SEQUENCE MODELS

Dr Usman Zia

Motivation: Need for Sequential Modeling

❓ Why do we need *Sequential Modeling*?

Motivation: Need for Sequential Modeling

Examples of Sequence data	Input Data	Output
• ***Speech Recognition***		*This is RNN*
• ***Machine Translation***	Hello, I am usman.	Hallo, ich bin Usman
• ***Language Modeling***	Recurrent neural ? based ? model	**network** **language**
• ***Named Entity Recognition***	David lives in Munich	**David** lives in **Munich** ***person*** ***location***
• ***Sentiment Classification***	There is nothing to like in this movie.	
• ***Video Activity Analysis***		Punching

Motivation: Need for Sequential Modeling

Inputs, Outputs can be different lengths in different examples

Example:

Sentence1: David lives in Munich

Sentence2: David Clark lives in Munich DE

Motivation: Need for Sequential Modeling

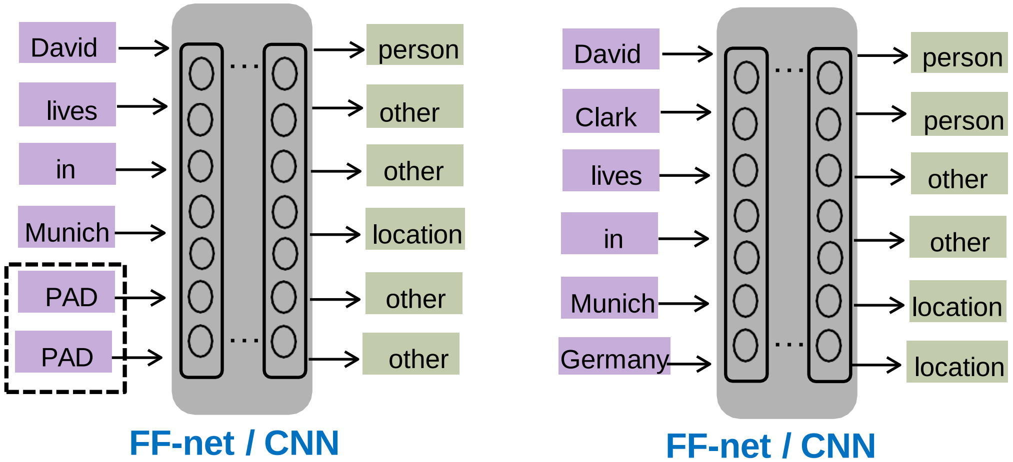
Inputs, Outputs can be different lengths in different examples

Example:

Sentence1: David lives in Munich

Sentence2: David Clark lives in Munich DE

Additional word 'PAD' i.e., padding



***FF-net: Feed-forward network**

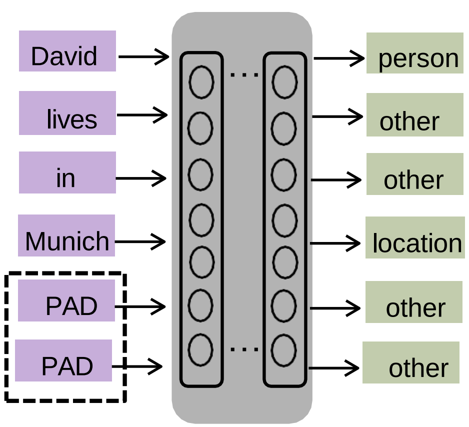
Motivation: Need for Sequential Modeling

Inputs, Outputs can be different lengths in different examples

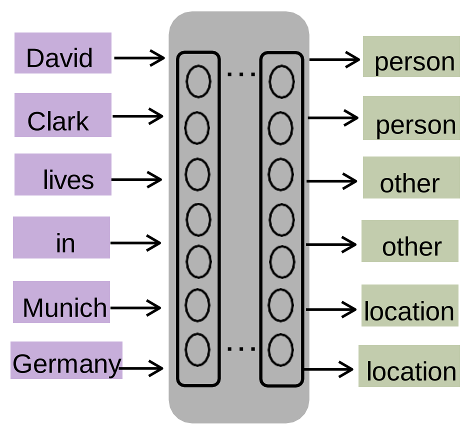
Example:

Sentence1: David lives in Munich

Sentence2: David Clark lives in Munich DE

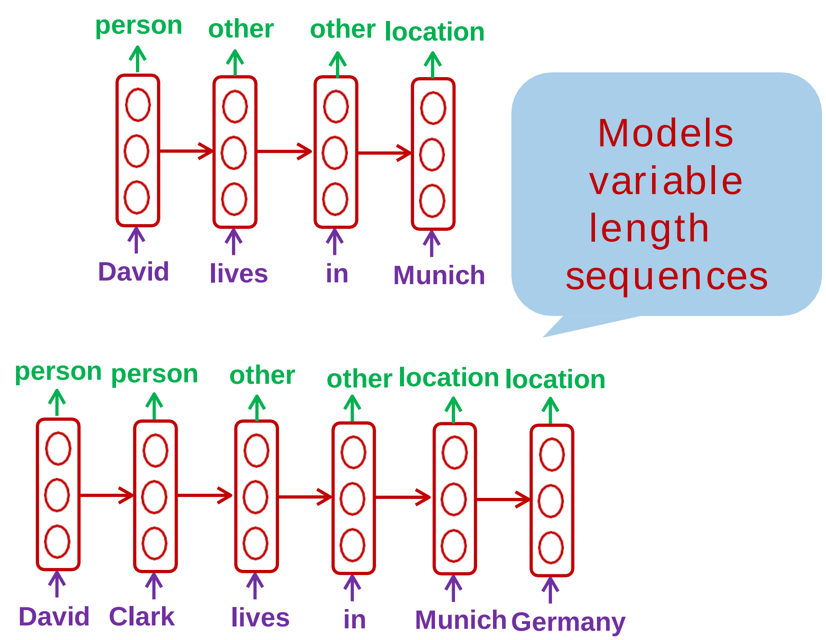


FF-net / CNN



FF-net / CNN

***FF-net: Feed-forward network**



Sequential model: RNN

Motivation: Need for Sequential Modeling

Share Features learned across different positions or time steps

Example:

Sentence1: *Market falls into bear territory* → *Trading/Marketing*

Sentence2: Bear falls into market territory → *UNK*

Same uni-gram statistics

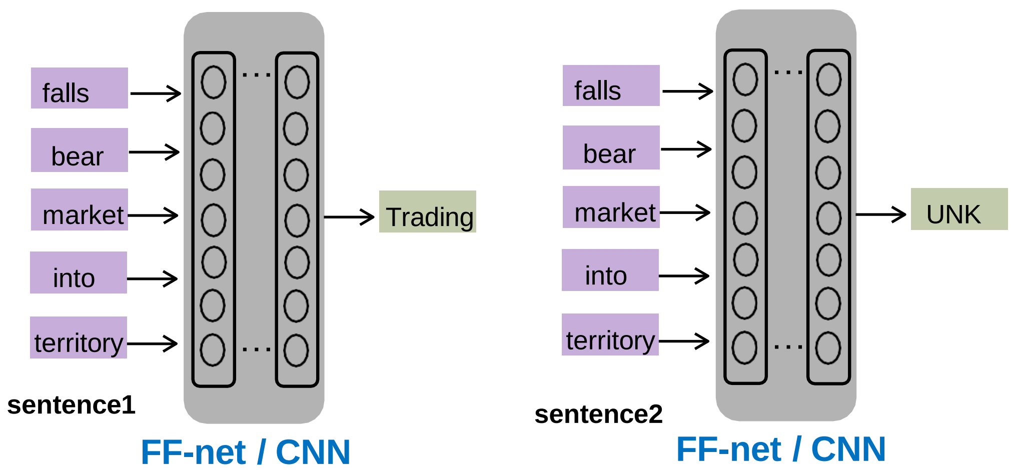
Motivation: Need for Sequential Modeling

Share Features learned across different positions or time steps

Example:

Sentence1: *Market falls into bear territory* → *Trading/Marketing*

Sentence2: *Bear falls into market territory* → *UNK*



No sequential or temporal modeling, i.e., order-less

Treats the two sentences the same

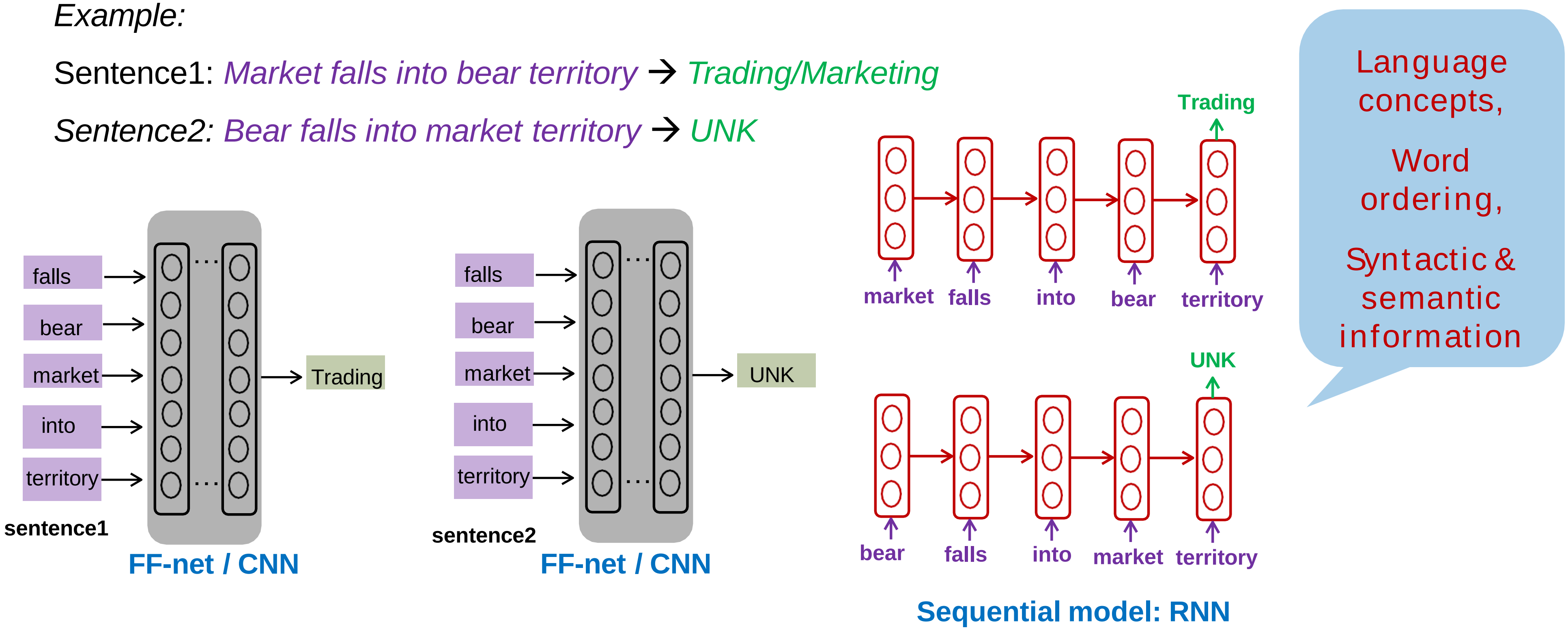
Motivation: Need for Sequential Modeling

Share Features learned across different positions or time steps

Example:

Sentence1: *Market falls into bear territory* → *Trading/Marketing*

Sentence2: Bear falls into market territory → *UNK*



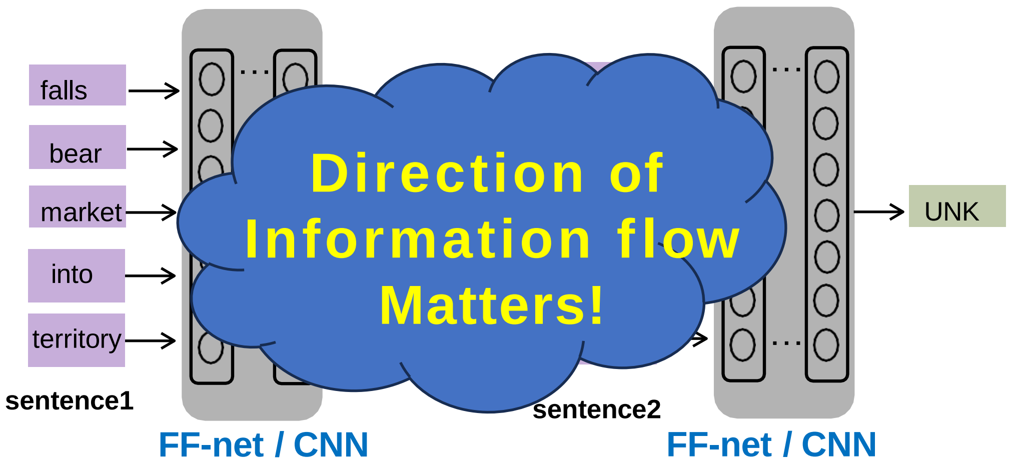
Motivation: Need for Sequential Modeling

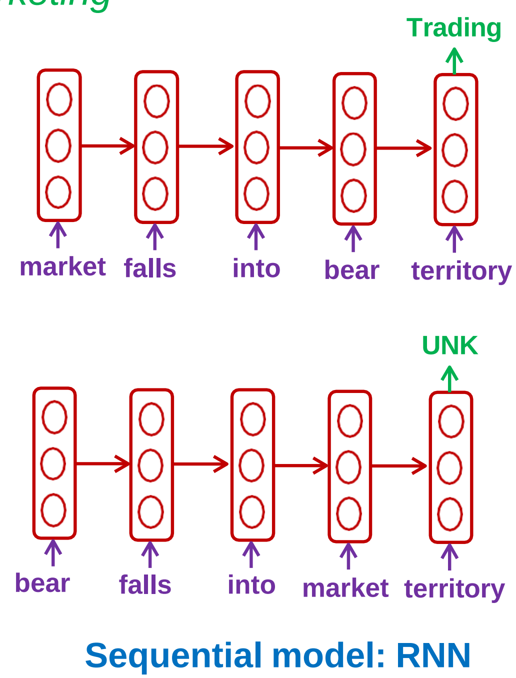
Share Features learned across different positions or time steps

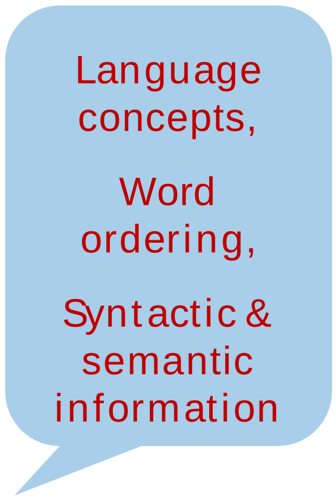
Example:

Sentence1: *Market falls into bear territory* → *Trading/Marketing*

Sentence2: Bear falls into market territory → *UNK*

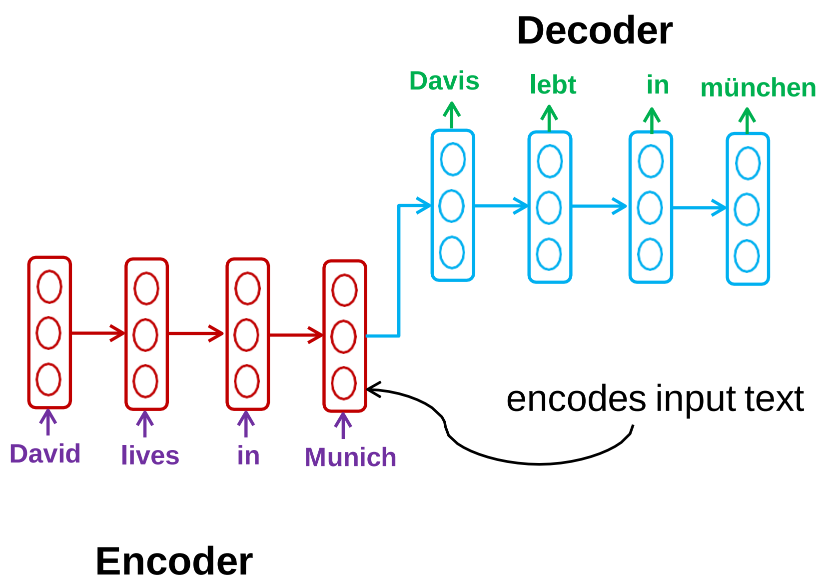






Motivation: Need for Sequential Modeling

Machine Translation: Different Input and Output sizes, incurring sequential patterns



Long Term and Short Dependencies

Short Term Dependencies

→ need recent information to perform the present task.

For example in a language model, predict the next word based on the previous ones.

"the clouds are in the **?***"* → *'sky'*

"the clouds are in the ***sky****"*

→ Easier to predict 'sky' given the context, i.e., *short term dependency*

Long Term Dependencies

→ Consider longer word sequence "I grew up in France……..............………………… I speak fluent ***French***."

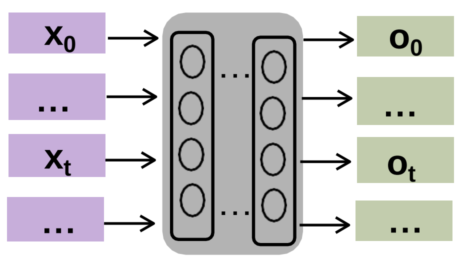
→ Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back.

Foundation of Recurrent Neural Networks

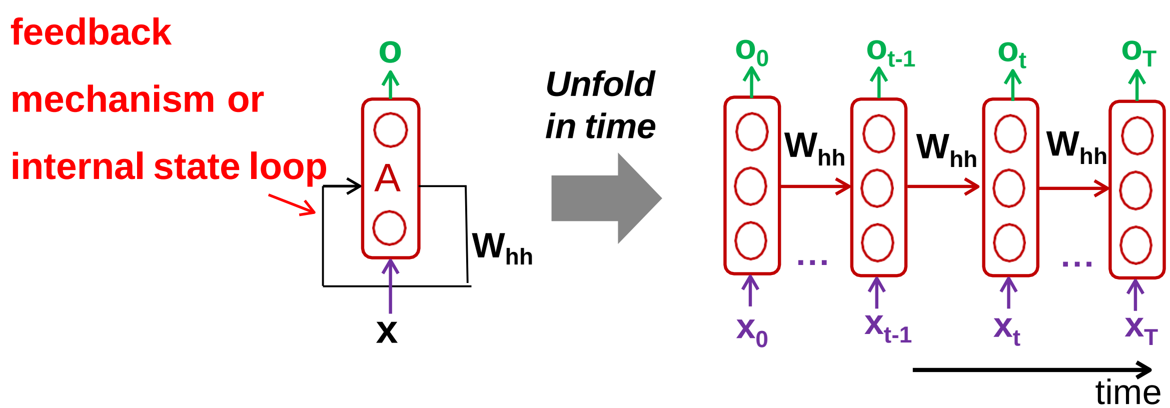
Goal

- model long term dependencies
- connect previous information to the present task
- model sequence of events with loops, allowing information to persist

Feed Forward NNets can not take time dependencies into account. Sequential data needs a Feedback Mechanism.

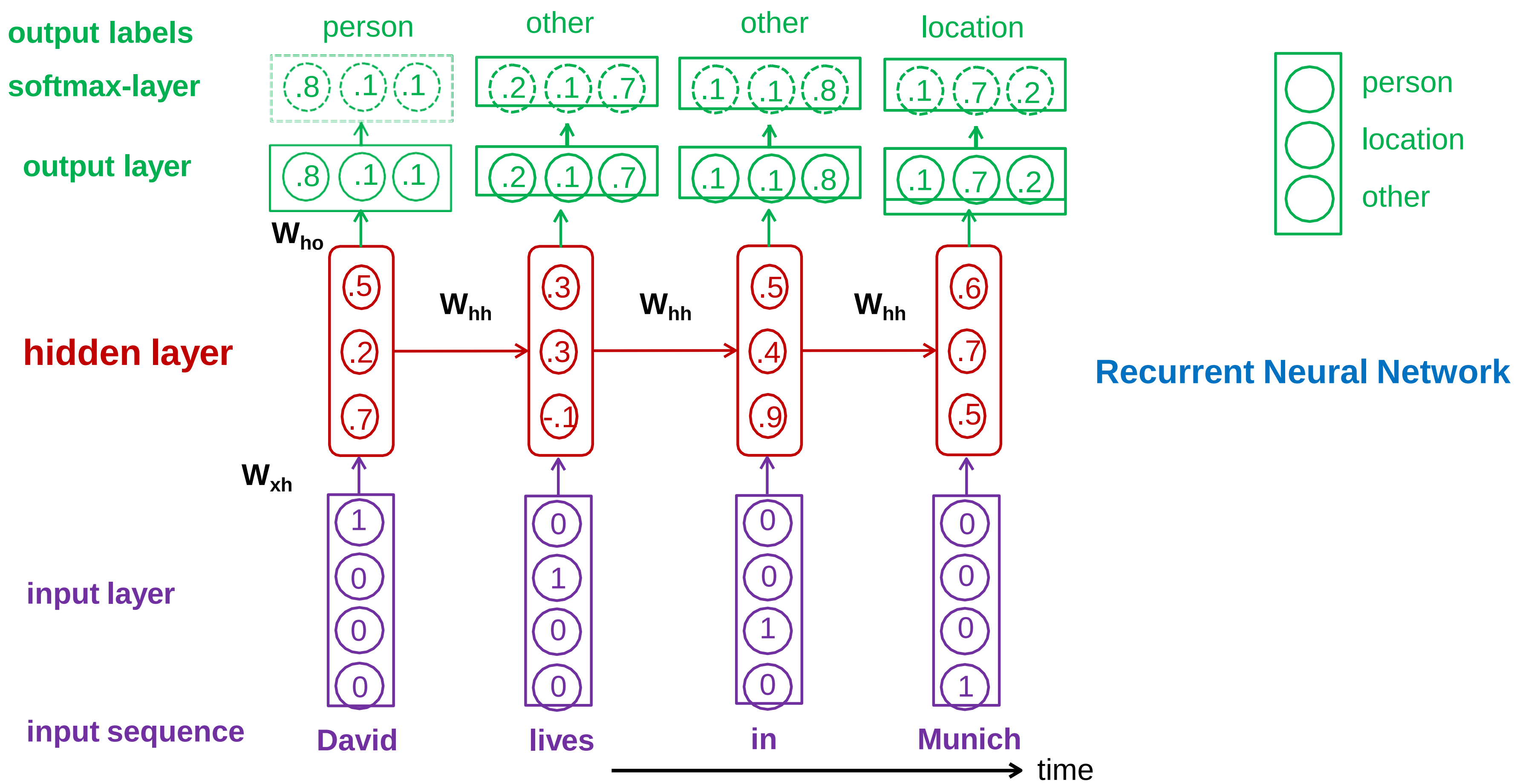


FF-net / CNN

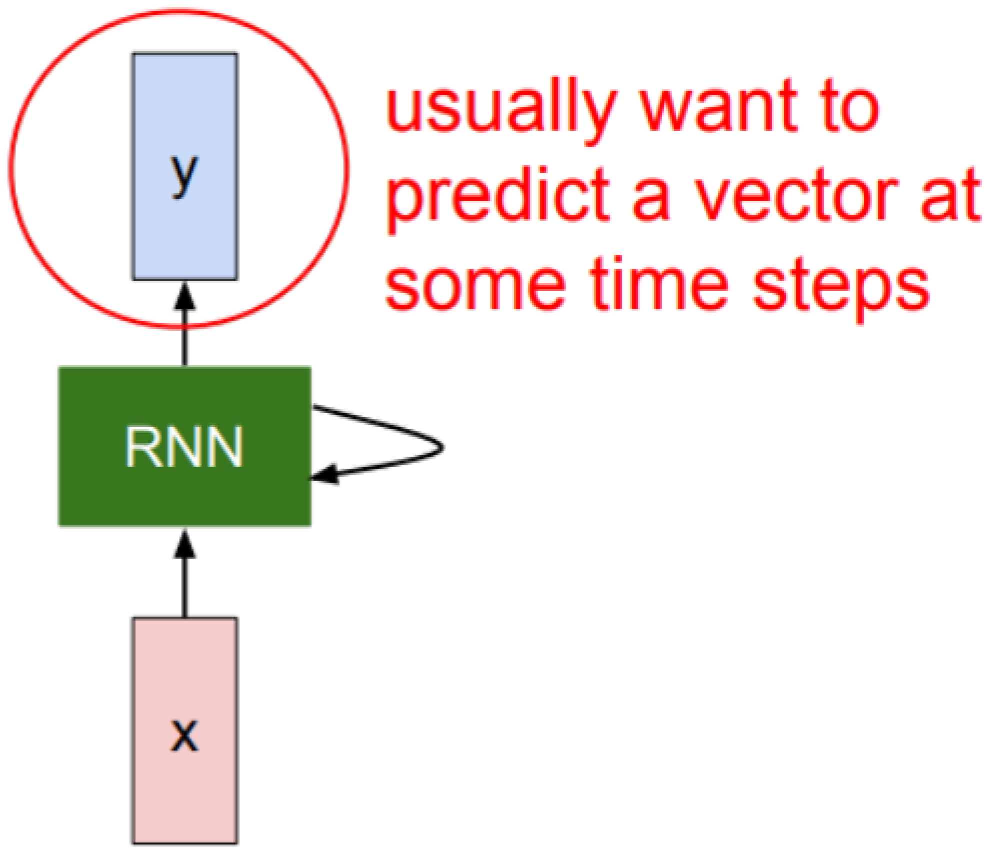


Recurrent Neural Network (RNN)

Foundation of Recurrent Neural Networks



Recurrent Neural Networks (RNNs)



Recurrent Neural Networks (RNNs)

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

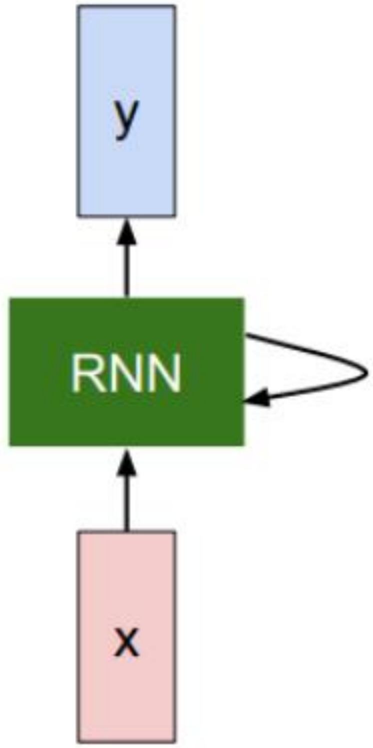
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

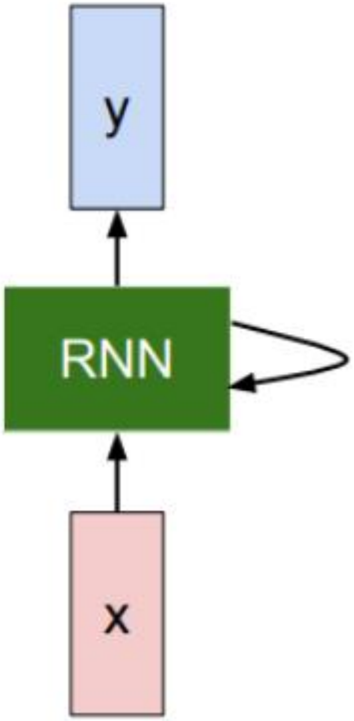
some function with parameters W



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

Recurrent Neural Networks (RNNs)

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



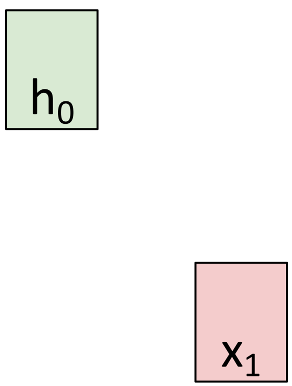
$$h_t = f_W(h_{t-1}, x_t)$$

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

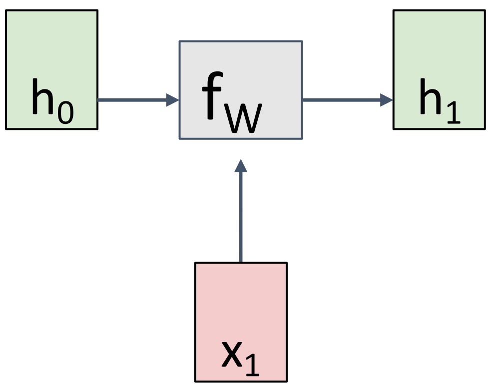
$$y_t = W_{hy}h_t$$

RNN Computational Graph

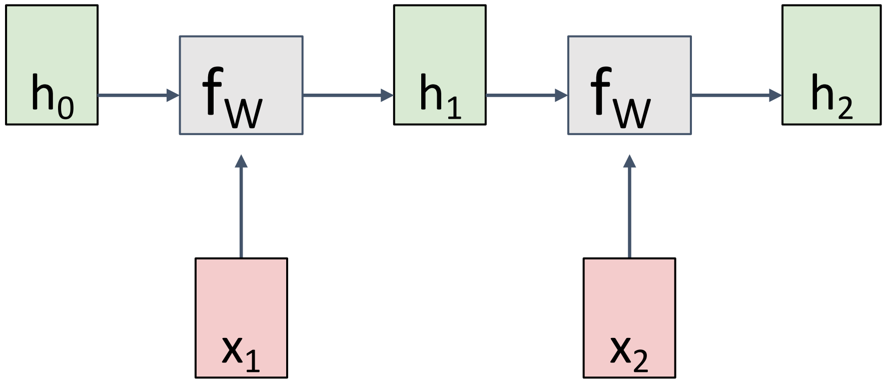
Initial hidden state
Either set to all 0,
Or learn it



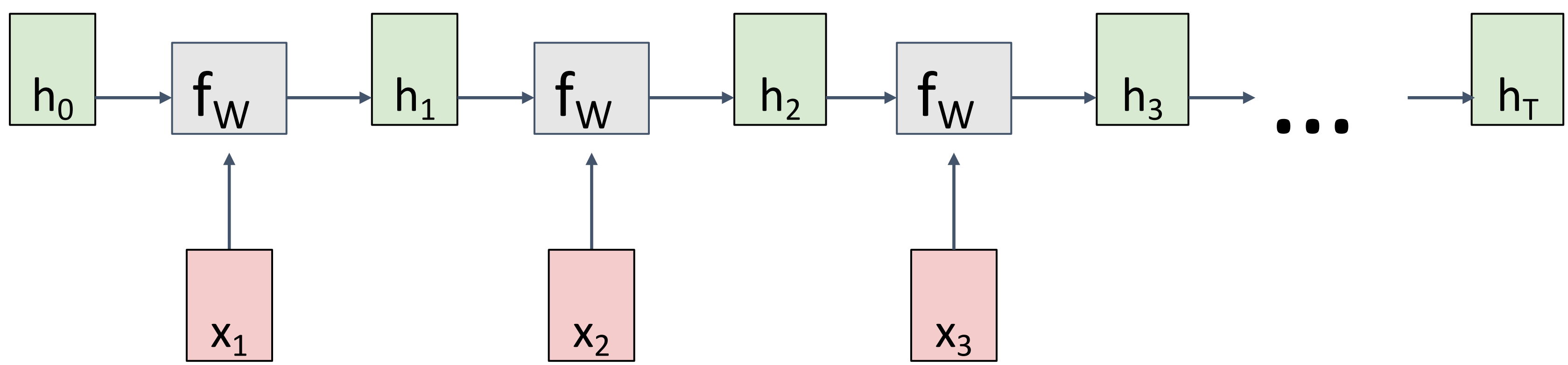
RNN Computational Graph



RNN Computational Graph

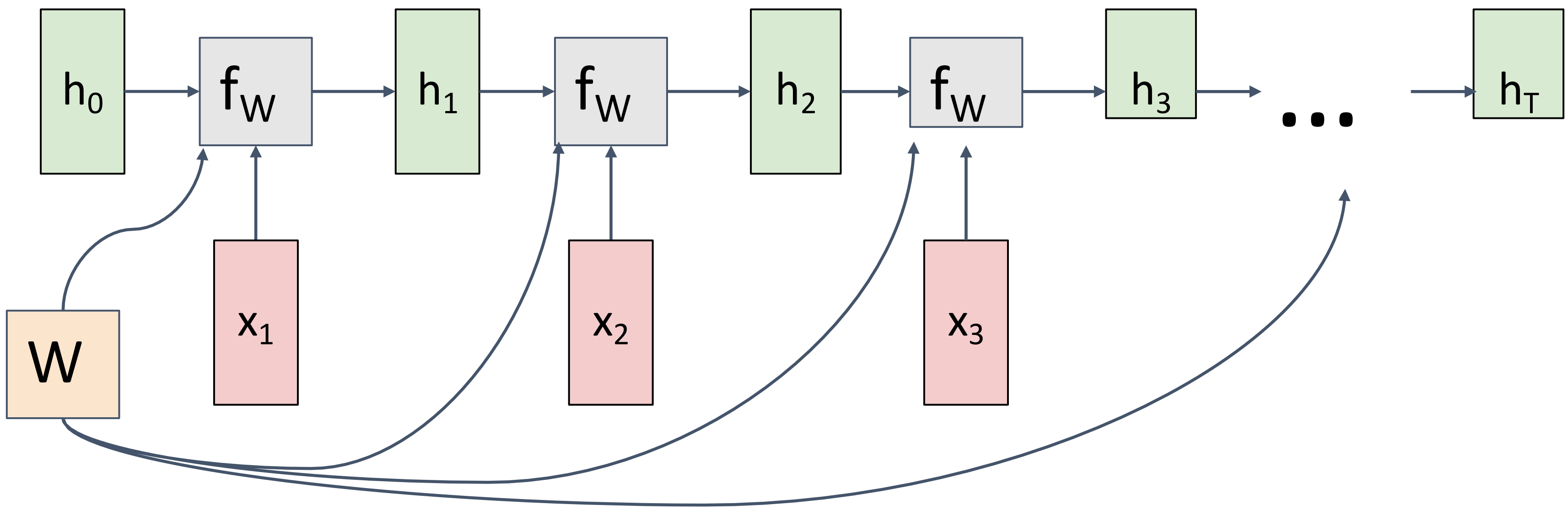


RNN Computational Graph

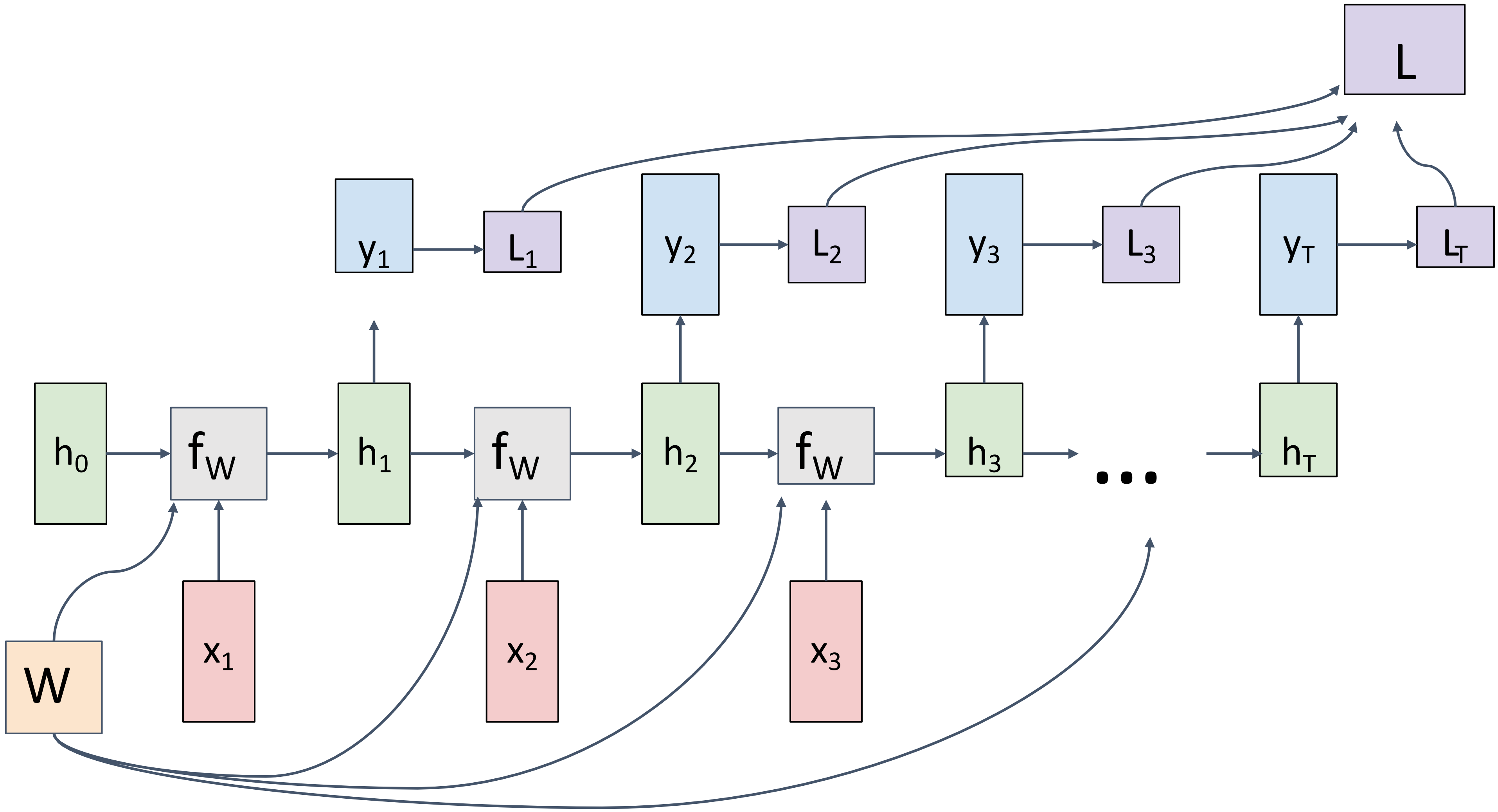


RNN Computational Graph

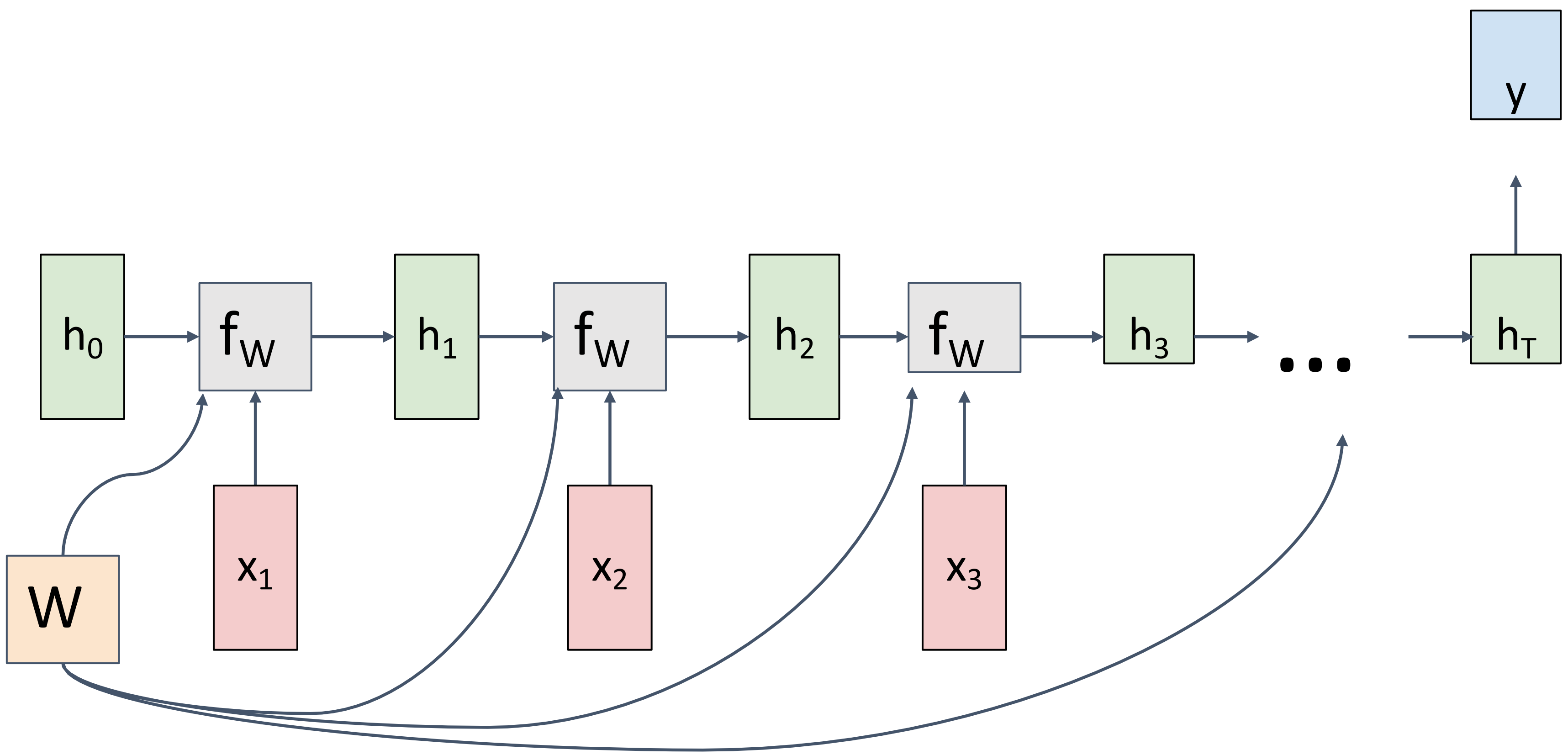
Re-use the same weight matrix at every time-step



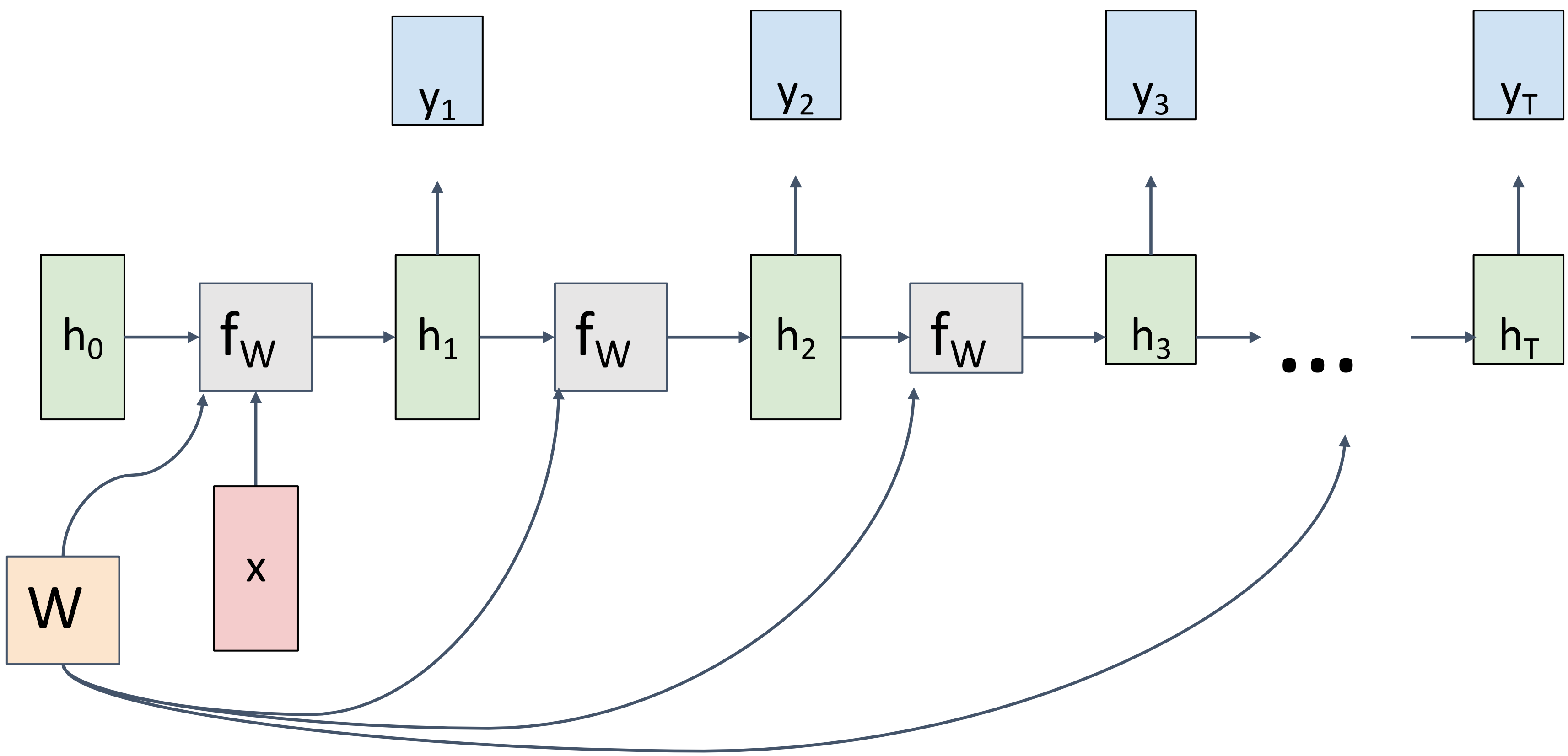
RNN Computational Graph (Many to Many)



RNN Computational Graph (Many to One)

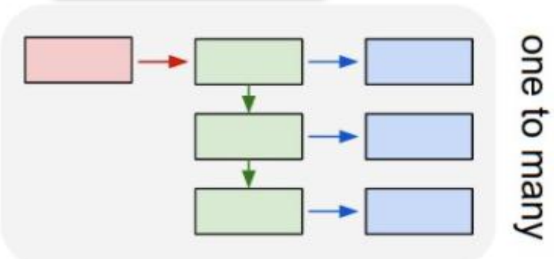
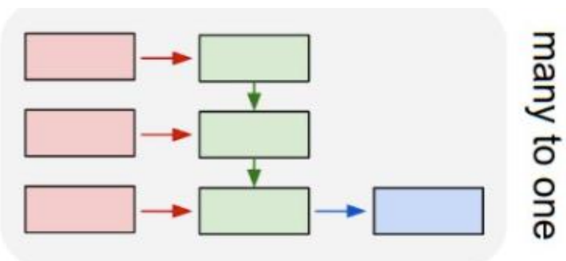
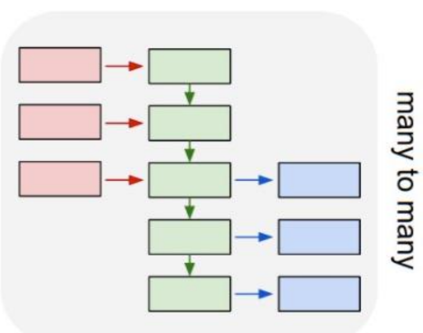


RNN Computational Graph (One to Many)



Recurrent Neural Networks (RNNs)

- Use cases for RNNs

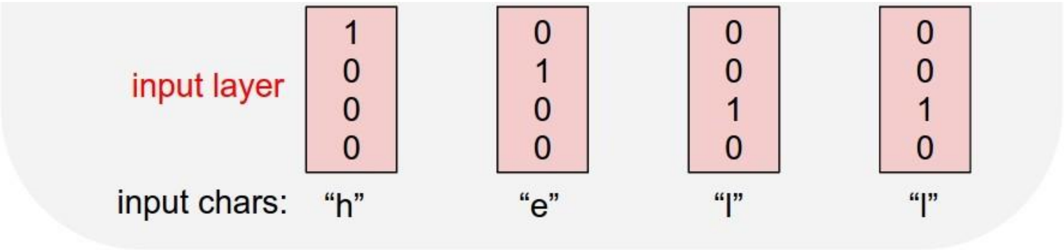
RNN	Application	Input	Output
	Image Captioning		A person riding a motorbike on dirt road
	Sentiment Analysis	Awesome movie. Highly recommended.	Positive
	Machine Translation Video Classification	Happy Eid	عيد مبارک

Example: Language Modeling

Given characters 1, 2, …, t,
model predicts character t

Training sequence: ”hello”

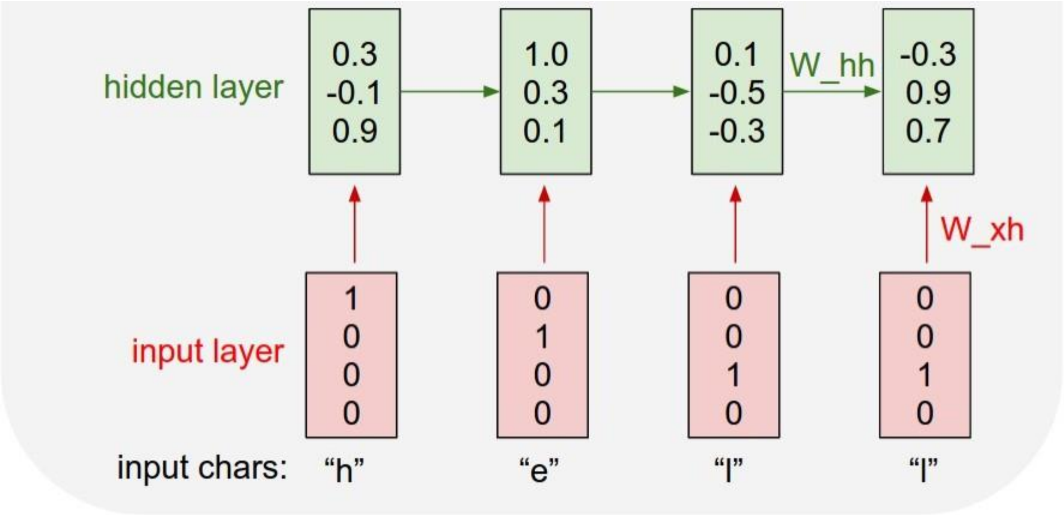
Vocabulary: [h, e, l, o]



Example: Language Modeling

Given characters 1, 2, …, t,
model predicts character t

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



Training sequence: "hello"

Vocabulary: [h, e, l, o]

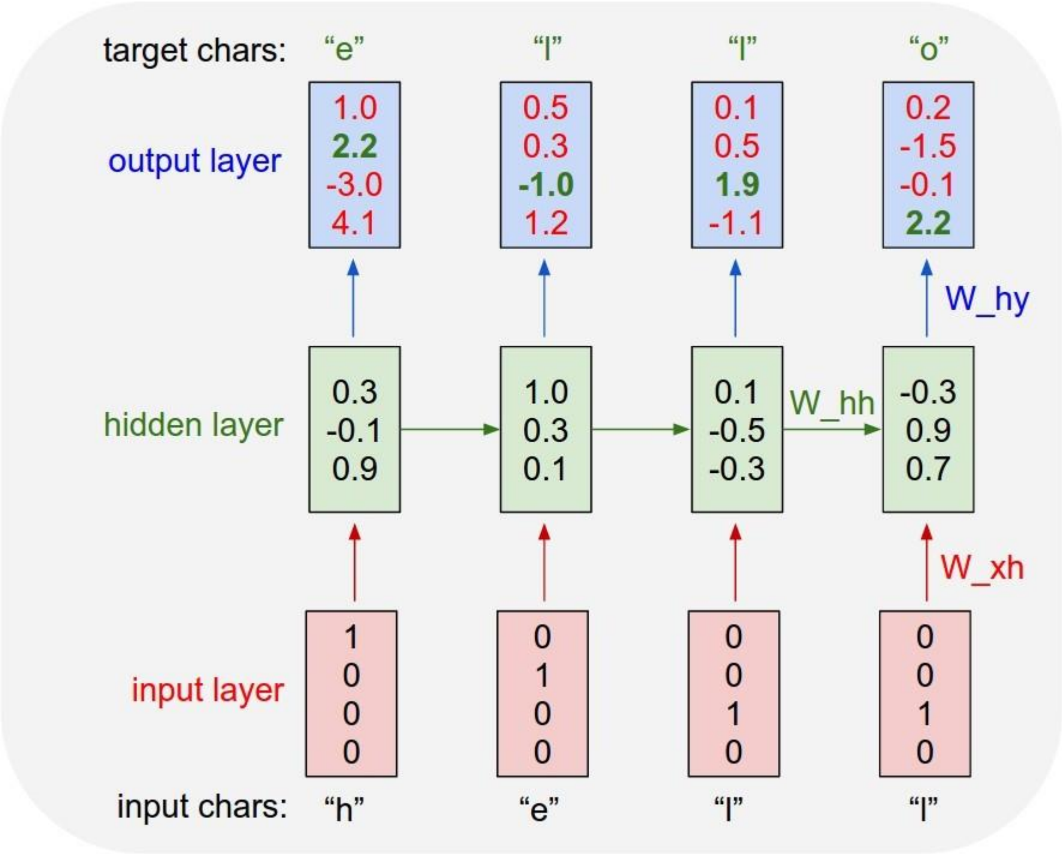
Example: Language Modeling

Given characters 1, 2, ..., t, model predicts character t

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



Example: Language Modeling

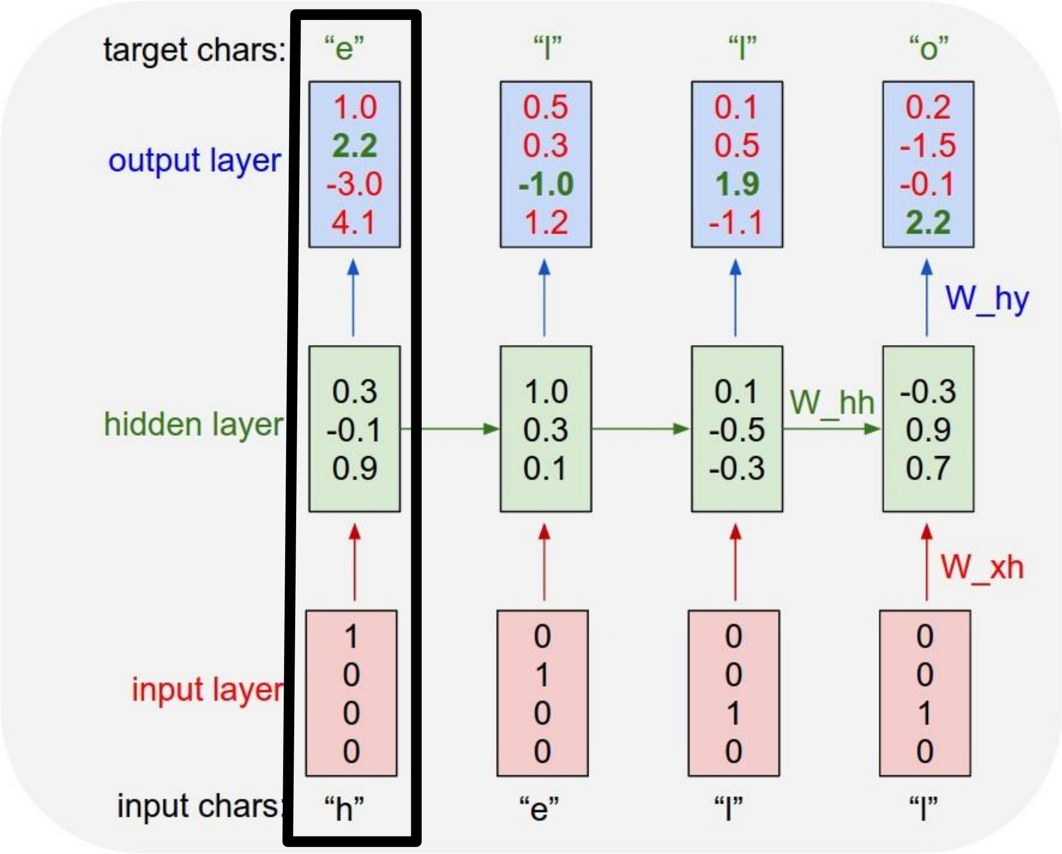
Given characters 1, 2, ..., t, model predicts character t

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

Given “h”, predict “e”



Example: Language Modeling

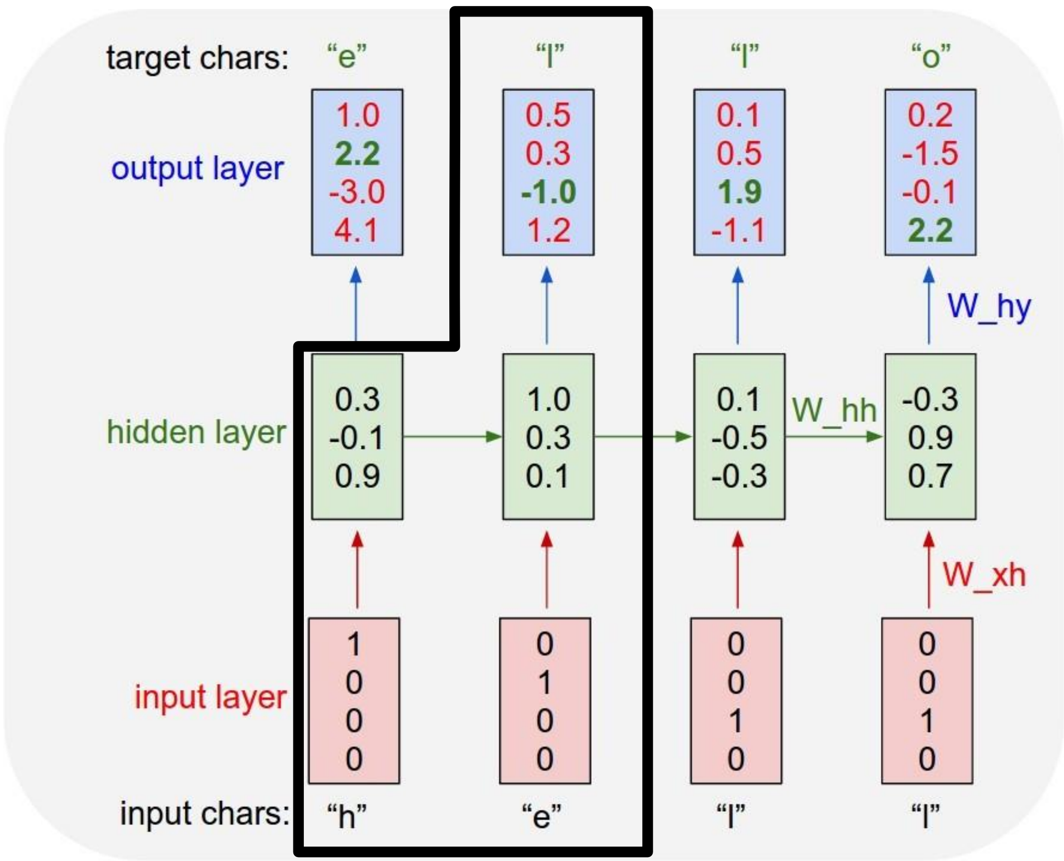
Given characters 1, 2, …, t, model predicts character t

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

Given “he”, predict “l”



Example: Language Modeling

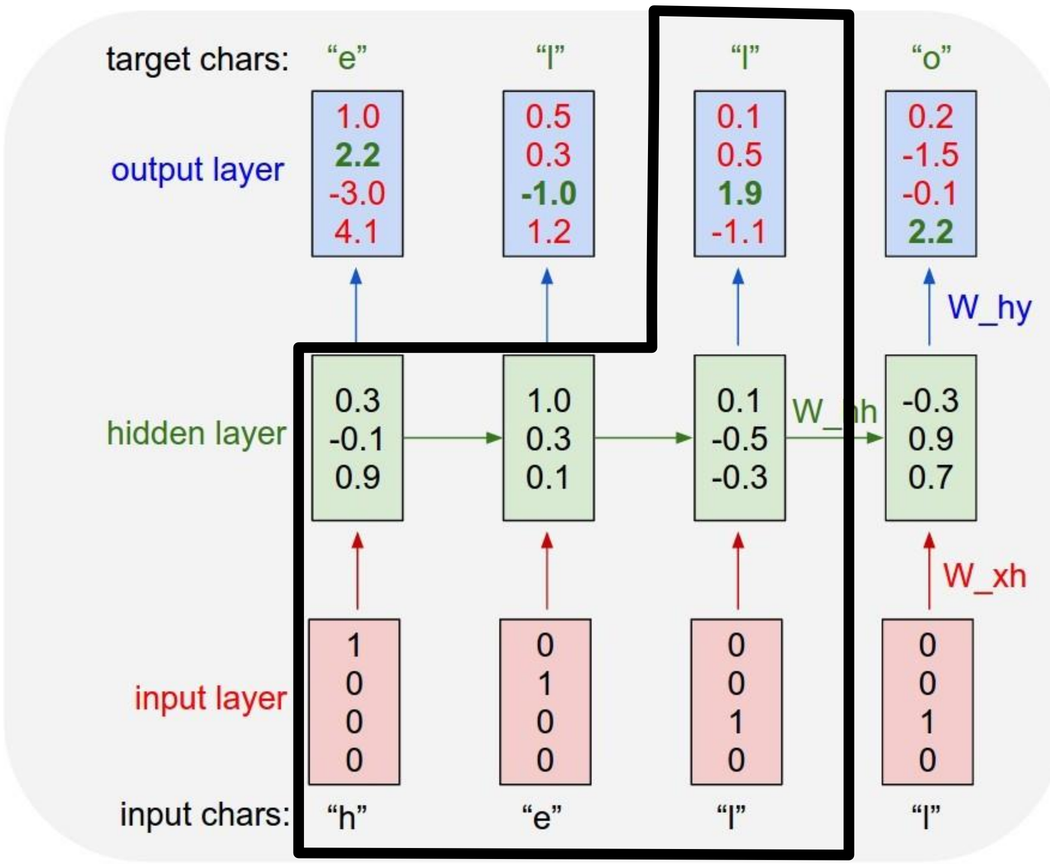
Given characters 1, 2, …, t, model predicts character t

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

Given “hel”, predict “l”



Example: Language Modeling

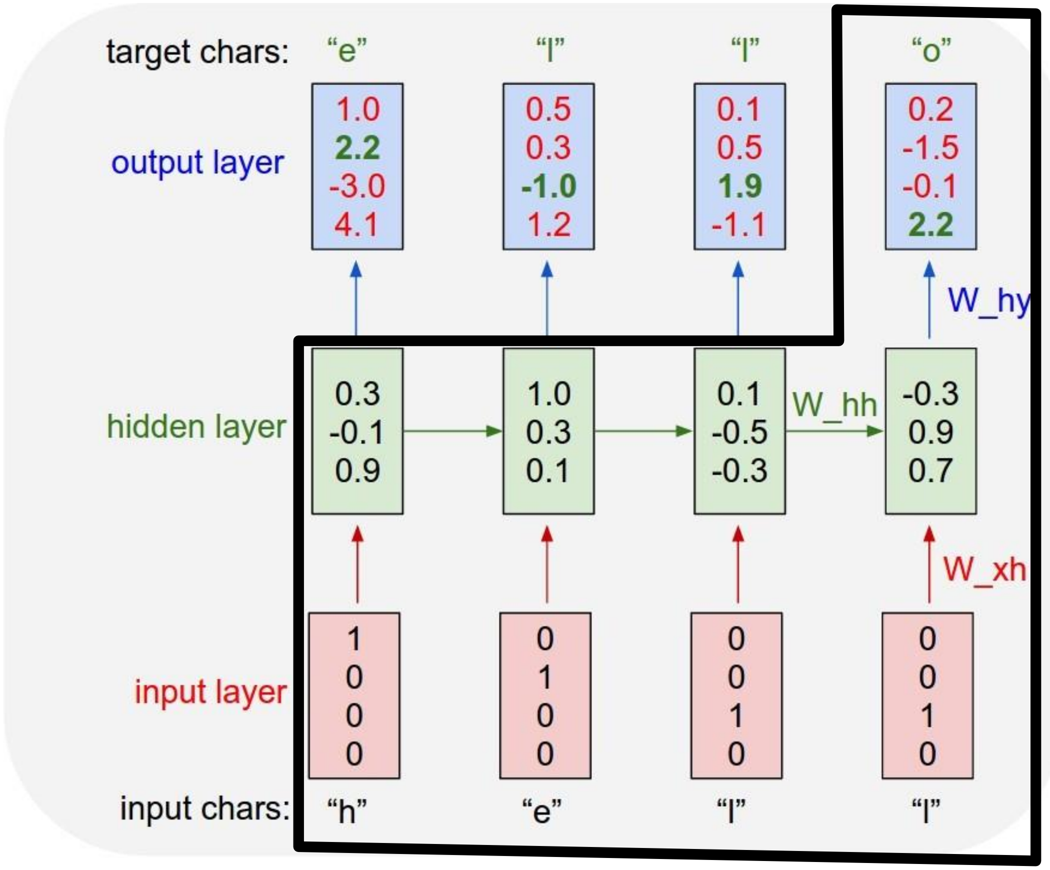
Given characters 1, 2, …, t, model predicts character t

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

Given “hell”, predict “o”

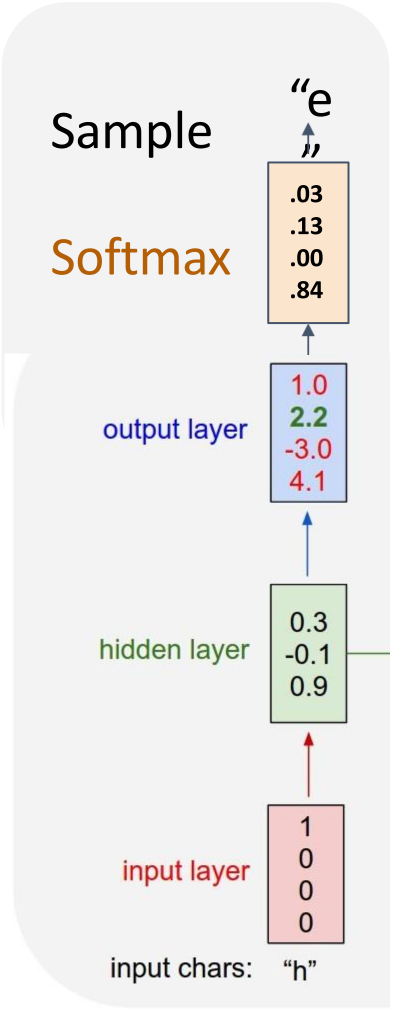


Example: Language Modeling

At test-time … ???

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

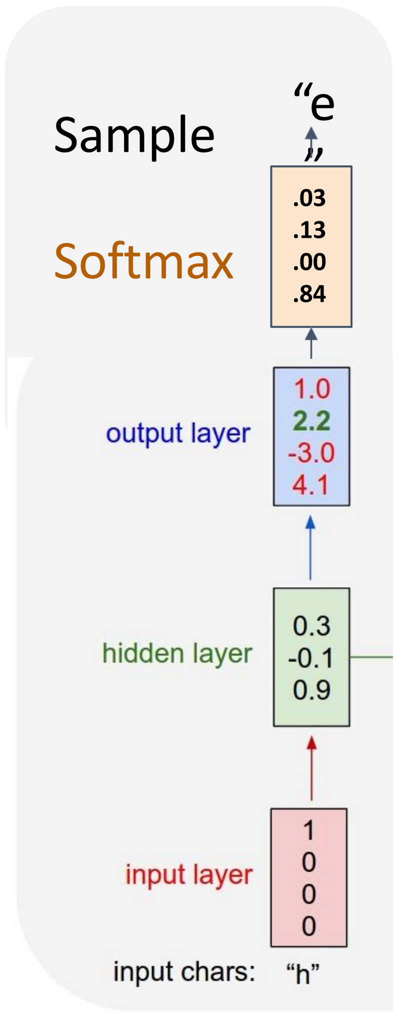


Example: Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

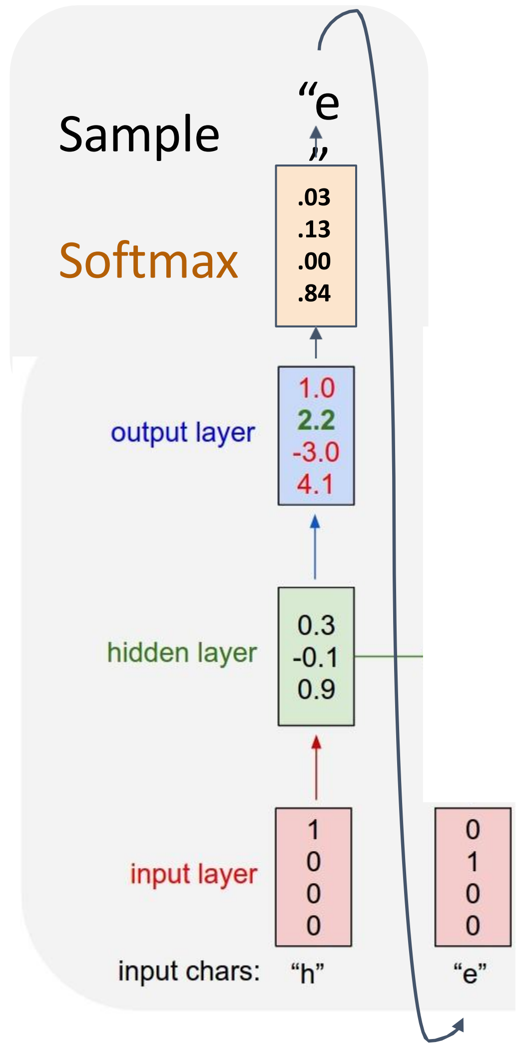


Example: Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

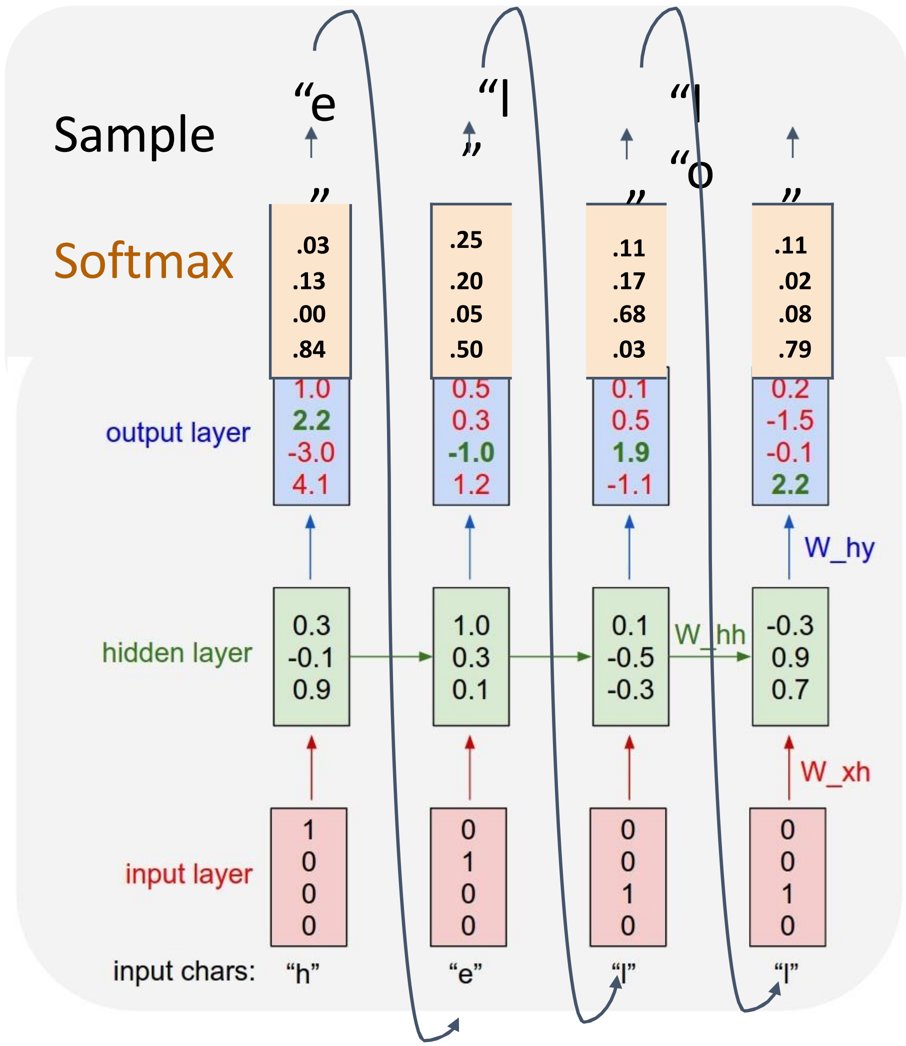


Example: Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

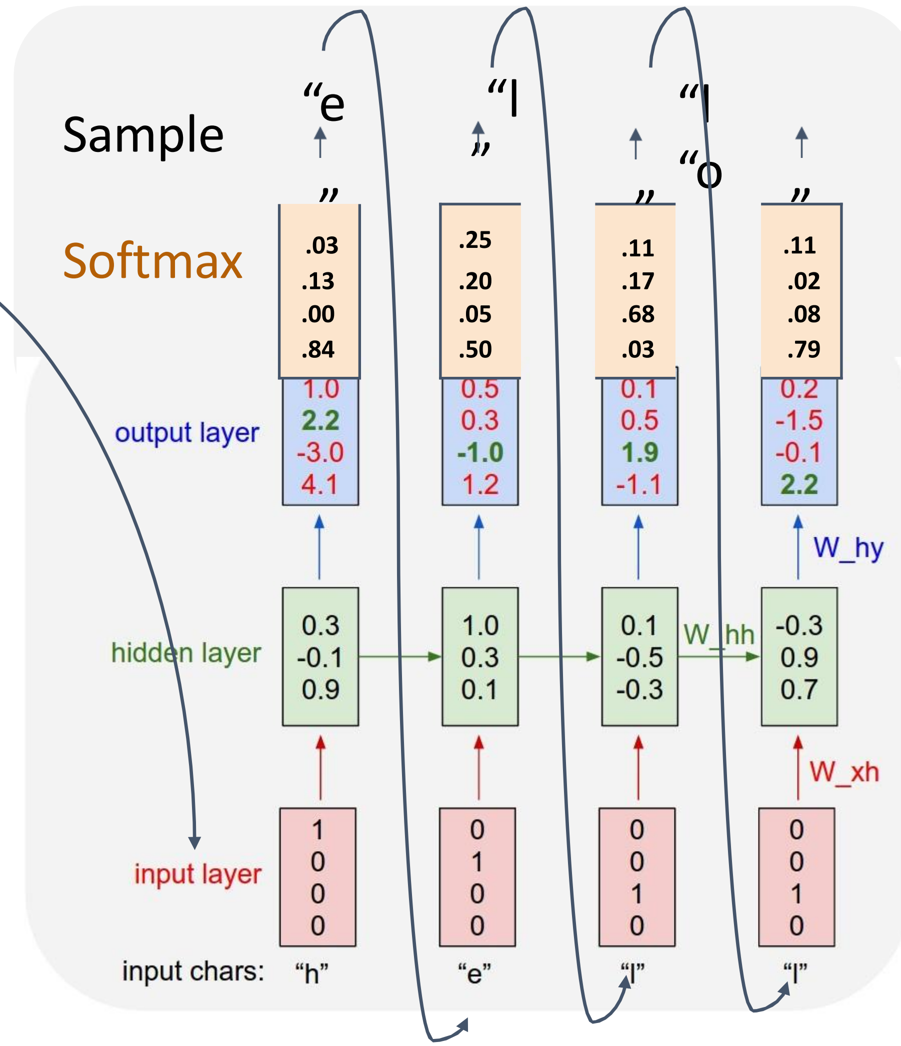


Example: Language Modeling

So far: encode inputs as **one-hot-vector**

$[w_{11}\ \ w_{12}\ \ w_{13}\ \ w_{14}]\ \ [1]\ \ [w_{11}]$
$[w_{21}\ w_{22}\ w_{23}\ w_{14}]\ \ [0]\ \ =\ \ [w_{21}]$
$[w_{31}\ w_{32}\ w_{33}\ w_{14}]\ [0]\ [w_{31}]$
$[0]$

Matrix multiply with a one-hot vector just extracts a column from the weight matrix. Often extract this into a separate **embedding** layer

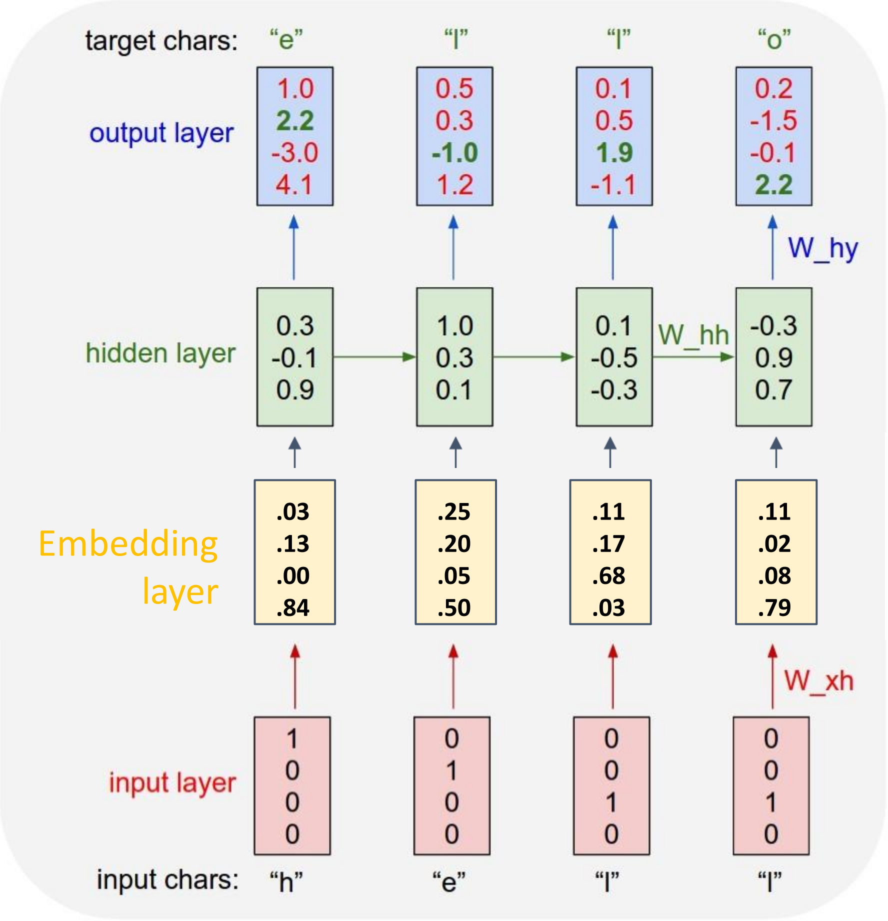


Example: Language Modeling

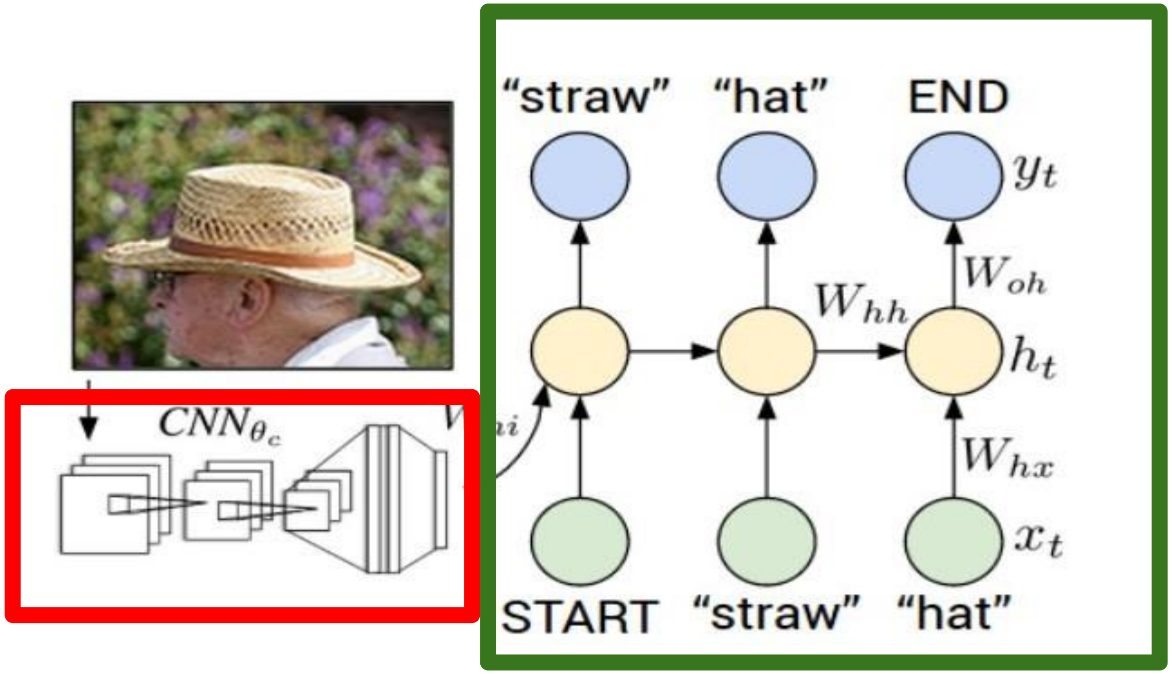
So far: encode inputs as **one-hot-vector**

$$
\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{14} \\ w_{31} & w_{32} & w_{33} & w_{14} \end{bmatrix}
\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
=
\begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix}
$$

Matrix multiply with a one-hot vector just extracts a column from the weight matrix. Often extract this into a separate **embedding** layer



Example: Image Captioning



Recurrent Neural Network

Convolutional Neural Network

Example: Image Captioning

Transfer learning: Take CNN trained on ImageNet, chop off last layer

Example: Image Captioning

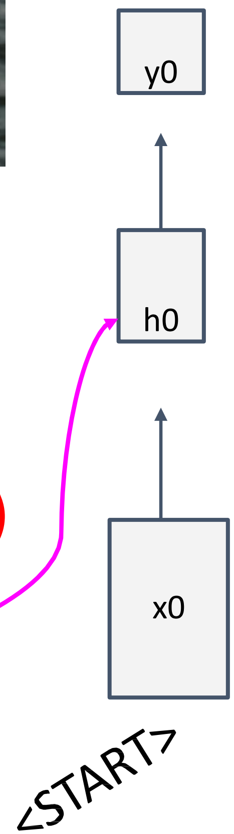
Example: Image Captioning

before:

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

now:

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



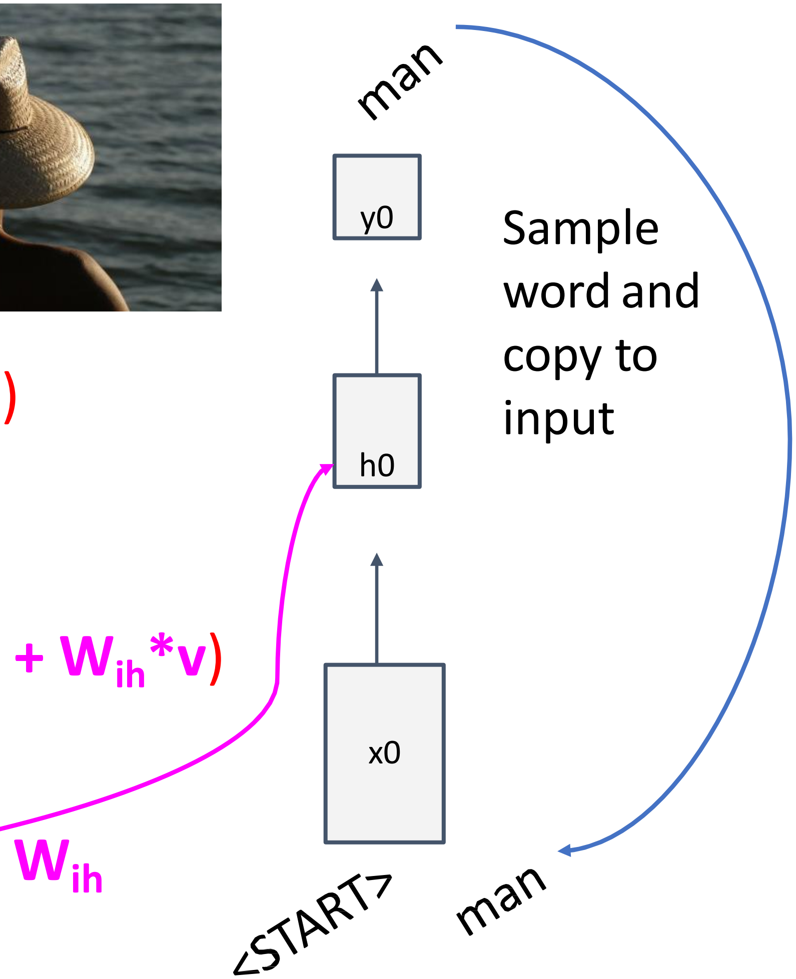
Example: Image Captioning

before:

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

now:

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



Sample word and copy to input

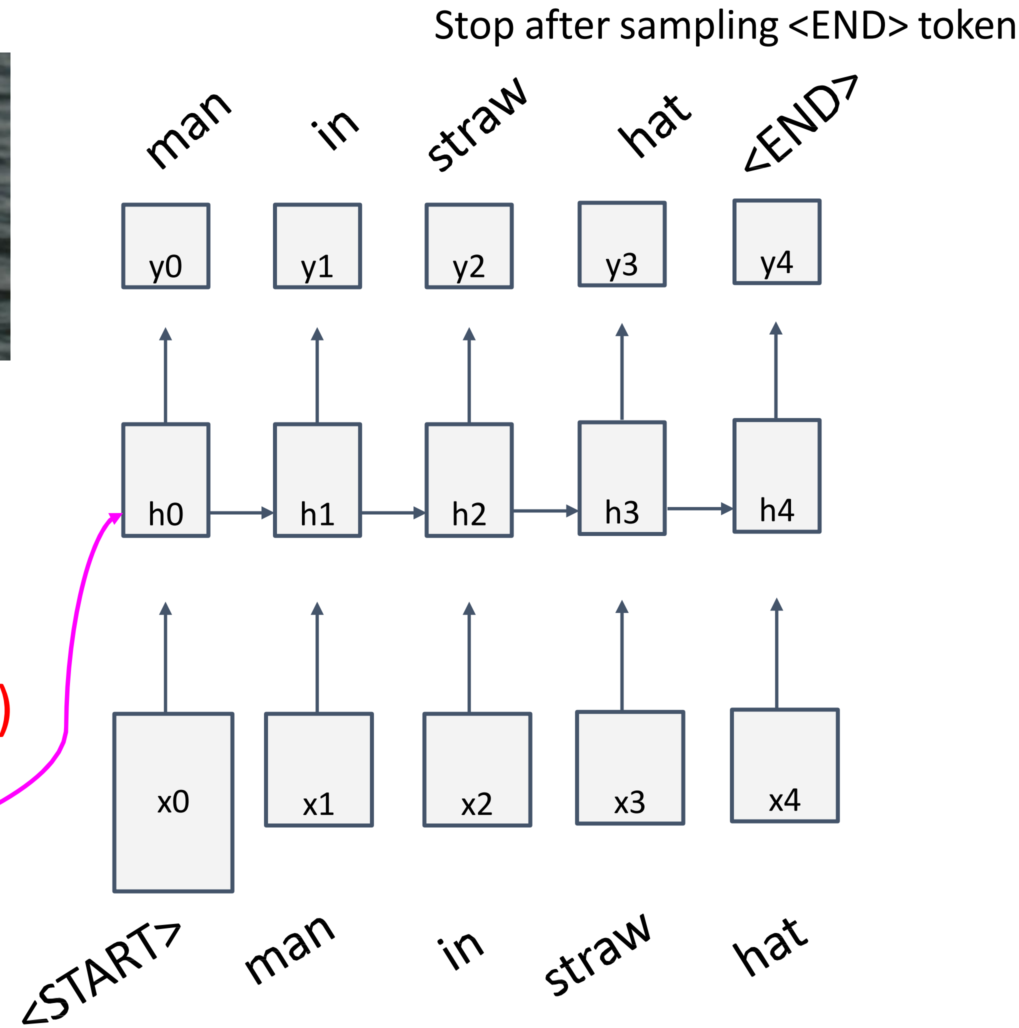
Example: Image Captioning

before:

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

now:

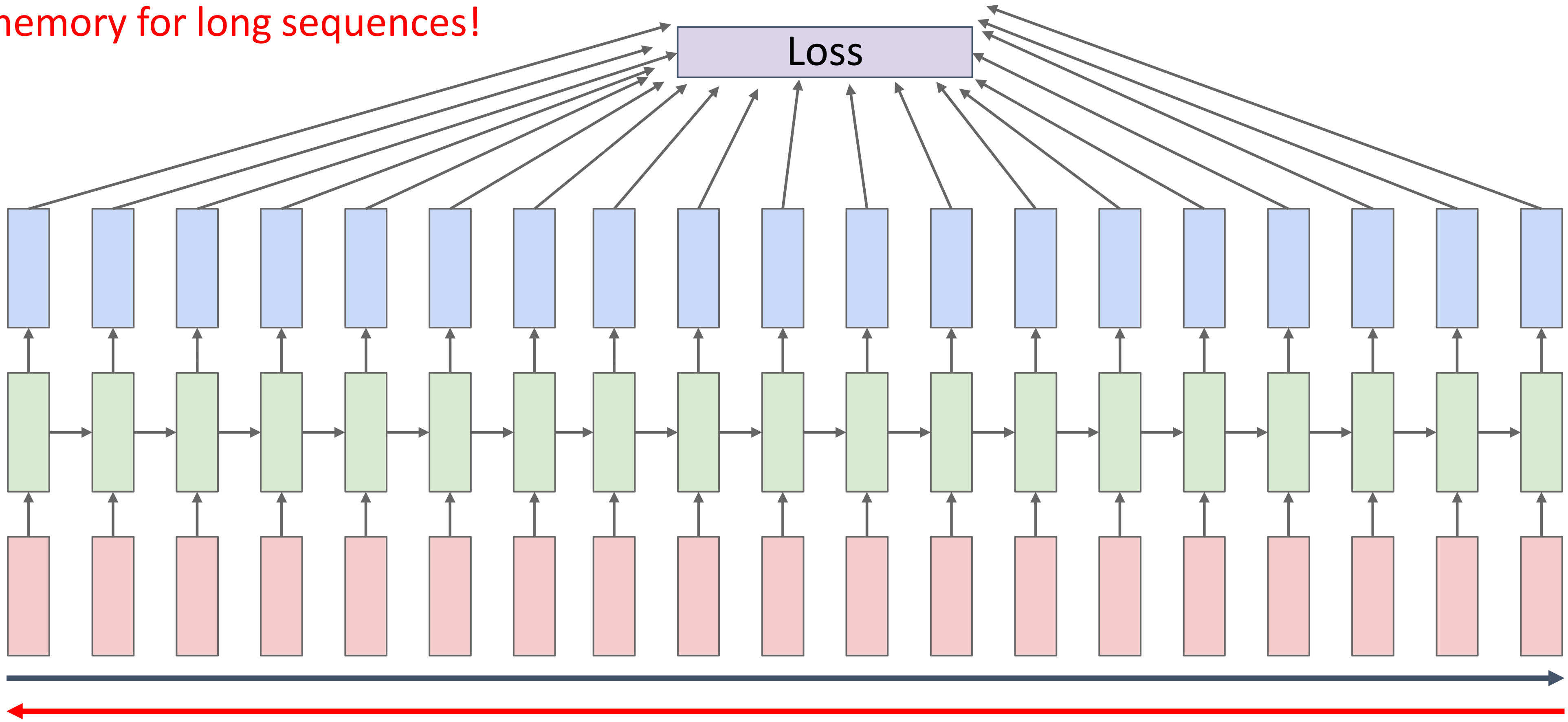
$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$$



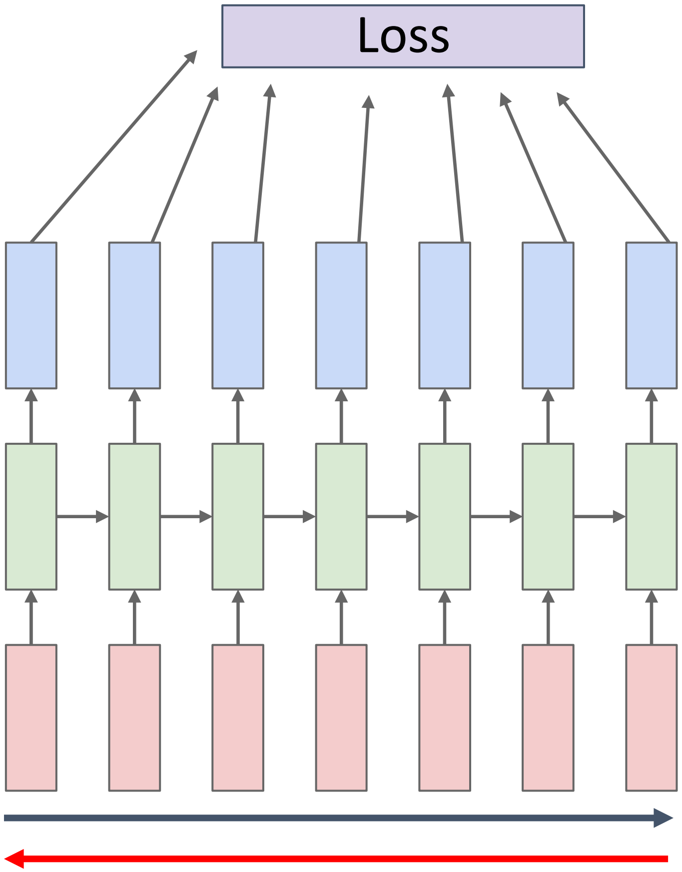
Backpropagation Through Time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

Problem: Takes a lot of memory for long sequences!

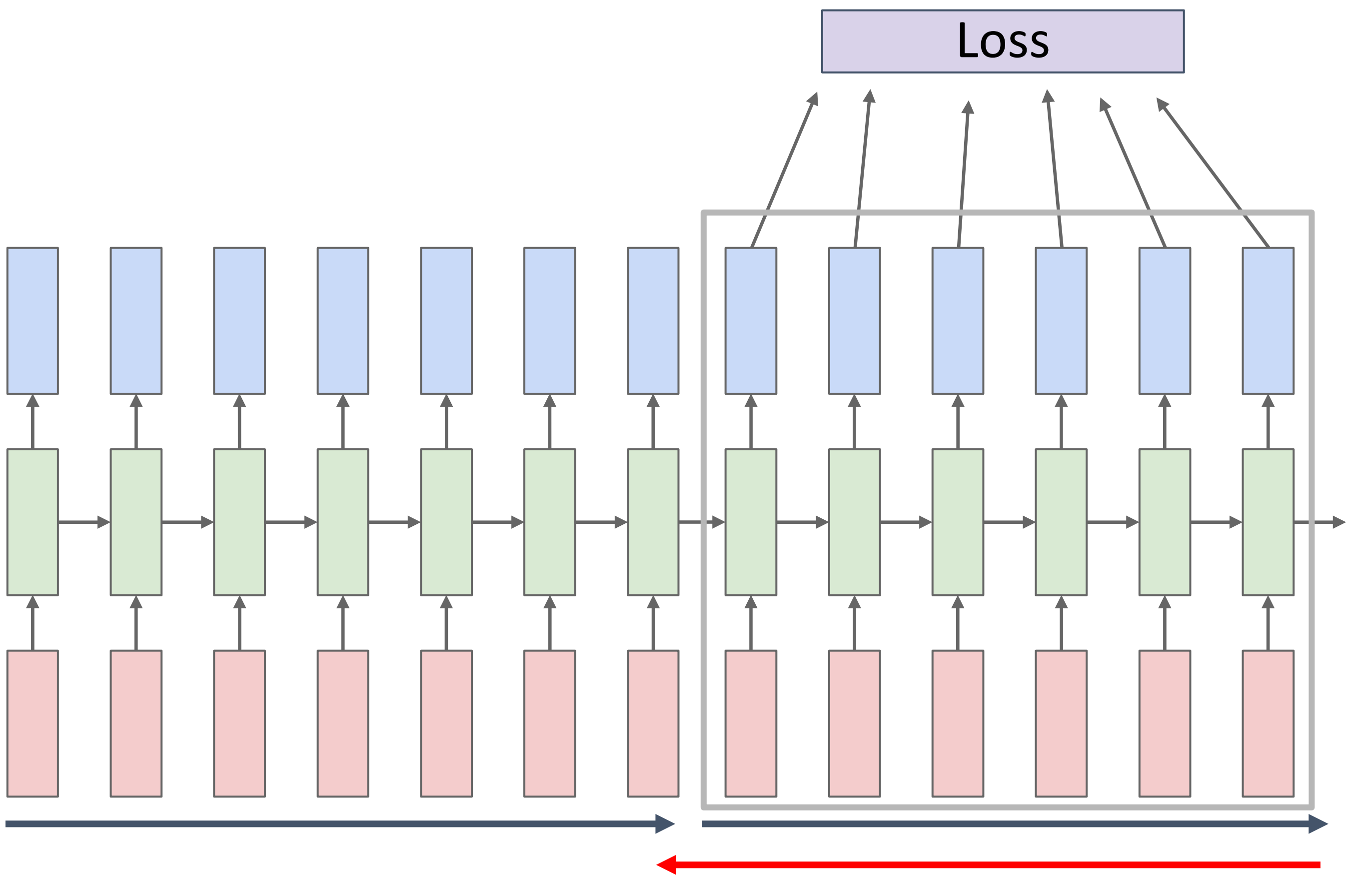


Truncated Backpropagation Through Time



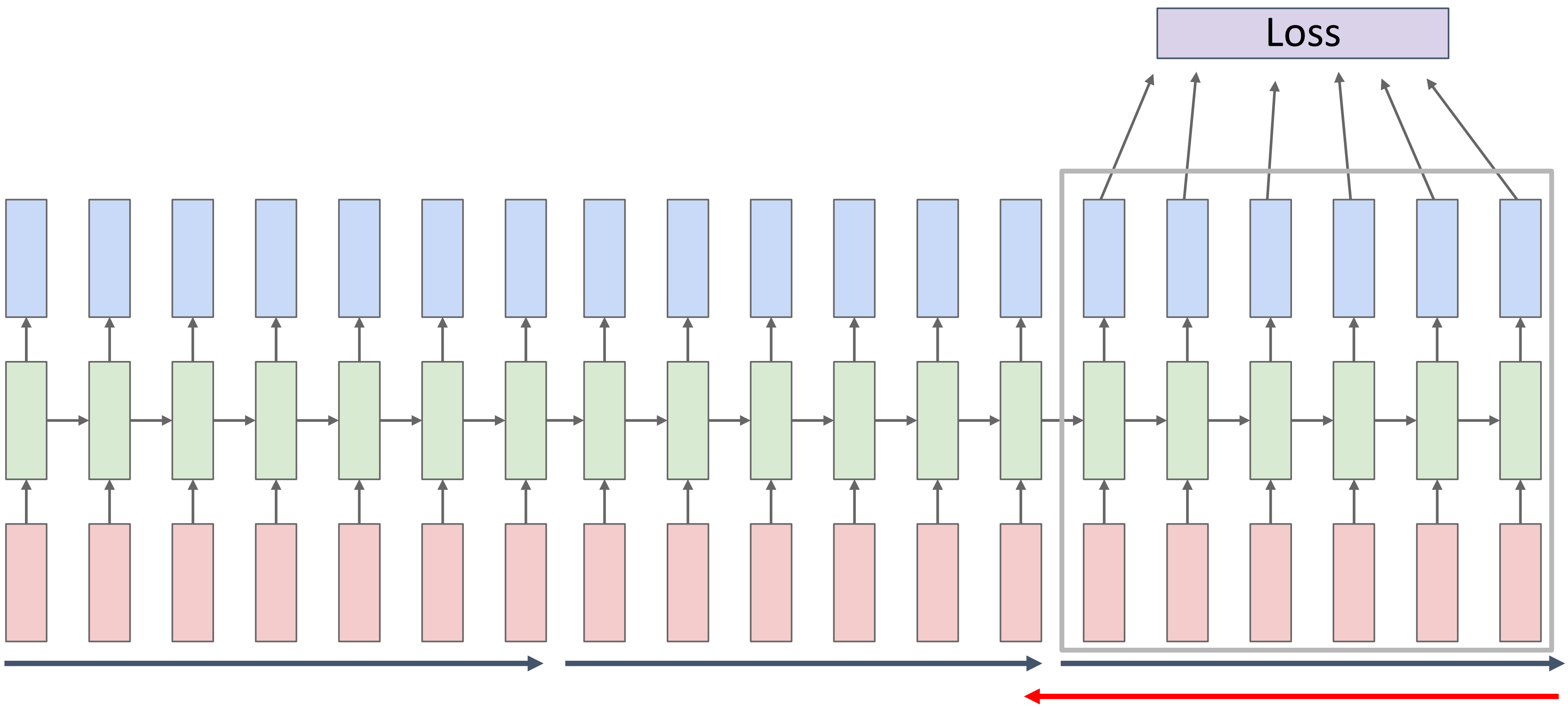
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation Through Time



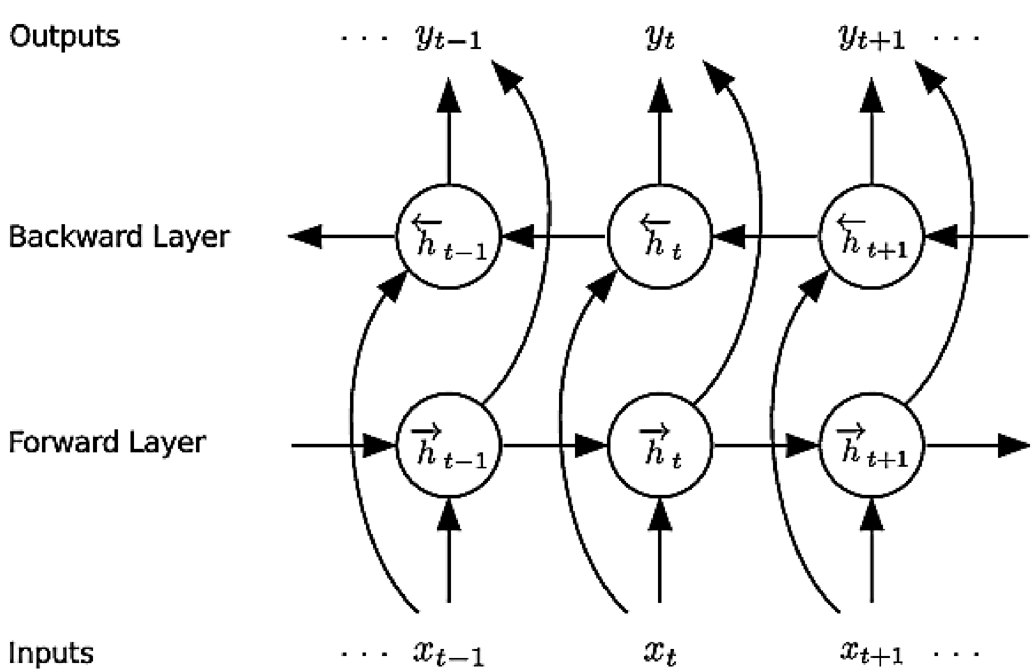
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation Through Time



Bidirectional RNNs

- ***Bidirectional RNNs*** incorporate both forward and backward passes through sequential data
  - The output may not only depend on the previous elements in the sequence, but also on future elements in the sequence
  - It resembles two RNNs stacked on top of each other



$$\overrightarrow{h}_t = \sigma(\overrightarrow{W}^{(hh)}\overrightarrow{h}_{t-1} + \overrightarrow{W}^{(hx)}x_t)$$

$$\overleftarrow{h}_t = \sigma(\overleftarrow{W}^{(hh)}\overleftarrow{h}_{t+1} + \overleftarrow{W}^{(hx)}x_t)$$

$$y_t = f(\,[\overrightarrow{h}_t;\overleftarrow{h}_t]\,)$$

Outputs both past and future elements

A recurrent neural language model

- How does this compare to n-gram models?

Improvements:

- Model size: O(V), not O(V^n)
- Sparsity (lack thereof)
- Sharing of representations across words
- Models long context

Remaining challenges:

A recurrent neural language model

- How does this compare to n-gram models?

Improvements:

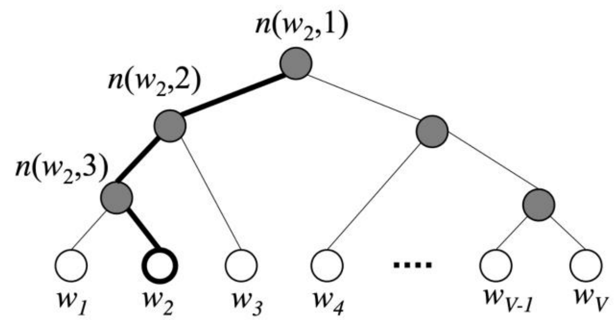
- Model size: O(V), not O(V^n)
- Sparsity (lack thereof)
- Sharing of representations across words
- Models long context

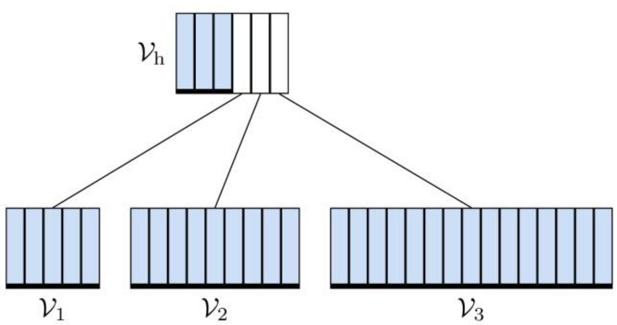
Remaining challenges:

- Softmax over large vocabulary
- High variance / overfitting
- Exploding and vanishing gradients

But what about that huge softmax over *V*?

- Same problem as word embeddings: don't want to score wrt entire vocab!
- Solutions:
  - Hierarchical softmax
  - Noise-contrastive estimation (NCE)
  - Adaptive softmax [Grave et al. 2017]





Problems learning RNNs

- In theory, should be able to propagate information over arbitrarily long contexts.
- In practice, RNNs suffer from **vanishing gradients** that decay to 0, or **exploding gradients** that increase towards infinity.

Figure: Goodfellow, Bengio and Courville. Deep Learning. MIT Press, 2016.

Problems learning RNNs

- In theory, should be able to propagate information over arbitrarily long contexts.
- In practice, RNNs suffer from **vanishing gradients** that decay to 0, or **exploding gradients** that increase towards infinity.
  - Exploding gradients mostly resolved by **gradient clipping**: thresholding gradient values, or rescaling them. Threshold/scale is a hyperparameter.

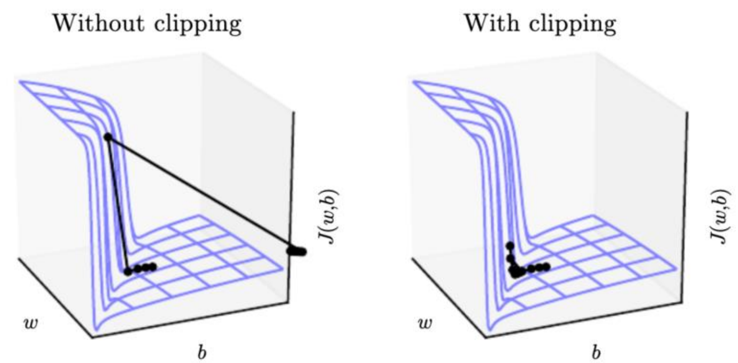


Figure: Goodfellow, Bengio and Courville. Deep Learning. MIT Press, 2016.

Problems learning RNNs

- In theory, should be able to propagate information over arbitrarily long contexts.
- In practice, RNNs suffer from **vanishing gradients** that decay to 0, or **exploding gradients** that increase towards infinity.
  - Exploding gradients mostly resolved by **gradient clipping**: thresholding gradient values, or rescaling them. Threshold/scale is a hyperparameter.
- Vanishing gradients mostly resolved by adding **gating** to the RNN composition function.

• Sigmoid activation function in RNN leads to this problem.
• Relu, in theory, avoids this problem but not in practice.