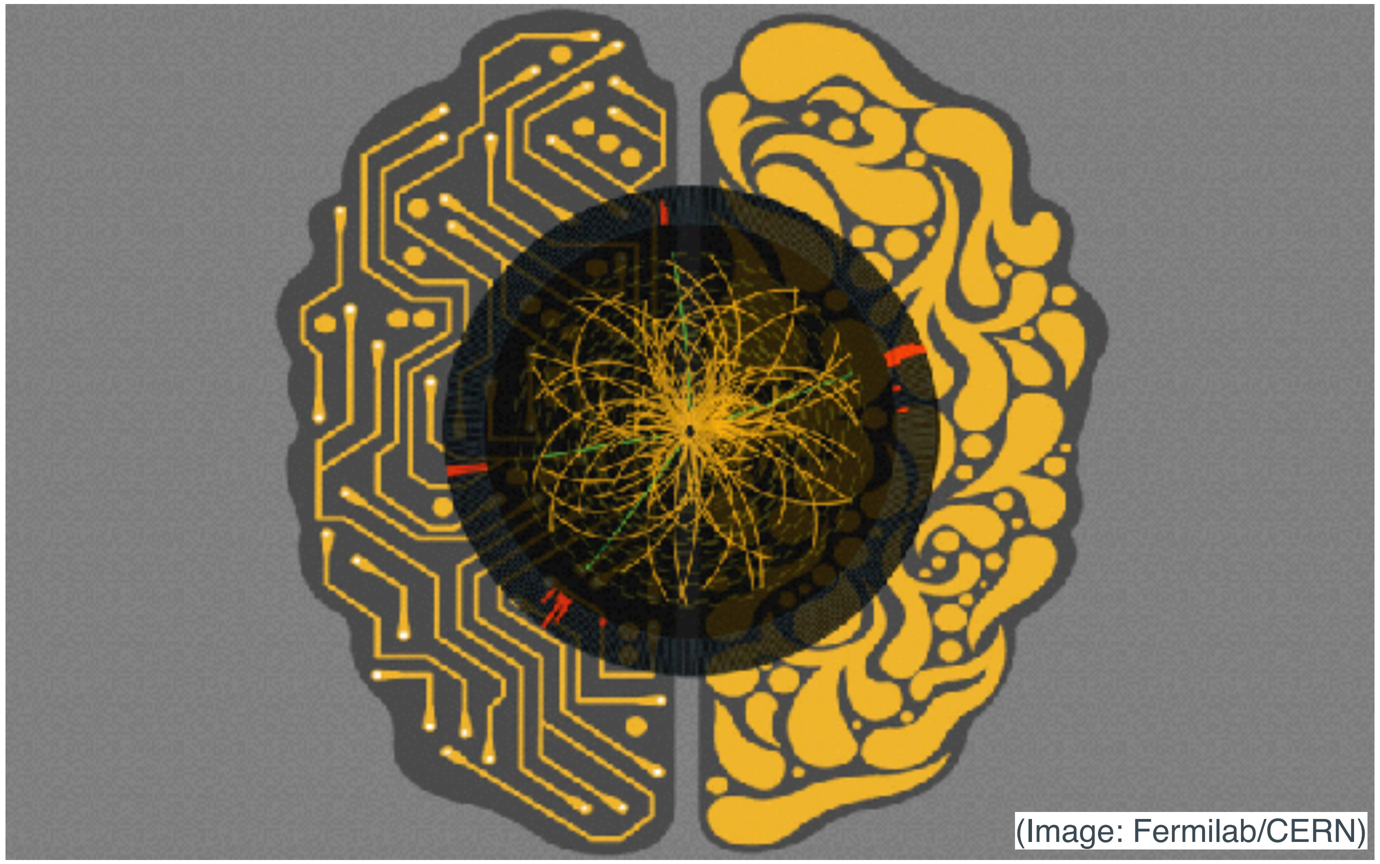
PHY 835: Collider Physics Phenomenology

Machine Learning in Fundamental Physics

Gary Shiu, UW-Madison



Lecture 12: Recurrent Neural Networks

Recap of Lecture 11

- Convolutional neural networks (CNNs)
- Convolutional Layer and Pooling layer
- Workflow for Deep Learning

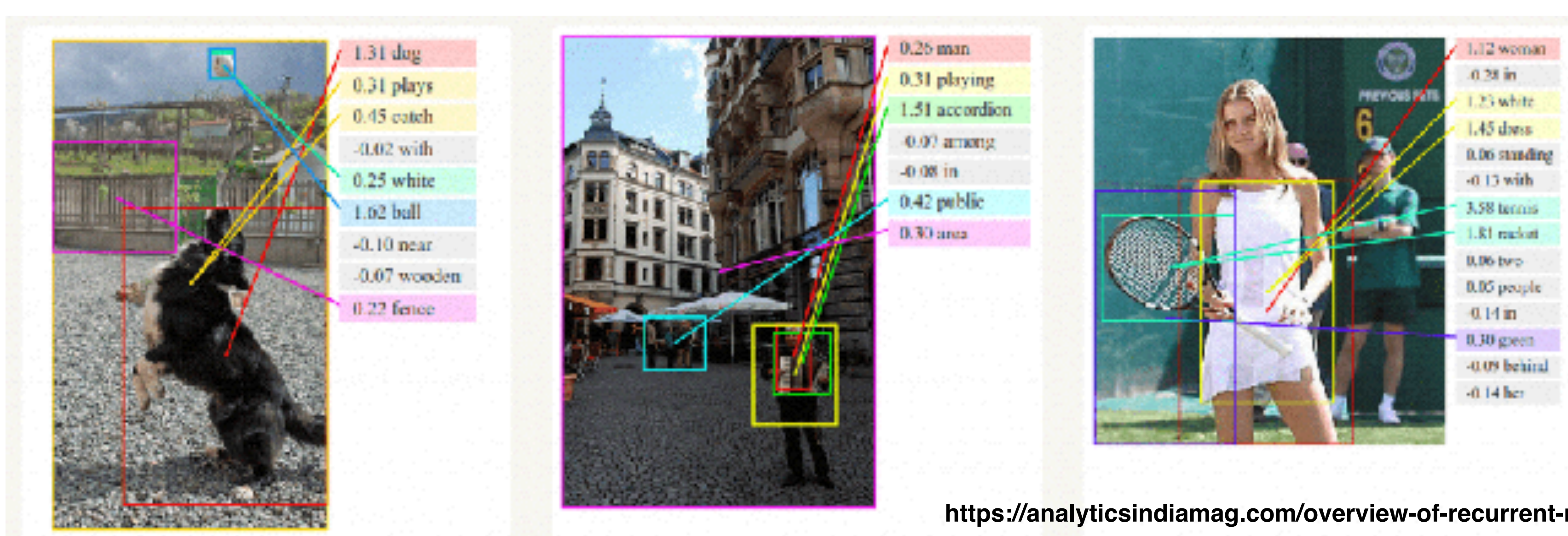
Outline for today

- Recurrent Neural Networks (RNNs)
- Teacher forcing
- Bi-directional RNNs
- Deep RNNs
- Long term dependencies and gated recurrent units

References: Deep Learning Book

Recurrent Neural Networks

- Processing time-sequenced data, though the time step t needs not refer to the passage of time. RNNs can be applied to images.
- Need not respect the notion of causality: the network may have connections that go backward in time.
- Like CNNs, essential to RNNs is parameter sharing — this enables generalization across different sequence lengths/positions in time.
- Example: *“I started my PhD study at UW-Madison in 2020”* and *“In 2020, I started my PhD study at UW-Madison”* have the same info.
- Can be used in combination of ConvNet:



https://analyticsindiamag.com/overview-of-recurrent-neural-networks-and-their-applications/

Basic Idea

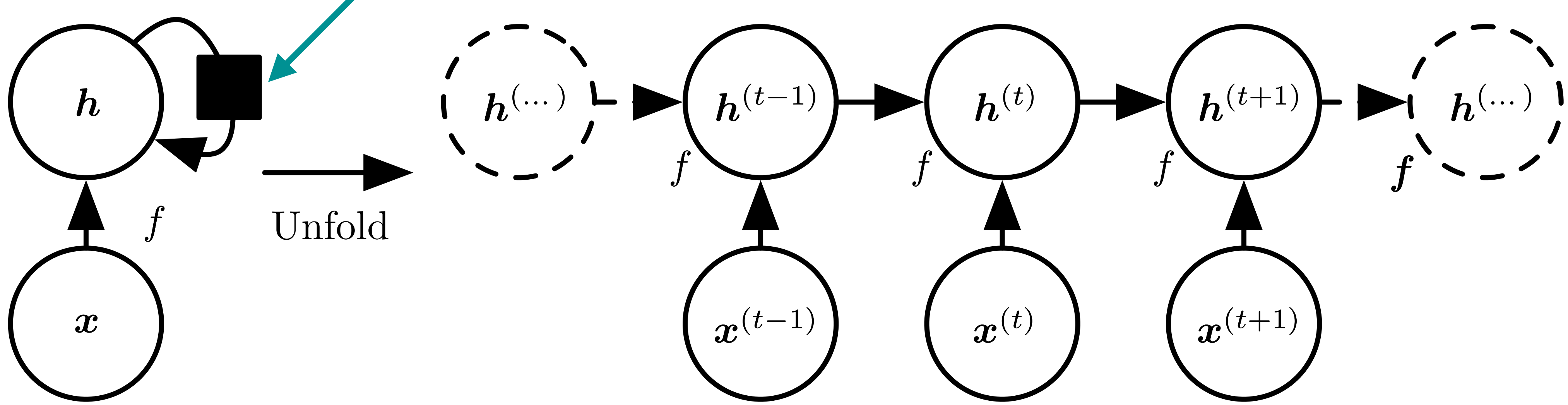
- Include cycles in computational graph which capture the influence of the present value of a variable on its own value at a future step.

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

where $h^{(t)}$ represents the state (hidden), and $x^{(t)}$ external signal.

indicates a delay of a single time step

output computed from h

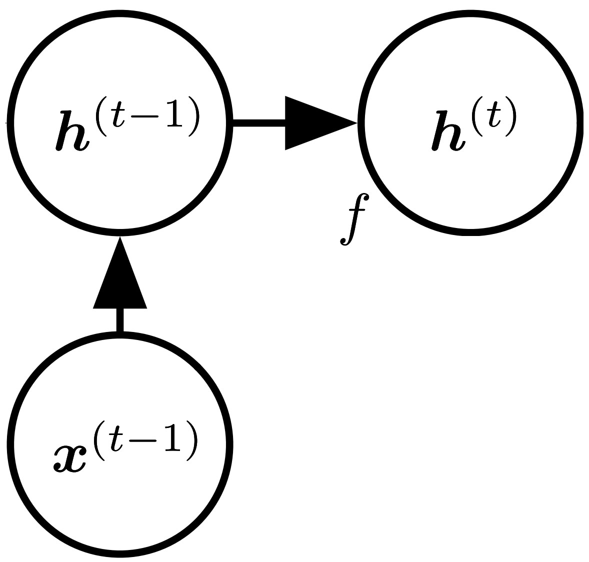


Circuit
(c.f. biological neural network)

Computational Graph
(information flow in time)

Advantages of RNNs

- Regardless of the sequence length, the learned model always has the same input size ⇒ generalization to different sequence lengths.



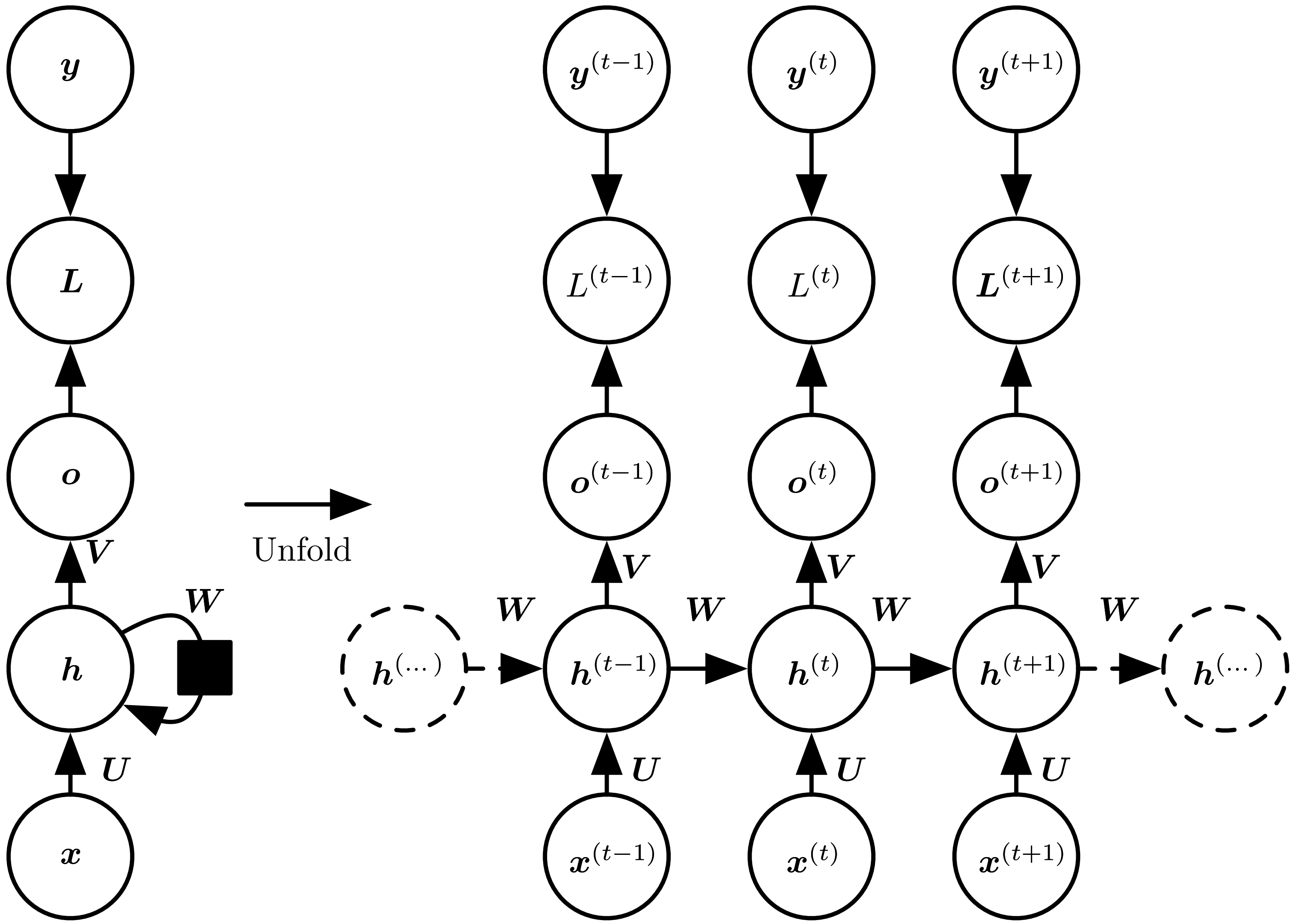
$$\begin{aligned}\boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).\end{aligned}$$

specified in terms of transition from one state to another state, rather than a variable-length history of states

- Parameter sharing: it is possible to use the same transition function function f with the same parameters at every time step ⇒ learn a single model, rather than a separate model for all possible time steps.

Any function involving recurrence can be considered a recurrent neural network.

RNN Types: Output at each time step



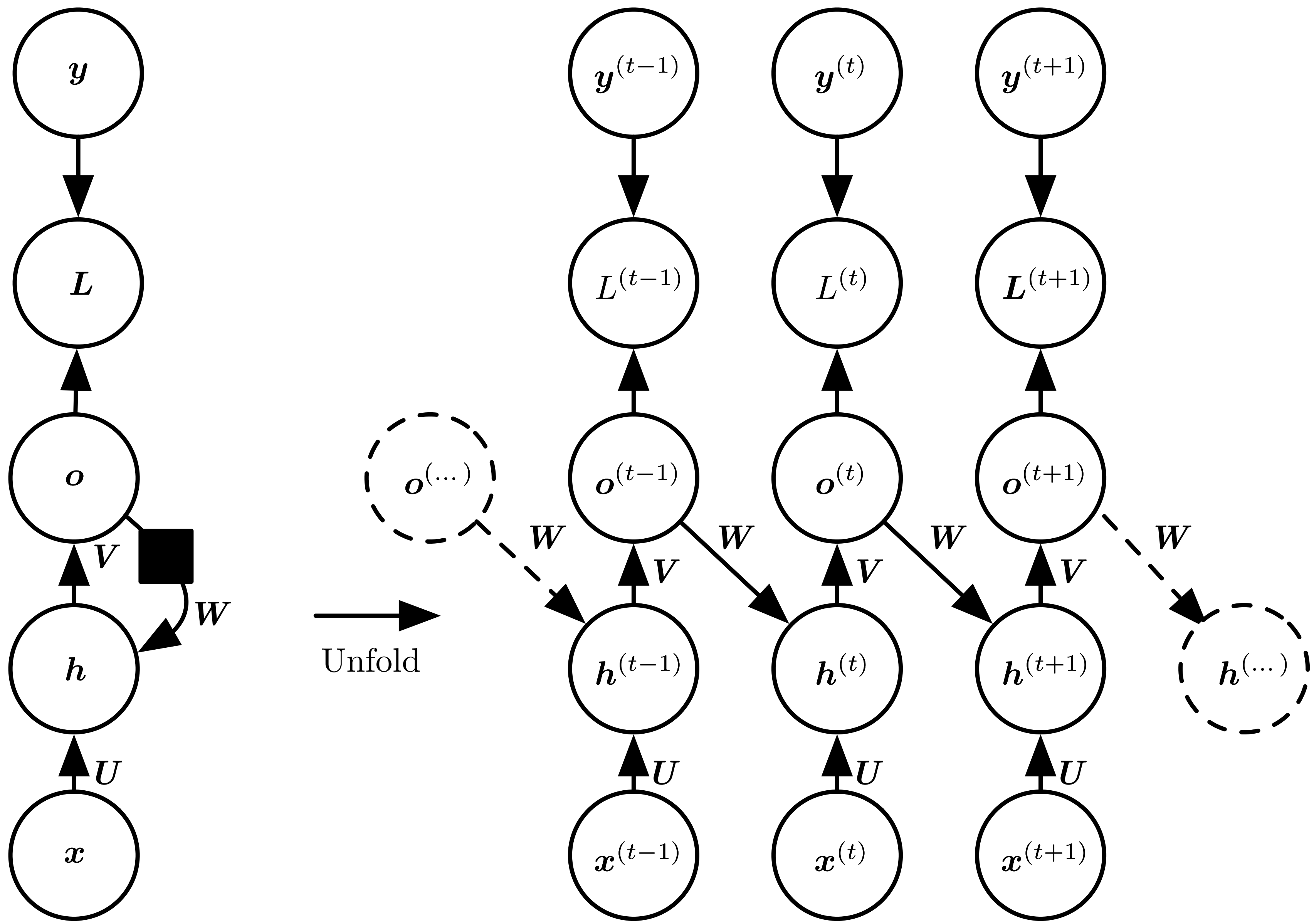
Example:

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

Learnable parameters:
b, c, U, V, W

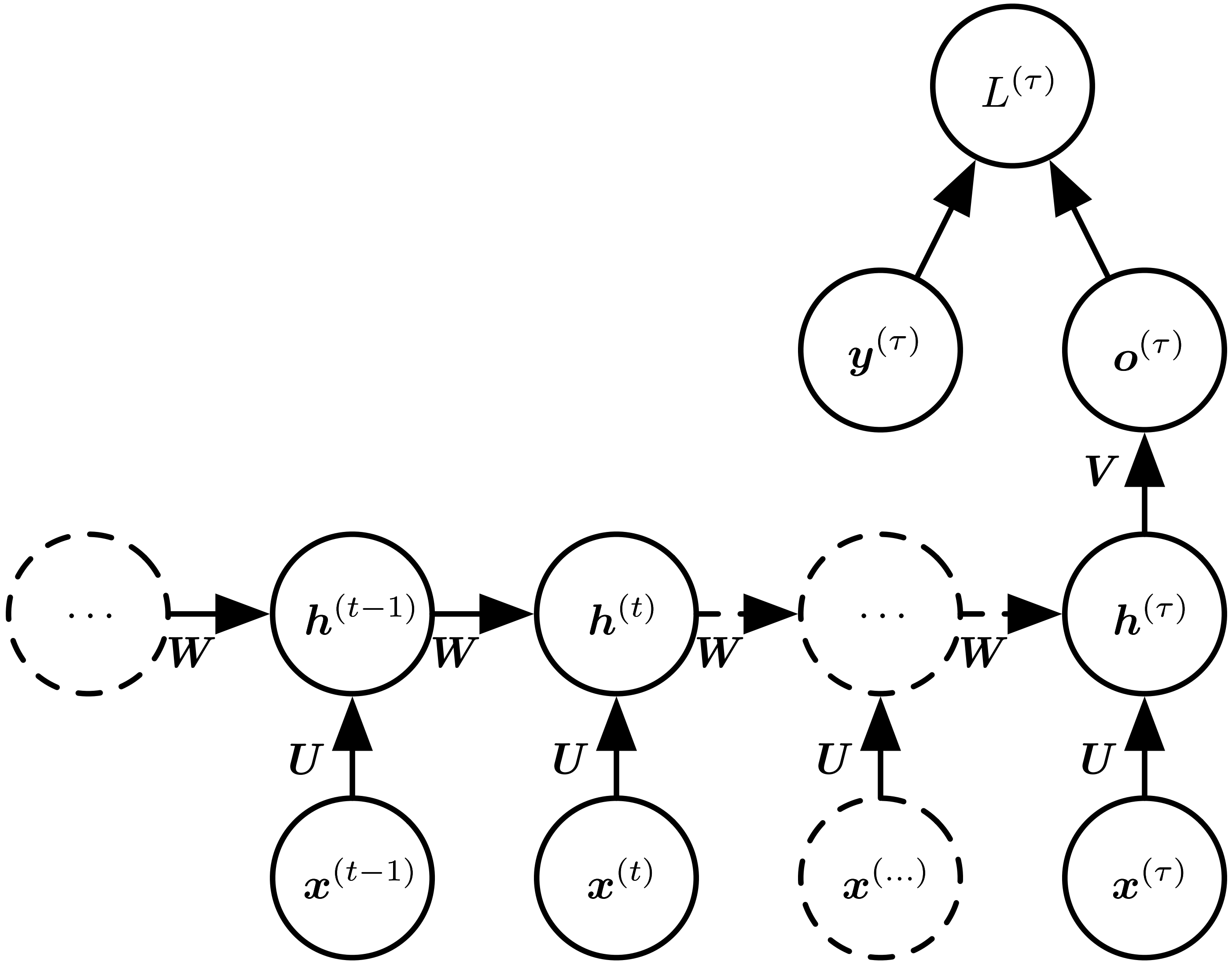
Gradient of loss function can be computed by back propagation through time (BBTT)

RNN Types: Output at t as input at $t+1$



Less expressive, but may be easier to train because each time step can be trained in isolation from the others, allowing greater parallelization during training.

RNN Types: Output after entire sequence



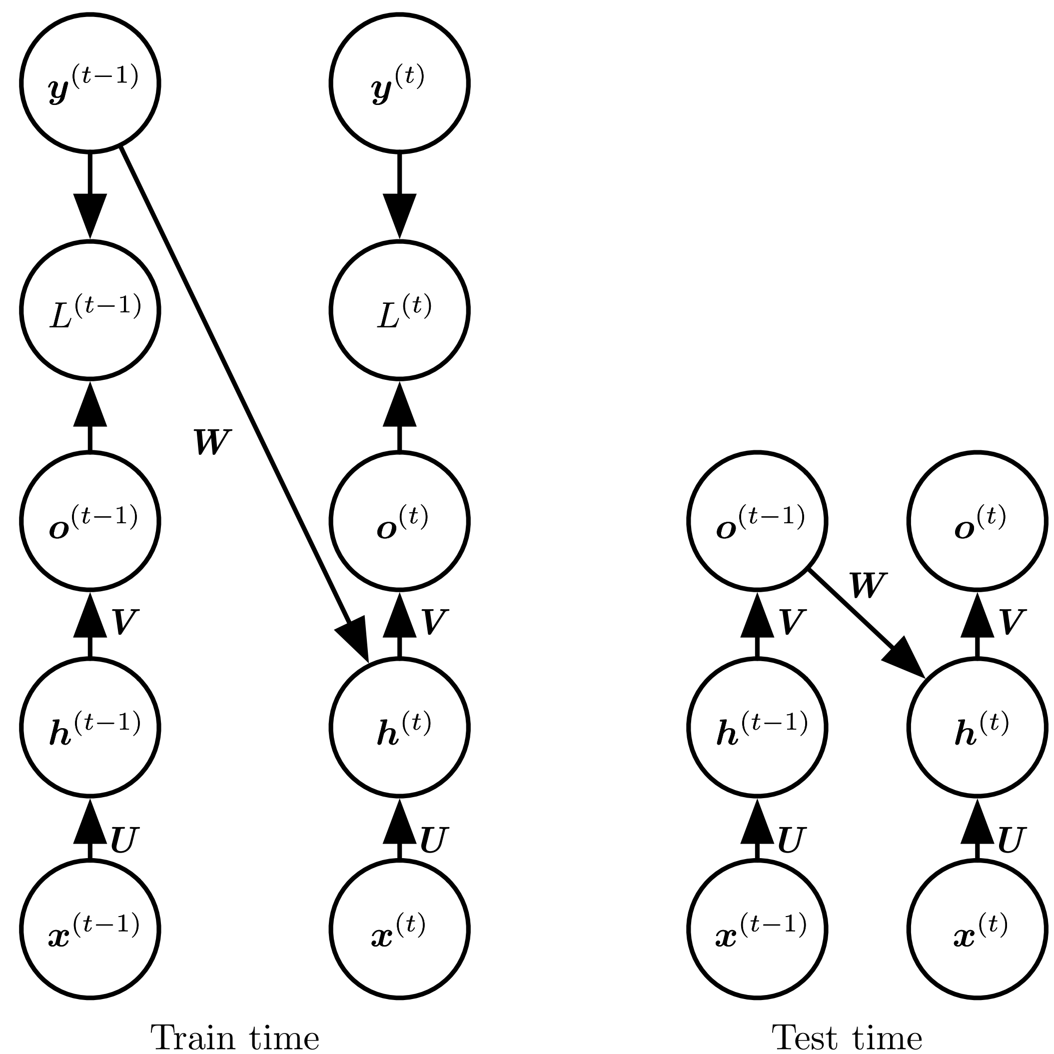
Teacher Forcing

- Problem of **hidden-to-hidden recurrence**: long computation & not parallelizable as the computation is sequential ⇒ Expensive to train

- **Output-to-hidden**: not expressive.

- Use **ground truth output** at t as input at $t + 1$ for the training set.

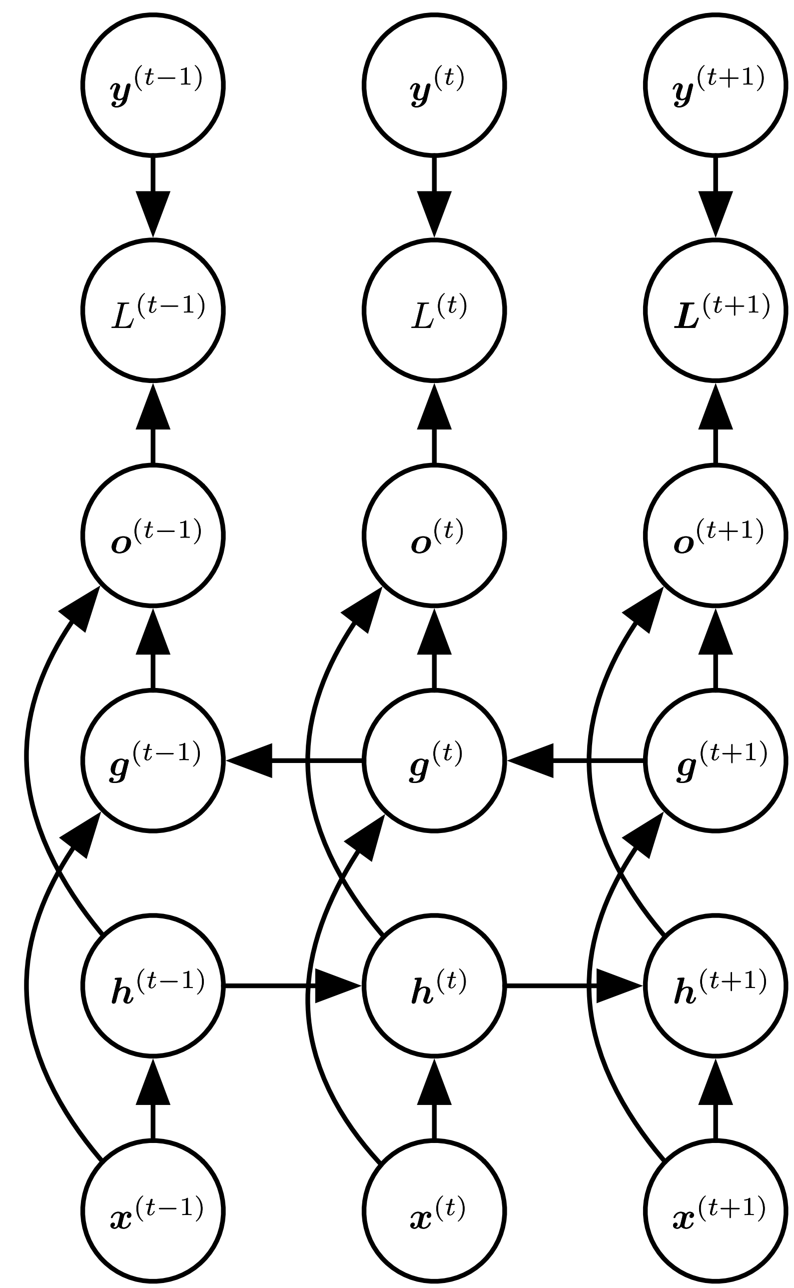
- **Disadvantage:** input that network sees during training might be quite different from that during testing.



Some models can be trained with both teacher forcing and BBTT randomly (Bengio et al, 2015)

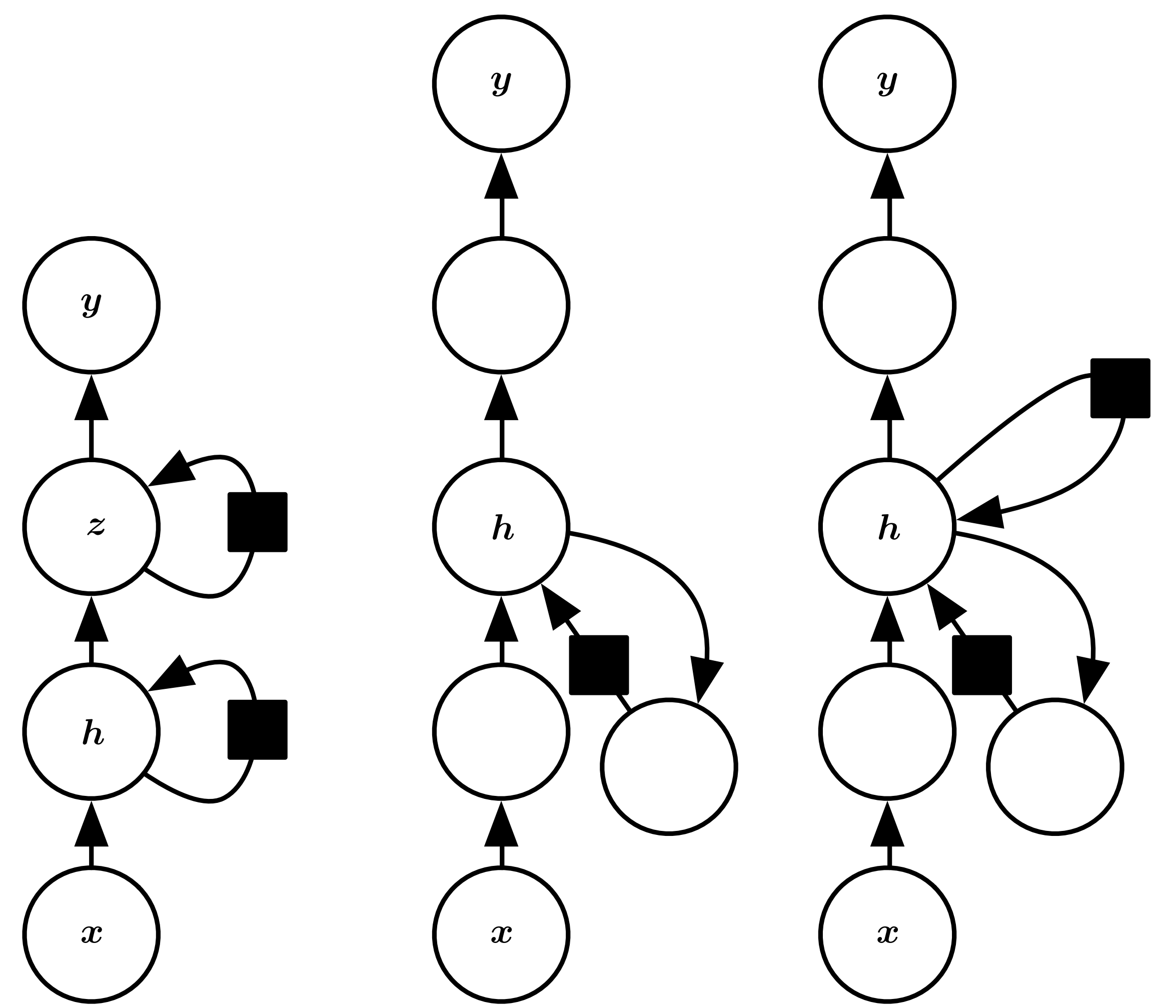
Bidirectional RNNs

- Example: speech recognition. Current sound may depend on the next few sounds (articulation, posing a question)
- Another example: Handwriting recognition.
- **Combination of 2 RNNs:** one moving forward in time, another backward in time.
- Generalization: multiple directions (e.g. 2D grid up, down, left, right).
- Compared with CNNs, RNNs applied to images are typically more expensive but allow for long-range lateral interactions between features in the same feature map.



Deep RNNs

- Make each computation U, V, W deep.
- Replace “linear function” with some deep network.
- Problem: optimization more involved (not a priori clear whether model is trainable).



(a) (b) (c)

Challenge of long-term dependencies

- The recurrence relation can be described as matrix multiplication:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top}\boldsymbol{h}^{(t-1)} = \left(\boldsymbol{W}^{t}\right)^{\top}\boldsymbol{h}^{(0)}$$

- Using the eigendecomposition of W:

$$\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top}$$

- The recurrence is amplified by the number of time steps:

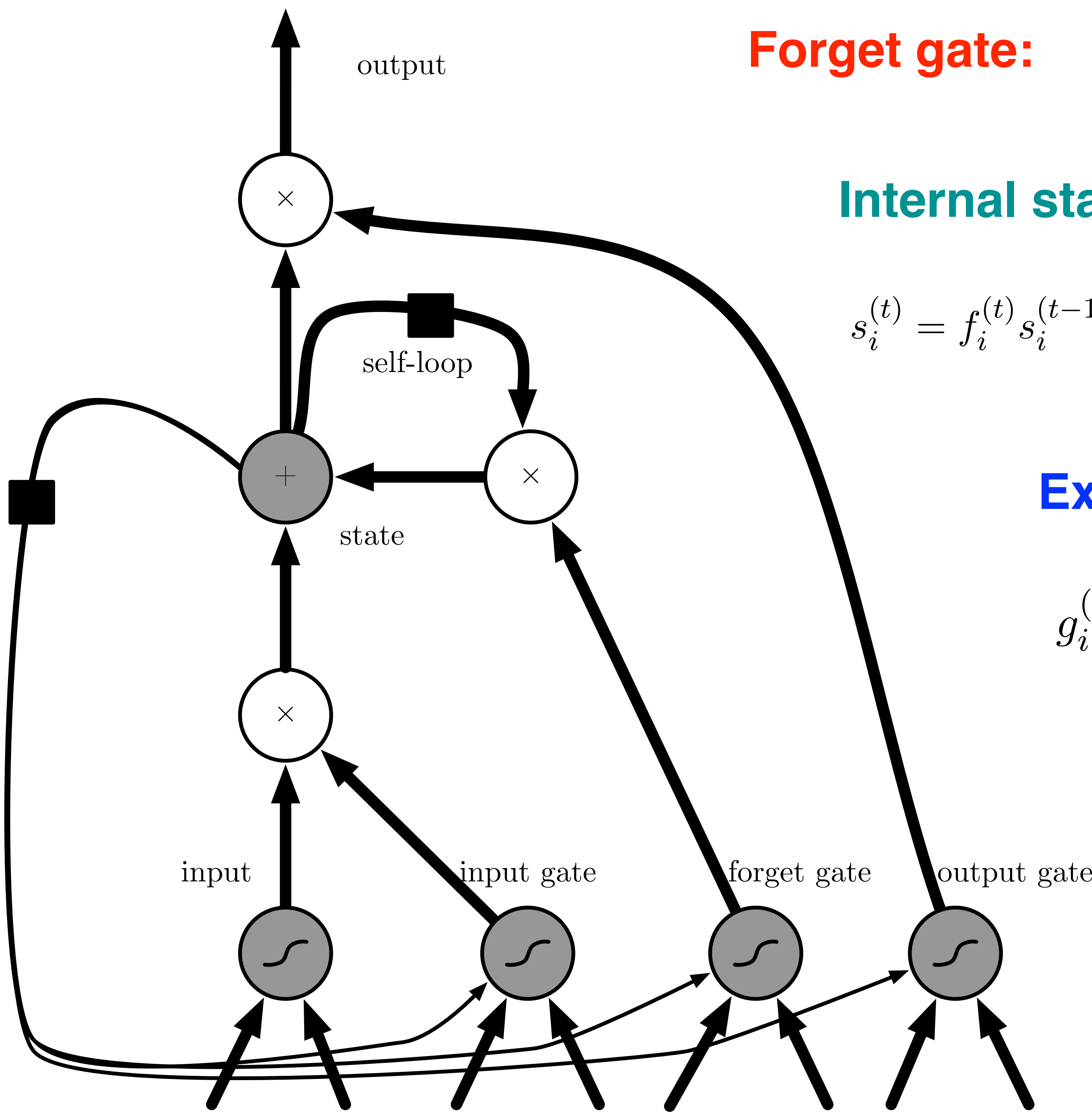
$$\boldsymbol{h}^{(t)} = \boldsymbol{Q}^{\top}\boldsymbol{\Lambda}^{t}\boldsymbol{Q}\boldsymbol{h}^{(0)}$$

- Let λ be the eigenvalues of Λ, $\lambda < 1\ (\lambda > 1) \Rightarrow h^{(t)}$ decays to zero (explodes); $h^{(0)}$ not aligned with the largest eigenvector discarded.
- Network trainable if gradient ≈ 0, gradient of long-term interaction « gradient of a short-term interaction.

Gated RNNs

- How to avoid difference between long- and short-term memory?
- **Idea:** Create paths through time that neither vanish nor explode.
- Change connection weights at each time step.
- **Idea:** Once relevant information of past states has been used, forget the old state. Use trainable parameters to decide when to do it.
- LSTM-layer (long-short-term-memory-layer)

LSTM



Forget gate:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

Internal state update:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

External input gate:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

Output gate:

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)},$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right)$$

Summary

- Recurrent Neural Networks (RNNs)
- Teacher forcing
- Bi-directional RNNs
- Deep RNNs
- Long term dependencies and gated recurrent units