Precept 6: RNNs

Samyak Gupta

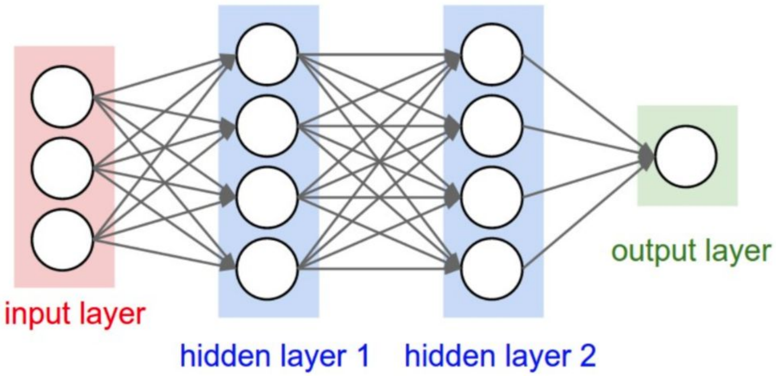
03/31/2023

Agenda

- Recurrent Neural Networks
- LSTMs and GRUs

Recap: Feed Forward Neural Networks (FFNs)

- The units are connected with no cycles
- The outputs from units in each layer are passed to units in the next higher layer
- No outputs are passed back to lower layers



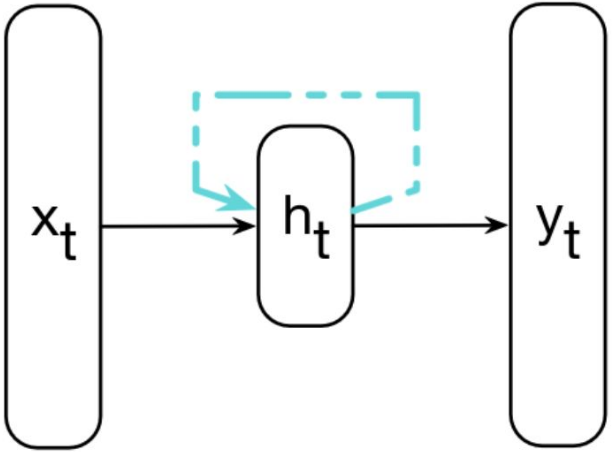
But FFNs are limited!

- Fixed input lengths
- Number of parameters scales with context window size
- Assume simultaneous access to entire window

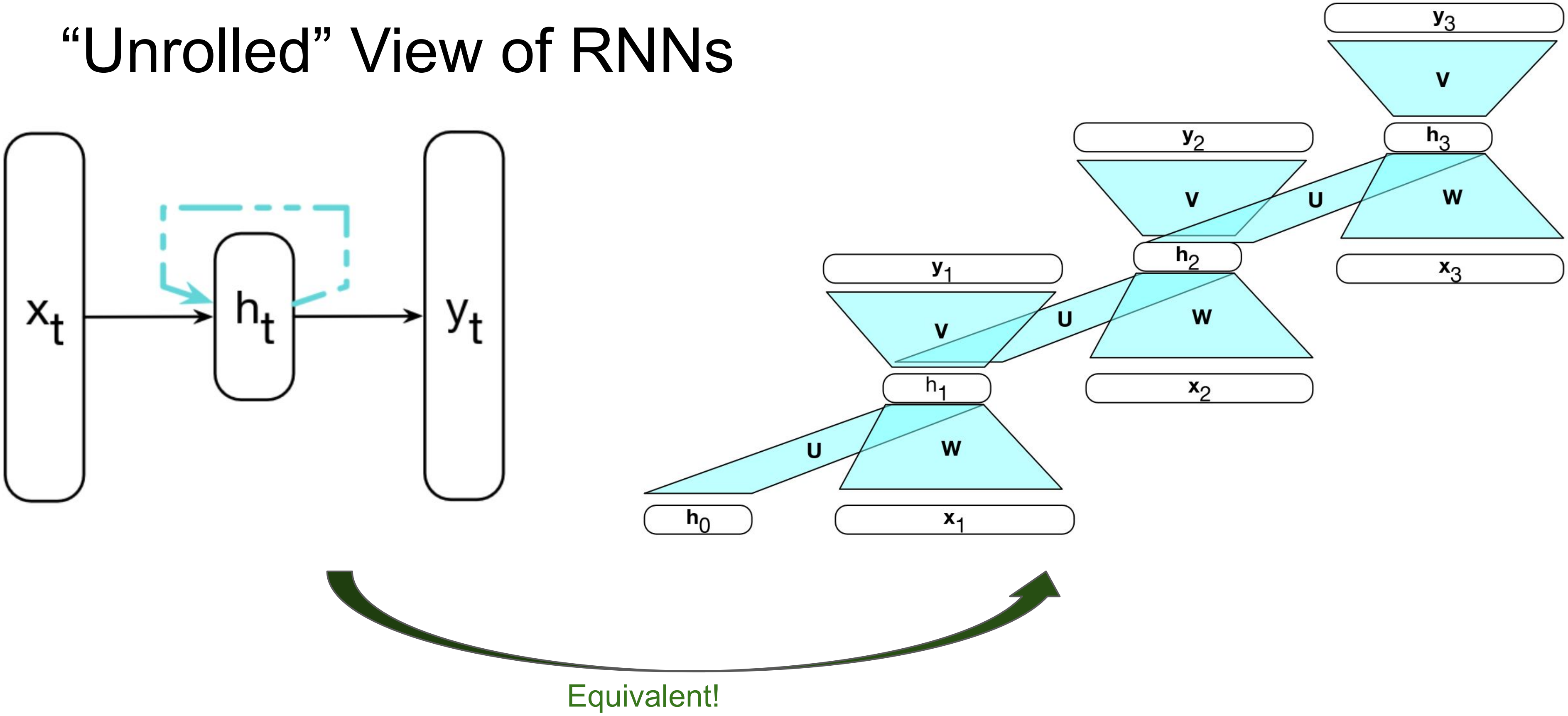
Recurrent Neural Networks (RNNs)

- A **recurrent** neural network is any network that contains a cycle within its network connections

$$
\begin{aligned}
\mathbf{h}_t &= g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t) \\
\mathbf{y}_t &= f(\mathbf{V}\mathbf{h}_t)
\end{aligned}
$$

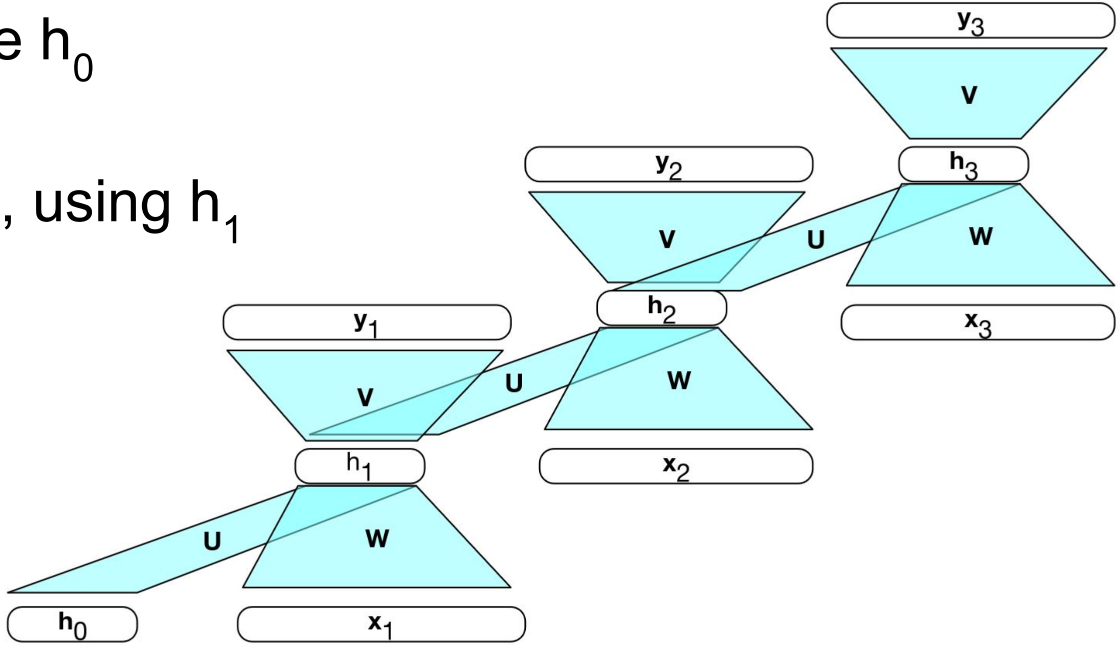


“Unrolled” View of RNNs



"Unrolled" View of RNNs

- Pick some starting state h_0
- Compute h_1 using h_0
- Compute the output y_1, using h_1 and some input x_1
- Compute h_2 using h_1
- …



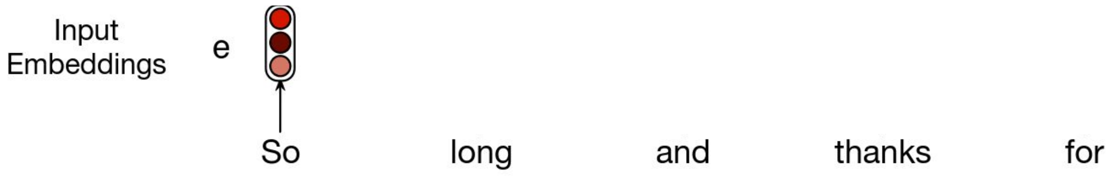
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish***"**

So long and thanks for

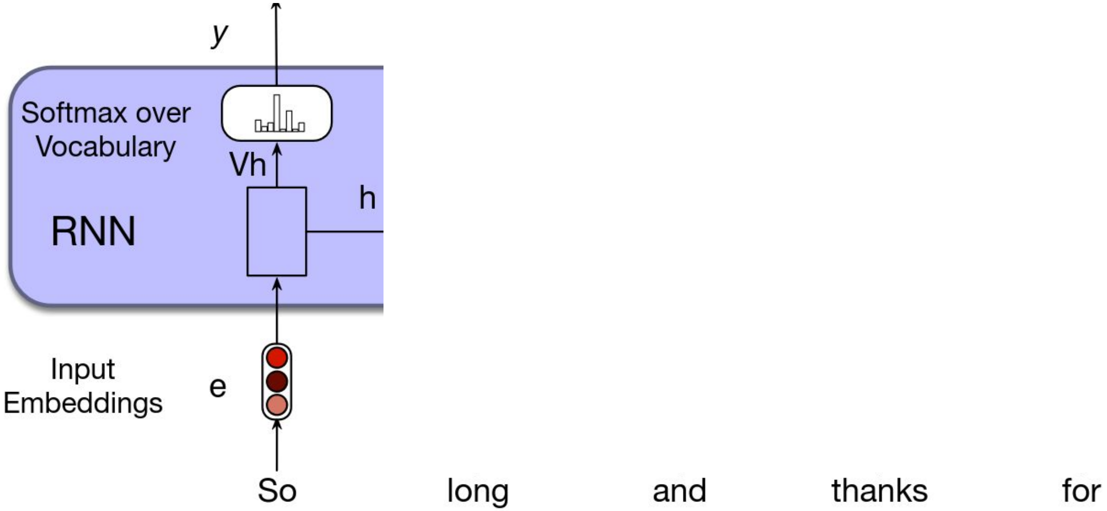
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish***"**



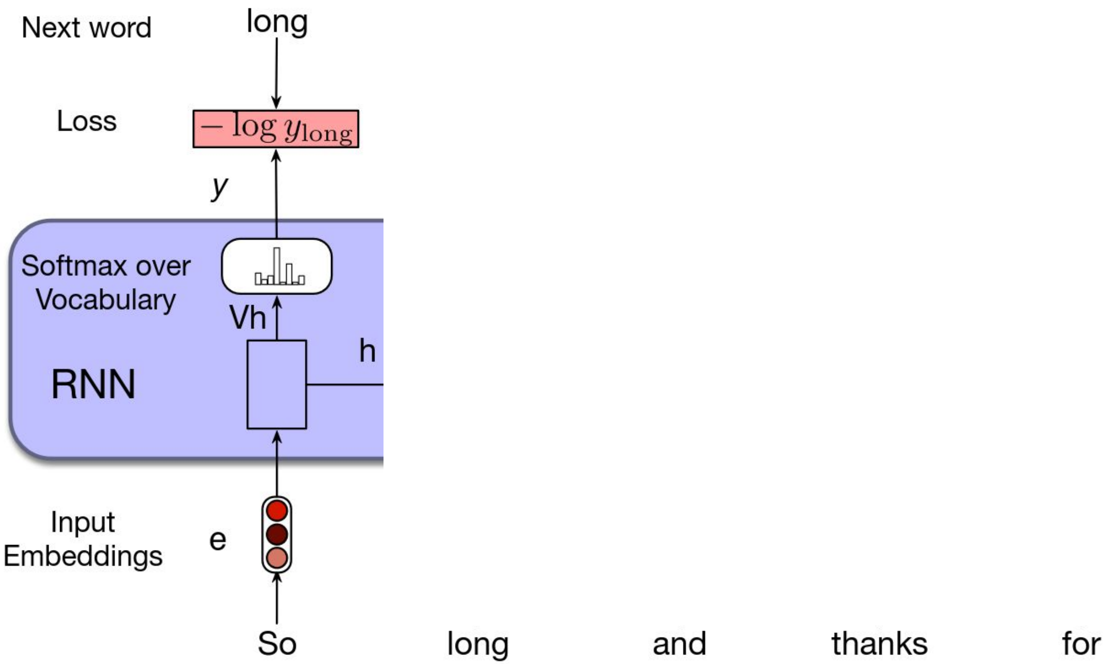
An Example: RNNs for Language Modelling

Predict the sentence “*So long and thanks for all the fish***”**



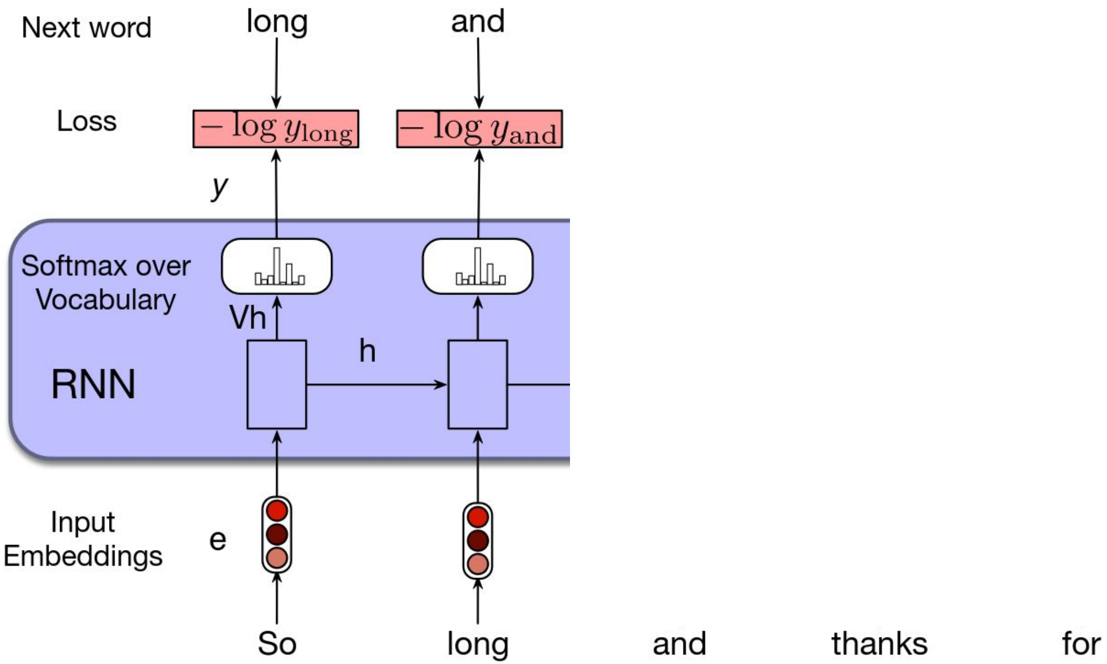
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish***"**



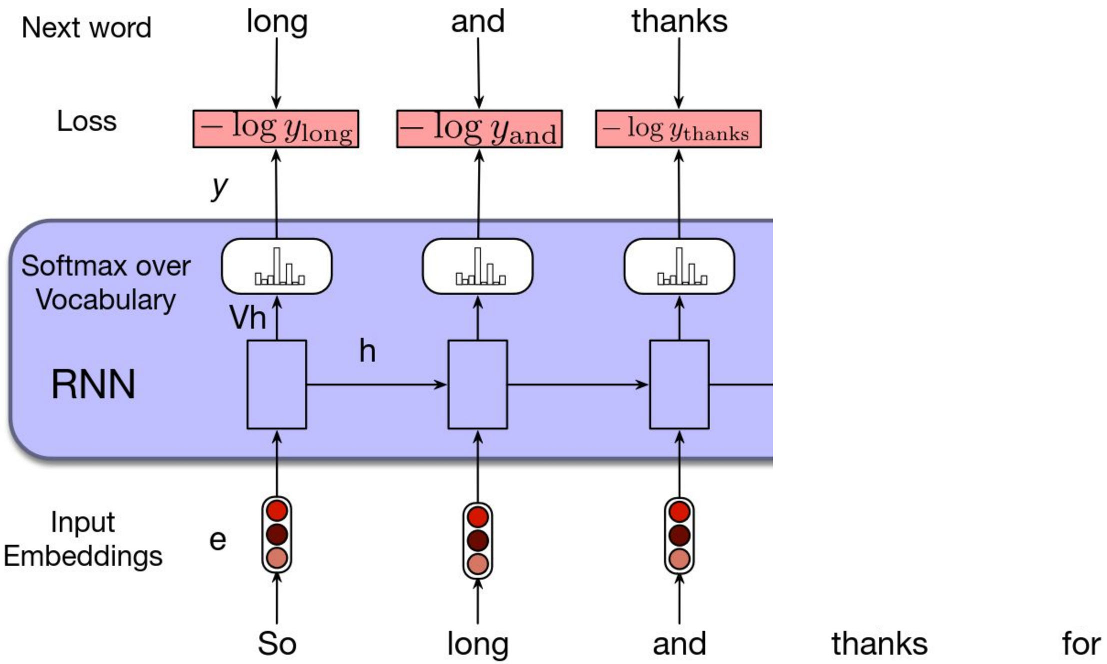
An Example: RNNs for Language Modelling

Predict the sentence “*So long and thanks for all the fish*”



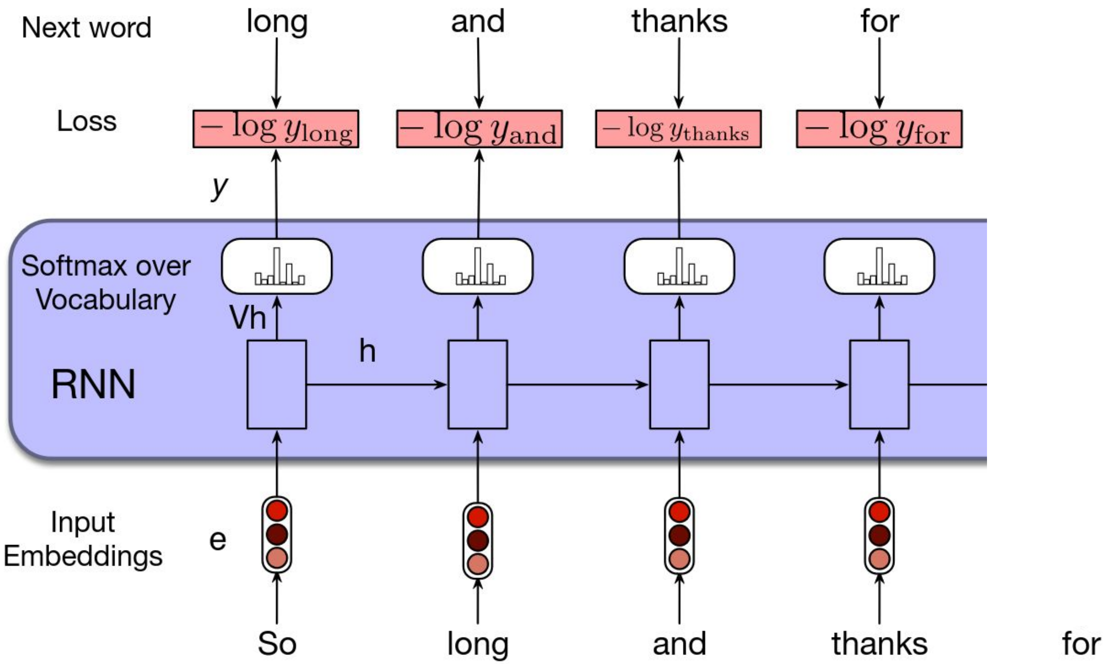
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish***"**



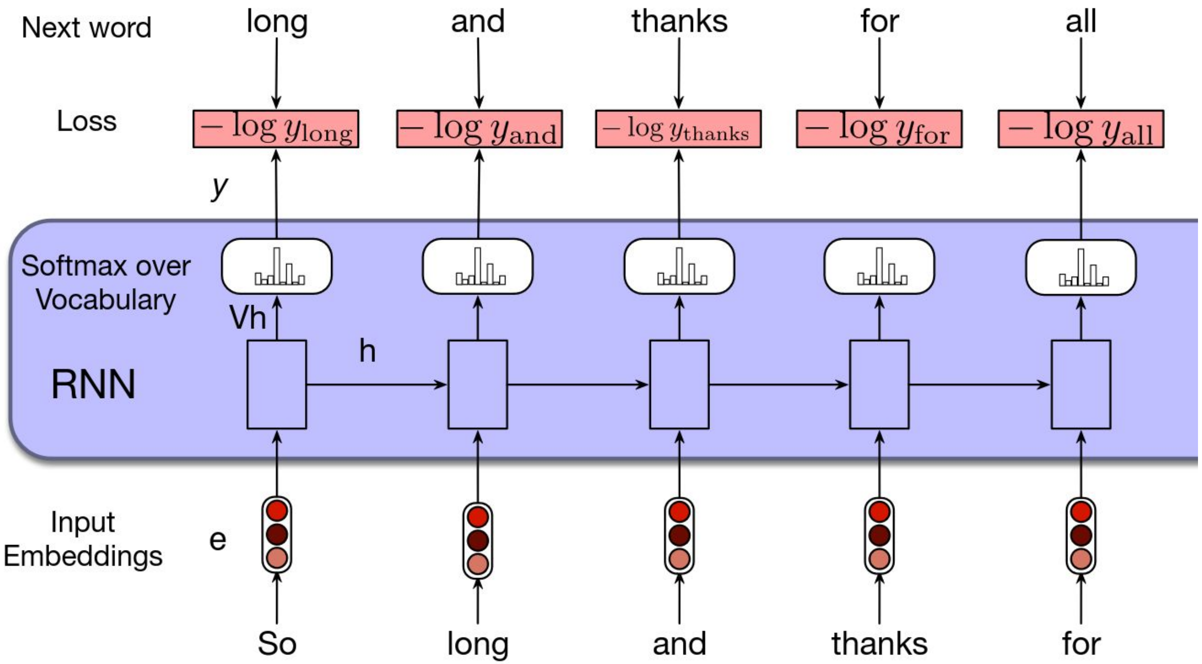
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish***"**



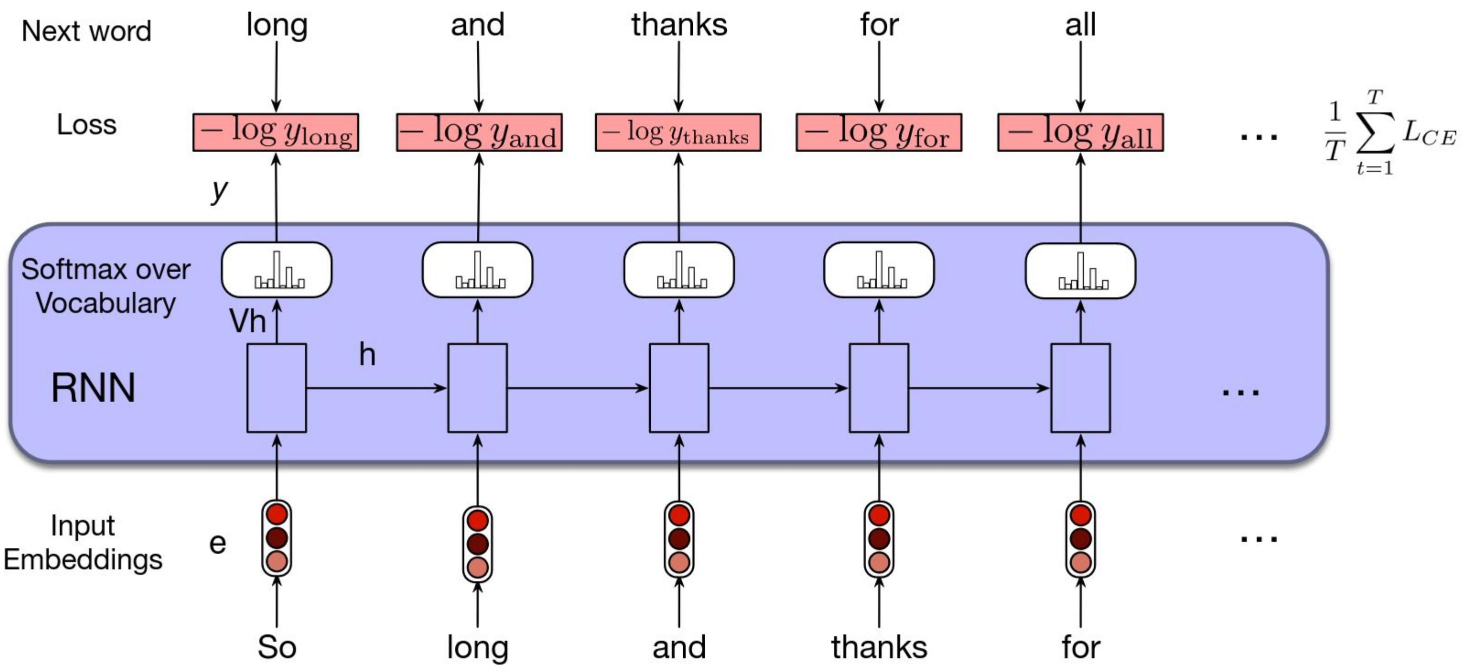
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish***"**



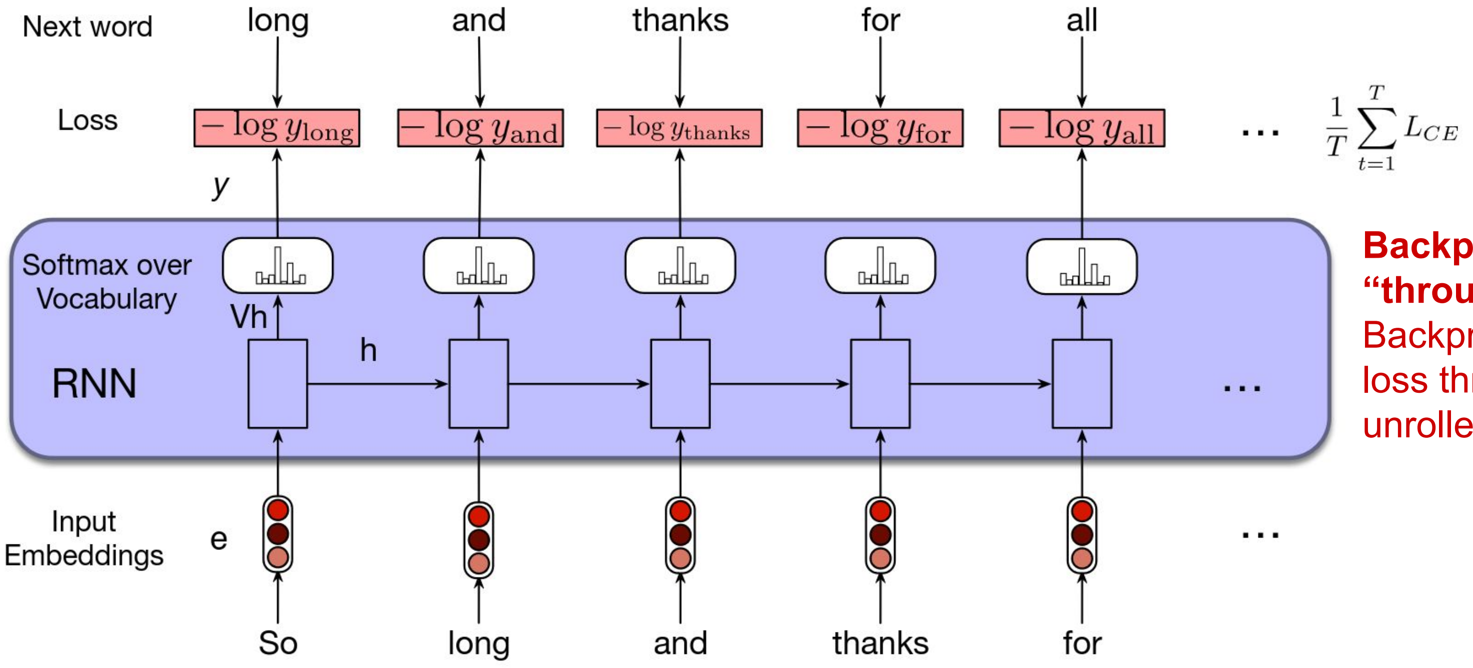
An Example: RNNs for Language Modelling

Predict the sentence "*So long and thanks for all the fish*"



An Example: RNNs for Language Modelling

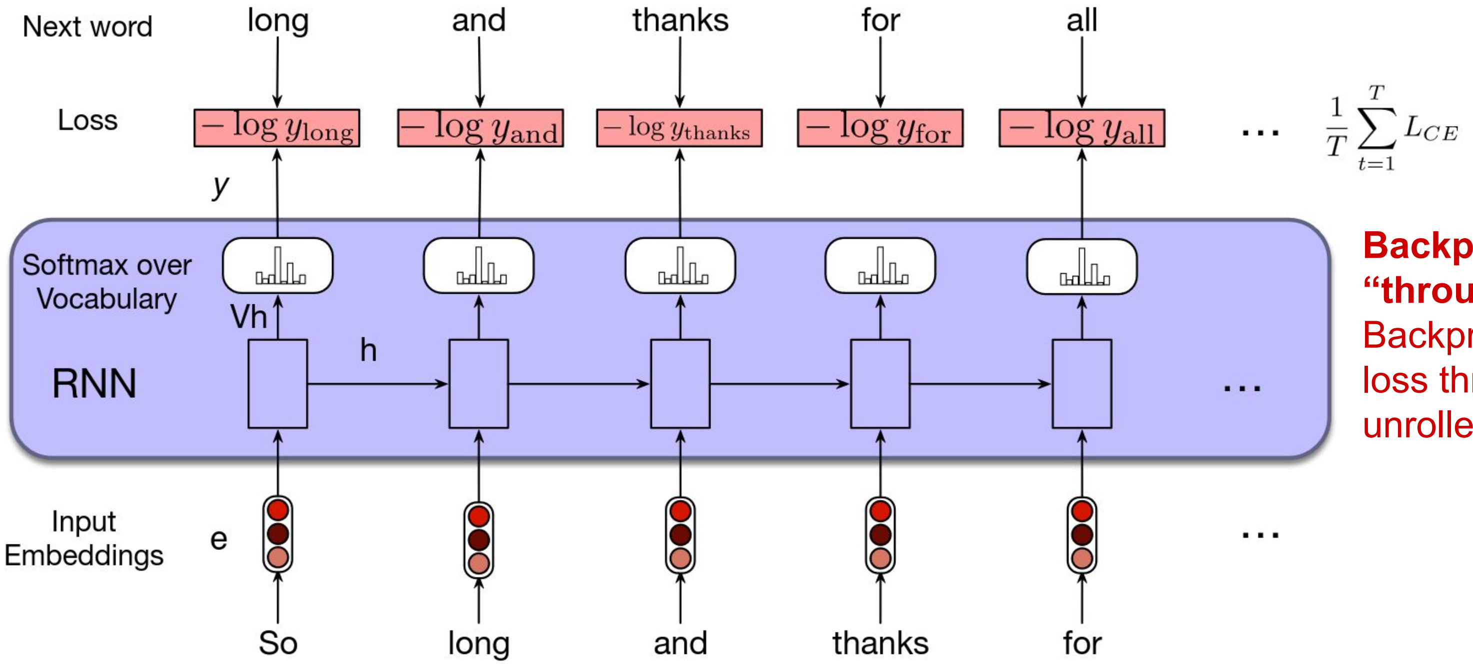
Predict the sentence "*So long and thanks for all the fish*"



Backpropagation "through time": Backpropagating the loss through this unrolled representation

An Example: RNNs for Language Modelling

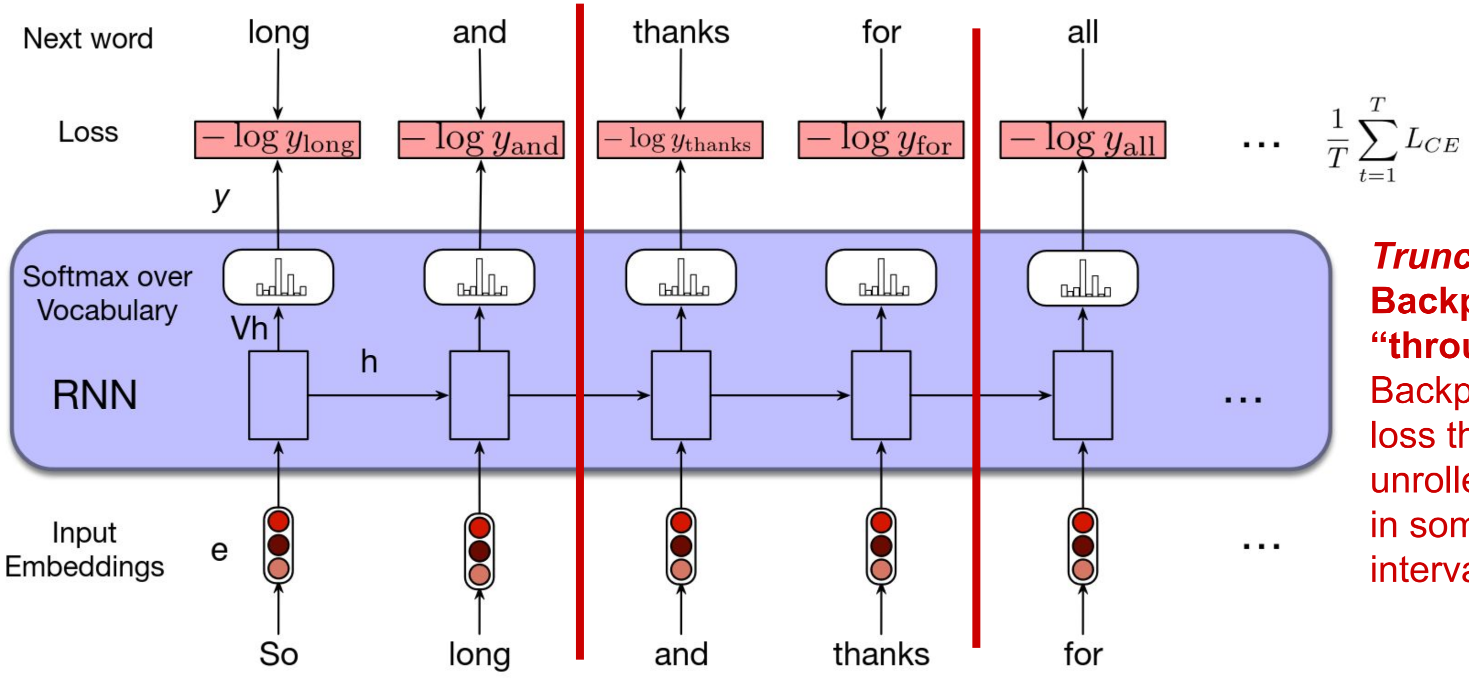
Predict the sentence "*So long and thanks for all the fish*"



Backpropagation "through time": Backpropagating the loss through this unrolled representation

Truncated Backpropagation Through Time

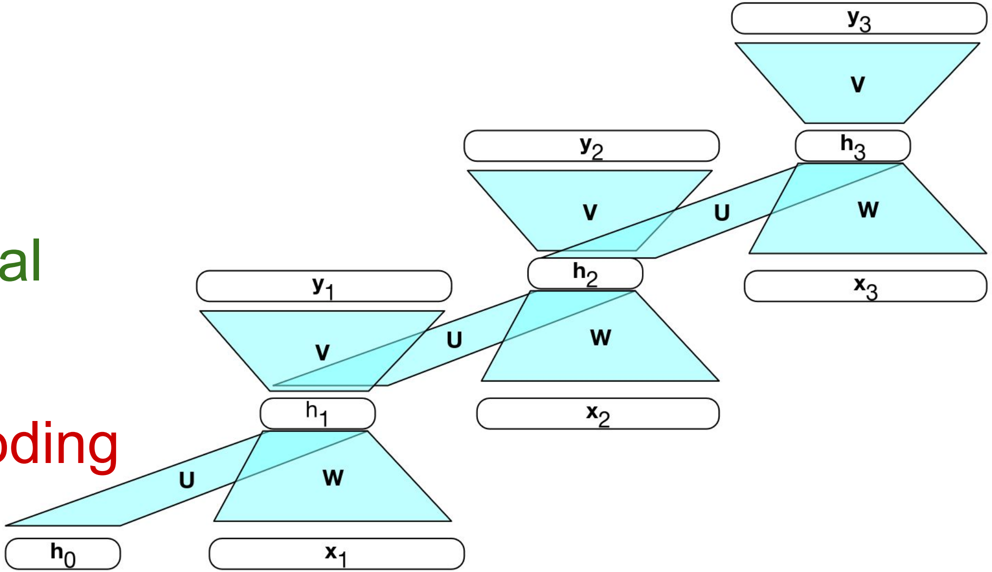
Predict the sentence “*So long and thanks for all the fish*”



***Truncated* Backpropagation “through time”**: Backpropagating the loss through this unrolled representation, in some discrete intervals

Tradeoffs of RNNs

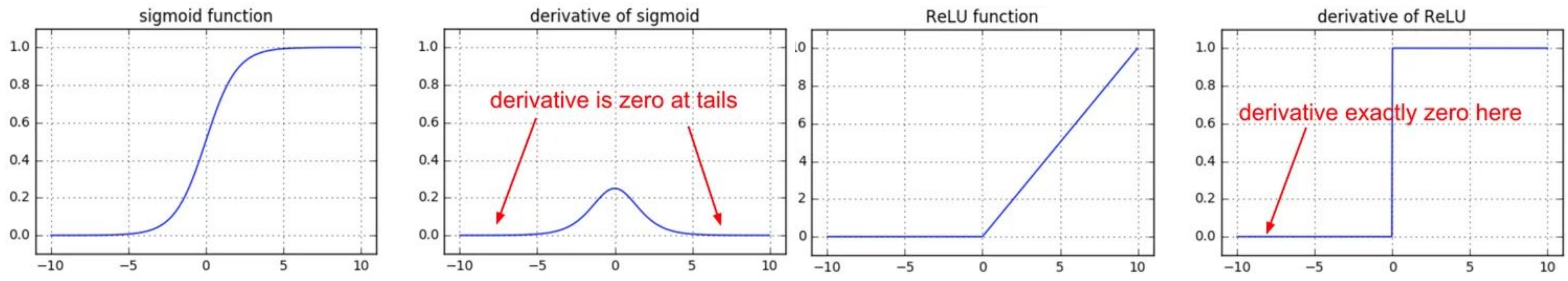
- Can handle arbitrary length inputs
- Reuse weights to reduce total model parameters
- Suffers from vanishing/exploding gradients
- Doesn't take full advantage of highly parallel hardware



An Aside: Some Intuitions on Gradient Issues

https://karpathy.medium.com/yes-you-should-understand-backprop-e2f06eab496b

- **Choice of activation function matters**

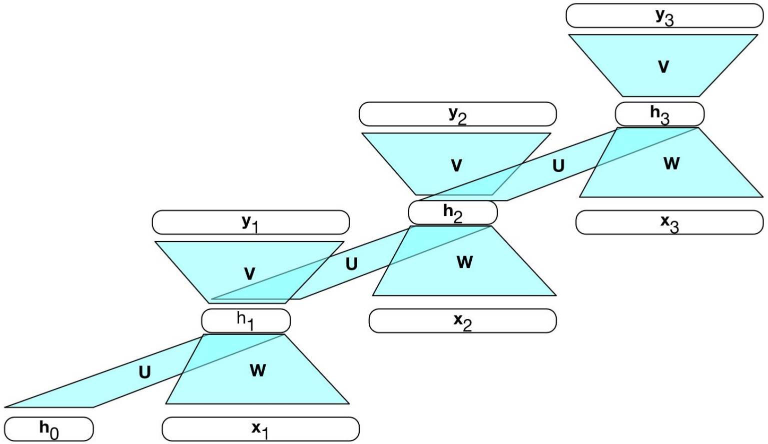


What if you had an unlucky initialization?

An Aside: Some Intuitions on Gradient Issues

https://karpathy.medium.com/yes-you-should-understand-backprop-e2f06eab496b

- Choice of activation function matters
- **Weight initialization matters**

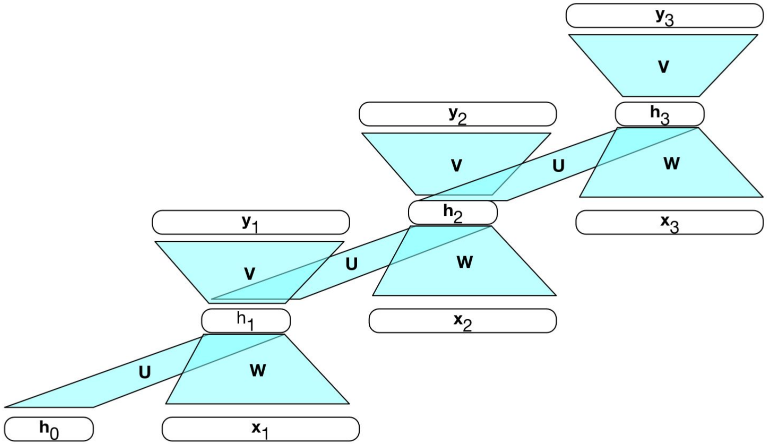


Whether gradients vanish/explode depends on the eigenvalues of weight matrices

An Aside: Some Intuitions on Gradient Issues

https://karpathy.medium.com/yes-you-should-understand-backprop-e2f06eab496b

- Choice of activation function matters
- **Weight initialization matters**



Whether gradients vanish/explode depends on the eigenvalues of weight matrices

Example: Consider simple RNN, with g = ReLU. Suppose all dimensions are 1.

$$\mathbf{h}_t = g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t)$$
$$\mathbf{y}_t = f(\mathbf{V}\mathbf{h}_t)$$

How to solve gradient issues?

Exploding Gradients

- "Clip" the gradients

What would the gradient [2, 2] be clipped to if the max allowed norm is 2?

$[\sqrt{2}, \sqrt{2}]$

How to solve gradient issues?

Exploding Gradients

- "Clip" the gradients

What would the gradient [2, 2] be clipped to if the max allowed norm is 2?

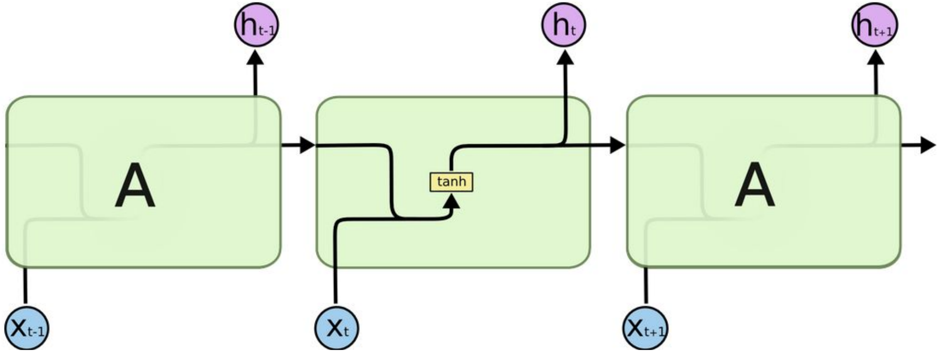
[$\sqrt{2}$, $\sqrt{2}$]

Vanishing Gradients

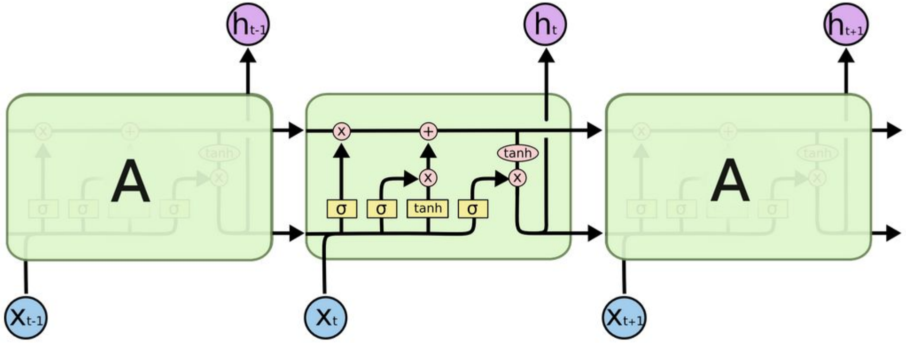
- Choose a different activation function
- Initialize weights properly
- **Use a different architecture (e.g. LSTM)**

LSTMs

Simple RNN

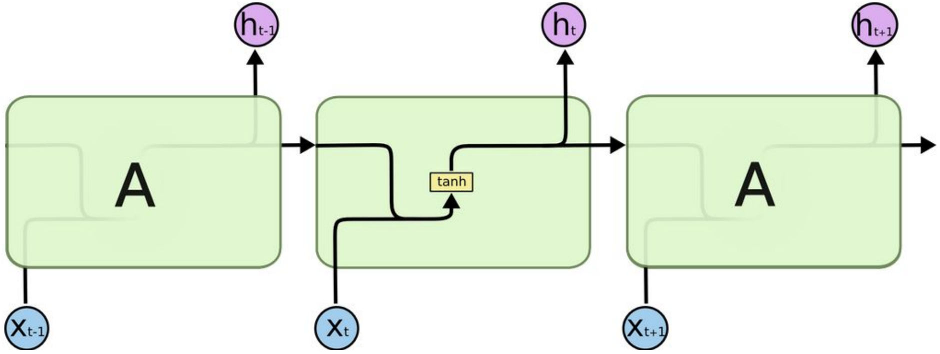


LSTM

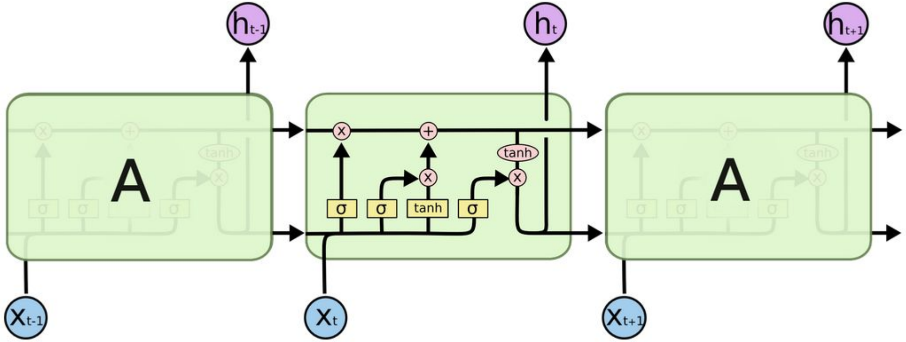


LSTMs

Simple RNN



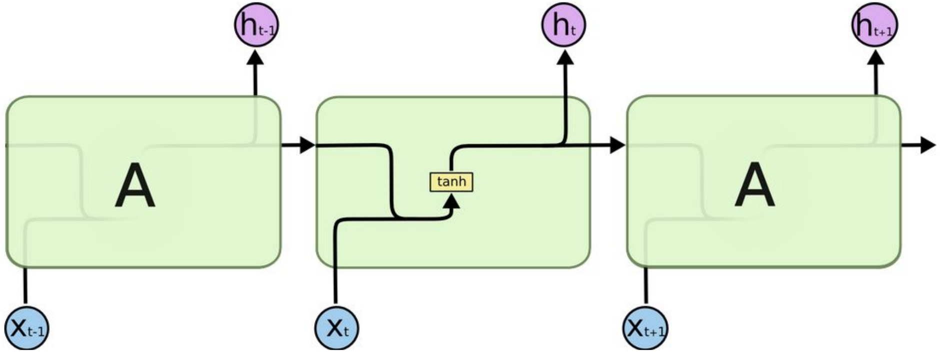
LSTM



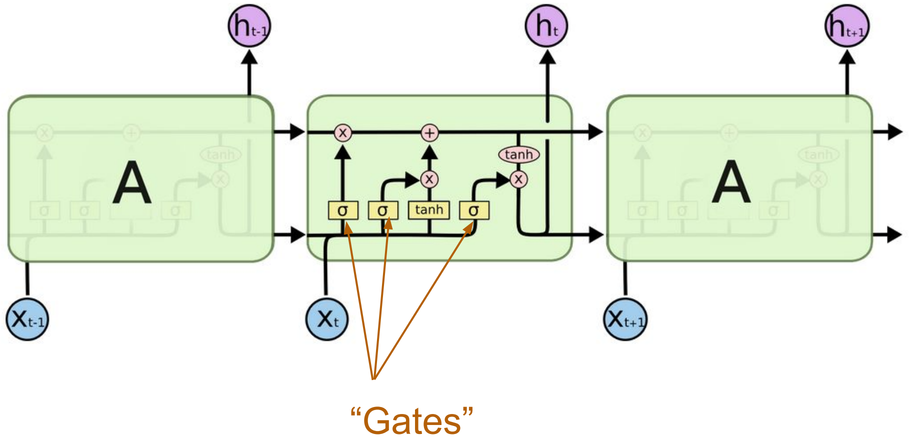
Two recurrent values!
(hidden and cell states)

LSTMs

Simple RNN



LSTM

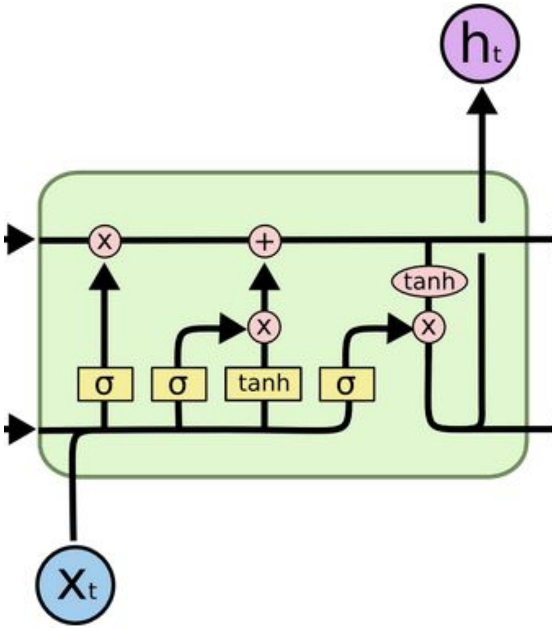


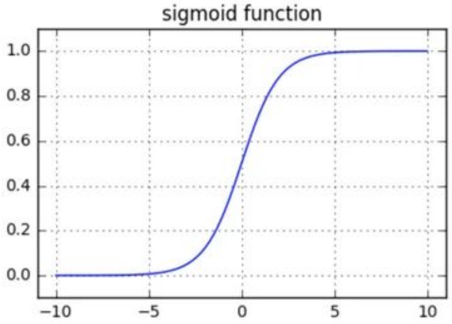
Two recurrent values!
(hidden and cell states)

LSTMs Broken Down

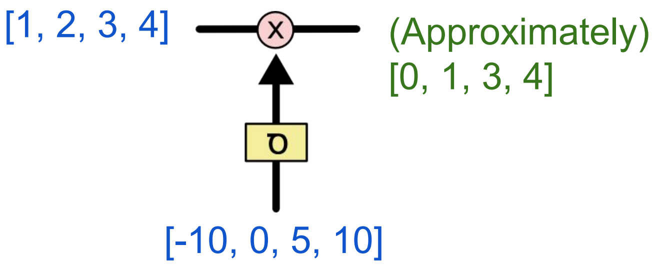
Gates (i.e. sigmoid followed by multiplication)

- Outputs value in range (0, 1)
- Intuitive meaning:
  - Close to 0 => “forget this value”
  - Close to 1 => “keep this value”





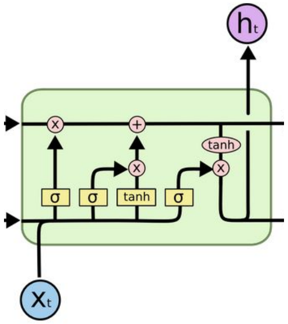
Example:



LSTM: Example scenario (Language Modelling)

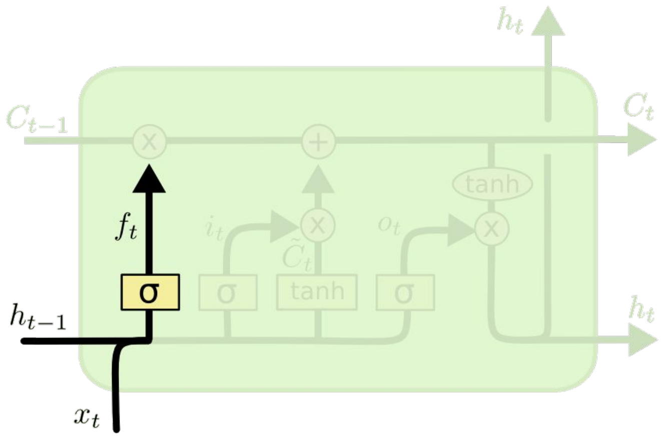
Suppose we are predicting the sentence “Jon is a boy. Sally is a girl.**”**

LSTM: Step 1



Suppose we are predicting the sentence "Jon is a boy. Sally is a girl."

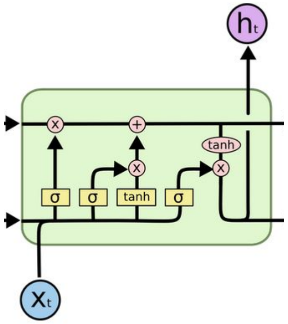
Step 1 (Forget gate): Discard information.
*"Given the current input and the previous hidden state, how much should I **discard from** the cell state?"*



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

e.g. When I see the word "Sally", I may want to discard existing information associated with the gender of the subject in the cell state (which may be carried over from the first half of the sentence)

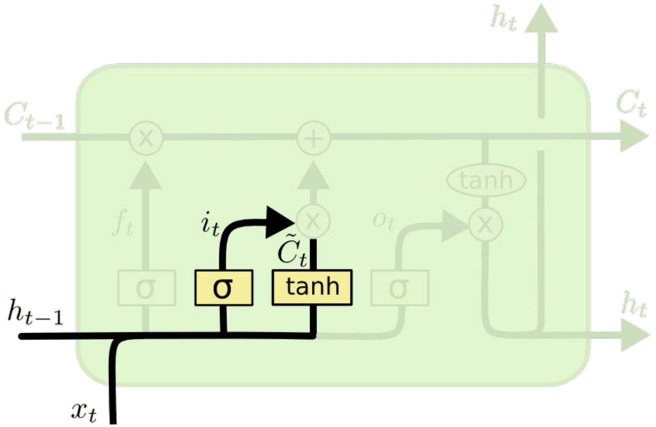
LSTM: Step 2



Suppose we are predicting the sentence "Jon is a boy. Sally is a girl."

Step 1 (Forget gate): Discard information.
"Given the current input and the previous hidden state, how much should I ***discard from*** *the cell state?"*

Step 2 (Input gate): Add new information.
"Given the current input and the previous hidden state, what should I ***add to*** *the cell state?"*

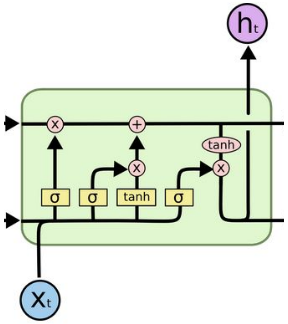


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

e.g. When I see the word "Sally", I may want to add information to the cell state indicating that the subject is female

LSTM: Step 3



Suppose we are predicting the sentence “Jon is a boy. Sally is a girl.”

Step 1 (Forget gate): Discard information.

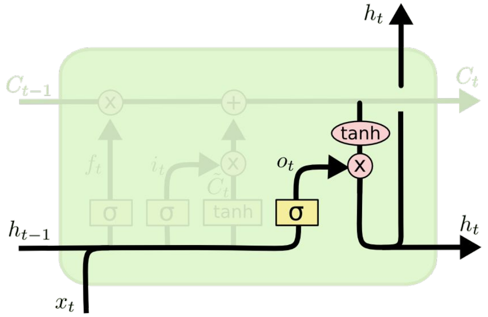
*“Given the current input and the previous hidden state, how much should I **discard from** the cell state?”*

Step 2 (Input gate): Add new information.

*“Given the current input and the previous hidden state, what should I **add to** the cell state?”*

Step 3 (Output gate): Compute the output.

“Given the current input, previous hidden state, and updated cell state, what should I output?”



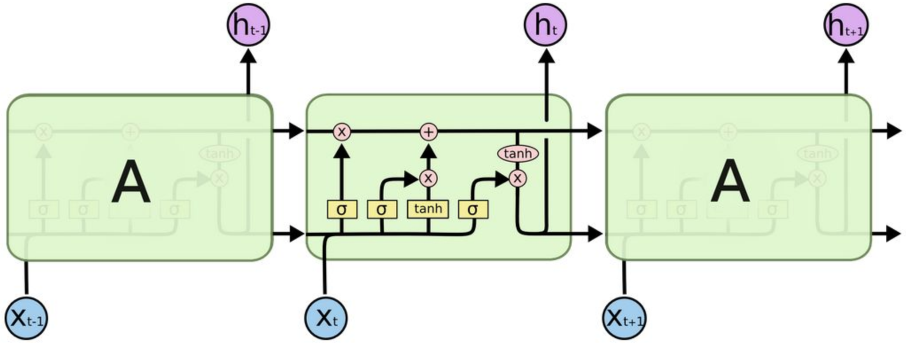
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

e.g. Predicting the word “girl” given that your cell state should contain gender information from when it saw Sally

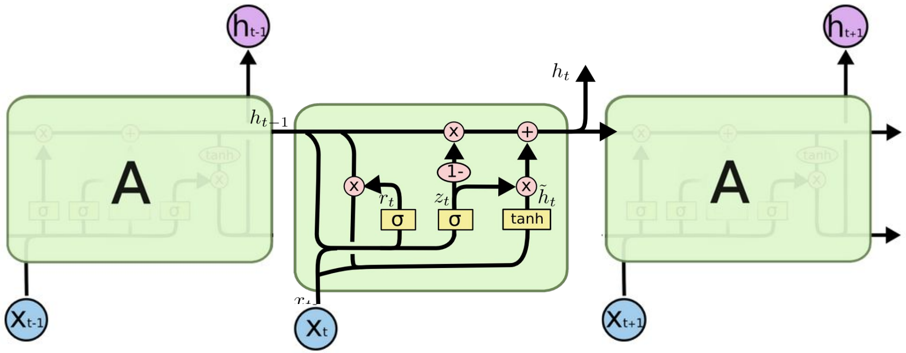
Gated Recurrent Units (GRUs)

LSTM



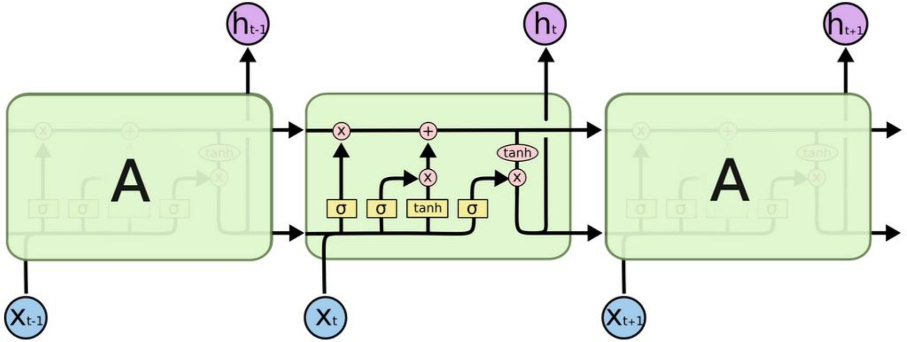
GRU

Only **one** recurrent state
Input and forget gates are combined!



Gated Recurrent Units (GRUs)

LSTM

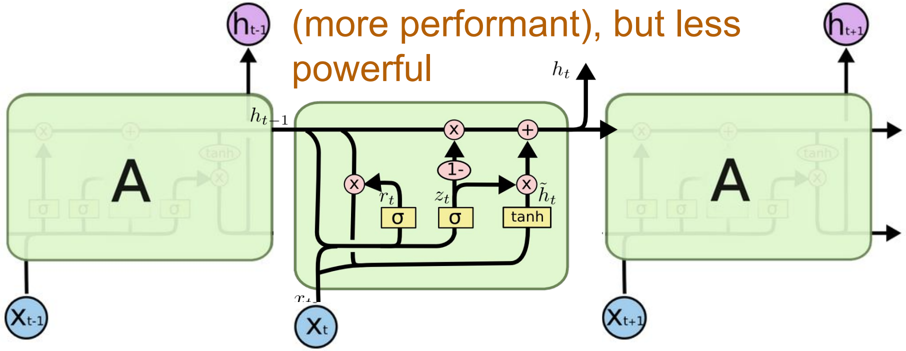


GRU

Only **one** recurrent state

Input and forget gates are combined!

Fewer parameters than LSTMs (more performant), but less powerful



Bidirectional RNNs

When we have access to an entire sequence $x_0, \ldots, x_n$ at once, we can improve performance using **bidirectional** RNNs