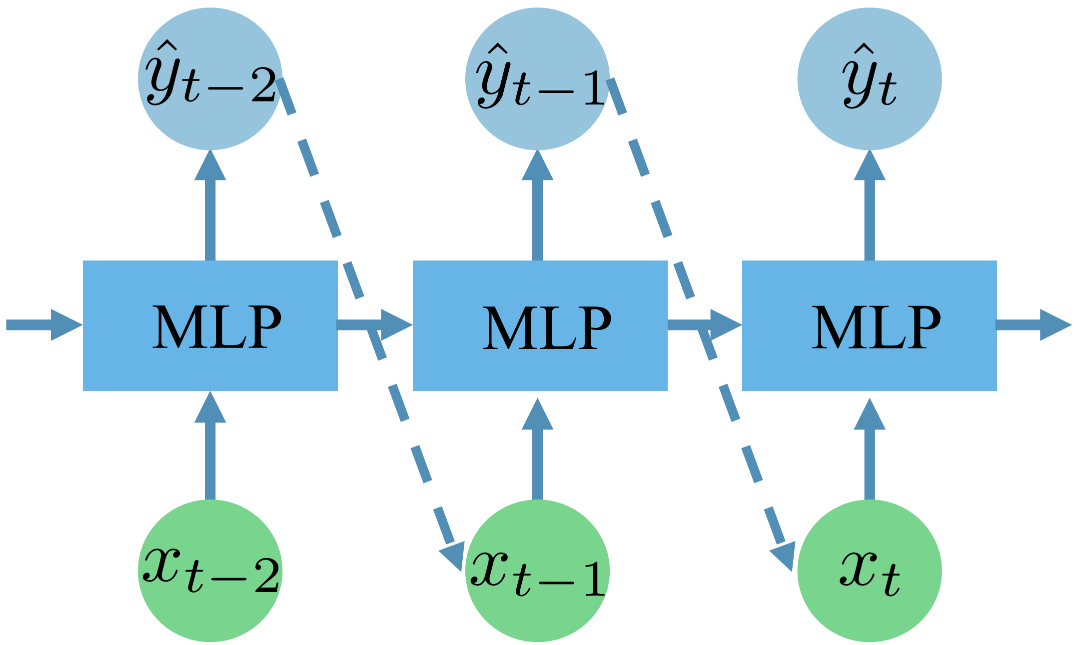
Previously on this week: Recurrent Architecture



Language model:

x - word embedding

This one generates a sentence.

$\hat{y}$ - probability distribution for the next word <- over a vocabulary.

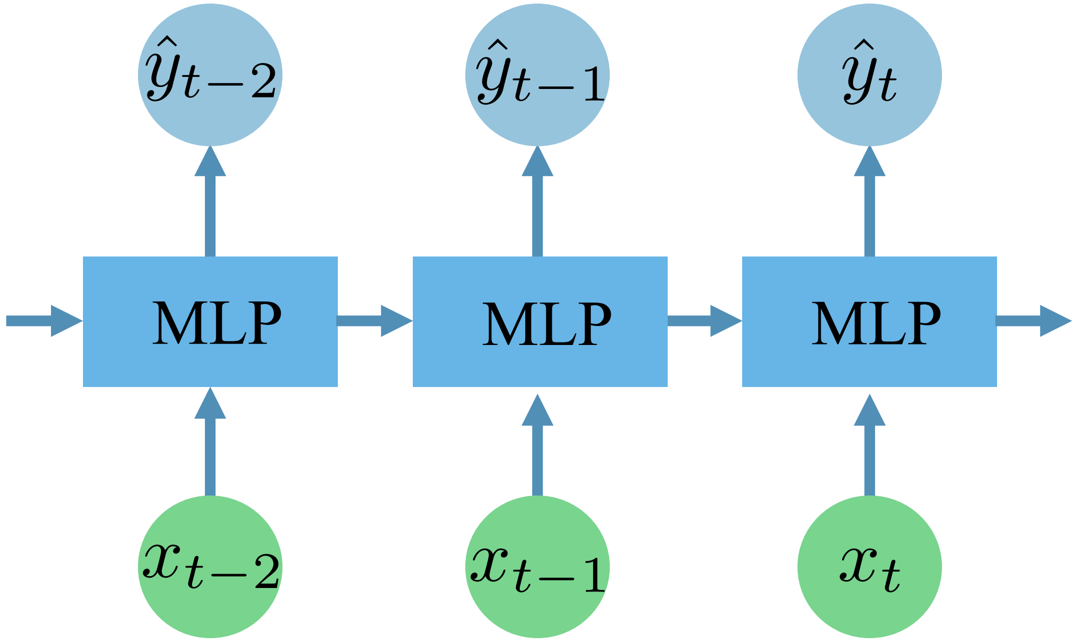
^

|

generate the word with highest probability distribution.

Use it on the next timestamp.

Previously on this week: Recurrent Architecture

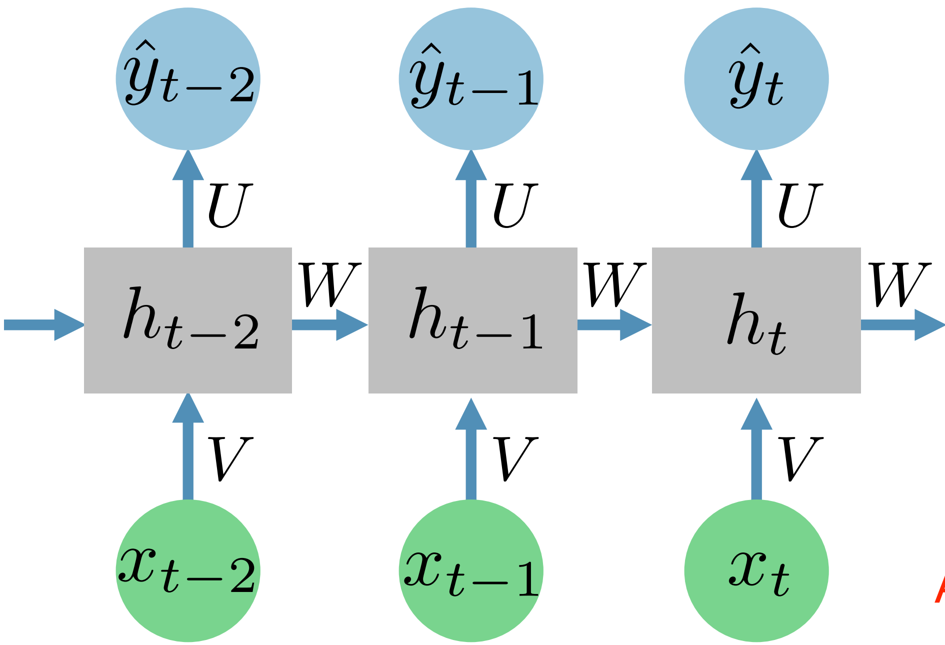


POS tagging:

x - word embedding

$\hat{y}$ - probability distribution for a POS tag of the current word

Recurrent Neural Network (RNN)



W, U and V are preserved throughout timesteps.
Also, same bias is used throughout timesteps

x - input

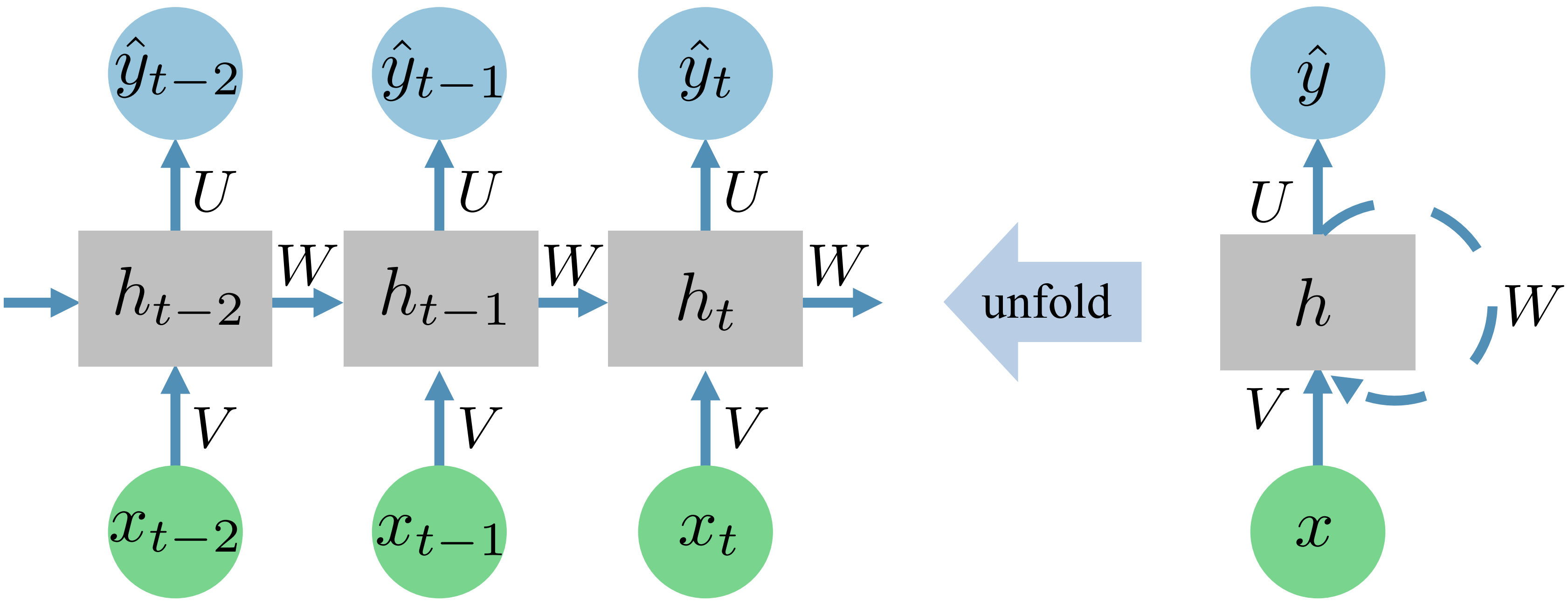
$\hat{y}$ - output (prediction)

h - hidden state

$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

$$\hat{y}_t = f_y(Uh_t + b_y)$$

Recurrent Neural Network (RNN)



x - input

$\hat{y}$ - output (prediction)

h - hidden state

$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

$$\hat{y}_t = f_y(Uh_t + b_y)$$

How to train RNN?

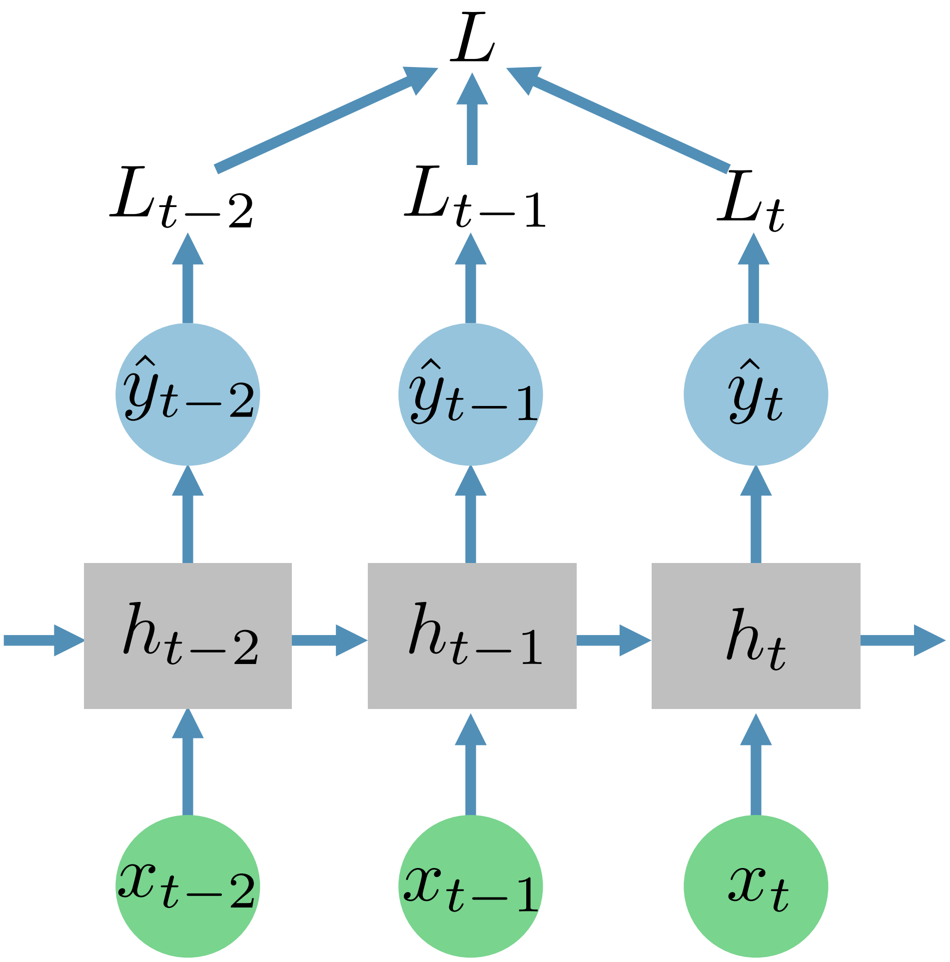
Let's consider an RNN in the unfolded form.

At each time step:

- y_t - true label
- $\hat{y}_t$ - prediction
- $L_t(y_t, \hat{y}_t)$ - some loss function

Loss:

$$L = \sum_i L_i(y_i, \hat{y}_i)$$



We can use Backpropagation to train the RNN!

Backpropagation Through Time (BPTT)

As usual we do forward and backward passes.

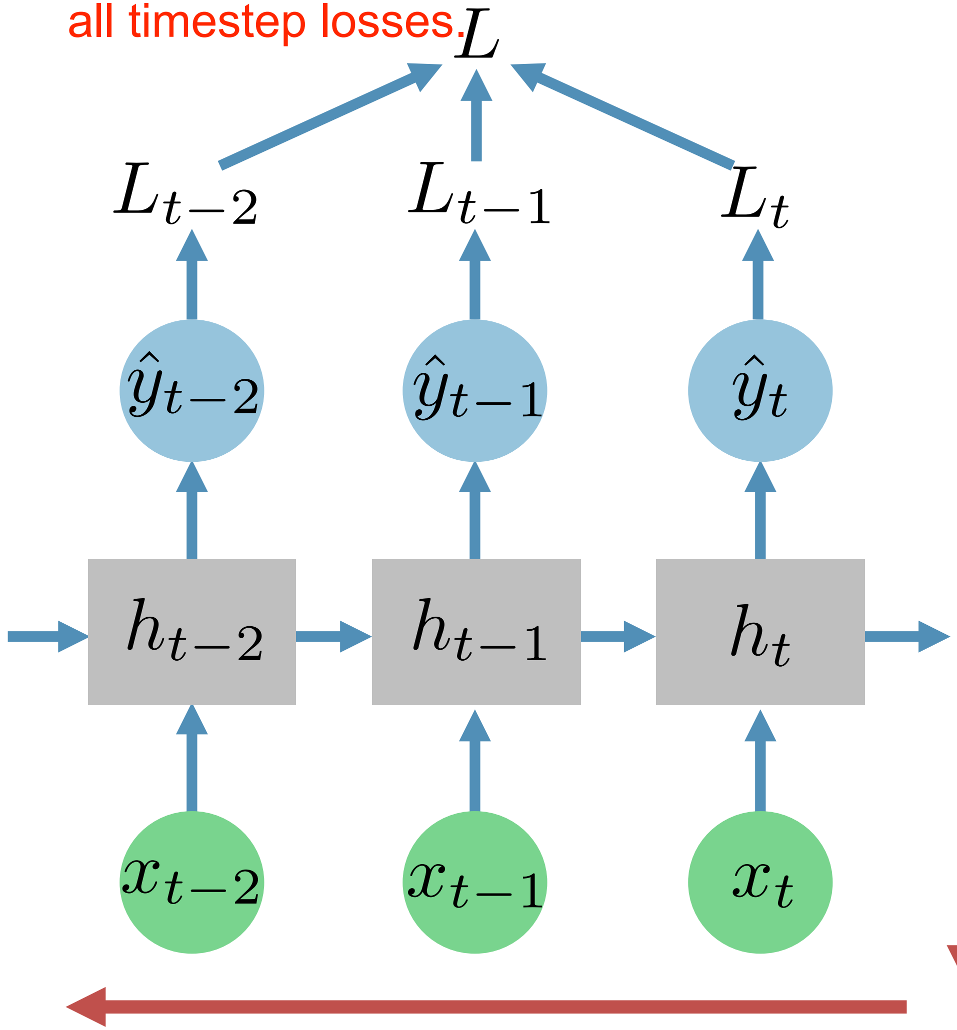
Forward pass:

$$h_t, \ \hat{y}_t, \ L_t, \ L$$

Backward pass:

$$\frac{\partial L}{\partial U}, \ \frac{\partial L}{\partial V}, \ \frac{\partial L}{\partial W}, \ \frac{\partial L}{\partial b_x}, \ \frac{\partial L}{\partial b_h}$$

The total loss L is an aggregate sum over all timestep losses.



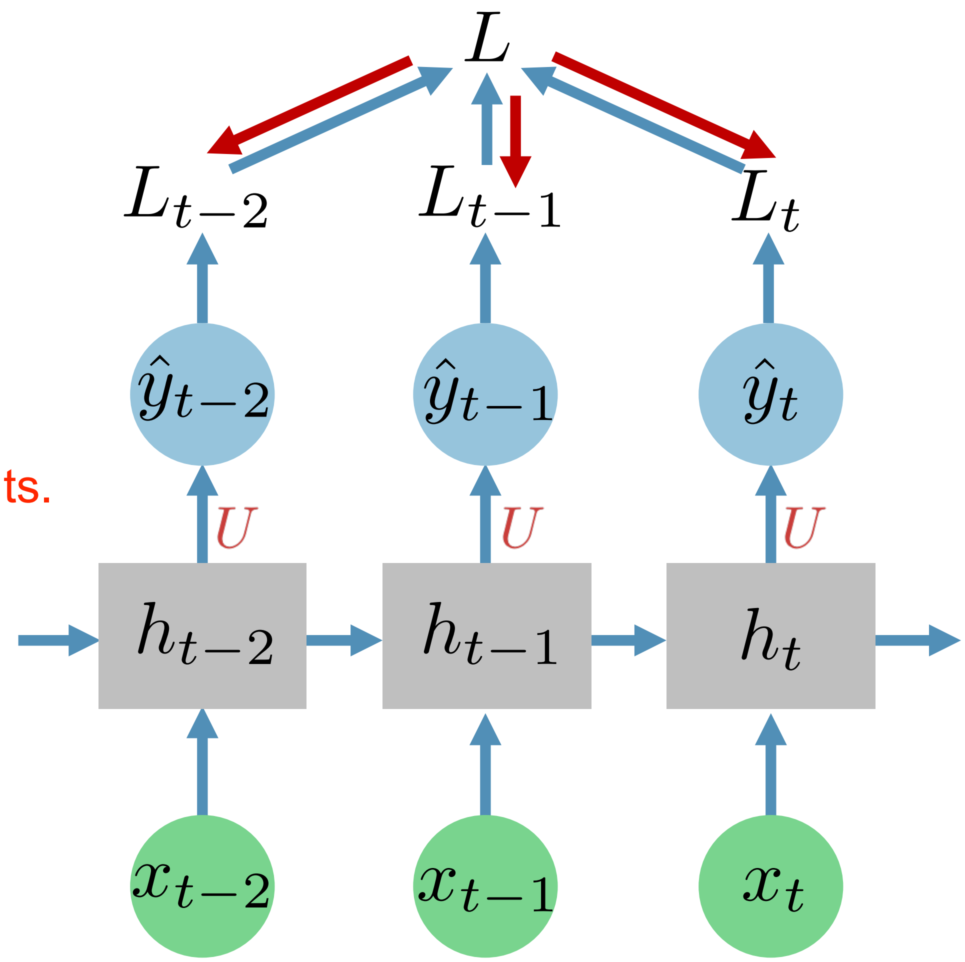
We backpropagate through layers and time.

Backpropagation Through Time (BPTT)

All weights are shared across time steps!

$$\frac{\partial L}{\partial U} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial U}$$

We sum the timestep loss gradients.



Backpropagation Through Time (BPTT)

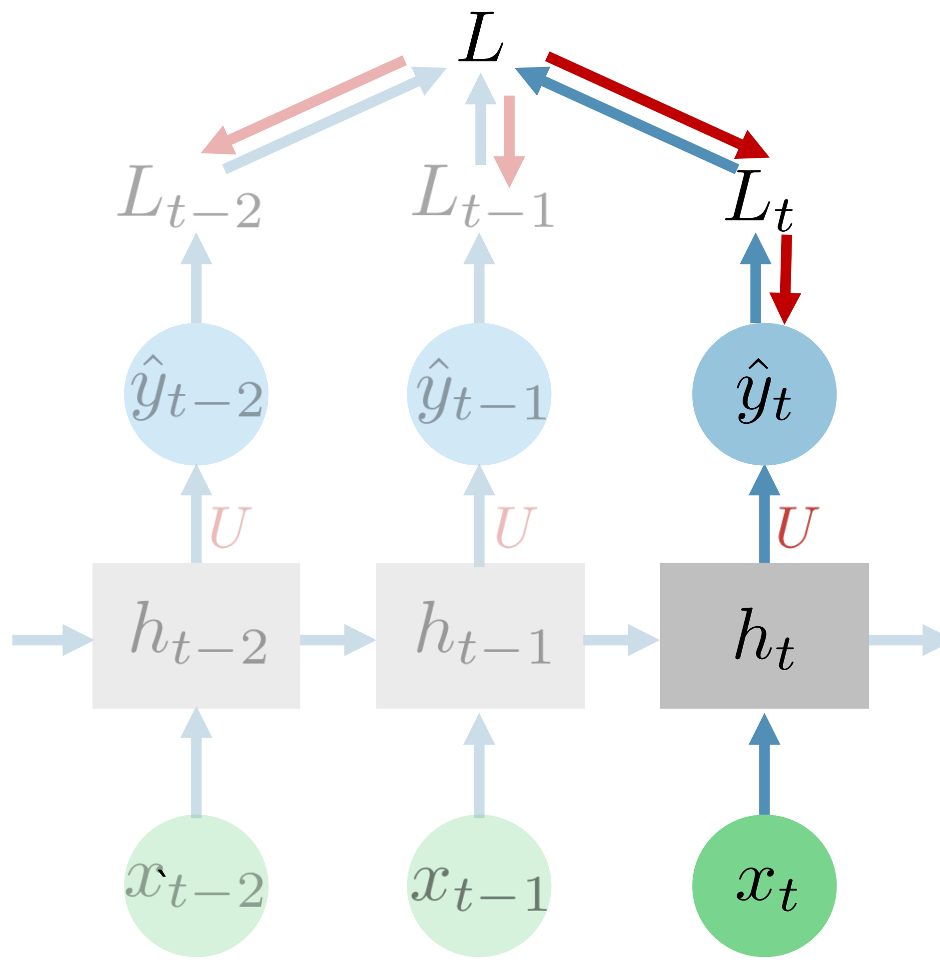
All weights are shared across time steps!

$$\frac{\partial L}{\partial U} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial U}$$

$$\frac{\partial L_t}{\partial U} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial U}$$

$$\hat{y}_t = f_y(\boxed{U} h_t + b_y)$$

this is the only dependence



f_y is probably some activation function. Trivial to compute. deriv of y_t WRT U.

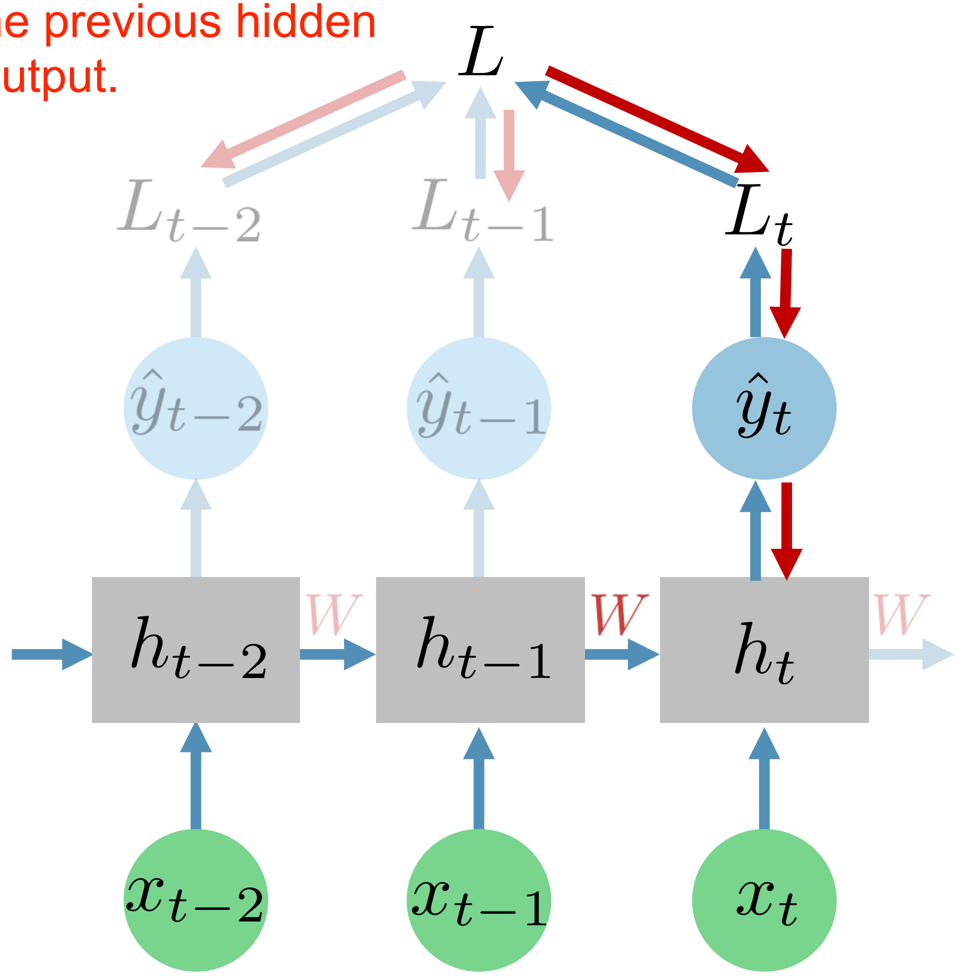
Backpropagation Through Time (BPTT)

Lets calculate W, The matrix used to dot the previous hidden output.

All weights are shared across time steps!

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$



Backpropagation Through Time (BPTT)

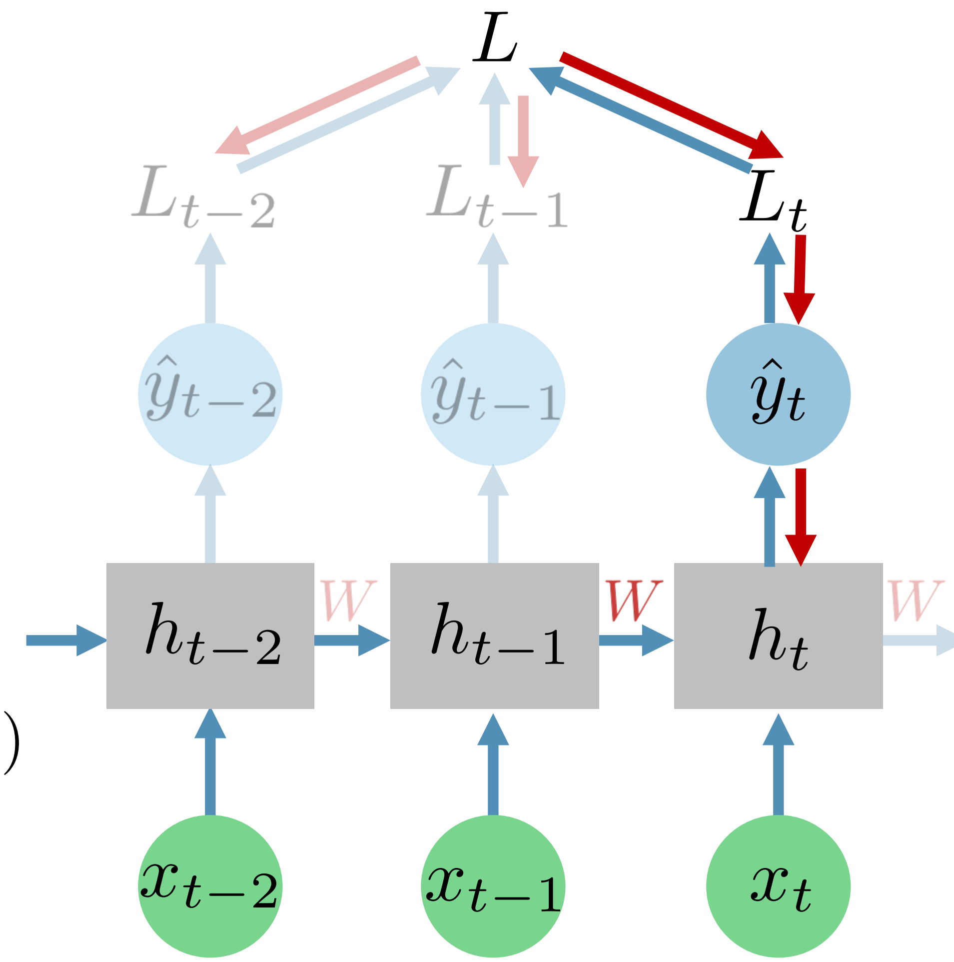
All weights are shared across time steps!

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\cancel{\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}}$$

$$h_t = f_h(Vx_t + W\boxed{h_{t-1}} + b_h)$$

We don't just depend on W. We depend on h_{t-1} too!



Backpropagation Through Time (BPTT)

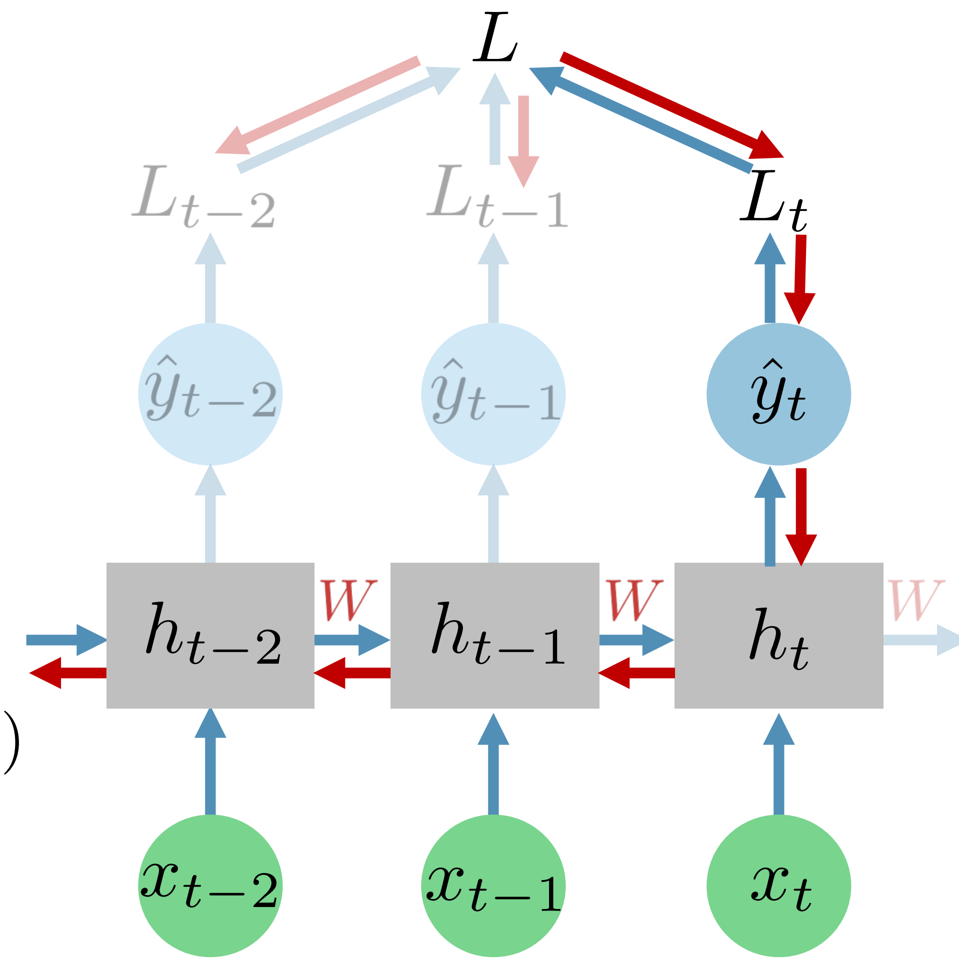
All weights are shared across time steps!

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\xcancel{\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}}$$

$$h_t = f_h(V x_t + W \boxed{h_{t-1}} + b_h)$$

Depends on h_{t-1} too!



$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \boxed{\left(\frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W} + \dots \right)}$$

Backpropagation Through Time (BPTT)

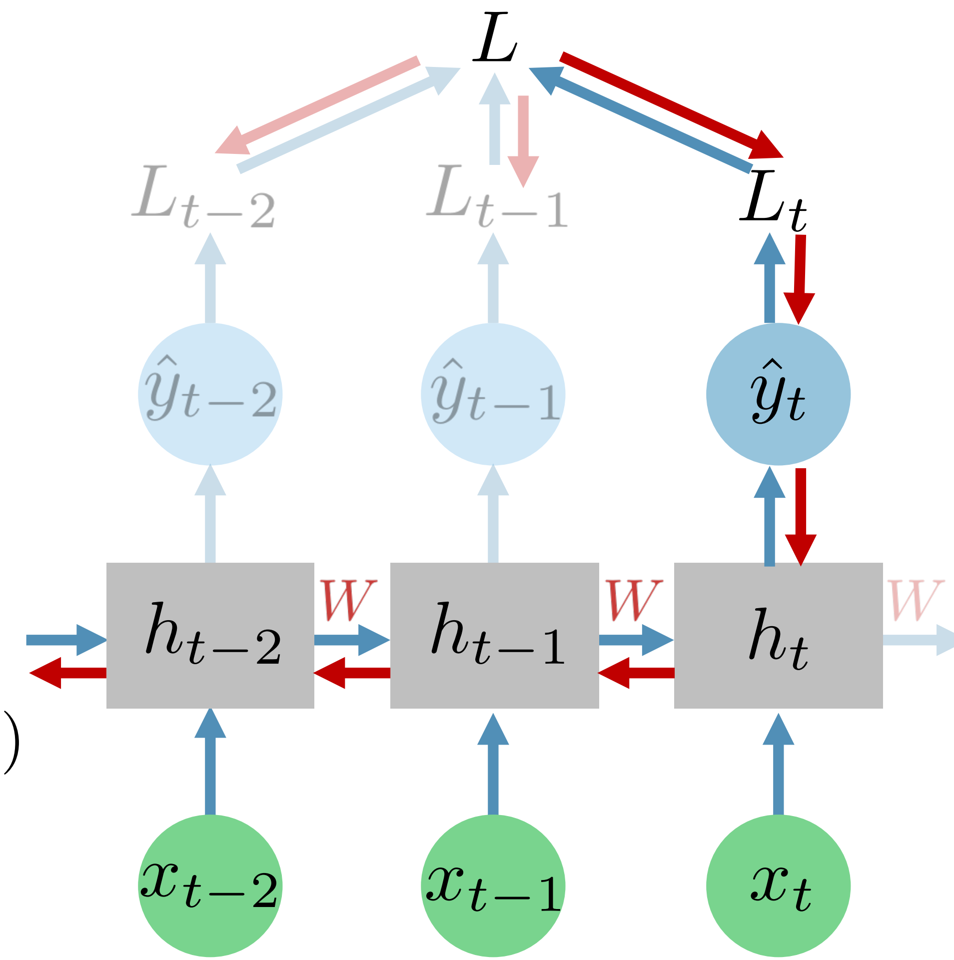
All weights are shared across time steps!

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\cancel{\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}}$$

$$h_t = f_h(Vx_t + W\boxed{h_{t-1}} + b_h)$$

Depends on h_{t-1} too!



$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \boxed{\sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}}$$

NOTE, the k term is used in the PROD index!

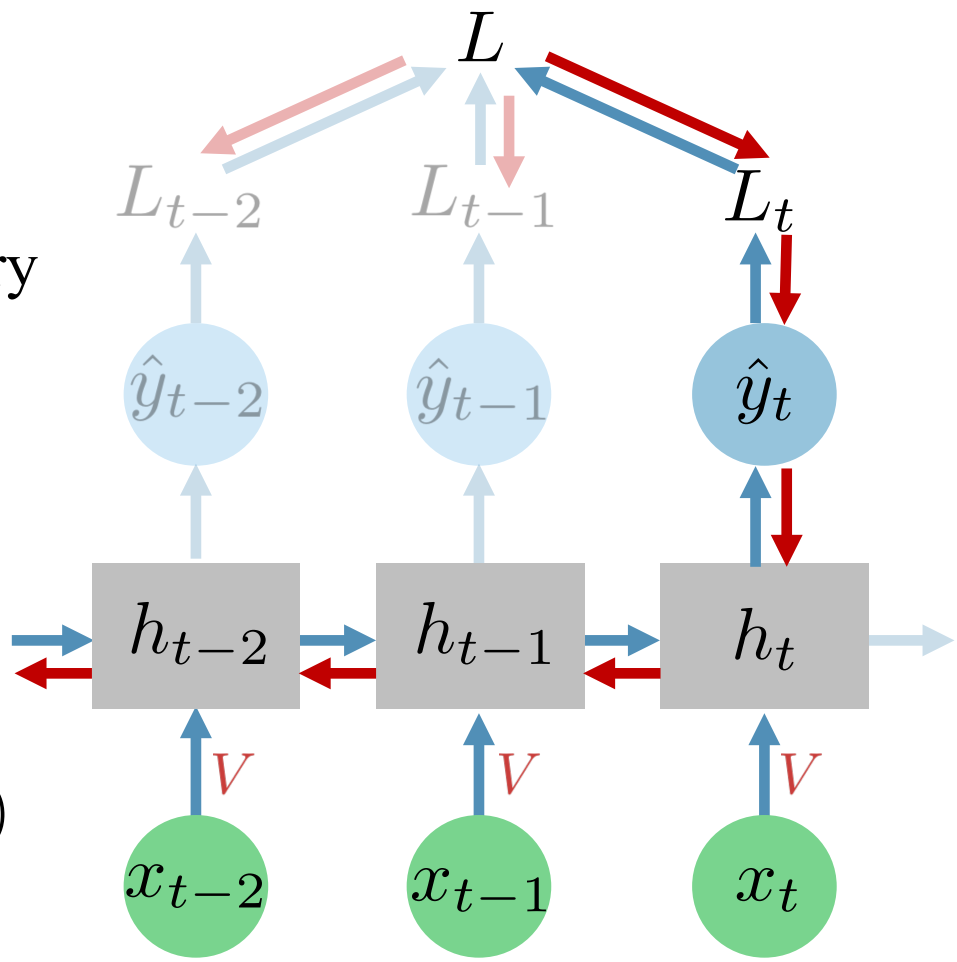
Backpropagation Through Time (BPTT)

And what about the last weight matrix V? Is it necessary to go backwards in time to calculate $\dfrac{\partial L}{\partial V}$?

Yes! Here we have the same situation as with W:

$$h_t = f_h(Vx_t + W\boxed{h_{t-1}} + b_h)$$

Depends on h_{t-1} too!



Summary

- We have learned what is a simple RNN.
- RNNs are trained using simple Backpropagation. BPTT is just a fancy name for it.

In the next video:

Is it really that simple to train an RNN?