NLP

Deep Learning

741.

Recurrent Neural Networks

(RNN)

Sequence representations

- Language is represented as sequences
  - Discourse is made of sentences
  - Sentences are made of words
  - Words are made of characters
- Language modeling
  - Too many parameters
  - HMM: ignore earlier history
- Long-distance dependencies
- Example 1
  - The girl ate the apple because she was hungry.
  - The boy ate the apple because he was hungry.
- Example 2
  - That night was fairly lonely.
  - That knight was fairly lonely.

http://www.singularis.ltd.uk/bifroest/misc/homophones-list.htm

Winograd Schema

- Classic example
  - The **city councilmen** refused **the demonstrators** a permit because **they [feared/advocated]** violence.

    Who refused the permit?
  - **The city councilmen** refused the demonstrators a permit because **they feared** violence.
  - The city councilmen refused **the demonstrators** a permit because **they advocated** violence.
- One more example
  - The trophy would not fit in the brown suitcase because it was too big (*small*).

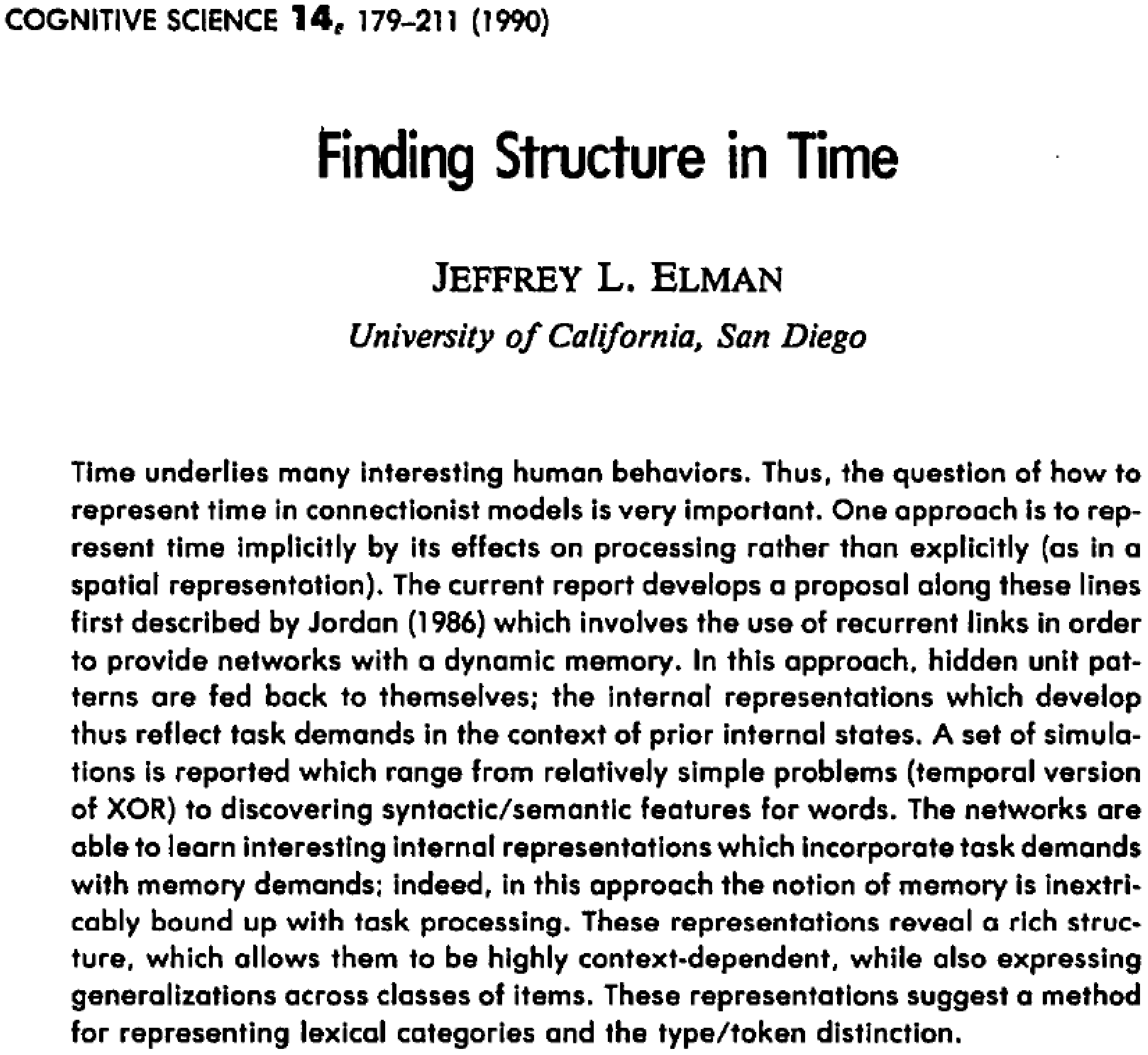
    What was too big (small)?

http://commonsensereasoning.org/winograd.html

Recurrent representations

- How can we represent entire sentences of variable length?
- Can we do this using a fixed size vector?
- Answer
  - Recurrent Neural Networks (RNN) [Elman 1990]

Recurrent Neural Networks



COGNITIVE SCIENCE **14**, 179–211 (1990)

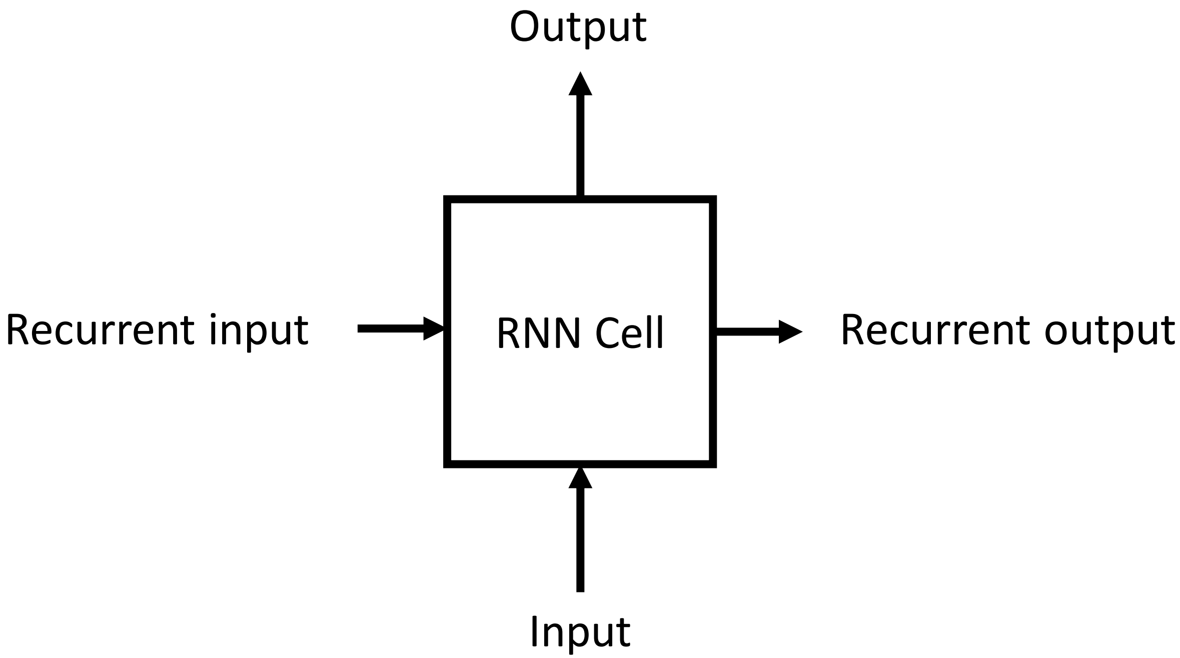
Finding Structure in Time

JEFFREY L. ELMAN

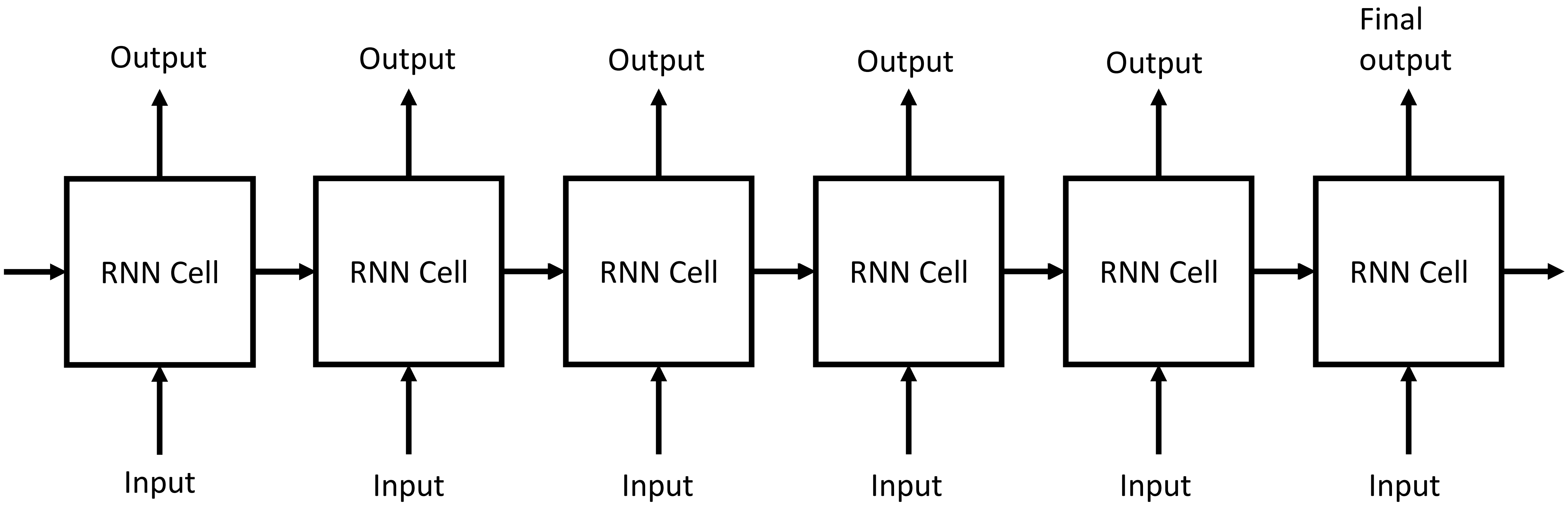
University of California, San Diego

Time underlies many interesting human behaviors. Thus, the question of how to represent time in connectionist models is very important. One approach is to represent time implicitly by its effects on processing rather than explicitly (as in a spatial representation). The current report develops a proposal along these lines first described by Jordan (1986) which involves the use of recurrent links in order to provide networks with a dynamic memory. In this approach, hidden unit patterns are fed back to themselves; the internal representations which develop thus reflect task demands in the context of prior internal states. A set of simulations is reported which range from relatively simple problems (temporal version of XOR) to discovering syntactic/semantic features for words. The networks are able to learn interesting internal representations which incorporate task demands with memory demands; indeed, in this approach the notion of memory is inextricably bound up with task processing. These representations reveal a rich structure, which allows them to be highly context-dependent, while also expressing generalizations across classes of items. These representations suggest a method for representing lexical categories and the type/token distinction.

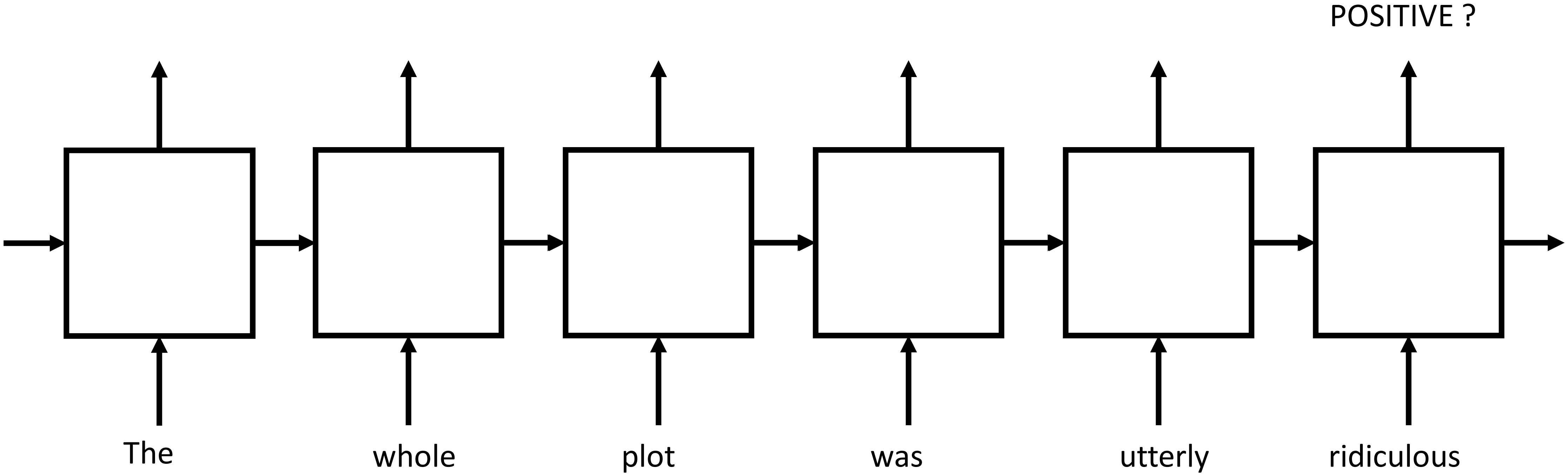
RNN Cell

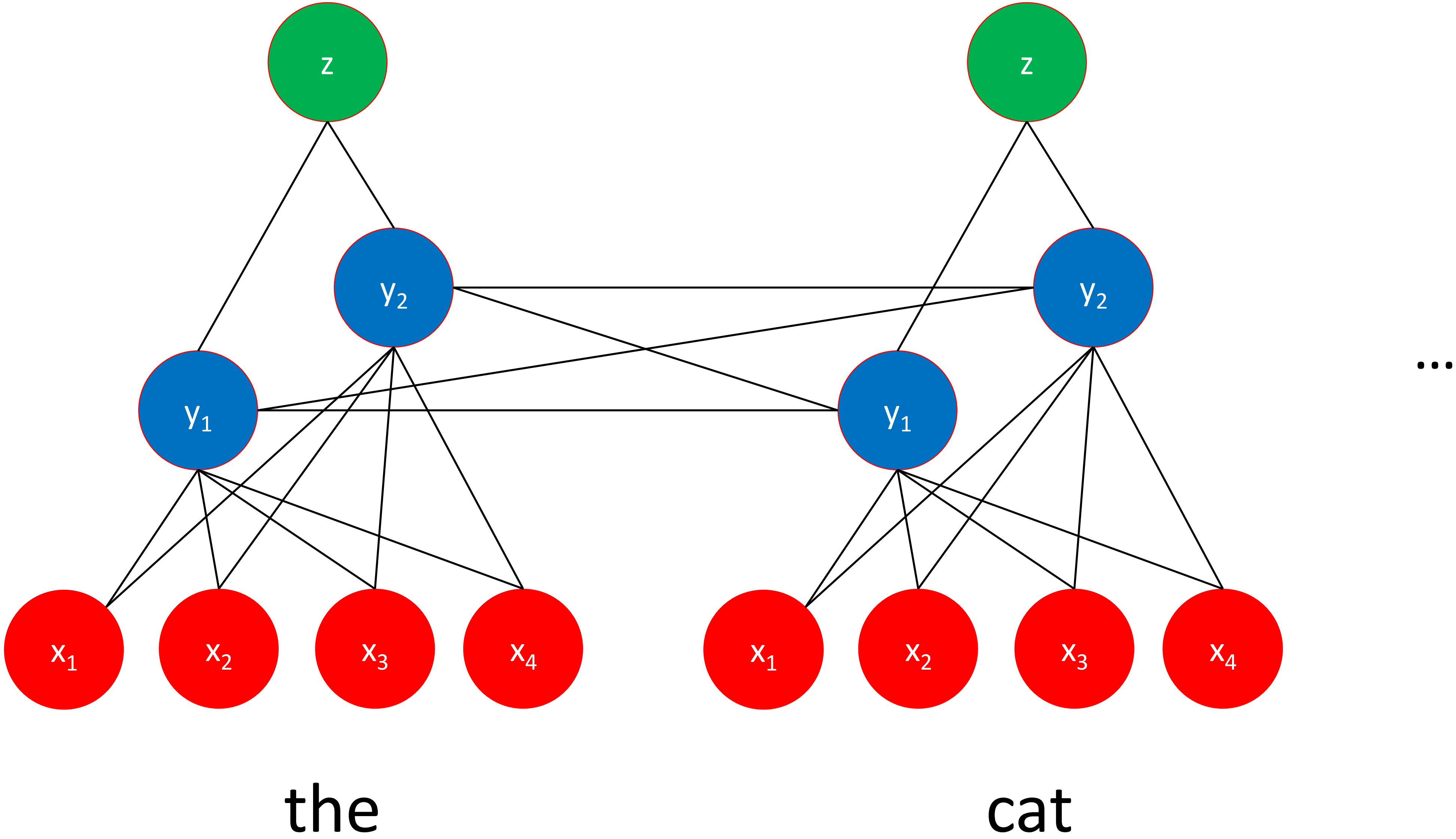


RNN Sequence



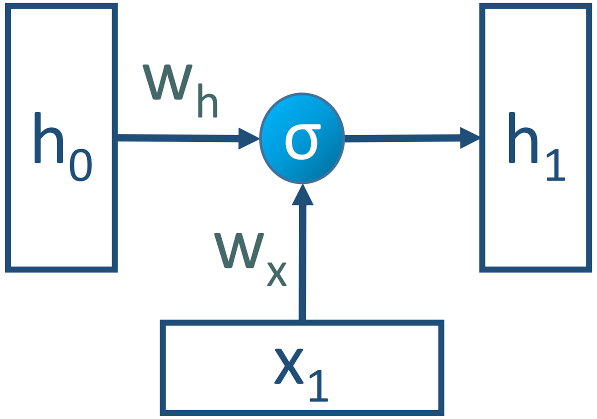
Example



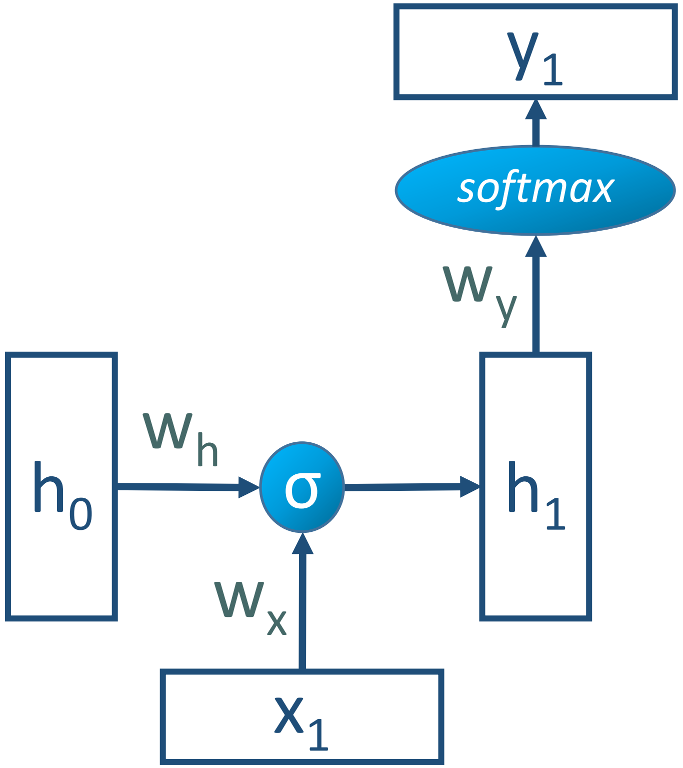

z
y$_2$
y$_1$
x$_1$
x$_2$
x$_3$
x$_4$
z
y$_2$
y$_1$
x$_1$
x$_2$
x$_3$
x$_4$
…
the
cat

Recurrent Neural Networks

$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$

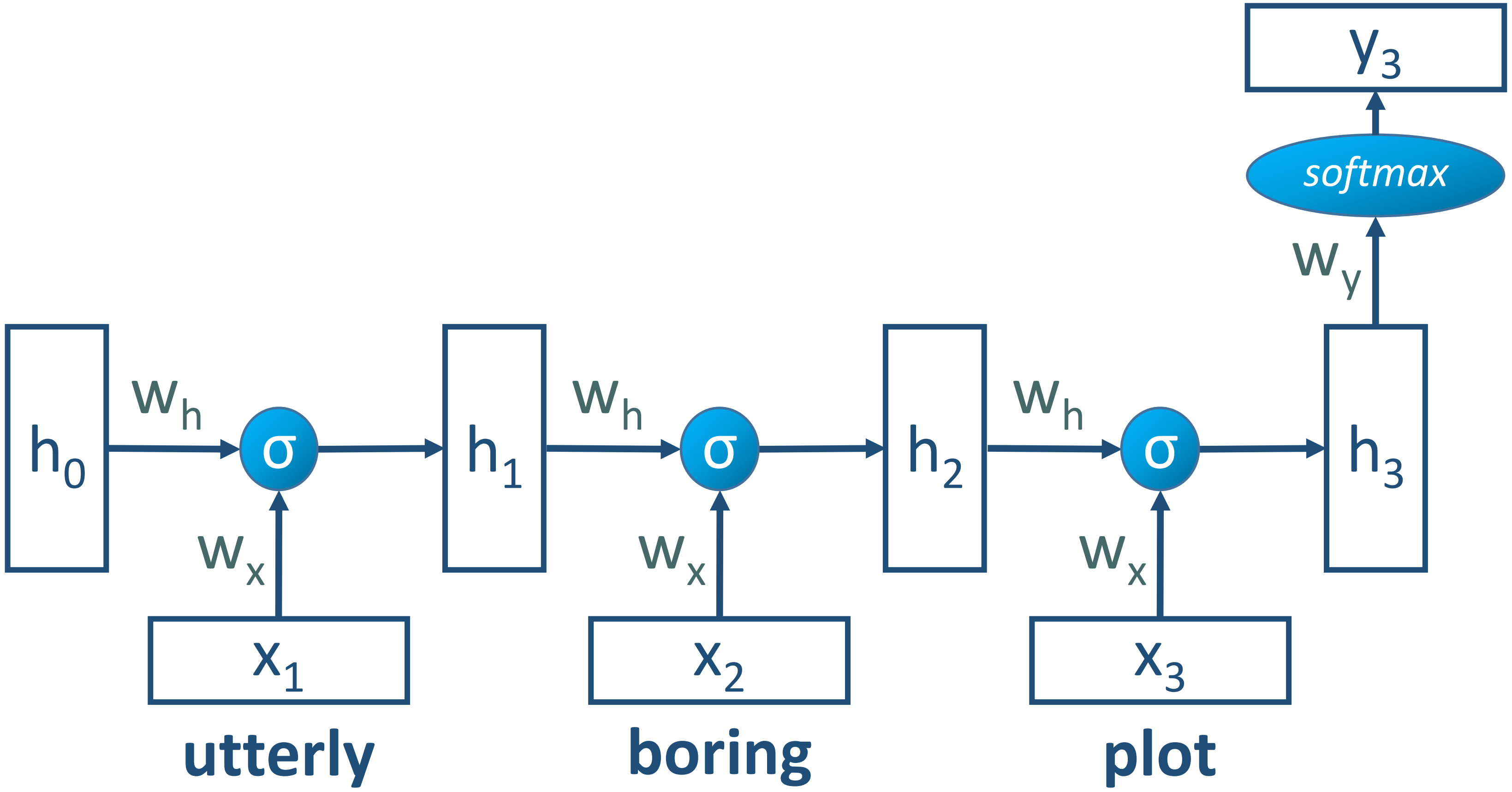


RNN

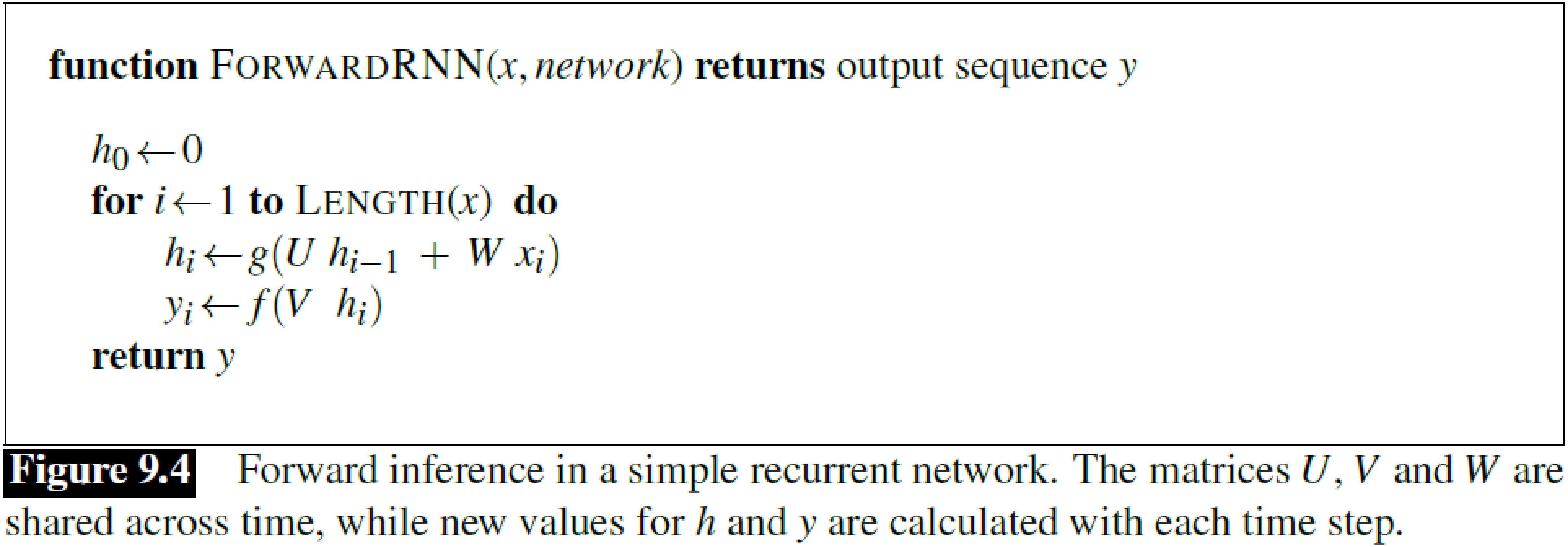


$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$
$$y_t = softmax(W_y h_t)$$

RNN


y_3
softmax
w_y
h_0
w_h
σ
h_1
w_h
σ
h_2
w_h
σ
h_3
w_x
w_x
w_x
x_1
x_2
x_3
utterly
boring
plot

Forward Inference

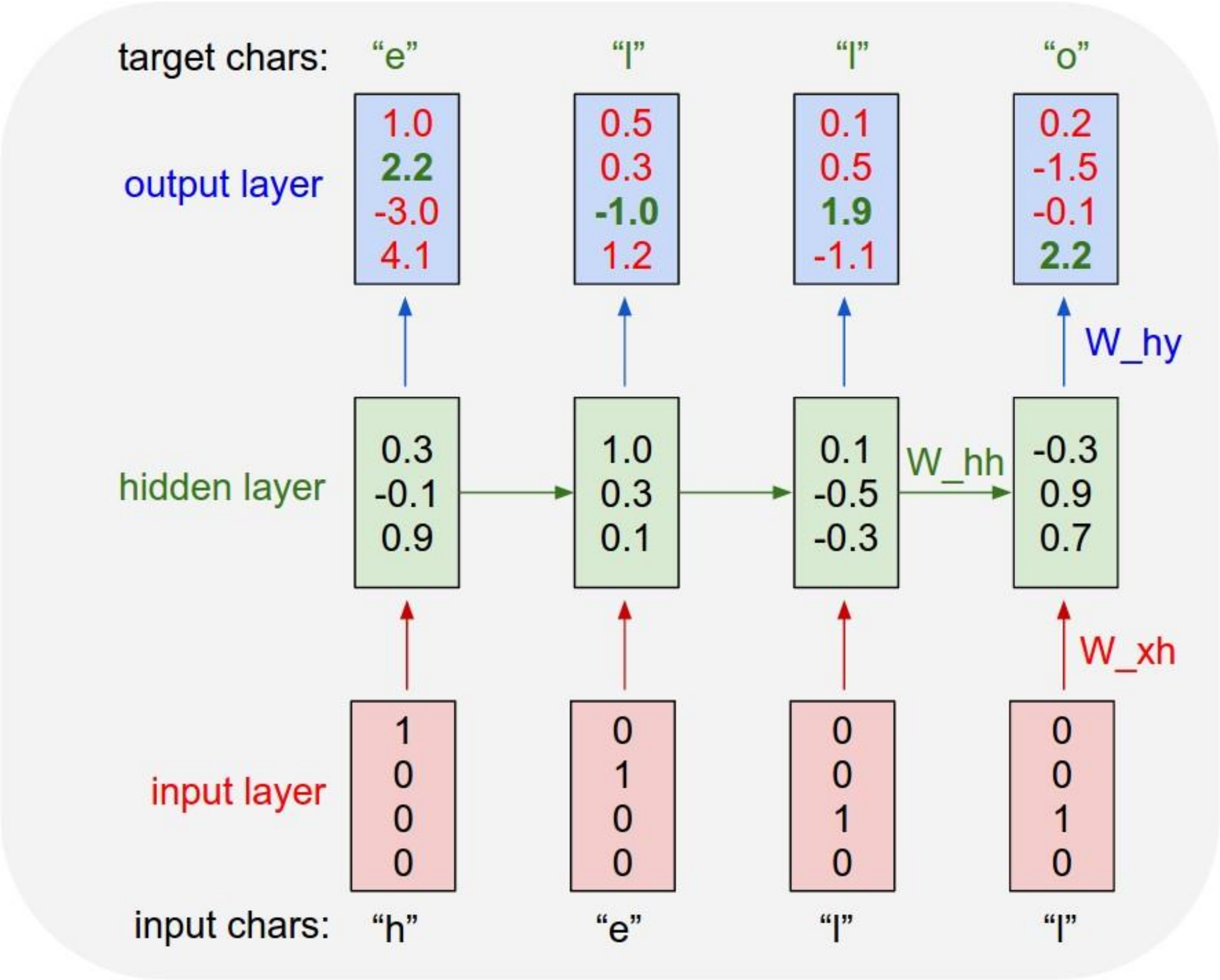


```
function ForwardRNN($x$, $network$) returns output sequence $y$

  $h_0 \leftarrow 0$
  for $i \leftarrow 1$ to Length($x$) do
      $h_i \leftarrow g(U\ h_{i-1}\ +\ W\ x_i)$
      $y_i \leftarrow f(V\ \ h_i)$
  return $y$
```

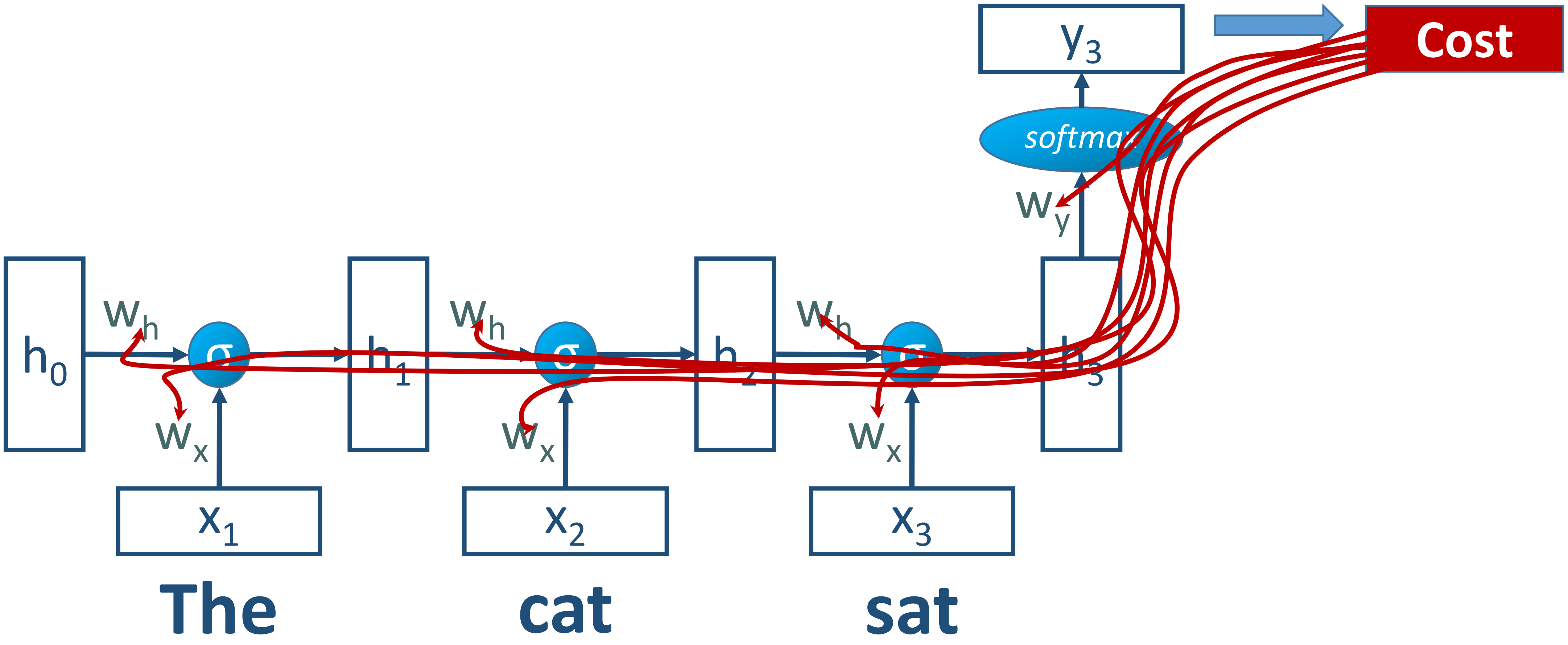
Figure 9.4 Forward inference in a simple recurrent network. The matrices U, V and W are shared across time, while new values for h and y are calculated with each time step.

Character-based RNN



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Updating Parameters of an RNN



Backpropagation through time

- The parameters are shared
- The derivatives are accumulated

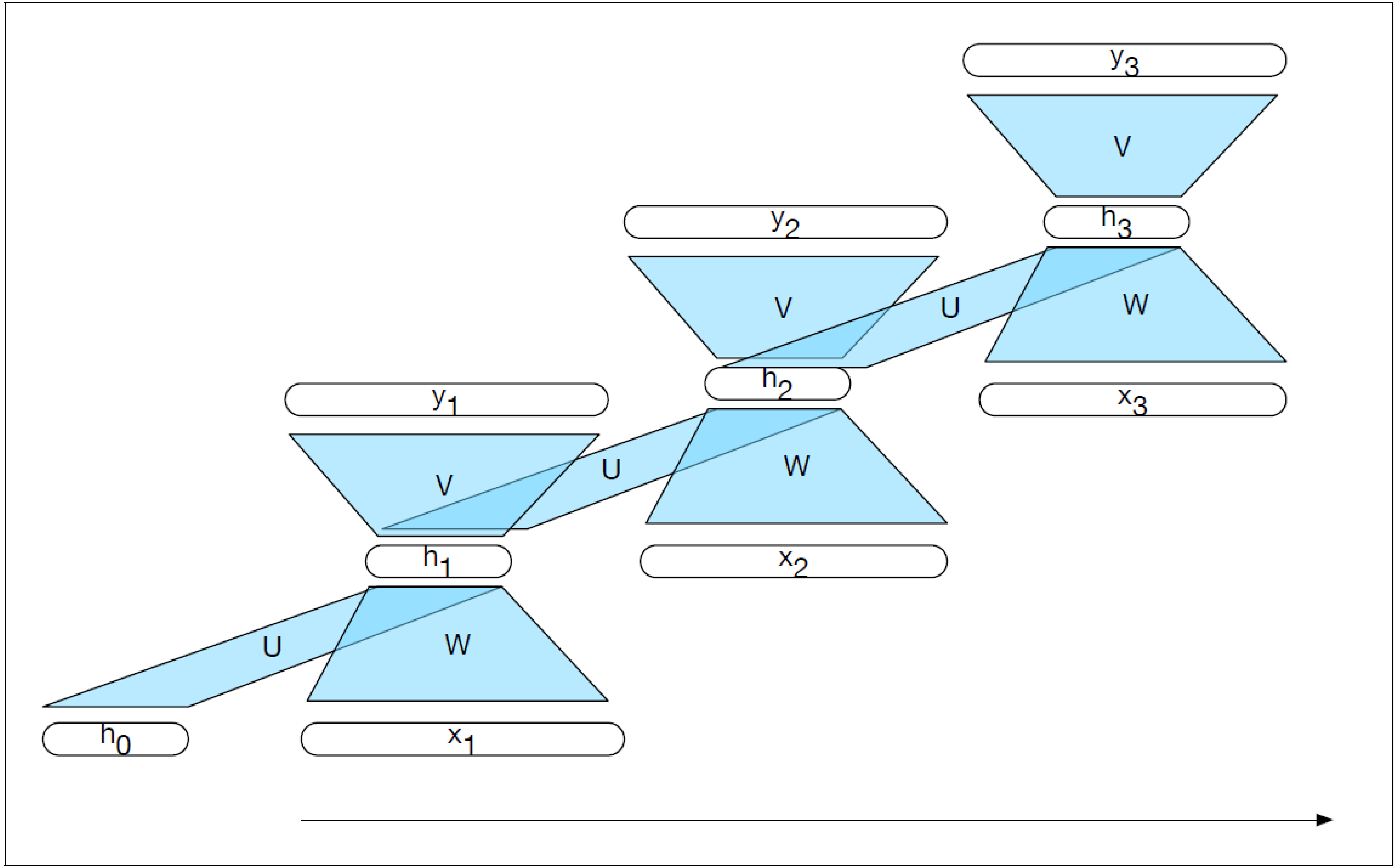


Figure 9.5 A simple recurrent neural network shown unrolled in time. Network layers are copied for each time step, while the weights U, V and W are shared in common across all time steps.

Backpropagation through time

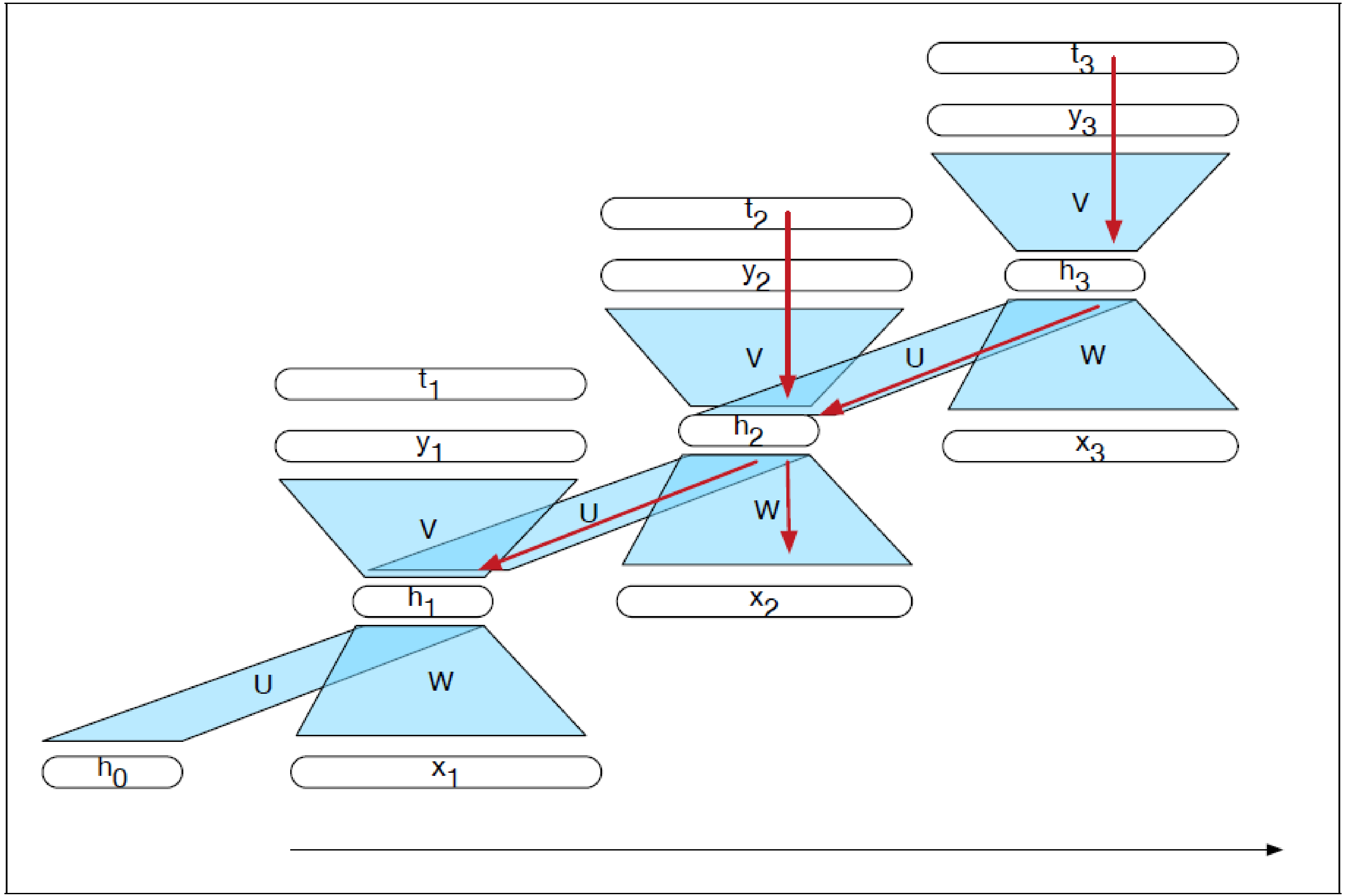


Figure 9.6 The backpropagation of errors in a simple RNN t_i vectors represent the targets for each element of the sequence from the training data. The red arrows illustrate the flow of backpropagated errors required to calculate the gradients for U, V and W at time 2. The two incoming arrows converging on h_2 signal that these errors need to be summed.

Generation with a neural language model

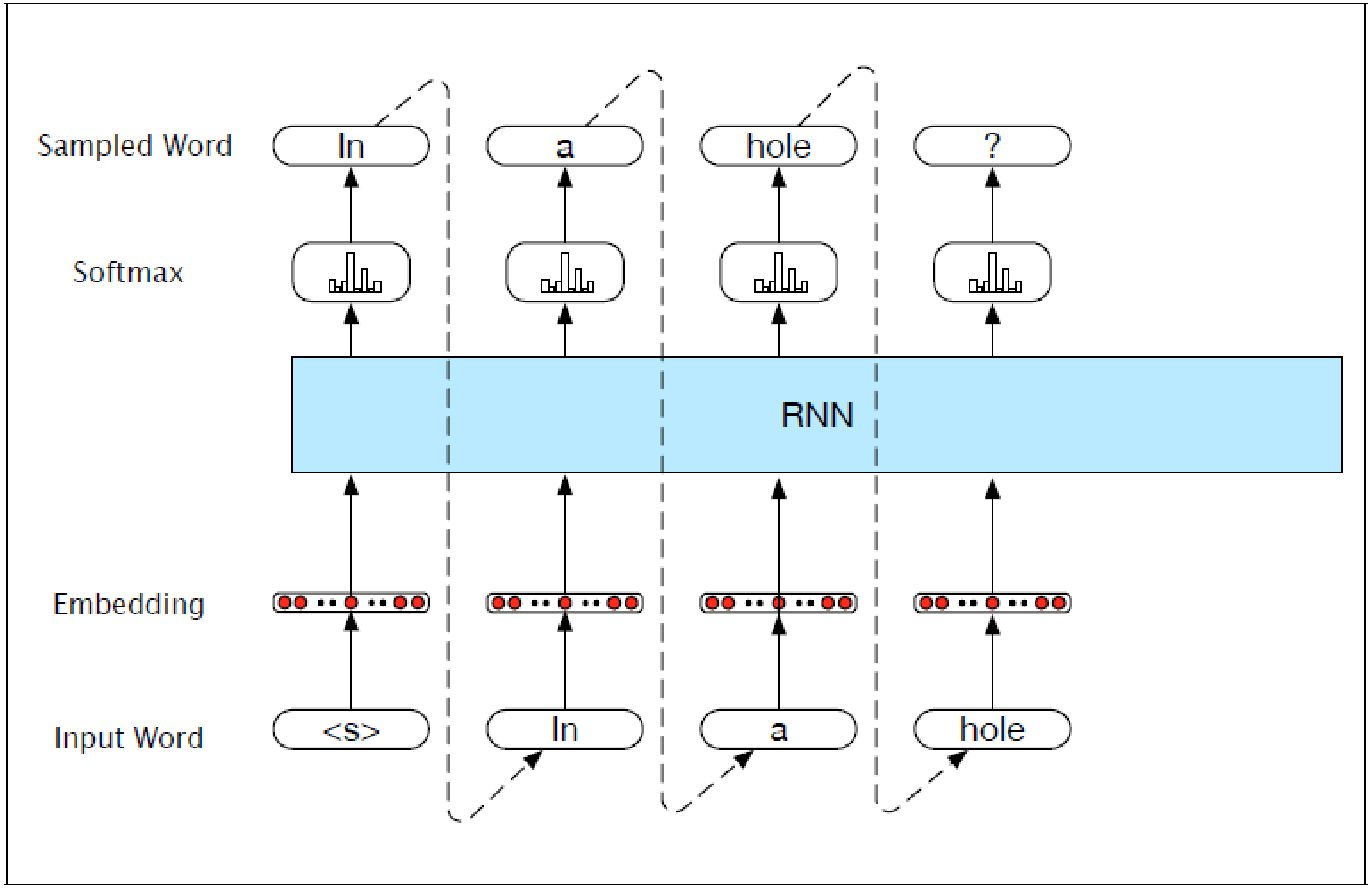


Figure 9.7 Autoregressive generation with an RNN-based neural language model.

Part of Speech Tagging

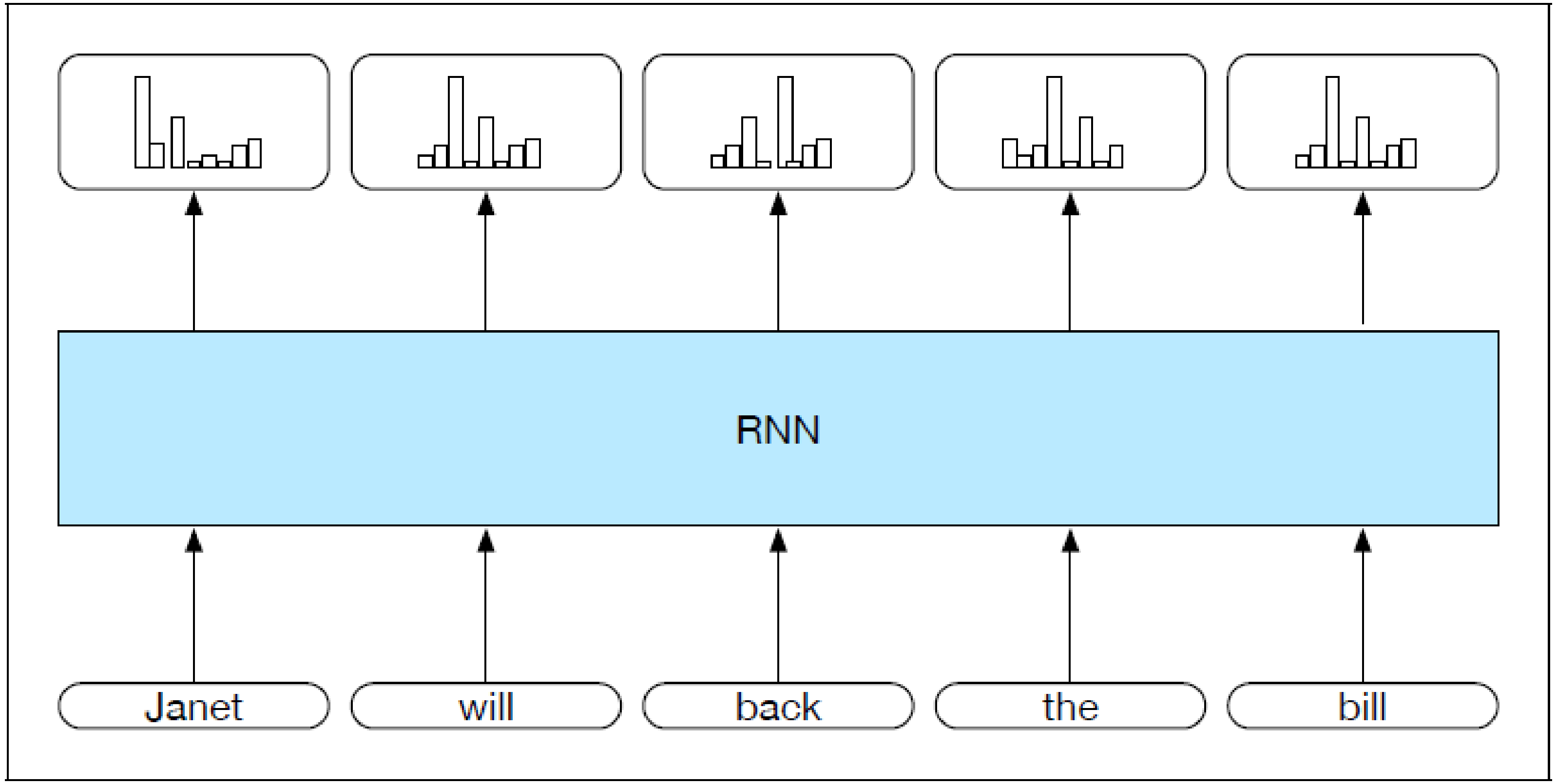


Figure 9.8 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

Sequence Classification

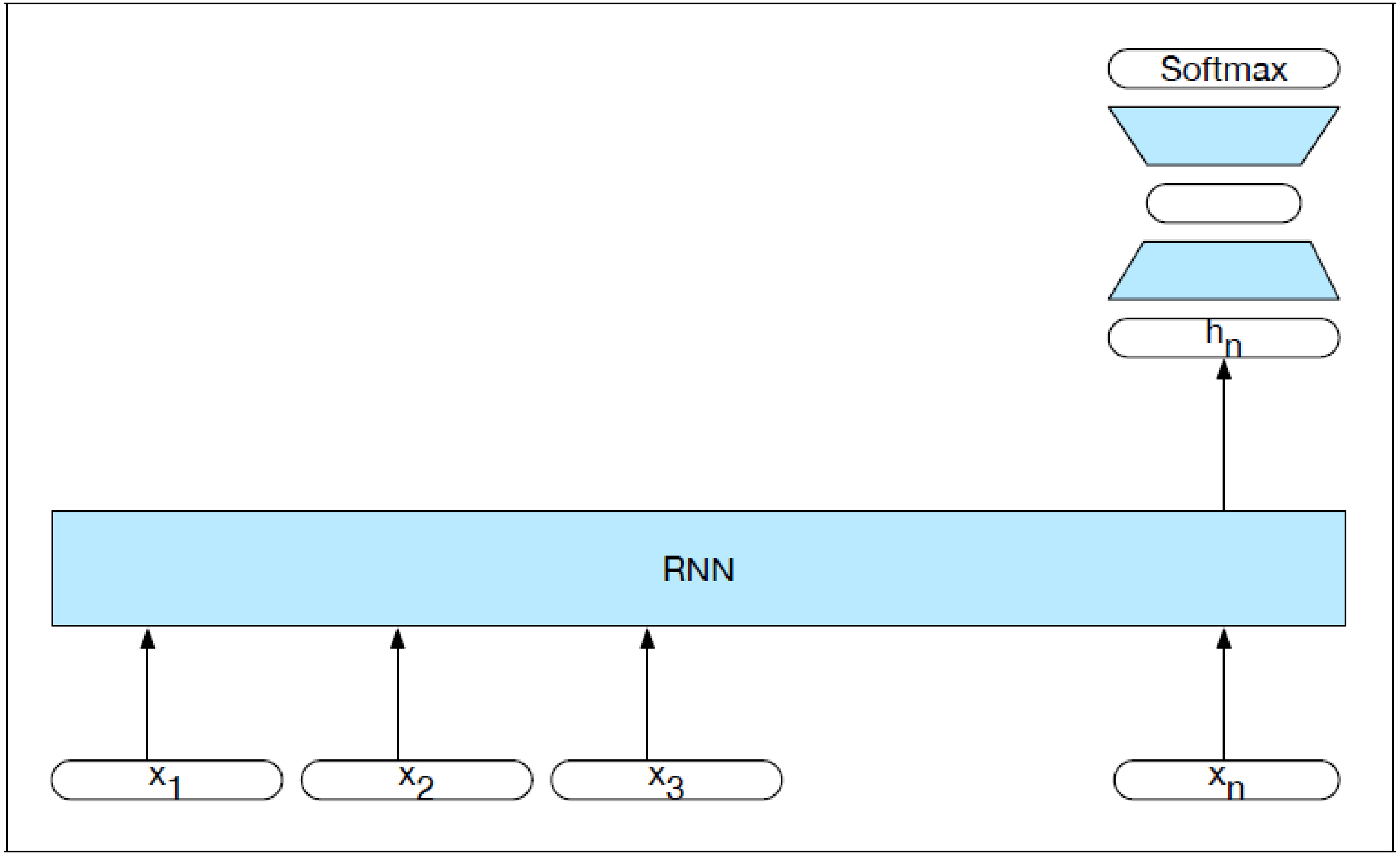


Figure 9.9 Sequence classification using a simple RNN combined with a feedforward network. The final hidden state from the RNN is used as the input to a feedforward network that performs the classification.

Uses of RNN

- RNNs are used to keep “memory”, just like finite-state automata
- They can be used as generators, acceptors, transducers (e.g., for machine translation), etc.
- Key application – language modeling: predicting the next word in the sequence

Other Applications

- Text generation
- Semantic role labeling:
  - http://www.aclweb.org/anthology/P15-1109
- Dependency parsing:
  - http://www.aclweb.org/anthology/K/K15/K15-1015.pdf

Stacked RNN

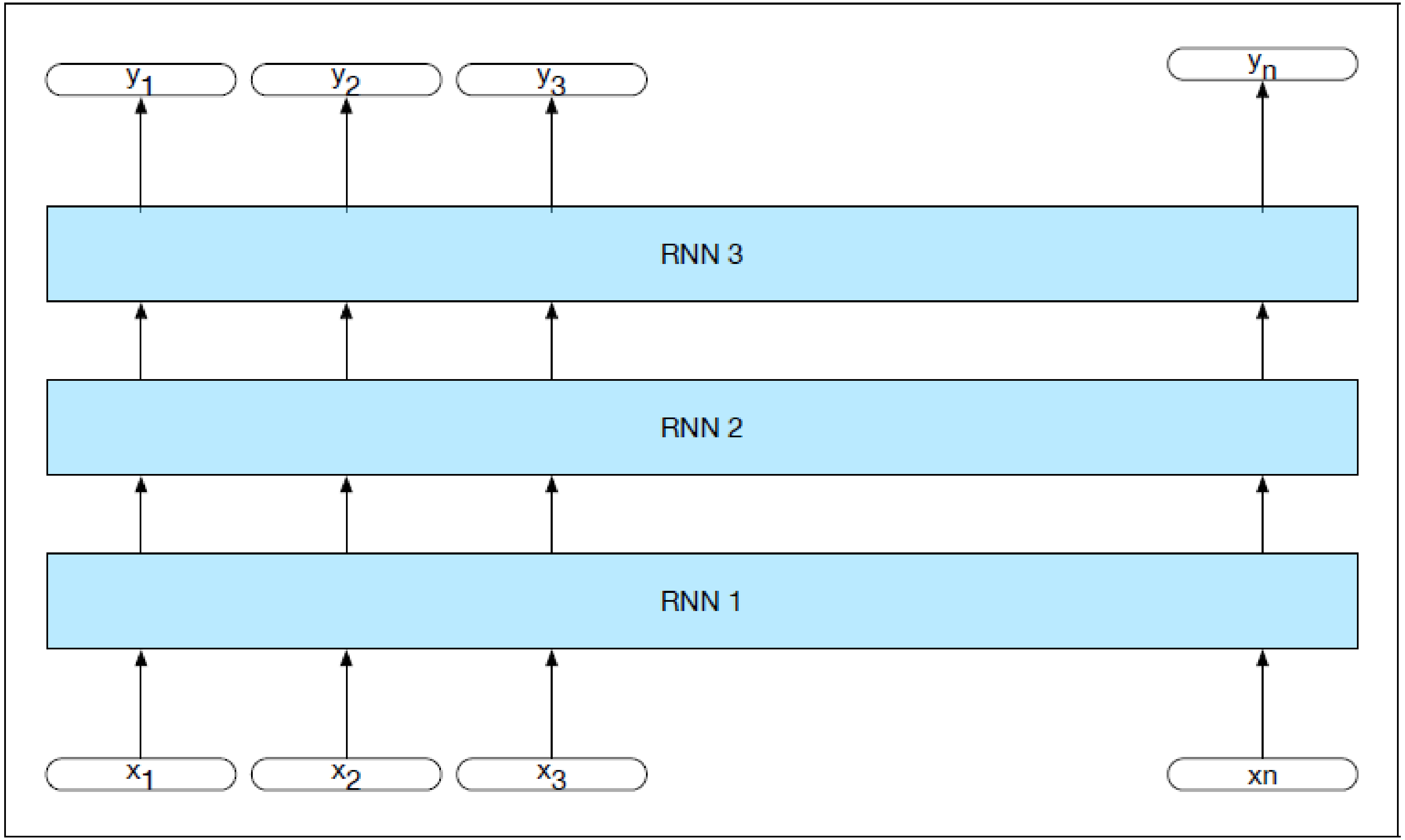


Figure 9.10 Stacked recurrent networks. The output of a lower level serves as the input to higher levels with the output of the last network serving as the final output.

Bidirectional RNN

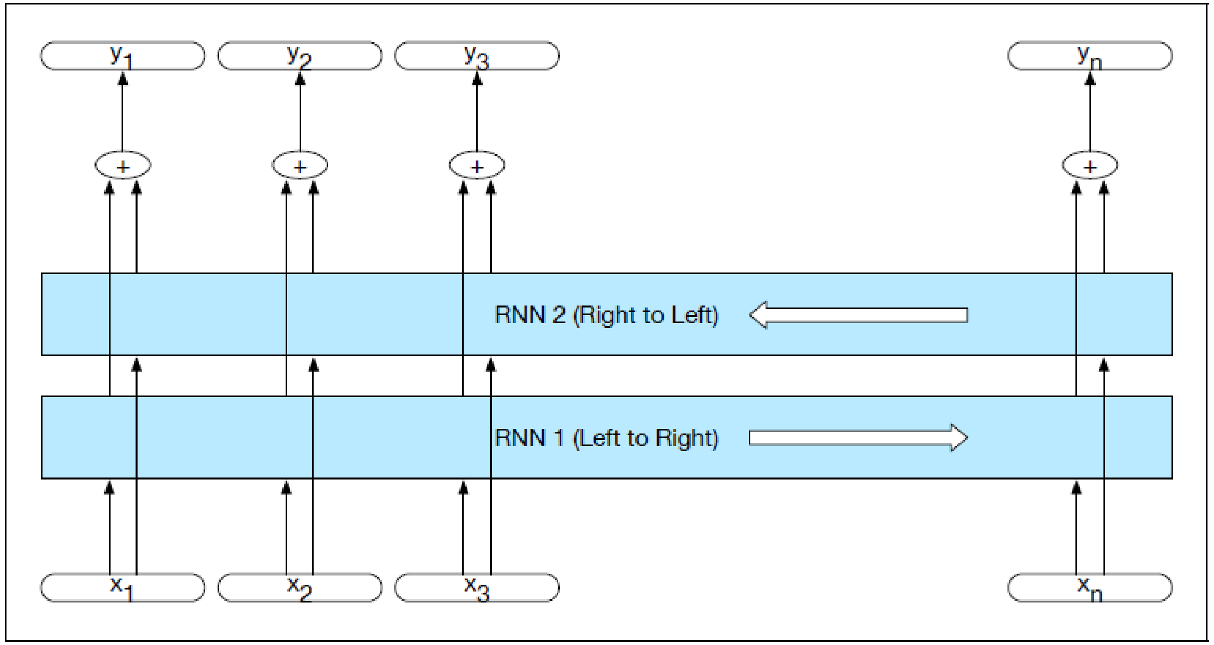


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions with the output of each model at each time point concatenated to represent the state of affairs at that point in time. The box wrapped around the forward and backward network emphasizes the modular nature of this architecture.

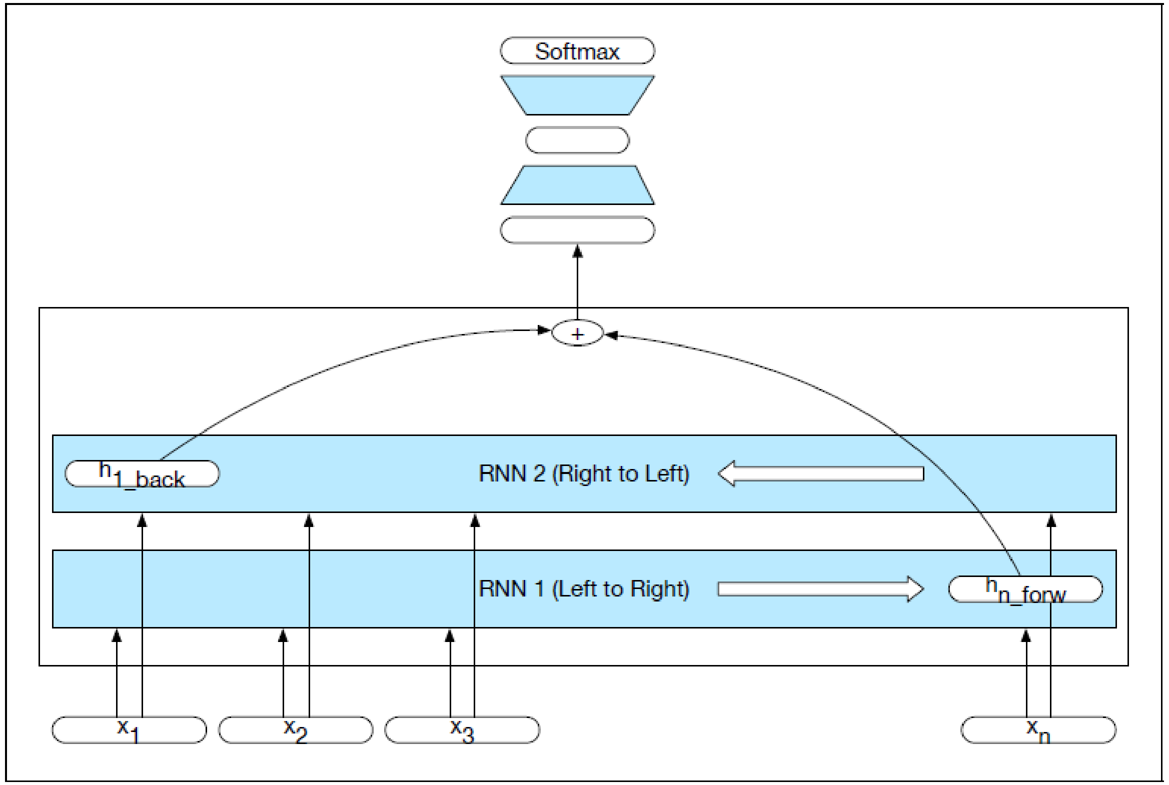


Figure 9.12 A bidirectional RNN for sequence classification. The final hidden units from the forward and backward passes are combined to represent the entire sequence. This combined representation serves as input to the subsequent classifier.

NLP

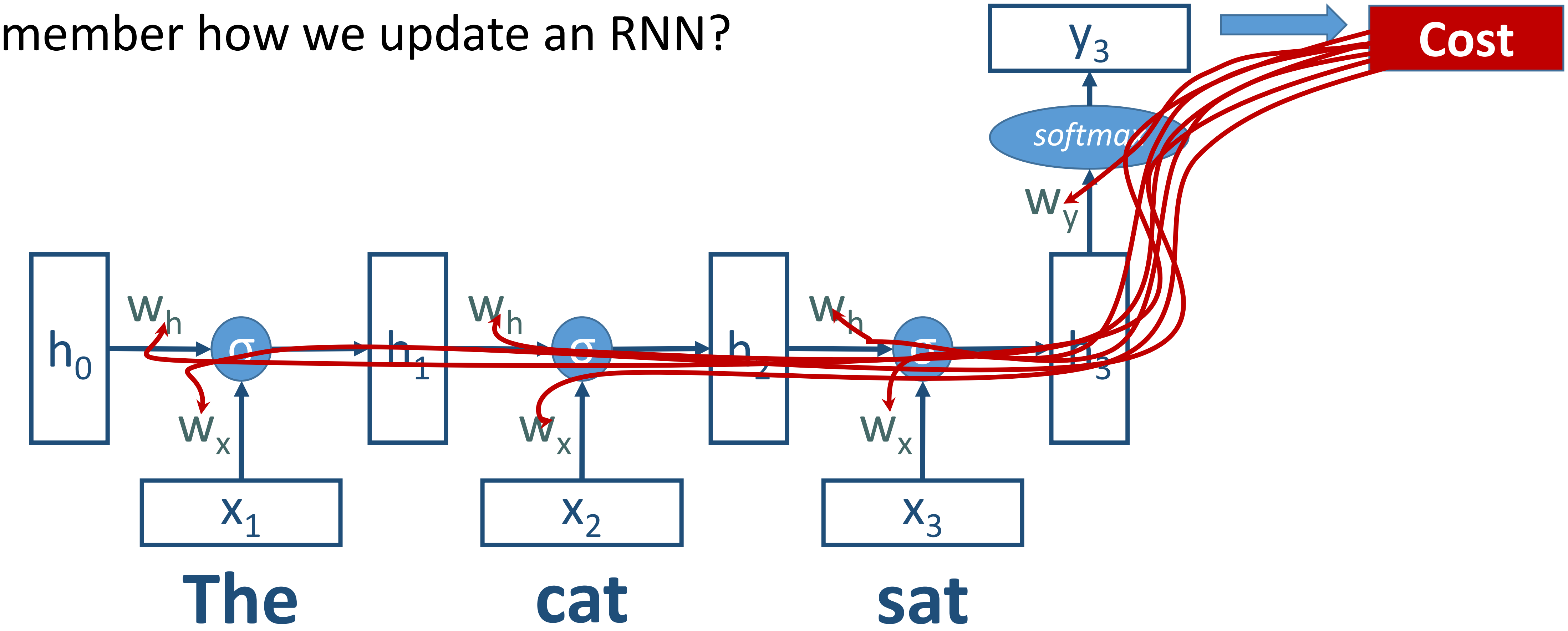
Deep Learning

742.

Long Short-Term Memory Networks LSTM

LSTM Motivation

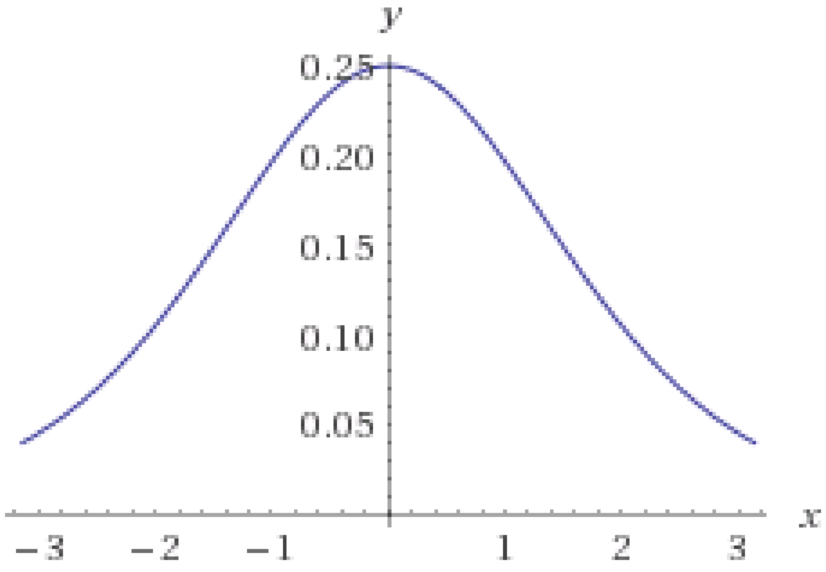
Remember how we update an RNN?



[slides from Catherine Finegan-Dollak]

The Vanishing Gradient Problem

- Deep neural networks use backpropagation.
- Back propagation uses the chain rule.
- The chain rule multiplies derivatives.
- Often these derivatives between 0 and 1.
- As the chain gets longer, products get smaller
- until they disappear.



Wolfram|Alpha

Derivative of sigmoid function

Or do they explode?

- With gradients larger than 1,
- you encounter the opposite problem
- with products becoming larger and larger
- as the chain becomes longer and longer,
- causing overlarge updates to parameters.
- This is the exploding gradient problem.

Vanishing/Exploding Gradients Are Bad

- If we cannot backpropagate very far through the network, the network cannot learn long-term dependencies.

- My dog [chase/chases] squirrels. ✓

vs.

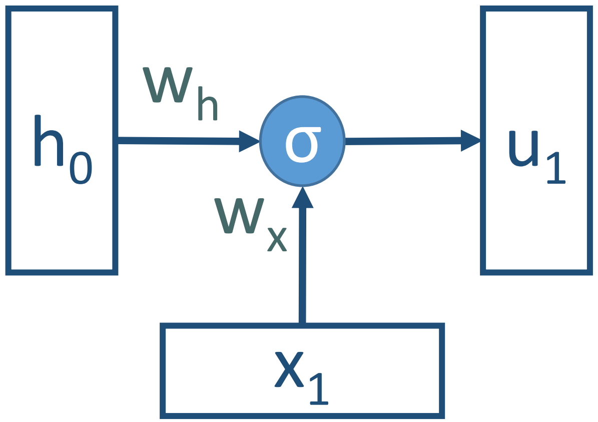
- My dog, whom I adopted in 2009, [chase/chases] squirrels. ✖

LSTM Solution

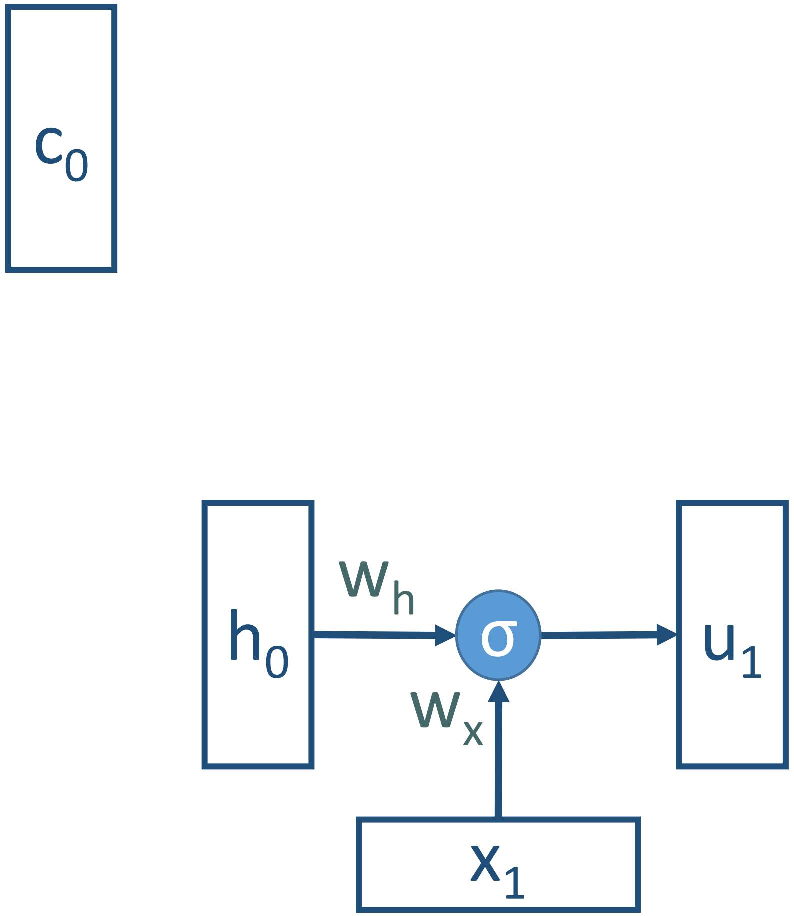
- Use memory cell to store information at each time step.
- Use “gates” to control the flow of information through the network.
    - Input gate: protect the current step from irrelevant inputs
    - Output gate: prevent the current step from passing irrelevant outputs to later steps
    - Forget gate: limit information passed from one cell to the next

Transforming RNN to LSTM

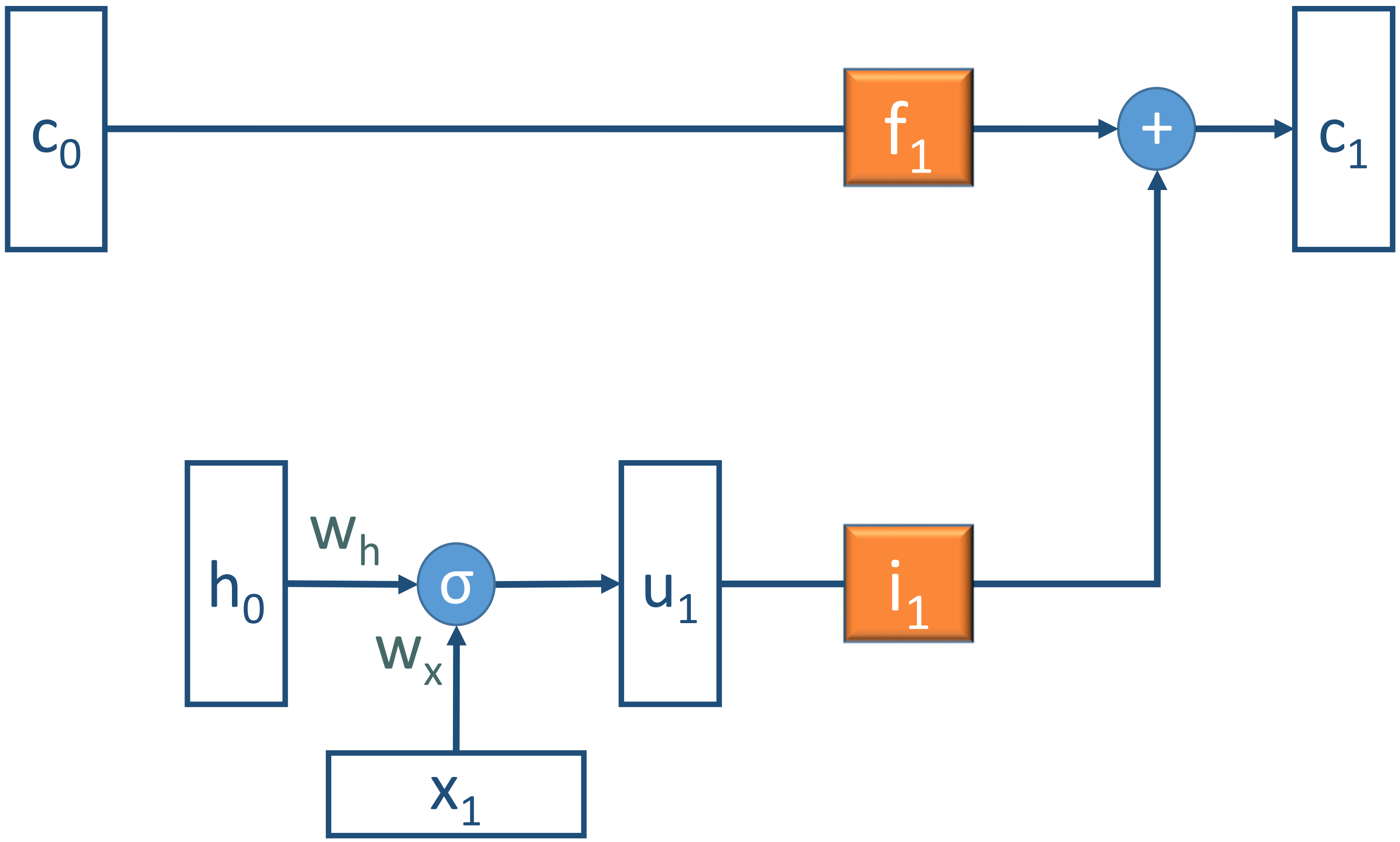
$$u_t = \sigma(W_h h_{t-1} + W_x x_t)$$



Transforming RNN to LSTM



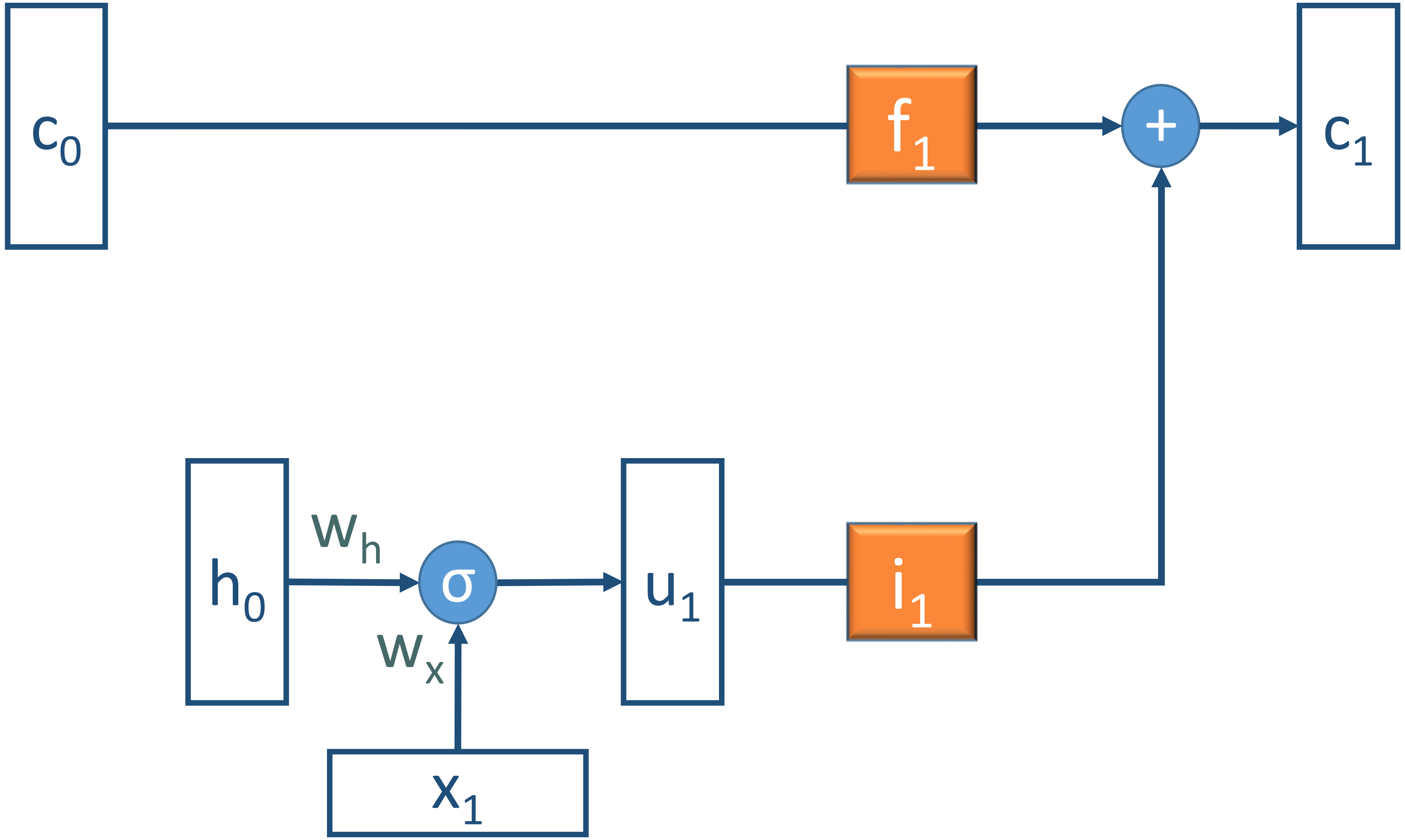
Transforming RNN to LSTM



$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

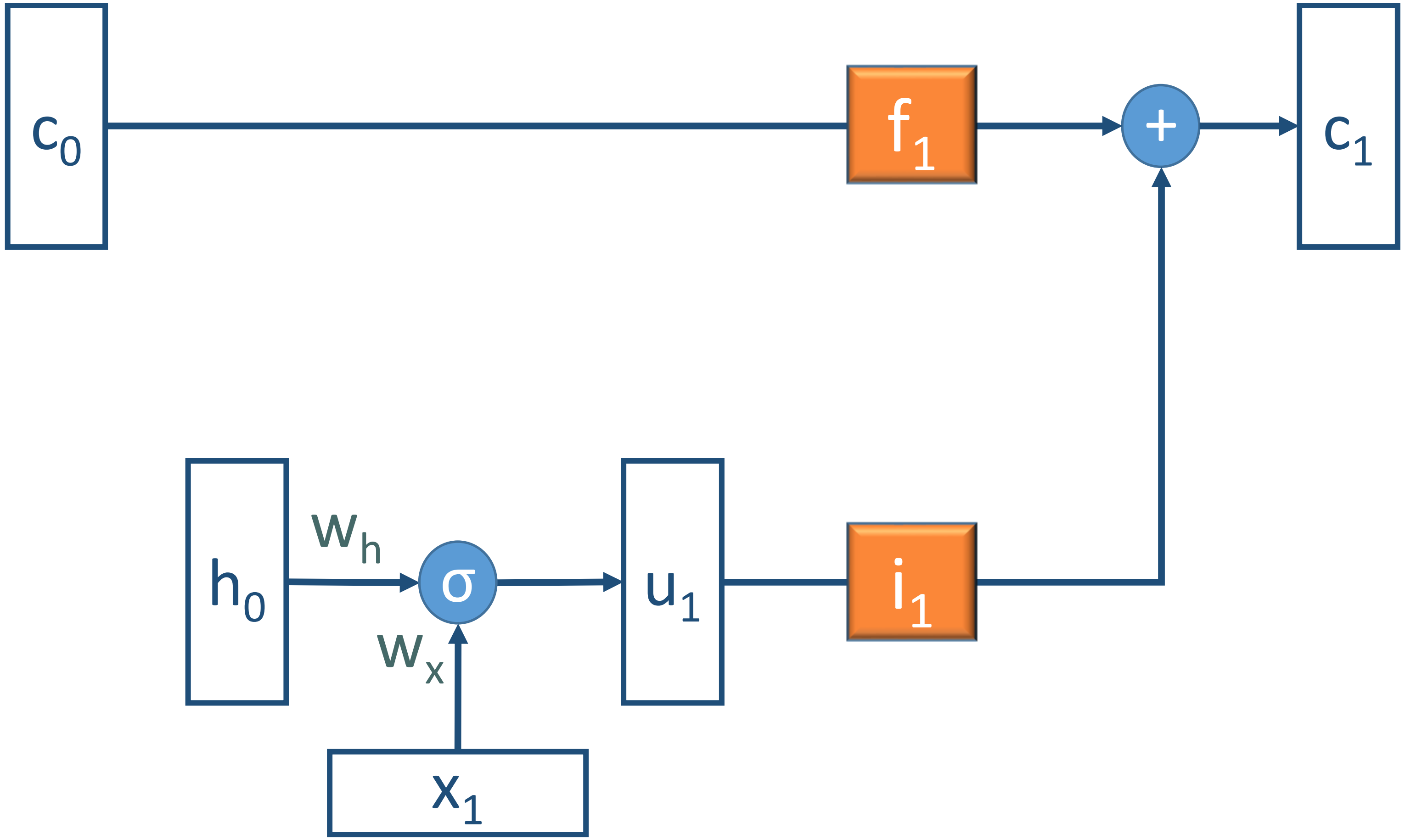
Elementwise (Hadamard) matrix product
(array product “*” in Python)

Transforming RNN to LSTM



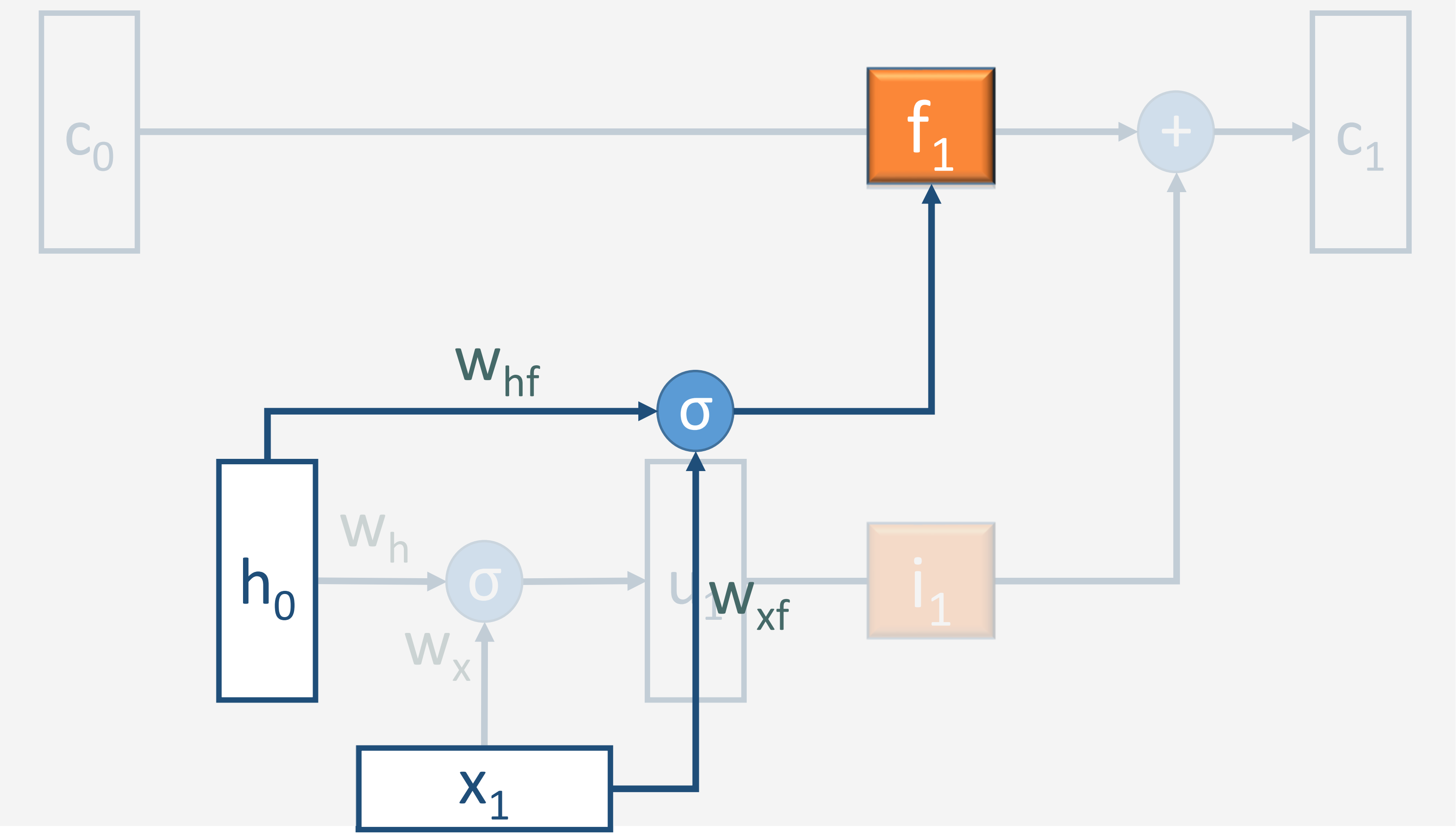
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

Transforming RNN to LSTM



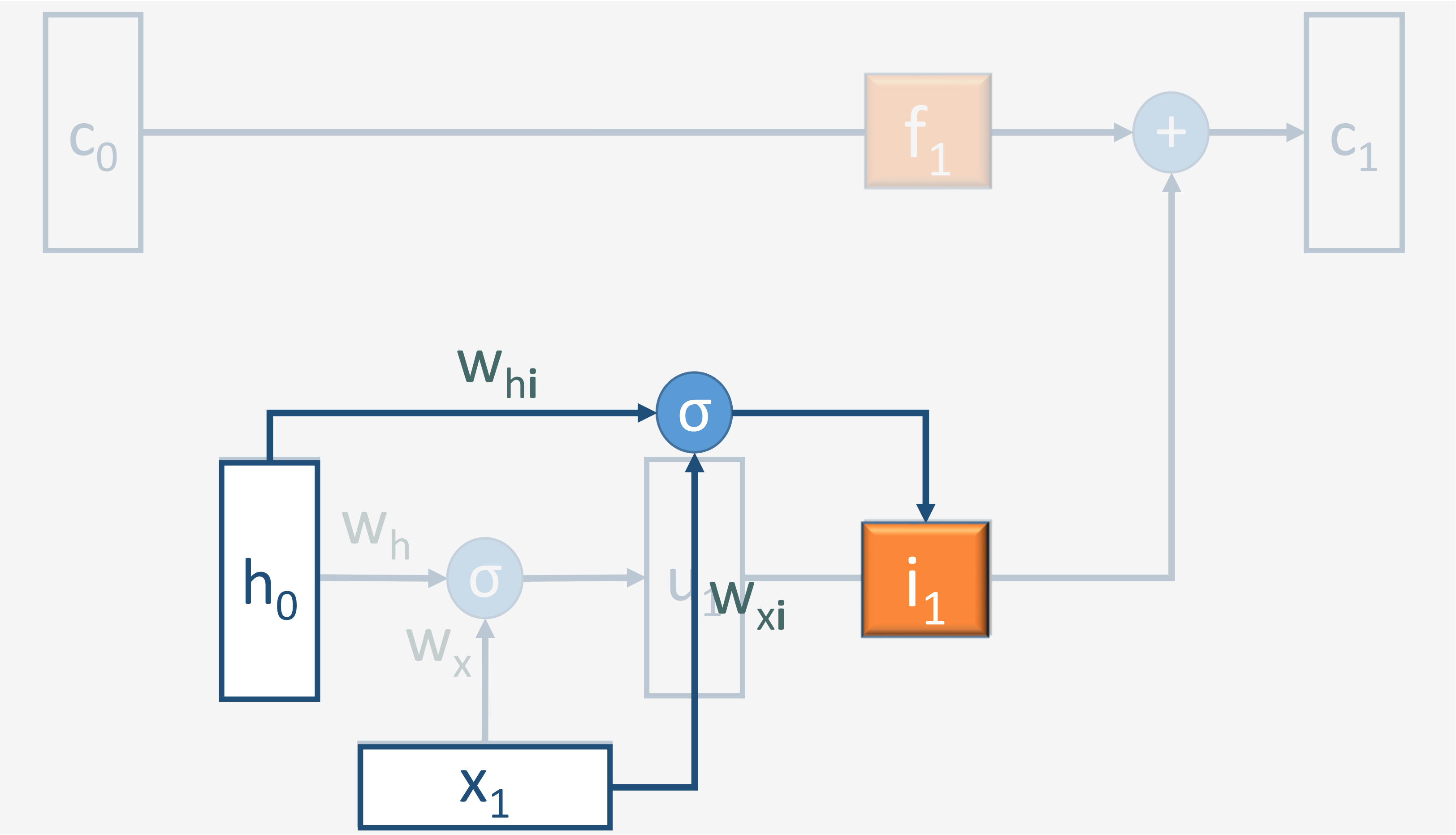
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

Transforming RNN to LSTM



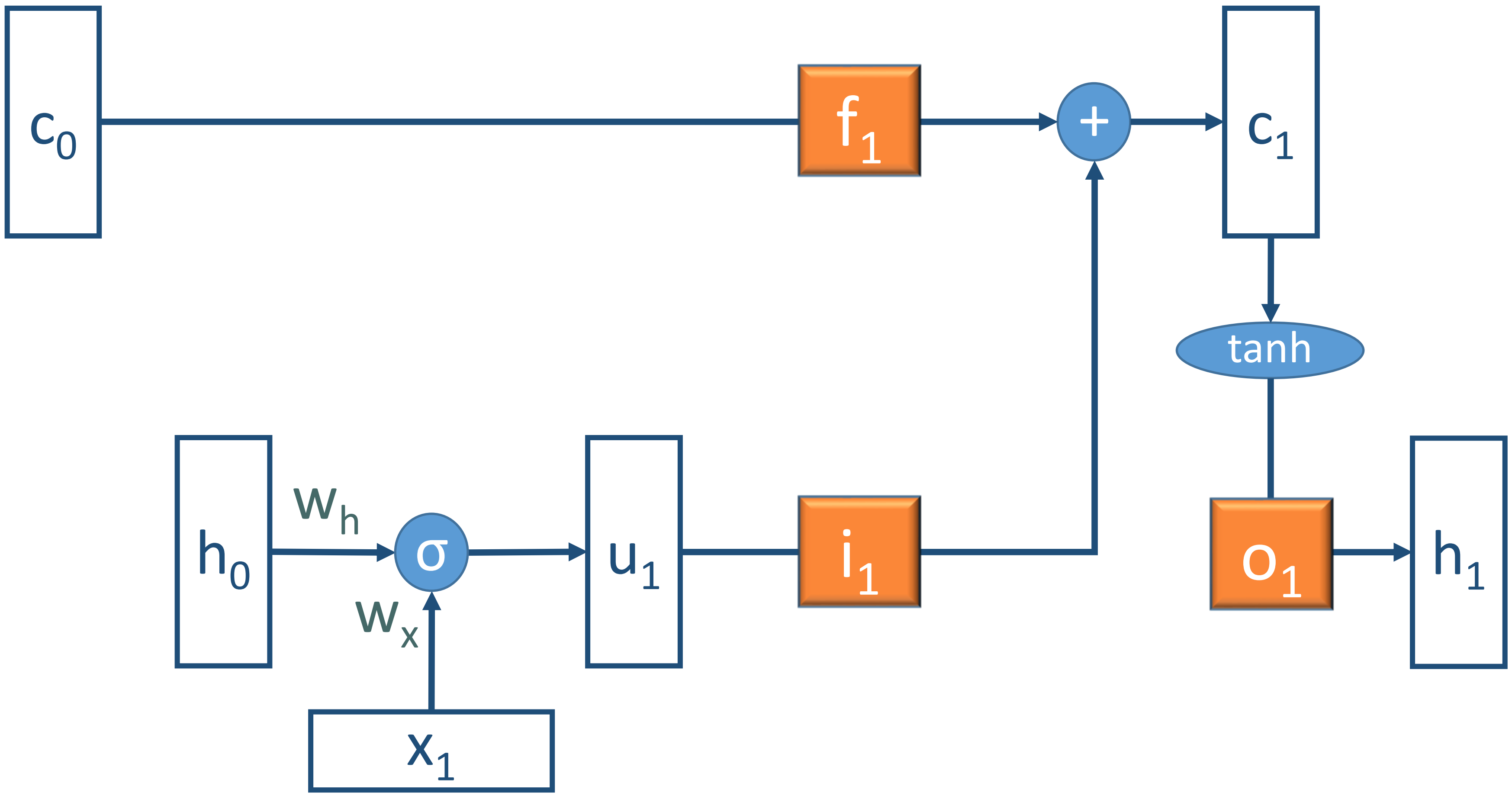
$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t)$$

Transforming RNN to LSTM



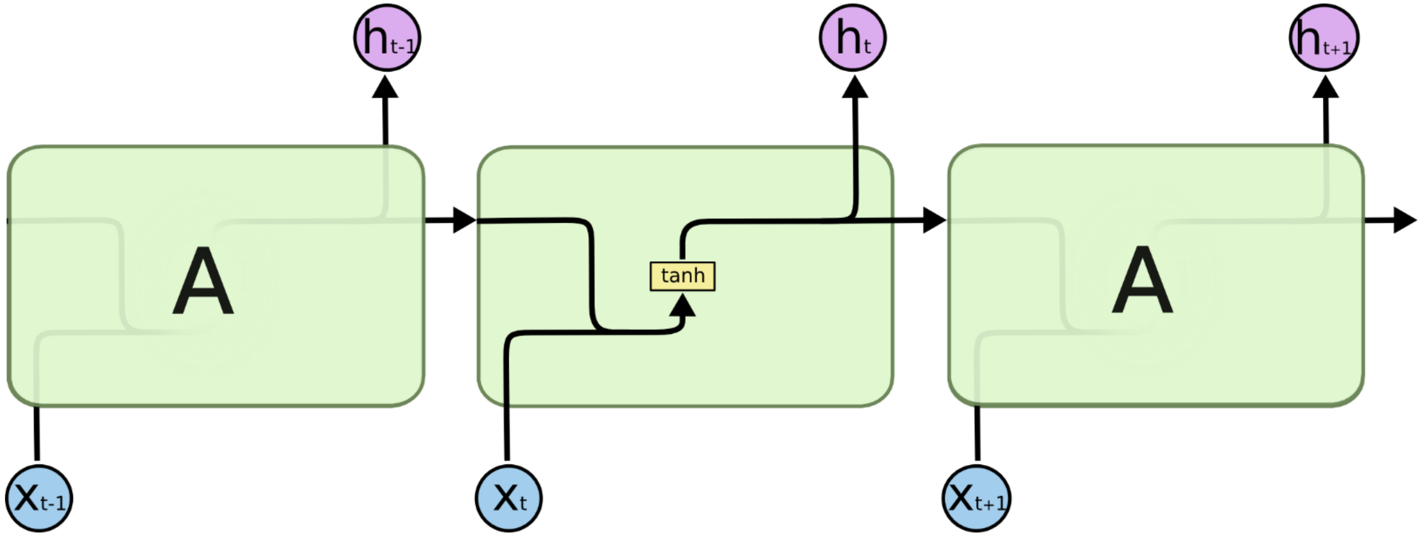
$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$$

Transforming RNN to LSTM

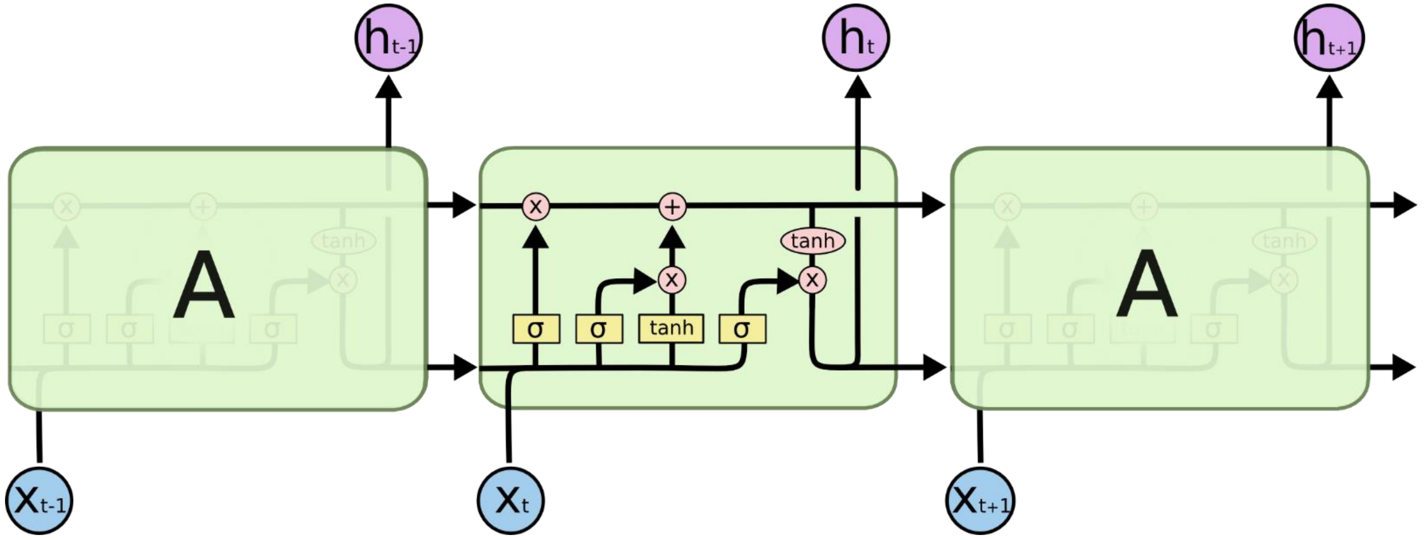


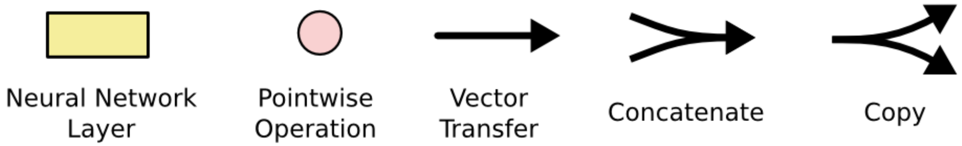
$$h_t = o_t \odot \tanh c_t$$

RNN

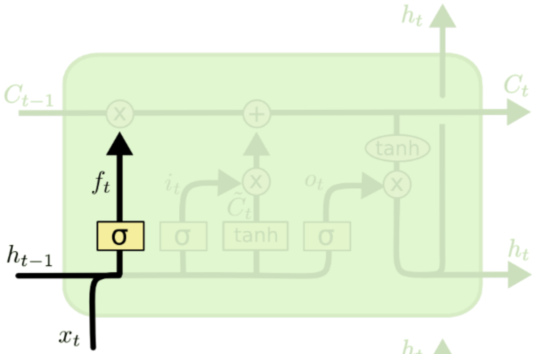


LSTM

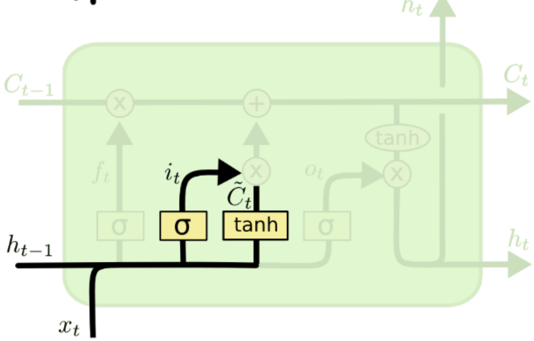




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

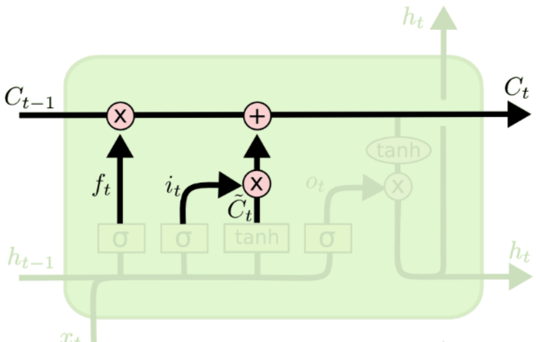


$$f_t = \sigma\left(W_f\cdot[h_{t-1},x_t] \ + \ b_f\right)$$

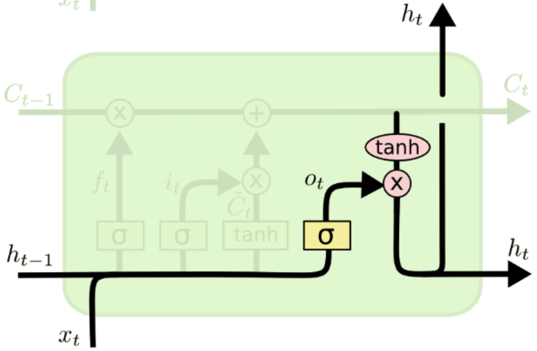


$$i_t = \sigma\left(W_i\cdot[h_{t-1},x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C\cdot[h_{t-1},x_t] \ + \ b_C)$$



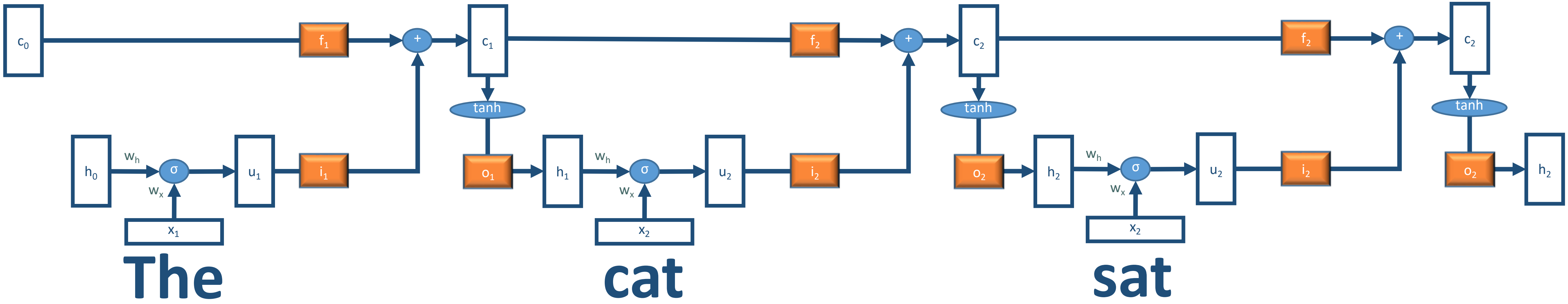
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma\left(W_o \ [\ h_{t-1},x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM for Sequences



LSTM Applications

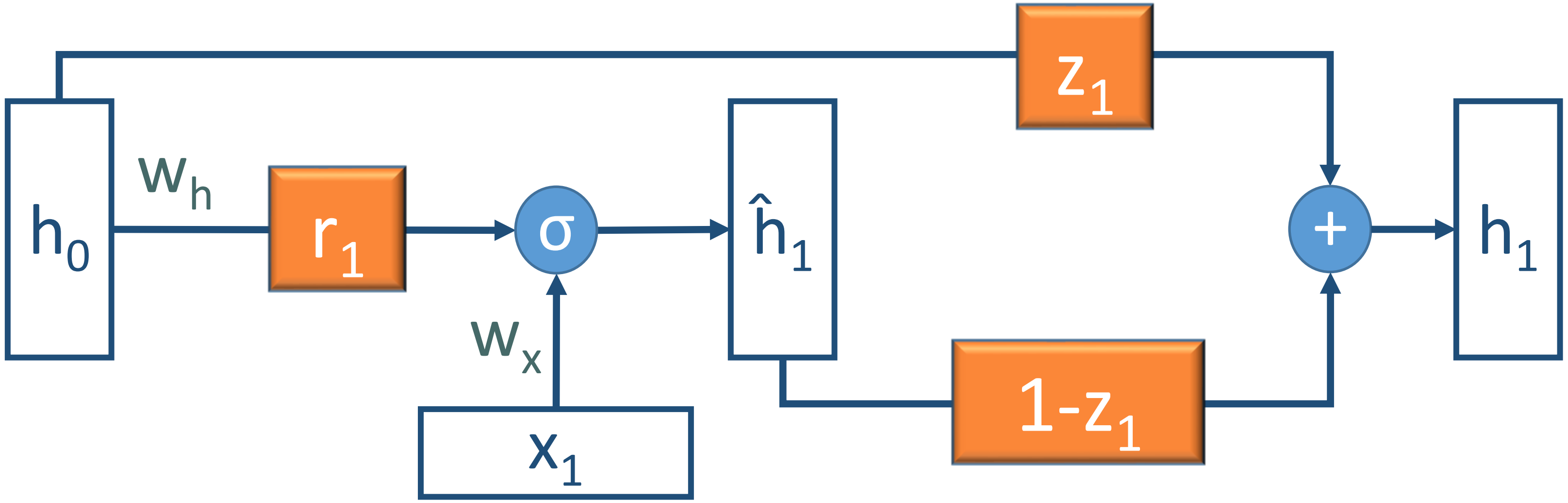
- Language identification (Gonzalez-Dominguez et al., 2014)
- Paraphrase detection (Cheng & Kartsaklis, 2015)
- Speech recognition (Graves, Abdel-Rahman, & Hinton, 2013)
- Handwriting recognition (Graves & Schmidhuber, 2009)
- Music composition (Eck & Schmidhuber, 2002) and lyric generation (Potash, Romanov, & Rumshisky, 2015)
- Robot control (Mayer et al., 2008)
- Natural language generation (Wen et al. 2015) (best paper at EMNLP)
- Named entity recognition (Hammerton, 2003)

Handwriting generation http://www.cs.toronto.edu/~graves/handwriting.html

Other Architectures

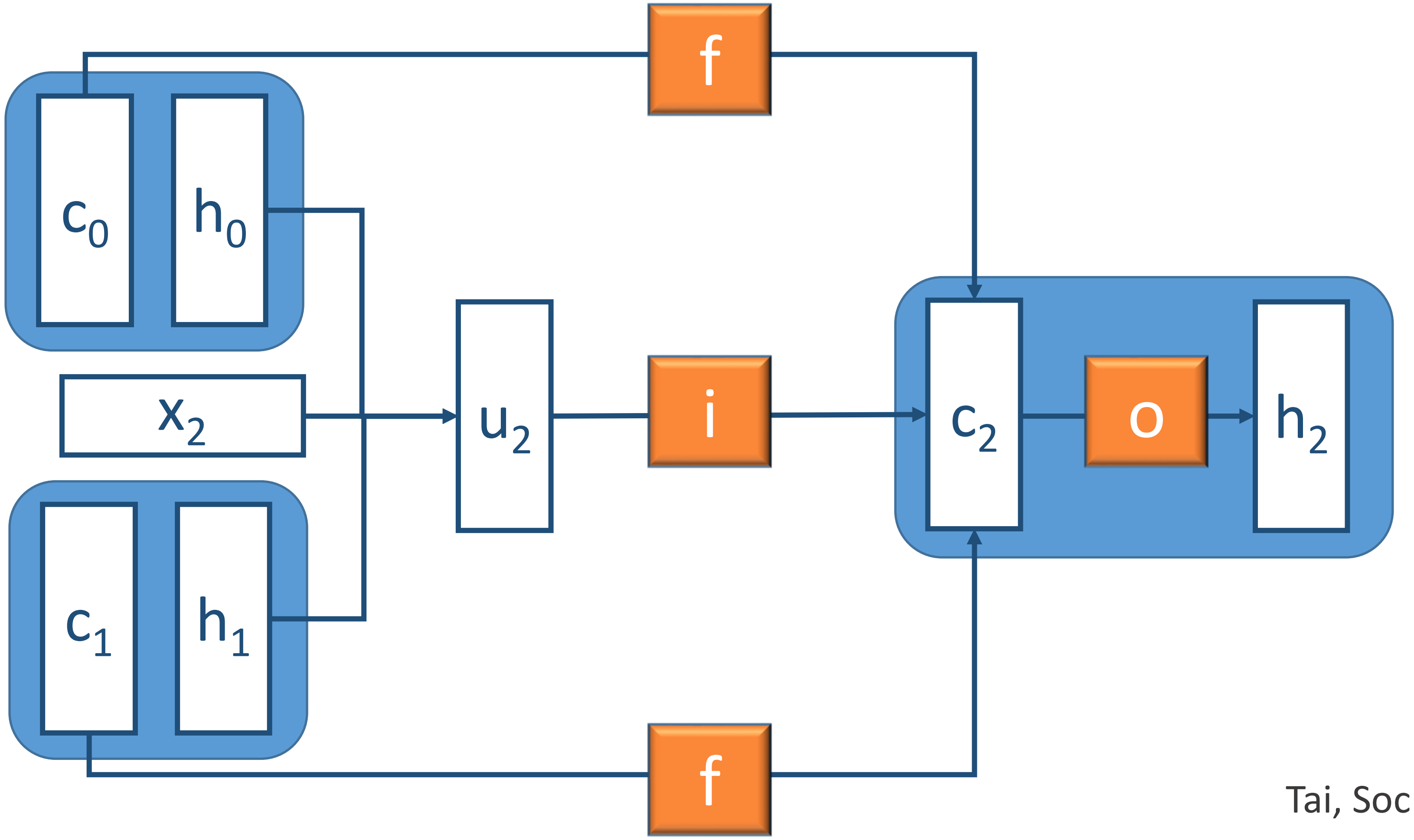
- Bidirectional LSTM
    - Concatenate two one-directional LSTMs
- Stacked LSTM

Related Architectures: GRU



Chung et al. (2014) reports comparable performance to LSTM

Related Architectures: Tree LSTMs



Tai, Socher, Manning 2015

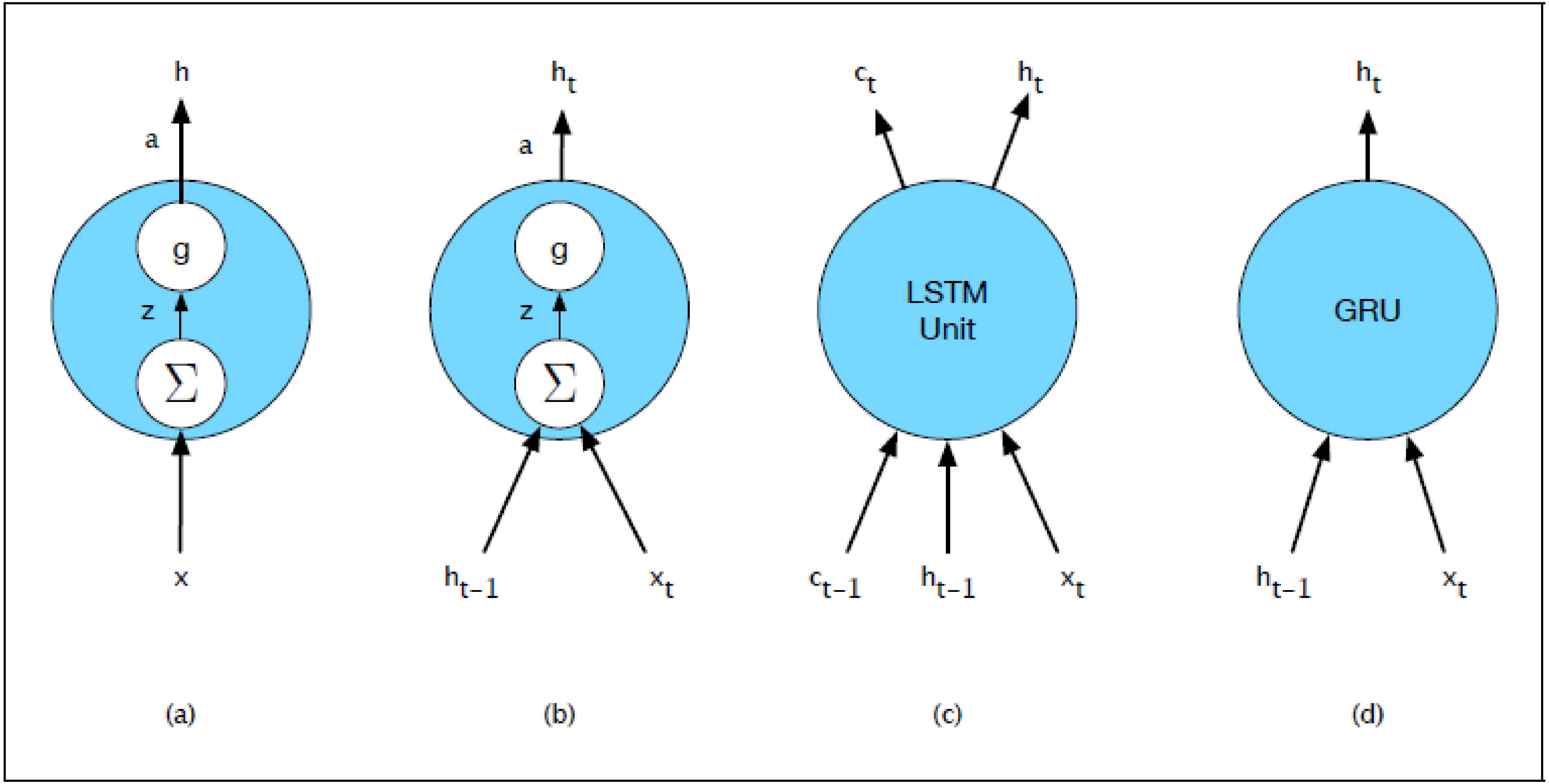


Figure 9.14 Basic neural units used in feedforward, simple recurrent networks (SRN), long short-term memory (LSTM) and gate recurrent units.

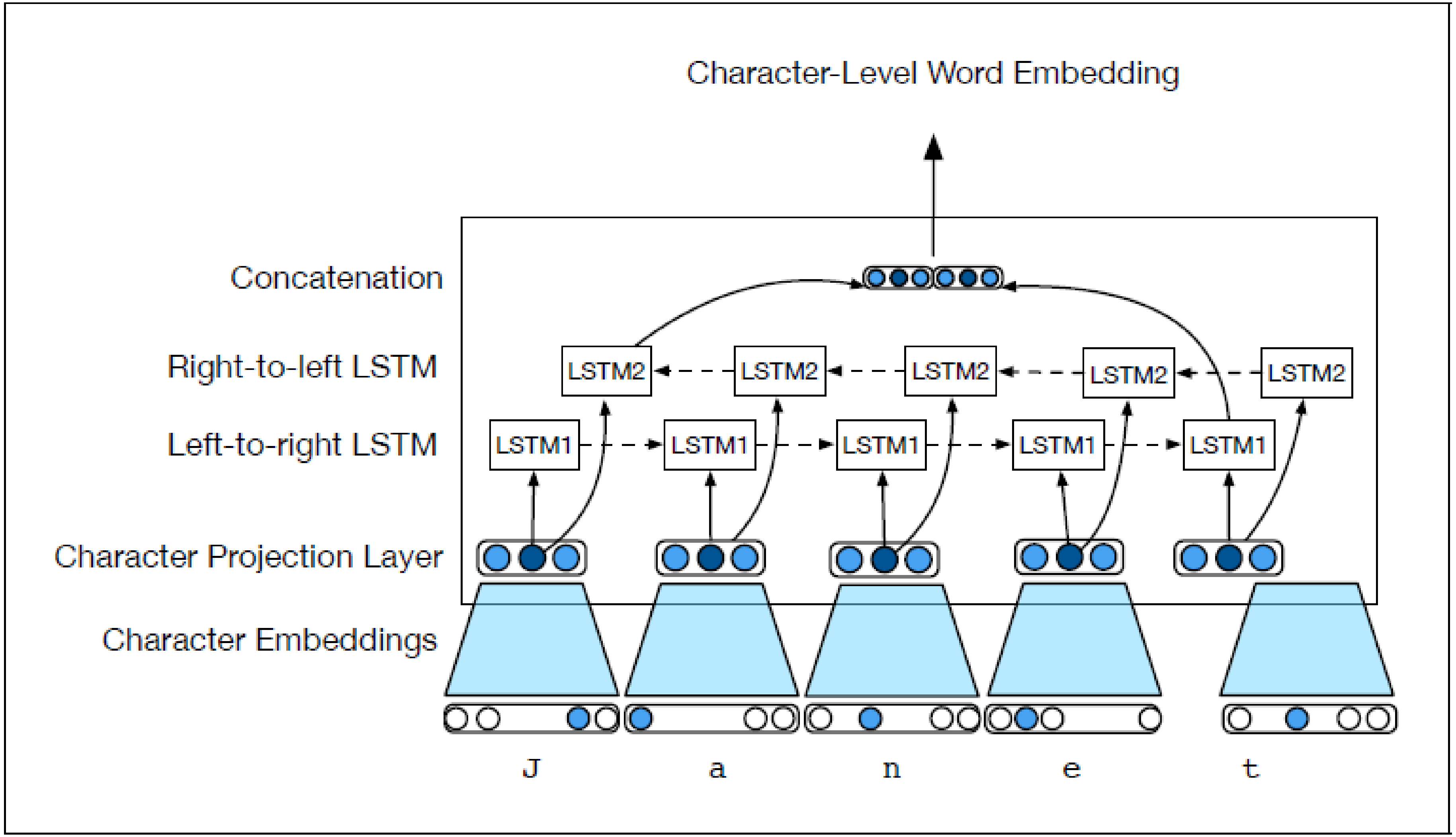


Figure 9.16 Bi-RNN accepts word character sequences and emits embeddings derived from a forward and backward pass over the sequence. The network itself is trained in the context of a larger end-application where the loss is propagated all the way through to the character vector embeddings.

External Link

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Next Steps

- Moving away from RNN/LSTM to other architectures, such as transformers
  - Later lecture
- (Chris Manning) the WMT 2016 final report has 44 instances of RNN whereas the WMT 2018 report has RNN 9 times and Transformer 63 times.

NLP

Deep Learning

743.

Compositionality

and Recursive Neural Networks

Non-compositionality

- BLACK CAT = BLACK + CAT
- BLACK MARKET ≠ BLACK + MARKET

Compositional Semantics

- Given:
  - Vector representations of individual words
- Needed:
  - Representations of units such as phrases, sentences
- Early work:
  - Pointwise sum, multiplication
  - Mitchell and Lapata 2010, Blacoe and Lapata 2012
- Neural methods:
  - Tree RNN, etc.

Recursive Neural Tensor Networks (RNTN)

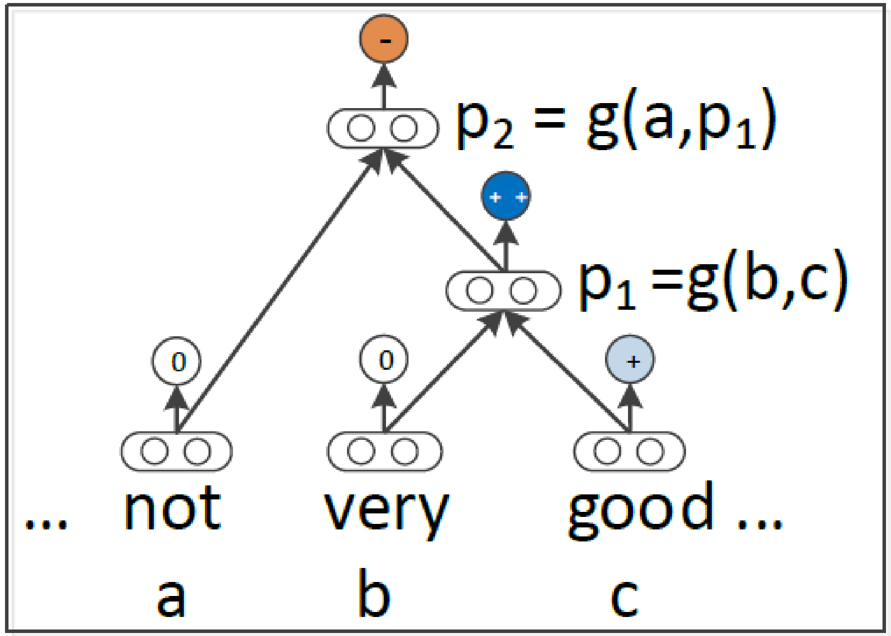
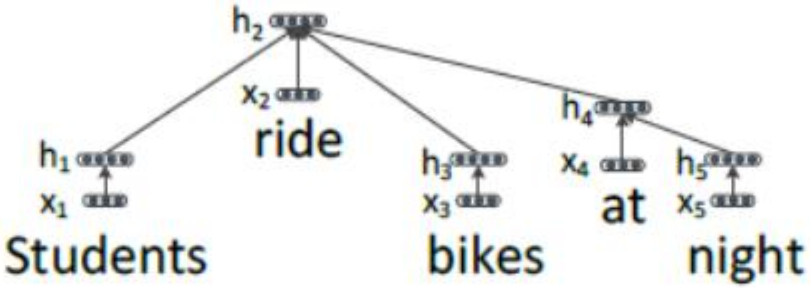


Figure 4: Approach of Recursive Neural Network models for sentiment: Compute parent vectors in a bottom up fashion using a compositionality function g and use node vectors as features for a classifier at that node. This function varies for the different models.



[Socher et al. 2013, 2014]

Recursive Neural Tensor Networks

Socher et al. (2013)
Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

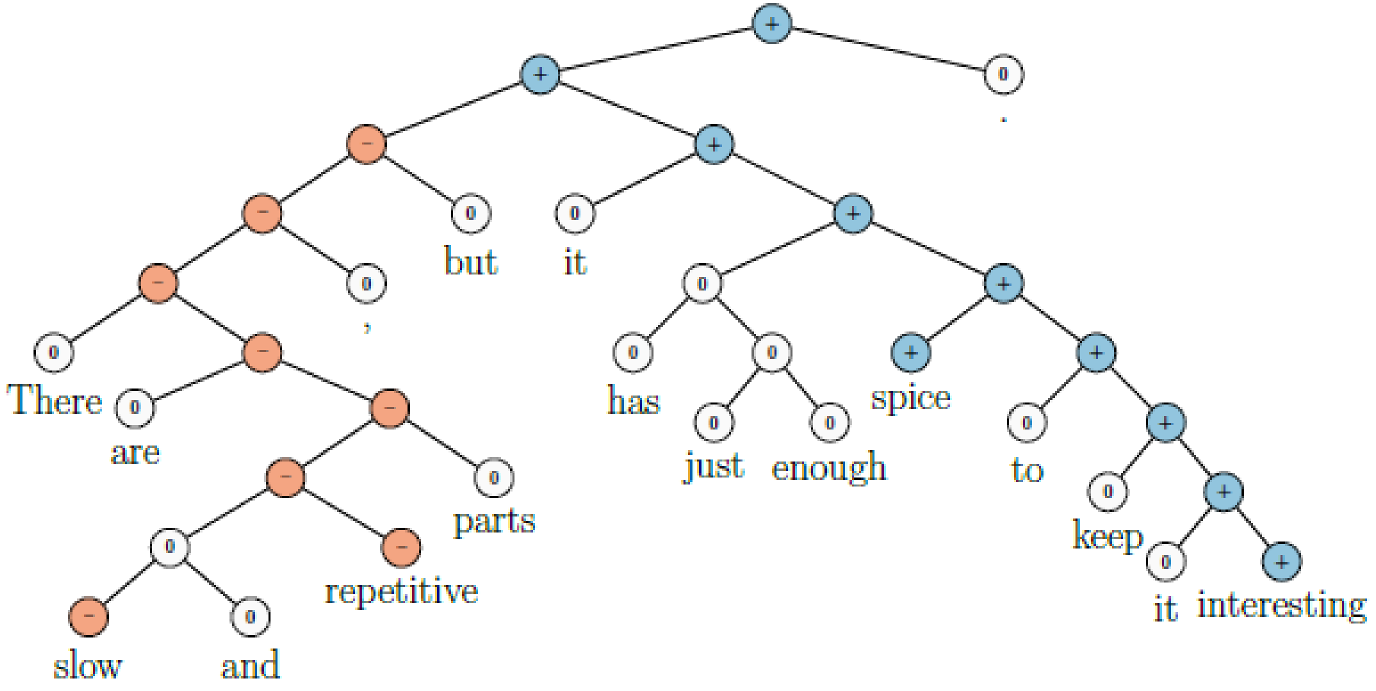
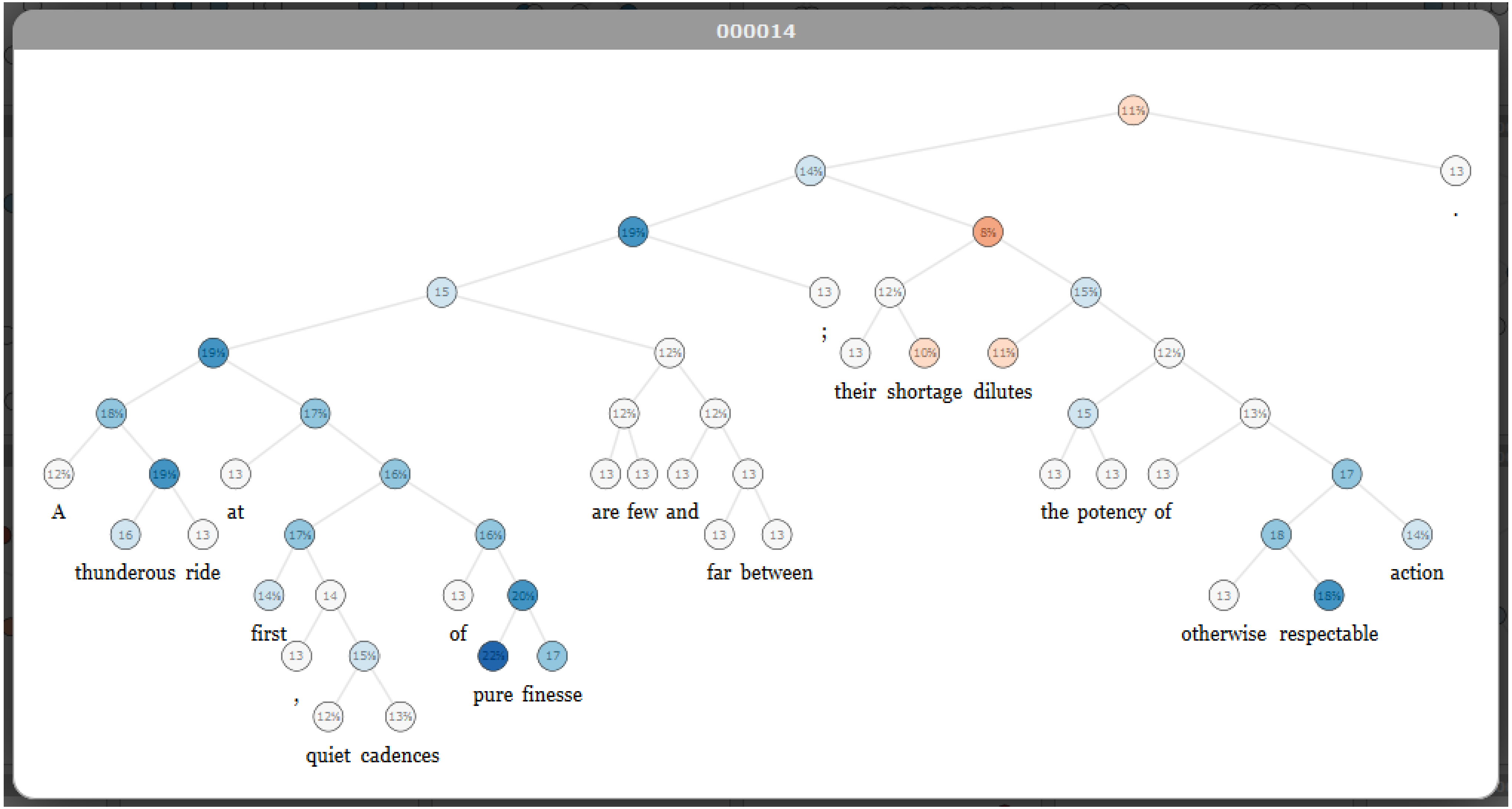
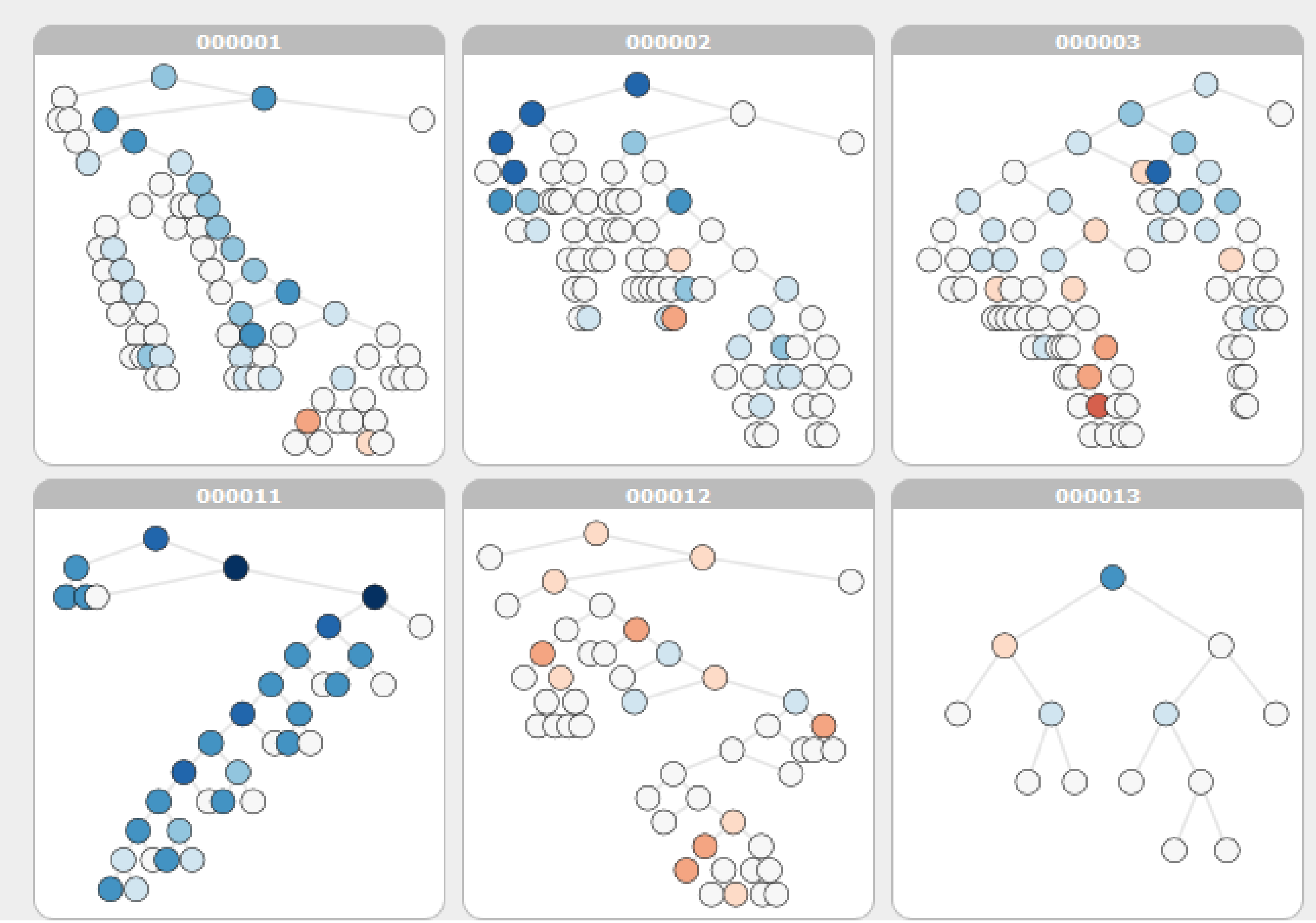


Figure 7: Example of correct prediction for contrastive conjunction X but Y.

Stanford Sentiment Treebank

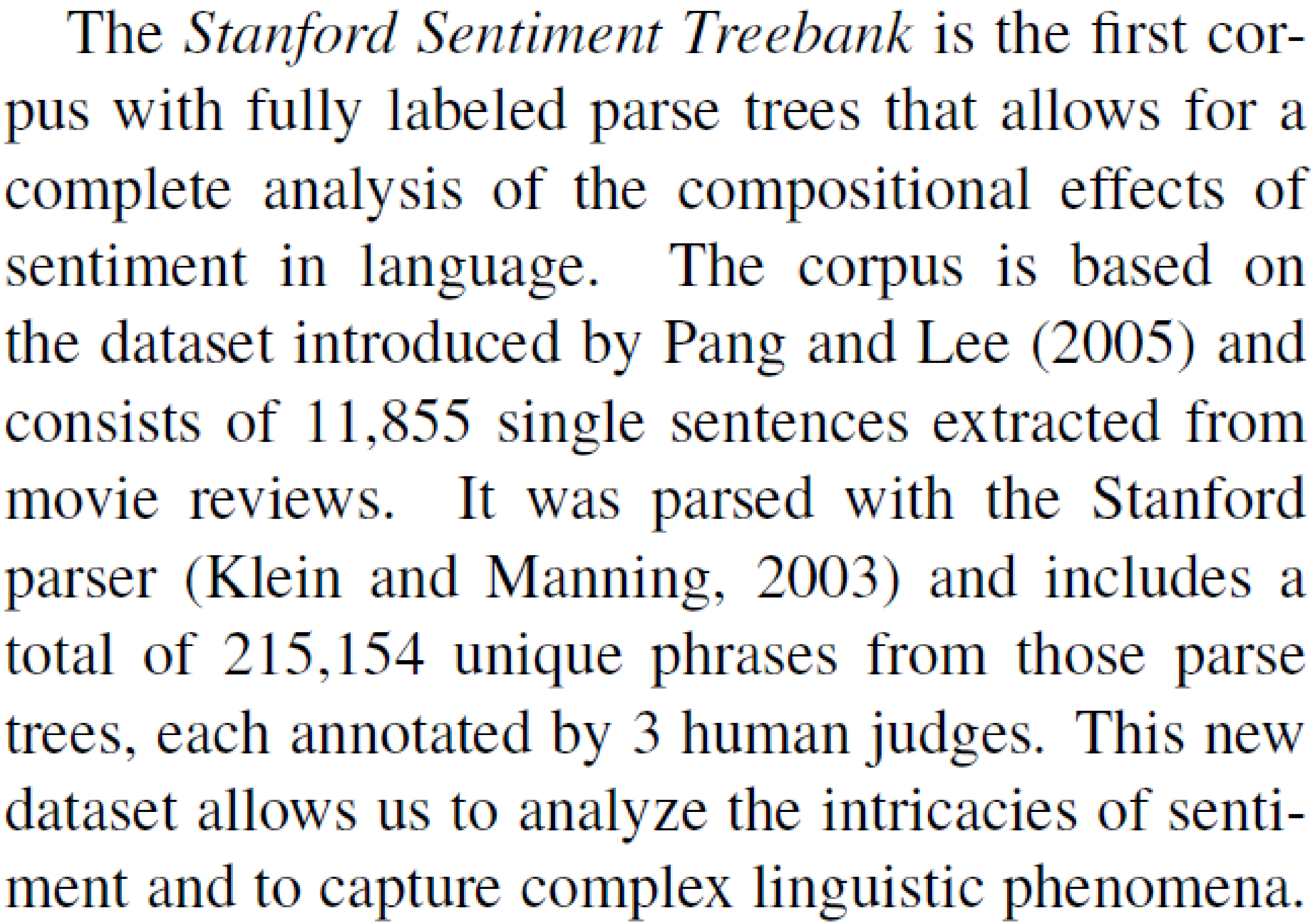

000014
A thunderous ride at first , quiet cadences of pure finesse are few and far between ; their shortage dilutes the potency of otherwise respectable action .
11%
14%
13
19%
8%
15
13
12%
15%
19%
12%
13
10%
11%
12%
18%
17%
12%
12%
15
13%
12%
19%
13
16%
13
13
13
13
13
13
13
17
16
13
17%
16%
13
13
18
14%
14%
14
13
20%
13
18%
13
15%
22%
17
12%
13%

Stanford Sentiment Treebank



https://nlp.stanford.edu/sentiment/treebank.ht
ml

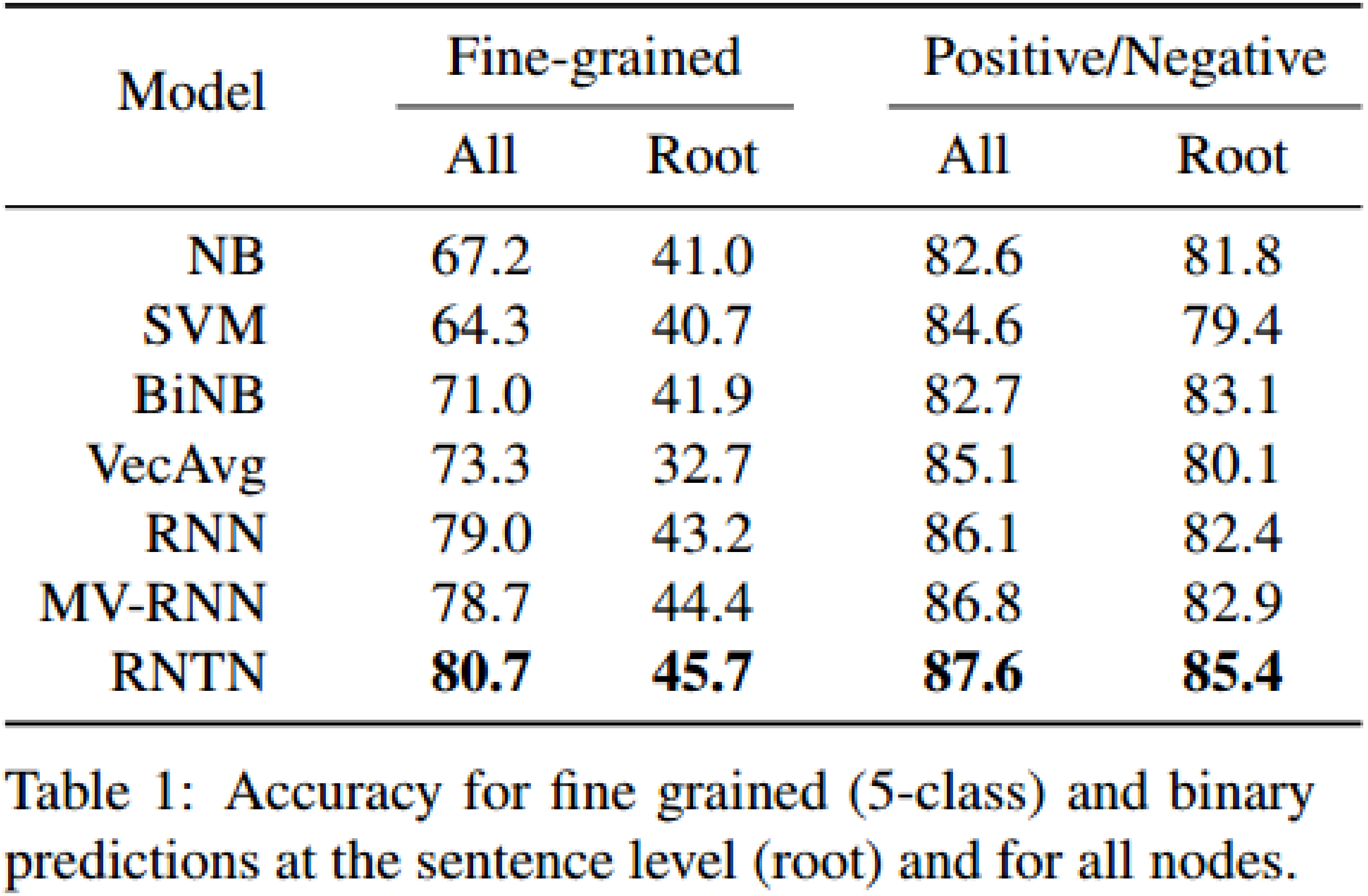
Stanford Sentiment Treebank



The *Stanford Sentiment Treebank* is the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus is based on the dataset introduced by Pang and Lee (2005) and consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser (Klein and Manning, 2003) and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. This new dataset allows us to analyze the intricacies of sentiment and to capture complex linguistic phenomena.

Recursive Neural Networks

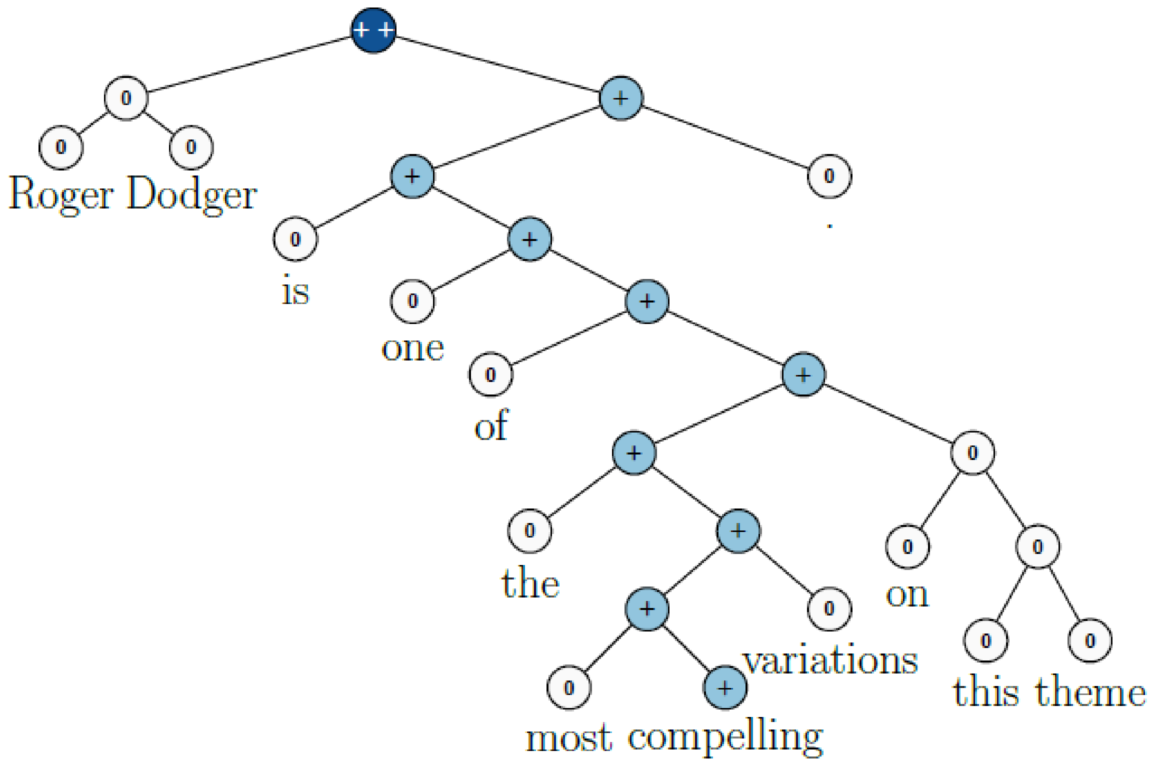
Socher et al. (2013)
Recursive Deep Models for
Semantic Compositionality
Over a Sentiment Treebank

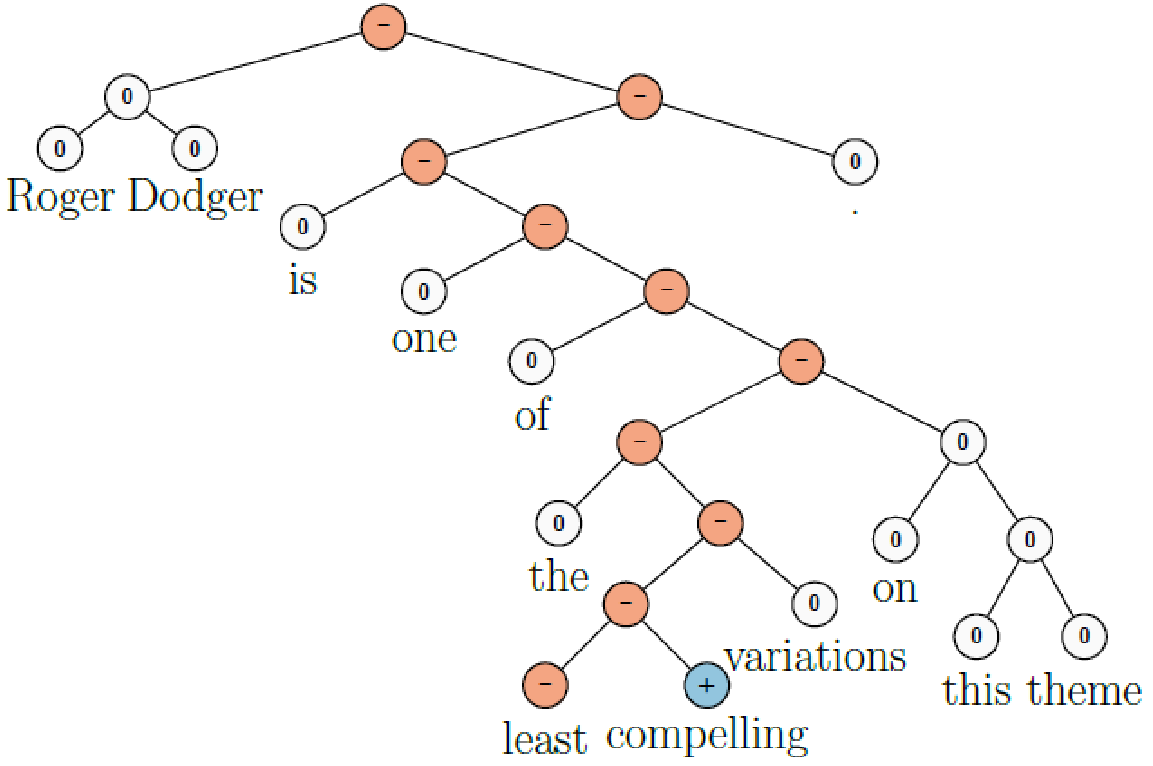


Model	Fine-grained		Positive/Negative	
	All	Root	All	Root
NB	67.2	41.0	82.6	81.8
SVM	64.3	40.7	84.6	79.4
BiNB	71.0	41.9	82.7	83.1
VecAvg	73.3	32.7	85.1	80.1
RNN	79.0	43.2	86.1	82.4
MV-RNN	78.7	44.4	86.8	82.9
RNTN	**80.7**	**45.7**	**87.6**	**85.4**

Table 1: Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

Dealing with Negation





[Socher et al. 2013]

Dealing with Negation

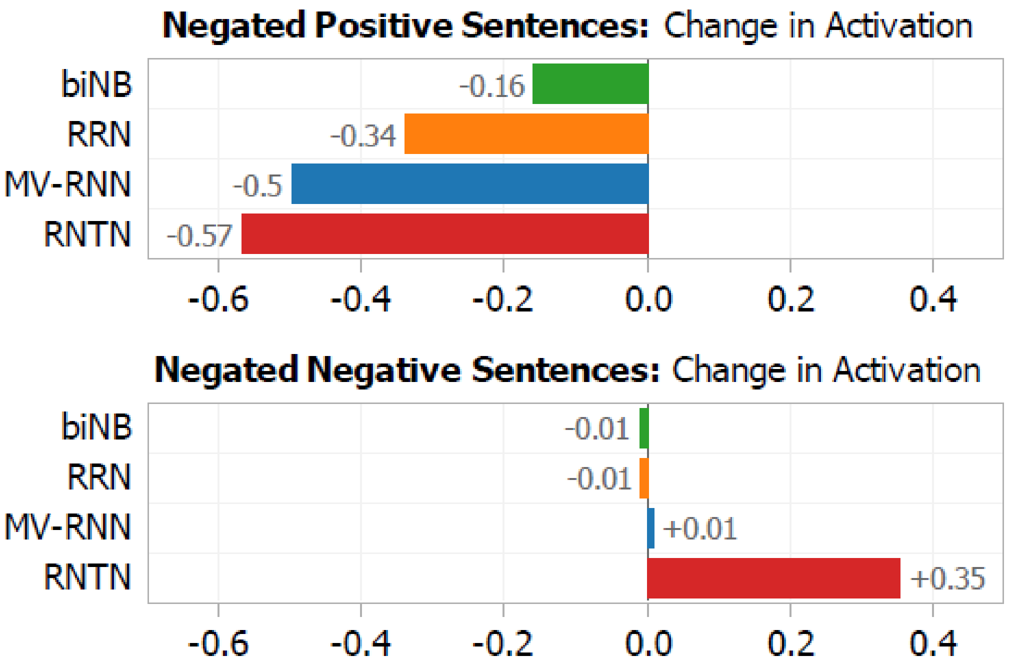
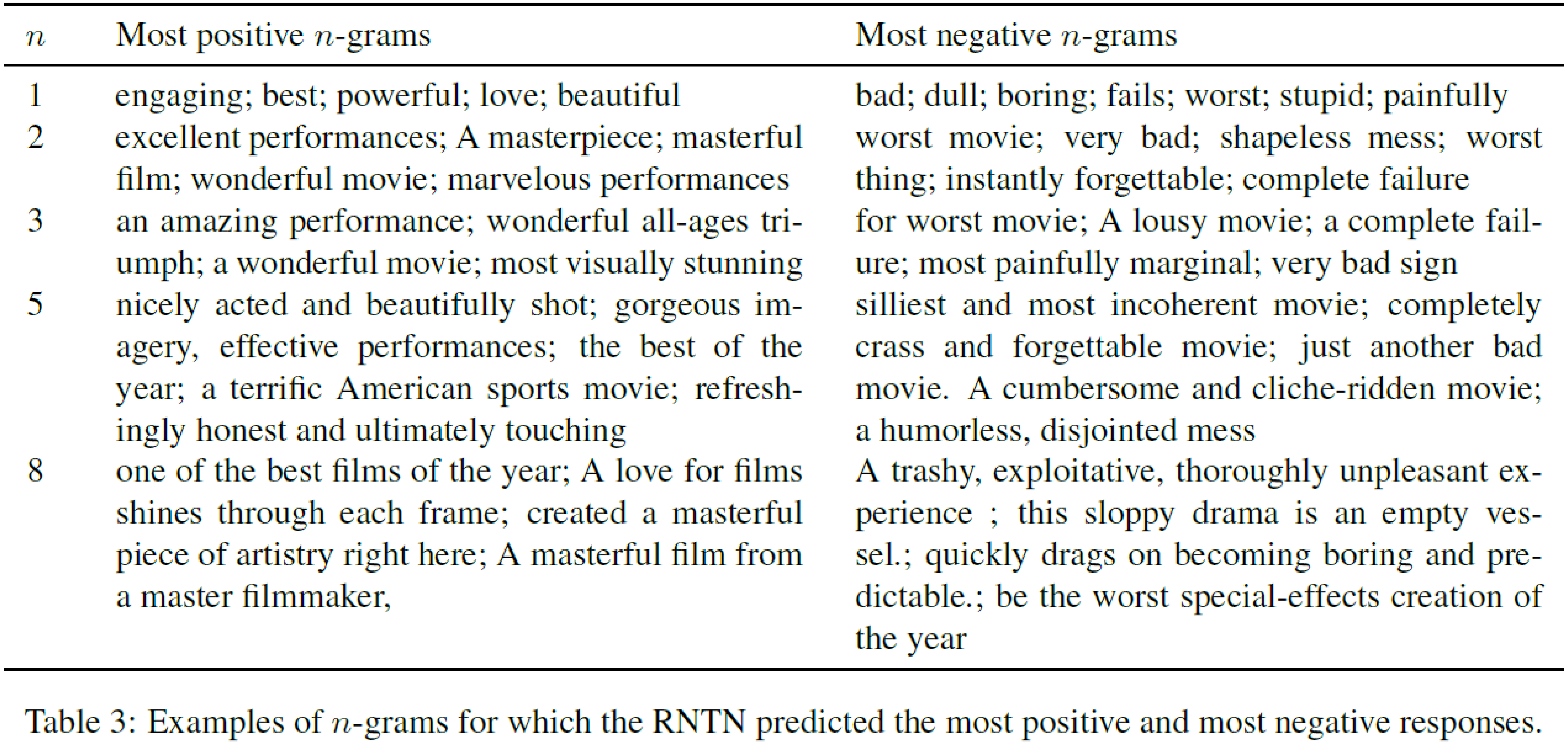


Figure 8: Change in activations for negations. Only the RNTN correctly captures both types. It decreases positive sentiment more when it is negated and learns that negating negative phrases (such as *not terrible*) should increase neutral and positive activations.

[Socher et al. 2013]

Dealing with Negation



n	Most positive n-grams	Most negative n-grams
1	engaging; best; powerful; love; beautiful	bad; dull; boring; fails; worst; stupid; painfully
2	excellent performances; A masterpiece; masterful film; wonderful movie; marvelous performances	worst movie; very bad; shapeless mess; worst thing; instantly forgettable; complete failure
3	an amazing performance; wonderful all-ages triumph; a wonderful movie; most visually stunning	for worst movie; A lousy movie; a complete failure; most painfully marginal; very bad sign
5	nicely acted and beautifully shot; gorgeous imagery, effective performances; the best of the year; a terrific American sports movie; refreshingly honest and ultimately touching	silliest and most incoherent movie; completely crass and forgettable movie; just another bad movie. A cumbersome and cliche-ridden movie; a humorless, disjointed mess
8	one of the best films of the year; A love for films shines through each frame; created a masterful piece of artistry right here; A masterful film from a master filmmaker,	A trashy, exploitative, thoroughly unpleasant experience ; this sloppy drama is an empty vessel.; quickly drags on becoming boring and predictable.; be the worst special-effects creation of the year

Table 3: Examples of n-grams for which the RNTN predicted the most positive and most negative responses.

[Socher et al. 2013]

NLP

Deep Learning

744.

Convolutional Neural Networks

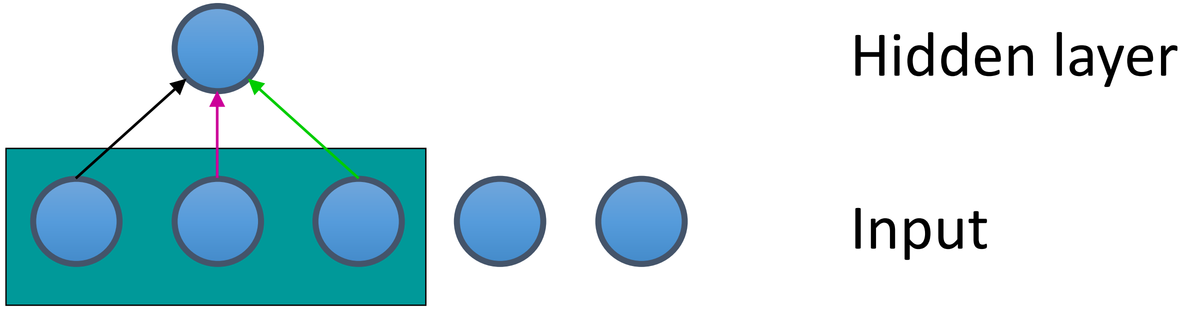
Cats’ Brains and CNNs

(Don’t do this to your cats. It can traumatize them.)

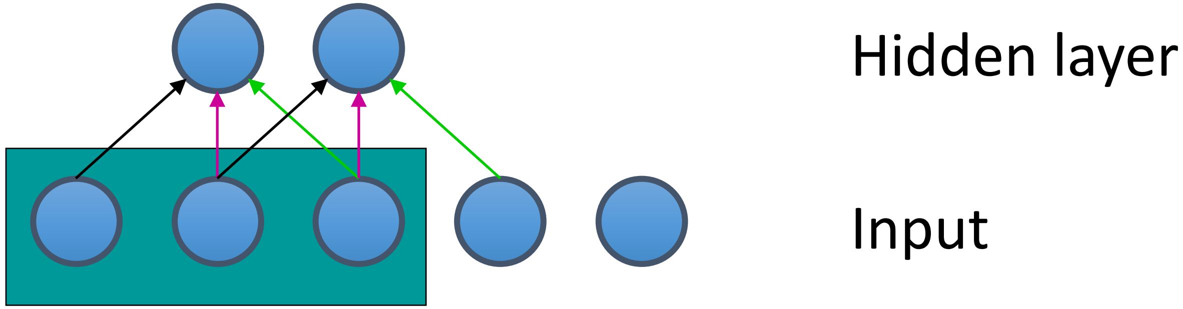
Short history of CNN

- Introduced by Hubel and Wiesel in the 50s and 60s
  - Neurons in the mammalian visual cortex respond to specific small patterns in the visual field
- Fukushima 1980
  - Propagating local features to higher layers
- LeCun 1989-
  - Training for handwriting recognition (MNIST) etc.
- Collobert 2011
  - use in NLP for semantic role labeling
- Krizhevsky et al. 2012
  - Object detection in ImageNet

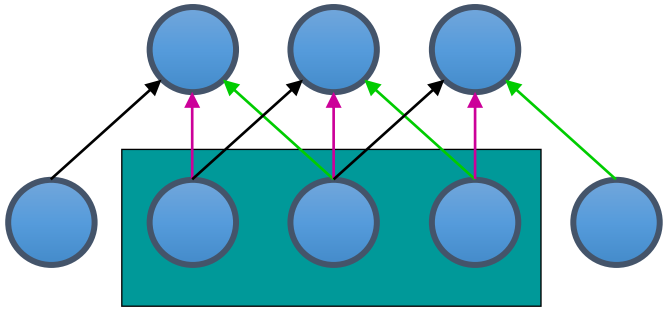
Basic Idea of CNNs



Basic Idea of CNNs



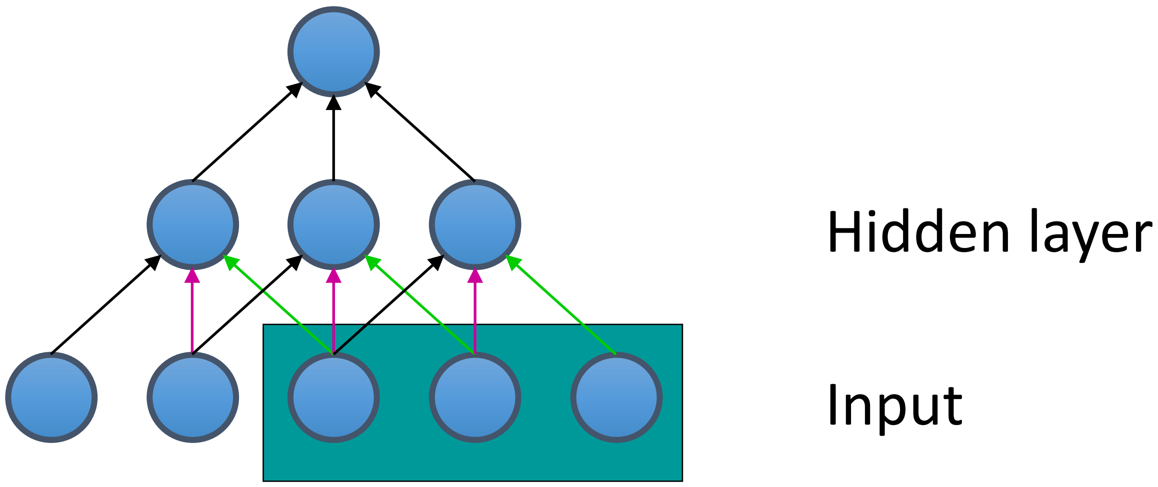
Basic Idea of CNNs



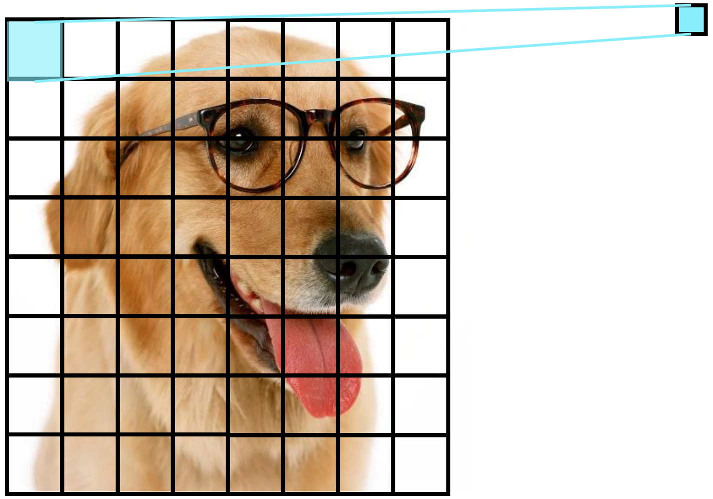
Hidden layer

Input

Basic Idea of CNNs

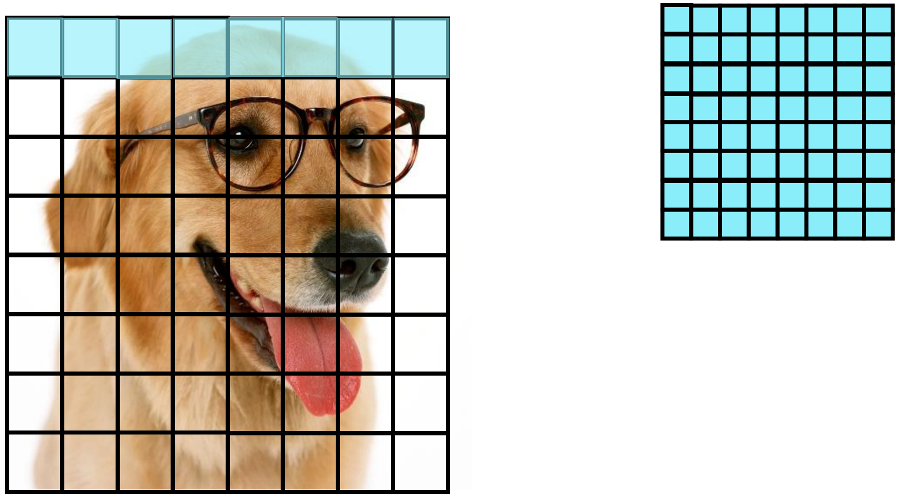


CNN for Image Classification



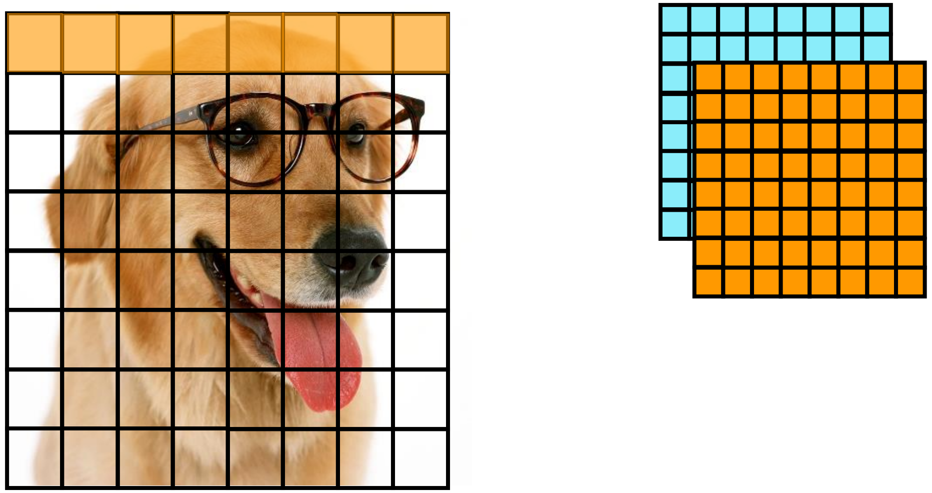
Convolutional Layer

CNN for Image Classification



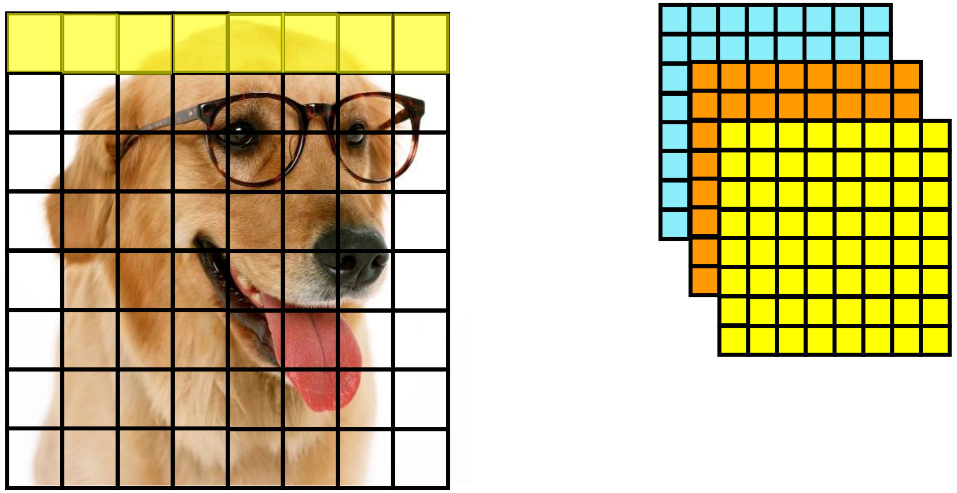
Convolutional Layer

CNN for Image Classification



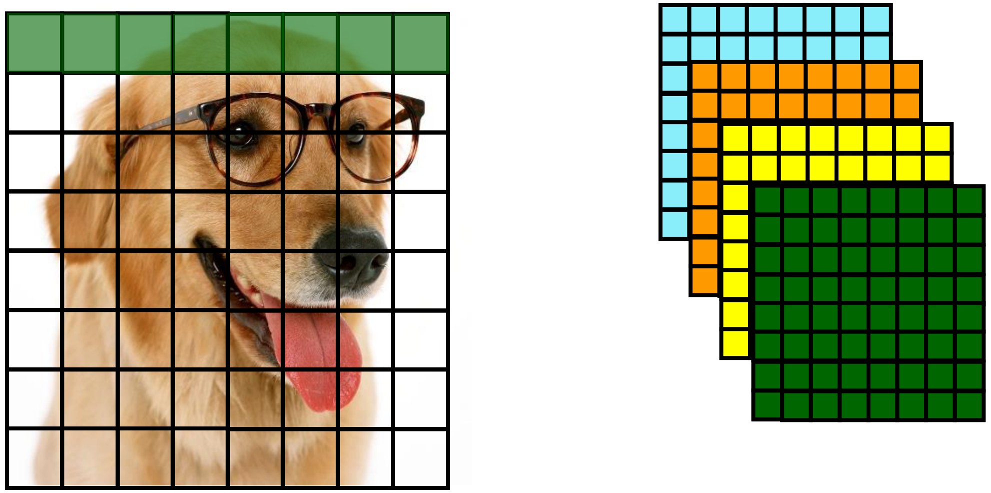
Convolutional Layer

CNN for Image Classification



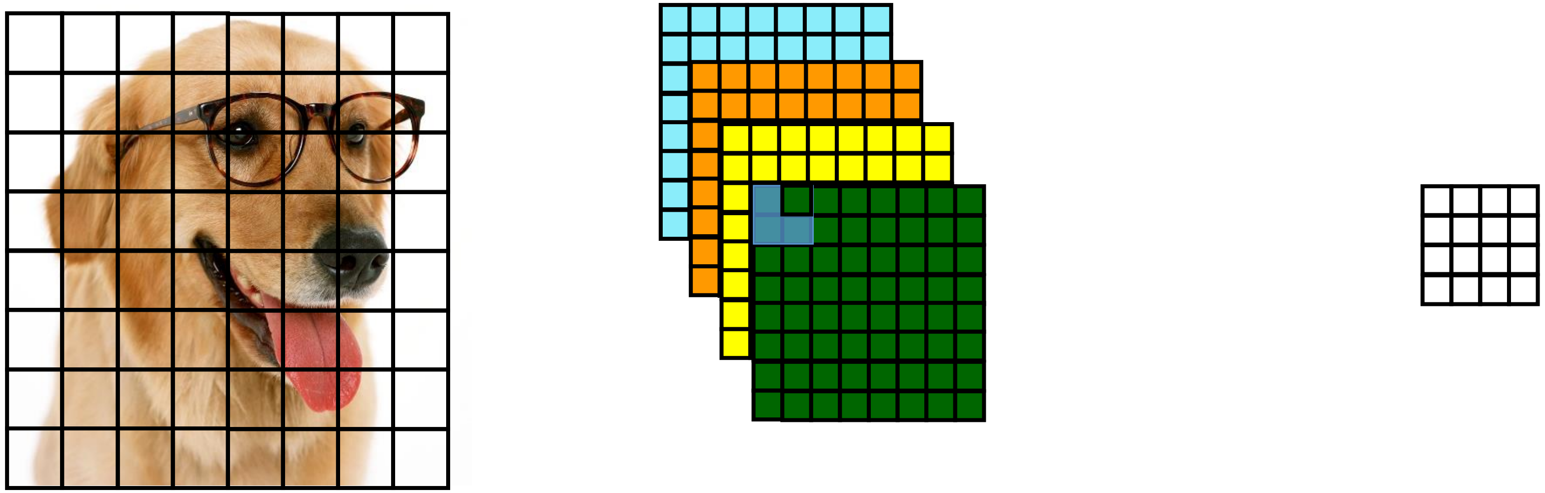
Convolutional Layer

CNN for Image Classification



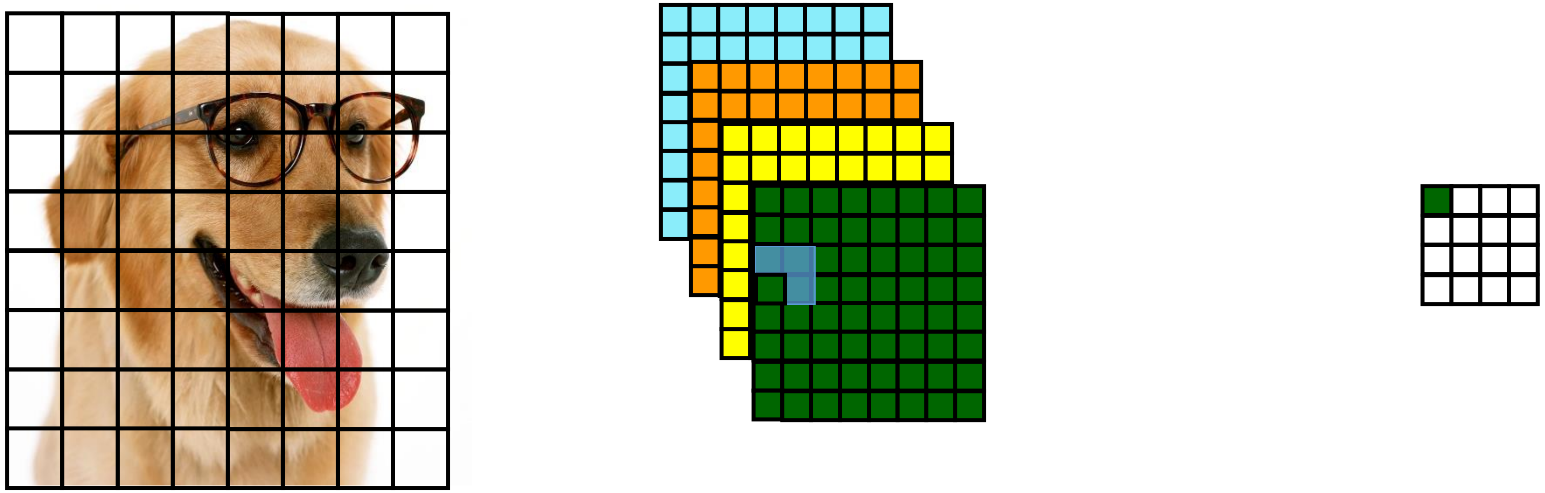
Convolutional Layer

CNN for Image Classification



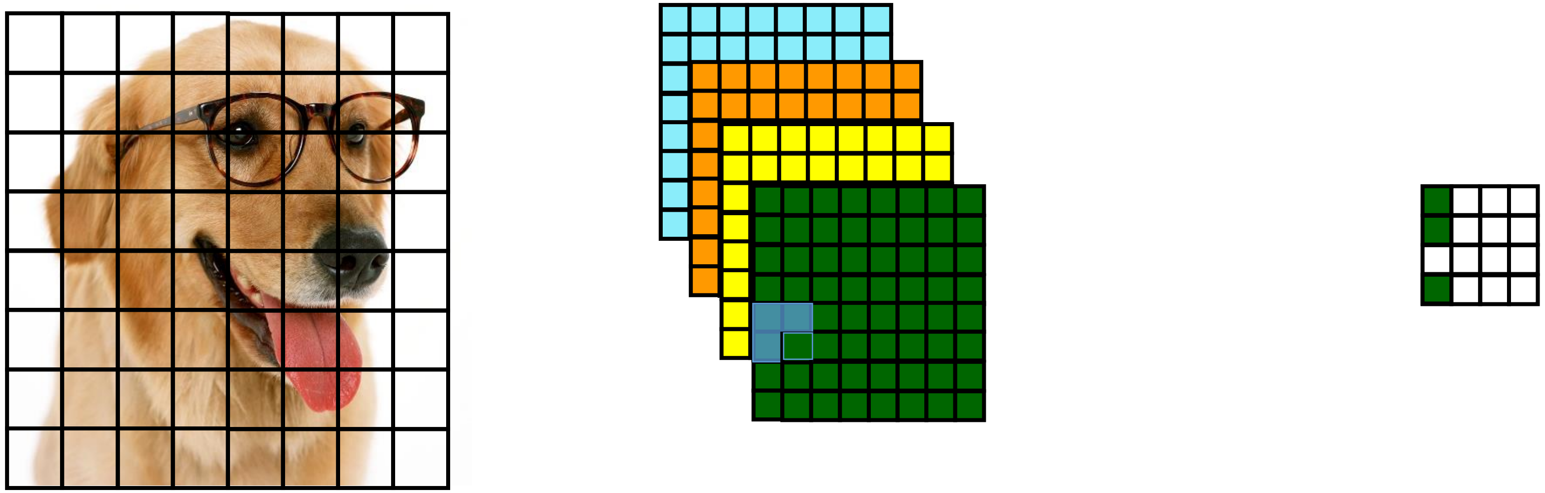
Max Pooling

CNN for Image Classification



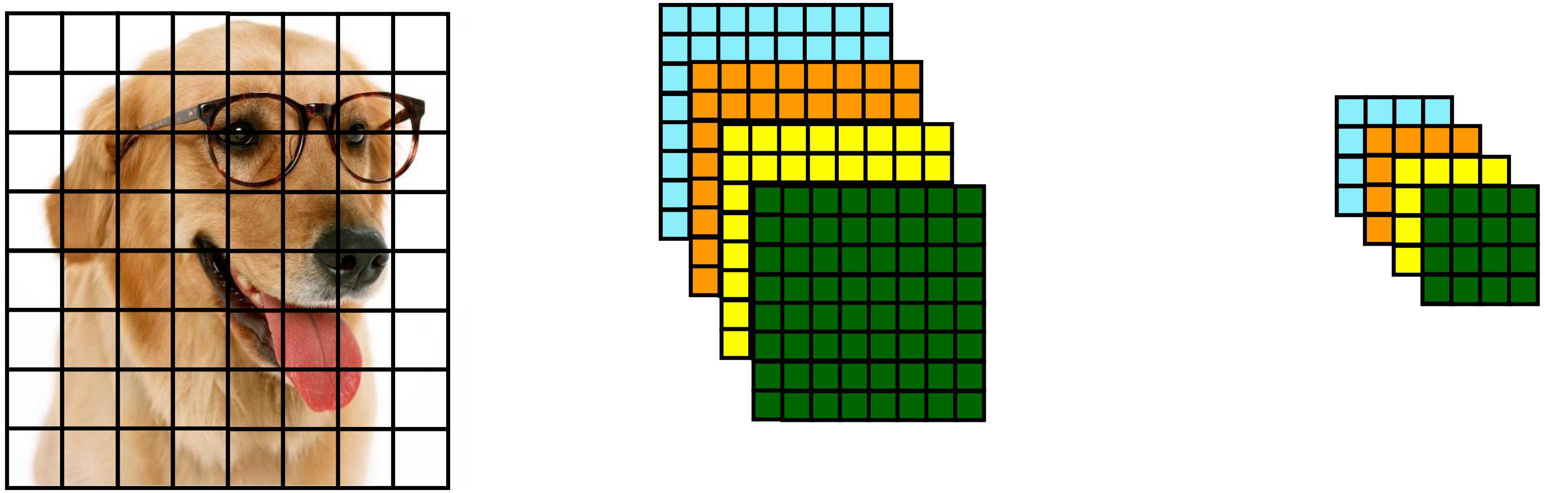
Max Pooling

CNN for Image Classification



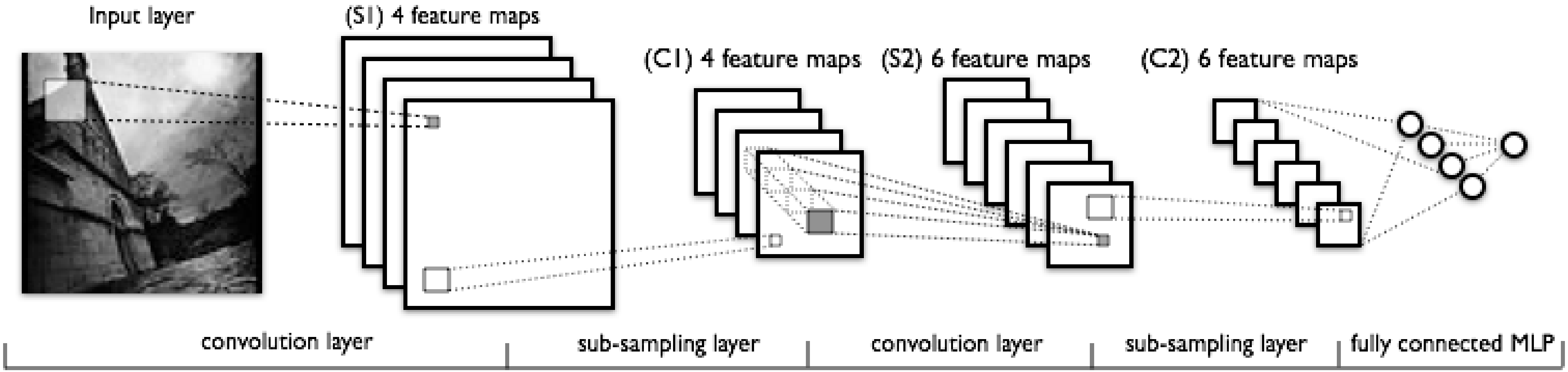
Max Pooling

CNN for Image Classification



Max Pooling

CNN for Image Classification



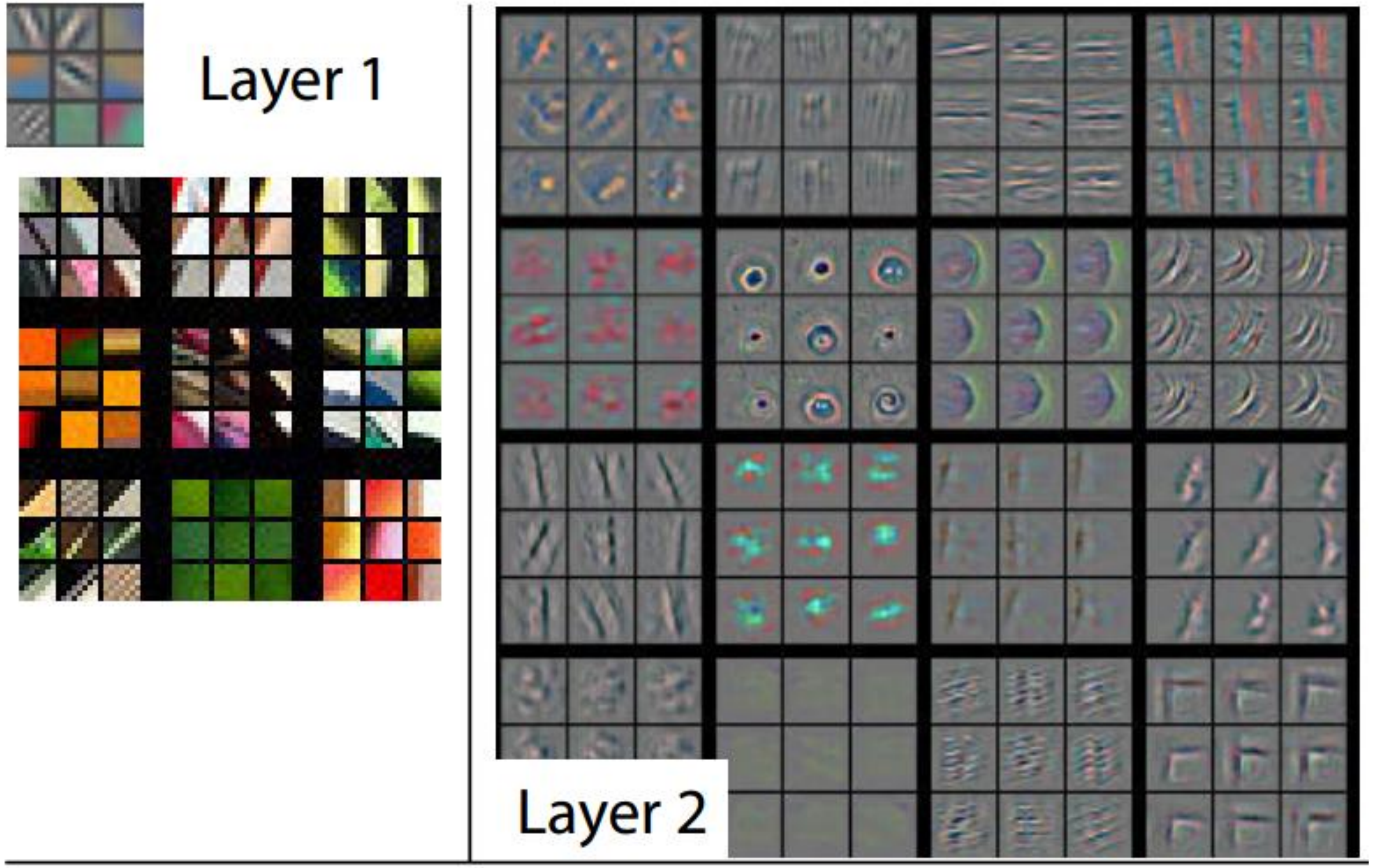
http://deeplearning.net/tutorial/lenet.html

[LeCun]

CNN for Image Classification

- How good are CNNs?
- “We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of **15.3%**, compared to **26.2%** achieved by the second-best entry.”
    - Krizhevsky et al., 2012: ImageNet Classification with Deep Convolutional Neural Networks
    - Competition to classify photos from ImageNet, http://www.image-net.org/

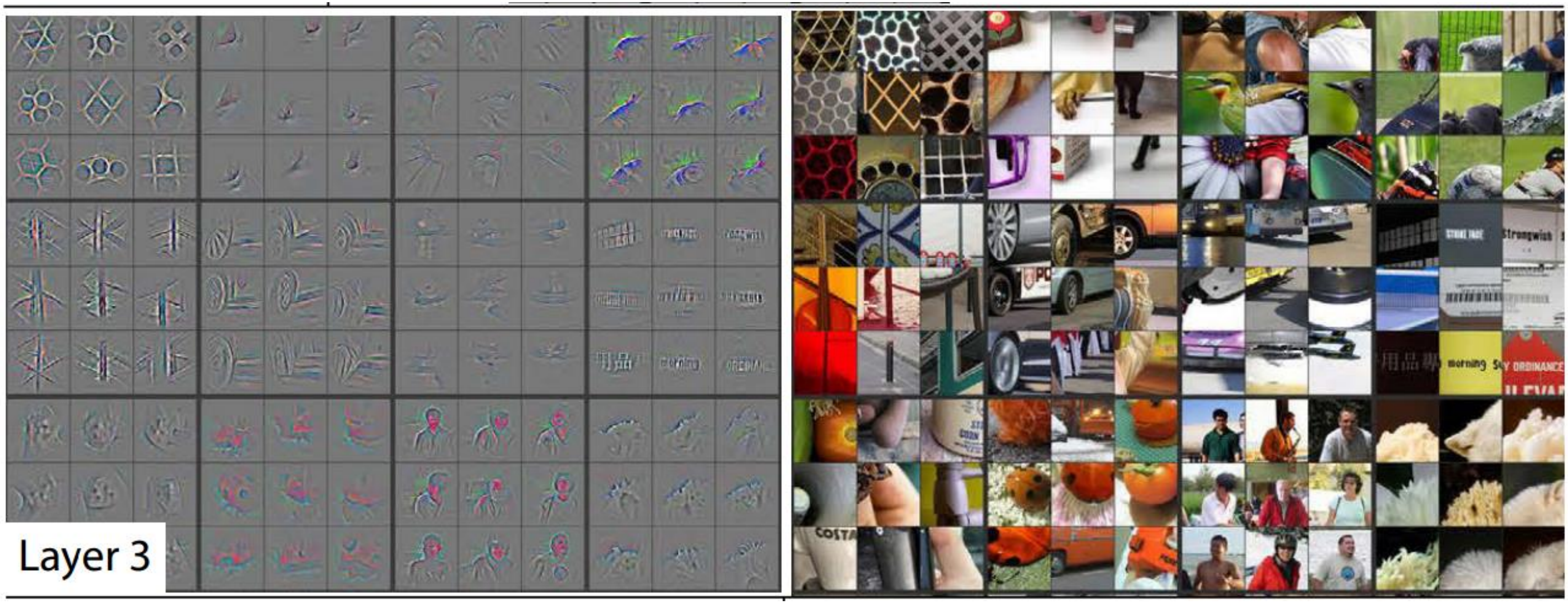
CNN for Image Classification



Zeiler and Fergus 2014

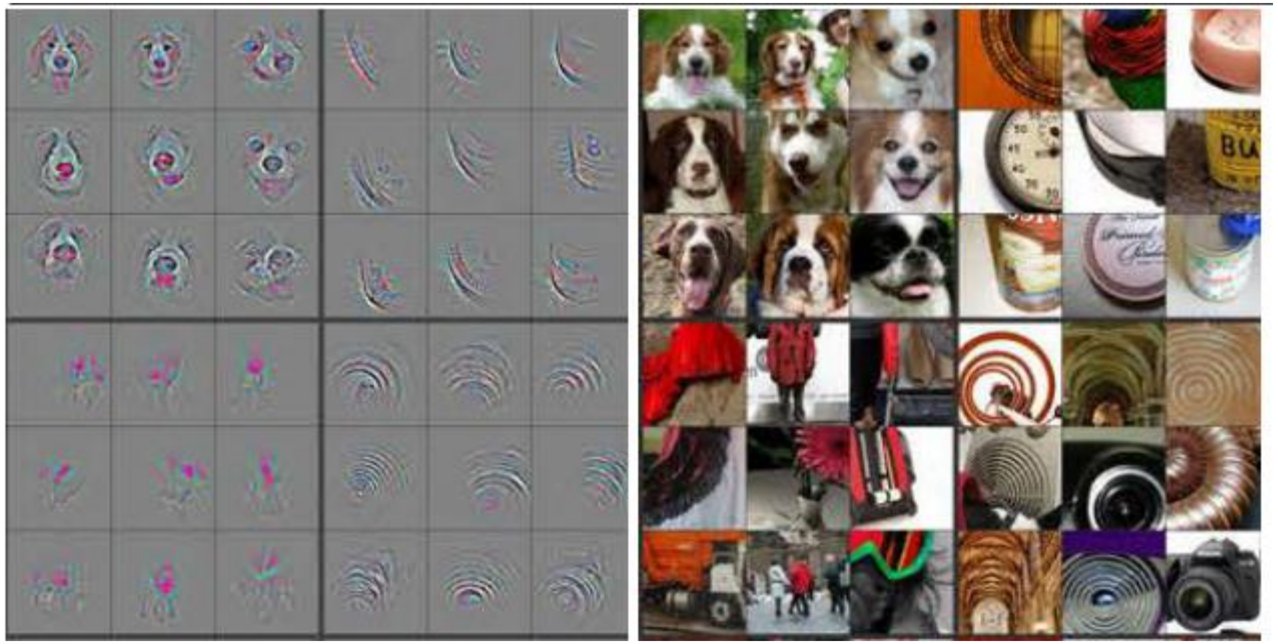
Learned Filters

CNN for Image Classification



Zeiler and Fergus 2014

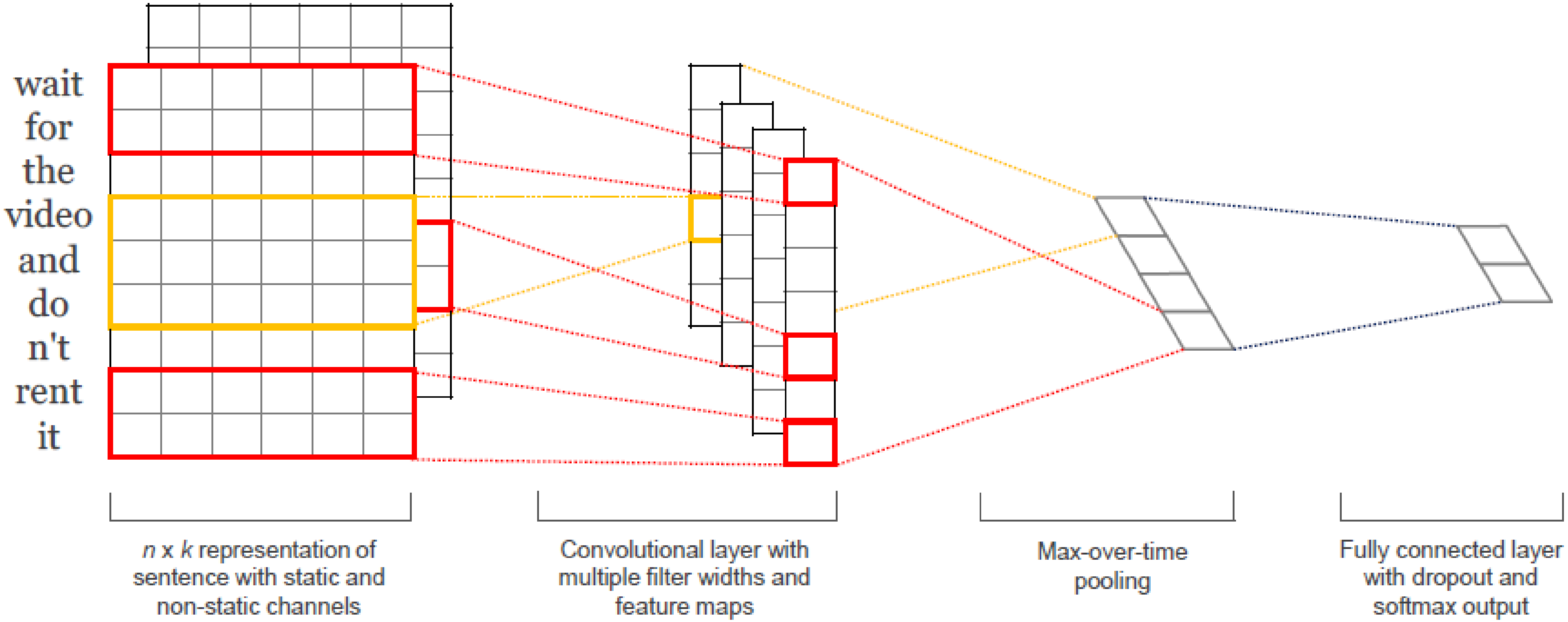
CNN for Image Classification



Layer 4

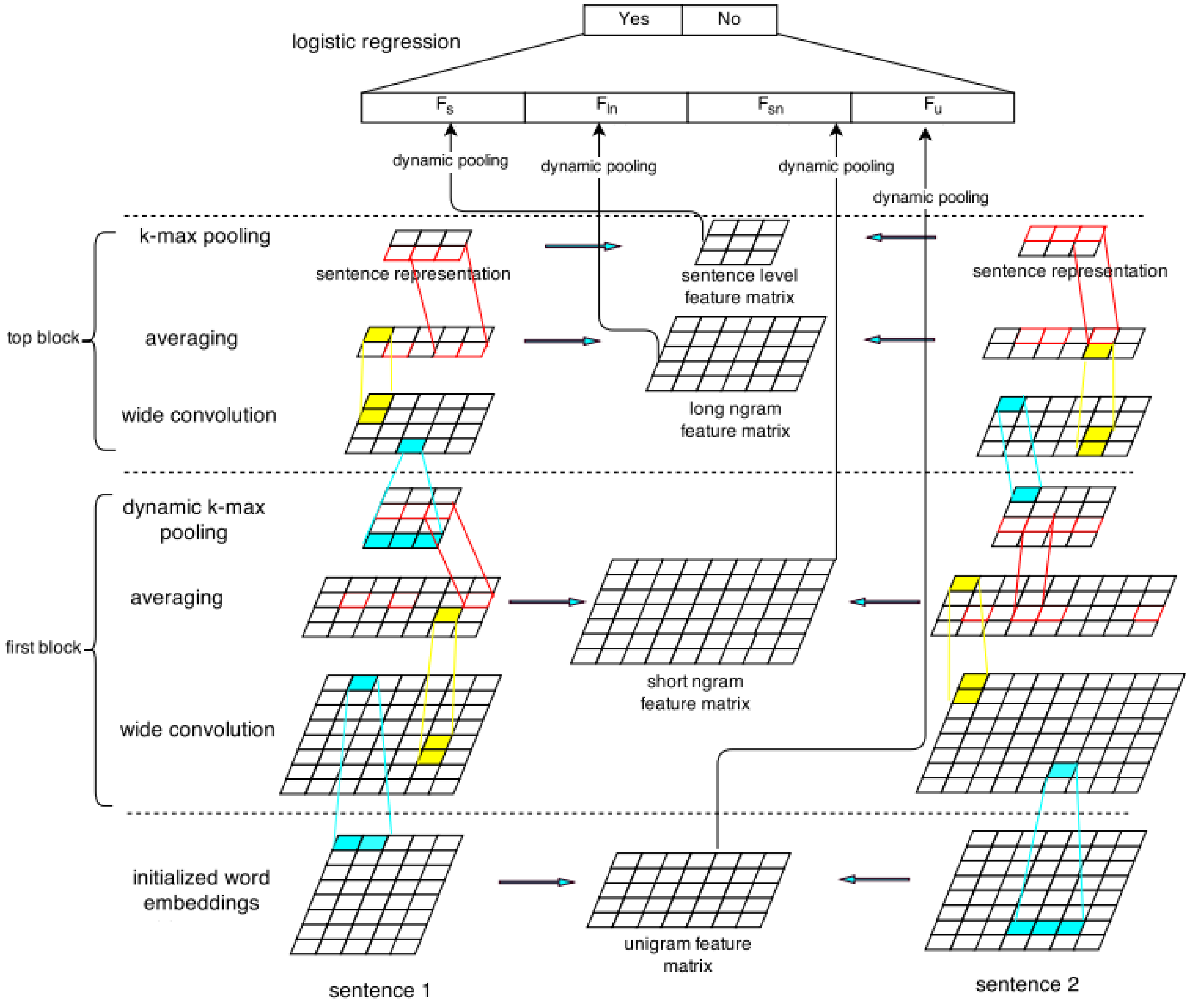
Zeiler and Fergus 2014

Sentence Classification


wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

[Kim 2014]

Paraphrase Detection

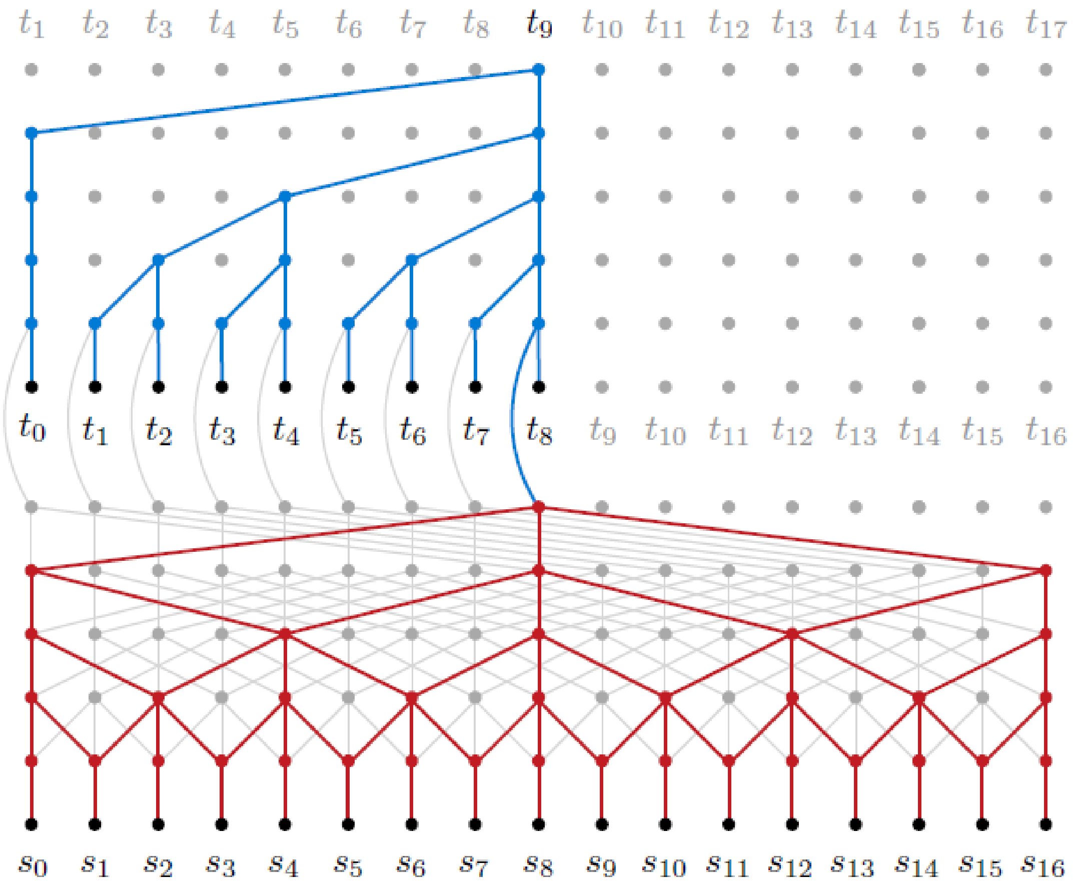


[Yin & Schütze 2015]

Adjustments to CNN

- Striding
  - Skip some of the possible input substrings
  - E.g., start at every other word
- Pooling
  - Reduction, e.g., average, max
  - k-max: did this feature appear at least k times
- Stacking
  - Same idea as with RNN and LSTM
- Dilating
  - (see next slide)

Dilated Convolution



t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17}
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16}
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15} s_{16}

Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.

[Kalchbrenner et al. 2016]

Using CNNs for NLP

- Convolutional Neural Network for Paraphrase Identification (Yin & Schütze 2015)
- Summarization-based Video Caption via Deep Neural Networks (Li et al. 2015)
- Question Answering over Freebase with Multi-Column Convolutional Neural Networks (Dong et al. 2015)
- Convolutional Neural Network Architectures for Matching Natural Language Sentences (Hu et al. 2015)
- Learning Semantic Representations Using Convolutional Neural Networks for Web Search (Sheng et al. 2015)
- Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts (dos Santos & Gatti 2014)
- Relation Extraction: Perspective from Convolutional Neural Networks (Nguyen & Grishman 2015)
- Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation (Sun et al. 2015)
- Modeling Interestingness with Deep Neural Networks (Gao 2015)

NLP