An Introduction to Attention, Transformers, GPT-x, BERT, and ViT

Transformers

In deep learning, attention is a term used to describe a computational mechanism that allows the network to enhance important parts of the data and diminish other parts. What is important is learned by the attention mechanism from the data and its context.

Attention mechanisms were essentially tacked on to the existing network architectures starting around 2014.

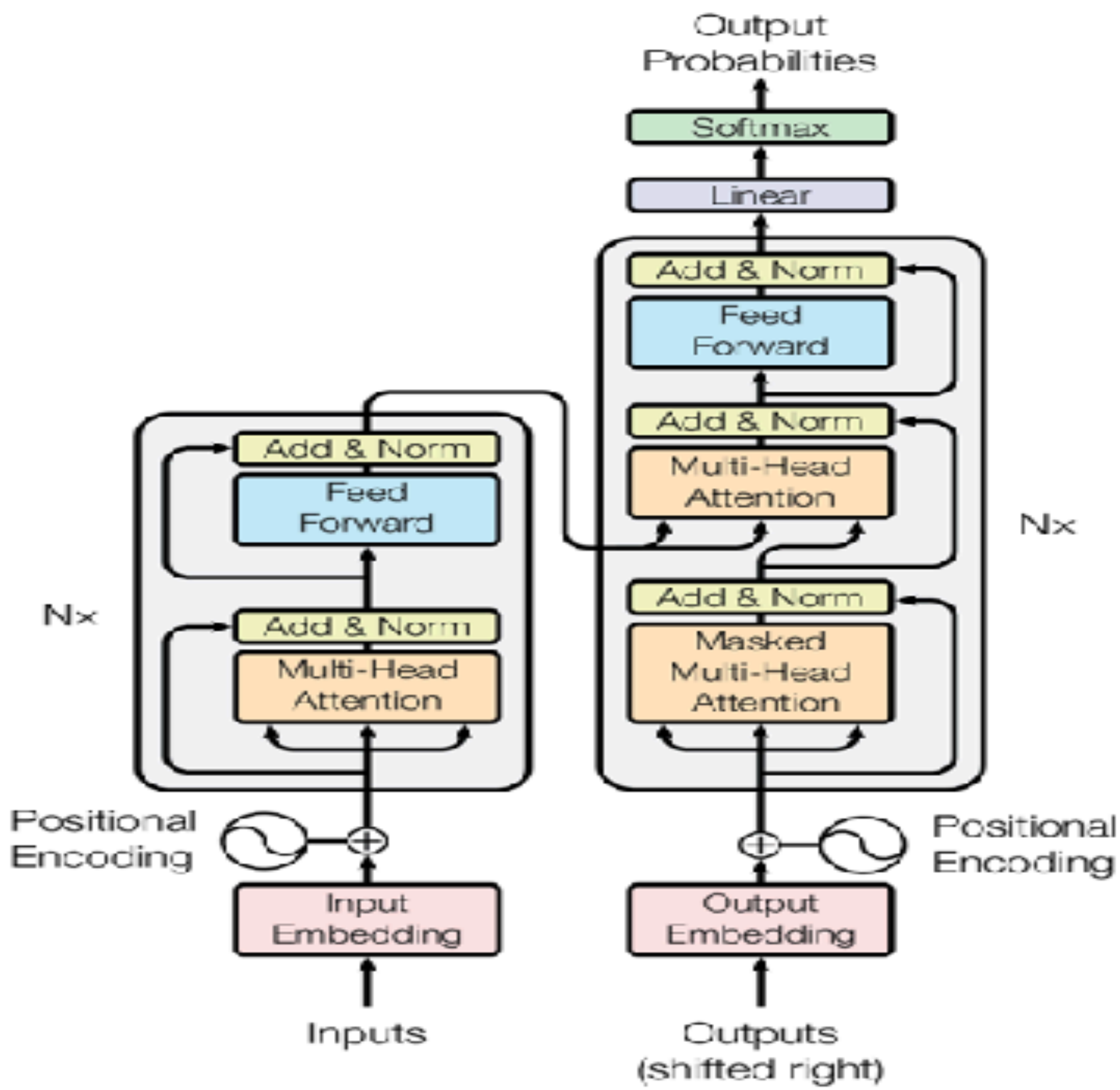
But in 2017, a new fully attention-based architecture was introduced called a **Transformer**. It was built for Machine Translation, but was quickly adapted to other NLP tasks.

This new architecture would give rise to jaw-dropping advances in NLP, begin to dominate in Computer Vision, and allow for the emergence of multi-modal networks combining text, images, video, audio, etc.

The Transformer

Vaswani et al. 2017

Attention is All You Need: Transformer



Vaswani et al. 2017

Transformer

The Transformer is originally designed as an encoder + decoder for machine translation, e.g., translating French to English.

The loss is standard cross entropy loss with teacher forcing:

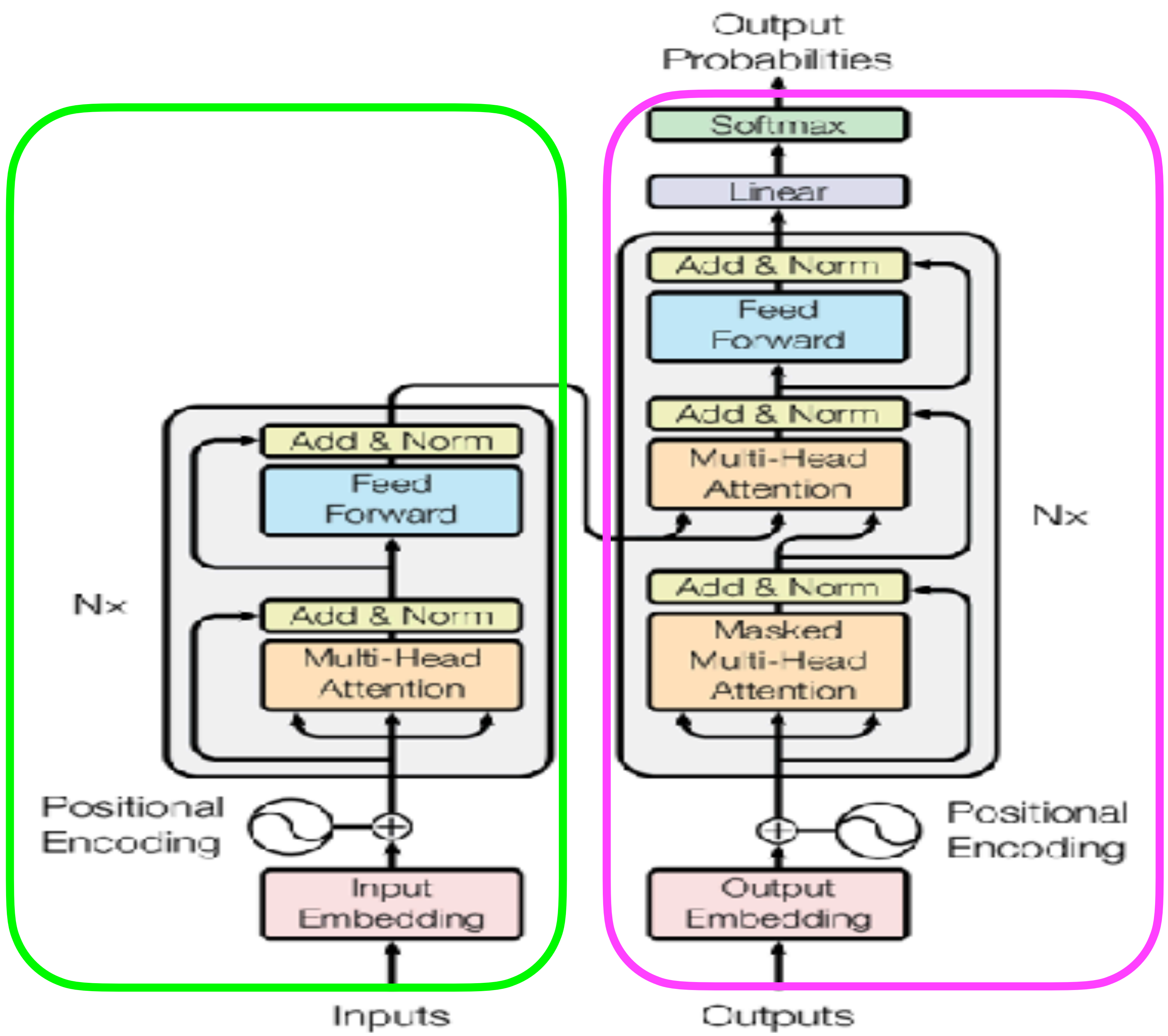
$$L(\theta) = -\sum_{t=1}^{N} \log p_\theta(w_t^B \,|\, w_{1:M}^A, w_{1:t-1}^B)$$

where p_θ is the model,

$w_{1:M}^A$ are words $w_1^A, \ldots, w_M^A$ in the source language,

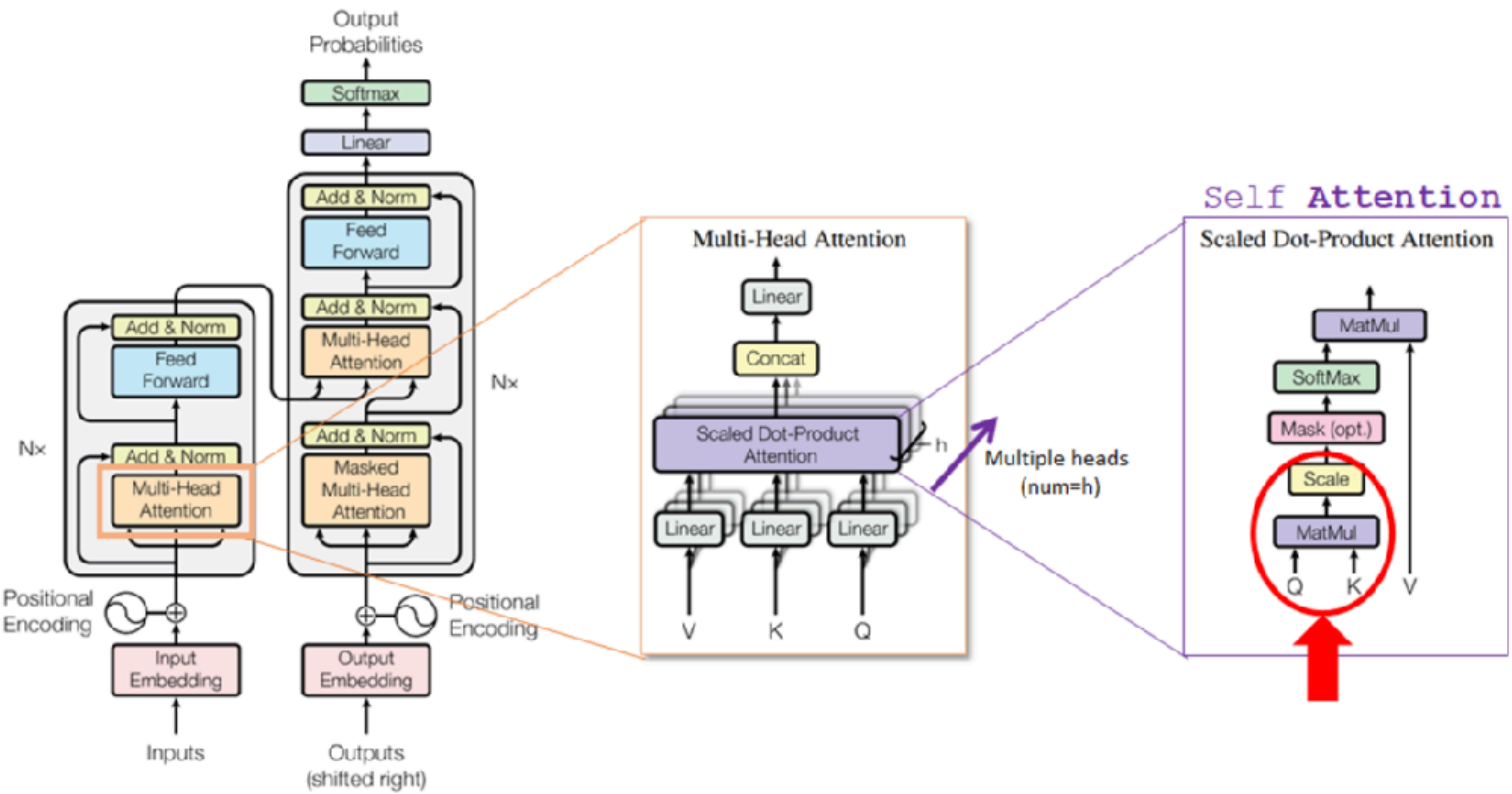
$w_{1:N}^B$ are words $w_1^B, \ldots, w_M^B$ in the target language

Transformer: Encoders + Decoders

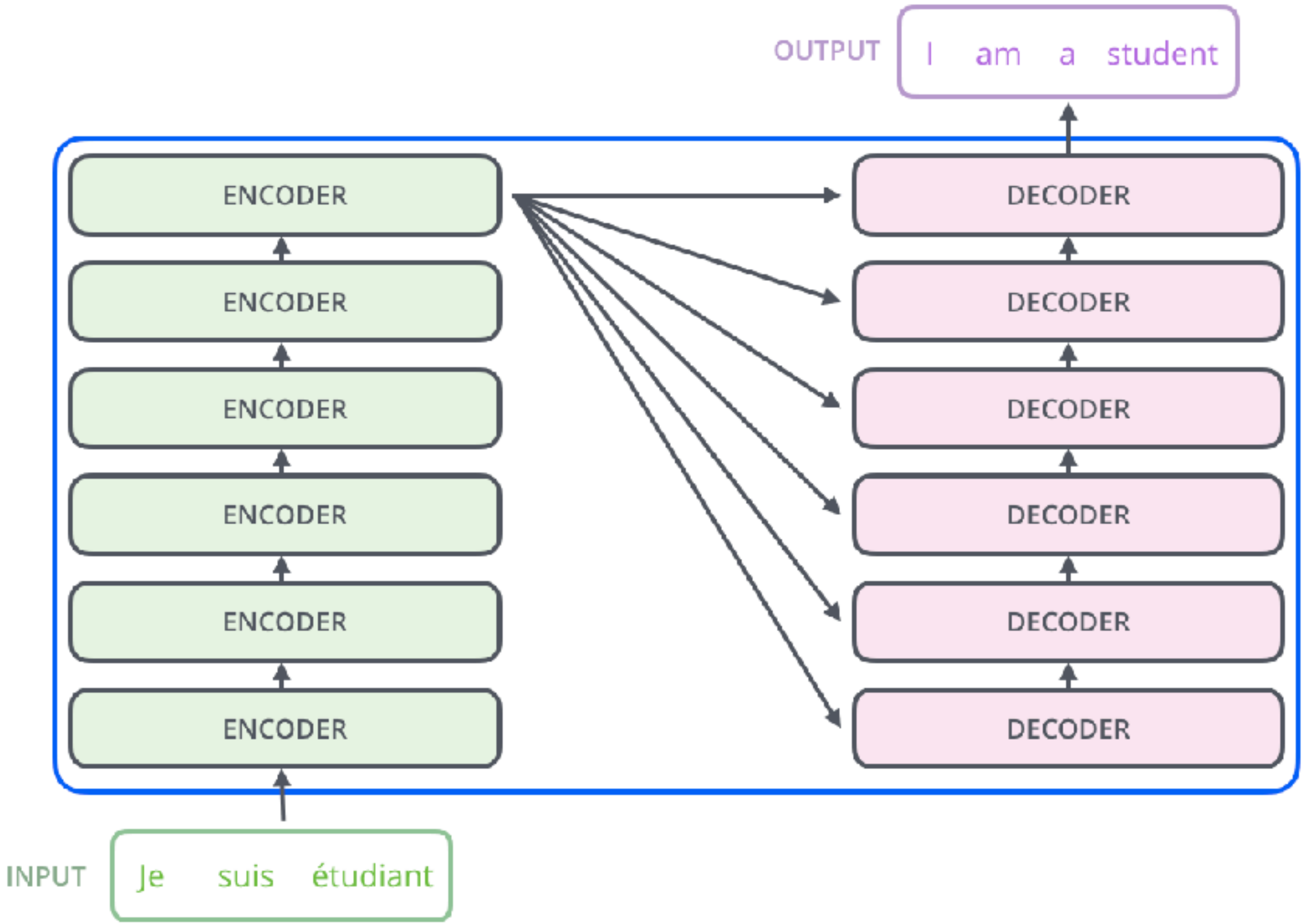


Encoders Decoders

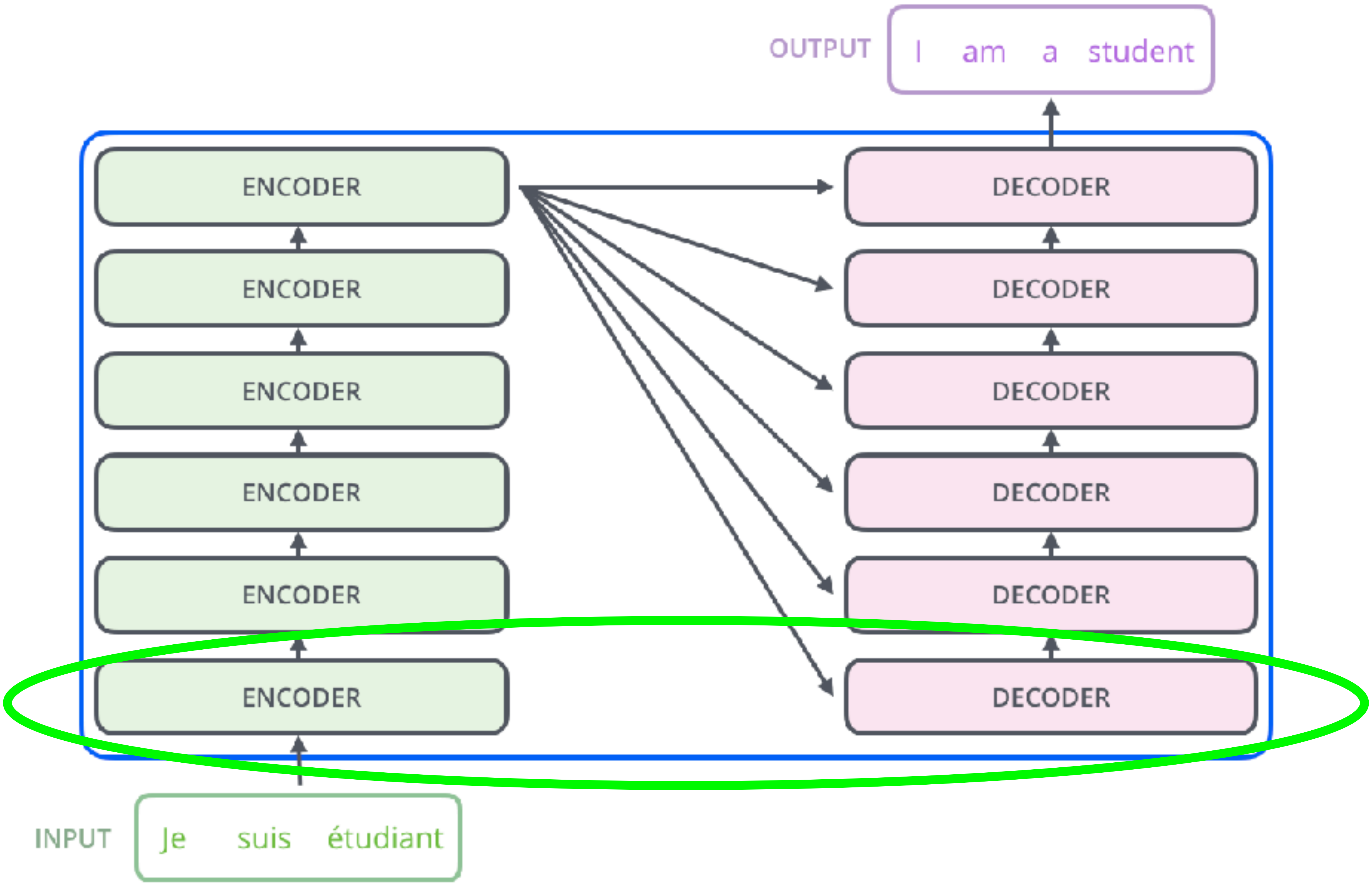
Transformer: Self-Attention



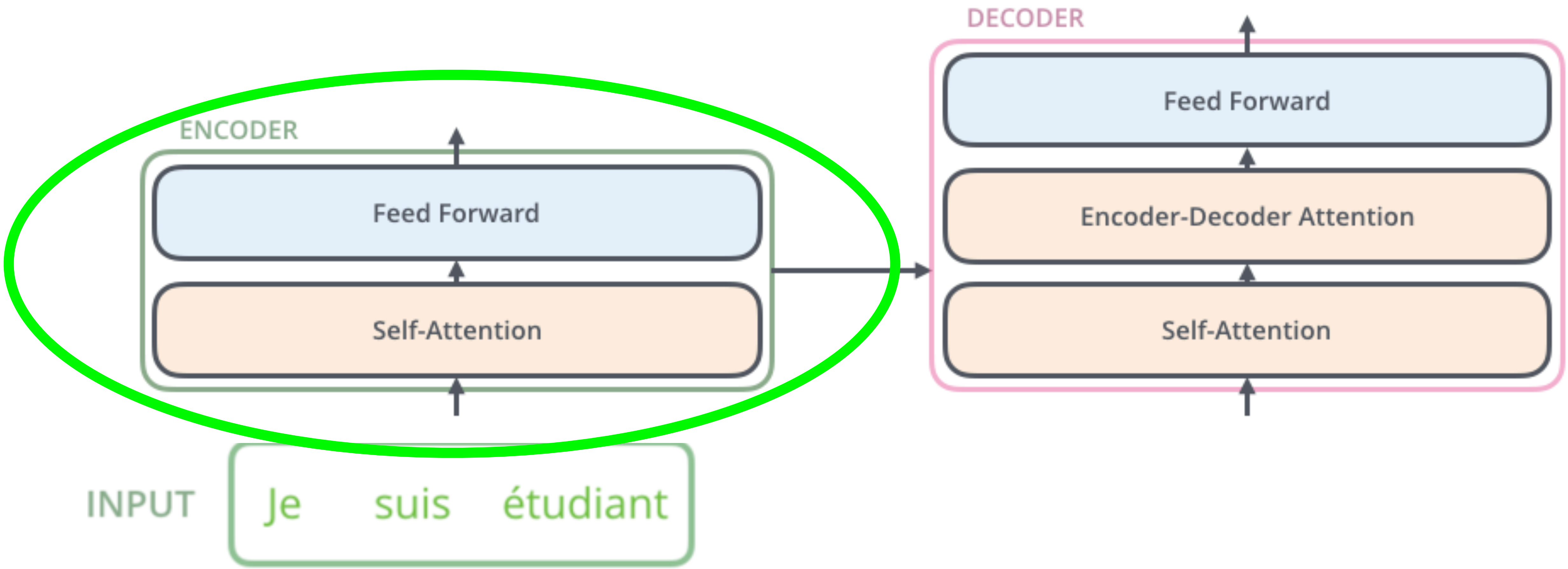
Transformer: Encoders + Decoders



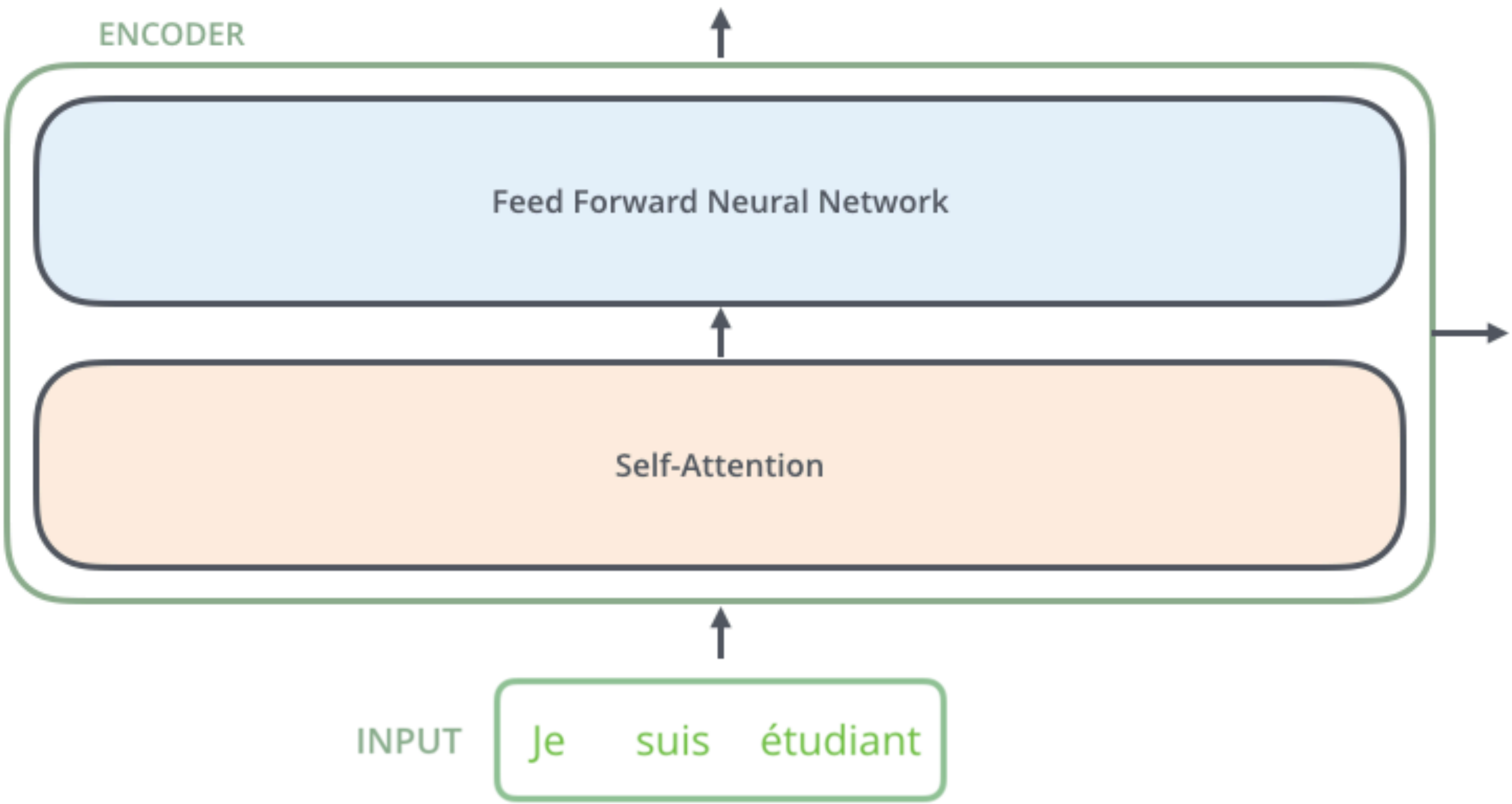
Transformer: Encoders + Decoders



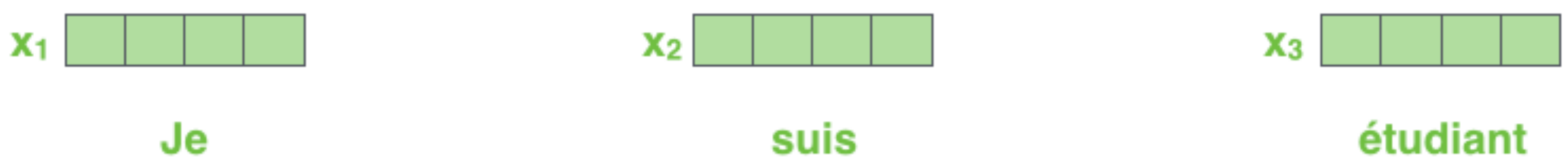
Transformer: Encoder + Decoder



Transformer: Encoder

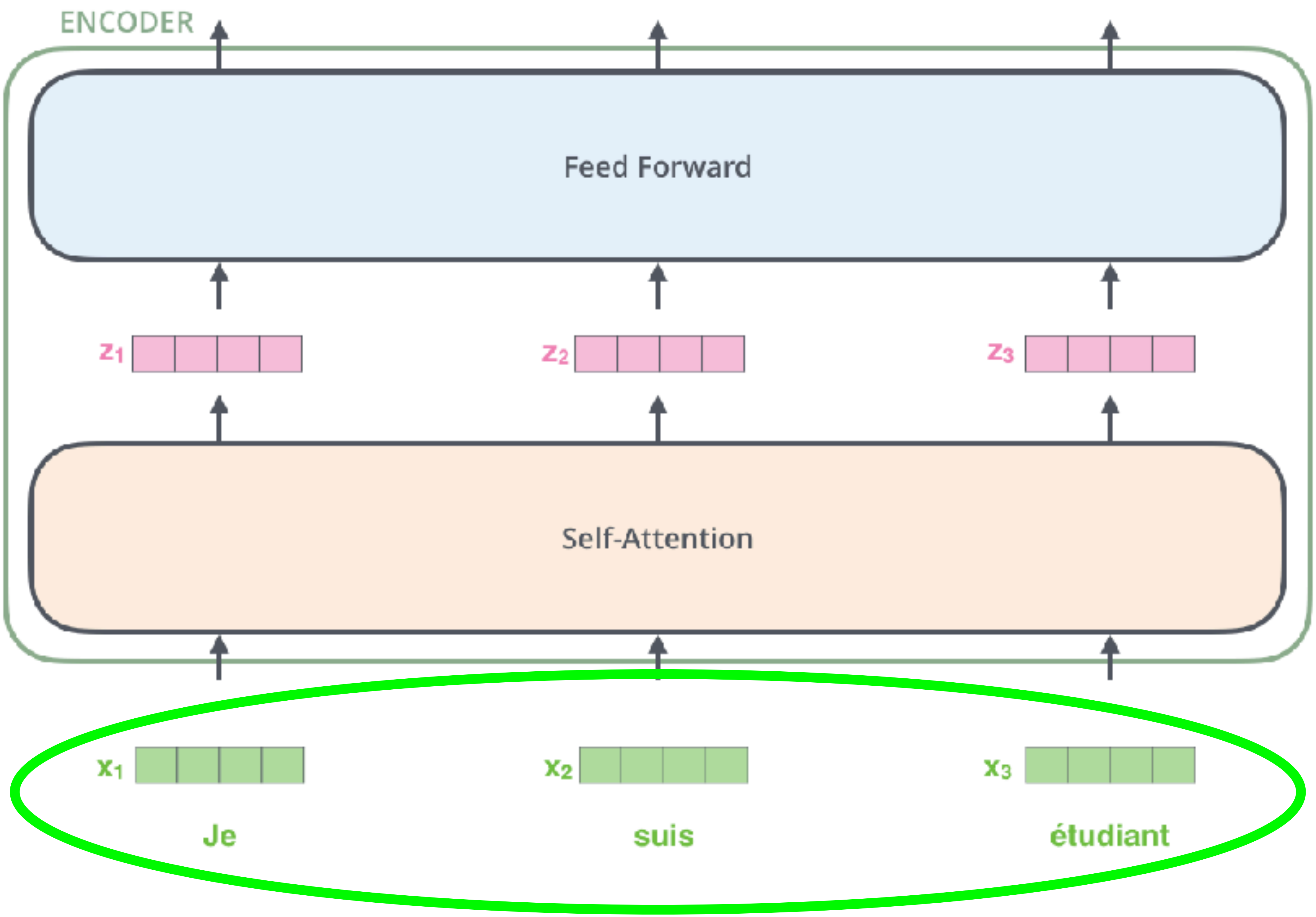


Transformer: Inputs Words Mapped to Vectors

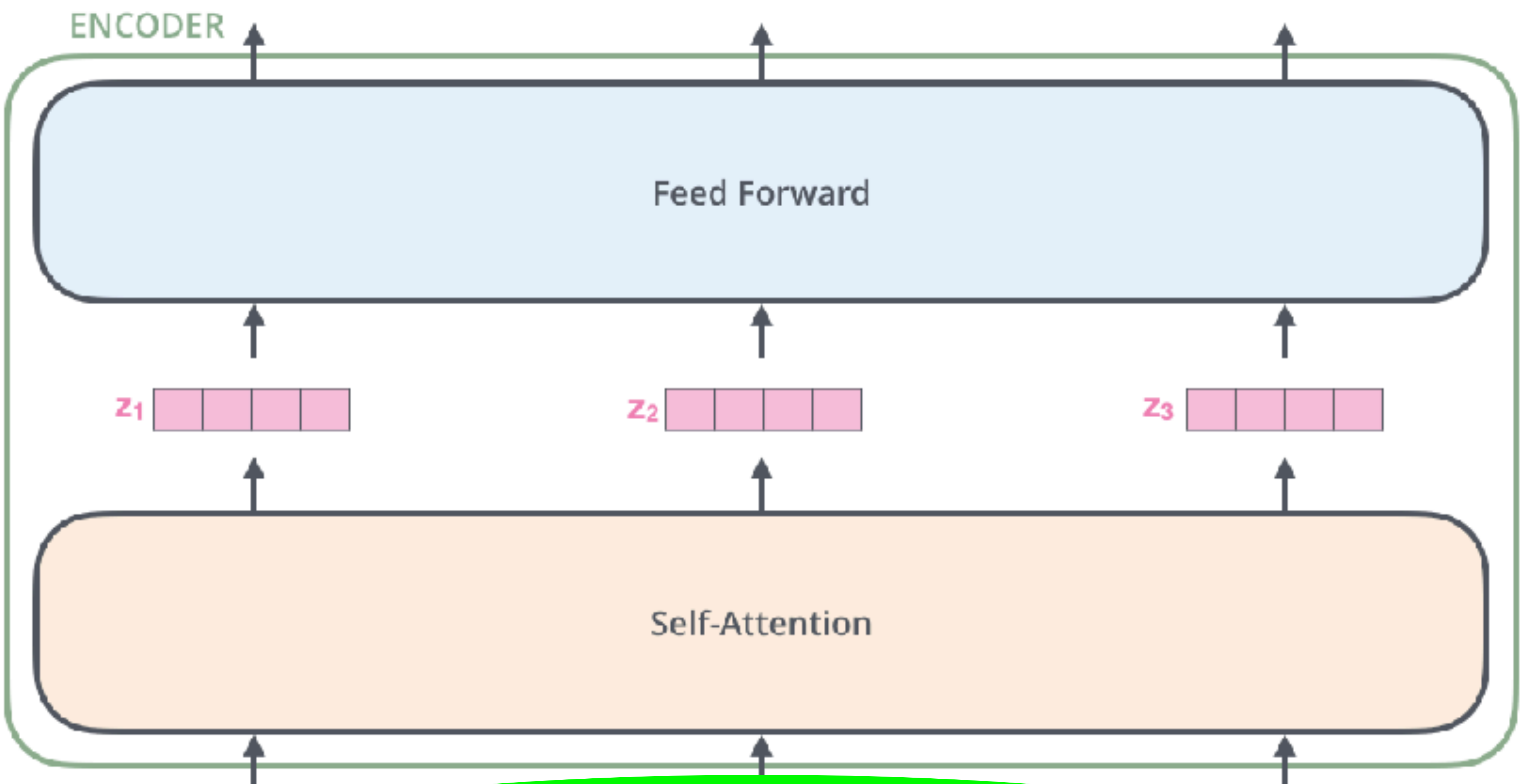


Before going into the network words are tokenized and mapped to vectors.

Transformer: Encoder

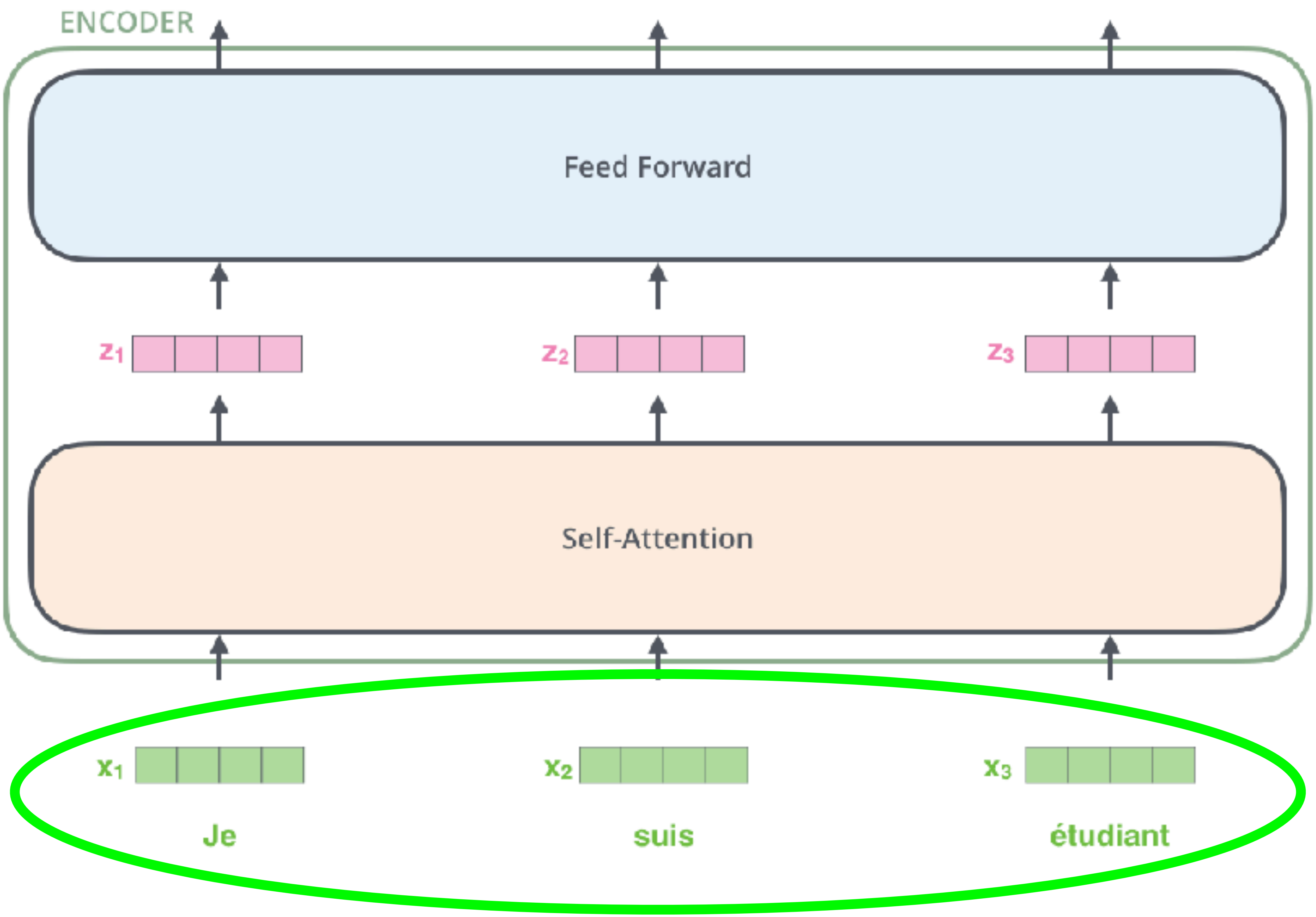


Transformer: Encoder

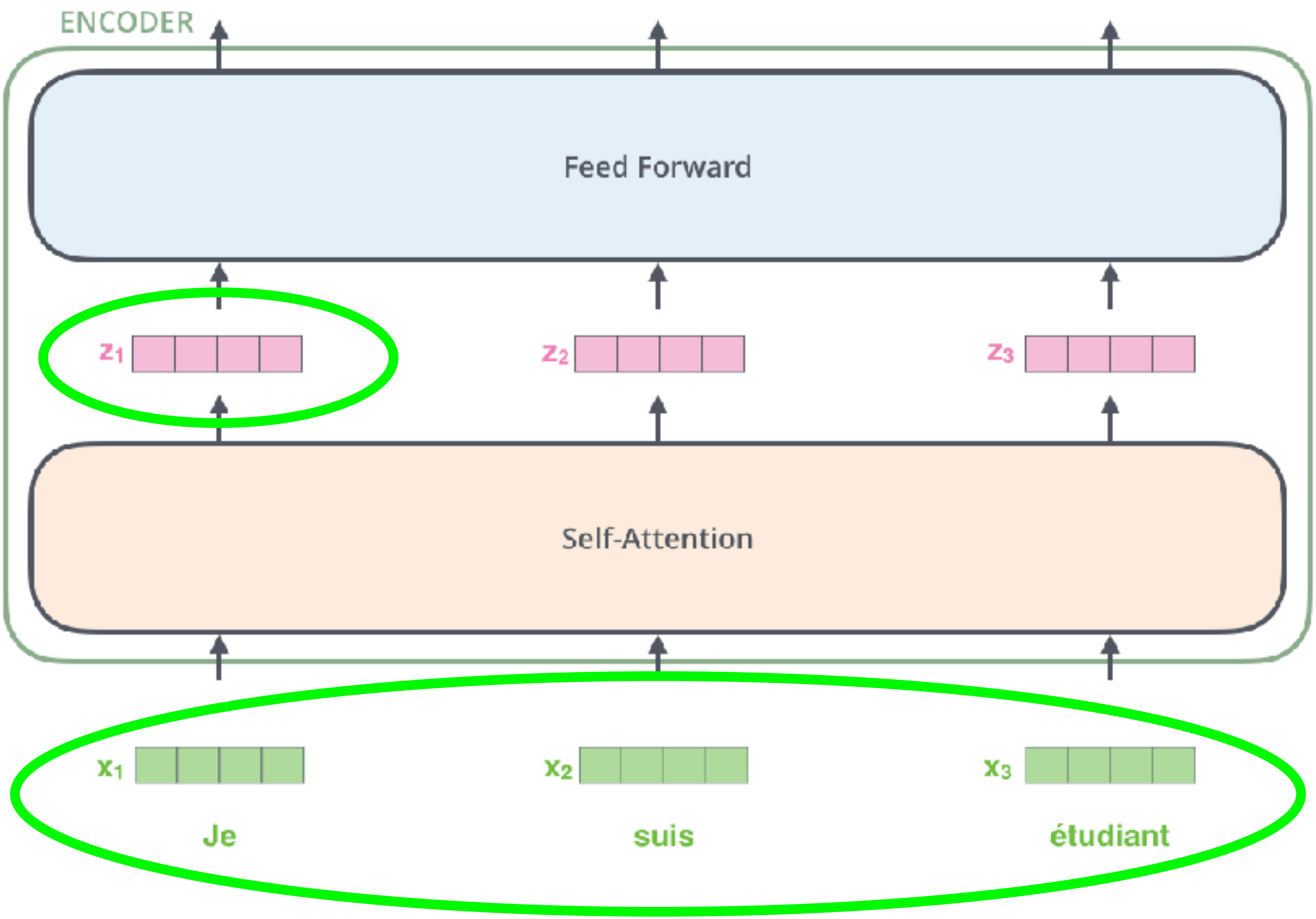


All word vectors go in at same time!

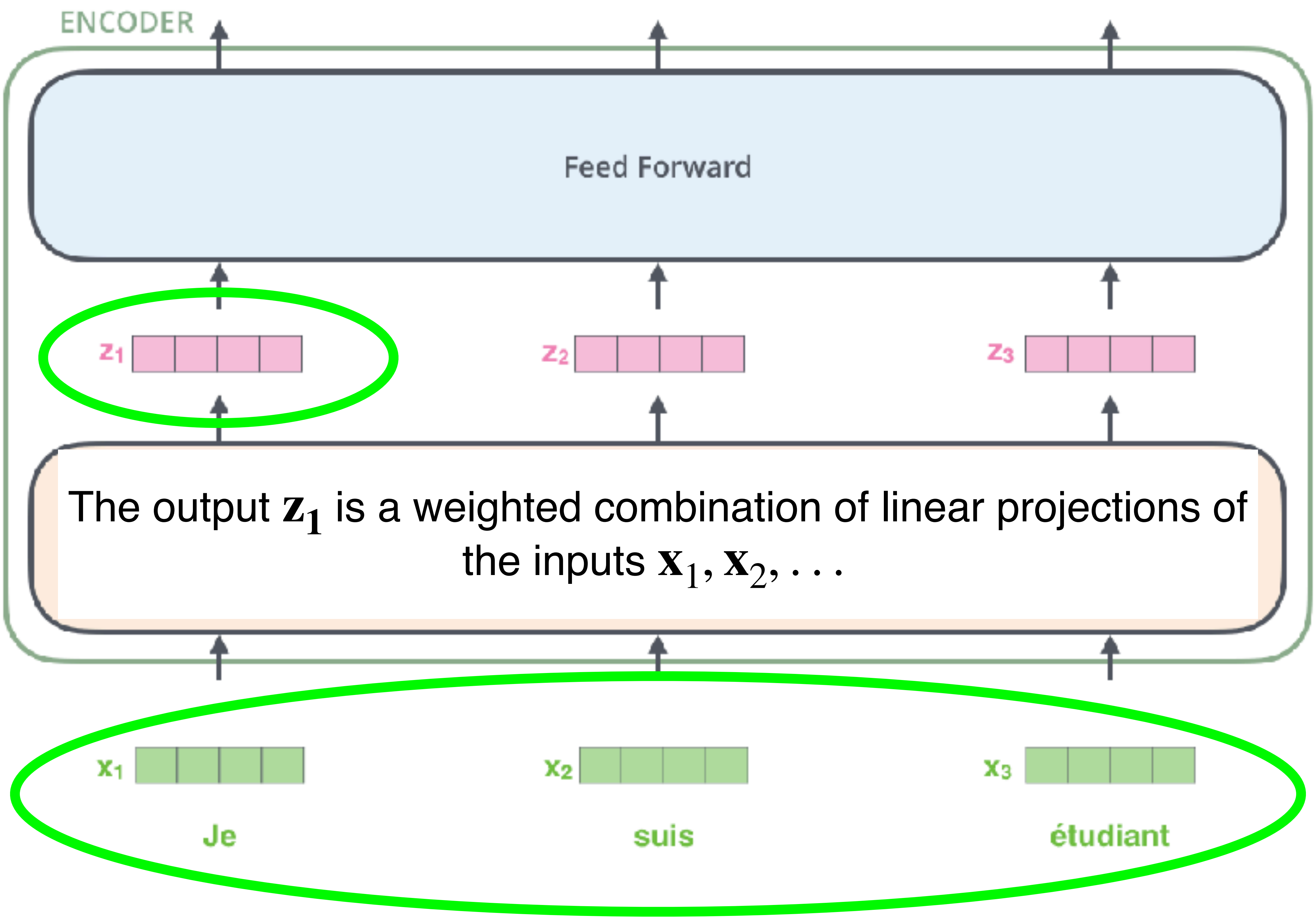
Transformer: Encoder



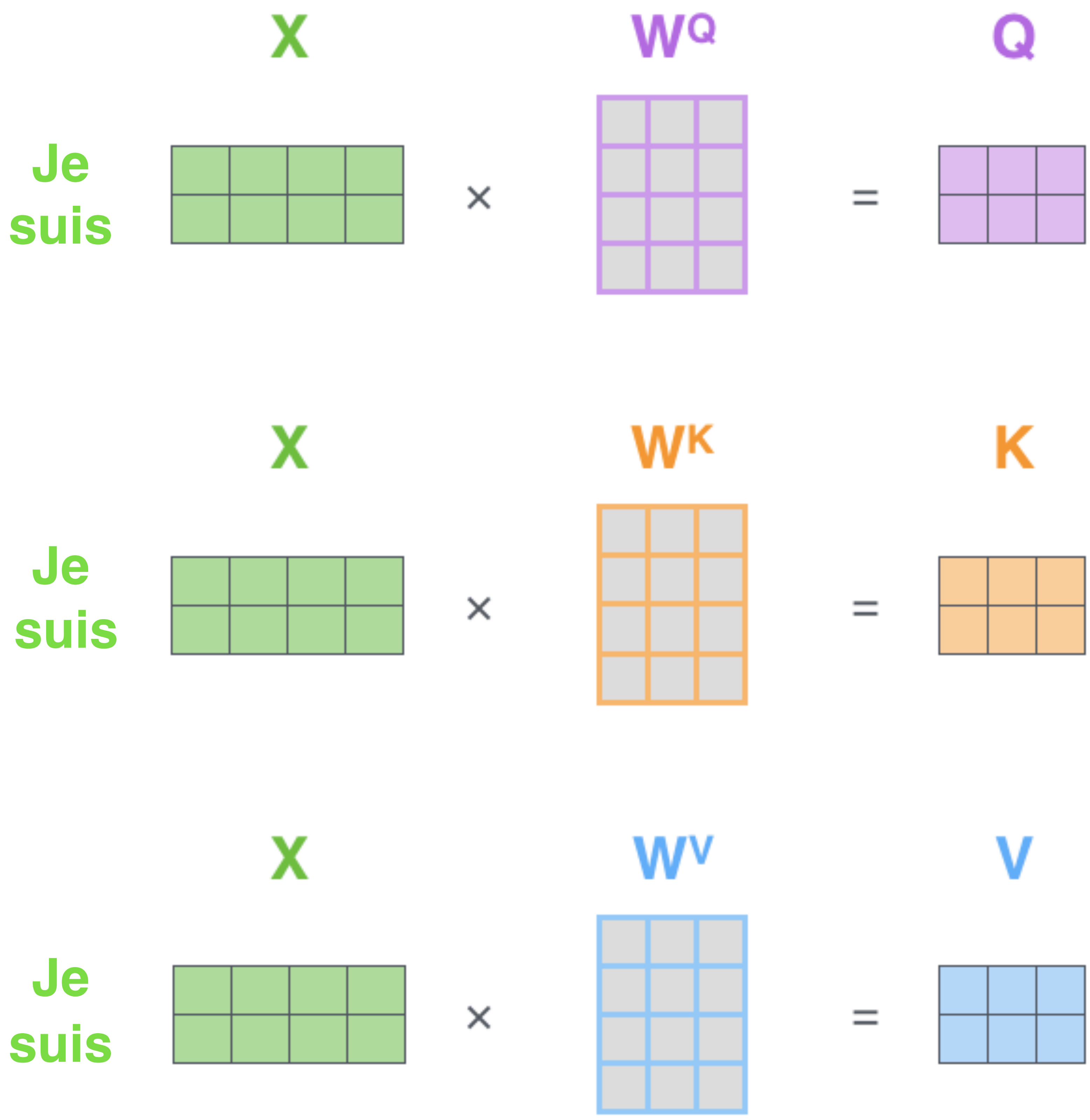
Transformer: Encoder



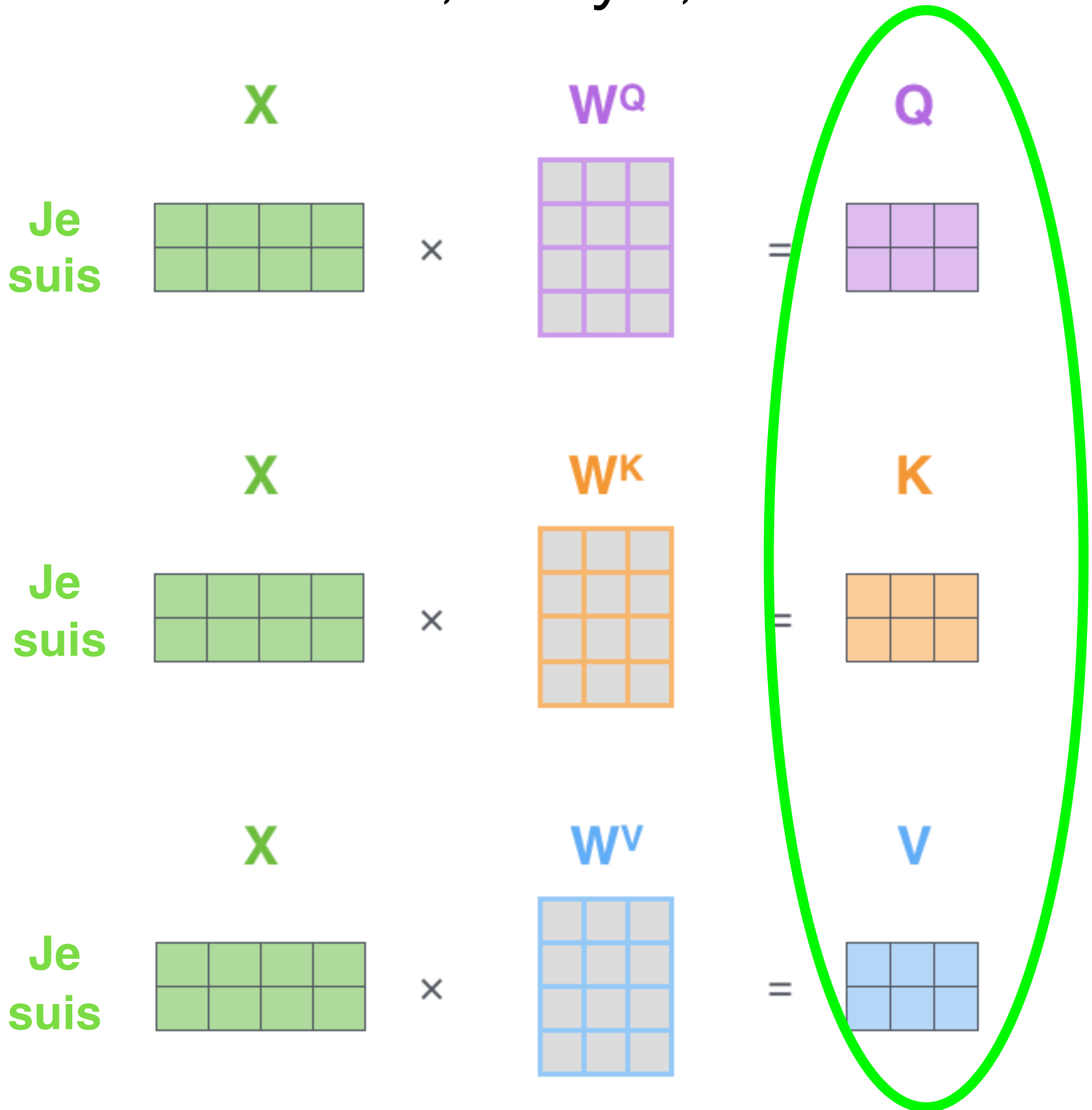
Transformer: Encoder



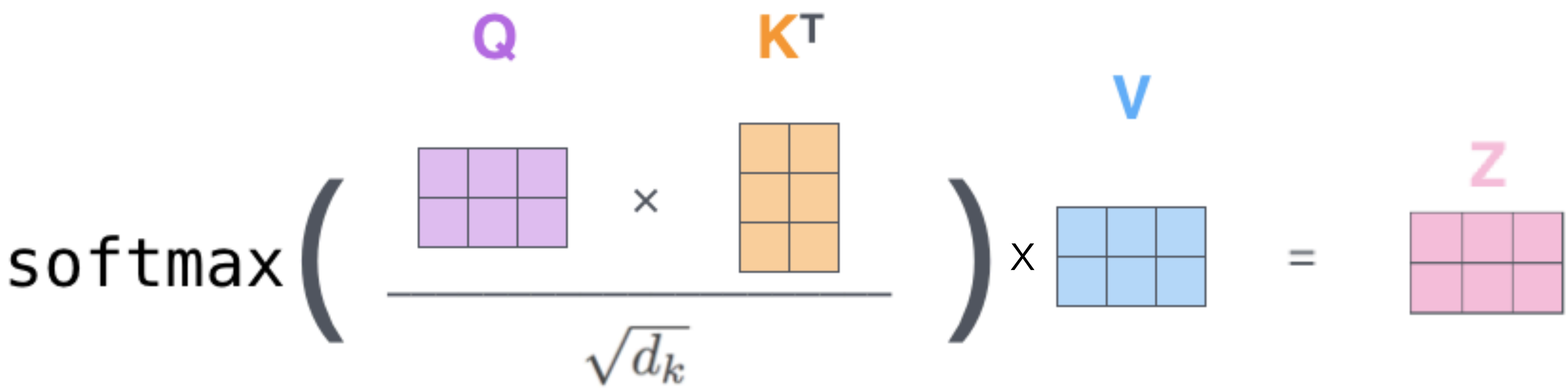
Queries, Keys, and Values



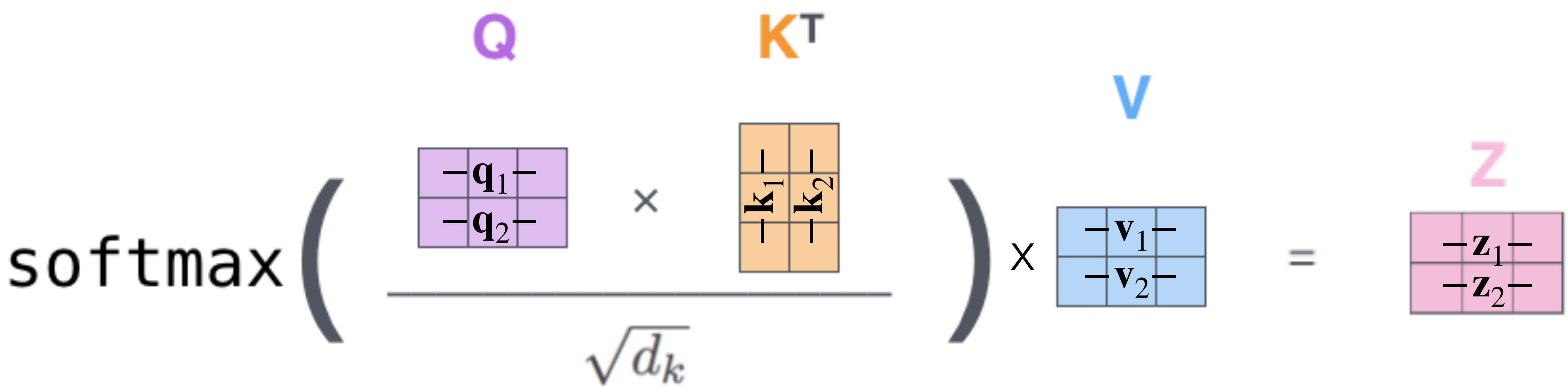
Queries, Keys, and Values



Queries, Keys, and Values



Queries, Keys, and Values



$$\mathbf{z}_1 = \frac{e^{\beta\, \mathbf{q}_1 \mathbf{k}_1^T}\, \mathbf{v}_1 + e^{\beta\, \mathbf{q}_1 \mathbf{k}_2^T}\, \mathbf{v}_2}{e^{\beta\, \mathbf{q}_1 \mathbf{k}_1^T} + e^{\beta\, \mathbf{q}_1 \mathbf{k}_2^T}}$$

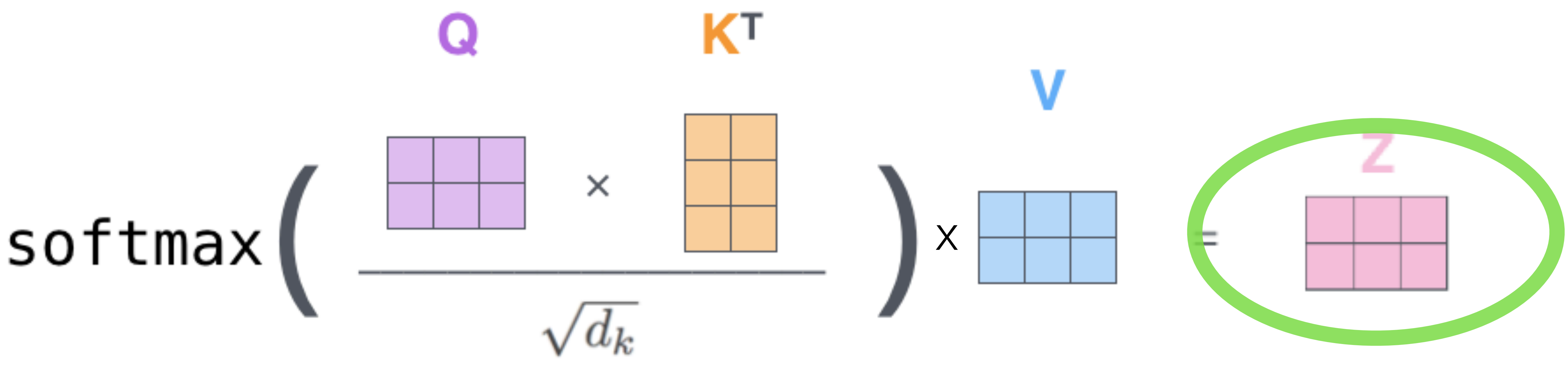
and

$$\mathbf{z}_2 = \frac{e^{\beta\, \mathbf{q}_2 \mathbf{k}_1^T}\, \mathbf{v}_1 + e^{\beta\, \mathbf{q}_2 \mathbf{k}_2^T}\, \mathbf{v}_2}{e^{\beta\, \mathbf{q}_2 \mathbf{k}_1^T} + e^{\beta\, \mathbf{q}_2 \mathbf{k}_2^T}}$$

where $\beta = \frac{1}{\sqrt{d_k}}$

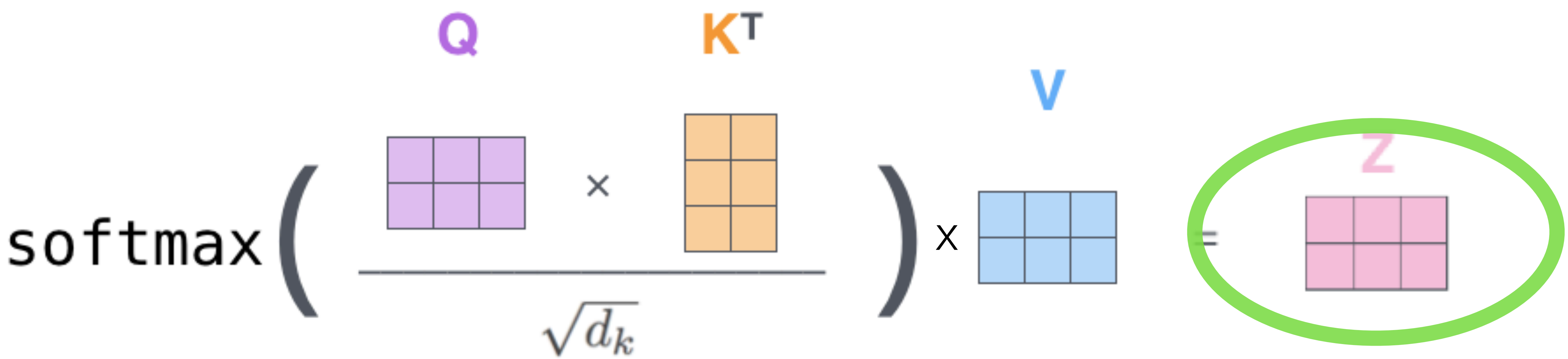
He often considered his memories as traffic on the congested NYC avenues, with his days at Columbia like a truck painted red with all horns blaring.

Queries, Keys, and Values



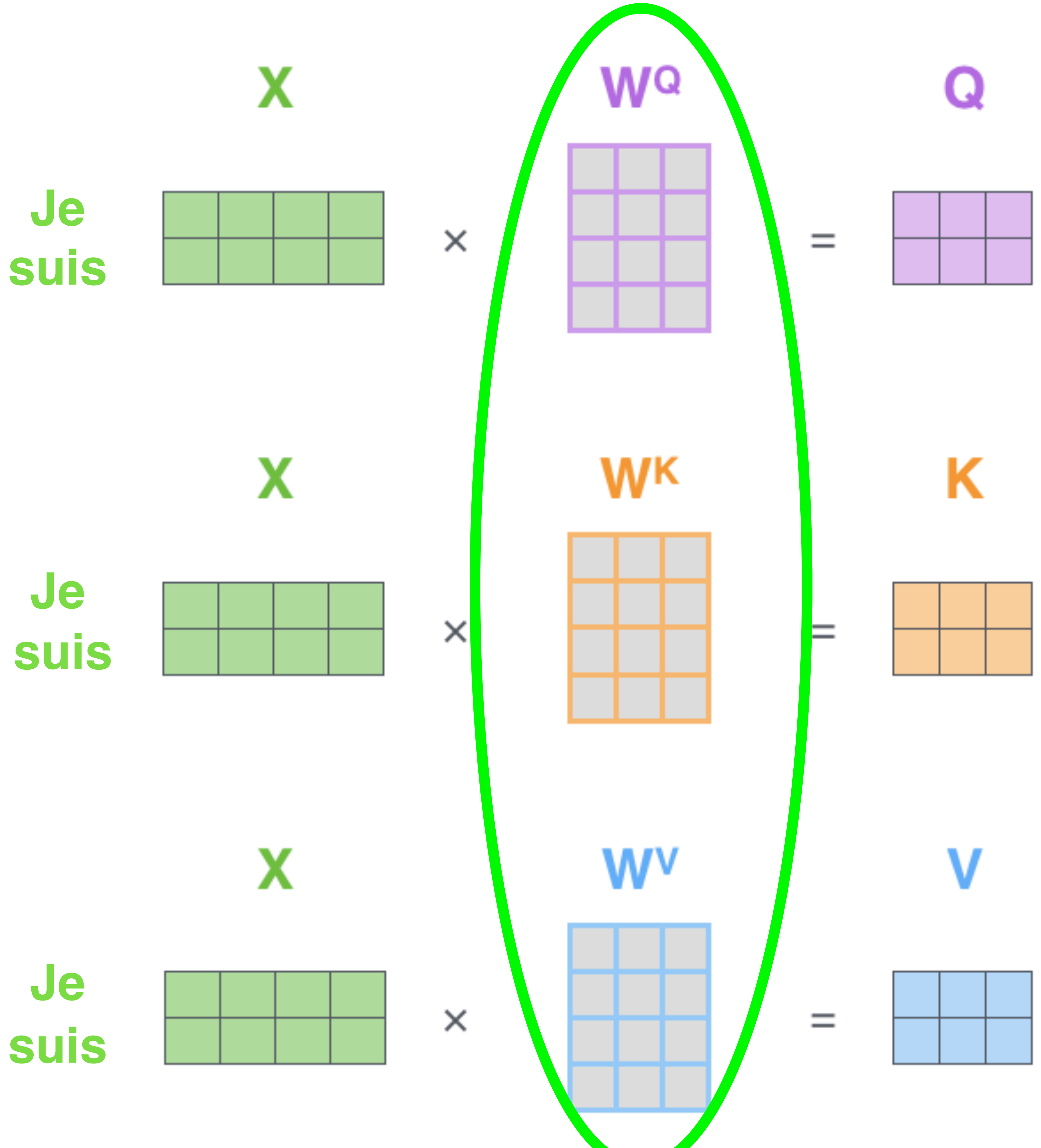
For each input word, we get a composite vector that combines representations from all words in the sequence through "self-attention."

Queries, Keys, and Values

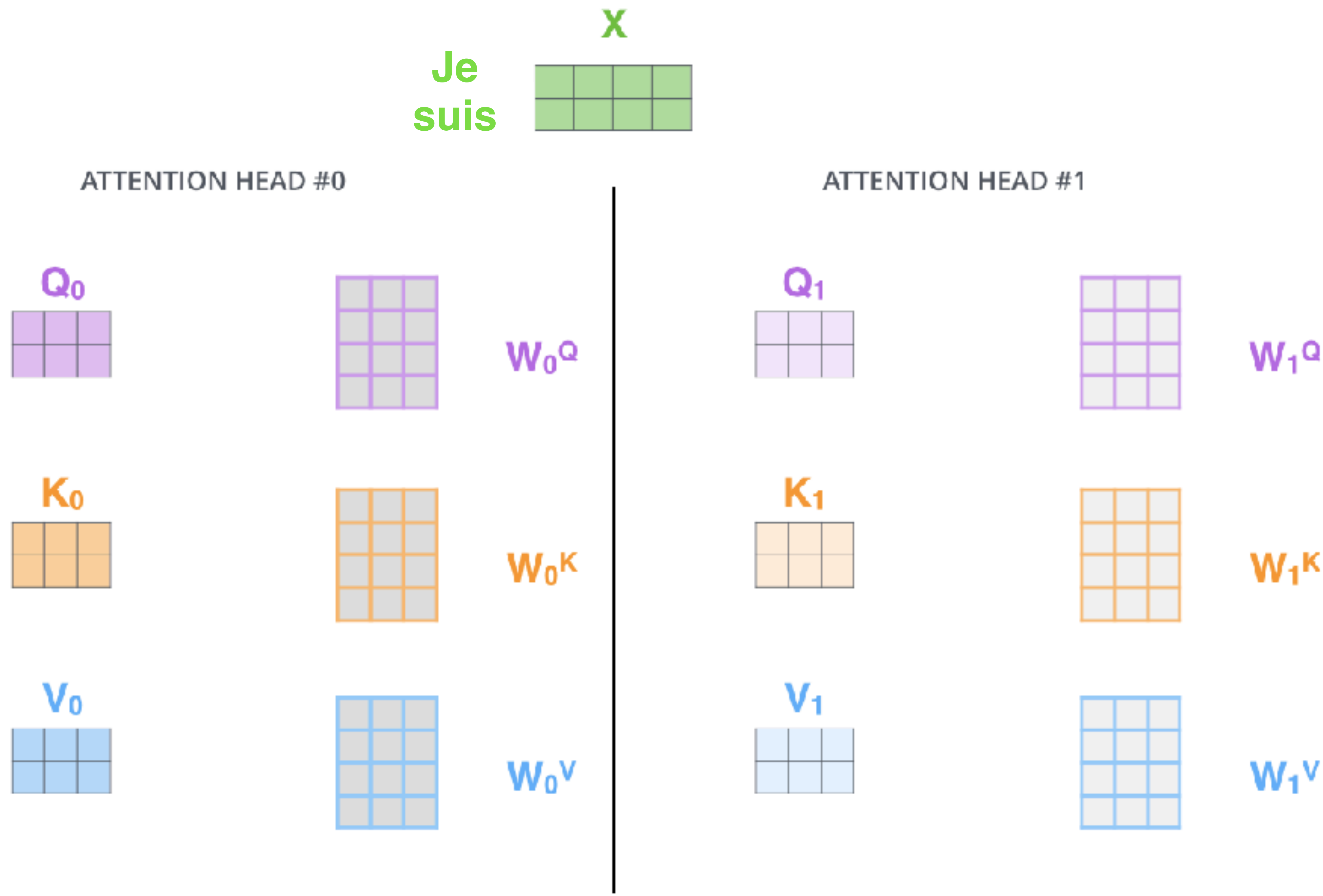


We got here using a single attention "head" and the parameters that define it in the matrices W^Q, W^K, and W^V.

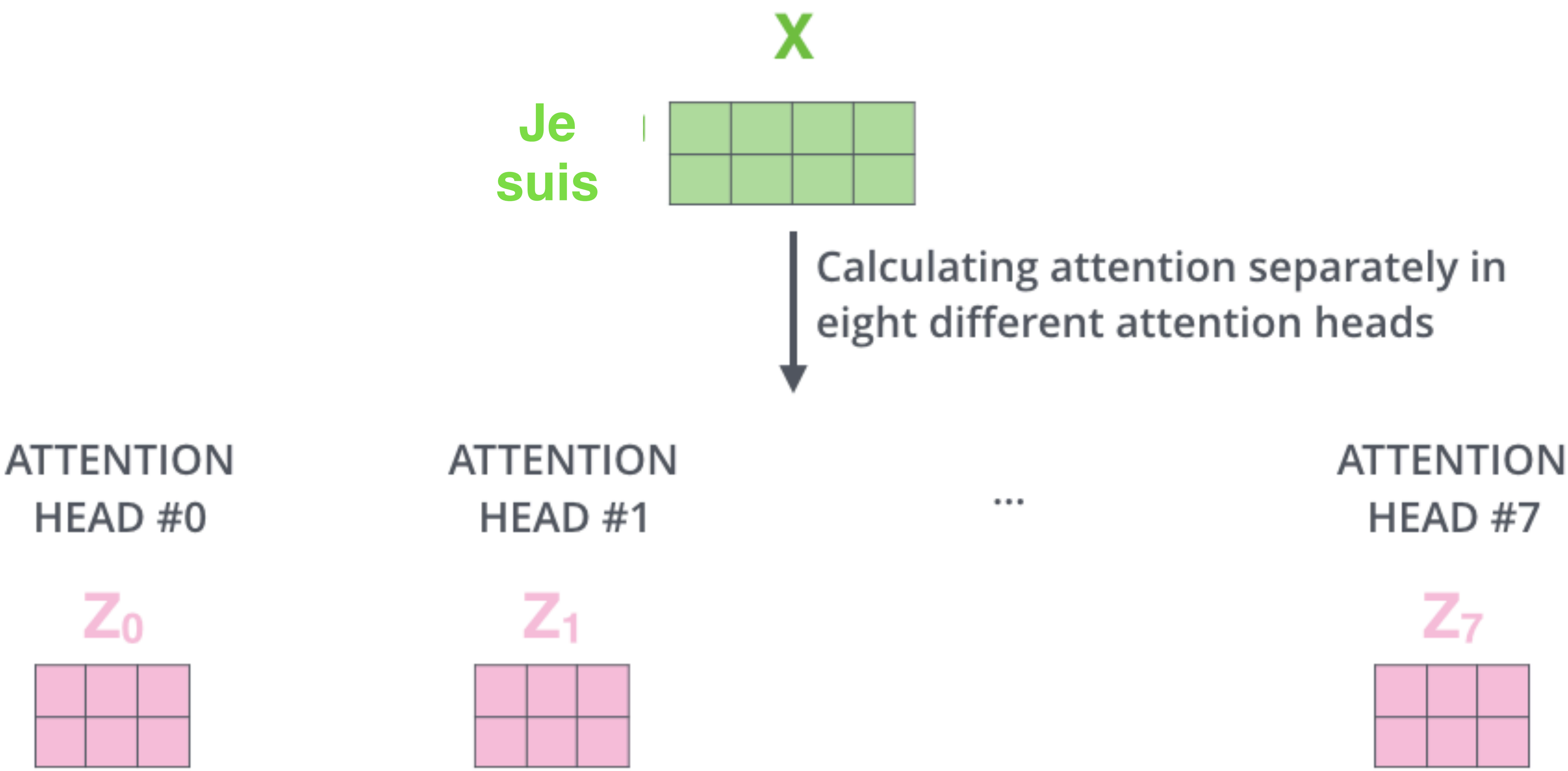
Queries, Keys, and Values



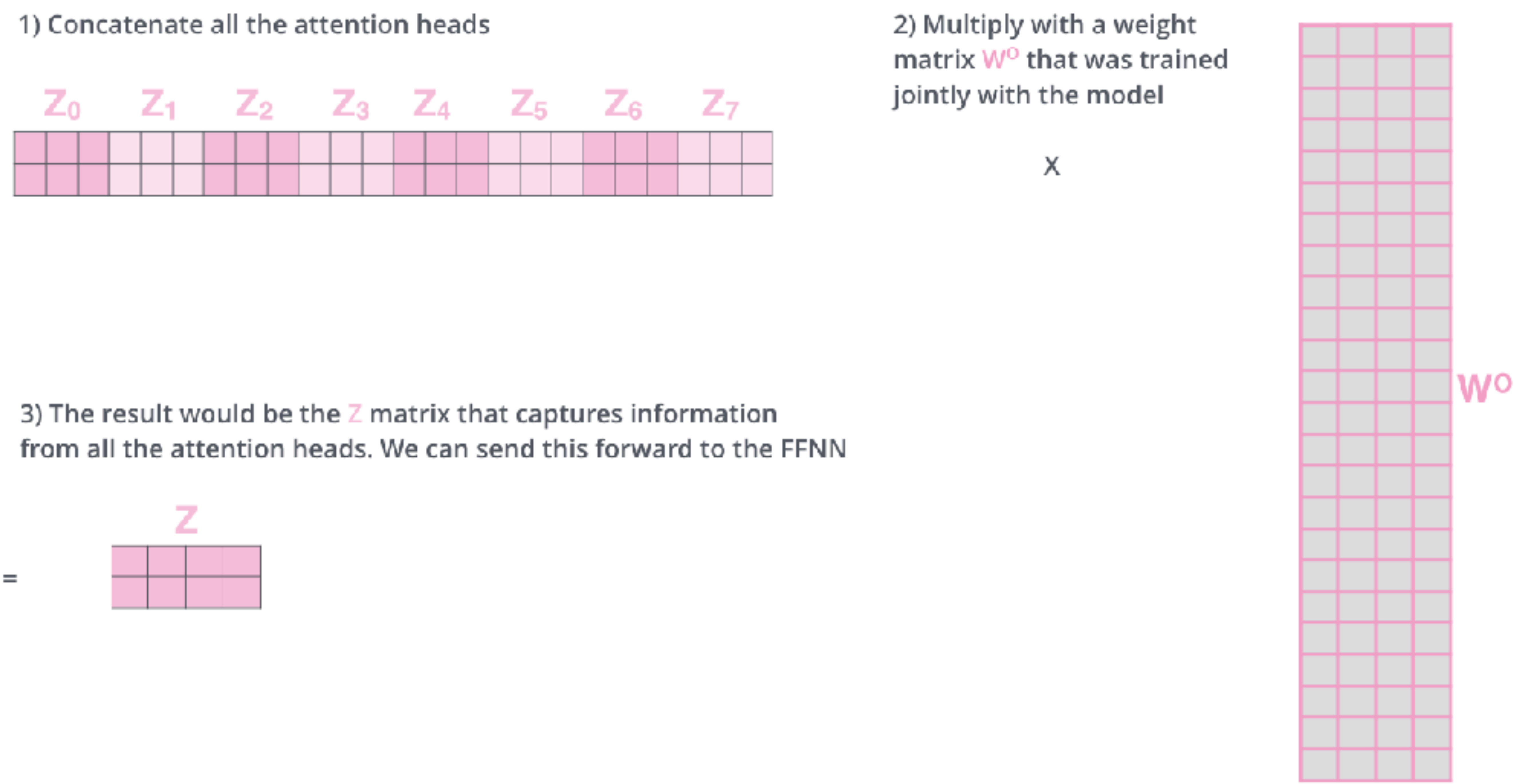
Multi-Headed Self-Attention



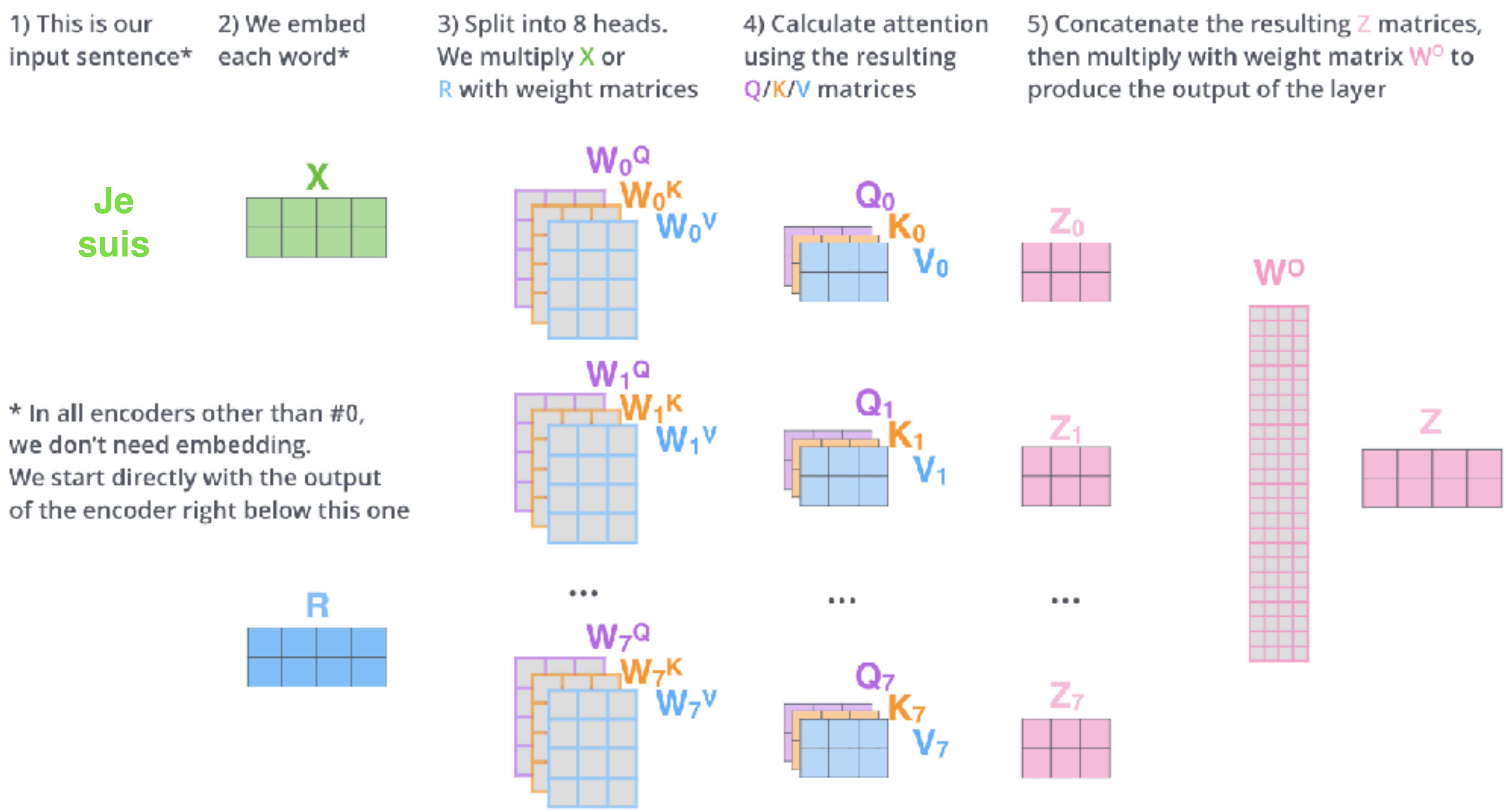
Multi-Headed Self-Attention



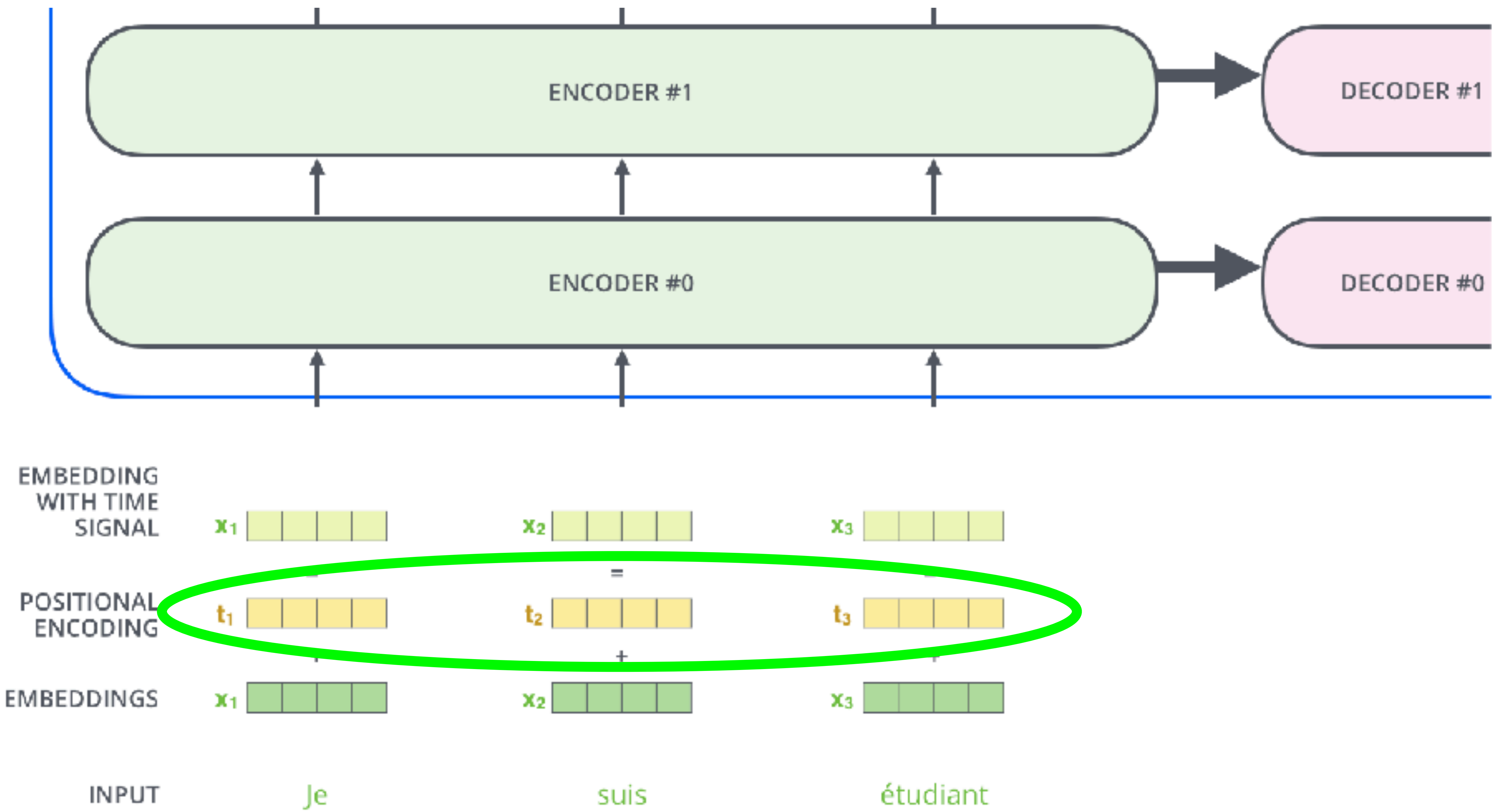
Multi-Headed Self-Attention



Multi-Headed Self-Attention



Input embeddings also have positional encodings



Positional encoding is naturally a fcn of position

$$\mathbf{p}(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \\ \vdots \\ p_d(t) \end{bmatrix}_{dx1}$$

d is embedding dimension

t is the position of the word in the sequence

Transformers use sinusoidal position encodings

$$\mathbf{p}(t) = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin(\omega_{d/2} t) \\ \cos(\omega_{d/2} t) \end{bmatrix}_{dx1}$$

where $\omega_k = \dfrac{1}{10000^{\frac{2k}{d}}}$

t is the position of the word in the sequence

Transformers use sinusoidal position encodings

Frequency decreases with increasing k

$$\mathbf{p}(t) = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin(\omega_{d/2} t) \\ \cos(\omega_{d/2} t) \end{bmatrix}_{dx1}$$

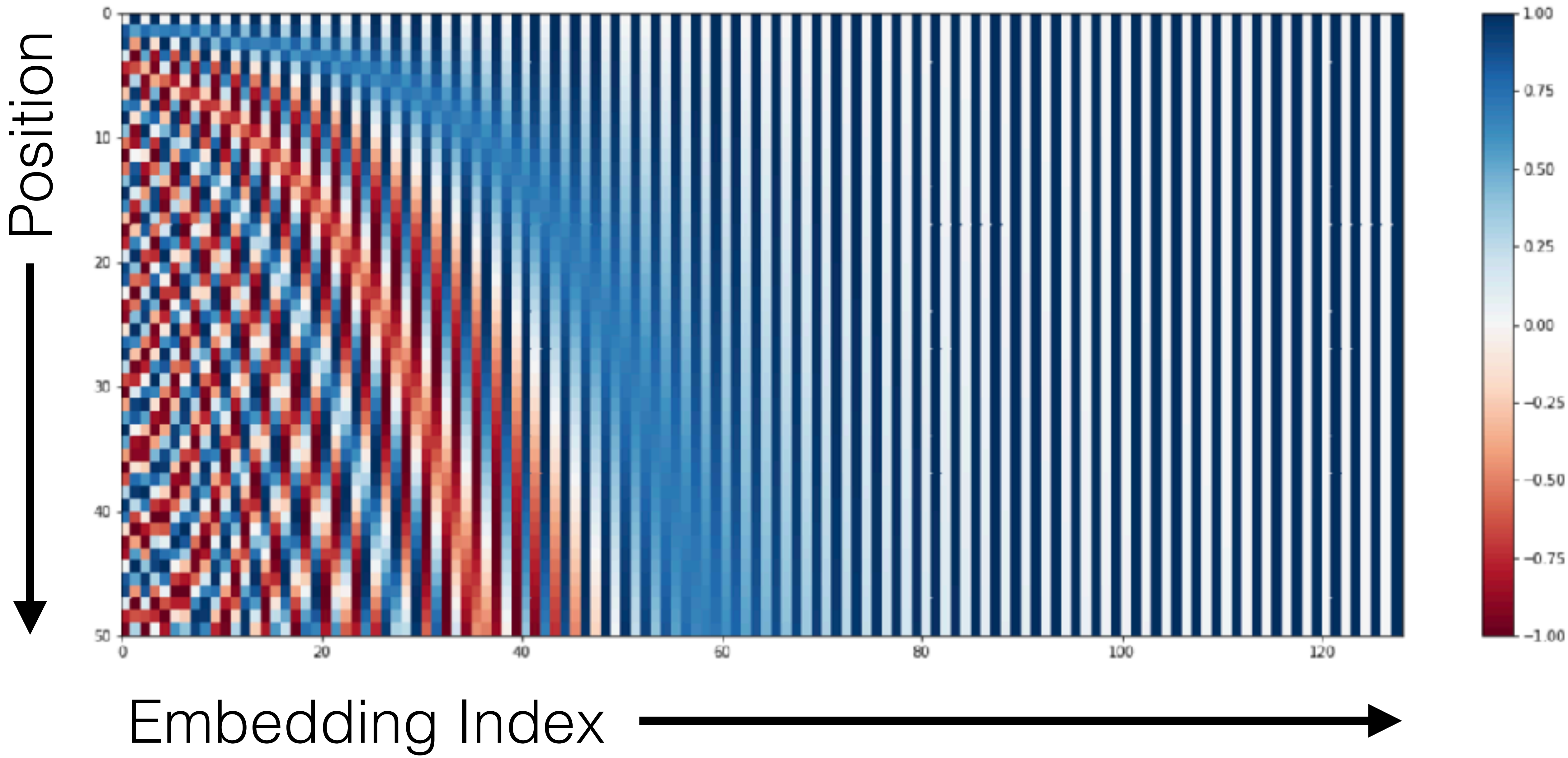
where $\omega_k = \dfrac{1}{10000^{\frac{2k}{d}}}$

Transformers use sinusoidal position encodings

$$\mathbf{p}(t) = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin(\omega_{d/2} t) \\ \cos(\omega_{d/2} t) \end{bmatrix}_{dx1}$$

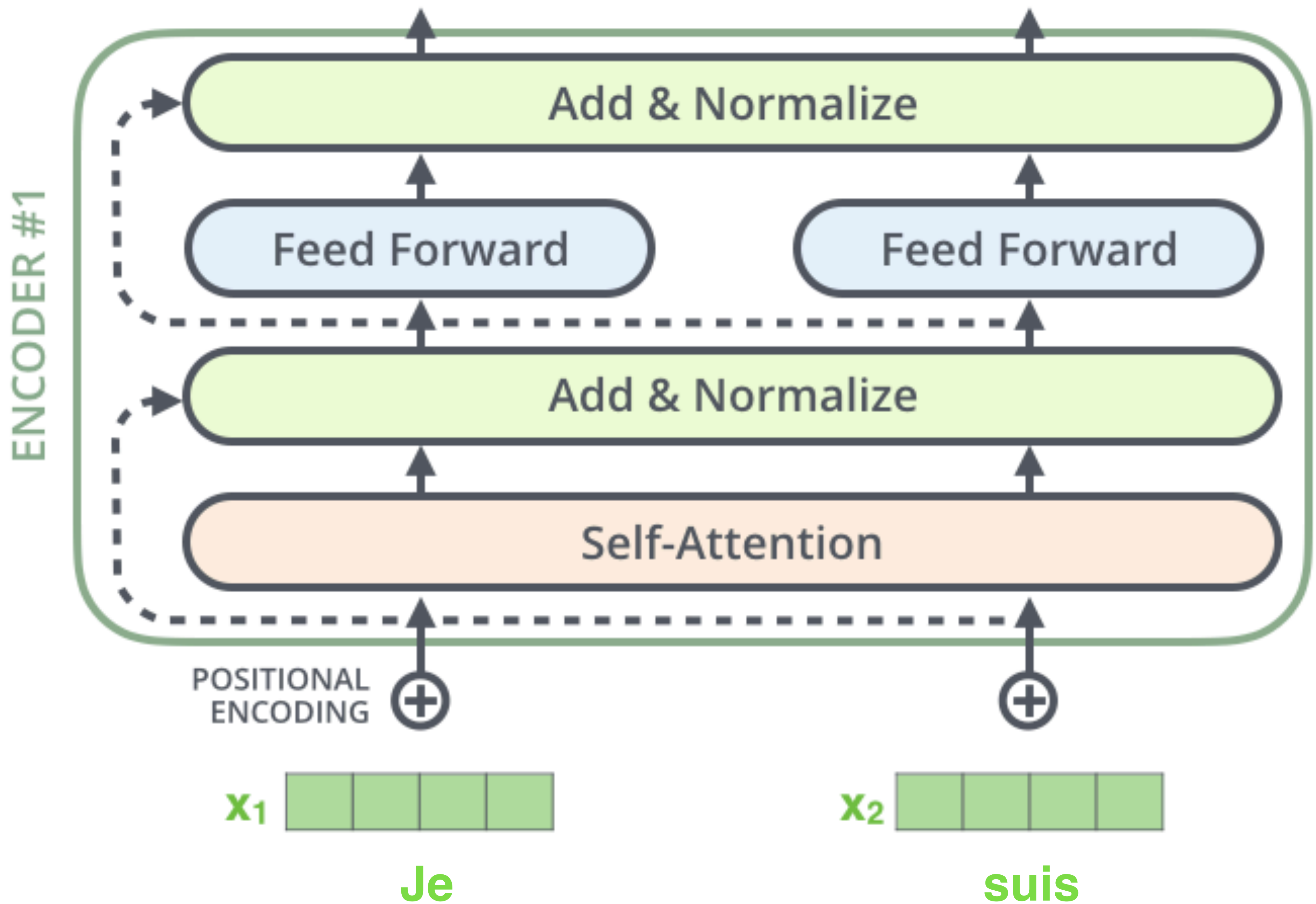
Each $\mathbf{p}(t)$ samples the same family of sinsusoidal fcns

Sinusoidal Position Encodings

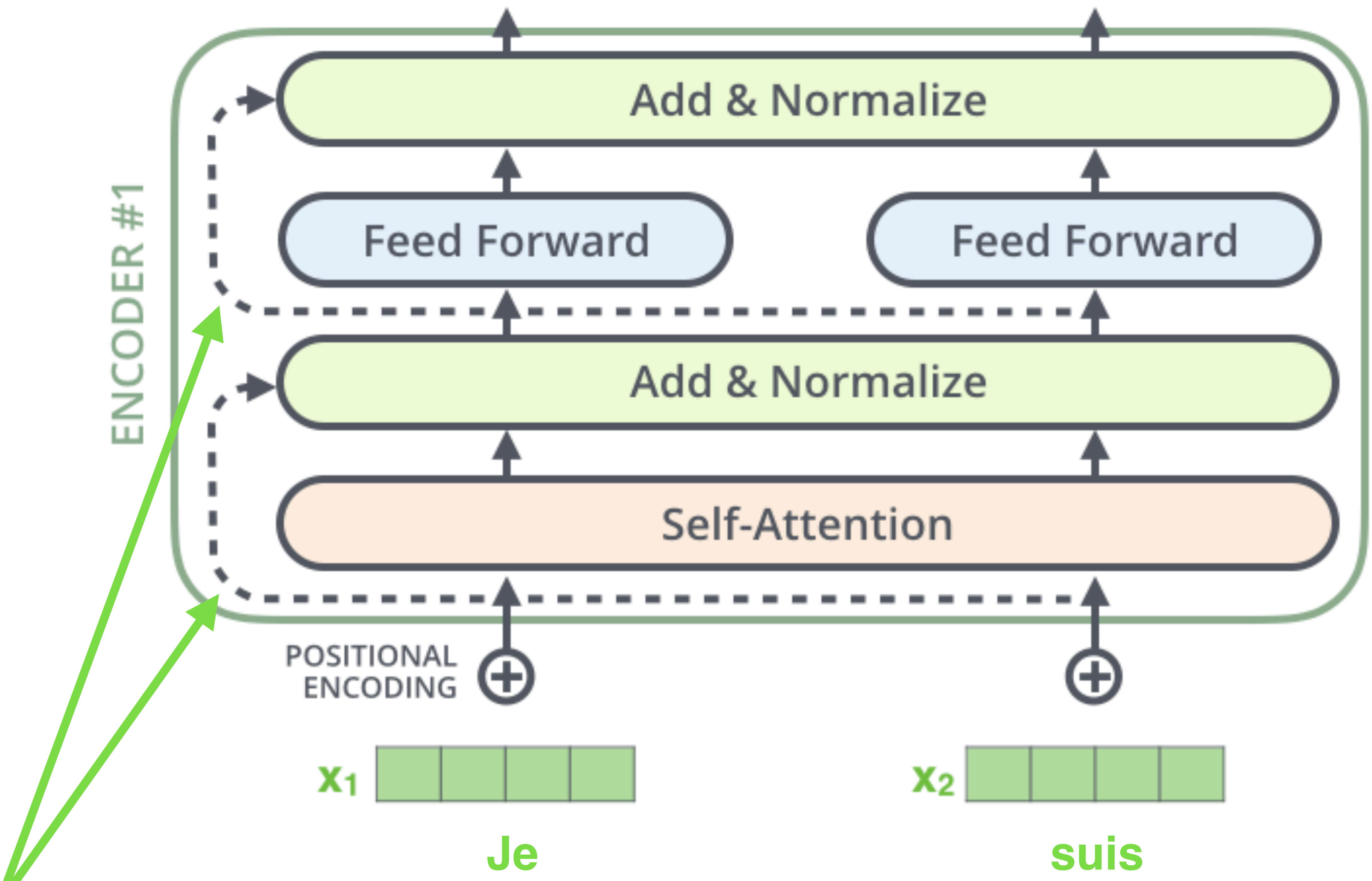


Each row is a positional encoding for a given position t

Residual Connections and Layer Normalizations

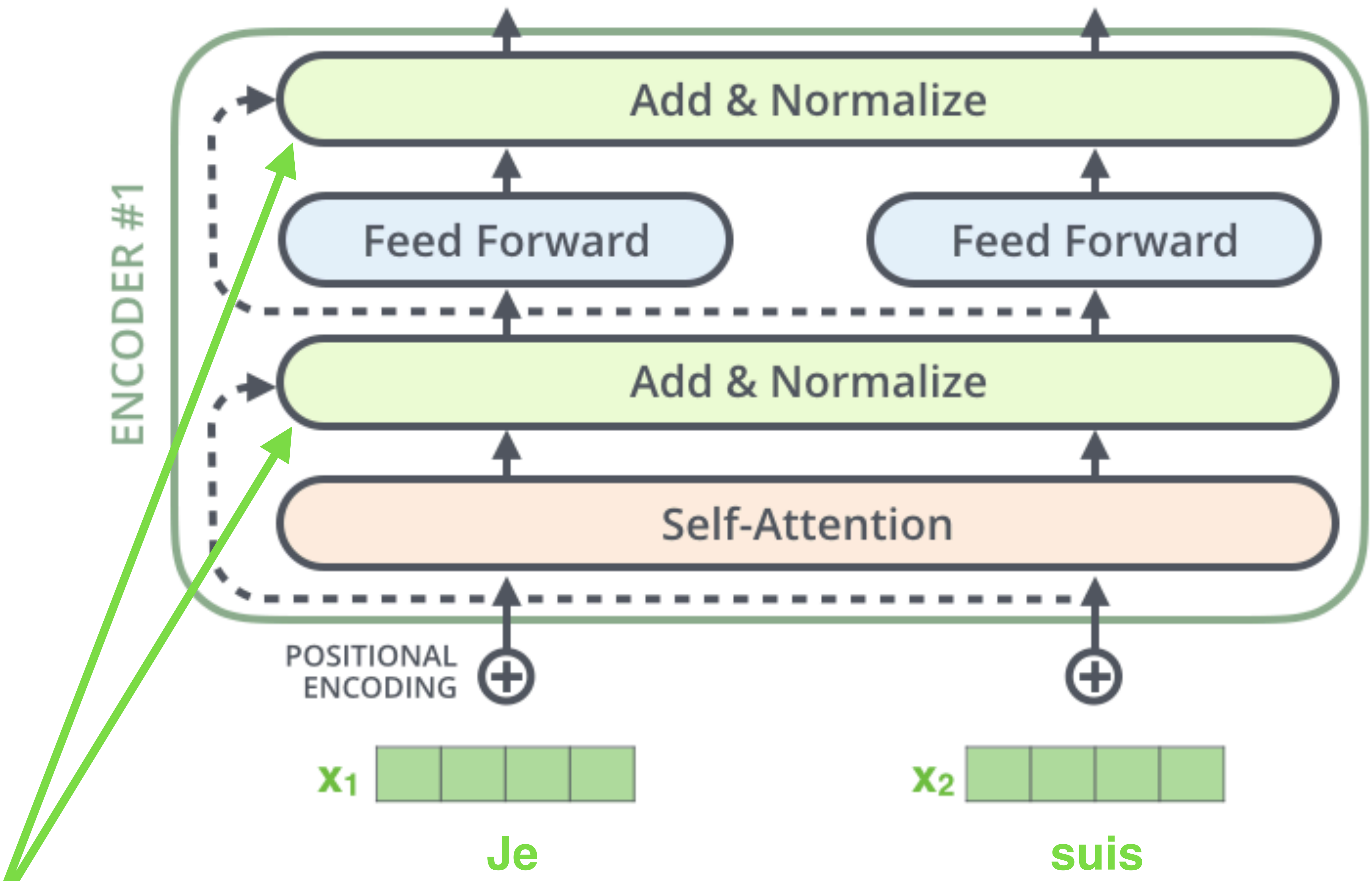


Residual Connections and Layer Normalizations



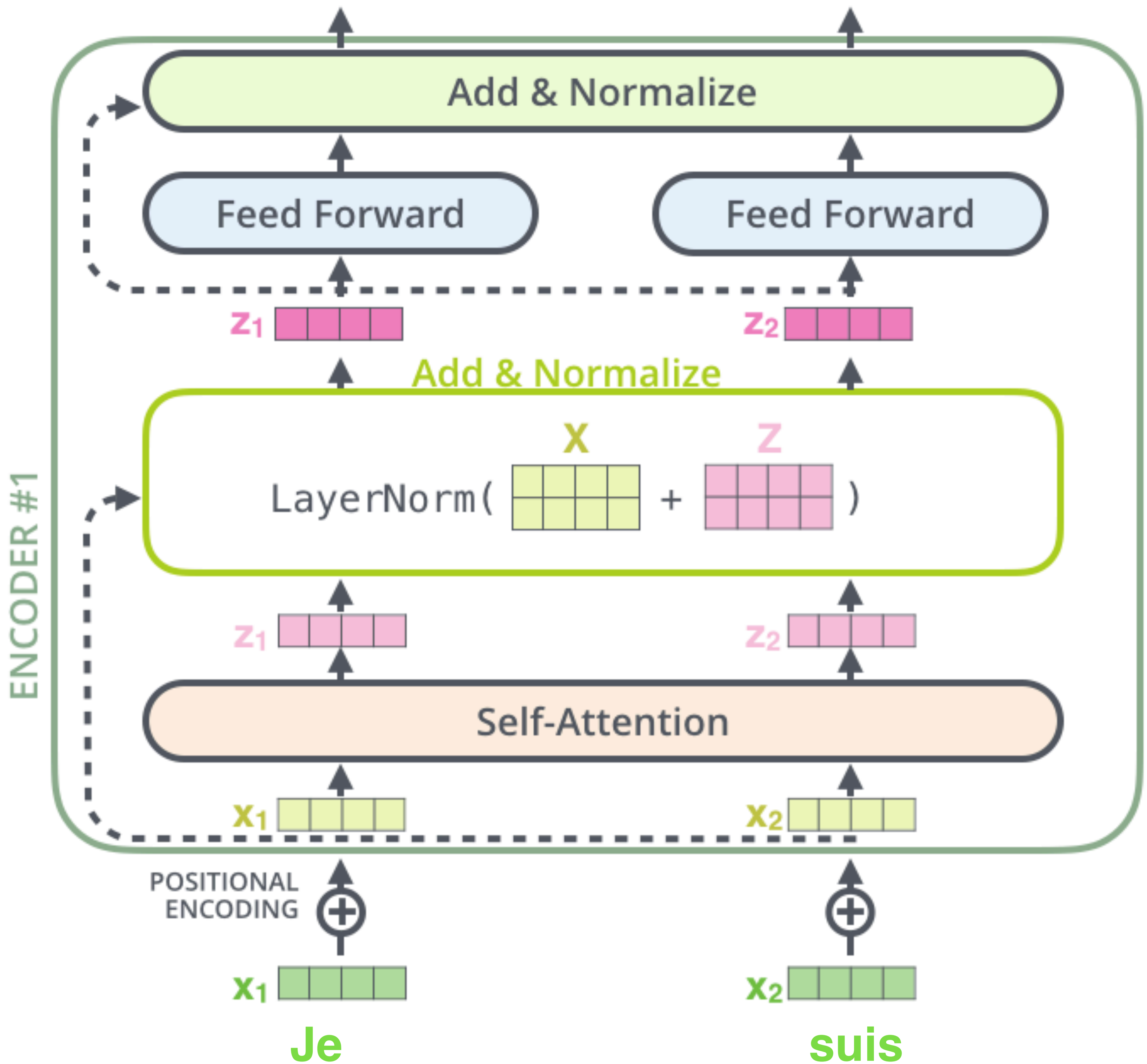
Residual connections are used to help gradient flow.

Residual Connections and Layer Normalizations

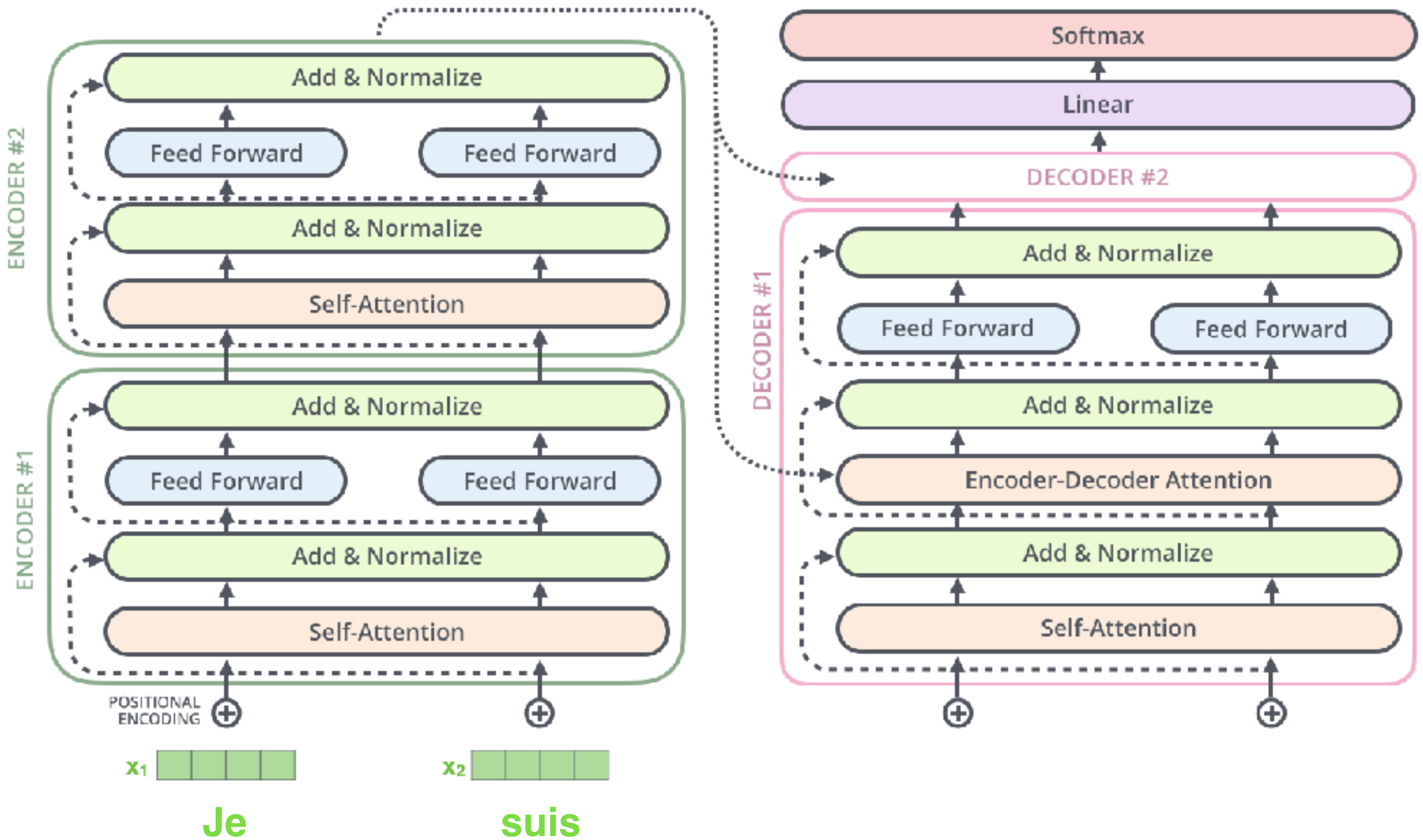


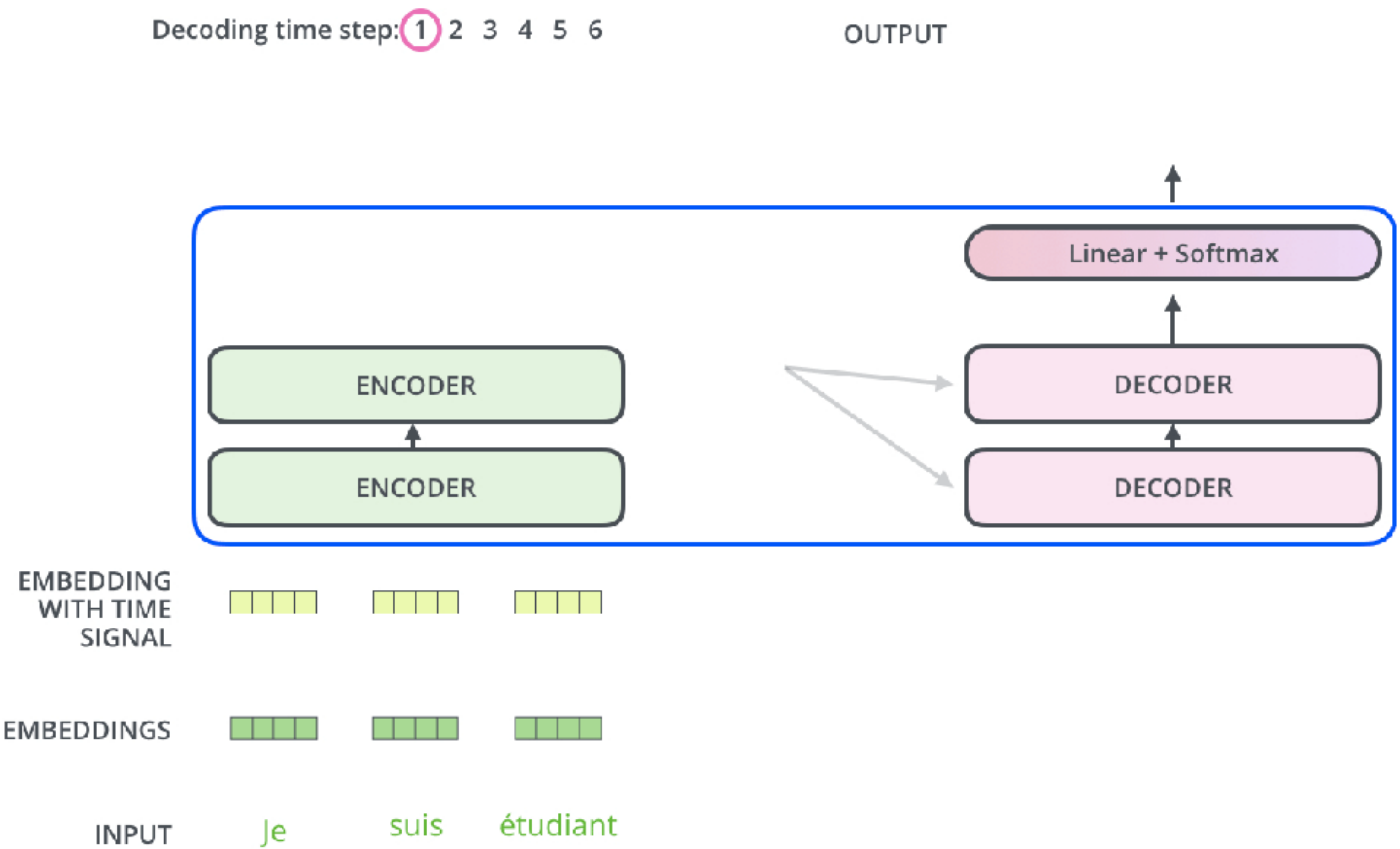
The outputs of the self-attention and ffnn layers are normalized.

Residual Connections and Layer Normalizations

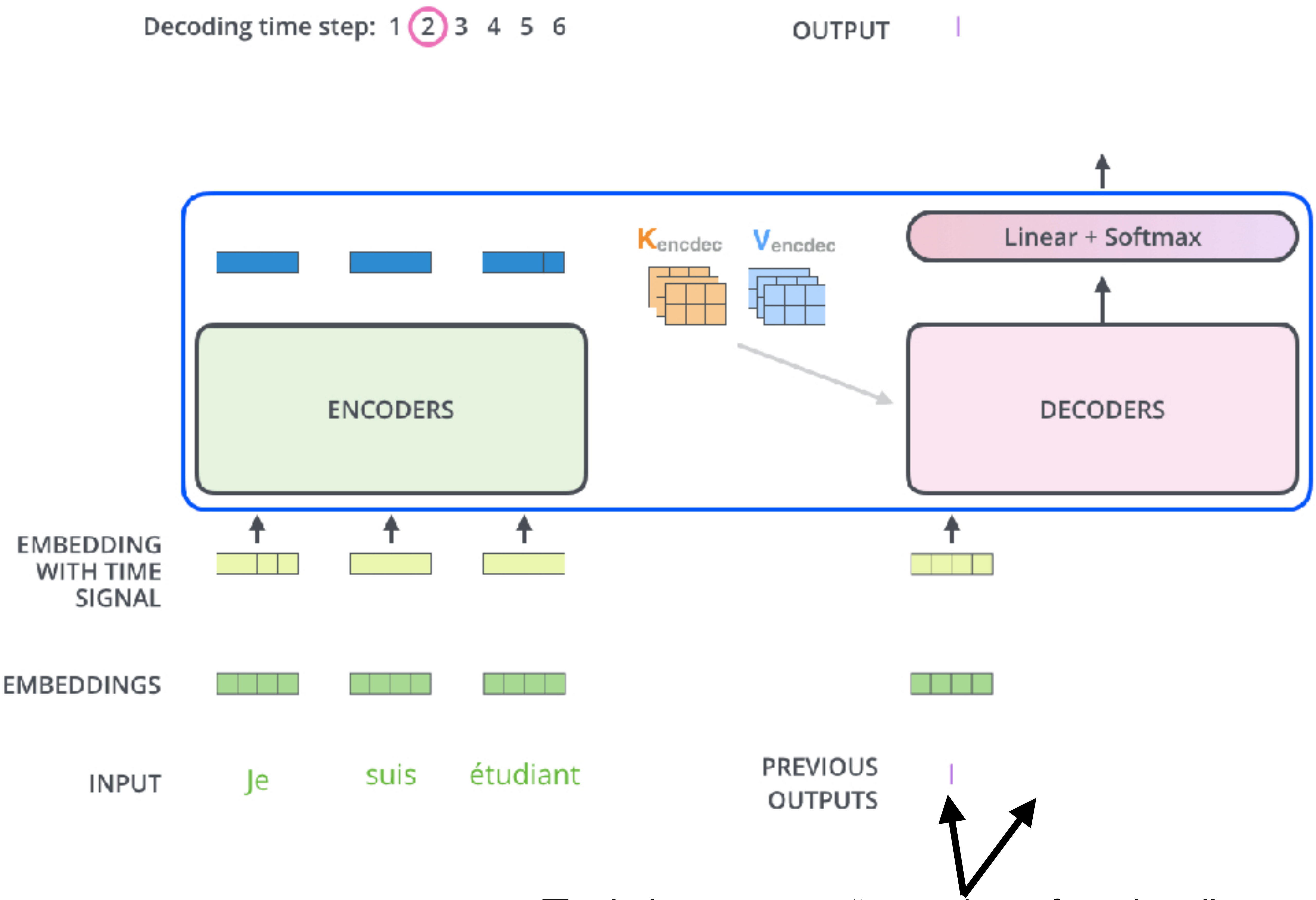


Transformer: Encoders and Decoders





Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant



Training uses “teacher forcing”

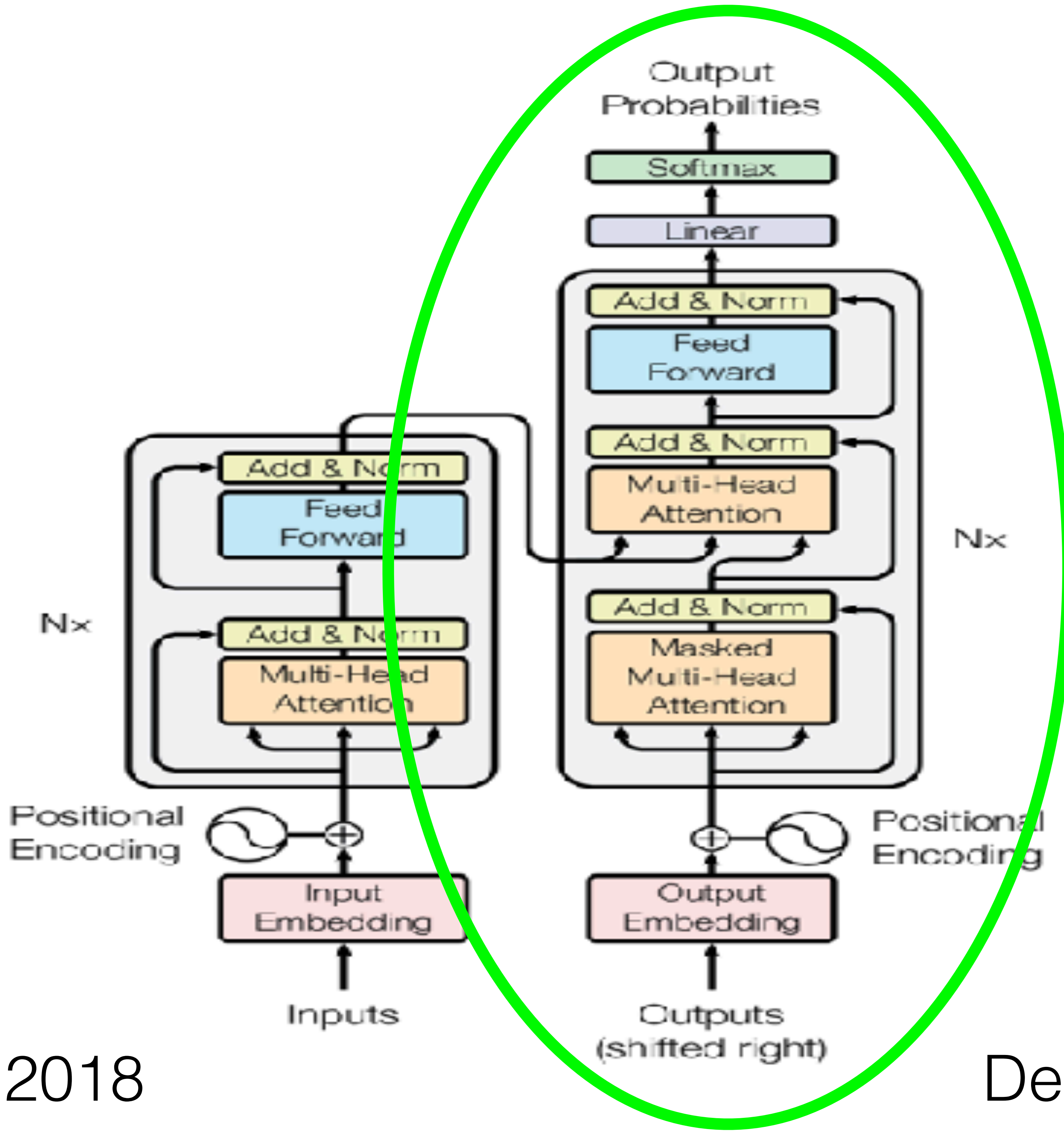
GPT-x:

Generative Pre-Training Transformer

Radford et al. 2018

GPT-x: Generative Pre-Training Transformer

OpenAI

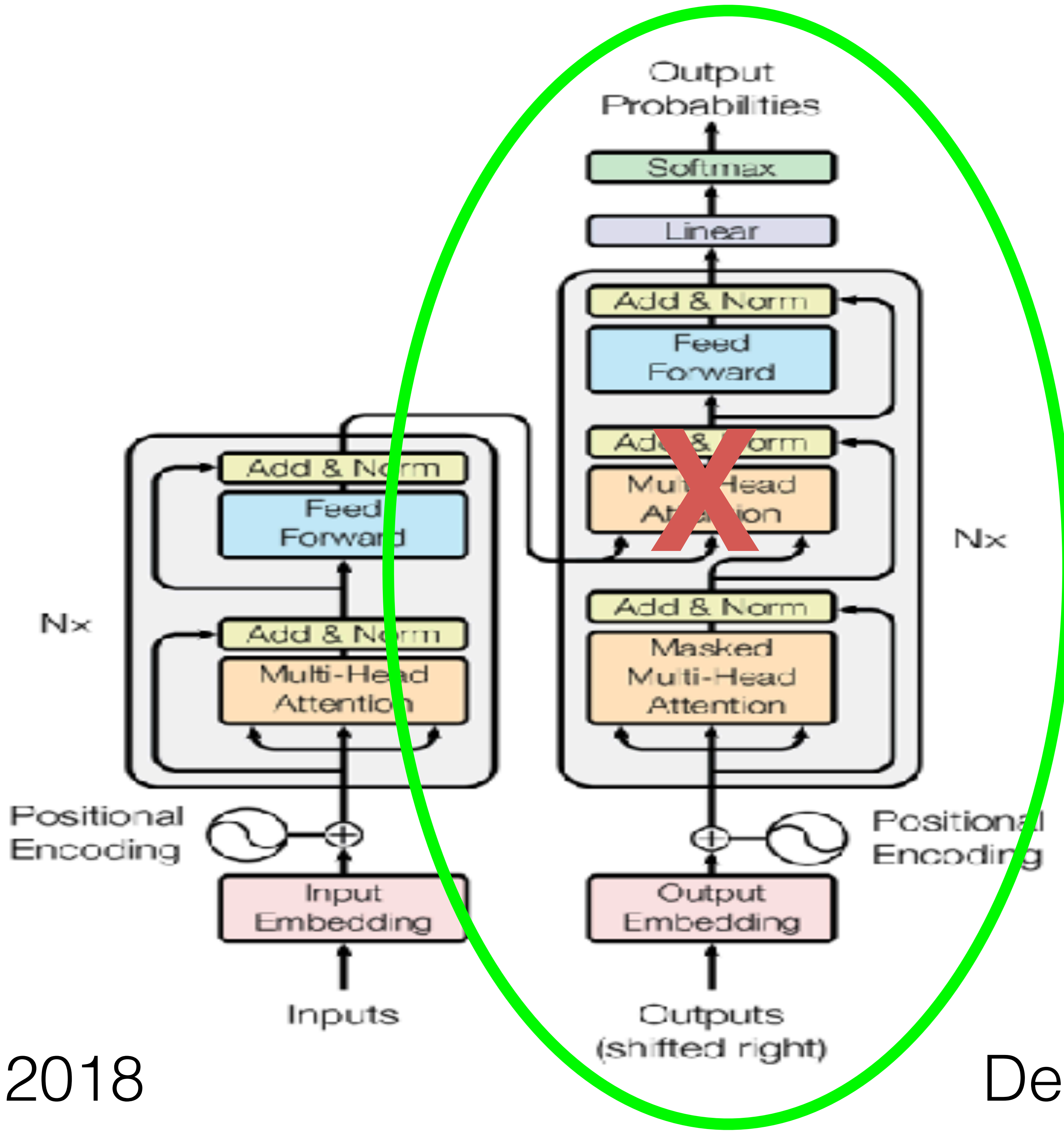


Radford et al. 2018

Decoder only

GPT-x: Generative Pre-Training Transformer

OpenAI



Radford et al. 2018

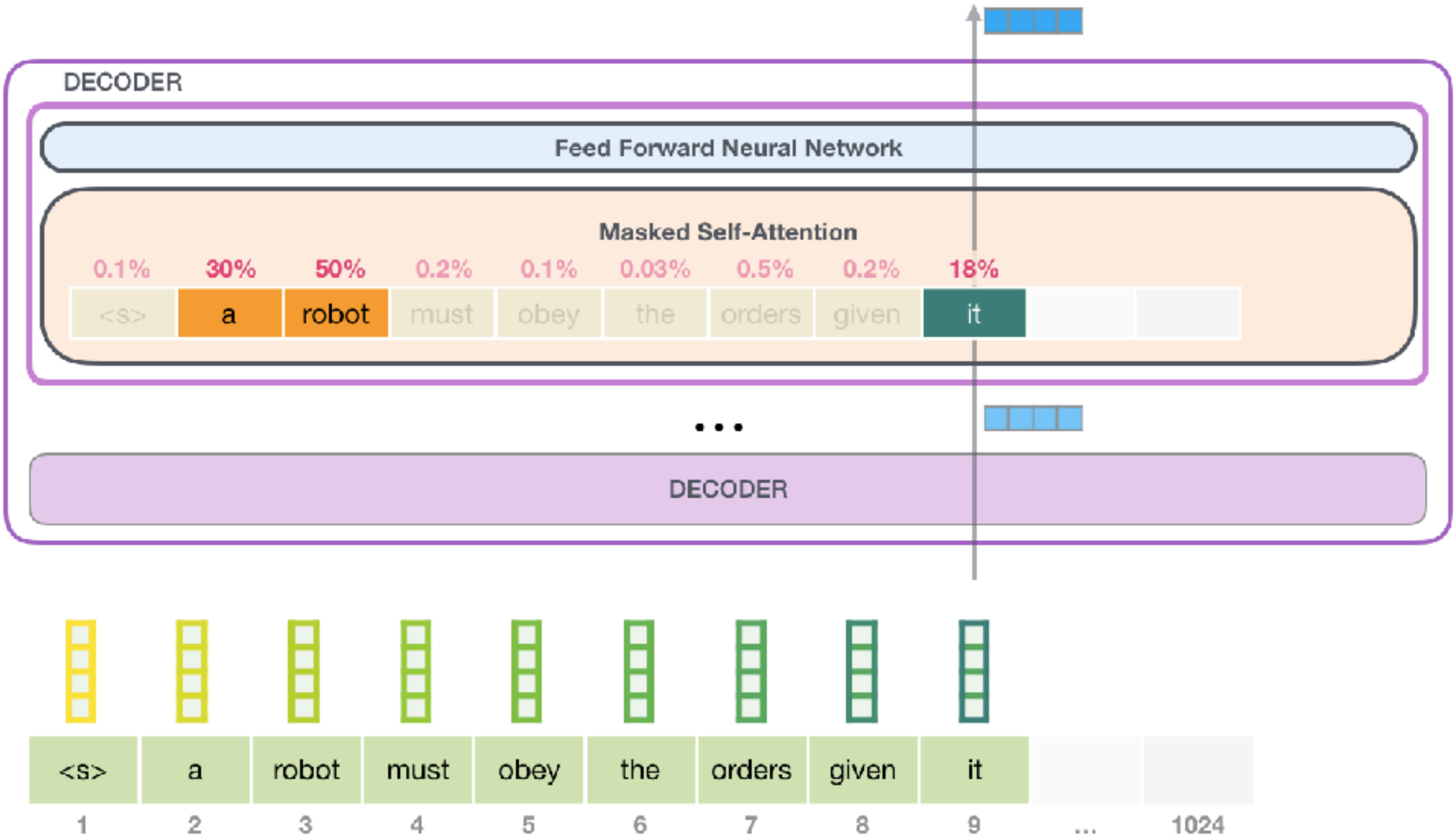
Decoder only

Causal Language Model

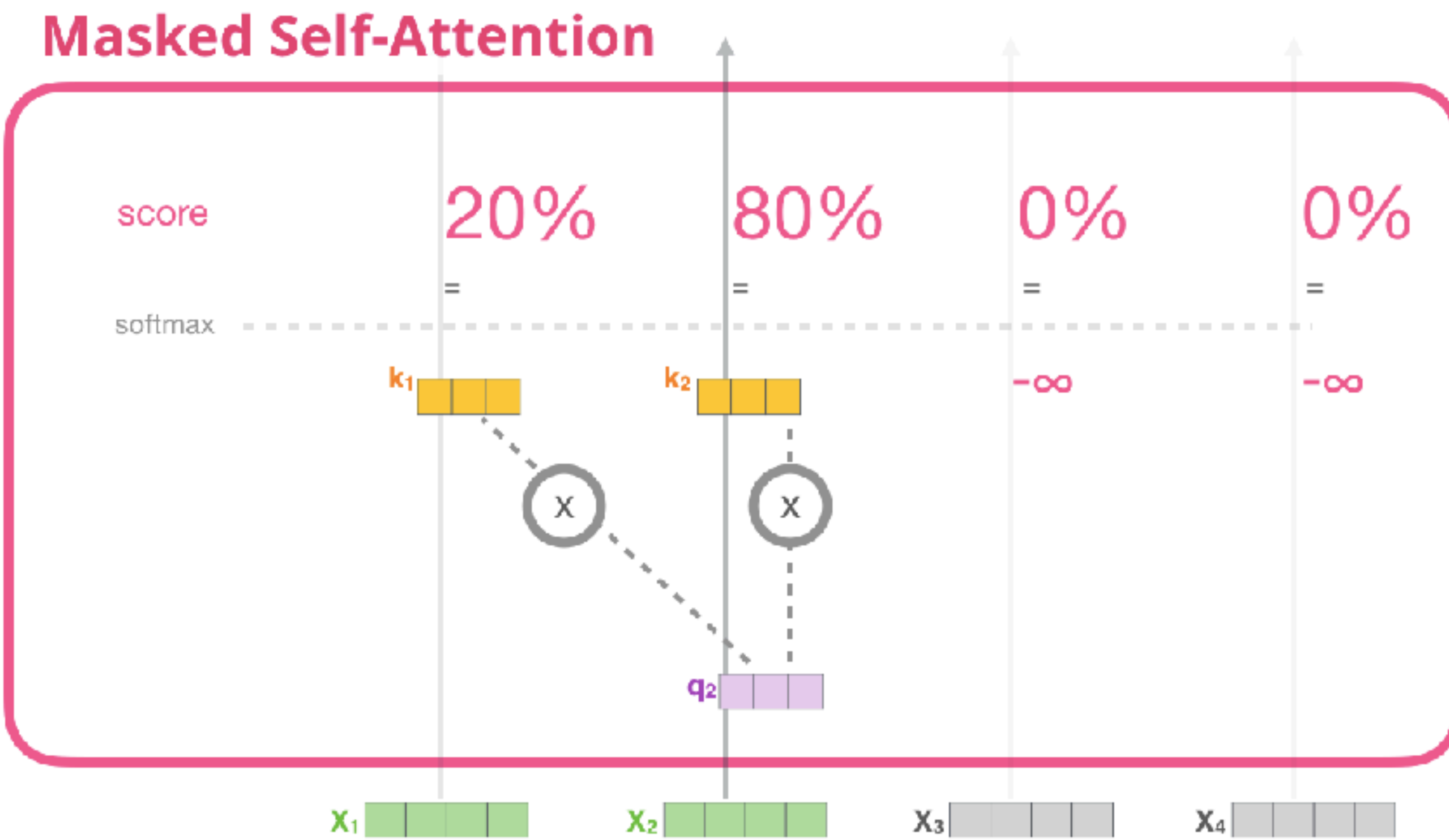
Consider a text sample as **sequence of words/tokens** $w_1, ..., w_n$, a causal language model p_θ computes cross entropy loss as:

$$L(\theta) = -\sum_{t=1}^{N} \log p_\theta(w_t | w_1, ..., w_{t-1})$$

GPT-x: Next Word Prediction



GPT-x: Masked Self-Attention



GPT-3: Training Data

Datasets	Quantity (Tokens)	Weight in Training Mix	Epochs elapsed when training for 300 BN tokens
Common Crawl (filtered)	410 BN	60%	0.44
WebText2	19 BN	22%	2.90
Books1	12 BN	8%	1.90
Books2	55 BN	8%	0.43
Wikipedia	3 BN	3%	3.40

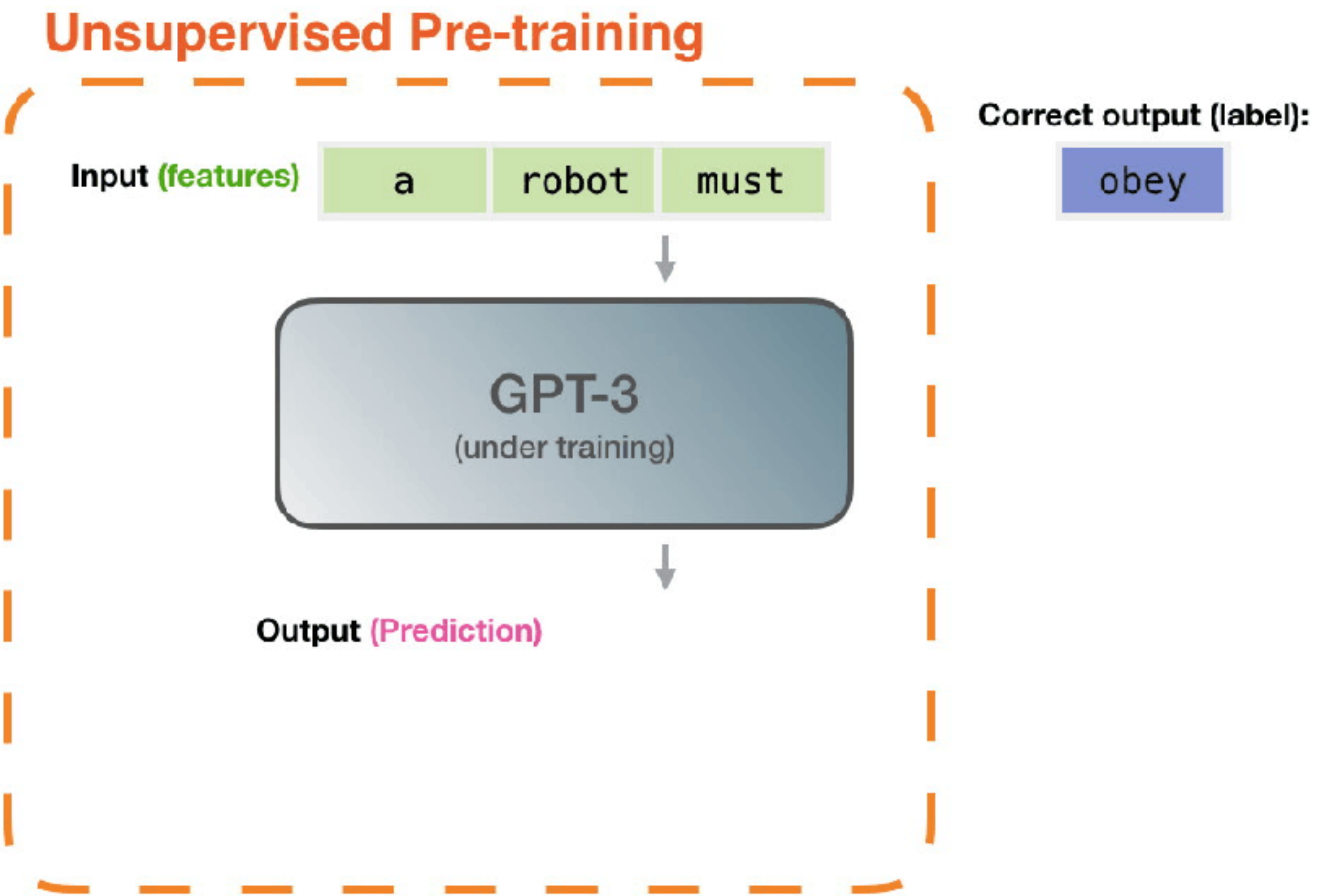
GPT-3: Training Data Samples

Text: Second Law of Robotics: A robot must obey the orders given it by human beings

Generated training examples

Example #	Input (features)	Correct output (labels)
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must
...		

GPT-3: Loss Computation



GPT-3: Loss Computation

$$L(\theta) = -\log p_{\theta}(< obey > \mid < a > , < robot > , < must >)$$

Uses standard cross-entropy loss: negative log likelihood of the correct token/word

GPT-3: Model Variants

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

GPT-3: Model Variants

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

GPT-3: Open AI API

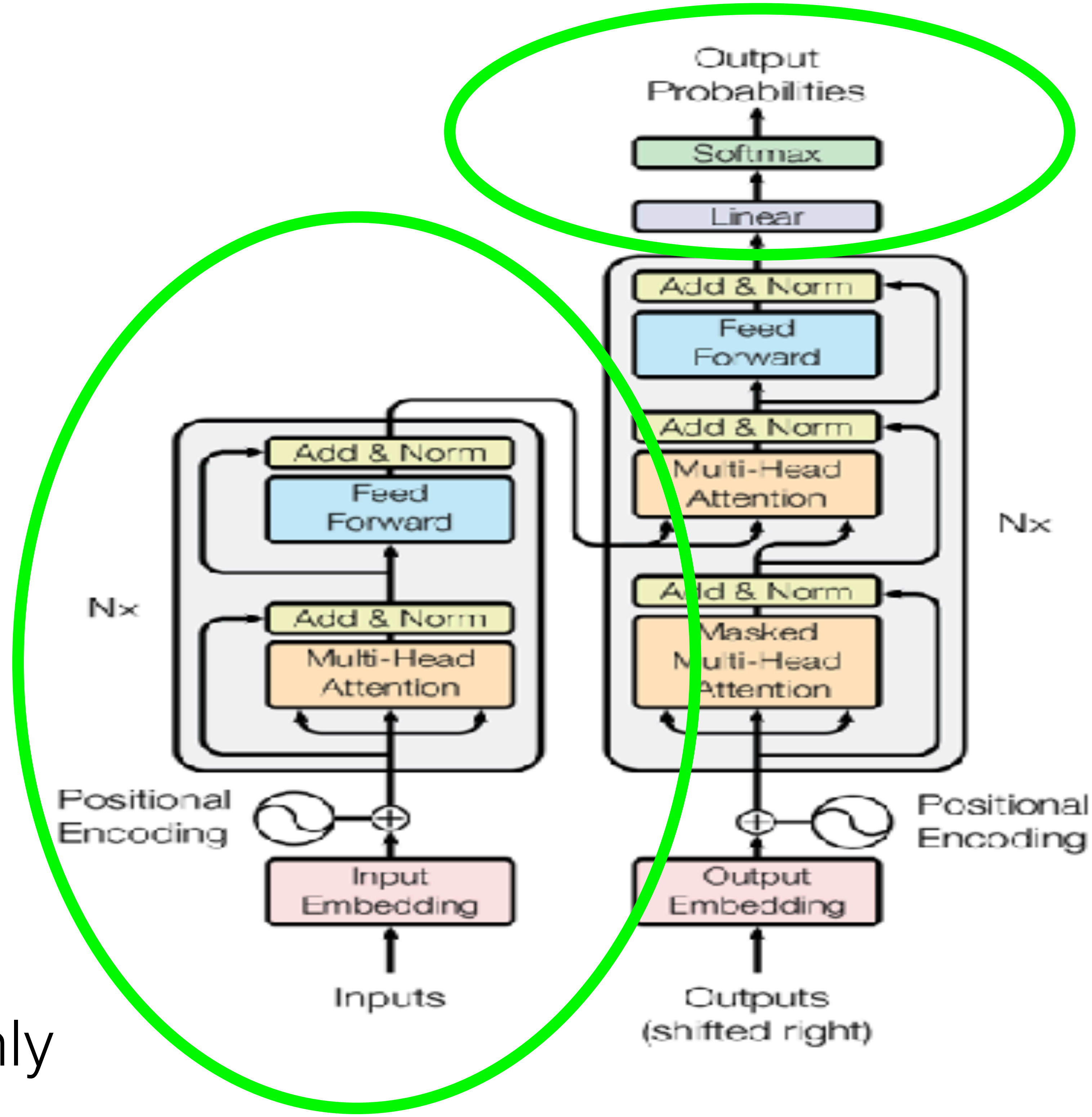
https://openai.com/api/

BERT:

Bidirectional Encoder Representations from Transformers

Devlin et al. 2019

BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers



Encoder only

BERT: Masked Language Model

- Designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.
- Trained using a ‘Masked Language Model’ (MLM) pre-training objective.
- MLM training is followed by training on ‘Next Sentence Prediction’ (NSP) task.

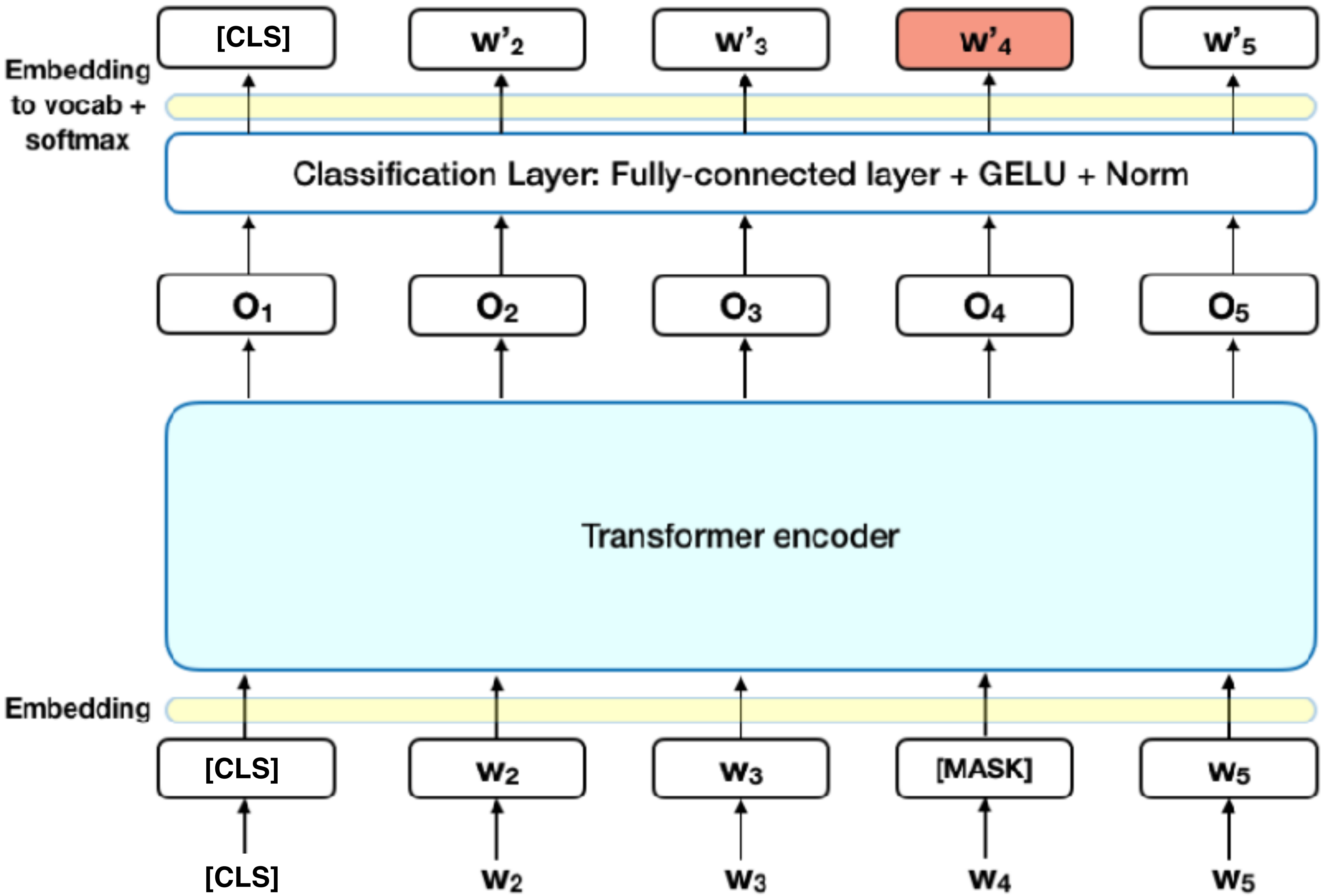
BERT: Pre-training Data

- BooksCorpus (800M Words)
- English Wikipedia (2500M Words)

BERT: Masked LM

- 15% of the words in the input sequence are masked randomly.
- A chosen token to be masked, is masked with the following tokens:
  - [MASK] - 80% of the time.
  - Random Token - 10% of the time.
  - Same Token - 10% of the time.
- Trained with Cross-Entropy loss.

BERT


[CLS]
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
[CLS]
w2
w3
[MASK]
w5
[CLS]
w2
w3
w4
w5

BERT: Next Sentence Prediction

- Sentence A and B chosen as sentence pair.
- Sentence pairs separated by [SEP] fed to the network.
- 50% of the time B is the actual next sentence that follows A. Labeled as ‘IsNext’.
- 50% of the time it is a random sentence. Labeled as ‘NotNext’.

BERT: Architecture

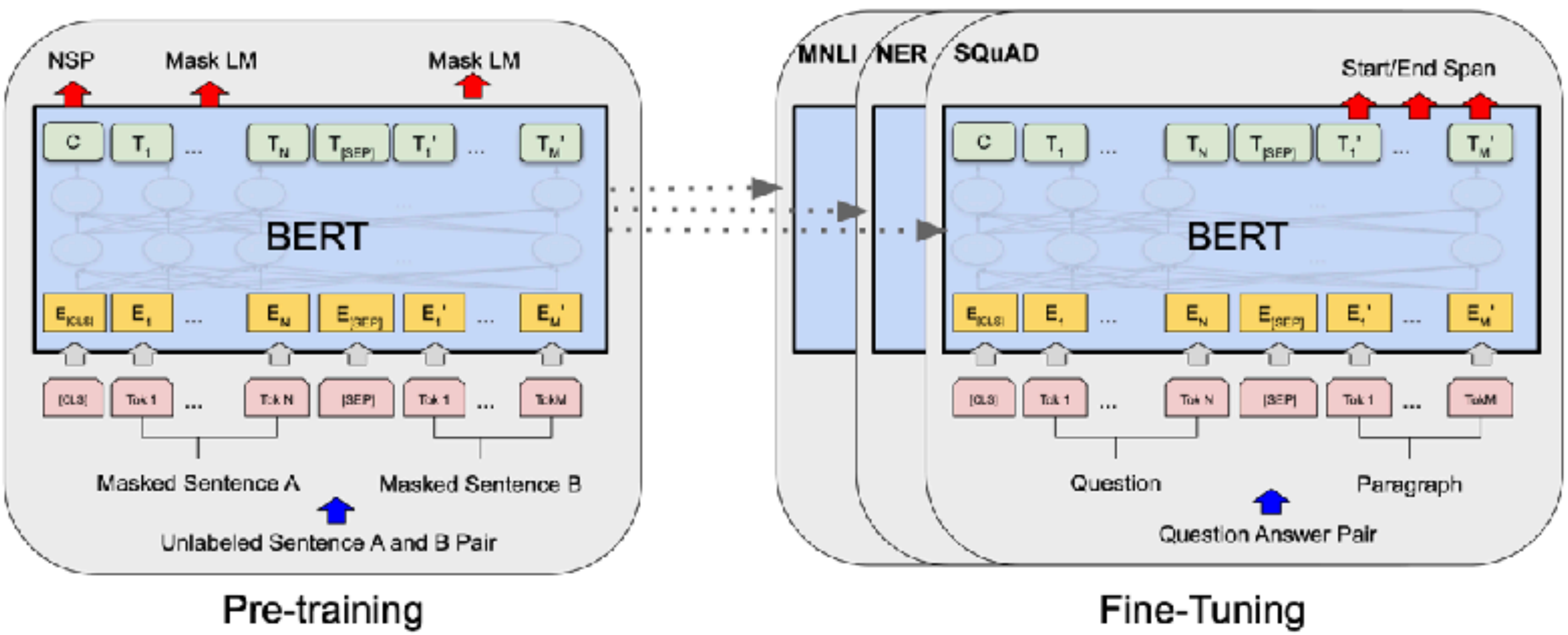


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

BERT: Architecture

BERT (Base)

- 12 Transformer Blocks (L=12)
- 768 dim. Hidden Embed. Size (H=768)
- 12 Self-Attention Heads (A=12)
- 110M parameters

BERT: Input Representation

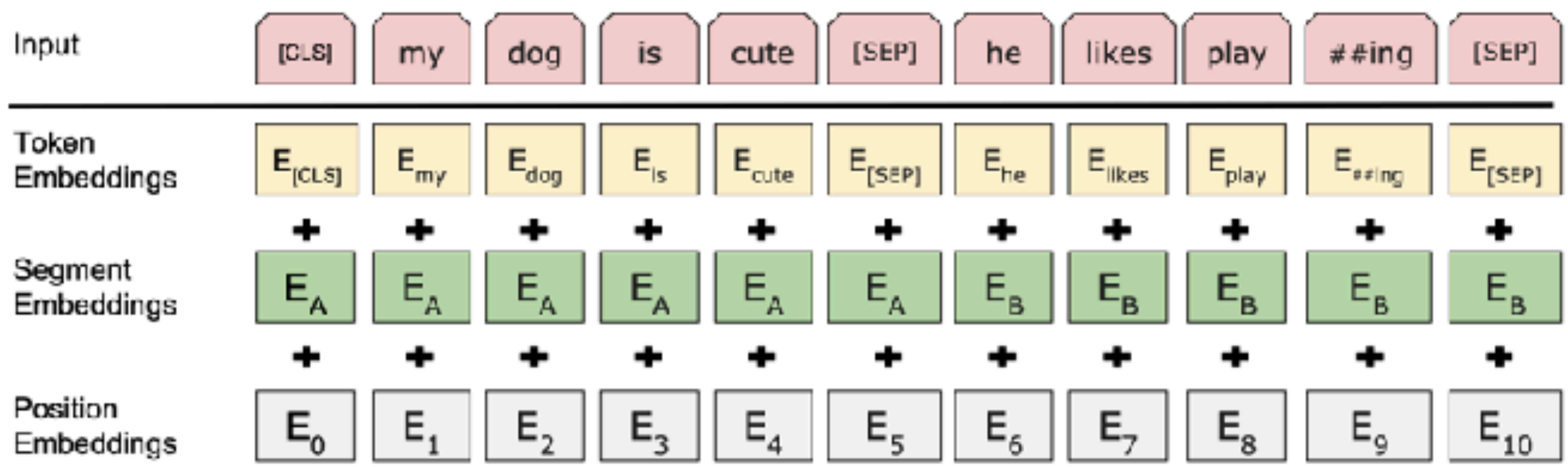


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT: Feature Extraction

- We can extract features from intermediate layers of the model.
- Although fine-tuning is recommended, features can be used without training.
- Model returns token-level features.

Variants of BERT

I’ve enjoyed preparing these slides, but I’m running out of time.

Maybe I could get a Large Language Model (LLM) like OpenAI’s ChatGPT to explain the variants of BERT…

ChatGPT on Variants of BERT

Let's ask ChatGPT about variants of the BERT model:

PB What are the many different variants of the large language model BERT?

BERT (Bidirectional Encoder Representations from Transformers) is a language model that was developed by researchers at Google in 2018. It has since been improved and extended by the research community, leading to the development of many different variants of the original BERT model. Some of the most commonly used variants of BERT include BERT-Large, BERT-Base, and BERT-Base Uncased.

Asking ChatGPT more directly in follow up question:

PB

Are there other variants like RoBERTa and if so could you list them?

Yes, there are other variants of BERT that have been developed by the research community, such as RoBERTa (Robustly Optimized BERT Pretraining Approach), ALBERT (A Lite BERT), and ELECTRA (Efficiently Learning an Encoder that Classifies Tokens Accurately). These variants of BERT have been designed to improve upon the original BERT model in various ways, such as by reducing the amount of training data required, increasing the model's efficiency, or improving its performance on certain tasks.

Asking ChatGPT for more detail on RoBERTa vs. BERT:

PB

What is the difference between BERT and RoBERTa?

BERT and RoBERTa are both large language models that use the Transformer architecture to process and generate text. However, there are several differences between the two models. BERT is a general-purpose language model that can be fine-tuned for a variety of natural language processing tasks, such as sentiment analysis and question answering. RoBERTa, on the other hand, is a variant of BERT that was specifically designed to improve performance on these tasks.

One of the main differences between BERT and RoBERTa is that RoBERTa was trained on a larger dataset and for a longer time than BERT, which makes it more accurate and more effective at natural language processing tasks. Additionally, RoBERTa uses a different training objective than BERT, which helps the model better capture the relationships between words in a sentence. These differences make RoBERTa a more powerful and effective language model than BERT for many natural language processing tasks.

ChatGPT previous answer was vague, pressing for details:

PB

How does RoBERTa's training objective differ from BERT's?

The training objective is the function that a machine learning model is trying to optimize during training. In the case of BERT and RoBERTa, both models are trained to maximize the likelihood of predicting the correct word in a sentence, given the words that come before and after it. However, the specific details of how this objective is implemented differ between the two models.

One of the main differences between BERT and RoBERTa's training objectives is that RoBERTa is trained using a larger dataset and for a longer time than BERT, which allows the model to better capture the relationships between words in a sentence. Additionally, RoBERTa uses a technique called dynamic masking, which randomly masks a different number of words in each training example. This makes the model more robust and allows it to better capture the meaning of longer sequences of words. These differences in the training objective help RoBERTa to perform better on natural language processing tasks than BERT.

What about the text-to-text T5 LLM:

PB

Ok, I get it. What about T5? How is T5 different from BERT?

T5 (Text-To-Text Transfer Transformer) is another large language model that was developed by researchers at Google in 2020. Like BERT, T5 uses the Transformer architecture to process and generate text, but there are several key differences between the two models.

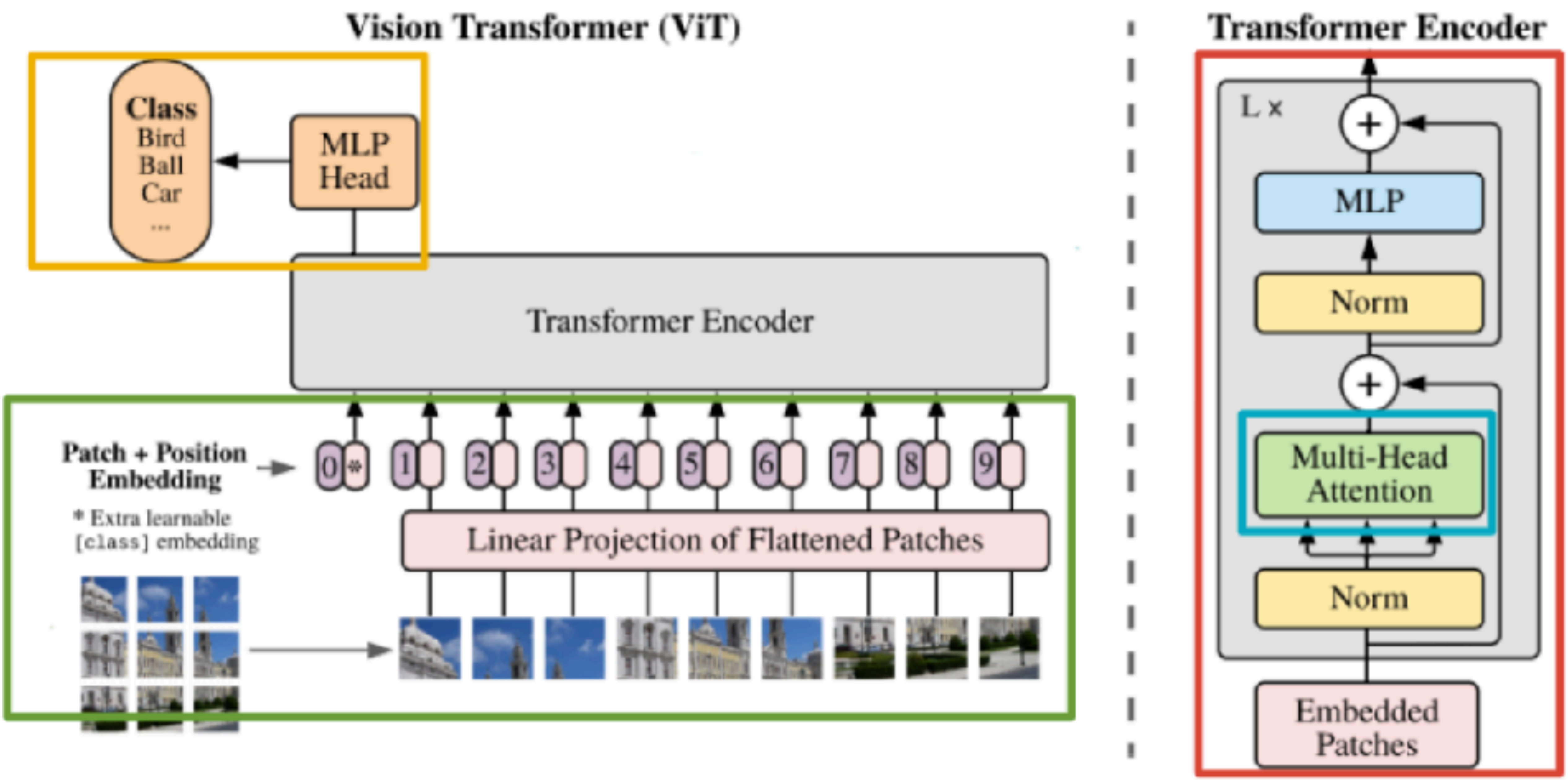
One of the main differences between T5 and BERT is that T5 is a general-purpose language model that can be fine-tuned for a wide range of natural language processing tasks, whereas BERT is primarily used for tasks such as sentiment analysis and question answering. This means that T5 can be used for a wider range of tasks than BERT, which makes it a more versatile and powerful language model.

Another key difference between T5 and BERT is the way they are trained. T5 is trained using a technique called transfer learning, which allows the model to learn from a large amount of unannotated text data. This makes T5 more data-efficient than BERT, which requires a large amount of labeled data to achieve good performance on natural language processing tasks. Additionally, T5 uses a different training objective than BERT, which helps the model better capture the meaning and structure of natural language. These

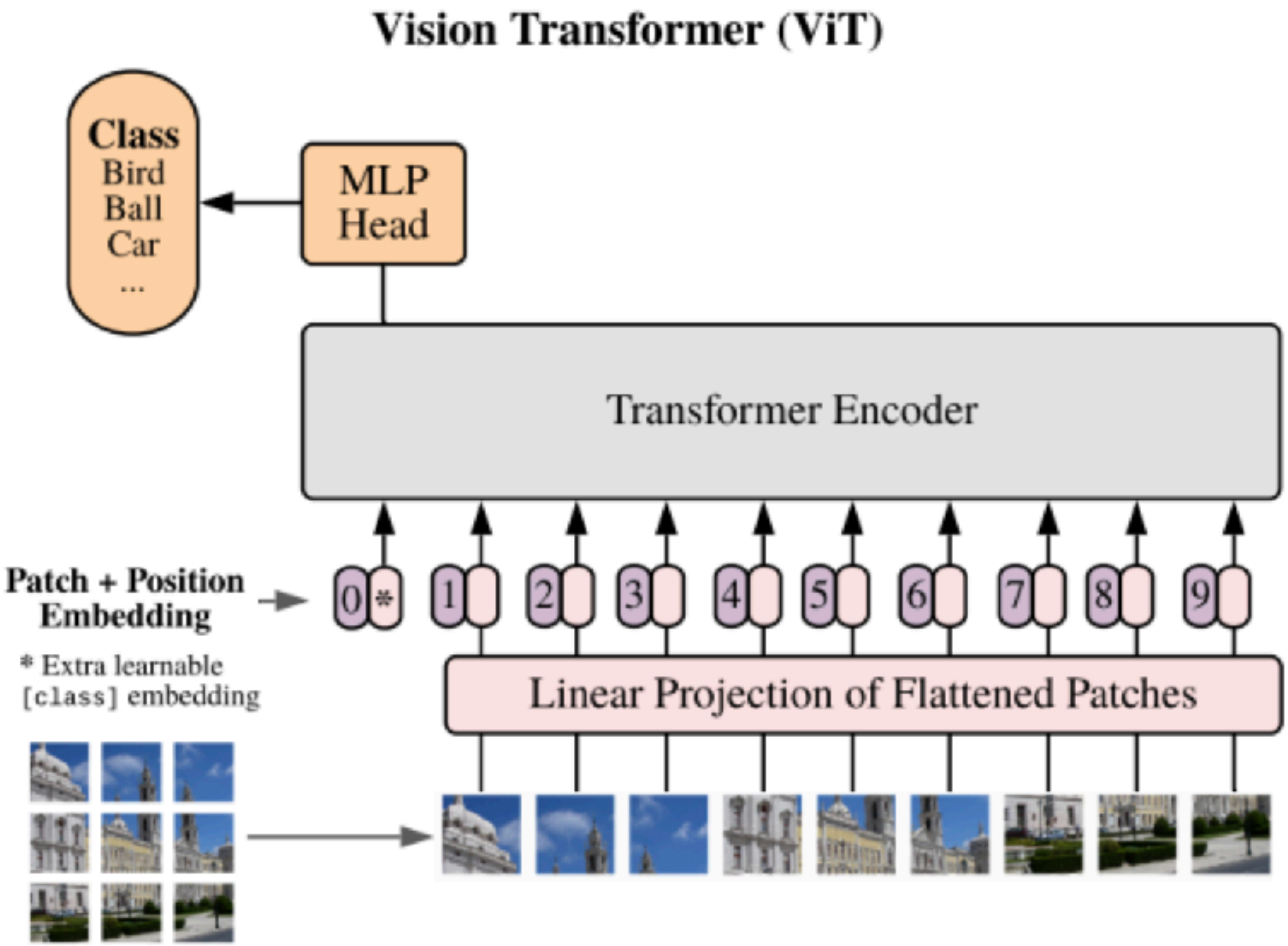
Vision Transformer

Dosovitskiy et al. 2020/2021

ViT: Vision Transformer



ViT: Vision Transformer



ViT: Position Encodings

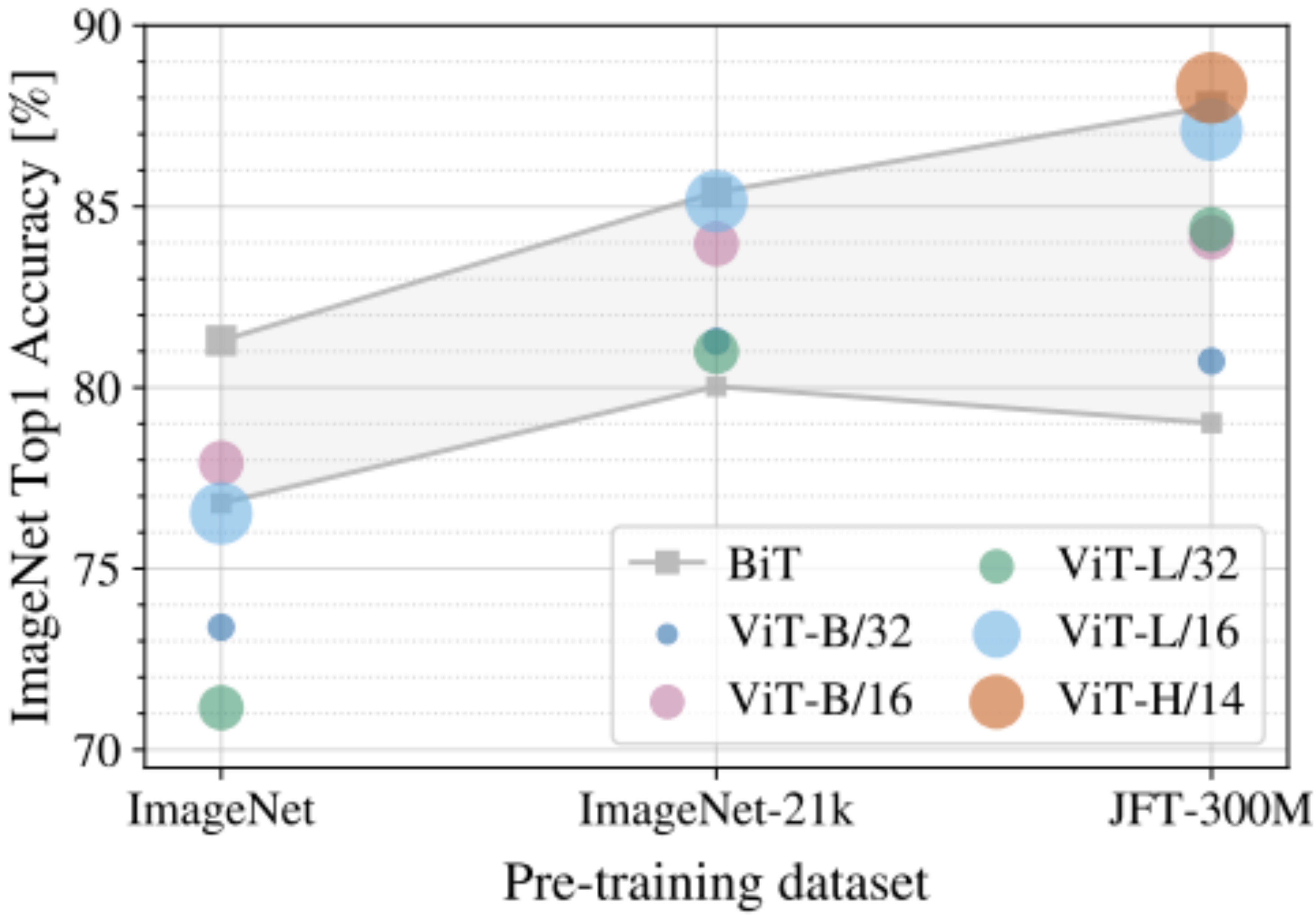
- The sinusoidal position encodings used for previous transformer models don't make sense for patches of images, as this encoding does a poor job at encoding the proximity of the patches.

- **For ViTs the position encodings are learnt by the model** instead of being imposed on the model as before.

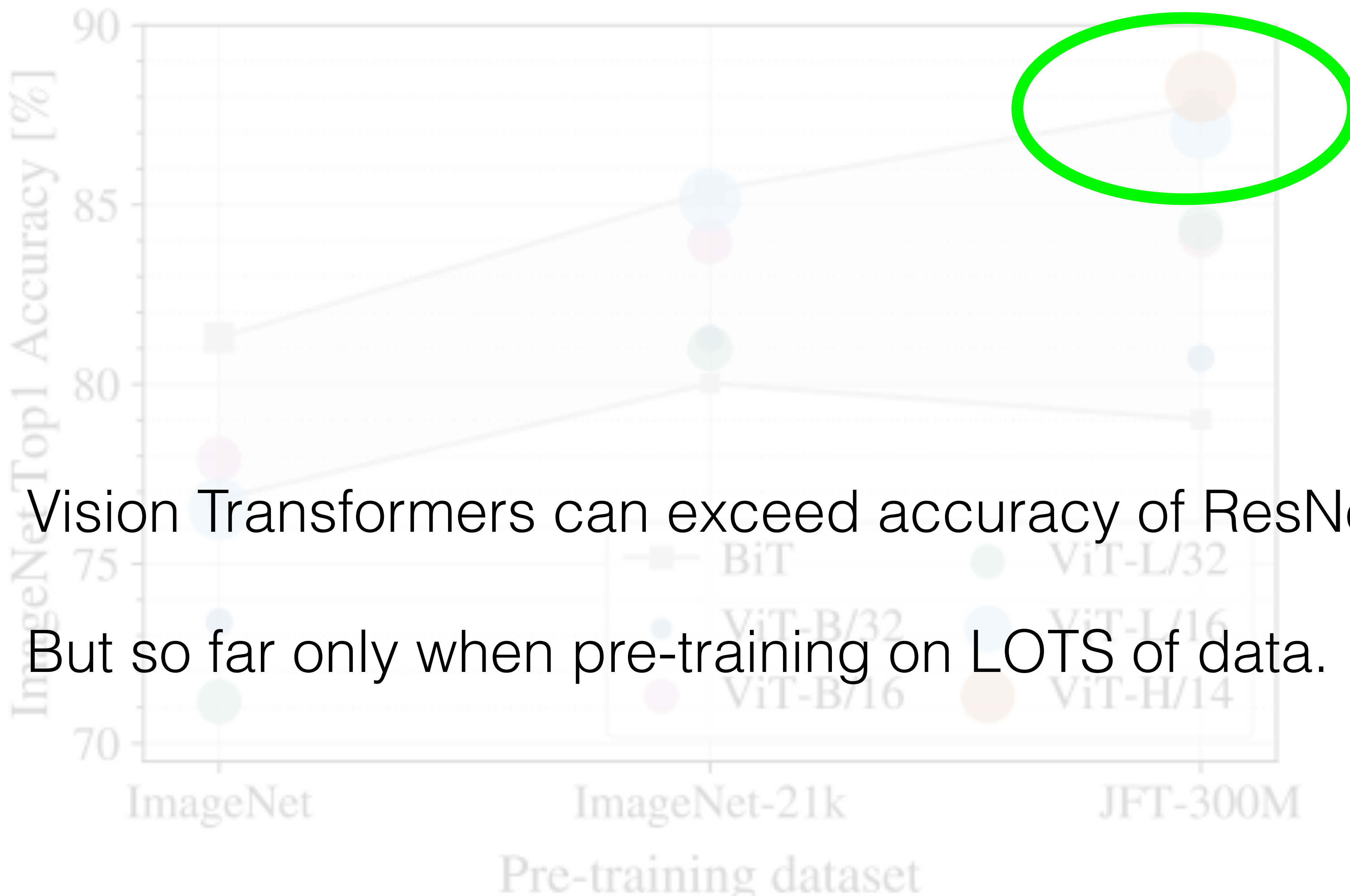
ViT: Vision Transformer

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

ViT: Vision Transformer

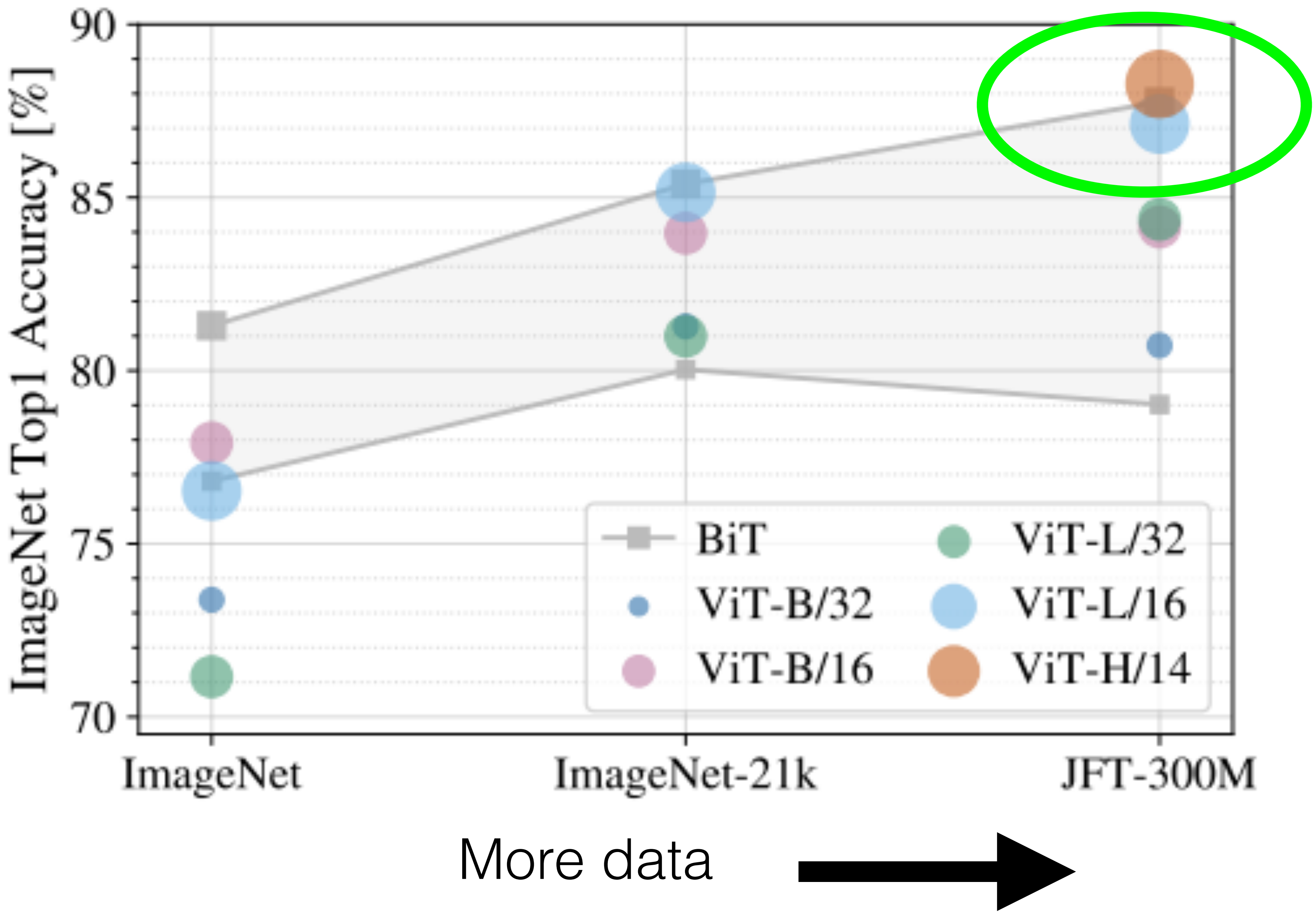


ViT: Vision Transformer

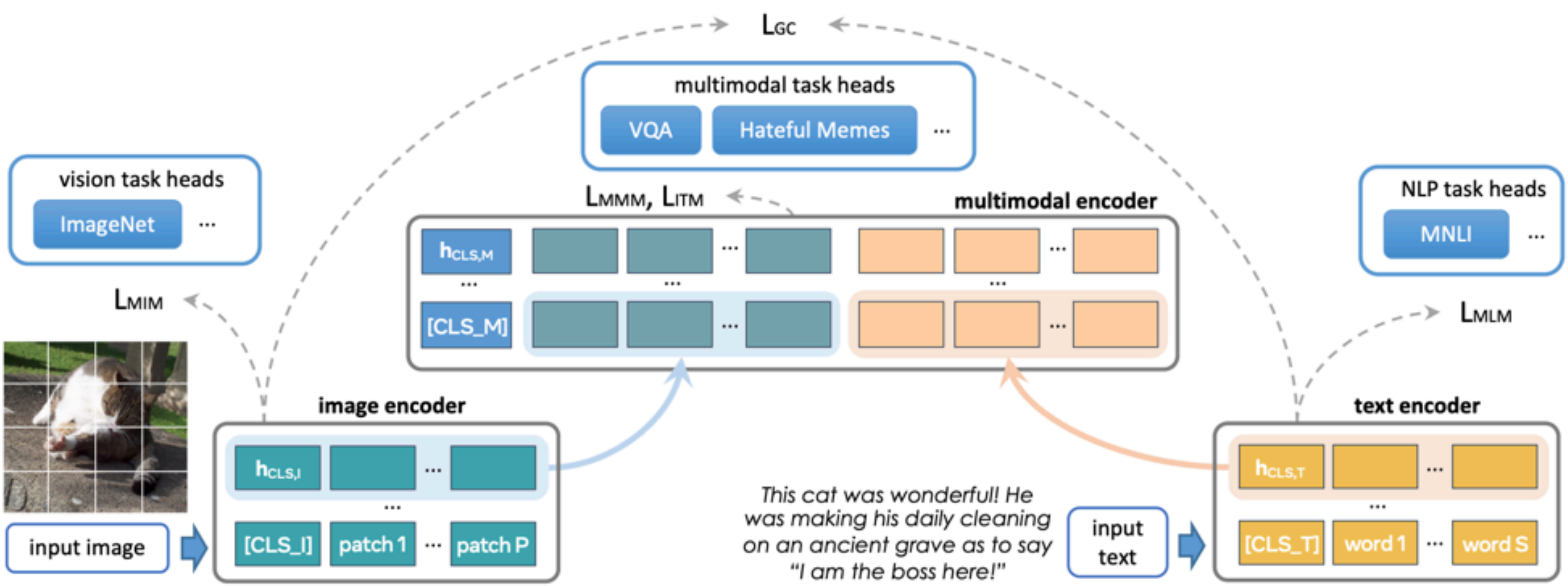


- Vision Transformers can exceed accuracy of ResNet.
- But so far only when pre-training on LOTS of data.

ViT: Vision Transformer



Multi-Modal: Vision and Language



Appendix A: Tokenization

Tokenization

But how do we decide on the set of words in our vocabulary?

- What about just using characters *a/A, b/B, c/C, d/D, e/E, ….*? On their own, they don't have enough meaning…
- We could choose all unique word strings as delineated by <space>, but there are far too many of these.
- We could first strip out punctuation and then choose all unique word strings, but still too many of these.
- Consider all these variations of the same base word *love: lovely, loving, lovingly, lovers*, etc. **We need to break words down into pieces—or tokens—to limit the vocabulary size**.

Byte Pair Encoding (BPE) Tokenization

But how do we decide on the set of words in our vocabulary?

- Let's start with our character set *a/A, b/B, c/C, d/D, e/E, etc. as our tokens*

- We can now **create a new token by combining the pair of existing tokens that is most represented** in the corpora.

- We repeat this process until we reach a predetermined limit on the vocabulary size (i.e., the number of unique tokens)

- Common words will get their own token and less common words will be broken down into a sequence of tokens.

Byte-Pair Encoding (BPE) Tokens

GPT-3 | Codex

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

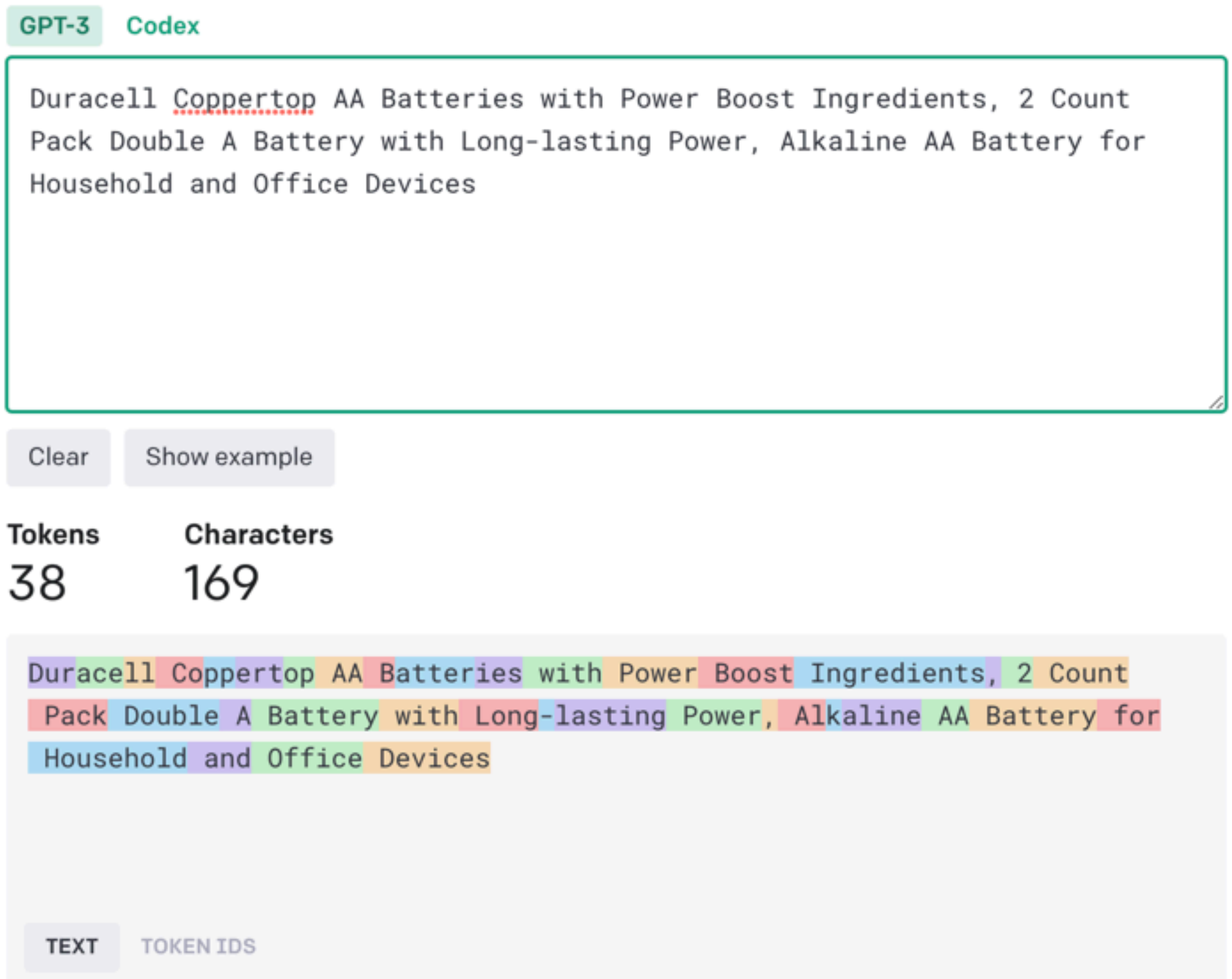
Clear | Show example

Tokens	Characters
116	556

Four score and seven years ago our fathers brought forth on this
continent, a new nation, conceived in Liberty, and dedicated to the
proposition that all men are created equal. Now we are engaged in a
great civil war, testing whether that nation, or any nation so conceived
and so dedicated, can long endure. We are met on a great battle-field of
that war. We have come to dedicate a portion of that field, as a final
resting place for those who here gave their lives that that nation might
live. It is altogether fitting and proper that we should do this.

TEXT | TOKEN IDS

Byte-Pair Encoding (BPE) Tokens



WordPiece Tokenization

WordPiece tokenization is much like BPE, but with a slightly different merge rule:

- It starts with characters *a/A, b/B, c/C, d/D, e/E, etc. as tokens*
- We can now **create a new token by combining the pair of existing tokens with the highest score:**

$$\text{score} = \frac{\text{frequency of pair}}{(\text{frequency of first token}) \times (\text{frequency of second token})}$$

- We repeat this process until we reach a predetermined limit on the vocabulary size (i.e., the number of unique tokens)
- This vocabulary of tokens is done before the model training

Appendix B: Common Crawl Dataset

Common Crawl Data