CPEN 455: Deep Learning

Lecture 6: Recurrent Neural Networks

Qi Yan, Muchen Li, Jiahe Liu, Qihang Zhang, Renjie Liao

University of British Columbia

Winter, Term 2, 2024

Outline

- Recurrent Neural Networks (RNNs)
  - Motivations
  - RNN basics
  - Long Short Term Memory (LSTM)
  - Back Propagation Through Time (BPTT)

Outline

- Recurrent Neural Networks (RNNs)
    - **Motivations**
    - RNN basics
    - Long Short Term Memory (LSTM)
    - Back Propagation Through Time (BPTT)

Sequential Data

- Text generation
- Stock price prediction
- Weather prediction
- Audio Signal

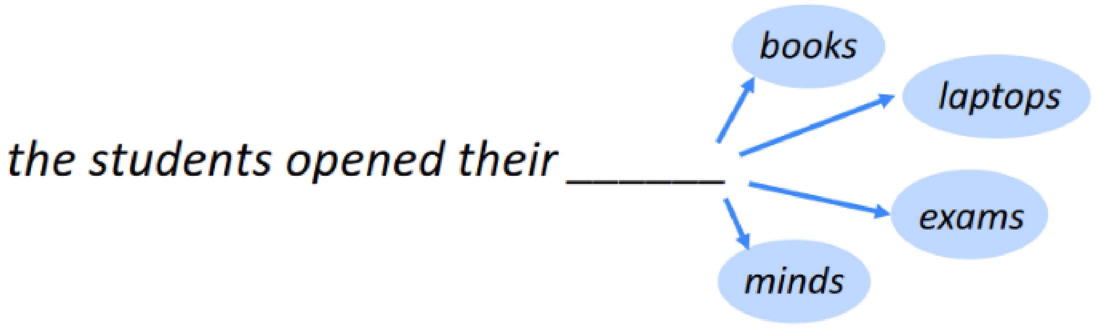


Image Credit: [1, 2]

Sequential Data

- Text generation
- Stock price prediction
- Weather prediction
- Audio Signal

- Simple solution: FCN or CNN
    - Fixed input/output length

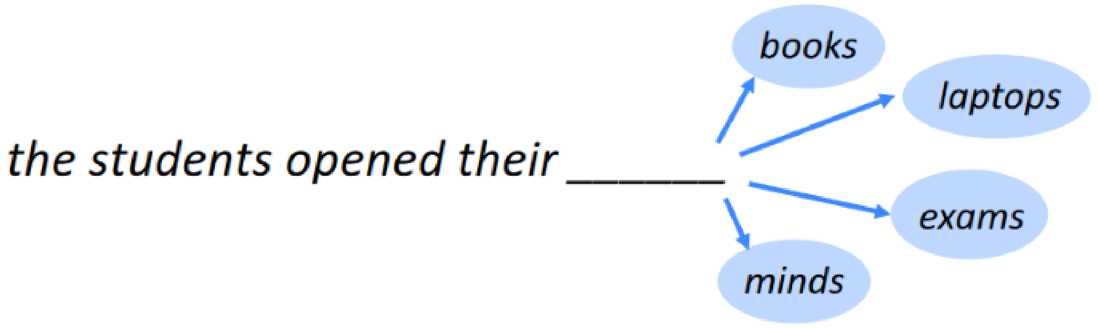
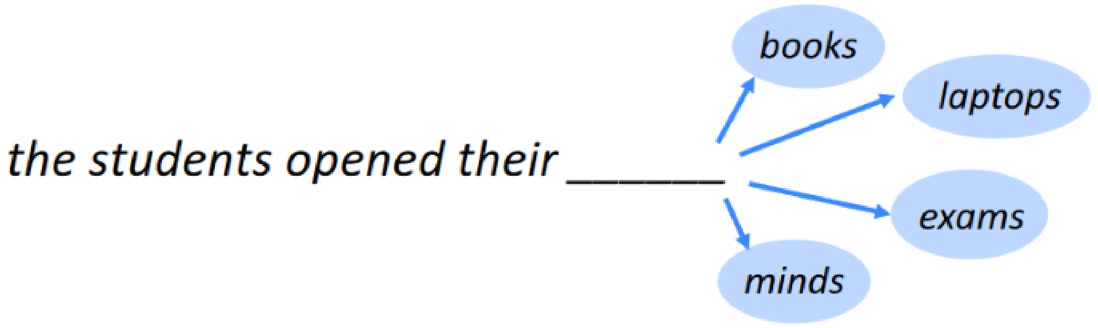


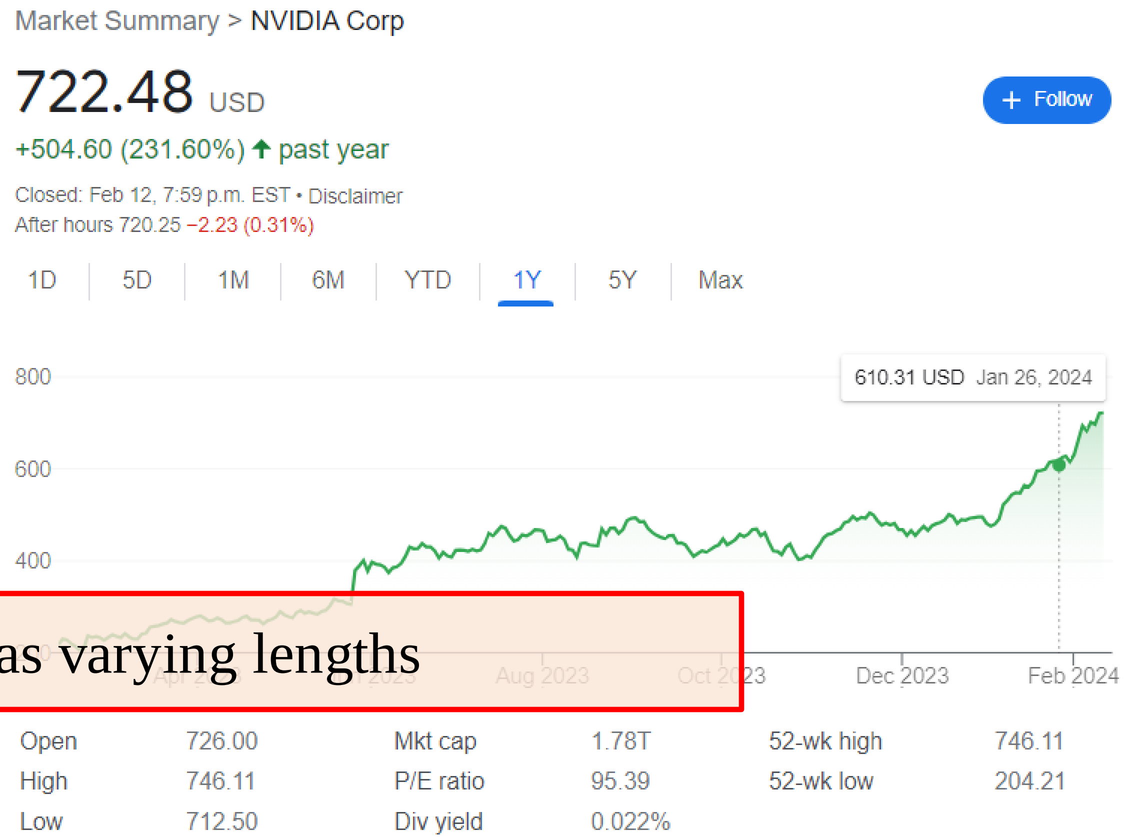
Image Credit: [1, 2]

Sequential Data

- Text generation
- Stock price prediction
- Weather prediction
- Audio Signal

- Simple solution: FCN or CNN
  - Fixed input/output length





Sequential data has varying lengths

Image Credit: [1, 2]

Processing Sequential Data in Various Lengths

- Sequence Models

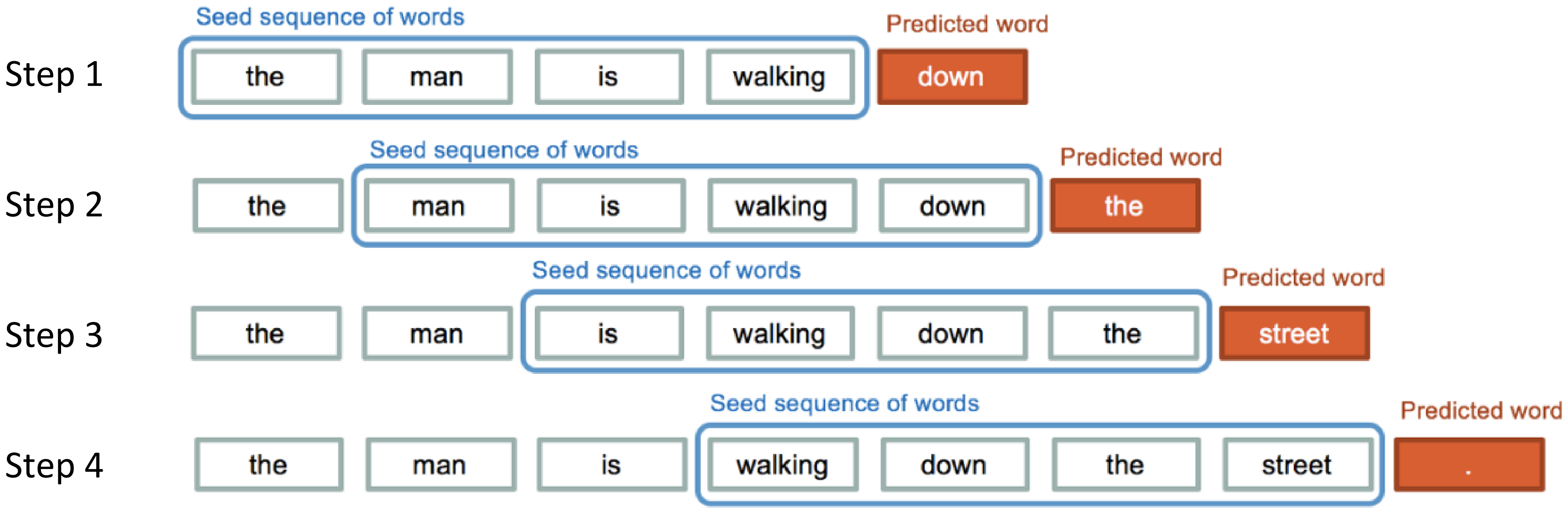
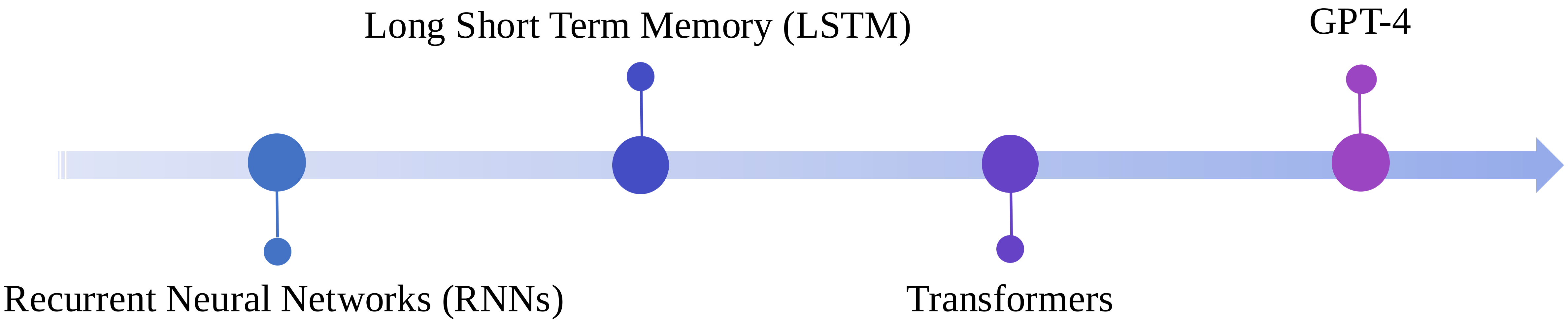
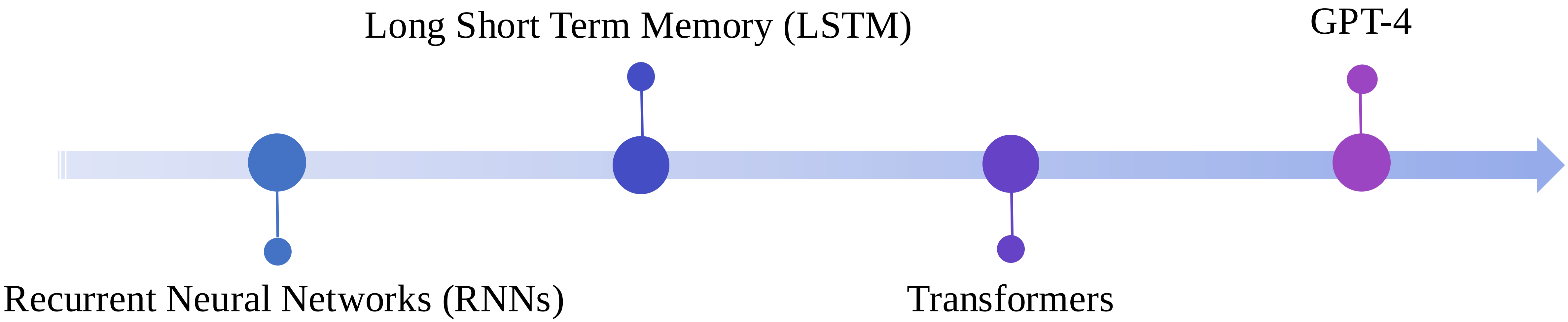


Image Credit: [3]

Evolution of Sequential Models



Evolution of Sequential Models



Why do we still need to learn RNN?

Evolution of Sequential Models

Long Short Term Memory (LSTM)

GPT-4

Recurrent Neural Networks (RNNs)

Advantages

- Sequential processing: natural fit for time series, speech recognition, language
- Parameter efficiency: shared weights across time steps.

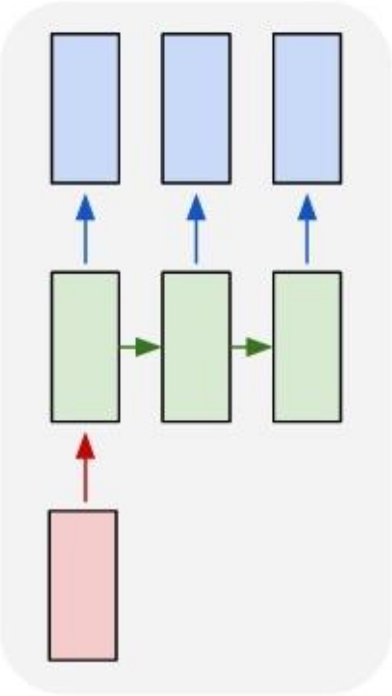
Transformers

Advantages

- Long-range dependencies: self-attention captures relationships between distant tokens
- Parallelization: processes input tokens simultaneously, faster training and inference.
- Scalability: state-of-the-art results in various NLP tasks, e.g., BERT, GPT, T5.

Applications

- One-to-many:
  - Music generation
  - Image captioning



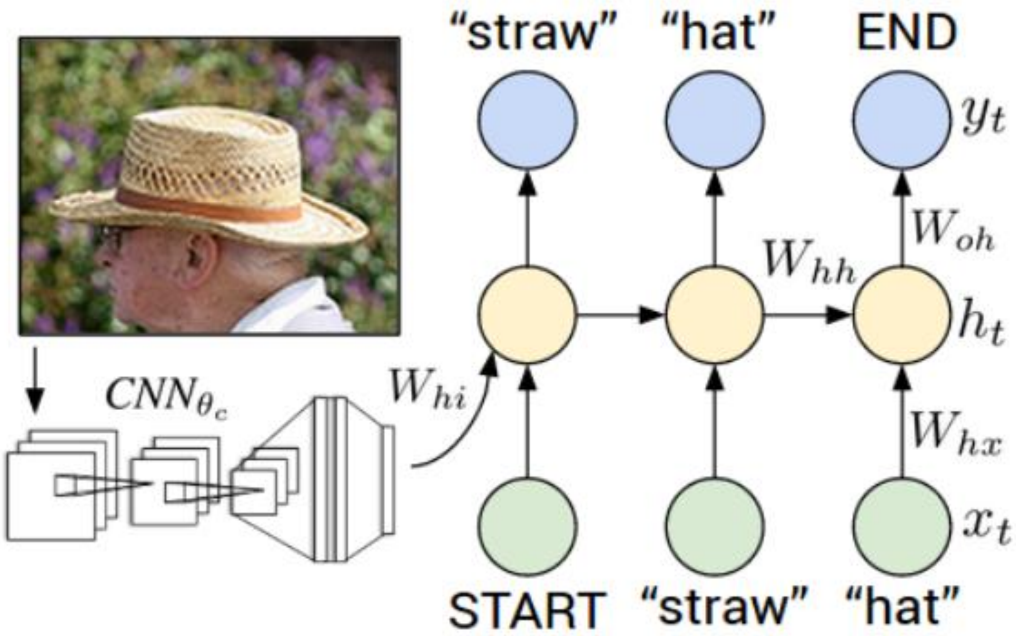
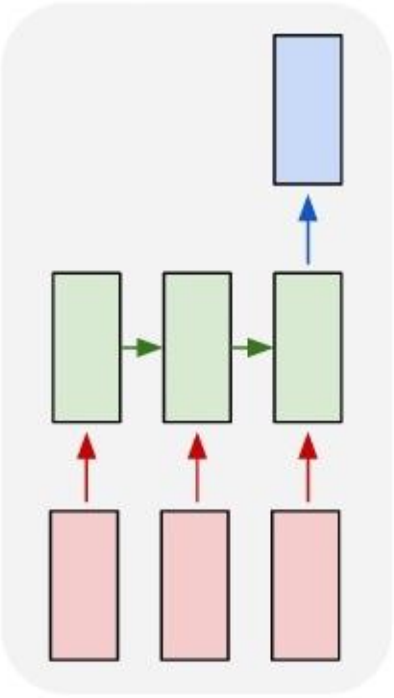


Image Credit: [4, 5]

Applications

- Many-to-one:
    - Sentiment classification



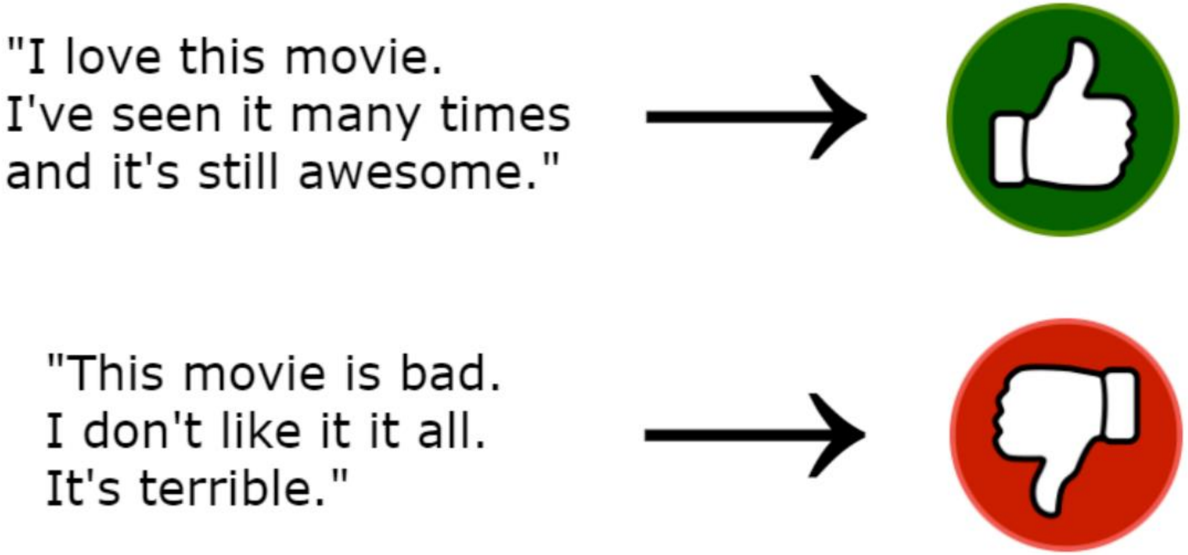
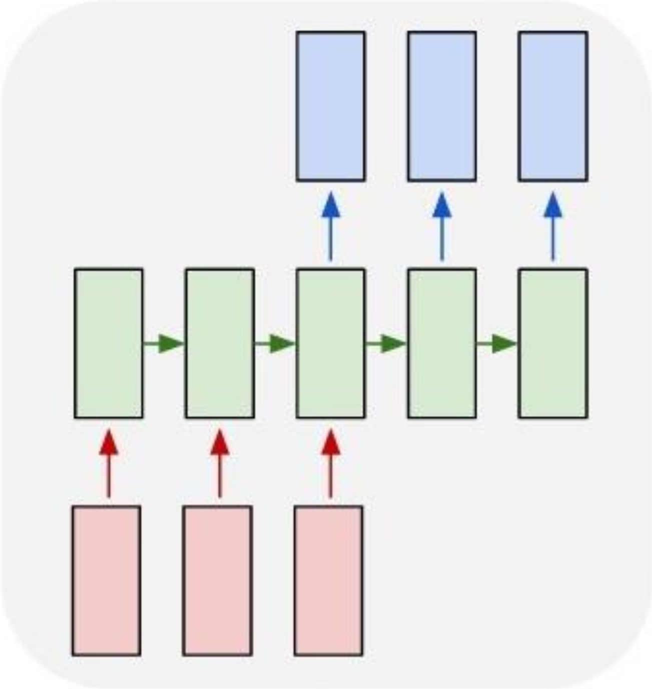


Image Credit: [4]

Applications

- Many-to-many:
    - Machine translation



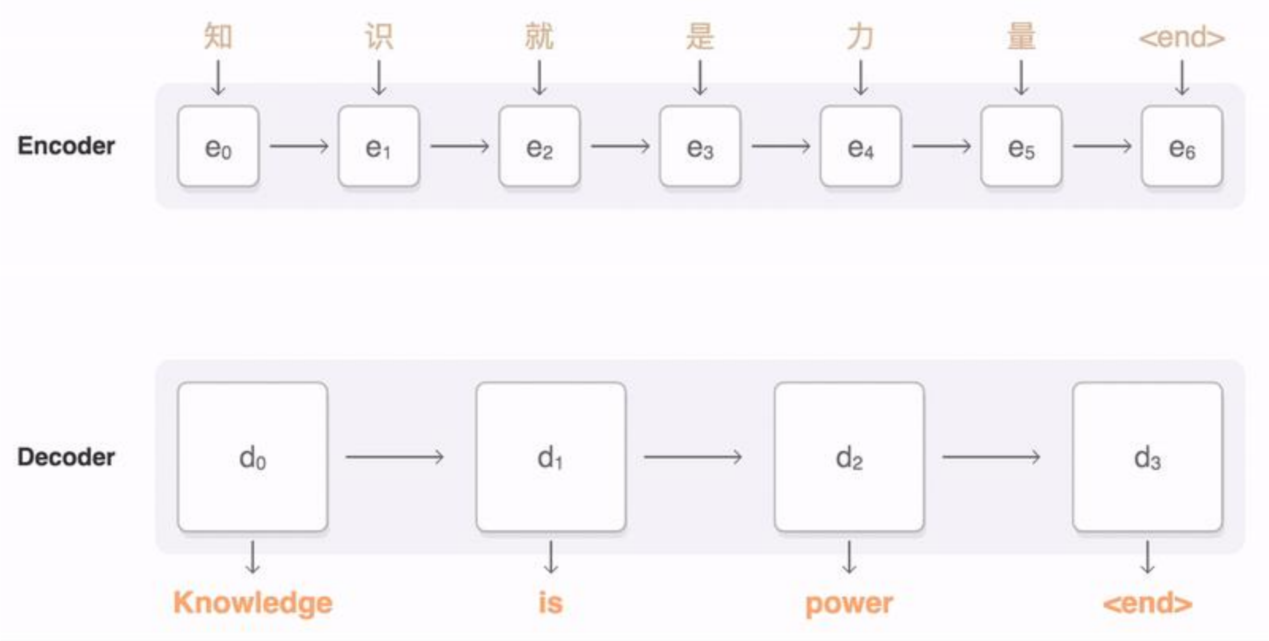
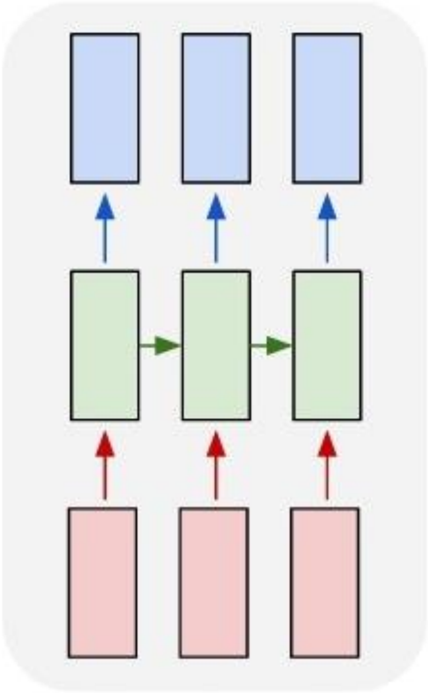


Image Credit: [4, 6]

Applications

- Many-to-many:
  - video classification on frame level



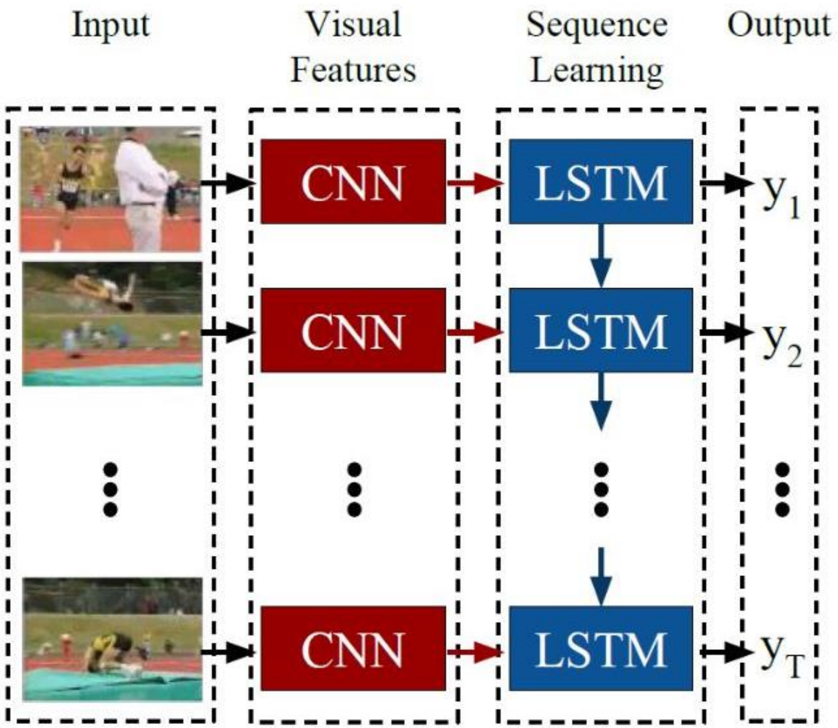


Image Credit: [4, 7]

Outline

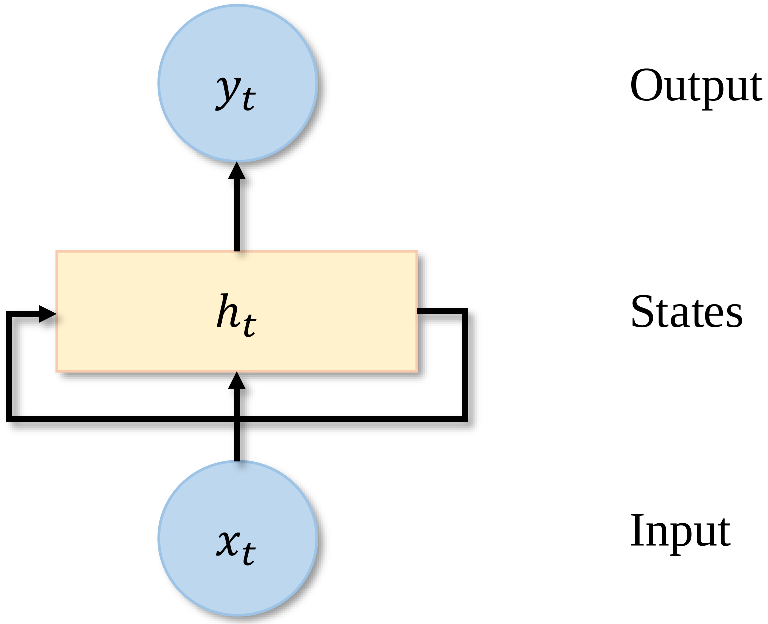
- Recurrent Neural Networks (RNNs)
  - Motivations
  - RNN basics
  - Long Short Term Memory (LSTM)
  - Back Propagation Through Time (BPTT)

Outline

- Recurrent Neural Networks (RNNs)
  - Motivations
  - **RNN basics**
  - Long Short Term Memory (LSTM)
  - Back Propagation Through Time (BPTT)

RNN Cell Unit

- Feedforward network: a neural network with no loops
- RNNs store information about previous data in the "state"
- Recurrently feeds output of activation function to itself



RNN Cell Unit

Recurrent neural networks (RNNs) are networks with loops, allowing information to persist [Rumelhart et al., 1986].

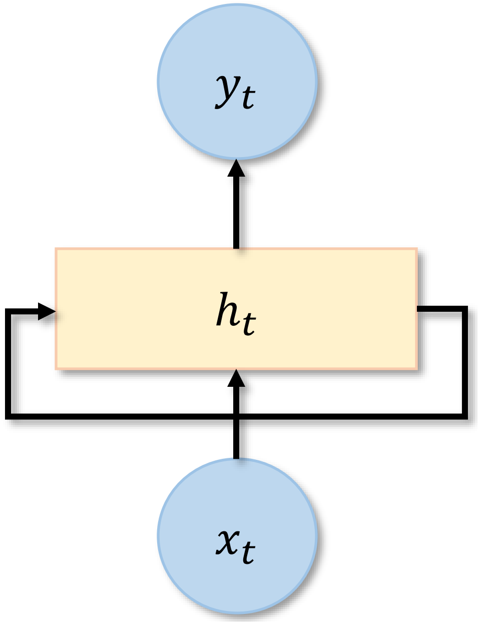
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

Some function with parameters W



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

RNN Cell Unit

Recurrent neural networks (RNNs) are networks with loops, allowing information to persist [Rumelhart et al., 1986].

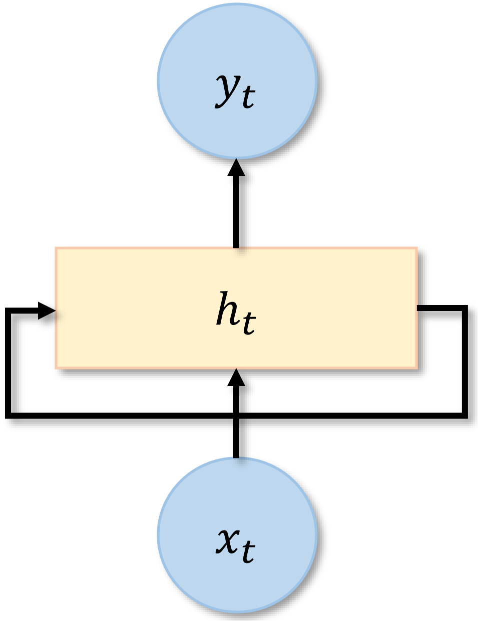
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

Some function with parameters W

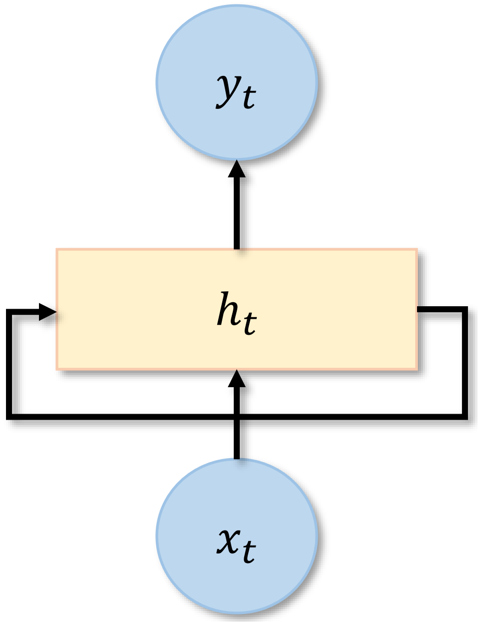


State variable summarizes/preserves the information observed so far.

RNN Readout

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

$$\underbrace{\boxed{y_t}}_{\text{output}} = \underbrace{\boxed{f_{W_{hy}}}}_{\text{another function with parameters W}}(\underbrace{\boxed{h_t}}_{\text{new state}})$$



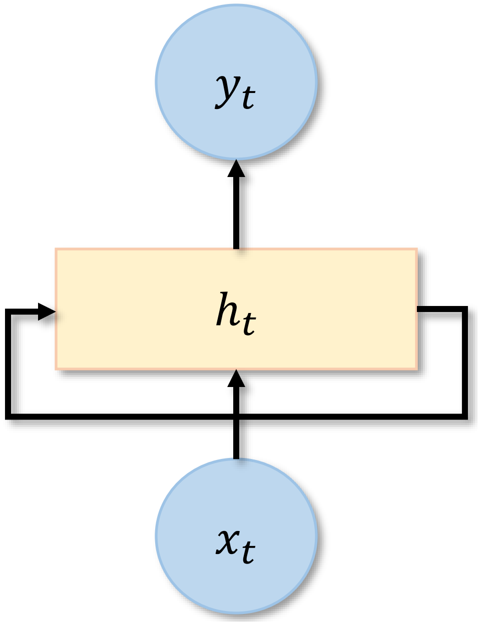
Formula of RNN (Vanilla)

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

$$\downarrow$$

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

$$y_t = f_{W_{hy}}(h_t)$$

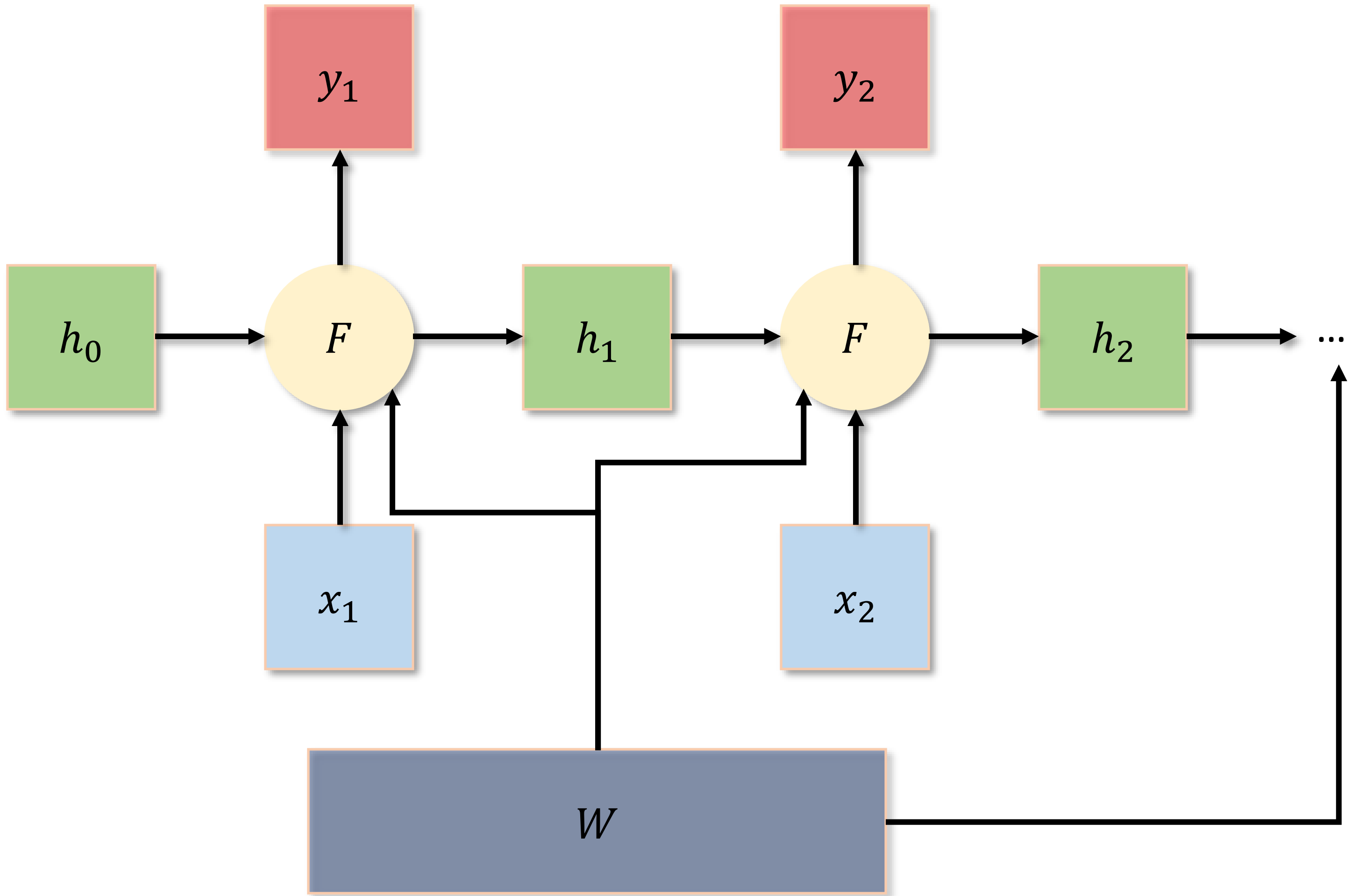


- x_t is the input at time t.
- h_t is the hidden state (memory) at time t.
- y_t is the output at time t.
- W_{hh}, W_{hx}, W_{hy} are distinct weights.
- weights are the same at all time steps

RNN Computational Graph

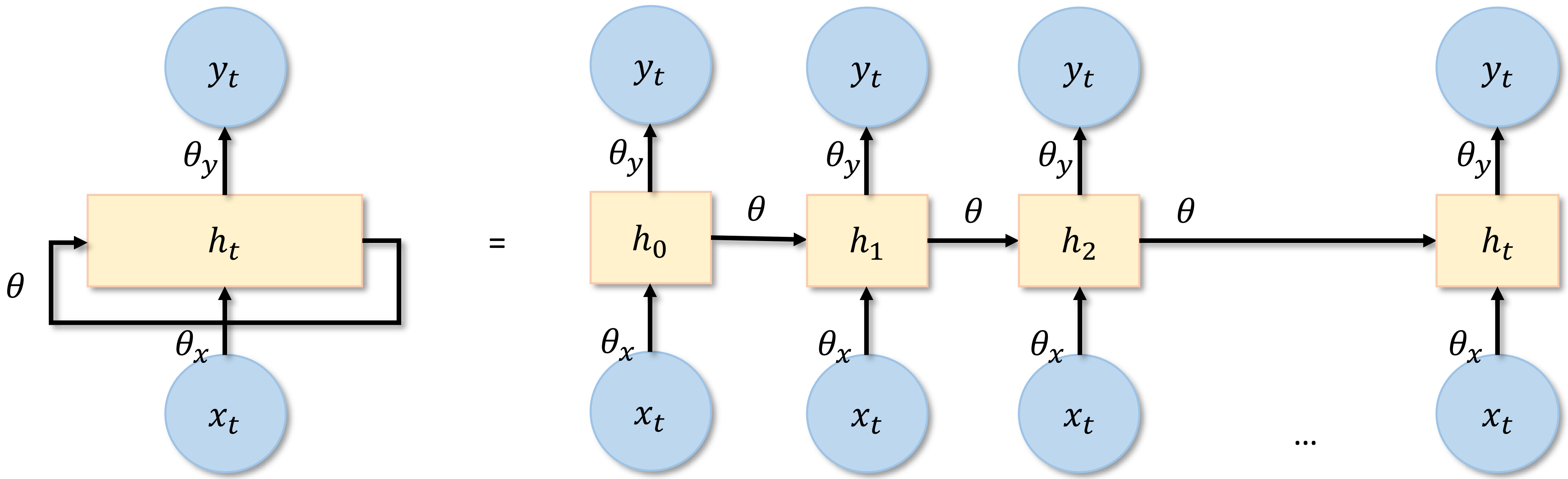
with shared (tied) weights

$$(h_1, y_1) = F(h_0, x_1, W)$$
$$(h_2, y_2) = F(h_1, x_2, W)$$



Parameter sharing

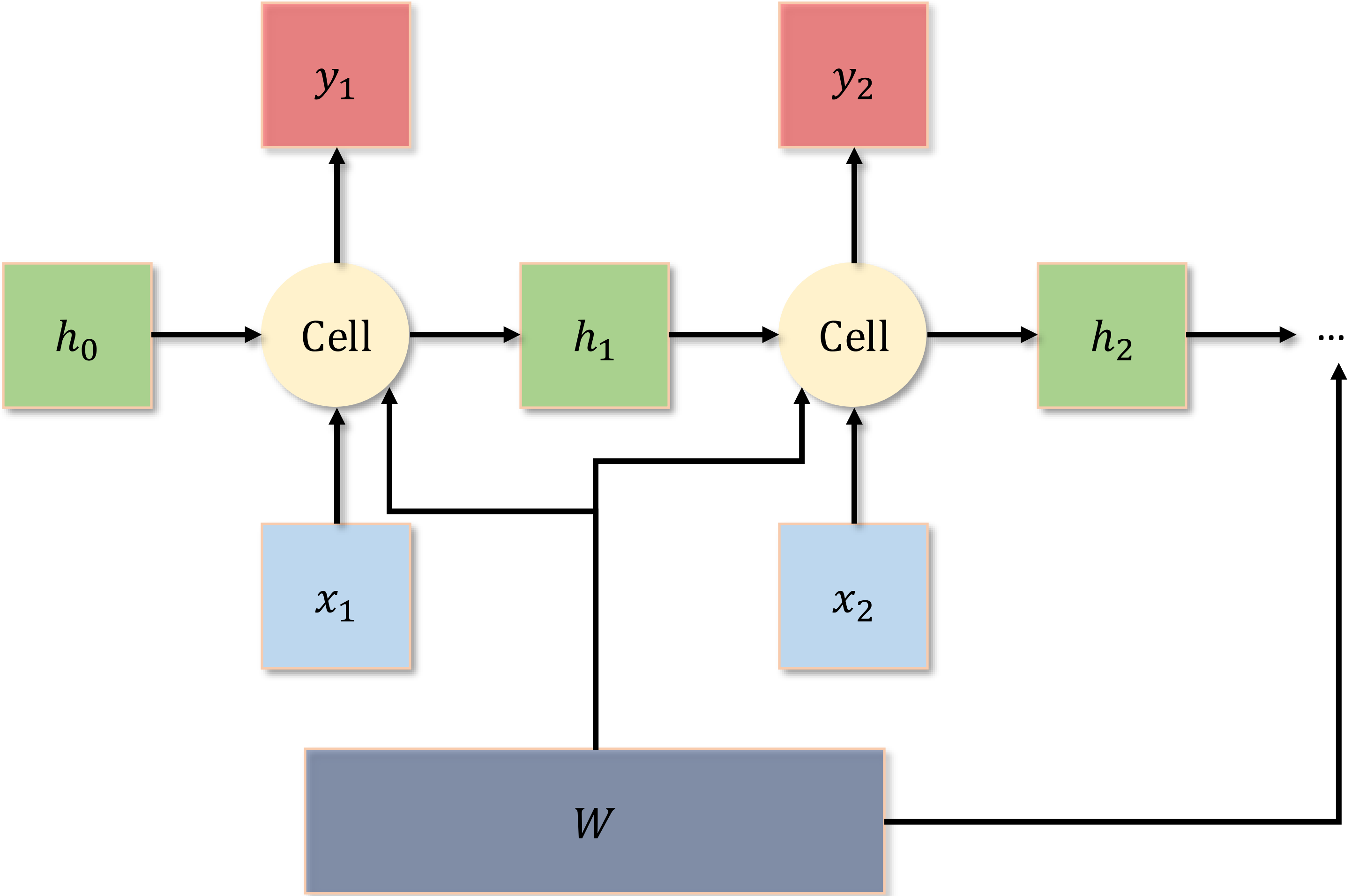
RNNs can be thought of as multiple copies of the same network, each passing a message to a successor.



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

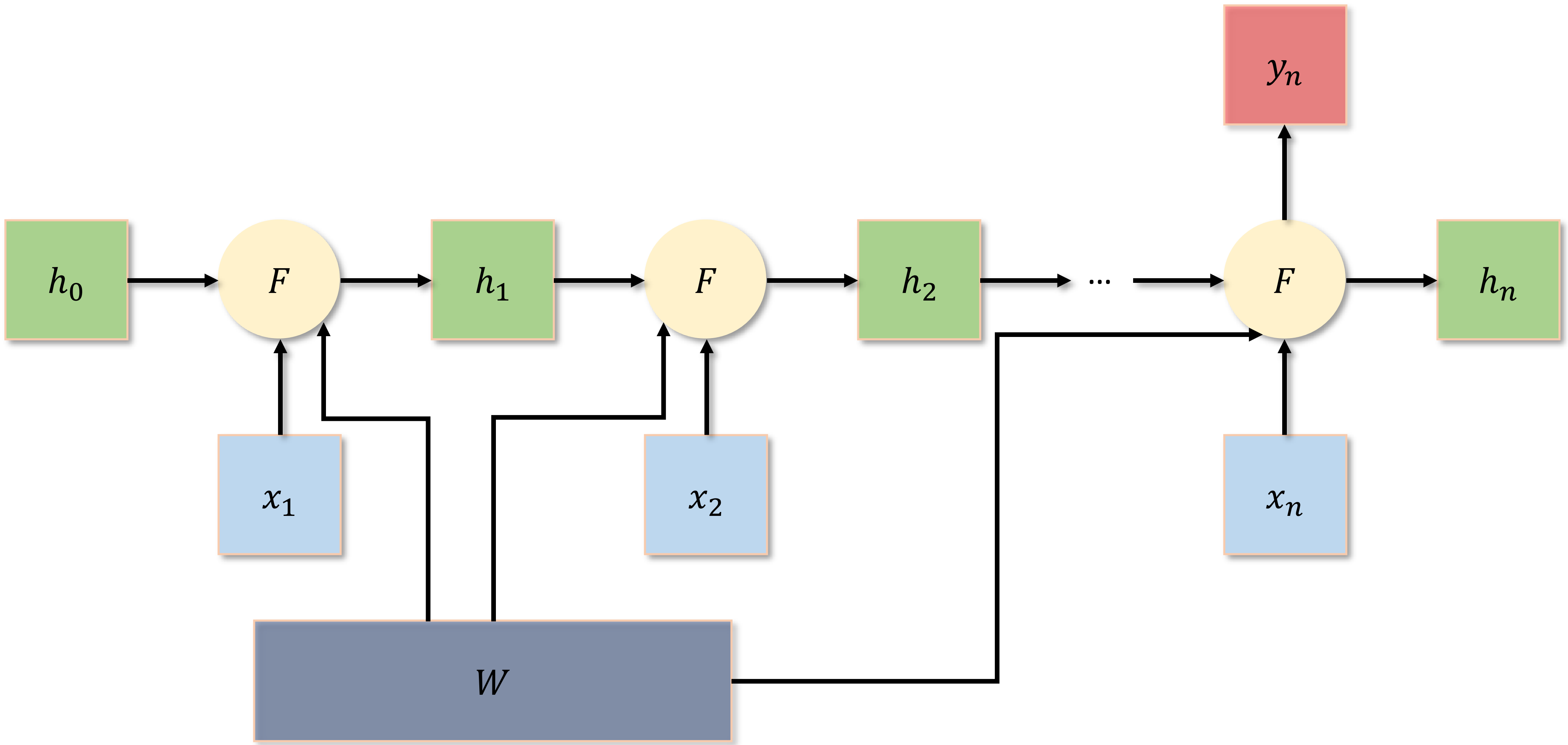
RNN Computational Graph

(x_1, x_2) Compress a length–2 sequence



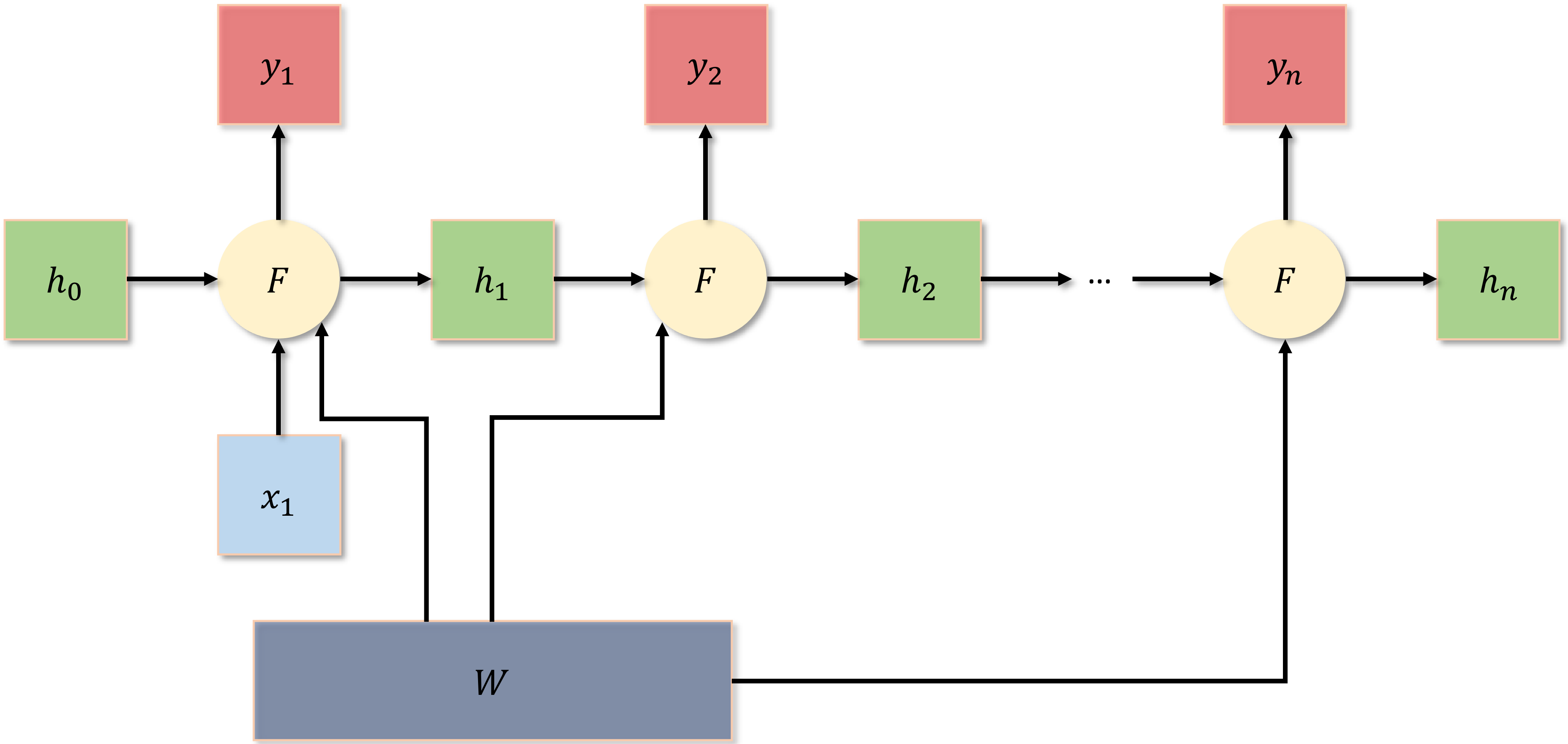
RNN Computational Graph

- Many-to-one



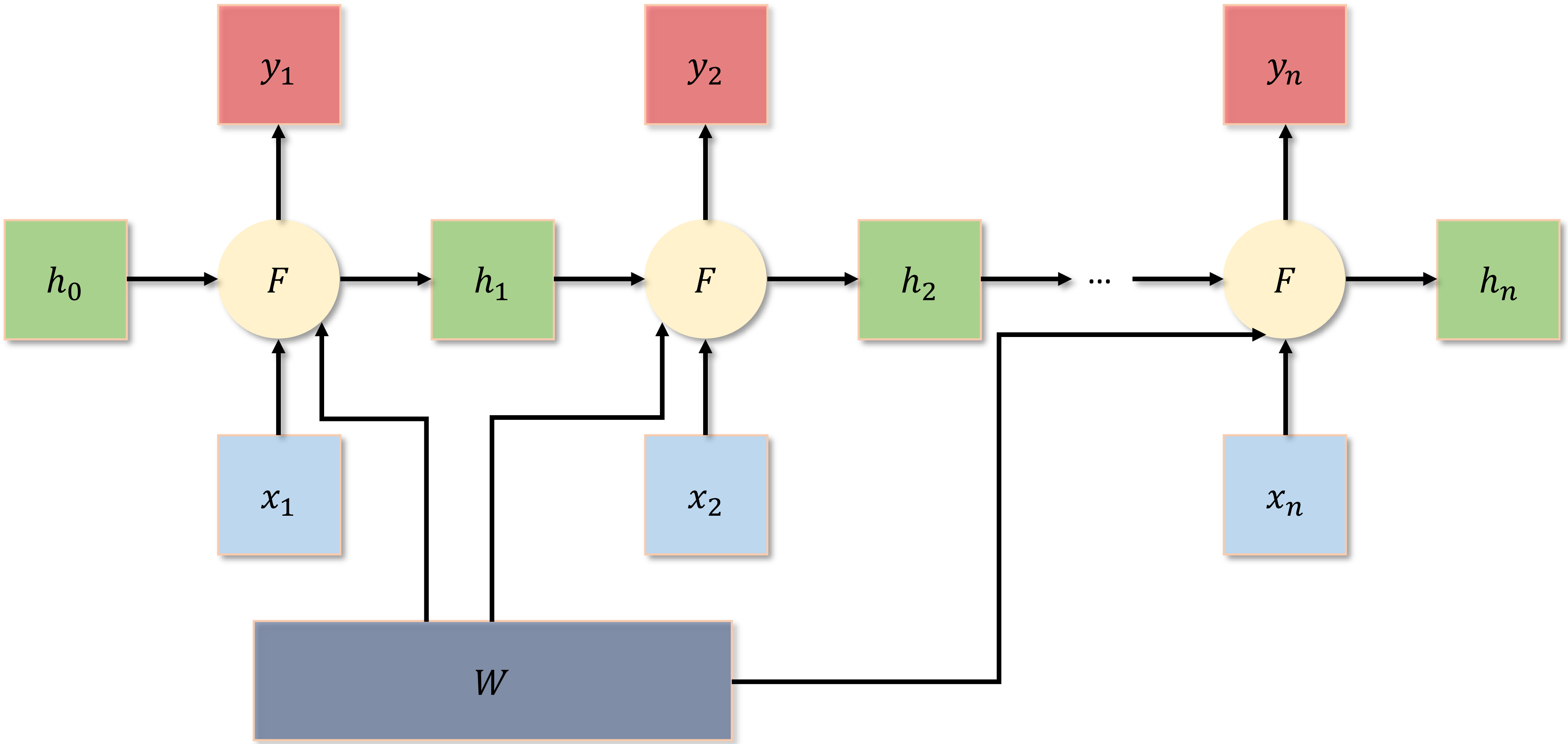
RNN Computational Graph

- One-to-many



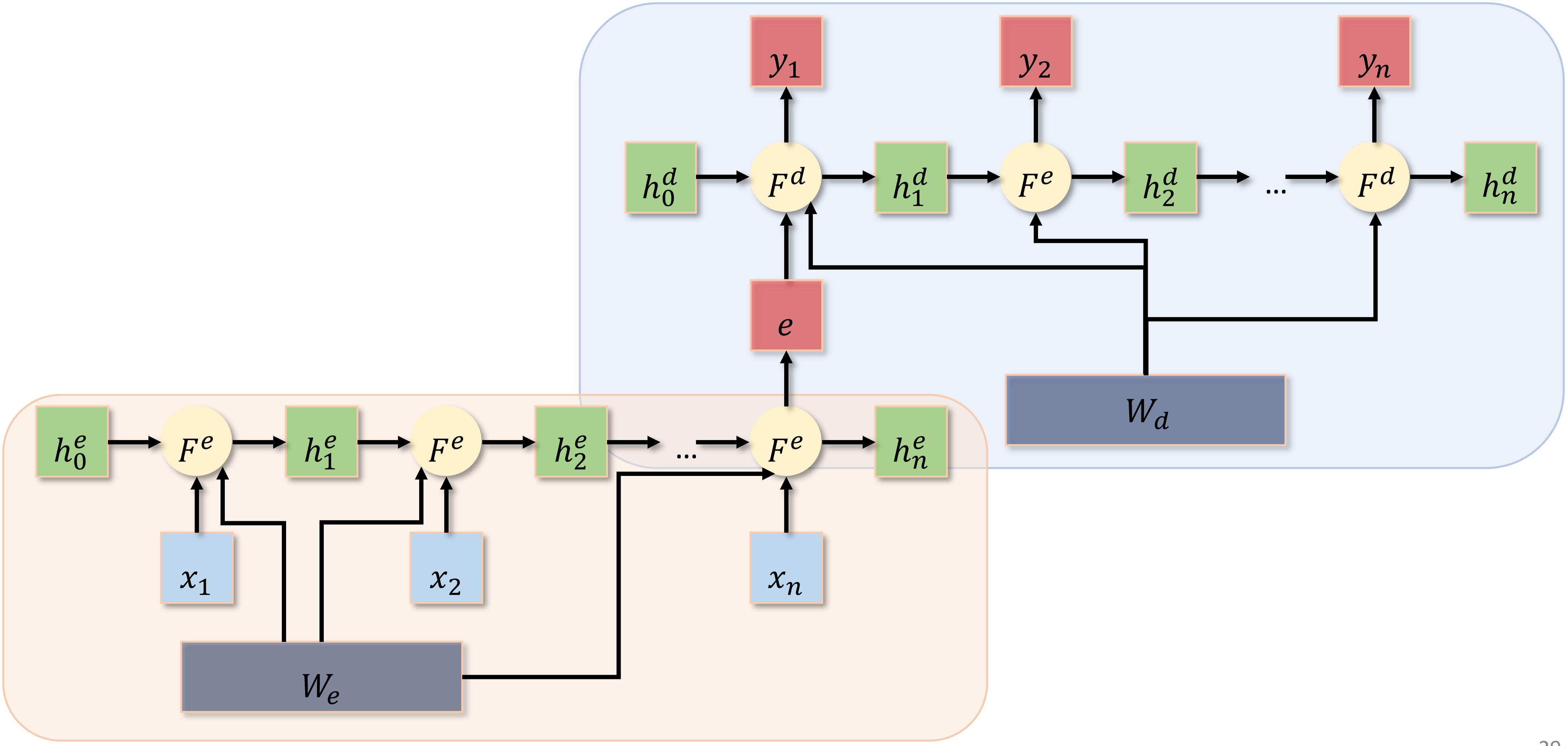
RNN Computational Graph

- Many-to-many



RNN Computational Graph

- Many-to-Many: Many-to-One + One-to-Many



Example: Character-level Language Model

Vacabulary:[h,e,l,o]

Example training sequece : “hello”

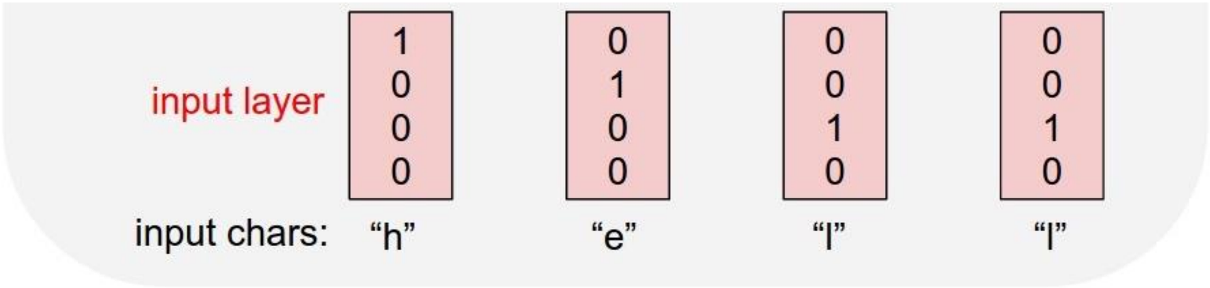


Image Credit: [4]

Example: Character-level Language Model

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

Vocabulary:[h,e,l,o]

Example training sequence : “hello”

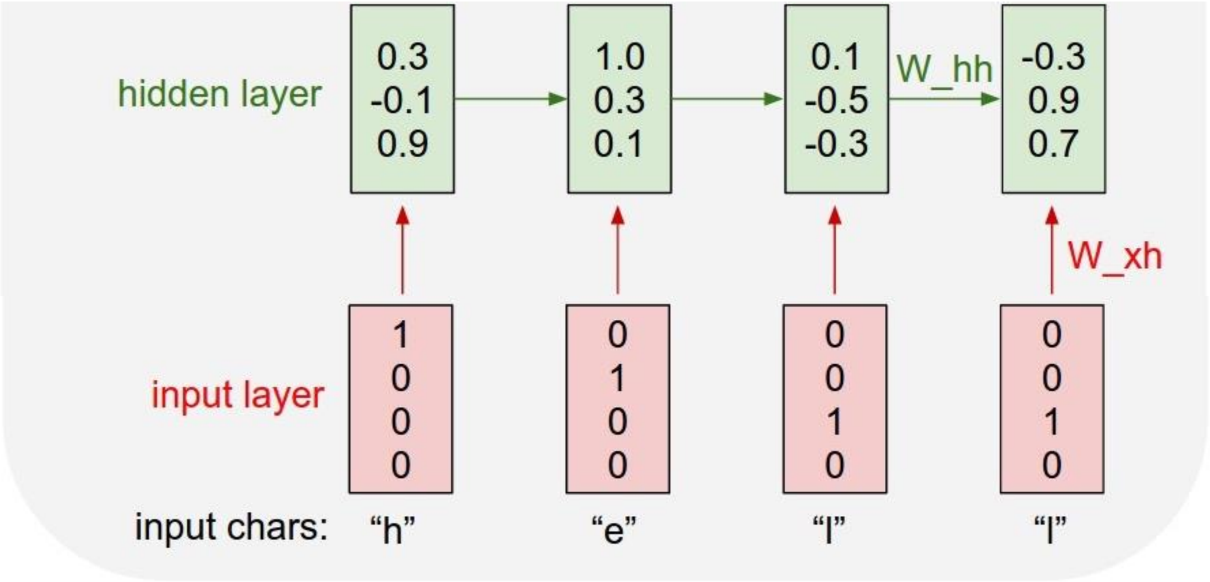


Image Credit: [4]

Example: Character-level Language Model

Vocabulary:[h,e,l,o]

Example training sequence : “hello”

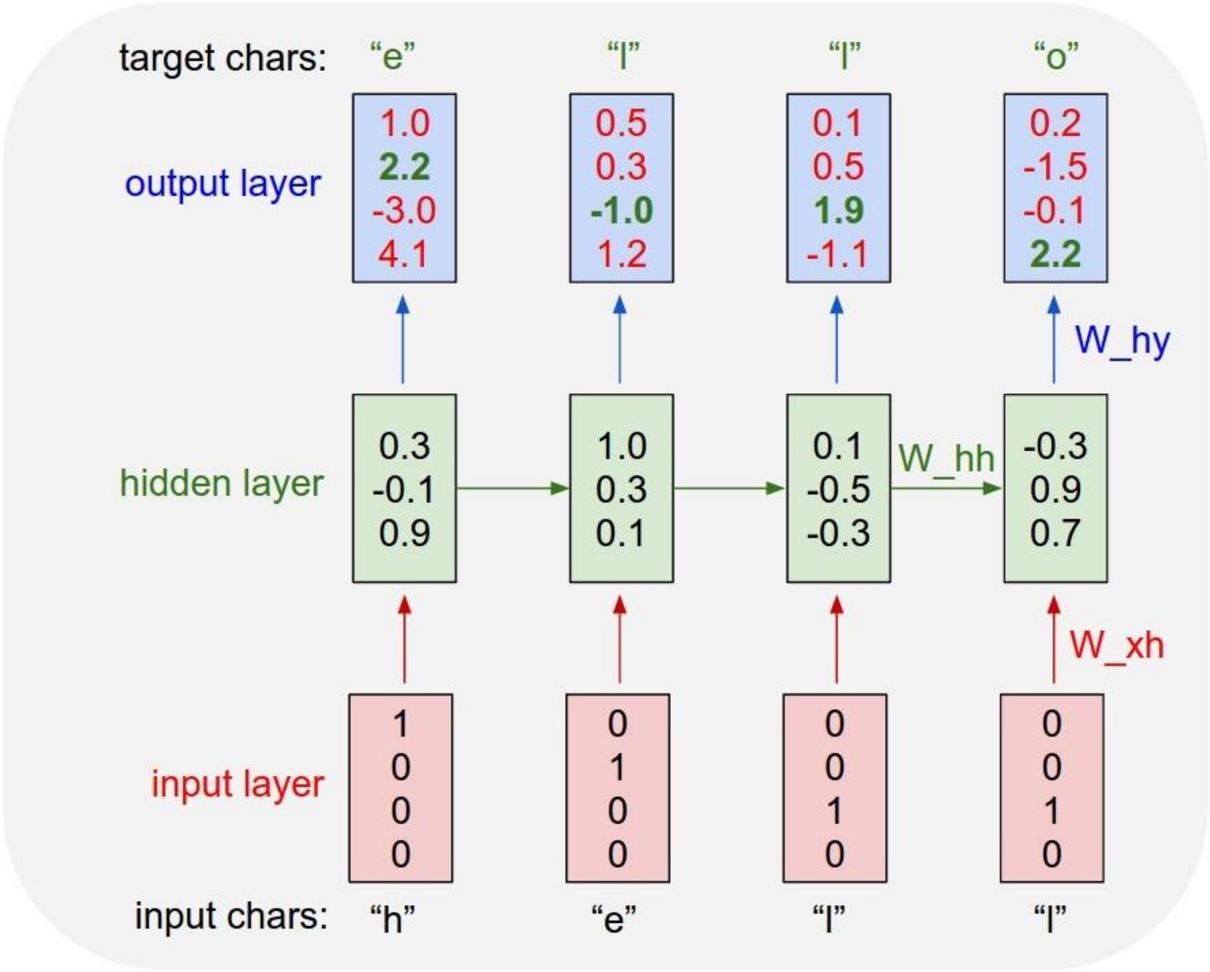


Image Credit: [4]

Example: Character-level Language Model

Vocabulary:[h,e,l,o]

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

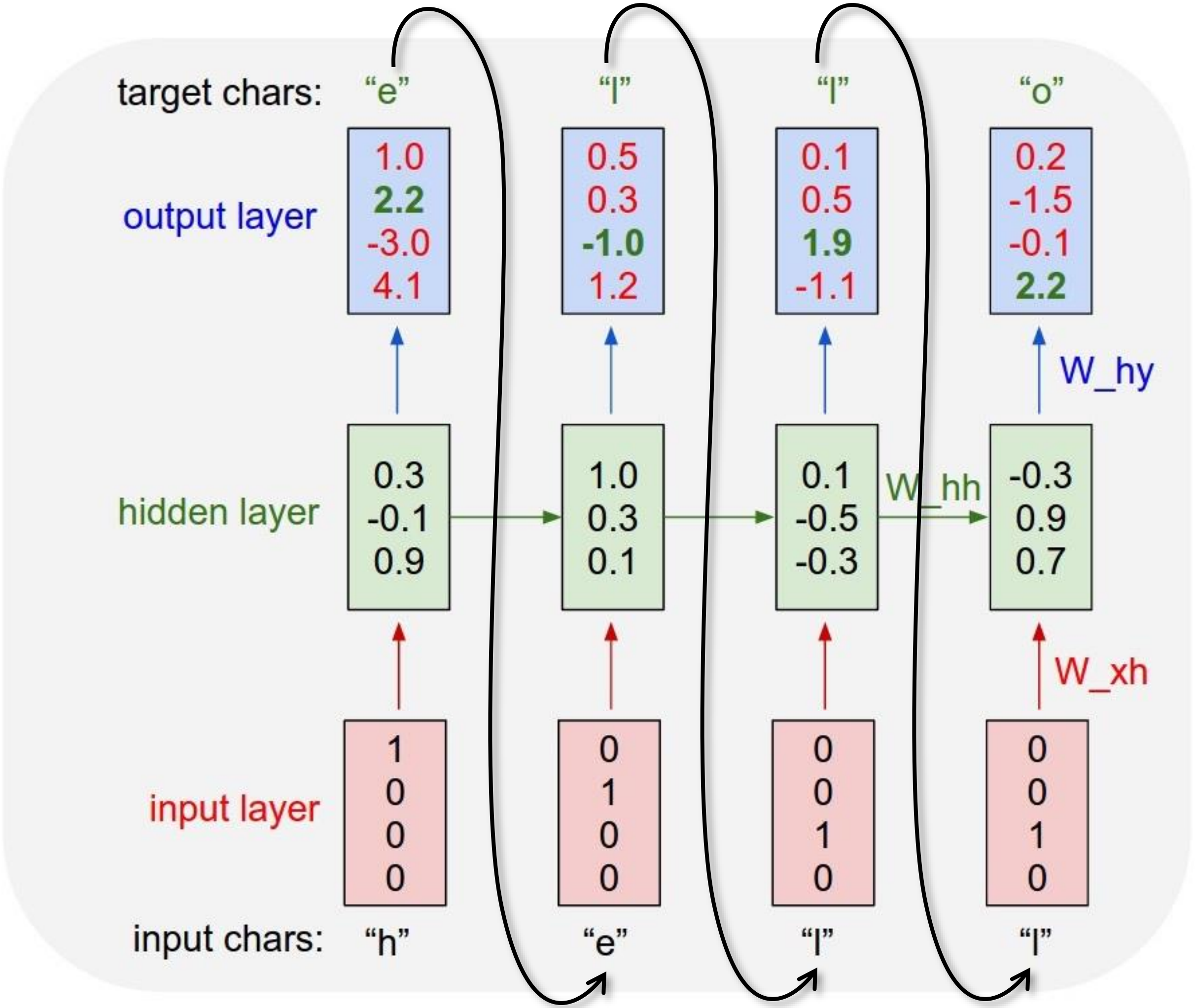


Image Credit: [4]

Outline

- Recurrent Neural Networks (RNNs)
  - Motivations
  - RNN basics
  - **Long Short Term Memory (LSTM)**
  - Back Propagation Through Time (BPTT)

The Problem of Long-term Dependencies

- In RNNs, during the gradient back propagation phase, the gradient signal can end up being multiplied many times.

- If the gradients are large
  - Exploding gradients, learning diverges
  - Solution: clip the gradients to a certain max value.

- If the gradients are small
  - Vanishing gradients, learning very slow or stops
  - Solution: introducing memory via LSTM, GRU, etc.

Long Short-Term Memory Networks

- Long Short-Term Memory (LSTM) networks are RNNs capable of learning long-term dependencies

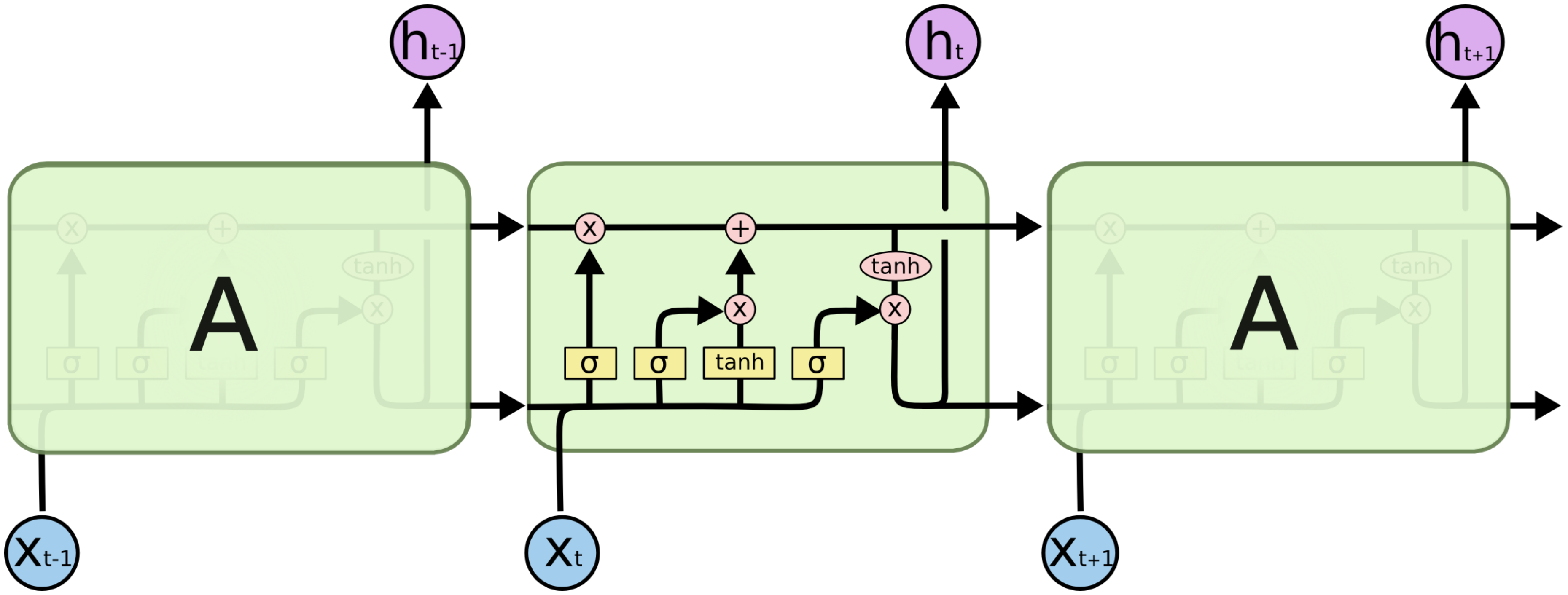
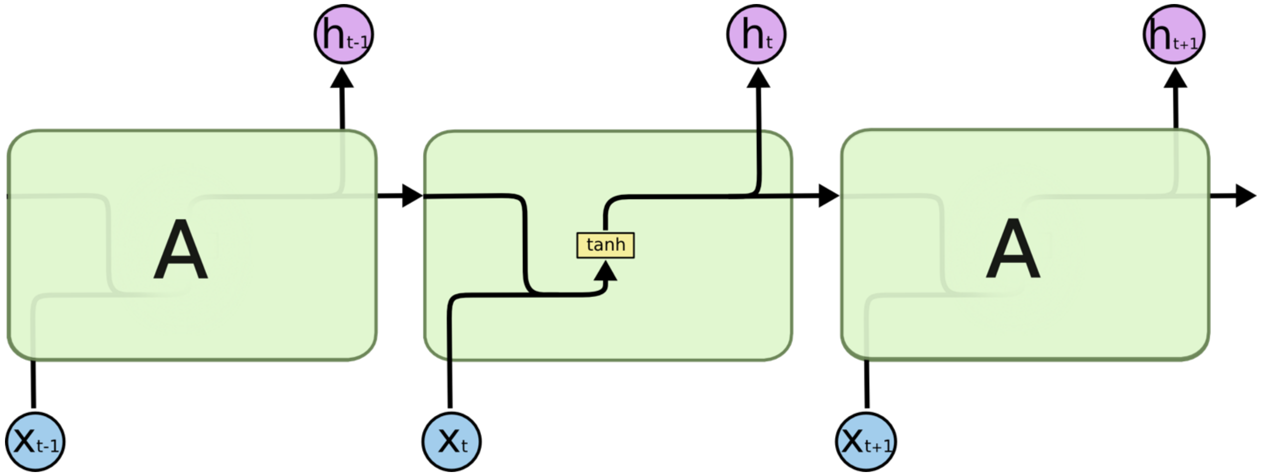


Image Credit: [4]

Vanilla RNN vs LSTM

RNN



LSTM

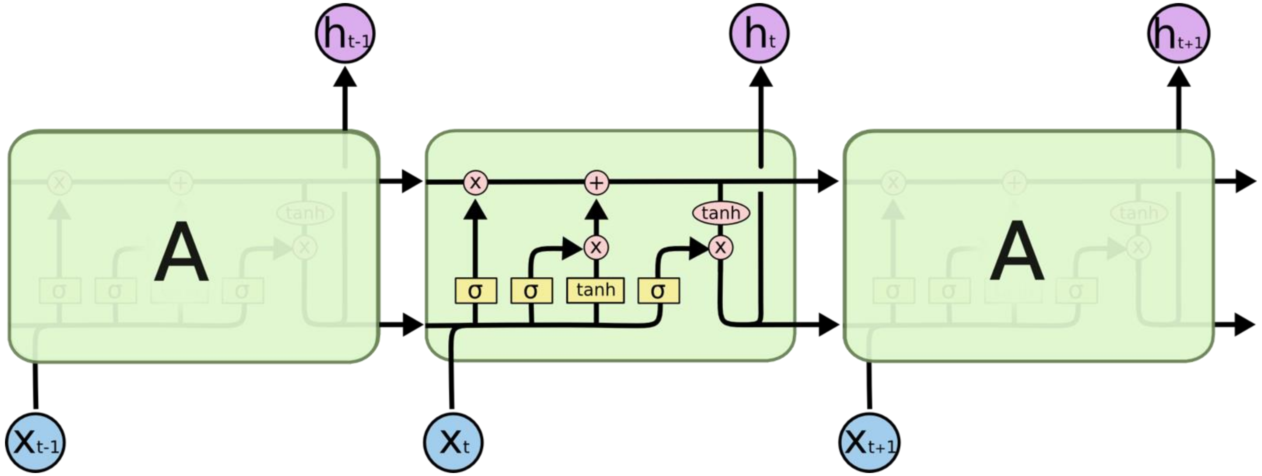
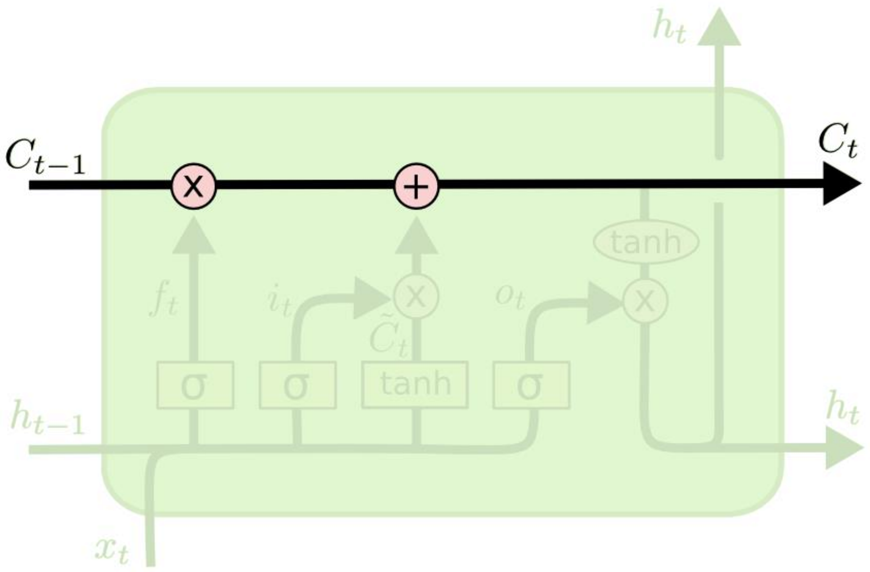


Image Credit: [4]

The Core Idea – Cell State

- The cell state is like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions.
- Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



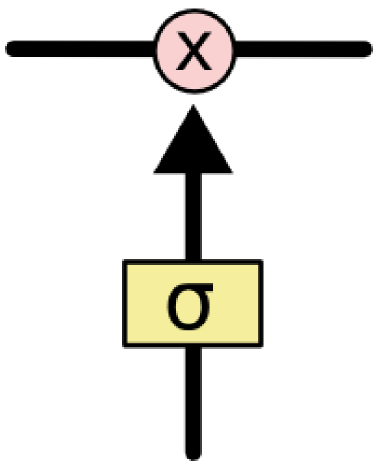
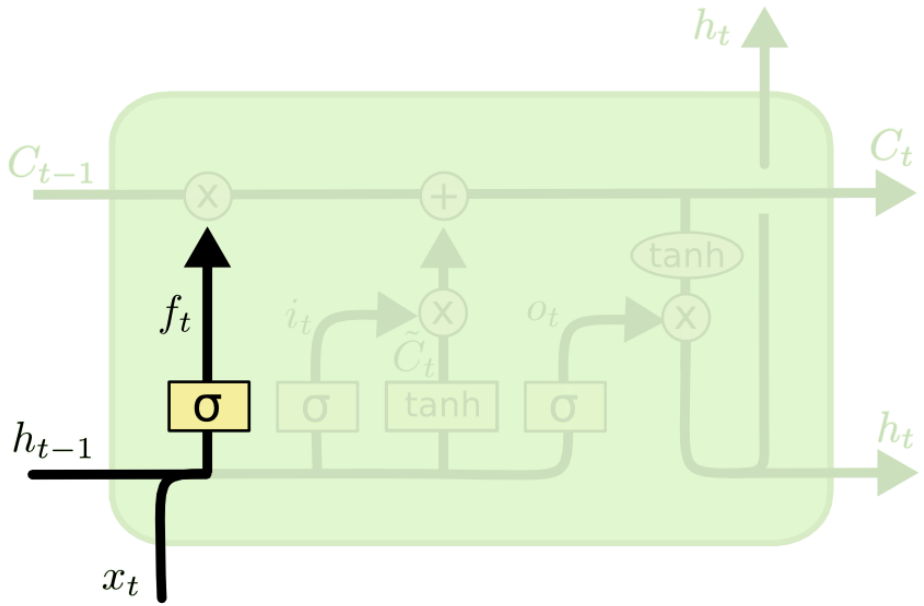


Image Credit: [4]

Step-by-Step LSTM Walk Through

- Forget gate layer:



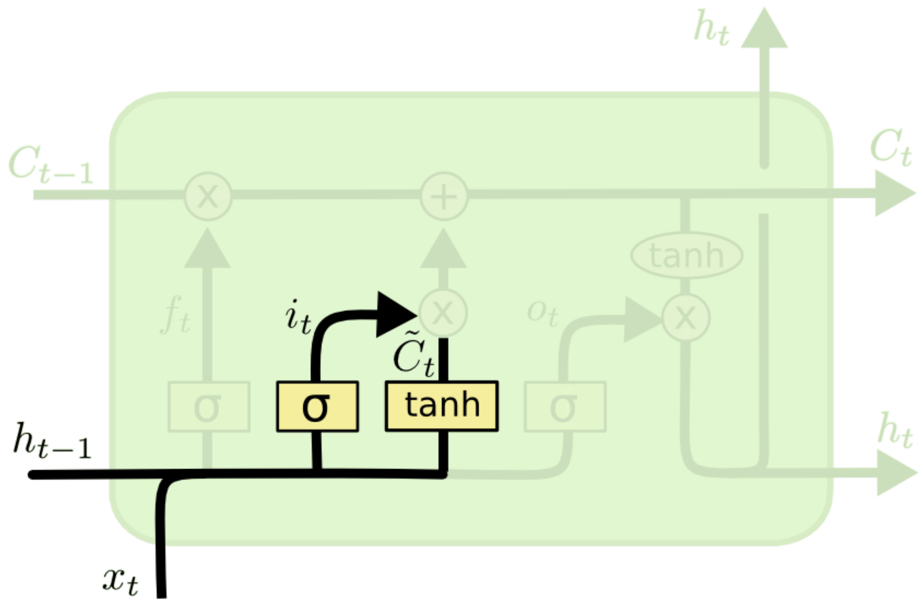
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Output: 0/1

Image Credit: [4]

Step-by-Step LSTM Walk Through

- Input gate layer + tanh layer:

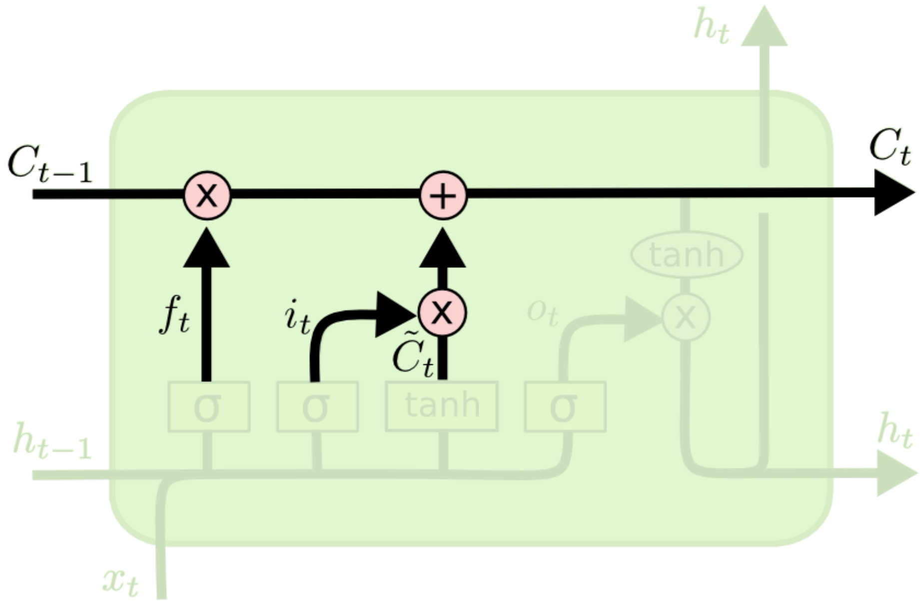


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Image Credit: [4]

Step-by-Step LSTM Walk Through

- Update Cell States:

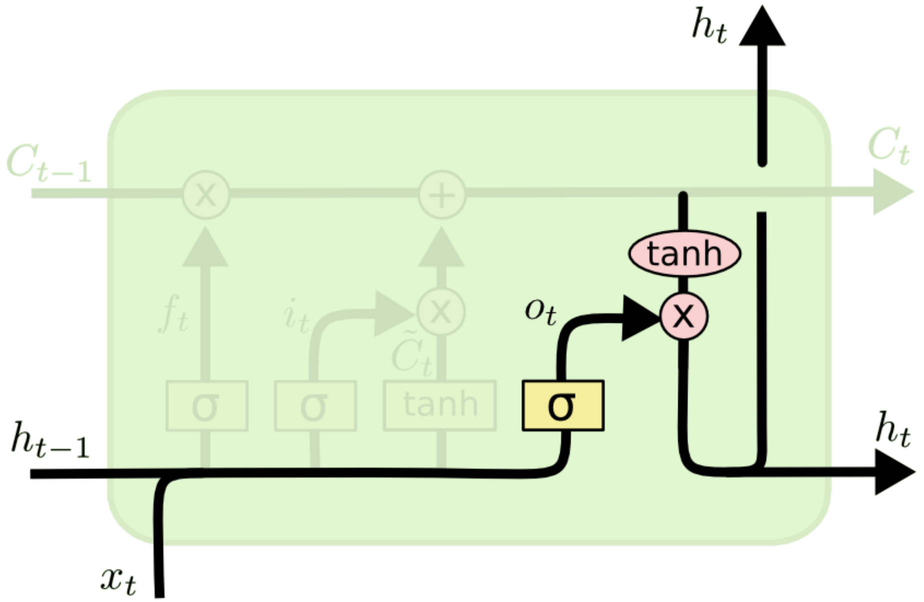


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Image Credit: [4]

Step-by-Step LSTM Walk Through

- Output:



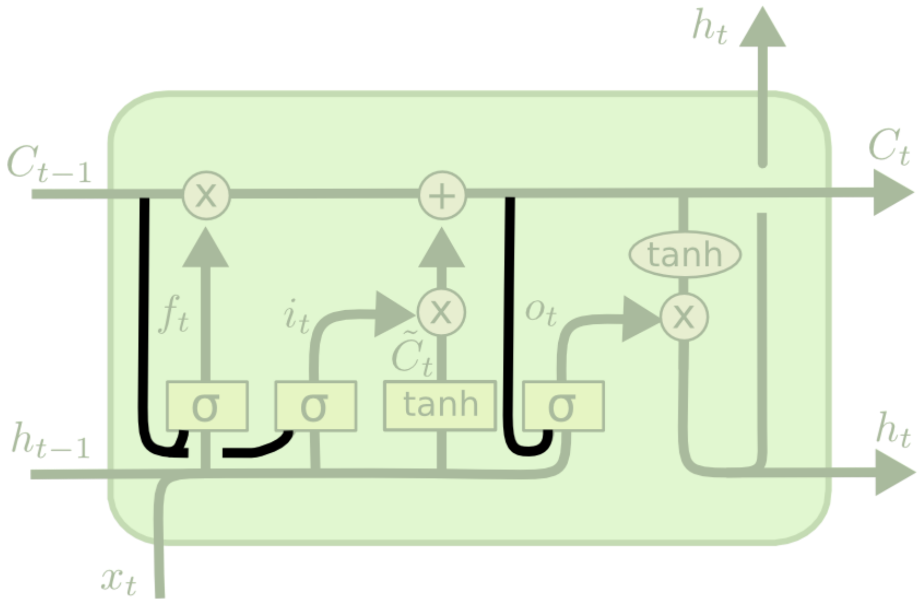
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

Image Credit: [4]

LSTM

•Allows "peeping into the memeory"



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

A memory cell using logistic and linear units with multiplicative interactions:

- Information gets into the cell whenever its input gate is on.
- Information is thrown away from the cell whenever its forget gate is off.
- Information can be read from the cell by turning on its output gate.

Image Credit: [4]

Outline

- Recurrent Neural Networks (RNNs)
  - Motivations
  - RNN basics
  - Long Short Term Memory (LSTM)
  - Back Propagation Through Time (BPTT)

Outline

- Recurrent Neural Networks (RNNs)
  - Motivations
  - RNN basics
  - Long Short Term Memory (LSTM)
  - **Back Propagation Through Time (BPTT)**

Back Propagation Through Time (BPTT)

- Applying backpropagation in RNNs is called *backpropagation through time* [9].
- This procedure requires us to expand (or unroll) the computational graph of an RNN one time step at a time.

BPTT: An Example

- Consider a vanilla RNN:
  - The RNN has no bias (absorbed into weight matrix)
  - Activation function is the identity function $\phi(x) = x$
- Notations:

$$\text{time step} : t$$

$$\text{input} : x_t \in \mathbb{R}^d$$

$$\text{target} : y_t \in \mathbb{R}$$

$$\text{hidden state} : h_t \in \mathbb{R}^h$$

$$\text{output} : o_t \in \mathbb{R}^q$$

BPTT: An Example

- RNN layer:

$$h_t = W_{\text{hx}} x_t + W_{\text{hh}} h_{t-1},$$
$$o_t = W_{\text{qh}} h_t,$$

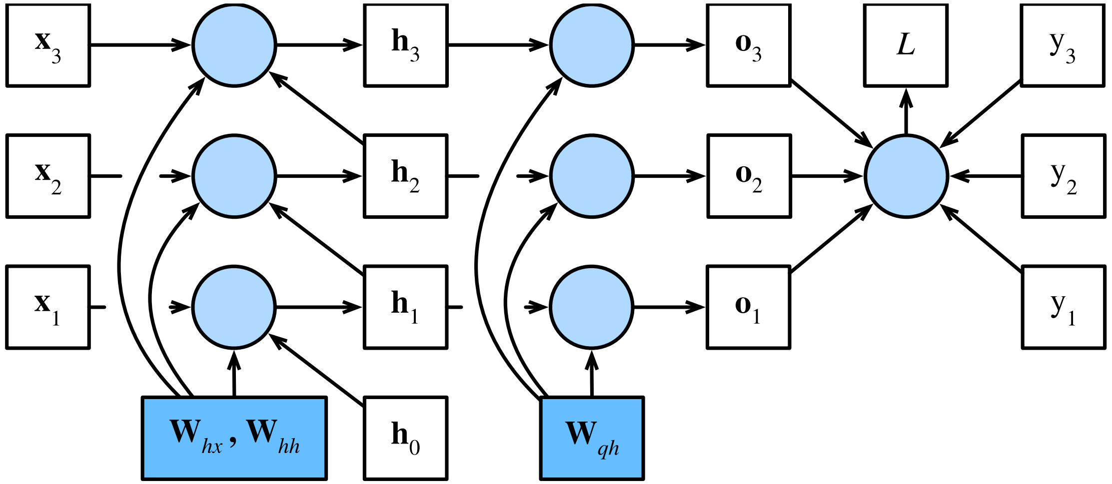
$$W_{\text{hx}} \in \mathbb{R}^{h \times d}, W_{\text{hh}} \in \mathbb{R}^{h \times h}, W_{\text{qh}} \in \mathbb{R}^{q \times h}$$

- Loss function (many-to-many):

$$L = \frac{1}{T} \sum_{t=1}^{T} l(o_t, y_t).$$

BPTT: An Example

- Computation graph:

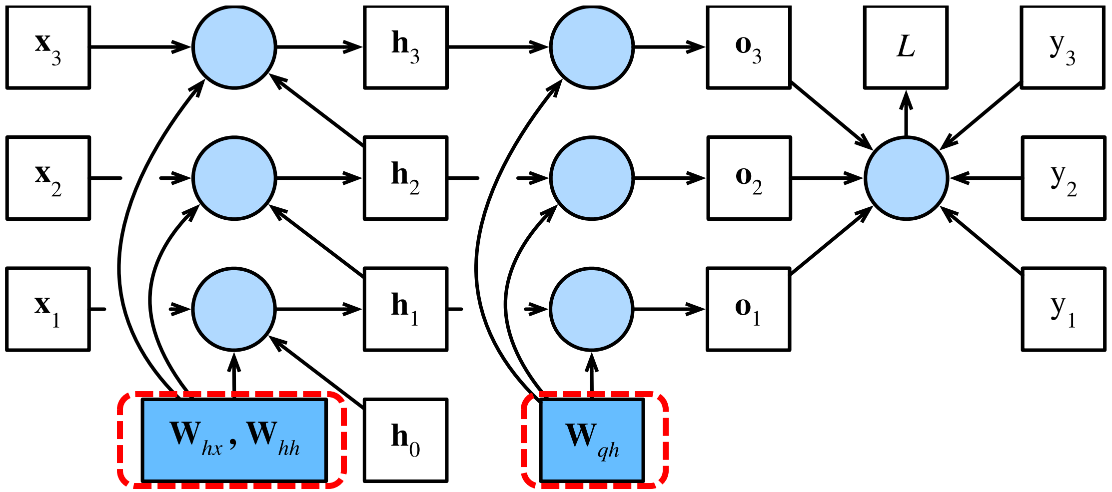


- Boxes represent variables (not shaded) or parameters (shaded) and circles represent operators.

Image Credit: [13]

BPTT: An Example

- Key components in gradient computation: $\partial L / \partial W_{\mathrm{hx}}, \partial L / \partial W_{\mathrm{hh}}, \partial L / \partial W_{\mathrm{qh}}$



- In the backpropagation, we traverse in the opposite direction of the arrows.

Image Credit: [13]

BPTT: An Example

- First, differentiating the loss w.r.t. the model output at any time step:

$$\frac{\partial L}{\partial o_t} = \frac{\partial l(o_t, y_t)}{T \cdot \partial o_t} \in \mathbb{R}^q.$$

BPTT: An Example

- First, differentiating the loss w.r.t. the model output at any time step:

$$\frac{\partial L}{\partial o_t} = \frac{\partial l(o_t, y_t)}{T \cdot \partial o_t} \in \mathbb{R}^q.$$

- Then, we can calculate the gradient of the loss w.r.t. to output layer parameter across all time steps:

$$\frac{\partial L}{\partial W_{\mathrm{qh}}} = \sum_{t=1}^{T} (\frac{\partial o_t}{\partial W_{\mathrm{qh}}})^\top \frac{\partial L}{\partial o_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial o_t} h_t^\top$$

BPTT: An Example

- First, differentiating the loss w.r.t. the model output at any time step:

$$\frac{\partial L}{\partial o_t} = \frac{\partial l(o_t, y_t)}{T \cdot \partial o_t} \in \mathbb{R}^q.$$

- Then, we can calculate the gradient of the loss w.r.t. to output layer parameter across all time steps:

$$\frac{\partial L}{\partial W_{\mathrm{qh}}} = \sum_{t=1}^{T} \left(\frac{\partial o_t}{\partial W_{\mathrm{qh}}}\right)^\top \frac{\partial L}{\partial o_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial o_t} h_t^\top$$

$$\partial L/\partial W_{\mathrm{hx}}, \partial L/\partial W_{\mathrm{hh}}, \underbrace{\partial L/\partial W_{\mathrm{qh}}}_{\text{done}}$$

BPTT: An Example

- Particularly, at the final time step T, the loss function L is related to hidden state h_T through only o_T :

$$\frac{\partial L}{\partial h_T} = (\frac{\partial o_T}{\partial h_T})^\top \frac{\partial L}{\partial o_T} = W_{\text{qh}}^\top \frac{\partial L}{\partial o_T}.$$

BPTT: An Example

- Particularly, at the final time step T, the loss function L is related to hidden state h_T through only o_T :

$$\frac{\partial L}{\partial h_T} = (\frac{\partial o_T}{\partial h_T})^\top \frac{\partial L}{\partial o_T} = W_{\text{qh}}^\top \frac{\partial L}{\partial o_T}.$$

- For the other time steps, we have:

$$\frac{\partial L}{\partial h_t} = (\frac{\partial h_{t+1}}{\partial h_t})^\top \frac{\partial L}{\partial h_{t+1}} + (\frac{\partial o_t}{\partial h_t})^\top \frac{\partial L}{\partial o_t} = W_{\text{hh}}^\top \frac{\partial L}{\partial h_{t+1}} + W_{\text{qh}}^\top \frac{\partial L}{\partial o_t}.$$

BPTT: An Example

- Particularly, at the final time step T, the loss function L is related to hidden state h_T through only o_T :

$$\frac{\partial L}{\partial h_T} = (\frac{\partial o_T}{\partial h_T})^\top \frac{\partial L}{\partial o_T} = W_{\text{qh}}^\top \frac{\partial L}{\partial o_T}.$$

- For the other time steps, we have:

$$\frac{\partial L}{\partial h_t} = (\frac{\partial h_{t+1}}{\partial h_t})^\top \boxed{\frac{\partial L}{\partial h_{t+1}}} + (\frac{\partial o_t}{\partial h_t})^\top \frac{\partial L}{\partial o_t} = W_{\text{hh}}^\top \boxed{\frac{\partial L}{\partial h_{t+1}}} + W_{\text{qh}}^\top \frac{\partial L}{\partial o_t}.$$

- Solving this recursion, it gives:

$$\frac{\partial L}{\partial h_t} = \sum_{i=t}^{T} \left(W_{\text{hh}}^\top\right)^{T-i} W_{\text{qh}}^\top \frac{\partial L}{\partial o_{T+t-i}}.$$

BPTT: An Example

Algorithm 1 BPTT

Inputs: Sequences $x_{1\ldots T}$, $o_{1\ldots T}$, $h_{1\ldots T}$, Loss L, (from forward pass), Truncation step K
for $t = T$ **down to** K **do**
    if $t = T$ **then**
        $\frac{\partial L}{\partial h_t} = W_{\text{qh}}^\top \frac{\partial L}{\partial o_t}$
    else
        $\frac{\partial L}{\partial h_t} = W_{\text{hh}}^\top \frac{\partial L}{\partial h_{t+1}} + W_{\text{qh}}^\top \frac{\partial L}{\partial o_t}$
    end if
end for
Calculate gradient with respect to weights:
$\frac{\partial L}{\partial W_{\text{hx}}} = \sum_{t=K}^{T} \left(\frac{\partial L}{\partial h_t} x_t^\top \right)$
$\frac{\partial L}{\partial W_{\text{hh}}} = \sum_{t=K}^{T} \left(\frac{\partial L}{\partial h_t} h_{t-1}^\top \right)$
$\frac{\partial L}{\partial W_{\text{qh}}} = \sum_{t=K}^{T} \left(\frac{\partial L}{\partial o_t} h_t^\top \right)$

BPTT: An Example

- Practical challenges of long sequences of matrix multiplication:

$$\frac{\partial L}{\partial h_t} = \sum_{i=t}^{T} \left(W_{\mathrm{hh}}^\top\right)^{T-i} W_{\mathrm{qh}}^\top \frac{\partial L}{\partial o_{T+t-i}}.$$

BPTT: An Example

- Practical challenges of long sequences of matrix multiplication:

$$\frac{\partial L}{\partial h_t} = \sum_{i=t}^{T} \left(W_{\mathrm{hh}}^\top\right)^{T-i} W_{\mathrm{qh}}^\top \frac{\partial L}{\partial o_{T+t-i}}.$$

- Powers of matrix $W_{\mathrm{hh}}^\top$ can be highly unstable.
  - eigenvalues smaller than 1 vanish
  - eigenvalues larger than 1 diverge

BPTT: An Example

- Practical challenges of long sequences of matrix multiplication:

$$\frac{\partial L}{\partial h_t} = \sum_{i=t}^{T} \left(W_{\mathrm{hh}}^\top\right)^{T-i} W_{\mathrm{qh}}^\top \frac{\partial L}{\partial o_{T+t-i}}.$$

- Powers of matrix $W_{\mathrm{hh}}^\top$ can be highly unstable.
  - eigenvalues smaller than 1 vanish
  - eigenvalues larger than 1 diverge

- This motivates the practical design of truncated in BPTT

BPTT: An Example

- For the remaining gradients $\partial L/\partial W_{\mathrm{hx}}, \partial L/\partial W_{\mathrm{hh}}$, we have:

$$\frac{\partial L}{\partial W_{\mathrm{hx}}} = \sum_{t=1}^{T} (\frac{\partial h_t}{\partial W_{\mathrm{hx}}})^\top \frac{\partial L}{\partial h_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} x_t^\top,$$

$$\frac{\partial L}{\partial W_{\mathrm{hh}}} = \sum_{t=1}^{T} (\frac{\partial h_t}{\partial W_{\mathrm{hh}}})^\top \frac{\partial L}{\partial h_t} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} h_{t-1}^\top,$$

BPTT: An Example

- For the remaining gradients $\partial L/\partial W_{\mathrm{hx}}, \partial L/\partial W_{\mathrm{hh}}$, we have:

$$\frac{\partial L}{\partial W_{\mathrm{hx}}} = \sum_{t=1}^{T} (\frac{\partial h_t}{\partial W_{\mathrm{hx}}})^\top \frac{\partial L}{\partial h_t} = \sum_{t=1}^{T} \boxed{\frac{\partial L}{\partial h_t}} x_t^\top,$$

$$\frac{\partial L}{\partial W_{\mathrm{hh}}} = \sum_{t=1}^{T} (\frac{\partial h_t}{\partial W_{\mathrm{hh}}})^\top \frac{\partial L}{\partial h_t} = \sum_{t=1}^{T} \boxed{\frac{\partial L}{\partial h_t}} h_{t-1}^\top,$$

- Note that we have computed $\partial L/\partial h_t$ in previous steps:

$$\frac{\partial L}{\partial h_t} = \sum_{i=t}^{T} \left(W_{\mathrm{hh}}^\top\right)^{T-i} W_{\mathrm{qh}}^\top \frac{\partial L}{\partial o_{T+t-i}}.$$

BPTT Analysis for general RNNs

RNN layer:

$$
\begin{aligned}
h_t &= f(x_t, h_{t-1}, w_\mathrm{h}),\\
o_t &= g(h_t, w_\mathrm{o}),
\end{aligned}
$$

Hence we have a chain of values:

$$
\{\ldots, (x_{t-1}, h_{t-1}, o_{t-1}), (x_t, h_t, o_t), \ldots\}
$$

Object function is evaluated over all T times as:

$$
L(x_1, \ldots, x_T, y_1, \ldots, y_T, w_\mathrm{h}, w_\mathrm{o}) = \frac{1}{T}\sum_{t=1}^{T} l(y_t, o_t)
$$

BPTT Analysis for general RNNs

Let's look at backpropagation, by chain rule, we have:

$$\begin{aligned}\frac{\mathrm{d}L}{\mathrm{d}w_\mathrm{h}} &= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial w_\mathrm{h}}\\ &= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_\mathrm{o})}{\partial h_t}\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}\end{aligned}$$

BPTT Analysis for general RNNs

Let's look at backpropagation, by chain rule, we have:

$$\frac{\mathrm{d}L}{\mathrm{d}w_\mathrm{h}} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial l(y_t, o_t)}{\partial w_\mathrm{h}}$$

$$= \frac{1}{T}\sum_{t=1}^{T} \underbrace{\boxed{\frac{\partial l(y_t, o_t)}{\partial o_t} \frac{\partial g(h_t, w_\mathrm{o})}{\partial h_t}}}_{\text{easy to compute}} \underbrace{\boxed{\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}}}_{h_{t-1} \text{ also depend on } w_h}$$

$$h_t = f(x_t, h_{t-1}, w_\mathrm{h})$$

easy to compute

h_{t-1} also depend on w_h

BPTT Analysis for general RNNs

Let's look at backpropagation, by chain rule, we have:

$$\begin{aligned}\frac{\mathrm{d}L}{\mathrm{d}w_\mathrm{h}} &= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial w_\mathrm{h}} \\ &= \frac{1}{T}\sum_{t=1}^{T}\underbrace{\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_\mathrm{o})}{\partial h_t}}_{\text{easy to compute}}\underbrace{\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}}_{h_{t-1}\text{ also depend on } w_h}\end{aligned}$$

$$h_t = f(x_t, h_{t-1}, w_\mathrm{h})$$

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial h_{t-1}}\frac{\mathrm{d}h_{t-1}}{\mathrm{d}w_\mathrm{h}}.$$

BPTT Analysis for general RNNs

Let's simplify notations by defining:

$$a_t = \frac{dh_t}{dw_h},$$
$$b_t = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h},$$
$$c_t = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}},$$

Given $a_t = b_t + c_t a_{t-1}$

And $a_0 = 0$

We have for t > 0:

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i.$$

BPTT Analysis for general RNNs

Let's simplify notations by defining:

$$a_t = \frac{dh_t}{dw_h},$$
$$b_t = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h},$$
$$c_t = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}},$$

$$\frac{dh_t}{dw_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}} \frac{dh_{t-1}}{dw_h}.$$

Given $a_t = b_t + c_t a_{t-1}$

And $a_0 = 0$

We have for t > 0:

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i.$$

BPTT Analysis for general RNNs

Let's simplify notations by defining:

$$a_t = \frac{dh_t}{dw_\mathrm{h}},$$

$$b_t = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}},$$

$$c_t = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial h_{t-1}},$$

$$\underbrace{\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}}_{a_t} = \underbrace{\frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}}_{b_t} + \underbrace{\frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial h_{t-1}}}_{c_t} \underbrace{\frac{\mathrm{d}h_{t-1}}{\mathrm{d}w_\mathrm{h}}}_{a_{t-1}}.$$

Given $a_t = b_t + c_t a_{t-1}$

And $a_0 = 0$

We have for t > 0:

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i.$$

BPTT Analysis for general RNNs

Let's simplify notations by defining:

$$a_t = \frac{dh_t}{dw_\mathrm{h}},$$
$$b_t = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}},$$
$$c_t = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial h_{t-1}},$$

Given $a_t = b_t + c_t a_{t-1}$

And $a_0 = 0$

We have for t > 0:

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i.$$

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial h_{t-1}} \frac{\mathrm{d}h_{t-1}}{\mathrm{d}w_\mathrm{h}}.$$

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}$$

BPTT Analysis for general RNNs

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}$$

- Can be computationally inefficient

- Gradients could explode or vanish

Truncated BPTT

- Truncation
  - Alternatively, we can truncate the sum after τ steps at $\partial h_{t-\tau}/\partial w_\mathrm{h}$

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial h_{t-1}} \frac{\mathrm{d}h_{t-1}}{\mathrm{d}w_\mathrm{h}}$$

$$\vdots$$

$$\frac{\mathrm{d}h_{t-\tau}}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_{t-\tau}, h_{t-1-\tau}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \frac{\partial f(x_{t-\tau}, h_{t-1-\tau}, w_\mathrm{h})}{\partial h_{t-1-\tau}} \frac{\mathrm{d}h_{t-1-\tau}}{\mathrm{d}w_\mathrm{h}}$$

  - We have:

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=t-\tau}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}$$

Randomized Truncated BPTT

- Randomized Truncation
  - We randomly truncate the sequence so that the gradients, e.g., $\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}$, are correct in expectation.

  - We introduce a sequence of independent variable ξ_t, with hyperparameter $0 \leq \pi_t \leq 1$

$$\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}} = \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \xi_j \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}.$$

where $P(\xi_t = 0) = 1 - \pi_t$

$$P(\xi_t = \pi_t^{-1}) = \pi_t$$

$$0 \leq \pi_t \leq 1$$

Randomized Truncated BPTT

- Given $\begin{aligned} &P(\xi_t = 0) = 1 - \pi_t \\ &P(\xi_t = \pi_t^{-1}) = \pi_t \\ &0 \leq \pi_t \leq 1 \end{aligned}$ and $\xi_1, \xi_2, \ldots, \xi_T$ being independent

- We have $\mathbb{E}[\xi_t] = \pi_t$ and $\mathbb{E}[\xi_i \xi_j] = \mathbb{E}[\xi_i]\mathbb{E}[\xi_j]$

$$
\begin{aligned}
\mathbb{E}\left[\frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}\right] &= \mathbb{E}\left[\frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} \xi_j \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}}\right)\frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}\right] \\
&= \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}} \mathbb{E}\left[\prod_{j=i+1}^{t} \xi_j\right]\right)\frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} \\
&= \frac{\partial f(x_t, h_{t-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}} \prod_{j=i+1}^{t} \mathbb{E}[\xi_j]\right)\frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} = \frac{\mathrm{d}h_t}{\mathrm{d}w_\mathrm{h}}
\end{aligned}
$$

Randomized Truncated BPTT

- Let's look at a special case of previous general form
- We can truncate by simply sampling a truncate step K uniformly from $1, \ldots, T$, so that we get a uniform probability $\frac{1}{T}$ to truncate at each time step.

- Let's map K to a size-$(\mathrm{T}-1)$ random binary vector $\boldsymbol{m}$ such that for any $k \in [T]$, we have

$$P\left(\mathbf{m} = \left[\mathbb{1}_{\{k\leq 1\}}, \mathbb{1}_{\{k\leq 2\}}, \ldots, \mathbb{1}_{\{k\leq T-1\}}\right]\right) = P(K=k) = \frac{1}{T}$$

- The expectation of $\boldsymbol{m}$ is:

$$\mathbb{E}[\mathbf{m}] = \left[\frac{1}{T}, \frac{2}{T}, \ldots, \frac{T-1}{T}\right]$$

Randomized Truncated BPTT

- To properly rescale the truncated gradient for unbiased estimation of the true gradient, we define $c_i = \frac{T}{i}$

$$\mathbb{E}\left[\frac{\mathrm{d}h_T}{\mathrm{d}w_\mathrm{h}}\right] = \mathbb{E}\left[\frac{\partial f(x_T, h_{T-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{T-1} c_i \mathbf{m}_i \left(\prod_{j=i+1}^{T} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}}\right) \frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}\right]$$

$$= \frac{\partial f(x_T, h_{T-1}, w_\mathrm{h})}{\partial w_\mathrm{h}} + \sum_{i=1}^{T-1} \mathbb{E}\left[c_i \mathbf{m}_i\right] \left(\prod_{j=i+1}^{T} \frac{\partial f(x_j, h_{j-1}, w_\mathrm{h})}{\partial h_{j-1}}\right) \frac{\partial f(x_i, h_{i-1}, w_\mathrm{h})}{\partial w_\mathrm{h}}$$

$$= \frac{\mathrm{d}h_T}{\mathrm{d}w_\mathrm{h}}$$

References

[1] https://www.auroria.io/language-models-are-few-shot-learners-analyzing-gpt-3-meta-learning-nlp/

[2] https://www.google.com/finance/quote/NVDA:NASDAQ

[3] https://github.com/chirag2796/RNN-LSTM-for-Text-Generation

[4] https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[5] Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions."

[6] xiandong79.github.io

[7] https://imerit.net/blog/using-neural-networks-for-video-classification-blog-all-pbm/

[8] https://www.sciencedirect.com/science/article/pii/S0959438818302009

[9] Werbos, Paul J.. “Backpropagation Through Time: What It Does and How to Do It.” *Proc. IEEE* 78 (1990): 1550-1560.

[10] Truncation: Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state network” approach.

[11] Randomized Truncation: Tallec, C., & Ollivier, Y. (2017). Unbiasing truncated backpropagation through time. ArXiv:1705.08209.

[12] https://d2l.ai/images/truncated-bptt.svg

[13] https://d2l.ai/images/rnn-bptt.svg

Questions?