# The Illustrated Transformer

Jay Alammar

Visualizing machine learning one concept at a time.

@JayAlammar on Twitter. YouTube Channel

## Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering & Artificial Intelligence

## References

Main

- https://jalammar.github.io/illustrated-transformer/

## Illustrated Transformer

### Transformer

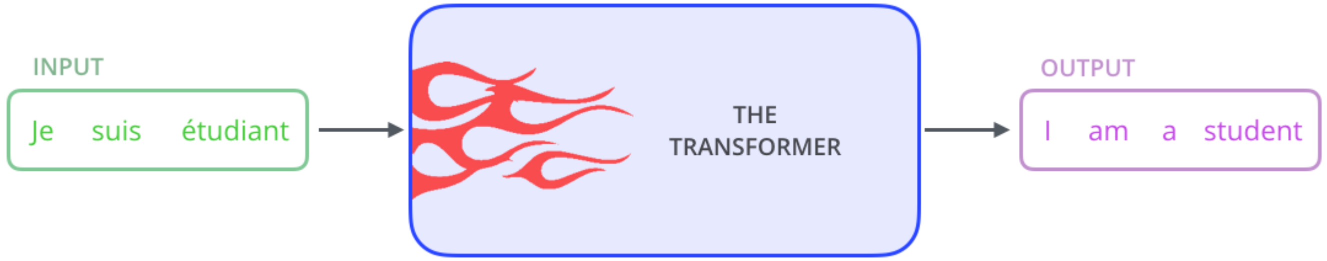
- A model that uses **attention** to boost the speed with which these models can be trained.
- The Transformers outperforms the Google Neural Machine Translation model in specific tasks.
- The biggest benefit, however, comes from how The Transformer lends itself to **parallelization**. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering.

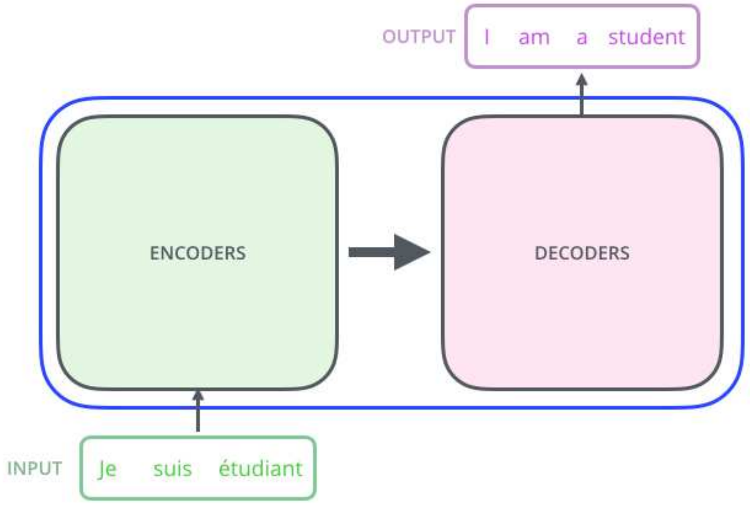
- A TensorFlow implementation of it is available as a part of the Tensor2Tensor package.
- Harvard's NLP group created a guide annotating the paper with PyTorch implementation.

### A High-Level Look

## Illustrated Transformer

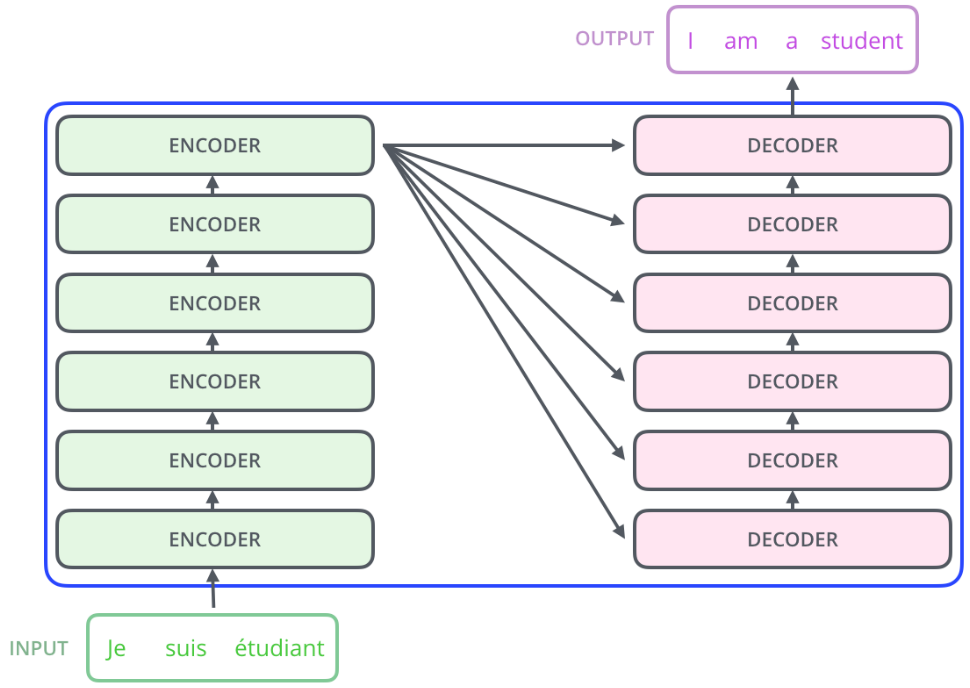
### A High-Level Look

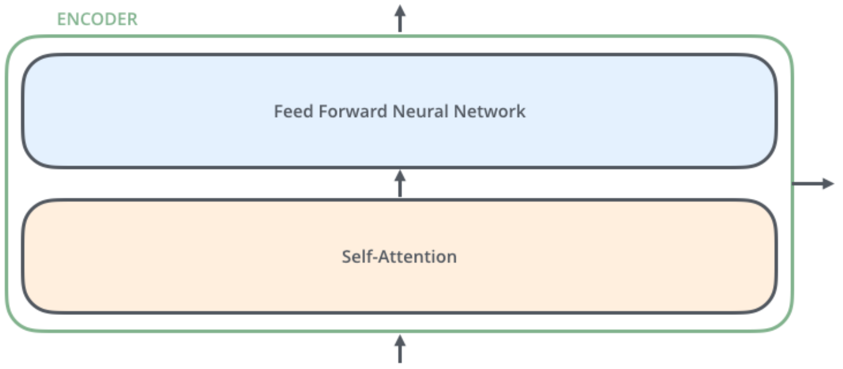


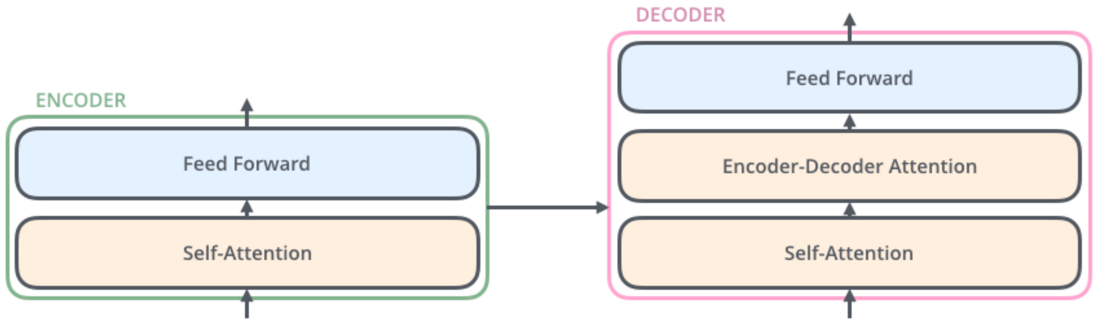


## Illustrated Transformer

### A High-Level Look

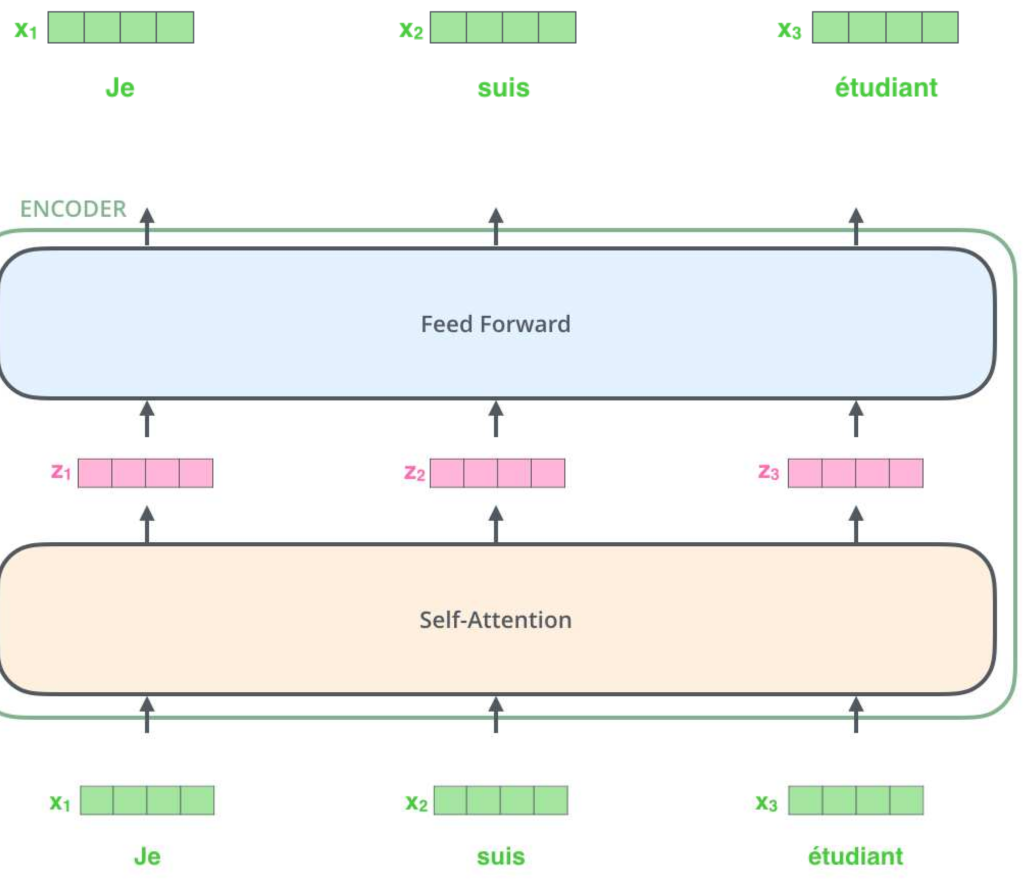




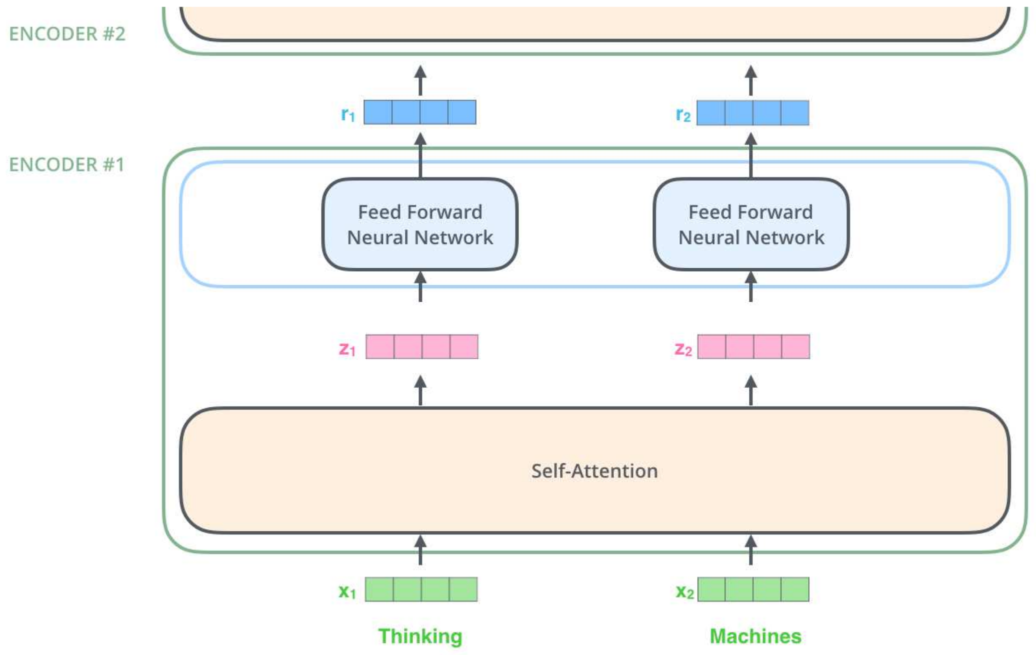


## Illustrated Transformer

### Bringing The Tensors Into The Picture

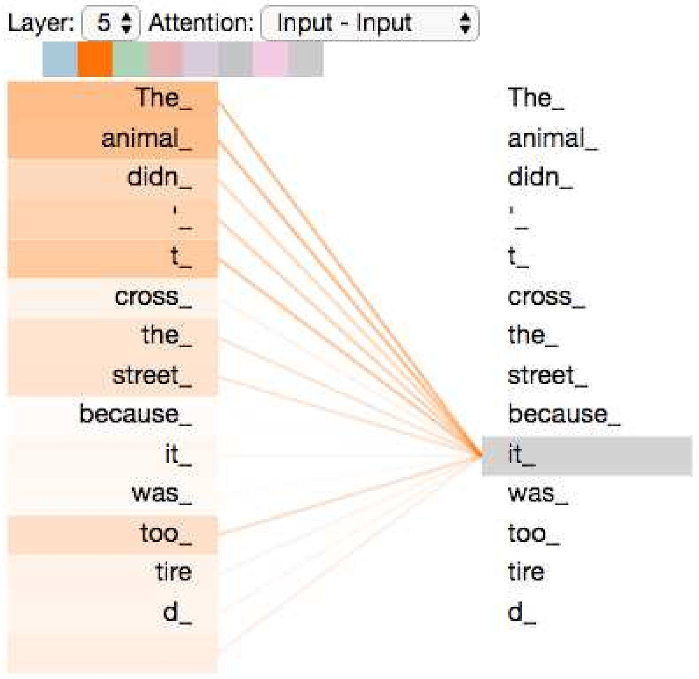


### Bringing The Tensors Into The Picture

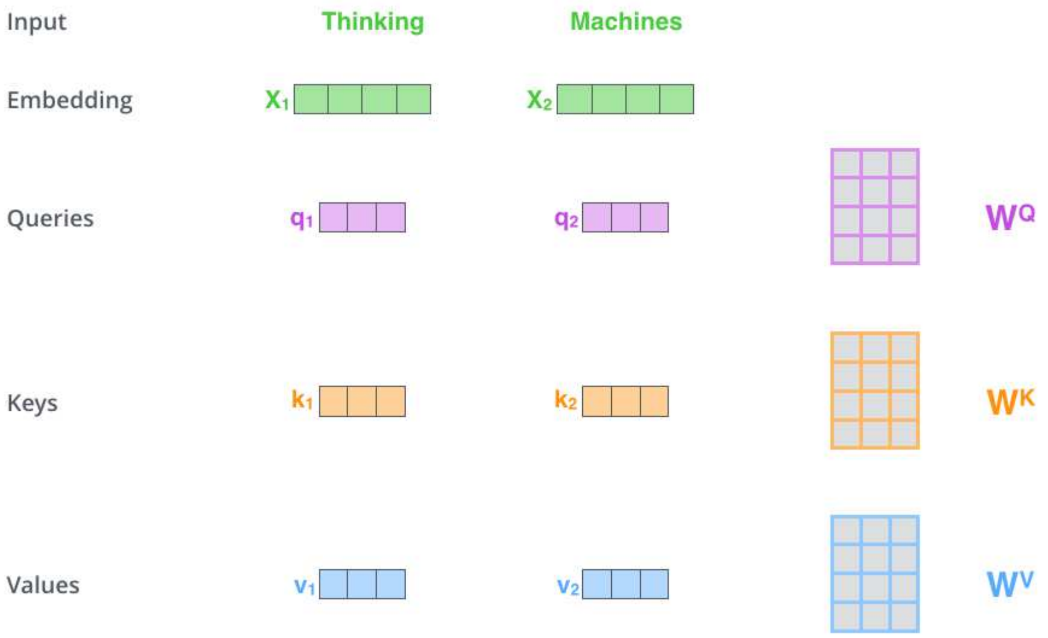


## Illustrated Transformer

### Self-Attention at a High Level

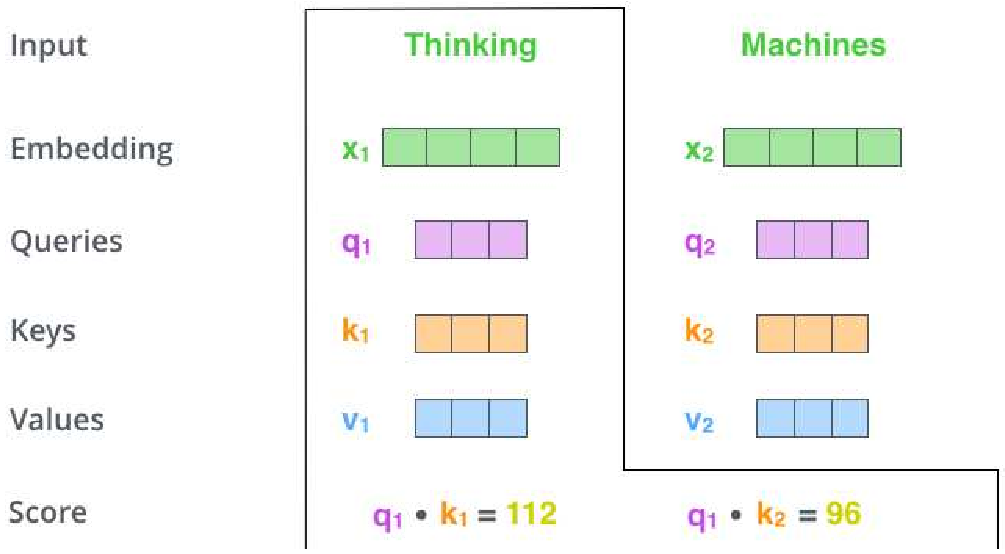


### Self-Attention in Detail

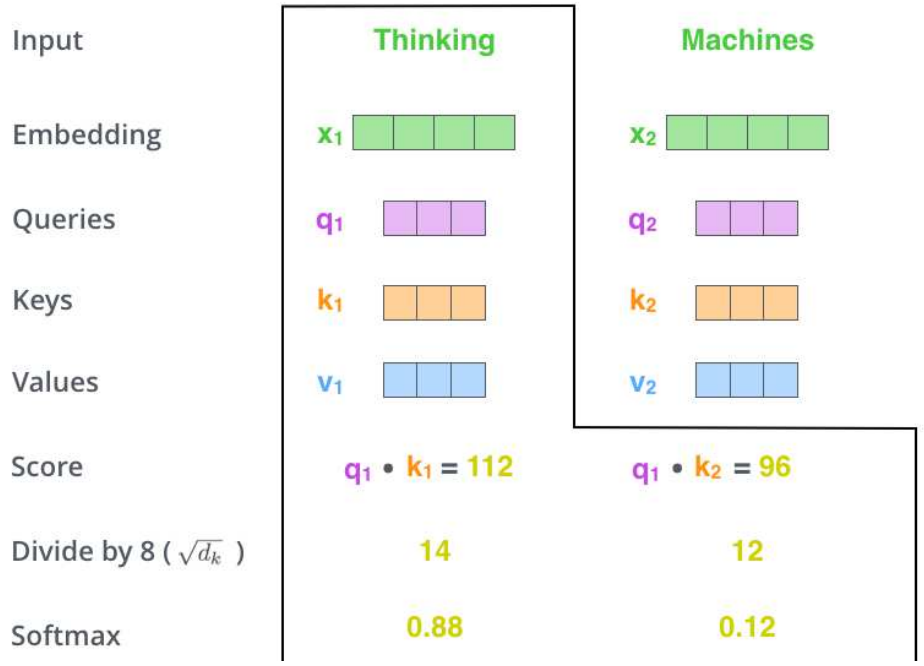


## Illustrated Transformer

### Self-Attention in Detail

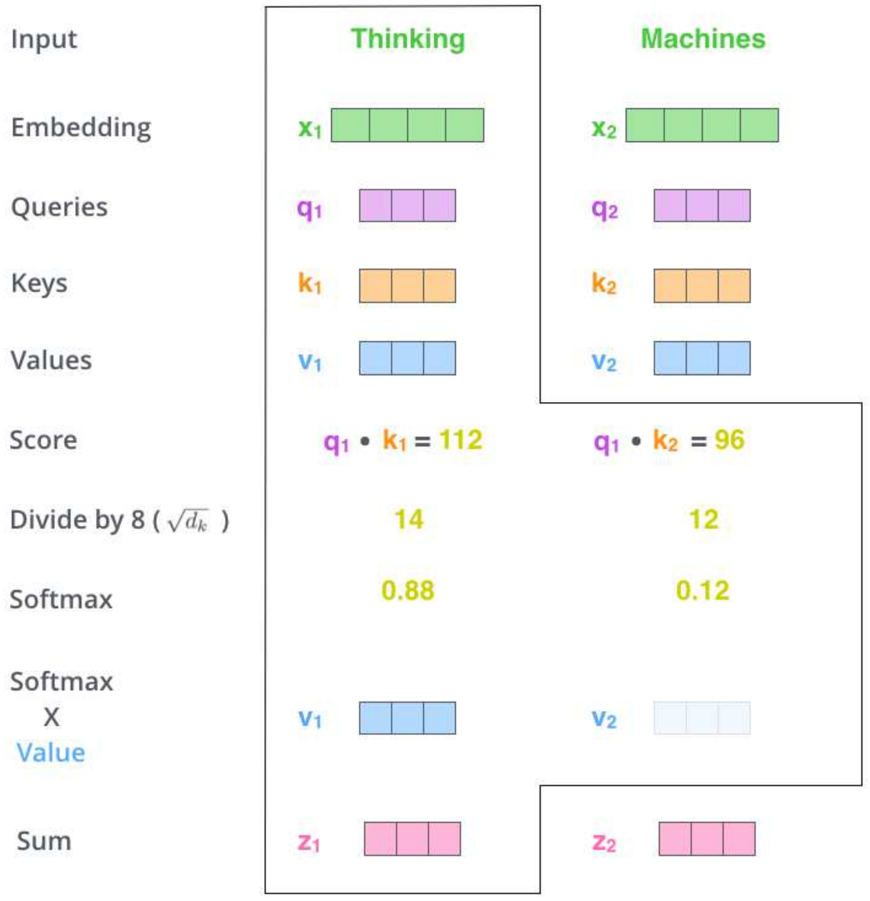


### Self-Attention in Detail

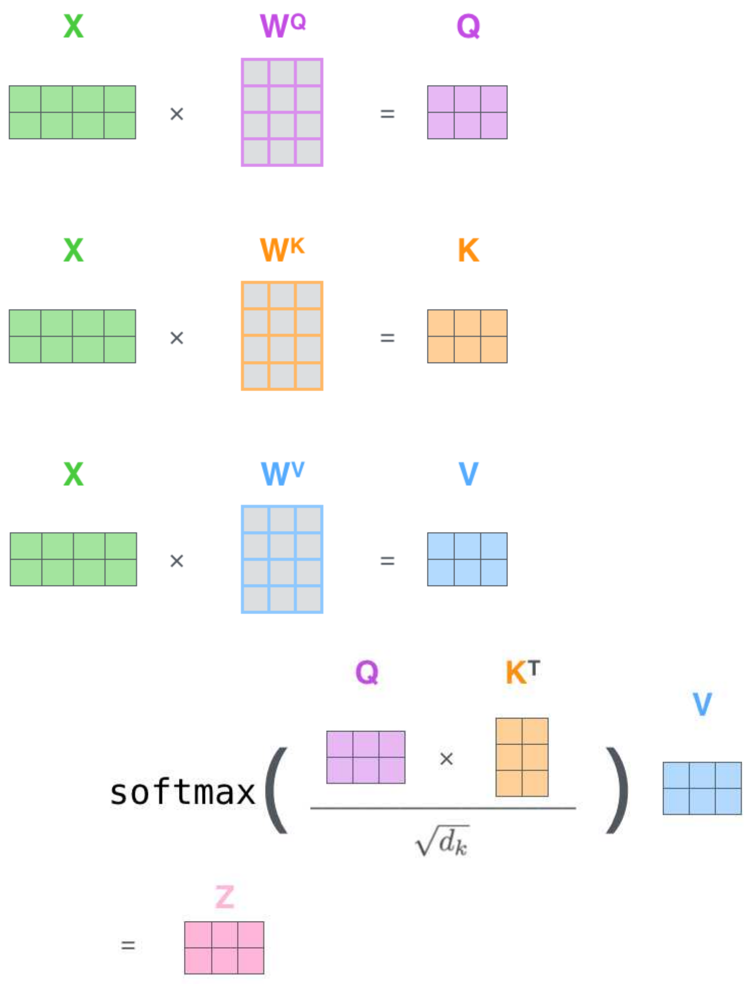


## Illustrated Transformer

### Self-Attention in Detail



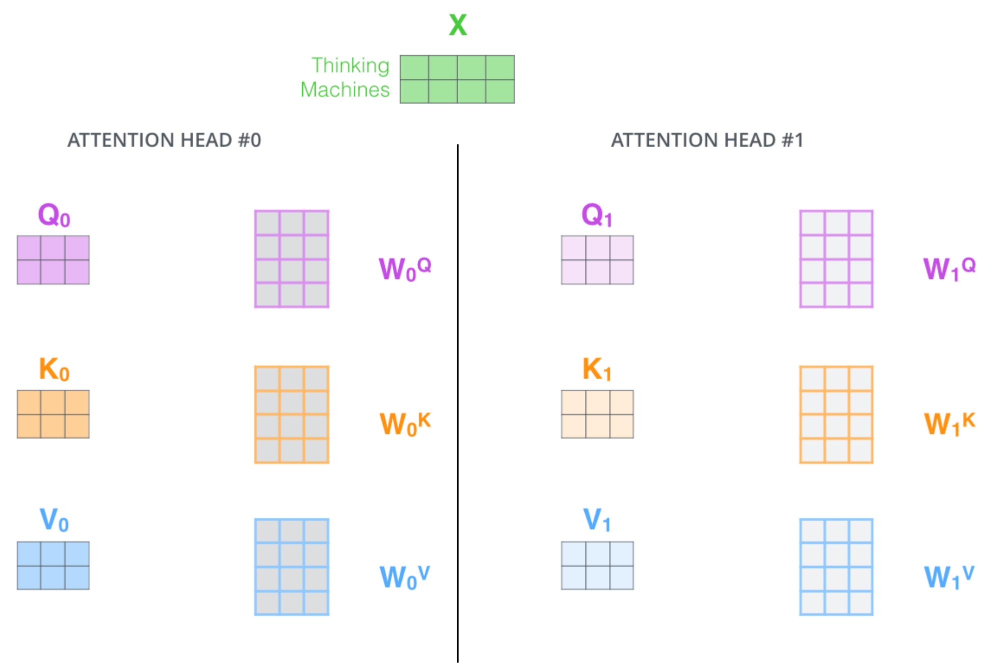
### Matrix Calculation of Self-Attention

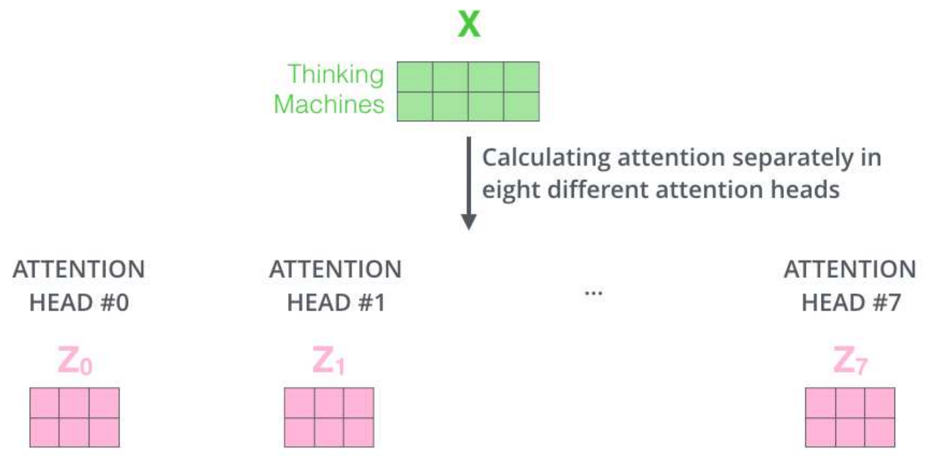


$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

# Illustrated Transformer

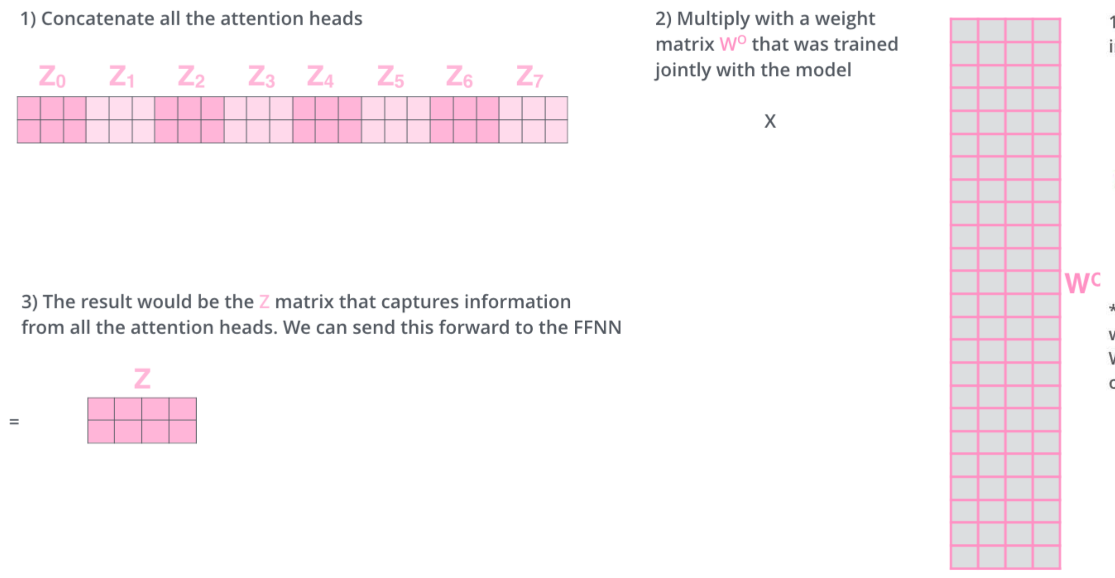
## The Beast With Many Heads





## Illustrated Transformer

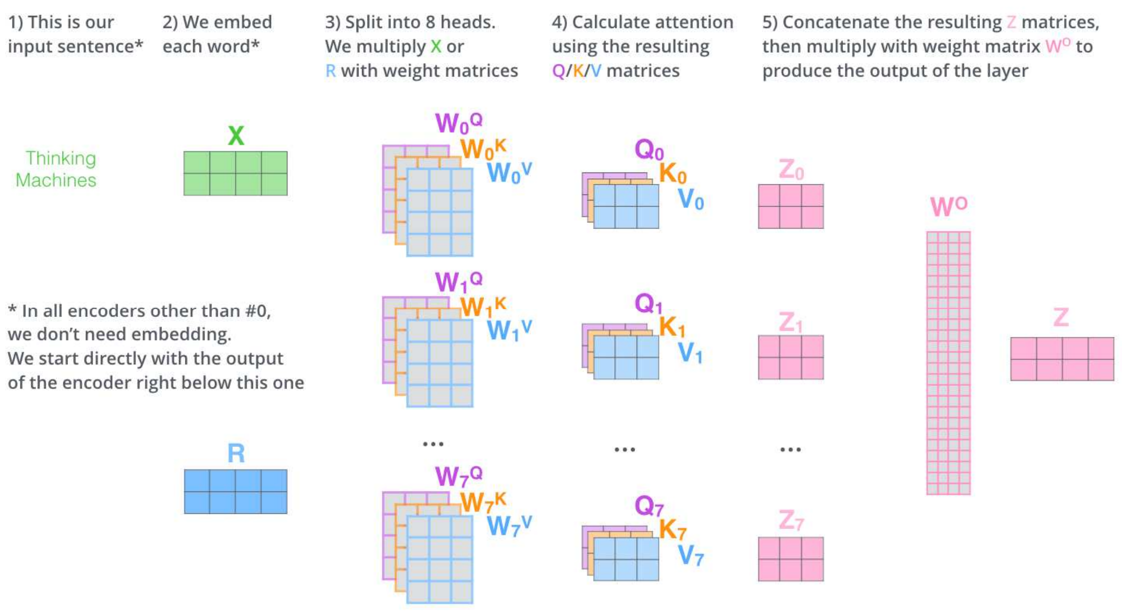
### The Beast With Many Heads



1) Concatenate all the attention heads

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

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



1) This is our input sentence*

2) We embed each word*

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

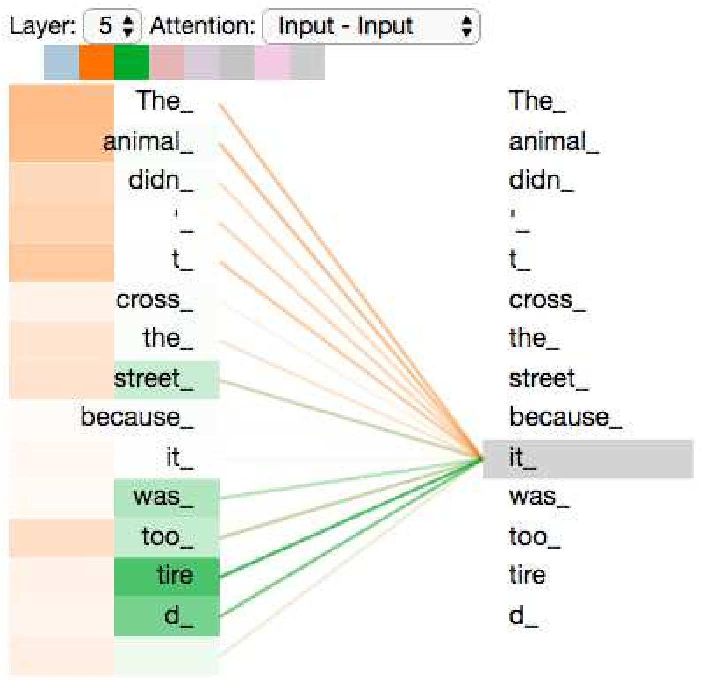
4) Calculate attention using the resulting $Q$/$K$/$V$ matrices

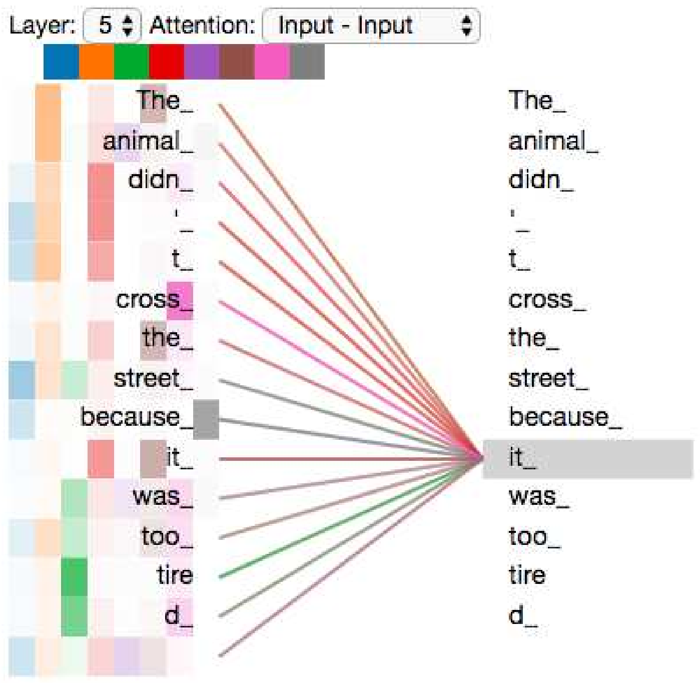
5) Concatenate the resulting $Z$ matrices, then multiply with weight matrix $W^O$ to produce the output of the layer

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

## Illustrated Transformer

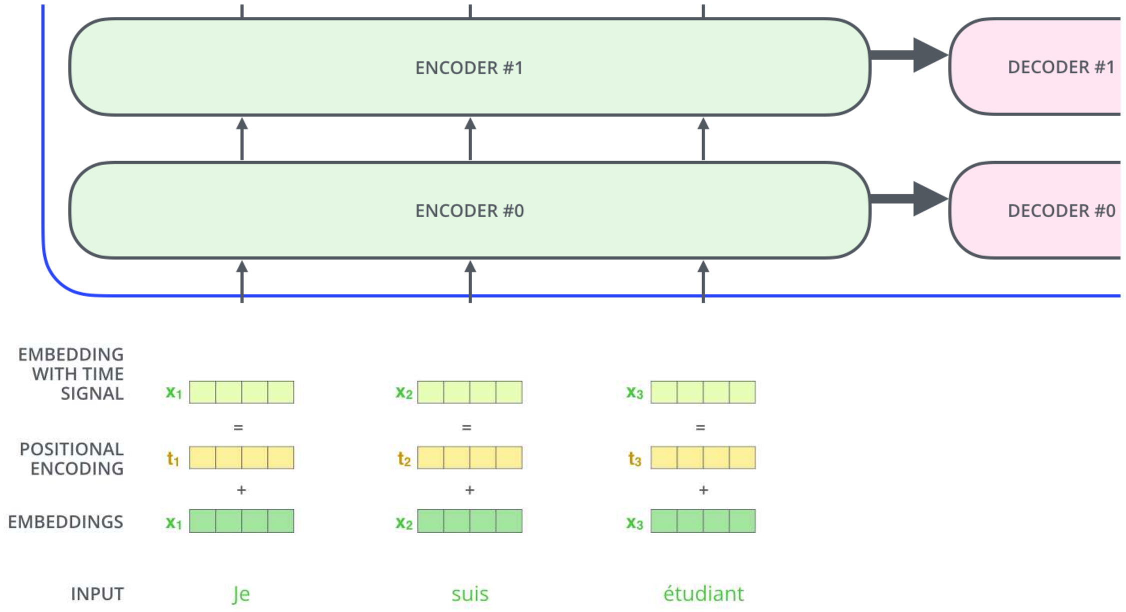
### The Beast With Many Heads

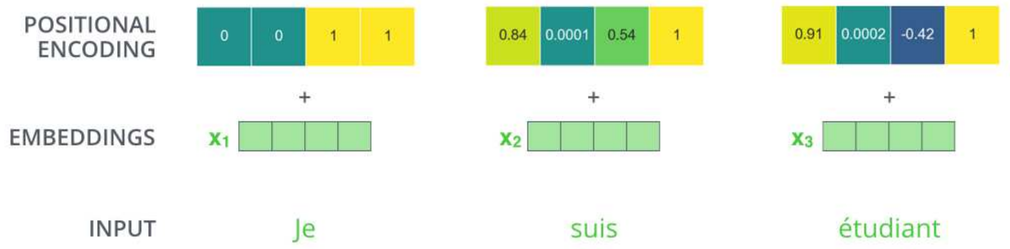




## Illustrated Transformer

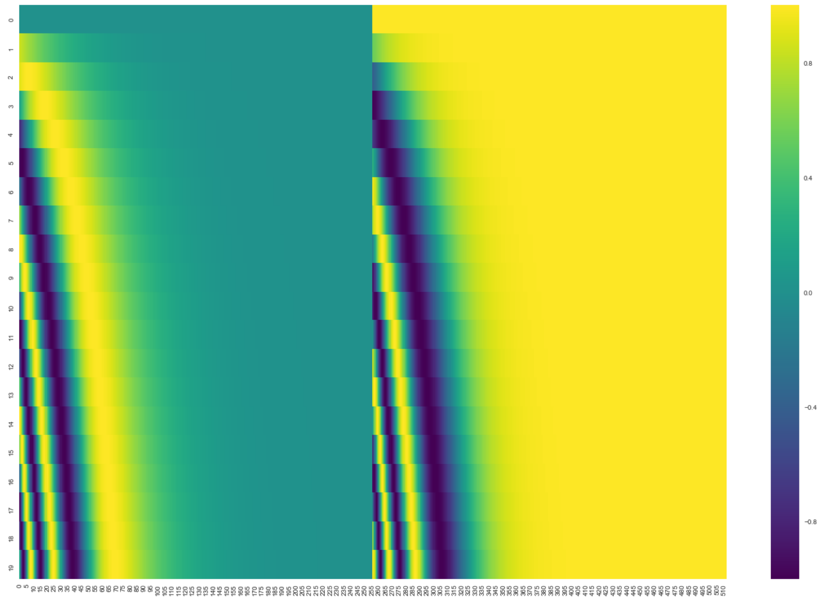
### Representing The Order of The Sequence Using Positional Encoding

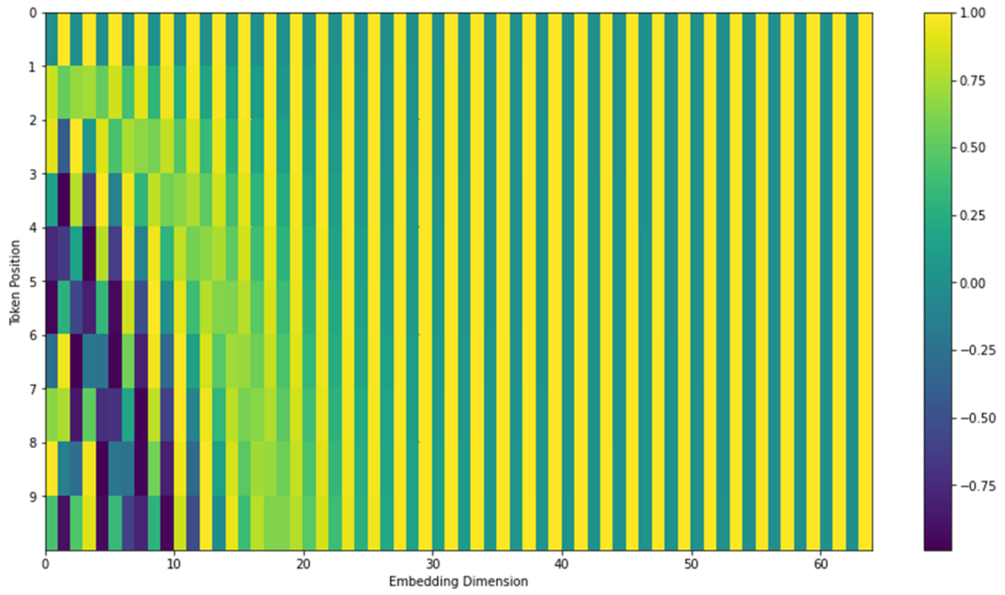




## Illustrated Transformer

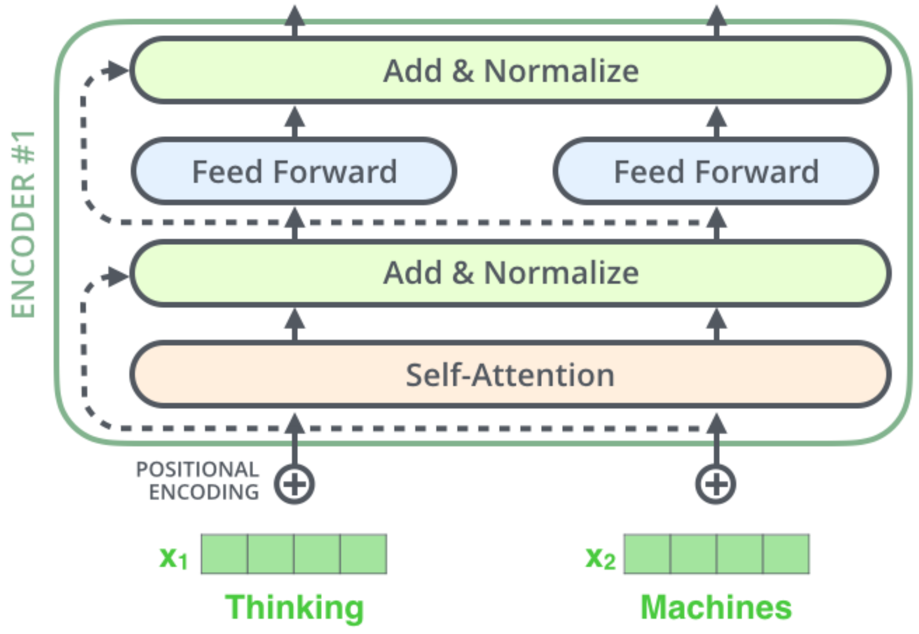
### Representing The Order of The Sequence Using Positional Encoding

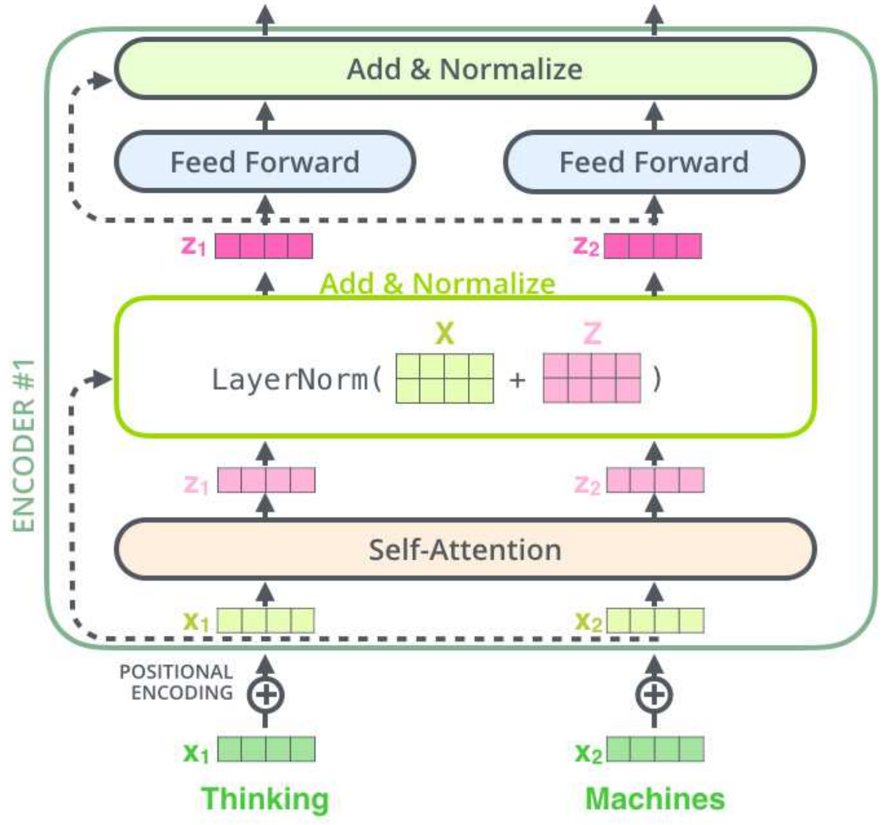




## Illustrated Transformer

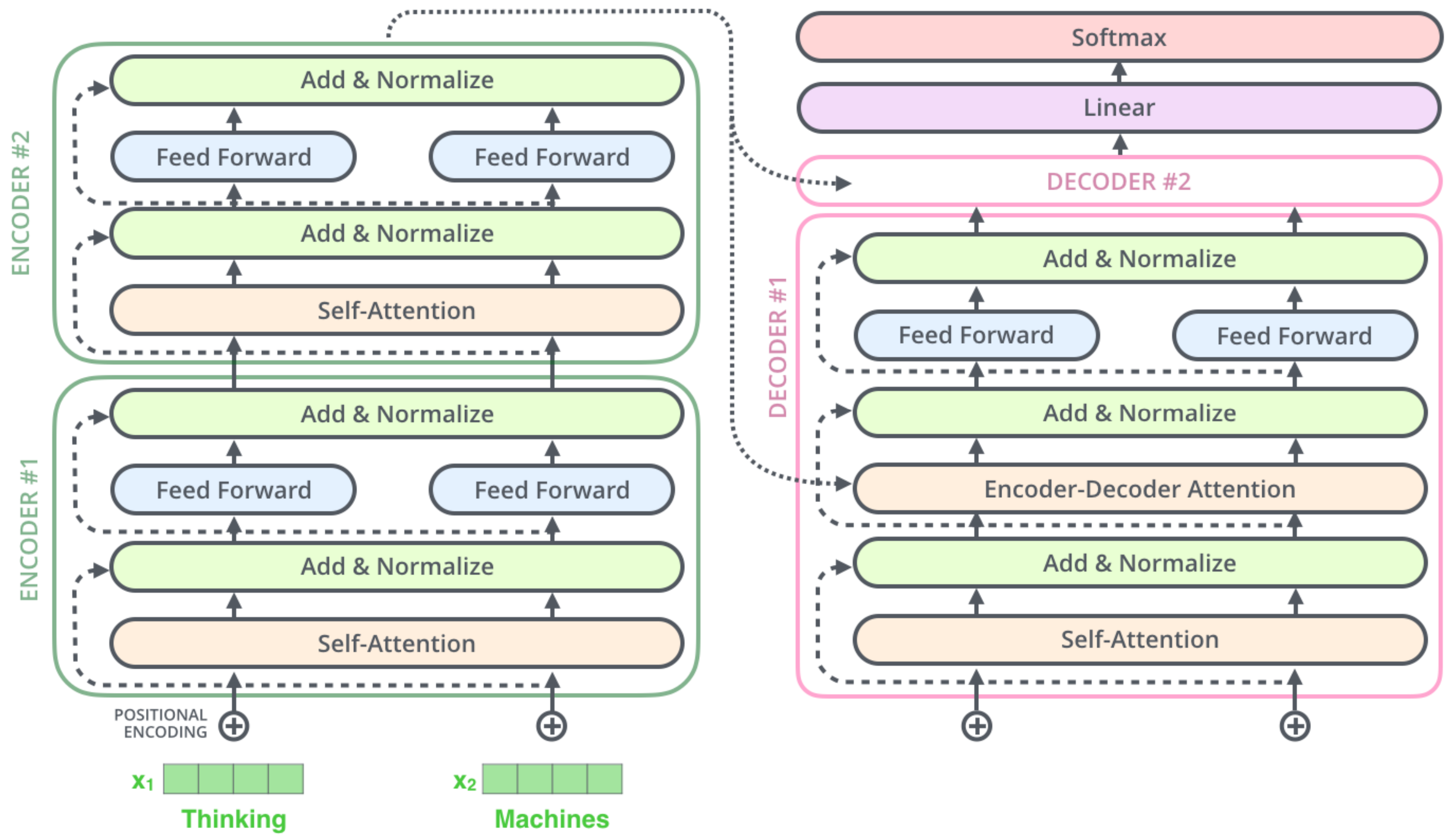
### The Residuals





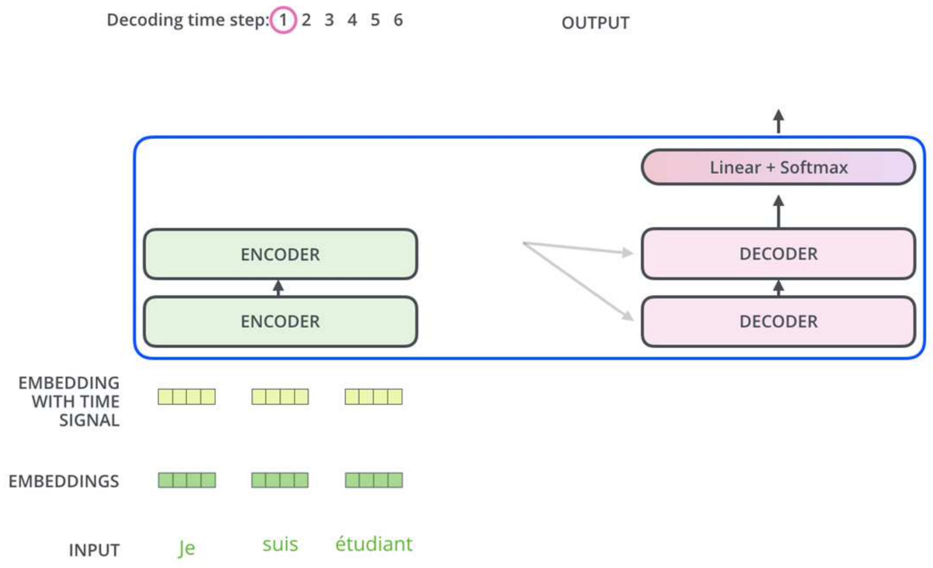
## Illustrated Transformer

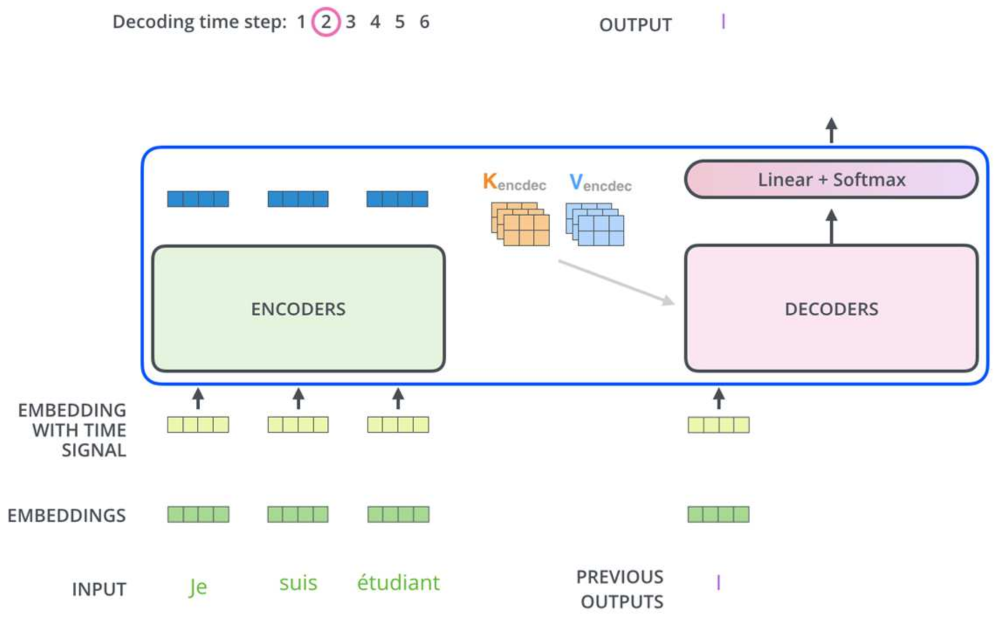
### The Residuals



## Illustrated Transformer

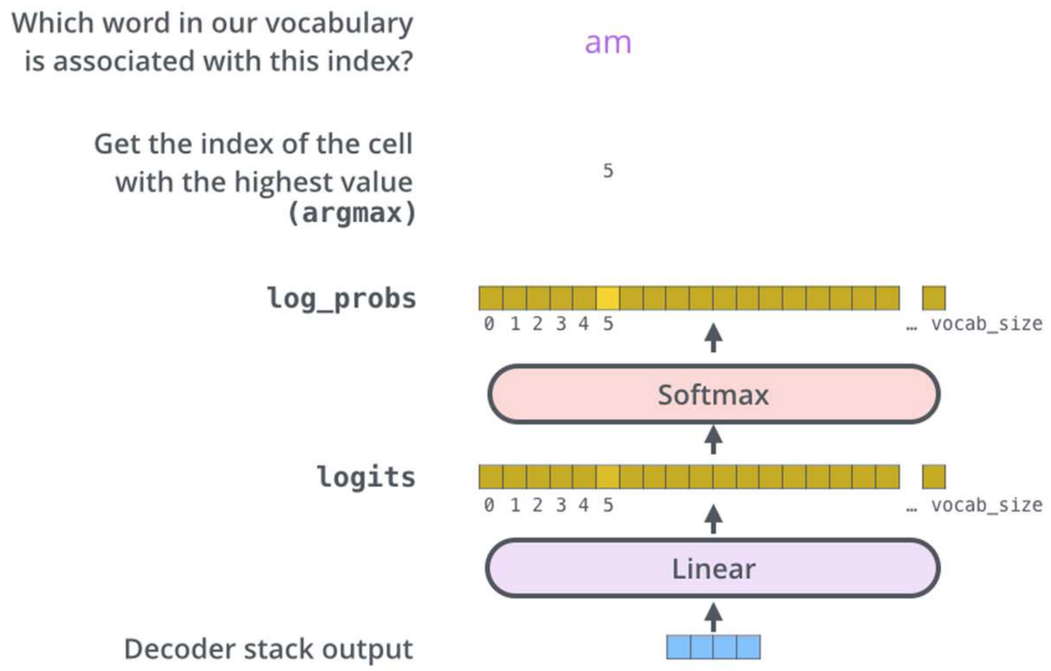
### The Decoder Side





## Illustrated Transformer

### The Final Linear and Softmax Layer


Which word in our vocabulary is associated with this index?
am
Get the index of the cell with the highest value (argmax)
5
log_probs
0 1 2 3 4 5 … vocab_size
Softmax
logits
0 1 2 3 4 5 … vocab_size
Linear
Decoder stack output

### Recap Of Training

Output Vocabulary

| WORD | a | am | I | thanks | student | <eos> |
|---|---|---|---|---|---|---|
| INDEX | 0 | 1 | 2 | 3 | 4 | 5 |

Output Vocabulary

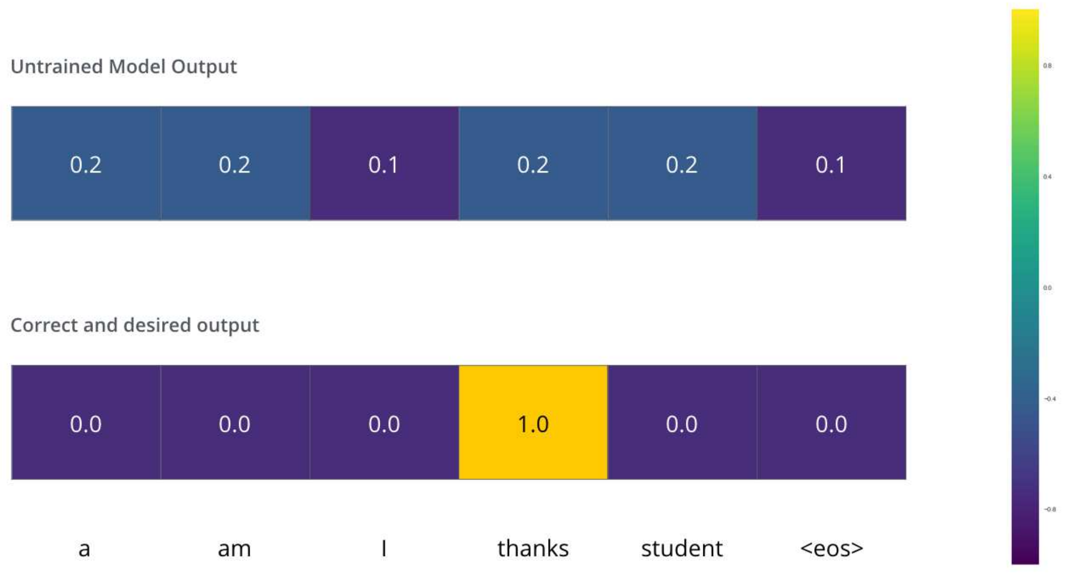
| WORD | a | am | I | thanks | student | <eos> |
|---|---|---|---|---|---|---|
| INDEX | 0 | 1 | 2 | 3 | 4 | 5 |

One-hot encoding of the word "am"

| | | | | | |
|---|---|---|---|---|---|
| 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |

## Illustrated Transformer

### The Loss Function

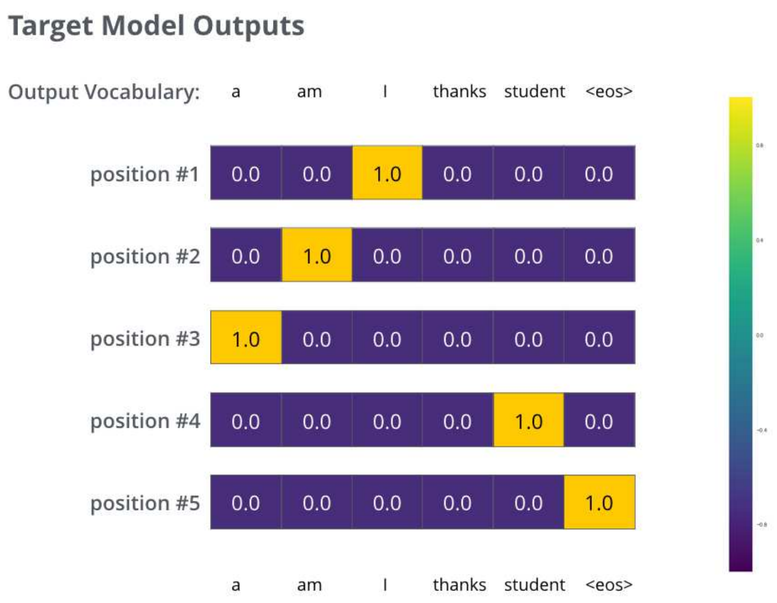


Untrained Model Output

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.2 | 0.1 | 0.2 | 0.2 | 0.1 |

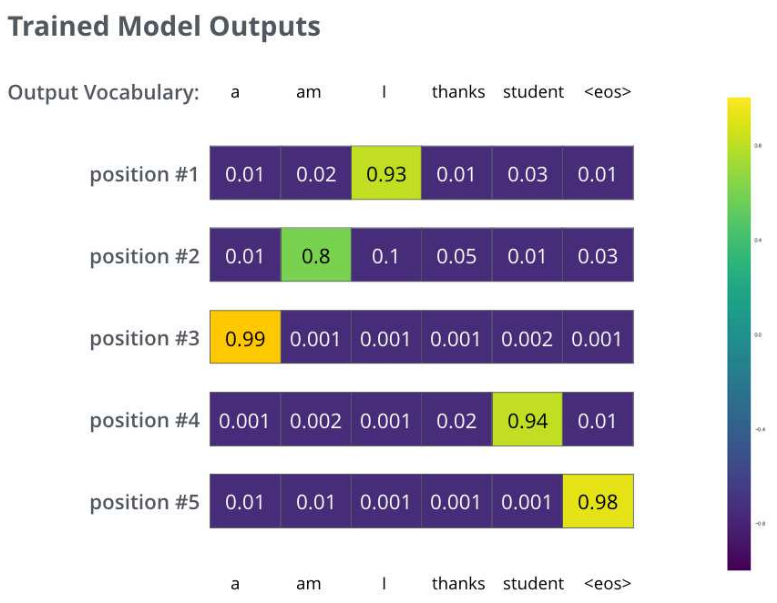
Correct and desired output

| | | | | | |
|---|---|---|---|---|---|
| 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| a | am | I | thanks | student | <eos> |



**Target Model Outputs**

| Output Vocabulary: | a | am | I | thanks | student | <eos> |
|---|---|---|---|---|---|---|
| position #1 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| position #2 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| position #3 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| position #4 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| position #5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| | a | am | I | thanks | student | <eos> |



**Trained Model Outputs**

| Output Vocabulary: | a | am | I | thanks | student | <eos> |
|---|---|---|---|---|---|---|
| position #1 | 0.01 | 0.02 | 0.93 | 0.01 | 0.03 | 0.01 |
| position #2 | 0.01 | 0.8 | 0.1 | 0.05 | 0.01 | 0.03 |
| position #3 | 0.99 | 0.001 | 0.001 | 0.001 | 0.002 | 0.001 |
| position #4 | 0.001 | 0.002 | 0.001 | 0.02 | 0.94 | 0.01 |
| position #5 | 0.01 | 0.01 | 0.001 | 0.001 | 0.001 | 0.98 |
| | a | am | I | thanks | student | <eos> |