Theories behind recurrent neural networks

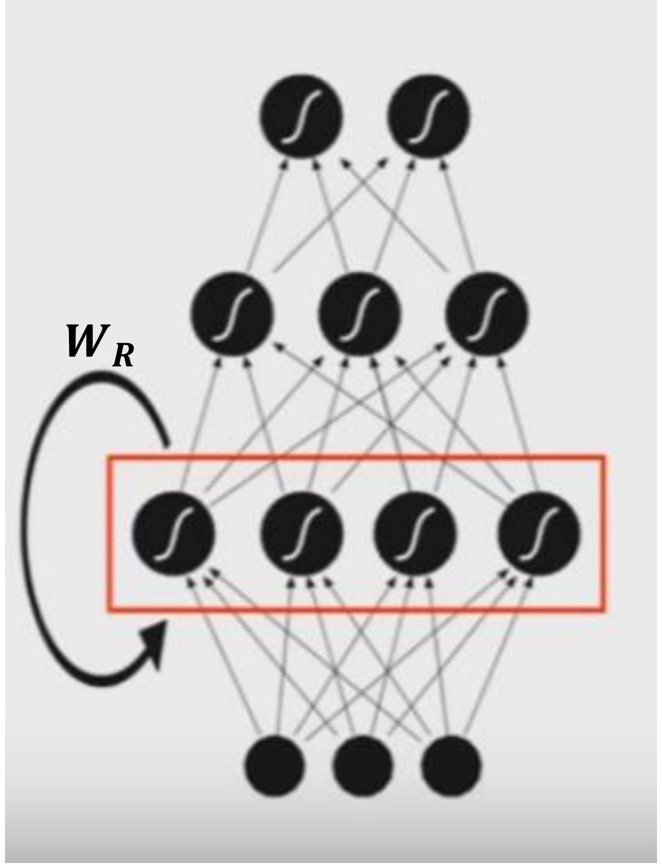
Content

- RNN Introduction
  - Variants on Recurrent nets
  - Training RNN
- Vanishing Gradients and Exploding Gradients
- Proposed Solutions
- Deep Belief Networks
- Memory Systems
  - Gated Recurrent Unit (GRU)
  - Long-Shot Term Memory (LSTM)
- Different Types of Recurrent Memory Models
  - Bi-directional RNN
  - Neural Turing Machine (NTM)
  - Hopfield Neural Networks (HNN)

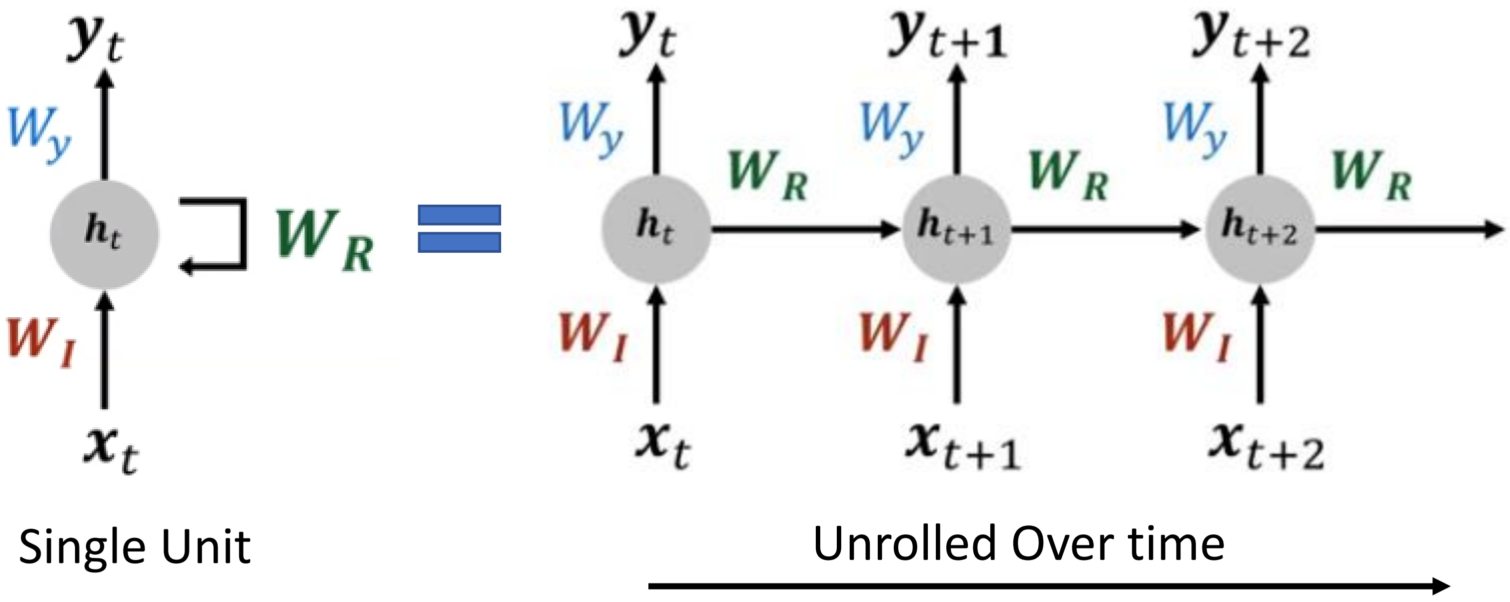
- **RNN Introduction**
  - Variants on Recurrent nets
  - Training RNN
- Vanishing Gradients and Exploding Gradients
- Proposed Solutions
- Deep Belief Networks
- Memory Systems
  - Gated Recurrent Unit (GRU)
  - Long-Shot Term Memory (LSTM)
- Different Types of Recurrent Memory Models
  - Bi-directional RNN
  - Neural Turing Machine (NTM)
  - Hopfield Neural Networks (HNN)

RNN Introduction

General perspective



If we consider only one single layer Neural Network



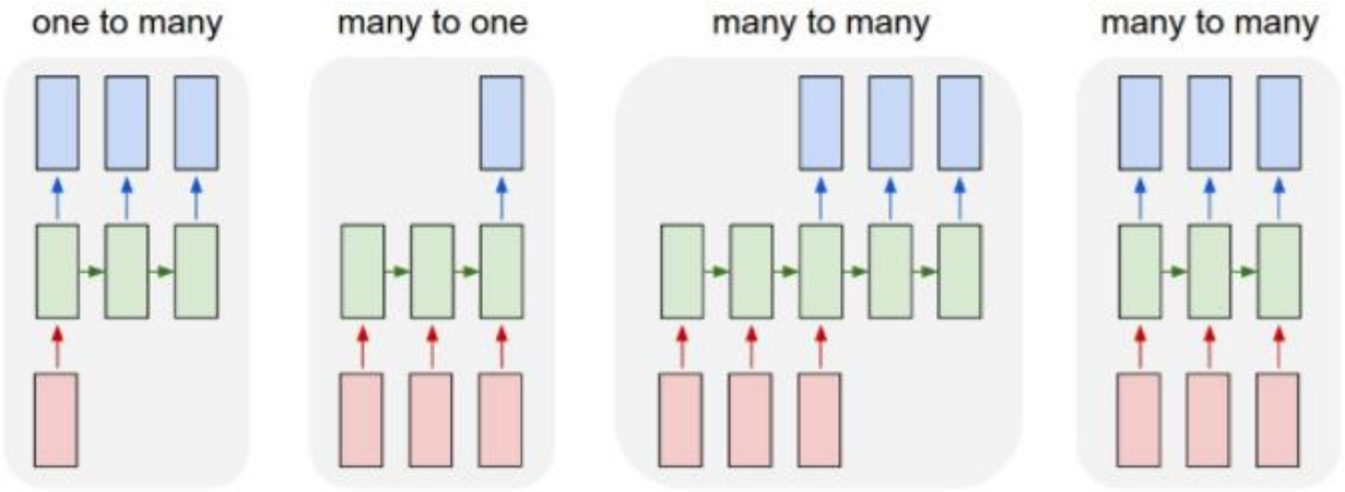
Single Unit

Unrolled Over time

$$\boldsymbol{h}^{(t)} = g_h(W_{\mathrm{I}}\boldsymbol{x}^{(t)} + W_{\mathrm{R}}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}^{(t)} = g_y(W_y\boldsymbol{h}^{(t)} + \boldsymbol{b}_y)$$

Variants on recurrent nets



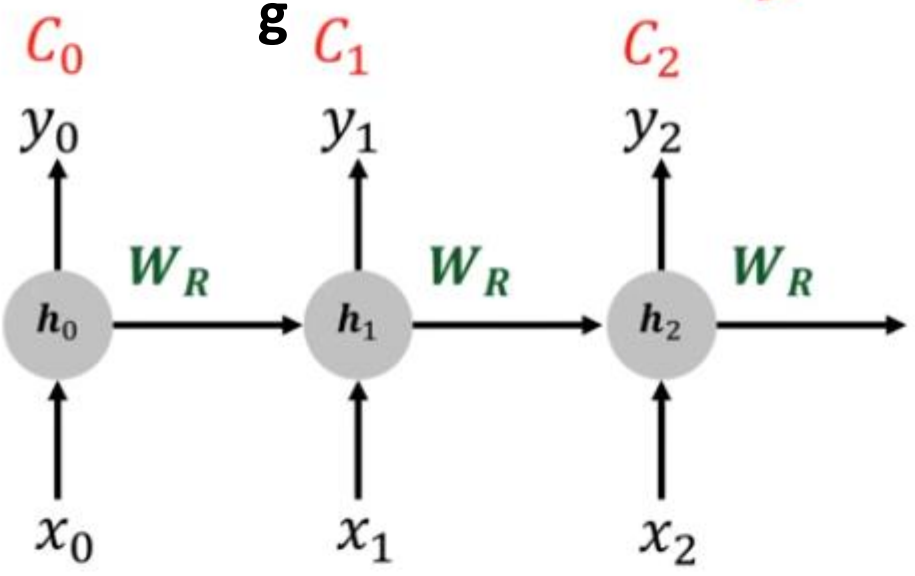
1 2 3 4

1 : Observe only single time series data and predict few time steps ahead
2 : Wait for some inputs and give an answer (What's your name? → Mike)
3 : This can be use to predict a time series ahead by observing historical time series data (Predicting one month head using previous month data)
4 : Online version (Do not wait for a specific amount of input data, Motion tracking of videos)

Training RNN (BPTT – Back Propagation Through Time)

$$\boldsymbol{h}^{(t)} = g_h\overbrace{(W_{\mathrm{I}}\boldsymbol{x}^{(t)} + W_{\mathrm{R}}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h}^{\mathbf{a}})$$

$$\boldsymbol{y}^{(t)} = g_y\underbrace{(W_y\boldsymbol{h}^{(t)} + \boldsymbol{b}_y}_{\mathbf{g}})$$



Combine via:

$$\frac{\partial C}{\partial W_R} = \sum_t \frac{\partial C_t}{\partial W_R}$$

$$\frac{\partial C_2}{\partial W_R} = \frac{\partial C_2}{\partial y_2}\frac{\partial y_2}{\partial g}\frac{\partial g}{\partial h_2}\frac{\partial h_2}{\partial a}\frac{\partial a}{\partial W_R}$$

But : $a = (W_1\boldsymbol{x}_2 + W_R\boldsymbol{h}_1 + \boldsymbol{b}_h)$

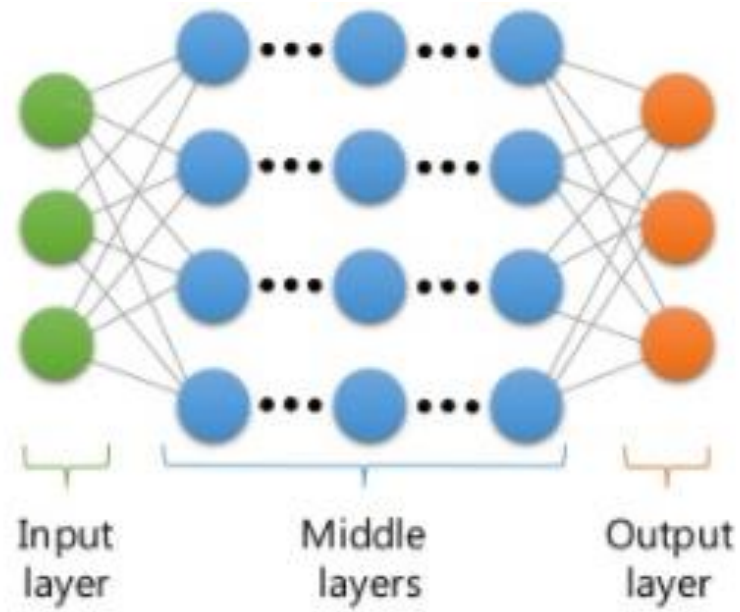
Now $\boldsymbol{h}_1$ depend on W_R too $\boldsymbol{h}_1 = g_h(W_1\boldsymbol{x}_1 + W_R\boldsymbol{h}_0 + \boldsymbol{b}_h)$

$$\frac{\partial C_{100}}{\partial W_R} = \frac{\partial C_{100}}{\partial y_{100}} \cdots W_R \frac{\partial h_{100}}{\partial a_{100}} \cdots W_R \frac{\partial h_{99}}{\partial a_{100}} \cdots$$

- RNN Introduction
  - Variants on Recurrent nets
  - Training RNN
- **Vanishing Gradients and Exploding Gradients**
- Proposed Solutions
- Deep Belief Networks
- Memory Systems
  - Gated Recurrent Unit (GRU)
  - Long-Shot Term Memory (LSTM)
- Different Types of Recurrent Memory Models
  - Bi-directional RNN
  - Neural Turing Machine (NTM)
  - Hopfield Neural Networks (HNN)

Vanishing and Exploding Gradient

$$h_t = W_R f(h_{t-1}) + W_x x_t$$
$$y_t = W_y f(h_t)$$
$$\frac{\partial C_t}{\partial W_R} = \frac{\partial C_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W_R}$$
$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}} = {W_R}^T diag[f'(\boldsymbol{h}_{i-1})]$$



Lets if we pick a basis function f and its derivative f' is always bounded by a constant γ_f. Then

$$\left\|\frac{\partial h_i}{\partial h_{i-1}}\right\| \le \left\|{W_R}^T\right\| \|diag[f'(\boldsymbol{h}_{i-1})]\| \le \gamma_{W_R}\gamma_f$$

$$\left\|\frac{\partial h_t}{\partial h_k}\right\| \le (\gamma_{W_R}\gamma_f)^{t-k}$$

$$\left\|\frac{\partial h_t}{\partial h_k}\right\| \leq (\gamma_{W_R}\gamma_f)^{t-k}$$

$(\gamma_{W_R}\gamma_f)$ < 1 → **Vanishing Gradient** $(\gamma_{W_R}\gamma_f)$ ≥ 1 → **Exploding Gradient**

Vanishing and Exploding Gradient will effect in the Gradient update of the learning process

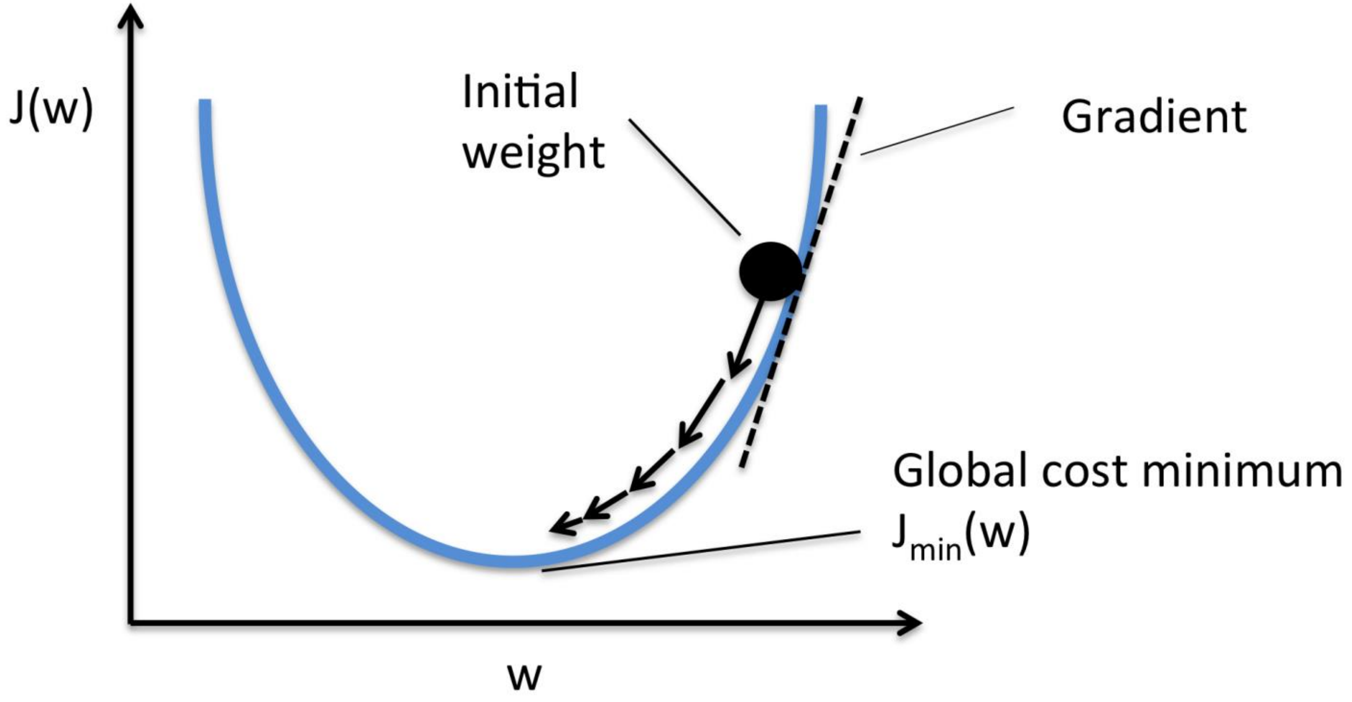
Gradient Descent Algorithm

$Repeat\ until\ convergence\ \{$

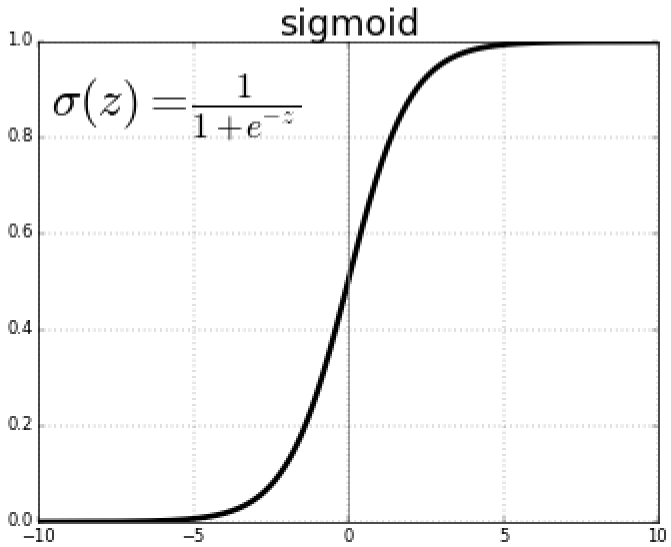
$$W_j = W_j - \alpha \underbrace{\frac{\boldsymbol{\partial}}{\boldsymbol{\partial W_j}} \boldsymbol{C}(\boldsymbol{y_t}, \boldsymbol{W_j})}_{\text{Gradient}}$$

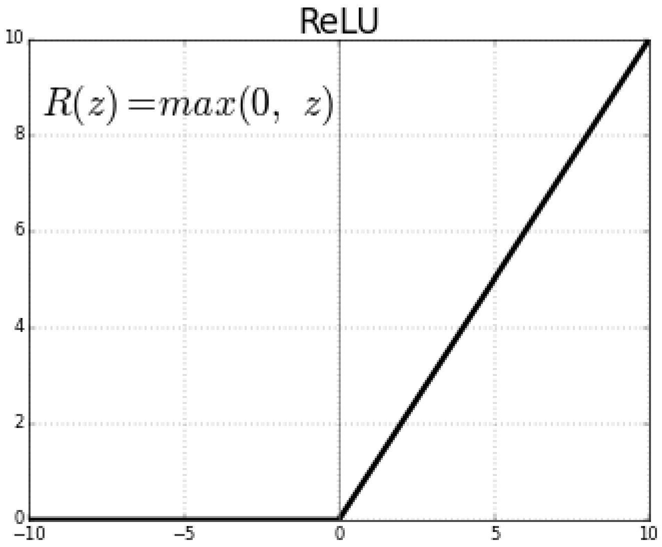
$\}$

Learning Rate



Proposed solutions

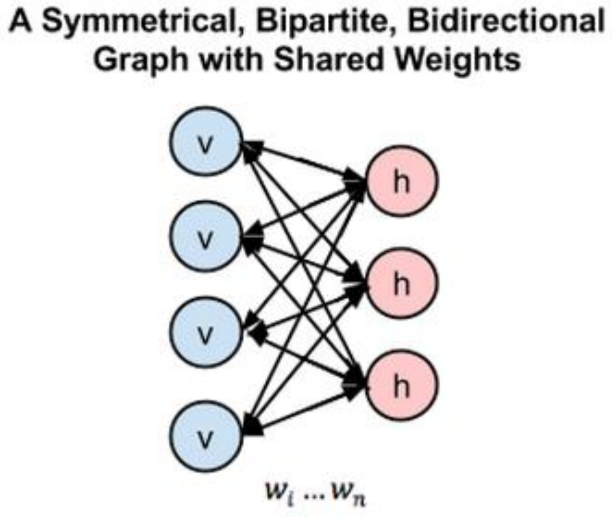




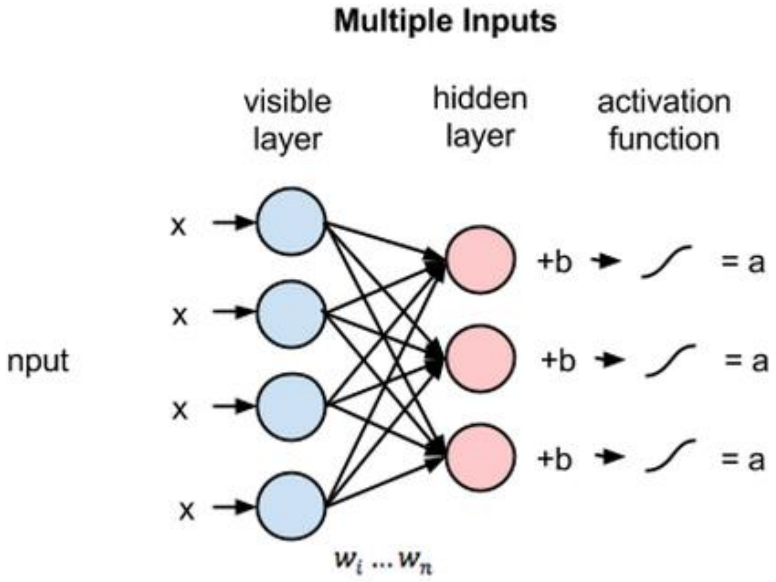
- Exploding Gradients
  - Truncated BPTT
  - Clip Gradients at threshold
  - **RMSprop** to adjust learning rate
- Vanishing Gradients
  - Weight initialization (Restricted Boltzmann Machine)
  - ReLu activation function (but this can lead to exploding gradient)
  - RMSprop
  - LSTM
  - GRU
- We can set the $\boldsymbol{W_R}$ = 1 , but then we cannot control the learning amount from the past data.

Weight initialization using RBM

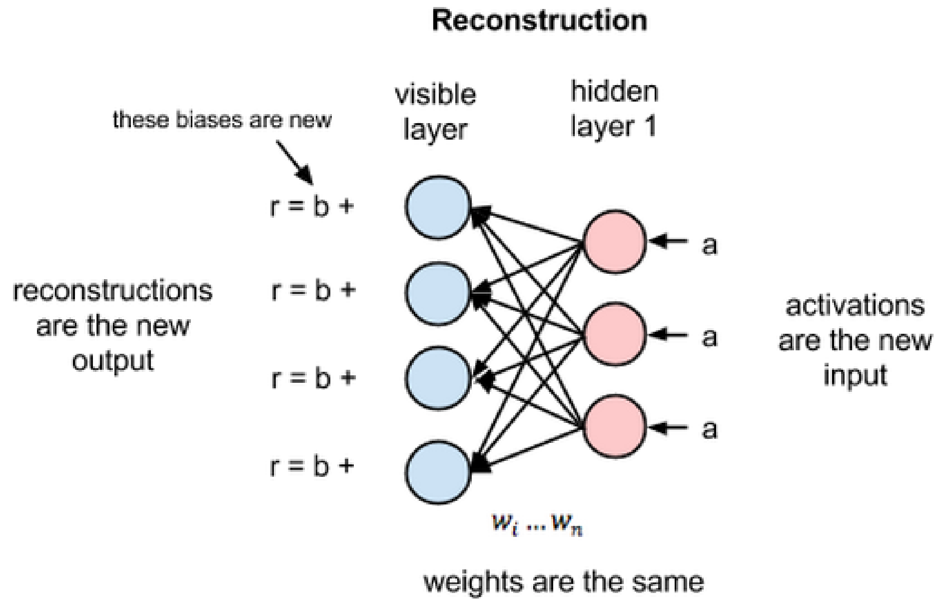
- Use Restricted Boltzmann Machine to pretrain the network
- Pretrain layer by layer and apply the softmax classification
- $\boldsymbol{CostFuction} = \|(\boldsymbol{Input\ x}) - (\boldsymbol{Recreation\ of\ x})\|$



Restricted Boltzmann Machine



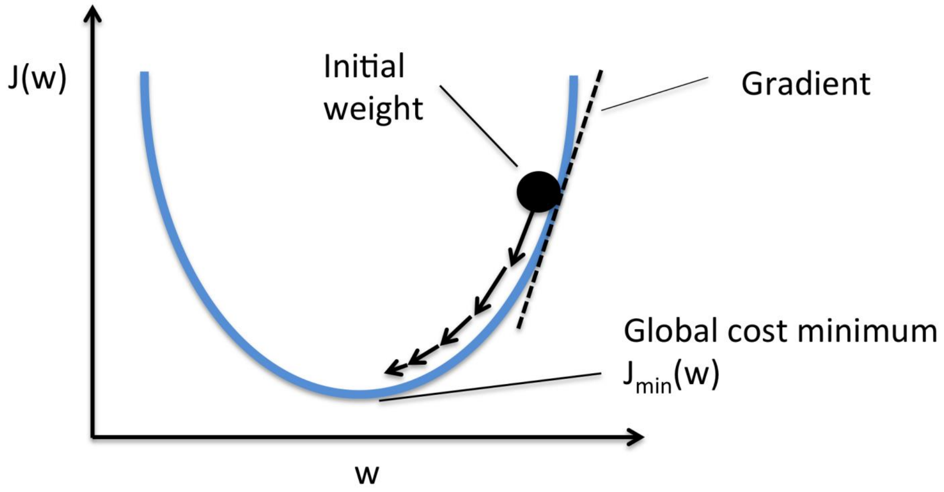
Forward pass

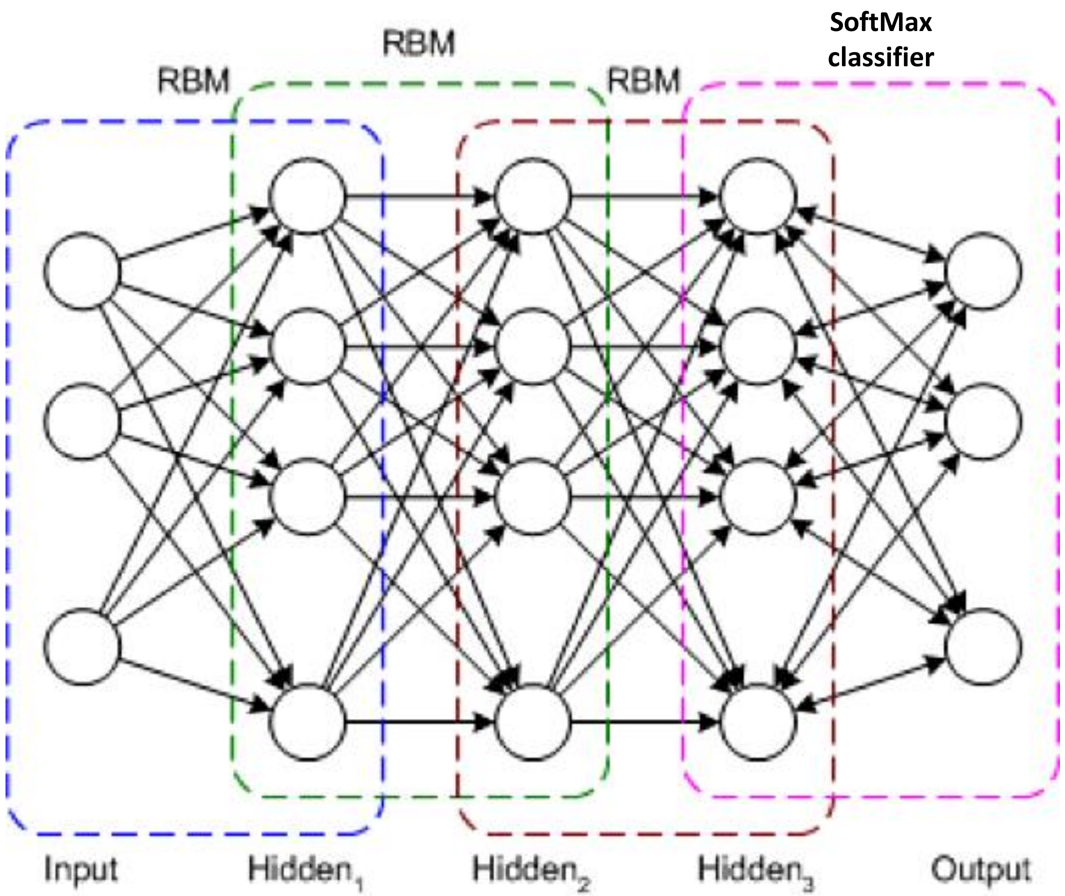


Backward pass

Deep Belief Network

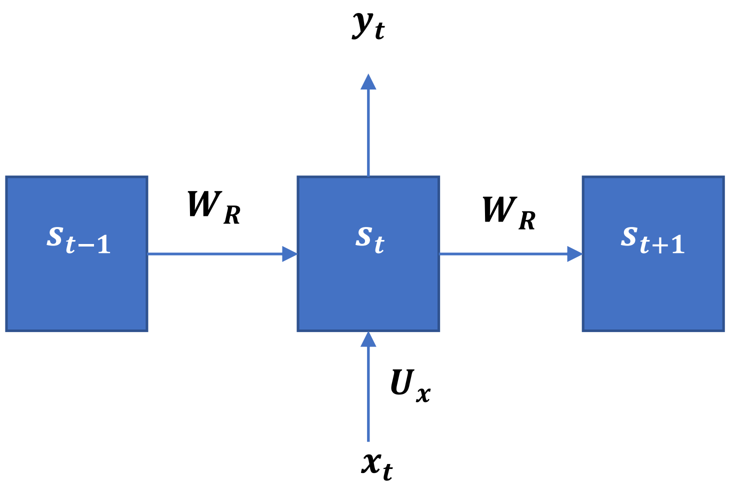
- Use RBM to pretrain the network
- Then it will initialize the weights closer to the global optimal position
- Now since the initializing weights are at a closer place to the optimal position, network will rapidly converge to its cost minimum position





- RNN Introduction
  - Variants on Recurrent nets
  - Training RNN
- Vanishing Gradients and Exploding Gradients
- Proposed Solutions
- Deep Belief Networks
- **Memory Systems**
  - Gated Recurrent Unit (GRU)
  - Long-Shot Term Memory (LSTM)
- Different Types of Recurrent Memory Models
  - Bi-directional RNN
  - Neural Turing Machine (NTM)
  - Hopfield Neural Networks (HNN)

Memory systems

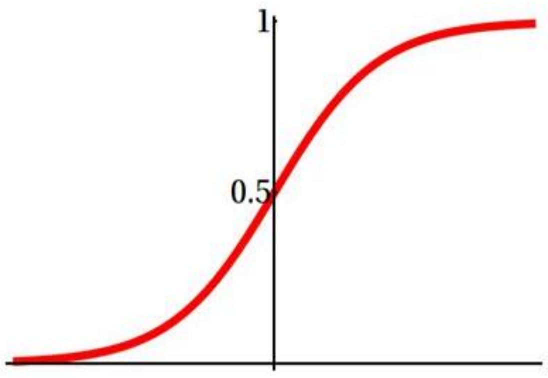


$$s_t = \varphi(W_R s_{t-1} + U_x x_t + b)$$

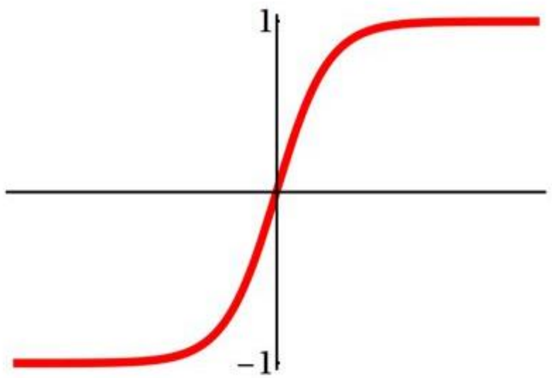
Nonlinearity is bad for long term memory
No selectivity (read all, overwrite all)

The RNN memory (state) should be protected only by + or – operations (write to the memory)

Be selective on choosing, what to read, write and forget



sigmoid



tanh

‘Write’ operation using addition

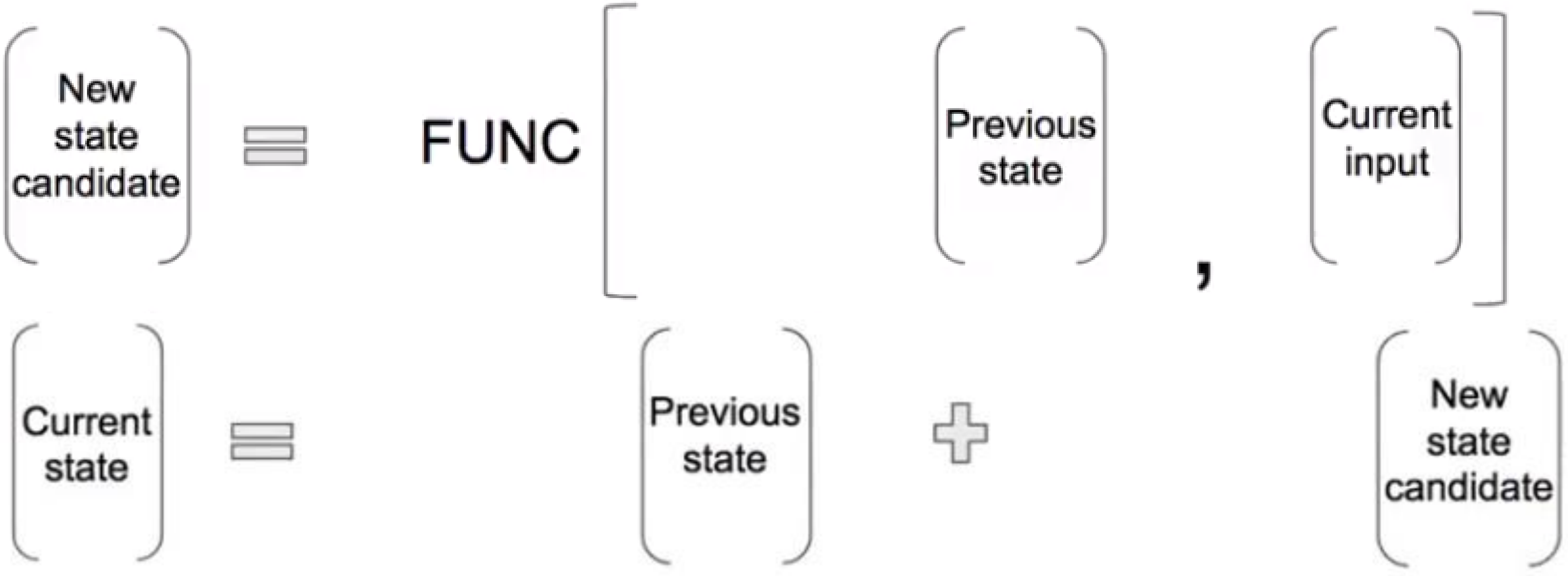
New state candidate is a vector of **positive** numbers

New state candidate is a vector of **negative** numbers

- Both cases, we don’t corrupt the whole memory
- Just add or subtract something

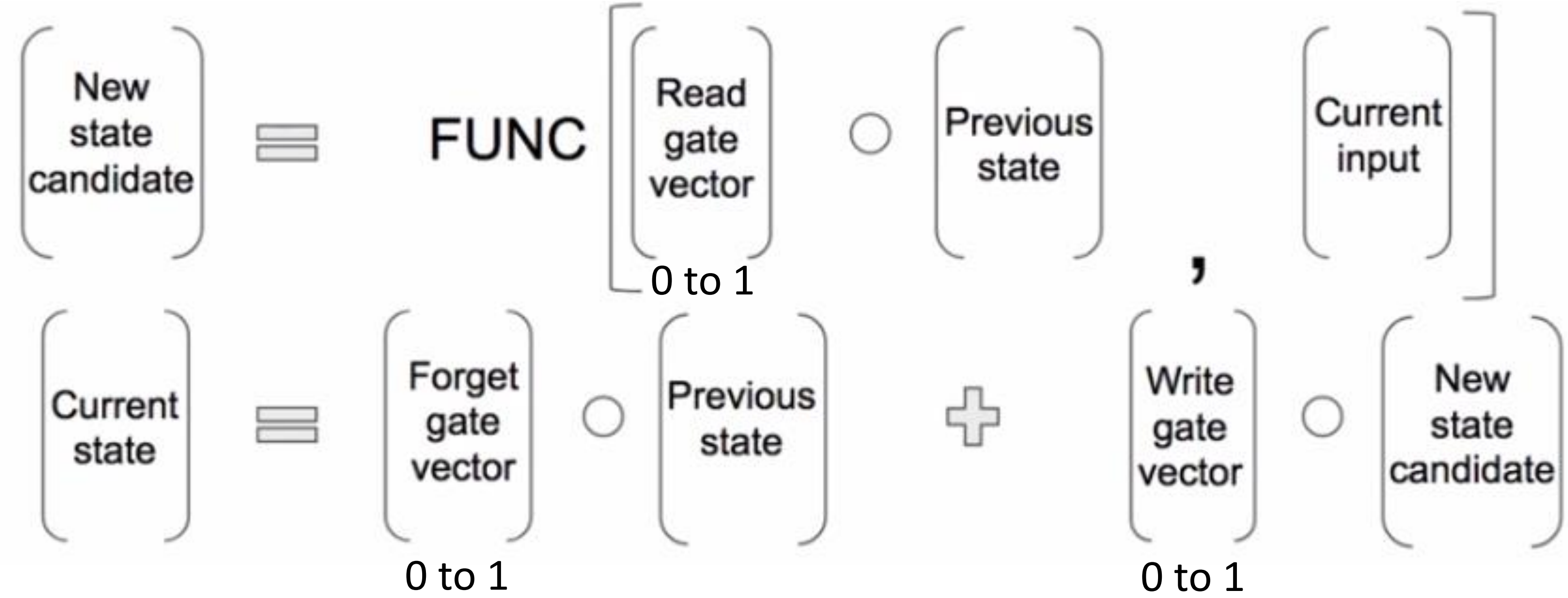
(Current state) = (Previous state) + (New state candidate)

Calculating New State Candidate



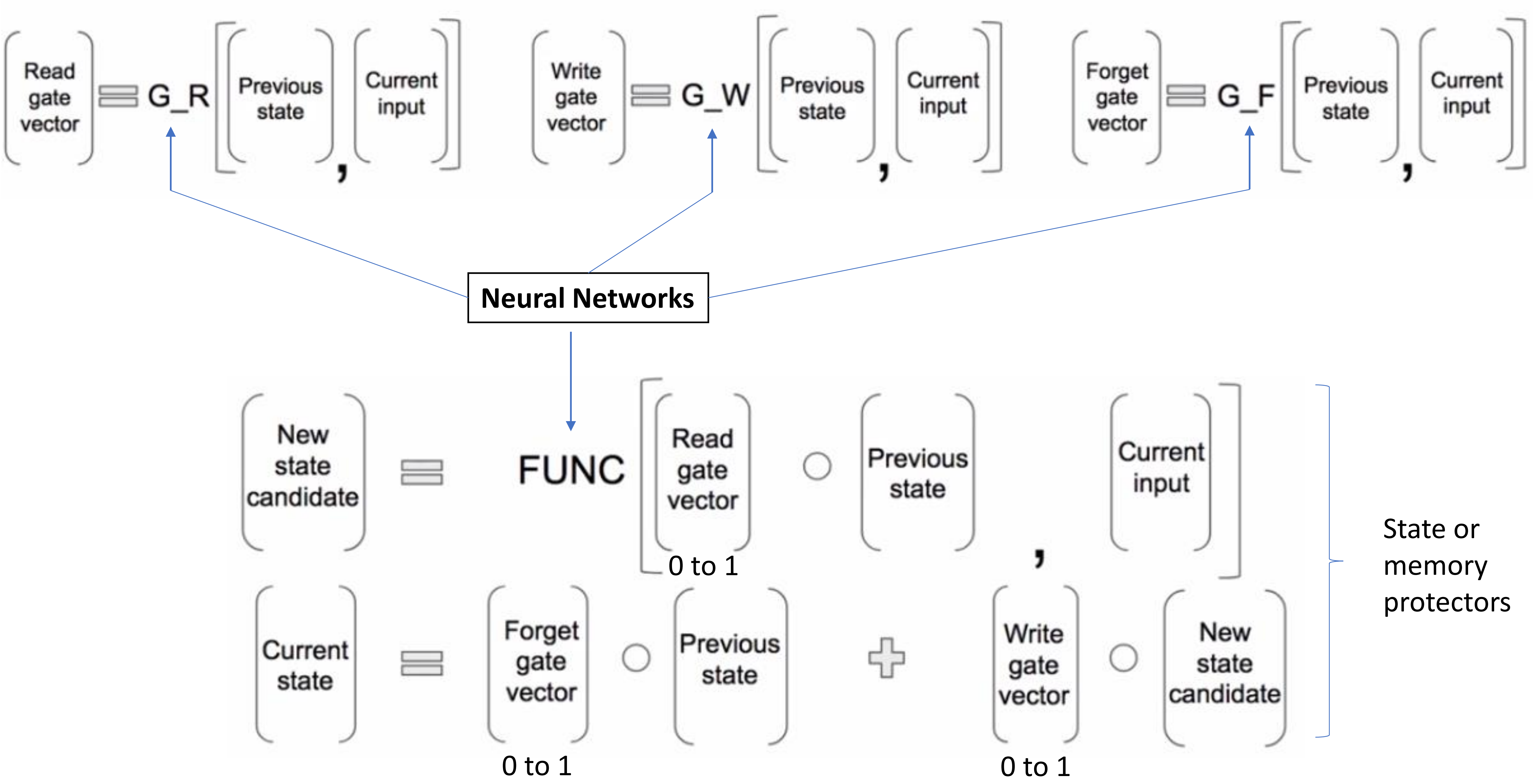
State or memory protectors

Calculating New State Candidate with selectivity

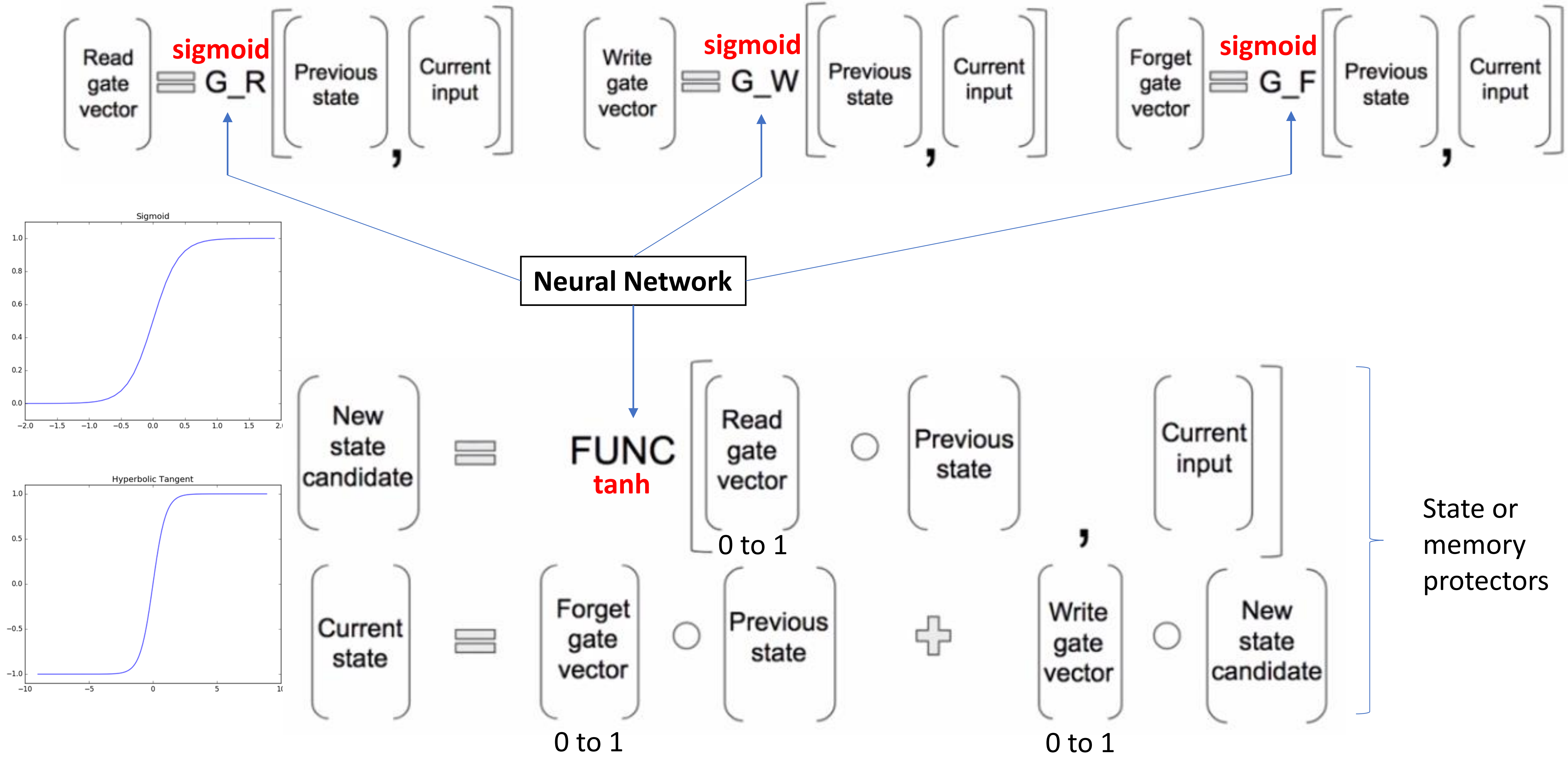


State or memory protectors

Selecting through gates



Selecting through gates

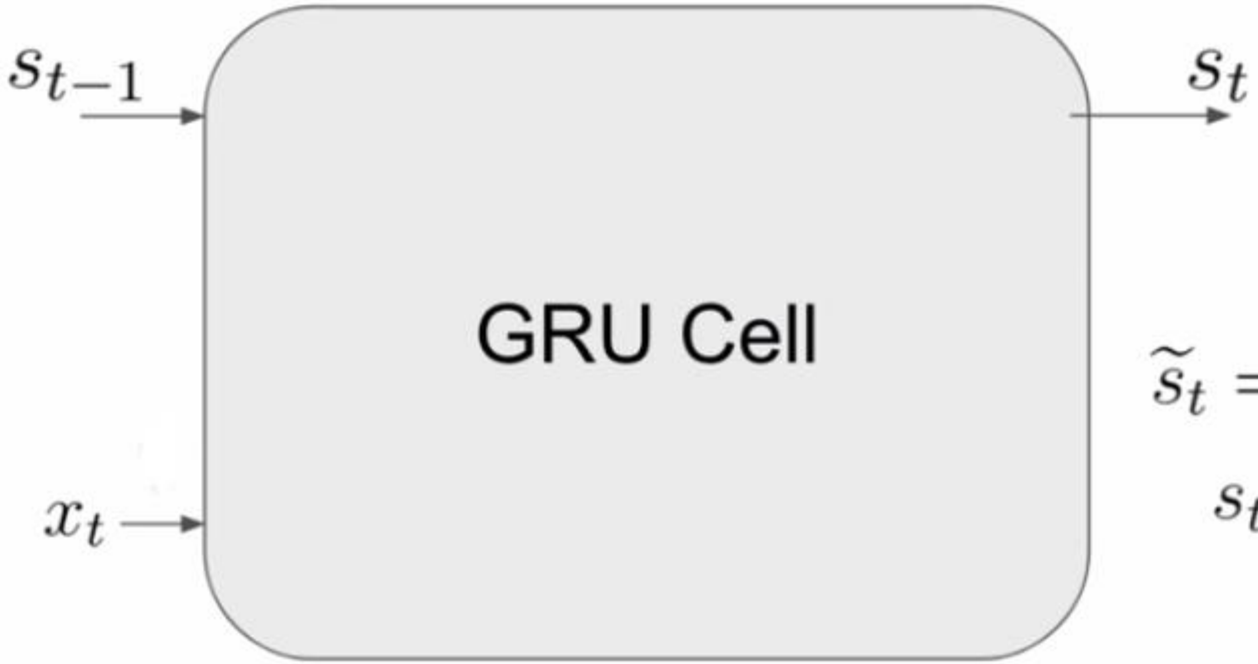


Reasons for activation functions

- Every gate use **sigmoid** activations in their neural networks
  - Because, gate must be able to control the flow of an incoming stream(memory from previous states)
  - If gate value is 1, then network accept everything which comes from next state
  - If gate value is 0, network will ignore every thing
- Neural network, which predicts the new state candidate uses a **tanh** activation
  - The new state candidate vector is used in the write operation, therefor both positive and negative values need to be there in the vector
  - If sigmoid is used instead of tanh, then all values will be positive and over flow can be occur

- RNN Introduction
  - Variants on Recurrent nets
  - Training RNN
- Vanishing Gradients and Exploding Gradients
- Proposed Solutions
- Deep Belief Networks
- Memory Systems
  - **Gated Recurrent Unit (GRU)**
  - Long-Shot Term Memory (LSTM)
- Different Types of Recurrent Memory Models
  - Bi-directional RNN
  - Neural Turing Machine (NTM)
  - Hopfield Neural Networks (HNN)

Gated Recurrent Unit (GRU)



GRU equations

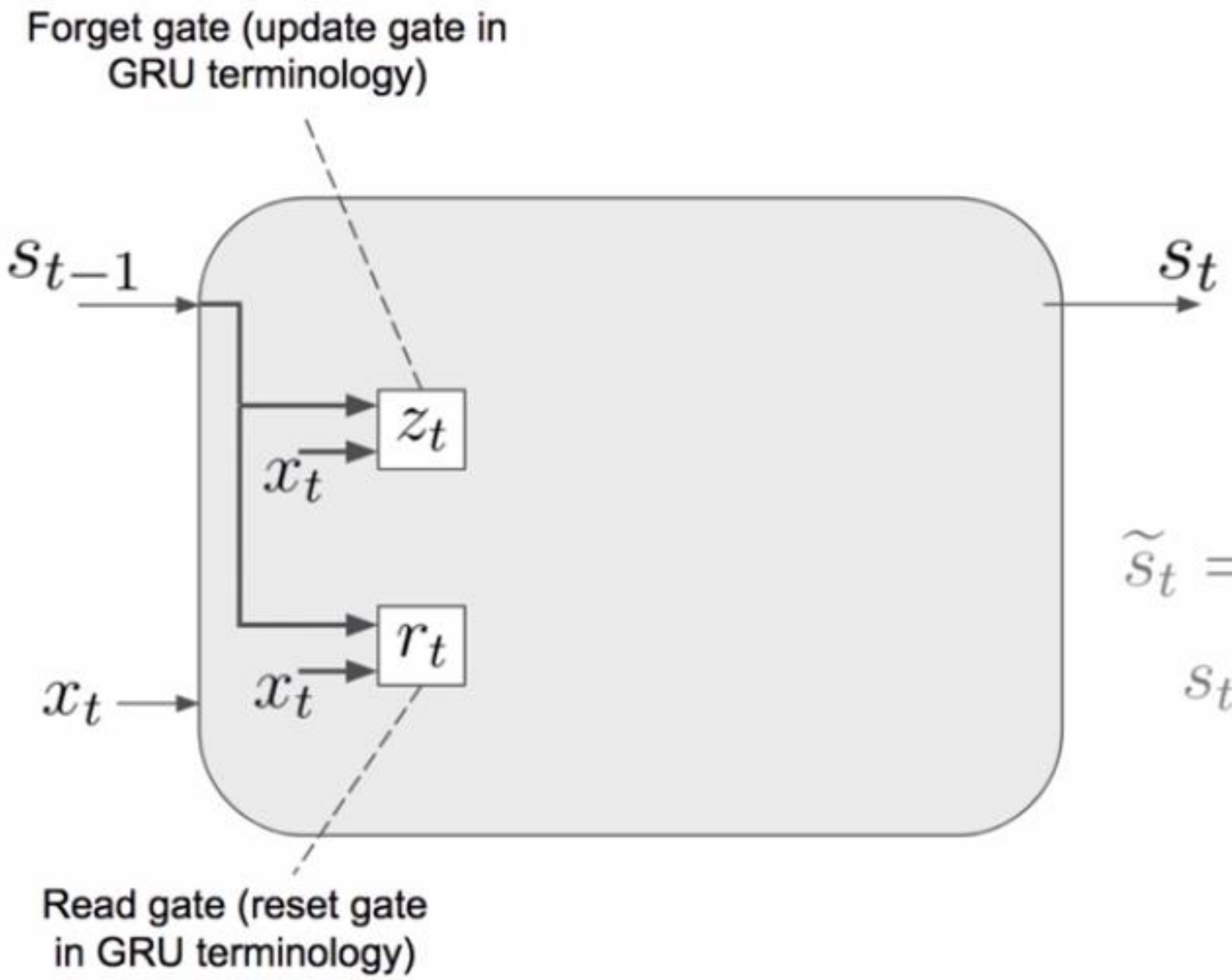
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r) \quad \text{Read gate}$$

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z) \quad \text{Forget gate}$$

$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b) \quad \text{State candidate}$$

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t \quad \text{Current State}$$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

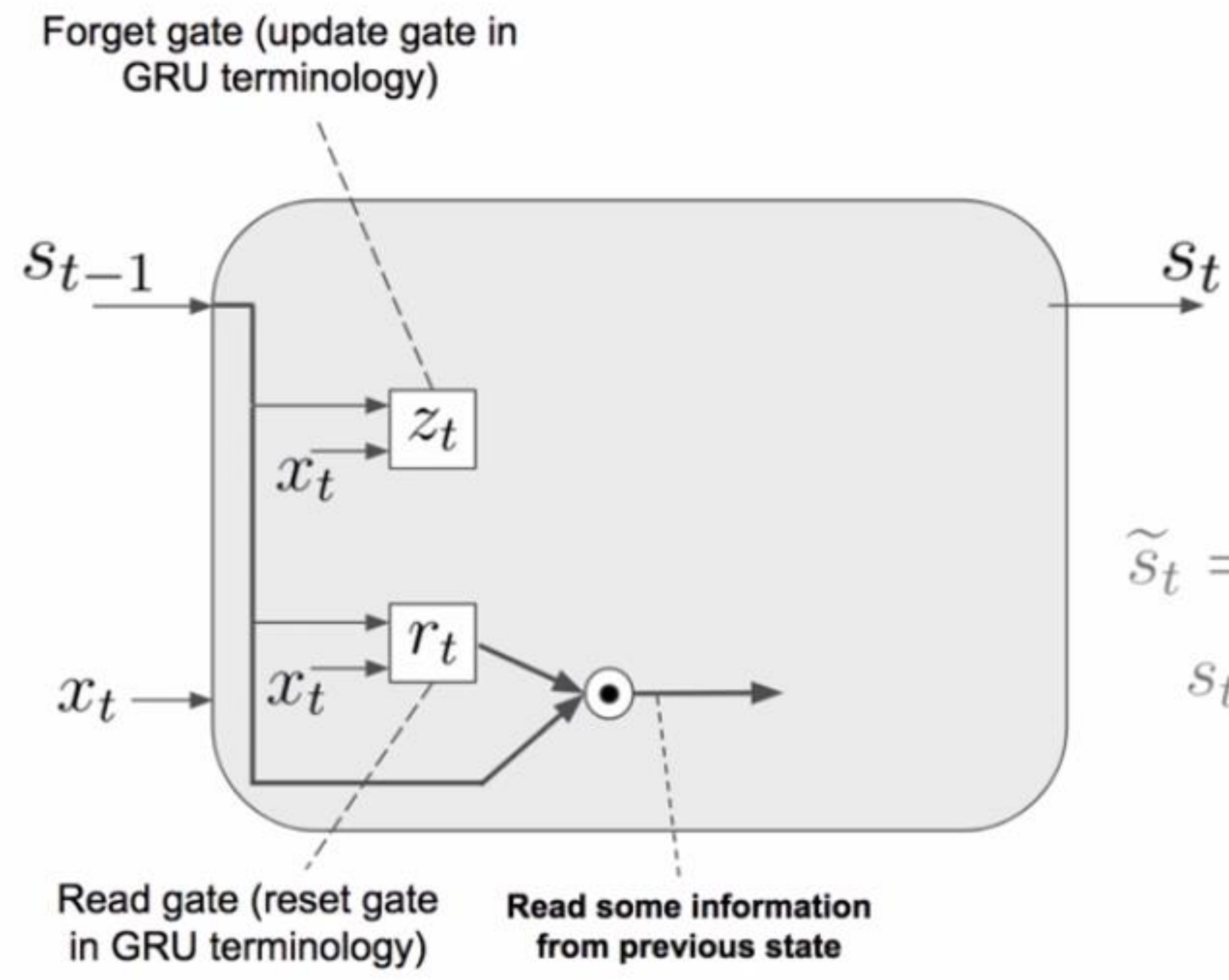
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t$$ Current State

$\sigma = sigmoid\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

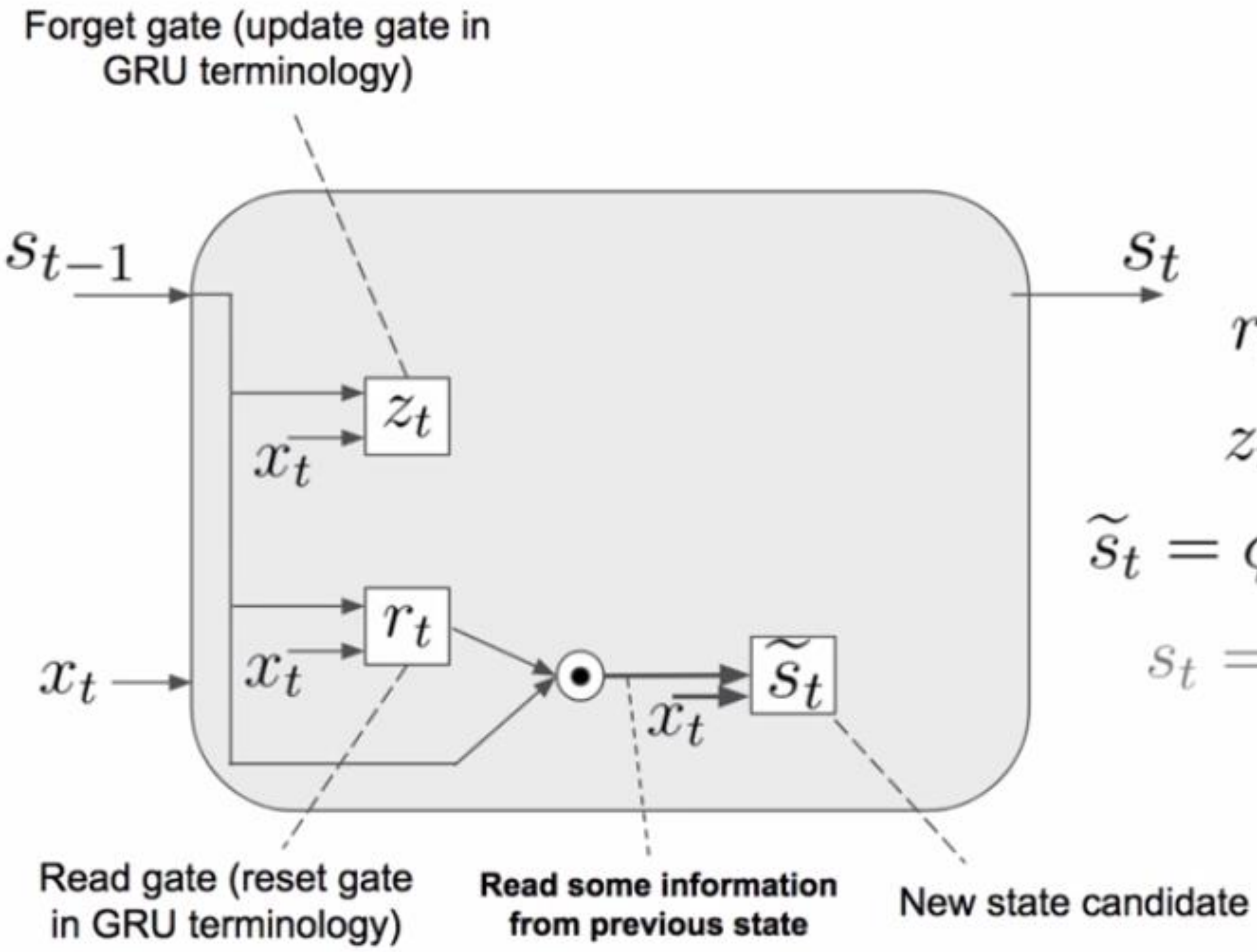
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t$$ Current State

$\sigma = sigmoid\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r) \quad \text{Read gate}$$

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z) \quad \text{Forget gate}$$

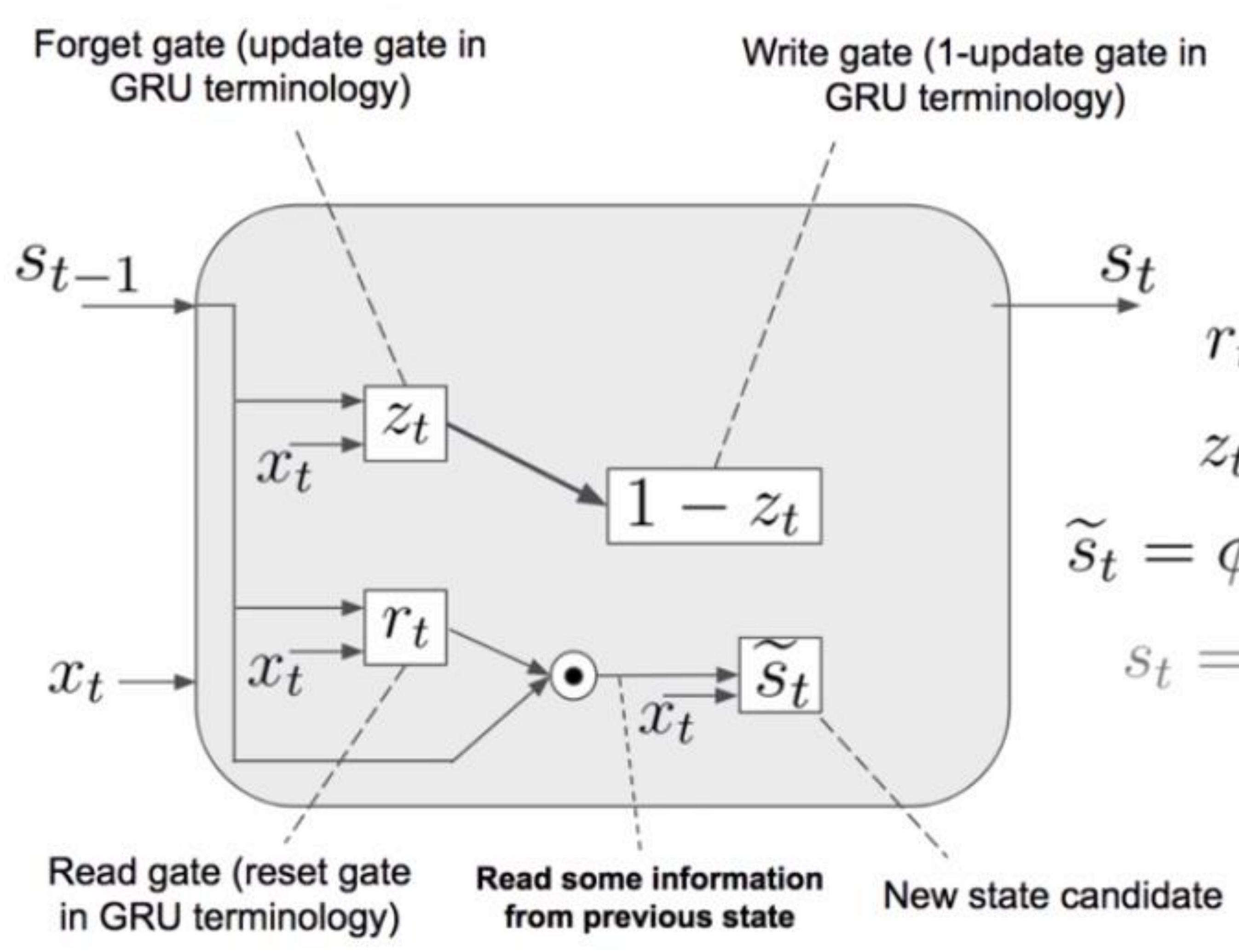
$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b) \quad \text{State candidate}$$

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t \quad \text{Current State}$$

$\sigma = sigmoid\ activation\ function$

$\emptyset = tanh\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r) \quad \text{Read gate}$$

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z) \quad \text{Forget gate}$$

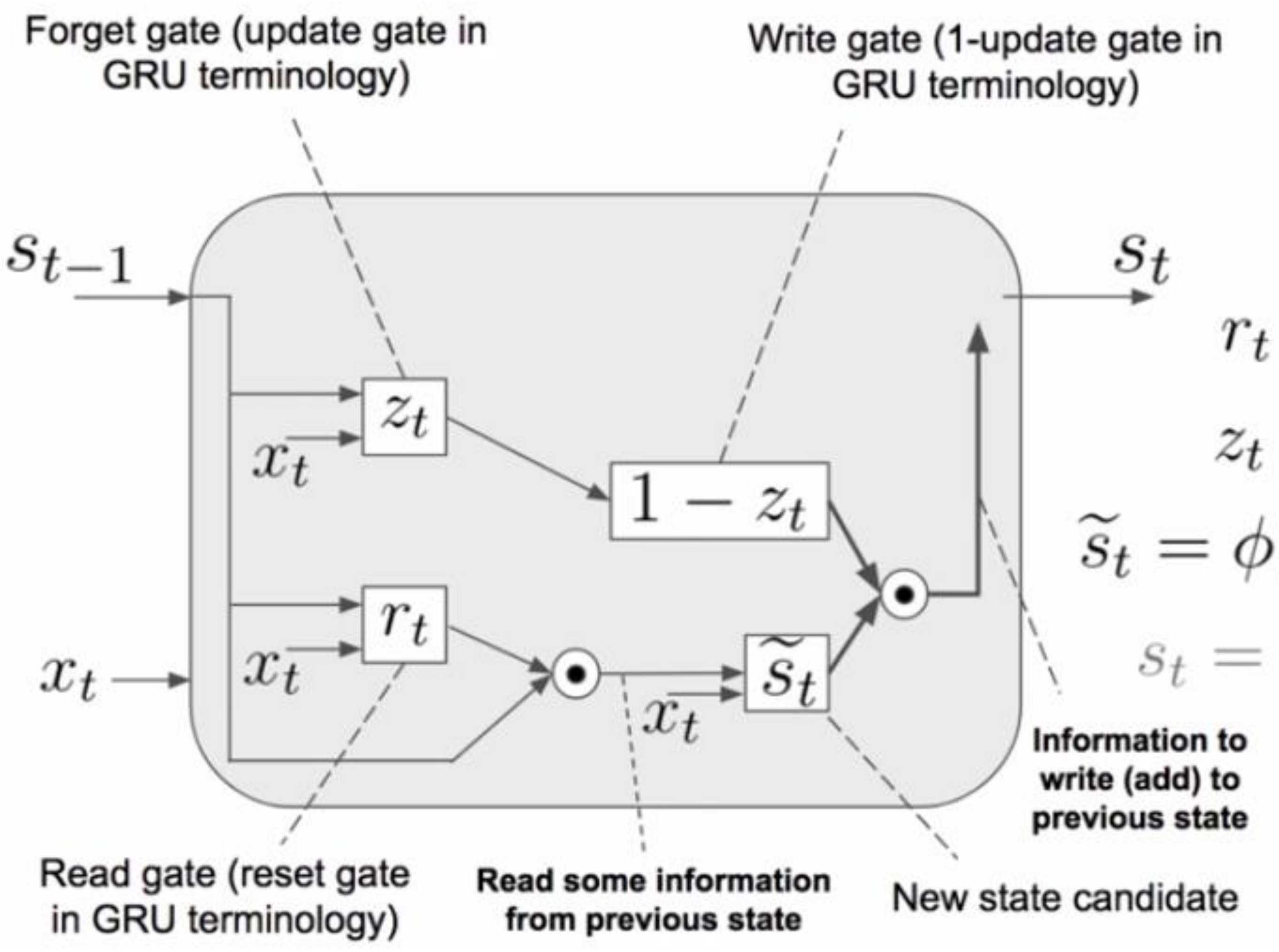
$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b) \quad \text{State candidate}$$

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t \quad \text{Current State}$$

$\sigma = sigmoid\ activation\ function$

$\emptyset = tanh\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

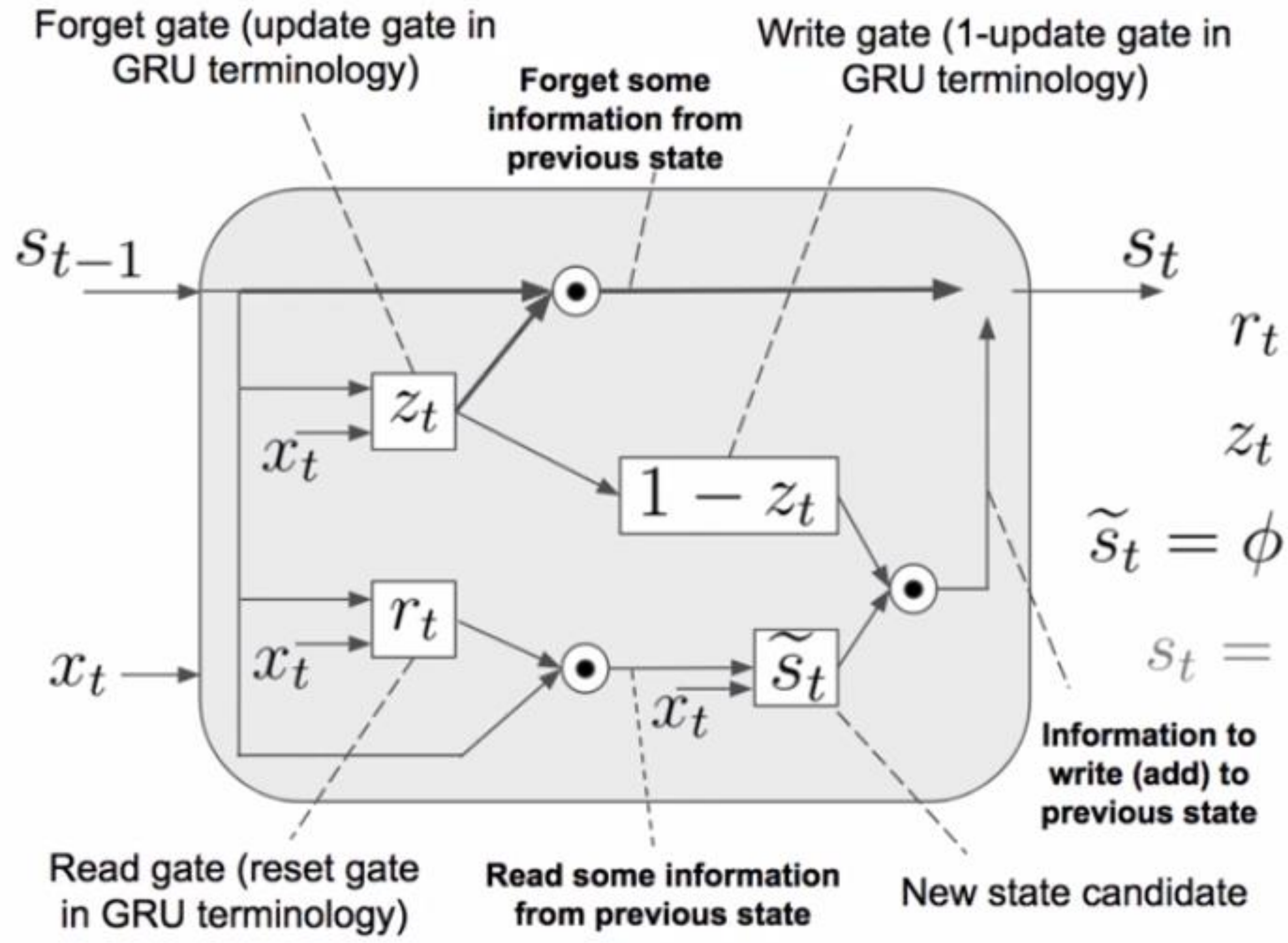
$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t$$ Current State

$\sigma = sigmoid\ activation\ function$

$\emptyset = tanh\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

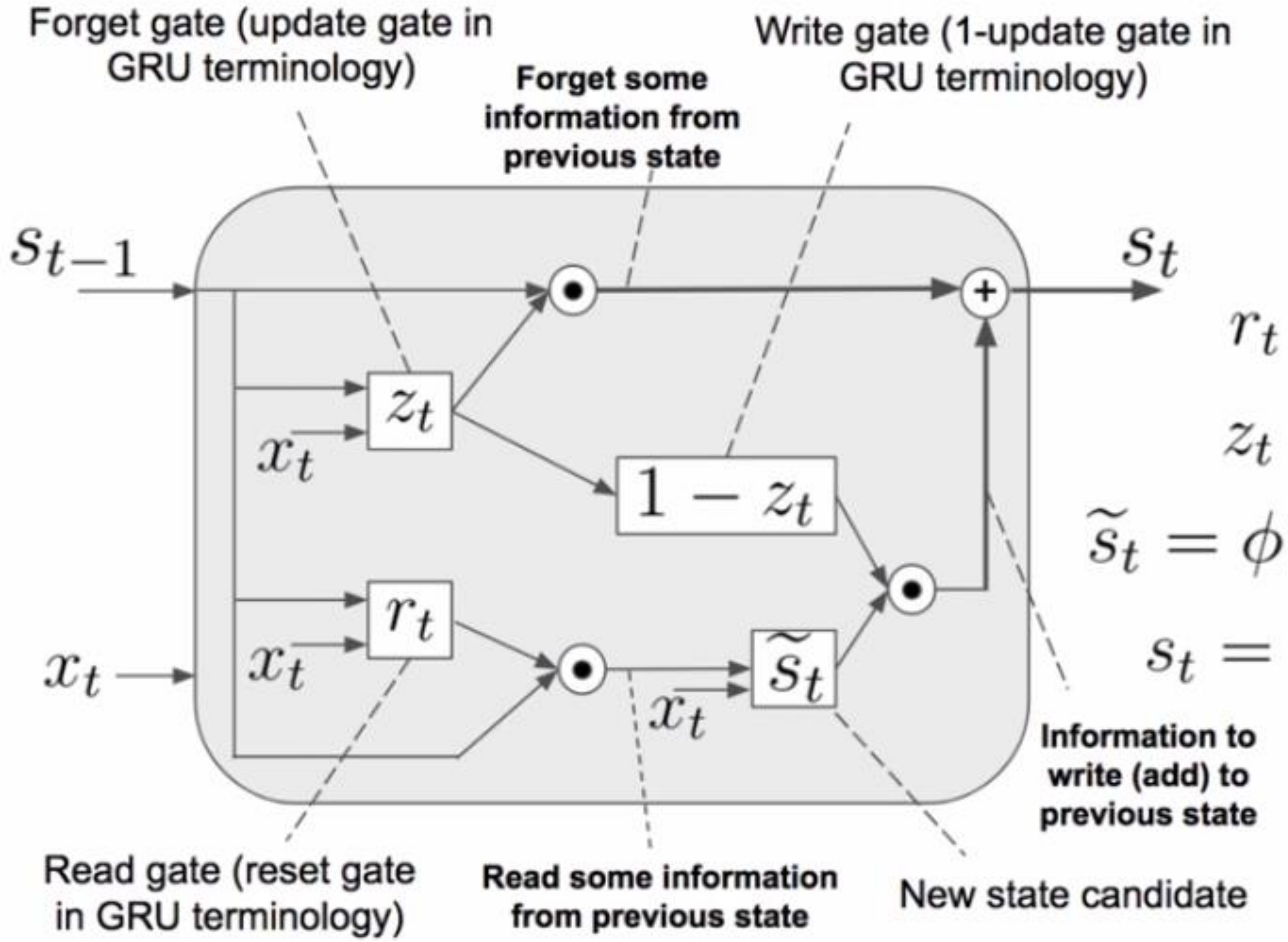
$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t$$ Current State

$\sigma = sigmoid\ activation\ function$

$\emptyset = tanh\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r) \quad \text{Read gate}$$

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z) \quad \text{Forget gate}$$

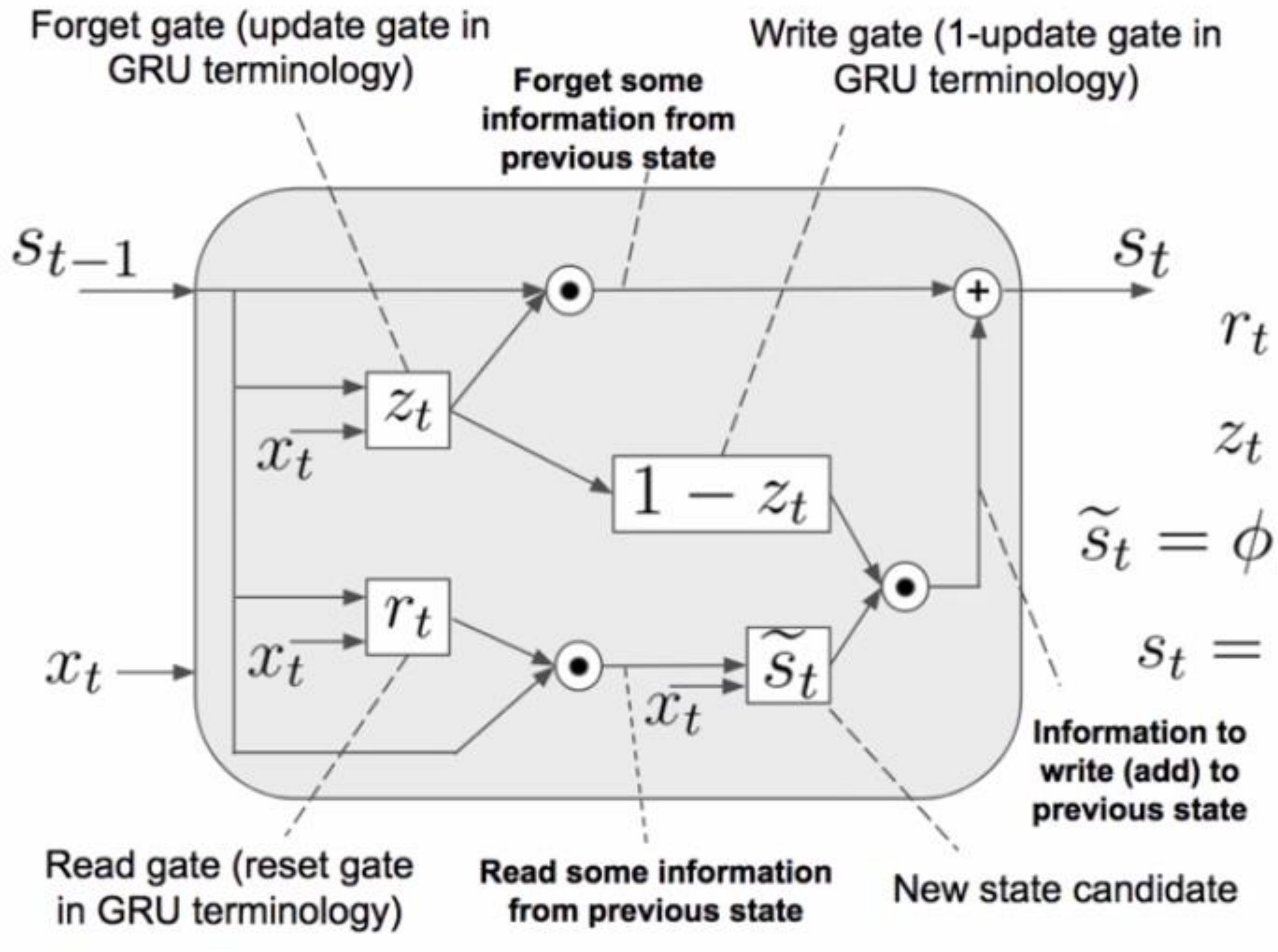
$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b) \quad \text{State candidate}$$

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t \quad \text{Current State}$$

$\sigma = sigmoid\ activation\ function$

$\emptyset = tanh\ activation\ function$

Gated Recurrent Unit (GRU)



GRU equations	In GRU terminology
$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$	Reset gate
$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$	Update gate
$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$	State candidate
$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \widetilde{s}_t$	Current State

$\sigma = sigmoid\ activation\ function$

$\emptyset = tanh\ activation\ function$

Gated Recurrent Unit (GRU)

GRU equations	In GRU terminology
$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$	Reset gate
$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$	Update gate
$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$	State candidate
$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \widetilde{s}_t$	Current State

Let's take a closer look at the way the gates operate

Gated Recurrent Unit (GRU)

GRU equations	In GRU terminology
$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$	Reset gate
$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$	Update gate
$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$	State candidate
$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \widetilde{s}_t$	Current State

$$\begin{pmatrix} \widetilde{s}_t^1 \\ \widetilde{s}_t^2 \\ \widetilde{s}_t^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ s_{t-1}^3 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \circ \begin{pmatrix} \widetilde{s}_t^1 \\ \widetilde{s}_t^2 \\ \widetilde{s}_t^3 \end{pmatrix}$$

Update gate = vector of zeros

Replace all memory content with the State Candidate

Gated Recurrent Unit (GRU)

GRU equations	In GRU terminology
$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$	Reset gate
$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$	Update gate
$\widetilde{s}_t = \phi(W\,(r_t \circ s_{t-1}) + U x_t + b)$	State candidate
$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \widetilde{s}_t$	Current State

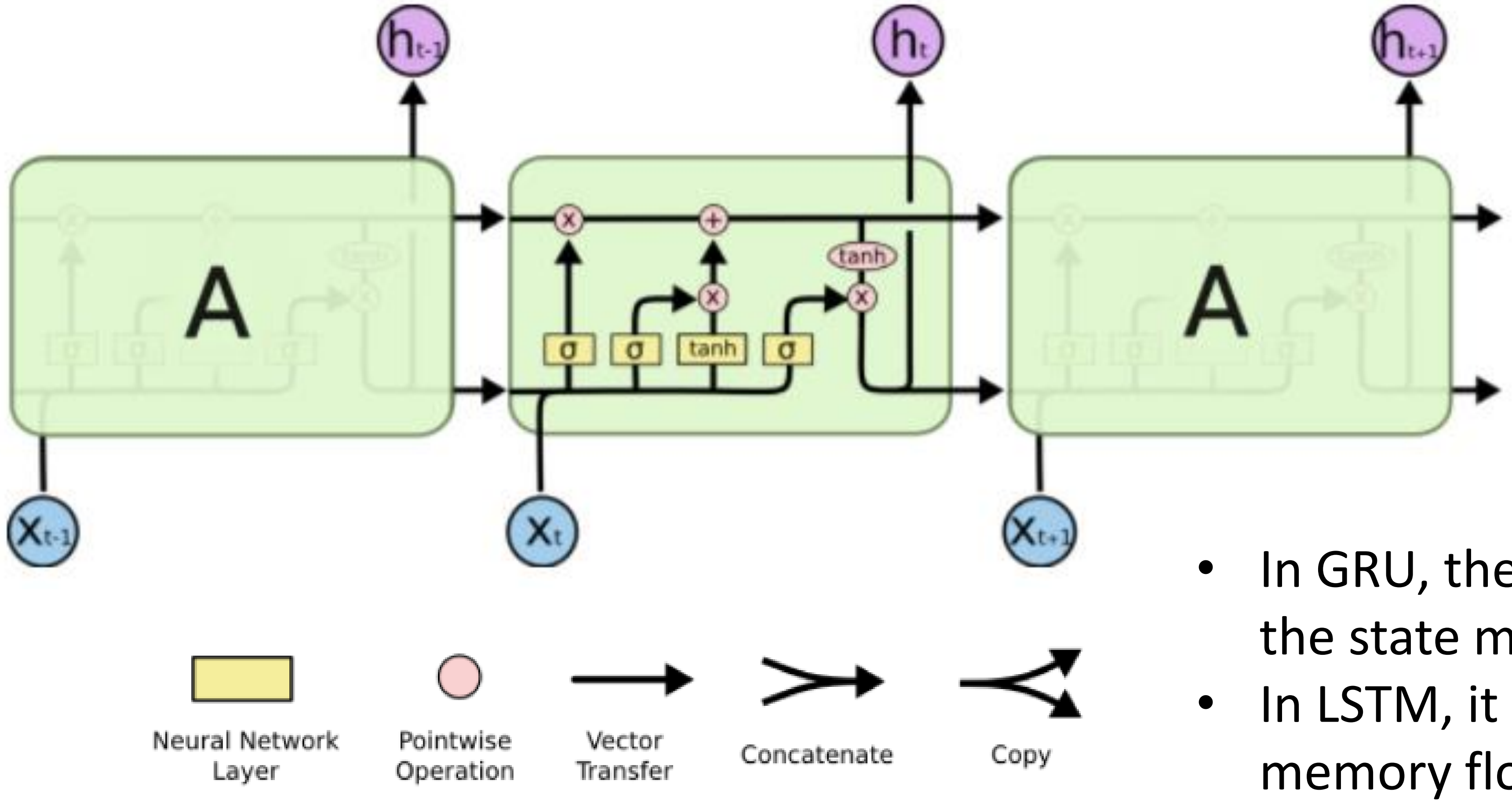
$$\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ s_{t-1}^3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \circ \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ s_{t-1}^3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} \widetilde{s}_t^1 \\ \widetilde{s}_t^2 \\ \widetilde{s}_t^3 \end{pmatrix}$$

Completely ignore current input and state candidate

Update gate = vector of ones ⇒ **The current state equals the previous state**

- RNN Introduction
    - Variants on Recurrent nets
    - Training RNN
- Vanishing Gradients and Exploding Gradients
- Proposed Solutions
- Deep Belief Networks
- Memory Systems
    - Gated Recurrent Unit (GRU)
    - **Long-Shot Term Memory (LSTM)**
- Different Types of Recurrent Memory Models
    - Bi-directional RNN
    - Neural Turing Machine (NTM)
    - Hopfield Neural Networks (HNN)

Long-Shot Term Memory (LSTM)



- In GRU, the hidden layer value and the state memory value are the same
- In LSTM, it has a separate state memory flow and a hidden layer value which is gated from the state memory

LSTM Cell

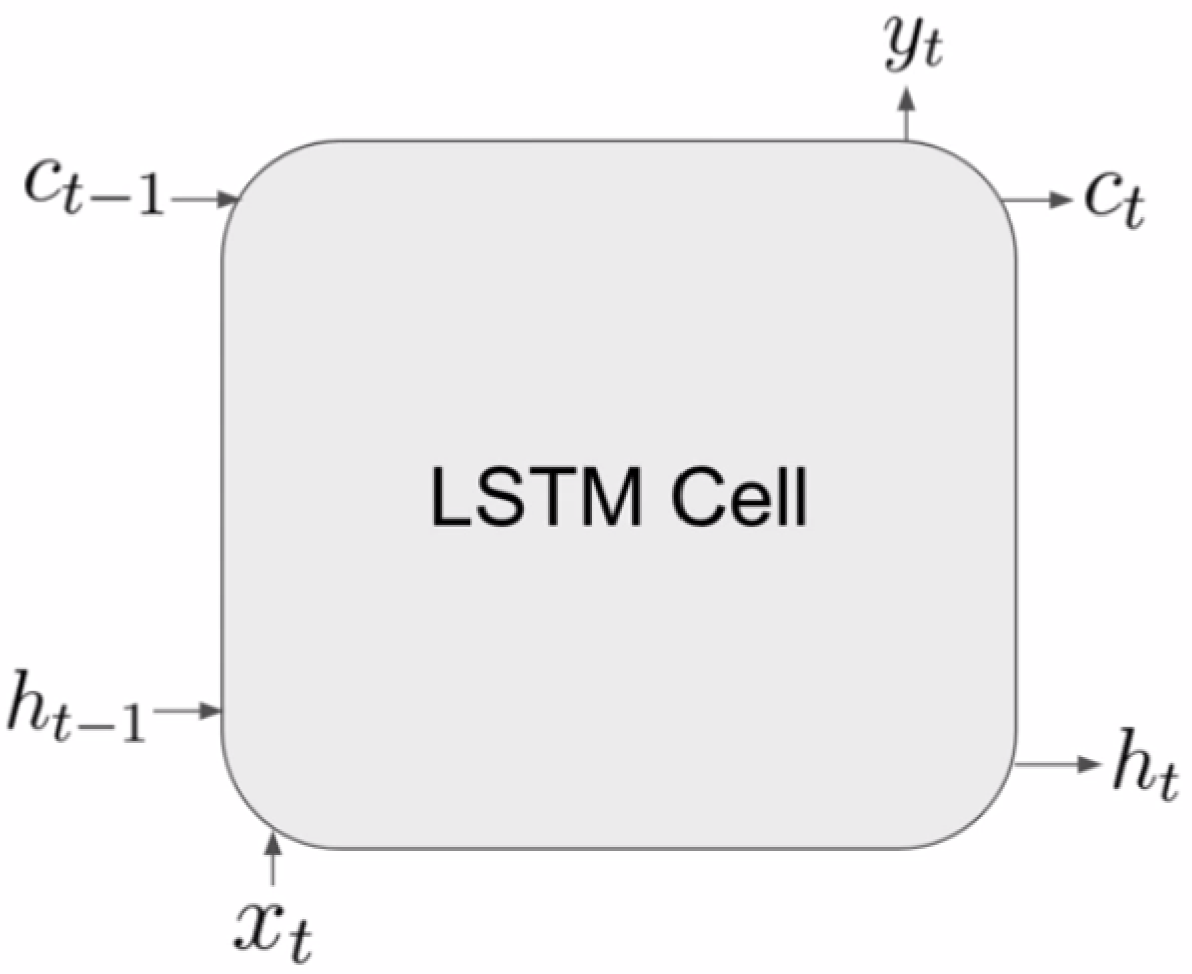
The state is a pair of vectors:

c_t — Memory Cell

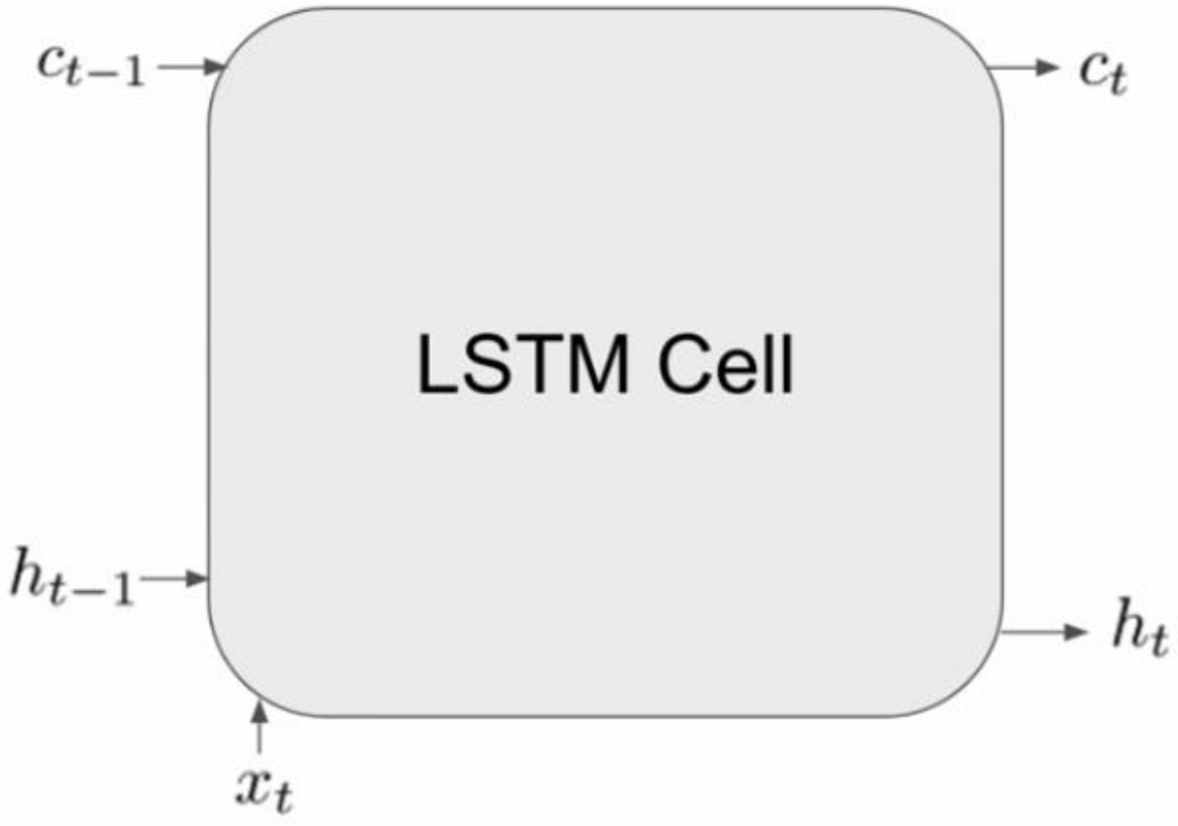
h_t — Shadow State (gated version of memory cell)

The output is a part of the state:

$y_t = h_t$ — Cell output



LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

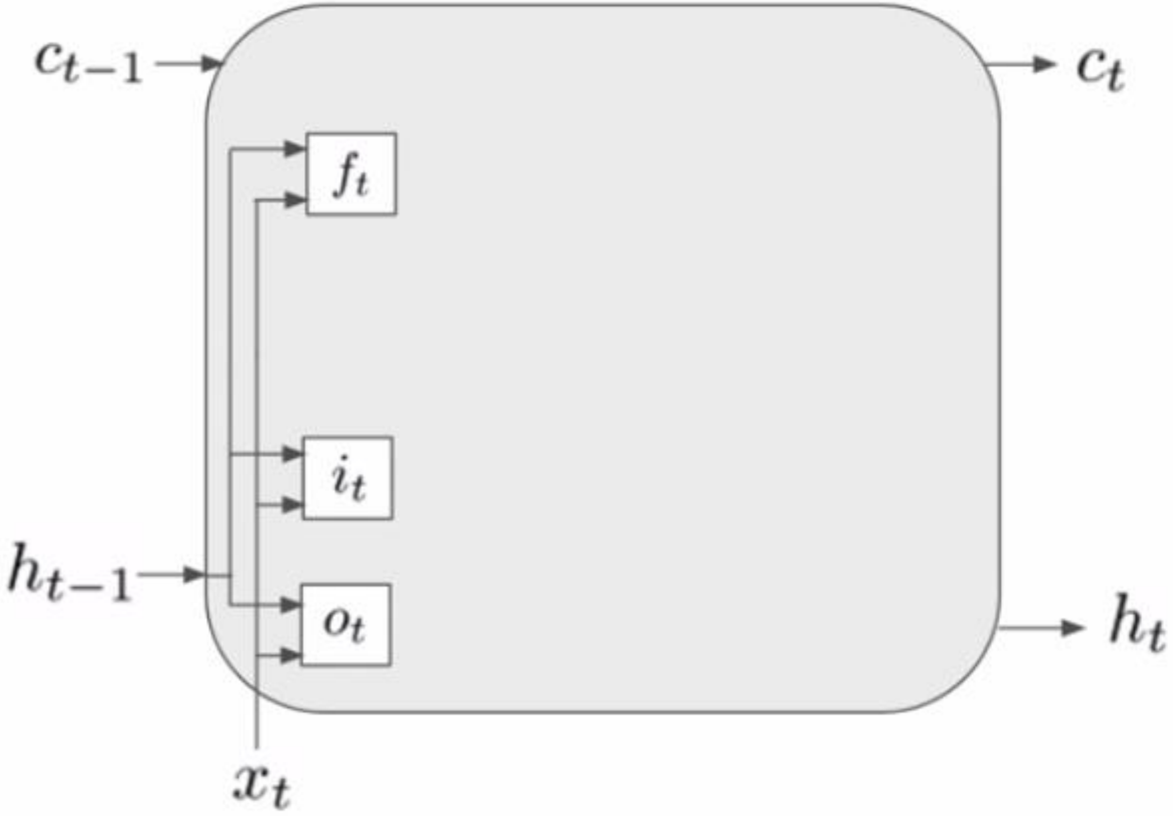
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

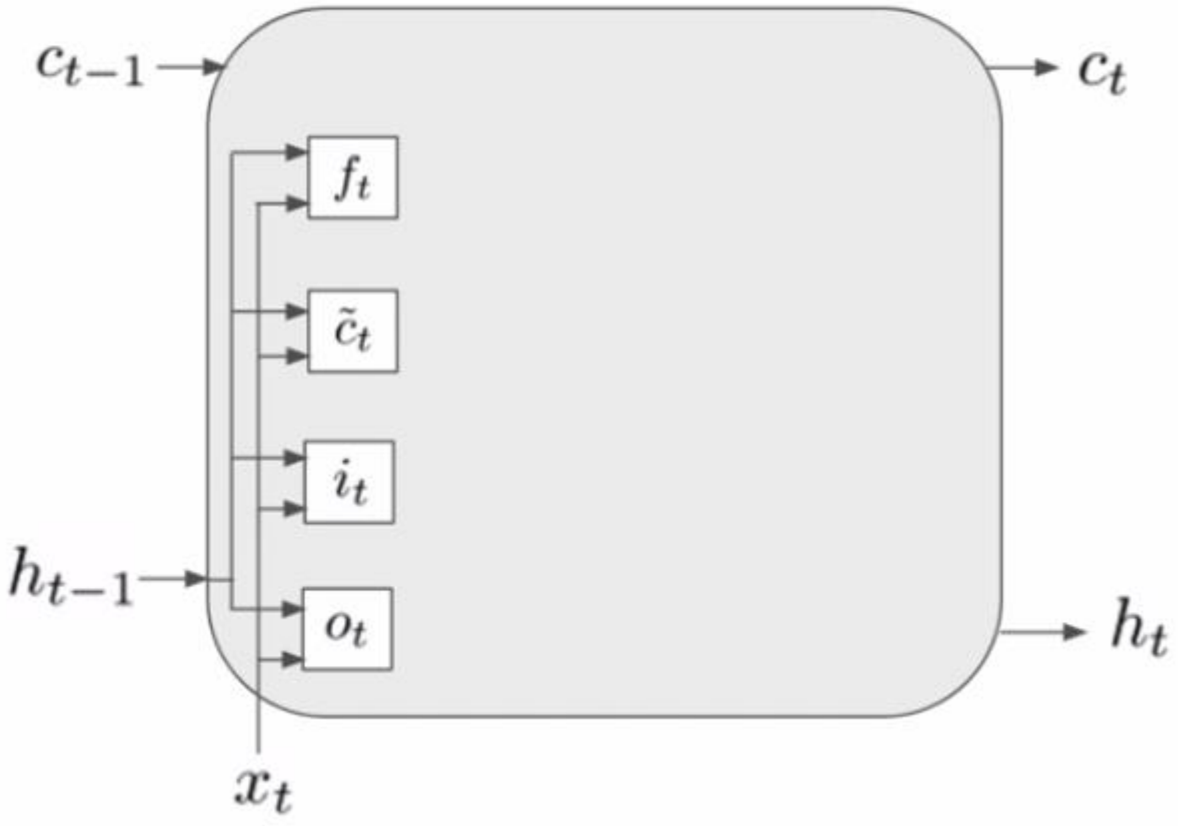
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

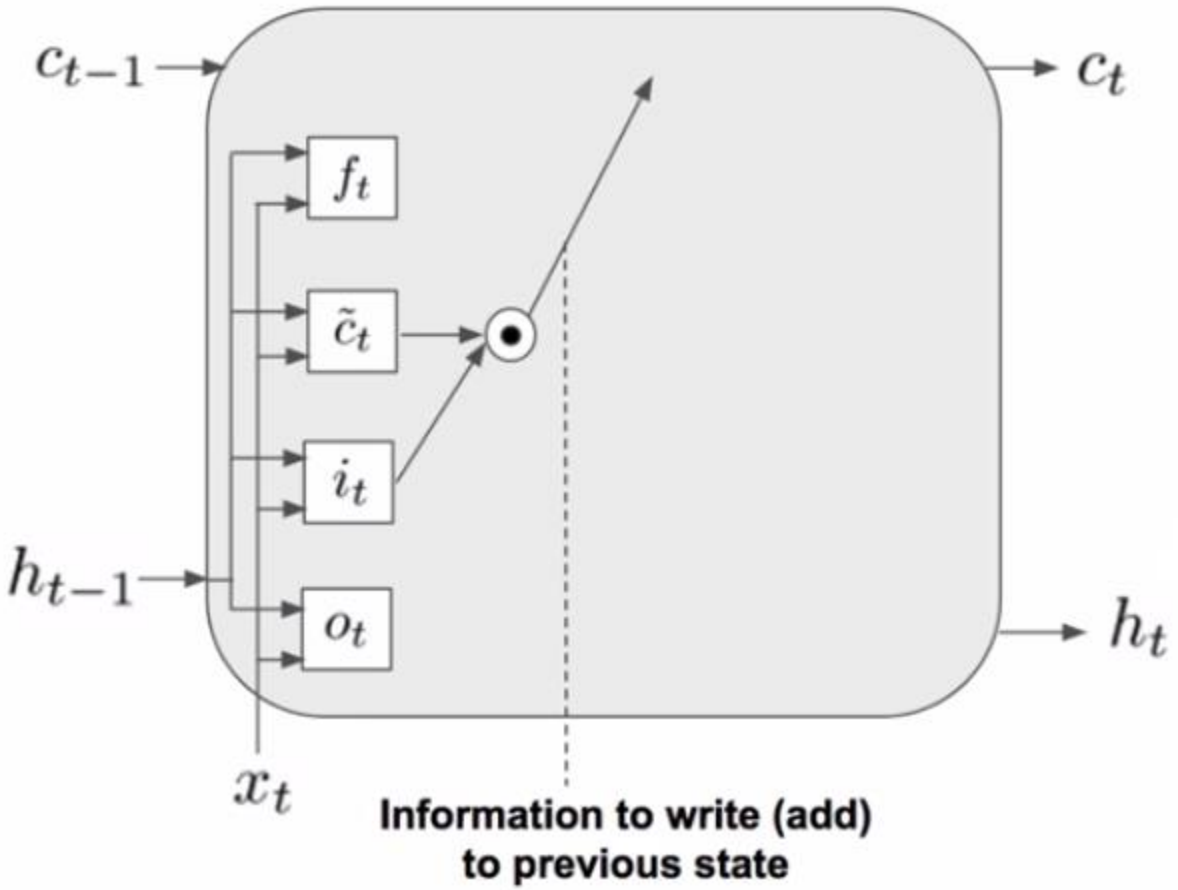
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

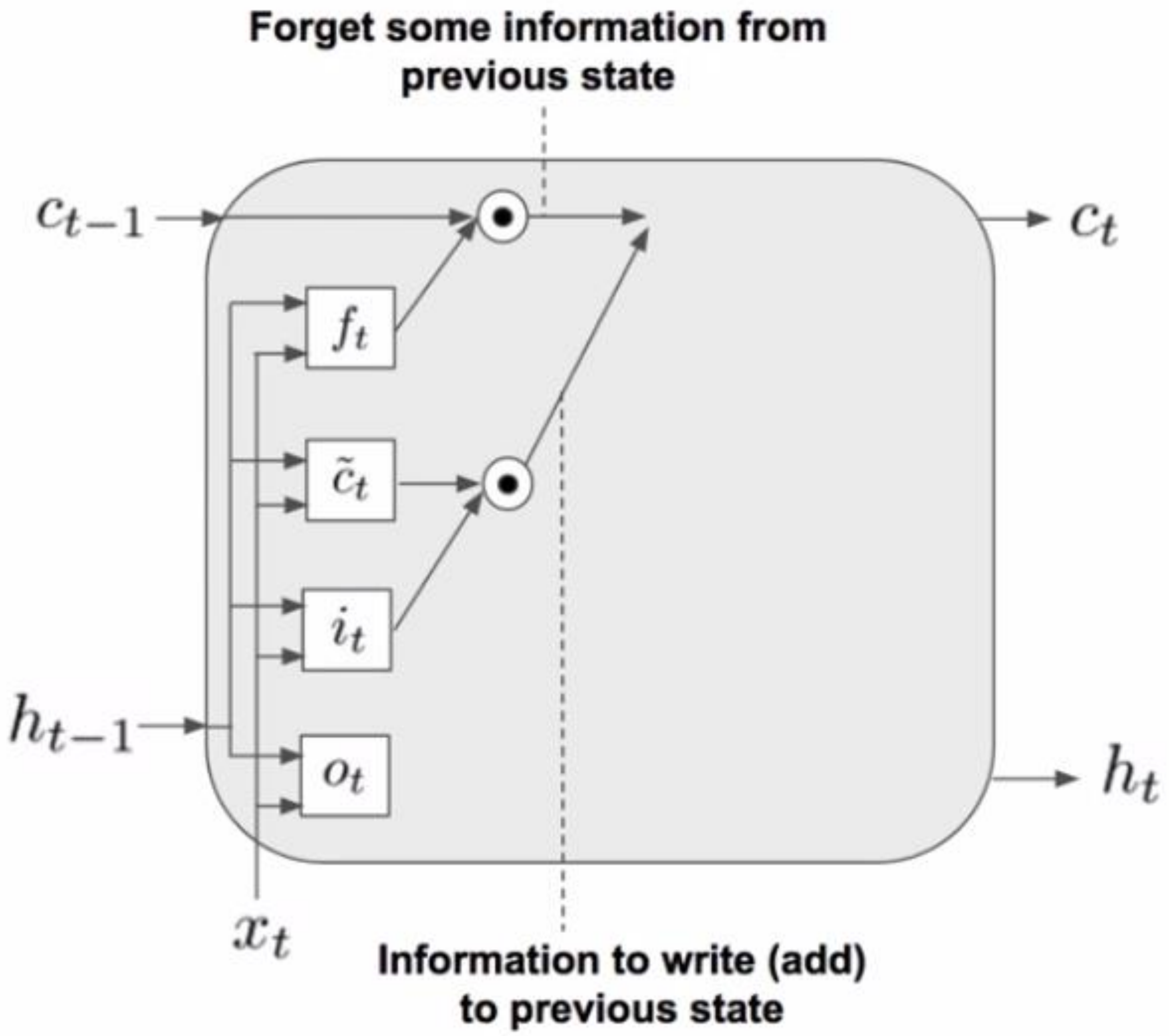
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

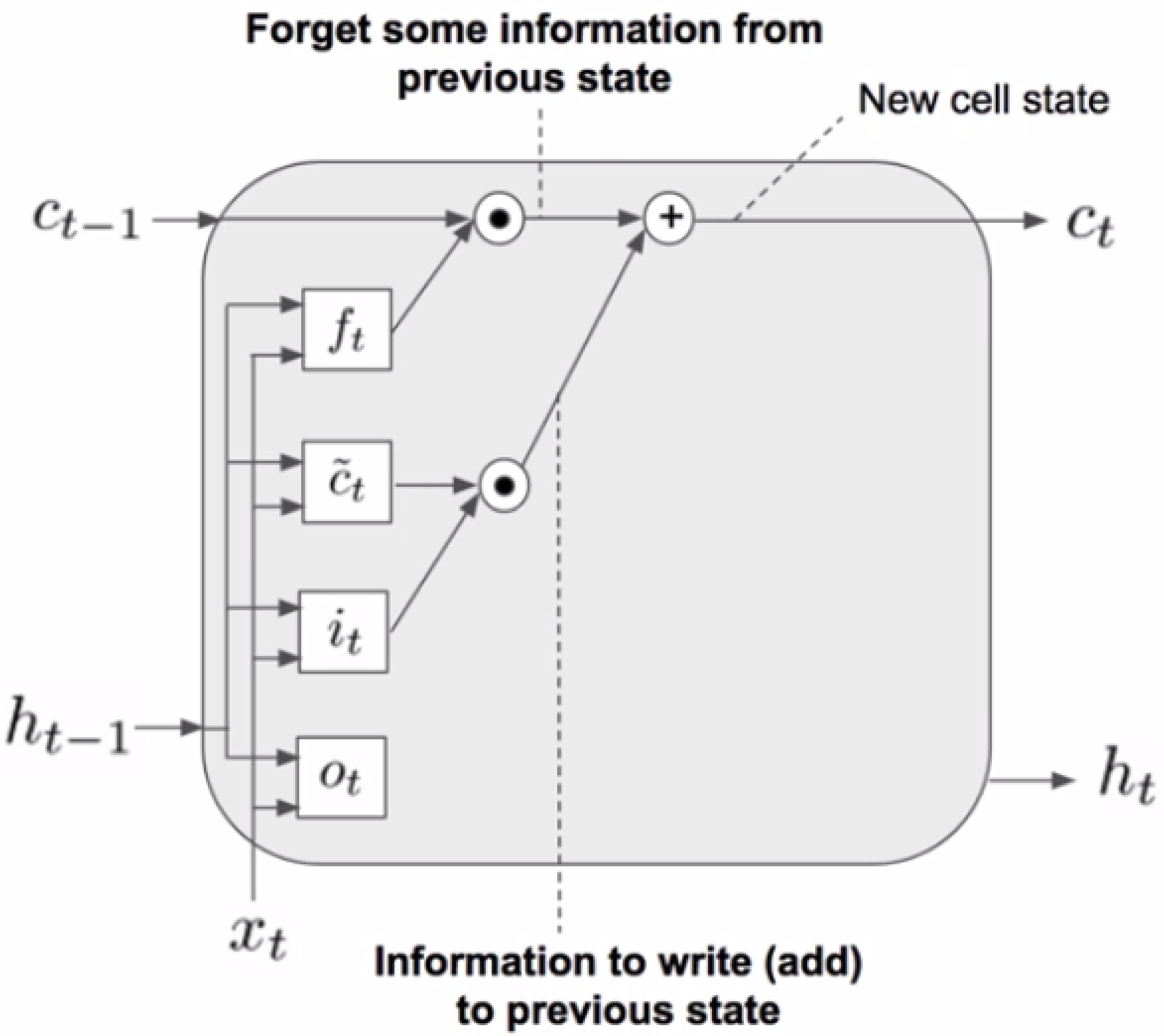
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

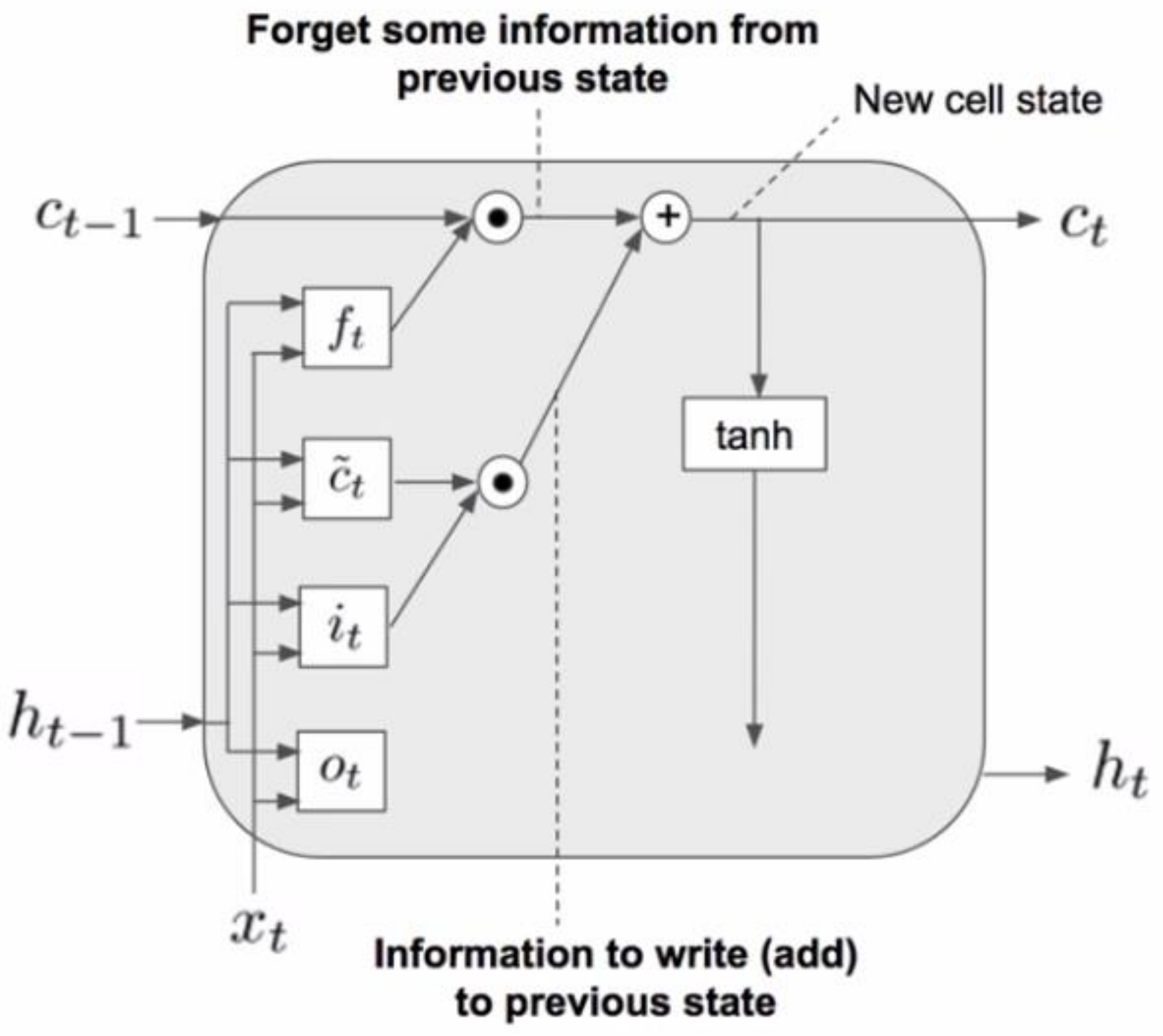
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

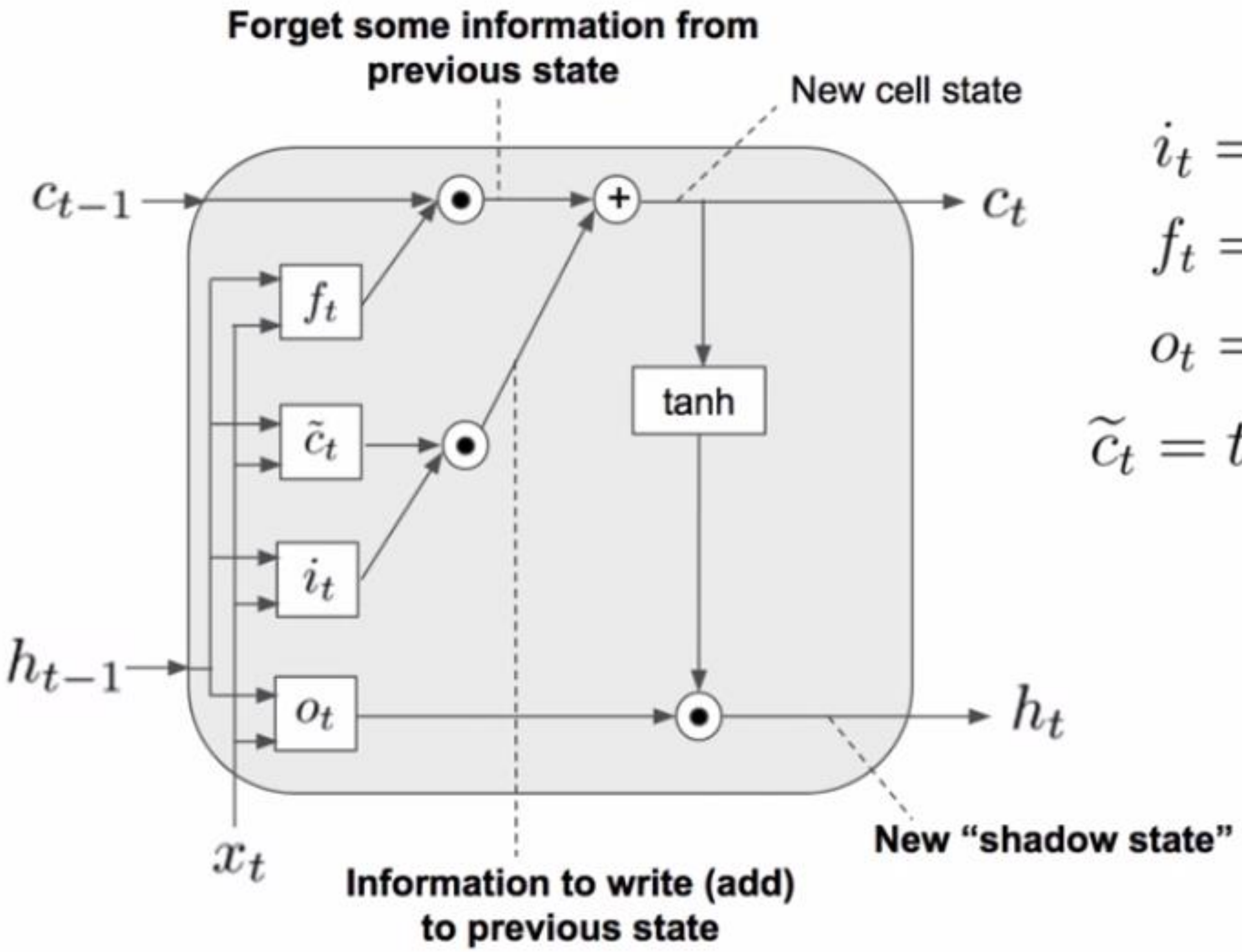
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

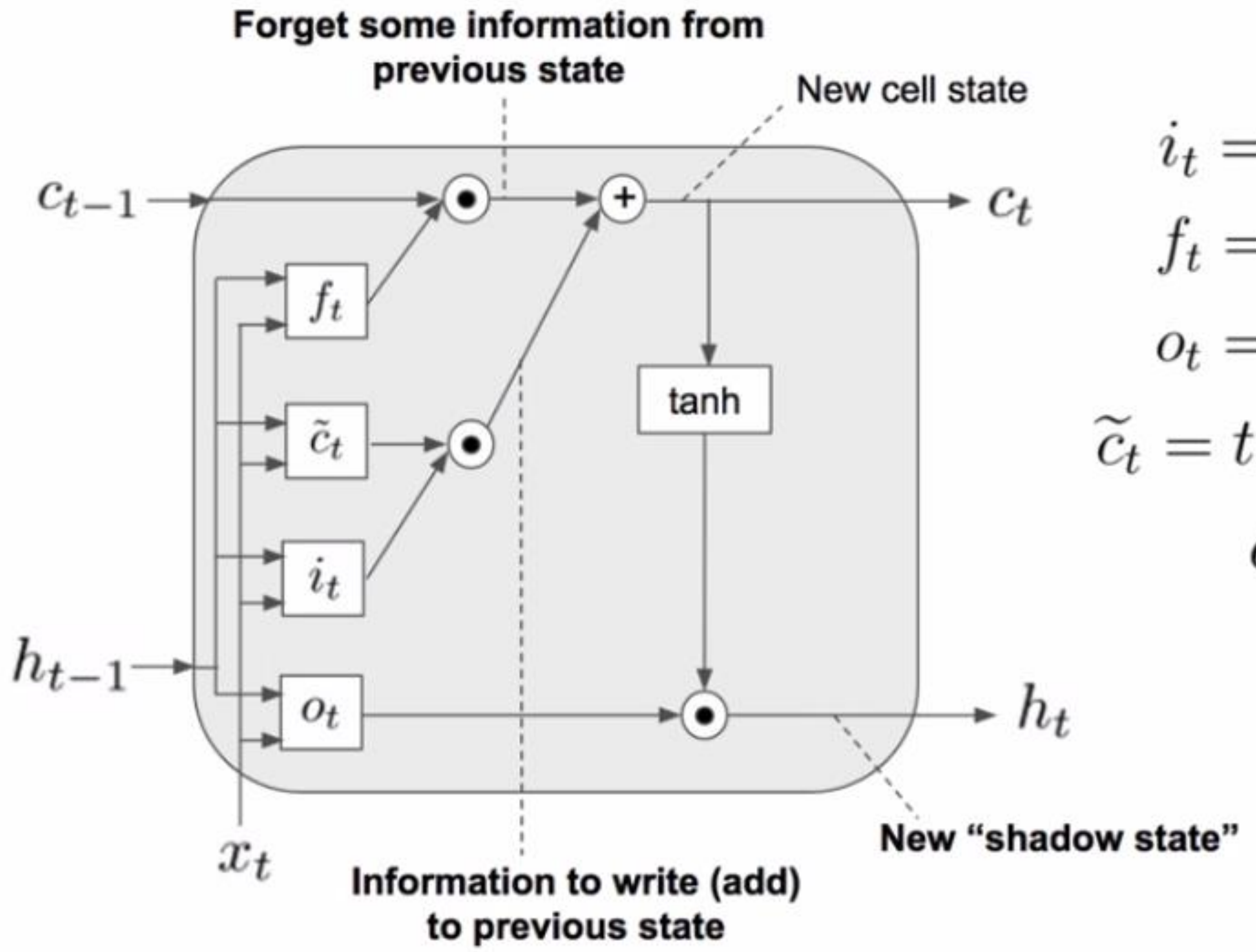
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

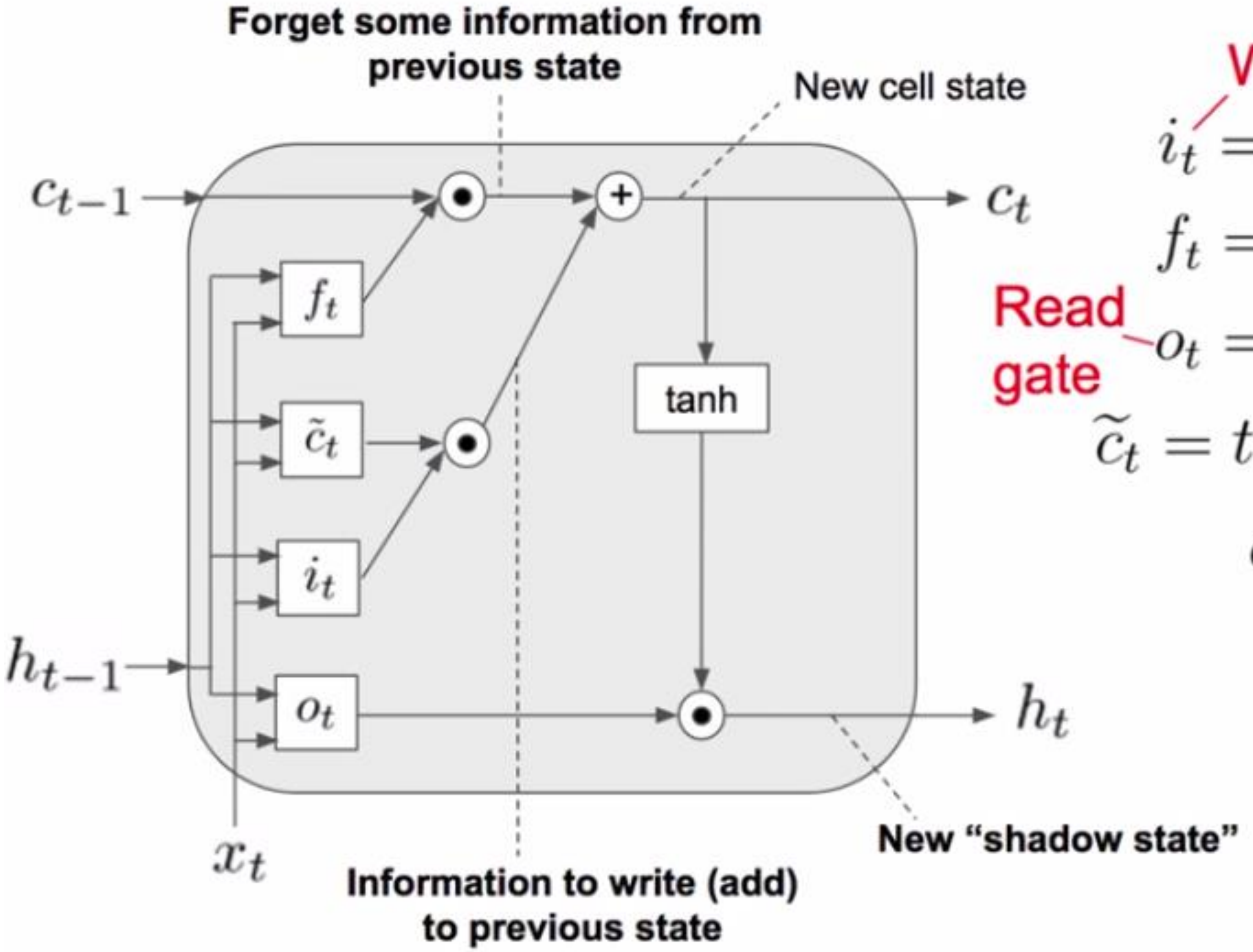
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

LSTM Cell functions



Conceptually, we lose information

LSTM equations

We use shadow state to calculate gates

Write gate

Read gate

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

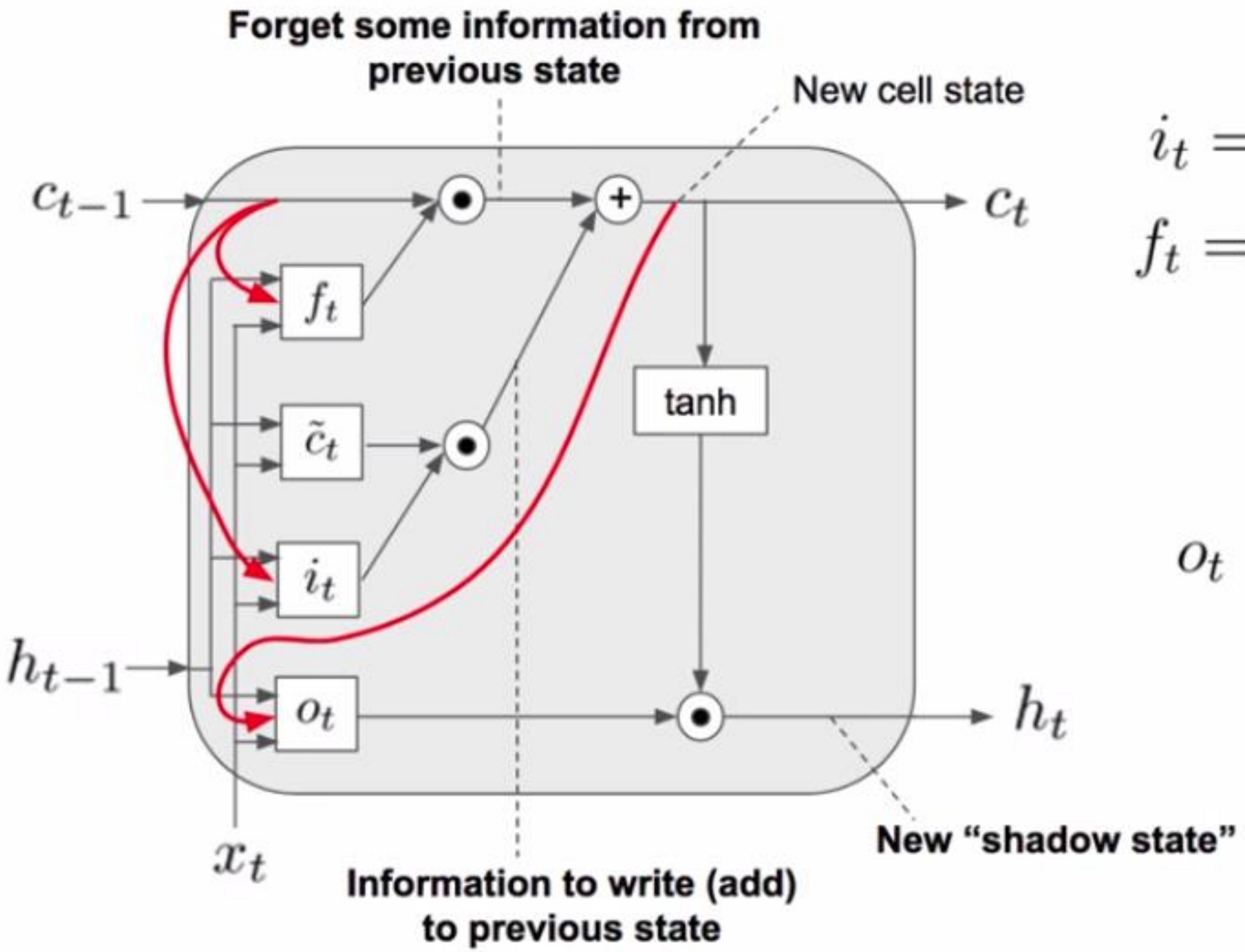
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

Read occurs after writing

LSTM Cell with Peephole connections



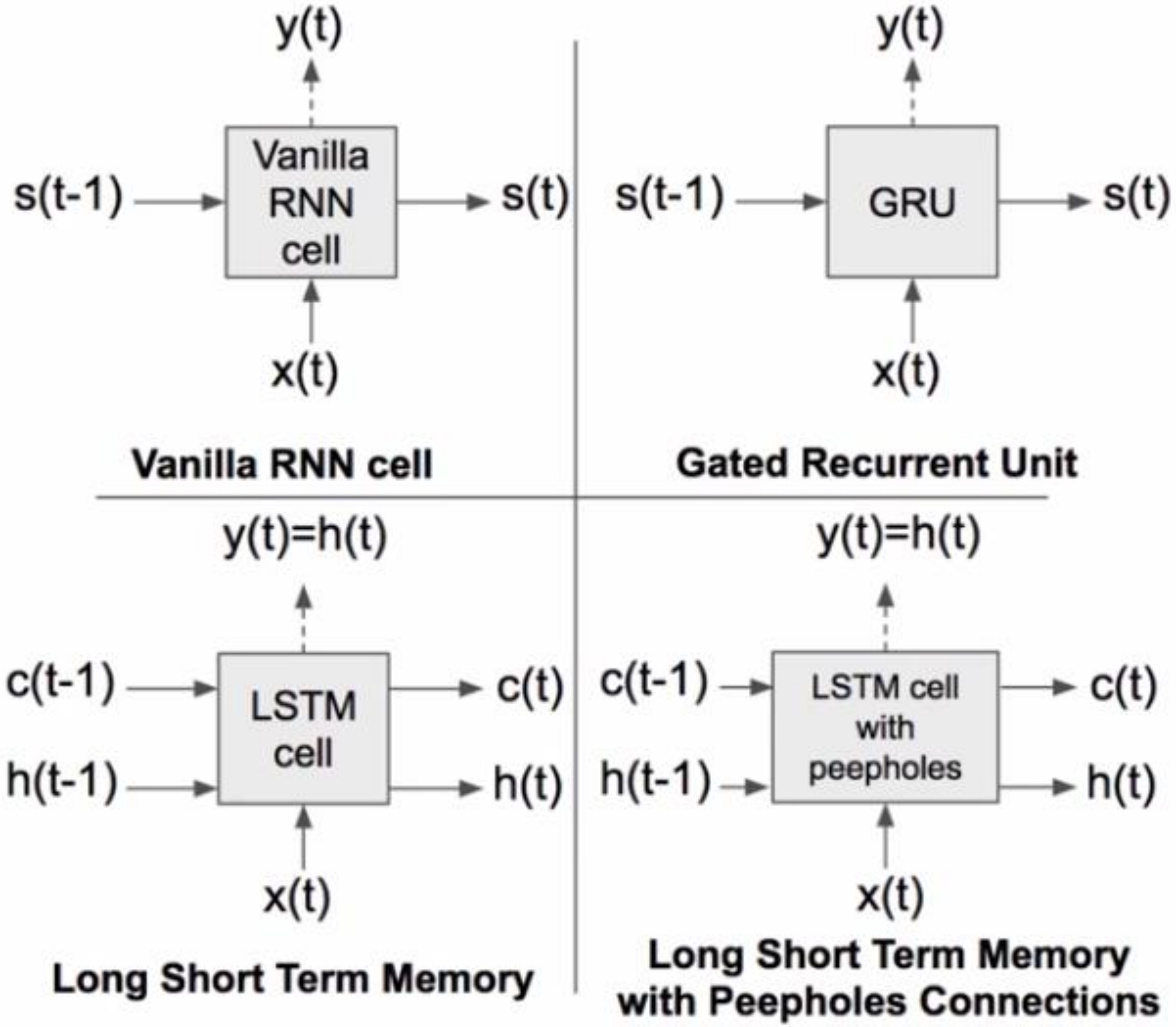
LSTM with Peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + \underline{P_i c_{t-1}} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + \underline{P_f c_{t-1}} + b_f)$$
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + \underline{P_o c_t} + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

- LSTM and GRU are the most widely used cells in production systems and to achieve state of the art

- GRU cell, perhaps, is the most intuitive one

- LSTM with Peephole connections was designed to attack potential loss of information of Basic LSTM cell



Vanilla RNN

Late 1980s - backpropagation through time to train Vanilla RNN

LSTM

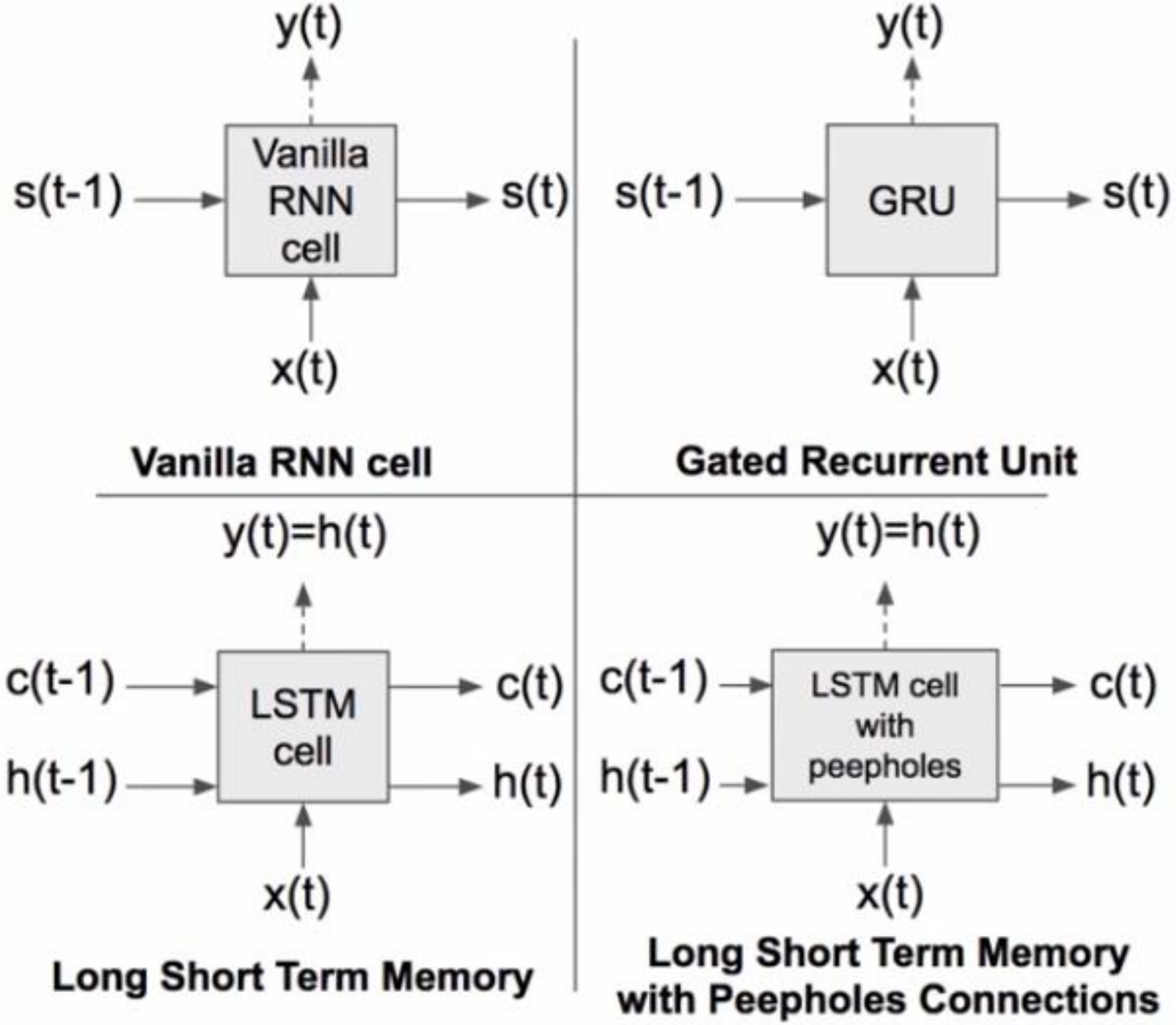
1997 - Long Short-Term Memory (S.Hochreiter, J.Schmidhuber)

LSTM with Peepholes

2000 - Recurrent nets that time and count (F.A. Gers ; J. Schmidhuber)

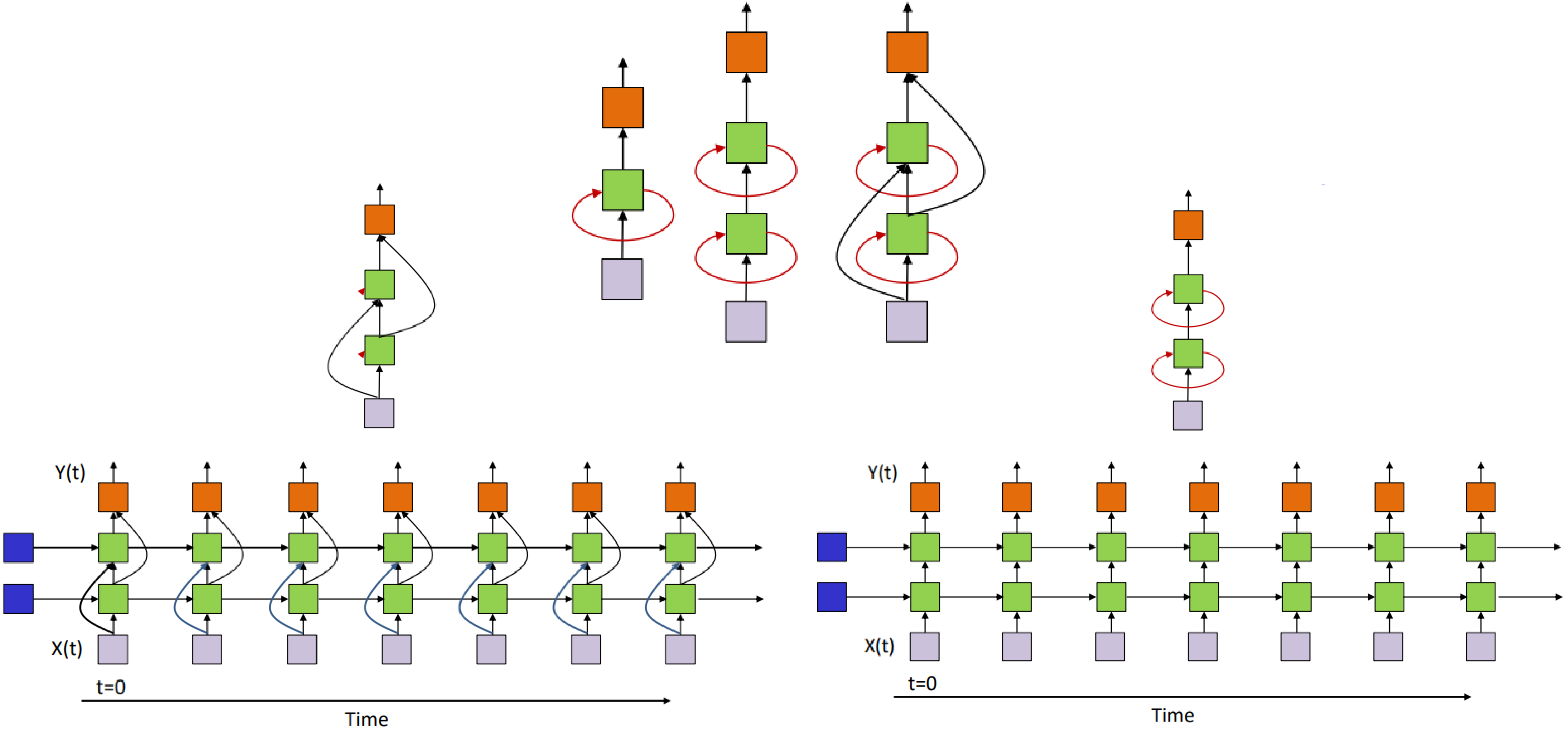
GRU

2014 - Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (Kyunghyun Cho, Yoshua Bengio, and others)

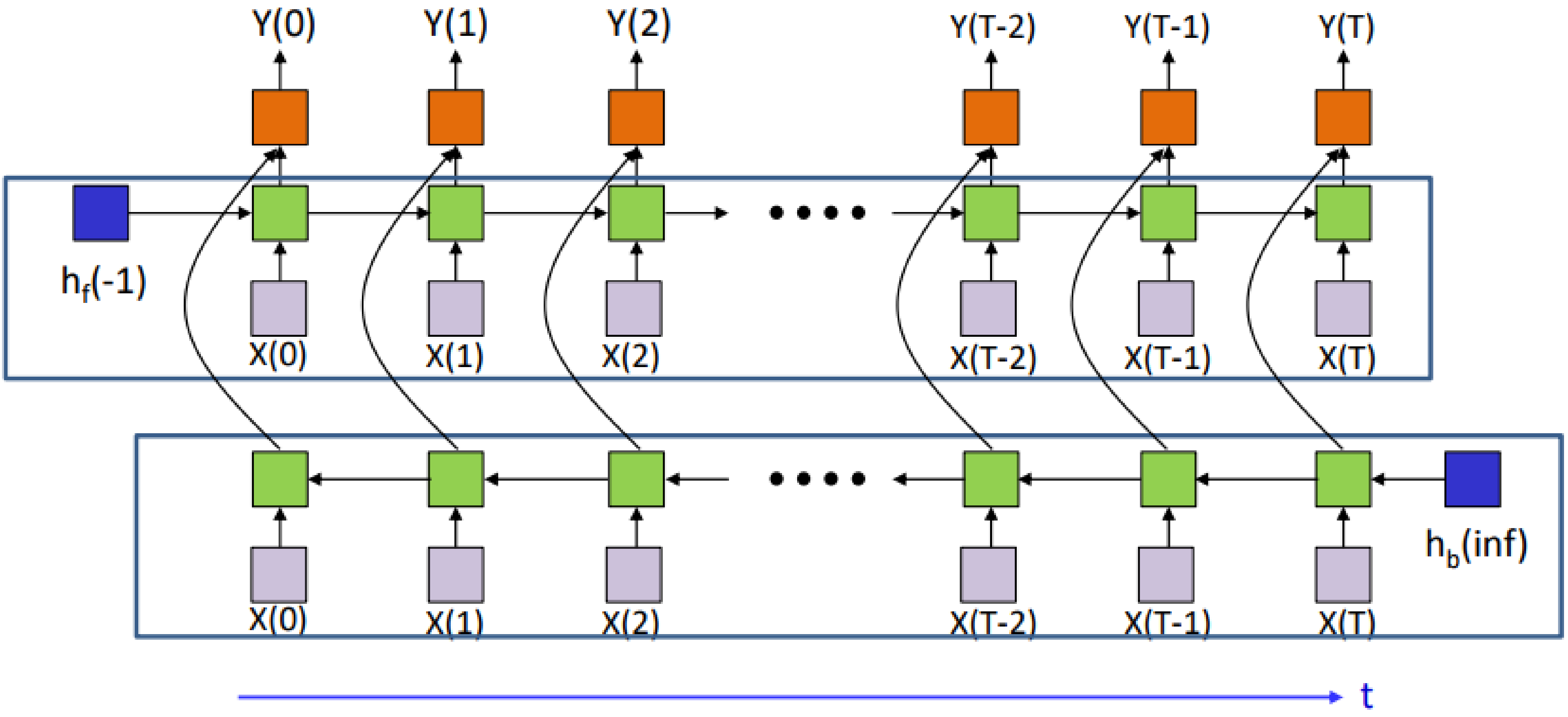


- RNN Introduction
  - Variants on Recurrent nets
  - Training RNN
- Vanishing Gradients and Exploding Gradients
- Proposed Solutions
- Deep Belief Networks
- Memory Systems
  - Gated Recurrent Unit (GRU)
  - Long-Shot Term Memory (LSTM)
- **Different Types of Recurrent Memory Models**
  - Bi-directional RNN
  - Neural Turing Machine (NTM)
  - Hopfield Neural Networks (HNN)

Different types of Recurrent Memory models



Bi-directional RNN

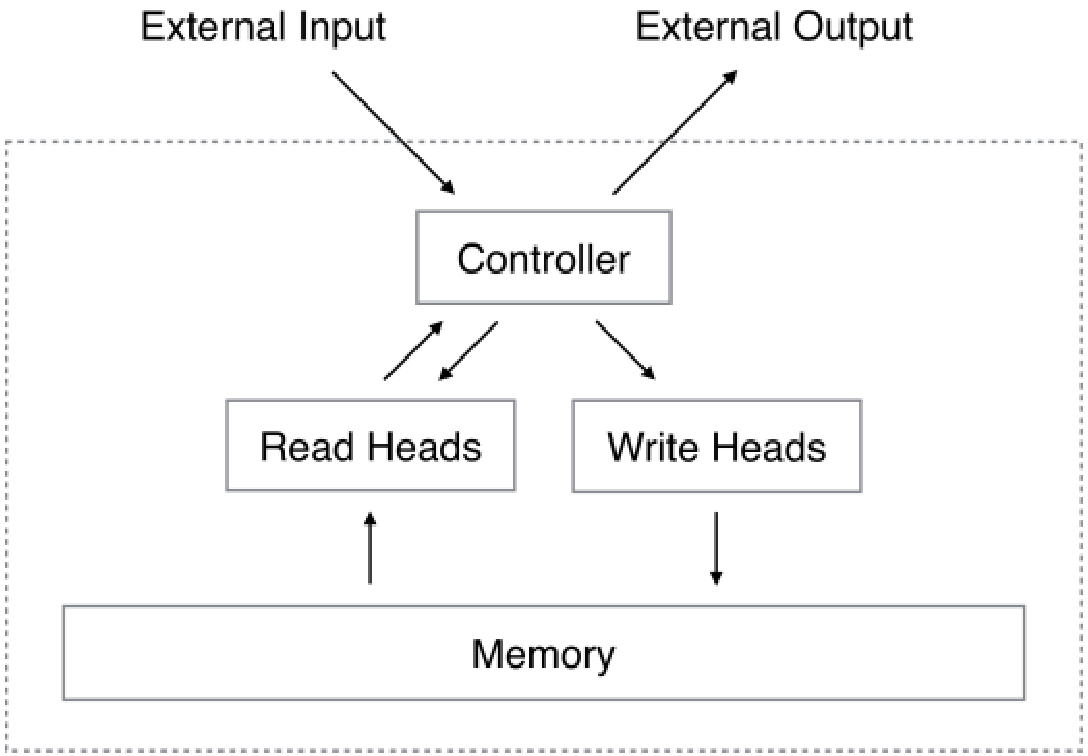


- A forward net process the data from t=0 to t=T
- A backward net processes it backward from t=T down to t=0

Neural Turing Machines

Alex Graves gravesa@google.com
Greg Wayne gregwayne@google.com
Ivo Danihelka danihelka@google.com

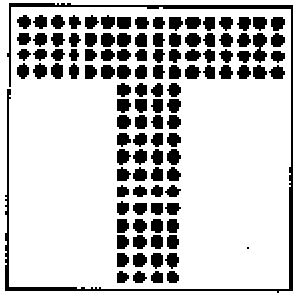
Google DeepMind, London, UK



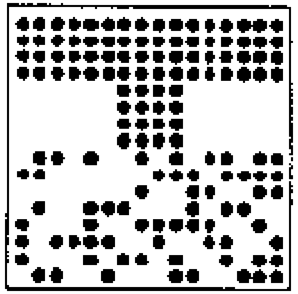
Two criticisms were answered against Neural networks

- neural networks with fixed-size inputs are seemingly unable to solve problems with variable-size inputs
- neural networks seem unable to bind values to specific locations in data structures. This ability of writing to and reading from memory is critical in the two information processing systems we have available to study: brains and computers

Hopfield Networks

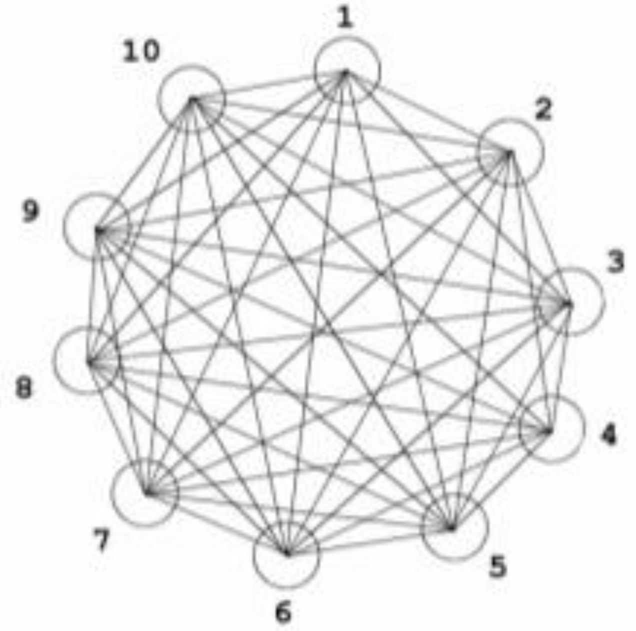


Original 'T'



half of image corrupted by noise

- Sub-type of recurrent neural nets
  - Fully recurrent
  - Weights are symmetric
  - Nodes can only be *on* or *off*
  - Random updating



- Learning: **Hebb rule** (cells that fire together wire together)

- Can recall a memory, if presented with a corrupt or incomplete version

→ **auto-associative** or
content-addressable memory

Thank You