



COS 484

Natural Language Processing

LII: Recurrent Neural Networks (II)

Spring 2022

Recap: Recurrent Neural Networks

$\mathbf{h}_0 \in \mathbb{R}^h$ is an initial state

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \in \mathbb{R}^h$$

\mathbf{h}_t : hidden states which store information from \mathbf{x}_1 to \mathbf{x}_t

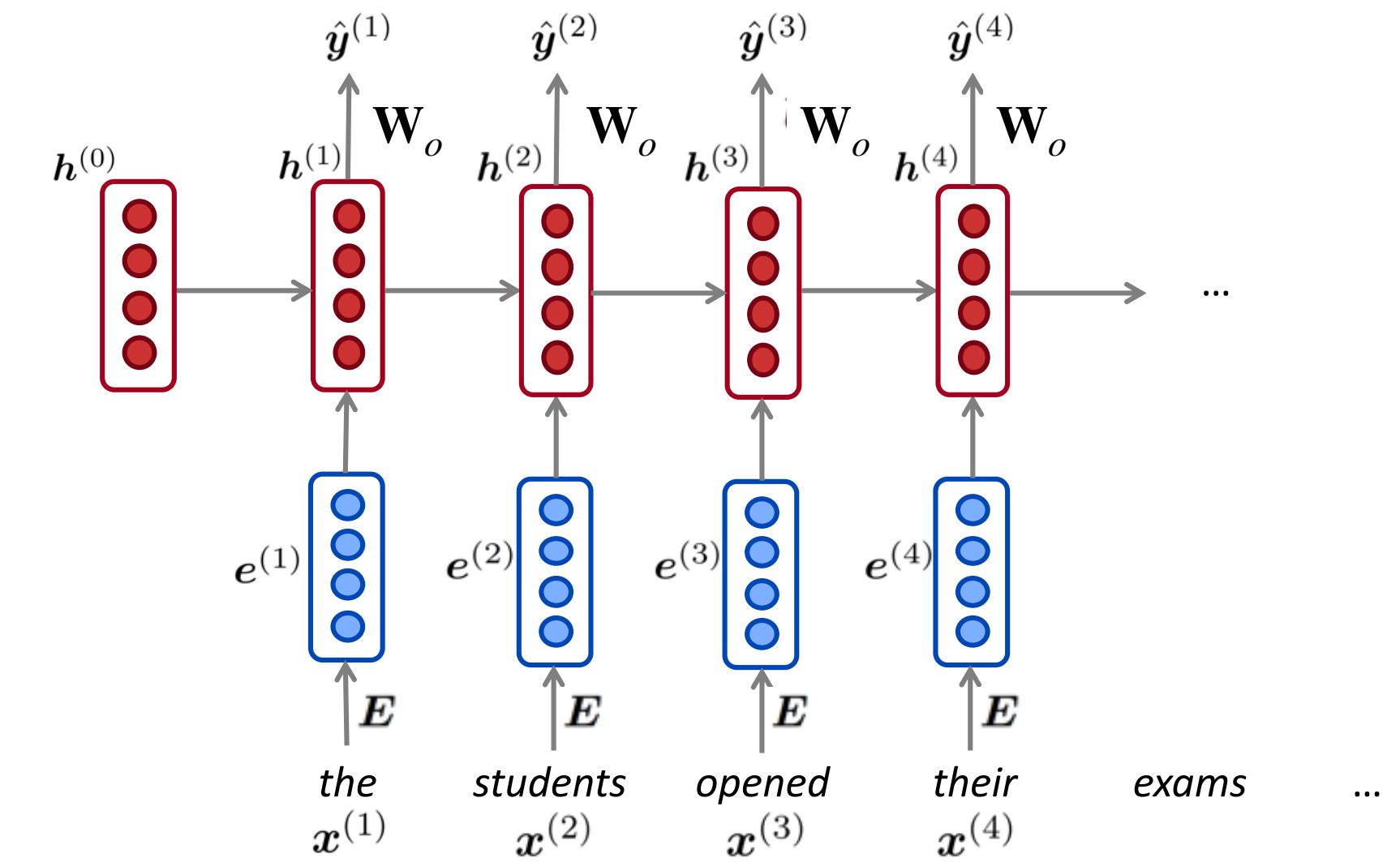
Simple RNNs:

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

g : nonlinearity (e.g. tanh),

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

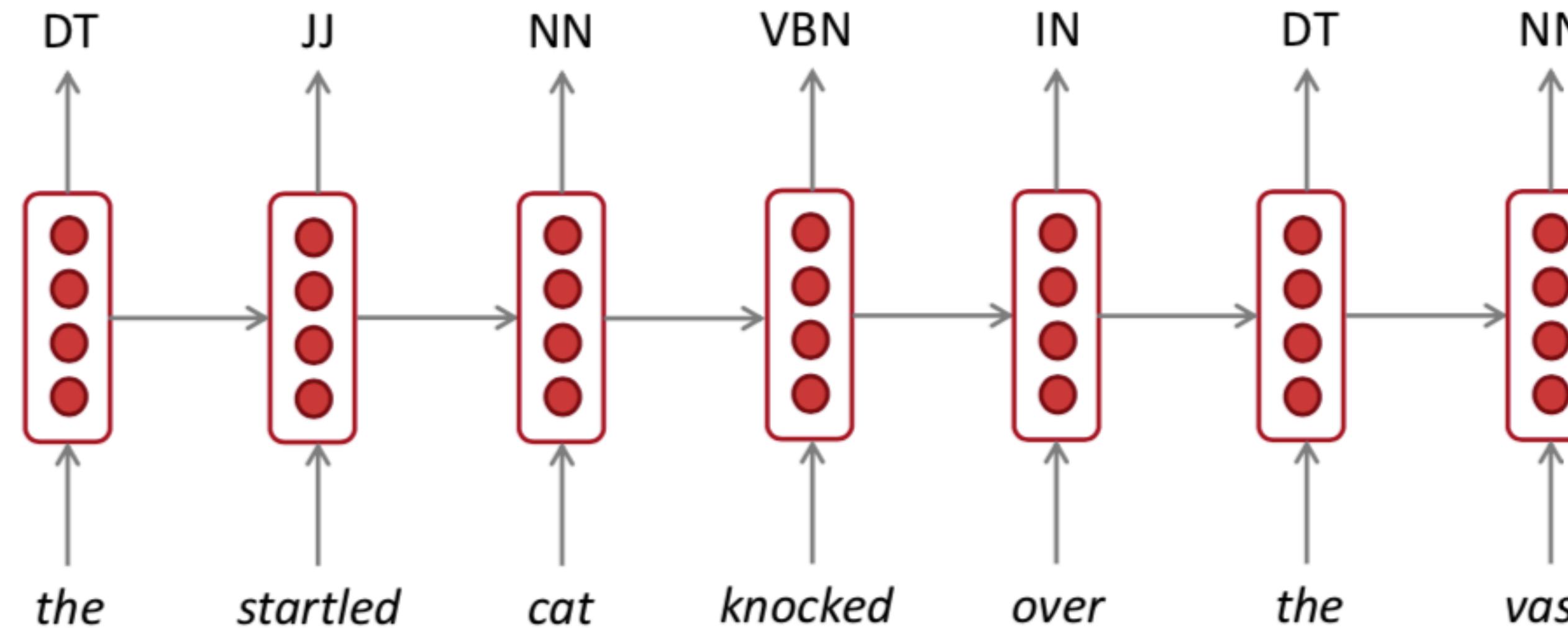
RNNLMs:



Application: Sequence Tagging

Input: a sentence of n words: x_1, \dots, x_n

Output: $y_1, \dots, y_n, y_i \in \{1, \dots, C\}$



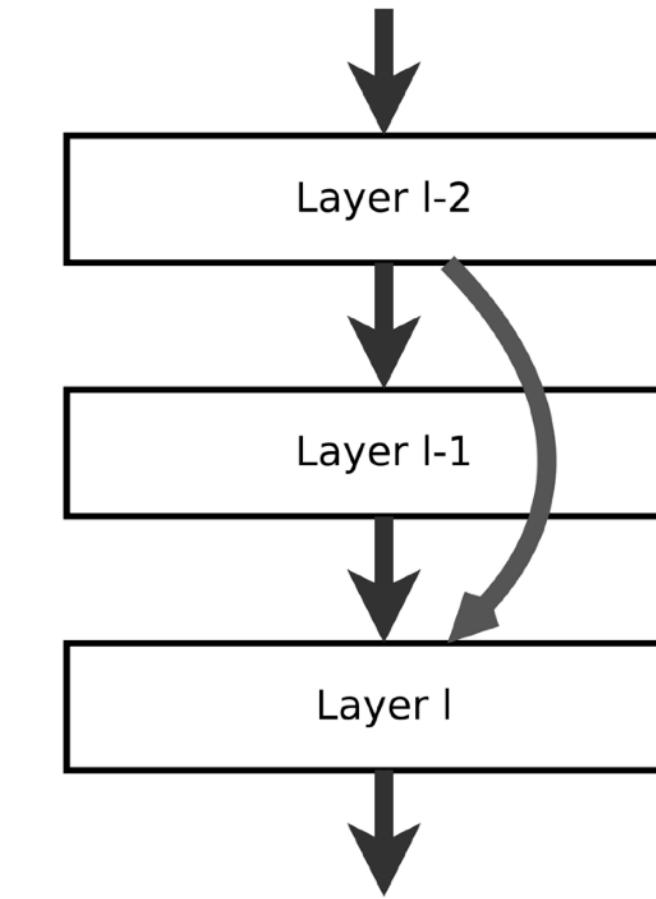
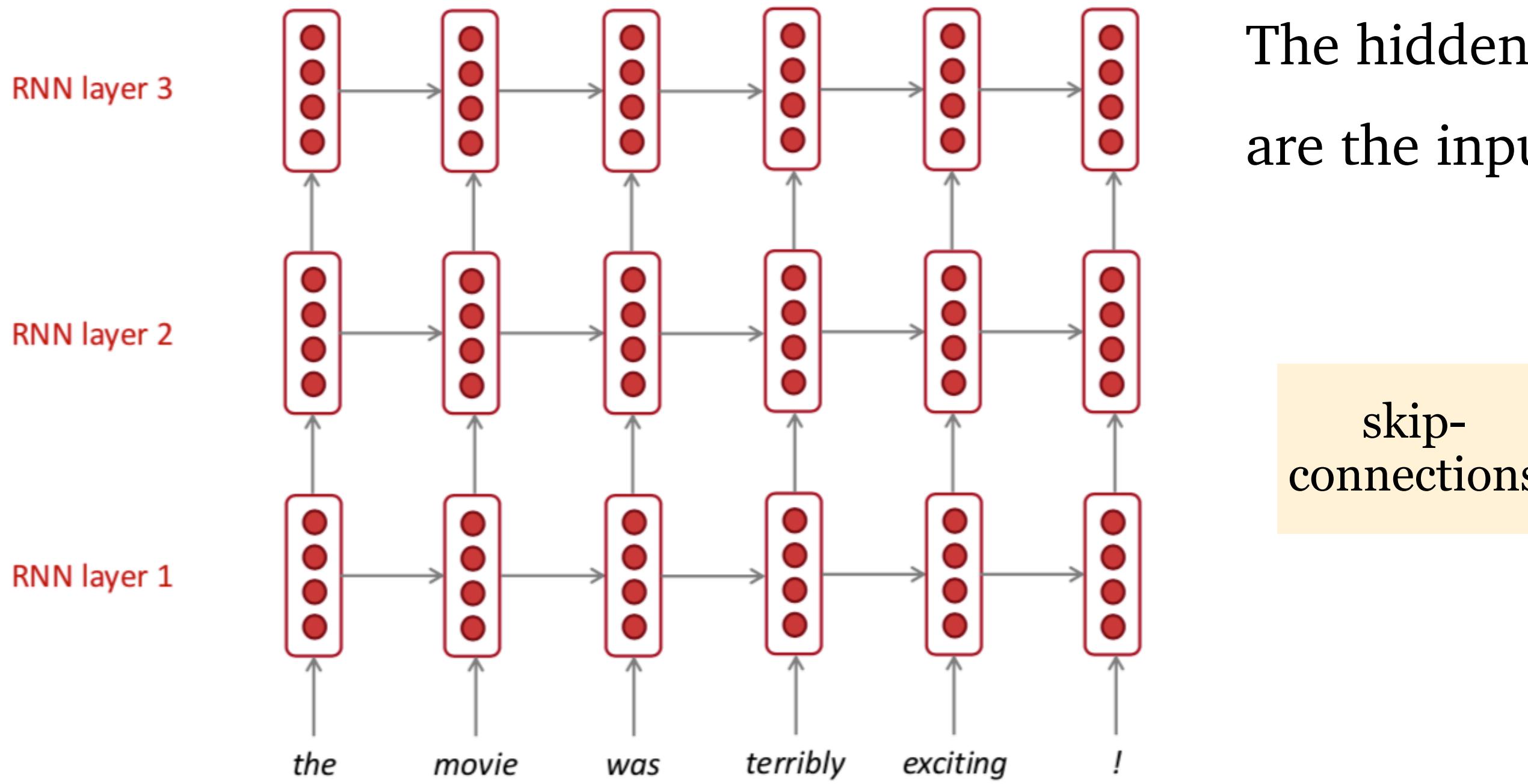
$$P(y_i = k) = \text{softmax}_k(\mathbf{W}_o \mathbf{h}_i)$$

$$\mathbf{W}_o \in \mathbb{R}^{C \times h}$$

$$L = -\frac{1}{n} \sum_{i=1}^n \log P(y_i = k)$$

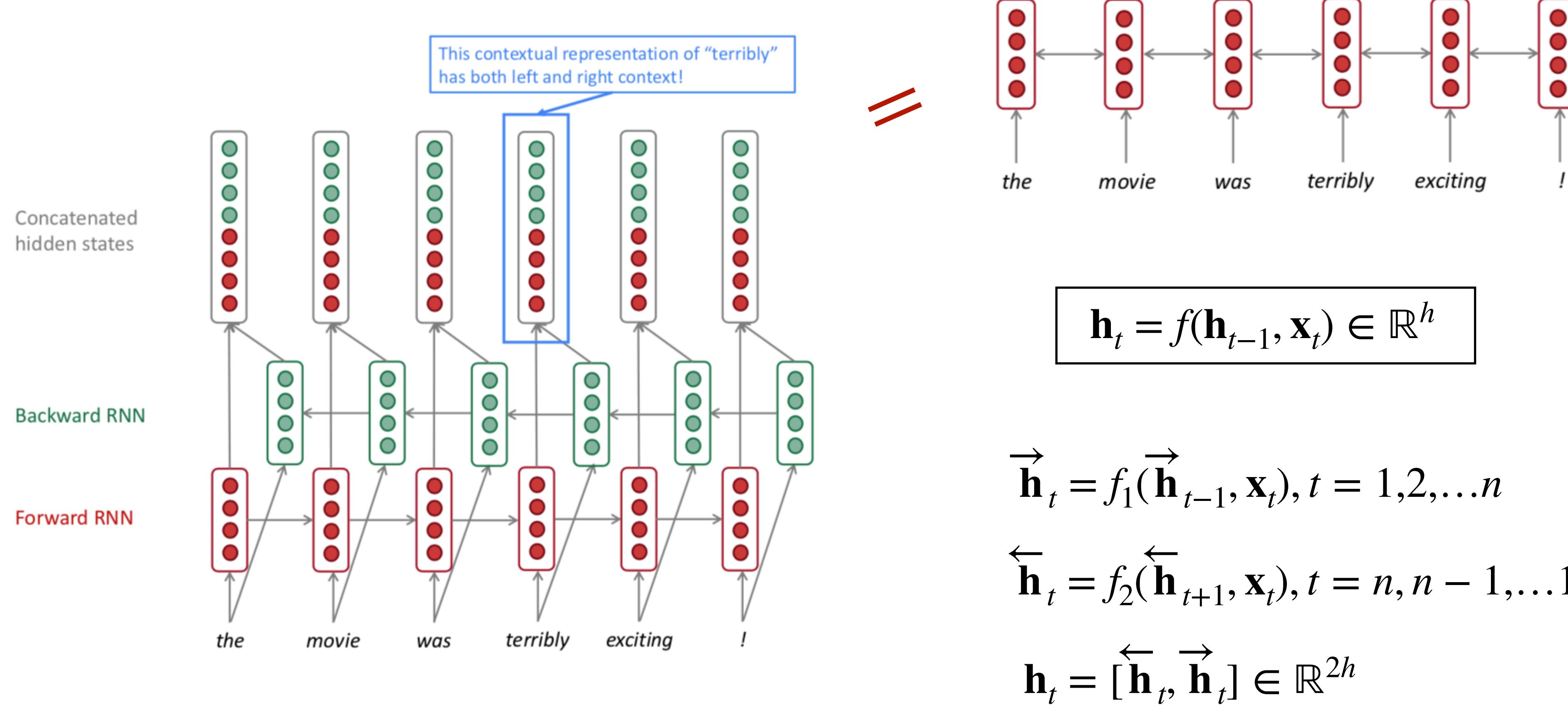
A better solution is to use RNNs + conditional random field (CRF), see Lample et al., 2016 for more details

Multi-layer RNNs

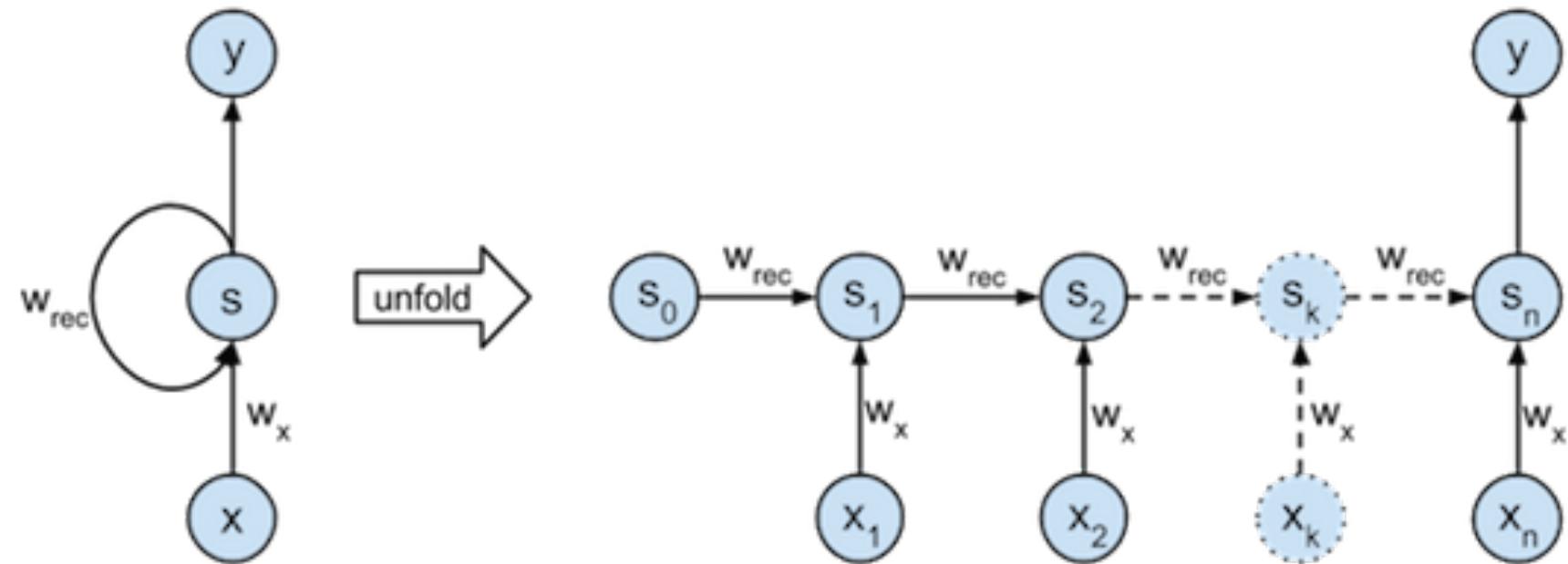


- In practice, using 2 to 4 layers is common (usually better than 1 layer)
- Transformer networks can be up to 24 layers with lots of skip-connections.

Bidirectional RNNs



Advanced RNN variants



$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \in \mathbb{R}^h$$

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

LSTMs

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

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

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

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

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{g}_t \odot \mathbf{i}_t$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t$$

GRUs

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

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

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

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

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

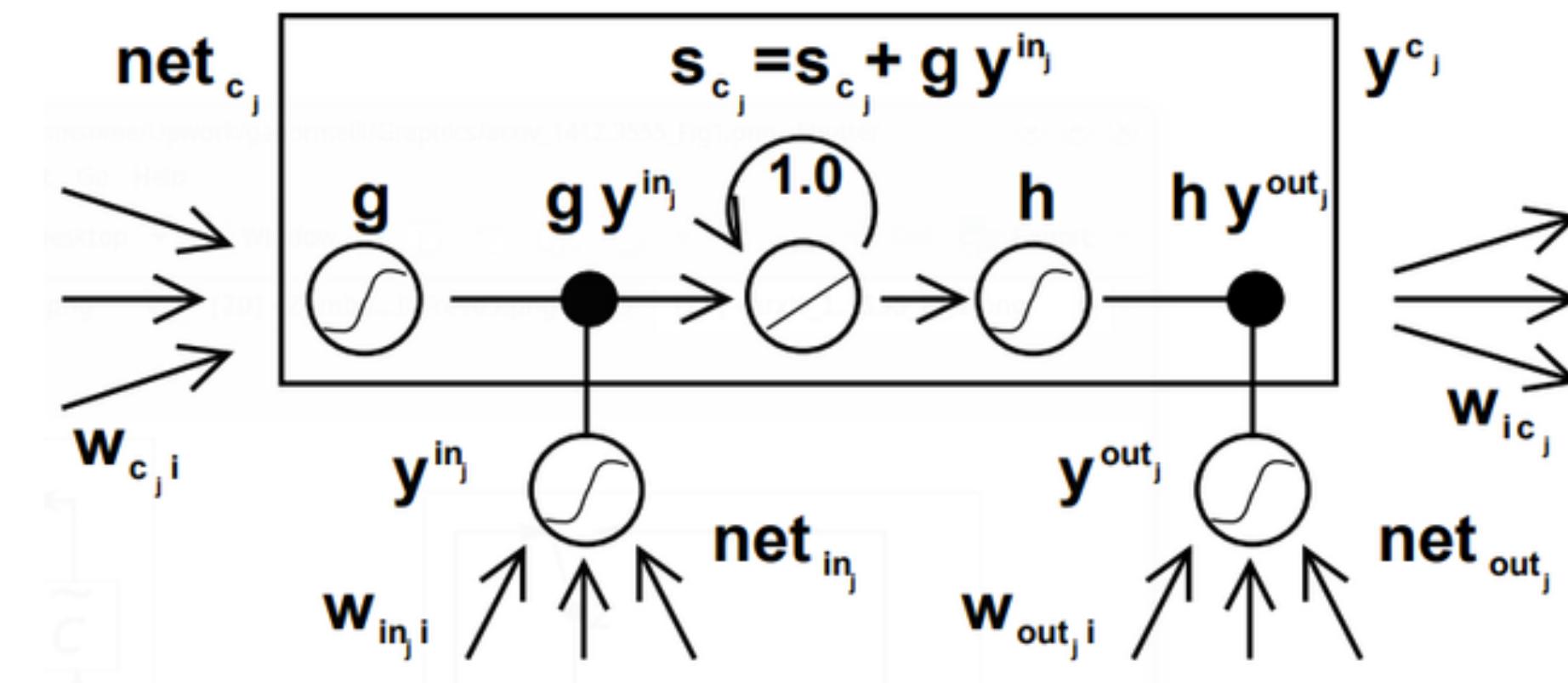
NEURAL COMPUTATION 9(8):1735–1780, 1997

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



Recap: Vanishing Gradient Problem

$$\mathbf{h}_2 = g(\mathbf{W}\mathbf{h}_1 + \mathbf{U}\mathbf{x}_2 + \mathbf{b})$$

$$\mathbf{h}_3 = g(\mathbf{W}\mathbf{h}_2 + \mathbf{U}\mathbf{x}_3 + \mathbf{b})$$

$$L_3 = -\log \hat{\mathbf{y}}_3(w_4)$$

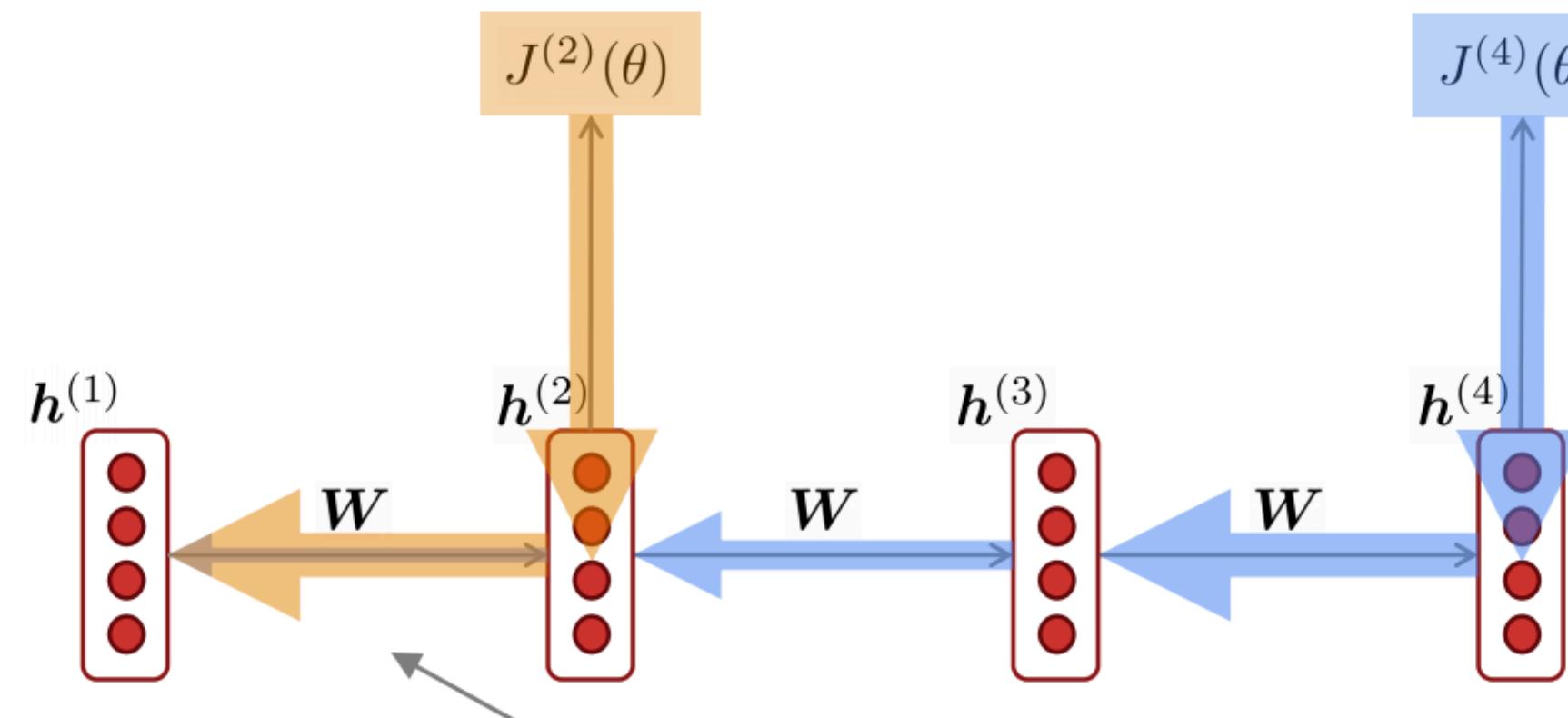
$$\frac{\partial L_3}{\partial \mathbf{W}} = \frac{\partial L_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{W}} + \frac{\partial L_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2}{\partial \mathbf{W}} + \frac{\partial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

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)

Recap: Vanishing Gradient Problem



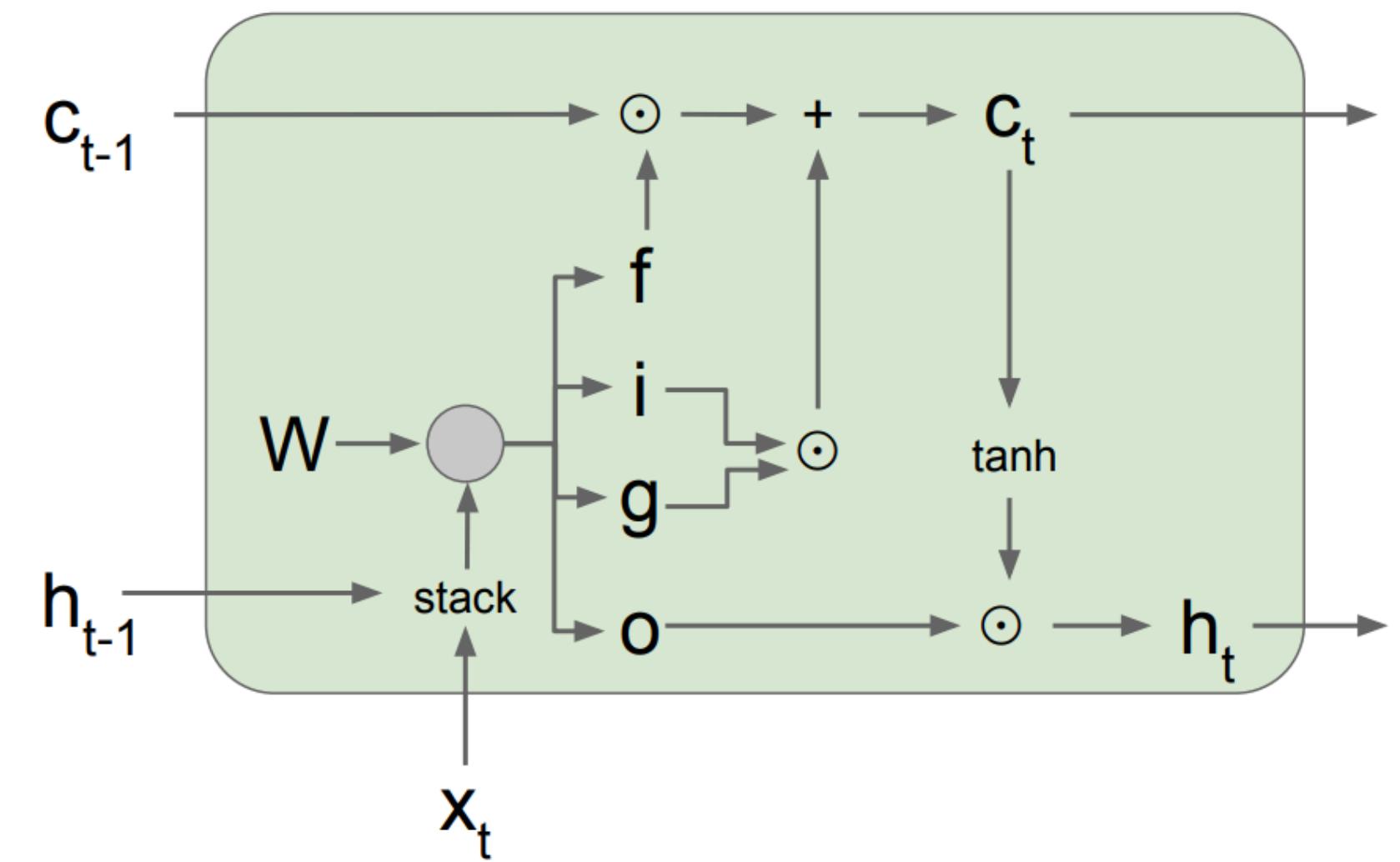
Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are updated only with respect to near effects, not long-term effects.

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**):

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

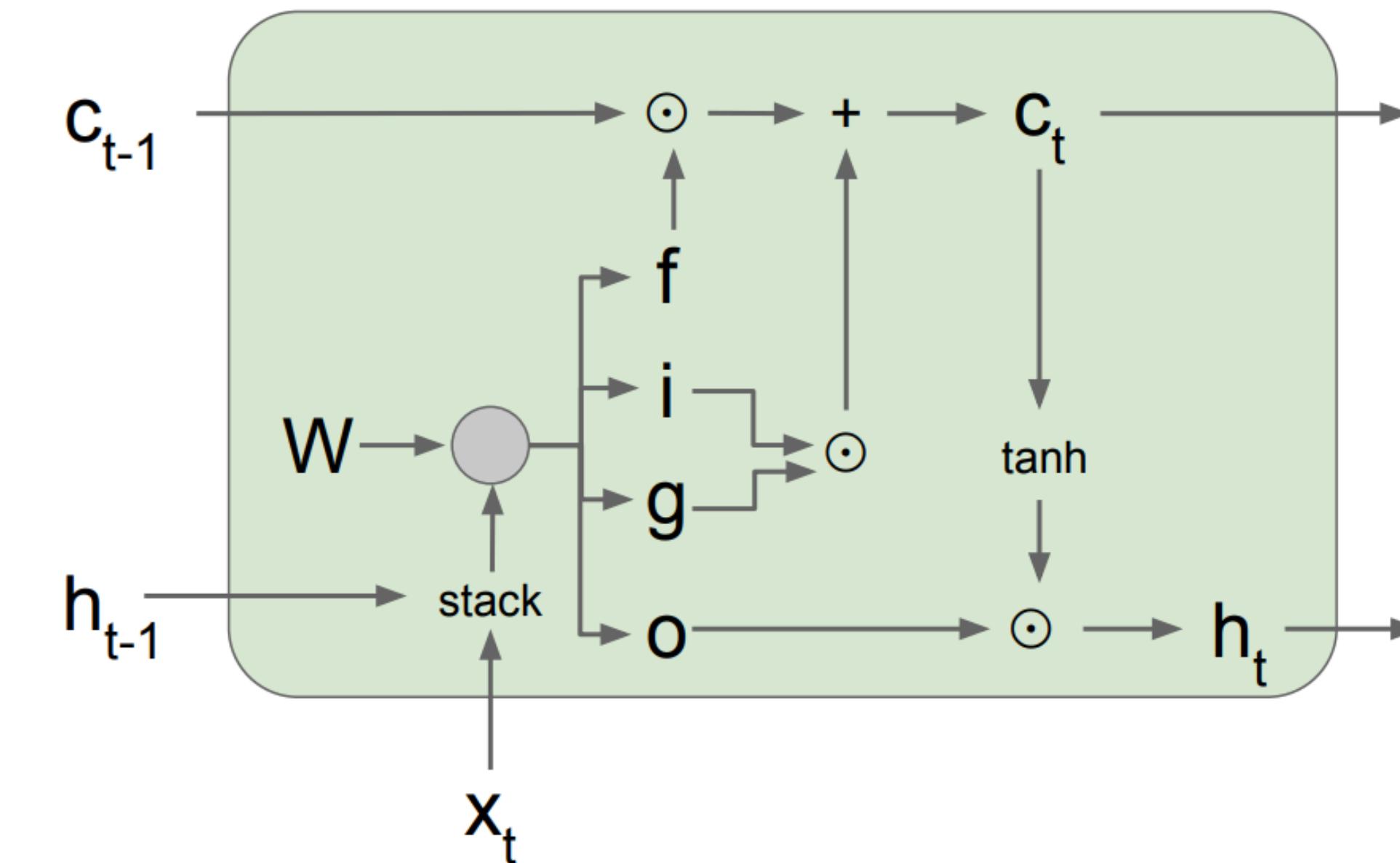
- New memory cell (**what to write**):

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

- 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

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

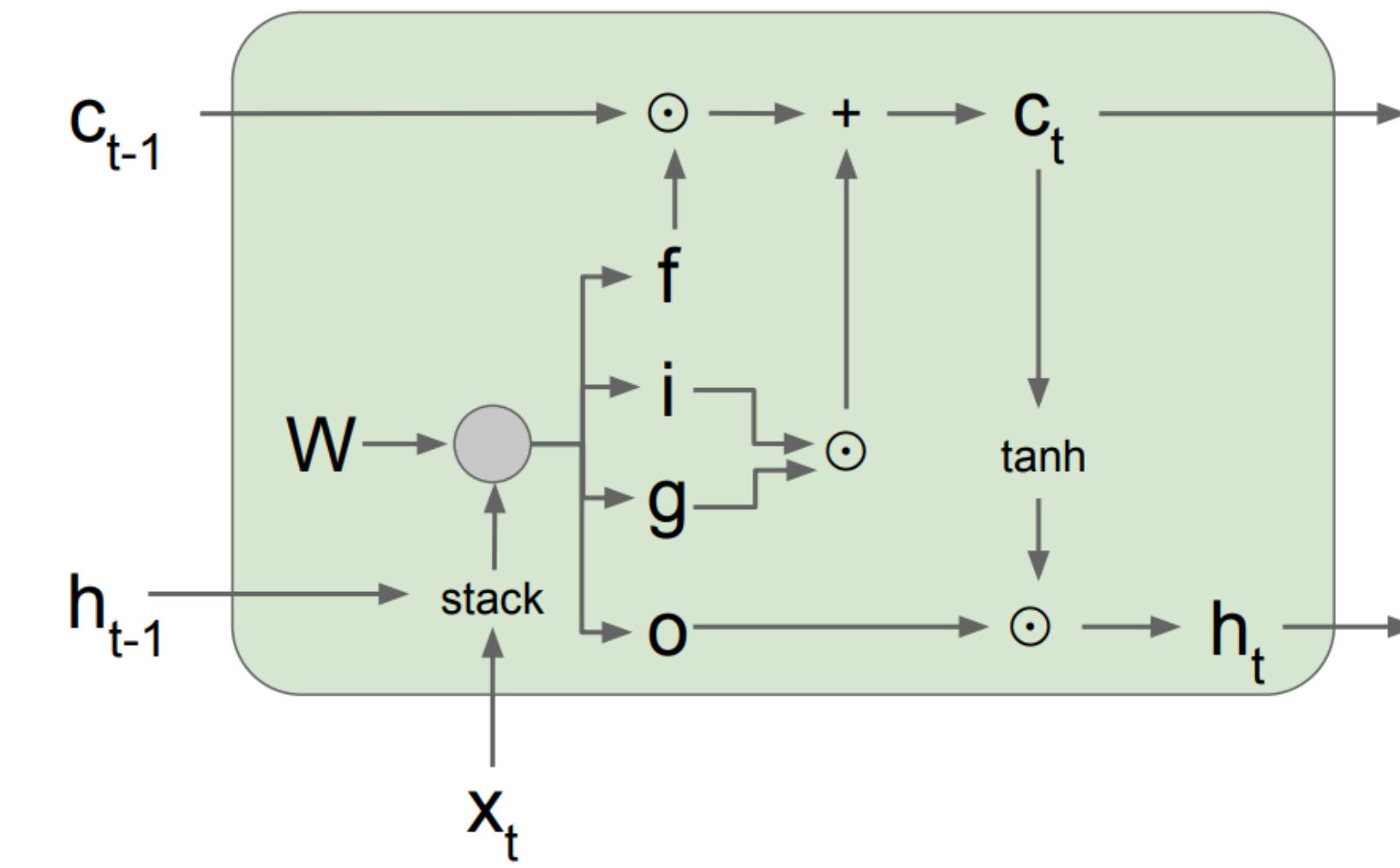
$$\mathbf{W} \in \mathbb{R}^{h \times h}, \mathbf{U} \in \mathbb{R}^{h \times d}, \mathbf{b} \in \mathbb{R}^h$$



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

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

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



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

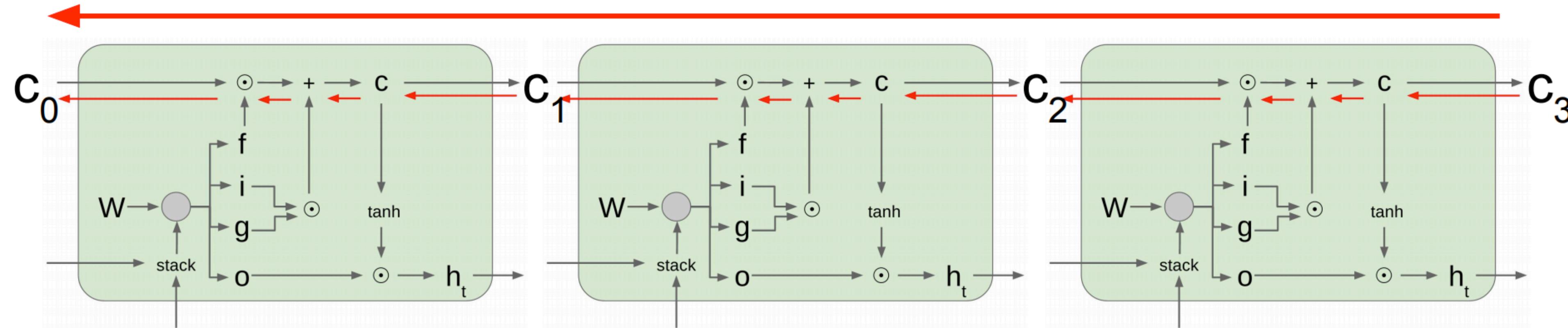
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

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

LSTMs: the formulation

Uninterrupted gradient flow!



- 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:

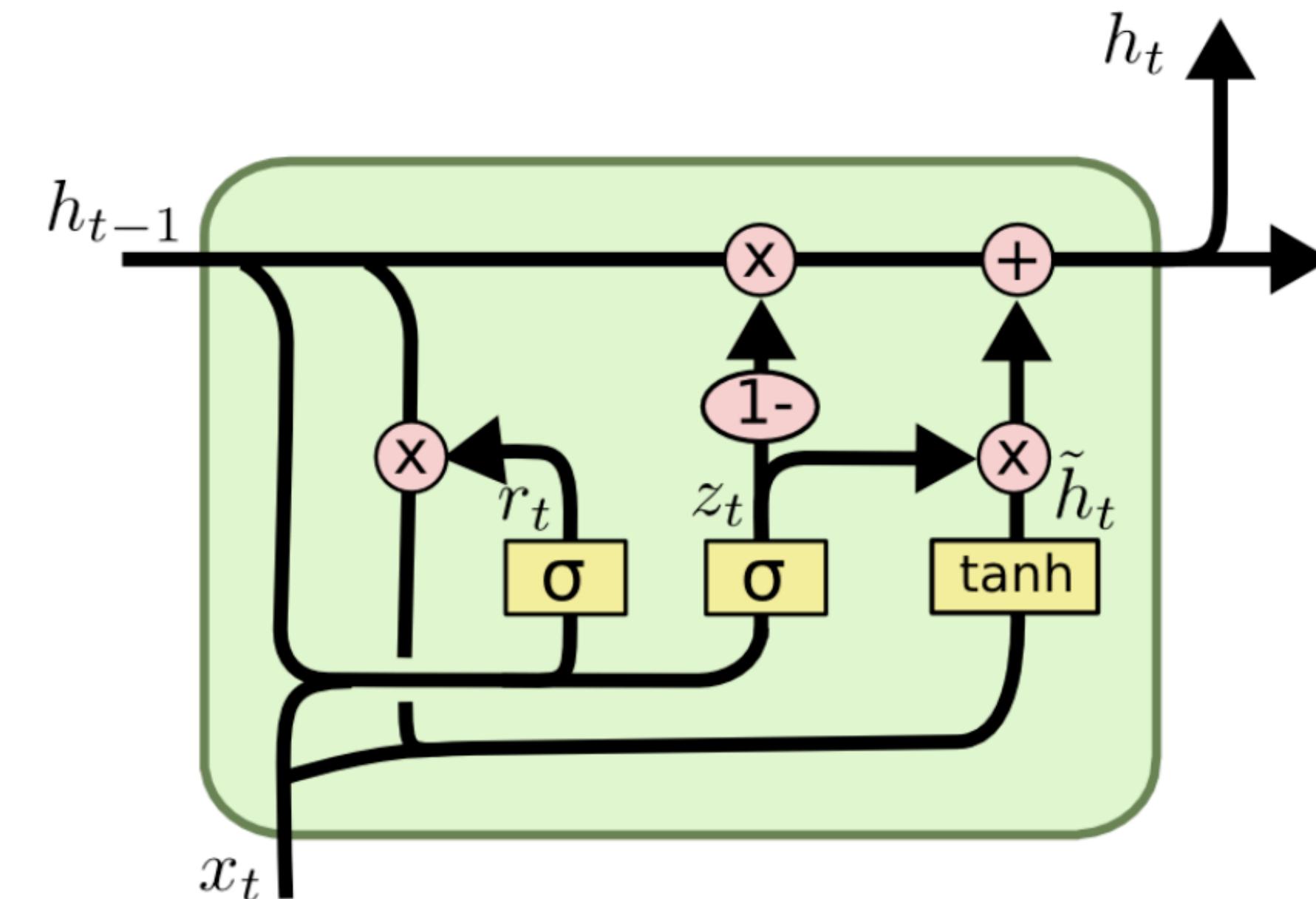
$$\mathbf{z}_t = \sigma(\mathbf{W}^z \mathbf{h}_{t-1} + \mathbf{U}^z \mathbf{x}_t + \mathbf{b}^z)$$

- 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!

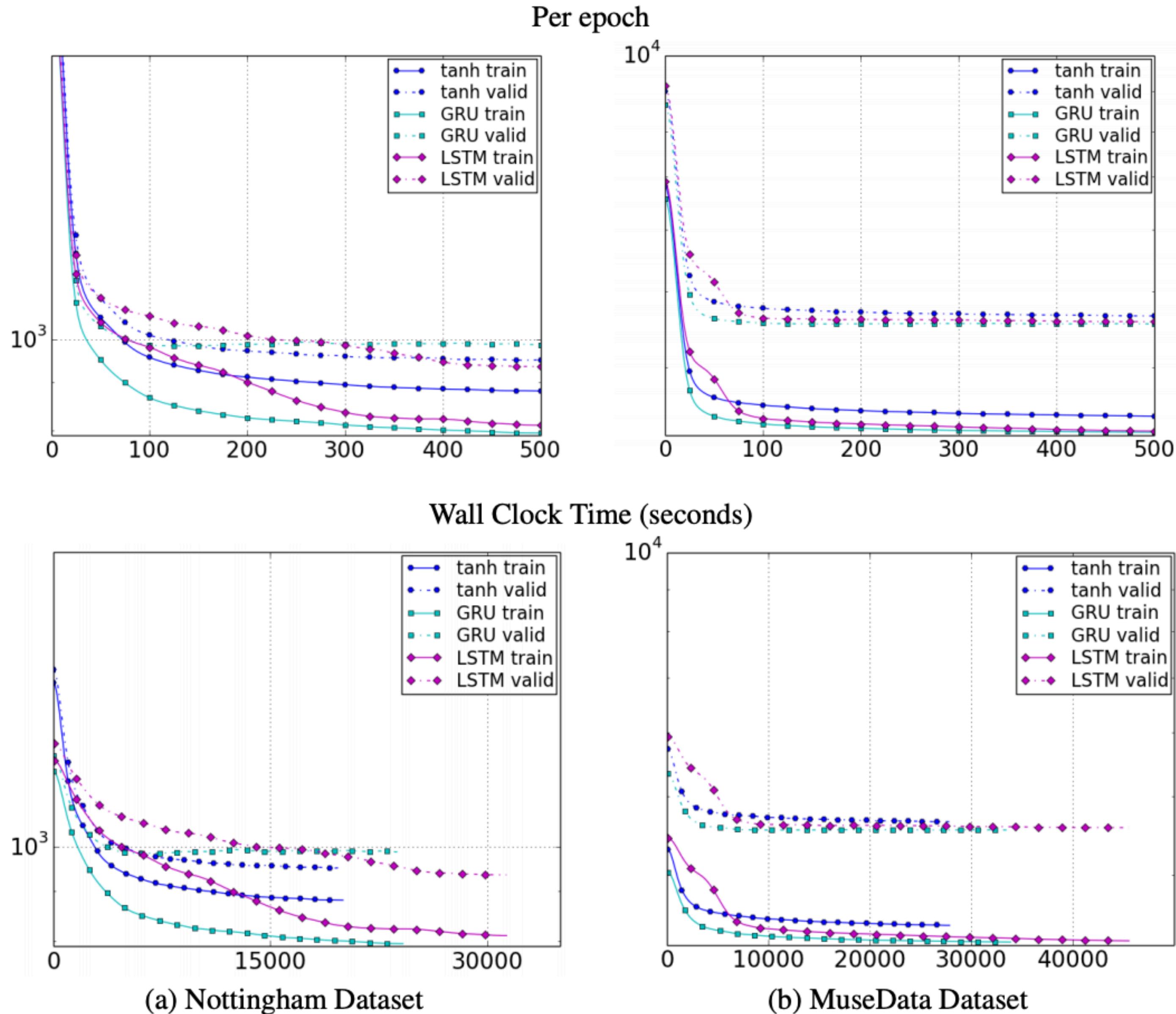


Q: What is the range of the hidden representations \mathbf{h}_t ?

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

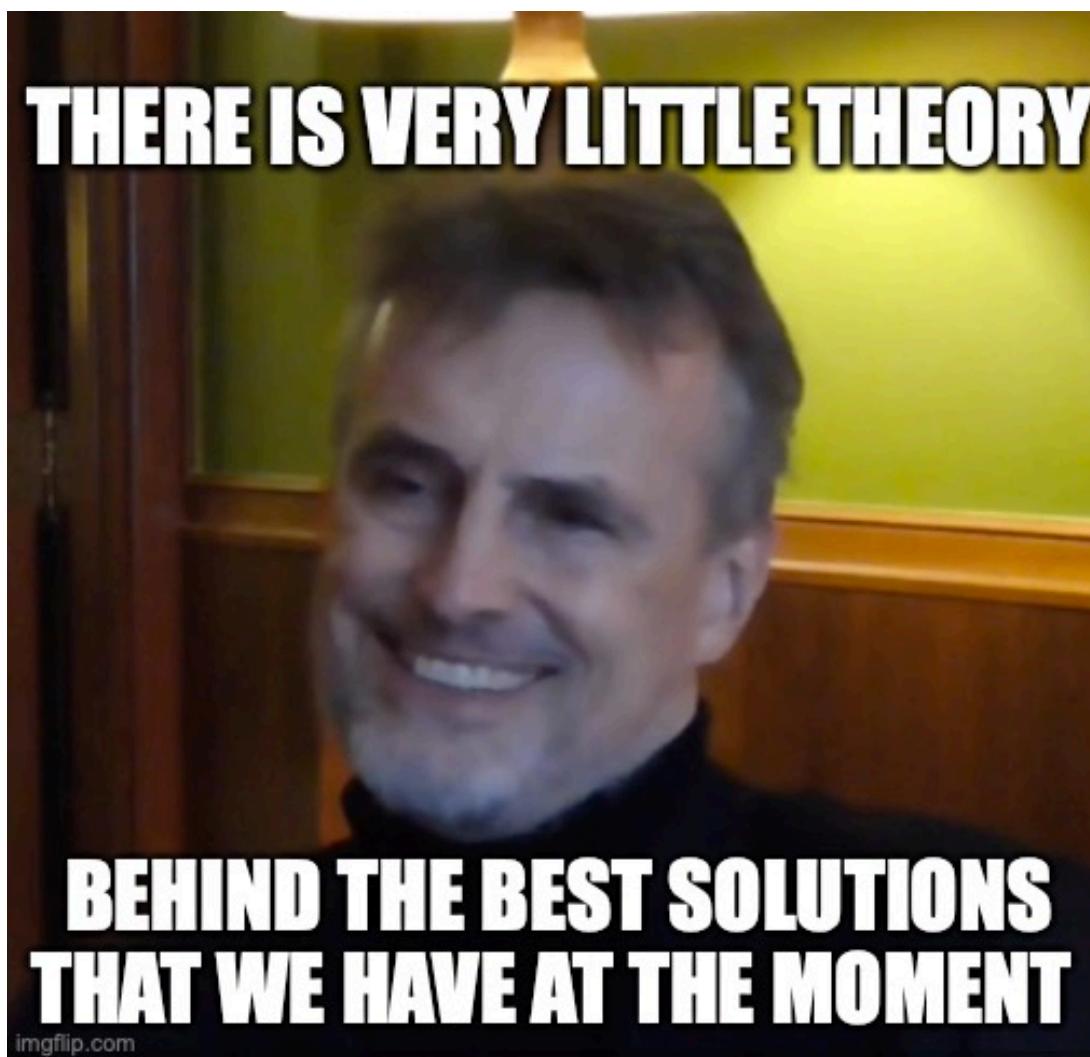
LSTMs vs GRUs

Music modeling

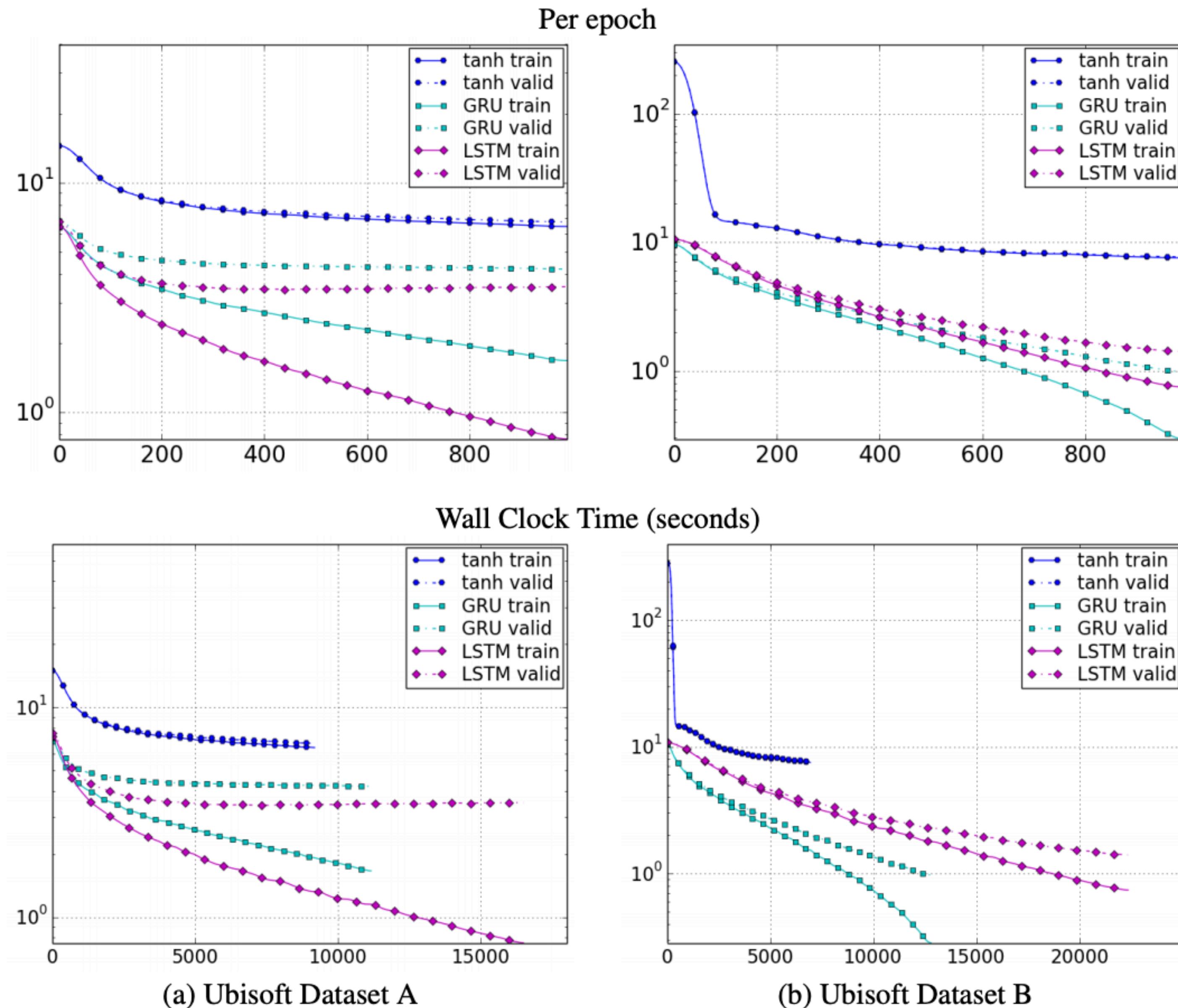


LSTMs vs GRUs

Speech signal modeling



<https://imgflip.com/i/495iim>
(only for fun!!!)



Recap: progress on language models

On the Penn Treebank (PTB) dataset

Metric: perplexity

Simple RNNs + LDA + Kneser-Ney 5-gram + cache

Model	#Param	Validation	Test
Mikolov & Zweig (2012) – RNN-LDA + KN-5 + cache	9M [‡]	-	92.0
Zaremba et al. (2014) – LSTM	20M	86.2	82.7
Gal & Ghahramani (2016) – Variational LSTM (MC)	20M	-	78.6
Kim et al. (2016) – CharCNN	19M	-	78.9
Merity et al. (2016) – Pointer Sentinel-LSTM	21M	72.4	70.9
Grave et al. (2016) – LSTM + continuous cache pointer [†]	-	-	72.1
Inan et al. (2016) – Tied Variational LSTM + augmented loss	24M	75.7	73.2
Zilly et al. (2016) – Variational RHN	23M	67.9	65.4
Zoph & Le (2016) – NAS Cell	25M	-	64.0
Melis et al. (2017) – 2-layer skip connection LSTM	24M	60.9	58.3
Merity et al. (2017) – AWD-LSTM w/o finetune	24M	60.7	58.8
Merity et al. (2017) – AWD-LSTM	24M	60.0	57.3
Ours – AWD-LSTM-MoS w/o finetune	22M	58.08	55.97
Ours – AWD-LSTM-MoS	22M	56.54	54.44
Merity et al. (2017) – AWD-LSTM + continuous cache pointer [†]	24M	53.9	52.8
Krause et al. (2017) – AWD-LSTM + dynamic evaluation [†]	24M	51.6	51.1
Ours – AWD-LSTM-MoS + dynamic evaluation [†]	22M	48.33	47.69

Are LSTMs and GRUs optimal?

MUT1:

$$\begin{aligned} z &= \text{sigm}(W_{xz}x_t + b_z) \\ r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\ h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z) \\ r &= \text{sigm}(x_t + W_{hr}h_t + b_r) \\ h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

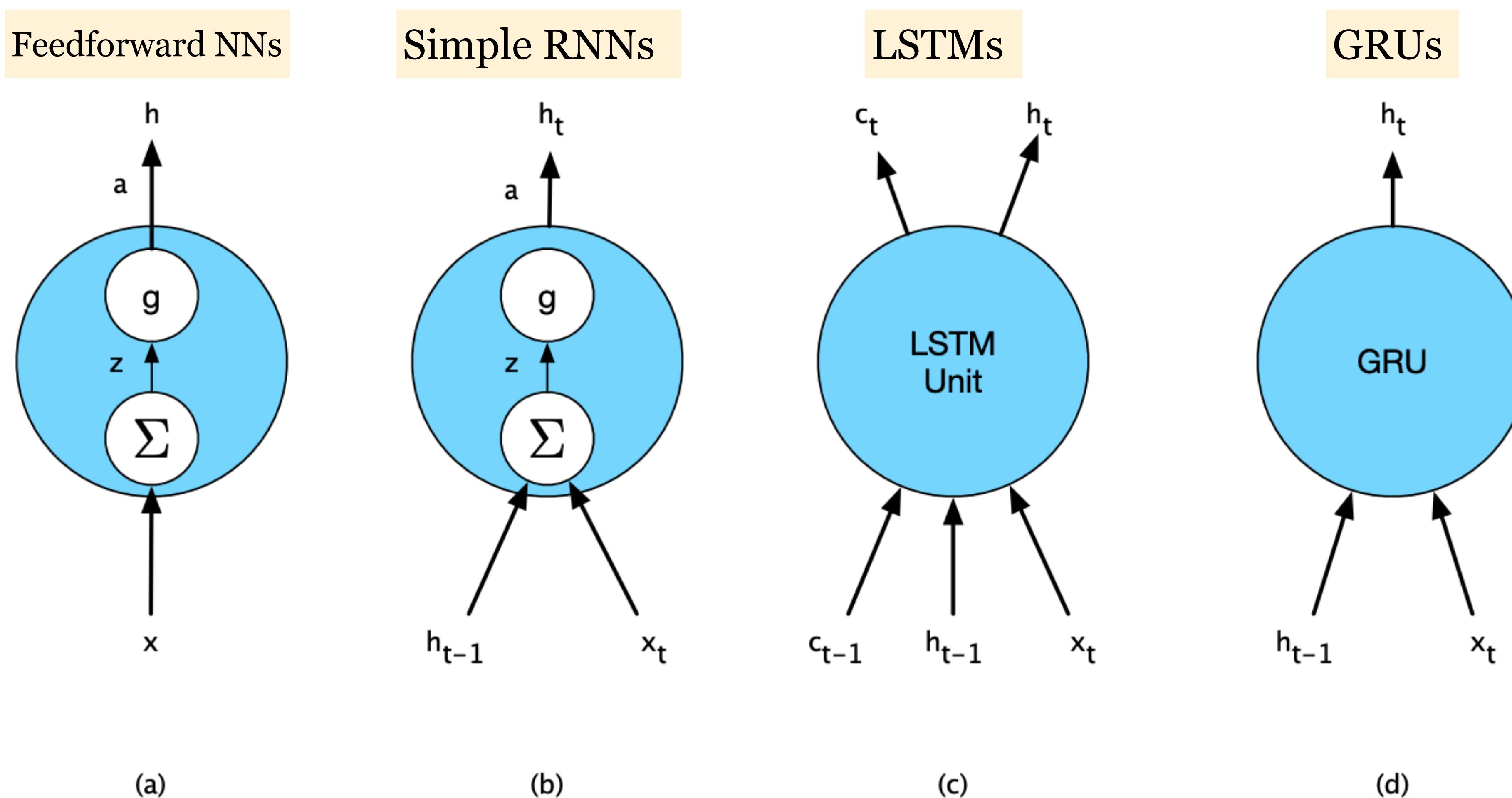
MUT3:

$$\begin{aligned} z &= \text{sigm}(W_{xz}x_t + W_{hz} \tanh(h_t) + b_z) \\ r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\ h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

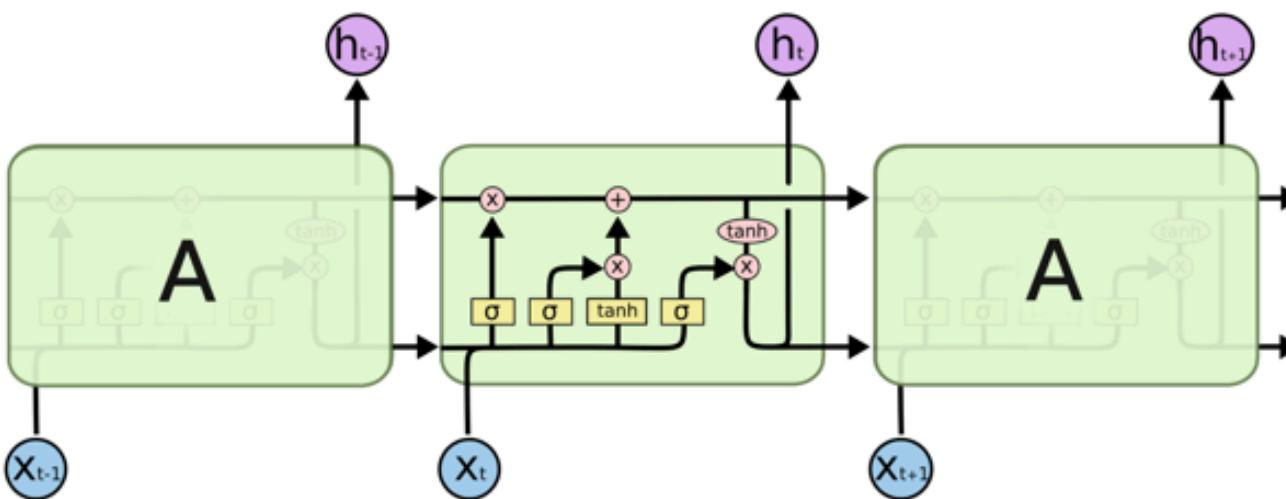
Arch.	Arith.	XML	PTB
Tanh	0.29493	0.32050	0.08782
LSTM	0.89228	0.42470	0.08912
LSTM-f	0.29292	0.23356	0.08808
LSTM-i	0.75109	0.41371	0.08662
LSTM-o	0.86747	0.42117	0.08933
LSTM-b	0.90163	0.44434	0.08952
GRU	0.89565	0.45963	0.09069
MUT1	0.92135	0.47483	0.08968
MUT2	0.89735	0.47324	0.09036
MUT3	0.90728	0.46478	0.09161

Arch.	5M-tst	10M-v	20M-v	20M-tst
Tanh	4.811	4.729	4.635	4.582 (97.7)
LSTM	4.699	4.511	4.437	4.399 (81.4)
LSTM-f	4.785	4.752	4.658	4.606 (100.8)
LSTM-i	4.755	4.558	4.480	4.444 (85.1)
LSTM-o	4.708	4.496	4.447	4.411 (82.3)
LSTM-b	4.698	4.437	4.423	4.380 (79.83)
GRU	4.684	4.554	4.559	4.519 (91.7)
MUT1	4.699	4.605	4.594	4.550 (94.6)
MUT2	4.707	4.539	4.538	4.503 (90.2)
MUT3	4.692	4.523	4.530	4.494 (89.47)

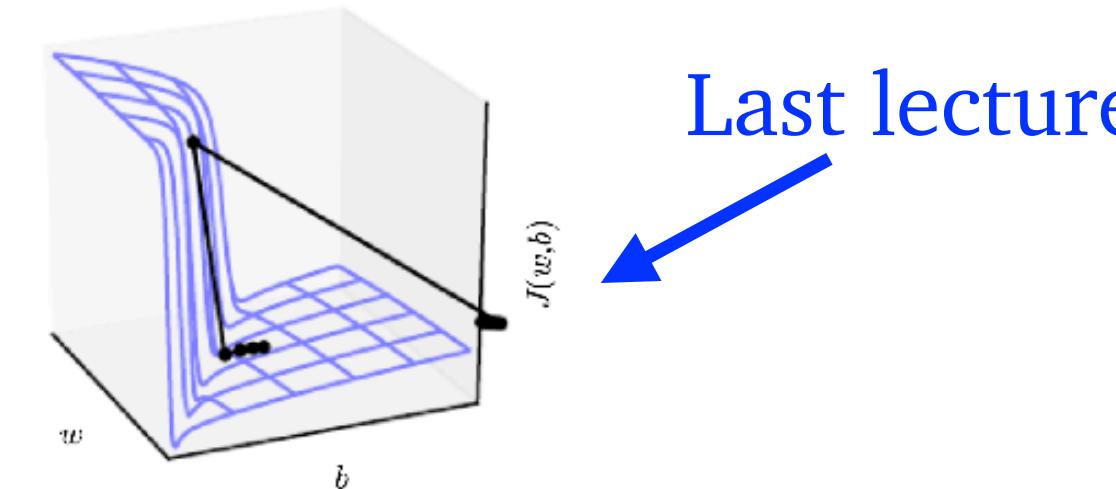
Comparison: FFNNs vs simple RNNs vs LSTMs vs GRUs



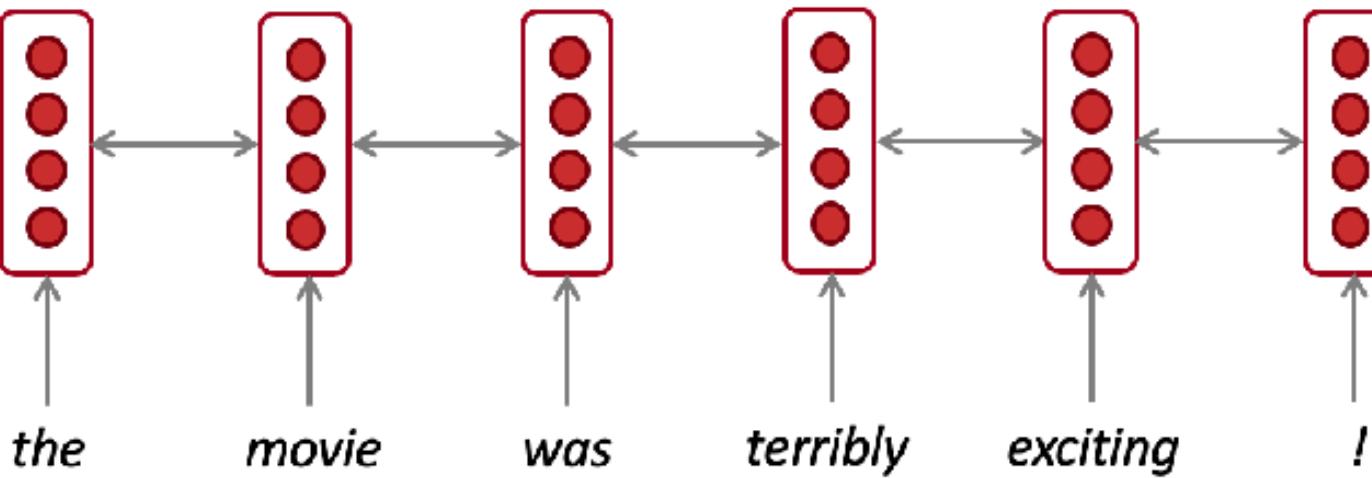
Practical takeaways



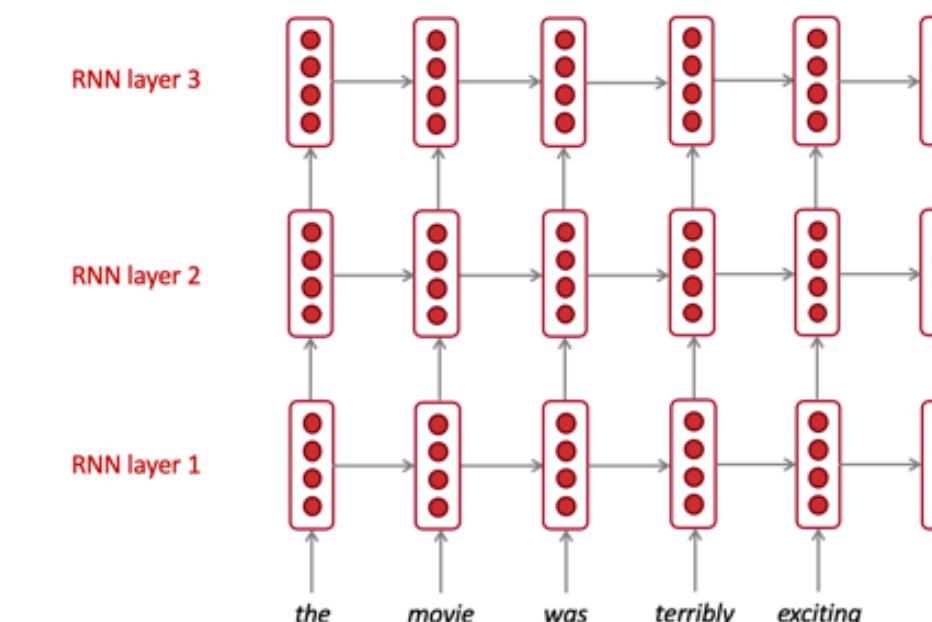
1. LSTMs are powerful



2. Clip your gradients



3. Use bidirectionality
when possible



4. Multi-layer RNNs are more powerful, but
you might need skip connections if it's deep

Neural Turing Machines

2014

Neural Turing Machines

- Explicit solution to address the memory issue in RNNs
- Add a controller network with explicit reading and writing to external memory
- Reading: convex combination of memory vectors $M_t(i)$

$$\mathbf{r}_t \leftarrow \sum_i w_t(i) \mathbf{M}_t(i),$$

- Writing consists of two components:
 - Erase:

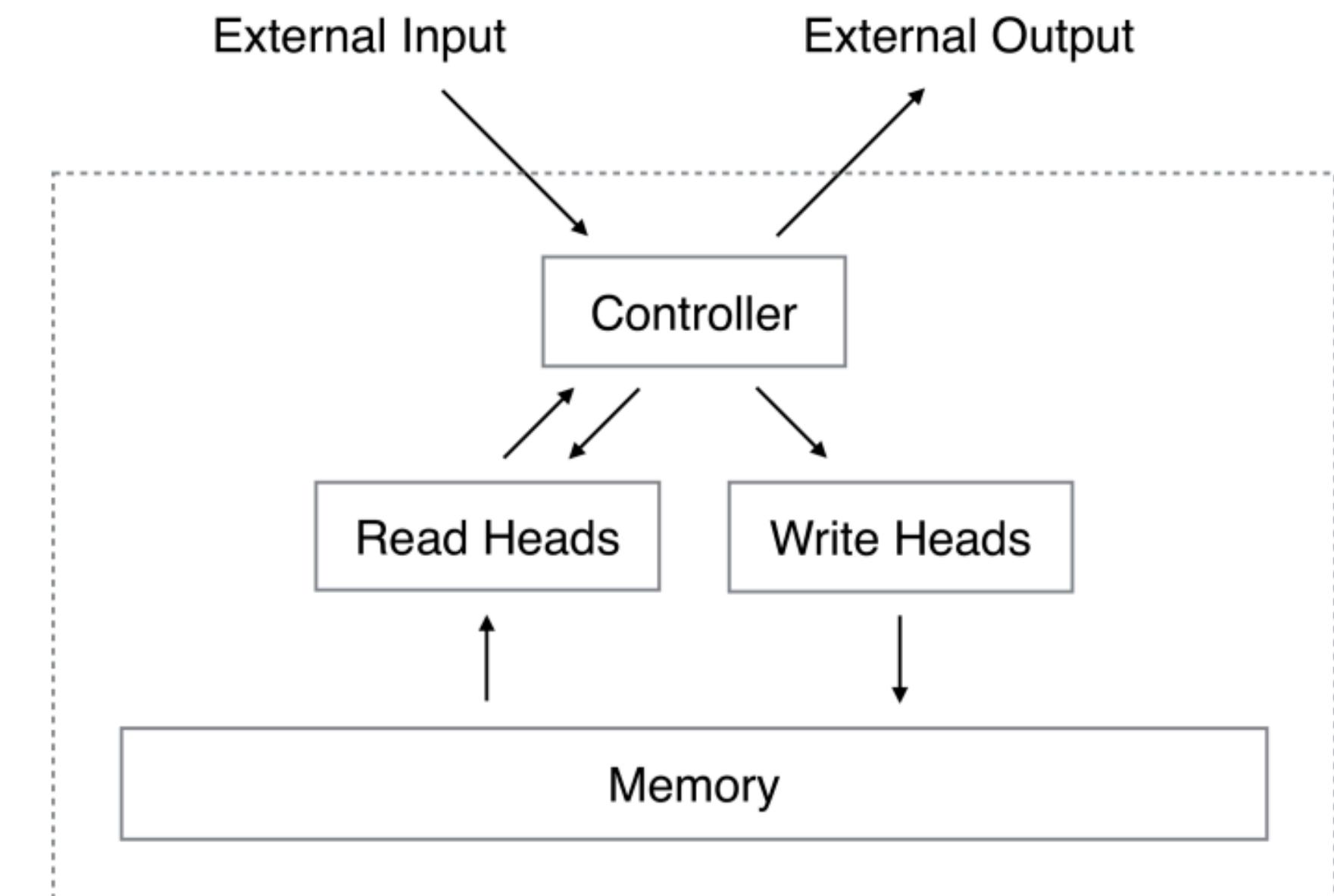
$$\tilde{\mathbf{M}}_t(i) \leftarrow \mathbf{M}_{t-1}(i) [1 - w_t(i) \mathbf{e}_t]$$

- Add:

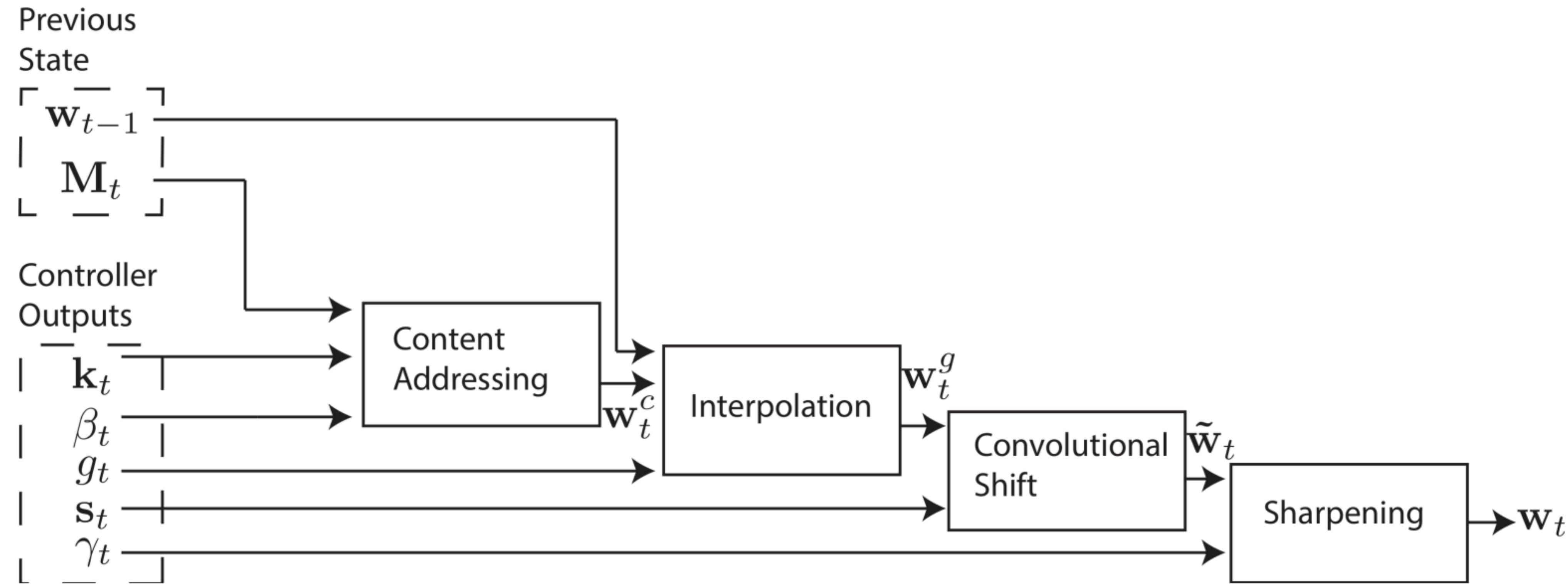
$$\mathbf{M}_t(i) \leftarrow \tilde{\mathbf{M}}_t(i) + w_t(i) \mathbf{a}_t.$$

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

Google DeepMind, London, UK

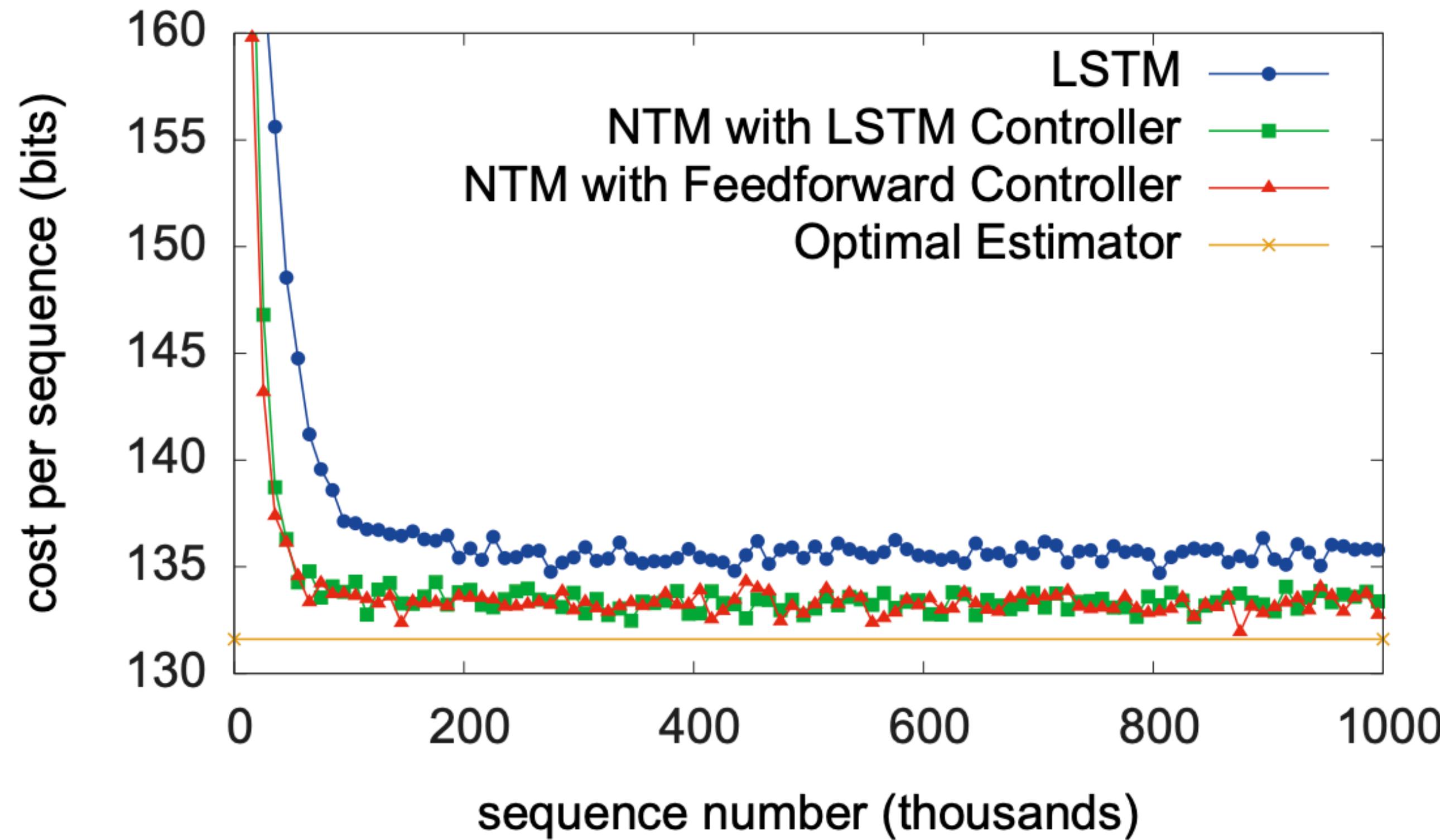


Neural Turing Machines



Use a form of attention to address memory in a end-to-end differentiable manner

Neural Turing Machines



Set of all 6-gram binary sequences
with randomly generated
probabilities

Dynamic N-Gram Learning Curves.

Simple recurrent units (SRU)

Simple Recurrent Units for Highly Parallelizable Recurrence

Tao Lei¹ Yu Zhang² Sida I. Wang^{1,3} Hui Dai¹ Yoav Artzi^{1,4}

¹ASAPP Inc. ²Google Brain ³Princeton University ⁴Cornell University
¹{tao, hd}@asapp.com ²ngyuzh@google.com
³sidaw@cs.princeton.edu ⁴yoav@cs.cornell.edu

2017

$$\begin{aligned}\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{v}_f \odot \mathbf{c}_{t-1} + \mathbf{b}_f) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot (\mathbf{W} \mathbf{x}_t) \\ \mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{v}_r \odot \mathbf{c}_{t-1} + \mathbf{b}_r) \\ \mathbf{h}_t &= \mathbf{r}_t \odot \mathbf{c}_t + (1 - \mathbf{r}_t) \odot \mathbf{x}_t\end{aligned}$$

- Lighter form of recurrent neural networks
- Enable high amounts of parallelism in computation, while maintaining expressivity of recurrent computation
- Use of CUDA kernels to maximize parallel operations

SRU vs other recurrent networks

Model	Size	CR	SUBJ	MR	TREC	MPQA	SST	Time
Best reported results:								
Wang and Manning (2013)		82.1	93.6	79.1	-	86.3	-	-
Kalchbrenner et al. (2014)		-	-	-	93.0	-	86.8	-
Kim (2014)		85.0	93.4	81.5	93.6	89.6	88.1	-
Zhang and Wallace (2017)		84.7	93.7	81.7	91.6	89.6	85.5	-
Zhao et al. (2015)		86.3	95.5	83.1	92.4	93.3	-	-
Our setup (default Adam, fixed word embeddings):								
CNN	360k	83.1±1.6	92.7±0.9	78.9±1.3	93.2±0.8	89.2±0.8	85.1±0.6	417
LSTM	352k	82.7±1.9	92.6±0.8	79.8±1.3	93.4±0.9	89.4±0.7	88.1±0.8	2409
QRNN (k=1)	165k	83.5±1.9	93.4±0.6	82.0±1.0	92.5±0.5	90.2±0.7	88.2±0.4	345
QRNN (k=1) + highway	204k	84.0±1.9	93.4±0.8	82.1±1.2	93.2±0.6	89.6±1.2	88.9±0.2	371
SRU (2 layers)	204k	84.9±1.6	93.5±0.6	82.3±1.2	94.0±0.5	90.1±0.7	89.2±0.3	320
SRU (4 layers)	303k	85.9±1.5	93.8±0.6	82.9±1.0	94.8±0.5	90.1±0.6	89.6±0.5	510
SRU (8 layers)	502k	86.4±1.7	93.7±0.6	83.1±1.0	94.7±0.5	90.2±0.8	88.9±0.6	879

