The Transformer: Attention is All You Need

Hyungu Kahng

2019.11.23

@ DMQA Open Seminar

Contents

- Seq2seq
- Seq2seq with Attention
- Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

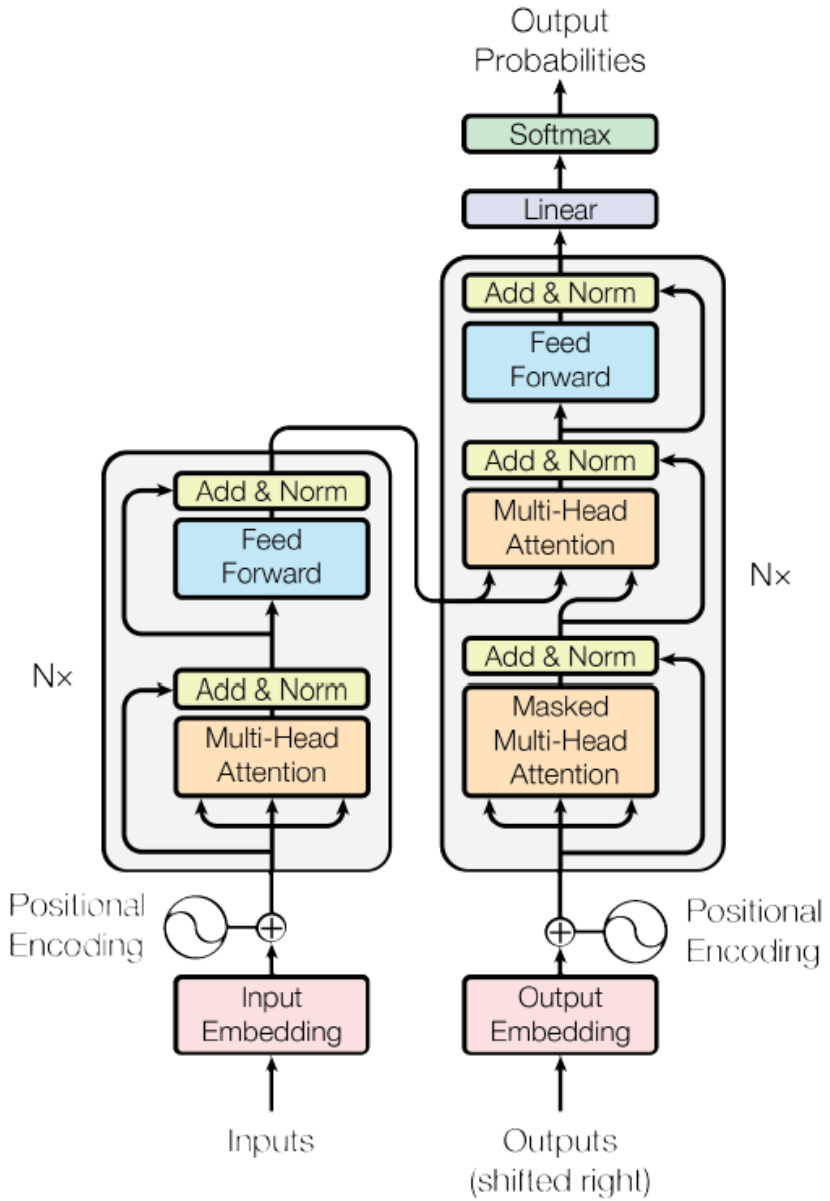
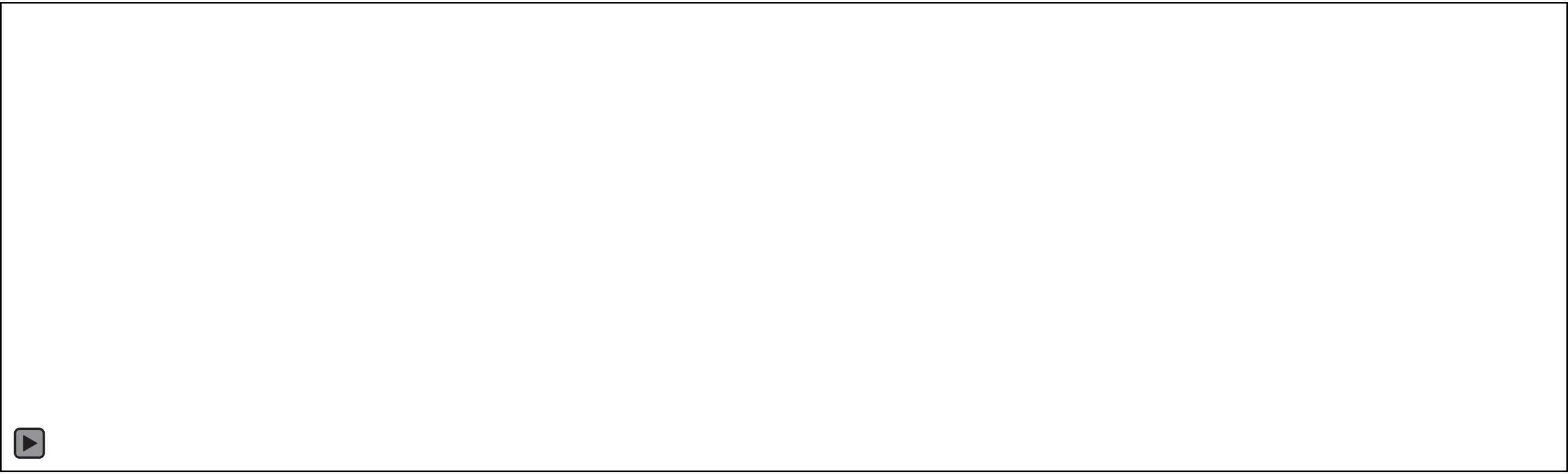


Figure 1: The Transformer - model architecture.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*(pp. 5998-6008).

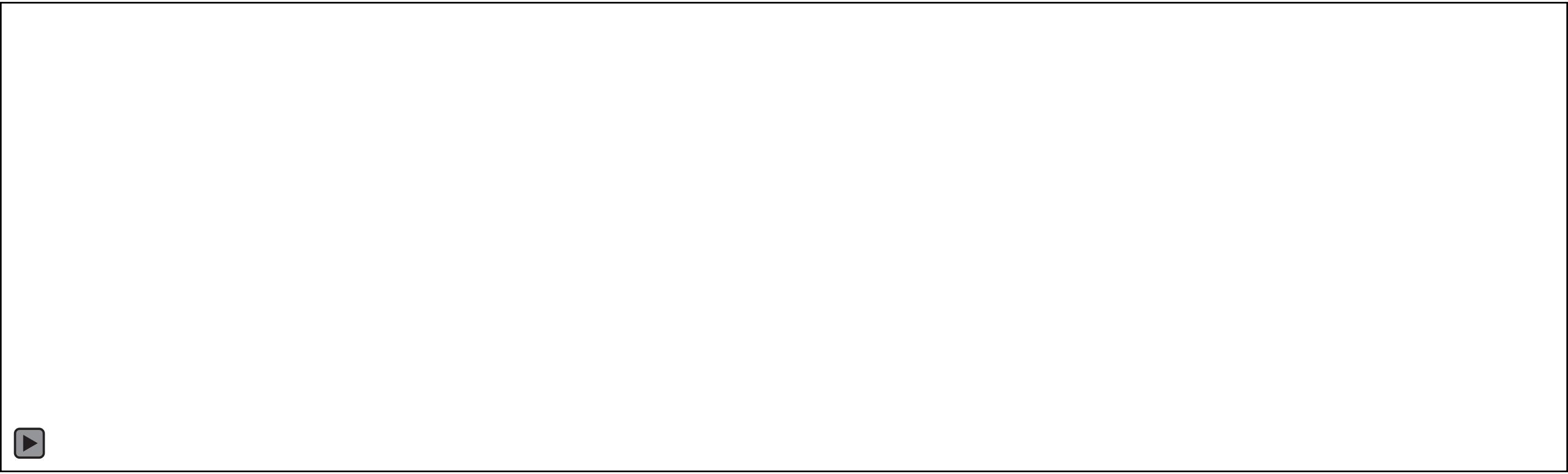
Seq2seq

Basics



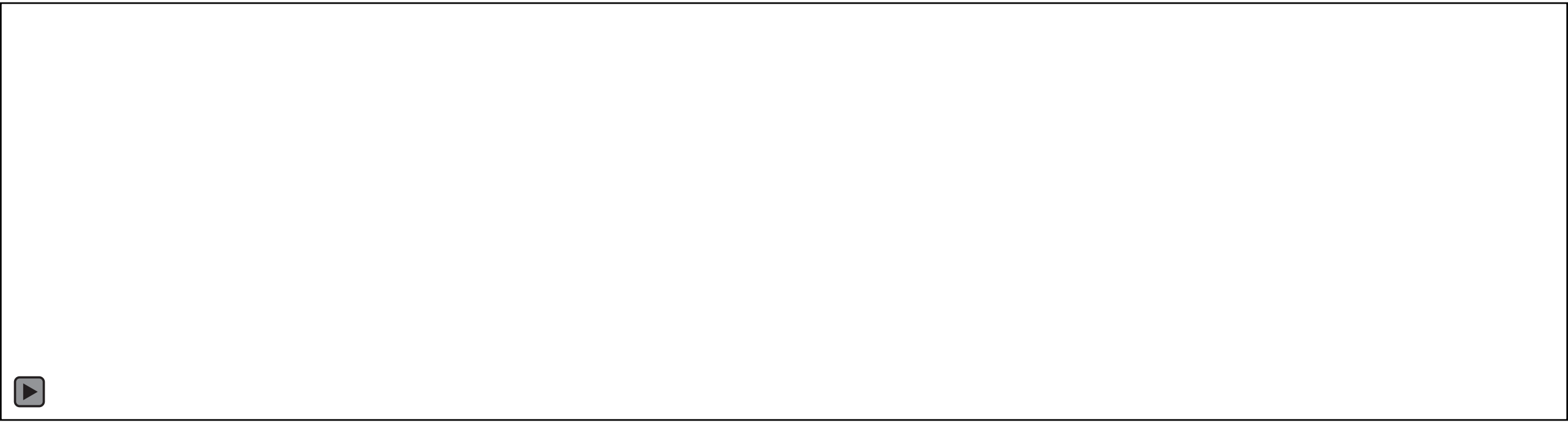
Seq2seq

Basics



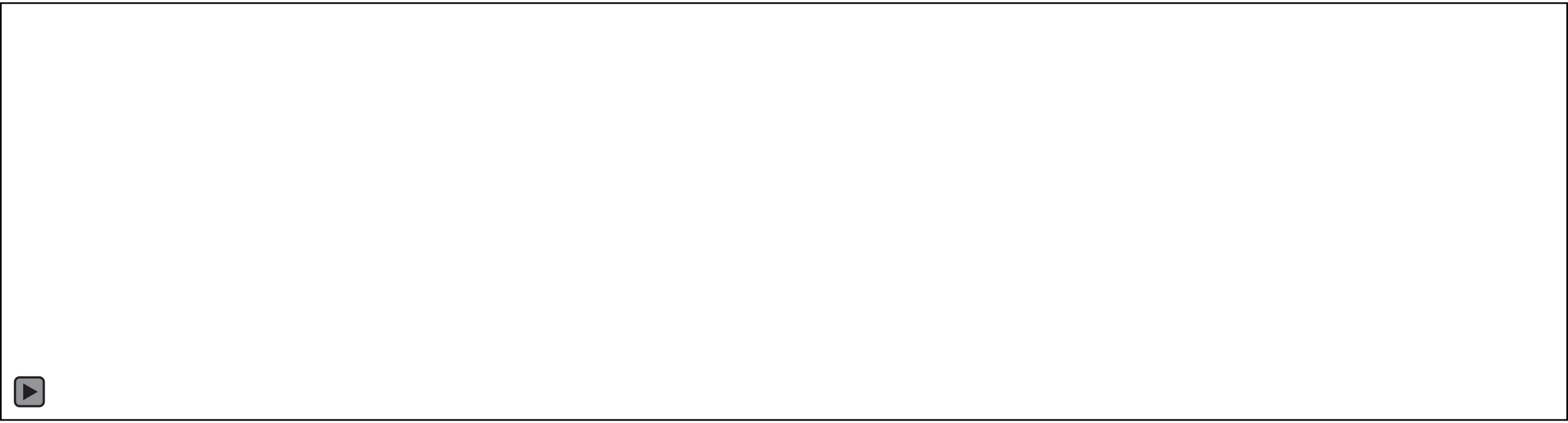
Seq2seq

Basics



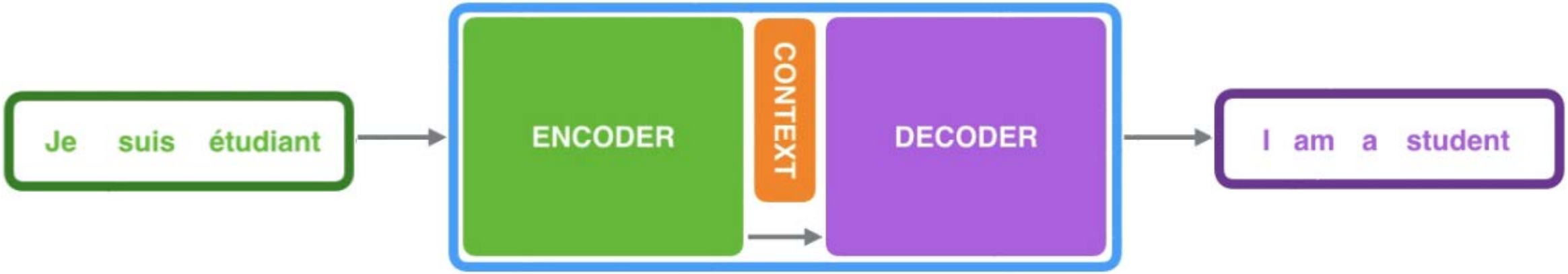
Seq2seq

Basics



Seq2seq

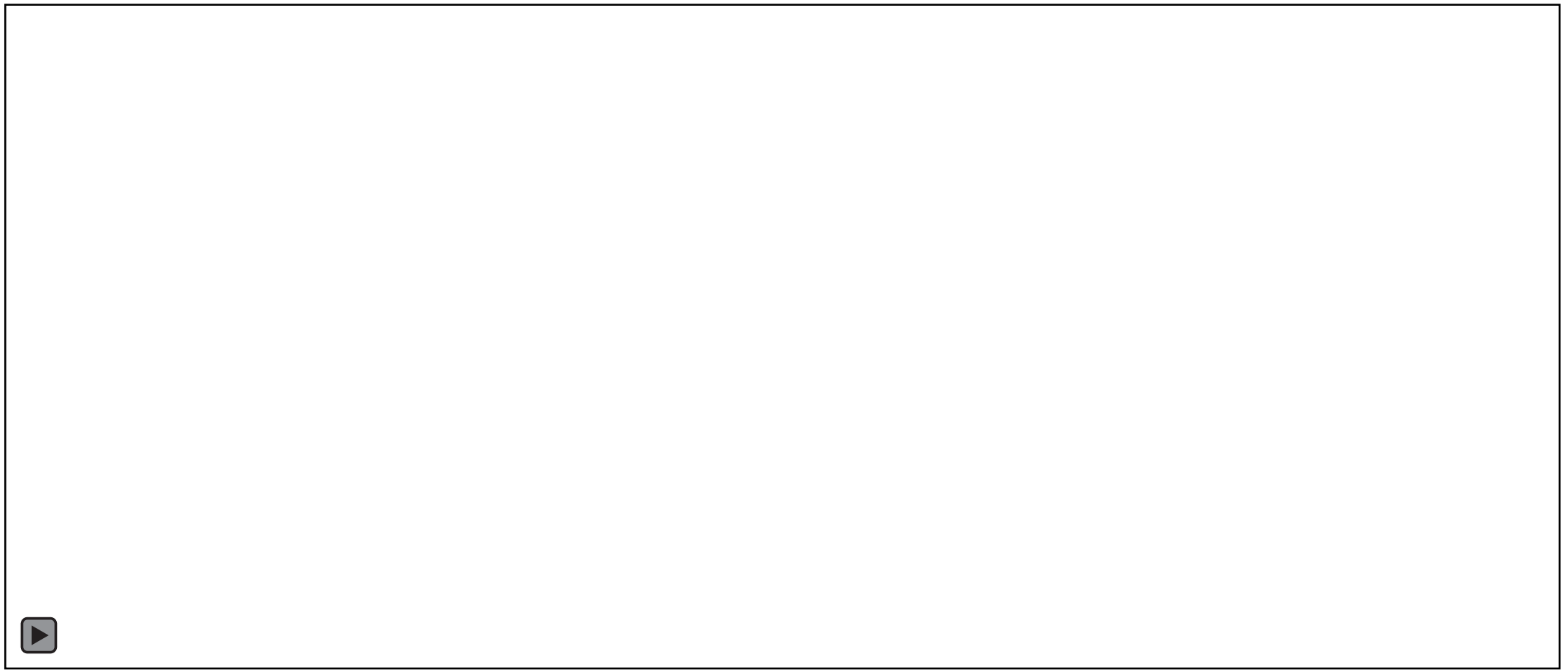
Basics



The context vector is a compressed encoding of the input sequence.
(= thought vector)

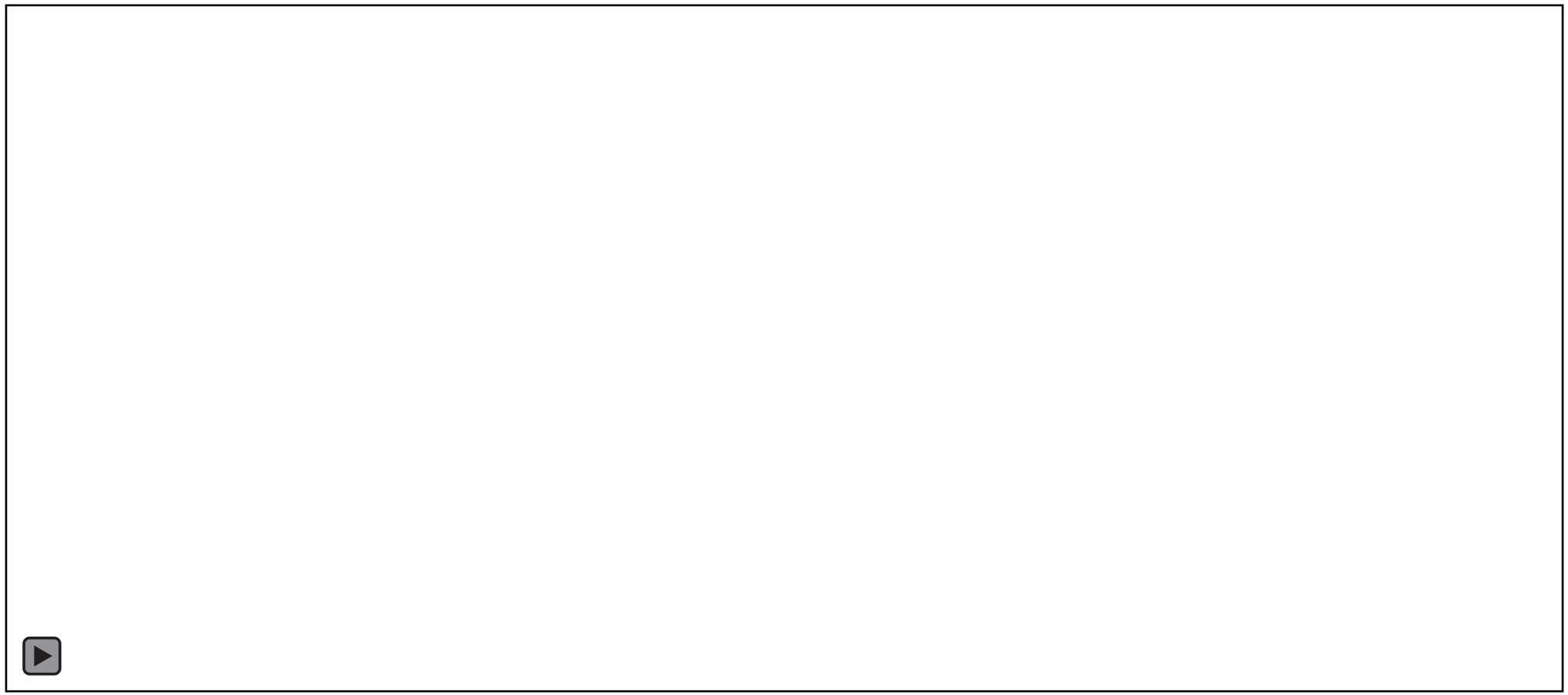
Seq2seq

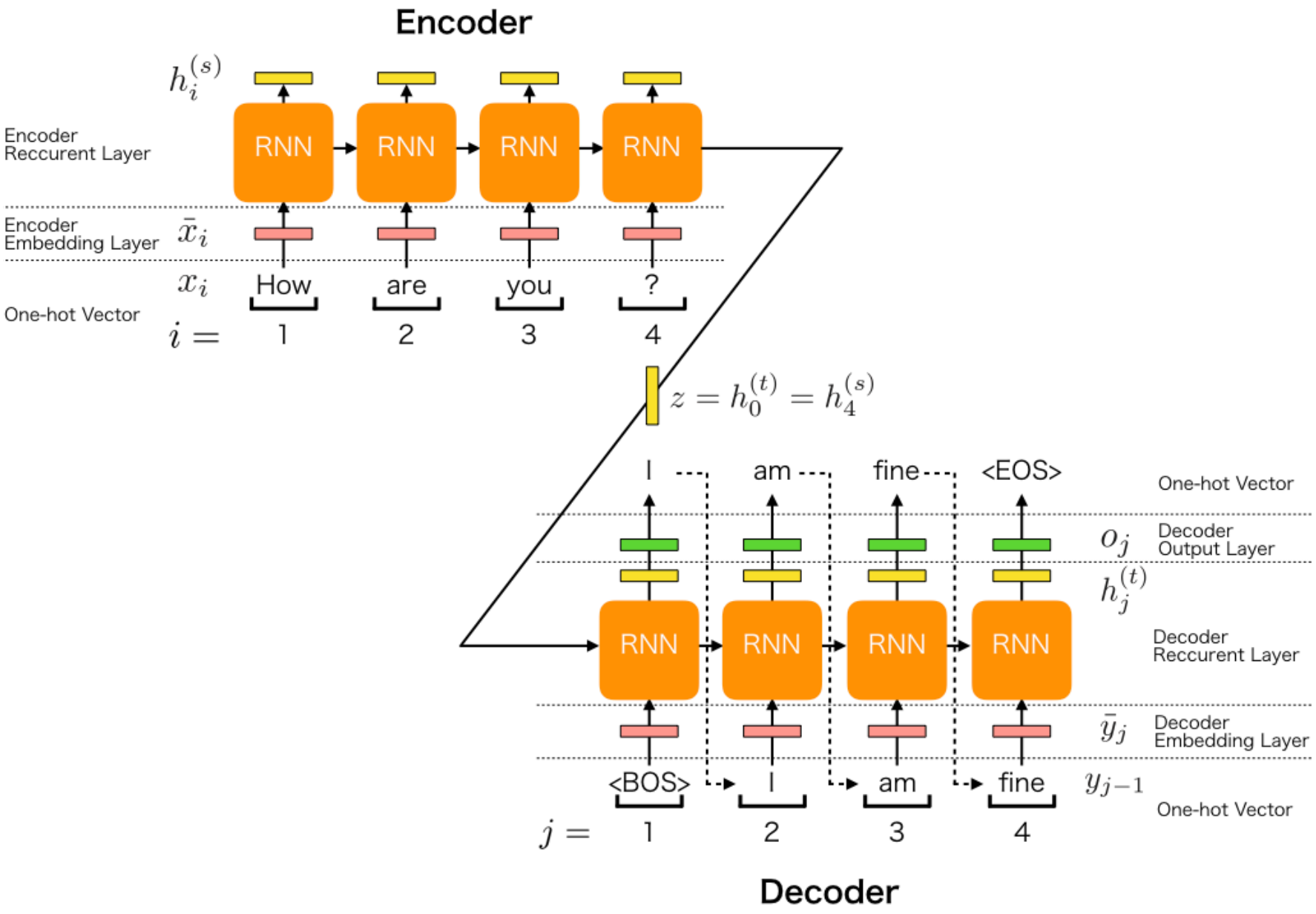
Basics



Seq2seq

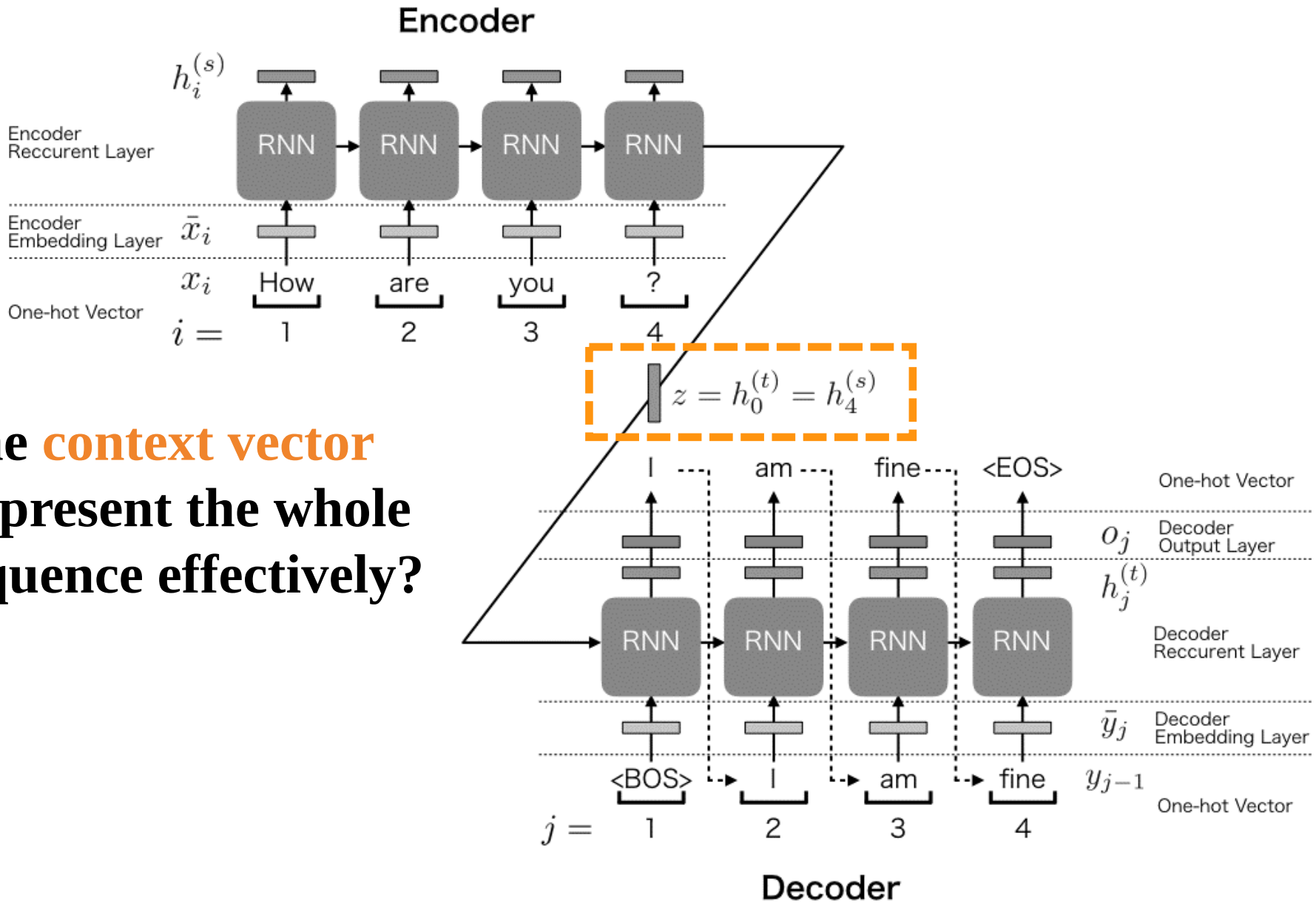
Basics





http://docs.chainer.org/en/stable/examples/seq2seq.html

Can the context vector solely represent the whole input sequence effectively?



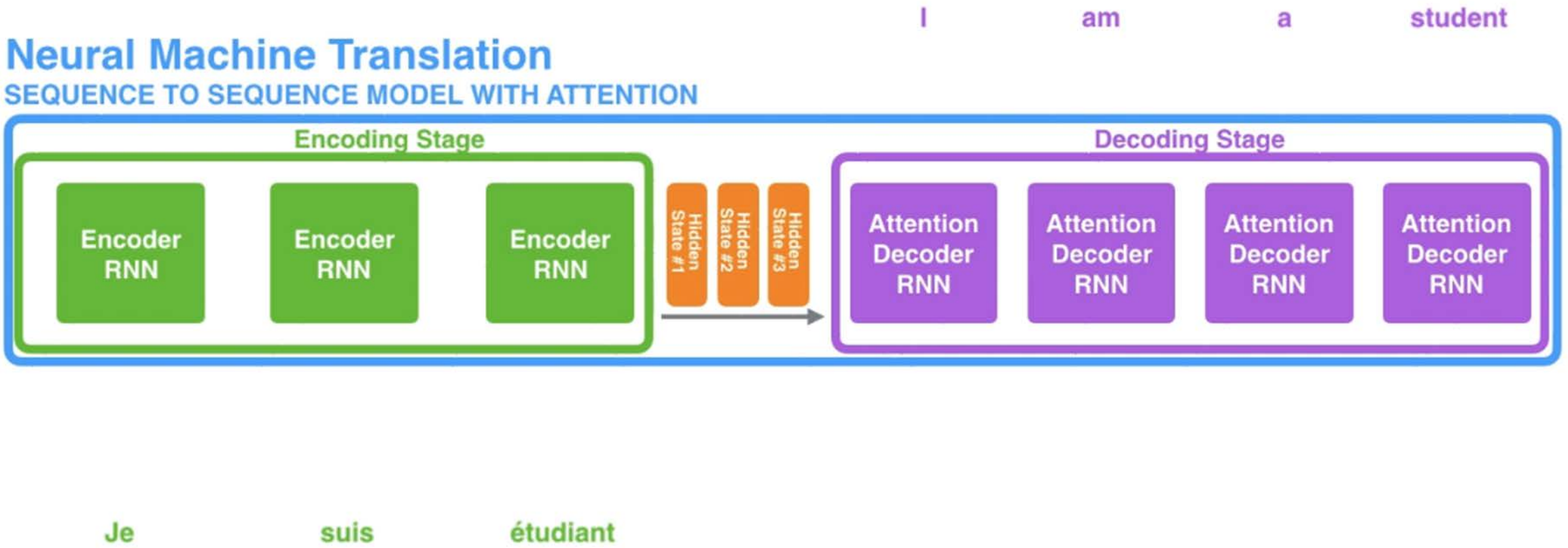
http://docs.chainer.org/en/stable/examples/seq2seq.html

Seq2seq with Attention

Basics (video)

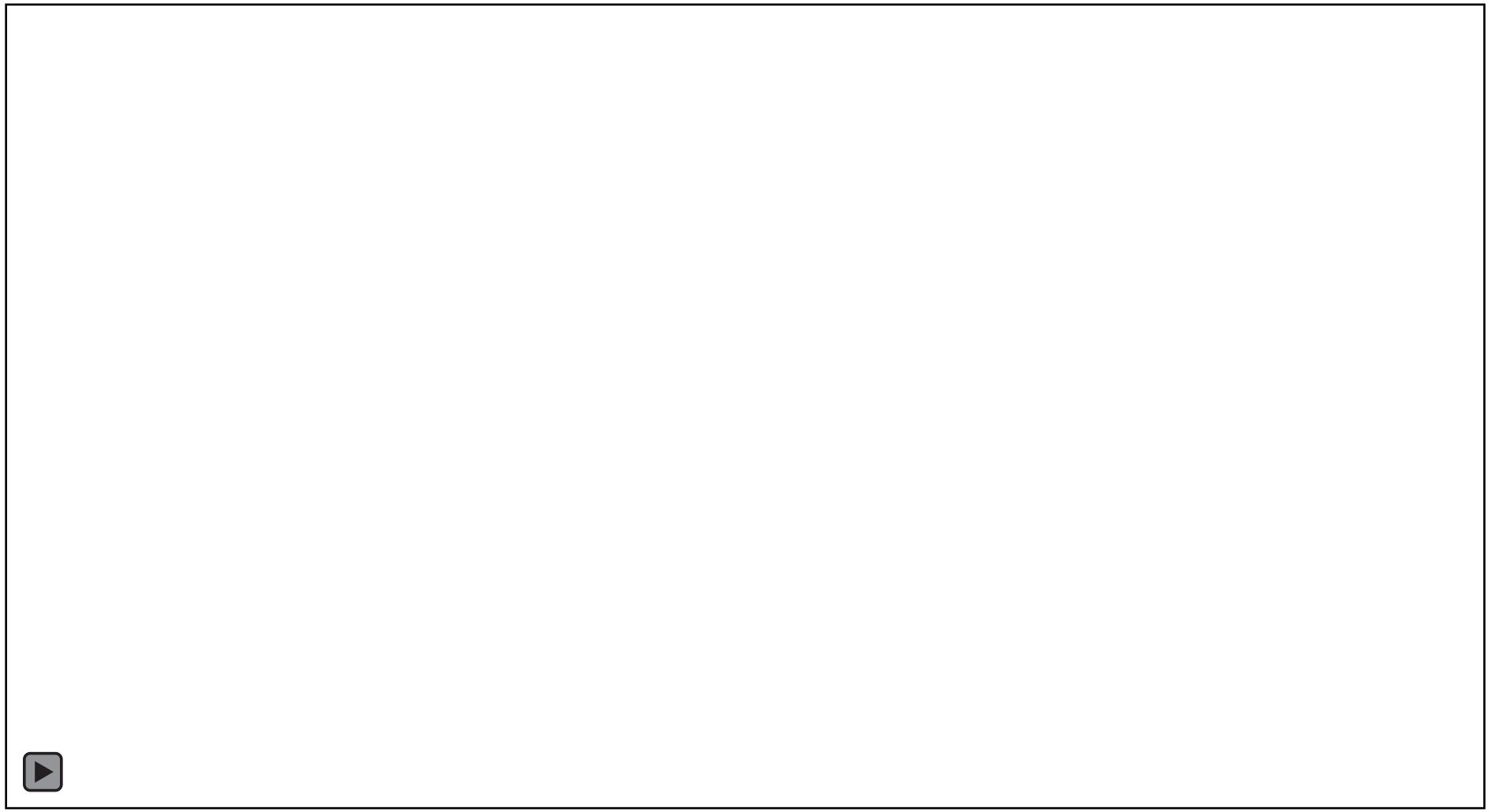
Seq2seq with Attention

Basics (image)



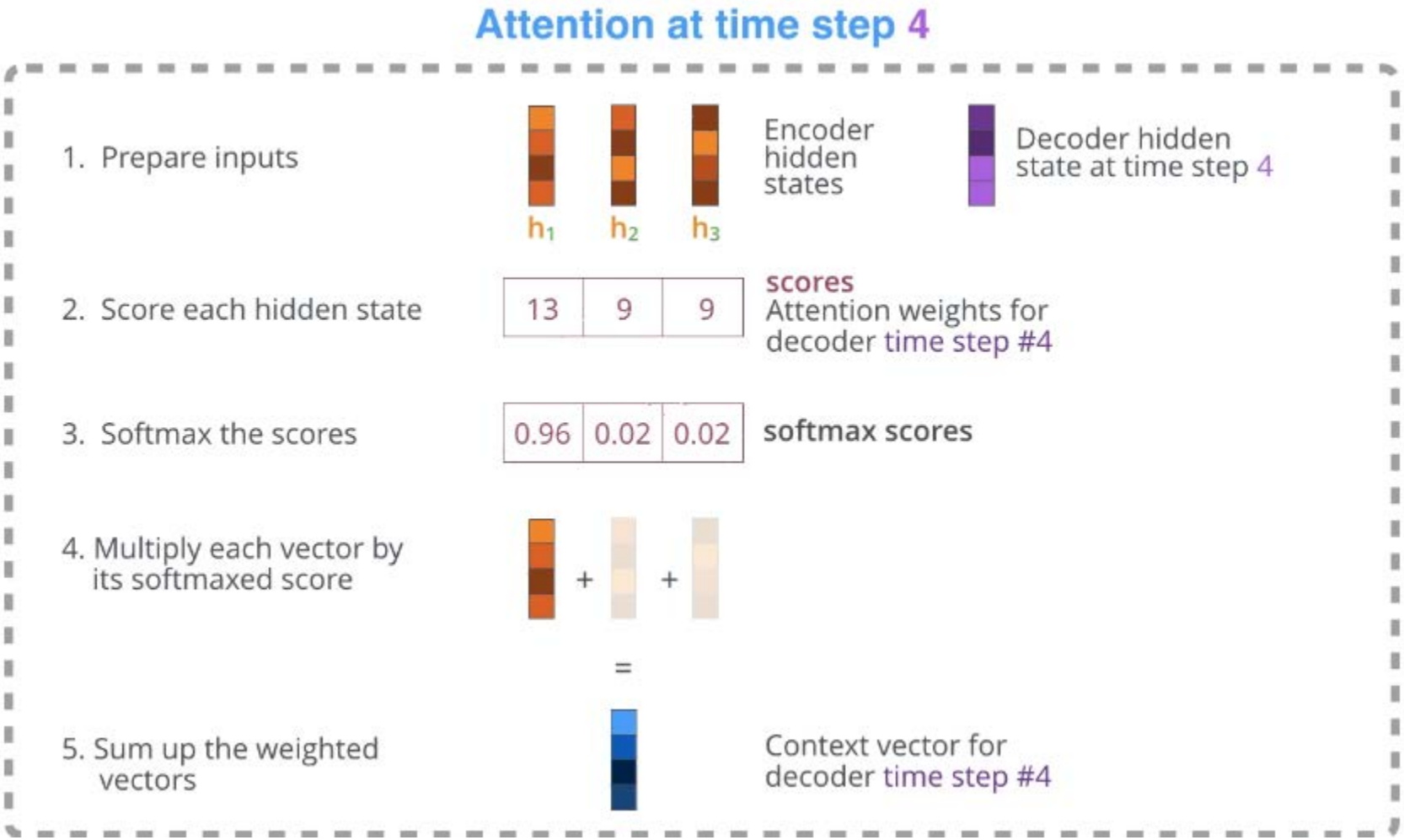
Seq2seq with Attention

Attention process (video)



Seq2seq with Attention

Attention process (image)



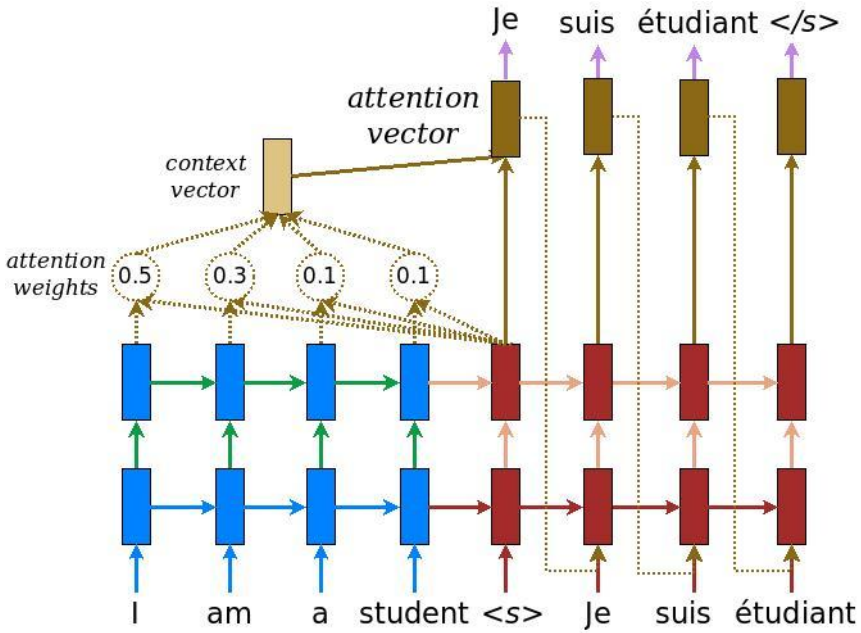
Seq2seq with Attention

Attention process (image)

How can we compute the score for each hidden state?

Seq2seq with Attention

Attention scores (equations)



$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

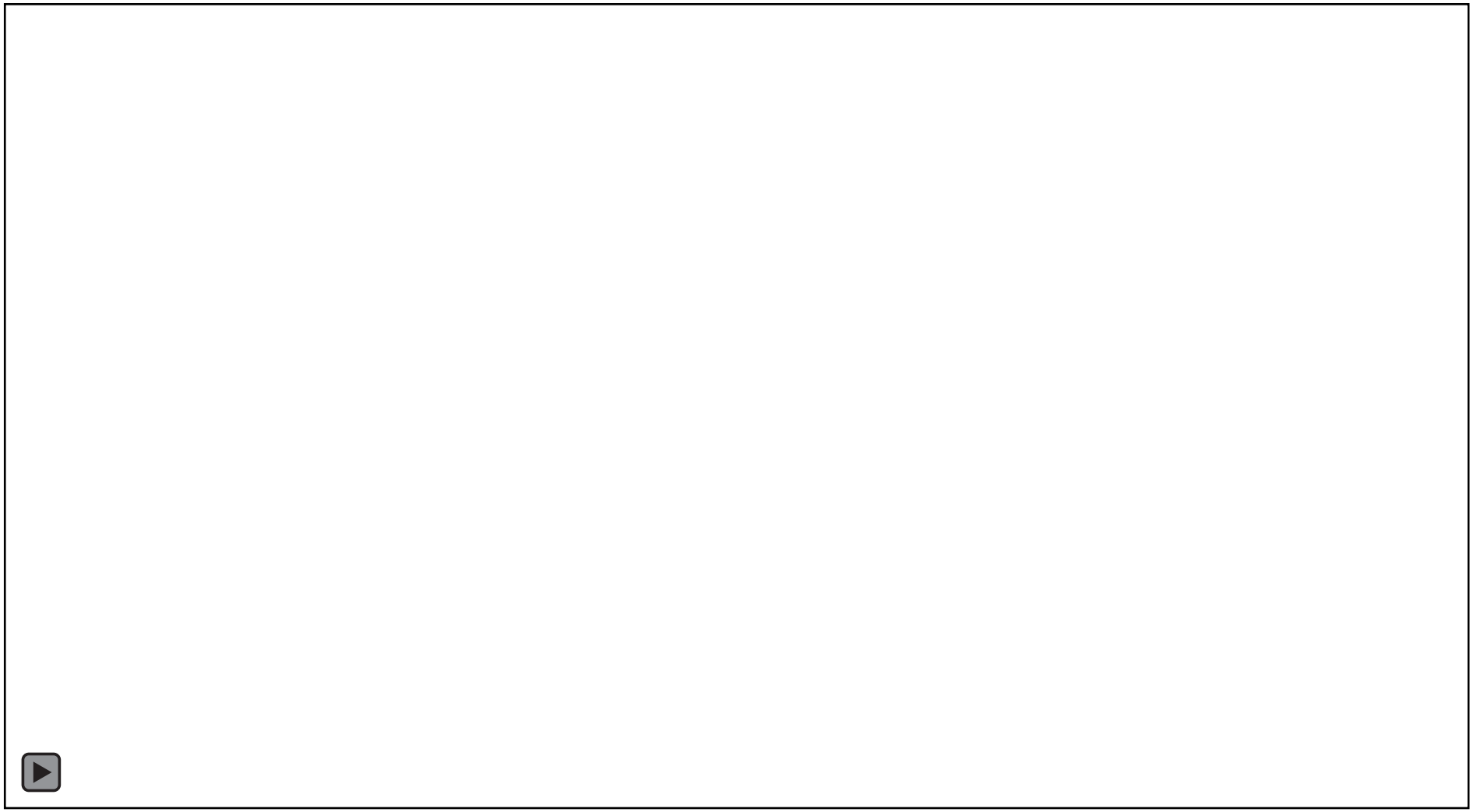
$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_1 \boldsymbol{h}_t + \boldsymbol{W}_2 \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \quad (4)$$

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

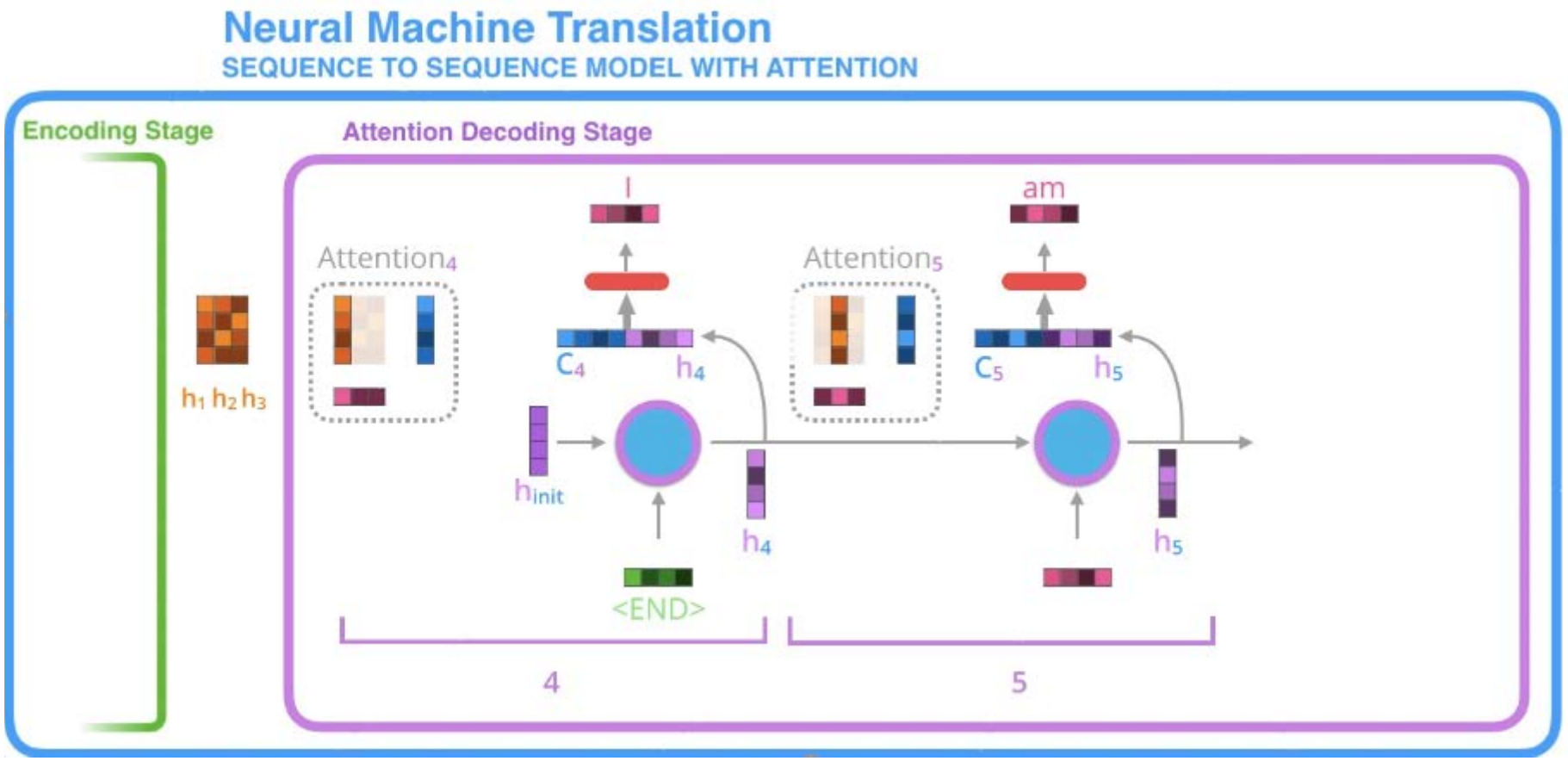
Seq2seq with Attention

Attention process (video)



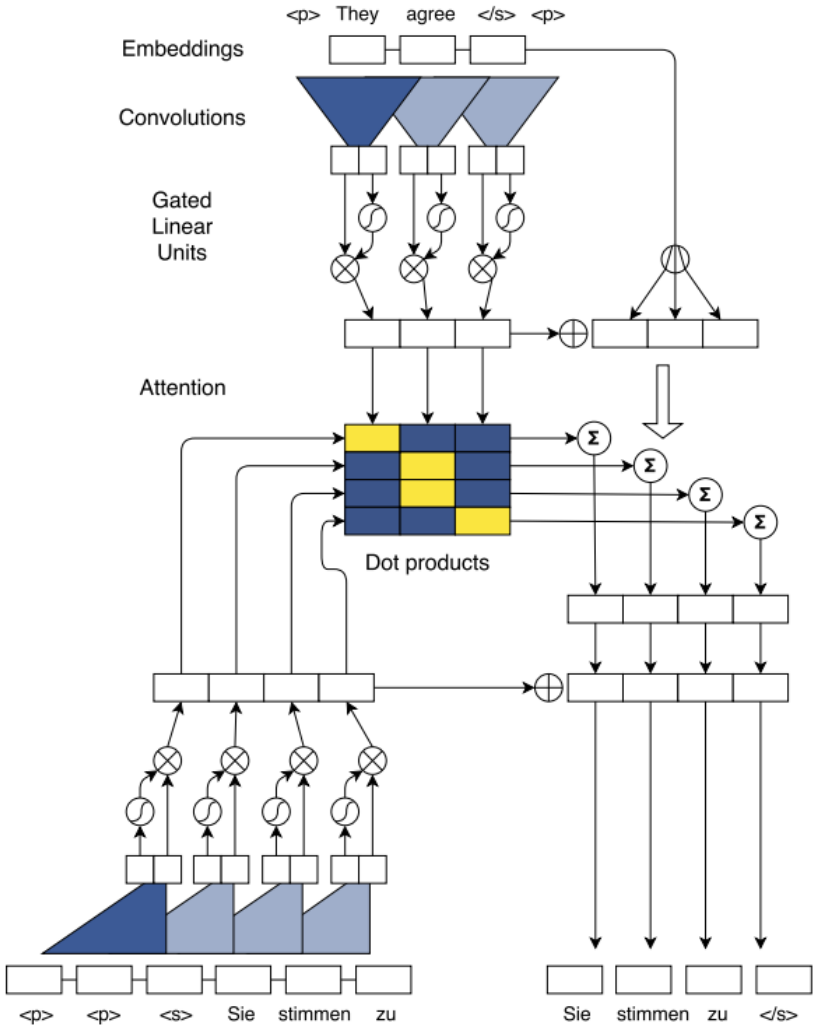
Seq2seq with Attention

Attention process (image)



ConvS2S

Gehring et al., Convolutional Sequence to Sequence Learning, 2017.

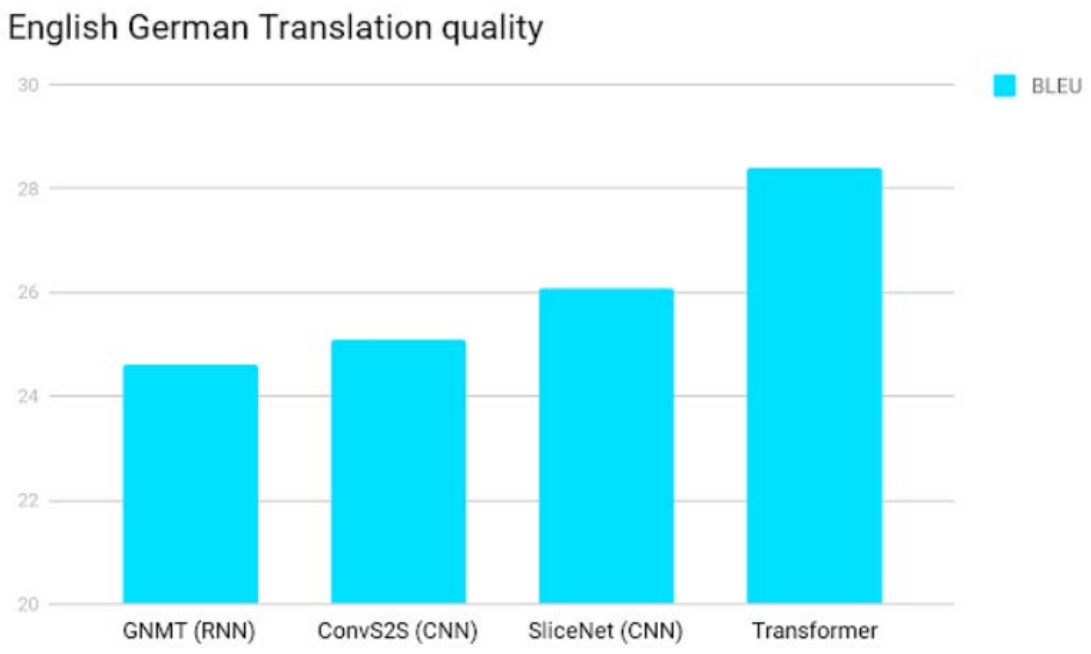


. la maison de Léa <end> .

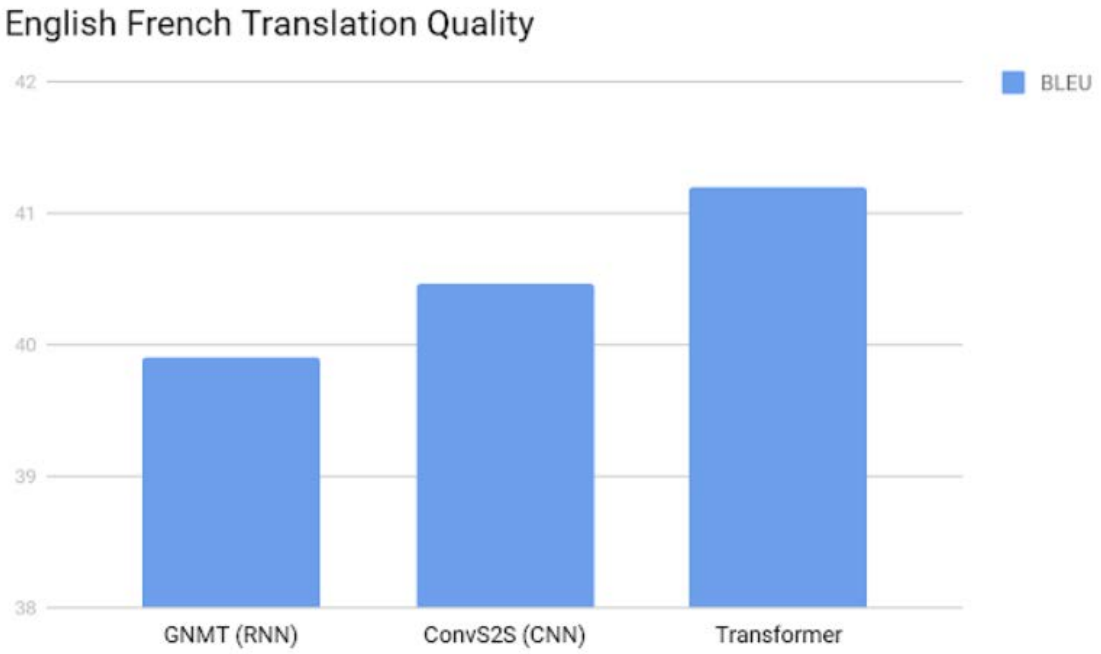
https://nlp.stanford.edu/seminar/details/mauli.pdf

Neural Machine Translation

Performance comparisons



BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to German translation benchmark.



BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to French translation benchmark.

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

Coreference Resolution

Example

The animal didn’t cross the street because it was too tired.

The animal didn’t cross the street because it was too wide.

Transformer

Overview (GIF)

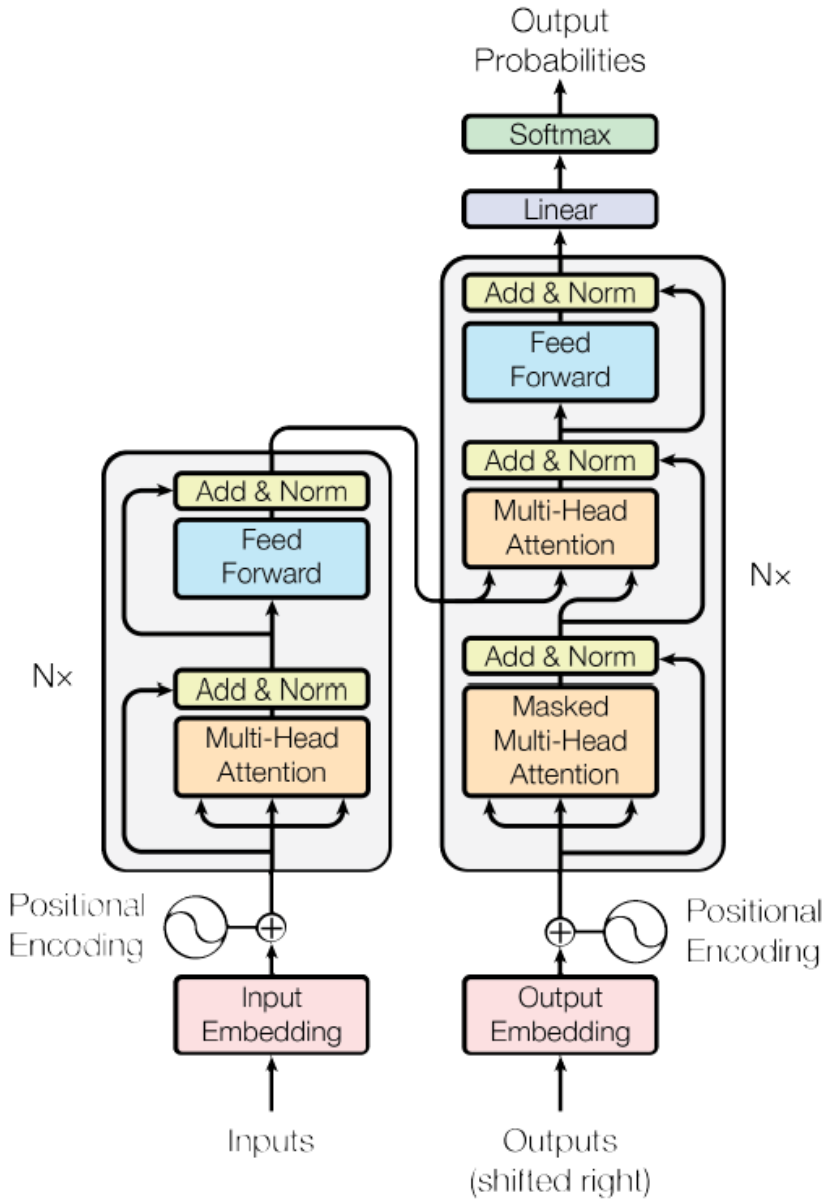


Figure 1: The Transformer - model architecture.

Transformer

Motivations & properties

- Efficient parallelization
- Reduce sequential computation
  - $\mathcal{O}(n),\ \mathcal{O}(logn) \rightarrow\ \boldsymbol{\mathcal{O}(1)}$
- Self-attention
- Encoder-decoder attention

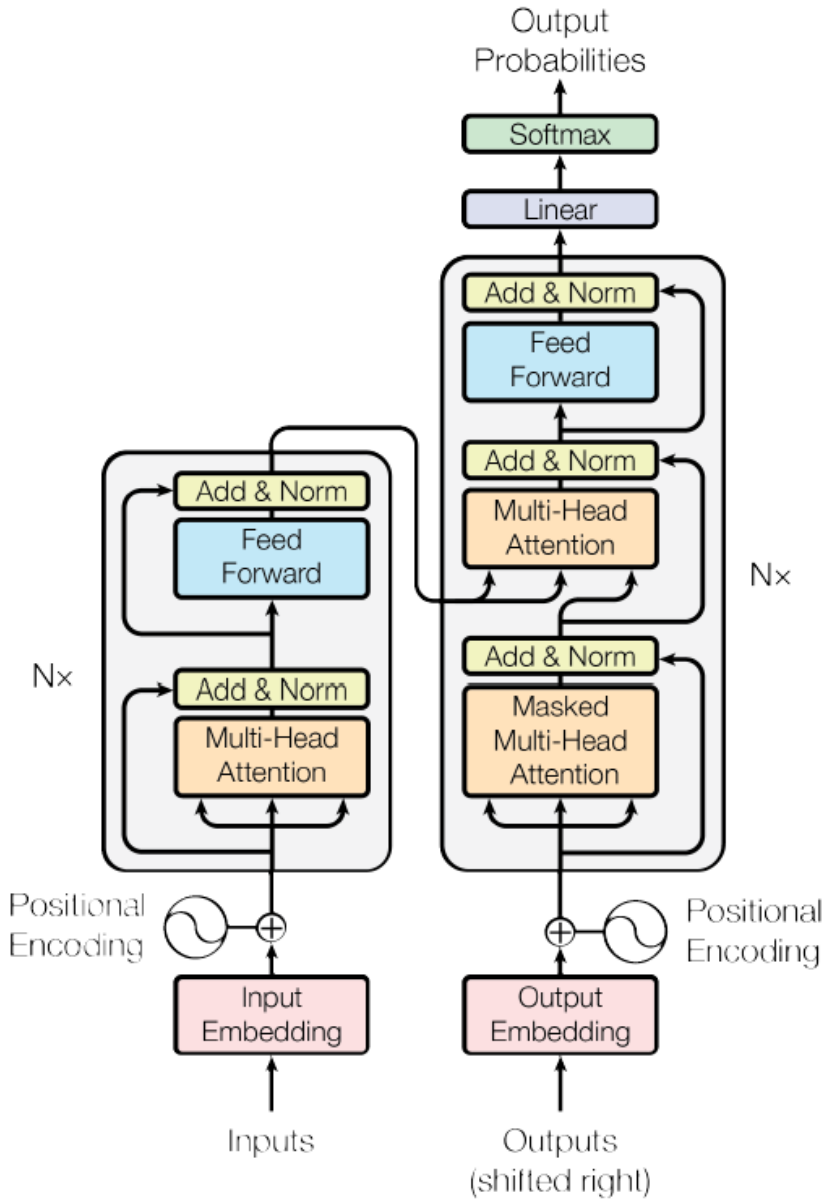
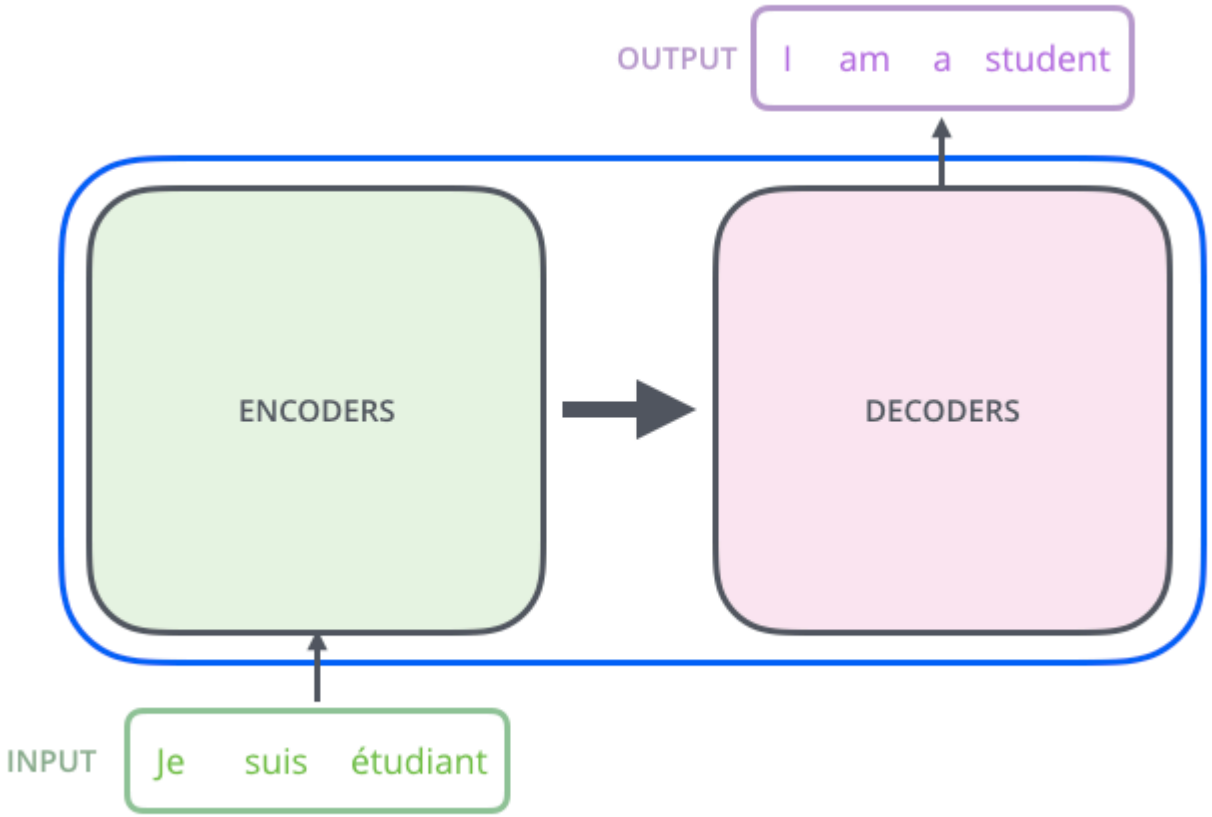


Figure 1: The Transformer - model architecture.

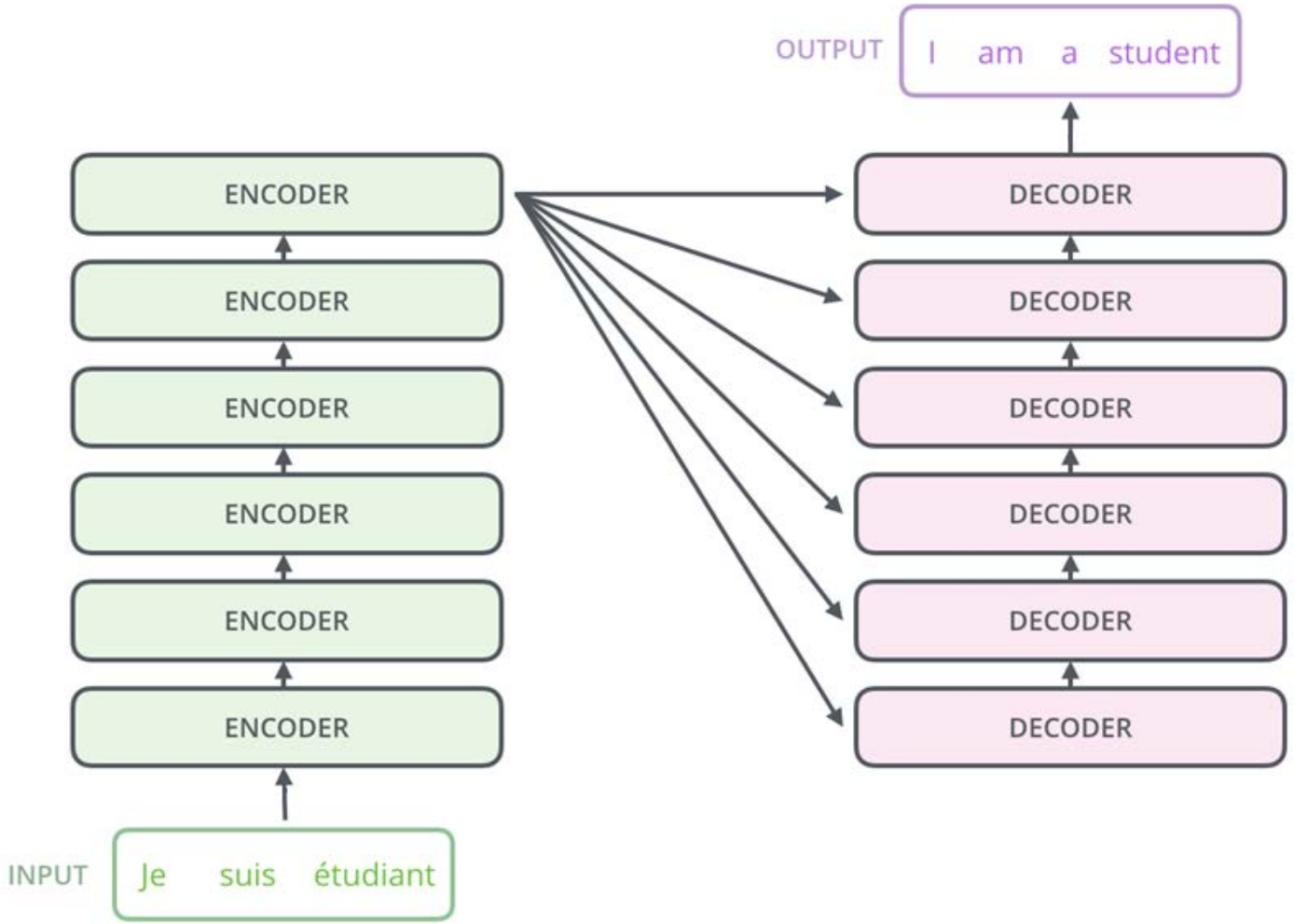
Transformer

Recap: encoder-decoder architecture



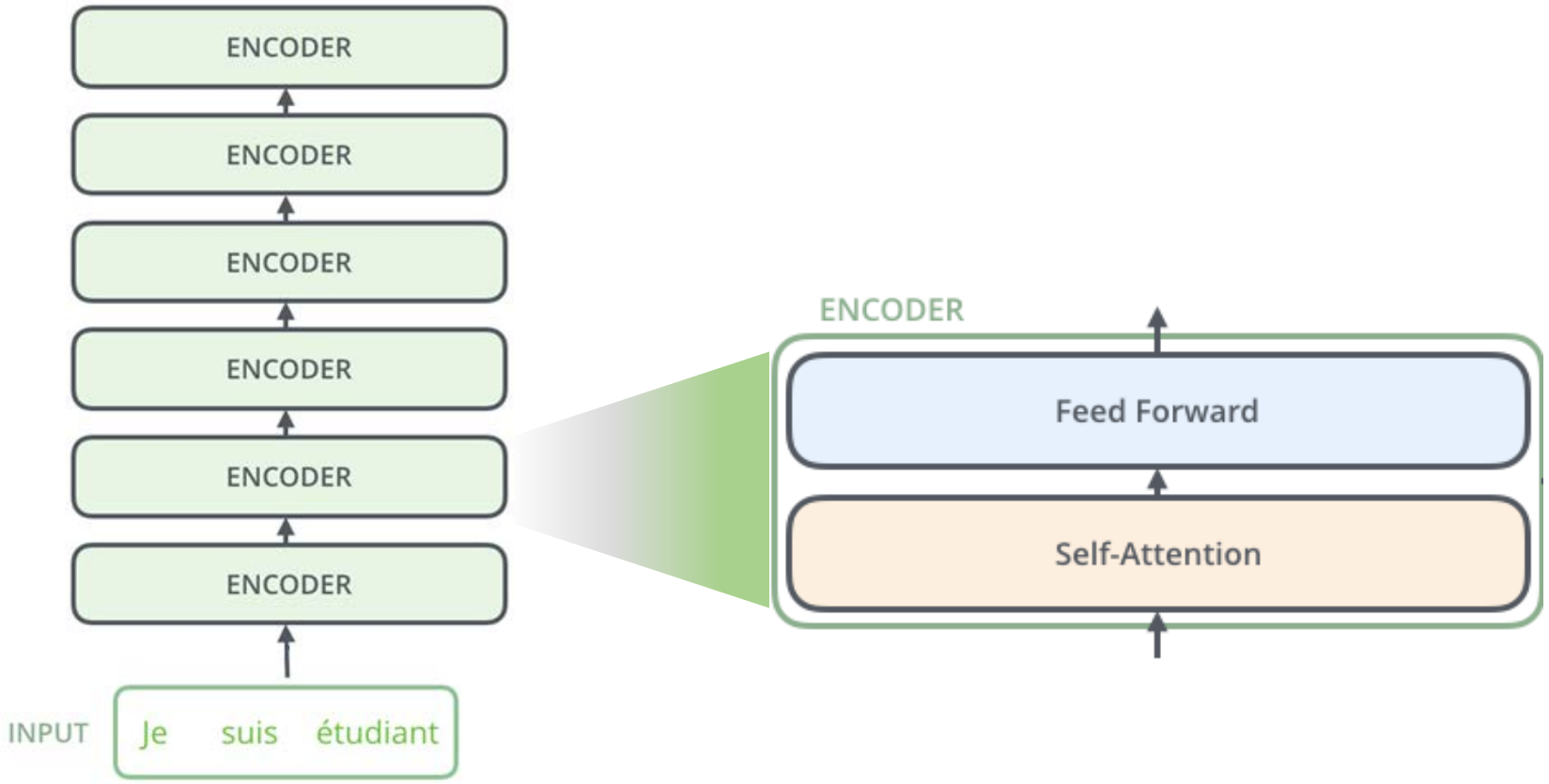
Transformer

Architecture (overview)



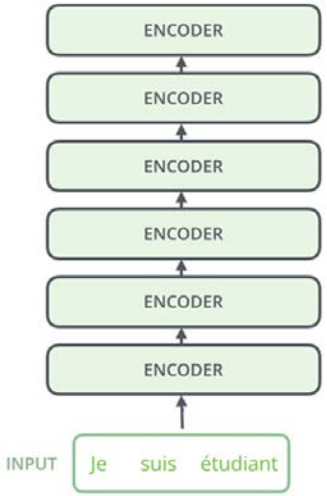
Transformer

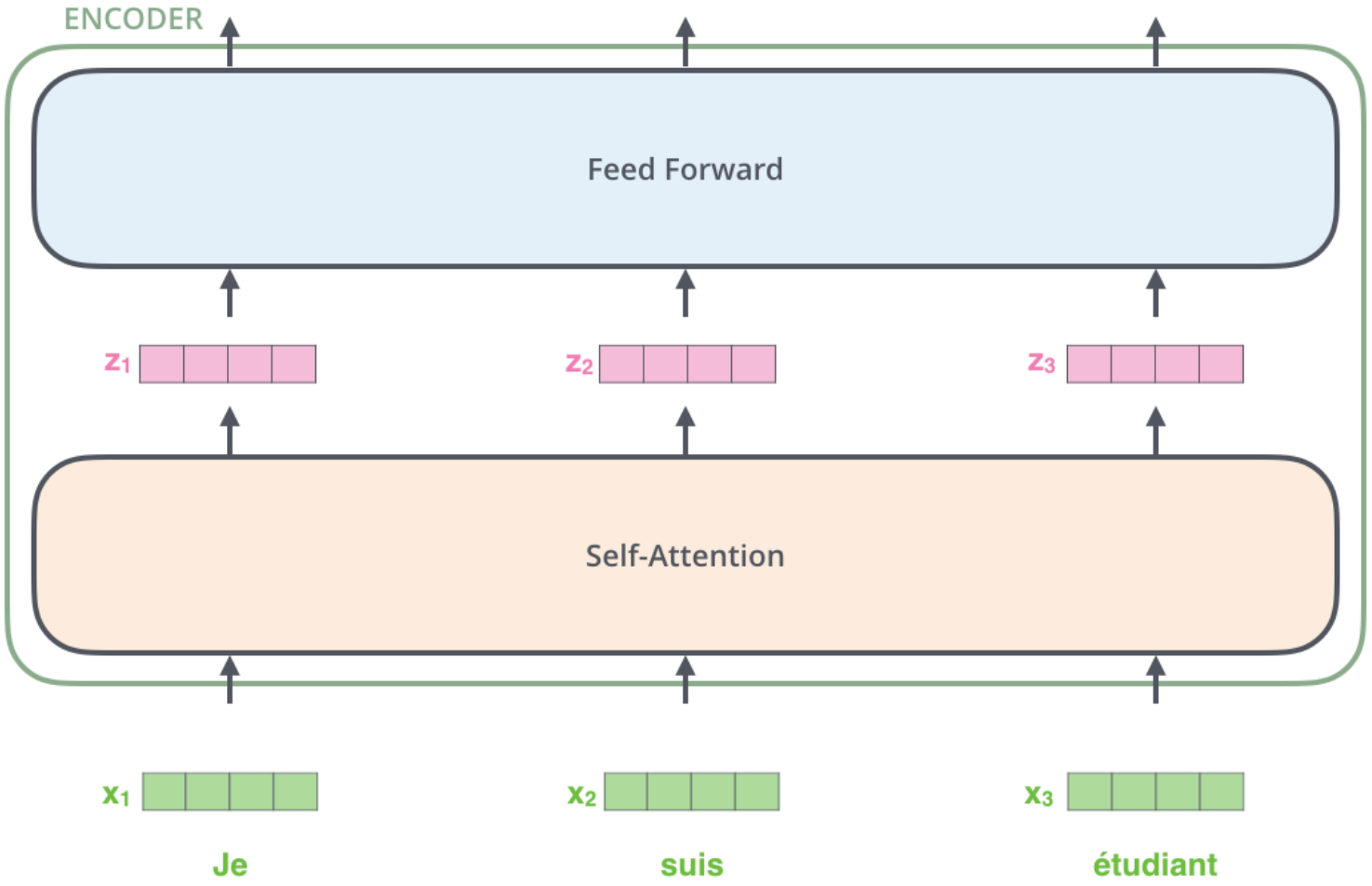
Architecture (encoder)



Transformer

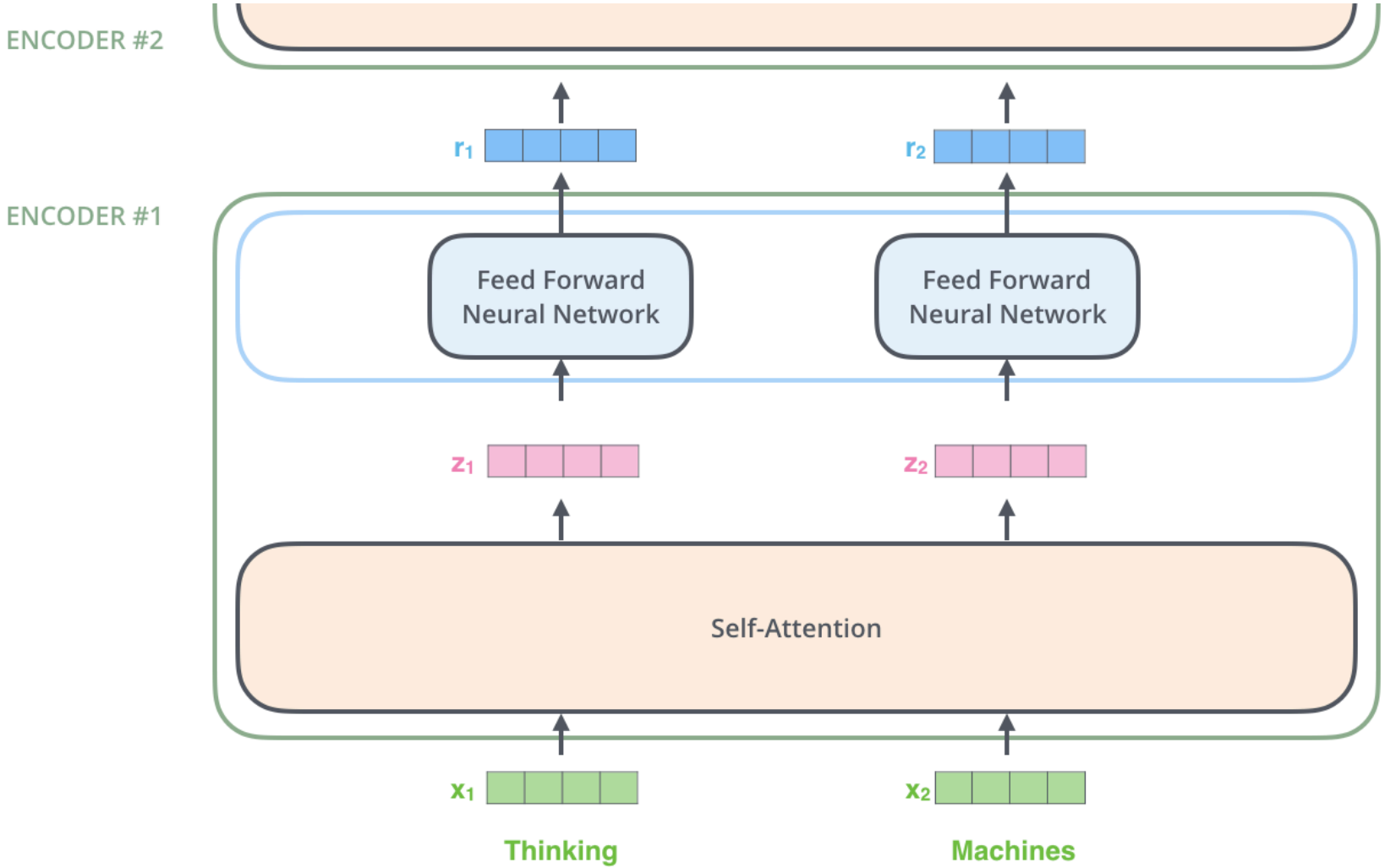
Architecture (encoder block)





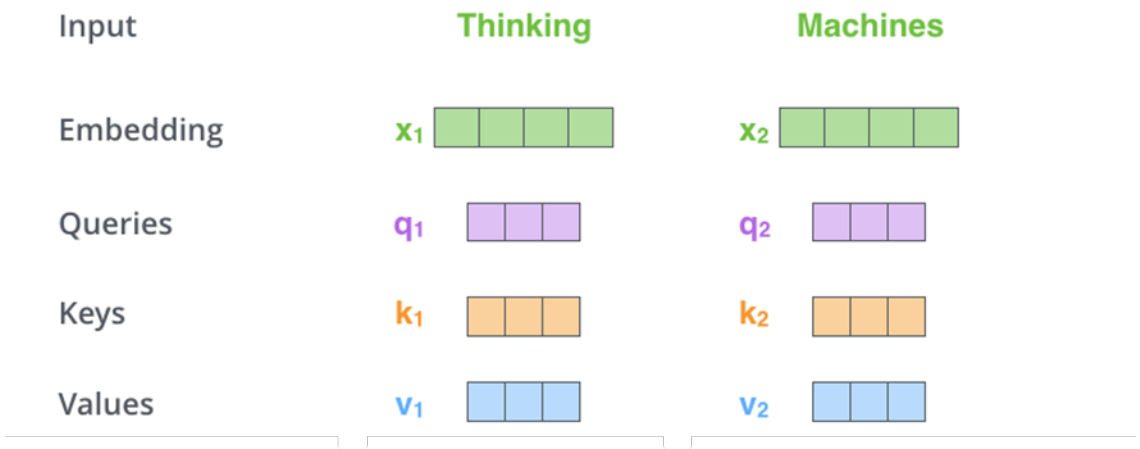
Transformer

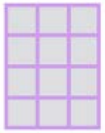
Architecture (encoder block)



Self-Attention

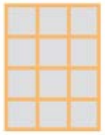
Step 1. Create 3 vectors (query, key, value) from each of the encoder's input vectors.





W^Q

$$q_i = W^Q x_i$$



W^K

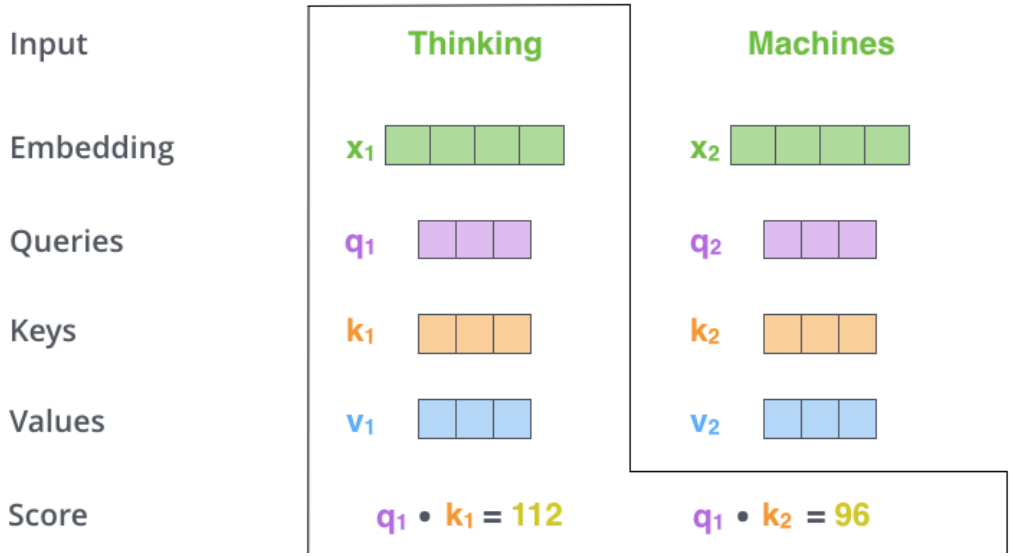
$$k_i = W^K x_i$$

W^V

$$v_i = W^V x_i$$

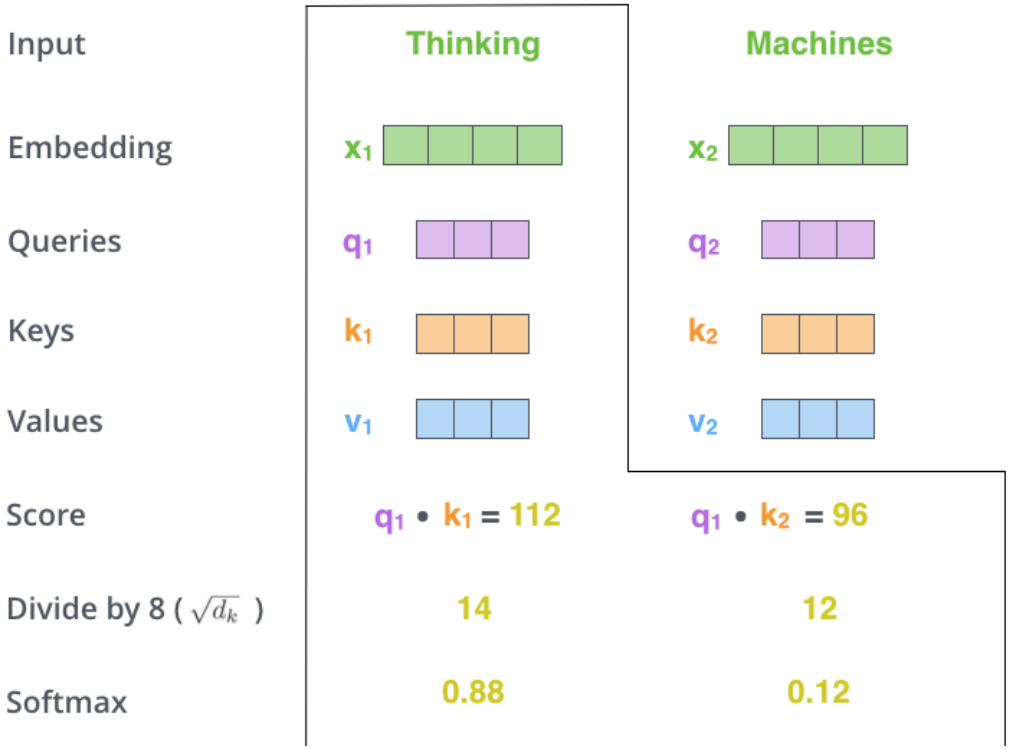
Self-Attention

Step 2. Calculate attention scores, for each word against all words in the sequence.



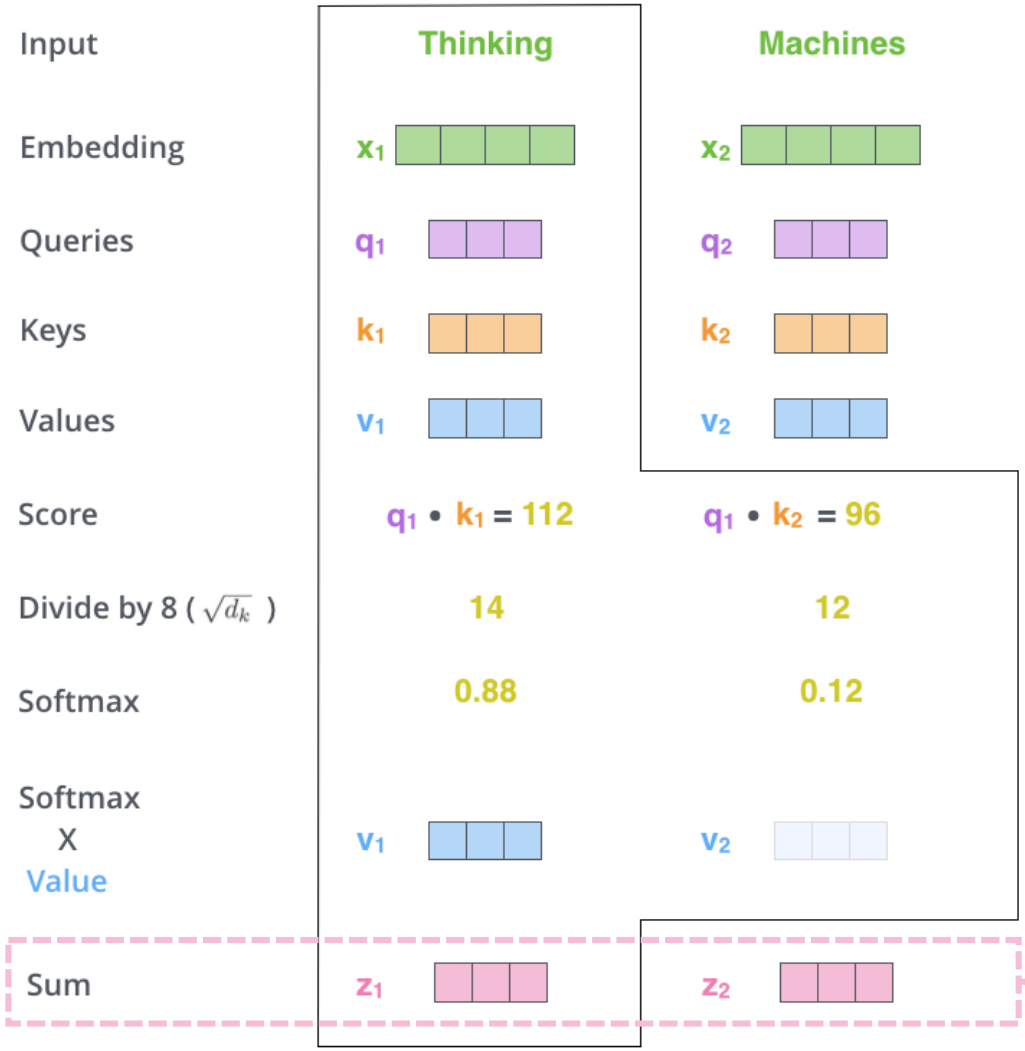
Self-Attention

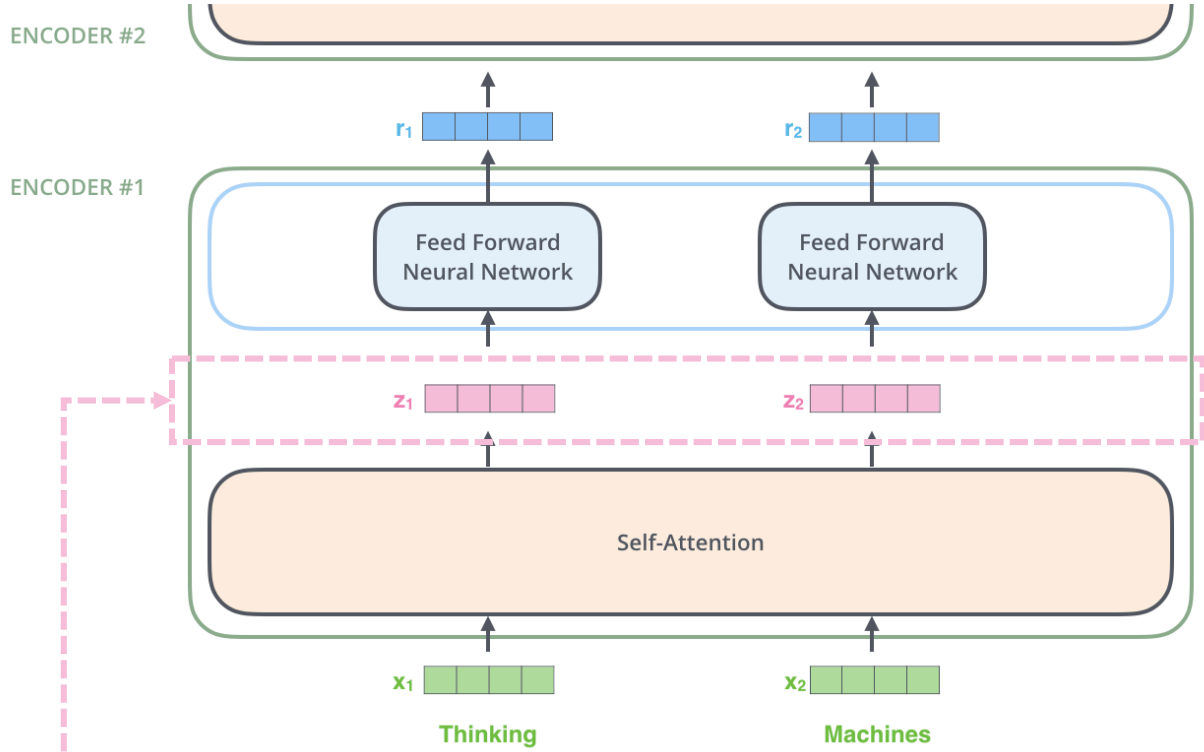
Step 3-4. Normalize the scores (divide with square root of dimension & apply softmax)



Self-Attention

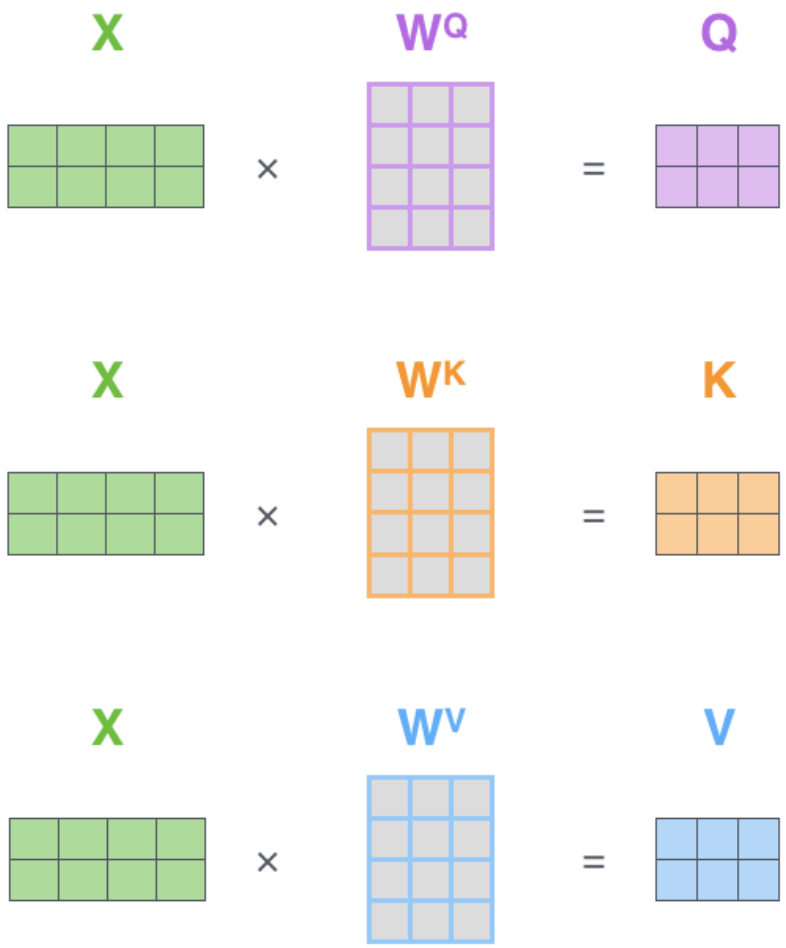
Step 5-6. Multiply each value vector with its score & sum up value vectors (only for multi-head)

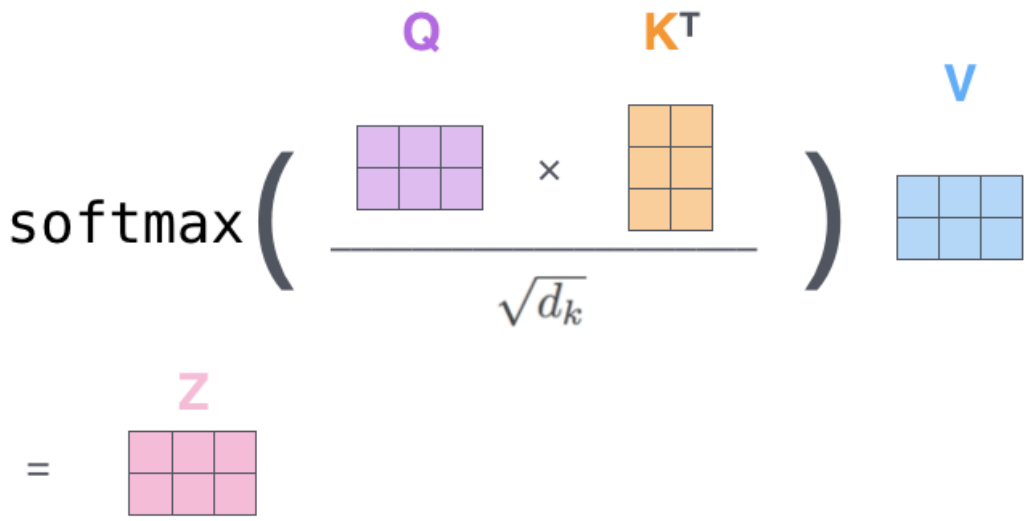




Self-Attention in Matrix Form

GPUs allow efficient parallelization of matrix multiplications





$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

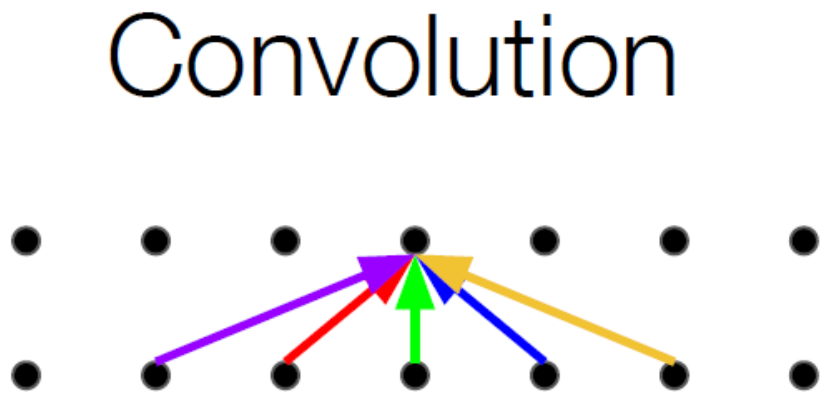
What’s missing from Self-Attention?

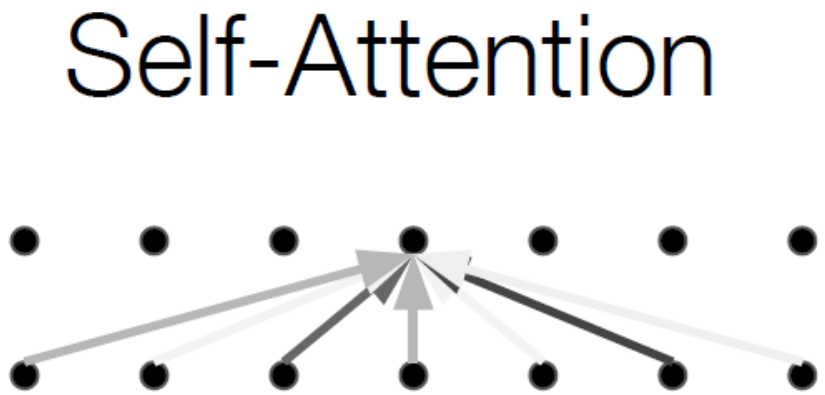
Comparisons

- Convolution: a different linear transformation for each relative position.

  → Allows you to distinguish what information came from where.

- Self-Attention: a weighted average ☹

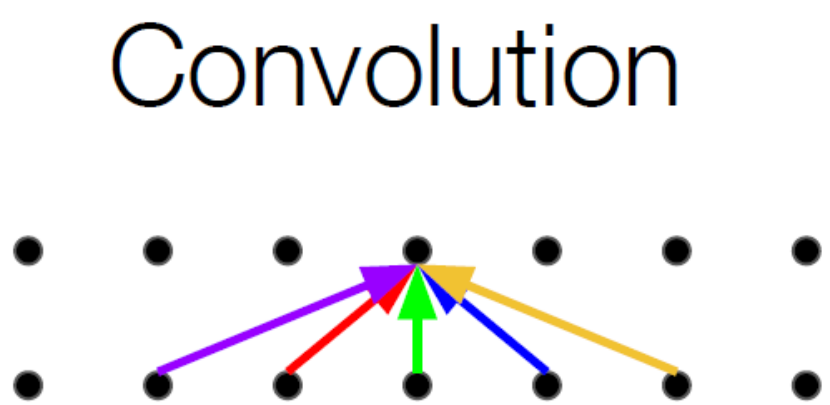


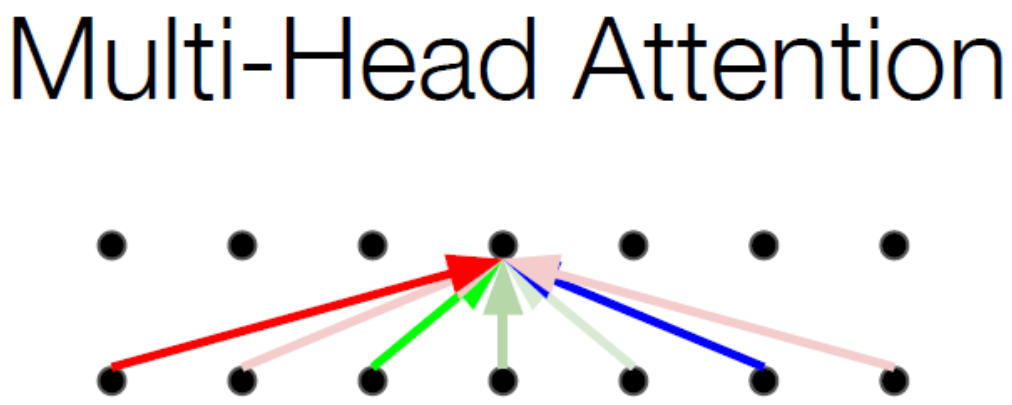


The Fix: Multi-Head Attention

More heads, more filters.

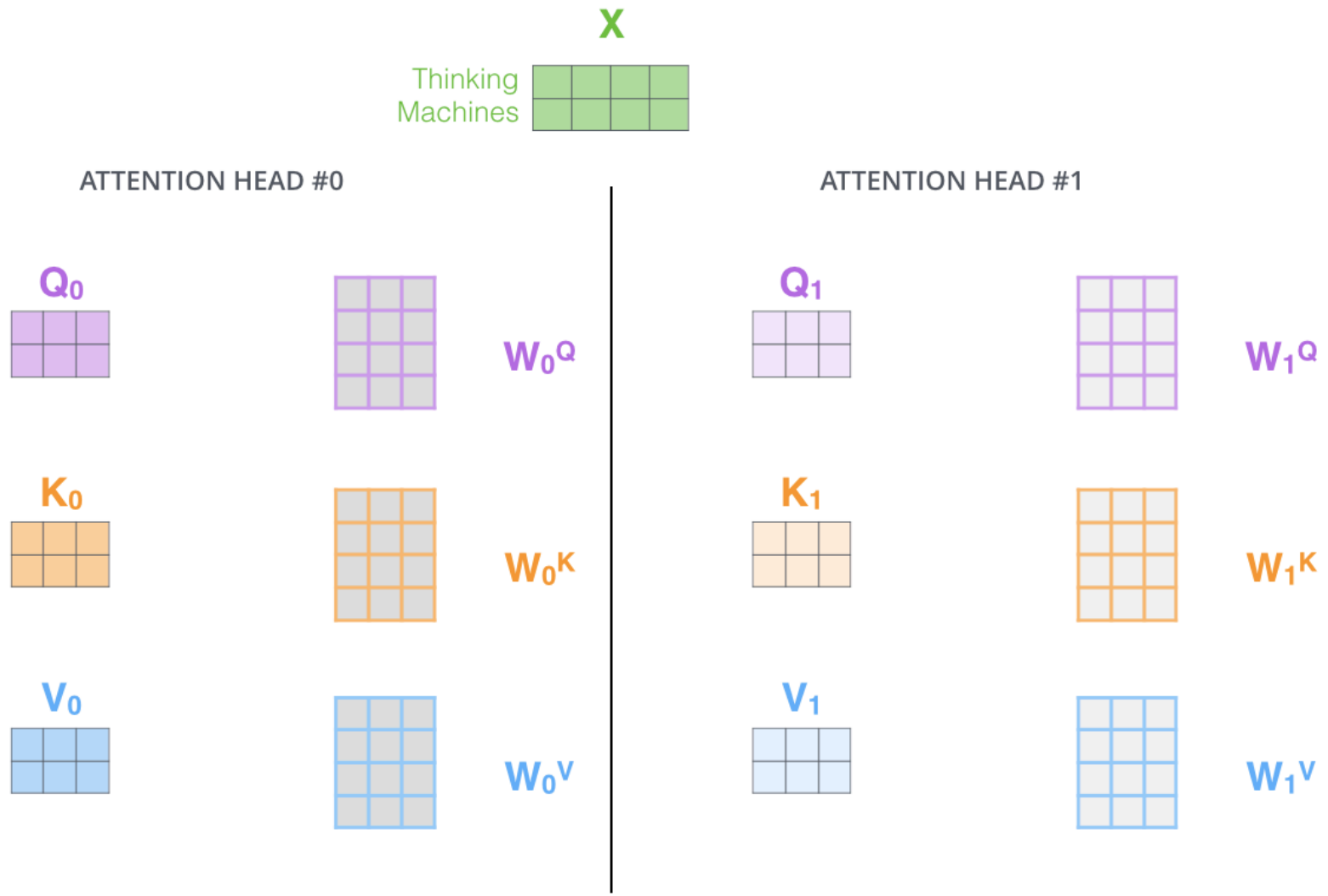
- Multiple attention layers (heads) in parallel (shown in different colors)
- Each head uses different linear transforms (Q, K, V)
- Different heads can learn different relationships





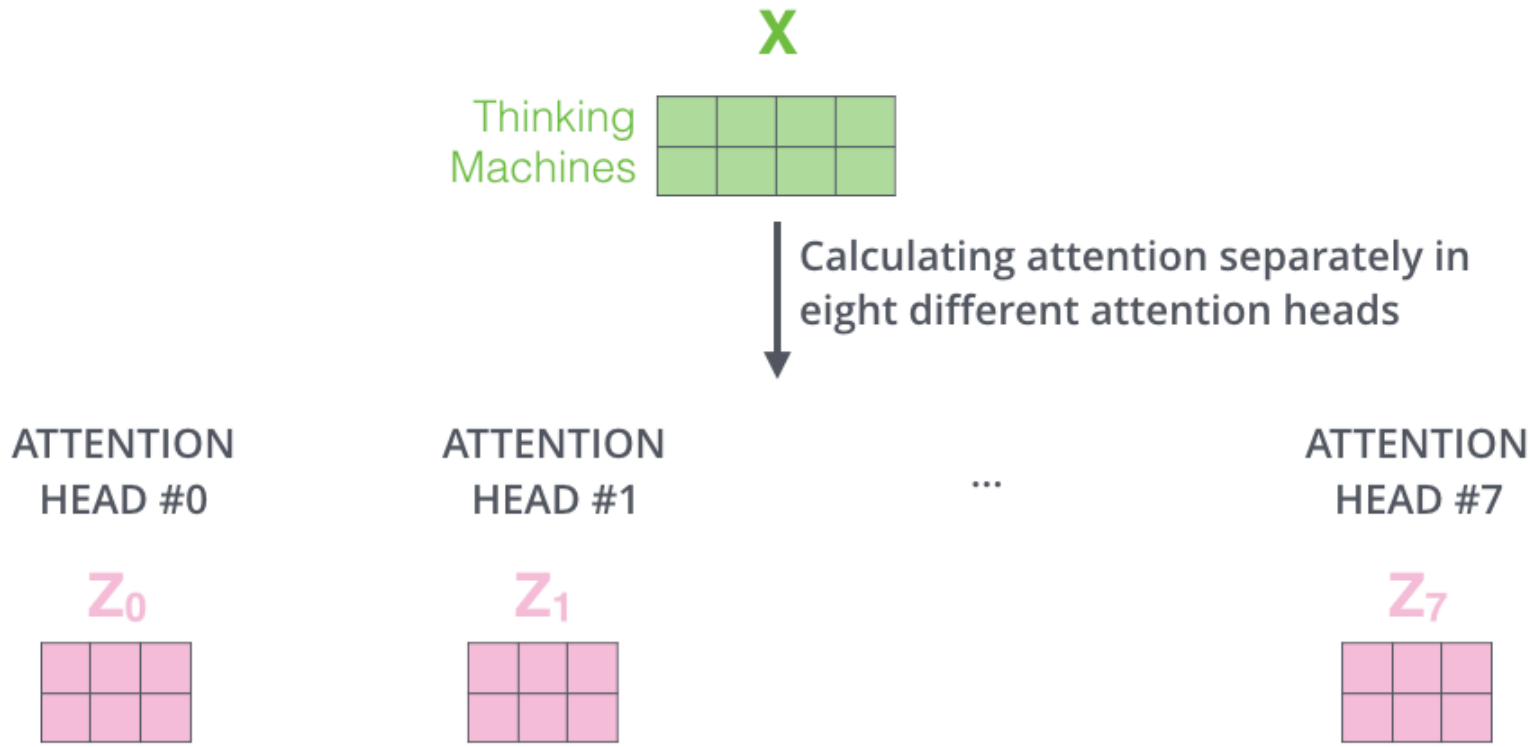
Multi-Head Attention

Maintain multiple sets of Q, K, V weight matrices



Multi-Head Attention

Number of heads = 8 (default from paper)



$$\text{MultiHead}(Q, K, V) = \text{Concatenate}(\text{head}_1, ..., \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

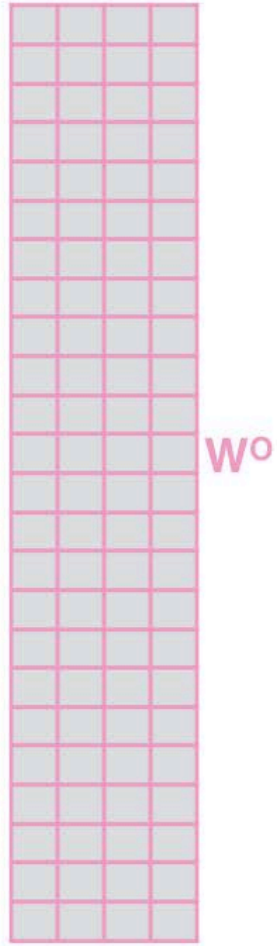
Multi-Head Attention

Concatenate & perform another linear transform

1) Concatenate all the attention heads

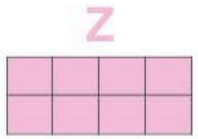
2) Multiply with a weight matrix W^O that was trained jointly with the model

X



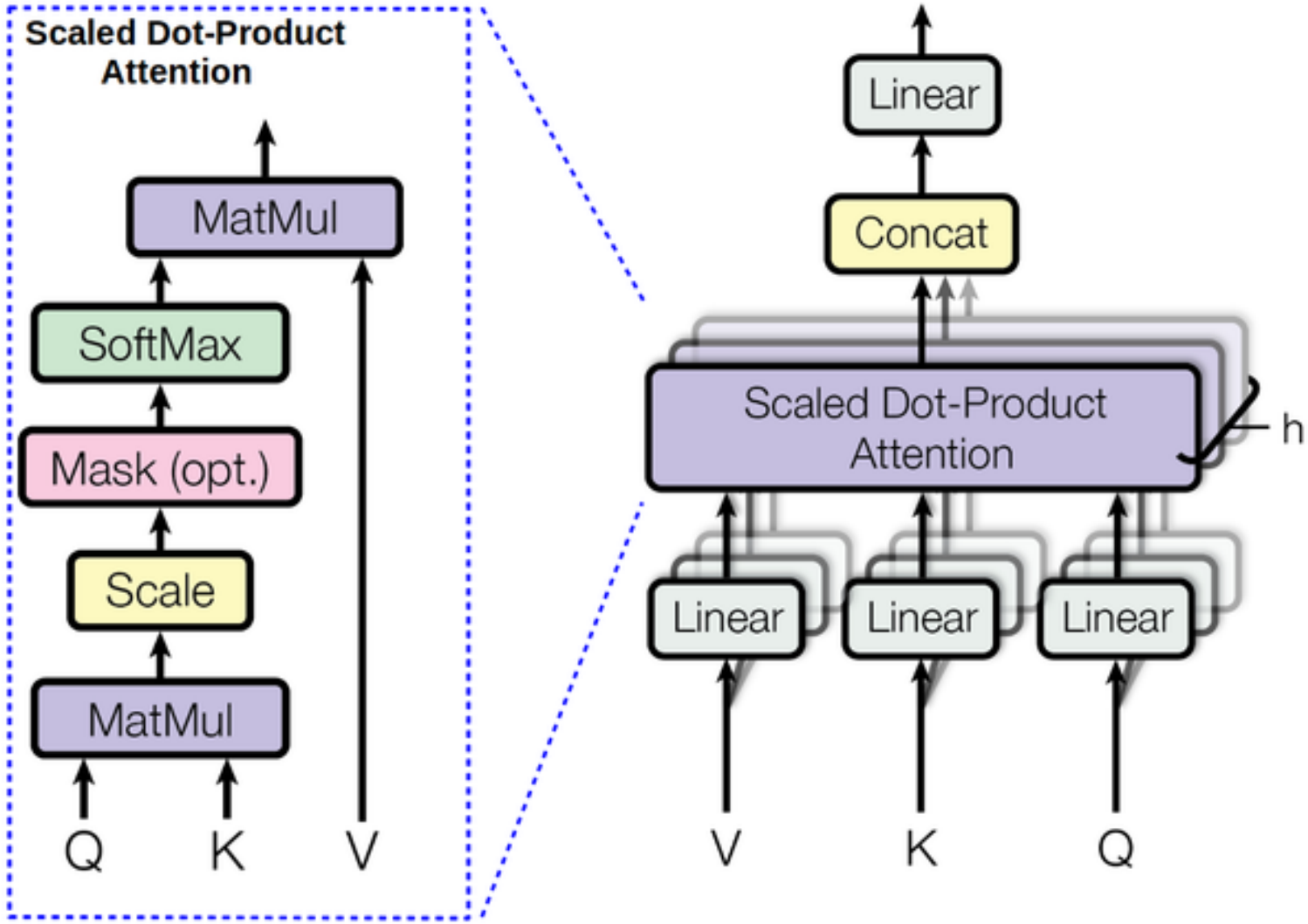
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



Multi-Head Attention

Figure from paper



1) This is our input sentence*

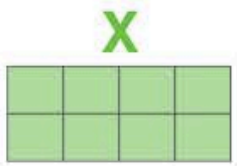
2) We embed each word*

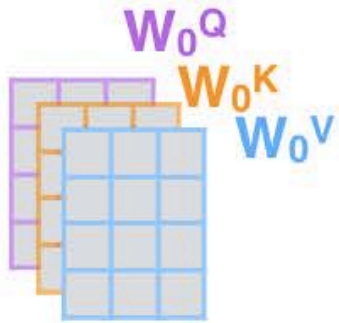
3) Split into 8 heads. We multiply X or R with weight matrices

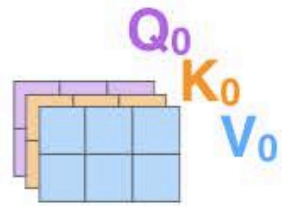
4) Calculate attention using the resulting Q/K/V matrices

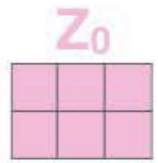
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

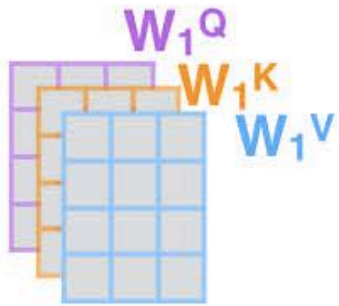
Thinking Machines

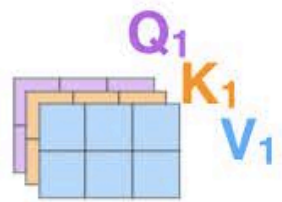


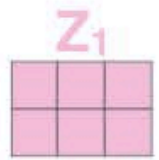


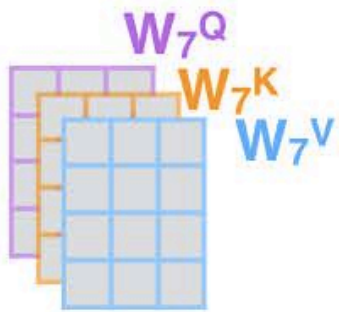


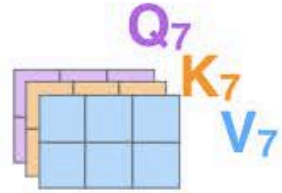




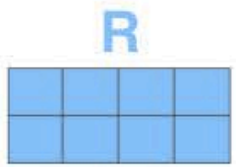








* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

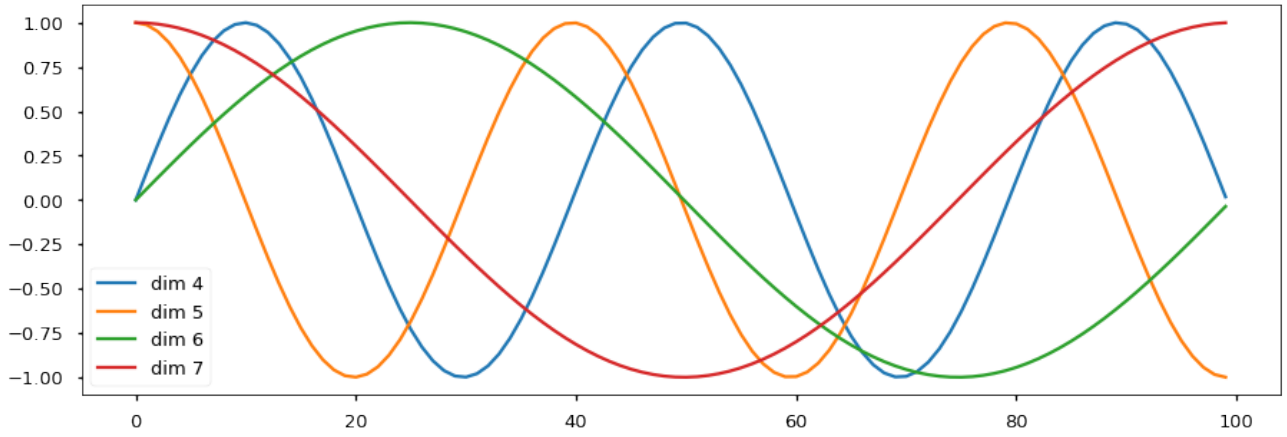


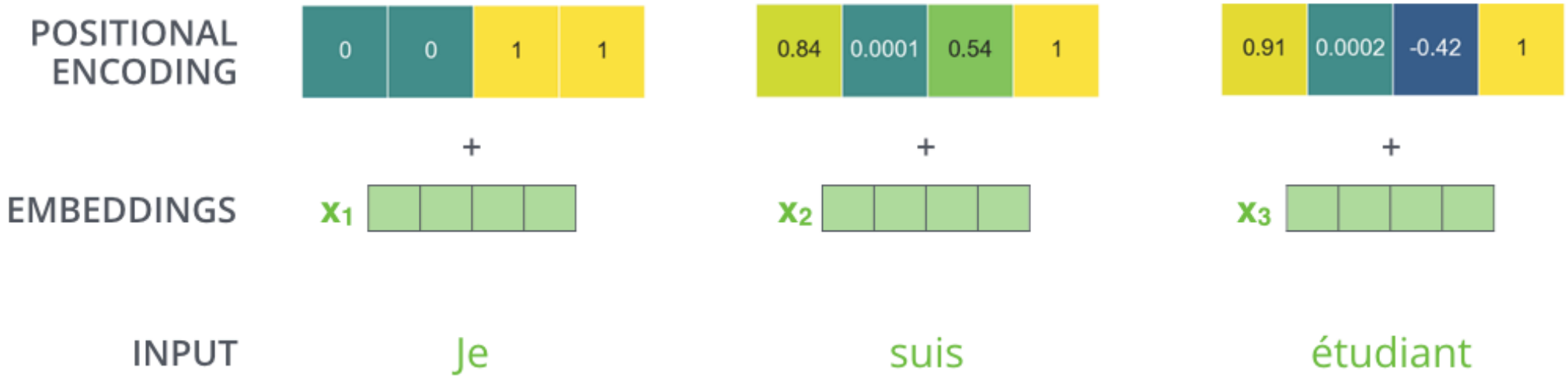
Positional Encoding

Representing the order of the sequence

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d_{model}}})$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}})$$

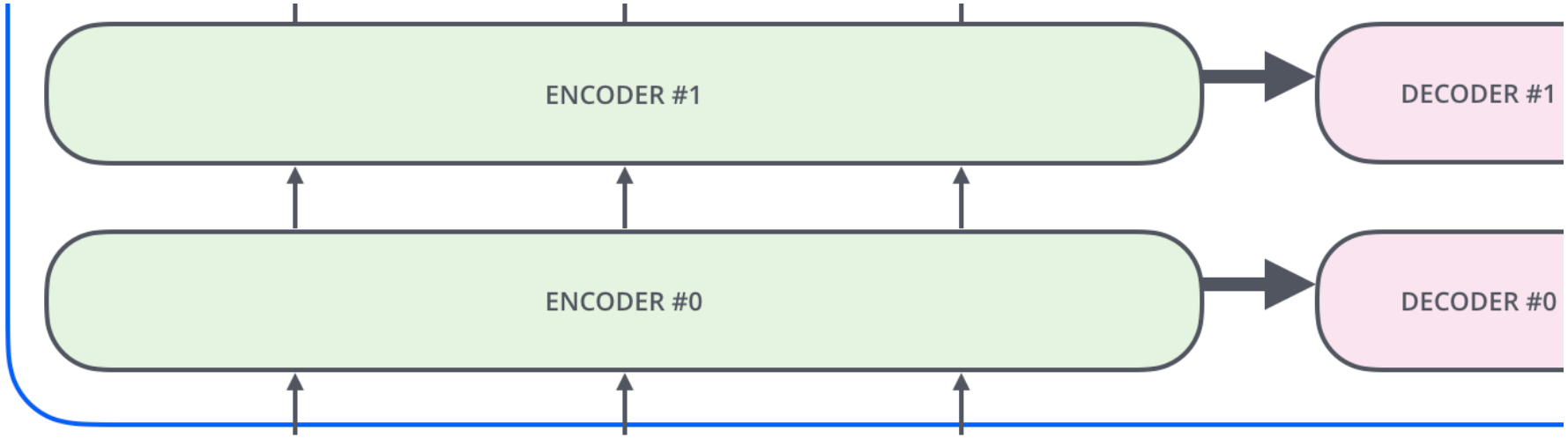


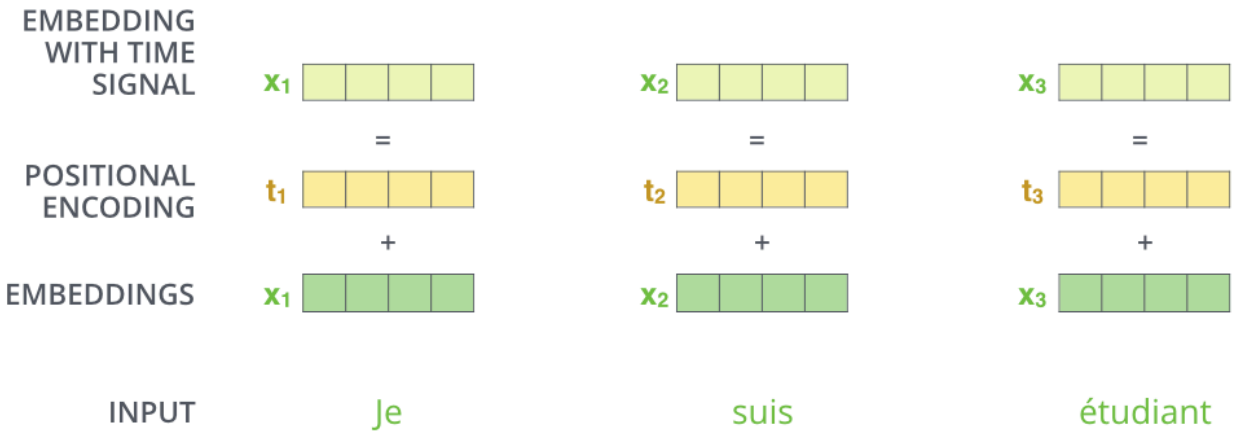


Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

Positional Encoding

Word embedding vector + positional encoding



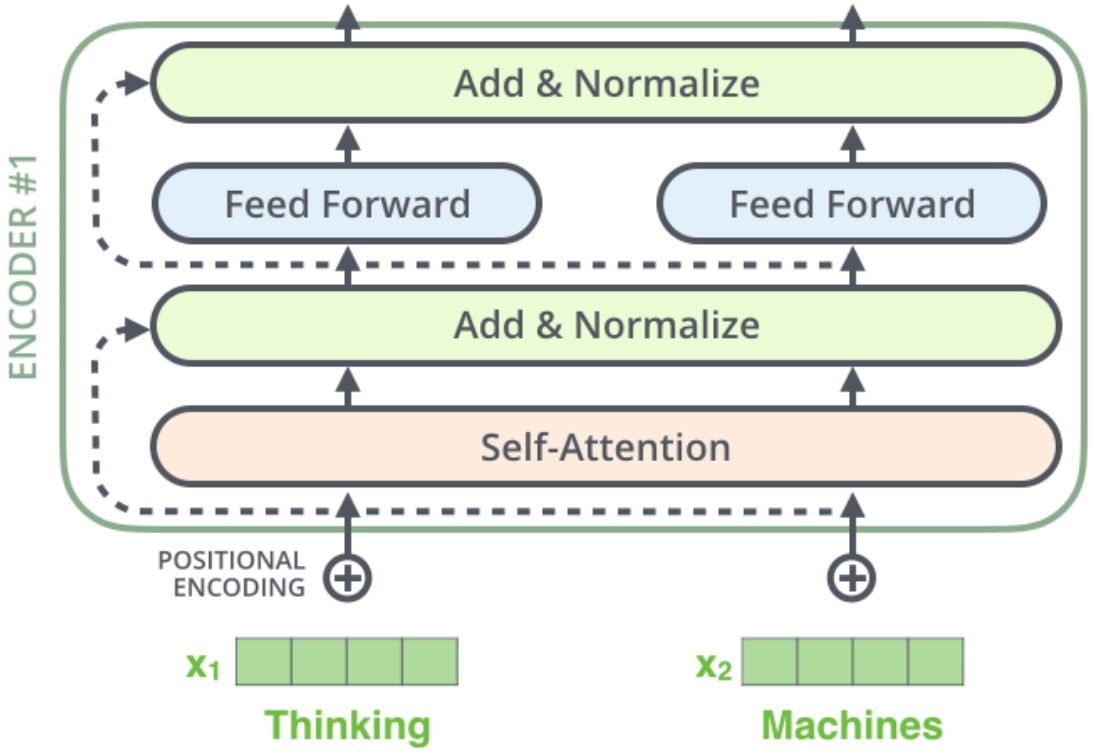


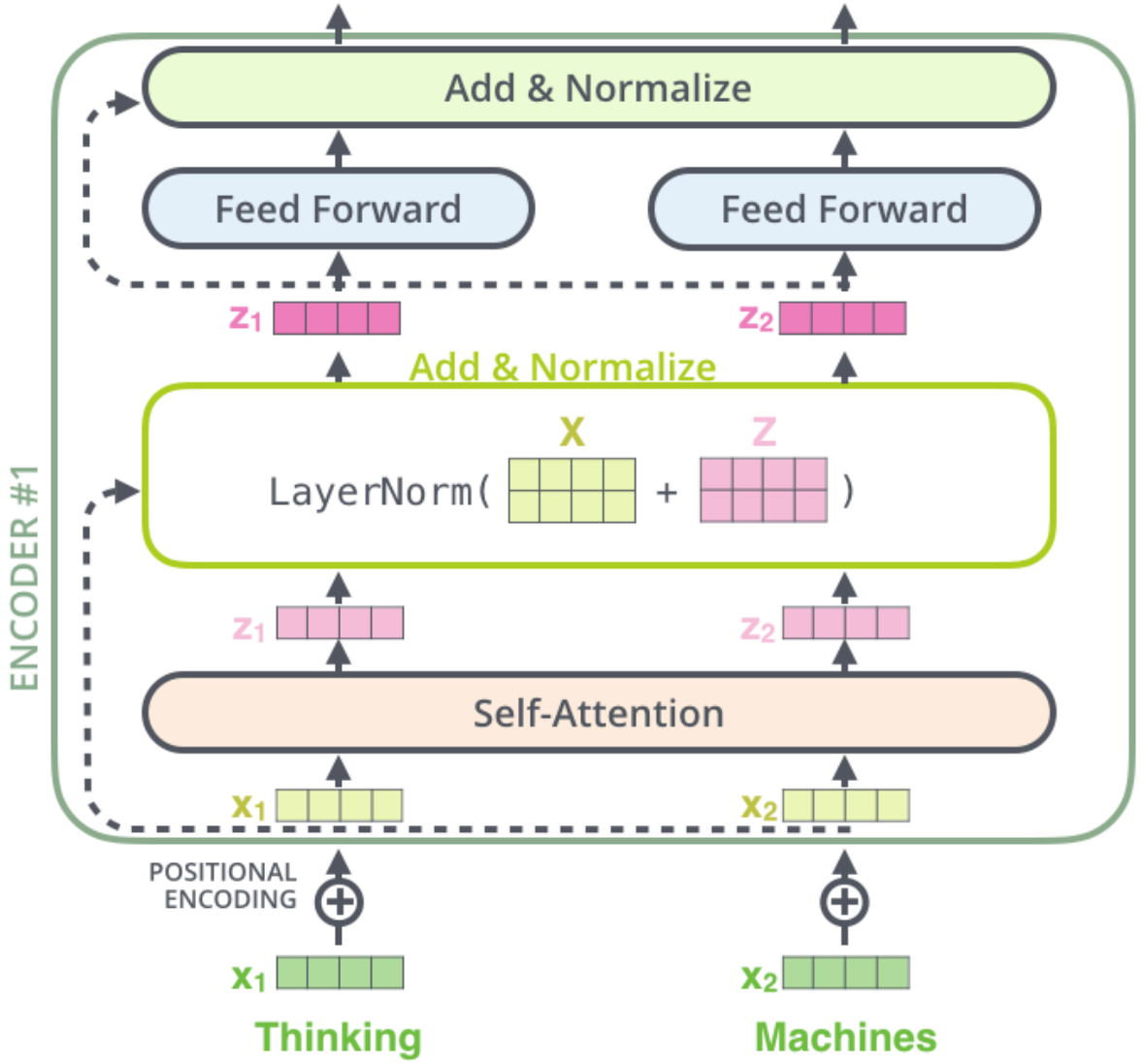
$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d_{model}}})$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}})$$

Encoder Wrap-Up

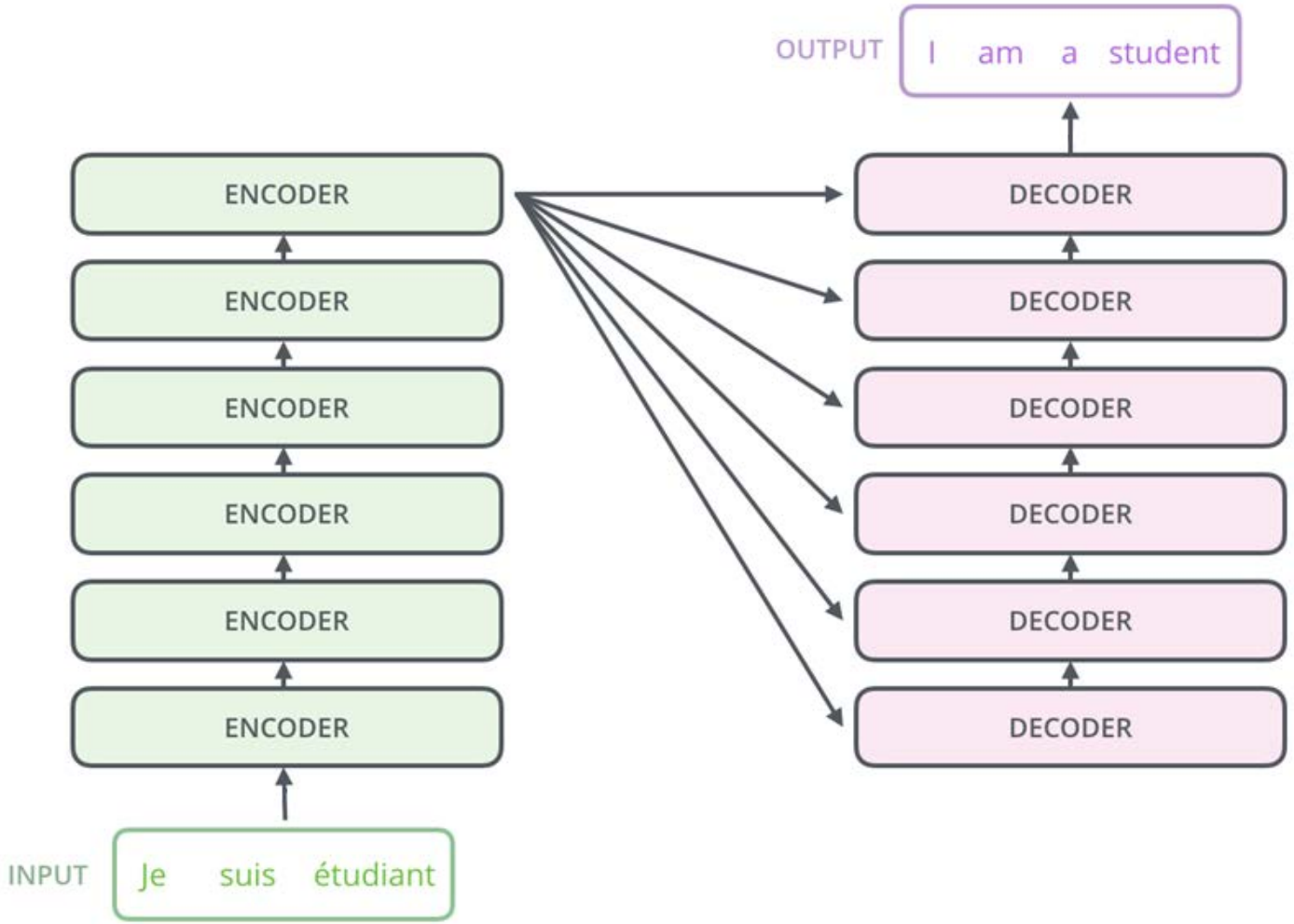
Architecture (overview)





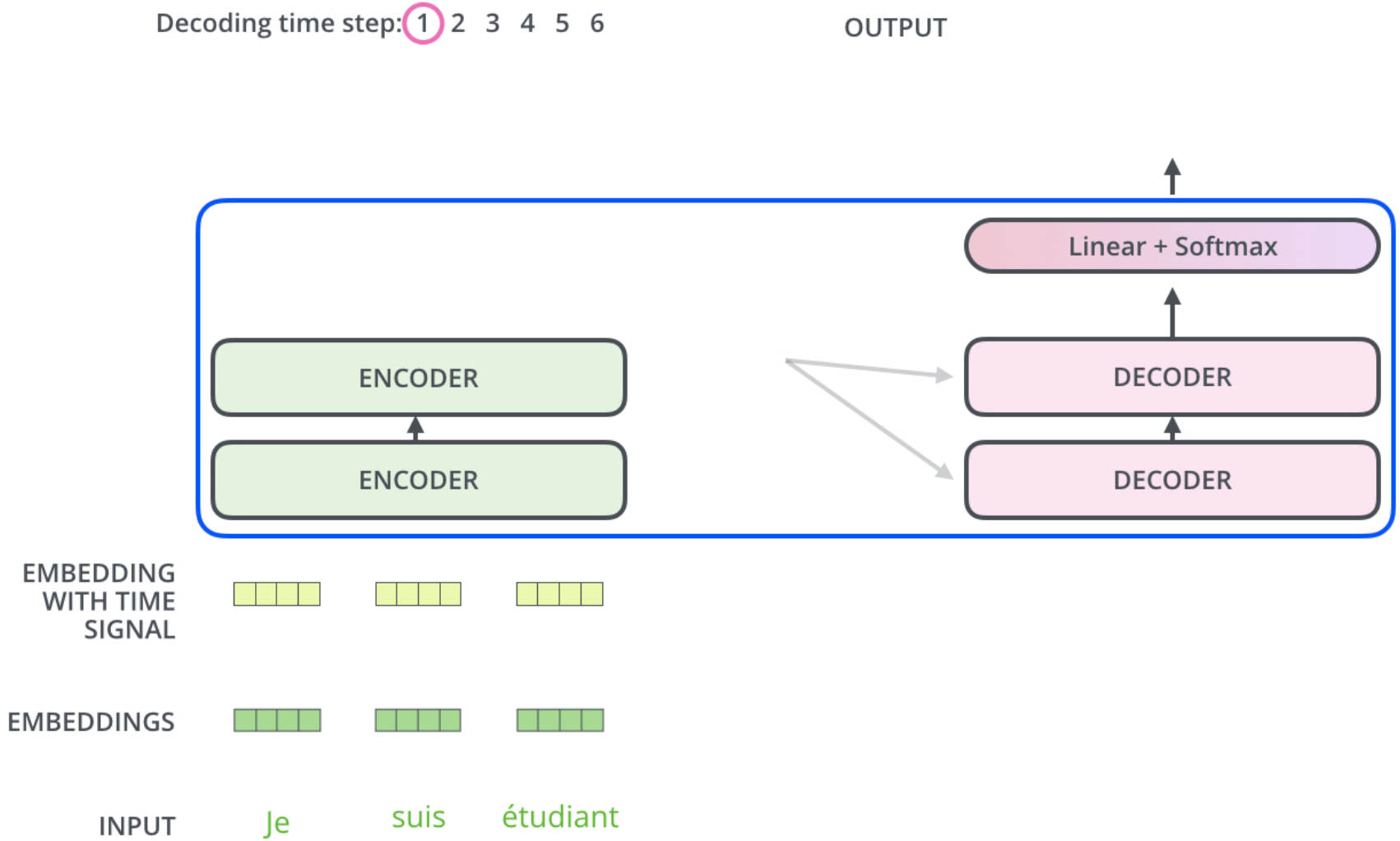
Transformer

Architecture (overview)



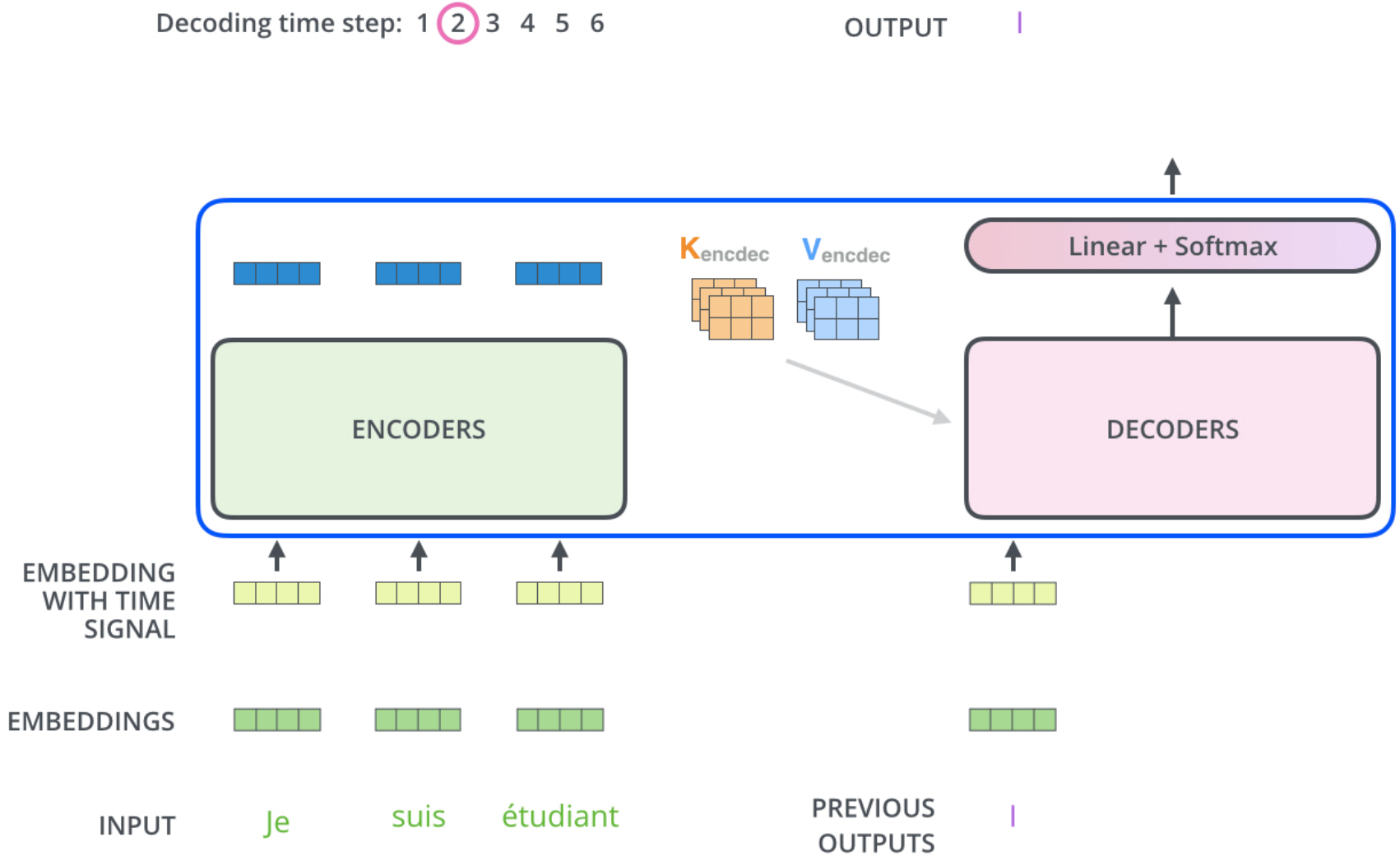
Encoder-Decoder Attention

Use the output of the encoder as keys and values matrices (K, V)



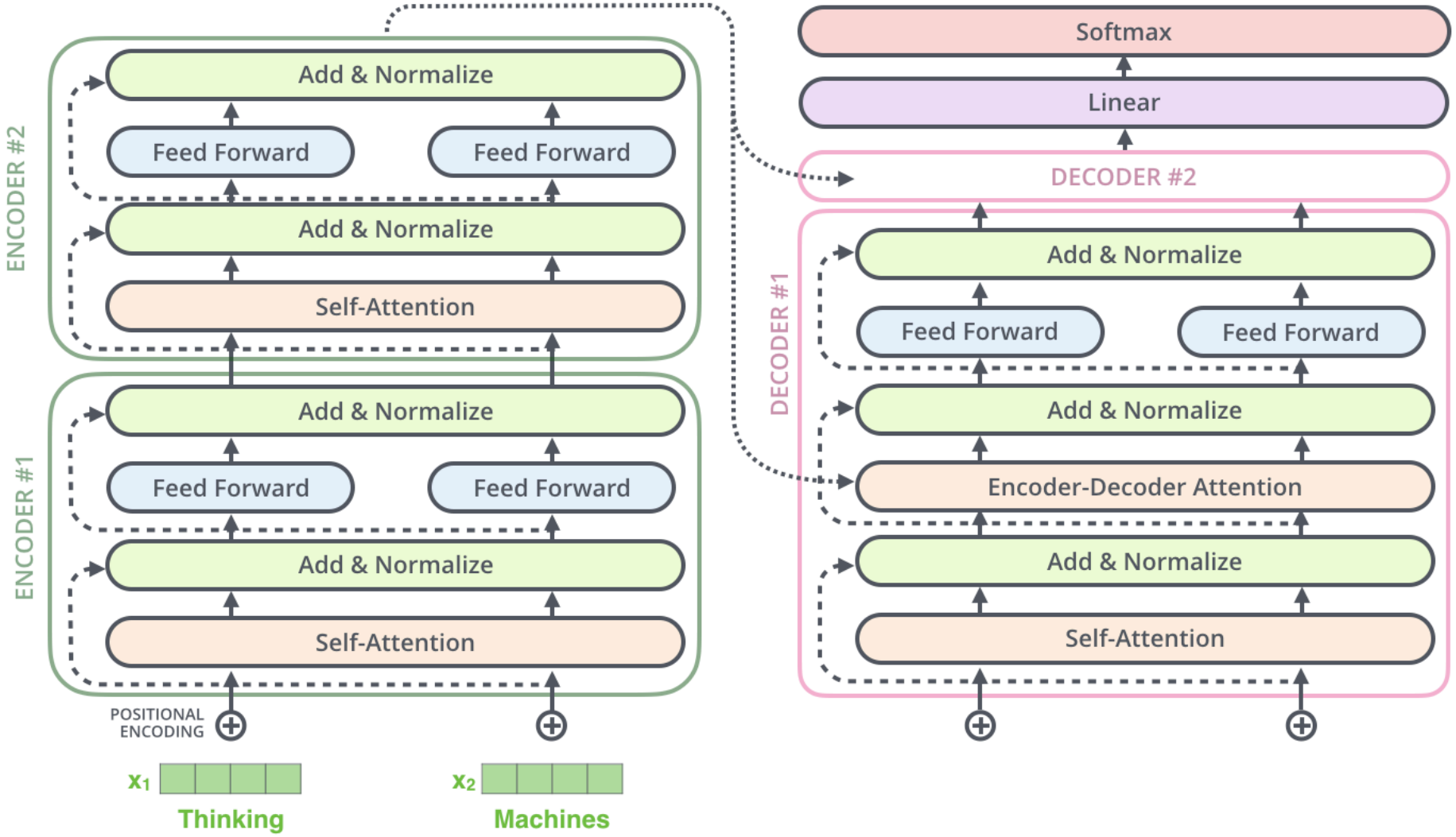
Encoder-Decoder Attention

Use as query (Q), the query computed from decoder self-attention of the layer below it.



Transformer

Architecture (encoder + decoder)



Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

Tricks for Training

Refer to paper

- Masking in decoder self-attention
- Learning rate scheduling
- Label Smoothing
- Beam Search
- …

Time Complexity of Self-Attention

Comparisons

In most cases, $n << d$

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

BLEU Scores & Training Costs

Higher the better

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Model Variations

Parameter tuning

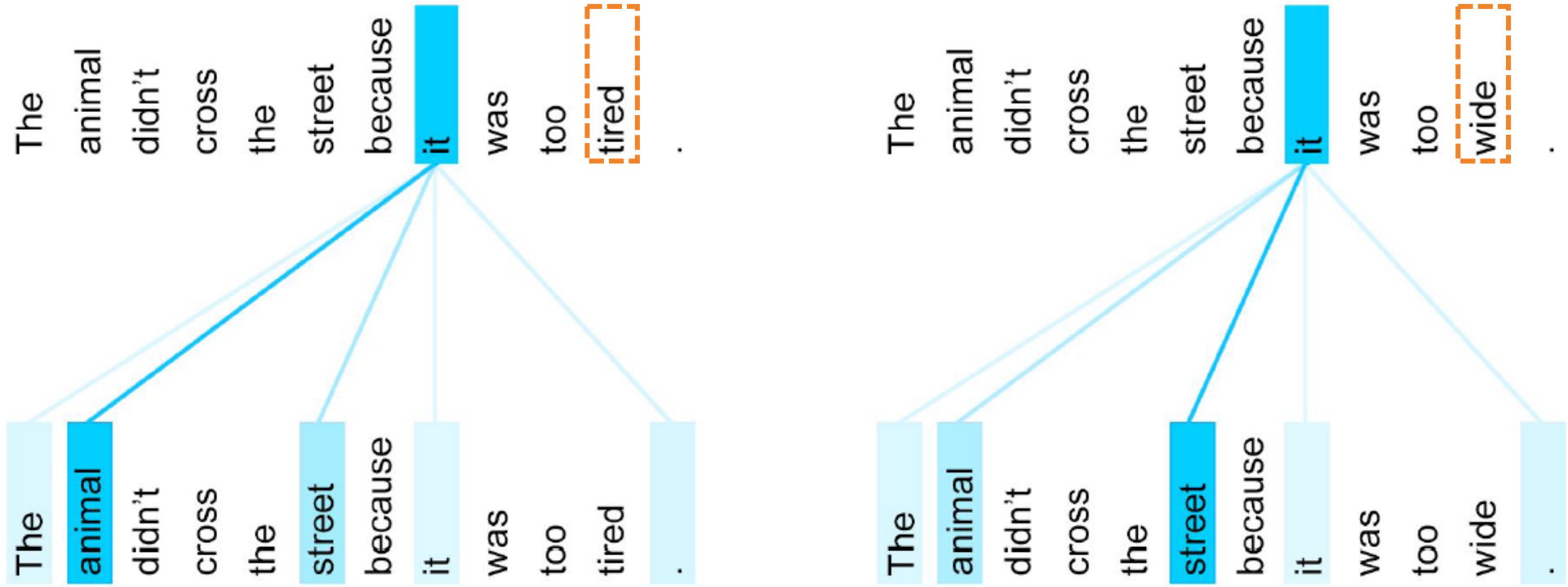
Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Coreference Resolution

Example

The animal didn’t cross the street because it was too _____.

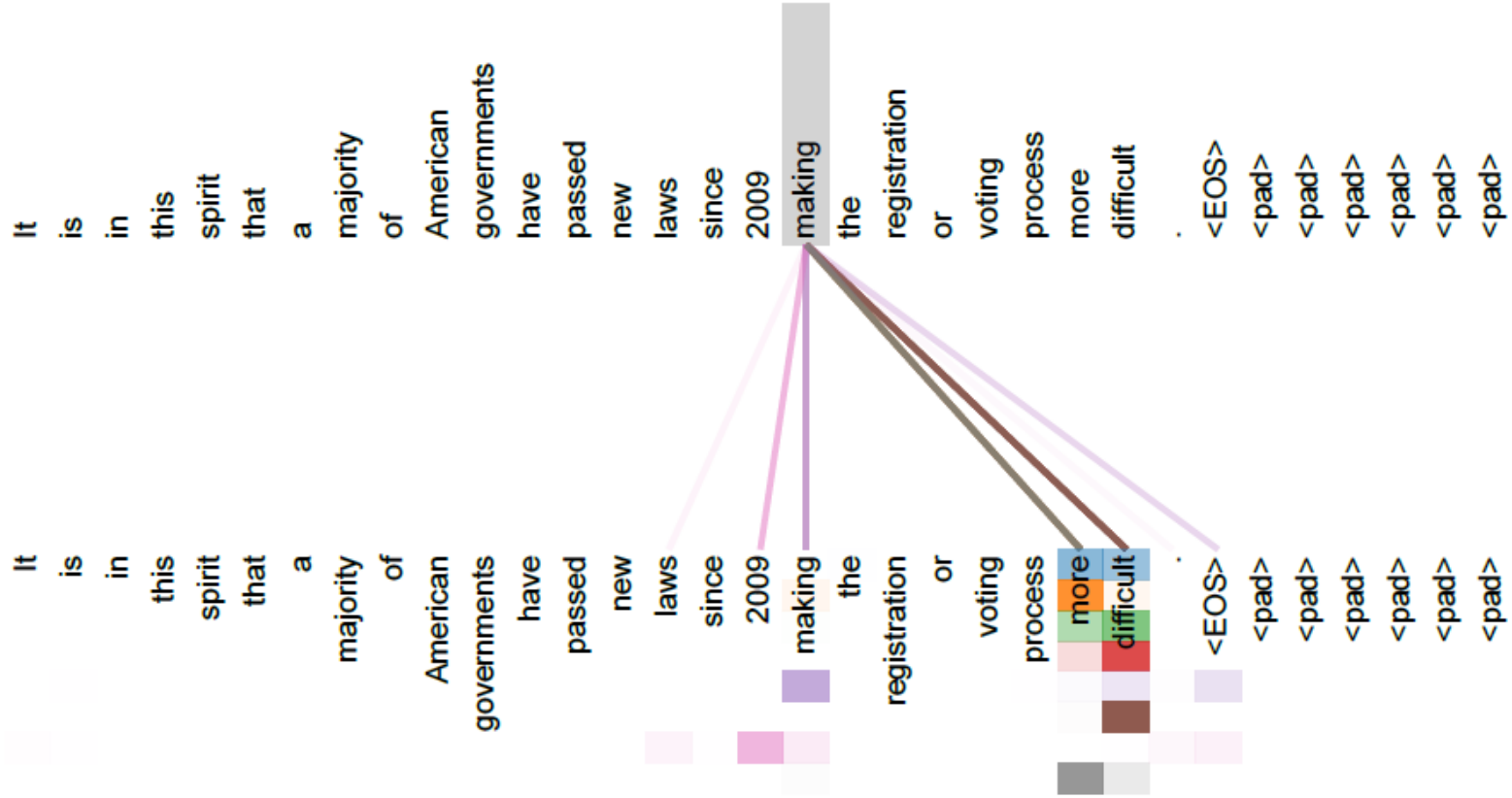


https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Attention Visualizations

Capturing long-range dependencies

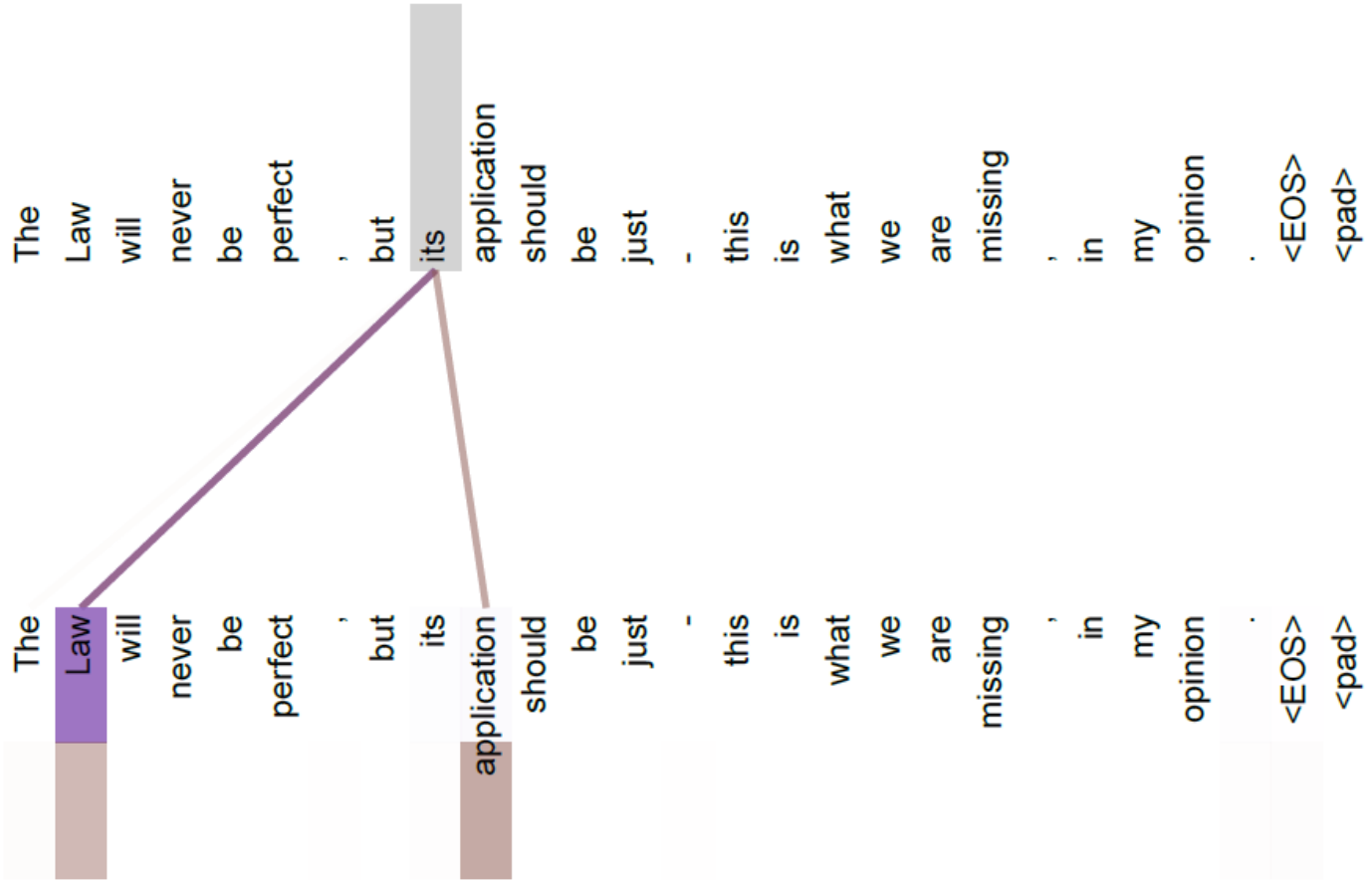
It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult.



Attention Visualizations

Anaphora resolution (the problem of resolving what a pronoun, or a noun refers to)

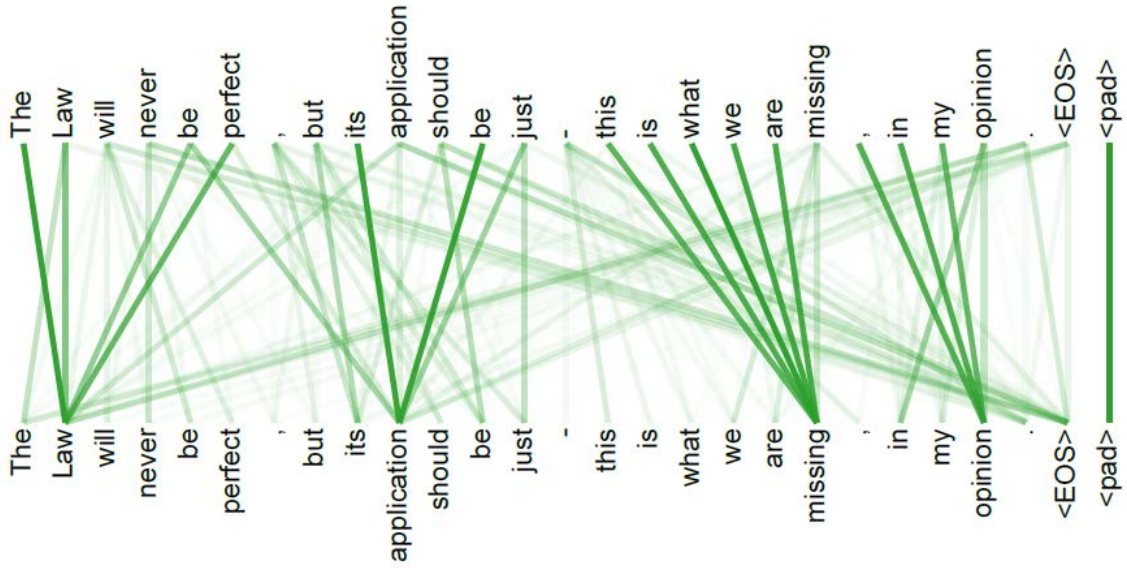
The law will never be perfect, but its application should be just – this is what we are missing, in my opinion.

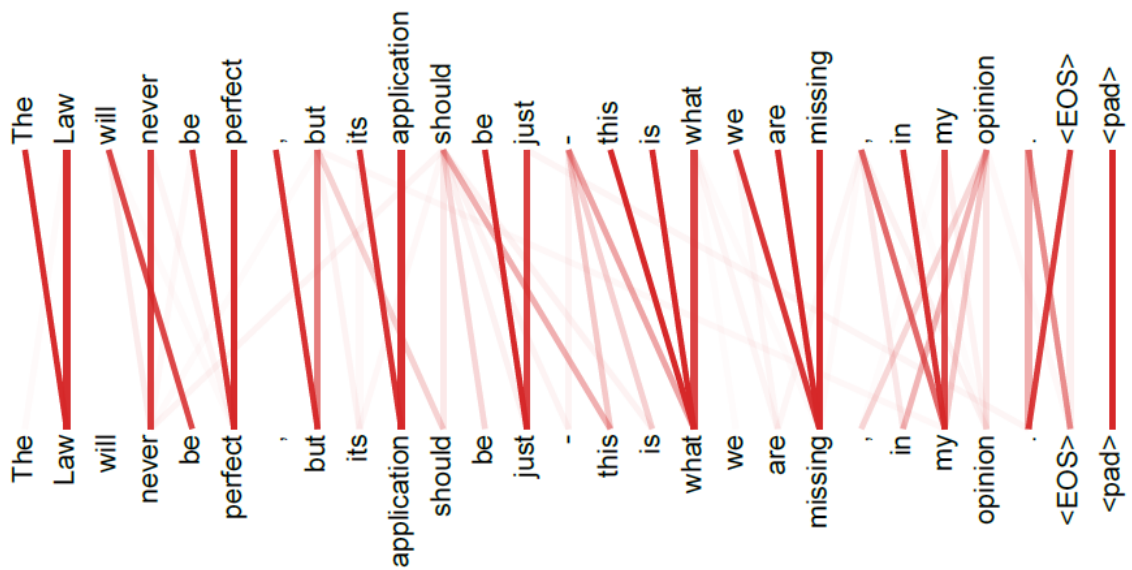


Attention Visualizations

Different attention heads learn different structural dependencies of the sentence

The law will never be perfect, but its application should be just – this is what we are missing, in my opinion.





Python Implementations

In both Tensorflow and Pytorch

- (Tensorflow) Tensor2Tensor library (github, jupyter notebook)
- (Pytorch) The Annotated Transformer (github, blog post)

BERT

Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018

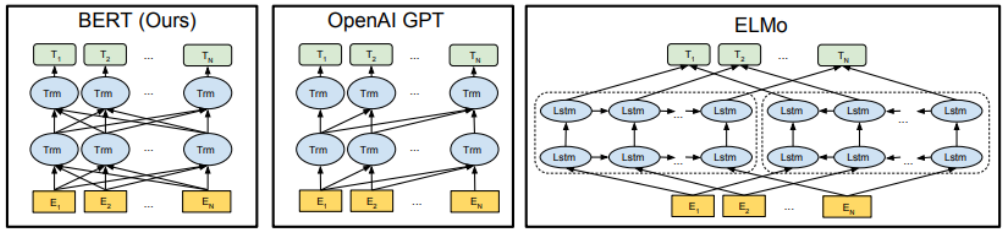


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

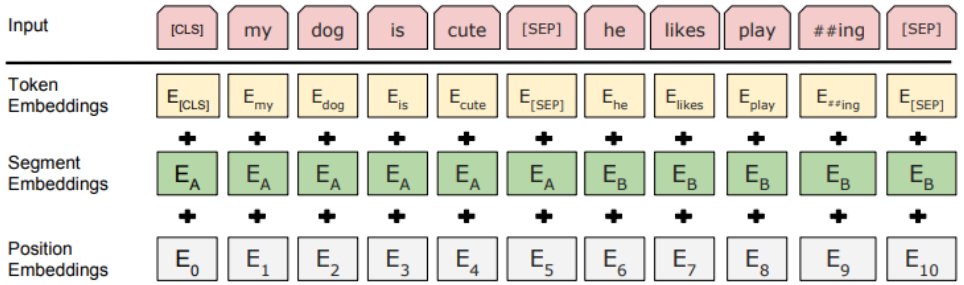


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

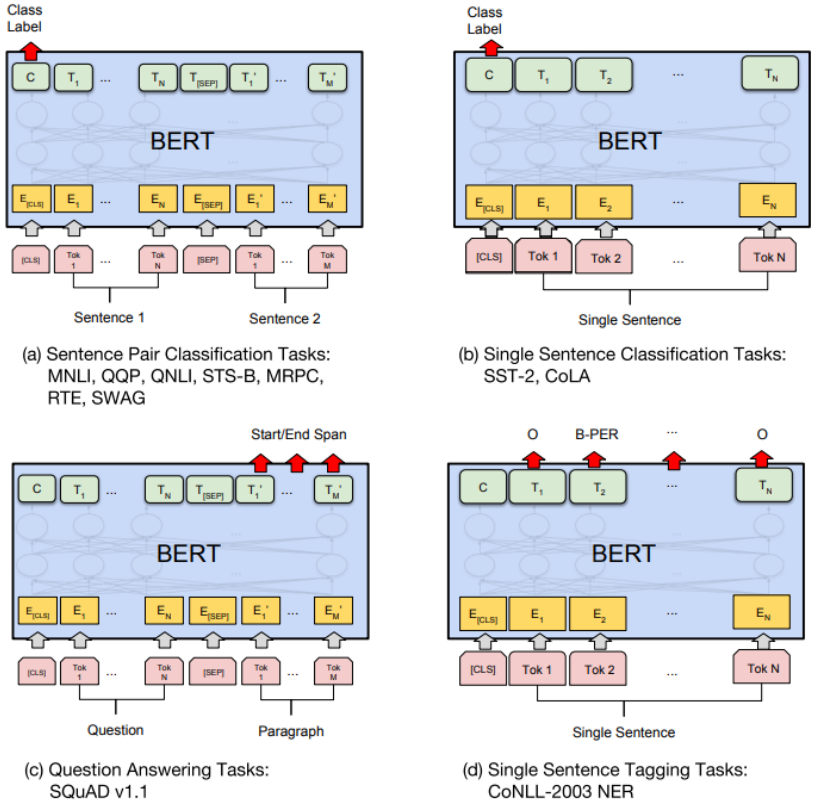


Figure 3: Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, [CLS] is the special symbol for classification output, and [SEP] is the special symbol to separate non-consecutive token sequences.

Summary

Key concepts

- RNN-based Seq2seq models suffer from the burden of sequential computation.
- The Transformer relies on an attention mechanism to draw global dependencies:
  - input ↔ input (encoder self-attention)
  - input ↔ output (encoder-decoder attention)
  - output ↔ output (decoder self-attention)

Reference

Papers, blog posts & videos

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008).
- Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.
- Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
- Britz, D., Goldie, A., Luong, M. T., & Le, Q. (2017). Massive exploration of neural machine translation architectures. arXiv preprint arXiv:1703.03906.
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.
- Press, O., & Wolf, L. (2016). Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859.
- https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- http://jalammar.github.io/illustrated-transformer/
- https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/#.W_aFTFxR1jU
- https://www.mihaileric.com/posts/transformers-attention-in-disguise/
- https://www.youtube.com/watch?v=rBCqOTEfxvg