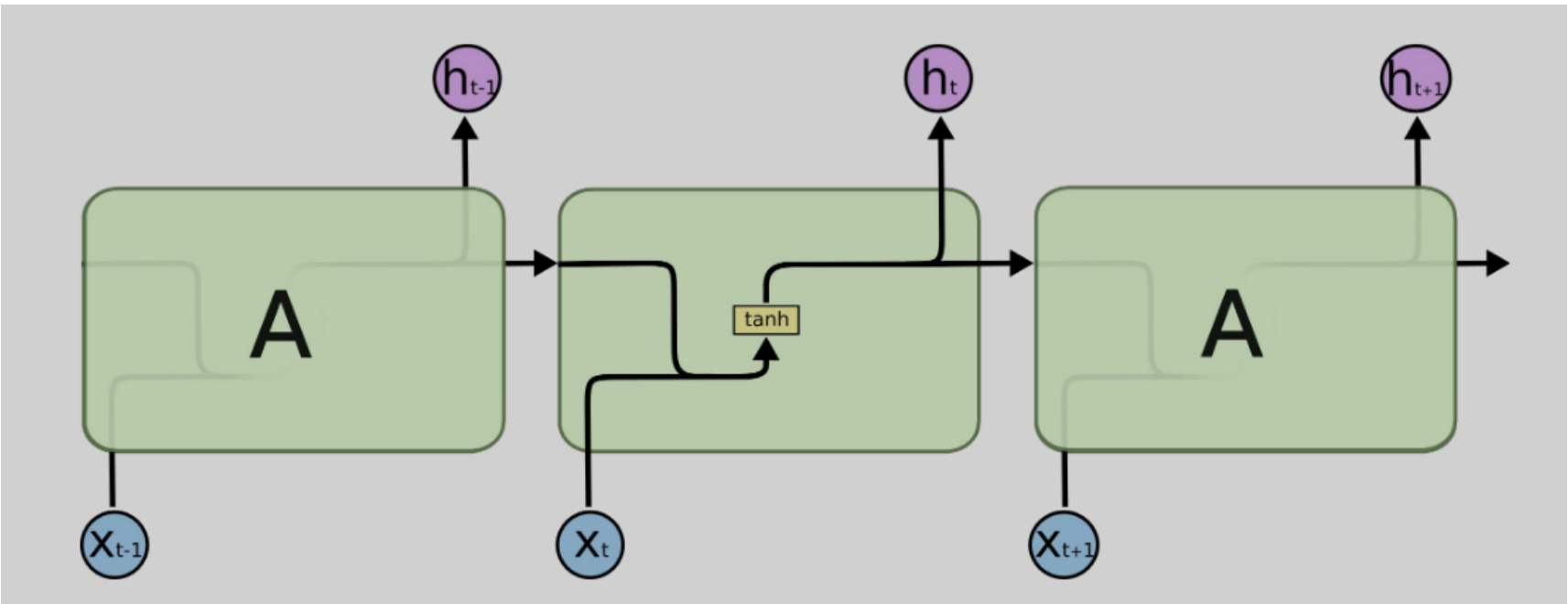
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

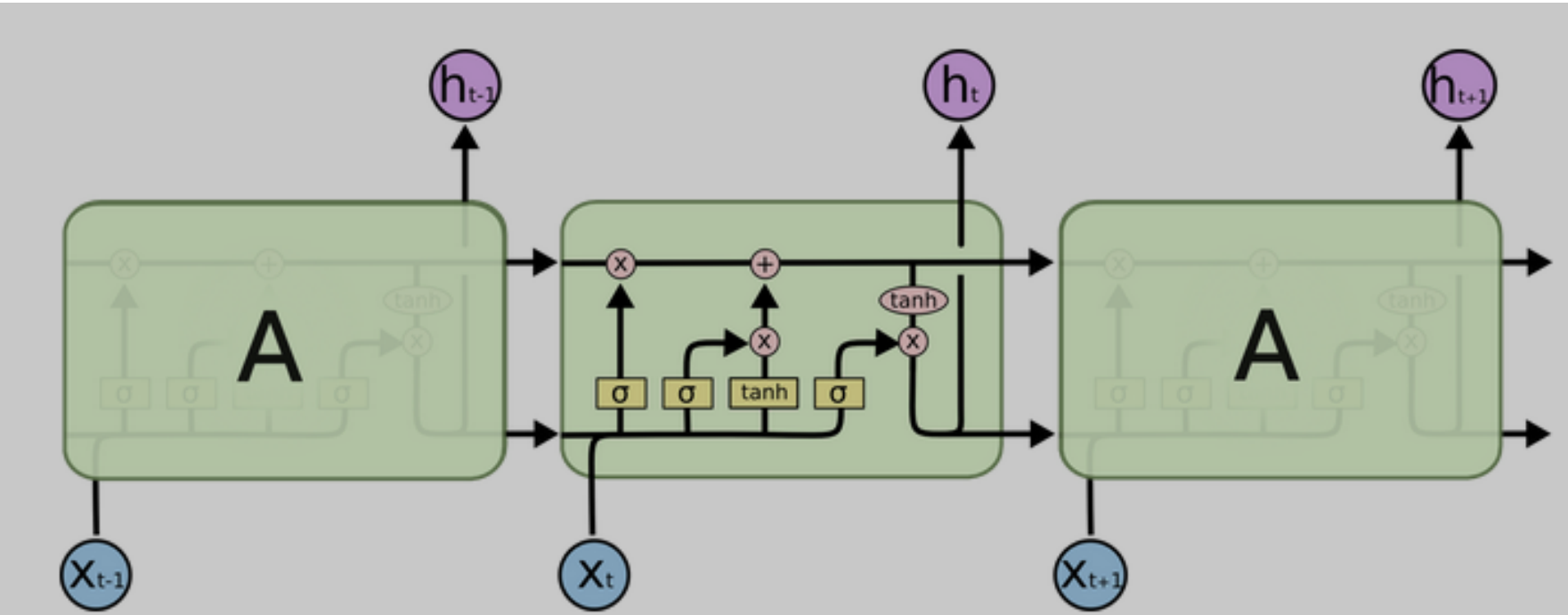
Long Short-Term Memory (LSTM)

Based on a standard RNN whose neuron activates with tanh



Cristopher Olah, "Understanding LSTM Networks" (2015)

Long Short-Term Memory (LSTM)



The repeating module in an LSTM contains four interacting layers.

Attention

Disadvantage of RNN/LSTM

- Suffer from memory-bandwidth limited problems.
- LSTMs still suffer from long range dependencies.
- Alternative? Transformer architecture (replace recurrence/convolution with attention).
- **Attention**: try to focus on a few relevant things, while ignoring others in deep neural networks.

`https://machinelearningmastery.com/the-attention-mechanism-from-scratch/`