

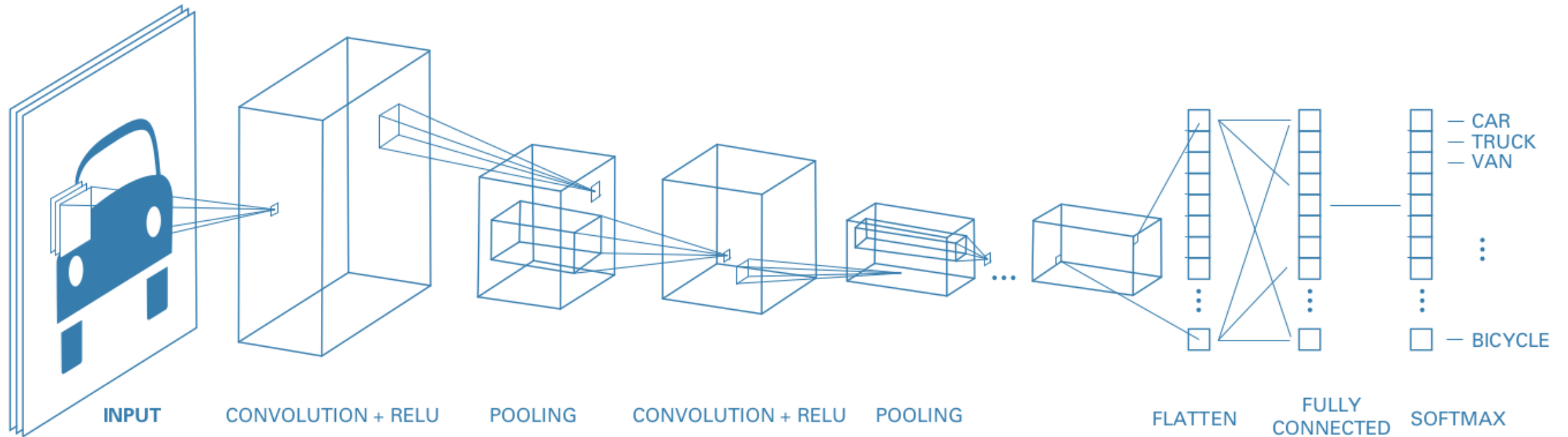
Deep Learning for Language: Recurrent Neural Networks, Attention

Robin Jia
USC CSCI 467, Spring 2023
February 28, 2023

Outline

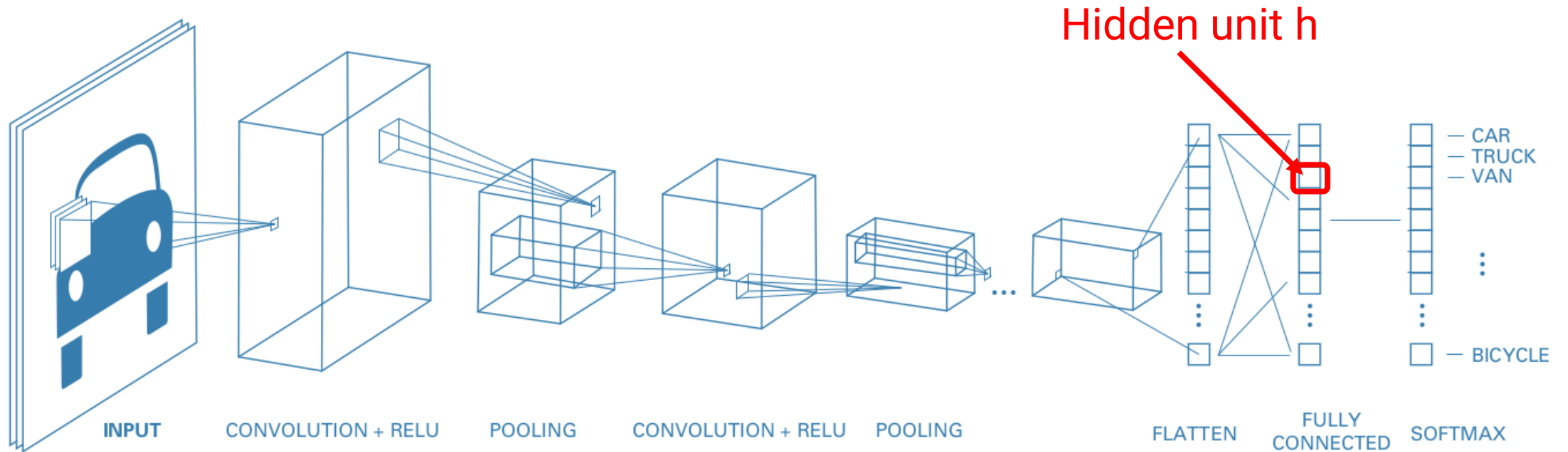
- Loose ends
 - How does backprop learn features?
 - Visualizing CNN features
- Recurrent Neural Networks for sequential data
- Sequence-to-sequence and Attention

How does backprop learn features?



- Every convolution & fully connected layer has (many) parameters
 - Convolutional: Kernel with $\#outChannels \times (\#inChannels \times K \times K + 1)$ params
 - Fully connected: $\#outDimensions \times (\#inDimensions + 1)$ params
- These all have to get learned by backprop + gradient descent on the loss

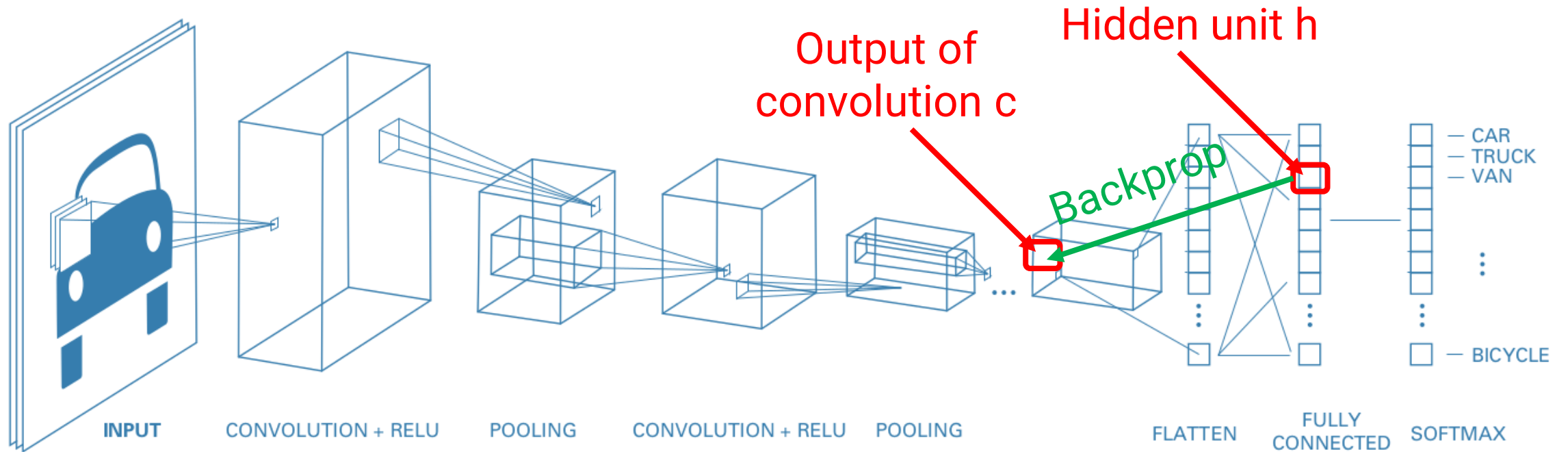
How does backprop learn features?



- Training example $(x^{(1)}, y^{(1)})$: $\partial(\text{Loss})/\partial(h) > 0$
 - Means that making h **smaller** leads to lower loss
- Training example $(x^{(2)}, y^{(2)})$: $\partial(\text{Loss})/\partial(h) < 0$
 - Means that making h **larger** leads to lower loss

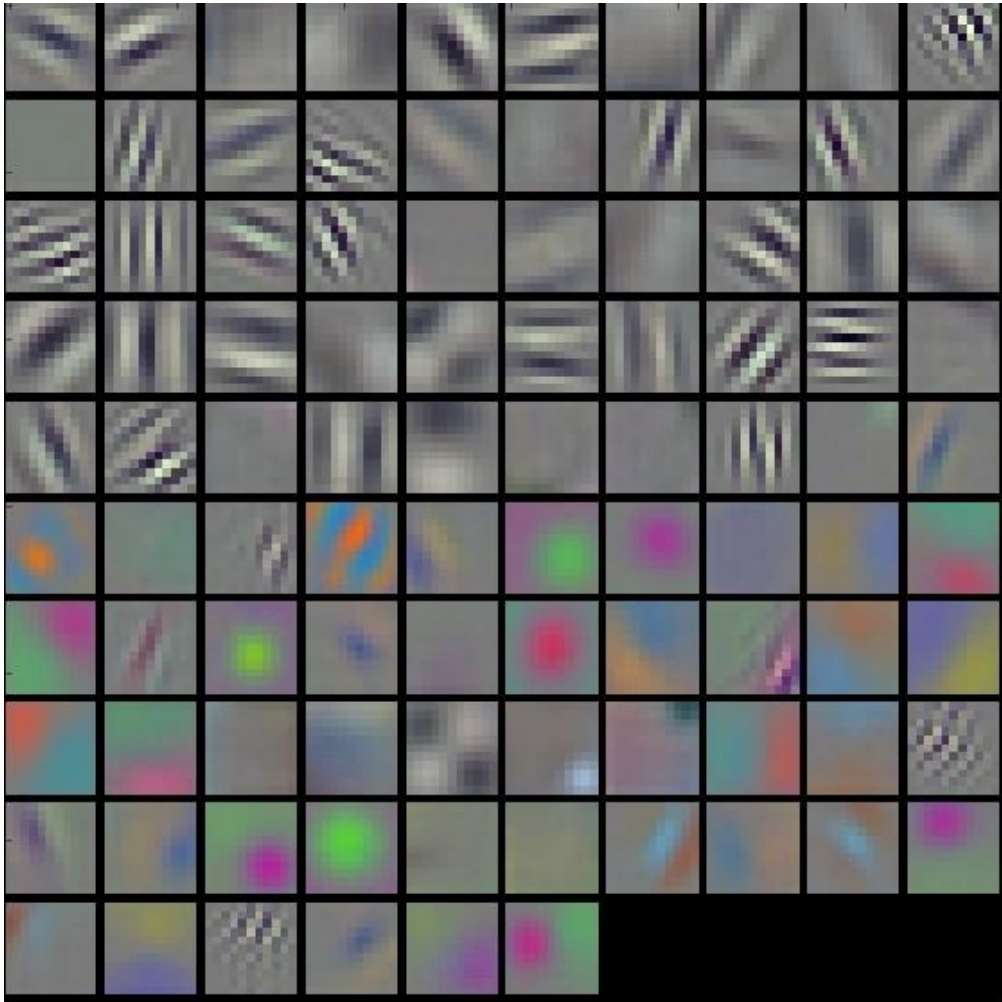
- h is output of “classifier”
- Gradient tunes classifier parameters to make output larger on some examples, smaller on others

How does backprop learn features?



- Backpropagation: Does making c bigger change h in good or bad way?
- Sum up these considerations over all hidden units that depend on c
- Train convolutional kernel parameters so that value of c leads to [values of h 's that lead to good outputs]
- And so on for earlier layers...

What features do CNNs learn?



- Kernels of AlexNet first layer
 - Each one is 3 (for RGB) x 11 x 11
- What is learned?
 - Edge detectors in different directions and widths
 - Patches of various colors

What features do CNNs learn?



Each Row: Images that activate a different neuron in 5th POOL layer of AlexNet

Outline

- Loose ends
 - How does backprop learn features?
 - Visualizing CNN features
- Recurrent Neural Networks for sequential data
- Sequence-to-sequence and Attention

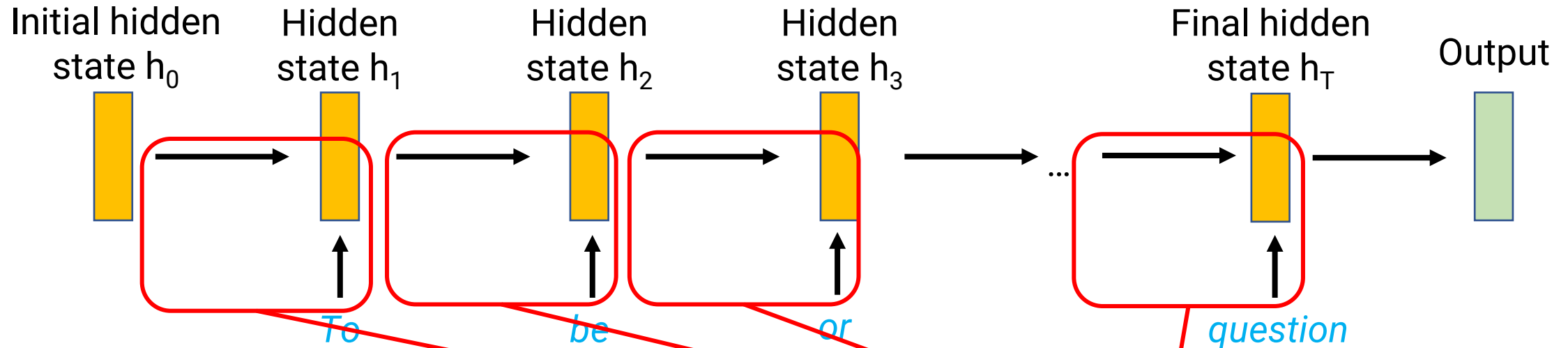
Note: Often there are many similar ways to achieve similar results
No one way of modeling is “correct”

I want you to remember the modeling ideas/concepts

Handling textual data

- Images: We assume inputs are fixed dimensional
 - Can crop/rescale as needed
- Text: Inputs are naturally variable-sized!
 - Example 1: *Amazing!*
 - Example 2: *There are many issues with this movie, such as...*
- Challenge: How can we use the **same** set of model parameters to handle inputs of any size?

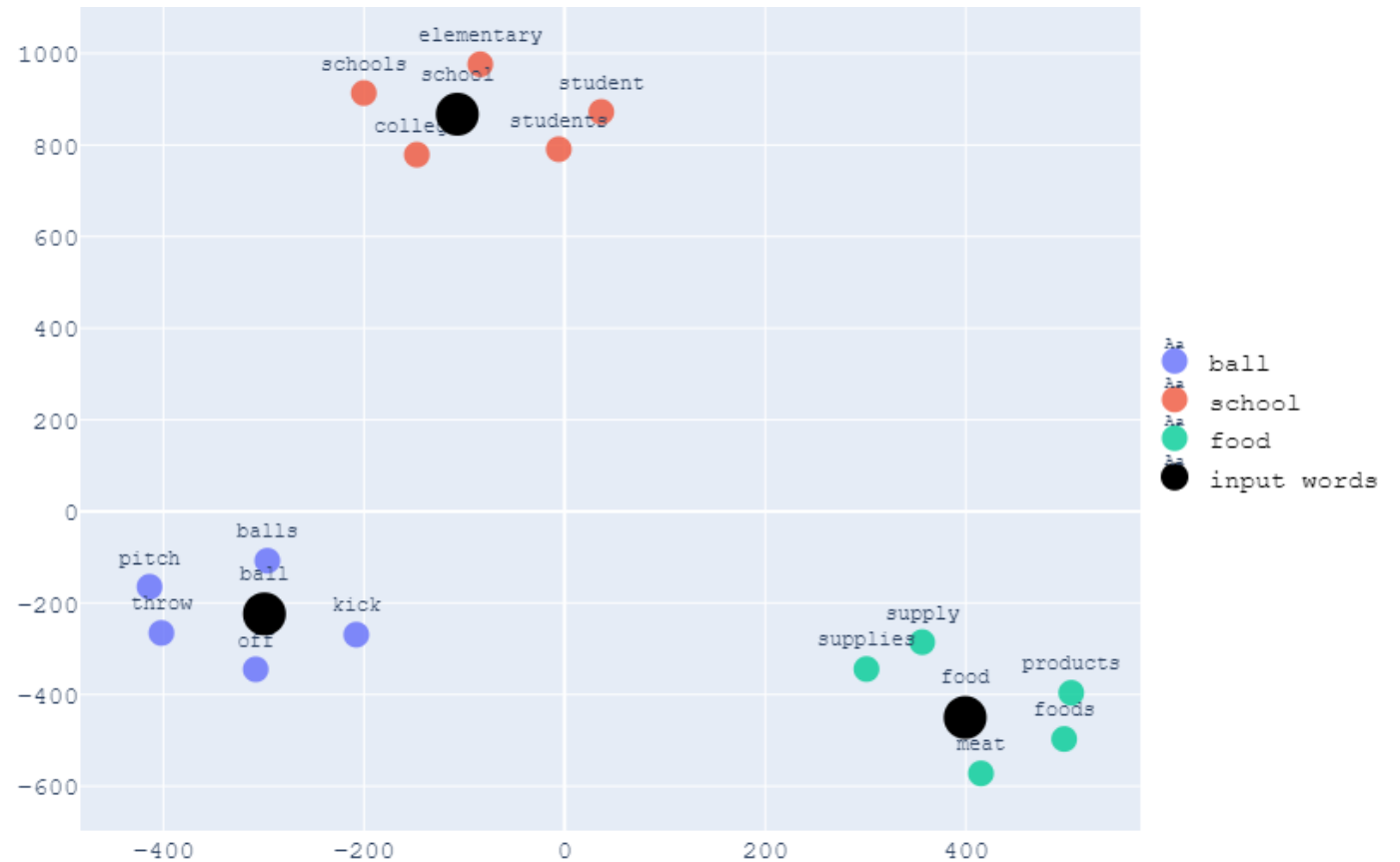
Recurrent Neural Networks (RNNs)



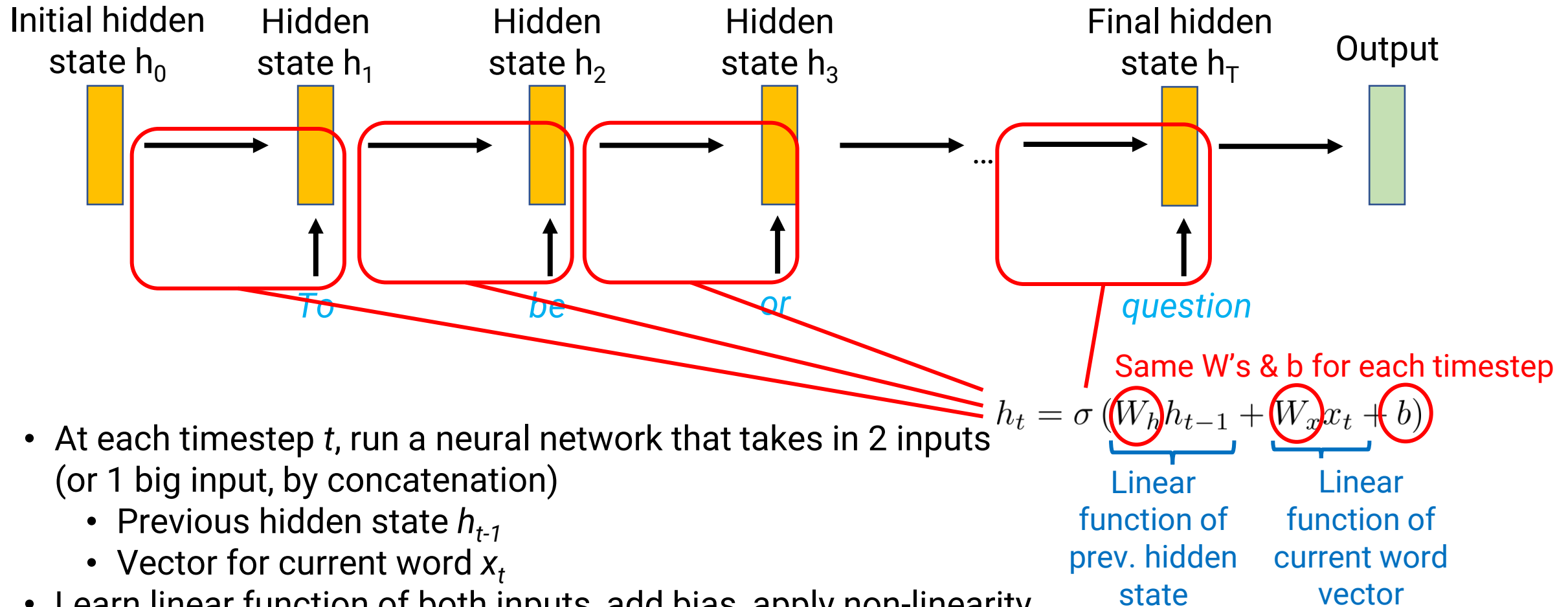
- Idea: Recurrence!
 - “Read” the input one word at a time
 - At each step, update the hidden state of the network
 - **Model parameters to do this update are same for each step**
- Each step is an application of the **same** neural network

Word Embeddings

- How do we “feed” the next word to the RNN?
- Want to learn a vector that represents each word
 - For each word w in vocabulary V , have vector v_w of size d
 - $|V| * d$ parameters needed
- Intuition: Similar words get similar vectors
 - More on learning word vectors later in the class!

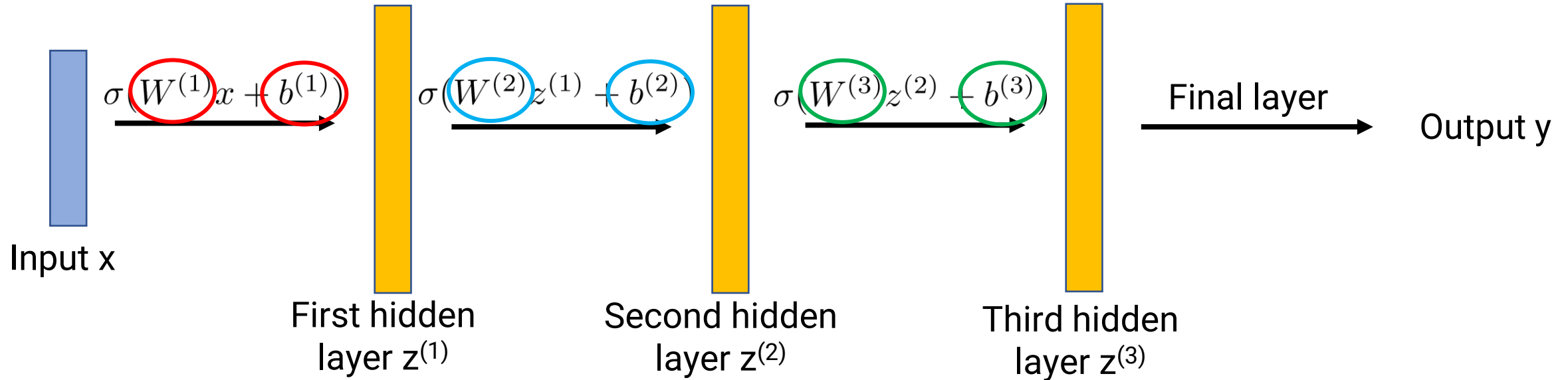


One RNN variant



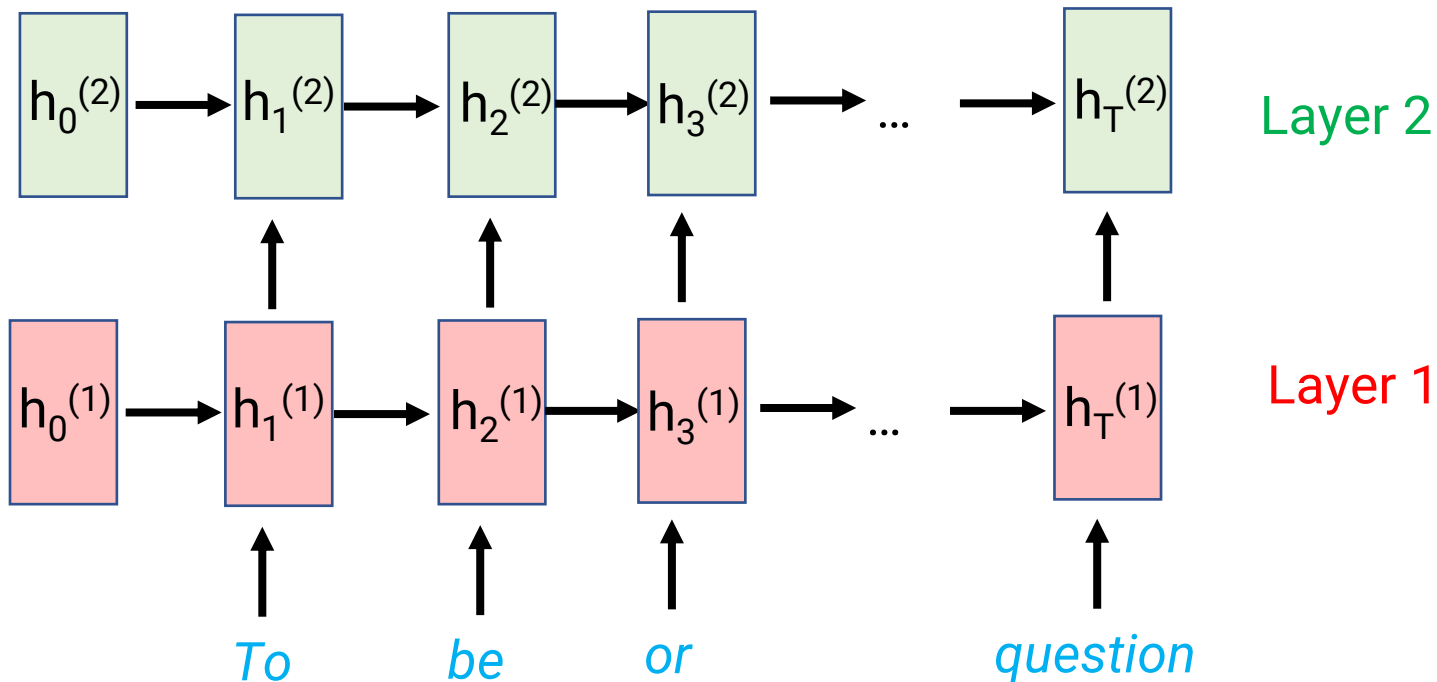
- At each timestep t , run a neural network that takes in 2 inputs (or 1 big input, by concatenation)
 - Previous hidden state h_{t-1}
 - Vector for current word x_t
- Learn linear function of both inputs, add bias, apply non-linearity
- Parameters: Recurrence params (W_h, W_x, b), initial hidden state h_0 , word vectors

Recurrence vs. Depth



- Deep networks (i.e., adding more layers)
 - Computation graph becomes longer
 - Number of parameters also grows; each step uses new parameters
- Recurrent neural networks
 - Computation graph becomes longer
 - Number of parameters **fixed**; each step uses **same parameters**

Recurrence and Depth



- You can have multiple layers of recurrence too!
 - Left-to-right axis (“time dimension”): Length is size of input, same parameters in each step
 - Top-to-bottom axis (“depth dimension”): Length is depth of network, different parameters in each row

Announcements

- HW2 due this Thursday
 - Pytorch not reproducible across different hardware
 - Still used in assignment as it is very widely used for deep learning
 - Ultimately we will grade by reading your code, not by checking if your numbers in the write-up are “correct”
- Proposals should be returned with feedback by Thursday
- Tuesday, March 7: Discussion of Midterm Report due March 23
- Section canceled March 10
 - We will stop doing HW review sections, as they seem less popular
 - Please still come to OH if you want clarifications on old HW problems

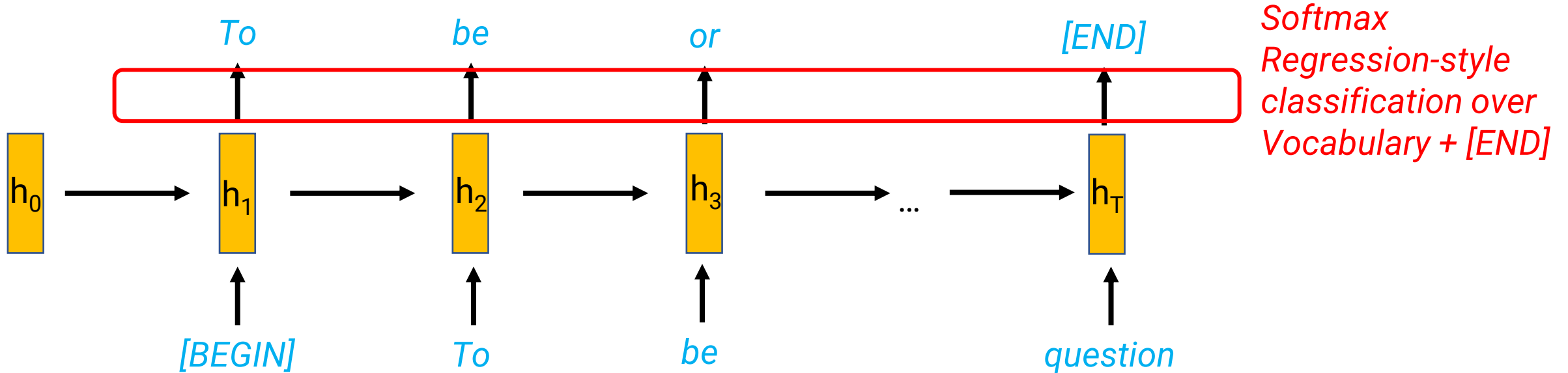
Outline

- Loose ends
 - How does backprop learn features?
 - Visualizing CNN features (cat neuron?)
- Recurrent Neural Networks for sequential data
- **Sequence-to-sequence and Attention**

How to use RNNs?

- Language modeling/text generation (“Decoder only”)
- Text classification (“Encoder only”)
- Sequence-to-sequence (“Encoder-decoder”)

Language Modeling (“Decoder only”)



- At each step, predict the next word given current hidden state
 - Essentially a softmax regression “head”—takes in hidden state, outputs distribution over Vocabulary + [END]
- Start with special $[BEGIN]$ token (so the first word model generates is first real word)
- One step’s output becomes next step’s input (“autoregressive”)
- To mark end of sequence, model should predict the $[END]$ token
- Called a “Decoder” because it looks at the hidden state and “decodes” the next word

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

The keys to the cabinet ___ (on the table)
plural singular

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

The *keys* to the cabinet *are* (on the table)
plural singular

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

*The **keys** to the cabinet by the door **are** (on the table)*

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

*The **keys** to the cabinet by the door on the left **are** (on the table)*

Advanced RNNs

- “Gated” RNNs (GRUs, LSTMs)
 - Better at holding on to long-range state
 - These are usually preferable to the RNN variant I showed today
 - They work the same way, but the recurrence relationship between previous hidden state and next hidden state is more complicated...

What do RNNs learn?

t t p : / / w w w . y n e t n e w s . c o m /] E n g l i s h - l a n g u a g e w e b s i t e o f I s r a e l ' s l a r
 t p : / / w w w b a c a h e t s . c o m / - x g l i s h l i n g u a g e s a i r s i t e o f t s l a e l i s s i n g
 d : x n e . w a e a . . a w a t o a . s & n t i a c a - s a r d e e l h o a n t b i s a n f a n r e i f ' a a t d
 m w - 2 p i i i s o e s s i s . / e r n . c] (d c e e n e p e s a a i k i i e e l e d h , i r t h r a o n s e , c o s e
 d r . < : a h b - n p t w t . x i g h / m a) T v d r y z i c o u e d l s u : t h a - o o t u , s t u i f l v e p e r y
 s t p , t c o a 2 d r u l w o c l e n s r] p . l l v a o d , , e y t c - n d m - o i b u v s] b b i m s u l t a t l y b n

g e s t n e w s p a p e r ' ' [[Y e d i o t h A h r o n o t h]] ' ' ' ' H e b r e w - l a n g u a g e p e r i o d
 e t a a w s p a p e r s o ' [[T e l t i (f e a n e m t i) ' ' * ' ' [e r r e w s l e n g u a g e : a r o s o d i
 i r s c o e e n a i T T h A o a i n n h S r m u w] e y s [' i n e i a ' s i w d d e ' h s o l r i f r :
 u s . . s e t l g o r s . a s a t C a r e e g ' a C l r i s z] i e ' : : , # : T A a a a a t B a s e e i l o ' i a n f v l
 - t u a e v r t i d , t B A m S u s y u t]] A s a o i g s]] , . : s M B o l o u s : T o u a - n : d w o a p n u
 a , d , i i u i t i c p .] (l S v H v t u s u i e D n o e g a n o . ,] : { C C u i b o h e C y b k s l s : r - e p c n t s

i c a l s : ' ' ' * ' ' [[G l o b e s]] ' ' [h t t p : / / w w w . g l o b e s . c o . i l /] b u s i n e s s d a
 c a l : ' ' ' * ' ' [T a a b a] ' ' ([t t p : / / w w w . b u o b a l . c o m u n / s A - y t i n e s s a e t
 s t l ' [h A e o v e l t s a h a d : x g e . w a o i r . r t o a . e l . i T & a i e g e o o y
 t t ' ' ' & [& & m C o e r o n e ' : : , i ' o d w . , : n i i i s a a u e . e n i / o m l c C . (e f t g i r i i u
 a ' n : , C : & : # * : a f D r u s u] l , . o m e l p < , d h a ; d e u o o t / i h n c s i f S , u r h o s t , t u n
 n k i <] : & 1 1 s T G u i t r s i , : b a c m r - x t p o b - g r e s i s l e r l n a f a D] l o s p t a d , i f r m

i l y * ' ' [[H a a r e t z | H a ' A r e t z]] ' ' [h t t p : / / w w w . h a a r e t z . c o . i l /] R e l a t i v
 l y * ' ' [[T e r r d n F e r a n t a h]] ' ' ([t t p : / / w w w . b o n m d s t . c o m u n / s - e s a t e o i
 r e ' ' h A i l n n t t e H a l s r c n o l ' s a h a d : x n e . w a a m r t d h e o h . o l . c & o p i n i v e
 k i . : * s C O S a n l t h i T i m ' l i] e : , i m c d w - 2 p h i i s e r d i t . i n a / c m f i . (a f l c a n a
 d s - ! [t B T C o m m g d]] W o n a a e , : . b a e r r . < t a i b - d u l c n n c / a r n e s i] l i c e y s t o
 n d s # & : G l D u v c c s a o S u c l t e l] z | , : o ' o m t] , : e o a 2 n i v f s r o o e i u n a l a) u v v r o

- Here: a character-level model (not word-level)
- Blue/Green: Low/high values of 1 neuron
- Below: Top-5 predictions for next character
- This neuron seems to detect whether we're inside a URL

What do RNNs learn?

* * * [[Jerusalem Report]] * * * [http://www.jrep.com/] Left-of-center En
 * * * [hToausal m aogurt]] * * * (http://www.bsinioom/ -iat af t enter (ng
 [' [Cassmene] Beaonds s a [ad : xne. waaaaoca. s &ato--nfhhlsu m-ouc
 ' s mFurnl s iaetal lsa ' : : , i ' cdw-2tpi i isoeg. er/. a) (oseswr -ci ddrs [mt
 : * : AqDenebi ut n | Ci pre e | , . b1emr. 9: ahb- npumugh nmp) Teir et u: eoseodsald
 #T&Tf Si wr p e]] aluvel ru, s : - mprt s < moa2deys hi lr] c. Augl , 1p, larc : fae

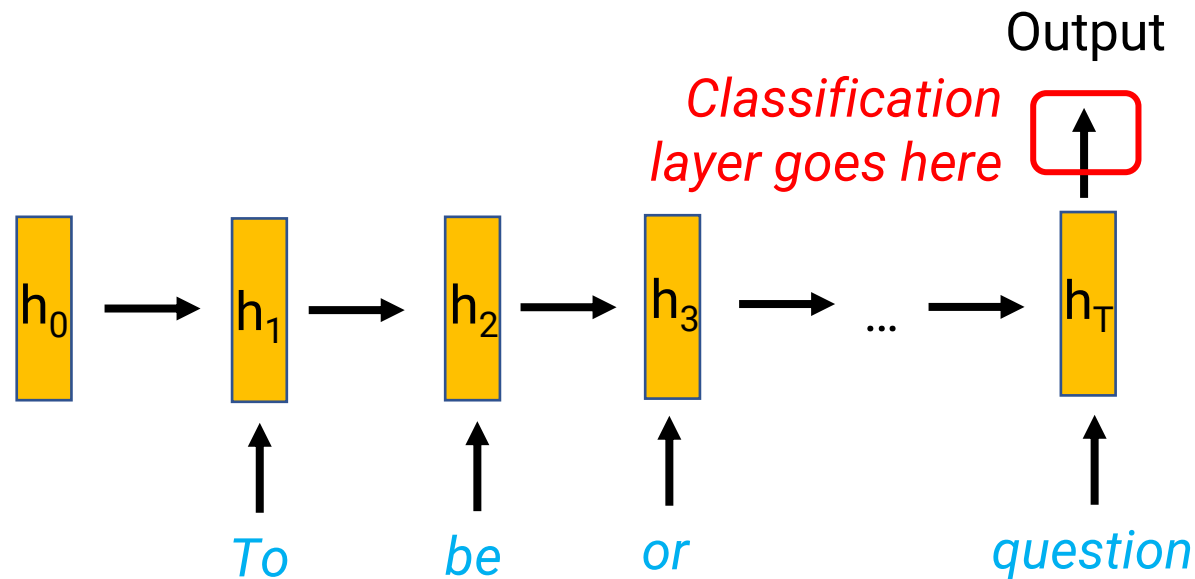
g l i s h [[weekly newspaper]] * * * [[YNet News]] * * * [http://www.ynetnews.c
 l i s h c [Caakly] cawspaper]] * * * [hTaA at]] * * * (http://www.bacahets.co
 i aci - l hSoip] i sec] enp] s . ' ' [Co *wess]] s a [ad : xne. waea. . awat oa
 een a , pCci et ned lox] gi ci | | s ' [sAmFeSahon] t ' : : , i momw-2 pi i i soessi s. / er
 syz . sf penn a | ruel | rra . ' # * : oDuFrei uep , : b1edr. < : ahb- npt wt . xi gh
 a dpe amAr bde or pi tee] dt s - | T { [BaAvTpoSwa o , . . oacstp , tcoa2drulwoclens

om/] English-language website of israel's largest newspaper ' ' [[Yed
 m/] - xgl ish languages a i r s i t e o f t s l a e l i s s i n g e t a a w s p a p e r s o ' [[Tel
 . s & n t i a c a - s a r d e e l h o a n t b i s a n f a n r e i f ' a a t d i r s c o e e n a i T T h A o a i
 n . c] (d c e e n e p e s a a i k i i e e l e d h , i r t h r a o n s e , c o s e u s . . s e t l g o r s . a s a t C a r e
 / m a) T v d r y z i c o u e d l s u : t h a - o o t u , s t u i f l v e p e r y - t u a e v r t i d , t B A m S u s y
 r] p . l l v a o d , , e y t c - n d m - o i b u v s] b b i m s u l t a t l y b n a , d , i i u i t i c p .] (l S v H v t u

i o t h A h r o n o t h]] * * * * Hebrew-language periodicals : ' ' * * * [[Globes]] *
 t i (f e a n e m t i]] * * * [e r r e w s l e n g u a g e : a r o s o d i c a l : ' ' * * * [T a a b a]] * *
 n n h S r m u w] e y] s [' i n e i a ' s i w d d e ' h s o l r i f r : s t l ']] [h A e o v e l t]] s
 e g ' a C l r i s z] i e ' : : , # : T A a a a a t B a s e e i l o ' i a n f v l t t ' ']] & [& m C o e r o n e ' : :
 u t]] A s a o i g s]] , . : s M B o l o u s : T o u a - n : d w o a p n u a ' n : , C : & : # * : a f D r u s u] l ,
 s u i e D n o e g a n o . ,] : { C C u i b o h e C y b k s l s : r - e p c n t s n k i <] : & 1 1 s T G u i t r s i ,

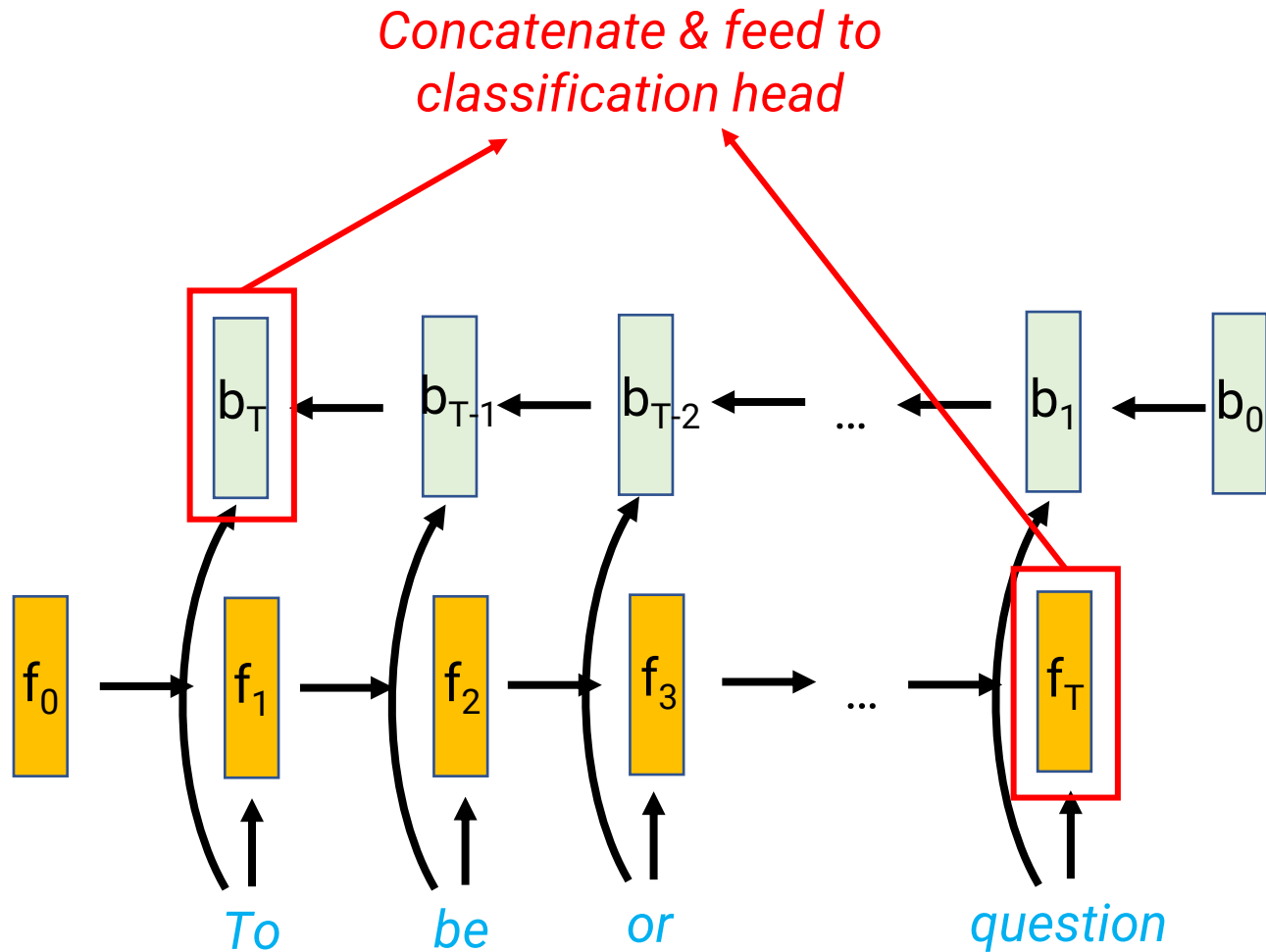
- Here: a character-level model (not word-level)
- Blue/Green: Low/high values of 1 neuron
- Below: Top-5 predictions for next character
- This neuron fires only inside a Markdown `[[link]]` (so it knows when to close the square brackets?)

Text classification (“Encoder only”)



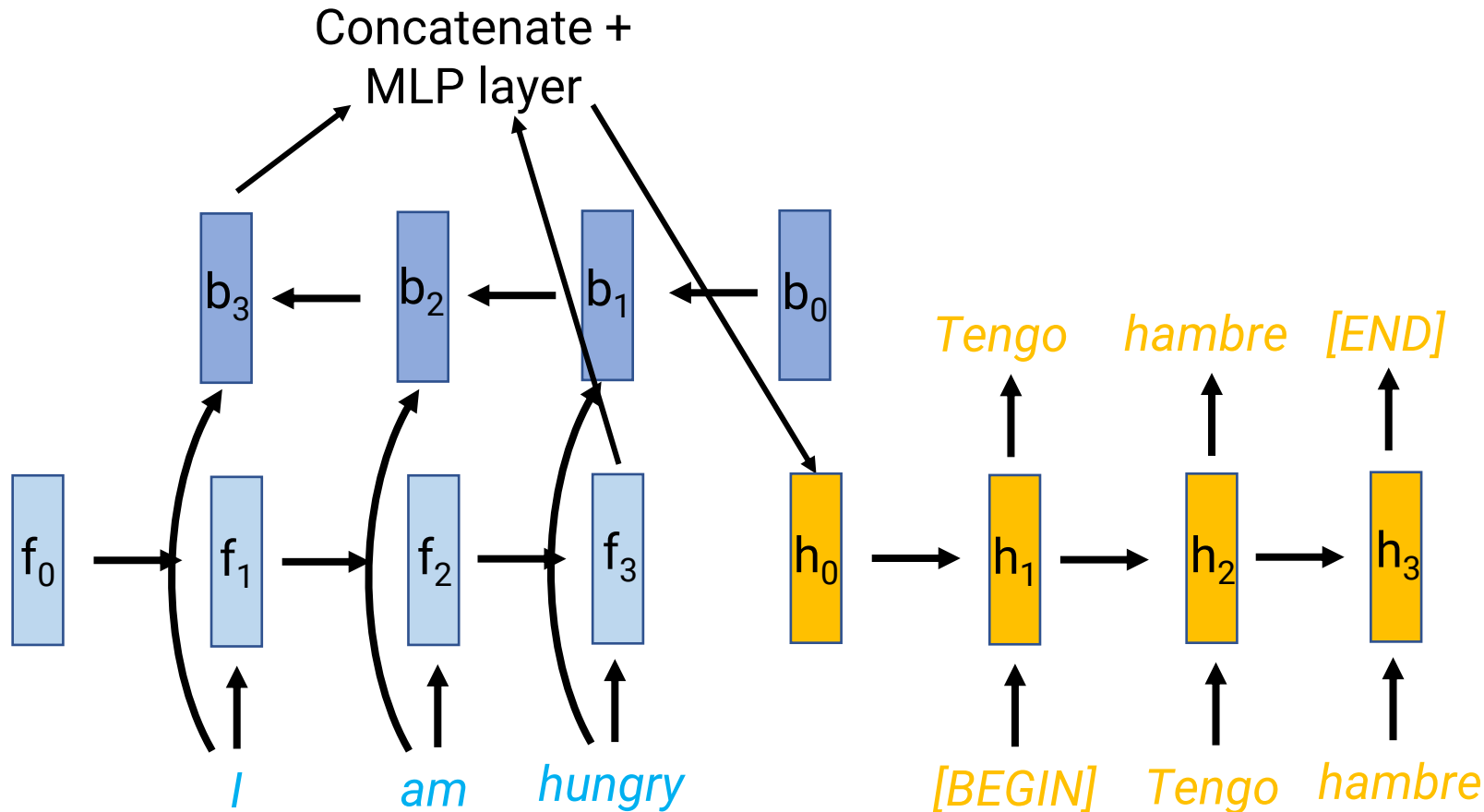
- First run an RNN over text
- Use the final hidden state as an “encoding” of the entire sequence
- Use this as features, train a classifier on top
- Downside: Later words processed better than early words (long range dependency issues)

Bi-directional encoders



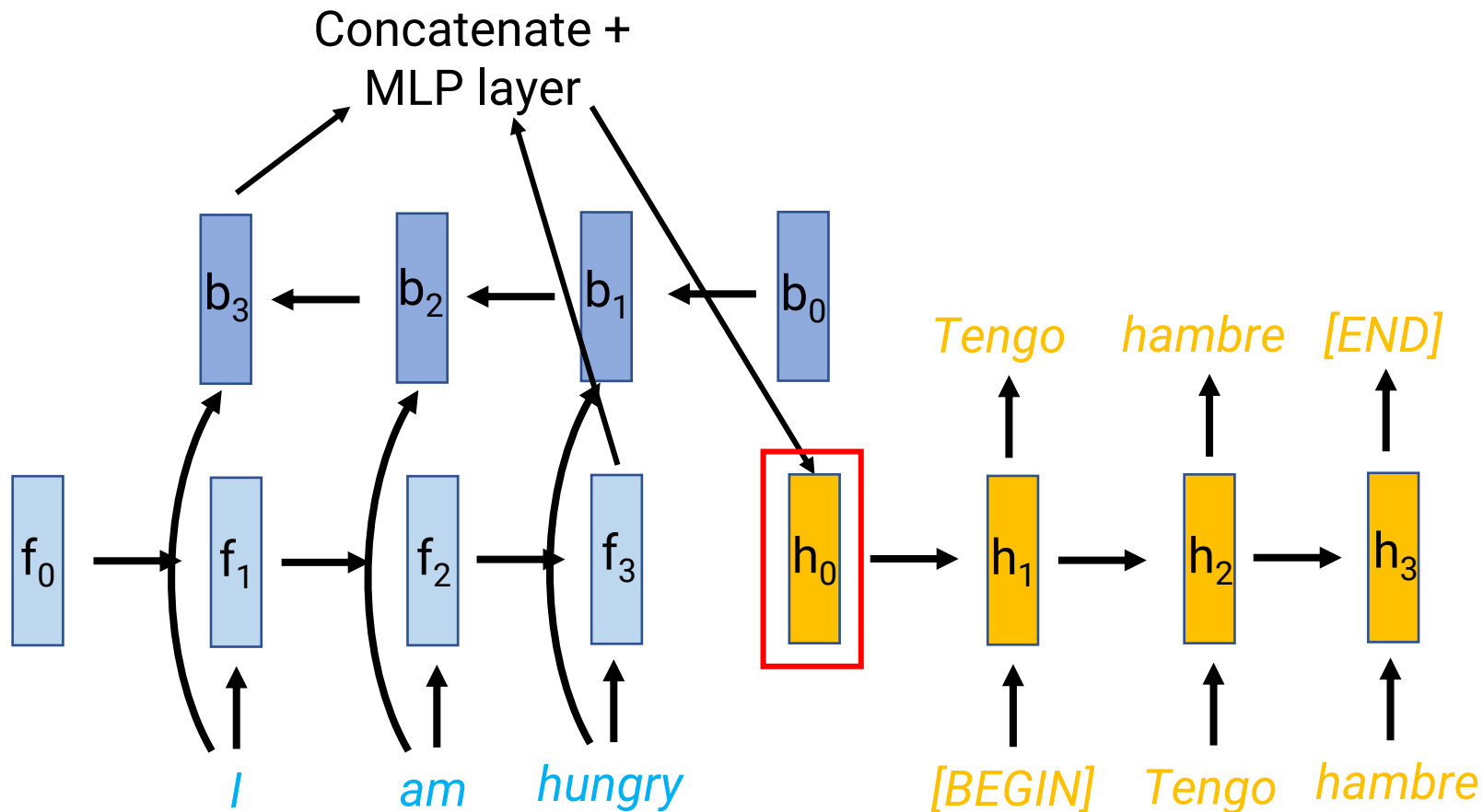
- Run one RNN left-to-right, and another one right-to-left
 - (I'll call forward-direction hidden states f_t , backward-direction hidden states b_t)
- Concatenate the 2 final hidden states as final representation
 - Note: This encoding is twice as large now—we've doubled the number of features passed to the final classifier

Sequence-to-sequence (“Encoder-decoder”)



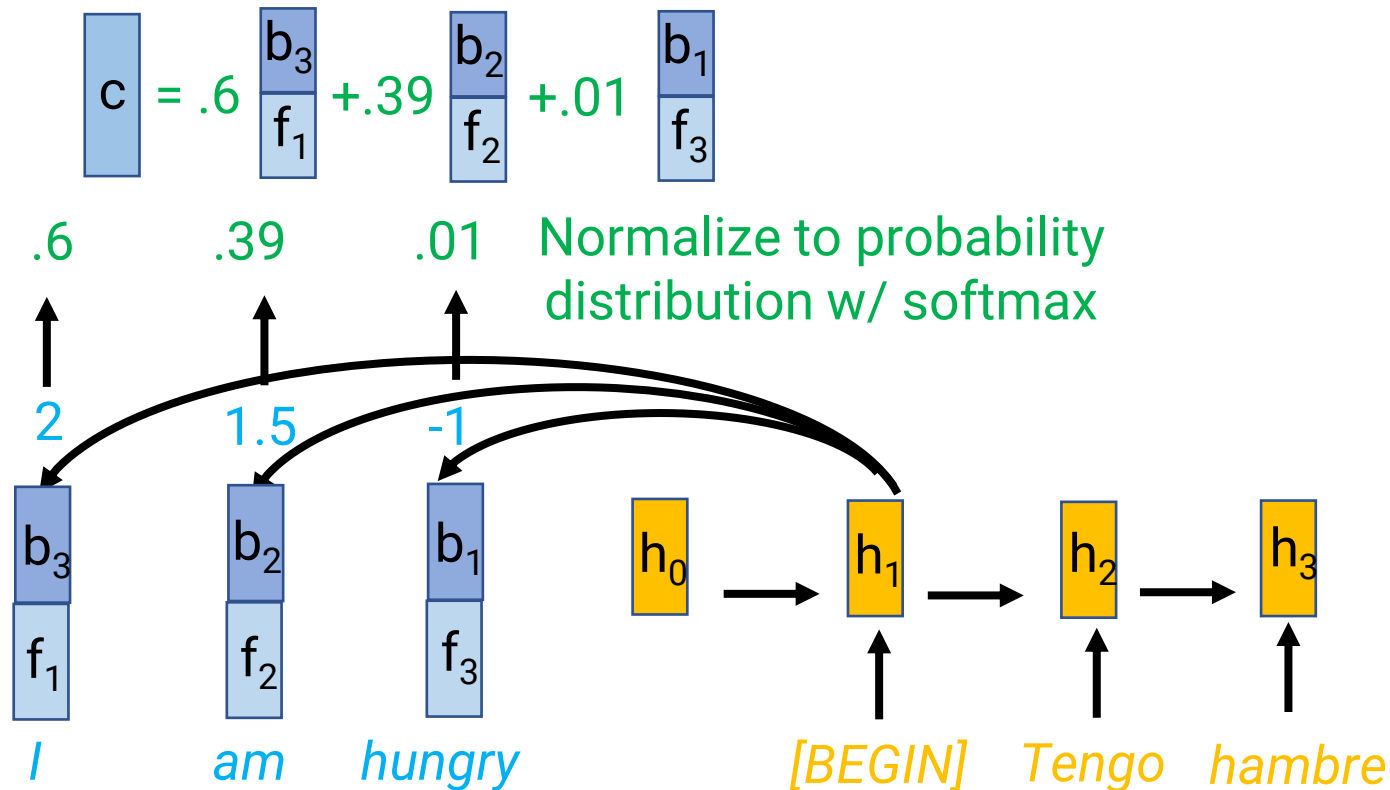
- Example: Machine Translation
 - Input = English text
 - Output = Spanish text
- Encoder: Process English sentence into vector
 - E.g. Bidirectional encoder + MLP layer to generate decoder's initial state
- Decoder: Use vector as initial hidden state and start doing language modeling in Spanish
- Vector space acts as a “shared language”

What's missing? Alignment



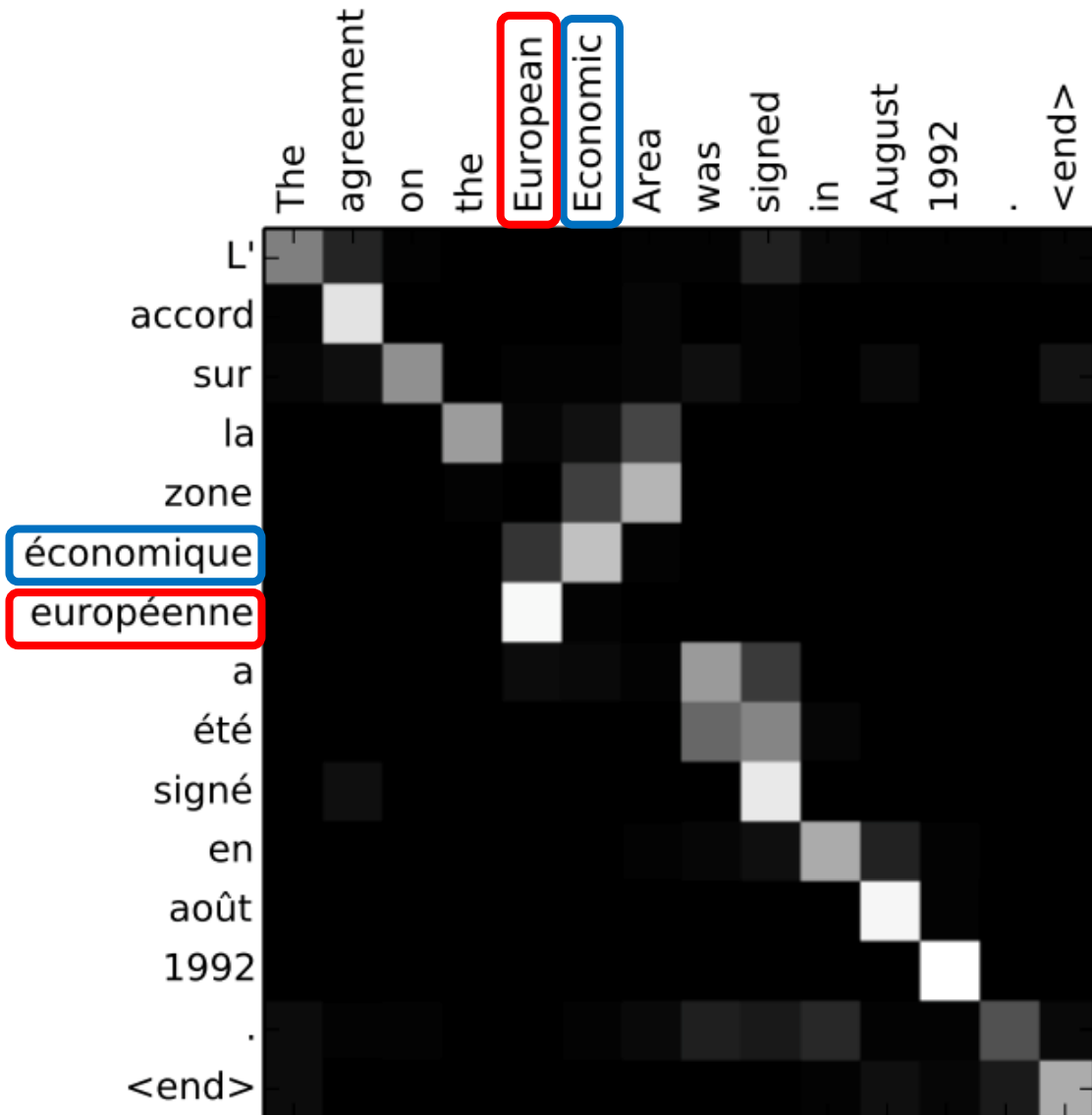
- Challenge: The single encoder output has to store information about the entire sentence in a single vector
- Would be much easier if we can "refer to our notes"
- Traditional MT: Alignment between input & output sentences
- Can we get a neural network to model alignments?

Attention



- Compute similarity between decoder hidden state and each encoder hidden state
 - E.g., dot product, if same size
- Normalize similarities to probability distribution with softmax
- “Context” vector c = weighted average of encoder states based on the probabilities
 - No new parameters (like ReLU/max pool)
 - Use c when computing decoder outputs or transitions
- Intuition
 - Step 1: Find similar input words
 - Step 2: Grab the encoder representation of those words
 - Step 3: Tell the decoder that this is relevant

Visualizing attention



- Source is English, Target is French
- Each row is a probability distribution over the English text
- Alignment makes sense, overcomes word order differences
 - When generating “**économique**” attend to “**Economic**”
 - When generating “**européenne**” attend to “**European**”

Conclusion

- Deep Learning for Language must deal with possibly long inputs
- RNNs handle arbitrarily long inputs with fixed number of parameters
- Challenges
 - Long range dependencies
 - Modeling alignments between input and output sequences
- Next time: Can Attention solve everything?