Recurrent Neural Networks

CMPUT 366: Intelligent Systems

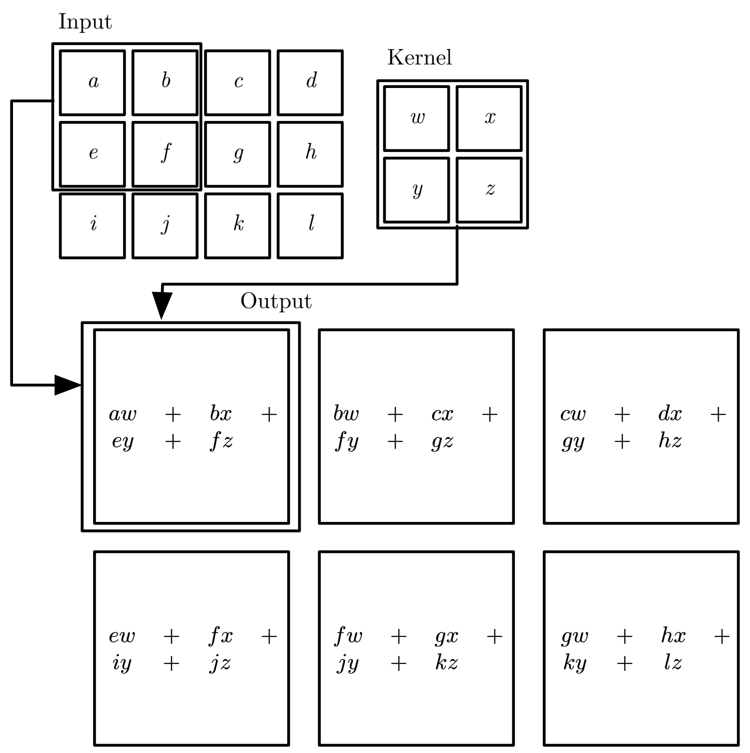
P&M §10.0-10.2, 10.10

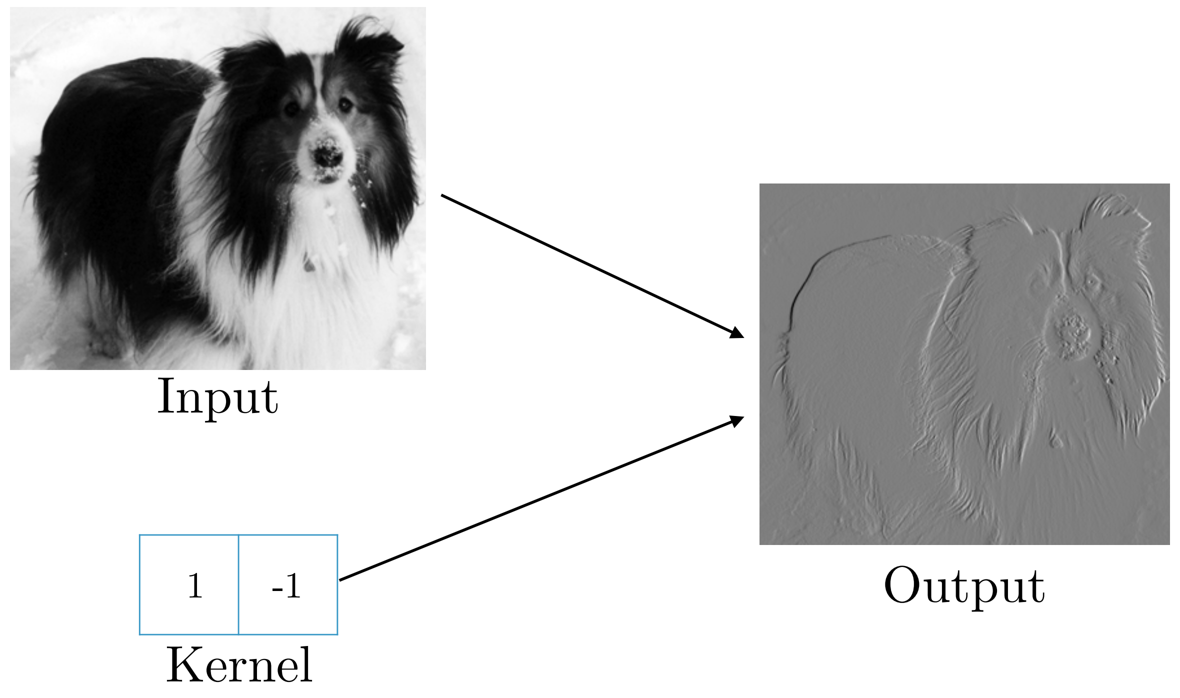
Lecture Outline

1. Recap
2. Unfolding Computations
3. Recurrent Neural Networks
4. Long Short-Term Memory

Recap: Convolutional Neural Networks

- Convolutional networks: Specialized architecture for **images**
- Number of **parameters** controlled by using **convolutions** and **pooling** operations instead of **dense connections**
- Fewer parameters means more **efficient to train**





(Images: Goodfellow 2016)

Sequence Modelling

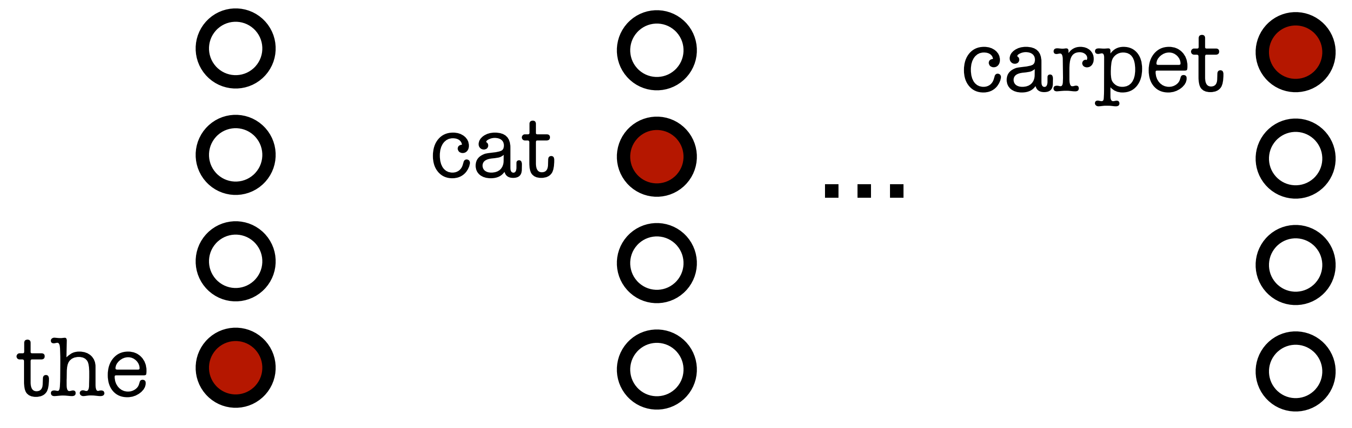
- For many tasks, especially involving language, we want to model the behaviour of **sequences**
- **Example:** Translation
  - The cat is on the carpet $\implies$ Le chat est sur le tapis
- **Example:** Sentiment analysis
  - This pie is great $\implies$ POSITIVE
  - This pie is okay, not great $\implies$ NEUTRAL
  - This pie is not okay $\implies$ NEGATIVE

Sequential Inputs

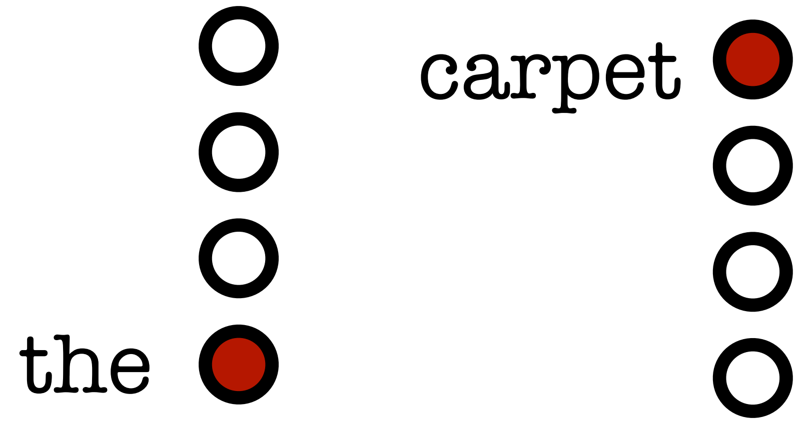
The cat is on the carpet

Question: How should we represent sequential input to a neural network?

1. 1-hot vector for **each word**
   (Sequence must be a particular length)

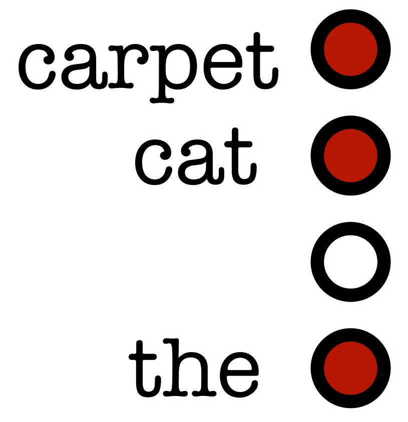


2. 1-hot vector for **last few words**
   (n-gram)



3. **Single vector** indicating each word that is present
   (bag of words)

4. Single vector summing the **semantic embeddings** of all the words

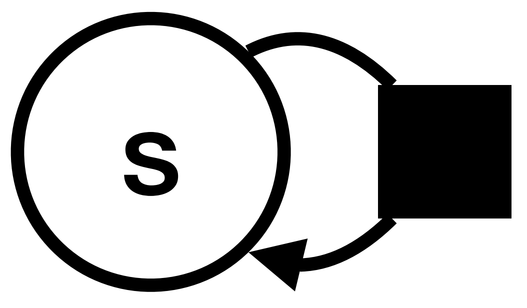


Dynamical Systems

- A **dynamical system** is a system whose state at time $t + 1$ depends on its state at time t:

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}; \theta)$$

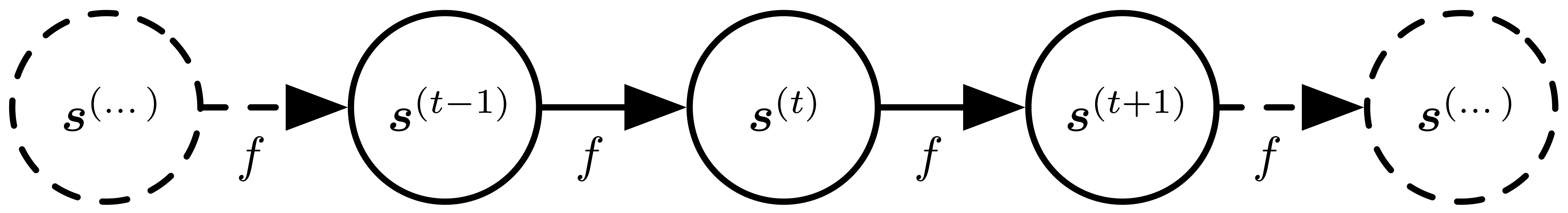
- An expression that depends on the same expression at an earlier time is **recurrent**.



Unfolding Computations

- A recurrent expression can be converted to a non-recurrent expression by **unfolding**:

$$
\begin{aligned}
\mathbf{s}^{(3)} &= f(\mathbf{s}^{(2)}; \theta) \\
&= f(f(\mathbf{s}^{(1)}; \theta); \theta)
\end{aligned}
$$



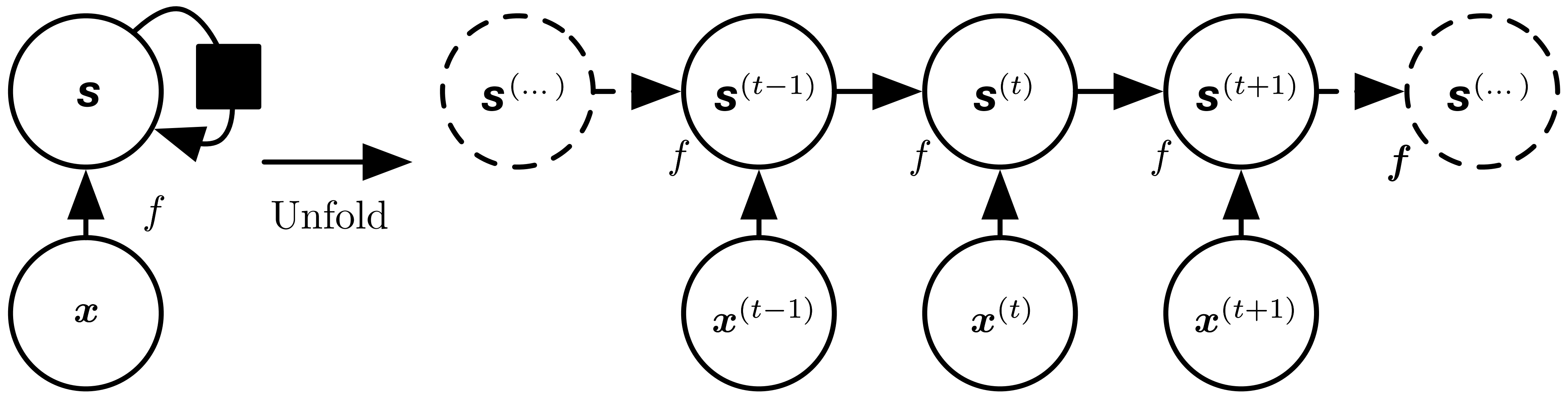
(Image: Goodfellow 2016)

External Signals

- Dynamical systems can also be driven by **external signals**:

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}, \mathbf{x}^{(t)}; \theta)$$

- These systems can also be represented by non-recurrent, unfolded computations:

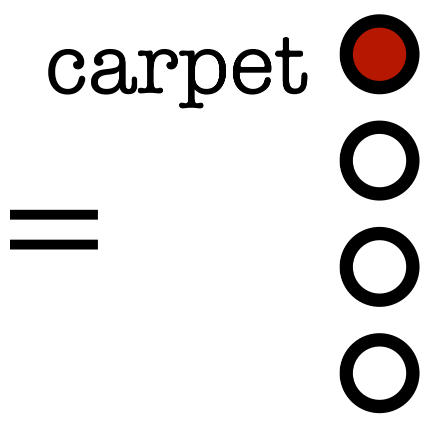


(Image: Goodfellow 2016)

Recurrent Neural Networks

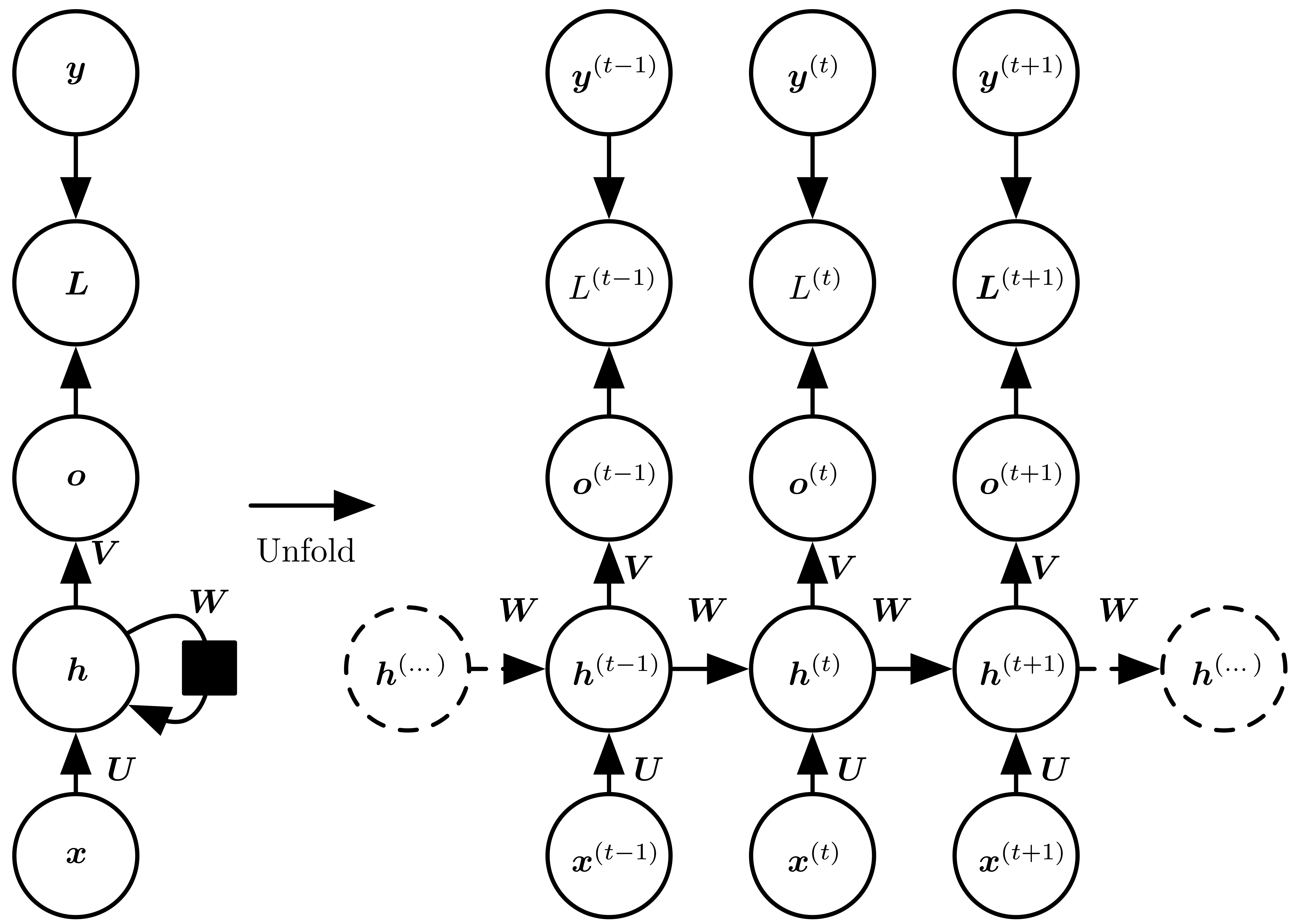
- Recurrent neural network: a specialized architecture for modelling **sequential data**
- Input presented **one element at a time**

$$\mathbf{x}^{(6)} =$$



- Parameter sharing by:
  - Treating the sequence as a system with **state**
  - Introducing hidden layers that **represent** state
  - Computing **state transitions** and **output** using **same functions** at each stage

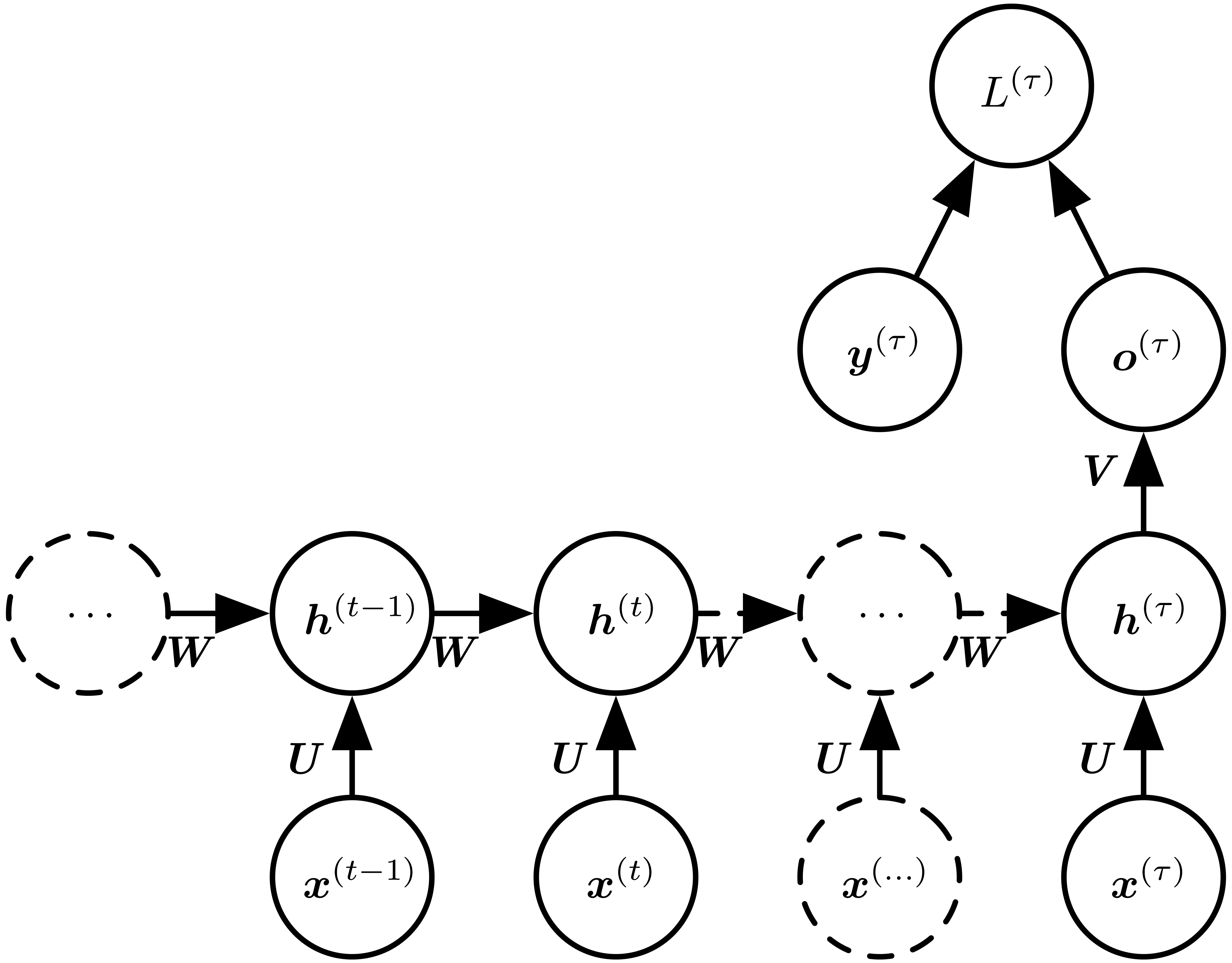
Recurrent Hidden Units: Sequence to Sequence



- **Input values** **x** connected to **hidden state** **h** by weights $\mathbf{U}$
- Hidden state **h** mapped to **output** **o** by weights $\mathbf{V}$
- Hidden state $\mathbf{h}^{(t-1)}$ connected to hidden state $\mathbf{h}^{(t)}$ by weights $\mathbf{W}$
- Gradients computed by **back propagation through time**: from final loss all the way back to initial input.
- All hidden states computed must be **stored** for computing gradients

(Image: Goodfellow 2016)

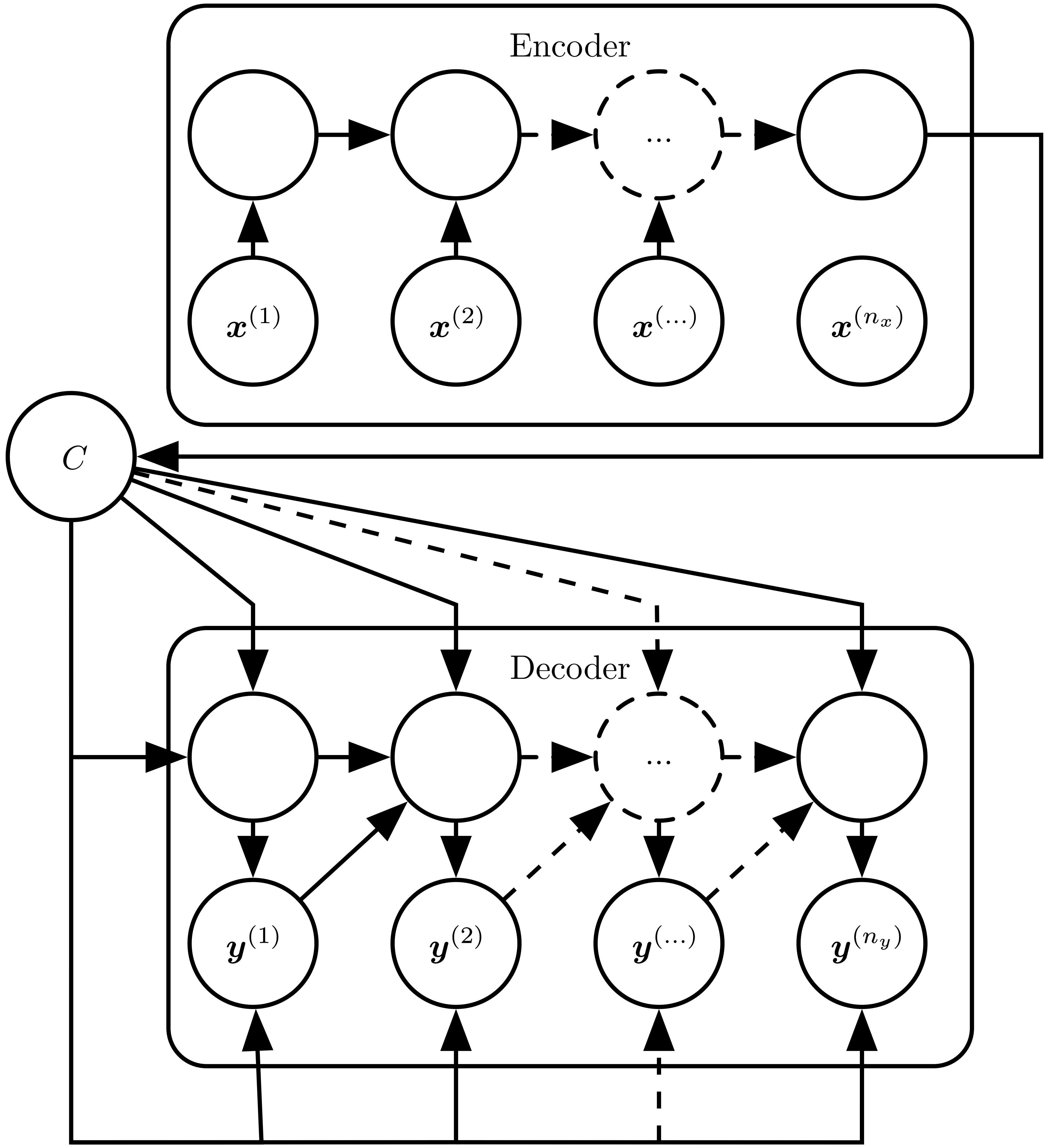
Recurrent Hidden Units: Sequence to Single Output



- Update state as inputs are provided
- Only compute a **single** output at the **end**
- $\mathbf{W}, \mathbf{U}$ still shared at every stage
- Back propagation through time still requires **evaluating every state** in gradient computation

(Image: Goodfellow 2016)

Encoder/Decoder Architecture for Sequence to Sequence

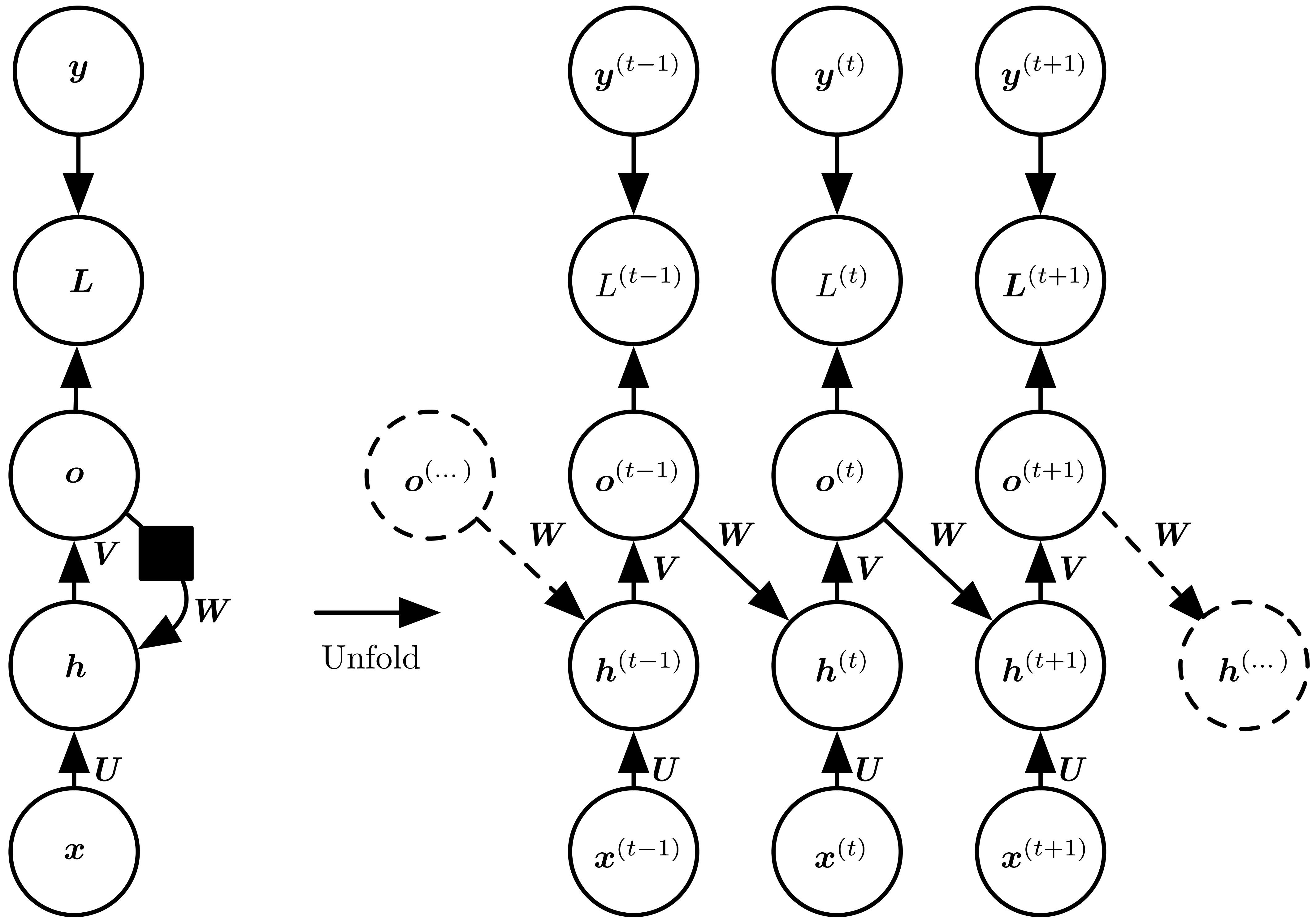


Can **combine approaches** for sequence-to-sequence:

1. Accept entire input to construct a single "**context**" output $\mathbf{C}$
2. Construct new sequence using context $\mathbf{C}$ as **only input**

(Image: Goodfellow 2016)

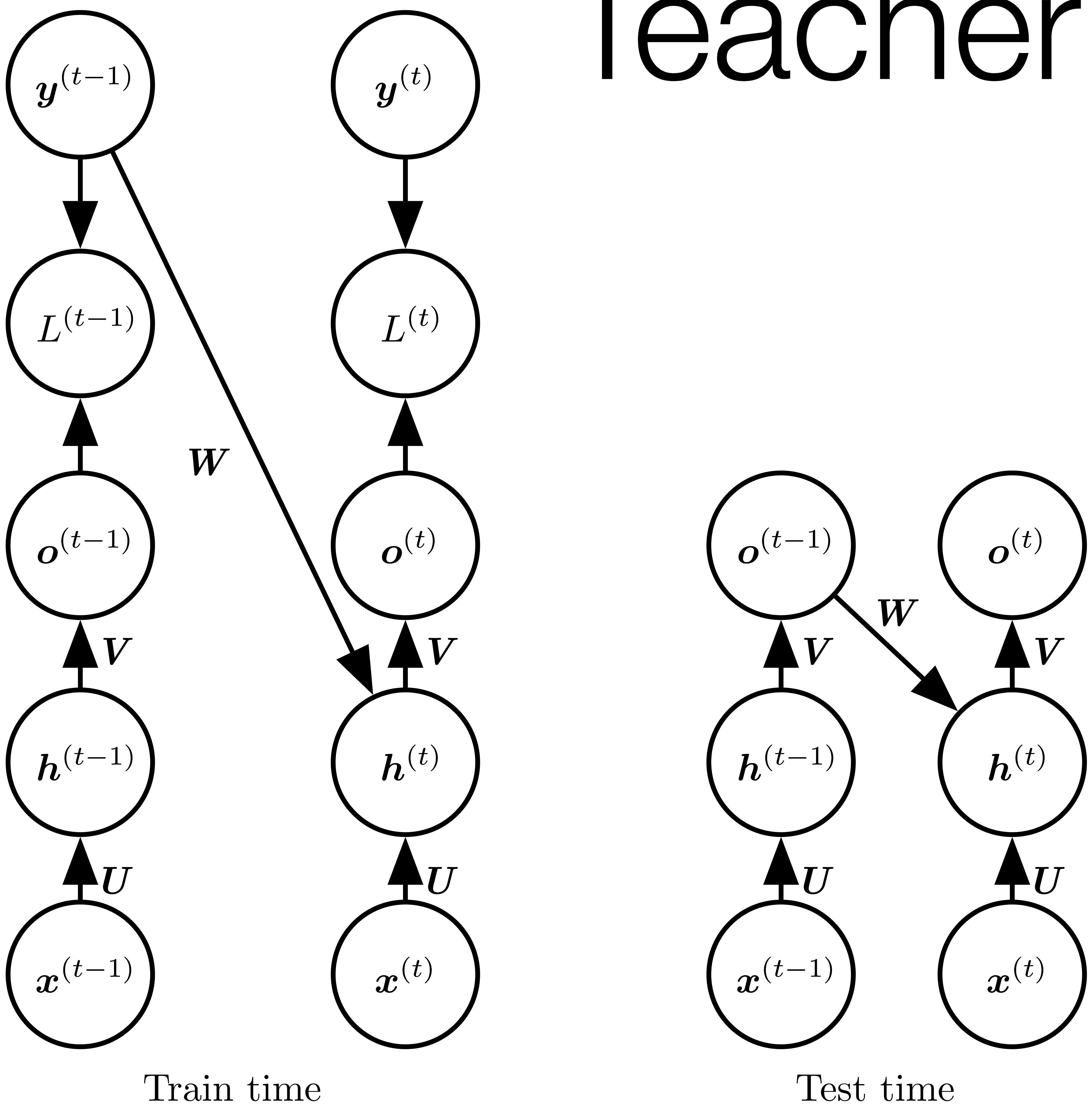
Recurrence through (only) Outputs



- Can have recurrence go from **output** (at $t-1$) to **hidden** (at t) instead of hidden to hidden
- Less general (**why?**)
- **Question:** Why would we want to do this?

(Image: Goodfellow 2016)

Teacher Forcing

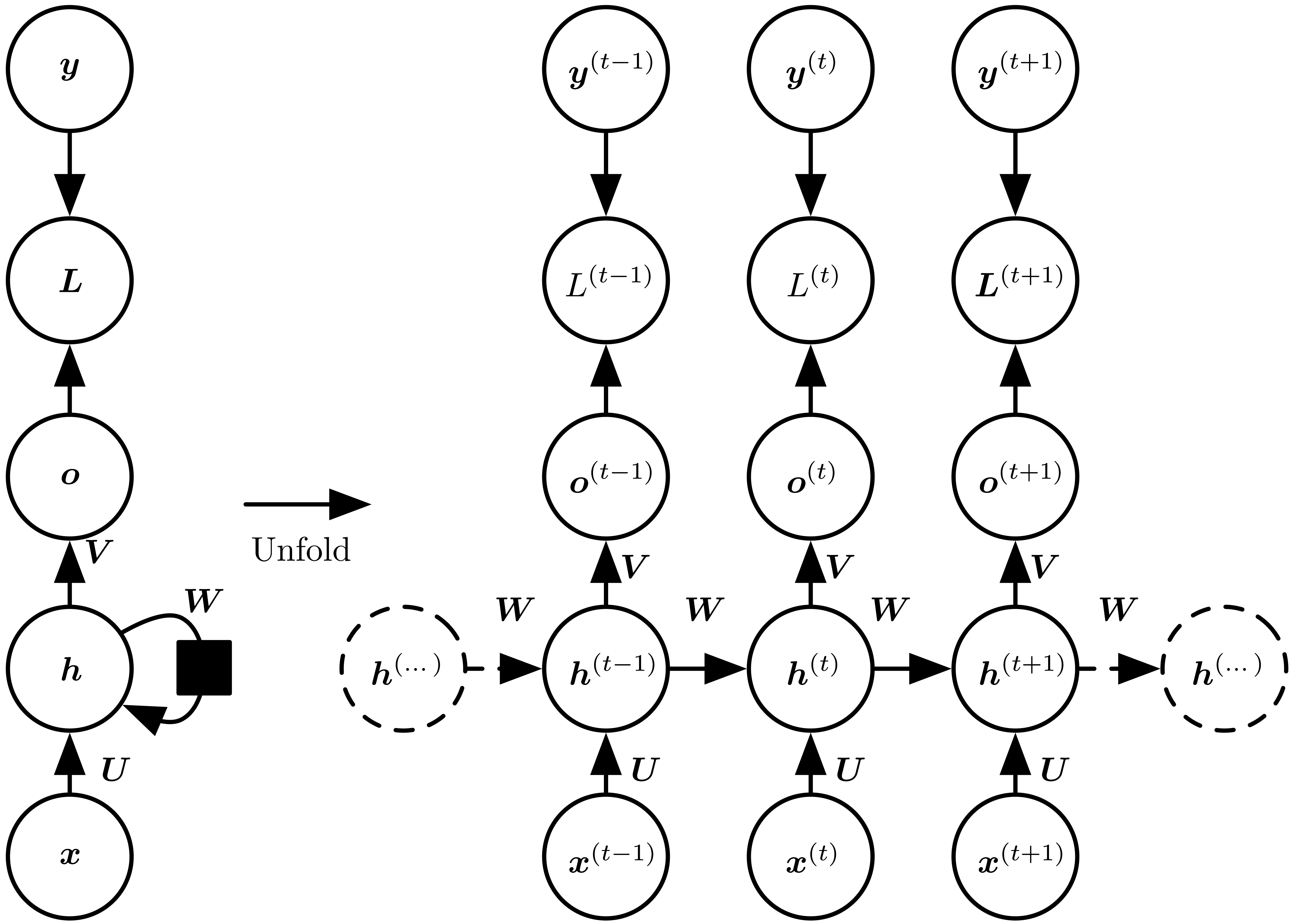


- Dependence on previous step is only on output, not hidden state
  - **Loss gradient** depends only on a **single transition**
  - Training can be **parallelized** (don't need to compute previous states to compute current state)

(Image: Goodfellow 2016)

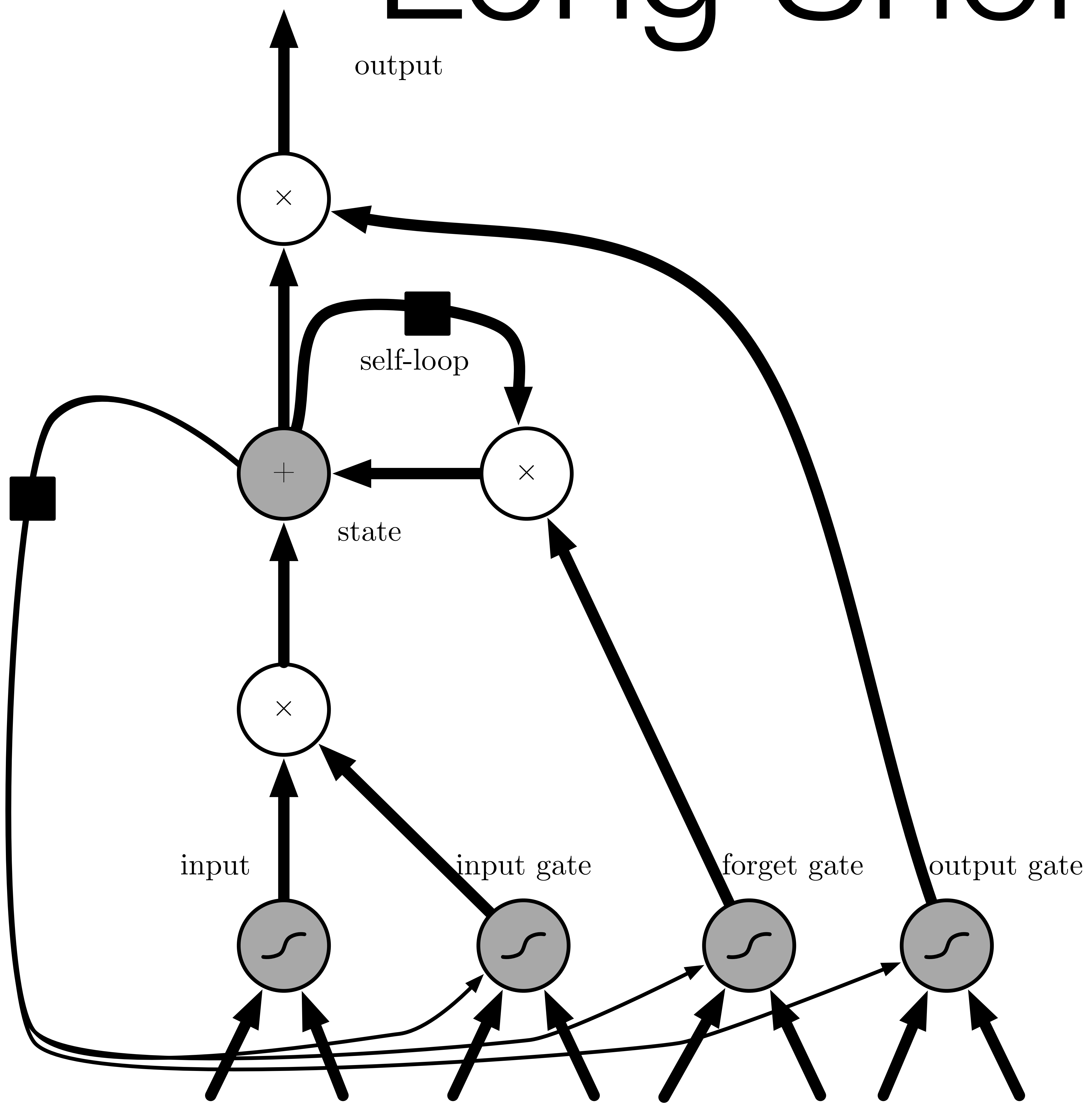
Long-Range Dependence

The submarine, which was the subject of a well known song by the Beatles, was yellow.



- Information sometimes needs to be **accumulated** for a long part of the sequence
- But **how long** an individual piece of information should be accumulated is **context-dependent**
- Often need to **accumulate** information in the state, and then **forget** it later

Long Short-Term Memory



- LSTM networks replace regular hidden units with **cells**
- Input feature computed with regular neuron
- Feature **accumulated** into state only if **input gate** allows it
- State **decays** according to value of **forget gate**
- Output can be **shut off** by the **output gate**

Summary

- Naïvely representing **sequential inputs** for a neural network requires infeasibly many input nodes (and hence **parameters**)
- Recurrent neural networks are a **specialized architecture** for handling sequential inputs
    - **State** accumulates across input elements
    - Each stage computed from **previous stage** using **same parameters**
- **Long short-term memory** (LSTM) cells allow **context-dependent** accumulation and forgetting