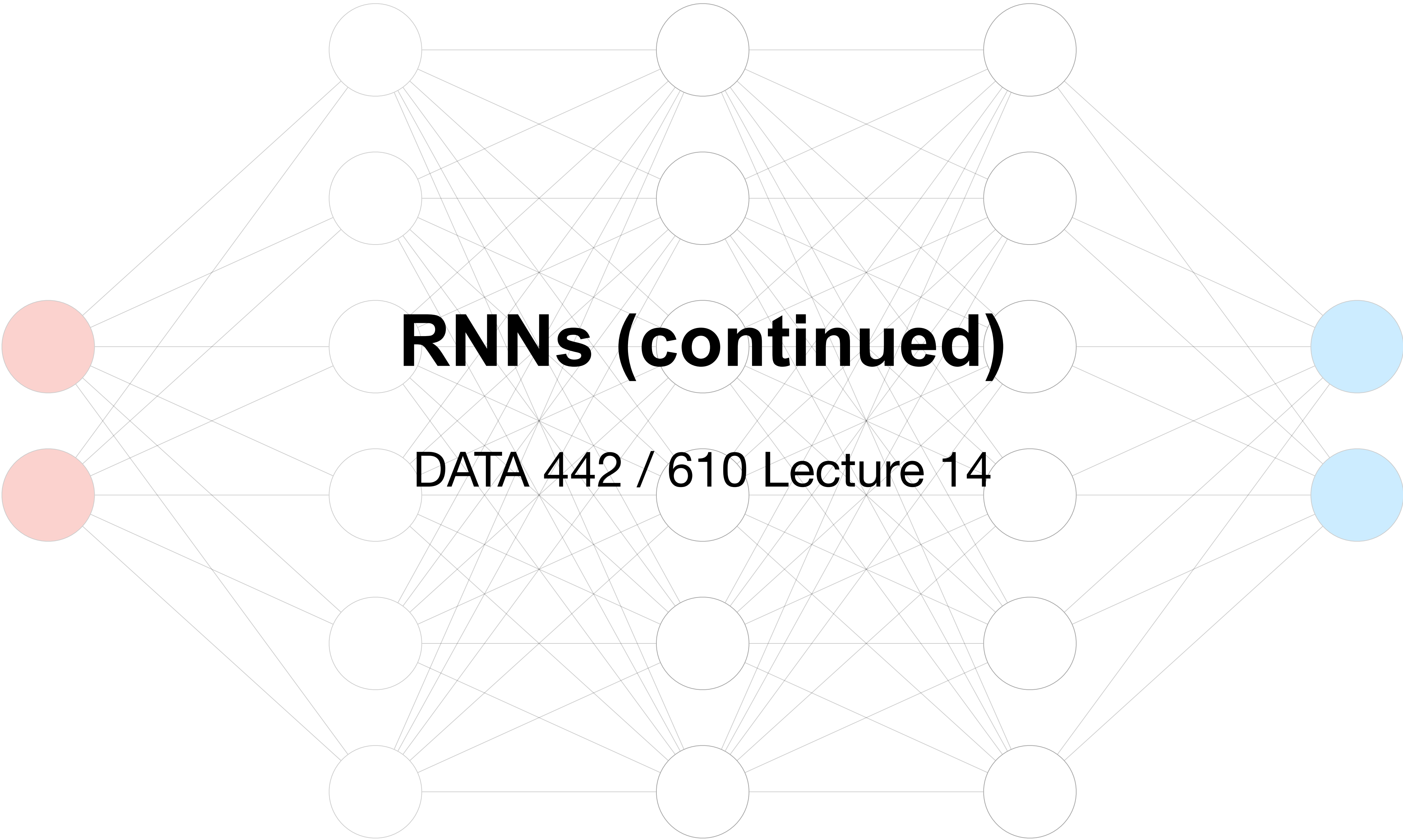




RNNs (continued)

DATA 442 / 610 Lecture 14

Outline

- Introduction to RNNs
- RNN classification revisited - IMDB sentiment dataset
- Time series forecasting

Reading: *Deep Learning*, Chapter 10

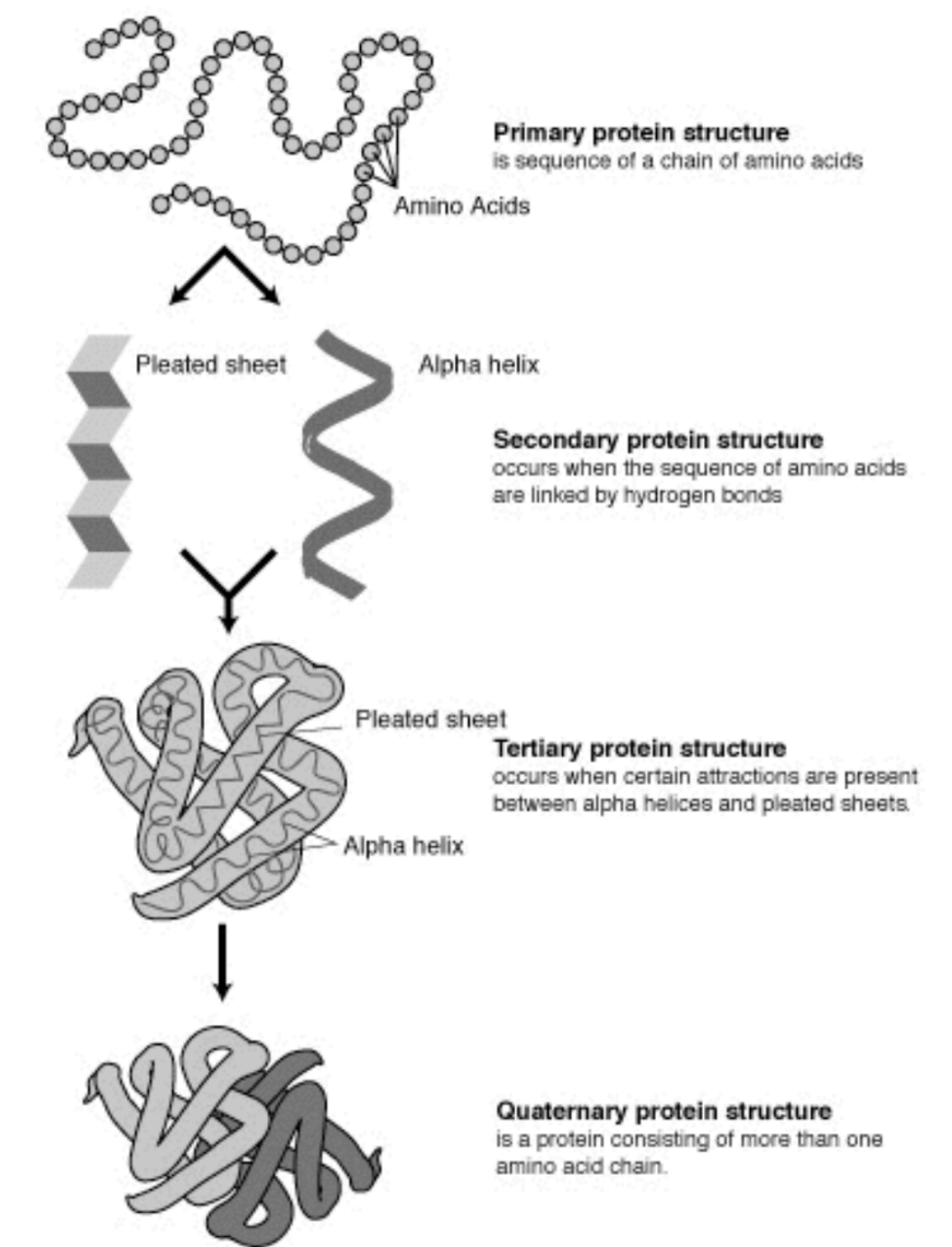
Sequential data

Sequential data appears in a particular order and can have arbitrary length

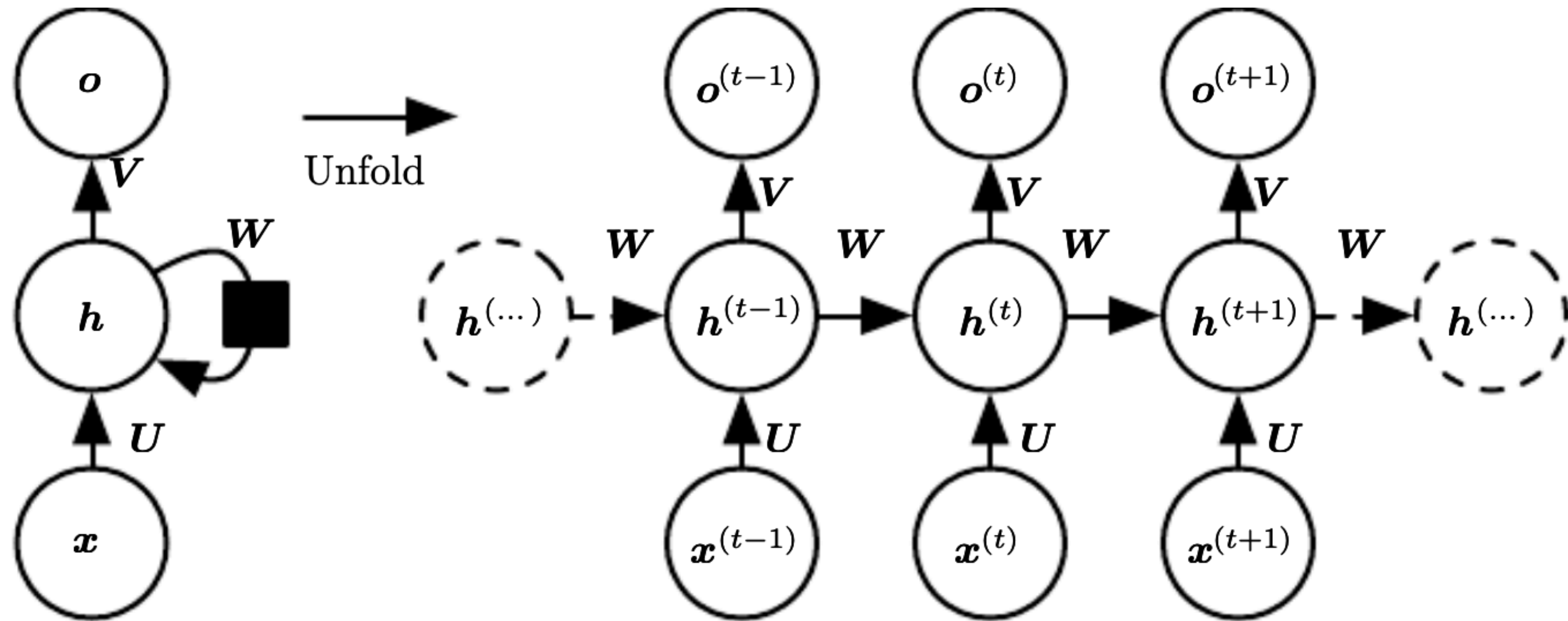
$$\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(t)}\} \quad t > 1$$

Examples:

- Time series data, e.g. stock prices each day, hour, minute
- Text, e.g. “Once upon a time there was a ...”
- DNA sequences, e.g. ACCTGCTGATTACA
- Amino acid chains (AlphaFold)

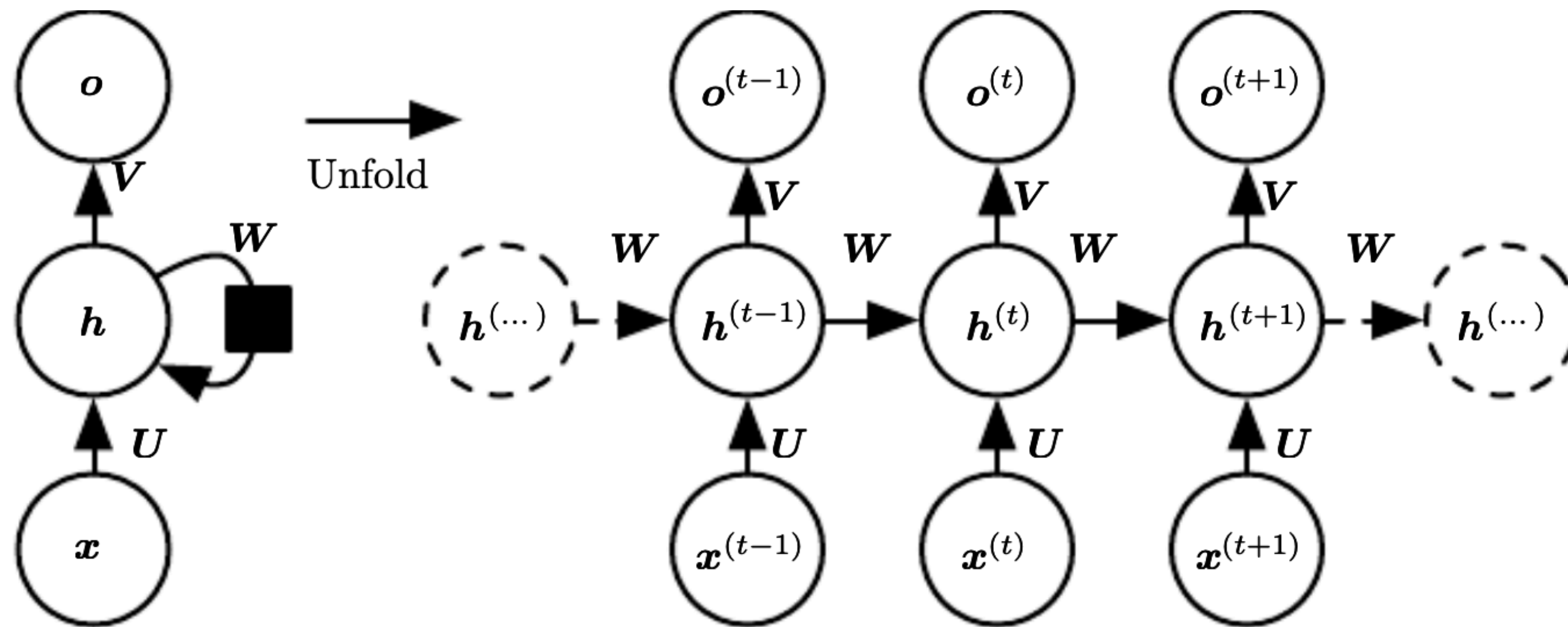


Recurrent neural networks



Recurrent neural networks share weights applied to both input sequence elements and the previous **hidden state**

RNN forward pass



(Ignore W , V , U I do not like this notation)

$W_{ab} = W_{a \rightarrow b}$ = weight matrix for element a influencing element b

Preactivation: $z^{(t)} = b_h + W_{hh} h^{(t-1)} + W_{xh} x^{(t)}$

Hidden state: $h^{(t)} = \tanh z^{(t)}$

Output: $b_o + W_{ho} h^{(t)}$

(Optional) $y^{(t)} = \sigma(o^{(t)})$ or other activation, or softmax

Typical RNN formulations

Sentiment analysis

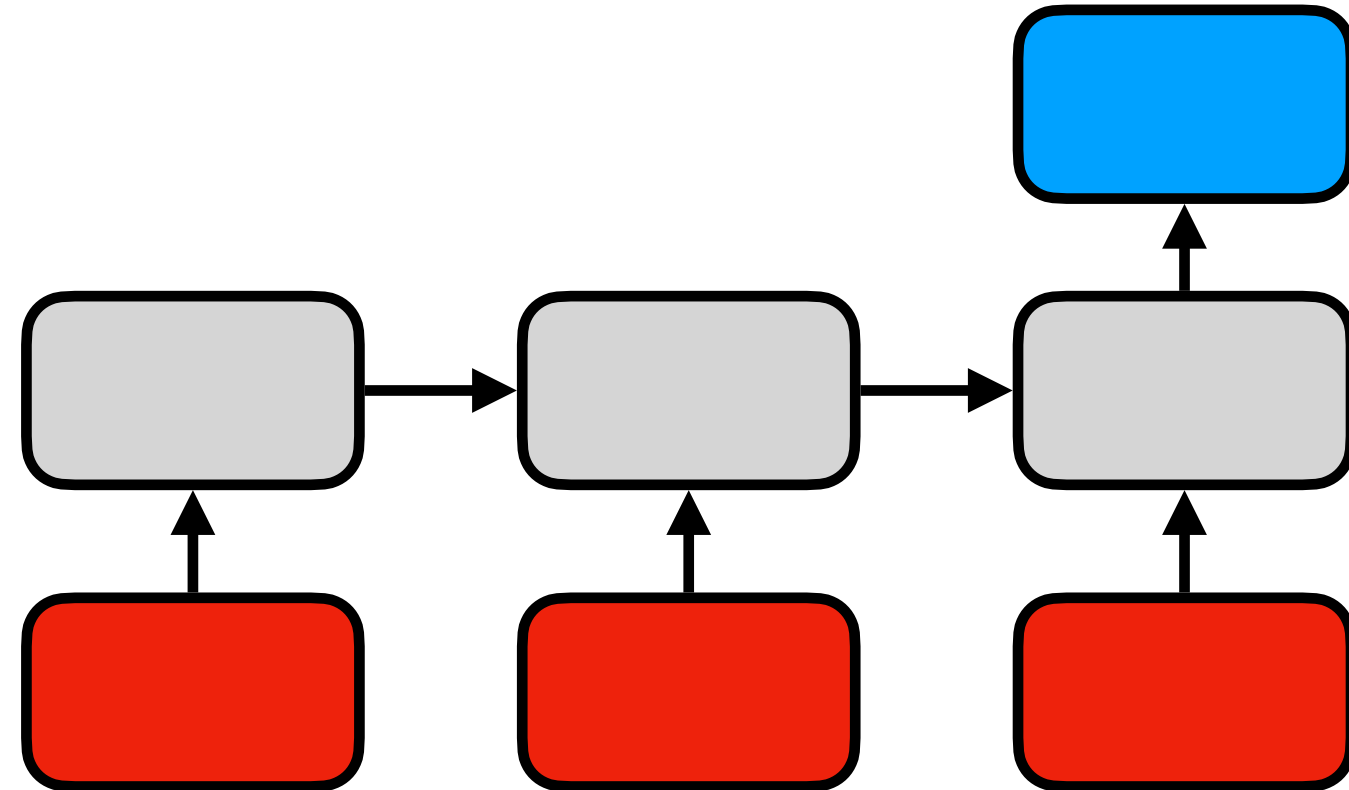
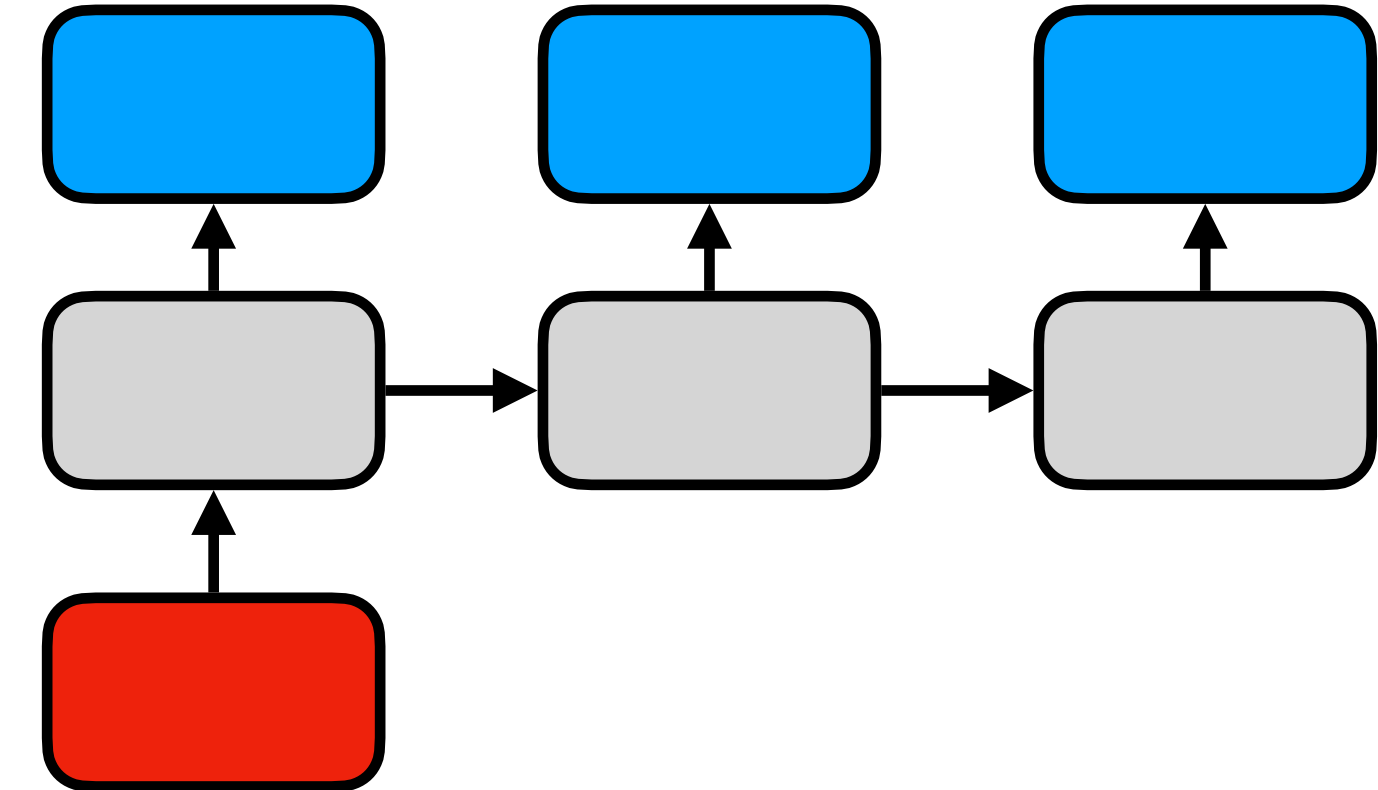
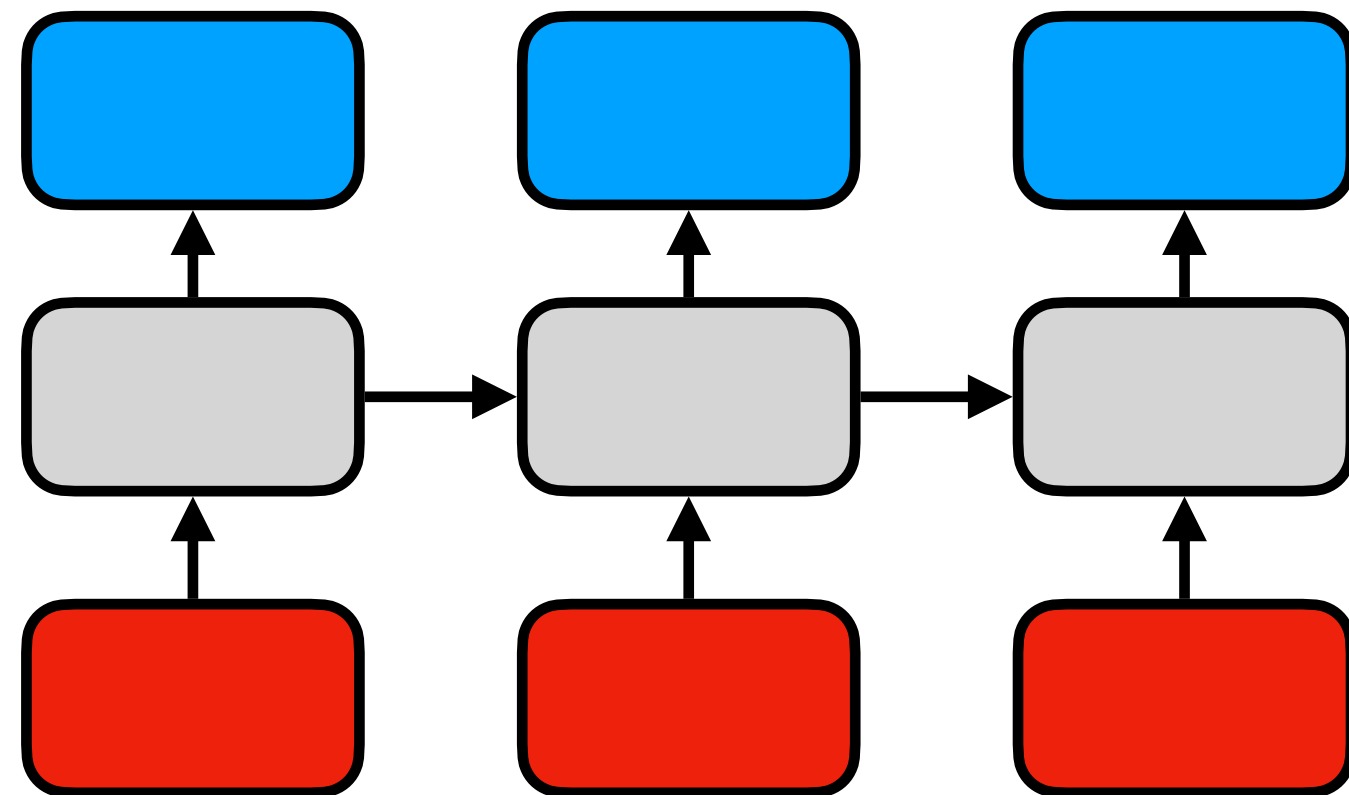


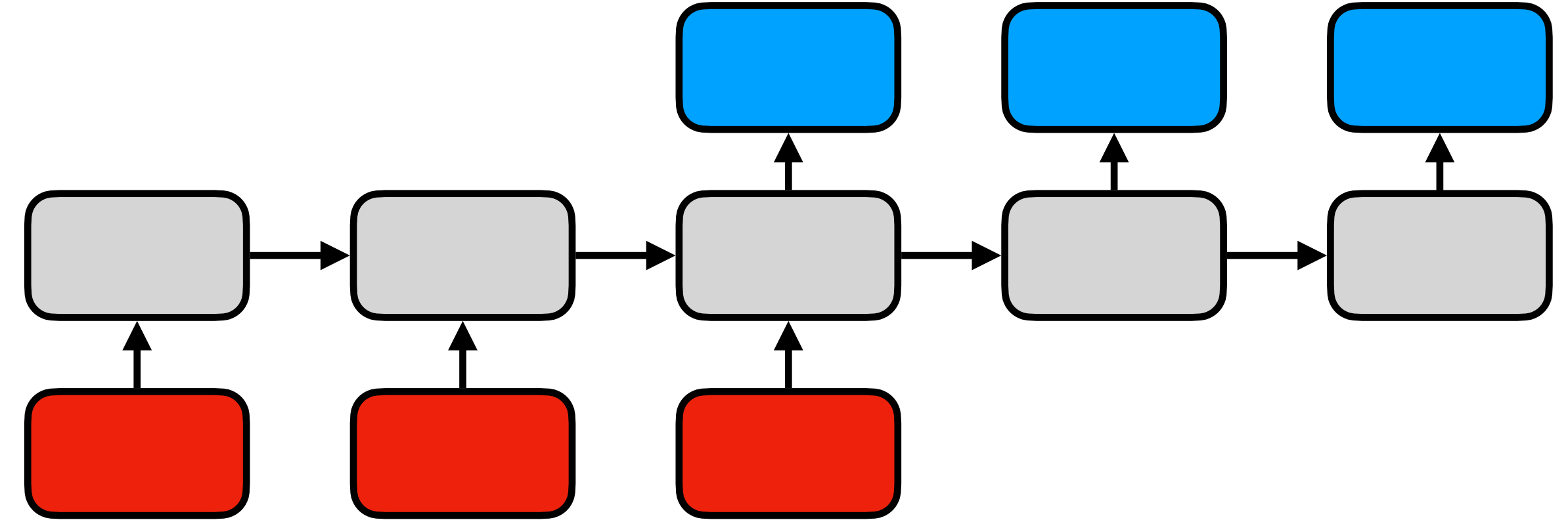
Image captioning



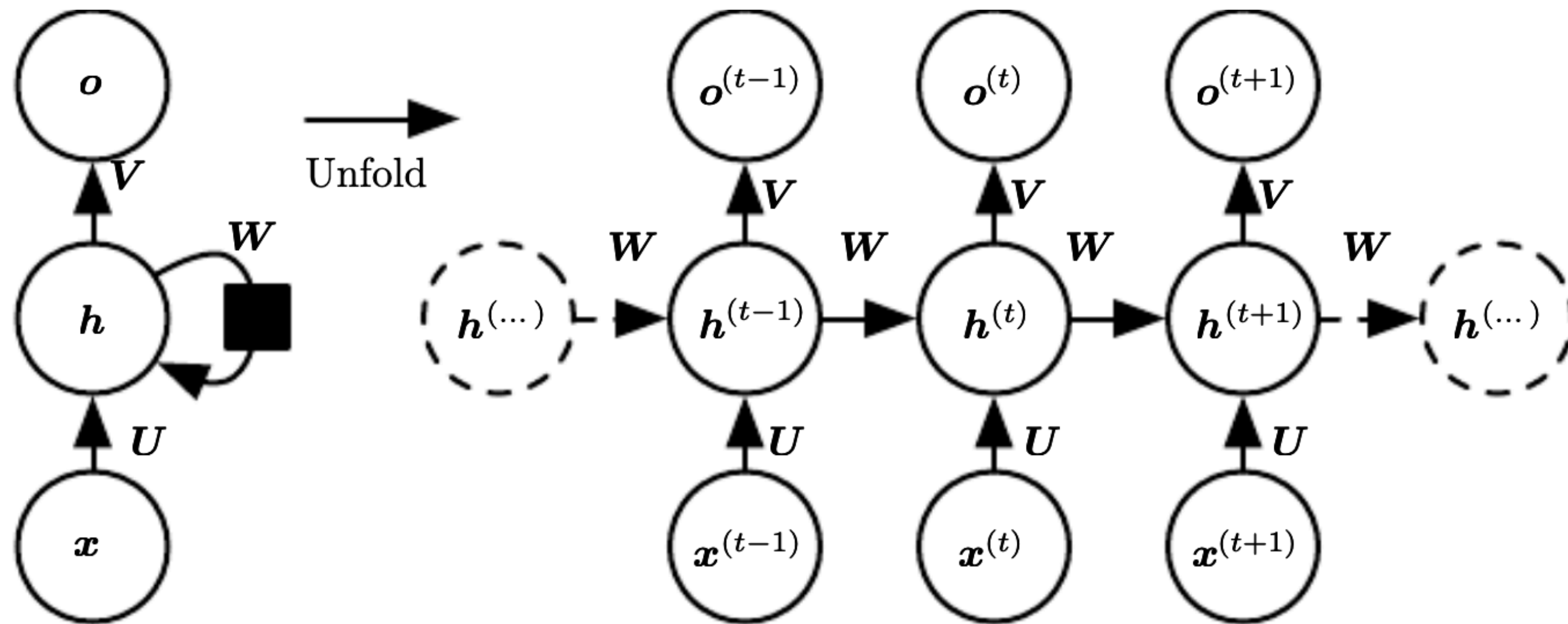
Video classification



Machine translation, text continuation



RNN backward pass



Just propagate backward through time, right?

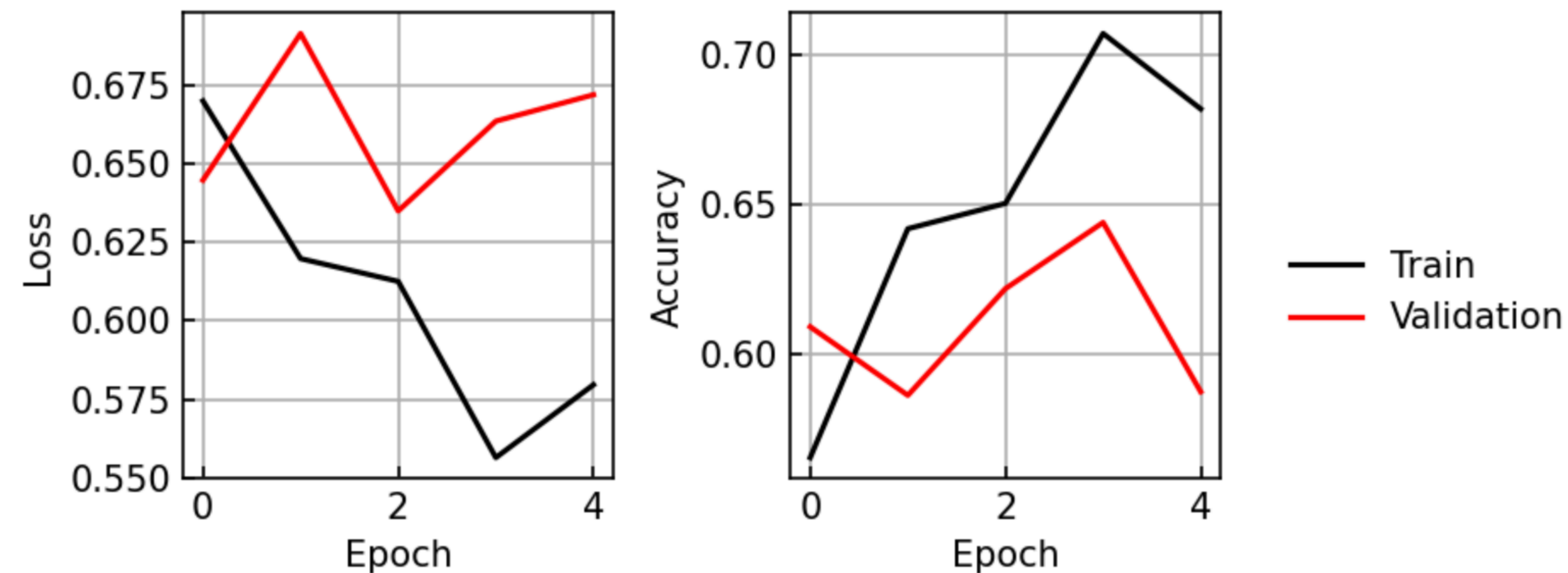
Because parameters are shared across time steps, gradients receive many contributions

$$\frac{\partial L}{\partial W_{hh}} = \sum_t \frac{\partial L}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial W_{hh}}$$

$$\frac{\partial L}{\partial h^{(t)}} = \frac{\partial L}{\partial h^{(t+1)}} \frac{\partial h^{(t+1)}}{\partial h^{(t)}} + \frac{\partial L}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}}$$

RNN training is **very** expensive...

A reminder of last time



IMDB Movie Sentiment Classification

▼ Preprocessing each sample

Tokenization

Vocabulary building

Sequence encoding

Putting it all together

Checking that it works

The neural network part

Comment: Being a "Wallace and Gromit-fan", I was looking forward for this full-length movie. Surprisingly I saw it at THE world-premiere in Vlissingen (NL), at the Film by the Sea festival. A wonderful feeling to be one of the first to see this very amusing and merry movie. It's about Wallace and Gromit (whom I believe don't need an introduction) having their own pest-control company in the city which is hosting a giant-vegetable contest in a few days. Everyone, including an eccentric baroness, is hoping his or her giant carrot or melon will win the Golden carrot. Unfortunately the town is plagued by lots of hungry rabbits. This is where W&G come in. They have their own cracking contraptions to control these cute creatures in a human way.

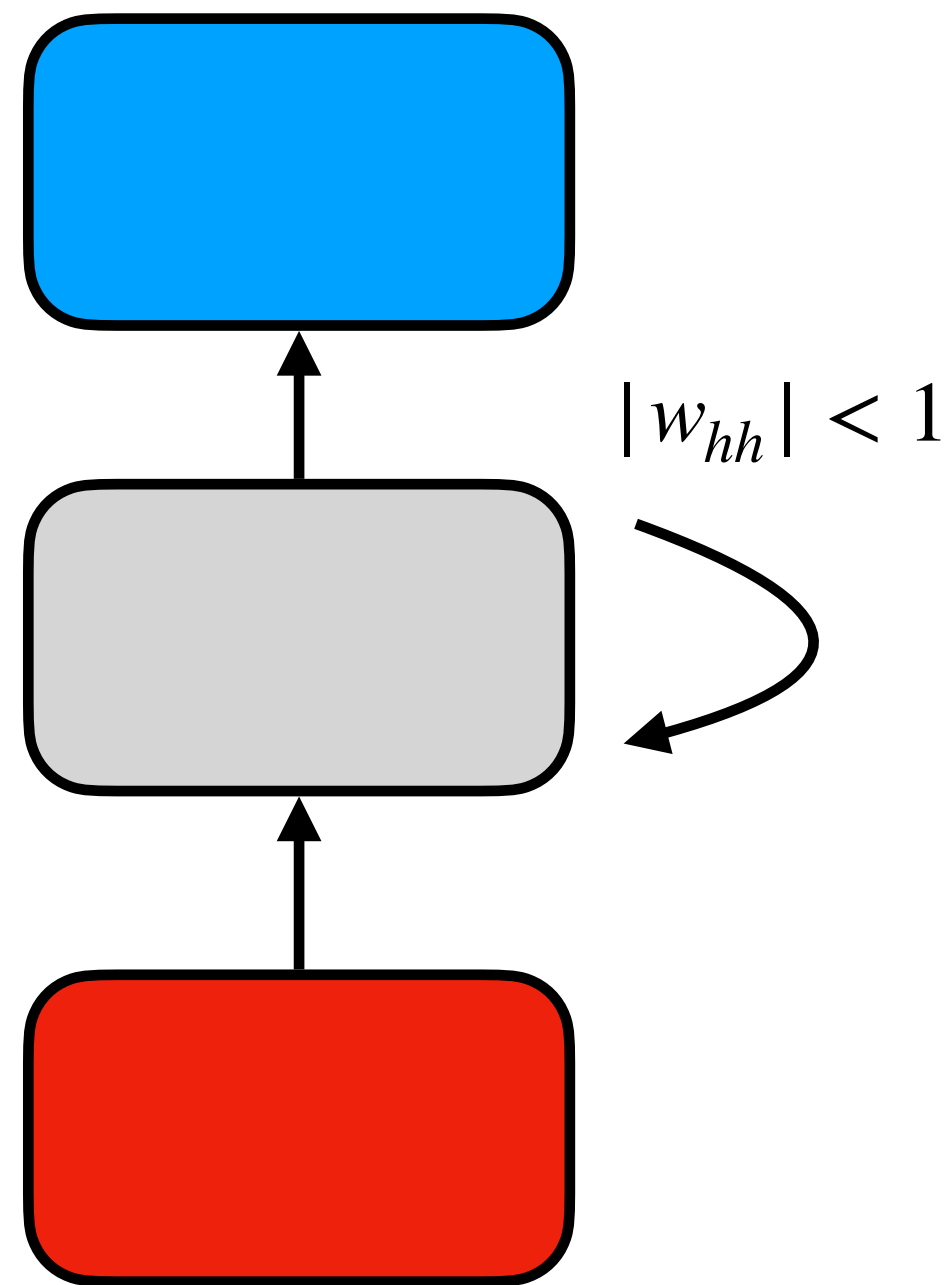
It's a very funny and colorful story. Anyone who liked the three preceding short movies of W&G (which are more than great!), will love this full-length movie. Nick Park really delivered a wonderful and original result with a great sense for humor. Like in Chicken Run, it truly amazes me how he can capture so much story and emotions in just a few frames. "Job well done, lad" ;-)

Oh yeah: The music was fantastic! It really completes the ride. Enjoy!

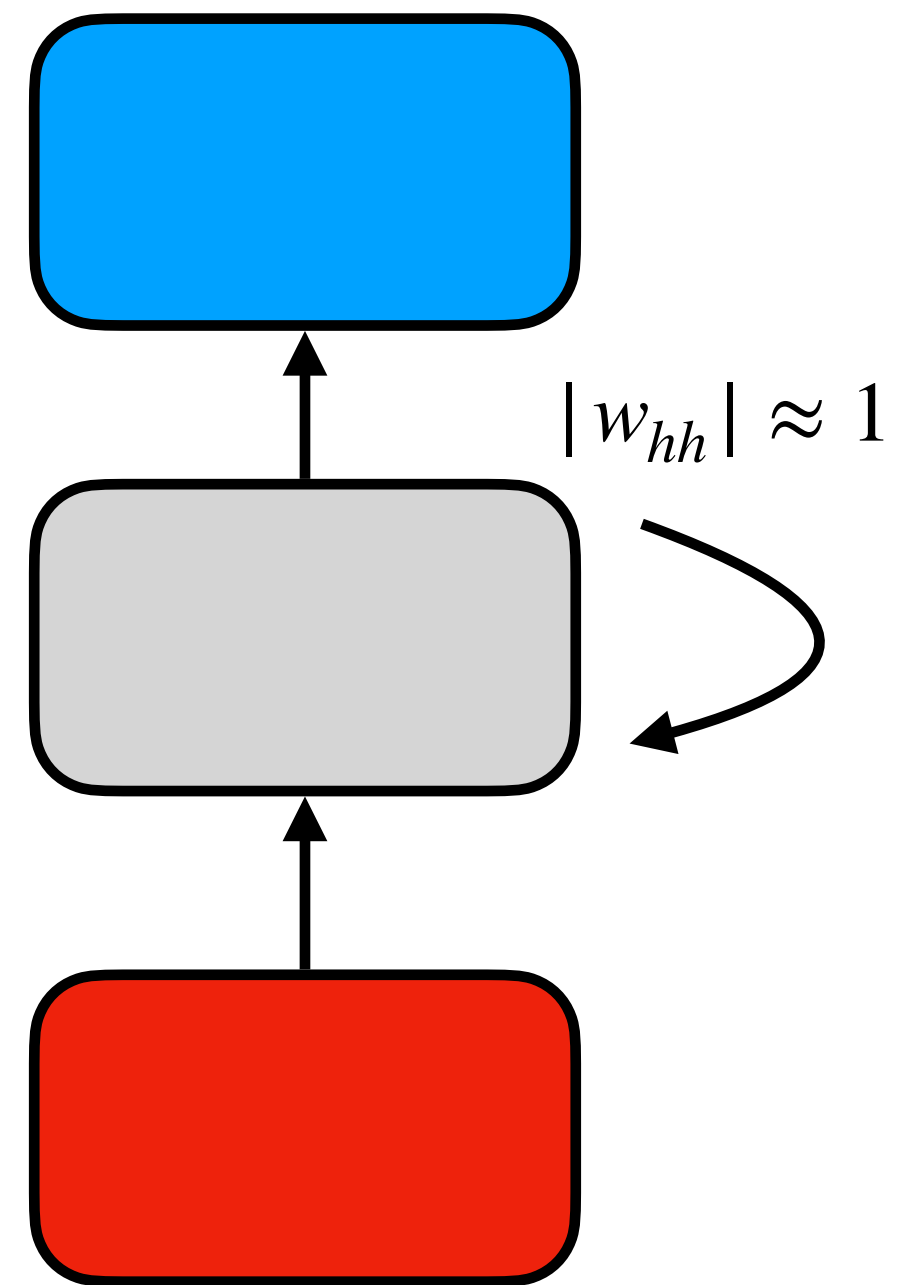
True sentiment: 1
Predicted probability: [0.3387 0.6613]

Challenges with long sequences

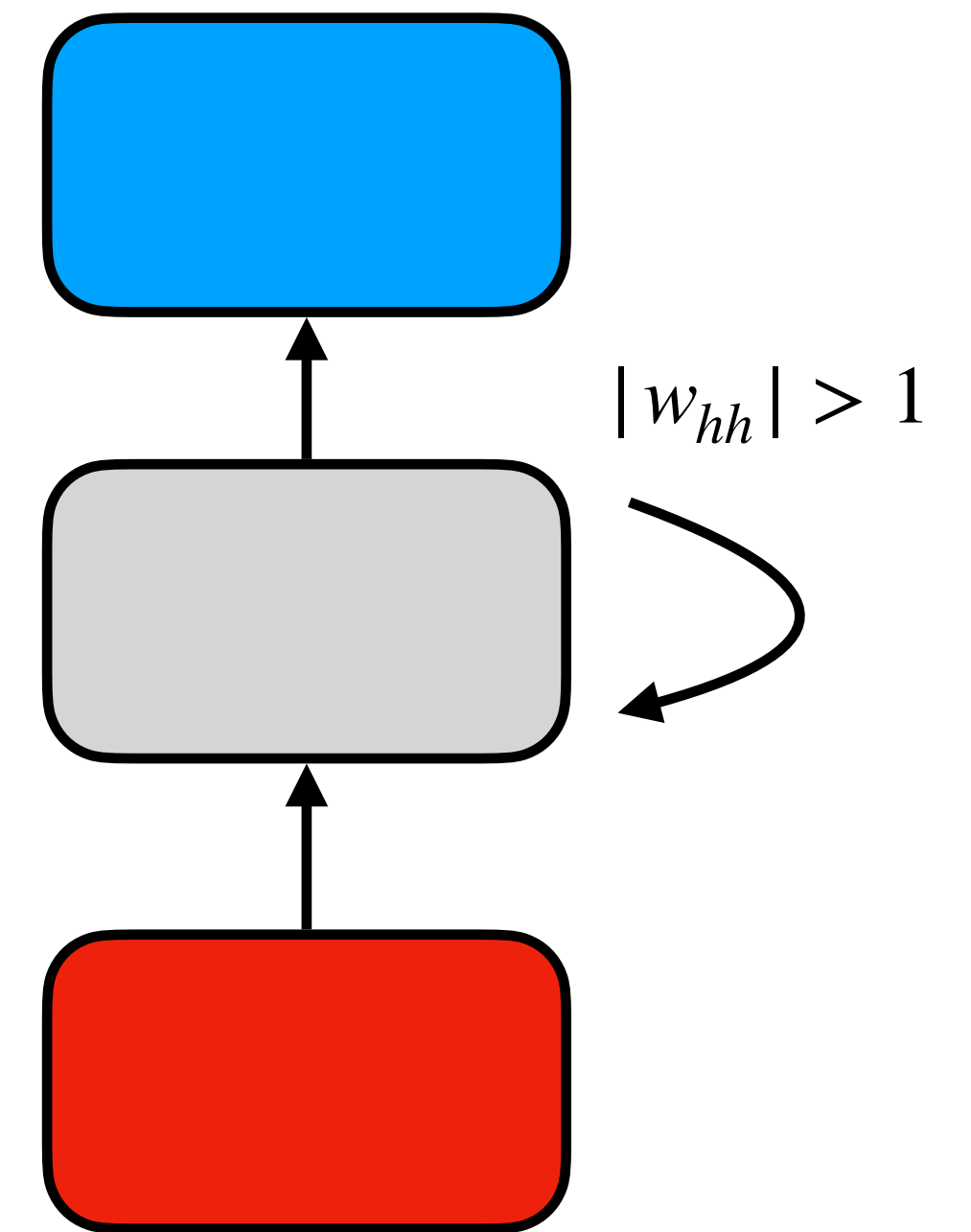
Vanishing gradient



Goldilocks zone



Exploding gradient



Strategies to handle vanishing/exploding gradients

Gradient clipping: `torch.nn.utils.clip_grad_norm_(model.parameters(), max_gradient_value)`

Truncated Backpropagation through time: don't backprop through the whole sequence

Long short-term memory (LSTM): gated connections allow the *model* to select whether information is propagated through time

More strategies

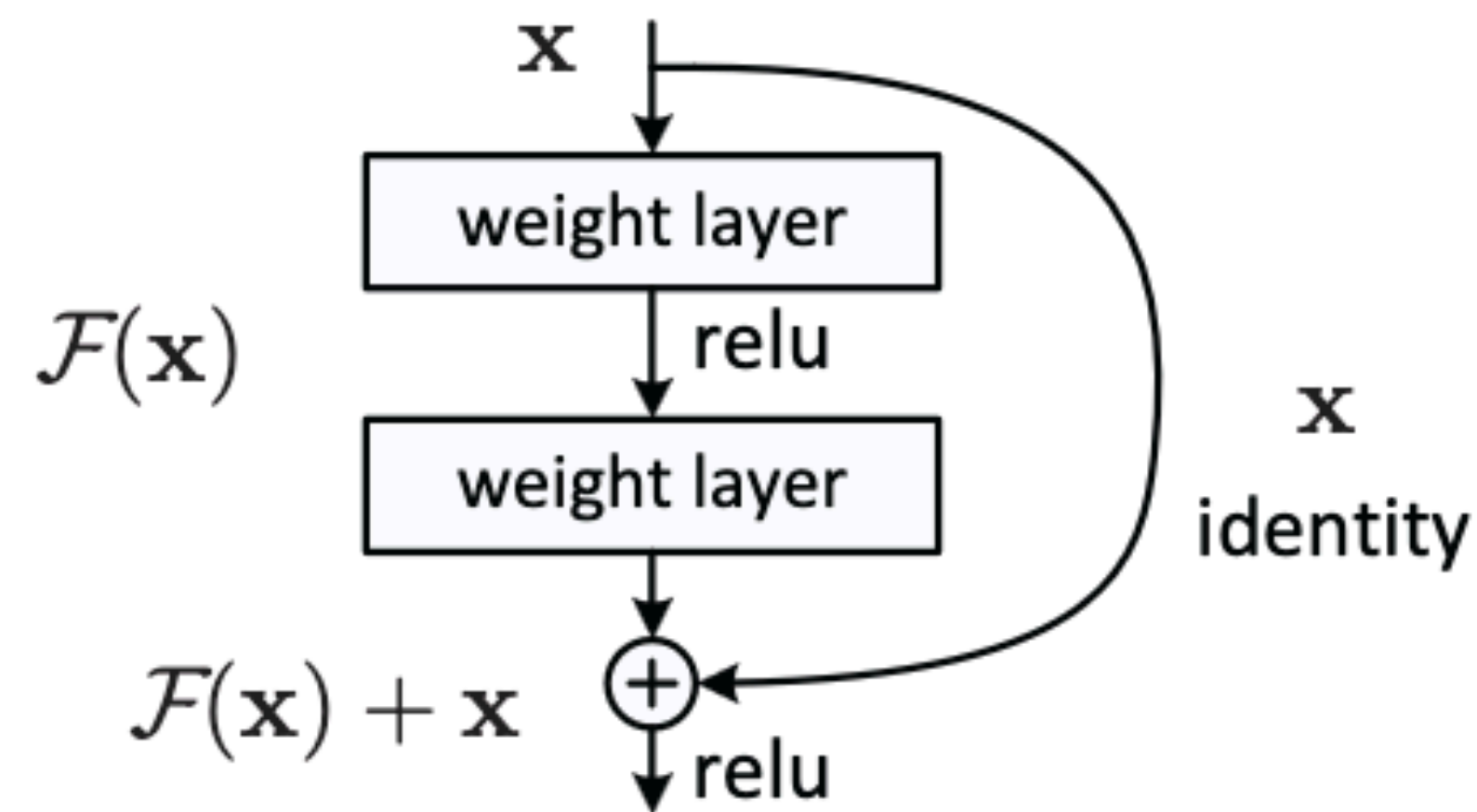


Figure 2. Residual learning: a building block.

Skip connections through time

$$h^{(t)} = \tanh \left(W_{dh} h^{(t-d)} + W_{hh} h^{(t-1)} + W_{ih} x^{(t)} \right) \quad d > 1$$

Leaky units

$$h^{(t)} = \alpha \tilde{h}^{(t)} + (1 - \alpha) h^{(t-1)}$$

Long Short Term Memory

Gating functions

$$i^{(t)} = \sigma \left(W_{ii}x^{(t)} + b_{ii} + W_{hi}h^{(t-1)} + b_{hi} \right)$$

$$f^{(t)} = \sigma \left(W_{if}x^{(t)} + b_{if} + W_{hf}h^{(t-1)} + b_{hf} \right)$$

$$o^{(t)} = \sigma \left(W_{io}x^{(t)} + b_{io} + W_{ho}h^{(t-1)} + b_{ho} \right)$$

Memory cell input

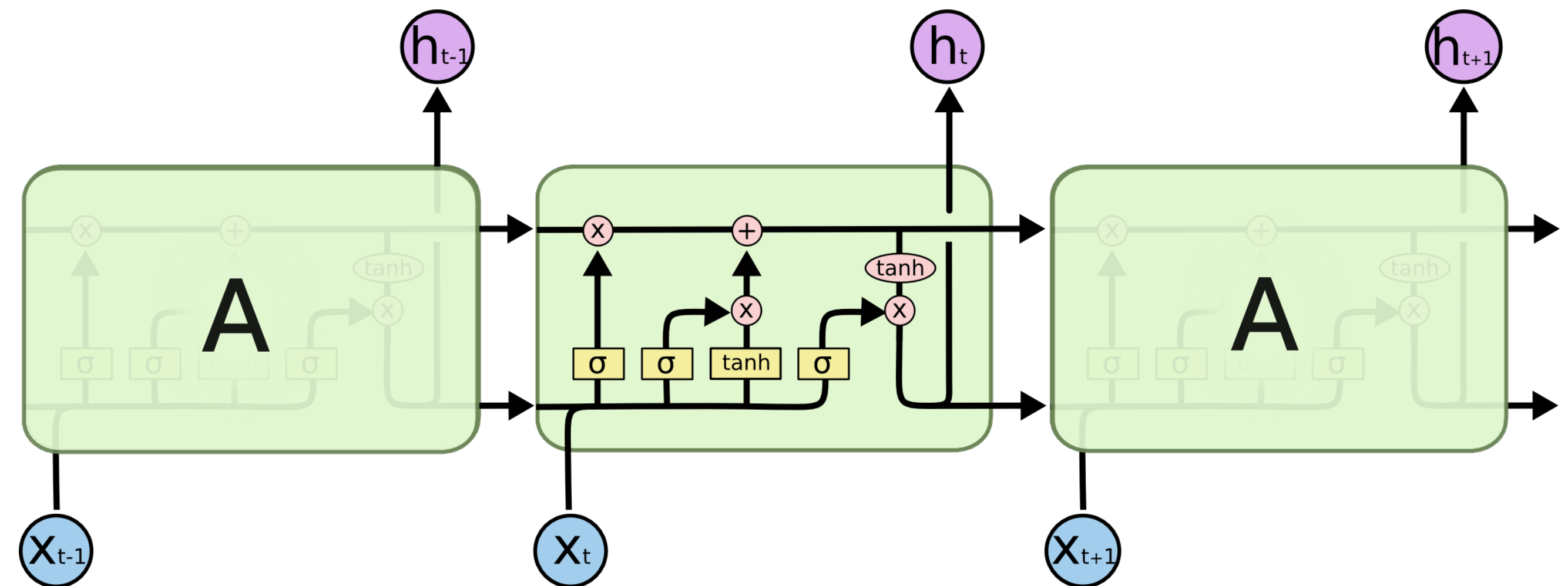
$$g^{(t)} = \tanh \left(W_{ig}x^{(t)} + b_{ig} + W_{hg}h^{(t-1)} + b_{hg} \right)$$

Memory cell update

$$c^{(t)} = f^{(t)}c^{(t-1)} + i^{(t)}g^{(t)}$$

Hidden layer update

$$h^{(t)} = o^{(t)} \tanh c^{(t)}$$

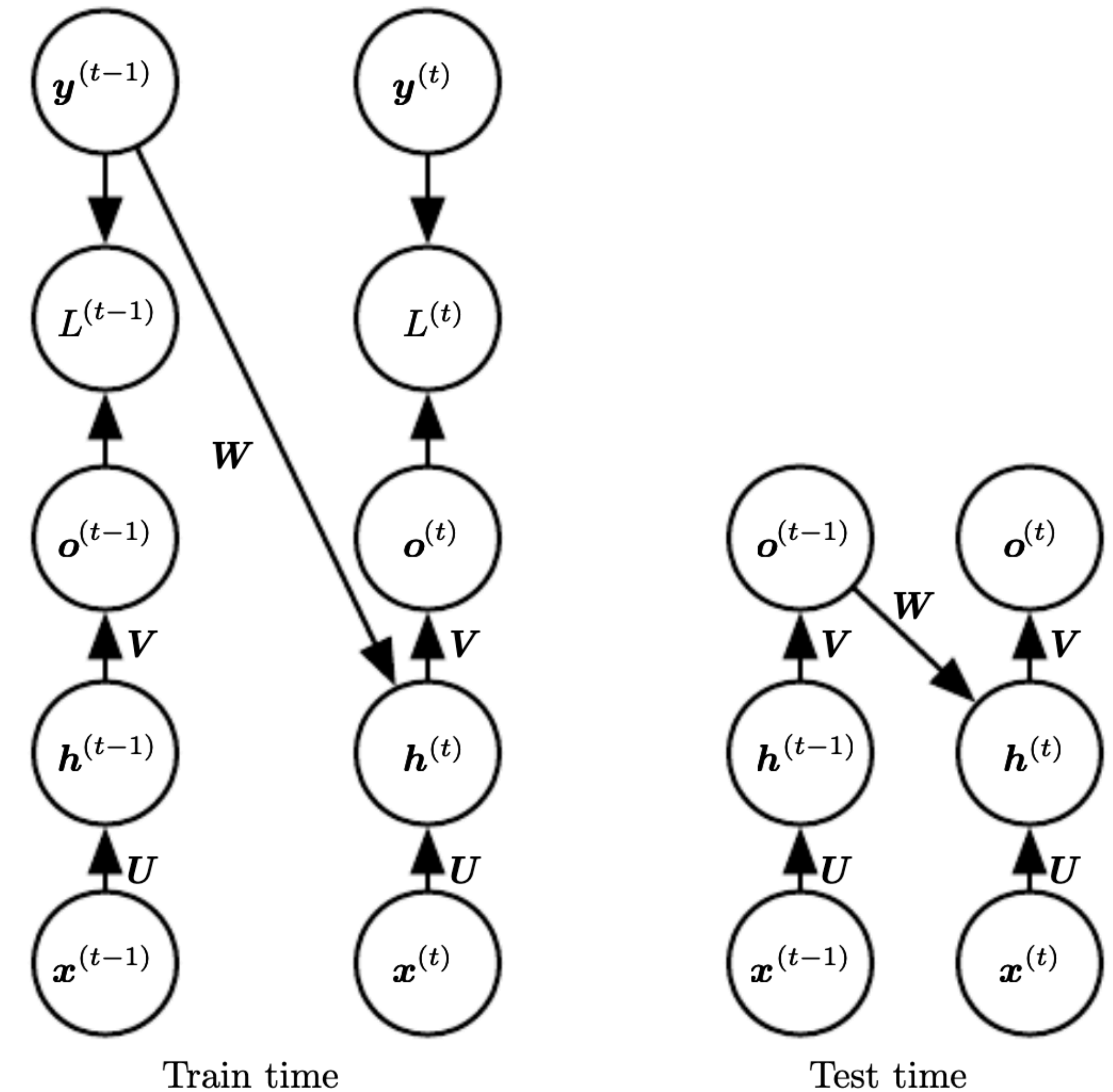


Hands-on with LSTM



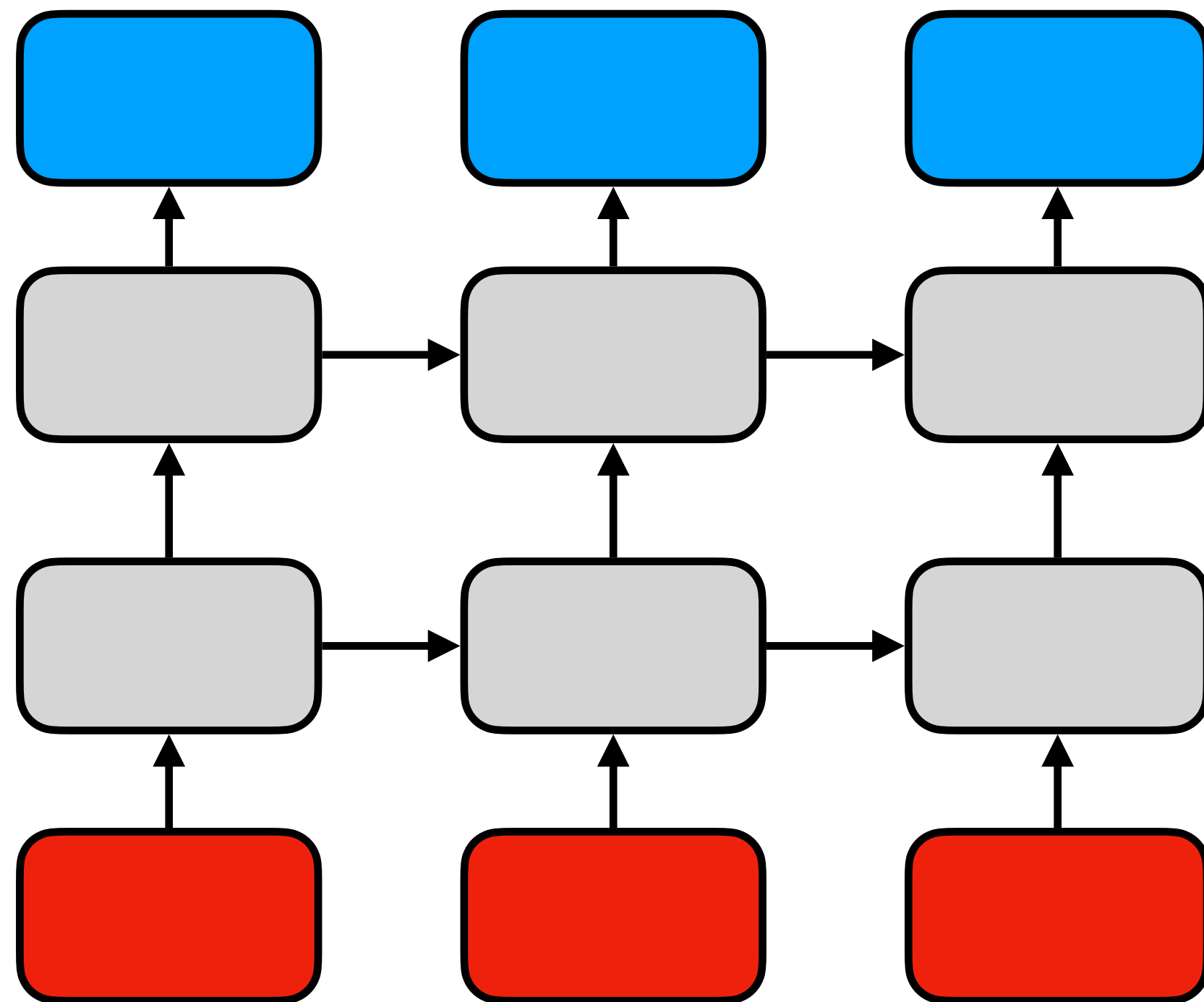
Teacher forcing: output-to-hidden recurrence

- Hidden-to-hidden training is slow because hidden states must be computed sequentially
- Output-to-hidden uses ground-truth values during training and model outputs during inference
- Gradients at each time step can be computed in parallel

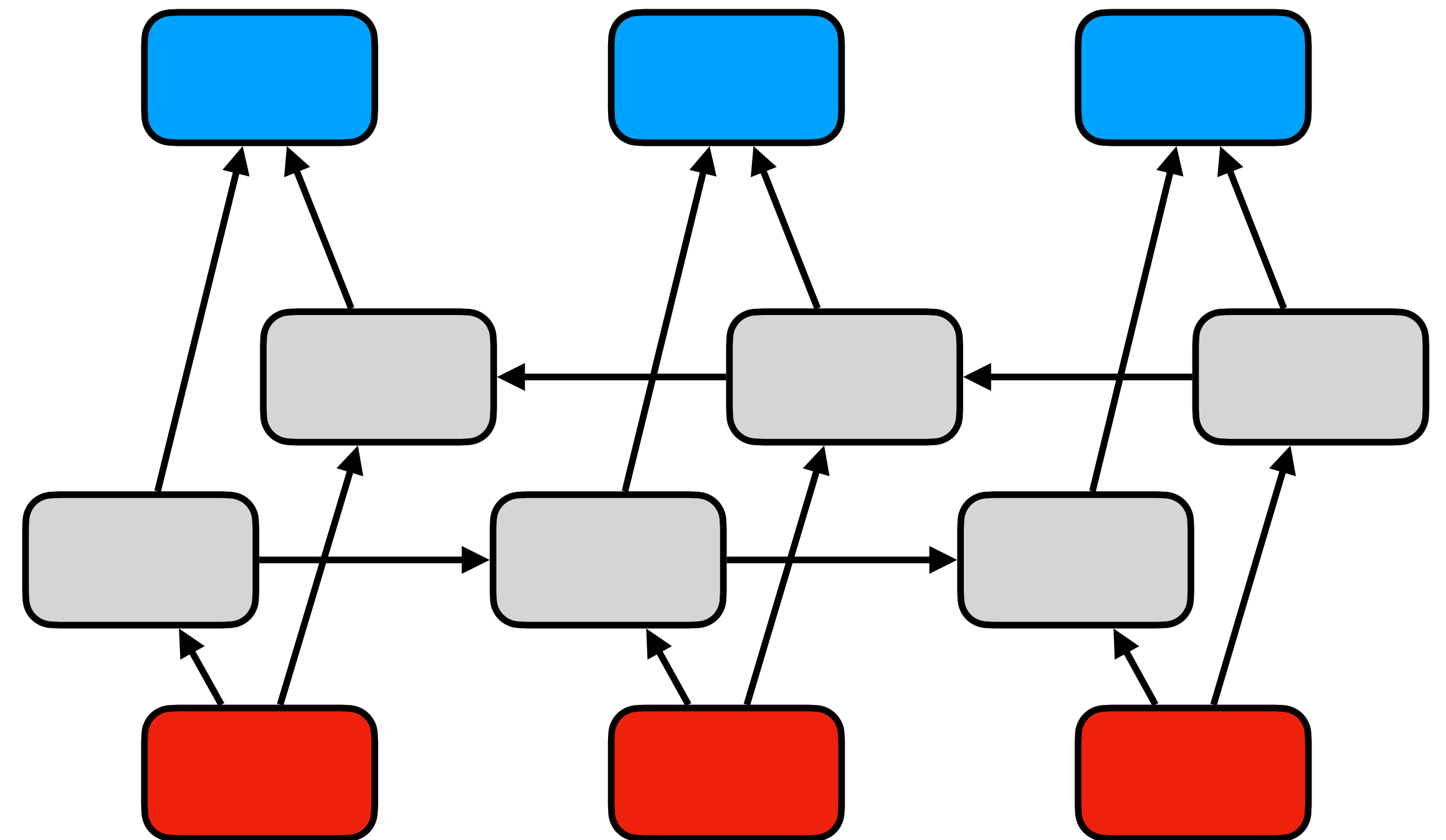


More RNN formulations

Deep RNN



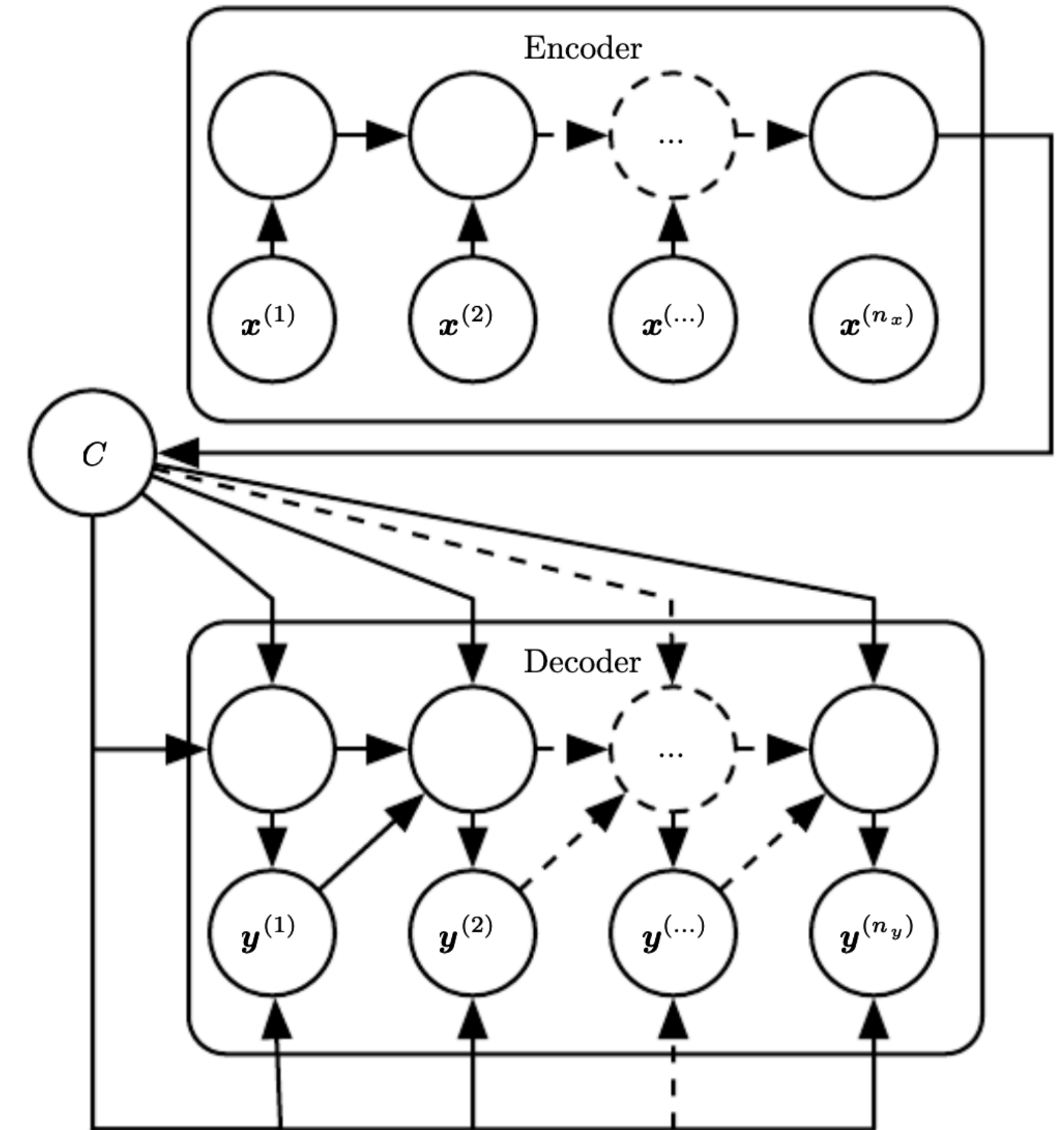
Bi-directional RNN



RNN for machine translation: Seq2seq

- Sequence to sequence aims to generate an arbitrary-length output from an arbitrary-length input
- An **encoder** transforms the input sequence into a **context variable** that holds relevant information
- A **decoder** uses the context variable to generate an output sequence
- Encoder/decoder trained to jointly maximize

$$\log p \left(y^{(1)}, \dots, y^{(n)} \mid x^{(1)}, \dots, x^{(m)} \right)$$



That's all!

- Slides will be posted on the course website: <https://colen-teaching.github.io/DATA442-Fall-2026>
- Lab demos will also be posted
- **PS2** due October 23
- **REMEMBER**: TA Office Hours Tuesday 12-1pm, Student Success Center