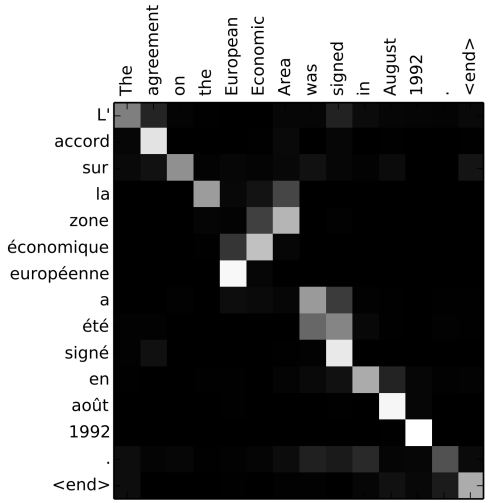
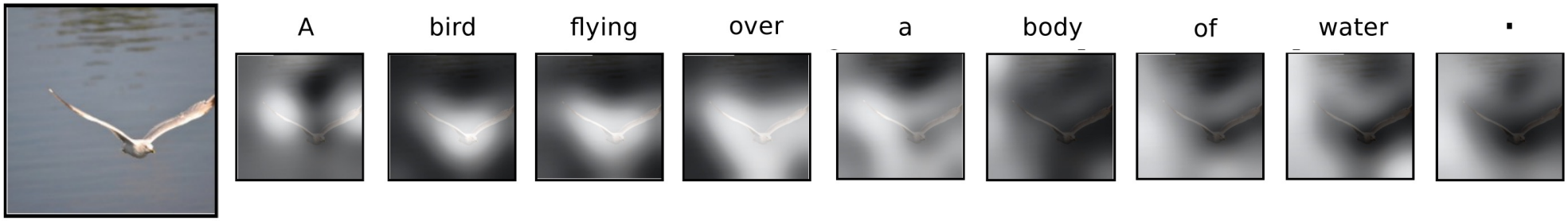
Sequence-to-sequence models with attention





Many slides adapted from J. Johnson

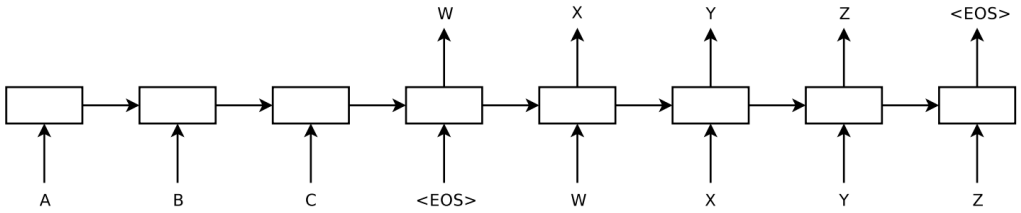
Outline

- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Image captioning with attention
- Transformers

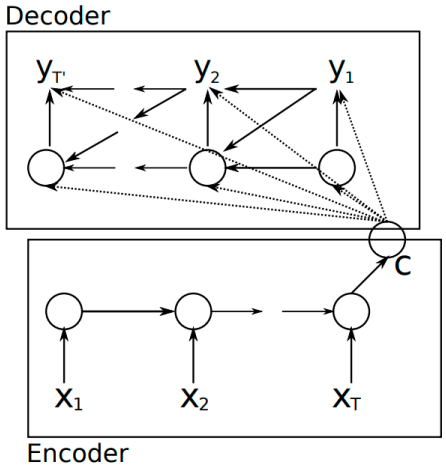
Sequence-to-sequence modeling: Machine translation

"We are eating bread"  "Estamos comiendo pan"

Sequence-to-sequence modeling with RNNs

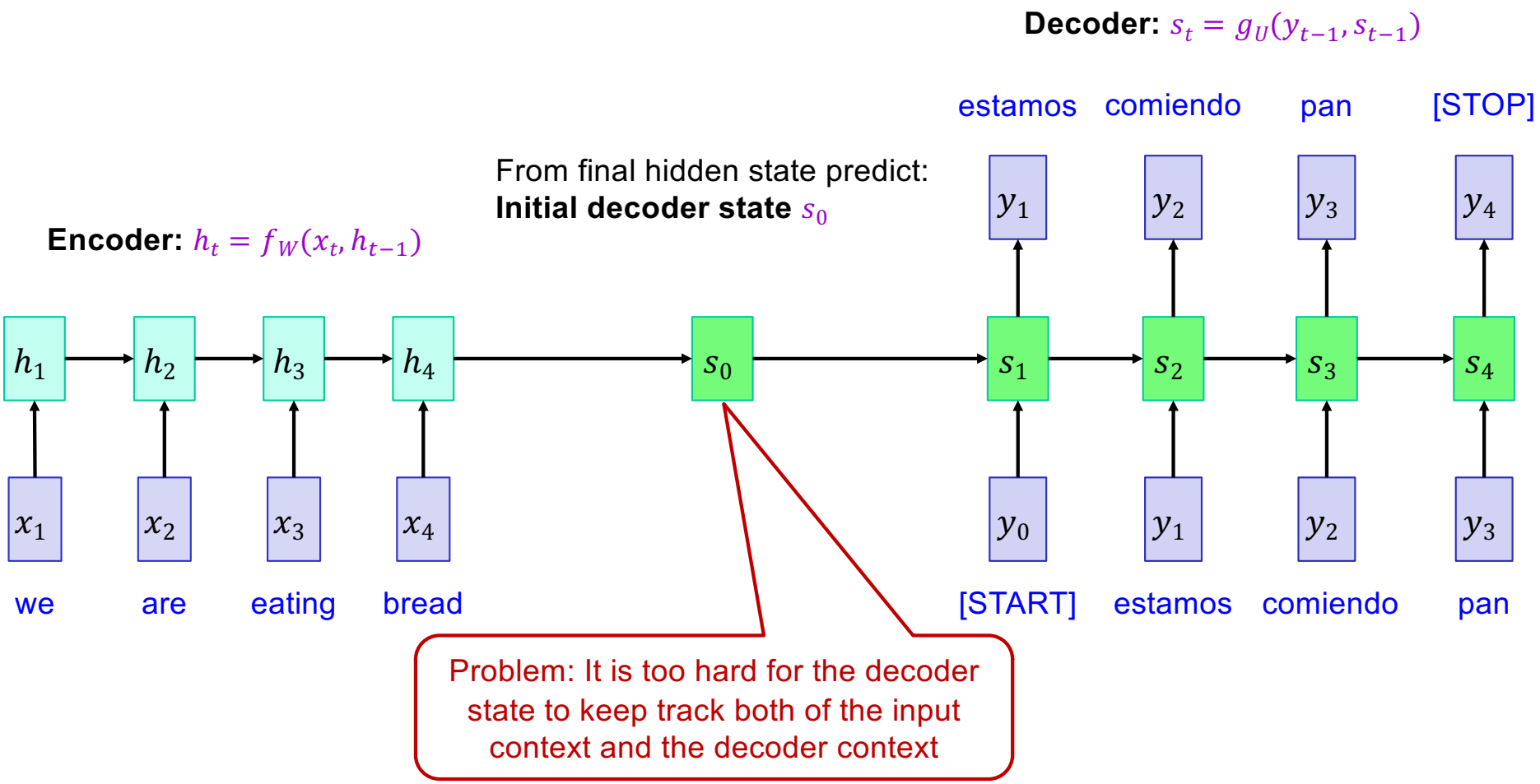


I. Sutskever, O. Vinyals, Q. Le, Sequence to Sequence Learning with Neural Networks, NeurIPS 2014



K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014

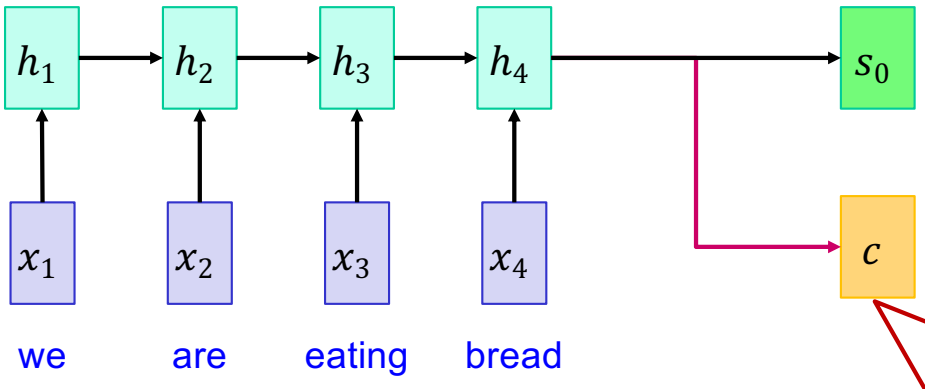
Sequence-to-sequence modeling with RNNs



Sequence-to-sequence modeling with RNNs

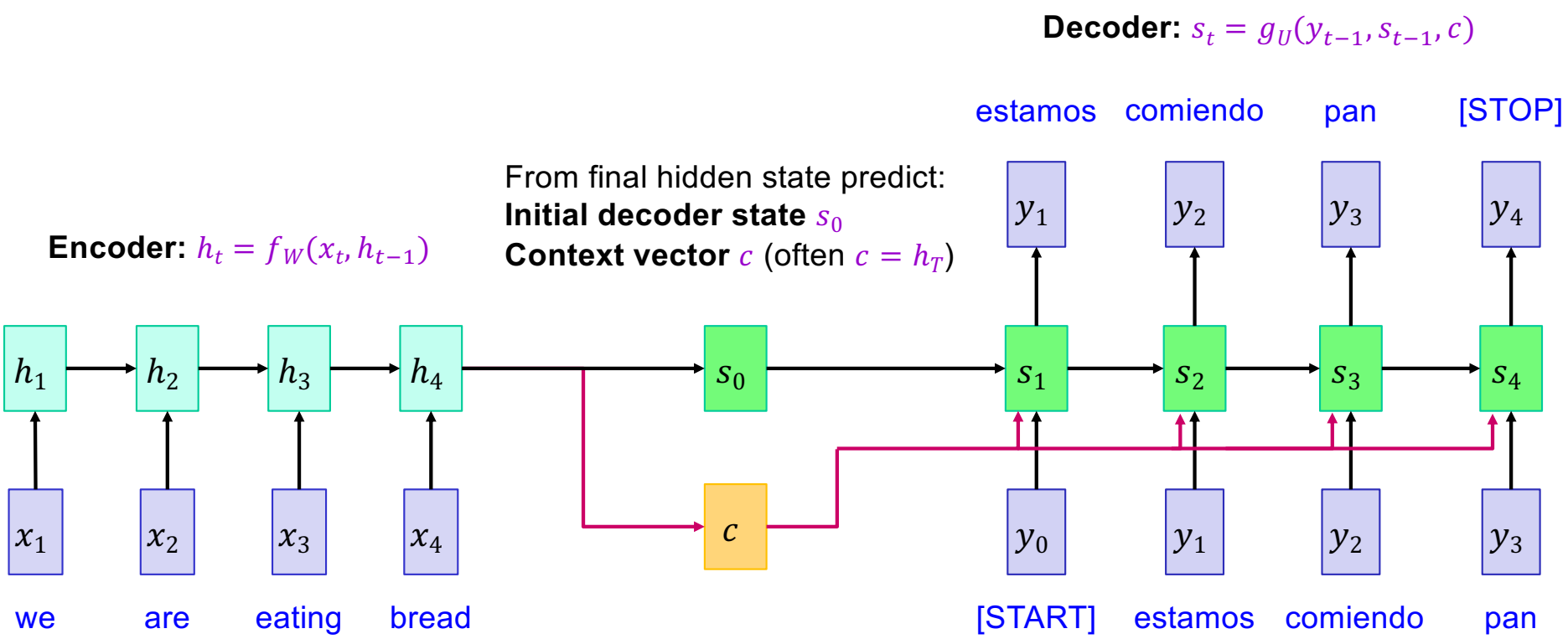
Encoder: $h_t = f_W(x_t, h_{t-1})$

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

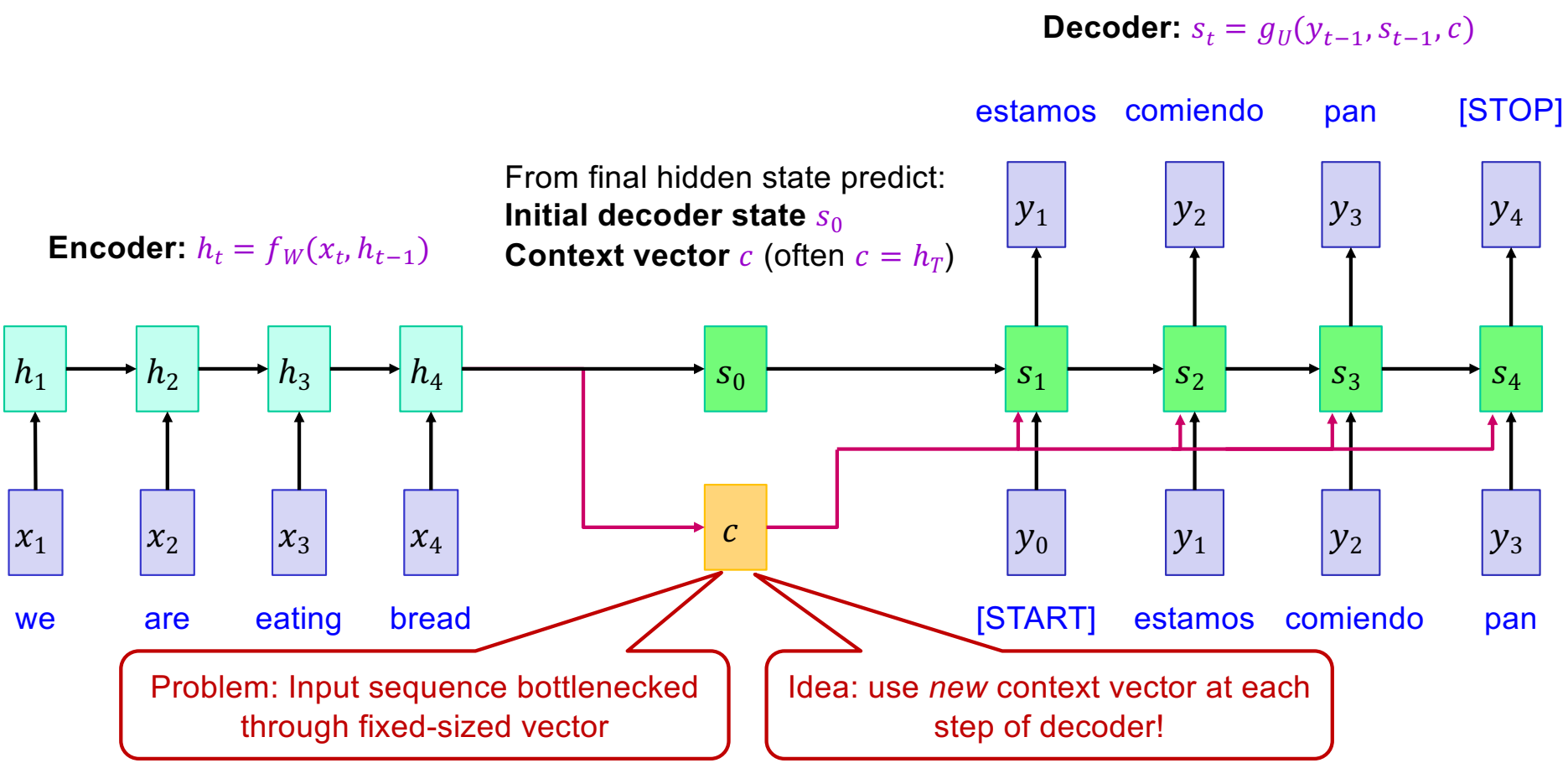


Solution: Introduce a separate context vector to “remember” the input sequence

Sequence-to-sequence modeling with RNNs

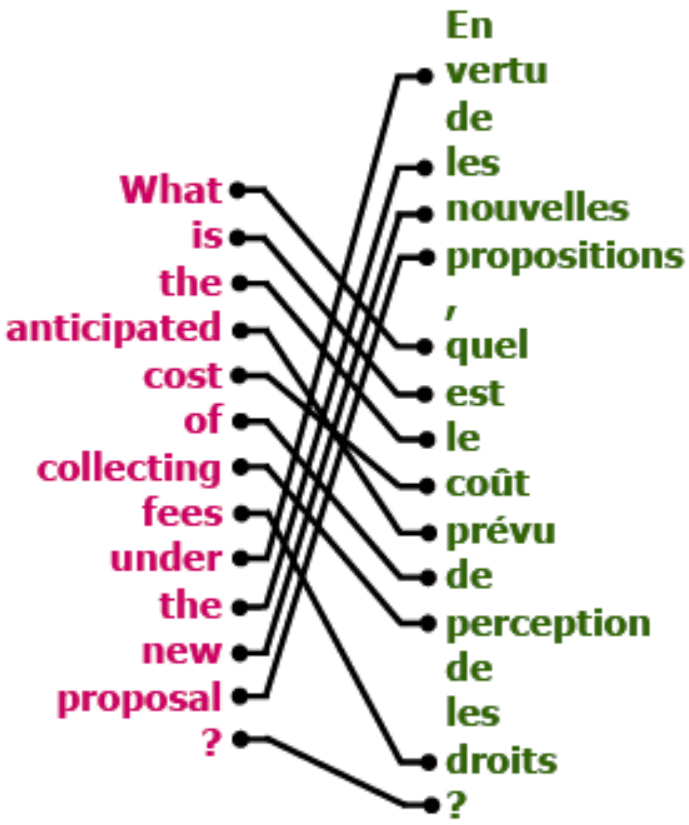


Sequence-to-sequence modeling with RNNs



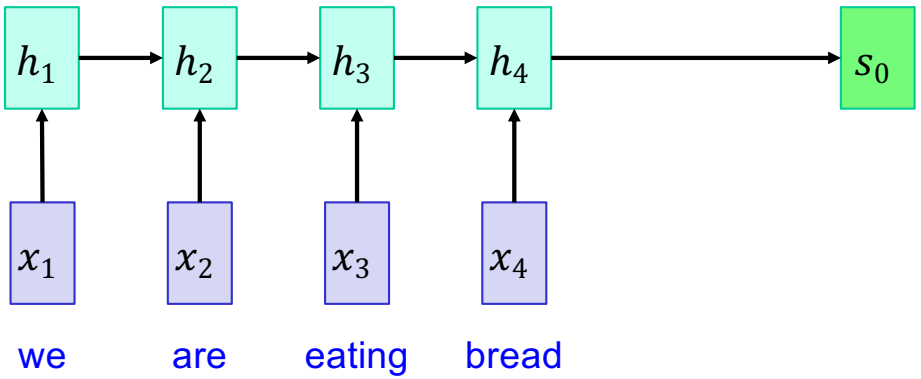
Sequence-to-sequence with RNNs *and attention*

- Intuition: translation requires *alignment*

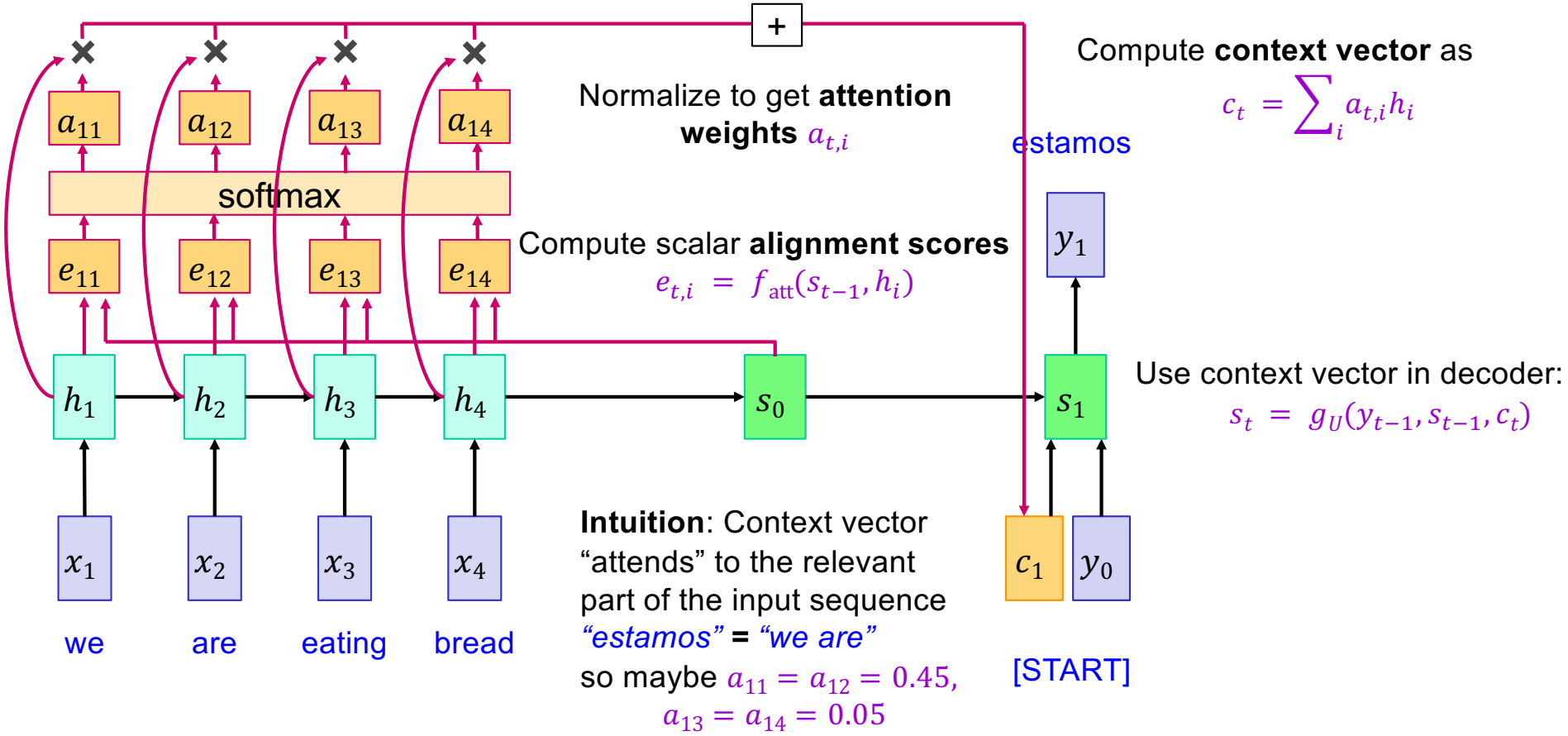


Sequence-to-sequence with RNNs and attention

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

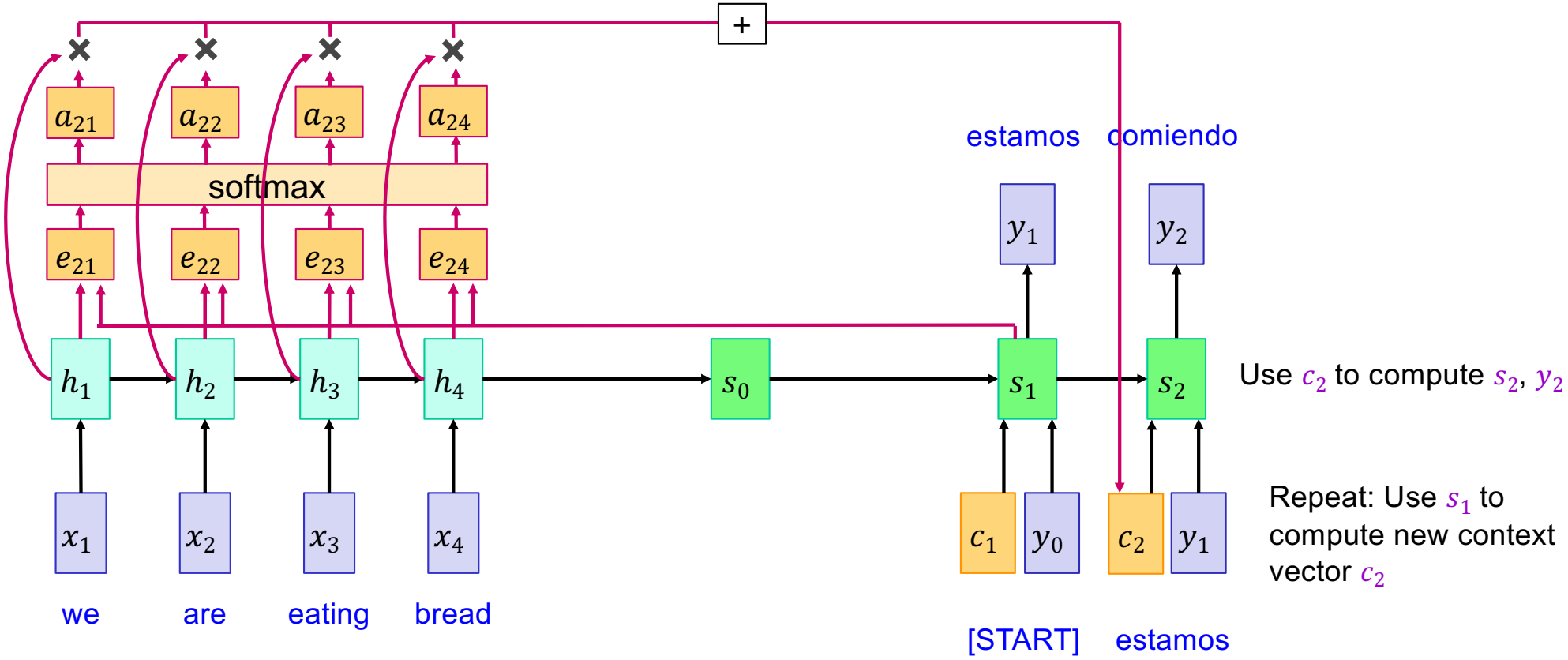


Sequence-to-sequence with RNNs and attention

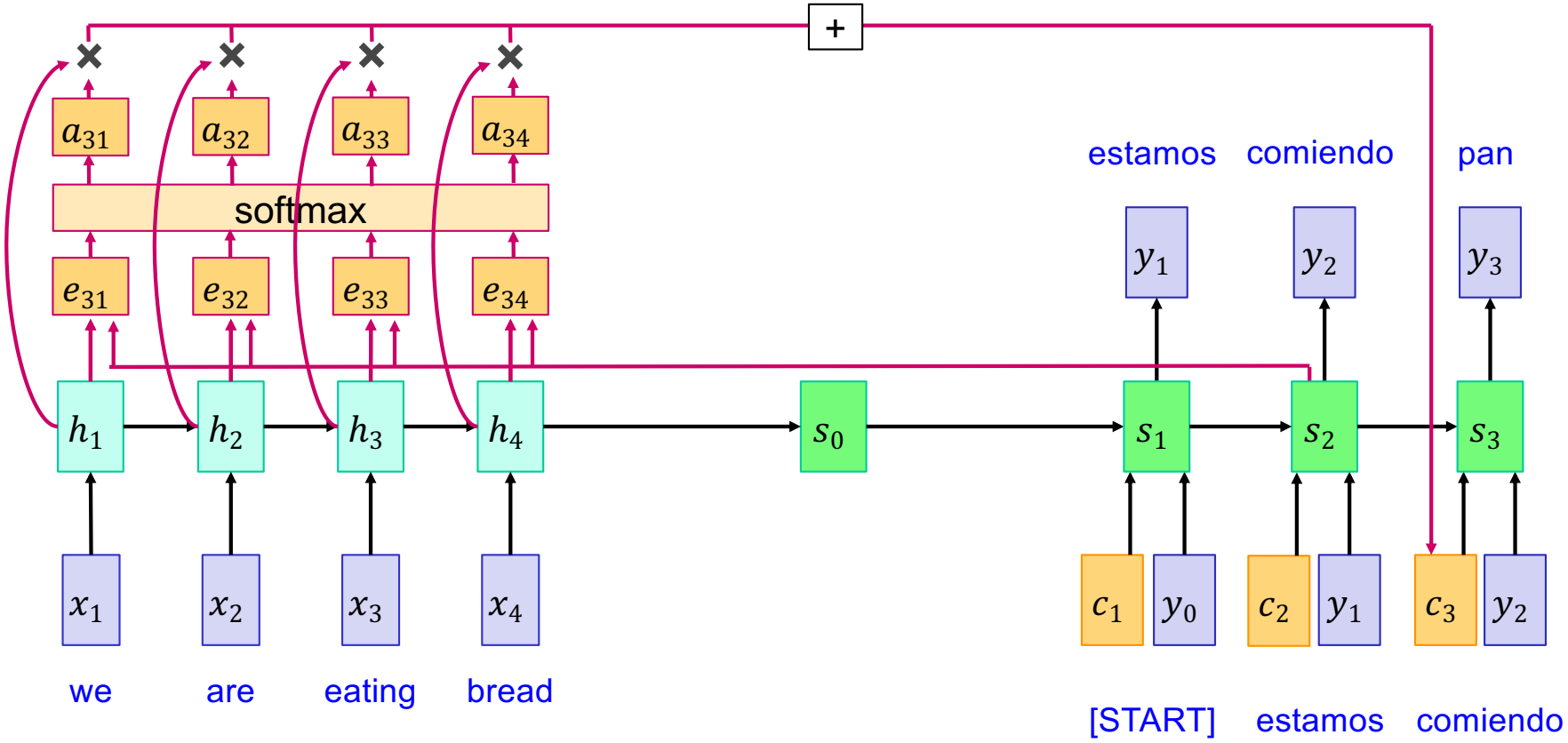


D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015

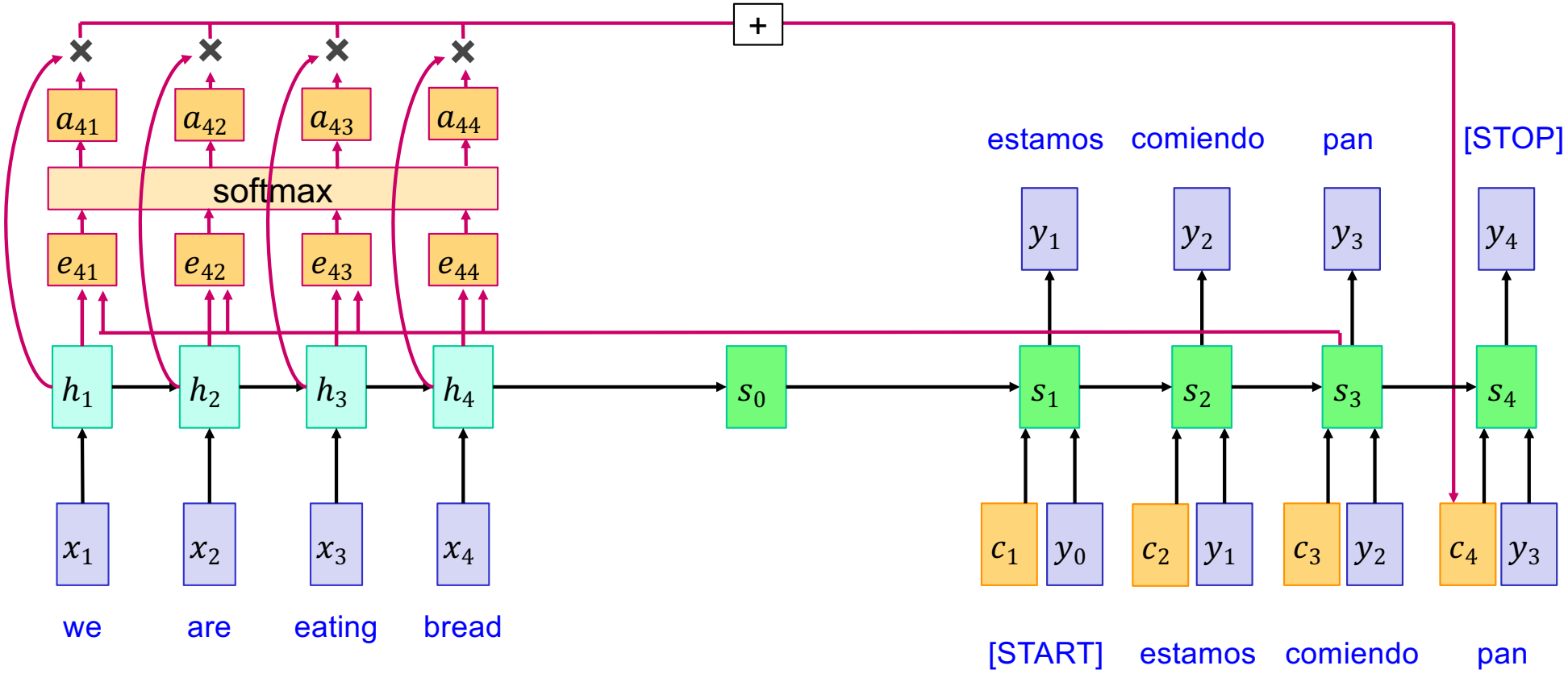
Sequence-to-sequence with RNNs and attention



Sequence-to-sequence with RNNs and attention

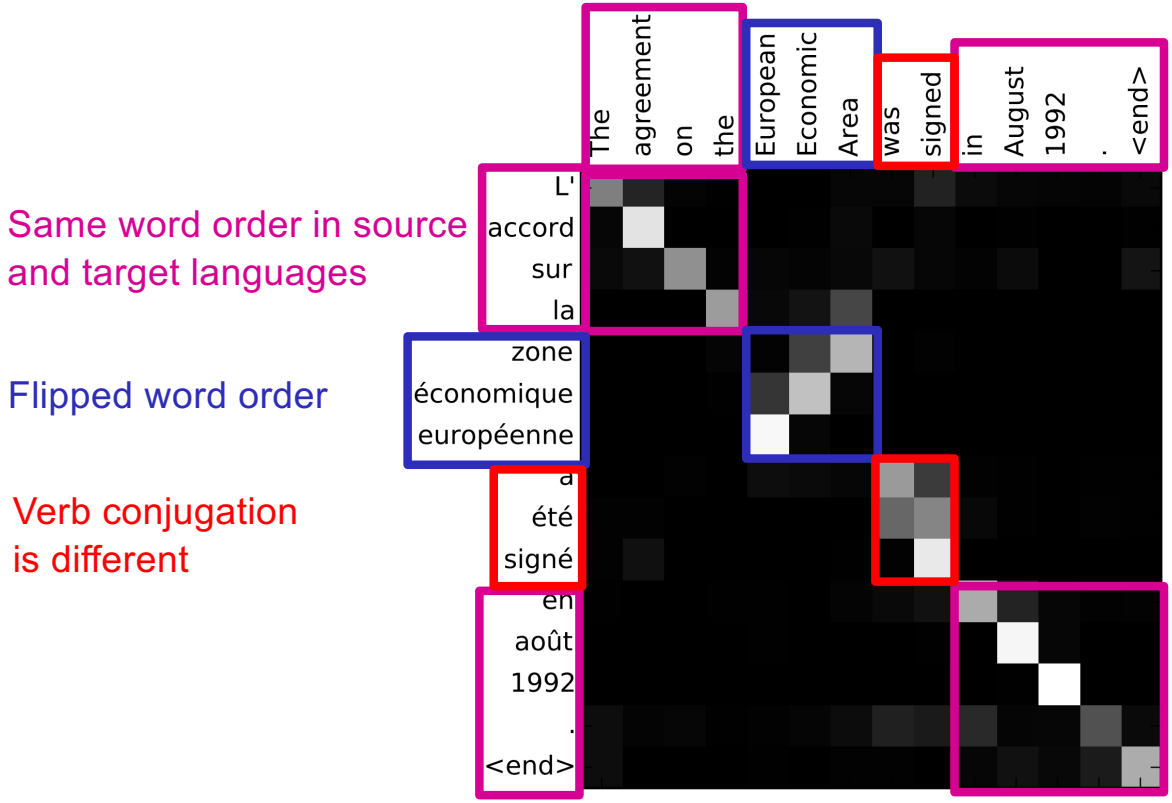


Sequence-to-sequence with RNNs and attention



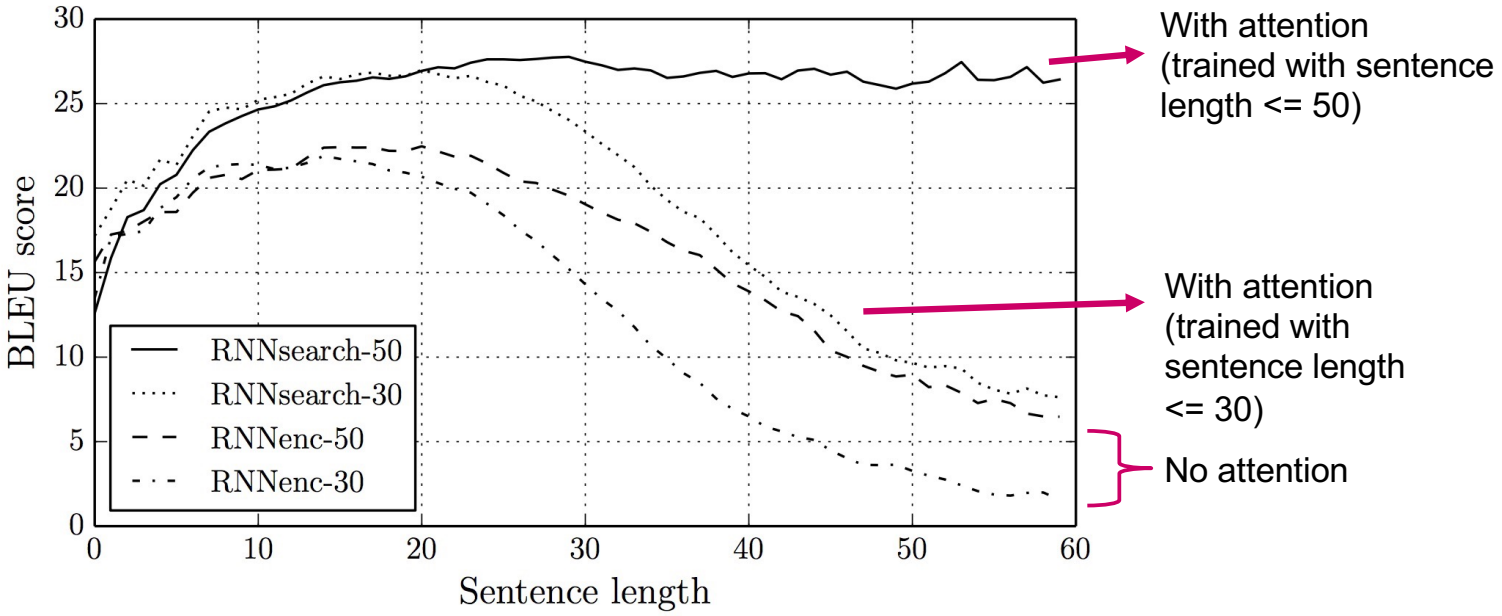
Sequence-to-sequence with RNNs and attention

- 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

Quantitative evaluation



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

Google Neural Machine Translation (GNMT)

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

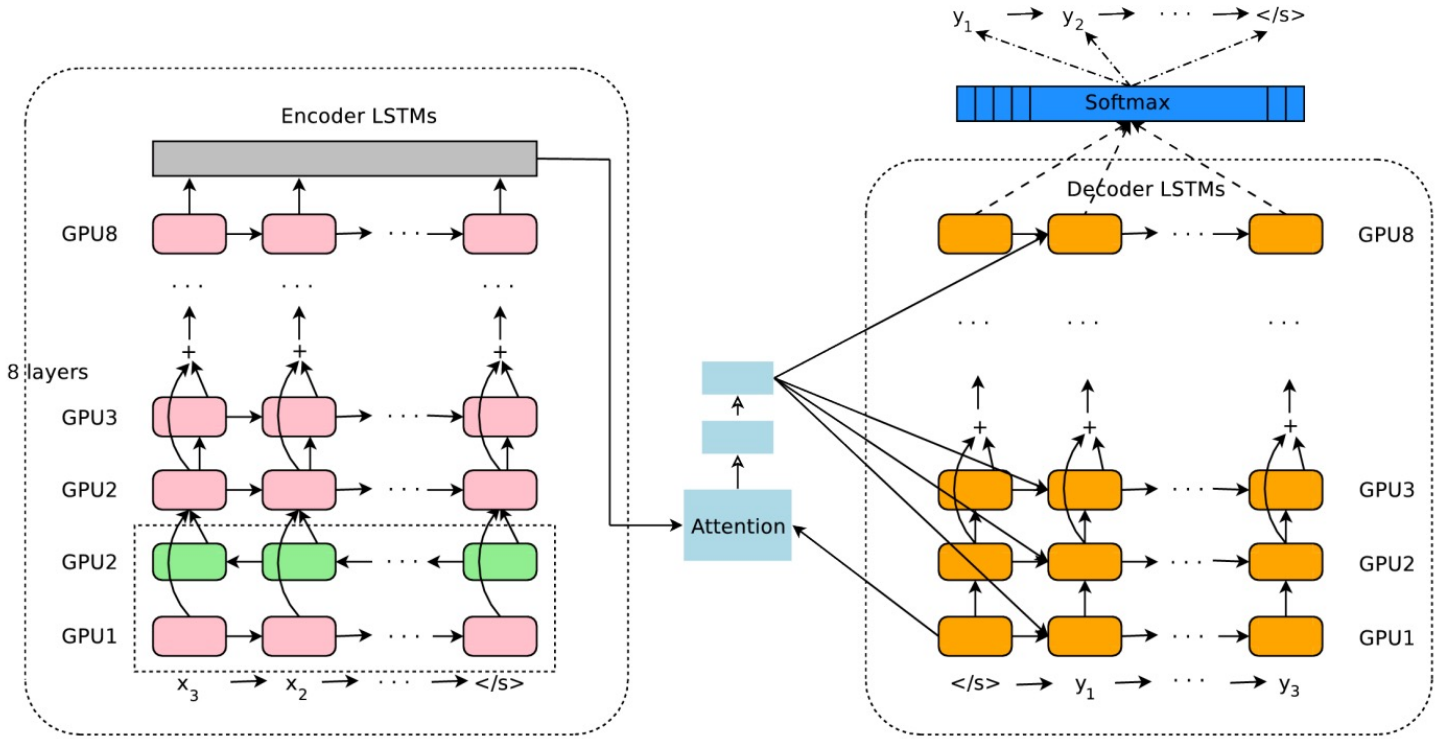
Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey,
Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser,
Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens,
George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa,
Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Y. Wu et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, arXiv 2016

https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html

Google Neural Machine Translation (GNMT)



Y. Wu et al., [Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation](), arXiv 2016

Google Neural Machine Translation (GNMT)

- **Standard training objective:** maximize log-likelihood of ground truth output given input:

$$\sum_i \log P_W(Y_i^* | X_i)$$

  - Only encourages the system to reproduce the reference sentences, does not induce a very good ranking on outputs that don't match reference sentences
  - Not related to task-specific evaluation metric (e.g., BLEU score)

- **Refinement objective:** expectation of rewards over possible predicted sentences Y:

$$\sum_i \sum_Y P_W(Y | X_i)\, R(Y, Y_i^*)$$

  - Use variant of BLEU score to compute reward
  - Reward is not differentiable -- need RL to train (initialize with ML-trained model)

Google Neural Machine Translation (GNMT)

- Human evaluation results on production data (500 randomly sampled sentences from Wikipedia and news websites)

Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.550	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

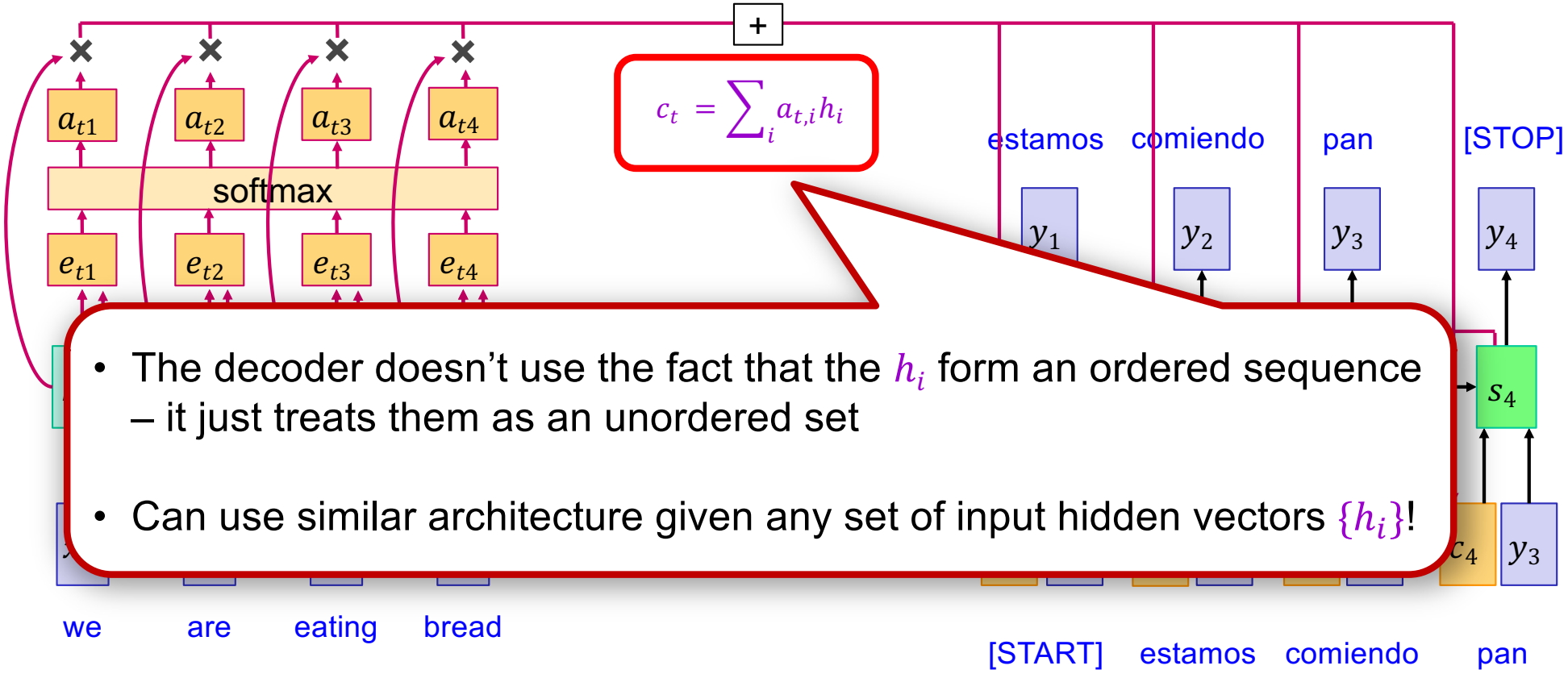
Side-by-side scores: range from 0 ("completely nonsense translation") to 6 ("perfect translation"), produced by human raters fluent in both languages

PBMT: Translation by phrase-based statistical translation system used by Google
GNMT: Translation by GNMT system
Human: Translation by humans fluent in both languages

Outline

- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Image captioning with attention

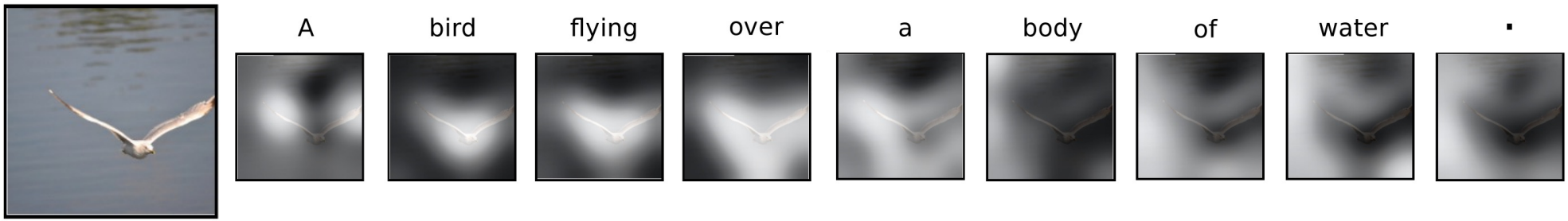
Generalizing attention



- 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 $\{h_i\}$!

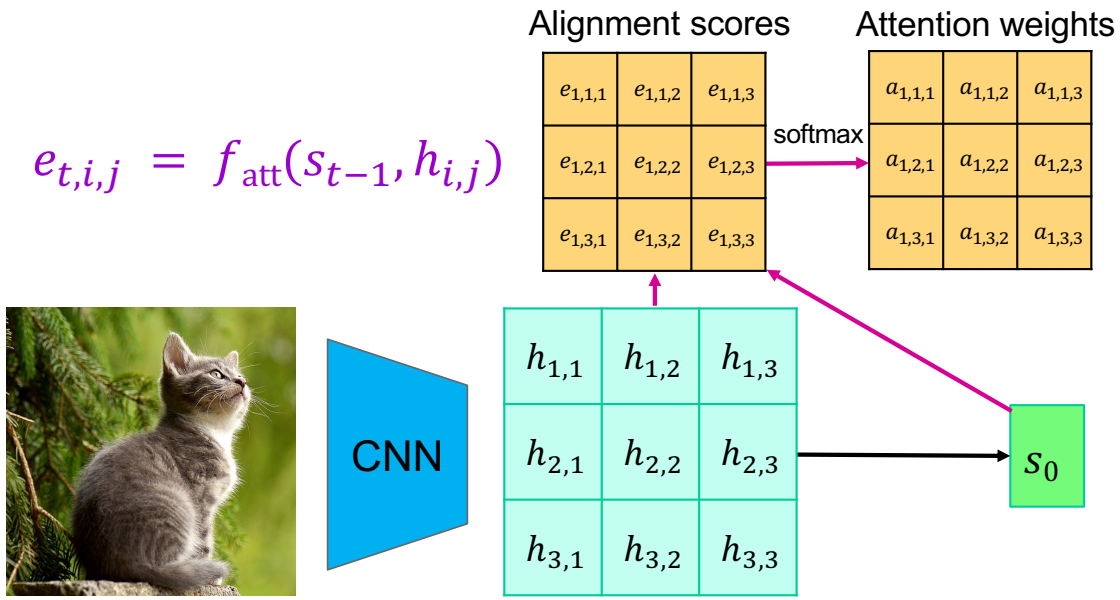
Image captioning with RNNs and attention

- 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



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

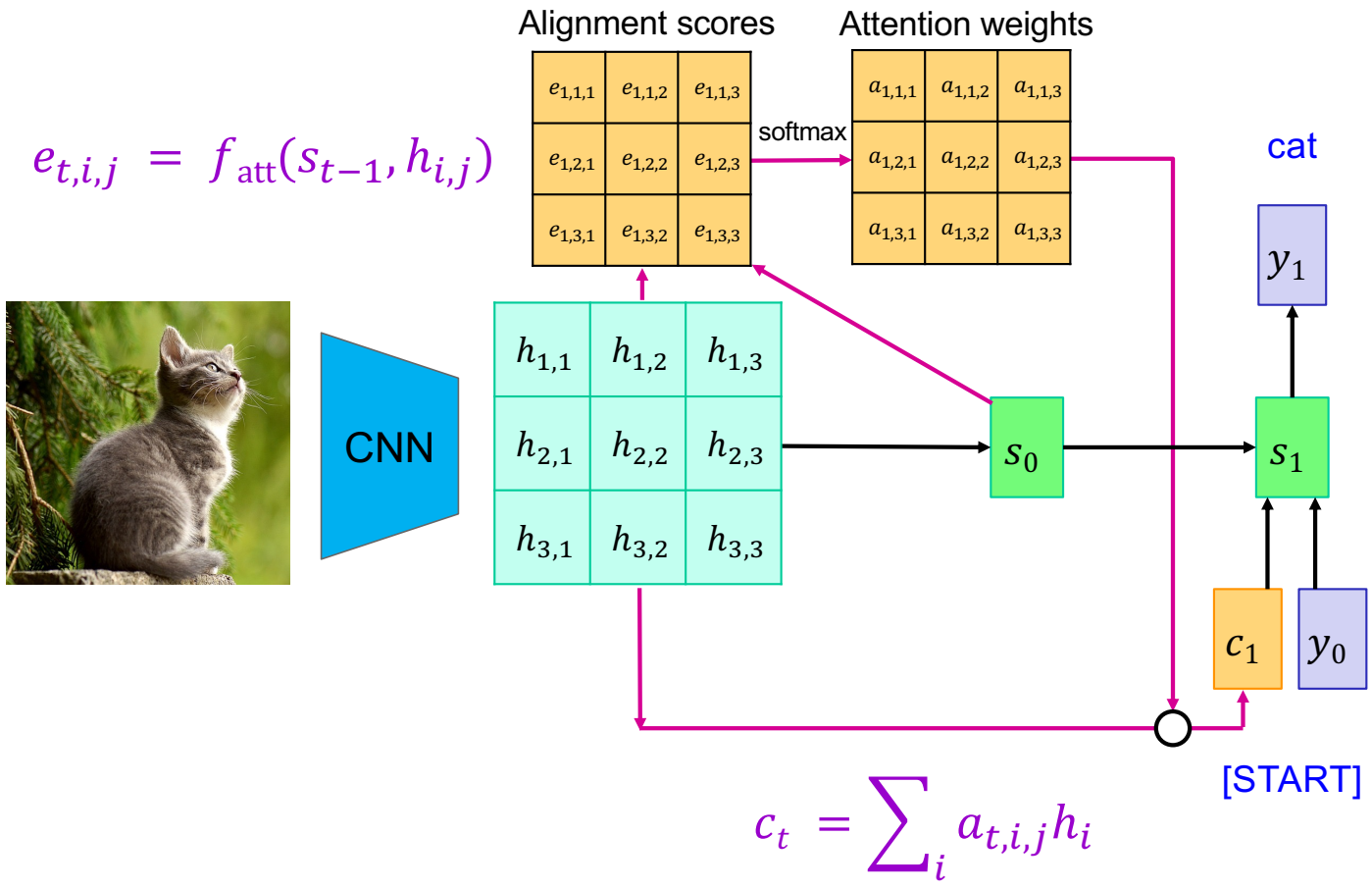


Image captioning with RNNs and attention

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

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

Alignment scores

$e_{2,1,1}$	$e_{2,1,2}$	$e_{2,1,3}$
$e_{2,2,1}$	$e_{2,2,2}$	$e_{2,2,3}$
$e_{2,3,1}$	$e_{2,3,2}$	$e_{2,3,3}$

softmax

Attention weights

$a_{2,1,1}$	$a_{2,1,2}$	$a_{2,1,3}$
$a_{2,2,1}$	$a_{2,2,2}$	$a_{2,2,3}$
$a_{2,3,1}$	$a_{2,3,2}$	$a_{2,3,3}$

$h_{1,1}$	$h_{1,2}$	$h_{1,3}$
$h_{2,1}$	$h_{2,2}$	$h_{2,3}$
$h_{3,1}$	$h_{3,2}$	$h_{3,3}$

s_0 s_1 s_2

y_1 y_2

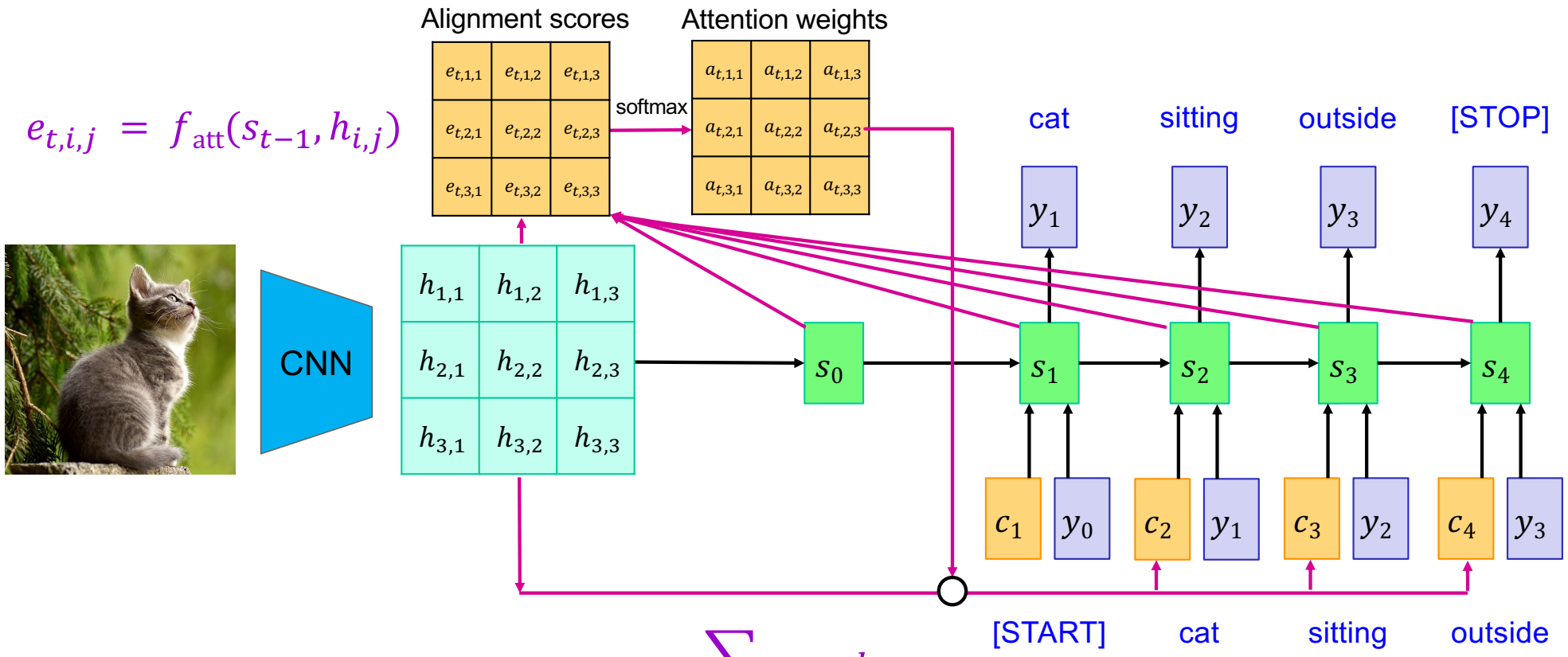
cat sitting

c_1 y_0 c_2 y_1

[START] cat

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

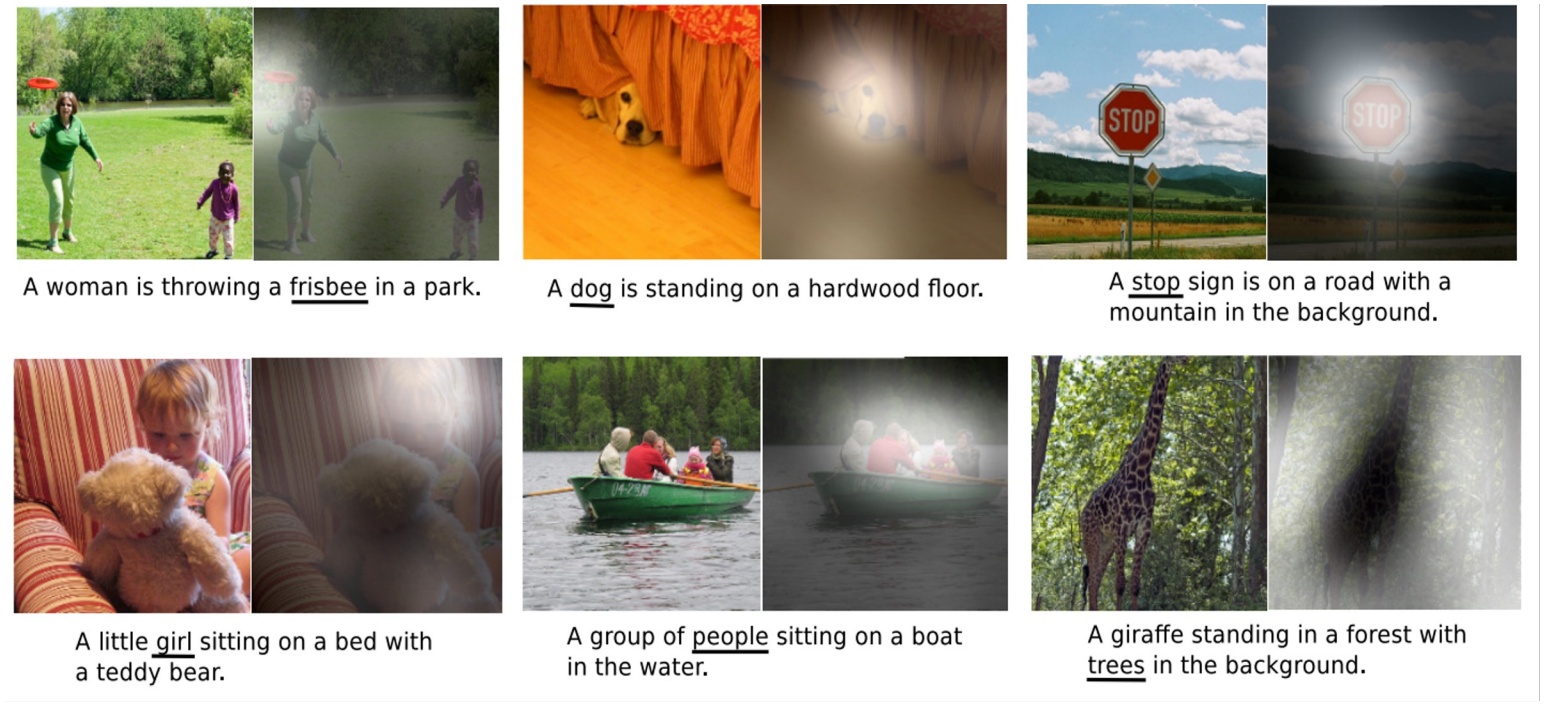
Image captioning with RNNs and attention



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

Example results

- Good captions

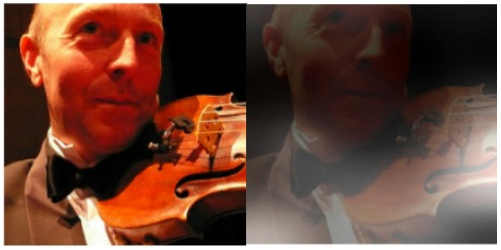


Example results

- 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.

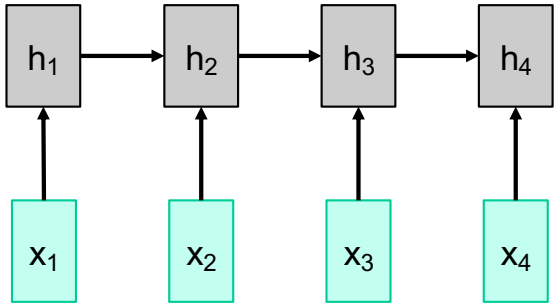
Quantitative results

Dataset	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC	63	41	27	-	-
	Soft-Attention	**67**	44.8	29.9	19.5	18.93
	Hard-Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC	66.3	42.3	27.7	18.3	-
	Soft-Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard-Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
COCO	Google NIC	66.6	46.1	32.9	24.6	-
	Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

Source

Sequence modeling beyond RNNs

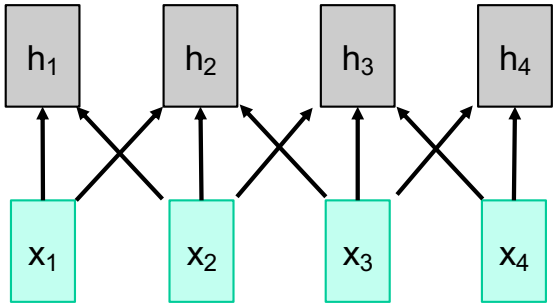
RNNs



Works on **ordered sequences**

- Pros: Not limited by fixed context size (in principle): After one RNN layer, h_T "sees" the whole sequence
- Con: Hidden states have limited expressive capacity
- Con: Not parallelizable: need to compute hidden states sequentially

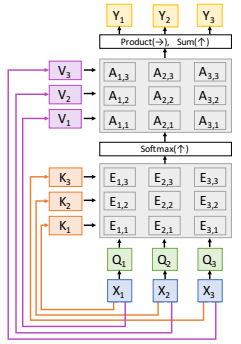
1D convolutional networks



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

Transformers



- Works on **sets of vectors**

Outline

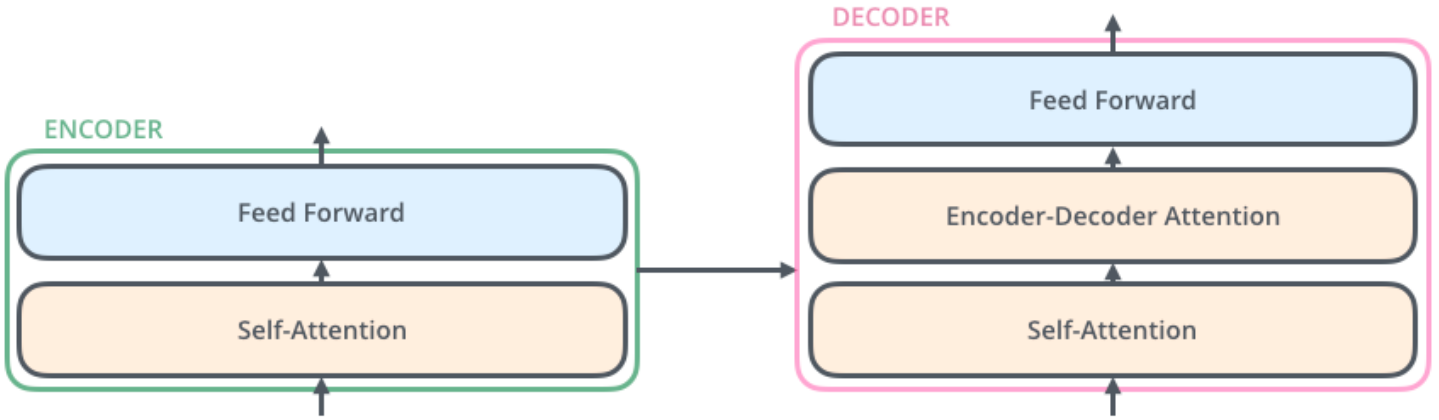
- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Image captioning with attention
- Transformers

Basic transformer model

- Sequence-to-sequence architecture using *only point-wise processing and attention* – no recurrent units or convolutions

Encoder: receives entire input sequence and outputs encoded sequence of the same length

Decoder: predicts next token conditioned on encoder output and previously predicted tokens



A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, NeurIPS 2017

Image source

Key-Value-Query attention model

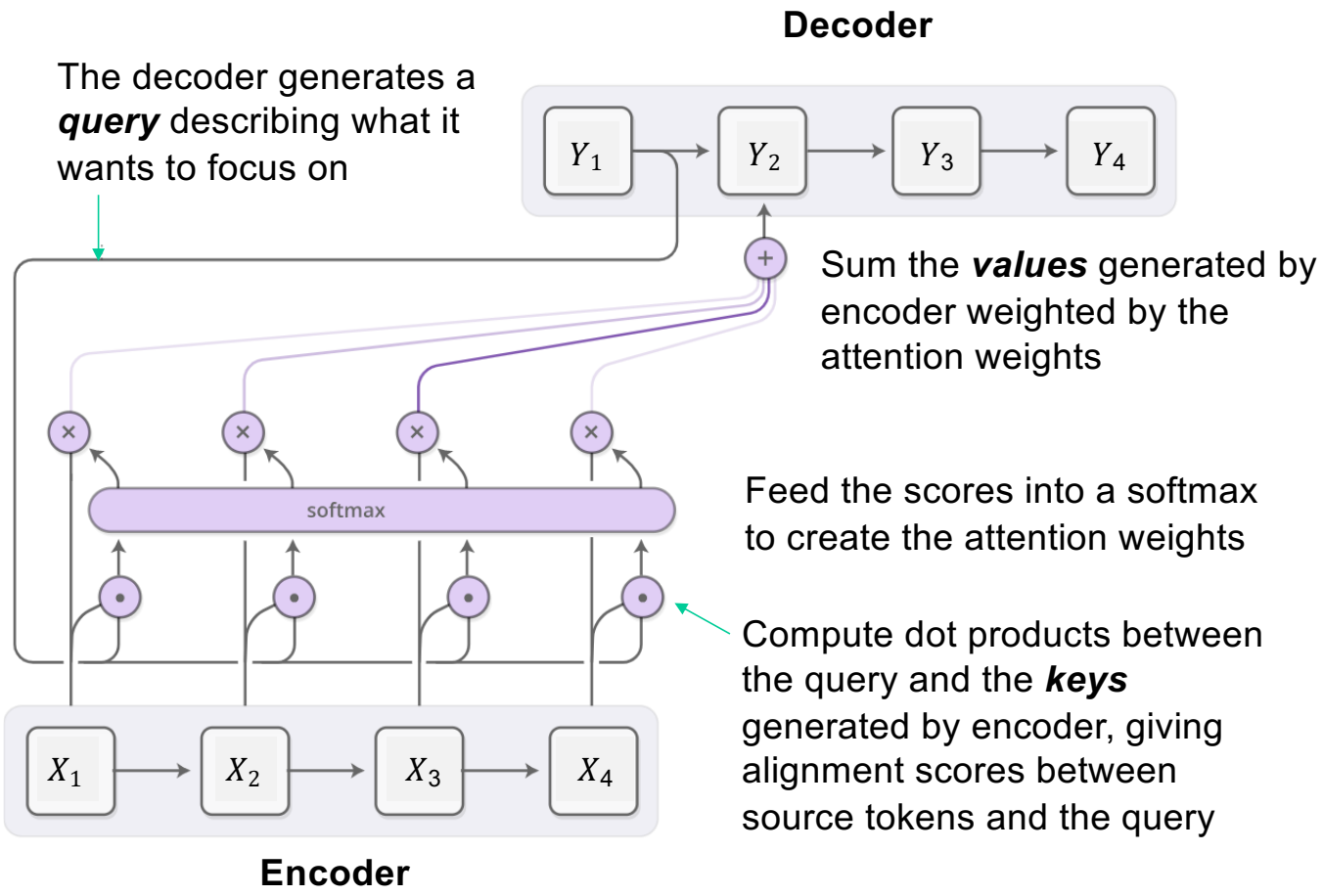


Image source

Key-Value-Query attention model

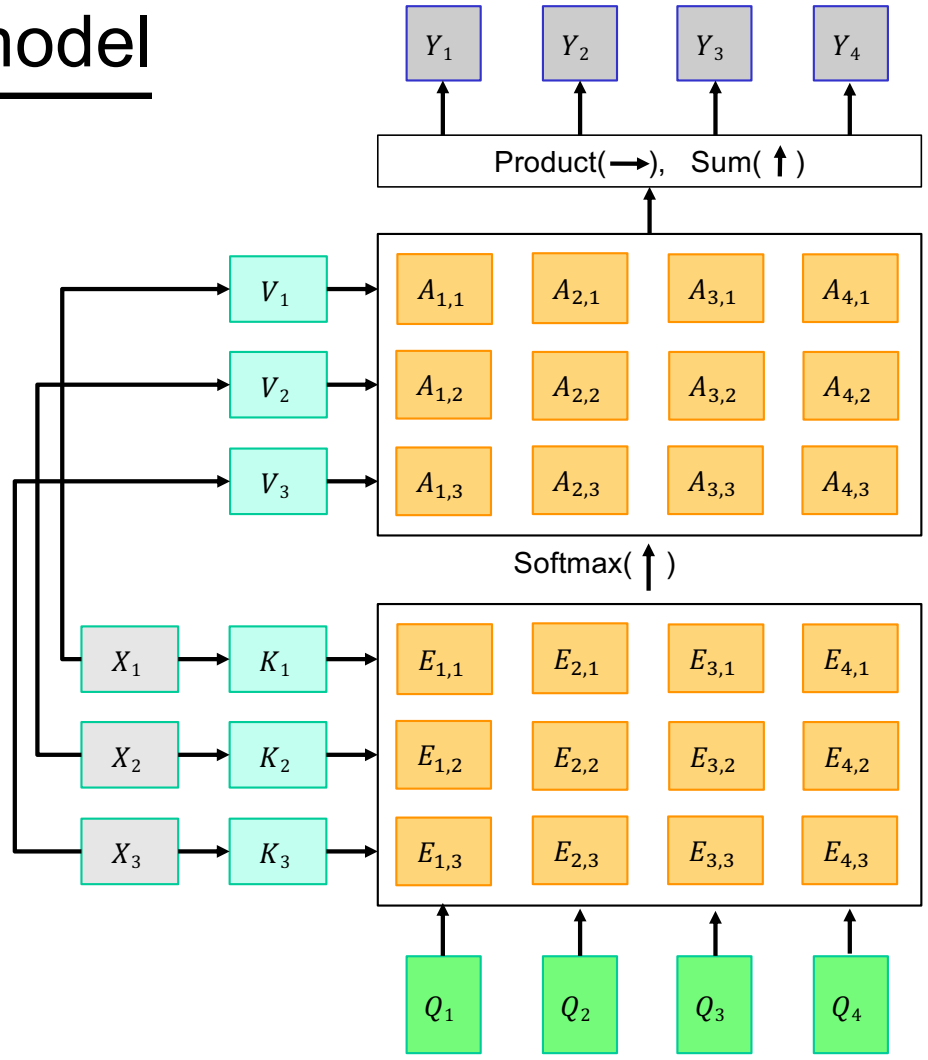
- Key vectors: $K = XW_K$
- Value Vectors: $V = XW_V$
- Query vectors
- Similarities: *scaled dot-product attention*

$$E_{i,j} = \frac{(Q_i \cdot K_j)}{\sqrt{D}} \text{ or } E = QK^T/\sqrt{D}$$

(D is the dimensionality of the keys)

- Attn. weights: $A = \text{softmax}(E, \text{dim} = 1)$
- Output vectors:

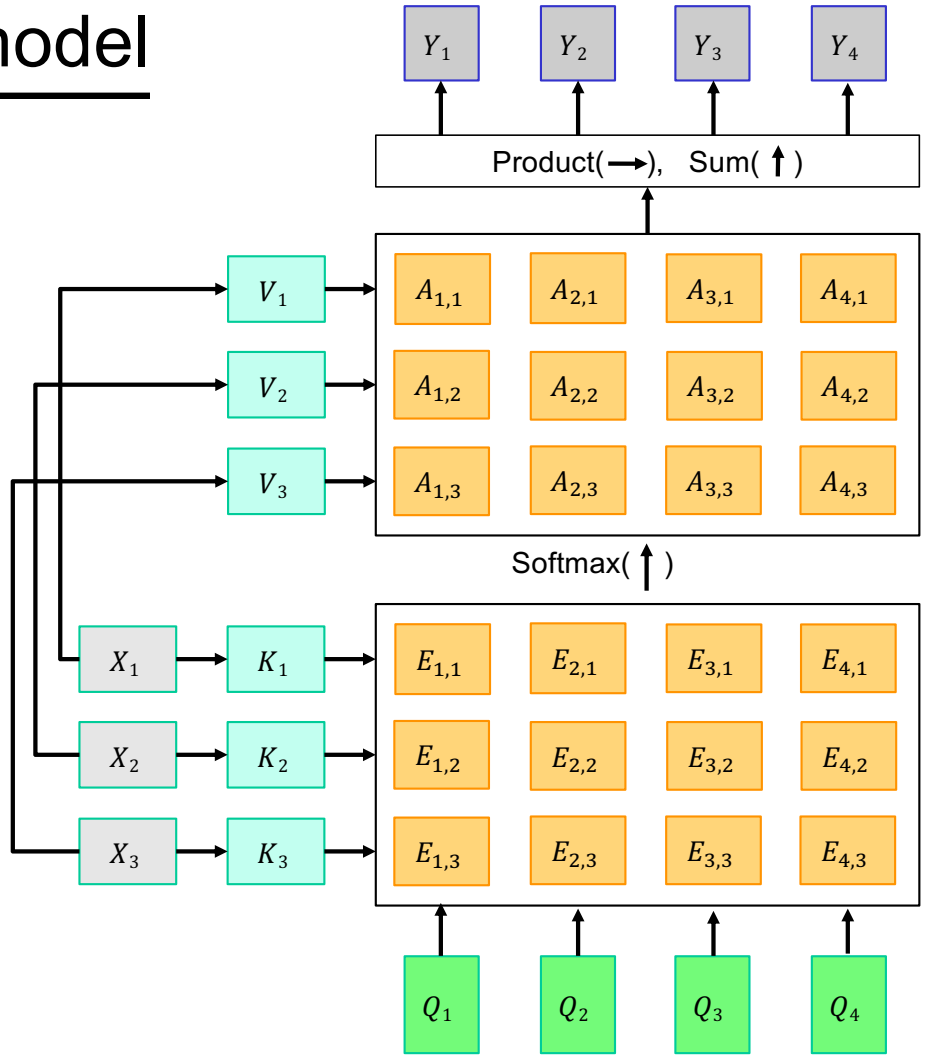
$$Y_i = \sum_j A_{i,j} V_j \text{ or } Y = AV$$



Adapted from J. Johnson

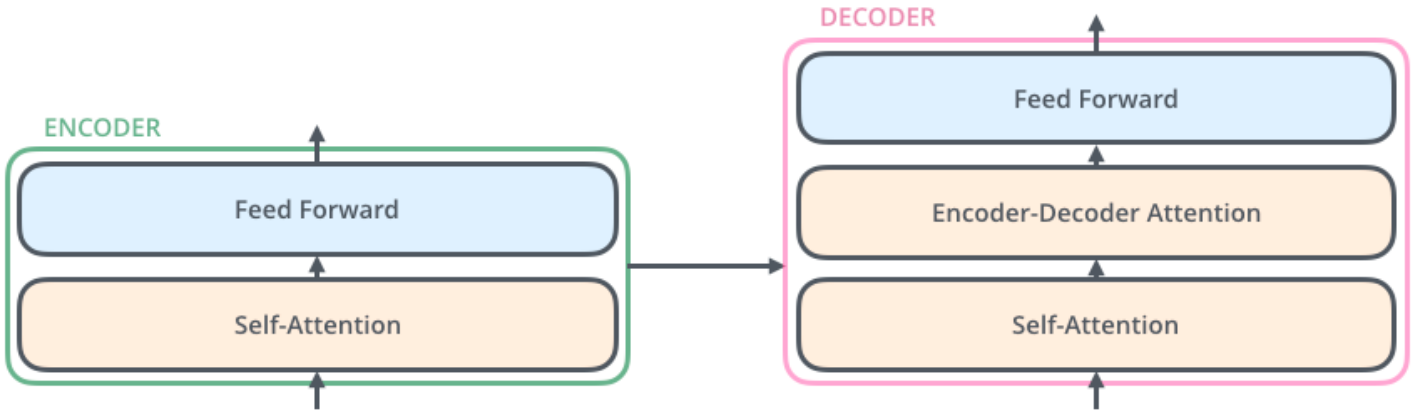
Key-Value-Query attention model

- How does permuting the order of the *queries* change the output?
- How does changing the order of the *keys/values* change the output?



Adapted from J. Johnson

Attention mechanisms



- **Encoder self-attention:** queries, keys, and values come from previous layer of encoder
- **Decoder self-attention:** values corresponding to future decoder outputs are masked out
- **Encoder-decoder attention:** queries come from previous decoder layer, keys and values come from output of encoder

Self-attention

- Used to capture context *within the sequence*

As we are encoding “it”, we should focus on “the animal”

As we are encoding “it”, we should focus on “the street”

Image source

Self-attention layer

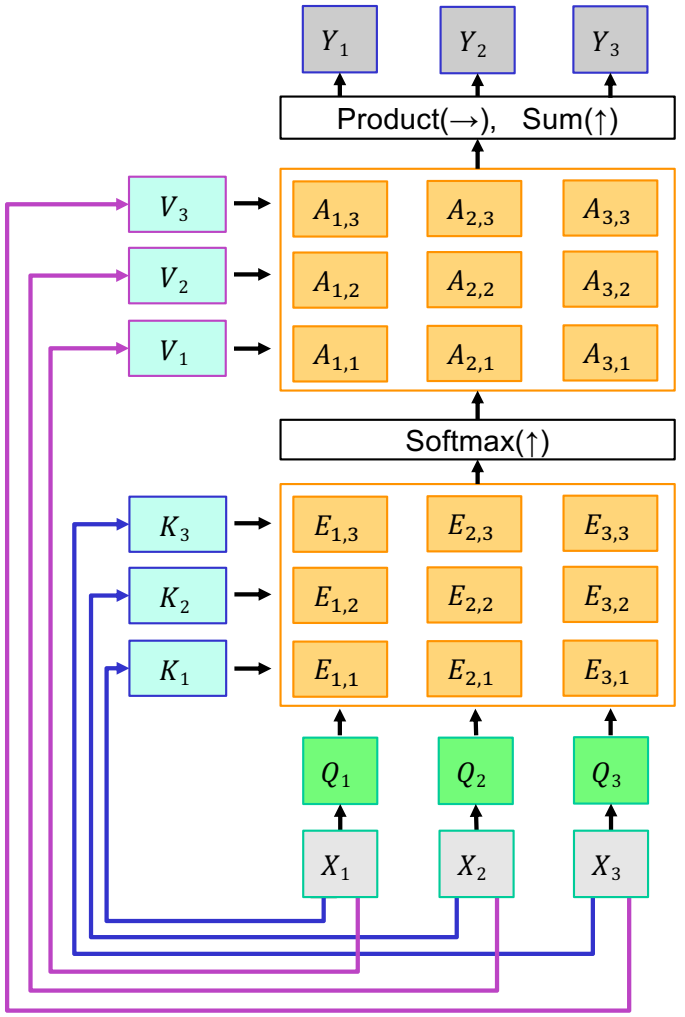
- Query vectors: $Q = XW_Q$
- Key vectors: $K = XW_K$
- Value vectors: $V = XW_V$
- Similarities: *scaled dot-product attention*

$$E_{i,j} = \frac{(Q_i \cdot K_j)}{\sqrt{D}} \text{ or } E = QK^T/\sqrt{D}$$

(D is the dimensionality of the keys)

- Attn. weights: $A = \text{softmax}(E, \text{dim} = 1)$
- Output vectors:

$$Y_i = \sum_j A_{i,j} V_j \text{ or } Y = AV$$

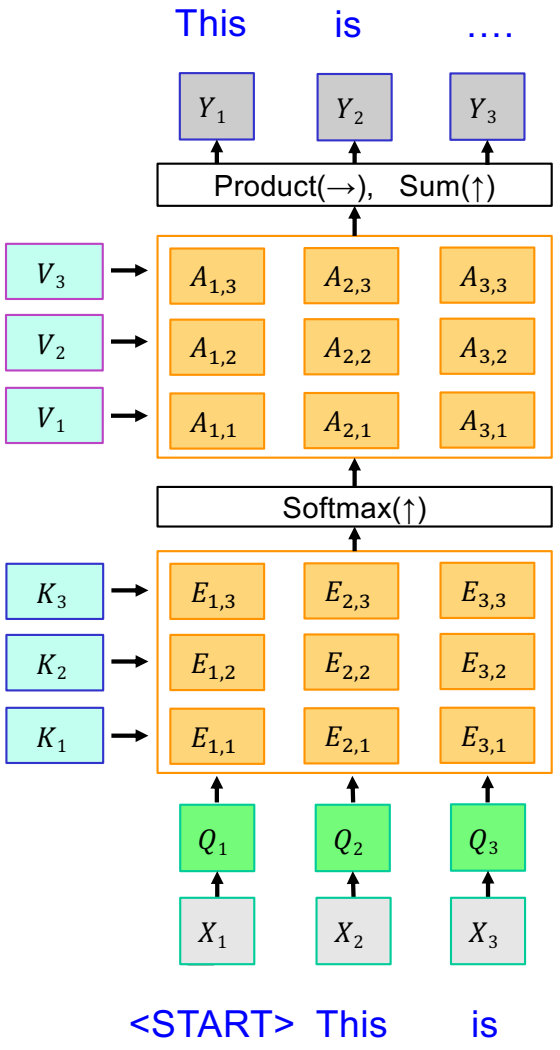


One query per input vector

Adapted from J. Johnson

Masked self-attention layer

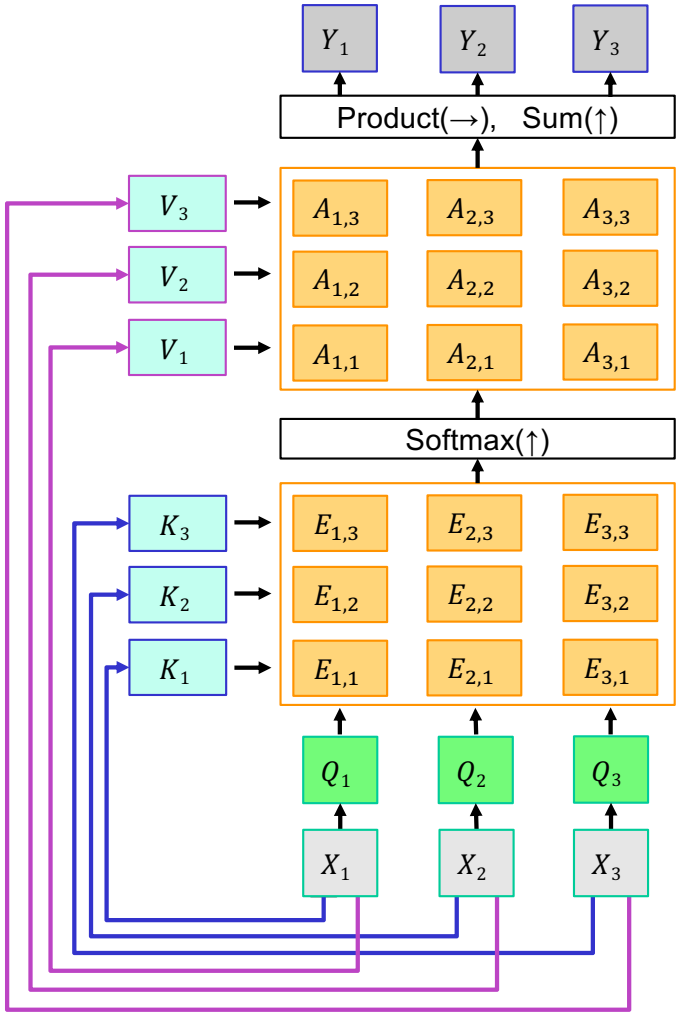
- The decoder should not “look ahead” in the output sequence



Adapted from J. Johnson

Masked self-attention layer

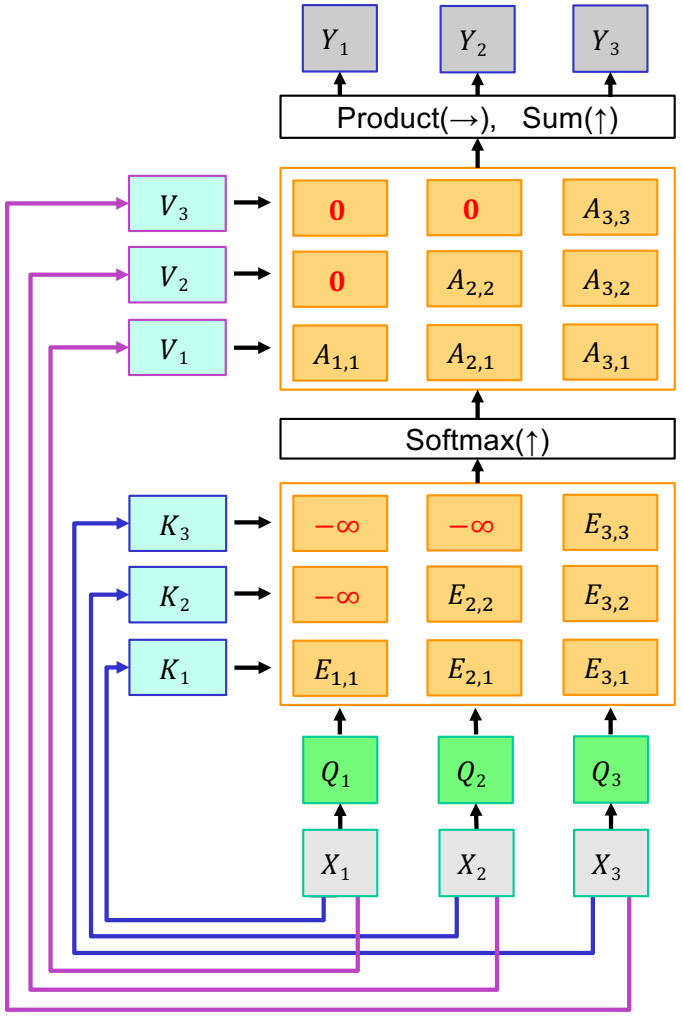
- The decoder should not “look ahead” in the output sequence



Adapted from J. Johnson

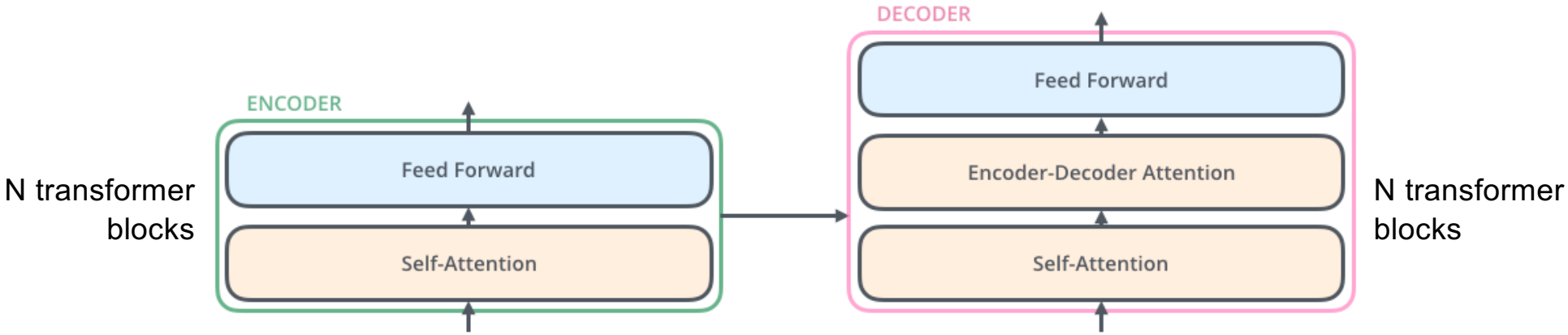
Masked self-attention layer

- The decoder should not “look ahead” in the output sequence



Adapted from J. Johnson

Attention mechanisms: Summary

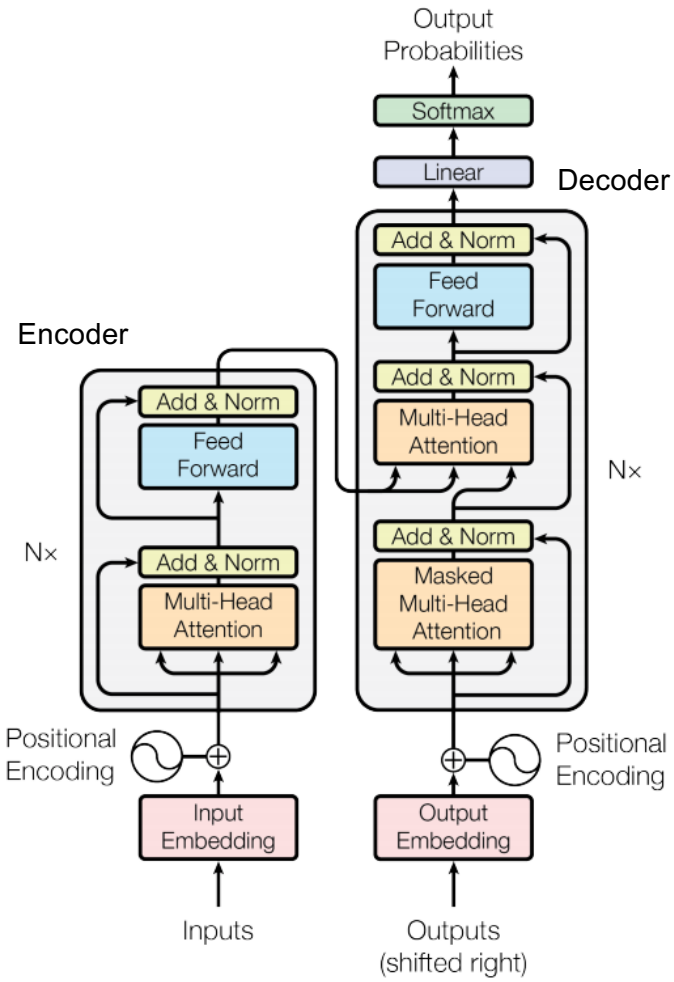


- **Encoder self-attention:** queries, keys, and values come from previous layer of encoder
- **Decoder self-attention:** values corresponding to future decoder outputs are masked out
- **Encoder-decoder attention:** queries come from previous decoder layer, keys and values come from output of encoder

Attention mechanisms: Illustration

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformer architecture: Details



A. Vaswani et al., Attention is all you need, NeurIPS 2017

Positional encoding

- To give transformer information about ordering of tokens, add function of position (based on sines and cosines) to every input

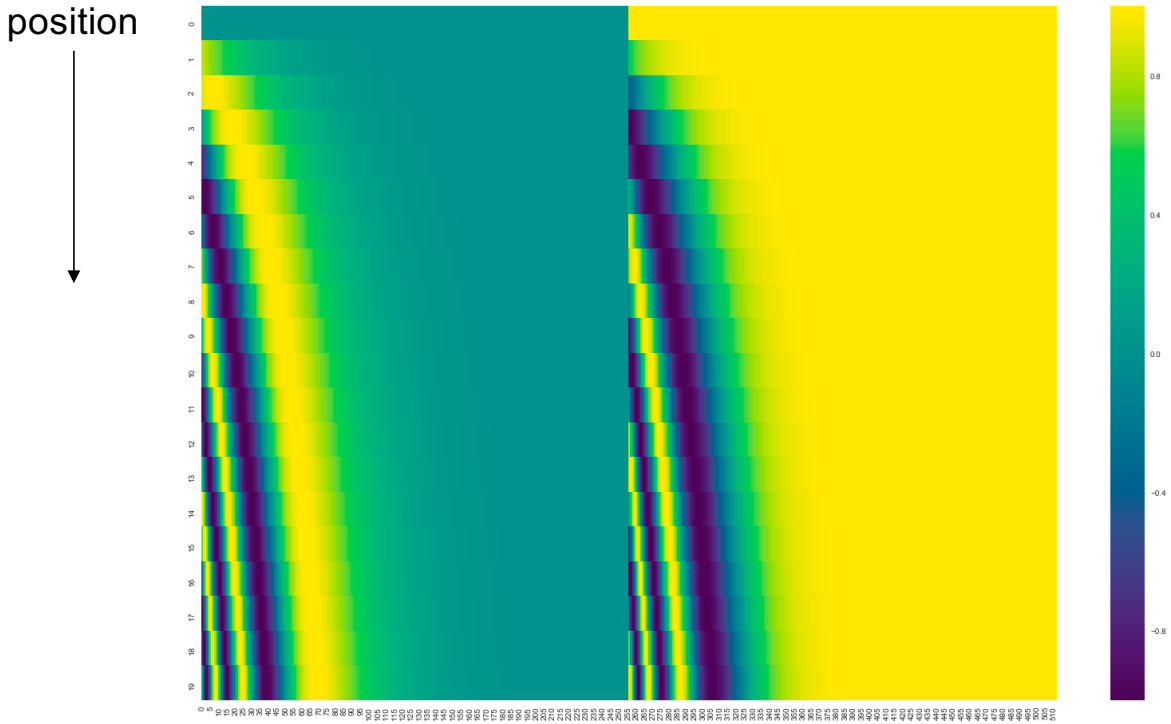
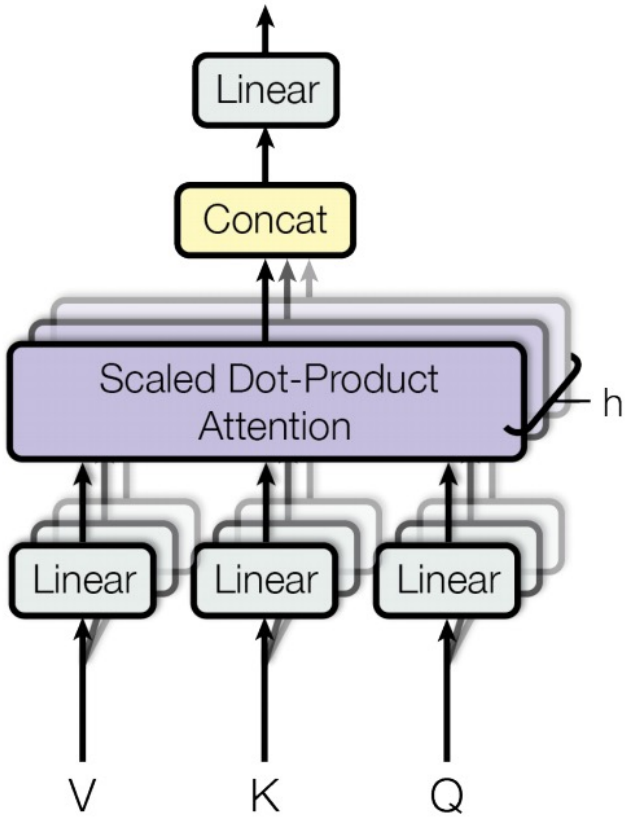


Image source

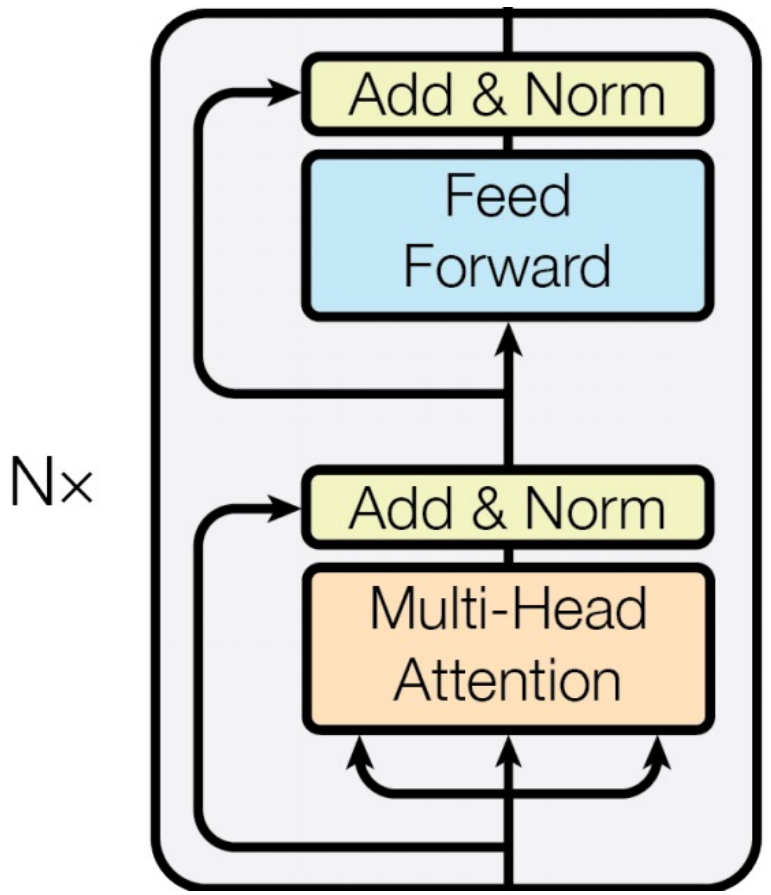
Multi-head attention

- Run h attention models in parallel on top of different linearly projected versions of Q, K, V; concatenate and linearly project the results
- Intuition: enables model to attend to different kinds of information at different positions (see visualization tool)

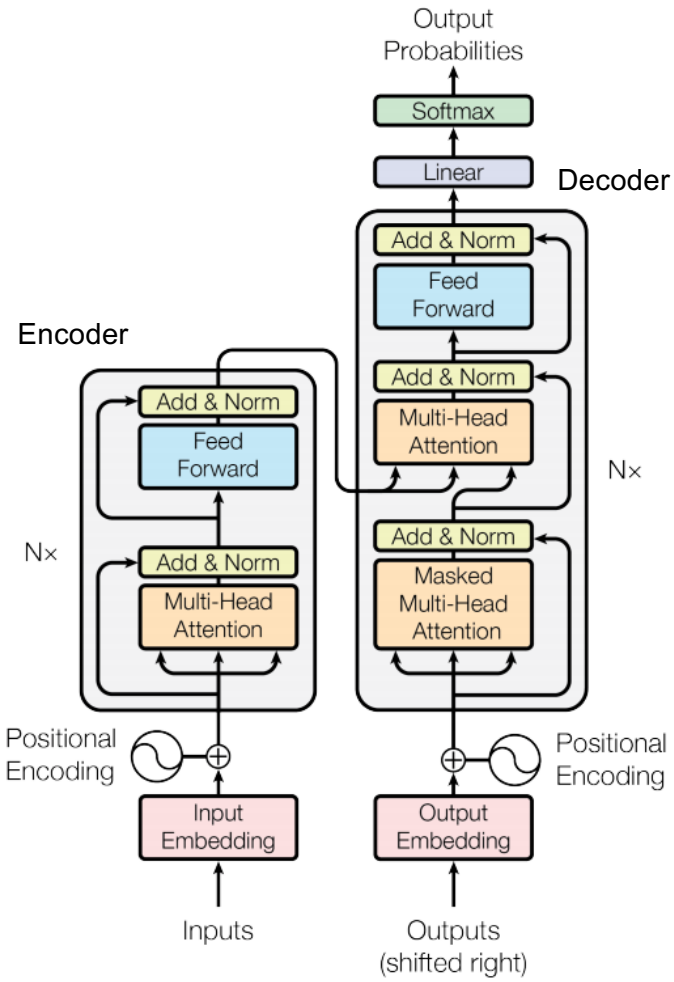


Transformer blocks

- A **Transformer** is a sequence of transformer blocks
  - Vaswani et al.: N=12 blocks, embedding dimension = 512, 6 attention heads
  - **Add & Norm:** residual connection followed by layer normalization
  - **Feedforward:** two linear layers with ReLUs in between, applied independently to each vector
- Attention is the only interaction between inputs!

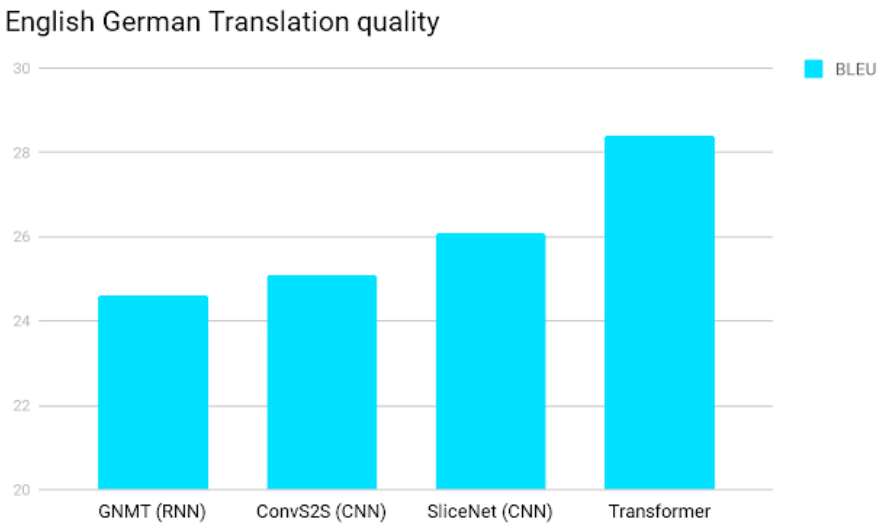


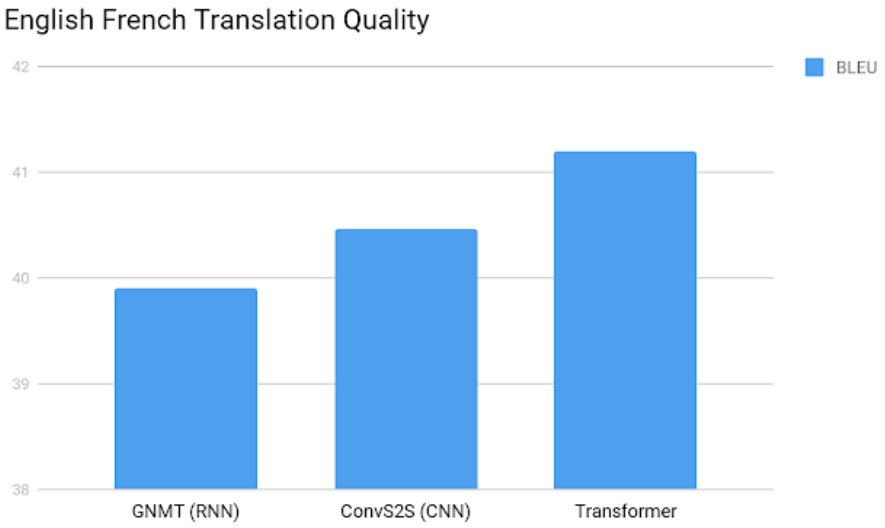
Transformer architecture: Zooming back out



A. Vaswani et al., Attention is all you need, NeurIPS 2017

Results

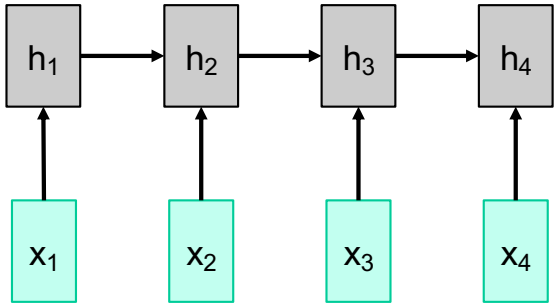




https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformers: Pros and cons

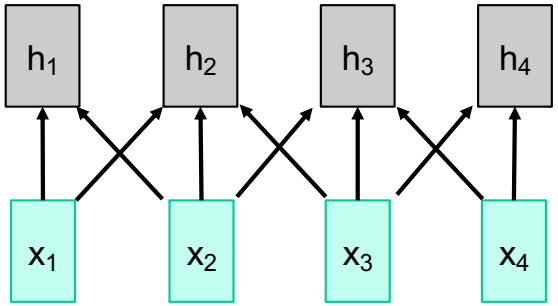
RNNs



Works on **ordered sequences**

- Pros: Not limited by fixed context size (in principle): 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

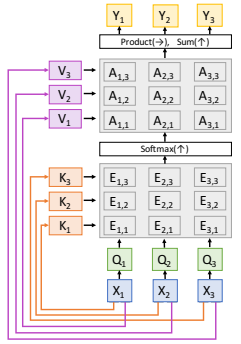
1D convolutional networks



Works on **multidimensional grids**

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

Transformers



- 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: cost of attention operator is *quadratic* in input size

Making transformers more efficient

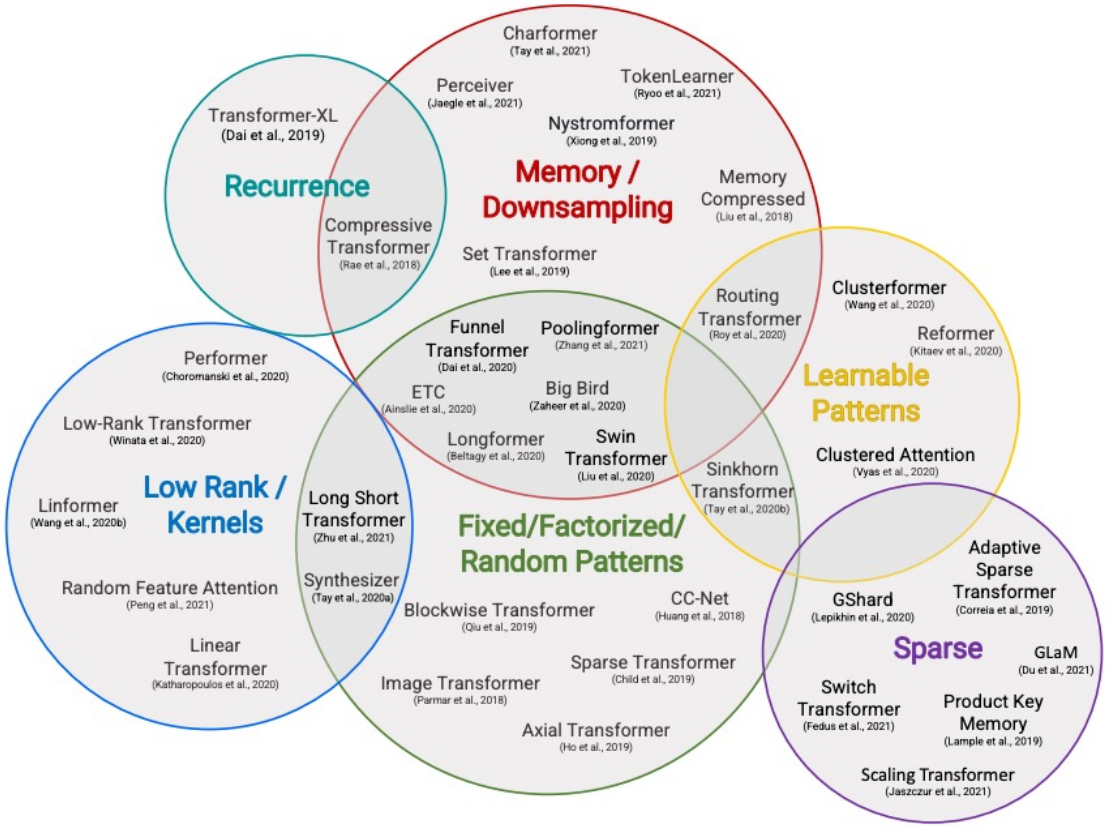


Figure 2: Taxonomy of Efficient Transformer Architectures.

Y. Tay et al. Efficient transformers: A survey. arXiv 2022