Machine Learning Department
School of Computer Science
Carnegie Mellon University

Recurrent Neural Networks (RNNs)

Matt Gormley, Henry Chai, Hoda Heidari
Lecture 18
Mar. 25, 2024

Reminders

- Homework 6: Learning Theory & Generative Models
  - Out: Mon, Mar 18
  - Due: Sun, Mar 24 at 11:59pm
- Exam 2: Thu, Mar 28, 7:00 pm - 9:00 pm

Q&A

Q:	Should we be extremely polite and not interrupt you if your slides are not visible?
A:	Please interrupt me.

CNN ARCHITECTURES

Convolutional Neural Network (CNN)

Typical Architectures

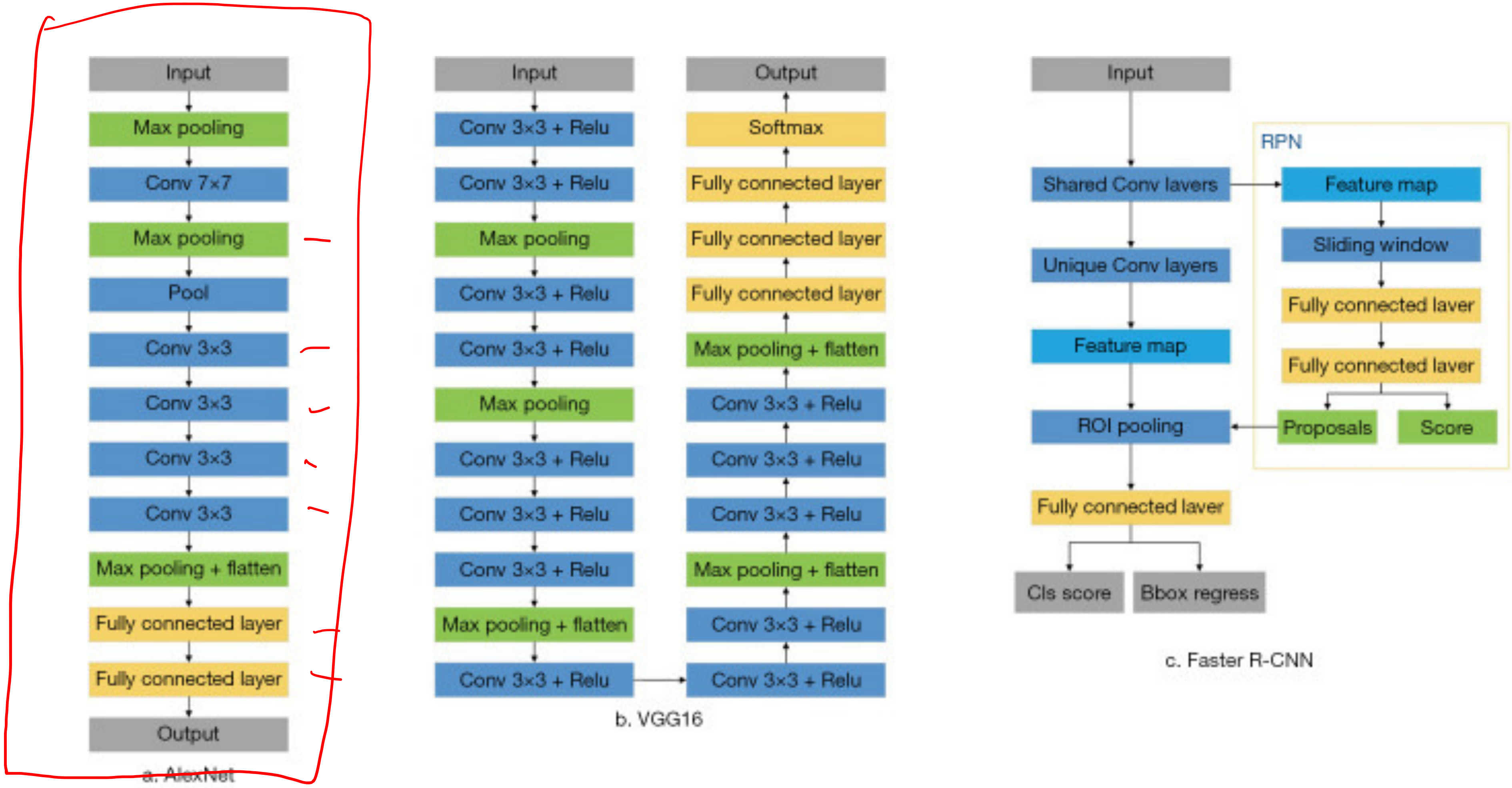


Figure from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7327346/

Convolutional Neural Network (CNN)

Typical Architectures

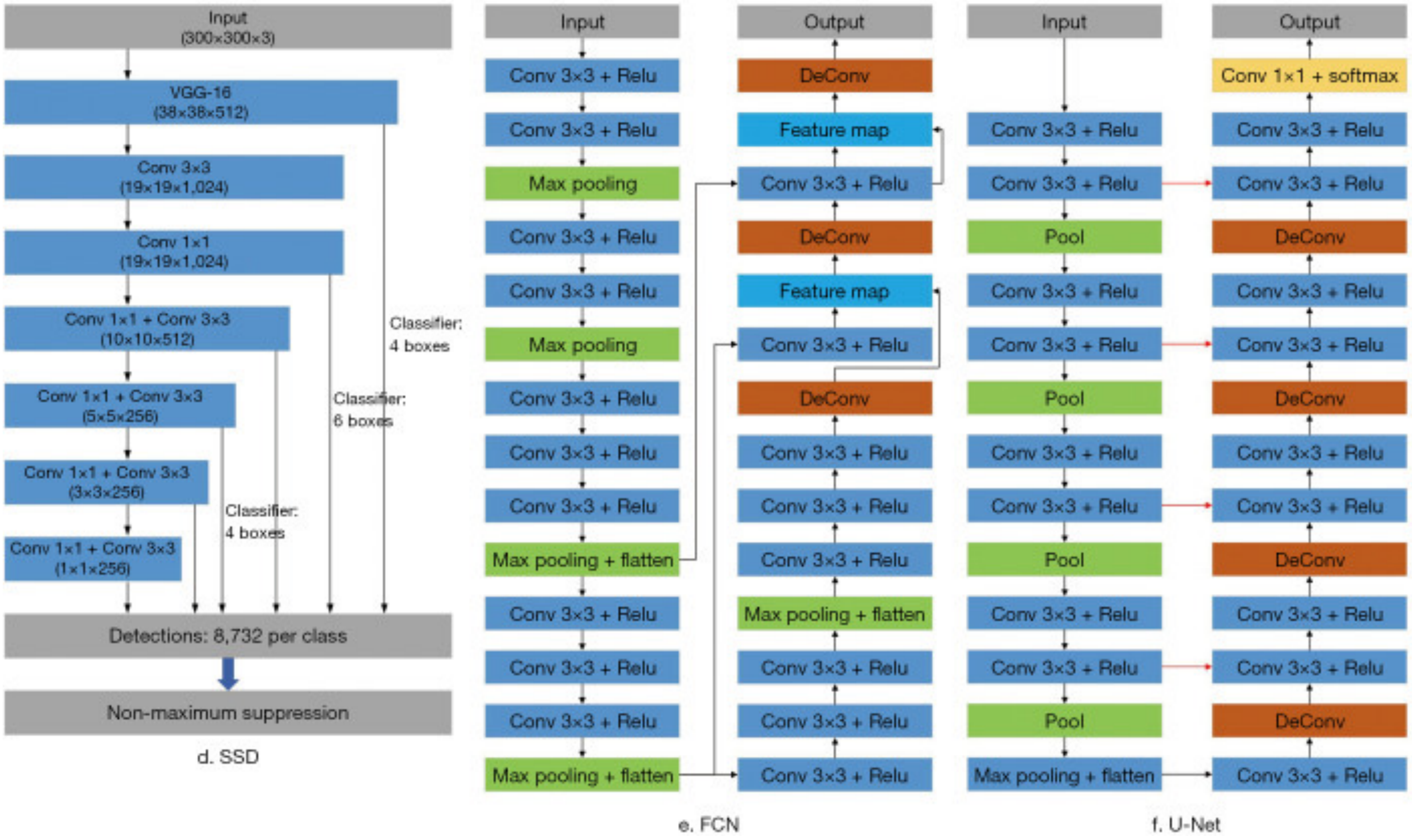
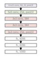


Figure from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7327346/

Convolutional Neural Network (CNN)

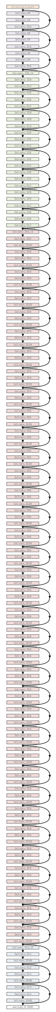
Typical Architectures

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Convolutional Layer

For a convolutional layer, how do we pick the kernel size (aka. the size of the convolution)?

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

2x2 Convolution

θ_{11}	θ_{12}
θ_{21}	θ_{22}

3x3 Convolution

θ_{11}	θ_{12}	θ_{13}
θ_{21}	θ_{22}	θ_{23}
θ_{31}	θ_{32}	θ_{33}

4x4 Convolution

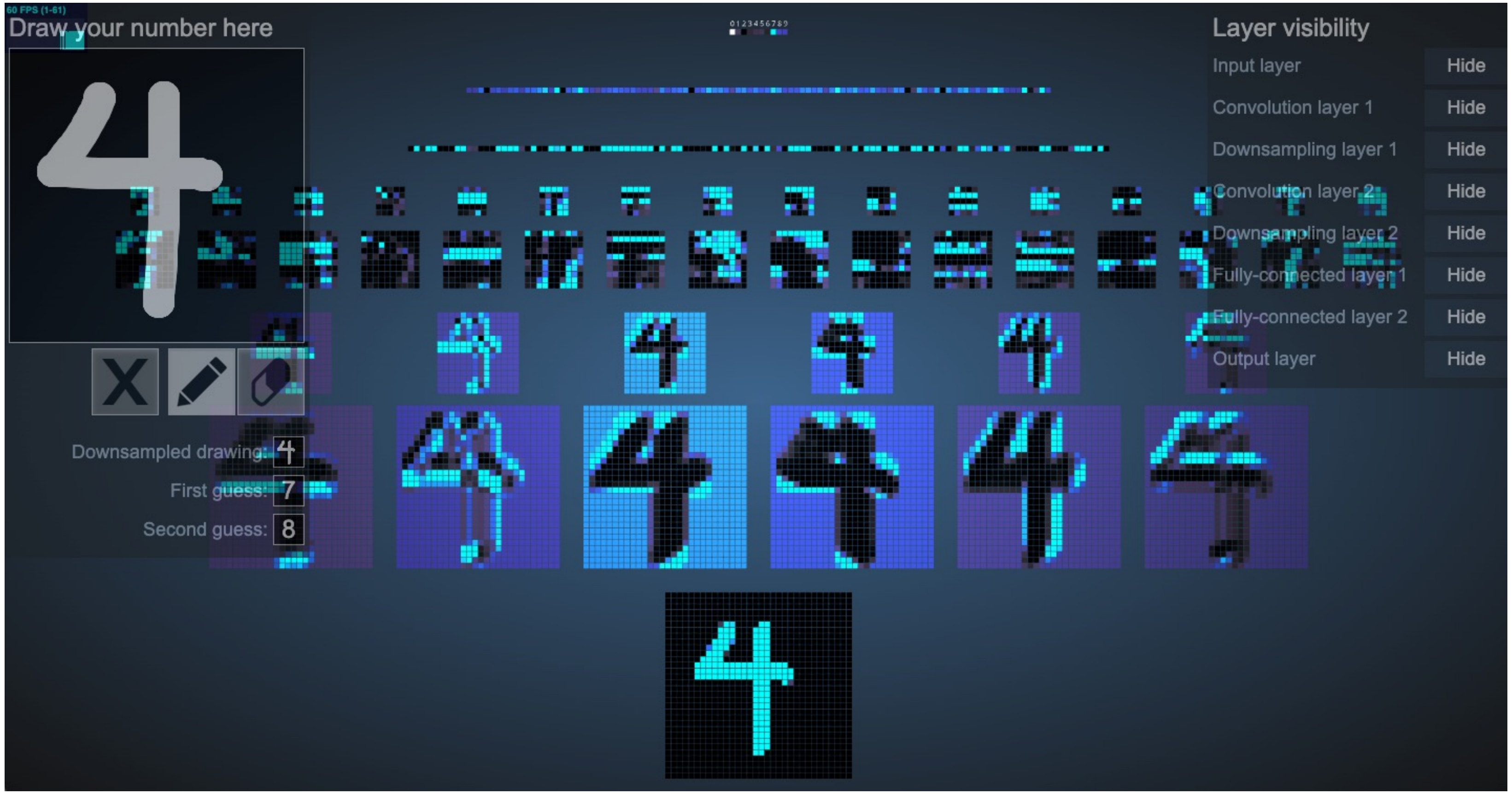
θ_{11}	θ_{12}	θ_{13}	θ_{14}
θ_{21}	θ_{22}	θ_{23}	θ_{24}
θ_{31}	θ_{32}	θ_{33}	θ_{34}
θ_{41}	θ_{42}	θ_{43}	θ_{44}

- A small kernel can only see a very small part of the image, but is fast to compute
- A large kernel can see more of the image, but at the expense of speed

CNN VISUALIZATIONS

Visualization of CNN

https://adamharley.com/nn_vis/cnn/2d.html



MNIST Digit Recognition with CNNs (in your browser)

https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

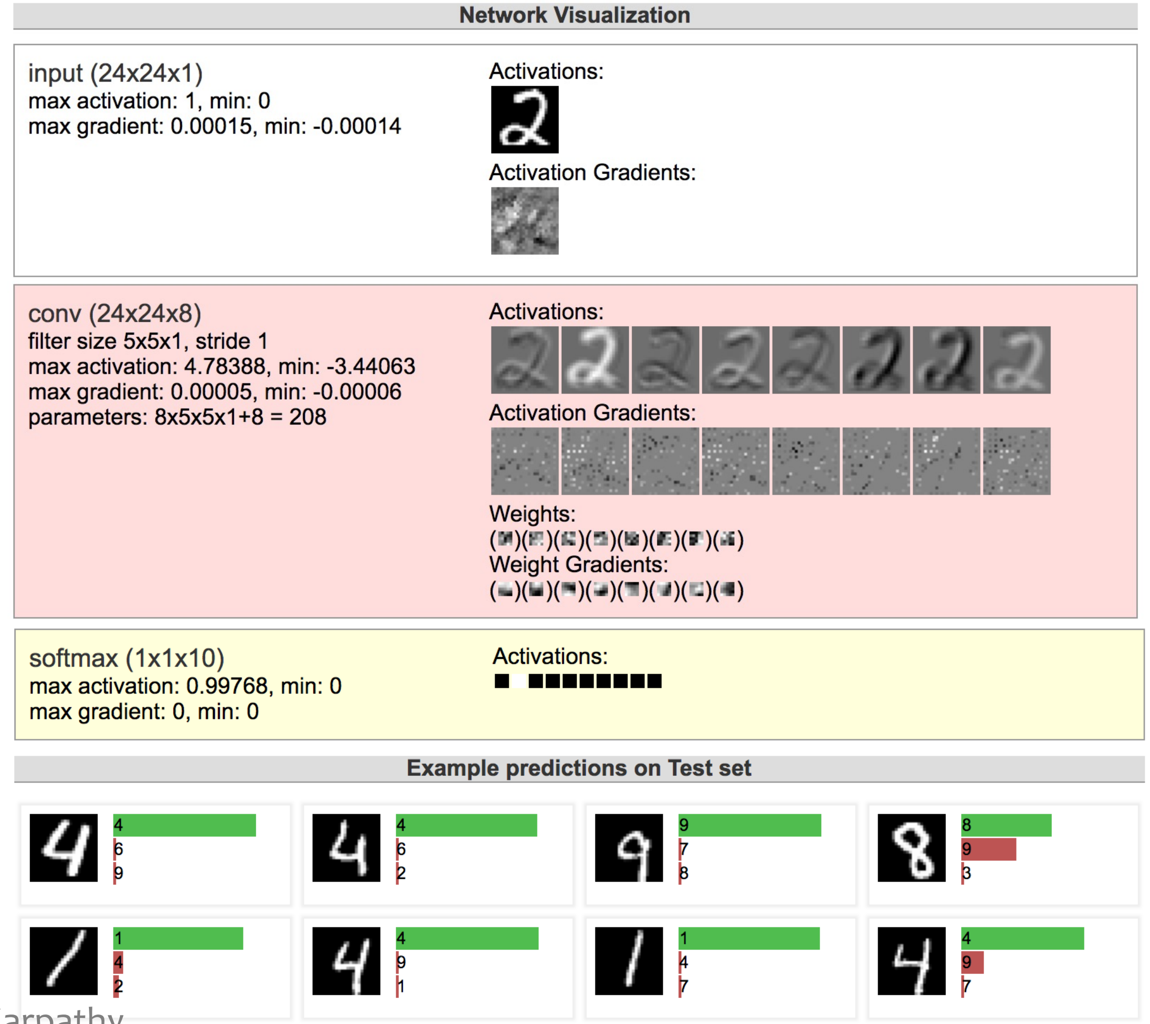


Figure from Andrej Karpathy

CNN Summary

CNNs

- Are used for all aspects of **computer vision**, and have won numerous pattern recognition competitions
- Able learn **interpretable features** at different levels of abstraction
- Typically, consist of **convolution** layers, **pooling** layers, **nonlinearities**, and **fully connected** layers

WORD EMBEDDINGS

Word Embeddings

Key Idea:

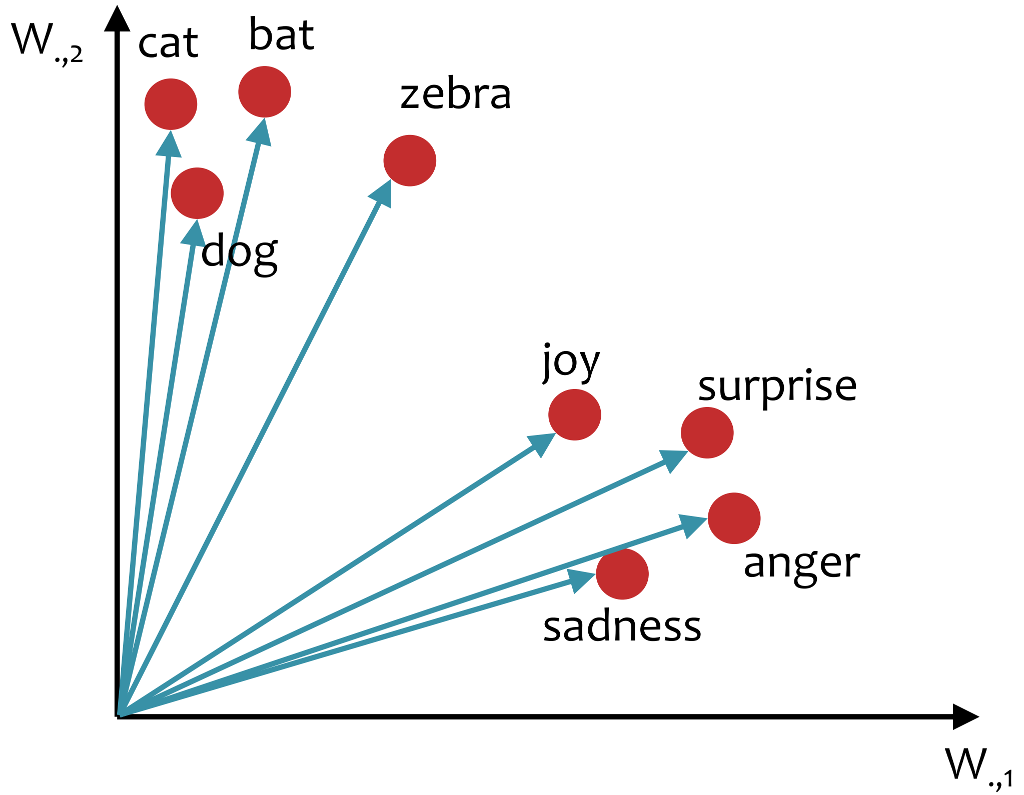
- represent each word in your vocabulary as a vector
- store as a V x D matrix where:
  V = number of words in vocab.
  D = vector's dimension

Modeling:

- define a model in which the vectors are parameters
- each copy of the word uses the same parameter vector
- train model so that similar words have high cosine similarity

W

anger	W_{11}	W_{12}
bat	W_{21}	W_{22}
cat	W_{31}	W_{32}
dog	W_{41}	W_{42}
joy	W_{51}	W_{52}
sadness	W_{61}	W_{62}
surprise	W_{71}	W_{72}
zebra	W_{81}	W_{82}



Word Embeddings

Key Idea:

- represent each word in your vocabulary as a vector
- store as a V x D matrix where:
  V = number of words in vocab.
  D = vector's dimension

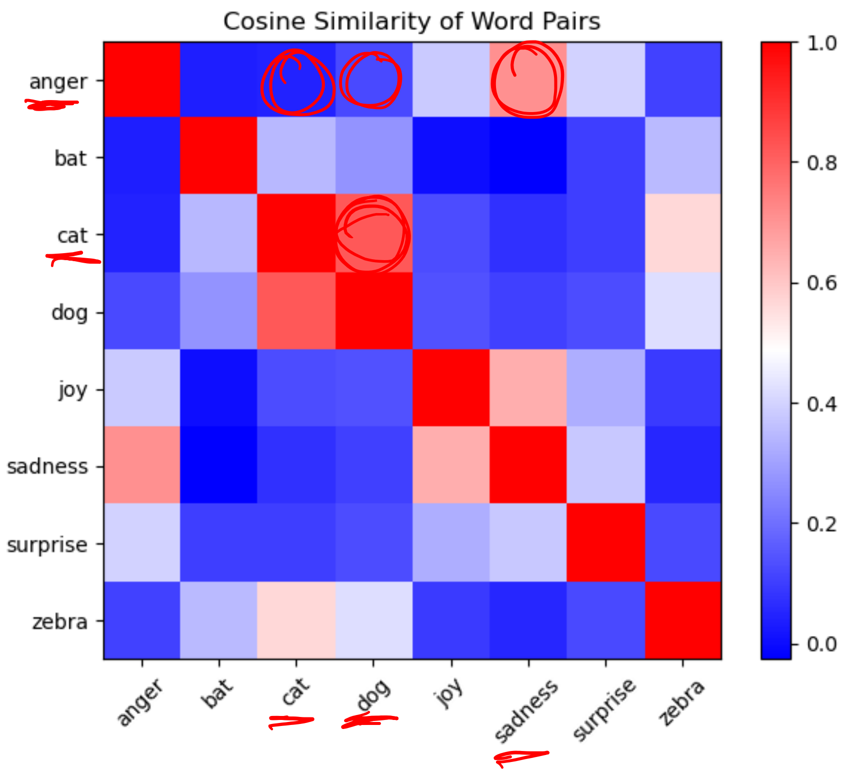
Modeling:

- define a model in which the vectors are parameters
- each copy of the word uses the same parameter vector
- train model so that similar words have high cosine similarity

W

aardvark	-2.3	0.0	-2.8	…	-4.5
anger	-2.8	-0.9	-1.7	…	-4.3
bat	-4.5	-1.3	0.6	…	-1.7
cat	3.5	-2.0	-2.3	…	-0.4
…				…	
joy	3.0	-0.6	-0.6	…	4.9
…				…	
zebra	-4.7	-4.2	-4.5	…	4.3

in a real use case, the typical embedding dimension is in the hundreds, e.g. D = 300

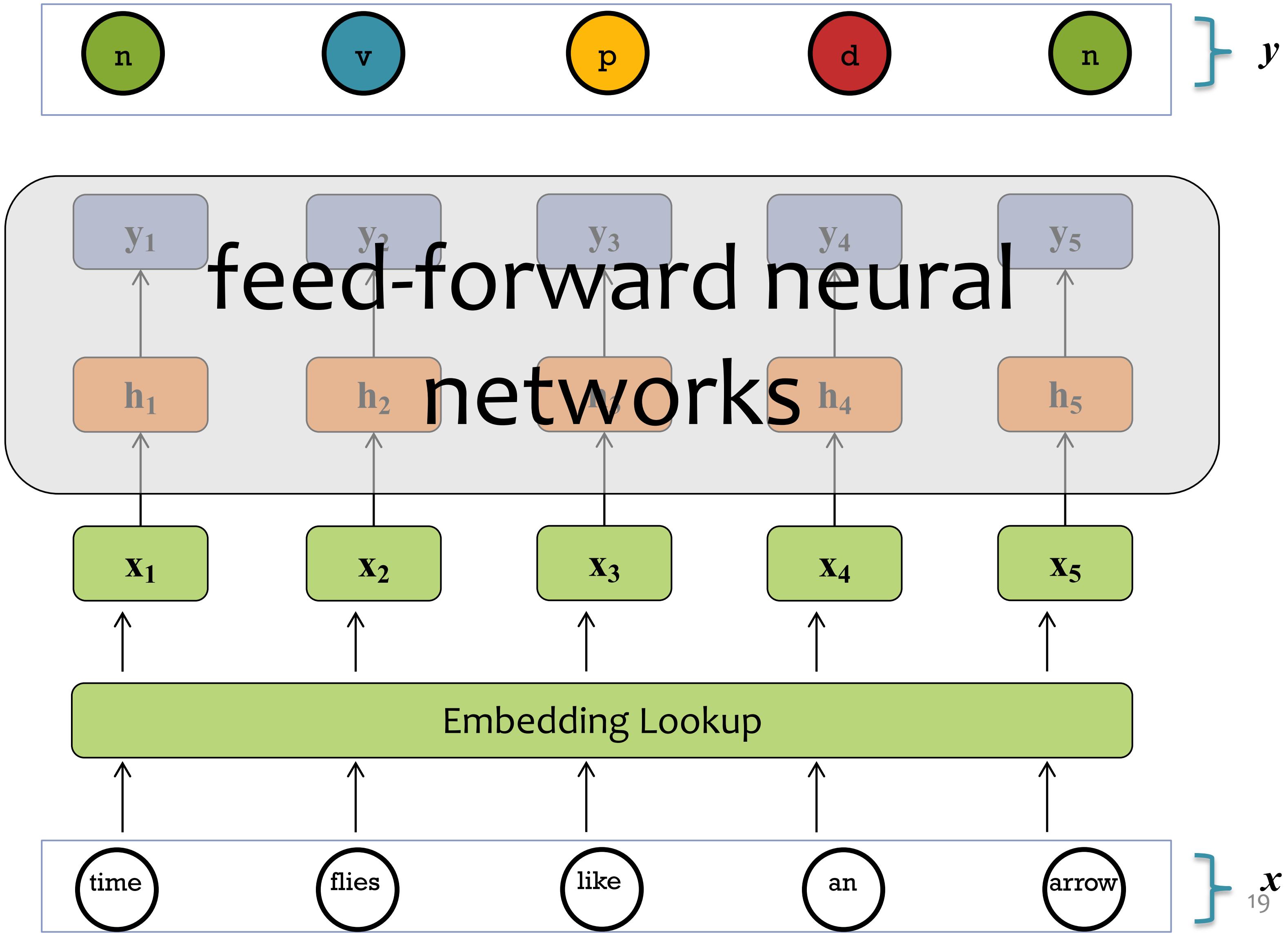


we can't visualize 300 dimensional vectors, but we can inspect their pairwise cosine similarities

Word Embeddings

In all the models we're about to consider (neural networks, RNNs, Transformers) that work with sentences…

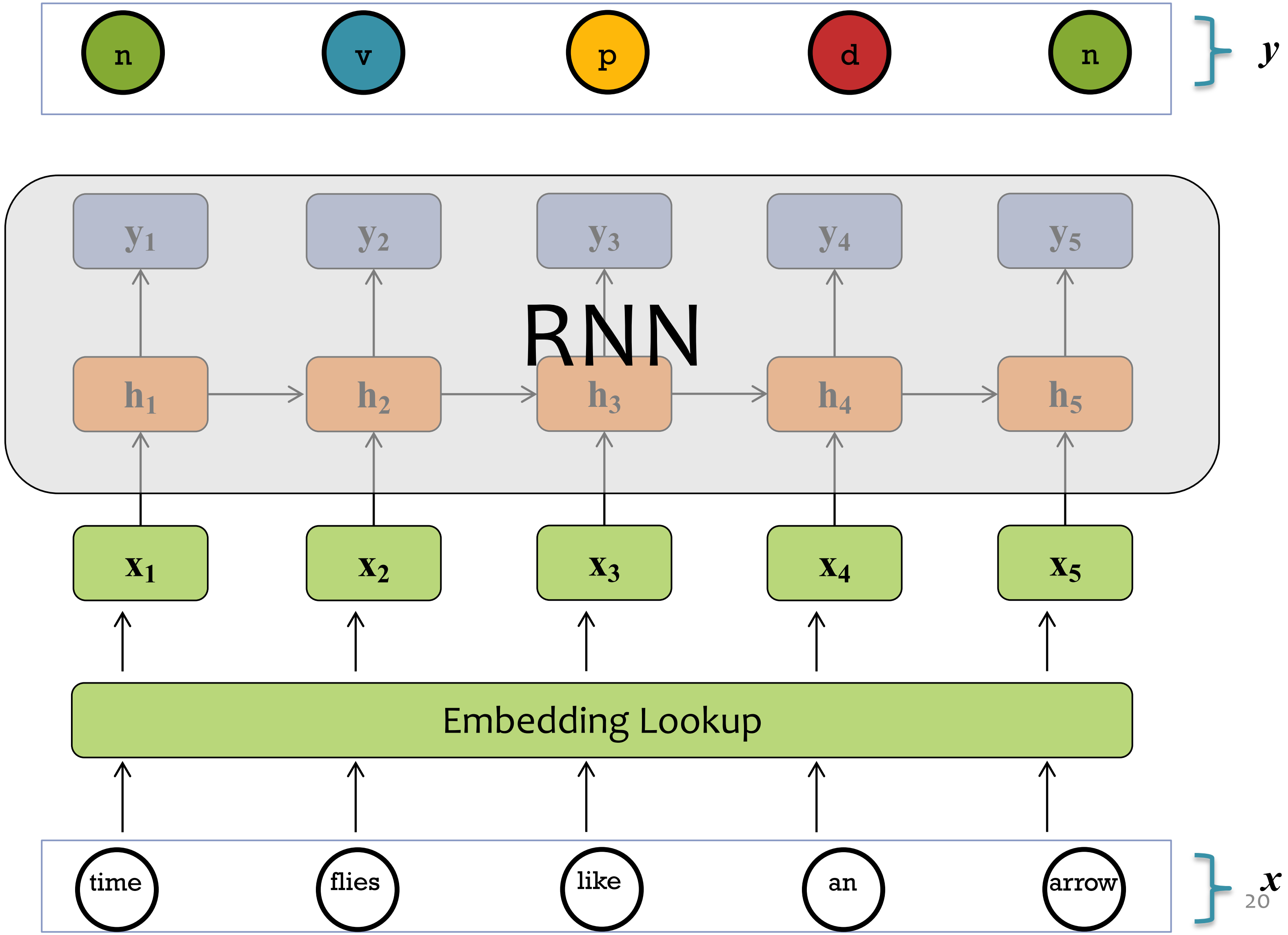
…the first step is always to look up the t'th word's embedding vector parameters and use said vector for the value of $\mathbf{x}_t$



Word Embeddings

In all the models we're about to consider (neural networks, RNNs, Transformers) that work with sentences…

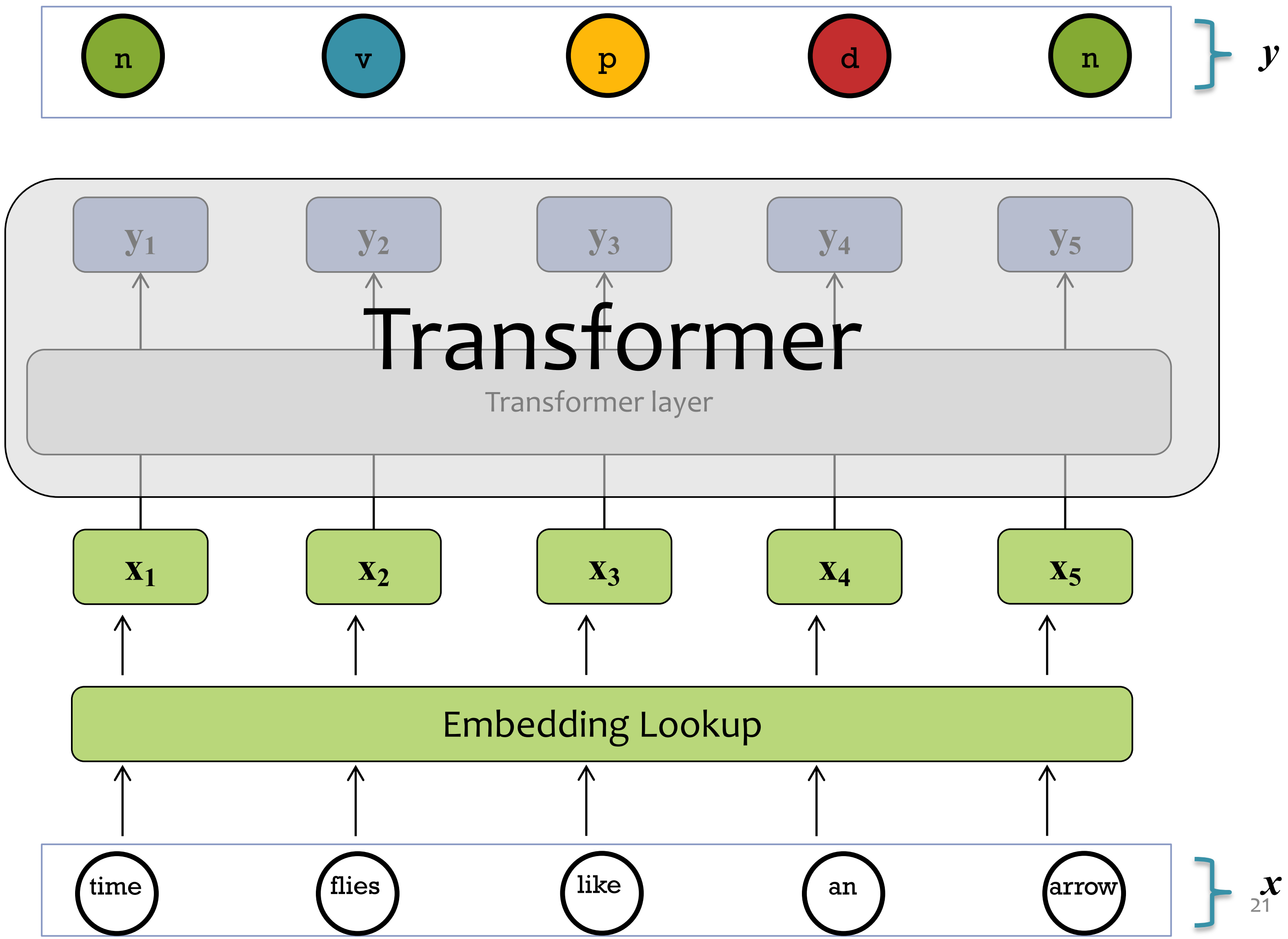
…the first step is always to look up the t'th word's embedding vector parameters and use said vector for the value of $\mathbf{x}_t$



Word Embeddings

In all the models we're about to consider (neural networks, RNNs, Transformers) that work with sentences…

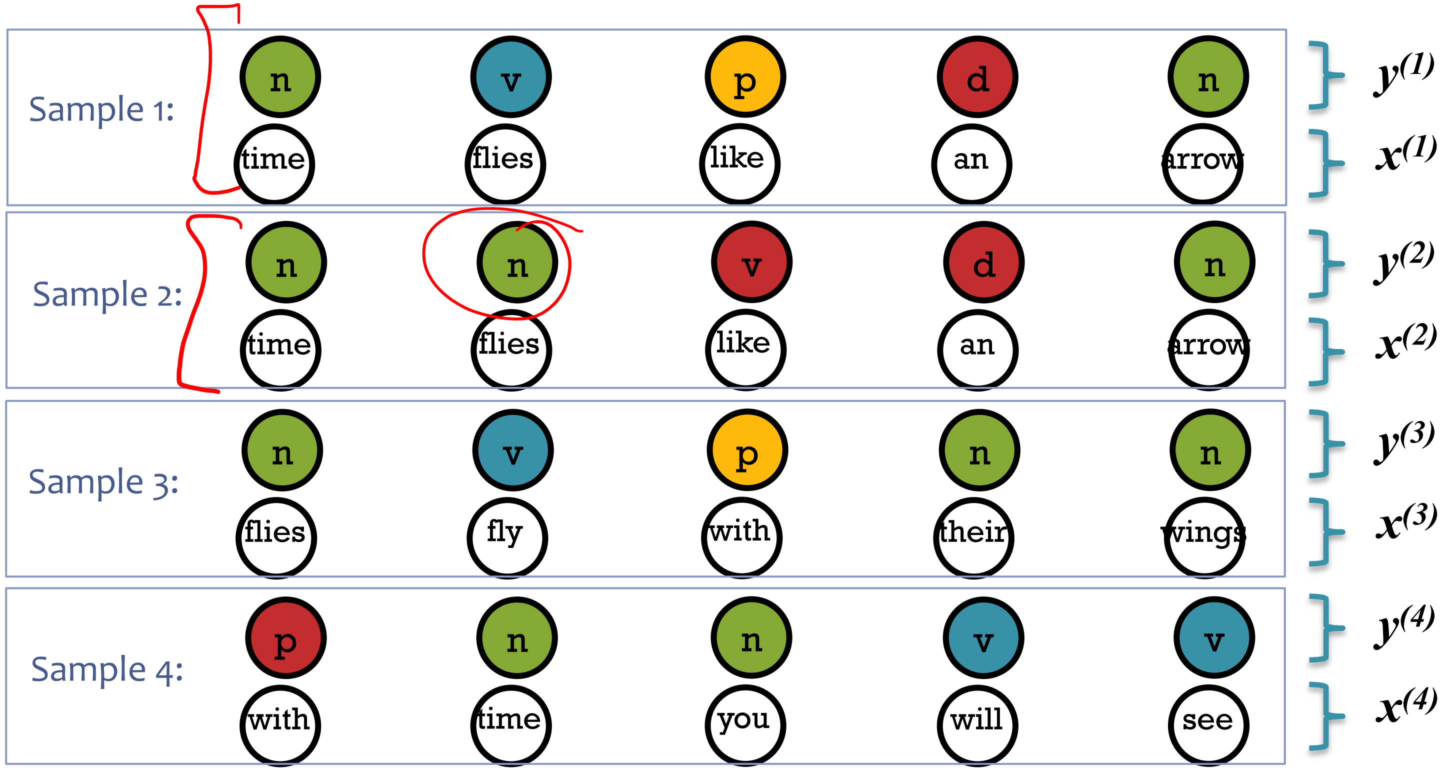
…the first step is always to look up the t'th word's embedding vector parameters and use said vector for the value of $\mathbf{x}_t$



SEQUENCE TAGGING

Dataset for Supervised Part-of-Speech (POS) Tagging

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



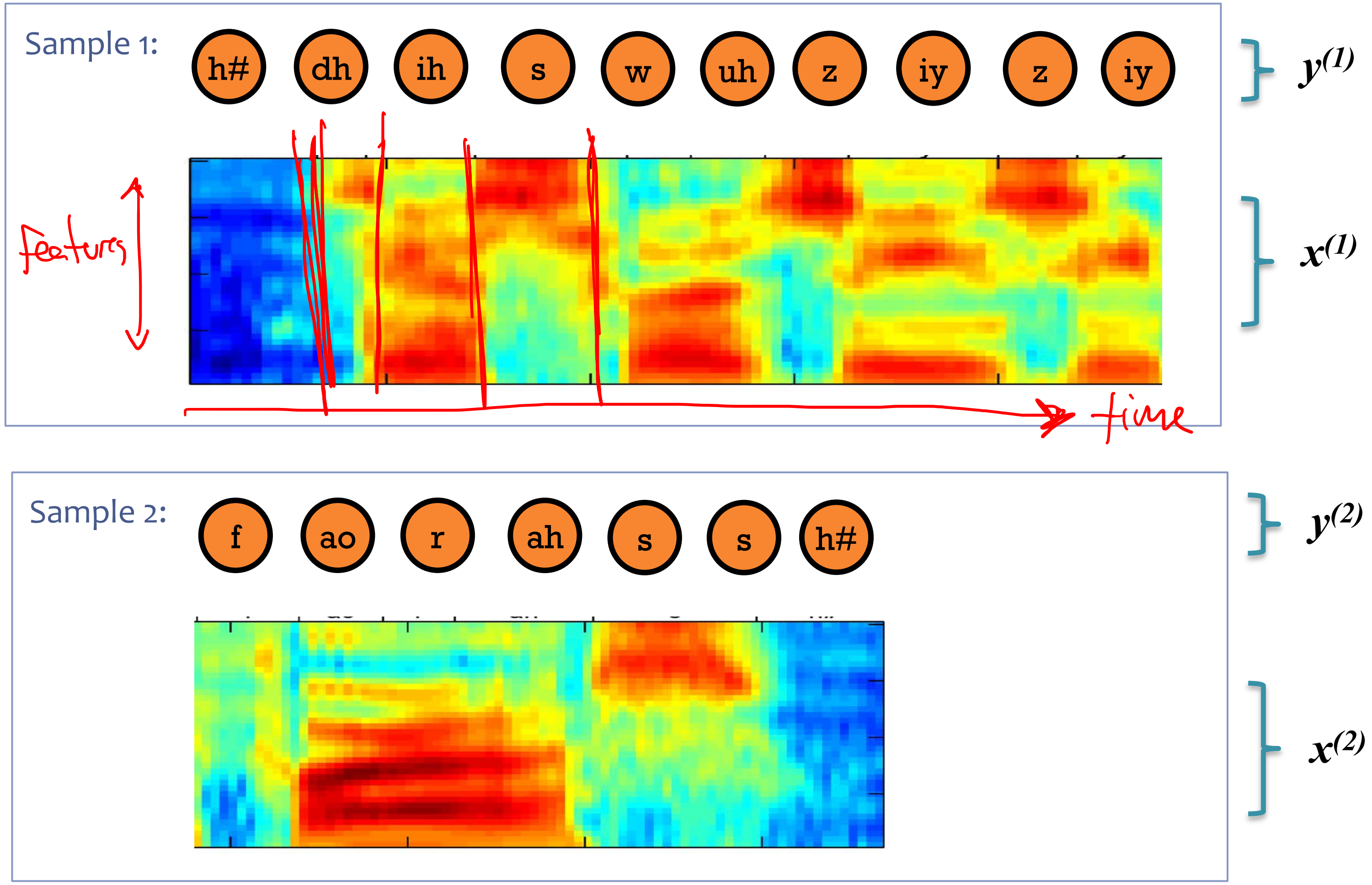
Dataset for Supervised Handwriting Recognition

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Figures from (Chatzis & Demiris, 2013)

Dataset for Supervised Phoneme (Speech) Recognition

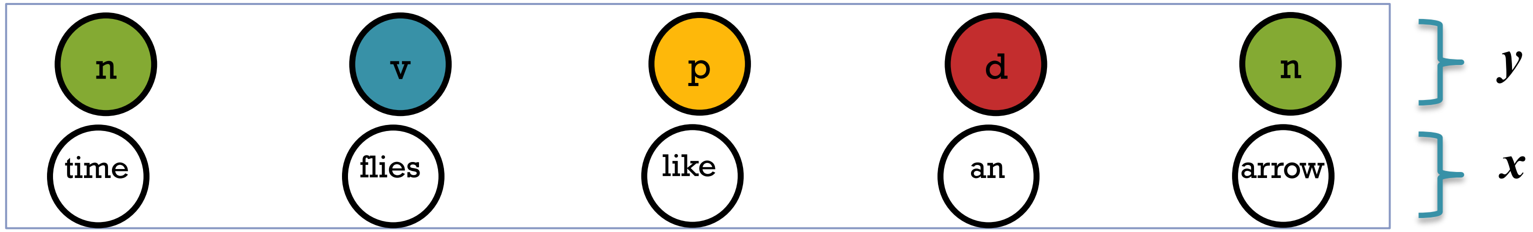
Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



Figures from (Jansen & Niyogi, 2013)

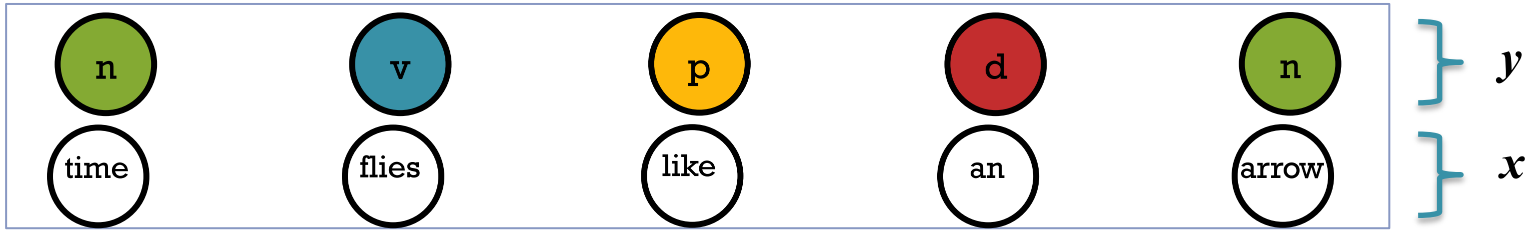
Time Series Data

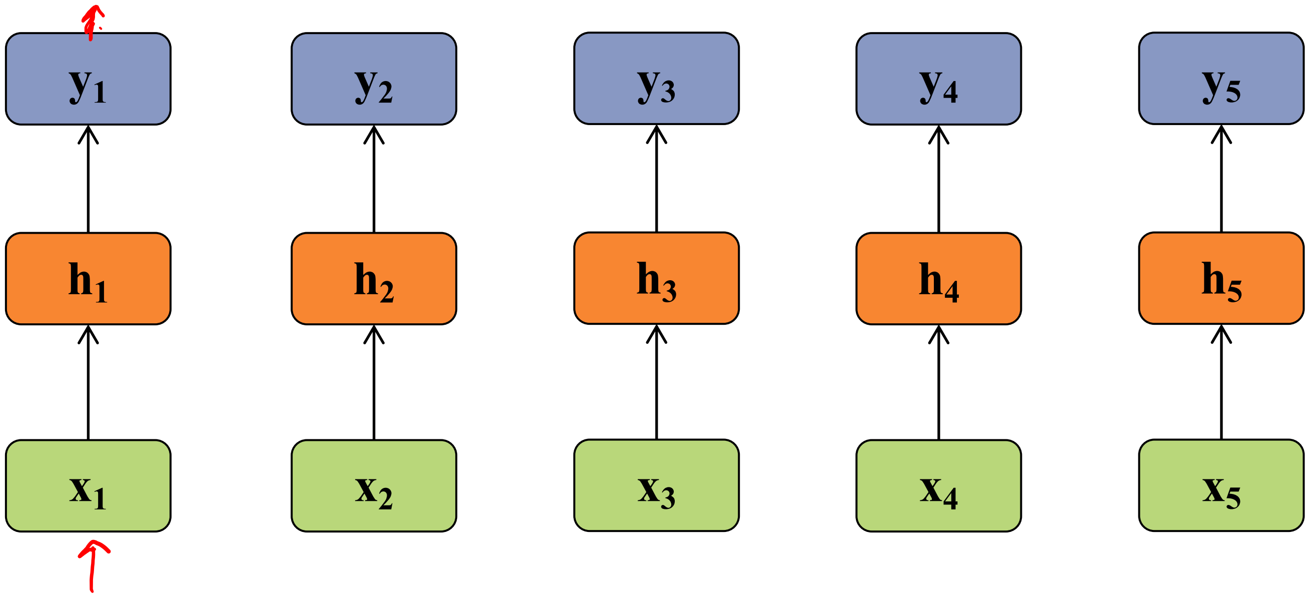
Question 1: How could we apply the neural networks we've seen so far (which expect **fixed size input/output**) to a prediction task with **variable length input/output**?



Time Series Data

Question 1: How could we apply the neural networks we've seen so far (which expect **fixed size input/output**) to a prediction task with **variable length input/output**?

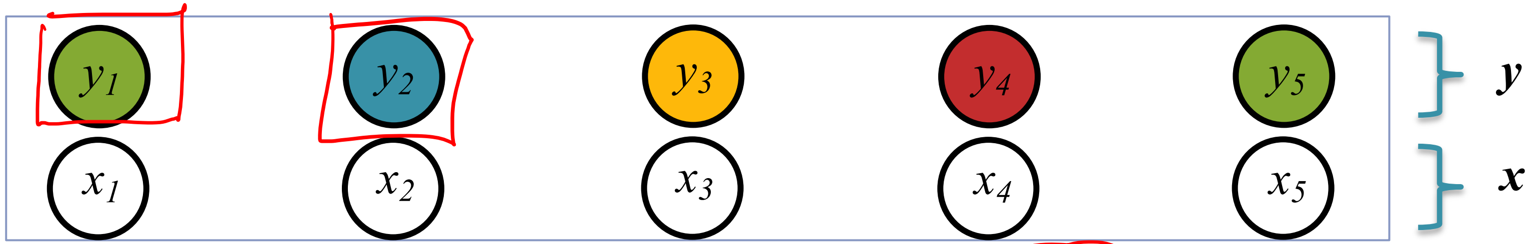




Time Series Data

Q1

Question 1: How could we incorporate context (e.g. words to the left/right, or tags to the left/right) into our solution?



Multiple Choice:

Working left-to-right, use features of…

	x_{i-1}	x_i	x_{i+1}	y_{i-1}	y_i	y_{i+1}
A	✓					
B				✓		
C	✓			✓		
D	✓			✓	✓	✓
E	✓	✓		✓	✓	✓
F	✓	✓	✓	✓		
G	✓	✓	✓	✓	✓	
H	✓	✓	✓	✓	✓	✓

B ← TOXIC

CORRECT → F, G

WRONG → H

$p(y=1|\vec{x}) = \sigma(\theta^T x)$

$p(y|x) = \sigma(\theta^T f(y,x))$

29%

16%

14%

RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNNs)

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

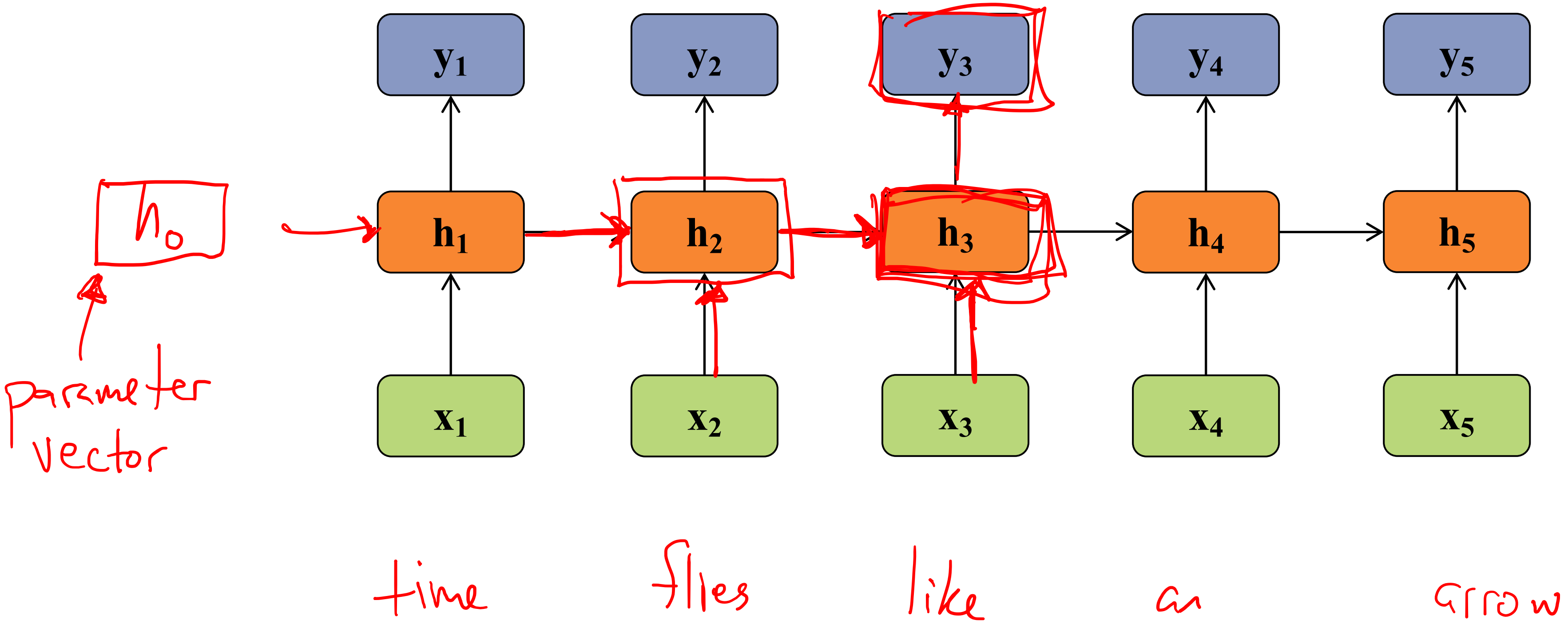
Definition of the RNN:

$\in \mathbb{R}^J$ $\quad \in \mathbb{R}^{J \times I}$ $\quad \in \mathbb{R}^{J \times J}$ $\quad \in \mathbb{R}^J$

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$y_t = W_{hy}h_t + b_y$$

$\mathbb{R}^{K \times J}$ $\quad \mathbb{R}^K$



$K = 10$

$J = 128$

$I = 300$

$T = 5$

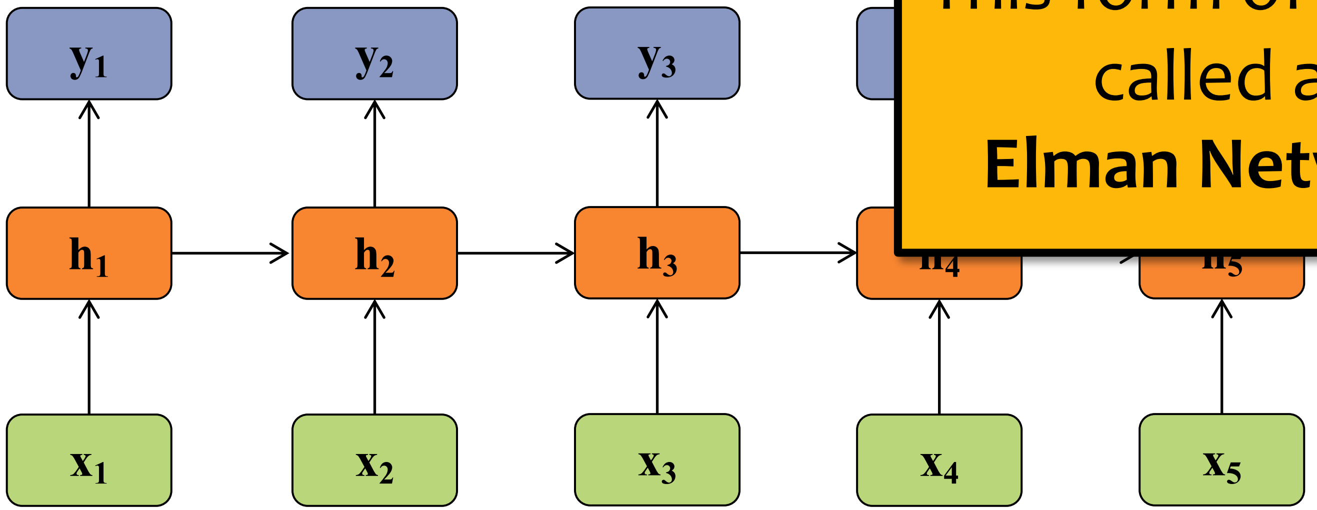
Recurrent Neural Networks (RNNs)

$$\begin{aligned} \text{inputs: } & \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I \\ \text{hidden units: } & \mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J \\ \text{outputs: } & \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K \\ \text{nonlinearity: } & \mathcal{H} \end{aligned}$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$
$$y_t = W_{hy}h_t + b_y$$

This form of RNN is called an **Elman Network**

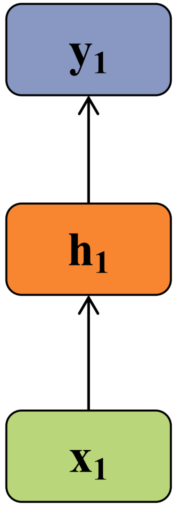


Recurrent Neural Networks (RNNs)

$$\text{inputs: } \mathbf{x} = (x_1, x_2, \dots, x_T), x_i \in \mathcal{R}^I$$
$$\text{hidden units: } \mathbf{h} = (h_1, h_2, \dots, h_T), h_i \in \mathcal{R}^J$$
$$\text{outputs: } \mathbf{y} = (y_1, y_2, \dots, y_T), y_i \in \mathcal{R}^K$$
$$\text{nonlinearity: } \mathcal{H}$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh} x_t + W_{hh} h_{t-1} + b_h\right)$$
$$y_t = W_{hy} h_t + b_y$$



- If $T=1$, then we have a standard feed-forward neural net with one hidden layer, which requires **fixed size inputs/outputs**
- By contrast, an RNN can handle arbitrary length inputs/outputs because T can vary from example to example
- The key idea is that we reuse the same parameters at every timestep, always building off of the previous hidden state

Background

A Recipe for Machine Learning

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose each of these:

– Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

– Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

4. Train with SGD:

(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

Background

A Recipe for Machine Learning

- Recurrent Neural Networks (RNNs) provide another form of **decision function**
- An RNN is just another differential function

2. Choose each of these:
   - Decision function

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

4. Train with SGD:

(take small steps opposite the gradient)

$$\eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

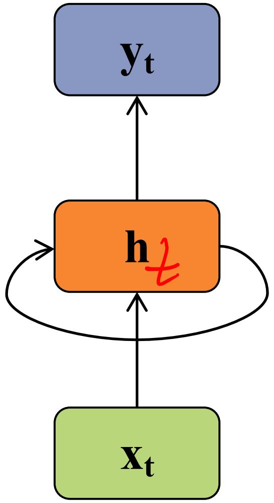
- We'll just need a method of computing the gradient efficiently
- Let's use Backpropagation Through Time...

Recurrent Neural Networks (RNNs)

$$\text{inputs: } \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$$
$$\text{hidden units: } \mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$$
$$\text{outputs: } \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$$
$$\text{nonlinearity: } \mathcal{H}$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$
$$y_t = W_{hy}h_t + b_y$$



Recurrent Neural Networks (RNNs)

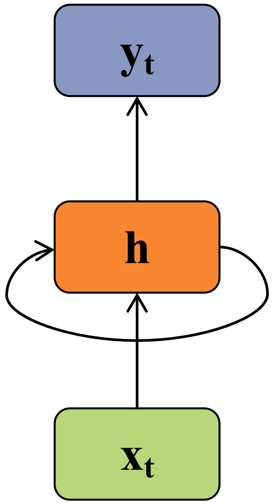
$$
\begin{aligned}
\text{inputs: } & \mathbf{x} = (x_1, x_2, \dots, x_T), x_i \in \mathcal{R}^I \\
\text{hidden units: } & \mathbf{h} = (h_1, h_2, \dots, h_T), h_i \in \mathcal{R}^J \\
\text{outputs: } & \mathbf{y} = (y_1, y_2, \dots, y_T), y_i \in \mathcal{R}^K \\
\text{nonlinearity: } & \mathcal{H}
\end{aligned}
$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

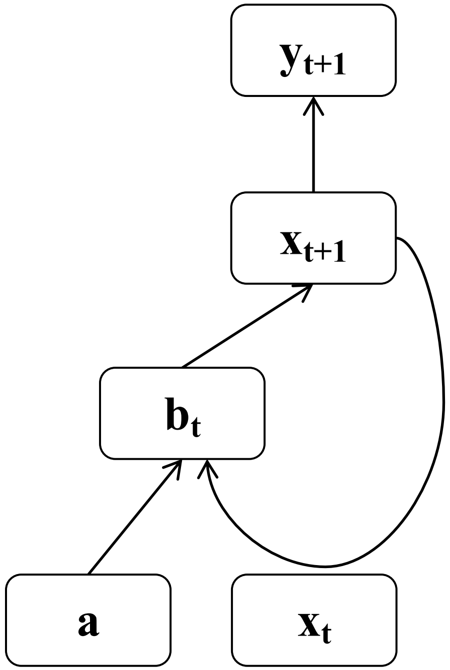
$$y_t = W_{hy}h_t + b_y$$

- By unrolling the RNN through time, we can **share parameters** and accommodate **arbitrary length** input/output pairs
- Applications: **time-series data** such as sentences, speech, stock-market, signal data, etc.



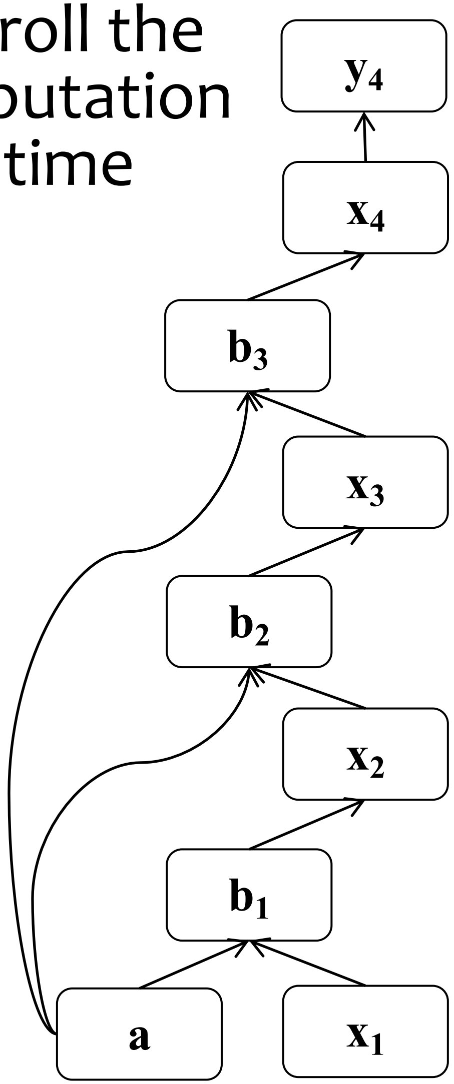
Background: Backprop through time

Recurrent neural network:



BPTT:

1. Unroll the computation over time



2. Run backprop through the resulting feed-forward network

(Robinson & Fallside, 1987)
(Werbos, 1988)
(Mozer, 1995)

Bidirectional RNN

$$\text{inputs: } \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$$

$$\text{hidden units: } \overrightarrow{\mathbf{h}} \text{ and } \overleftarrow{\mathbf{h}}$$

$$\text{outputs: } \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$$

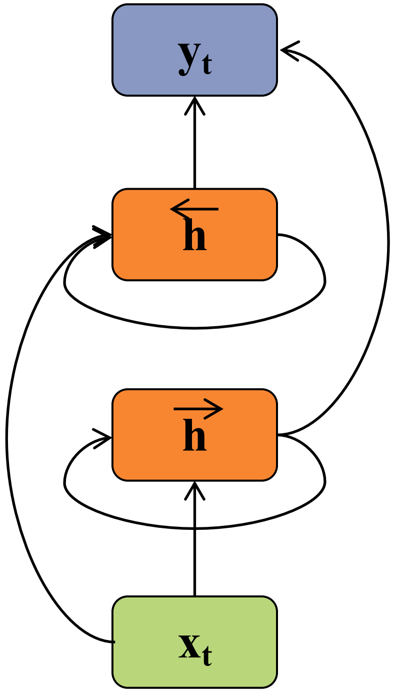
$$\text{nonlinearity: } \mathcal{H}$$

Recursive Definition:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Bidirectional RNN

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

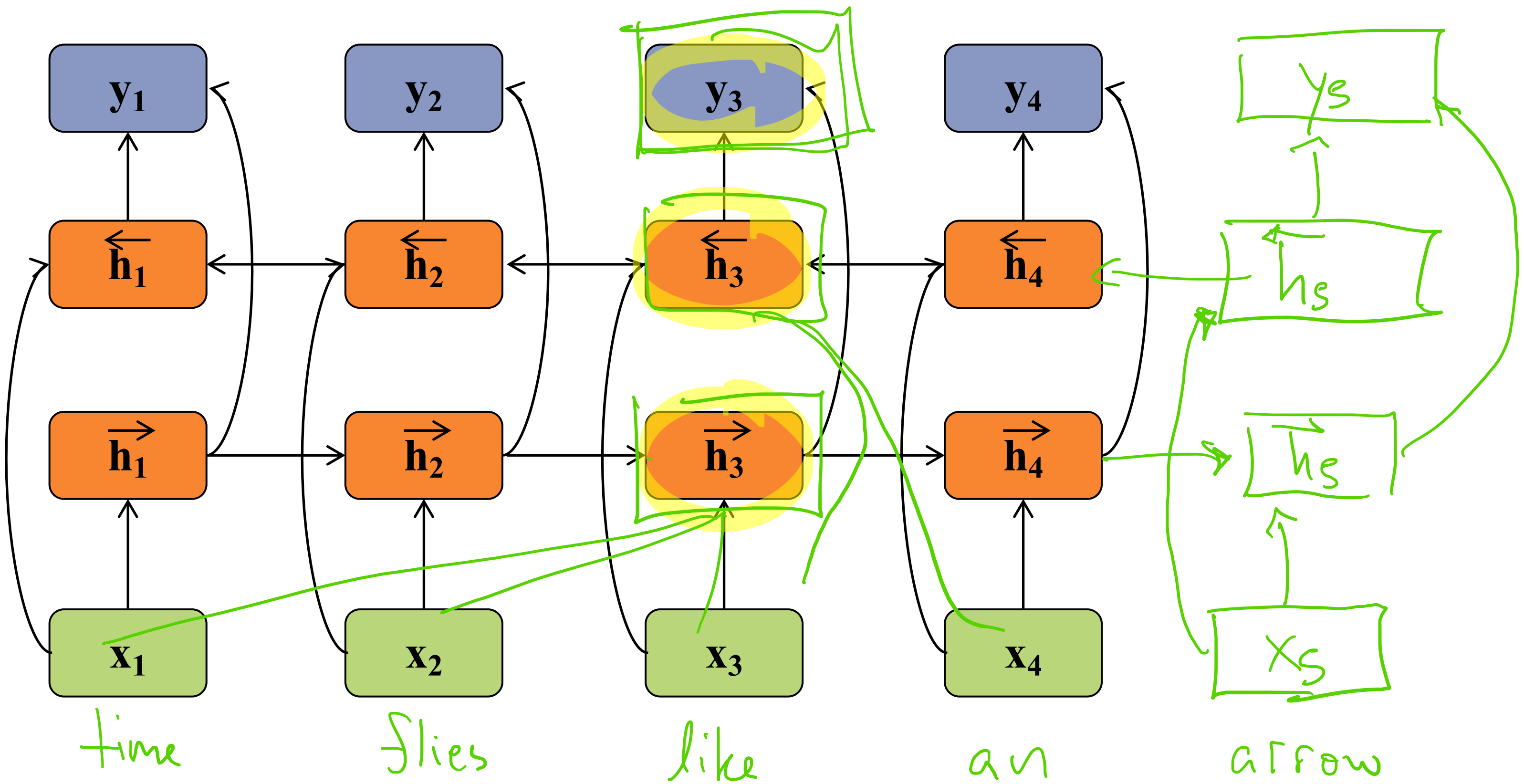
nonlinearity: $\mathcal{H}$

Recursive Definition:

$$\boxed{\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)}$$

$$\boxed{\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)}$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Bidirectional RNN

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

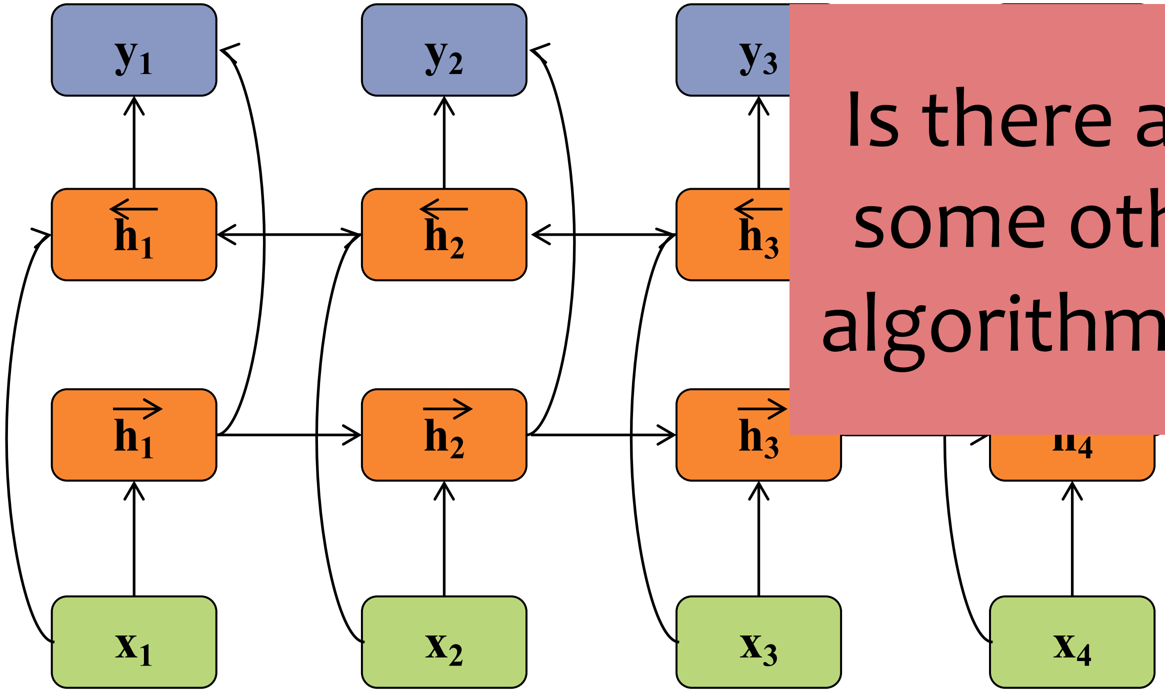
nonlinearity: $\mathcal{H}$

Recursive Definition:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$



Is there an analogy to some other recursive algorithm(s) we know?

Deep RNNs

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

Recursive Definition:

$$h_t^n = \mathcal{H}\left(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n\right)$$

$$y_t = W_{h^N y} h_t^N + b_y$$

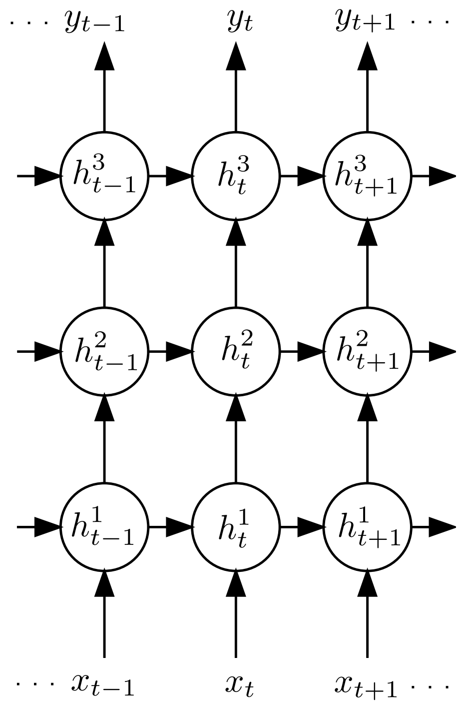


Figure from (Graves et al., 2013)

Deep Bidirectional RNNs

$$\text{inputs: } \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$$
$$\text{outputs: } \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$$
$$\text{nonlinearity: } \mathcal{H}$$

- Notice that the upper level hidden units have input from **two previous layers** (i.e. wider input)
- Likewise for the output layer
- What analogy can we draw to DNNs, DBNs, DBMs?

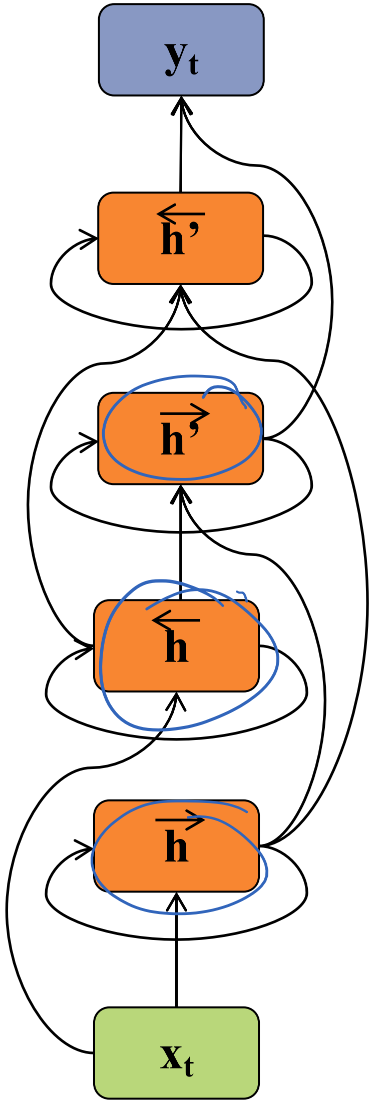


Figure from (Graves et al., 2013)

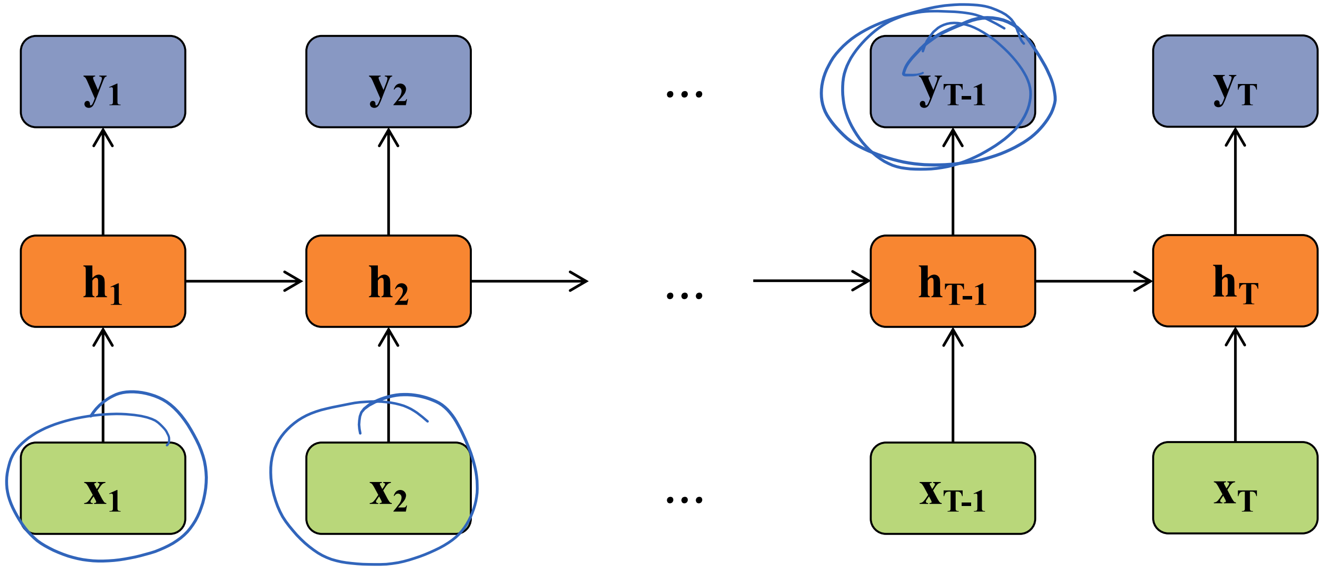
LSTMS

RNNs and Forgetting

Long Short-Term Memory (LSTM)

Motivation:

- Standard RNNs have trouble learning long distance dependencies
- LSTMs combat this issue



Long Short-Term Memory (LSTM)

$$h_t = \mathcal{H}(\underbrace{W_{xh}x_t} + W_{hh}h_{t-1} + b_h)$$

Motivation:

- Vanishing gradient problem for Standard RNNs
- Figure shows sensitivity (darker = more sensitive) to the input at time t=1

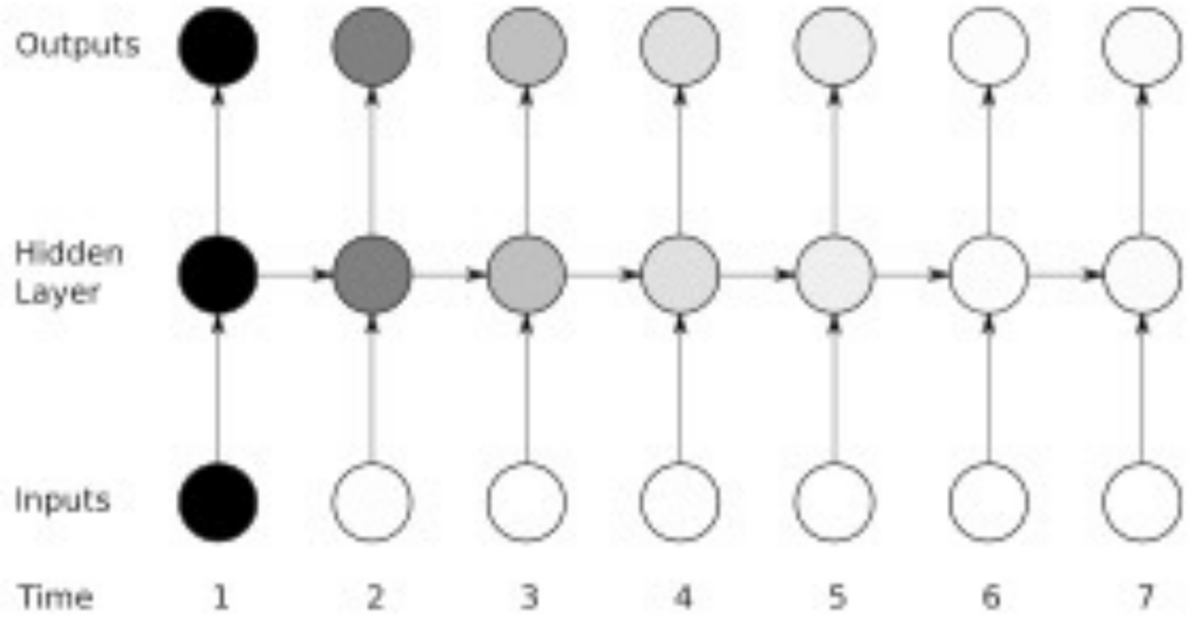


Figure from (Graves, 2012)

Long Short-Term Memory (LSTM)

Motivation:

- LSTM units have a rich internal structure
- The various “gates” determine the propagation of information and can choose to “remember” or “forget” information

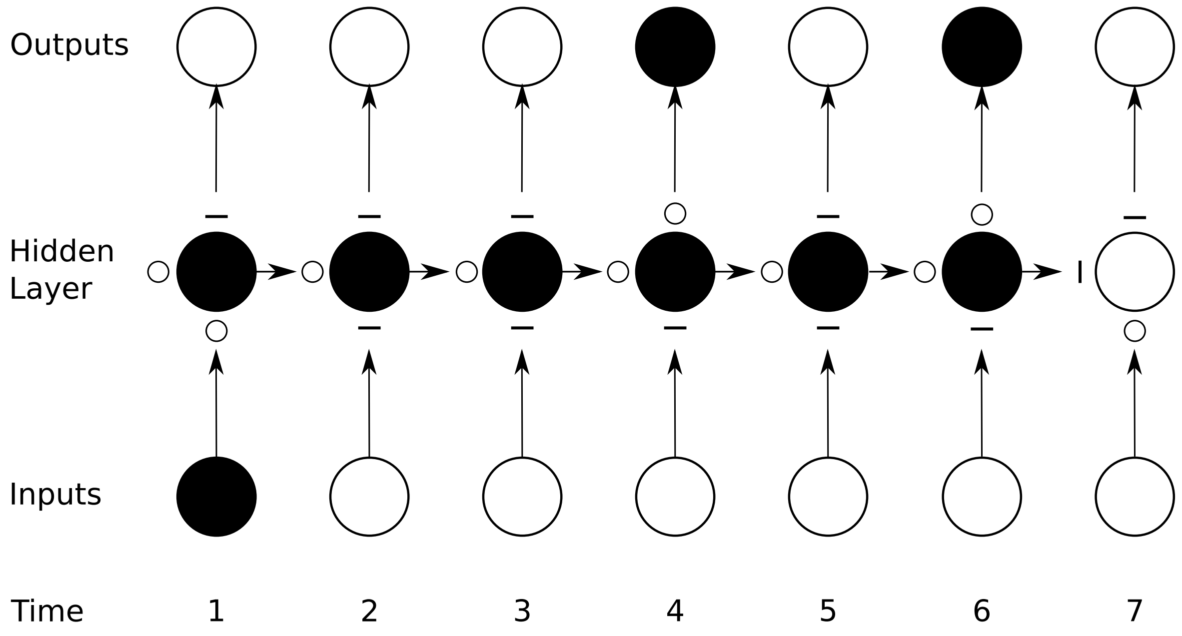
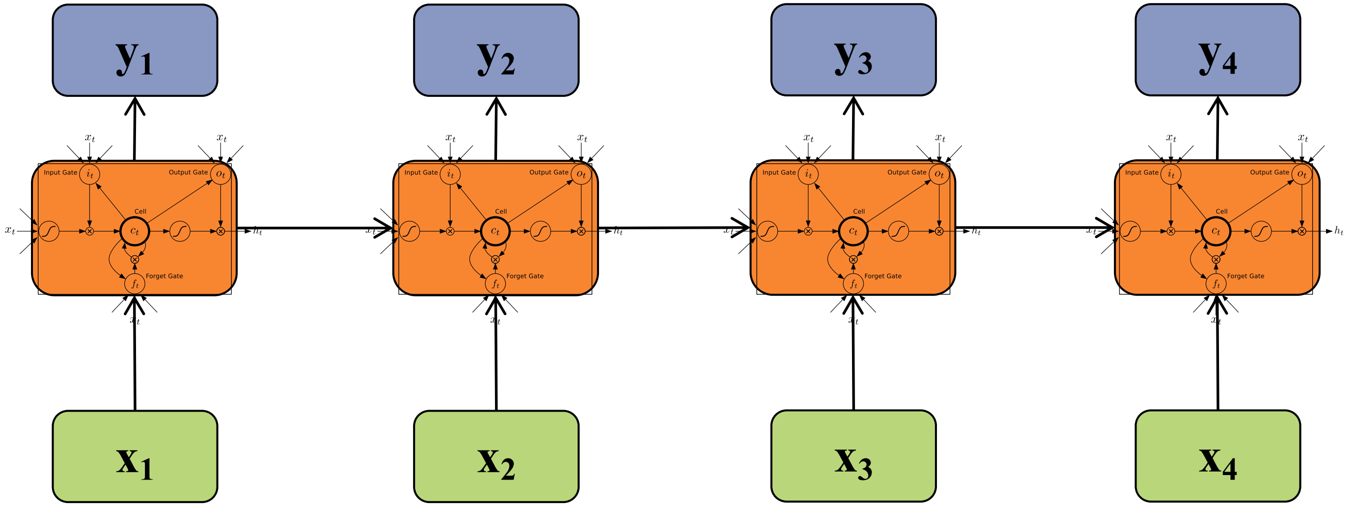


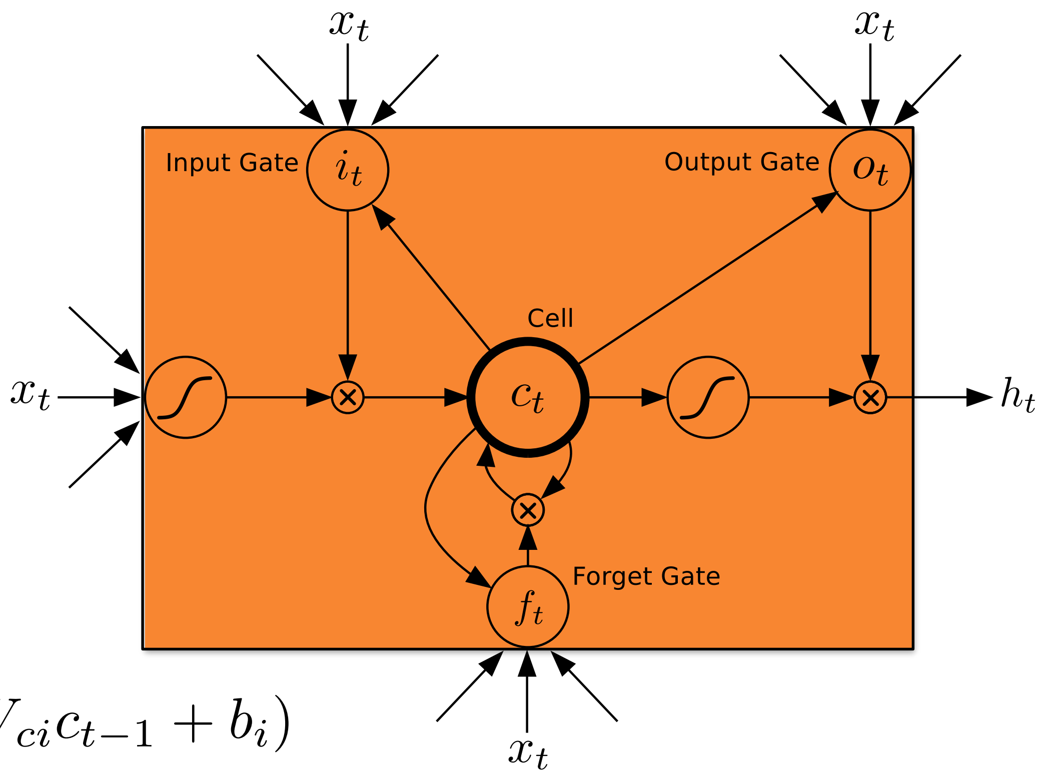
Figure from (Graves, 2012)

Long Short-Term Memory (LSTM)



Long Short-Term Memory (LSTM)

- **Input gate:** masks out the standard RNN inputs
- **Forget gate:** masks out the previous cell
- **Cell:** stores the input/forget mixture
- **Output gate:** masks out the values of the next hidden



$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)$$

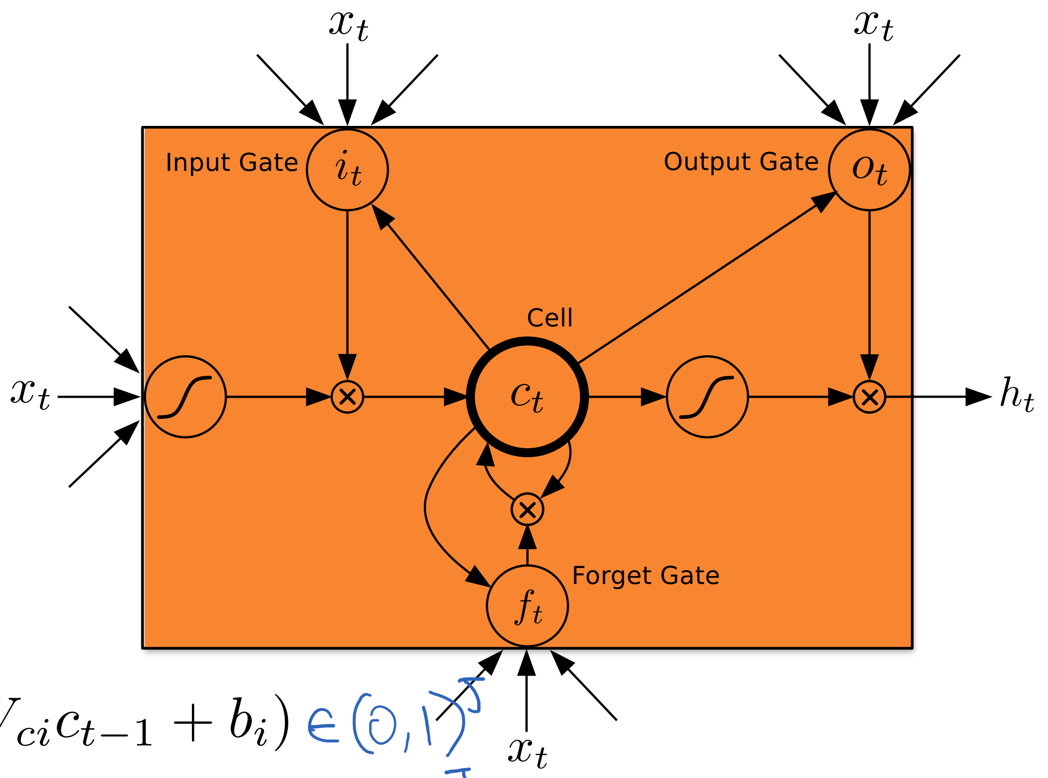
$$h_t = o_t \tanh(c_t)$$

$$y_t = W_{yh} h_t + b_y$$

Figure from (Graves et al., 2013)

Long Short-Term Memory (LSTM)

- **Input gate:** masks out the standard RNN inputs
- **Forget gate:** masks out the previous cell
- **Cell:** stores the input/forget mixture
- **Output gate:** masks out the values of the next hidden



The cell is the LSTM's long term memory, and helps control information flow over time steps

The hidden state is the output of the LSTM cell

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \in (0,1)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \in (0,1)$$

$$\boxed{c_t} = f_t c_{t-1} + i_t \boxed{\tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)}$$

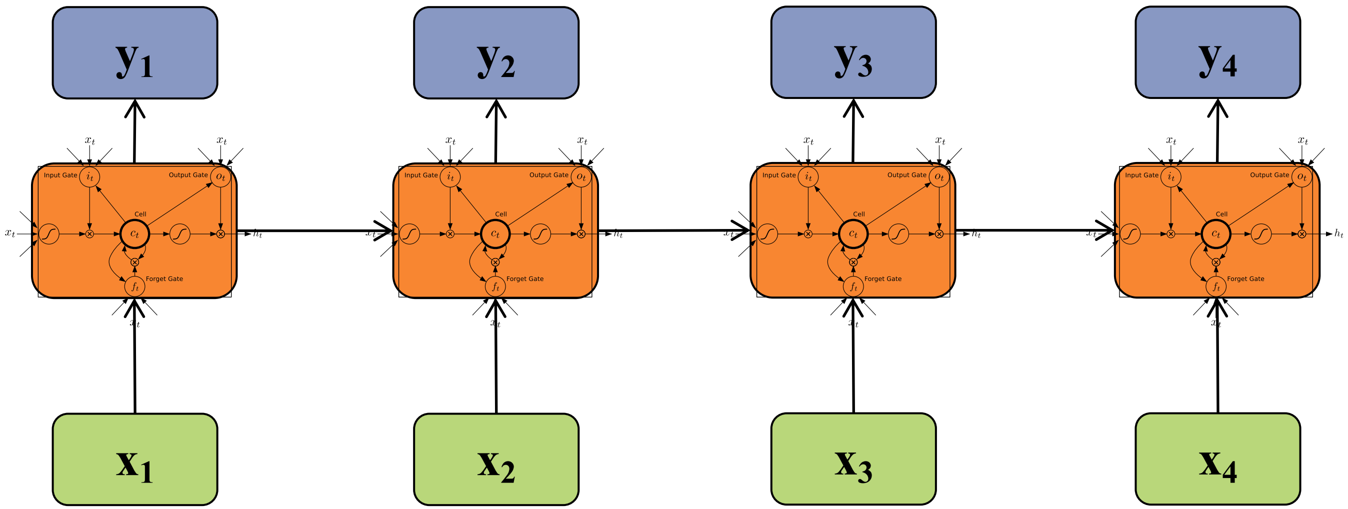
$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)$$

$$\boxed{h_t} = o_t \tanh(c_t)$$

Identical to the Elman's networks hidden state

Figure from (Graves et al., 2013)

Long Short-Term Memory (LSTM)



Deep Bidirectional LSTM (DBLSTM)

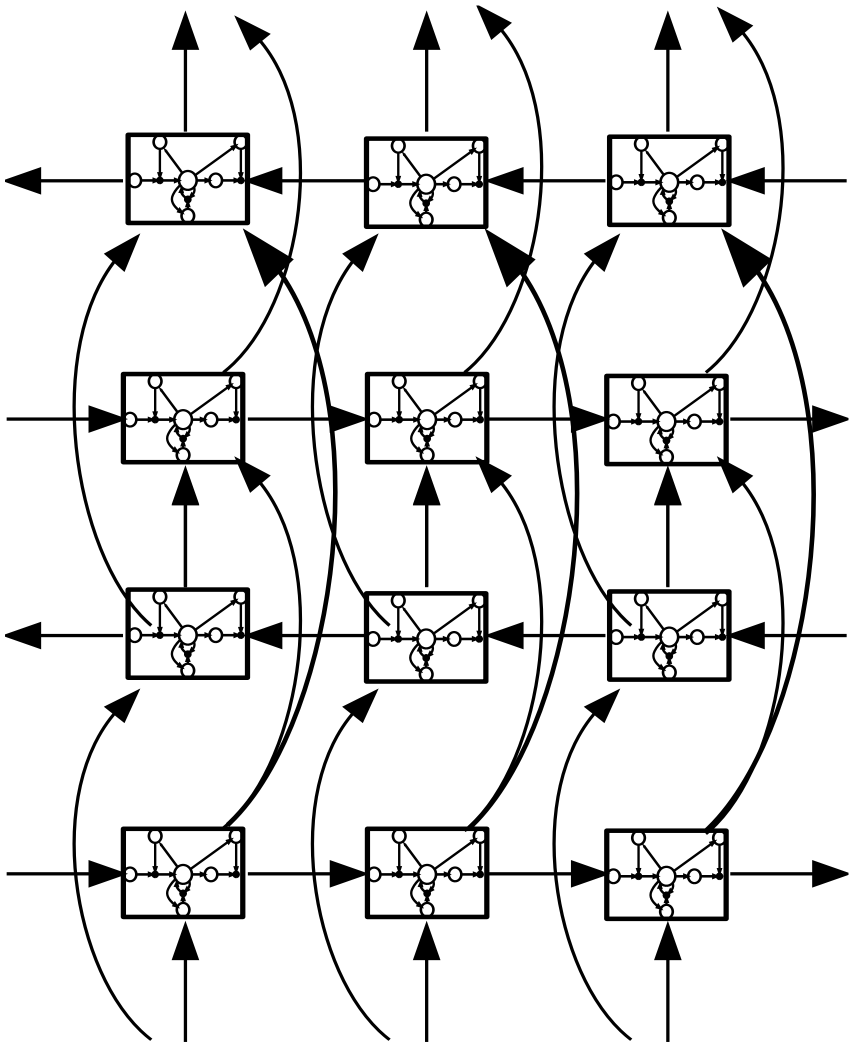
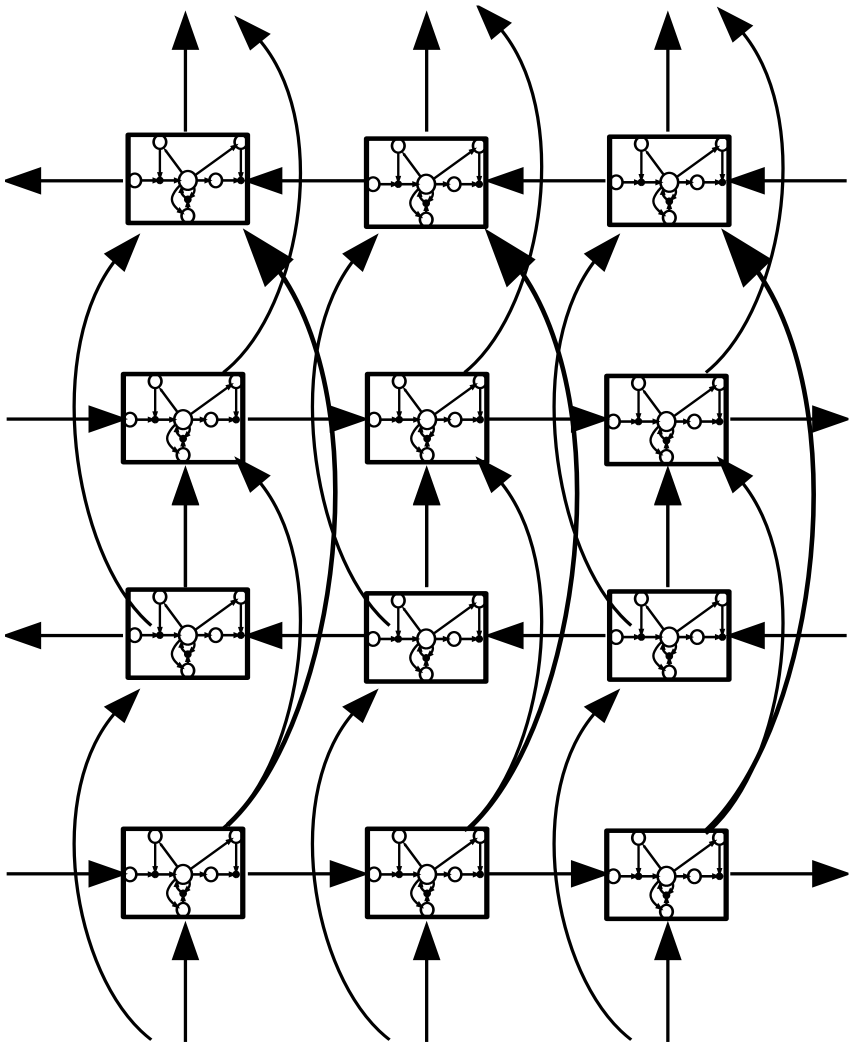


Figure from (Graves et al., 2013)

- Figure: input/output layers not shown
- **Same general topology** as a Deep Bidirectional RNN, but with **LSTM units** in the hidden layers
- No additional **representational power** over DBRNN, but **easier to learn** in practice

Deep Bidirectional LSTM (DBLSTM)



How important is this particular architecture?

Jozefowicz et al. (2015) **evaluated 10,000 different LSTM-like architectures** and found several variants that worked just as well on several tasks.

Figure from (Graves et al., 2013)

Why not just use LSTMs for everything?

Everyone did, for a time.

But…

1. They still have **difficulty** with **long-range dependencies**
2. Their computation is **inherently serial**, so can't be easily parallelized on a GPU
3. Even though they (mostly) solve the vanishing gradient problem, they can still suffer from **exploding gradients**

RNN / LSTM RESULTS

Dataset for Supervised Named Entity Recognition (NER)

- **Goal:** label the spans of persons, locations, organizations, times, etc. (aka. entities)
- **Data Representation:** to cast as a sequence tagging problem, we use Begin-Inside-Outside (BIO) tagging
- BIO tags distinguish between adjacent entities of the same type

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Sample 1:	B-PER	I-PER	O	B-LOC	I-LOC		$\boldsymbol{y}^{(1)}$
	Tenzing	Norgay PER	climbed	Mount	Everest LOC		$\boldsymbol{x}^{(1)}$
Sample 2:	B-PER	O	B-LOC	I-LOC			$\boldsymbol{y}^{(2)}$
	Obama	visits	Paris	France			$\boldsymbol{x}^{(2)}$
Sample 3:	B-PER	I-PER	B-ORG	I-ORG	O	O	$\boldsymbol{y}^{(3)}$
	Steve	Jobs'	Apple	Inc.	changed	tech	$\boldsymbol{x}^{(3)}$
Sample 4:	B-LOC	B-LOC	O	O			$\boldsymbol{y}^{(4)}$
	Spain LOC	Italy LOC	win	medals			$\boldsymbol{x}^{(4)}$

LSTM Empirical Results

- CoNLL-2003 is the most prominent dataset for NER
- F1 – higher is better
- blue dots are methods that use an LSTM
- an LSTM is the primary model behind the state-of-the-art (*ACE + document-context*)

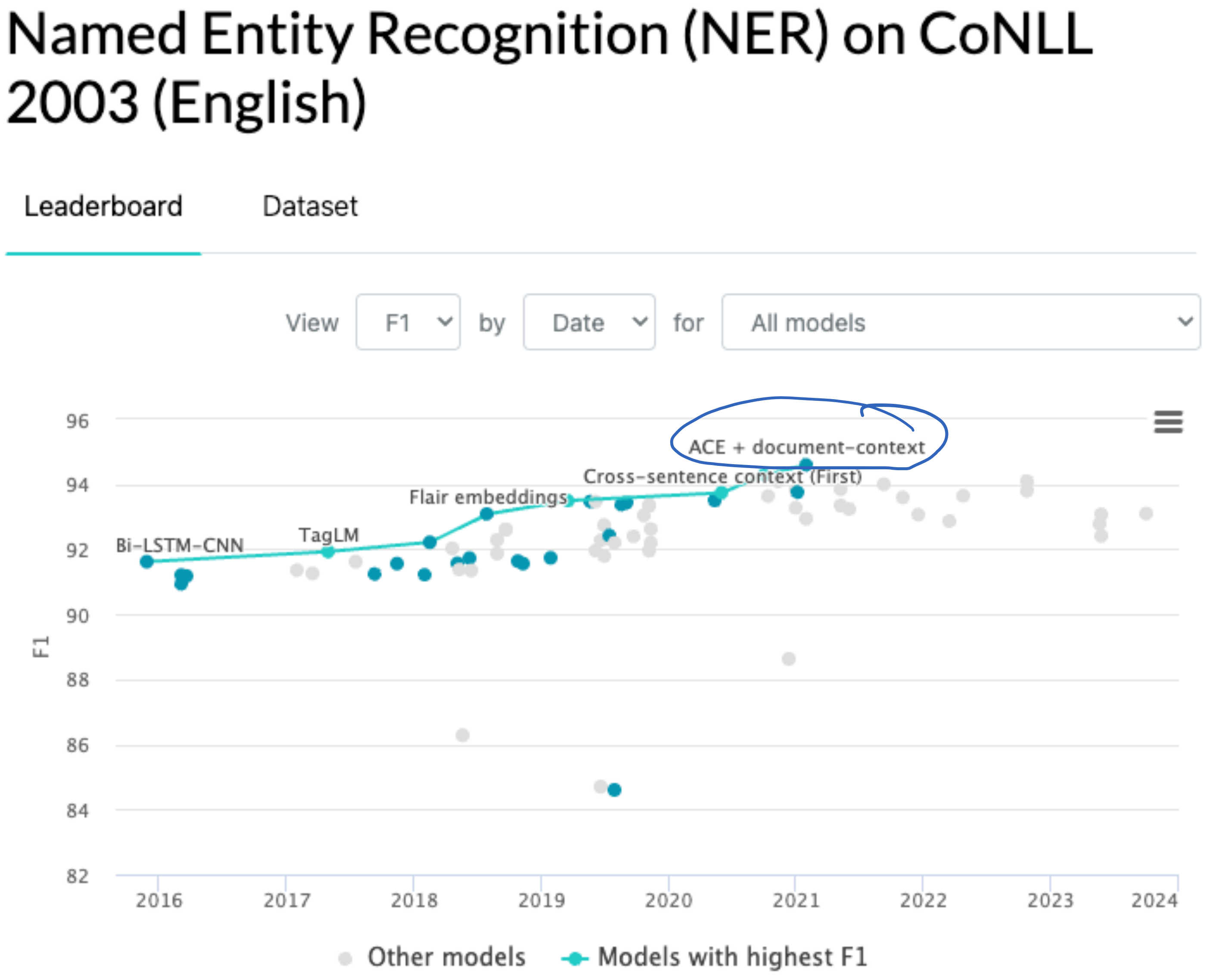


Figure from https://paperswithcode.com/sota/named-entity-recognition-ner-on-conll-2003

BACKGROUND: HUMAN LANGUAGE TECHNOLOGIES

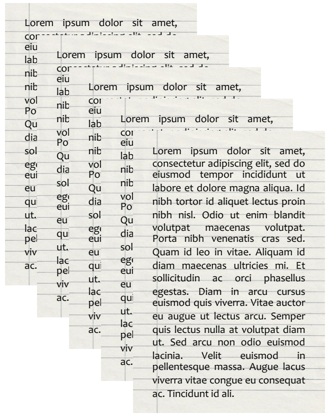
Human Language Technologies

Speech Recognition

Machine Translation

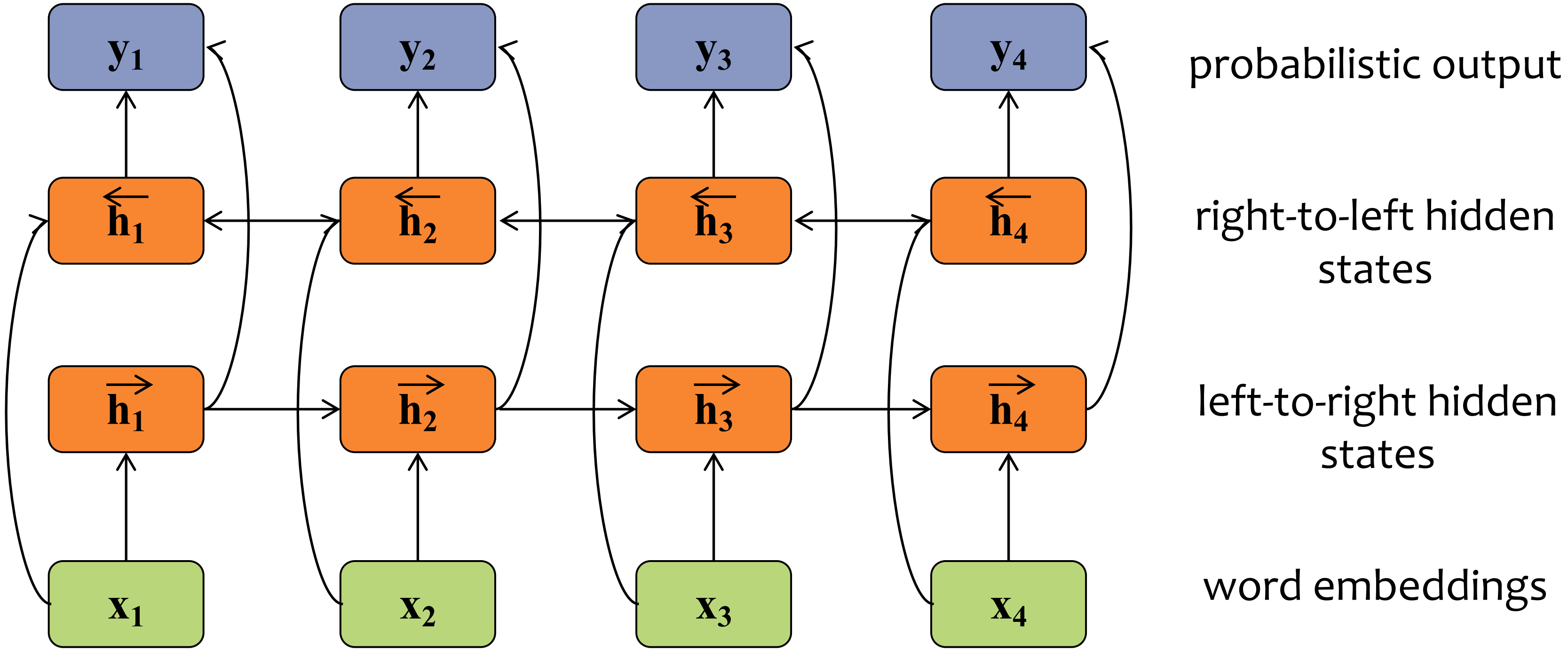
기계 번역은 특히 영어와 한국어와 같은 언어 쌍의 경우 매우 어렵습니다.

Summarization



Bidirectional RNN

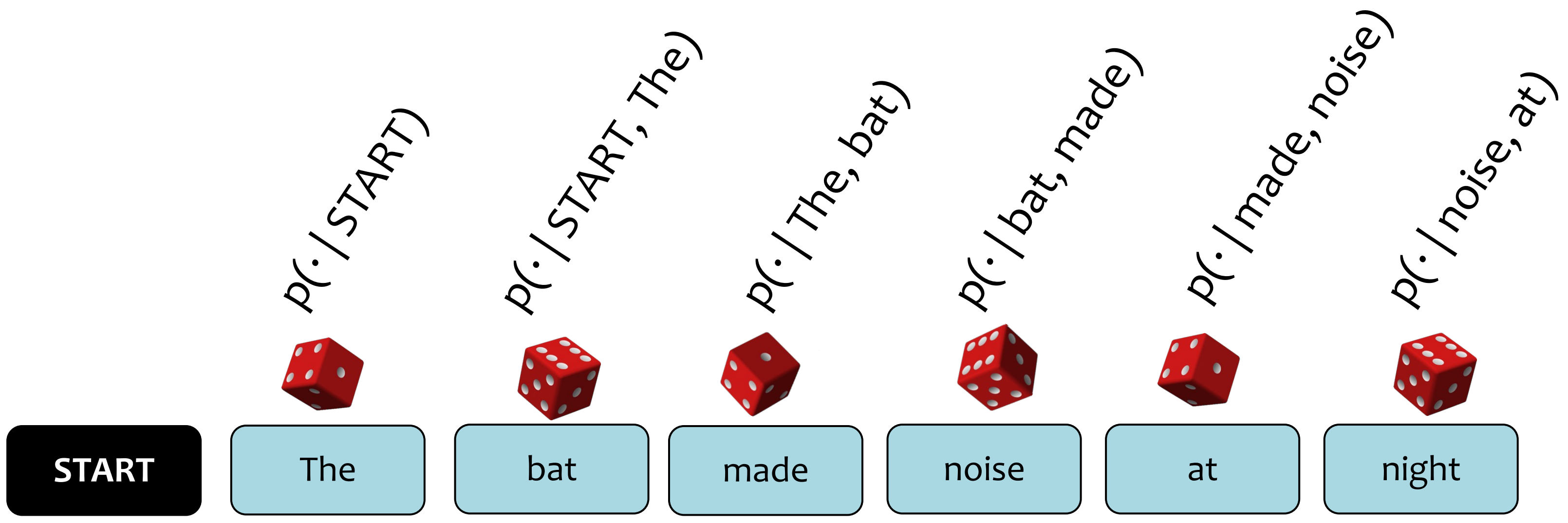
RNNs are a now commonplace backbone in deep learning approaches to natural language processing



BACKGROUND:
N-GRAM LANGUAGE MODELS

n-Gram Language Model

- *Goal*: Generate realistic looking sentences in a human language
- *Key Idea*: condition on the last n-1 words to sample the nth word



n-Gram Language Model

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=2)

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

$p(w_1, w_2, w_3, \ldots , w_6) =$

The	$p(w_1)$
The bat	$p(w_2 \mid w_1)$
bat made	$p(w_3 \mid w_2)$
made noise	$p(w_4 \mid w_3)$
noise at	$p(w_5 \mid w_4)$
at night	$p(w_6 \mid w_5)$

n-Gram Language Model

<u>Question</u>: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=3)

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

$p(w_1, w_2, w_3, \ldots, w_6) =$

The						$p(w_1)$
The	bat					$p(w_2 \mid w_1)$
The	bat	made				$p(w_3 \mid w_2, w_1)$
	bat	made	noise			$p(w_4 \mid w_3, w_2)$
		made	noise	at		$p(w_5 \mid w_4, w_3)$
			noise	at	night	$p(w_6 \mid w_5, w_4)$

n-Gram Language Model

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=3) $$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$$

$p(w_1, w_2, w_3, \ldots, w_6) =$

The $p(w_1)$

The $p(w_2 \mid w_1)$

The

Note: This is called a **model** because we made some **assumptions** about how many previous words to condition on (i.e. only n-1 words)

Learning an n-Gram Model

Question: How do we **learn** the probabilities for the n-Gram Model?

$p(w_t \mid w_{t-2} = \text{The}, w_{t-1} = \text{bat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
ate	0.015
…	
flies	0.046
…	
zebra	0.000

$p(w_t \mid w_{t-2} = \text{made}, w_{t-1} = \text{noise})$

w_t	$p(\cdot \mid \cdot, \cdot)$
at	0.020
…	
pollution	0.030
…	
zebra	0.000

$p(w_t \mid w_{t-2} = \text{cows}, w_{t-1} = \text{eat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
corn	0.420
…	
grass	0.510
…	
zebra	0.000

Learning an n-Gram Model

Question: How do we **learn** the probabilities for the n-Gram Model?

Answer: From data! Just **count** n-gram frequencies

…the **cows eat** grass…
…our **cows eat hay** daily…
…factory-farm **cows eat** corn…
…on an organic farm, **cows eat hay** and…
…do your **cows eat** grass or corn?...
…what do **cows eat if** they have…
…**cows eat** corn when there is no…
…which **cows eat which** foods depends…
…if **cows eat** grass…
…when **cows eat** corn their stomachs…
…should we let **cows eat** corn?...

$p(w_t \mid w_{t-2} = \text{cows}, w_{t-1} = \text{eat})$

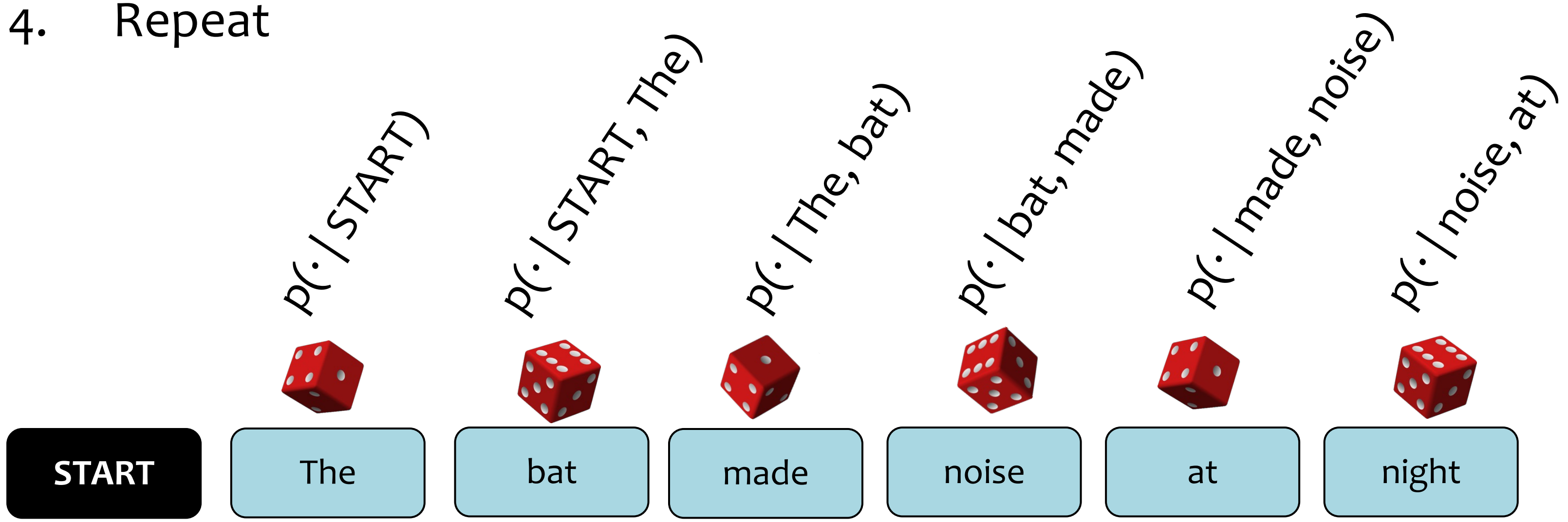
w_t	$p(\cdot \mid \cdot, \cdot)$
corn	4/11
grass	3/11
hay	2/11
if	1/11
which	1/11

Sampling from a Language Model

<u>*Question*</u>: How do we sample from a Language Model?

<u>*Answer*</u>:

1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat



Sampling from a Language Model

Question: How do we sample from a Language Model?
Answer:

1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat

Training Data (Shakespeaere)	**5-Gram Model**
I tell you, friends, most charitable care ave the patricians of you. For your wants, Your suffering in this dearth, you may as well Strike at the heaven with your staves as lift them Against the Roman state, whose course will on The way it takes, cracking ten thousand curbs Of more strong link asunder than can ever Appear in your impediment. For the dearth, The gods, not the patricians, make it, and Your knees to them, not arms, must help.	`Approacheth, denay. dungy Thither! Julius think: grant,--O Yead linens, sheep's Ancient, Agreed: Petrarch plaguy Resolved pear! observingly honourest adulteries wherever scabbard guess; affirmation--his monsieur; died. jealousy, chequins me. Daphne building. weakness: sun-rise, cannot stays carry't, unpurposed. prophet-like drink; back-return 'gainst surmise Bridget ships? wane; interim? She's striving wet;`

RECURRENT NEURAL NETWORK (RNN) LANGUAGE MODELS

Recurrent Neural Networks (RNNs)

inputs: $\mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$

hidden units: $\mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$

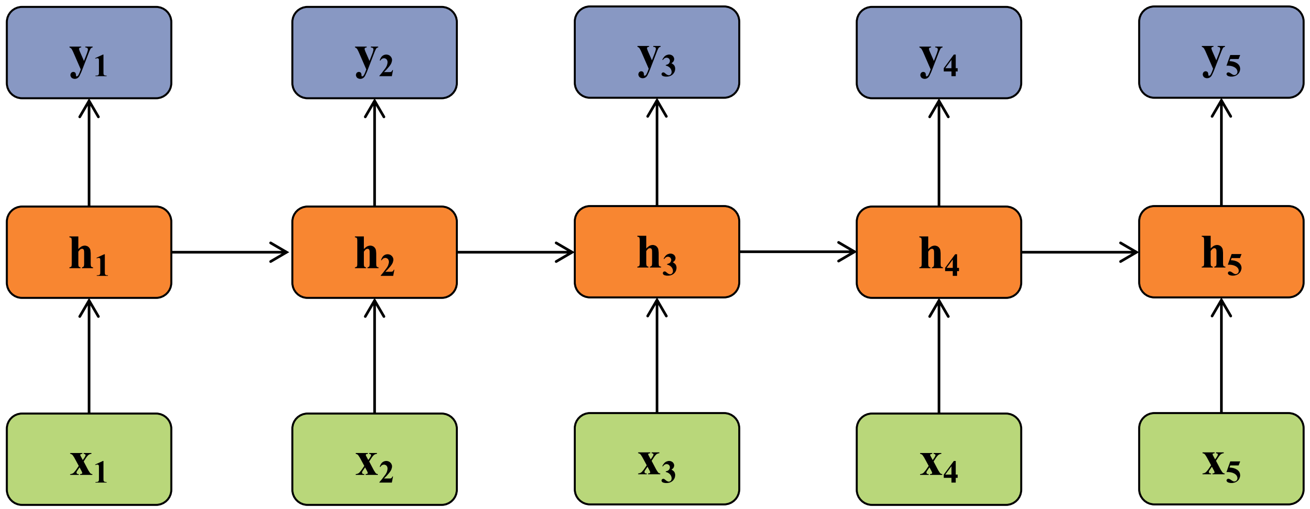
outputs: $\mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$

nonlinearity: $\mathcal{H}$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$y_t = W_{hy}h_t + b_y$$



Recall…

The Chain Rule of Probability

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

Chain rule of probability: $$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

$p(w_1, w_2, w_3, \ldots, w_6) =$

The

$p(w_1)$

The

The

The

The

The

Note: This is called the chain **rule** because it is **always** true for every probability distribution

$p(w_6 \mid w_5, w_4, w_3, w_2, w_1)$

RNN Language Model

RNN Language Model: $p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid f_{\boldsymbol{\theta}}(w_{t-1}, \ldots, w_1))$

$p(w_1, w_2, w_3, \ldots, w_6) =$

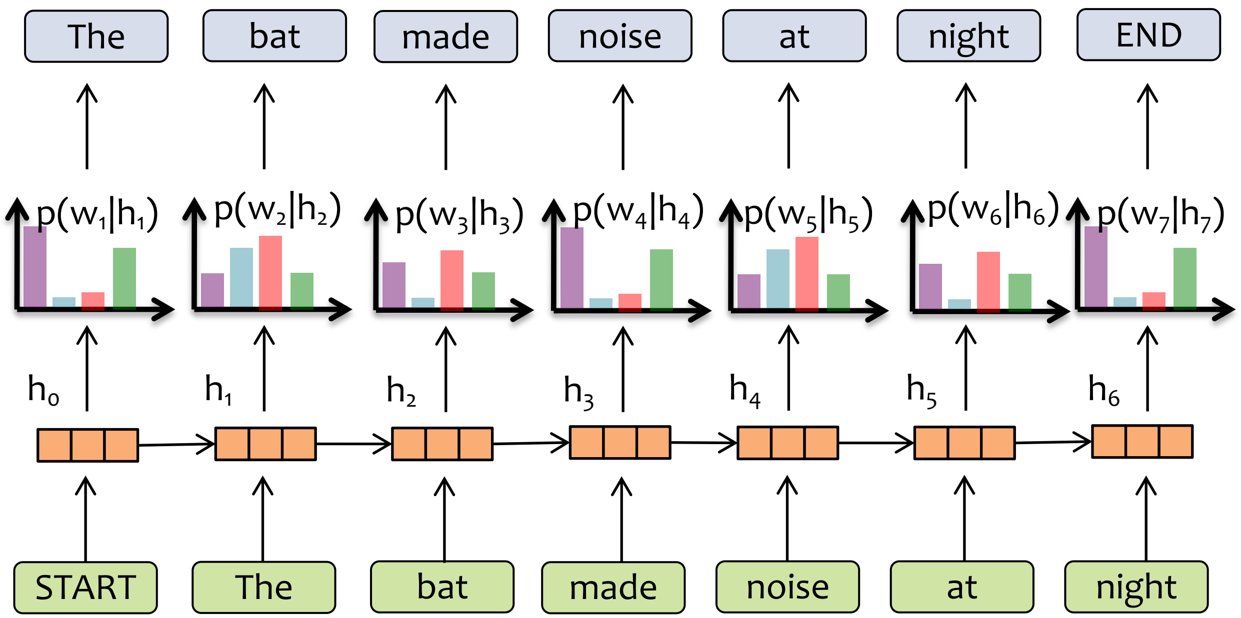
The						$p(w_1)$
The	bat					$p(w_2 \mid f_\theta(w_1))$
The	bat	made				$p(w_3 \mid f_\theta(w_2, w_1))$
The	bat	made	noise			$p(w_4 \mid f_\theta(w_3, w_2, w_1))$
The	bat	made	noise	at		$p(w_5 \mid f_\theta(w_4, w_3, w_2, w_1))$
The	bat	made	noise	at	night	$p(w_6 \mid f_\theta(w_5, w_4, w_3, w_2, w_1))$

Key Idea:

(1) convert all previous words to a **fixed length vector**

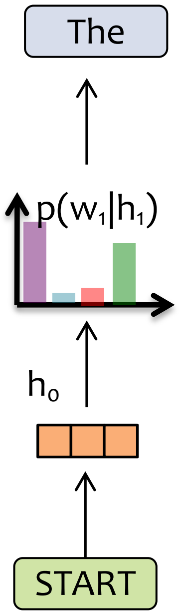
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector

RNN Language Model



<u>Key Idea</u>:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

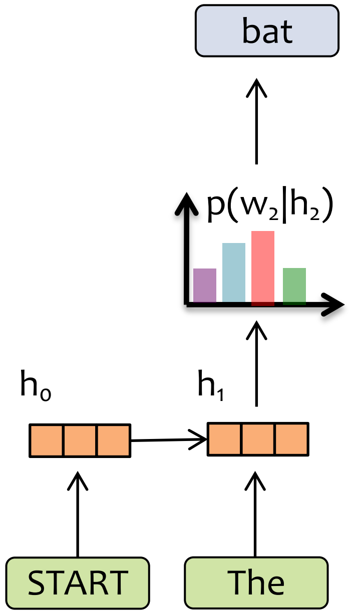


<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

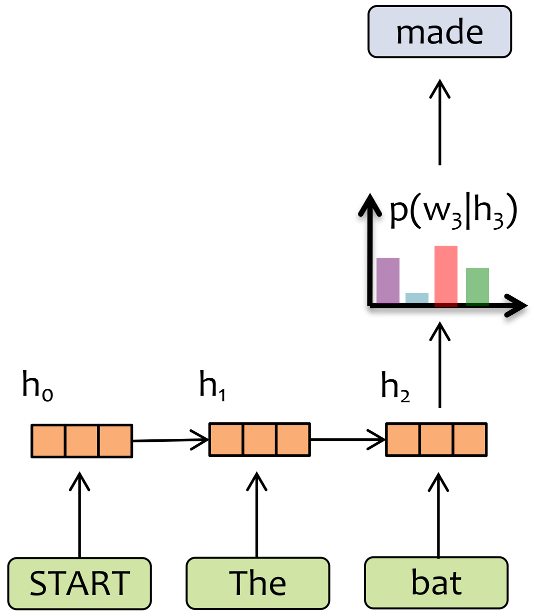
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

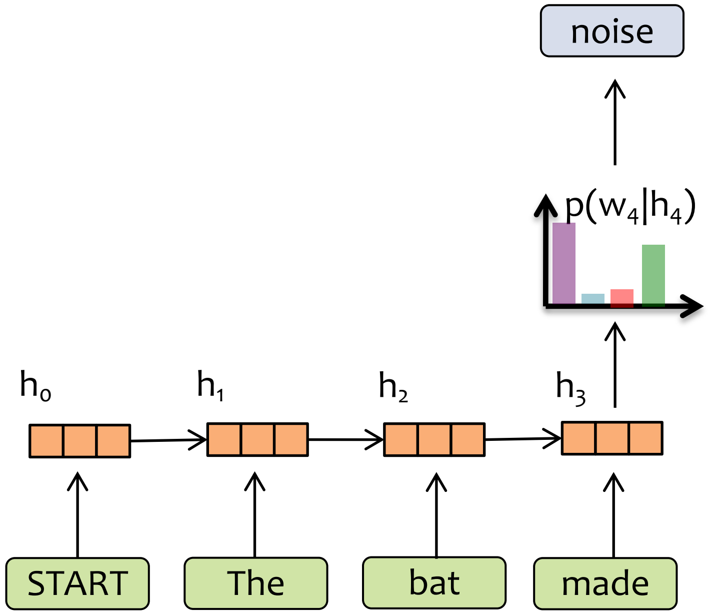


<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

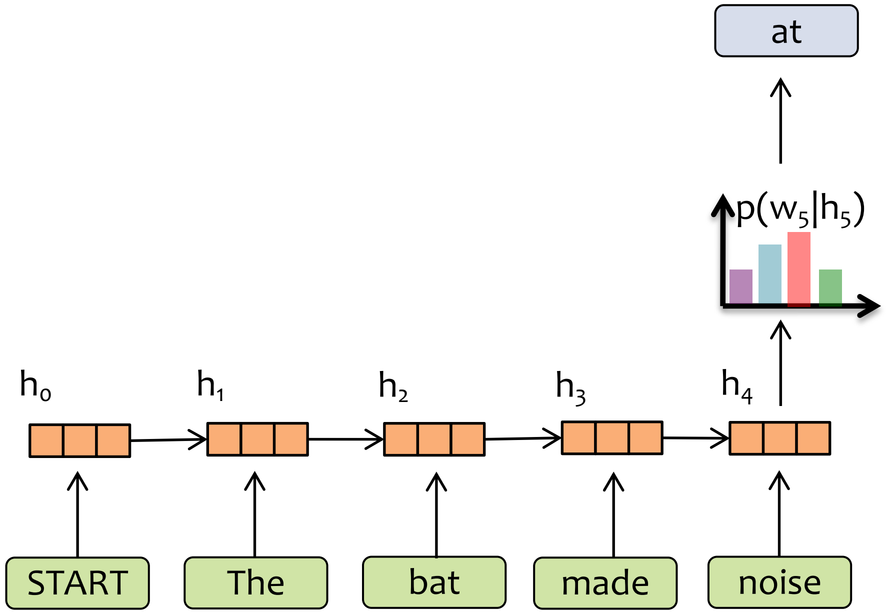
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

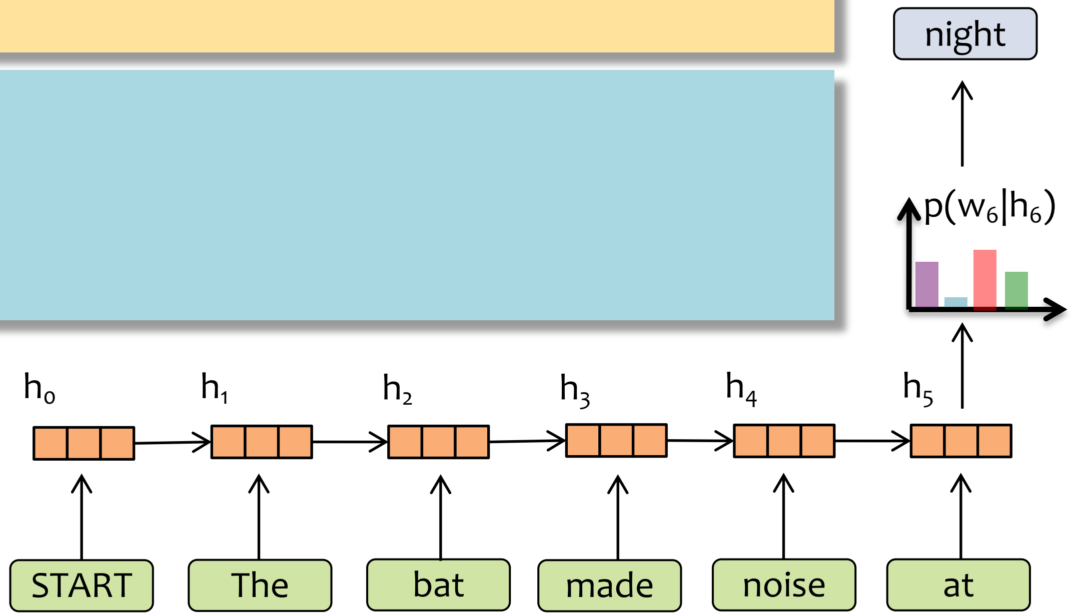


Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

Question: How can we create a distribution $p(w_t|h_t)$ from h_t?

Answer:

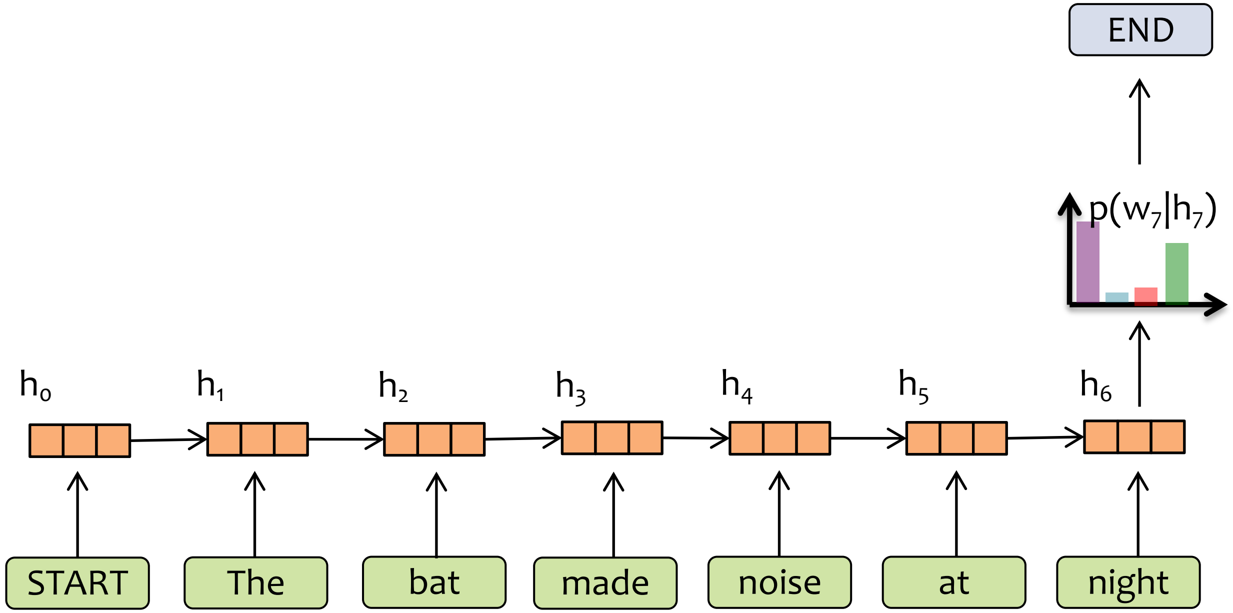


Key Idea:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

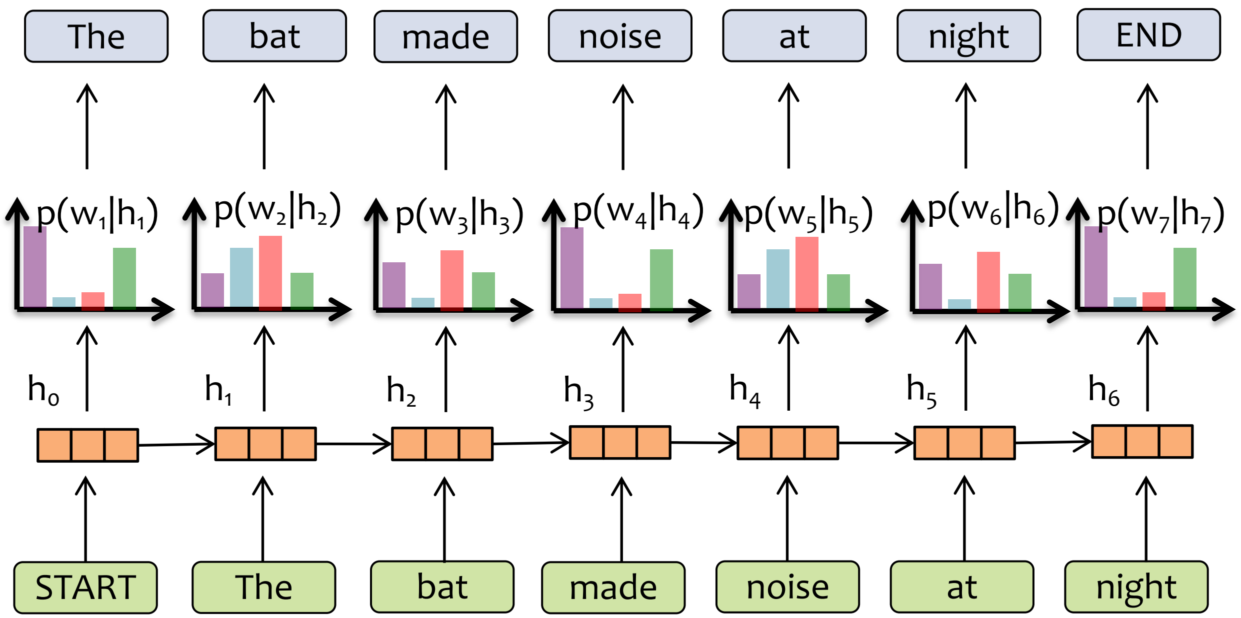


<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

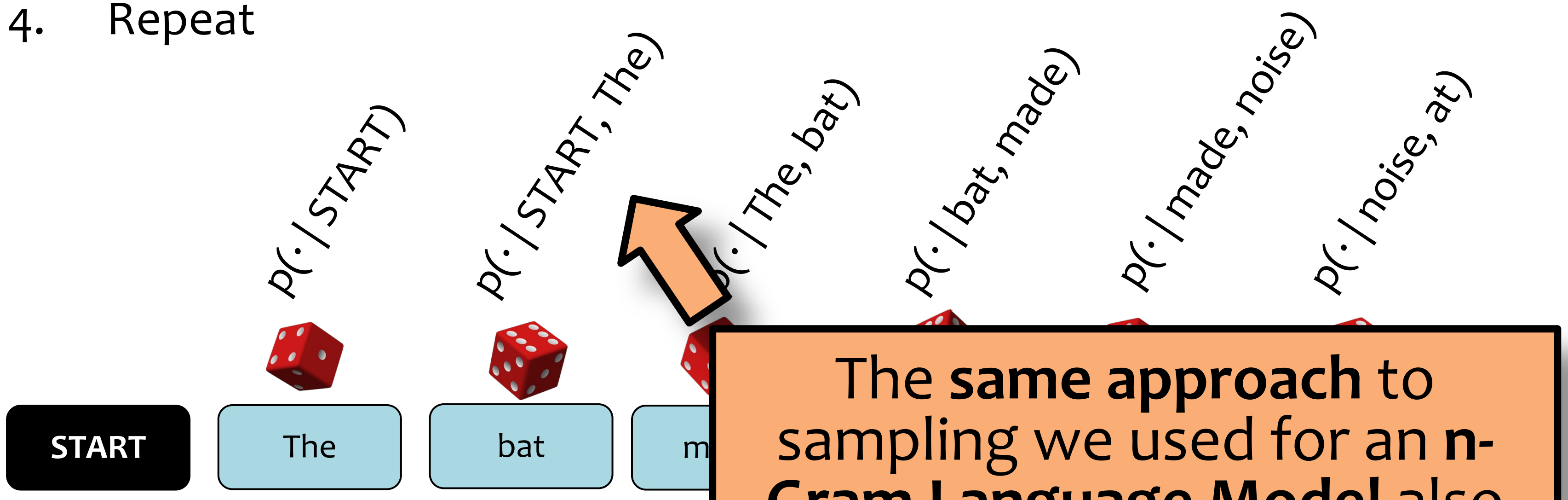
RNN Language Model



$$p(w_1, w_2, w_3, \ldots , w_T) = p(w_1 \mid h_1)\, p(w_2 \mid h_2) \ldots\ p(w_2 \mid h_T)$$

Sampling from a Language Model

Question: How do we sample from a Language Model?
Answer:
1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat



The **same approach** to sampling we used for an **n-Gram Language Model** also works here for an **RNN Language Model**

Sampling from an RNN-LM

??

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy m
there My power to give thee but so much
service in the noble bondman here, Would
her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

??

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is
ender; and, for your love, I would be
, as I must, for my own honour, if he
ore, out of my love to you, I came hither
withal, that either you might stay him
from his intend
r brook such disgrace well as he
shall run into, in tha
is a thing of his own search and
altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Which is the real Shakespeare?!

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

Sampling from an RNN-LM

Shakespeare's As You Like It

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

RNN-LM Sample

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

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

Sampling from an RNN-LM

RNN-LM Sample

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

Shakespeare's As You Like It

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

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

Sampling from an RNN-LM

??

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy m
there My power to give thee but so much
service in the noble bondman here, Would
her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

??

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is
ender; and, for your love, I would be
, as I must, for my own honour, if he
ore, out of my love to you, I came hither
to acquaint you withal, that either you might stay him from his intendr
r brook such disgrace well as he shall run into, in tha
is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Which is the real Shakespeare?!

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

SEQUENCE TO SEQUENCE MODELS

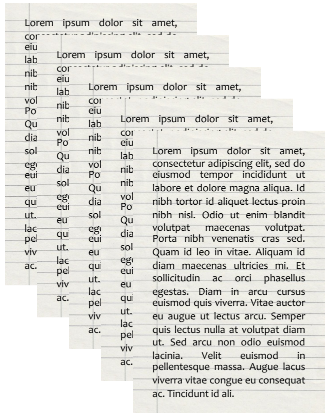
Sequence to Sequence Model

Speech Recognition

Machine Translation

기계 번역은 특히 영어와 한국어와 같은 언어 쌍의 경우 매우 어렵습니다.

Summarization



Sequence to Sequence Model

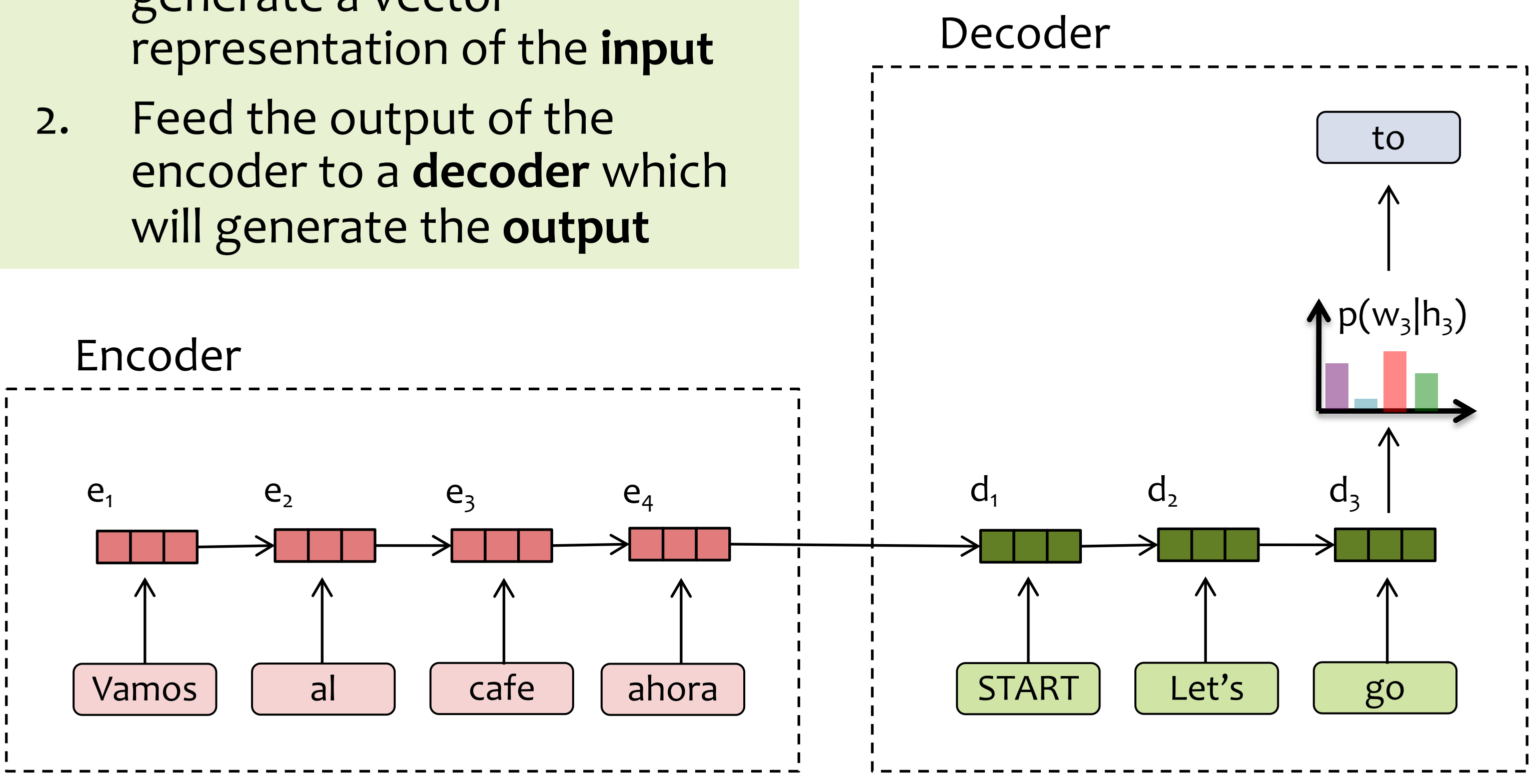
Now suppose you want generate a sequence conditioned on another input

Key Idea:

1. Use an **encoder** model to generate a vector representation of the **input**
2. Feed the output of the encoder to a **decoder** which will generate the **output**

Applications:

- translation: Spanish → English
- summarization: article → summary
- speech recognition: speech signal → transcription



Deep Learning Objectives

You should be able to…

- Implement the common layers found in Convolutional Neural Networks (CNNs) such as linear layers, convolution layers, max-pooling layers, and rectified linear units (ReLU)
- Explain how the shared parameters of a convolutional layer could learn to detect spatial patterns in an image
- Describe the backpropagation algorithm for a CNN
- Identify the parameter sharing used in a basic recurrent neural network, e.g. an Elman network
- Apply a recurrent neural network to model sequence data
- Differentiate between an RNN and an RNN-LM