Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

Lecture 3.2: Recurrent Networks

Louis-Philippe Morency

* Original version co-developed with Tadas Baltrusaitis

Lecture Objectives

- Sequential modeling with convolutional networks
- Word representations
  - Distributional hypothesis
  - Learning neural representations
- Language models and sequence modeling tasks
- Recurrent neural networks
  - Gated recurrent neural networks
  - Long Short-Term Memory (LSTM) model
    - Multi-view LSTM
  - Backpropagation through time

Sequential Modeling with Convolutional Networks

Modeling Temporal and Sequential Data

How to represent a video sequence?

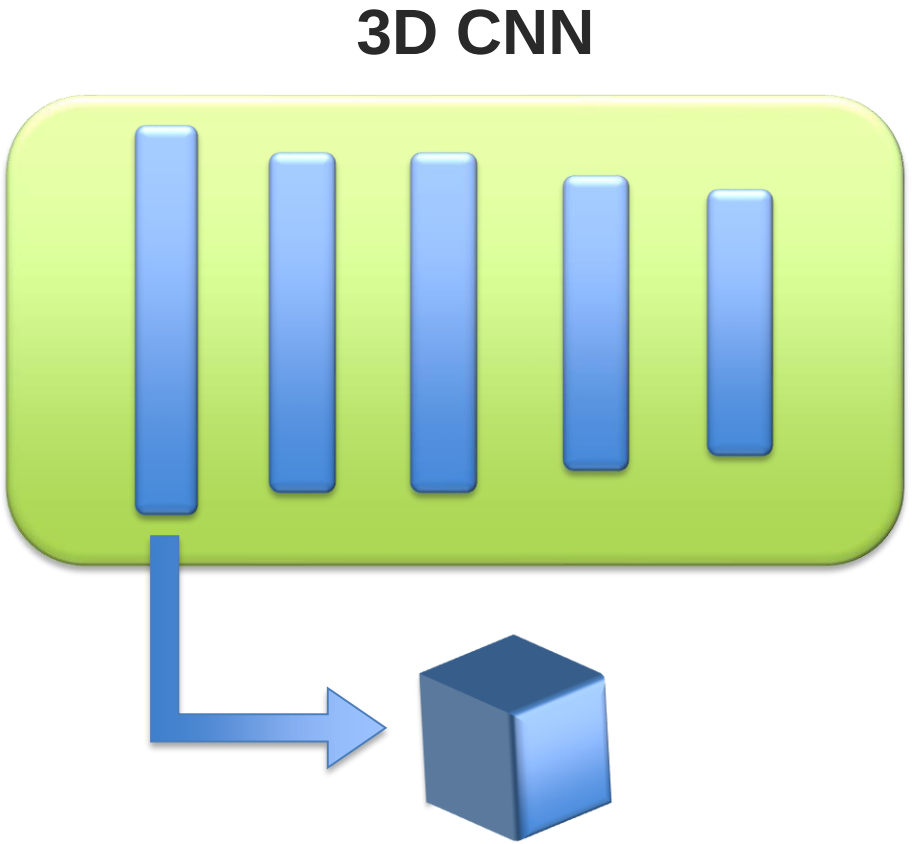
One option: Recurrent Neural Networks

(more about this on Thursday)

3D CNN

Input as a 3D tensor

(stacking video images)



First layer with 3D kernels

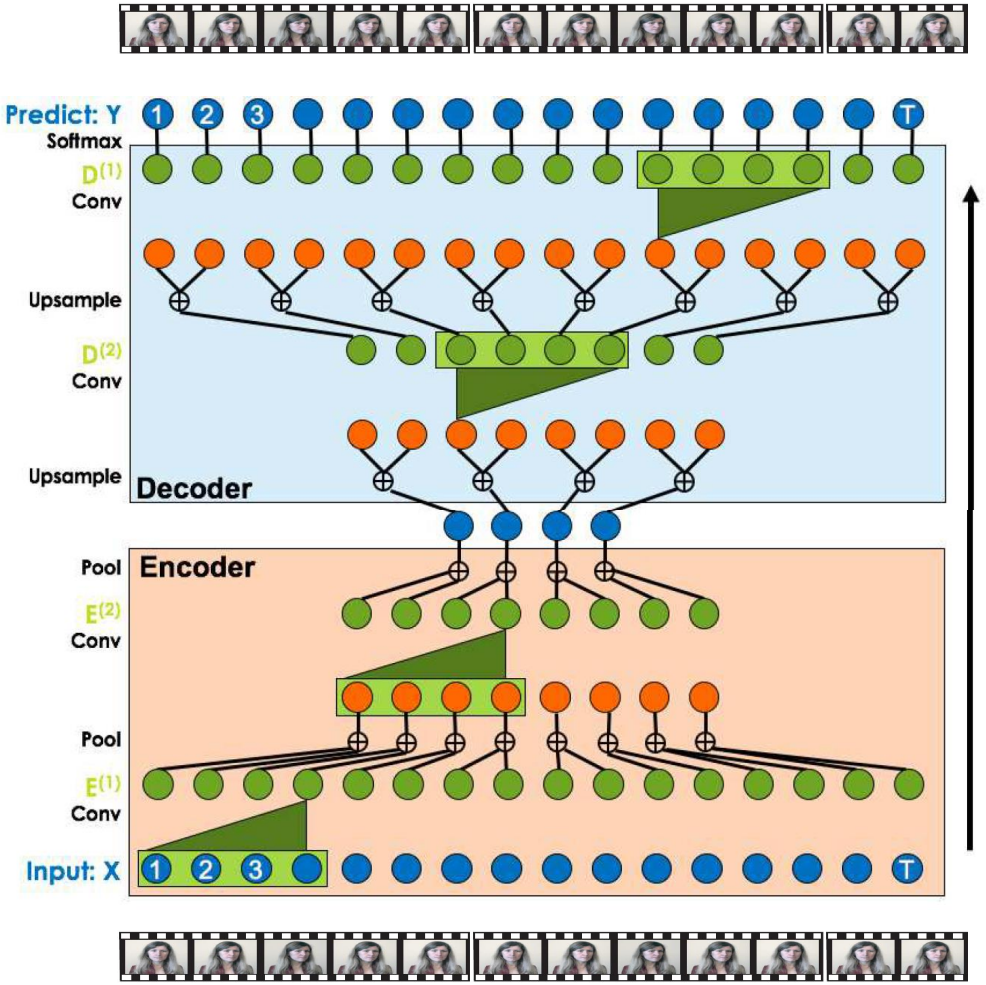
Time-Delay Neural Network

Alexander Waibel, Phoneme Recognition Using Time-Delay Neural Networks, SP87-100, Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE), December, 1987,Tokyo, Japan.

Temporal Convolution Network (TCN) [Lea et al., CVPR 2017]

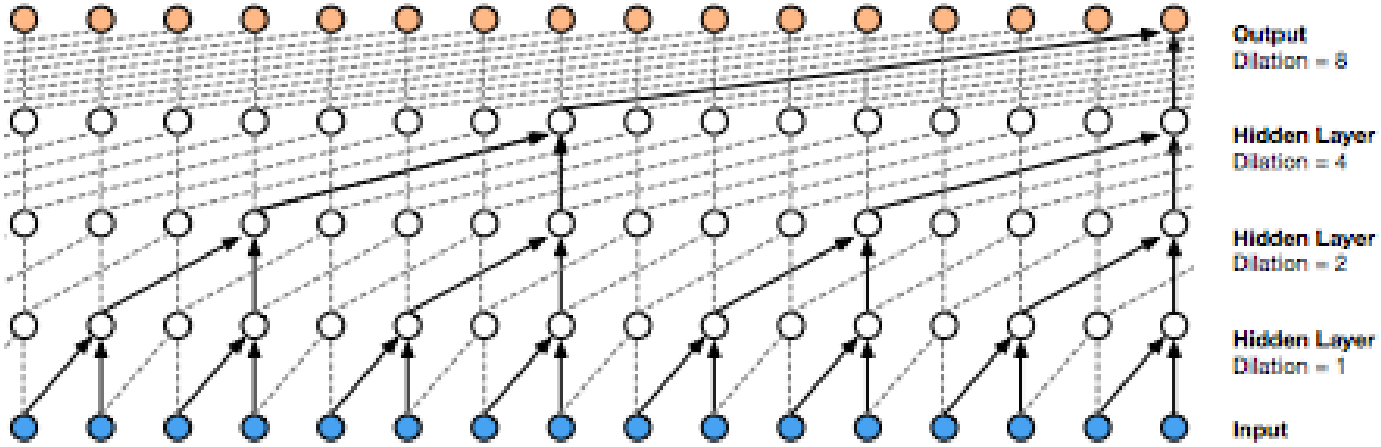
Decoder

Encoder



Dilated TCN Model [Lea et al., CVPR 2017]

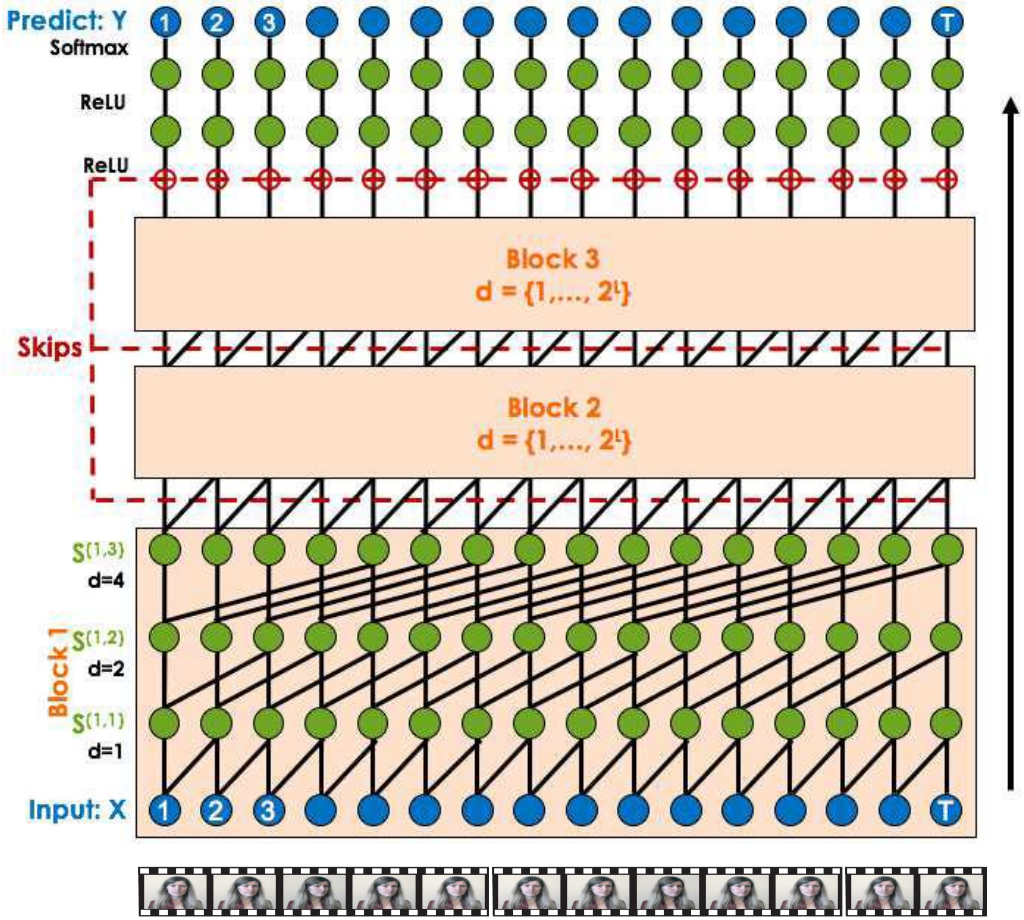
Dilated Convolutions



Dilation of 4: Step size of 4 when convoluting

+ Skip connections to help with deep modeling

Dilated TCN Models [Lea et al., CVPR 2017]



Representing Words: Distributed Semantics

What is the meaning of “bardiwac”?

- He handed her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia’s sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

⇒ bardiwac is a heavy red alcoholic beverage made from grapes

The Distributional Hypothesis

- Distribution Hypothesis (DH) [Lenci 2008]
  - At least certain aspects of the meaning of lexical expressions depend on their distributional properties in the linguistic contexts
  - The degree of semantic similarity between two linguistic expressions α and β is a function of the similarity of the linguistic contexts in which α and β can appear
- Weak and strong DH
  - Weak view as a quantitative method for semantic analysis and lexical resource induction
  - Strong view as a cognitive hypothesis about the form and origin of semantic representations; assuming that word distributions in context play a specific *causal role* in forming meaning representations.

Geometric interpretation

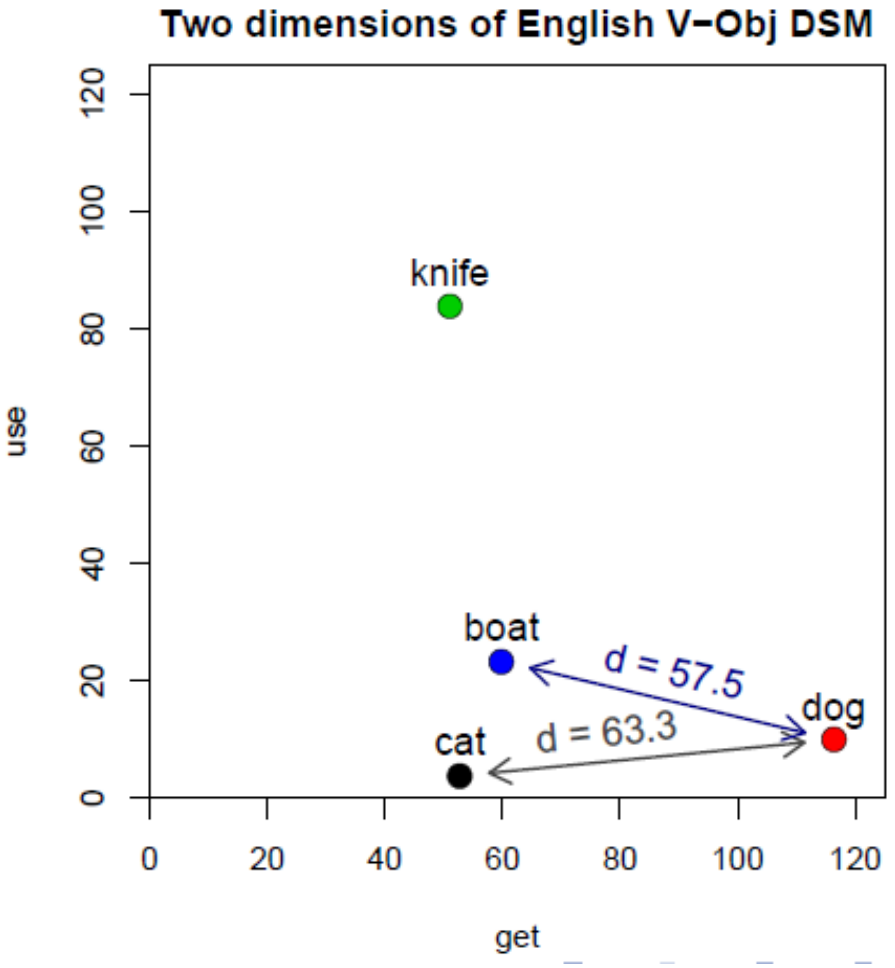
- row vector $\mathbf{x}_{dog}$ describes usage of word *dog* in the corpus
- can be seen as coordinates of point in *n*-dimensional Euclidean space R^n

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	115	83	10	42	33	17
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
banana	11	2	2	0	18	0

co-occurrence matrix M

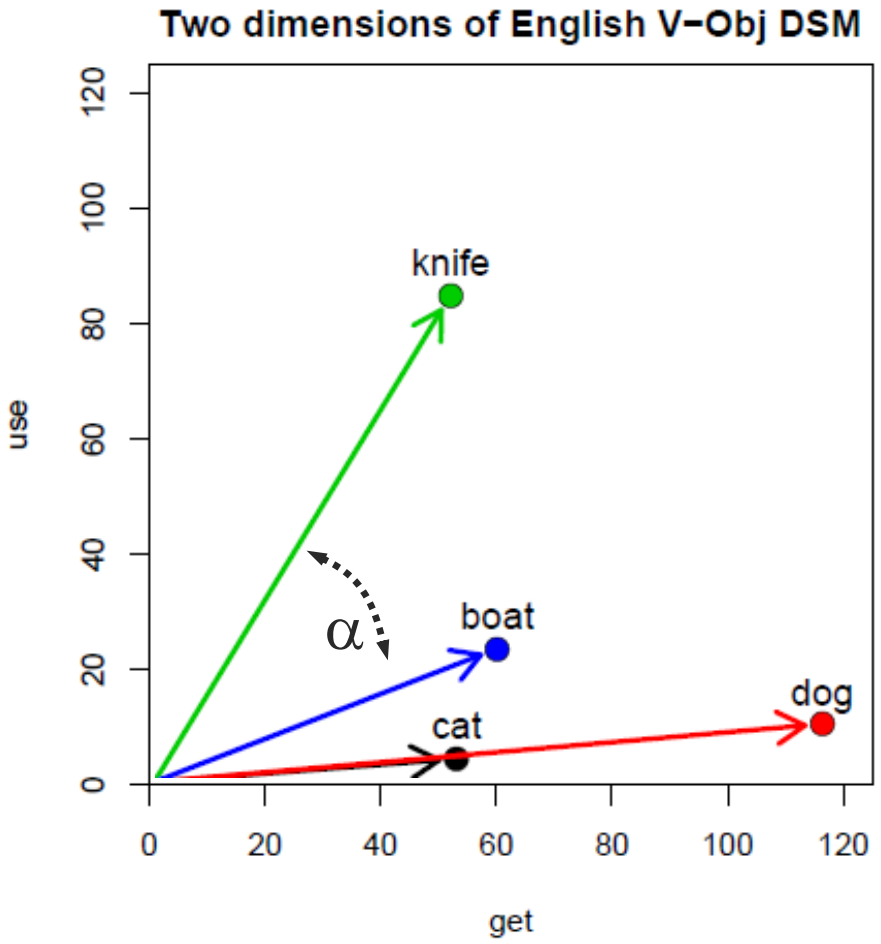
Distance and similarity

- illustrated for two dimensions: *get* and *use*: $\mathbf{x}_{dog}$ = (115, 10)
- similarity = spatial proximity (Euclidean distance)
- location depends on frequency of noun ($f_{dog} \approx 2.7 \cdot f_{cat}$)

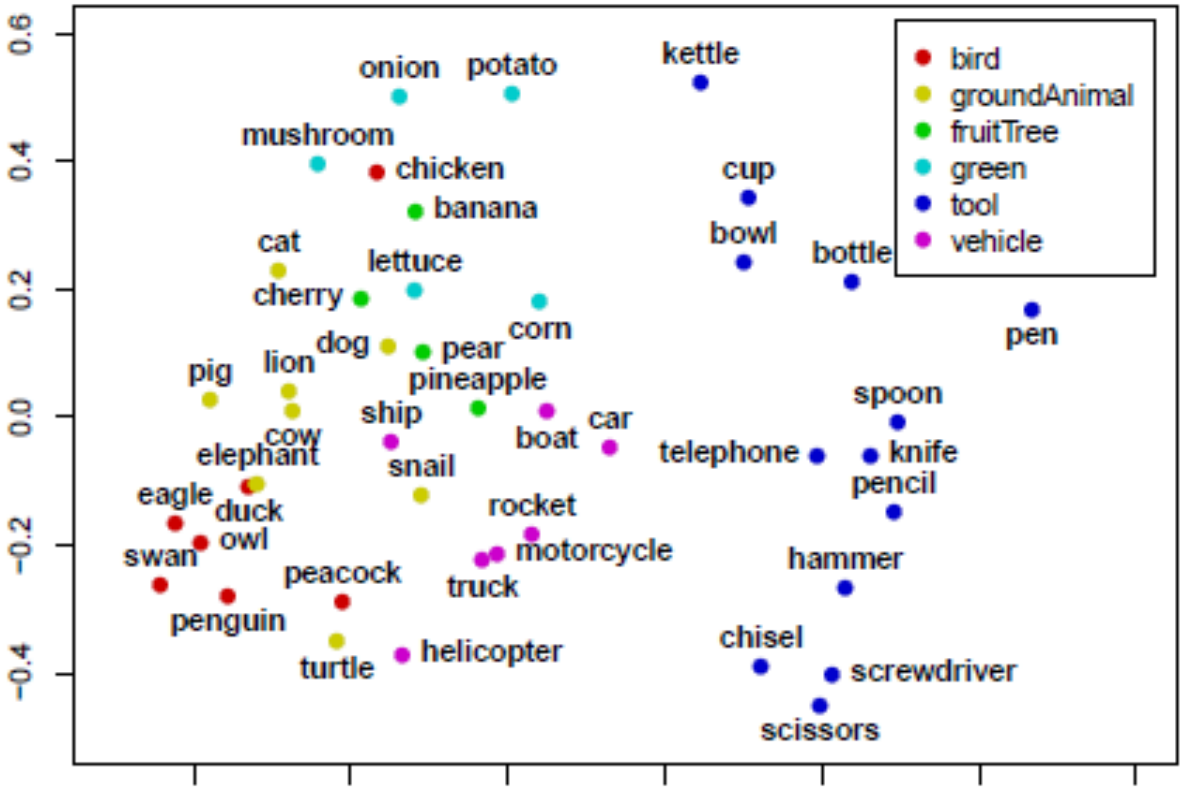


Angle and similarity

- direction more important than location
- normalise "length" $||\mathbf{x}_{dog}||$ of vector
- or use angle α as distance measure



Semantic maps



Learning Neural Word Representations

How to learn neural word representations?

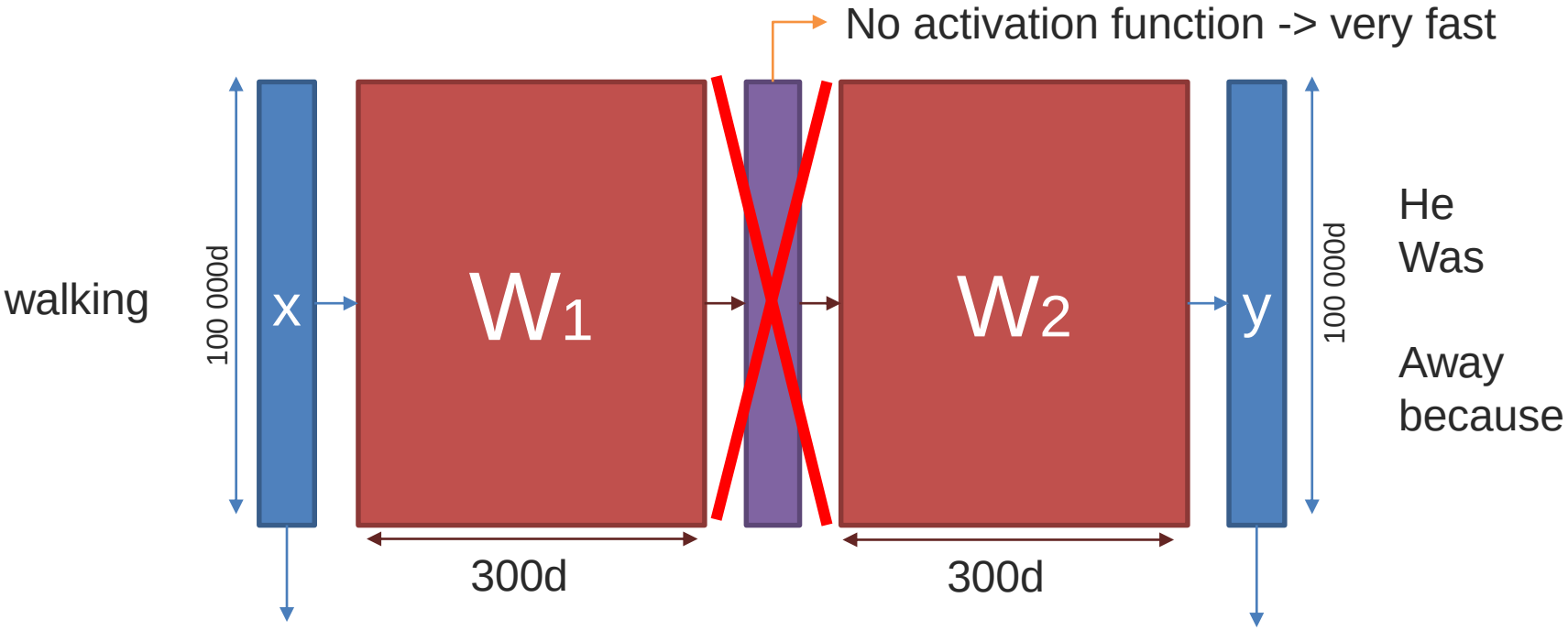
Distribution hypothesis: Approximate the word meaning by its surrounding words

Words used in a similar context will lie close together

Instead of capturing co-occurrence counts directly, predict surrounding words of every word

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

How to learn neural word representations?



Word2vec algorithm: https://code.google.com/p/word2vec/

How to use these word representations

If we would have a vocabulary of 100 000 words:

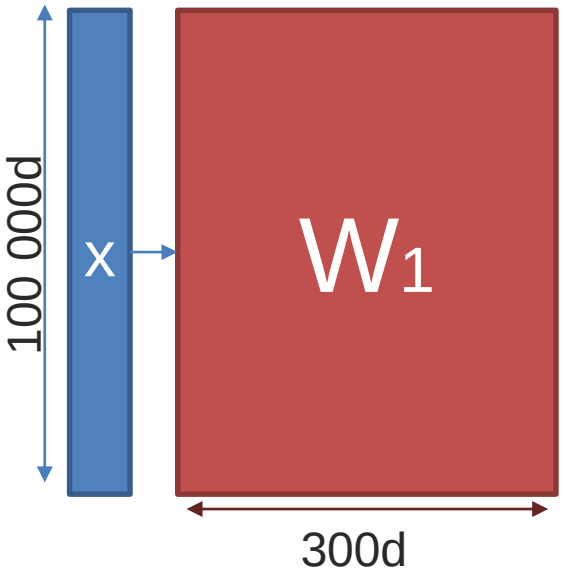
Classic NLP: 100 000 dimensional vector

Walking: [0; 0; 0; 0;….; 0; 0; 1; 0;…; 0; 0]

Running: [0; 0; 0; 0;….; 0; 0; 0; 0;…; 1; 0]

Similarity = 0.0

Transform: x'=x*W



Goal: 300 dimensional vector

Walking: [0,1; 0,0003; 0;….; 0,02; 0.08; 0,05]

Running: [0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Similarity = 0.9

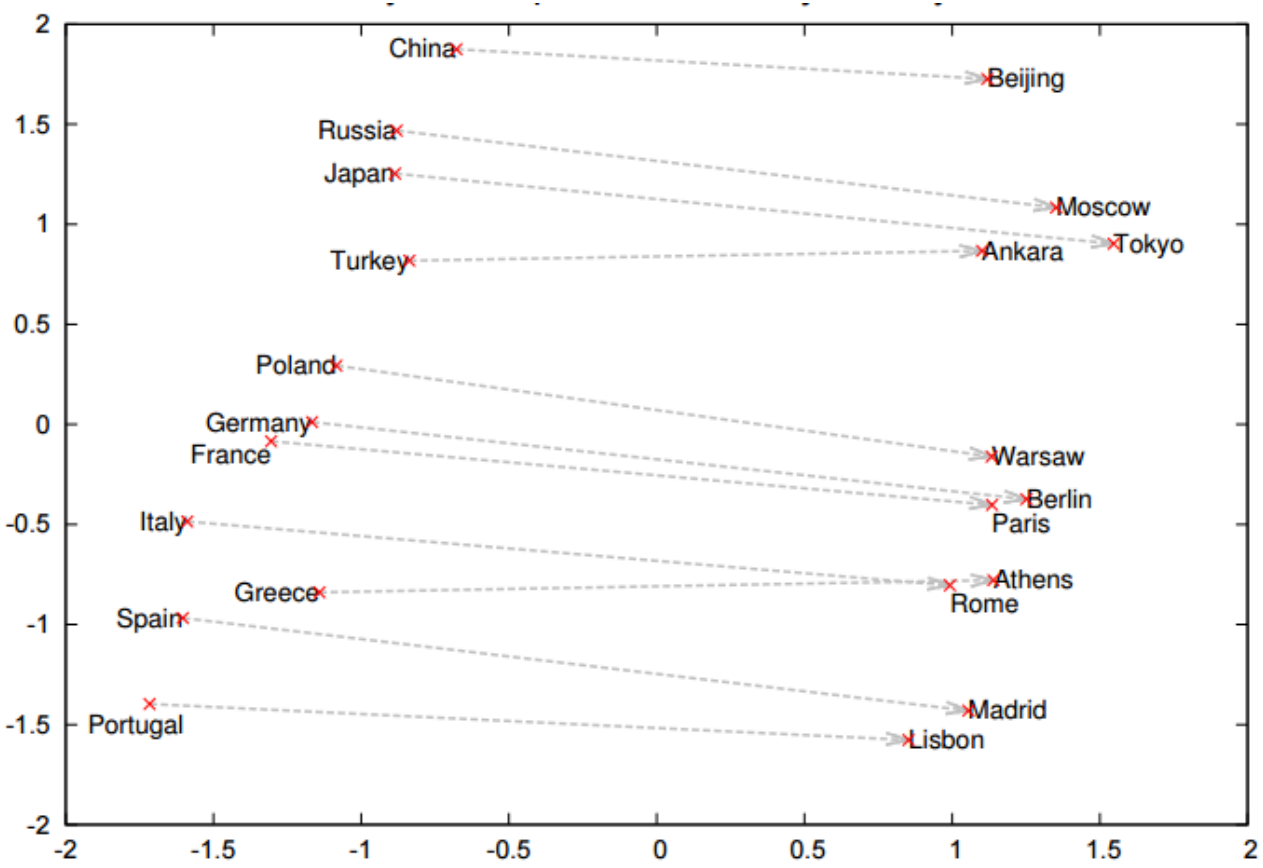
Vector space models of words

- While learning these word representations, we are actually building a vector space in which all words reside with certain relationships between them
- Encodes both syntactic and semantic relationships
- This vector space allows for algebraic operations:

$$\text{Vec(king)} - \text{vec(man)} + \text{vec(woman)} \approx \text{vec(queen)}$$

Why linear algebra is working?

Vector space models of words: semantic relationships



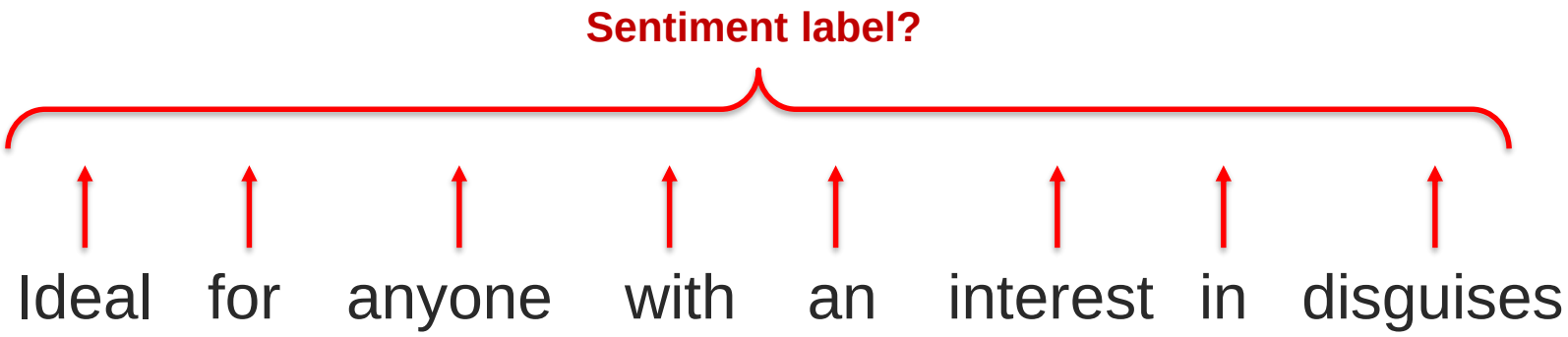
Trained on the Google news corpus with over 300 billion words

Language Sequence Modeling Tasks

Sequence Modeling: Sequence Label Prediction

Prediction →

Sentiment ?
(positive or negative)



Sequence Modeling: Sequence Prediction

Prediction →

Part-of-speech ?
(noun, verb,…)

POS?	POS?	POS?	POS?	POS?	POS?	POS?	POS?
Ideal	for	anyone	with	an	interest	in	disguises

Sequence Modeling: Sequence Representation

Learning →

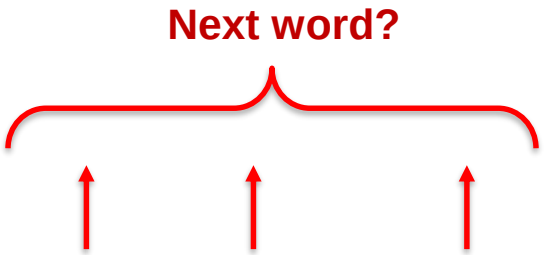
Sequence representation

[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

Sequence Modeling: Language Model

Prediction → Language Model



Ideal for anyone with an interest in disguises

Application: Speech Recognition

$$\arg\max_{wordsequence} P(wordsequence \mid acoustics) =$$

$$\arg\max_{wordsequence} \frac{P(acoustics \mid wordsequence) \times P(wordsequence)}{P(acoustics)}$$

$$\arg\max_{wordsequence} P(acoustics \mid wordsequence) \times P(wordsequence)$$

Language model

Application: Language Generation

Embedding

[0,1;
0,0004;
….;
0.09;
0,05]

Generation →

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

Example: Image captioning

→

[0,1;
0,0004;
….;
0.09;
0,05]

→

The man at bat readies to swing at the pitch while the umpire looks on.

N-Gram Language Model Formulations

- Word sequences

$$w_1^n = w_1 \ldots w_n$$

- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- Bigram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

- N-gram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

Evaluating Language Model: Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

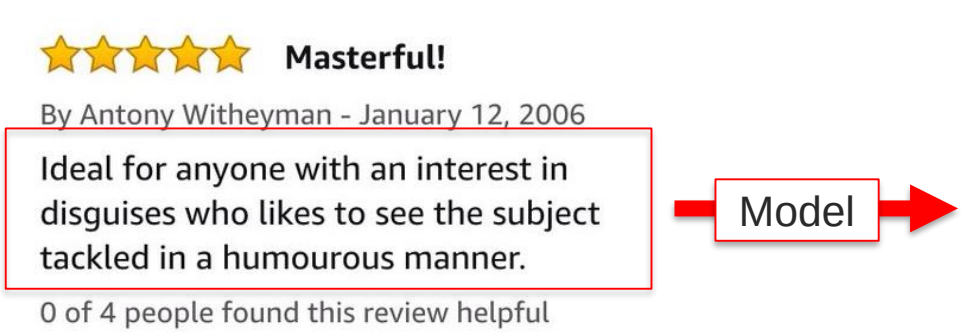
Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Challenges in Sequence Modeling



- Part-of-speech ?
  (noun, verb,…)
- Sentiment ?
  (positive or negative)
- Language Model
- Sequence representation

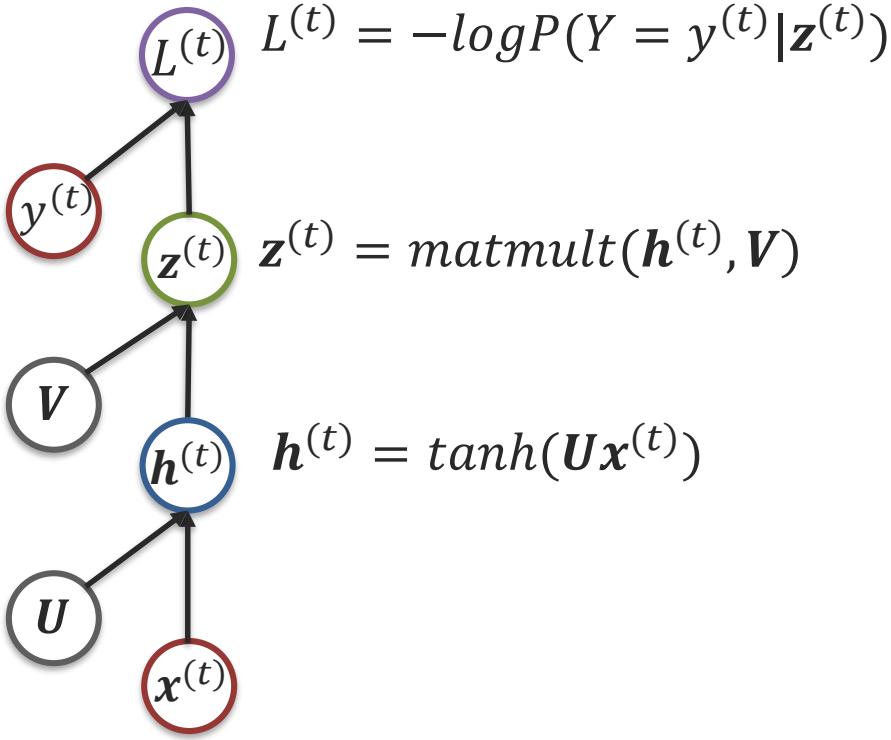
Main Challenges:

- Sequences of variable lengths (e.g., sentences)
- Keep the number of parameters at a minimum
- Take advantage of possible redundancy

Recurrent Neural Networks

Recurrent Neural Network

Feedforward Neural Network



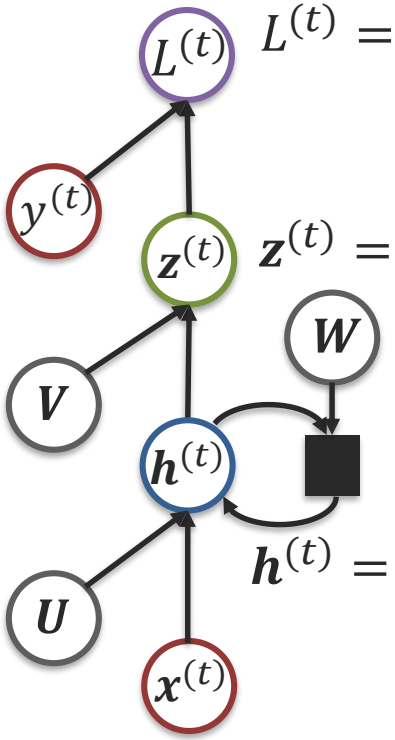
$$L^{(t)} = -logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{U}\boldsymbol{x}^{(t)})$$

Recurrent Neural Networks

$$L = \sum_t L^{(t)}$$



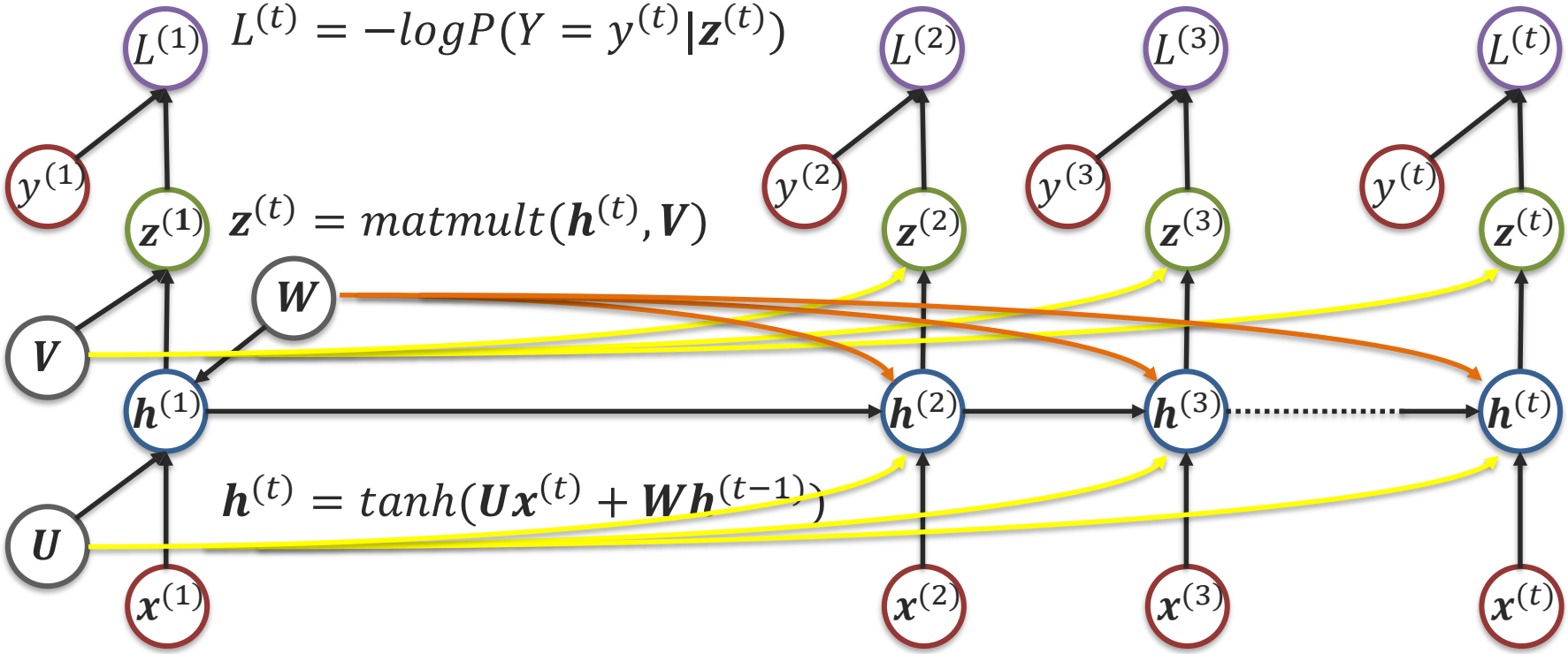
$$L^{(t)} = -logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

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

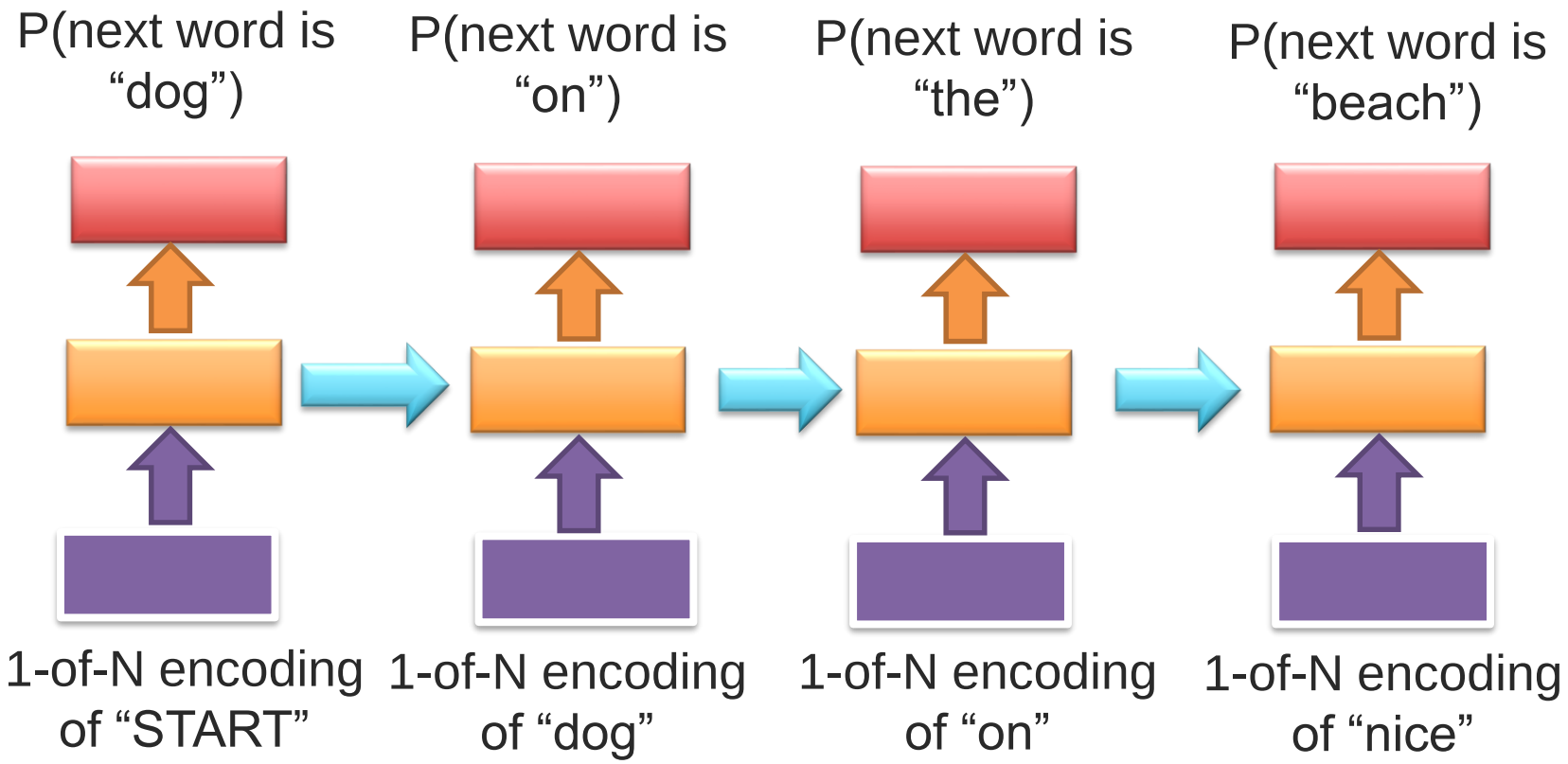
Recurrent Neural Networks - Unrolling

$$L = \sum_t L^{(t)}$$



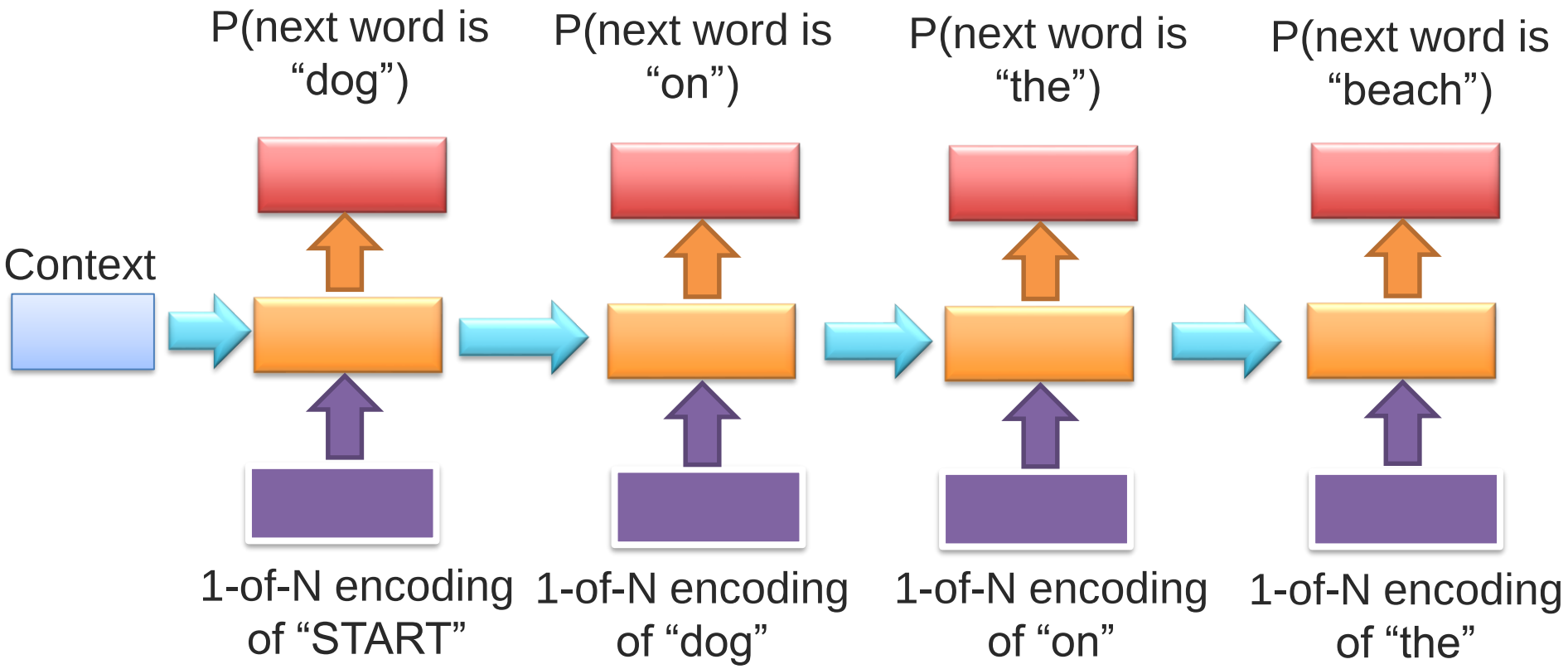
Same model parameters are used for all time parts.

RNN-based Language Model



- Models long-term information

RNN-based Sentence Generation (Decoder)

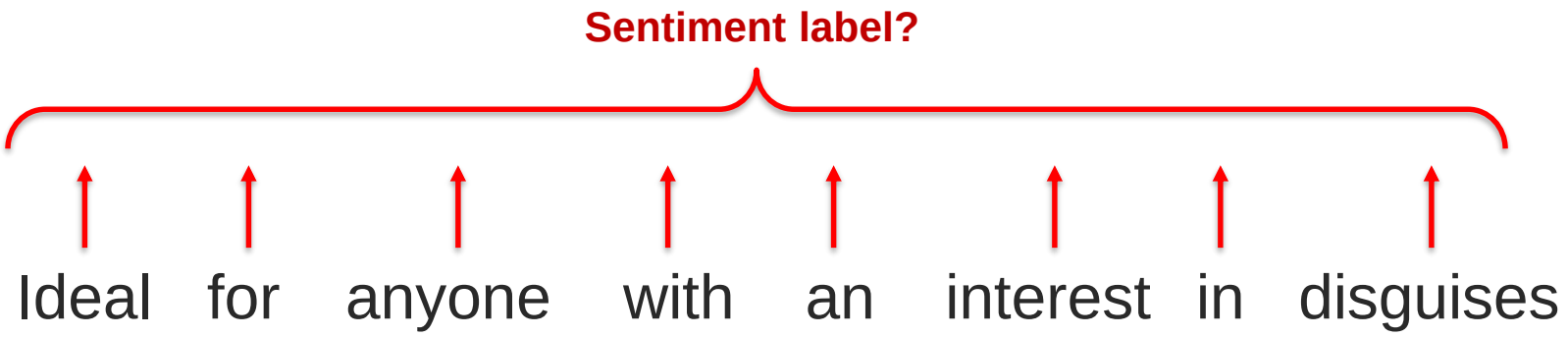


- Models long-term information

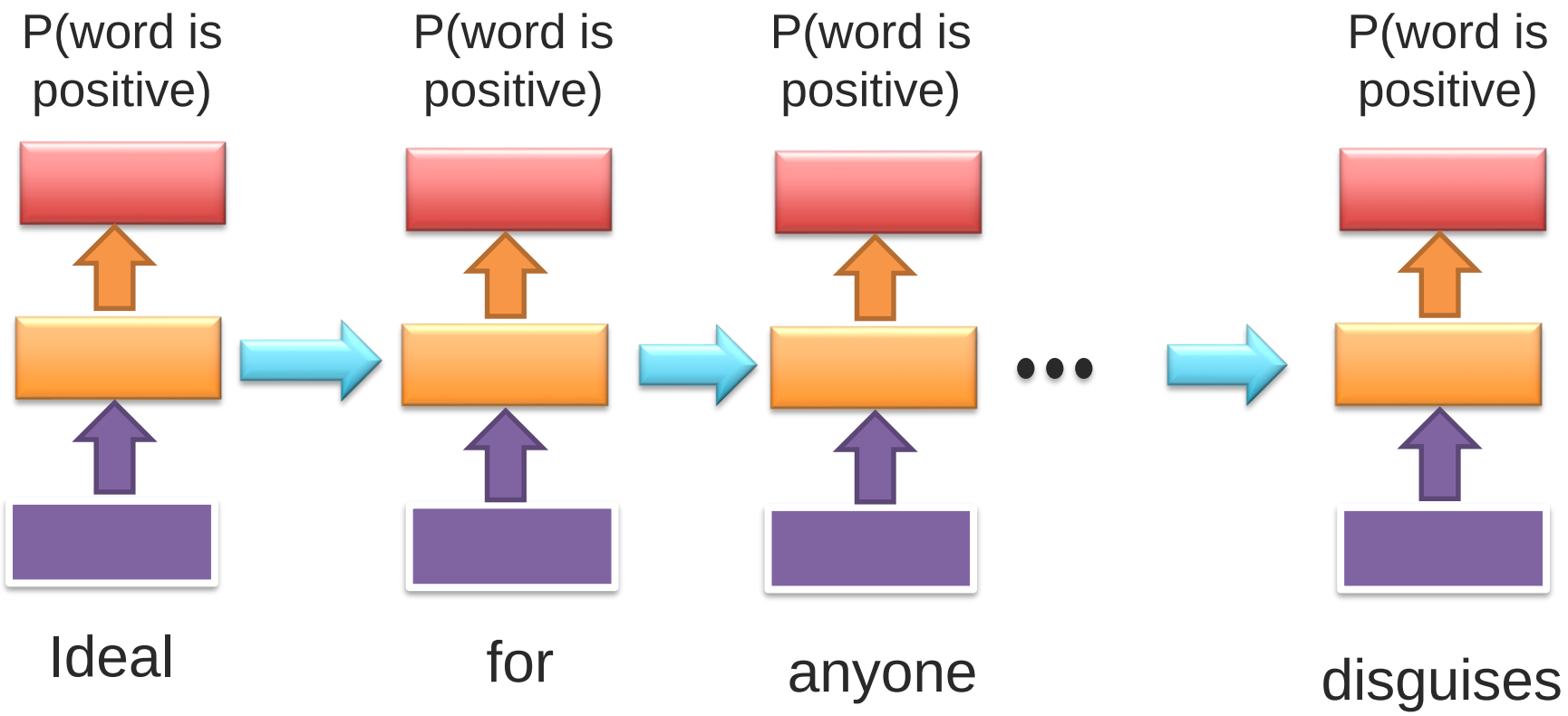
Sequence Modeling: Sequence Prediction

Prediction →

Sentiment ?
(positive or negative)

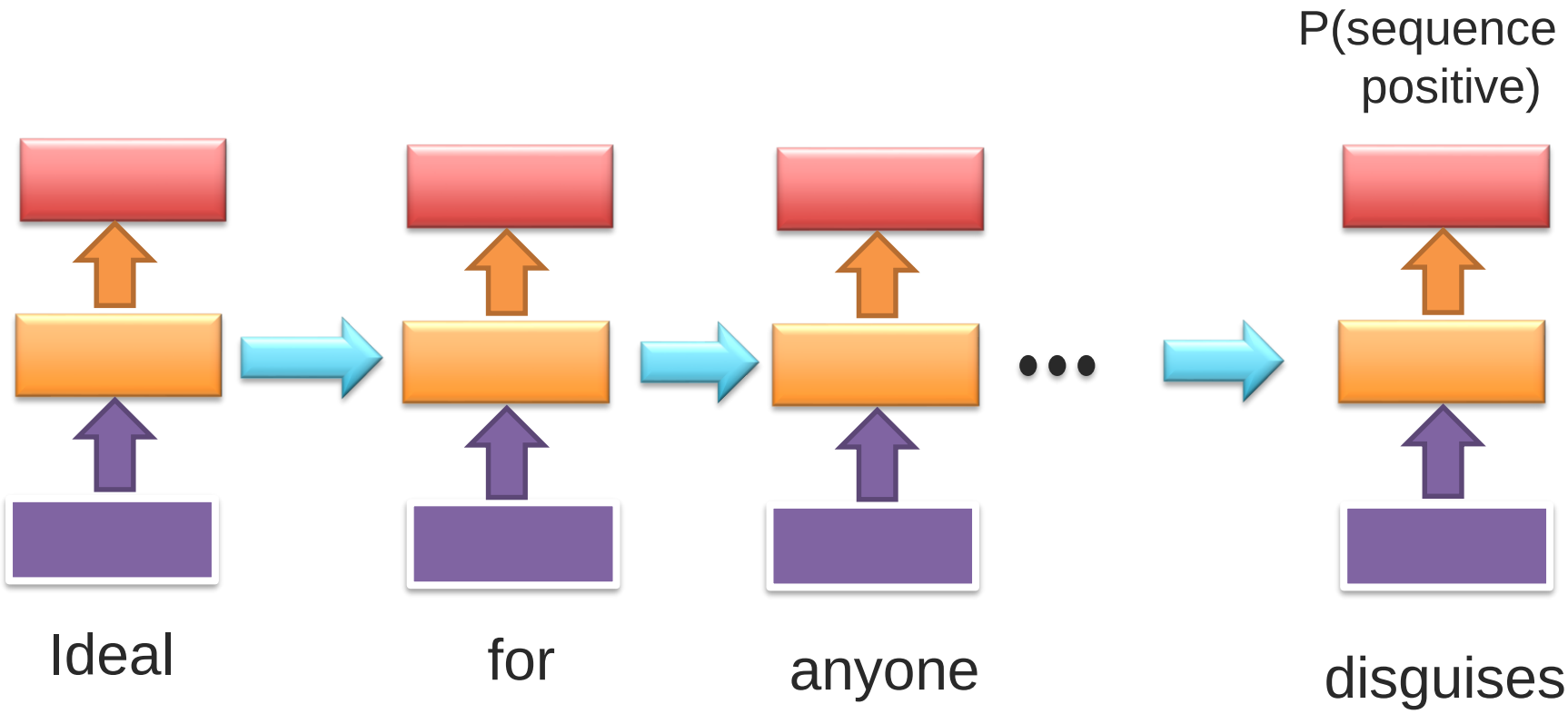


RNN for Sequence Prediction



$$L = \frac{1}{N}\sum_{t} L^{(t)} = \frac{1}{N}\sum_{t} -logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

RNN for Sequence Prediction



$$L = L^{(N)} = -logP(Y = y^{(N)}|\boldsymbol{z}^{(N)})$$

Sequence Modeling: Sequence Representation

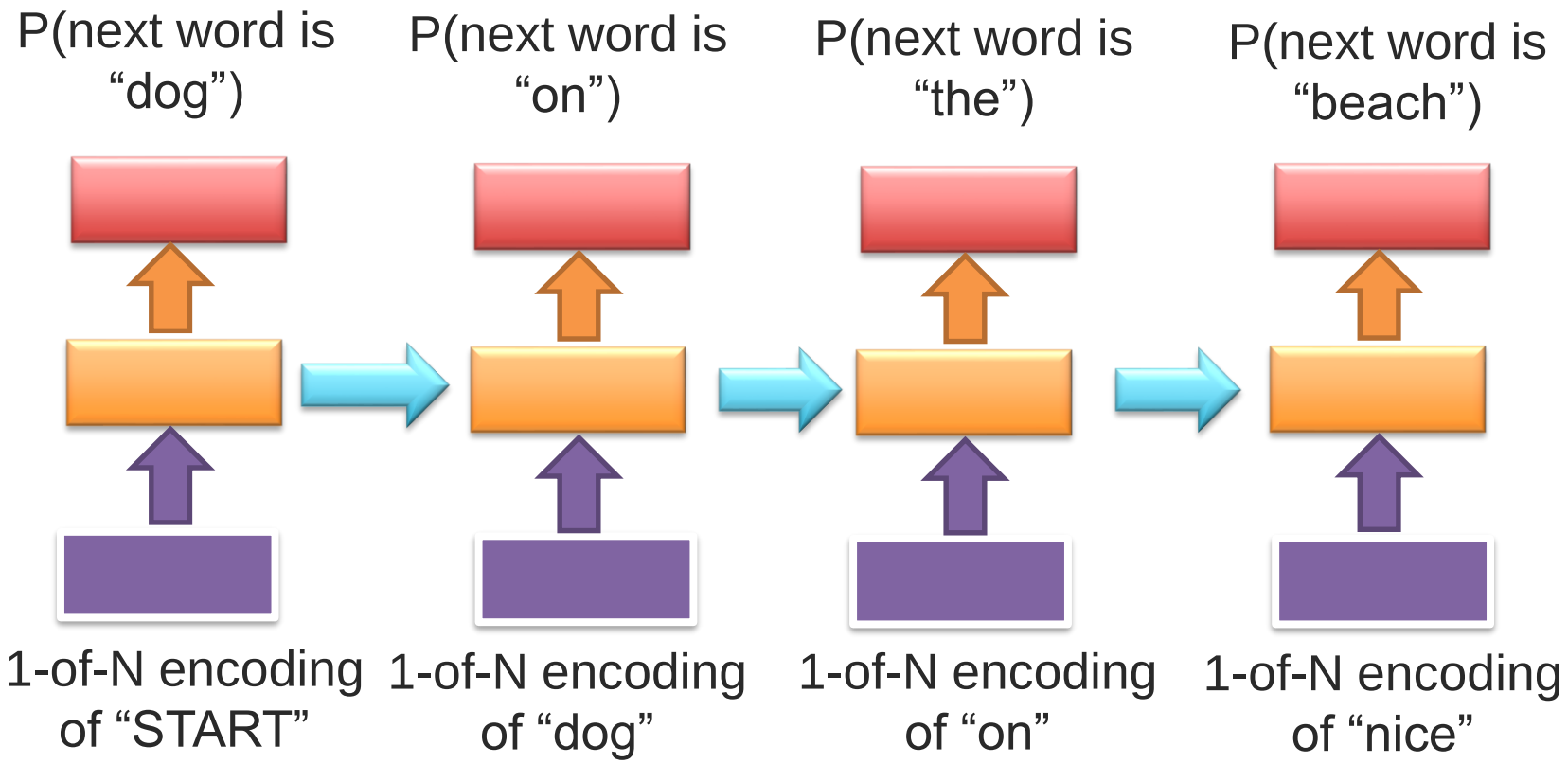
Learning →

Sequence representation

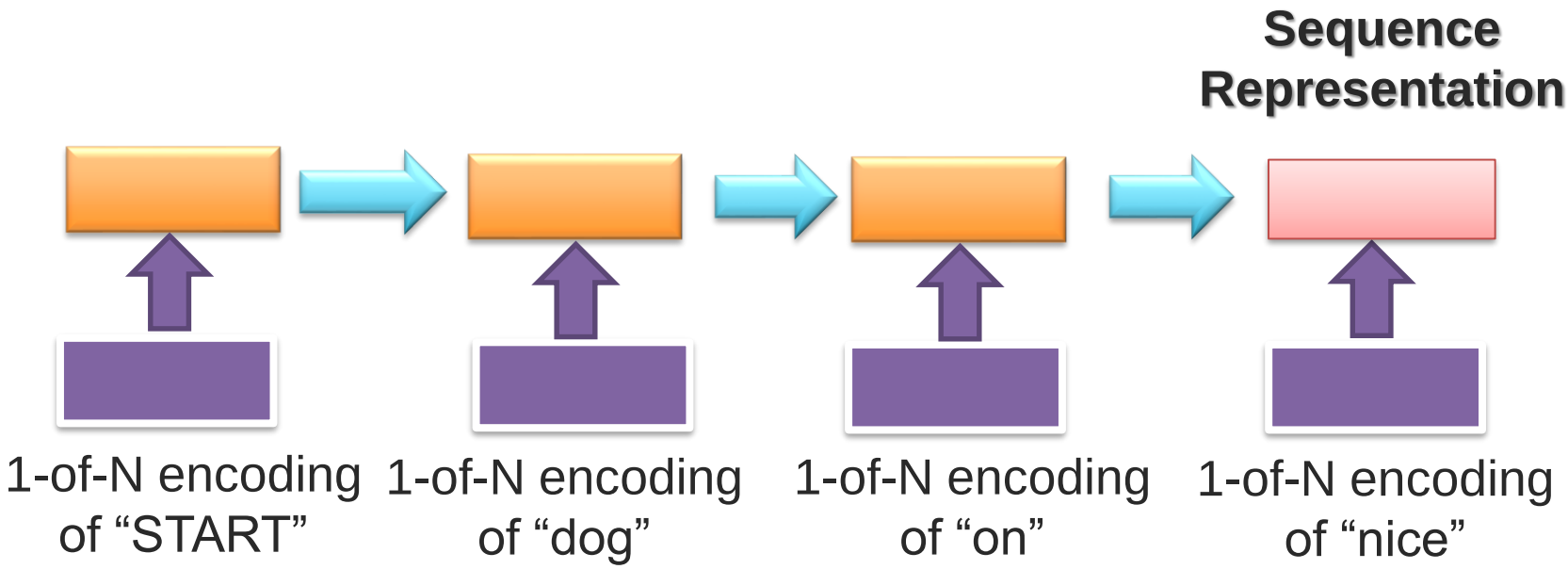
[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

RNN for Sequence Representation

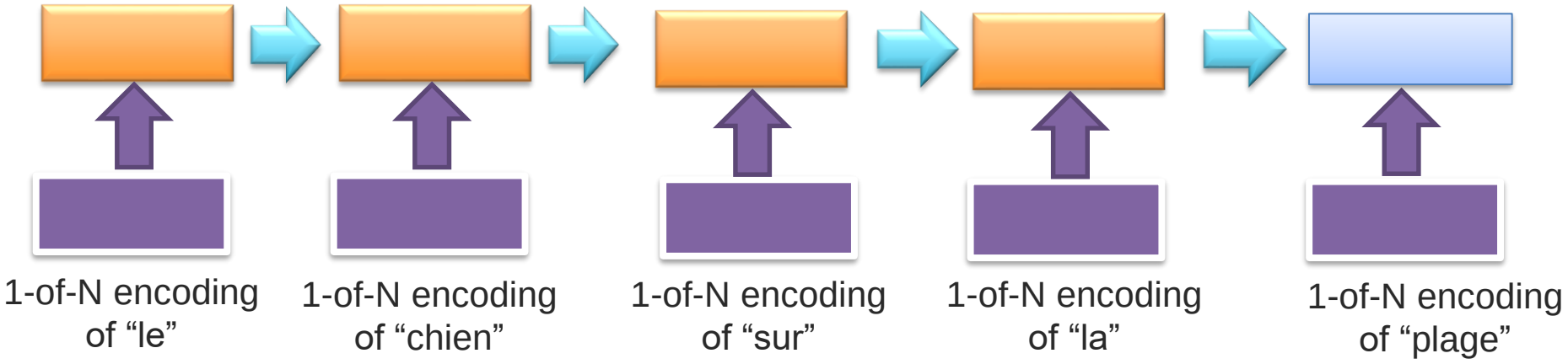


RNN for Sequence Representation (Encoder)

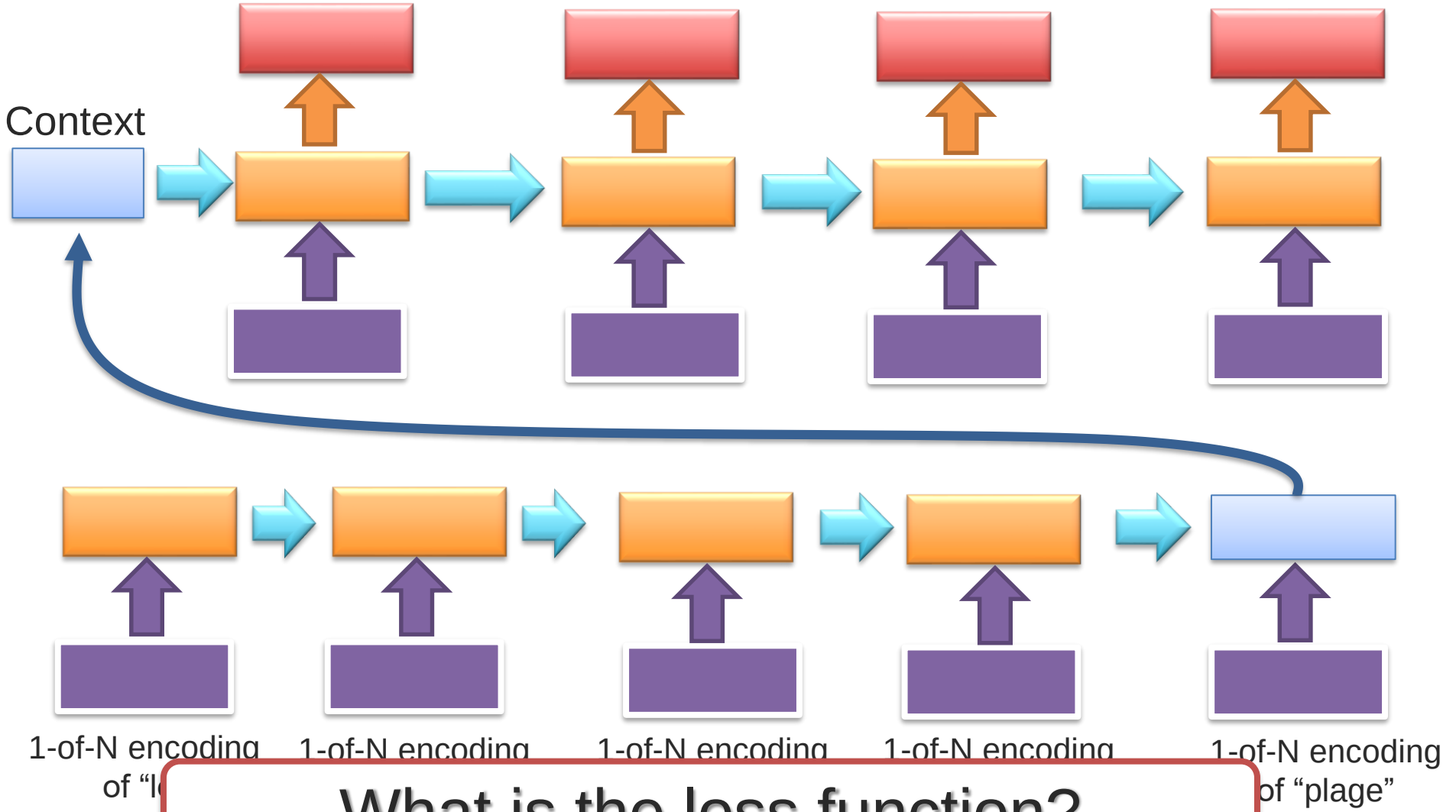


RNN-based for Machine Translation

Le chien sur la plage → The dog on the beach



Encoder-Decoder Architecture



What is the loss function?

Related Topics

- Character-level “language models”
  - Xiang Zhang, Junbo Zhao and Yann LeCun, Character-level Convolutional Networks for Text Classification, NIPS 2015

  http://arxiv.org/pdf/1509.01626v2.pdf

- Skip-though: embedding at the sentence level
  - Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler. Skip-Thought Vectors, NIPS 2015

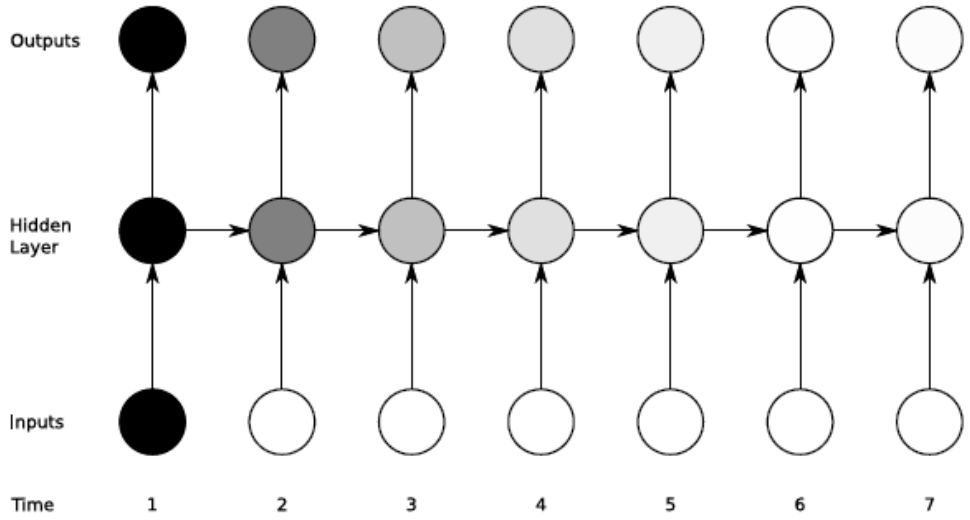
  http://arxiv.org/pdf/1506.06726v1.pdf

Gated Recurrent Neural Networks

Long-term Dependencies

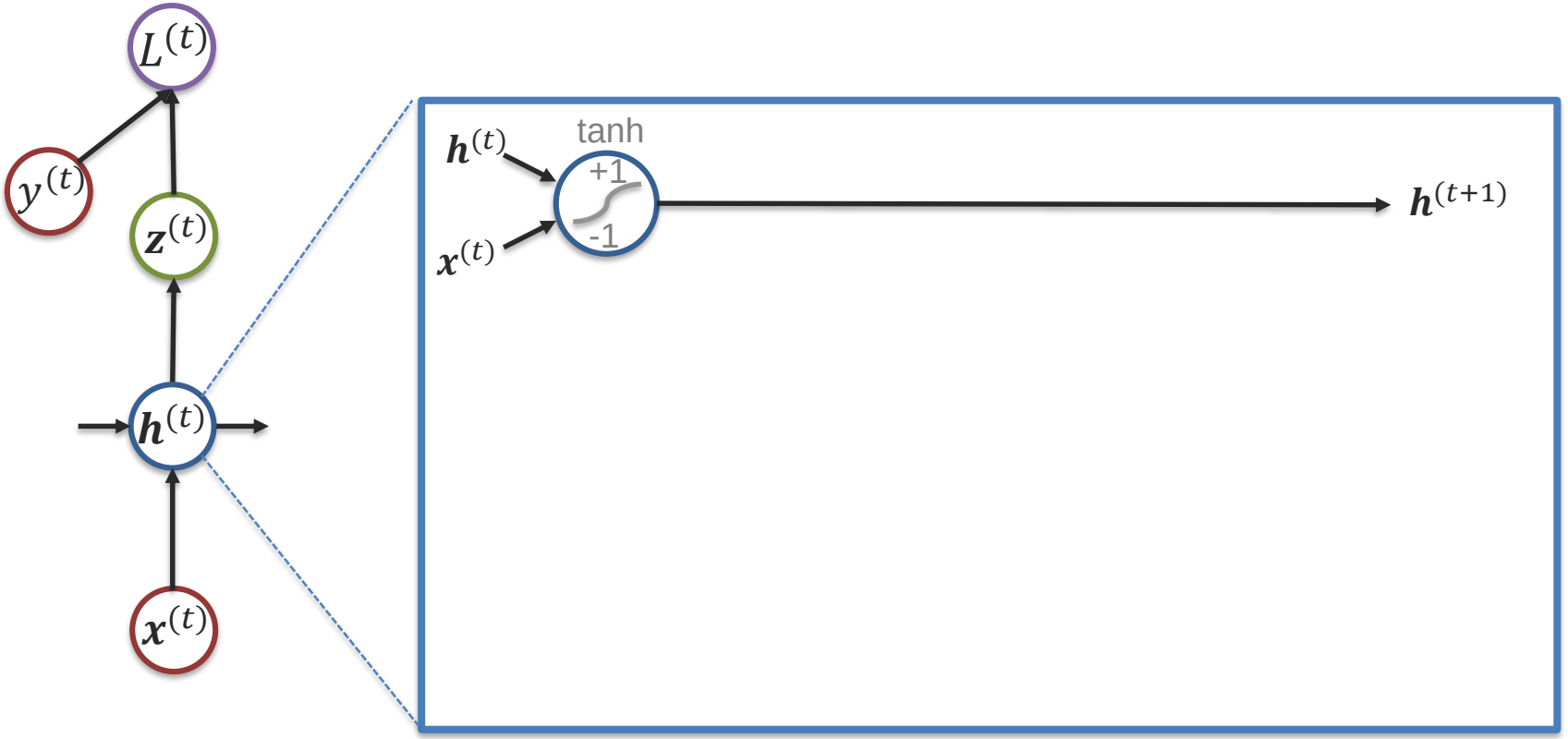
Vanishing gradient problem for RNNs:

$$\boldsymbol{h}^{(t)} \sim tanh(\boldsymbol{W}\boldsymbol{h}^{(t-1)})$$



- The influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it cycles around the network's recurrent connections.

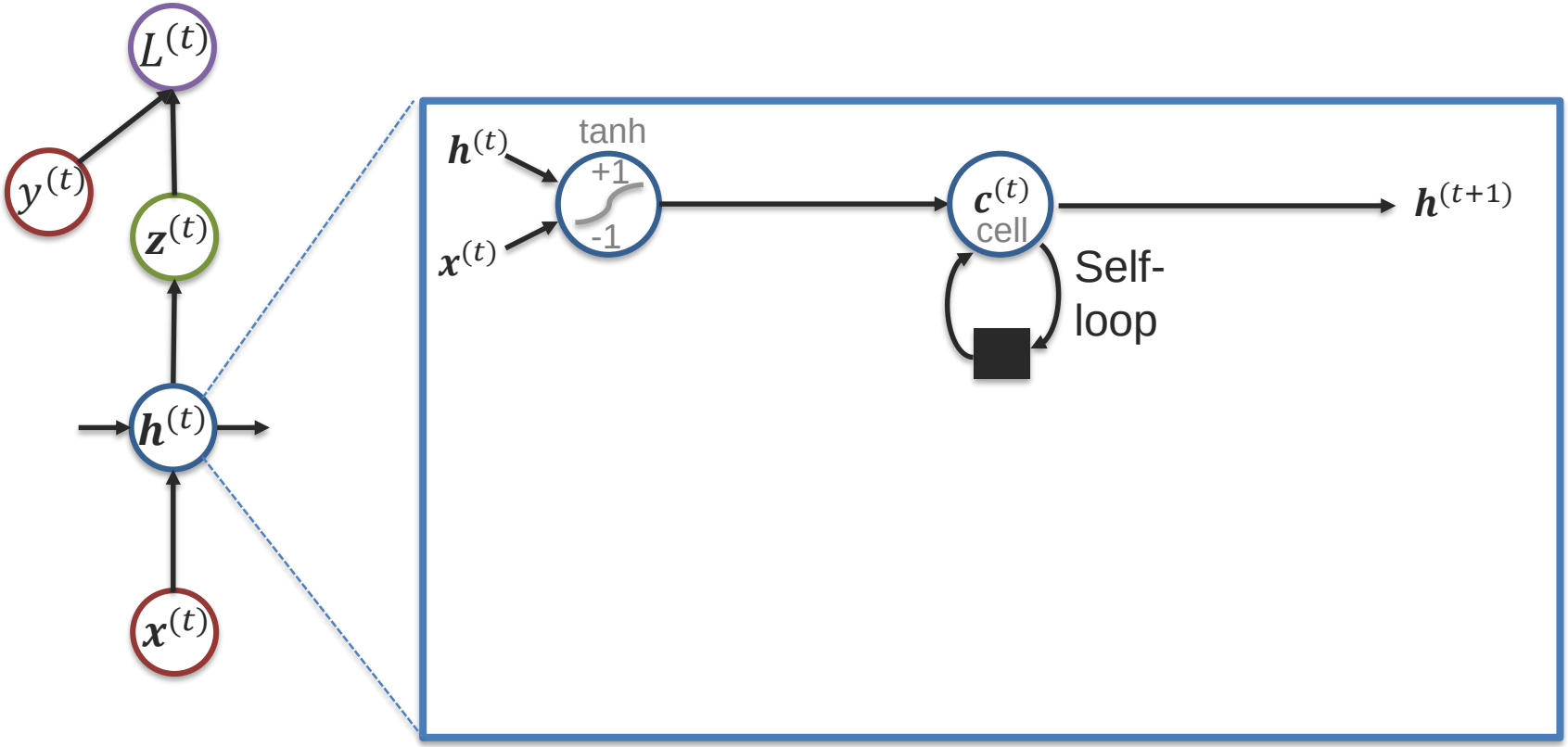
Recurrent Neural Networks



LSTM ideas: (1) "Memory" Cell and Self Loop

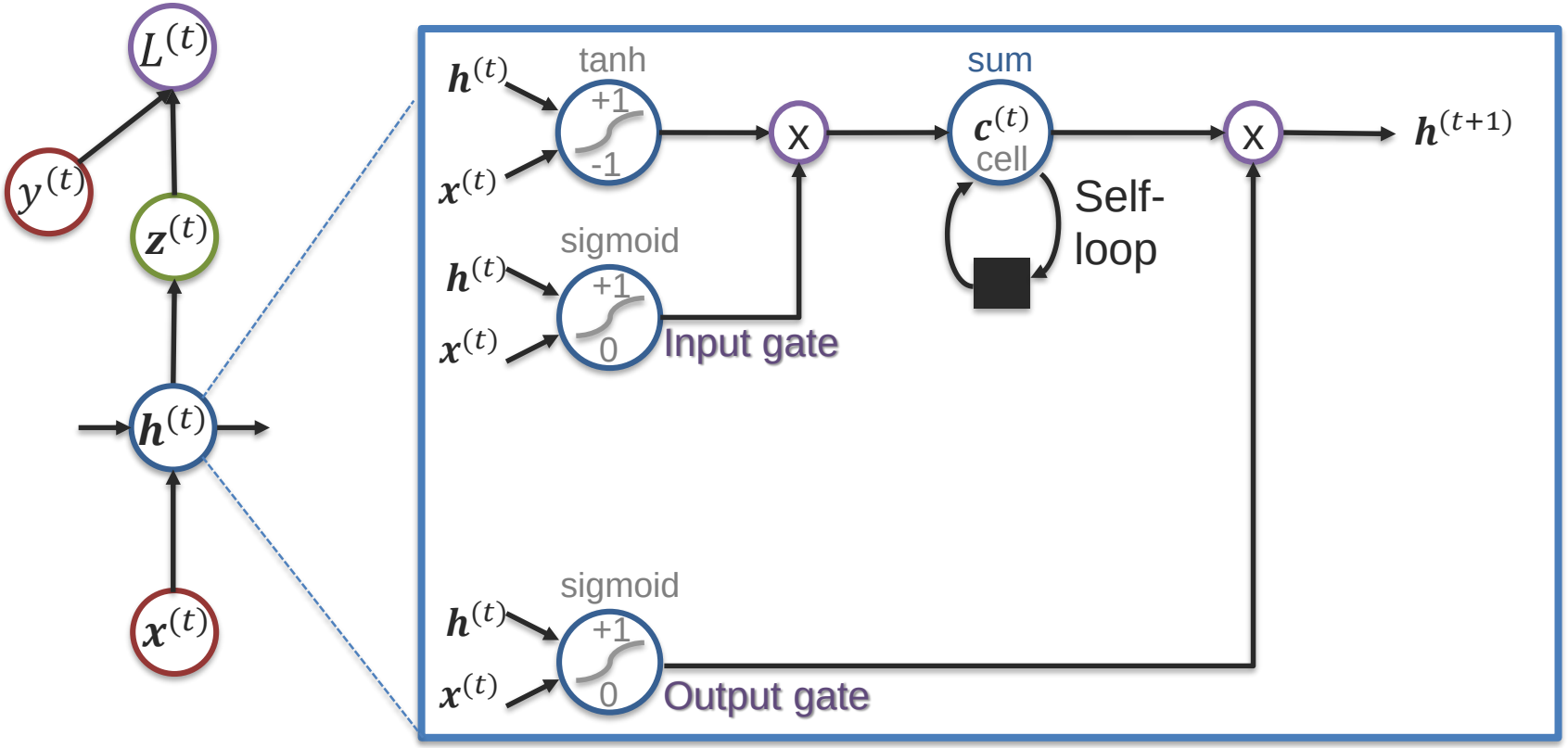
[Hochreiter and Schmidhuber, 1997]

Long Short-Term Memory (LSTM)



LSTM Ideas: (2) Input and Output Gates

[Hochreiter and Schmidhuber, 1997]

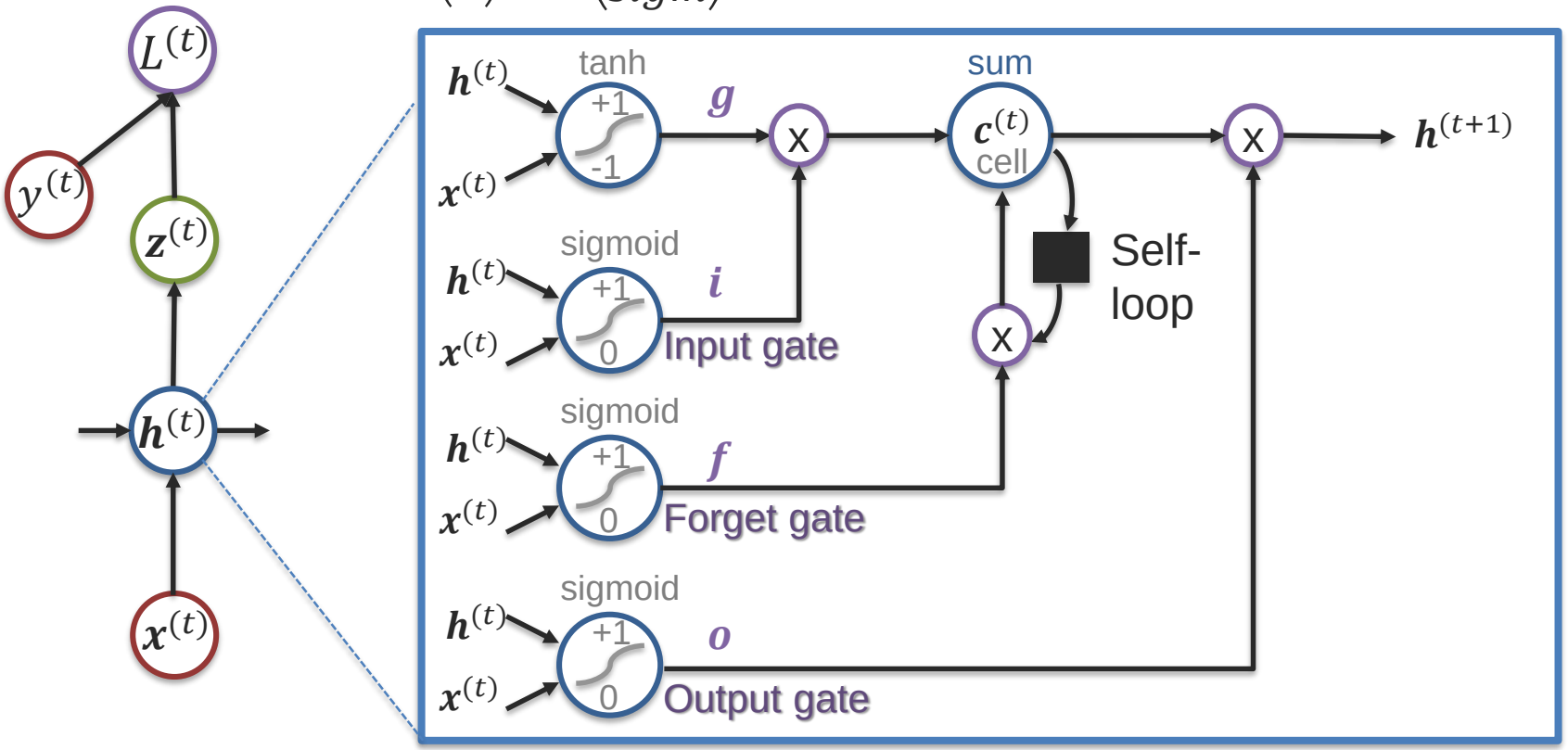


LSTM Ideas: (3) Forget Gate [Gers et al., 2000]

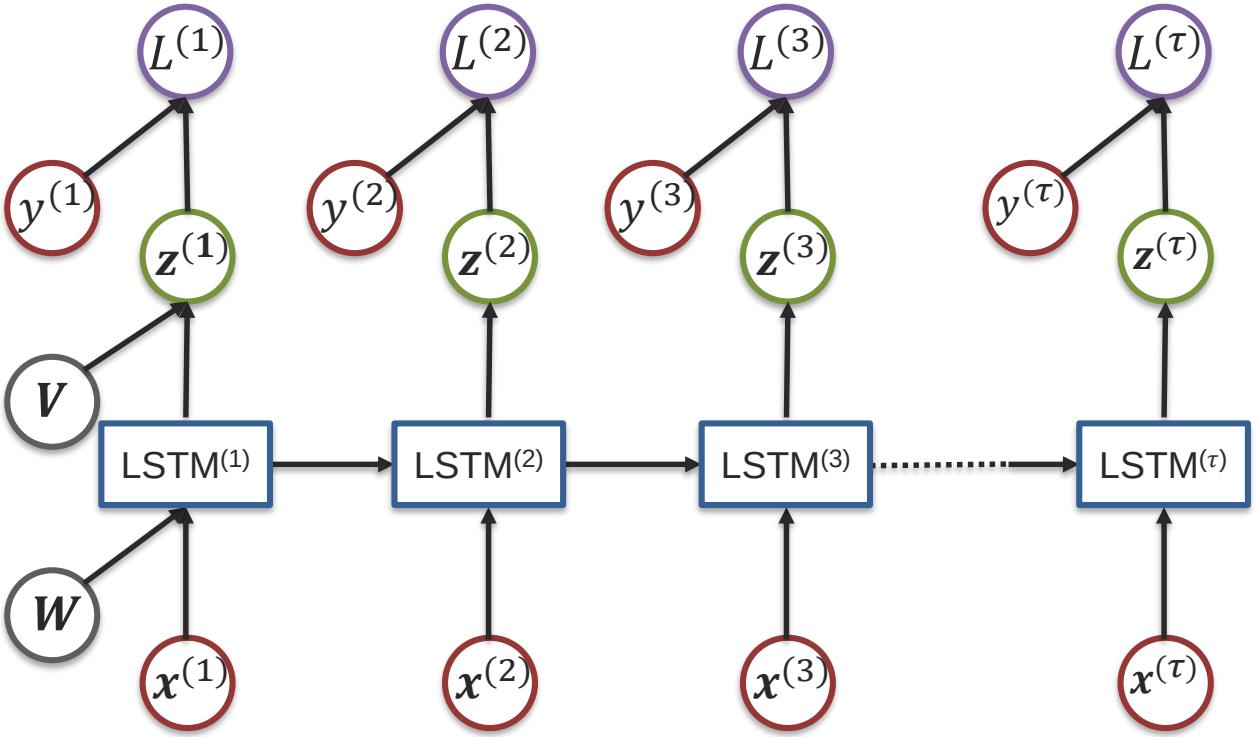
$$\begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{i} \\ \boldsymbol{f} \\ \boldsymbol{o} \end{pmatrix} = \begin{pmatrix} tanh \\ sigm \\ sigm \\ sigm \end{pmatrix} \boldsymbol{W} \begin{pmatrix} \boldsymbol{h}^{(t)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}$$

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

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

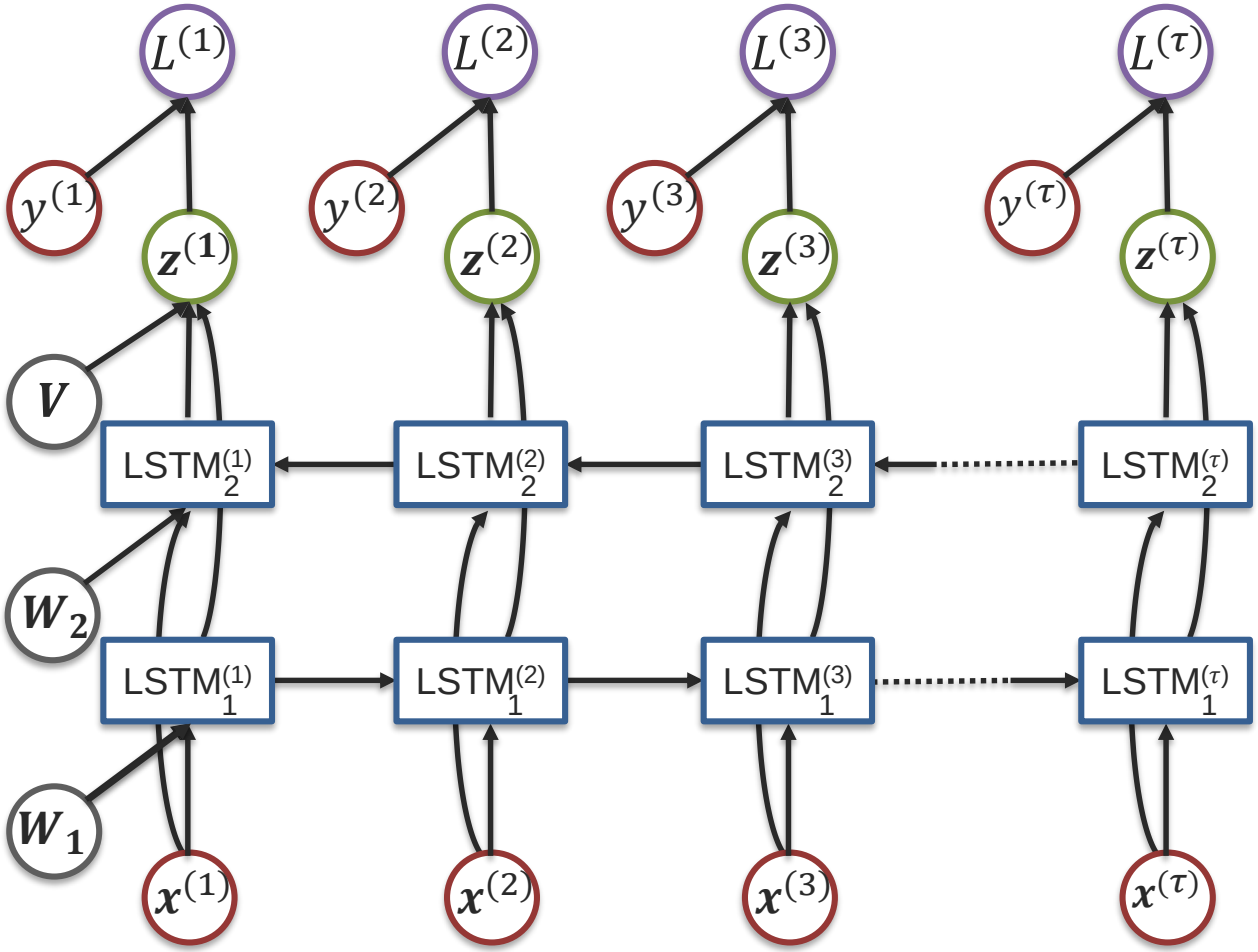


Recurrent Neural Network using LSTM Units

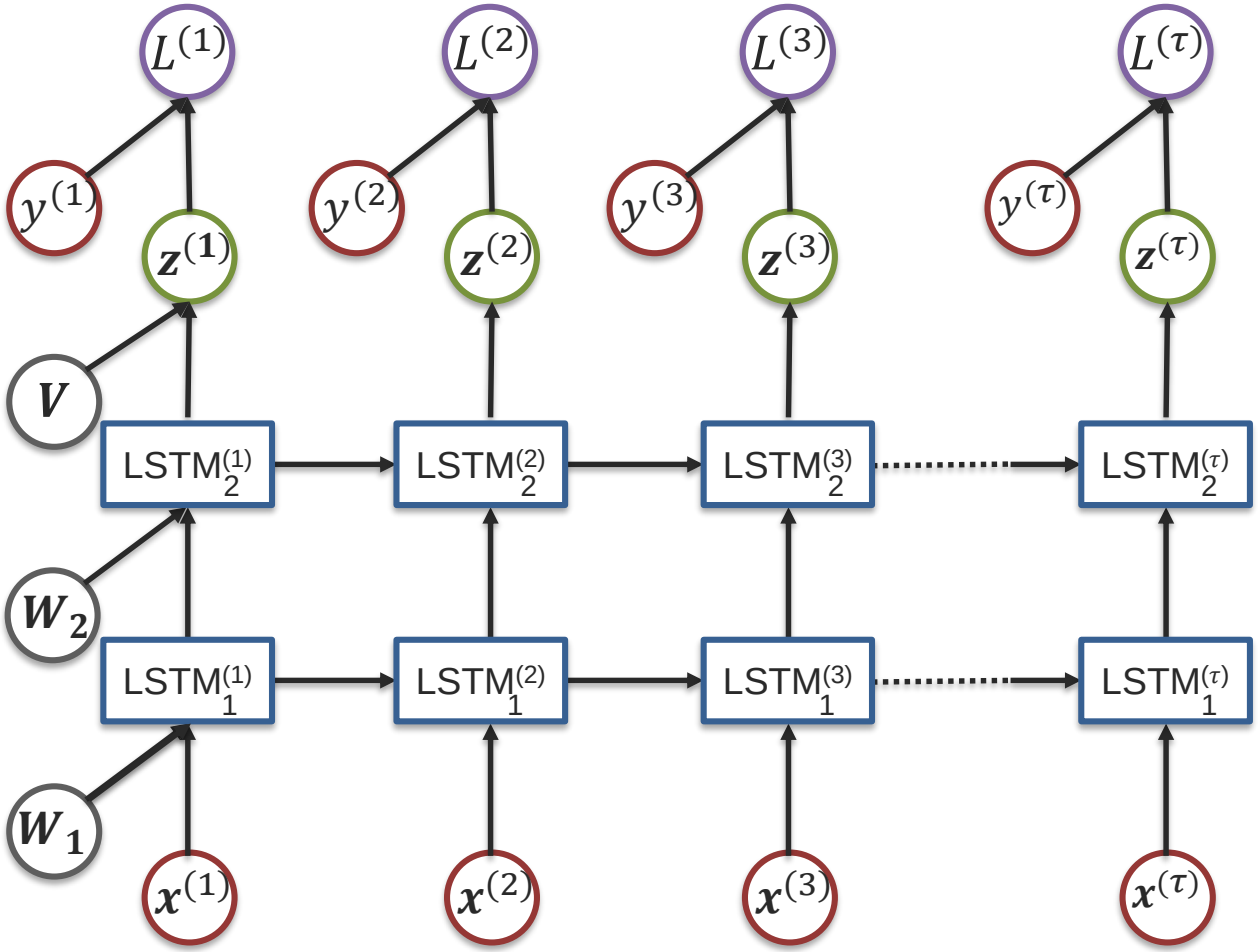


Gradient can still be computer using backpropagation!

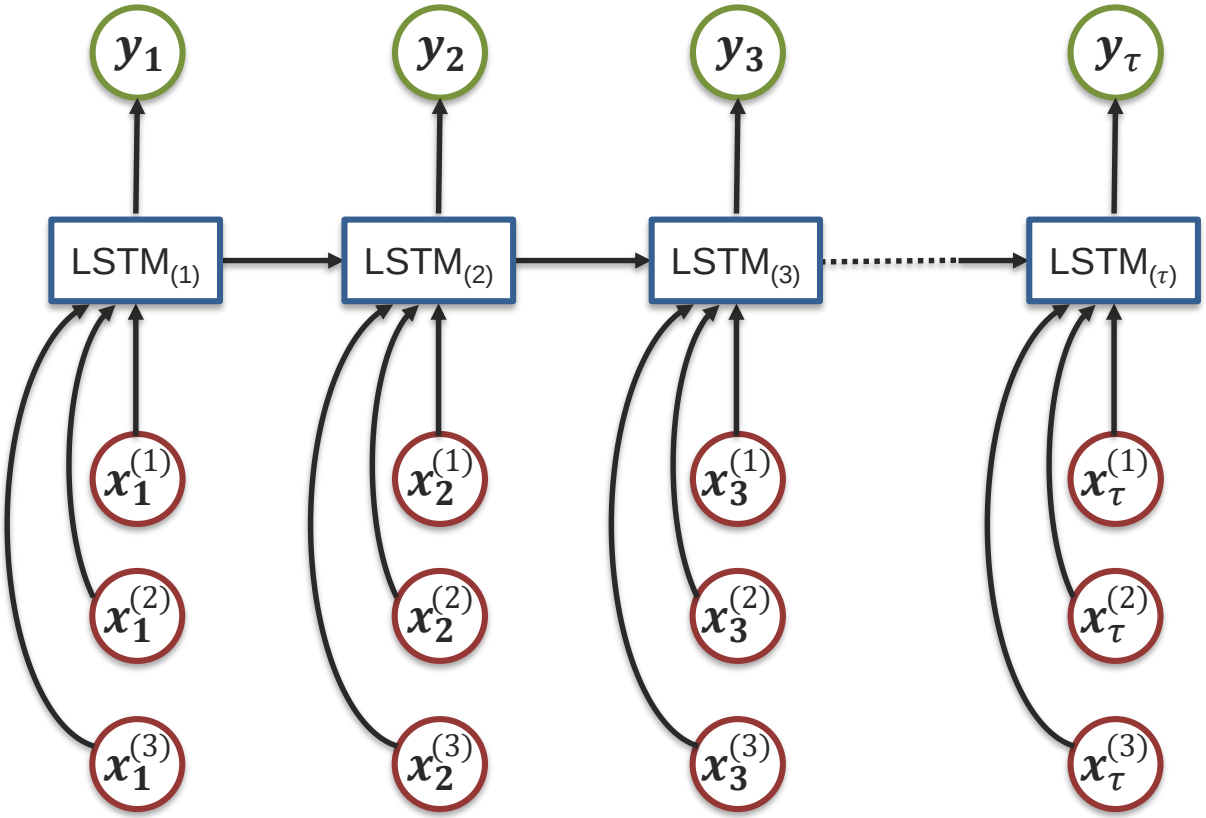
Bi-directional LSTM Network



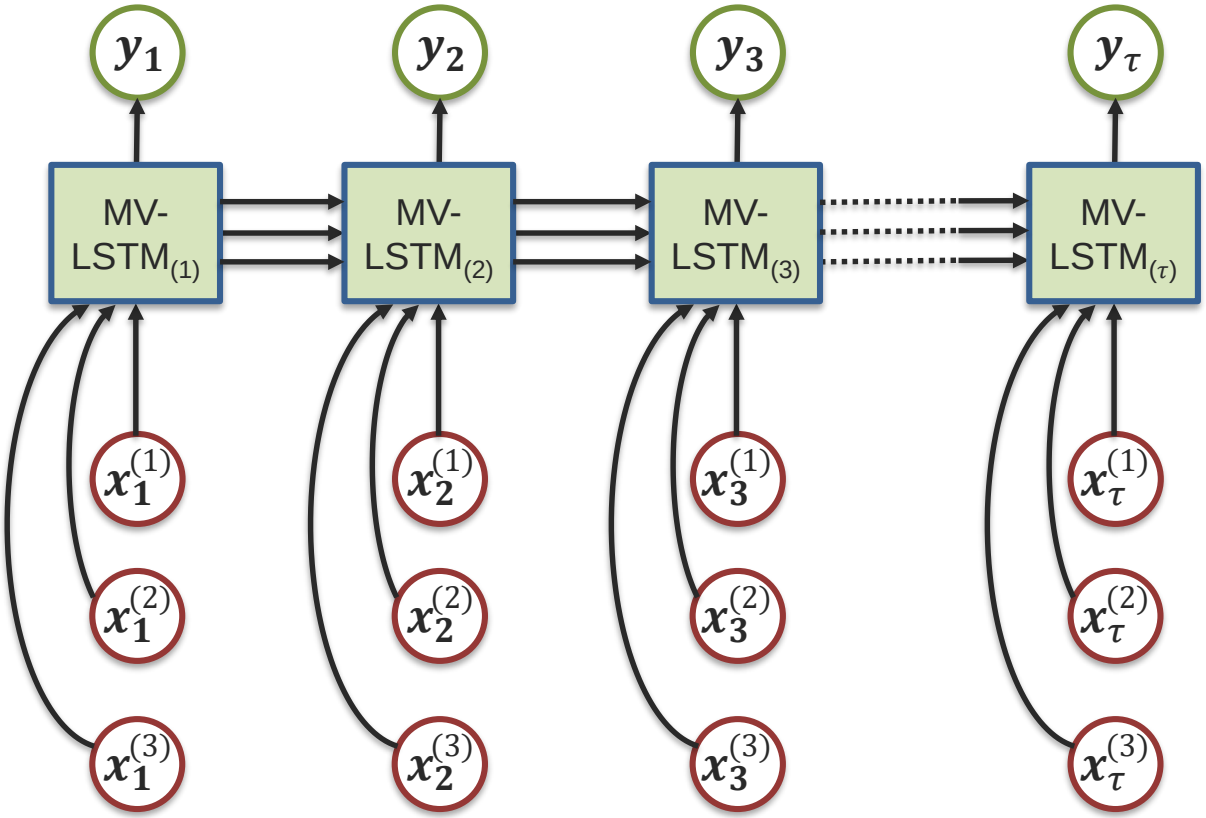
Deep LSTM Network



Multimodal Sequence Modeling – Early Fusion

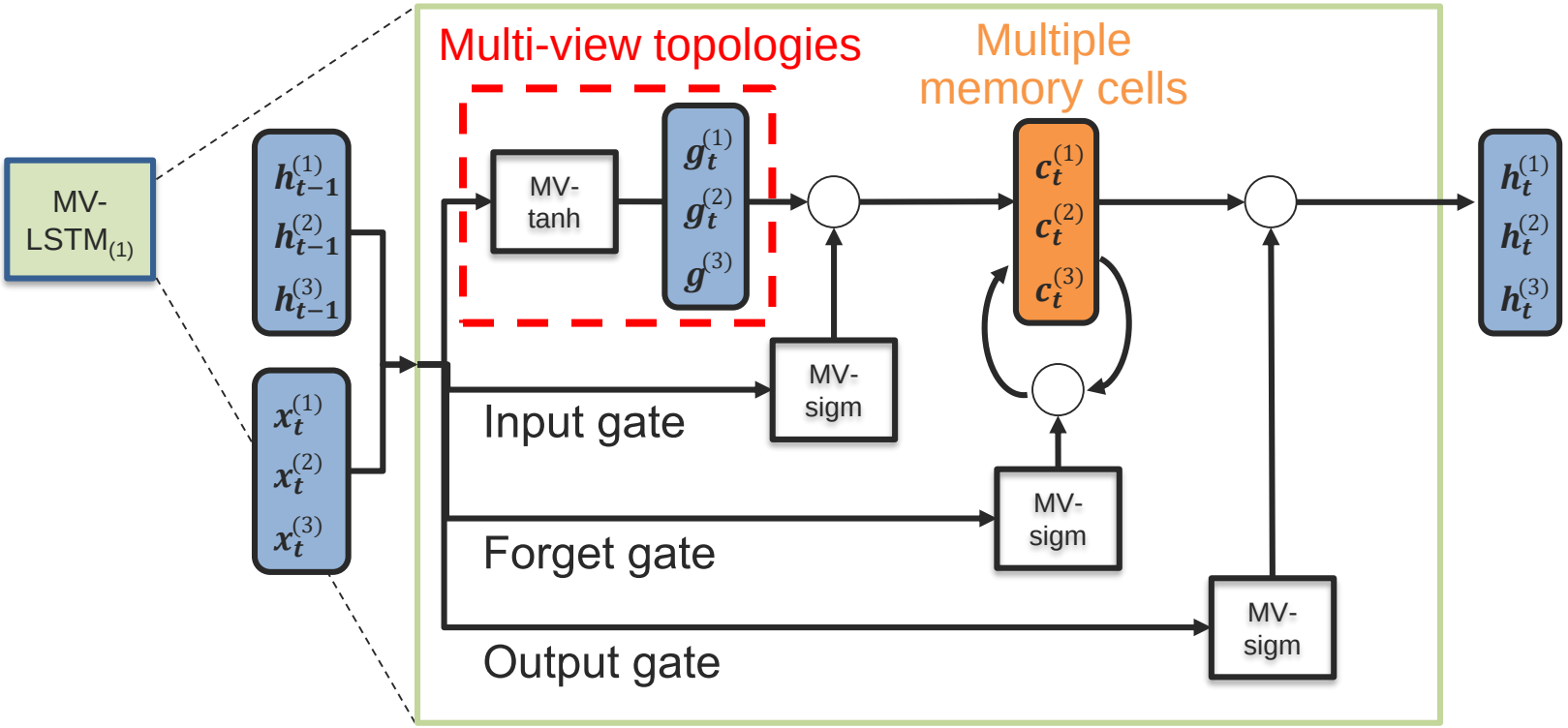


Multi-View Long Short-Term Memory (MV-LSTM)



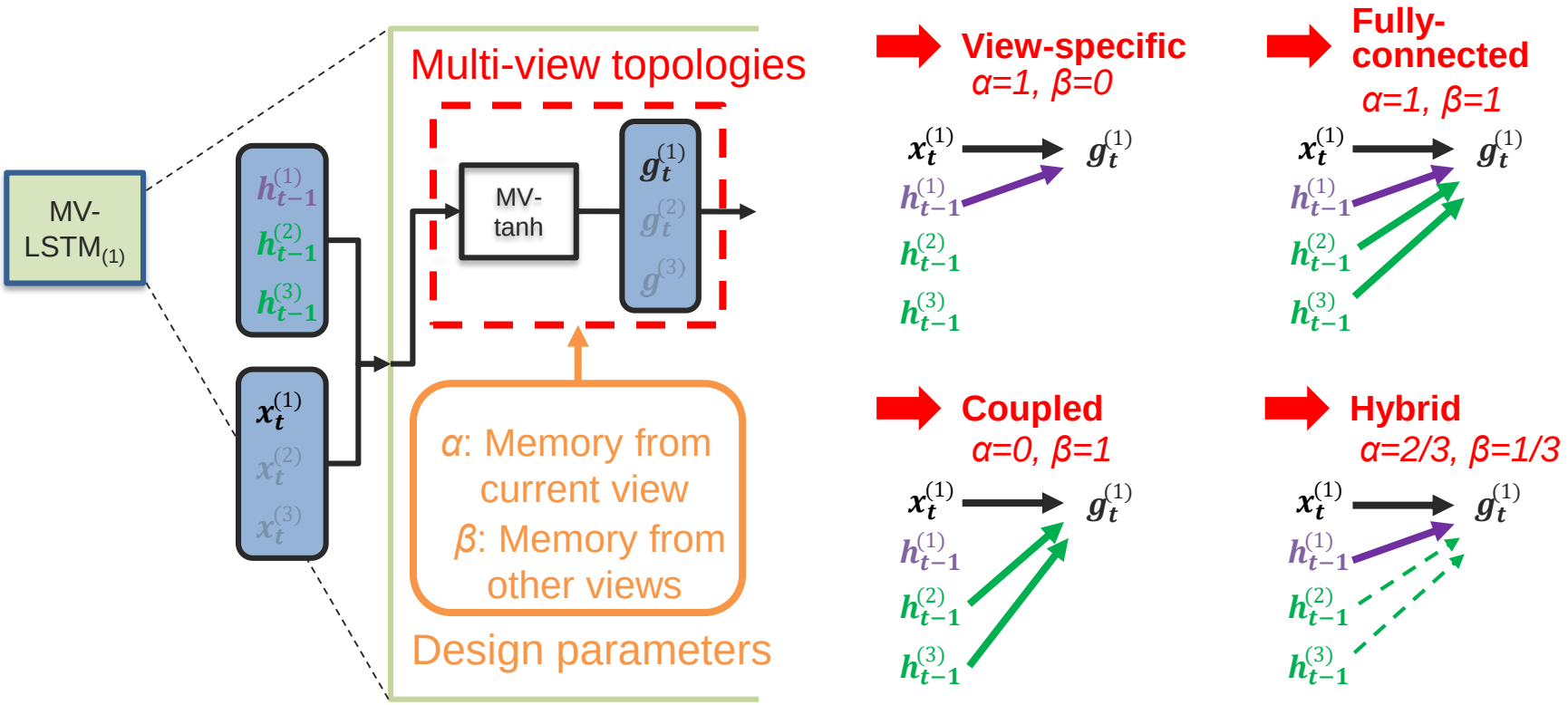
[Shyam, Morency, et al. Extending Long Short-Term Memory for Multi-View Structured Learning, **ECCV**, 2016]

Multi-View Long Short-Term Memory



[Shyam, Morency, et al. Extending Long Short-Term Memory for Multi-View Structured Learning, **ECCV**, 2016]

Topologies for Multi-View LSTM



[Shyam, Morency, et al. Extending Long Short-Term Memory for Multi-View Structured Learning, **ECCV**, 2016]

Multi-View Long Short-Term Memory (MV-LSTM)

Multimodal prediction of children engagement

Class labels	Model	Precision	Recall	F1
Easy to engage	LSTM (Early fusion)	0.75	0.81	0.78
	MV-LSTM Full	0.81	0.81	0.81
	MV-LSTM Coupled	0.79	0.81	0.80
	MV-LSTM Hybrid	**0.80**	**0.86**	**0.83**
Difficult to engage	LSTM (Early fusion)	0.63	0.55	0.59
	MV-LSTM Full	0.68	0.68	0.68
	MV-LSTM Coupled	0.67	0.64	0.65
	MV-LSTM Hybrid	**0.74**	**0.64**	**0.68**

[Shyam, Morency, et al. Extending Long Short-Term Memory for Multi-View Structured Learning, **ECCV**, 2016]

Backpropagation Through Time

Optimization: Gradient Computation

Vector representation:

$$\nabla_{\boldsymbol{x}} y = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \frac{\partial y}{\partial x_3}\right]$$

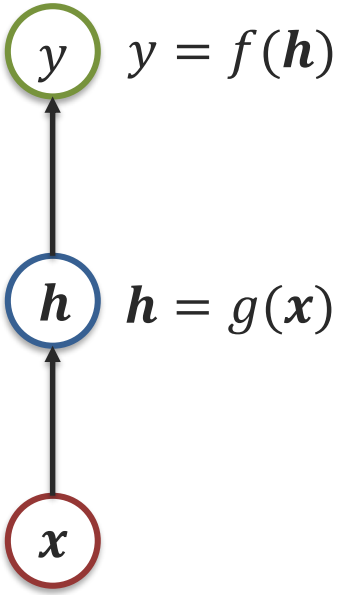
Gradient

$$\nabla_{\boldsymbol{x}} y = \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T \nabla_{\boldsymbol{h}} y$$

"backprop" Gradient

"local" Jacobian

(matrix of size $|h| \times |x|$ computed using partial derivatives)



Backpropagation Algorithm

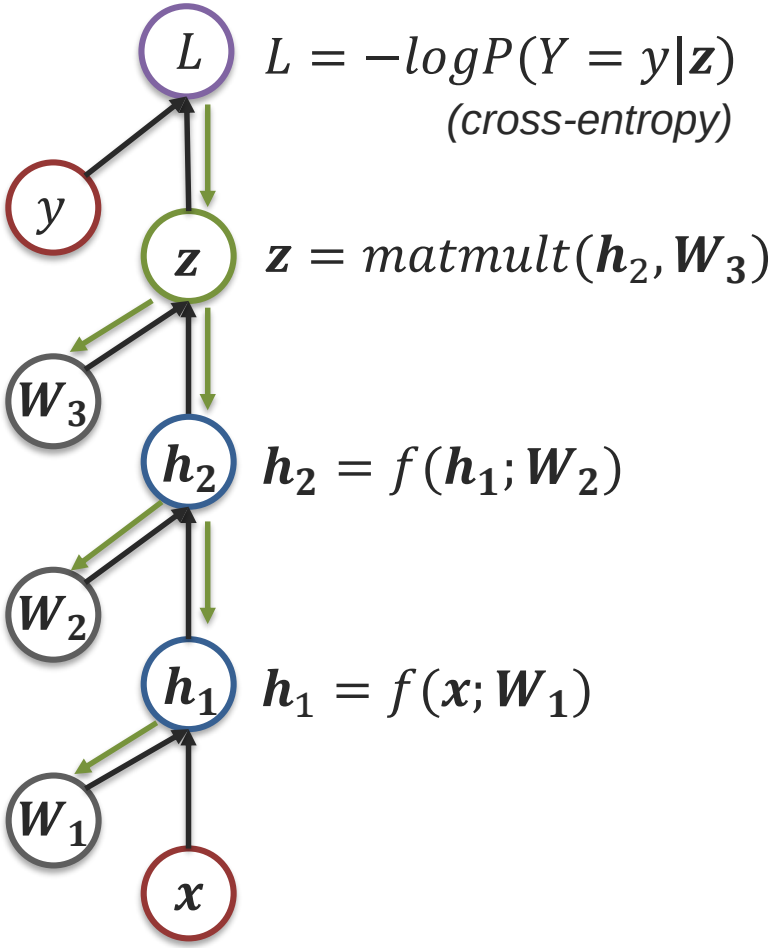
Forward pass

- Following the graph topology, compute value of each unit

Backpropagation pass

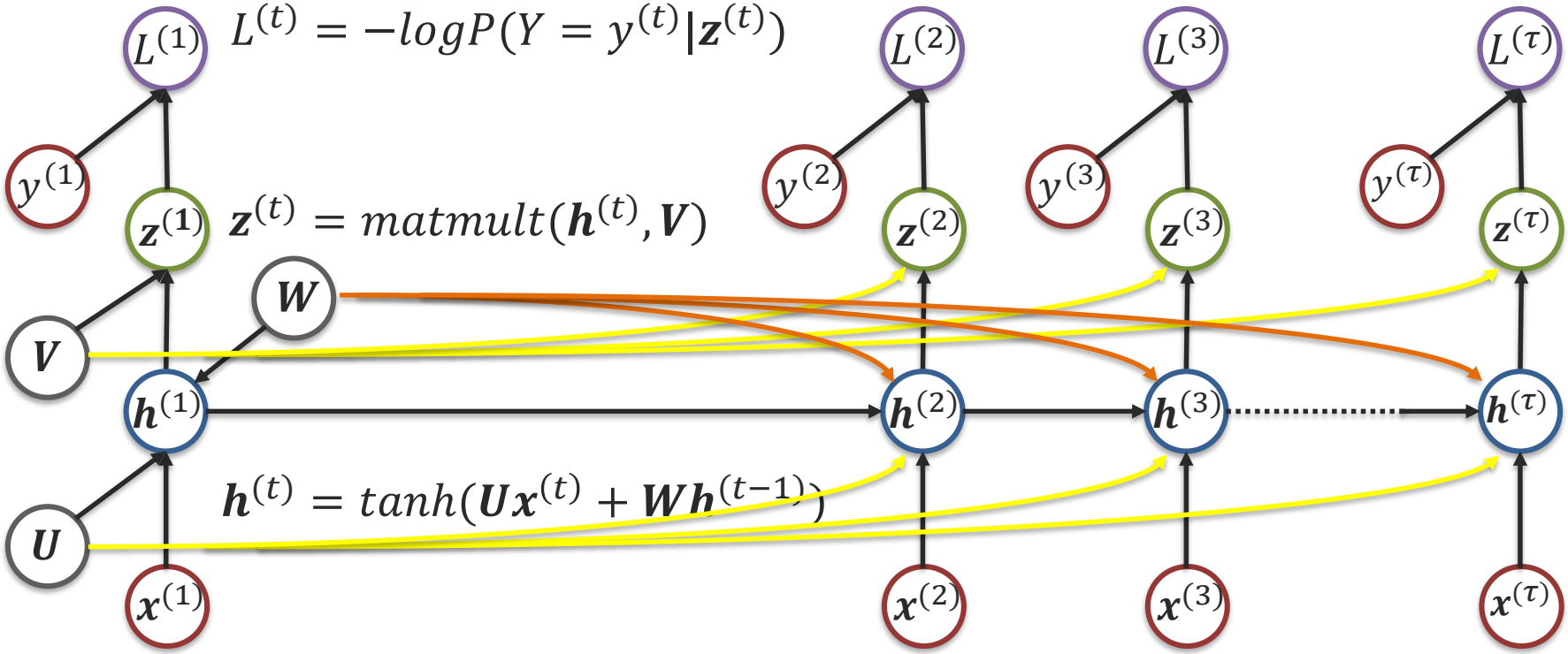
- Initialize output gradient = 1
- Compute "local" Jacobian matrix using values from forward pass
- Use the chain rule:

Gradient = "local" Jacobian x "backprop" gradient



Recurrent Neural Networks

$$L = \sum_t L^{(t)}$$



Backpropagation Through Time

$$L = \sum_t L^{(t)} = -\sum_t logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

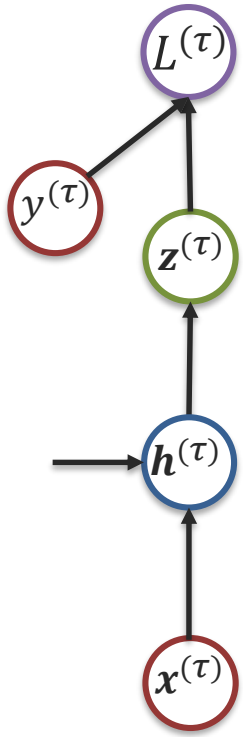
Gradient = "backprop" gradient x "local" Jacobian

$L^{(\tau)}$ or $L^{(t)}$ $\quad \frac{\partial L}{\partial L^{(t)}} = 1$

$\boldsymbol{z}^{(\tau)}$ or $\boldsymbol{z}^{(t)}$ $\quad (\nabla_{\boldsymbol{z}^{(t)}} L)_i = \frac{\partial L}{\partial z_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial z_i^{(t)}} = sigmoid(z_i^t) - \mathbf{1}_{i,y^{(t)}}$

$\boldsymbol{h}^{(\tau)}$ $\quad \nabla_{\boldsymbol{h}^{(\tau)}} L = \nabla_{\boldsymbol{z}^{(\tau)}} L \frac{\partial \boldsymbol{z}^{(\tau)}}{\partial \boldsymbol{h}^{(\tau)}} = \nabla_{\boldsymbol{z}^{(\tau)}} L \boldsymbol{V}$

$\boldsymbol{h}^{(t)} \rightarrow \boldsymbol{h}^{(t+1)}$ $\quad \nabla_{\boldsymbol{h}^{(t)}} L = \nabla_{\boldsymbol{z}^{(t)}} L \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} + \nabla_{\boldsymbol{z}^{(t+1)}} L \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$



Backpropagation Through Time

$$L = \sum_t L^{(t)} = -\sum_t logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

Gradient = "backprop" gradient x "local" Jacobian

$$\boldsymbol{V} \quad \nabla_{\boldsymbol{V}} L = \sum_t \left(\nabla_{\boldsymbol{z}^{(t)}} L\right) \frac{\partial \boldsymbol{z}^{(t)}}{\partial \boldsymbol{V}}$$

$$\boldsymbol{W} \quad \nabla_{\boldsymbol{W}} L = \sum_t \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{W}}$$

$$\boldsymbol{U} \quad \nabla_{\boldsymbol{U}} L = \sum_t \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{U}}$$