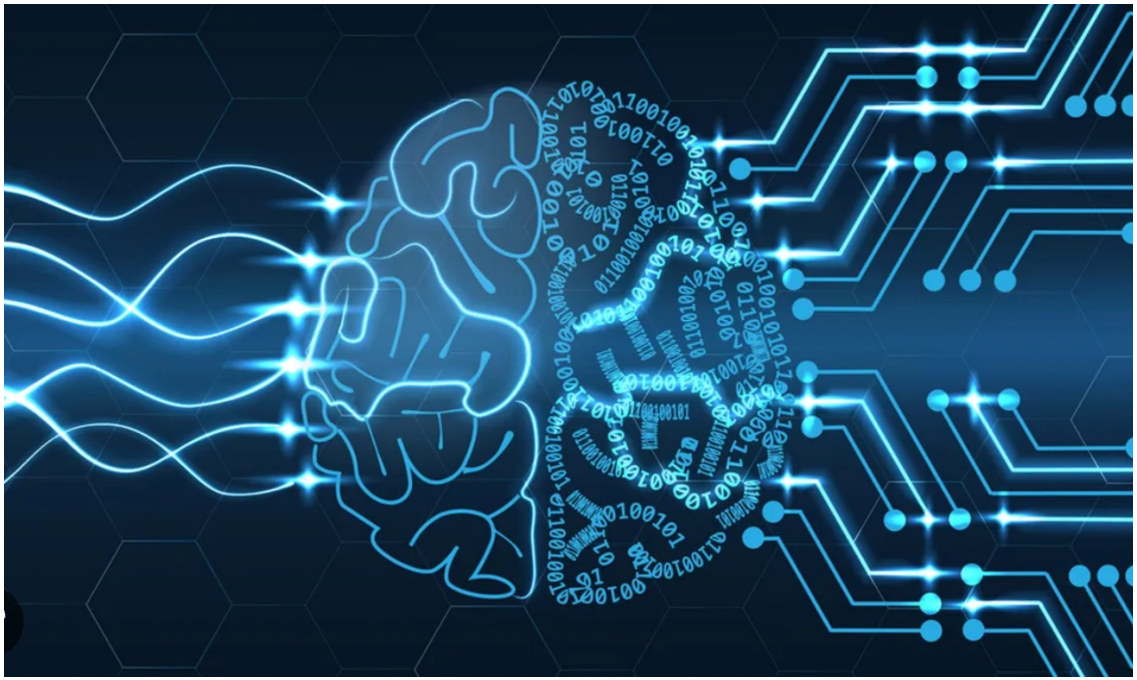
CS 4100: Introduction to AI

Wayne Snyder
Northeastern University

Lecture 23 – Deep Learning – RNNs, Encoder-Decoders, Transformers



Plan for this week

Plan for this Lecture:

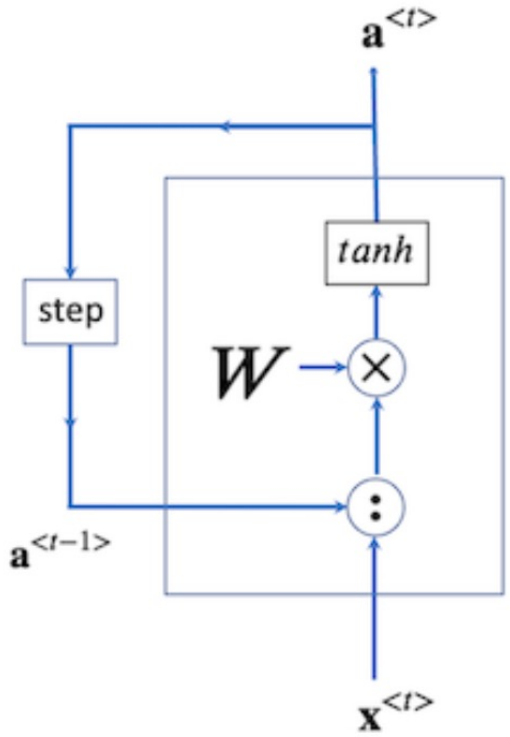
- RNNs, GRNNs, GRUs, LSTMs
- Building Language Models with RNNs
- Encoder-Decoders
- Adding Atttention

- Next Lecture: Optional! On Transformers and Large Language Models (BERT, GPT, etc.)

- Last lecture: The possibility of general machine intelligence; machine consciousness

Recurrent Neural Networks: RNNs, GRUs, LSTMs

Basic RNN layer: Recurrent data path serves as a memory between time steps for sequence data:



$$\mathbf{a}^{<0>} = \mathbf{0}$$

```
for t = 1,2,... ,T:
```

$$\mathbf{a}^{<t>} = tanh(W(\mathbf{a}^{<t-1>} : \mathbf{x}^{<t>}))$$

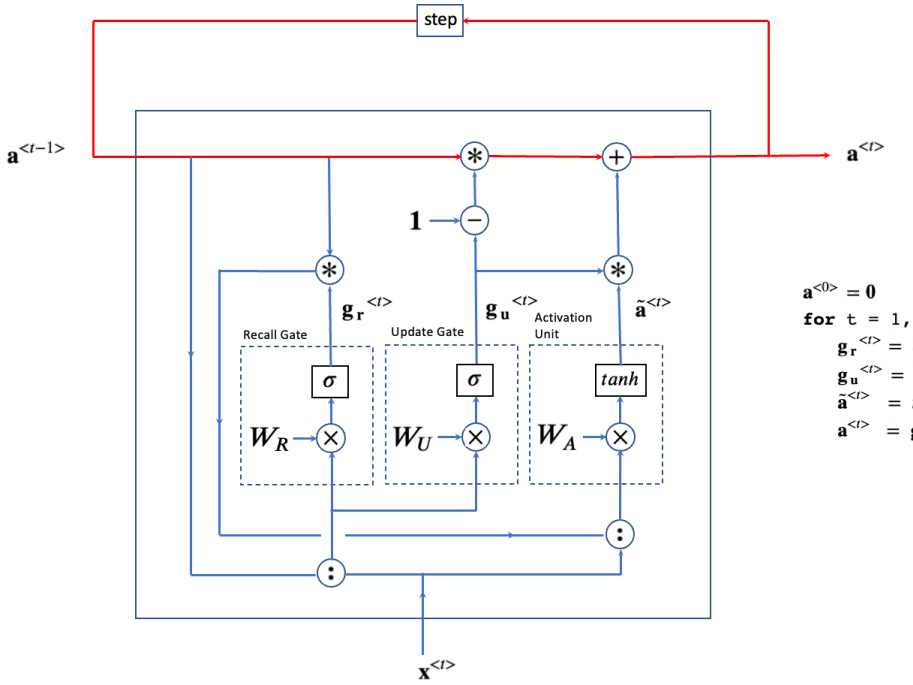
$$\mathbf{x}^{<1>}, \mathbf{x}^{<2>}, \ldots, \mathbf{x}^{<T>},$$

$$W = \begin{bmatrix} \mathbf{w}'_1 & \mathbf{w}_1 \\ \mathbf{w}'_2 & \mathbf{w}_2 \\ \vdots & \\ \mathbf{w}'_n & \mathbf{w}_n \end{bmatrix} = \begin{bmatrix} w'_{1,1} & w'_{1,2} & \cdots & w'_{1,n} & w_{1,1} & w_{1,2} & \cdots & w_{1,k} \\ w'_{2,1} & w'_{2,2} & \cdots & w'_{2,n} & w_{2,1} & w_{2,2} & \cdots & w_{2,k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w'_{n,1} & w'_{n,2} & \cdots & w'_{n,n} & w_{n,1} & w_{n,2} & \cdots & w_{n,k} \end{bmatrix}$$

Recurrent Neural Networks: RNNs, GRUs, LSTMs

Gated Recurrence Unit layers add individual units to decide

- The activation (output) signal;
- How the memory is used in creating the current activation;
- What to remember from the current time step:



$a^{<0>} = 0$

for t = 1,2,... ,T:

$g_r^{<t>} = \sigma(W_R(a^{<t-1>}:x^{<t>}))$

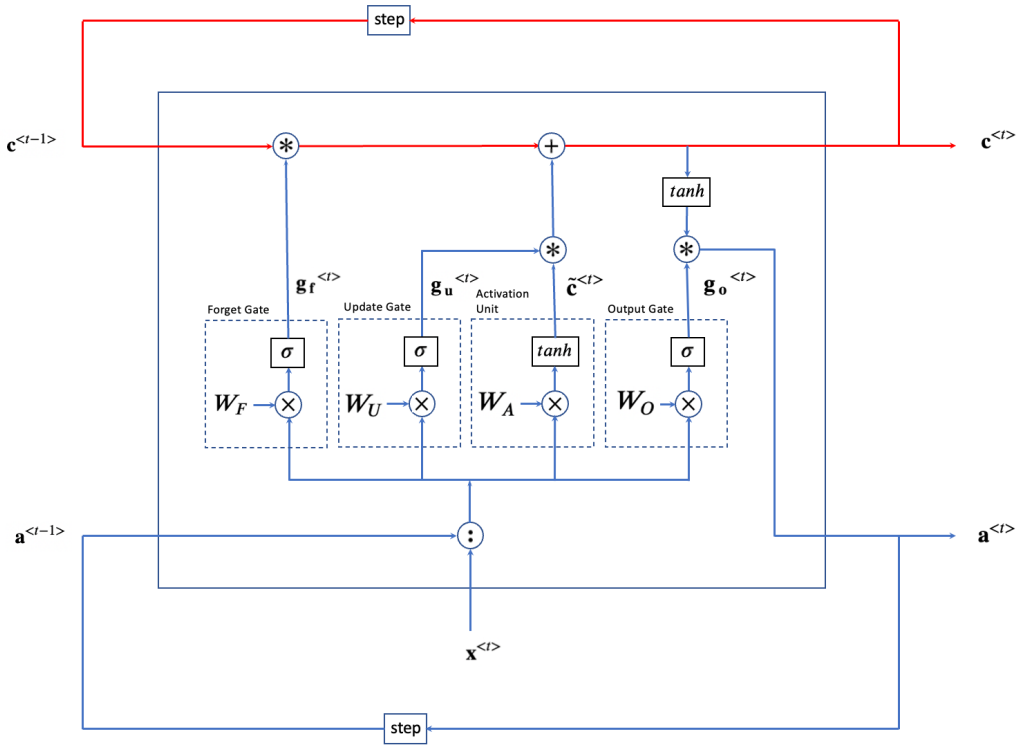
$g_u^{<t>} = \sigma(W_U(a^{<t-1>}:x^{<t>}))$

$\tilde{a}^{<t>} = tanh(W_A((g_r^{<t>} * a^{<t-1>}):x^{<t>}))$

$a^{<t>} = g_u^{<t>} * \tilde{a}^{<t>} + (1 - g_u^{<t>}) * a^{<t-1>}$

Recurrent Neural Networks: RNNs, GRUs, LSTMs

Long Short Term Memory layers use a separate "carry" path for the memory, 4 gates, and calculate the activation from the memory and the current inputs:

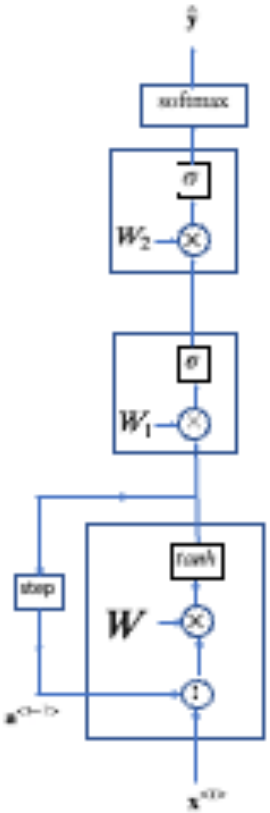


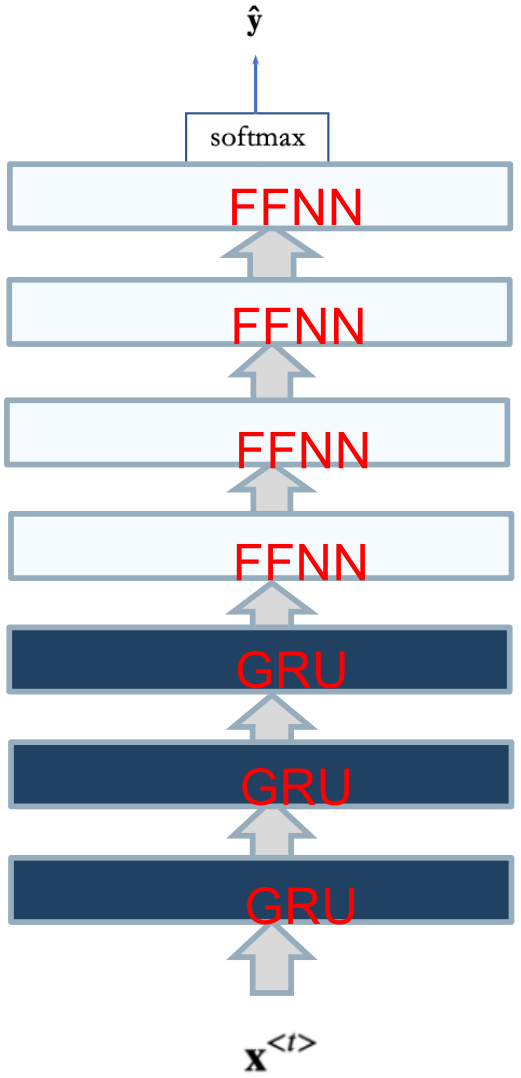
Recurrent Neural Networks: RNNs, GRUs, LSTMs

How are Networks with Recurrent Layers Designed?

Deep Networks

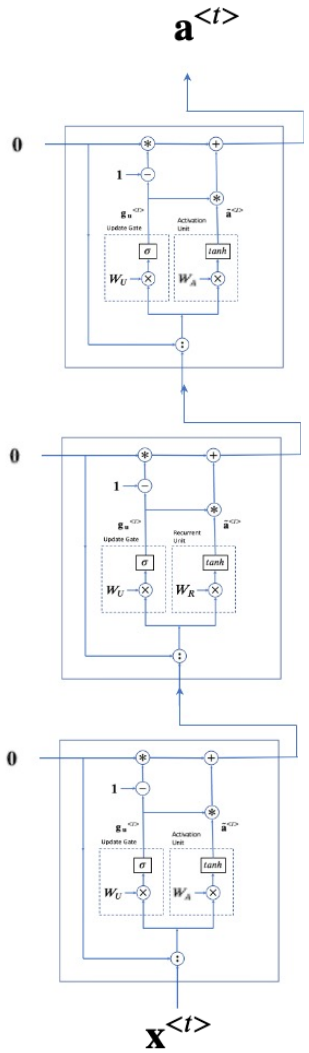
Generally, networks for sequence data such as text have recurrent layers processing the sequence, and feed forward layers interpreting and producing output such as a classification.





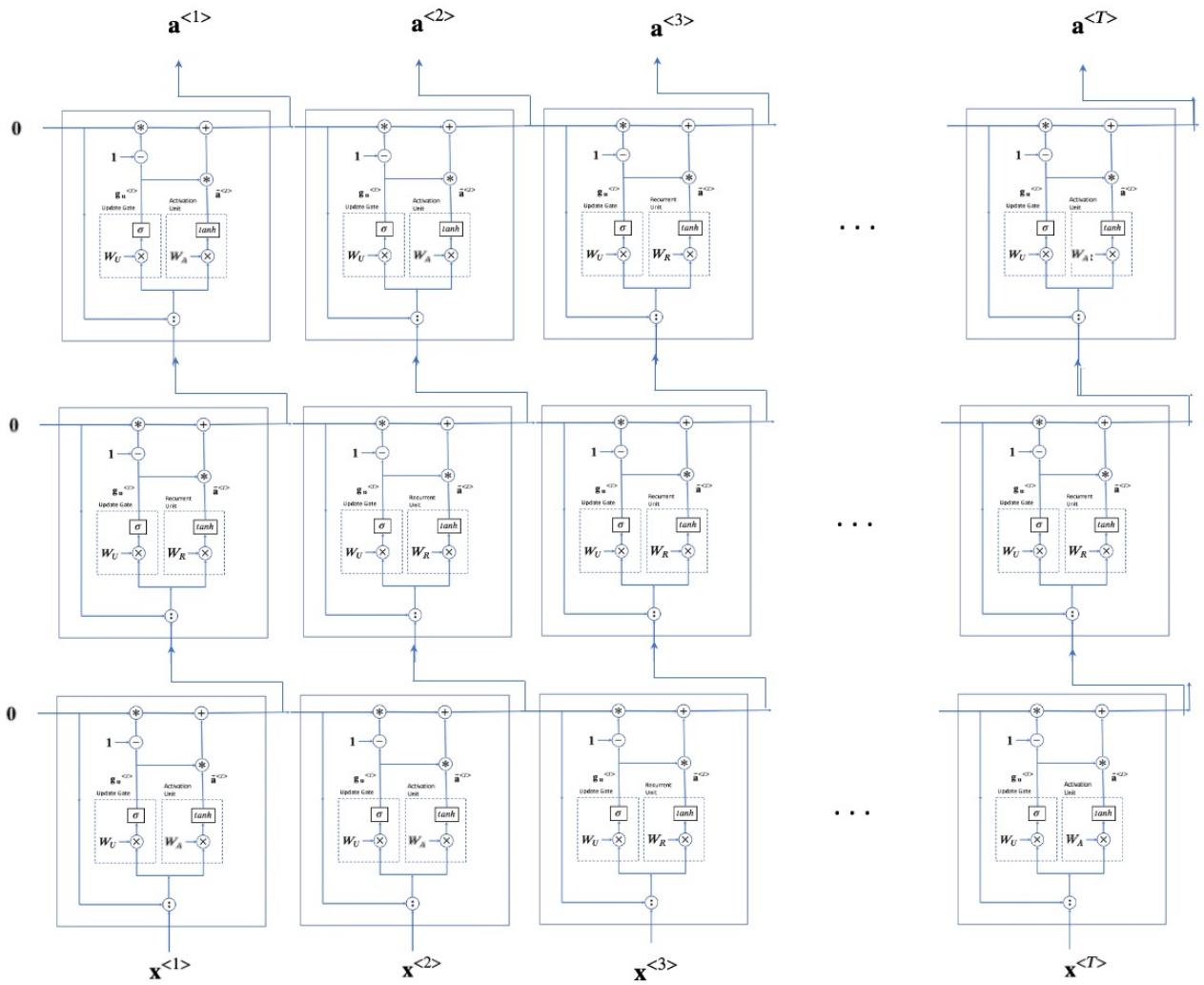
Recurrent Neural Networks: RNNs, GRUs, LSTMs

Unrolling a deep RNN network reveals a very complicated design!



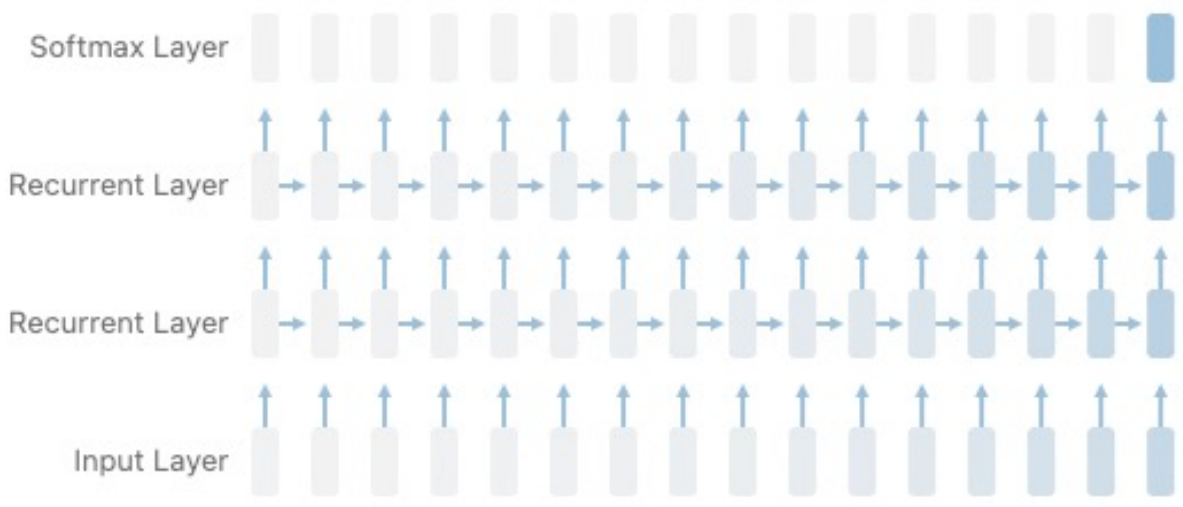
Recurrent Neural Networks: RNNs, GRUs, LSTMs

Unrolling a deep RNN network reveals a very complicated design!



Recurrent Neural Networks: RNNs, GRUs, LSTMs

Does this solve all our problems? Unfortunately not, due to the vanishing gradients problem: unrolling through time makes the network very large and preserving information (through weights) over long distances is a problem:

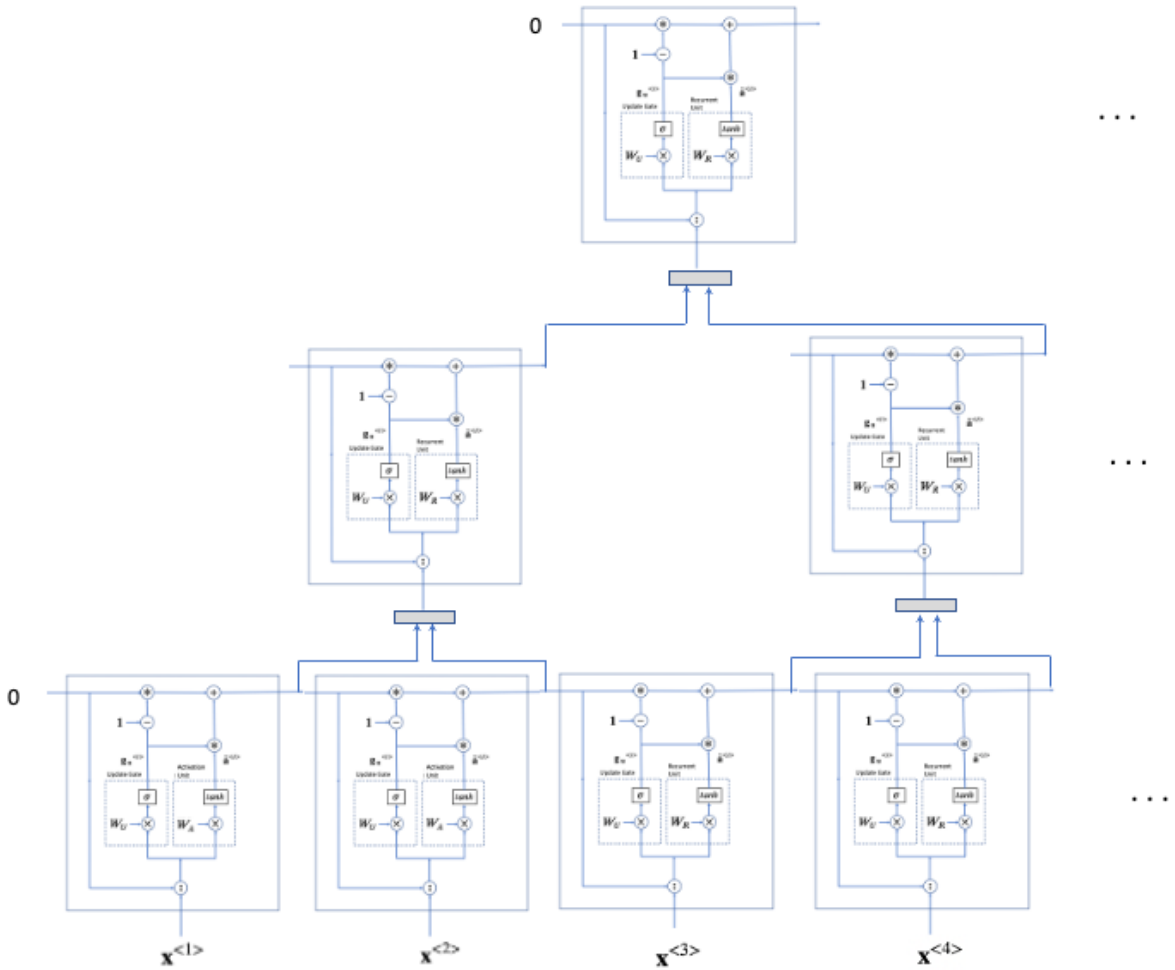


Vanishing Gradient: where the contribution from the earlier steps becomes insignificant in the gradient for the vanilla RNN unit.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

MANY different designs have been proposed, with advantages and disadvantages.

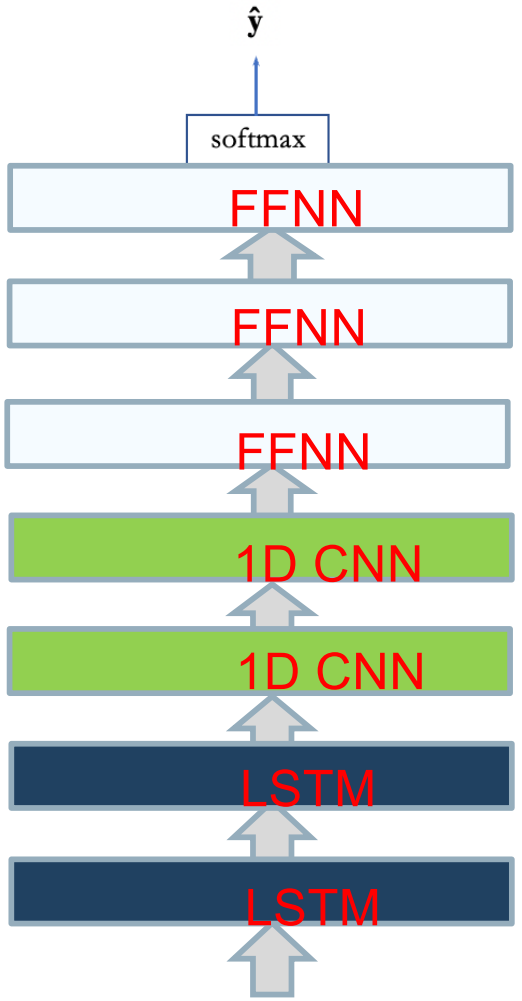
Idea 1: Tree-structured network which combines lower levels using some aggregating function (weighted) sum, perhaps controlled by a gate.



Recurrent Neural Networks: RNNs, GRUs, LSTMs

MANY different designs have been proposed, with advantages and disadvantages.

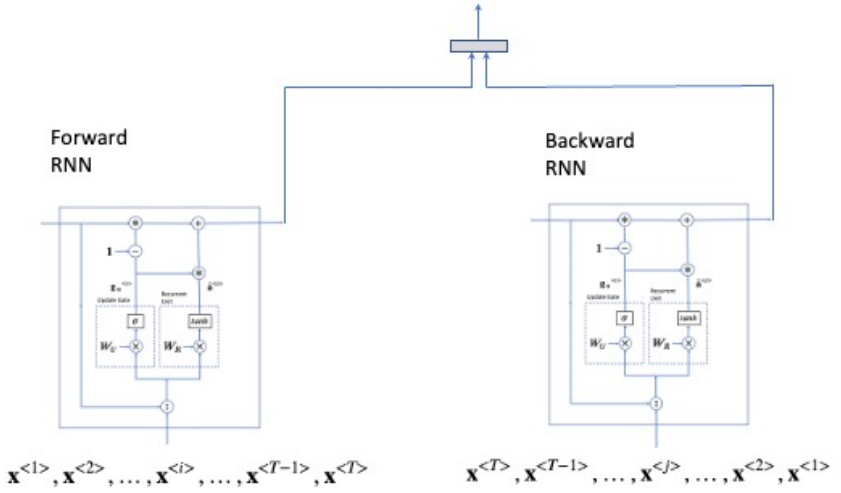
Idea 2: Apply 1D Convolutions to the RNN layers.



Recurrent Neural Networks: RNNs, GRUs, LSTMs

MANY different designs have been proposed, with advantages and disadvantages.

Idea 3: Bidirectional RNN (BRNN): Combine result of running two RNNs on forward and reverse sequence simultaneously. Results are fed to the next layer, usually by concatenation.



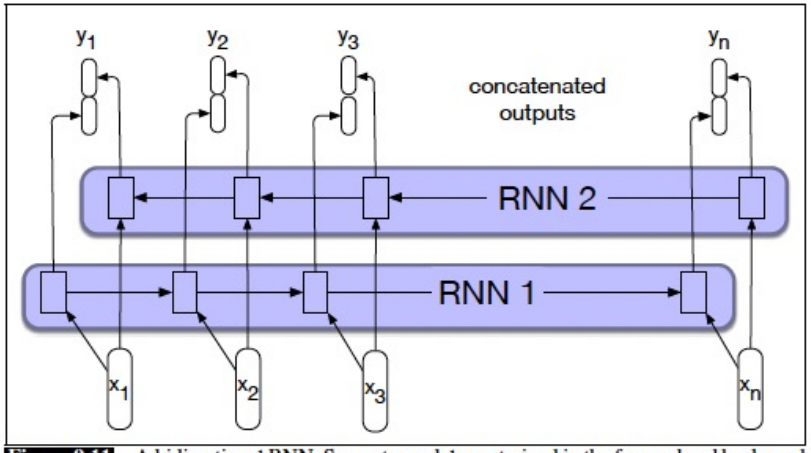


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

Training RNNs with Sequence Data: Classification

Example 1: Sentence classification

Easy! Just use the last output and proceed as usual!

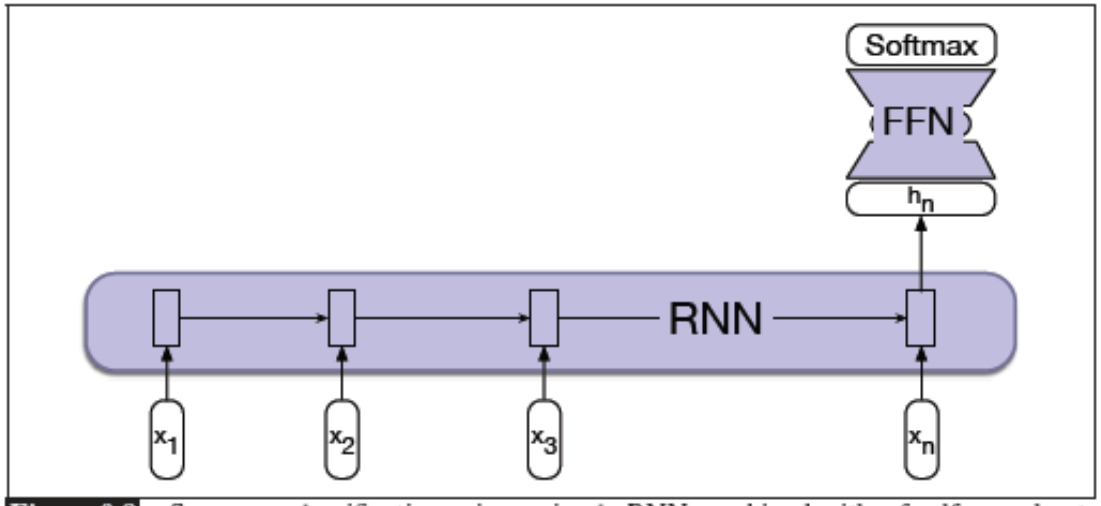


Figure 9.8 Sequence classification using a simple RNN combined with a feedforward network. The final hidden state from the RNN is used as the input to a feedforward network that performs the classification.

NOTE: From now on, we'll show every RNN unrolled through time, though you should always remember that there is a for loop controlling the whole process.)

Recurrent Neural Networks: RNNs, GRUs, LSTMs

Training RNNs with Sequence Data: POS Tagging

Example 2: Part-of-Speech Tagging

In POS Tagging, the input is a sentence, and the output is a sequence of multinomial classifications into parts of speech:

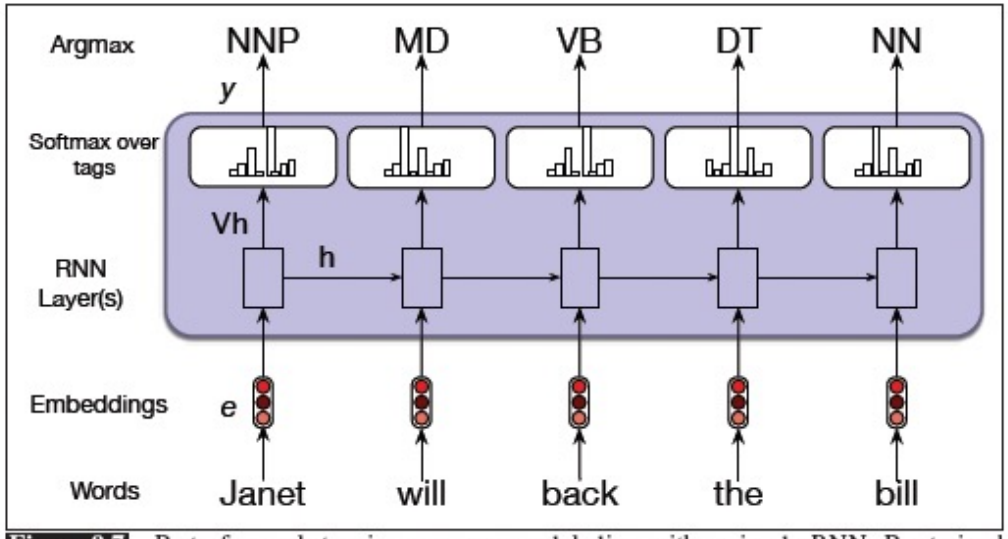


Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

NOTE: From now on, we'll show every RNN unrolled through time, though you should always remember that there is a for loop controlling the whole process.)

Recurrent Neural Networks: RNNs, GRUs, LSTMs

Example 3: A sentence generator using the trained RNN can generate sentences by picking the most likely next word in each step, until it generates the end-of-sentence token </s>:

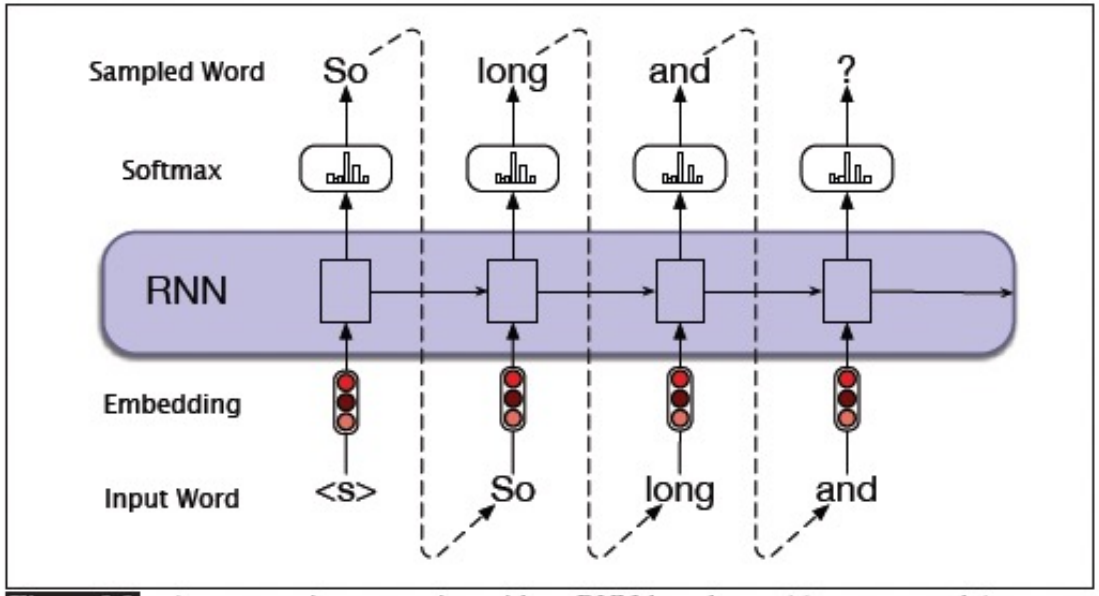


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

An important variation of the autoregressive generator is to give the RNN a context at the beginning; here we give the generator the start-of-sentence token <s> (which says that the next word is one that must follow <s>, duh...).

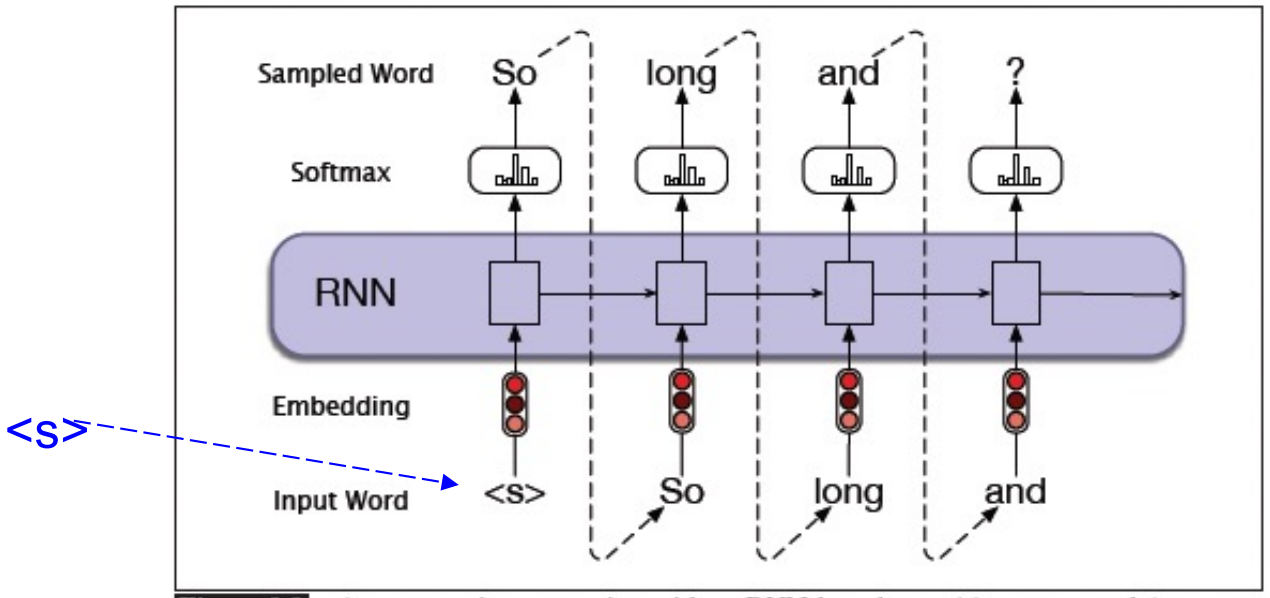
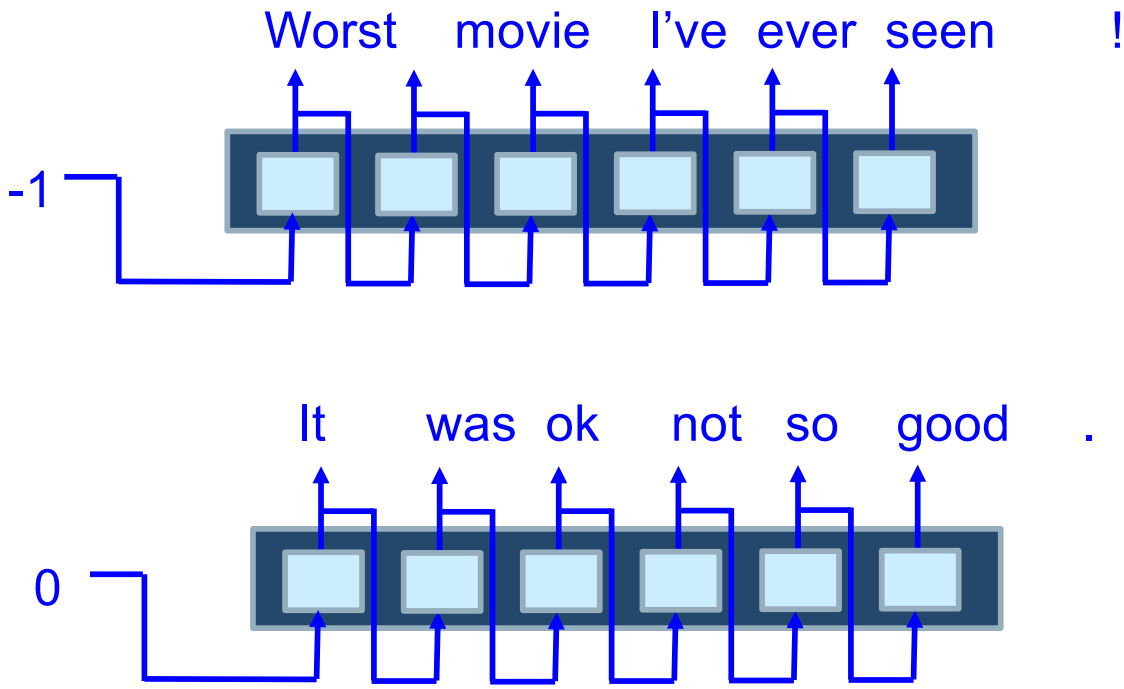


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

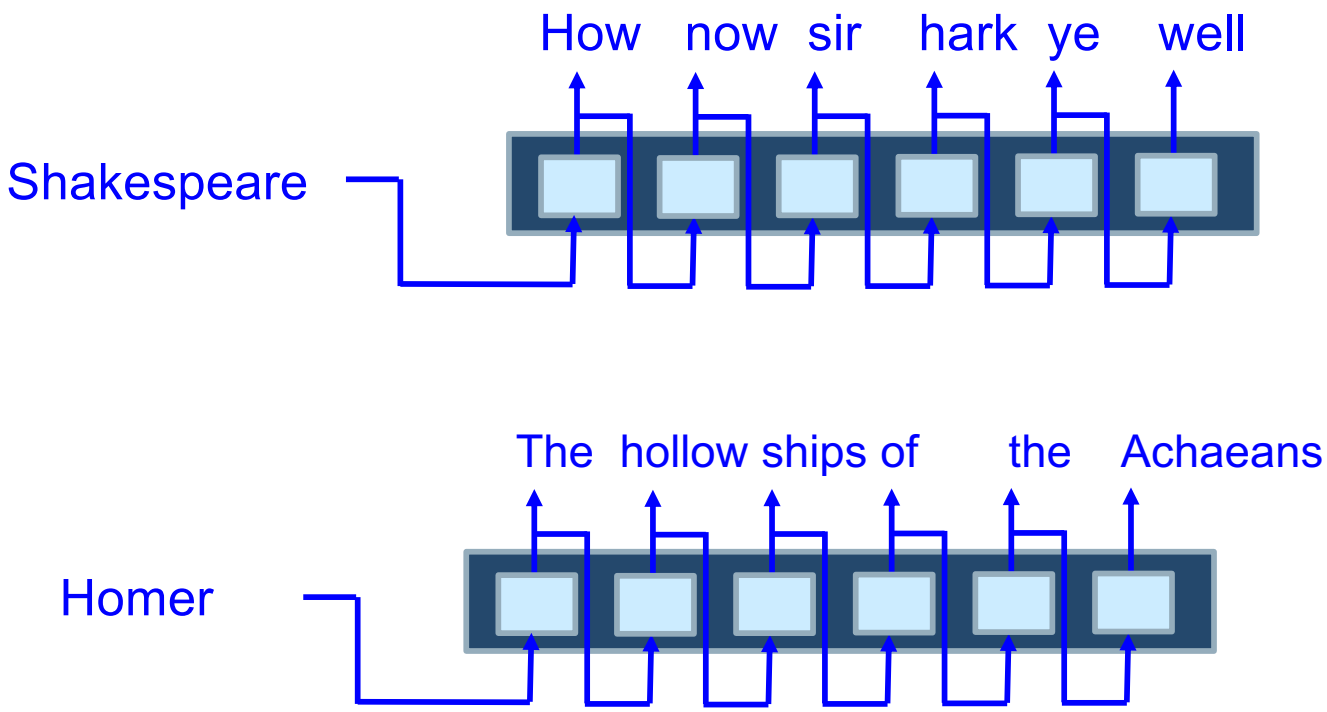
But this context could be anything! For example, we could give it an integer:

-1 Negative movie review
0 Neutral movie review
1 Positive movie review



Recurrent Neural Networks: RNNs, GRUs, LSTMs

Or it could be an author to imitate:



Recurrent Neural Networks: RNNs, GRUs, LSTMs

Digression on a significant problem (and solution) in language generation: The RNN makes local decisions about the most likely next word. However, a series of such local decisions will not necessarily find the globally most likely sentence (cf. gradient descent, which has the same problem).

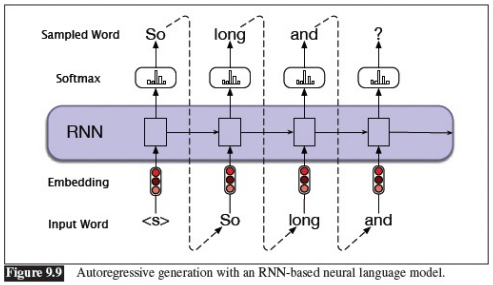


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

The usual optimization is Beam Search:

1. Pick the "width of the beam" N (at each iteration, we will store the N most likely sequences of words);
2. Generate a list of the N most likely words to start a sentence, and concatenate them with <s>;
3. At each iteration, examine ALL possible next words in the sequence; toss all but the N most likely sequences;
4. Repeat until </s> is generated. Return the most likely sentence.

Note: sentences might be different lengths; stop when sequence ends in </s>.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

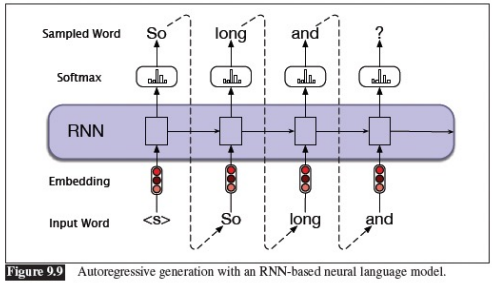
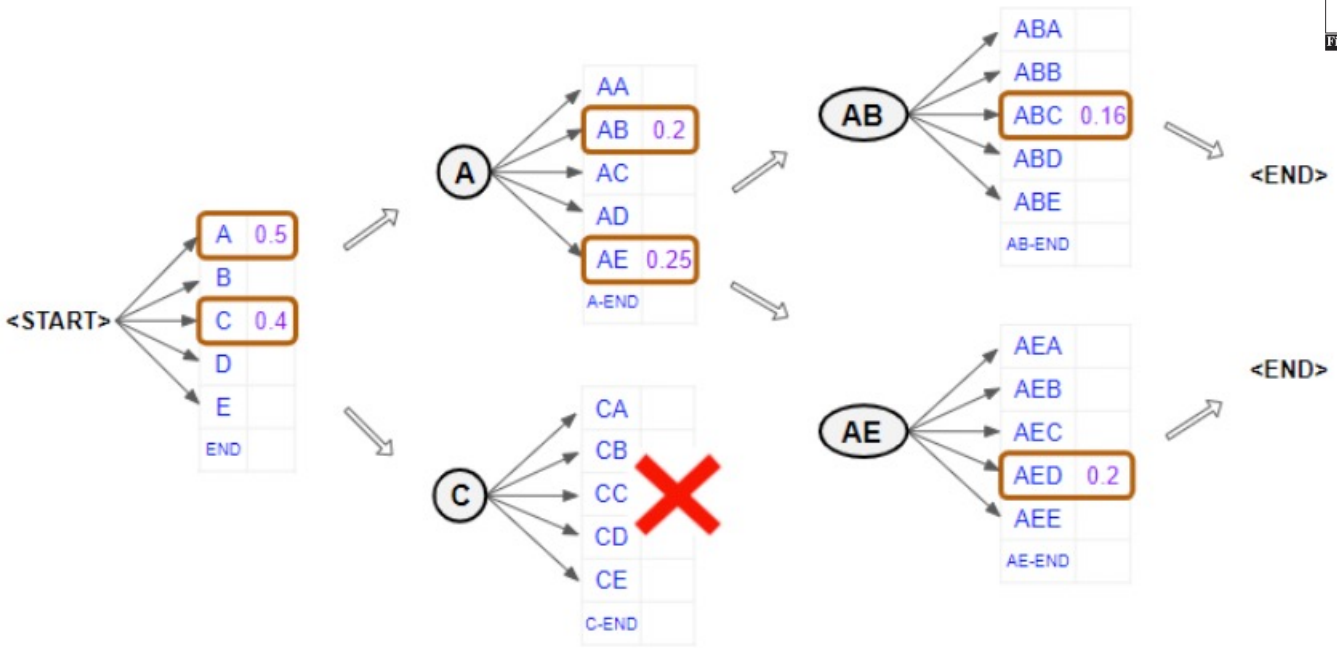


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Example of Beam Search with N = 2 using letters instead of words:



Result:
AED

Punchline: Beam search is not guaranteed to find the optimal sequence, but as a heuristic it works very well. There is an obvious efficiency/performance tradeoff. Common values of N are 10, 100, 1000.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

The Encoder-Decoder architecture combines sequence-to-one and a one-to-sequence models:

The encoder is "just" an ordinary RNN, producing a context as its result;
The decoder is a generator, producing the most likely output given the context.

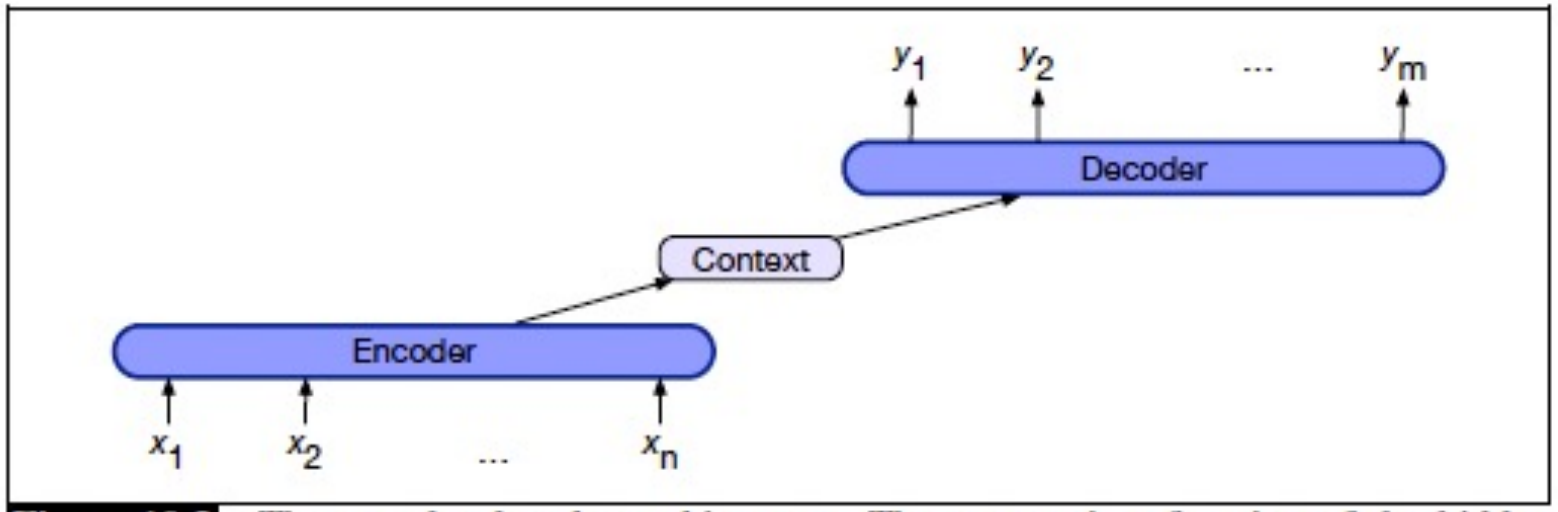
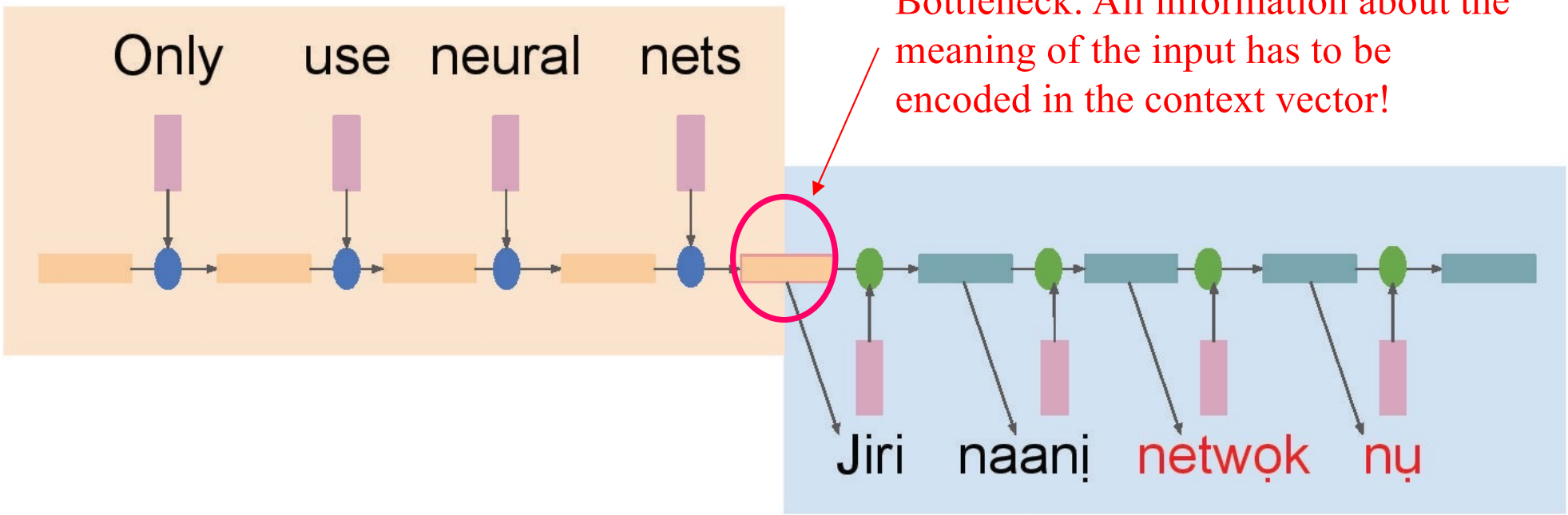


Figure 10.3 The encoder-decoder architecture. The context is a function of the hidden representations of the input, and may be used by the decoder in a variety of ways.

Recurrent Neural Networks: RNNs, GRUs, LSTMs

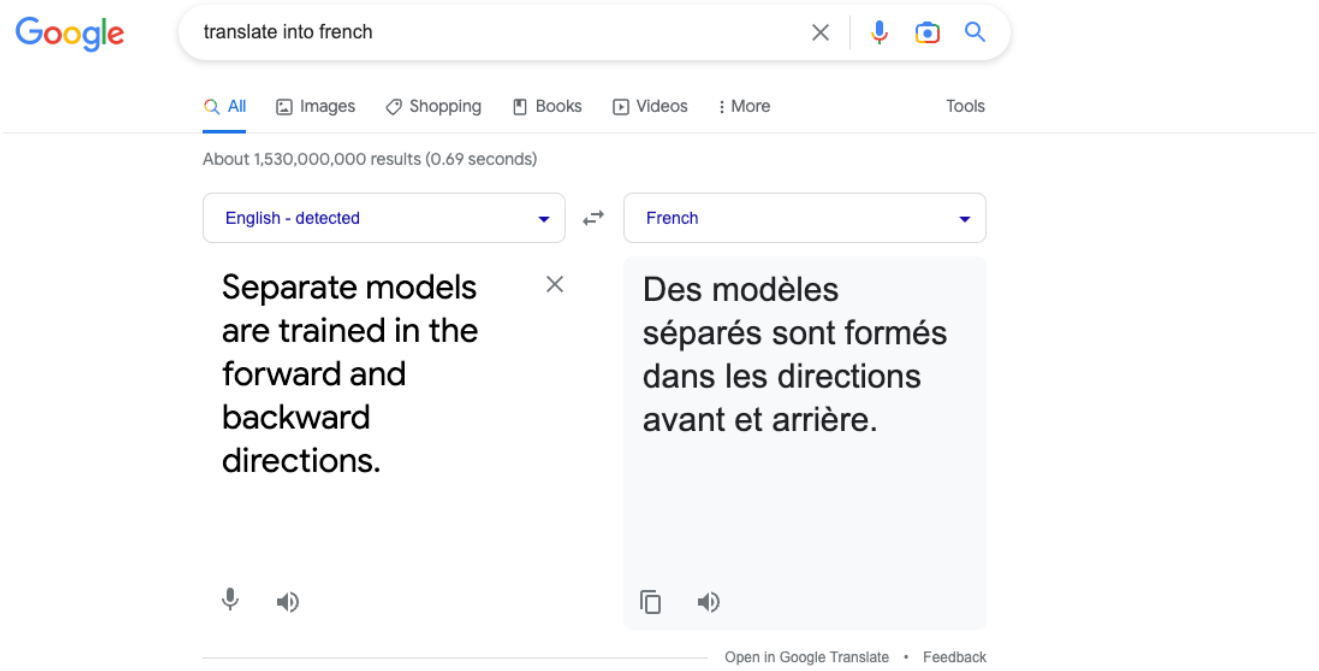
Problem: If we want to do translation between languages, the context has to hold a LOT of information! This motivates the notion of attention and the transformer model we will study next time.

Given the diagram below, what problem do you foresee when translating progressively longer sentences?



Deep Learning: The Attention Architecture

To see the motivation for Attention, consider translating “Separate models are trained in the forward and backward directions.” into various languages....



Deep Learning: The Attention Architecture

Machine Translation with BRNN

Notice that, as with most modern languages, the words have a similar sequential order, but there is some variation in position:

French:

Des modeles séparés sont formés dan les directions avant et arrière.

Separate models are trained in the forward and backward directions.

Spanish

Los modelos separados se entrenan en las direcciones hacia adelante

Separate models are trained in the forward and backward directions.

Deep Learning: The Attention Architecture

Machine Translation with BRNN

A BRNN can help with this problem, because when it creates the activation vector at one point in the sequence, it has access to the backward and forward context:

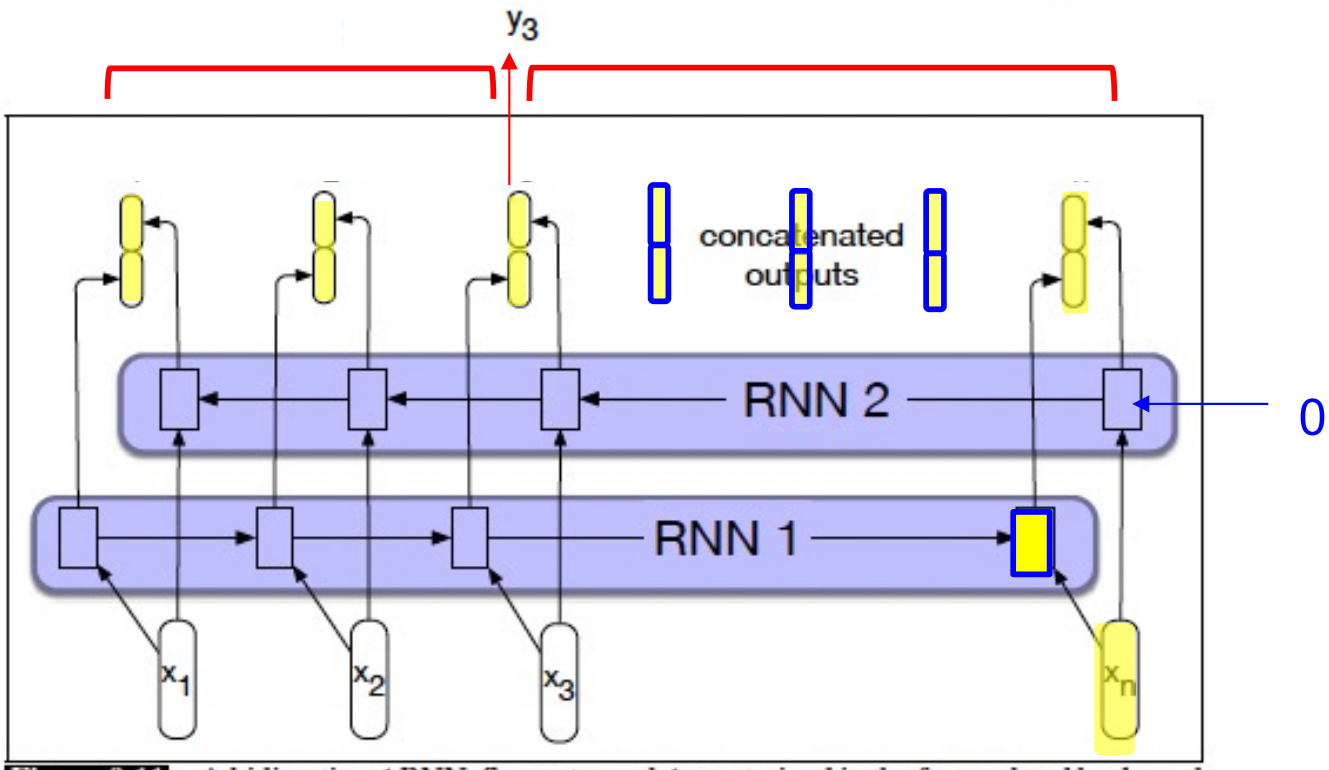


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Deep Learning: The Attention Architecture

Machine Translation with BRNN

However, the ability to remember context ”fades” the farther you are from the current activation, and it would be useful to have more control of specific words in the forward and backward context.

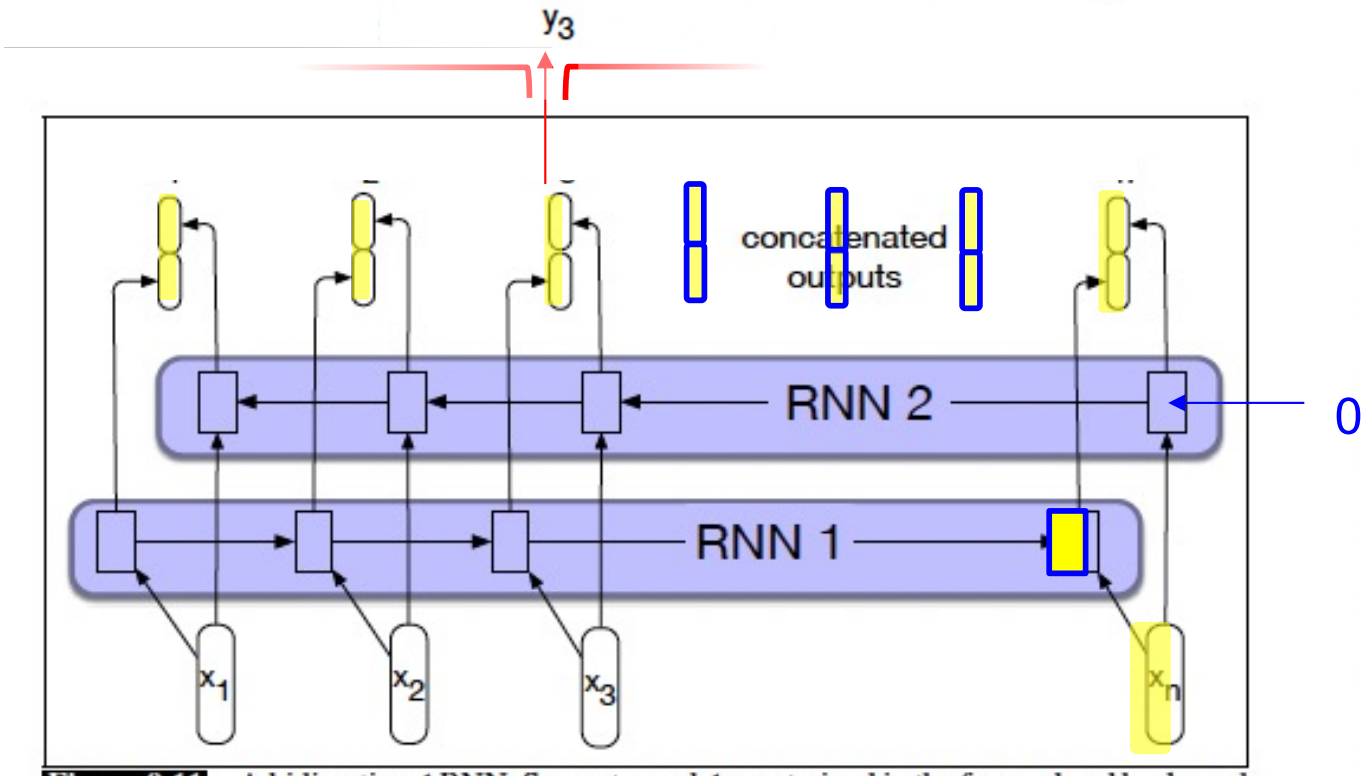
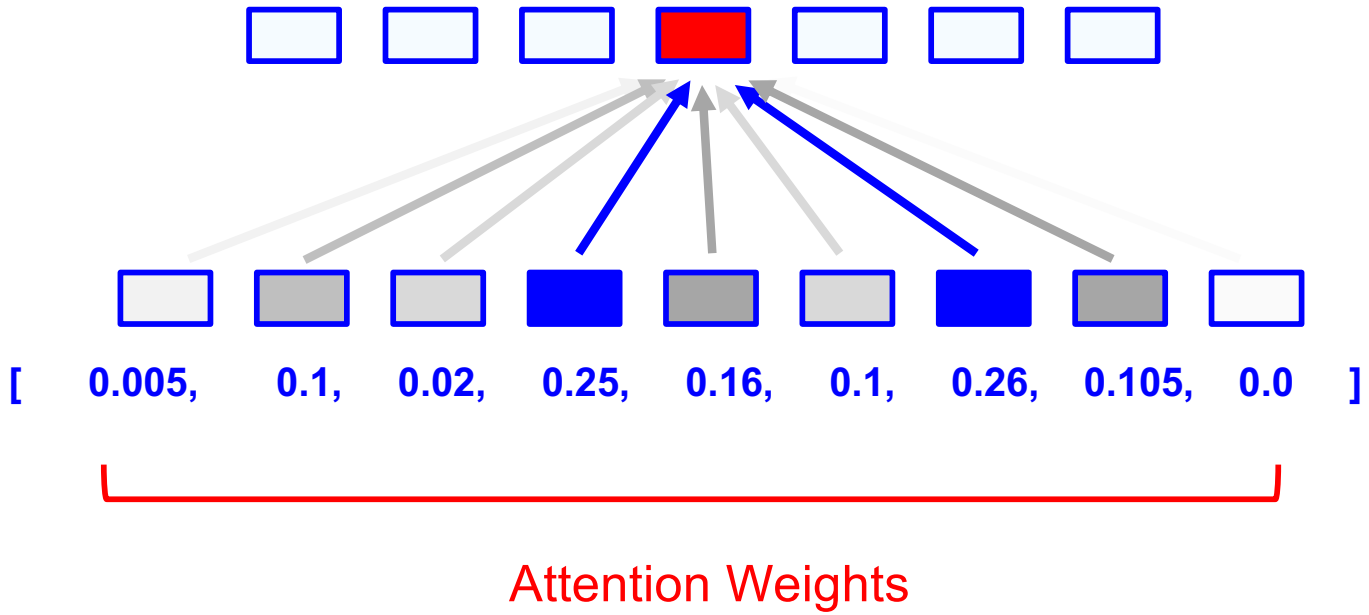


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention

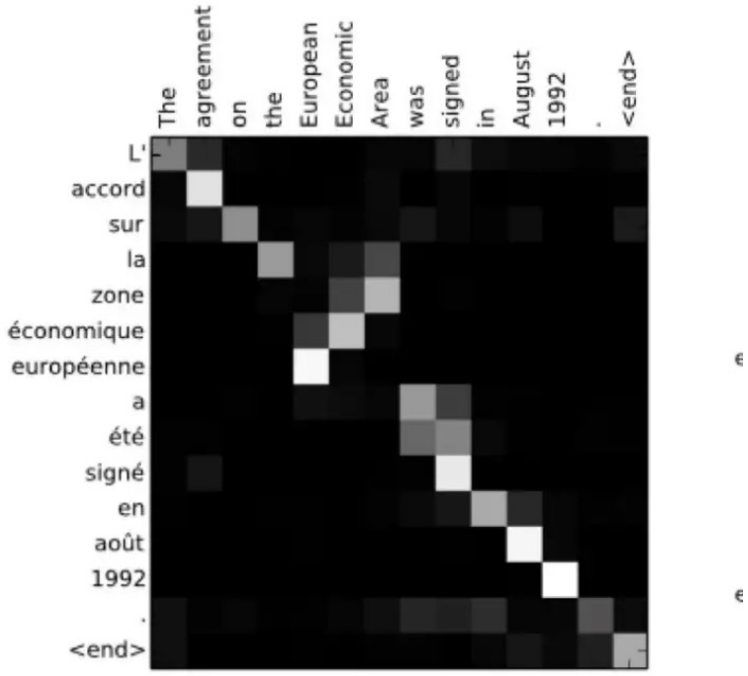
Attention refers the ability to focus on particular words in the backward and forward context; the pattern of what words matter in which context can be learned by the network. The pattern can be represented by a probability distribution over the sequence of input tokens:



Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention

The basic idea of attention is to train a FFNN to produce these attention weights for each output token; the resulting 2D matrix of attention weights shows the relationship between the words in the input sentence and those in the output sentence.



Deep Learning: The Attention Architecture

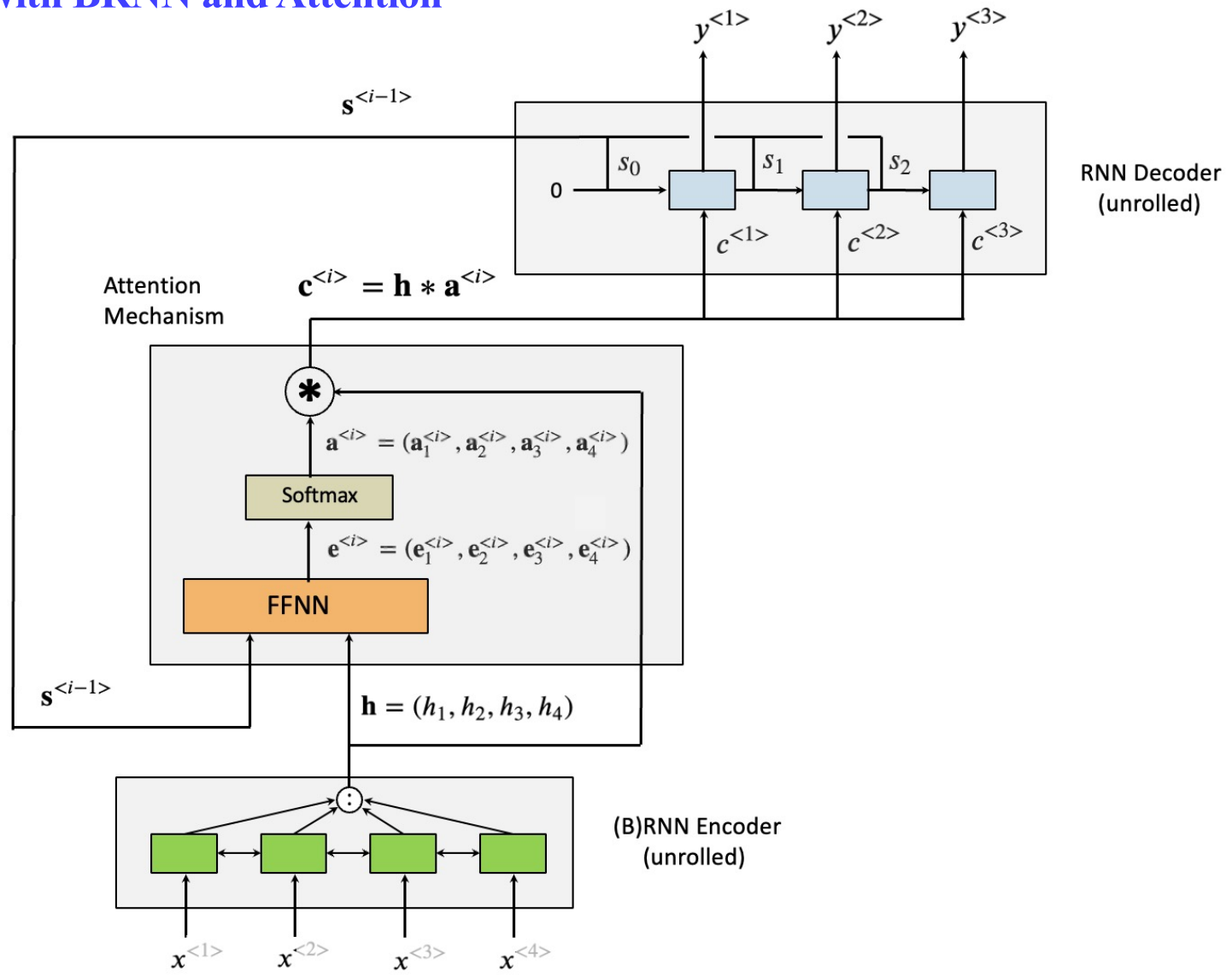
Machine Translation with BRNN and Attention

We add a FFNN between the encoder and the decoder to calculate the attention vector $\mathbf{a}^{<i>}$ for each output token.

Input to FFNN is output vector **h** from Encoder and activation vector $\mathbf{s}^{<i-1>}$ from Decoder.

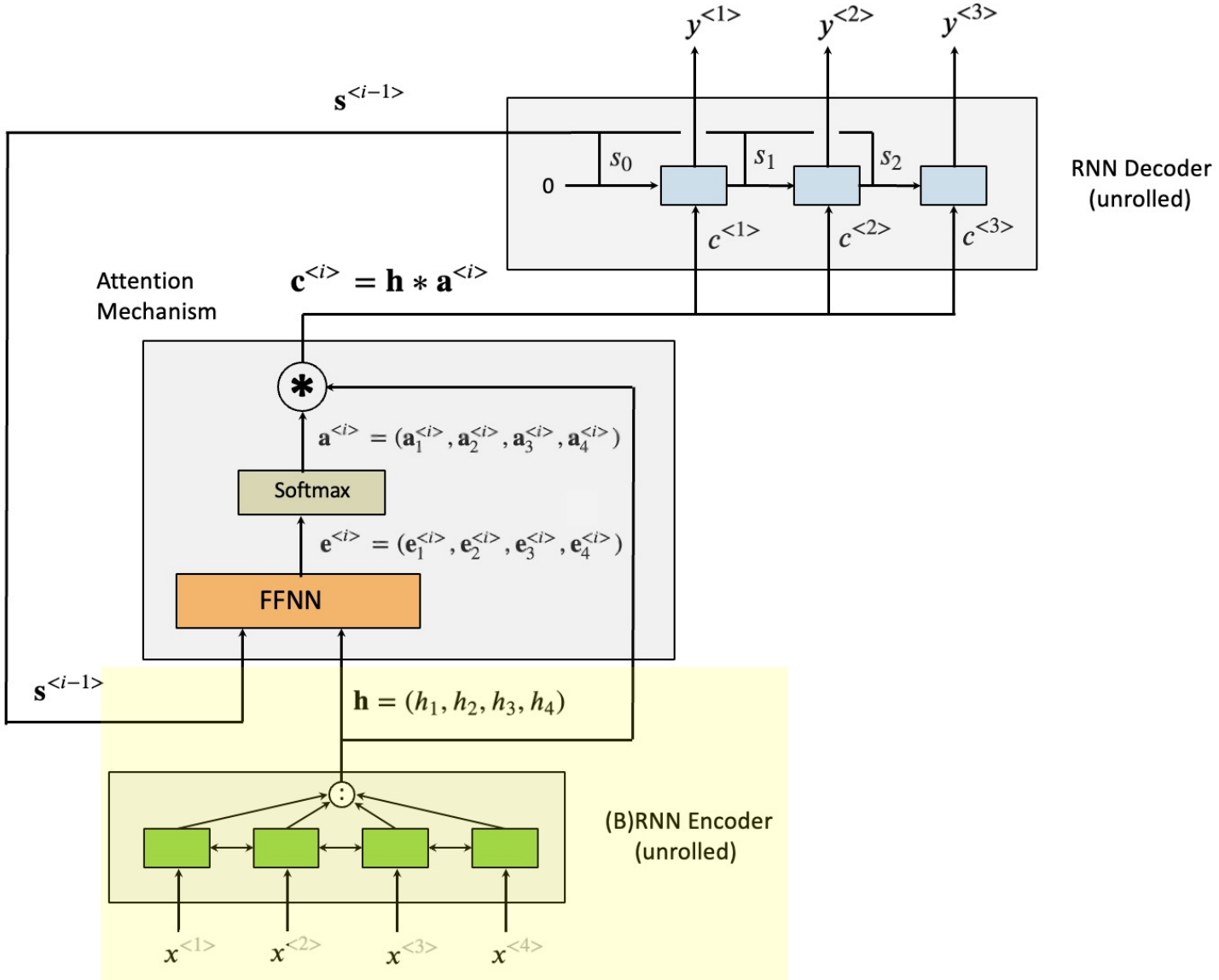
Attention weight vector $\mathbf{a}^{<i>}$ is softmax of output $\mathbf{e}^{<i>}$ from FFNN.

Element-wise product of $\mathbf{a}^{<i>}$ and **h** produces the context vector $\mathbf{c}^{<i>}$.



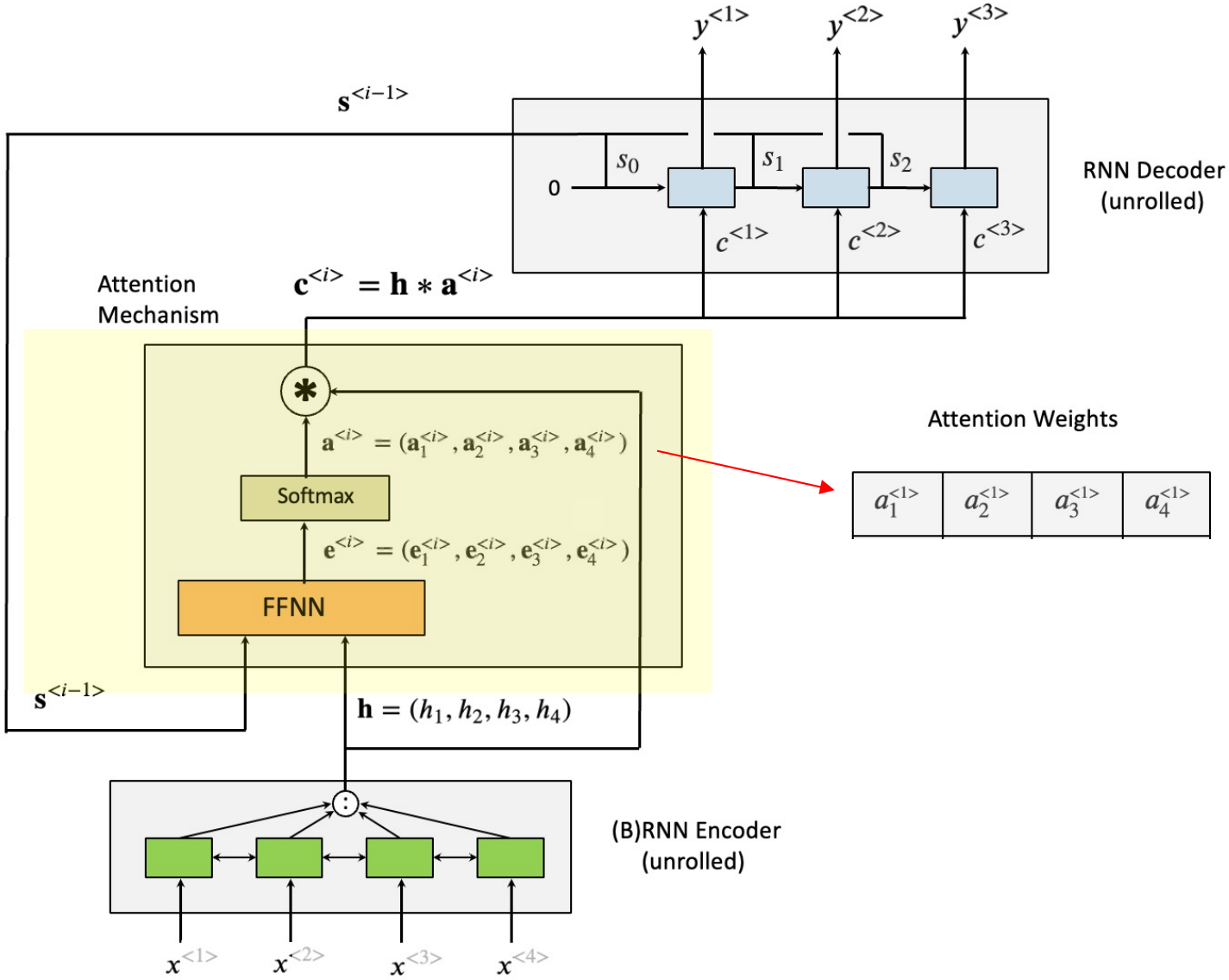
Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention



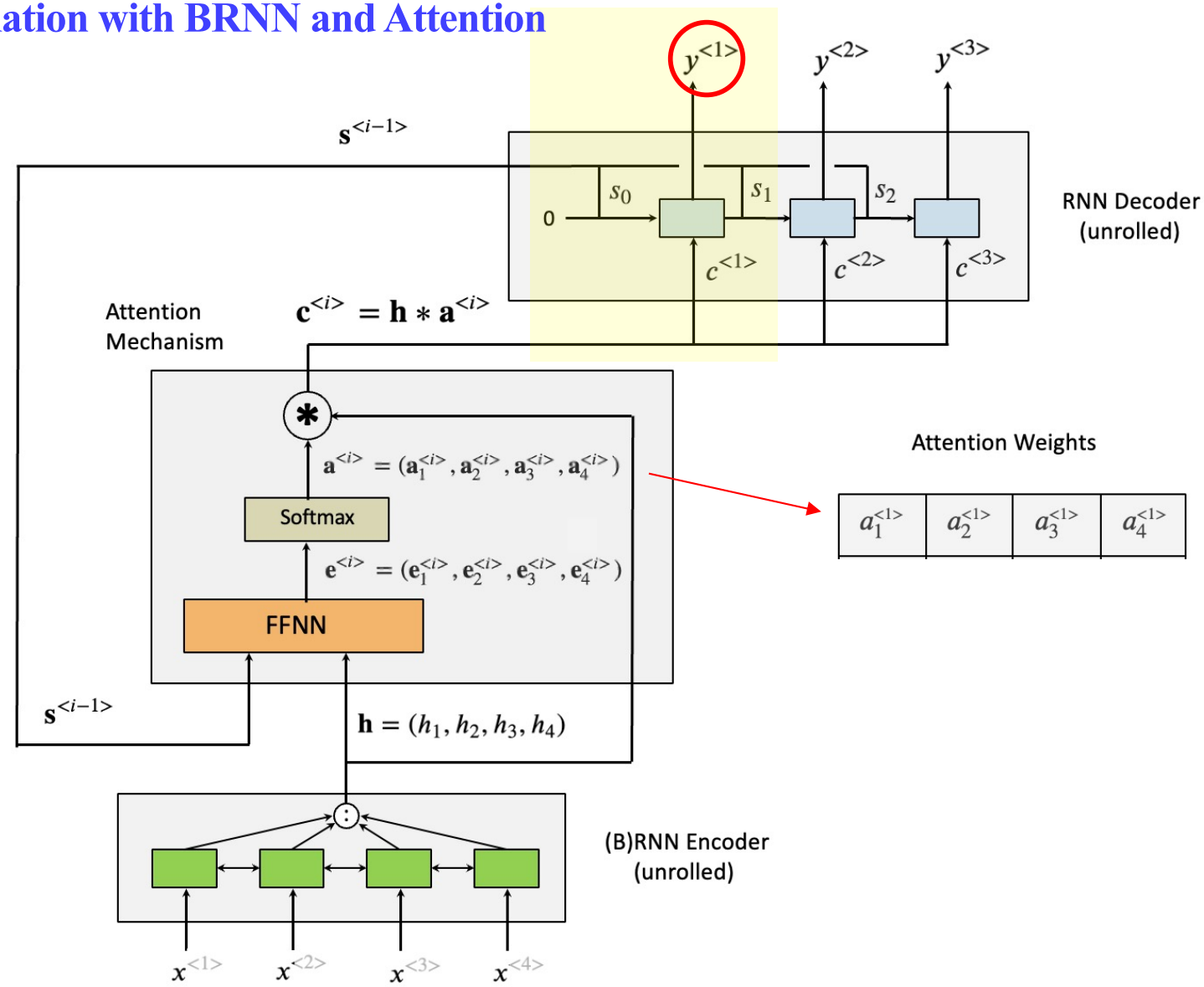
Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention



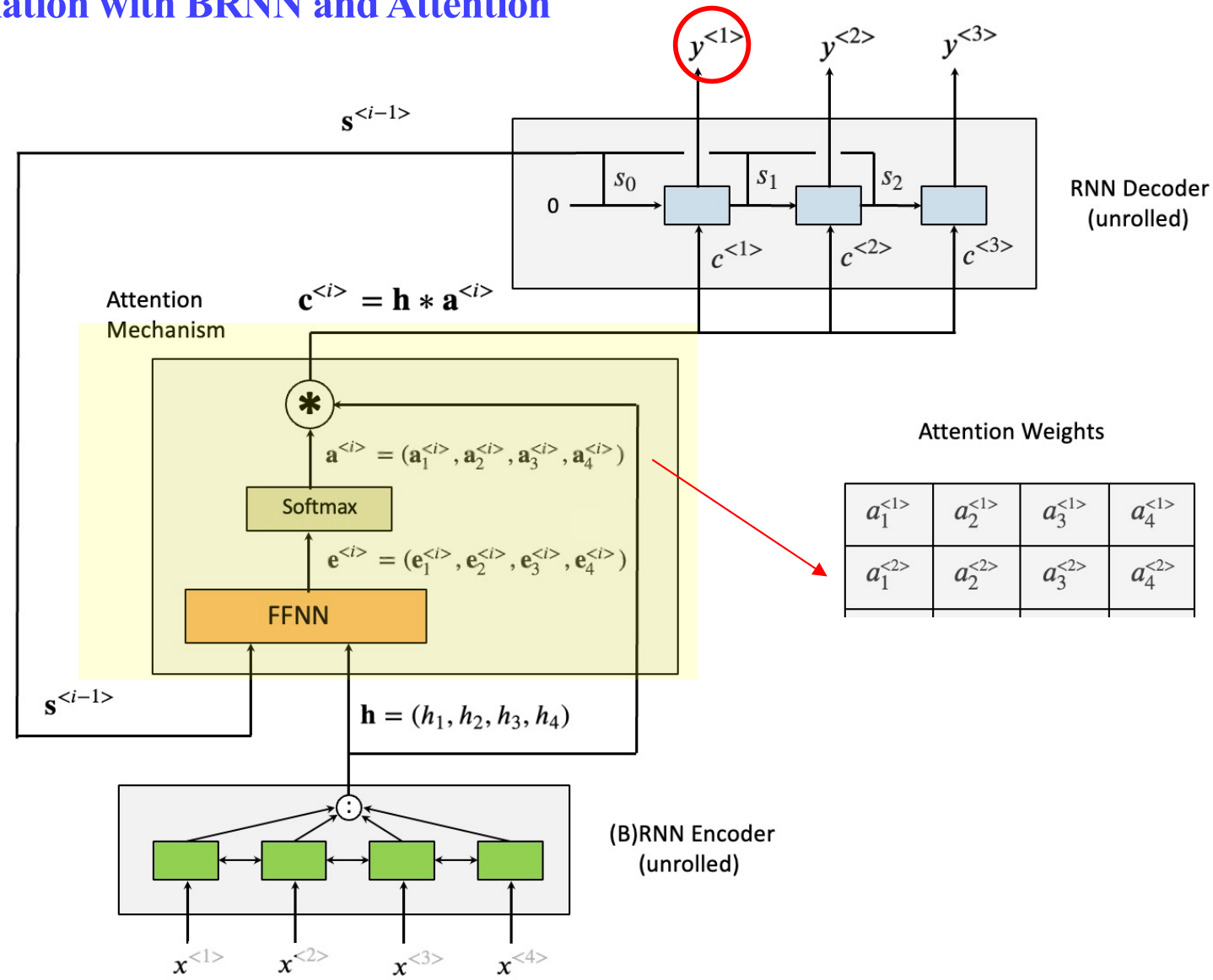
Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention



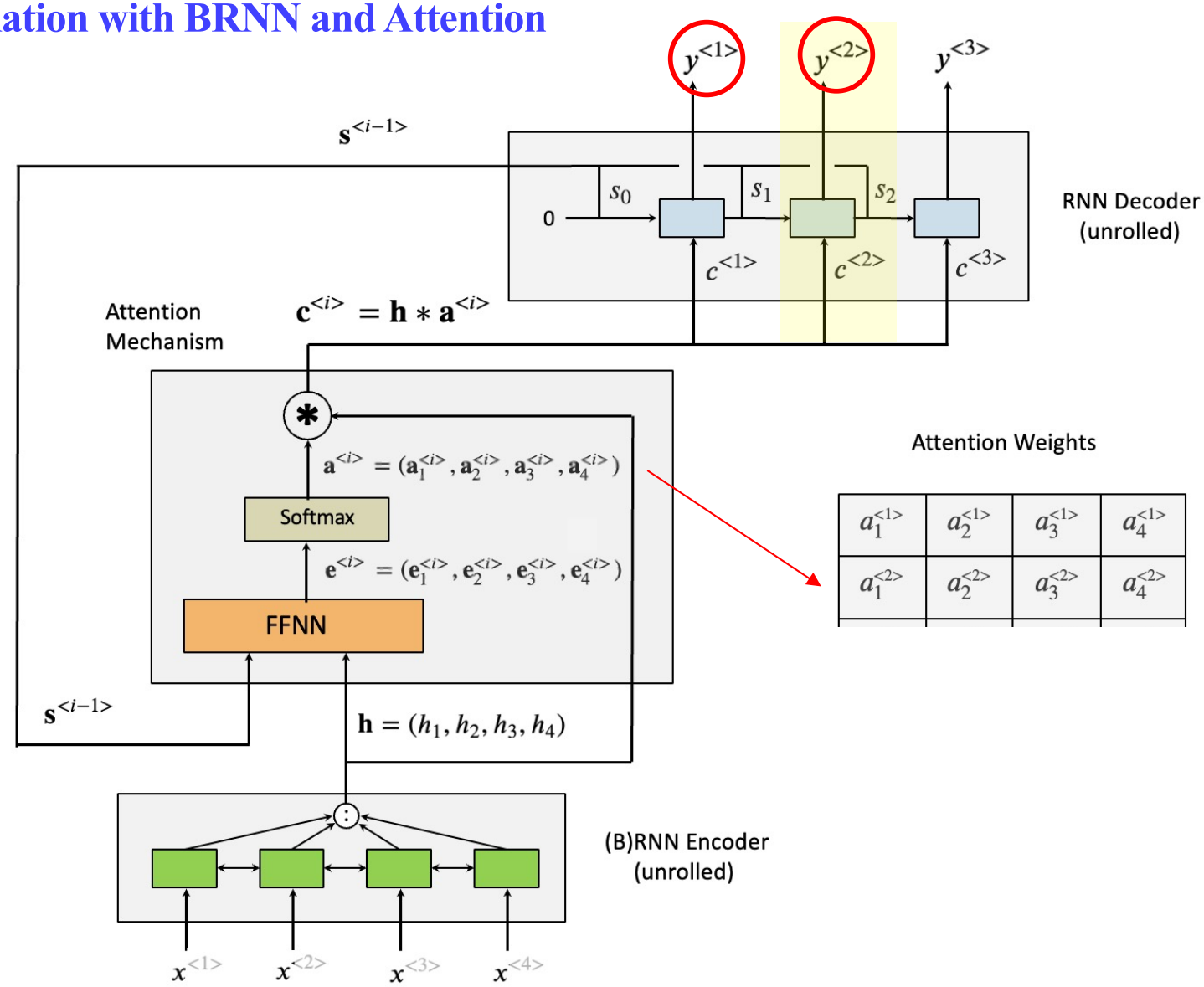
Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention



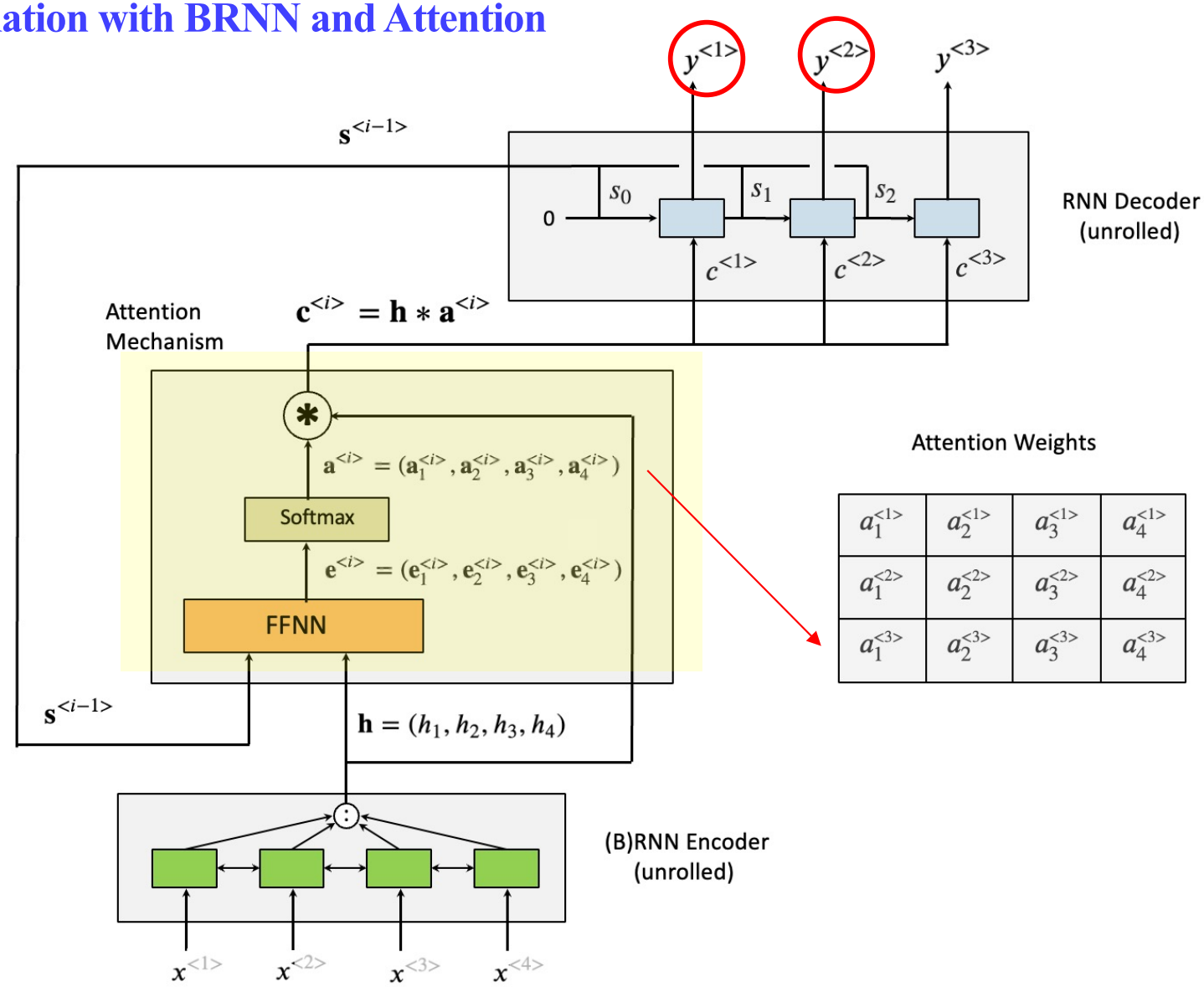
Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention



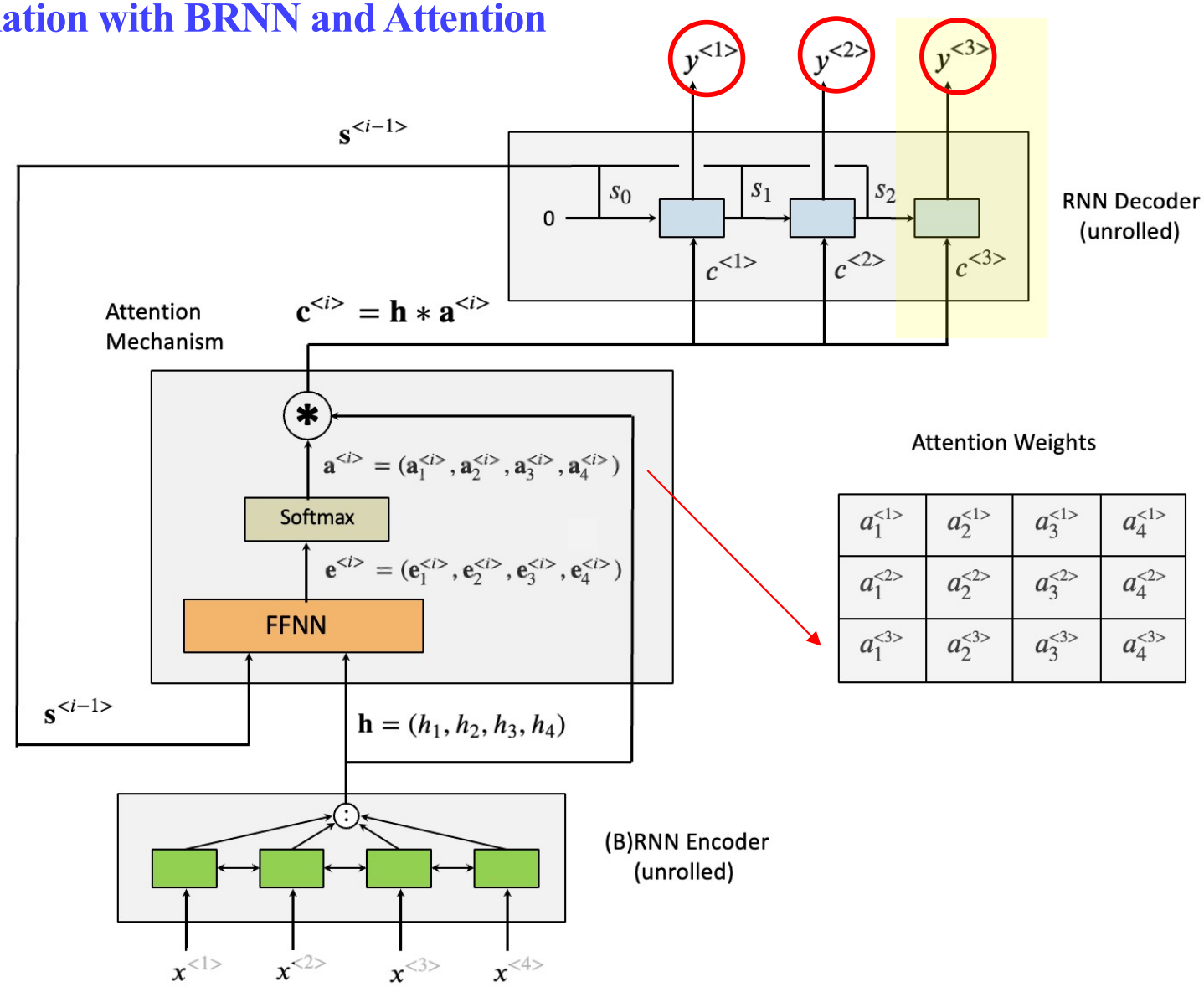
Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention



Deep Learning: The Attention Architecture

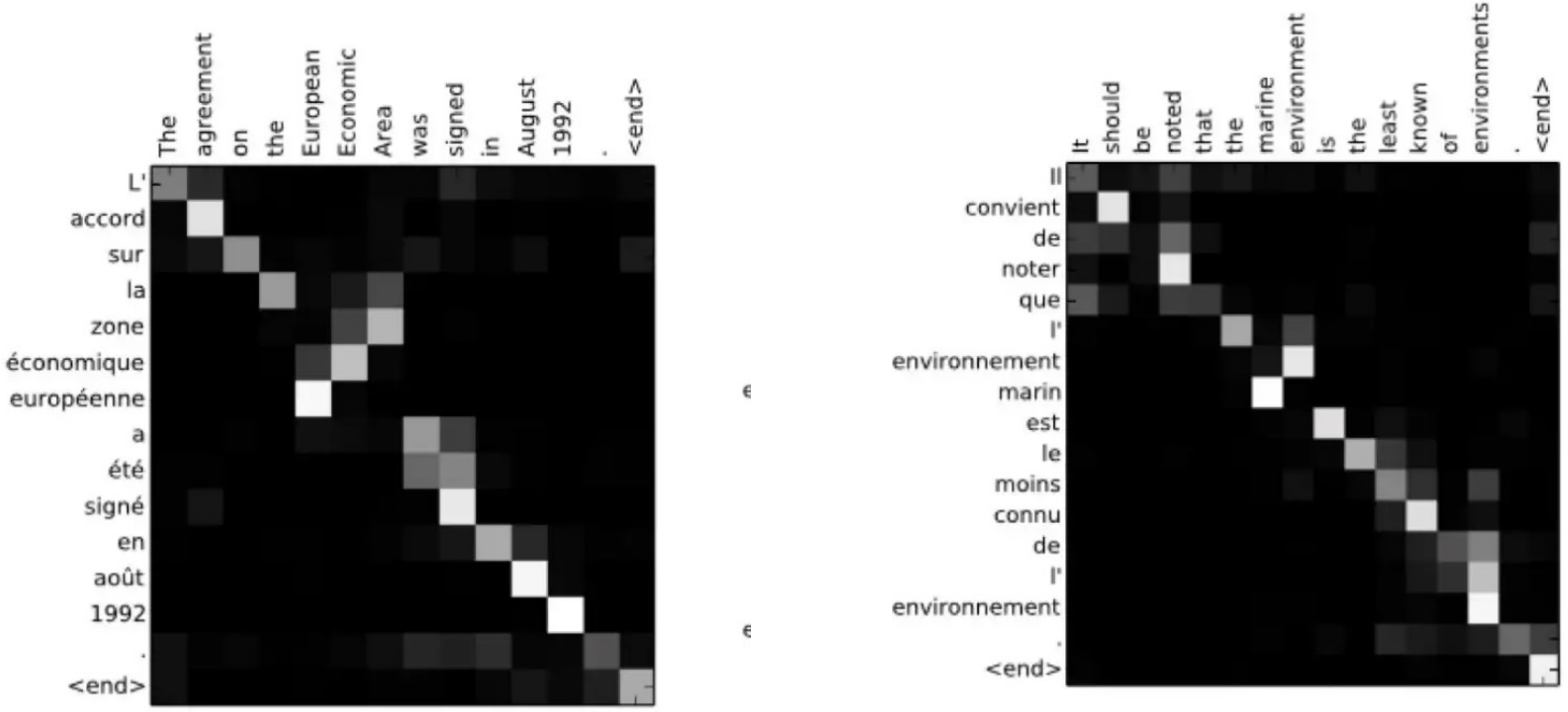
Machine Translation with BRNN and Attention



Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention

Displaying the activation matrix shows how attention was applied to the translation:



Deep Learning: The Attention Architecture

Machine Translation with BRNN and Attention

Attention can be used in many other contexts such as Automatic Speech Recognition and Image Captioning:

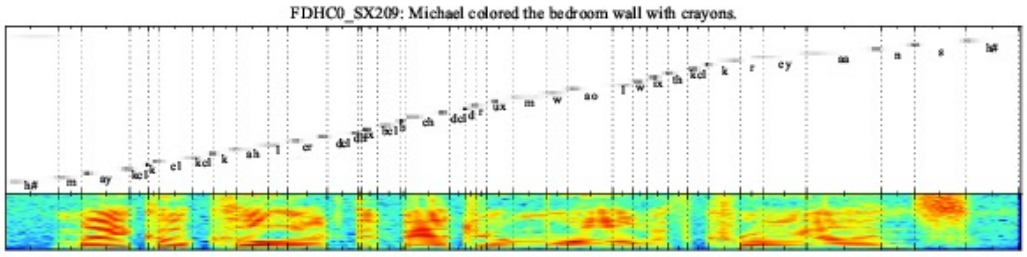


Figure 3: Alignments produced by the baseline model. The vertical bars indicate ground truth phone location from TIMIT. Each row of the upper image indicates frames selected by the attention mechanism to emit a phone symbol. The network has clearly learned to produce a left-to-right alignment with a tendency to look slightly ahead, and does not confuse between the repeated "kcl-k" phrase. Best viewed in color.

Fig. 7. "A woman is throwing a frisbee in a park." (Image source: Fig. 6(b) in Xu