Machine Learning 10-601/301

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

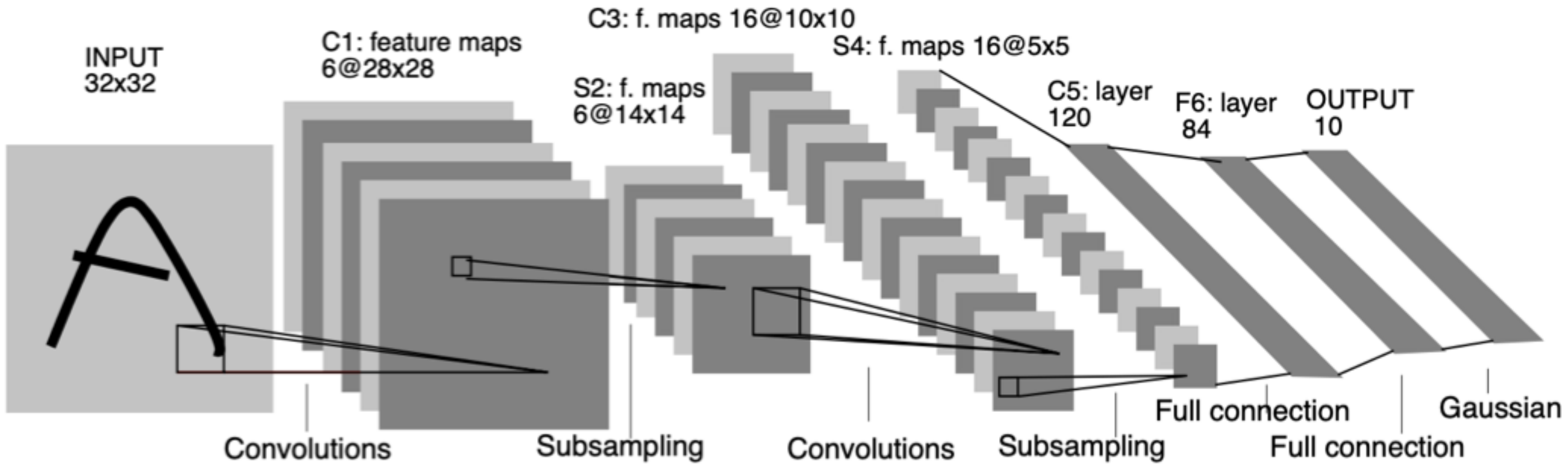
March 115, 2021

This section:
- Convolutional neural nets
- Recurrent neural nets
- LSTMs
- Sequence to sequence models

Reading:
- optional: Mitchell: Chapter 4
- Note Mitchell book now downloadable

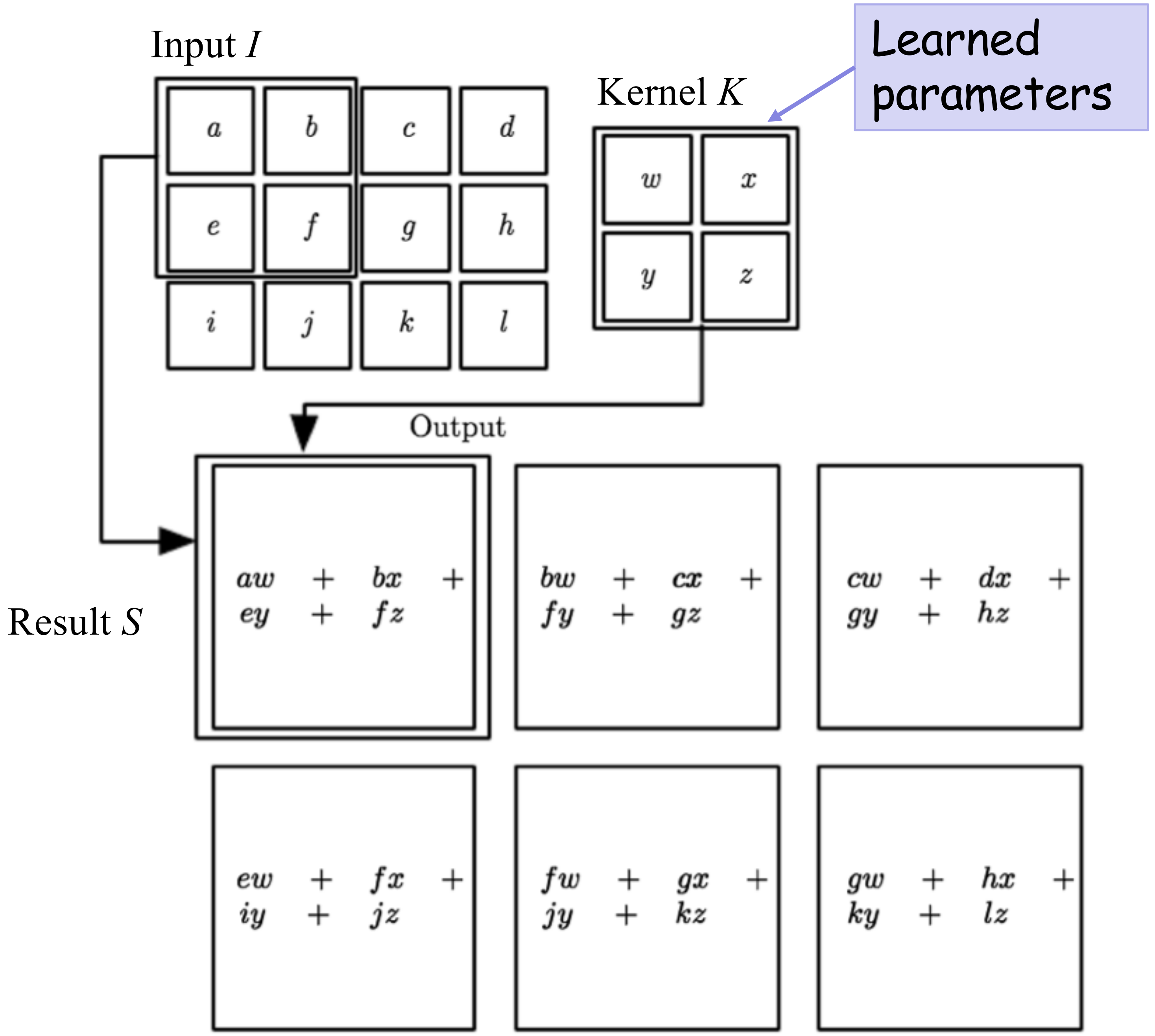
Convolutional Neural Nets

A Convolutional Neural Net for Handwritten Digit recognition: LeNet5* [LeCun, et al., 1998]



* In the 1998 LeNet5 paper output layer was a Gaussian RBF layer, though today we would use Softmax to obtain probabilities as outputs

Convolution Layer



$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

[from Goodfellow et al.]

Convolution : yields invariance to input translation

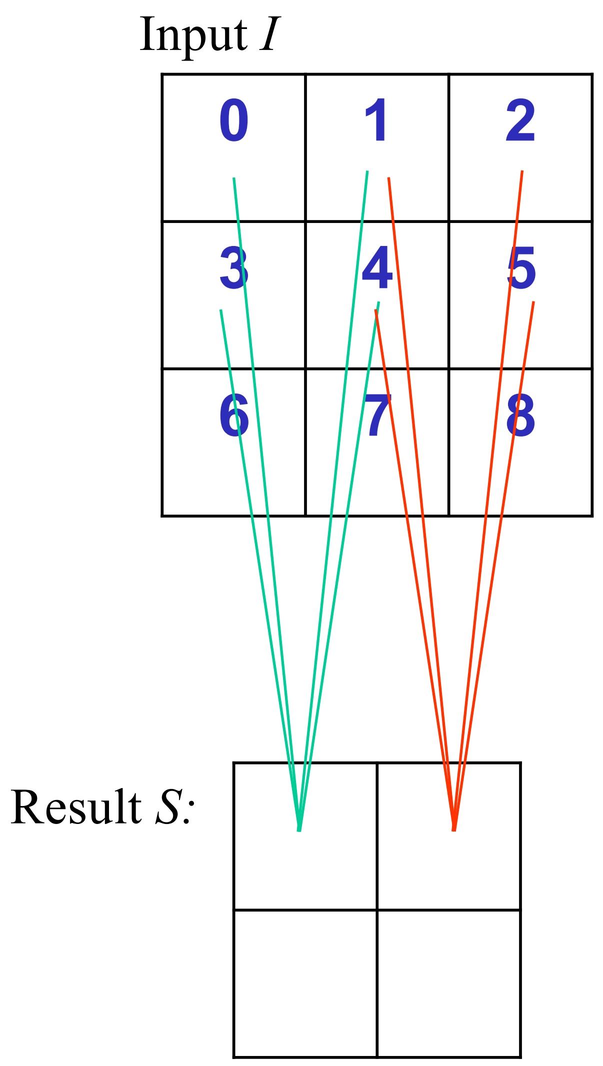
Input I Kernel K Result S

$$\begin{array}{|c|c|c|}\hline 0 & 1 & 2 \\ \hline 3 & 4 & 5 \\ \hline 6 & 7 & 8 \\ \hline \end{array} \; * \; \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 1 & 0 \\ \hline \end{array} \; = \; \begin{array}{|c|c|}\hline 5 & 9 \\ \hline 17 & \\ \hline \end{array}$$

Trained parameters Output activations

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

Convolution as parameter sharing

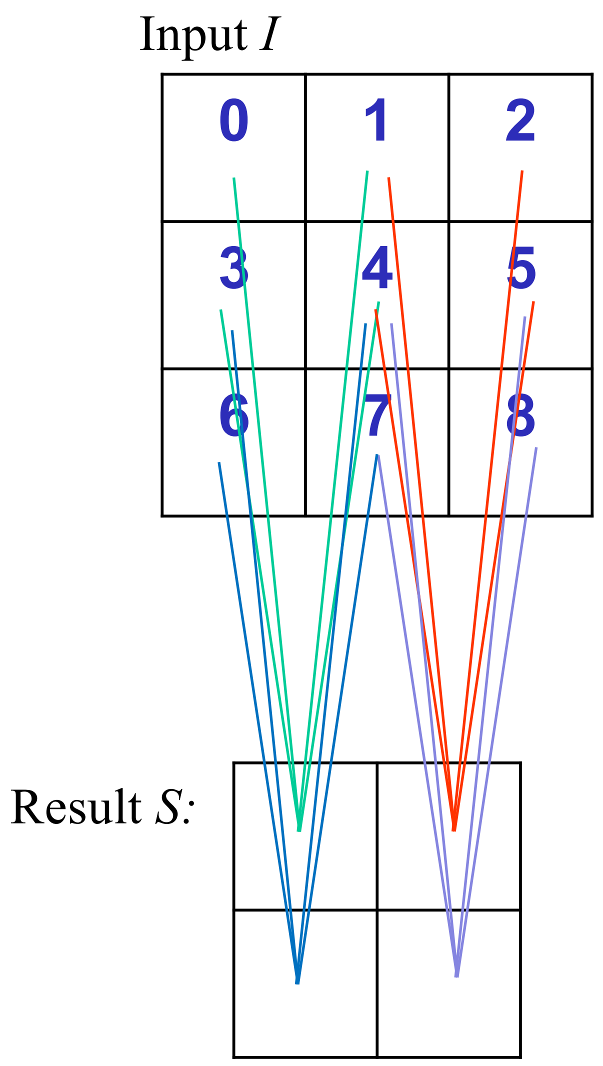


$$* \quad \overset{\text{Kernel } K}{\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}} = \overset{\text{Result } S}{\begin{bmatrix} 5 & 9 \\ 17 & 21 \end{bmatrix}}$$

Trained parameters   Output activations

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

Convolution as parameter sharing



$*$

Kernel K

1	2
1	0

=

Result S

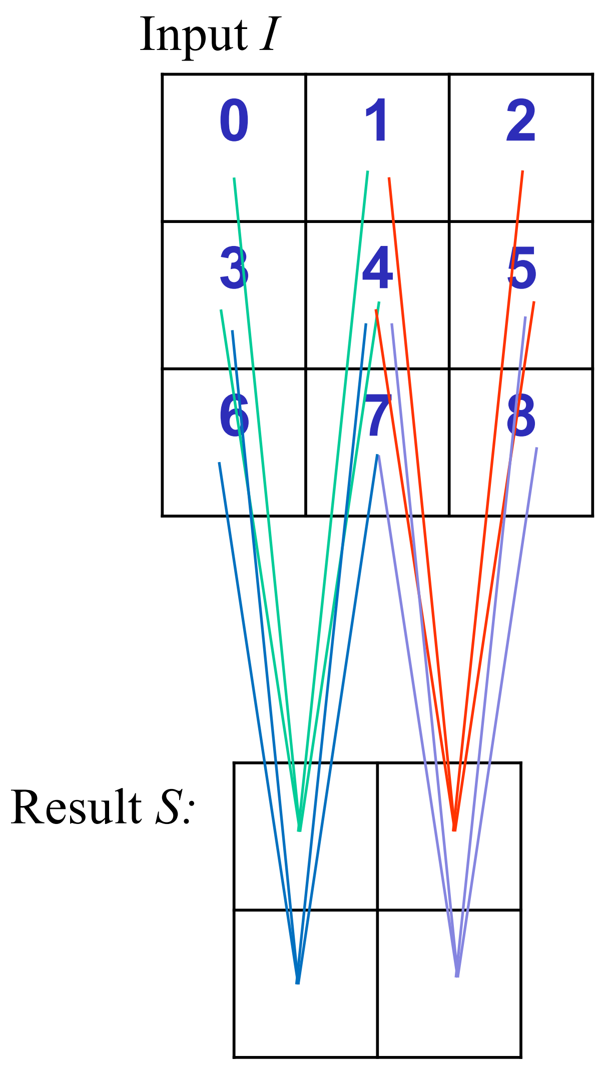
5	9
17	21

Trained parameters

Output activations

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

How do we calculate gradient components $\frac{\partial J(\theta)}{\partial K(m,n)}$?



$$* \quad \begin{array}{c} \text{Kernel } K \\ \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 1 & 0 \\ \hline \end{array} \end{array} \quad = \quad \begin{array}{c} \text{Result } S \\ \begin{array}{|c|c|} \hline 5 & 9 \\ \hline 17 & 21 \\ \hline \end{array} \end{array}$$

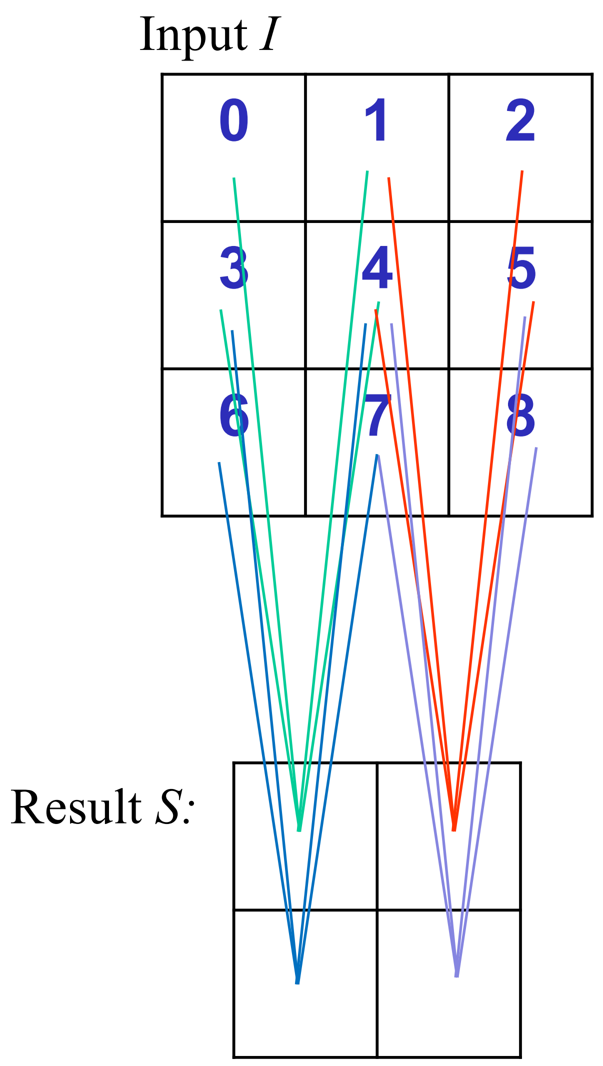
Trained parameters

Output activations

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

How do we calculate gradient components $\frac{\partial J_d(\theta)}{\partial K(m,n)}$ for training example d?

$\theta = \{K(0,0), K(0,1), K(1,0) \dots K(M-1, N-1)\}$



*****

Kernel K

1	2
1	0

=

Result S

5	9
17	21

Trained parameters

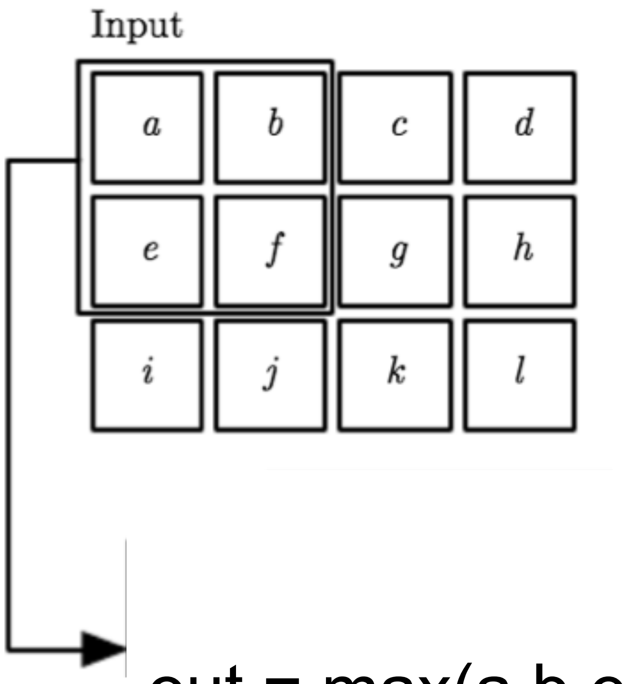
Output activations

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

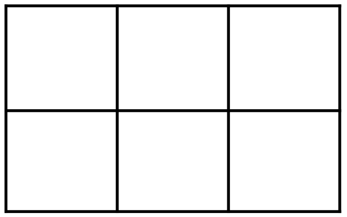
$$\frac{\partial J_d(\theta)}{\partial K(m,n)} = \sum_{(i,j) \in \text{output map } S} \frac{\partial J_d(\theta)}{\partial S(i,j)} \frac{\partial S(i,j)}{\partial K(m,n)}$$

$$= \sum_{(i,j) \in \text{output map } S} \frac{\partial J_d(\theta)}{\partial S(i,j)} I(i+m, j+n)$$

Maxpool Layer



out = max(a,b,e,f)

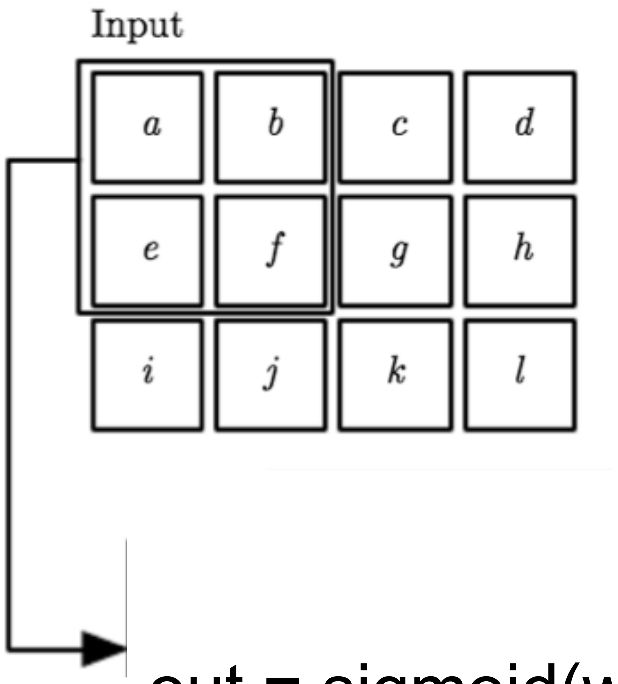


What is derivative of out with respect to inputs?

e.g., if a=2,b=3,e=2,f=4

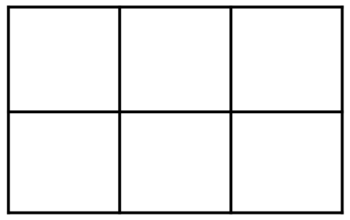
[from Goodfellow et al.]

Subsampling Layer In LeNet



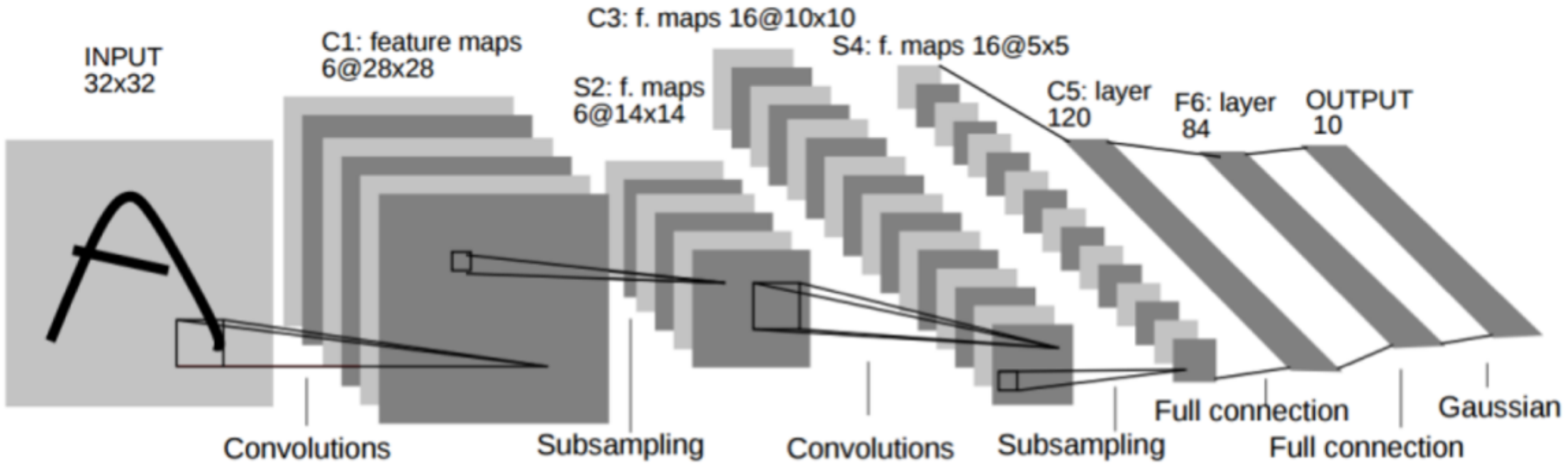
out = sigmoid(w_0 + w_1(a+b+e+f))

What is derivative of out with respect to inputs?

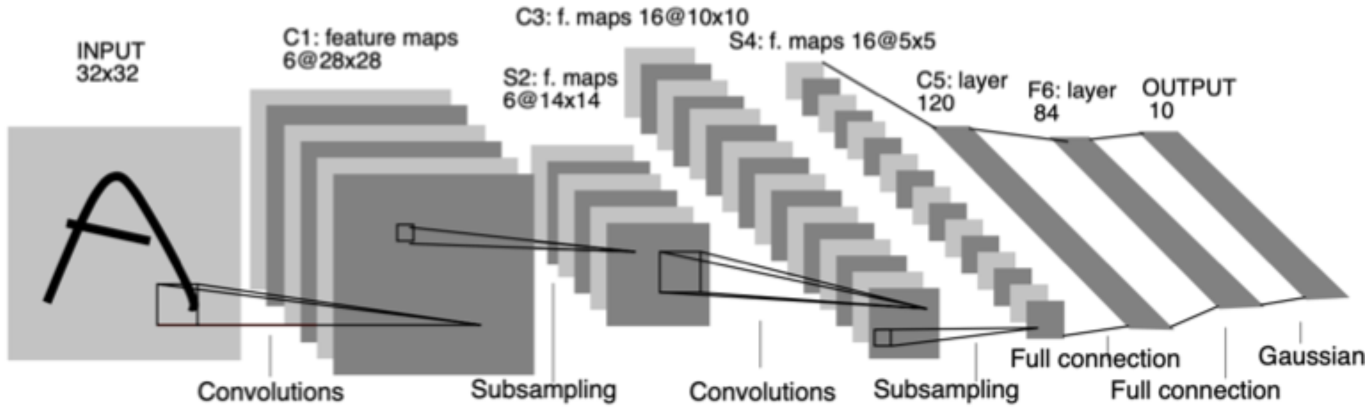


[from Goodfellow et al.]

A Convolutional Neural Net for Handwritten Digit recognition: LeNet5*



LeNet5 details



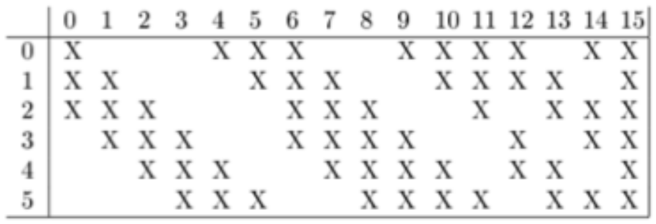
- [LeCun et al., 1998]

- C1 is a **<u>convolution layer</u>** using 6 distinct 5x5 kernels, stride 1, creating 6 distinct channels of 28x28 feature maps, each based on one kernel. Total trainable parameters:
  156

- S2 is a **<u>subsampling layer</u>**, creating 6 channels, one each from the corresponding channel of C1. Values are based on a 2x2 input kernel, stride 2 (so no overlap) and the value output to the S2 map is out = sigmoid($w_0+w_1(x_1+x_2+x_3+x_4)$), where x_i's are the four inputs to the 2x2 kernel. Total trainable parameters:

- C3 is a **<u>convolutional layer</u>**, using 16 kernels to produce 16 feature maps. Each kernel is connected to several 5x5 neighborhoods at identical locations in a subset of the 6 channels of S2 as shown below. Total trainable parameters: 1,516

- S4 subsamples C3, just like S2 samples C1

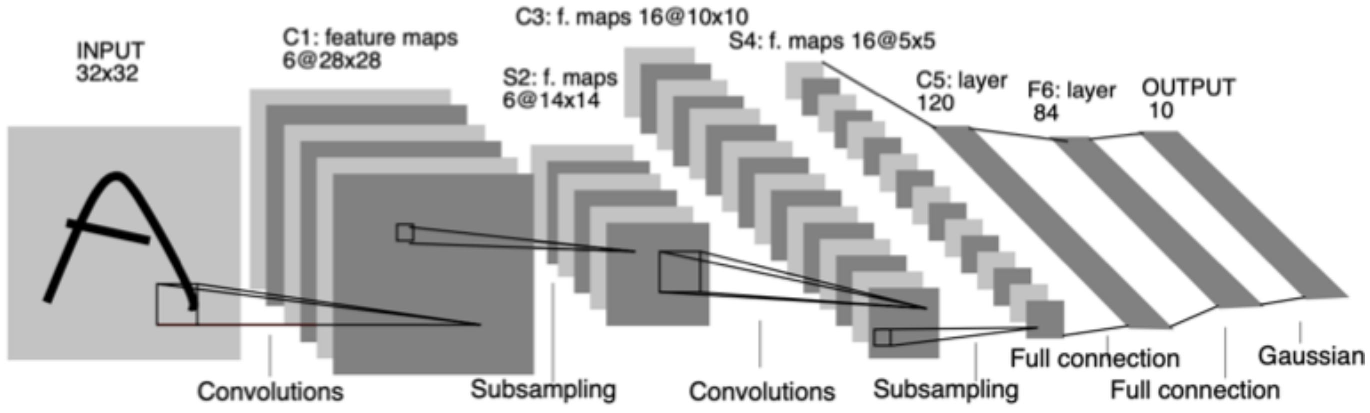


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

LeNet5 details

- [LeCun et al., 1998]



- C1 is a **convolution layer** using 6 distinct 5x5 kernels, stride 1, creating 6 distinct channels of 28x28 feature maps, each based on one kernel. Total trainable parameters: 156

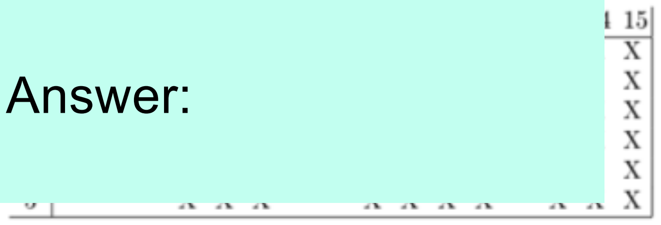
- S2 is a **subsampling layer**, creating 6 channels, one each from the corresponding channel of C1. Values are based on a 2x2 input kernel, stride 2 (so no overlap) and the value output to the S2 map is out = sigmoid(w_0+w_1(x_1+x_2+x_3+x_4)), where x_i's are the four inputs to the 2x2 kernel. Total trainable parameters:

- C3 is a **convolutional layer**, using 16 kernels to produ nel is connected to several 5x5 neighborhoods at identical channels of S2 as shown below. Total trainable param

- S4 subsamples C3, just like S2 samples C1

Poll Question 2:
How many total trainable parameters are in layer S2?

Answer:

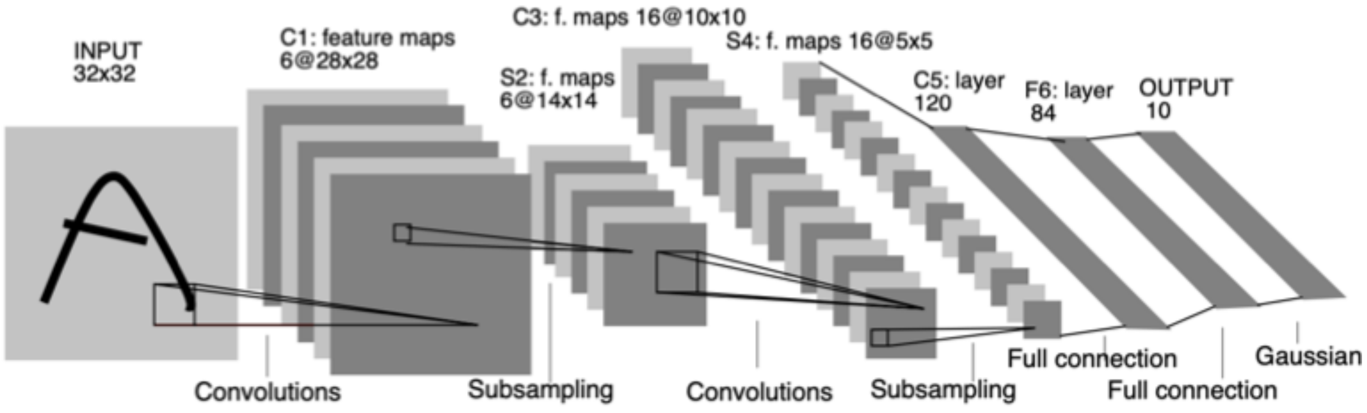


15
X
X
X
X
X
X

TABLE I
Each column indicates which feature map in S2 are combined by the units in a particular feature map of C3.

LeNet5 details

- [LeCun et al., 1998]

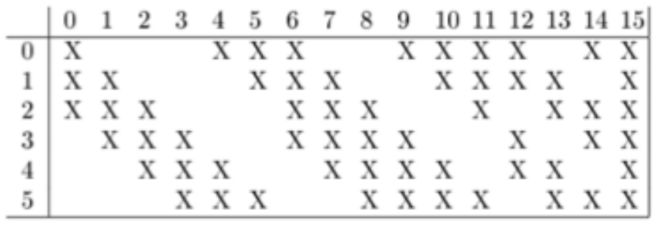


- C1 is a **convolution layer** using 6 distinct 5x5 kernels, stride 1, creating 6 distinct channels of 28x28 feature maps, each based on one kernel. Total trainable parameters: 156

- S2 is a **subsampling layer**, creating 6 channels, one each from the corresponding channel of C1. Values are based on a 2x2 input kernel, stride 2 (so no overlap) and the value output to the S2 map is out = sigmoid($w_0+w_1(x_1+x_2+x_3+x_4)$), where x_i's are the four inputs to the 2x2 kernel. Total trainable parameters: 12

- C3 is a **convolutional layer**, using 16 kernels to produce 16 feature maps. Each kernel is connected to several 5x5 neighborhoods at identical locations in a subset of the 6 channels of S2 as shown below. Total trainable parameters: 1,516

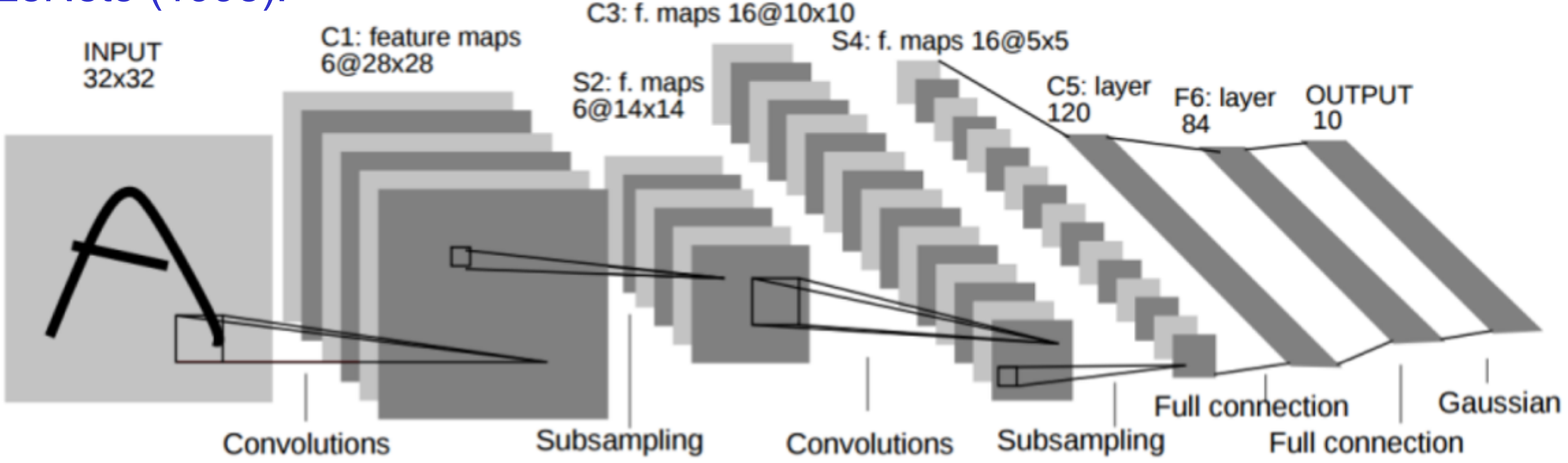
- S4 subsamples C3, just like S2 samples C1



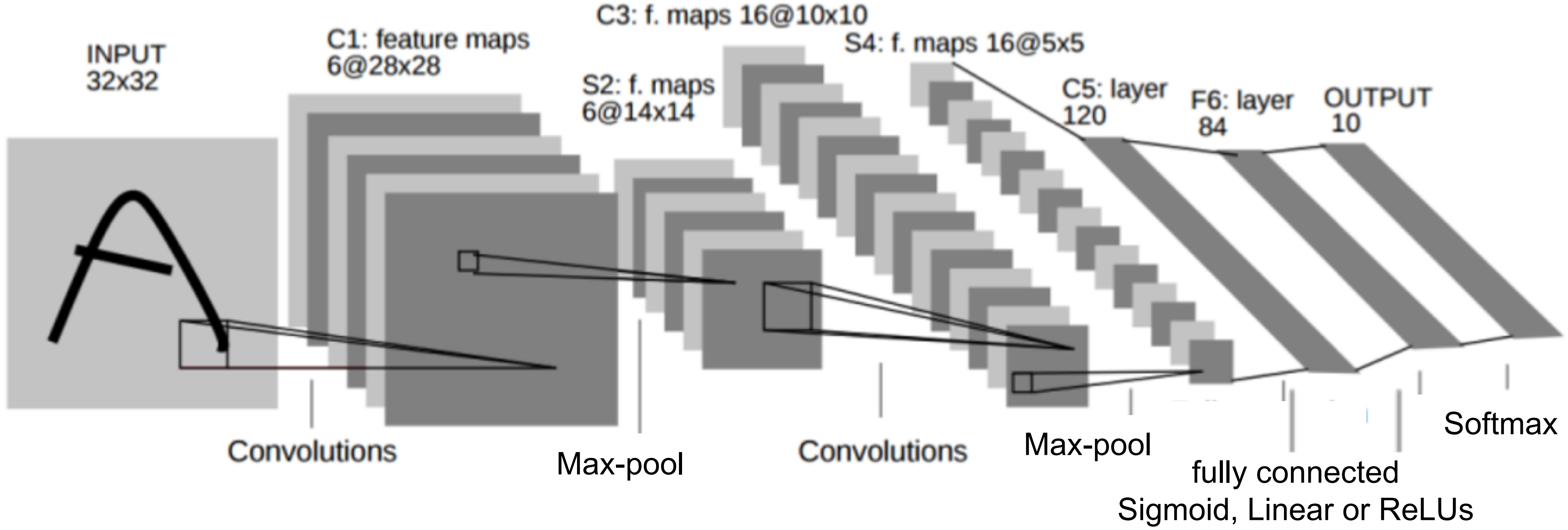
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

LeNet5 (1998):



More typical 2021 Convolutional Net:



Softmax Layer: Predict **Probability Distribution** over discrete-valued labels

- Logistic Regression: when Y has two possible values

$$P(Y=1|X=\langle X_1,\ldots X_n\rangle)=\frac{1}{1+\exp(w_0+\sum_i w_iX_i)}$$

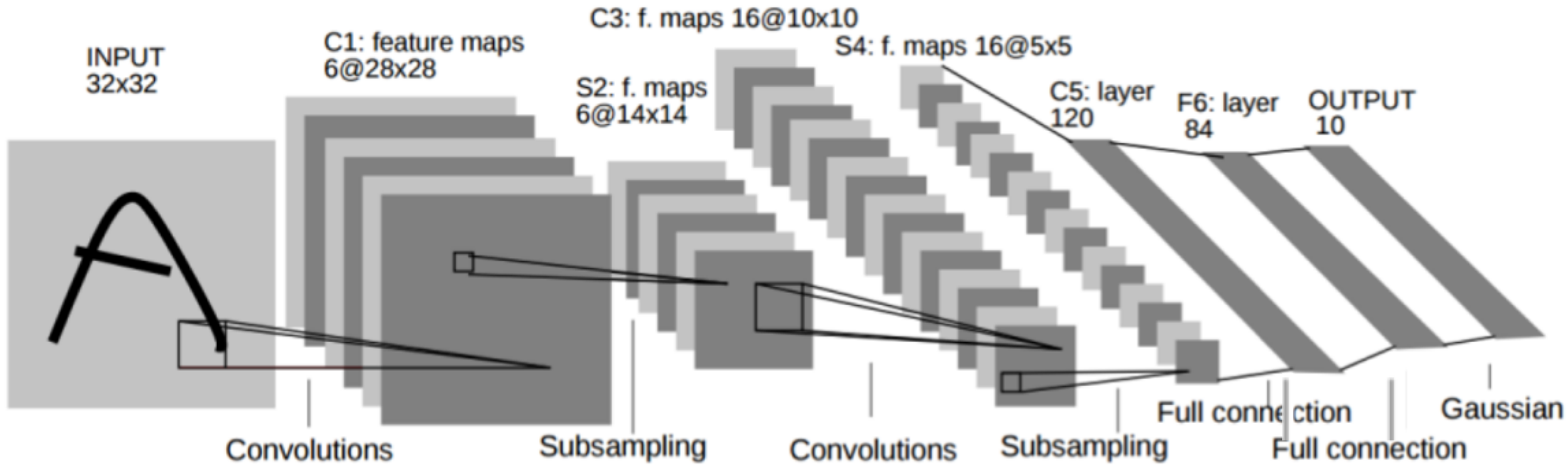
$$P(Y=0|X=\langle X_1,\ldots X_n\rangle)=\frac{\exp(w_0+\sum_i w_iX_i)}{1+\exp(w_0+\sum_i w_iX_i)}$$

- Softmax: when *Y* has *R* values $\{y_1 \ldots y_R\}$, then learn *R* sets of weights to predict *R* output probabilities

$$P(Y=y_k|X)=\frac{exp(w_{k0}+\sum_i w_{ki}X_i)}{\sum_{j=1}^{R} exp(w_{j0}+\sum_i w_{ji}X_i)}$$

Note neural network now has R outputs instead of just 1

A Convolutional Neural Net for Handwritten Digit recognition: LeNet



- Shrinking size of feature maps
- Multiple channels
- LeNet-5 Demos: http://yann.lecun.com/exdb/lenet/index.html
    - Vary scale
    - Vary stroke width
    - Squeeze
    - Noisy-2, Noisy-4

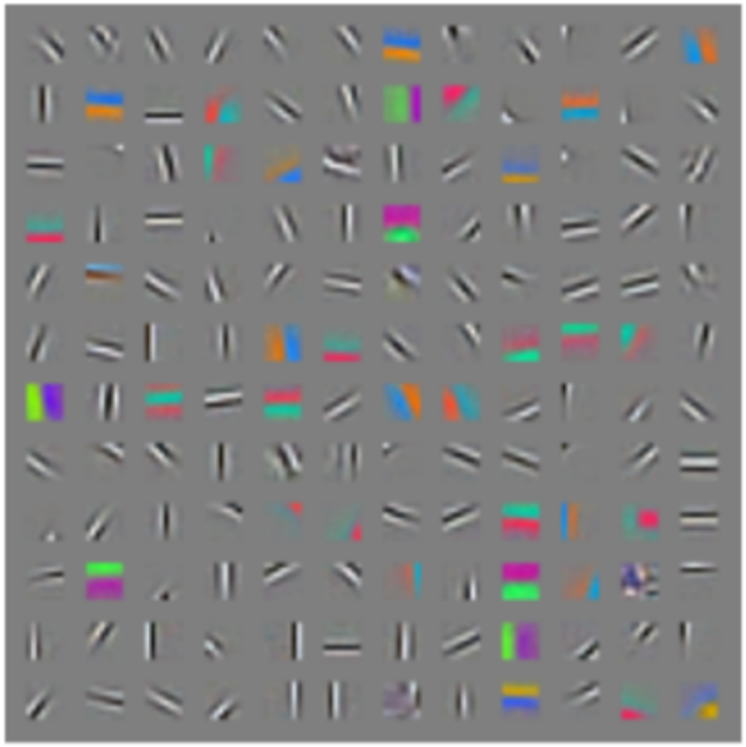
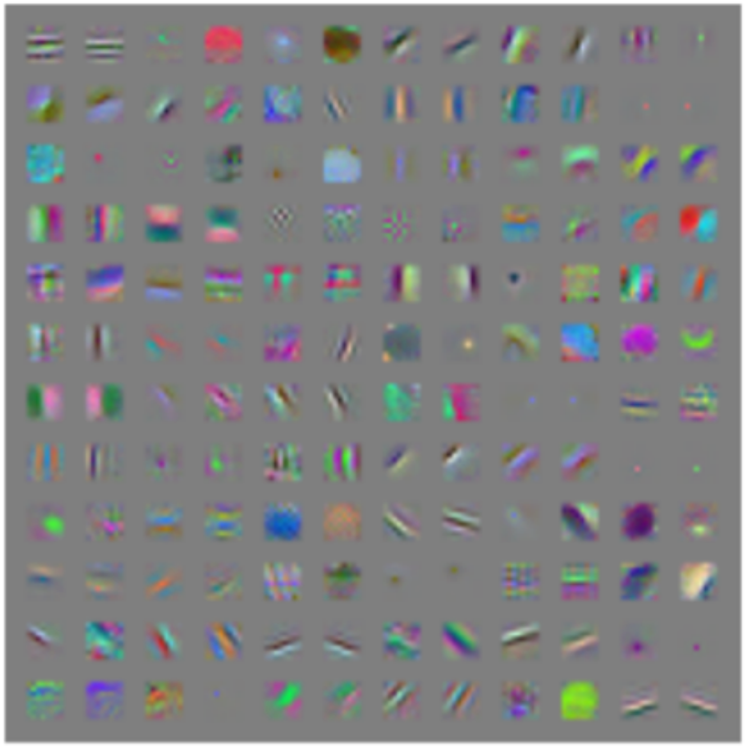
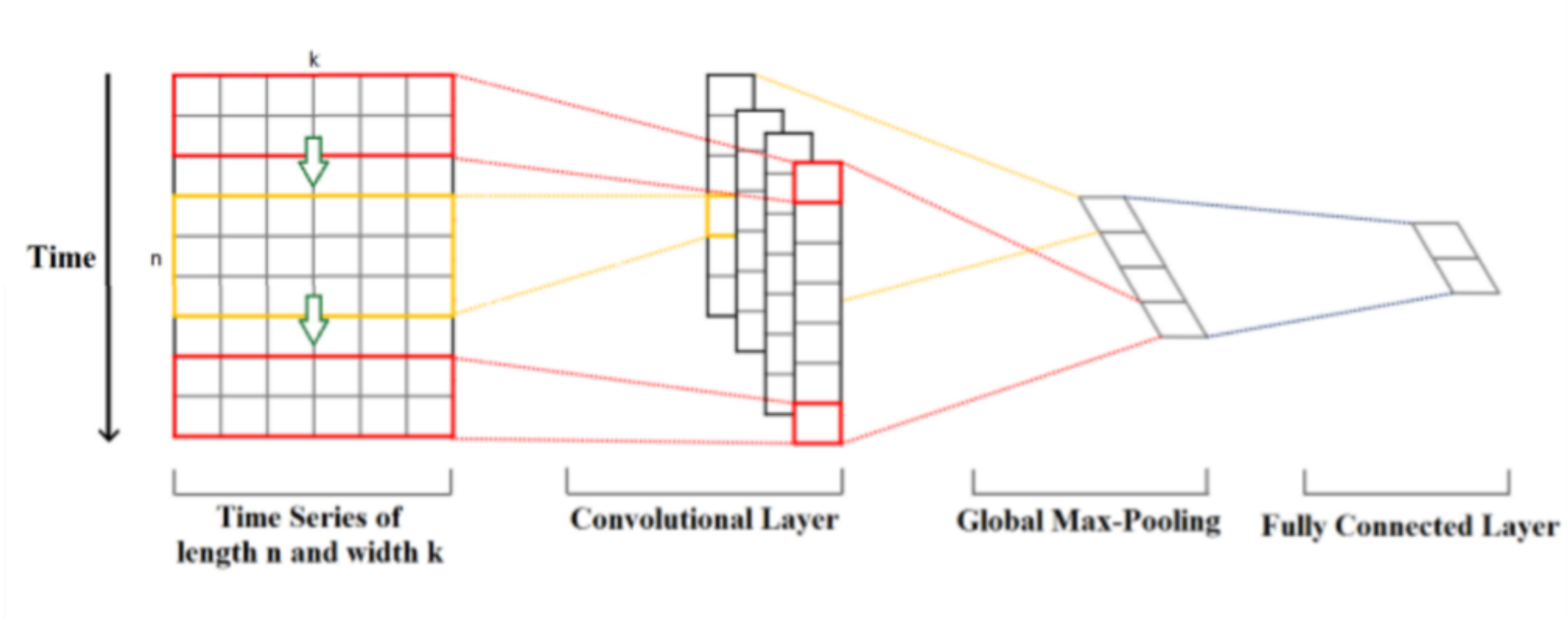



Figure 9.19: Many machine learning algorithms learn features that detect edges or specific colors of edges when applied to natural images. These feature detectors are reminiscent of the Gabor functions known to be present in primary visual cortex. *(Left)*Weights learned by an unsupervised learning algorithm (spike and slab sparse coding) applied to small image patches. *(Right)*Convolution kernels learned by the first layer of a fully supervised convolutional maxout network. Neighboring pairs of filters drive the same maxout unit.

[from Goodfellow et al.]

Convolutional networks for time series → invariance across time



[from Margarita Granat]

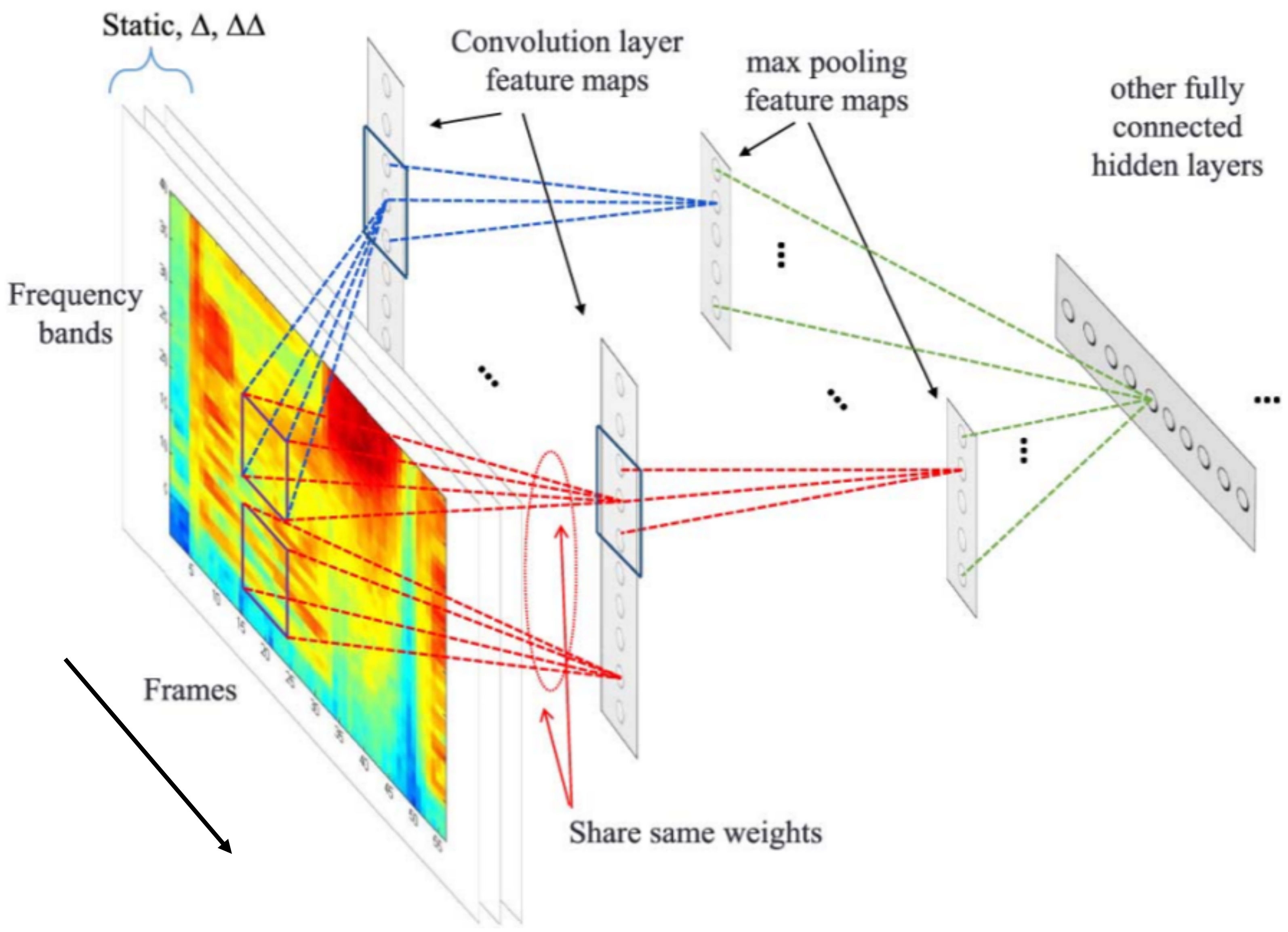


Fig. 3. An illustration of the regular CNN that uses so-called full weight sharing. Here, a 1-D convolution is applied along frequency bands.

[Abdel-Hamid, et al., Convolutional Neural Networks for Speech Recognition, IEEE, 2014]

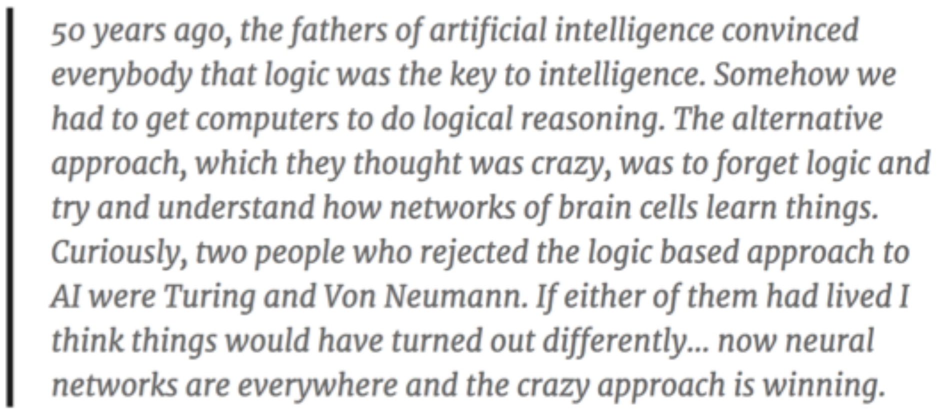
Convolutional Neural Nets

- Convolution across space, time
- Parameter sharing
- Translation invariance
- Scaling
- Multiple channels of “feature maps”
- Architecture with multiple types of layers
- Popular for perception problems

Recurrent Neural Nets for Sequential Data

Sequences

- Words, Letters

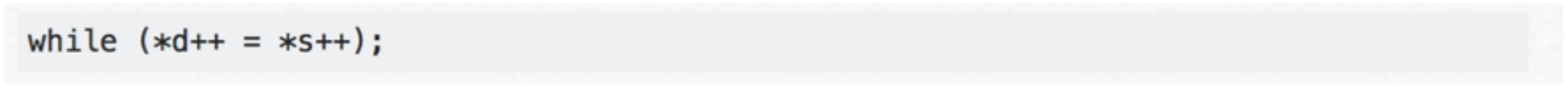


50 years ago, the fathers of artificial intelligence convinced everybody that logic was the key to intelligence. Somehow we had to get computers to do logical reasoning. The alternative approach, which they thought was crazy, was to forget logic and try and understand how networks of brain cells learn things. Curiously, two people who rejected the logic based approach to AI were Turing and Von Neumann. If either of them had lived I think things would have turned out differently... now neural networks are everywhere and the crazy approach is winning.

- Speech

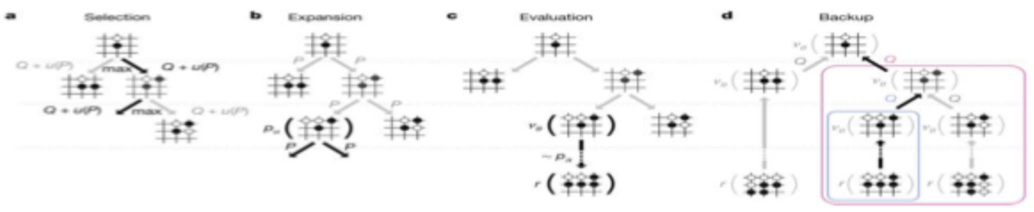
- Images, Videos

- Programs



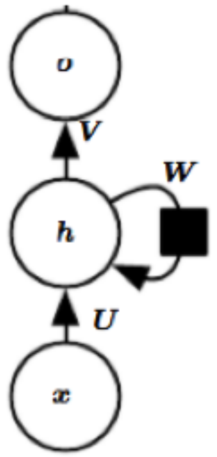
```
while (*d++ = *s++);
```

- Sequential Decision Making (RL)



Recurrent Networks

- Key idea: recurrent network uses (part of) its state at t as input for t+1



$$\mathbf{o_t} = \phi_2(\mathbf{V}\mathbf{h_t} + \mathbf{b_o})$$

$$\mathbf{h_t} = \underbrace{\phi_1}_{\text{Nonlinearity}}(\mathbf{U}\mathbf{x} + \mathbf{W}\underbrace{\mathbf{h_{t-1}}}_{\text{Hidden State at previous time step}} + \mathbf{b_h})$$

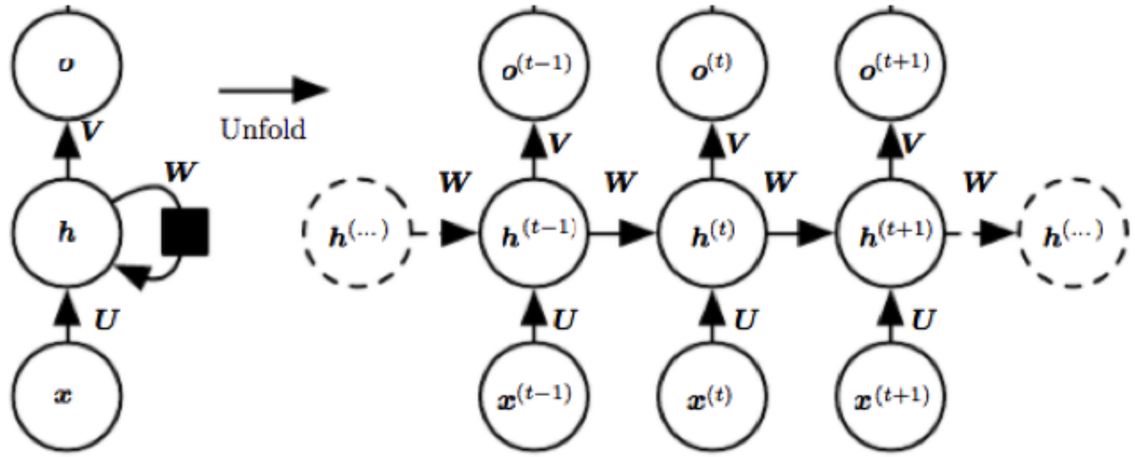
Nonlinearity

Hidden State at previous time step

[Goodfellow et al., 2016]

Recurrent Networks

- Key idea: recurrent network uses (part of) its state at t as input for t+1



$$\mathbf{o_t} = \phi_2(\mathbf{V h_t} + \mathbf{b_o})$$

$$\mathbf{h_t} = \phi_1(\mathbf{Ux} + \mathbf{W h_{t-1}} + \mathbf{b_h})$$

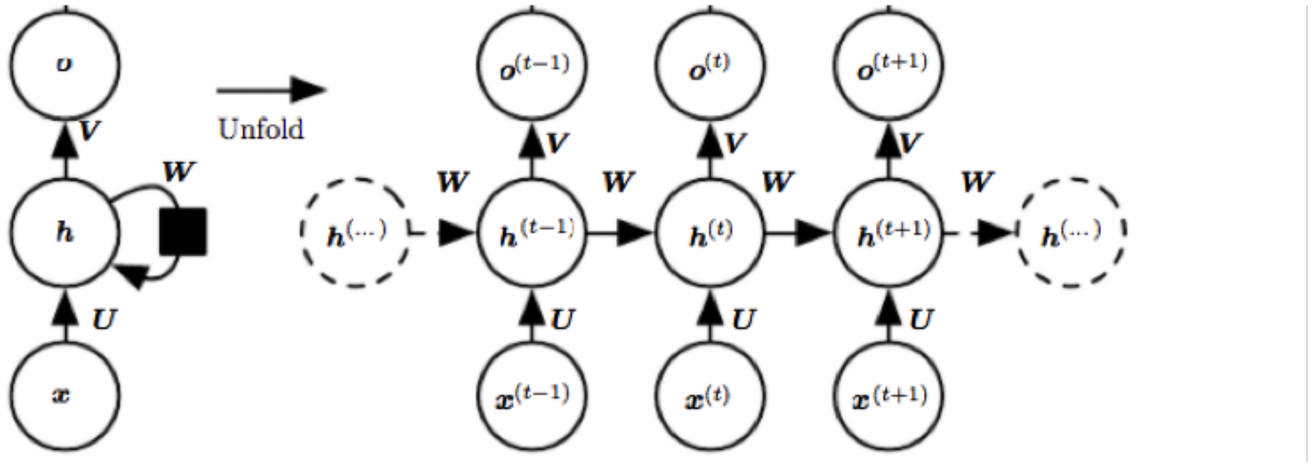
Nonlinearity

Hidden State at previous time step

[Goodfellow et al., 2016]

Recurrent Networks

- Key idea: recurrent network uses (part of) its state at t as input for t+1



$$\mathbf{o_t} = \phi_2(\mathbf{Vh_t} + \mathbf{b_o})$$

$$\mathbf{h_t} = \phi_1(\mathbf{Ux} + \mathbf{Wh_{t-1}} + \mathbf{b_h})$$

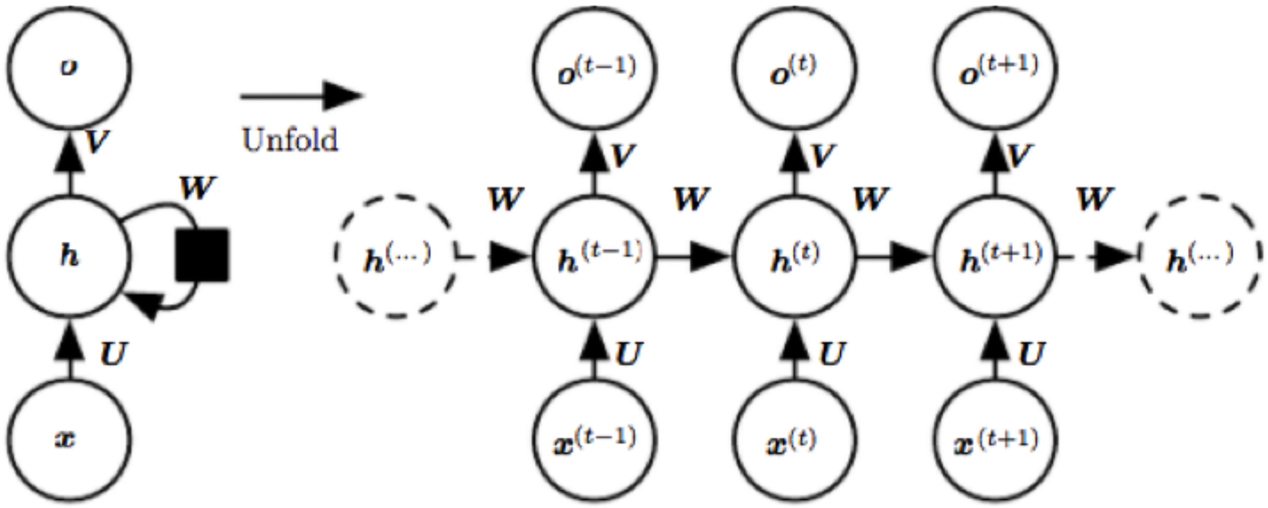
Another example of parameter sharing, like CNNs

[Goodfellow et al., 2016]

Training Recurrent Networks

Key principle for training:

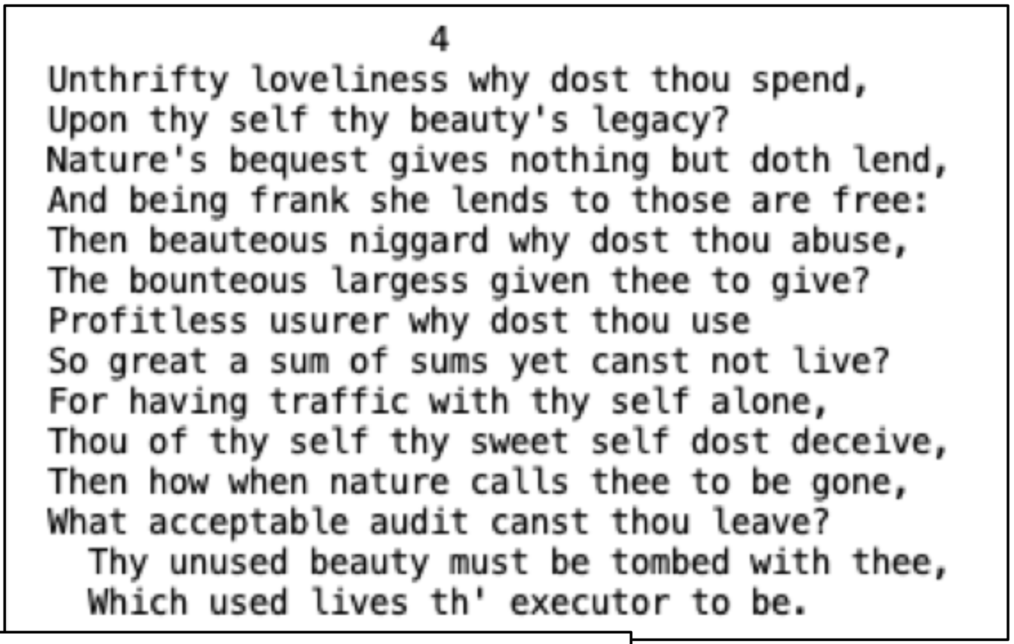
1. Treat as if unfolded in time, resulting in directed acyclic graph
2. Note shared parameters in unfolded net → sum the gradients



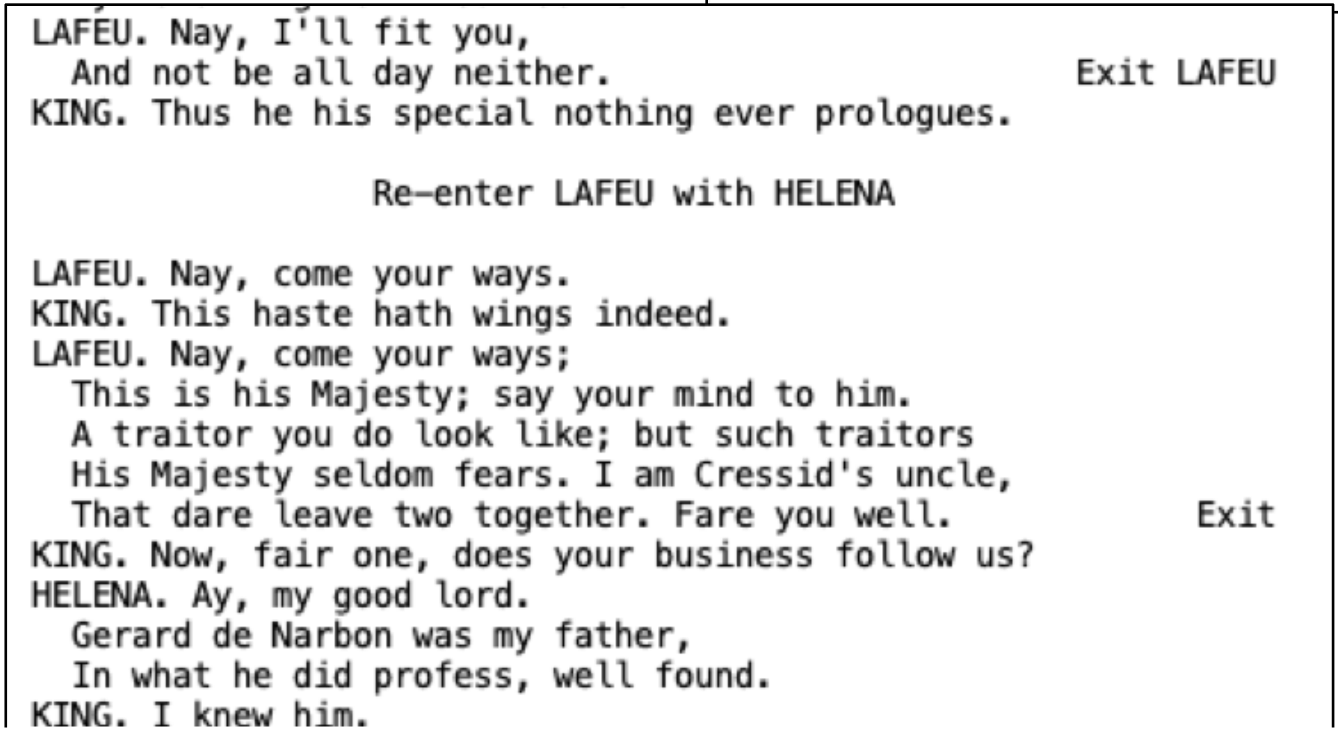
[Goodfellow et al., 2016]

Example: RNN to predict next character in string

- Train on entire works of Shakespeare
- 5,448,482 characters, 84 unique
- Python code online with today's slides



```
                    4
Unthrifty loveliness why dost thou spend,
Upon thy self thy beauty's legacy?
Nature's bequest gives nothing but doth lend,
And being frank she lends to those are free:
Then beauteous niggard why dost thou abuse,
The bounteous largess given thee to give?
Profitless usurer why dost thou use
So great a sum of sums yet canst not live?
For having traffic with thy self alone,
Thou of thy self thy sweet self dost deceive,
Then how when nature calls thee to be gone,
What acceptable audit canst thou leave?
  Thy unused beauty must be tombed with thee,
  Which used lives th' executor to be.
```

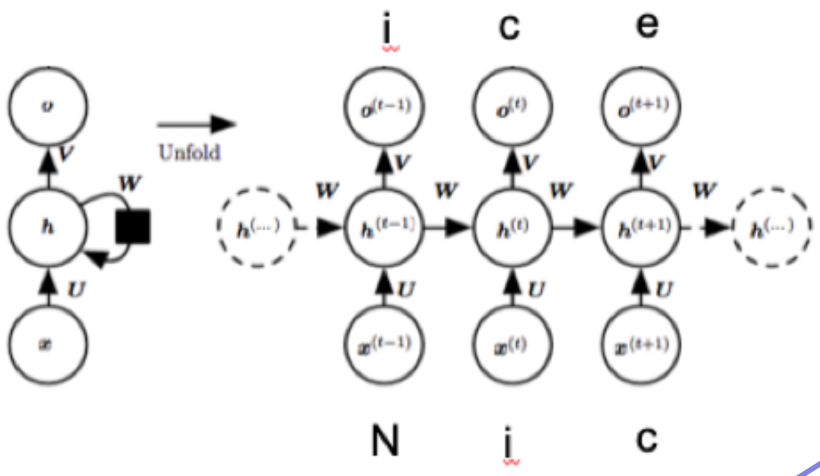


```
LAFEU. Nay, I'll fit you,
  And not be all day neither.                         Exit LAFEU
KING. Thus he his special nothing ever prologues.

                 Re-enter LAFEU with HELENA

LAFEU. Nay, come your ways.
KING. This haste hath wings indeed.
LAFEU. Nay, come your ways;
  This is his Majesty; say your mind to him.
  A traitor you do look like; but such traitors
  His Majesty seldom fears. I am Cressid's uncle,
  That dare leave two together. Fare you well.            Exit
KING. Now, fair one, does your business follow us?
HELENA. Ay, my good lord.
  Gerard de Narbon was my father,
  In what he did profess, well found.
KING. I knew him.
```

Example: RNN to predict next character in string



84 unique characters in this dataset

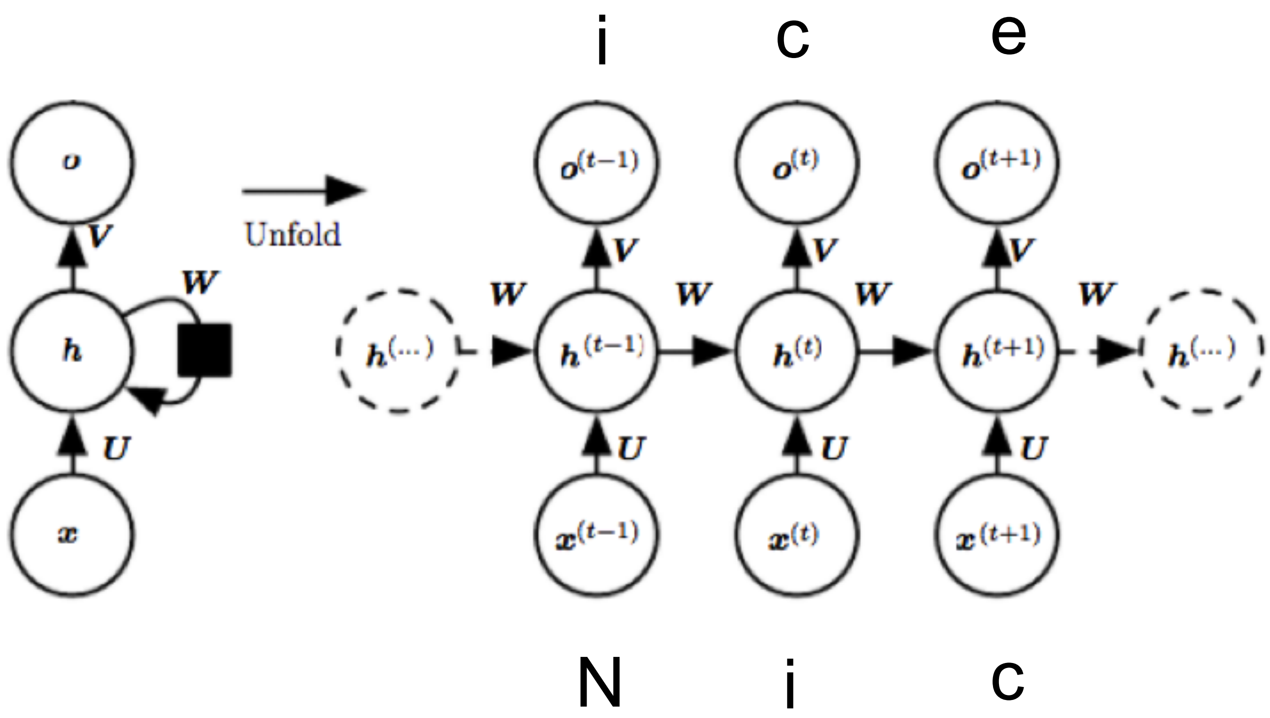
- x_t : input character, encode 1-hot, 84 dimensions
- h_t : hidden layer, 100 dimension
- o_t : predicted next character, softmax, 84 dimensions

$$Pr[\text{next char is } c|x_t, x_{t-1}, \ldots] = \frac{\exp(O^{(t)}(c))}{\sum_{i=1}^{84} \exp(O^{(t)}(i))}$$

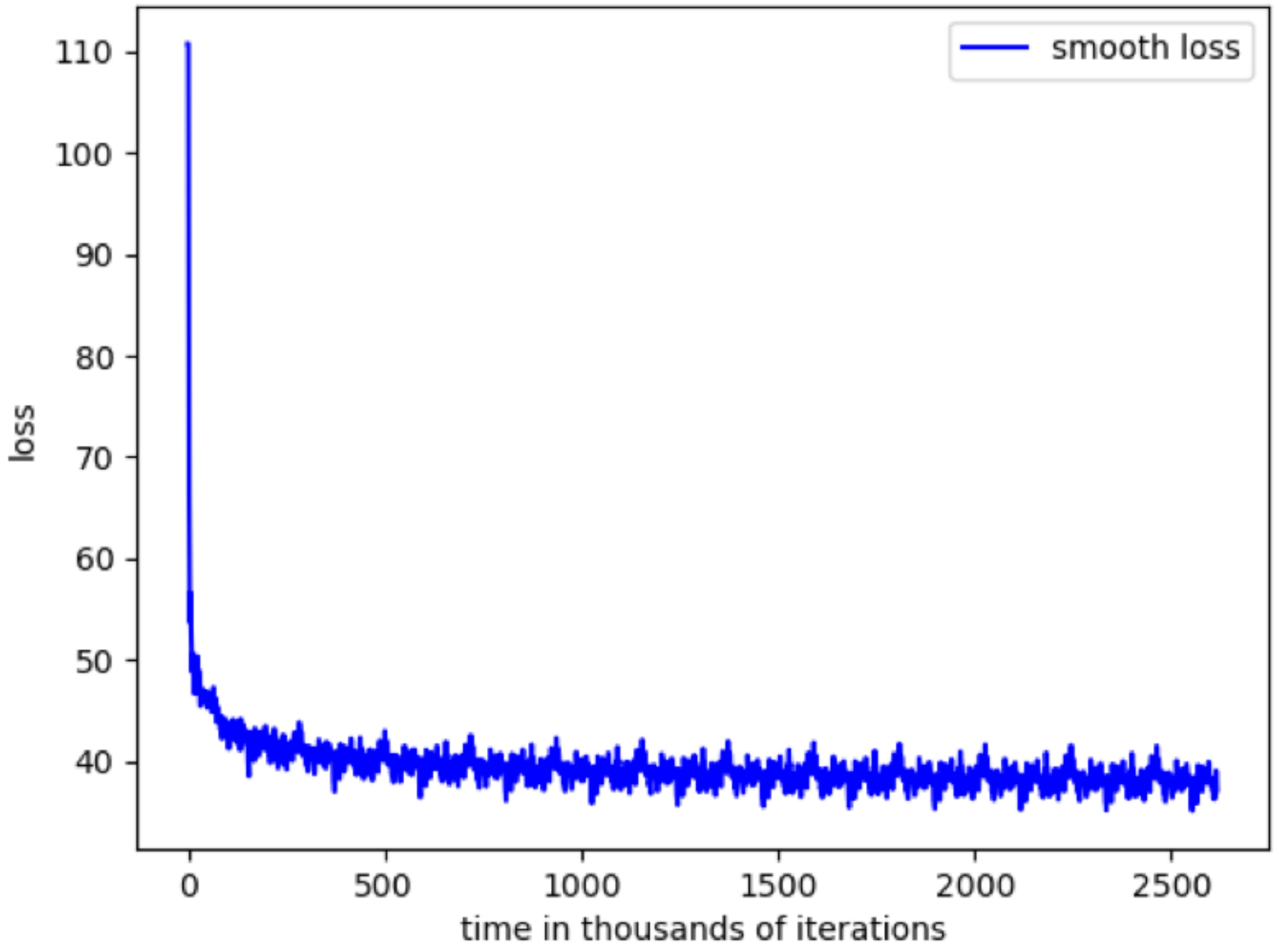
$$\mathbf{h}^{(\mathbf{t})} = \tanh(\mathbf{W_{hh}h^{(t-1)}} + \mathbf{W_{xh}x^{(t)}} + \mathbf{b_h})$$

$$\mathbf{o^t} = \mathbf{W_{ho}h^{(t)}} + \mathbf{b_o}$$

Example: RNN to predict next character in string

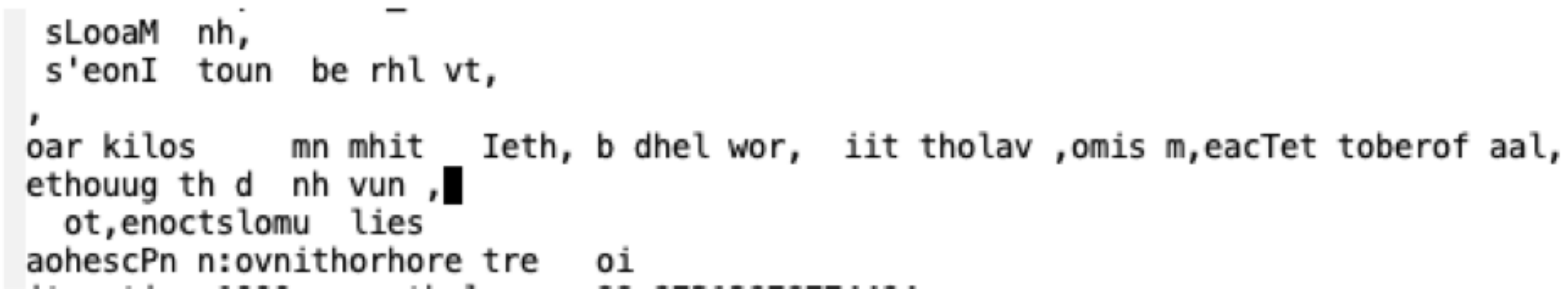


Training loss



Generated strings at different stages of training

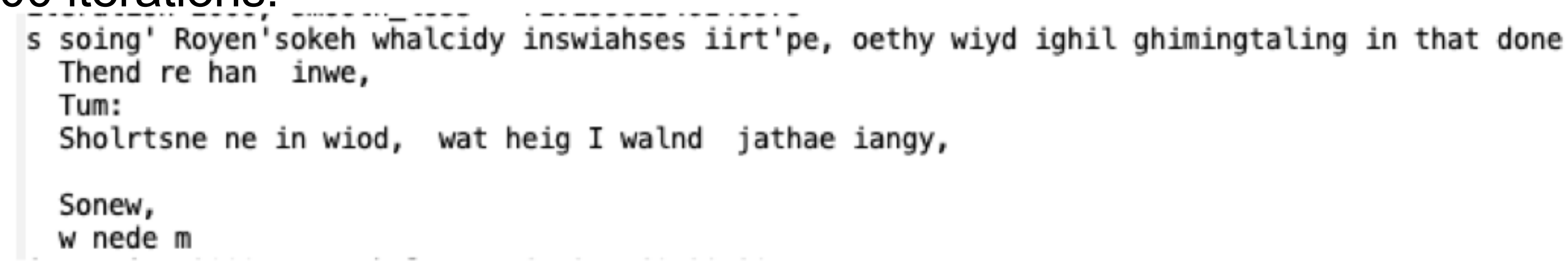
0 iterations:



```
 sLooaM  nh,
 s'eonI  toun  be rhl vt,

,
oar kilos     mn mhit   Ieth, b dhel wor,  iit tholav ,omis m,eacTet toberof aal,
ethouug th d  nh vun ,
  ot,enoctslomu  lies
aohescPn n:ovnithorhore tre   oi
```

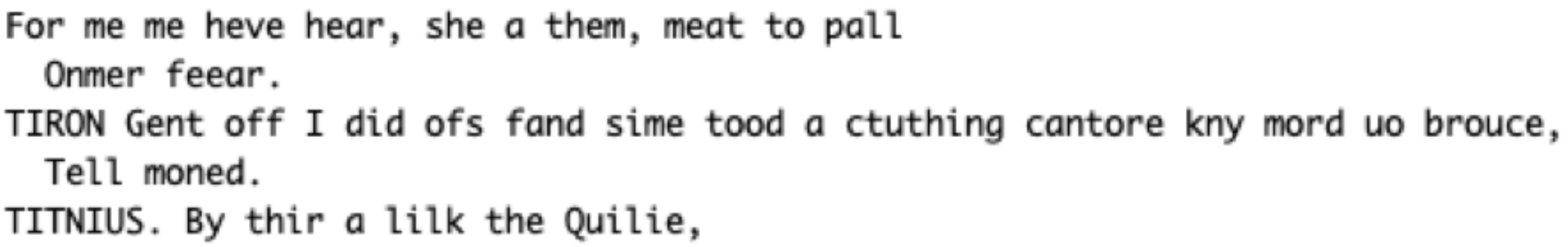
2000 iterations:



```
s soing' Royen'sokeh whalcidy inswiahses iirt'pe, oethy wiyd ighil ghimingtaling in that done
  Thend re han  inwe,
  Tum:
  Sholrtsne ne in wiod,  wat heig I walnd  jathae iangy,

  Sonew,
  w nede m
```

200000 iterations:



```
For me me heve hear, she a them, meat to pall
  Onmer feear.
TIRON Gent off I did ofs fand sime tood a ctuthing cantore kny mord uo brouce,
  Tell moned.
TITNIUS. By thir a lilk the Quilie,
```

Example: Language Models to Predict next word

context					*target*	$P(w_t \mid w_{t-1}, w_{t-2}, \ldots w_{t-5})$
the	cat	sat	on	the	**mat**	0.15
$w_{t\text{-}5}$	$w_{t\text{-}4}$	$w_{t\text{-}3}$	$w_{t\text{-}2}$	$w_{t\text{-}1}$	w_t	
the	cat	sat	on	the	**rug**	0.12
the	cat	sat	on	the	**hat**	0.09
the	cat	sat	on	the	**dog**	0.01
the	cat	sat	on	the	**the**	0
the	cat	sat	on	the	**sat**	0
the	cat	sat	on	the	**robot**	?
the	cat	sat	on	the	**printer**	?

Slide Credit: Piotr Mirowski

Chain Rule

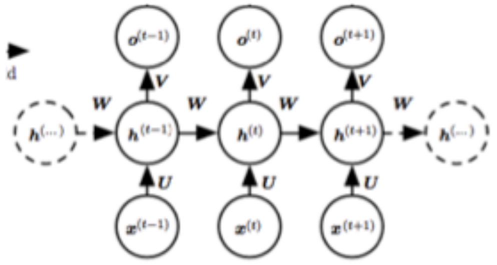
$$\theta^* = \arg\max_{\theta} \log P_\theta(w_1, \ldots, w_T)$$

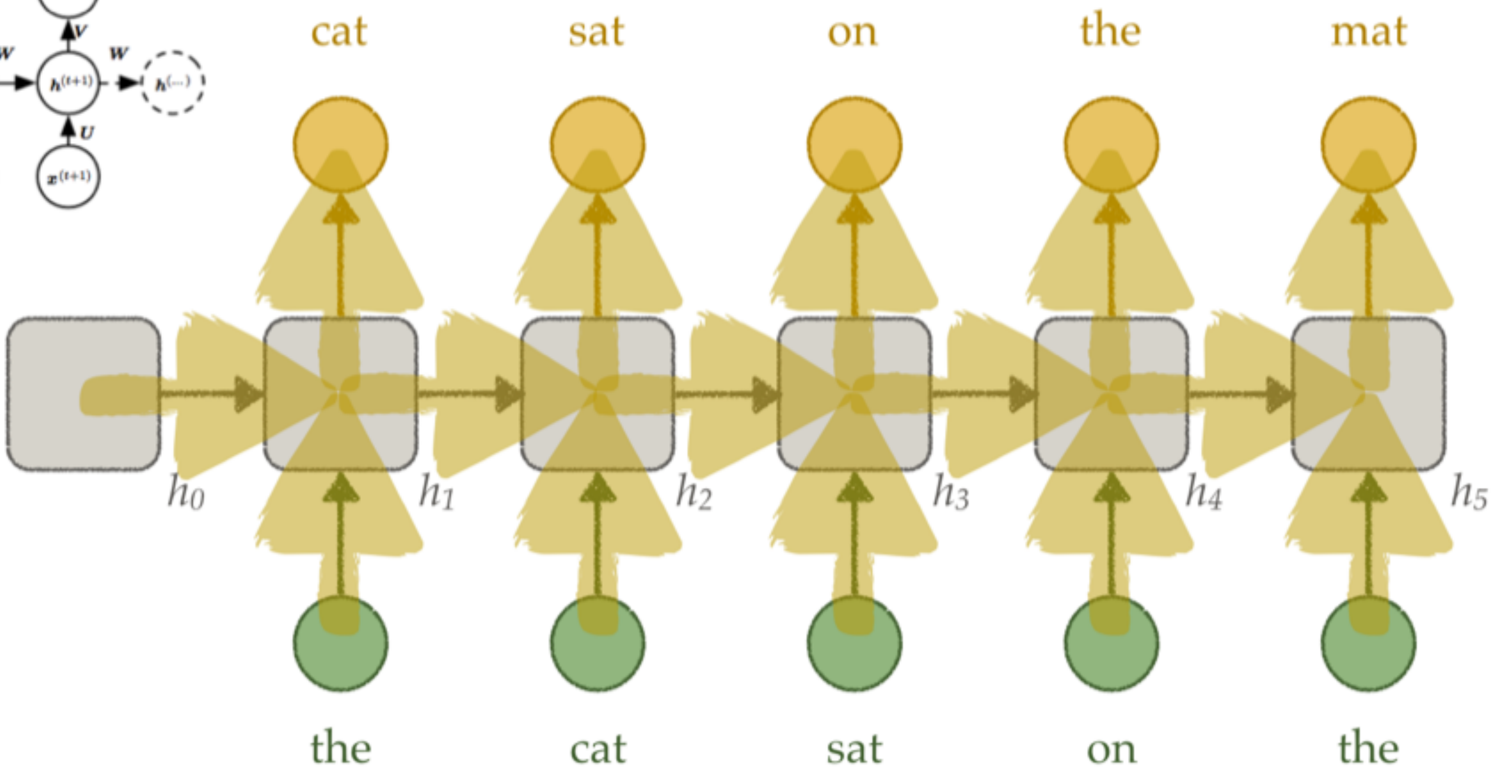
$$P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \| w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \| w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \| w_3, w_2, w_1)$
the	cat	sat	on	**the**	mat	$P(w_5 \| w_4, w_3, w_2, w_1)$
the	cat	sat	on	the	**mat**	$P(w_6 \| w_5, w_4, w_3, w_2, w_1)$

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



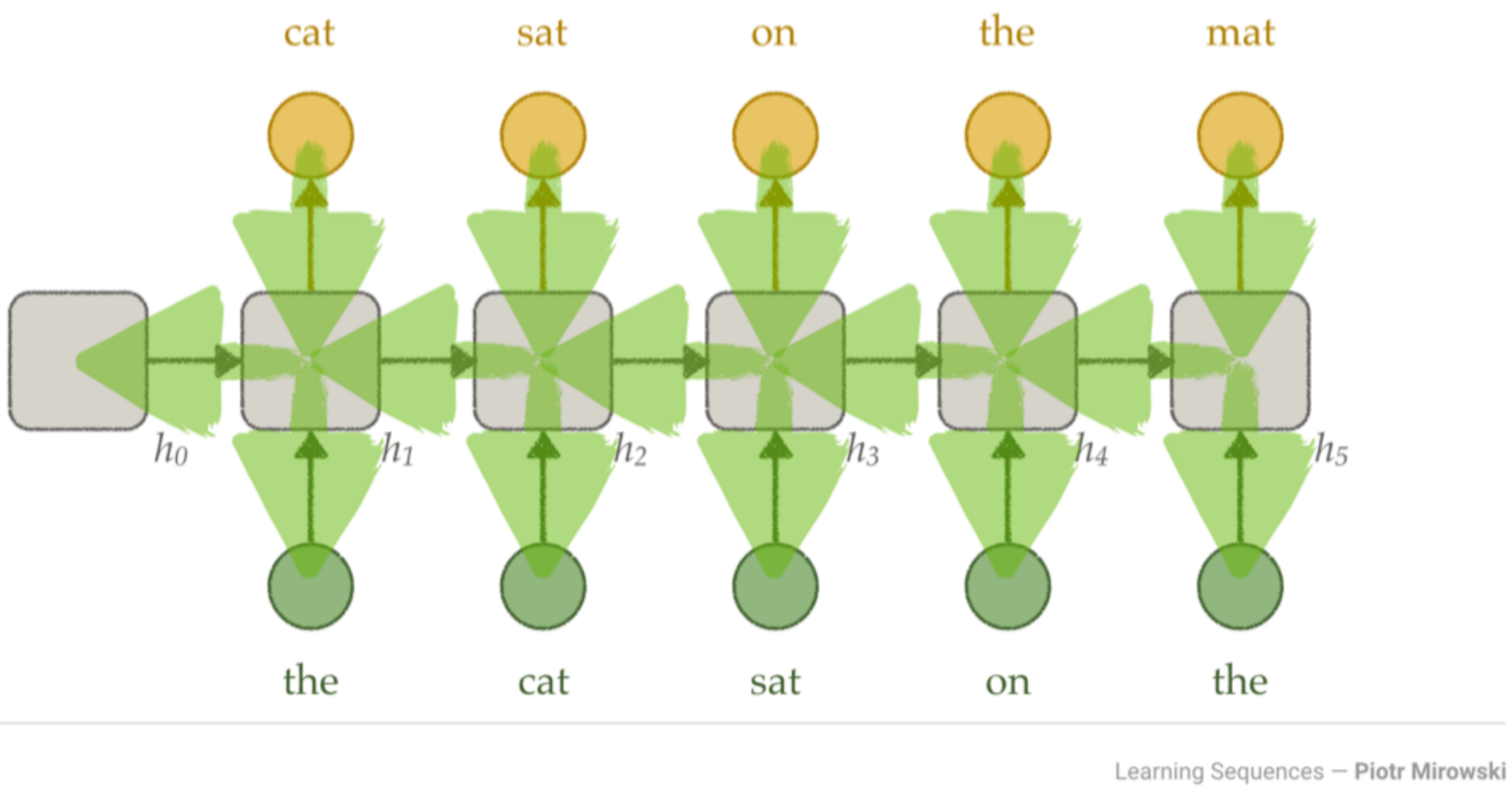


Learning Sequences — **Piotr Mirowski**

- Forward Pass

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models

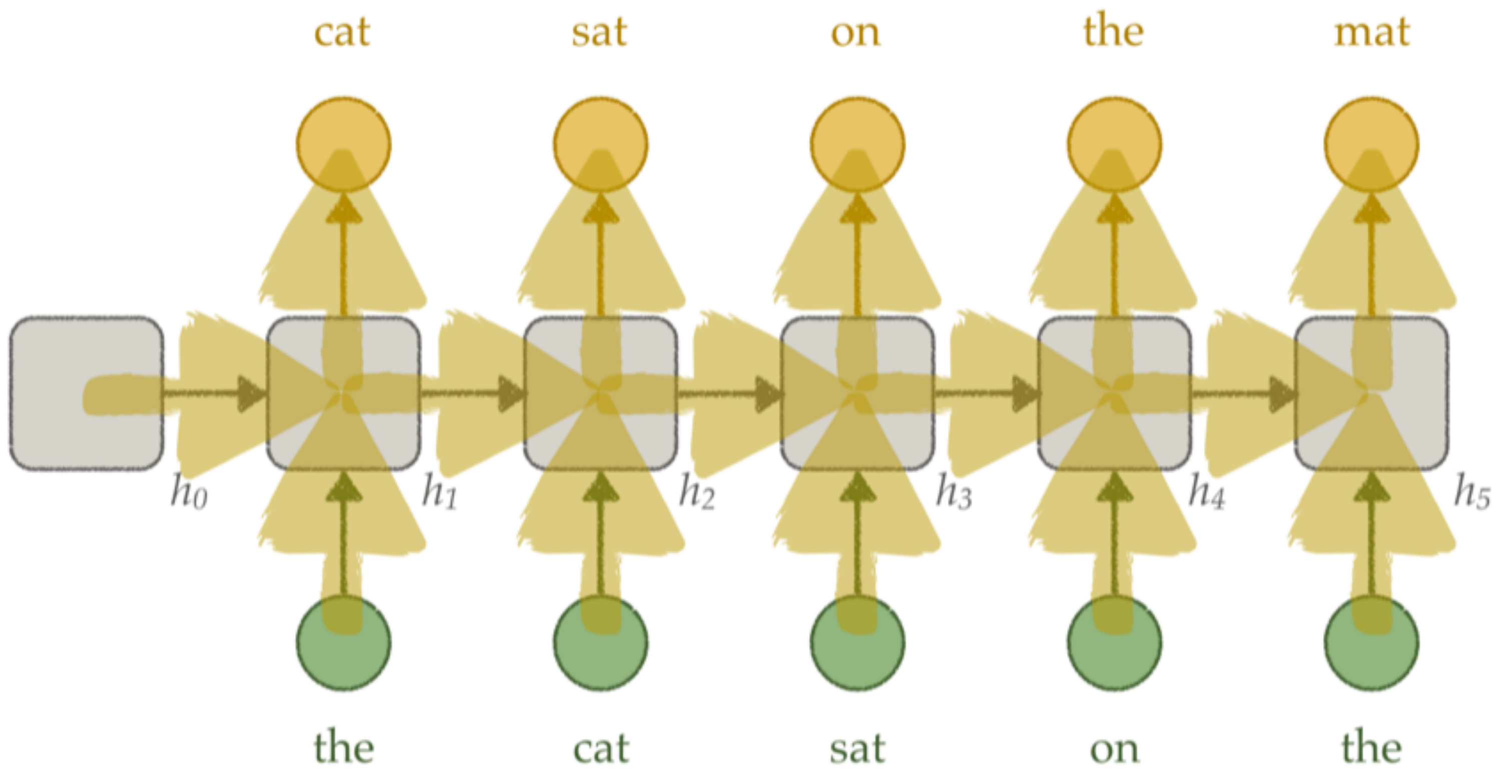


- Backward Pass

* problem: vanishing and/or exploding gradients

Slide Credit: Piotr Mirowski

Recurrent Neural Network Language Models



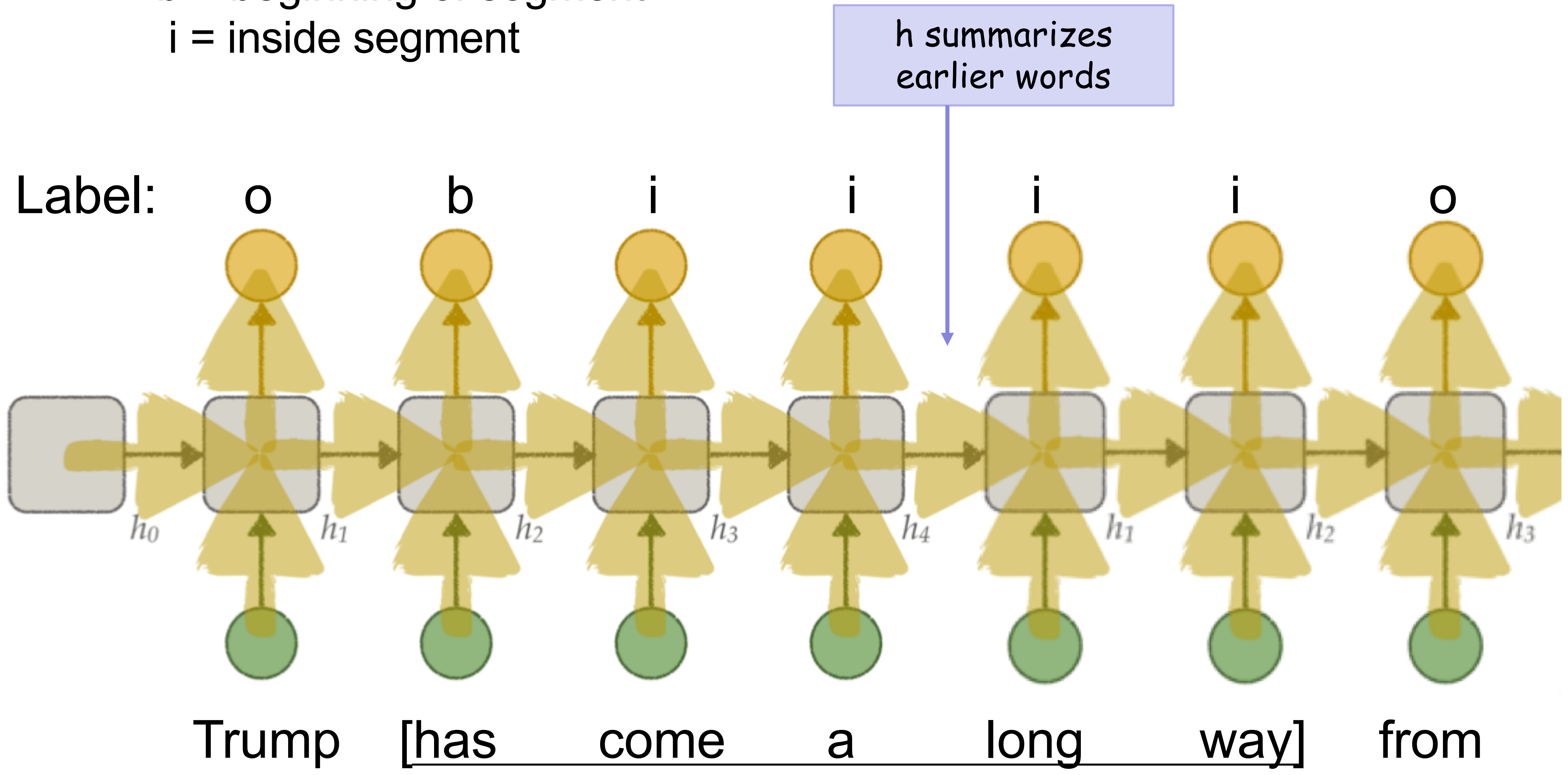
- Learned hidden representations of context useful for:
  - part of speech labeling
  - sentiment analysis
  - information extraction

- Predict label for each word, instead of predicting next word

Slide Credit: Piotr Mirowski

Example: Opinion Mining

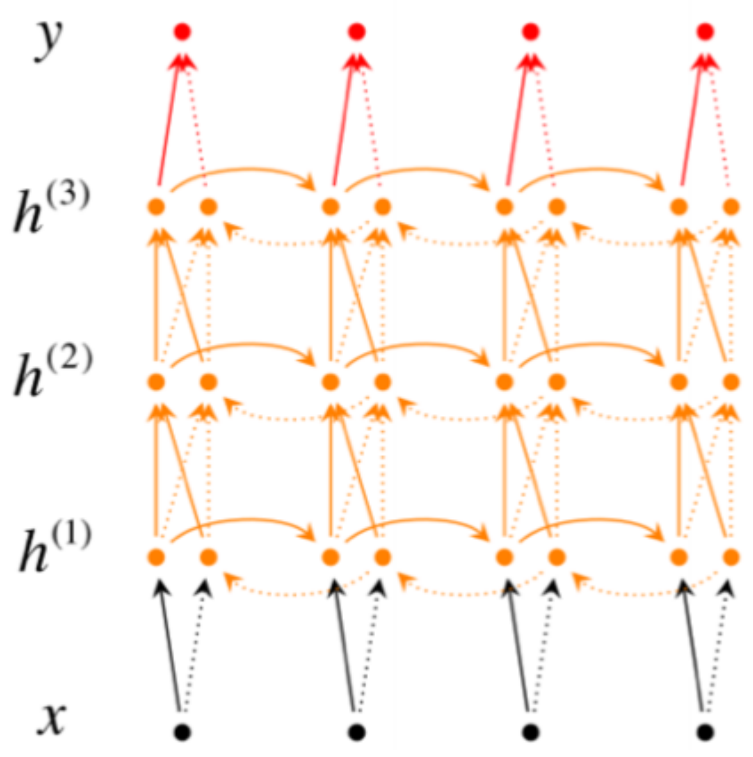
Label opinion segments by labeling each word.
- o = outside
- b = beginning of segment
- i = inside segment



[Irsoy & Cardie, 2014]

Deep Bidirectional Recurrent Network

[Irsoy & Cardie, 2014]



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Two additional ideas:

- Multiple layers to compute y from x
- A left-to-right RNN, plus right-to-left RNN

Example:

- Y label values {begin, inside, outside} for each word, to label contiguous text segments indicating opinions. [Irsoy & Cardie, 2014]

Deep Bidirectional Recurrent Network: Opinion Mining

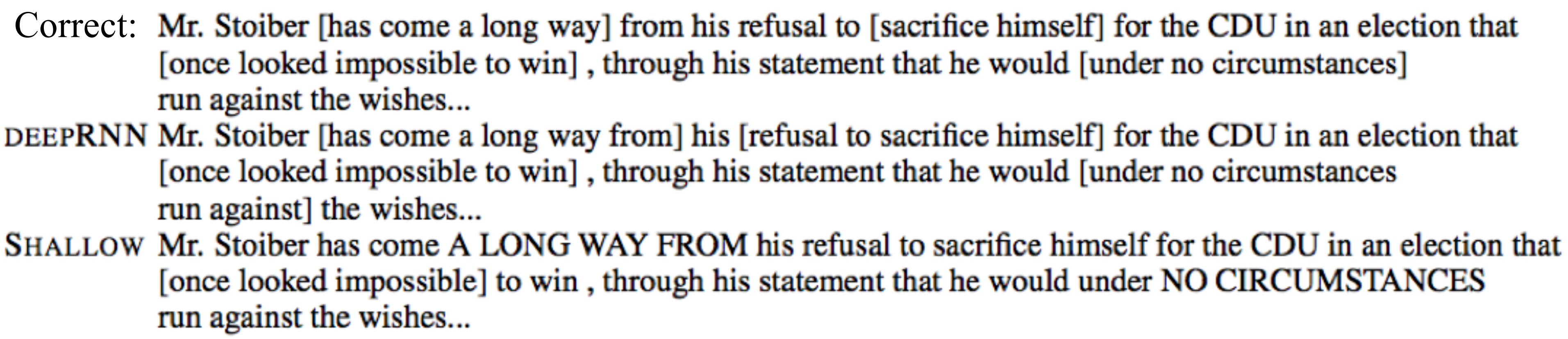
Mr. Stoiber [has come a long way] from his refusal to …

Y: o o b i i i i o o o o

o = outside

b = begin

i = inside



Correct: Mr. Stoiber [has come a long way] from his refusal to [sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances] run against the wishes...

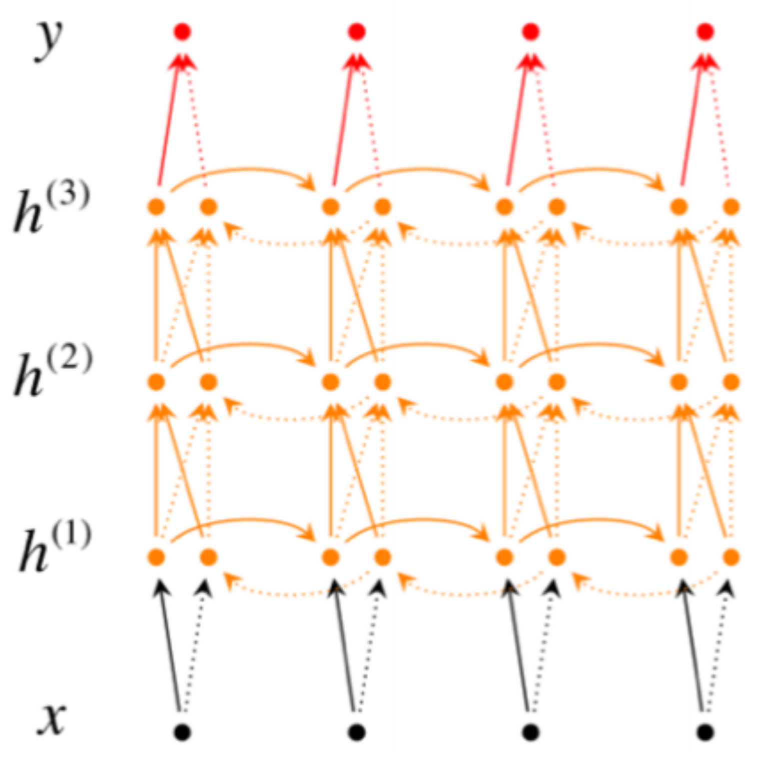
DEEPRNN Mr. Stoiber [has come a long way from] his [refusal to sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances run against] the wishes...

SHALLOW Mr. Stoiber has come A LONG WAY FROM his refusal to sacrifice himself for the CDU in an election that [once looked impossible] to win , through his statement that he would under NO CIRCUMSTANCES run against the wishes...

Figure 3: DEEPRNN Output vs. SHALLOWRNN Output. In each set of examples, the gold-standard annotations are shown in the first line. Tokens assigned a label of Inside with no preceding Begin tag are shown in ALL CAPS.

[Irsoy & Cardie, 2014]

Deep Bidirectional Recurrent Network



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Two additional ideas:

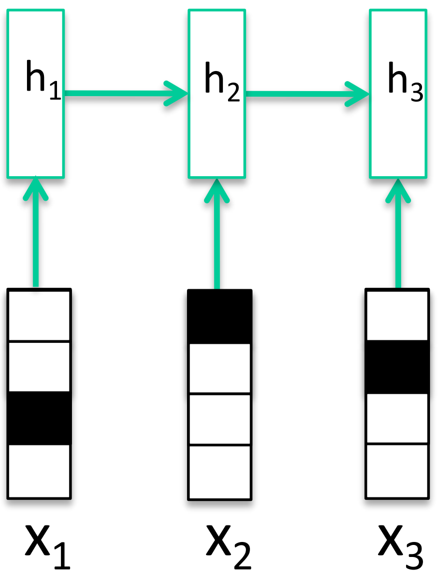
- Multiple layers to compute y from x
- A left-to-right RNN, plus right-to-left RNN

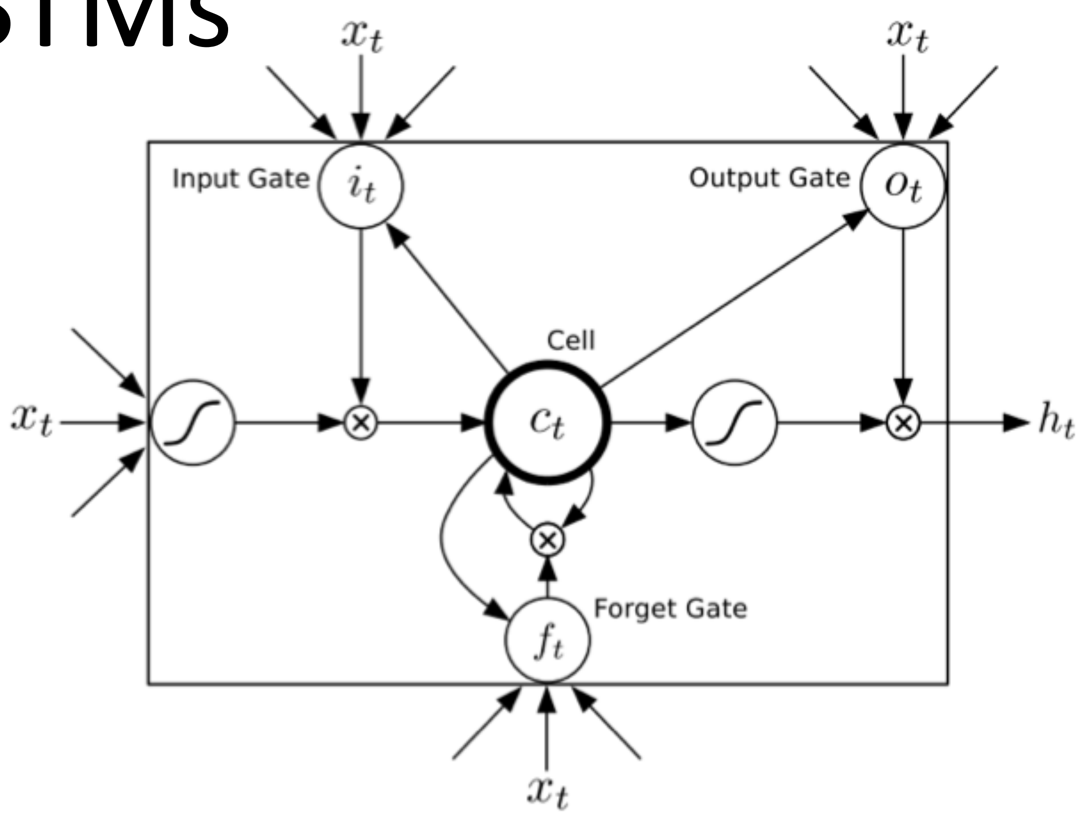
Example:

- Y label values {begin, inside, outside} for each word, to label contiguous text segments indicating opinions. [Irsoy & Cardie, 2014]

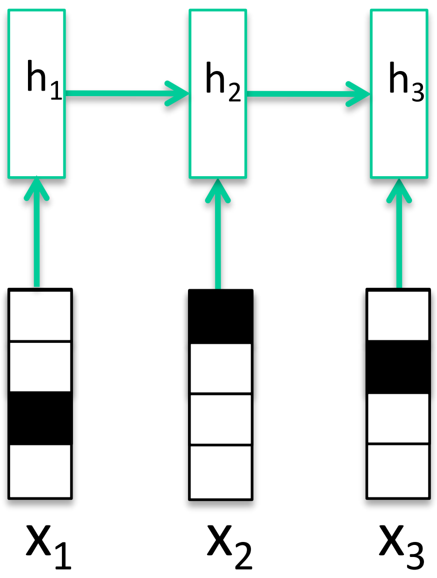
Long Short Term Memory

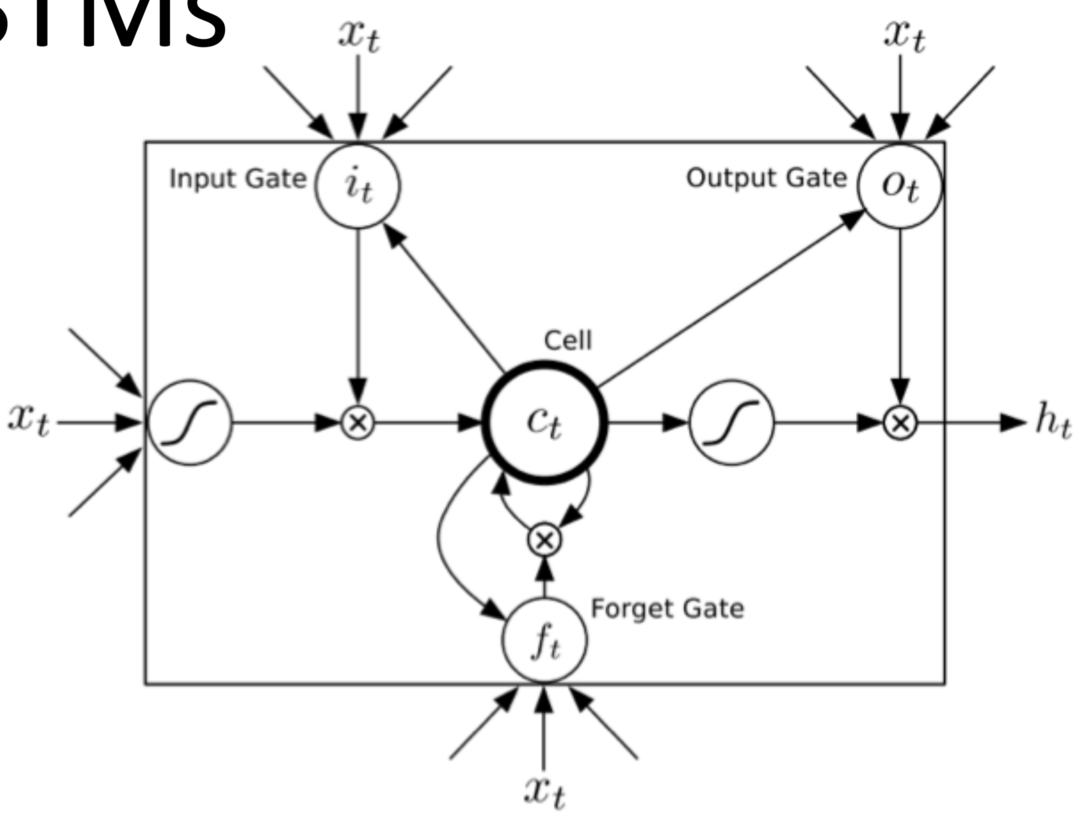
LSTMs





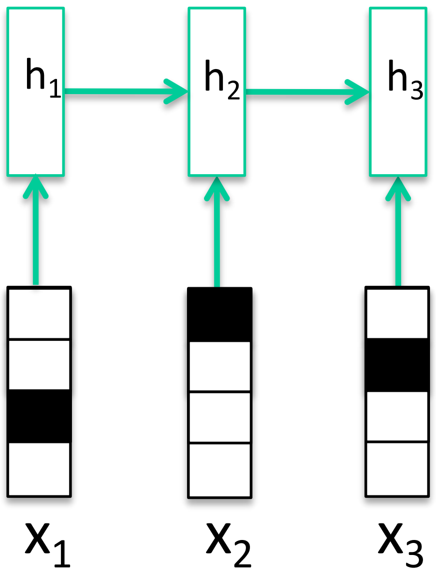
LSTMs

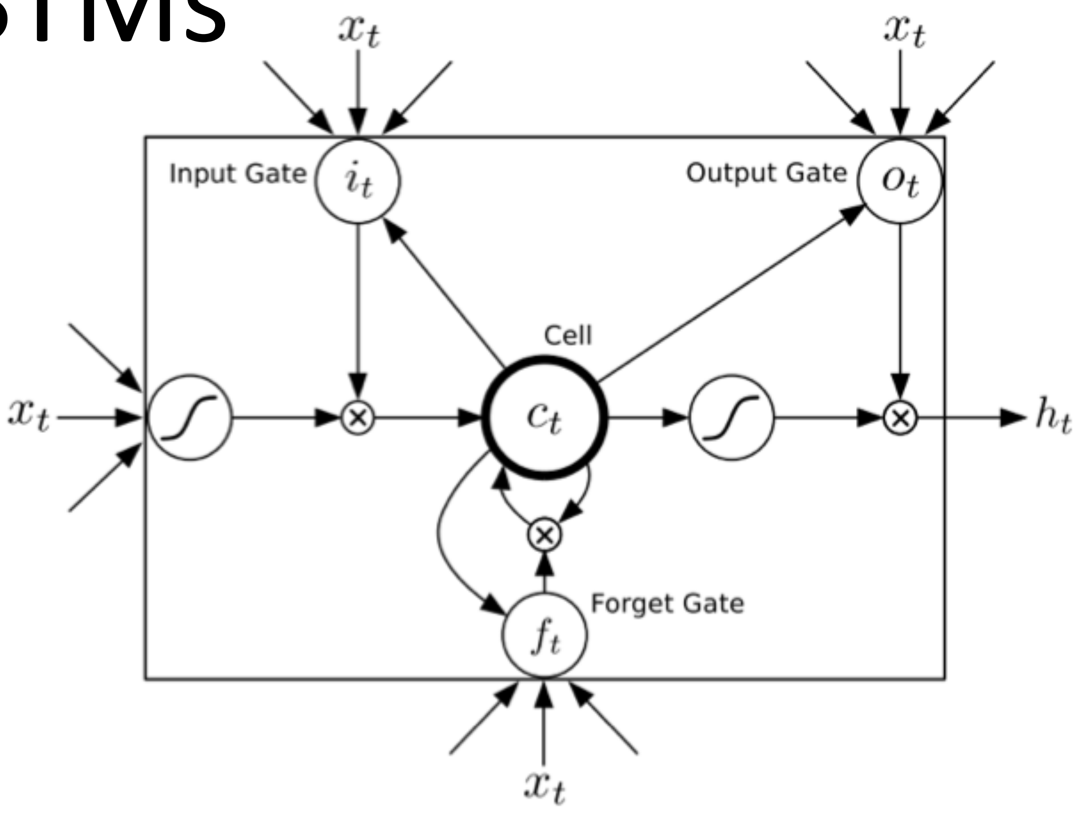




$$\mathbf{i}_t = \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i\right),$$

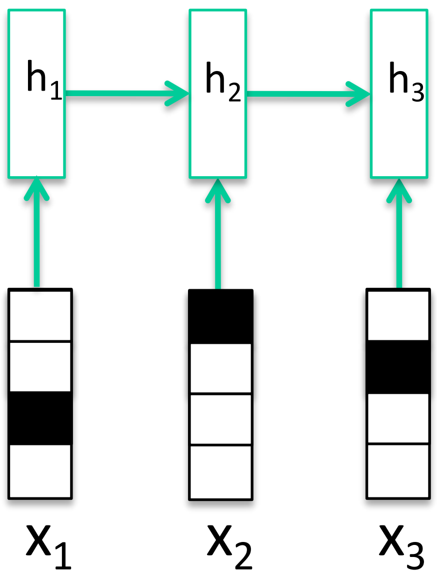
LSTMs

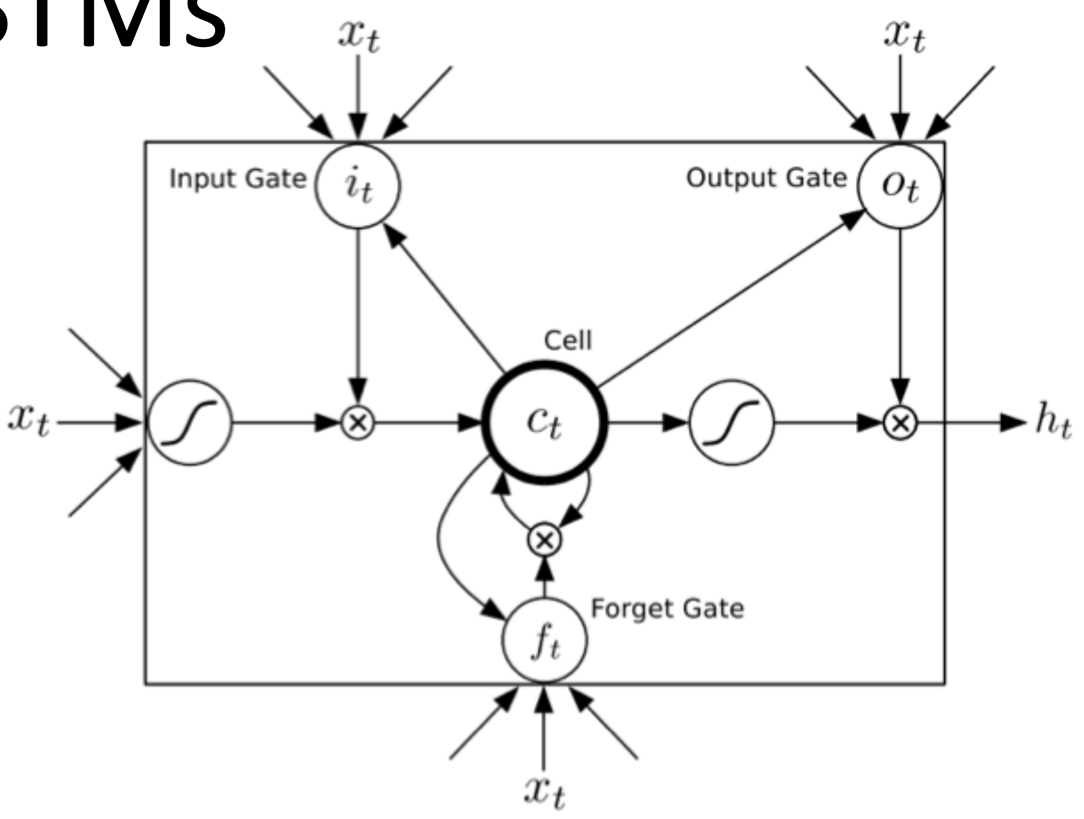




$$\mathbf{i}_t = \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i\right),$$
$$\mathbf{f}_t = \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f\right),$$

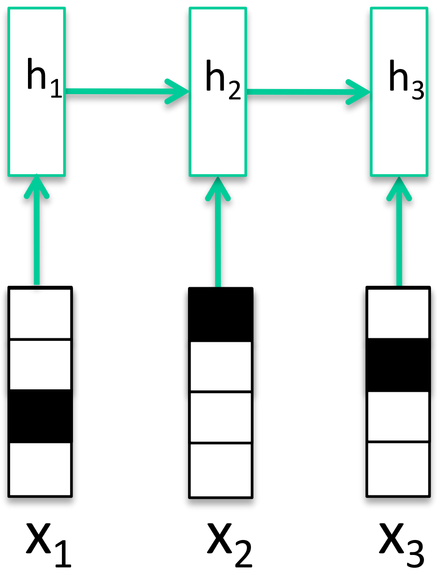
LSTMs

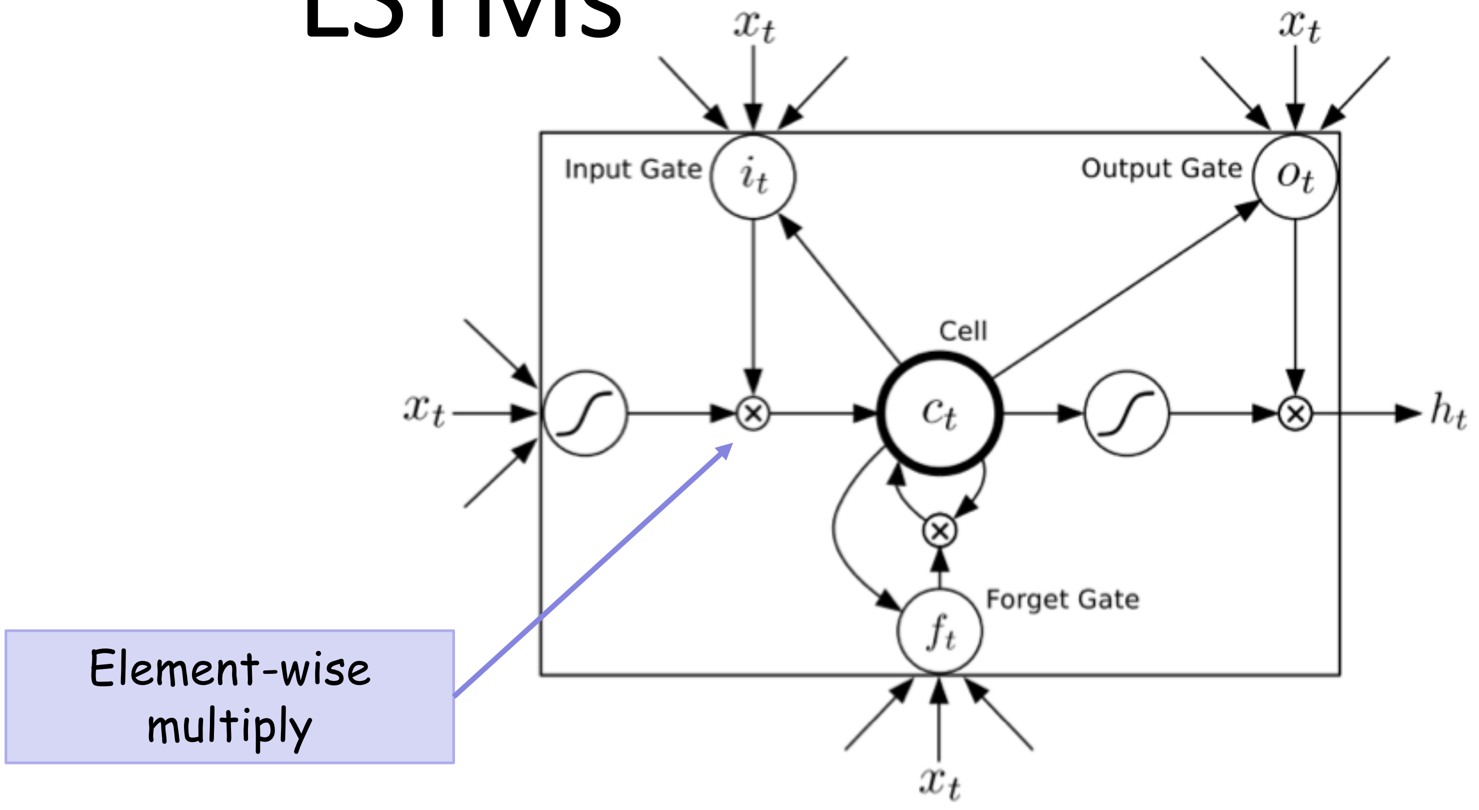




$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i\right),\\
\mathbf{f}_t &= \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f\right),\\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh\left(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right),
\end{aligned}$$

LSTMs

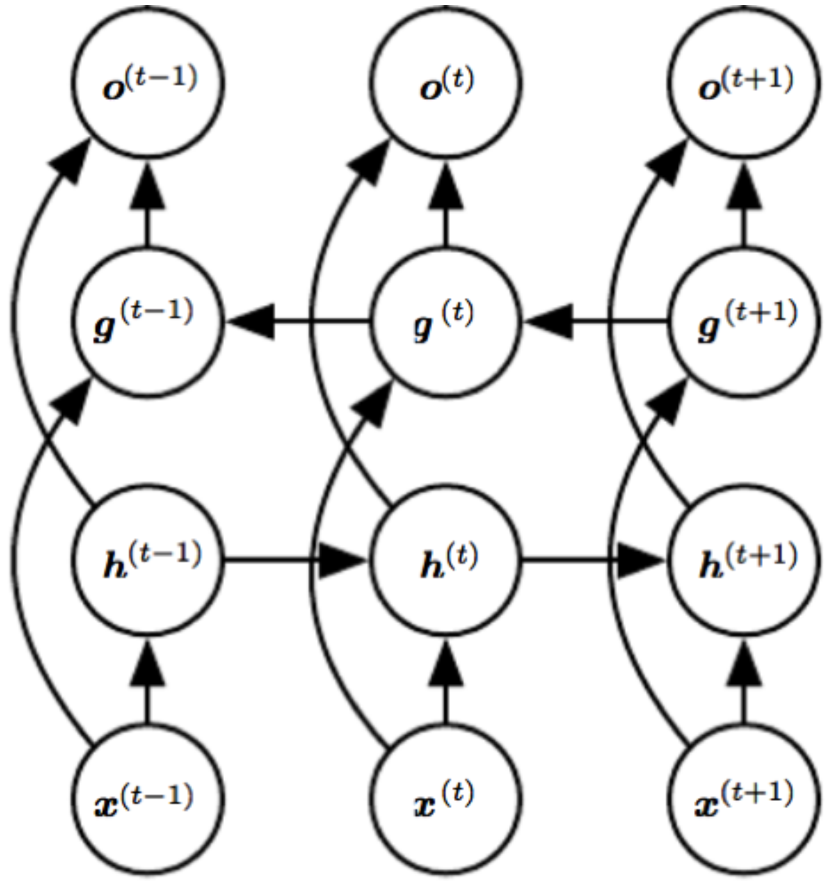




$$
\begin{aligned}
\mathbf{i}_t &= \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i\right), \\
\mathbf{f}_t &= \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f\right), \\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh\left(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\mathbf{o}_t &= \sigma\left(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{co}\mathbf{c}_t + \mathbf{b}_o\right), \\
\mathbf{h}_t &= \mathbf{o}_t \tanh(\mathbf{c}_t).
\end{aligned}
$$

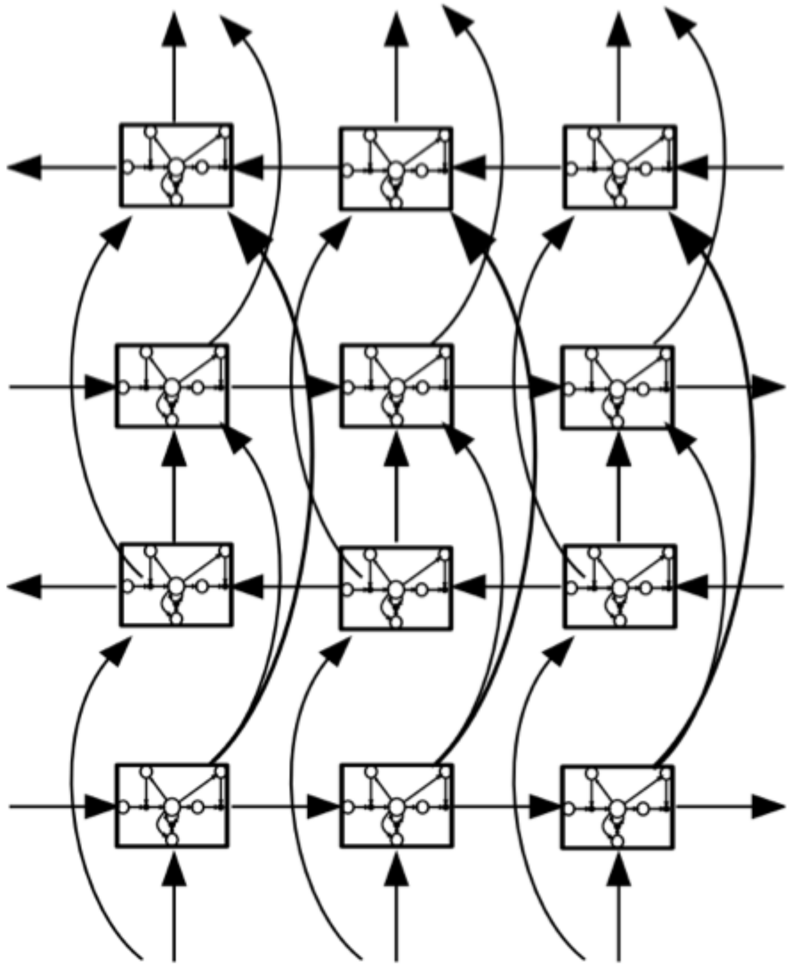
Bi-directional Recurrent Neural Networks

- Key idea: processing of word at position t can depend on following words too, not just preceding words



[Goodfellow et al., 2016]

Deep Bidirectional LSTM Network



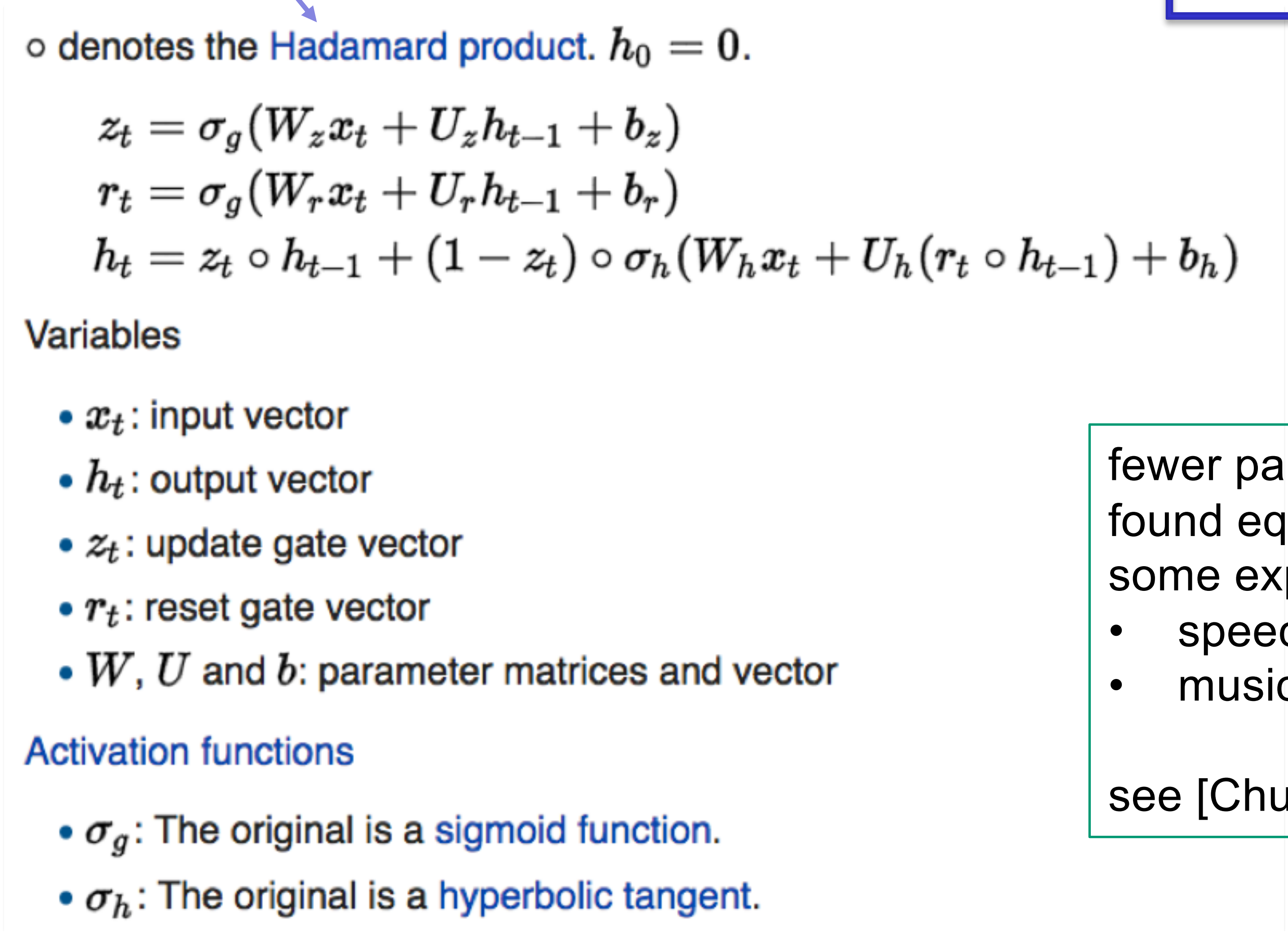
[“Hybrid Speech Recognition with Deep Bidirectional LSTM,” Graves et al., 2013]

Gated Recurrent Units (GRUs)

Optional material – won't be on exam

Element-wise multiply

$$\begin{tikzcd} h_{t-1} \arrow[r] & \boxed{\text{GRU}} \arrow[r] & h_t \\ & x_t \arrow[u] & \end{tikzcd}$$



$\circ$ denotes the Hadamard product. $h_0 = 0$.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

Variables

- x_t: input vector
- h_t: output vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

Activation functions

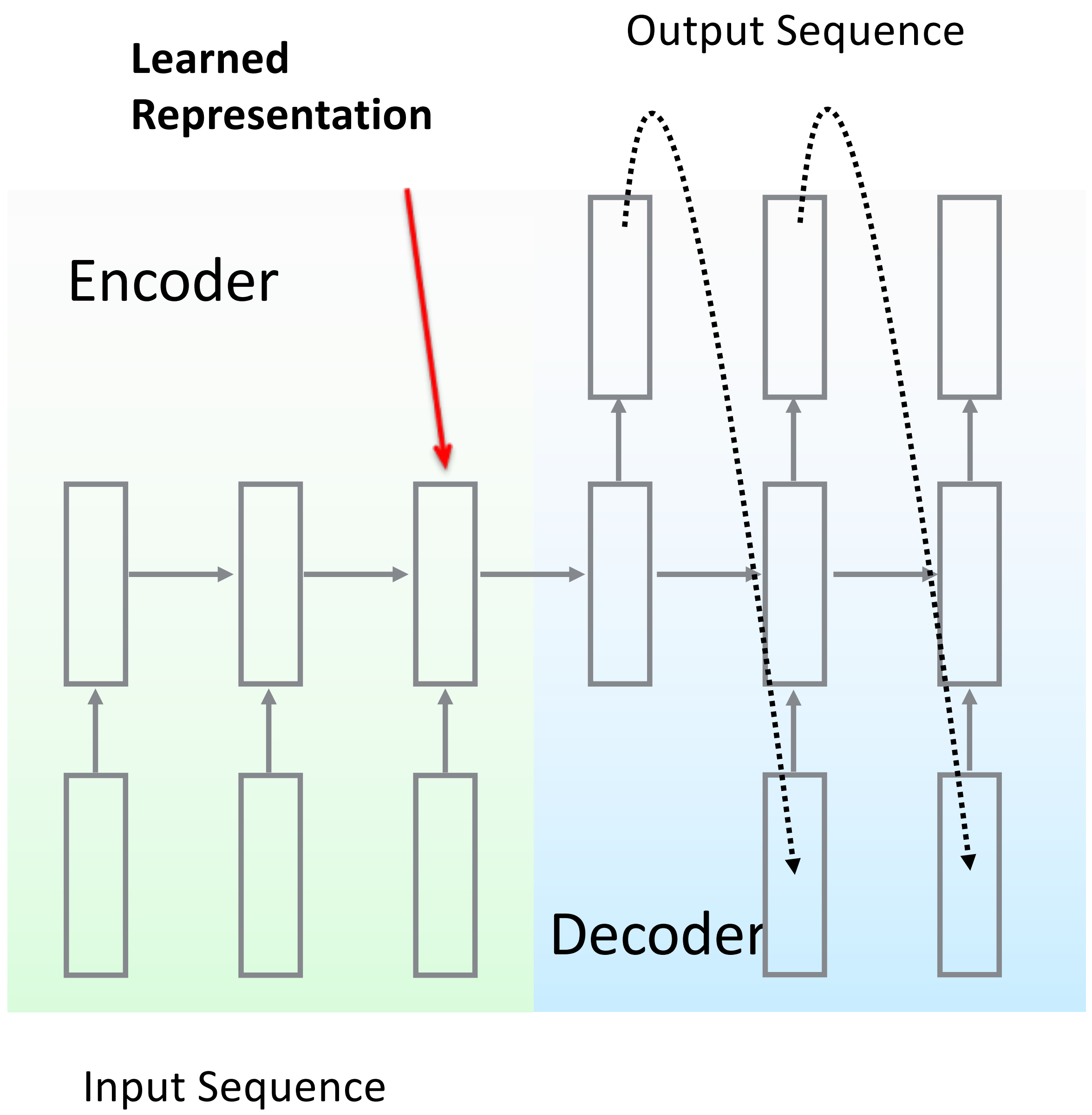
- σ_g: The original is a sigmoid function.
- σ_h: The original is a hyperbolic tangent.

fewer parameters than LSTM
found equally effective in some experiments involving

- speech recognition
- music analysis

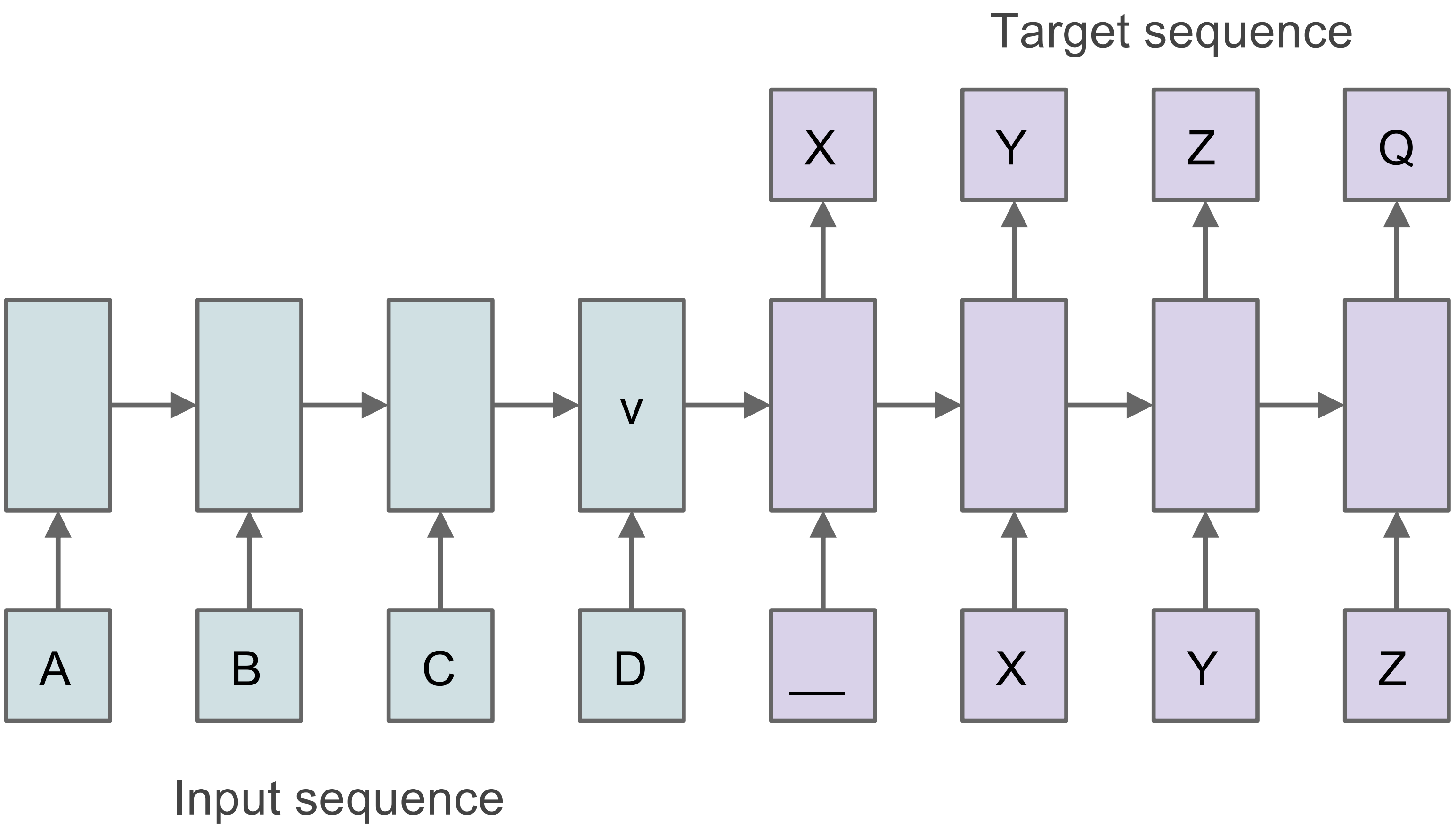
see [Chung et al., 2014]

Sequence to Sequence Learning



- RNN Encoder-Decoders for Machine Translation (Sutskever et al. 2014; Cho et al. 2014; Kalchbrenner et al. 2013, Srivastava et.al., 2015)

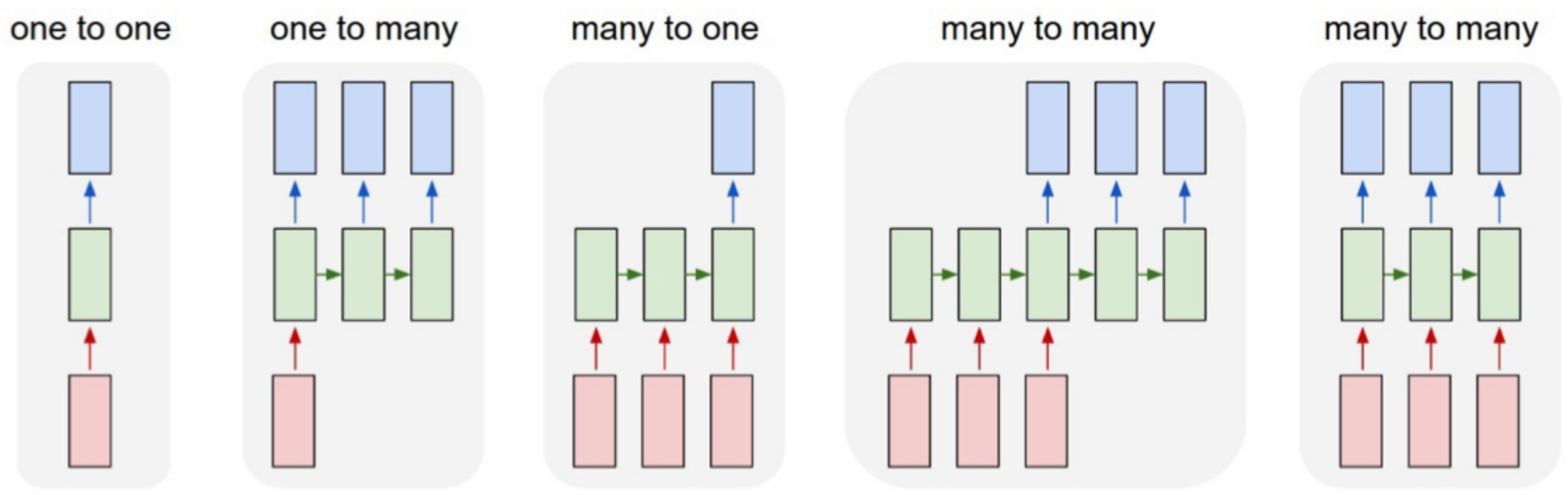
Seq2Seq



$$P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

Sequence to Sequence Models

- Natural language processing is concerned with tasks involving language data



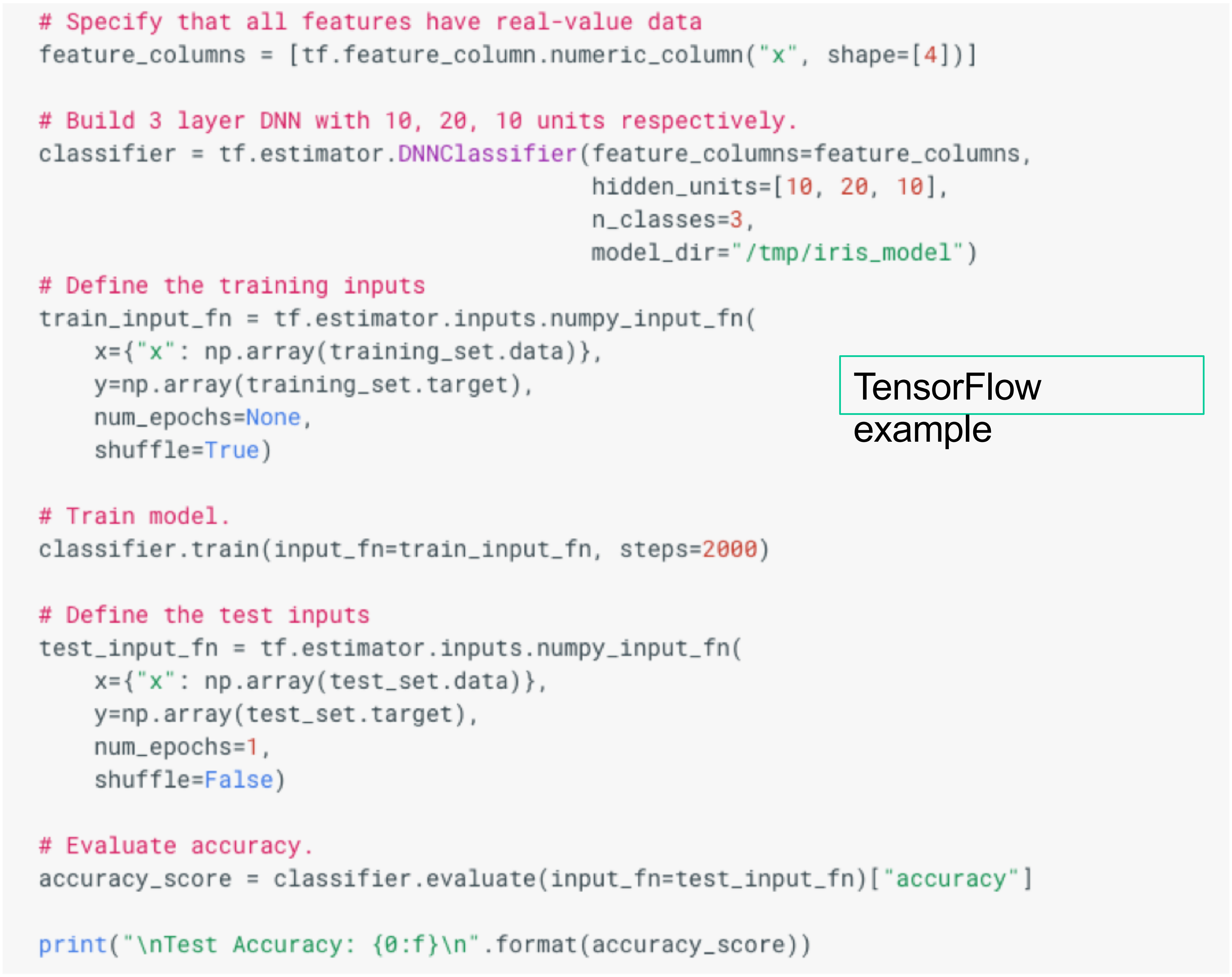
Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks

Programming Frameworks for Deep Nets

- Pytorch (Facebook)
- TensorFlow (Google)
- TFLearn (runs on top of TensorFlow, but simpler to use)
- Theano (University of Montreal)
- CNTK (Microsoft)
- Keras (can run on top of Theano, CNTK, TensorFlow)

Many support use of Graphics Processing Units (GPU’s)

Major factor in dissemination of Deep Network technology



```
# Specify that all features have real-value data
feature_columns = [tf.feature_column.numeric_column("x", shape=[4])]

# Build 3 layer DNN with 10, 20, 10 units respectively.
classifier = tf.estimator.DNNClassifier(feature_columns=feature_columns,
                                        hidden_units=[10, 20, 10],
                                        n_classes=3,
                                        model_dir="/tmp/iris_model")
# Define the training inputs
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": np.array(training_set.data)},
    y=np.array(training_set.target),
    num_epochs=None,
    shuffle=True)

# Train model.
classifier.train(input_fn=train_input_fn, steps=2000)

# Define the test inputs
test_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": np.array(test_set.data)},
    y=np.array(test_set.target),
    num_epochs=1,
    shuffle=False)

# Evaluate accuracy.
accuracy_score = classifier.evaluate(input_fn=test_input_fn)["accuracy"]

print("\nTest Accuracy: {0:f}\n".format(accuracy_score))
```

TensorFlow example

Modern Deep Networks: 2021 vs 1987

- vastly more online data
- GPU's, TPU's
- Heterogenous units
  - Relu, sigmoid, tanh, linear
- including memory units
  - LSTM, GRU, …
- wild new architectures
  - 100 layers deep, bidirectional LSTMs, Convolutional nets widespread ...
- new ideas for gradient descent
  - dropout, batch normalization, weight initialization, ...
- unification with probabilistic models
  - train to output probabilities
- frameworks like TensorFlow

What you should know:

- Representation learning
  - Hidden layers re-represent inputs in form to predict outputs
  - Autoencoders
  - Sometimes reused widely (e.g., word2vec word embeddings)
- Convolutional neural networks
  - Convolution provides translation invariance
  - Network stages with reducing spatial resolution, Mult. channels,…
- Recurrent neural networks
  - Learn to represent history in time series
  - Backpropagation as unfolding in time
  - LSTM memory units
- Neural architectures
  - Shared parameters across multiple computations
  - Layers with different structures/functions
  - Probabilistic classification → output Softmax layer