
ATTENTION IS ALL YOU NEED

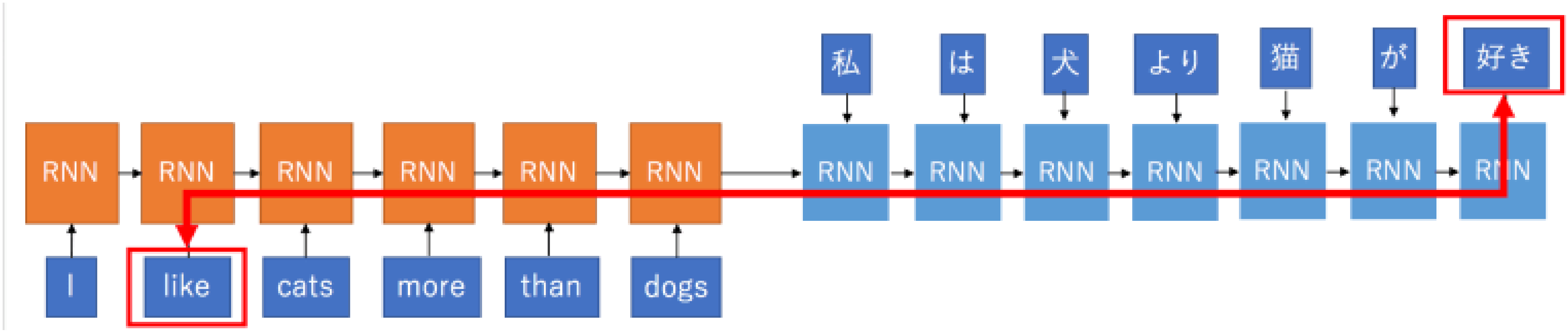
INTRODUCTION

Sequence Transduction

- Convert a *sequence* to another *sequence*

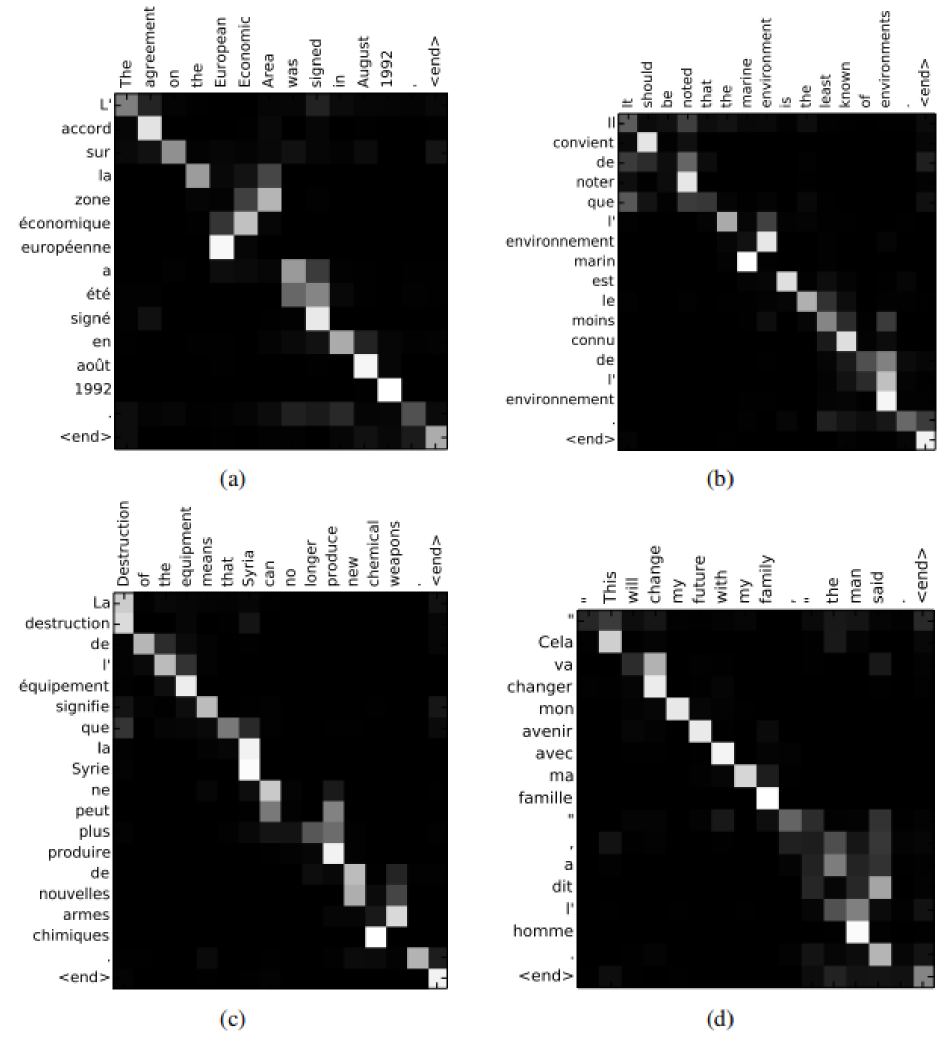
Past Works – RNNs

- Sequential nature disallows parallelization within training samples
- Long distance between input and output



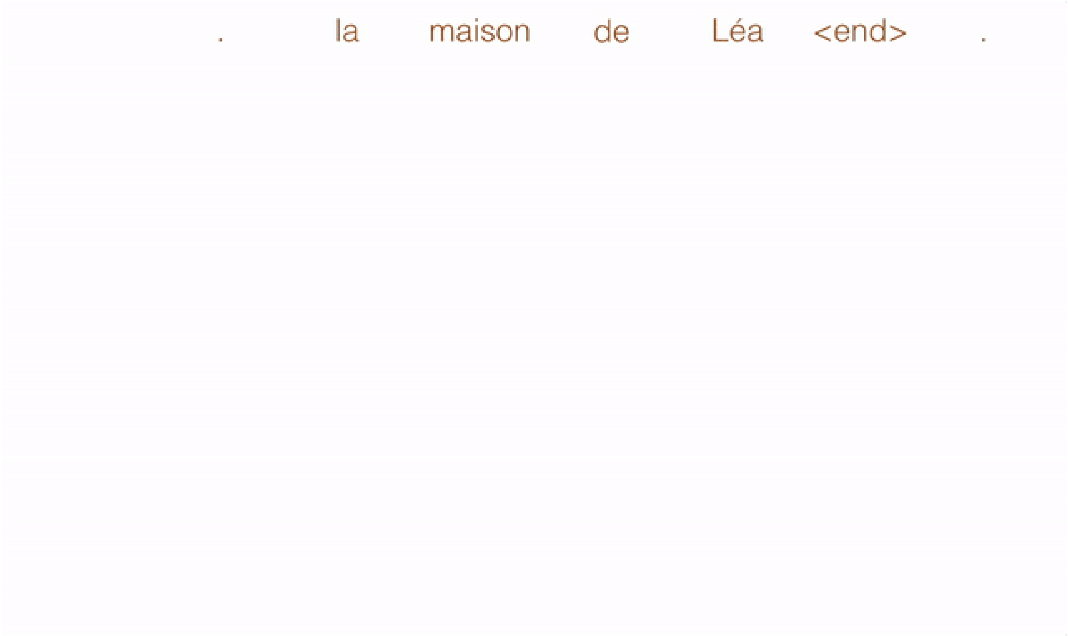
Past Works – Attention

- Allow modelling dependencies disregarding distance
- Used with RNNs or CNNs



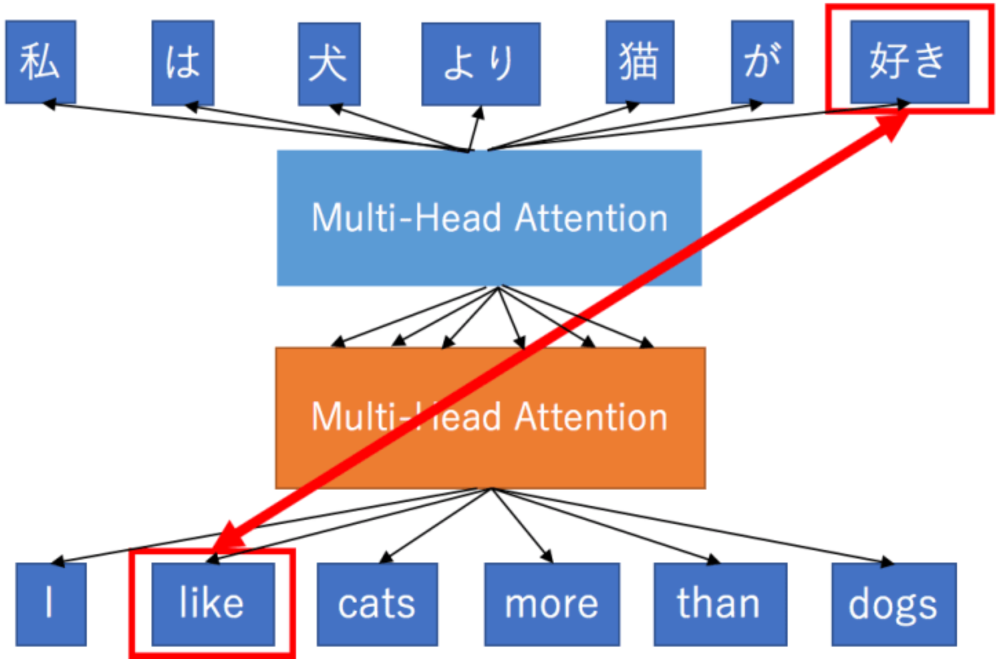
Past Works – CNN

- Allow parallelism
- Difficult to learn dependencies between distant positions



Our Work – Transformer

- Rely entirely on *self-attention* to compute representations
- Allows for greater parallelization and short path lengths

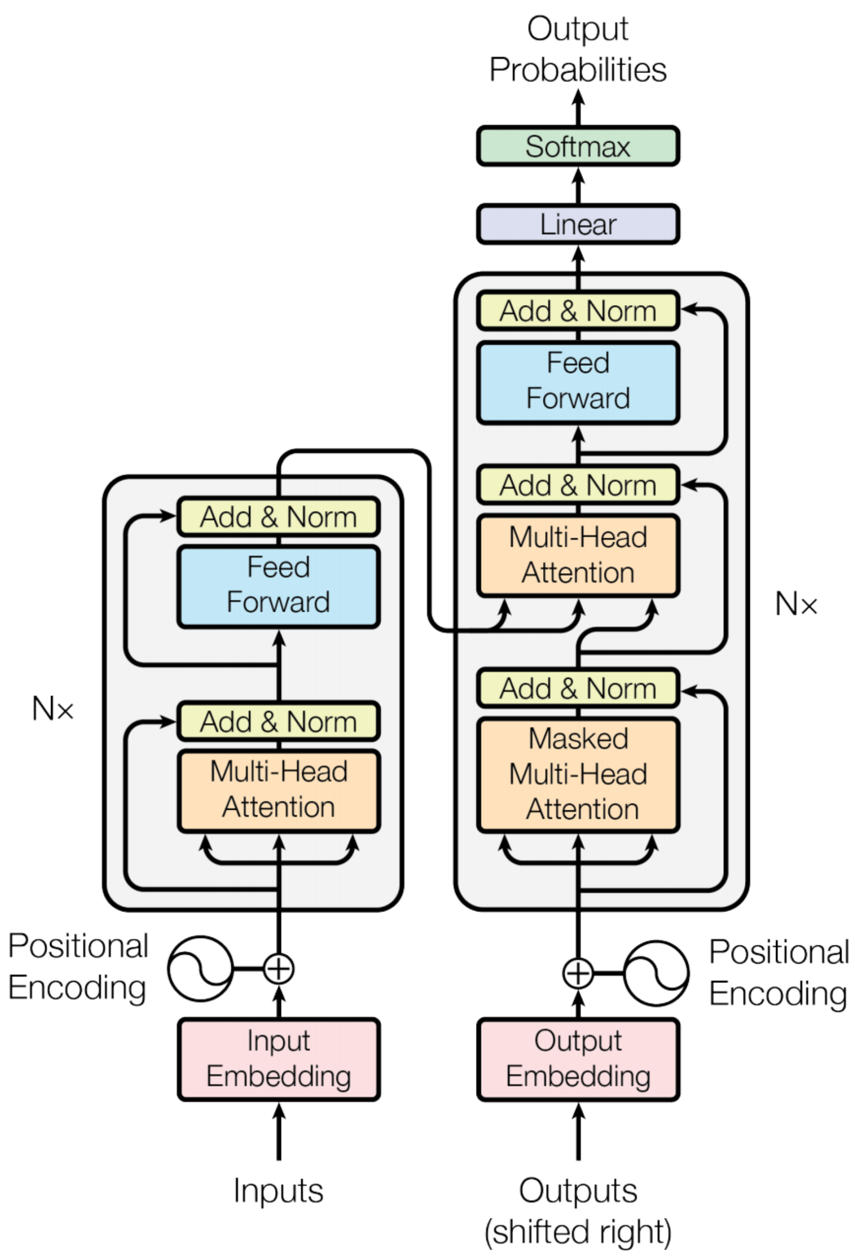


Sneak Peek

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

MODEL ARCHITECTURE

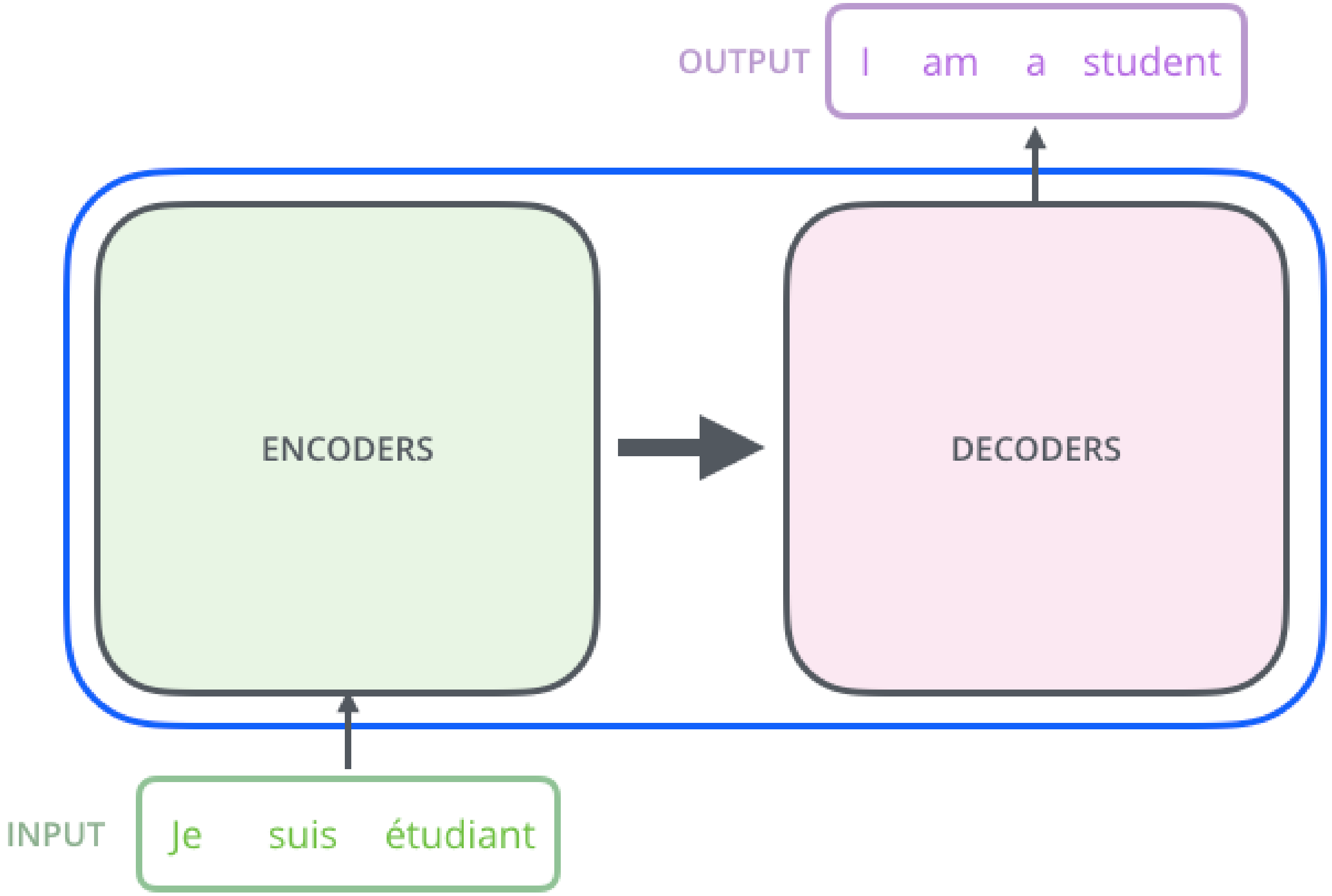


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
N×
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
N×
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

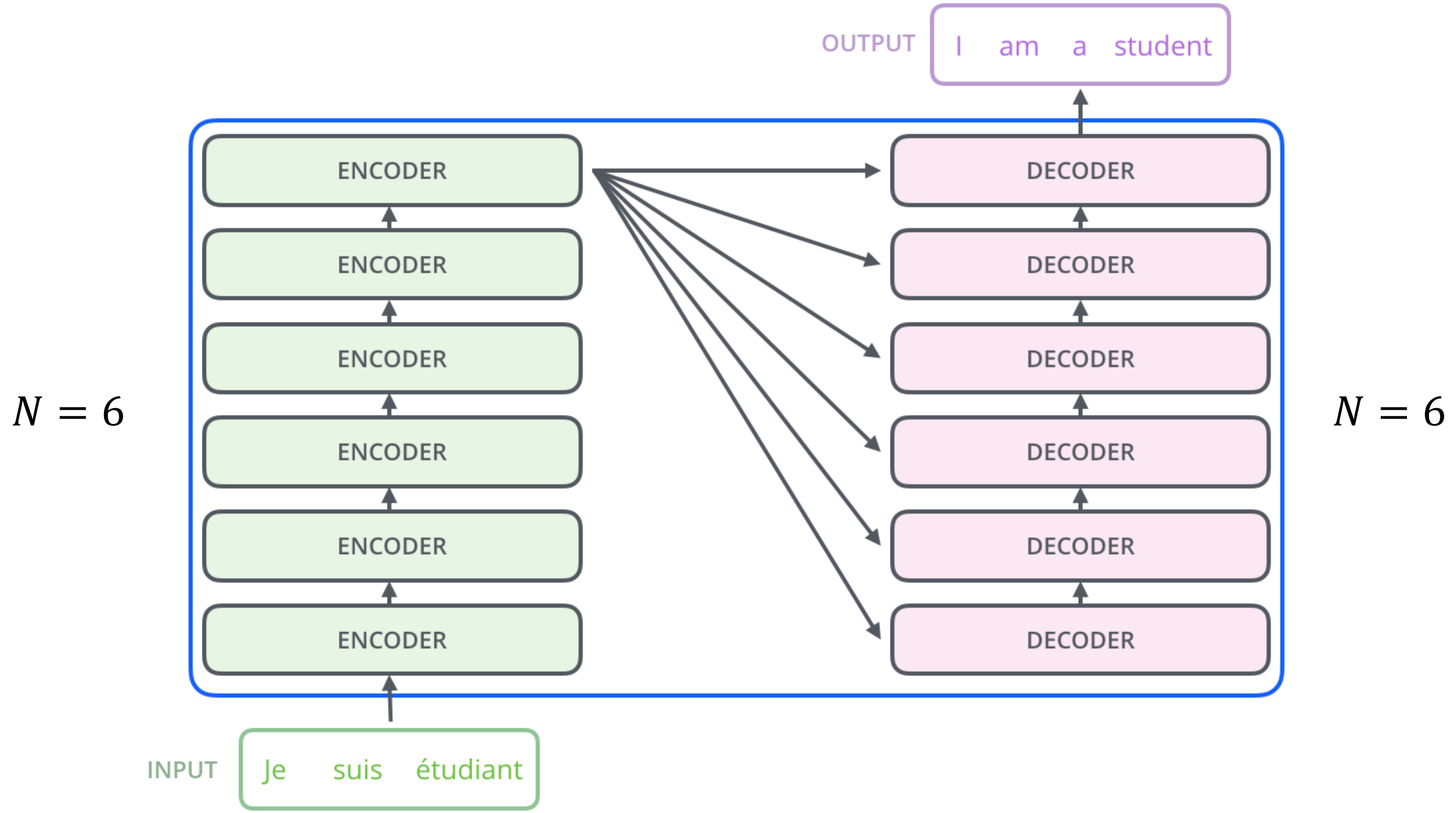
MODEL ARCHITECTURE

ENCODER AND DECODER STACKS

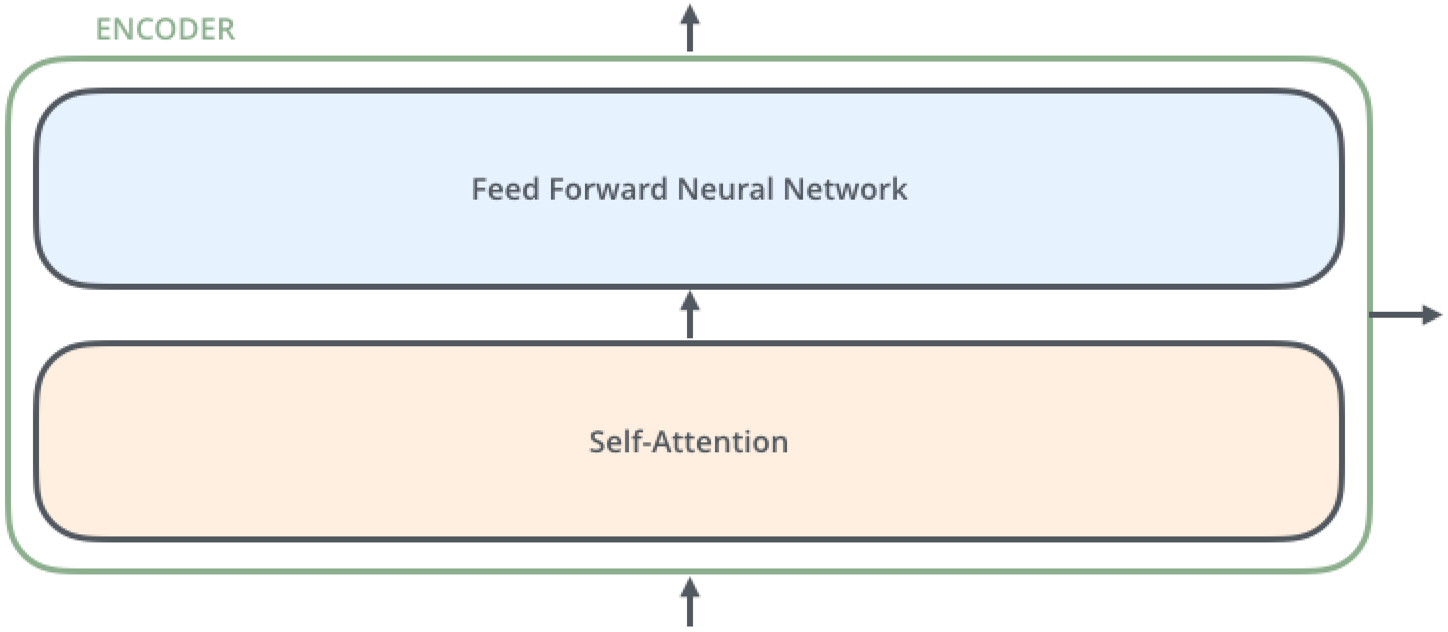
Encoder-Decoder Structure



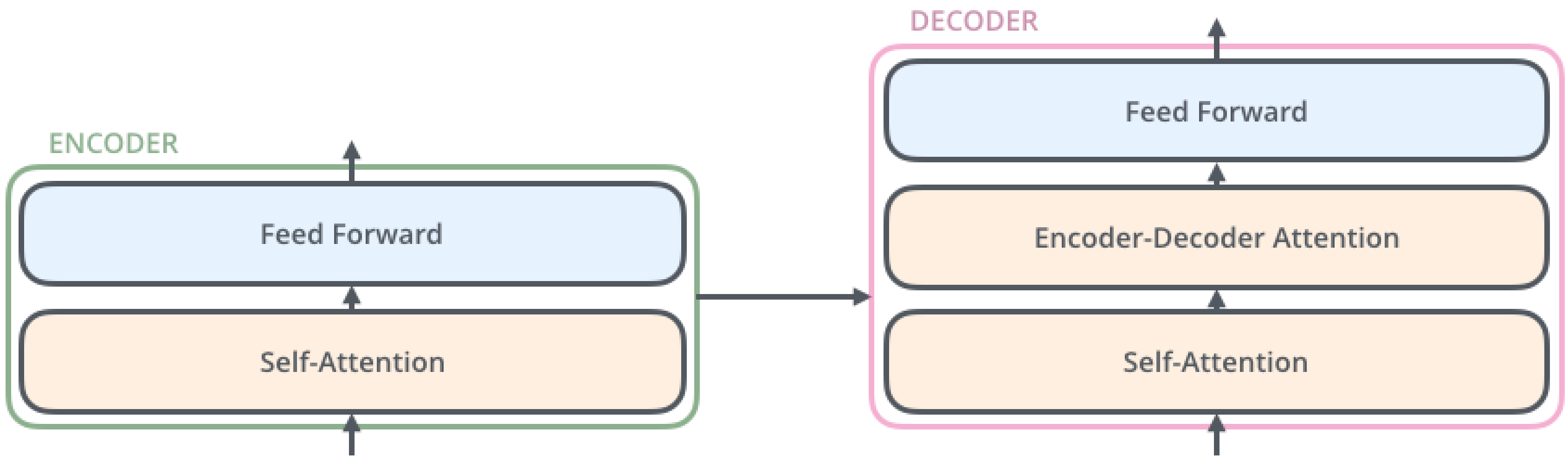
Stacked Layers



Encoder Sub-layers

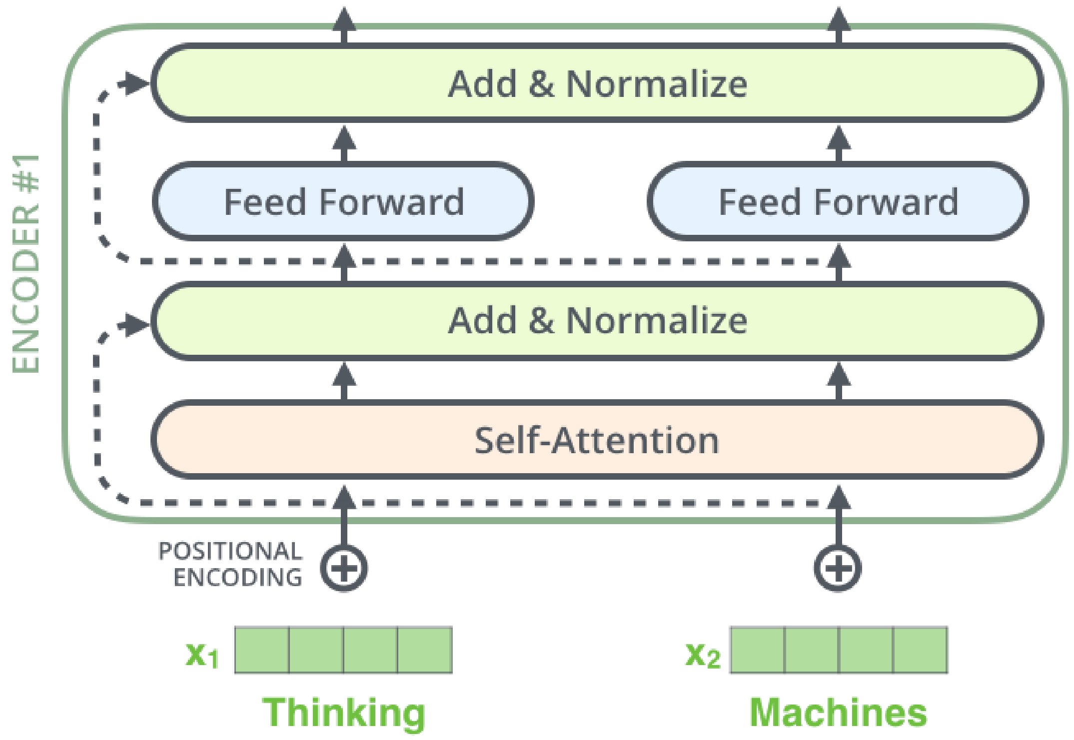


Decoder Sub-layers

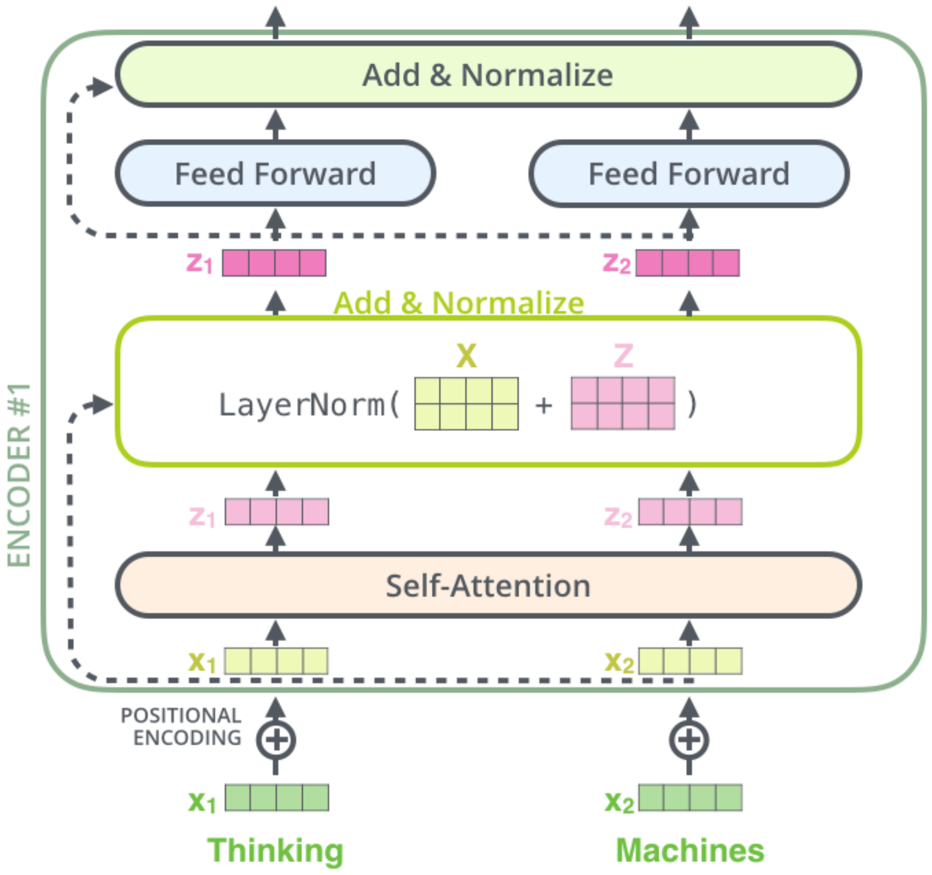


Residual Connections

- Each sub-layer has residual connection



Layer Normalization



MODEL ARCHITECTURE

ATTENTION

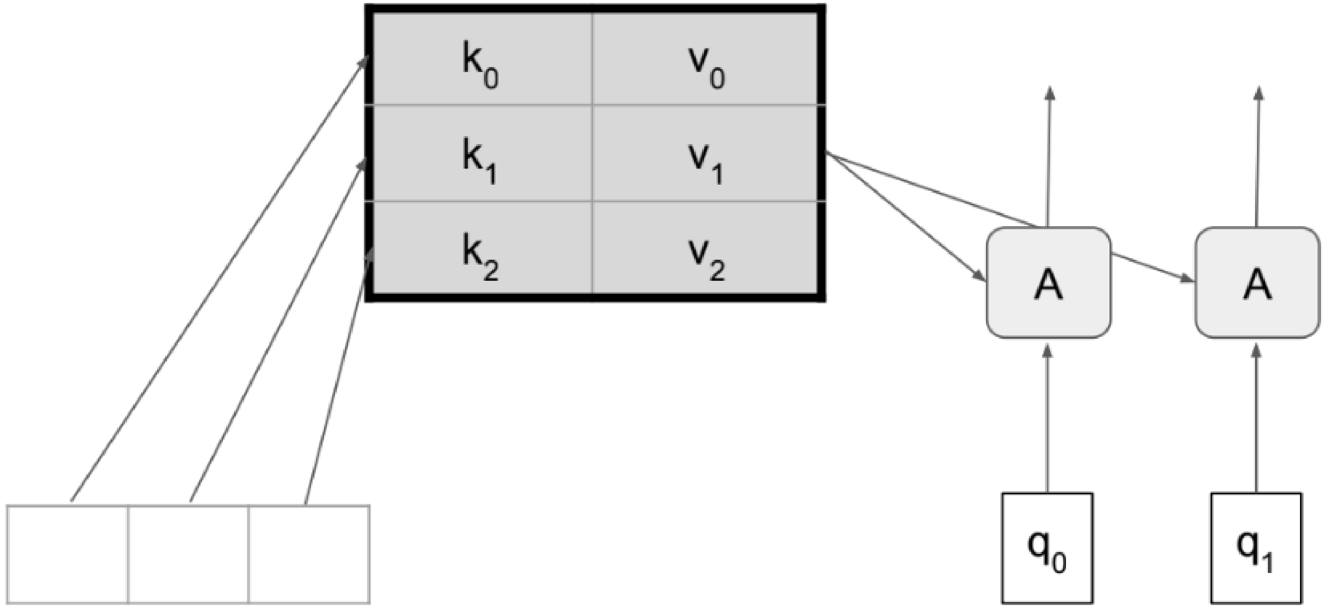
What is Attention?

- “An attention function can be described as mapping a **query** and a set of **key-value** pairs to an output, where the query, keys, values, and output are all vectors”

- There is additive and dot-product attention
  - Dot-product attention is more efficient

Dot-Product Attention

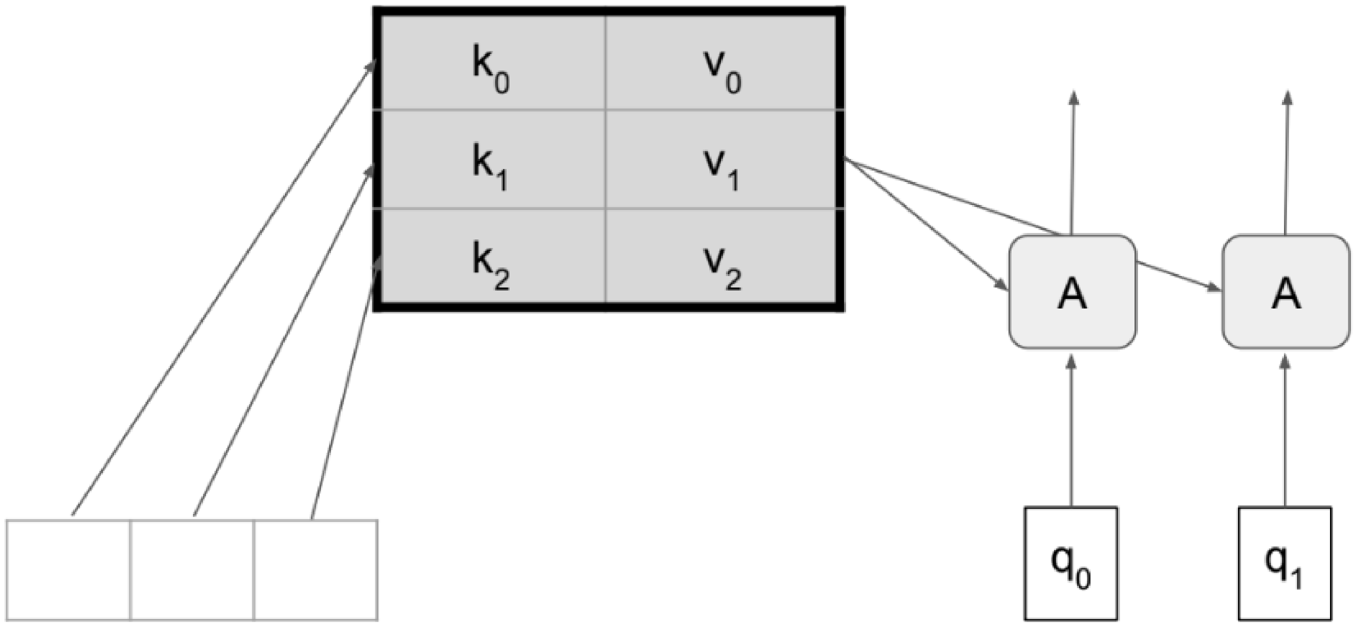
$$A(Q, K, V) = softmax(QK^T)V$$



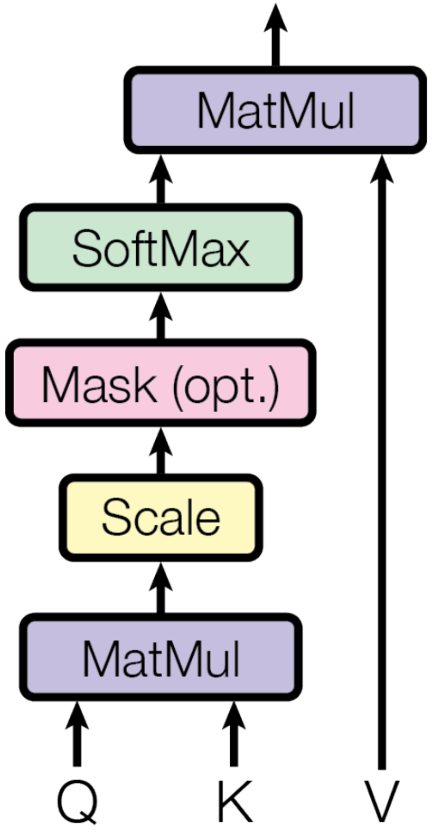
Scaled Dot-Product Attention

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

Scale $\sqrt{d_k}$

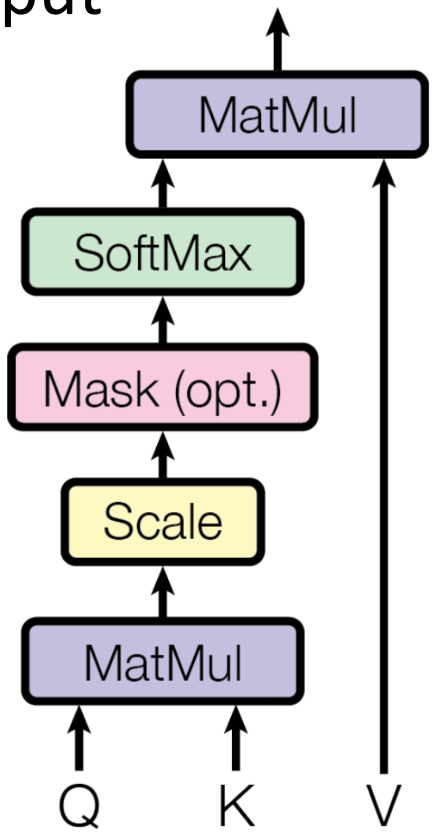


Scaled Dot-Product Attention



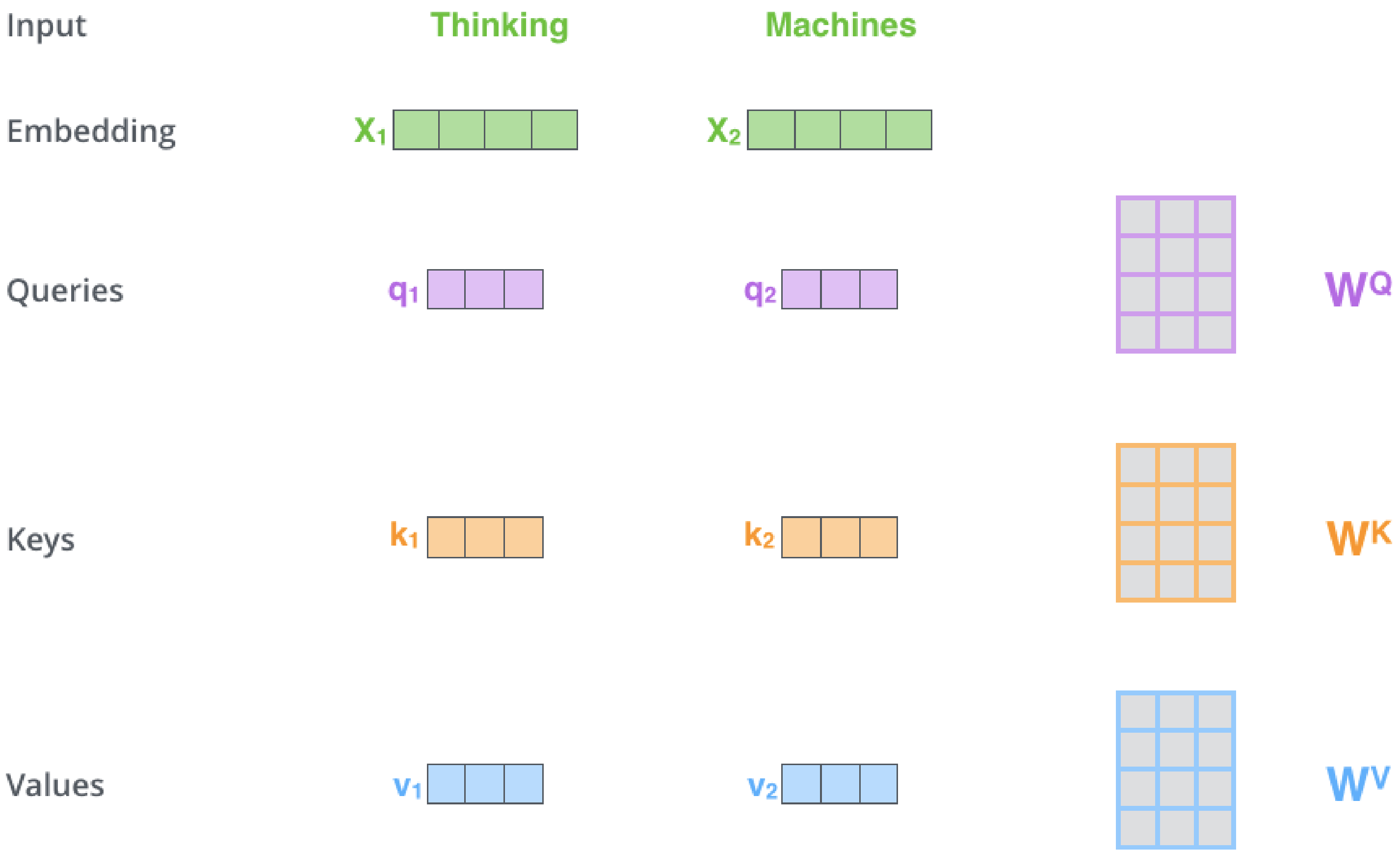
Scaled Dot-Product Attention: Vectors

- "Mapping a **query** and a set of **key-value pairs** to an output"
  - $query, key \in \mathbb{R}^{d_k}$
  - $value \in \mathbb{R}^{d_v}$

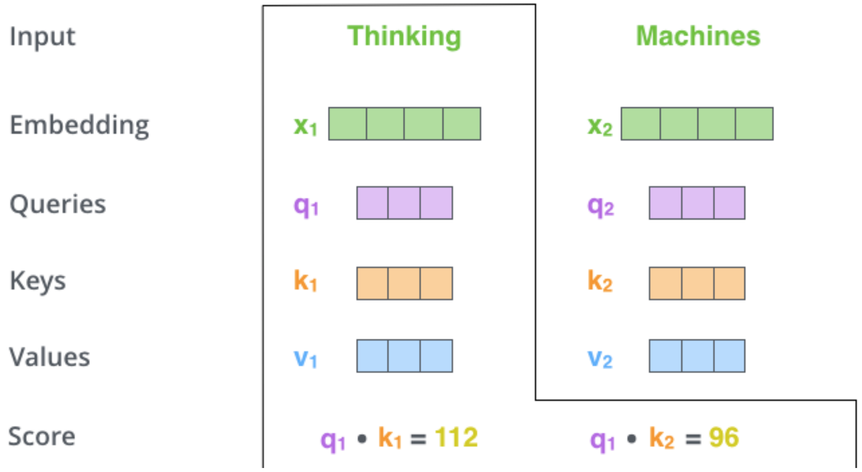


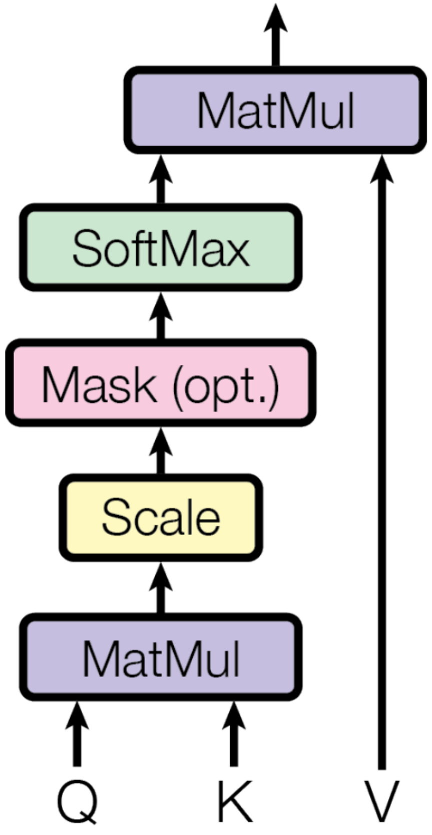
1. Compute dot product of query with all keys
2. Divide each by $\sqrt{d_k}$
3. Apply softmax to get weights of values

Scaled Dot-Product Attention: Vectors

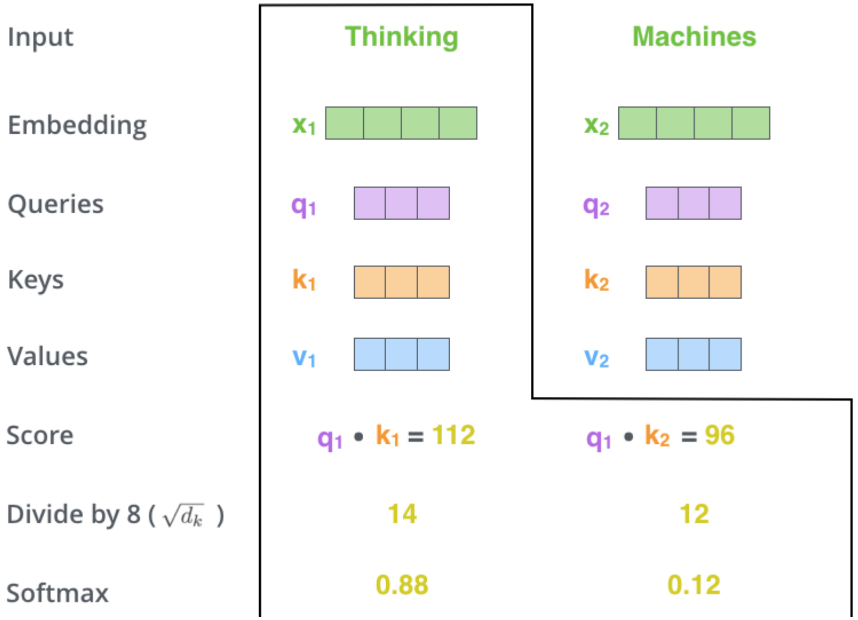


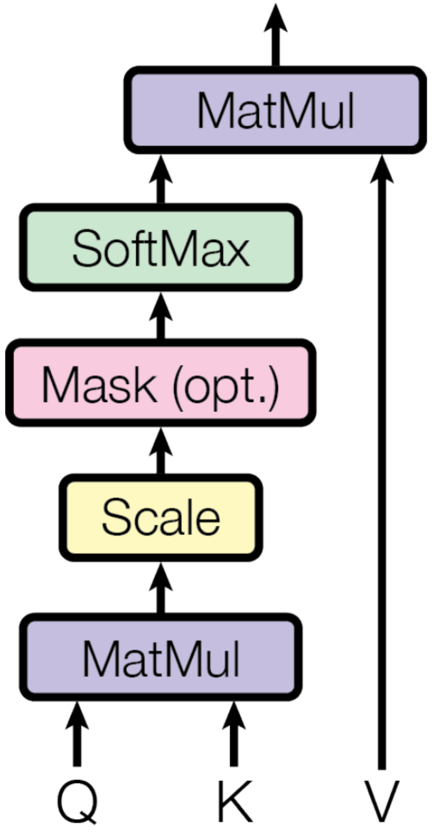
Scaled Dot-Product Attention: Vectors



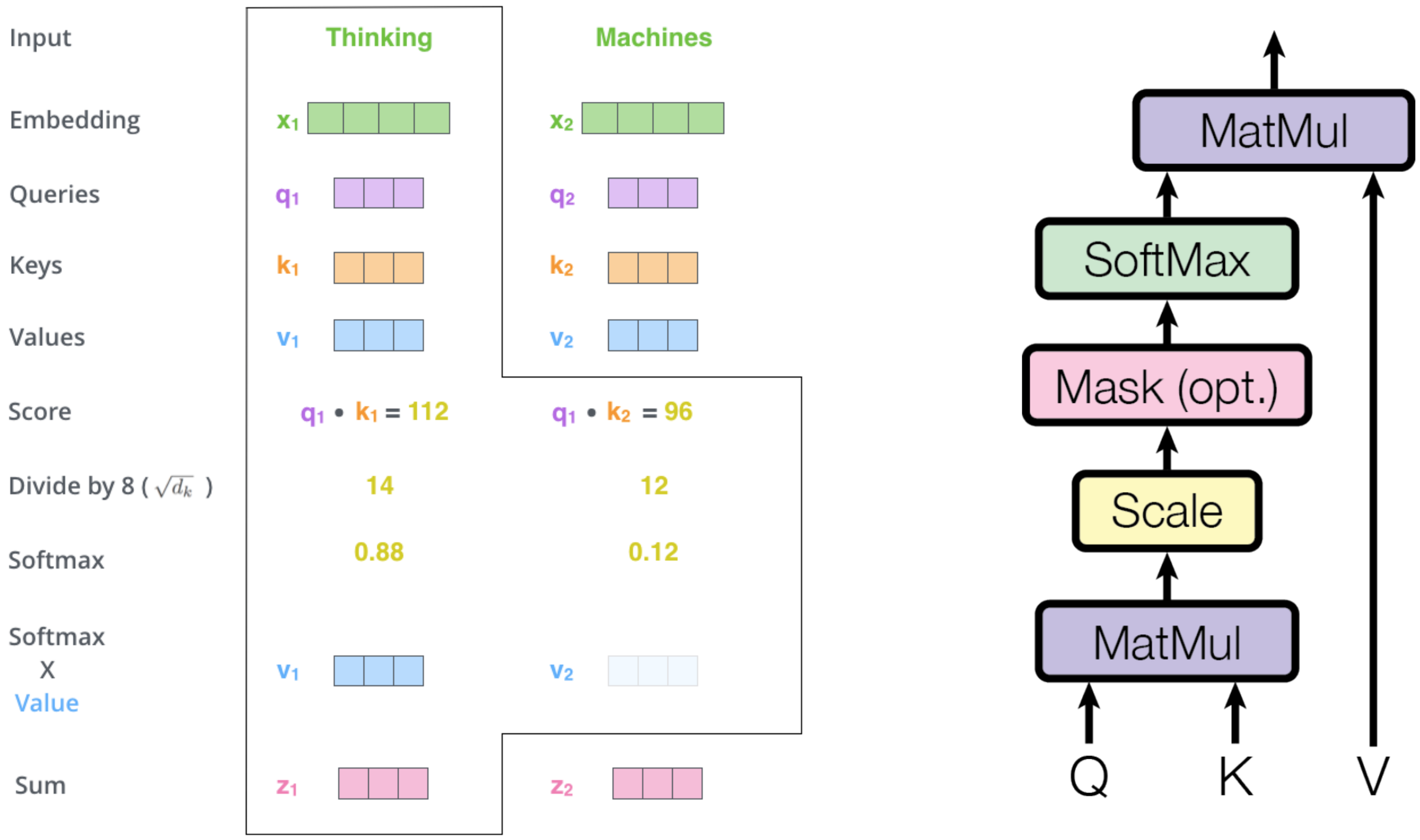


Scaled Dot-Product Attention: Vectors

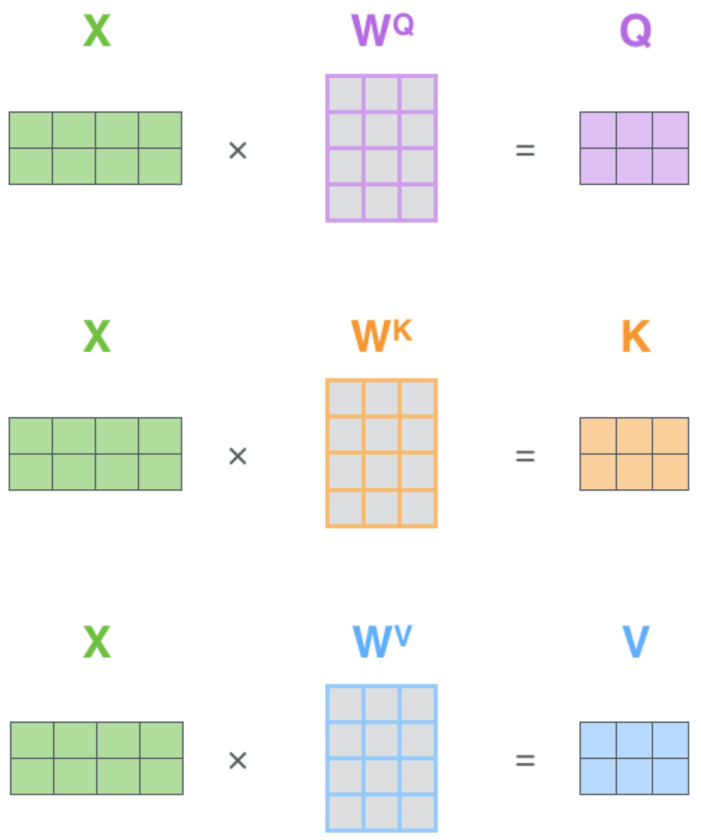


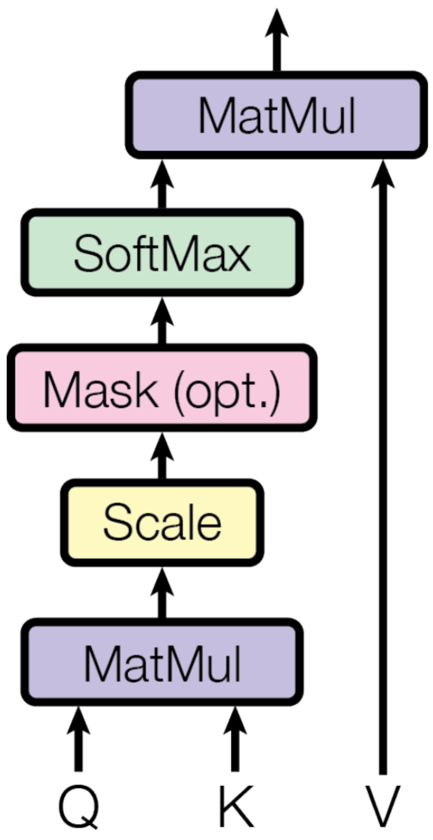


Scaled Dot-Product Attention: Vectors

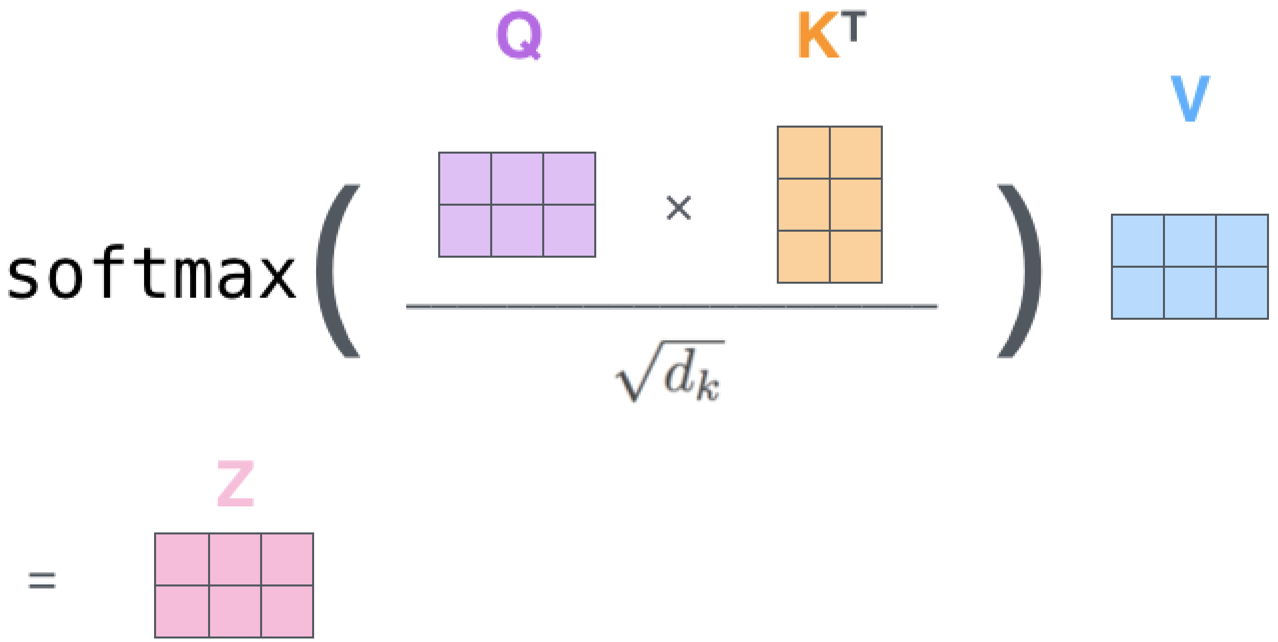


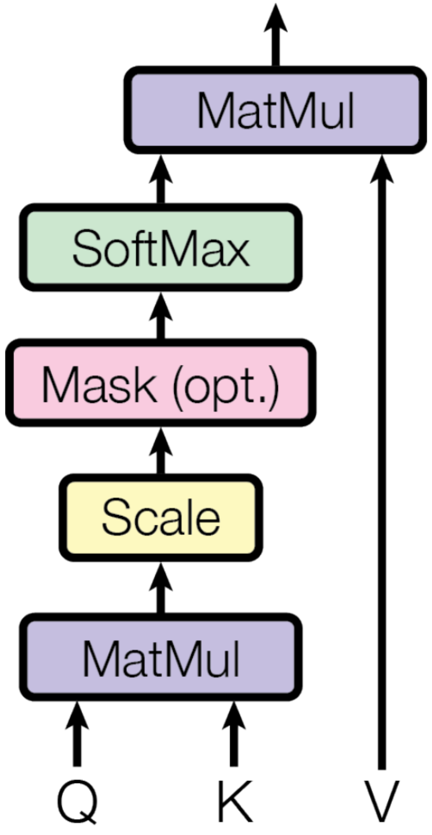
Scaled Dot-Product Attention: Matrix Form



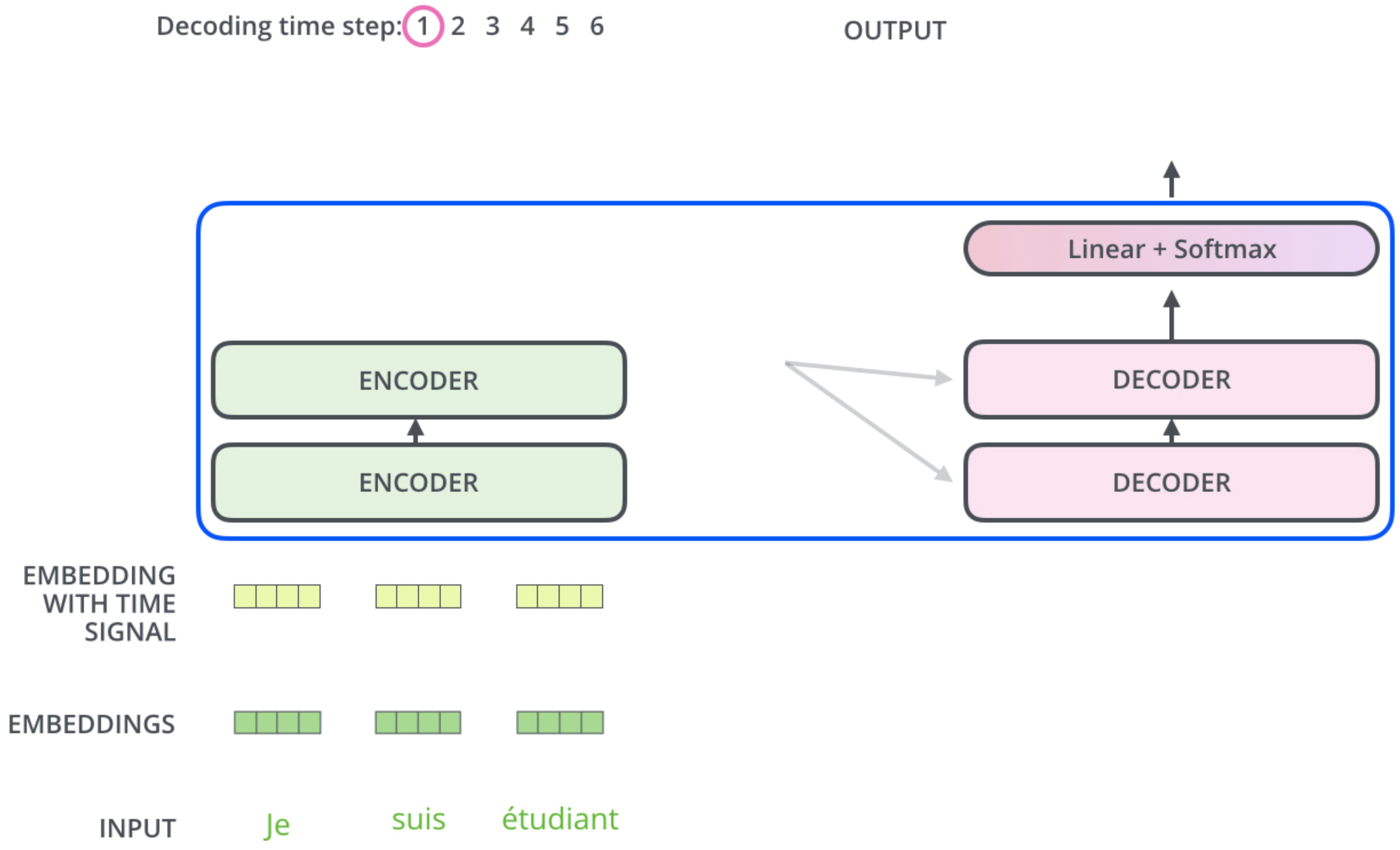


Scaled Dot-Product Attention: Matrix Form

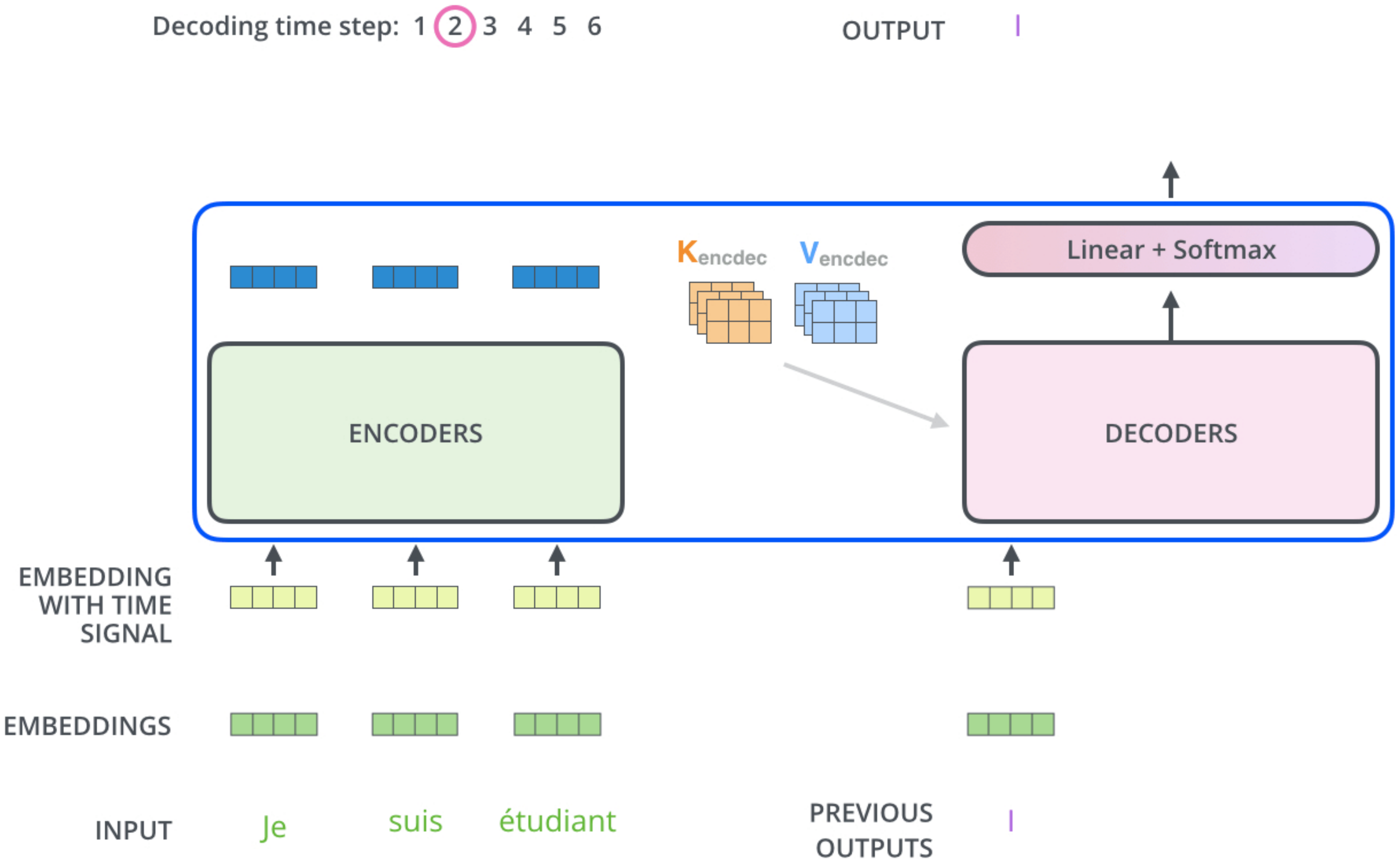




Decoder Attention: One step



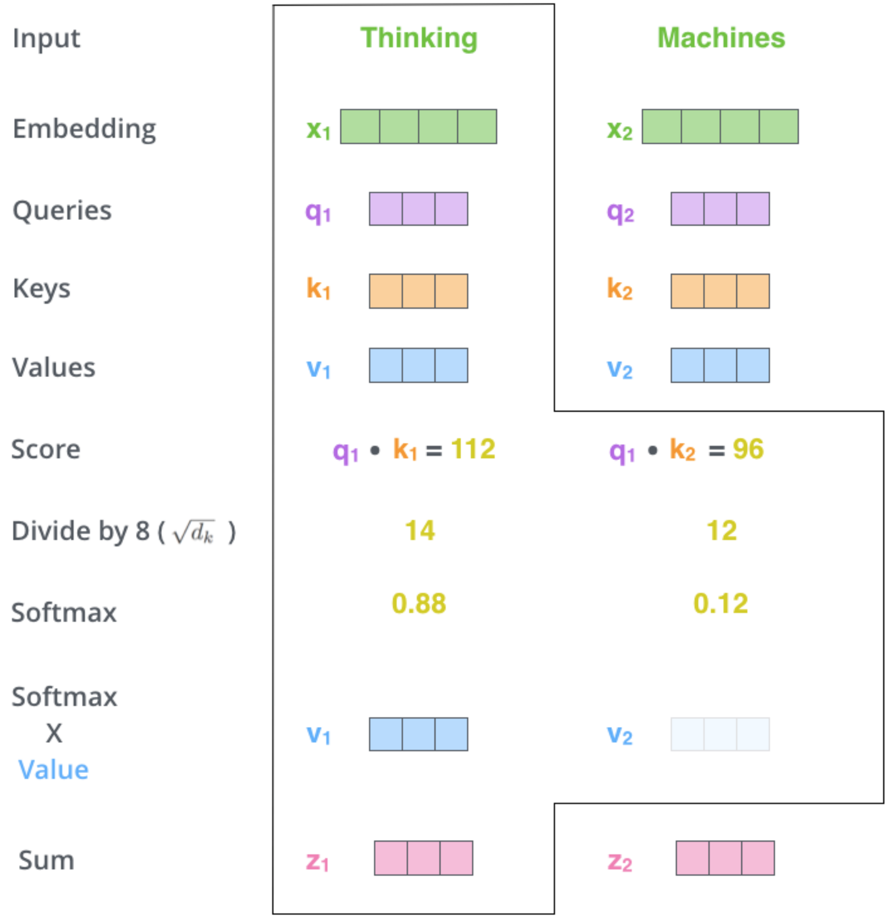
Decoder Attention: Remainder



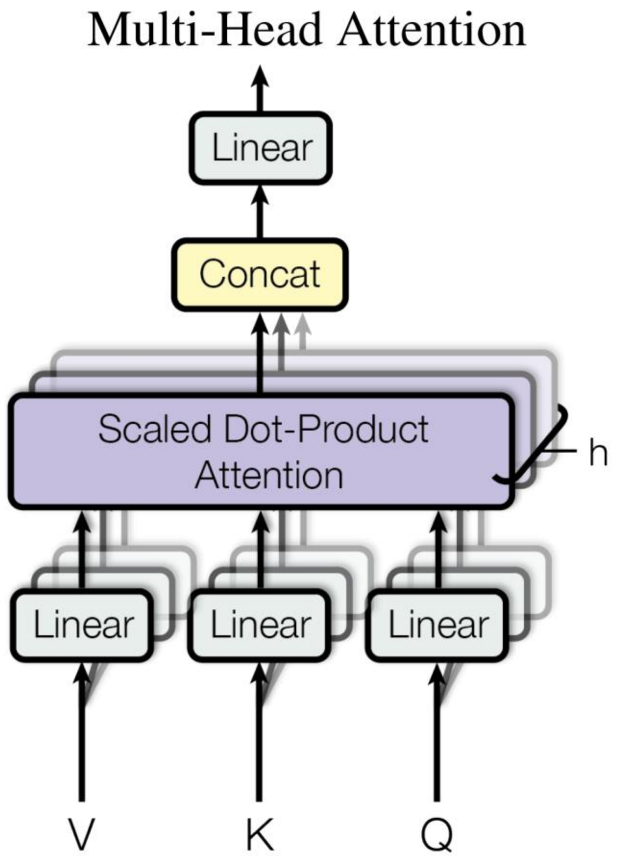
Visualization of Attentions

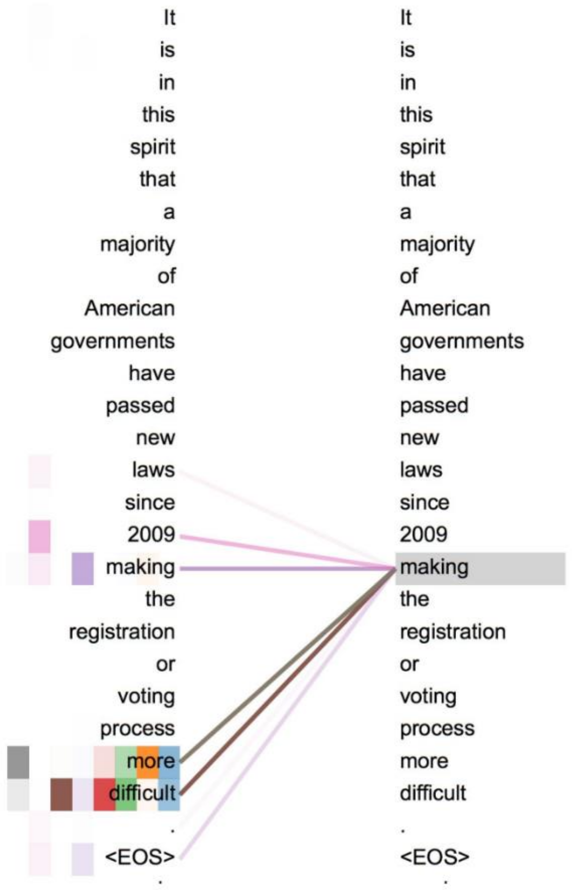
What’s Missing from Self-Attention?

- Self-attention is just a weighted average of values
- Can we use different linear transformations like CNN filters?



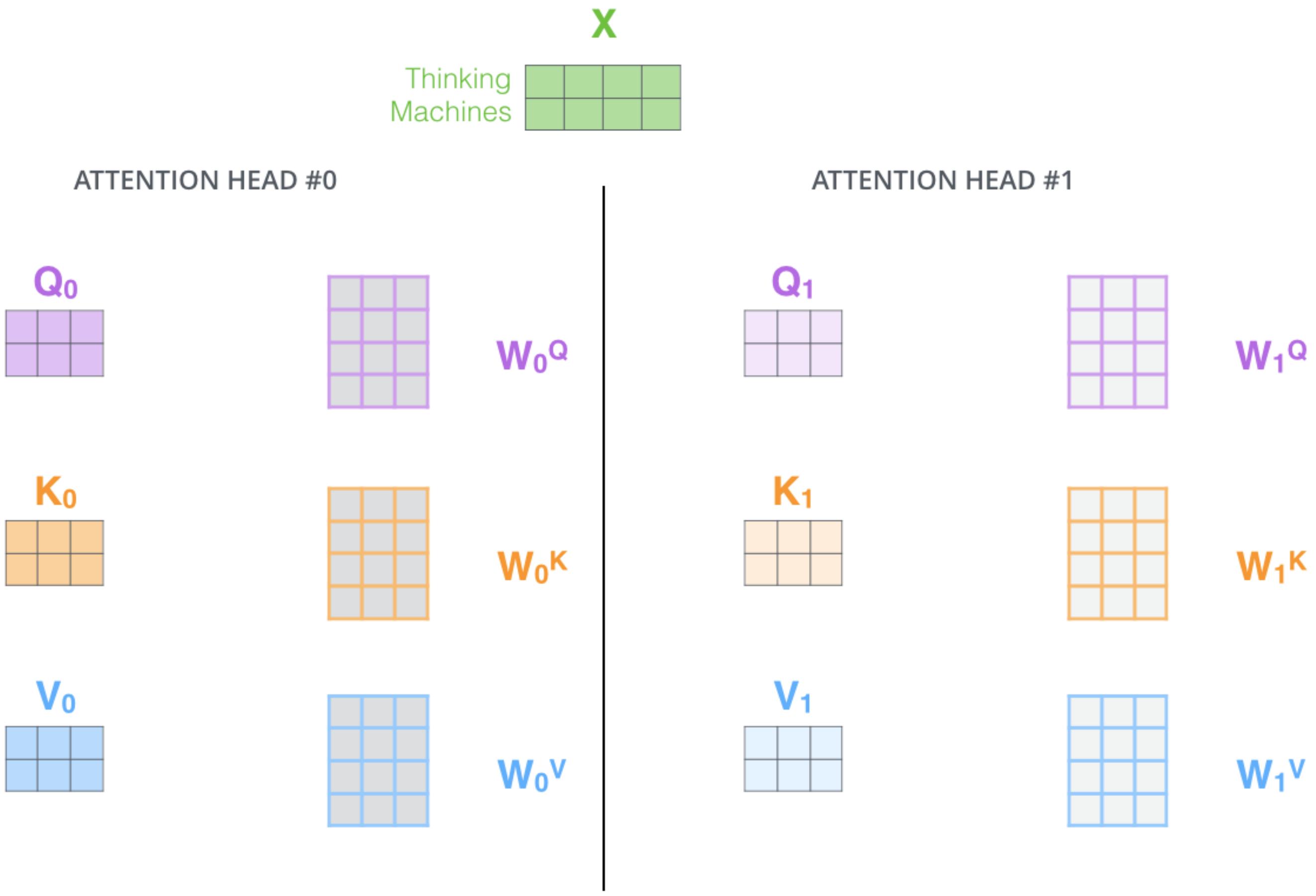
Multi-Head Attention



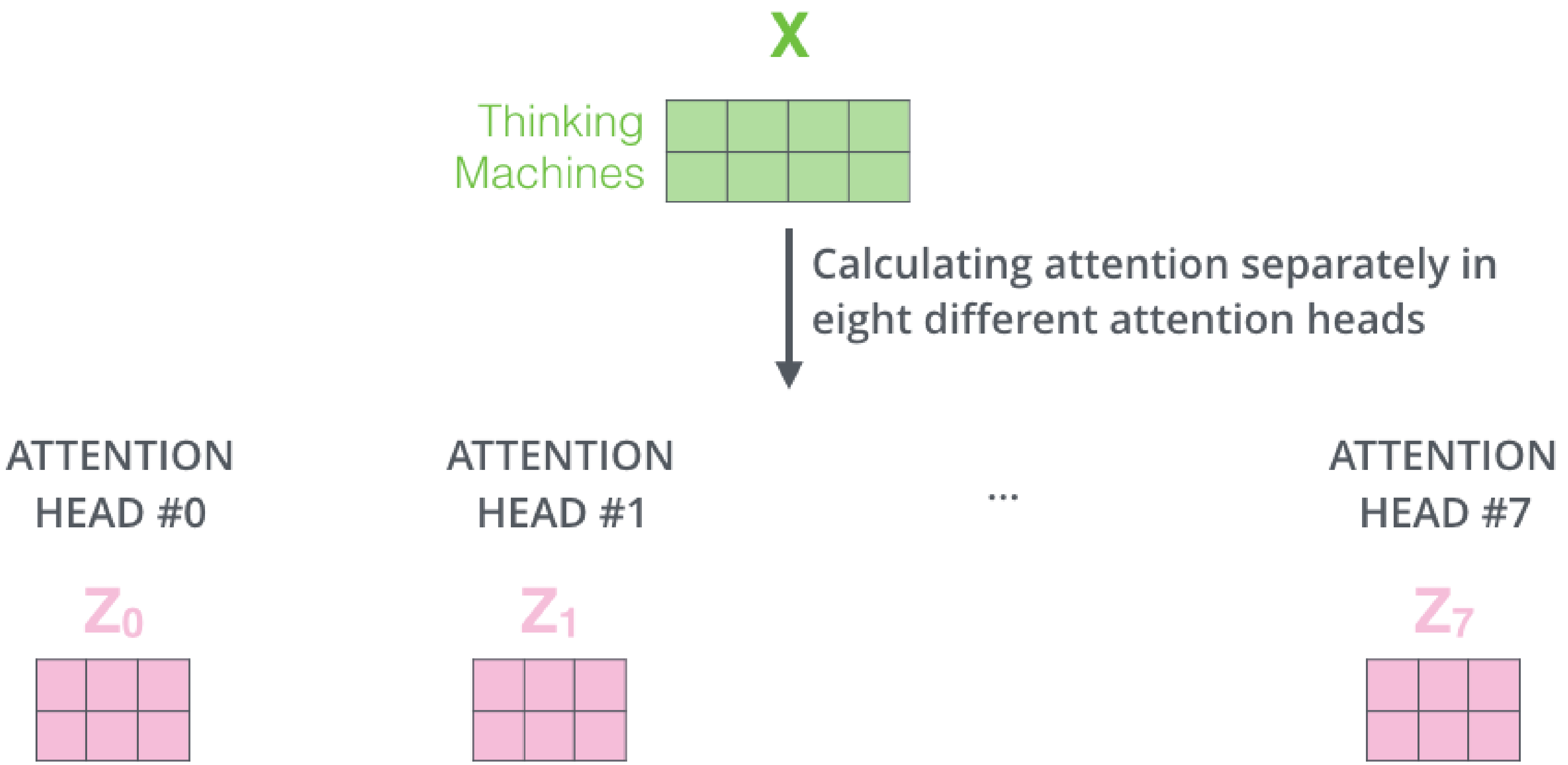


$h = 8$

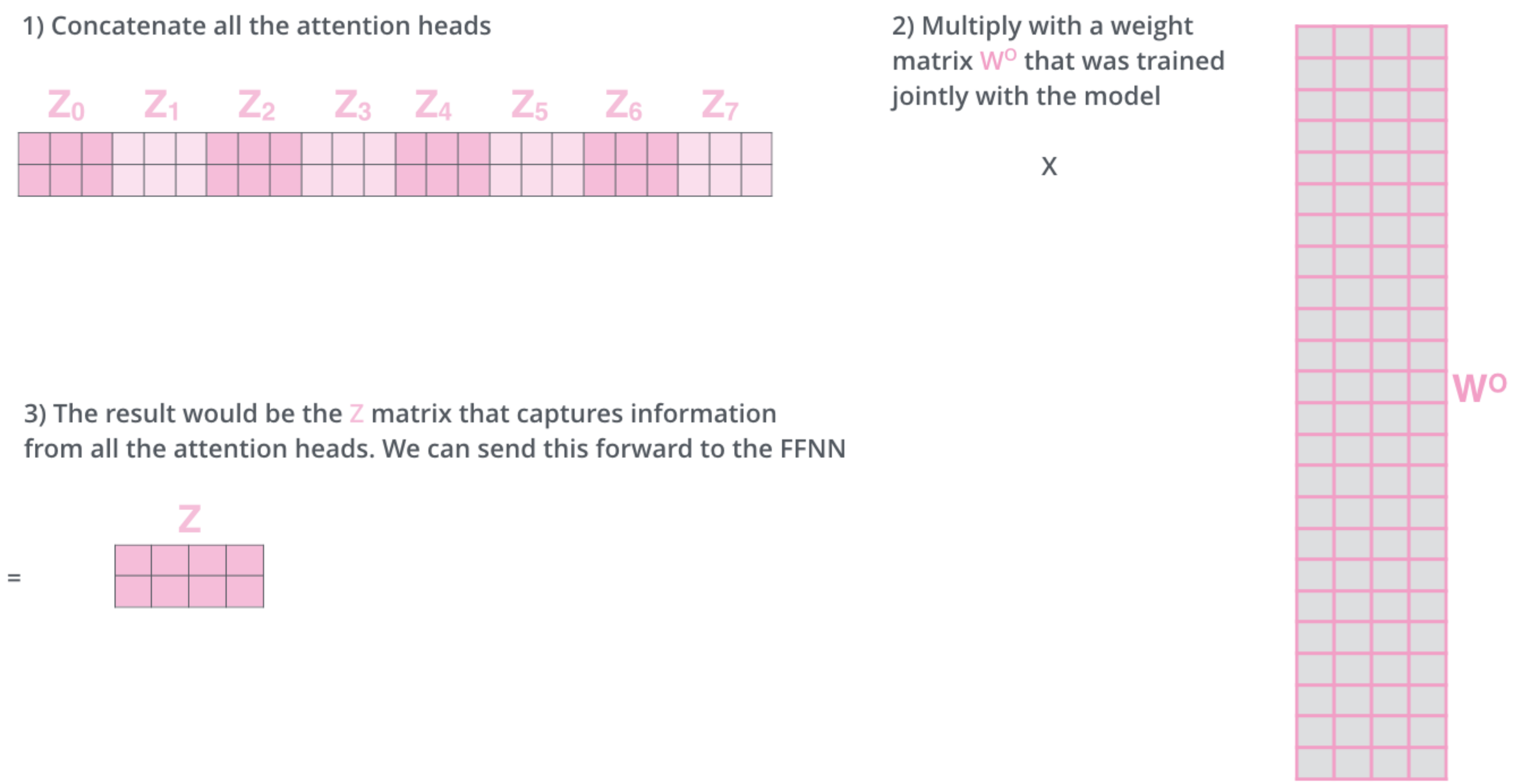
Multi-Head Attention



Multi-Head Attention: 8 Heads



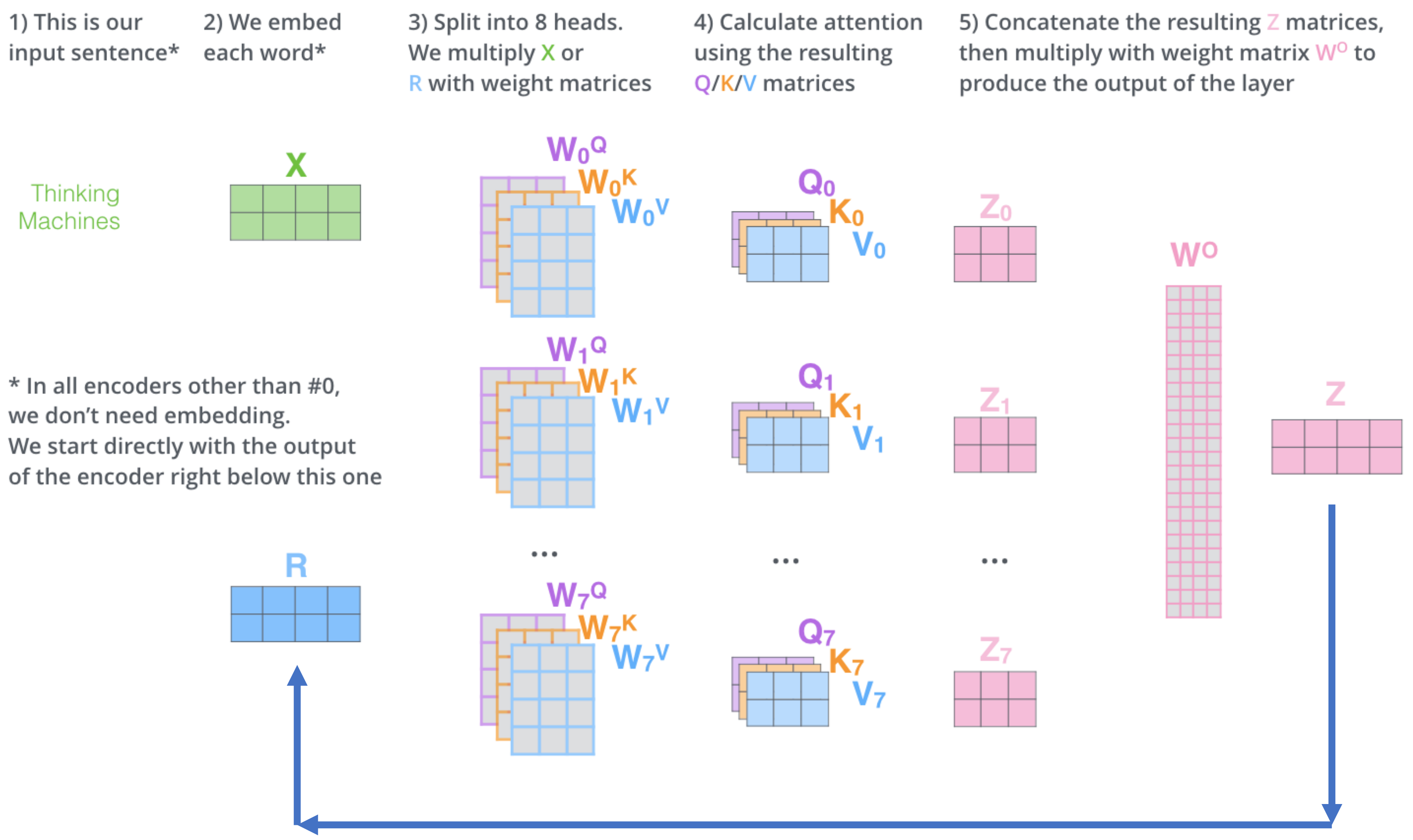
Multi-Head Attention: Concatenation



Multi-Head Attention: Details

- Each Q, K, V matrices are randomly initialized
- To have similar computational cost as single-head attention, we use smaller $d_k = d_v = \frac{d_{model}}{h} = 64$.

Attention: Recap



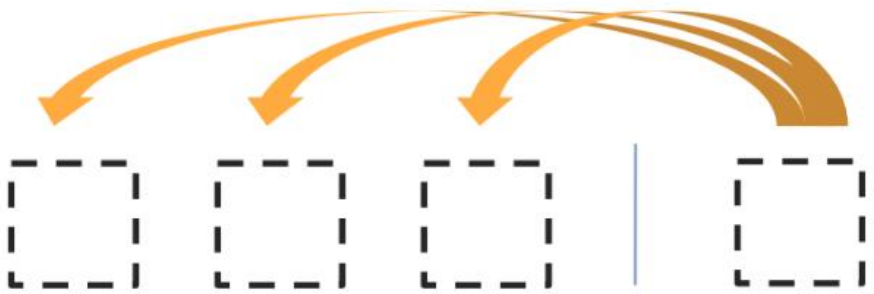
Applications of Attention

1. Encoder-Decoder Attention
   - Query from previous decoder layer
   - Key, Value from encoder output
2. Encoder Self-Attention
   - Query, Key, Value from previous encoder layer
3. Decoder Self-Attention
   - Query, Key, Value from previous decoder layer
   - Prevent leftward information flow through **masking**

Masking for Decoder Self-Attention

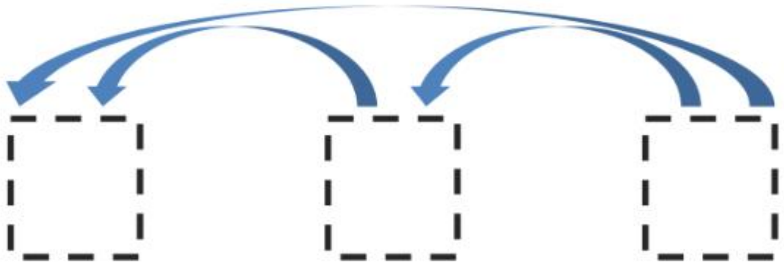
- Decoder cannot look at subsequent positions
- Decoder attention should be **masked**

Applications of Attention



Encoder-Decoder Attention

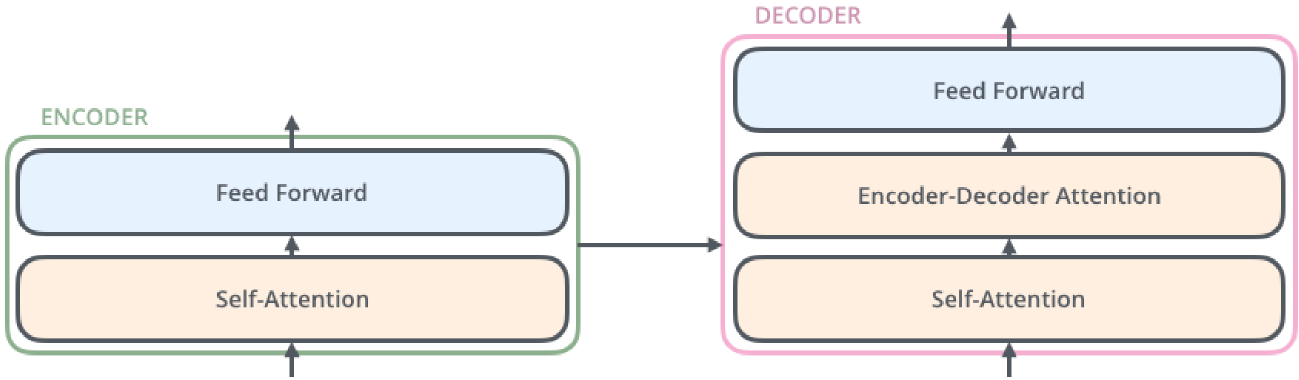
Encoder Self-Attention



MaskedDecoder Self-Attention

MODEL ARCHITECTURE
POSITION-WISE FEED-FORDWARD NETWORKS

Position-wise Feed-Forward Networks



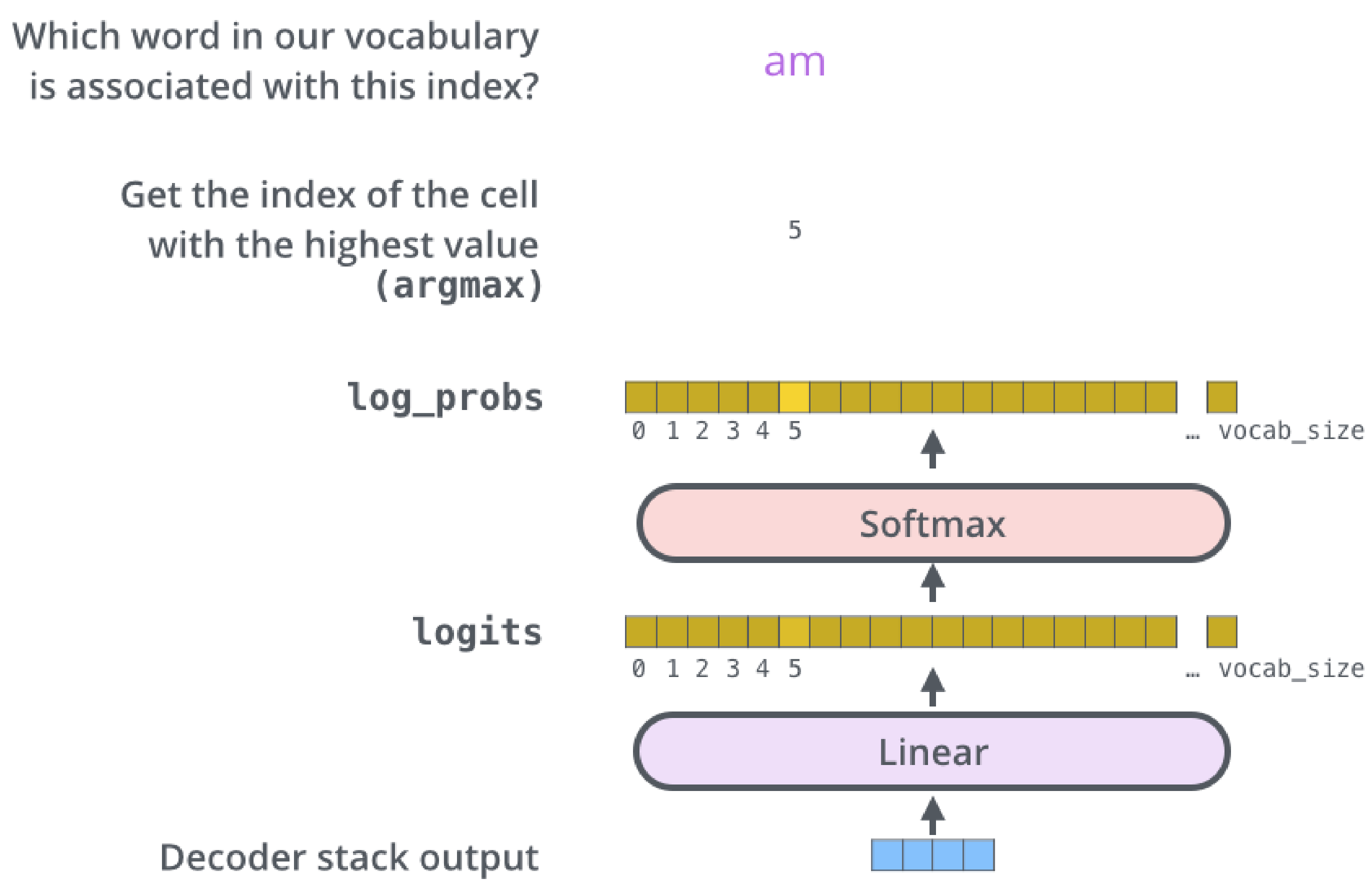
$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- Same linear transformation across different positons
- Different parameters for each layer

MODEL ARCHITECTURE

EMBEDDINGS AND SOFTMAX

Decoder Softmax



MODEL ARCHITECTURE

POSITIONAL ENCODING

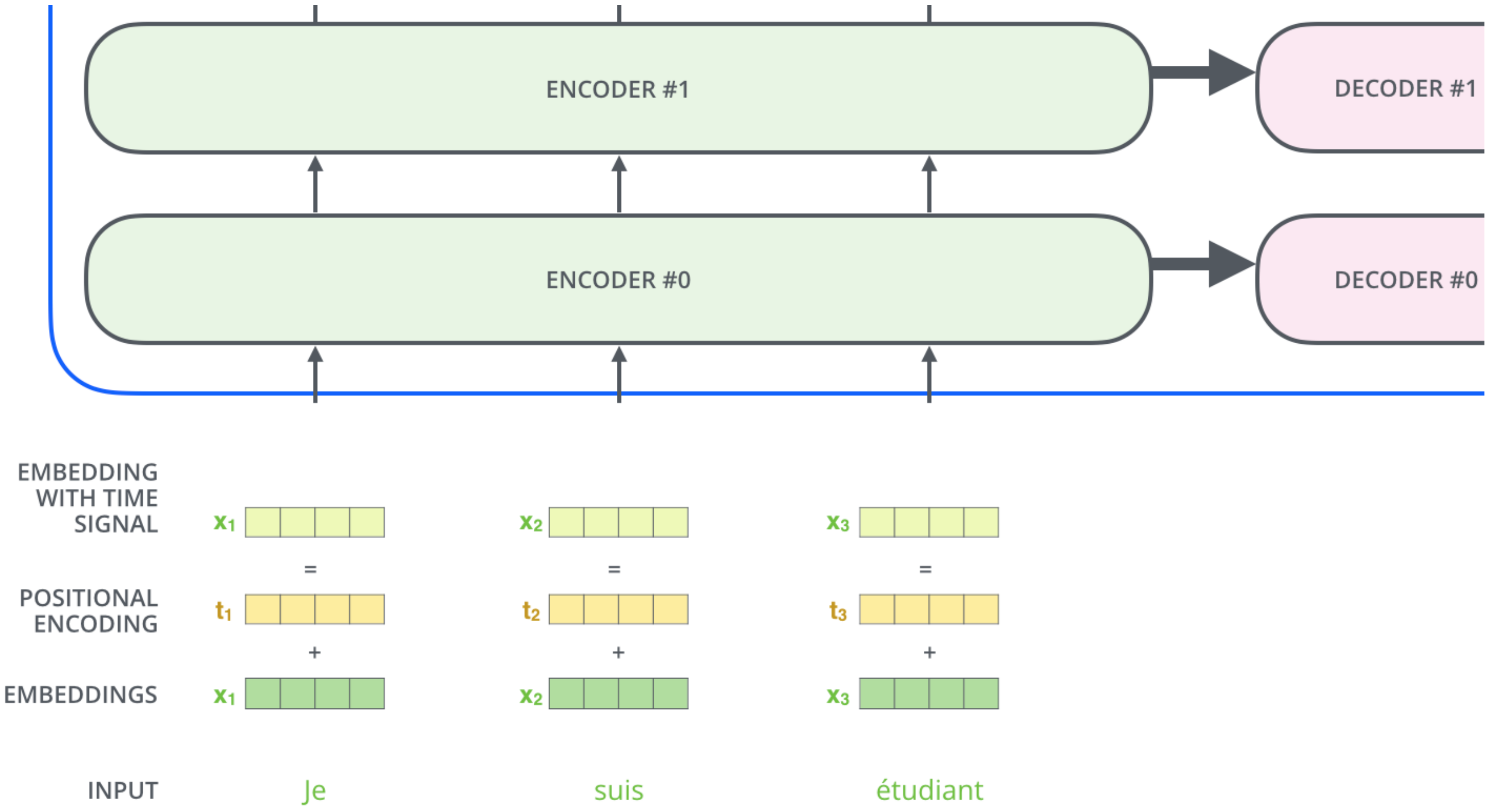
Rationale

- Transformer uses neither recurrence nor convolution
- Need to inject positional information

→ **Add** Positional Encoding (Learned or **Fixed**)

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

Positional Encoding



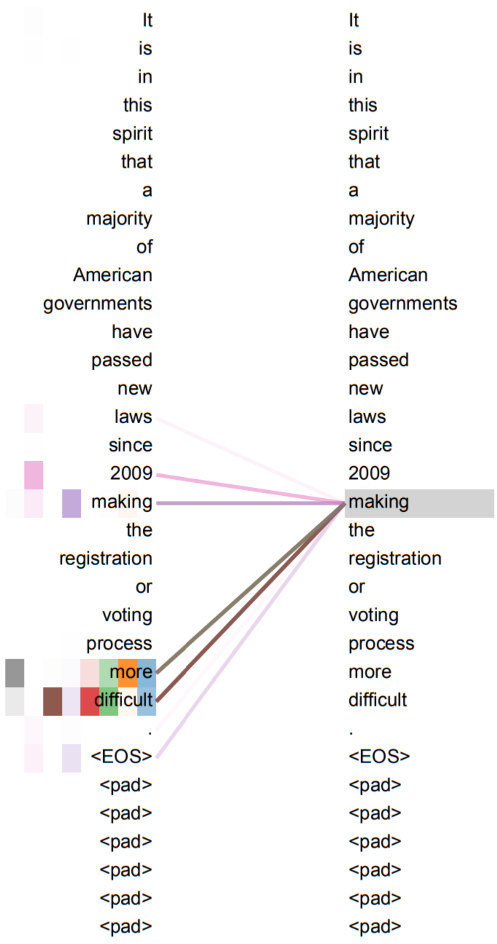
WHY SELF-ATTENTION

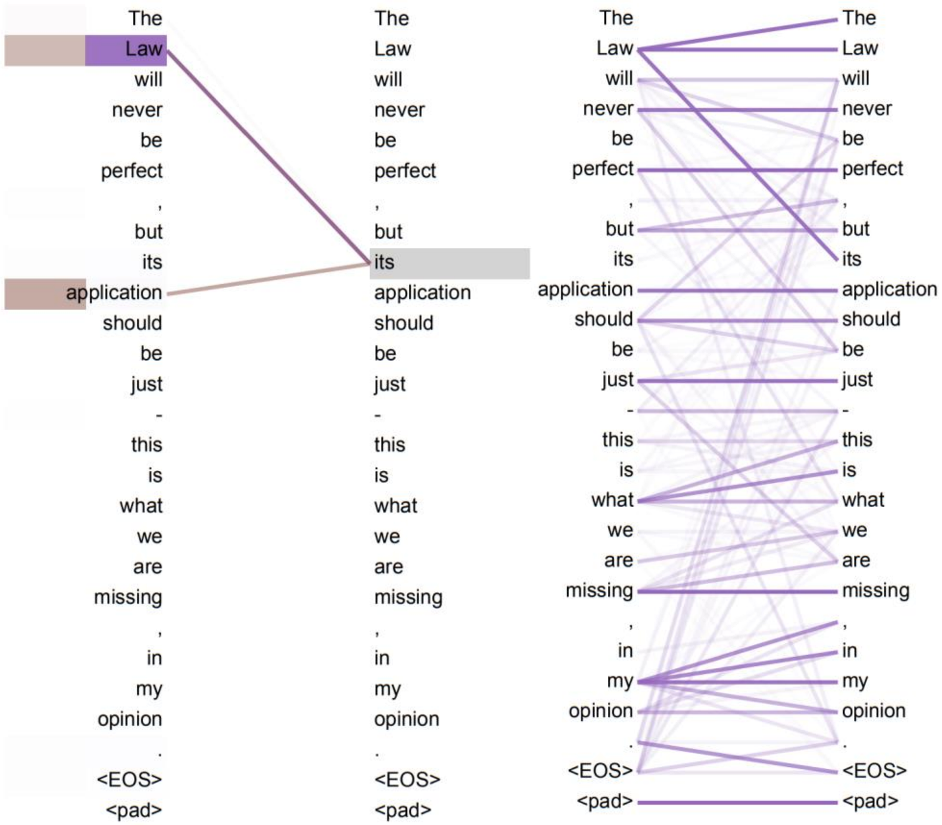
Comparison with Previous Methods

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

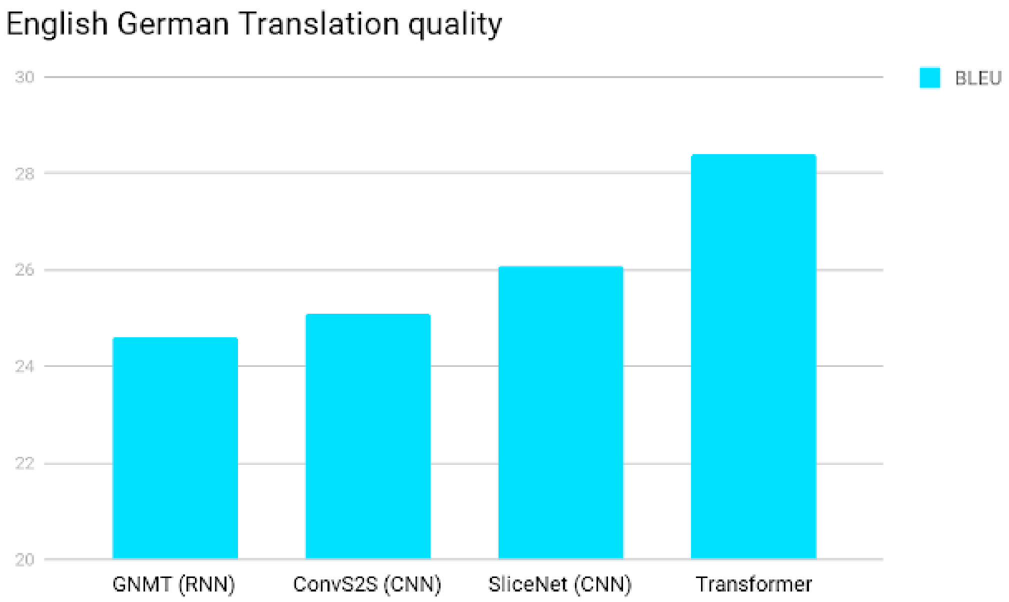
Interpretability

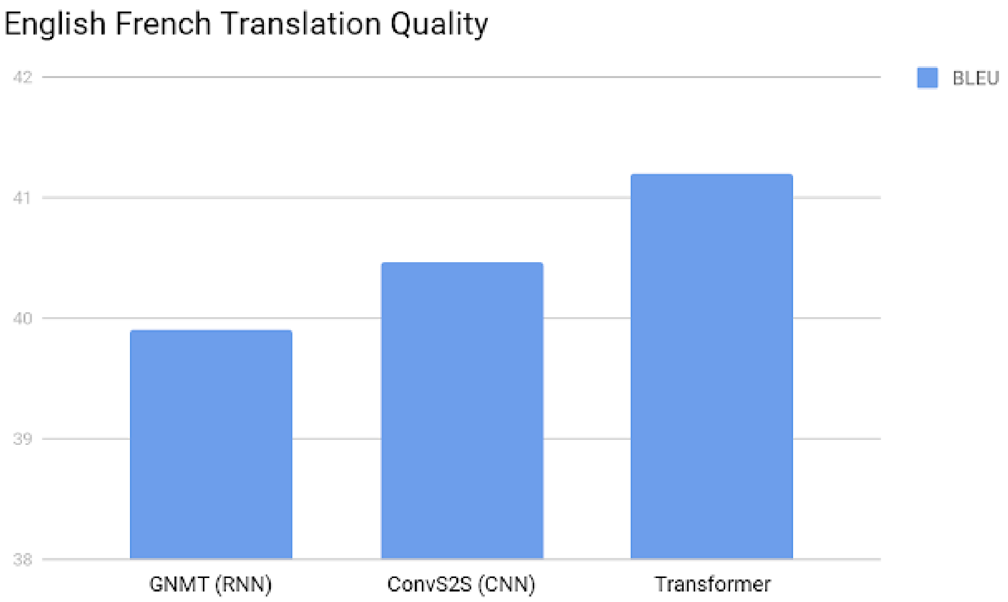




RESULTS

BLEU Scores


English German Translation quality
BLEU
30
28
26
24
22
20
GNMT (RNN)
ConvS2S (CNN)
SliceNet (CNN)
Transformer


English French Translation Quality
BLEU
42
41
40
39
38
GNMT (RNN)
ConvS2S (CNN)
Transformer

BLEU Scores and Training Cost

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Variations

	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

Generalization to other tasks

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

THANK YOU FOR YOUR ATTENTION

Resources

Posts

- *The Illustrated Transformer* by Jay Alammar
- *Paper Dissected: "Attention is All You Need" Explained* by Keita Kurita
- *The Transformer – Attention is all you need* by Michal Chromiak
- Transformer by Jakob Uszkoreit (Google AI Blog)

Video

- *Tensor2Tensor Transformers* by Lukasz Kaiser

Resources

Code

- Tensor2Tensor (Official)
- Tensor2Tensor Colab (Official)
- Annotated Transformer (Harvard NLP)

Slides

- End-to-End AI