
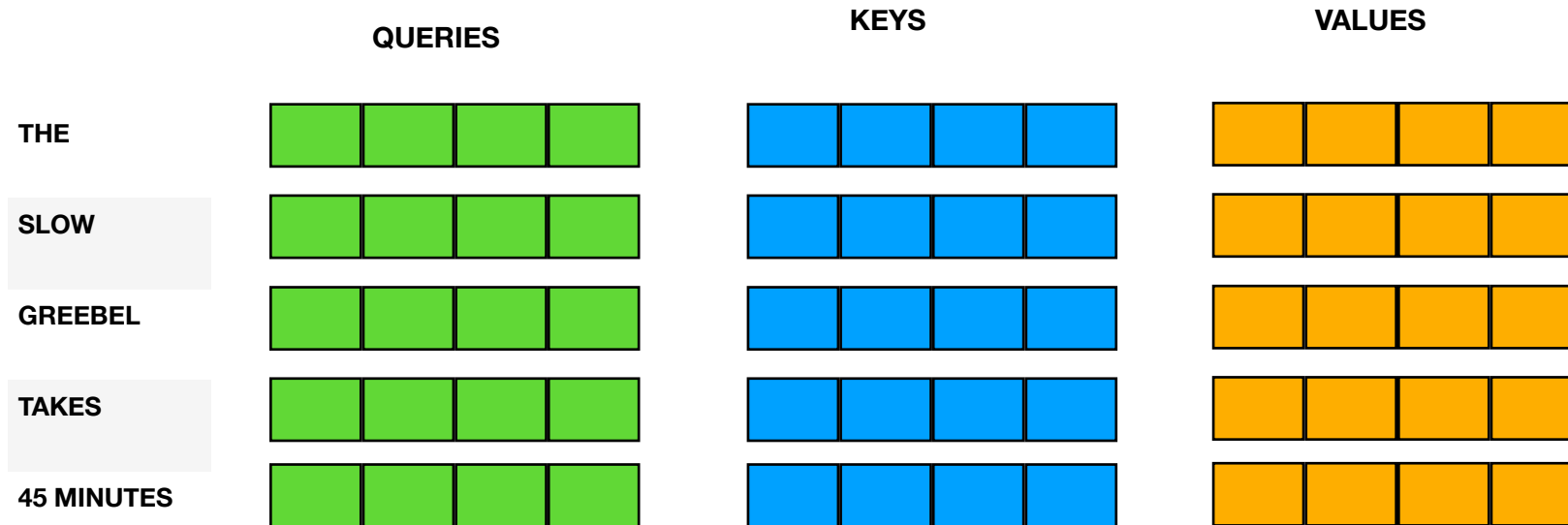


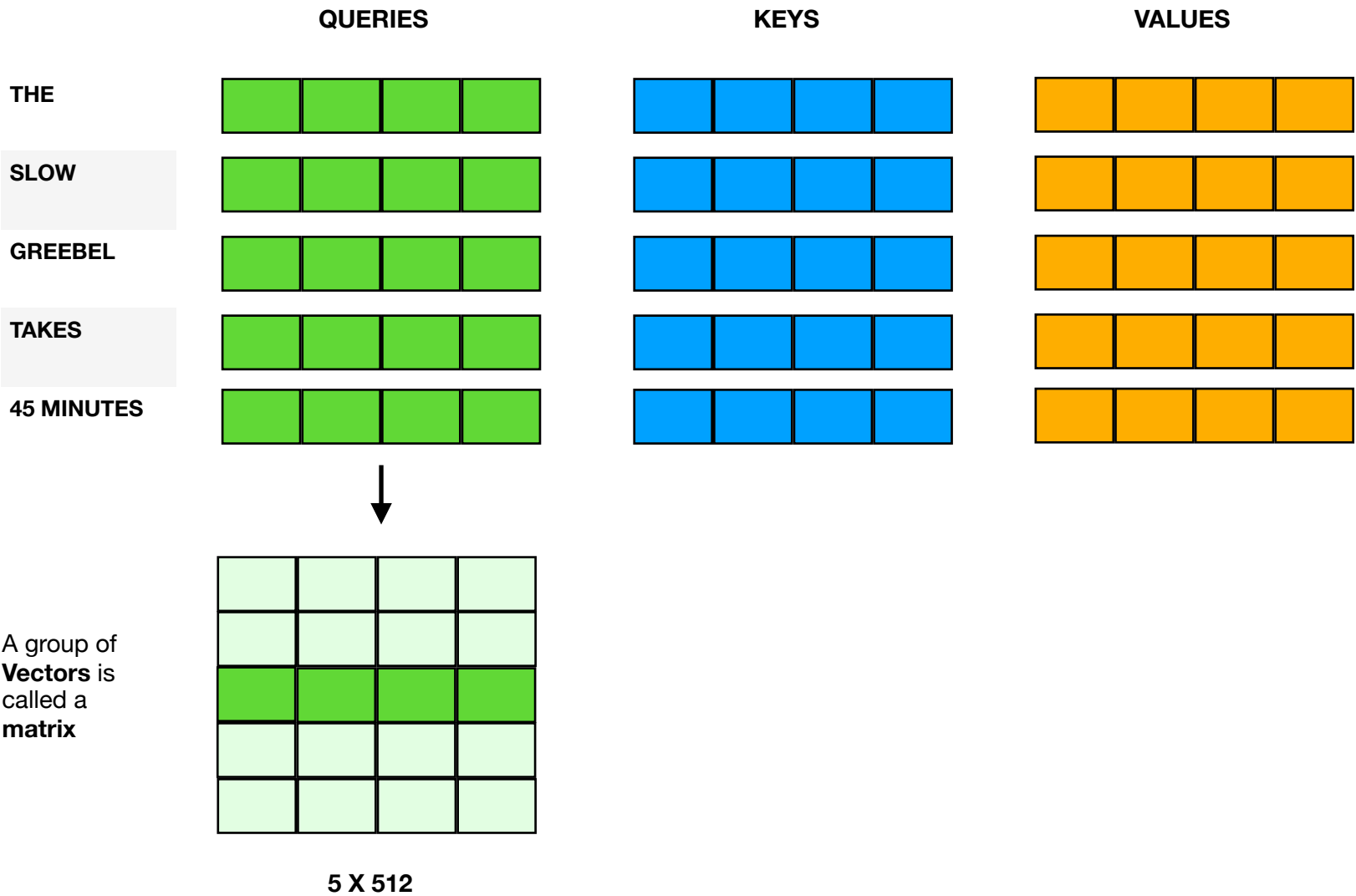
# Scaled Dot Product Attention

There are 512 dimension size for each embedding (word representation), lets say each of the block is **128** dimensions

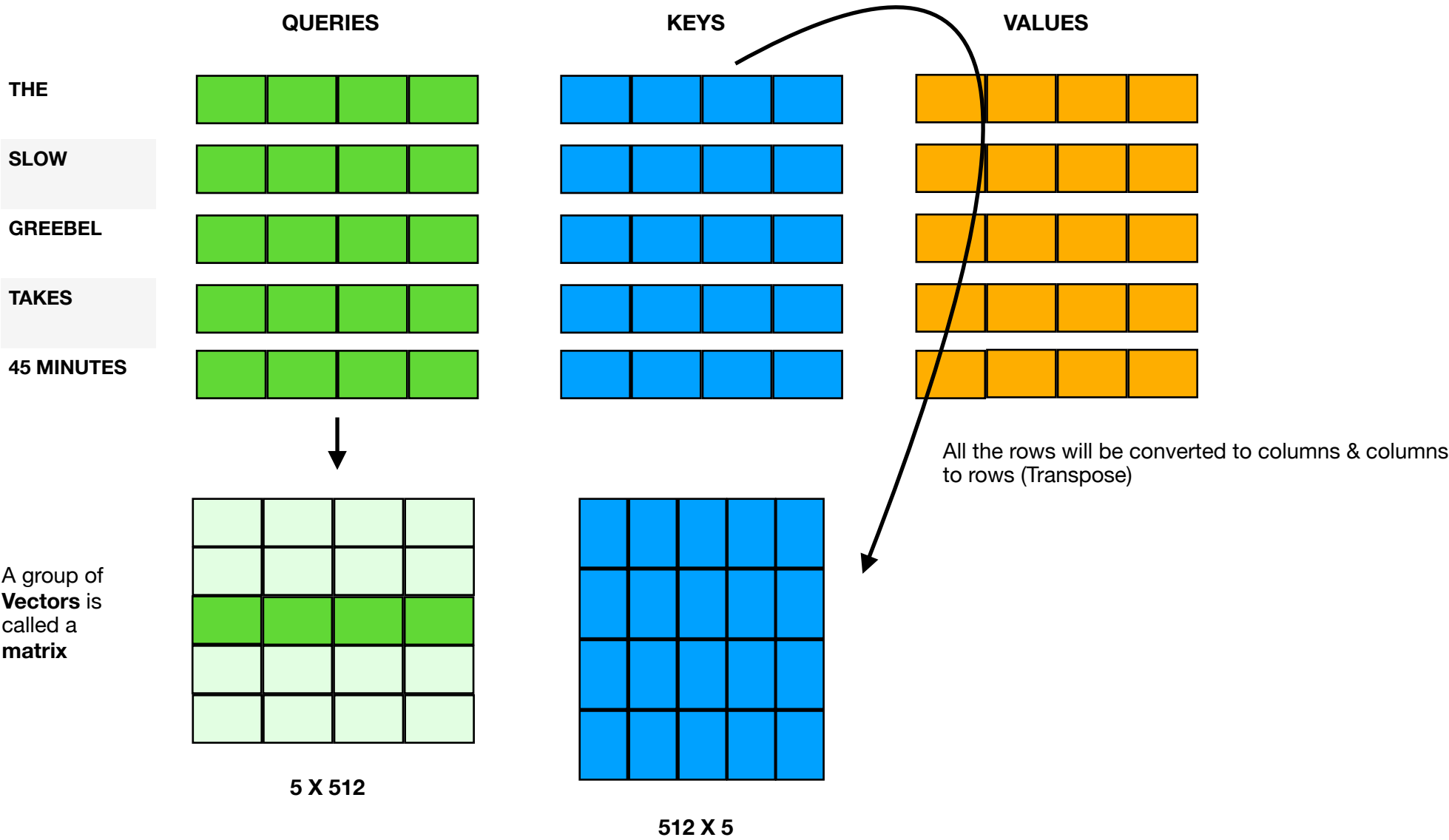
 = 512 dimension



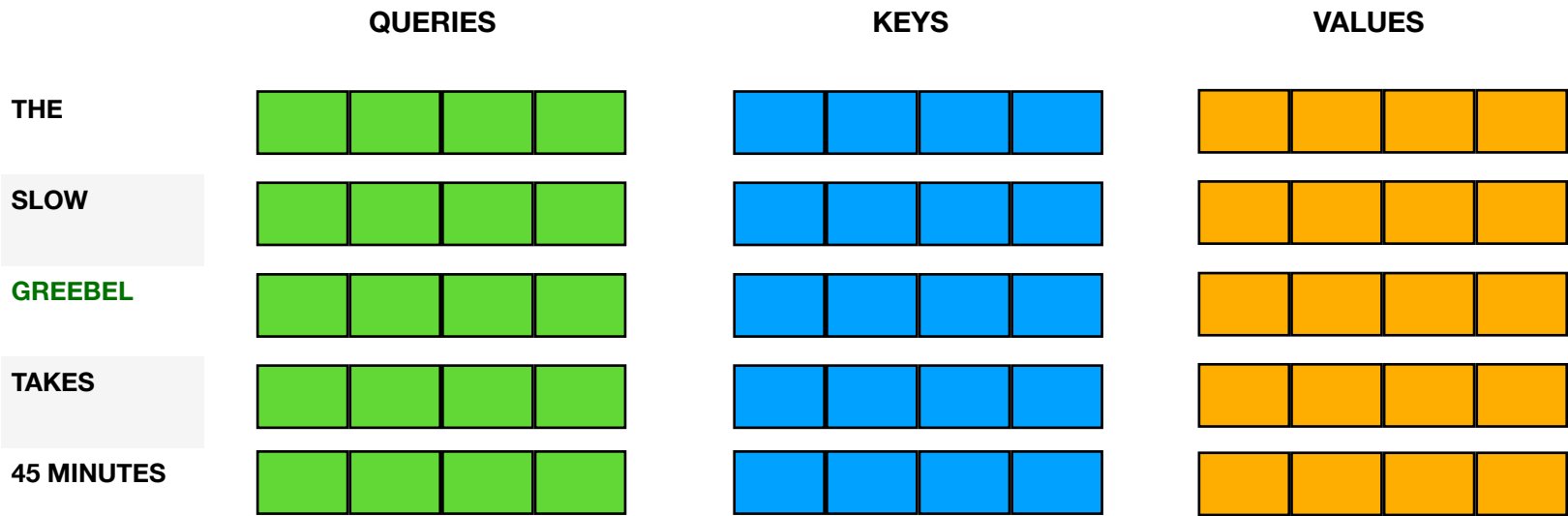
There are 512 dimension size for each embedding (word representation), lets say each of the block is **128** dimensions



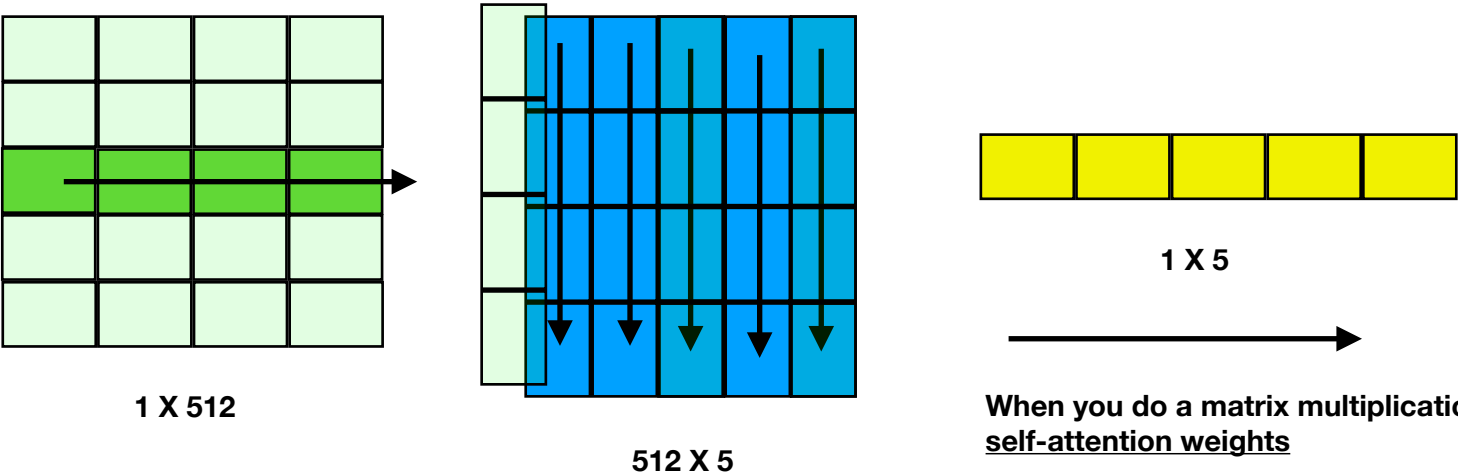
There are 512 dimension size for each embedding (word representation), lets say each of the block is **128** dimensions



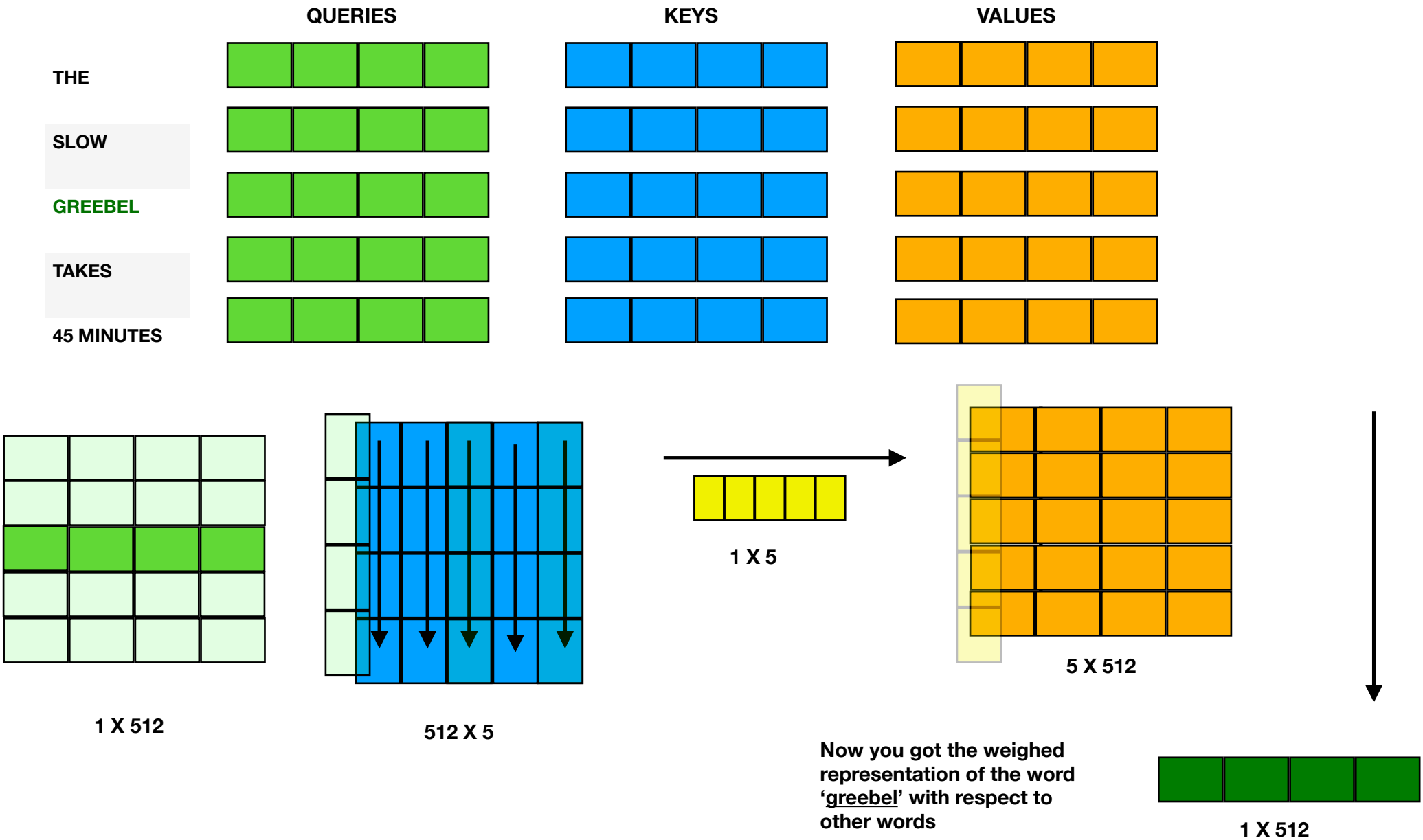
There are 512 dimension size for each embedding (word representation), lets say each of the block is **128** dimensions



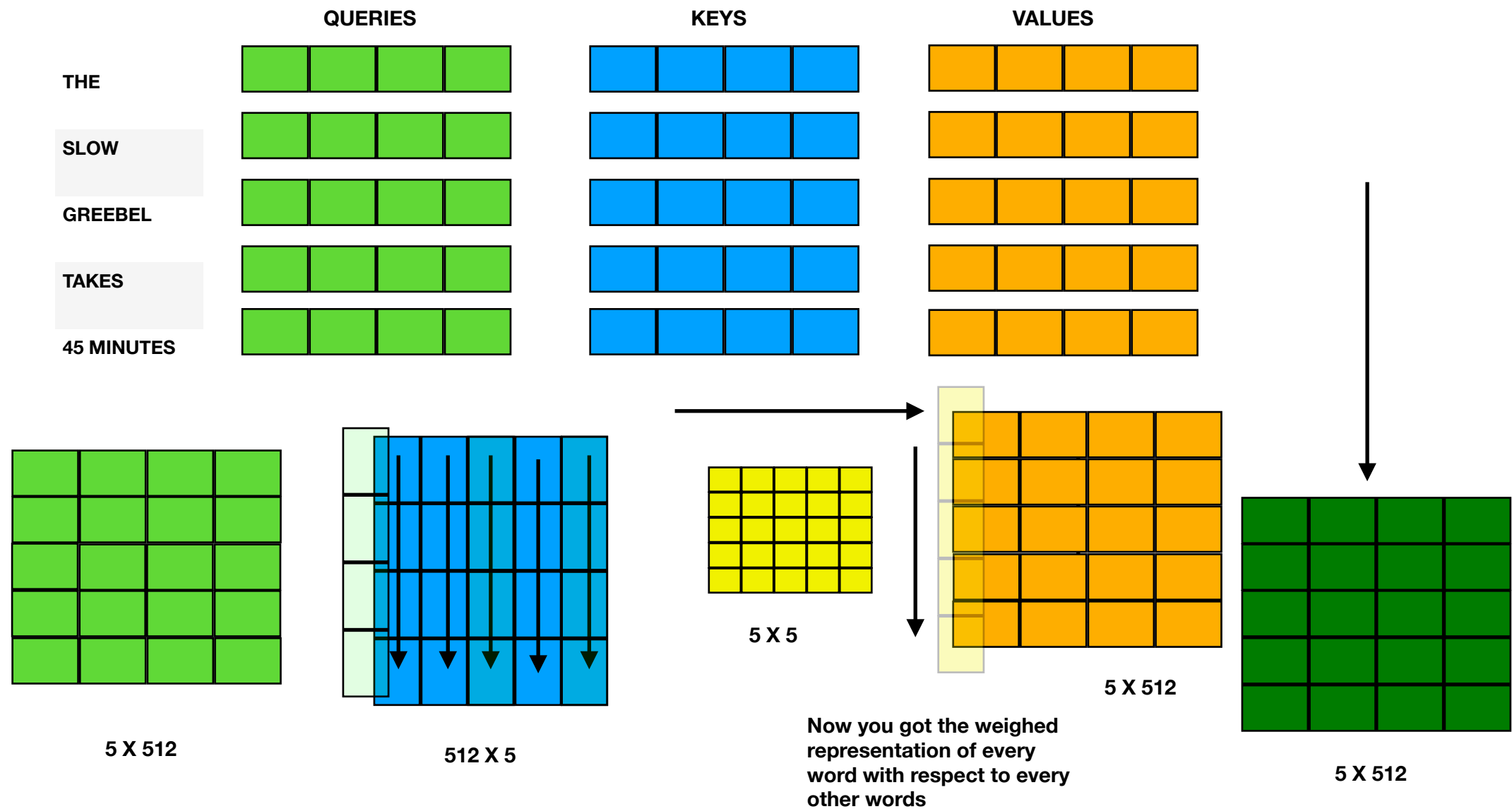
A group of **Vectors** is called a **matrix**



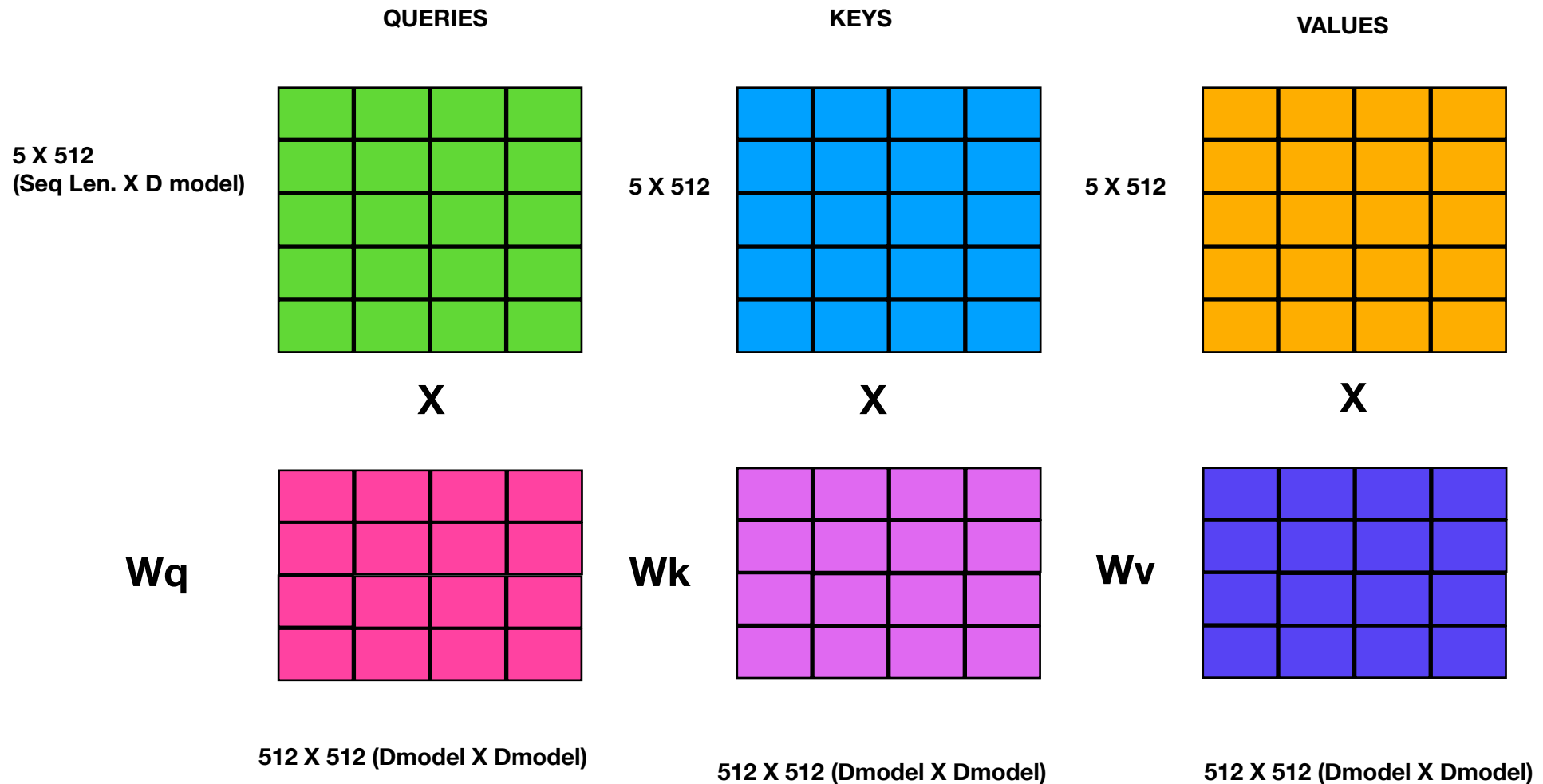
Now you got the weighed representation of the word 'greebel' with respect to other words



Now the Transformers does not know the meaning of **any word**, not just greebel

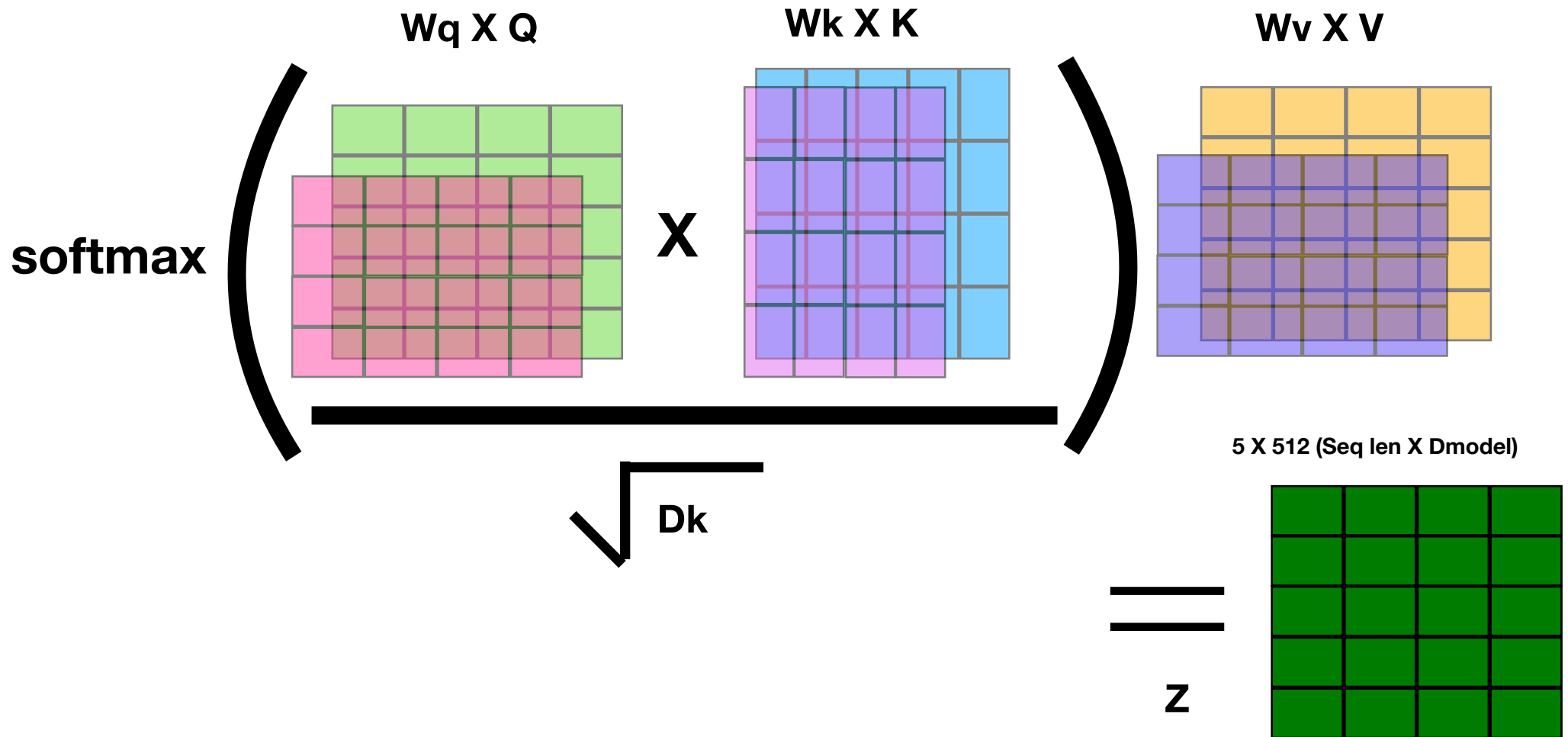


I want to train my transformer model to learn some weights, because I am not just learning the meaning of the words in the previous sentence!



While training these weights are adjusted/learned so as to get the meaning of words or create contextualised word embeddings

I want to train my transformer model to learn some weights, because I am not just learning the meaning of the words in the previous sentence!

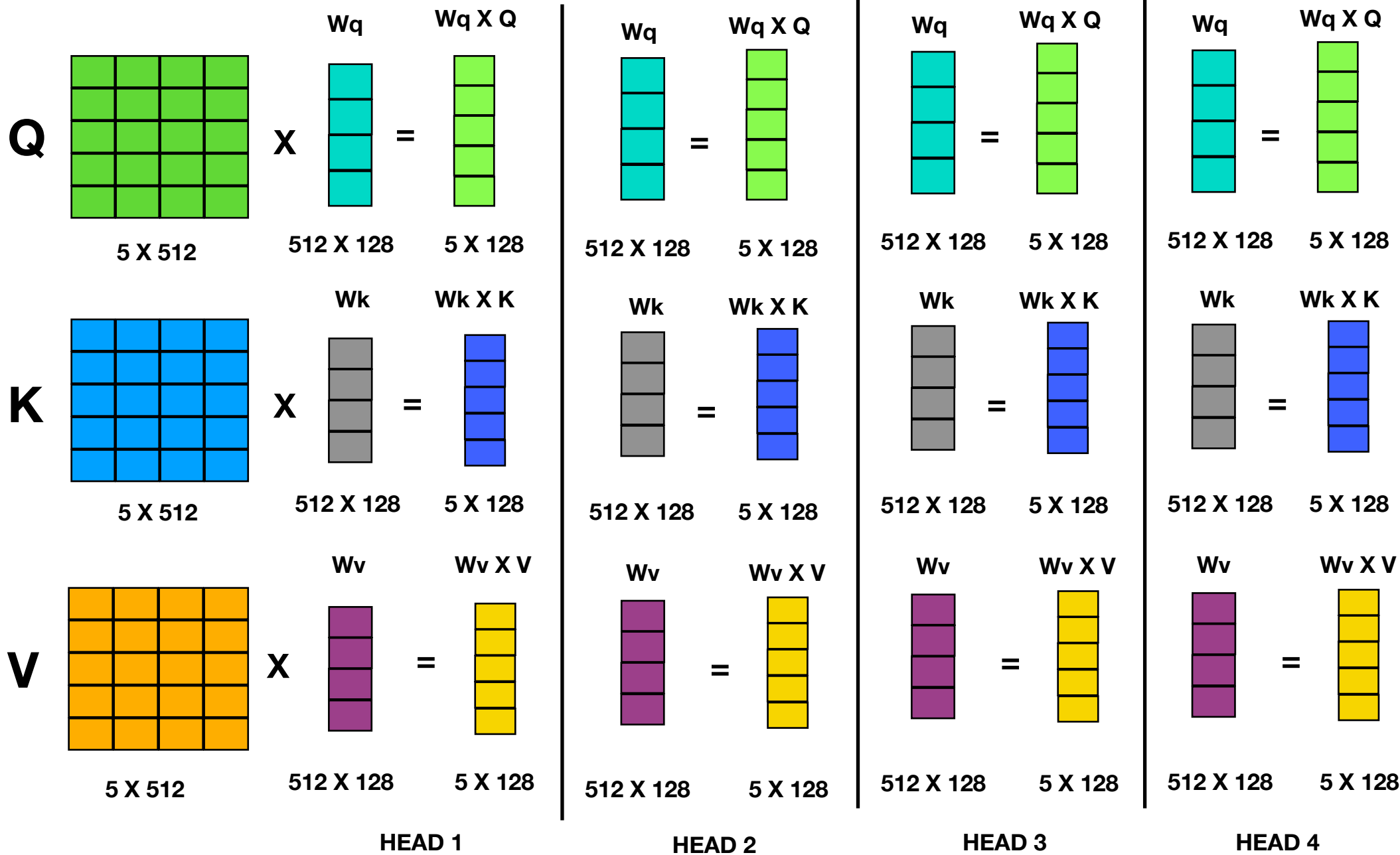


For every feed-forward step the above process is carried out multiple times to get a deep contextualised embeddings

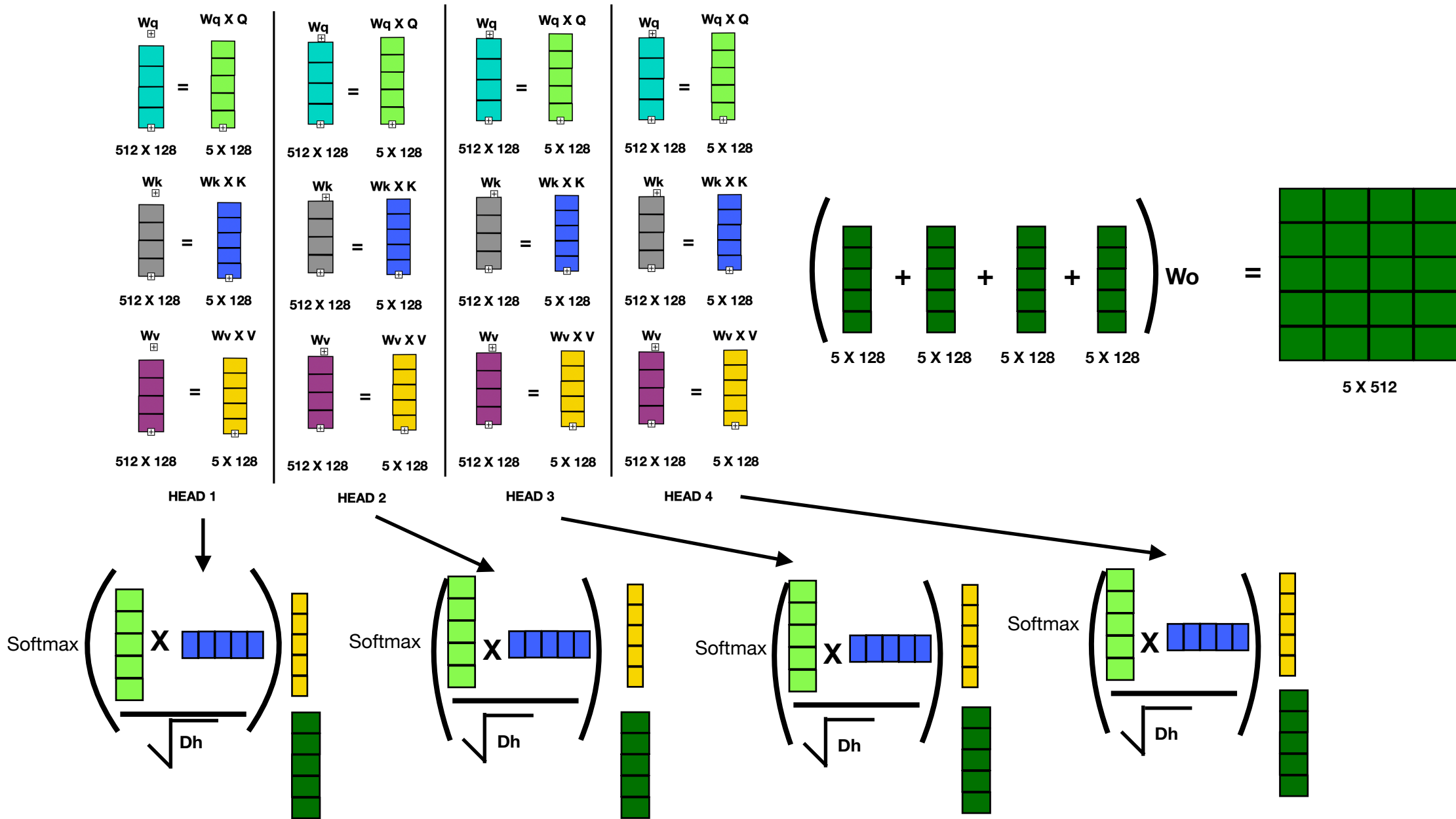
In scaled dot-product attention, the dot products are scaled by the size of the embedding vectors so that we don't get too many large numbers during training that can cause the softmax to saturate.



# Multi Head Attention

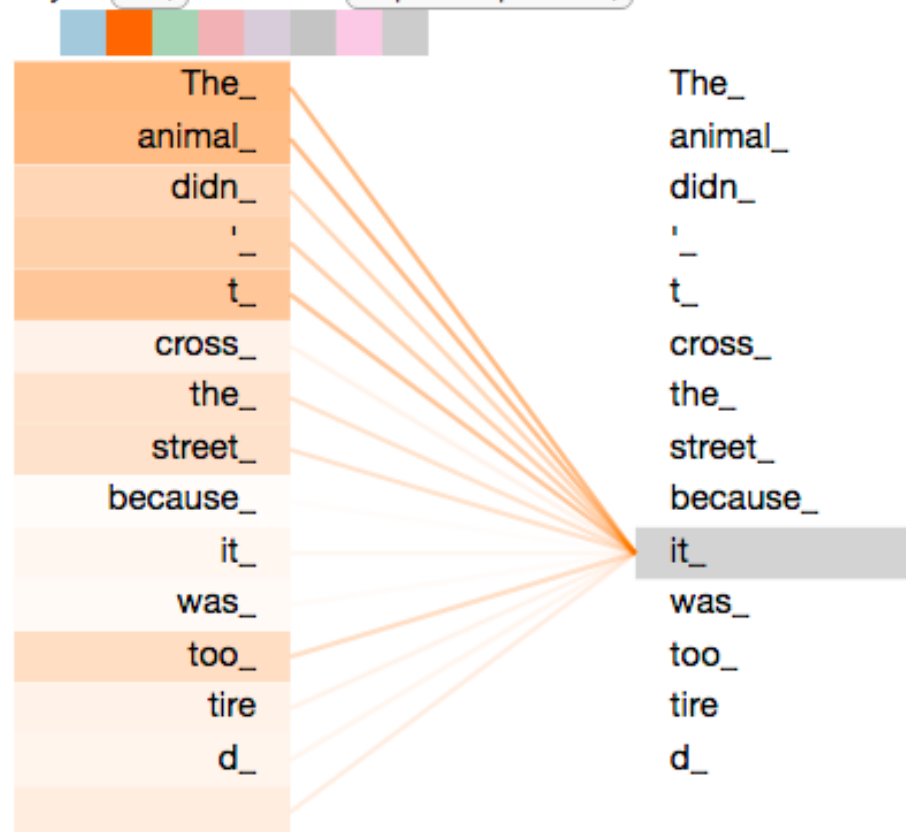


# Multi Head Attention

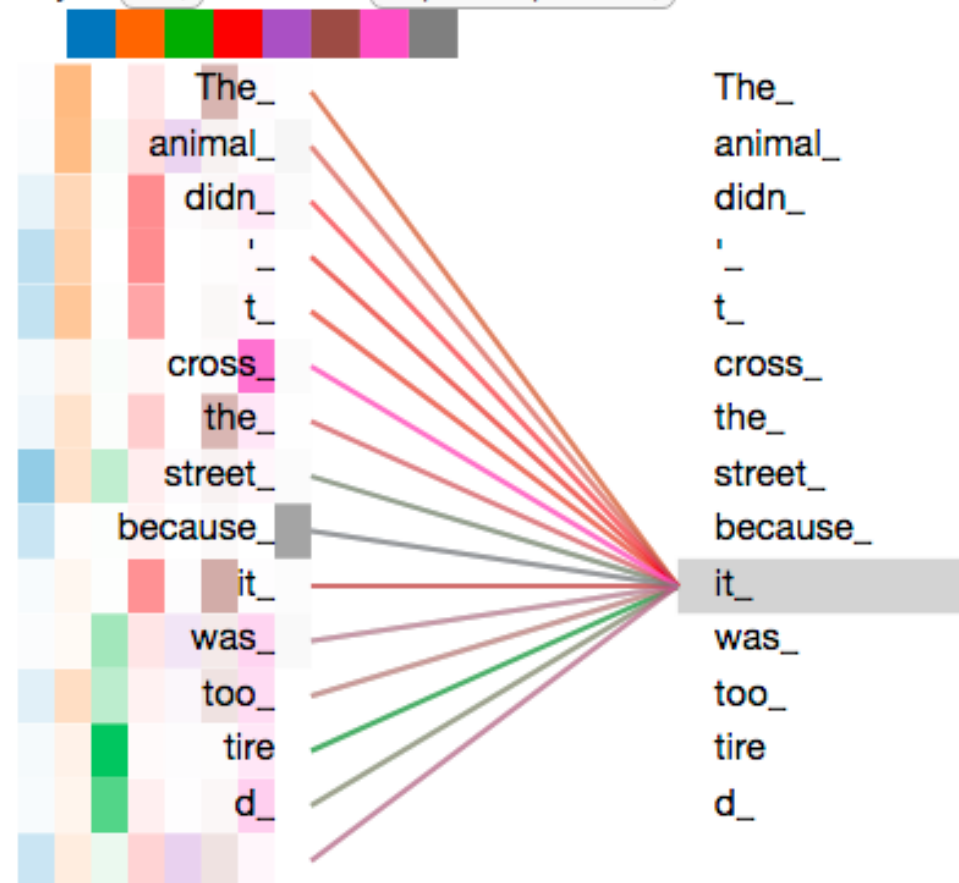


# Attention Visualised

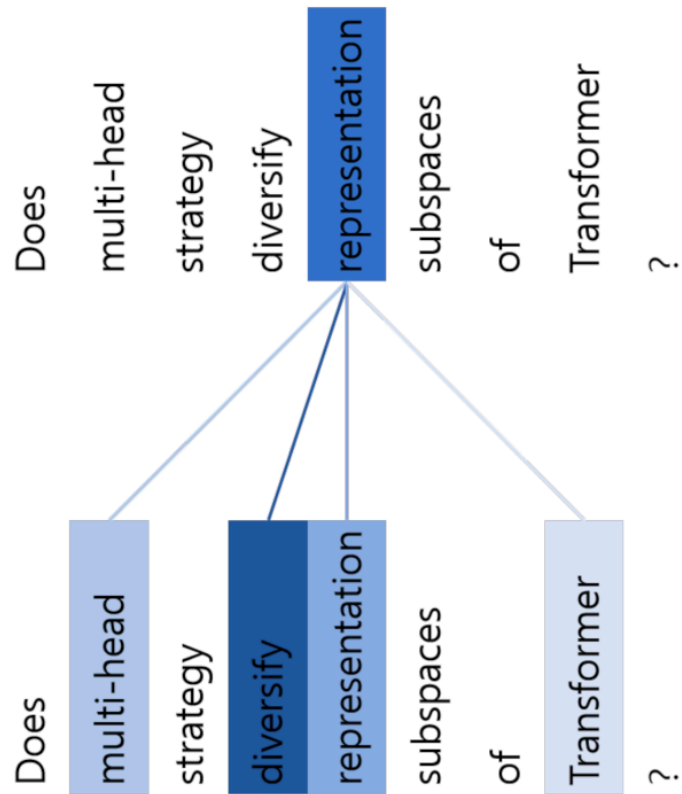
Layer: 5 Attention: Input - Input



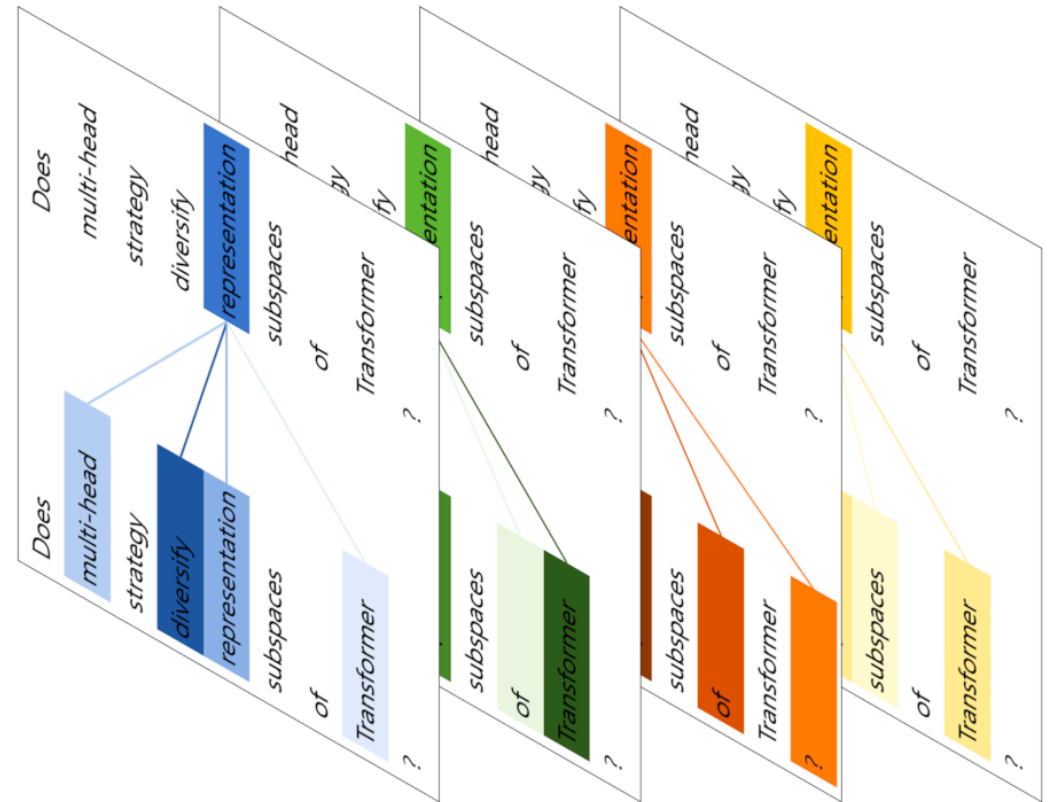
Layer: 5 Attention: Input - Input



# Attention Visualised

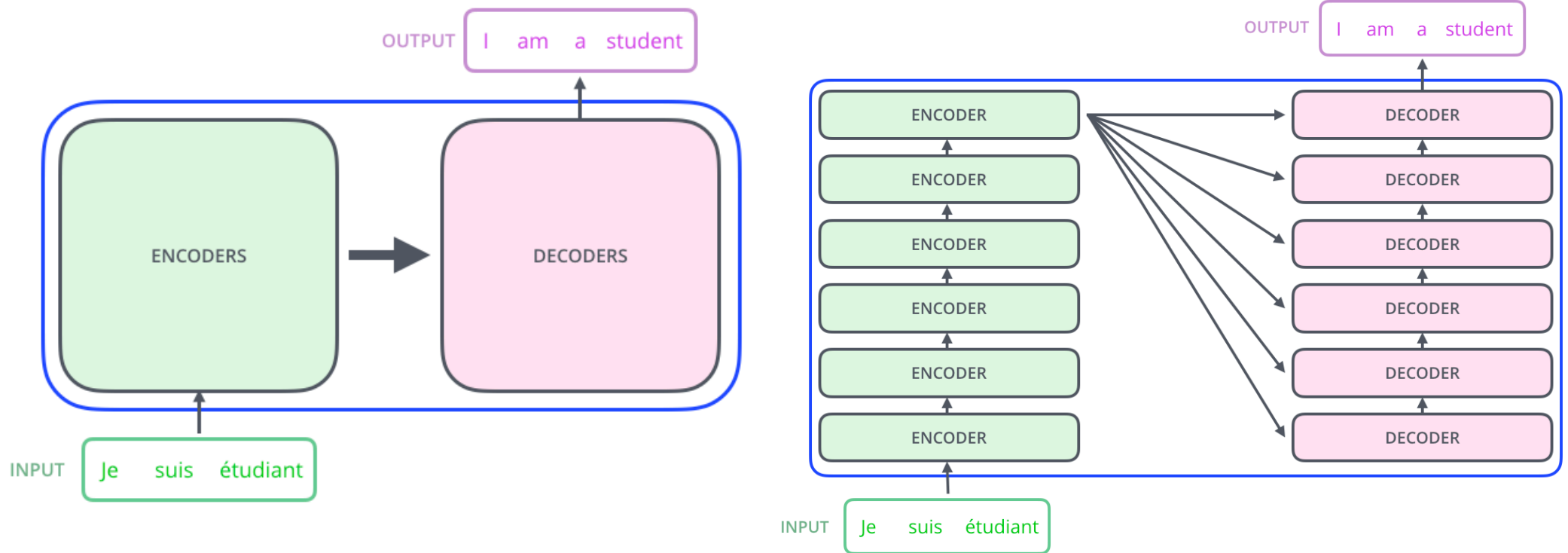


(a) Single-head attention



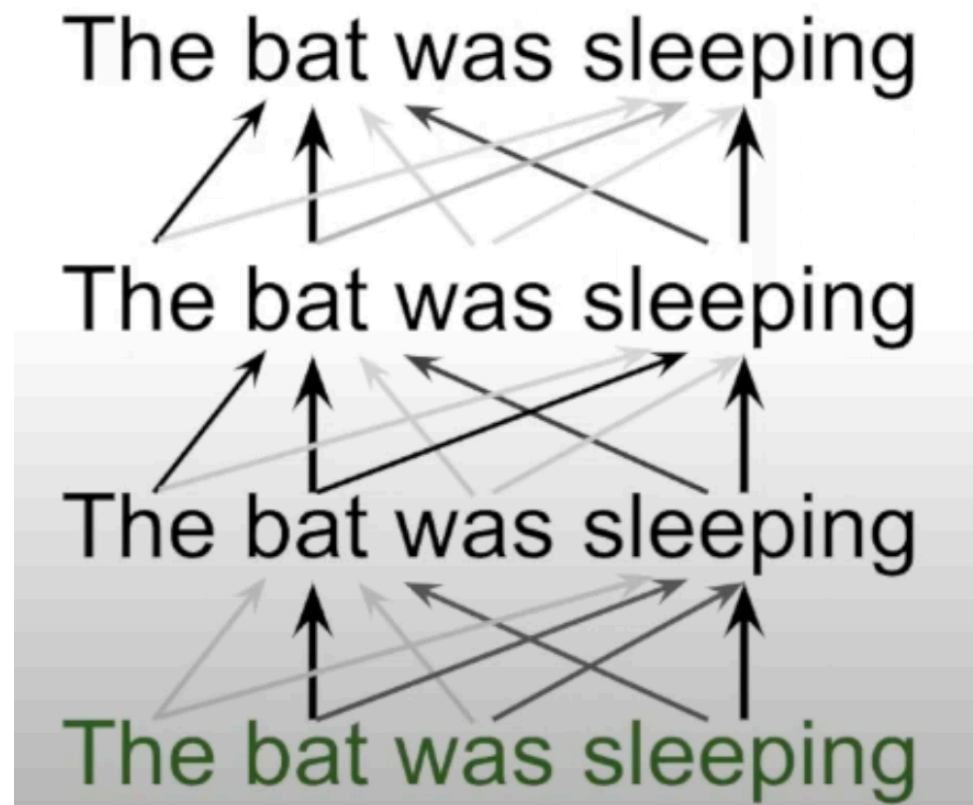
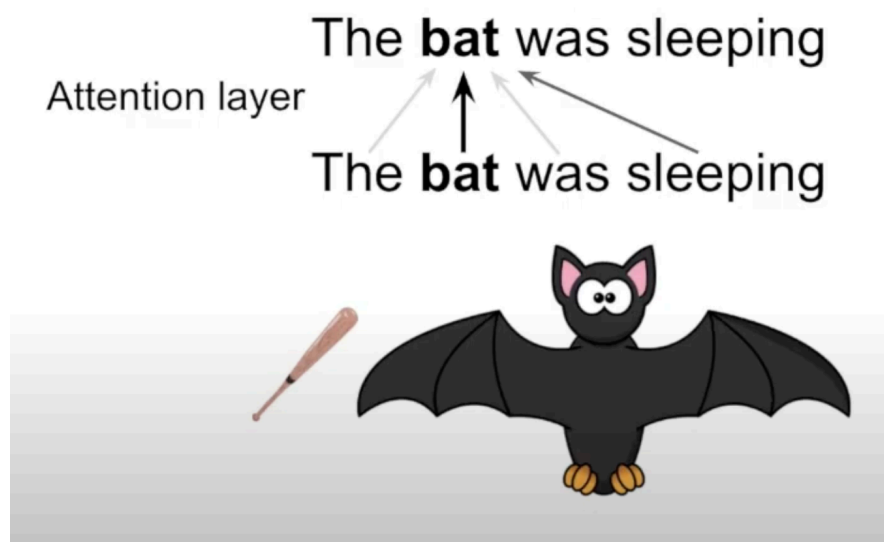
(b) Multi-head attention

# Encoder Decoder: Where this Multi-head attention fits in?



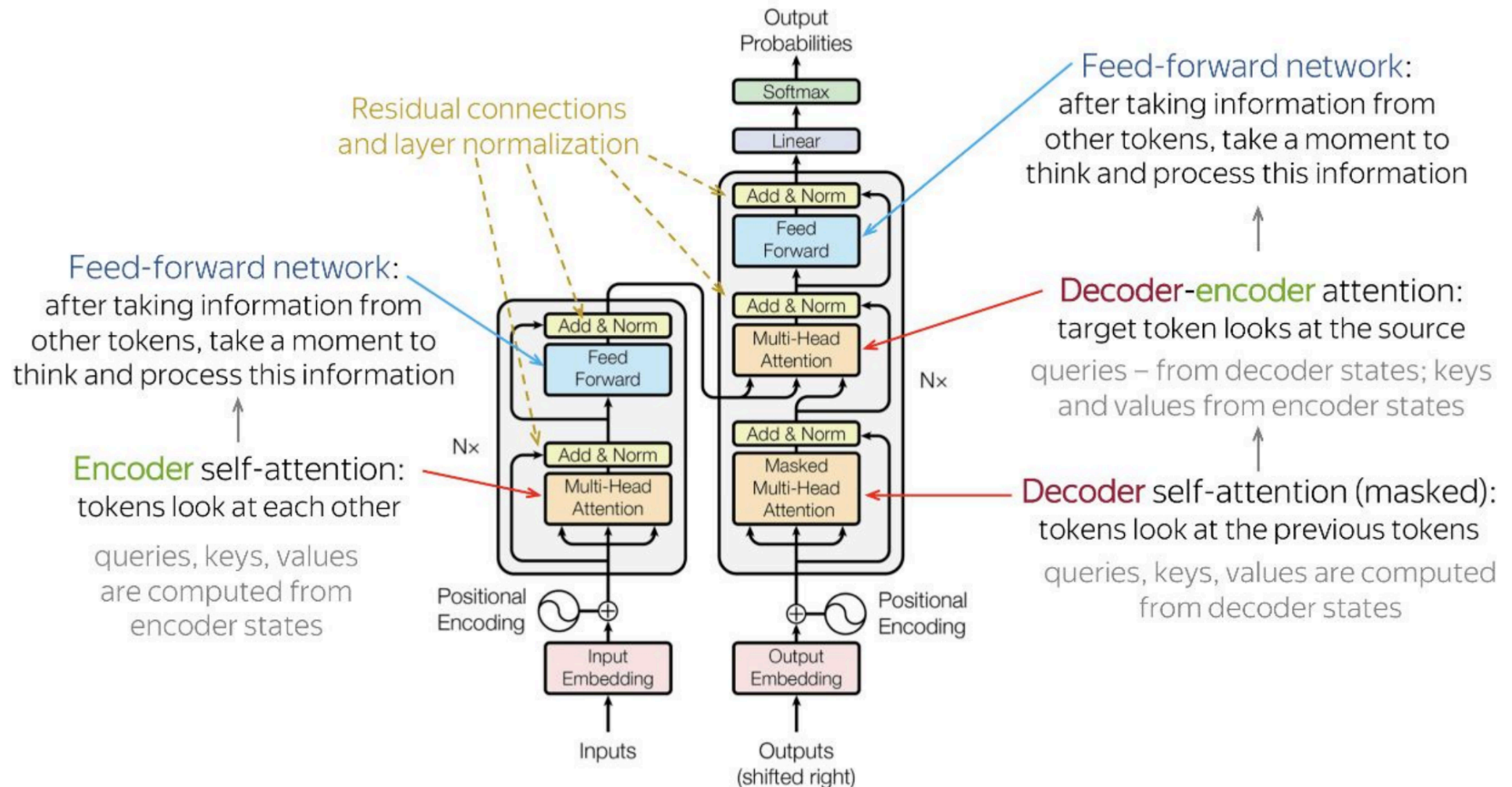
**Note:** The input dimension is same as the output dimension always (batch size, seq length, D model)

# Encoder Decoder: Where this Multi-head attention fits in?



**Note:** The input dimension is same as the output dimension always (batch size, seq length, D model)

# What are the missing pieces?



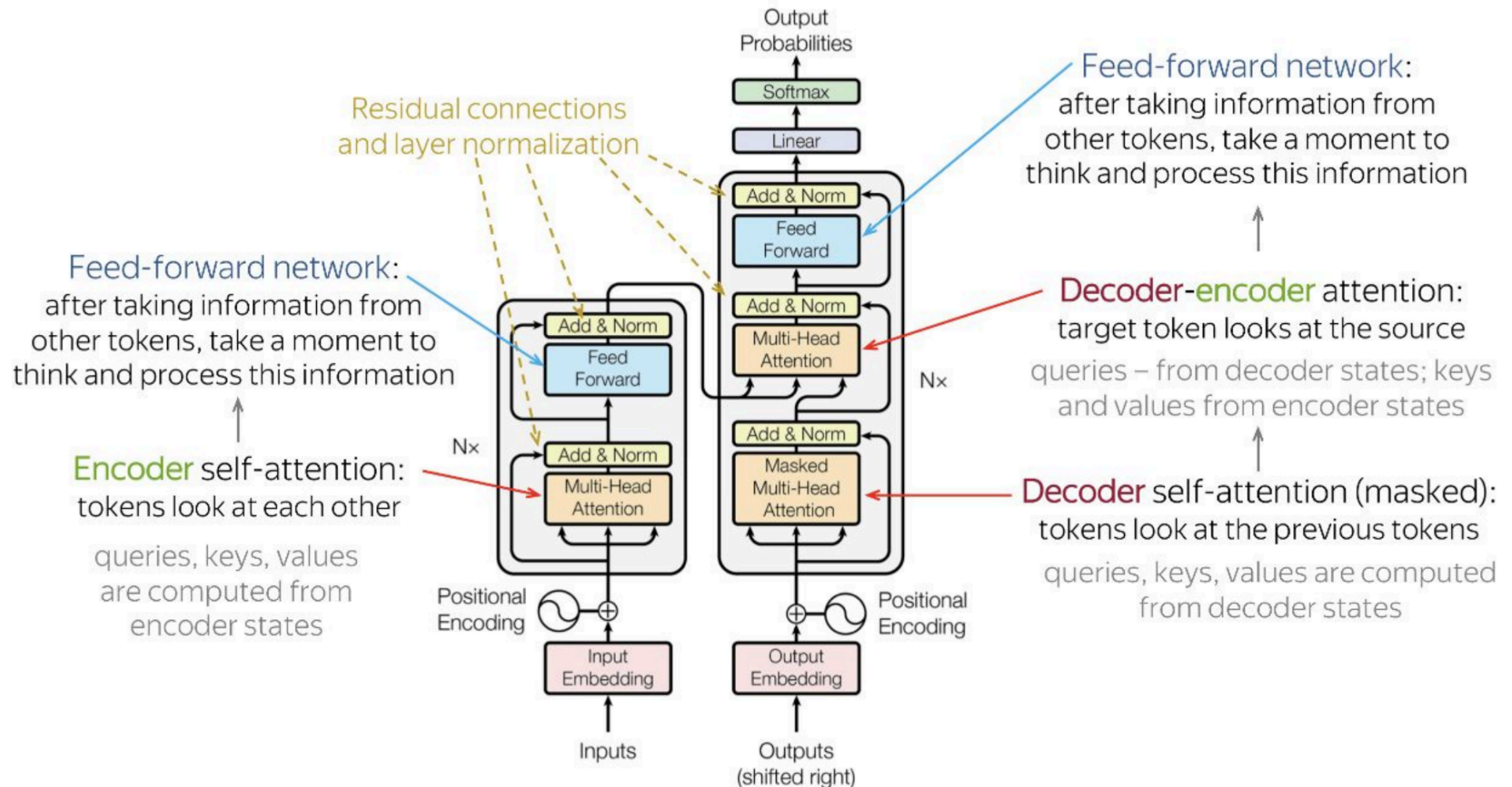
# Appendix 1: Positional Embeddings

## What is positional encoding?

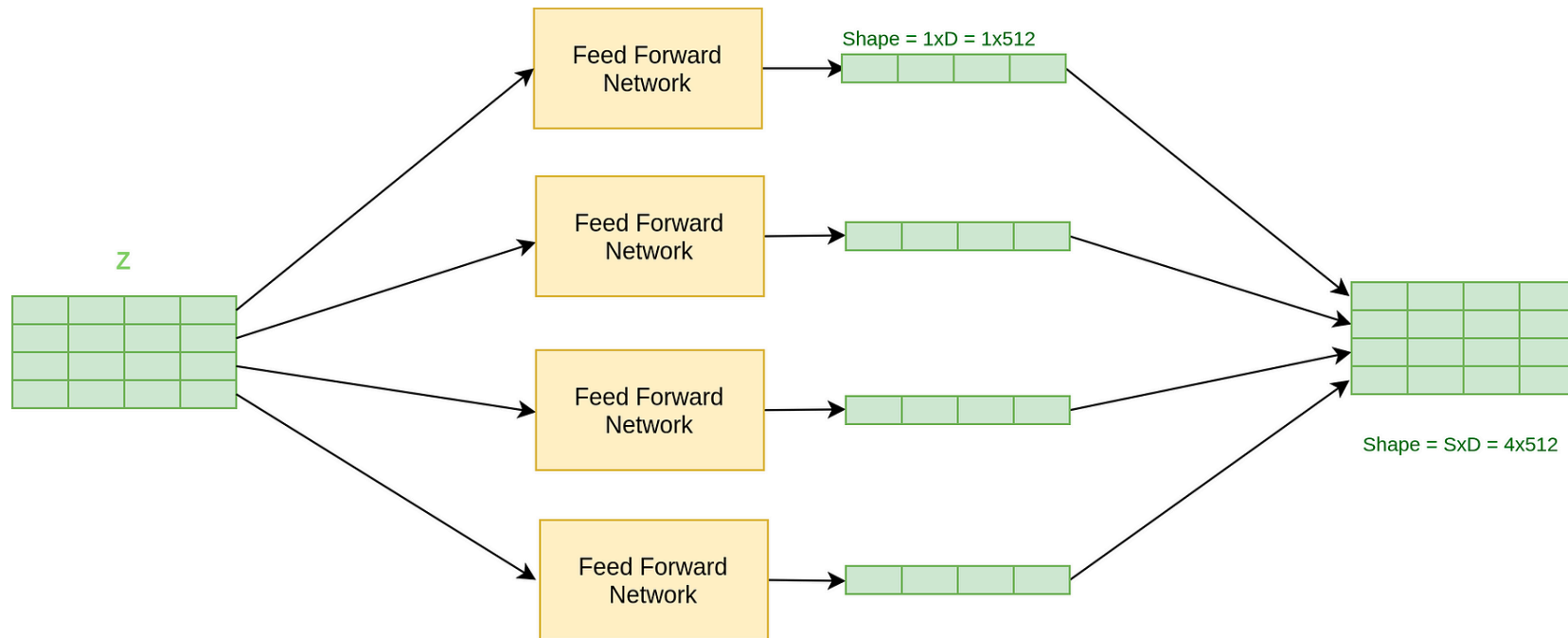
Original sentence	YOUR	CAT	IS	A	LOVELY	CAT
<b>Embedding</b> (vector of size 512)	952.207	171.411	621.659	776.562	6422.693	171.411
	5450.840	3276.350	1304.051	5567.288	6315.080	3276.350
	1853.448	9192.819	0.565	58.942	9358.778	9192.819
	...	...	...	...	...	...
	1.658	3633.421	7679.805	2716.194	2141.081	3633.421
	2671.529	8390.473	4506.025	5119.949	735.147	8390.473
<b>Position Embedding</b> (vector of size 512). Only computed once and reused for every sentence during training and inference.	+	+	+	+	+	+
	...	1664.068	...	...	...	1281.458
	...	8080.133	...	...	...	7902.890
	...	2620.399	...	...	...	912.970
	...	...	...	...	...	3821.102
	...	9386.405	...	...	...	1659.217
<b>Encoder Input</b> (vector of size 512)	=	=	=	=	=	=
	...	1835.479	...	...	...	1452.869
	...	11356.483	...	...	...	11179.24
	...	11813.218	...	...	...	10105.789
	...	...	...	...	...	...
	...	13019.826	...	...	...	5292.638
	...	11510.632	...	...	...	15409.093



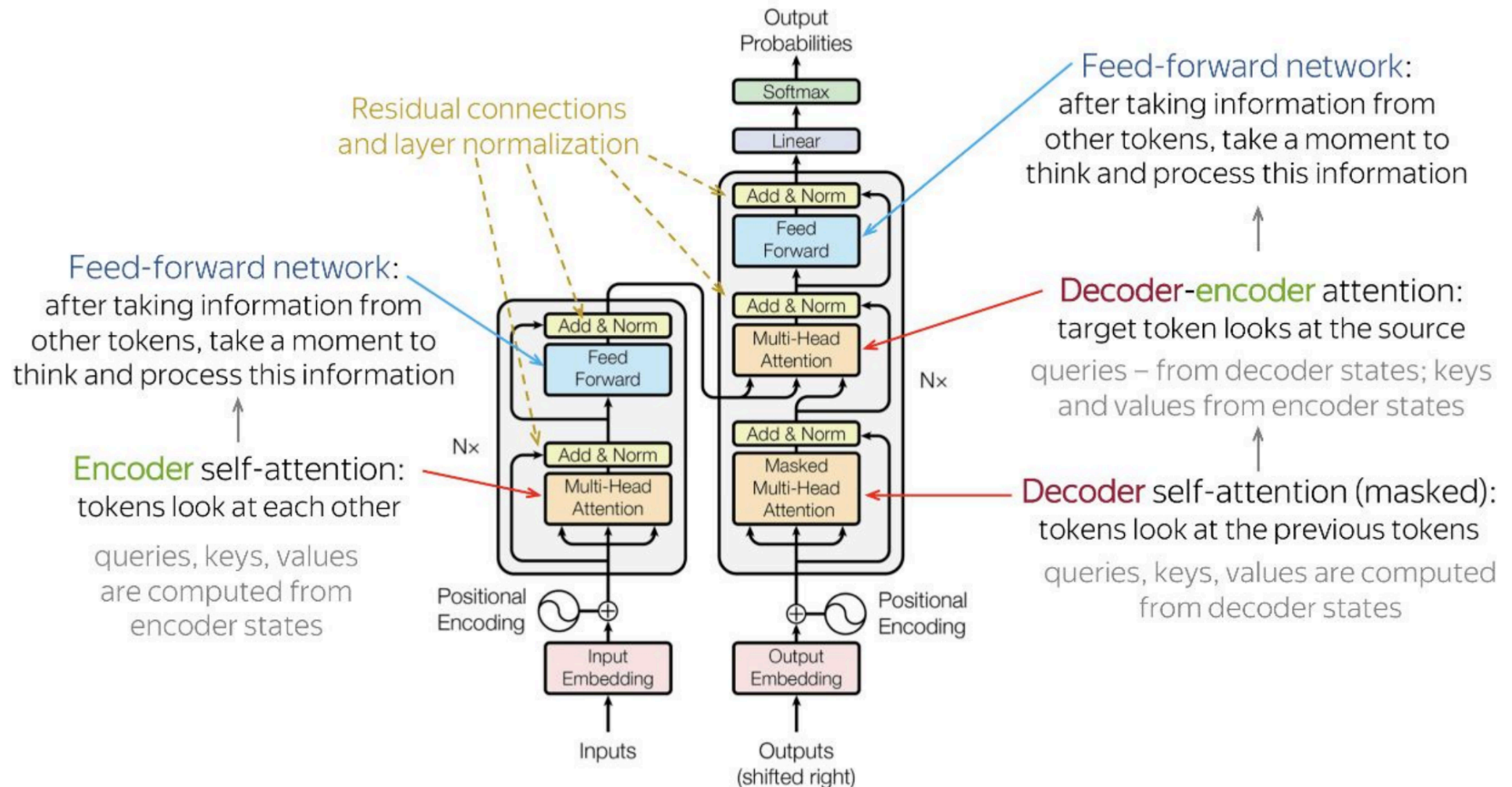
# What are the missing pieces?



## Appendix 2: Position-wise Feed-Forward Networks



# What are the missing pieces?



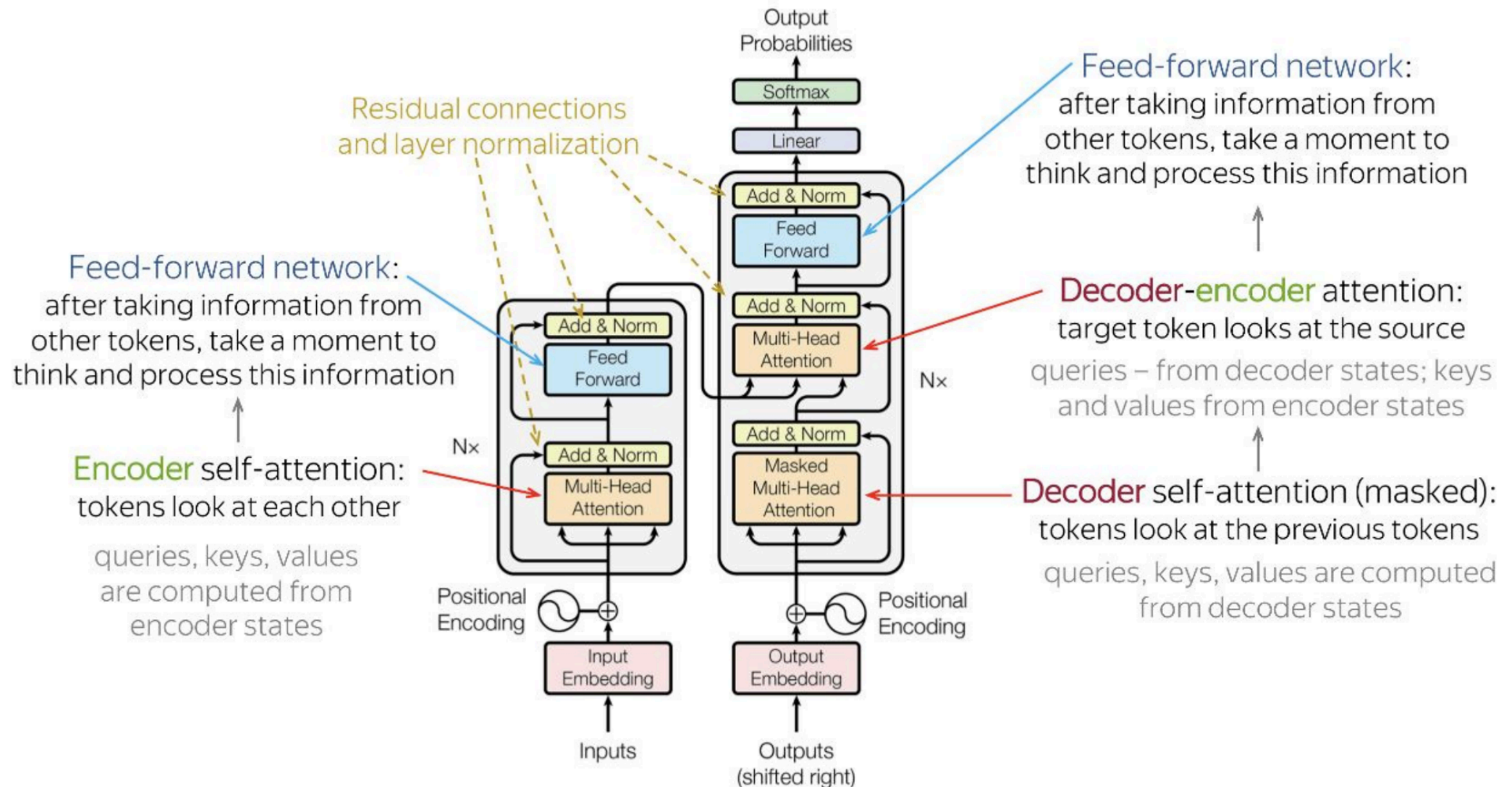
## Appendix 3: Position-wise Feed-Forward Networks

Our goal is to make the model causal: it means the output at a certain position can only depend on the words on the previous positions. The model **must not** be able to see future words.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

All the values above the diagonal are replace with  $-\infty$  before applying the softmax,which will replace them with zero.

# What are the missing pieces?



# Appendix 4: Encoder Decoder Attention

