Long Short-Term Memory RNNs (LSTMs)

- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the **vanishing gradients problem**.
  - Everyone cites that paper but really a crucial part of the modern LSTM is from Gers et al. (2000)

LONG SHORT-TERM MEMORY

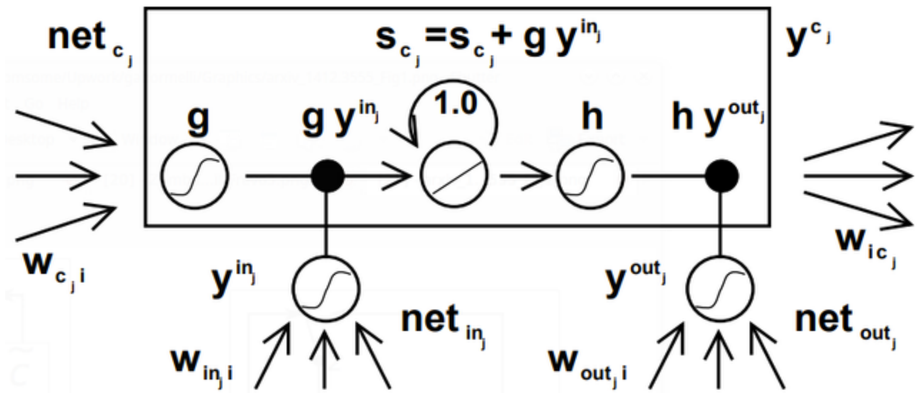
Neural Computation 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Learning to Forget: Continual Prediction with LSTM

Felix A. Gers
Jürgen Schmidhuber
Fred Cummins
IDSIA, 6900 Lugano, Switzerland



[advanced]

Recap: Vanishing Gradient Problem

$$\mathbf{h}_2 = g(\mathbf{W}\mathbf{h}_1 + \mathbf{U}\mathbf{x}_2 + \mathbf{b})$$
$$\mathbf{h}_3 = g(\mathbf{W}\mathbf{h}_2 + \mathbf{U}\mathbf{x}_3 + \mathbf{b})$$
$$L_3 = -\log \hat{\mathbf{y}}_3(w_4)$$

$$\frac{\partial L_3}{\partial \mathbf{W}} = \frac{\partial L_3}{\partial \mathbf{h}_3}\frac{\partial \mathbf{h}_3}{\partial \mathbf{W}} + \frac{\partial L_3}{\partial \mathbf{h}_3}\frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2}\frac{\partial \mathbf{h}_2}{\partial \mathbf{W}} + \frac{\partial L_3}{\partial \mathbf{h}_3}\frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2}\frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1}\frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$$

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

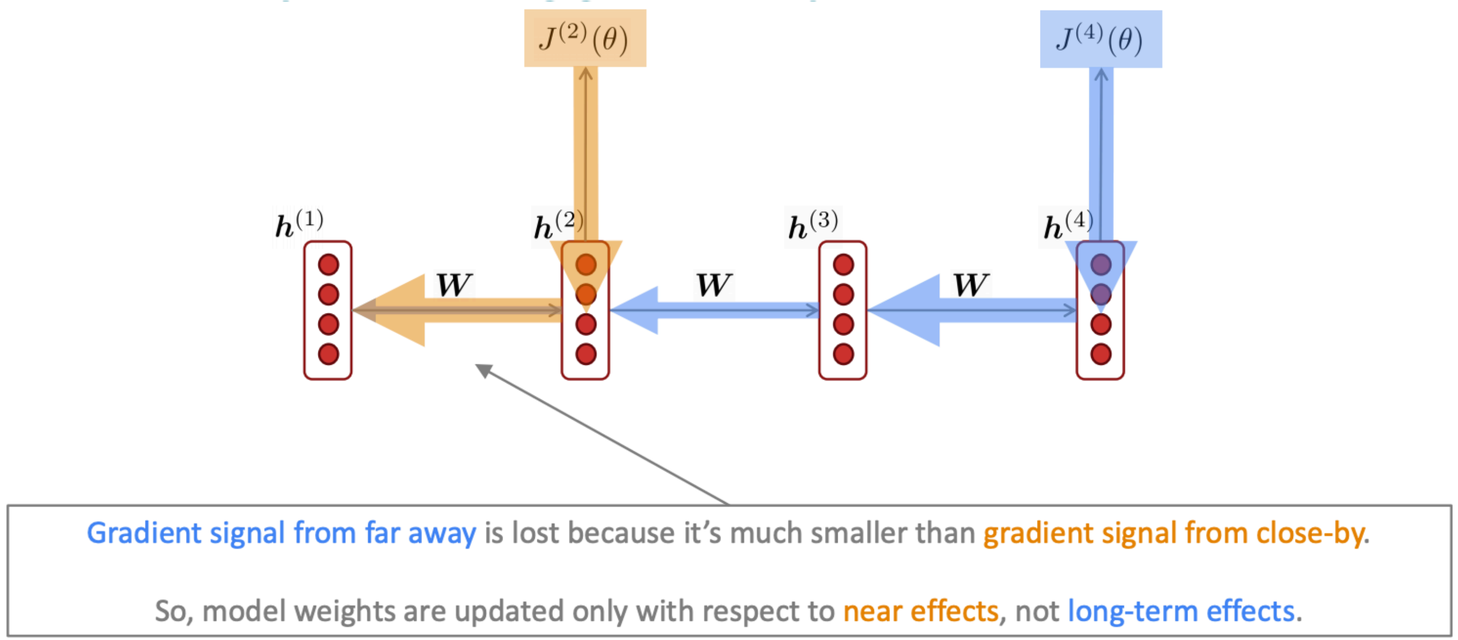
$$\frac{\partial L}{\partial \mathbf{W}} = -\frac{1}{n}\sum_{t=1}^{n}\sum_{k=1}^{t}\frac{\partial L_t}{\partial \mathbf{h}_t}\left(\prod_{j=k+1}^{t}\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}\right)\frac{\partial \mathbf{h}_k}{\partial \mathbf{W}}$$

If k and t are far away, the gradients are very easy to grow/shrink exponentially

(called the gradient exploding or gradient vanishing problem)

[advanced]

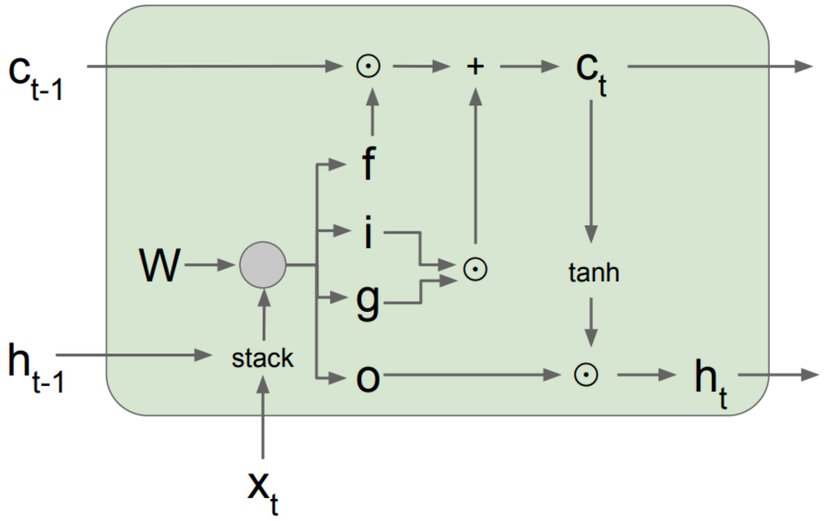
Recap: Vanishing Gradient Problem



When she tried to print her **tickets**, she found that the printer was out of toner.
She went to the stationery store to buy more toner. It was very overpriced.
After installing the toner into the printer, she finally printed her _____

LSTMs: The intuition

- Key idea: turning **multiplication** into **addition** and using "**gates**" to control how much information to add/erase
- At each time step, instead of re-writing the hidden state $\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b})$, there is also a cell state $\mathbf{c}_t \in \mathbb{R}^h$ which stores **long-term information**
  - We write to/erase information from $\mathbf{c}_t$ after each step
  - We read $\mathbf{h}_t$ from $\mathbf{c}_t$

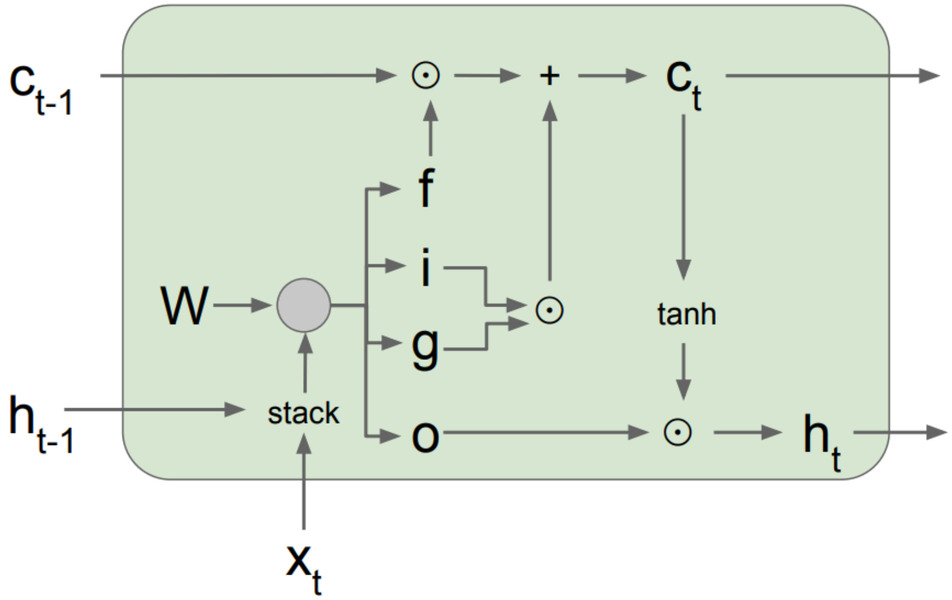


LSTMs: the formulation

- Input gate **(how much to write)**:
  $\mathbf{i}_t = \sigma(\mathbf{W}^i\mathbf{h}_{t-1} + \mathbf{U}^i\mathbf{x}_t + \mathbf{b}^i) \in \mathbb{R}^h$
- Forget gate **(how much to erase)**:
  $\mathbf{f}_t = \sigma(\mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{U}^f\mathbf{x}_t + \mathbf{b}^f) \in \mathbb{R}^h$
- Output gate **(how much to reveal)**:
  $\mathbf{o}_t = \sigma(\mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{U}^o\mathbf{x}_t + \mathbf{b}^{(o)}) \in \mathbb{R}^h$
- New memory cell **(what to write)**:
  $\mathbf{g}_t = \tanh(\mathbf{W}^g\mathbf{h}_{t-1} + \mathbf{U}^g\mathbf{x}_t + \mathbf{b}^g) \in \mathbb{R}^h$
- Final memory cell: $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$

element-wise product

- Final hidden cell: $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$



$\mathbf{h}_0, \mathbf{c}_0 \in \mathbb{R}^h$ are initial states (usually set to $\mathbf{0}$)

LSTMs: the formulation

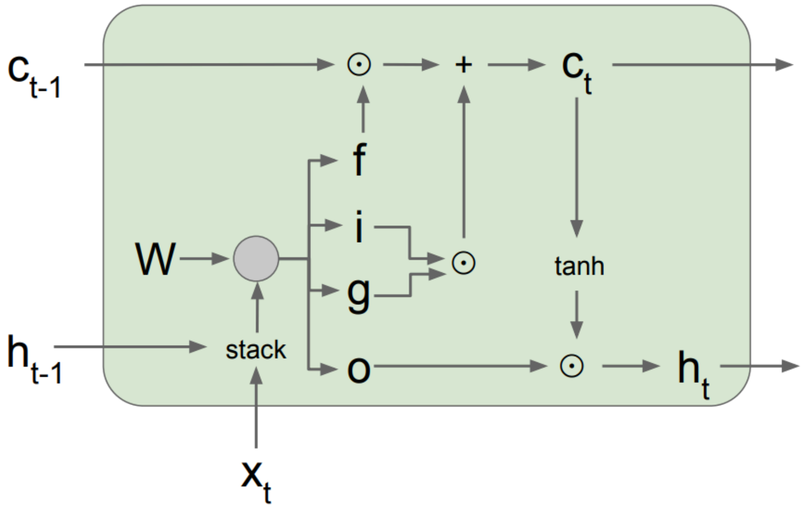
- LSTMs has 4x parameters compared to simple RNNs:

Input dimension: d, hidden size: h

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \in \mathbb{R}^h$$

$$\mathbf{W} \in \mathbb{R}^{h\times h}, \mathbf{U} \in \mathbb{R}^{h\times d}, \mathbf{b} \in \mathbb{R}^h$$

$$\mathbf{W}^i, \mathbf{W}^f, \mathbf{W}^g, \mathbf{W}^o \in \mathbb{R}^{h\times h}$$
$$\mathbf{U}^i, \mathbf{U}^f, \mathbf{U}^g, \mathbf{U}^o \in \mathbb{R}^{h\times d}$$
$$\mathbf{b}^i, \mathbf{b}^f, \mathbf{b}^g, \mathbf{b}^o \in \mathbb{R}^h$$

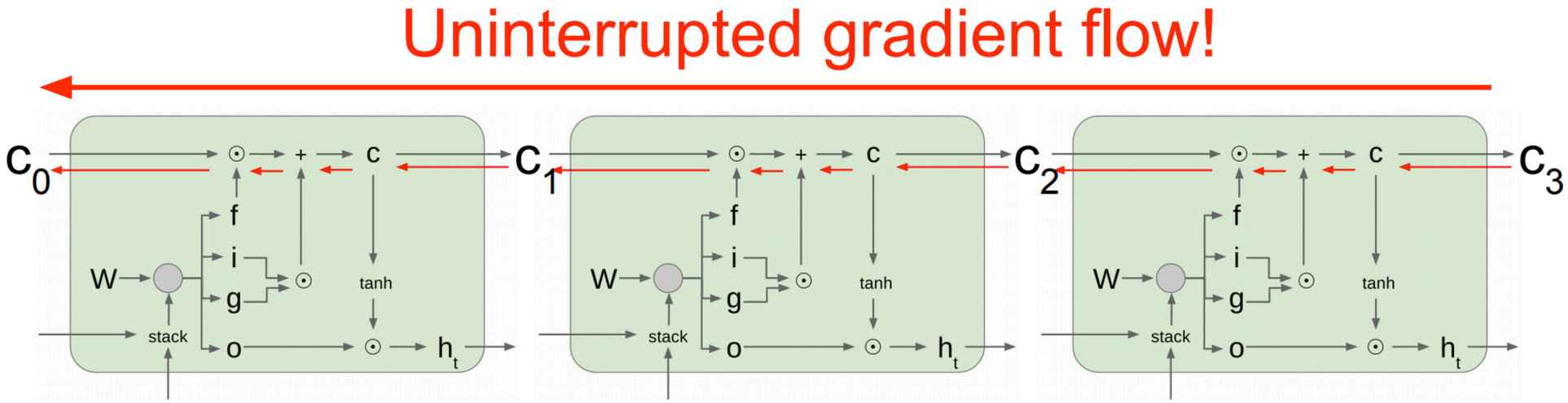


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

Q: What is the range of the hidden representations $\mathbf{h}_t$?

LSTMs: the formulation



- LSTM doesn't guarantee that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies

- LSTMs were invented in 1997 but finally got working from 2013-2015.

Gated Recurrent Units (GRUs)

- Introduced by Kyunghyun Cho in 2014:

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho
Bart van Merriënboer **Caglar Gulcehre** Université de Montréal
firstname.lastname@umontreal.ca

Dzmitry Bahdanau Jacobs University, Germany
d.bahdanau@jacobs-university.de

Fethi Bougares **Holger Schwenk** Université du Maine, France
firstname.lastname@lium.univ-lemans.fr

Yoshua Bengio Université de Montréal, CIFAR Senior Fellow
find.me@on.the.web

- Simplified 3 gates to 2 gates: **reset** gate and **update** gate, without an explicit cell state

Gated Recurrent Units (GRUs)

- Reset gate:

$$\mathbf{r}_t = \sigma(\mathbf{W}^r \mathbf{h}_{t-1} + \mathbf{U}^r \mathbf{x}_t + \mathbf{b}^r)$$

- Update gate:

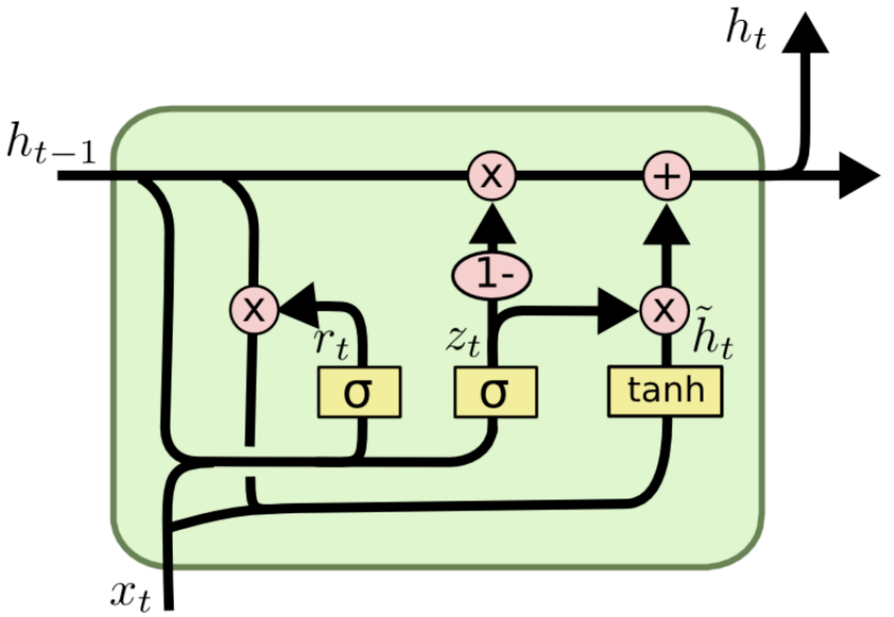
$$\mathbf{z}_t = \sigma(\mathbf{W}^z \mathbf{h}_{t-1} + \mathbf{U}^z \mathbf{x}_t + \mathbf{b}^z)$$

- New hidden state:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

merge input and forget gate!

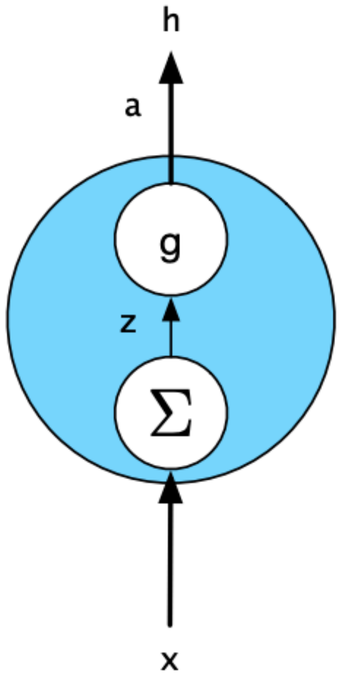


Q: What is the range of the hidden representations $\mathbf{h}_t$?

Q: How many parameters are there compared to simple RNNs?

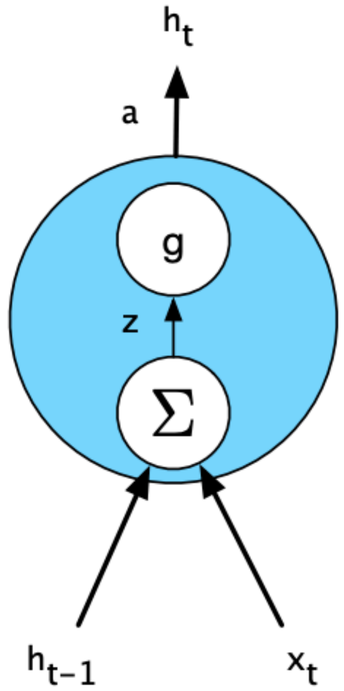
Comparison: FFNNs vs simple RNNs vs LSTMs vs GRUs

Feedforward NNs



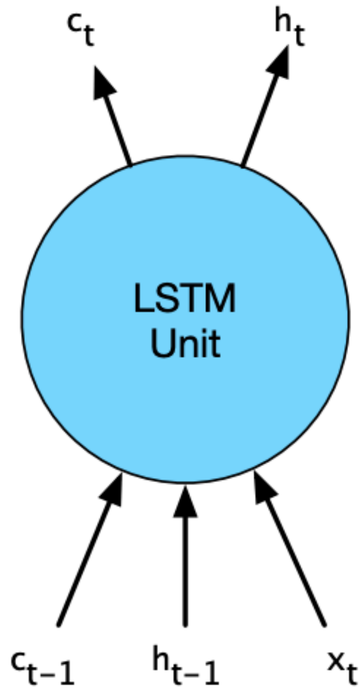
(a)

Simple RNNs



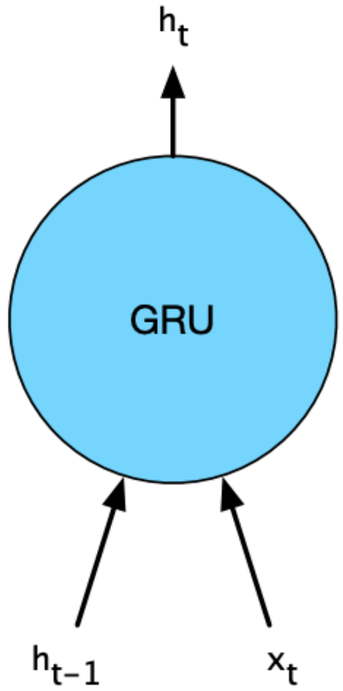
(b)

LSTMs



(c)

GRUs



(d)