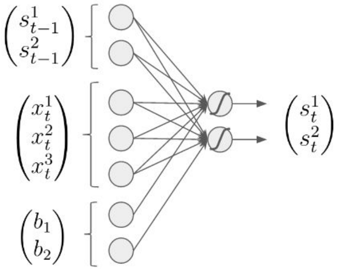
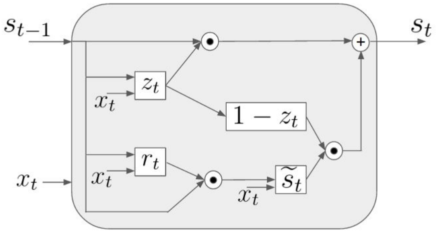
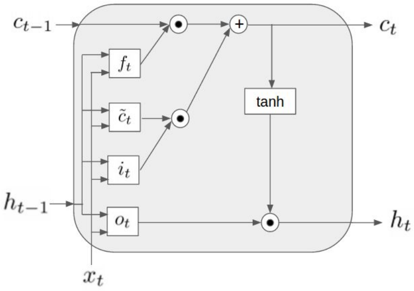
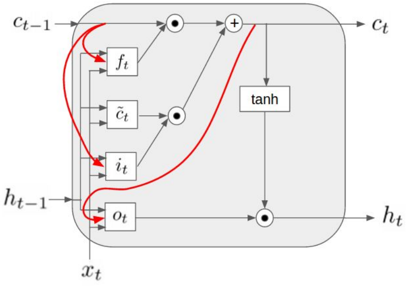
Review from {} deepsystems.ai

Evolution:
from vanilla RNN to GRU & LSTMs



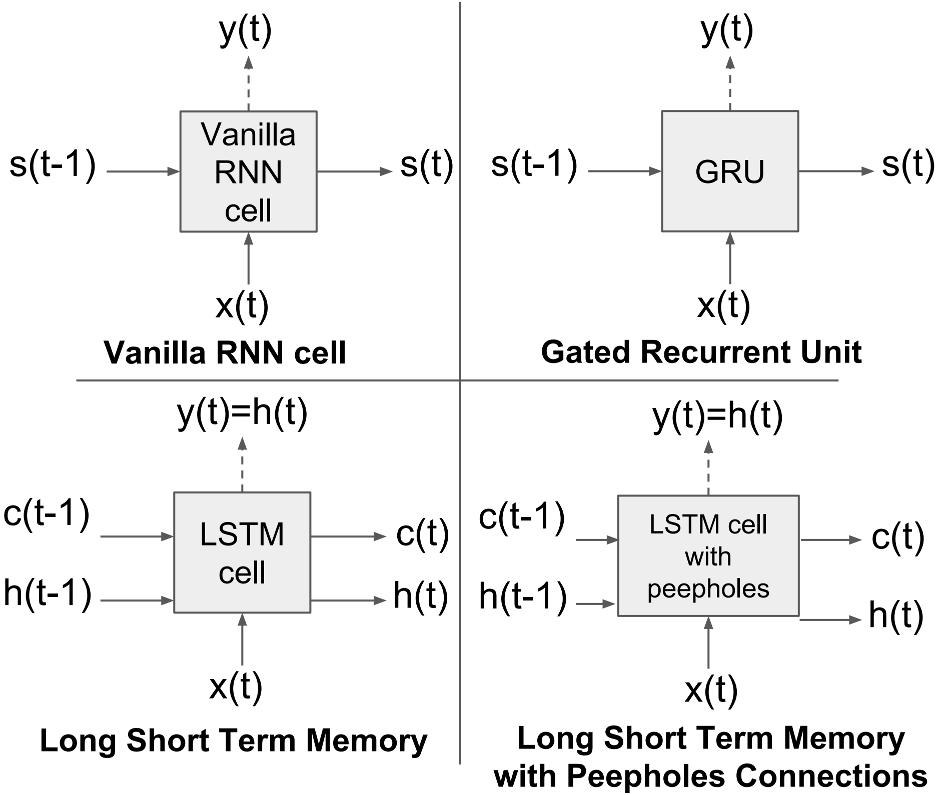






By the end of the talk you will learn

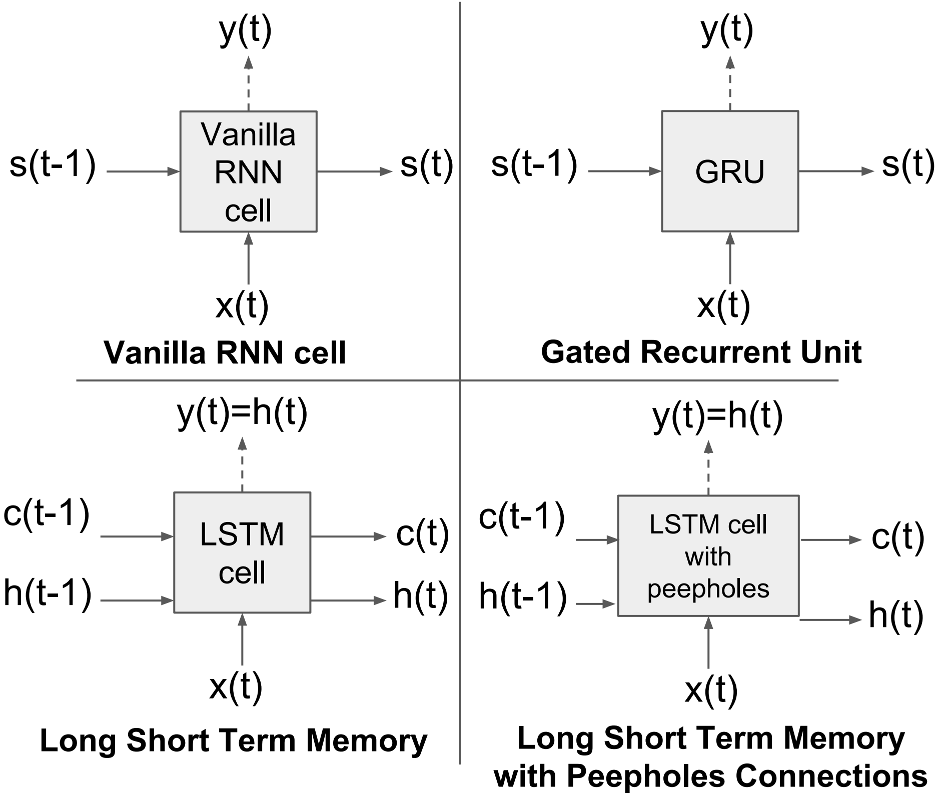
For the most common RNN Cells:



By the end of the talk you will learn

For the most common RNN Cells:

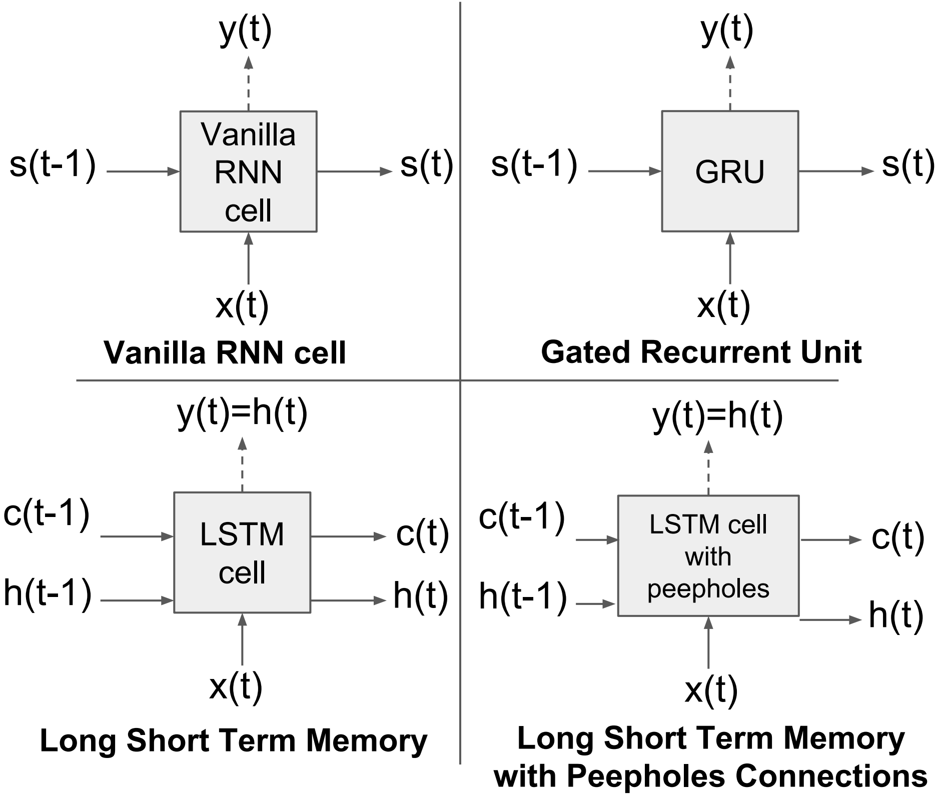
- What's inside & how it works



By the end of the talk you will learn

For the most common RNN Cells:

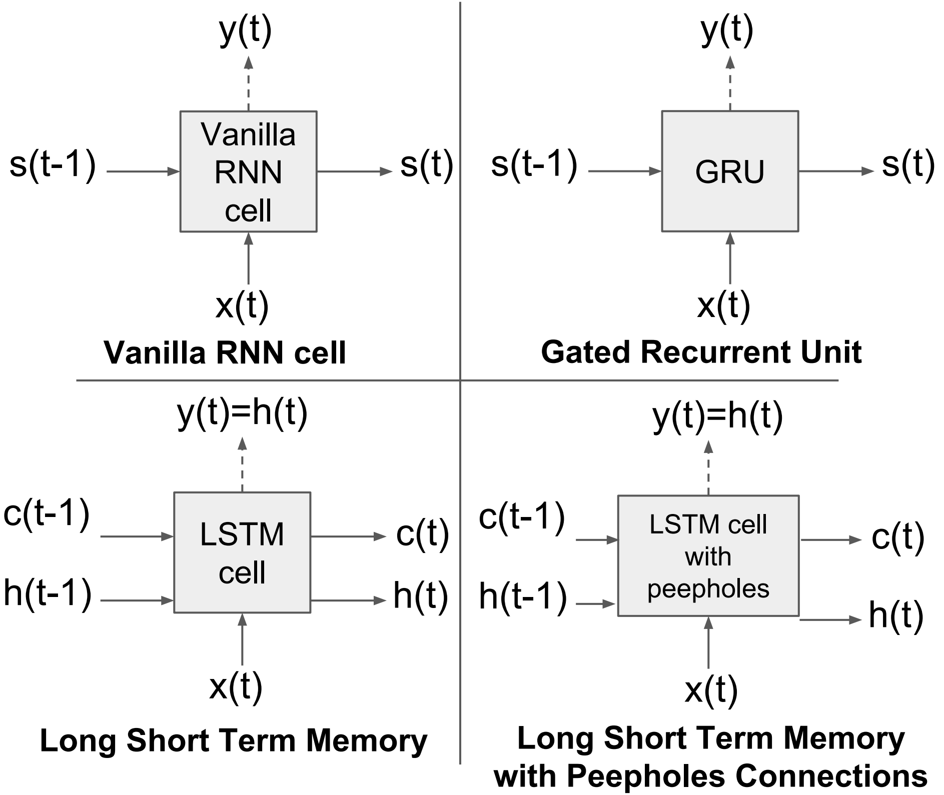
- What's inside & how it works
- Intuition behind



By the end of the talk you will learn

For the most common RNN Cells:

- What’s inside & how it works
- Intuition behind
- Advantages and potential problems



deepsystems.ai

Talk outline

Talk outline

1. RNN: the cell & simple examples

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

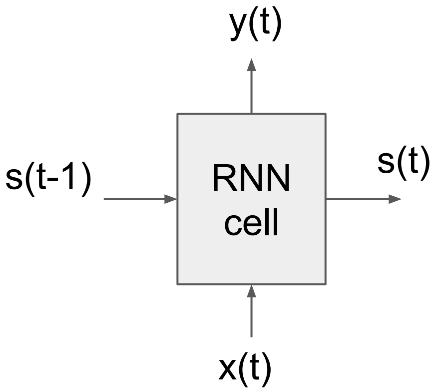
Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

RNN Cell

rnn cell output (optional, in most cases y(t)=s(t))

Hidden state at t-1
(previous state)



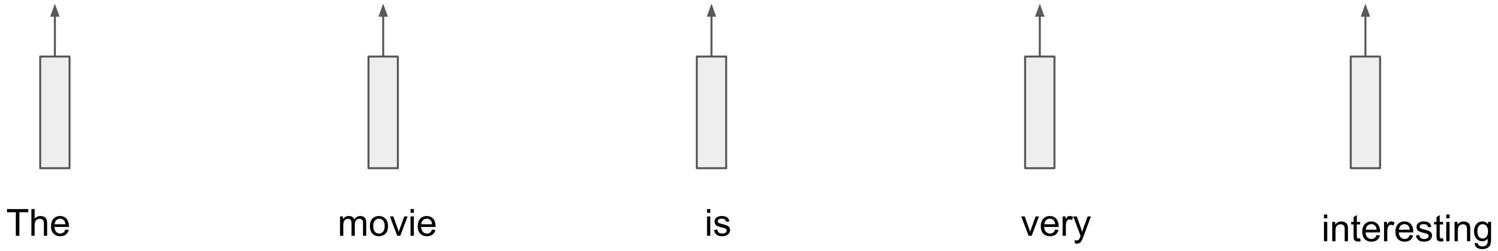
Hidden state at time t
(current state)

Input vector
(optional)

RNN example 1: Sentiment analysis

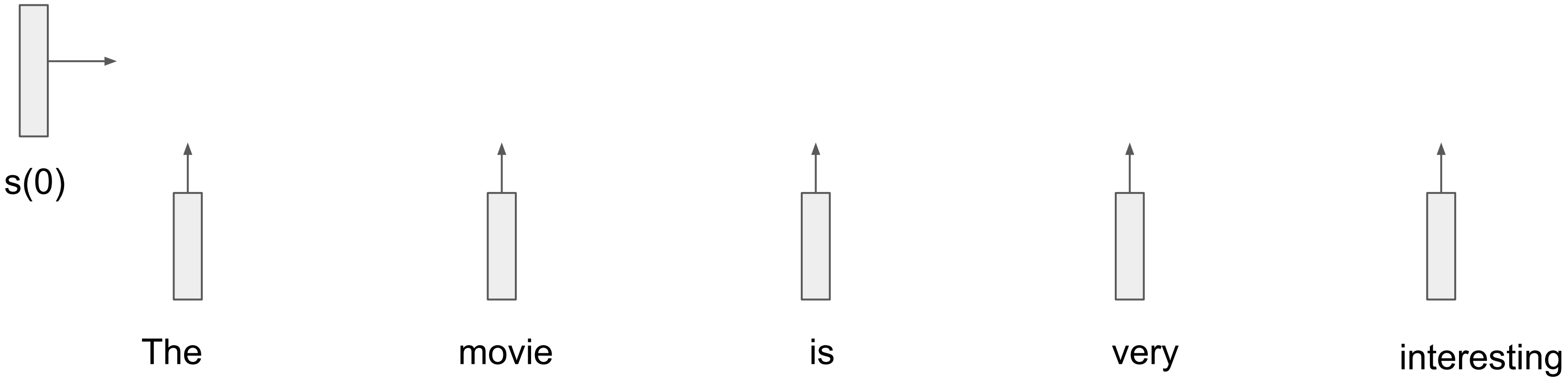
The movie is very interesting

RNN example 1: Sentiment analysis



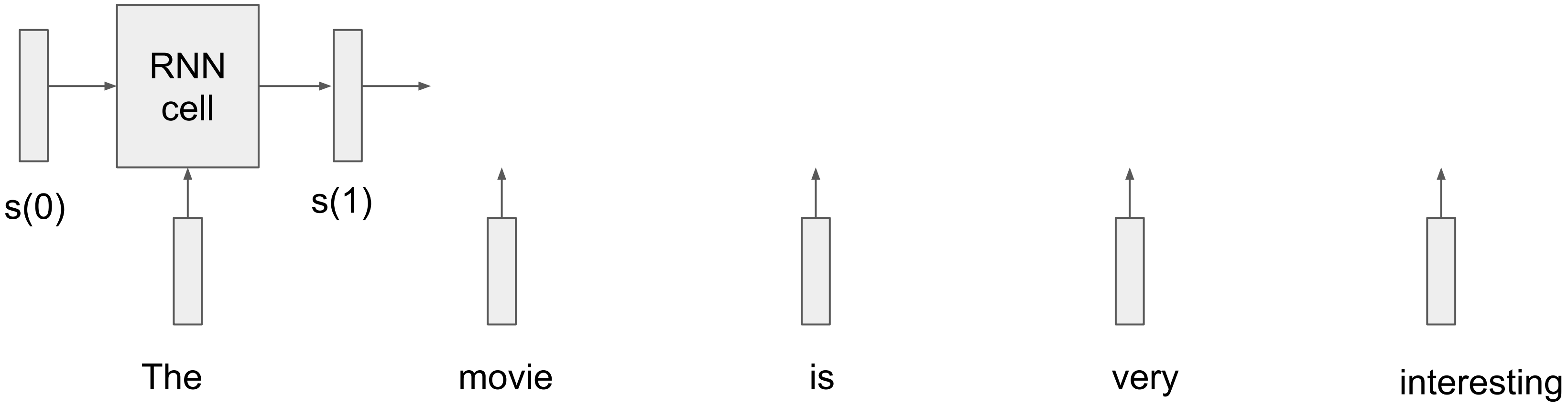
deepsystems.ai

RNN example 1: Sentiment analysis



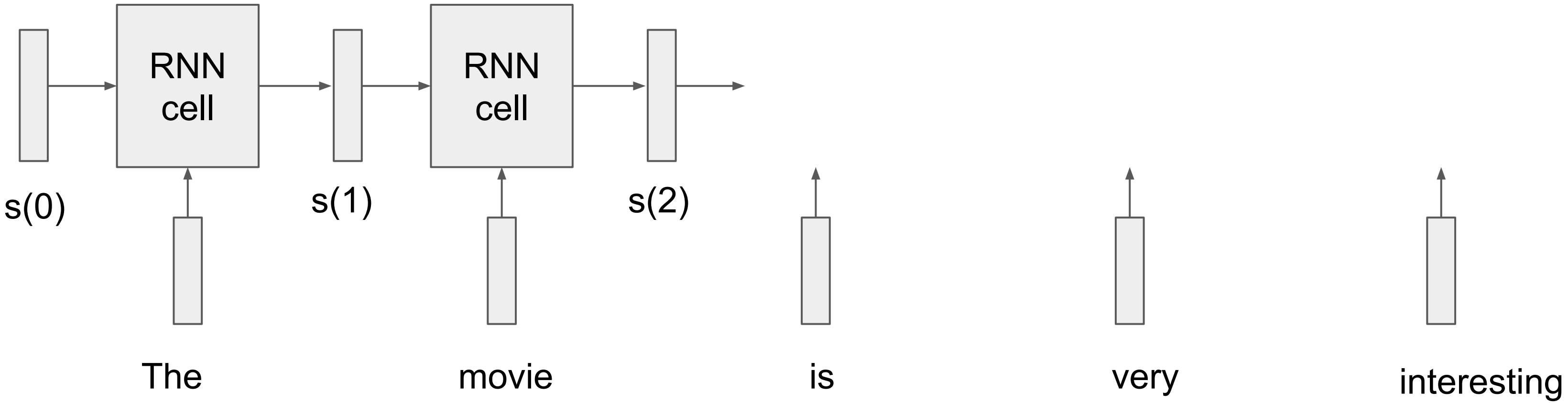
deepsystems.ai

RNN example 1: Sentiment analysis

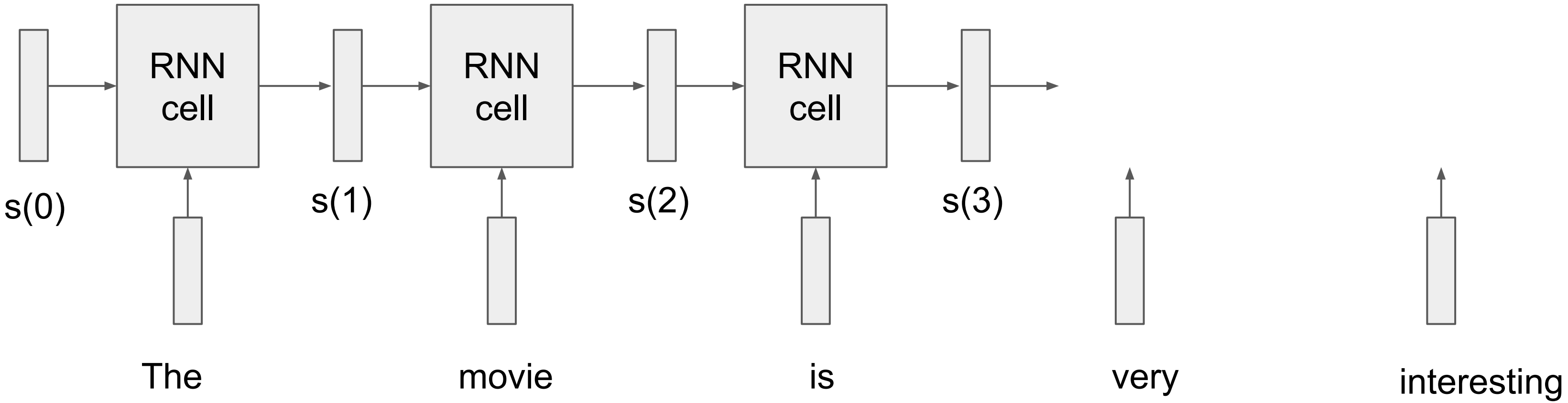


deepsystems.ai

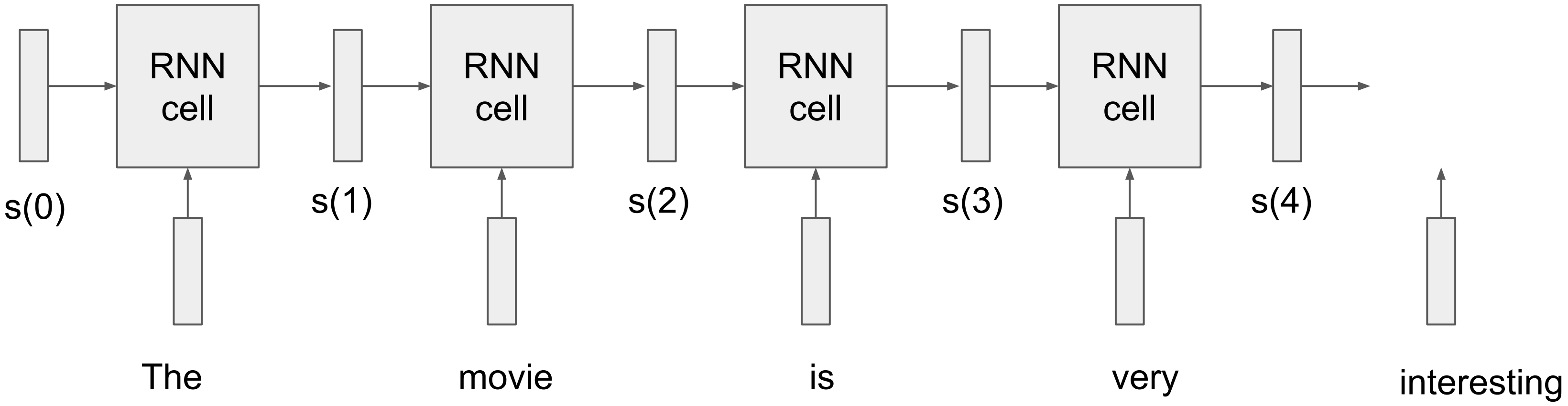
RNN example 1: Sentiment analysis



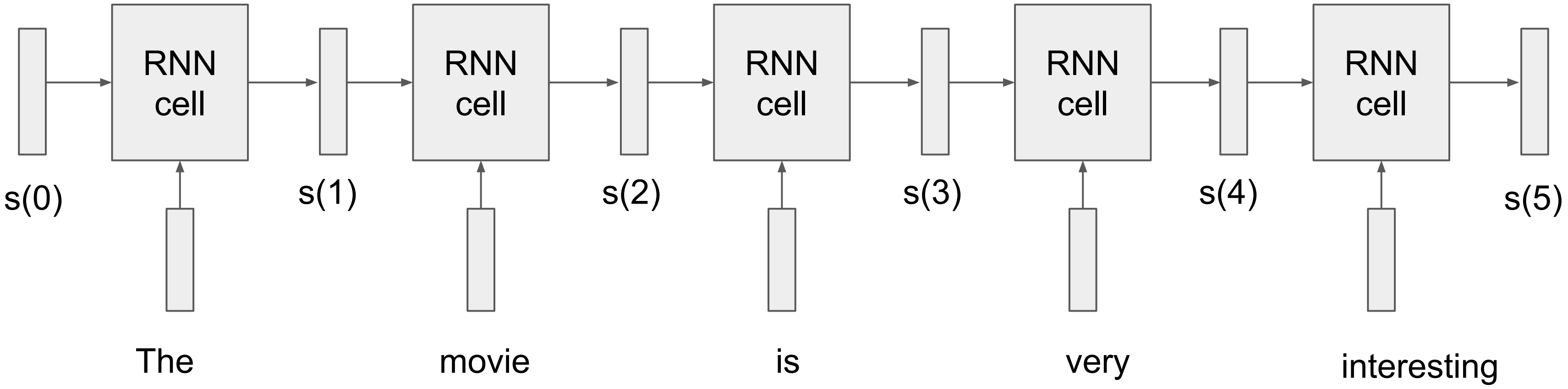
RNN example 1: Sentiment analysis



RNN example 1: Sentiment analysis

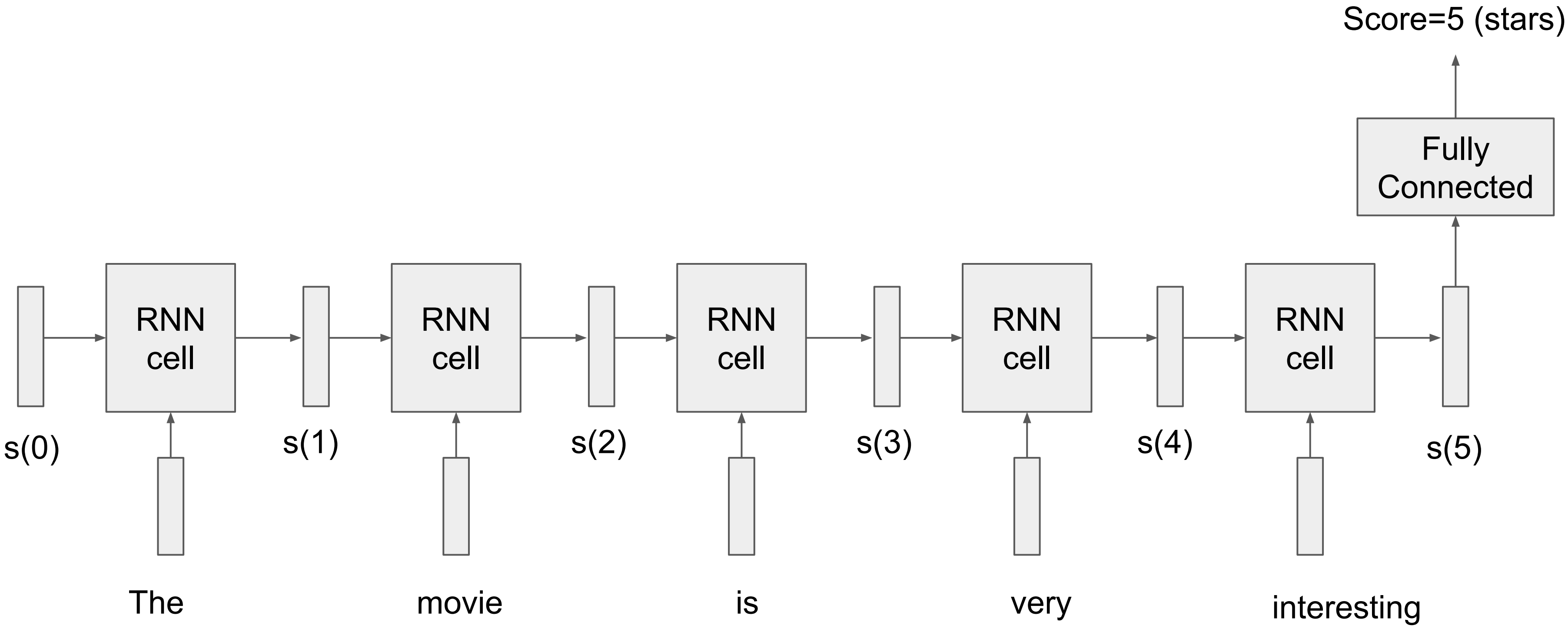


RNN example 1: Sentiment analysis



deepsystems.ai

RNN example 1: Sentiment analysis



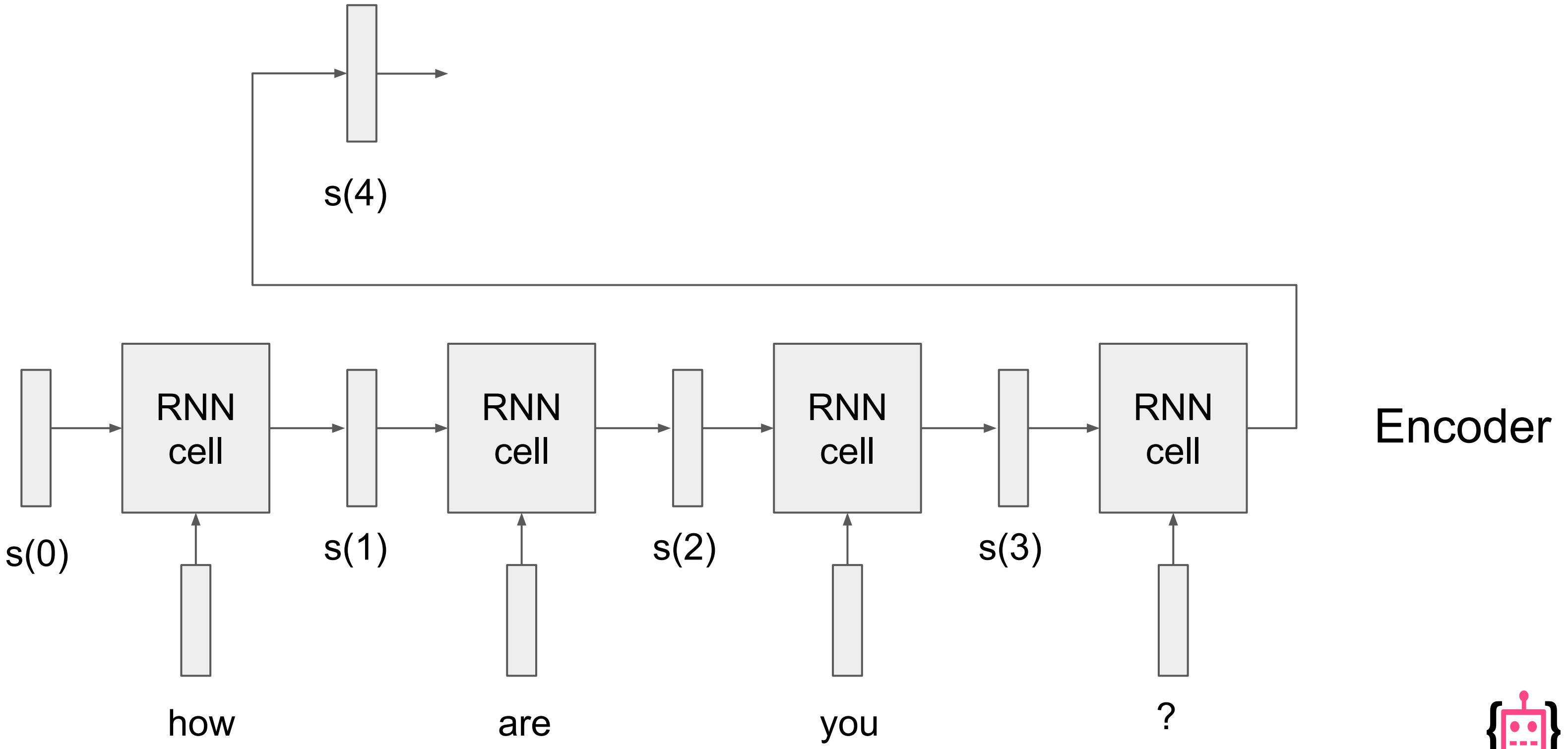
deepsystems.ai

RNN example 2: English to slang English

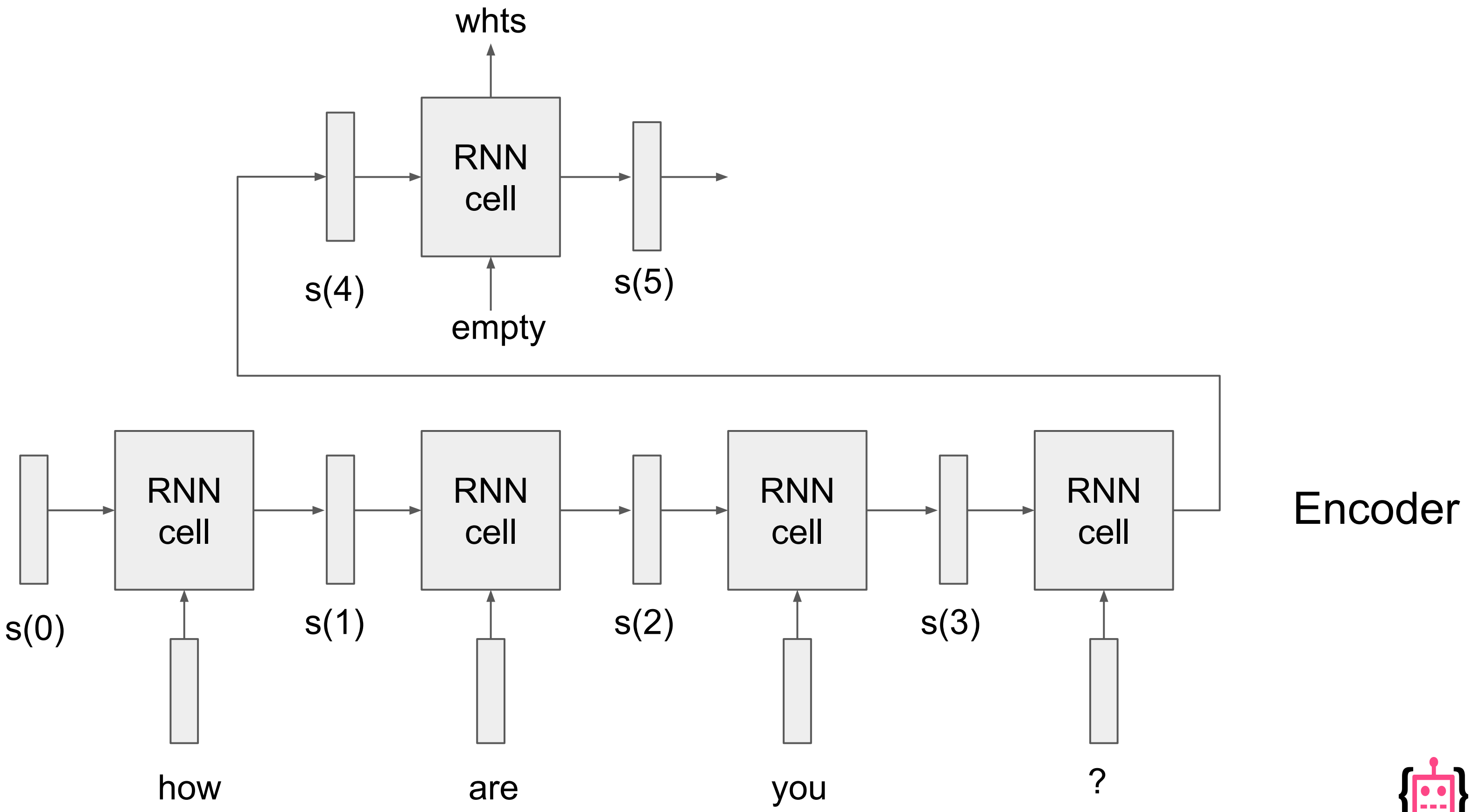
RNN example 2: English to slang English

how are you ?

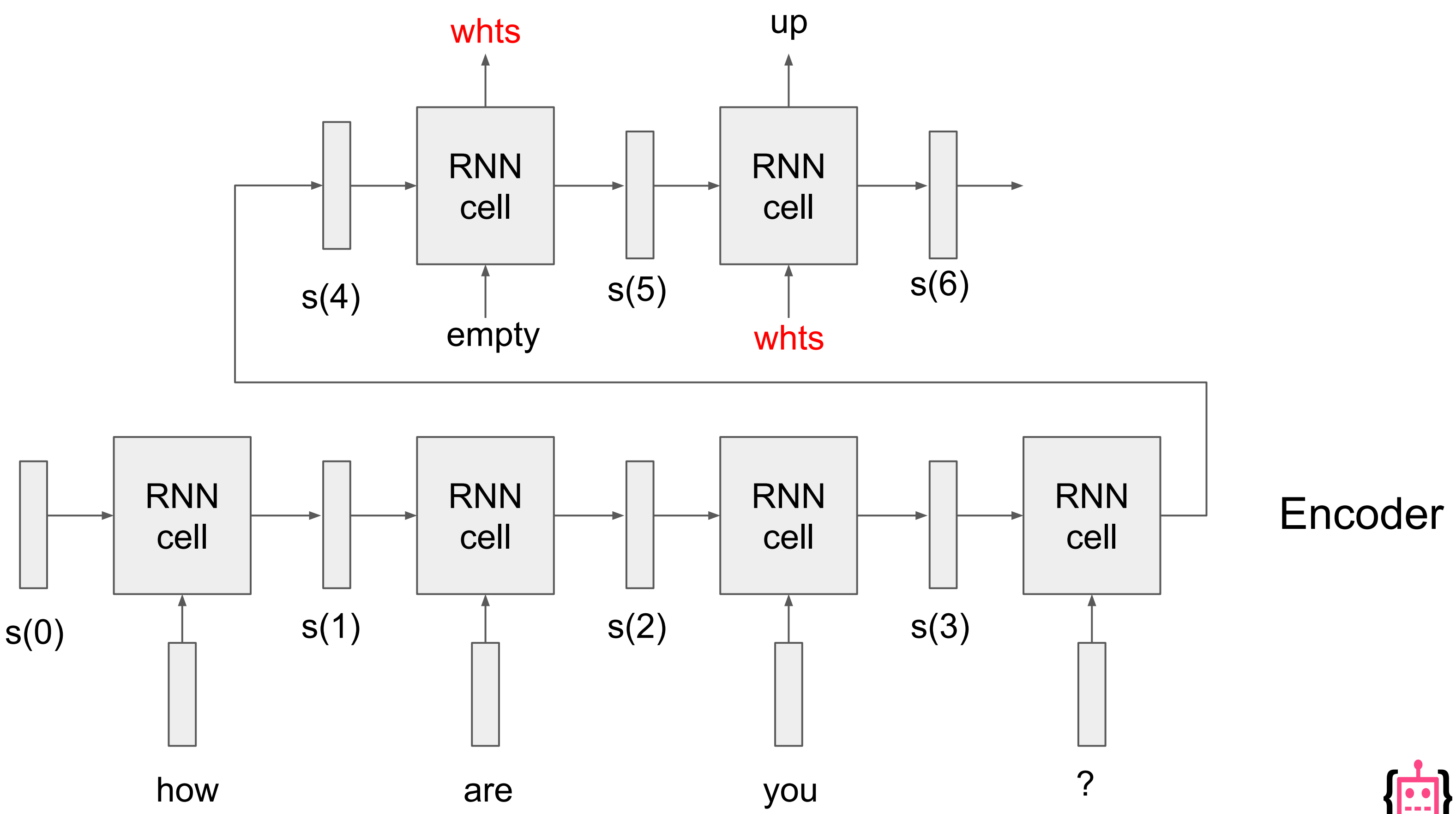
RNN example 2: English to slang English



RNN example 2: English to slang English

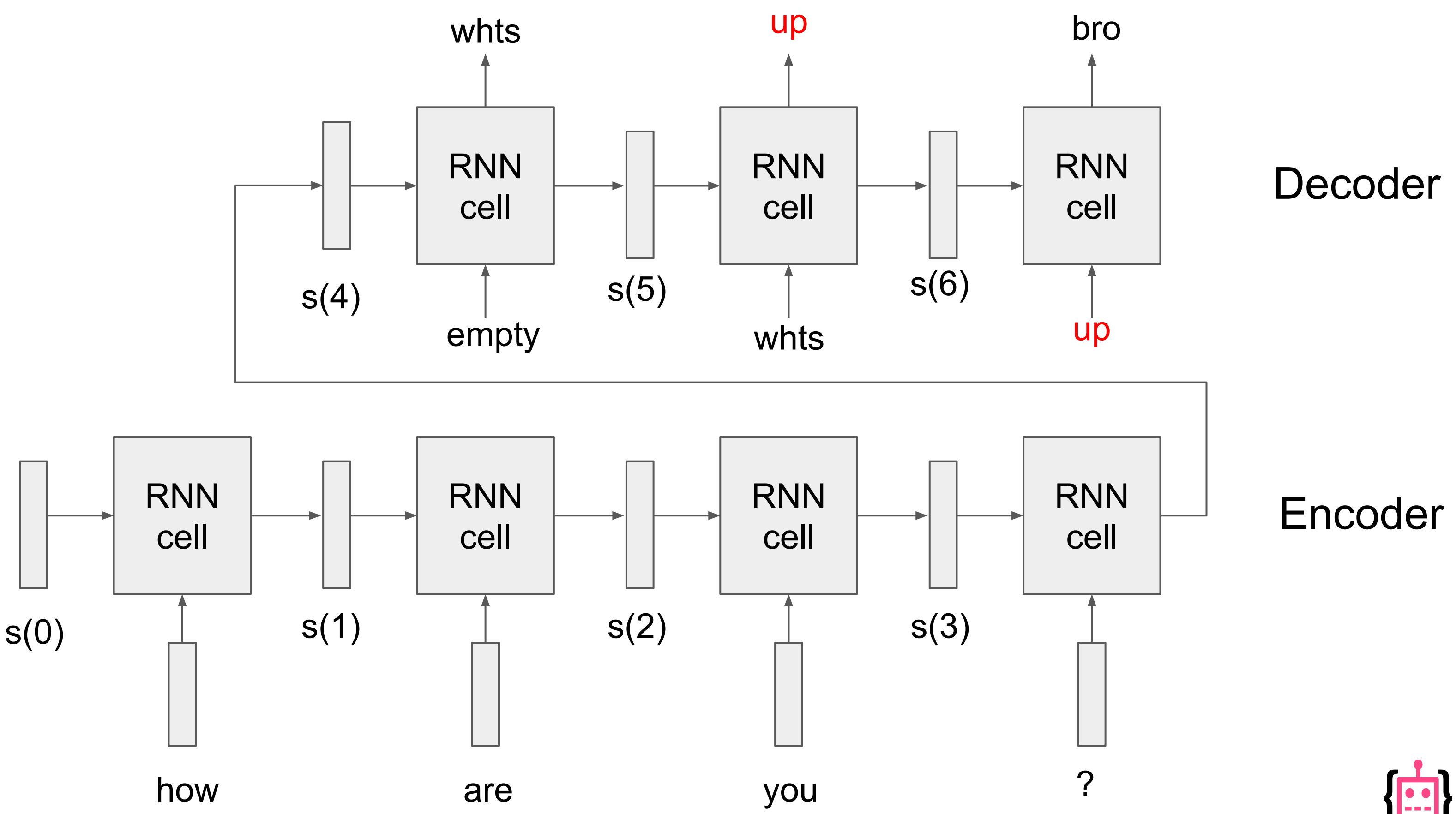


RNN example 2: English to slang English



deepsystems.ai

RNN example 2: English to slang English



deepsystems.ai

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

RNNs: Key Aspects (or ways to state of the art)

RNN Cell structure
(What's inside the cell)

- Legacy
  - Vanilla rnn cell
- Widely used
  - GRU
  - LSTM
- Other alternatives
  - LSTM with peepholes connections
  - MI-LSTM

RNN Topology
(How the cells interconnected)

- Single/Multilayer
- Encoder/Decoder
- Bidirectional
- Grid LSTM
- Tree LSTM

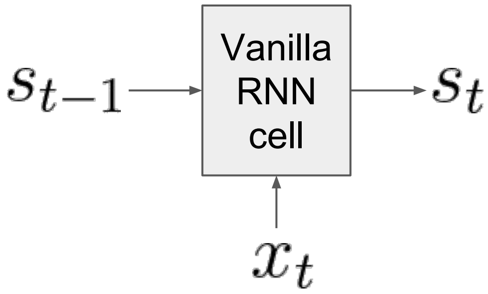
Additional components
(How to make it work)

- Attention
- Regularization
- Normalization
- Share something
- Unshare something (hyper lstm)
- CTC loss

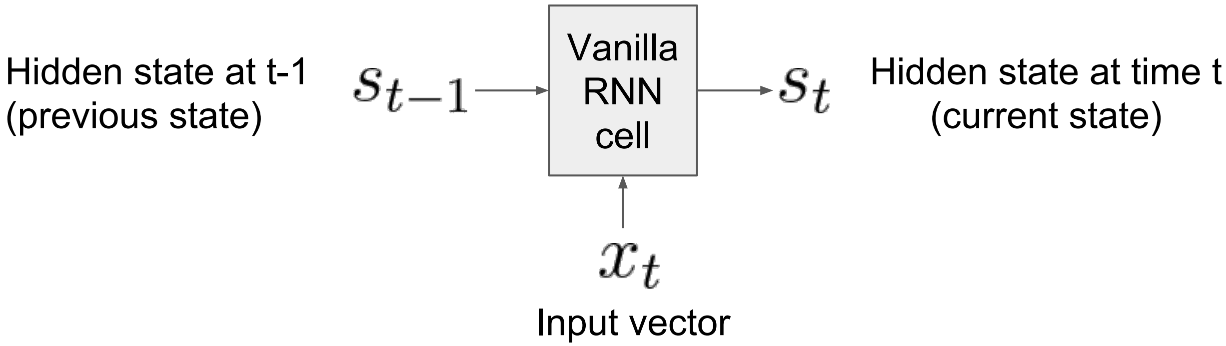
Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

Vanilla RNN



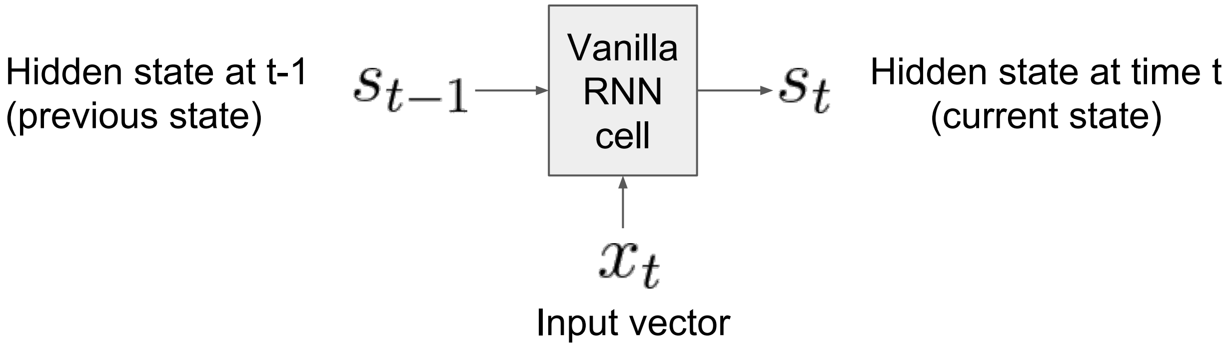
Vanilla RNN



deepsystems.ai

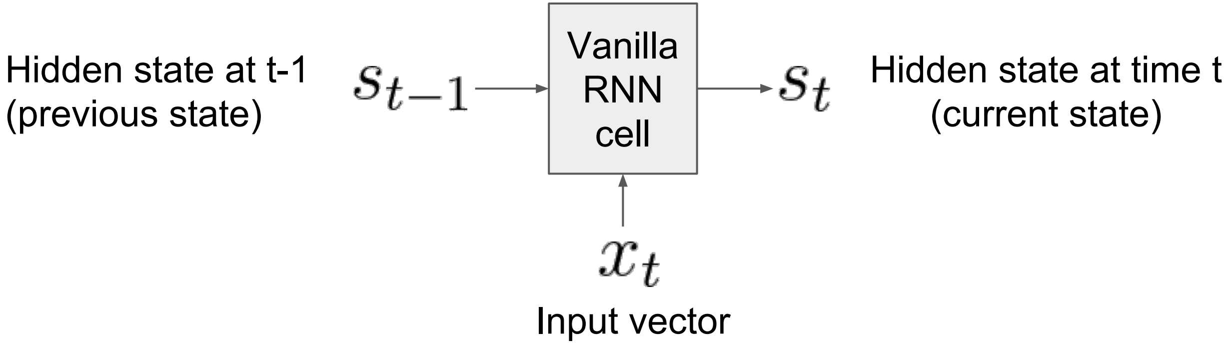
Vanilla RNN

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



Vanilla RNN

$$\overset{(n \times 1)}{s_t} = \varphi(\overset{(n \times n)}{W}\,\overset{(n \times 1)}{s_{t-1}} + \overset{(n \times m)}{U}\,\overset{(m \times 1)}{x_t} + \overset{(n \times 1)}{b})$$

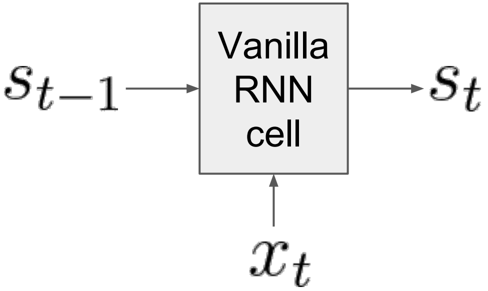


deepsystems.ai

Vanilla RNN

$$\overset{(n \times 1)}{s_t} = \varphi(\overset{(n \times n)}{W}\,\overset{(n \times 1)}{s_{t-1}} + \overset{(n \times m)}{U}\,\overset{(m \times 1)}{x_t} + \overset{(n \times 1)}{b})$$

Hidden state at t-1
(previous state)



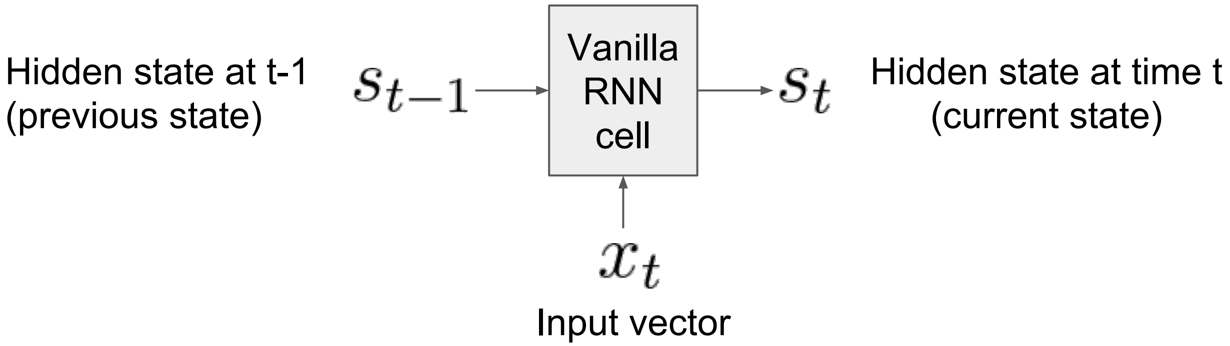
Hidden state at time t
(current state)

Input vector

Let’s show that Vanilla RNN is just Single Layer Network (with feedback)

Vanilla RNN

$$\underset{(n \times 1)}{s_t} = \varphi(\underset{(n \times n)}{W} \underset{(n \times 1)}{s_{t-1}} + \underset{(n \times m)}{U} \underset{(m \times 1)}{x_t} + \underset{(n \times 1)}{b})$$



Let's show that Vanilla RNN is just Single Layer Network (with feedback)

Let $n = 2, m = 3$

Vanilla RNN

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

$n = 2$ (state size),
$m = 3$ (input size)

Vanilla RNN

$n = 2$ (state size),
$m = 3$ (input size)

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

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

Vanilla RNN

$n = 2$ (state size),
$m = 3$ (input size)

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

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11} s_{t-1}^1 + w_{12} s_{t-1}^2 \\ w_{21} s_{t-1}^1 + w_{22} s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} x_t^1 + u_{12} x_t^2 + u_{13} x_t^3 \\ u_{21} x_t^1 + u_{22} x_t^2 + u_{23} x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

Vanilla RNN

$n = 2$ (state size),
$m = 3$ (input size)

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

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11} s_{t-1}^1 + w_{12} s_{t-1}^2 \\ w_{21} s_{t-1}^1 + w_{22} s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} x_t^1 + u_{12} x_t^2 + u_{13} x_t^3 \\ u_{21} x_t^1 + u_{22} x_t^2 + u_{23} x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11} s_{t-1}^1 + w_{12} s_{t-1}^2 + u_{11} x_t^1 + u_{12} x_t^2 + u_{13} x_t^3 \\ w_{21} s_{t-1}^1 + w_{22} s_{t-1}^2 + u_{21} x_t^1 + u_{22} x_t^2 + u_{23} x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

Vanilla RNN

$n = 2$ (state size),
$m = 3$ (input size)

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

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11}s_{t-1}^1 + w_{12}s_{t-1}^2 \\ w_{21}s_{t-1}^1 + w_{22}s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11}x_t^1 + u_{12}x_t^2 + u_{13}x_t^3 \\ u_{21}x_t^1 + u_{22}x_t^2 + u_{23}x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11}s_{t-1}^1 + w_{12}s_{t-1}^2 + u_{11}x_t^1 + u_{12}x_t^2 + u_{13}x_t^3 \\ w_{21}s_{t-1}^1 + w_{22}s_{t-1}^2 + u_{21}x_t^1 + u_{22}x_t^2 + u_{23}x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

Vanilla RNN

$n = 2$ (state size),

$m = 3$ (input size)

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

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

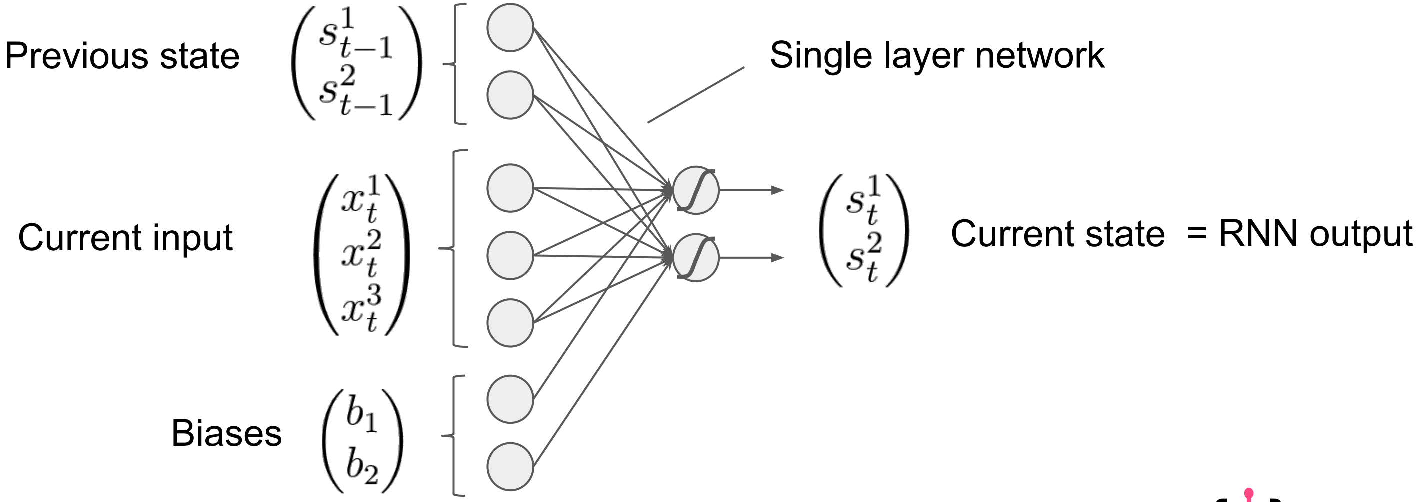
$$= \varphi(\begin{pmatrix} w_{11}s_{t-1}^1 + w_{12}s_{t-1}^2 \\ w_{21}s_{t-1}^1 + w_{22}s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11}x_t^1 + u_{12}x_t^2 + u_{13}x_t^3 \\ u_{21}x_t^1 + u_{22}x_t^2 + u_{23}x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11}s_{t-1}^1 + w_{12}s_{t-1}^2 + u_{11}x_t^1 + u_{12}x_t^2 + u_{13}x_t^3 \\ w_{21}s_{t-1}^1 + w_{22}s_{t-1}^2 + u_{21}x_t^1 + u_{22}x_t^2 + u_{23}x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

$$= \varphi(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$ - single layer network

Vanilla RNN

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$



Vanilla RNN

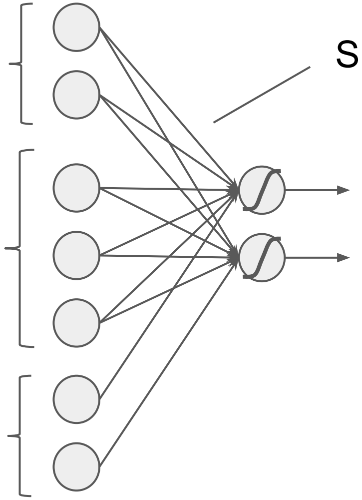
$$s_t = \varphi(W_c[s_{t-1}, x_t] + b)$$

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix})$$

Previous state $\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix}$

Current input $\begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix}$

Biases $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$



Single layer network

$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix}$ Current state = RNN output

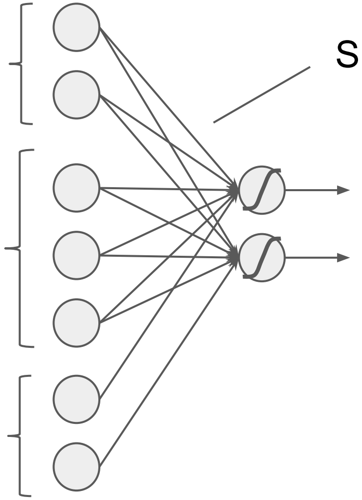
Vanilla RNN

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi\left(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} & b_1 \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} & b_2 \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \\ 1 \end{pmatrix}\right)$$

Previous state $\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix}$

Current input $\begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix}$

Biases $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$



Single layer network

$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix}$ Current state = RNN output

Vanilla RNN

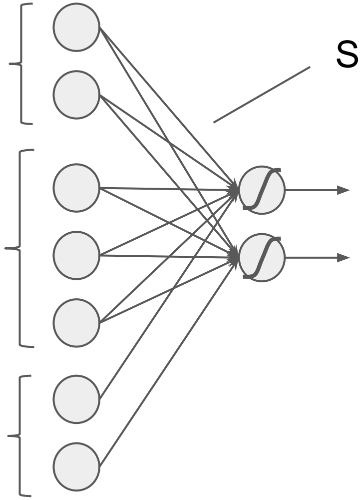
$$s_t = \varphi(W_b[s_{t-1}, x_t, 1])$$

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} & b_1 \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} & b_2 \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \\ 1 \end{pmatrix})$$

Previous state $\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix}$

Current input $\begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix}$

Biases $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$



Single layer network

$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix}$ Current state = RNN output

Vanilla RNN

$$s_t = \varphi(W_b[s_{t-1}, x_t, 1])$$

$$s_t = \varphi(W_b[s_{t-1}, x_t])$$

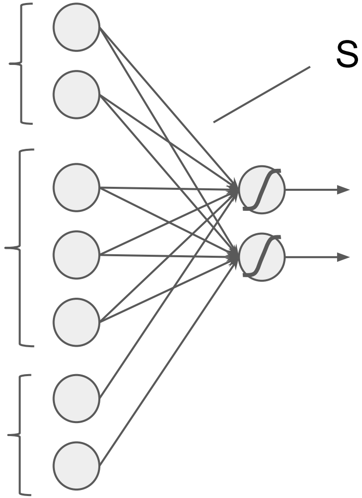
But often “1” is omitted

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} & b_1 \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} & b_2 \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \\ 1 \end{pmatrix})$$

Previous state $\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix}$

Current input $\begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix}$

Biases $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$



Single layer network

$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix}$ Current state = RNN output

Vanilla RNN. Notations

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

sometimes written as

$$s_t = \varphi(W_c[s_{t-1}, x_t] + b)$$

sometimes as

$$s_t = \varphi(W_b[s_{t-1}, x_t])$$

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

What’s wrong with Vanilla RNNs?

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

What’s wrong with Vanilla RNNs?

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

They fail to solve complex tasks (the ones that have practical applications)

What's wrong with Vanilla RNNs?

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

They fail to solve complex tasks (the ones that have practical applications)

Main problems:

What's wrong with Vanilla RNNs?

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

They fail to solve complex tasks (the ones that have practical applications)

Main problems:

Information morphing (fundamental)

What's wrong with Vanilla RNNs?

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

They fail to solve complex tasks (the ones that have practical applications)

Main problems:

Information morphing (fundamental)

↓

Gradient vanishing (technical)

What's wrong with Vanilla RNNs?

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

They fail to solve complex tasks (the ones that have practical applications)

Main problems:

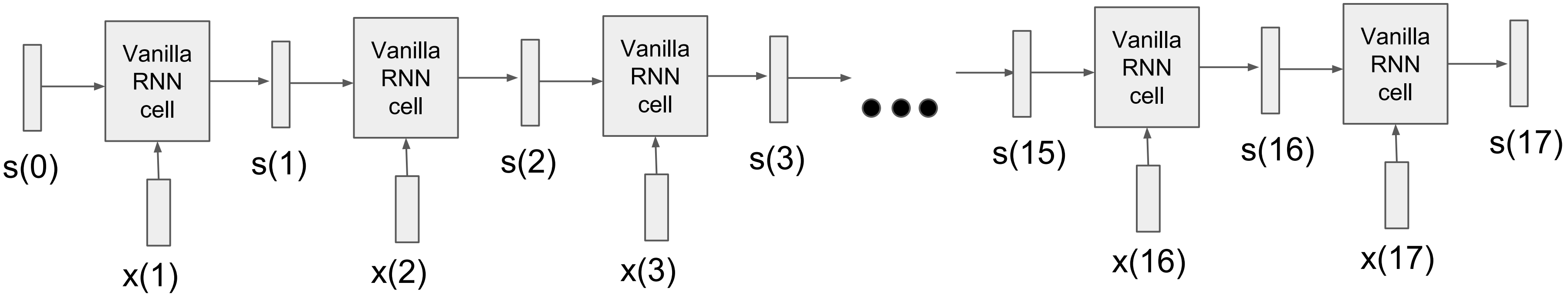
Information morphing (fundamental) ⇨ Inability to keep the memory content for more than a few time steps

⇩

Gradient vanishing (technical)

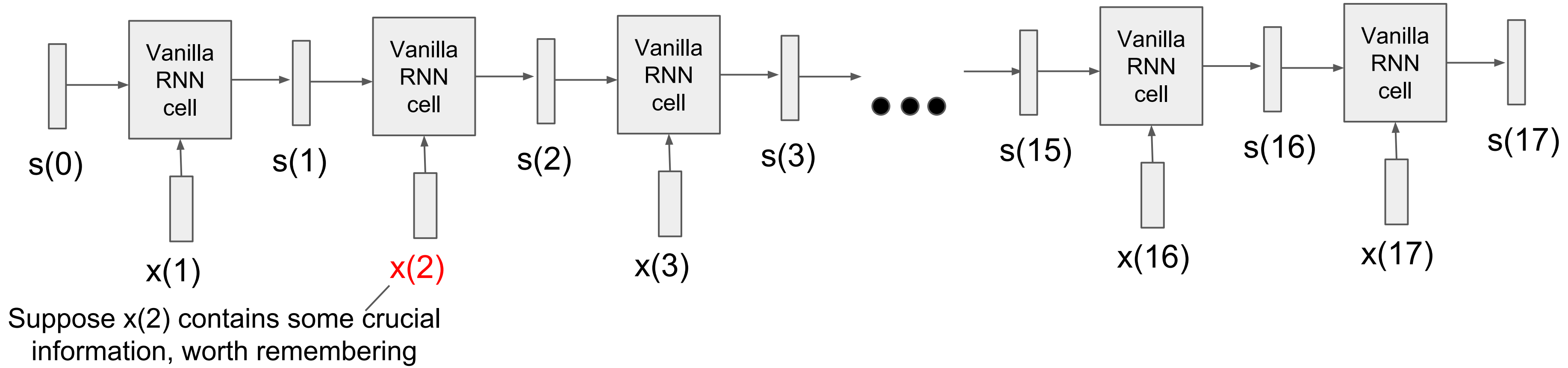
Information morphing

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



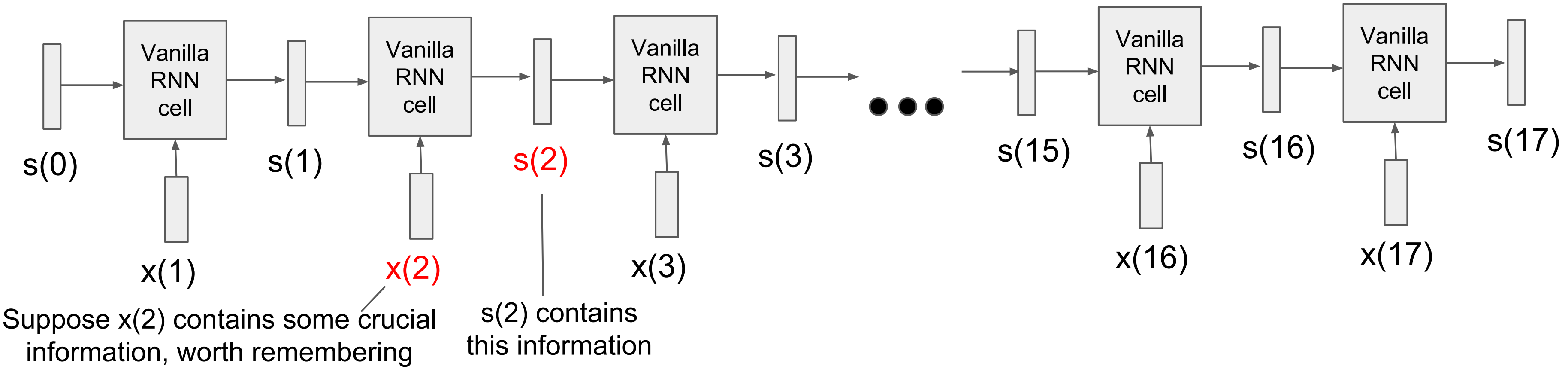
Information morphing

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



Information morphing

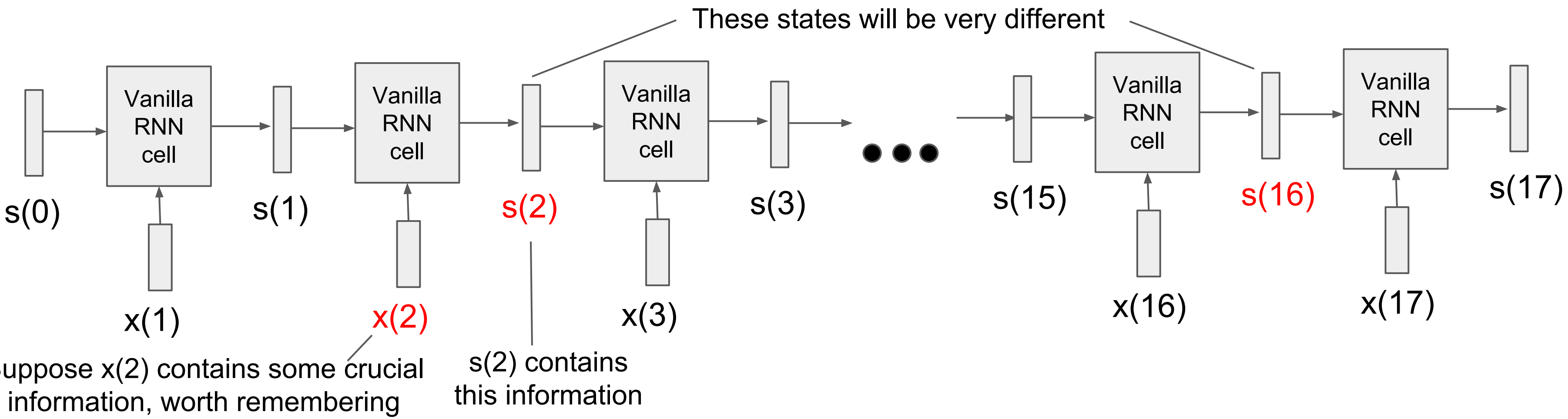
$$s_t = \varphi(W s_{t-1} + U x_t + b)$$



deepsystems.ai

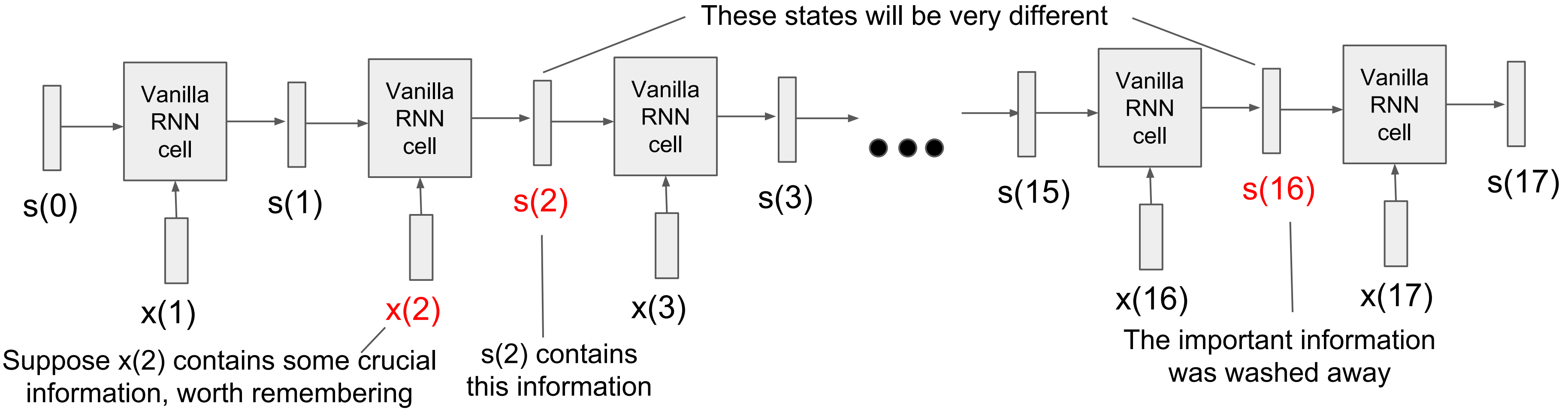
Information morphing

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



Information morphing

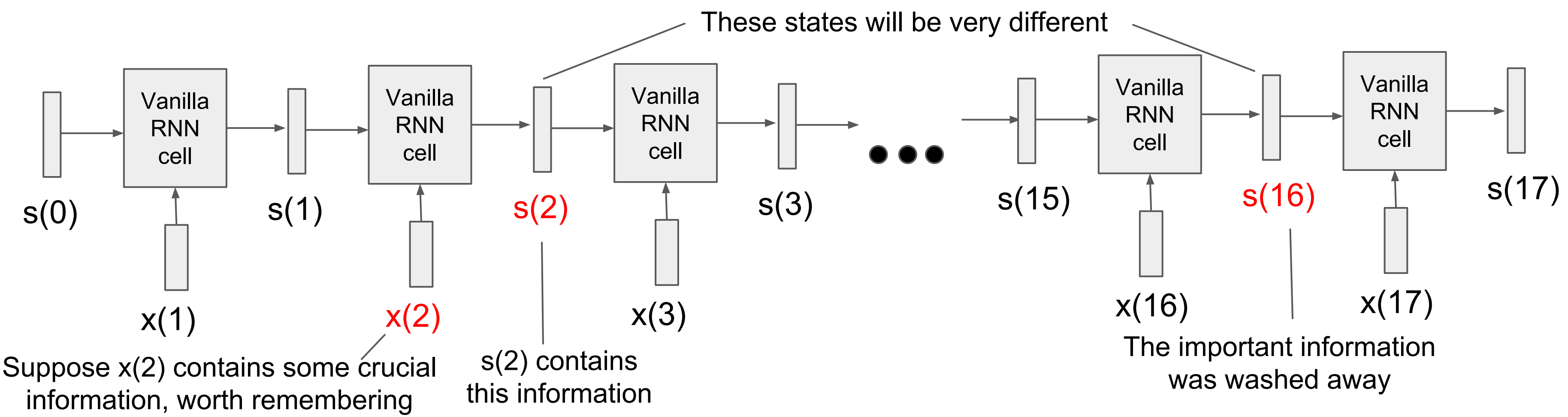
$$s_t = \varphi(W s_{t-1} + U x_t + b)$$



Information morphing

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

Why this happens?



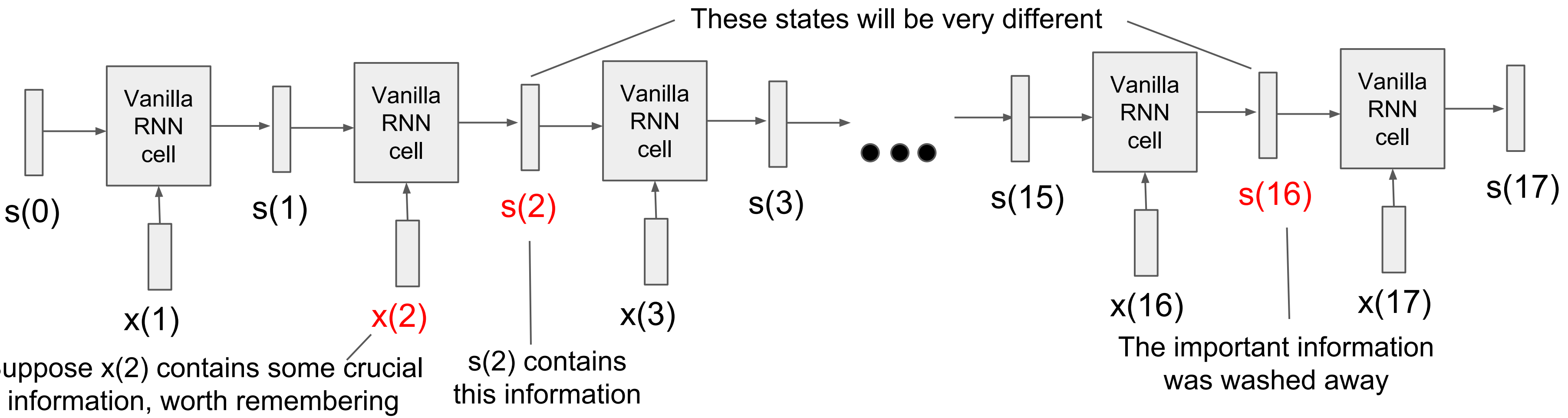
deepsystems.ai

Information morphing

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

Why this happens?

The rnn memory (state) should be protected: use only + or - operations to write to the memory



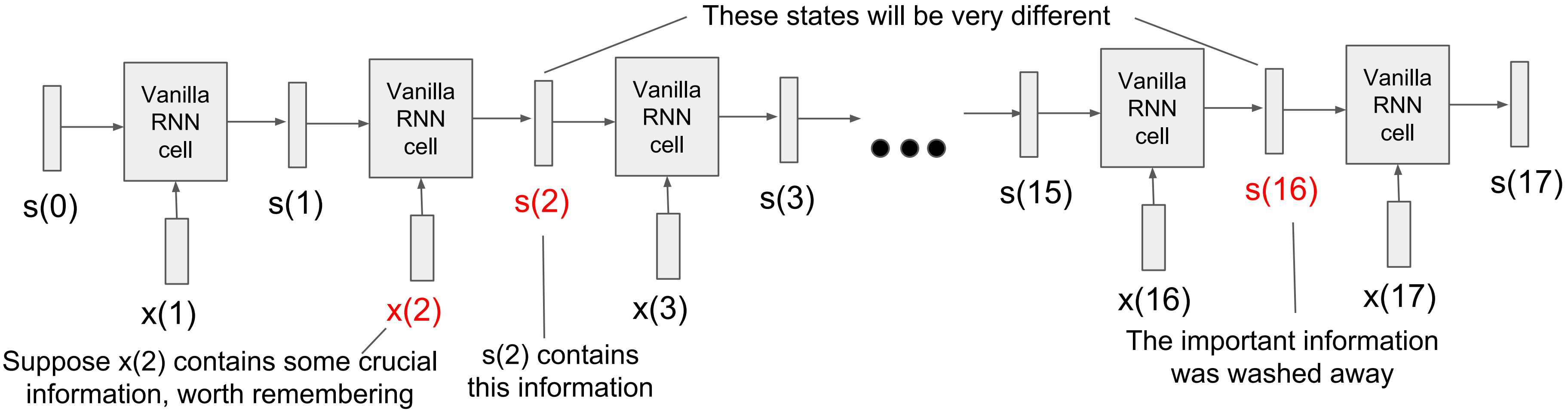
Information morphing

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

Nonlinearity is bad for long term memory

The rnn memory (state) should be protected: use only + or - operations to write to the memory

Why this happens?



Information morphing

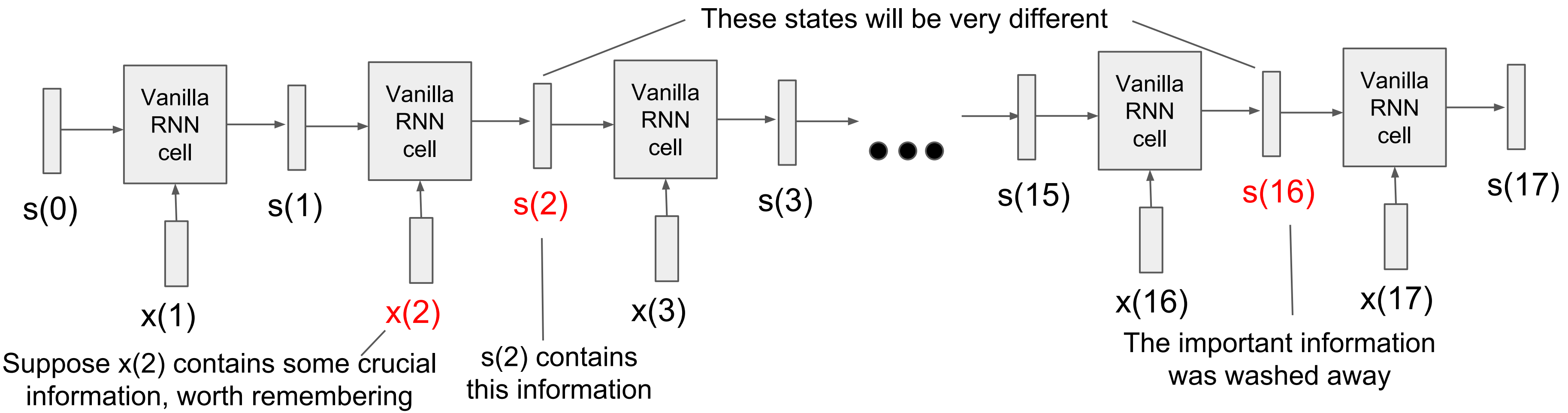
$$s_t = \varphi(W s_{t-1} + U x_t + b)$$

Why this happens?

Nonlinearity is bad for long term memory

The rnn memory (state) should be protected: use only + or - operations to write to the memory

Be selective: choose what to read, write and forget



Information morphing

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

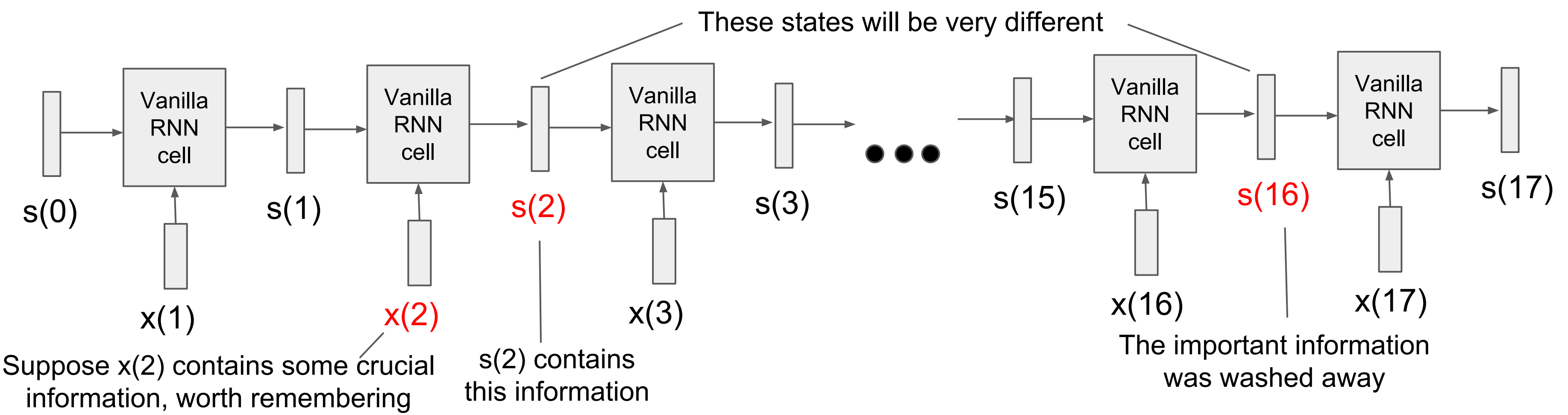
Nonlinearity is bad for long term memory

No selectivity (read all, overwrite all)

Why this happens?

The rnn memory (state) should be protected: use only + or - operations to write to the memory

Be selective: choose what to read, write and forget



deepsystems.ai

Information morphing

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

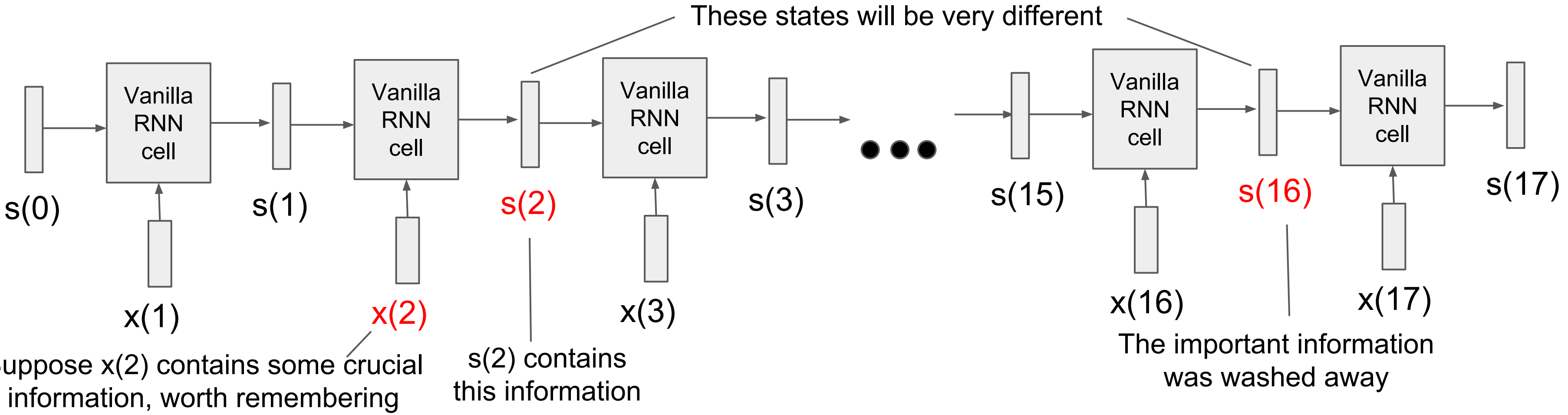
Why this happens?

Nonlinearity is bad for long term memory

No selectivity (read all, overwrite all)

The rnn memory (state) should be protected: use only + or - operations to write to the memory

Be selective: choose what to read, write and forget



Protecting the state & selectivity through gates

$$\begin{pmatrix}\text{Current}\\\text{state}\end{pmatrix} = \begin{pmatrix}\text{Previous}\\\text{state}\end{pmatrix} + \begin{pmatrix}\text{New}\\\text{state}\\\text{candidate}\end{pmatrix}$$

Protecting the state & selectivity through gates

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

Current state = Previous state + New state candidate

Protecting the state & selectivity through gates

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

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

Current state = Previous state + New state candidate

deepsystems.ai

Protecting the state & selectivity through gates

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

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

We don't corrupt the whole memory, just add or subtract something

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

Protecting the state & selectivity through gates

Current state = Previous state + New state candidate

Protecting the state & selectivity through gates

New state candidate = FUNC [Previous state , Current input]

Current state = Previous state + New state candidate

Protecting the state & selectivity through gates

New state candidate = FUNC [Previous state , Current input]

Current state = Previous state + New state candidate

Very important pattern - protecting the state

Protecting the state & selectivity through gates

(New state candidate) = FUNC [(Previous state) , (Current input)]

(Current state) = (Forget gate vector) ○ (Previous state) + ○ (New state candidate)

Protecting the state & selectivity through gates

$$\text{New state candidate} = \text{FUNC}\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Current state} = \text{Forget gate vector} \circ \text{Previous state} + \text{Write gate vector} \circ \text{New state candidate}$$

Protecting the state & selectivity through gates

$$\left(\begin{matrix}\text{New}\\\text{state}\\\text{candidate}\end{matrix}\right) = \text{FUNC}\left[\left(\begin{matrix}\text{Read}\\\text{gate}\\\text{vector}\end{matrix}\right) \circ \left(\begin{matrix}\text{Previous}\\\text{state}\end{matrix}\right), \left(\begin{matrix}\text{Current}\\\text{input}\end{matrix}\right)\right]$$

$$\left(\begin{matrix}\text{Current}\\\text{state}\end{matrix}\right) = \left(\begin{matrix}\text{Forget}\\\text{gate}\\\text{vector}\end{matrix}\right) \circ \left(\begin{matrix}\text{Previous}\\\text{state}\end{matrix}\right) + \left(\begin{matrix}\text{Write}\\\text{gate}\\\text{vector}\end{matrix}\right) \circ \left(\begin{matrix}\text{New}\\\text{state}\\\text{candidate}\end{matrix}\right)$$

Protecting the state & selectivity through gates

$$\begin{pmatrix}\text{Forget}\\ \text{gate}\\ \text{vector}\end{pmatrix} = G\_F\left[\begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix}, \begin{pmatrix}\text{Current}\\ \text{input}\end{pmatrix}\right]$$

$$\begin{pmatrix}\text{New}\\ \text{state}\\ \text{candidate}\end{pmatrix} = \text{FUNC}\left[\begin{pmatrix}\text{Read}\\ \text{gate}\\ \text{vector}\end{pmatrix} \circ \begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix}, \begin{pmatrix}\text{Current}\\ \text{input}\end{pmatrix}\right]$$

$$\begin{pmatrix}\text{Current}\\ \text{state}\end{pmatrix} = \begin{pmatrix}\text{Forget}\\ \text{gate}\\ \text{vector}\end{pmatrix} \circ \begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix} + \begin{pmatrix}\text{Write}\\ \text{gate}\\ \text{vector}\end{pmatrix} \circ \begin{pmatrix}\text{New}\\ \text{state}\\ \text{candidate}\end{pmatrix}$$

Protecting the state & selectivity through gates

$$\text{Write gate vector} = G\_W\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Forget gate vector} = G\_F\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{New state candidate} = \text{FUNC}\left[\text{Read gate vector} \circ \text{Previous state}, \text{Current input}\right]$$

$$\text{Current state} = \text{Forget gate vector} \circ \text{Previous state} + \text{Write gate vector} \circ \text{New state candidate}$$

Protecting the state & selectivity through gates

$$\text{Read gate vector} = G\_R\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Write gate vector} = G\_W\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Forget gate vector} = G\_F\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{New state candidate} = \text{FUNC}\left[\text{Read gate vector} \circ \text{Previous state}, \text{Current input}\right]$$

$$\text{Current state} = \text{Forget gate vector} \circ \text{Previous state} + \text{Write gate vector} \circ \text{New state candidate}$$

Protecting the state & selectivity through gates

(Read gate vector) = G_R [(Previous state), (Current input)]

(Write gate vector) = G_W [(Previous state), (Current input)]

(Forget gate vector) = G_F [(Previous state), (Current input)]

Single layer network

(New state candidate) = FUNC [(Read gate vector) ○ (Previous state), (Current input)]

(Current state) = (Forget gate vector) ○ (Previous state) + (Write gate vector) ○ (New state candidate)

Protecting the state & selectivity through gates

(Read gate vector) = G_R [(Previous state), (Current input)]

(Write gate vector) = G_W [(Previous state), (Current input)]

(Forget gate vector) = G_F [(Previous state), (Current input)]

Single layer network

What kind of activation function are used?

(New state candidate) = FUNC [(Read gate vector) ○ (Previous state), (Current input)]

(Current state) = (Forget gate vector) ○ (Previous state) + (Write gate vector) ○ (New state candidate)

Protecting the state & selectivity through gates

Read gate vector = G_R [Previous state , Current input]

Write gate vector = G_W [Previous state , Current input]

Forget gate vector = G_F [Previous state , Current input]

sigmoid

sigmoid

sigmoid

Single layer network

What kind of activation function are used?

New state candidate = FUNC [Read gate vector ○ Previous state , Current input]

Current state = Forget gate vector ○ Previous state + Write gate vector ○ New state candidate

Protecting the state & selectivity through gates

Read gate vector = G_R [Previous state, Current input]

Write gate vector = G_W [Previous state, Current input]

Forget gate vector = G_F [Previous state, Current input]

sigmoid

sigmoid

sigmoid

Single layer network

What kind of activation function are used?

tanh

New state candidate = FUNC [Read gate vector ∘ Previous state, Current input]

Current state = Forget gate vector ∘ Previous state + Write gate vector ∘ New state candidate

Protecting the state & selectivity through gates

$$\begin{pmatrix}\text{Read}\\ \text{gate}\\ \text{vector}\end{pmatrix} = G\_R\left[\begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix}, \begin{pmatrix}\text{Current}\\ \text{input}\end{pmatrix}\right] \qquad \begin{pmatrix}\text{Write}\\ \text{gate}\\ \text{vector}\end{pmatrix} = G\_W\left[\begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix}, \begin{pmatrix}\text{Current}\\ \text{input}\end{pmatrix}\right] \qquad \begin{pmatrix}\text{Forget}\\ \text{gate}\\ \text{vector}\end{pmatrix} = G\_F\left[\begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix}, \begin{pmatrix}\text{Current}\\ \text{input}\end{pmatrix}\right]$$

$$\begin{pmatrix}\text{New}\\ \text{state}\\ \text{candidate}\end{pmatrix} = \text{FUNC}\left[\begin{pmatrix}\text{Read}\\ \text{gate}\\ \text{vector}\end{pmatrix} \circ \begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix}, \begin{pmatrix}\text{Current}\\ \text{input}\end{pmatrix}\right]$$

$$\begin{pmatrix}\text{Current}\\ \text{state}\end{pmatrix} = \begin{pmatrix}\text{Forget}\\ \text{gate}\\ \text{vector}\end{pmatrix} \circ \begin{pmatrix}\text{Previous}\\ \text{state}\end{pmatrix} + \begin{pmatrix}\text{Write}\\ \text{gate}\\ \text{vector}\end{pmatrix} \circ \begin{pmatrix}\text{New}\\ \text{state}\\ \text{candidate}\end{pmatrix}$$

Protecting the state & selectivity through gates

$$\text{Read gate vector} = G\_R\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Write gate vector} = G\_W\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Forget gate vector} = G\_F\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{New state candidate} = \text{FUNC}\left[\text{Read gate vector} \circ \text{Previous state}, \text{Current input}\right]$$

$$\text{Current state} = \text{Forget gate vector} \circ \text{Previous state} + \text{Write gate vector} \circ \text{New state candidate}$$

Protecting the state & selectivity through gates

(Read gate vector) = G_R [(Previous state), (Current input)]

(Write gate vector) = G_W [(Previous state), (Current input)]

(Forget gate vector) = G_F [(Previous state), (Current input)]

(New state candidate) = FUNC [(**Read gate vector**) ∘ (**Previous state**), (Current input)]

(Current state) = (Forget gate vector) ∘ (Previous state) + (Write gate vector) ∘ (New state candidate)

Protecting the state & selectivity through gates

(Read gate vector) = G_R[(Previous state), (Current input)]

(Write gate vector) = G_W[(Previous state), (Current input)]

(Forget gate vector) = G_F[(Previous state), (Current input)]

(New state candidate) = FUNC[(**Read gate vector**) $\circ$ (**Previous state**), (**Current input**)]

(Current state) = (Forget gate vector) $\circ$ (Previous state) + (Write gate vector) $\circ$ (New state candidate)

Protecting the state & selectivity through gates

$$\text{Read gate vector} = G\_R\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Write gate vector} = G\_W\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Forget gate vector} = G\_F\left[\text{Previous state}, \text{Current input}\right]$$

$$\textbf{New state candidate} = \text{FUNC}\left[\textbf{Read gate vector} \circ \textbf{Previous state}, \textbf{Current input}\right]$$

$$\text{Current state} = \text{Forget gate vector} \circ \text{Previous state} + \text{Write gate vector} \circ \text{New state candidate}$$

Protecting the state & selectivity through gates

$$\text{Read gate vector} = G\_R\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Write gate vector} = G\_W\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{Forget gate vector} = G\_F\left[\text{Previous state}, \text{Current input}\right]$$

$$\text{New state candidate} = \text{FUNC}\left[\text{Read gate vector} \circ \text{Previous state}, \text{Current input}\right]$$

$$\text{Current state} = \text{Forget gate vector} \circ \text{Previous state} + \textbf{Write gate vector} \circ \textbf{New state candidate}$$

Protecting the state & selectivity through gates

Read gate vector = G_R [Previous state , Current input]

Write gate vector = G_W [Previous state , Current input]

Forget gate vector = G_F [Previous state , Current input]

New state candidate = FUNC [Read gate vector $\circ$ Previous state , Current input]

Current state = **Forget gate vector** $\circ$ **Previous state** + **Write gate vector** $\circ$ **New state candidate**

Protecting the state & selectivity through gates

(Read gate vector) = G_R [(Previous state), (Current input)]

(Write gate vector) = G_W [(Previous state), (Current input)]

(Forget gate vector) = G_F [(Previous state), (Current input)]

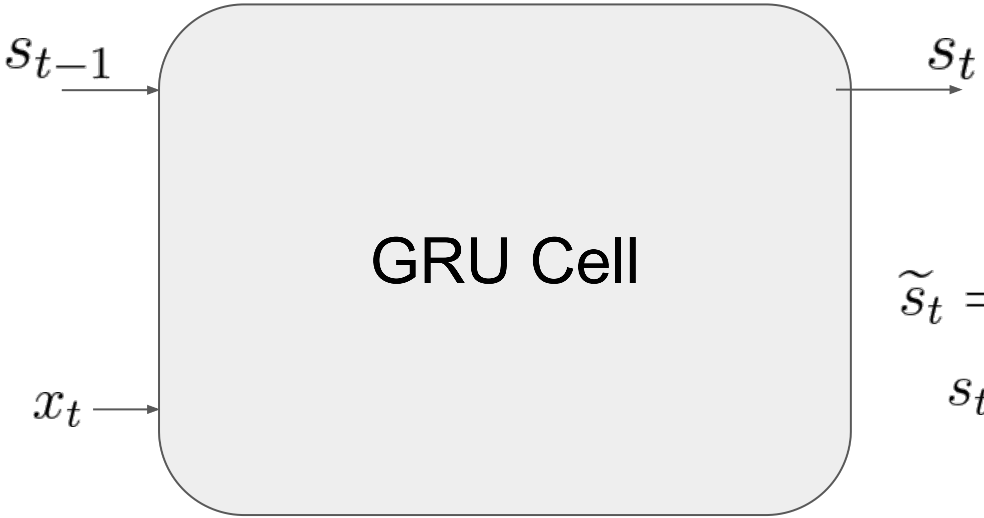
(New state candidate) = FUNC [(Read gate vector) ∘ (Previous state), (Current input)]

(Current state) = (Forget gate vector) ∘ (Previous state) + (Write gate vector) ∘ (New state candidate)

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

GRU Cell step by step



GRU equations

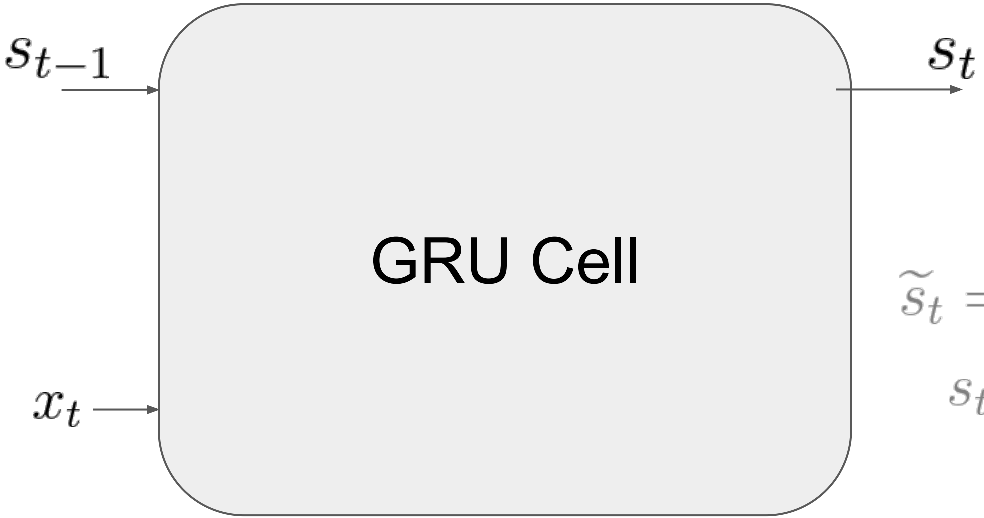
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r) \quad \text{Read gate}$$

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

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

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

GRU Cell step by step



GRU equations

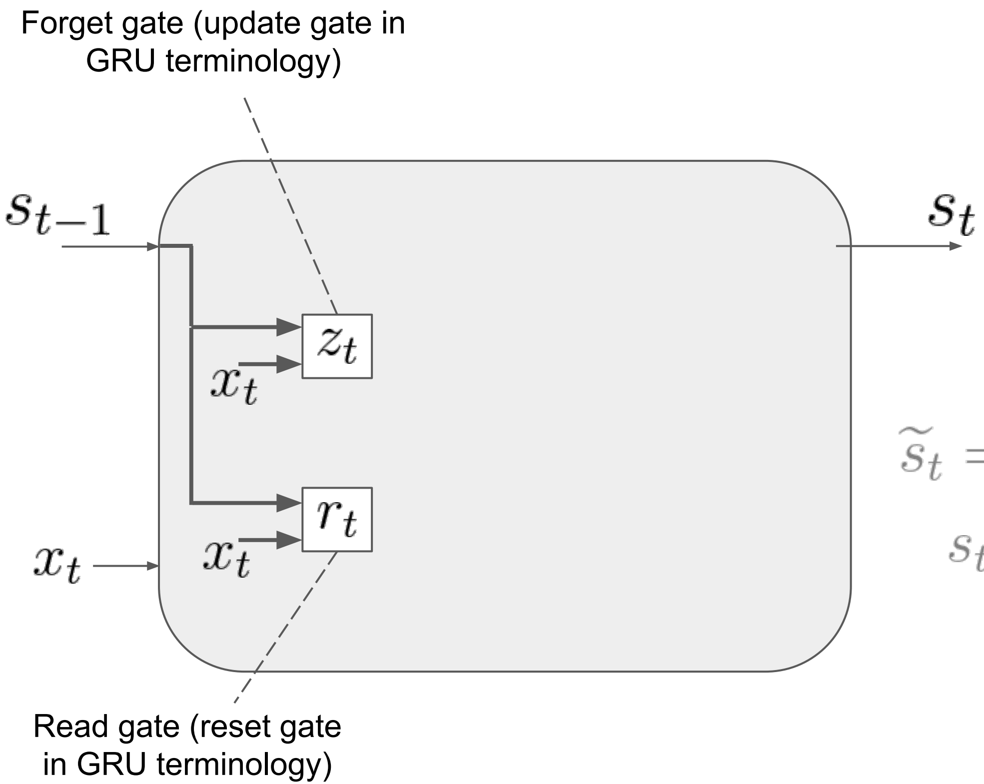
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

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

GRU Cell step by step



GRU equations

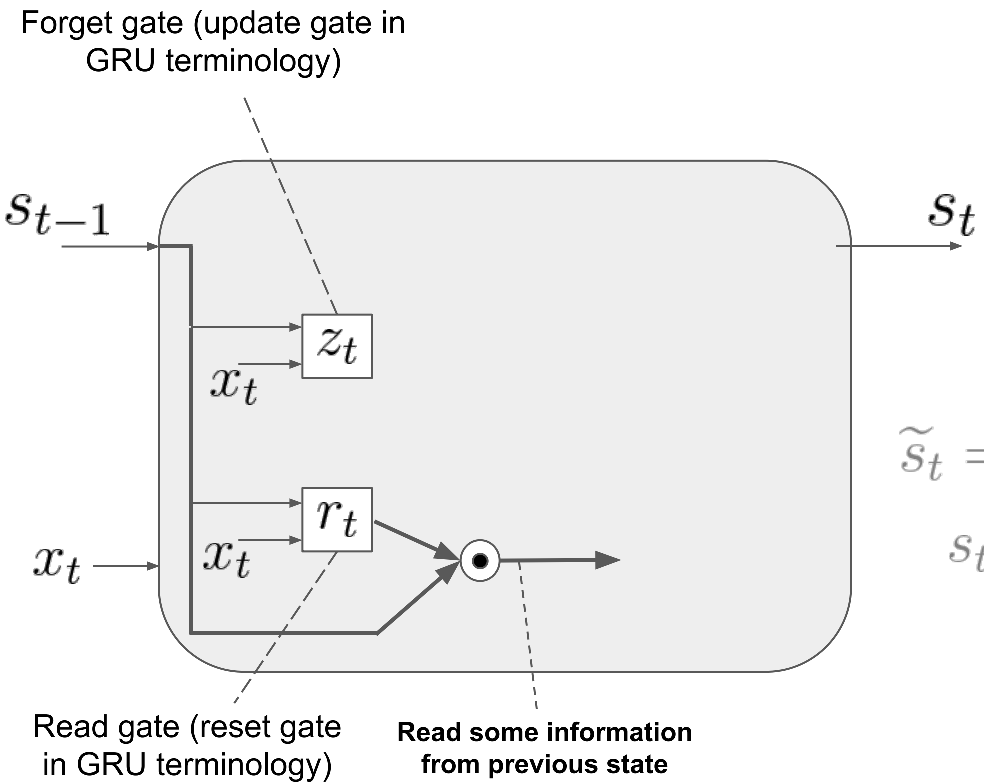
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

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

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

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

GRU Cell step by step



GRU equations

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

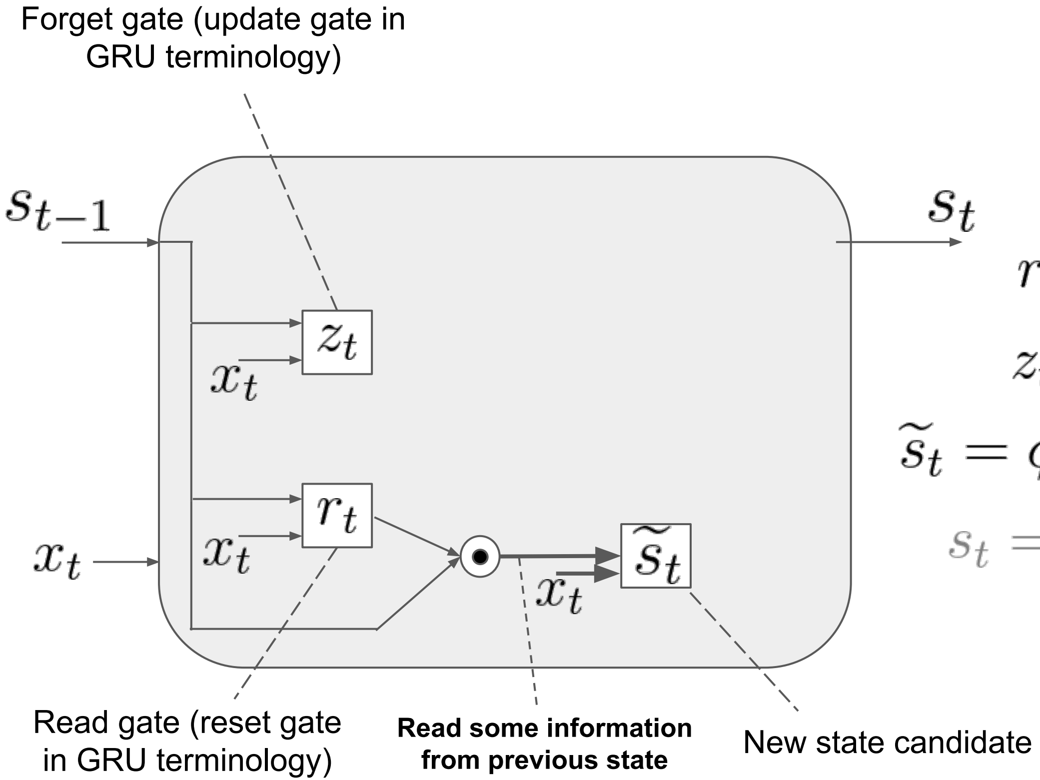
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

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

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

deepsystems.ai

GRU Cell step by step



GRU equations

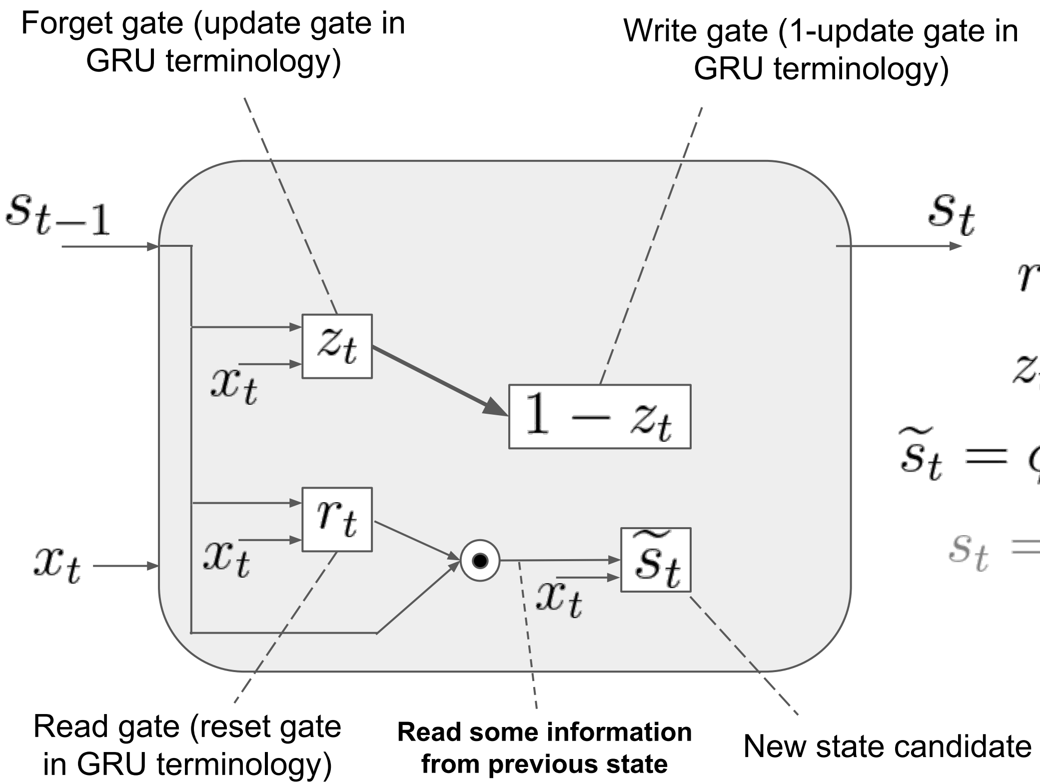
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

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

GRU Cell step by step



GRU equations

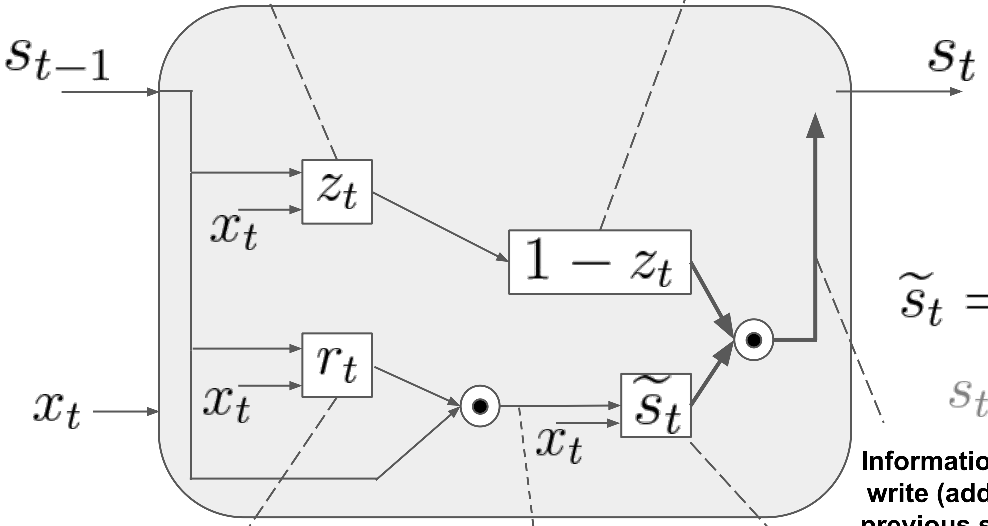
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

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

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

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

GRU Cell step by step



GRU equations

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

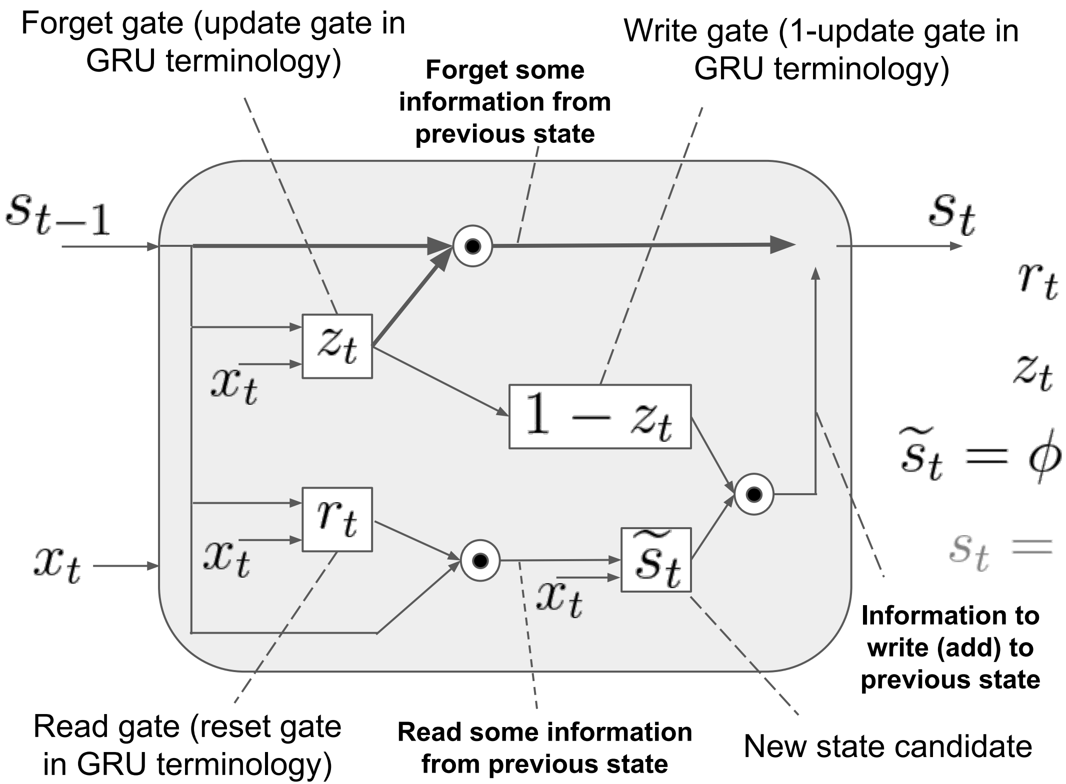
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

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

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

deepsystems.ai

GRU Cell step by step



GRU equations

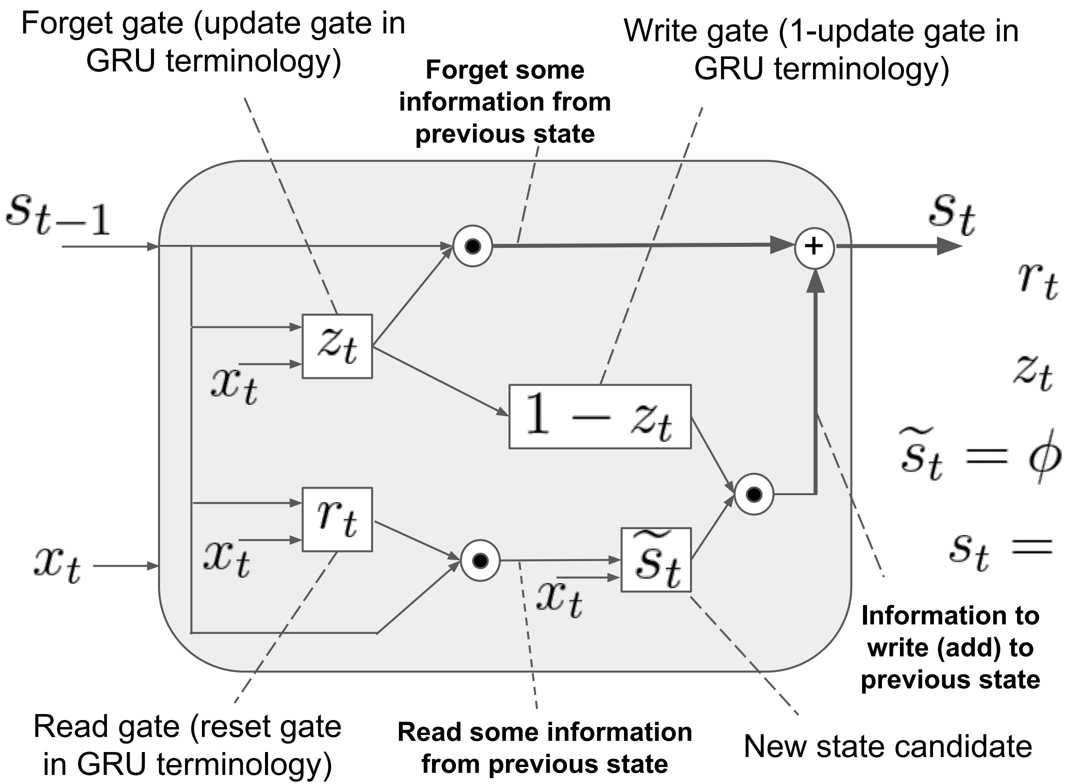
$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

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

GRU Cell step by step



GRU equations

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

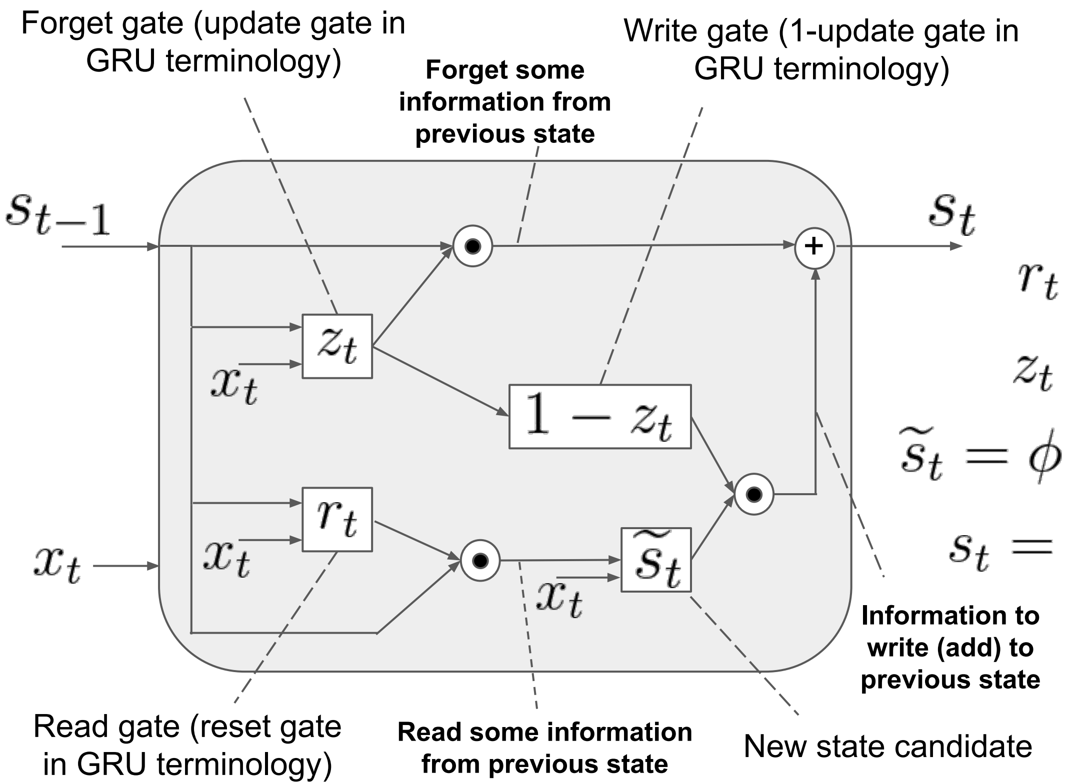
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z) \quad \text{Forget gate}$$

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

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

deepsystems.ai

GRU Cell step by step



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

GRU Cell step by step

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

GRU Cell step by step

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

GRU Cell step by step

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

$$\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ s_{t-1}^3 \end{pmatrix}$$

GRU Cell step by step

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

$$\begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ s_{t-1}^3 \end{pmatrix} \qquad \begin{pmatrix} \widetilde{s}_t^1 \\ \widetilde{s}_t^2 \\ \widetilde{s}_t^3 \end{pmatrix}$$

deepsystems.ai

GRU Cell step by step

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

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

Update gate = vector of zeros

GRU Cell step by step

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

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

Update gate = vector of zeros

GRU Cell step by step

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

GRU Cell step by step

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**

GRU Cell step by step

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

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

Update gate = vector of ones

GRU Cell step by step

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

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

Update gate = vector of ones

GRU Cell step by step

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

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

Update gate = vector of ones

GRU Cell step by step

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

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

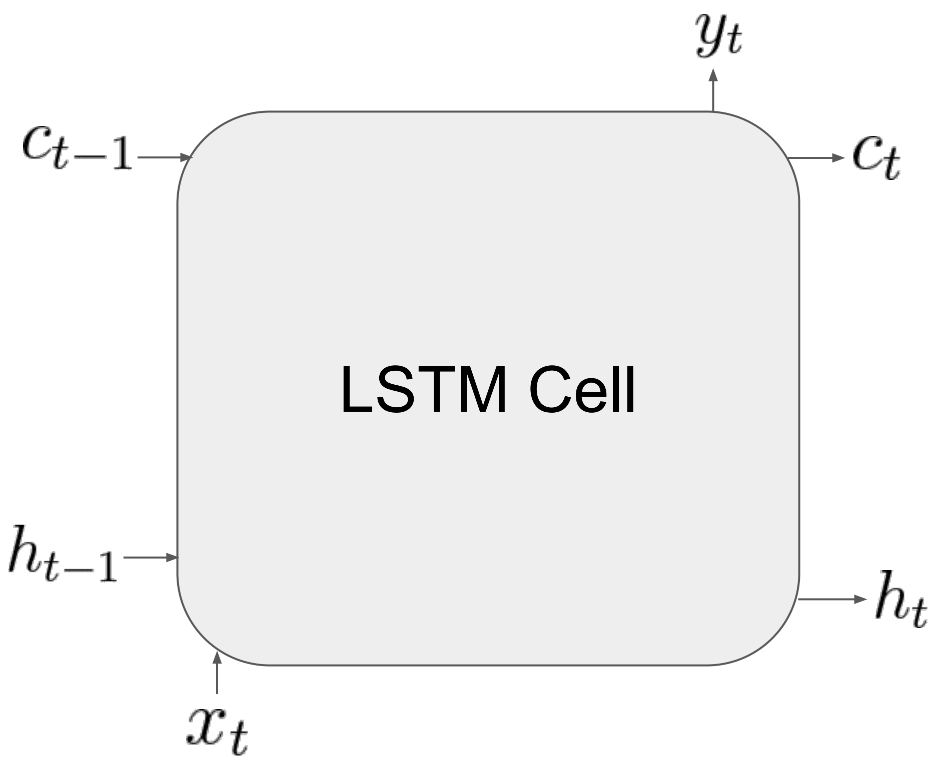
Completely ignore current input and state candidate

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

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

LSTM Cell

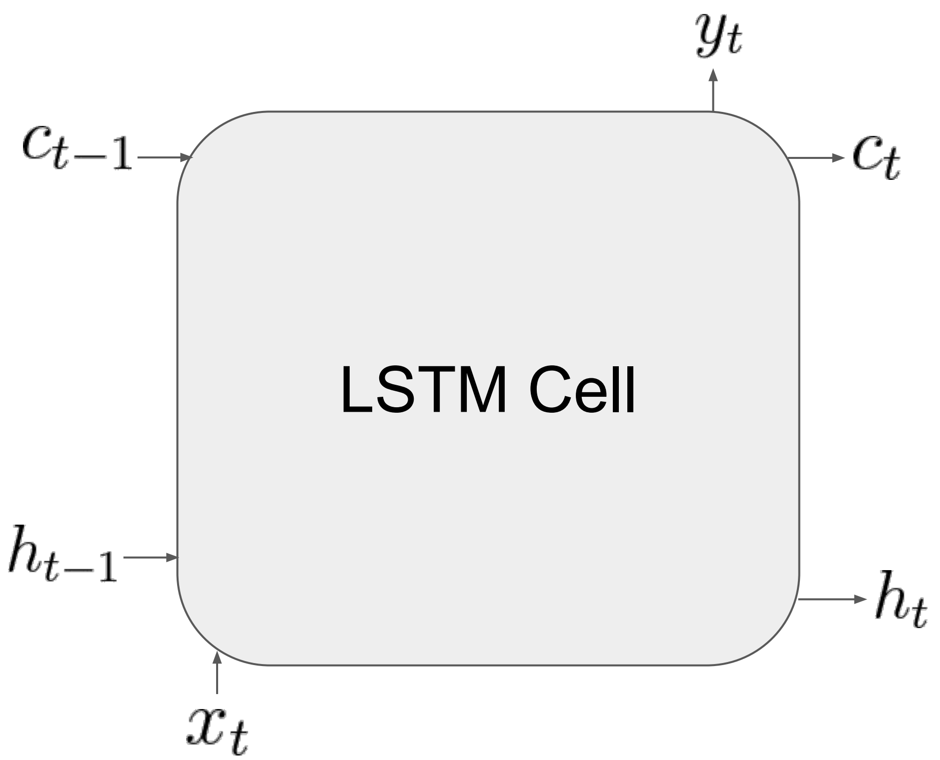


LSTM Cell

The state is a pair of vectors:

c_t — Memory Cell

h_t — Shadow State (gated version of memory cell)



LSTM Cell

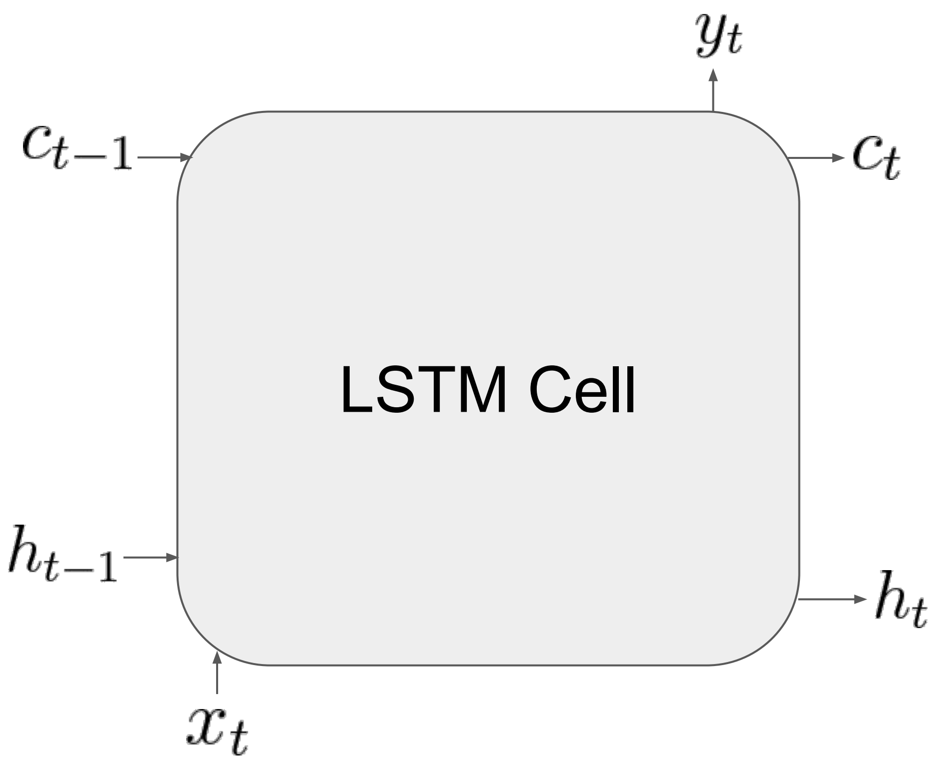
The state is a pair of vectors:

c_t — Memory Cell

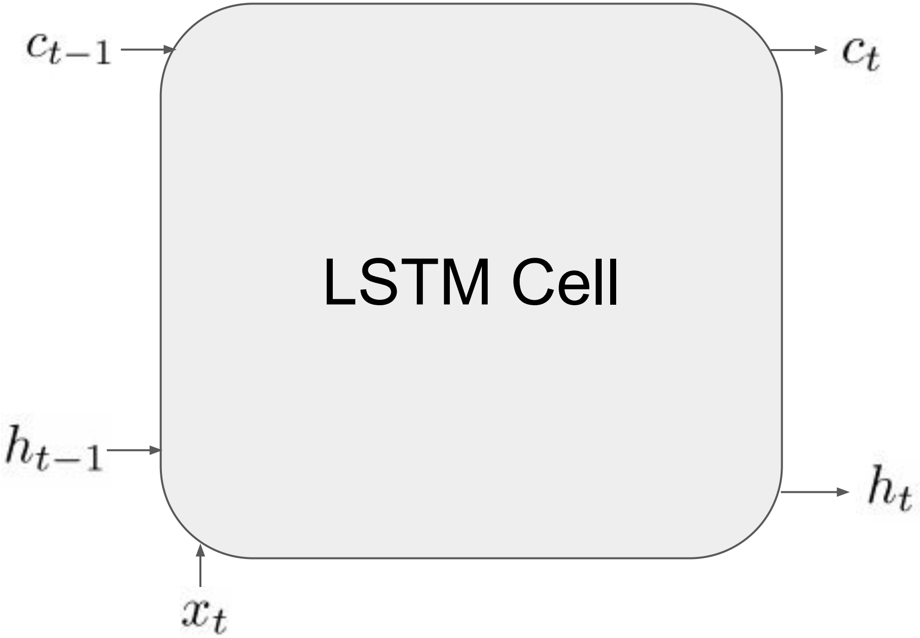
h_t — Shadow State (gated version of memory cell)

The output is a part of the state:

$y_t = h_t$ — Cell output



LSTM Cell



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

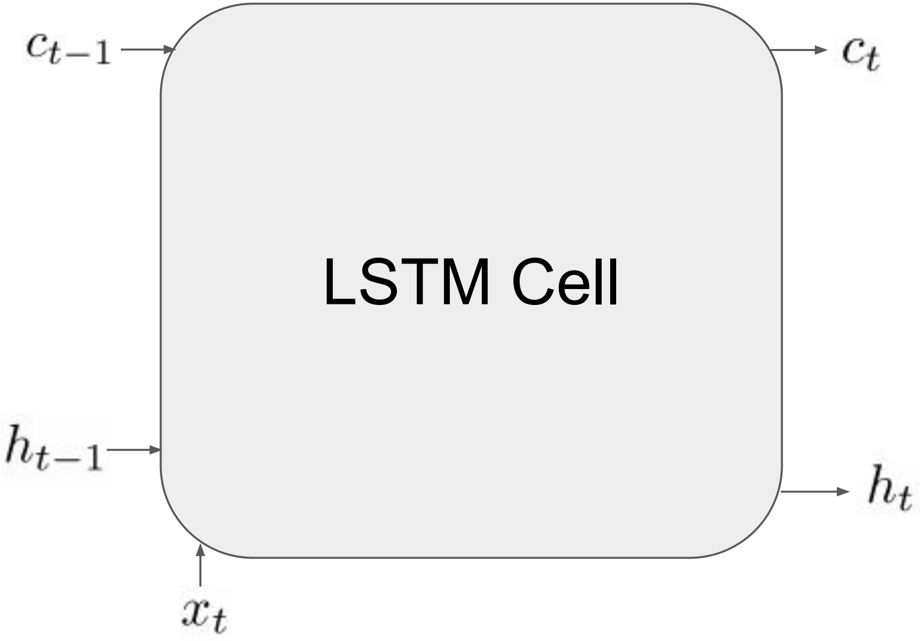
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

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

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

$$y_t = h_t$$ Cell Output

LSTM Cell



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

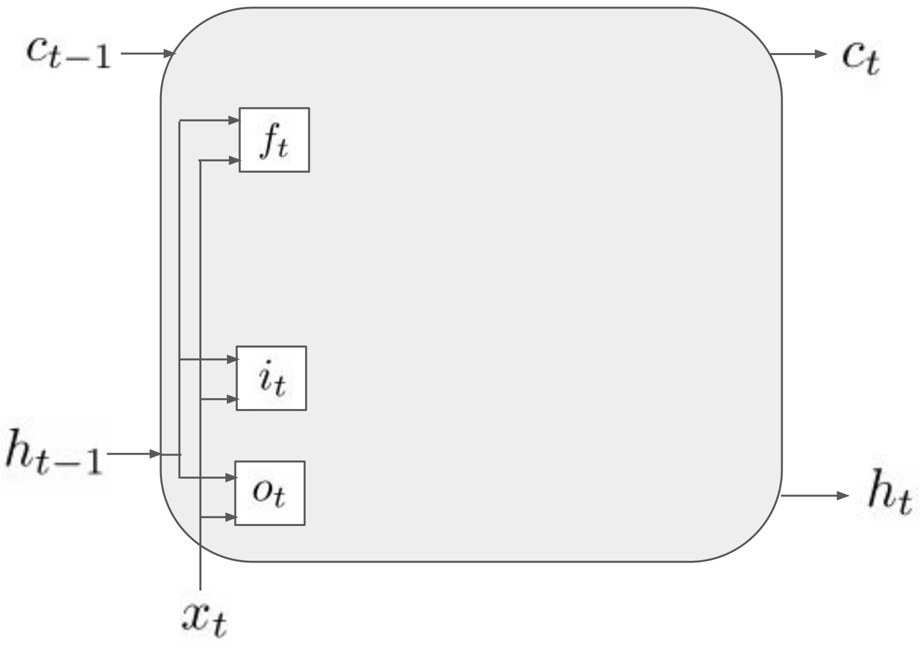
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

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

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

$$y_t = h_t$$ Cell Output

LSTM Cell



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

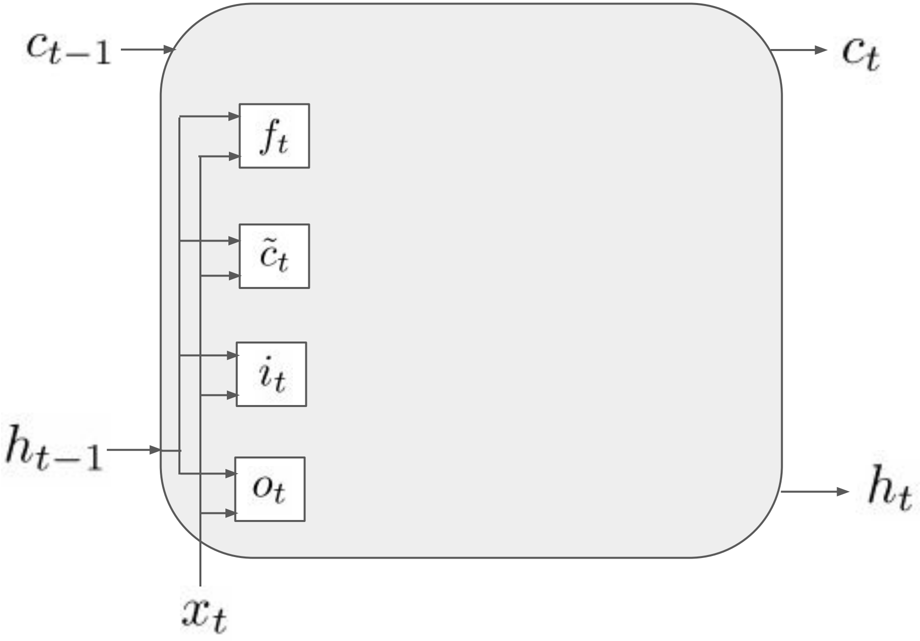
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell



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

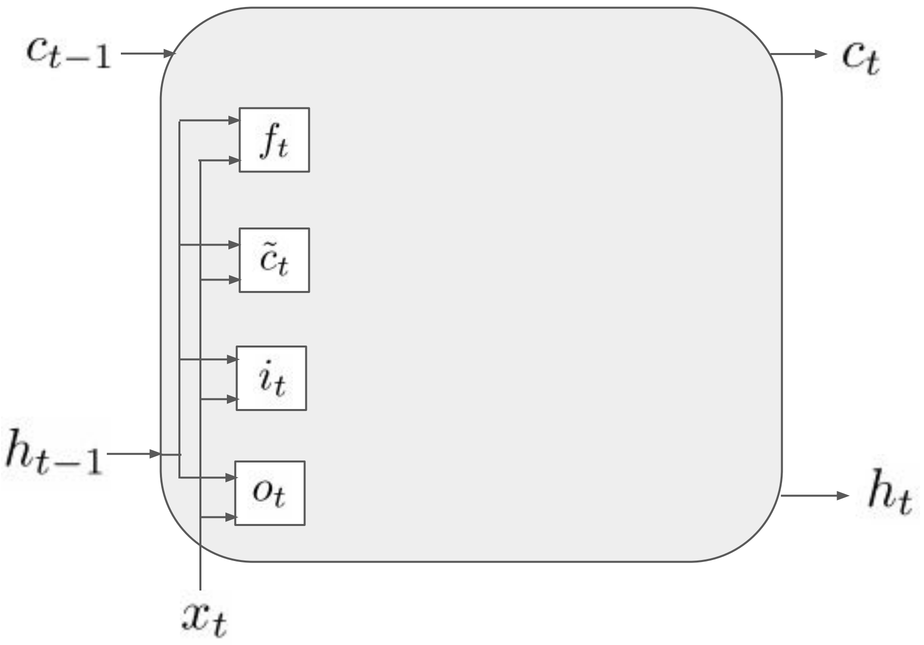
$h_t = o_t \circ tanh(c_t)$ Shadow state

$y_t = h_t$ Cell Output

deepsystems.ai

LSTM Cell

If h_{t-1} is a gated version of a memory content, where is a potential problem?



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

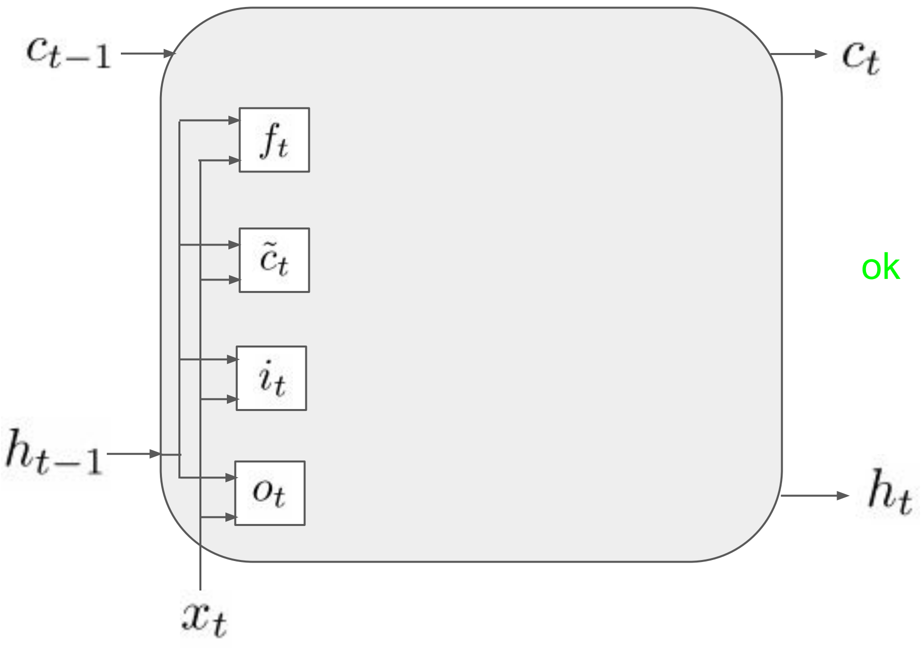
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

LSTM Cell

If h_{t-1} is a gated version of a memory content, where is a potential problem?



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

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

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

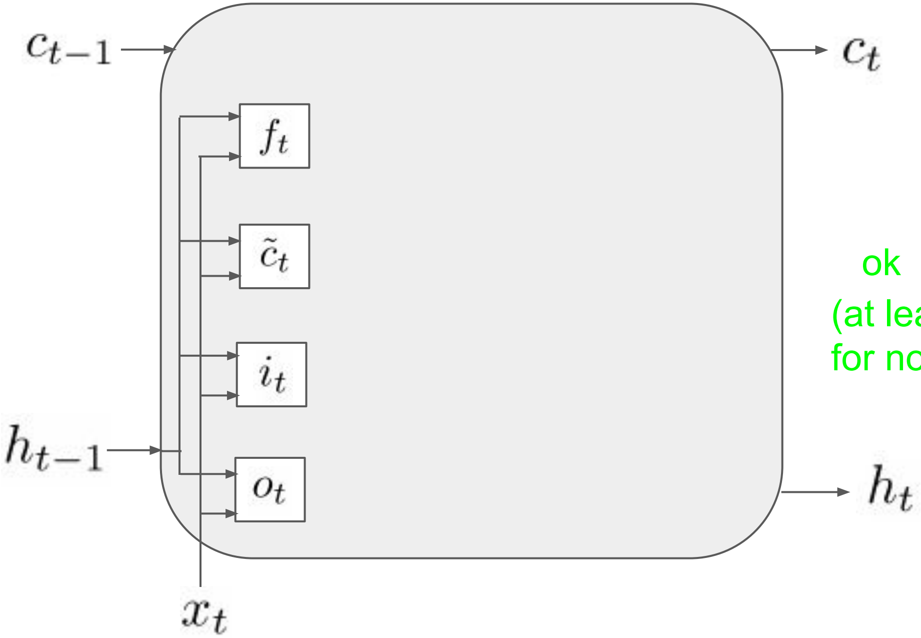
$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell

If h_{t-1} is a gated version of a memory content, where is a potential problem?



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

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

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

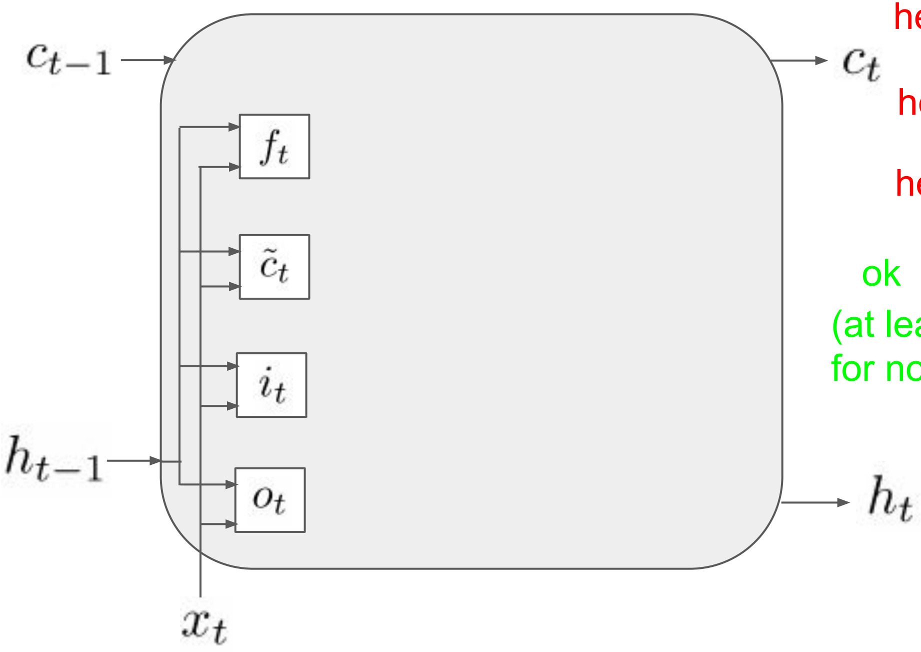
$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell

If h_{t-1} is a gated version of a memory content, where is a potential problem?



LSTM equations

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

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

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

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

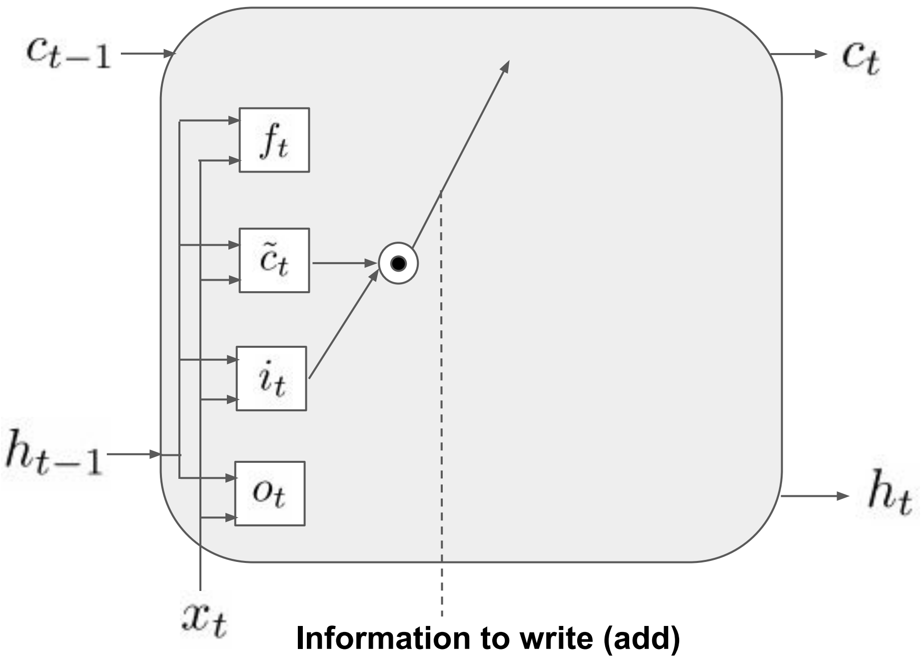
$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$ Memory cell

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

$y_t = h_t$ Cell Output

deepsystems.ai

LSTM Cell



Information to write (add) to previous state

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

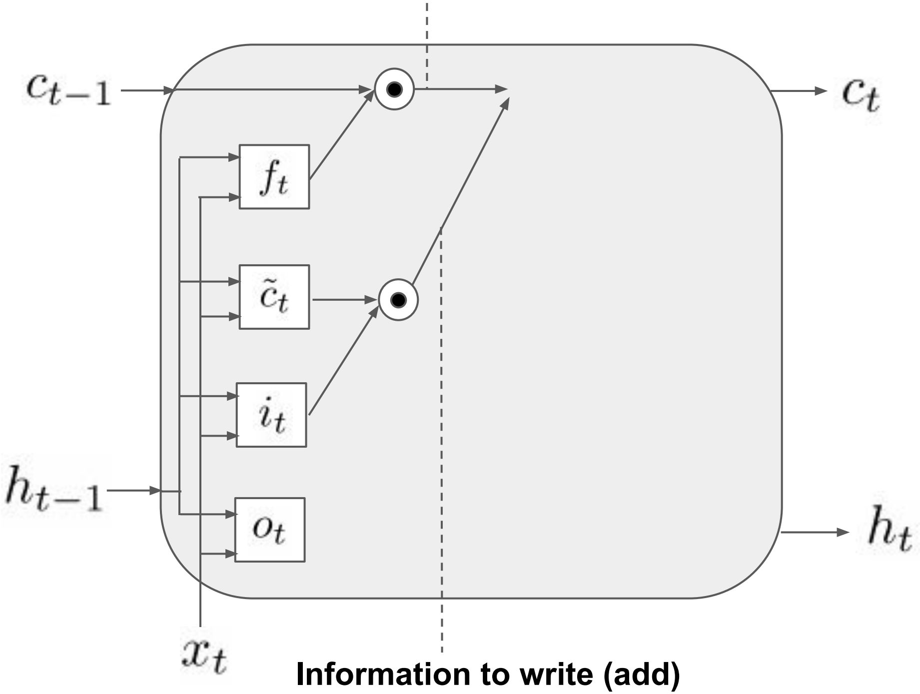
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

LSTM Cell

Forget some information from previous state



Information to write (add) to previous state

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

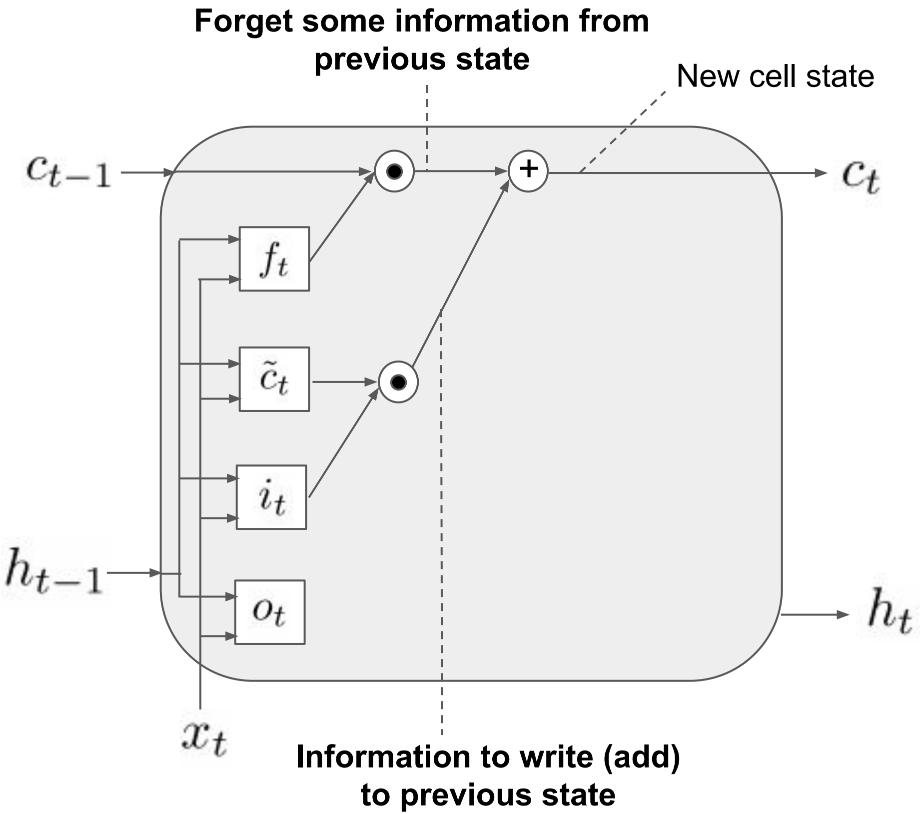
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell



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

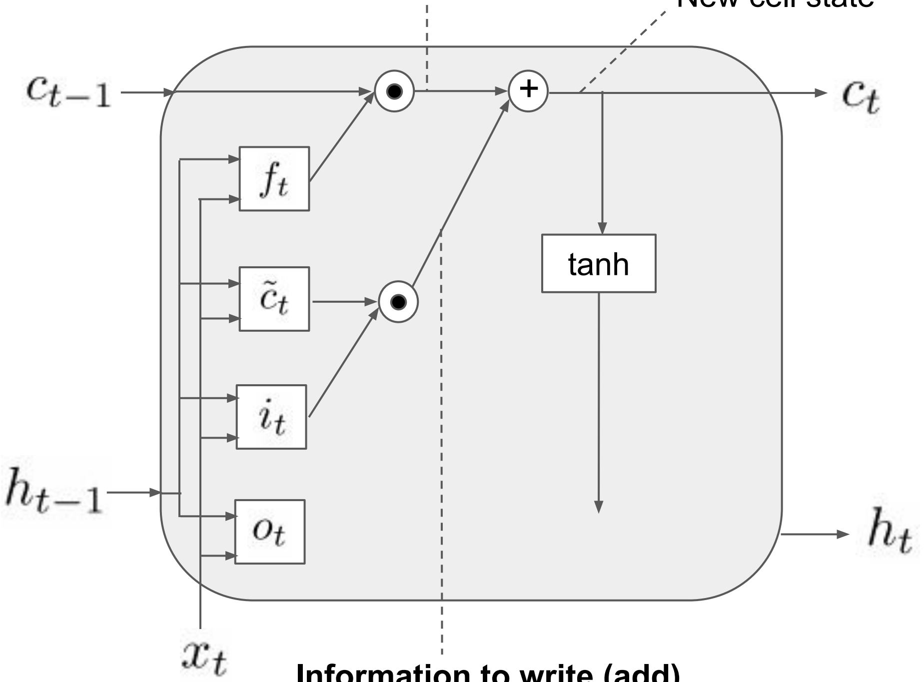
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell



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

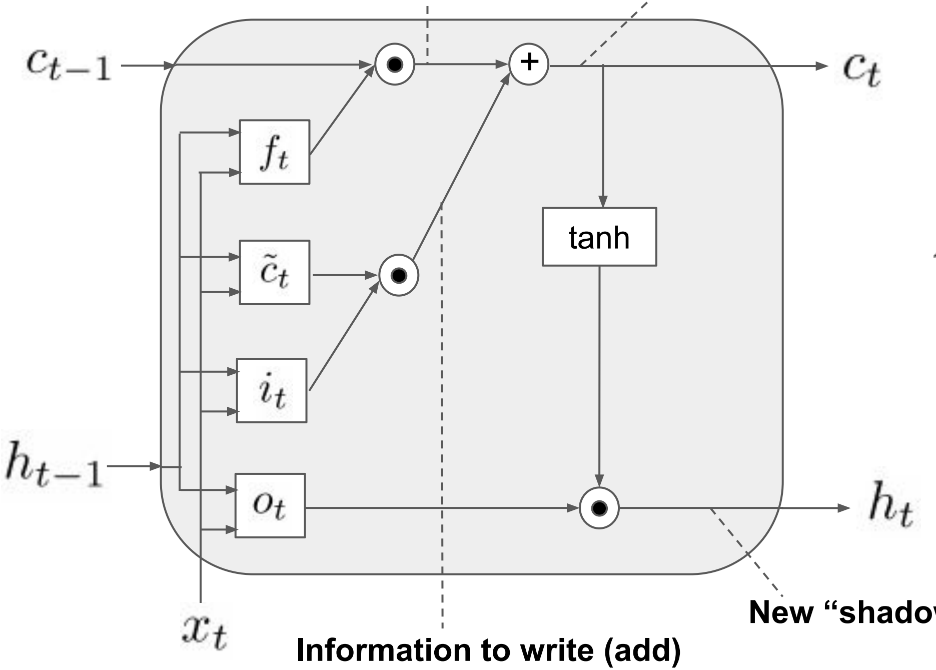
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

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

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

$$y_t = h_t$$ Cell Output

LSTM Cell



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

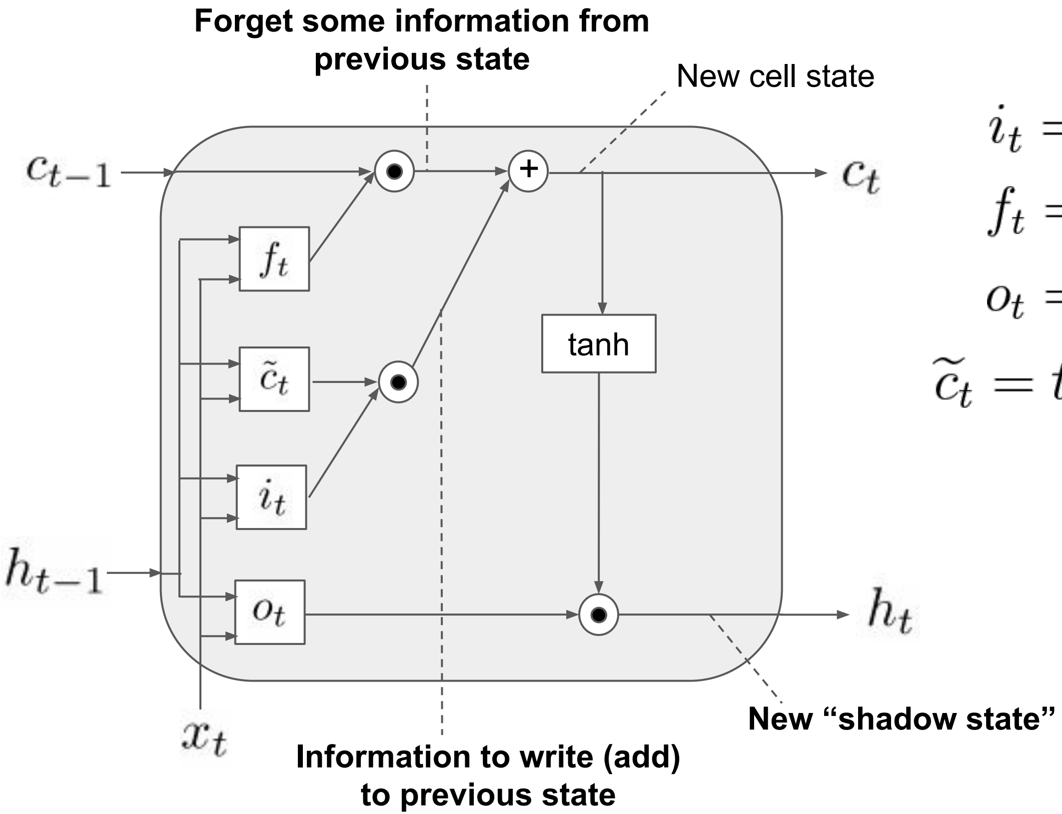
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell



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

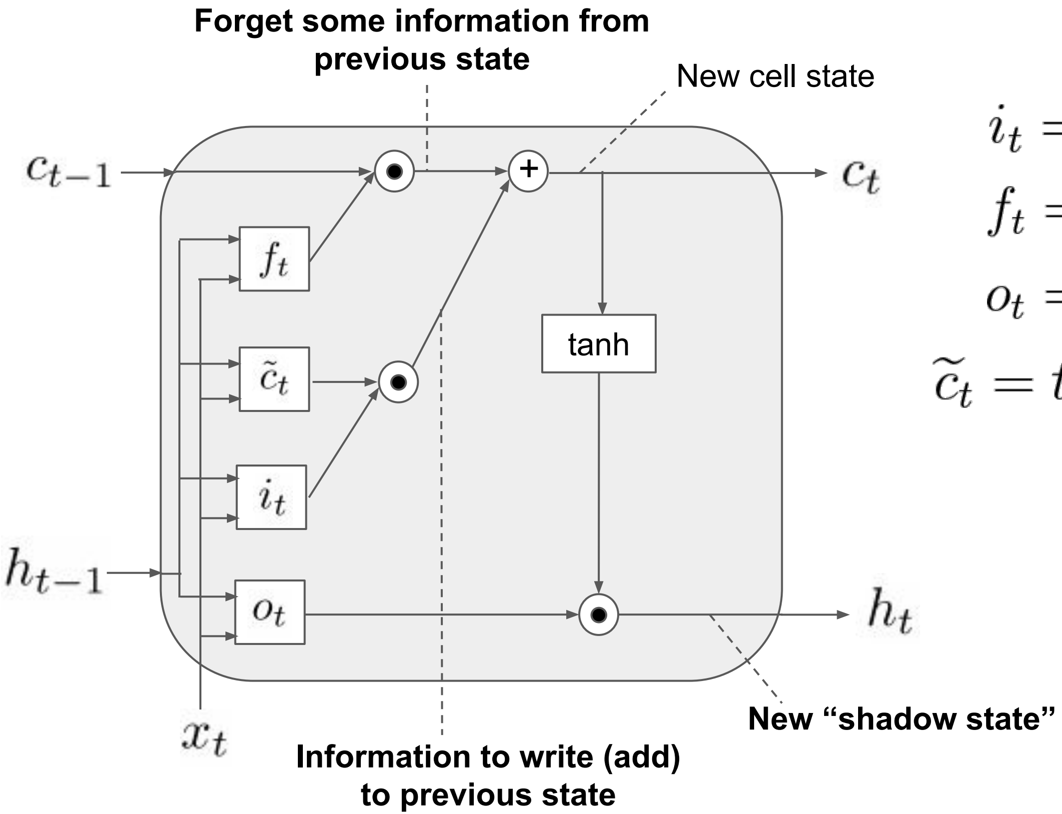
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

deepsystems.ai

LSTM Cell



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

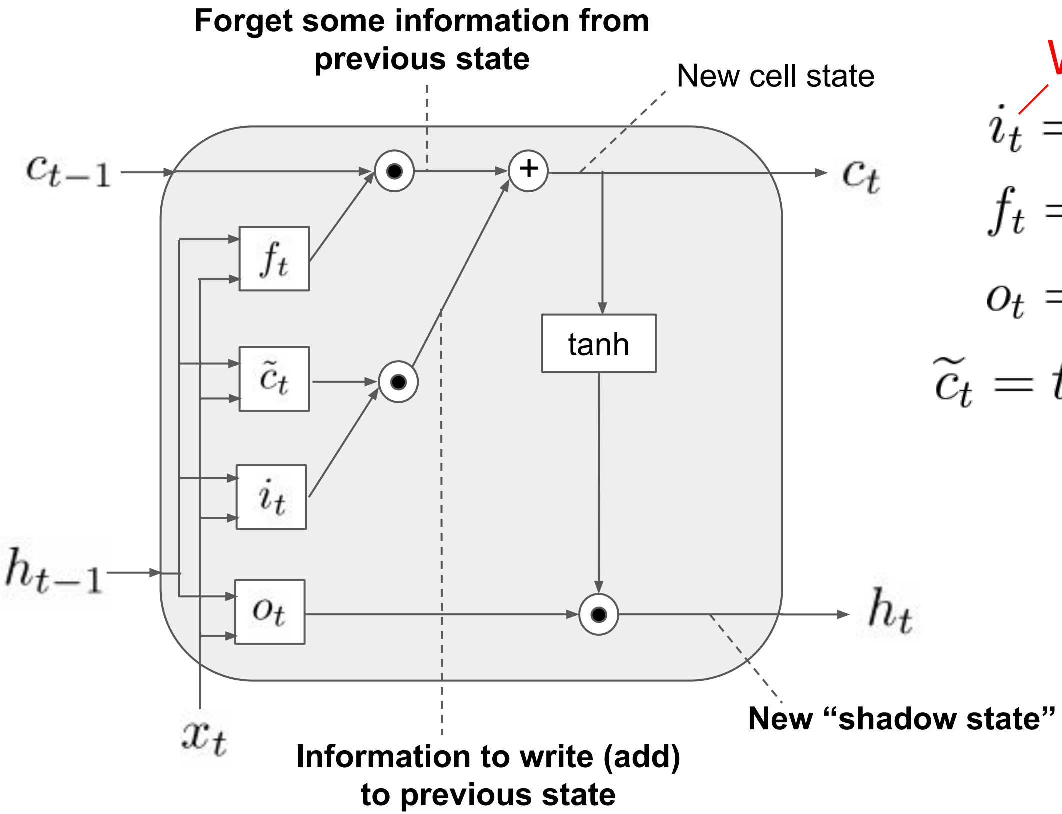
$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

Where are Read/Write gates?

deepsystems.ai

LSTM Cell



LSTM equations

Write 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

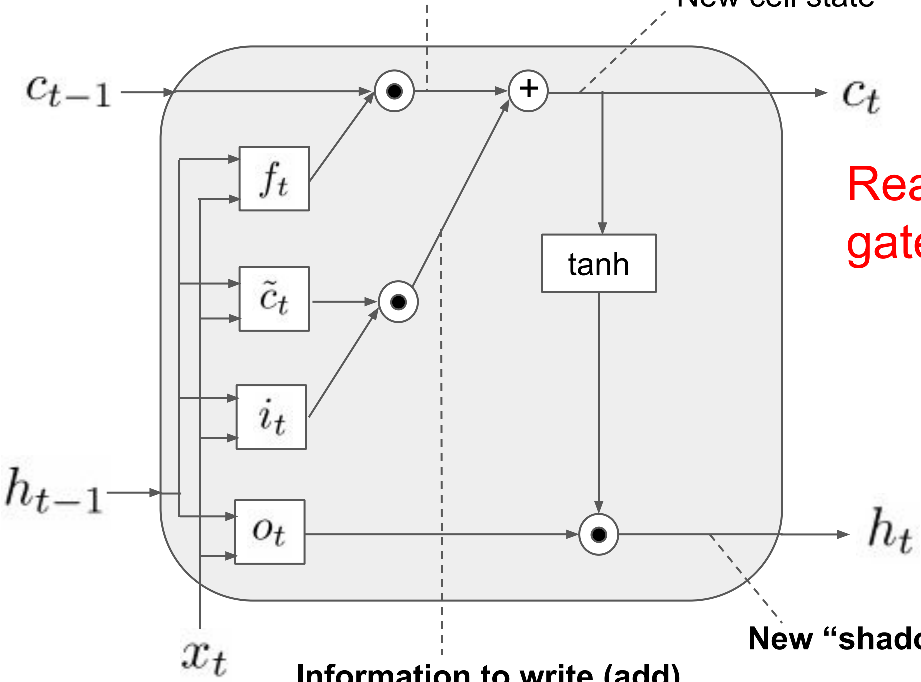
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

Where are Read/Write gates?

LSTM Cell



LSTM equations

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

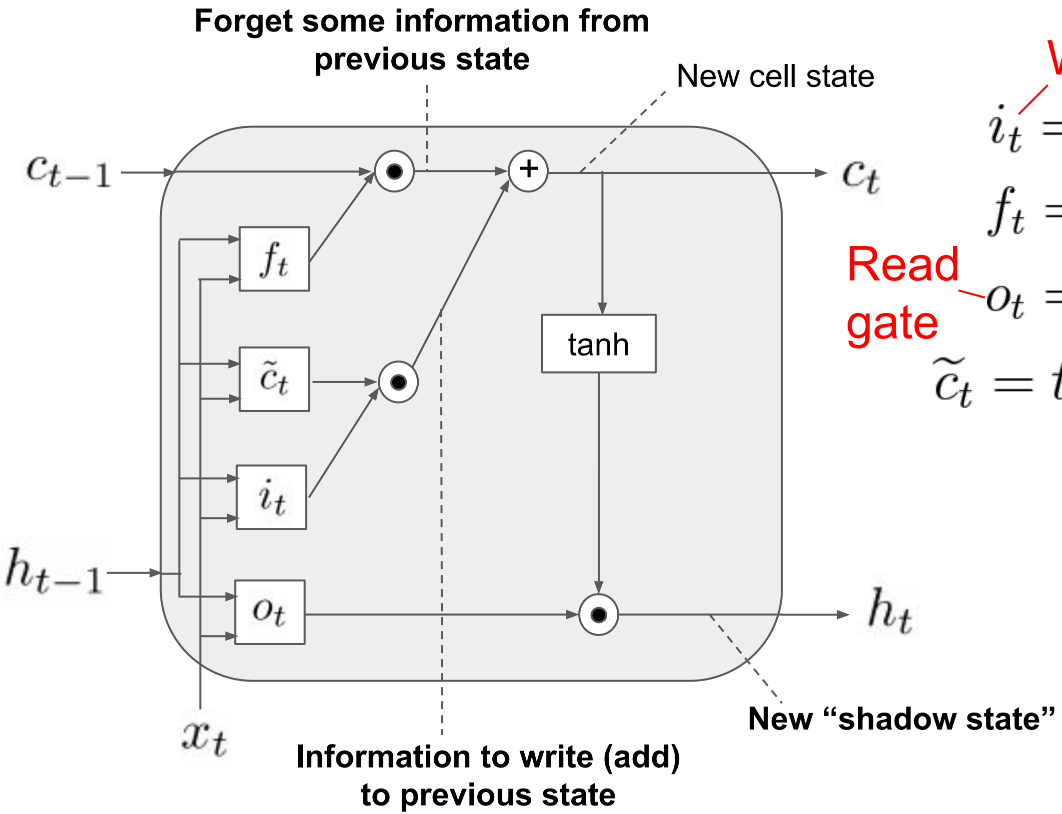
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

Where are Read/Write gates?

LSTM Cell



LSTM equations

Write 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

Read 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

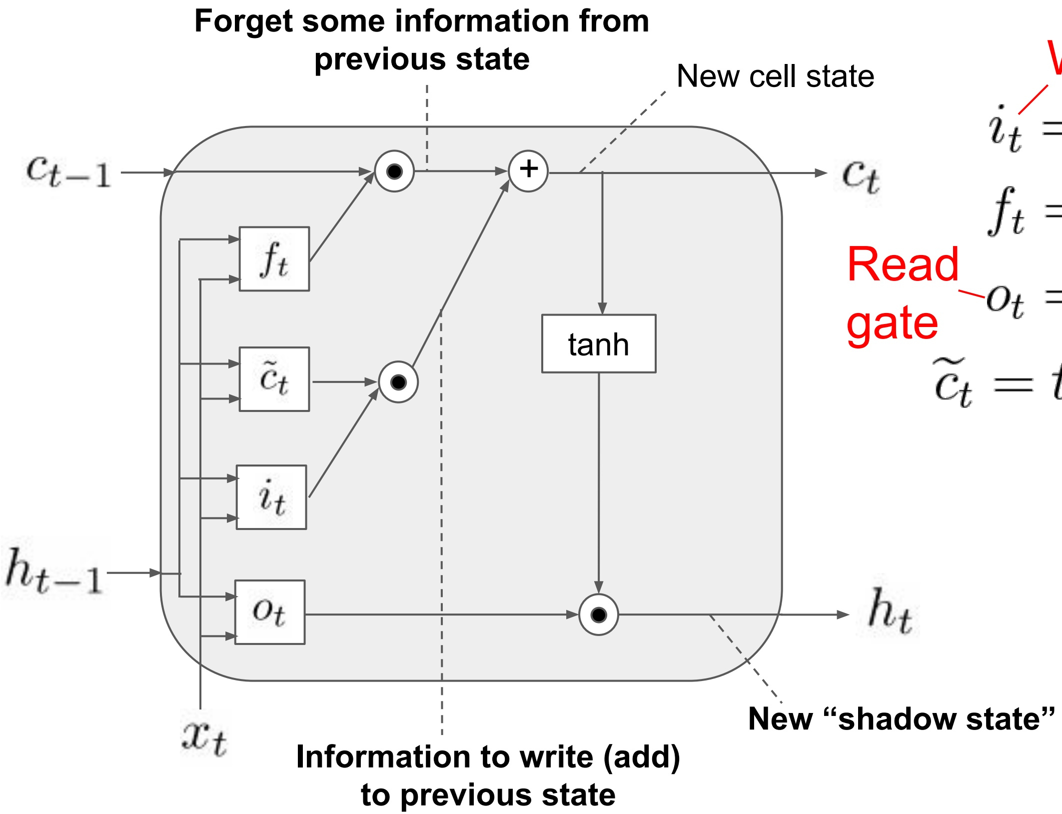
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

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

$$y_t = h_t$$ Cell Output

Reading occurs after writing

LSTM Cell



We use shadow state to calculate gates

LSTM equations

Write 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

Read 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

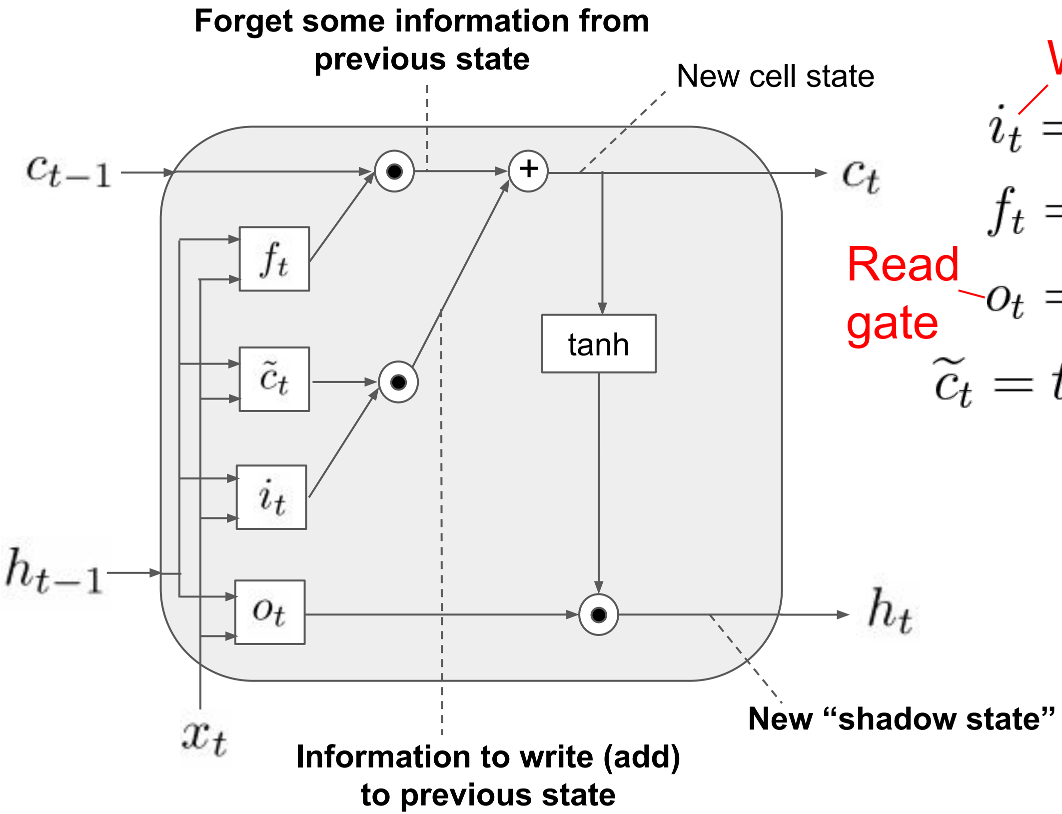
$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output

Reading occurs after writing

deepsystems.ai

LSTM Cell



Conceptually, we lose information

We use shadow state to calculate gates

LSTM equations

Write 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

Read gate

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

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

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

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

$$y_t = h_t$$ Cell Output

Read occurs after writing

deepsystems.ai

LSTM Cell with Peephole connections

LSTM equations

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

$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + P_i c_{t-1} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + P_f c_{t-1} + b_f)$$

LSTM Cell with Peephole connections

LSTM equations

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

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

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

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

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

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

$$y_t = h_t$$

LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + \underline{P_i c_{t-1}} + b_i)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + \underline{P_f c_{t-1}} + b_f)$$

deepsystems.ai

LSTM Cell with Peephole connections

LSTM equations

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

$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + \underline{P_i c_{t-1}} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + \underline{P_f c_{t-1}} + b_f)$$

$$\tilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

LSTM Cell with Peephole connections

LSTM equations

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

$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + \underline{P_i c_{t-1}} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + \underline{P_f c_{t-1}} + b_f)$$

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + P_o c_t + b_o)$$

deepsystems.ai

LSTM Cell with Peephole connections

LSTM equations

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

$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + \underline{P_i c_{t-1}} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + \underline{P_f c_{t-1}} + b_f)$$

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + \underline{P_o c_t} + b_o)$$

LSTM Cell with Peephole connections

LSTM equations

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

$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

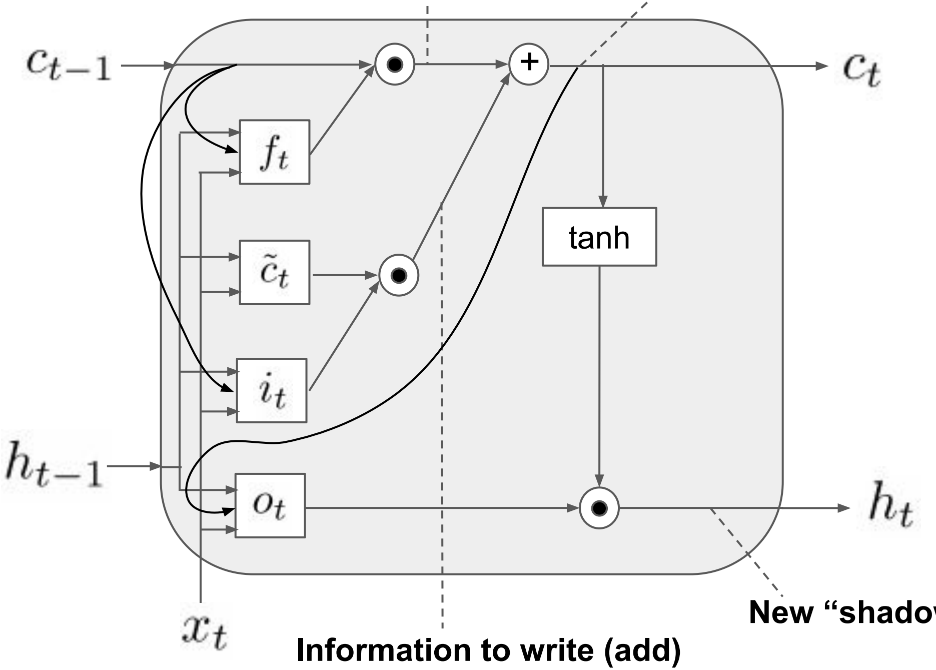
LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + P_i c_{t-1} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + P_f c_{t-1} + b_f)$$

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + P_o c_t + b_o)$$

$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

LSTM Cell with Peephole connections

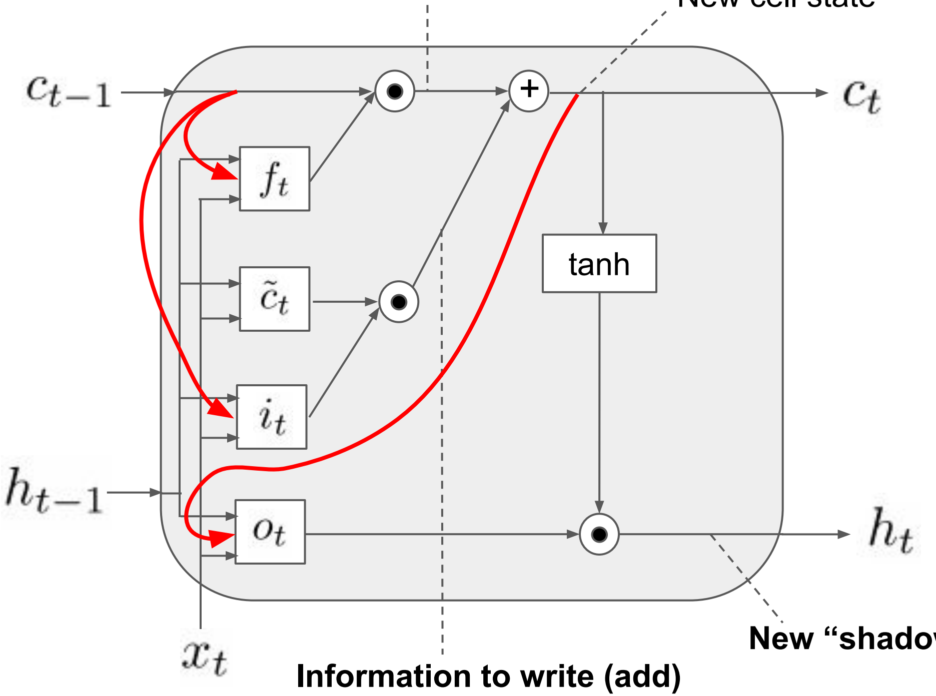


LSTM with Peephole connections

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

deepsystems.ai

LSTM Cell with Peephole connections



LSTM with Peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + \underline{P_i c_{t-1}} + b_i)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + \underline{P_f c_{t-1}} + b_f)$$

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

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

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + \underline{P_o c_t} + b_o)$$

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

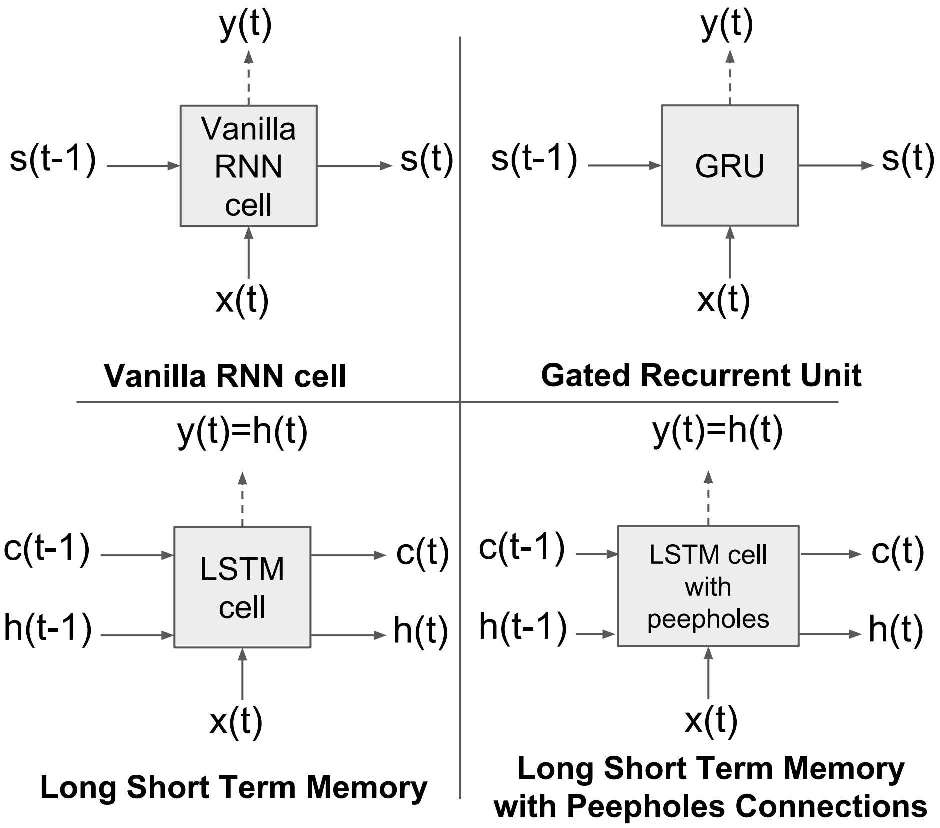
$$y_t = h_t$$

deepsystems.ai

Talk outline

1. RNN: the cell & simple examples
2. Key aspects (or ways to state of the art)
3. Vanilla RNN
4. Problems with Vanilla RNN and motivation for the more powerful cells
5. GRU: step by step
6. LSTM: step by step
7. LSTM with peephole connections
8. Conclusions

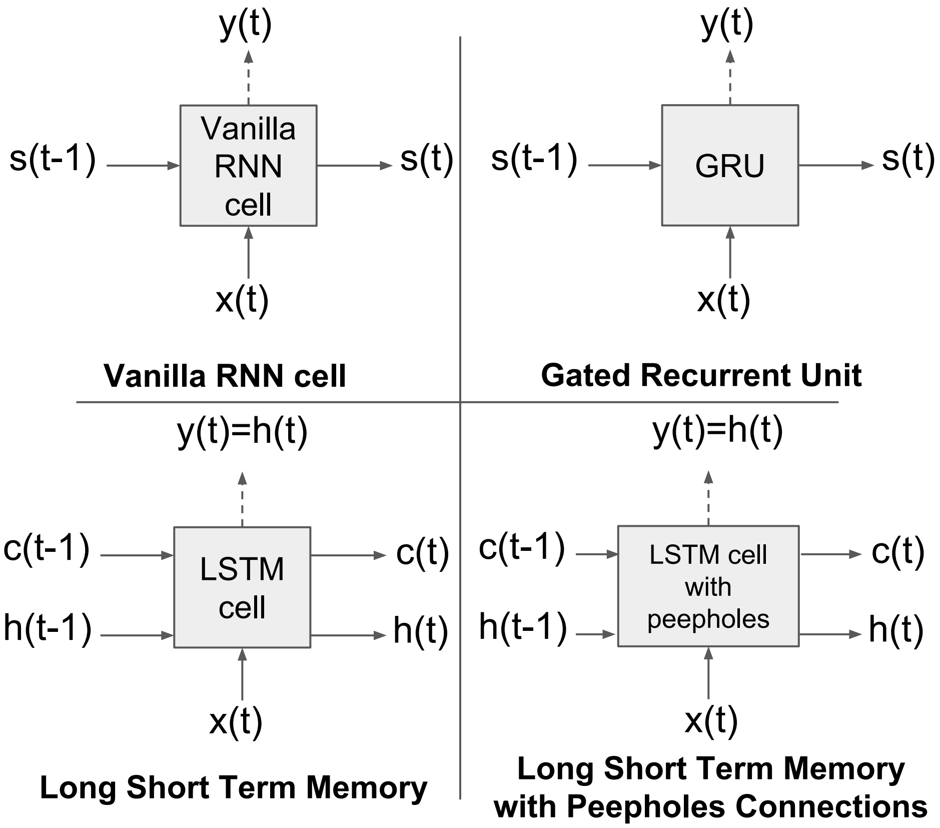
Let's conclude



Let’s conclude

Vanilla RNN

Late 1980s - backpropagation through time to train Vanilla RNN



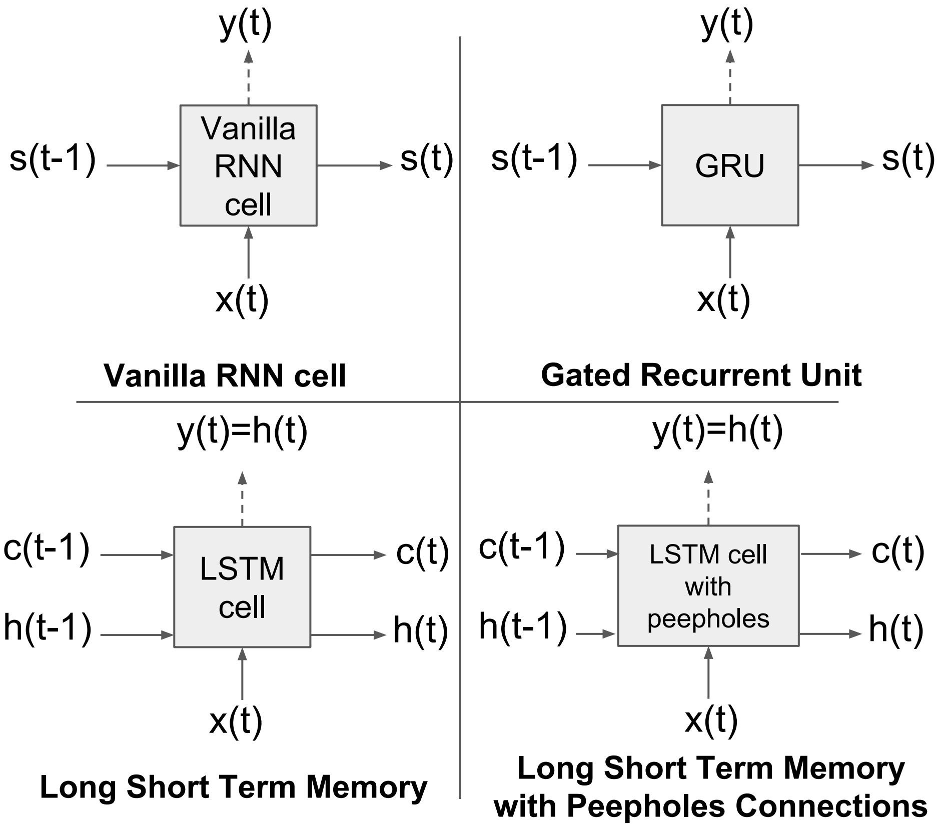
Let’s conclude

Vanilla RNN

Late 1980s - backpropagation through time to train Vanilla RNN

LSTM

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



Let’s conclude

Vanilla RNN

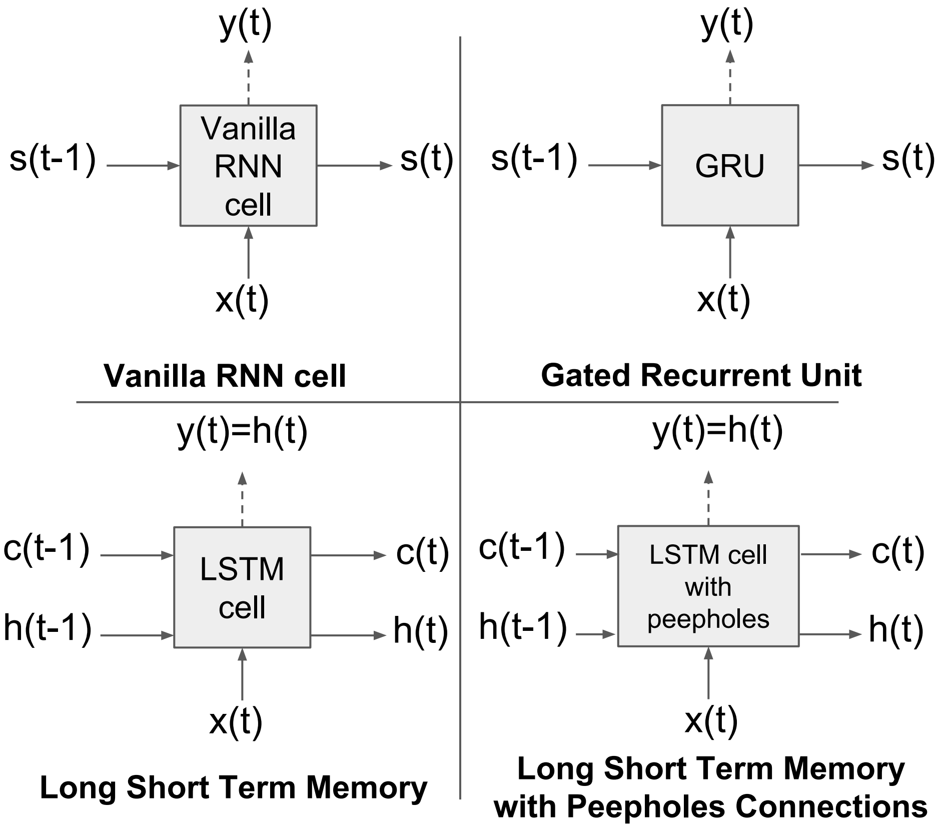
Late 1980s - backpropagation through time to train Vanilla RNN

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)



deepsystems.ai

Let's conclude

Vanilla RNN

Late 1980s - backpropagation through time to train Vanilla RNN

LSTM

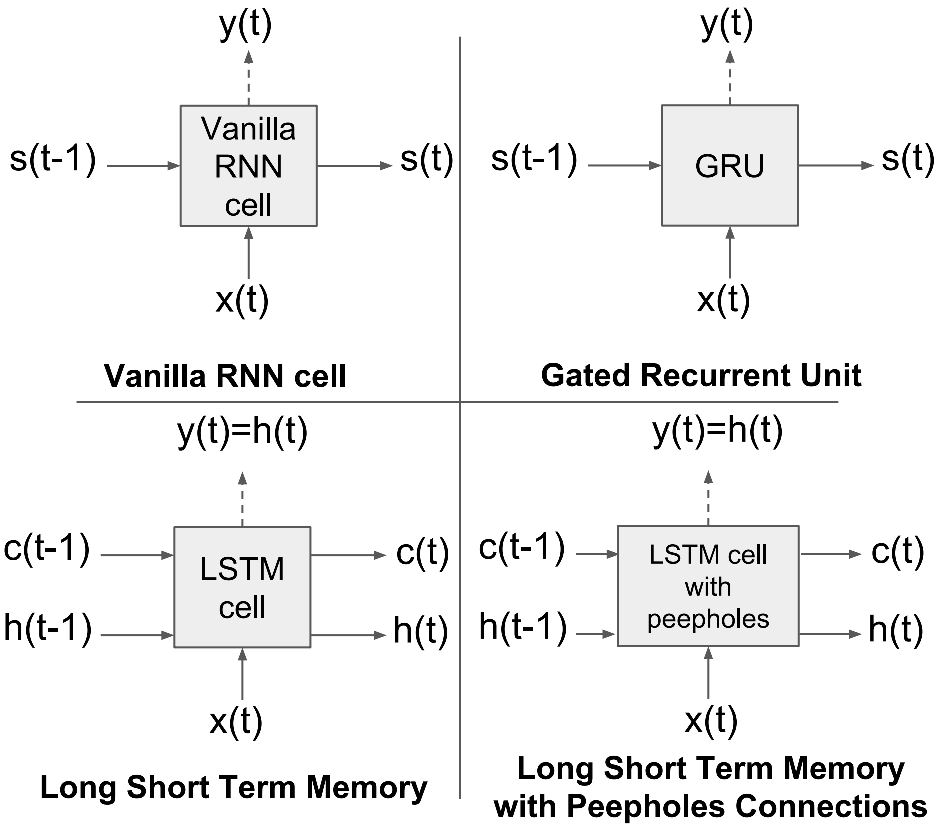
1997 - Long Short-Term Memory (S.Hochreiter, J.Schmidhuber)

LSTM with Peepholes

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

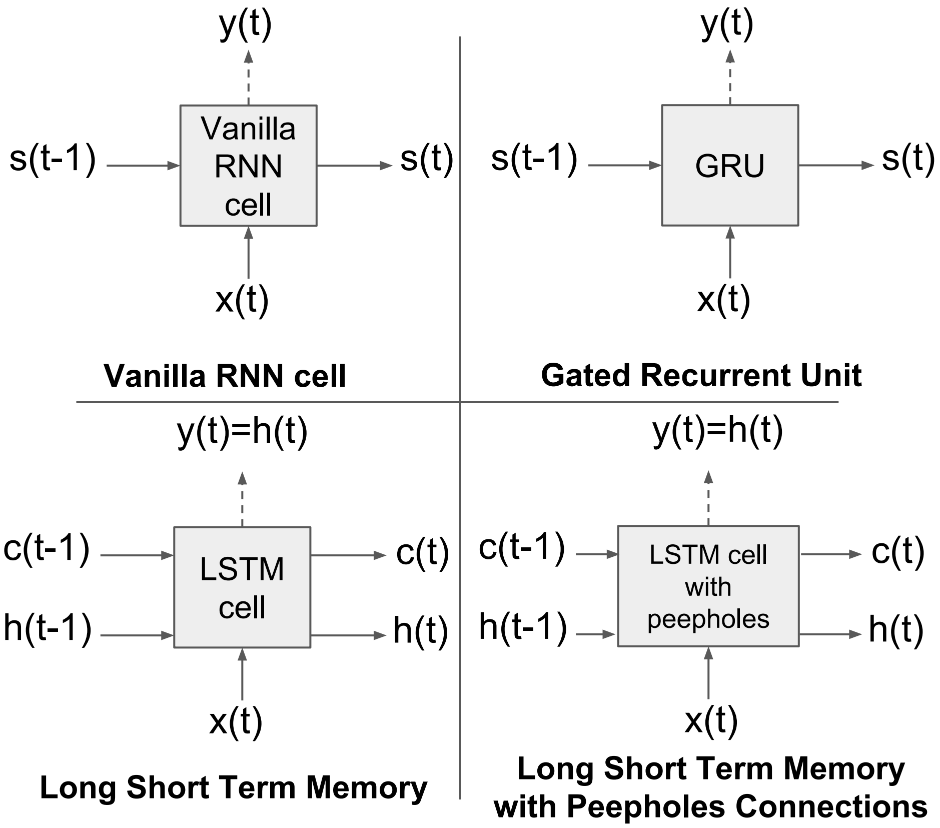
GRU

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



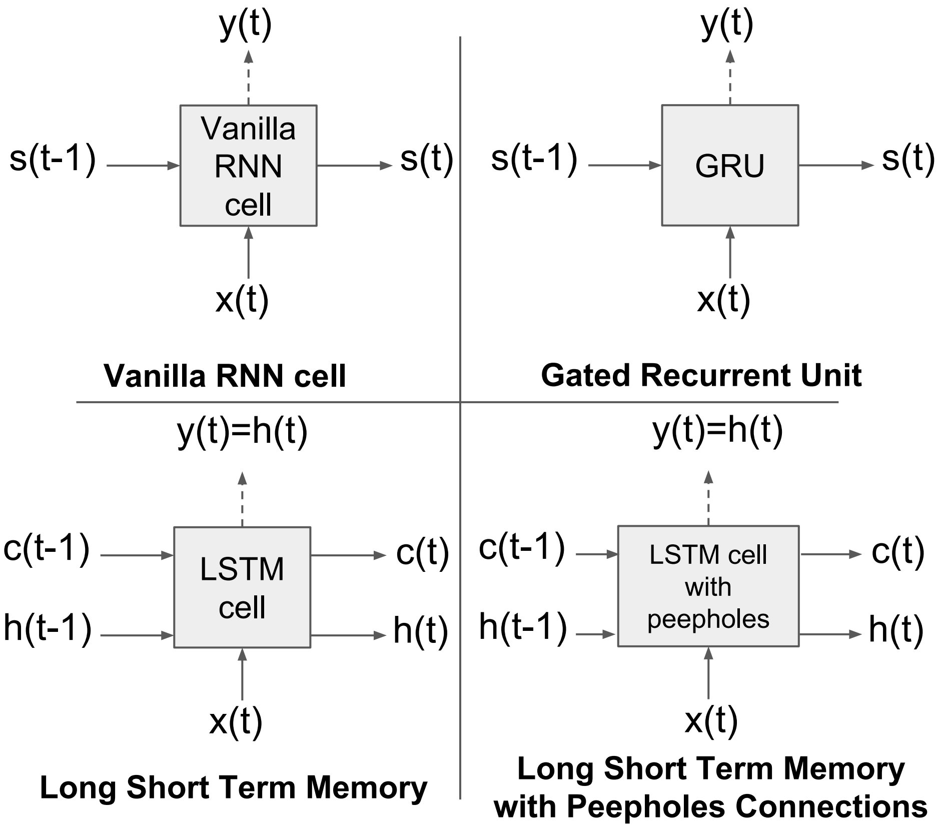
Let's conclude

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



Let's conclude

- 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

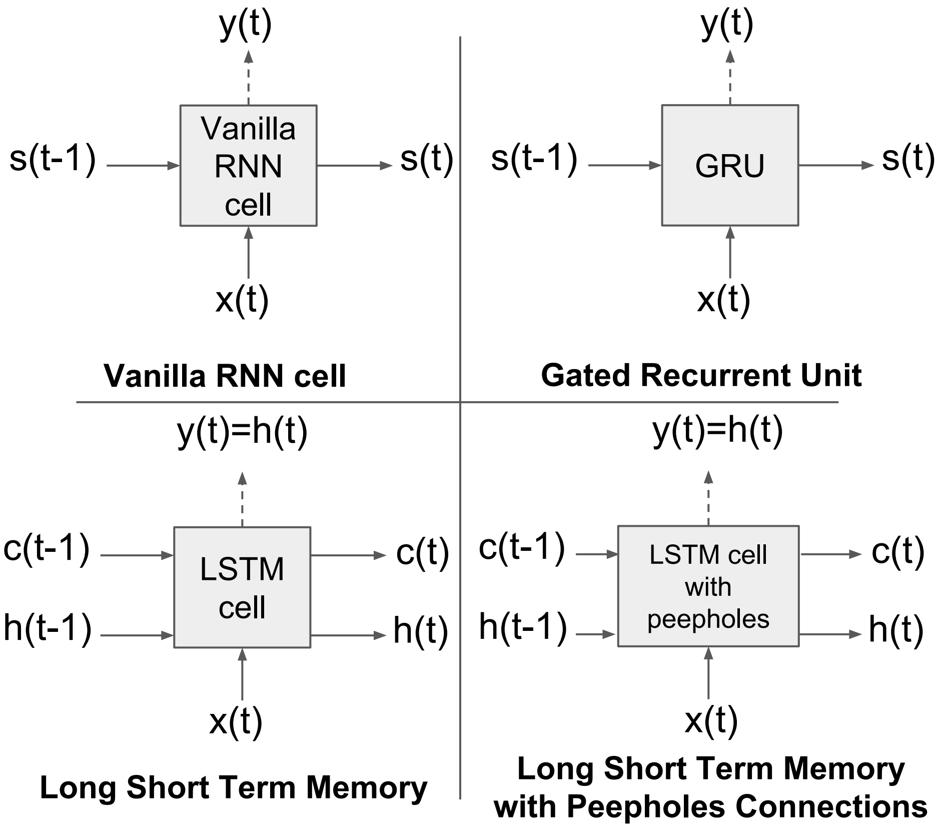


Let’s conclude

- 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



deepsystems.ai

Resources

Great Analysis, Tons of intuition

Written Memories: Understanding, Deriving and Extending the LSTM by R2RT

LSTM: original paper

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

LSTM with Peephole connections

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

GRU

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (K.Cho, Y.Bengio, and others)

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling (J.Chung, C.Gulcehre, K.Cho, Y.Bengio)

Thank you

Our Website:

deepsystems.ai

Thank you

Our Website:

deepsystems.ai

Products:

supervise.ly - Dataset management, annotation and preparation service

Thank you

Our Website:

deepsystems.ai

Products:

supervise.ly - Dataset management, annotation and preparation service

movix.ai - Interactive, lstm-based movie recommender system

Thank you

Our Website:

deepsystems.ai

Products:

supervise.ly - Dataset management, annotation and preparation service

movix.ai - Interactive, lstm-based movie recommender system

Outsource projects:

Our team is looking for business partners to make exciting deep learning solutions.