Lecture 9: RNN, Seq2Seq & Attention Part2

Olexandr Isayev

Department of Chemistry, CMU

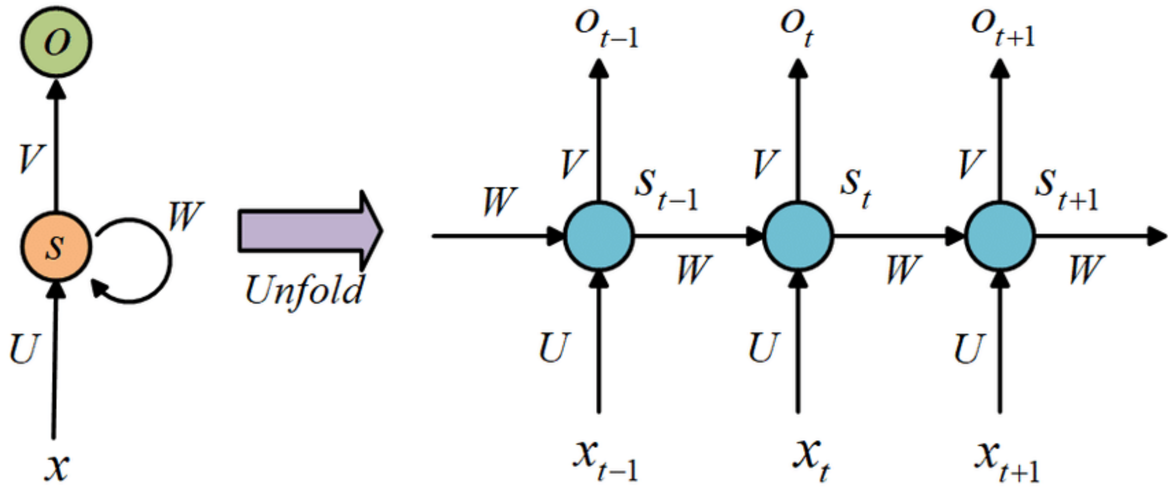
olexandr@cmu.edu

Home Assignment #3

In Class Project

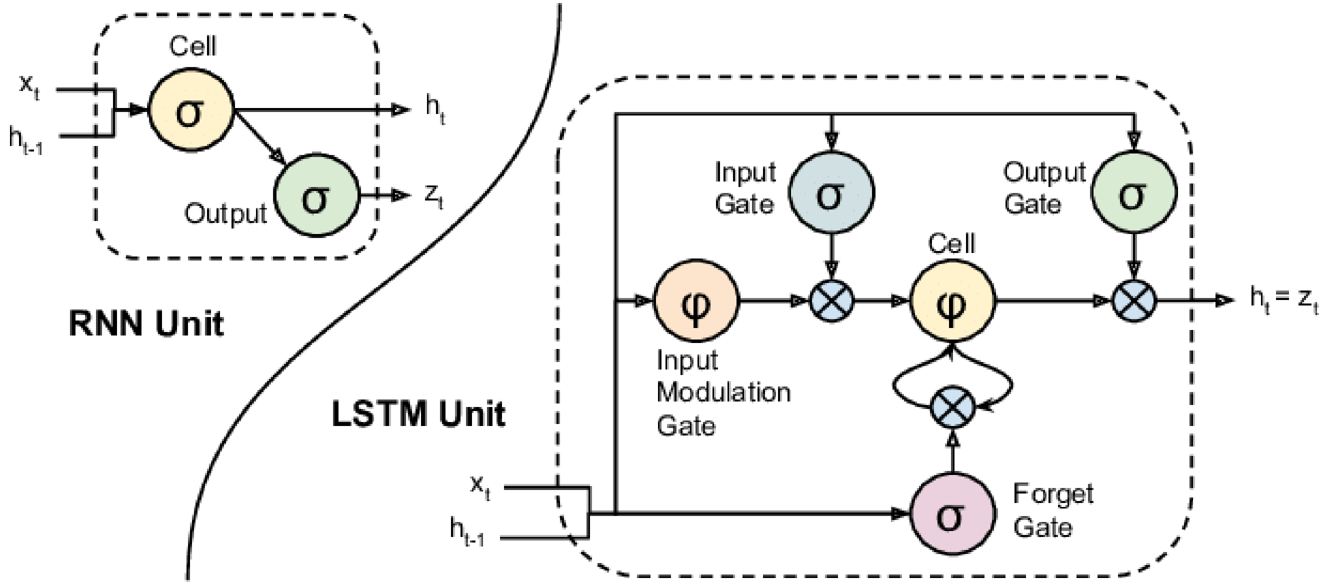
Main Concept of RNNs

RNN cell

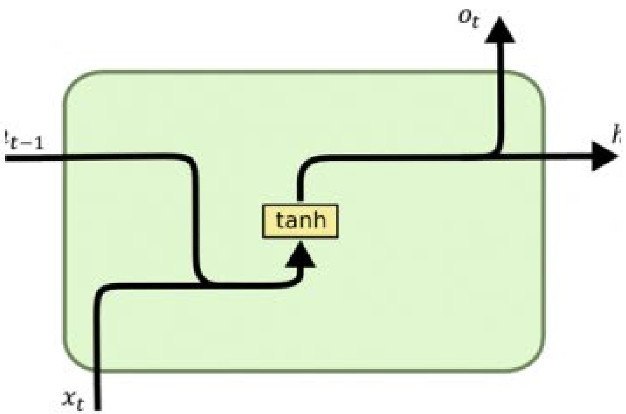


Time

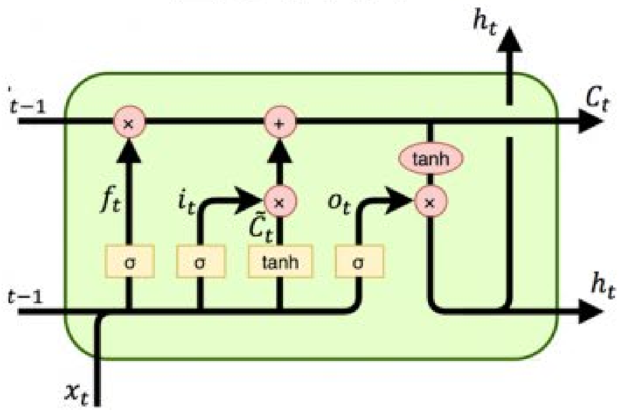
RNN Cells



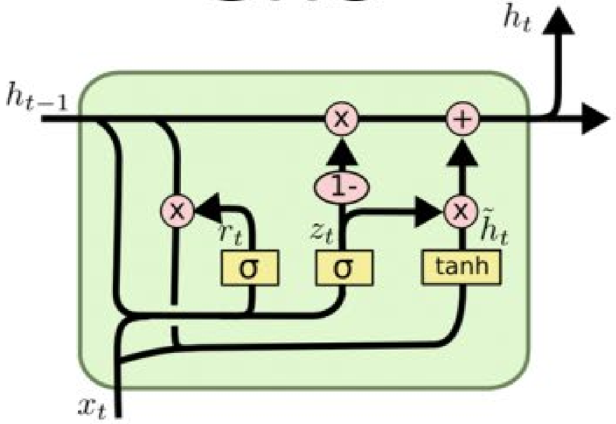
RNN



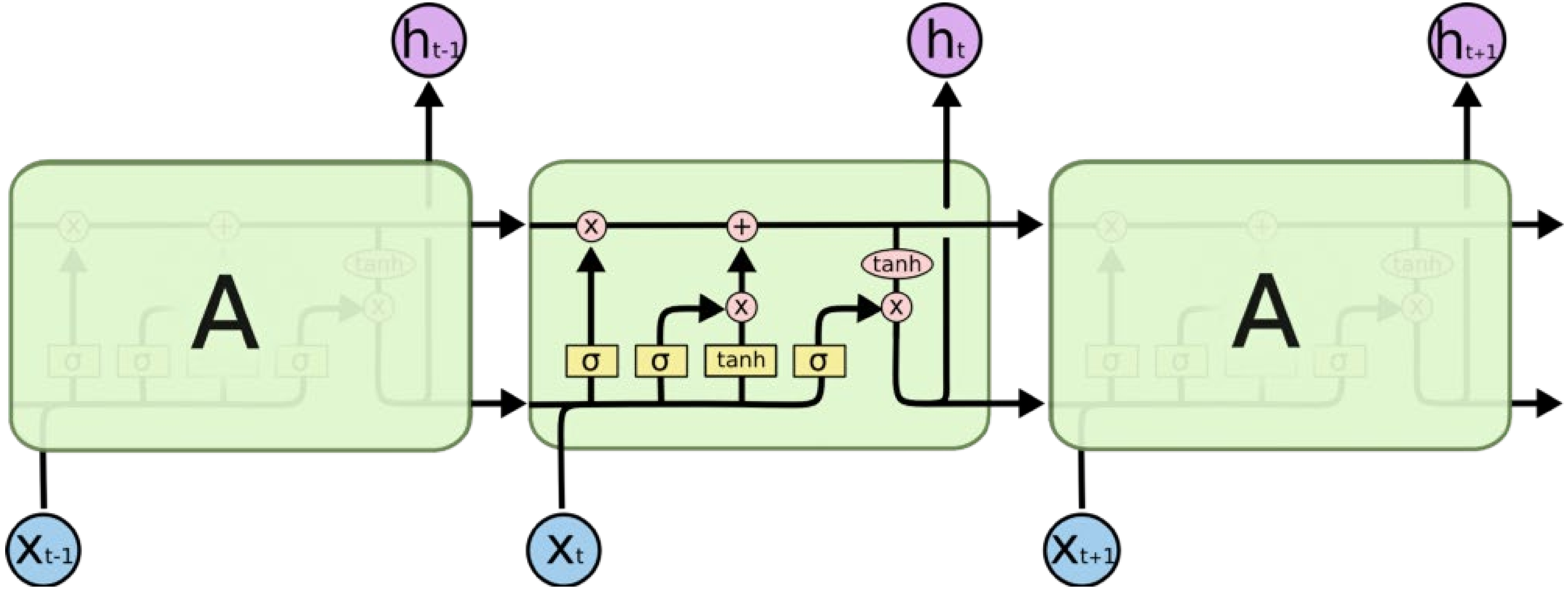
LSTM



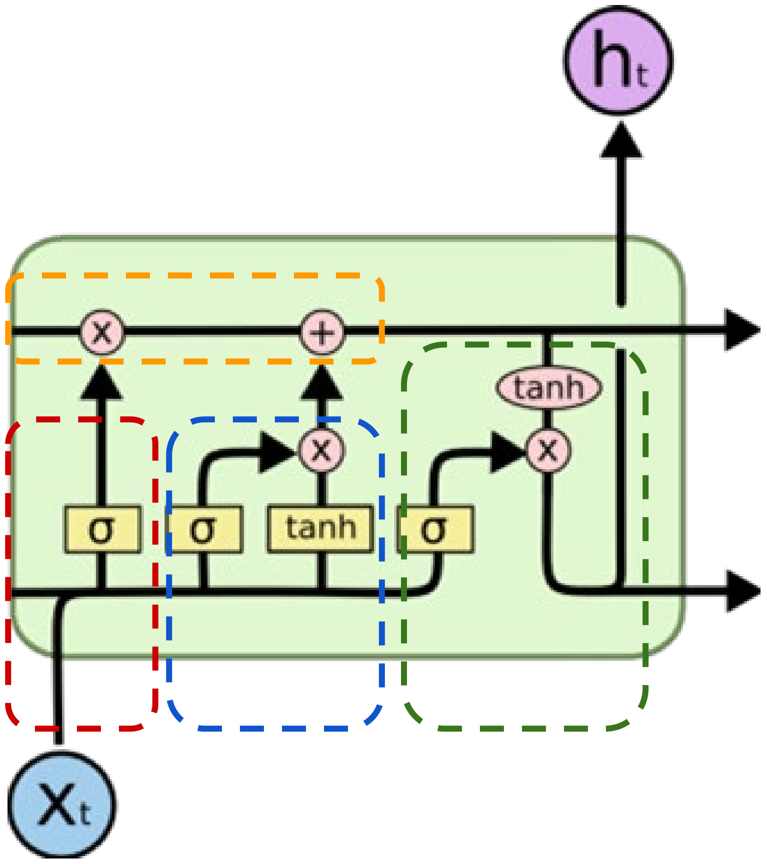
GRU



LSTM: Long short term memory



LSTM big picture ...



Cell State

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Forget Gate

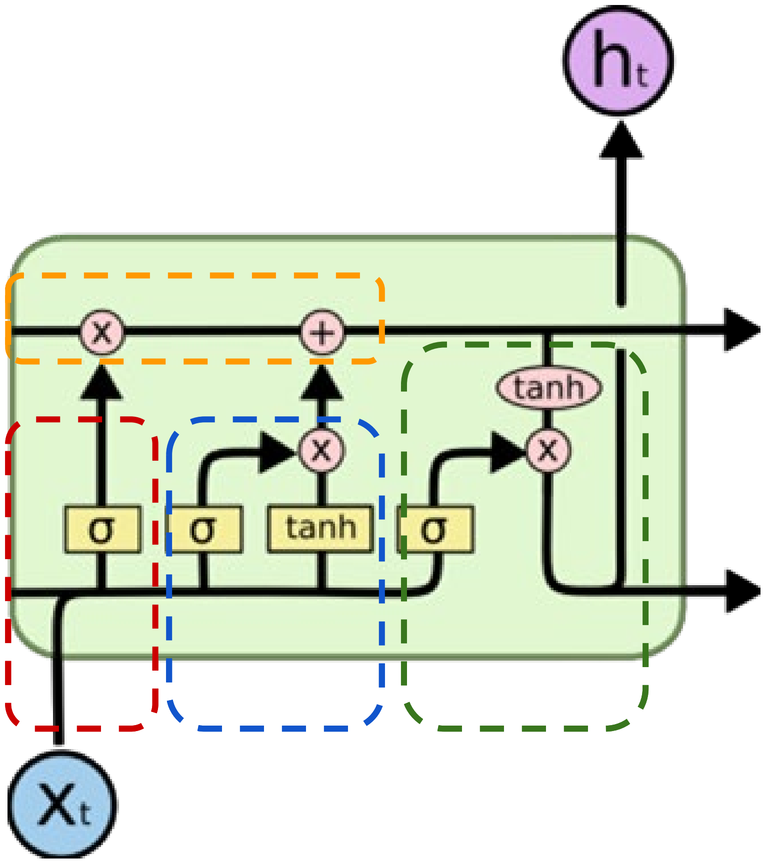
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_{t-1}] + b_f)$$

Input Gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_f)$$

LSTM big picture …

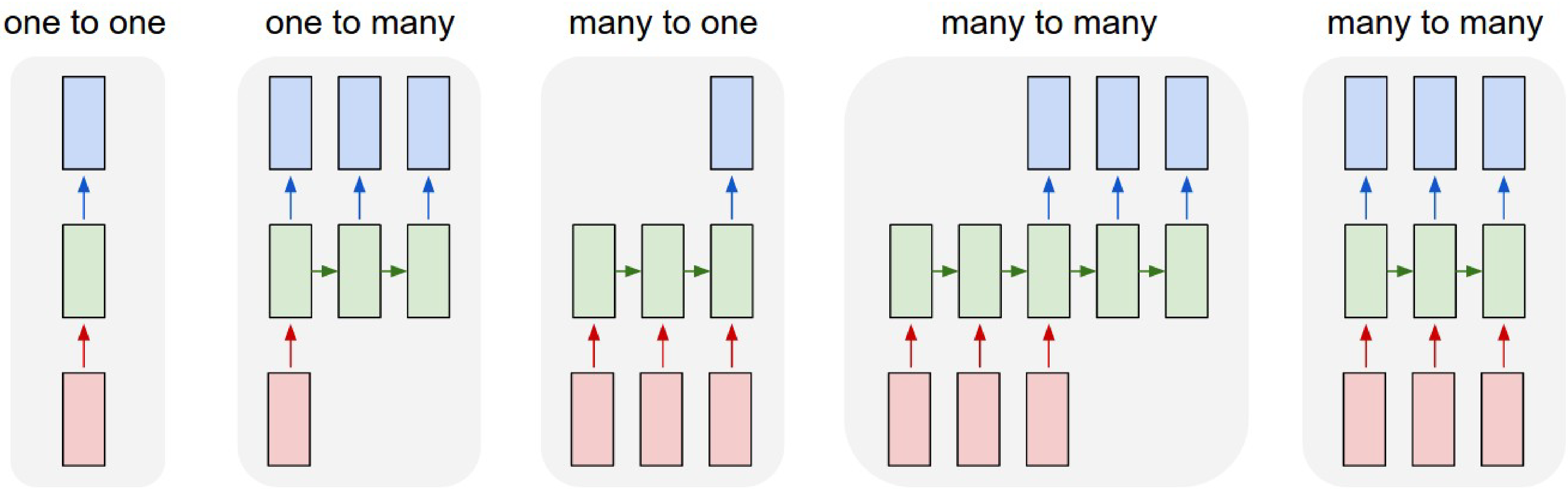


1. LSTM are recurrent neural network with a cell and a hidden state, boths of these are updated in each step and can be thought as memories.

2. Cell states work as a long term memory and the updates depends on the relation between the hidden state in t -1 and the input.

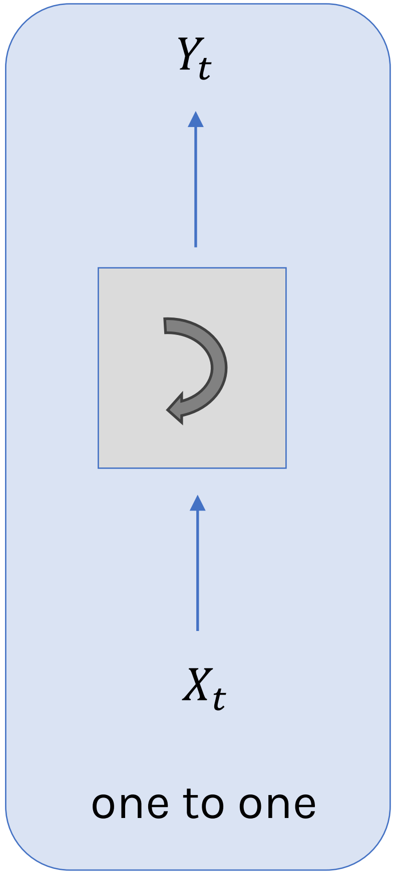
3. The hidden state of the next step is a transformation of the cell state and the output (which is the section that is in general used to calculate our loss, ie information that we want in a short memory).

What makes Recurrent Networks so special?



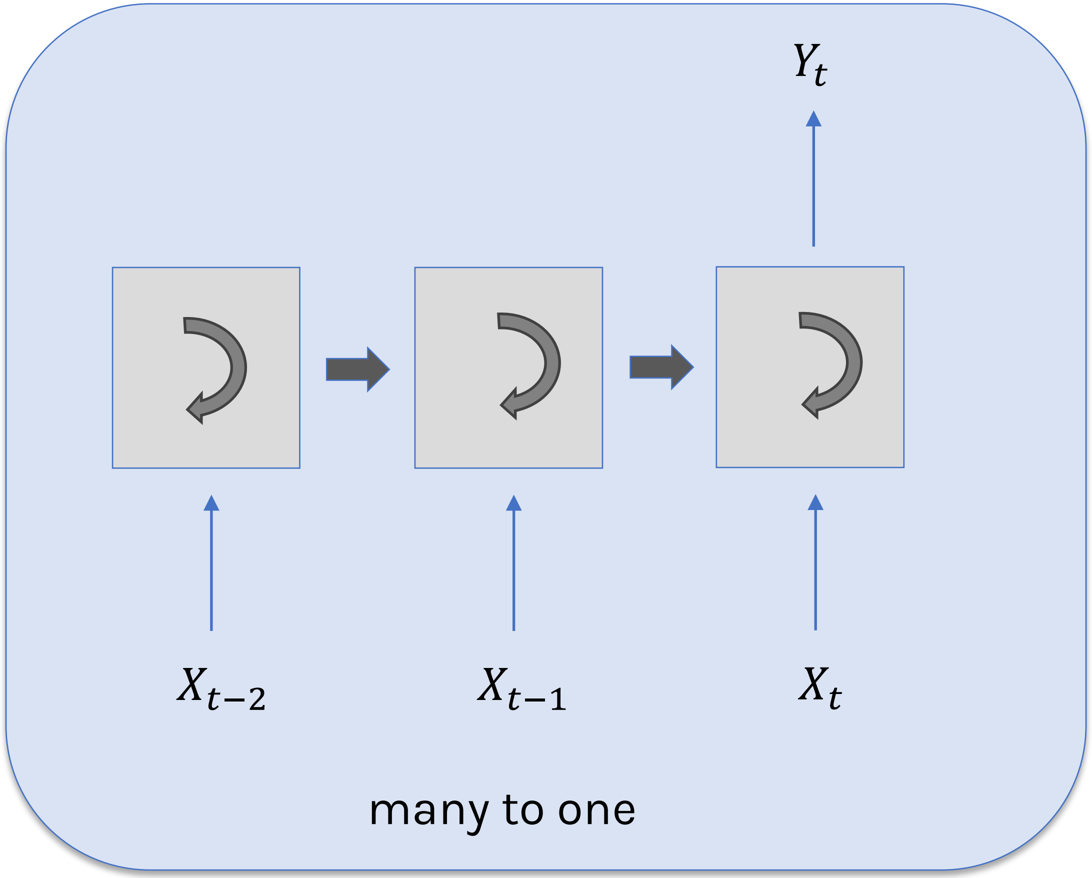
From: http://karpathy.github.io/

RNN Structures



- The **one to one** structure is useless.
- It takes a single input and it produces a single output.
- Not useful because the RNN cell is making little use of its unique ability to remember things about its input sequence

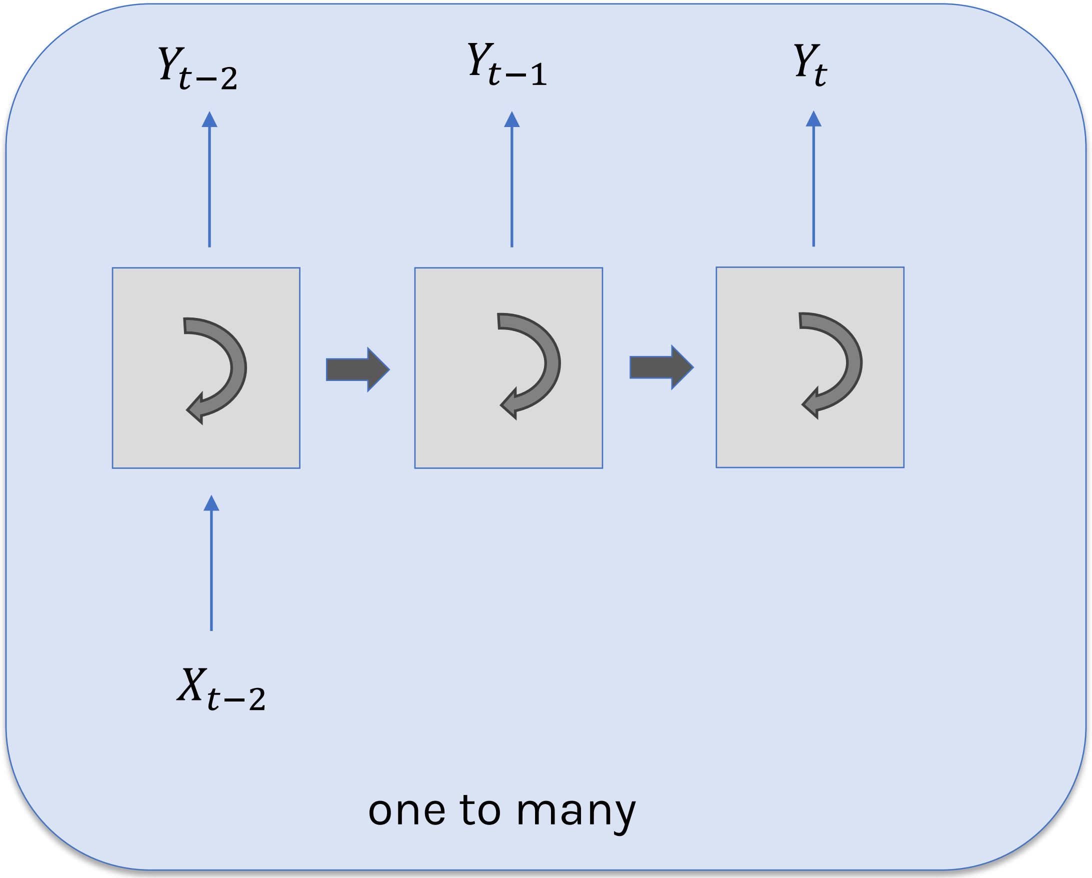
RNN Structures (cont)



The **many to one** structure reads in a sequence and gives us back a single value.

Example: Sentiment analysis, where the network is given a piece of text and then reports on some quality inherent in the writing. A common example is to look at a movie review and determine if it was positive or negative.

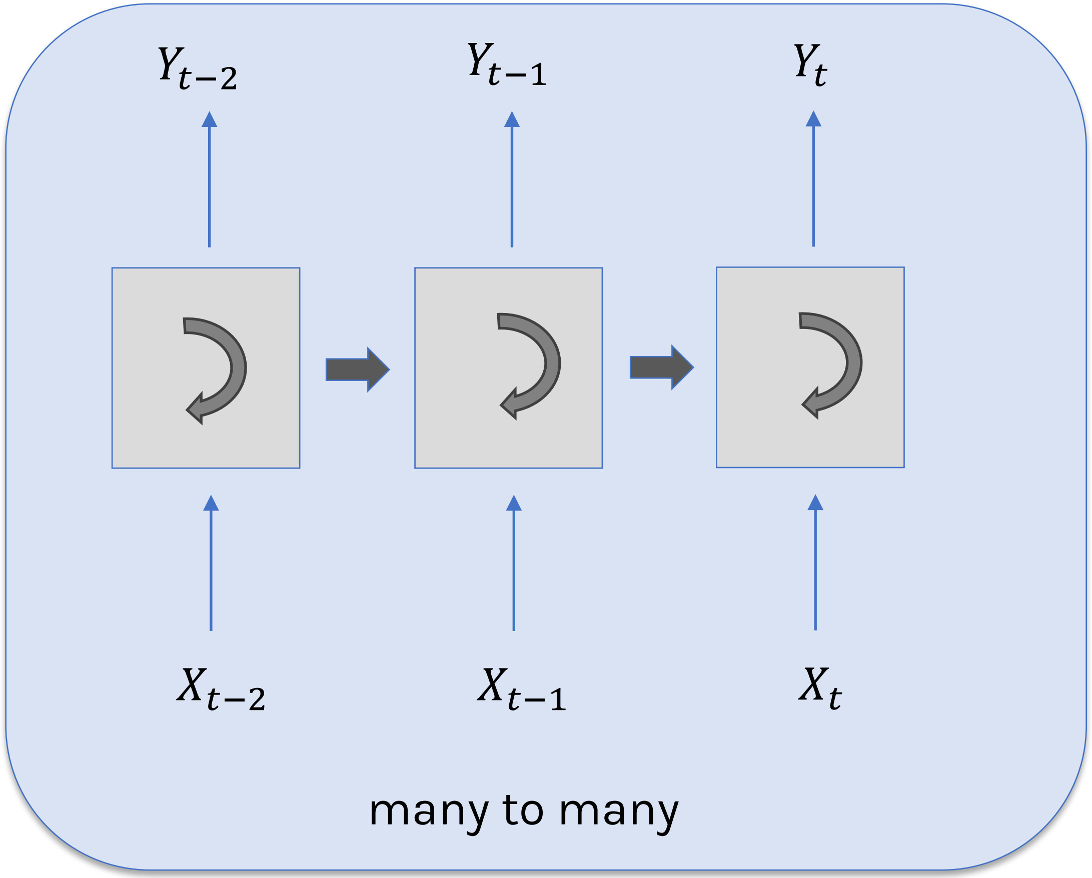
RNN Structures (cont)



The **one to many** takes in a single piece of data and produces a sequence.
For example we give it the starting note for a song, and the network produces the rest of the melody for us.

Generative models

RNN Structures (cont)



The **many to many** structures are in some ways the most interesting. used for machine translation.

Seq2seq models

Language translation

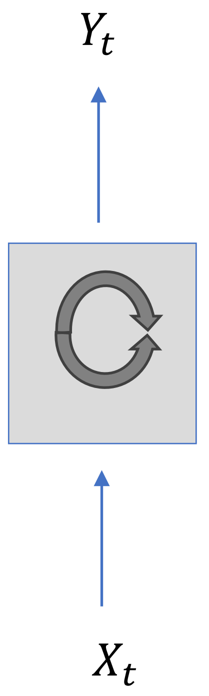
...

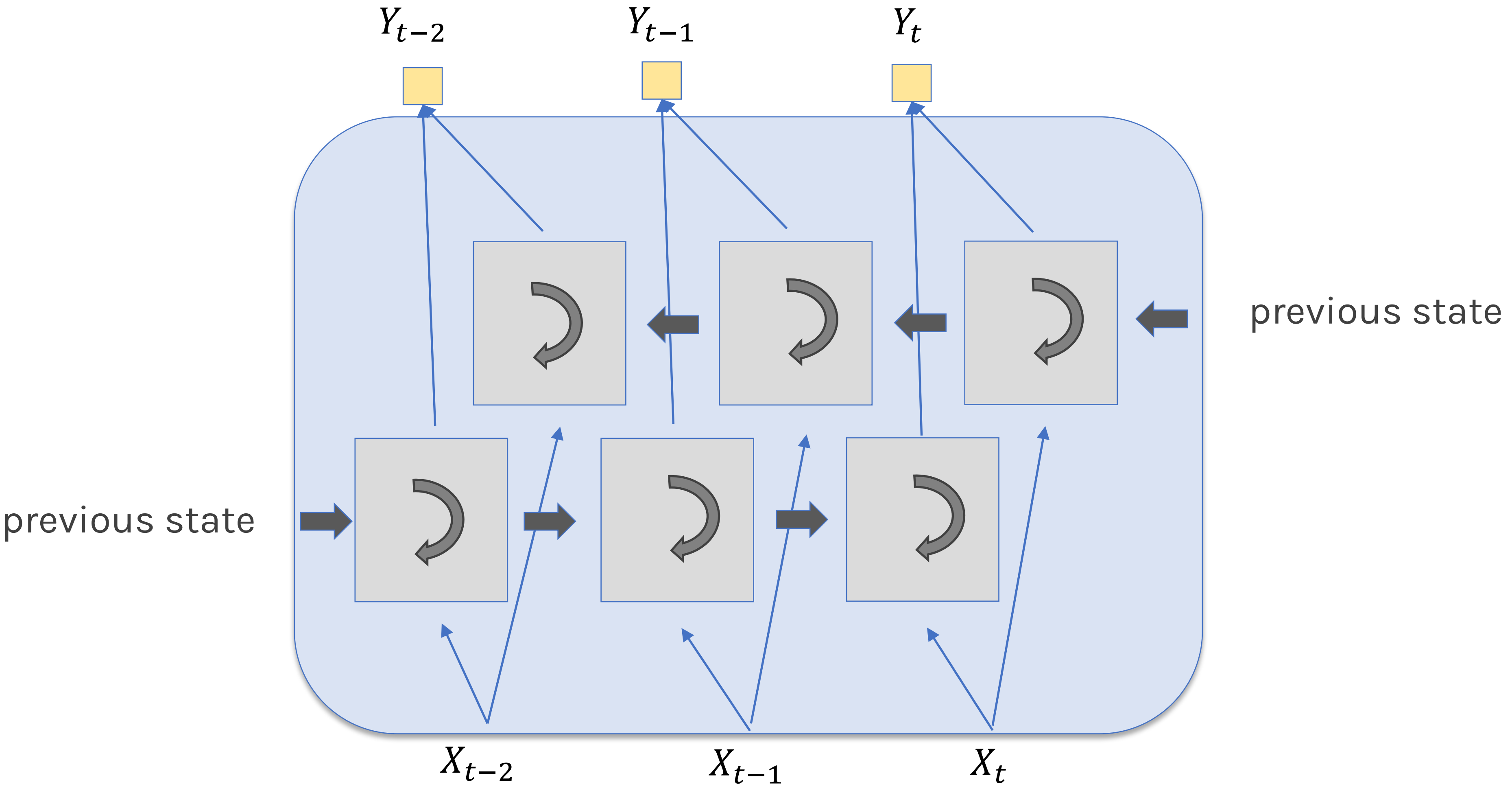
Bidirectional

- LSTM and RNN are designed to analyze sequence of values.
- For example: *Patrick said he needs a vacation.*
- *he* here means *Patrick* and we know this because *Patrick* was before the word *he.*
- However consider the following sentence:
- *He needs to work more, Peter said about Patrick.*
- Bidirectional RNN or BRNN or bidirectional LSTM or BLSTM when using LSTM units.

Bidirectional (cond)

symbol for a BRNN



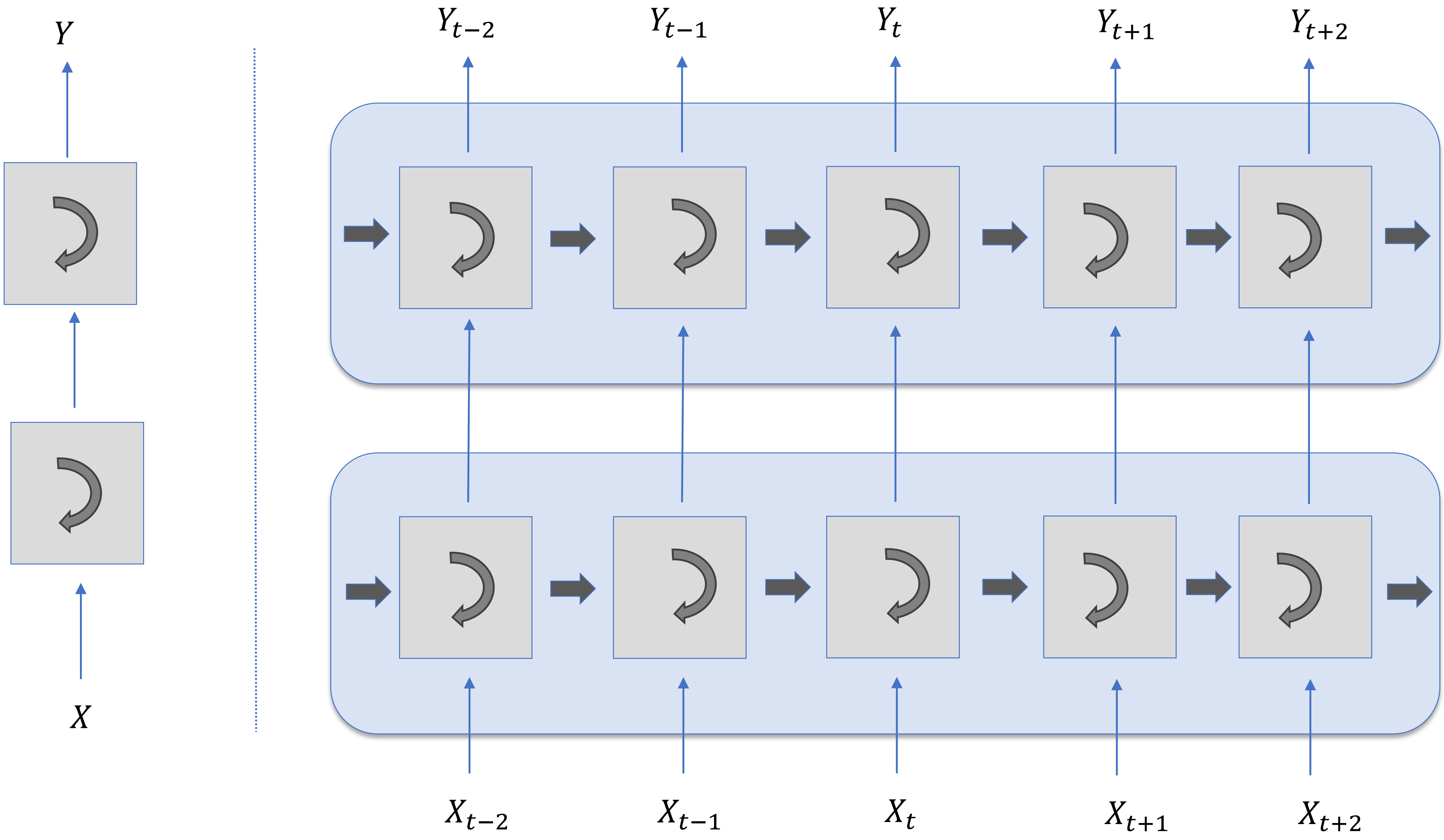


Deep RNN

- LSTM units can be arranged in layers, so that each the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective “deep” refers to these multiple layers.

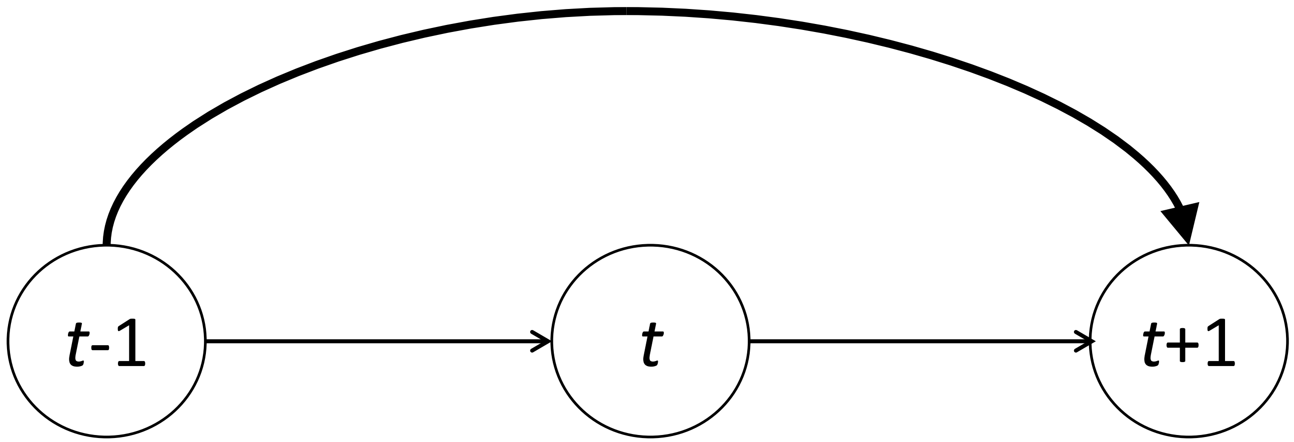
- Each layer feeds the LSTM on the next layer
- First time step of a feature is fed to the first LSTM, which processes that data and produces an output (and a new state for itself).
- That output is fed to the next LSTM, which does the same thing, and the next, and so on.
- Then the second time step arrives at the first LSTM, and the process repeats.

Deep RNN



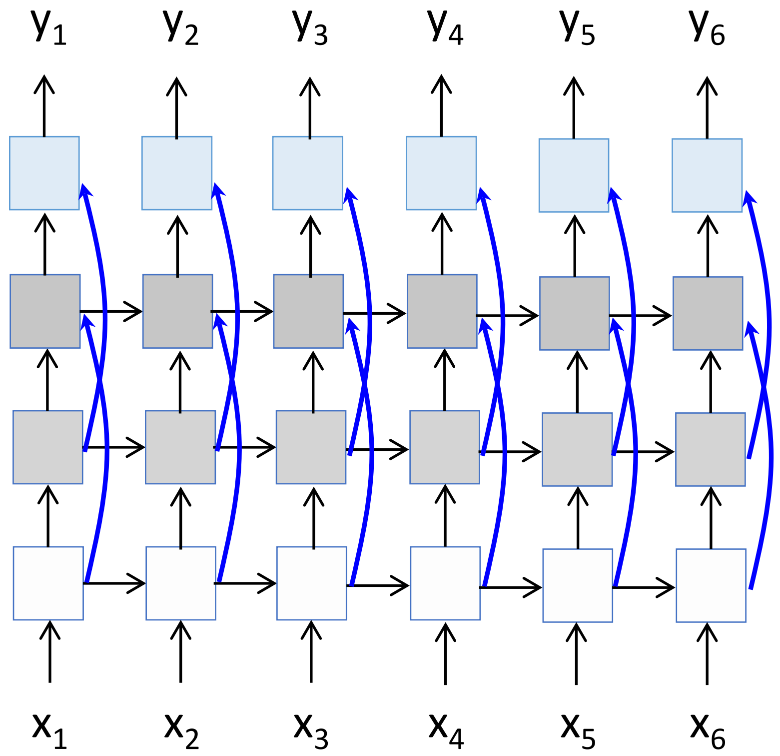
Skip Connections

- Add additional connections between units *d* time steps apart
- Creating paths through time where gradients neither vanish or explode



Multi-layer RNNs

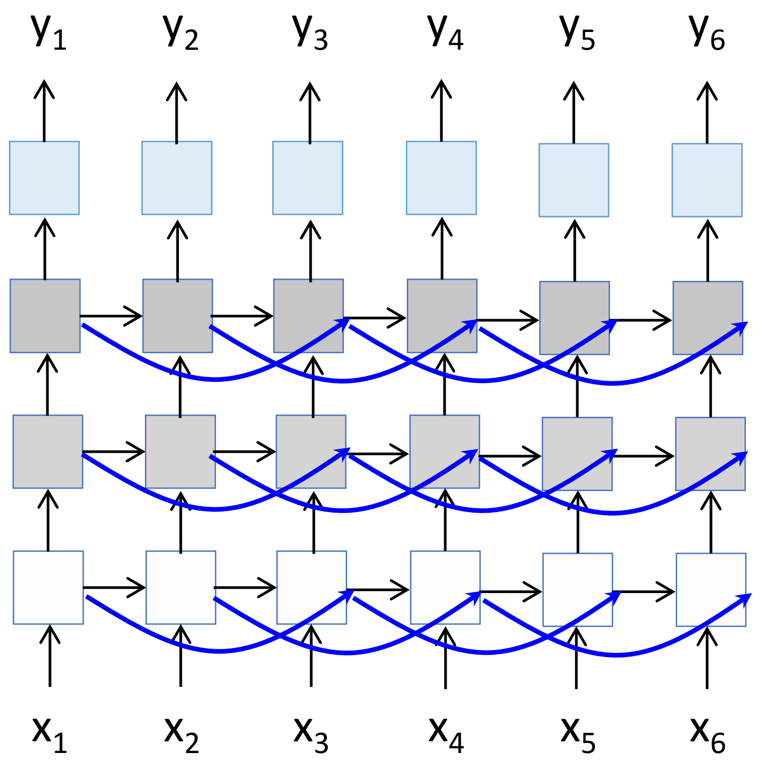
- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Multi-layer RNNs

- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Language modeling: Character RNN

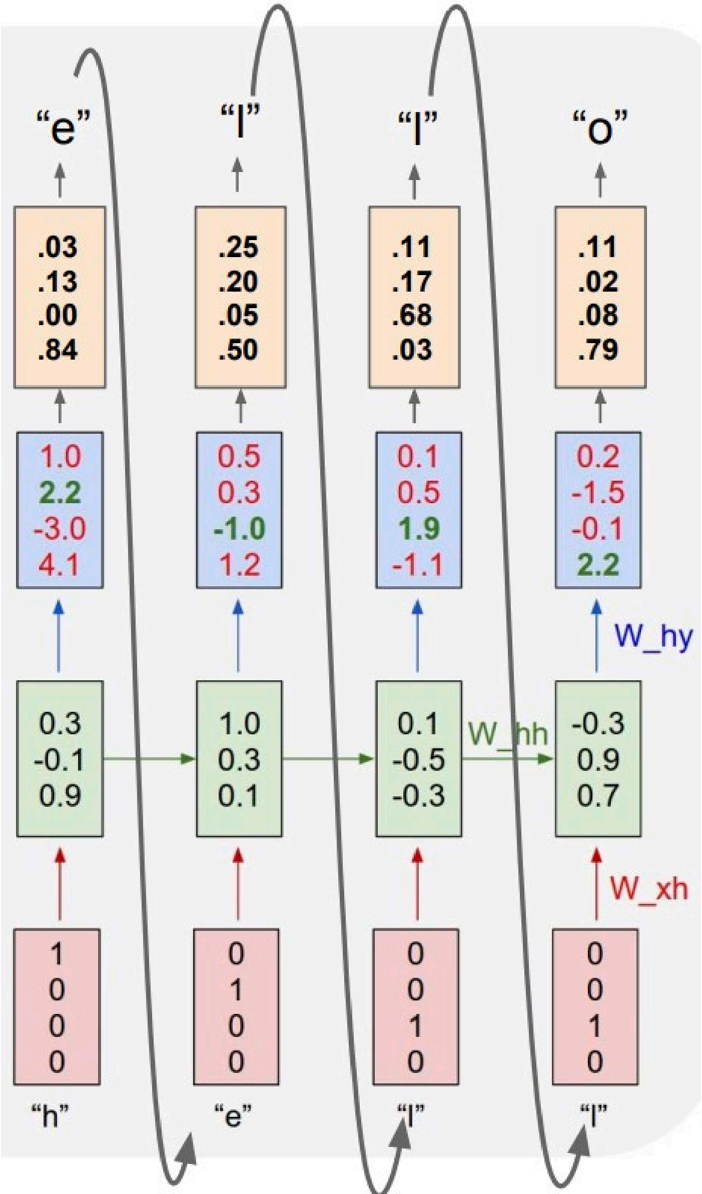
Output symbol y_i

Output layer
(linear + softmax)

Hidden state h_i

One-hot encoding x_i

Input symbol



$$p(y_1, y_2, \dots, y_n)$$
$$= \prod_{i=1}^{n} p(y_i | y_1, \dots, y_{i-1})$$
$$\approx \prod_{i=1}^{n} P_W(y_i | h_i)$$

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

Language modeling: Character RNN

100th iteratio n

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

300th iteration

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

700th iteration

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

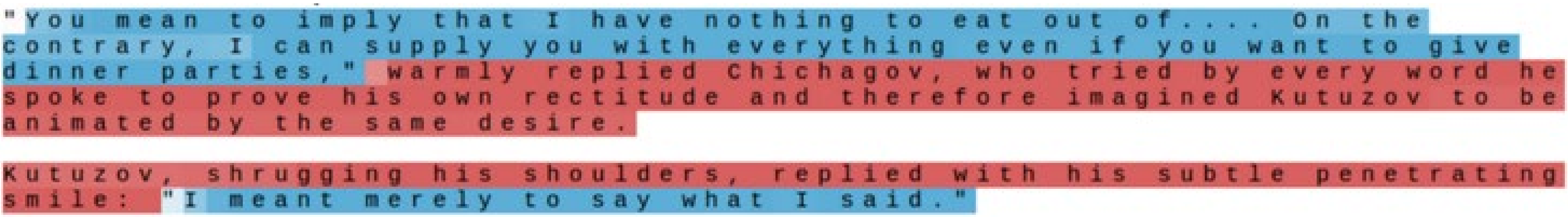
2000th

iteration

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

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

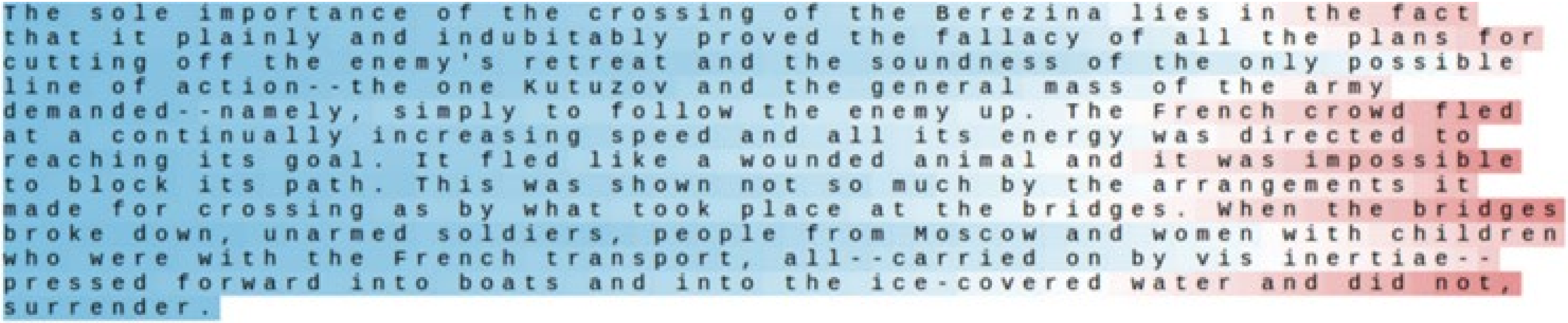
Searching for interpretable hidden units



quote detection cell

A. Karpathy, J. Johnson, and L. Fei-Fei, [Visualizing and Understanding Recurrent Networks](), ICLR Workshop 2016

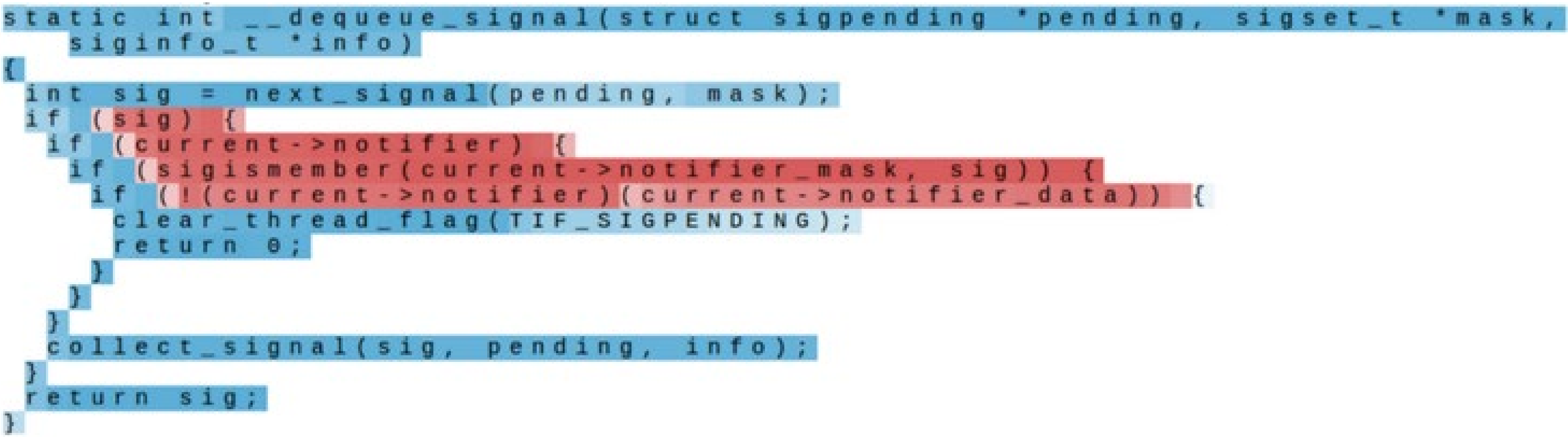
Searching for interpretable hidden units



line position tracking cell

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

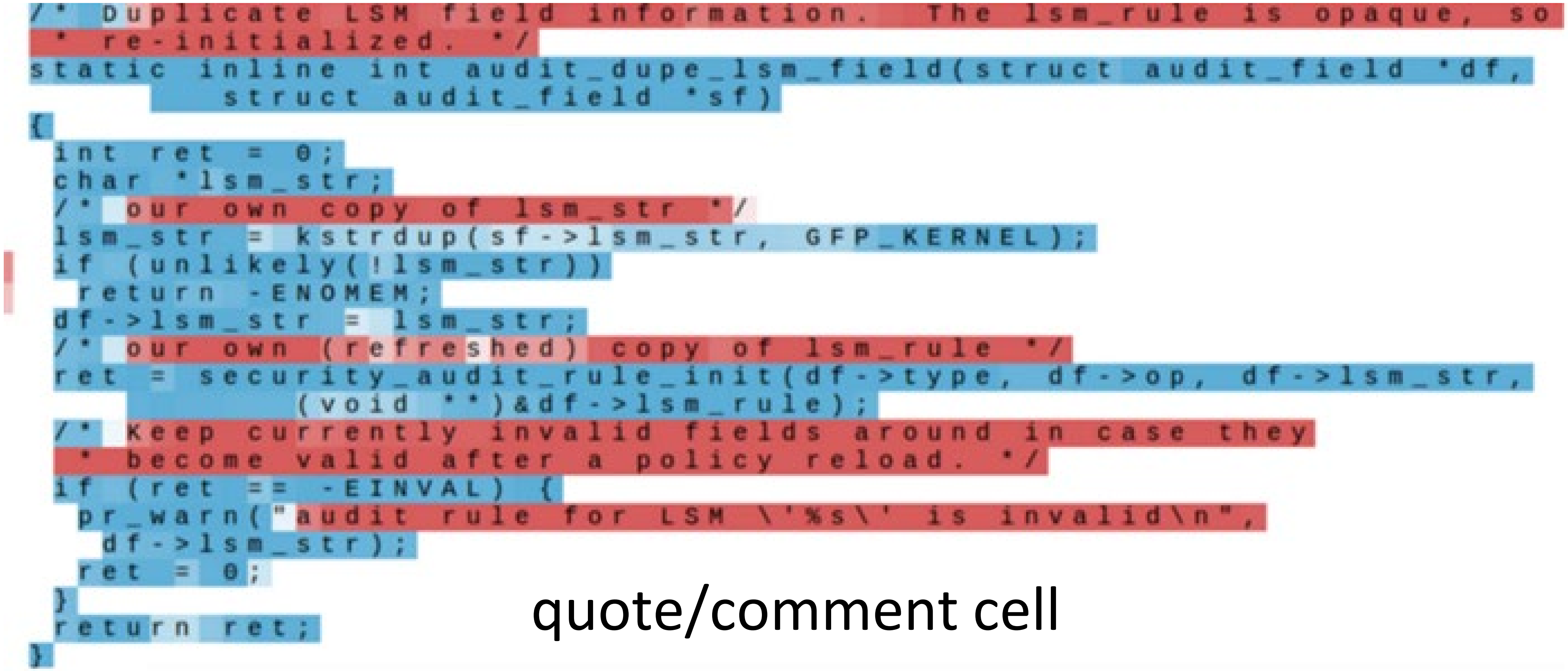
Searching for interpretable hidden units



if statement cell

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

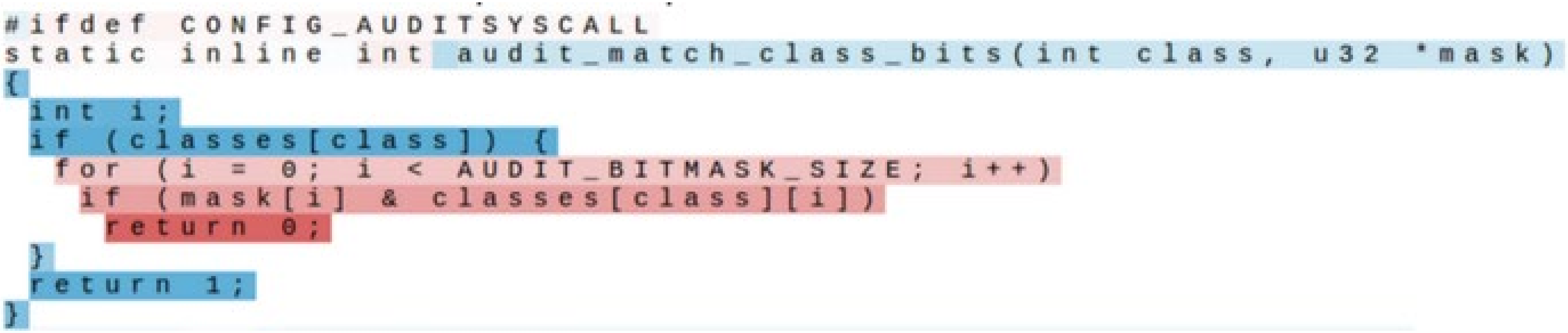
Searching for interpretable hidden units



quote/comment cell

A. Karpathy, J. Johnson, and L. Fei-Fei, [Visualizing and Understanding Recurrent Networks](), ICLR Workshop 2016

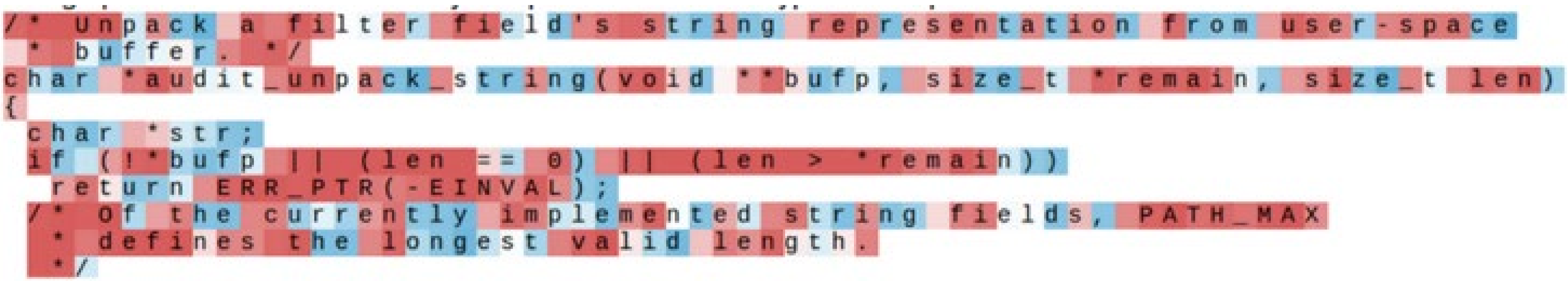
Searching for interpretable hidden units



code depth cell

A. Karpathy, J. Johnson, and L. Fei-Fei, [Visualizing and Understanding Recurrent Networks](), ICLR Workshop 2016

Searching for interpretable hidden units

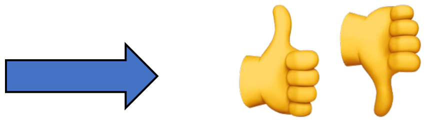


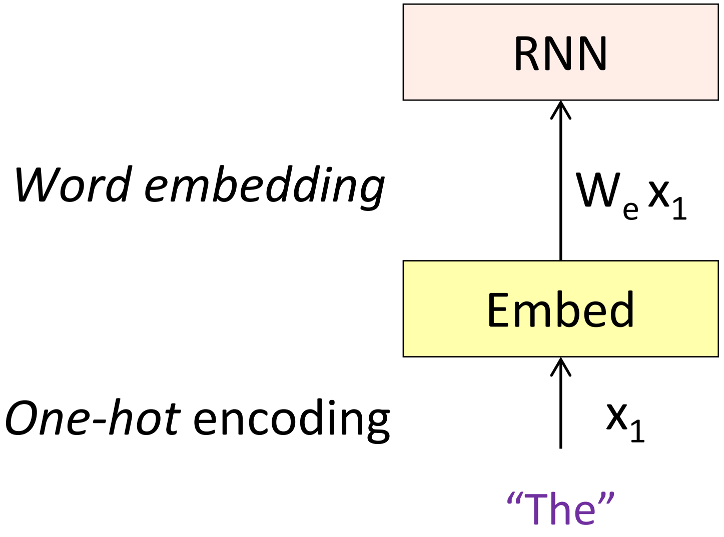
¯_(ツ)_/¯

A. Karpathy, J. Johnson, and L. Fei-Fei, [Visualizing and Understanding Recurrent Networks](), ICLR Workshop 2016

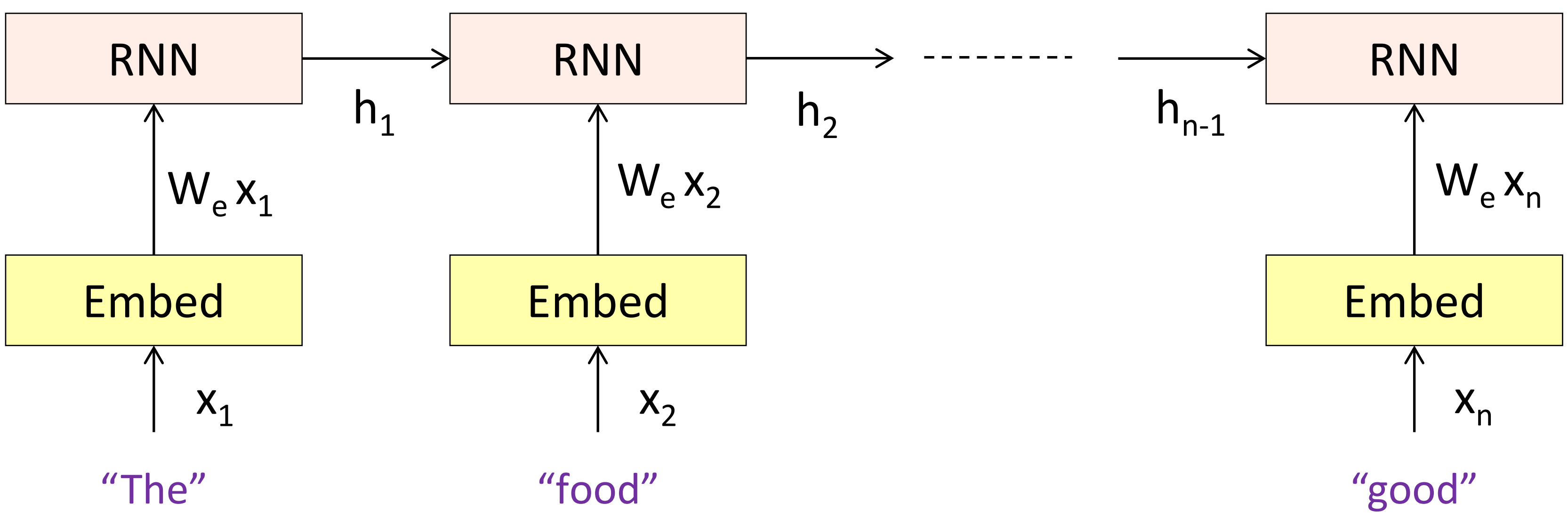
Sequence classification

"The food is usually not so good"

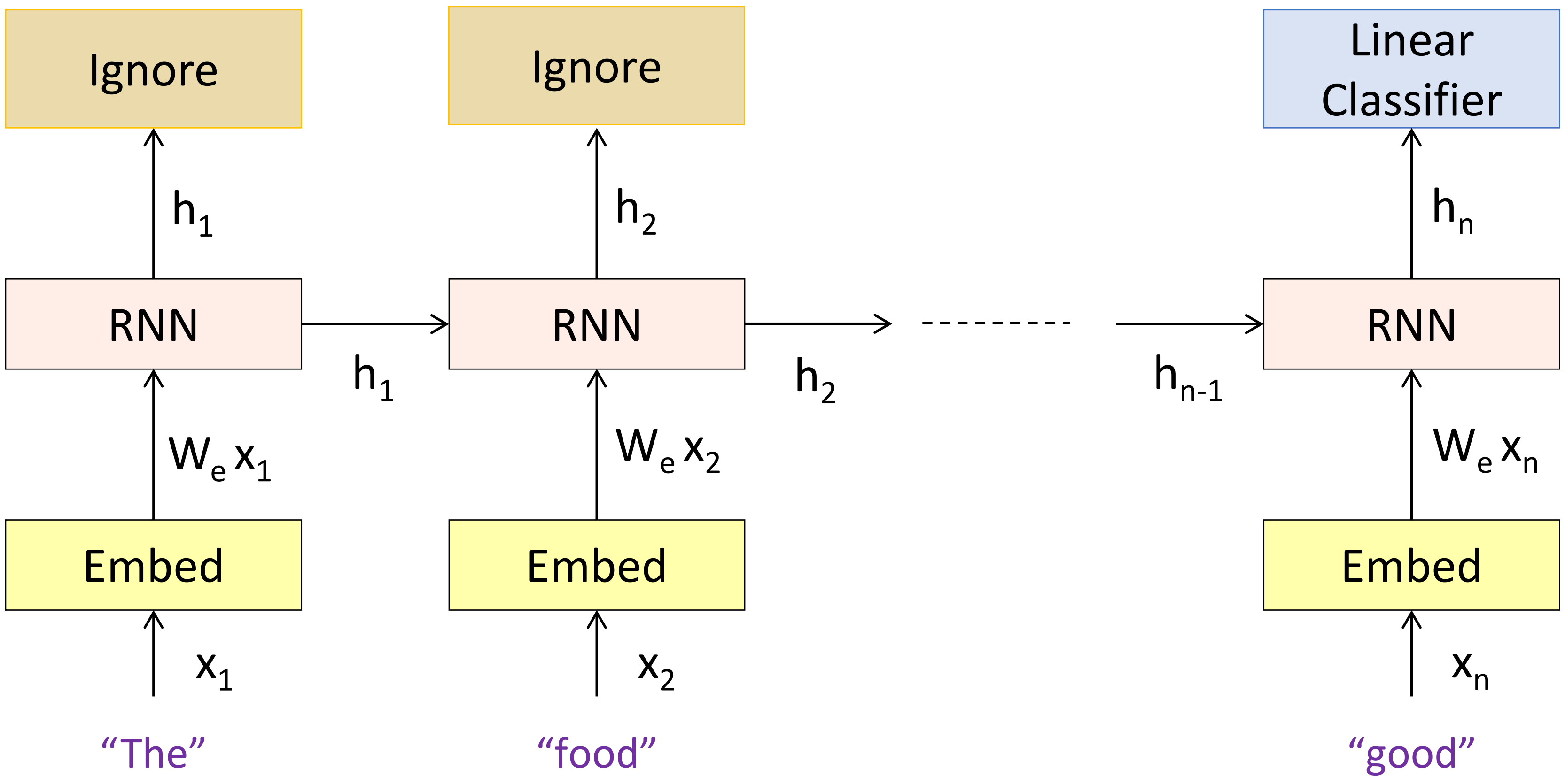




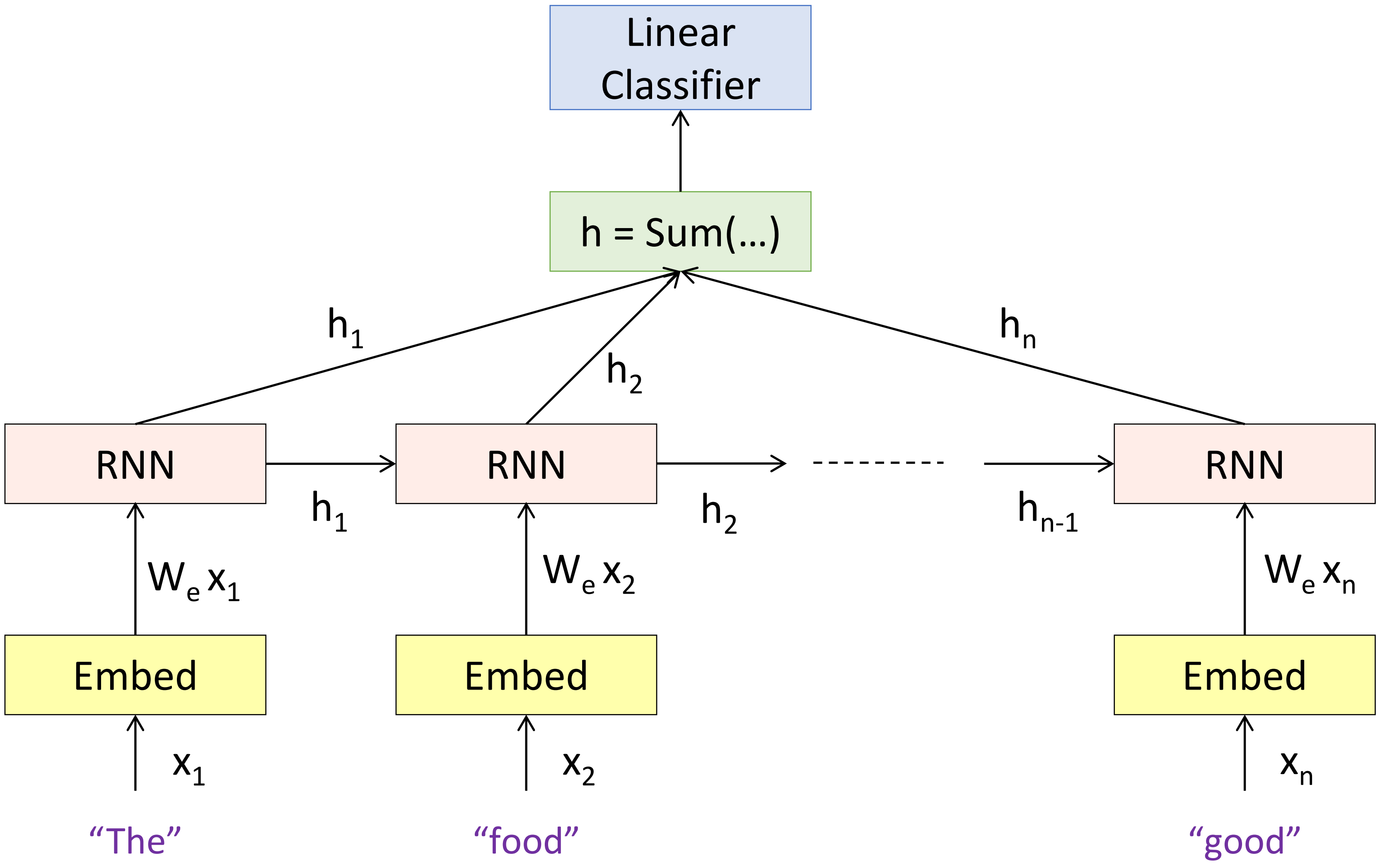
Sequence classification



Sequence classification



Sequence classification



http://deeplearning.net/tutorial/lstm.html

Sequence classification

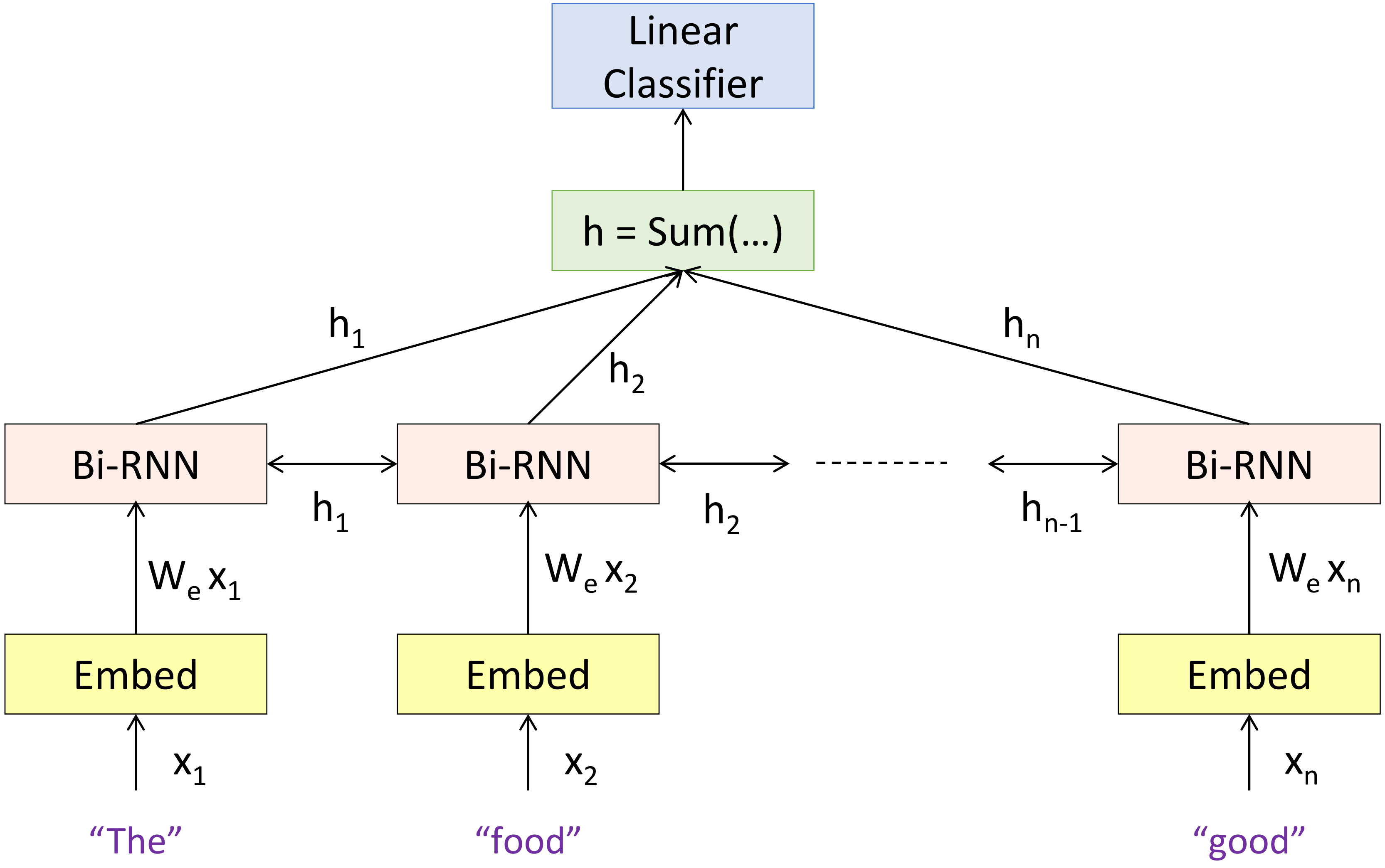
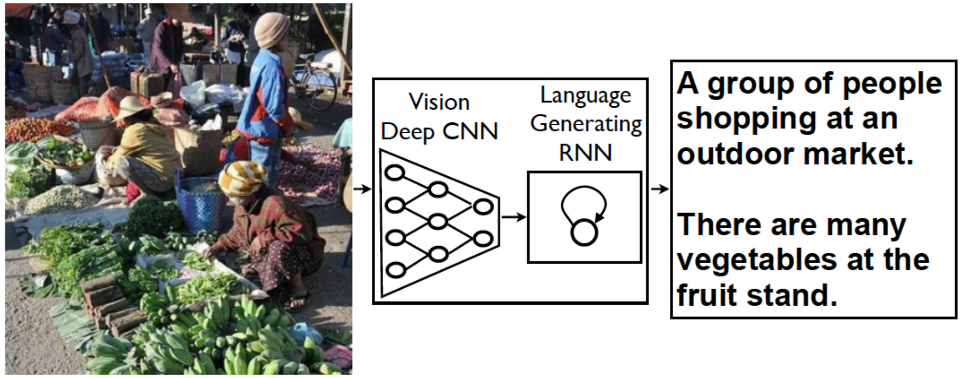
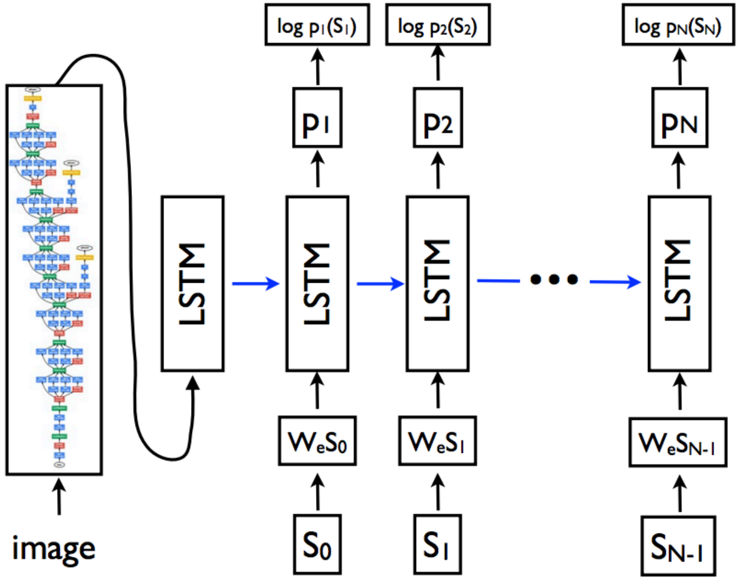


Image caption generation



Training time

- Maximize likelihood of reference captions



Log-likelihood of next reference word

Softmax probability over next word

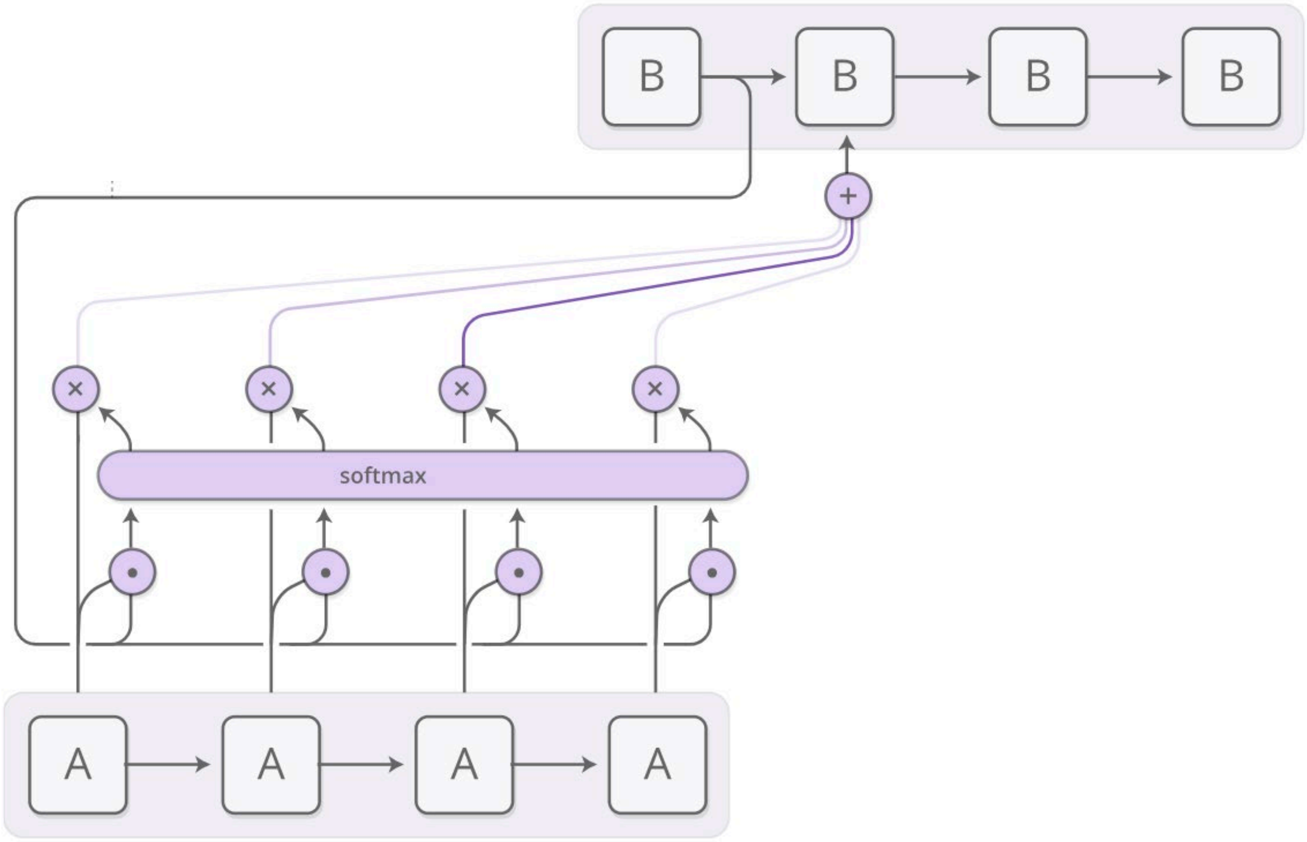
Word embedding

Words of reference caption (one-hot encoding)

O. Vinyals, A. Toshev, S. Bengio, D. Erhan, [Show and Tell: A Neural Image Caption Generator](), CVPR 2015

Image caption generation: Example outputs

Sequence-to-sequence models with attention



Many slides adapted from [J. Johnson]

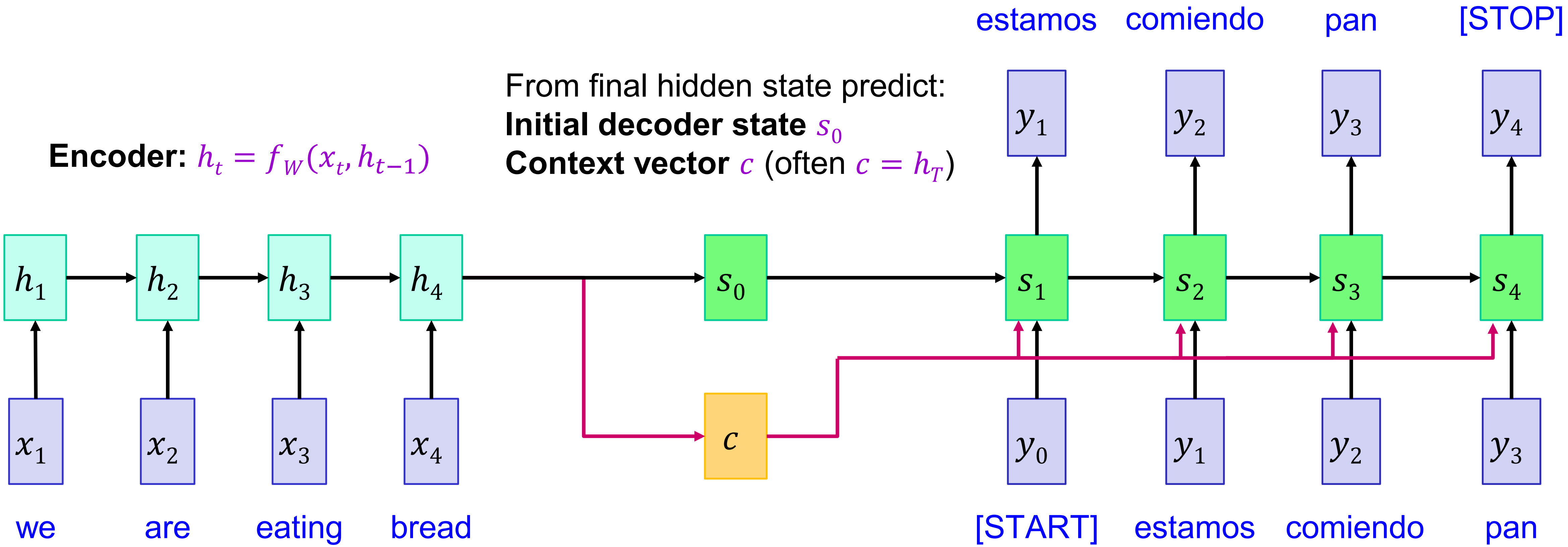
“We are eating bread” ➡ “Estamos comiendo pan”

Sequence-to-sequence with RNNs

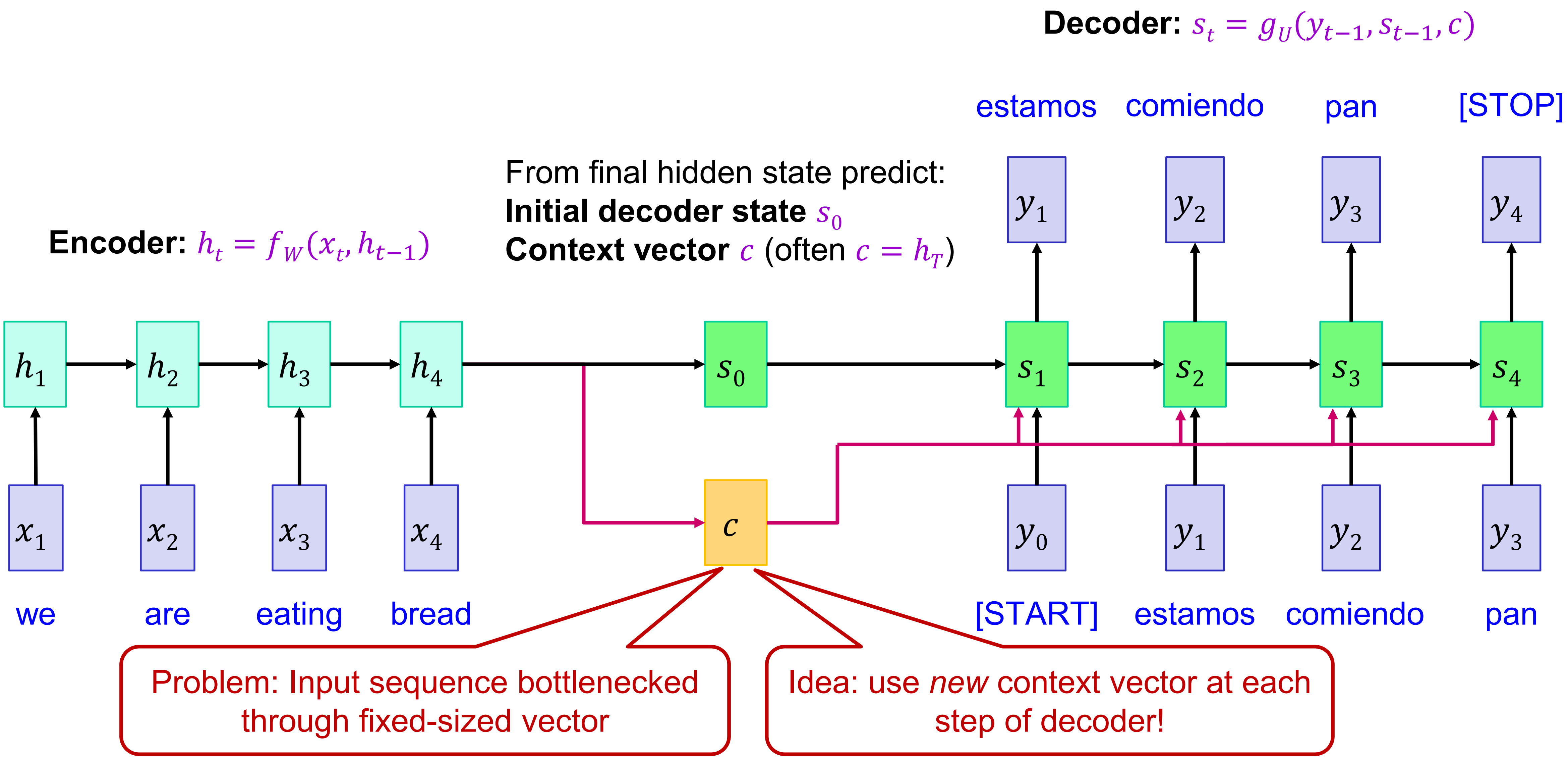
Decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c)$

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c = h_T$)

Encoder: $h_t = f_W(x_t, h_{t-1})$



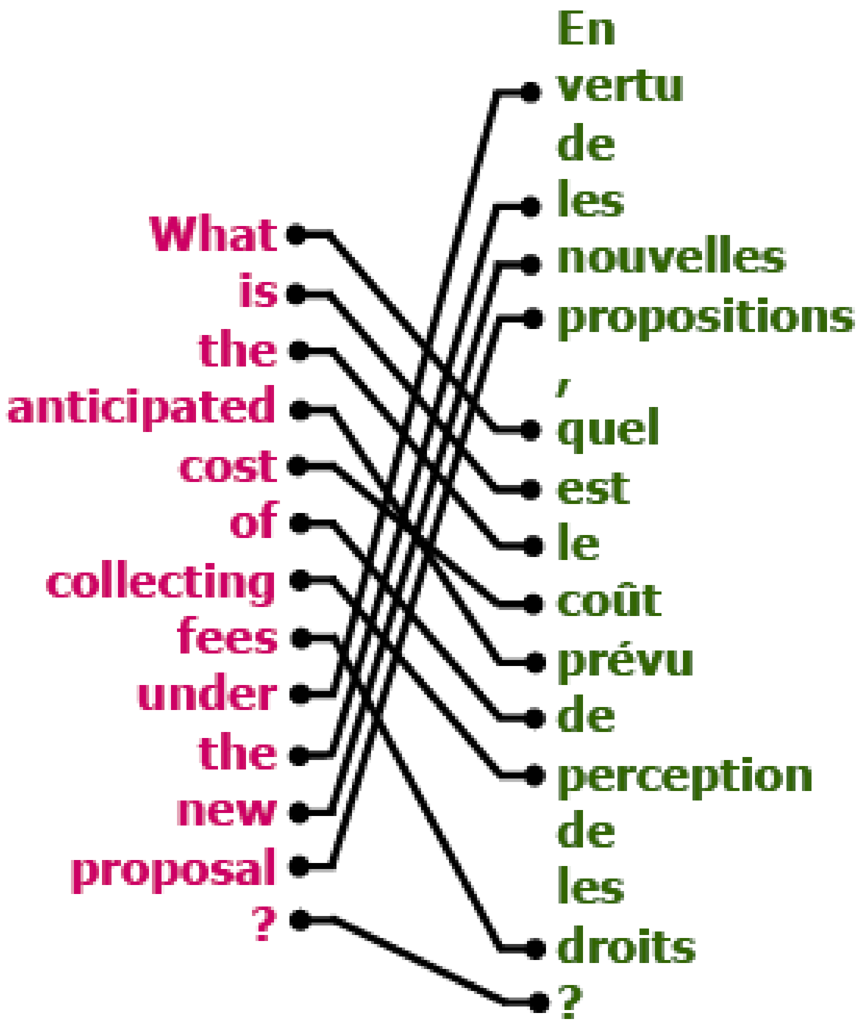
Sequence-to-sequence with RNNs



A. Sutskever, O. Vinyals, Q. Le, Sequence to sequence learning with neural networks, NeurIPS 2014

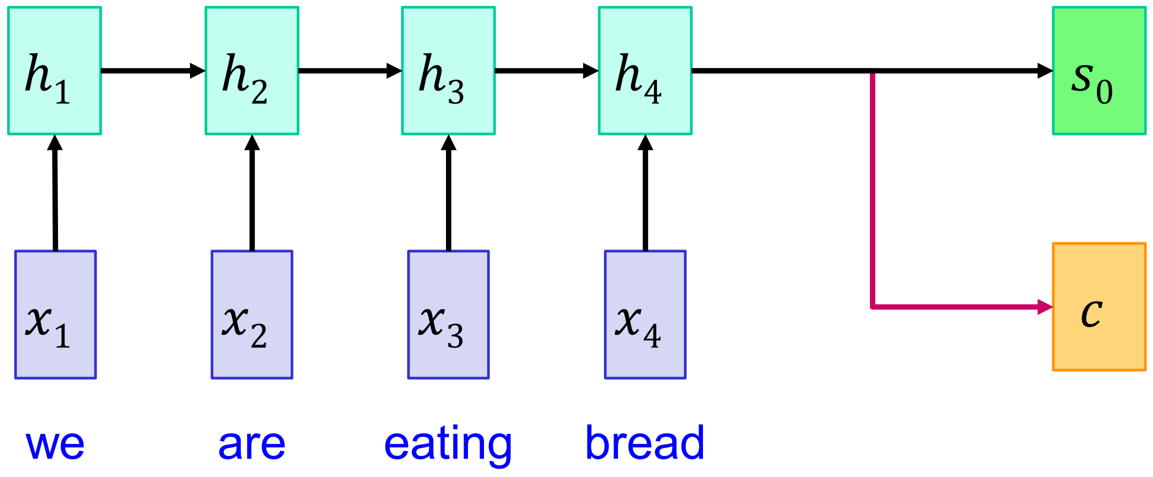
Sequence-to-sequence with RNNs *and attention*

- Intuition: translation requires *alignment*

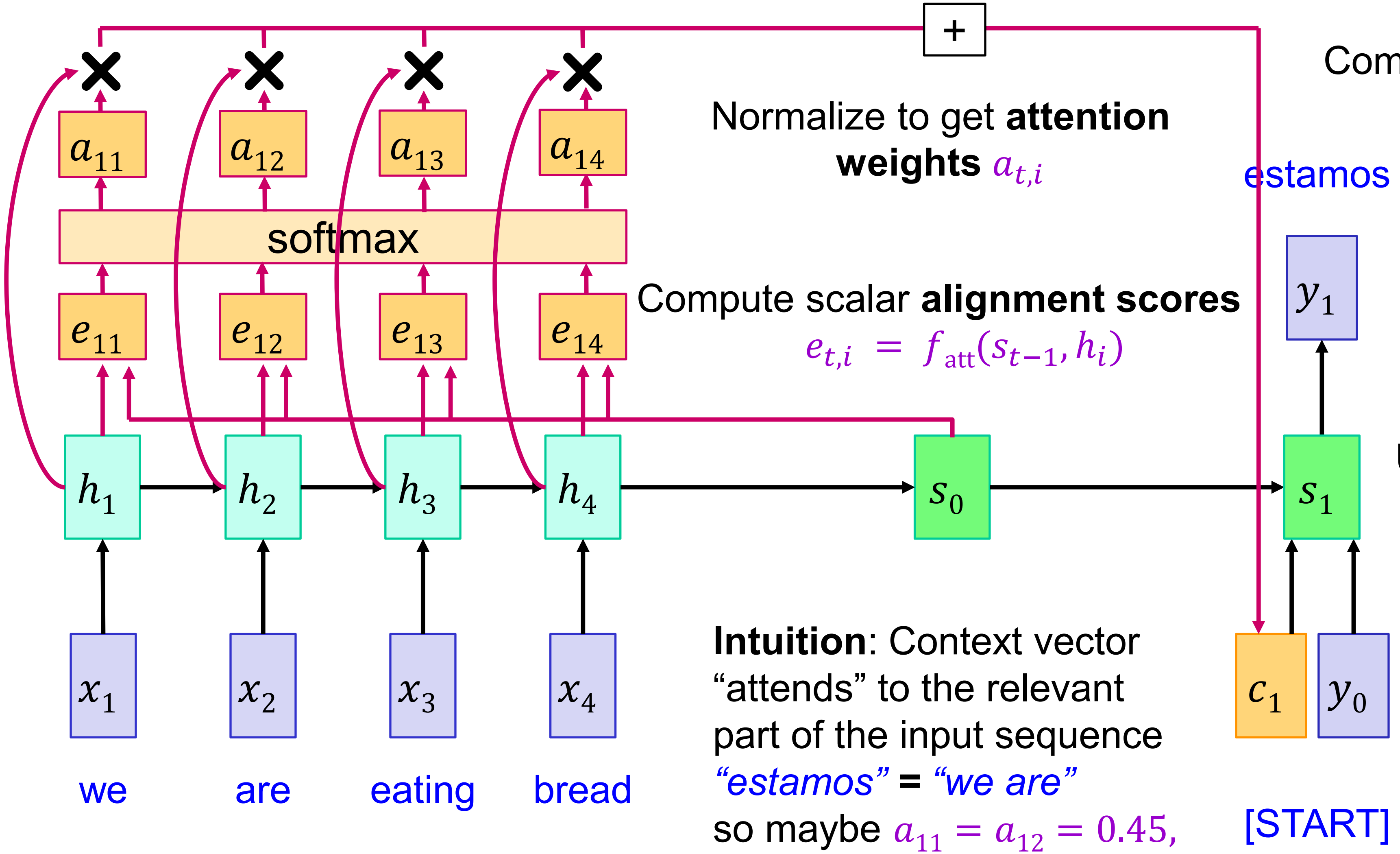


Sequence-to-sequence with RNNs and attention

- At each timestep of decoder, context vector "looks at" different parts of the input sequence



Sequence-to-sequence with RNNs and attention



Compute **context vector** as

$$c_t = \sum_i a_{t,i} h_i$$

Normalize to get **attention weights** $a_{t,i}$

Compute scalar **alignment scores**

$$e_{t,i} = f_{\text{att}}(s_{t-1}, h_i)$$

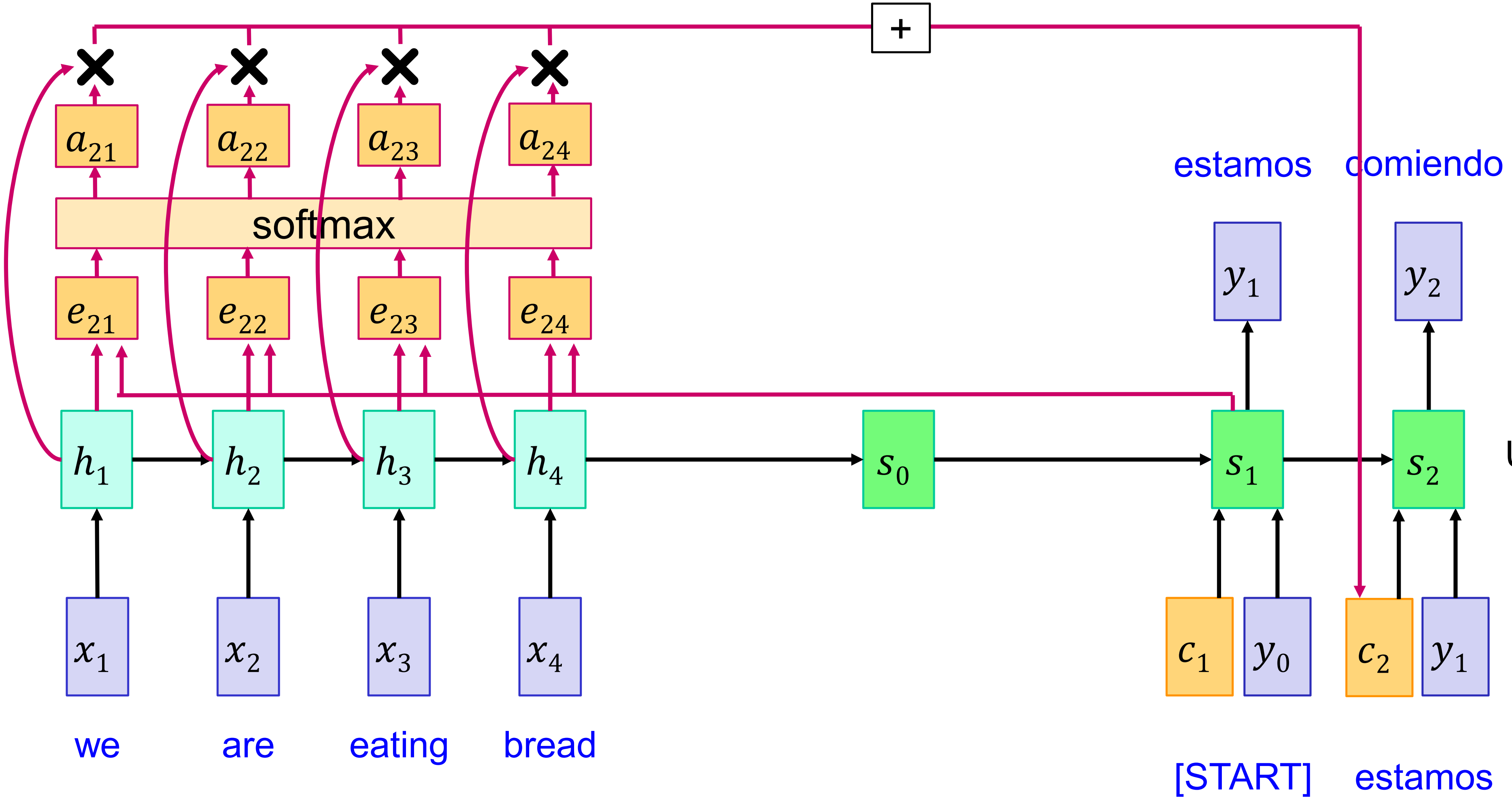
Use context vector in decoder:

$$s_t = g_U(y_{t-1}, s_{t-1}, c_t)$$

Intuition: Context vector "attends" to the relevant part of the input sequence
"estamos" **=** *"we are"*
so maybe $a_{11} = a_{12} = 0.45$, $a_{13} = a_{14} = 0.05$

D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015

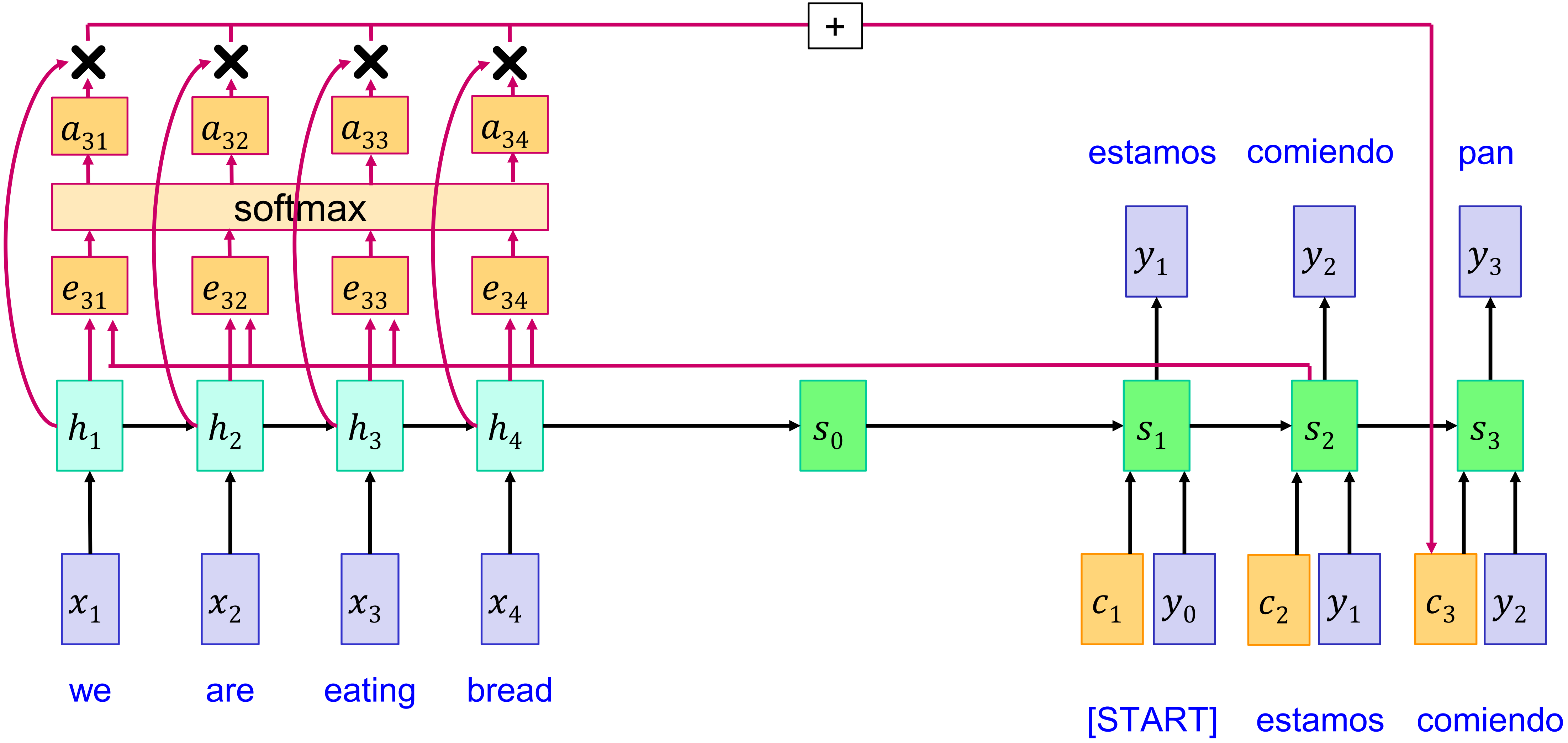
Sequence-to-sequence with RNNs and attention



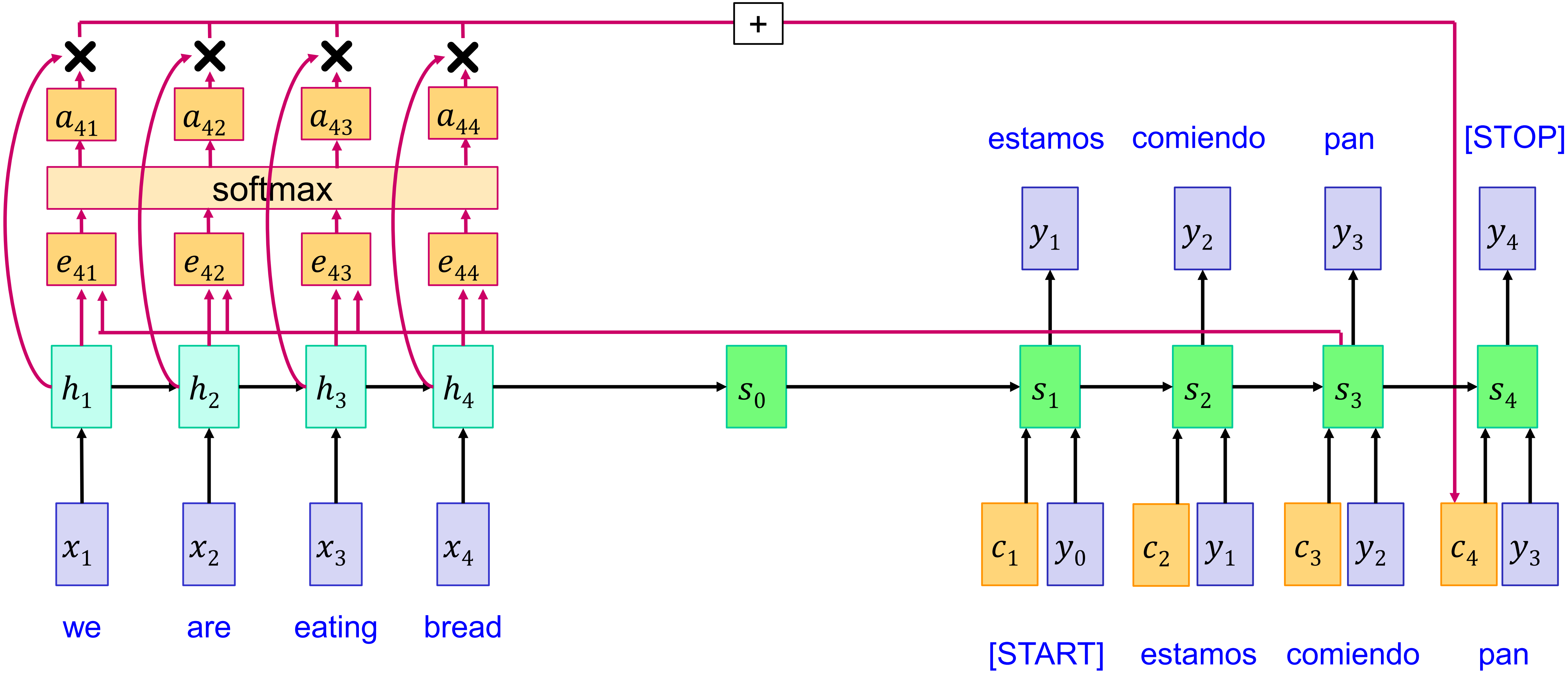
Use c_2 to compute s_2, y_2

Repeat: Use s_1 to compute new context vector c_2

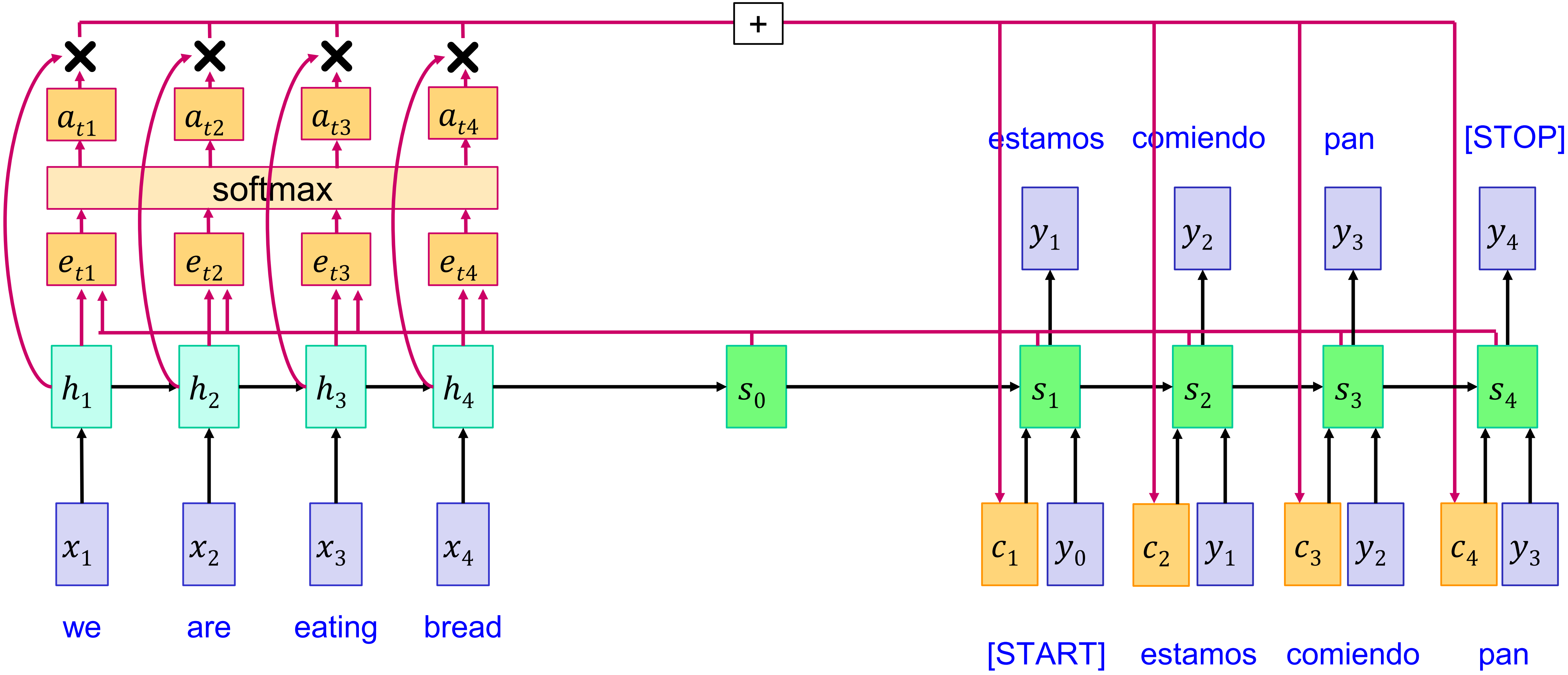
Sequence-to-sequence with RNNs and attention



Sequence-to-sequence with RNNs and attention

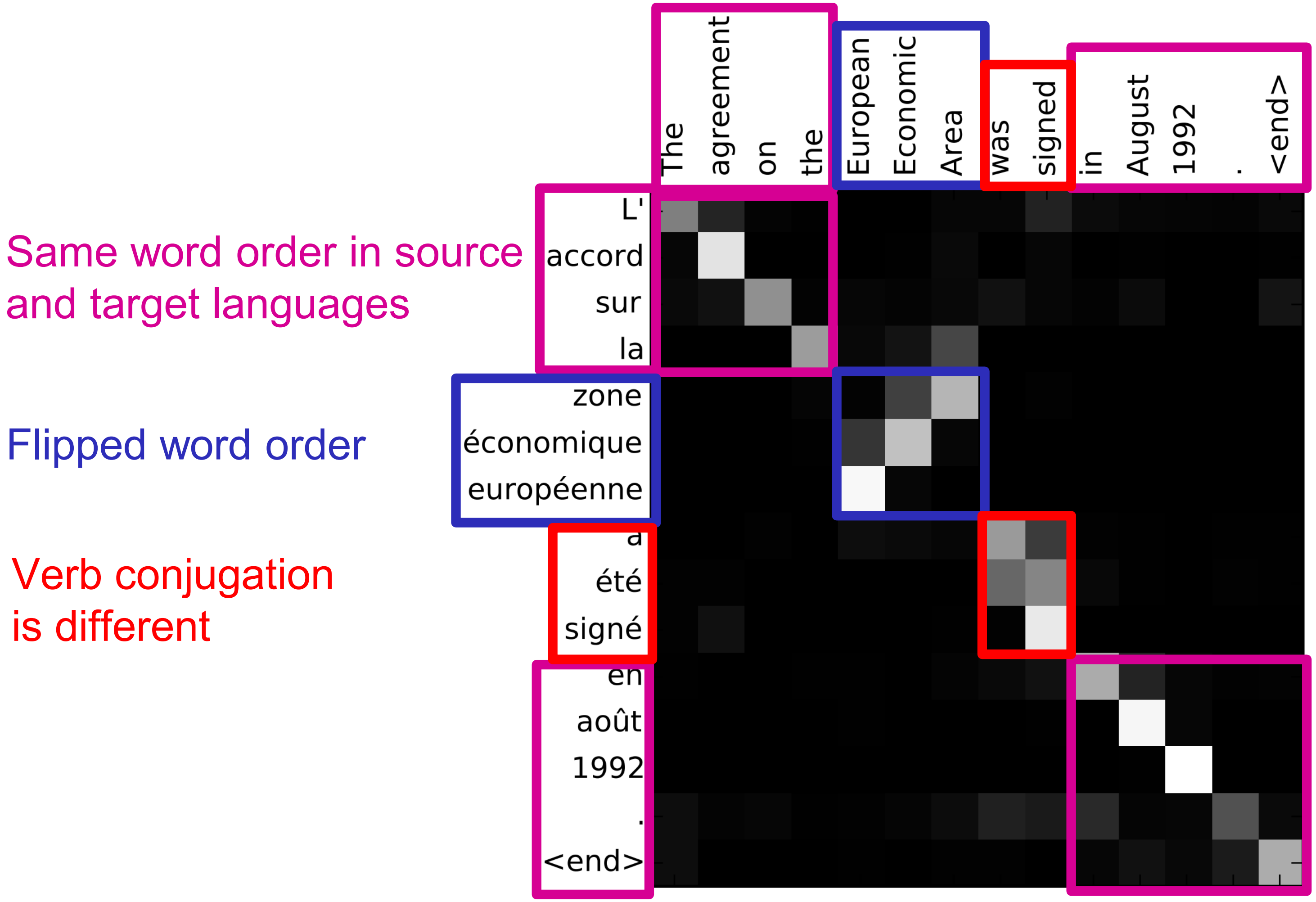


Sequence-to-sequence with RNNs and attention



Sequence-to-sequence with RNNs and attention

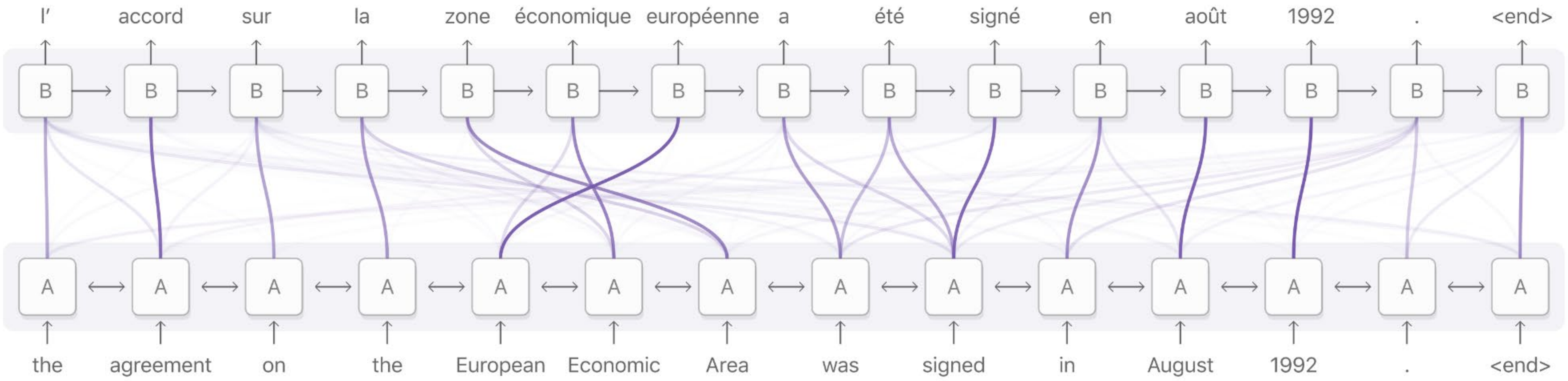
- Visualizing attention weights (English source, French target):



D. Bahdanau, K. Cho, Y. Bengio, [Neural Machine Translation by Jointly Learning to Align and Translate](), ICLR 2015

Attention: Weights Visualization

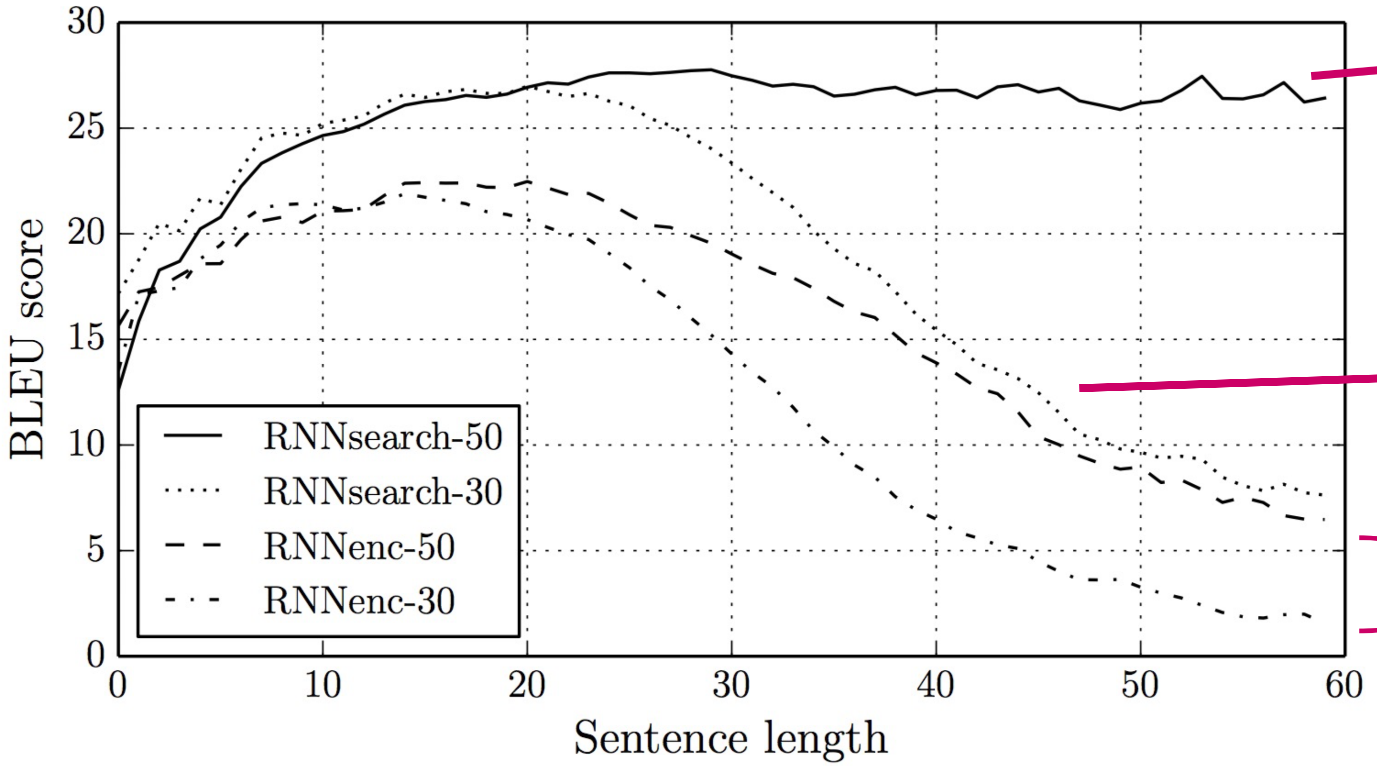
Decoder RNN (target language: French)



Encoder RNN (source language: English)

Figure is from https://distill.pub/2016/augmented-rnns/

Quantitative evaluation



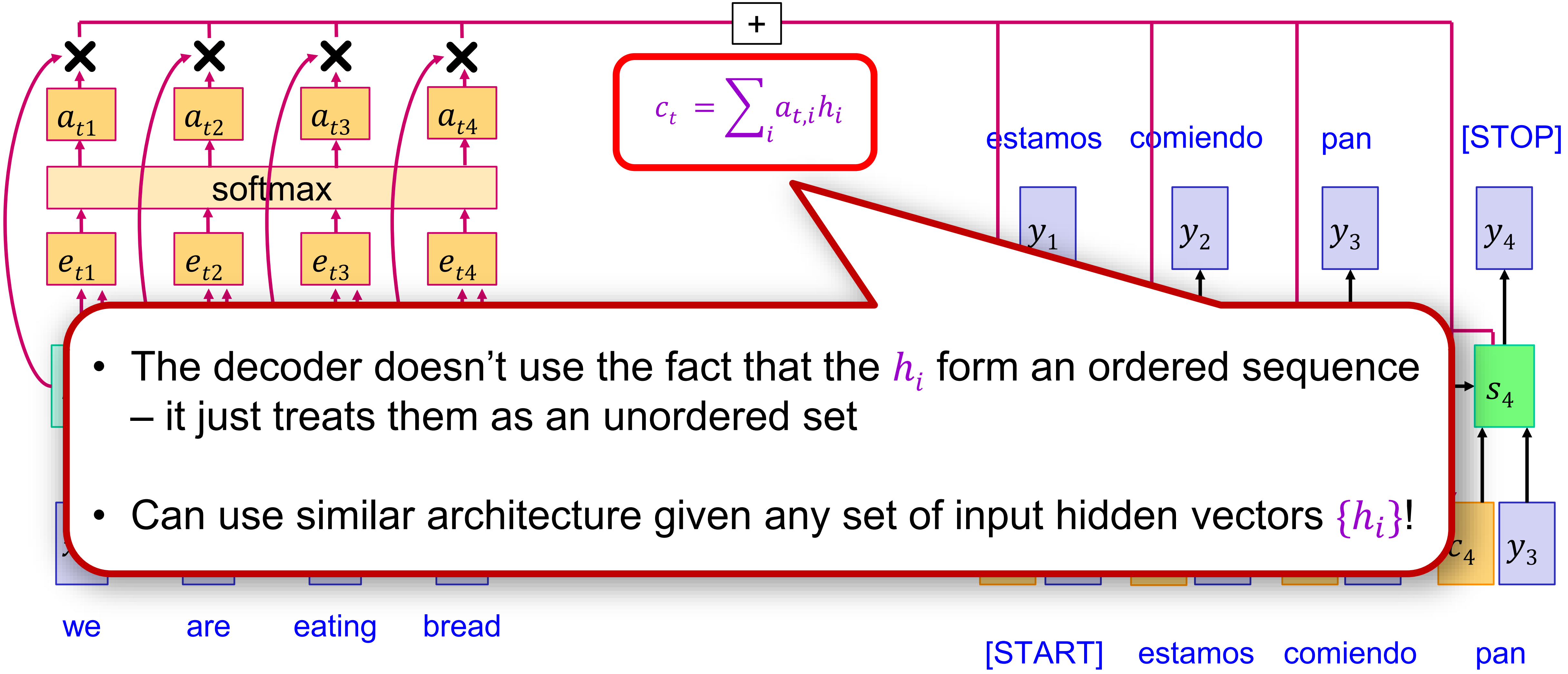
With attention (trained with sentence length <= 50)

With attention (trained with sentence length <= 30)

No attention

D. Bahdanau, K. Cho, Y. Bengio, [Neural Machine Translation by Jointly Learning to Align and Translate](), ICLR 2015

Generalizing attention

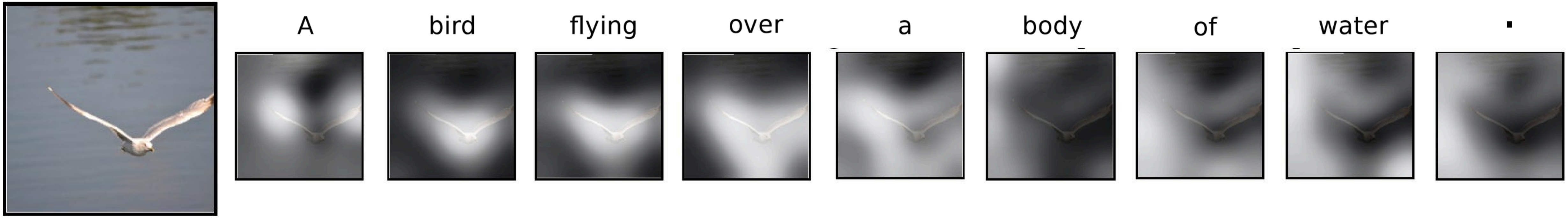


- The decoder doesn't use the fact that the h_i form an ordered sequence – it just treats them as an unordered set

- Can use similar architecture given any set of input hidden vectors $\{h_i\}$!

Image captioning with RNNs and attention

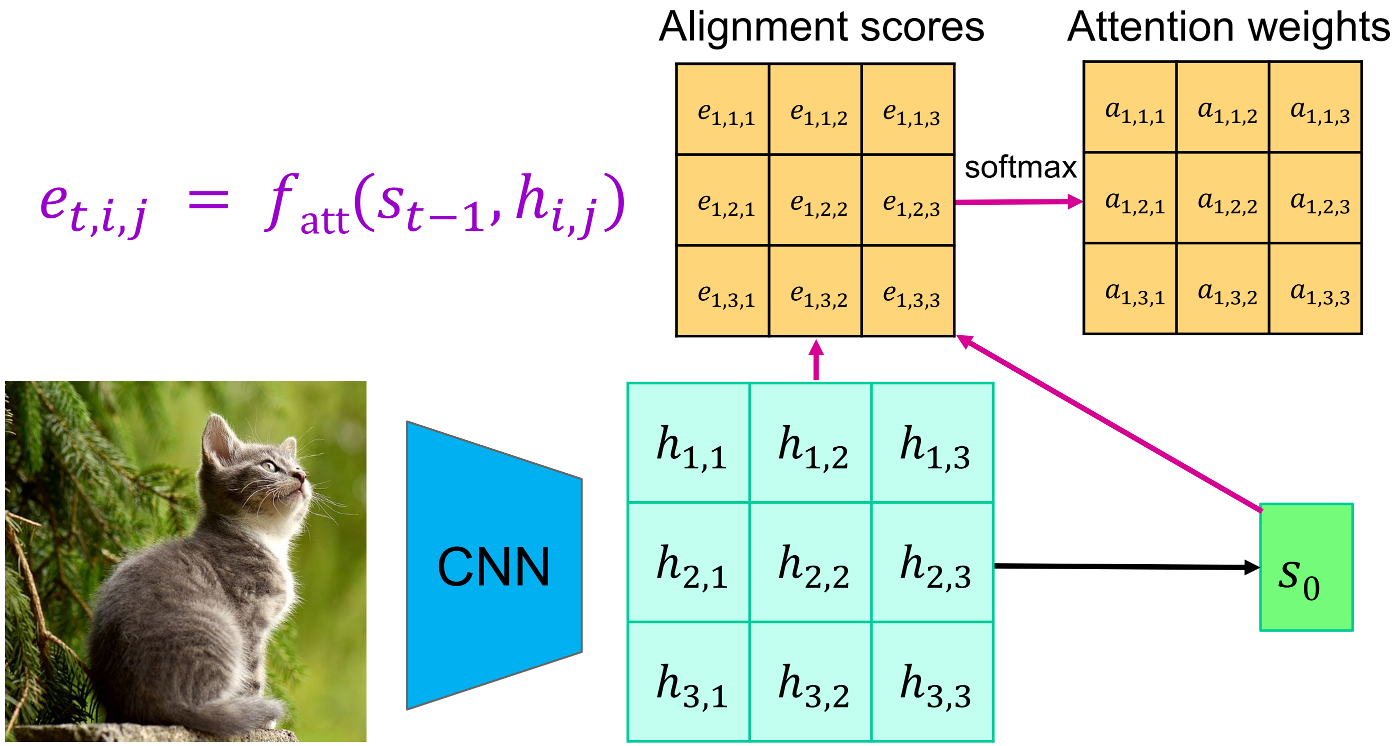
- Idea: pay attention to different parts of the image when generating different words
- Automatically learn this *grounding* of words to image regions without direct supervision



K. Xu et al., Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

Image captioning with RNNs and attention

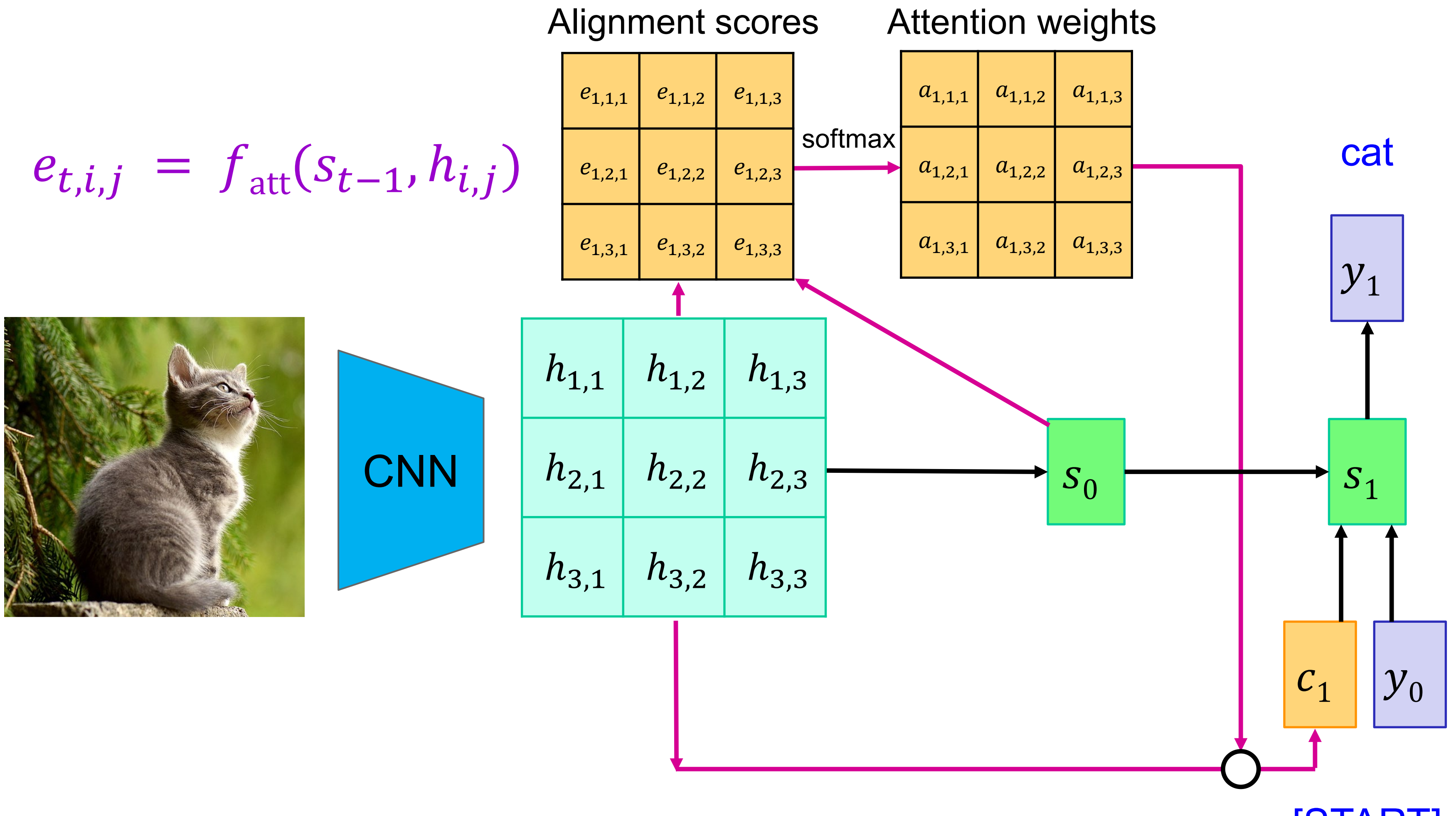
$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$



Use CNN to extract a grid of features

K. Xu et al., Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

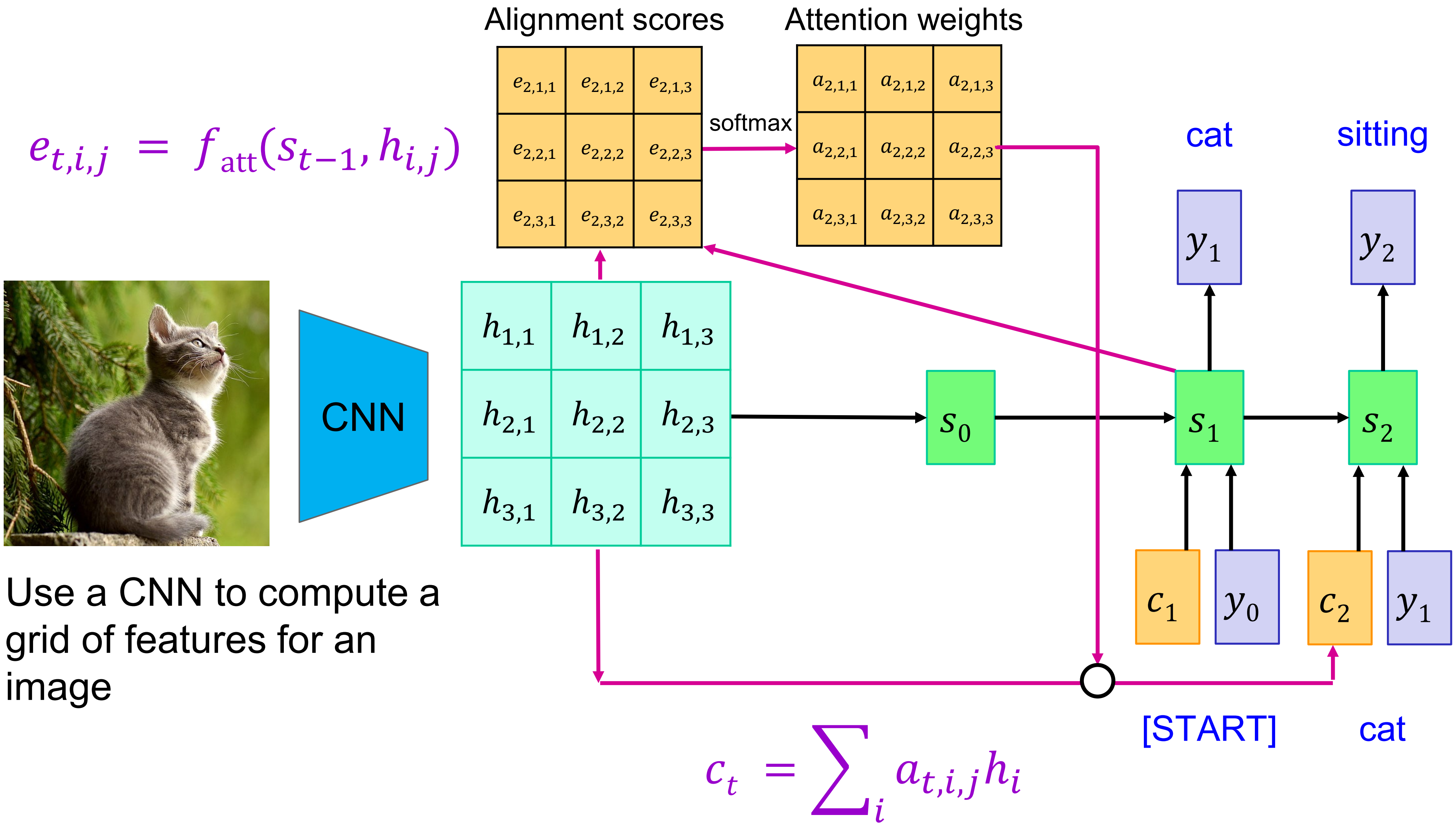
Image captioning with RNNs and attention



$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$

$$c_t = \sum_i a_{t,i,j} h_i$$

Image captioning with RNNs and attention

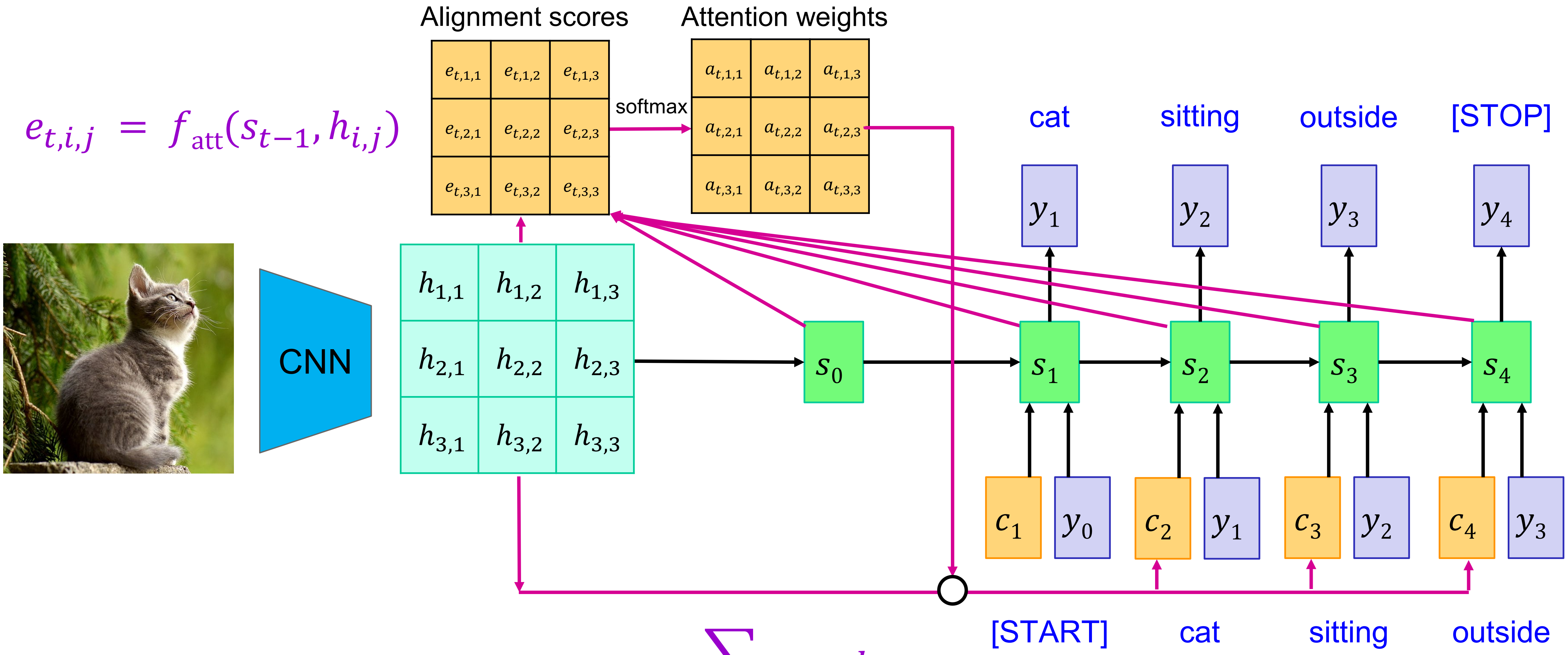


$$e_{t,i,j} = f_{\text{att}}(s_{t-1}, h_{i,j})$$

$$c_t = \sum_i a_{t,i,j} h_i$$

Use a CNN to compute a grid of features for an image

Image captioning with RNNs and attention



$$c_t = \sum_i a_{t,i,j} h_i$$

Each time step of decoder uses a different context vector that looks at different parts of the input image

Example results

- Good captions

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

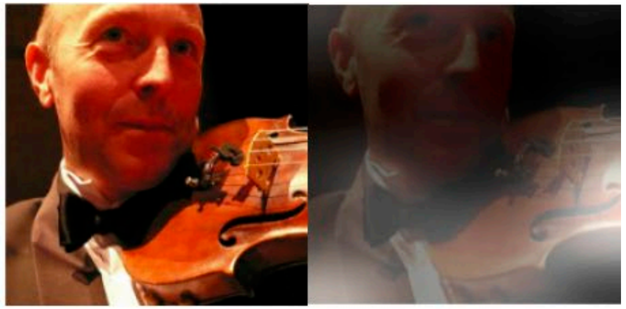
A giraffe standing in a forest with trees in the background.

Example results

- Mistakes

A large white bird standing in a forest.

A woman holding a clock in her hand.



A man wearing a hat and
a hat on a skateboard.

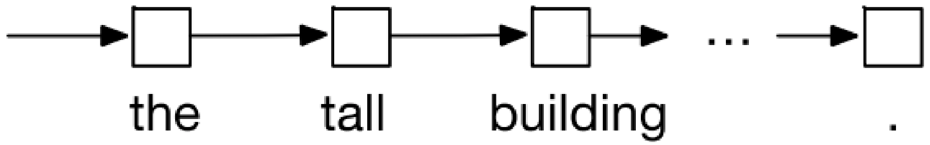
A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Recurrent vs. convolutional sequence models

- **Recurrent models:**
  - Treat input as ordered sequence (inherently sequential processing)
  - Build up context using the hidden vector



- **Convolutional models:**
  - Treat input as a grid indexed by time and feature dimension
  - Build up context using multiple layers of convolutions
  - Processing can be parallel at training time, but convolutions must be *causal*

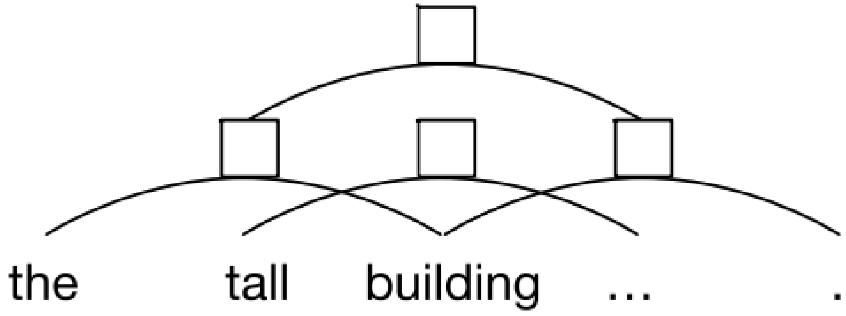


Image source

WaveNet

- Goal: generate raw audio
  - Represented as sequence of 16-bit integer values (can be quantized to 256 discrete levels), 16K samples per second
- Applications: text-to-speech, music generation
  - Also works for speech recognition

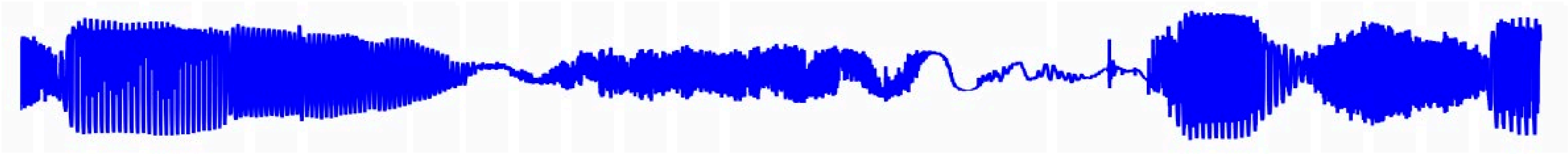


Figure 1: A second of generated speech.

A. van den Oord et al., [WaveNet: A generative model for raw audio](), arXiv 2016

WaveNet

- Training time: compute predictions of all timesteps in parallel (conditioned on ground truth)

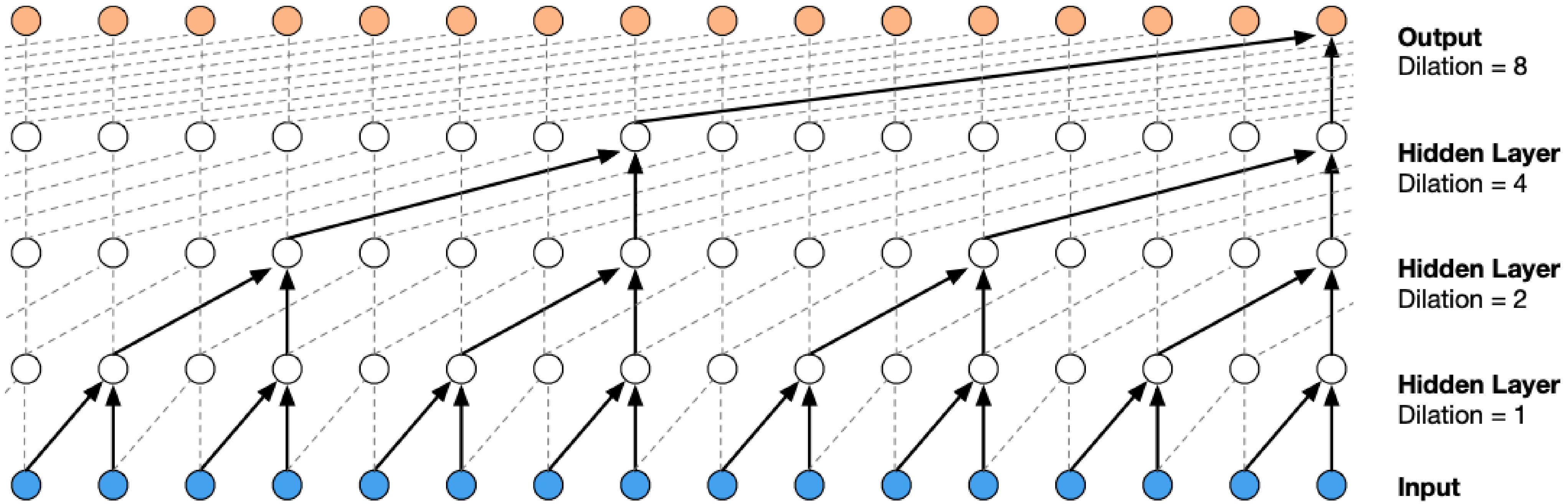


Image source

WaveNet

- Test time: feed each predicted sample back into the model to make prediction at next timestep

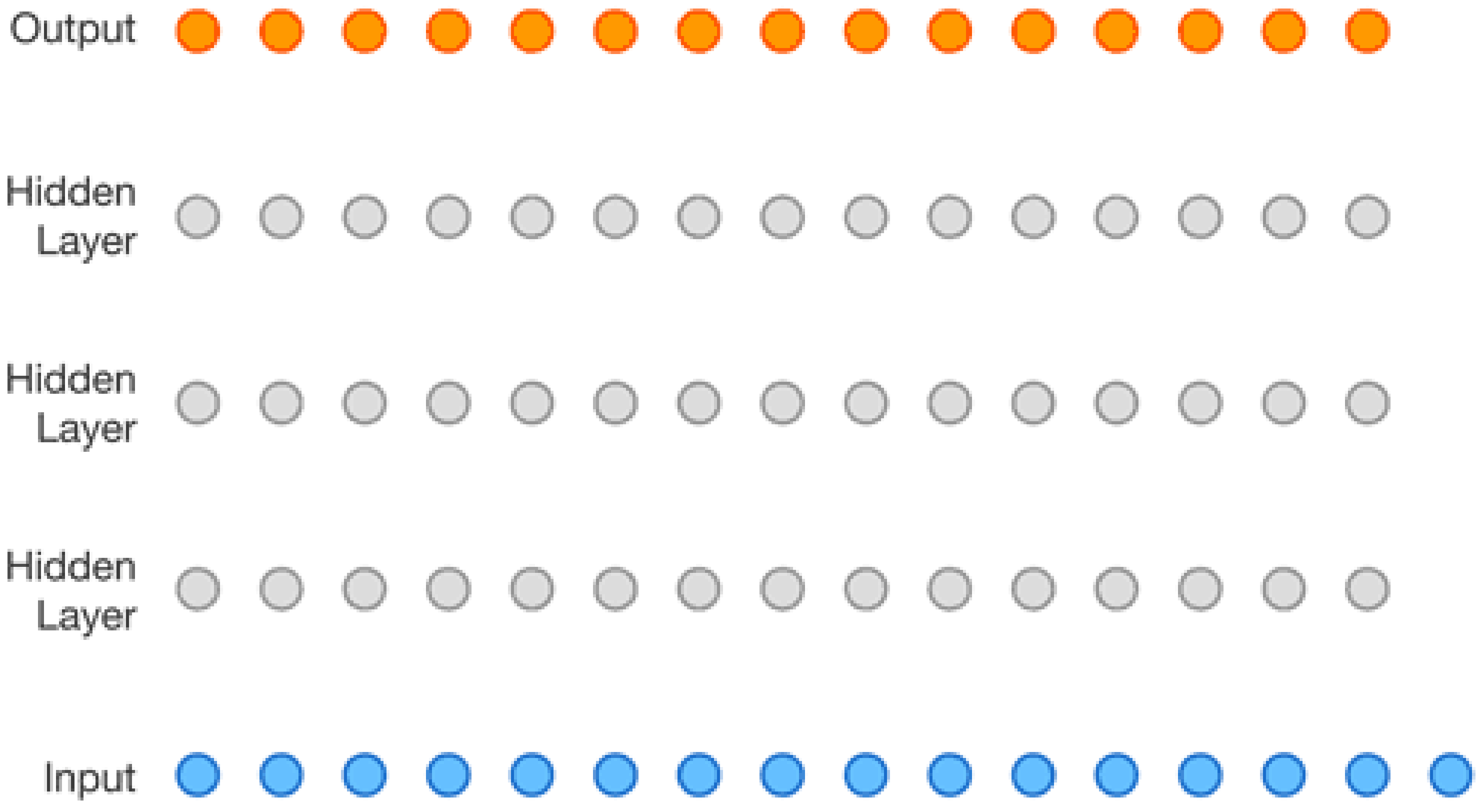


Image source

WaveNet: Results

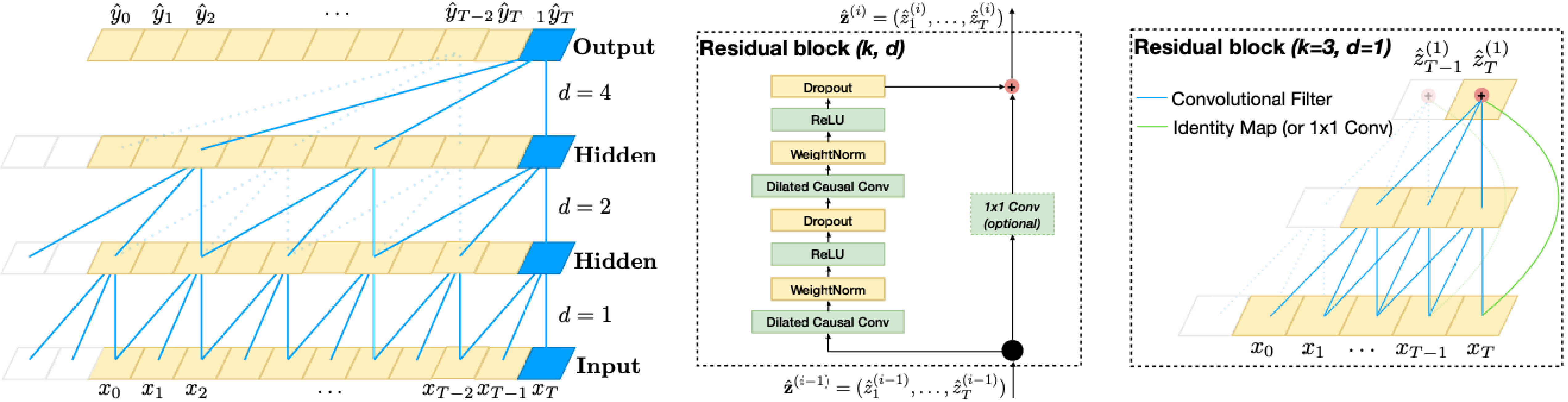
- Text-to-speech with different speaker identities:

- Generated sample of classical piano music:

https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

Temporal convolutional networks (TCNs)

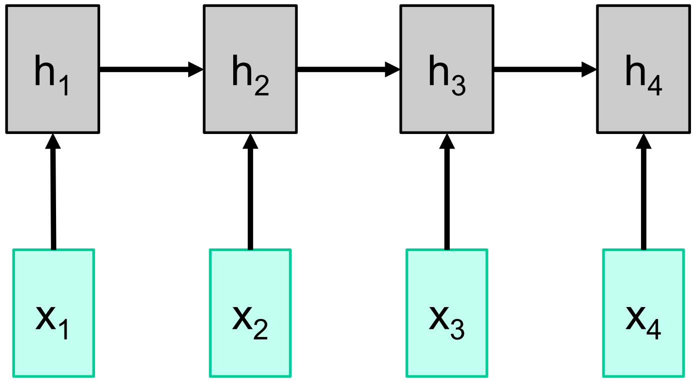
- TCNs can be competitive with RNNs for a variety of sequence modeling tasks



S. Bai, J. Kolter, and V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, arXiv 2018

Different ways of processing sequences

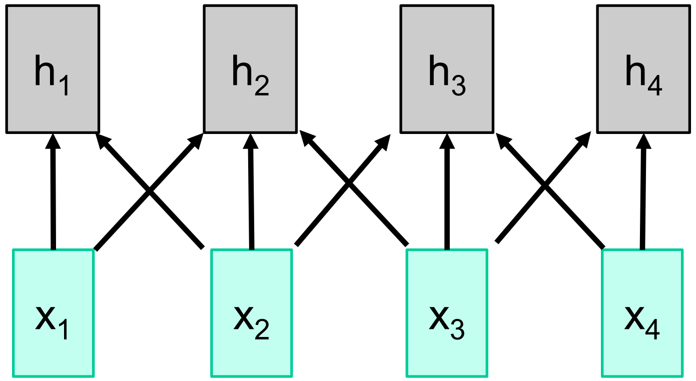
RNN



Works on **ordered sequences**

- Pros: Good for long sequences: After one RNN layer, h_T ”sees” the whole sequence
- Con: Not parallelizable: need to compute hidden states sequentially
- Con: Hidden states have limited expressive capacity

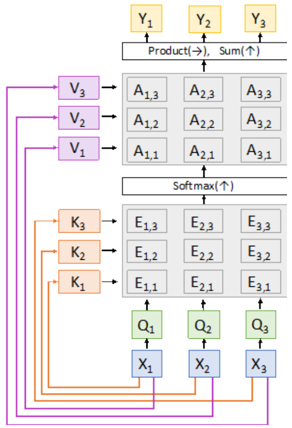
1D convolutional network



Works on **multidimensional grids**

- Pro: Each output can be computed in parallel (at training time)
- Con: Bad at long sequences: Need to stack many conv layers for outputs to “see” the whole sequence

Transformer



- Works on **sets of vectors**
- Pro: Good at long sequences: after one self-attention layer, each output “sees” all inputs!
- Pro: Each output can be computed in parallel (at training time)
- Con: Memory-intensive