Natural Language Processing

Transformer and Self-Attention

Outline

1. Issues with RNNs
2. Attention as a Solution
3. Self-Attention
4. Transformer Encoder-Decoder
5. Transformer's Achievements
6. Transformer Variants

Issues with RNNs: Linear Interaction Distance

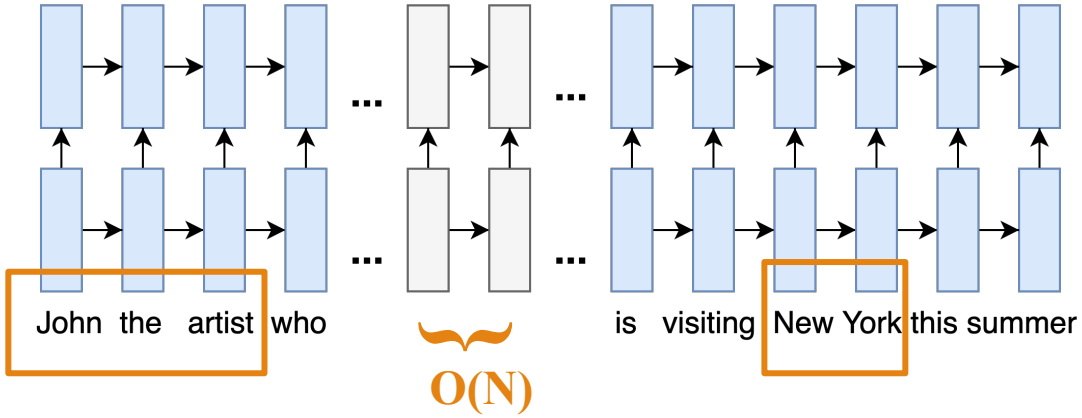
- In RNNs, the degree of words interact with each other is decided by their **distance**, but we already know that linear importance is not the right way to understand a sentence...

A Long Relative Clause

John, the artist who has painted many Murals in Vienna, is visiting New York this summer.

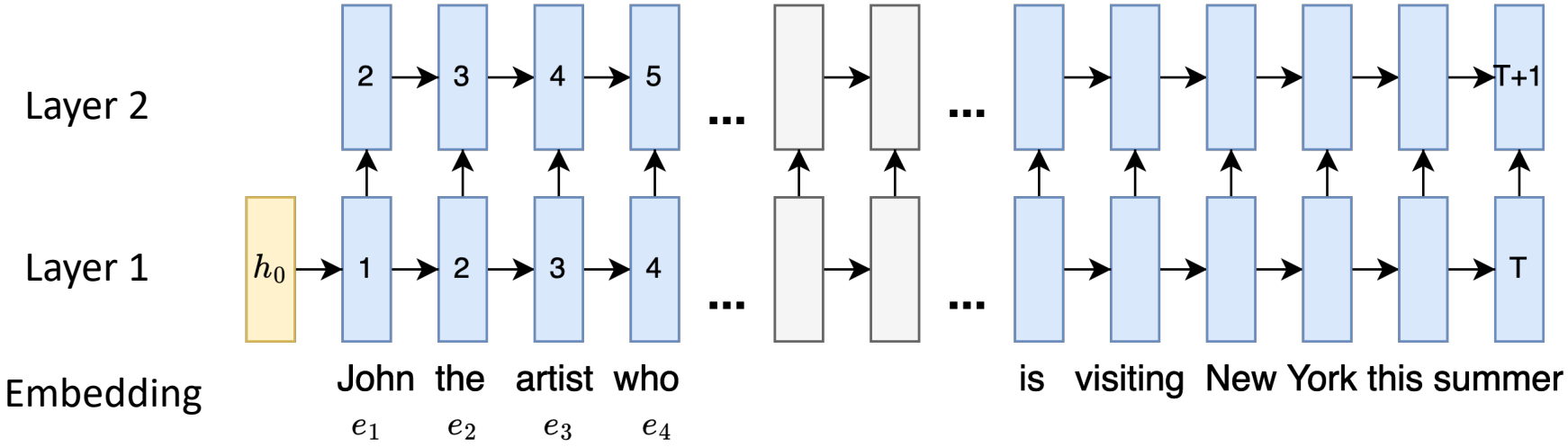
Where is John the artist visiting this summer?
A) Vienna
B) New York

▷ But the RNN may fail to identify **New York** as the correct answer since **the info of "John" has been propagated for almost O(*N*) (*N*: sequence length).**



Issues with RNNs: No Parallelizability

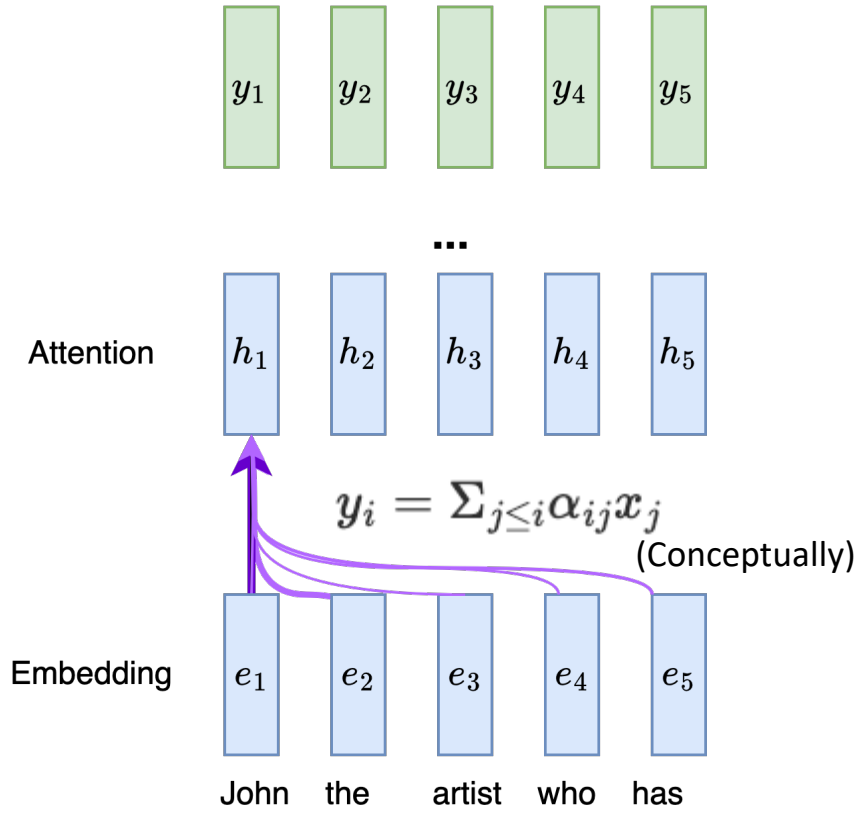
- Past hidden states need to be **fully** computed **before the next hidden state** to be ready for computation, which means it is **unparallelizable** between each timestep. This prevents RNN's use on large corpus or long text.



△ This is a 2-layer, uni-directional RNN. Each blue rectangle is a hidden state at the time-step of the layer. The number indicates the **min # of steps required for the hidden state to be computed**.

Attention as a Solution

- Attention within a sentence
- Every word attend to all words in the sentence between layers (solving **linear interaction distance** issue).
- α_{ij}: importance score/attention score of word i to word j.

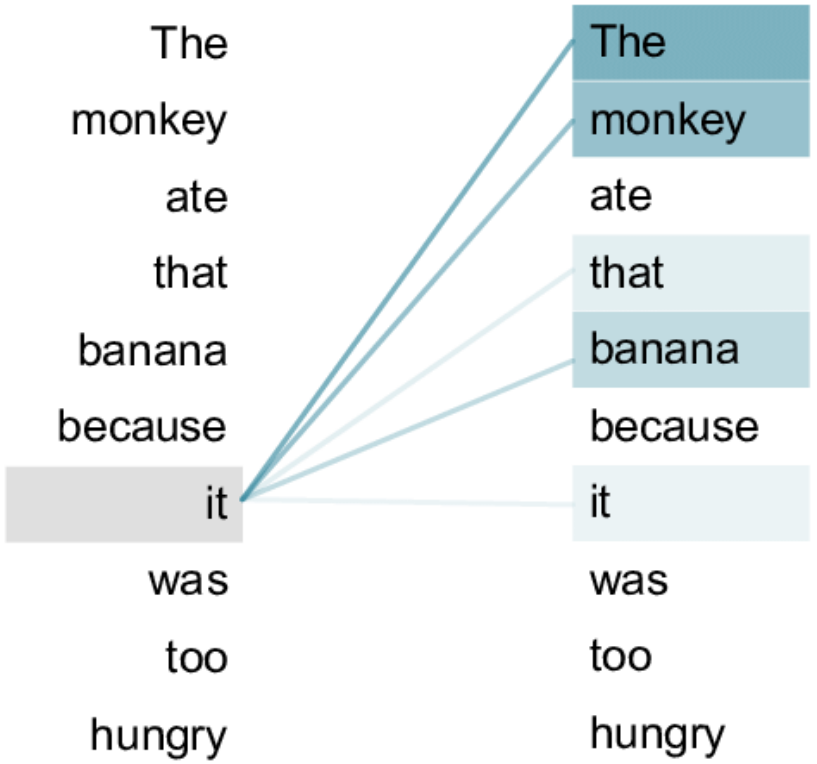


▷ We train the weight matrix α for each layer to learn "the attention score to be placed for each word on one another word." Width of the arrow (roughly) denote how large or small each α_{ij} is.

Self-Attention

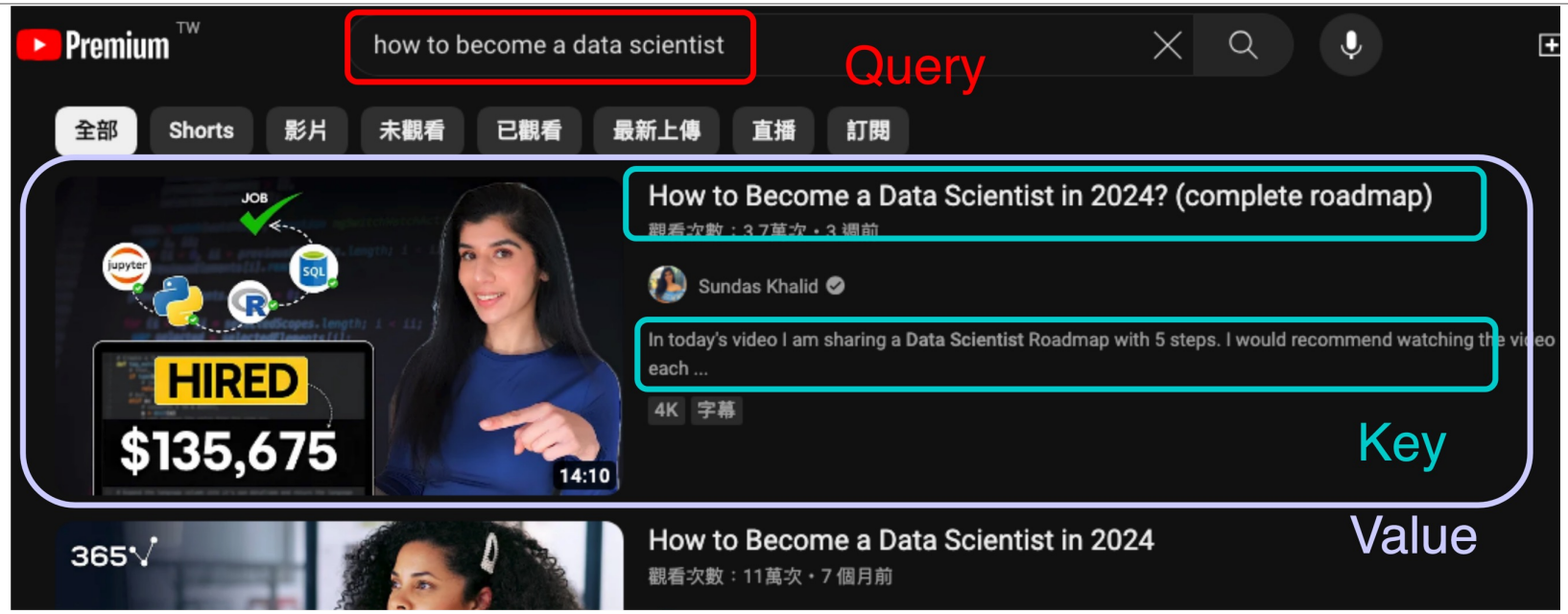
- Attention within a sentence
- Every word attend to all words in the same sentence.

Source: An example of the self-attention mechanism following long-distance... | Download Scientific Diagram (researchgate.net)



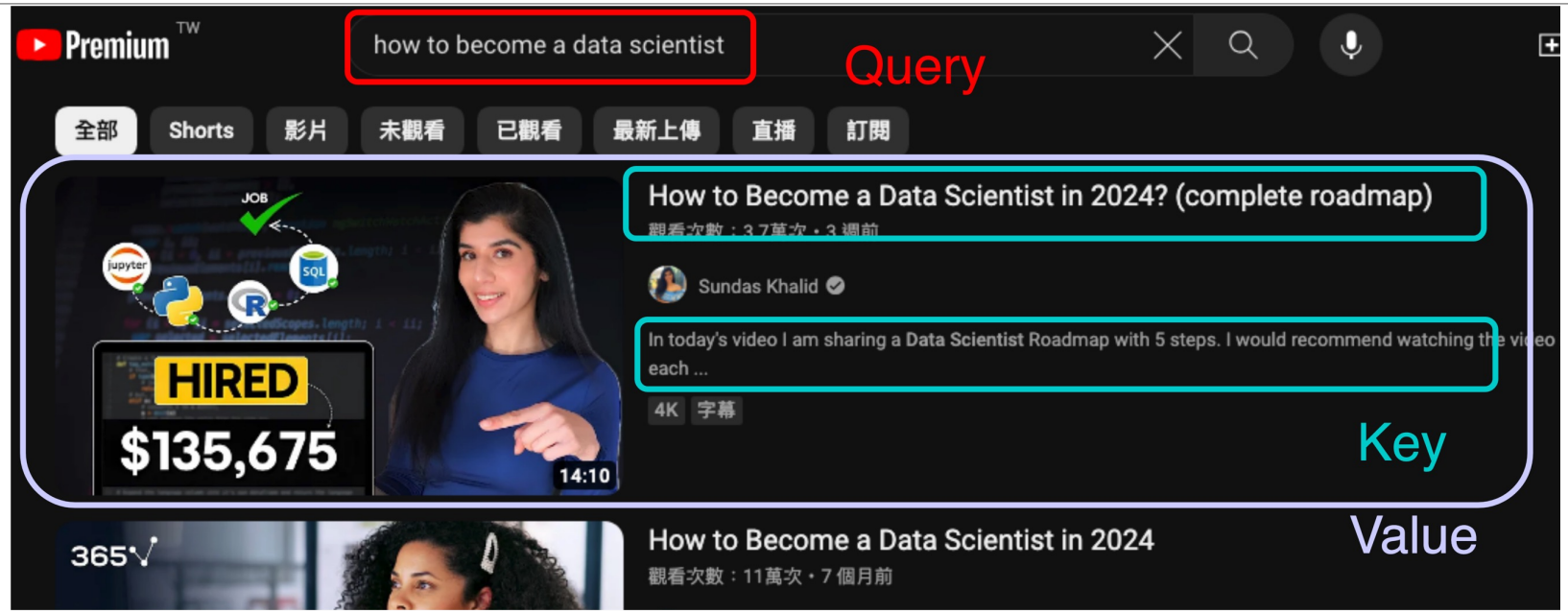
QKV Attention

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).



- The self-attention proposed in Vaswani et al. 2017 is **query-key-value attention (QKV attention)**.
- What is **query-key-value attention**?

QKV Attention

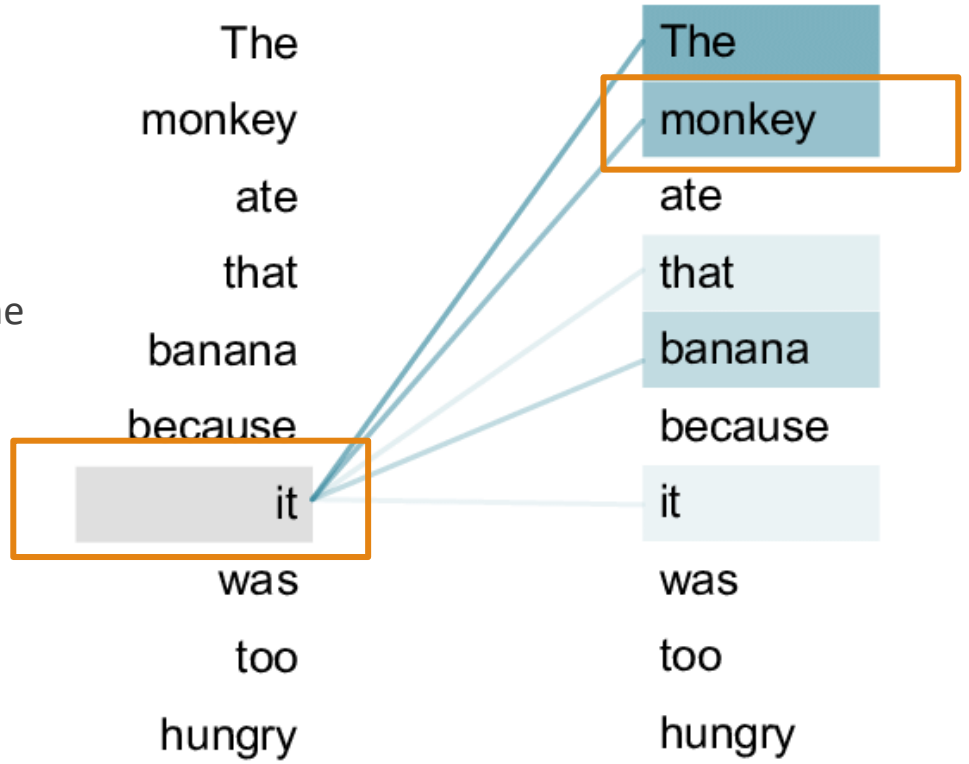


- The self-attention proposed in Vaswani et al. 2017 is **query-key-value attention (QKV attention)**.
- What is **query-key-value attention**?
  - The whole process is analogous to a **retrieval system**.
  - We **query** the search engine (eg. Youtube), which tries to map our query to the **keys** such as video title and video descriptions in its database, and then return some best matched videos (**values**).

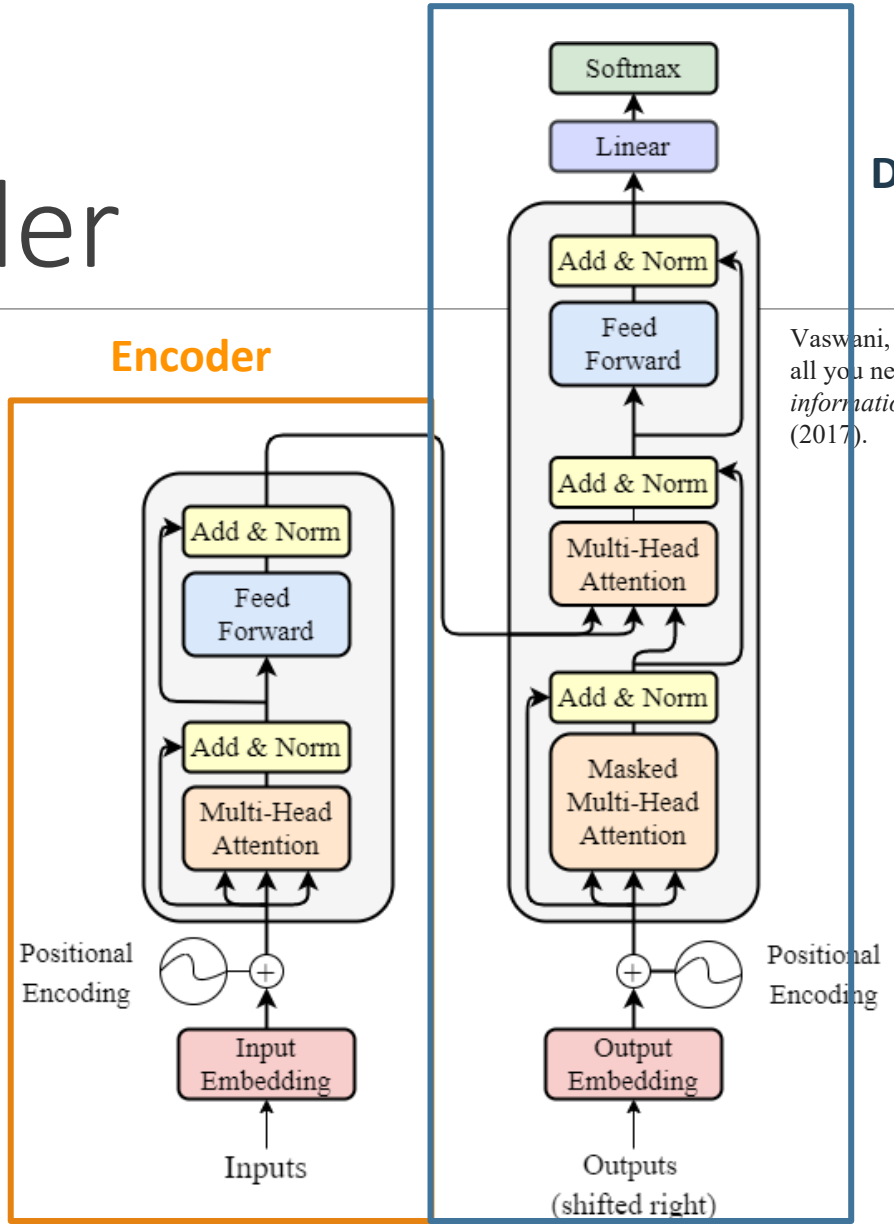
QKV Attention

- Back to the Natural Language example
  - The **query** is the information being looked for
    - Coreference Resolution object of “it”
  - The **key** is the context or reference
    - This is fairly abstract...
    - Each word *w* in the sentence gives hints of what “it” could be; eg. the **relative distance between *w* and it**, ***`because`, `was`*** **(neighboring) part-of-speech tags**, etc.
  - The **value** is the content being searched
    - One of the word in the sentence is the answer (monkey)

Source: An example of the self-attention mechanism following long-distance... | Download Scientific Diagram (researchgate.net)



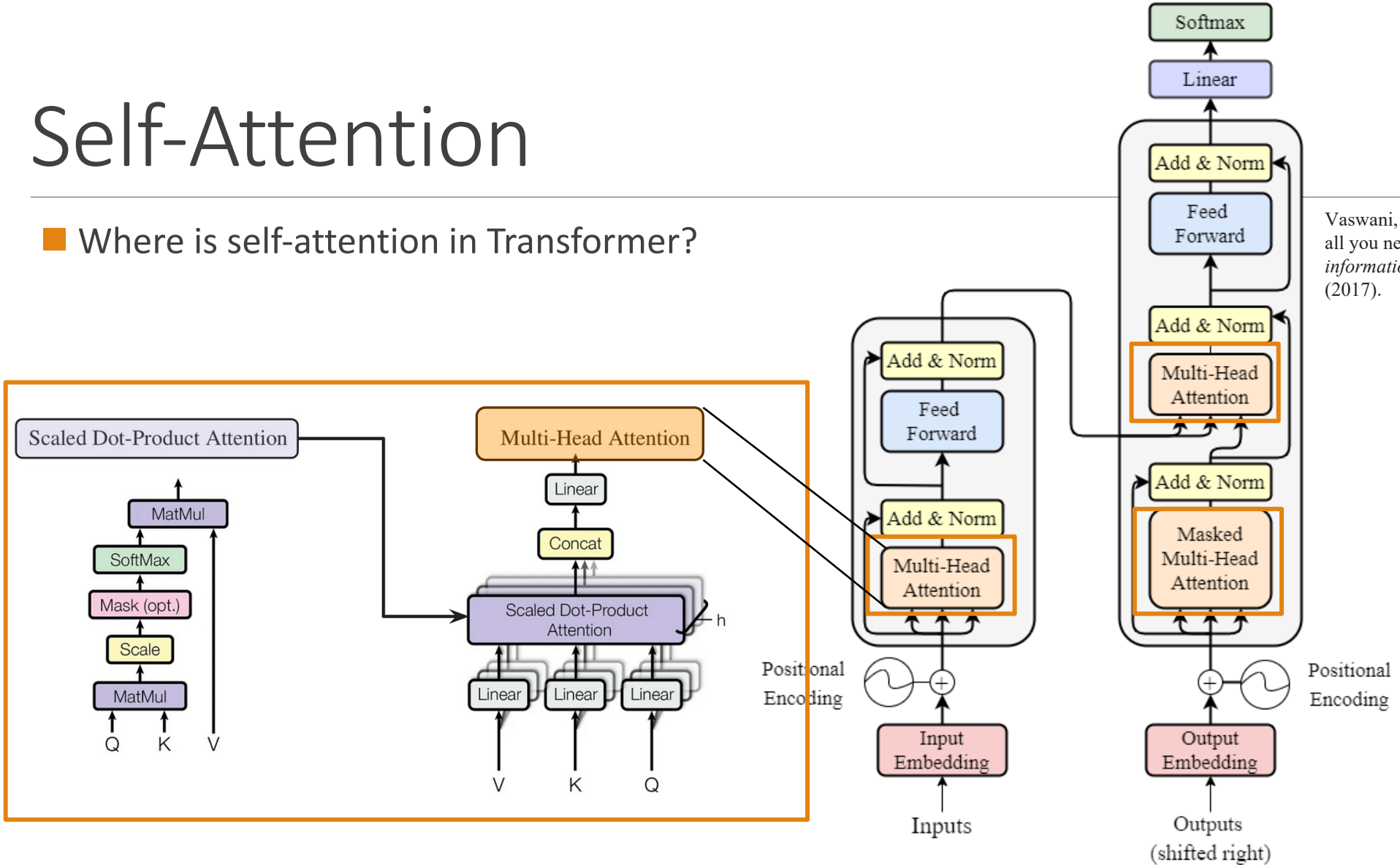
Transformer Encoder-Decoder



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Self-Attention

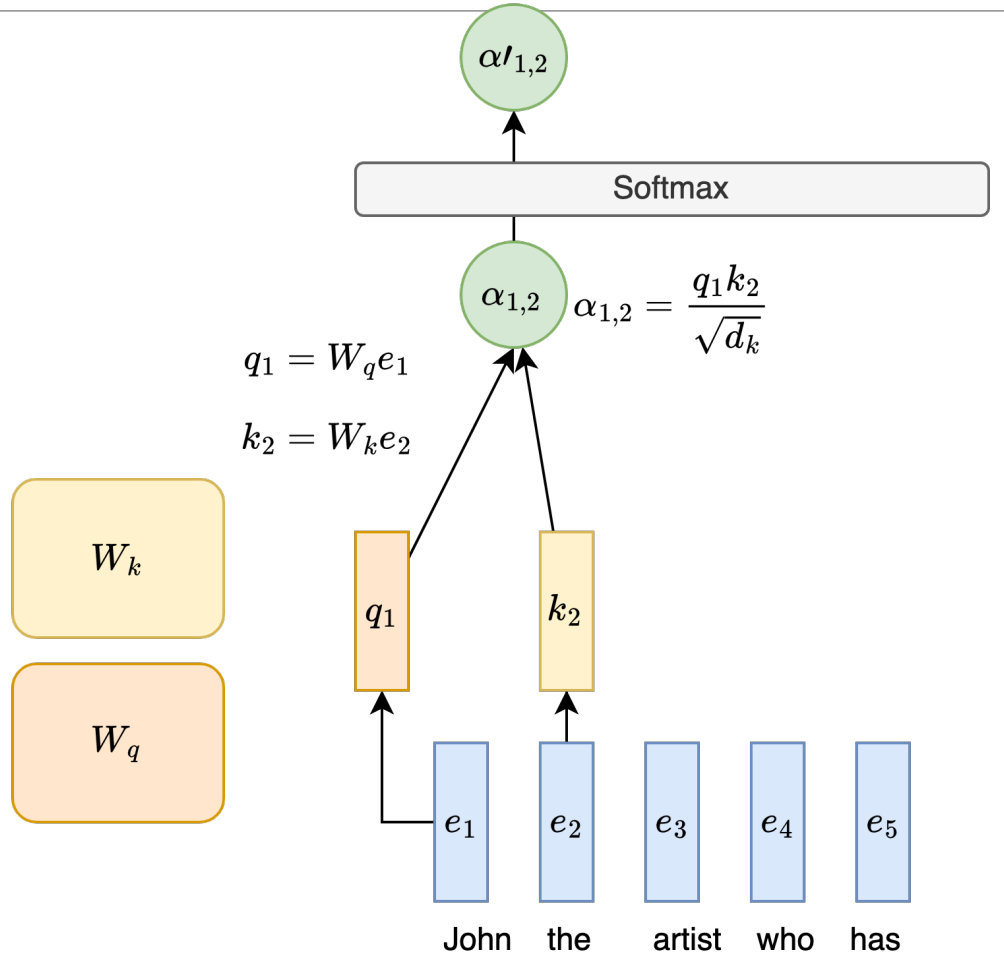
- Where is self-attention in Transformer?



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

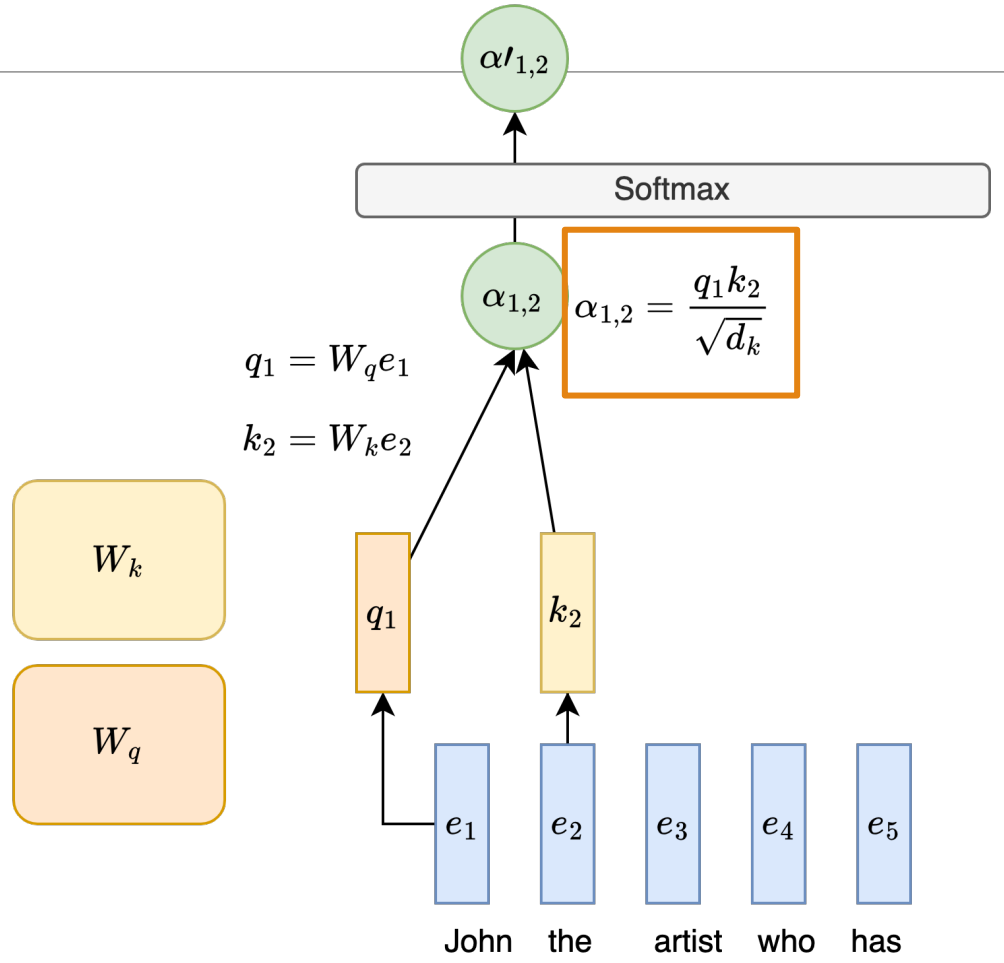
Self-Attention

- Let $e_1, .., e_N; e_i \in \mathbb{R}^d$ be input embeddings of the text sequence.
- Initialize 3 matrices W_q, W_k, W_v.
  - The matrices are used to project the input from $e_1, .., e_N$ to $q_1, .., q_N$, $k_1, .., k_N$ and $v_1, .., v_N$ respectively.
- Do a **scaled dot product** to get a scalar $\alpha_{1,2}$, meaning **"attention score from word 1 to 2."**



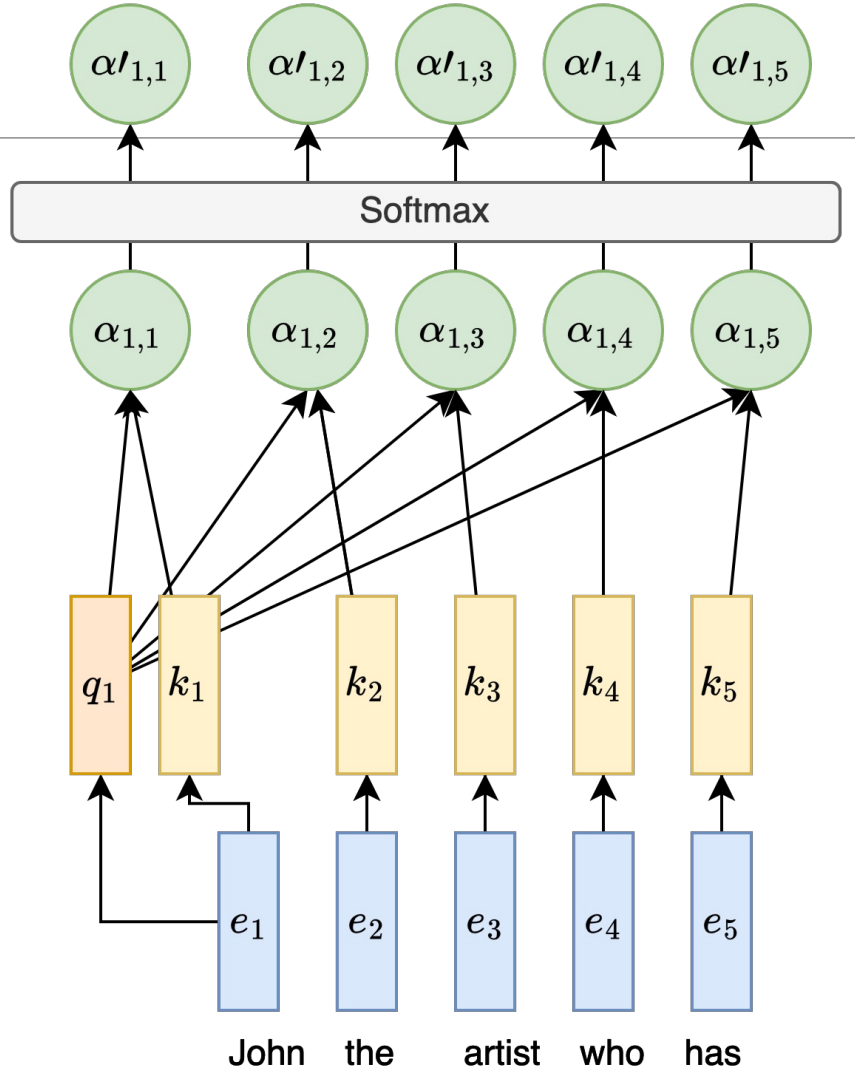
Attention Score

- The original Transformer uses **scaled dot product** (there are many others).
  - $score(e_i, e_j) = \frac{q_i k_j}{\sqrt{d_k}}$
  - d_k (= dim of keys = dim of queries)
  - **Why Scaling?**
    - Because **greater d_k with Softmax() results in vanishing gradients**.
    - By initializing all queries and keys to have $\mu = 0, \sigma = 1$, we can ensure $score(e_i, e_j)$ to be numerically stable, as long as we scaled by $\sqrt{d_k}$.



Attention Score

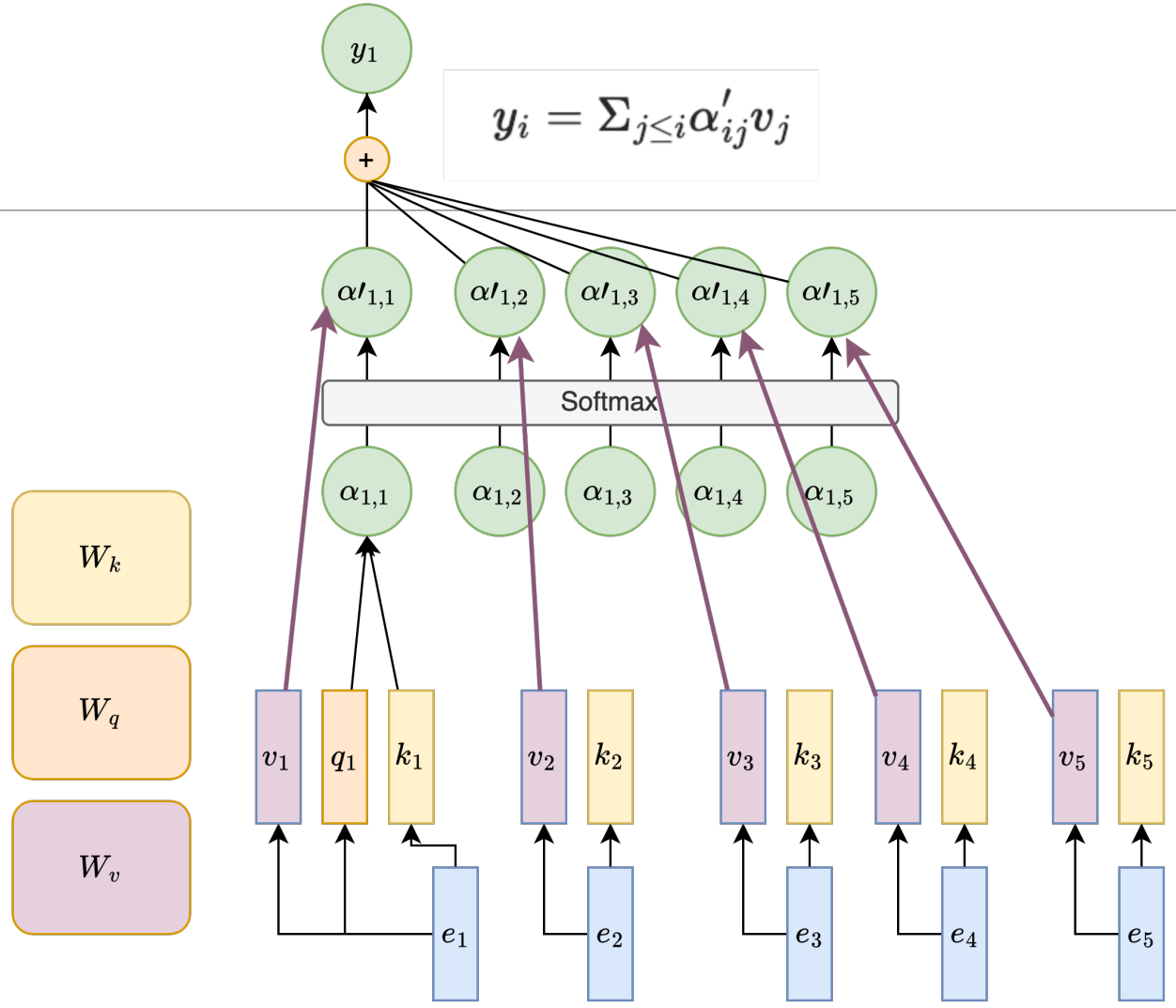
- For every pair of i, j, We calculate the scores. The figure illustrates q = 1.



$$\alpha'_{1,i} = \frac{\exp(\alpha_{1,i})}{\sum_j \exp(\alpha_{1,j})}$$

Self-Attention

- The figure illustrates when q = 1.
- The → indicates scalar vector multiplication.



Self-Attention

- Mathematically, in terms of a pair of indices i, j:

Project the sequence vectors to q,k,v.

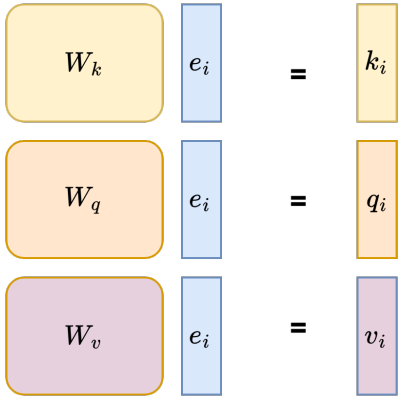
$$q_i = W^Q x_i; k_i = W^K x_i; v_i = W^V x_i$$

$$score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = score(x_i, x_j) \quad \forall j \leq i$$

$$\alpha'_{ij} = softmax(\alpha_{ij}) = \frac{exp(score(x_i, x_j))}{\Sigma_{k=1}^{i} exp(score(x_i, x_k))} \quad \forall j \leq i$$

$$y_i = \Sigma_{j \leq i} \alpha'_{ij} v_j$$



Self-Attention

- Mathematically, in terms of a pair of indices i, j:

$$q_i = W^Q x_i; k_i = W^K x_i; v_i = W^V x_i$$

$$score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = score(x_i, x_j) \quad \forall j \leq i$$

$$\alpha'_{ij} = softmax(\alpha_{ij}) = \frac{exp(score(x_i, x_j))}{\Sigma_{k=1}^{i} exp(score(x_i, x_k))} \quad \forall j \leq i$$

$$y_i = \Sigma_{j \leq i} \alpha'_{ij} v_j$$

Calculate Attention Score/Weight

Self-Attention

- Mathematically, in terms of a pair of indices i, j:

$$q_i = W^Q x_i; k_i = W^K x_i; v_i = W^V x_i$$

$$score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = score(x_i, x_j) \quad \forall j \leq i$$

$$\alpha'_{ij} = softmax(\alpha_{ij}) = \frac{exp(score(x_i, x_j))}{\Sigma_{k=1}^{i} exp(score(x_i, x_k))} \quad \forall j \leq i$$

$$y_i = \Sigma_{j \leq i} \alpha'_{ij} v_j$$

Calculate Attn-Weighted Average on v

Self-Attention

- Mathematically, in matrix notation:
  - ***N***: sequence length
  - ***d***: hidden dimension, original Transformer uses d = 1024.
- Solves the parallelizability problem!!

$$X \in \mathbb{R}^{N \times d}; W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$$

$$Q = XW^Q; K = XW^K, V = XW^V$$

$$A' = Softmax(\frac{QK^T}{\sqrt{d_k}})$$

$$Y = A'V$$

Self-Attention

- Mathematically, in matrix notation:
  - ***N***: sequence length
  - ***d***: hidden dimension, original Transformer uses d = 1024.
- Solves the parallelizability problem!!

$$X \in \mathbb{R}^{N \times d}; W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$$

$$Q = XW^Q; K = XW^K, V = XW^V$$

$$A' = Softmax(\frac{QK^T}{\sqrt{d_k}})$$

$$Y = A'V$$

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

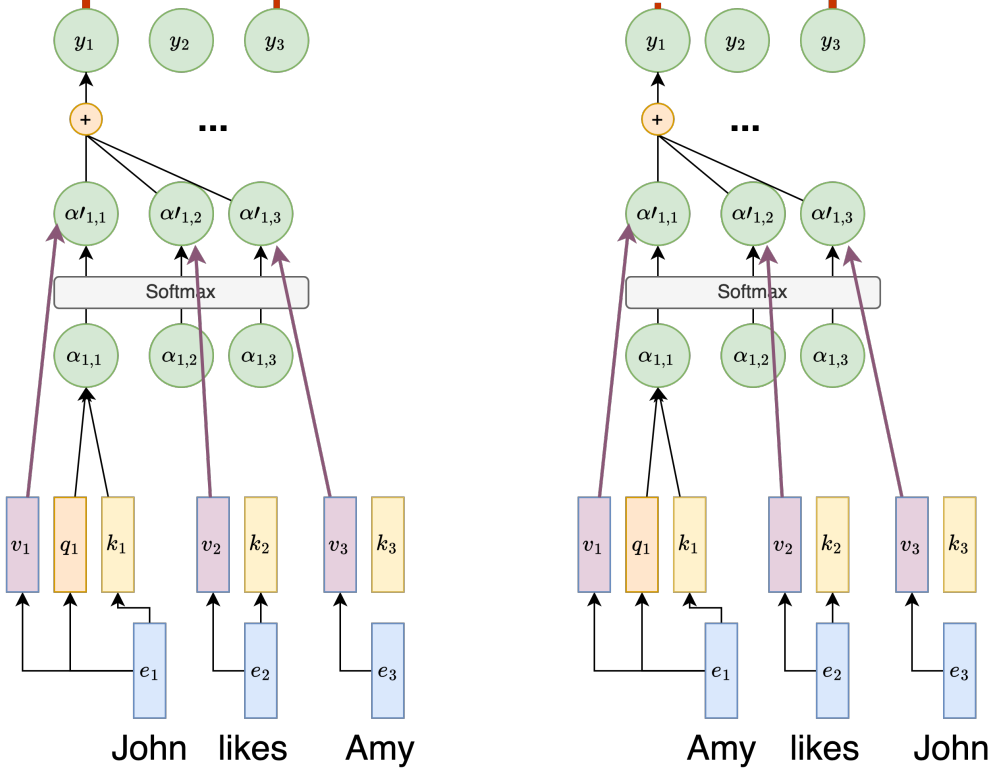
$$Y \in \mathbb{R}^{N \times d}$$

Lack of Position Modeling

- Recall the 2 issues of RNNs:
  - **Linear interaction distance**
    - **[solved]** by directly learning attention weight instead of decaying the degree by distance.
  - **Lack of parallelizability**
    - **[solved]** by using matrix multiplication.

- But ... **We have no way to know the relative distance of one word to another!**
  - The same word, eg. “Amy”, in the 2 sentences are encoded to the same y (with the same weight matrices).

With same Ws,
(Left) y_1 = (Right) y_3
(Left) y_3 = (Right) y_1

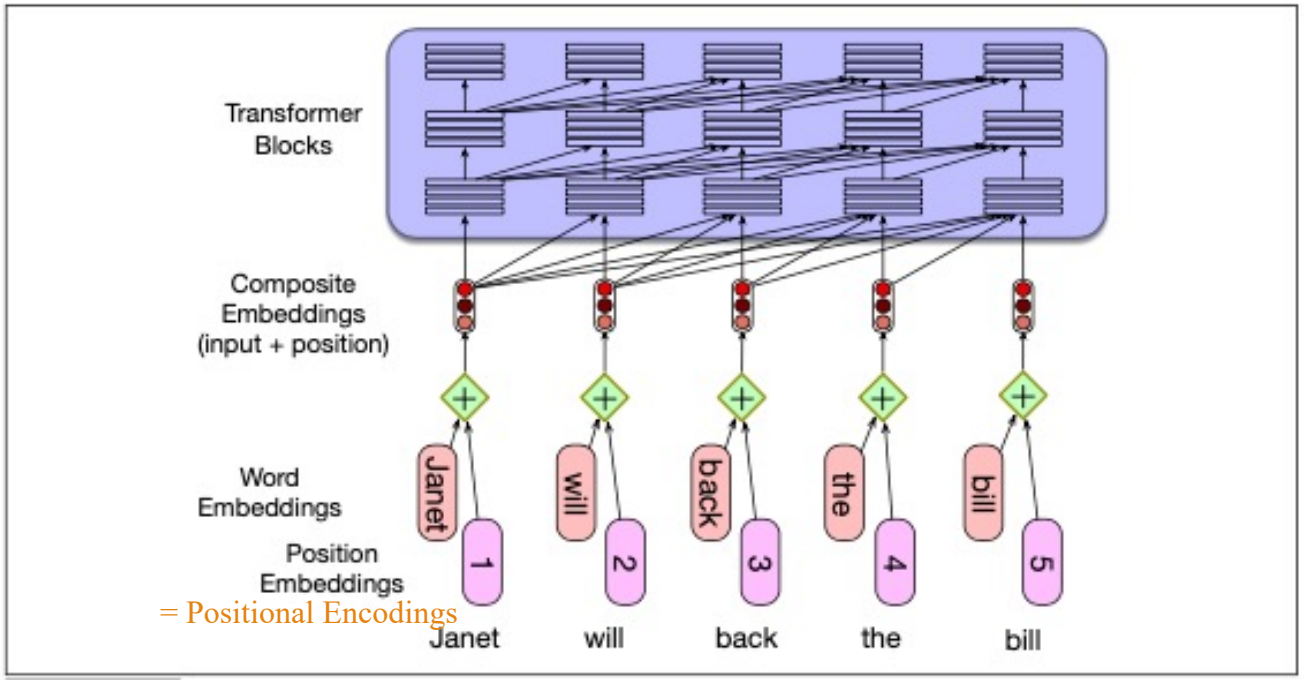


Problematic because these 2 sentences convey different meanings, but same words are encoded identically!

Positional Encoding

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

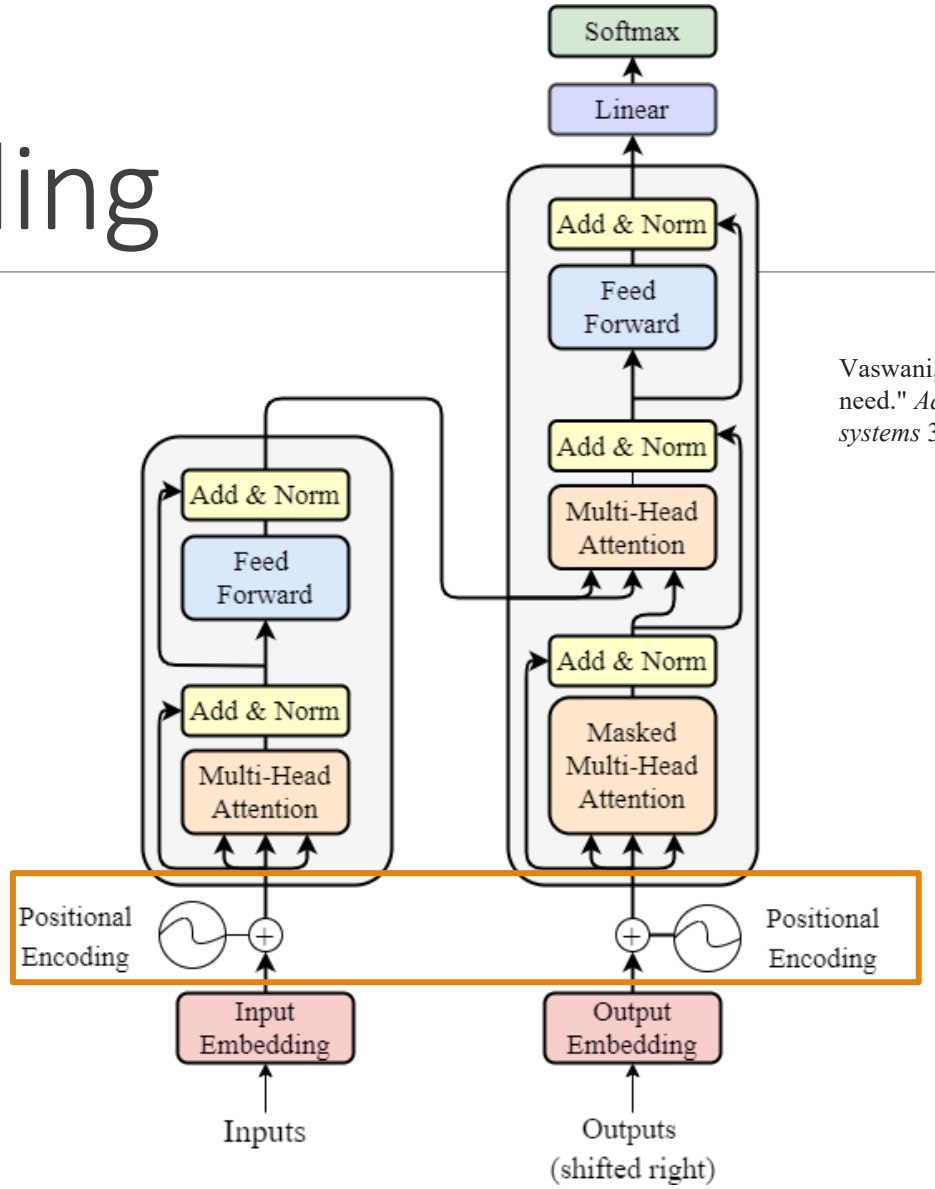
- The solution is to add some values to **tell the model the positional info of each token**.
- The RHS figure does not work, why?
  - **Lack of Magnitude Boundary**: The value could get very large as sequence length increases.
  - **Lack of Length Boundary**: Suppose maximum length 50 in training set, maximum length 53 in tst set ➜ positional embedding 51, 52, 53 are NOT trained.



Source: Jurafsky, D. & Martin, J. Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. 3rd Edition draft (2020). Figure 10.6 in Chapter 10.

Positional Encoding

- Where is the Positional Encoding inside the transformer?
- In brief, it is a way to tell the model the position of each token in a sequence.



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Positional Encoding

- Again, there are many positional encoding approaches. We only introduce Vaswani's (2017).
- A cyclic/periodic solution
- Do not train positional encodings.
- t: timestep, $\overrightarrow{p_t} \in \mathbb{R}^d$: positional encoding,

$i < d$: positional encoding element index

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

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

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Positional Encoding

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, t	i=0	i=1	i=2	i=3
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

$\overrightarrow{p_0}$

Positional Encoding Matrix for the sequence 'I am a robot'

Source: A Gentle Introduction to Positional Encoding in Transformer Models, Part 1 - MachineLearningMastery.com

Positional Encoding

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, t	i=0	i=1	i=2	i=3
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

$\overrightarrow{p_1}$

Positional Encoding Matrix for the sequence 'I am a robot'

Source: A Gentle Introduction to Positional Encoding in Transformer Models, Part 1 - MachineLearningMastery.com

Positional Encoding

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, t	i=0	i=1	i=2	i=3
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

$\overrightarrow{p_2}$

Positional Encoding Matrix for the sequence 'I am a robot'

Source: A Gentle Introduction to Positional Encoding in Transformer Models, Part 1 - MachineLearningMastery.com

Positional Encoding

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, t	i=0	i=1	i=2	i=3
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

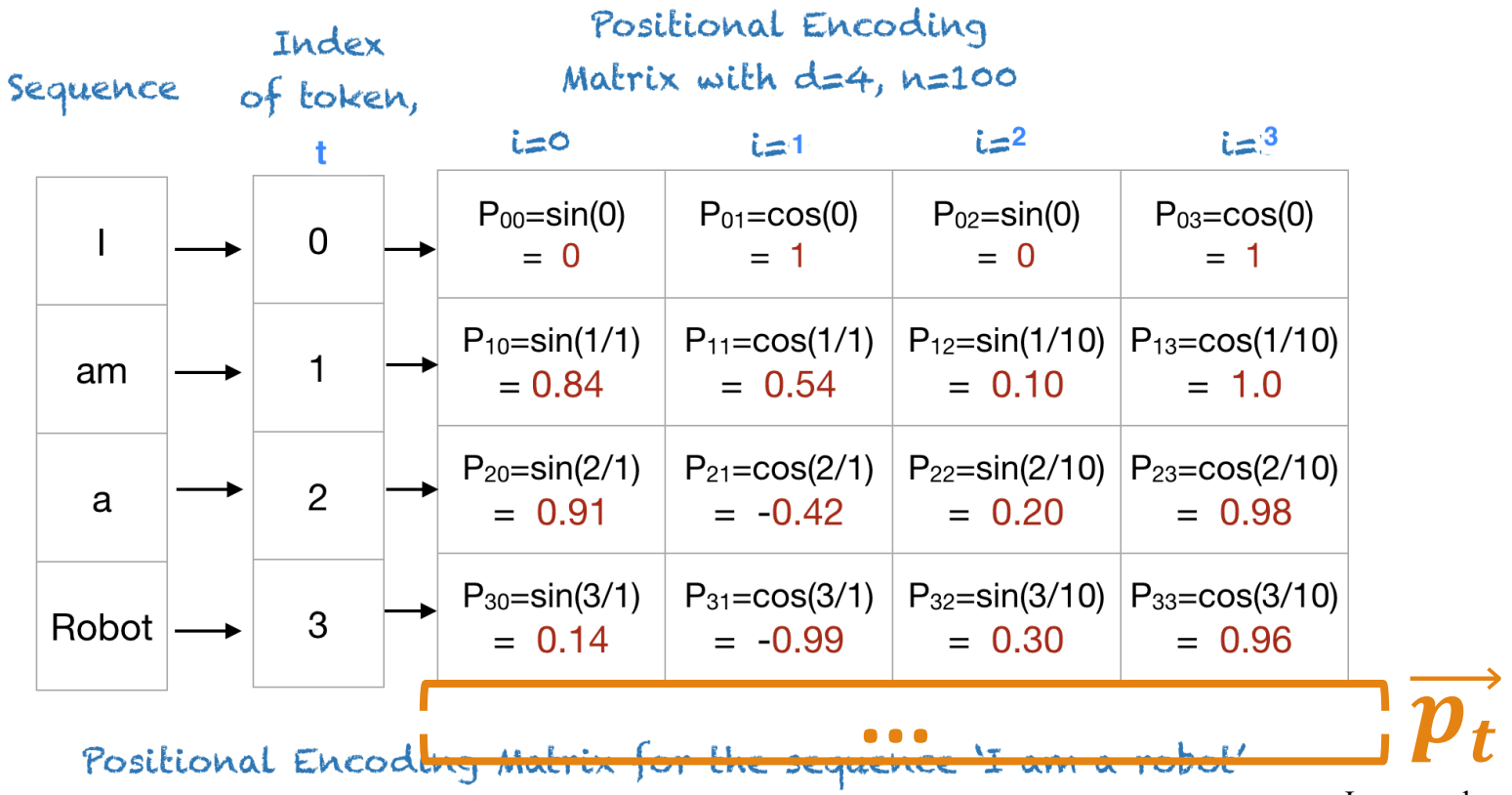
$\overrightarrow{p_3}$

Positional Encoding Matrix for the sequence 'I am a robot'

Source: A Gentle Introduction to Positional Encoding in Transformer Models, Part 1 - MachineLearningMastery.com

Positional Encoding

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

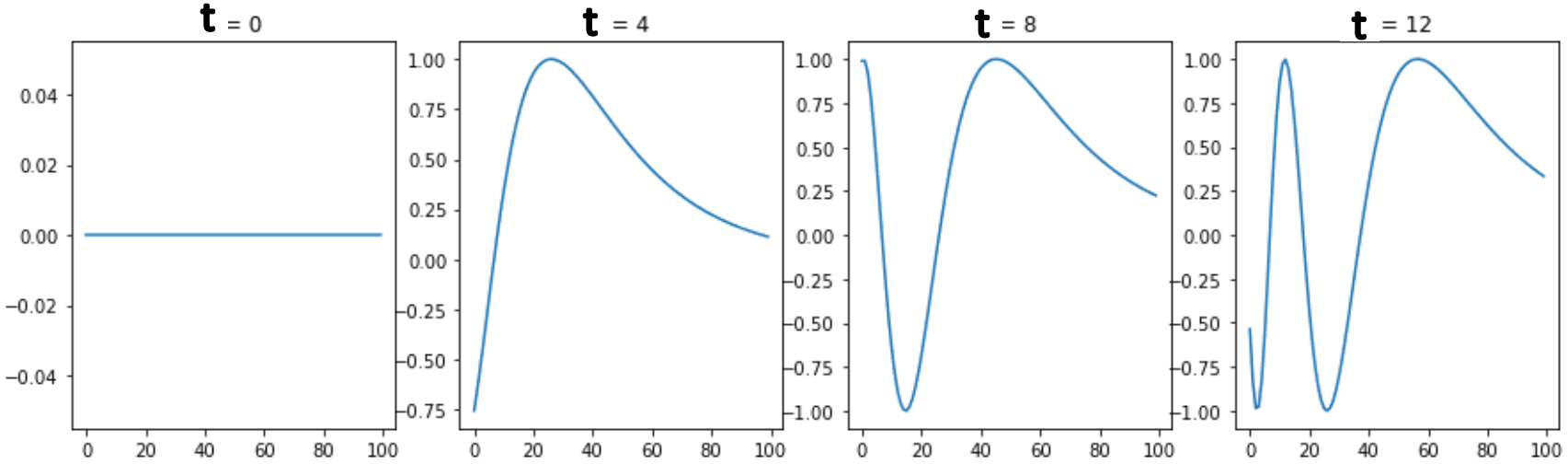


Source: A Gentle Introduction to Positional Encoding in Transformer Models, Part 1 - MachineLearningMastery.com

Positional Encoding

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

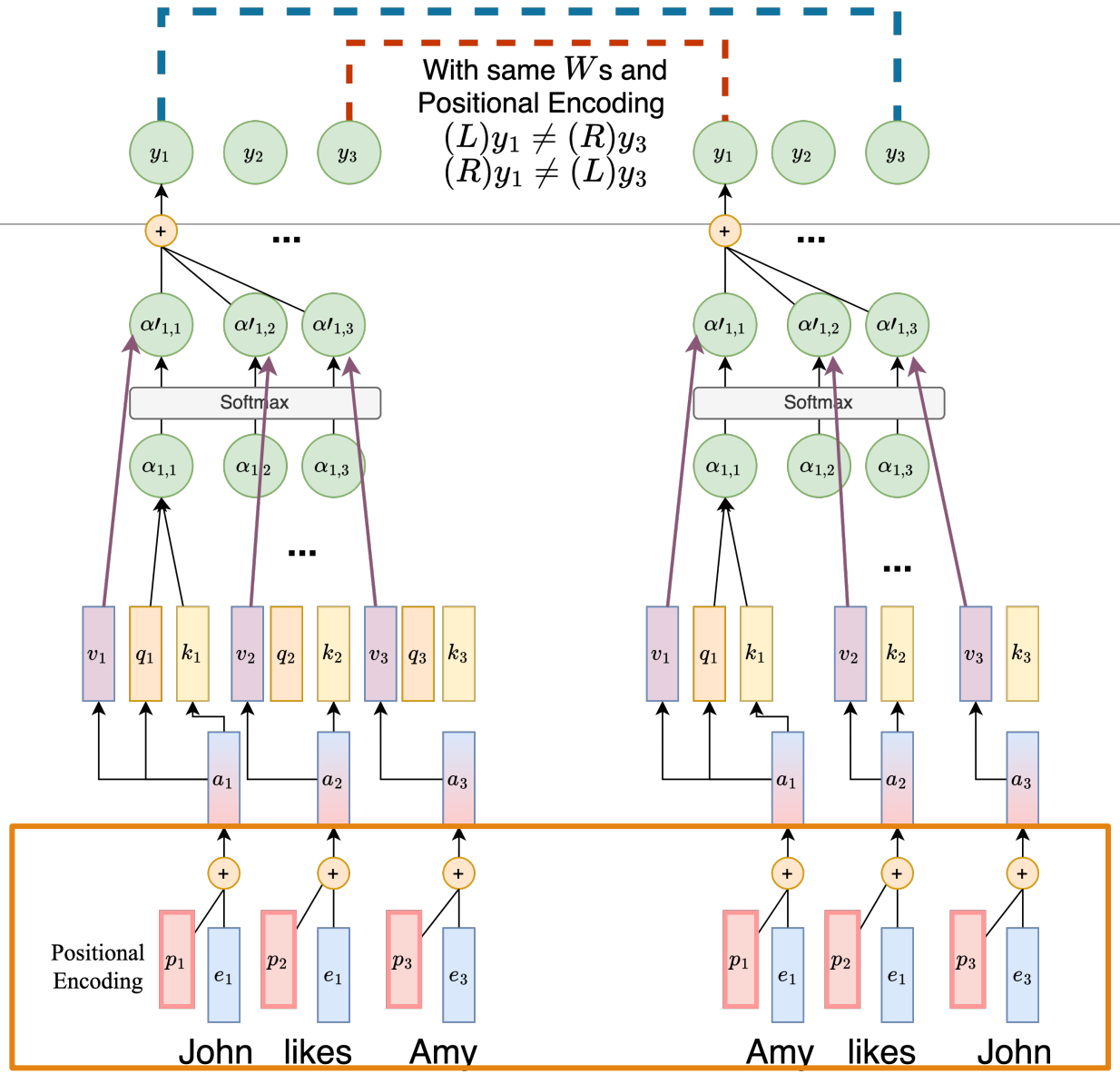
- t: timestep, $\overrightarrow{p_t} \in \mathbb{R}^d$: positional encoding
- Each positional embedding at position t is a sinusoidal wave. Closer positions have closer patterns.
- **Solved** **Lack of Magnitude Boundary**: Numeric range in [-1, 1].
- **Solved** **Lack of Length Boundary**: Cyclic, can go on forever.



$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Source: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Positional Encoding



Multi-head Attention

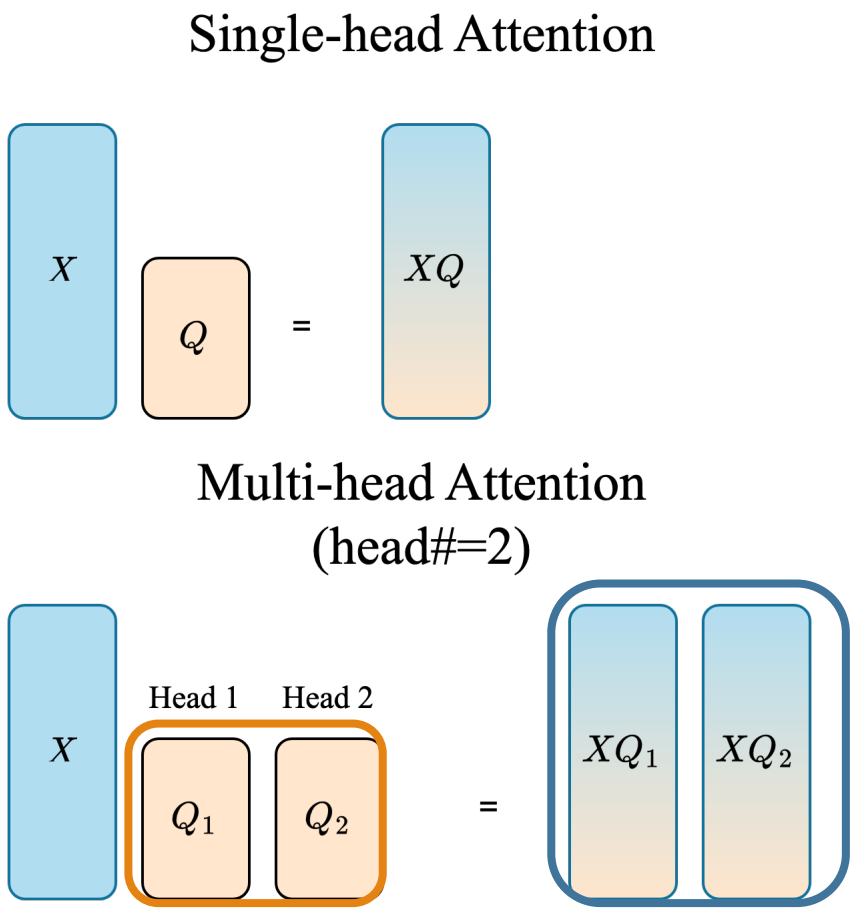
- In p.9, we describe self-attention as QKV attention, where
  - **we query the search engine**
  - **search engine tries to map our query to the keys**
  - **output some best matched videos (values)**
- What if we want to **have queries focusing on different aspects**?
  - For example, one set of queries focusing on s**emantic similarity**, another set focusing on **passive/active voice.**
  - We need **multiple** attention mechanisms, i.e. **multiple (attention) heads!**

Multi-head Attention

- Recall p.20, where we have explained the **Attention()** function.
- Multi-head Attention is essentially multiple self-attentions...

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

▷ In computation, we can group the $Q_1, Q_2, \ldots$ matrices into one Q matrix, hence transform the computation process into single-head attention.



Self-Attention (p.20)

- Mathematically, in matrix notation:
  - ***N***: sequence length
  - ***d***: hyperparameter, hidden dimension, original Transformer uses d = 1024.
- Solves the parallelizability problem!!

$$X \in \mathbb{R}^{N \times d}; W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$$

$$Q = XW^Q; K = XW^K, V = XW^V$$

$$A' = Softmax(\frac{QK^T}{\sqrt{d_k}})$$

$$Y = A'V$$

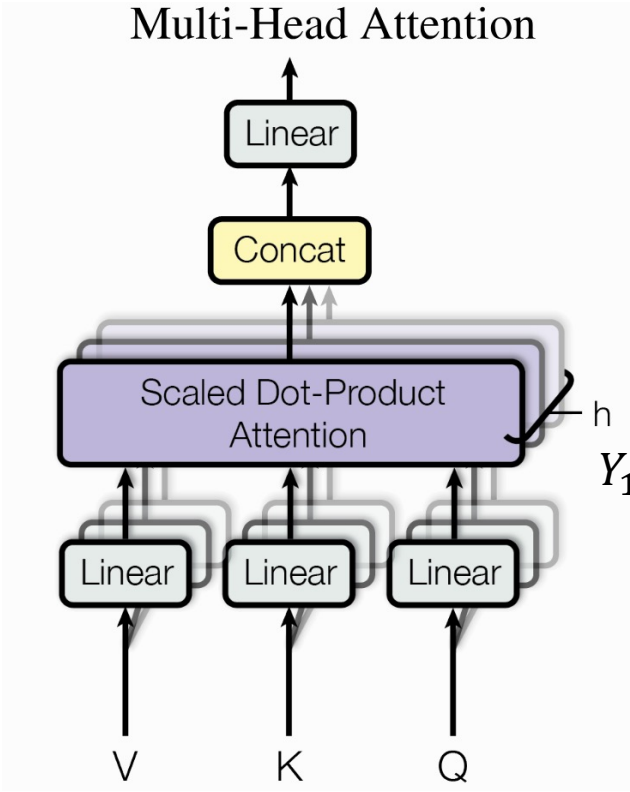
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V = Y$$

$$Y \in \mathbb{R}^{N \times d}$$

Multi-head Attention

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$$

$\in \mathbb{R}^{N \times (h \cdot d)}$



$Y_h \in \mathbb{R}^{N \times d}$

...

$Y_2 \in \mathbb{R}^{N \times d}$

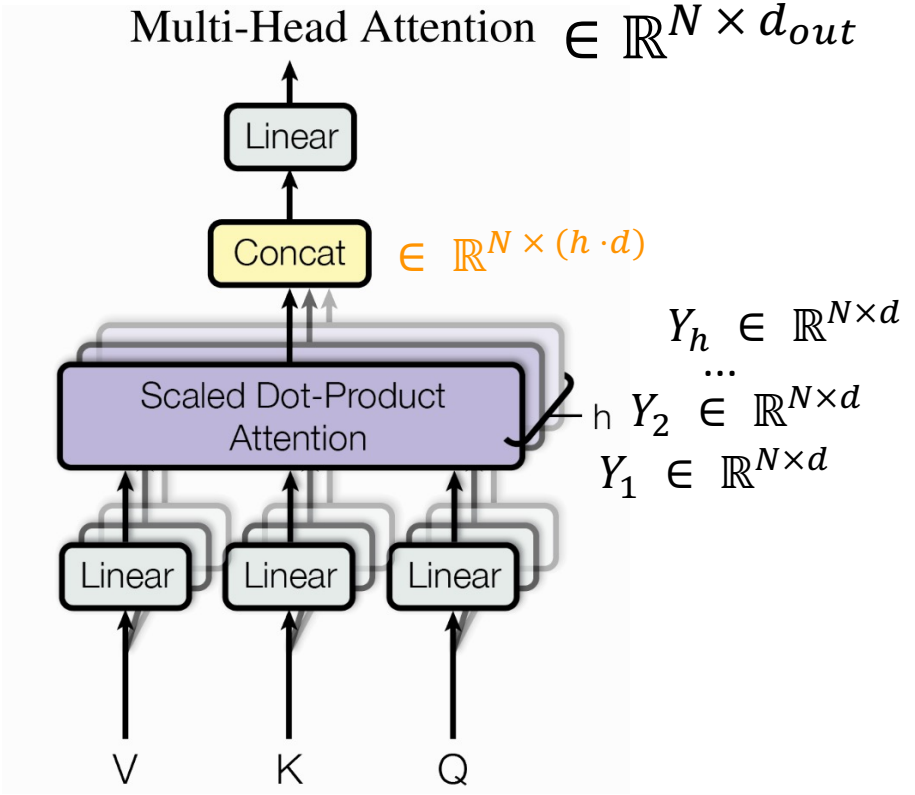
$Y_1 \in \mathbb{R}^{N \times d}$

Multi-head Attention

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$$

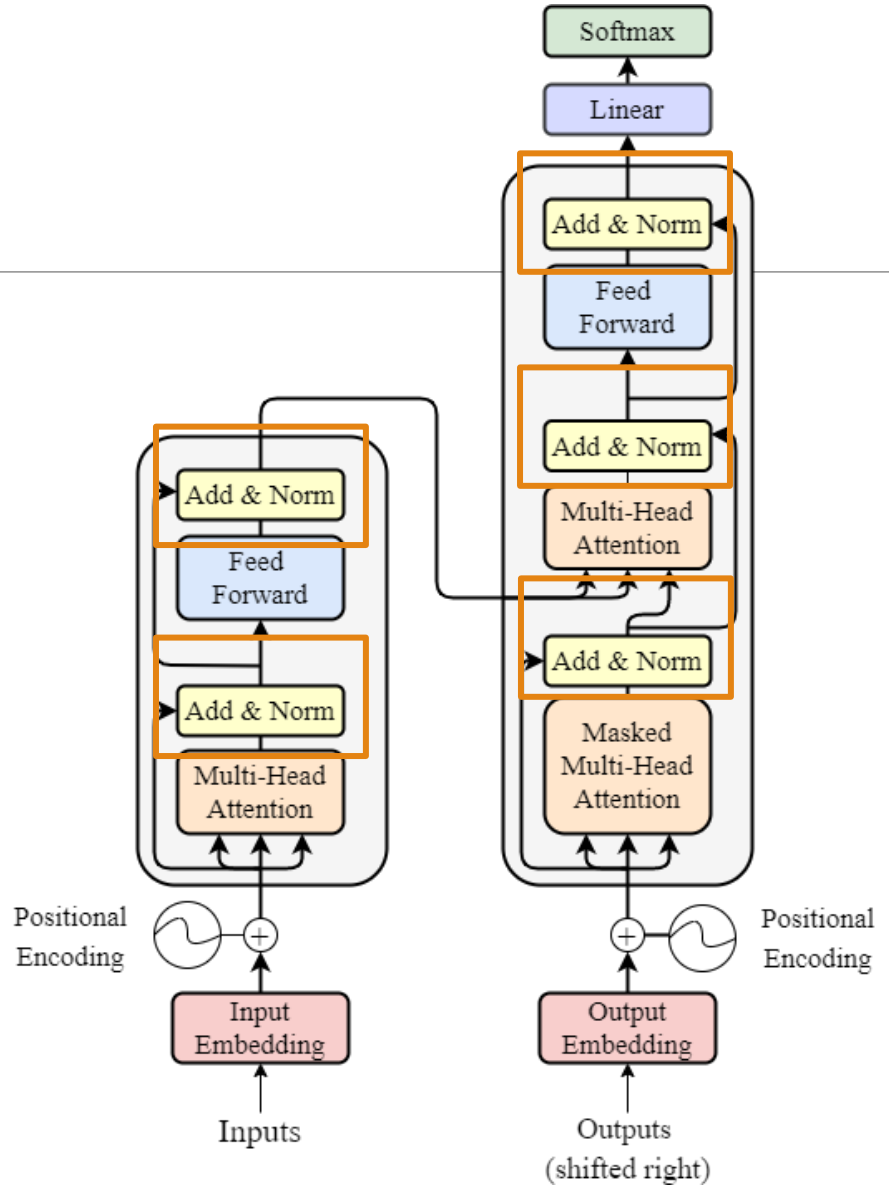
$\in \mathbb{R}^{N \times (h \cdot d)}$ $\in \mathbb{R}^{(h \cdot d) \times d_{out}}$

- With W^O, the concated head outputs can be projected to a preferred dimension (hyperparameter: d_{out}).
- No worries on how head# changes dimension of outputs!



Add & Norm

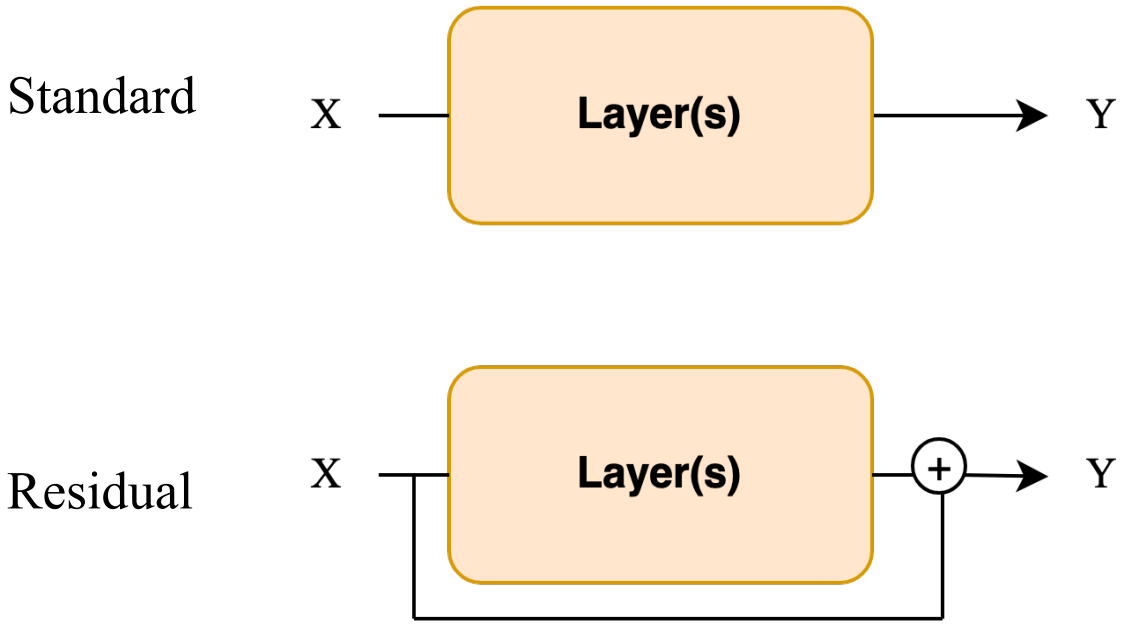
- Add: Residual Connection
- Norm: Layer Normalization



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Add & Norm

- Research shows that Residual Connection (He, Kaiming, et al., 2016) **stabilizes training**.
- Let the layers in between learn the residual (i.e. $Y - X$).



He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016).

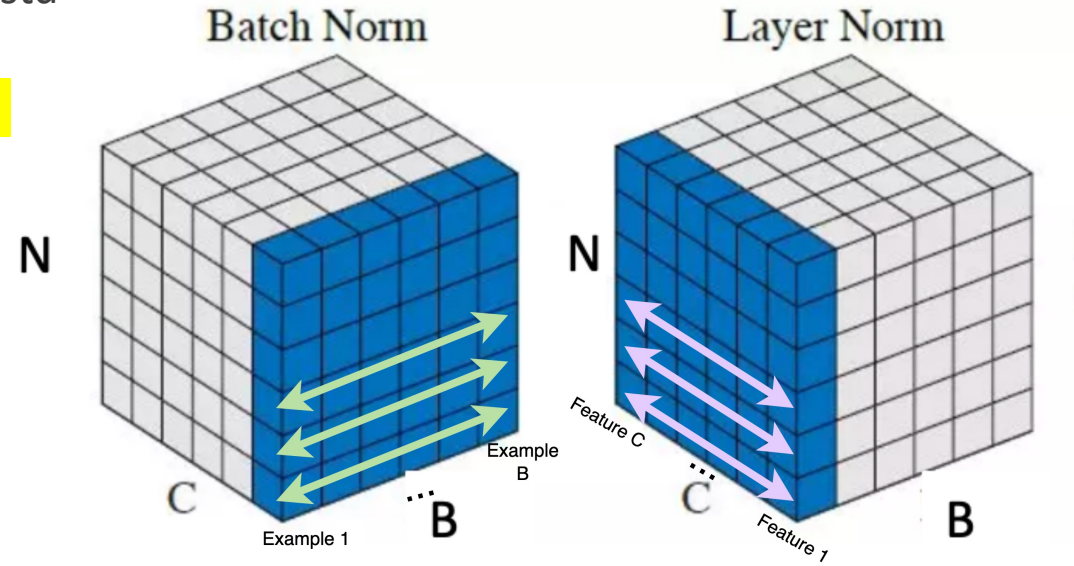
Add & Norm

B: Batch Size
C: Channels / Embedding Feature Dimension
N: Sequence length

- Normalization is also a way to stabilize training.
  - Batch Norm: For the same embedding dimension in the same sequence index, calculate its mean μ and std σ **across the samples within the same batch**.
  - Layer Norm: For an input vector, calculate its mean and std **across embedding dimension.**
  - $\mathbf{norm}(\boldsymbol{X}) = \frac{\boldsymbol{X}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}$
- Vaswani et al. (2017) used Layer Normalization.

Why LN?

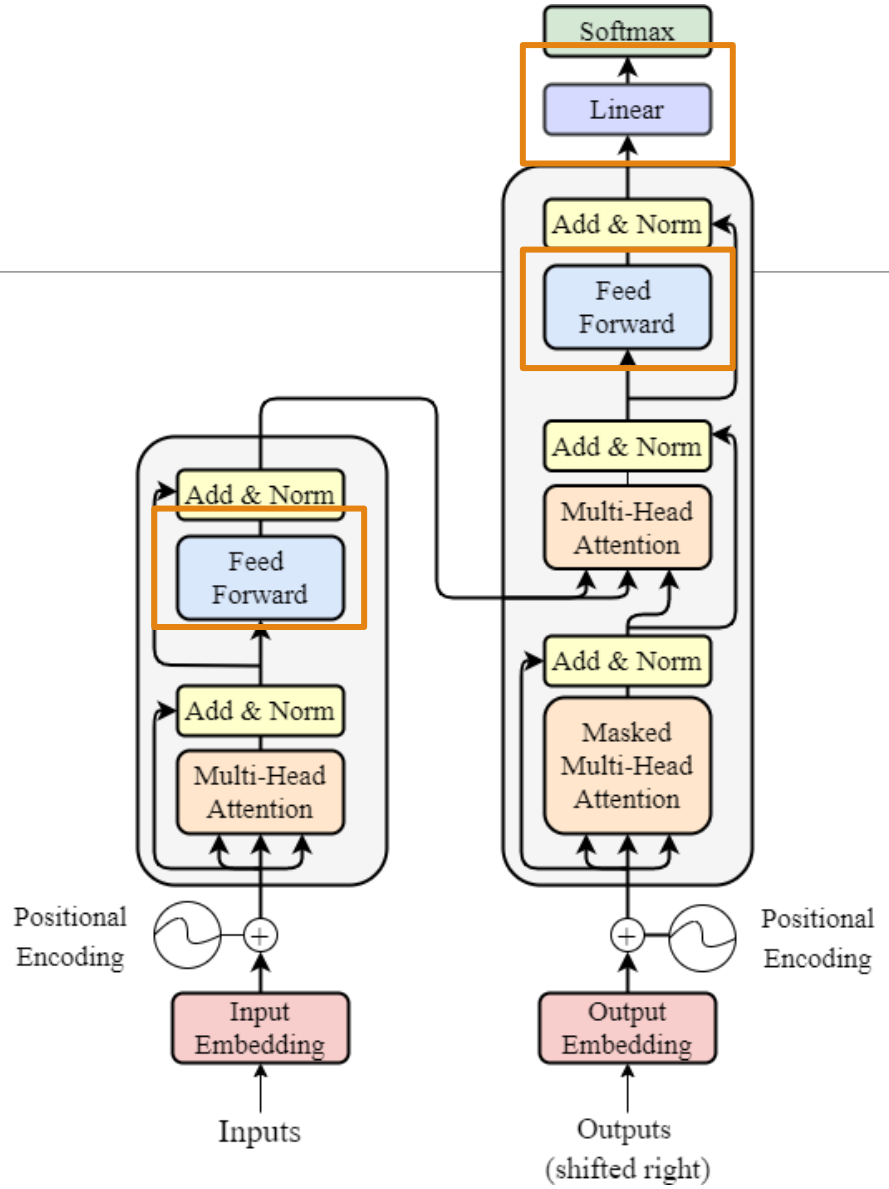
- No specific exp. from Vaswani et al., 2017.
- BN needs batch mean and std during inference, unsuitable for varying seq length in NLP.
- LN yields good results.



Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." *arXiv preprint arXiv:1607.06450* (2016).

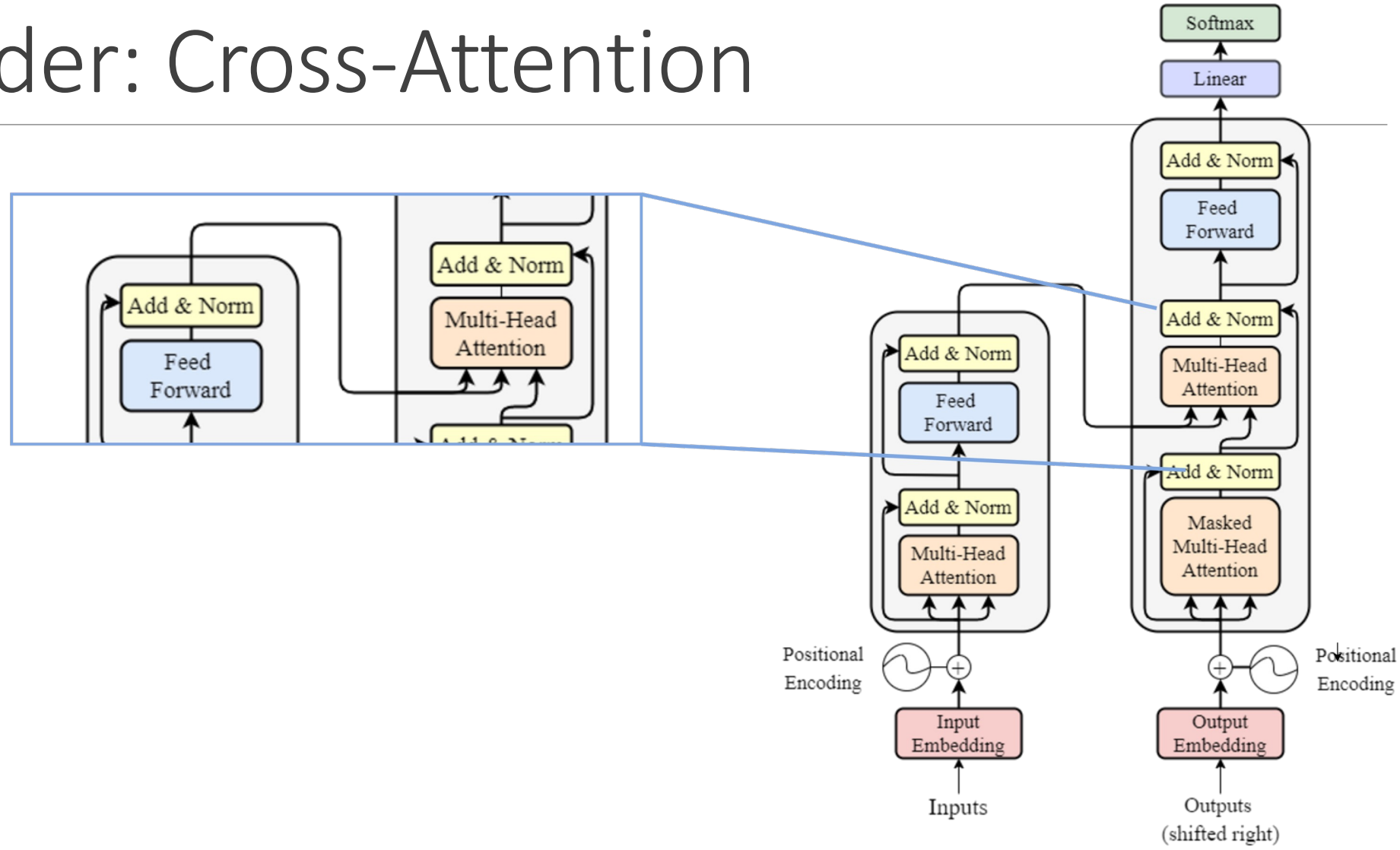
Feed Forward

- Simply multiply by a weight matrix.
- Used to project to a specific dimension.

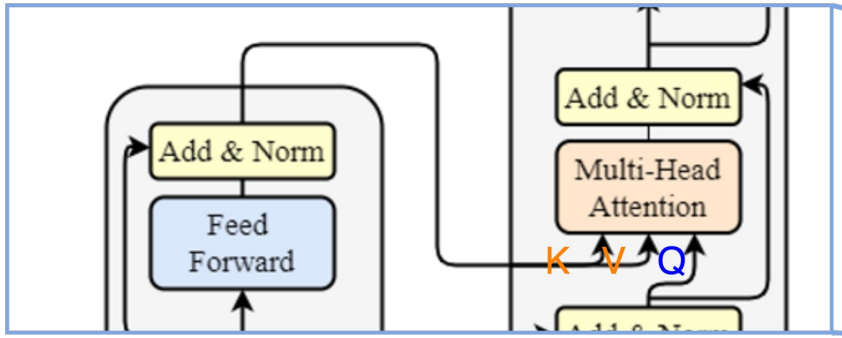


Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

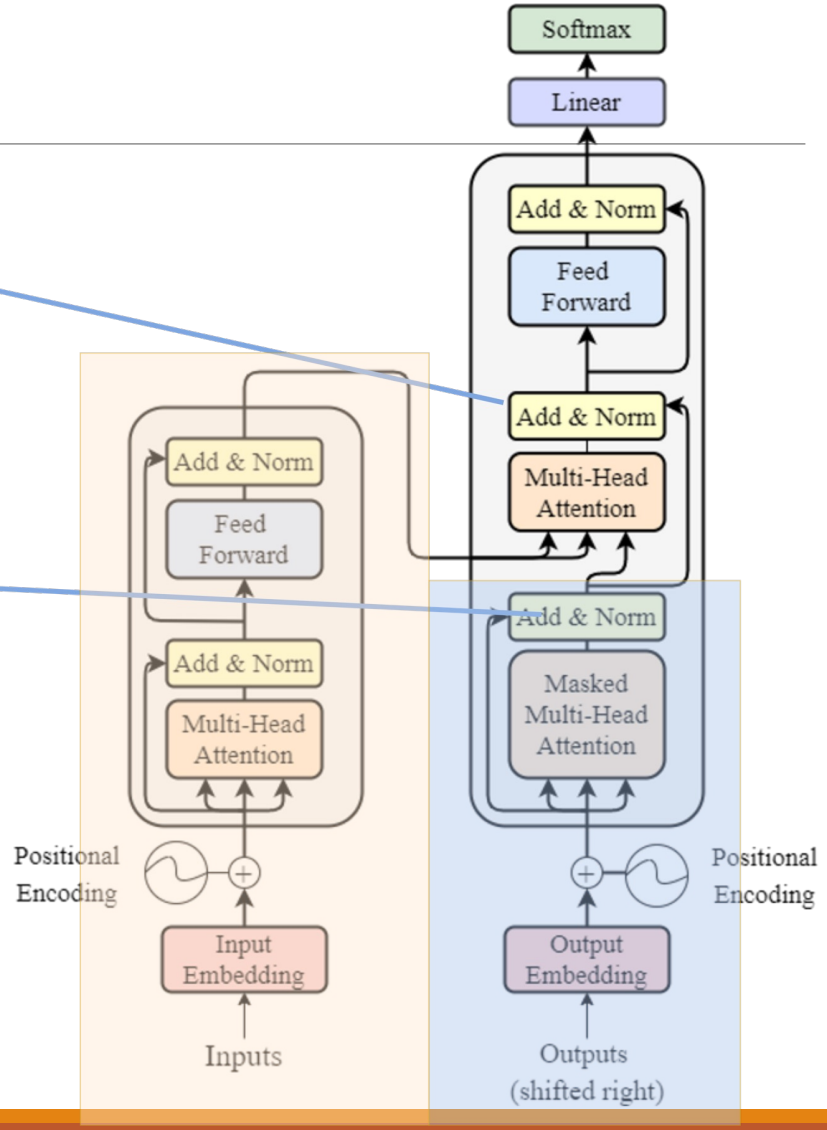
Decoder: Cross-Attention



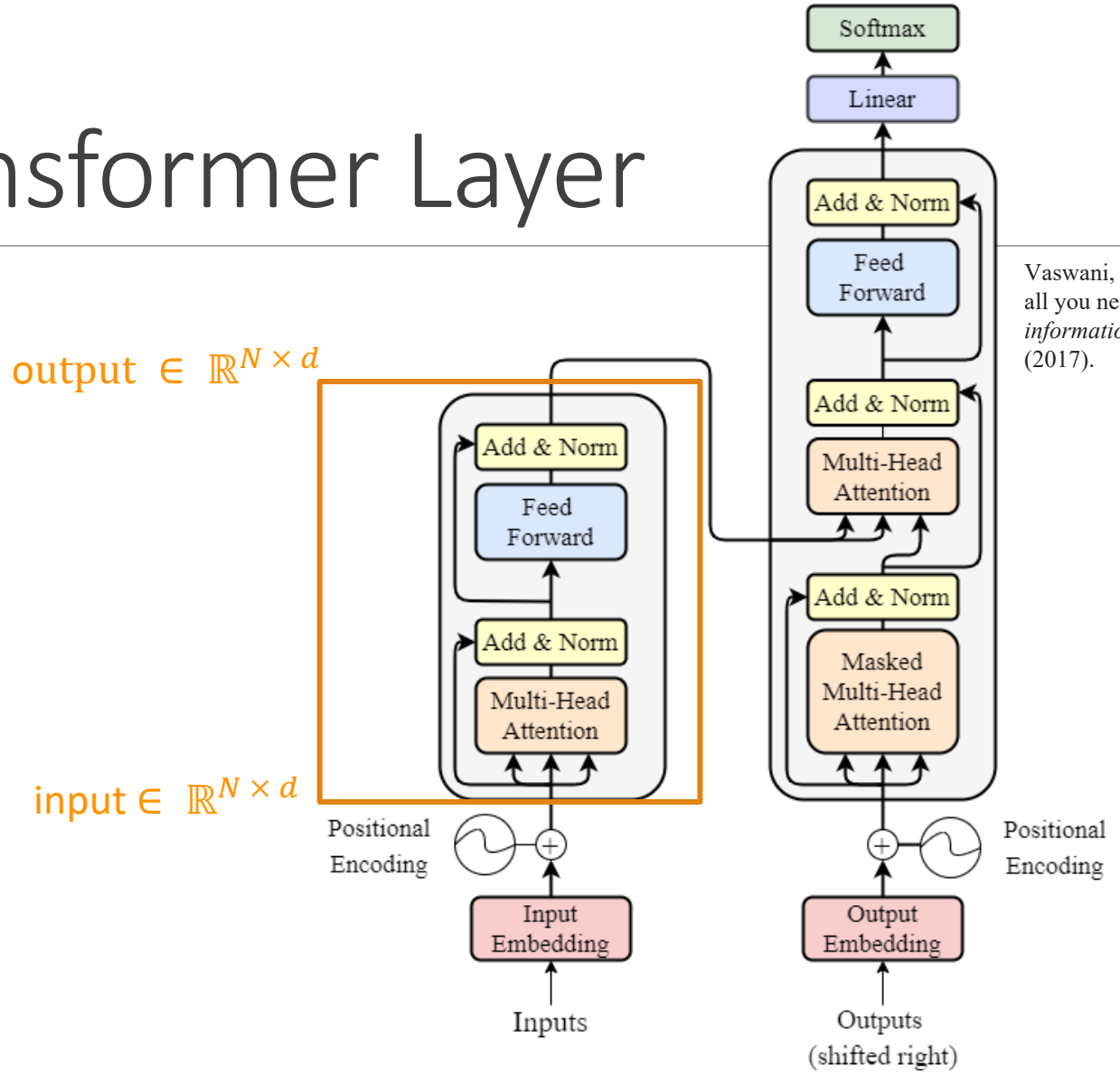
Decoder: Cross-Attention



K, V uses the output from encoder as their X;
Q uses the decoder's information as X.



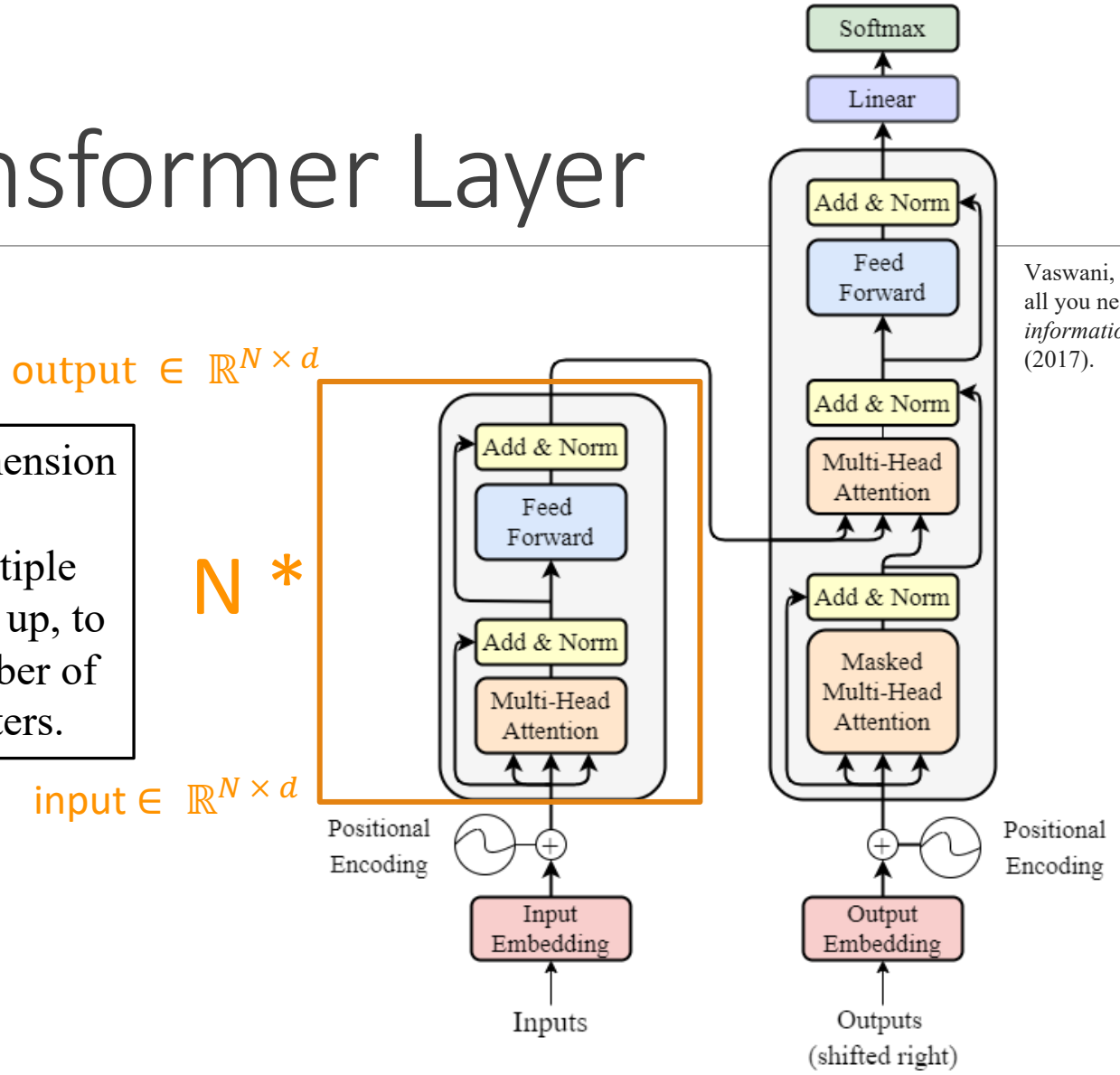
of Transformer Layer



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

of Transformer Layer

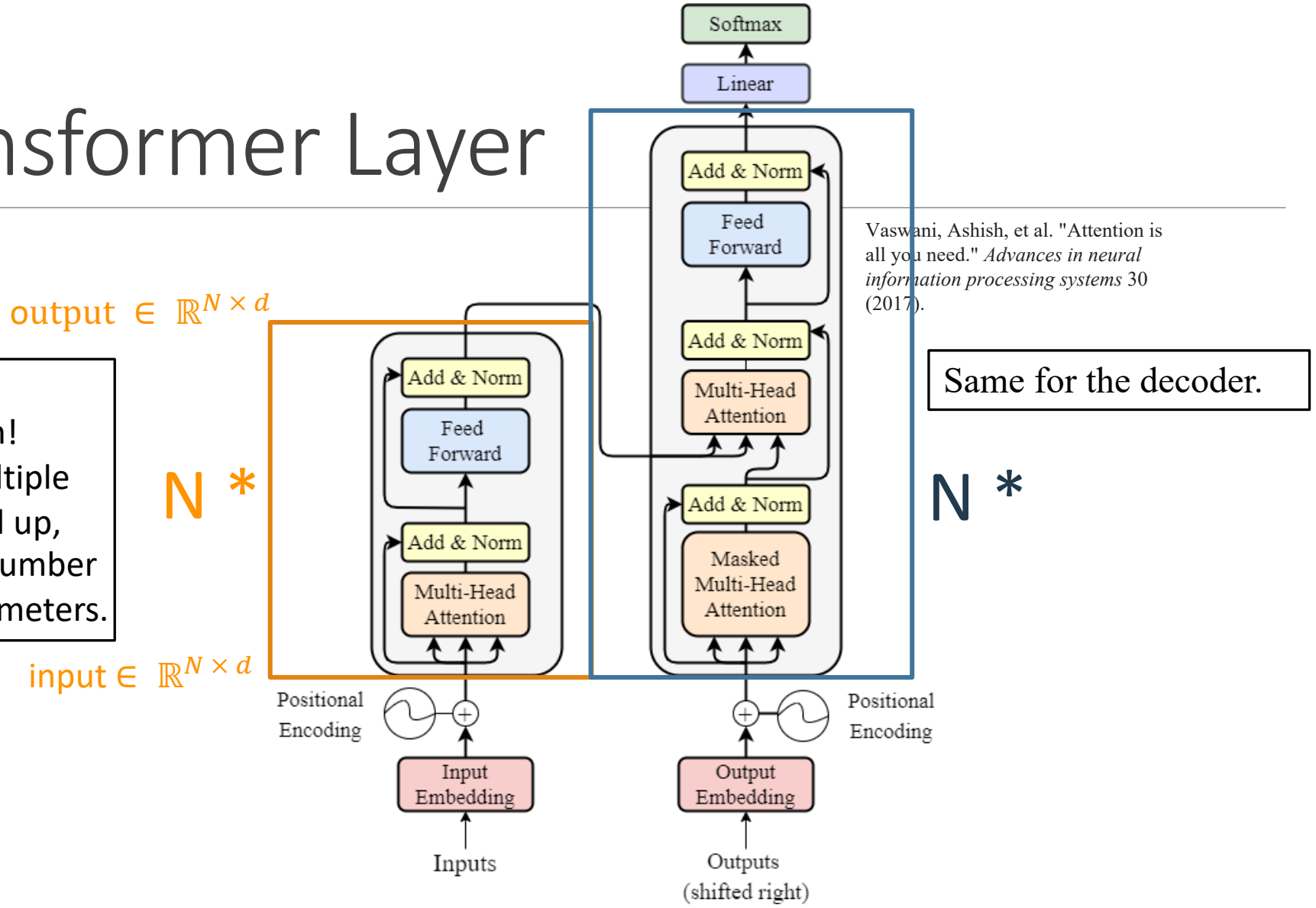
Input, output dimension match!
We can have multiple encoders stacked up, to increase the number of trainable parameters.



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

of Transformer Layer

Input, output dimension match!
We can have multiple encoders stacked up, to increase the number of trainable parameters.



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

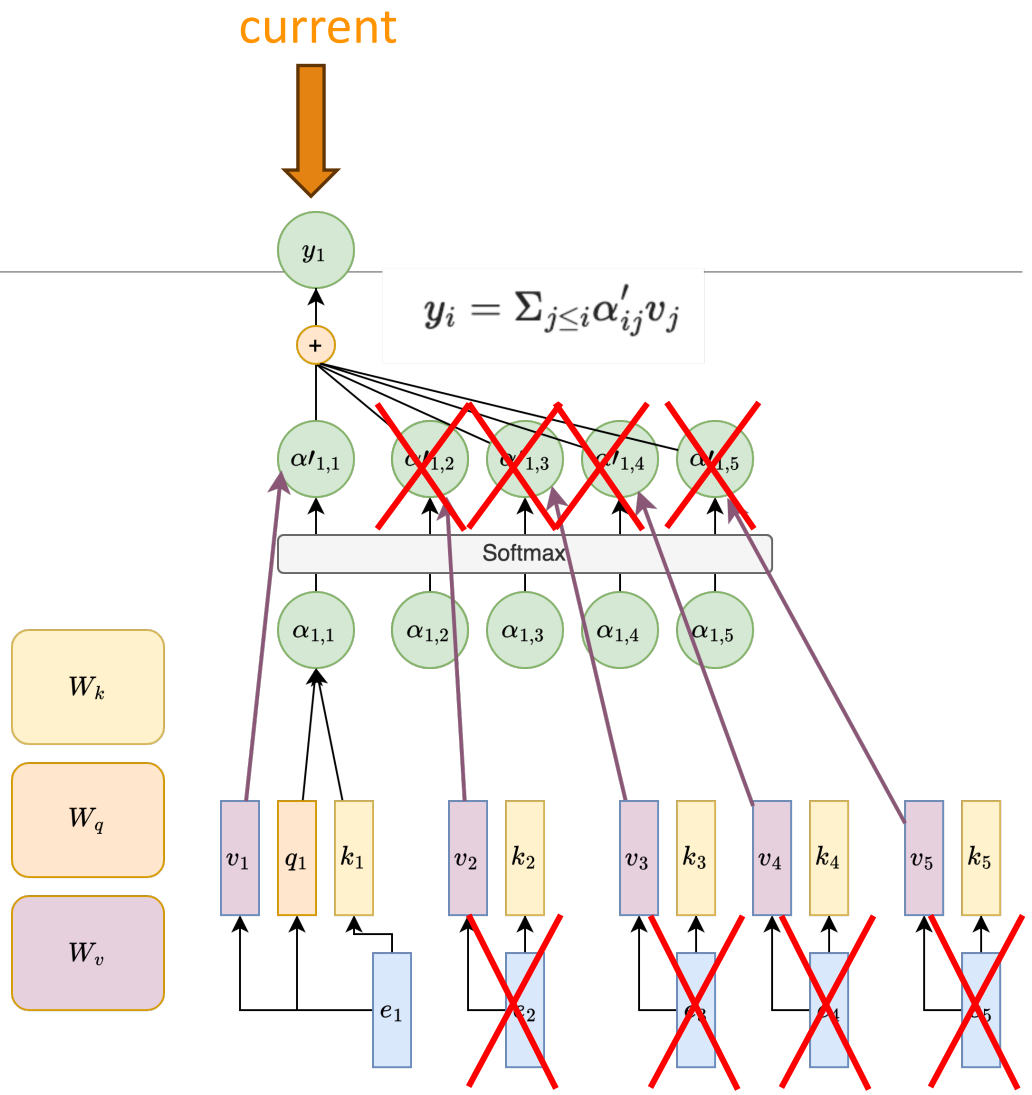
Same for the decoder.

Encoder & Decoder

- Recall that the decoder-encoder architecture accepts a sequence, encodes it, and then outputs another sequence.
  - Encoder sees the input sequence. Decoder **predicts** the output sequence.
- In transformer, both encoder and decoder applies the Multi-head Attention block(s).

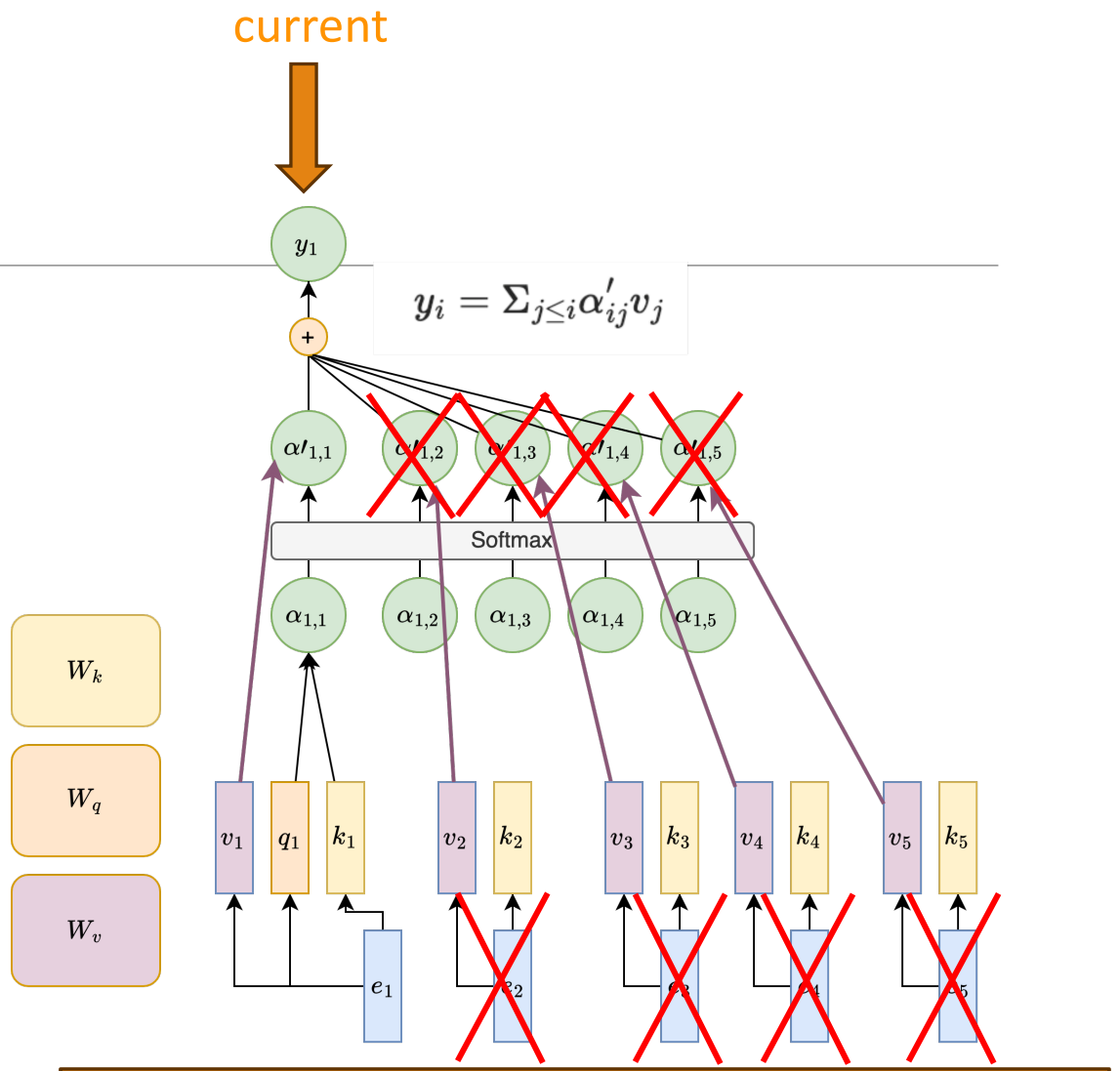
Encoder & Decoder

- Recall that the decoder-encoder architecture accepts a sequence, encodes it, and then outputs another sequence.
  - Encoder sees the input sequence. Decoder **predicts** the output sequence.
- In transformer, both encoder and decoder applies the Multi-head Attention block(s).



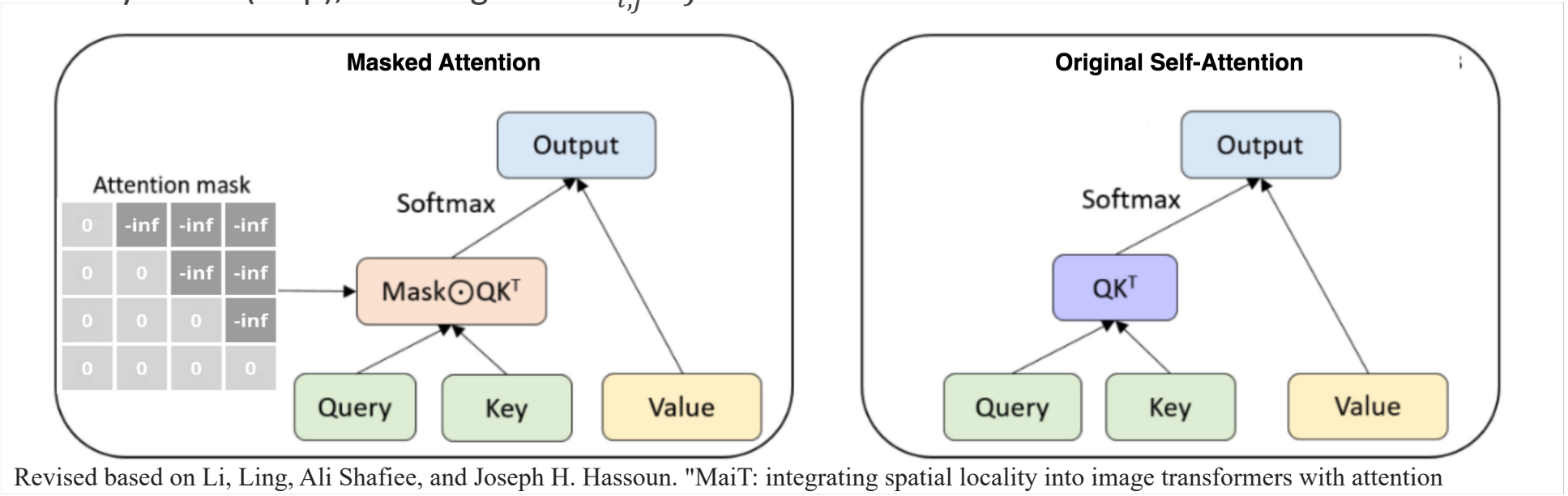
Encoder & Decoder

- Recall that the decoder-encoder architecture accepts a sequence, encodes it, and then outputs another sequence.
  - Encoder sees the input sequence. Decoder **predicts** the output sequence.
- In transformer, both encoder and decoder applies the Multi-head Attention block(s).
- However, according to the calculation of self-attention, **we need future tokens to calculate attention score for the current token.**



Decoder: Masked-Attention

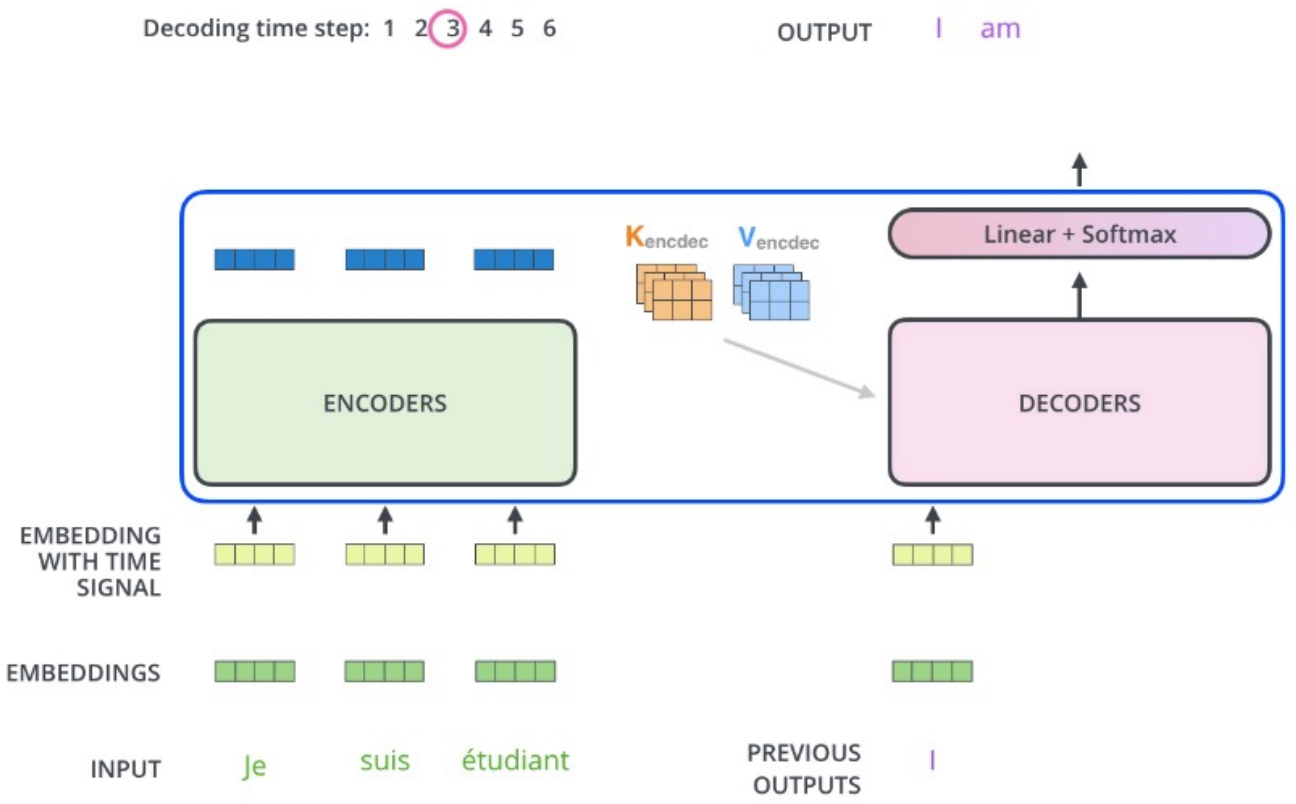
- We **mask the future tokens position in QK attention** before Softmax().
- By masking the right-upper triangle, **the decoder calculates attention scores only on the tokens it can see** (since $Softmax(-inf) = 0$).
  - For every token (step), it averages the $\alpha'_{i,j}$ $\forall\, j \leq i$.



Revised based on Li, Ling, Ali Shafiee, and Joseph H. Hassoun. "MaiT: integrating spatial locality into image transformers with attention masks." (2021).

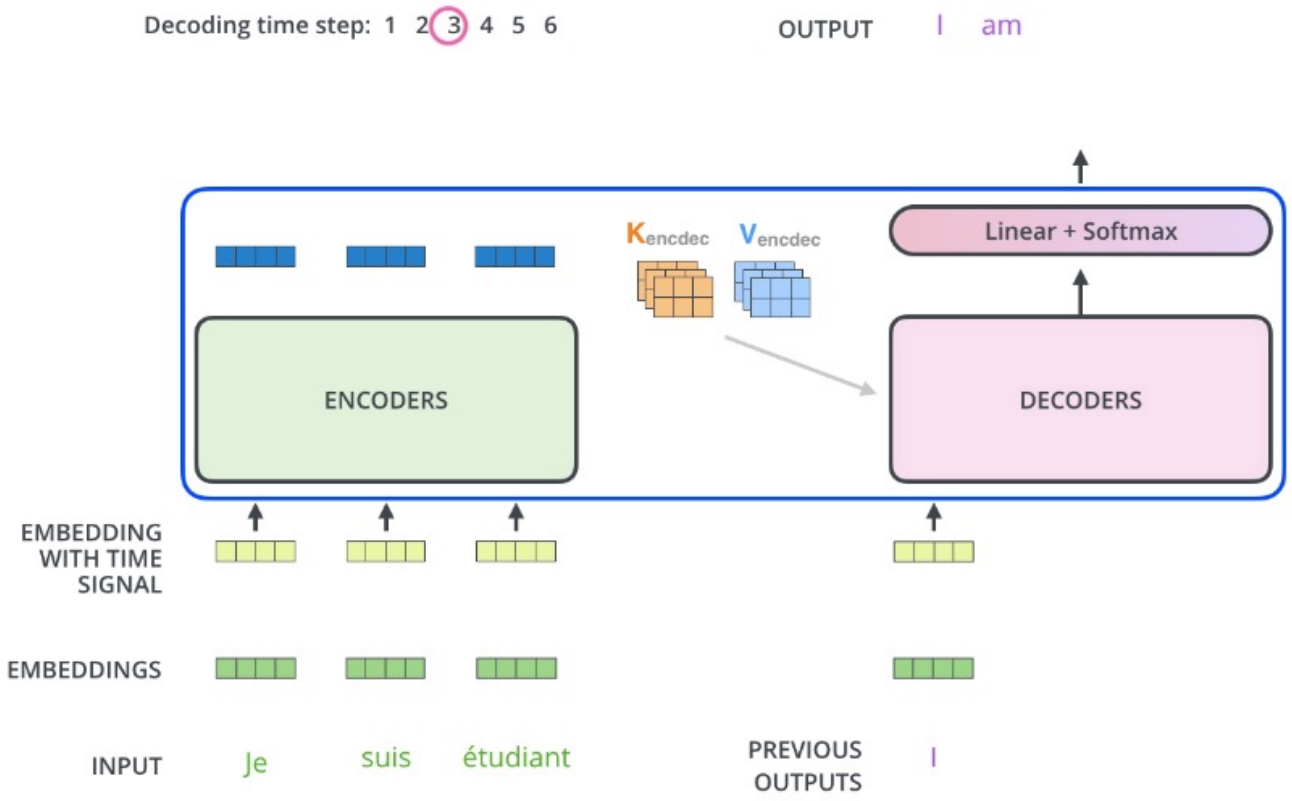
Decoder: Autoregressive Decoding

- In seq2seq problem, the decoder works by generating a token at one step.



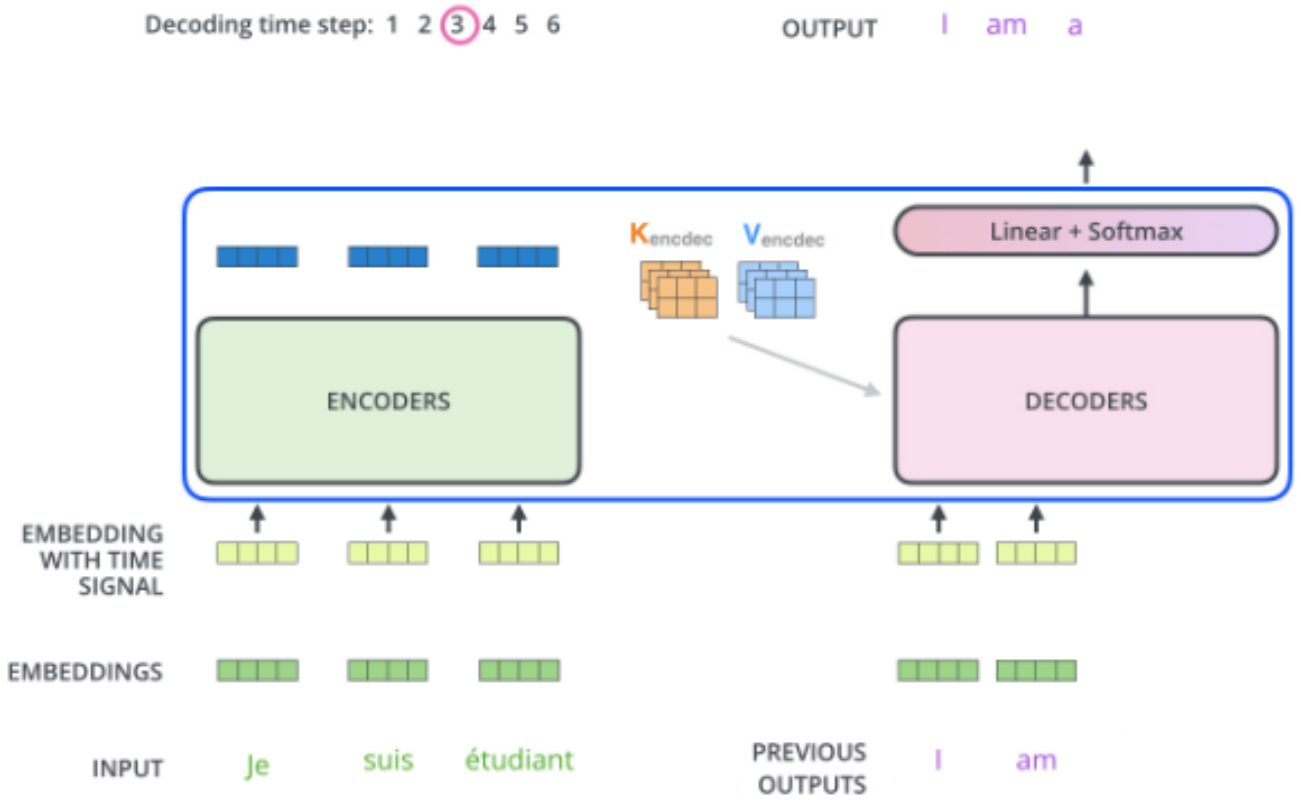
https://jalammar.github.io/illustrated-transformer/

Decoder: Autoregressive Decoding



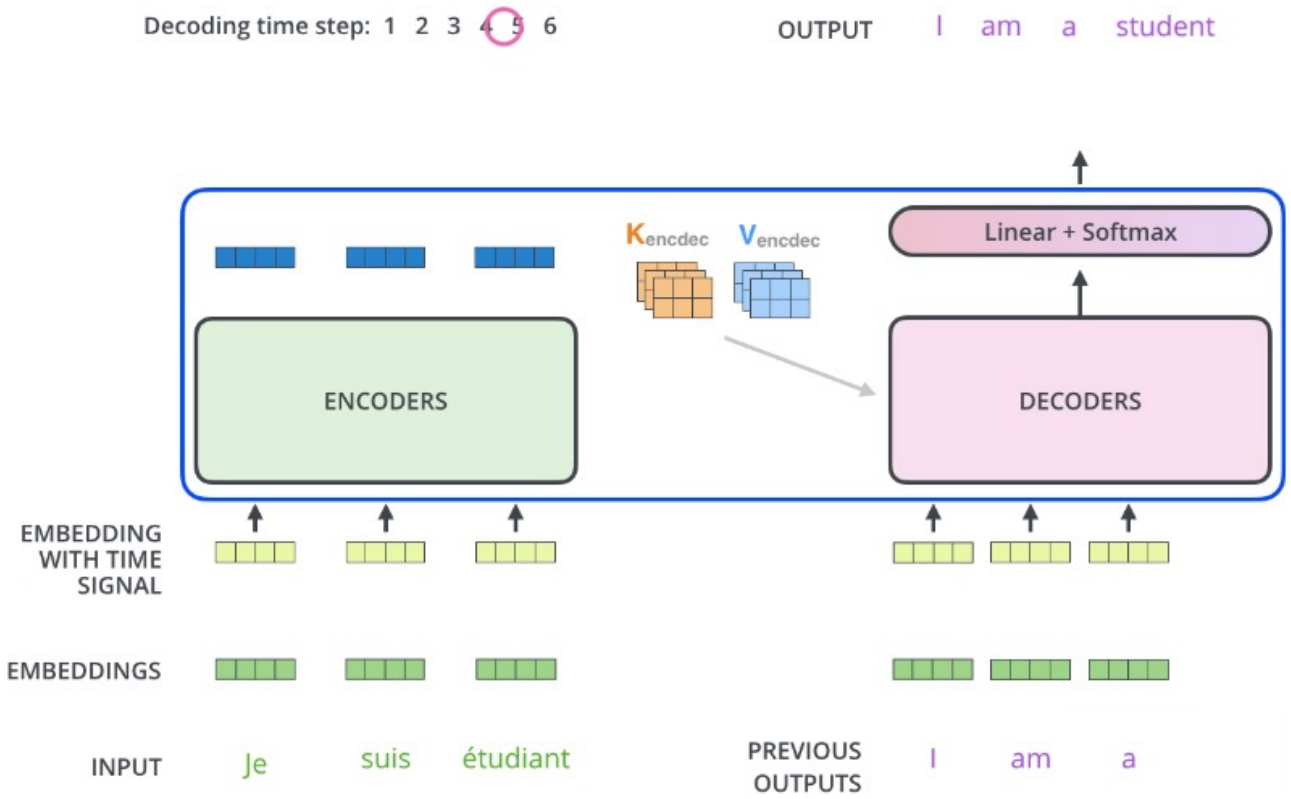
https://jalammar.github.io/illustrated-transformer/

Decoder: Autoregressive Decoding



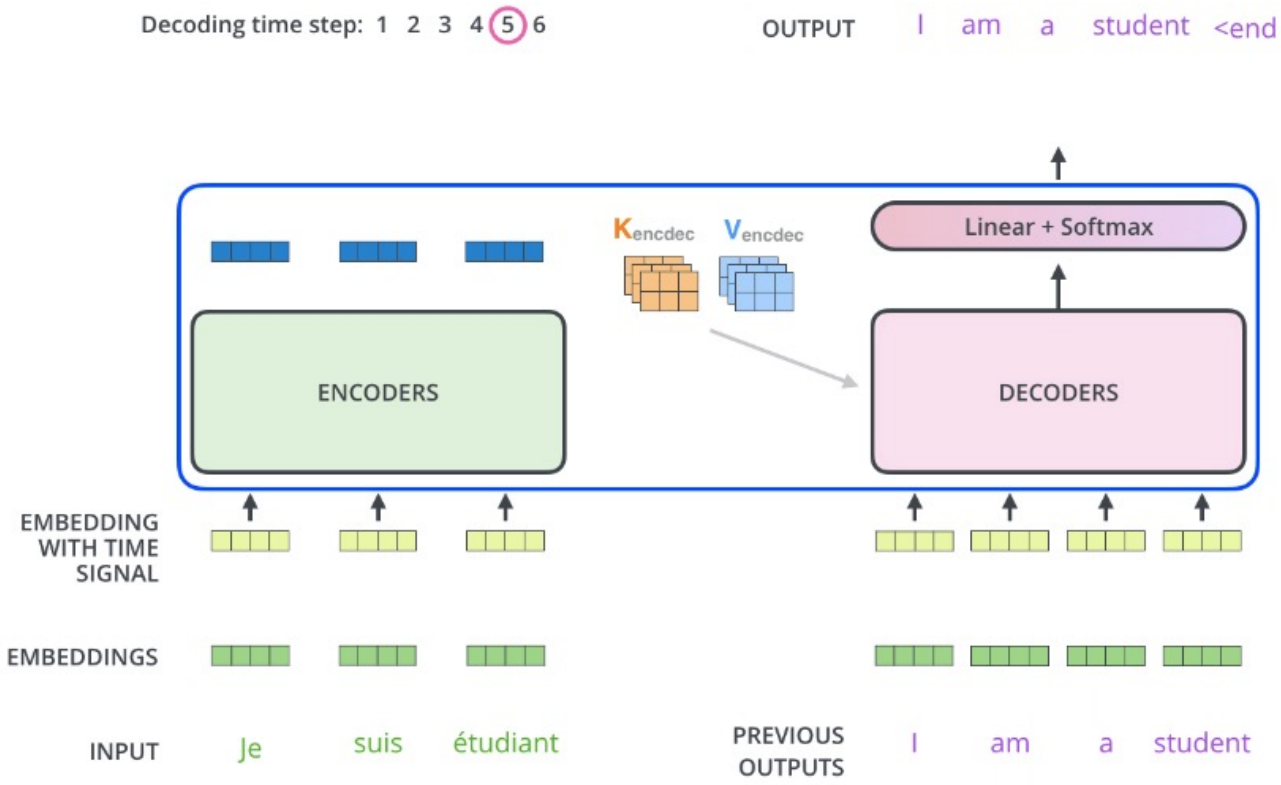
https://jalammar.github.io/illustrated-transformer/

Decoder: Autoregressive Decoding



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

Decoder: Autoregressive Decoding



<eos> marks the stopping point for the autoregressive process.

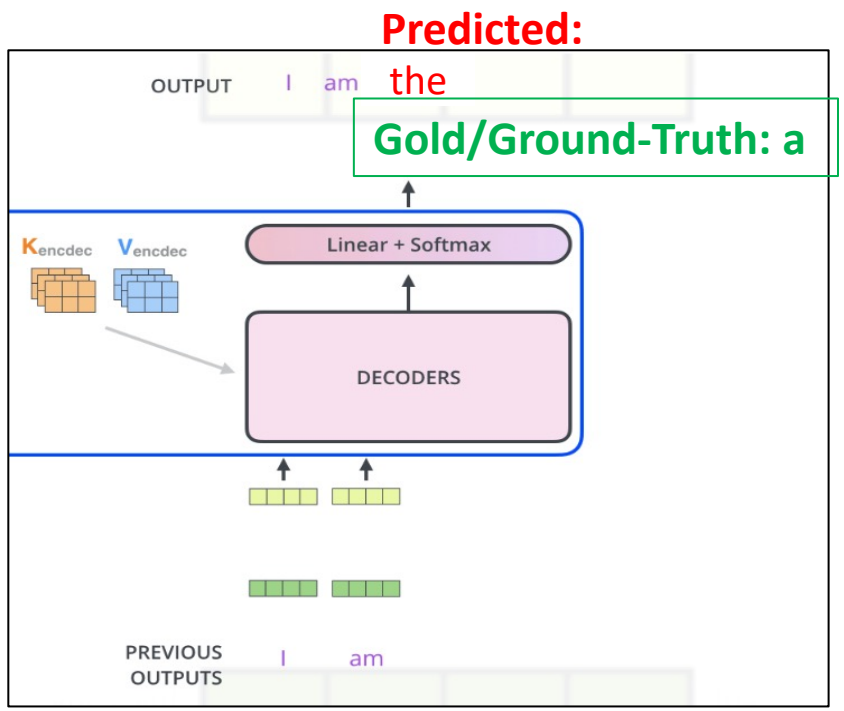
https://jalammar.github.io/illustrated-transformer/

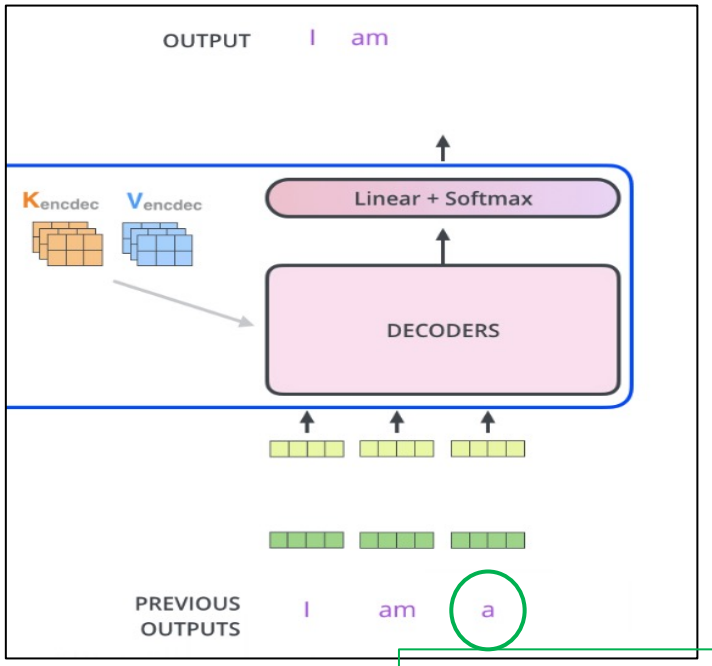
Decoder: Autoregressive Decoding

- In **testing stage**, generated token at i is then concated with previous tokens and fed into the model to predict token at $i + 1$.
- In **training stage**, **teacher forcing** is used (i.e. no matter what token is generated at step i, the gold token at i is used; it is concatenated and fed into the model to predict).

Decoding in **Training Stage**: Teacher-Forcing

Suppose at step 2 (0-indexed), the predicted highest-prob token is “the” instead of ground-truth “a” …





In training, output tokens are not important, only **word probability distributions** are used here for loss calculation

Use Gold token for input!!

Transformer's Achievements

1. In original paper, State-of-The-Art (SoTA) of **machine translation**.

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**	$2.3 \cdot 10^{19}$	

6 encoders,
6 decoders,
h = 512, 100K train steps.

6 encoders,
6 decoders,
h = 1024, 300K train steps.

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
https://arxiv.org/abs/1706.03762

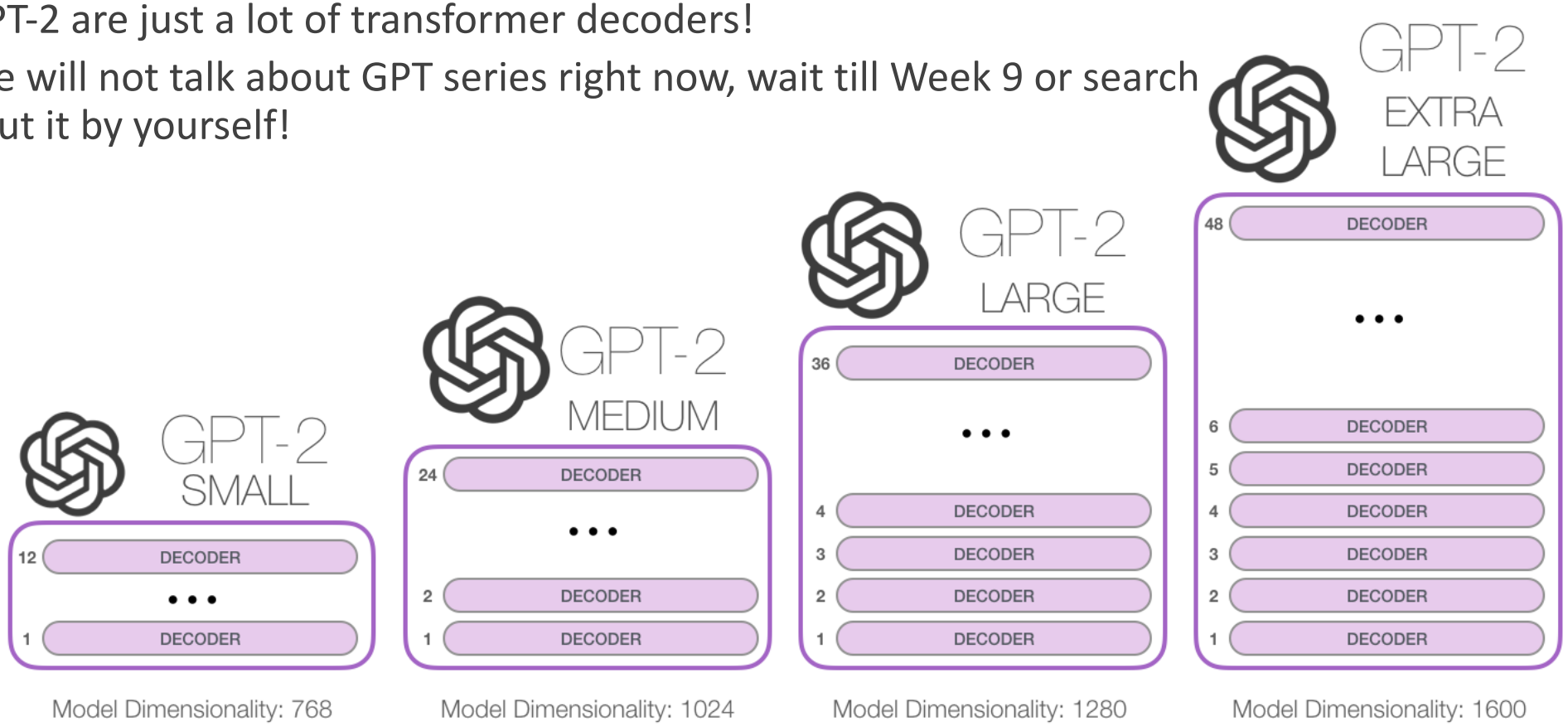
Transformer's Achievements

- (**Chat)GPT** is just a pretrained transformer (it is just n layers of transformer decoders).
- **Pretraining** is similar **to let the model read through a lot of books**, and **get a sense of the context word distribution so that it can infer the meaning.**
  - For more on **linguistic theory**, check "distributional hypothesis."
  - For more on **pretraining**, wait till next week (W7) 's topic of BERT and its Family!

Source: https://stock.adobe.com/tw/search?k=robot+reading&asset_id=570899814

GPT-2

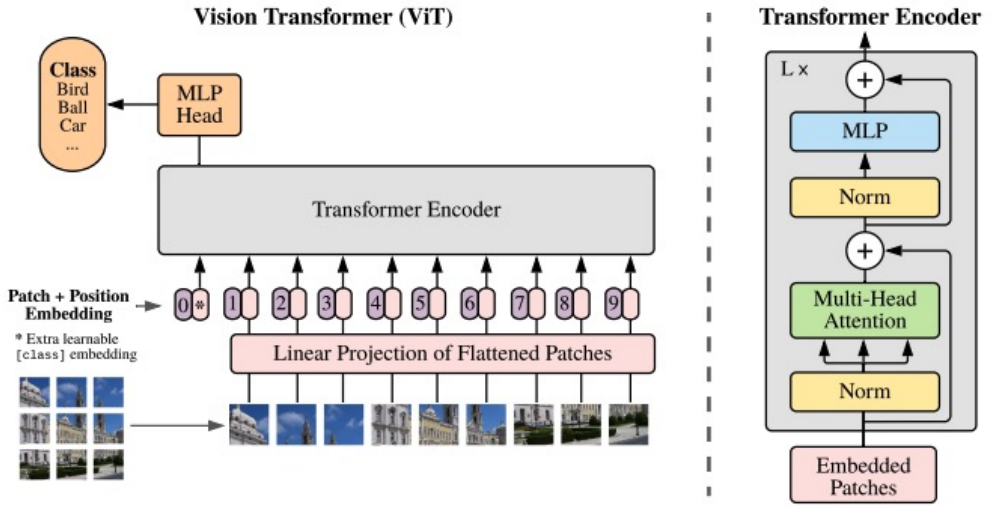
- GPT-2 are just a lot of transformer decoders!
- We will not talk about GPT series right now, wait till Week 9 or search about it by yourself!



Source: https://jalammar.github.io/illustrated-gpt2/

Vision Transformer (ViT)

- Google Research
- Apply transformer on vision tasks...
  - Split an image into image pathes.
  - Concat the pathes into a sequence.
  - Add positional encoding
  - Send the sequence to our transformer encoder. That's it!
- Outperform BiTL (ResNet with some improvements).
- Learn well on large-scale datasets.



Alexey Dosovitskiy, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." *International Conference on Learning Representations*. 2021.

Pretrained on which dataset

Testing on which dataset

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Transformer Variants (there are many more)

- **Universal Transformer**
  - Adds Adaptive Computation Time
  - Dehghani, Mostafa, et al. "Universal transformers." arXiv preprint arXiv:1807.03819 (2018).
- **Longformer**
  - Tackles long documents
  - Iz Beltagy, et al. (2020). Longformer: The Long-Document Transformer. arXiv:2004.05150.
- **Roformer**
  - Changes the design of positional encoding
  - Su, Jianlin, et al. "RoFormer: Enhanced Transformer with Rotary Position Embedding. CoRR abs/2104.09864 (2021)." arXiv preprint arXiv:2104.09864 (2021).
- **Vision Transformer**
  - Tackles vision tasks
  - Alexey Dosovitskiy, et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” *International Conference on Learning Representations*. 2021.

Q & A