Machine Learning Department
School of Computer Science
Carnegie Mellon University

RNN LMs
+
Transformer LMs

Matt Gormley, Henry Chai
Lecture 18
Oct. 30, 2024

Reminders

- Homework 6: Learning Theory & Generative Models
  - Out: Sun, Oct 27
  - Due: Sat, Nov 2, 11:59pm
- Exam 2: Thu, Nov 7, 6:45 pm - 8:45 pm

EXAM 2 LOGISTICS

Exam 2

- **Time / Location**
  - **Time: Thu, Nov. 7, 6:45pm – 8:45pm**
  - **Location & Seats:** You have all been split across multiple rooms. Everyone has an assigned seat in one of these room. Please watch Piazza carefully for announcements.
- **Logistics**
  - Covered material: Lecture 8 – Lecture 16
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
  - No electronic devices
  - You are allowed to **bring** one 8½ x 11 sheet of notes (front and back, ~~handwritten with pen/pencil or tablet~~)

Topics for Exam 1

- Foundations
  - Probability, Linear Algebra, Geometry, Calculus
  - Optimization
- Important Concepts
  - Overfitting
  - Experimental Design
- Classification
  - Decision Tree
  - KNN
  - Perceptron
- Regression
  - Linear Regression

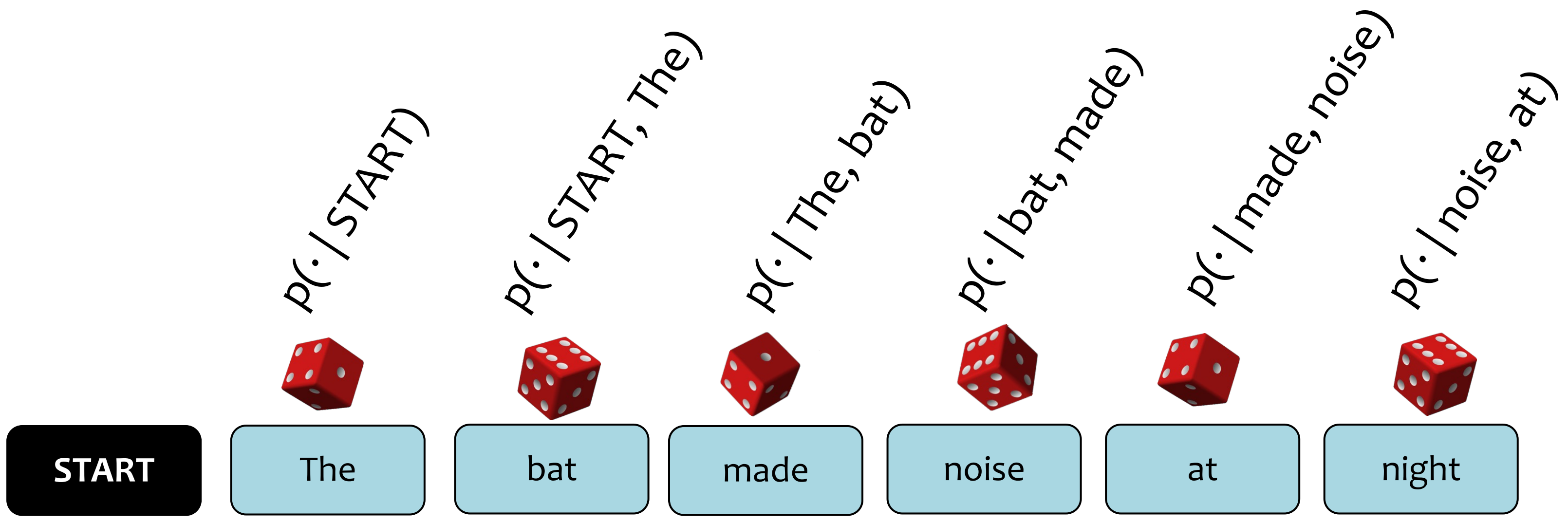
Topics for Exam 2

- Classification
  - Binary Logistic Regression
- Important Concepts
  - Stochastic Gradient Descent
  - Regularization
  - Feature Engineering
- Feature Learning
  - Neural Networks
  - Basic NN Architectures
  - Backpropagation
- Learning Theory
  - PAC Learning
  - MLE / MAP
- Societal Impacts of ML
- Regression
  - Linear Regression

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 n^{th} word



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=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)

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

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

The

$p(w_1)$

The

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

<u>Question</u>: 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

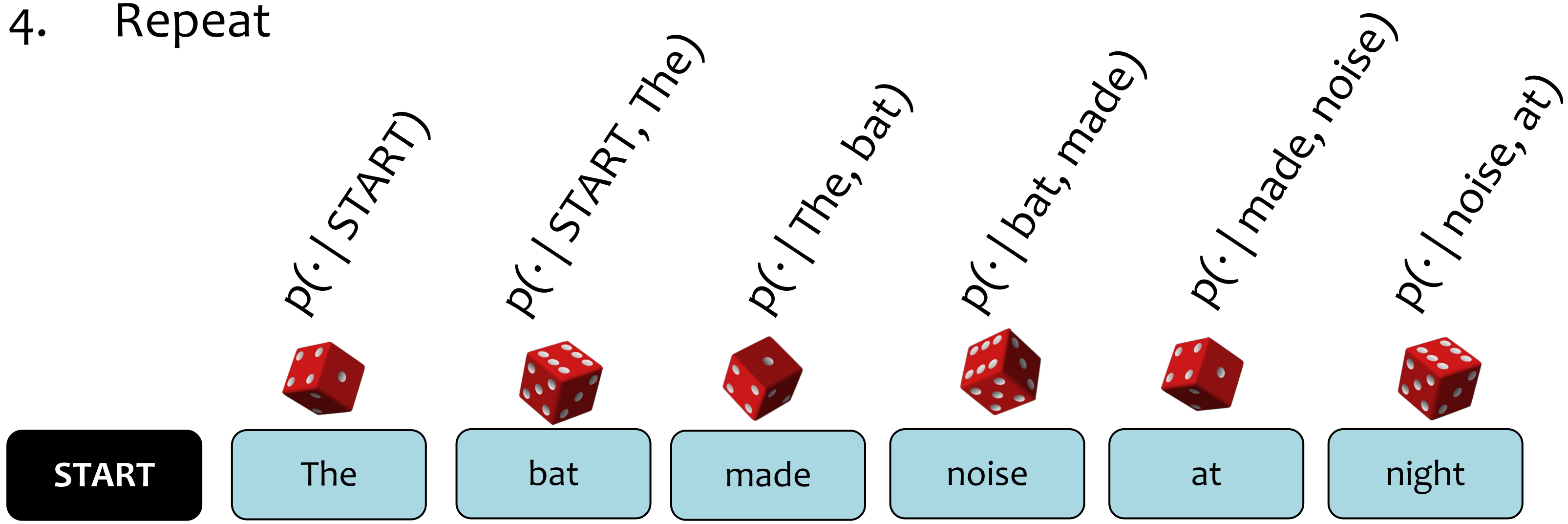
- …the **cows eat** grass…
- …our **cows eat hay** daily…
- …factory-farm **cows eat** corn…
- …on an organic farm, **cows eat hay** and…
- …do your **cows eat** grass or corn?...
- …what do **cows eat if** they have…
- …**cows eat** corn when there is no…
- …which **cows eat which** foods depends…
-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

Elman Network

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:

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

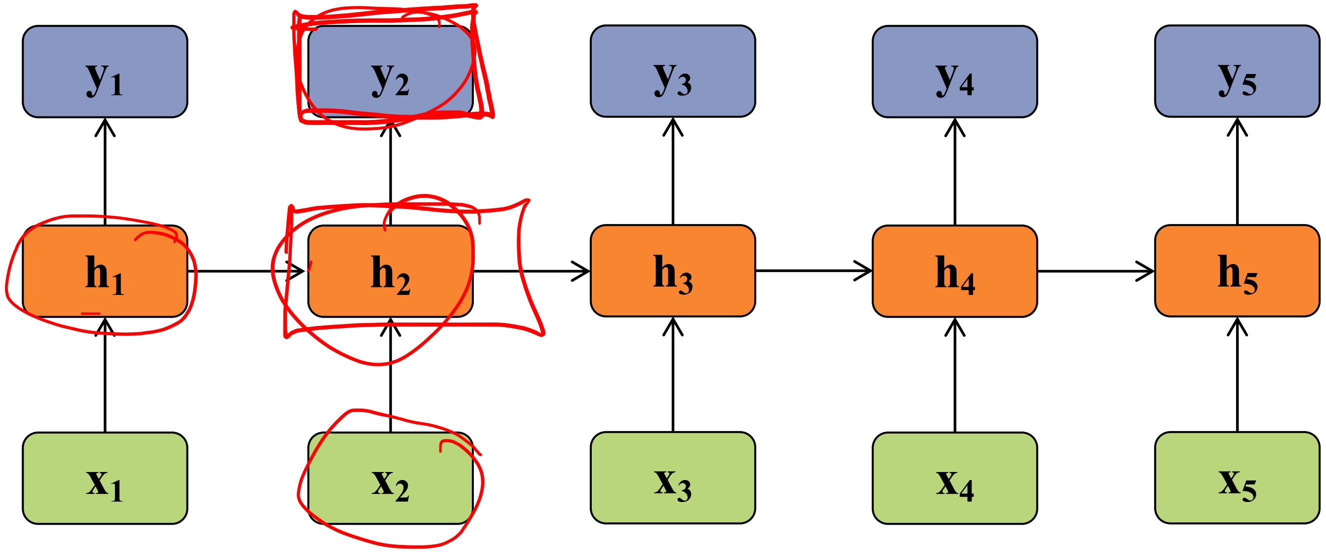
$$y_t = W_{hy}h_t + b_y$$

Params

W_{xh} b_h

W_{hh}

W_{hy} b_y



The Chain Rule of Probability

Recall...

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

The

The

The

The

The

$p(w_1)$

$p(w_2 \mid w_1)$

$\ldots w_1)$

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

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

$=$
$p(w_6)$
$p(w_5 \mid w_6)$
$p(w_4 \mid w_{5,6})$
$\vdots$
$p(w_1 \mid w_2, \ldots, w_6)$

RNN Language Model

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

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

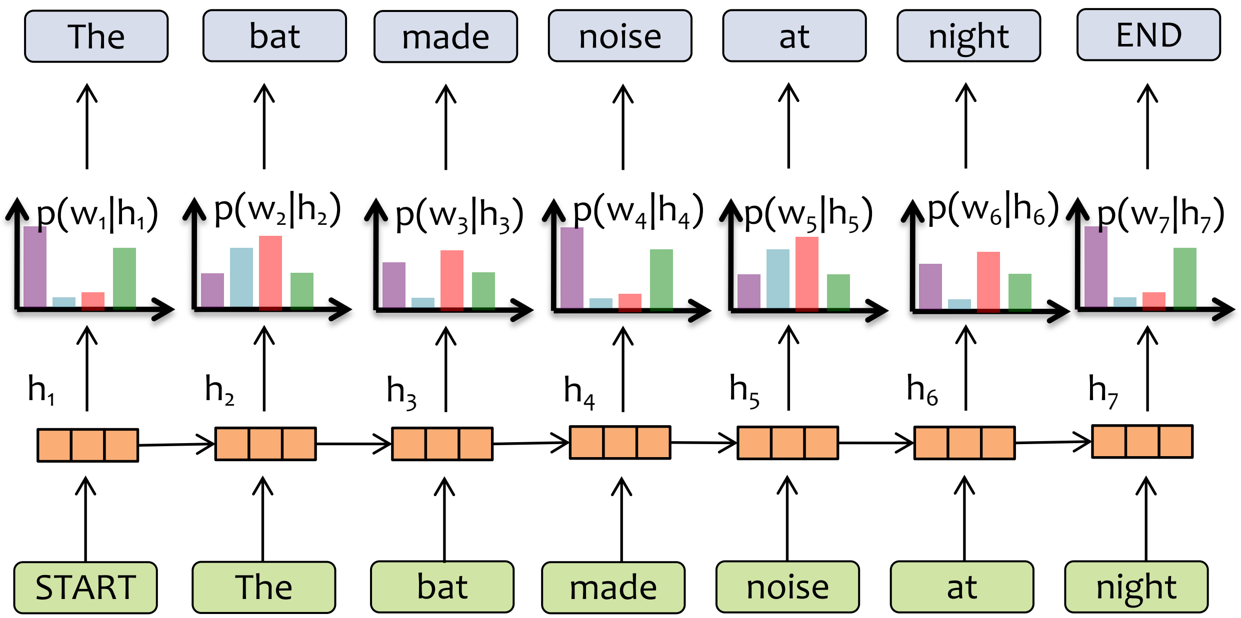
The						$p(w_1)$
The	bat					$p(w_2 \mid f_\theta(w_1))$
The	bat	made				$p(w_3 \mid f_\theta(w_2, w_1))$
The	bat	made	noise			$p(w_4 \mid f_\theta(w_3, w_2, w_1))$
The	bat	made	noise	at		$p(w_5 \mid f_\theta(w_4, w_3, w_2, w_1))$
The	bat	made	noise	at	night	$p(w_6 \mid f_\theta(w_5, w_4, w_3, w_2,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

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

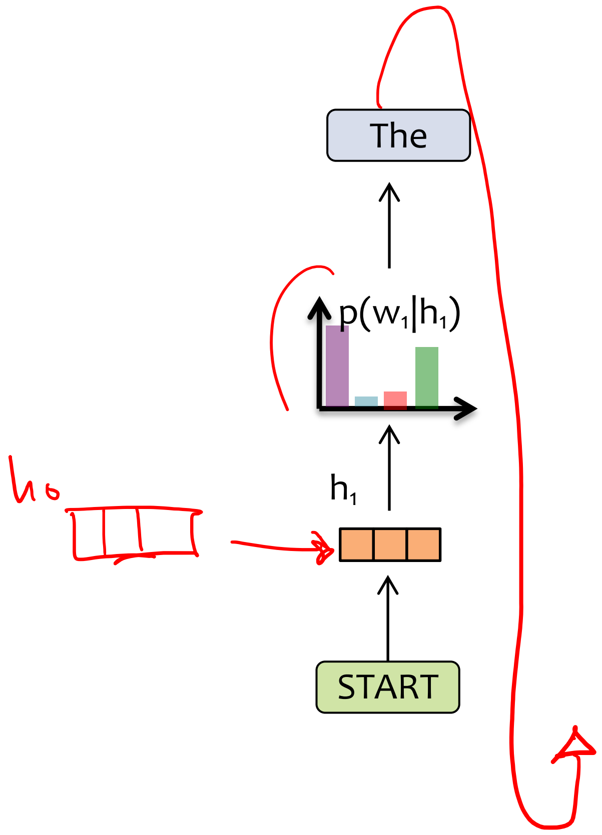


Key Idea:

(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

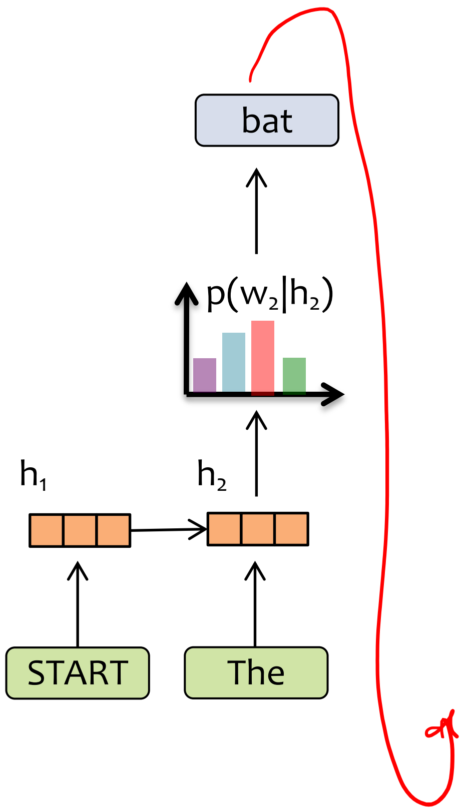


Key Idea:

(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

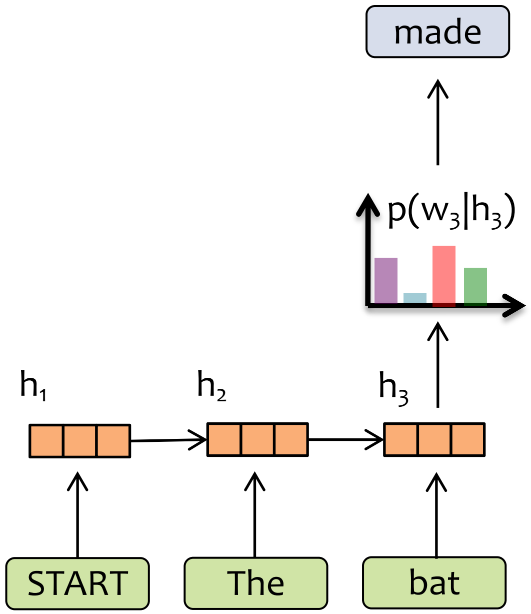


Key Idea:

(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

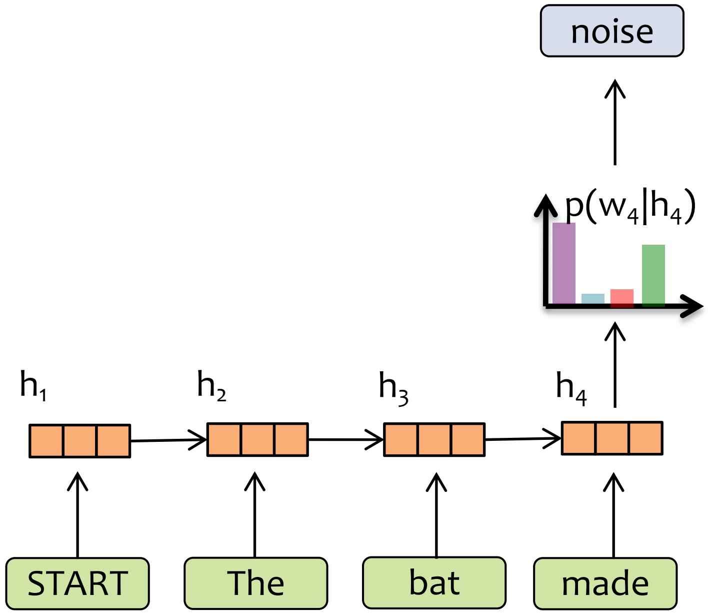


Key Idea:

(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

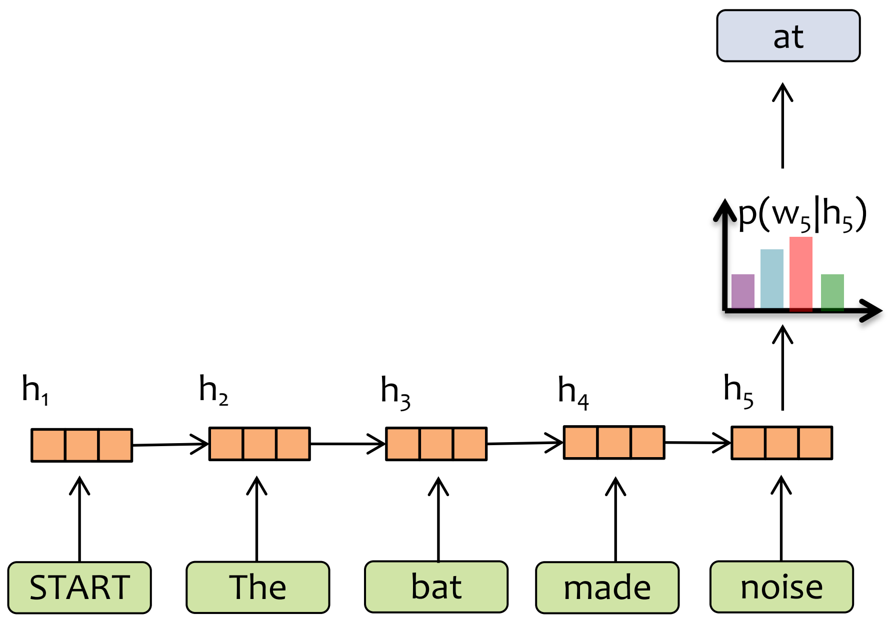


__Key Idea__:

(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 $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

Poll Q1

RNN Language Model

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

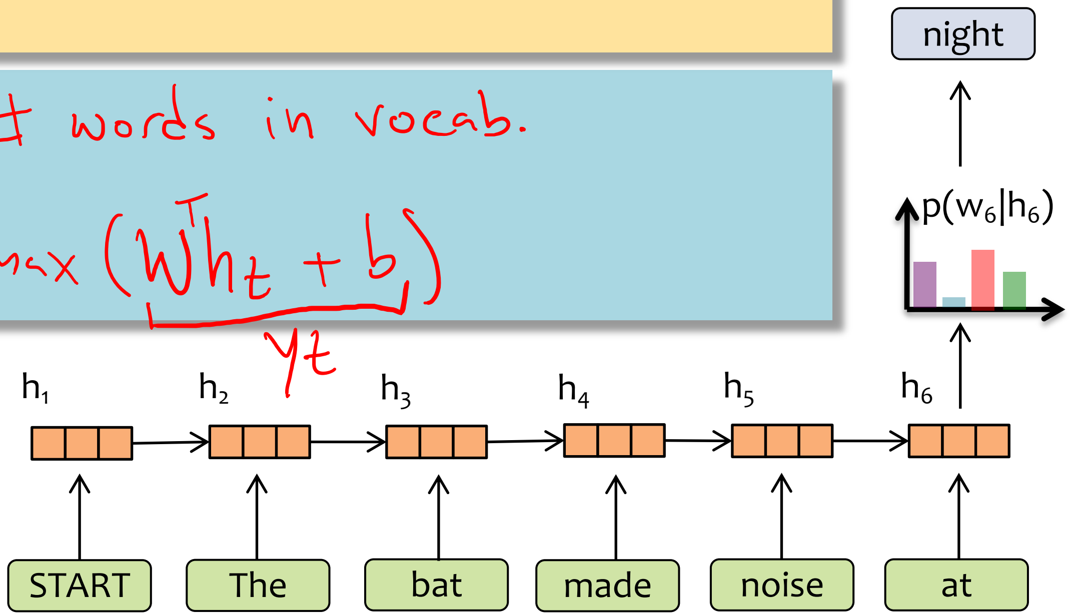
Answer: V = # words in vocab.

$$p(w_t|h_t) = \text{softmax}(\underbrace{W^T h_t + b}_{y_t})$$

$h_t \in \mathbb{R}^J$

$W \in \mathbb{R}^{J \times V}$

$b \in \mathbb{R}^V$

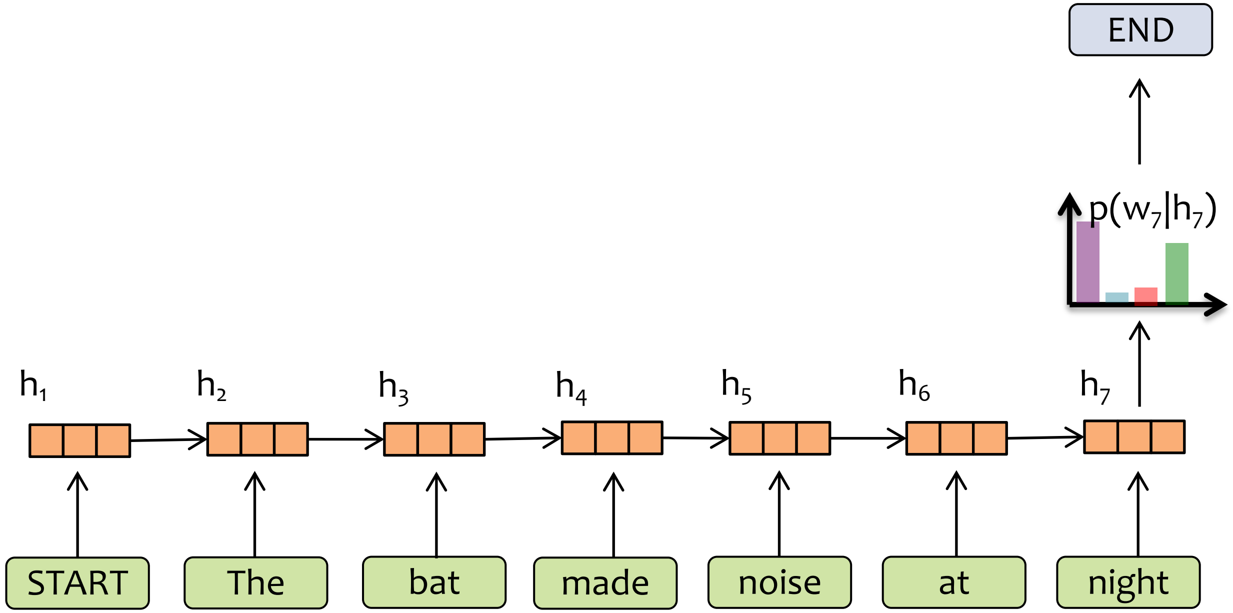


Key Idea:

(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

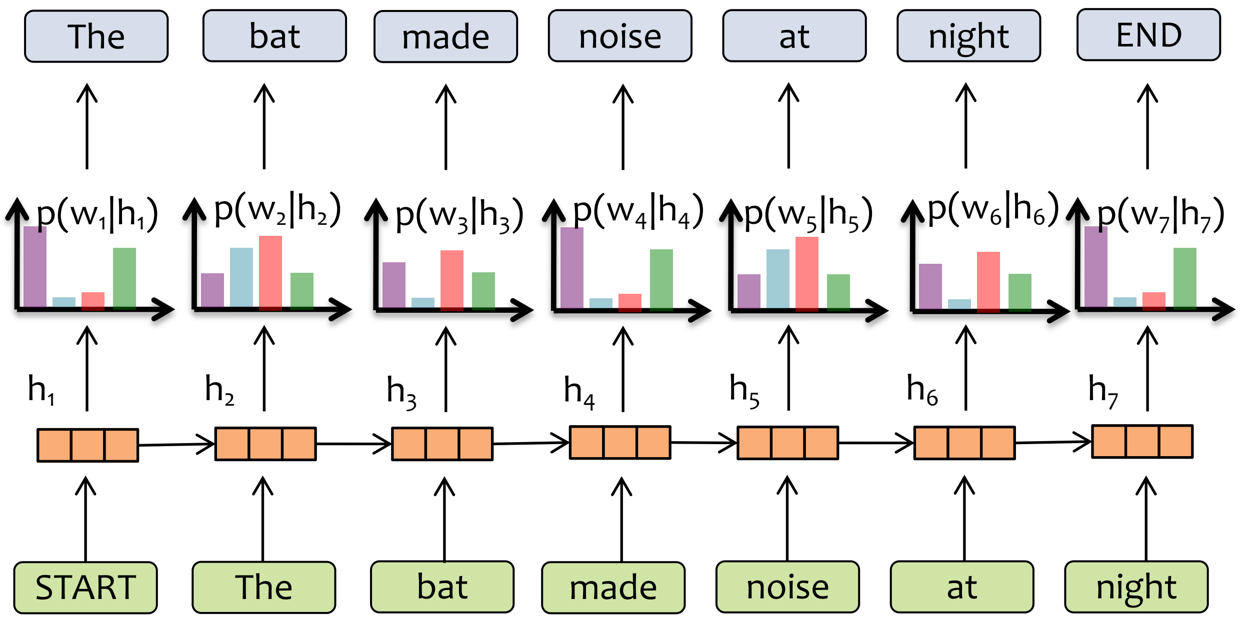


<u>*Key Idea*</u>:

(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

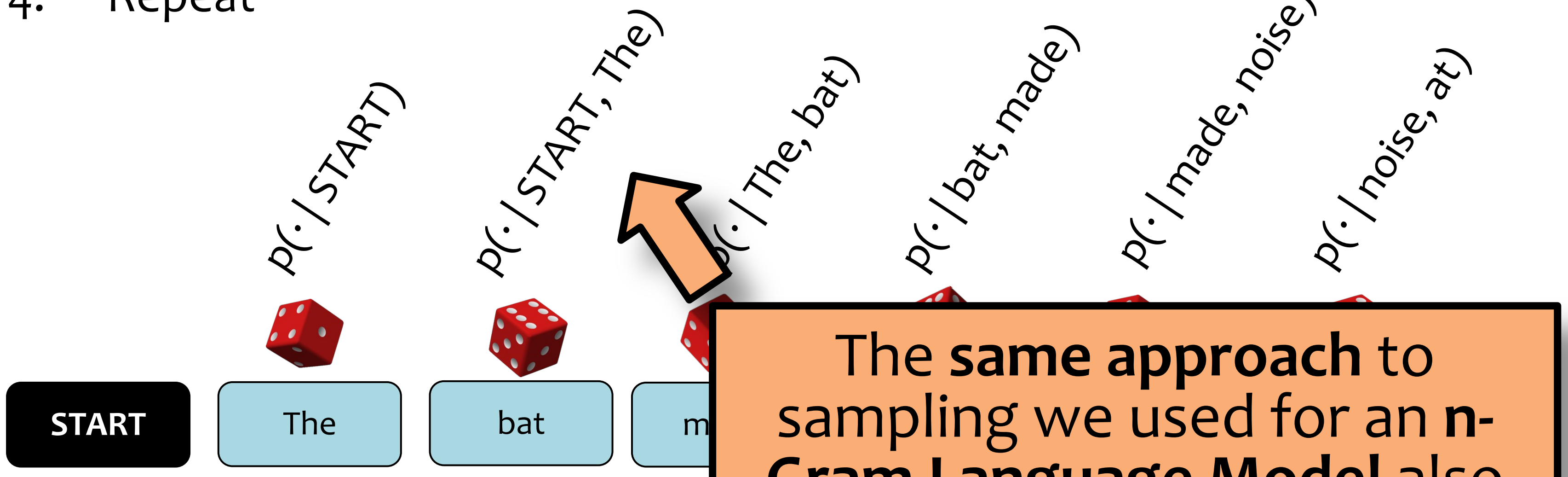


$$p(w_1, w_2, w_3, \ldots, w_T) = p(w_1 \mid h_1)\, \underbrace{p(w_2 \mid h_2)} \ldots\, p(w_T \mid h_T)$$

$p(w_t | h_t)$

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 …re, out of my love to you, I came hither to acquaint you 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 s…t, 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 …re, 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?!

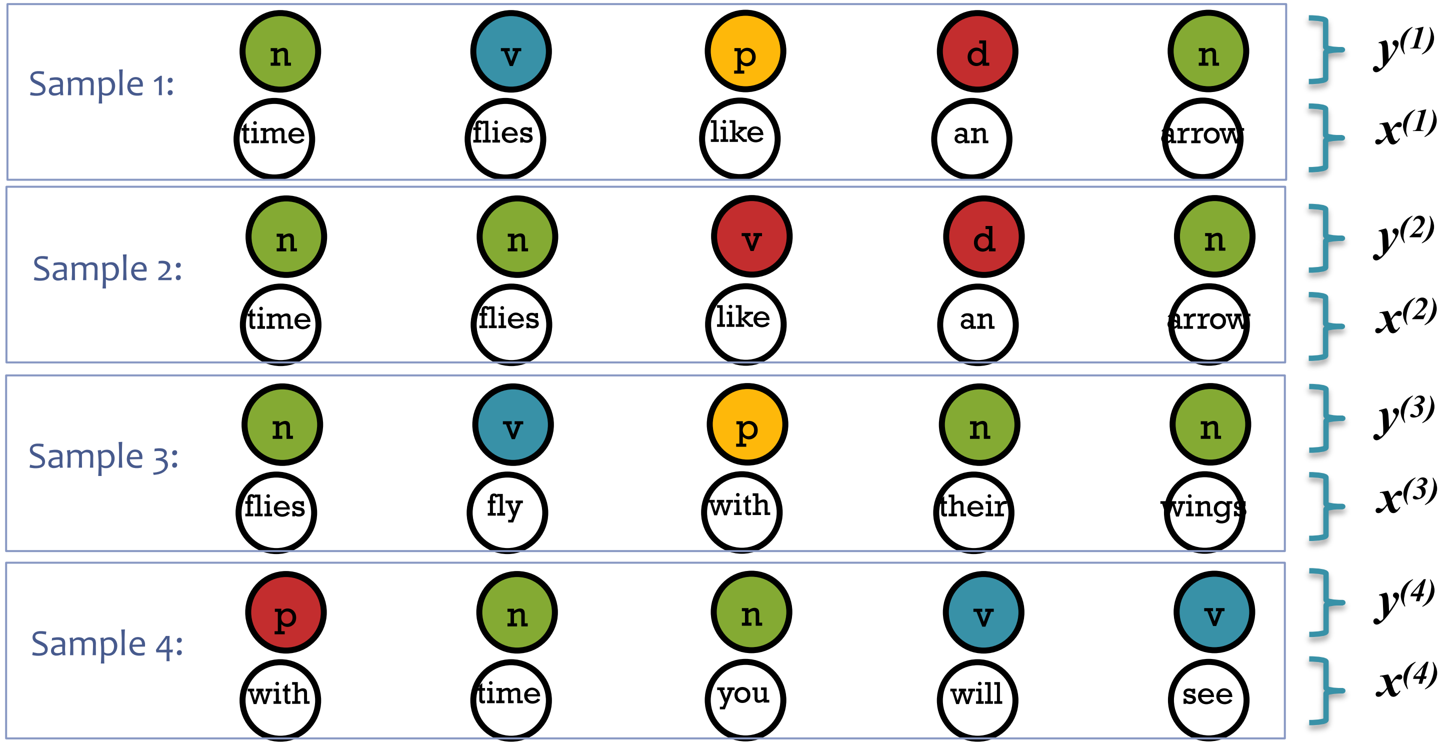
Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

LEARNING AN RNN

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

Recall...

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



SGD and Mini-batch SGD

Recall...

Algorithm 1 SGD

```
Initialize θ^(0)
s = 0
for t = 1, 2, ..., T do
    for i ∈ shuffle(1, ..., N) do
        Select the next training point (x_i, y_i)
        Compute the gradient g^(s) = ∇J_i(θ^(s−1))
        Update parameters θ^(s) = θ^(s−1) − η g^(s)
        Increment time step s = s + 1
    Evaluate average training loss J(θ) = (1/n) Σ_{i=1}^{n} J_i(θ)
return θ^(s)
```

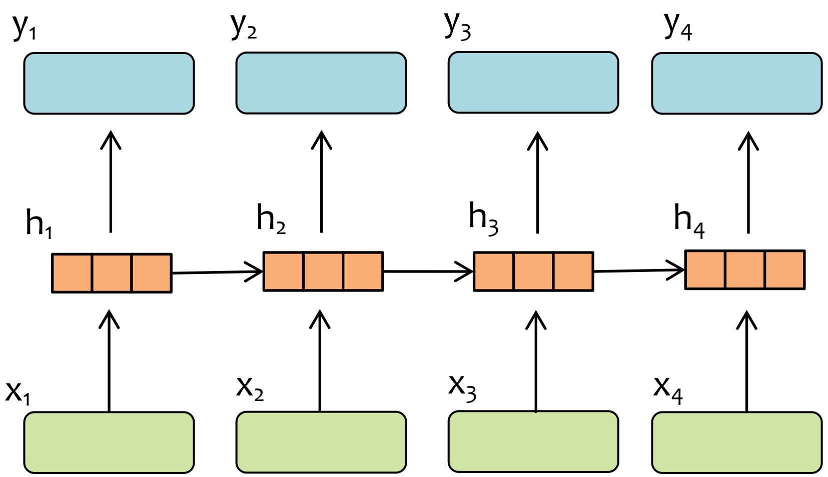
SGD and Mini-batch SGD

Recall...

Algorithm 1 Mini-Batch SGD

Initialize $\theta^{(0)}$
Divide examples $\{1, \ldots, N\}$ randomly into batches $\{I_1, \ldots, I_B\}$
where $\bigcup_{b=1}^{B} I_b = \{1, \ldots, N\}$ and $\bigcap_{b=1}^{B} I_b = \emptyset$
$s = 0$
for $t = 1, 2, \ldots, T$ **do**
    for $b = 1, 2, \ldots, B$ **do**
        Select the next batch I_b, where $m = |I_b|$
        Compute the gradient $g^{(s)} = \frac{1}{m} \sum_{i \in I_b} \nabla J_i(\theta^{(s)})$
        Update parameters $\theta^{(s)} = \theta^{(s-1)} - \eta g^{(s)}$
        Increment time step $s = s + 1$
    Evaluate average training loss $J(\theta) = \frac{1}{n} \sum_{i=1}^{n} J_i(\theta)$
return $\theta^{(s)}$

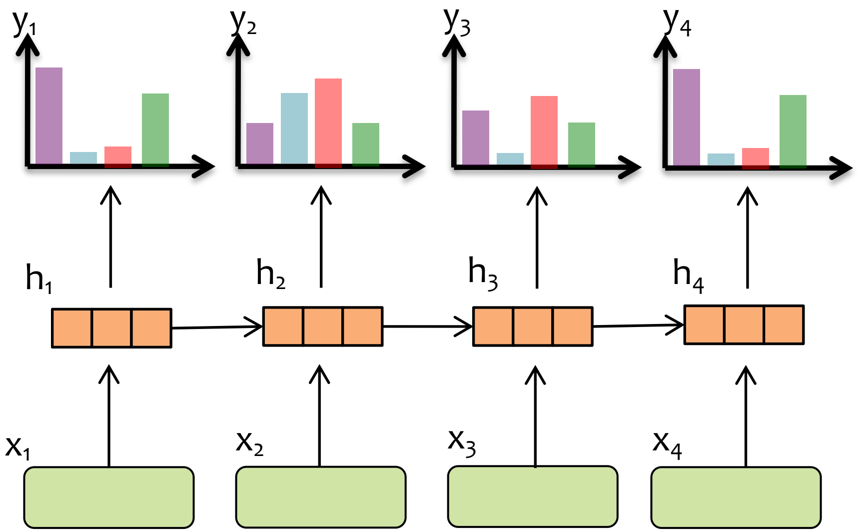
RNN



Algorithm 1 Elman RNN

procedure FORWARD($x_{1:T}, W_{ah}, W_{ax}, b_a, W_{yh}, b_y$)
    Initialize the hidden state h_0 to zeros
    for t in 1 to T **do**
        Receive input data at time step t: x_t
        Compute the hidden state update:
            $a_t = W_{ah} \cdot h_{t-1} + W_{ax} \cdot x_t + b_a$
            $h_t = \sigma(a_t)$
        Compute the output at time step t:
            $y_t = W_{yh} \cdot h_t + b_y$

RNN

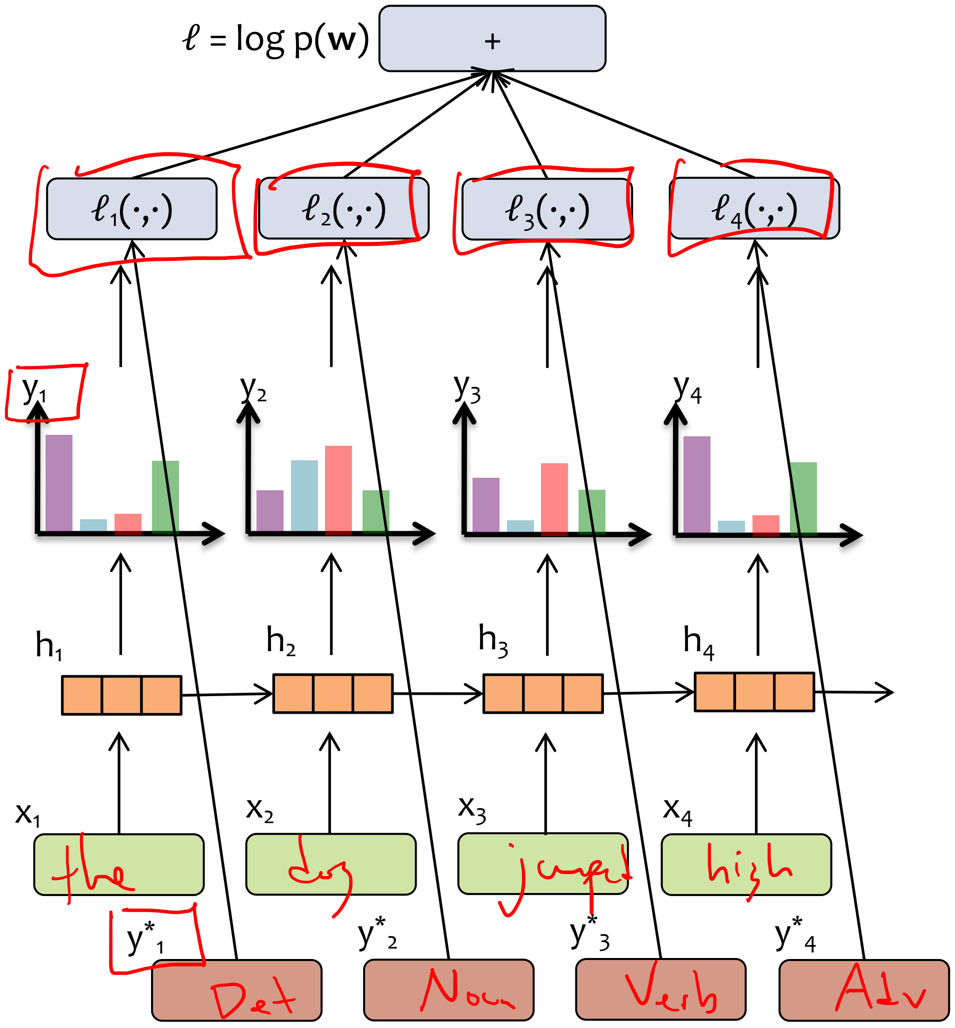


Algorithm 1 Elman RNN

---

procedure FORWARD($x_{1:T}, W_{ah}, W_{ax}, b_a, W_{yh}, b_y$)
    Initialize the hidden state h_0 to zeros
    for t in 1 to T **do**
        Receive input data at time step t: x_t
        Compute the hidden state update:
            $a_t = W_{ah} \cdot h_{t-1} + W_{ax} \cdot x_t + b_a$
            $h_t = \sigma(a_t)$
        Compute the output at time step t:
            $y_t = \text{softmax}(W_{yh} \cdot h_t + b_y)$

RNN + Loss



Algorithm 1 Elman RNN + Loss

procedure FORWARD($x_{1:T}, y^*_{1:T} W_{ah}, W_{ax}, b_a, W_{yh}, b_y$)

Initialize the hidden state h_0 to zeros

for t in 1 to T **do**

Receive input data at time step t: x_t

Compute the hidden state update:

$$a_t = \boxed{W_{ah}} \cdot h_{t-1} + \boxed{W_{ax}} \cdot x_t + \boxed{b_a}$$

$$h_t = \sigma(a_t)$$

Compute the output at time step t:

$$y_t = \text{softmax}(\boxed{W_{yh}} \cdot h_t + \boxed{b_y})$$

Compute the cross-entropy loss at time step t:

$$\ell_t = -\sum_{k=1}^{K} (y^*_t)_k \log((y_t)_k)$$

Compute the total loss:

$$\ell = \sum_{t=1}^{T} \ell_t$$

$$J^{(i)}(\theta) = \ell$$

LEARNING AN RNN-LM

Learning a Language 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

MLE for n-gram LM

- This counting method gives us the **maximum likelihood estimate** of the n-gram LM parameters
- We can derive it in the usual way:
  - **Write the likelihood** of the sentences under the n-gram LM
  - **Set the gradient to zero** and impose the constraint that the probabilities sum-to-one
  - **Solve** for the MLE

Learning a Language Model

MLE for Deep Neural LM

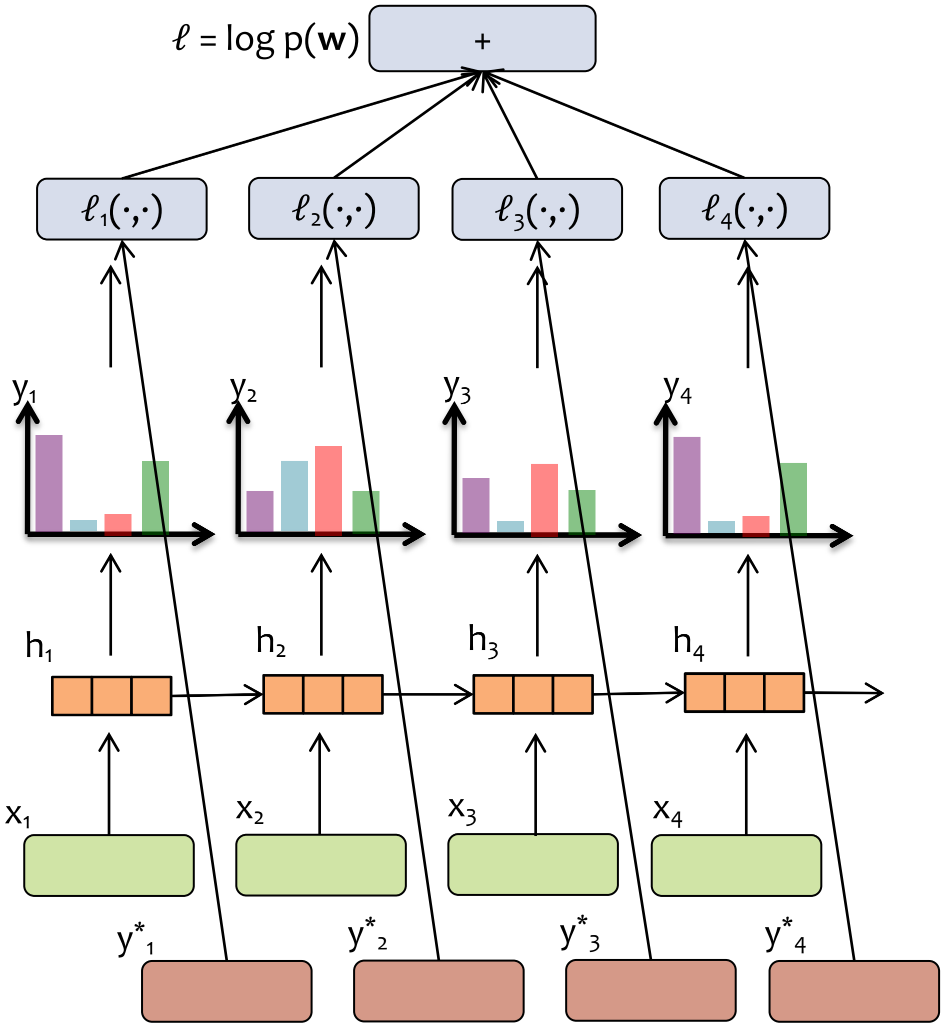
- We can also use maximum likelihood estimation to learn the parameters of an RNN-LM or Transformer-LM too!
- But **not in closed form** – instead we follow a different recipe:
  - Write the **likelihood** of the sentences under the Deep Neural LM model
  - Compute the **gradient** of the (batch) likelihood w.r.t. the parameters **by AutoDiff**
  - Follow the negative gradient using **Mini-batch SGD** (or your favorite optimizer)

MLE for n-gram LM

- This counting method gives us the **maximum likelihood estimate** of the n-gram LM parameters
- We can derive it in the usual way:
  - **Write the likelihood** of the sentences under the n-gram LM
  - **Set the gradient to zero** and impose the constraint that the probabilities sum-to-one
  - **Solve** for the MLE

RNN + Loss

How can we use this to compute the loss for an RNN-LM?



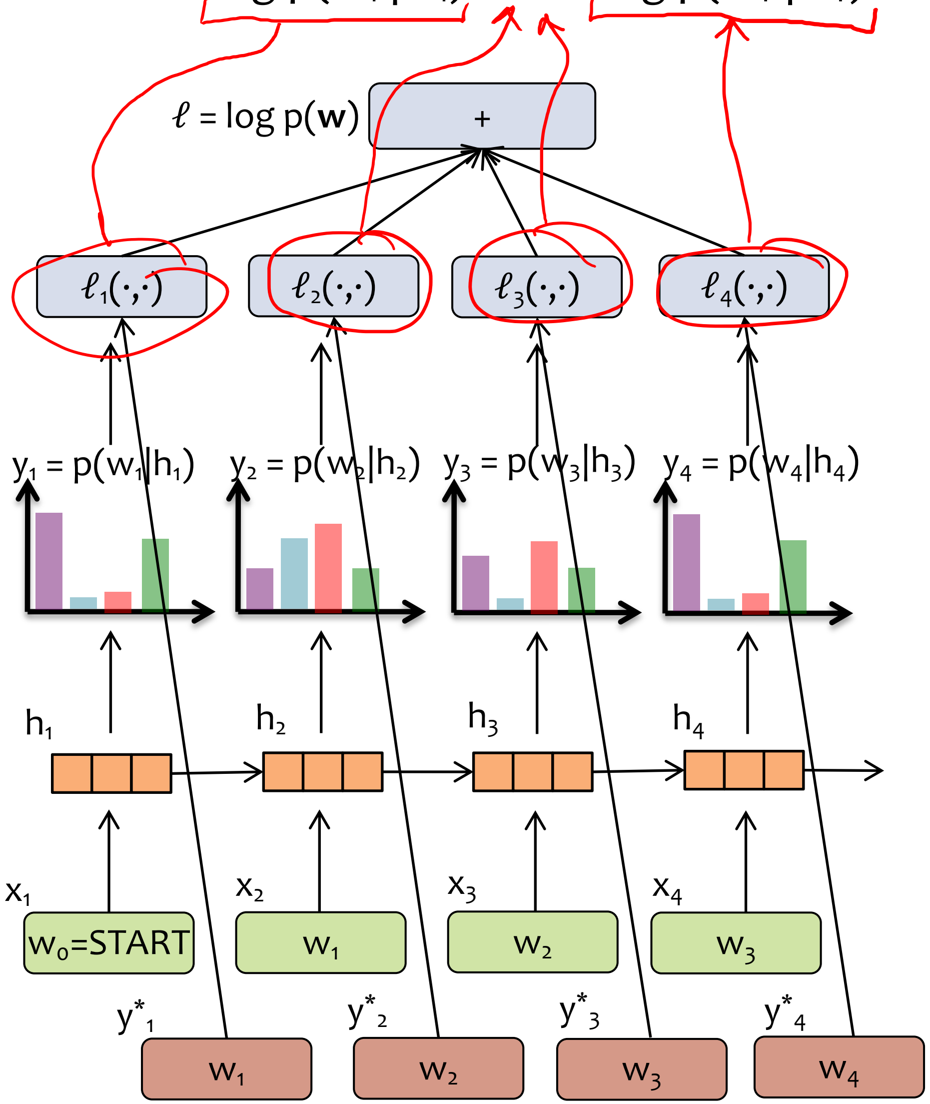
Algorithm 1 Elman RNN + Loss

procedure FORWARD($x_{1:T}, y^*_{1:T} W_{ah}, W_{ax}, b_a, W_{yh}, b_y$)
    Initialize the hidden state h_0 to zeros
    for t in 1 to T **do**
        Receive input data at time step t: x_t
        Compute the hidden state update:
            $a_t = W_{ah} \cdot h_{t-1} + W_{ax} \cdot x_t + b_a$
            $h_t = \sigma(a_t)$
        Compute the output at time step t:
            $y_t = \text{softmax}(W_{yh} \cdot h_t + b_y)$
        Compute the cross-entropy loss at time step t:
            $\ell_t = -\sum_{k=1}^{K} (y^*_t)_k \log((y_t)_k)$
    Compute the total loss:
        $\ell = \sum_{t=1}^{T} \ell_t$

RNN-LM + Loss

How can we use this to compute the loss for an RNN-LM?

$\log p(\mathbf{w}) = \log p(w_1, w_2, w_3, \ldots, w_T)$
$= \log p(w_1 \mid h_1) + \ldots + \log p(w_T \mid h_T)$



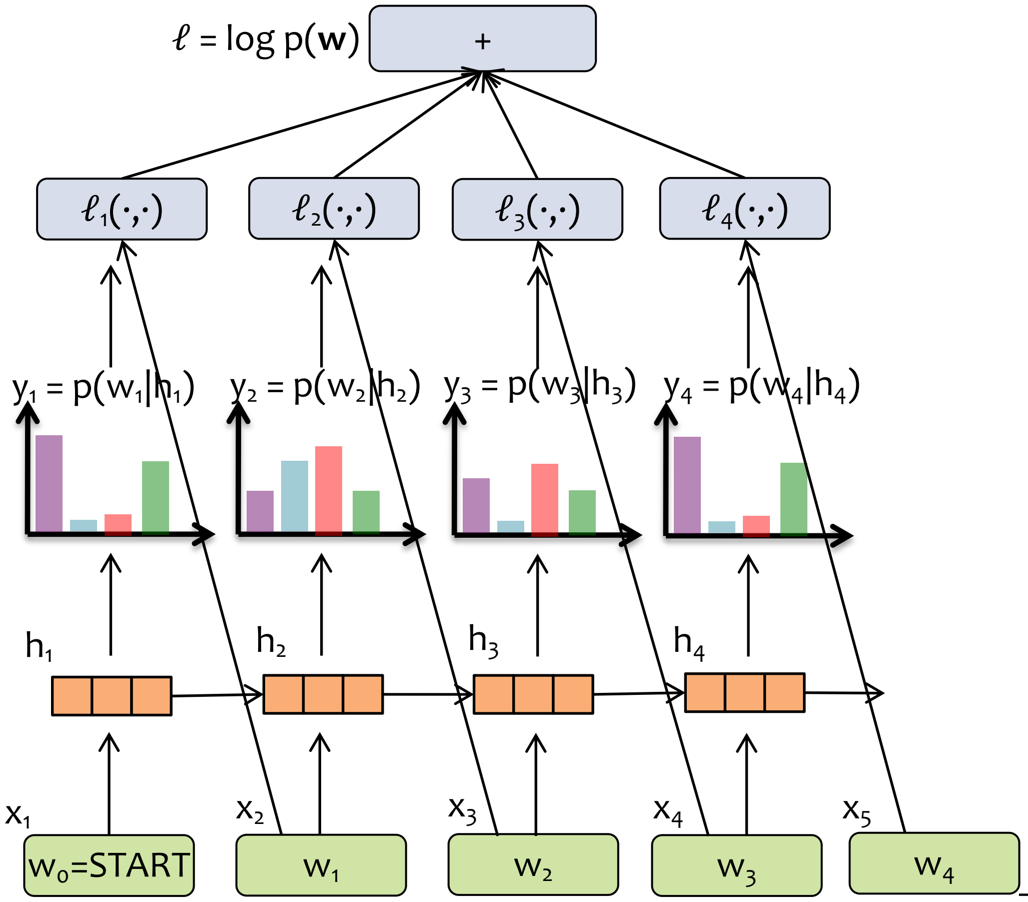
Algorithm 1 Elman RNN + Loss

1: **procedure** FORWARD($x_{1:T}, y^*_{1:T} W_{ah}, W_{ax}, b_a, W_{yh}, b_y$)
2: Initialize the hidden state h_0 to zeros
3: **for** t in 1 to T **do**
4: Receive input data at time step t: x_t
5: Compute the hidden state update:
6: $a_t = W_{ah} \cdot h_{t-1} + W_{ax} \cdot x_t + b_a$
7: $h_t = \sigma(a_t)$
8: Compute the output at time step t:
9: $y_t = \text{softmax}(W_{yh} \cdot h_t + b_y)$
10: Compute the cross-entropy loss at time step t:
11: $\ell_t = -\sum_{k=1}^{K} (y^*_t)_k \log((y_t)_k)$
12: Compute the total loss:
13: $\ell = \sum_{t=1}^{T} \ell_t$

RNN-LM + Loss

How can we use this to compute the loss for an RNN-LM?

$$-\log p(\mathbf{w}) = -\log p(w_1, w_2, w_3, \ldots, w_T)$$
$$= -\log p(w_1 \mid h_1) + \ldots + \log p(w_T \mid h_T)$$



Algorithm 1 Elman RNN + Loss

procedure FORWARD($\overbrace{\boxed{x_{1:T}, y^*_{1:T}}}^{w_{1:T}}$ $W_{ah}, W_{ax}, b_a, W_{yh}, b_y$)

Initialize the hidden state h_0 to zeros

for t in 1 to T **do**

Receive input data at time step t: x_t

Compute the hidden state update:

$a_t = W_{ah} \cdot h_{t-1} + W_{ax} \cdot \underbrace{\cancel{x_t}}_{w_{t-1}} + b_a$

$h_t = \sigma(a_t)$

Compute the output at time step t:

$y_t = \text{softmax}(W_{yh} \cdot h_t + b_y)$

Compute the cross-entropy loss at time step t:

$\ell_t = -\sum_{k=1}^{K} (\underbrace{\cancel{y^*_t}}_{w_t})_k \log(\underbrace{(y_t)_k}_{y_{t,k}})$

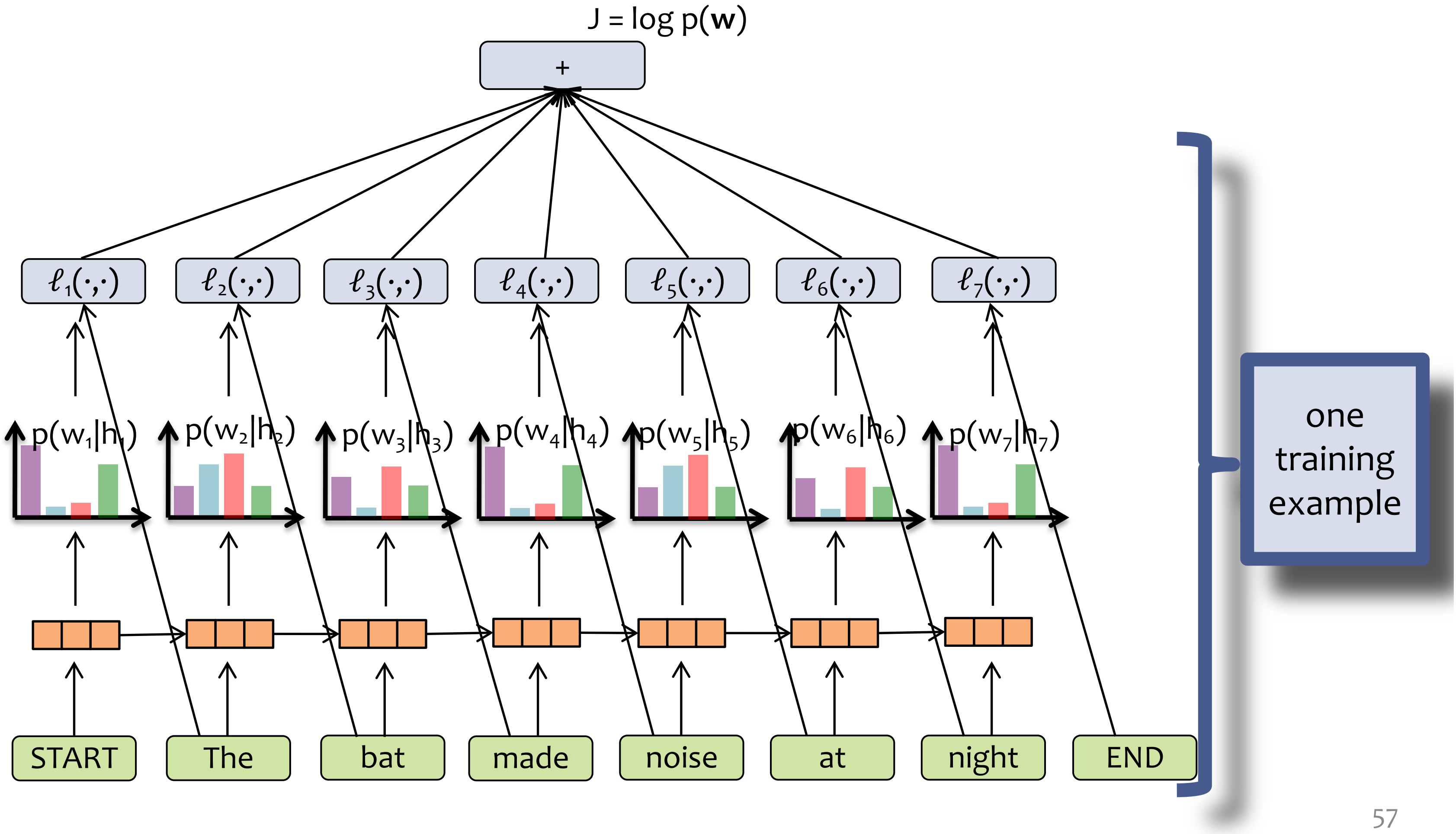
Compute the total loss:

$\ell = \sum_{t=1}^{T} \ell_t$

Learning an RNN-LM

- Each training example is a sequence (e.g. sentence), so we have training data D = {$\mathbf{w}^{(1)}$, $\mathbf{w}^{(2)}$, …, $\mathbf{w}^{(N)}$}
- The objective function for a Deep LM (e.g. RNN-LM or Tranformer-LM) is typically the log-likelihood of the training examples:
  $$J(\boldsymbol{\theta}) = \Sigma_i \log p_{\boldsymbol{\theta}}(\mathbf{w}^{(i)})$$
- We train by mini-batch SGD (or your favorite flavor of mini-batch SGD)

$$\log p(\mathbf{w}) = \log p(w_1, w_2, w_3, \ldots, w_T)$$
$$= \log p(w_1 \mid h_1) + \log p(w_2 \mid h_2) + \ldots + \log p(w_T \mid h_T)$$



LARGE LANGUAGE MODELS

How large are LLMs?

Comparison of some recent **large language models** (LLMs)

Model	Creators	Year of release	Training Data (# tokens)	Model Size (# parameters)
GPT-2	OpenAI	2019	~10 billion (40Gb)	1.5 billion
GPT-3	OpenAI	2020	300 billion	175 billion
PaLM	Google	2022	780 billion	540 billion
Chinchilla	DeepMind	2022	1.4 trillion	70 billion
LaMDA (cf. Bard)	Google	2022	1.56 trillion	137 billion
LLaMA	Meta	2023	1.4 trillion	65 billion
LLaMA-2	Meta	2023	2 trillion	70 billion
GPT-4	OpenAI	2023	?	? (1.76 trillion)
Gemini (Ultra)	Google	2023	?	? (1.5 trillion)
LLaMA-3	Meta	2024	15 trillion	405 billion

What is ChatGPT?

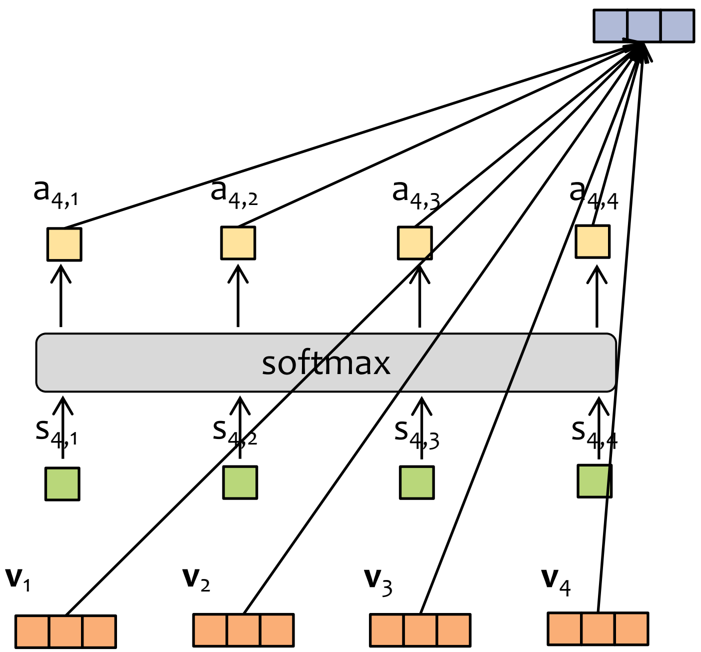
- ChatGPT is a large (in the sense of having many parameters) language model, fine-tuned to be a dialogue agent
- The base language model is GPT-3.5 which was trained on a large quantity of text

Transformer Language Models

MODEL: GPT

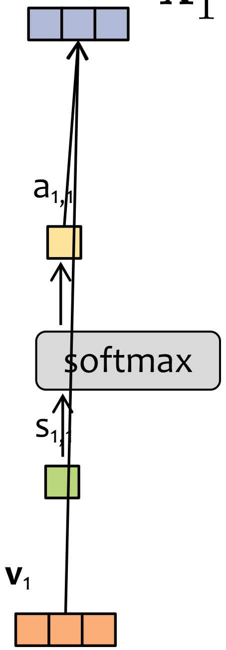
Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

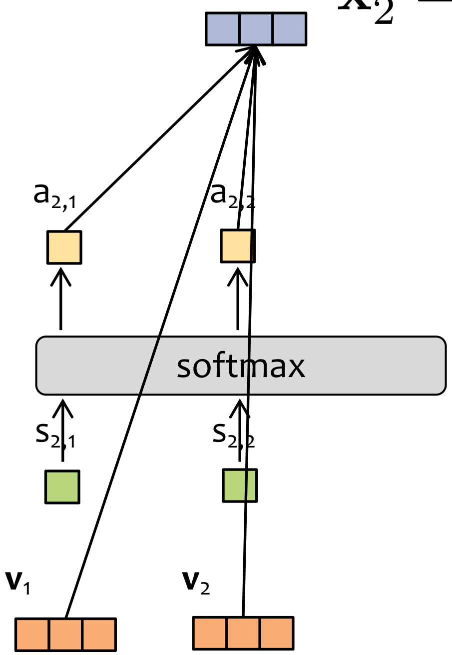


Attention

$$\mathbf{x}'_1 = \sum_{j=1}^{1} a_{1,j}\mathbf{v}_j$$

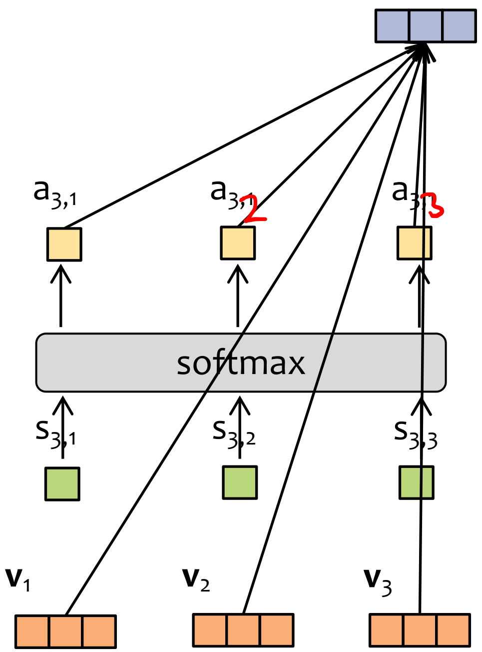


Attention



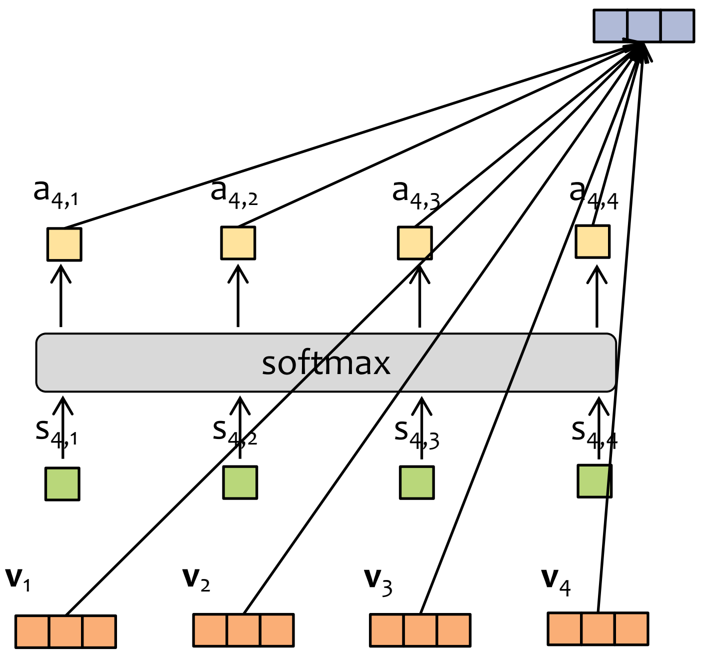
$$\mathbf{x}'_2 = \sum_{j=1}^{2} a_{2,j}\mathbf{v}_j$$

Attention



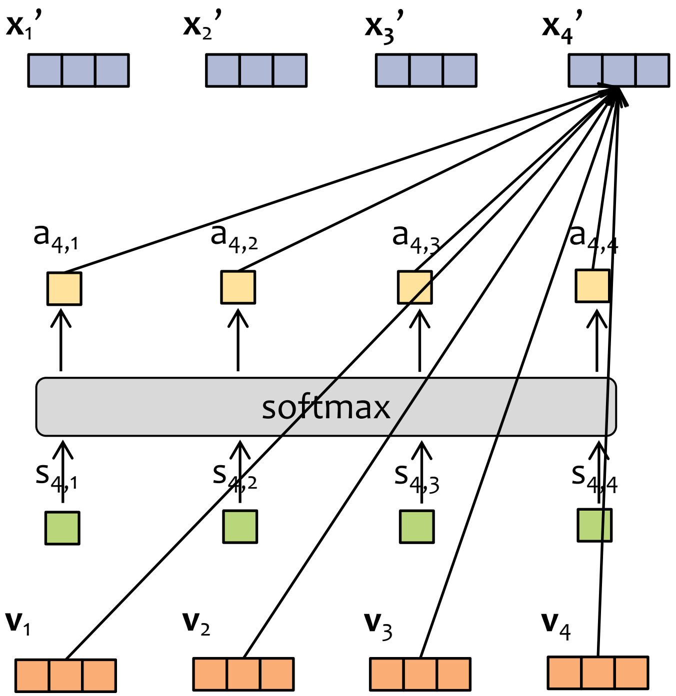
$$\mathbf{x}'_3 = \sum_{j=1}^{3} a_{3,j}\mathbf{v}_j$$

Attention



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

Attention



$$\mathbf{x}'_t = \sum_{j=1}^{t} a_{t,j}\mathbf{v}_j$$

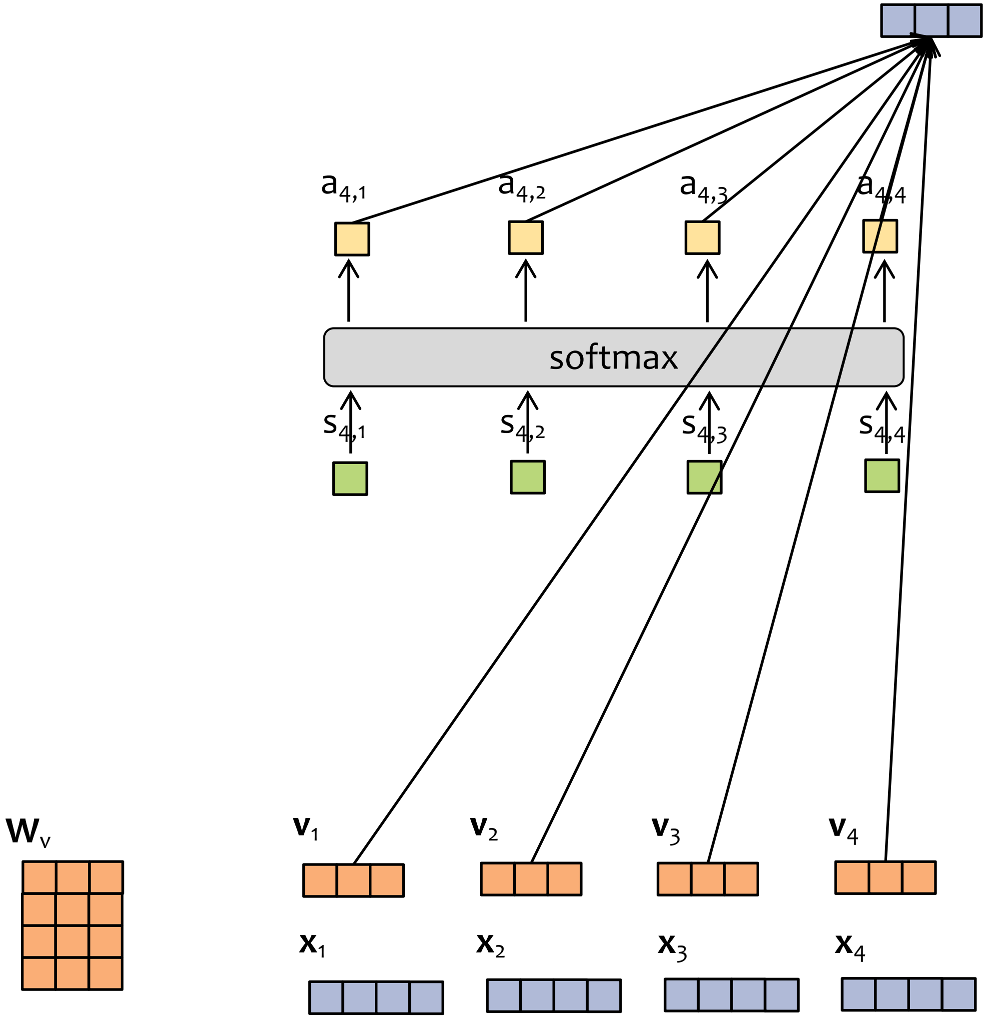
attention weights

scores

values

Scaled Dot-Product Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

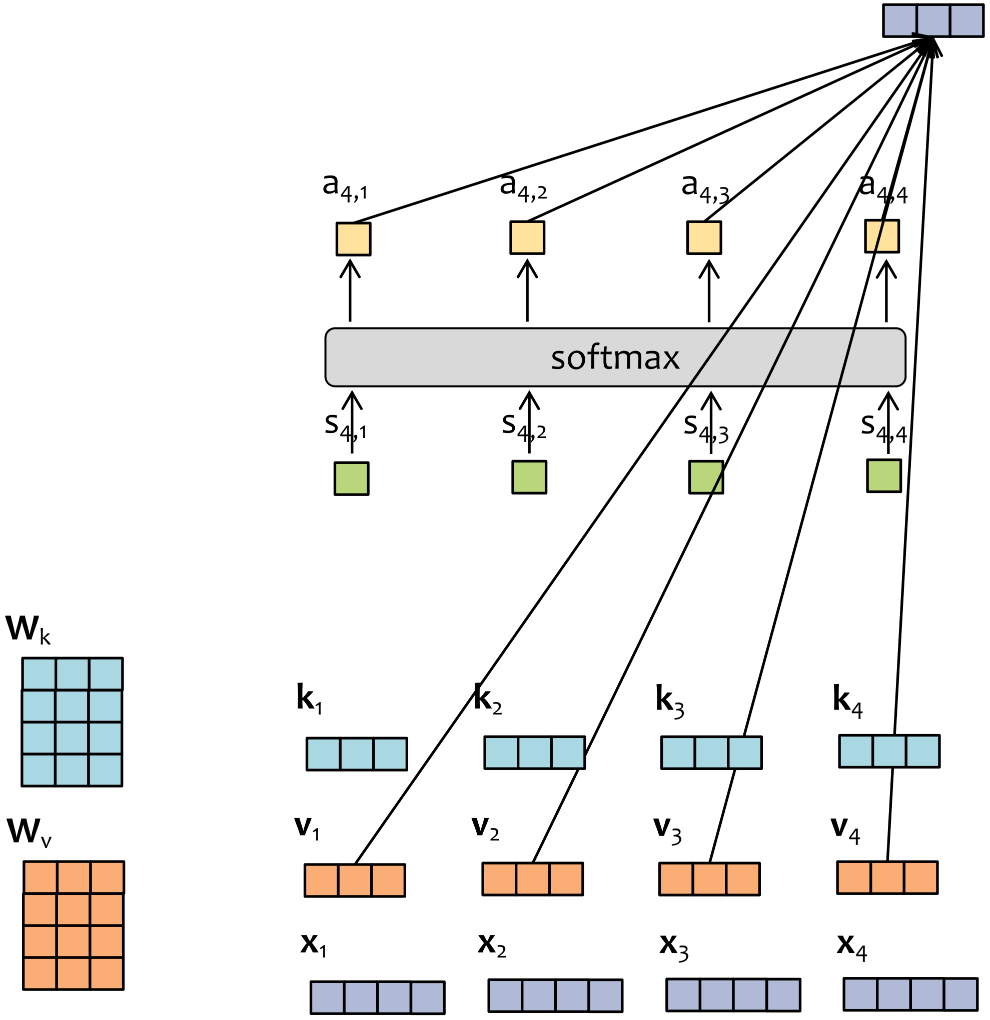


$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$$

values

Scaled Dot-Product Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j} \mathbf{v}_j$$

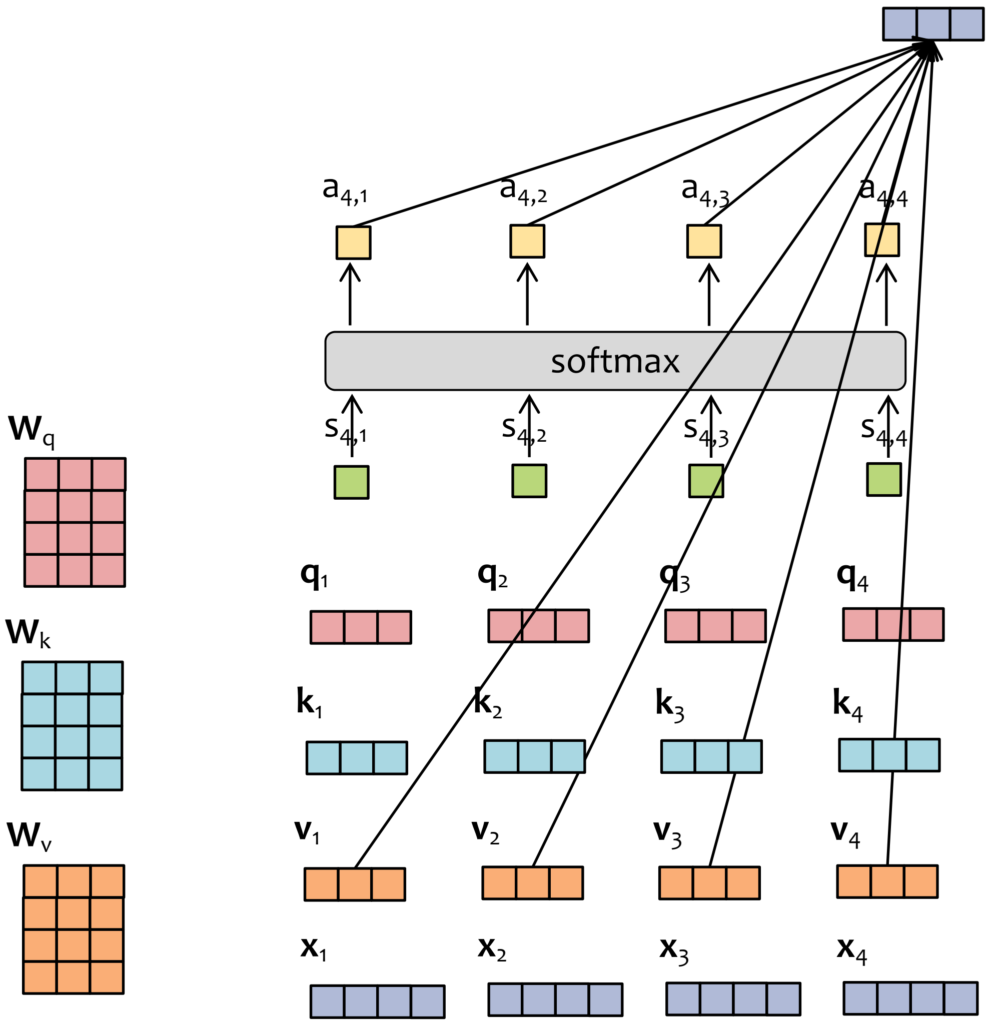


$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j \qquad \text{keys}$$

$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j \qquad \text{values}$$

Scaled Dot-Product Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

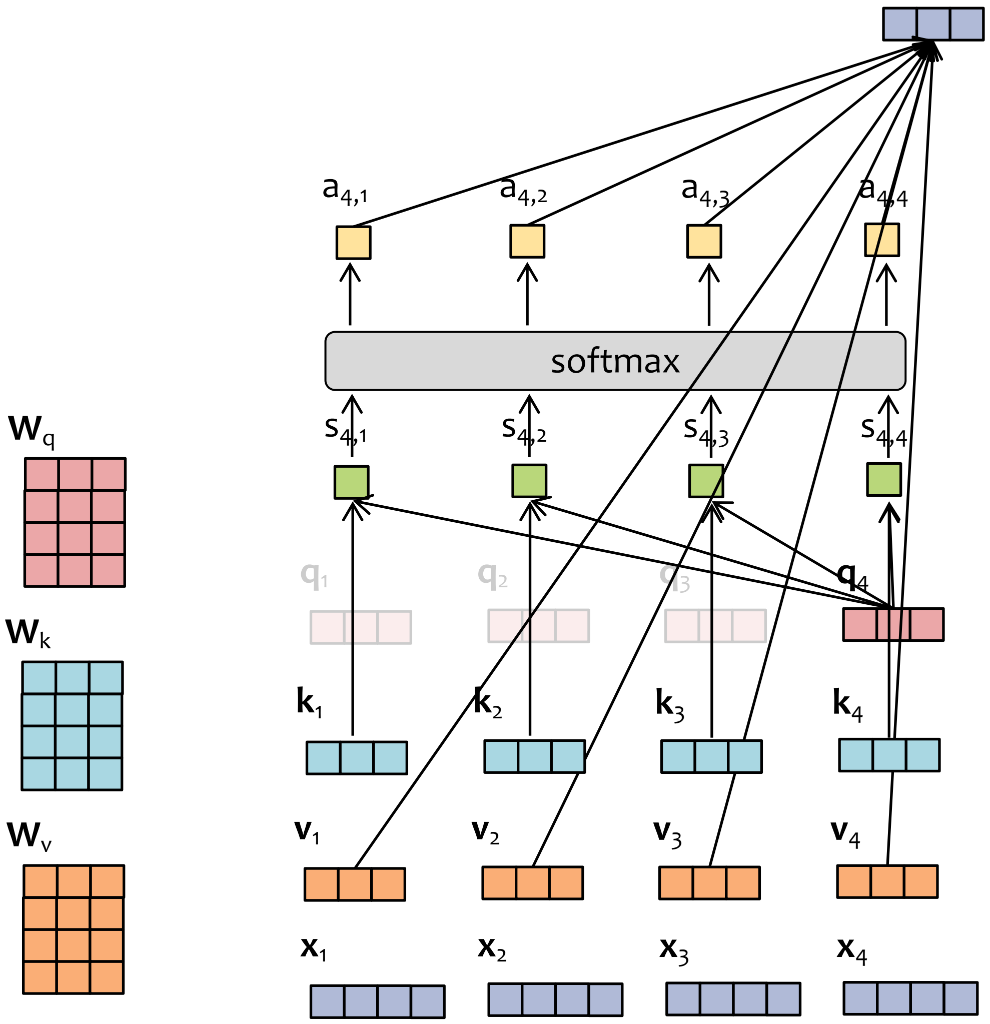


$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$$ queries

$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$$ keys

$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j} \mathbf{v}_j$$

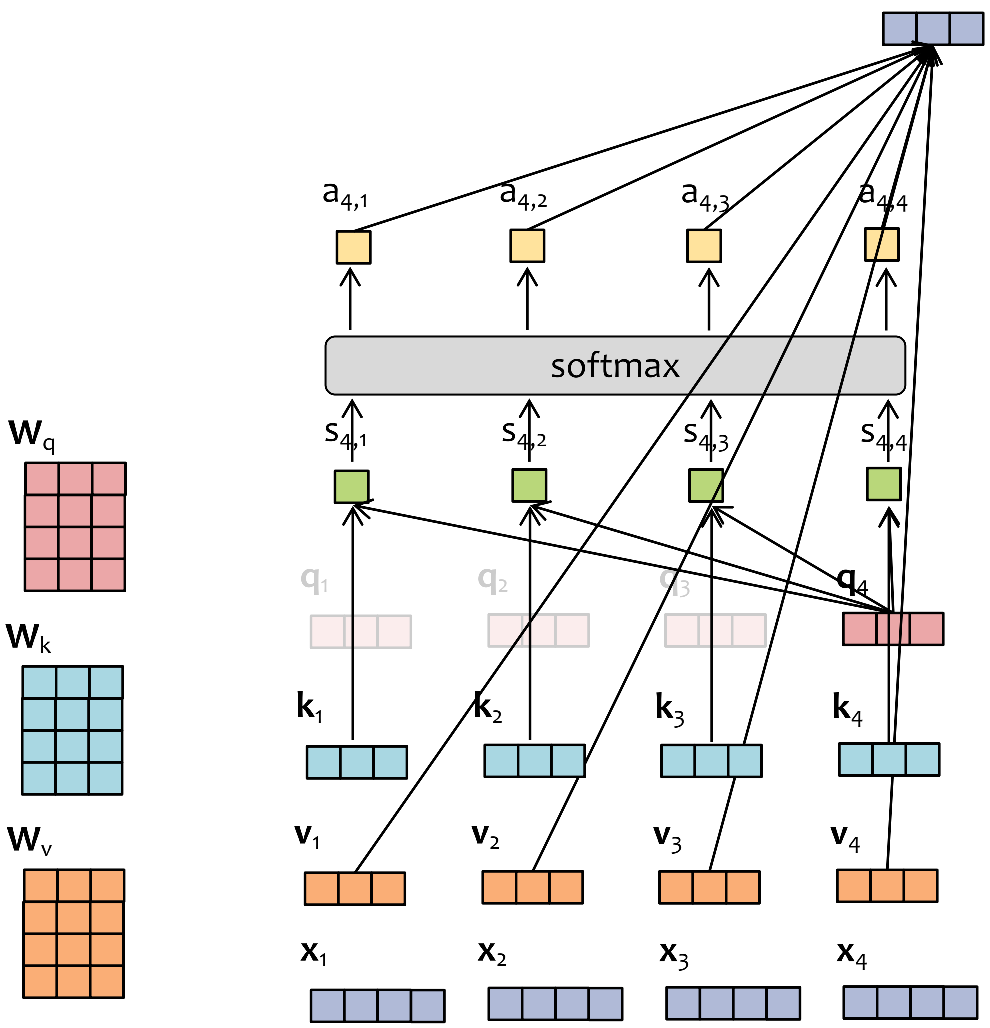
$$s_{4,j} = \mathbf{k}_j^T \mathbf{q}_4 / \sqrt{d_k}$$ scores

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$$ queries

$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$$ keys

$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

$\mathbf{a}_4 = \text{softmax}(\mathbf{s}_4)$ attention weights

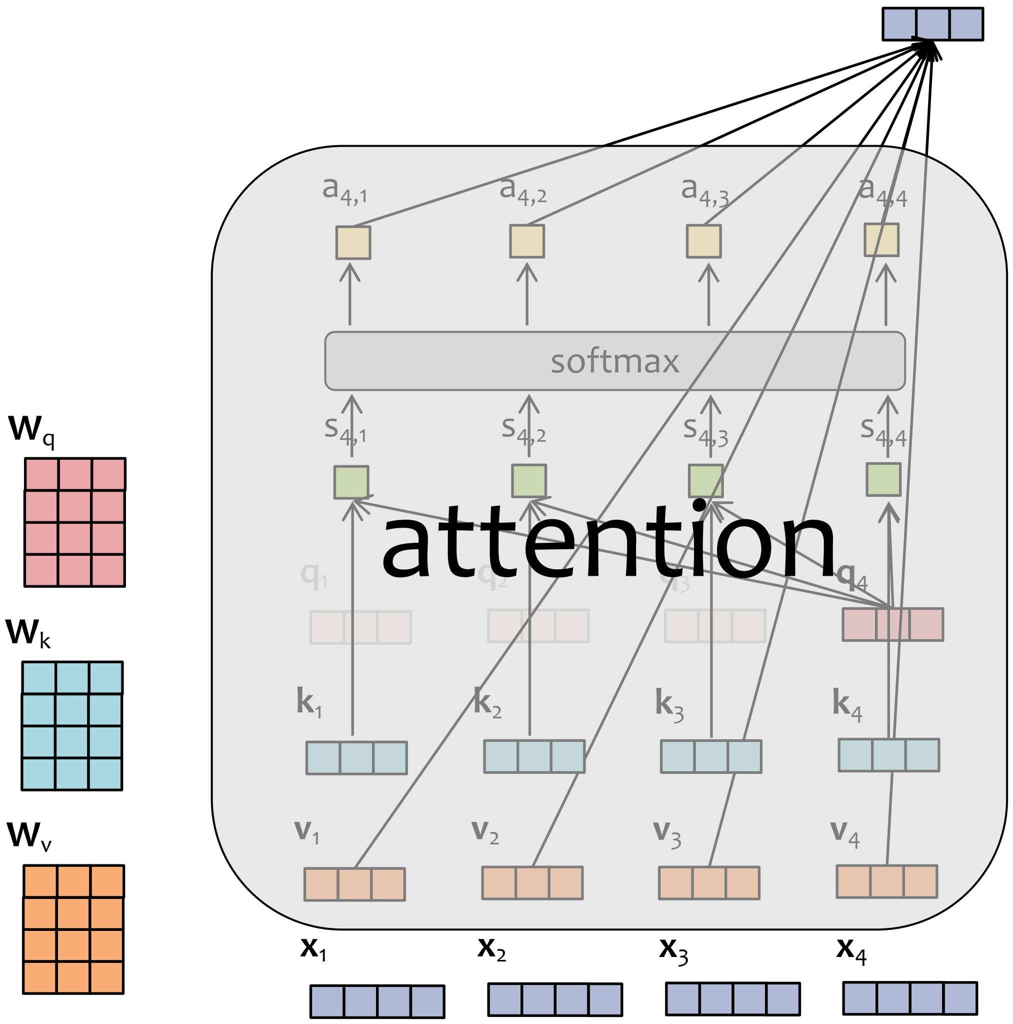
$s_{4,j} = \mathbf{k}_j^T\mathbf{q}_4/\sqrt{d_k}$ scores

$\mathbf{q}_j = \mathbf{W}_q^T\mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T\mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T\mathbf{x}_j$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

$\mathbf{a}_4 = \text{softmax}(\mathbf{s}_4)$ attention weights

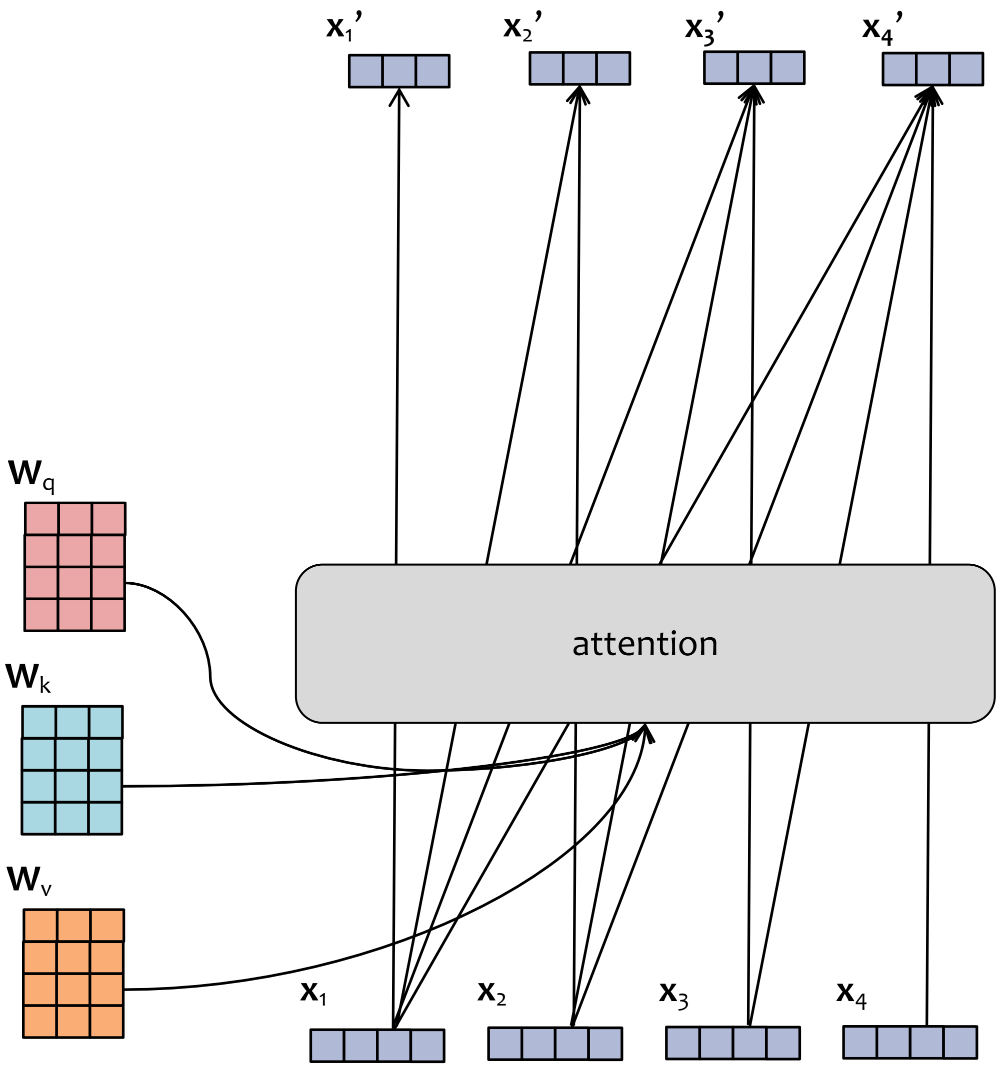
$s_{4,j} = \mathbf{k}_j^T \mathbf{q}_4 / \sqrt{d_k}$ scores

$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_t = \sum_{j=1}^{t} a_{t,j}\mathbf{v}_j$$

$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t)$ attention weights

$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{d_k}$ scores

$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Animation of 3D Convolution

Recall…

http://cs231n.github.io/convolutional-networks/

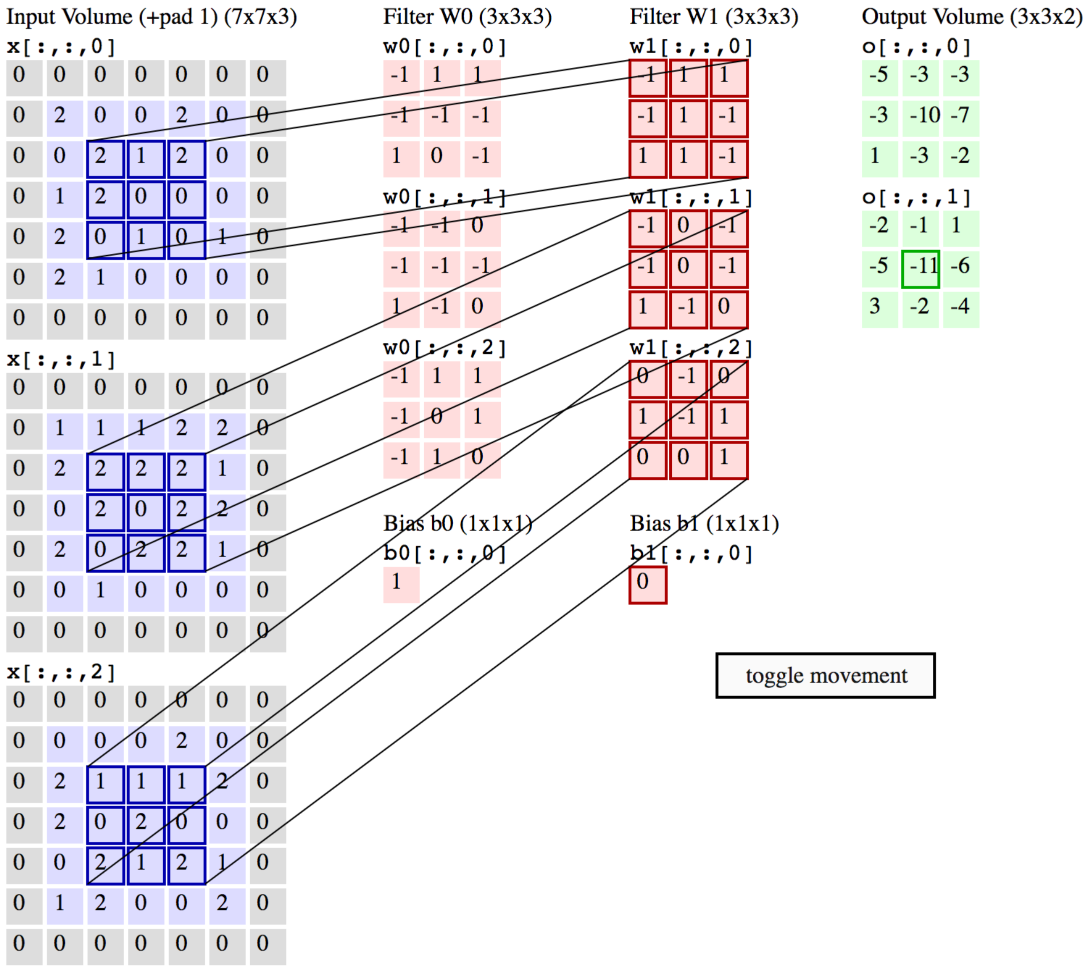
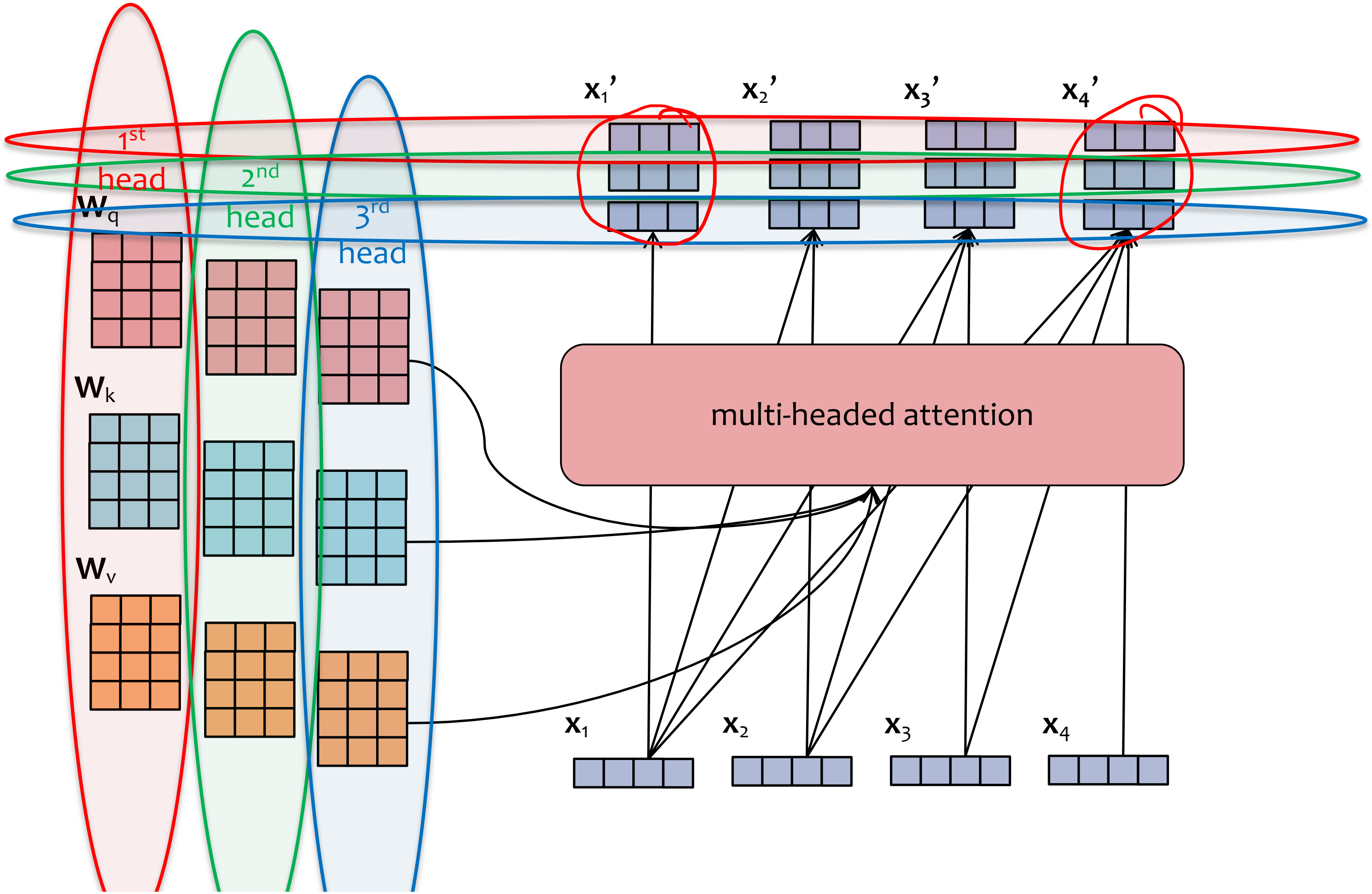


Figure from Fei-Fei Li & Andrej Karpathy & Justin Johnson (CS231N)

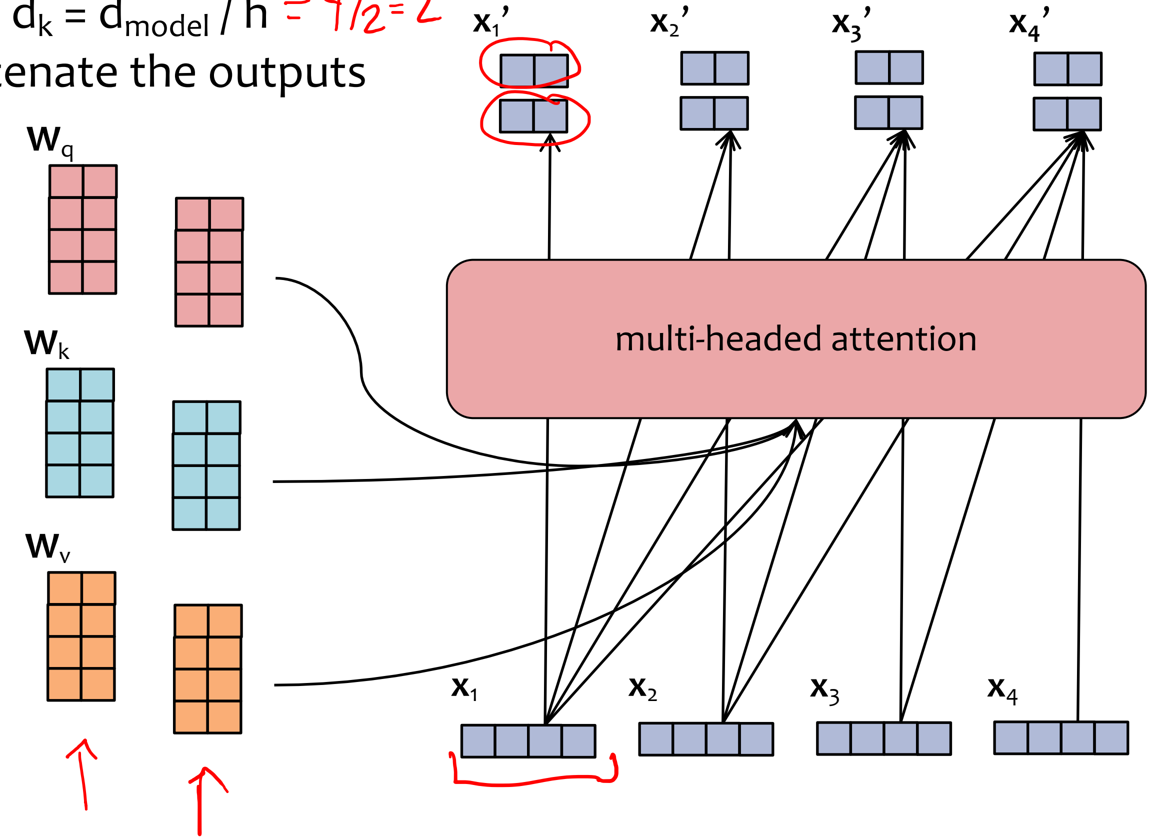
Multi-headed Attention



- Just as we can have **multiple channels** in a **convolution** layer, we can use **multiple heads** in an **attention** layer
- Each head gets **its own parameters**
- We can **concatenate** all the outputs to get a single vector for each time step

Multi-headed Attention

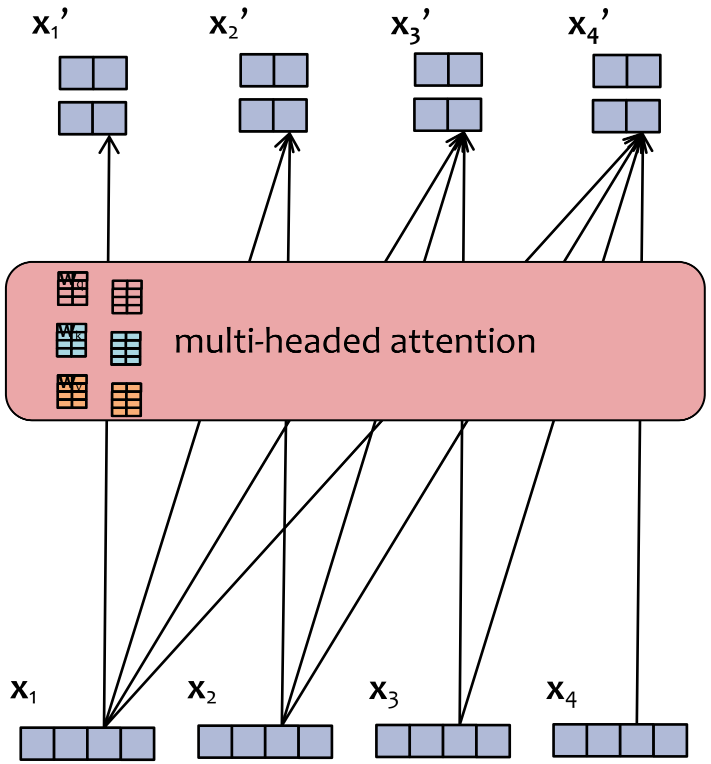
- To ensure the dimension of the **input** embedding $\mathbf{x}_t$ is the same as the **output** embedding $\mathbf{x}_t'$, Transformers usually choose the embedding sizes and number of heads appropriately:
  - d_{model} = dim. of inputs $= 4$
  - d_k = dim. of each output
  - h = # of heads $= 2$
  - Choose $d_k = d_{model} / h = 4/2 = 2$
- Then concatenate the outputs



- Just as we can have **multiple channels** in a **convolution** layer, we can use **multiple heads** in an **attention** layer
- Each head gets **its own parameters**
- We can **concatenate** all the outputs to get a single vector for each time step

Multi-headed Self- Attention

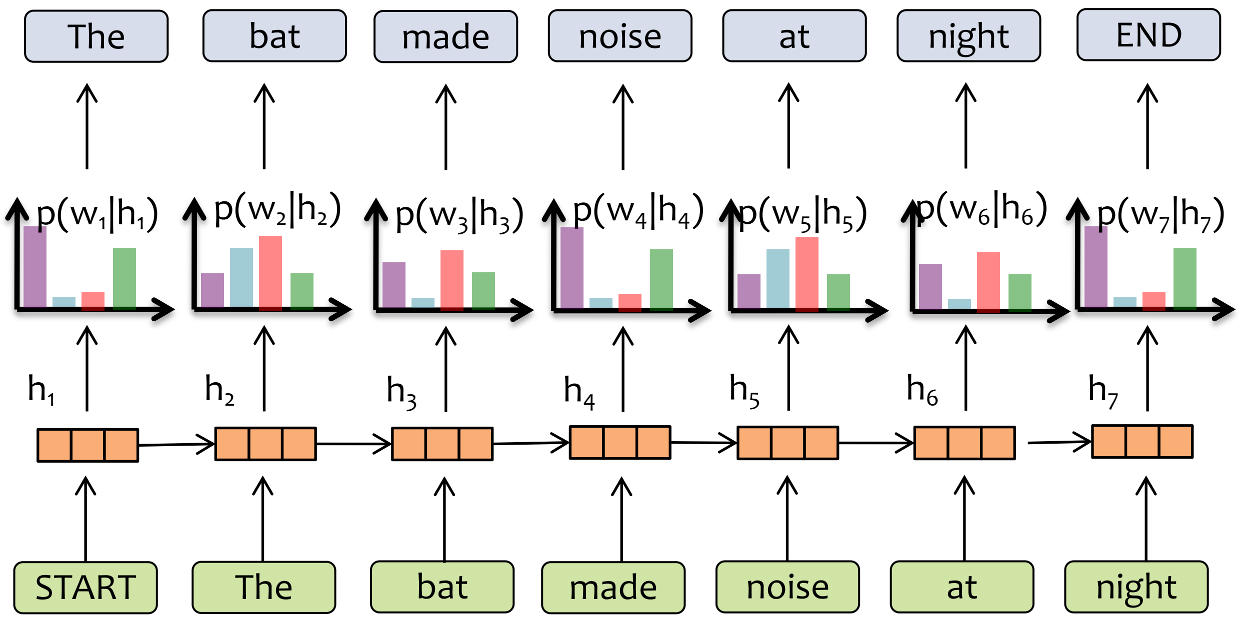
- To ensure the dimension of the **input** embedding $\mathbf{x}_t$ is the same as the **output** embedding $\mathbf{x}_t'$, Transformers usually choose the embedding sizes and number of heads appropriately:
  - d_{model} = dim. of inputs
  - d_k = dim. of each output
  - h = # of heads
  - Choose $d_k = d_{model}$ / h
- Then concatenate the outputs



- Just as we can have **multiple channels** in a **convolution** layer, we can use **multiple heads** in an **attention** layer
- Each head gets **its own parameters**
- We can **concatenate** all the outputs to get a single vector for each time step

RNN Language Model

Recall...



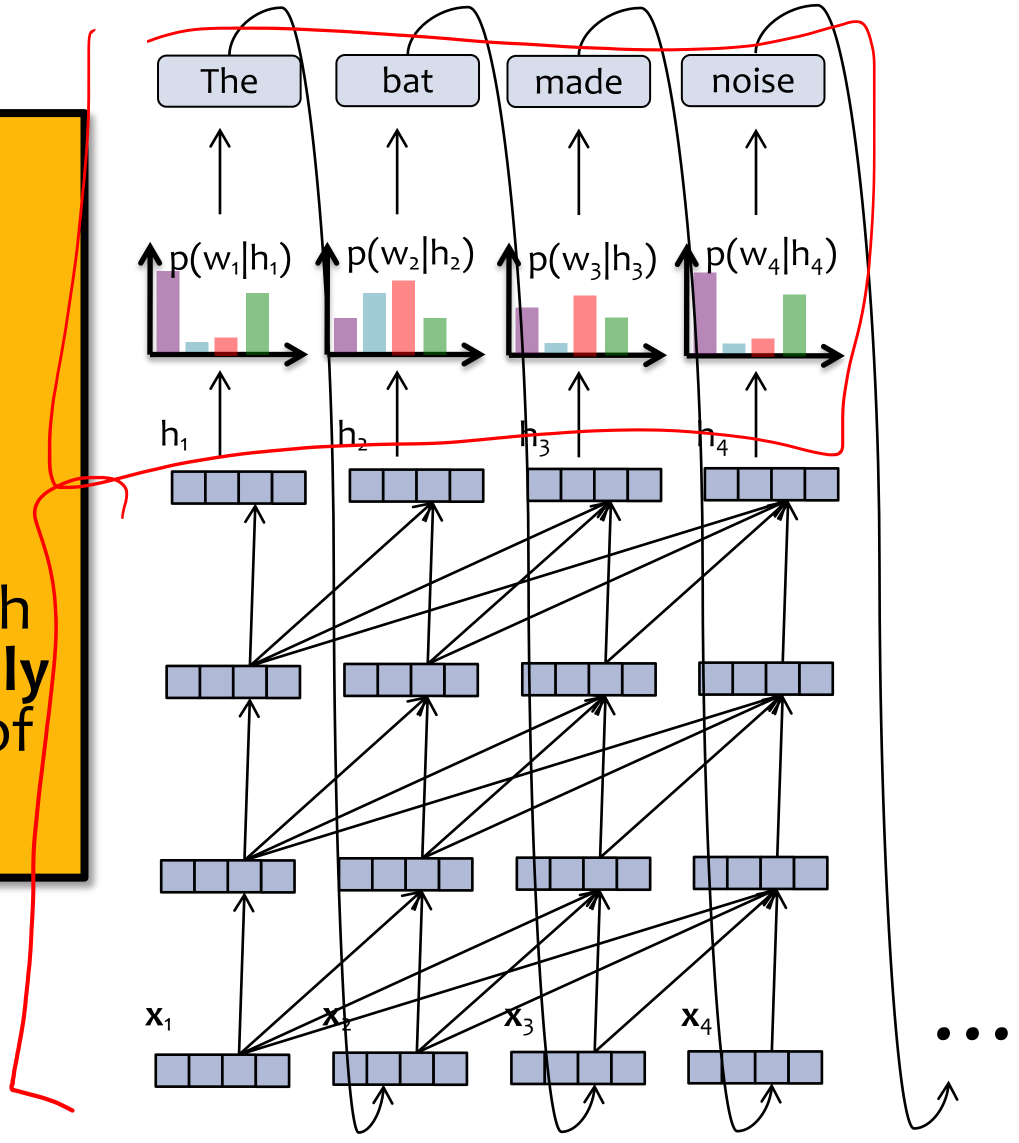
<u>Key Idea</u>:

(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 $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

Transformer Language Model

Important!
- RNN computation graph grows **linearly** with the number of input tokens
- Transformer-LM computation graph grows **quadratically** with the number of input tokens



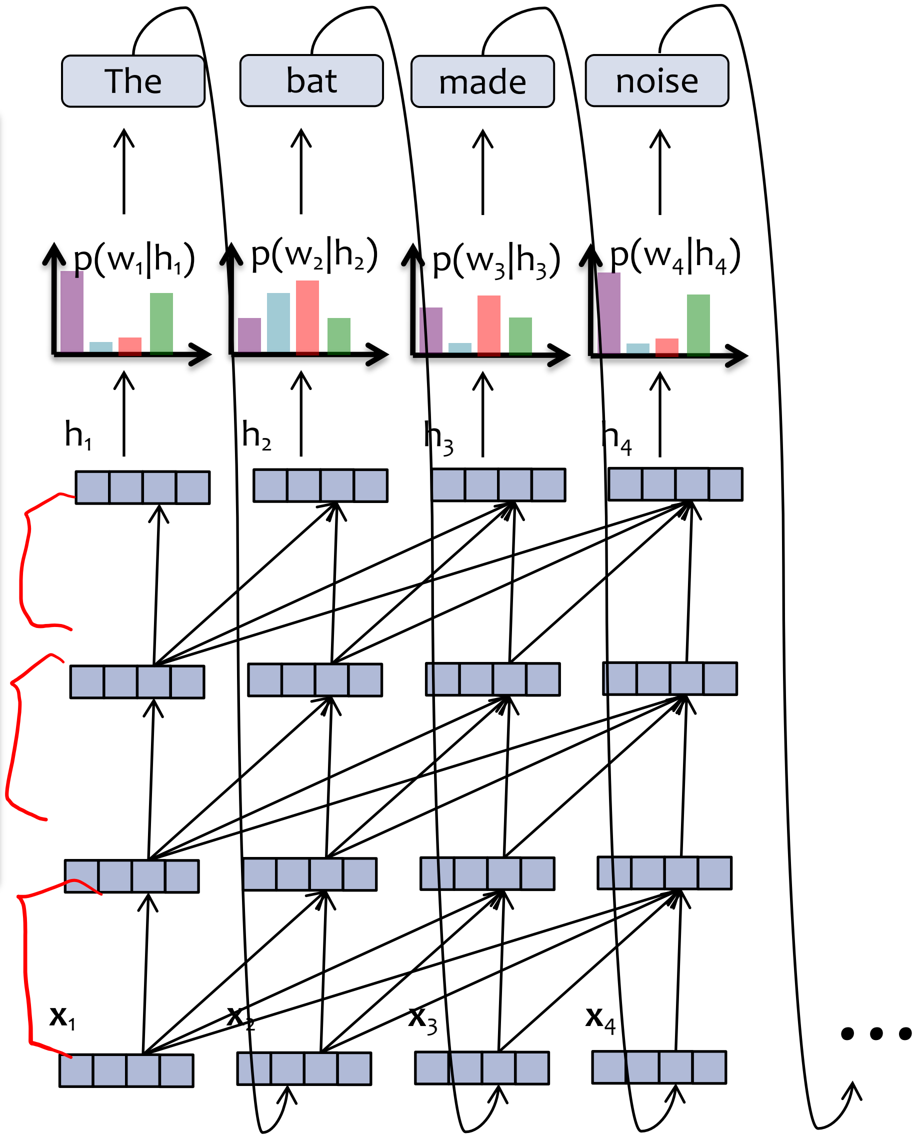
Each hidden vector looks back at the hidden vectors of the **current and previous timesteps in the previous layer.**

The language model part is just like an RNN-LM!

Transformer Language Model

Important!

- RNN computation graph grows **linearly** with the number of input tokens
- Transformer-LM computation graph grows **quadratically** with the number of input tokens



Each layer of a Transformer LM consists of several **sublayers**:

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of the **current and previous timesteps in the previous layer.**

The language model part is just like an RNN-LM!

Layer Normalization

- *The Problem:* **internal covariate shift** occurs during training of a deep network when a small change in the low layers amplifies into a large change in the high layers
- *One Solution:* **Layer normalization** normalizes each layer and learns elementwise gain/bias
- Such normalization allows for higher learning rates (for **faster convergence**) without issues of diverging gradients

Given input $\mathbf{a} \in \mathbb{R}^K$, LayerNorm computes output $\mathbf{b} \in \mathbb{R}^K$:

$$\mathbf{b} = \gamma \odot \frac{\mathbf{a} - \mu}{\sigma} \oplus \boldsymbol{\beta}$$

where we have mean $\mu = \frac{1}{K} \sum_{k=1}^{K} a_k$,

standard deviation $\sigma = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (a_k - \mu)^2}$,

and parameters $\gamma \in \mathbb{R}^K, \boldsymbol{\beta} \in \mathbb{R}^K$.

$\odot$ and $\oplus$ denote elementwise multiplication and addition.

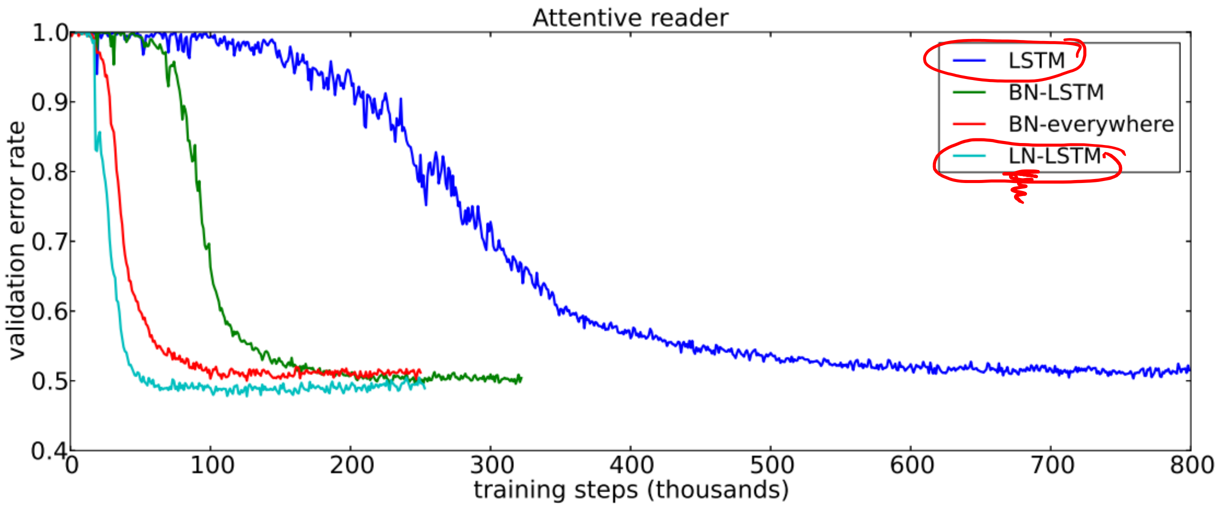
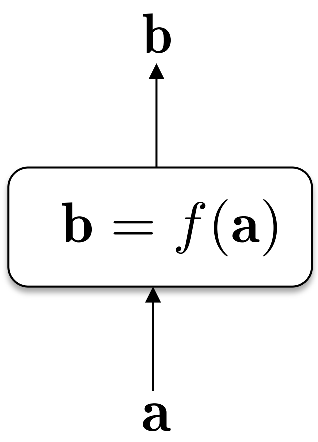


Figure from https://arxiv.org/pdf/1607.06450.pdf

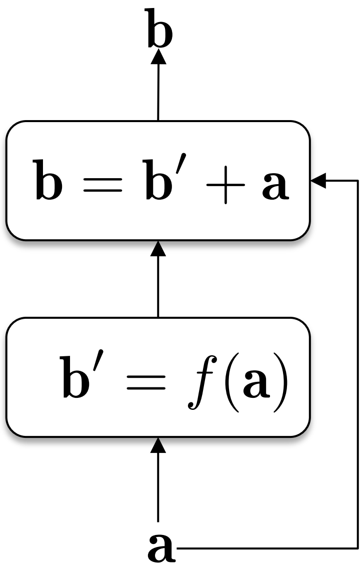
Residual Connections

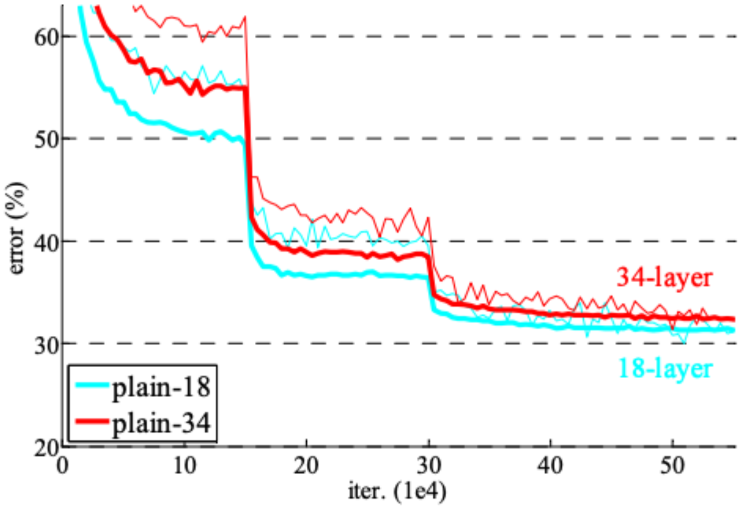
- *The Problem:* as network depth grows very large, a **performance degradation** occurs that is not explained by overfitting (i.e. train / test error both worsen)
- *One Solution:* **Residual connections** pass a copy of the input alongside another function so that information can flow more directly
- These residual connections allow for **effective training of very deep networks** that perform better than their shallower (though still deep) counterparts

Plain Connection



Residual Connection





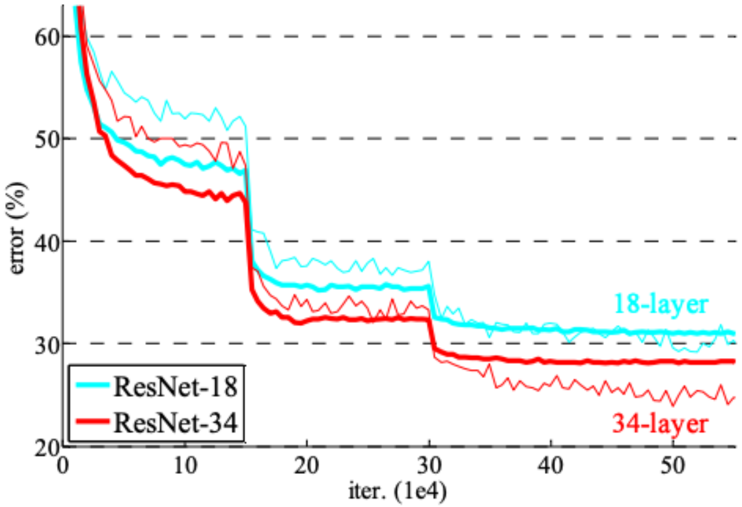
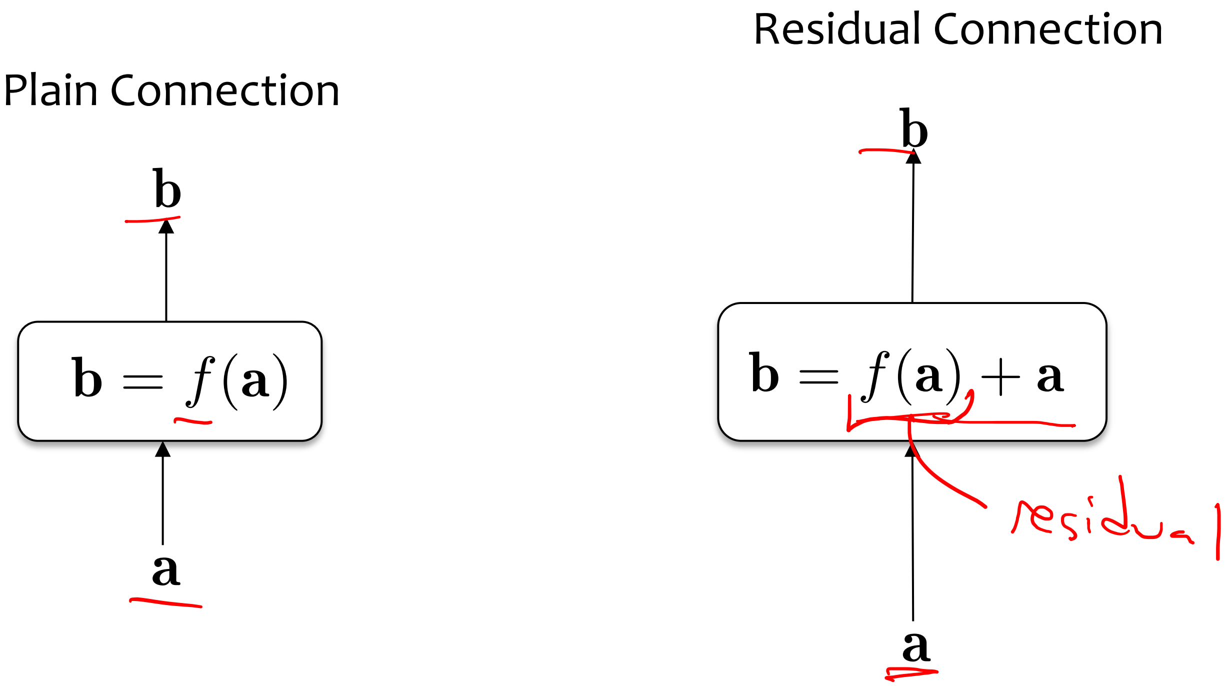


Figure from https://arxiv.org/pdf/1512.03385.pdf

Residual Connections

- *The Problem:* as network depth grows very large, a **performance degradation** occurs that is not explained by overfitting (i.e. train / test error both worsen)
- *One Solution:* **Residual connections** pass a copy of the input alongside another function so that information can flow more directly
- These residual connections allow for **effective training of very deep networks** that perform better than their shallower (though still deep) counterparts



Why are residual connections helpful?

Instead of f(a) having to learn a full transformation of a, f(a) only needs to learn an additive modification of a (i.e. the residual).

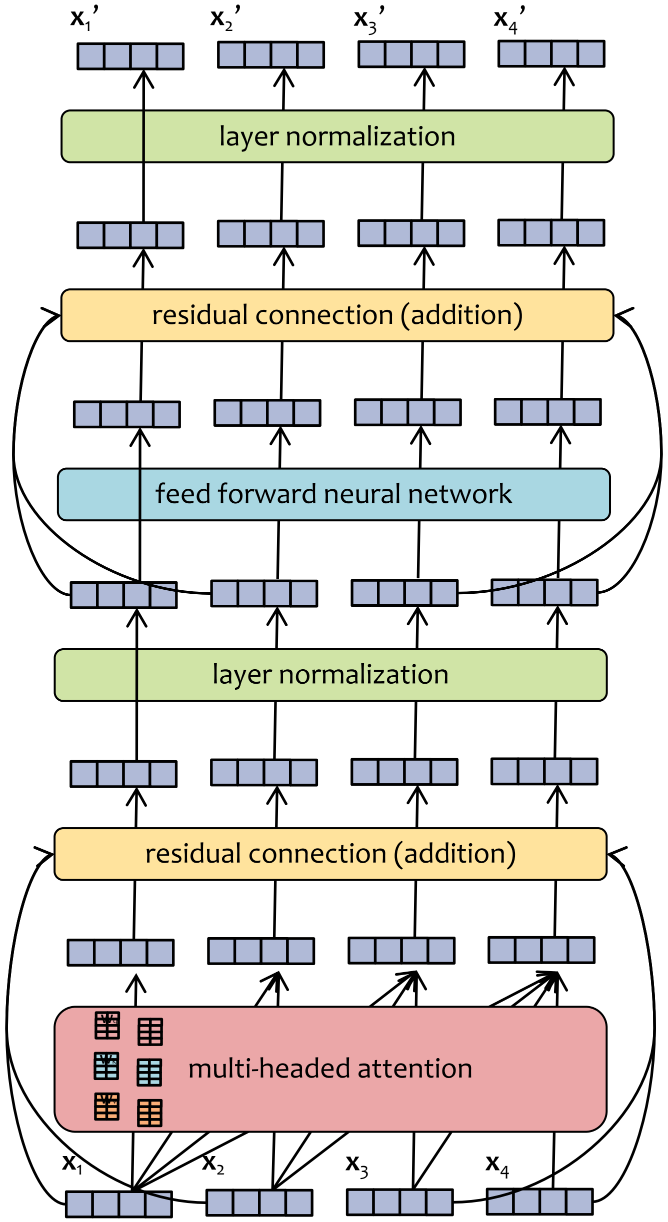
Figure from https://arxiv.org/pdf/1512.03385.pdf

Transformer Layer

Post-LN Version:

This is the version of the Transformer Layer that was introduced in the original paper in 2017.

The LayerNorm modules occur at the end of each set of 3 layers.



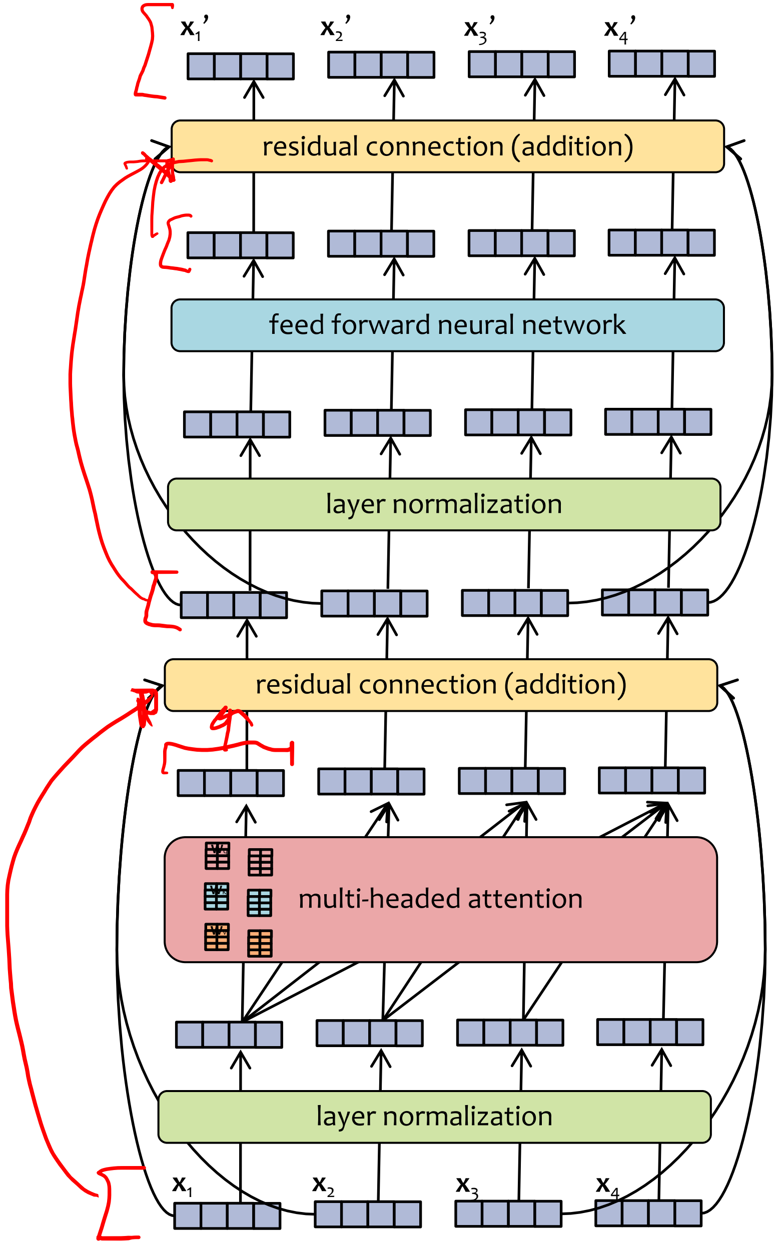
Each layer of a Transformer LM consists of several **sublayers:**

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Transformer Layer

Pre-LN Version:

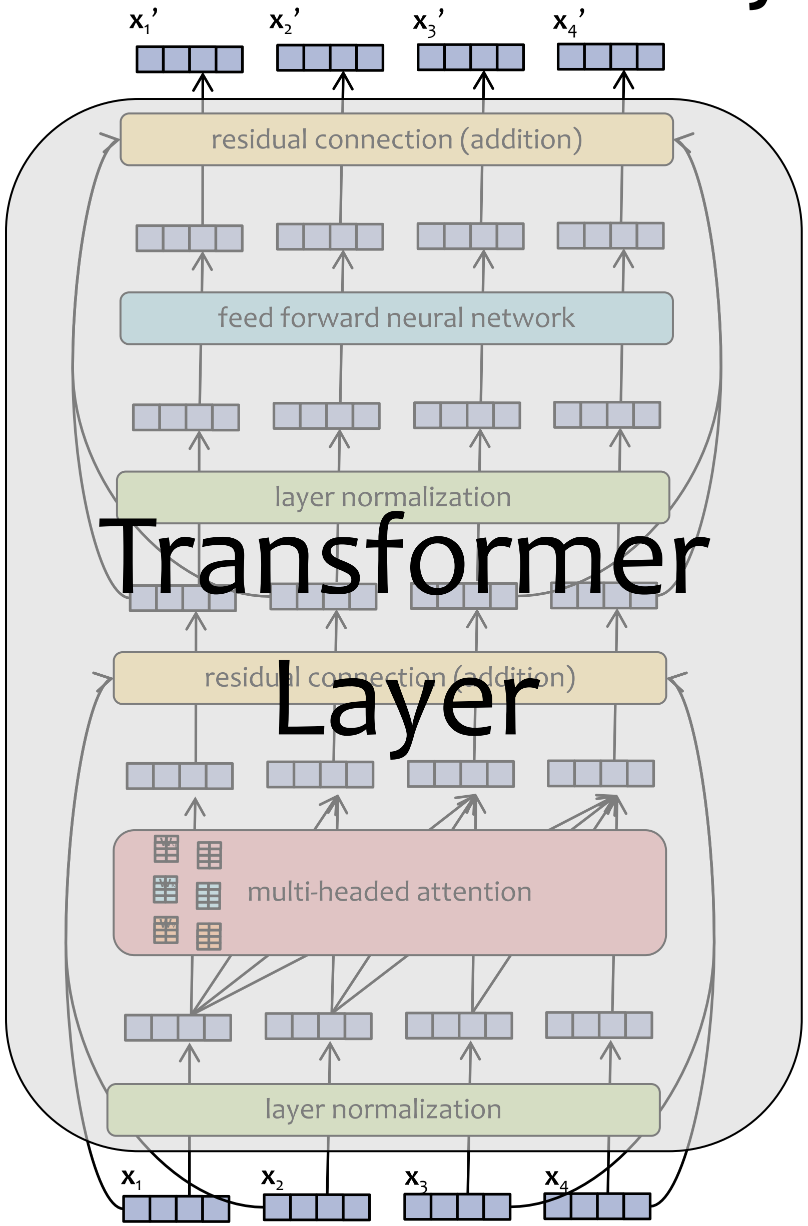
However, subsequent work found that reordering such that the LayerNorm's came at the beginning of each set of 3 layers, the multi-headed attention and feed-forward NN layers tend to be better behaved (i.e. tricks like warm-up are less important).



Each layer of a Transformer LM consists of several **sublayers:**

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

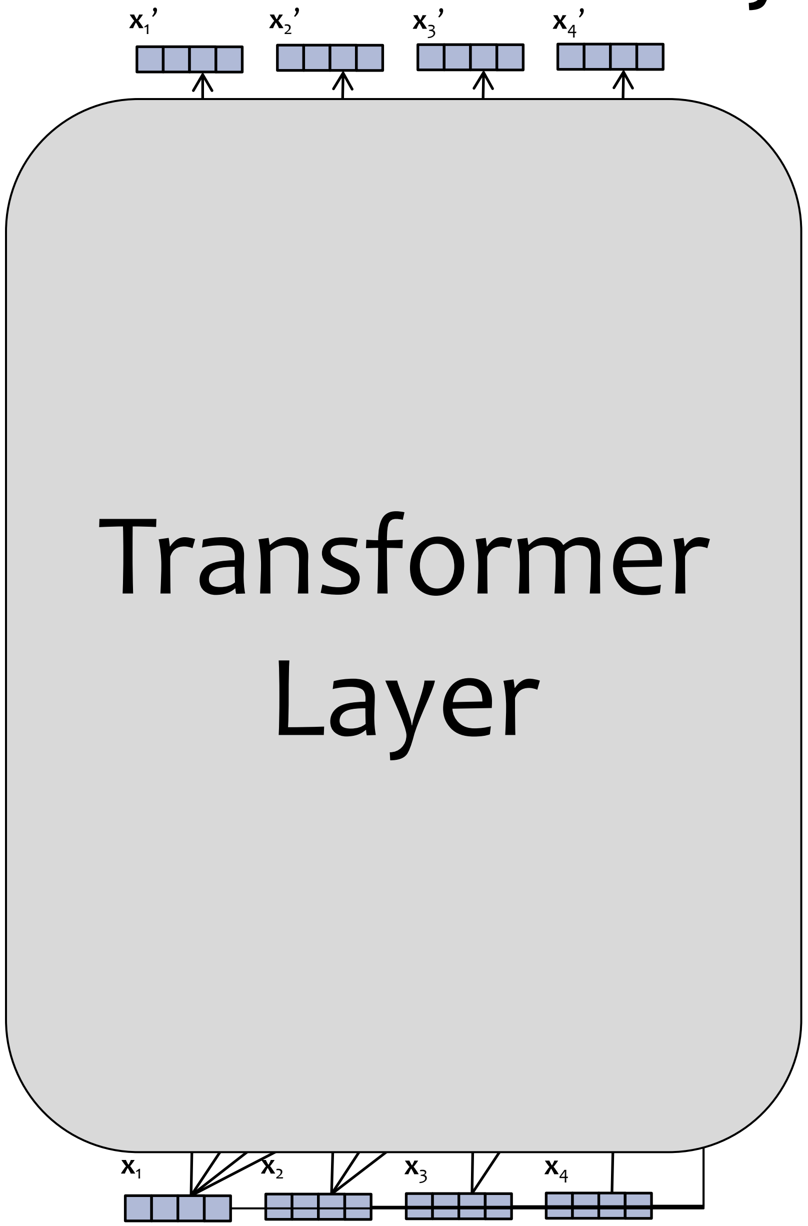
Transformer Layer



Each layer of a Transformer LM consists of several **sublayers:**

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Transformer Layer



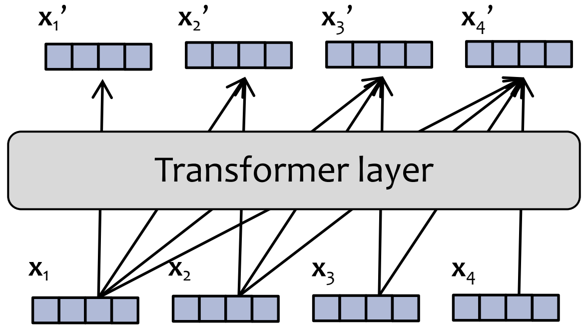
Each layer of a Transformer LM consists of several **sublayers:**

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

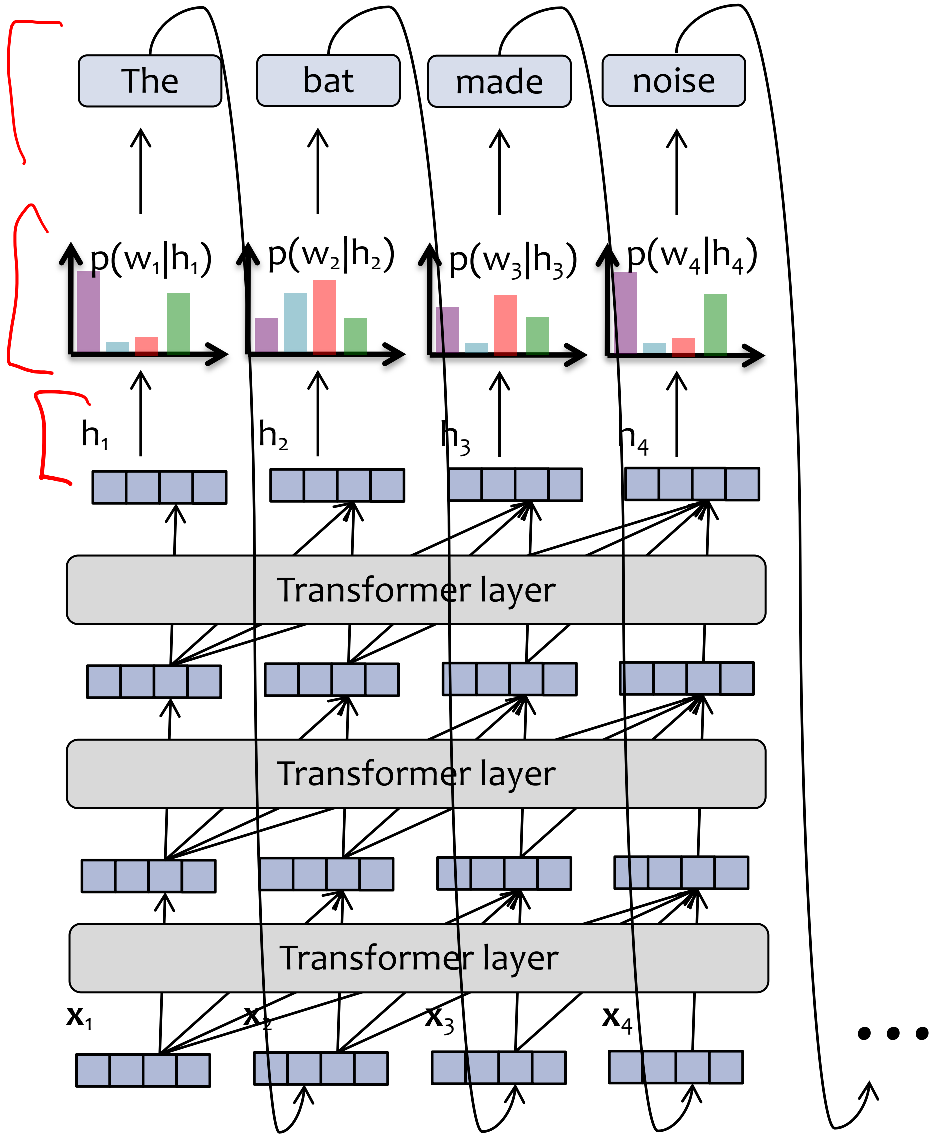
Transformer Layer

Each layer of a Transformer LM consists of several **sublayers:**

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections



Transformer Language Model



Each layer of a Transformer LM consists of several **sublayers:**

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of the **current and previous timesteps in the previous layer.**

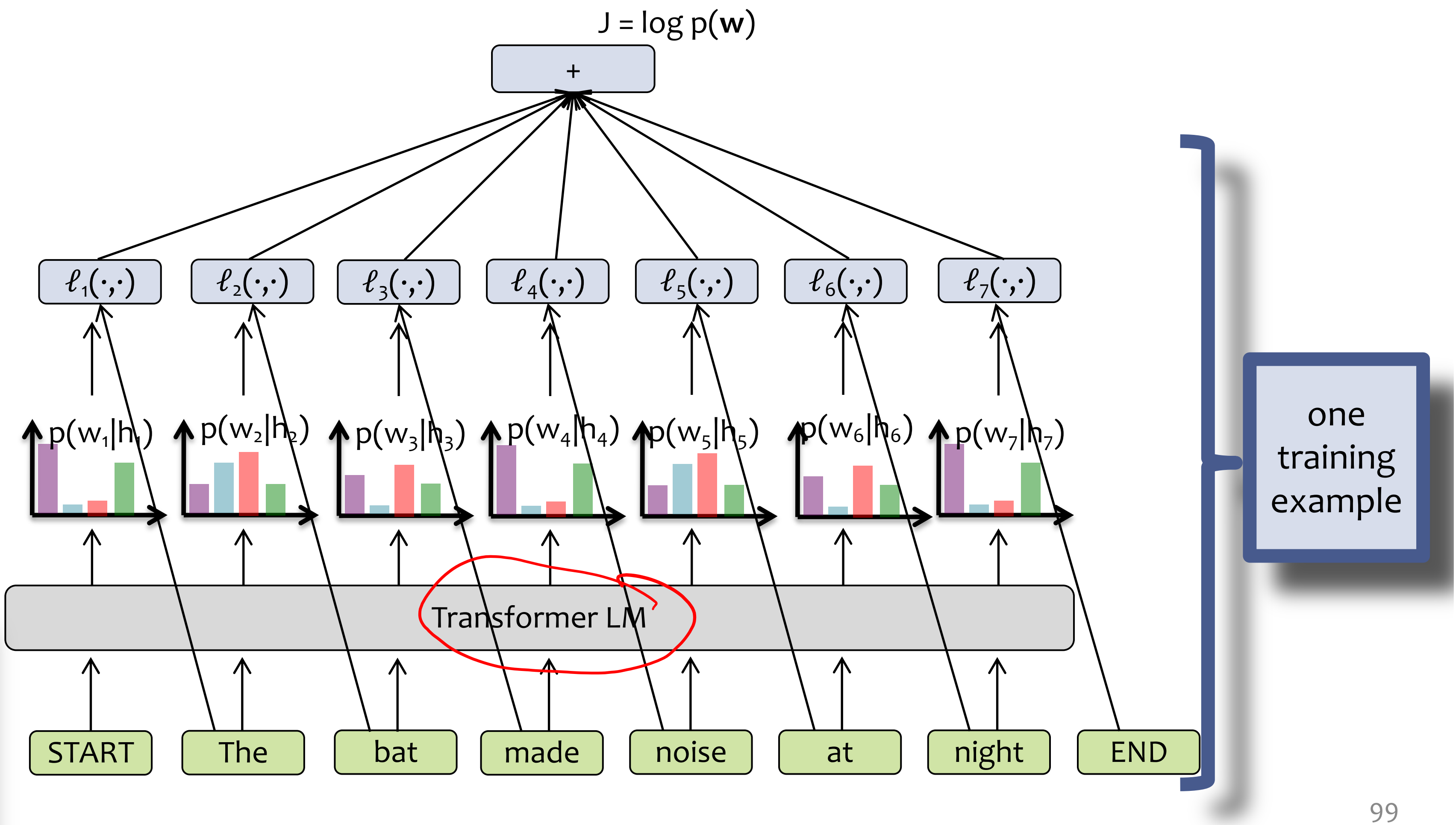
The language model part is just like an RNN-LM.

LEARNING A TRANSFORMER LM

Learning a Transformer LM

- Each training example is a sequence (e.g. sentence), so we have training data D = {$\mathbf{w}^{(1)}$, $\mathbf{w}^{(2)}$, … , $\mathbf{w}^{(N)}$}
- The objective function for a Deep LM (e.g. RNN-LM or Tranformer-LM) is typically the log-likelihood of the training examples:
$$J(\boldsymbol{\theta}) = \Sigma_i \log p_{\boldsymbol{\theta}}(\mathbf{w}^{(i)})$$
- We train by mini-batch SGD (or your favorite flavor of mini-batch SGD)

$$\log p(\mathbf{w}) = \log p(w_1, w_2, w_3, \ldots , w_T)$$
$$= \log p(w_1 \mid h_1) + \log p(w_2 \mid h_2) + \ldots + \log p(w_2 \mid h_T)$$



Training a Transformer-LM is the same, except we swap in a different deep language model.

GPT-3

- GPT stands for Generative Pre-trained Transformer
- GPT is just a Transformer LM, but with a huge number of parameters

d_{model}

Model	# layers	dimension of states	dimension of inner states	# attention heads	# params
GPT (2018)	12	768	3072	12	117M
GPT-2 (2019)	48	1600	--	--	1542M
GPT-3 (2020)	96	12288	4*12288	96	175000M