

CS11-747 Neural Networks for NLP

# Convolutional Networks for Text

Graham Neubig



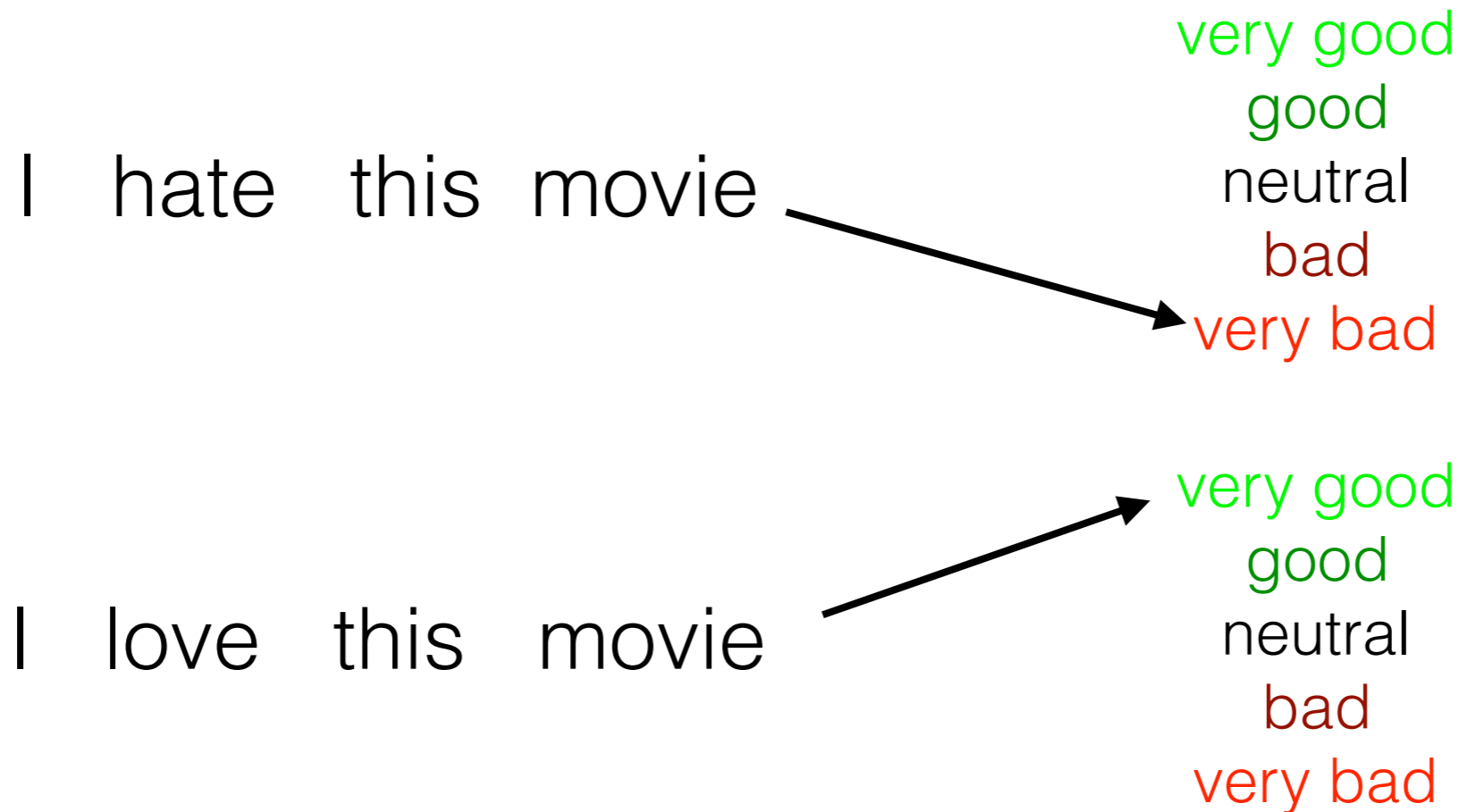
**Carnegie Mellon University**

**Language Technologies Institute**

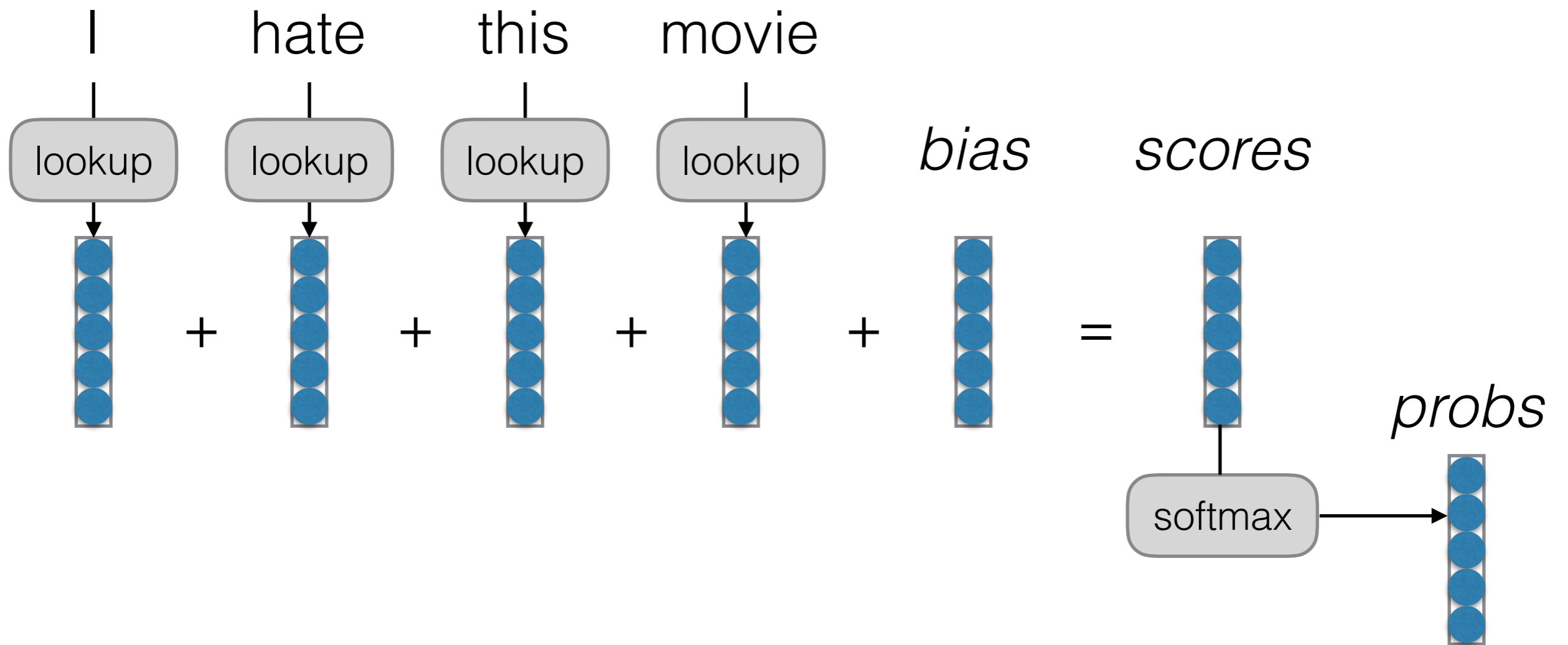
Site

<https://phontron.com/class/nn4nlp2017/>

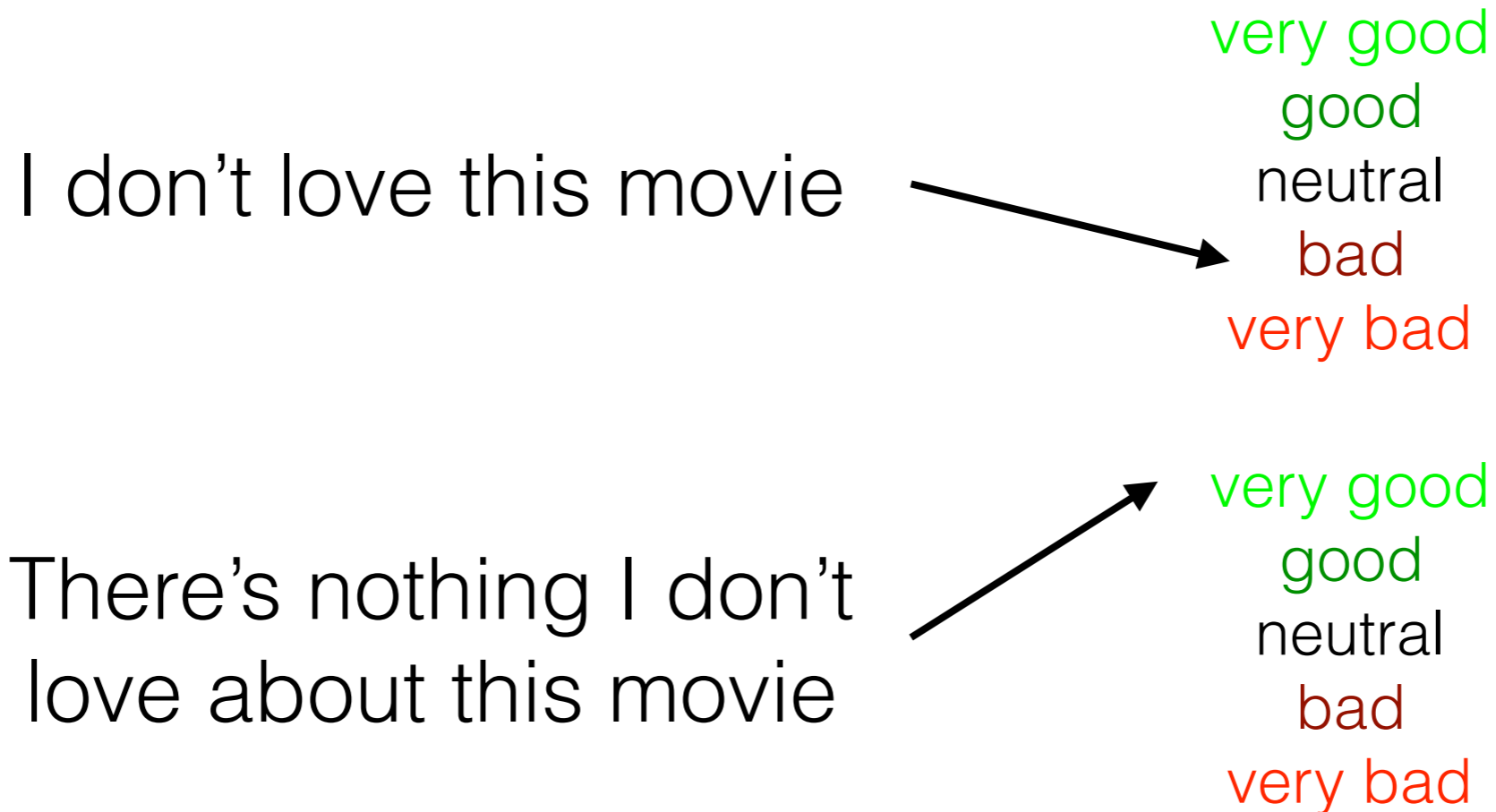
# An Example Prediction Problem: Sentence Classification



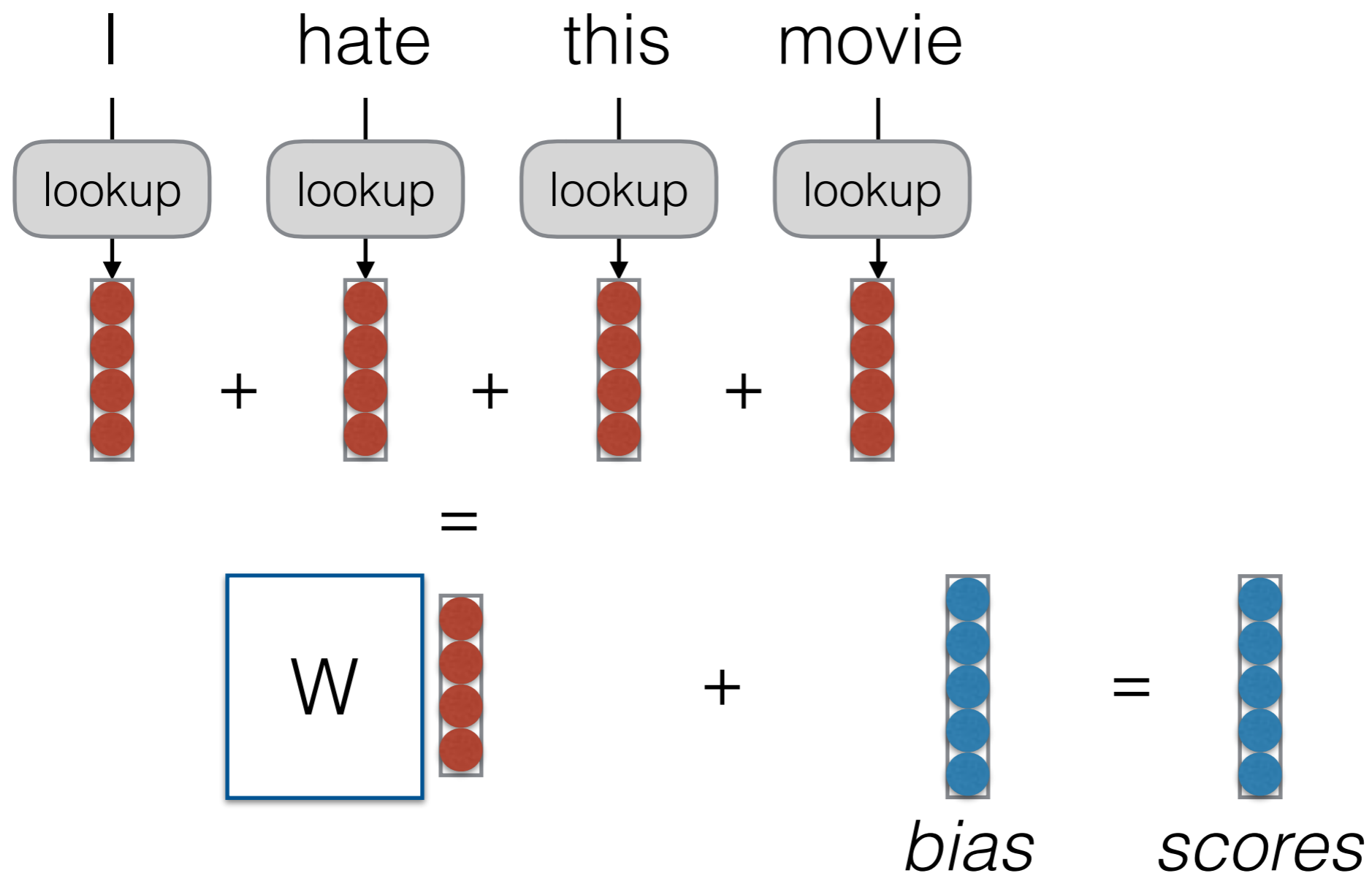
# A First Try: Bag of Words (BOW)



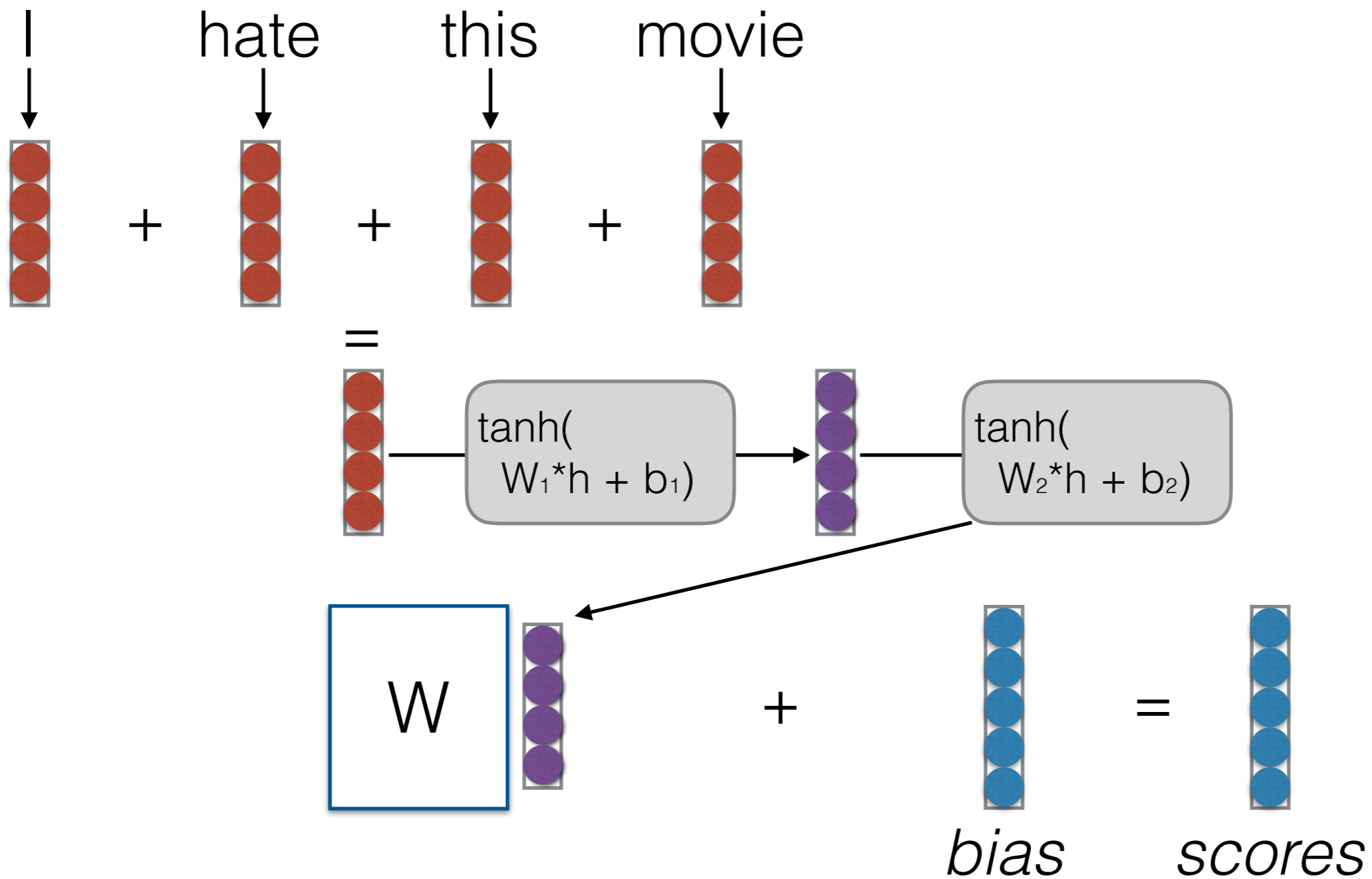
# Build It, Break It



# Continuous Bag of Words (CBOW)



# Deep CBOW



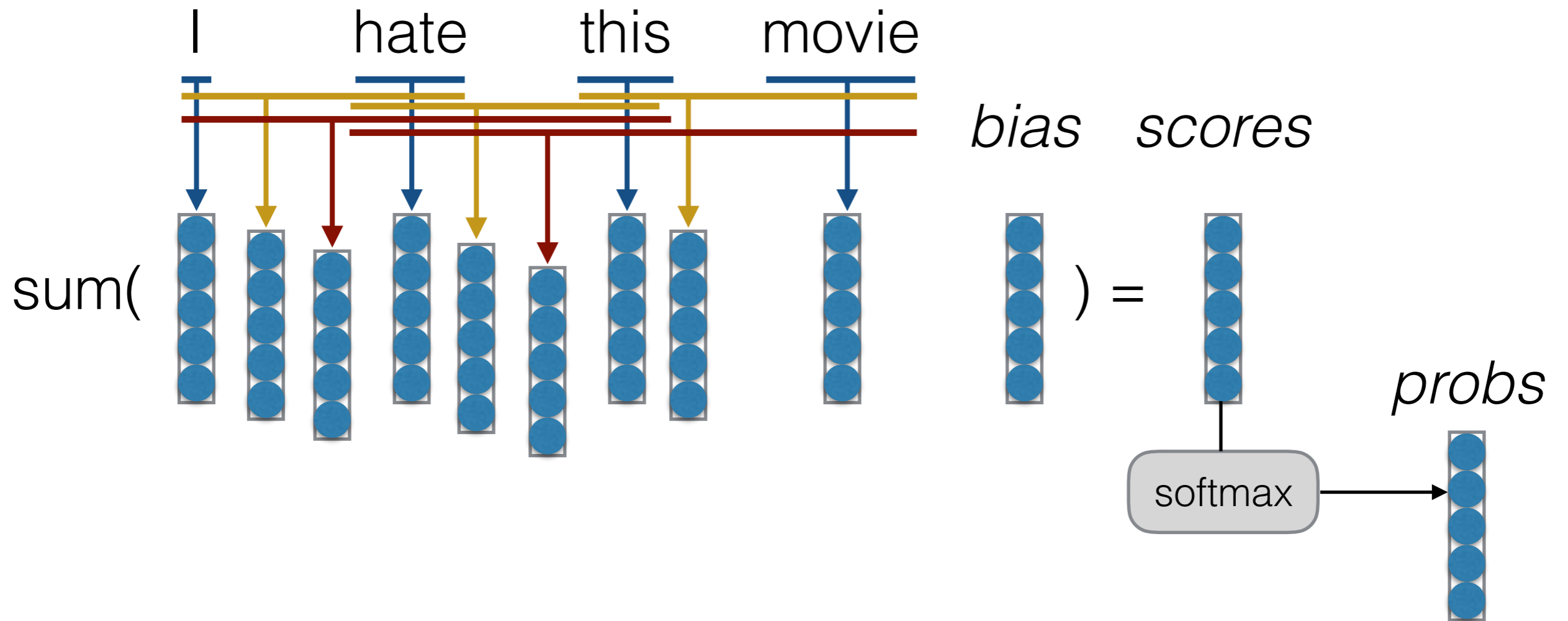
# What do Our Vectors Represent?

- We can learn feature combinations (a node in the second layer might be “feature 1 AND feature 5 are active”)
- e.g. capture things such as “not” AND “hate”
- BUT! Cannot handle “not hate”

# Handling Combinations



# Bag of n-grams



# Why Bag of n-grams?

- Allow us to capture combination features in a simple way “don’t love”, “not the best”
- Works pretty well



François Chollet @fchollet · 2 Nov 2016

We are releasing an open dataset for theorem proving, HolStep: [openreview.net/forum?id=ryuxY...](https://openreview.net/forum?id=ryuxY...) - can you beat our 83% accuracy baseline?

1 51 123



Hal Daumé III @haldaume3 · 2 Nov 2016

.@fchollet sure, I'll play. 85%, took me about an hour. (totally possible I did something wrong in preprocessing though!)

```
cat train/* | ./holstep2w.pl | shuffle | wu --binary --loss_function logistic --ngram 5 -k -c --passes
-b33 -f model.ngram6 --holdout off
0.368470 0.247070 16384 16384,0 -1.0000 -1.0000 2888
0.239238 0.209106 32768 32768,0 -1.0000 -1.0000 4791
0.210735 0.182281 65536 65536,0 1.0000 1.0000 2082
0.184732 0.158798 131072 131072,0 1.0000 1.0000 4023
0.168405 0.149028 262144 262144,0 -1.0000 -1.0000 9368
0.159111 0.137817 524288 524288,0 1.0000 1.0000 1881
0.148981 0.126872 1048576 1048576,0 1.0000 1.0000 4494
0.127713 0.116435 2097152 2097152,0 1.0000 1.0000 1928
0.101621 0.091249 4194304 4194304,0 -1.0000 -1.0000 1199
0.088621 0.088621 8388608 8388608,0 1.0000 -1.0000 1323

Finished run
number of examples per pass = 2018046
cases used = 8
weighted example sum = 10068280.000000
weighted label sum = 0.000000
average loss = 0.982794
total constant = 0.000000
best constant's loss = 0.693147
total feature number = 29140509425

cat best/* | ./holstep2w.pl | wu --binary -i model.ngram6 -t
average loss = 0.146743

cat holstep2w.pl
#!/usr/bin/perl -w
use strict;

my $conJName = ''; my $conJText = ''; my $conJTok = '';
my $depName = ''; my $depText = ''; my $depTok = '';
while (0) {
  chomp;
  if (0) {
    s/"/"/g; $conJName = $_;
    $_ = 0; die if not /"/; s/"/"/g; $conJText = tokenize($_);
    $_ = 0; die if not /"/; s/"/"/g; $conJTok = $_;
  } elsif (0) {
    s/"/"/g; $depName = $_;
    $_ = 0; die if not /"/; s/"/"/g; $depText = tokenize($_);
    $_ = 0; die if not /"/; s/"/"/g; $depTok = $_;
  } elsif (0) {
    my $stepLabel = (0) ? 1 : -1;
    s/"/"/g; my $stepText = tokenize($_);
    print "stepLabel: ", $stepLabel, " conJName: ", $conJName, " conJText: ", $conJText, " conJTok: ", $conJTok, " depName: ", $depName, " depText: ", $depText, " depTok: ", $depTok, "\n";
  } size { die 1; }
}

sub wu {
  my ($t) = @_;
  chomp $t; $t =~ s/"/"/g; $t =~ s/"/"/g;
  return $t;
}

sub tokenize {
  my ($t) = @_;
  $t =~ s/"/"/g; $t =~ s/"/"/g;
  return $t;
}
```

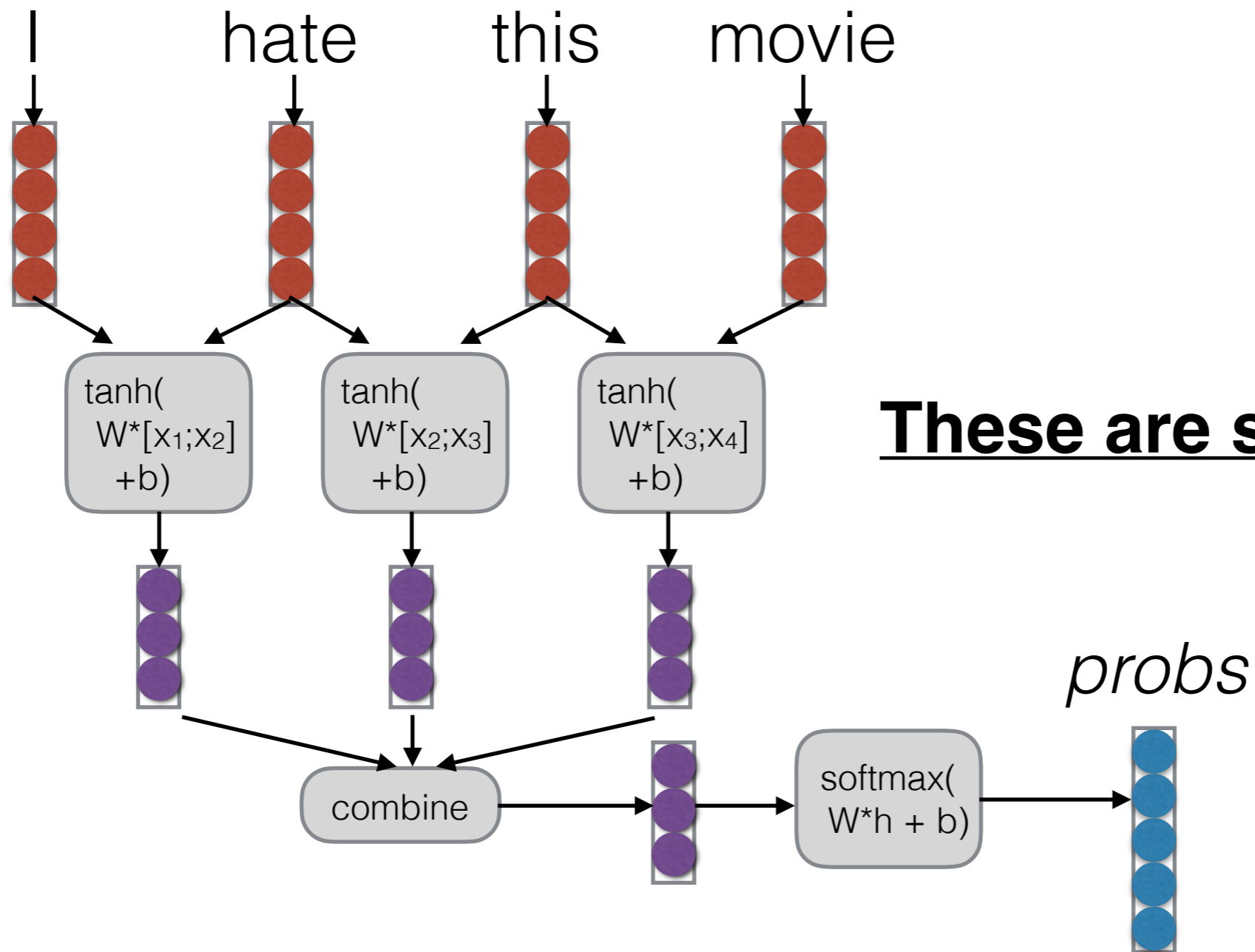
# What Problems w/ Bag of n-grams?

- Same as before: parameter explosion
- No sharing between similar words/n-grams

# Time Delay/ Convolutional Neural Networks

# Time Delay Neural Networks

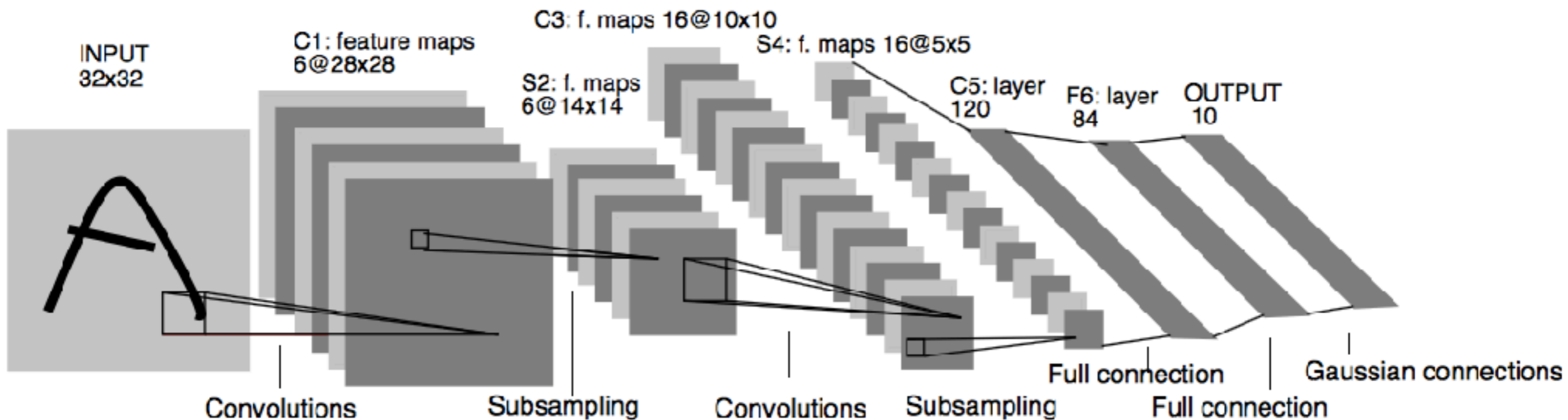
(Waibel et al. 1989)



**These are soft 2-grams!**

# Convolutional Networks

(LeCun et al. 1997)



Parameter extraction performs a 2D sweep, not 1D

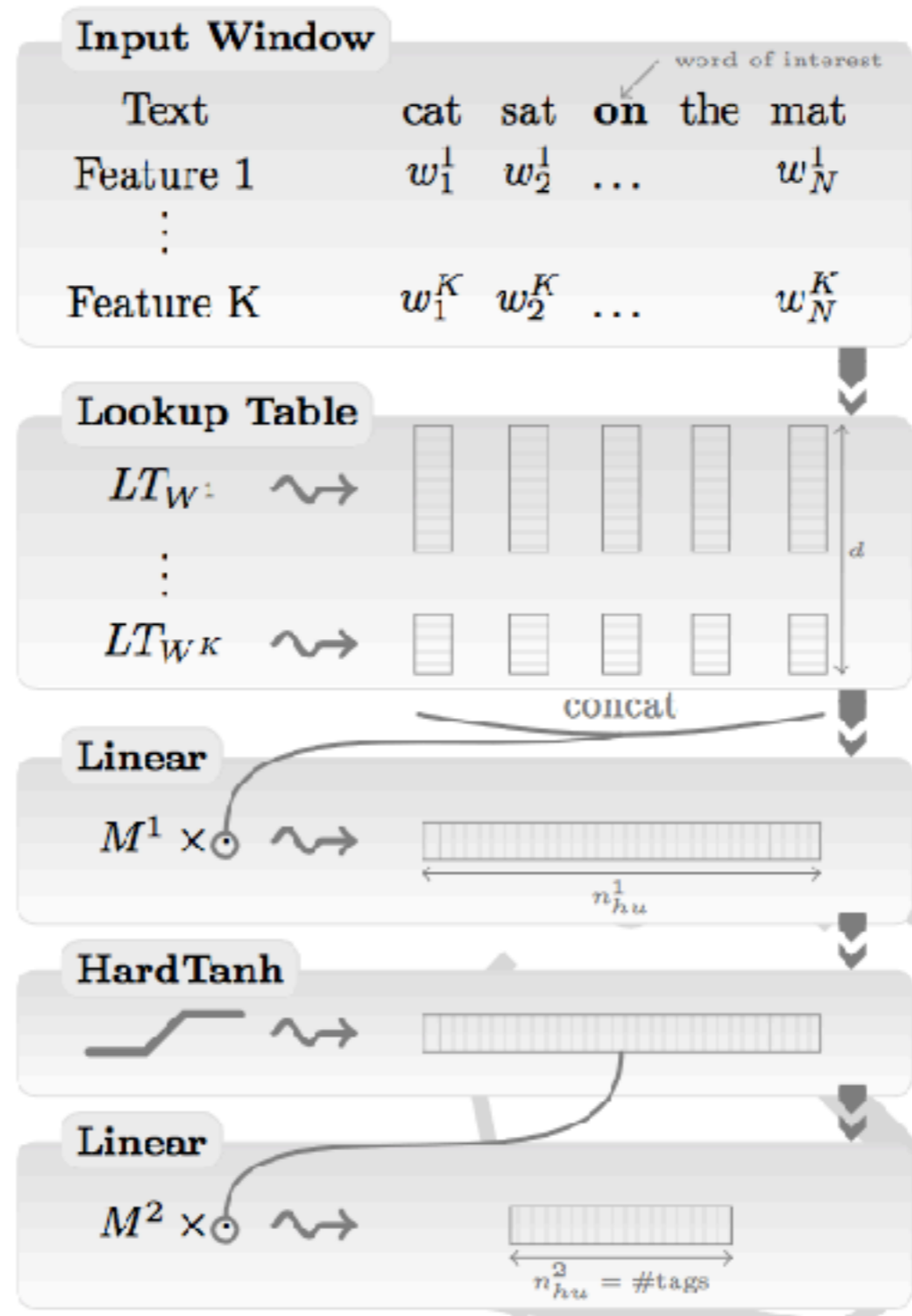
# CNNs for Text

(Collobert and Weston 2011)

- 1D convolution  $\approx$  Time Delay Neural Network
  - But often uses terminology/functions borrowed from image processing
- Two main paradigms:
  - **Context window modeling:** For tagging, etc. get the surrounding context before tagging
  - **Sentence modeling:** Do convolution to extract n-grams, pooling to combine over whole sentence

# CNNs for Tagging

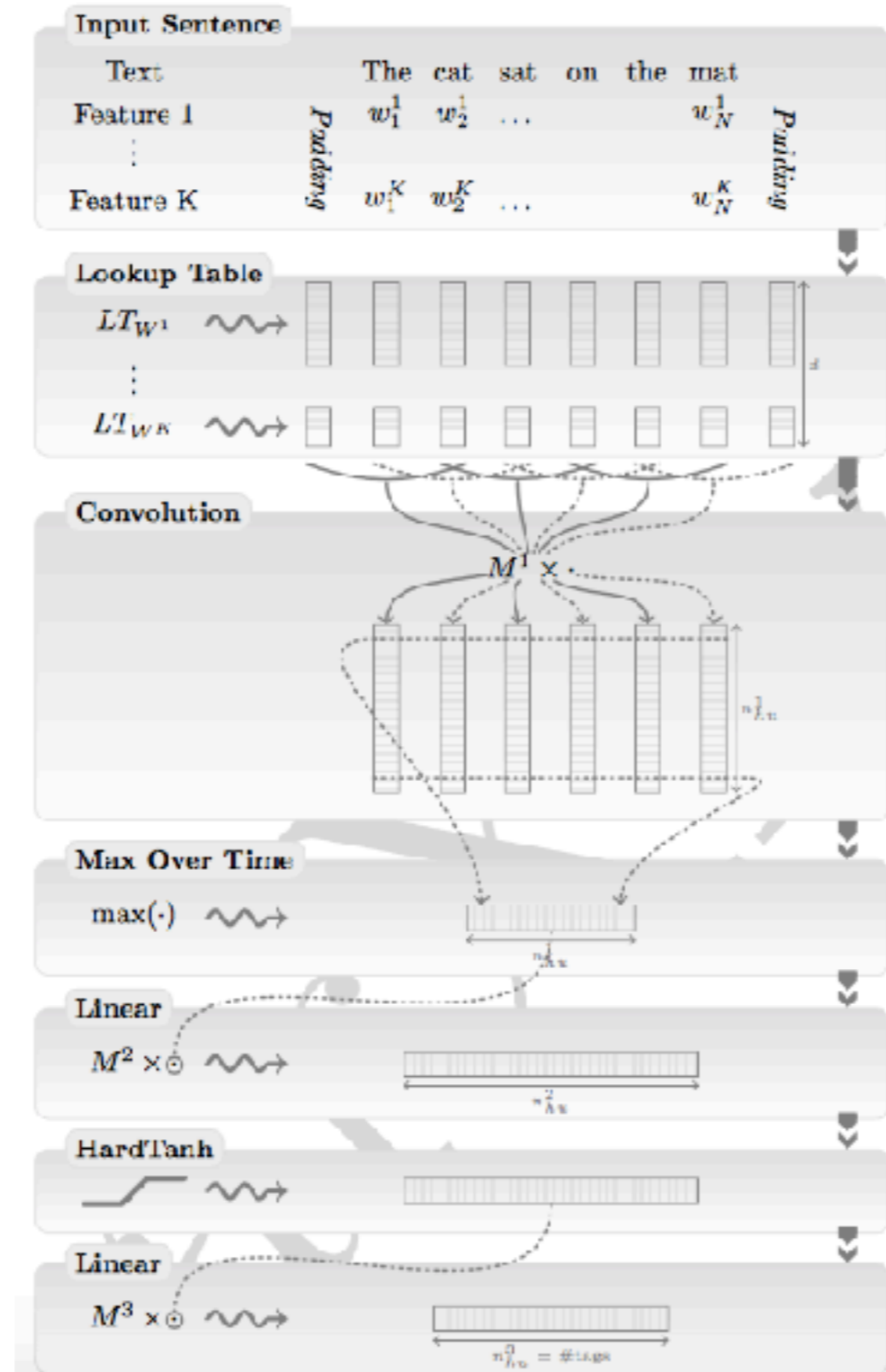
(Collobert and Weston 2011)





# CNNs for Sentence Modeling

(Collobert and Weston 2011)

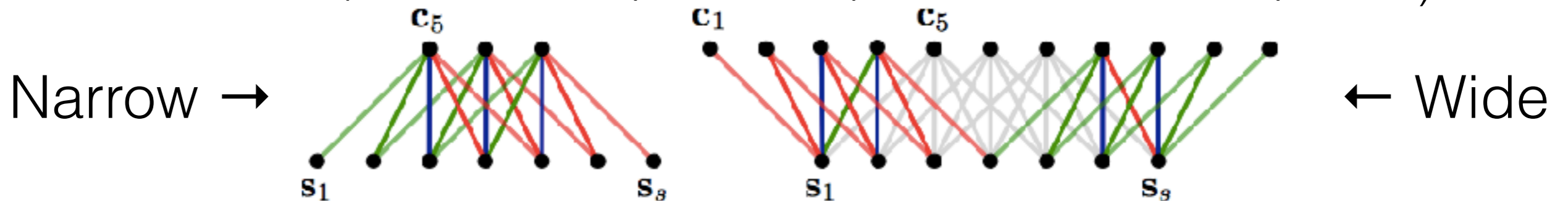


# Standard conv2d Function

- 2D convolution function takes input + parameters
- **Input:** 3D tensor
  - rows (e.g. words), columns, features (“channels”)
- **Parameters/Filters:** 4D tensor
  - rows, columns, input features, output features

# Padding/Striding

- **Padding:** After convolution, the rows and columns of the output tensor are either
  - = to rows/columns of input tensor (“same” convolution)
  - = to rows/columns of input tensor minus the size of the filter plus one (“valid” or “narrow”)
  - = to rows/columns of input tensor plus filter minus one (“wide”)



- **Striding:** It is also common to skip rows or columns (e.g. a stride of [2,2] means use every other)

# Pooling

- Pooling is like convolution, but calculates some reduction function feature-wise
- **Max pooling:** “Did you see this feature anywhere in the range?” (most common)
- **Average pooling:** “How prevalent is this feature over the entire range”
- **k-Max pooling:** “Did you see this feature up to k times?”
- **Dynamic pooling:** “Did you see this feature in the beginning? In the middle? In the end?”

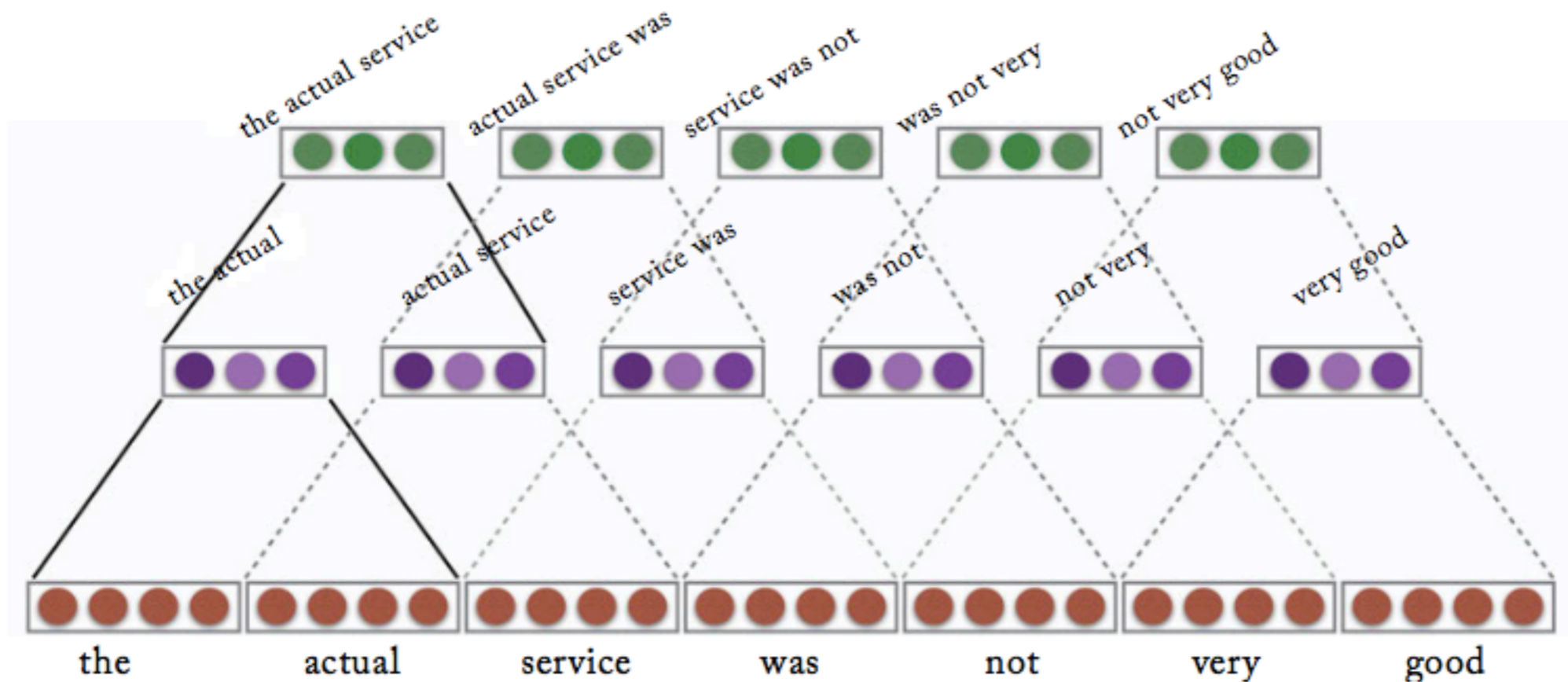
Let's Try It!

`cnn-class.py`

# Stacked Convolution

# Stacked Convolution

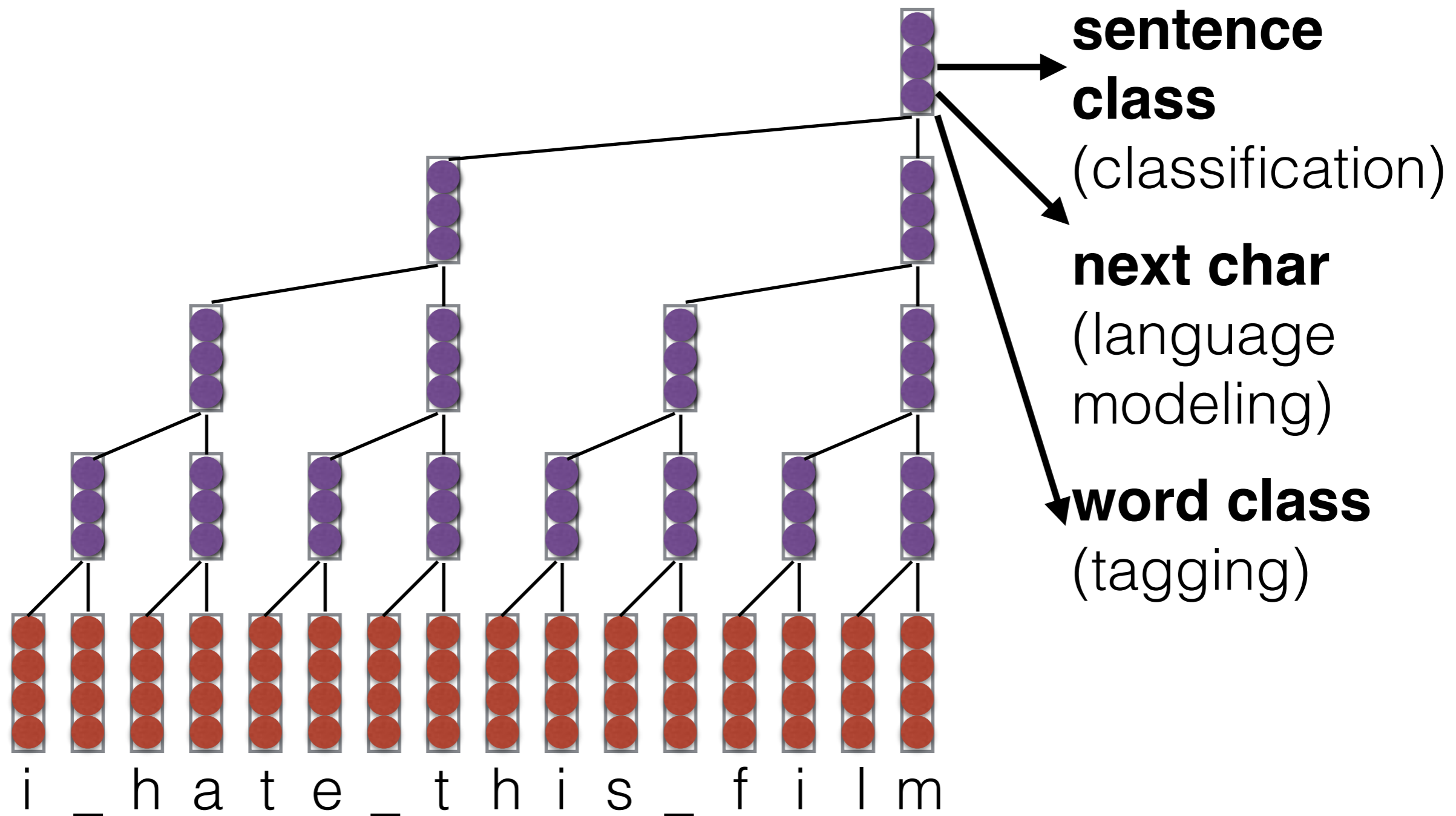
- Feeding in convolution from previous layer results in larger area of focus for each feature



# Dilated Convolution

(e.g. Kalchbrenner et al. 2016)

- **Gradually increase stride:** low-level to high-level





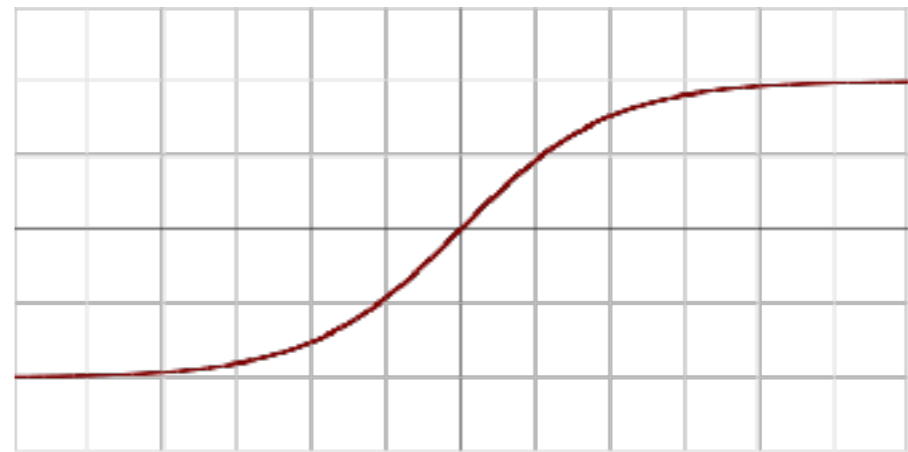
# An Aside: Nonlinear Functions

- Proper choice of a non-linear function is essential in stacked networks

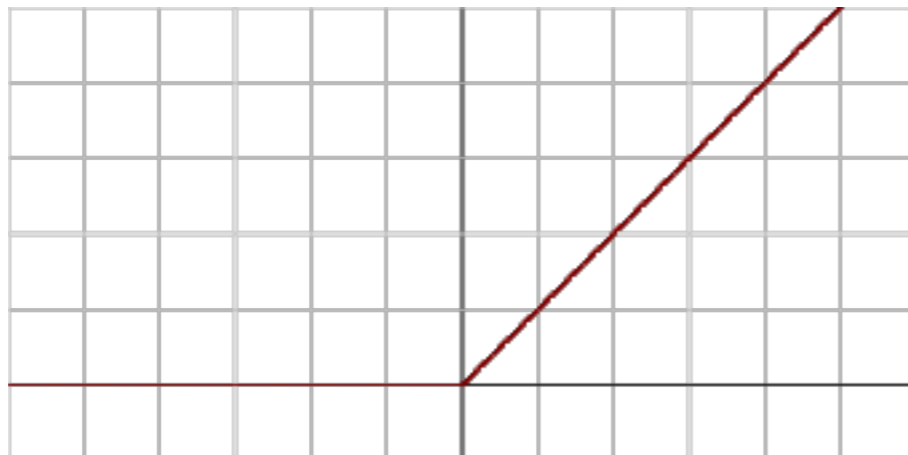
step



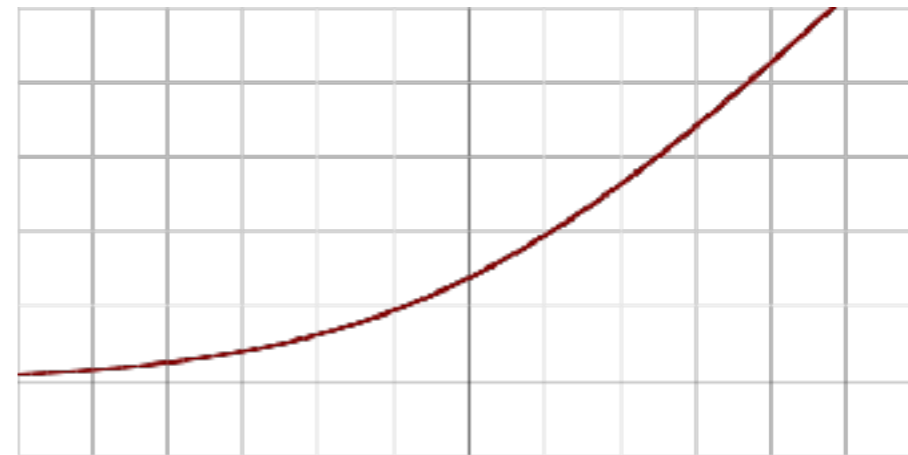
tanh



rectifier  
(ReLU)



soft  
plus



- Functions such as ReLU or softplus often work better at preserving gradients

# Why (Dilated) Convolution for Modeling Sentences?

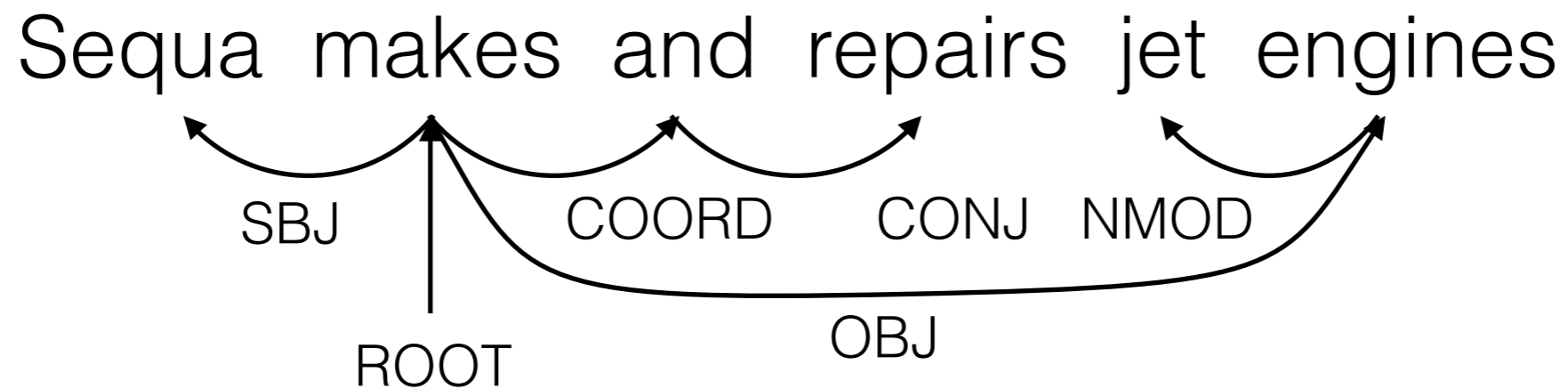
- In contrast to recurrent neural networks (next class)
- + Fewer steps from each word to the final representation: RNN  $O(N)$ , Dilated CNN  $O(\log N)$
- + Easier to parallelize on GPU
- - Slightly less natural for arbitrary-length dependencies
- - A bit slower on CPU?

# Structured Convolution

# Why Structured Convolution?

- Language has structure, would like it to localize features
- e.g. noun-verb pairs very informative, but not captured by normal CNNs

# Example: Dependency Structure



# Tree-structured Convolution

(Ma et al. 2015)

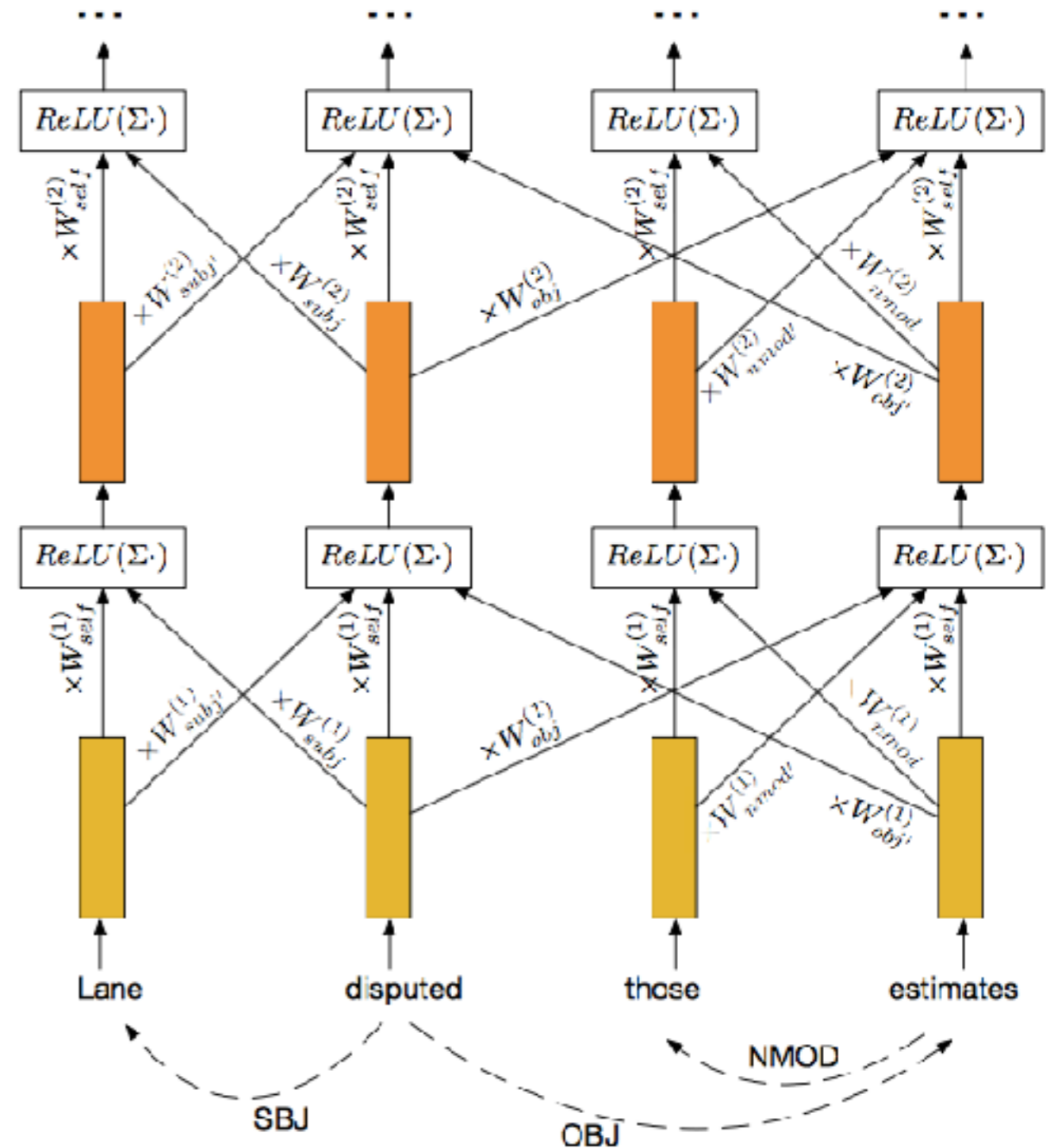
- Convolve over parents, grandparents, siblings

ancestor paths		siblings	
$n$	pattern(s)	$n$	pattern(s)
3		2	
4		3	
5		4	

# Graph Convolution

(e.g. Marcheggiani et al. 2017)

- Convolution is shaped by graph structure
- For example, dependency tree is a graph with
  - Self-loop connections
  - Dependency connections
  - Reverse connections



# Convolutional Models of Sentence Pairs



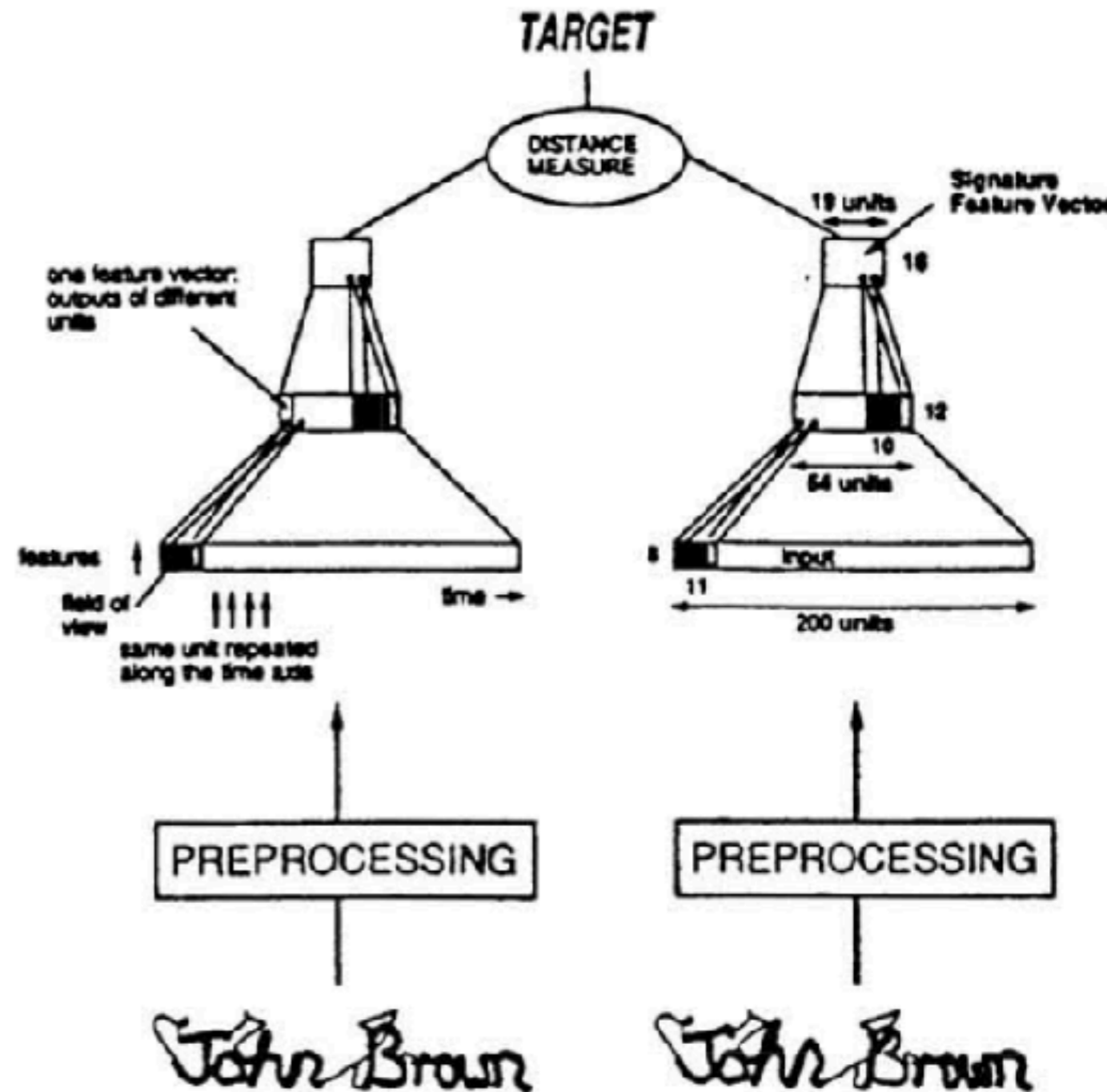
# Why Model Sentence Pairs?

- Paraphrase identification / sentence similarity
- Textual entailment
- Retrieval
- (More about these specific applications in two classes)

# Siamese Network

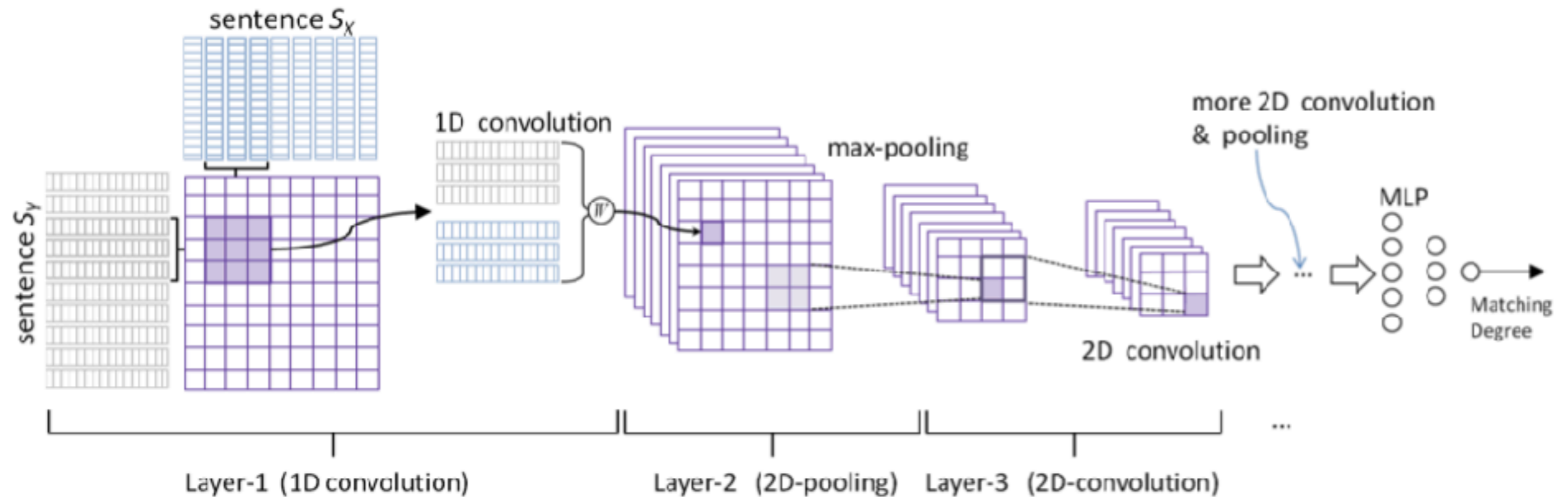
(Bromley et al. 1993)

- Use the same network, compare the extracted representations
- (e.g. Time-delay networks for signature recognition)



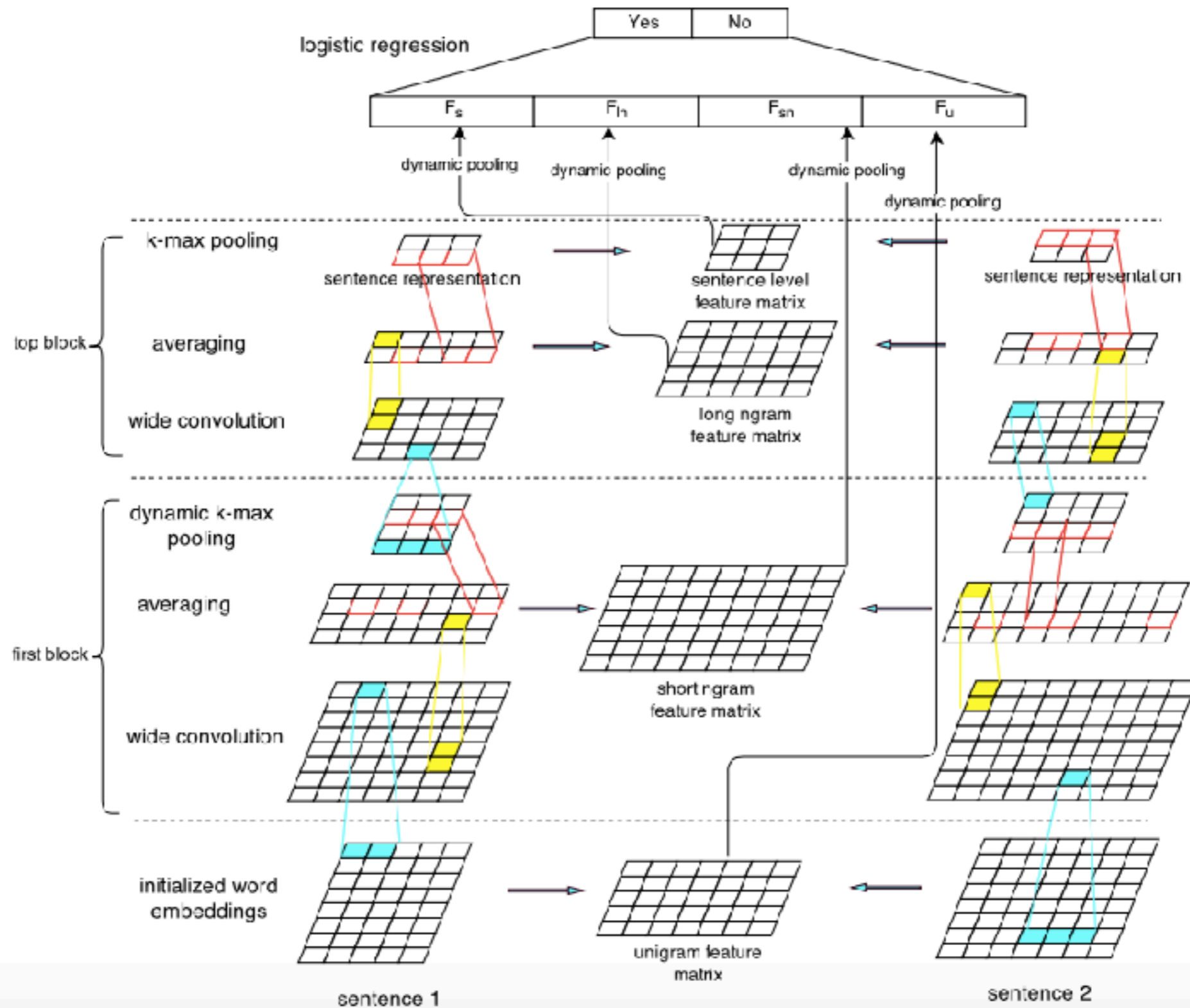
# Convolutional Matching Model (Hu et al. 2014)

- Concatenate sentences into a 3D tensor and perform convolution



- Shown more effective than simple Siamese network

# Convolutional Features + Matrix-based Pooling (Yin and Schutze 2015)



# Understanding CNN Results

# Why Understanding?

- Sometimes we want to know why model is making predictions (e.g. is there bias?)
- Understanding extracted features might lead to new architectural ideas
- Visualization of filters, etc. easy in vision but harder in NLP; other techniques can be used

# Maximum Activation

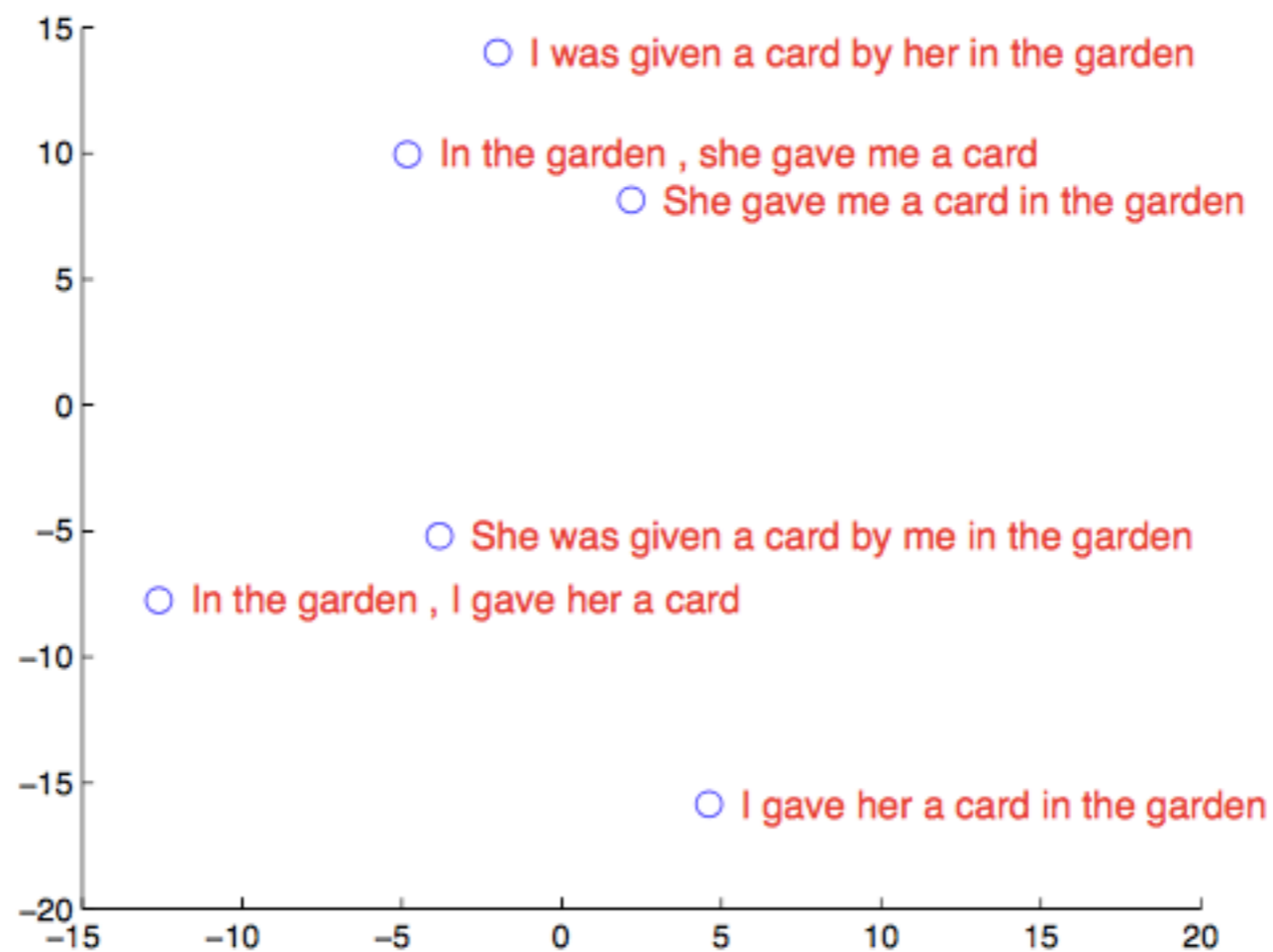
- Calculate the hidden feature values for whole data, find section of image/sentence that results in max value



Example: Karpathy 2016

# PCA/t-SNE Embedding of Feature Vector

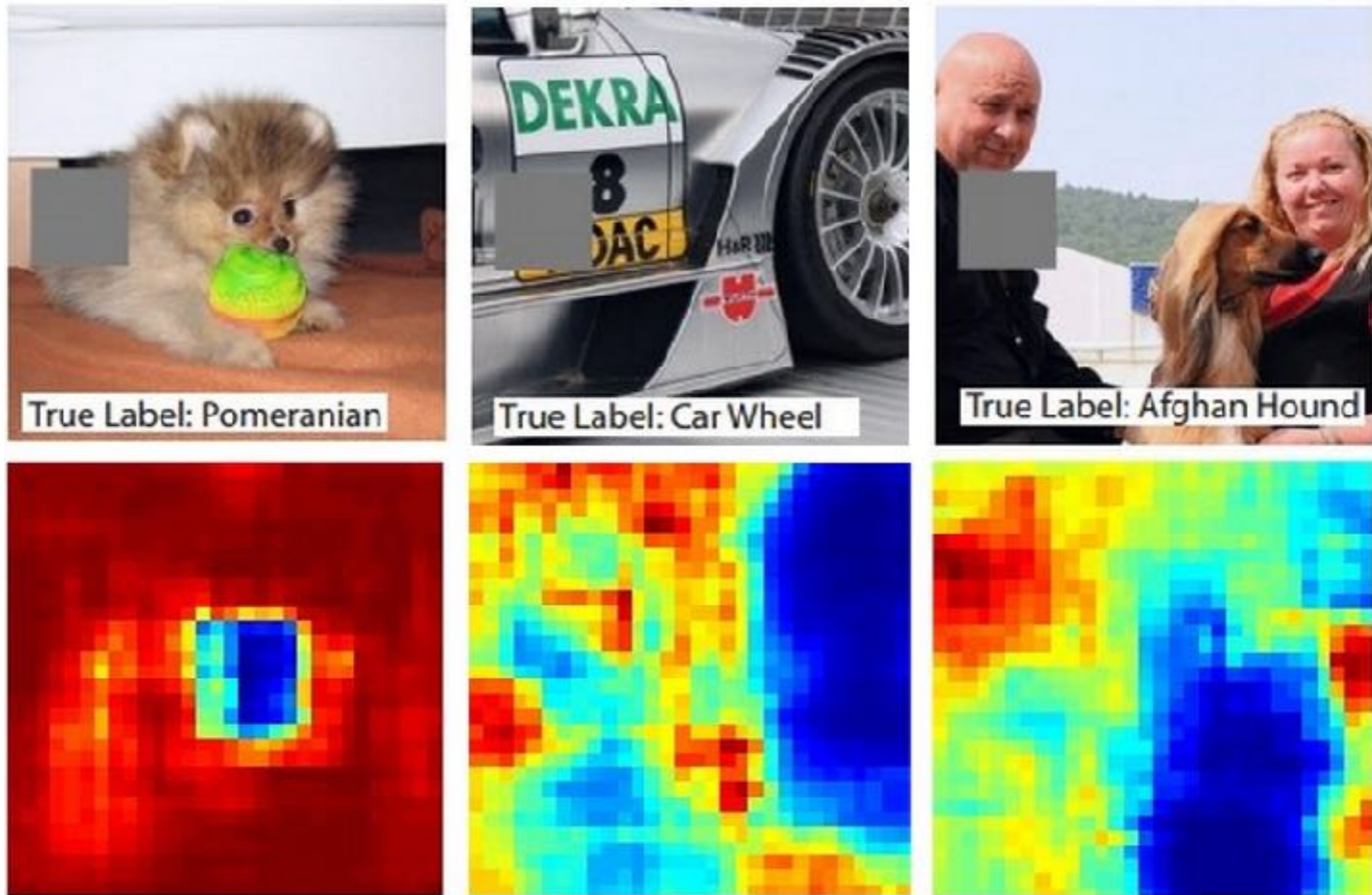
- Do dimension reduction on feature vectors





# Occlusion

- Blank out one part at a time (in NLP, word?), and measure the difference from the final representation/prediction



Example: Karpathy 2016

Let's Try It!

`cnn-activation.py`

Questions?