Recurrent Neural Networks

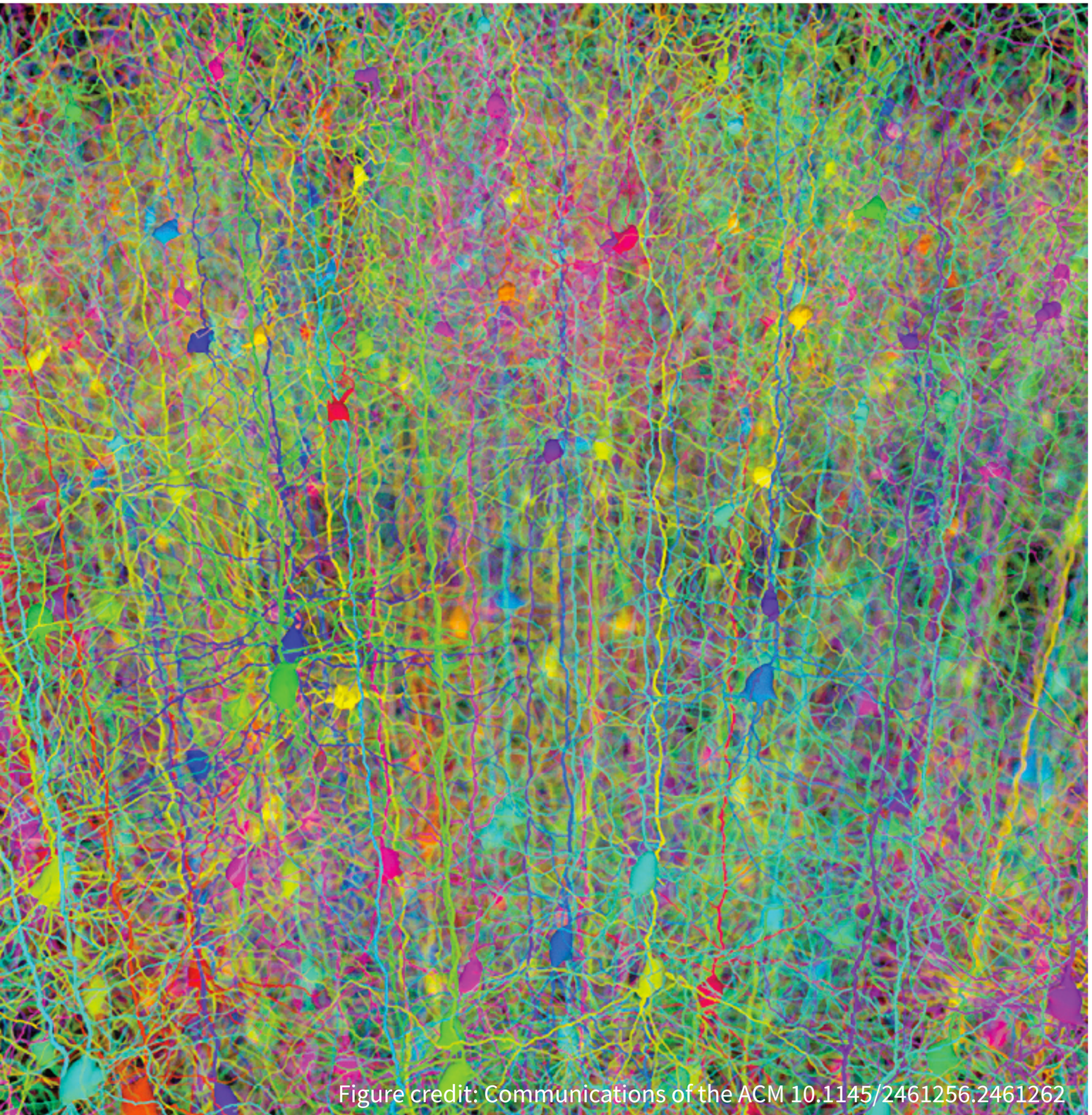


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

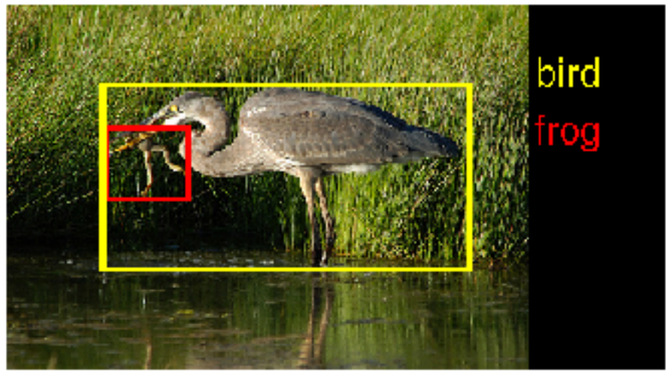
CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

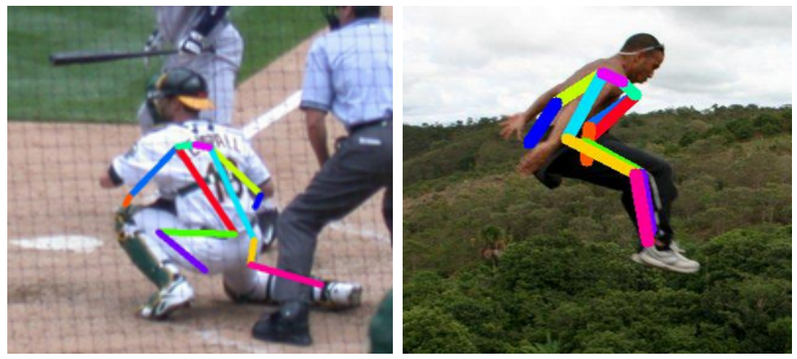
UNIVERSITY of GUELPH
CHANGING LIVES
IMPROVING LIFE

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

Deep Learning Successes

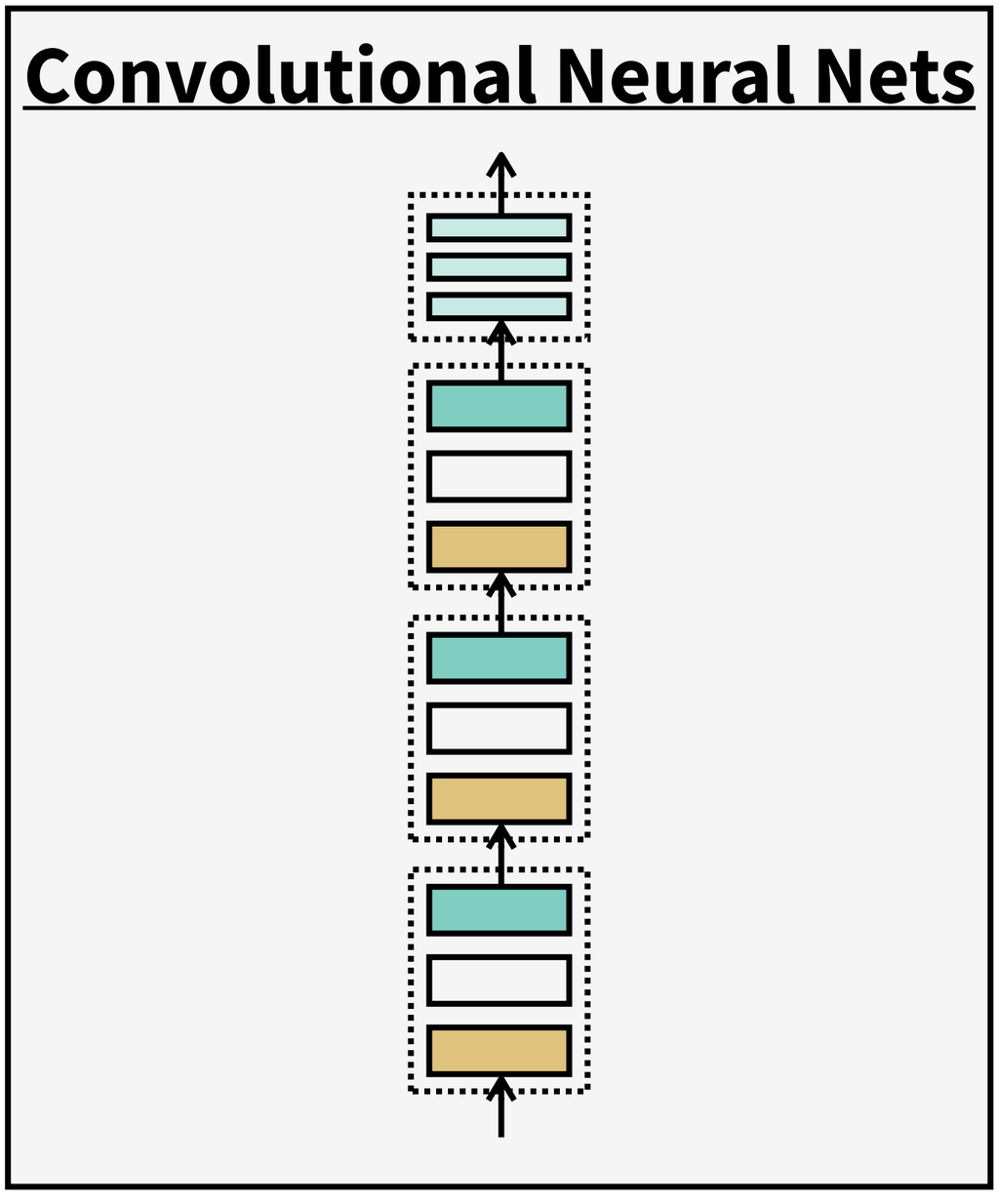
- Vision
  - Object recognition and detection
  - Pose estimation and activity recognition
- Speech
  - Speech recognition
  - Speaker authentication
- Natural Language Processing
  - Question answering
  - Machine translation





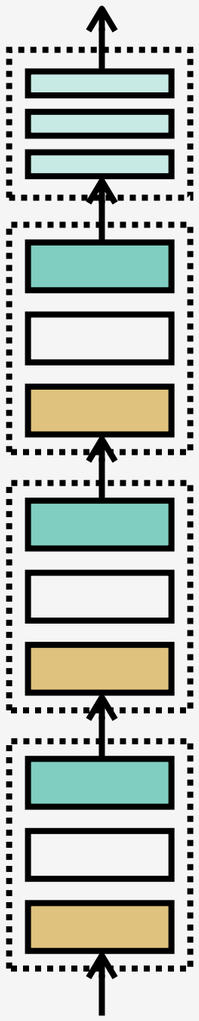
Two “Hammers”

Two “Hammers”

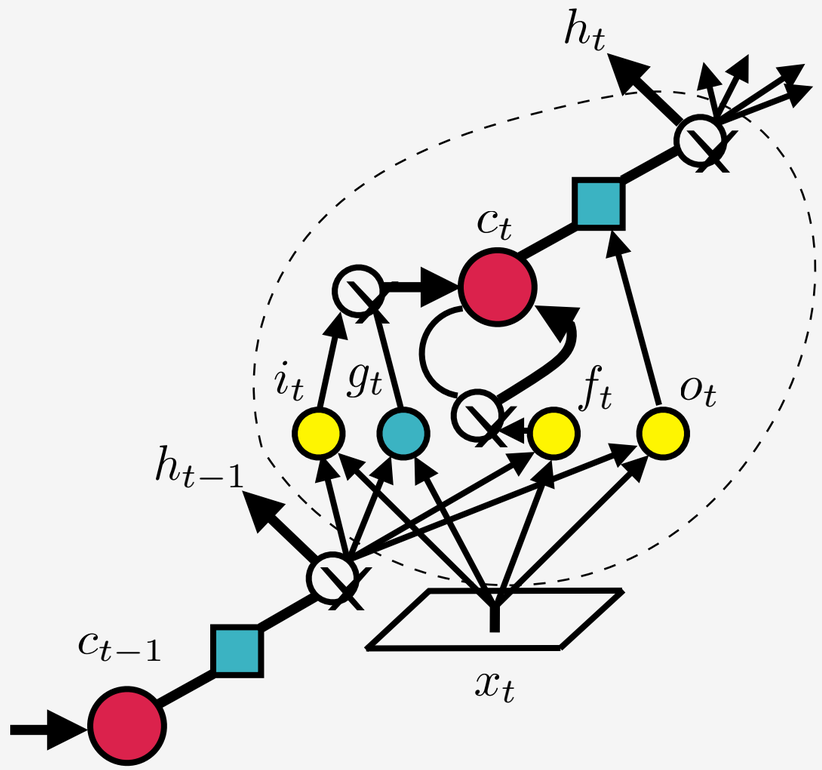


Two “Hammers”

Convolutional Neural Nets



Recurrent Neural Nets

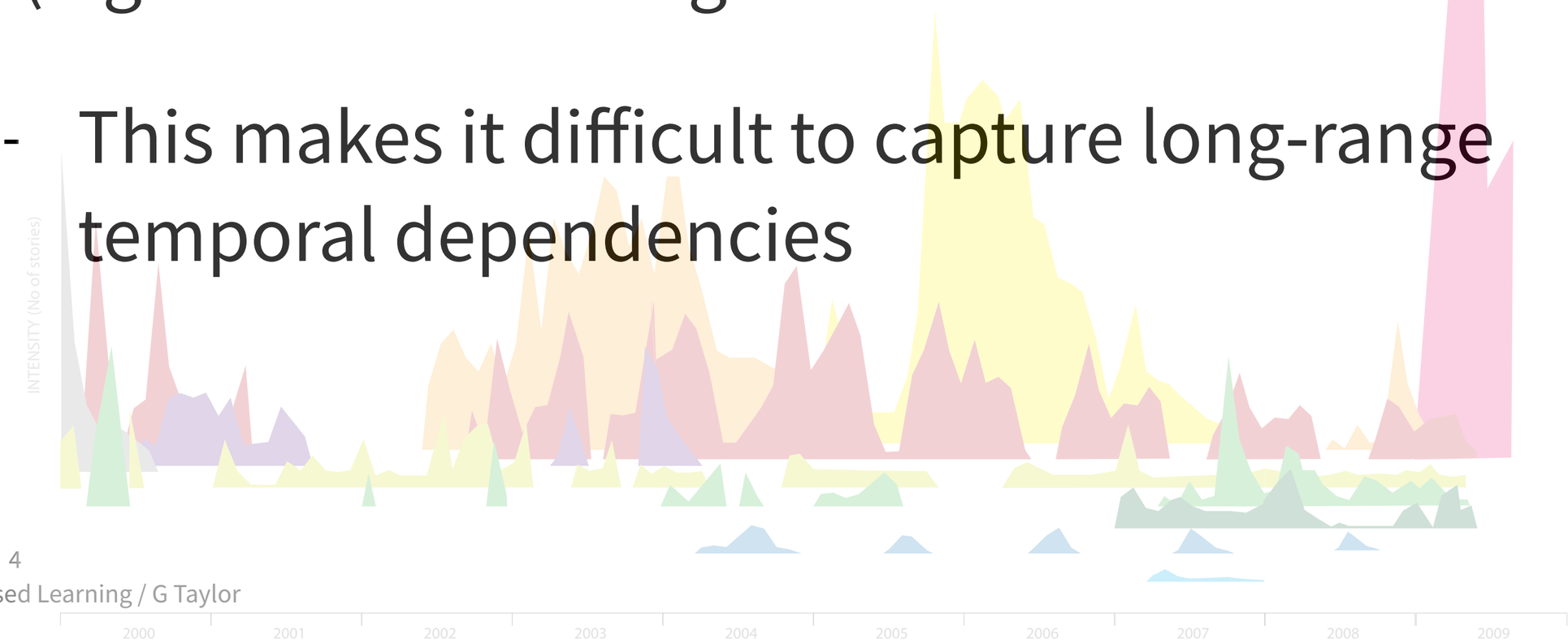


Learning Sequence Representations

- Time is an integral part of many human behaviours (e.g. motion, reasoning, dialogue)

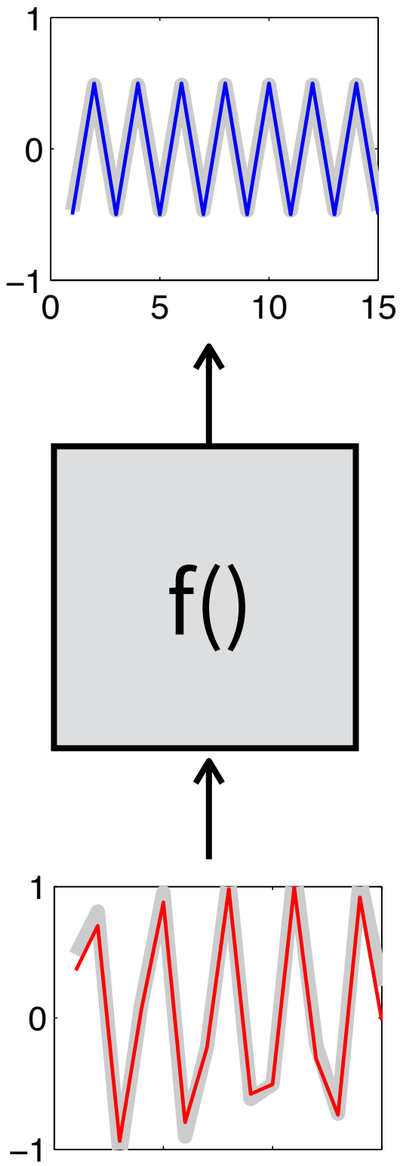
- Easiest way to cope with time is to simply ignore it (e.g. model fixed-length windows of observations)

- This makes it difficult to capture long-range temporal dependencies

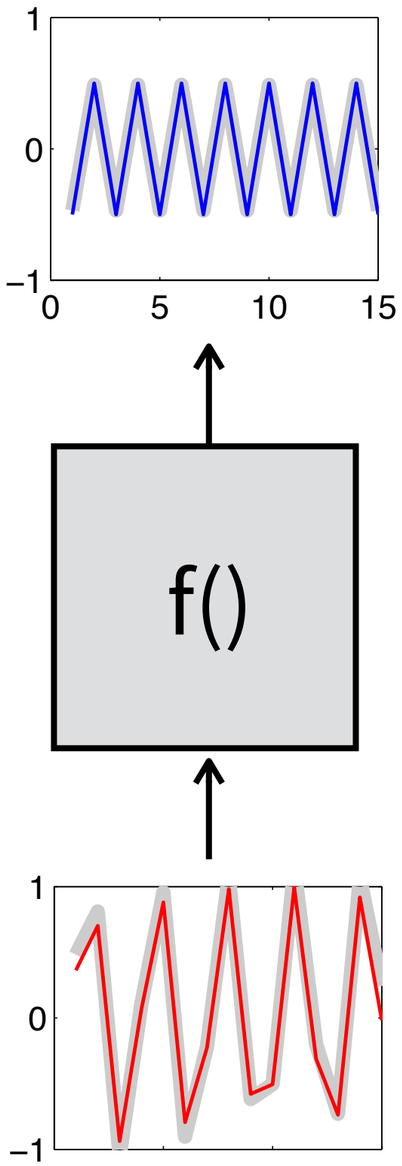


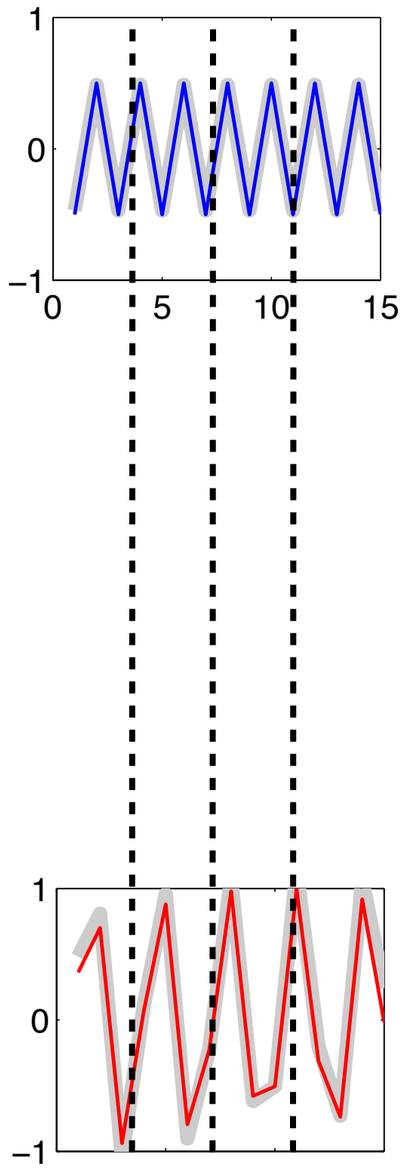
Modeling fixed-length windows?

Modeling fixed-length windows?

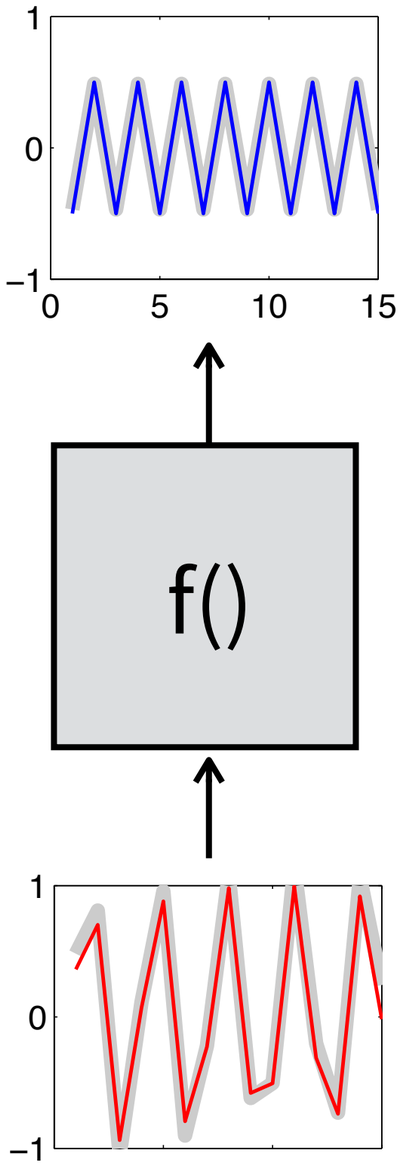


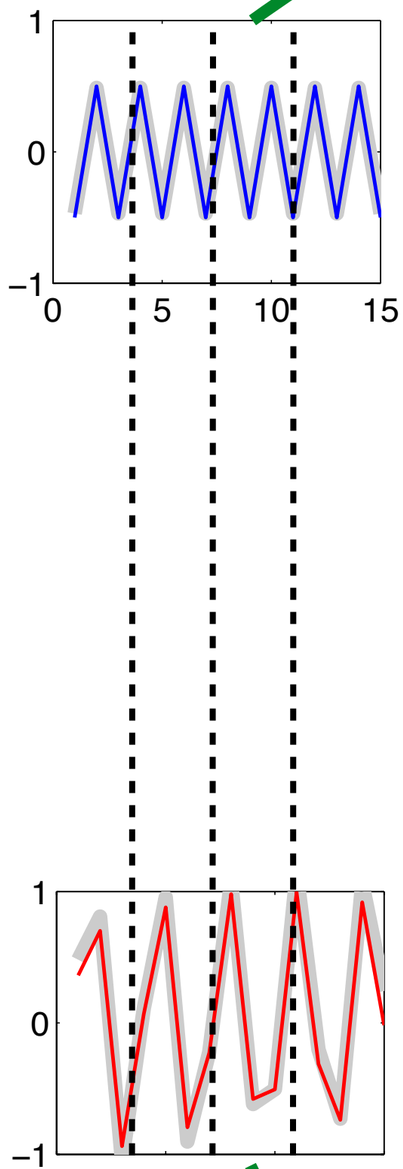
Modeling fixed-length windows?

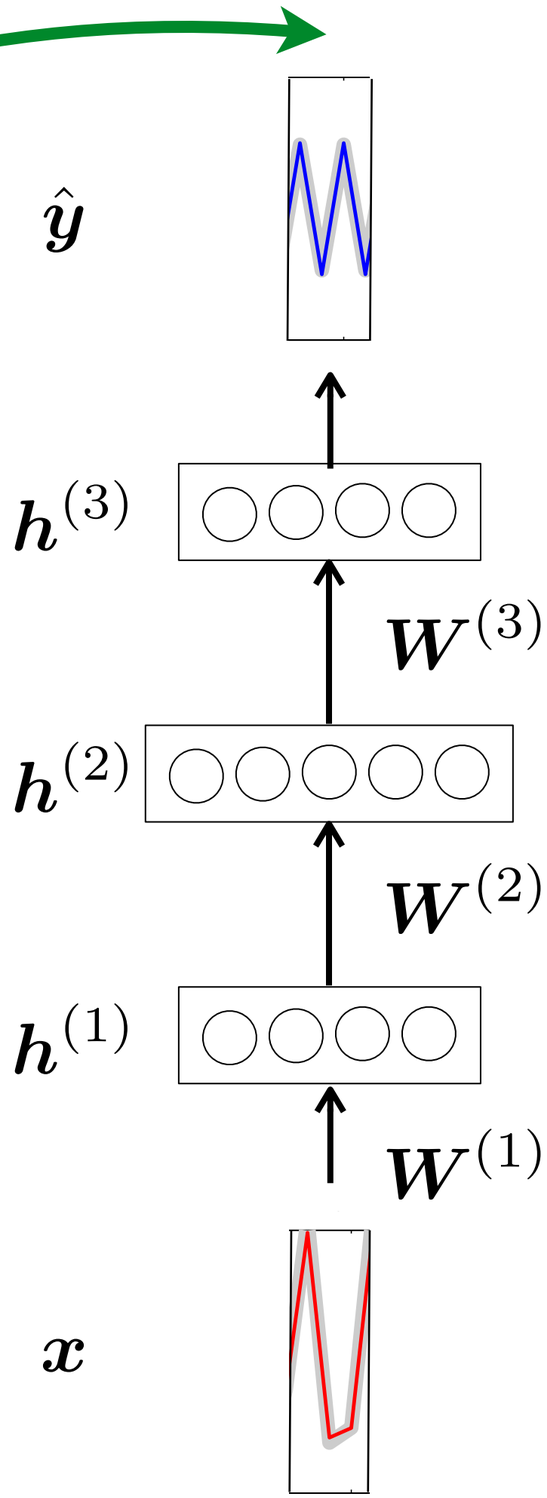




Modeling fixed-length windows?

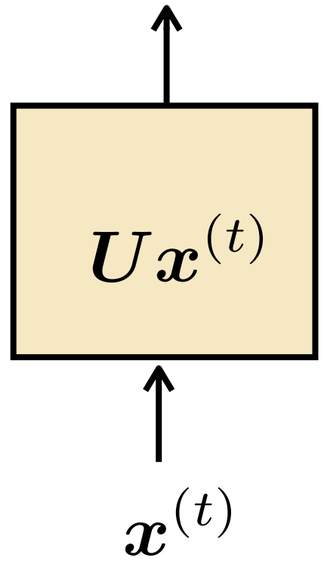




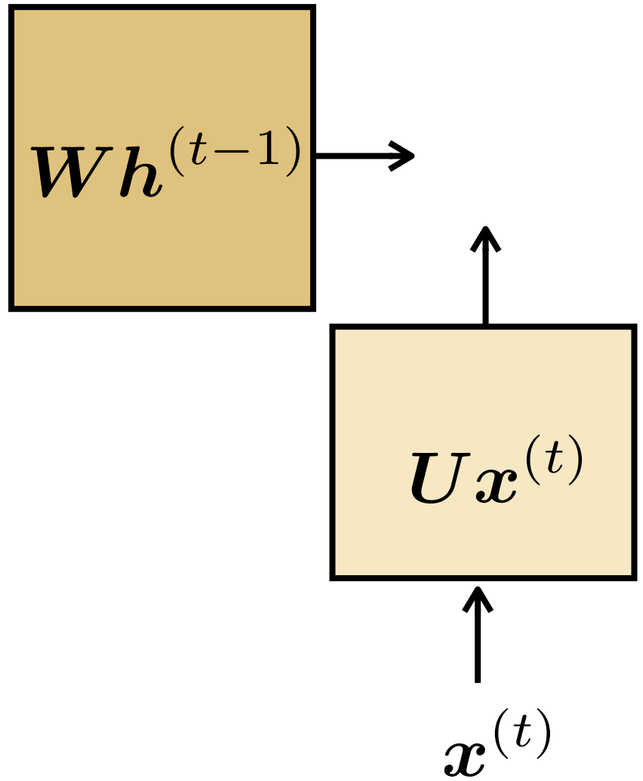


Recurrent Neural Networks

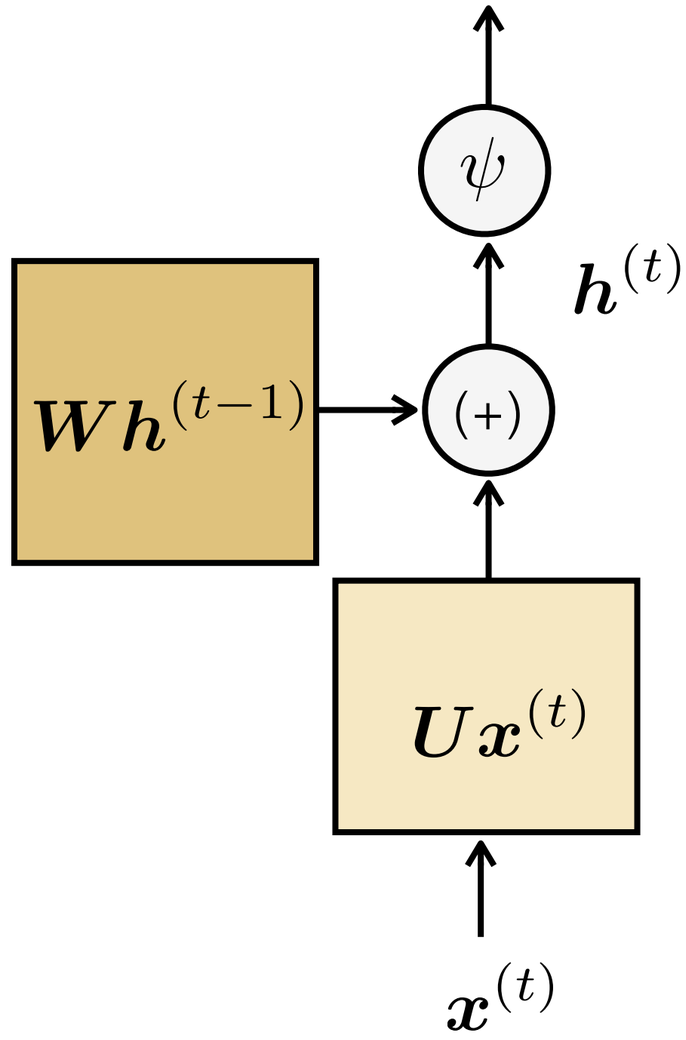
Recurrent Neural Networks



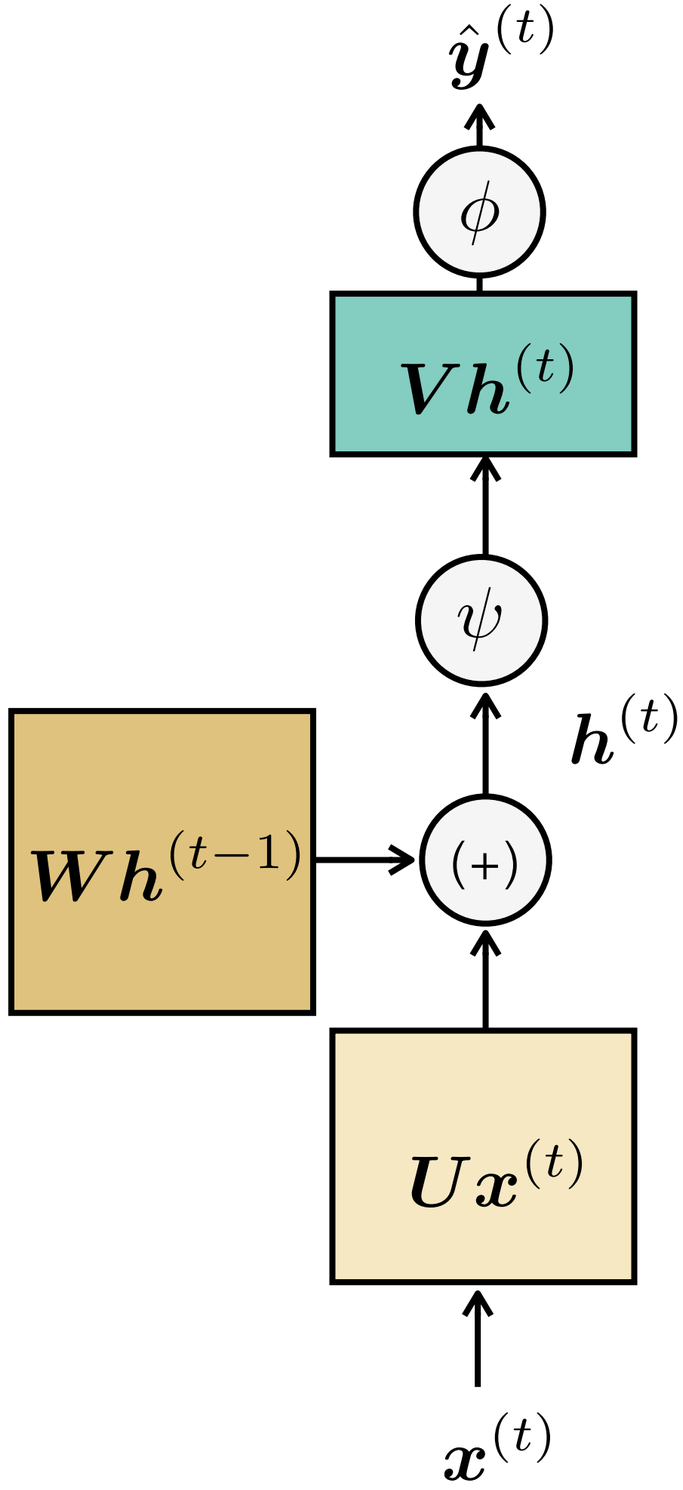
Recurrent Neural Networks



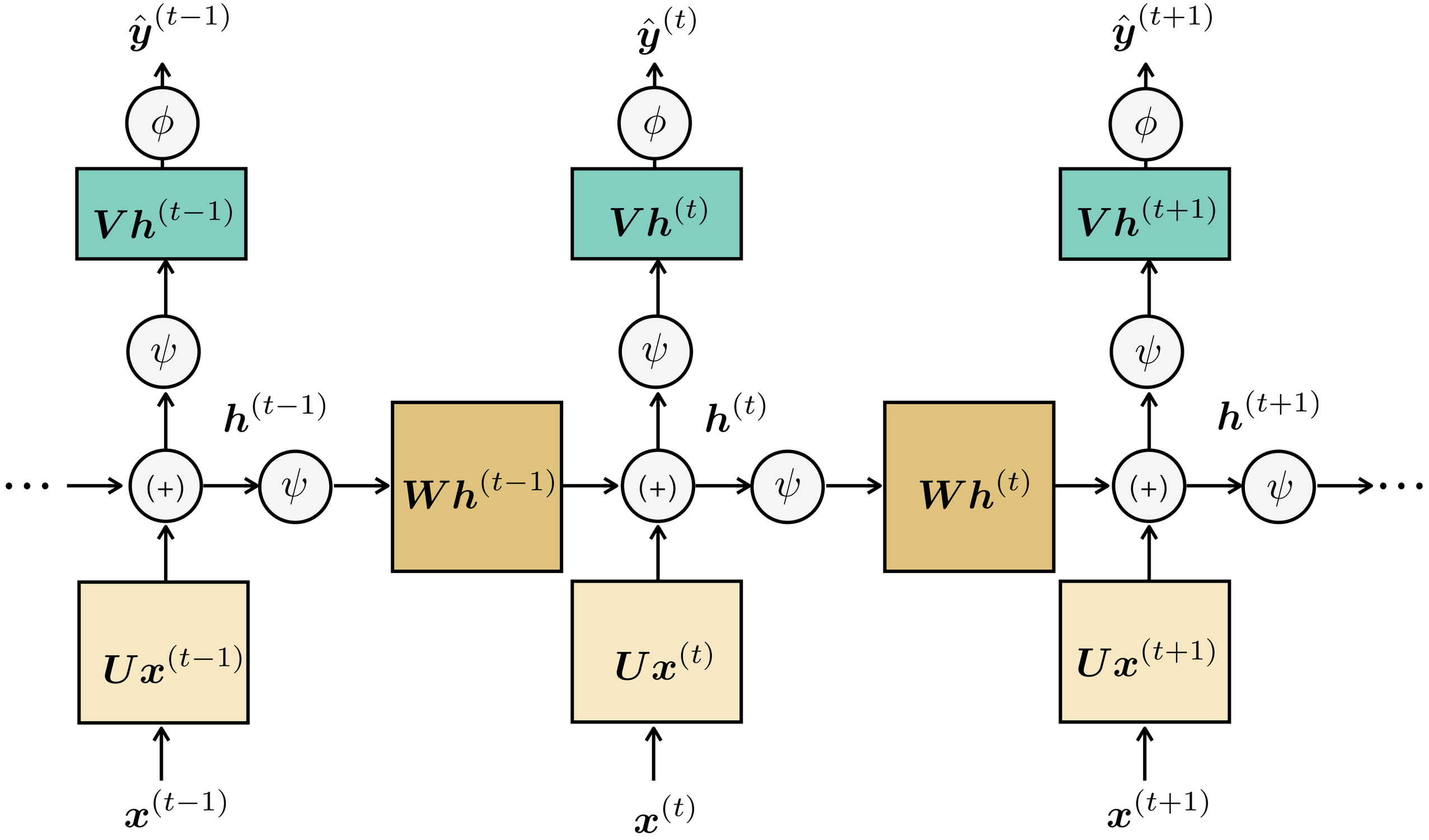
Recurrent Neural Networks



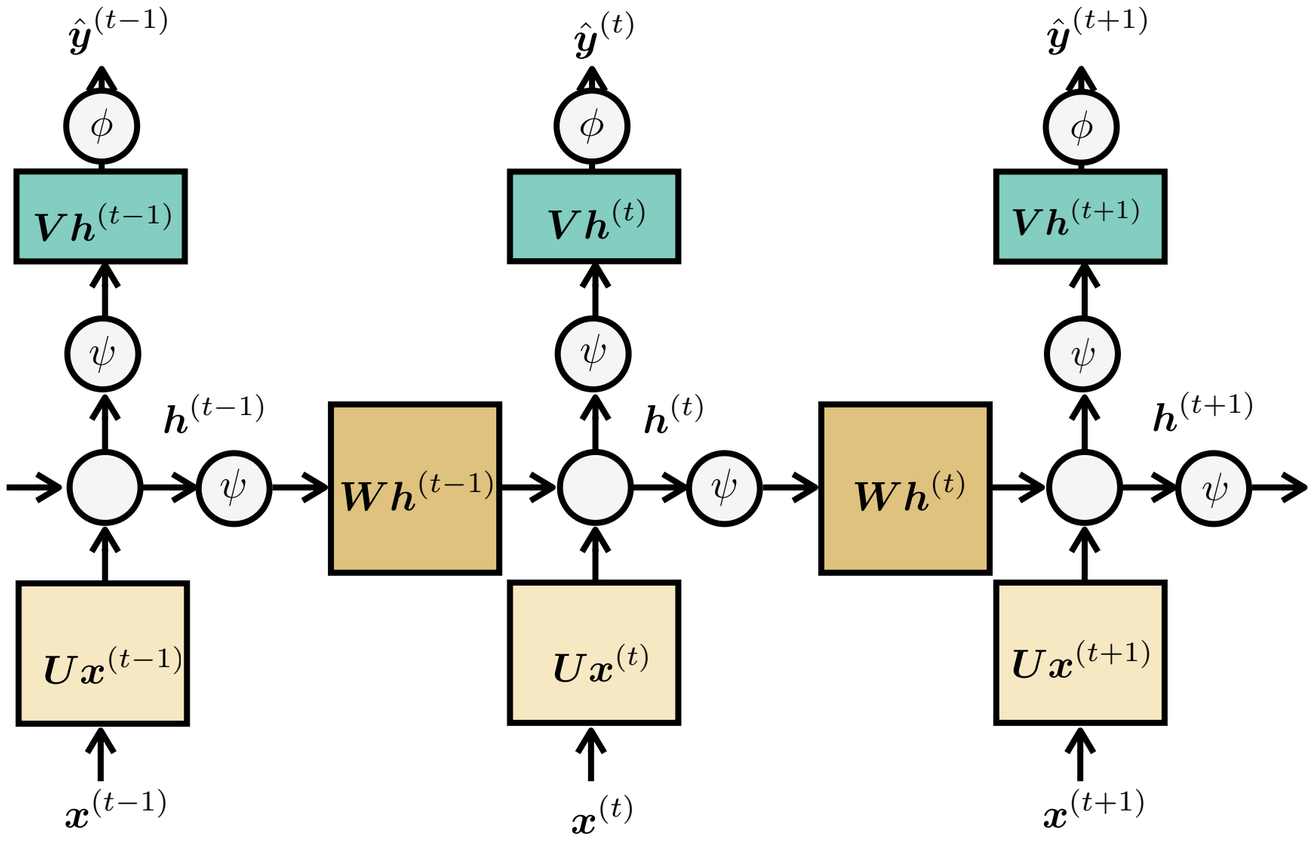
Recurrent Neural Networks



Recurrent Neural Networks



RNN (Update)



$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \psi(\boldsymbol{a}^{(t)})$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$

$$\hat{\boldsymbol{y}}^{(t)} = \phi(\boldsymbol{o}^{(t)})$$

RNNs: The Promise

- Possibly high-dimensional, distributed, internal representation and nonlinear dynamics allow model, in theory, model complex time series
  - and capture long-range dependencies!
- Exact gradients can be computed exactly via Backpropagation Through Time

Generality

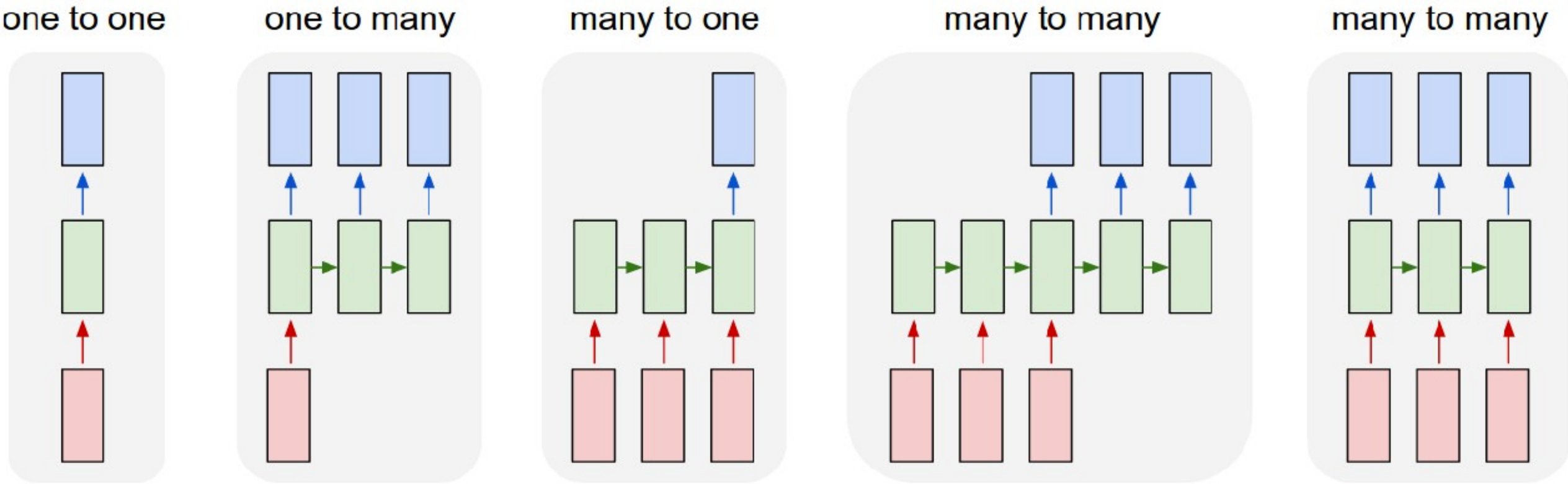
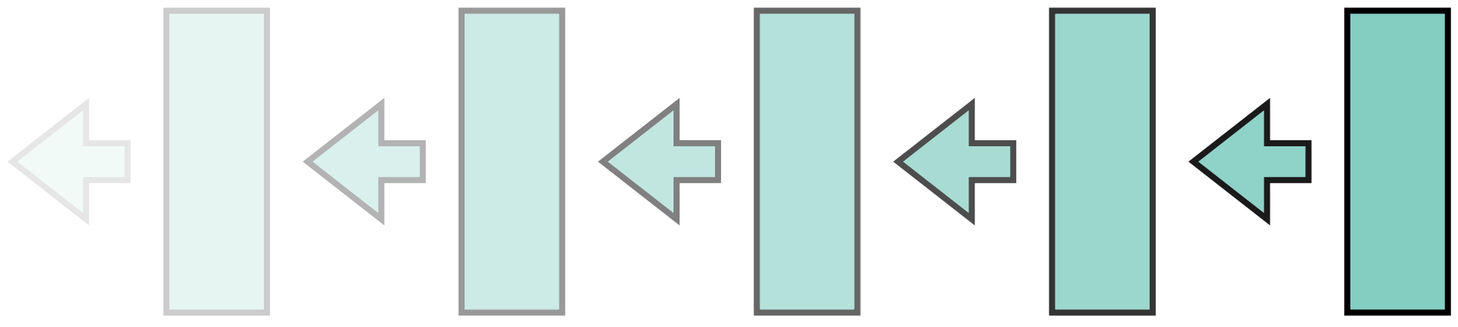


Figure: Andrej Karpathy
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs: The Reality

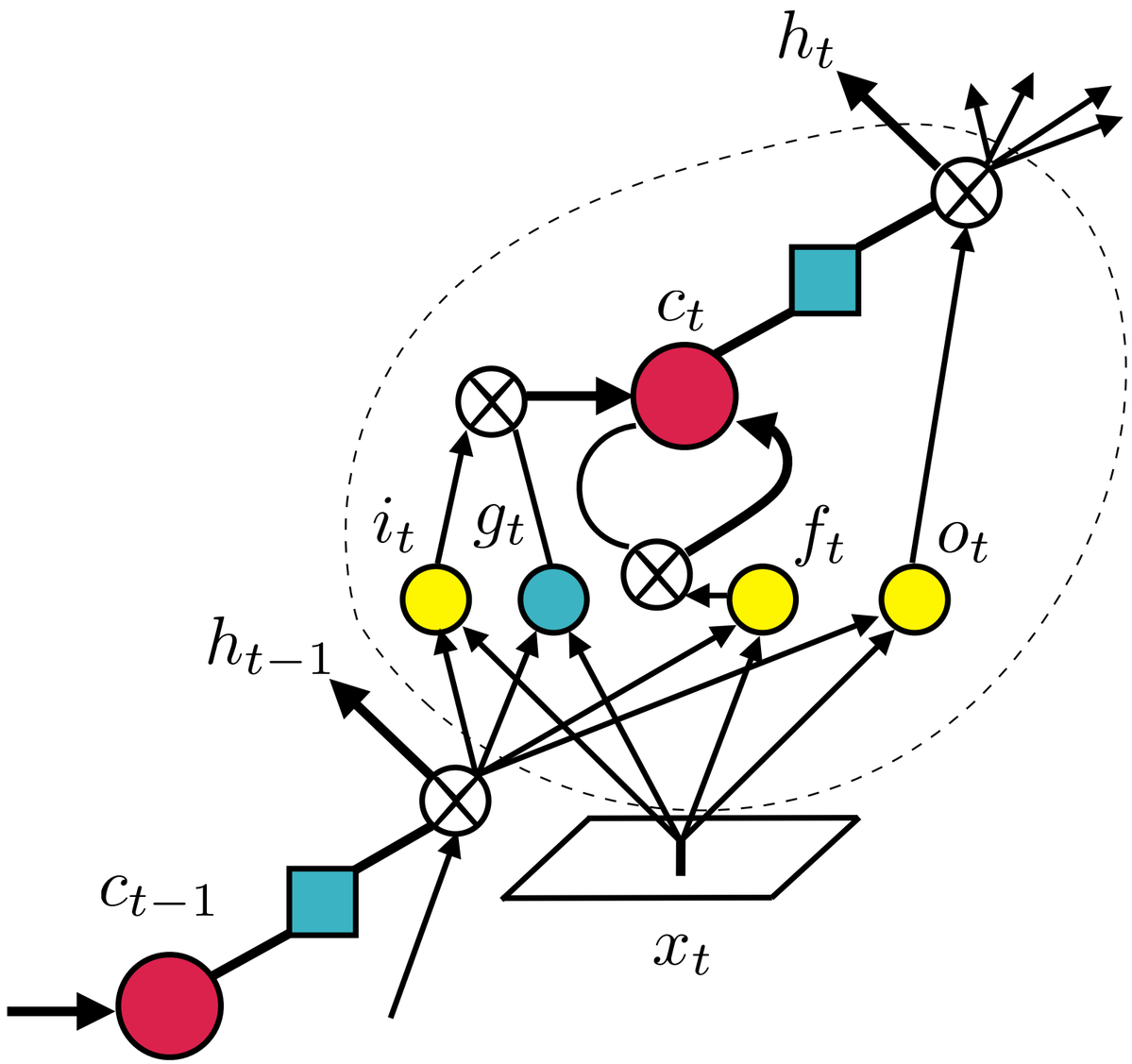
- A powerful and flexible architecture - but?
  - Training RNNs via gradient descent fails on simple problems (typically memorization tasks)
  - Attributed to “vanishing” or “exploding” gradients

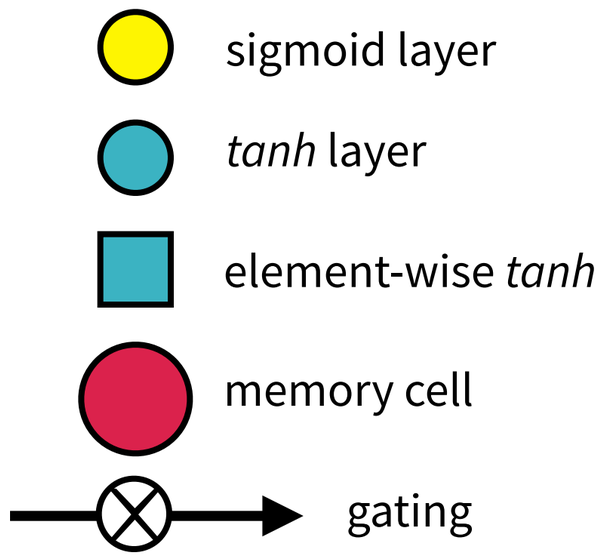


Solutions: Architectural

- Gating 
  - Long Short-term Memory (Hochreiter et al. 1997)
  - Gated Recurrent Units (Chung et al. 2014)
- “Structured” depth (Pascanu et al. 2014)
- Rectifiers (Le, Jaitly and Hinton 2015) 
- Units operating at different timescales (Hihi and Bengio 1996, Koutnik et al. 2014, Chung et al. 2017)

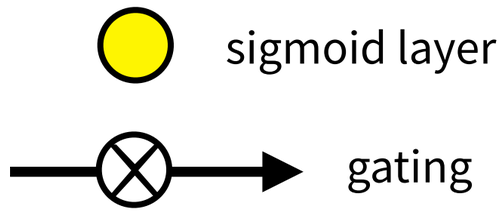
LSTM

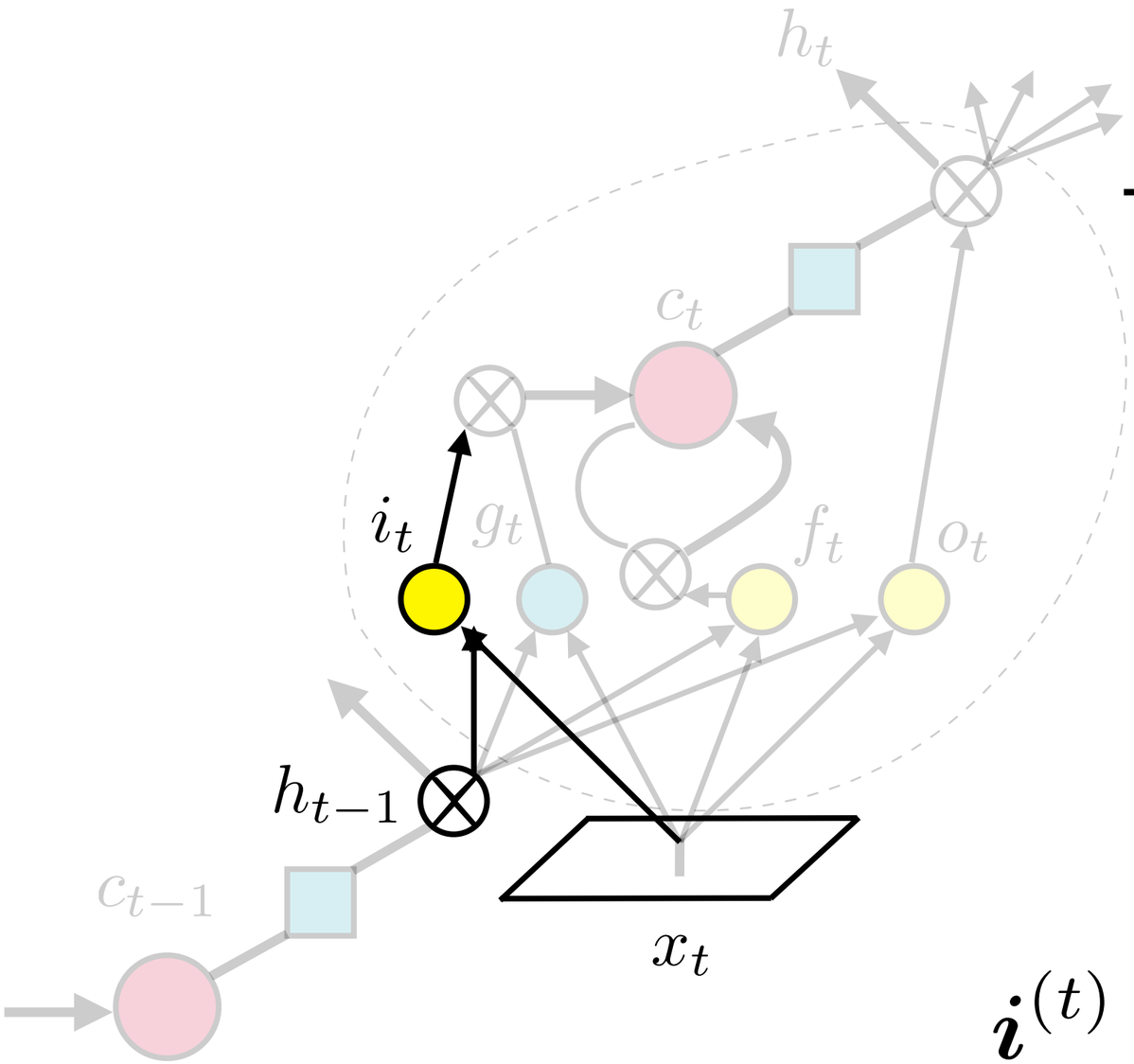




RNN with memory cells
designed to retain long-term information

LSTM - Input gate

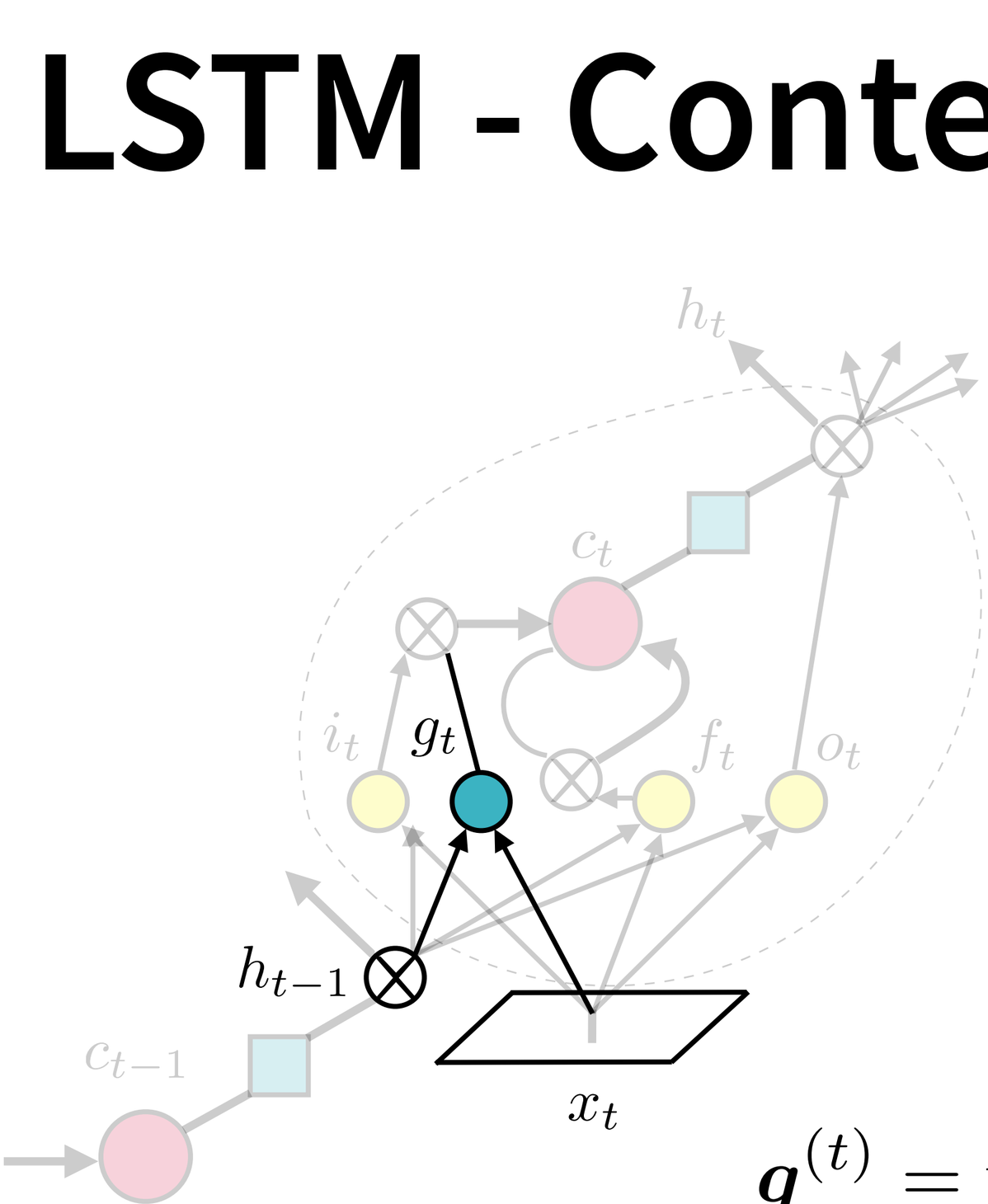




The input gate regulates the flow of new information into the cell

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{b}^i + \boldsymbol{W}^i \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

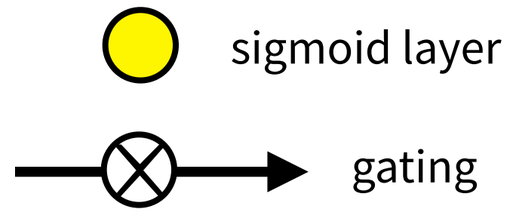
LSTM - Content

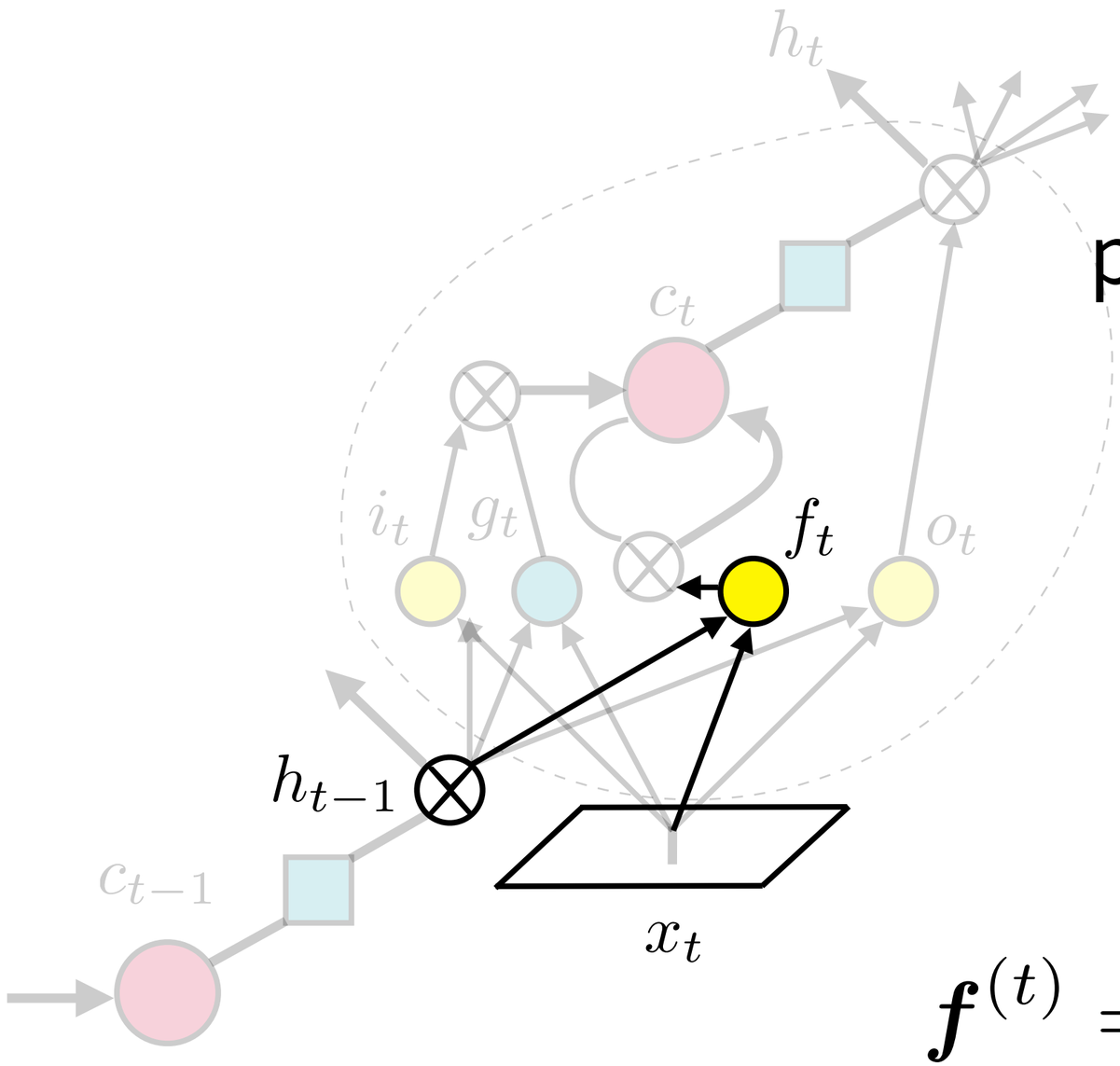


The new information that flows into the cell is a function of the past hidden state and the input (similar to RNN)

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^{g} + \boldsymbol{W}^{g}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

LSTM - Forget gate

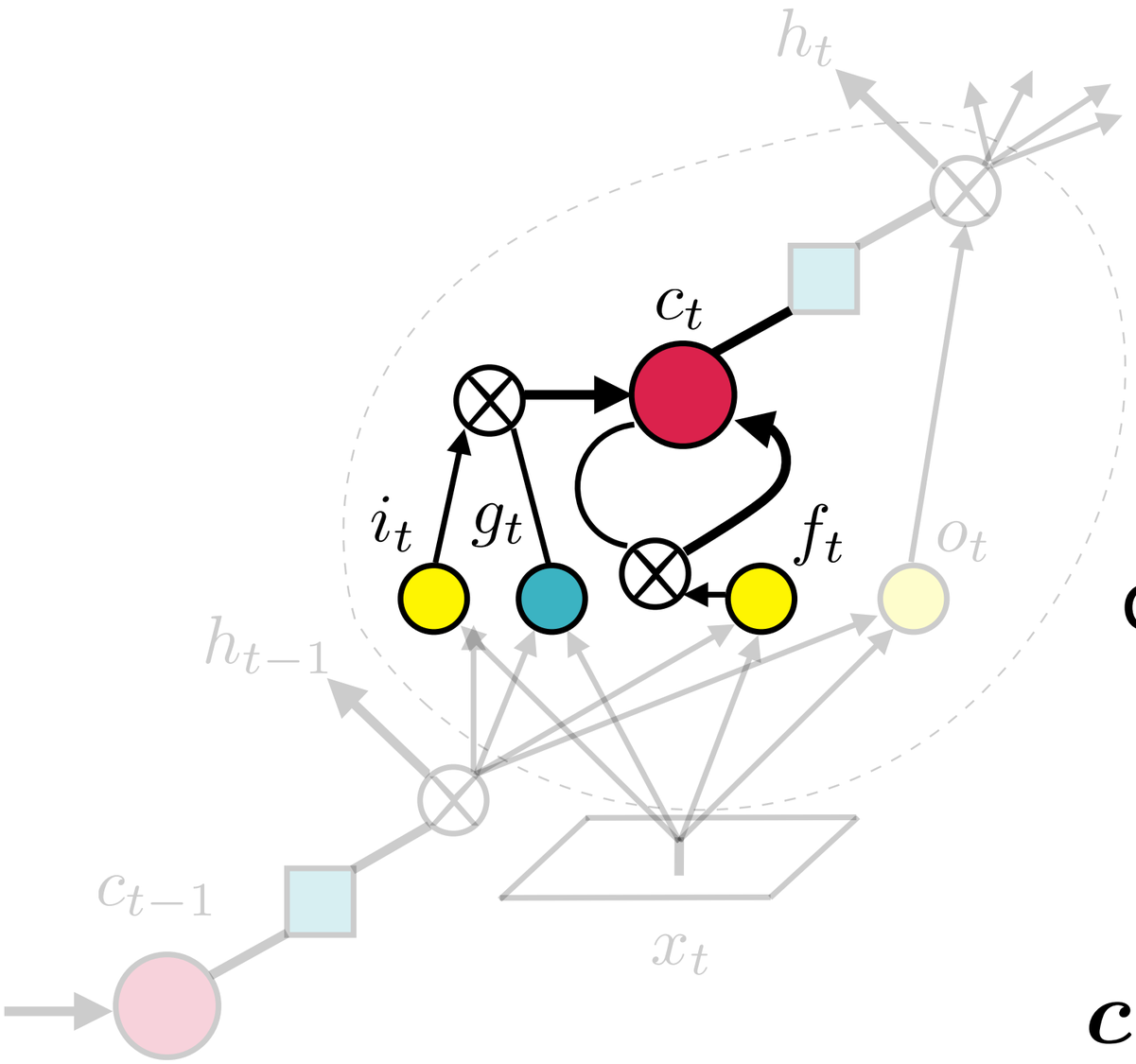


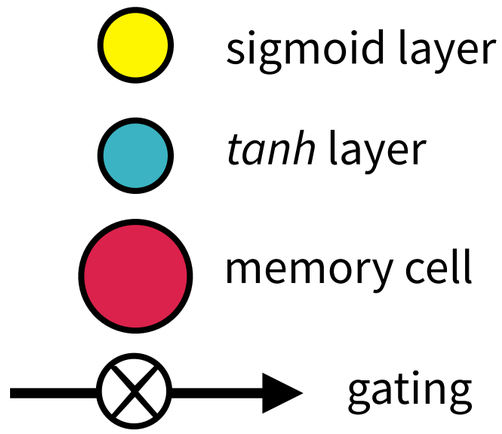


The forget gate regulates preservation of content in the cell

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{b}^f + \boldsymbol{W}^f \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

LSTM - Cell update

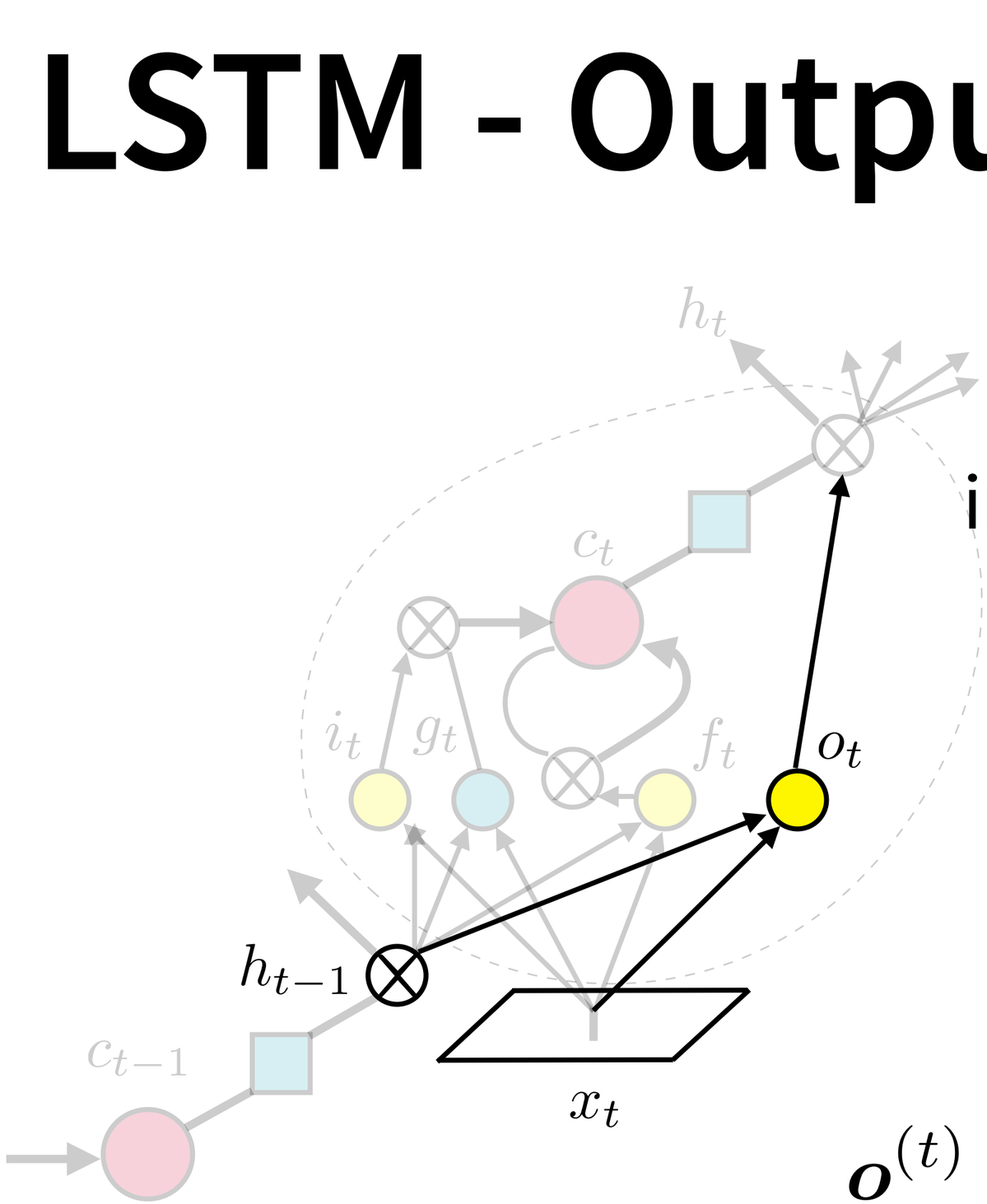




The cell is updated with a combination of new and existing information

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

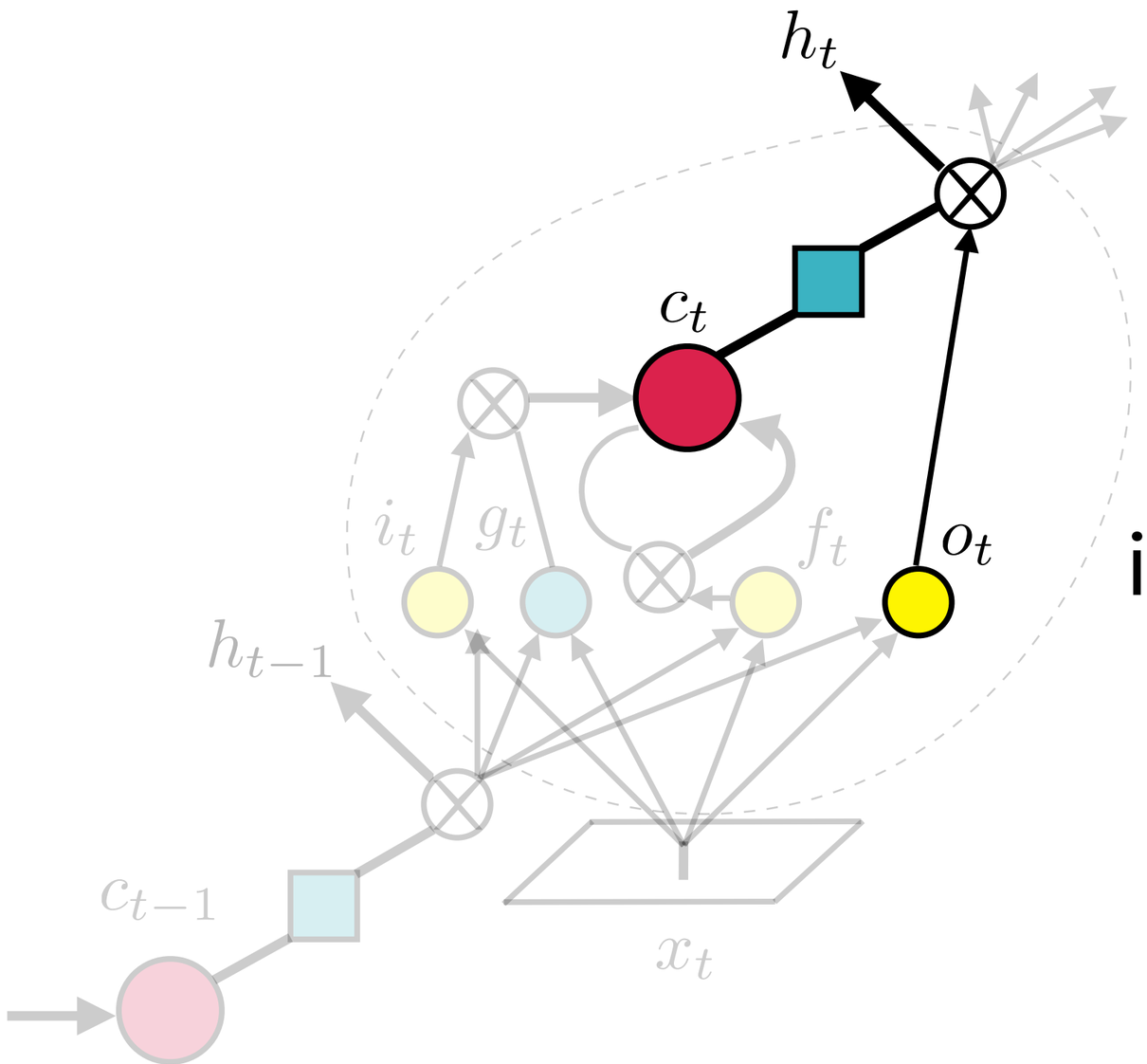
LSTM - Output gate

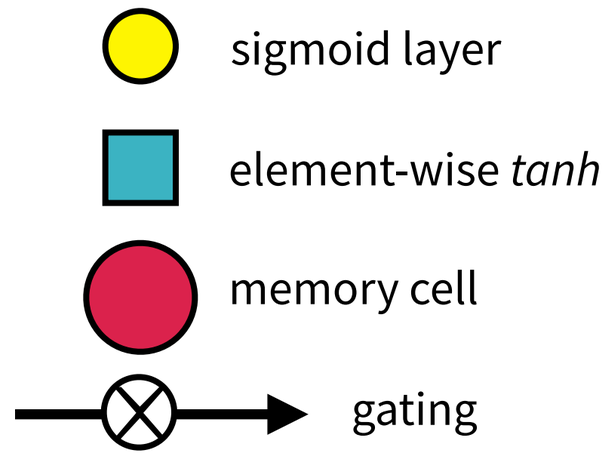


The output gate allows information to flow out of the cell

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{b}^o + \boldsymbol{W}^o \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

LSTM - New state

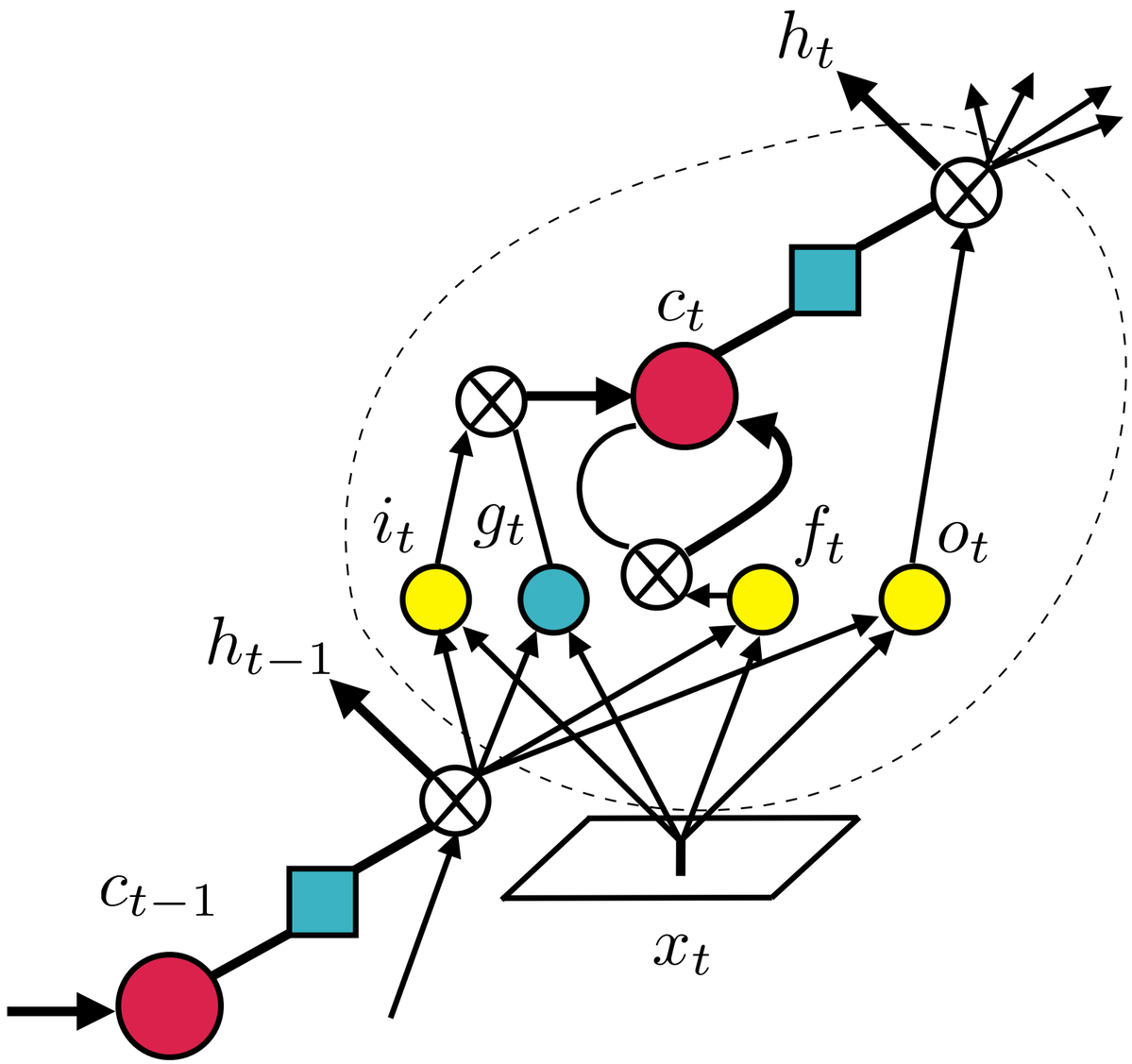


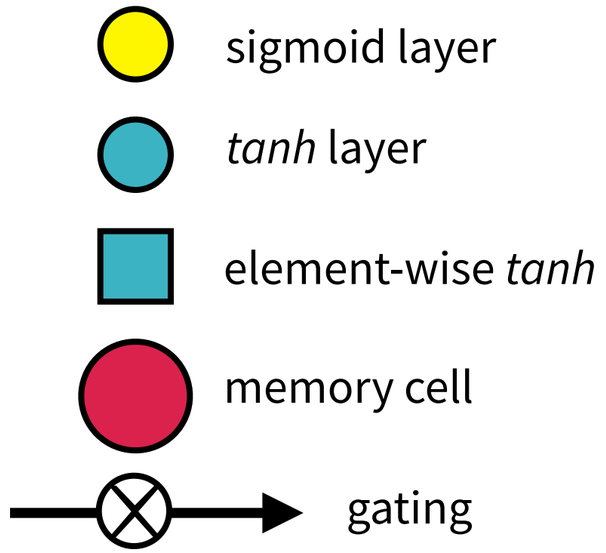


The hidden state update is a nonlinear function of the cell regulated by the output gate

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

LSTM (recap)





$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{b}^i + \boldsymbol{W}^i \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{b}^f + \boldsymbol{W}^f \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

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

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{b}^o + \boldsymbol{W}^o \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

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

Recommended reading:
Chris Olah: Understanding LSTMs (blog)
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

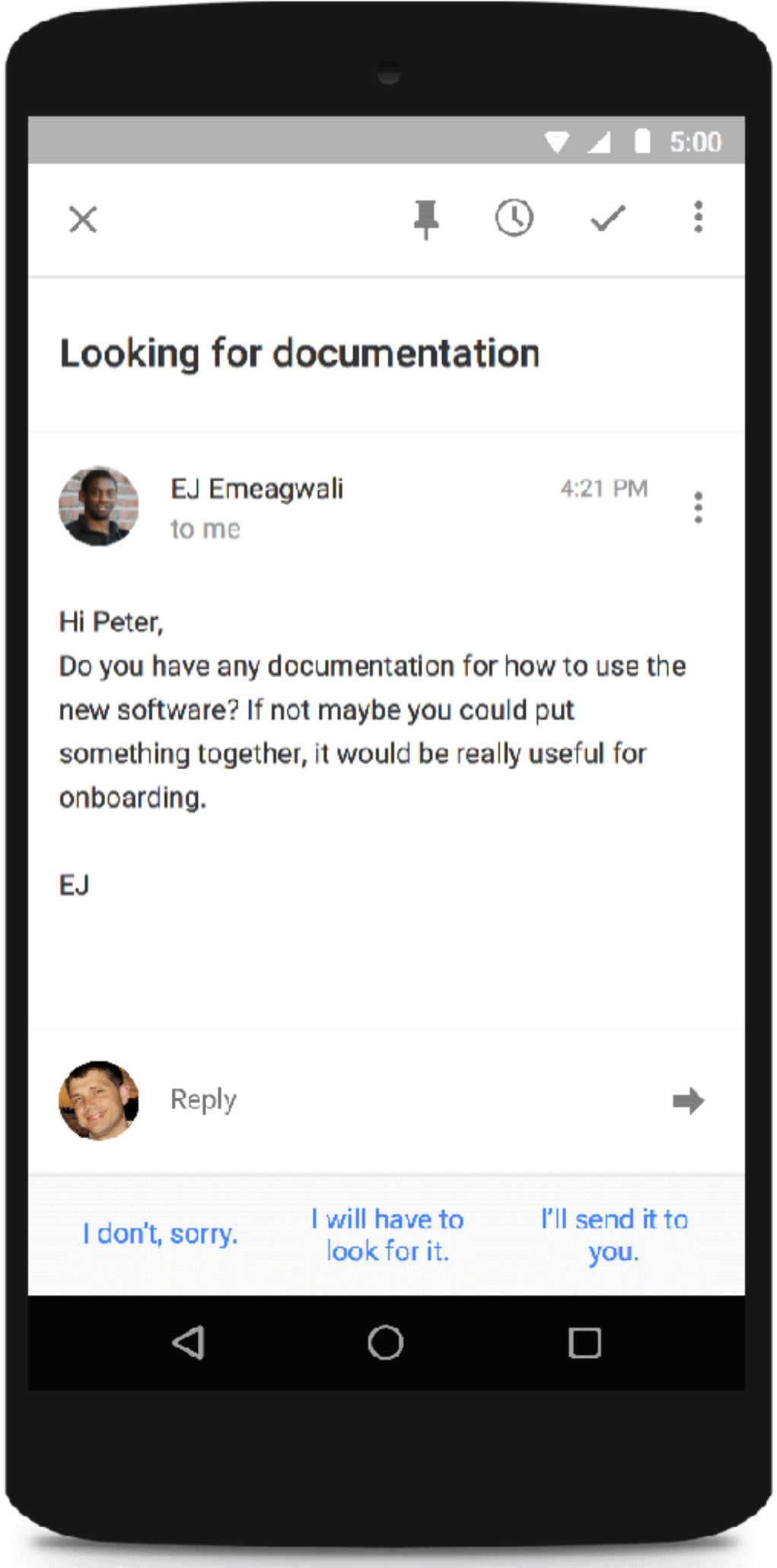


Figure: Google

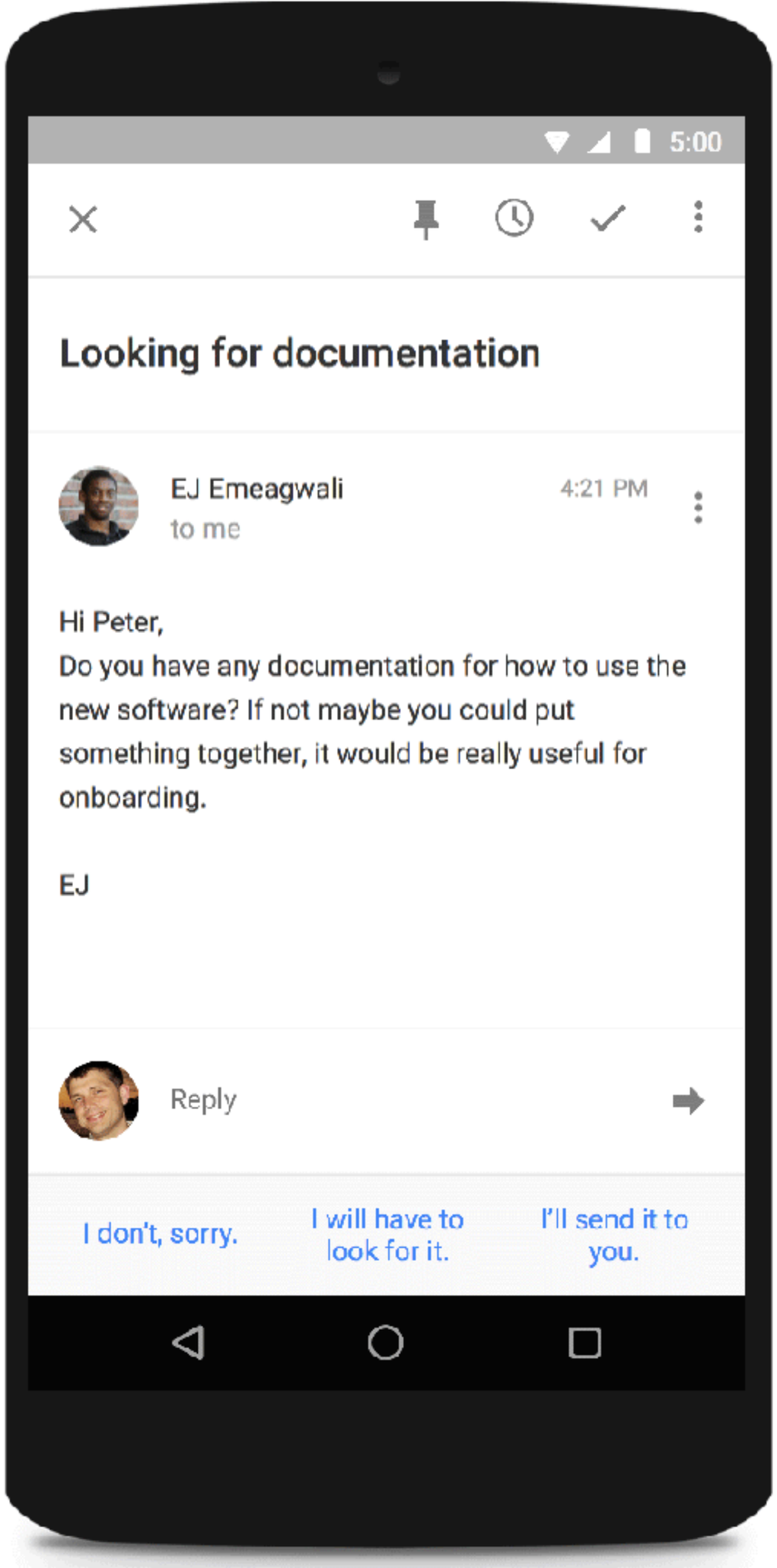
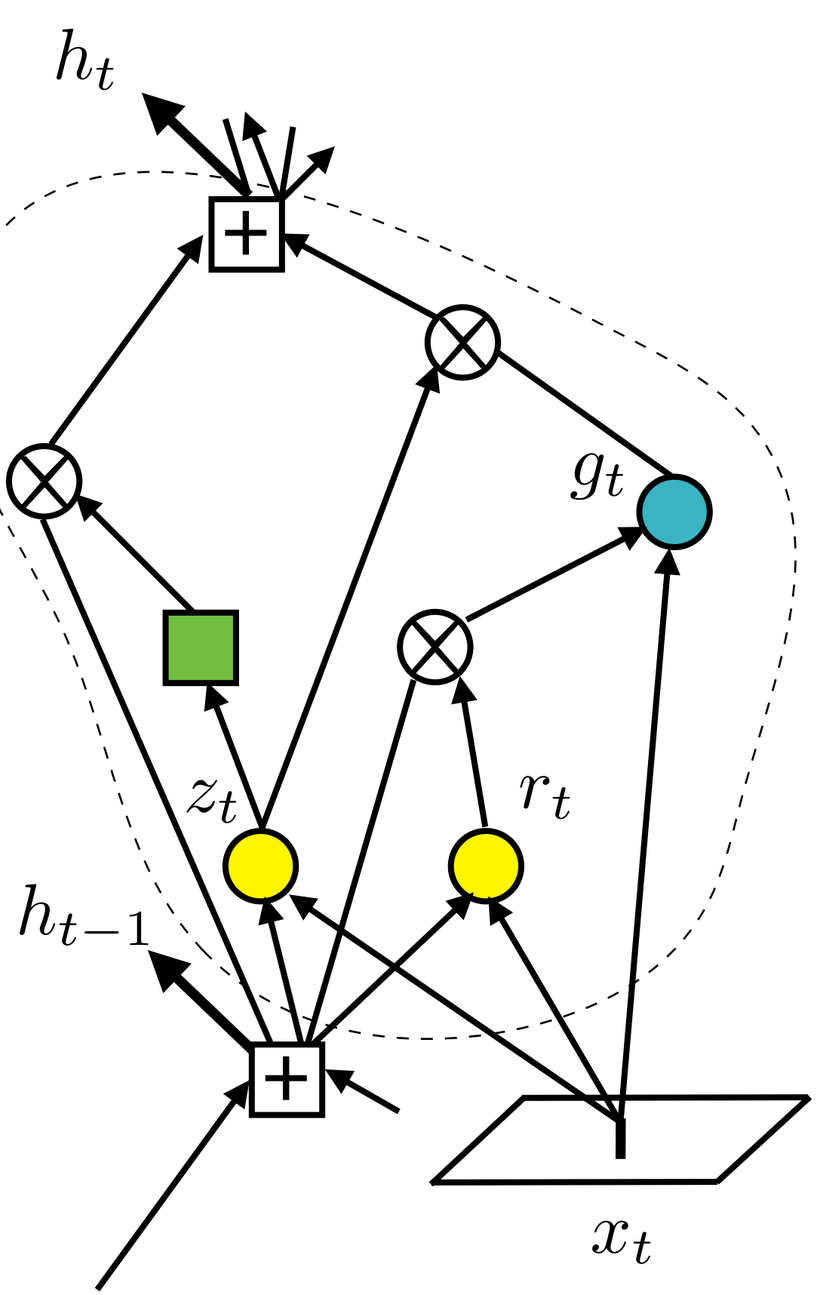
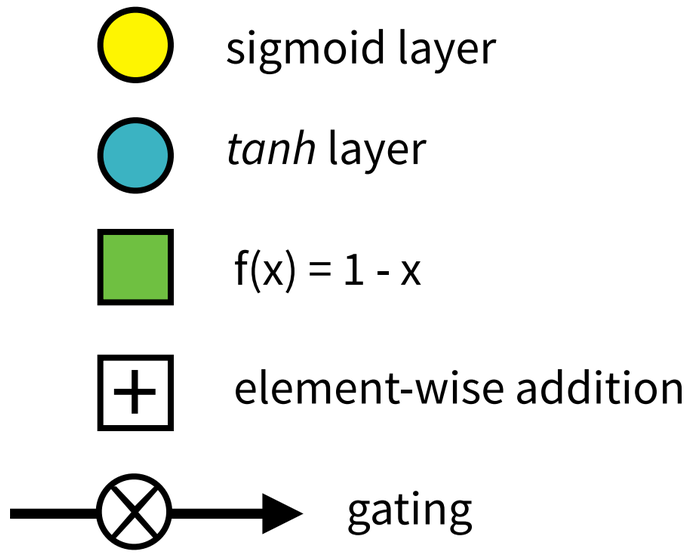


Figure: Google

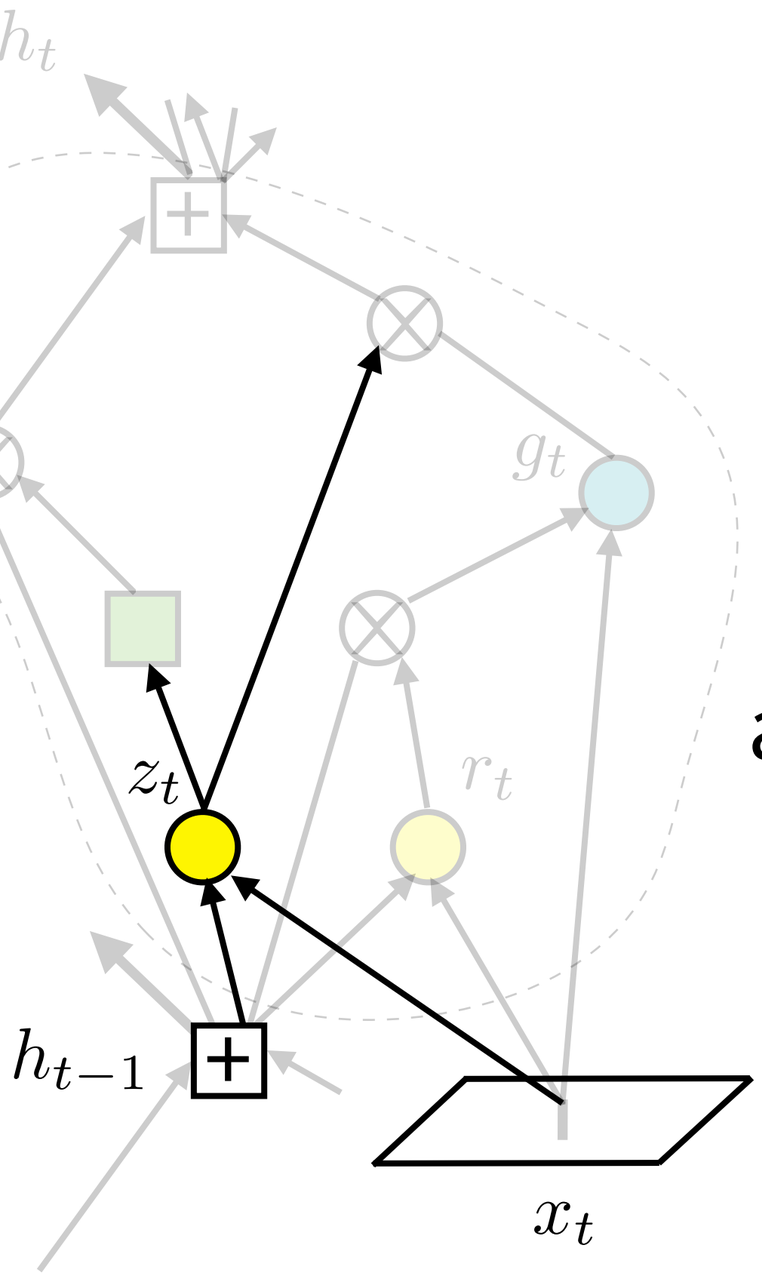
Gated Recurrent Units (GRU)

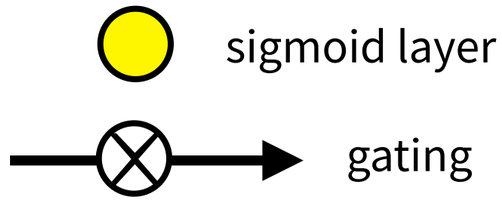




Combine forget and input gates into a single “update gate”
Merges cell state and hidden state

GRU - Update gate

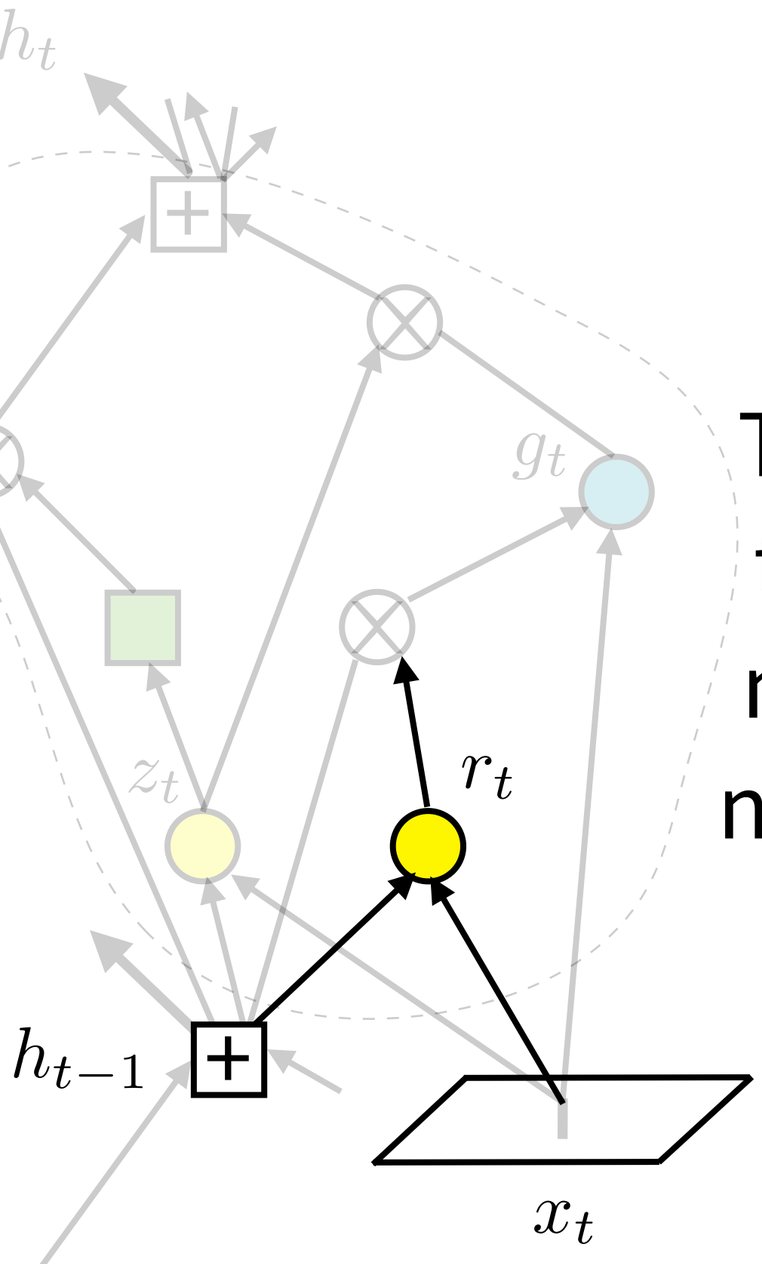


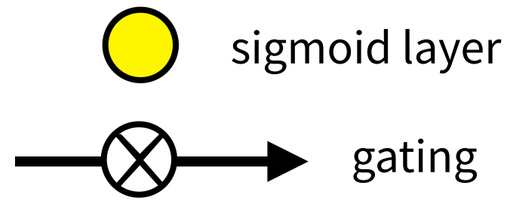


The update gate simultaneously controls the forgetting factor and the decision to update the state

$$\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{b}^z + \boldsymbol{W}^z \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

GRU - Reset gate

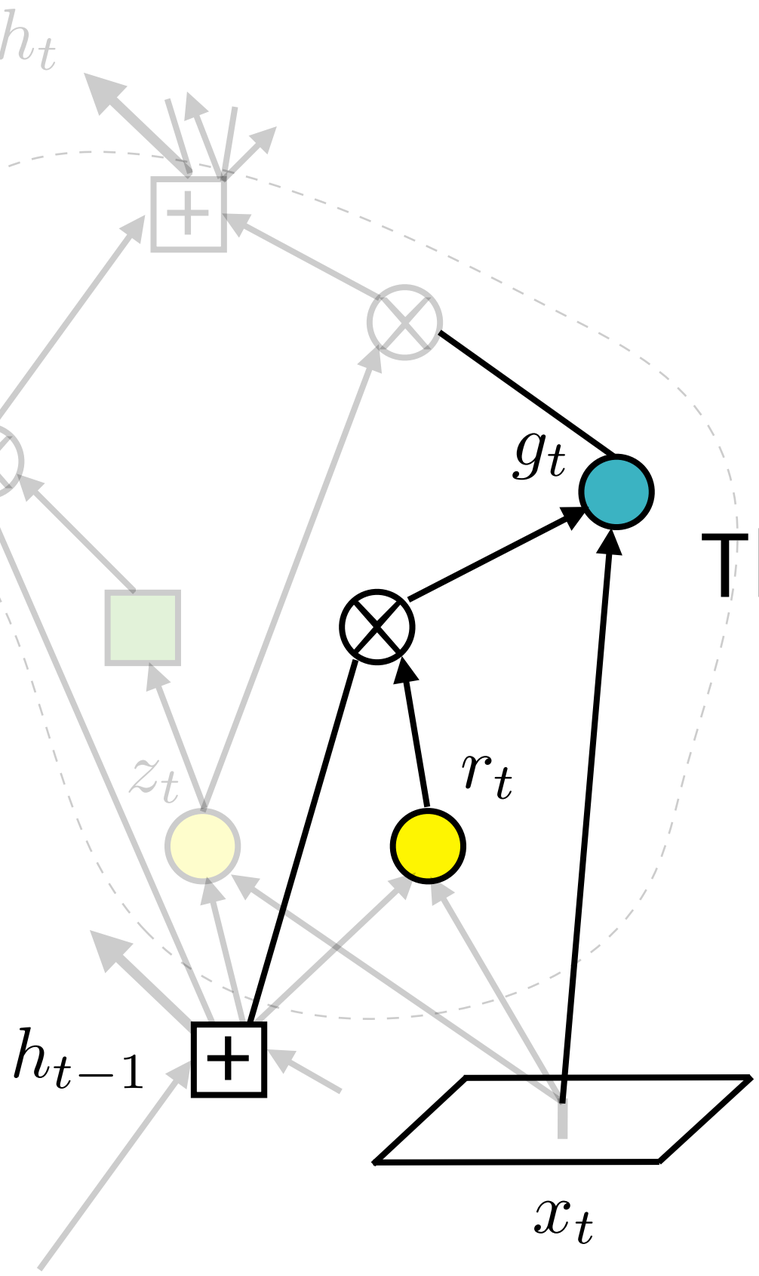


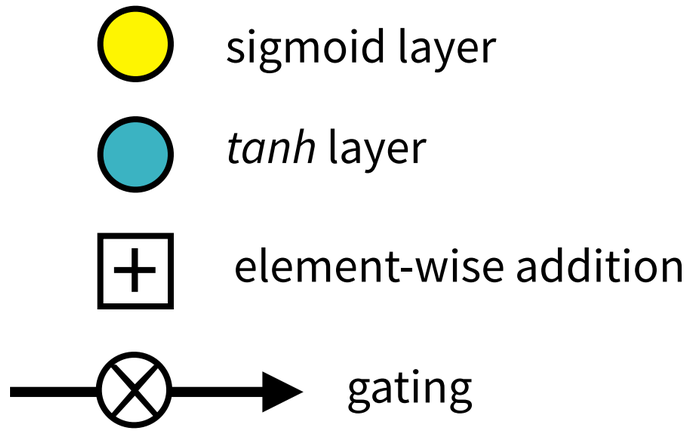


The reset gate controls which part of the old state is used to compute the next state, introducing an additional nonlinear effect b/w past and present

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{b}^r + \boldsymbol{W}^r \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

GRU - Target state

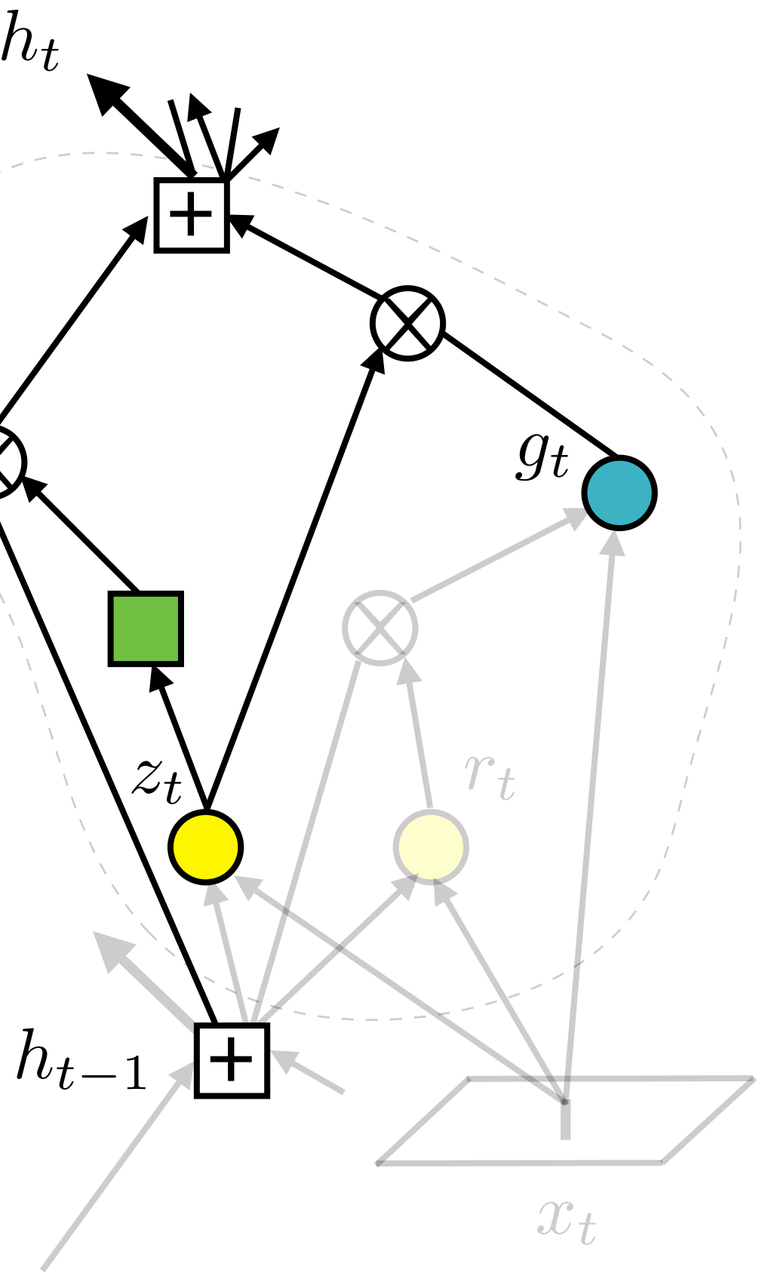


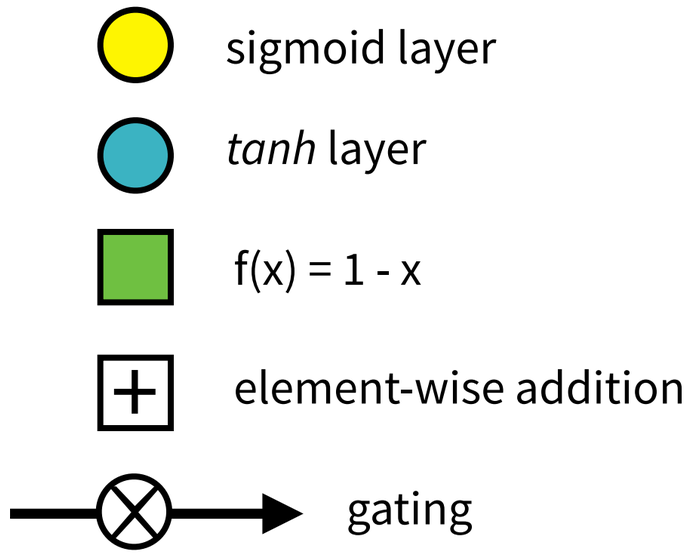


The target state is a nonlinear function of the previous state and input

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g \left[\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

GRU - New state

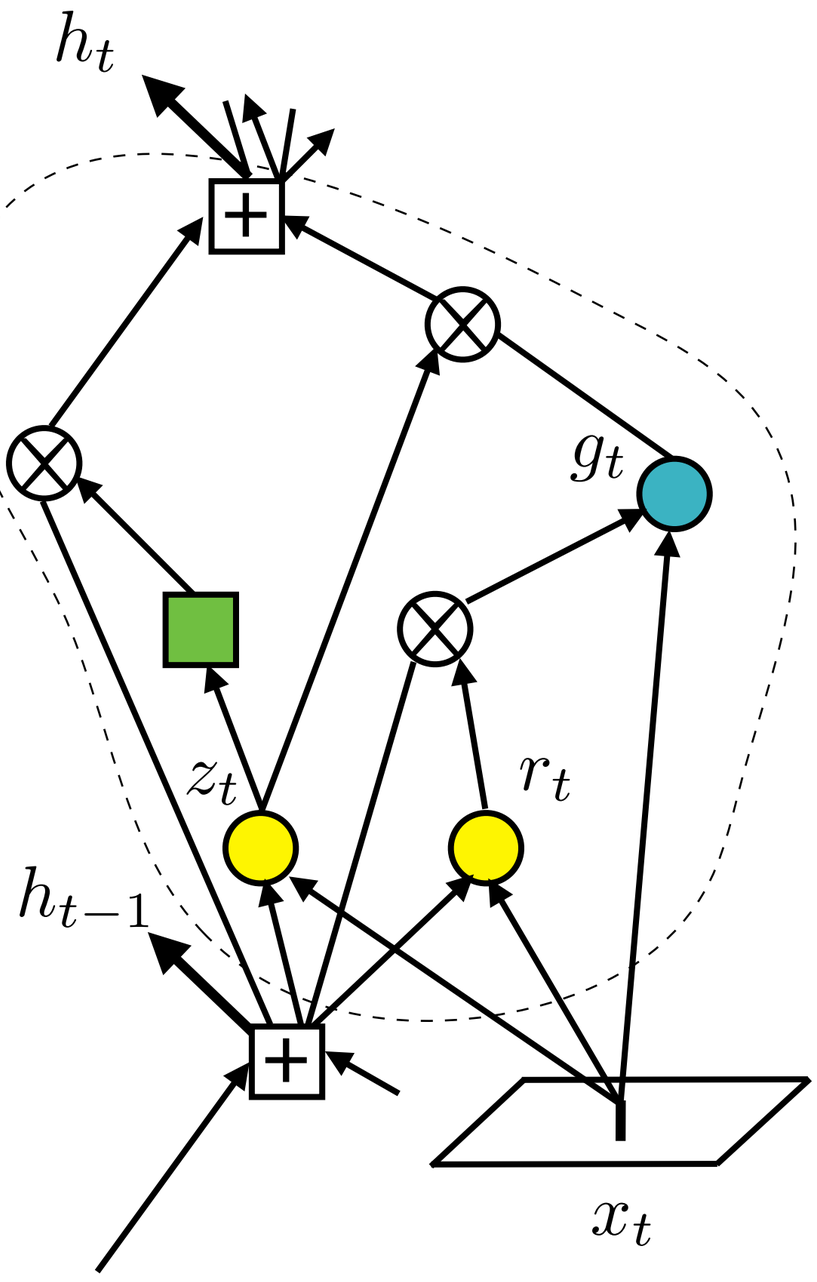


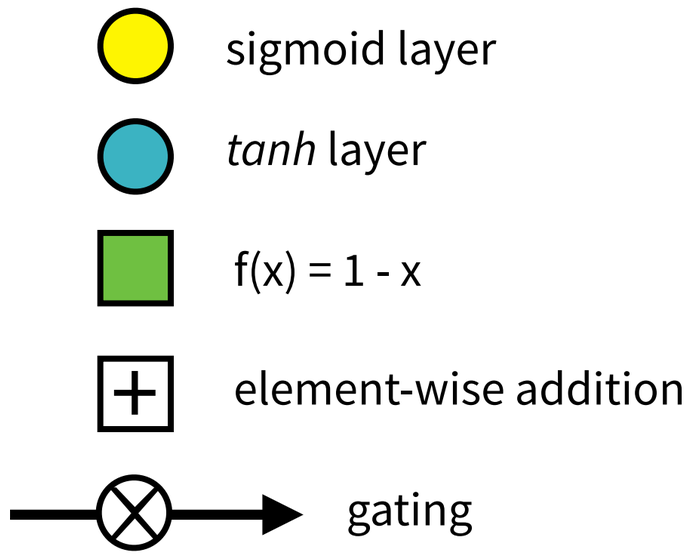


The new state is a function of the target state and previous state

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \boldsymbol{g}^{(t)}$$

GRU (recap)





$$\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{b}^z + \boldsymbol{W}^z \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{b}^r + \boldsymbol{W}^r \left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

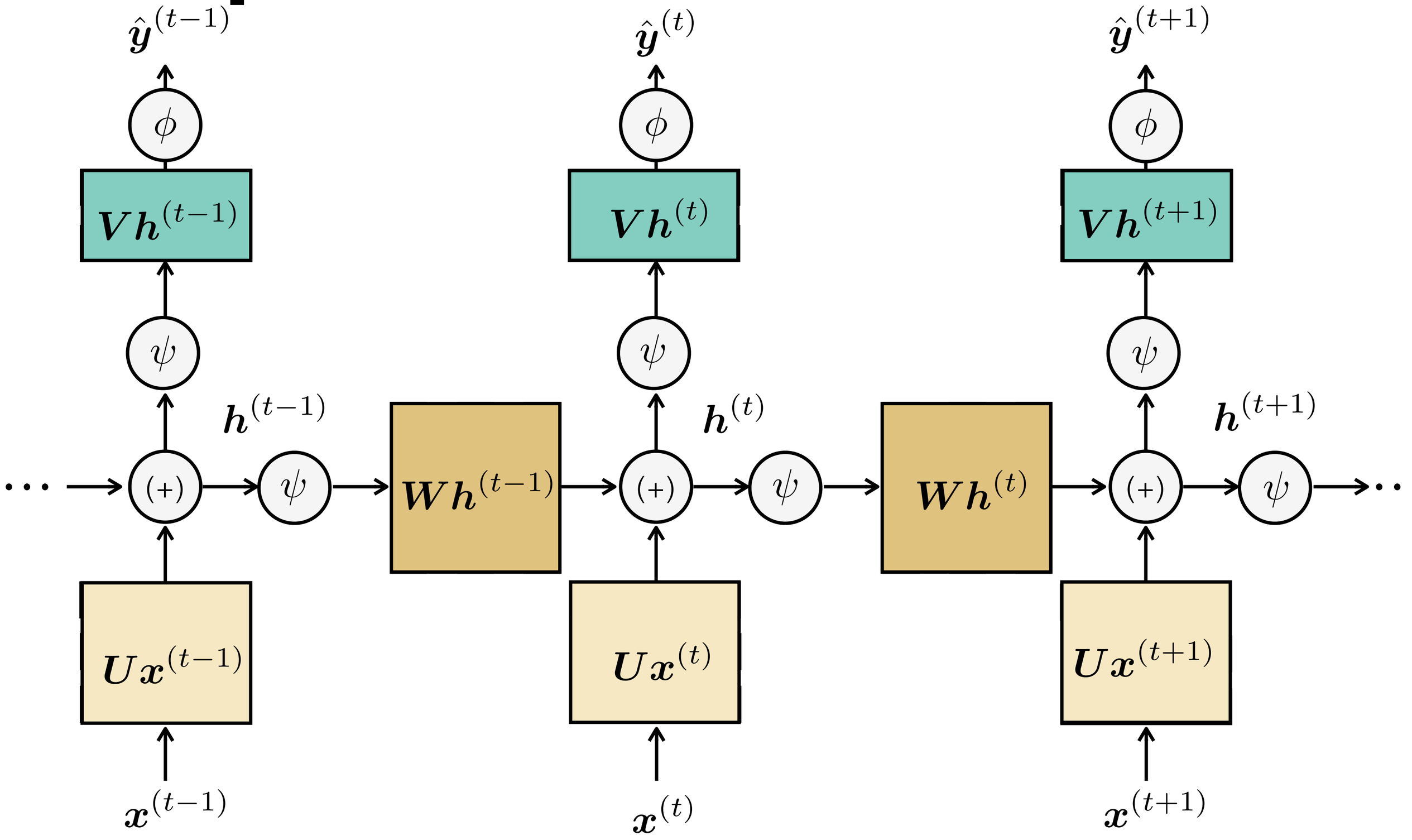
$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g \left[\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \boldsymbol{g}^{(t)}$$

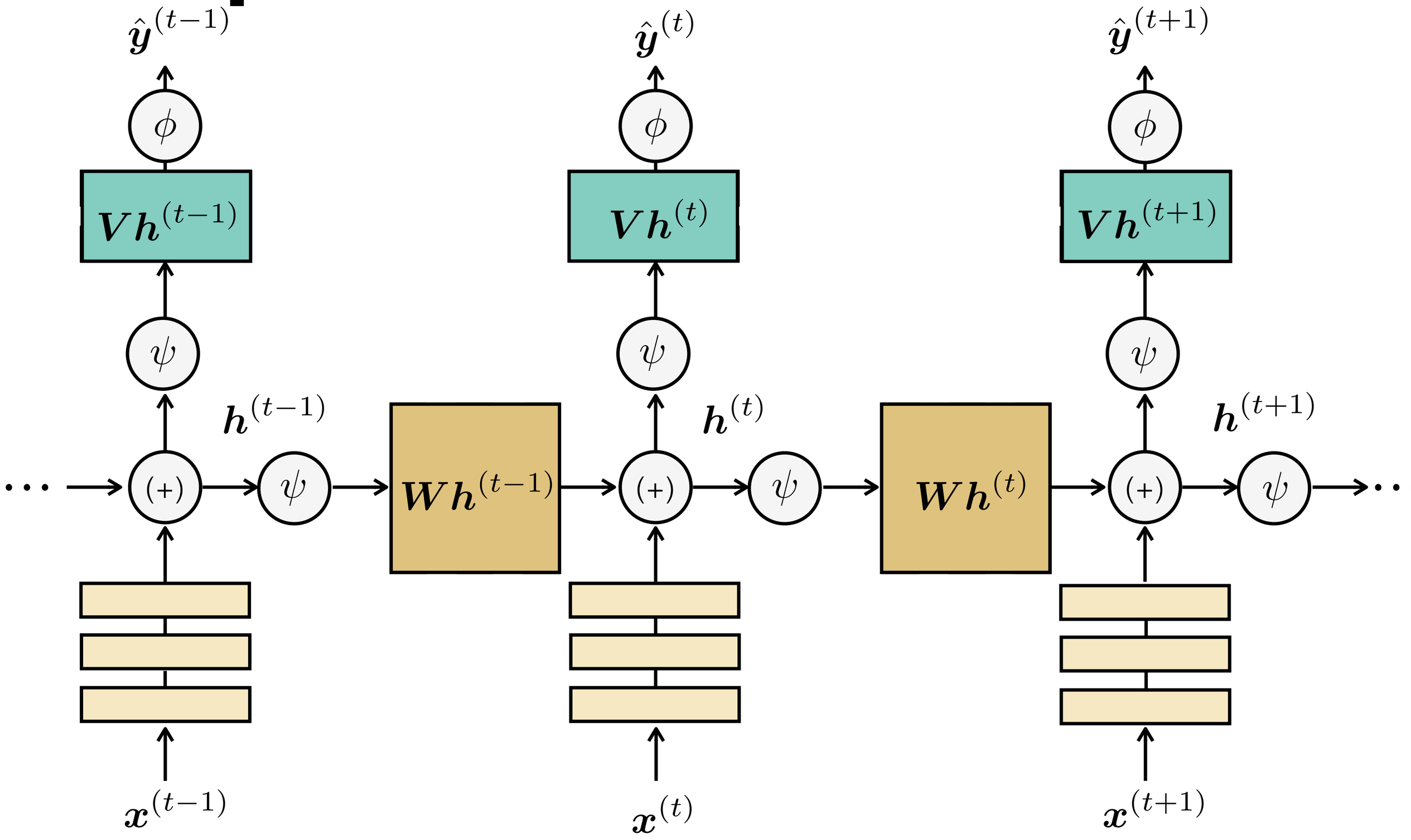
LSTM: A Search Space Odyssey

- So many possibilities!
- Several investigations over architectural variations of the LSTM and GRU found that no variant beat the "baseline" over a wide variety of tasks (Greff et al., 2015), (Jozefowicz et al., 2015)
- Grief et al. (2015) found the forget gate and output activation function to be the LSTM's most critical parts

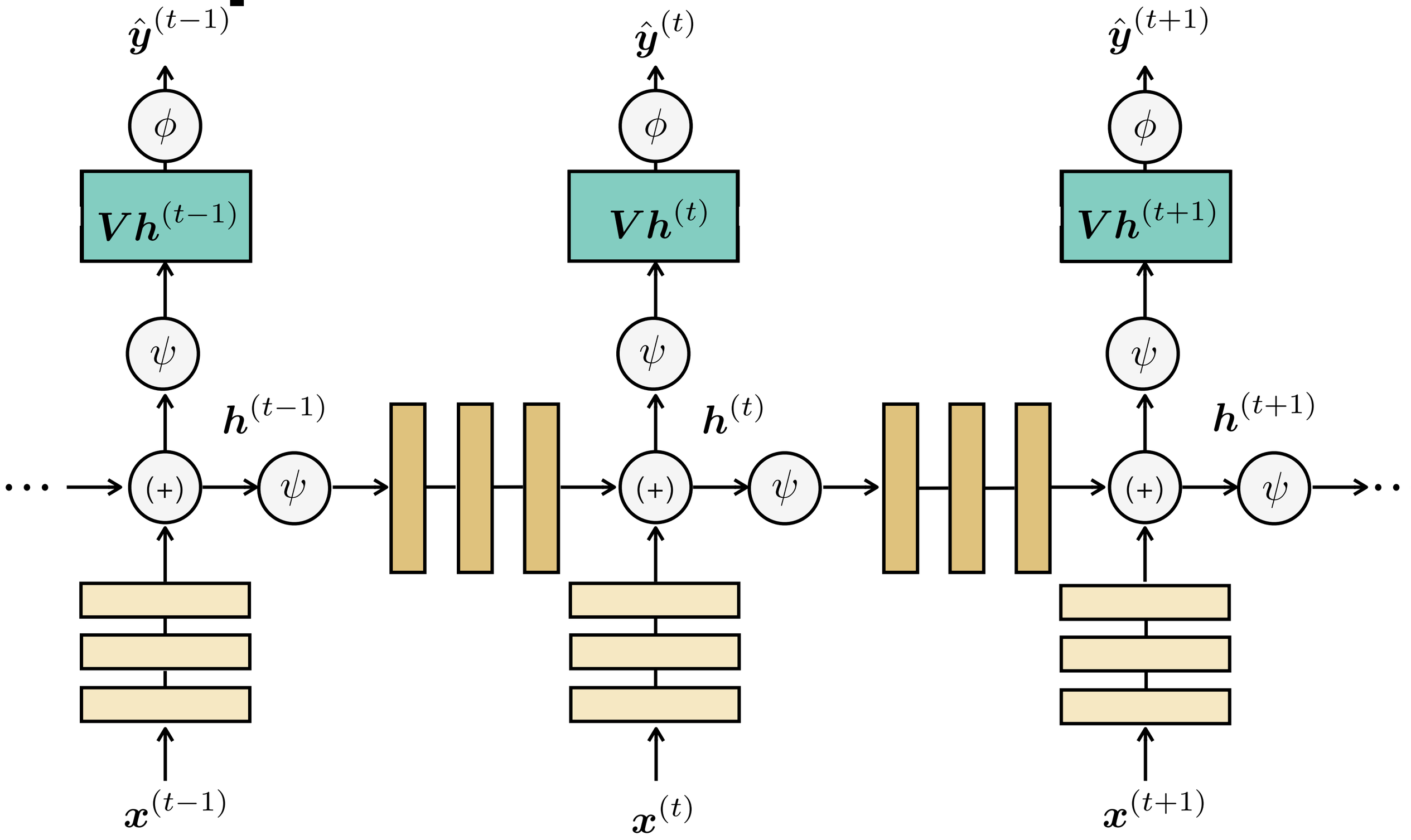
Deep RNNs


$\hat{\boldsymbol{y}}^{(t-1)}$
$\hat{\boldsymbol{y}}^{(t)}$
$\hat{\boldsymbol{y}}^{(t+1)}$
ϕ
$\boldsymbol{V}\boldsymbol{h}^{(t-1)}$
$\boldsymbol{V}\boldsymbol{h}^{(t)}$
$\boldsymbol{V}\boldsymbol{h}^{(t+1)}$
ψ
$\boldsymbol{h}^{(t-1)}$
$\boldsymbol{h}^{(t)}$
$\boldsymbol{h}^{(t+1)}$
(+)
$\boldsymbol{W}\boldsymbol{h}^{(t-1)}$
$\boldsymbol{W}\boldsymbol{h}^{(t)}$
$\boldsymbol{U}\boldsymbol{x}^{(t-1)}$
$\boldsymbol{U}\boldsymbol{x}^{(t)}$
$\boldsymbol{U}\boldsymbol{x}^{(t+1)}$
$\boldsymbol{x}^{(t-1)}$
$\boldsymbol{x}^{(t)}$
$\boldsymbol{x}^{(t+1)}$

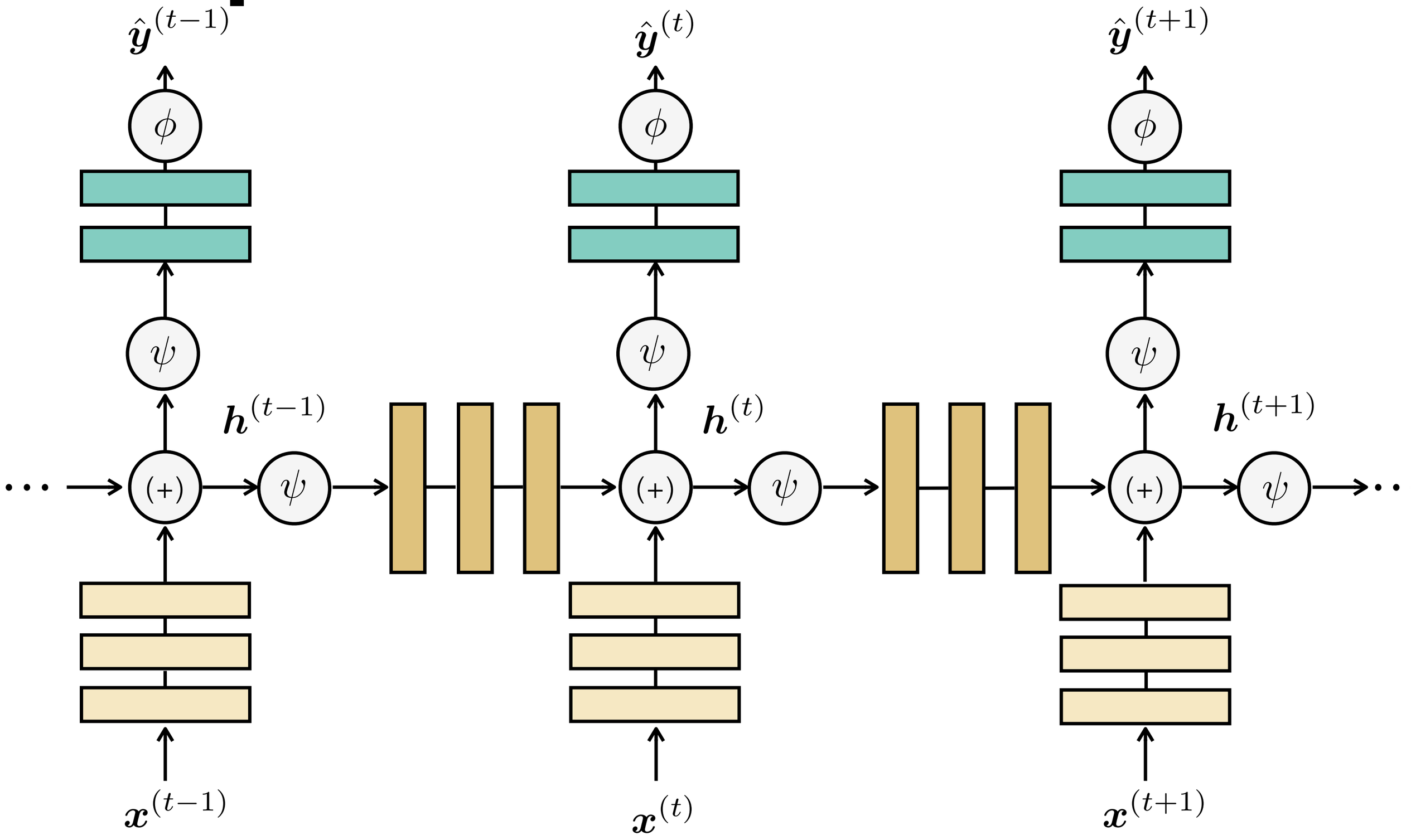
Deep RNNs


$\hat{\boldsymbol{y}}^{(t-1)}$
$\hat{\boldsymbol{y}}^{(t)}$
$\hat{\boldsymbol{y}}^{(t+1)}$
ϕ
$\boldsymbol{V}\boldsymbol{h}^{(t-1)}$
$\boldsymbol{V}\boldsymbol{h}^{(t)}$
$\boldsymbol{V}\boldsymbol{h}^{(t+1)}$
ψ
$\boldsymbol{h}^{(t-1)}$
$\boldsymbol{h}^{(t)}$
$\boldsymbol{h}^{(t+1)}$
(+)
$\boldsymbol{W}\boldsymbol{h}^{(t-1)}$
$\boldsymbol{W}\boldsymbol{h}^{(t)}$
$\boldsymbol{x}^{(t-1)}$
$\boldsymbol{x}^{(t)}$
$\boldsymbol{x}^{(t+1)}$

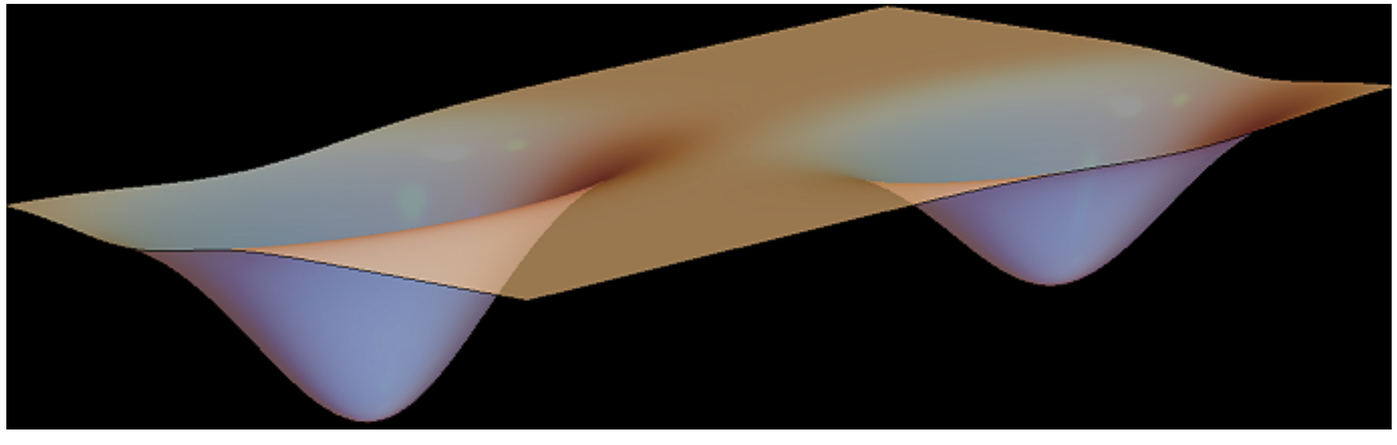
Deep RNNs


$\hat{\boldsymbol{y}}^{(t-1)}$
$\hat{\boldsymbol{y}}^{(t)}$
$\hat{\boldsymbol{y}}^{(t+1)}$
ϕ
$\boldsymbol{V}\boldsymbol{h}^{(t-1)}$
$\boldsymbol{V}\boldsymbol{h}^{(t)}$
$\boldsymbol{V}\boldsymbol{h}^{(t+1)}$
ψ
$\boldsymbol{h}^{(t-1)}$
$\boldsymbol{h}^{(t)}$
$\boldsymbol{h}^{(t+1)}$
(+)
$\boldsymbol{x}^{(t-1)}$
$\boldsymbol{x}^{(t)}$
$\boldsymbol{x}^{(t+1)}$

Deep RNNs


$\hat{\boldsymbol{y}}^{(t-1)}$
$\hat{\boldsymbol{y}}^{(t)}$
$\hat{\boldsymbol{y}}^{(t+1)}$
ϕ
ψ
$\boldsymbol{h}^{(t-1)}$
$\boldsymbol{h}^{(t)}$
$\boldsymbol{h}^{(t+1)}$
(+)
$\boldsymbol{x}^{(t-1)}$
$\boldsymbol{x}^{(t)}$
$\boldsymbol{x}^{(t+1)}$

Solutions: Optimization



- 2nd-order methods (Martens and Sutskever 2011)
- Reservoirs (Jaeger and Haas 2004)
- Gradient clipping/normalization (Pascanu et al. 2012, Mikolov 2012)
- Careful initialization (Sutskever et al. 2013)

Summary (and Advice!)

- Keep the architecture simple!
  - GRUs, gradient clipping, careful initialization
- Use well-tested, well-supported open-source tools:
  - Tensor Flow, PyTorch, Caffe
- Don’t forget the flexibility of input-output structure