Natural Language Processing

Lecture 12: Deep Learning and NLP

Big picture: natural language analyzers

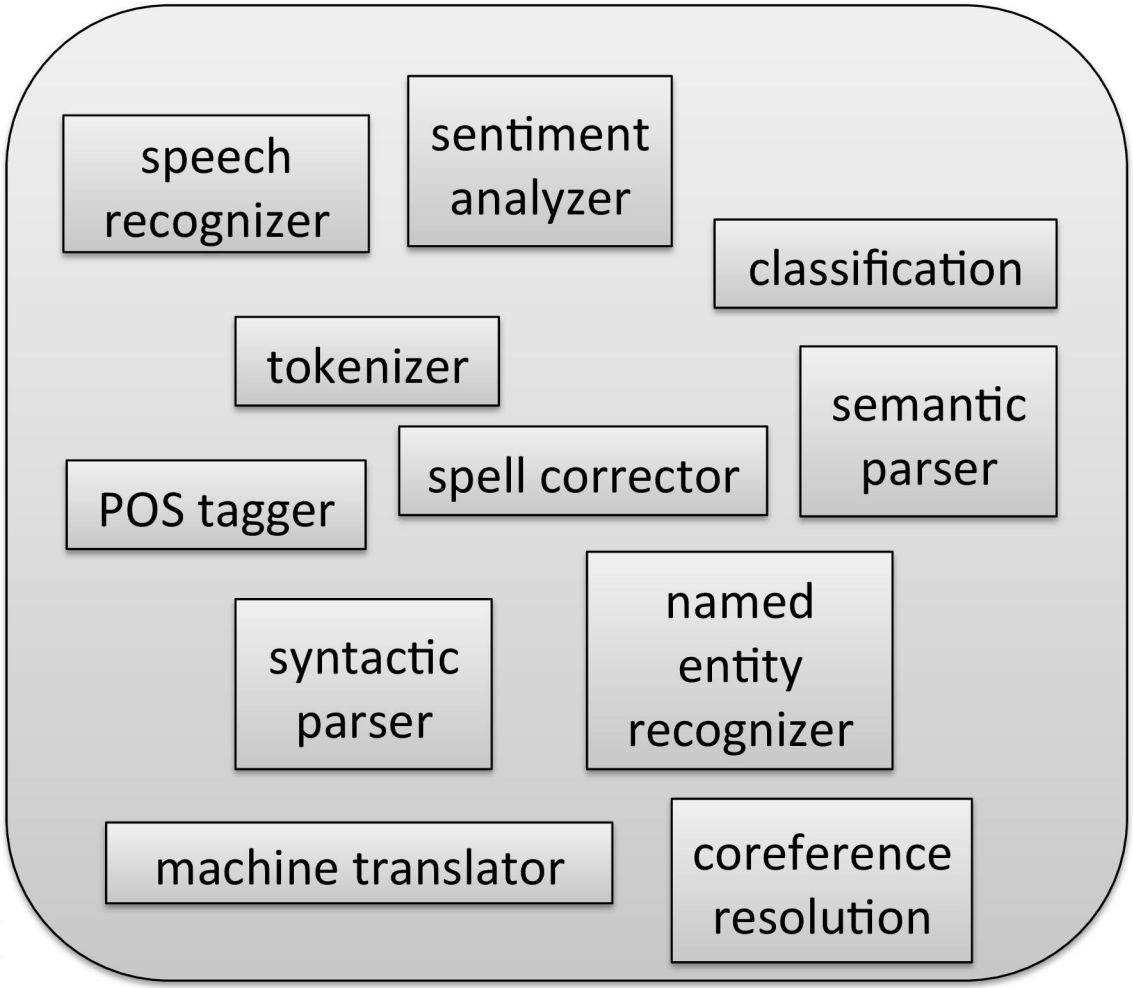
Natural language input signal:

- Web page
- Question
- Search query
- Tweet
- Voice command

Output analysis:

- Question
- Answer
- Command to a robot
- Trending topics

Big picture: natural language analyzers



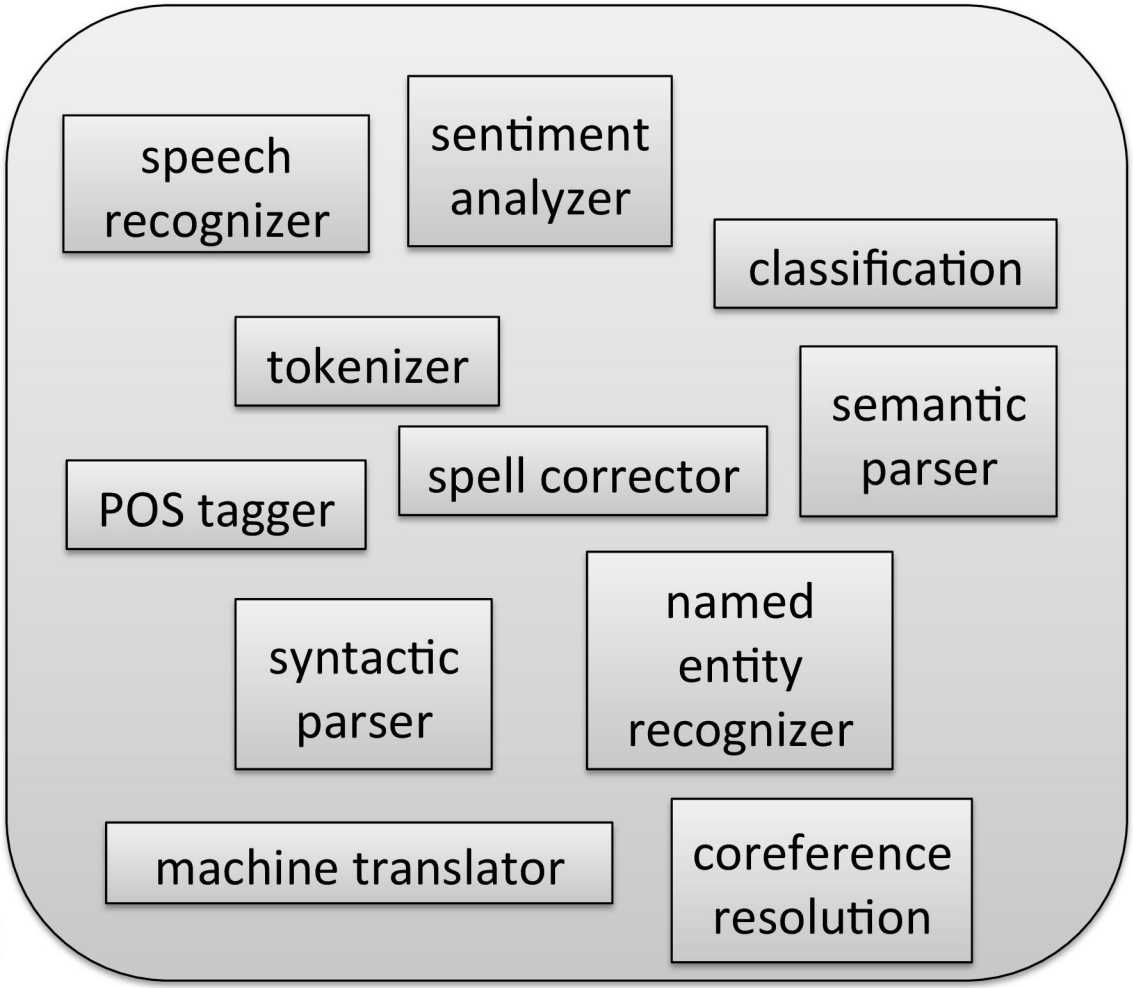
Natural language input signal:

- Web page
- Question
- Search query
- Tweet
- Voice command

Output analysis:

- Question
- Answer
- Command to a robot
- Trending topics

Today: deep learning for NLP components



Natural language input signal:

- Web page
- Question
- Search query
- Tweet
- Voice command

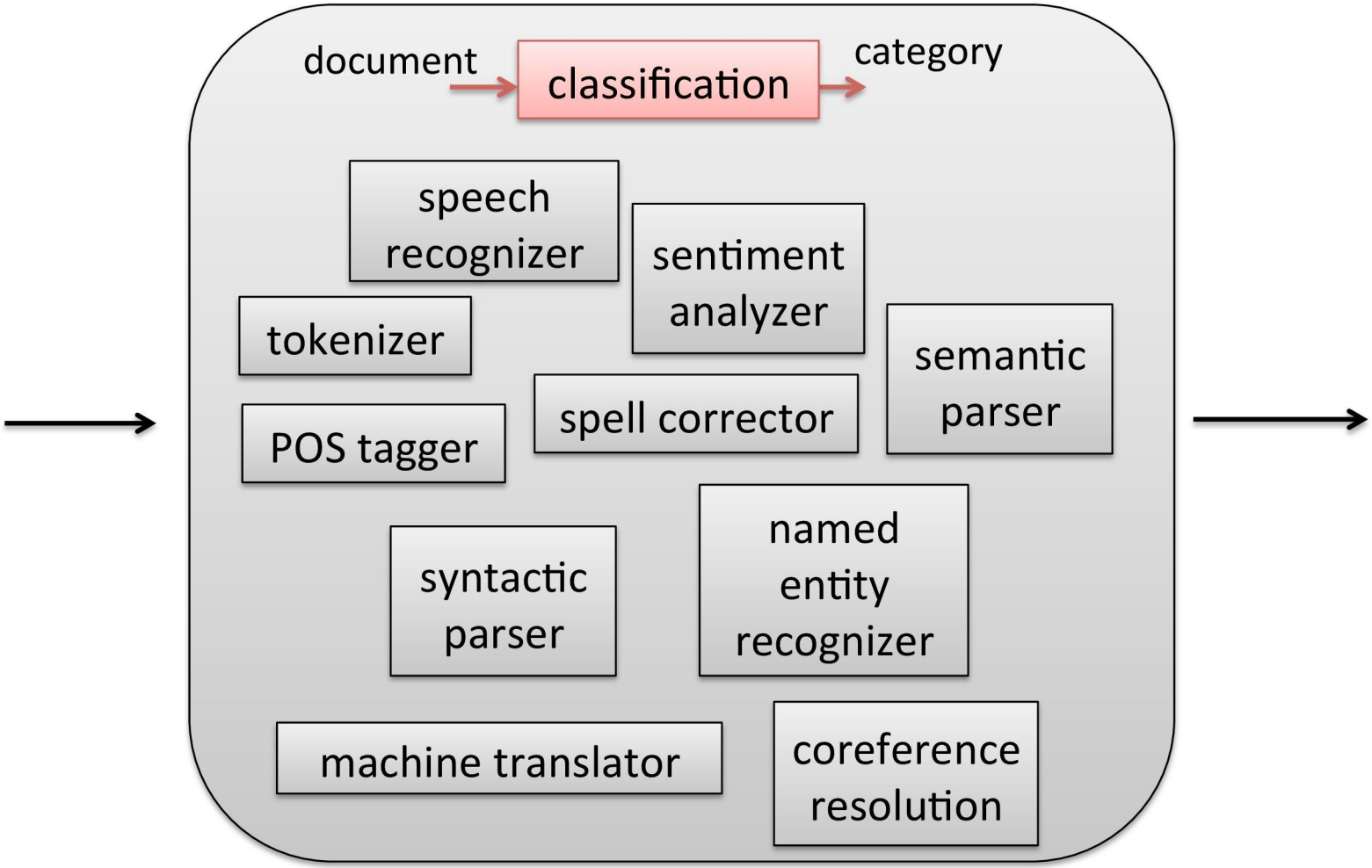
Output analysis:

- Question
- Answer
- Command to a robot
- Trending topics

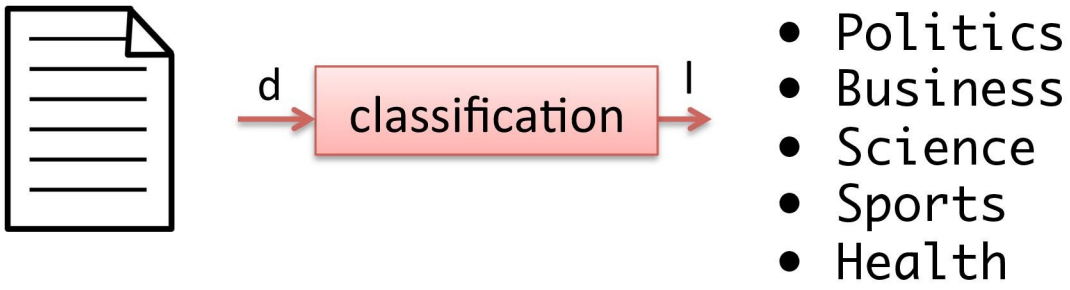
Agenda

- Big picture
- **Why deep learning?**
- Building blocks of a deep neural network
- How to train deep neural networks
- Important results

Running example: document classification



Running example: document classification



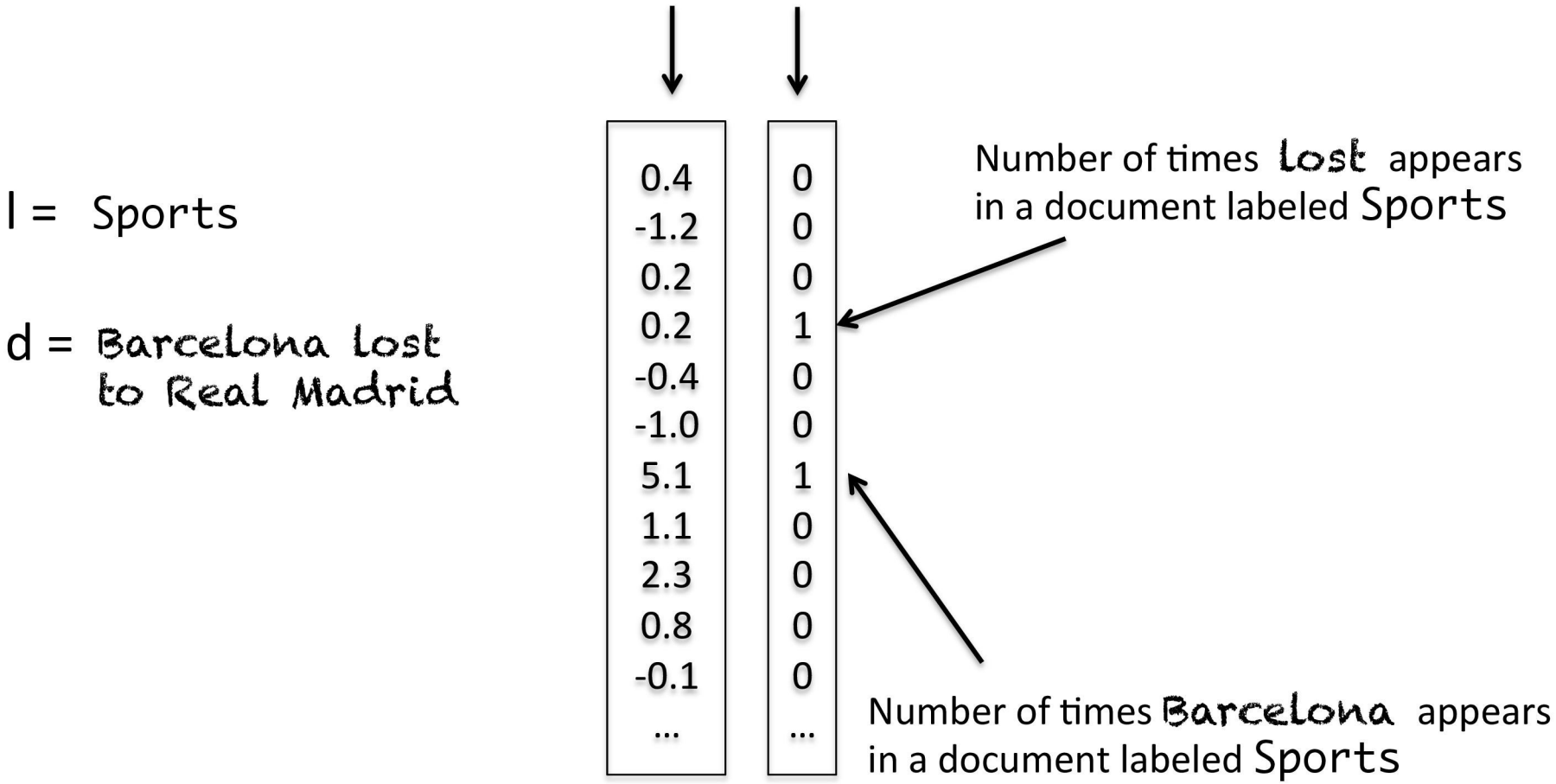
$$\text{output} = \text{argmax}_l\ f(l, d)$$

Sports

Barcelona lost to Real Madrid

How to define f(l, d): linear models

Linear models: f(l, d) = w . g(l,d)



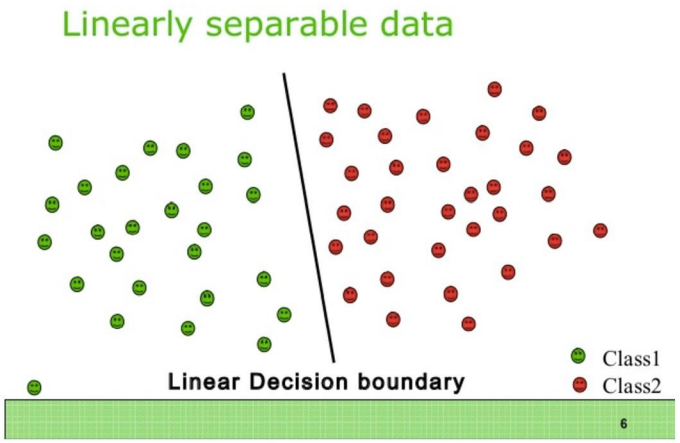
How to define f(l, d): linear models

Linear models: f(l, d) = w . g(l,d)

- Easy to implement
- Easy to optimize w

Two possible improvements:

- Define more complex functions
- Find better representations of (l,d)



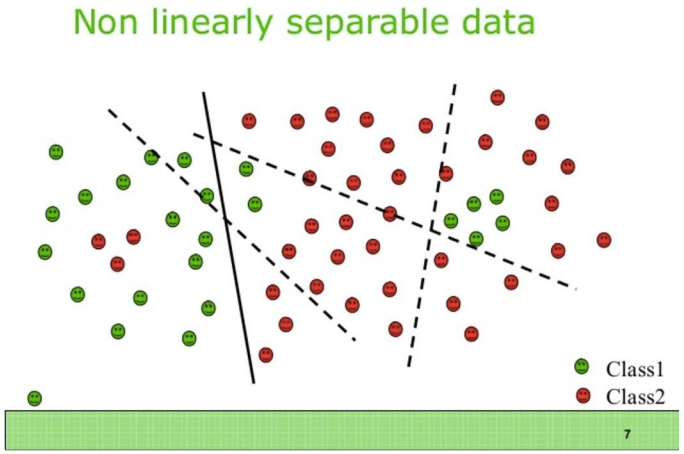


Figure credits: Barbara Rosario

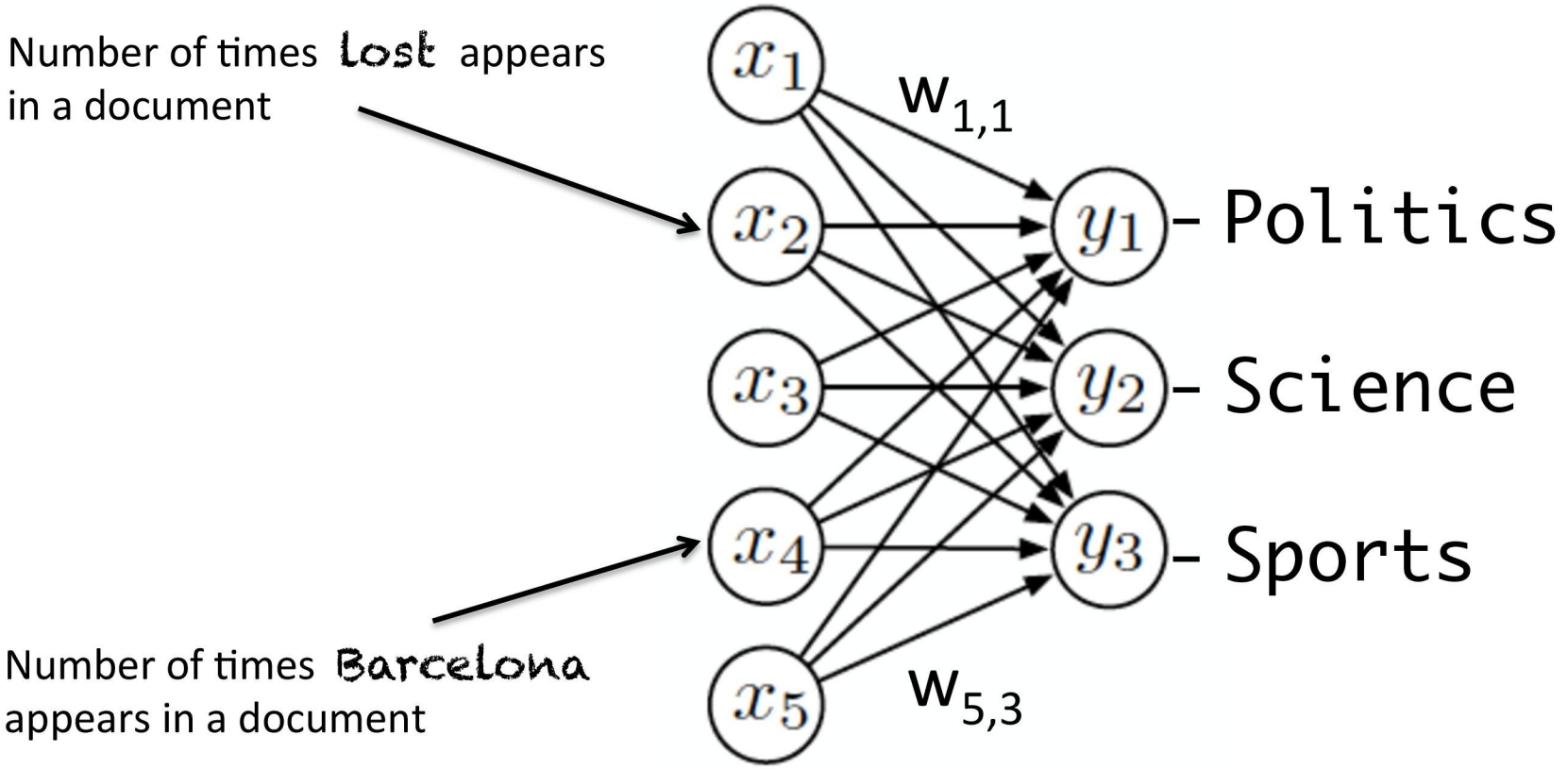
Agenda

- Big picture
- Why deep learning?
- **Building blocks of a deep neural network**
- How to train deep neural networks
- Important results

neural network v1.0: linear model

Linear models: f(l, d) = w . g(l,d) = w(l) . x(d)

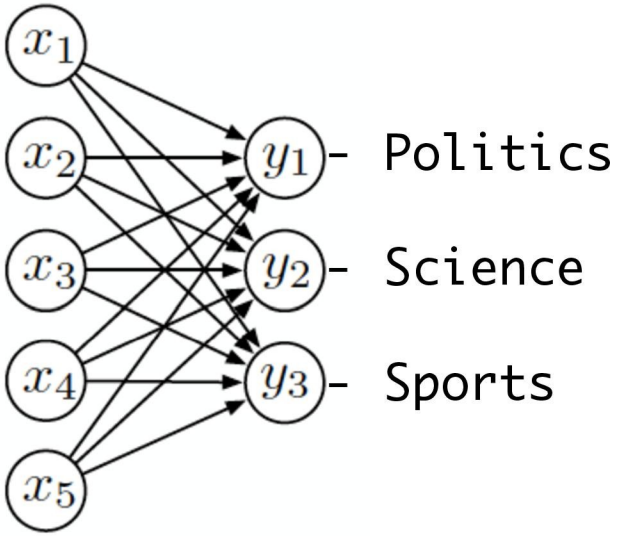
e.g., $y1 = x_1 w_{1,1} + x_2 w_{2,1} + x_3 w_{3,1} + x_4 w_{4,1} + x_5 w_{5,1} = w(1) . x(d)$



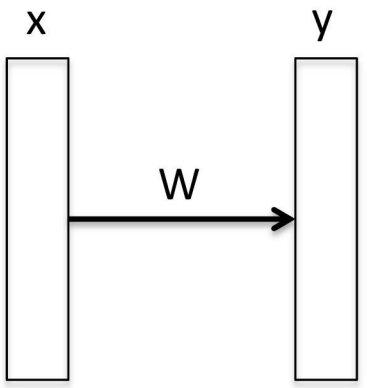
neural network v1.0: linear model

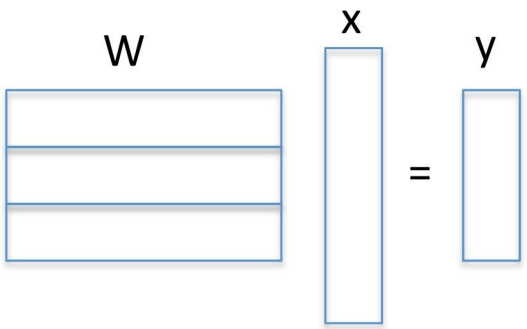
Linear models: f(l, d) = w . g(l,d) = w(l) . x(d)

e.g., $y1 = x_1 w_{1,1} + x_2 w_{2,1} + x_3 w_{3,1} + x_4 w_{4,1} + x_5 w_{5,1} = w(1) . x(d)$



same as

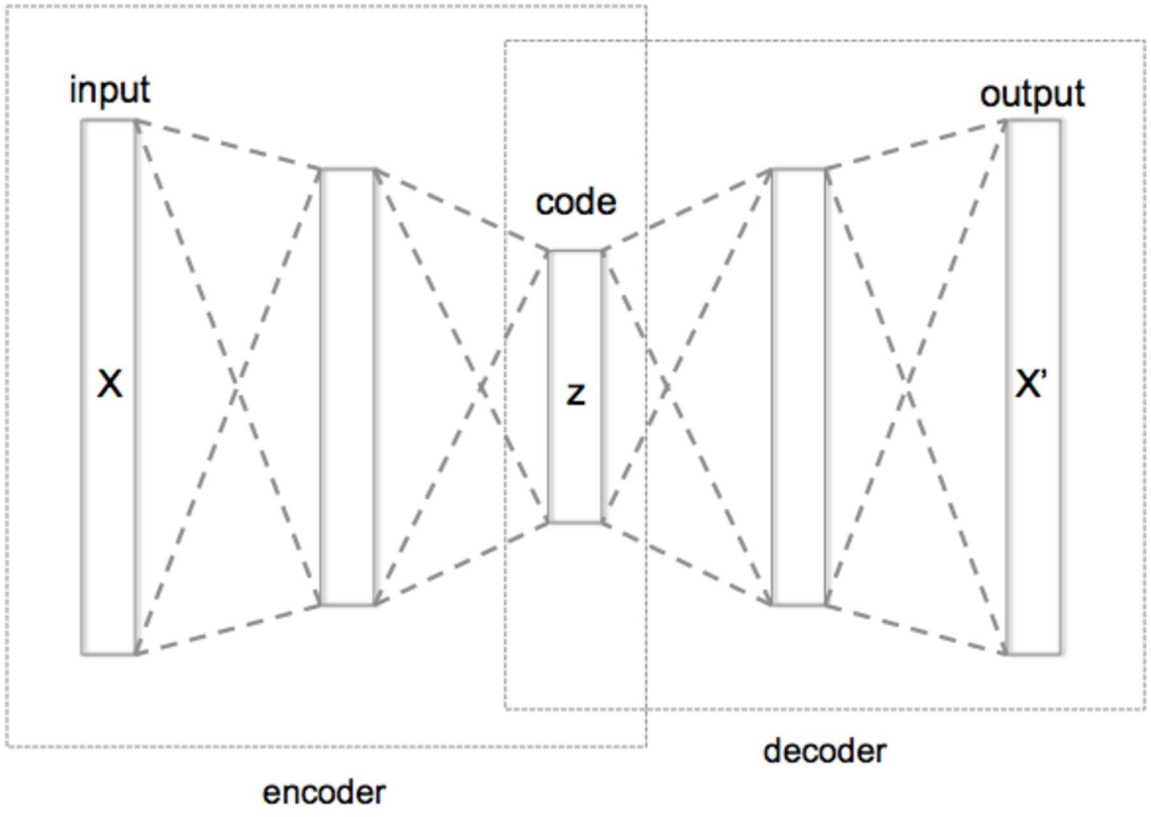




similar words still share no parameters!

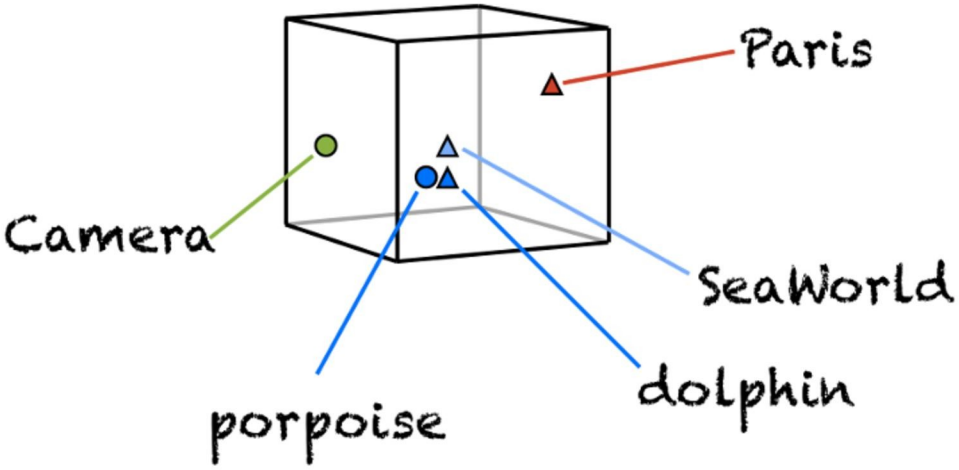
neural network v2.0: representation learning

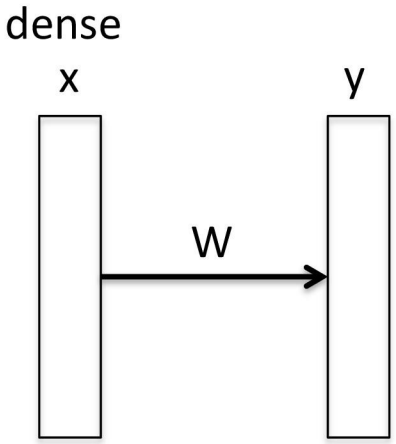
Big idea: induce low-dimensional dense feature representations of high-dimensional objects



neural network v2.1: representation learning

Big idea: embed words in a dense vector space and use the word embeddings as dense features

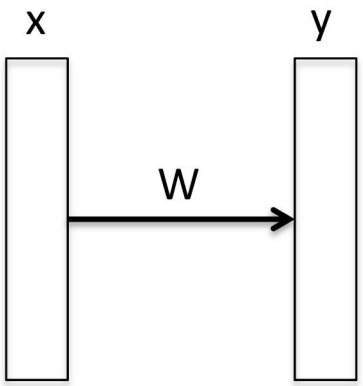




Did this really solve the problem?

neural network v3.0: complex functions

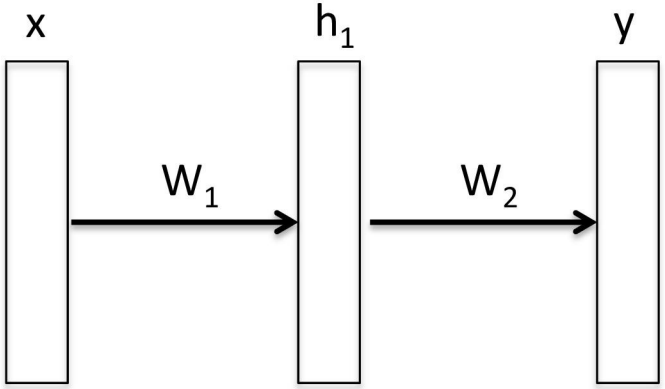
Big idea: define more complex functions by adding a hidden layer



$$y = W x$$

neural network v3.0: complex functions

Big idea: define more complex functions by adding a hidden layer

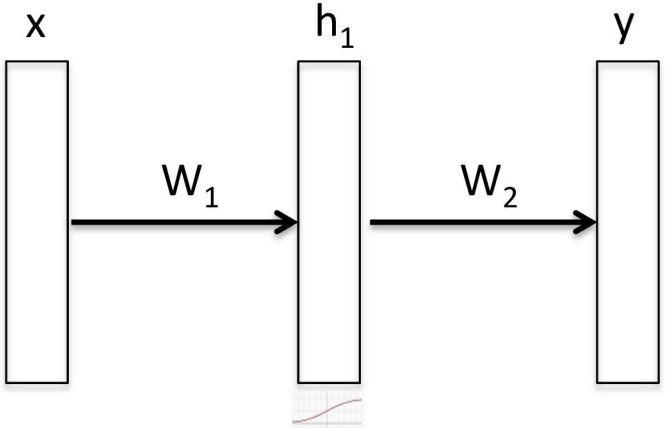


$$y = W_2 h_1 = W_2 (W_1 x)$$
$$= W x$$

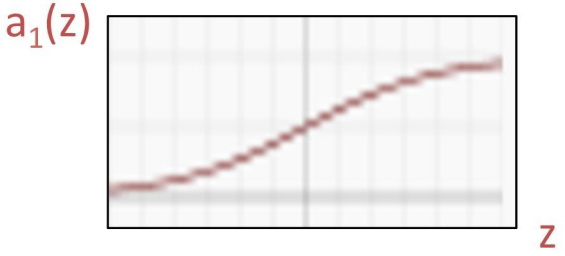
Wait! Is that true?!

neural network v3.0: complex functions

Big idea: define more complex functions by adding a hidden layer



non-linear functions,
e.g., logistic function
$a_1(z) = 1 / (1 + e^{-z})$



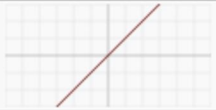
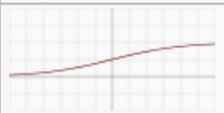
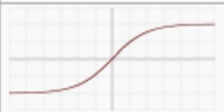
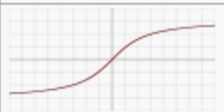
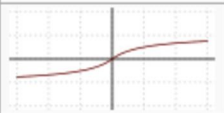
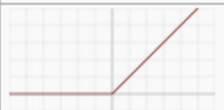
$$y = W_2 h_1 = W_2 \boxed{a_1(W_1 x)}$$

induced features

Universal approximation theorem
Cybenko., G. (1989)

neural network v3.0: complex functions

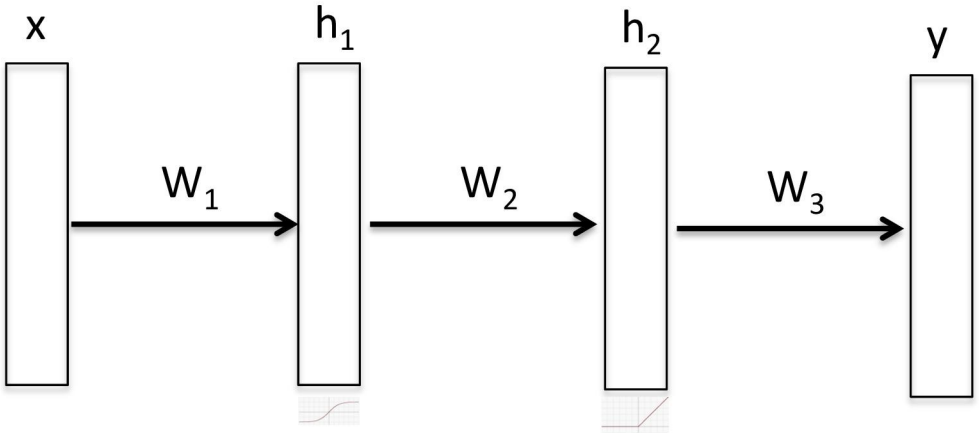
Popular activation/transfer/non-linear functions:

Name	Plot	Equation	Derivative	Range	Order of continuity
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	C^∞
Softsign [7]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$	$(-1, 1)$	C^1
Rectified Linear Unit (ReLU)[8]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0

https://en.wikipedia.org/wiki/Activation_function

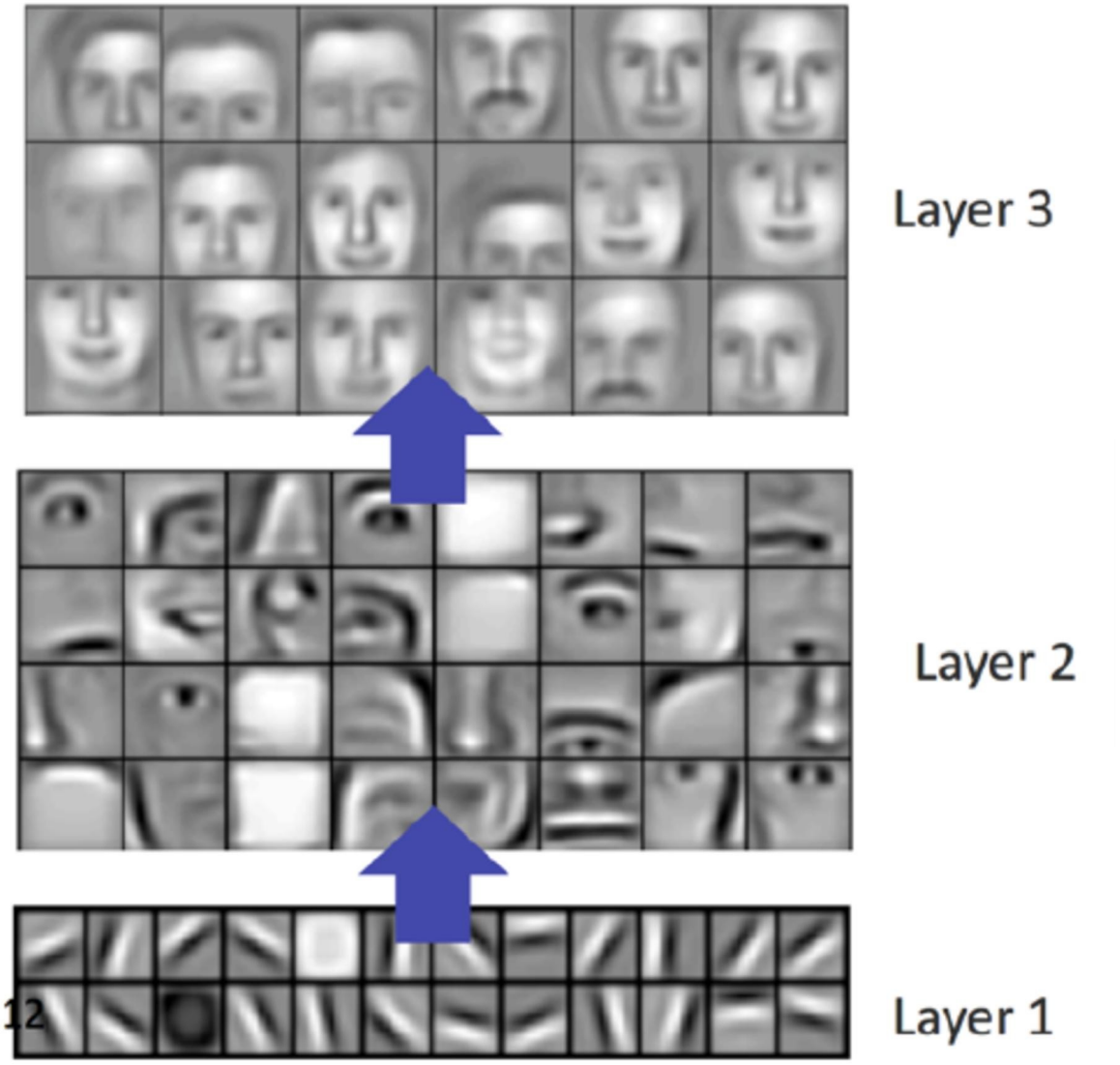
neural network v3.5: “deeper” networks

Wait but why do we need more layers?



$$y = W_3 h_2 = W_3 a_2(W_2 a_1(W_1 x))$$

neural network v3.5: “deeper” networks



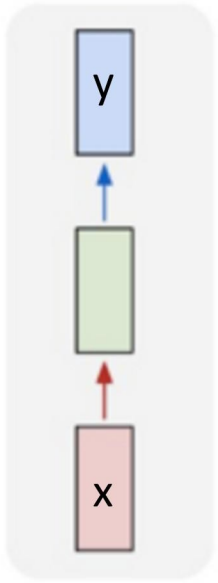
12

neural network v4.0: recurrent neural networks

Big idea: use hidden layers to represent sequential state

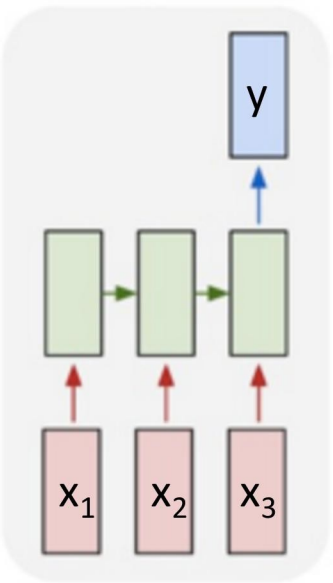
Feed-forward neural networks

How did we represent x for document classification?



Real .... Madrid
=
Real Madrid

Recurrent neural networks



Real .... Madrid
≠
Real Madrid

Do we share parameters across states?

Figure credits: Andrej Karpathy

neural network v4.0: recurrent neural networks

How to compute the hidden layers?

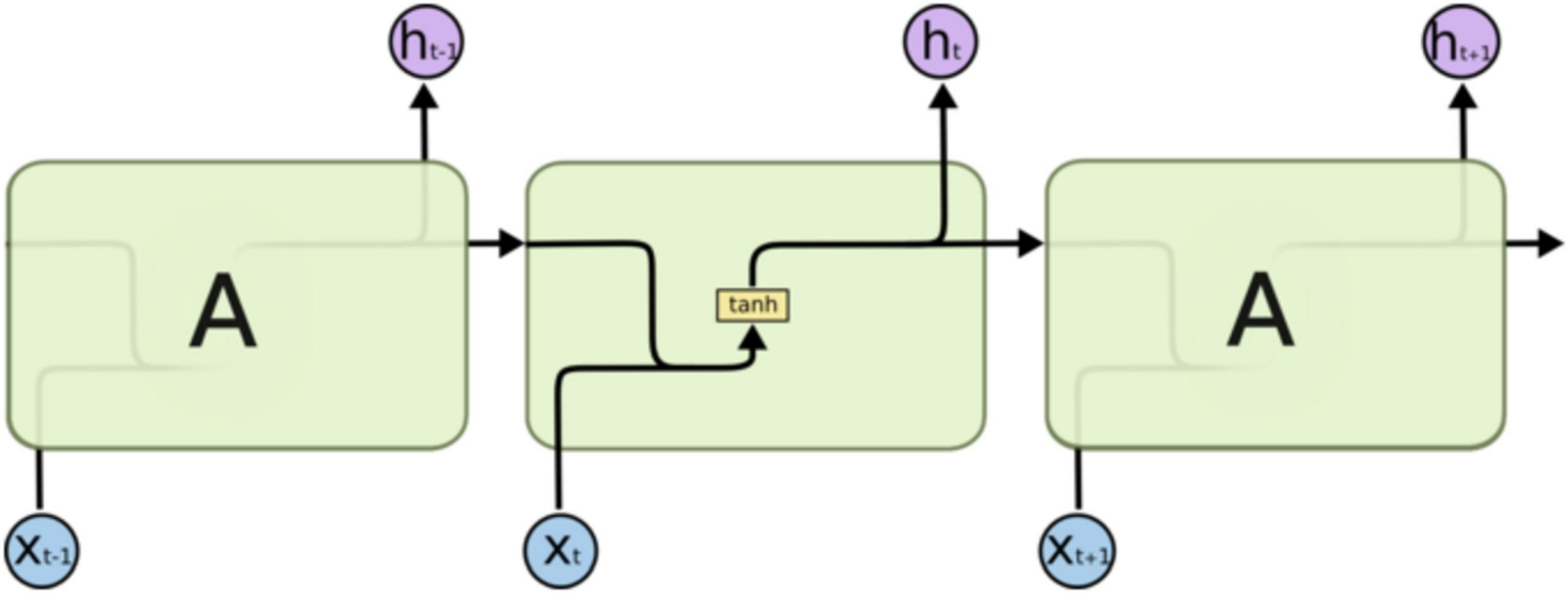


Figure credits: Christopher Olah

neural network v4.1: output sequences

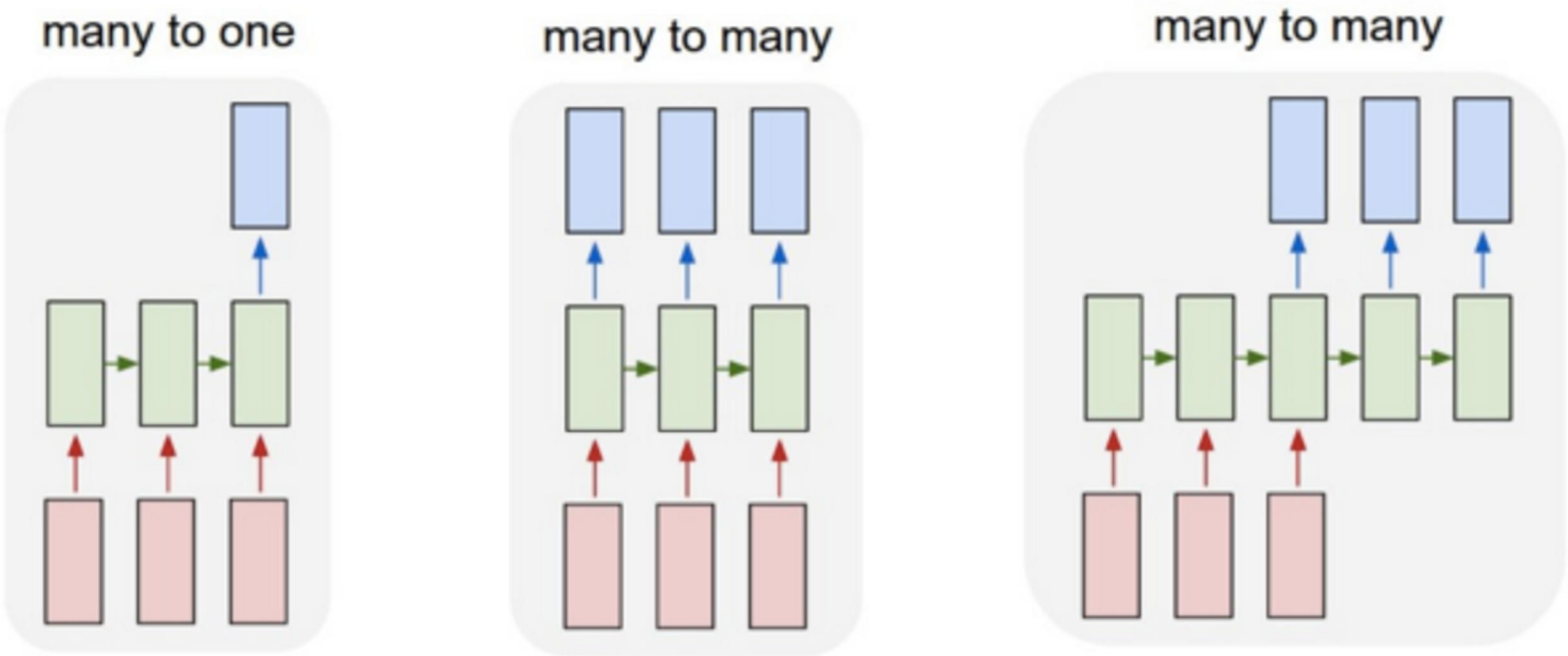


Figure credits: Andrej Karpathy

neural network v4.1: output sequences

Example:

Character-level language models

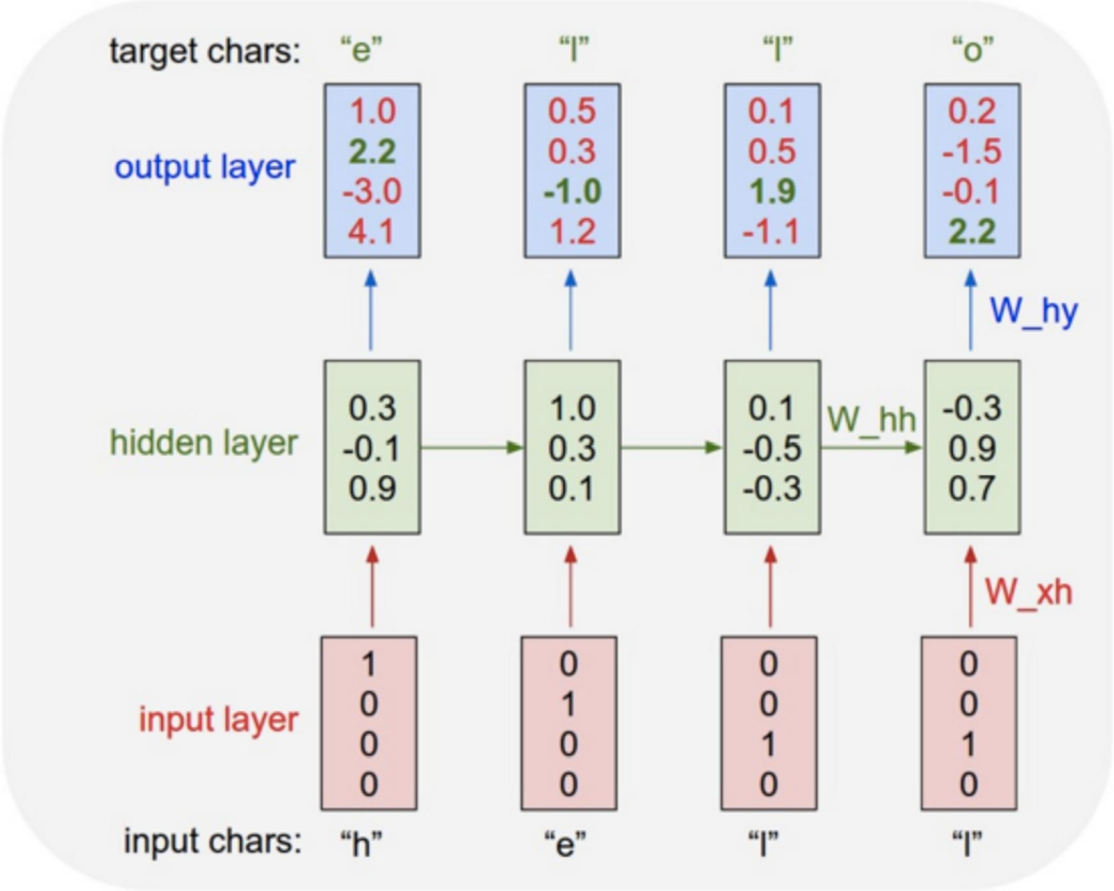


Figure credits: Andrej Karpathy

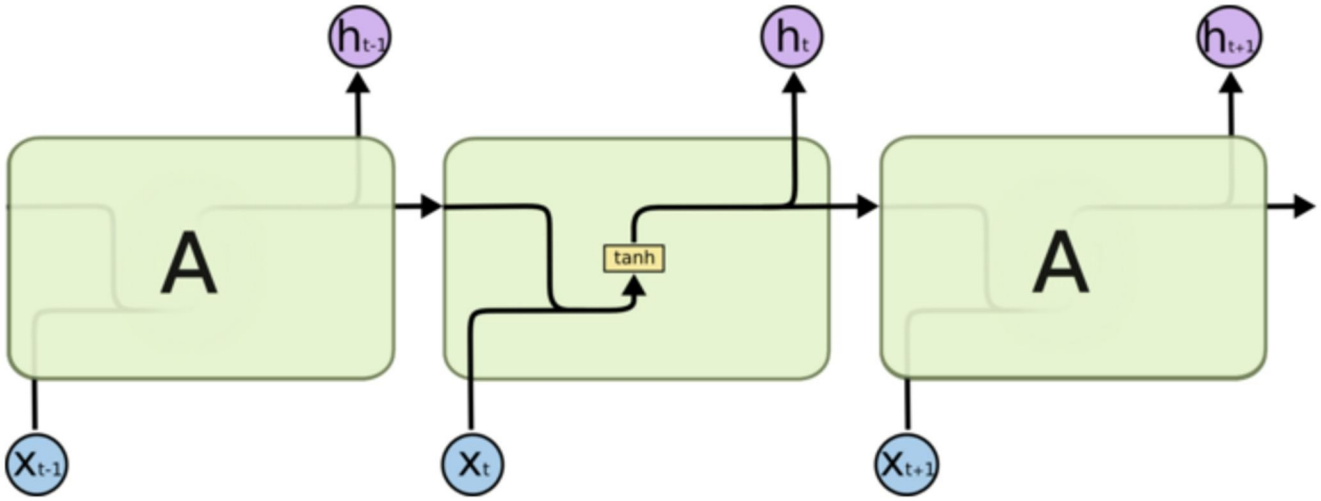
neural network v4.1: output sequences

Sample output:

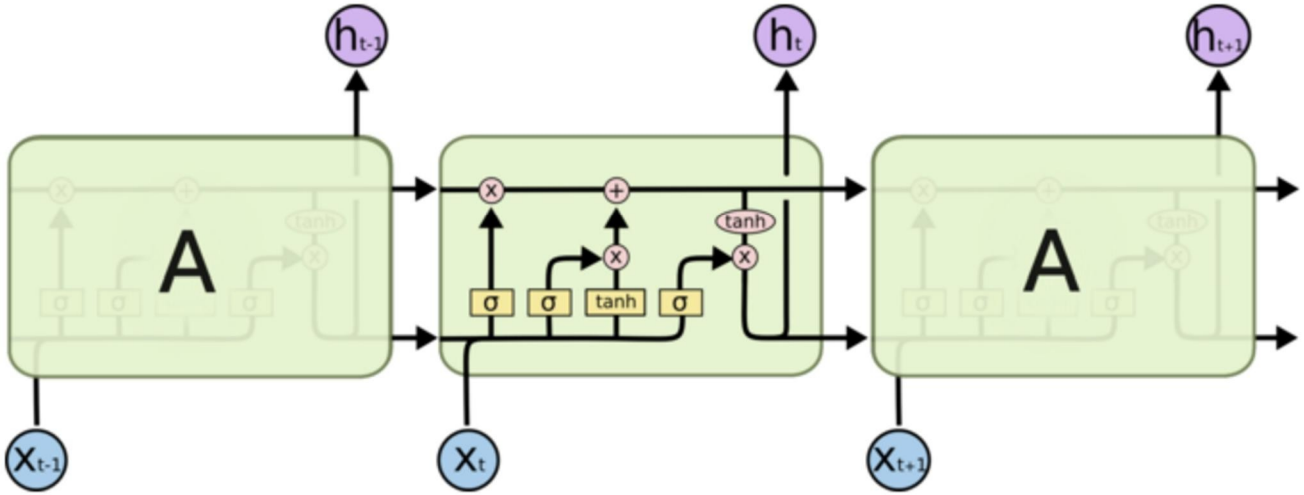
Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

Credits: Andrej Karpathy

neural network v4.2: Long-Short Term Memory



Regular RNNs



LSTMs

Figure credits: Christopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs/

neural network v4.2: Long-Short Term Memory

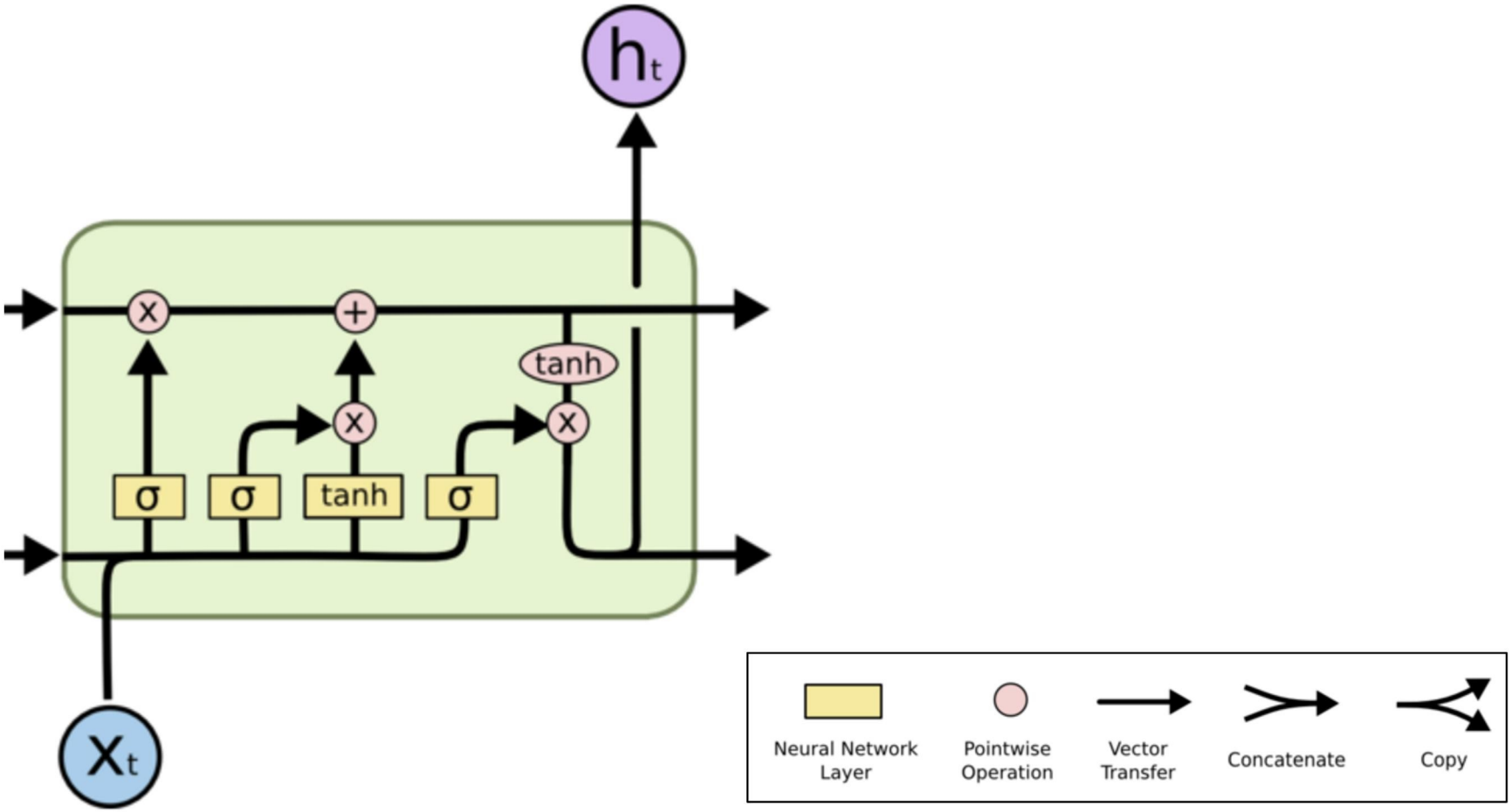
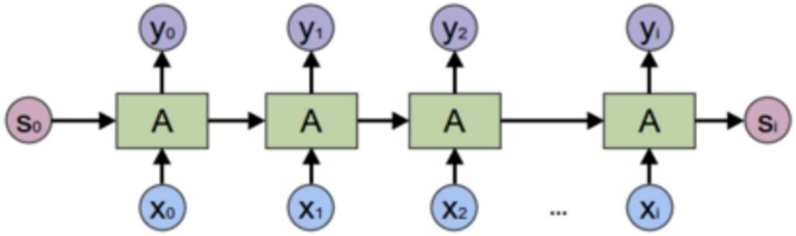


Figure credits: Christopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs/

neural network v4.3: bidirectional RNNs

Unidirectional RNNs



Bidirectional RNNs

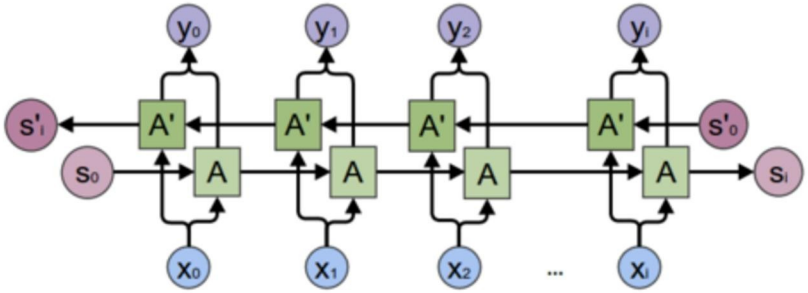
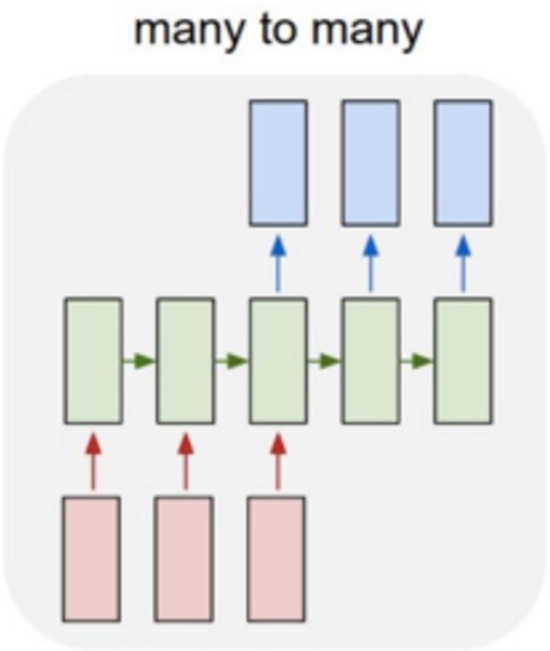


Figure credits: Christopher Olah

neural network v4.4: attention models



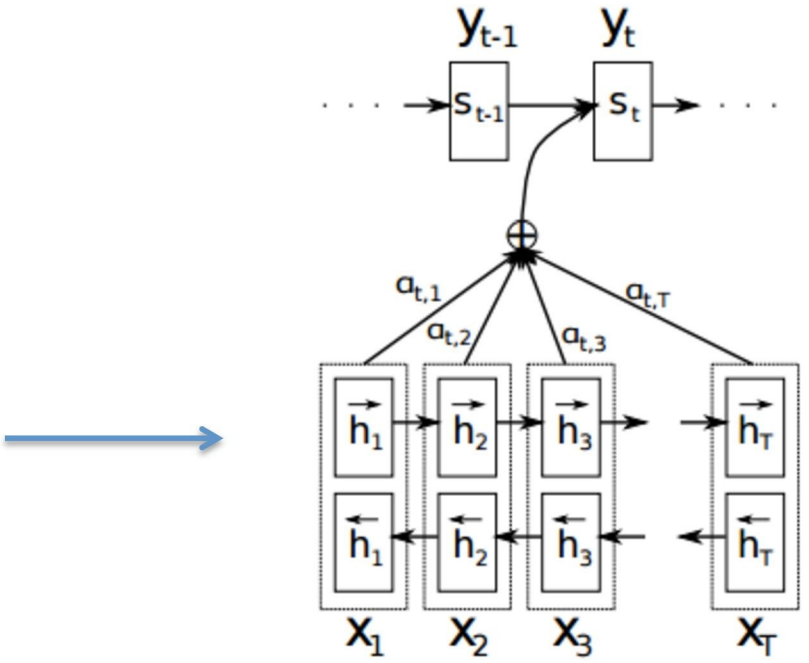


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Bahdanau et al. (2015)

neural network v5: convolutional NN

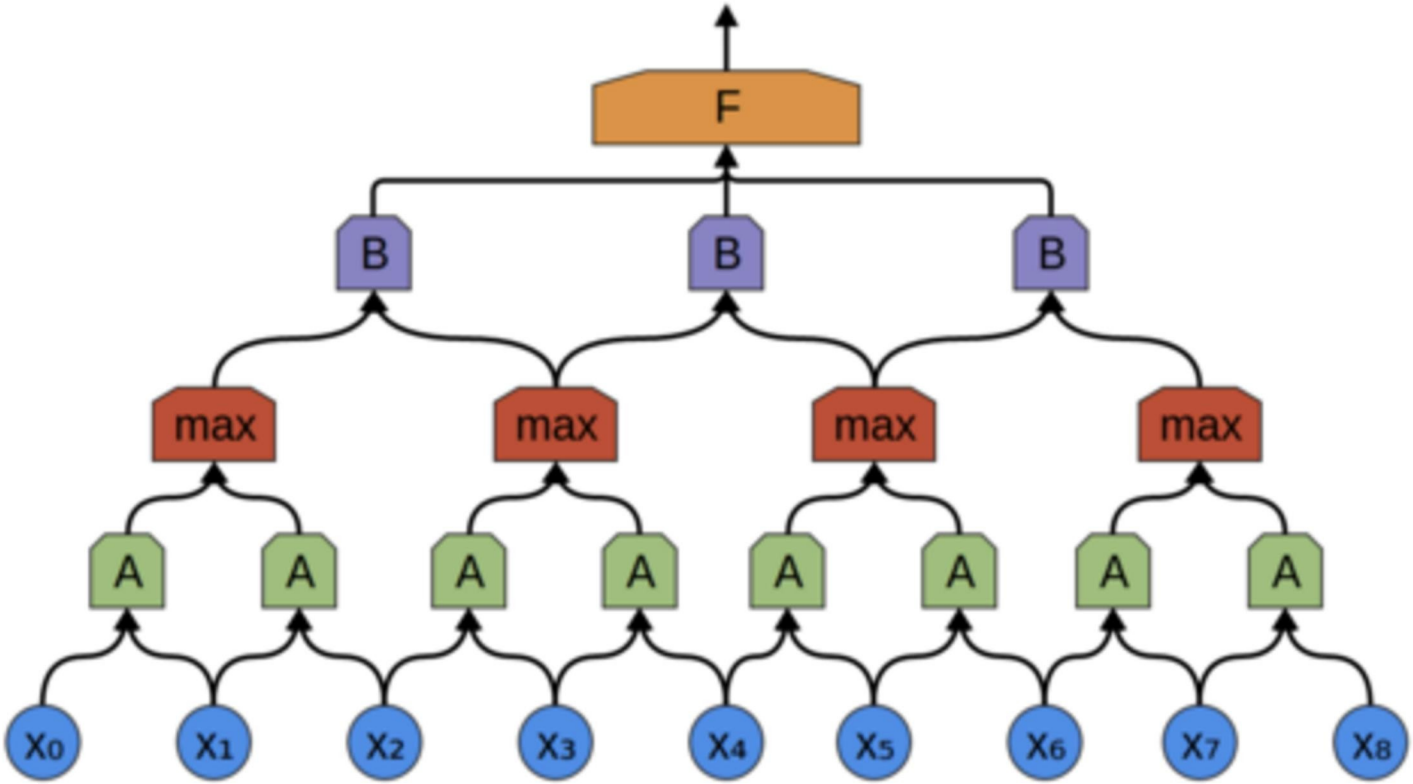
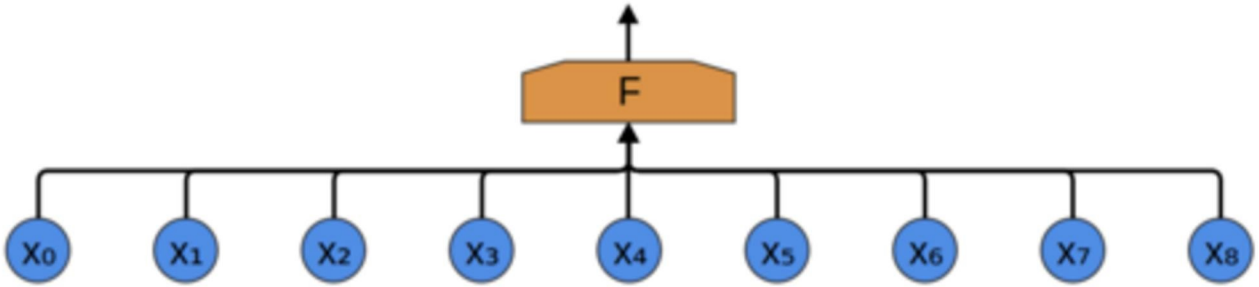


Figure credits: Christopher Olah

neural network v5: convolutional NN

Feed-forward NN



Convolutional NN

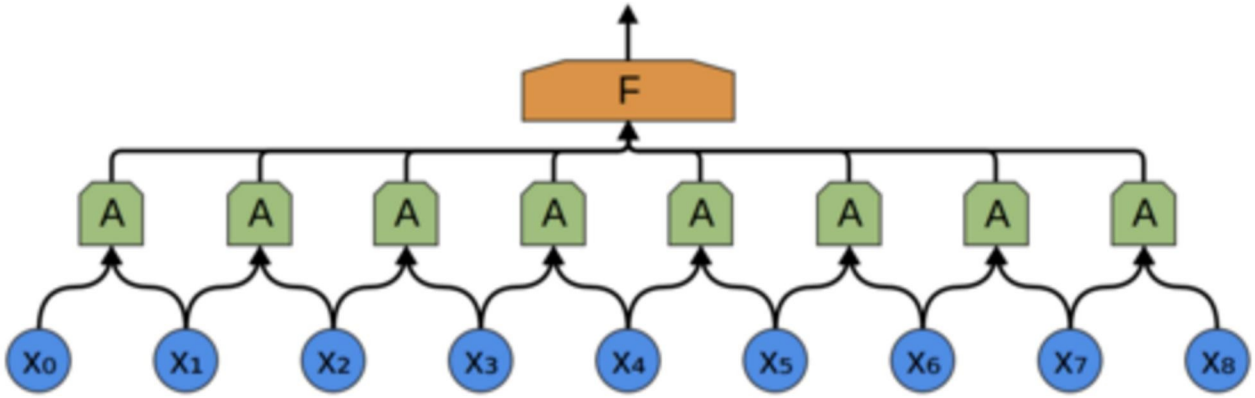
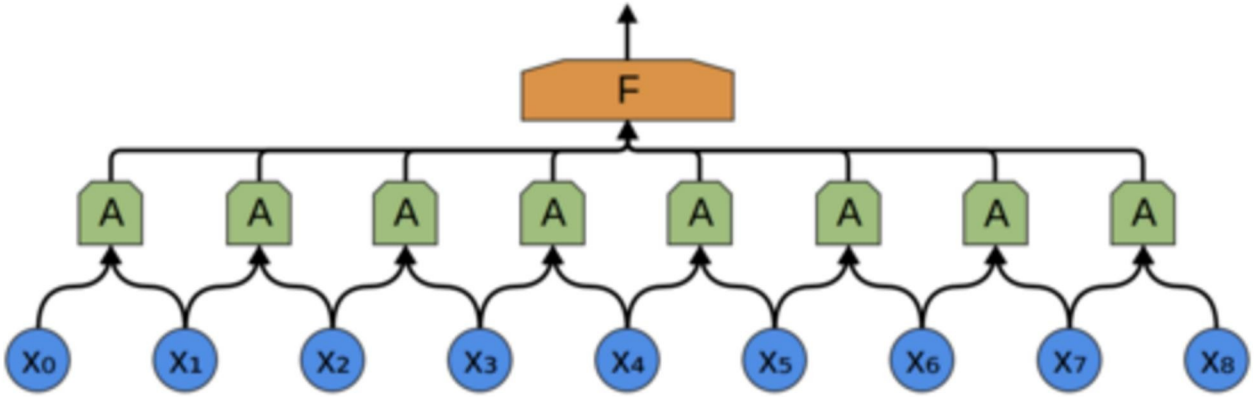


Figure credits: Christopher Olah

neural network v5: convolutional NN

Convolutional layer



convolutional layer 2

convolutional layer 1

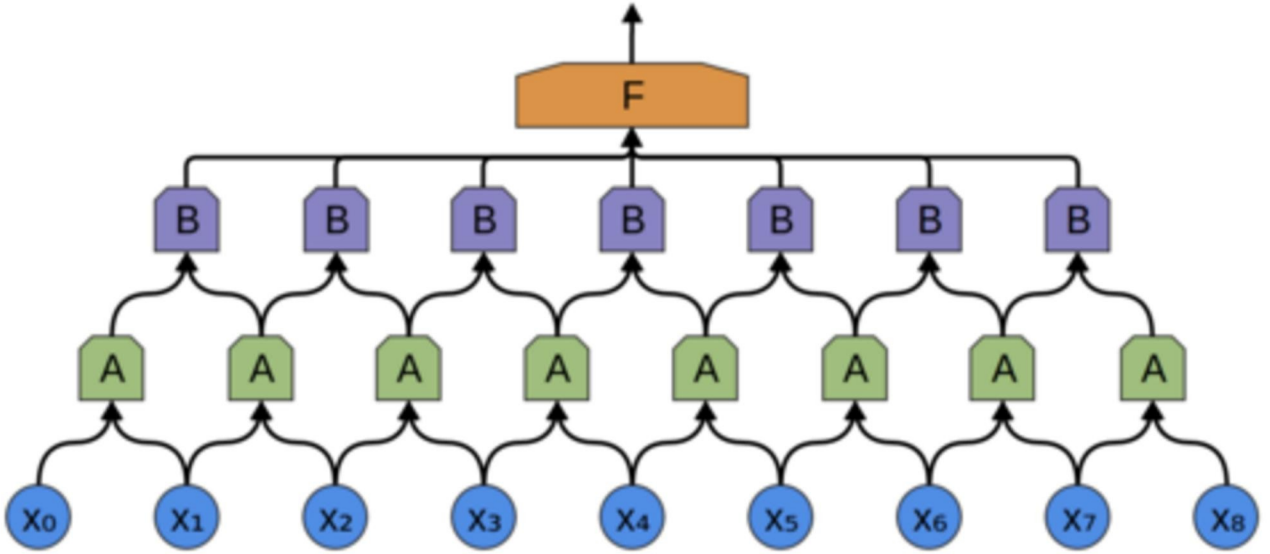


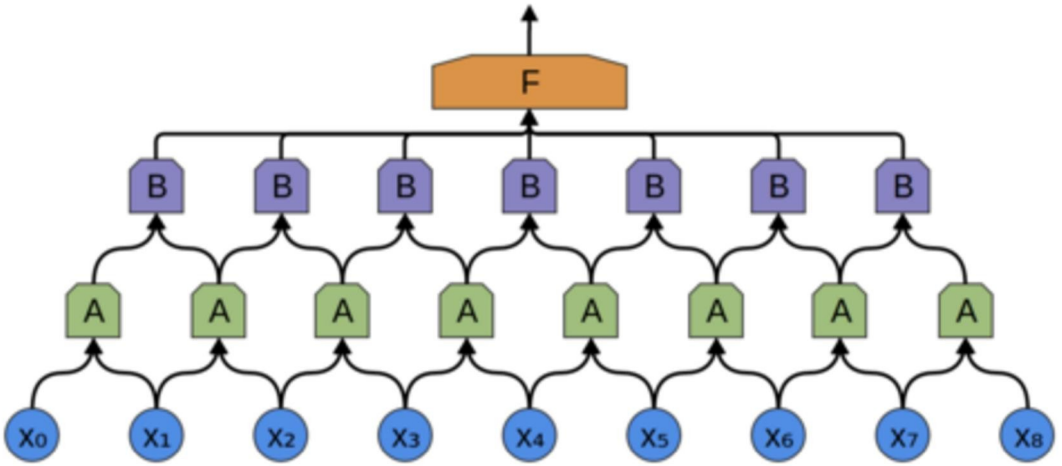
Figure credits: Christopher Olah

Do we share parameters of different convolutions?
In the same layer? In different layers?

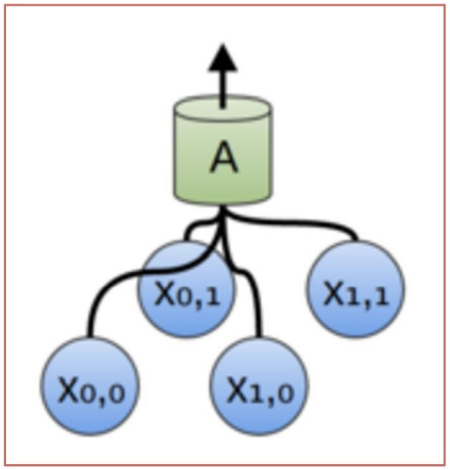
neural network v5: convolutional NN

convolutional layer 2

convolutional layer 1



2D convolutions



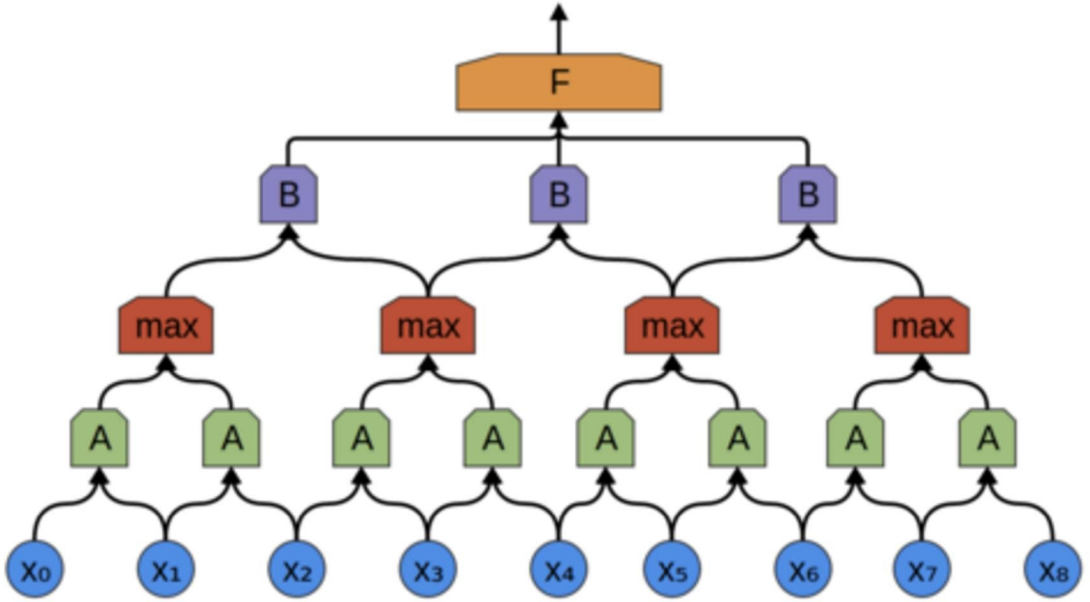
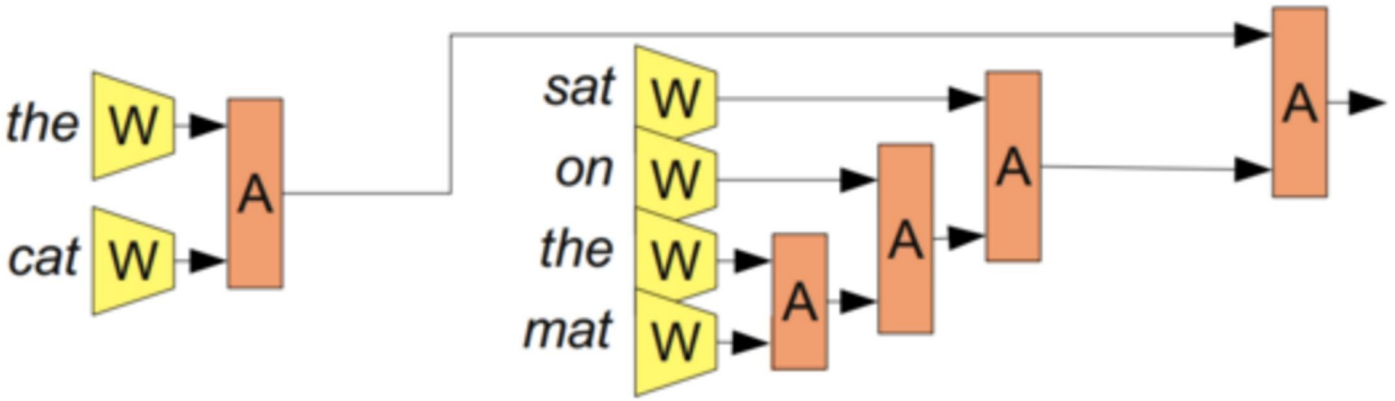


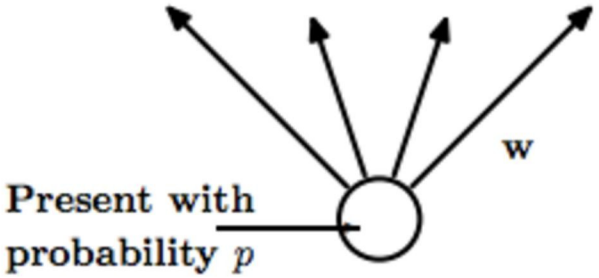
Figure credits: Christopher Olah

neural network v5.1: recursive NNs

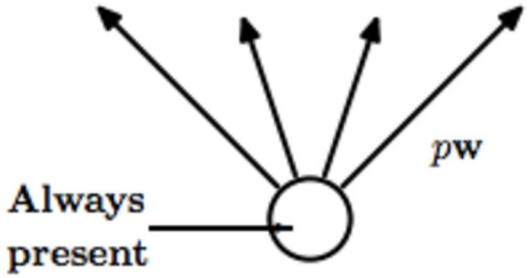


(From Bottou (2011))

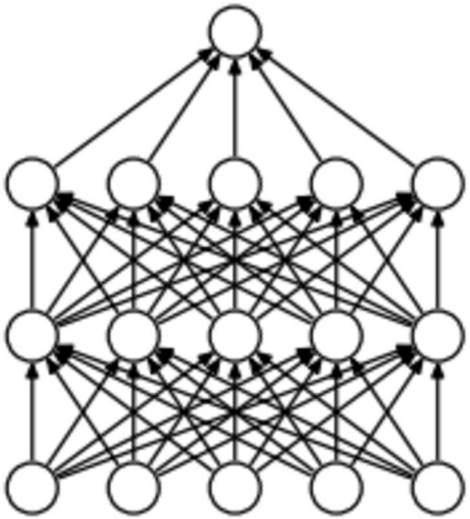
neural network v6: dropout



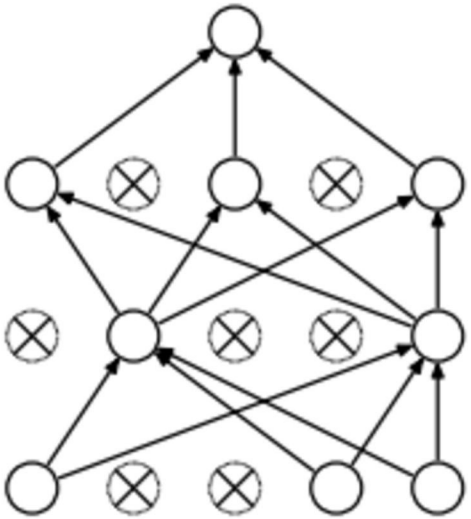
(a) At training time



(b) At test time



(a) Standard Neural Net



(b) After applying dropout.

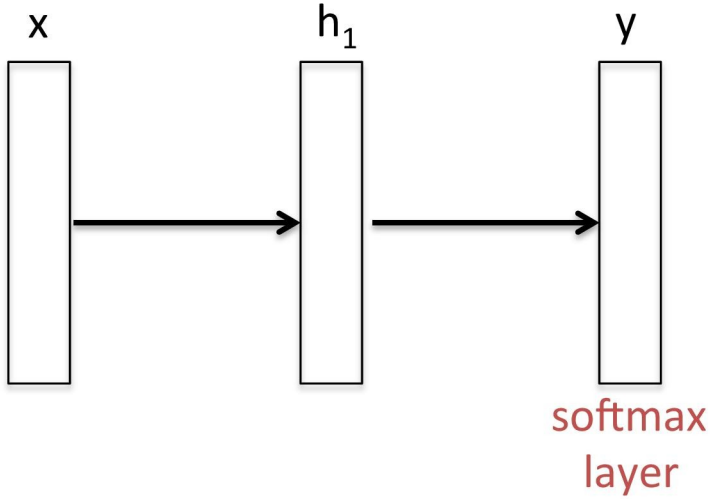
Agenda

- Big picture
- Why deep learning?
- Building blocks of a deep neural network
- **How to train deep neural networks**
- Important results

How to train NN models?

- argmax_l f(d, l) only tells us which label to predict.
- Supervised learning (need input/output pairs)
- Loss function: e.g., cross-entropy between empirical distribution and model distribution

$$- \log p(l^* | d) = - \log \left(e^{f(d, l^*)} / \Sigma_l e^{f(d, l)} \right)$$



Regression problems?

Mean square error

$E[(y - y^*)^2]$

How to optimize the loss?

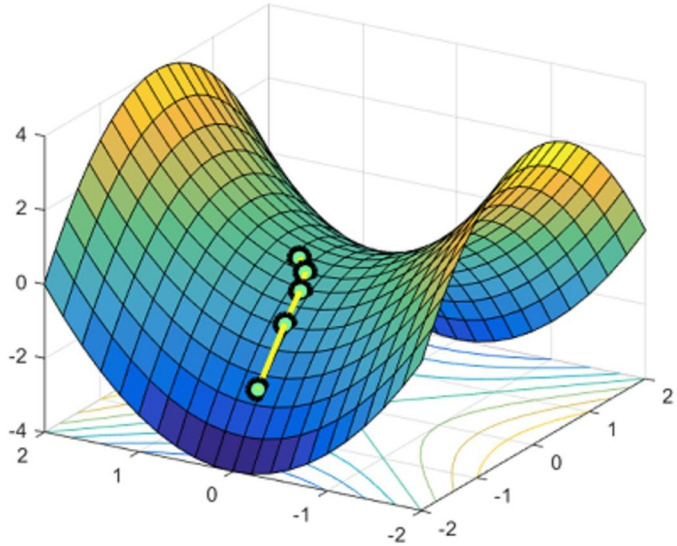
- Stochastic gradient descent

for $i = 1, 2, \ldots$

Pick random training example t and compute:

$$\boldsymbol{g}^{(i)} = \left.\frac{\partial \mathcal{L}(x_t, y_t)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(i)}}$$

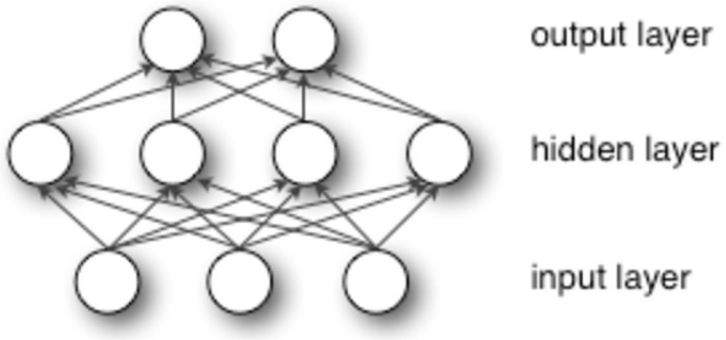
$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \boldsymbol{g}^{(i)}$$

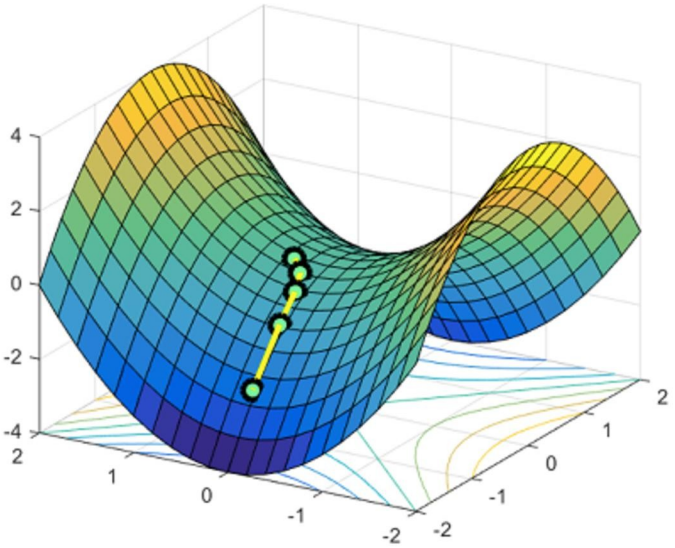


How to optimize the loss?

- Parameter initialization
  - Break the symmetry
  - Use small values
  - Random restarts

  - Popular choice: uniform with mean=zero and variance = 1 / size of previous layer





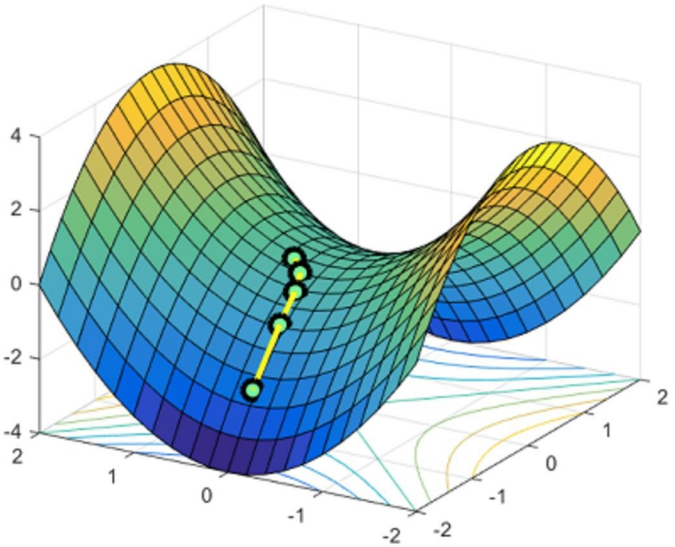
http://andyljones.tumblr.com/post/110998971763/an-explanation-of-xavier-initialization

How to optimize the loss?

- Other optimization methods
  - Variants of stochastic gradient descent

    (e.g., averaged SGD, SGD with momentum)

    See http://research.microsoft.com/pubs/192769/tricks-2012.pdf
  - Adagrad
  - Adam
  - Adadelta



How to optimize the loss?

- Computing gradients: the hard way
  - Analytically derive the expression that represents the gradient with respect to each input.
  - Compute that expression.
- Computing gradients: automatic differentiation
  - Translate the loss function into a sequence of *atomic operations*
  - Hard-code the differentiation of each atomic operation with respect to its parameters is hard-coded.
  - Recursively compute the gradient of the loss function with respect to model parameters using chain rule.

How to optimize the loss: automatic differentiation

$y = \log \sin^2 x$ What is the derivative at x_0 ?

components	***range***	***differential***	***d-range***
$y = f(u) = \log u$	$\mathbb{R}$	$\frac{dy}{du} = \frac{1}{u}$	$\mathbb{R}$
$u = g(v) = v^2$	$\mathbb{R}$	$\frac{du}{dv} = 2v$	$\mathbb{R}$
$v = h(x) = \sin x$	$\mathbb{R}$	$\frac{dv}{dx} = \cos x$	$\mathbb{R}$

$$\left.\frac{dy}{dx}\right|_{x=x_0} = \left.\frac{dy}{du}\right|_{u=g(h(x_0))} \cdot \left.\frac{du}{dv}\right|_{v=h(x_0)} \cdot \left.\frac{dv}{dx}\right|_{x=x_0}$$

How to optimize the loss: automatic differentiation

$$y = \sum_{i=1}^{n} (\mathbf{W} \exp \mathbf{x})_i \quad \text{where} \quad \mathbf{x} \in \mathbb{R}^d \quad \text{and} \quad \mathbf{W} \in \mathbb{R}^{n \times d}$$

components	***range***	***differential***	***d-range***
$y = f(\mathbf{u}) = \sum_{i=1}^{n} u_i$	$\mathbb{R}$	$\frac{\partial y}{\partial \mathbf{u}} = \mathbf{1}$	$\mathbb{R}^{1 \times n}$
$\mathbf{u} = g(\mathbf{v}) = \mathbf{W}\mathbf{v}$	$\mathbb{R}^n$	$\frac{\partial \mathbf{u}}{\partial \mathbf{v}} = \mathbf{W}$	$\mathbb{R}^{n \times d}$
$\mathbf{v} = h(\mathbf{x}) = \exp \mathbf{x}$	$\mathbb{R}^d$	$\frac{\partial \mathbf{v}}{\partial \mathbf{x}} = \operatorname{diag}(\exp \mathbf{x})$	$\mathbb{R}^{d \times d}$

$$\left.\frac{dy}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \underbrace{\left.\frac{dy}{d\mathbf{u}}\right|_{\mathbf{u}=g(h(\mathbf{x}_0))}}_{\in \mathbb{R}^n} \cdot \underbrace{\left.\frac{d\mathbf{u}}{d\mathbf{v}}\right|_{\mathbf{v}=h(\mathbf{x}_0)}}_{\in \mathbb{R}^d} \cdot \underbrace{\left.\frac{d\mathbf{v}}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}}_{\in \mathbb{R}^d}$$

$\mathbb{R}^{1 \times n}$ $\mathbb{R}^{n \times d}$ $\mathbb{R}^{d \times d}$

How to optimize the loss: deep learning libraries

Library	Class	Time (ms)	forward (ms)	backward (ms)
Nervana-neon-fp16	ConvLayer	176	58	118
Nervana-neon-fp32	ConvLayer	211	69	141
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	242	86	156
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	268	94	174
fbfft (Torch)	SpatialConvolutionCuFFT	342	114	227
TensorFlow	conv2d	349	101	248
Chainer	Convolution2D	620	135	484
cudaconvnet2*	ConvLayer	723	176	547
CuDNN[R2] *	cudnn.SpatialConvolution	810	234	576
Caffe	ConvolutionLayer	823	355	468
Torch-7 (native)	SpatialConvolutionMM	878	379	499
CL-nn (Torch)	SpatialConvolutionMM	963	388	574
Caffe-CLGreenTea	ConvolutionLayer	2857	616	2240

https://github.com/soumith/convnet-benchmarks/

Also see: CMU's locally grown library at https://github.com/clab/cnn

Agenda

- Big picture
- Why deep learning?
- Building blocks of a deep neural network
- How to train deep neural networks
- **Important results**

Major results: language modeling

Neural Language Model

[Mikolov et al. Interspeech 2011]

Model \ WSJ ASR task	Eval WER
KN5 Baseline	**17.2**
Discriminative LM	**16.9**
Recurrent NN combination	**14.4**

MSR MAVIS Speech System

[Dahl et al. 2012; Seide et al. 2011; following Mohamed et al. 2011]

"The algorithms represent the first time a company has released a deep-neural-networks (DNN)-based speech-recognition algorithm in a commercial product."

15

Acoustic model & training	Recog \ WER	RT03S FSH	Hub5 SWB
GMM 40-mix, BMMI, SWB 309h	1-pass –adapt	**27.4**	**23.6**
DBN-DNN 7 layer x 2048, SWB 309h	1-pass –adapt	**18.5** (–33%)	**16.1** (–32%)
GMM 72-mix, BMMI, FSH 2000h	*k*-pass +adapt	**18.6**	**17.1**

Slide credit: Richard Socher

Major results: image classification

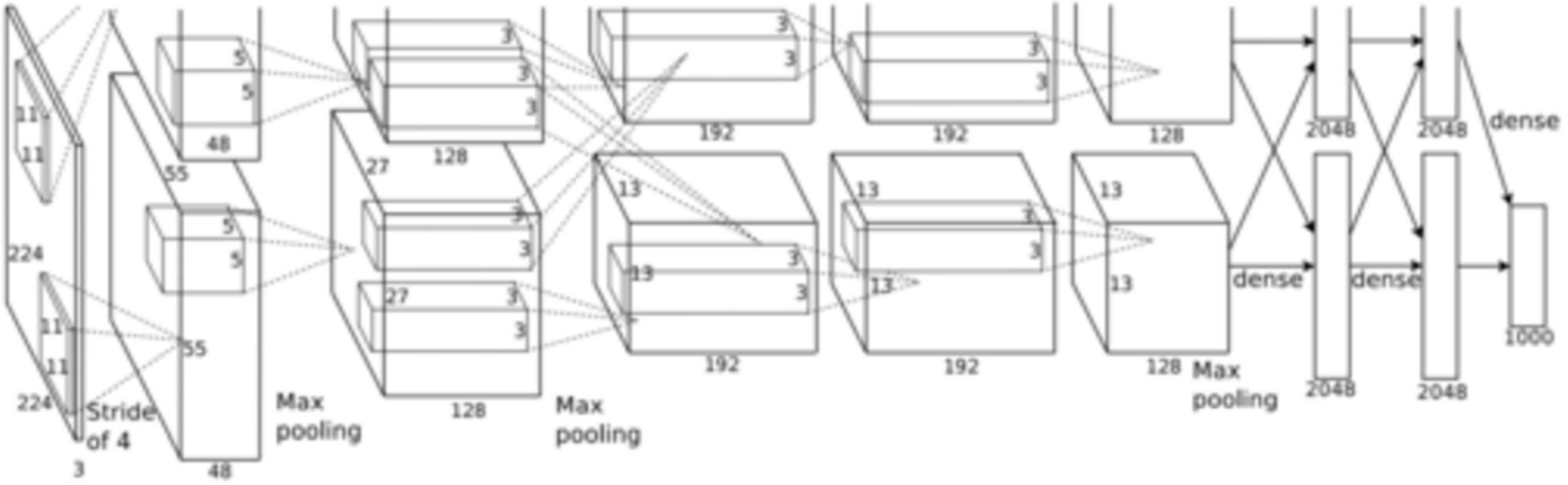
Krizehvsky et al. (2012)

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

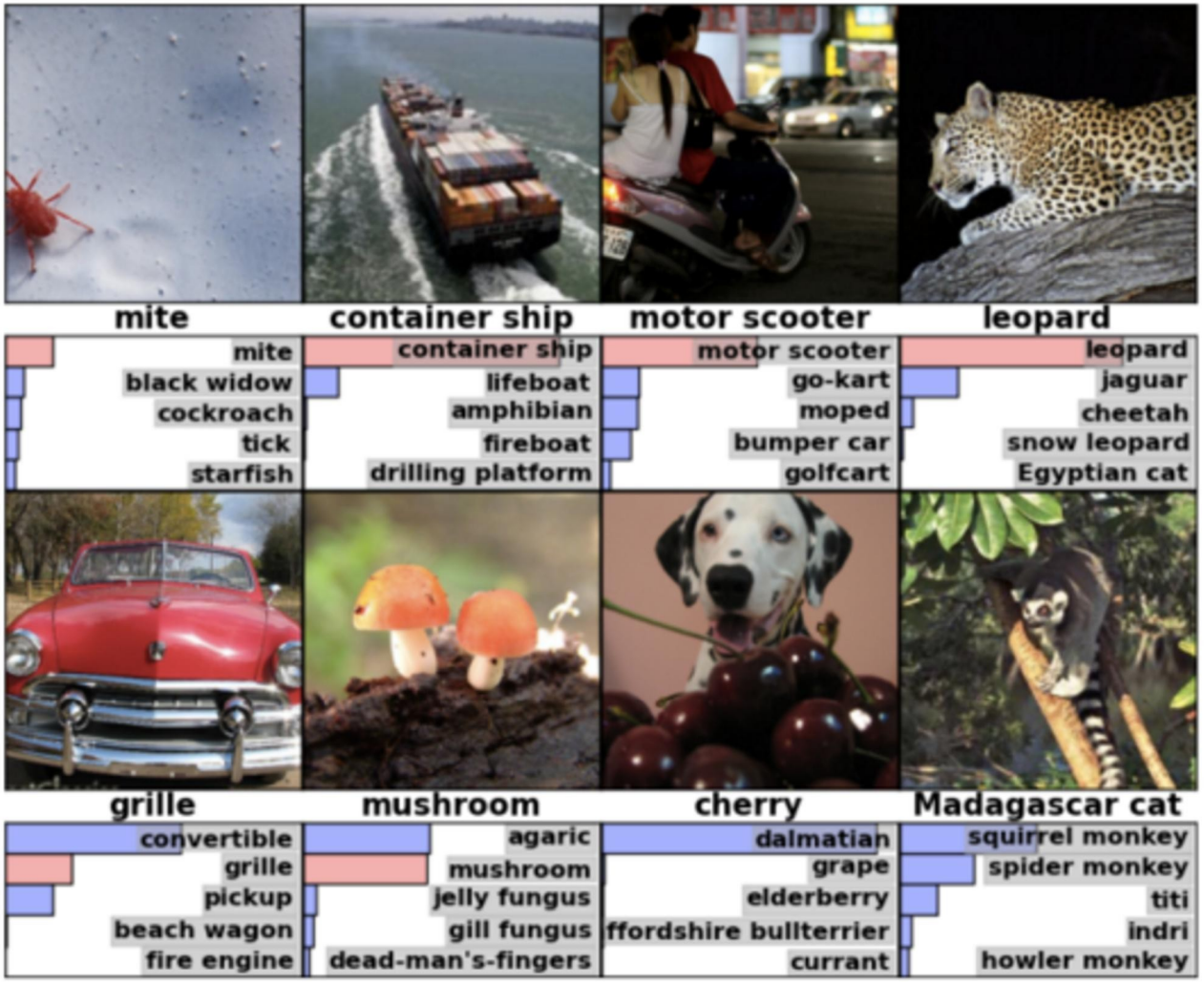
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.



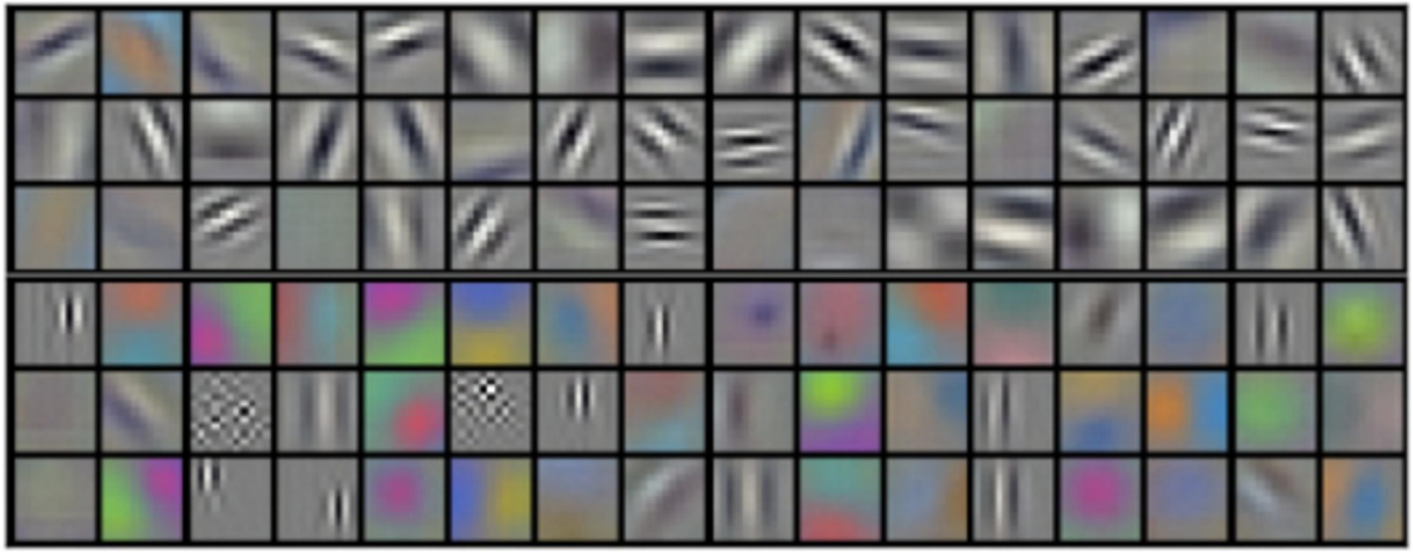
Major results: ImageNet

Krizehvsky et al. (2012): positive and negative examples



Major results: ImageNet

Krizehvsky et al. (2012): sample convolution filters



Major results: speech recognition

Graves et al. (2013)

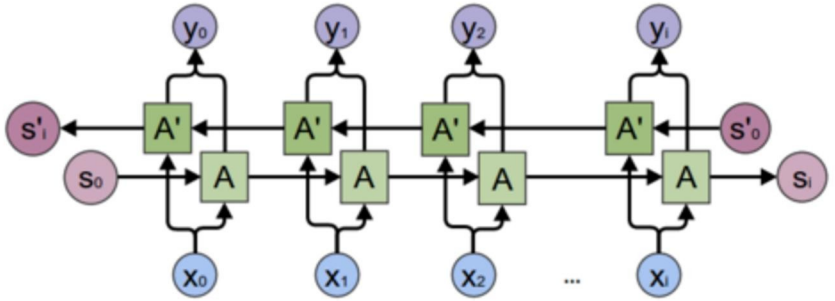
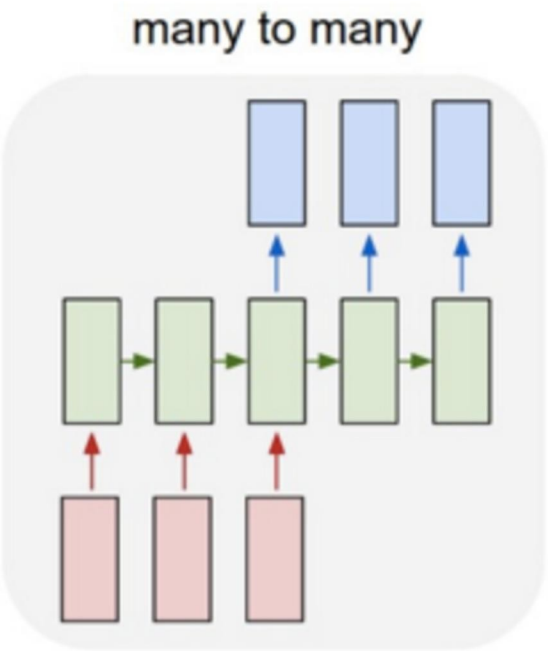


Table 1. TIMIT Phoneme Recognition Results. 'Epochs' is the number of passes through the training set before convergence. 'PER' is the phoneme error rate on the core test set.

NETWORK	WEIGHTS	EPOCHS	PER
CTC-3L-500H-TANH	3.7M	107	37.6%
CTC-1L-250H	0.8M	82	23.9%
CTC-1L-622H	3.8M	87	23.0%
CTC-2L-250H	2.3M	55	21.0%
CTC-3L-421H-UNI	3.8M	115	19.6%
CTC-3L-250H	3.8M	124	18.6%
CTC-5L-250H	6.8M	150	18.4%
TRANS-3L-250H	4.3M	112	18.3%
PRETRANS-3L-250H	**4.3M**	**144**	**17.7%**

Major results: translation

Sutskever et al. (2014)



Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

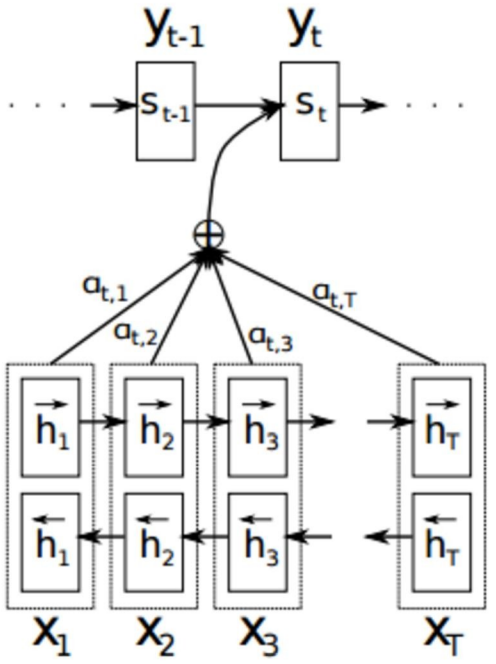
Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
State of the art [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	~45

Table 2: Methods that use neural networks together with an SMT system on the WMT'14 English to French test set (ntst14).

Major results: translation

Bahdanau et al. (2015)



Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

Table 1: BLEU scores of the trained models computed on the test set. The second and third columns show respectively the scores on all the sentences and, on the sentences without any unknown word in themselves and in the reference translations. Note that RNNsearch-50* was trained much longer until the performance on the development set stopped improving. (○) We disallowed the models to generate [UNK] tokens when only the sentences having no unknown words were evaluated (last column).

Major results: dependency parsing

Chen and Manning (2014)

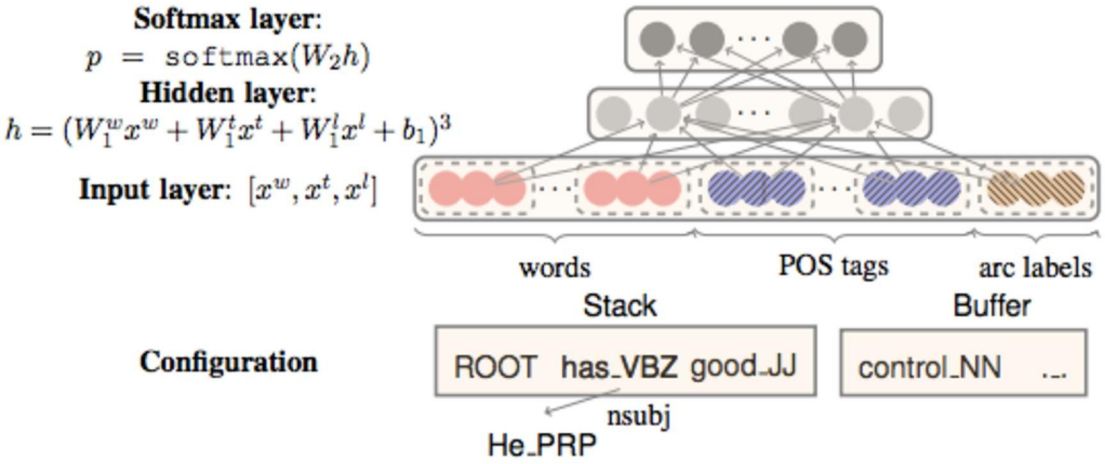


Figure 2: Our neural network architecture.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	**92.0**	**89.7**	**91.8**	**89.6**	**654**

Table 5: Accuracy and parsing speed on PTB + Stanford dependencies.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	**84.0**	82.1	83.0	81.2	6
Our parser	**84.0**	**82.4**	**83.9**	**82.4**	**936**

Table 6: Accuracy and parsing speed on CTB.

Major results: dependency parsing

Dyer et al. (2015)

	Development		Test	
	UAS	LAS	UAS	LAS
S-LSTM	**93.2**	**90.9**	**93.1**	**90.9**
−POS	93.1	90.4	92.7	90.3
−pretraining	92.7	90.4	92.4	90.0
−composition	92.7	89.9	92.2	89.6
S-RNN	92.8	90.4	92.3	90.1
C&M (2014)	92.2	89.7	91.8	89.6

Table 1: English parsing results (SD)

	Development		Test	
	UAS	LAS	UAS	LAS
S-LSTM	**87.2**	**85.9**	**87.2**	**85.7**
−POS	82.8	79.8	82.2	79.1
−pretraining	86.3	84.7	85.7	84.1
−composition	85.8	84.0	85.3	83.6
S-RNN	86.3	84.7	86.1	84.6
C&M (2014)	84.0	82.4	83.9	82.4

Table 2: Chinese parsing results (CTB5)

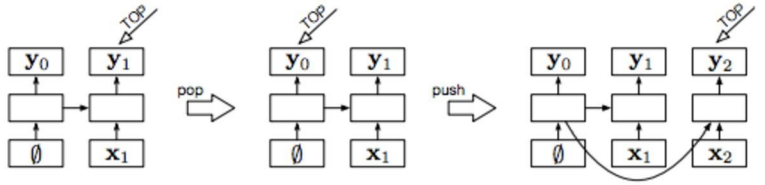


Figure 1: A stack LSTM extends a conventional left-to-right LSTM with the addition of a stack pointer (notated as TOP in the figure). This figure shows three configurations: a stack with a single element (left), the result of a pop operation to this (middle), and then the result of applying a push operation (right). The boxes in the lowest rows represent stack contents, which are the inputs to the LSTM, the upper rows are the outputs of the LSTM (in this paper, only the output pointed to by TOP is ever accessed), and the middle rows are the memory cells (the $\mathbf{c}_t$'s and $\mathbf{h}_t$'s) and gates. Arrows represent function applications (usually affine transformations followed by a nonlinearity), refer to §2.1 for specifics.

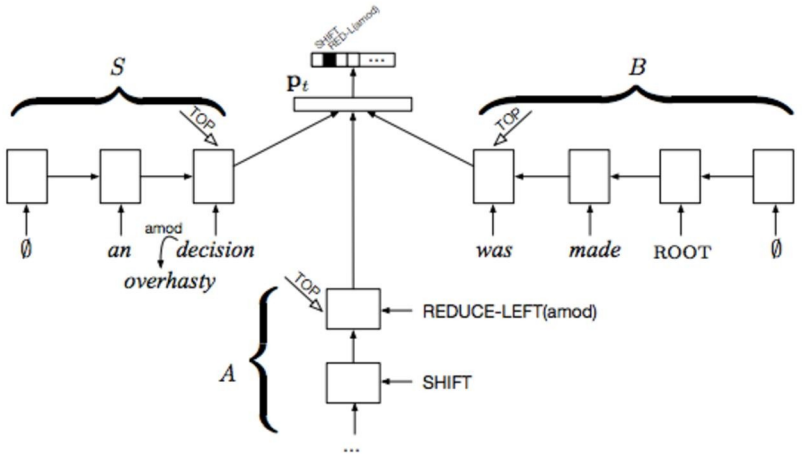


Figure 2: Parser state computation encountered while parsing the sentence "*an overhasty decision was made*." Here S designates the stack of partially constructed dependency subtrees and its LSTM encoding; B is the buffer of words remaining to be processed and its LSTM encoding; and A is the stack representing the history of actions taken by the parser. These are linearly transformed, passed through a ReLU nonlinearity to produce the parser state embedding $\mathbf{p}_t$. An affine transformation of this embedding is passed to a softmax layer to give a distribution over parsing decisions that can be taken.

Pre-trained Models

- Word2vec
  - PyTorch interface, on/off line access
- Elmo/BERT/*Bert*/XLNET
  - Contextual Word Embeddings (transformers)
  - BERT beat 11 standard NLP task (11 Oct 18)
- GPT (2-3)
  - Massive language models

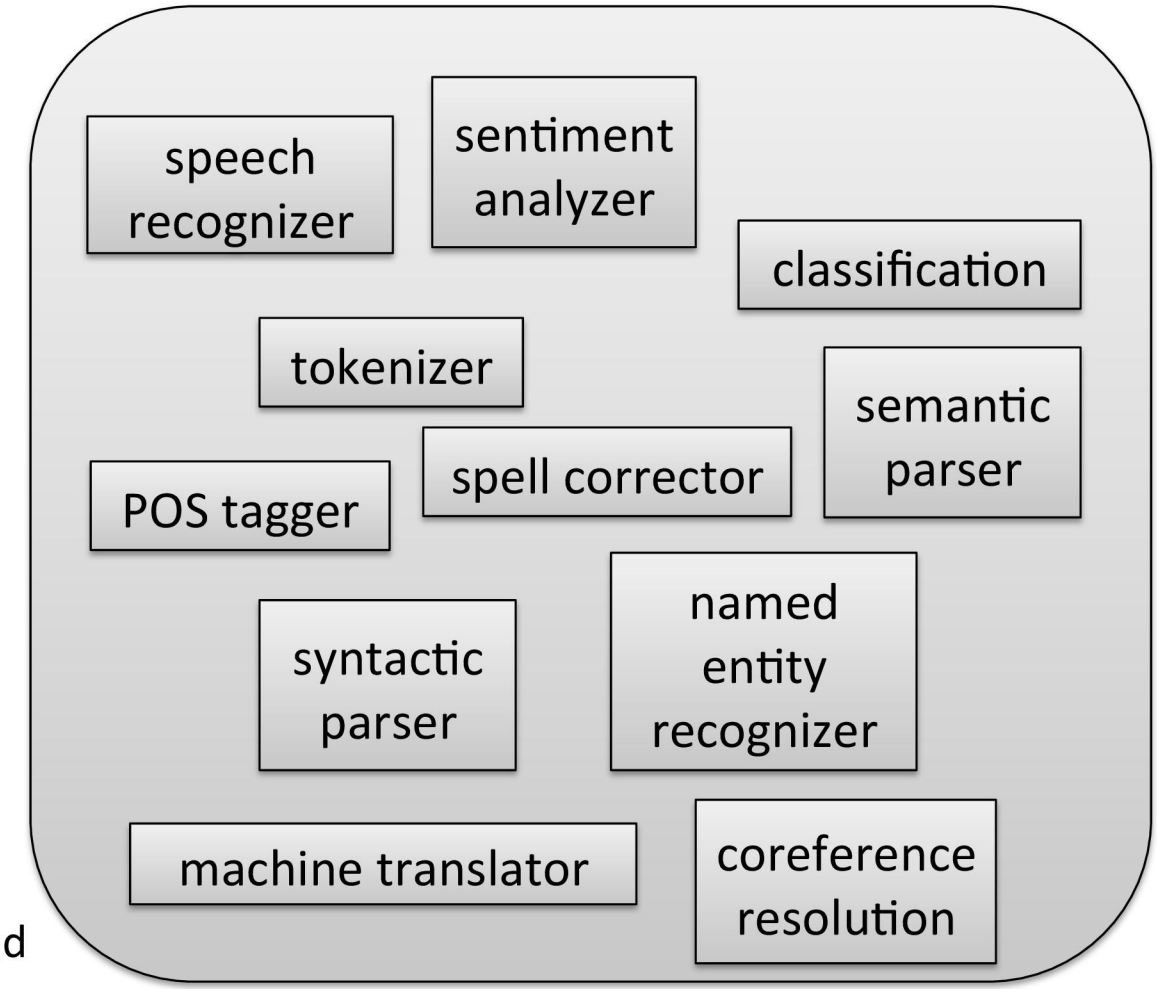
So how do I use Neural Nets?

- Good sample code available
  - https://huggingface.co
  - Code, tutorials, walk throughs
- Most publications include code (and data)
  - not always “good” code but good start
- Books “NN for NLP” but out of date quickly
  - On-line tutorials may be more up to date

BERT for Document Classification

- e.g. Huggingface tutorial
- Include multiple examples
- Step-by-step data preparation
  - Normalization, Tokenization, train/test split
  - (resampling, augmentation) inspections
  - Training, evaluation, tuning
- Where do I get GPU time to do this?
  - Google Colab
  - Buy them from Amazon :-)
  - Start your own GPU company … profit

Open question: can we do without the intermediate linguistic abstractions?



Natural language input signal:

- Web page
- Question
- Search query
- Tweet
- Voice command

Output analysis:

- Question
- Answer
- Command to a robot
- Trending topics

Do we need NLP structure

- Maybe not
  - for easy tasks, in well resource languages
- Maybe still
  - For hard tasks or in low resource cases
- You should test it
  - Testing has final say
- Maybe we should be doing harder tasks …
  - Common sense, understanding, etc,
  - Controlled generation, novel prediction