

# Outline

- Issues with RNNs
- Comparison with Transformers



# Neural Machine Translation



wie

sind

sie

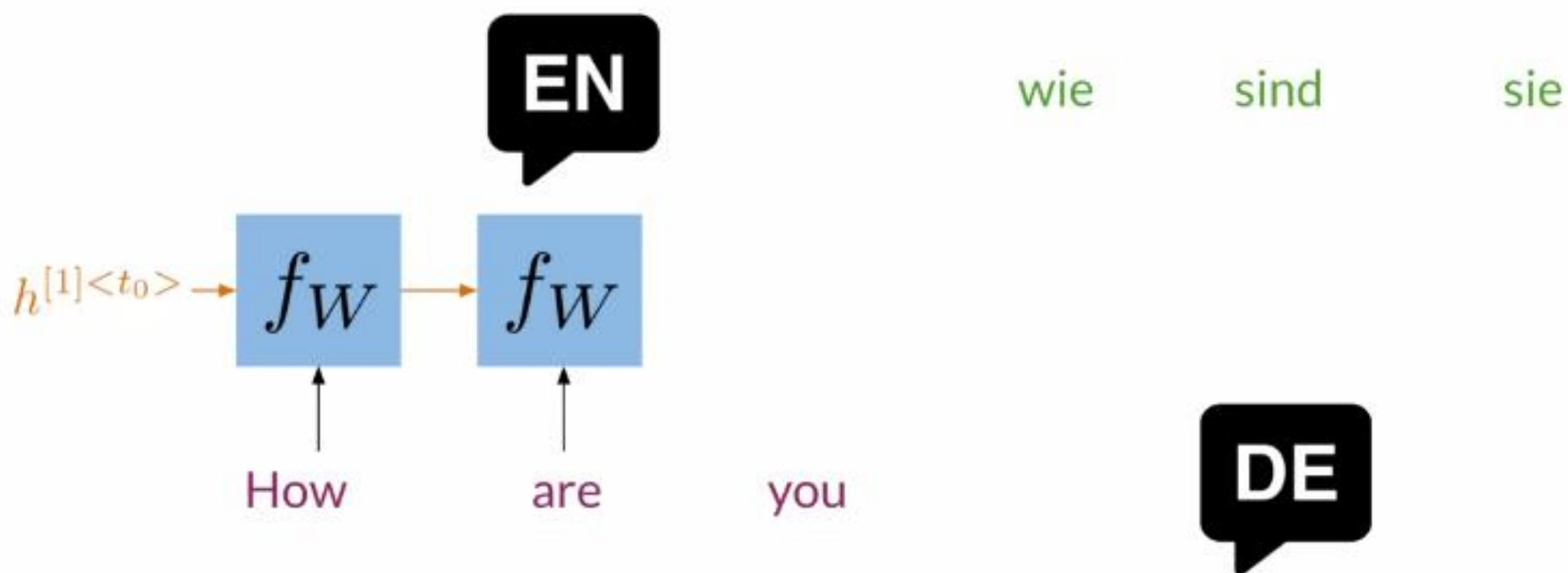
How are you



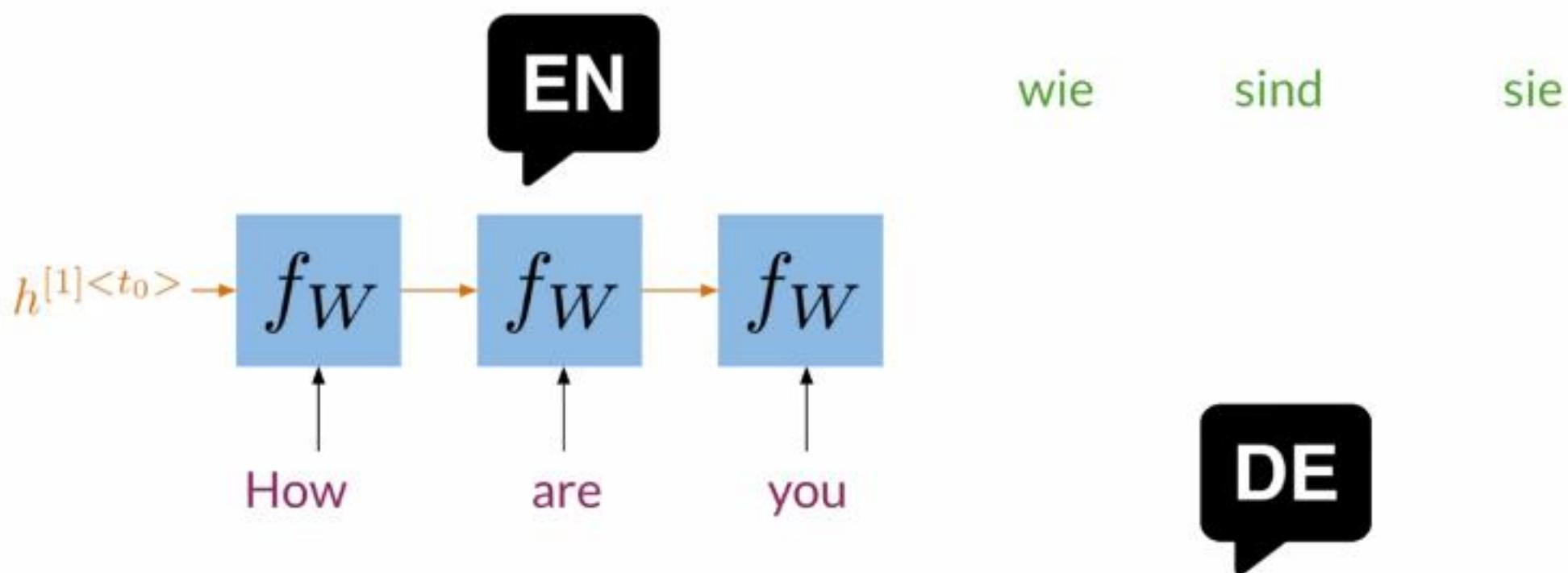
# Neural Machine Translation



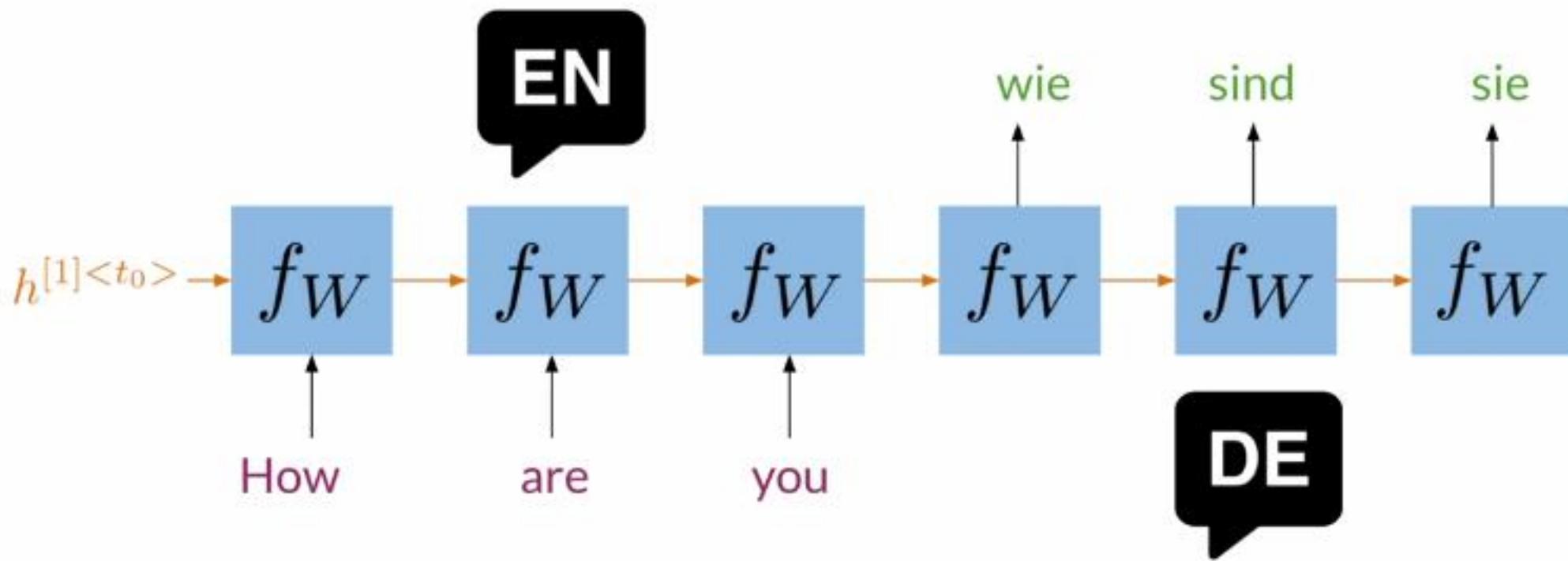
# Neural Machine Translation



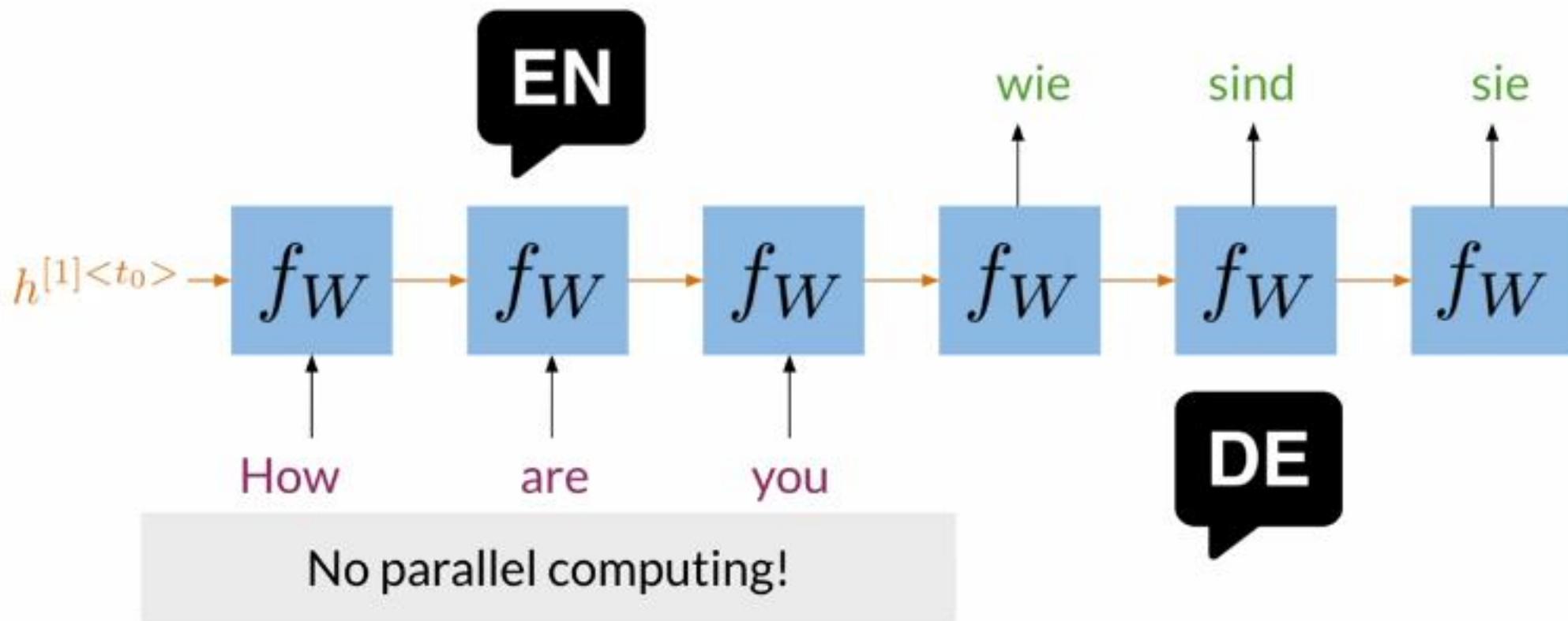
# Neural Machine Translation



# Neural Machine Translation

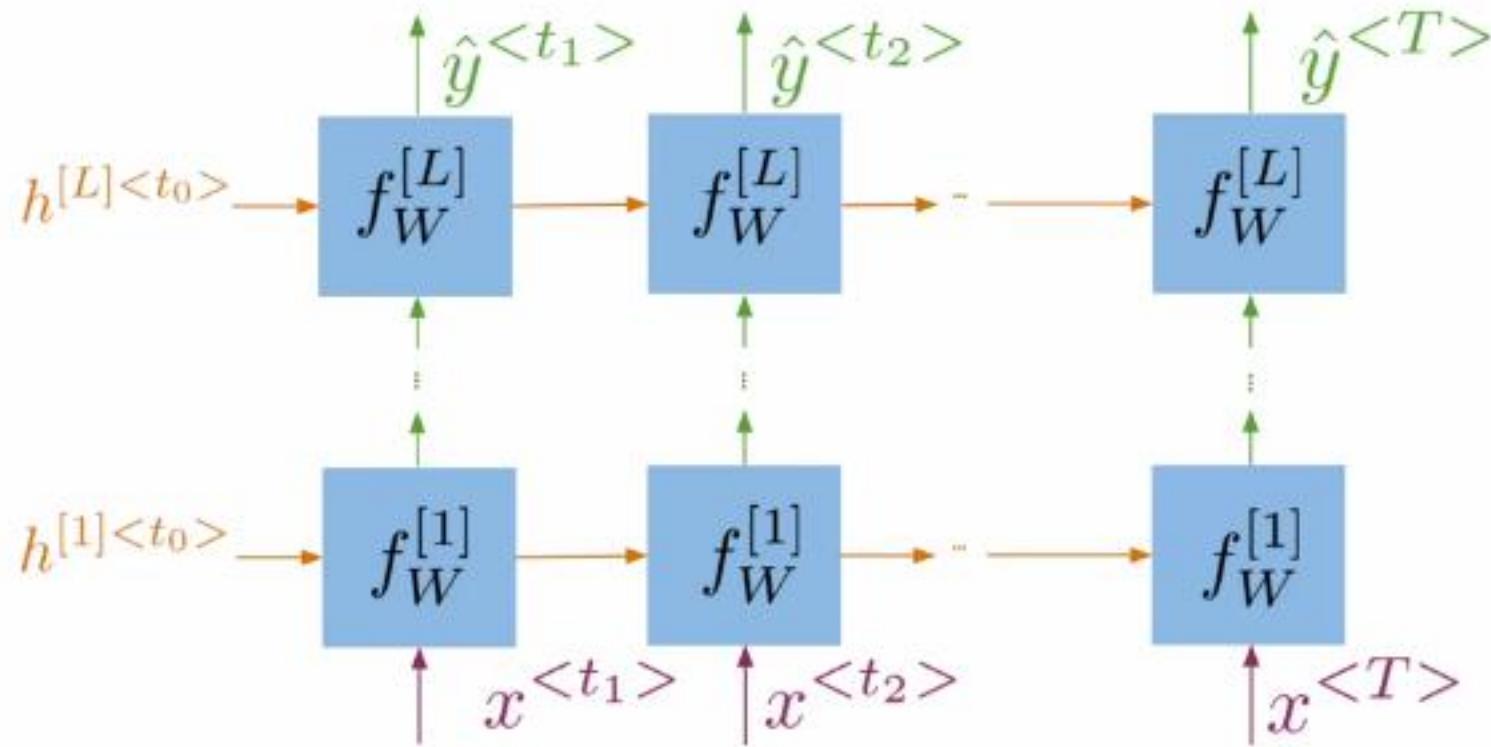


# Neural Machine Translation

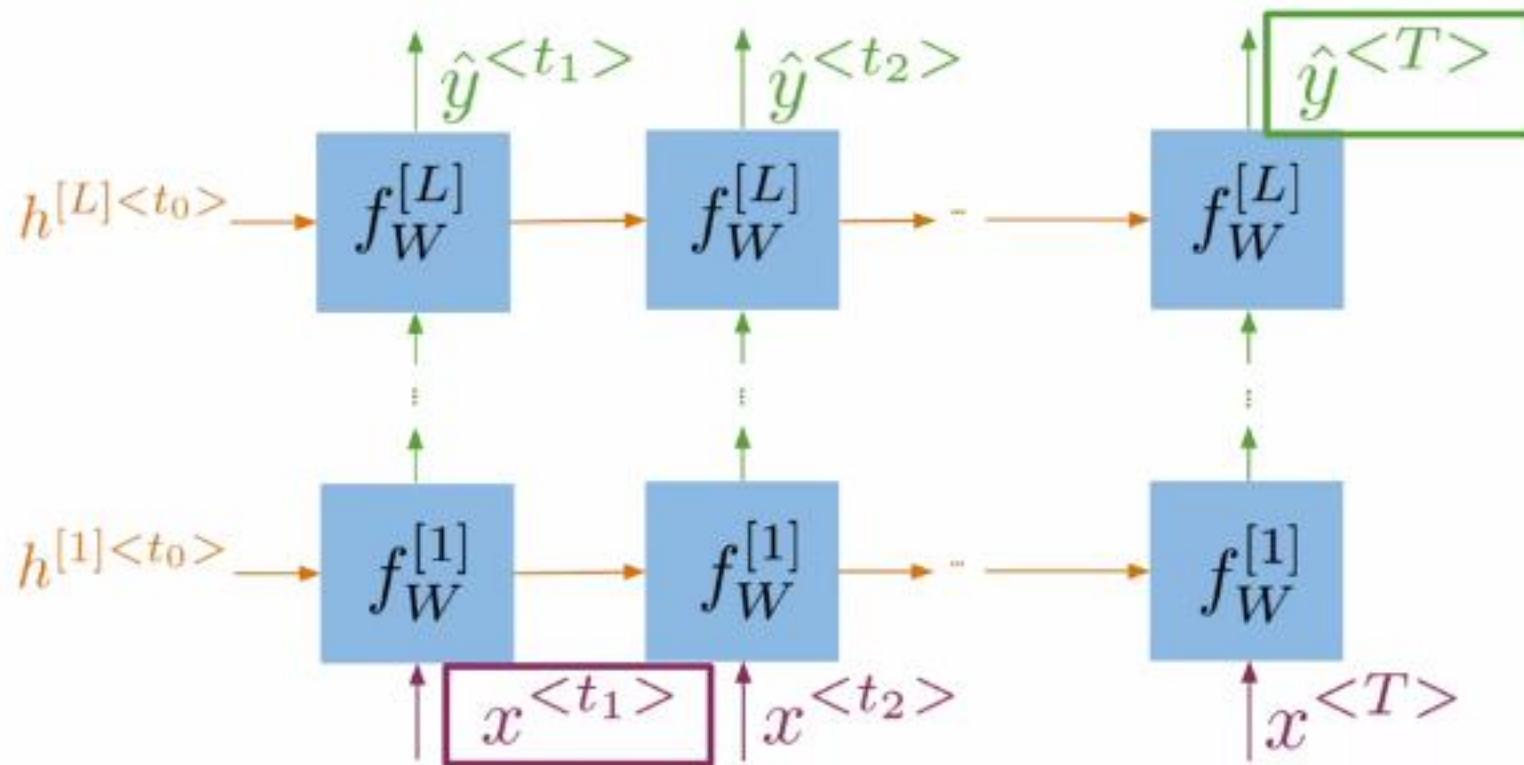


# Seq2Seq Architectures

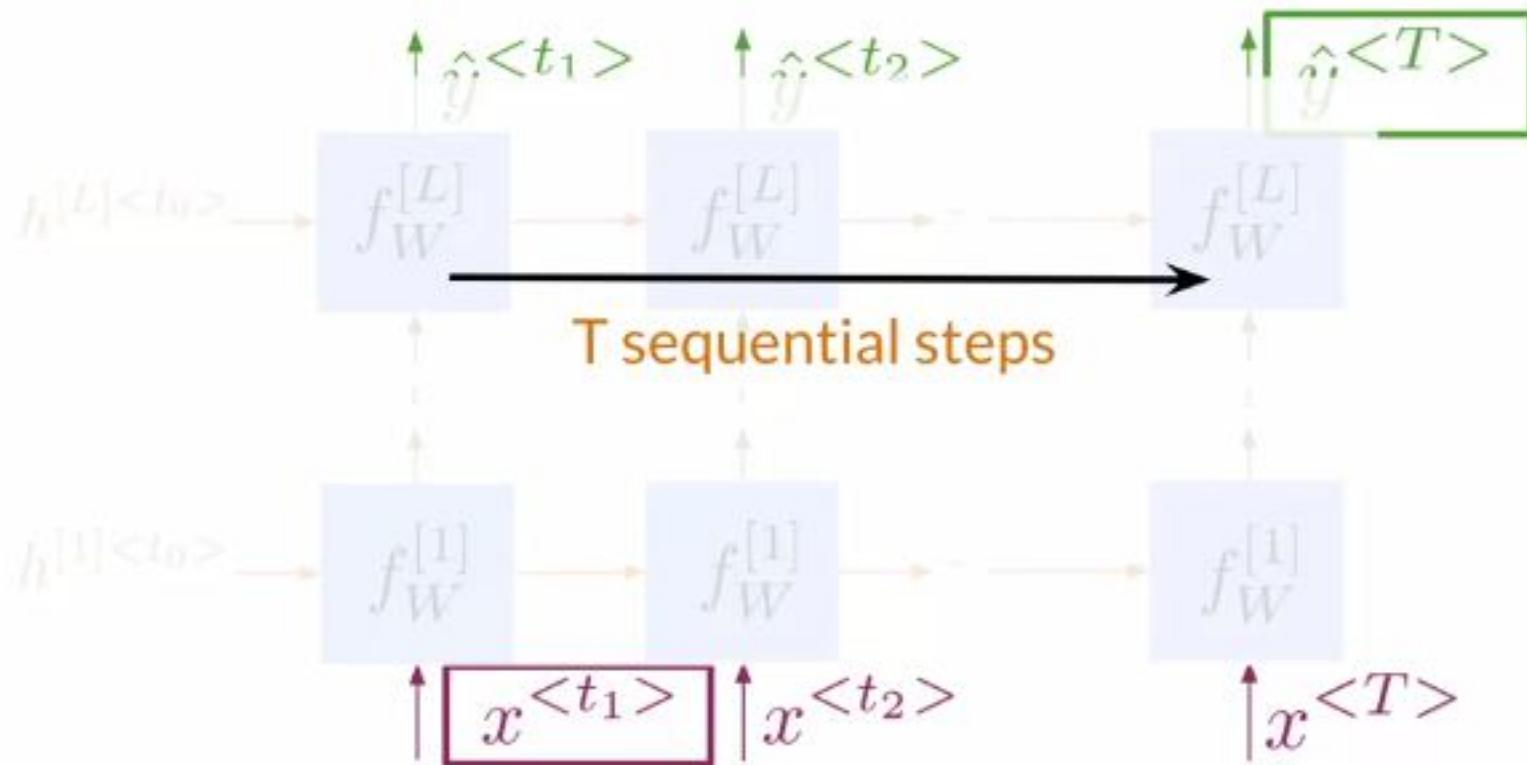
# Seq2Seq Architectures



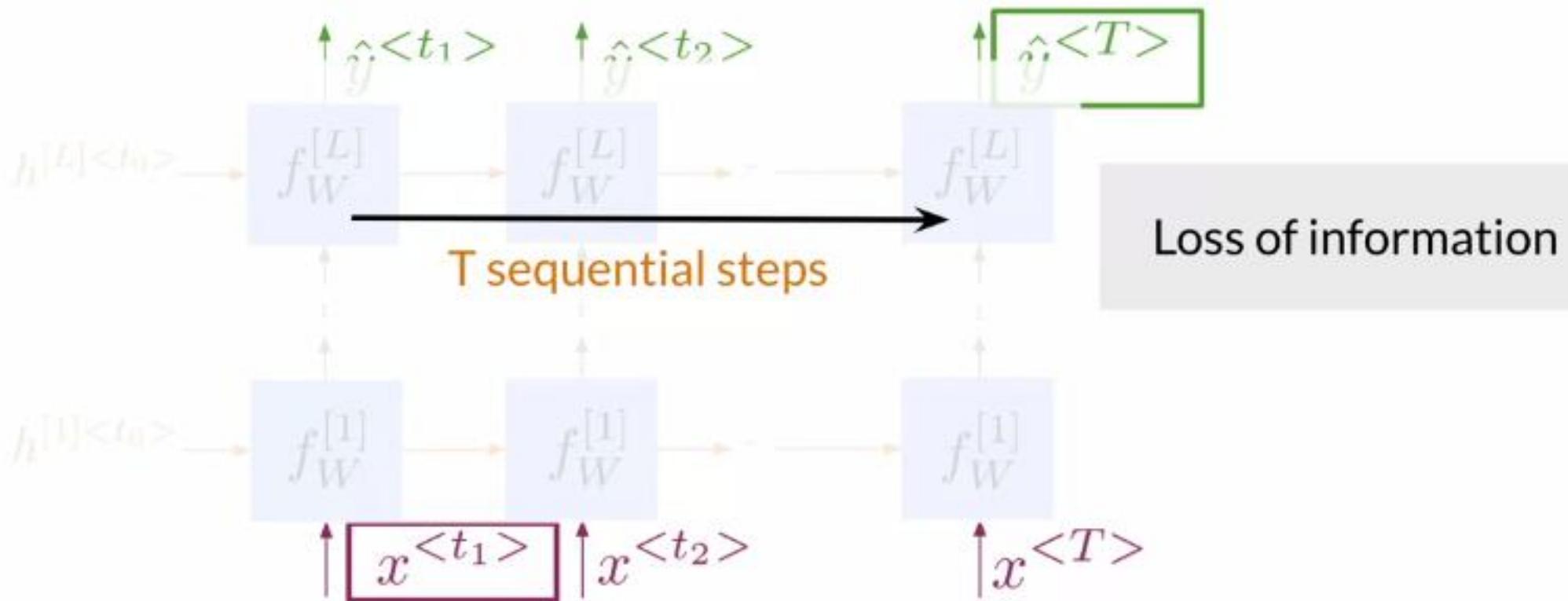
# Seq2Seq Architectures



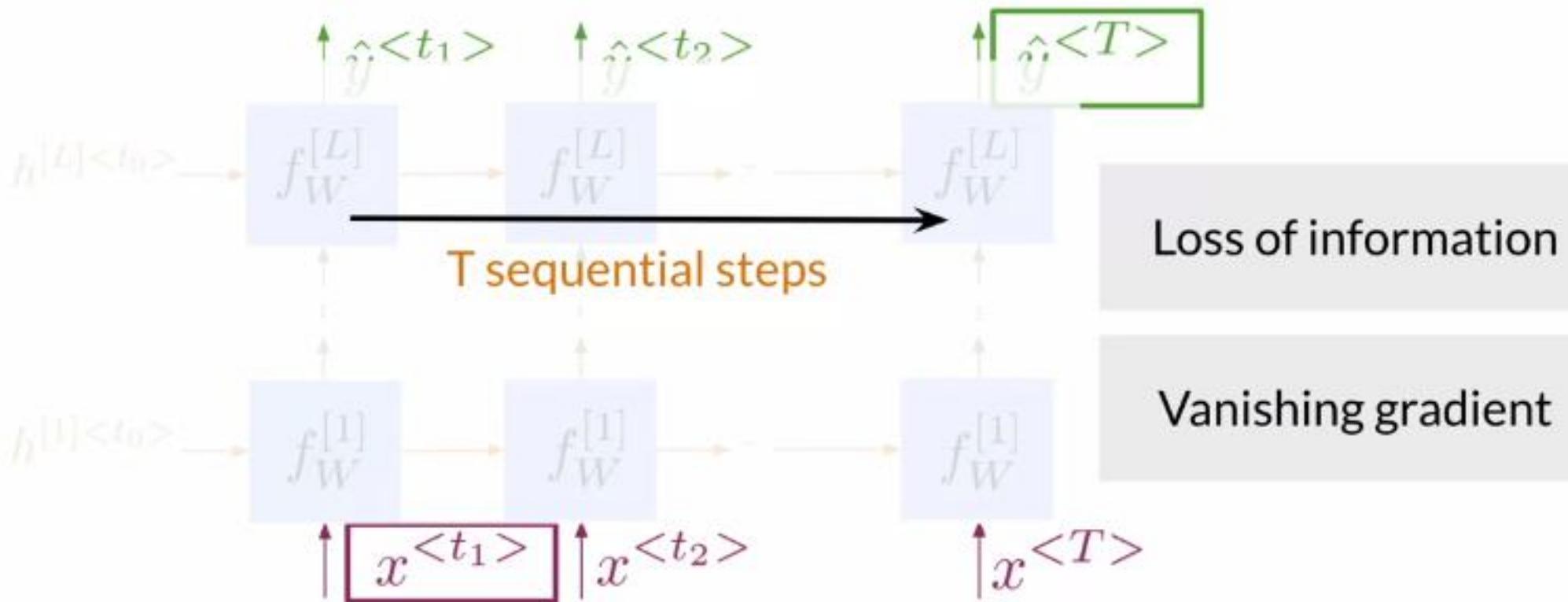
# Seq2Seq Architectures



# Seq2Seq Architectures

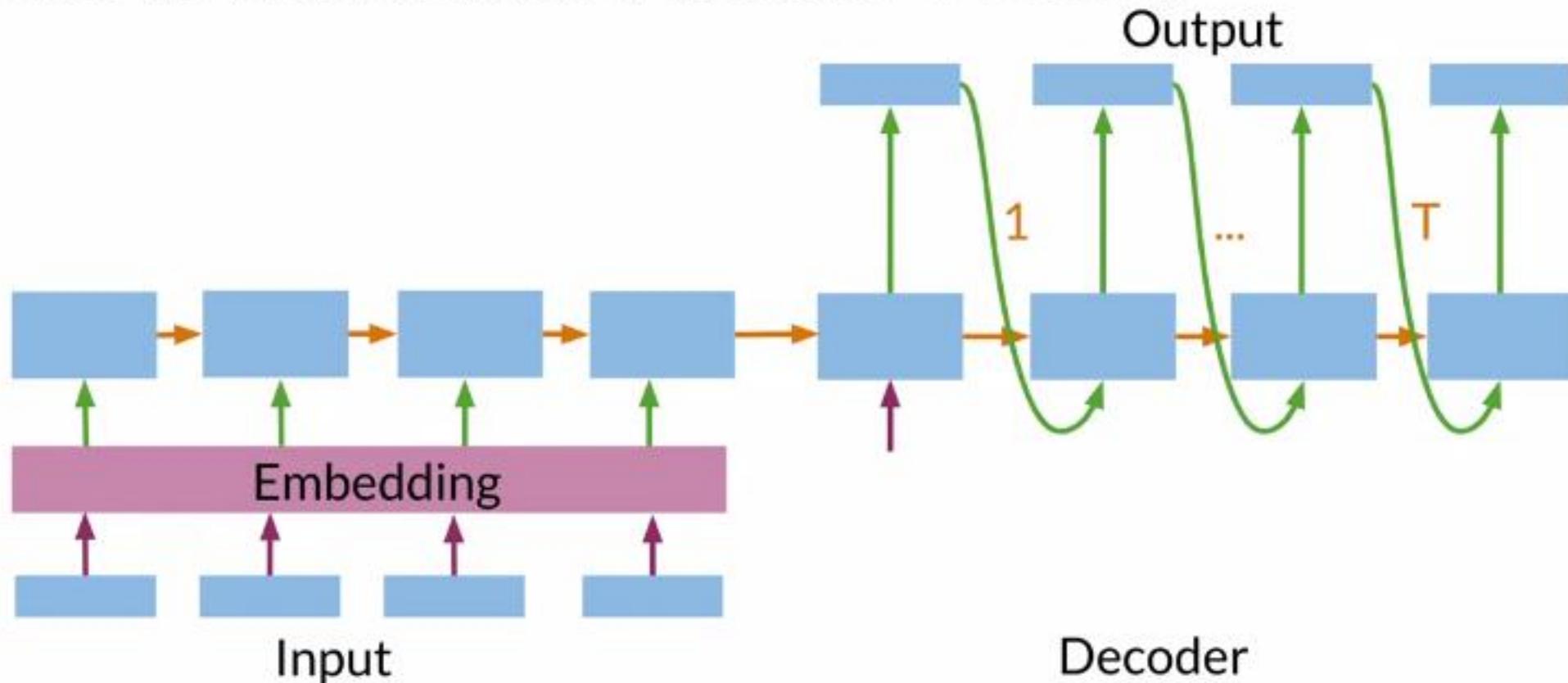


# Seq2Seq Architectures

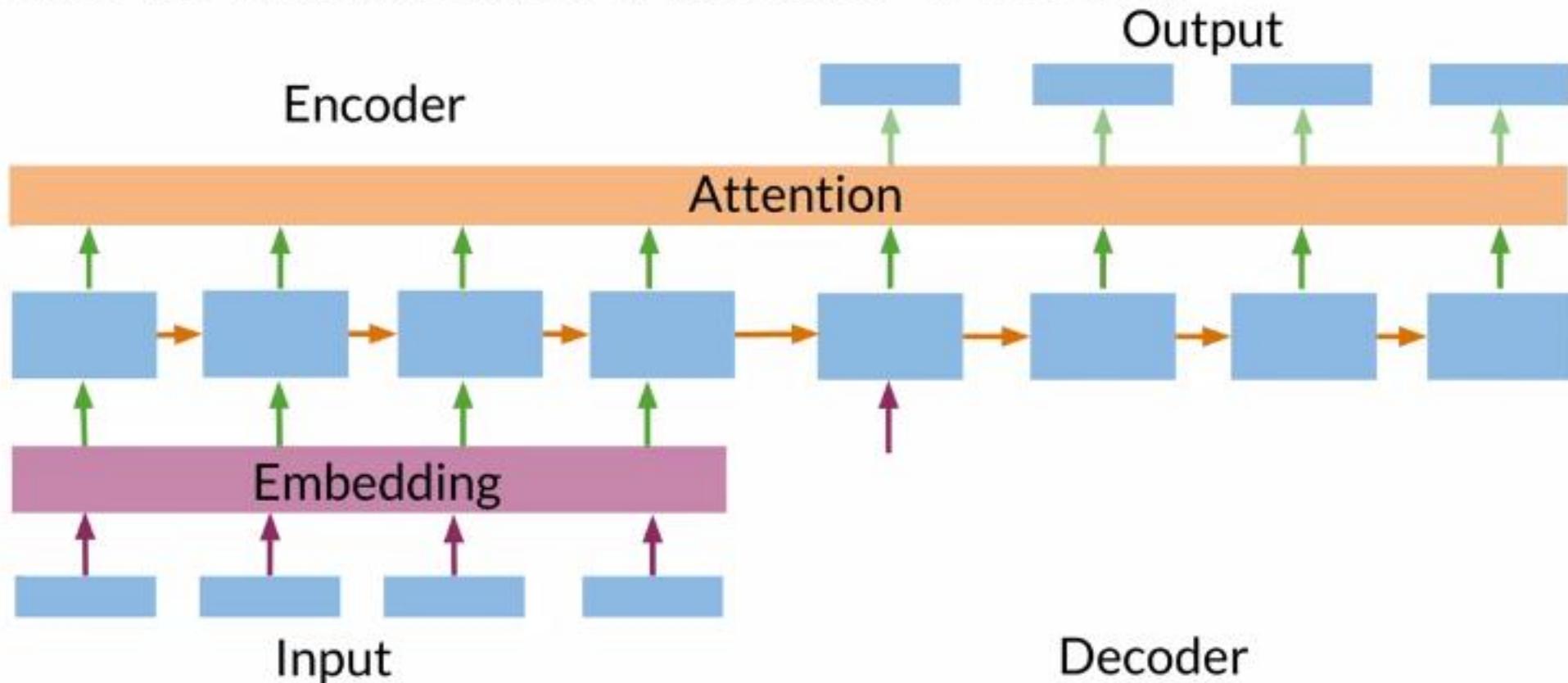


# RNNs vs Transformer: Encoder-Decoder

# RNNs vs Transformer: Encoder-Decoder

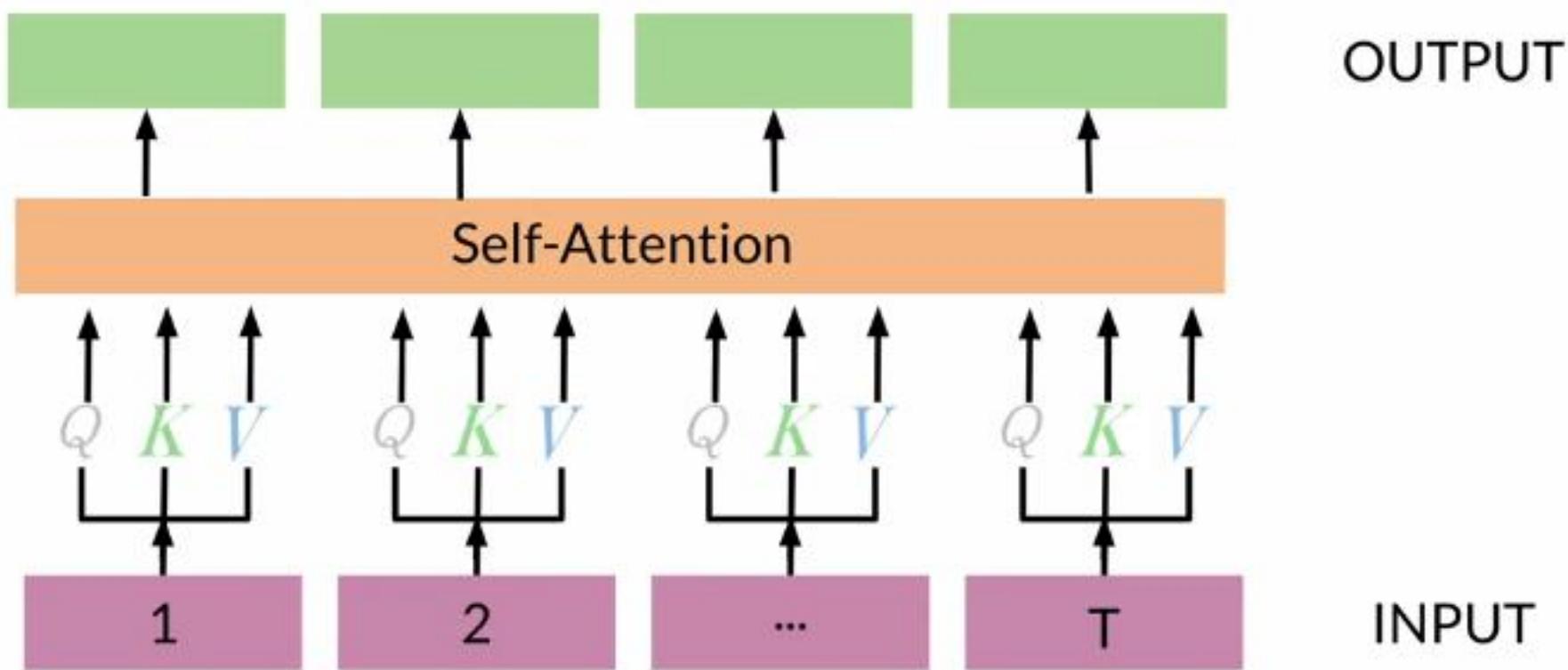


# RNNs vs Transformer: Encoder-Decoder

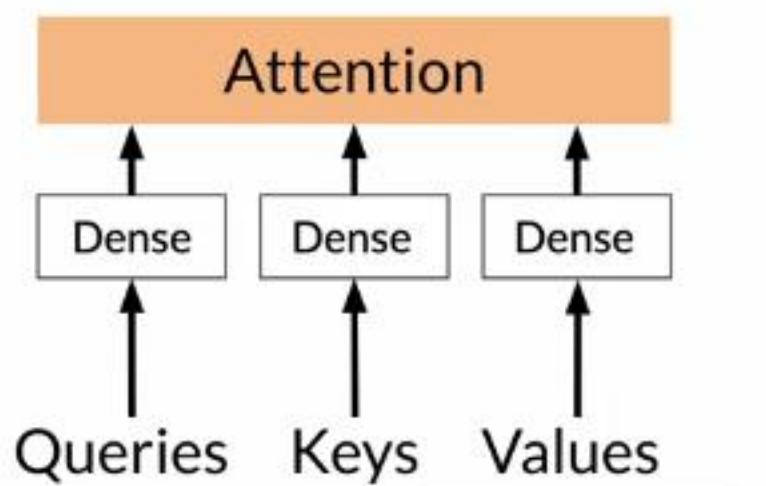


# RNNs vs Transformer: Multi-headed attention

# RNNs vs Transformer: Multi-headed attention



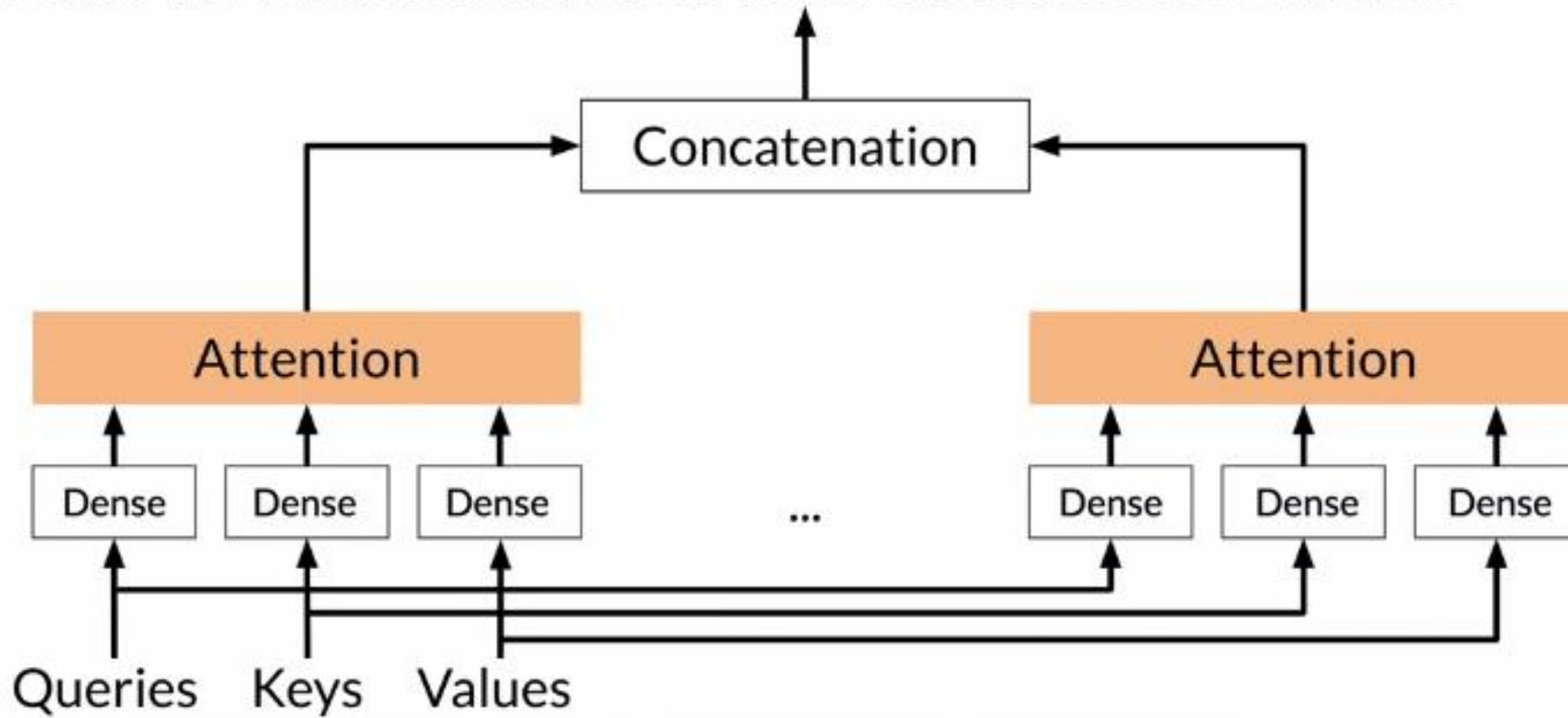
# RNNs vs Transformer: Multi-headed attention



# RNNs vs Transformer: Multi-headed attention



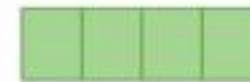
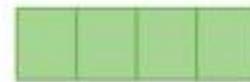
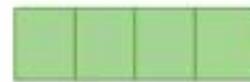
# RNNs vs Transformer: Multi-headed attention



# RNNs vs Transformer: Positional Encoding

# RNNs vs Transformer: Positional Encoding

EMBEDDINGS



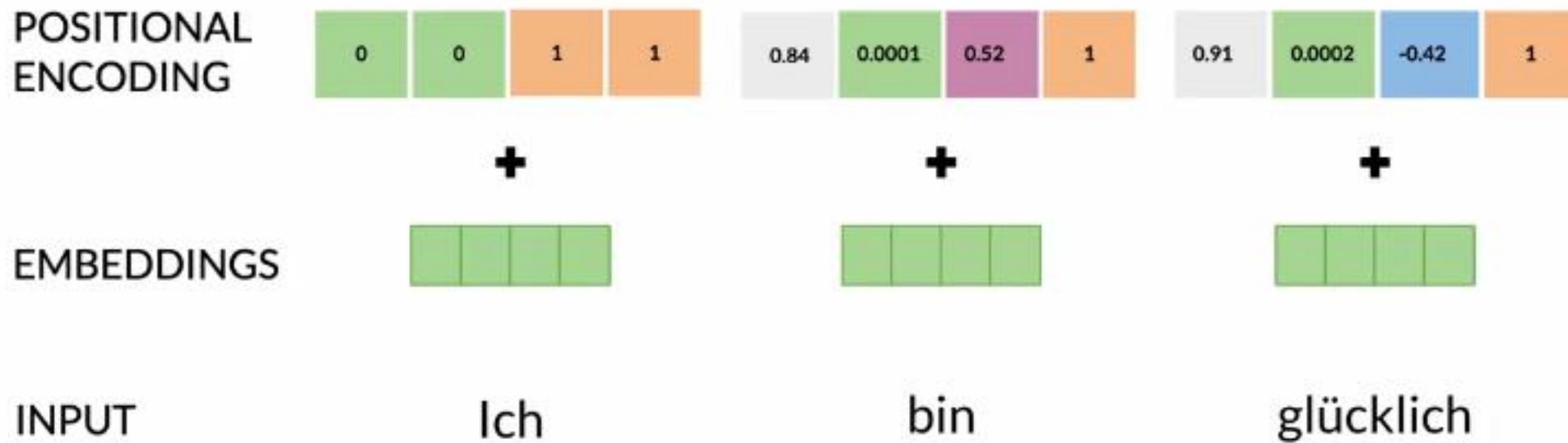
INPUT

Ich

bin

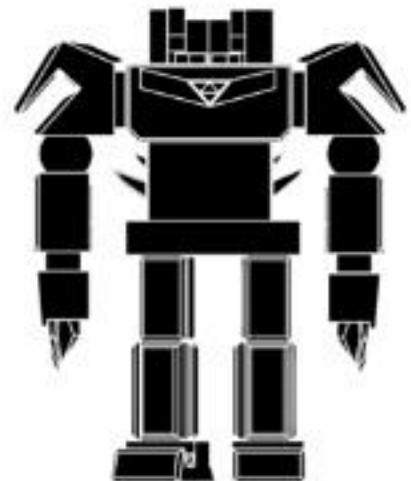
glücklich

# RNNs vs Transformer: Positional Encoding



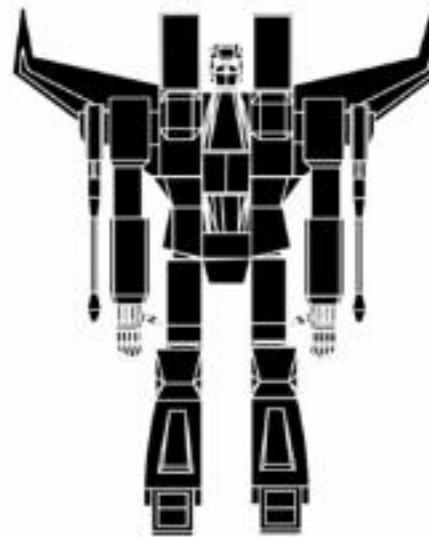
# Summary

- In RNNs parallel computing is difficult to implement
- For long sequences in RNNs there is loss of information
- In RNNs there is the problem of vanishing gradient
- Transformers help with all of the above



# Outline

- Transformers applications in NLP
- Some Transformers
- Introduction to T5



# Transformer NLP applications

# Transformer NLP applications

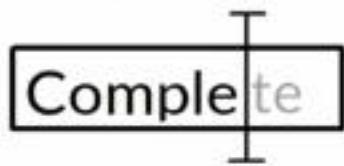


Text  
summarization

# Transformer NLP applications



Text  
summarization

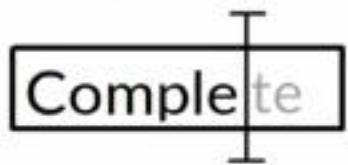


Auto-Complete

# Transformer NLP applications



Text  
summarization



Auto-Complete

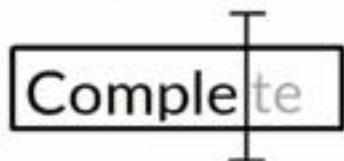
The	wind	blows	hard
Article	Noun	Verb	Adjective

Named entity  
recognition (NER)

# Transformer NLP applications



Text  
summarization



Auto-Complete

The	wind	blows	hard
Article	Noun	Verb	Adjective

Named entity  
recognition (NER)



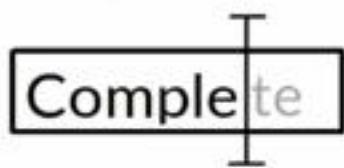
Question  
answering (Q&A)

# Transformer NLP applications



Text  
summarization

Translation



Auto-Complete

The	wind	blows	hard
Article	Noun	Verb	Adjective

Named entity  
recognition (NER)

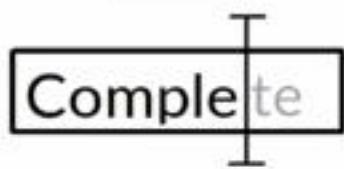


Question  
answering (Q&A)

# Transformer NLP applications



Text  
summarization



Auto-Complete

Translation



Chat-bots



The	wind	blows	hard
Article	Noun	Verb	Adjective

Named entity  
recognition (NER)

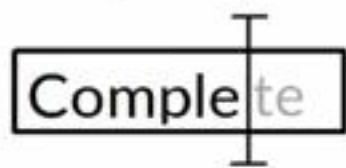


Question  
answering (Q&A)

# Transformer NLP applications



Text  
summarization



Auto-Complete

The	wind	blows	hard
Article	Noun	Verb	Adjective

Named entity  
recognition (NER)



Question  
answering (Q&A)

Translation



Chat-bots



Other NLP tasks

[Sentiment Analysis](#)  
[Market Intelligence](#)  
[Text Classification](#)  
[Character Recognition](#)  
[Spell Checking](#)

# State of the Art Transformers

Radford, A., et al. (2018)  
Open AI

GPT-2: Generative Pre-training for  
Transformer

# State of the Art Transformers

Radford, A., et al. (2018)  
Open AI

GPT-2: Generative Pre-training for  
Transformer

Devlin, J., et al. (2018)  
Google AI Language

BERT: Bidirectional Encoder  
Representations from Transformers

# State of the Art Transformers

Radford, A., et al. (2018)  
Open AI

GPT-2: Generative Pre-training for  
Transformer

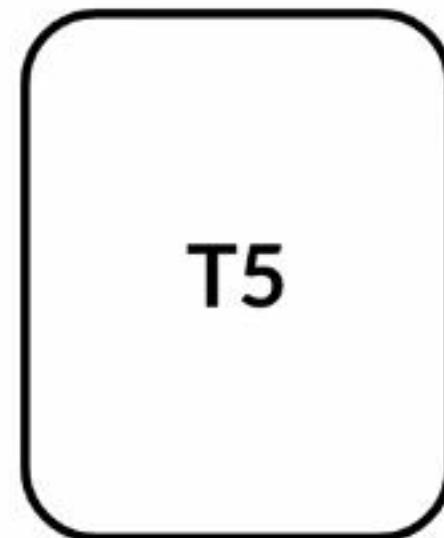
Devlin, J., et al. (2018)  
Google AI Language

BERT: Bidirectional Encoder  
Representations from Transformers

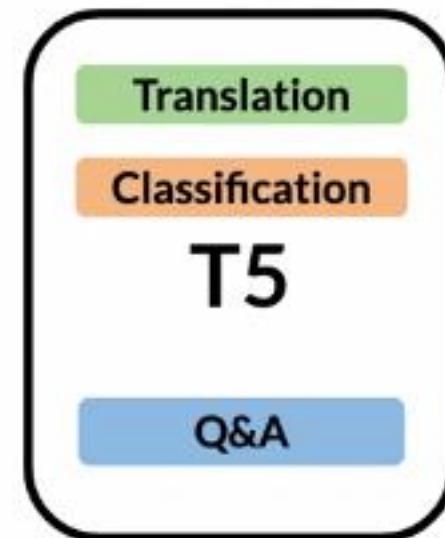
Colin, R., et al. (2019)  
Google

T5: Text-to-text transfer transformer

# T5: Text-To-Text Transfer Transformer

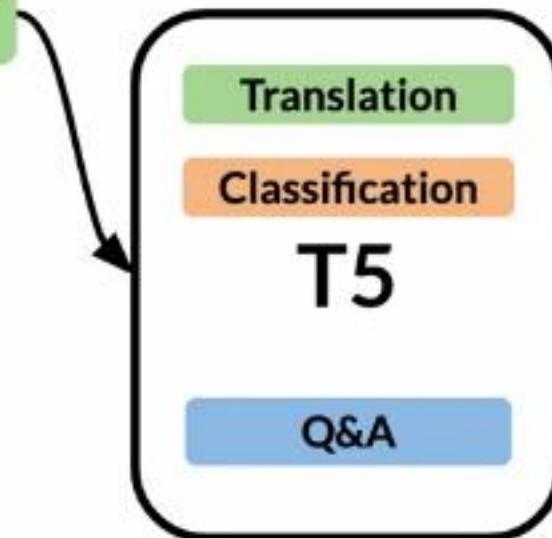


# T5: Text-To-Text Transfer Transformer



# T5: Text-To-Text Transfer Transformer

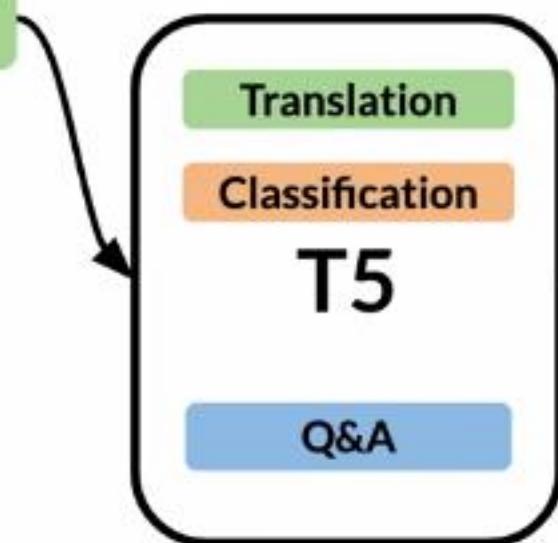
Translate English into German: "I am happy"



# T5: Text-To-Text Transfer Transformer

Translate English into German: "I am happy"

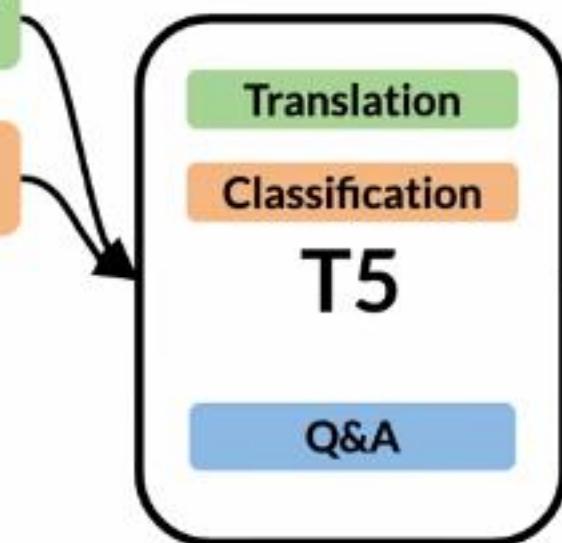
"Ich bin glücklich"



# T5: Text-To-Text Transfer Transformer

Translate English into German: "I am happy"

Cola sentence: "He bought fruits and."

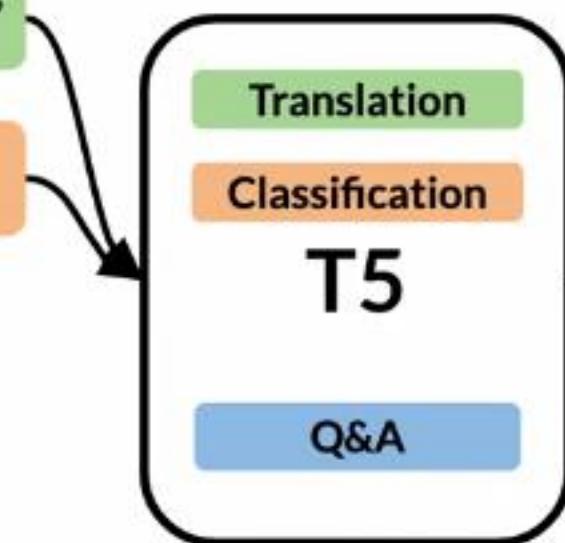


"Ich bin glücklich"

# T5: Text-To-Text Transfer Transformer

Translate English into German: "I am happy"

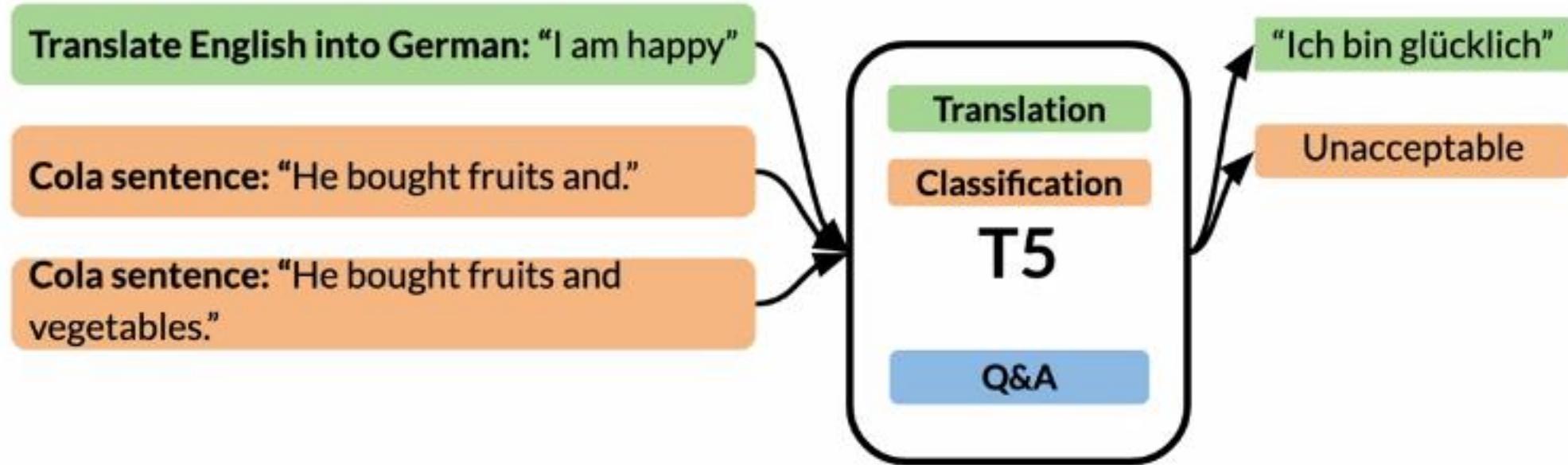
Cola sentence: "He bought fruits and."



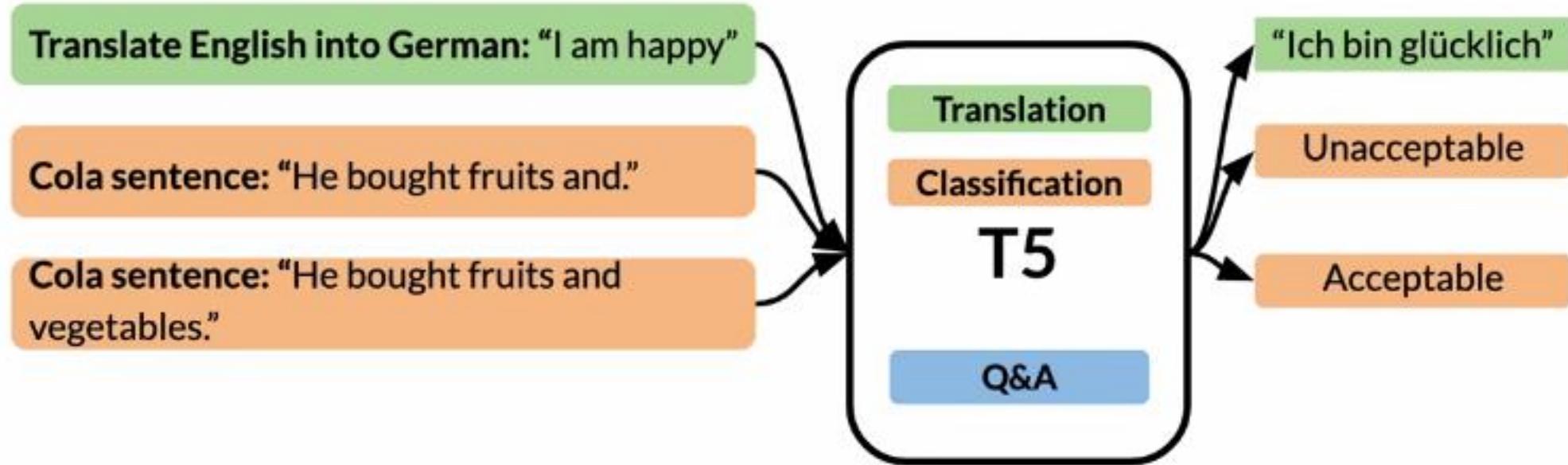
"Ich bin glücklich"

Unacceptable

# T5: Text-To-Text Transfer Transformer



# T5: Text-To-Text Transfer Transformer



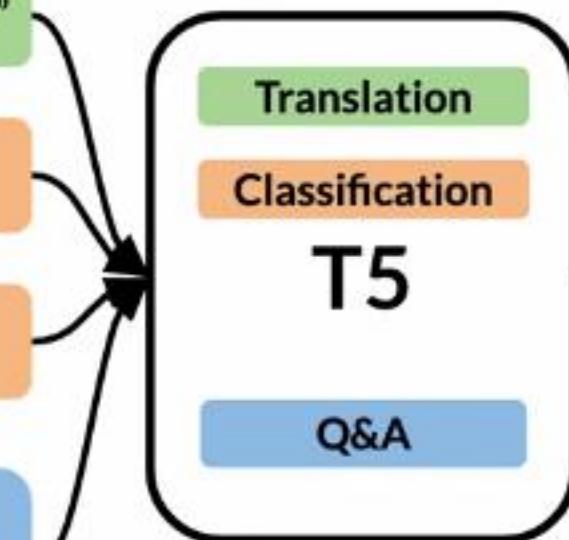
# T5: Text-To-Text Transfer Transformer

Translate English into German: "I am happy"

Cola sentence: "He bought fruits and."

Cola sentence: "He bought fruits and vegetables."

Question: Which volcano in Tanzania is the highest mountain in Africa?

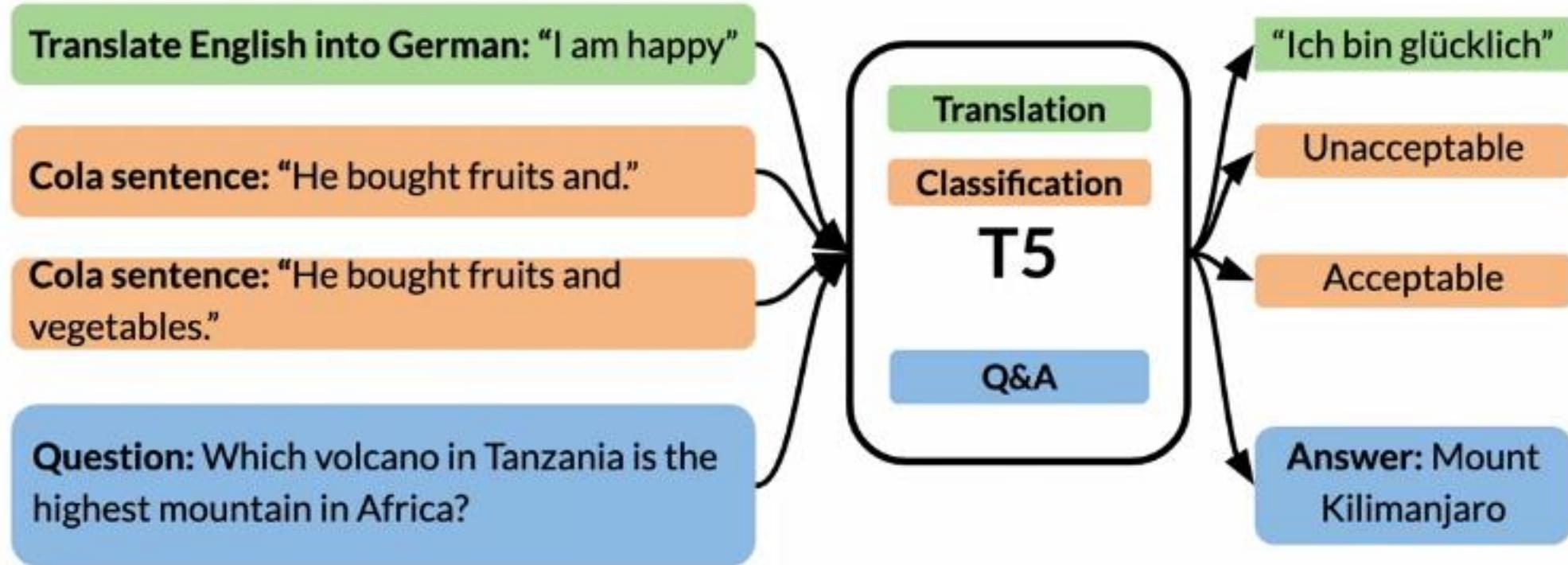


"Ich bin glücklich"

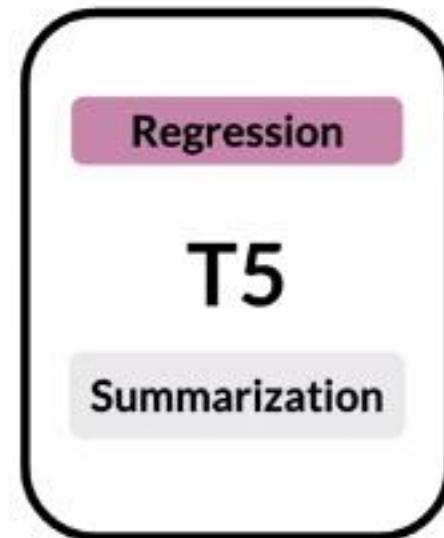
Unacceptable

Acceptable

# T5: Text-To-Text Transfer Transformer

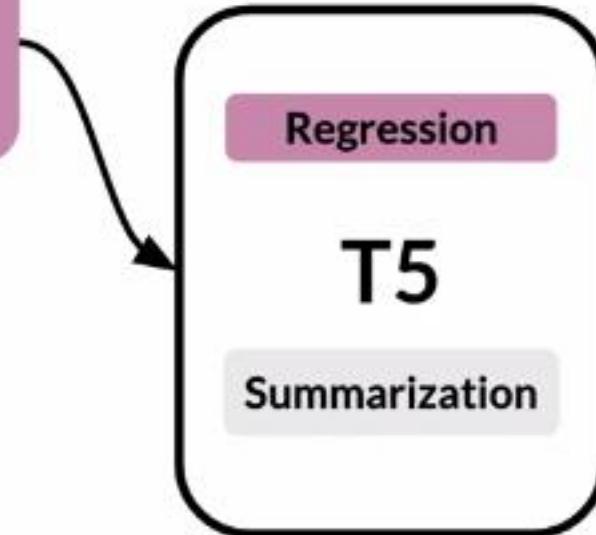


# T5: Text-To-Text Transfer Transformer



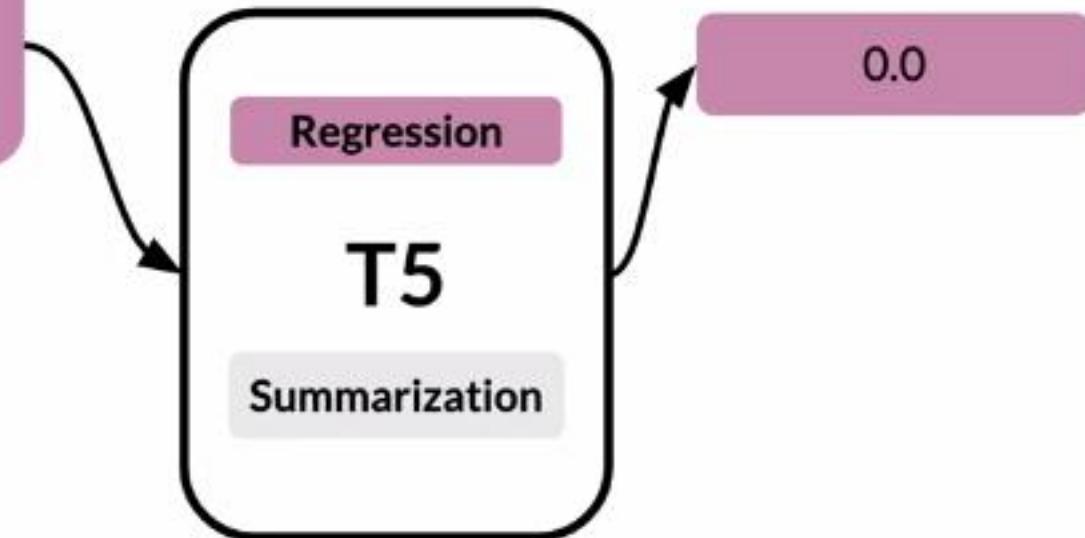
# T5: Text-To-Text Transfer Transformer

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."



# T5: Text-To-Text Transfer Transformer

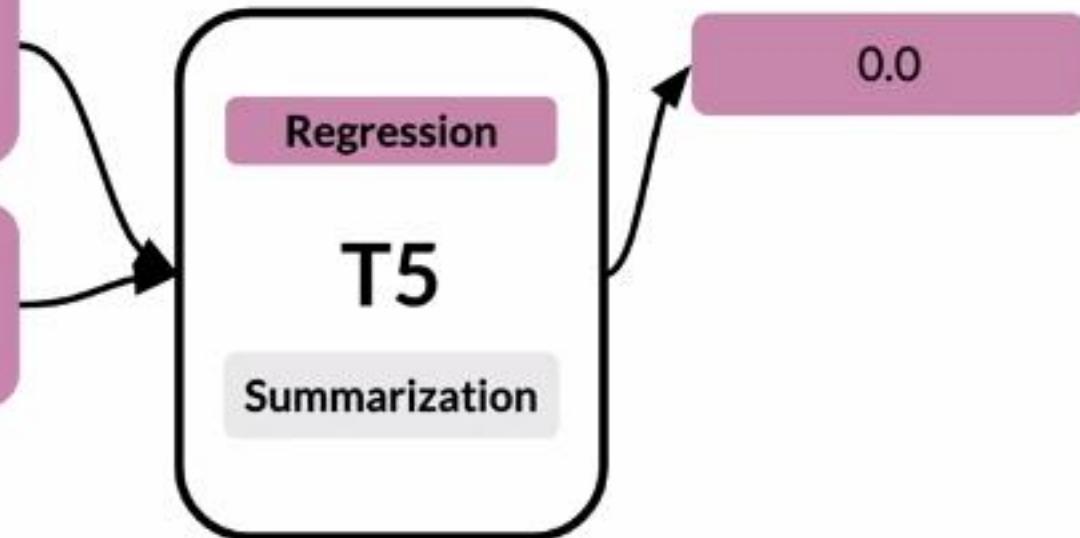
**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."



# T5: Text-To-Text Transfer Transformer

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."

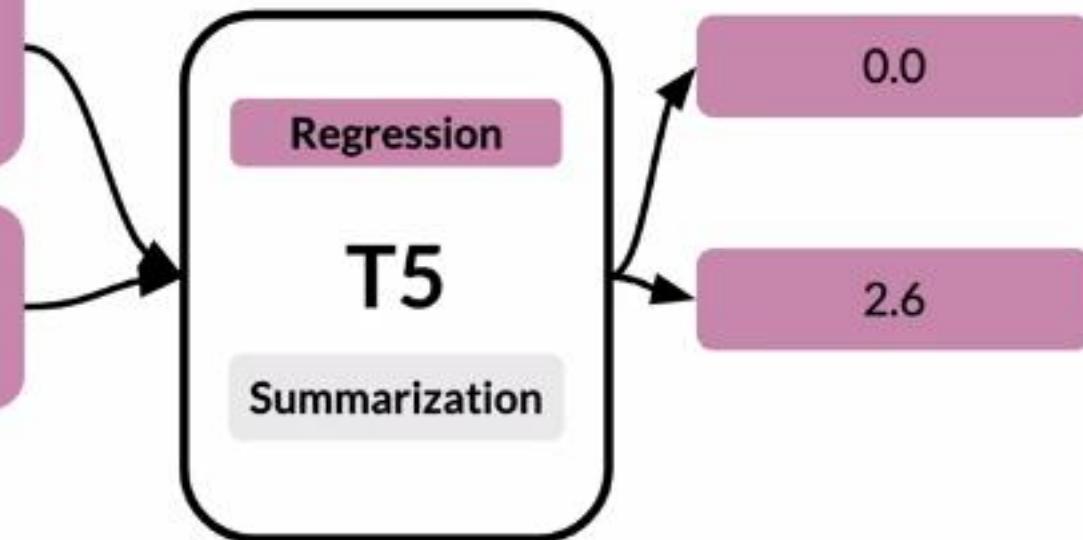
**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "Cats, dogs, and cows are domesticated."



# T5: Text-To-Text Transfer Transformer

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "Cats, dogs, and cows are domesticated."

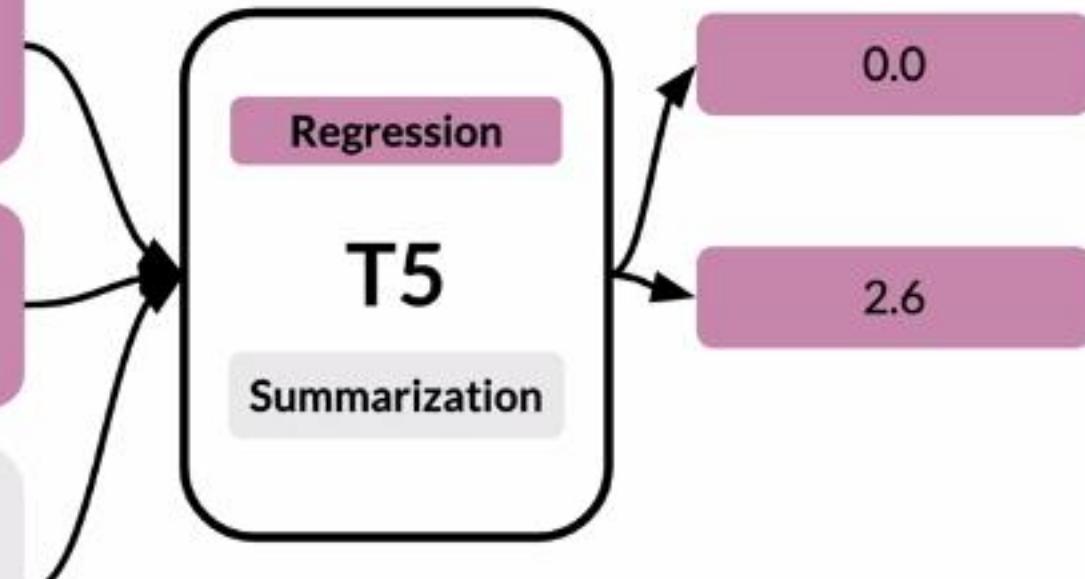


# T5: Text-To-Text Transfer Transformer

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "Cats, dogs, and cows are domesticated."

**Summarize:** "State authorities dispatched emergency crews Tuesday to survey the damage after an onslaught of severe weather in mississippi..."

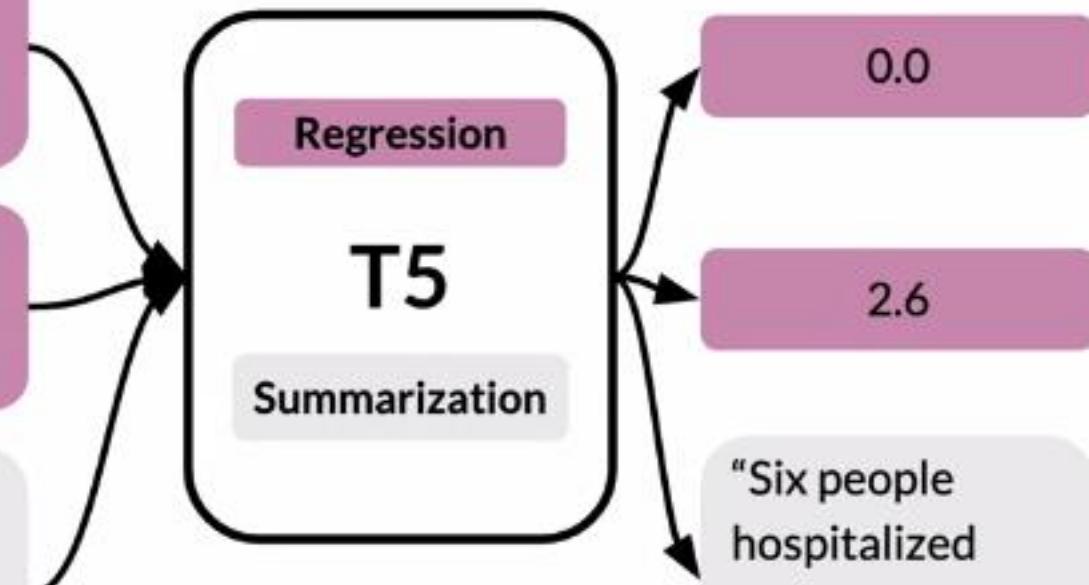


# T5: Text-To-Text Transfer Transformer

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."

**Stsb sentence1:** "Cats and dogs are mammals." **Sentence2:** "Cats, dogs, and cows are domesticated."

**Summarize:** "State authorities dispatched emergency crews Tuesday to survey the damage after an onslaught of severe weather in mississippi..."



# T5 Demo

The image shows a user interface for a T5 demo. At the top center, a blue button labeled "question 2" is visible. Below it, a question is displayed: "What colour hair did Charles Dickens' character David Copperfield have?". Underneath the question, there are two input fields. The first input field, labeled "You", contains the text "P| I". To the right of this field is a "SUBMIT" button and a circular icon containing the number "1". The second input field, labeled "T5", contains the text "...". To the right of this field is a "LOCKED IN!" button and another circular icon containing the number "1". A large black rectangular redaction box covers the bottom portion of the screen.

# T5 Demo



# T5 Demo

The screenshot shows a user interface for a question-and-answer application. At the top, a large black bar covers most of the screen. Below it, a blue button labeled "question 1" is centered. The main content area has a white background. A question is displayed: "What country is the largest oil producer in Africa?". Two entries are shown under the heading "You": "Nigeria" and "Nigerian". Both entries are followed by a green "CORRECT!" button and a purple circle containing the number "1". Below this, under the heading "T5", is the entry "Nigeria", which is also followed by a green "CORRECT!" button and a purple circle containing the number "1". A blue "NEXT QUESTION" button is centered below the entries. At the bottom of the main content area, a light gray bar displays the text "Correct answer: nigeria" and a link "▶ See all accepted answers". The bottom of the screen features a black navigation bar with icons for play, volume, and settings, along with a progress bar showing "0:09 / 0:26".

# T5 Demo

The screenshot shows a trivia quiz interface. At the top, a blue bar contains the text "question 2". Below it, a question is displayed: "What colour hair did Charles Dickens' character David Copperfield have?". Two answer options are shown in red boxes: "HOPE :(" and "RED". Each option has a blue circular button with the number "1" next to it. A cursor is hovering over the "RED" button. Below the options is a large blue "NEXT QUESTION" button. Underneath the button, the text "Correct answer: RED" is displayed, followed by a link "▶ See all accepted answers". At the bottom of the screen, there is a black navigation bar with icons for back, forward, search, and other controls, along with a timestamp "0:14 / 0:26".

question 2

What colour hair did Charles Dickens' character David Copperfield have?

You      T5

HOPE :(  
RED

1  
1

NEXT QUESTION

Correct answer: RED

▶ See all accepted answers

0:14 / 0:26

# T5 Demo

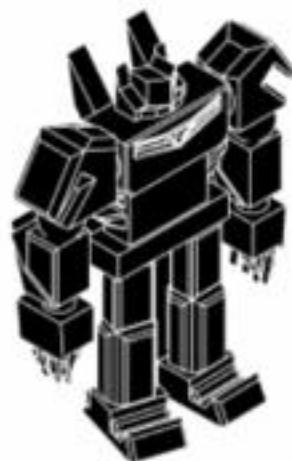
The screenshot shows a user interface for a T5 demo. At the top, there is a large black rectangular redaction box. Below it, a blue button labeled "question 3" is centered. The main content area contains a question: "Who wrote the 1961 novel The Prime of Miss Jean Brodie ?". Below the question, there are two rows for responses. The first row is for "You", showing a text input field containing "Iain M. Banks" and a "SUBMIT" button. To the right of the input field are two circular icons, each containing the number "1". The second row is for "T5", showing a text input field containing "Iain M. Banks" and a "SUBMIT" button. To the right of the input field are two circular icons, each containing the number "1". At the bottom of the interface is a black navigation bar with various icons and a progress bar indicating "0:15 / 0:26".

# T5 Demo

The image shows a user interface for a T5 demo. At the top, there is a large black rectangular area. Below it, a blue button labeled "question 3" is centered. The main content area contains a question: "Who wrote the 1961 novel The Prime of Miss Jean Brodie ?". Below the question, there is a form. On the left, it says "You:" followed by a text input field containing "Muriel Spark". To the right of the input field is a blue "SUBMIT" button. To the right of the submit button are two circular icons, each containing the number "1". Below the "You:" section, there is another row with "T5" on the left, a text input field with "Muriel Spark" in it, and a blue "LOCKED IN!" button to its right. This row also has two circular icons with the number "1" to its right. At the bottom of the interface is a black bar with a red progress bar showing "0.29 / 0.26". On the far right of this bar are several small white icons.

# Summary

- Transformers are suitable for a wide range of NLP applications
- GPT-2, BERT and T5 are the cutting-edge Transformers
- T5 is a powerful multi-task transformer



# Outline

- Introducing attention (Translation example)
- Mathematics behind Attention



# Introducing attention - Translation example



# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).

I am happy



# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).

I am happy



Ich bin glücklich

# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).

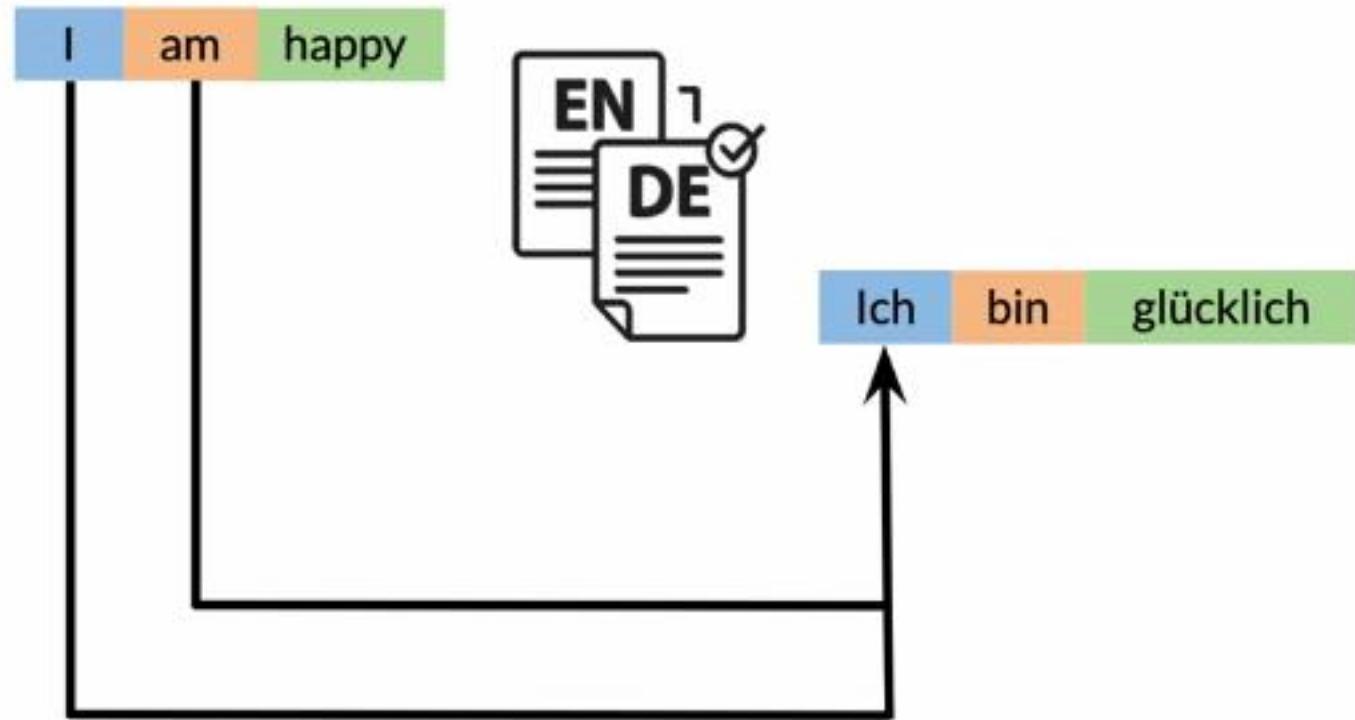
I am happy



Ich bin glücklich

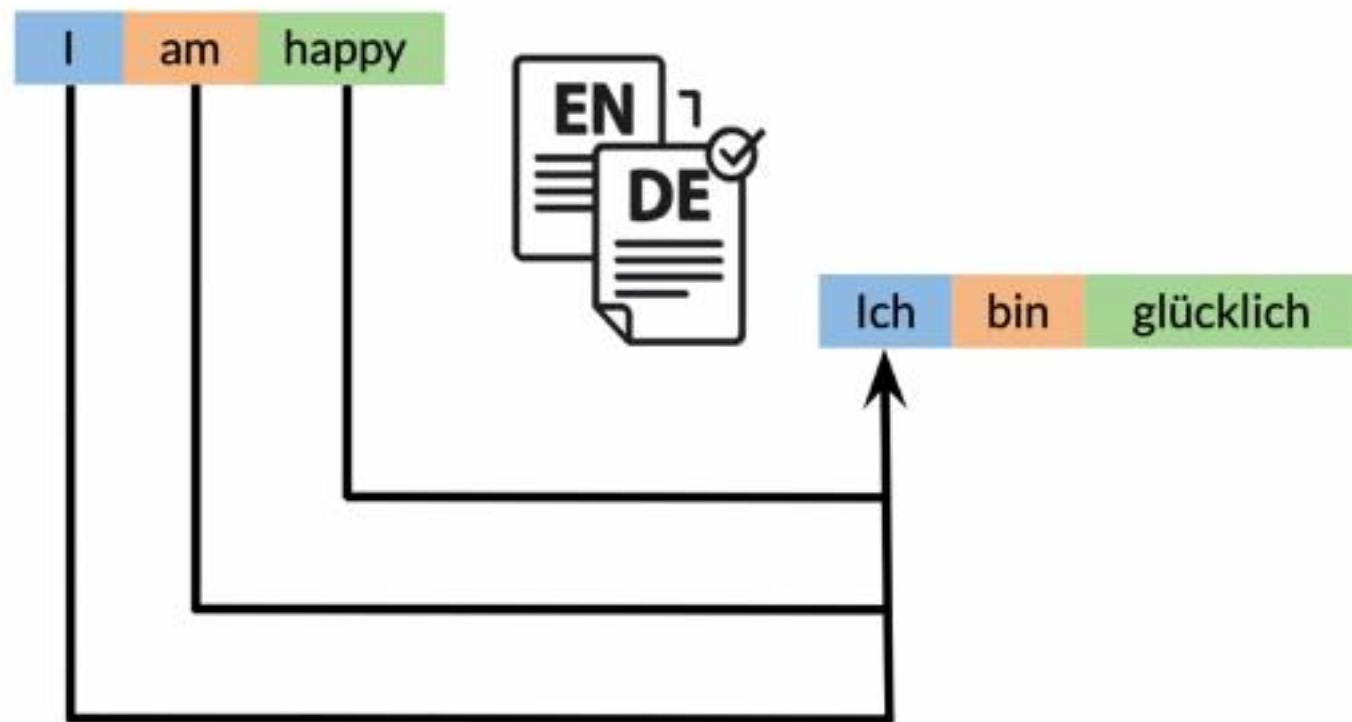
# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.



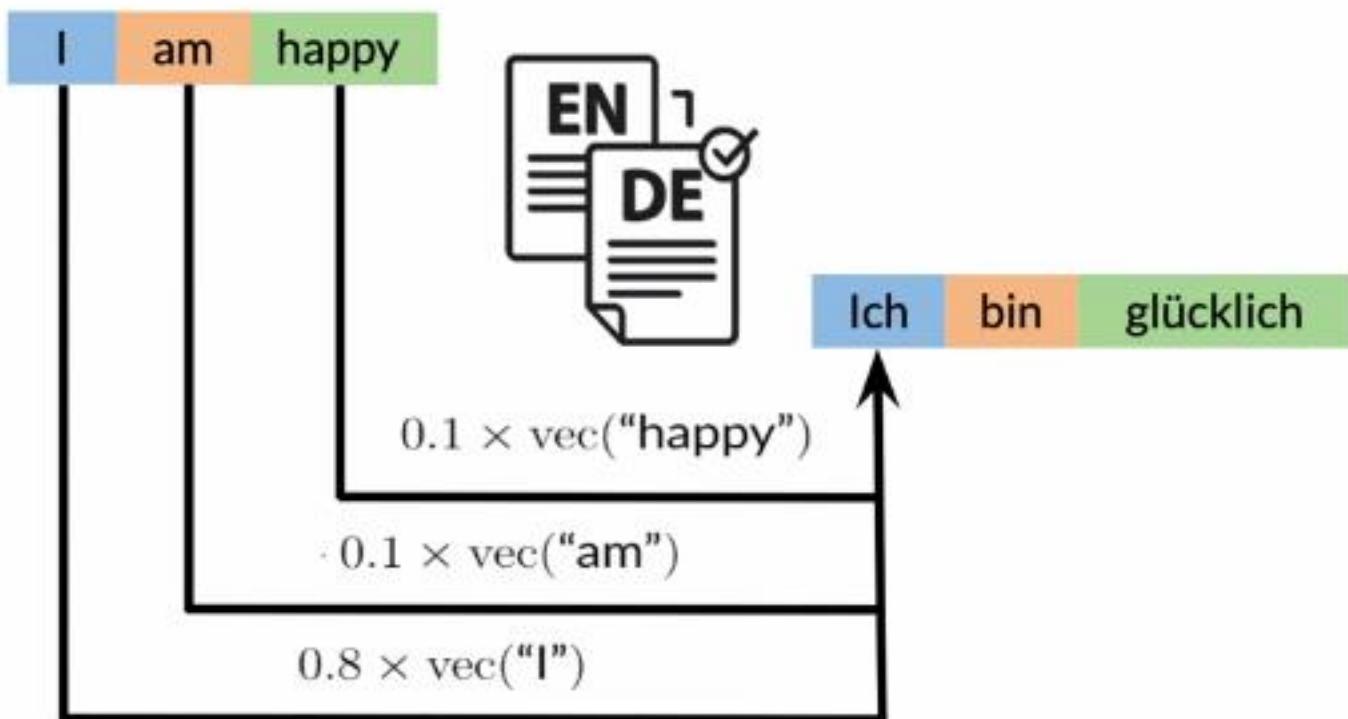
# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.



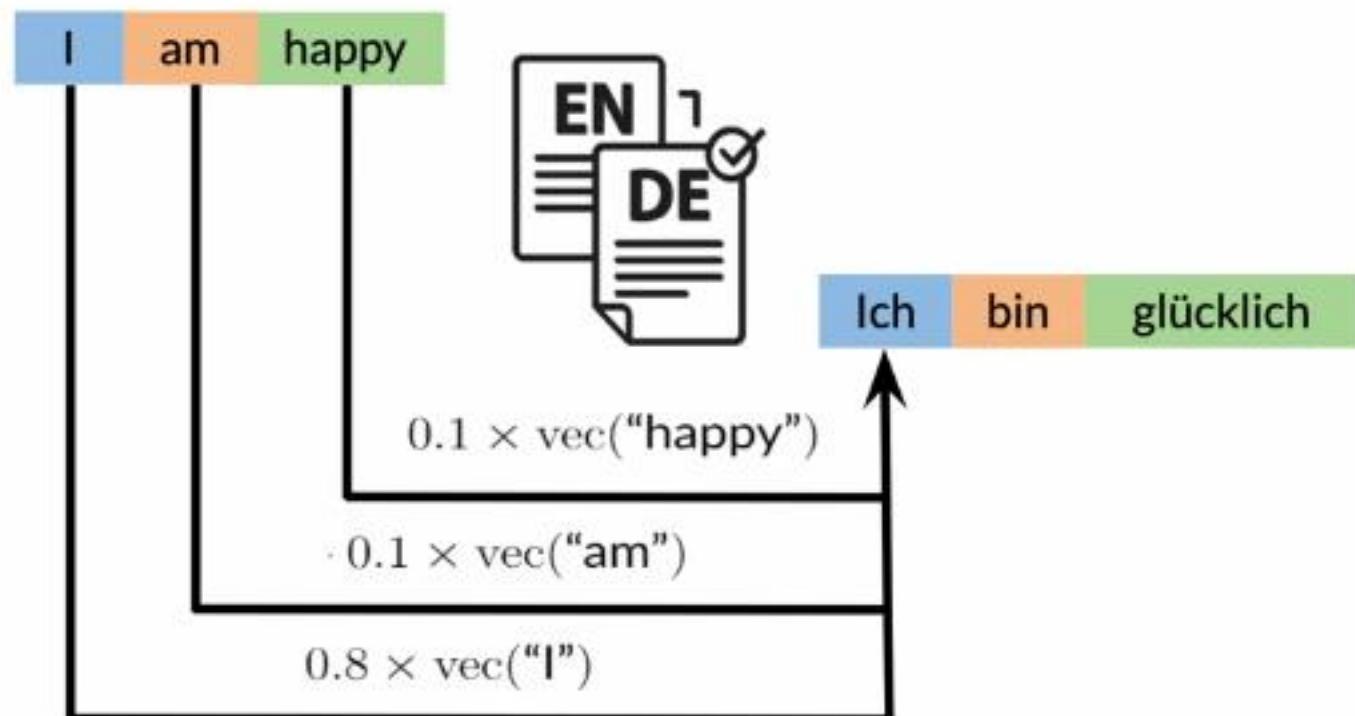
# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.



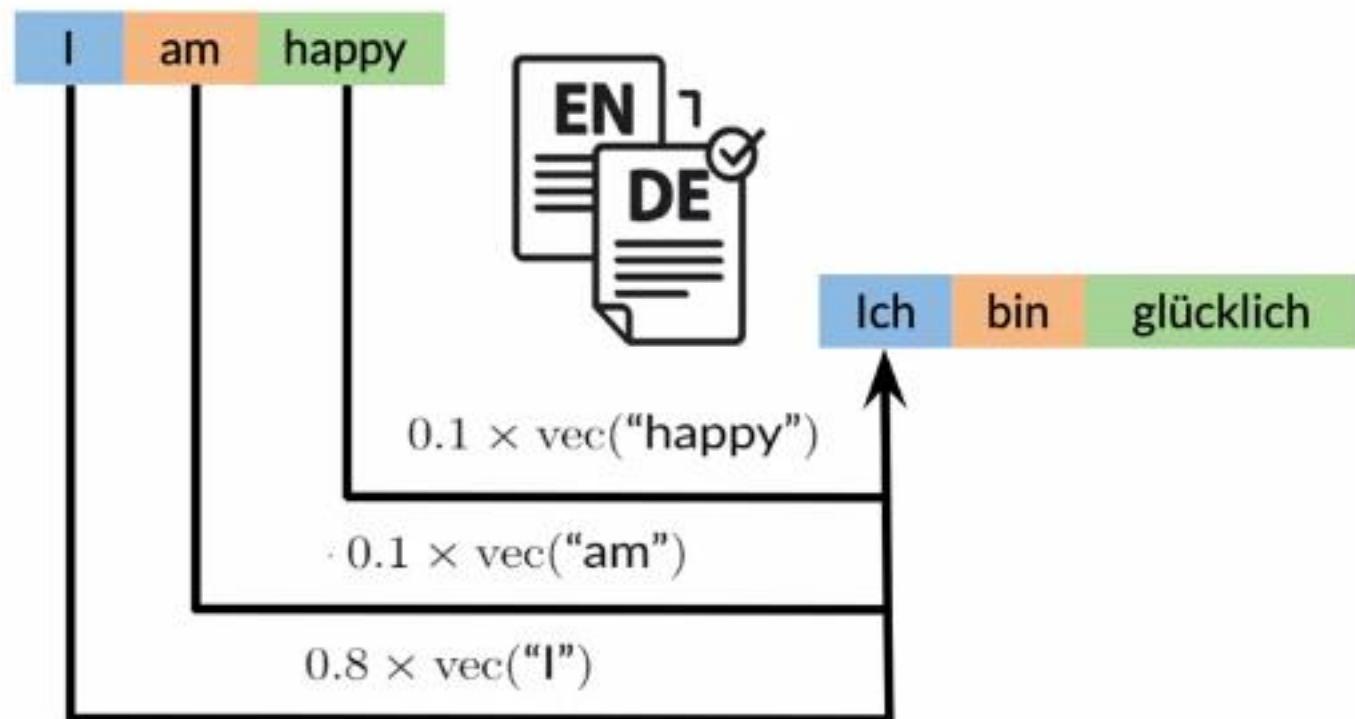
# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



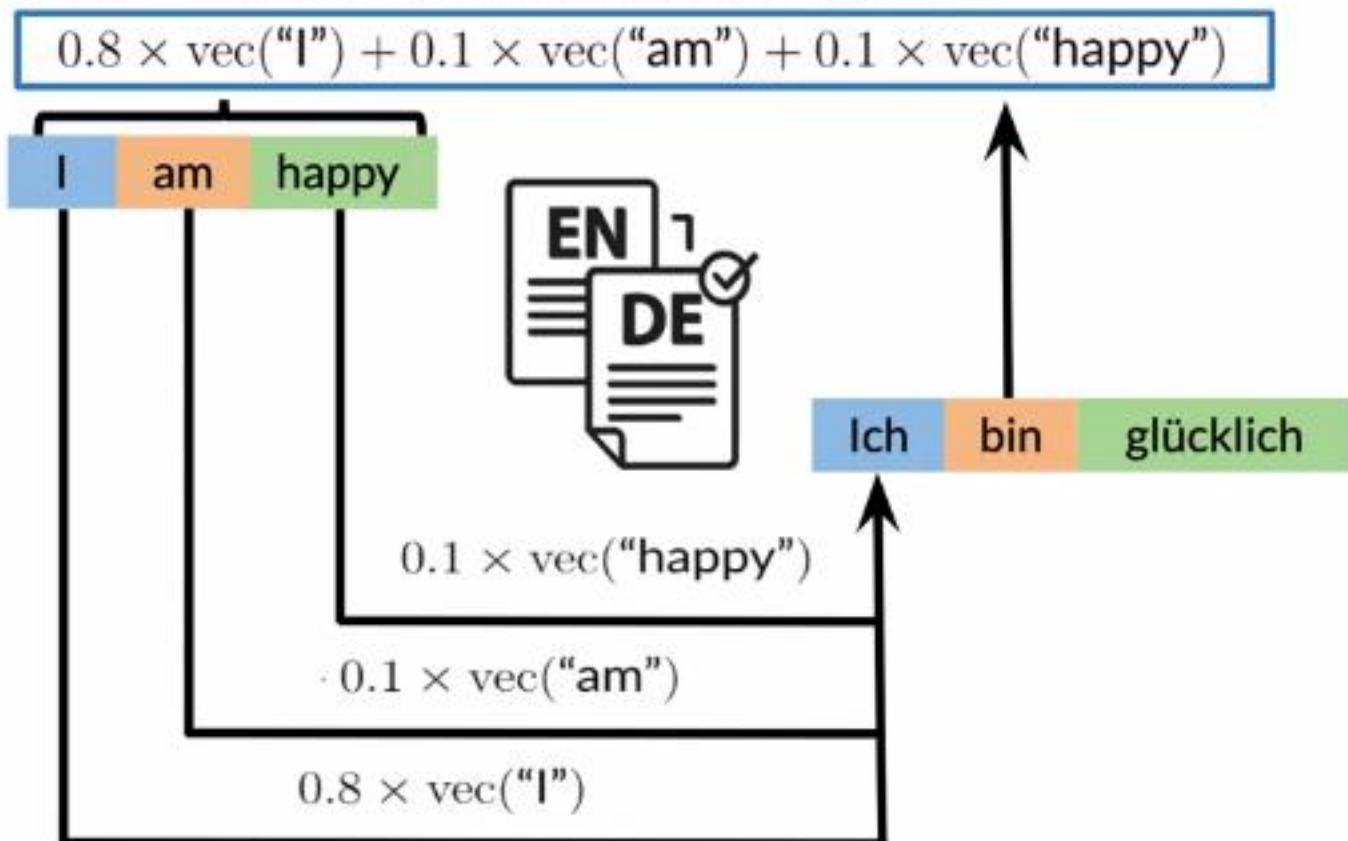
# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



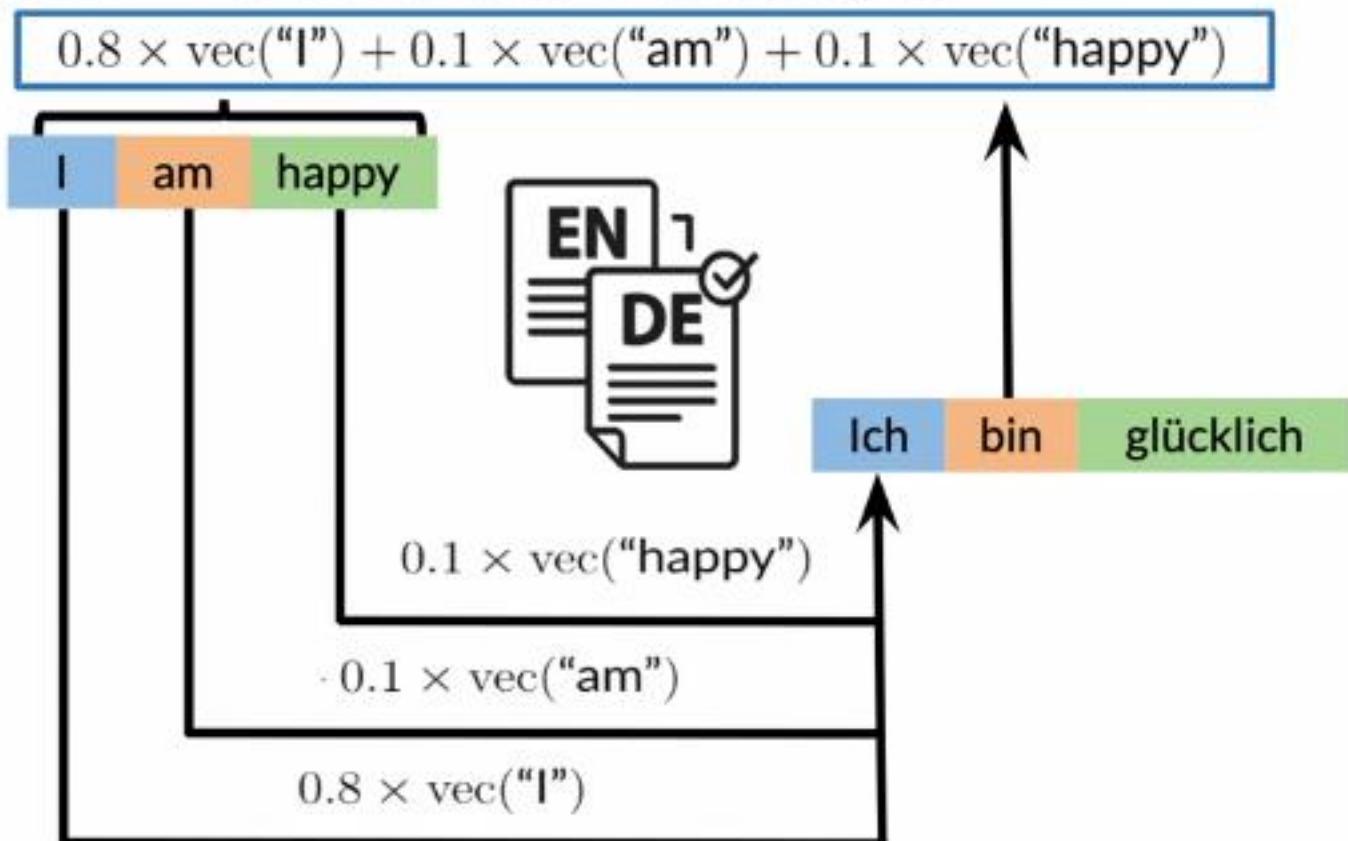
# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



# Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



# Queries, Keys and Values

# Queries, Keys and Values

$Q$

$K$

$V$

# Queries, Keys and Values

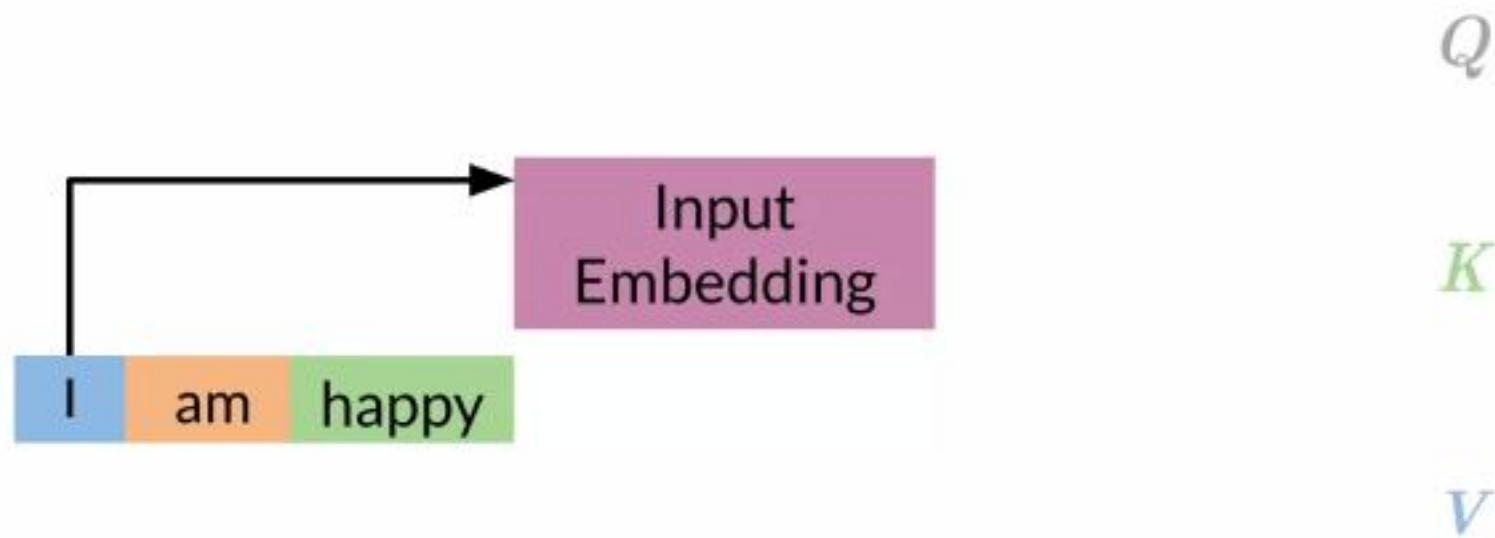
$Q$

$K$

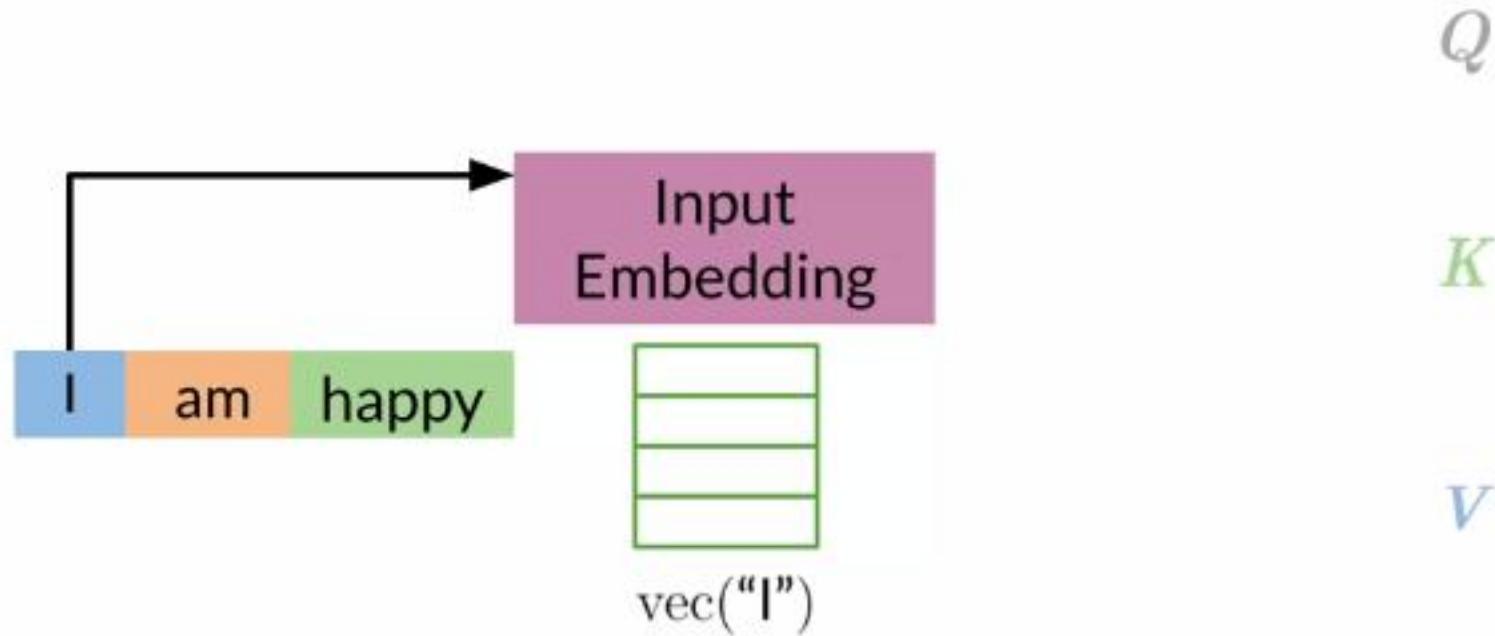
$V$

I am happy

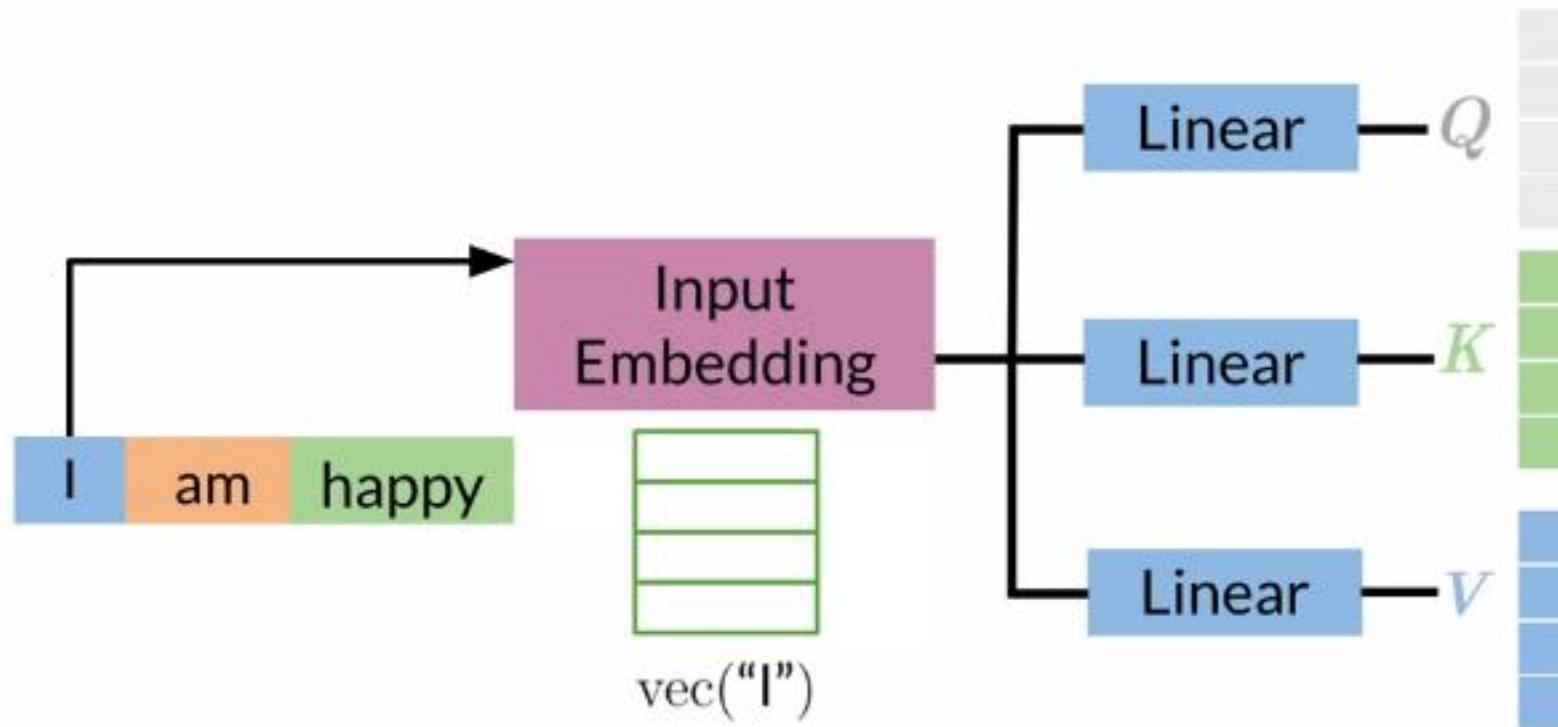
# Queries, Keys and Values



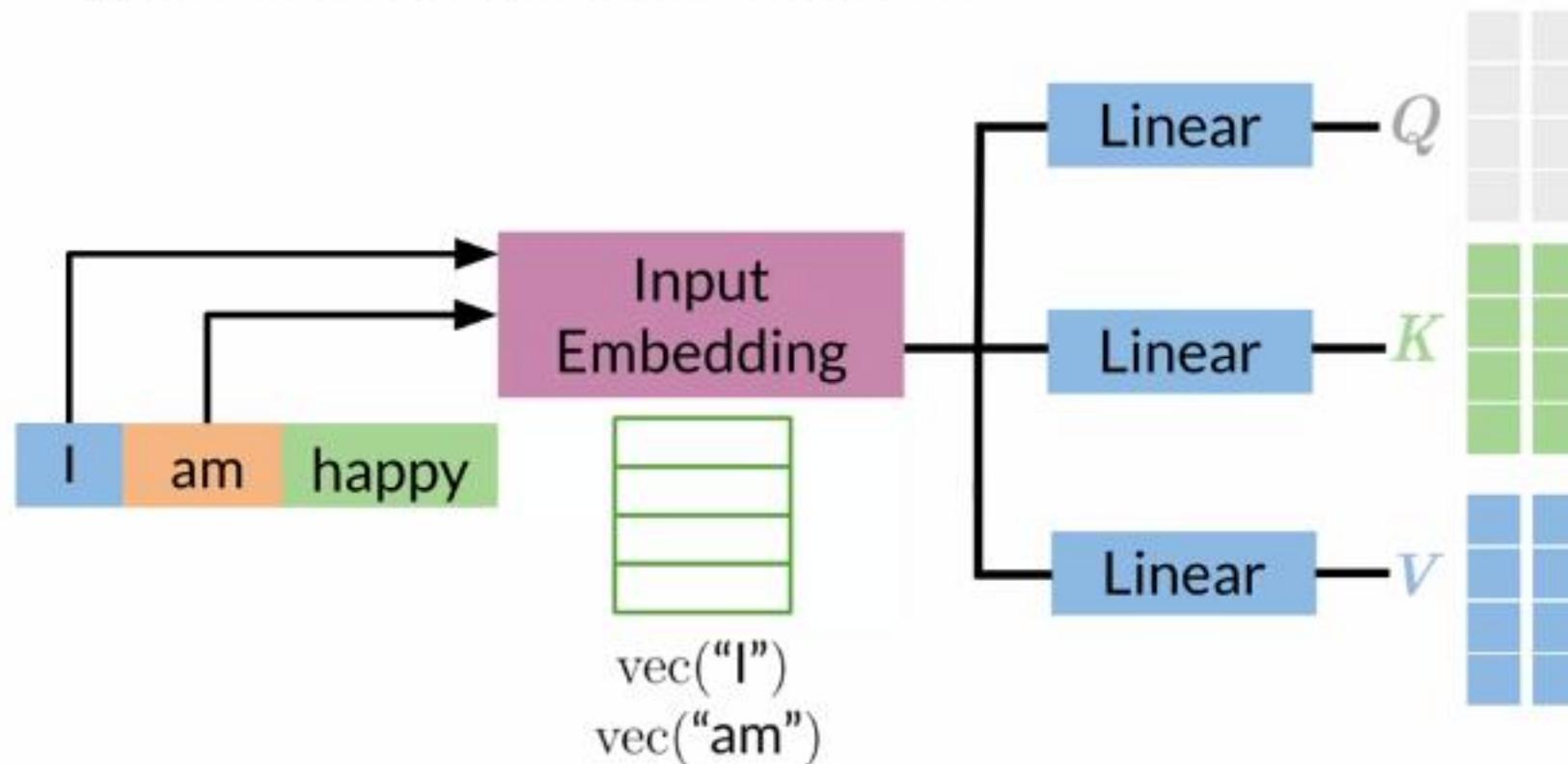
# Queries, Keys and Values



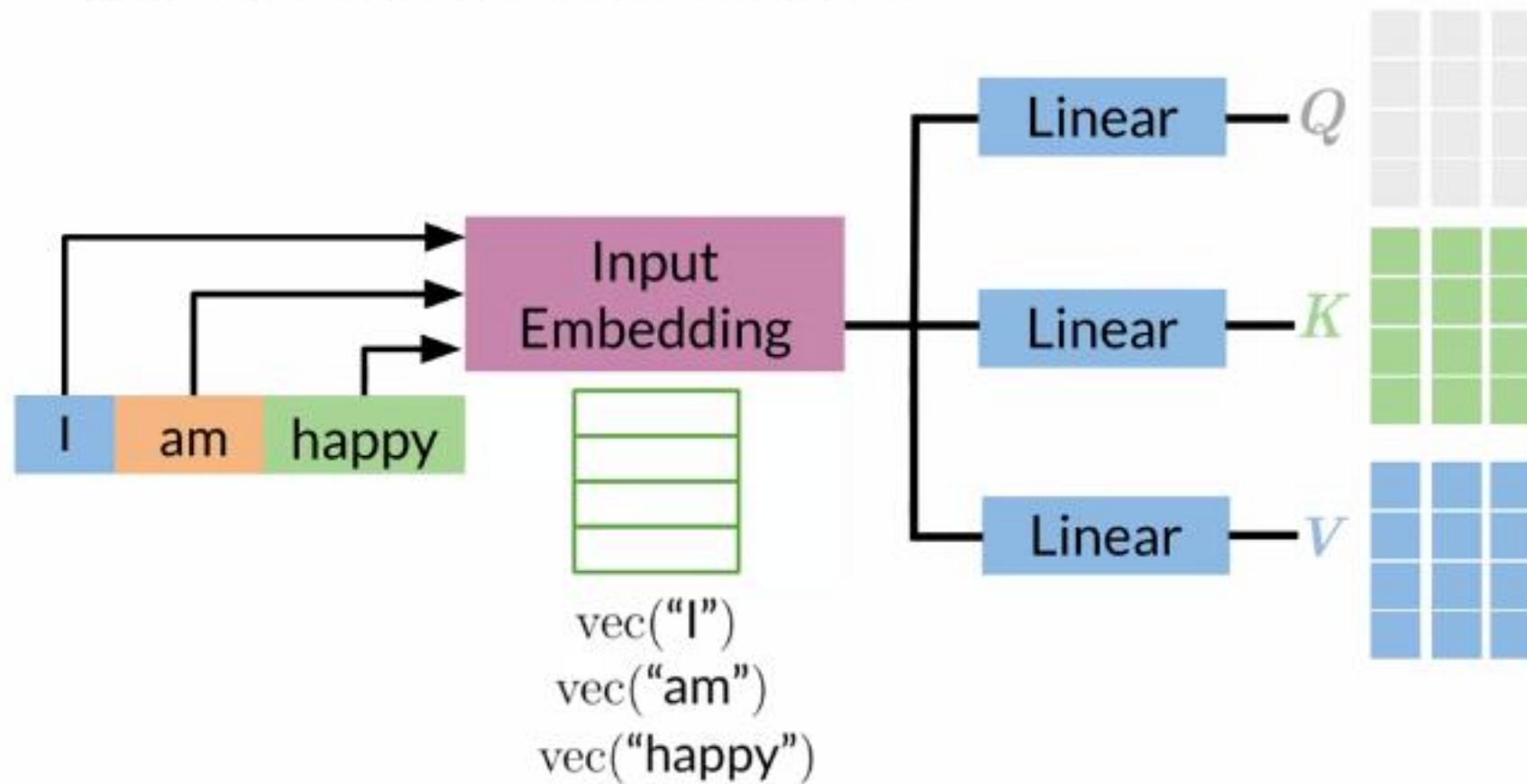
# Queries, Keys and Values



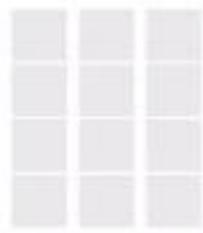
# Queries, Keys and Values



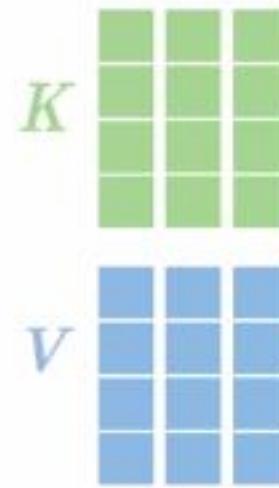
# Queries, Keys and Values



# Concept of attention



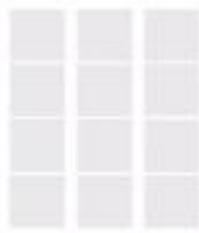
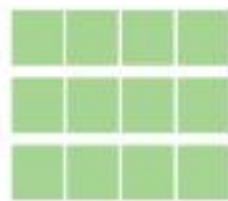
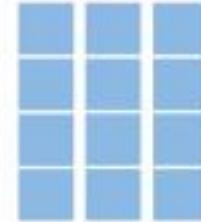
$Q$



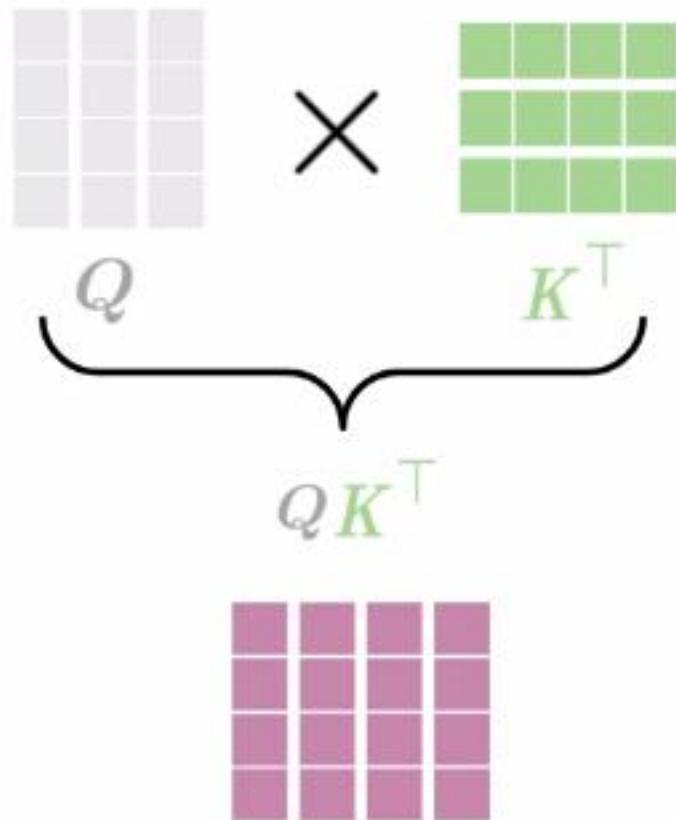
$K$

$V$

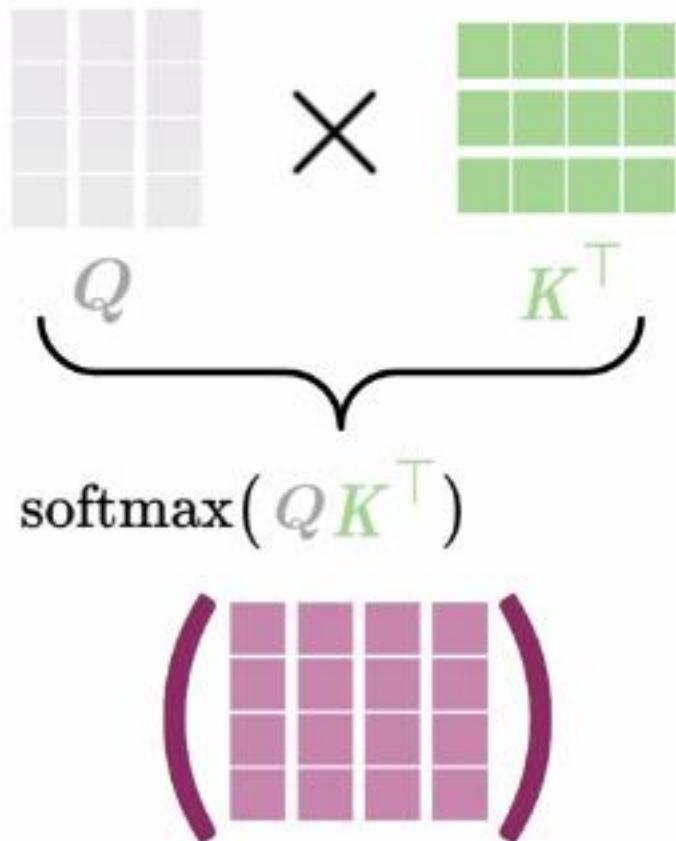
# Concept of attention

 $Q$  $K^\top$  $V$

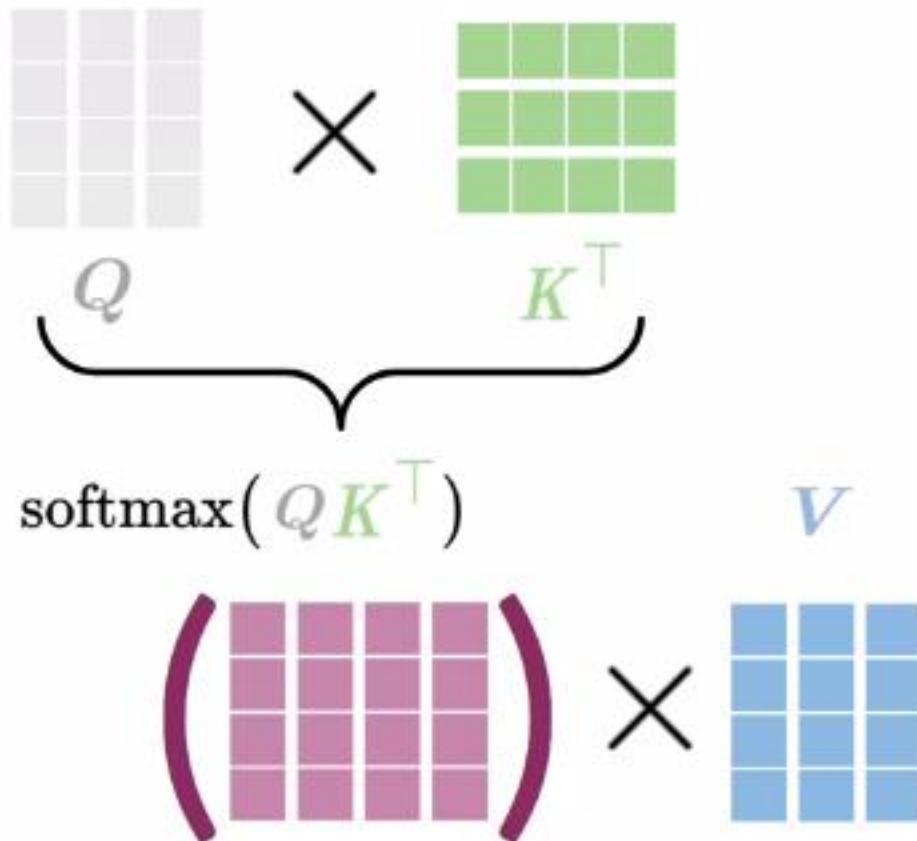
# Concept of attention



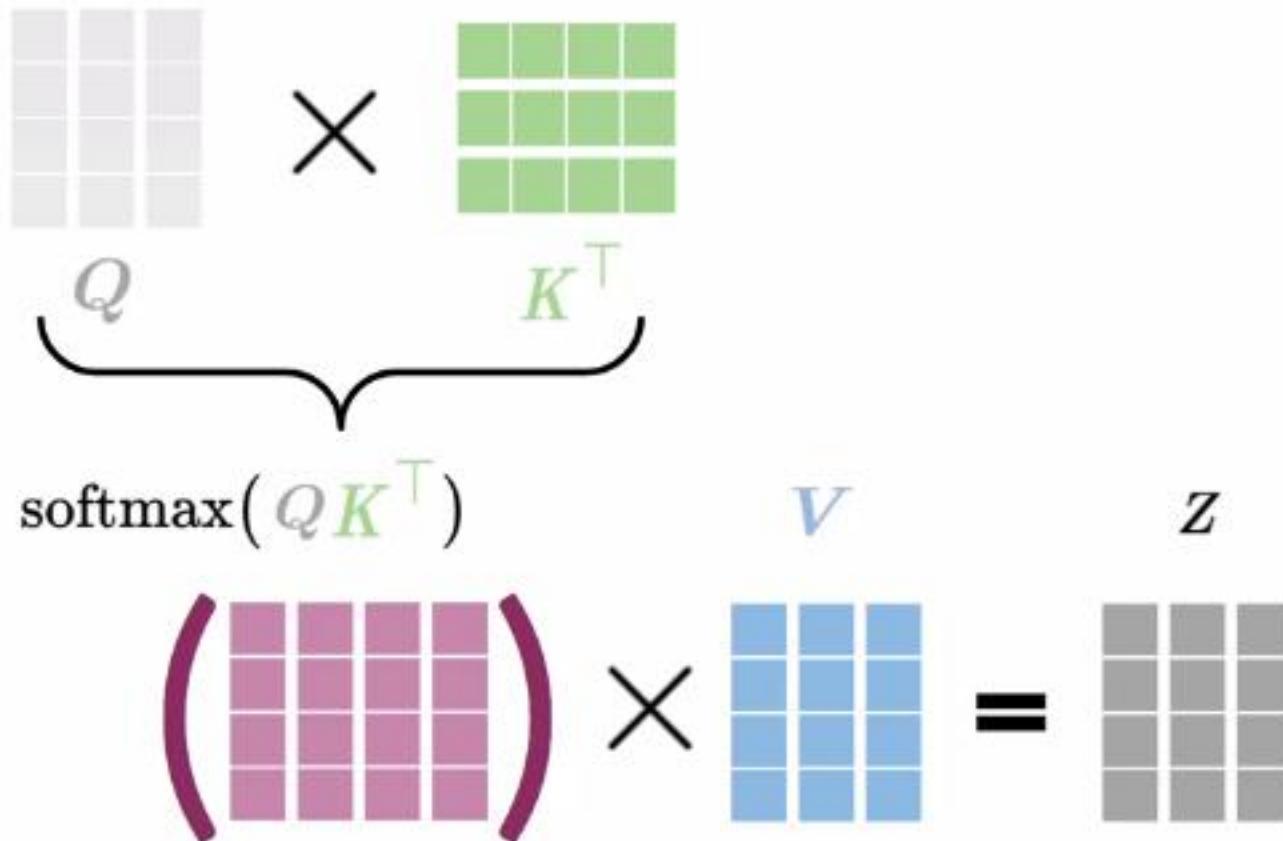
# Concept of attention



# Concept of attention



# Concept of attention



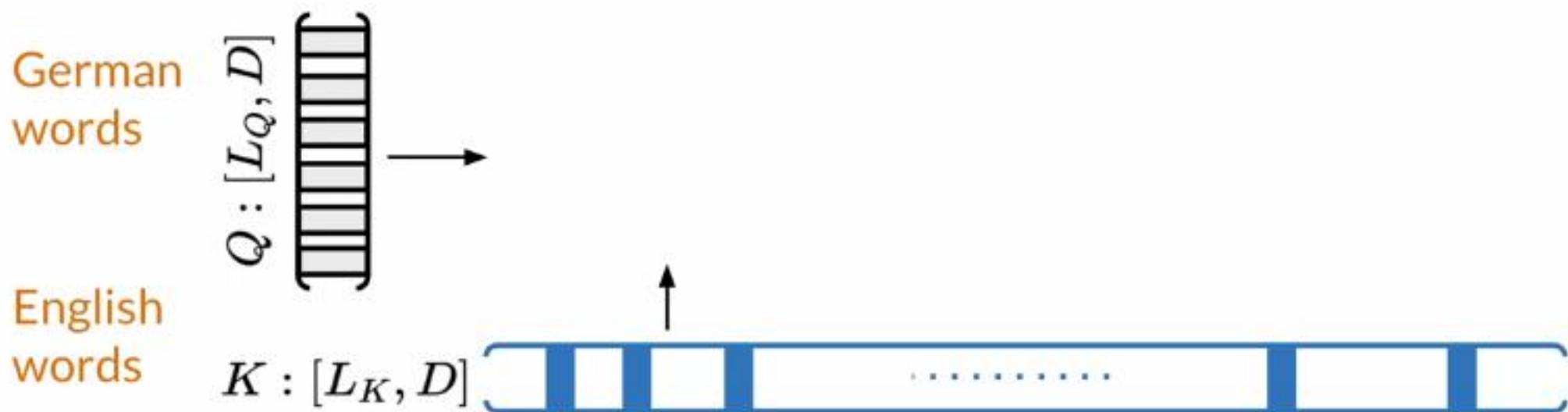
# Attention math

English  
words

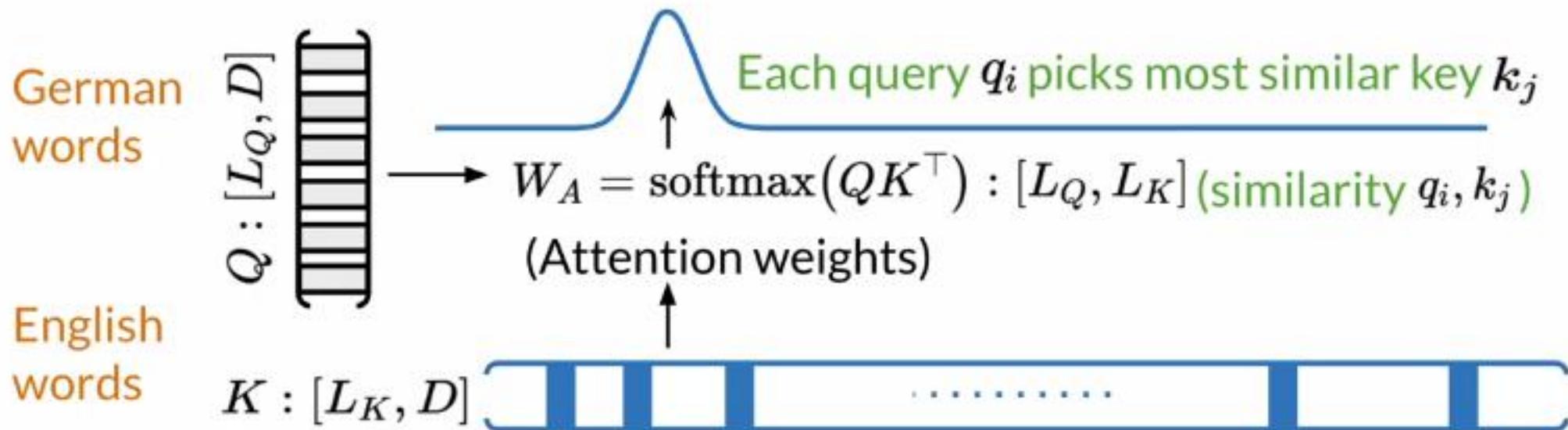
$K : [L_K, D]$



# Attention math



# Attention math



# Attention math

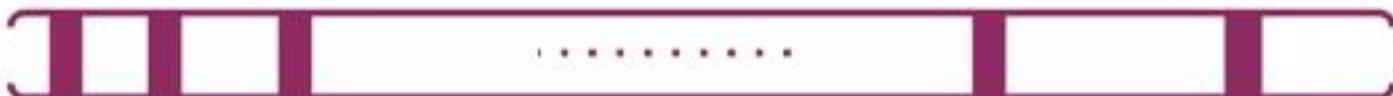
Embedding

English words

$$Z = W_A V : [L_Q, D] \quad (\text{Result})$$

Query  $q_i$  gets  $v_j$  from the most similar  $k_j$

$$V : [L_K, D]$$



German words

$$Q : [L_Q, D]$$



Each query  $q_i$  picks most similar key  $k_j$

$$W_A = \text{softmax}(QK^\top) : [L_Q, L_K] \quad (\text{similarity } q_i, k_j)$$

(Attention weights)

English words

$$K : [L_K, D]$$



# Attention formula

# Attention formula

$Z$   
(Result)

# Attention formula

$$Z = \text{attention}(Q, K, V)$$

(Result)

## Attention formula

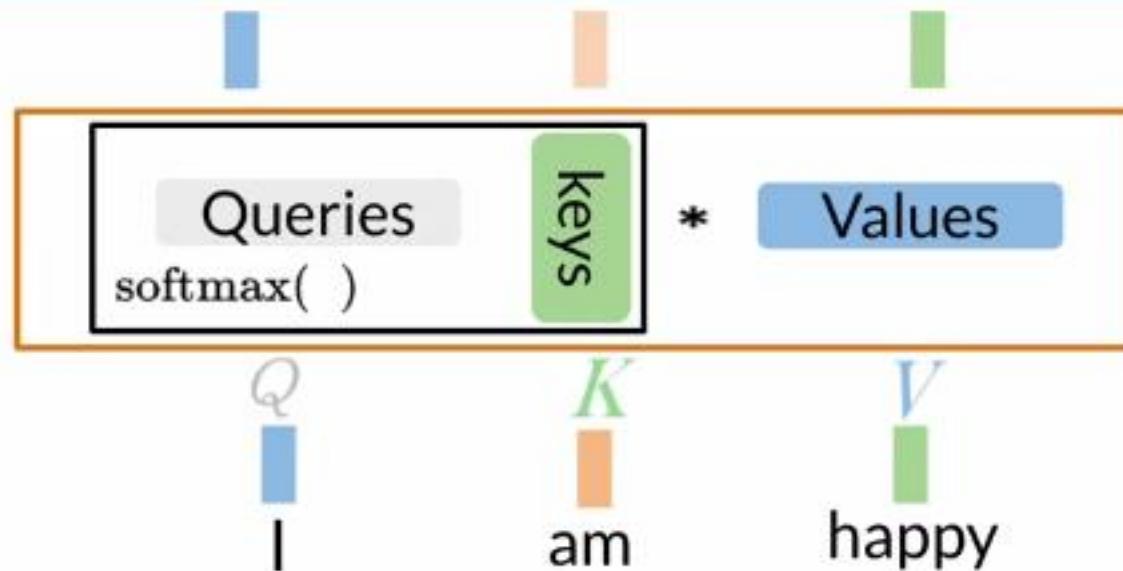
$$Z = \text{attention}(Q, K, V) = \text{softmax}(QK^\top)V$$

(Result)

# Attention formula

$$Z = \text{attention}(Q, K, V) = \text{softmax}(QK^\top)V = W_A V$$

(Result)



# Attention formula

$$Z = \text{attention}(Q, K, V) = \text{softmax}(QK^T)V = W_A V$$

(Result)



$Q$   
I

$K$   
am

$V$   
happy

# Summary

- Dot-product Attention is essential for Transformer
- The input to Attention are queries, keys, and values
- A softmax function makes attention more focused on best keys
- GPUs and TPUs is advisable for matrix multiplications



# Outline

- Ways of Attention
- Overview of Causal Attention
- Math behind causal attention



# Three ways of attention

# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)

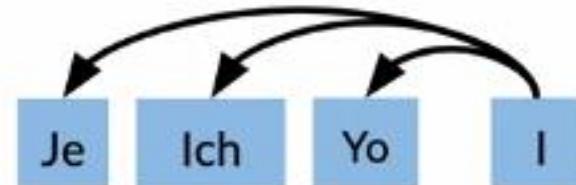
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)

Je Ich Yo I

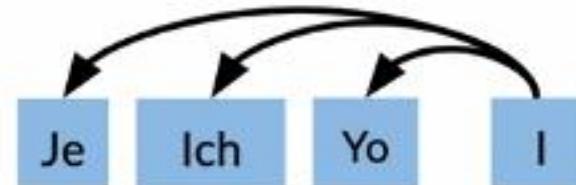
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)



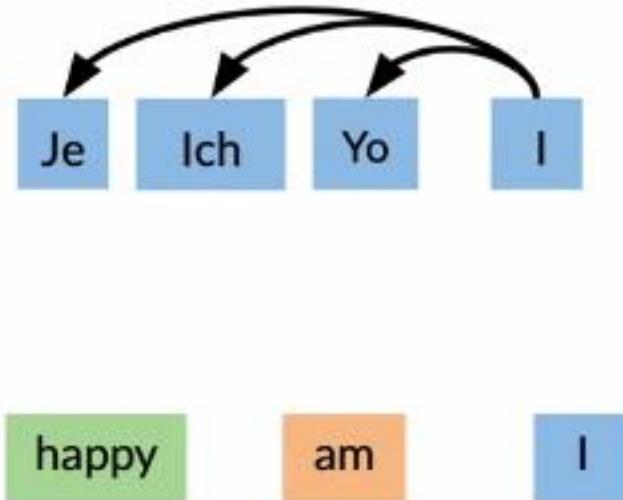
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)



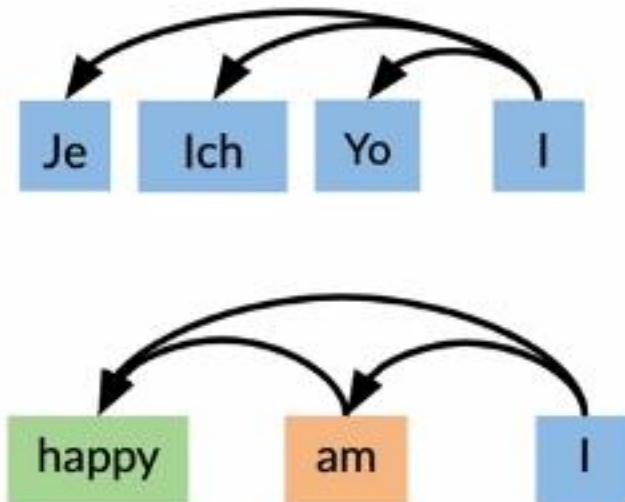
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)



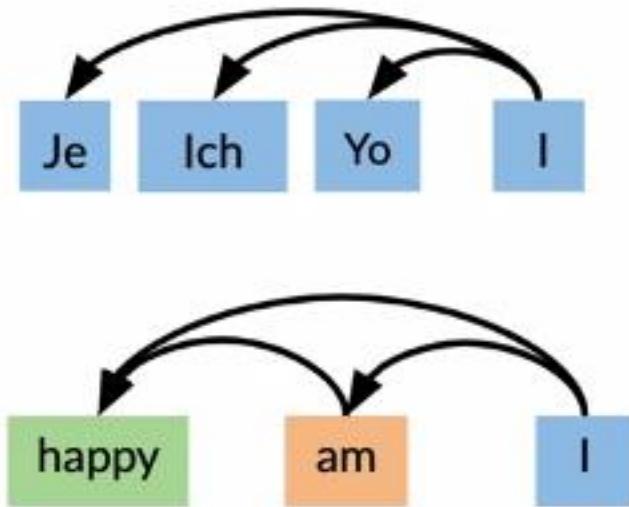
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)



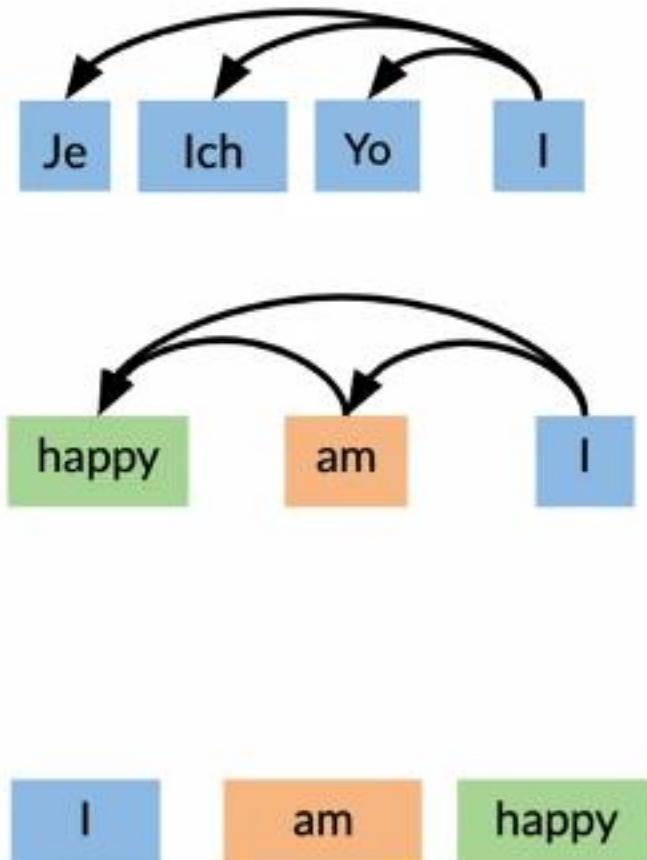
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)
- **Bi-directional self attention:** In one sentence, words look at both previous and future words



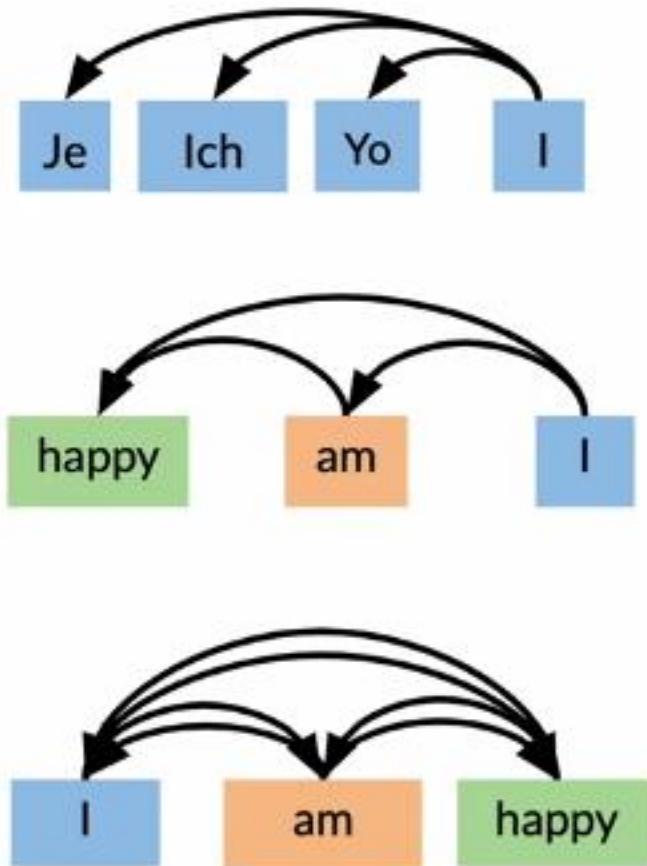
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)
- **Bi-directional self attention:** In one sentence, words look at both previous and future words



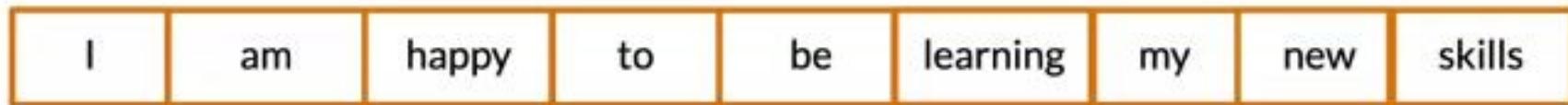
# Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)
- **Bi-directional self attention:** In one sentence, words look at both previous and future words



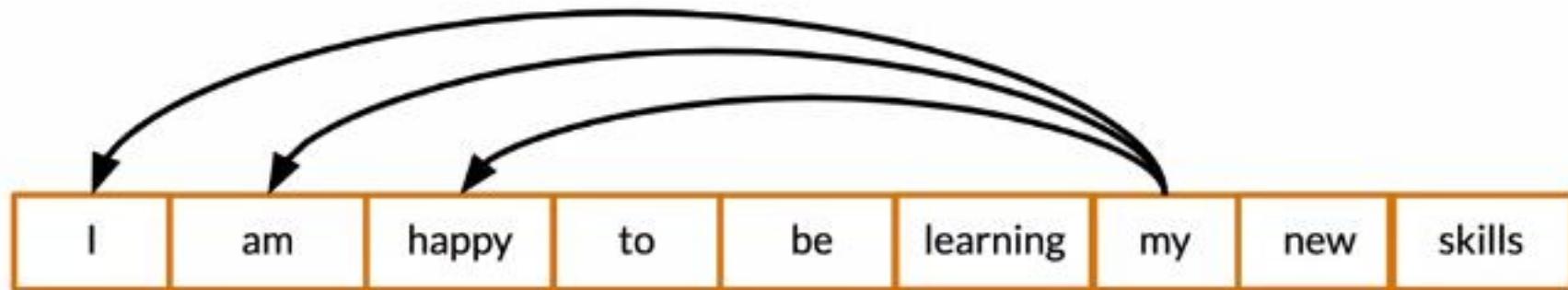
# Causal attention

- Queries and keys are words from the same sentence



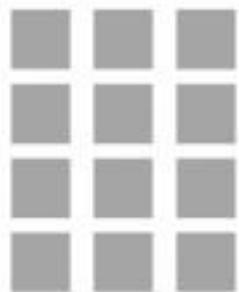
# Causal attention

- Queries and keys are words from the same sentence
- Queries should only be allowed to look at words before



# Causal attention math

# Causal attention math



$Q$

# Causal attention math

$$\begin{matrix} \text{Gray Grid} \\ Q \end{matrix} \times \begin{matrix} \text{Green Grid} \\ K^\top \end{matrix}$$

# Causal attention math

$$\begin{matrix} \begin{matrix} \textcolor{gray}{\square} & \textcolor{gray}{\square} & \textcolor{gray}{\square} \\ \textcolor{gray}{\square} & \textcolor{gray}{\square} & \textcolor{gray}{\square} \\ \textcolor{gray}{\square} & \textcolor{gray}{\square} & \textcolor{gray}{\square} \\ \textcolor{gray}{\square} & \textcolor{gray}{\square} & \textcolor{gray}{\square} \end{matrix} & \times & \begin{matrix} \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \\ \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \\ \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \\ \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} & \textcolor{green}{\square} \end{matrix} \\ Q & & K^\top \\ \begin{matrix} \textcolor{magenta}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \textcolor{magenta}{\square} & \textcolor{magenta}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \textcolor{magenta}{\square} & \textcolor{magenta}{\square} & \textcolor{magenta}{\square} & \textcolor{blue}{\square} \\ \textcolor{magenta}{\square} & \textcolor{magenta}{\square} & \textcolor{magenta}{\square} & \textcolor{blue}{\square} \end{matrix} & & K^\top \\ QK^\top & & \end{matrix}$$

# Causal attention math

$$\begin{matrix} \begin{matrix} Q & \times & K^\top \\ \downarrow & & \downarrow \\ QK^\top & + & \end{matrix} \end{matrix}$$

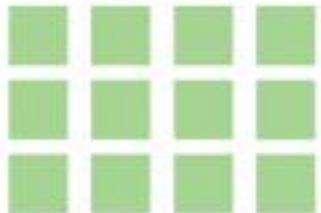
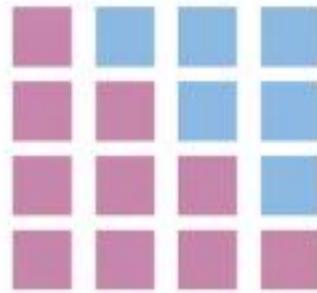
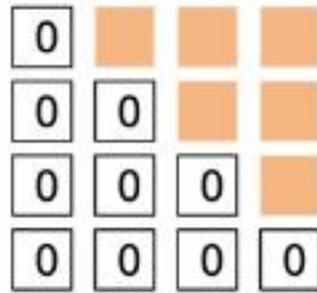
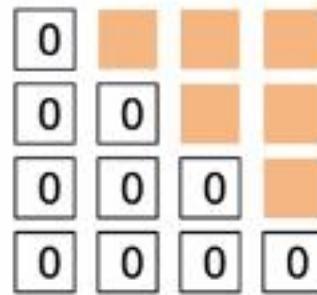
The diagram illustrates the computation of causal attention. It shows two input matrices,  $Q$  and  $K^\top$ , being multiplied together. The result of this multiplication is then added to another term, represented by a plus sign.

$Q$  is a 4x4 matrix with gray squares.  $K^\top$  is a 4x4 matrix with green squares. The result of their multiplication,  $QK^\top$ , is a 4x4 matrix where each row has three pink squares and one blue square. This pattern indicates that each row in  $Q$  attends to the first three columns of  $K^\top$ .

# Causal attention math

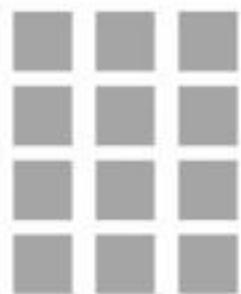
$$\begin{array}{c} \begin{matrix} \text{gray} & \text{gray} & \text{gray} \\ \text{gray} & \text{gray} & \text{gray} \\ \text{gray} & \text{gray} & \text{gray} \\ \text{gray} & \text{gray} & \text{gray} \end{matrix} \times \begin{matrix} \text{green} & \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} & \text{green} \end{matrix} + \begin{matrix} 0 & \text{orange} & \text{orange} & \text{orange} \\ 0 & 0 & \text{orange} & \text{orange} \\ 0 & 0 & 0 & \text{orange} \\ 0 & 0 & 0 & 0 \end{matrix} \\ Q \quad \quad \quad K^\top \quad \quad \quad M \end{array}$$
$$\begin{array}{c} \begin{matrix} \text{magenta} & \text{blue} & \text{blue} & \text{blue} \\ \text{magenta} & \text{magenta} & \text{blue} & \text{blue} \\ \text{magenta} & \text{magenta} & \text{magenta} & \text{blue} \\ \text{magenta} & \text{magenta} & \text{magenta} & \text{blue} \end{matrix} + \begin{matrix} 0 & \text{orange} & \text{orange} & \text{orange} \\ 0 & 0 & \text{orange} & \text{orange} \\ 0 & 0 & 0 & \text{orange} \\ 0 & 0 & 0 & 0 \end{matrix} \\ QK^\top \quad \quad \quad M \end{array}$$

# Causal attention math

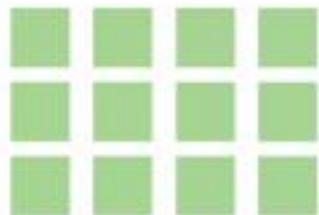
 $Q$  $\times$  $K^\top$  $QK^\top$  $+$  $M$  $+$  $M$ 

→ Minus infinity -in practice, a huge negative number

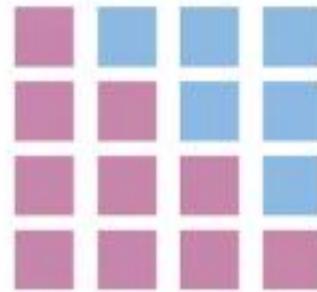
# Causal attention math

$$Q$$


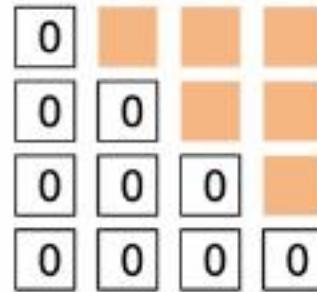
 $\times$ 

$$K^\top$$


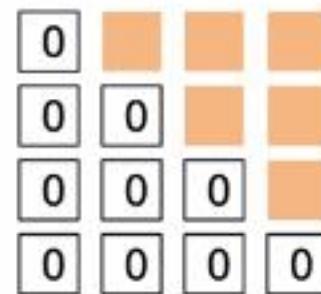
 $Q$ 

$$Q K^\top$$


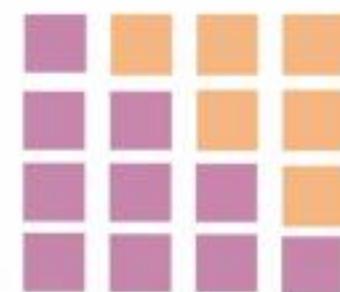
 $K^\top$  $+$ 

$$M$$


→ Minus infinity -in practice, a huge negative number

$$M$$


 $=$ 

$$Q K^\top + I M$$


 $M$  $=$

# Causal attention math

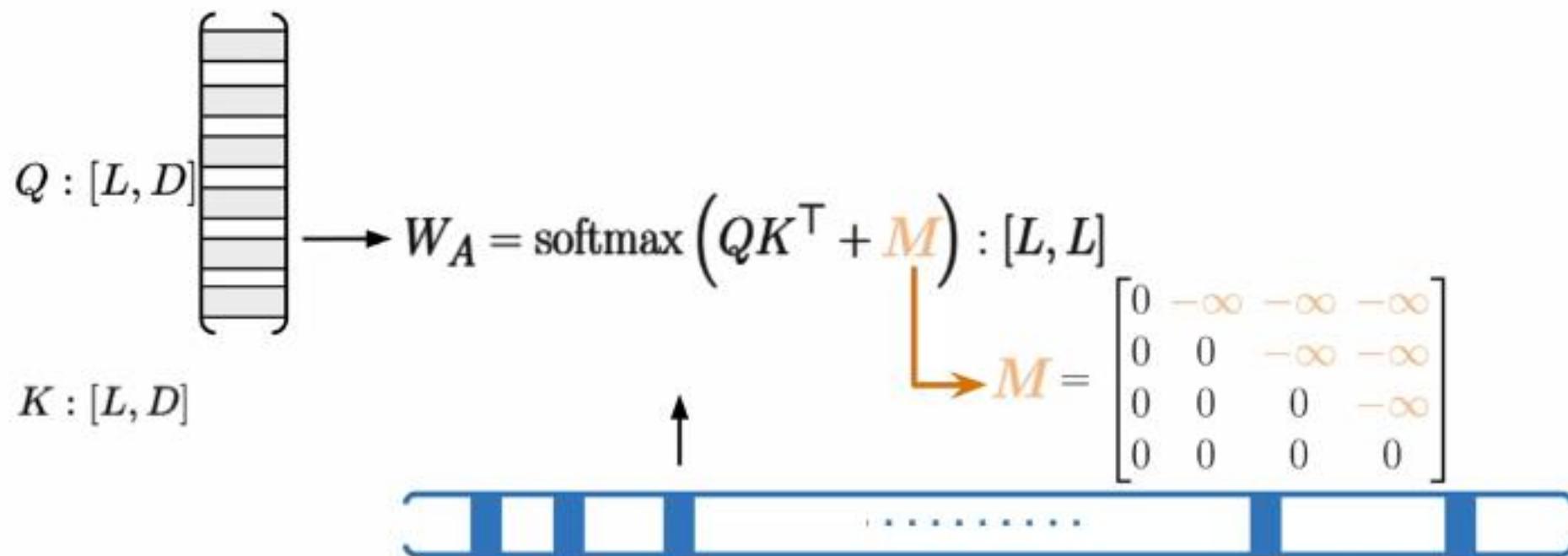
$K : [L, D]$

$$W_A = \text{softmax} \left( QK^\top + \mathbf{M} \right) : [L, L]$$

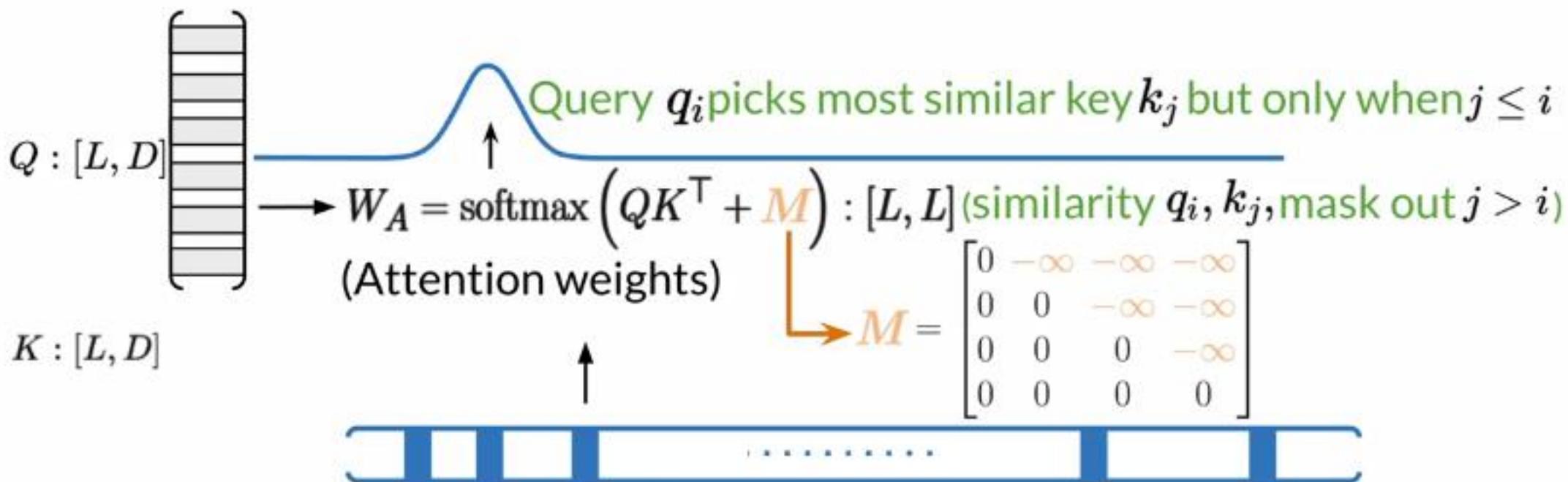
$\mathbf{M} = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$



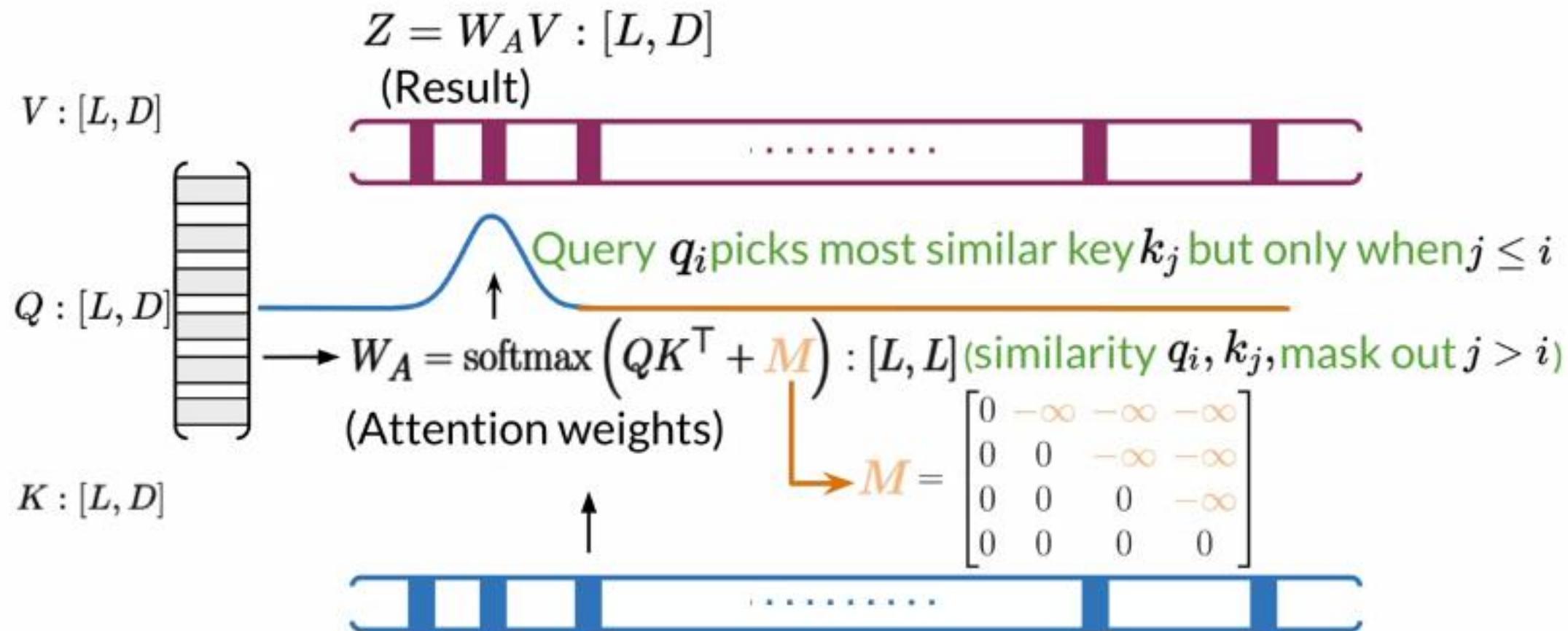
# Causal attention math



# Causal attention math



# Causal attention math



# Summary

- There are three main ways of Attention: Encoder/Decoder, Causal and Bi-directional type
- In causal attention, queries and keys come from the same sentence and queries search among words before only

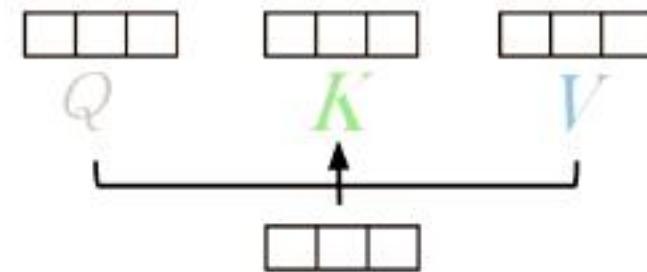


# Outline

- Intuition of Multi-Head Attention
- Scaled dot-product and concatenation
- Multi-Head Attention formula

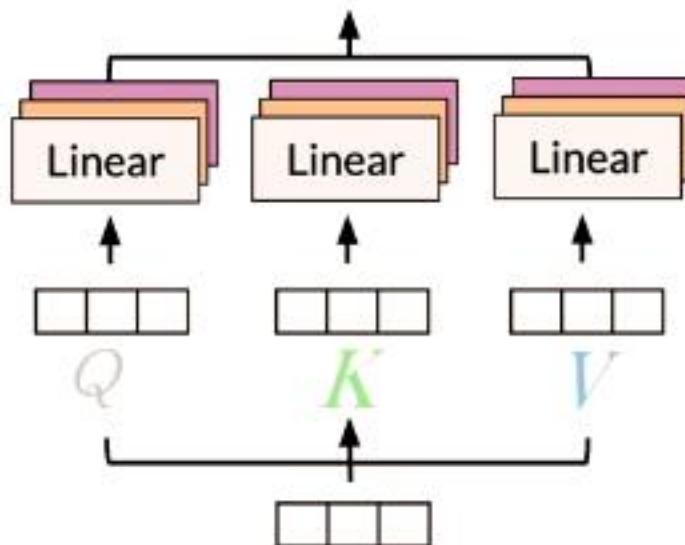


# Multi-Head Attention



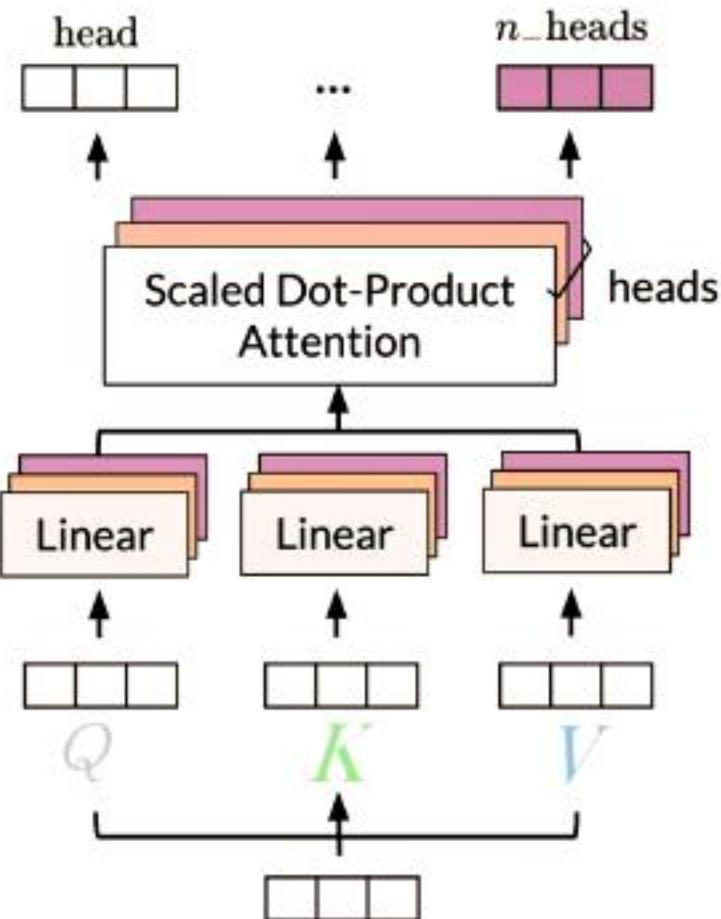
# Multi-Head Attention

- Each head uses different linear transformations to represent words



# Multi-Head Attention

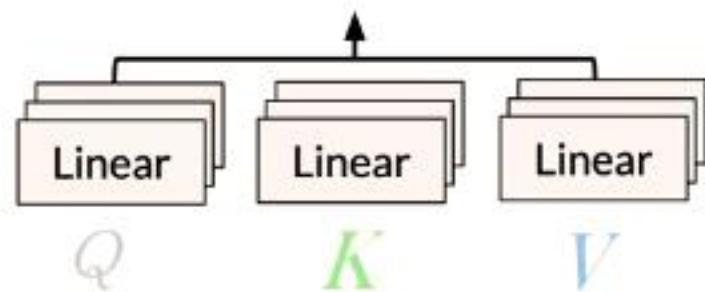
- Each head uses different linear transformations to represent words
- Different heads can learn different relationships between words



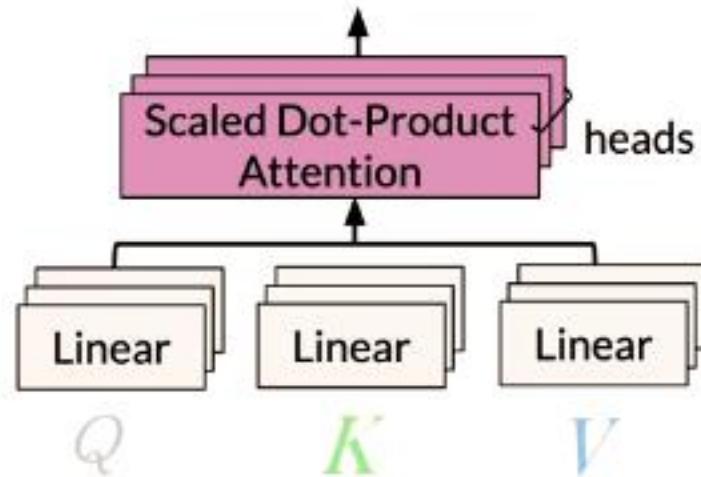
# Multi-Head Attention - Overview

$Q$        $K$        $V$

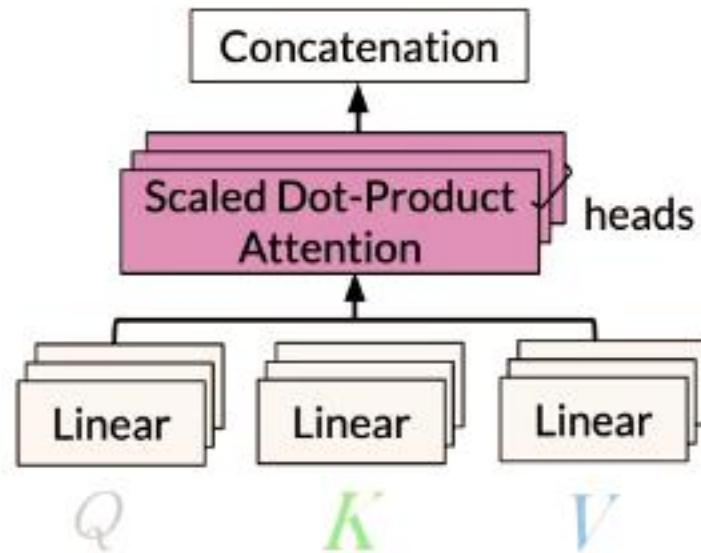
# Multi-Head Attention - Overview



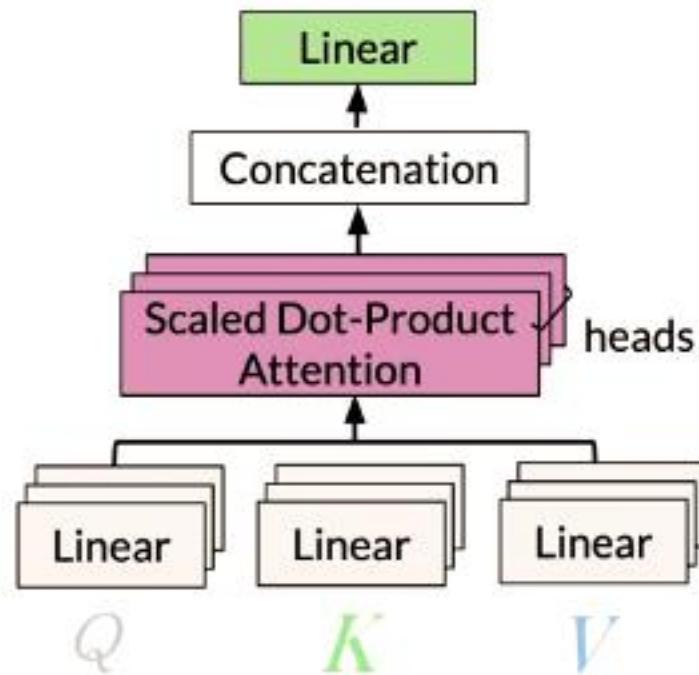
# Multi-Head Attention - Overview



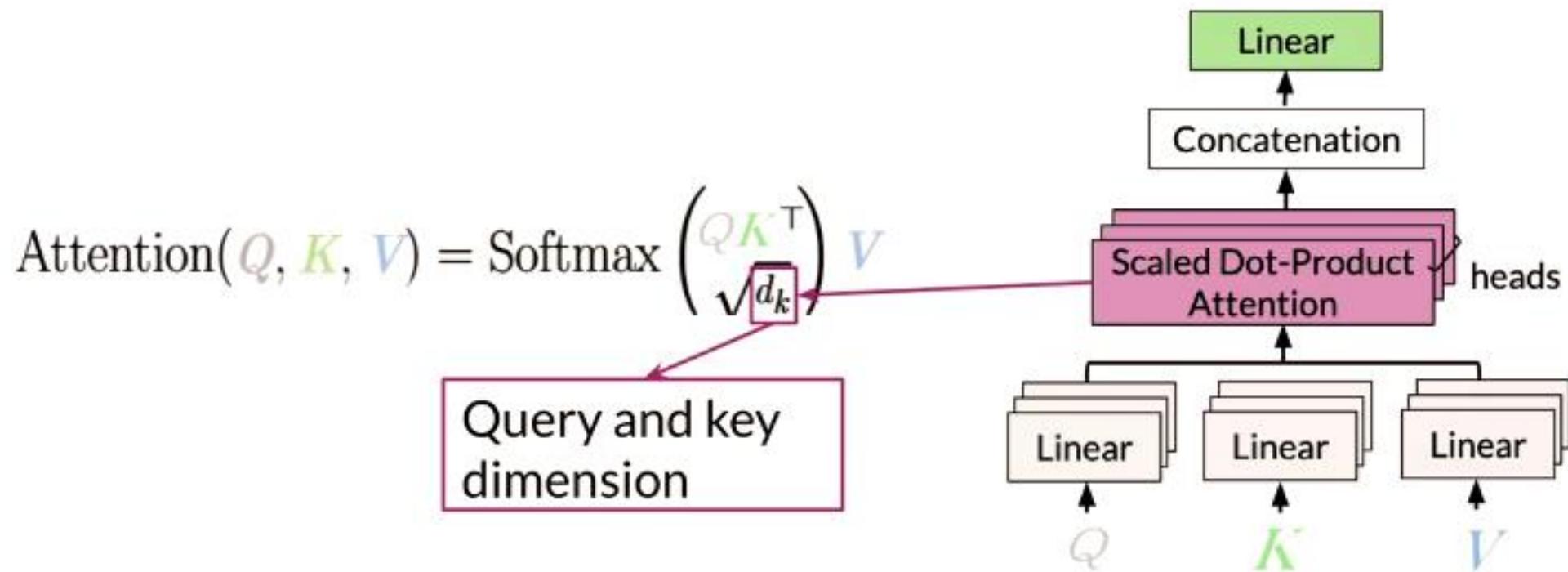
# Multi-Head Attention - Overview



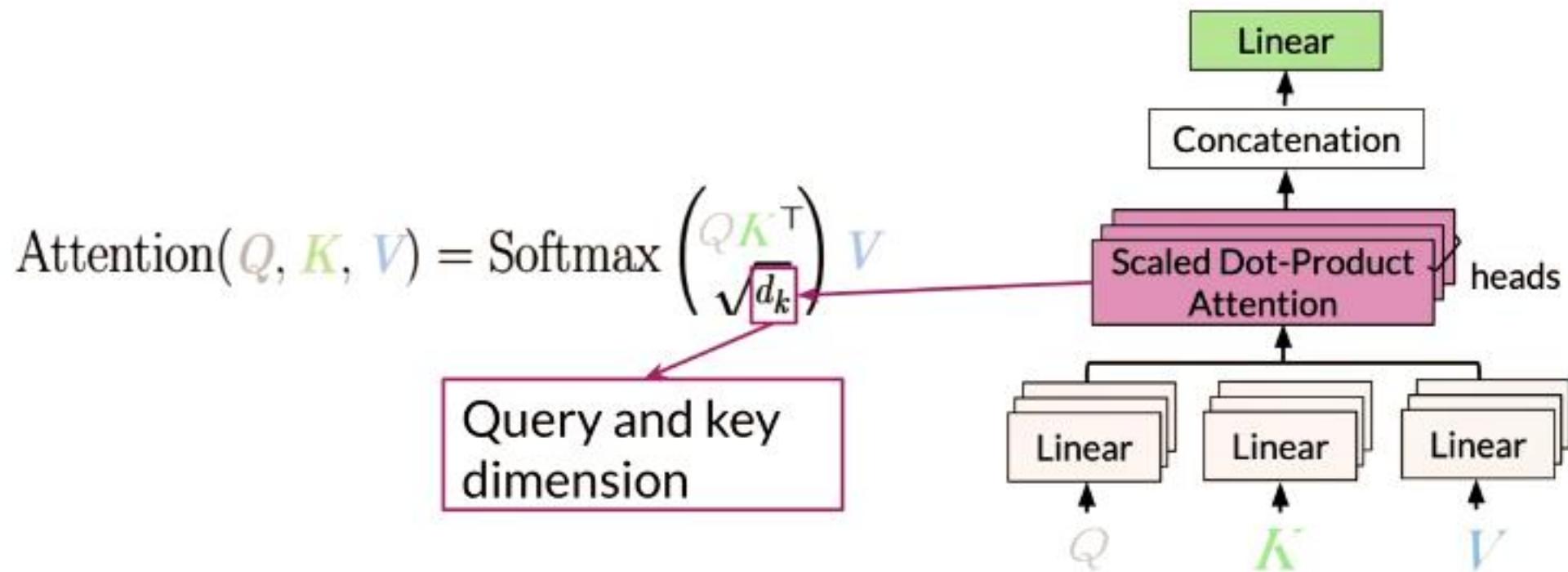
# Multi-Head Attention - Overview



# Multi-Head Attention - Scaled dot product



# Multi-Head Attention - Scaled dot product



# Multi-Head Attention - Concatenation

- Input( $Q$ ,  $K$ ,  $V$ ): [batch, length, d\_model]

$\uparrow$        $\uparrow$        $\uparrow$   
 $Q$        $K$        $V$

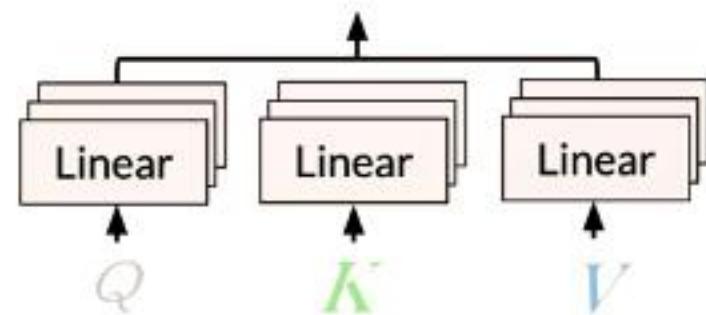
# Multi-Head Attention - Concatenation

- Input( $Q$ ,  $K$ ,  $V$ ): [batch, length,  $d_{model}$ ]  
512, 1024

$\uparrow$        $\uparrow$        $\uparrow$   
 $Q$        $K$        $V$

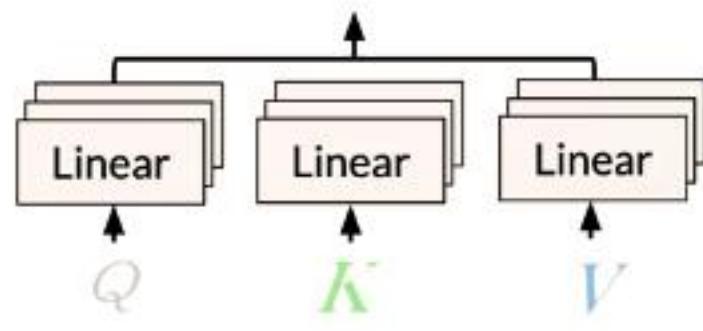
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length, d\_model]
- Linear layer: [batch, length, n\_heads \* d\_head]



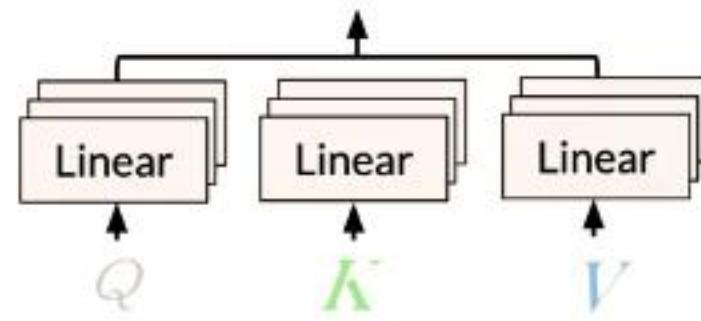
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length, d\_model]
- Linear layer: [batch, length, n\_heads \* d\_head]



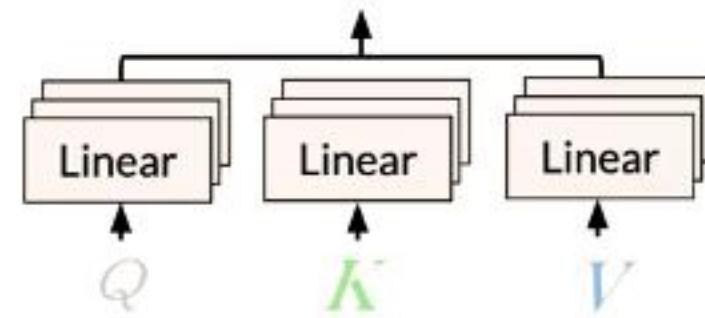
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length,  $d_{model}$ ]
- Linear layer: [batch, length,  $n_{heads} * d_{head}$ ]  
4, 6, 16, ...



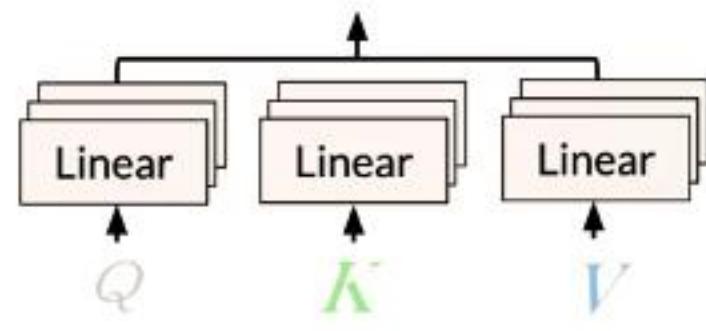
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length,  $d_{model}$ ]
- Linear layer: [batch, length,  $n_{heads} * d_{head}$ ]  
 $\downarrow \quad \downarrow$   
4, 6, 16, ...    64, 128



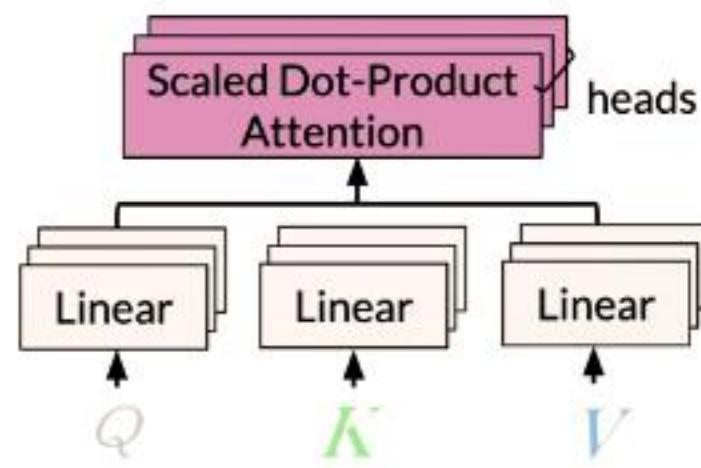
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length, d\_model]
- Linear layer: [batch, length, n\_heads \* d\_head]
- Transpose: [batch, n\_heads, length, d\_head]



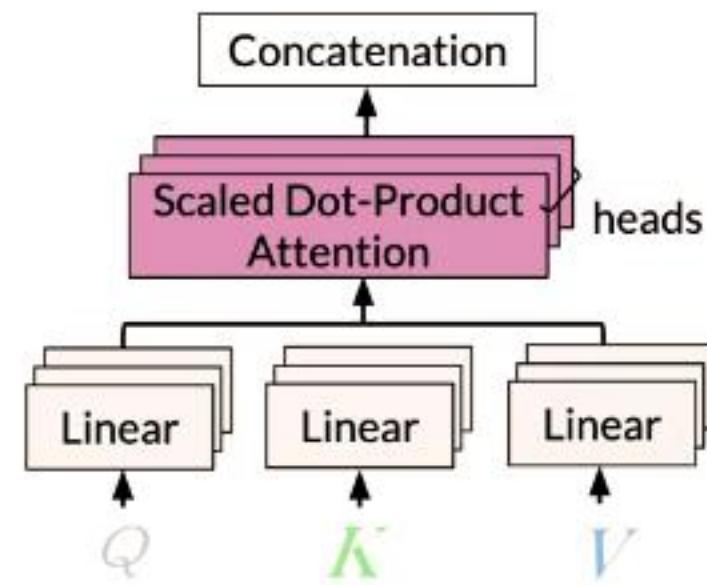
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length, d\_model]
- Linear layer: [batch, length, n\_heads \* d\_head]
- Transpose: [batch, n\_heads, length, d\_head]
- Apply attention treating n\_heads like batch



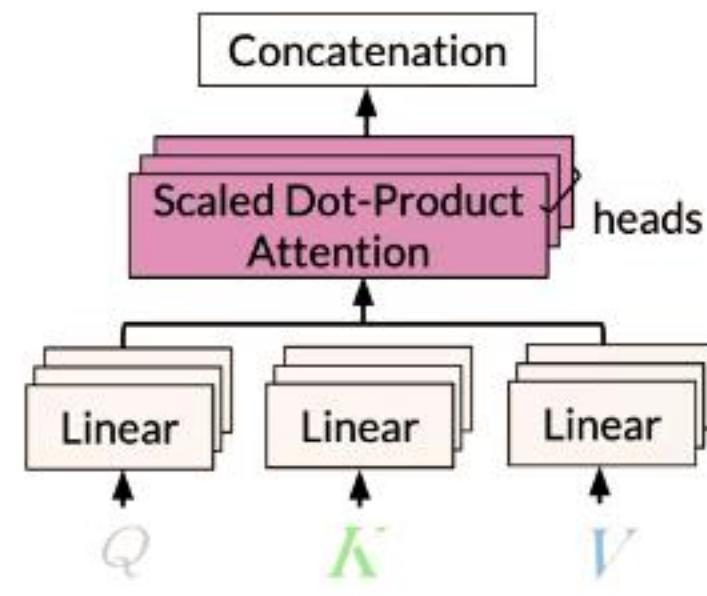
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length, d\_model]
- Linear layer: [batch, length, n\_heads \* d\_head]
- Transpose: [batch, n\_heads, length, d\_head]
- Apply attention treating n\_heads like batch
- Result shape: [batch, n\_heads, length, d\_head]



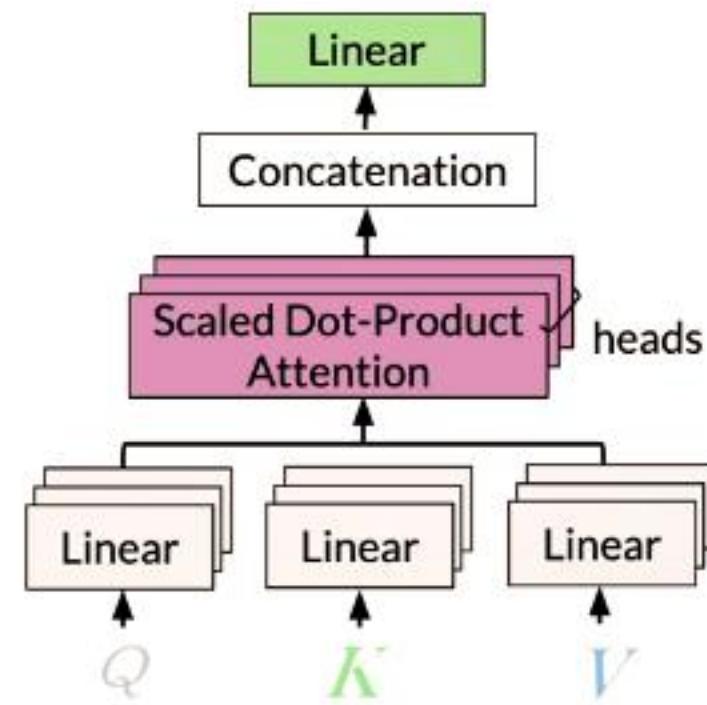
# Multi-Head Attention - Concatenation

- Input( $Q, K, V$ ): [batch, length, d\_model]
- Linear layer: [batch, length, n\_heads \* d\_head]
- Transpose: [batch, n\_heads, length, d\_head]
- Apply attention treating n\_heads like batch
- Result shape: [batch, n\_heads, length, d\_head]
- Transpose: [batch, length, n\_heads \* d\_head]



# Multi-Head Attention - Concatenation

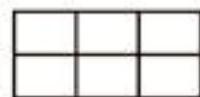
- Input( $Q, K, V$ ): [batch, length,  $d_{model}$ ]
- Linear layer: [batch, length,  $n_{heads} * d_{head}$ ]
- Transpose: [batch,  $n_{heads}$ , length,  $d_{head}$ ]
- Apply attention treating  $n_{heads}$  like batch
- Result shape: [batch,  $n_{heads}$ , length,  $d_{head}$ ]
- Transpose: [batch, length,  $n_{heads} * d_{head}$ ]
- Linear layer into: [batch size, length,  $d_{model}$ ]



# Multi-Head Attention math

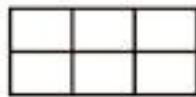
# Multi-Head Attention math

Embedding

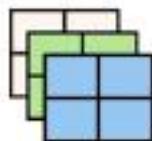


# Multi-Head Attention math

Embedding

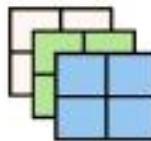


$Q_i \textcolor{brown}{K}_i \textcolor{teal}{V}_i$

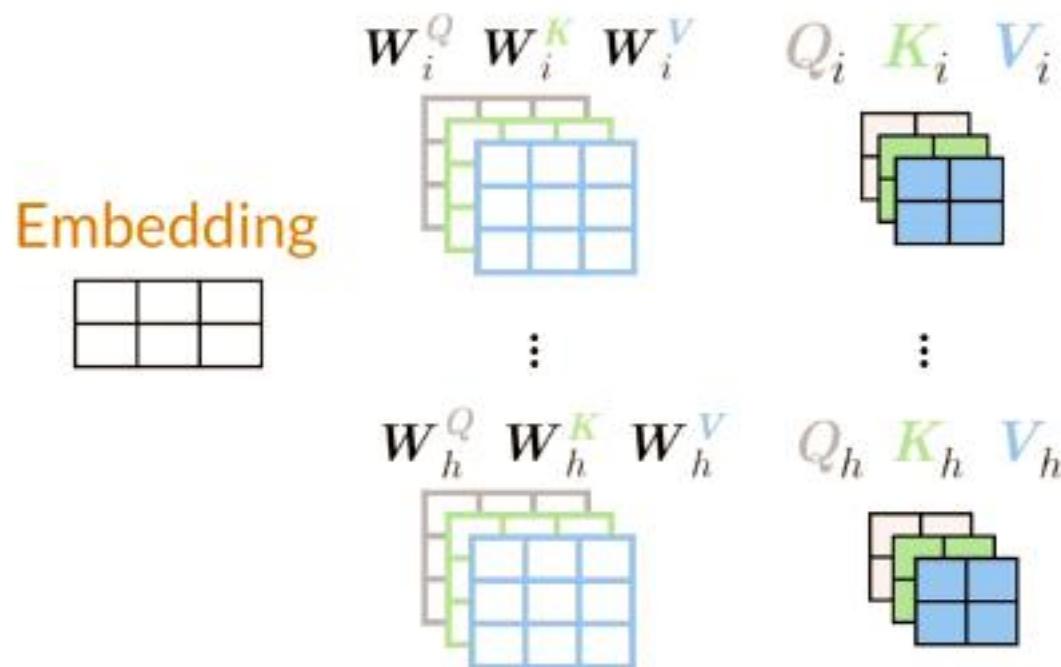


:

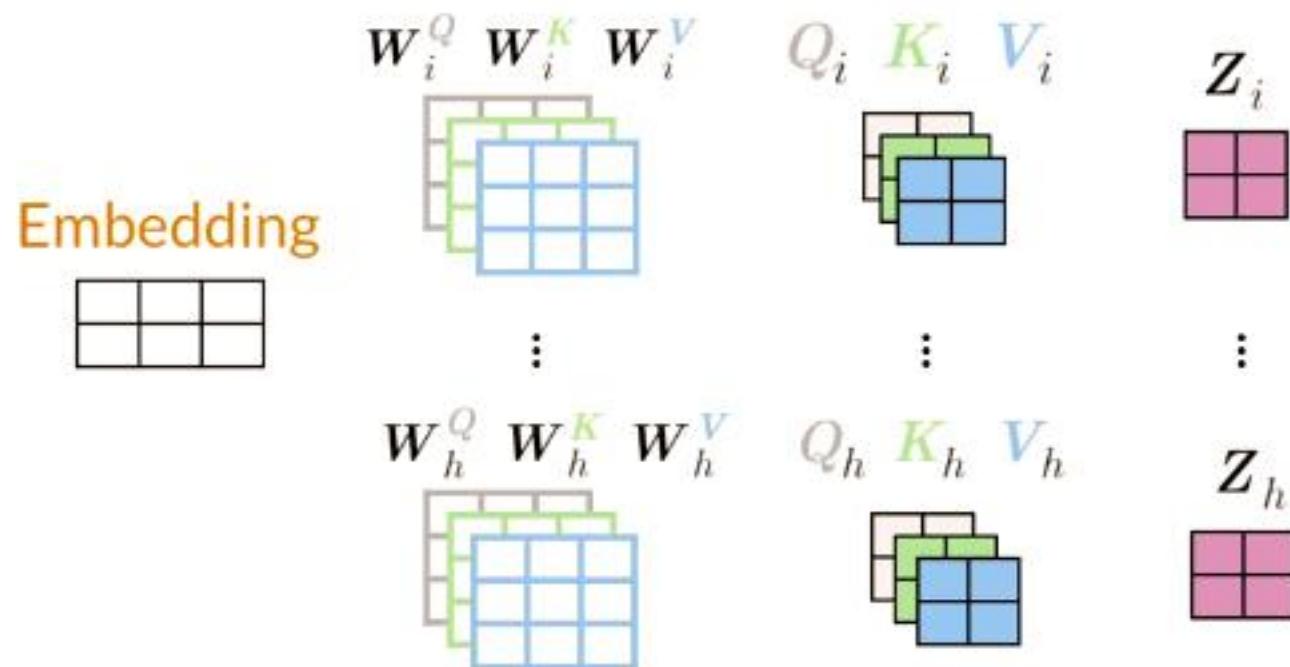
$Q_h \textcolor{brown}{K}_h \textcolor{teal}{V}_h$



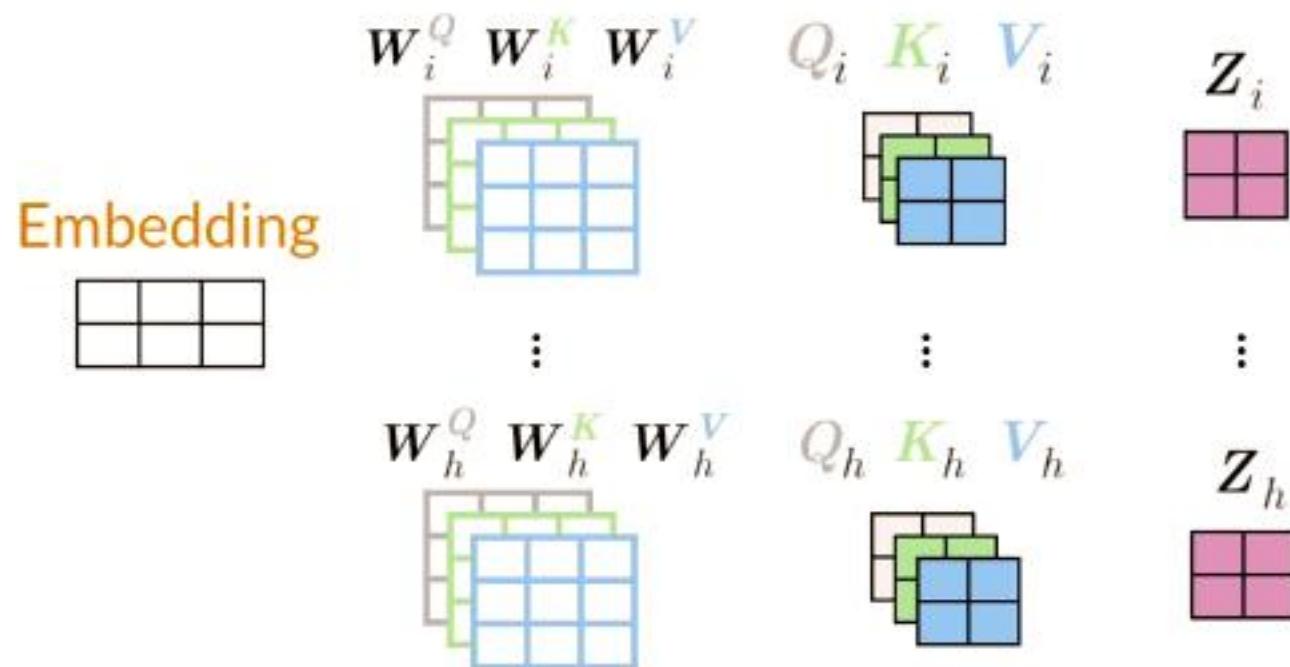
# Multi-Head Attention math



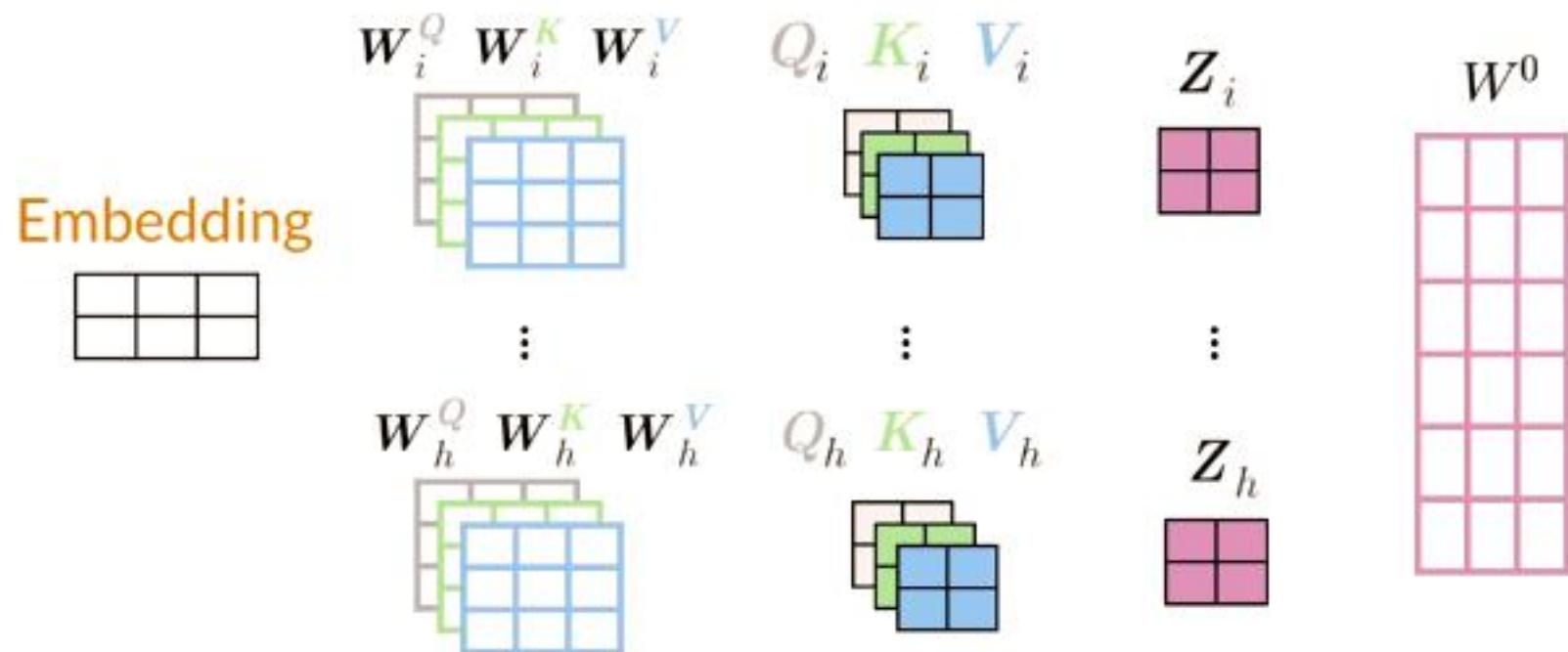
# Multi-Head Attention math



# Multi-Head Attention math



# Multi-Head Attention math



# Multi-Head Attention Formula

$$Q \text{ } K \text{ } V \quad Q \text{ } K \text{ } V$$

# Multi-Head Attention Formula

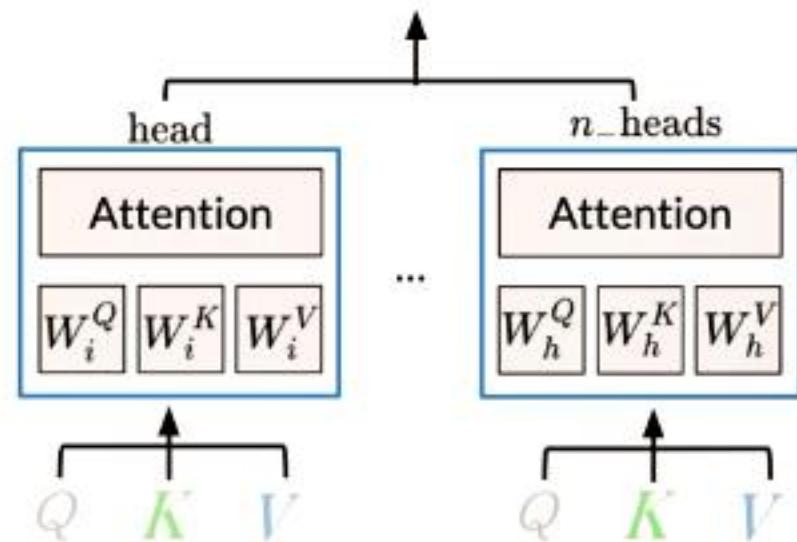
$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^0$$



# Multi-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^0$$

where  $h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

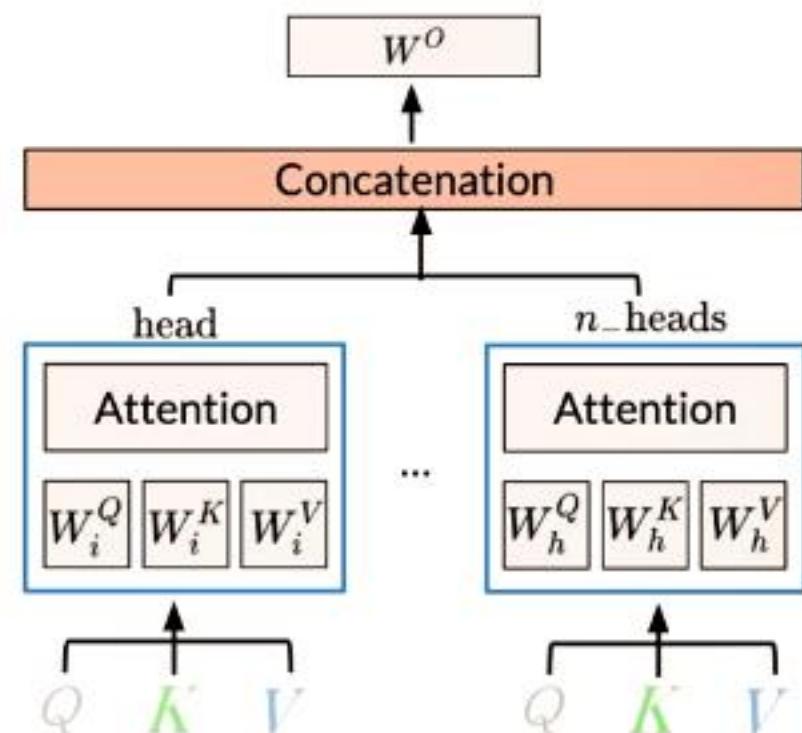


# Multi-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^0$$

where  $h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

Each head  $h_i$  is the attention function of **Query, Key and Value** with trainable parameters  $(W_i^Q, W_i^K, W_i^V)$

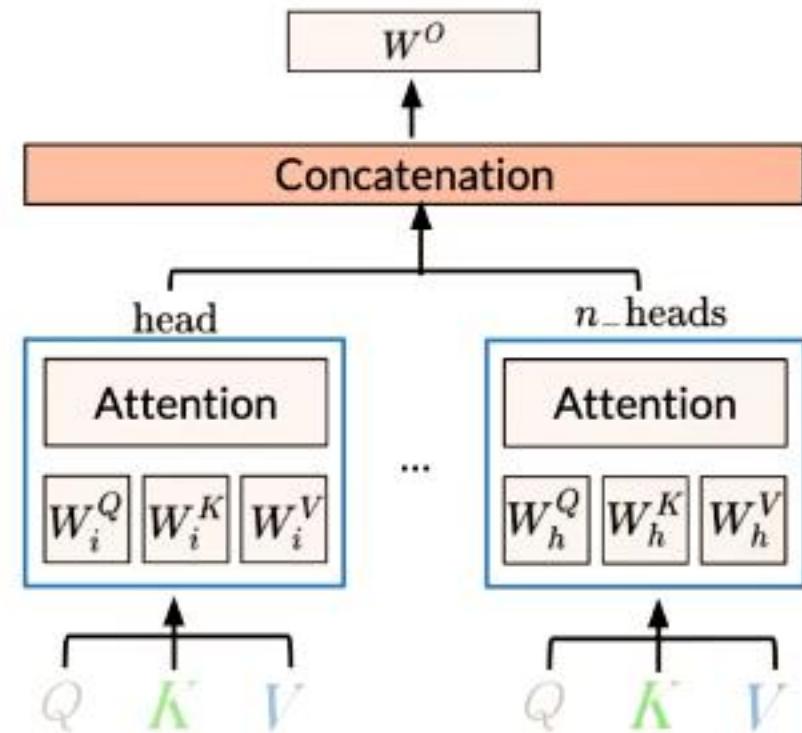


# Multi-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^0$$

$$\text{where } h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Each head  $h_i$  is the attention function of **Query, Key and Value** with trainable parameters  $(W_i^Q, W_i^K, W_i^V)$



# Summary

- Different heads can learn different relationship between words
- Scaled dot-product is adequate for Multi-Head Attention
- Multi-Headed models attend to information from different representations at different positions



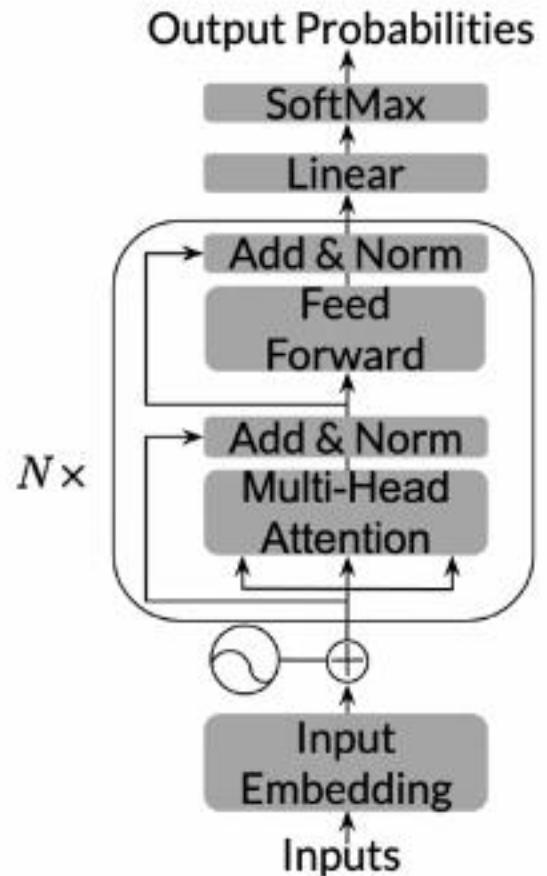
# Outline

- Overview of Transformer decoder
- Implementation (decoder and feed-forward block)



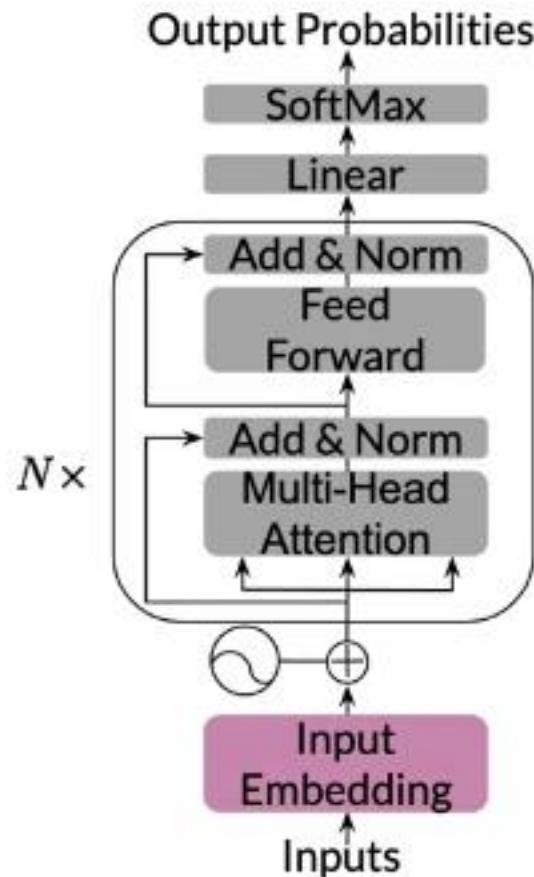
# Transformer decoder

## Overview



# Transformer decoder

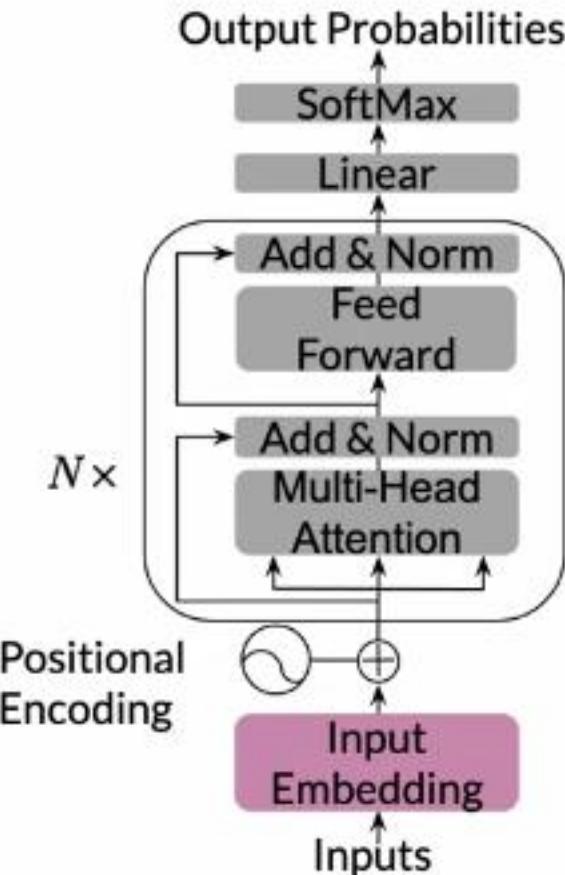
## Overview



- input: sentence or paragraph
  - we predict the next word

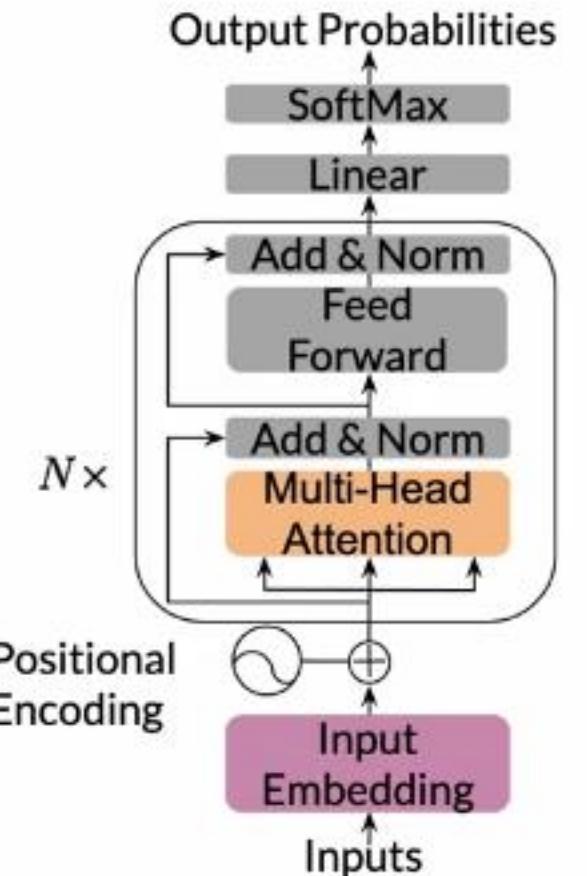
# Transformer decoder

## Overview



- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )

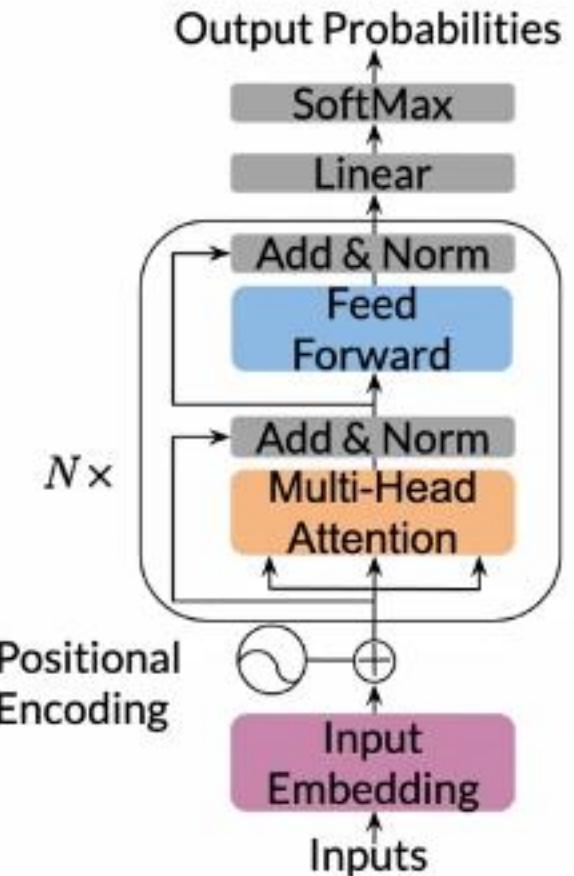
# Transformer decoder



## Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )
- multi-head attention looks at previous words

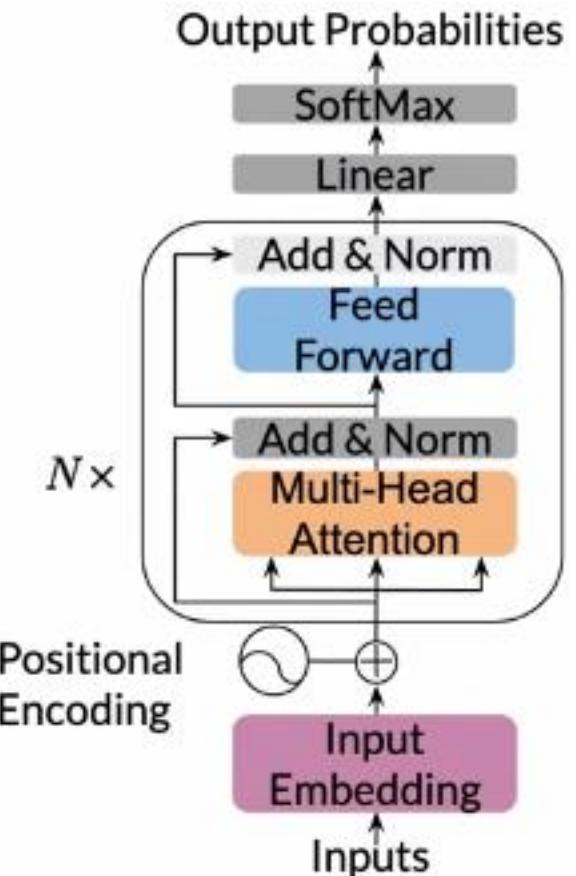
# Transformer decoder



## Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!

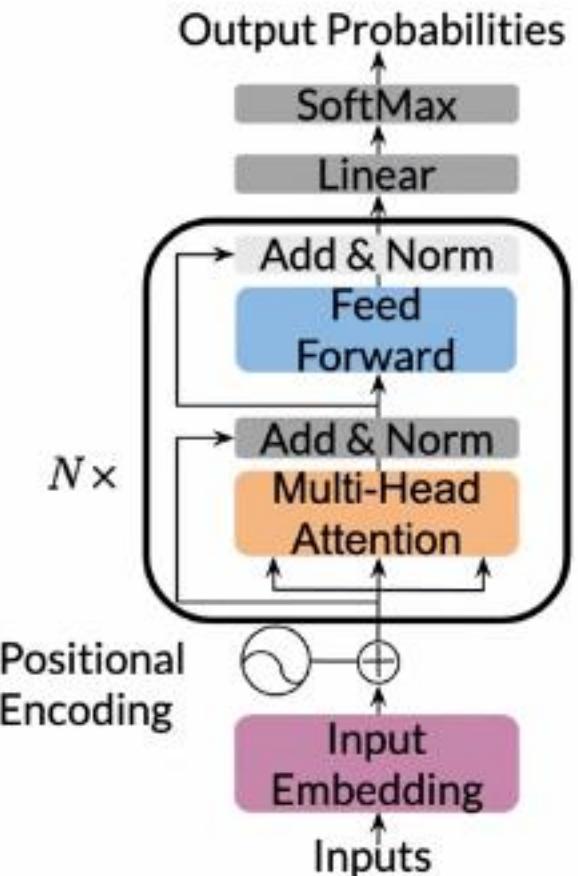
# Transformer decoder



## Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!
- residual connection with layer normalization

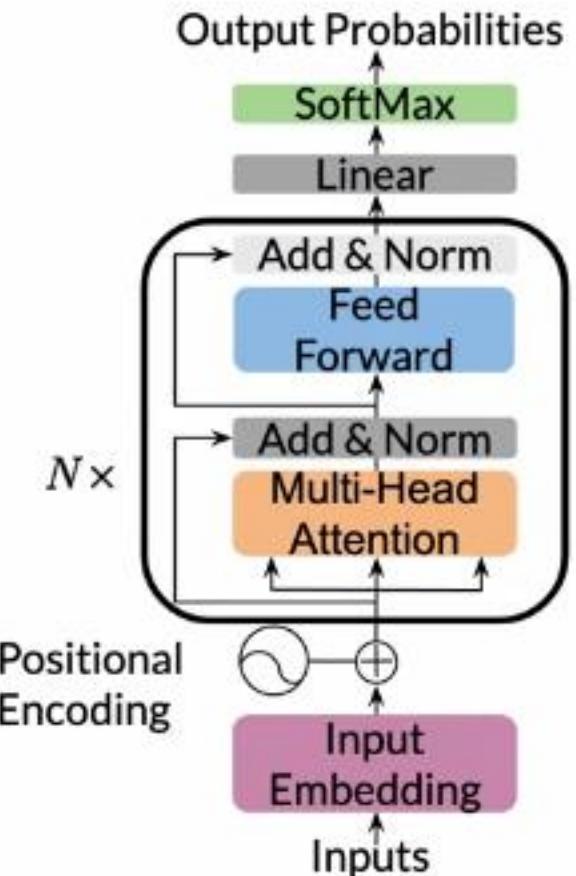
# Transformer decoder



## Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!
- residual connection with layer normalization
- repeat N times

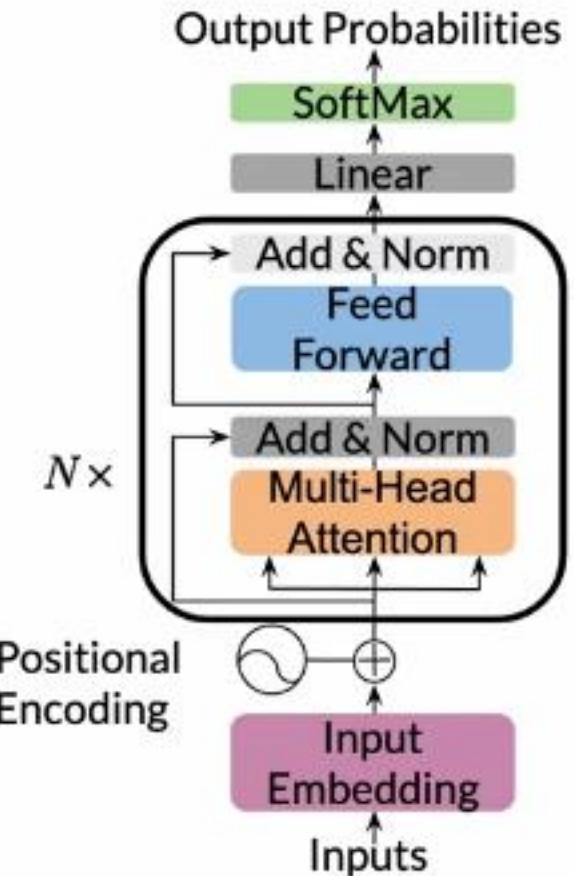
# Transformer decoder



## Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!
- residual connection with layer normalization
- repeat N times
- dense layer and softmax for output

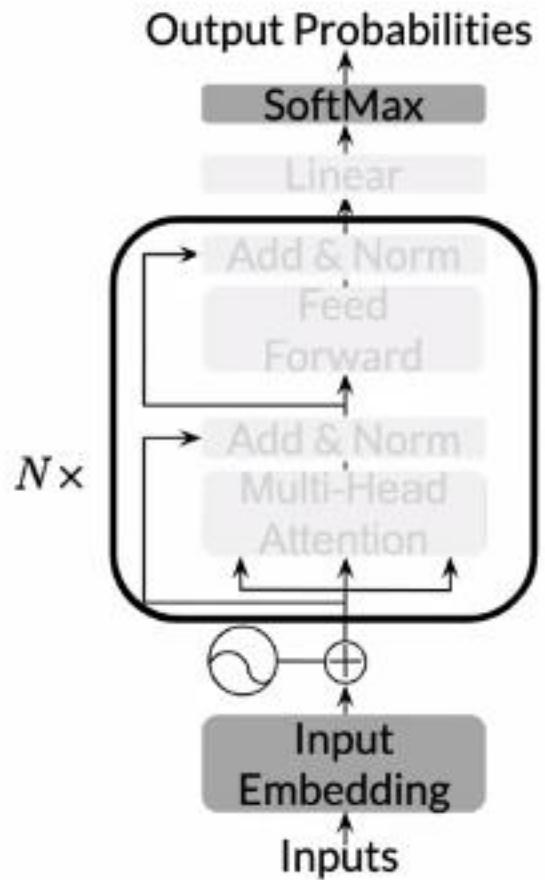
# Transformer decoder



## Overview

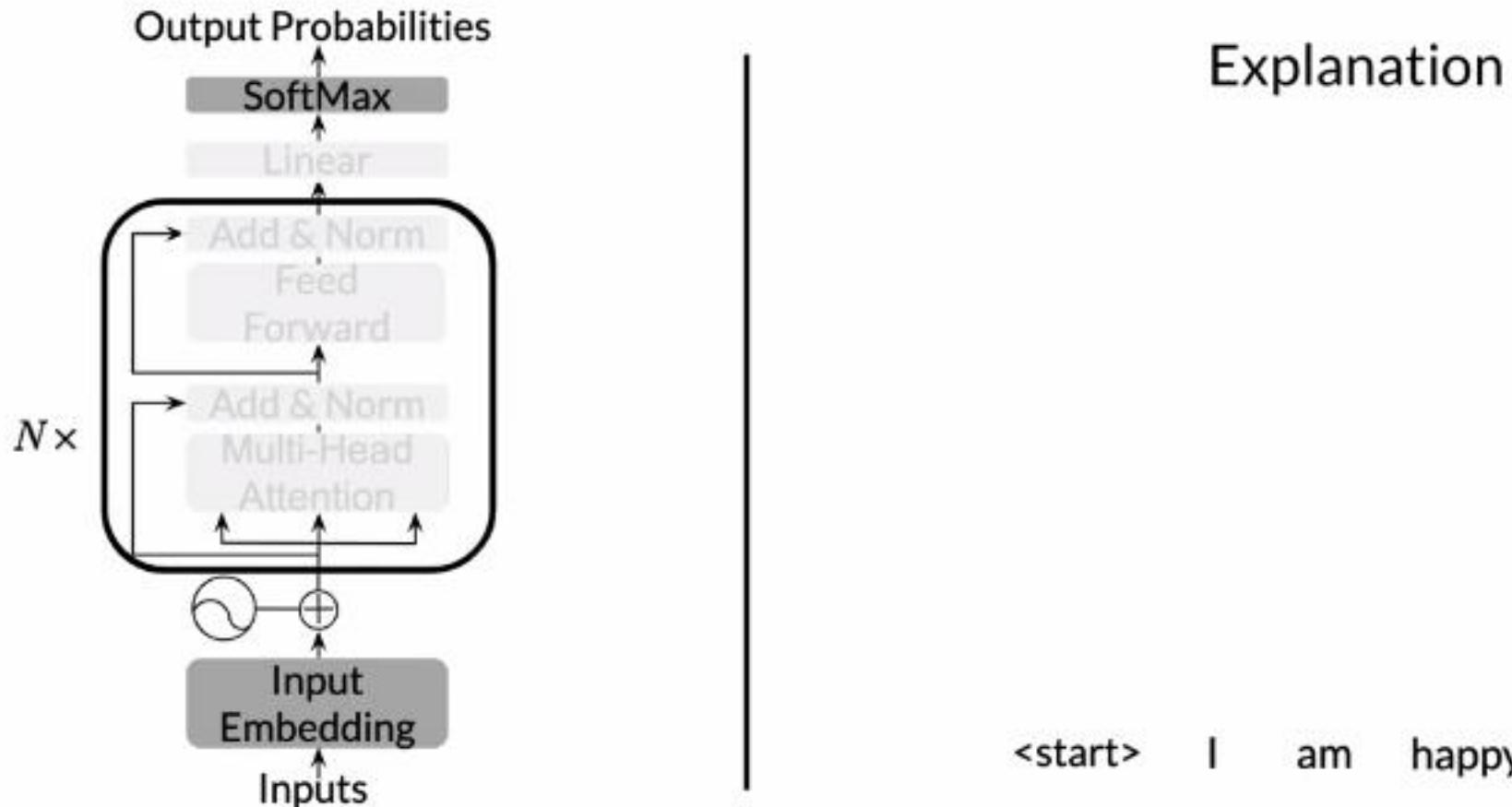
- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encoding
  - (vectors representing  $\{0, 1, 2, \dots, K\}$ )
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!
- residual connection with layer normalization
- repeat N times
- dense layer and softmax for output

# Transformer decoder

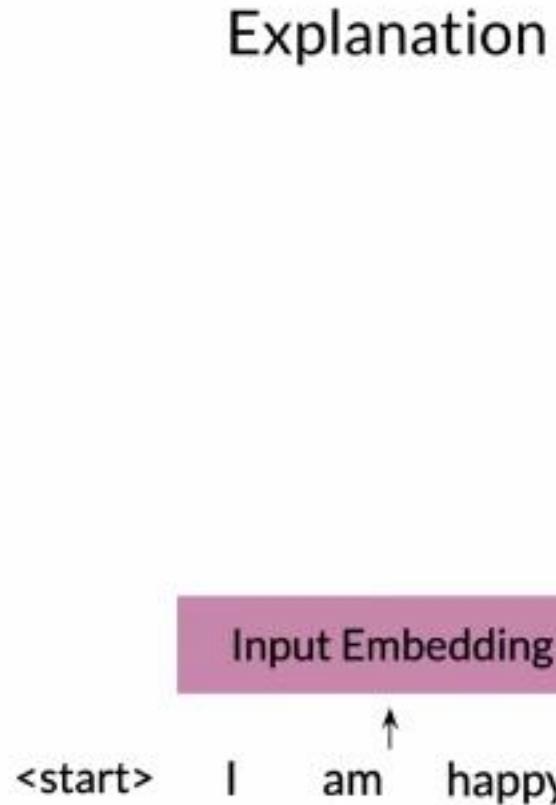
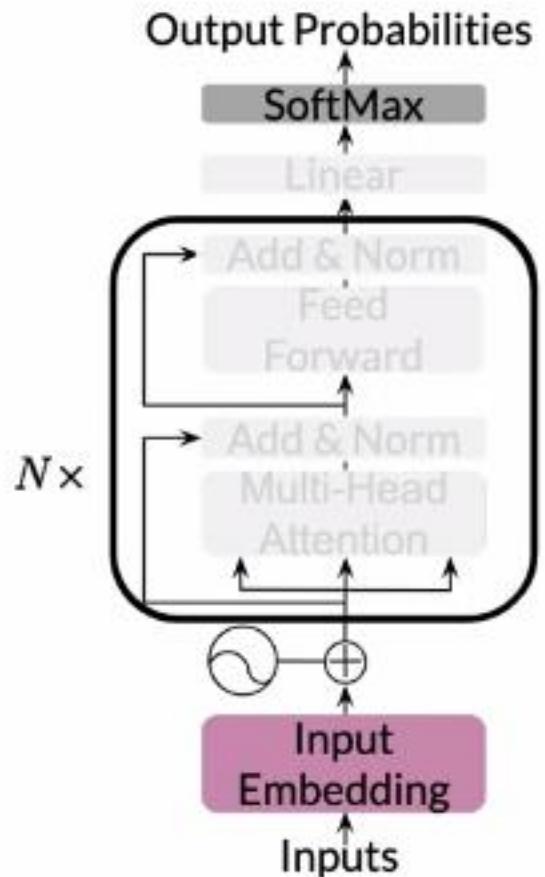


Explanation

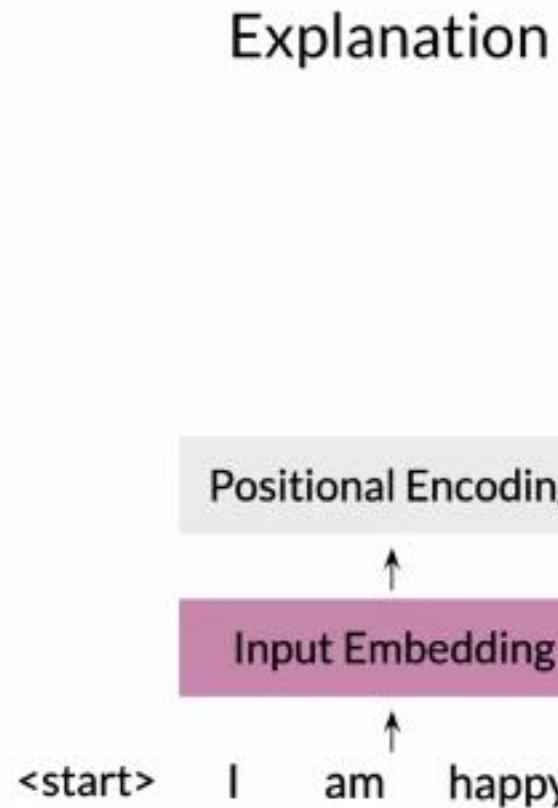
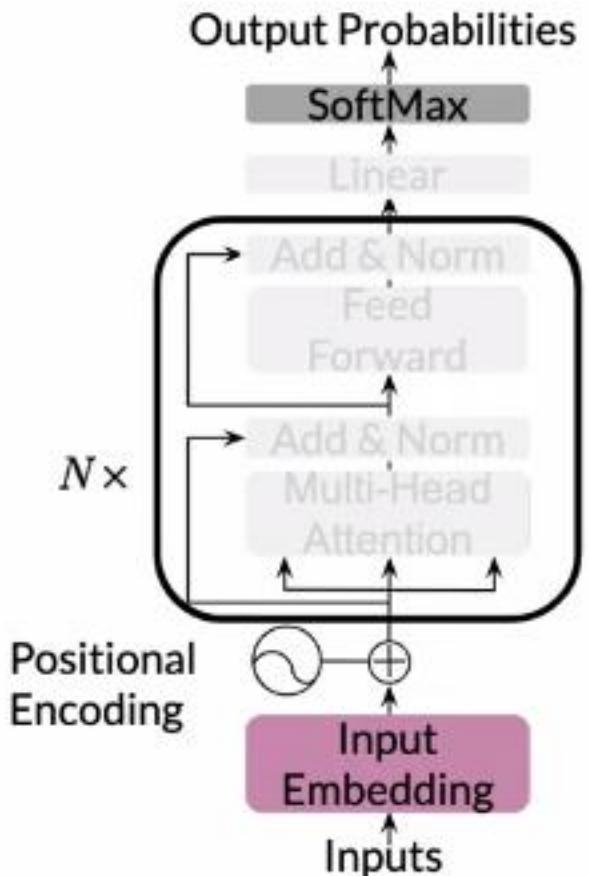
# Transformer decoder



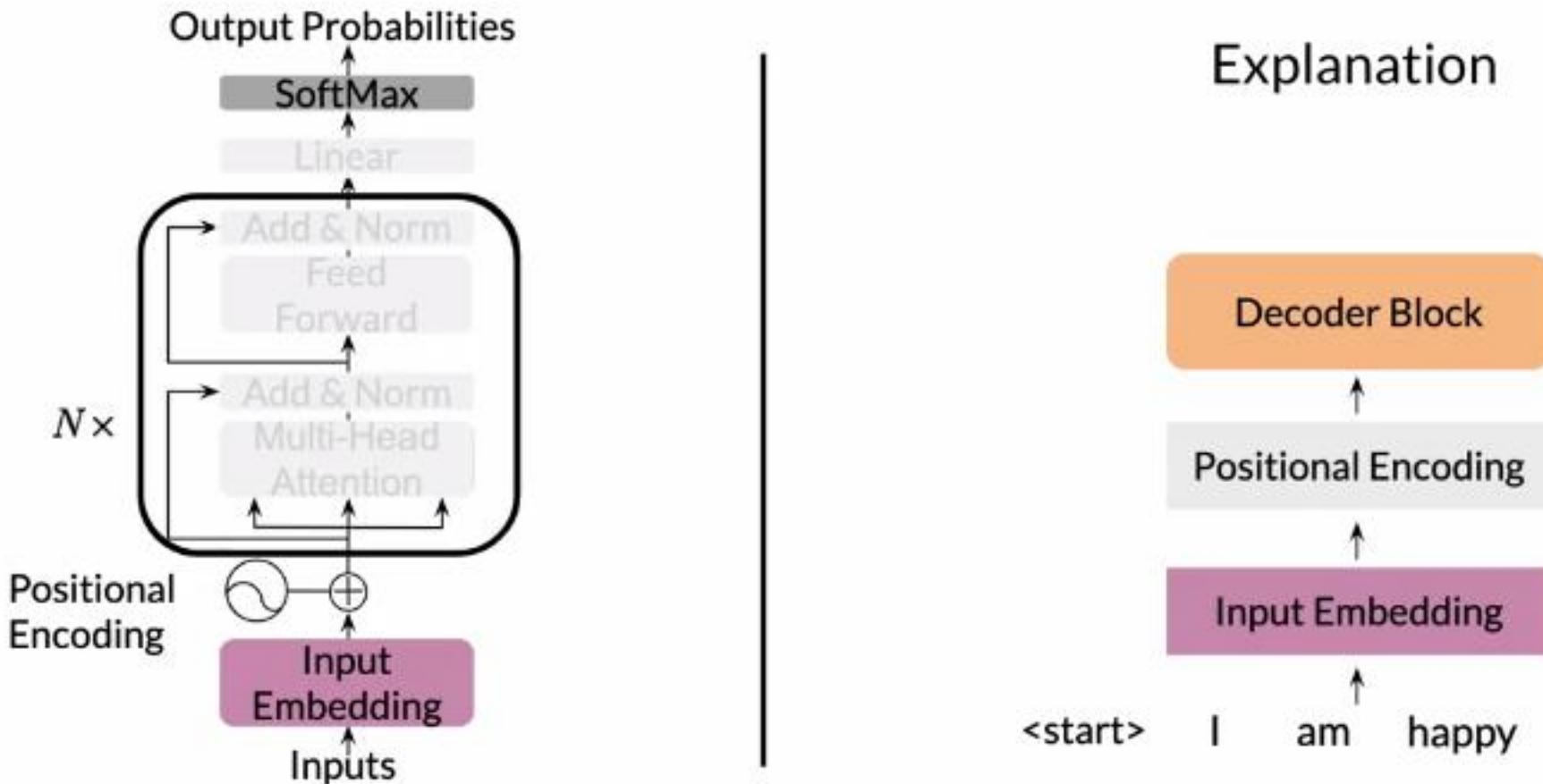
# Transformer decoder



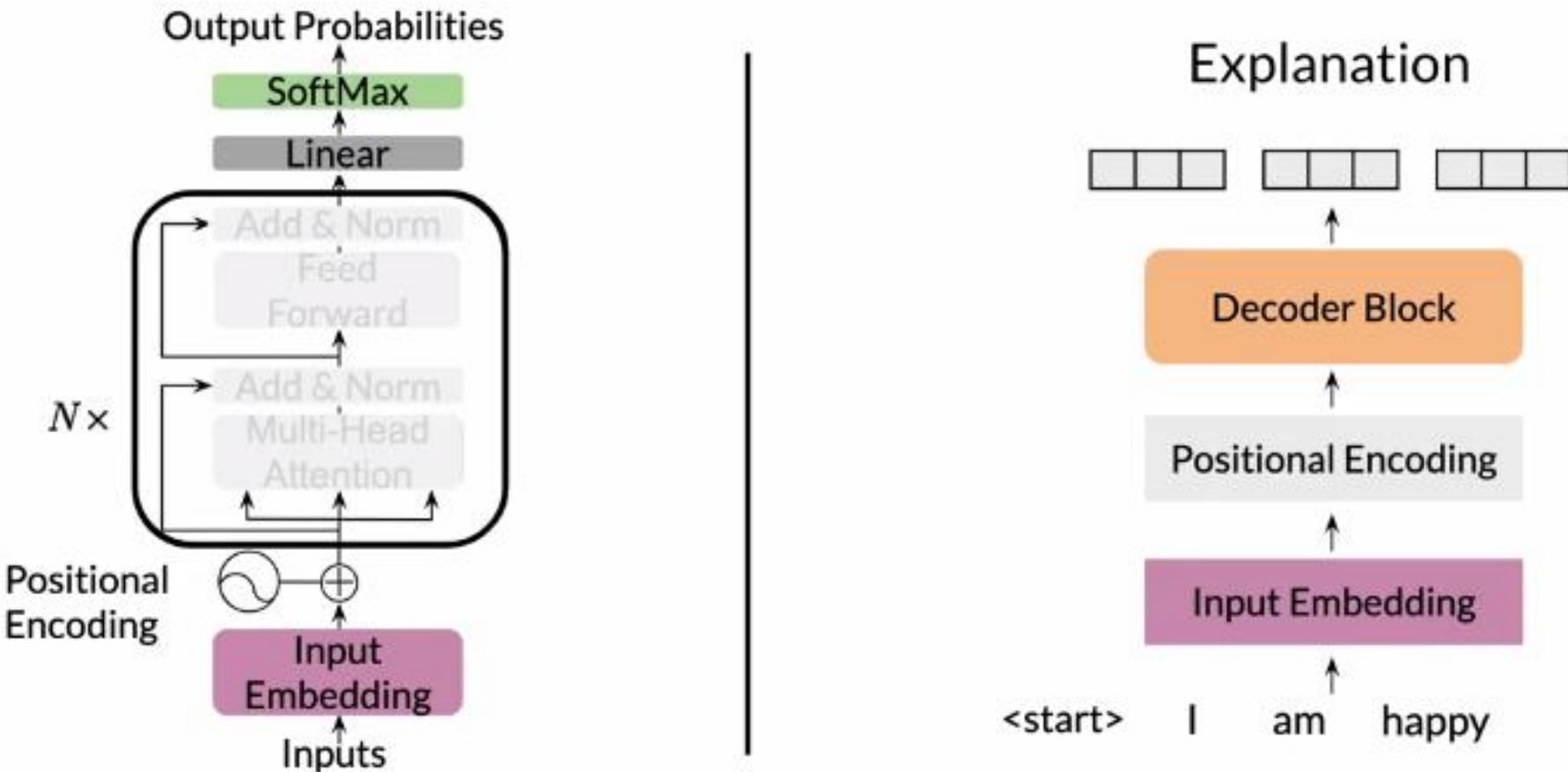
# Transformer decoder



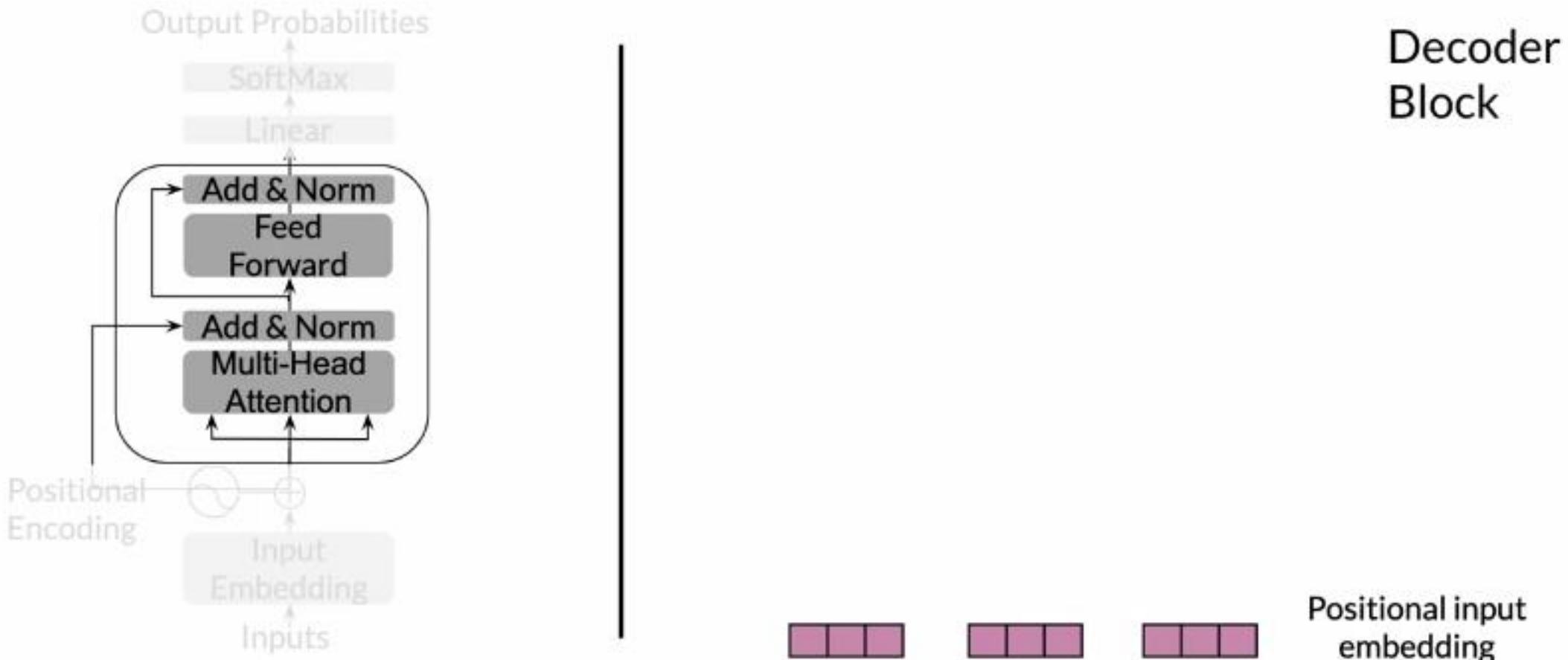
# Transformer decoder



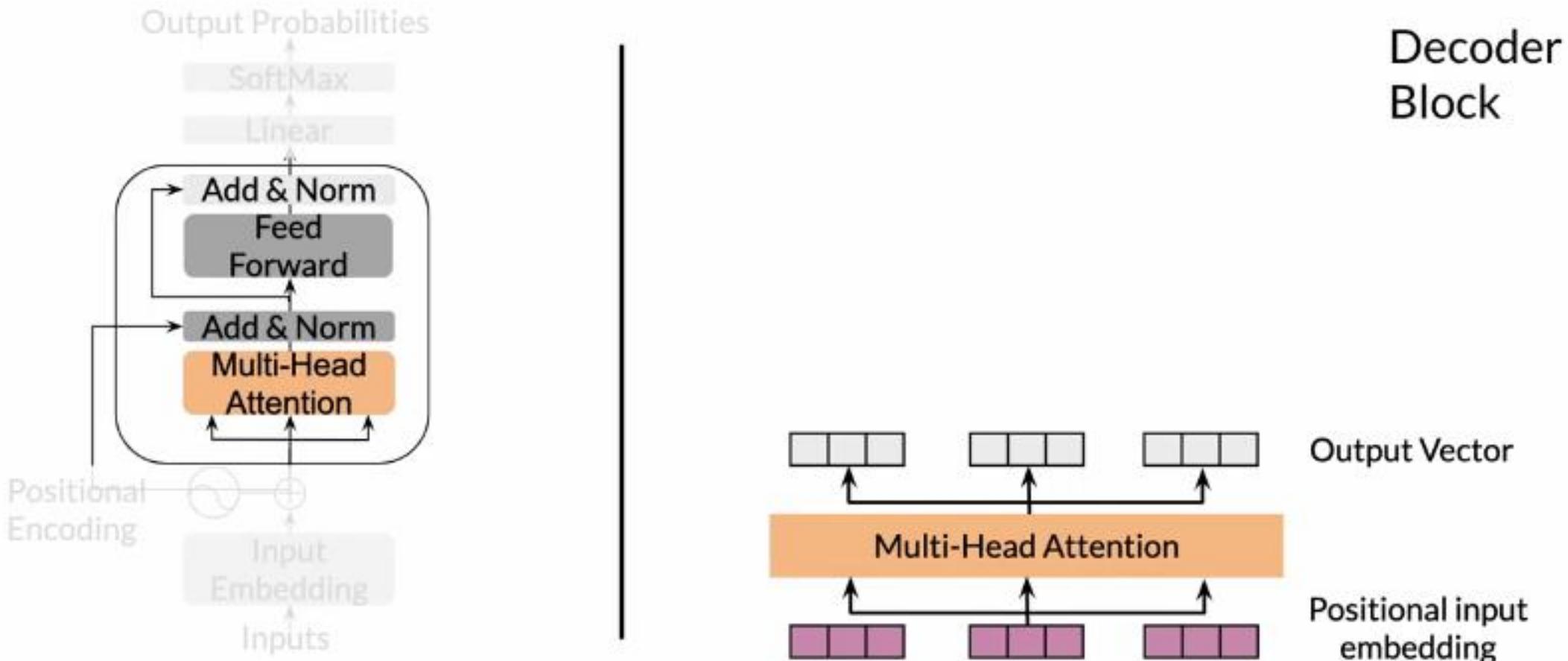
# Transformer decoder



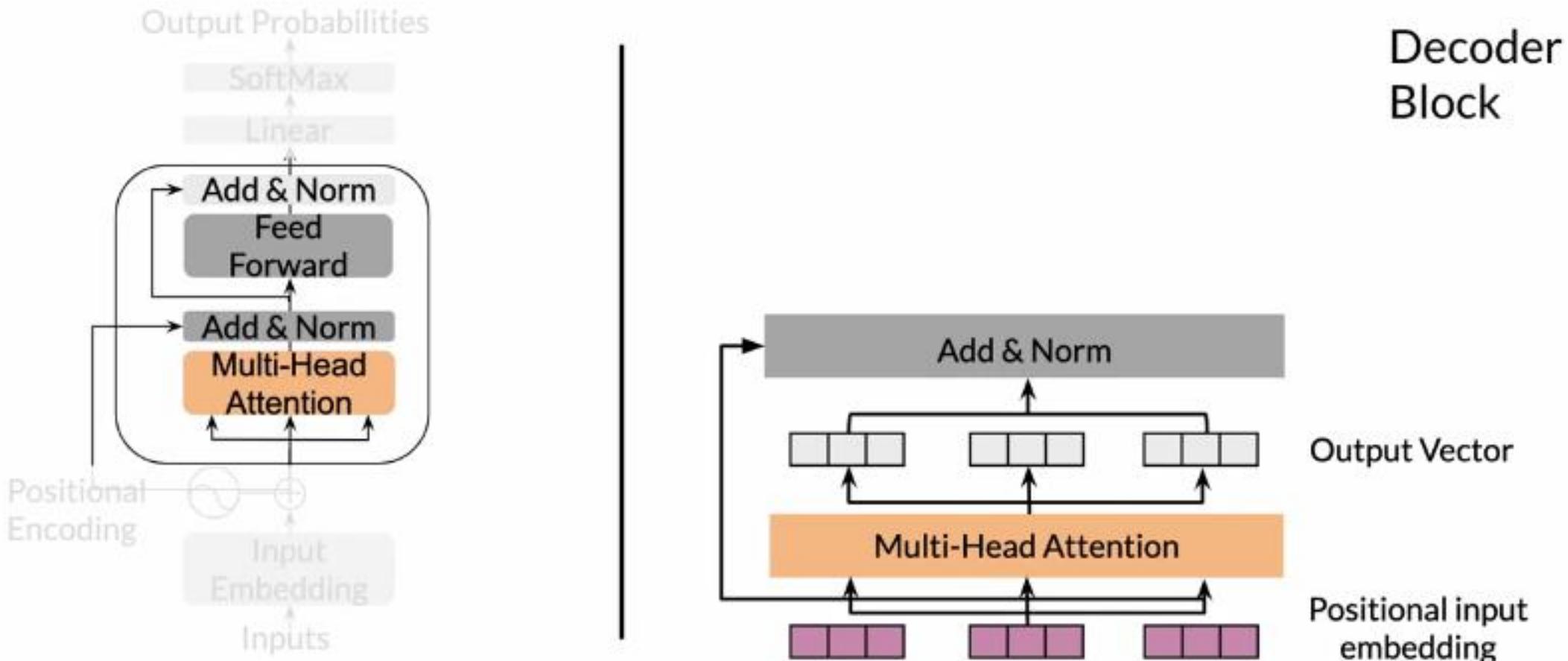
# The Transformer decoder



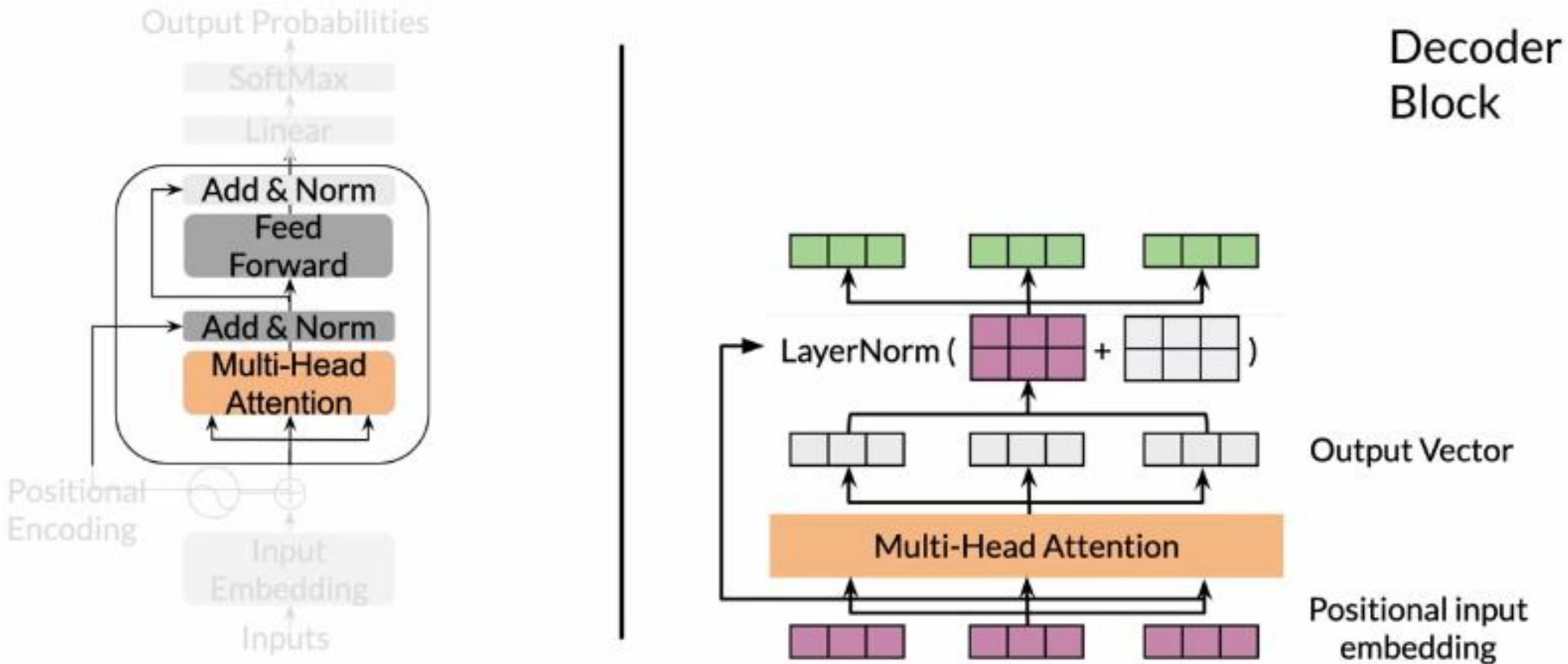
# The Transformer decoder



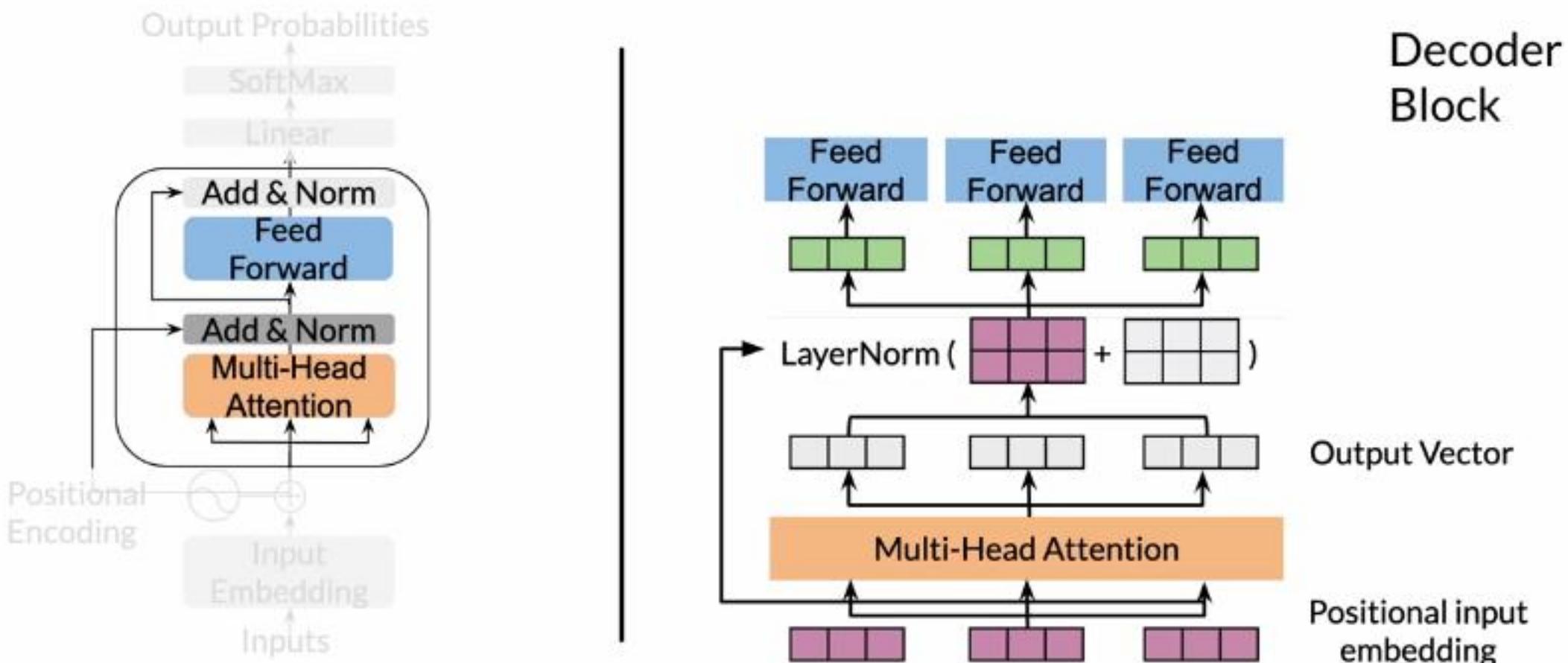
# The Transformer decoder



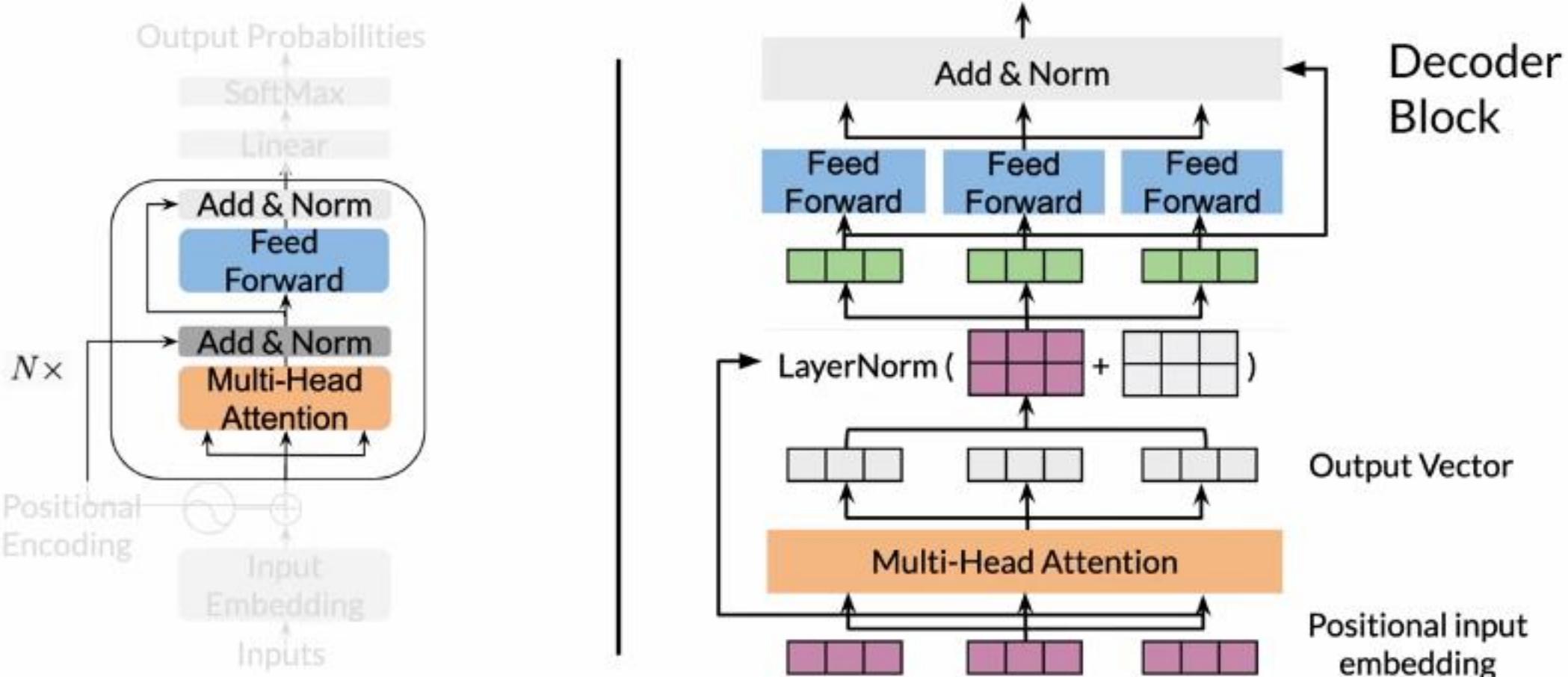
# The Transformer decoder



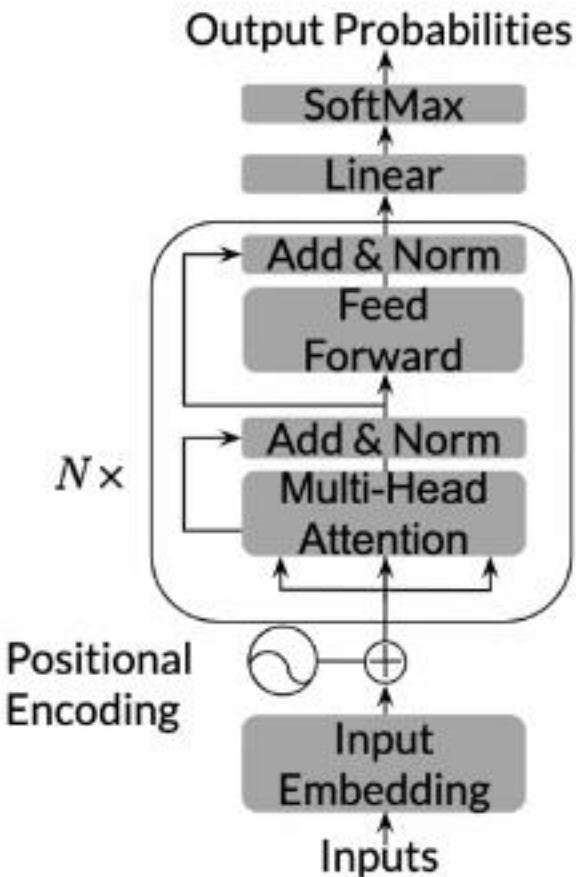
# The Transformer decoder



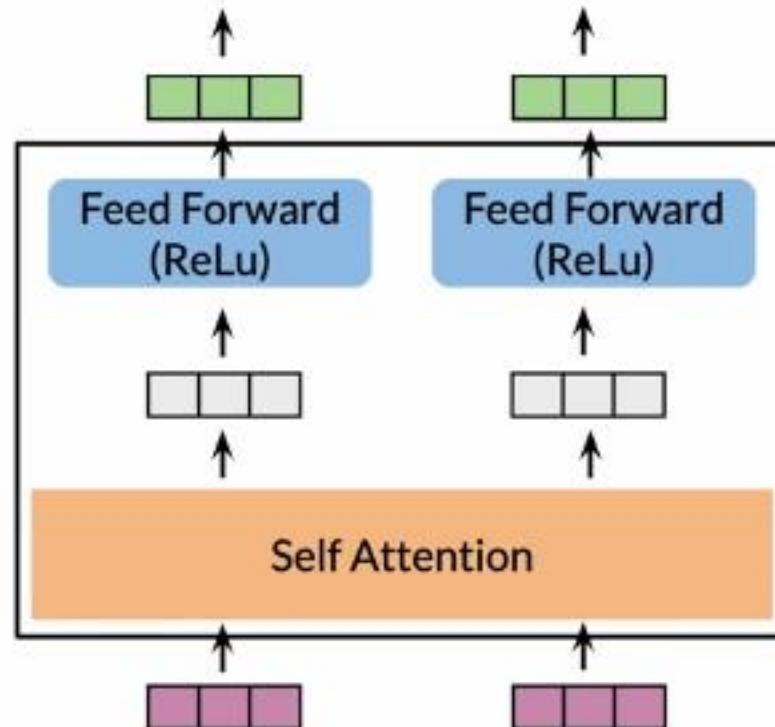
# The Transformer decoder



# The Transformer decoder



Feed forward layer



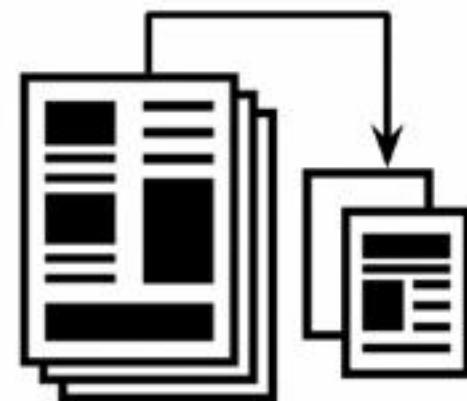
# Summary

- Transformer decoder mainly consists of three layers
- Decoder and feed-forward blocks are the core of this model code
- It also includes a module to calculate the cross-entropy loss

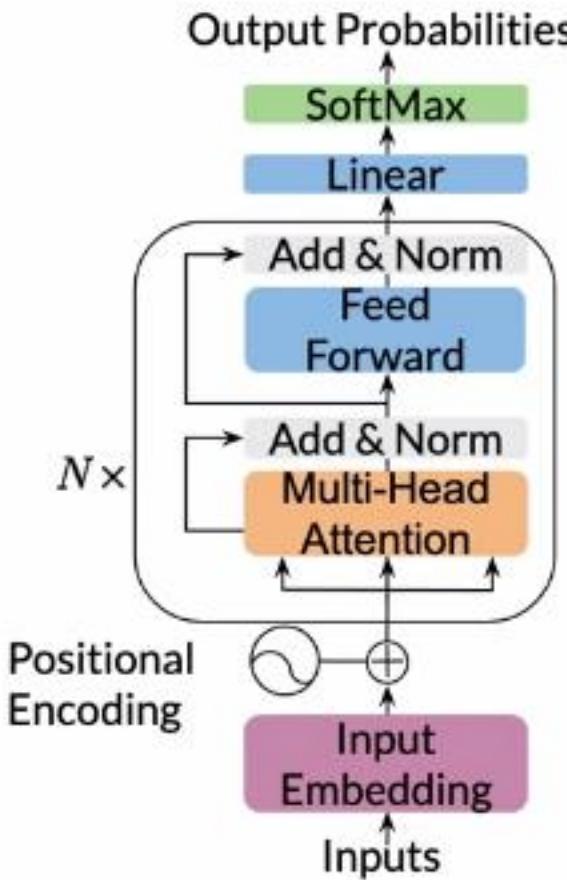


# Outline

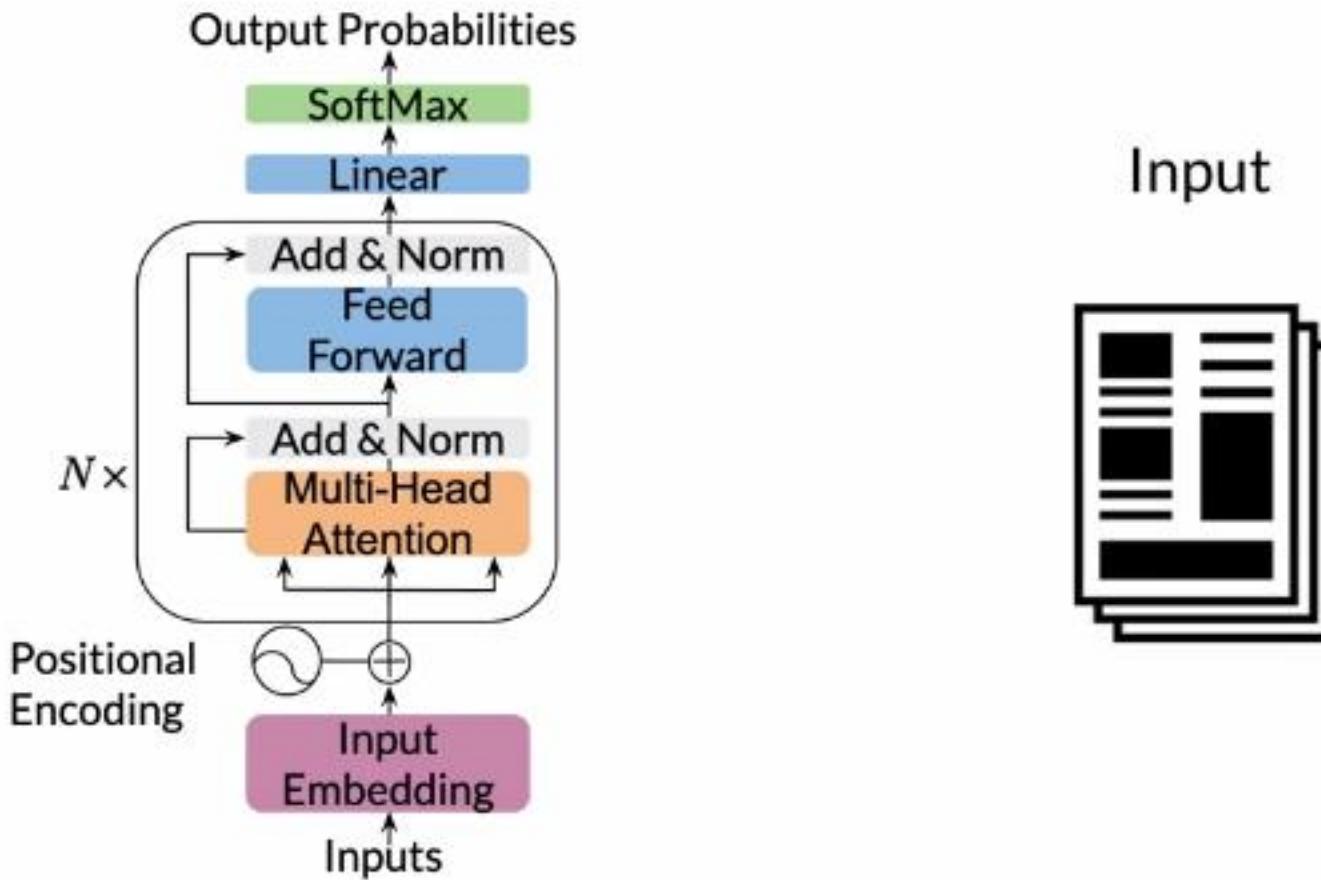
- Overview of Transformer summarizer
- Technical details for data processing
- Inference with a Language Model



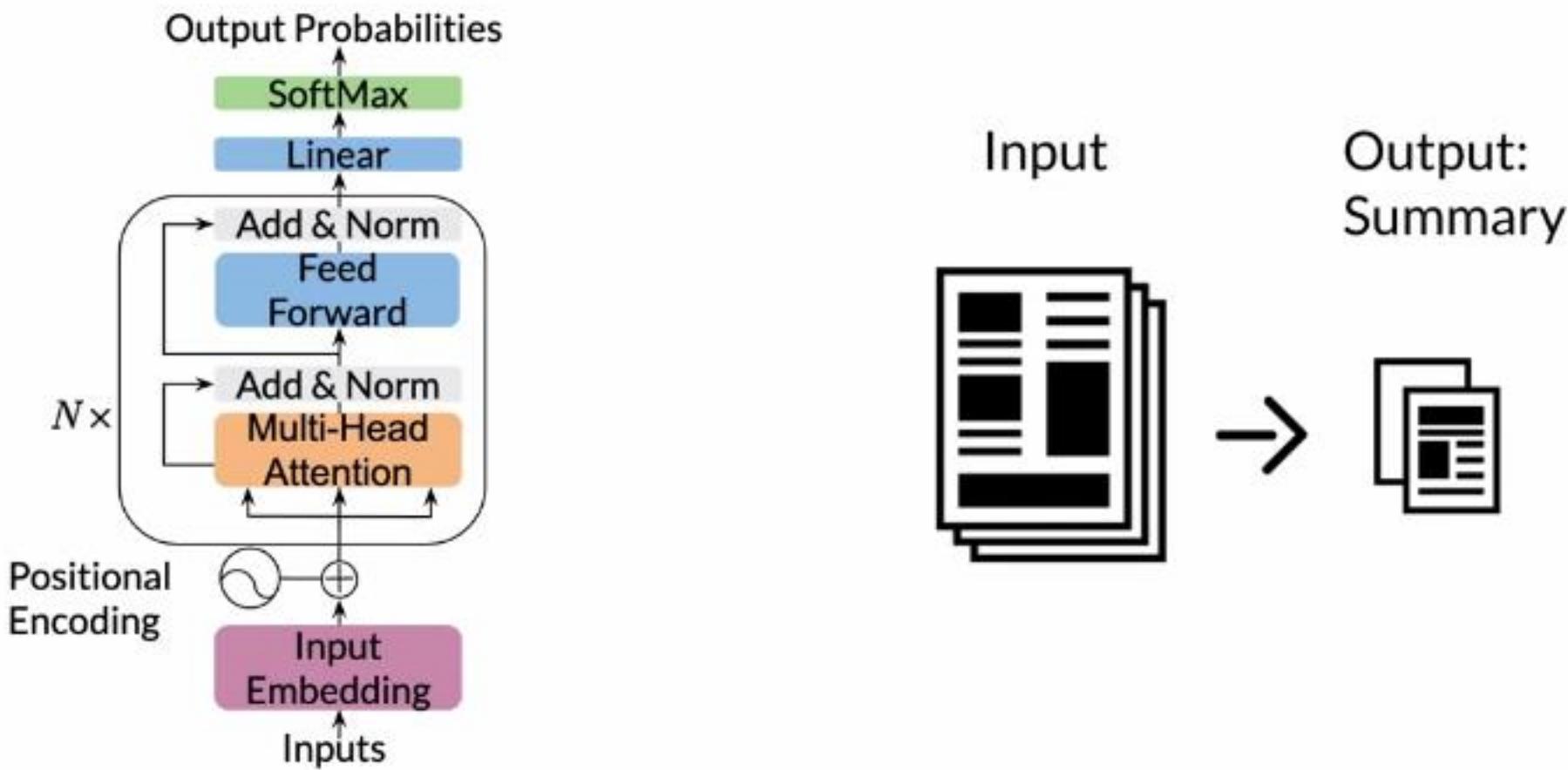
# Transformer for summarization



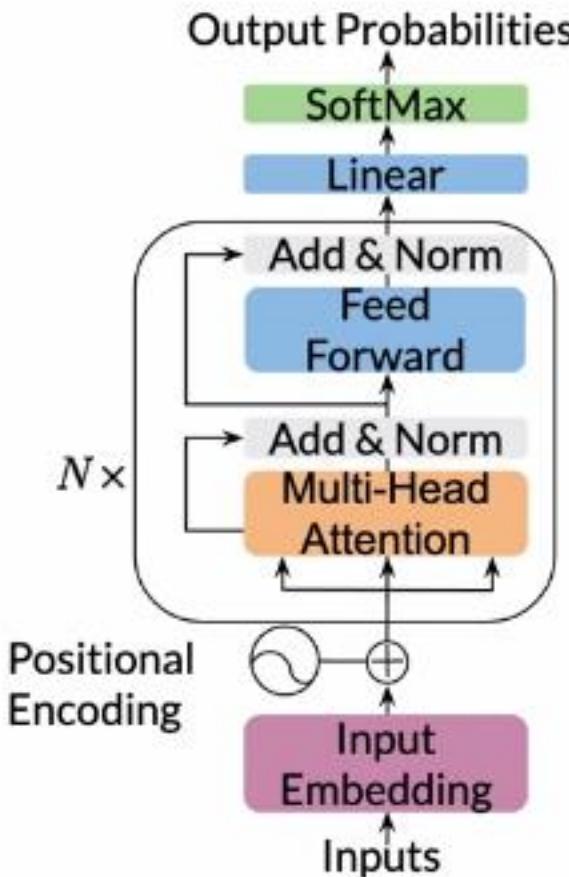
# Transformer for summarization



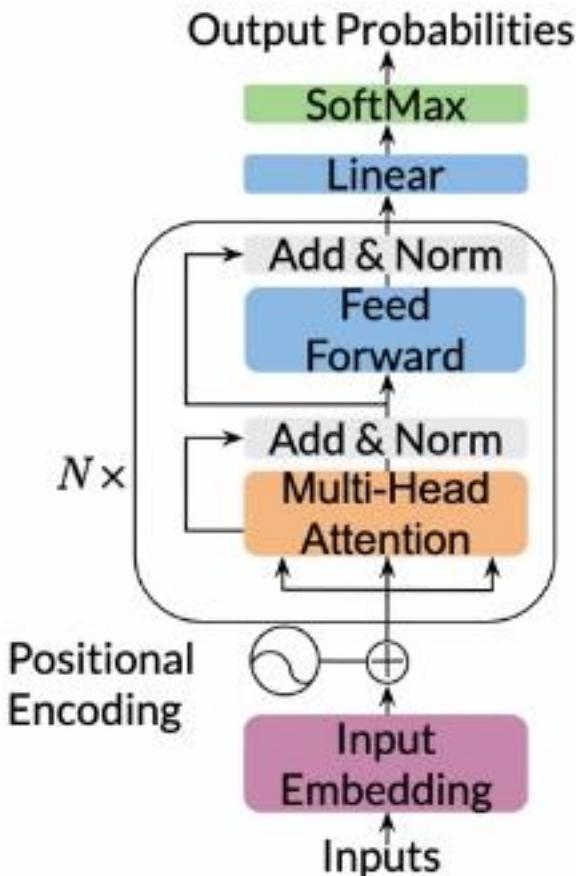
# Transformer for summarization



# Technical details for data processing



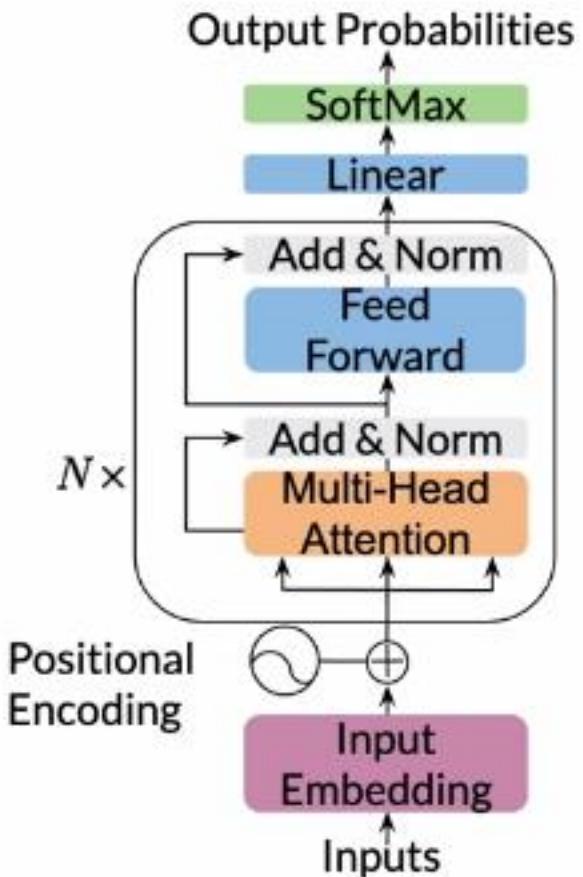
# Technical details for data processing



**Model Input:**

ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...

# Technical details for data processing



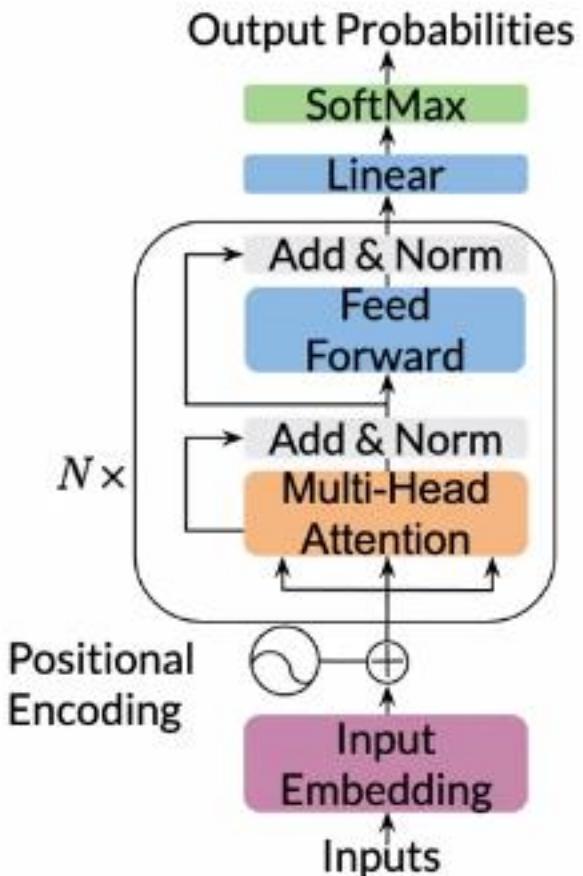
**Model Input:**

ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...

**Tokenized version:**

[2,3,5,2,1,3,4,7,8,2,5,1,2,3,6,2,1,0,0]

# Technical details for data processing



**Model Input:**

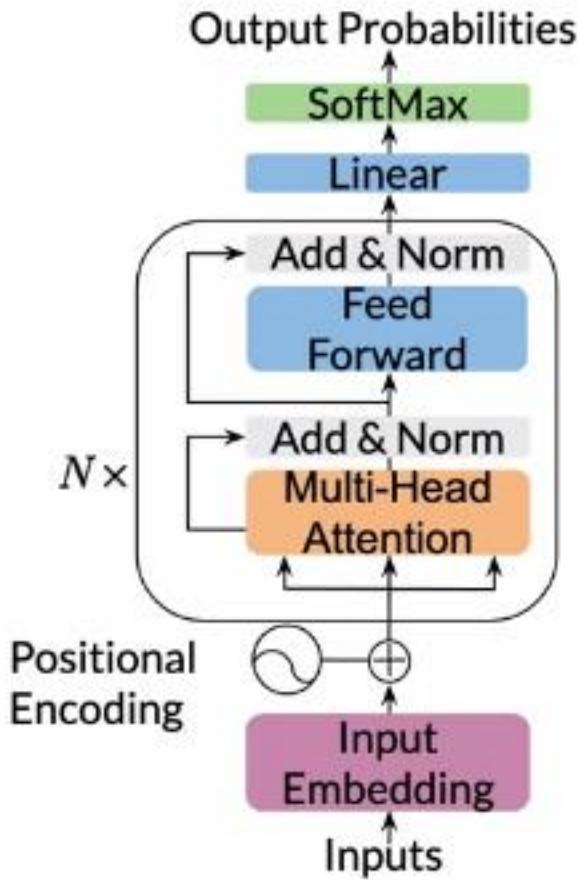
ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...

**Tokenized version:**

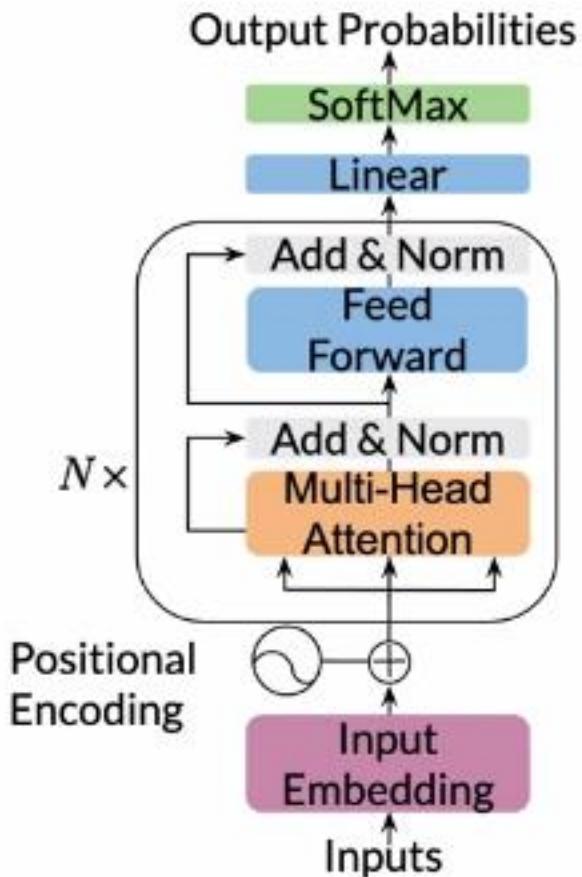
[2,3,5,2,1,3,4,7,8,2,5,1,2,3,6,2,1,0,0]

Loss weights: 0s until the first <EOS> and then 1 on the start of the summary.

# Cost function



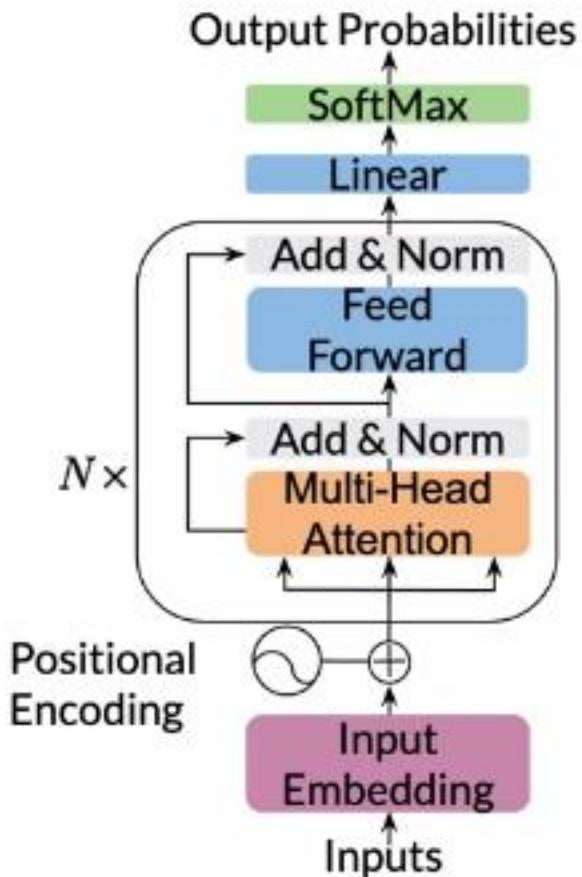
# Cost function



$$J = -\frac{1}{m} \sum_j^m \sum_i^K y_j^i \log \hat{y}_j^i$$



# Cost function



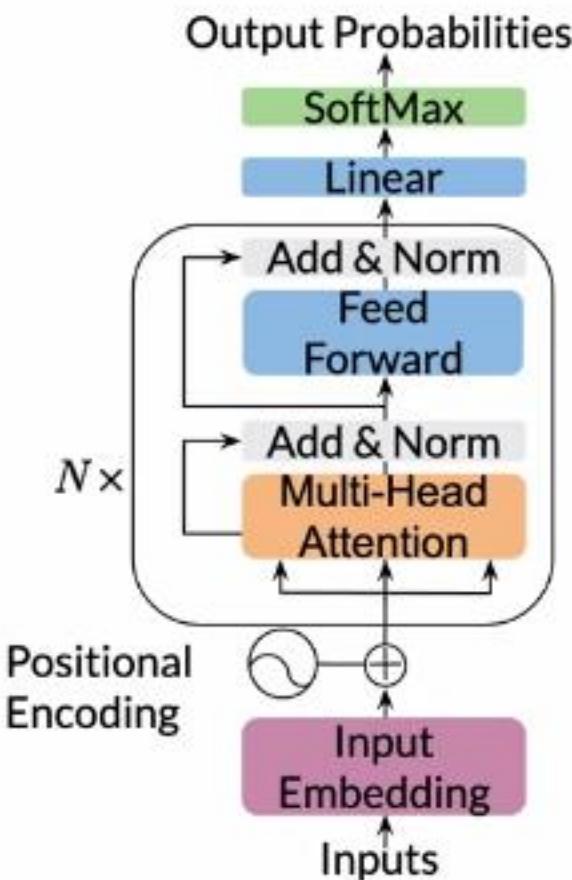
$$J = -\frac{1}{m} \sum_j^m \sum_i^K y_j^i \log \hat{y}_j^i$$

$j$  : over summary

$i$  : batch elements



# Cost function



## Cross entropy loss

$$J = -\frac{1}{m} \sum_j^m \sum_i^K y_j^i \log \hat{y}_j^i$$

$j$  : over summary

$i$  : batch elements



# Inference with a Language Model

# Inference with a Language Model

## Model input:

[Article] <EOS> [Summary] <EOS>

# Inference with a Language Model

## Model input:

[Article] <EOS> [Summary] <EOS>

## Inference:

# Inference with a Language Model

## Model input:

[Article] <EOS> [Summary] <EOS>

## Inference:

- Provide: [Article] <EOS>

# Inference with a Language Model

## Model input:

[Article] <EOS> [Summary] <EOS>

## Inference:

- Provide: [Article] <EOS>
- Generate summary word-by-word
  - until the final <EOS>

# Inference with a Language Model

## Model input:

[Article] <EOS> [Summary] <EOS>

## Inference:

- Provide: [Article] <EOS>
- Generate summary word-by-word
  - until the final <EOS>
- Pick the next word by random sampling
  - each time you get a different summary!

# Summary

- For summarization, a weighted loss function is optimized
- Transformer Decoder summarizes predicting the next word using
- The transformer uses tokenized versions of the input

