Transformers (cont'd)

CS 6804: Frontier AI Systems

Spring 2026

https://tuvllms.github.io/ai-seminar-spring-2026/

Tu Vu

VIRGINIA TECH

Logistics

- Office hours starting this Friday 2:45 – 3:45 PM
    - both in-person (D&DS Rm 374) and via Zoom (link available on Piazza/Discord)

- Student presentations
    - Search for teammates on Piazza/Discord or reach out to me at cs6804instructors@gmail.com
    - Google form for submitting group information available on Piazza/Discord (due EOD this Friday 1/30)

AI news

- Starting from next week

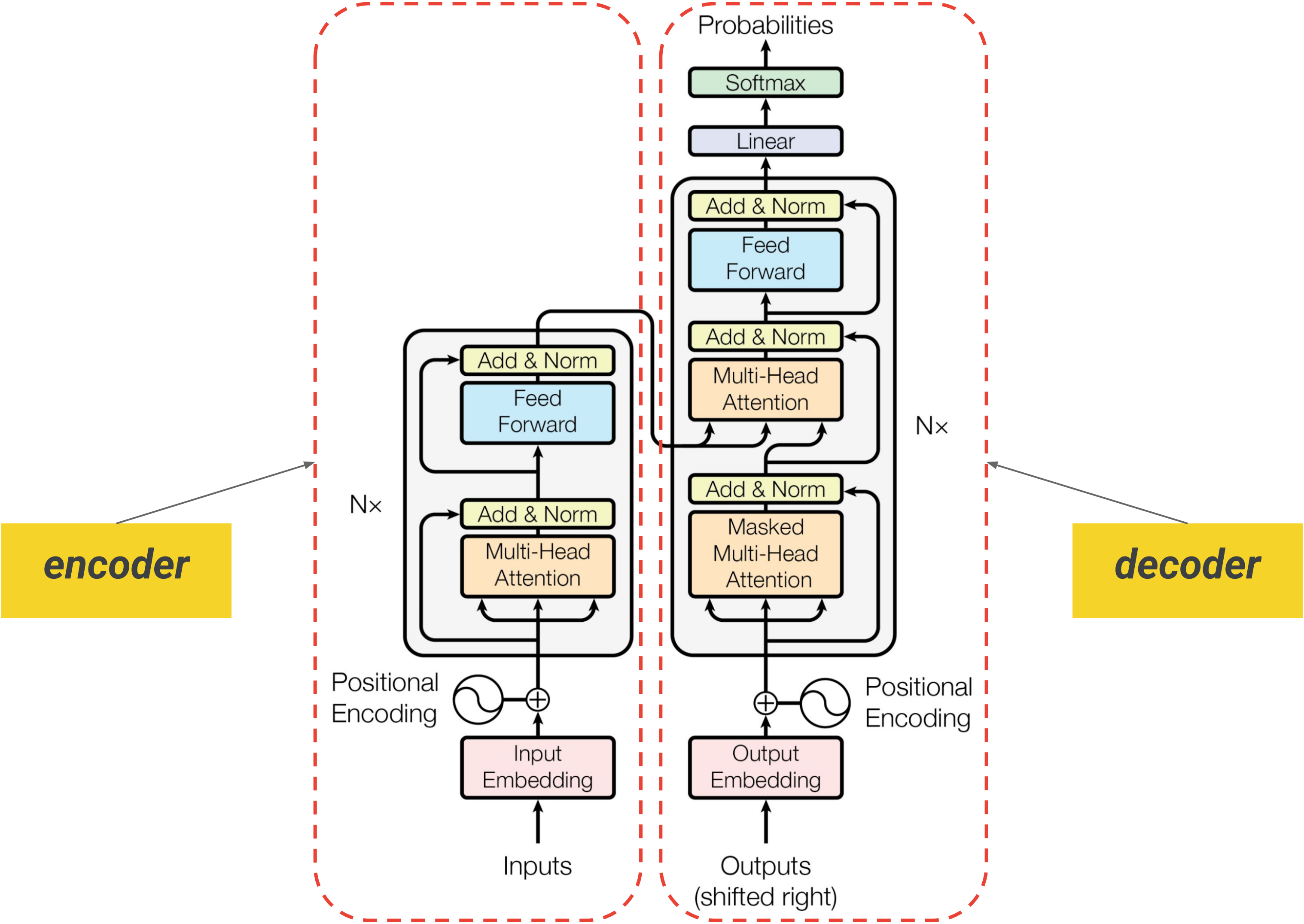
This course

- **Six** (?) weeks of me lecturing, so that we are all roughly on the same page
    - the overall class background wasn't as much as I thought

- Rest of semester: student presentations and discussions of assigned papers

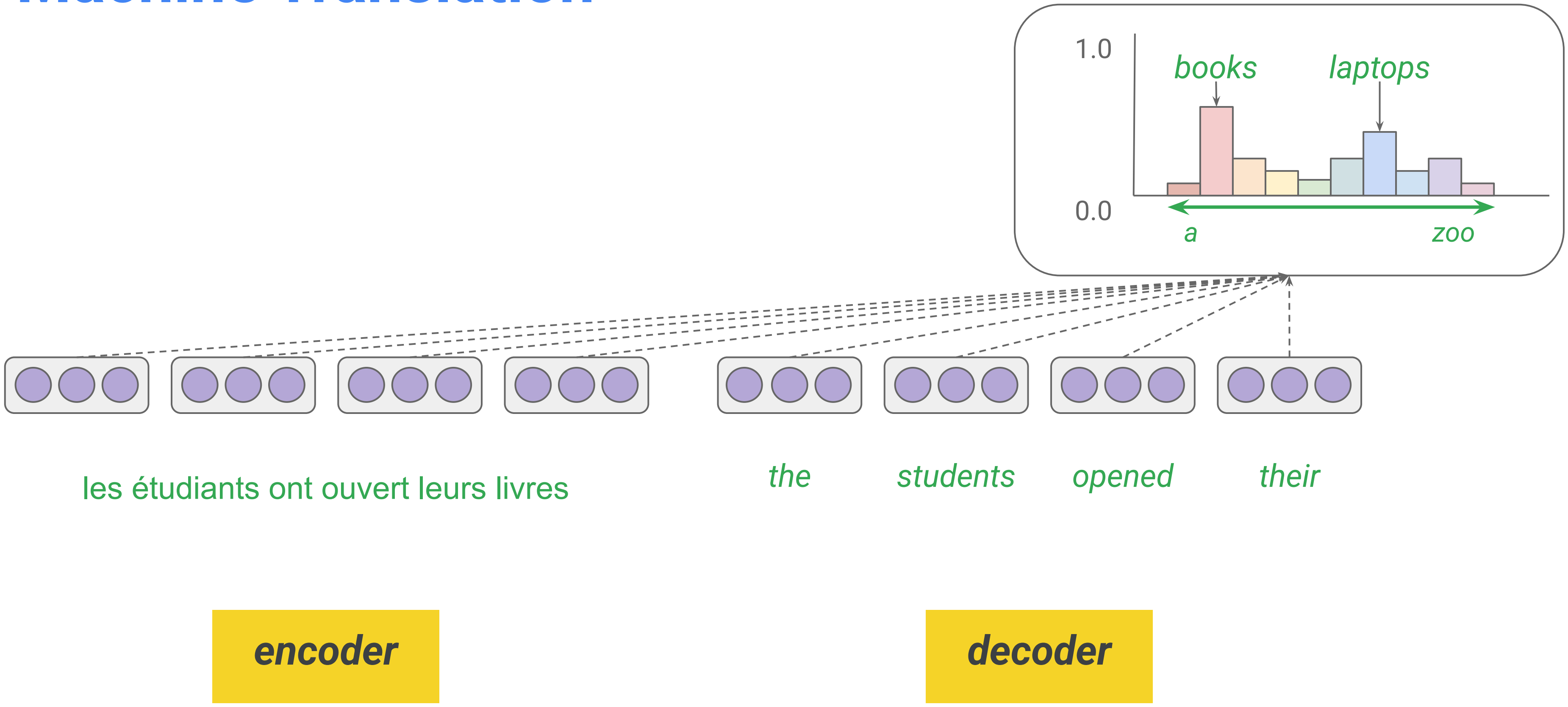
Different Transformers architectures

- Encoder-only
  - BERT
- Encoder-decoder
  - T5
- Decoder-only
  - GPT



Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
encoder
decoder

Machine Translation



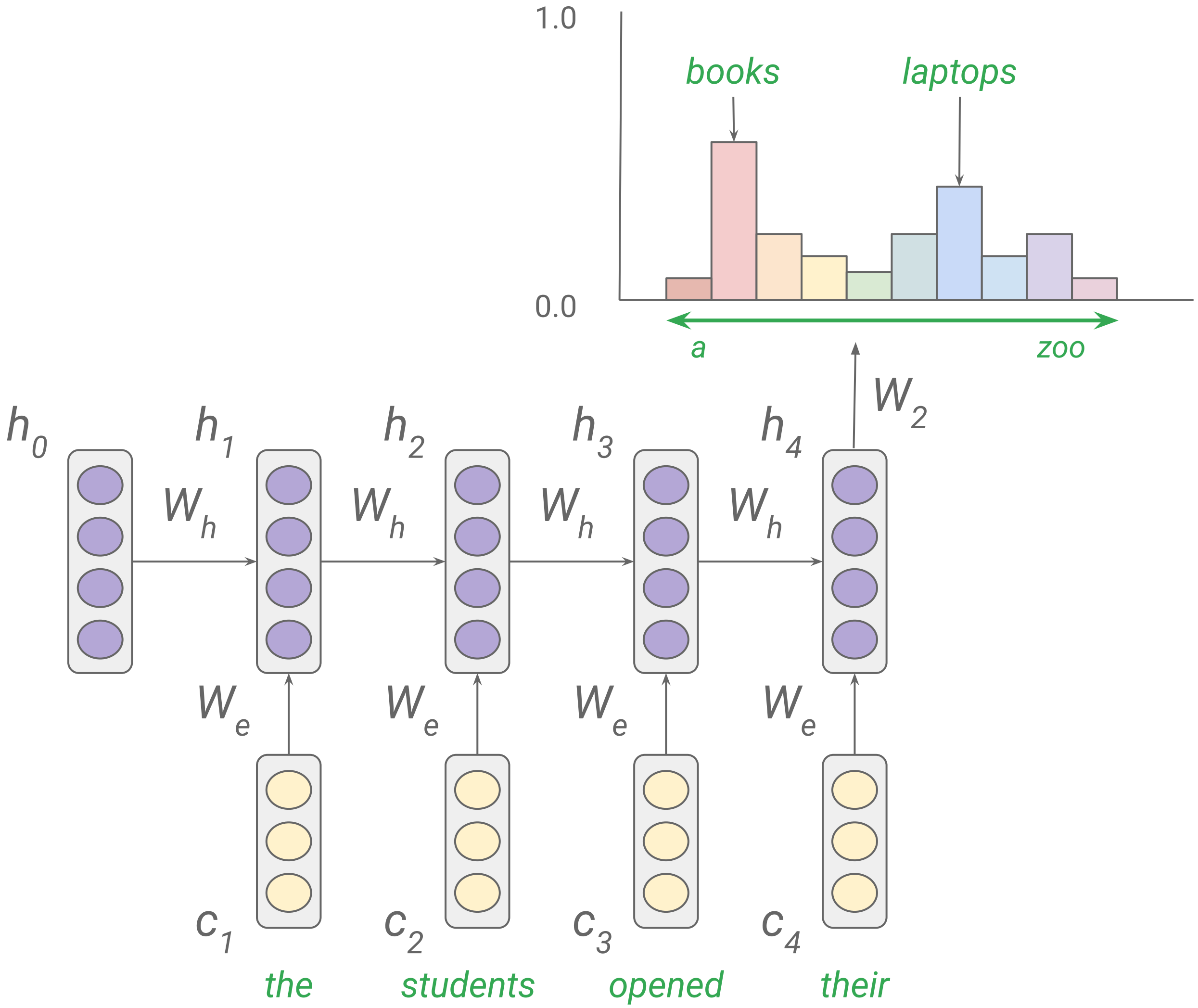
Recurrent neural networks (RNNs)

hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c^t)$$

output distribution

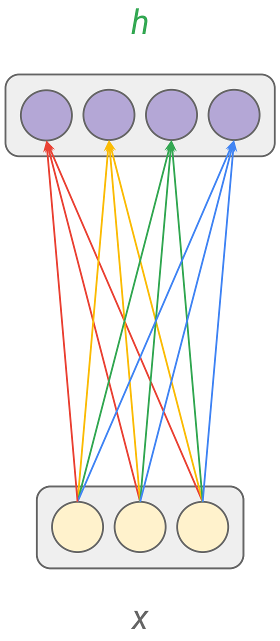
$$\hat{y} = softmax(W_2 h^{(n-1)})$$



Problems with RNNs

- Bottleneck representation issue
- Lack of parallelism

Matrix-vector multiplication



linear projection

Matrix A (dimensions 4×3):

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Vector x (dimensions 3×1):

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Resulting vector b (dimensions 4×1):

$$b = A \cdot x = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 \end{bmatrix}$$

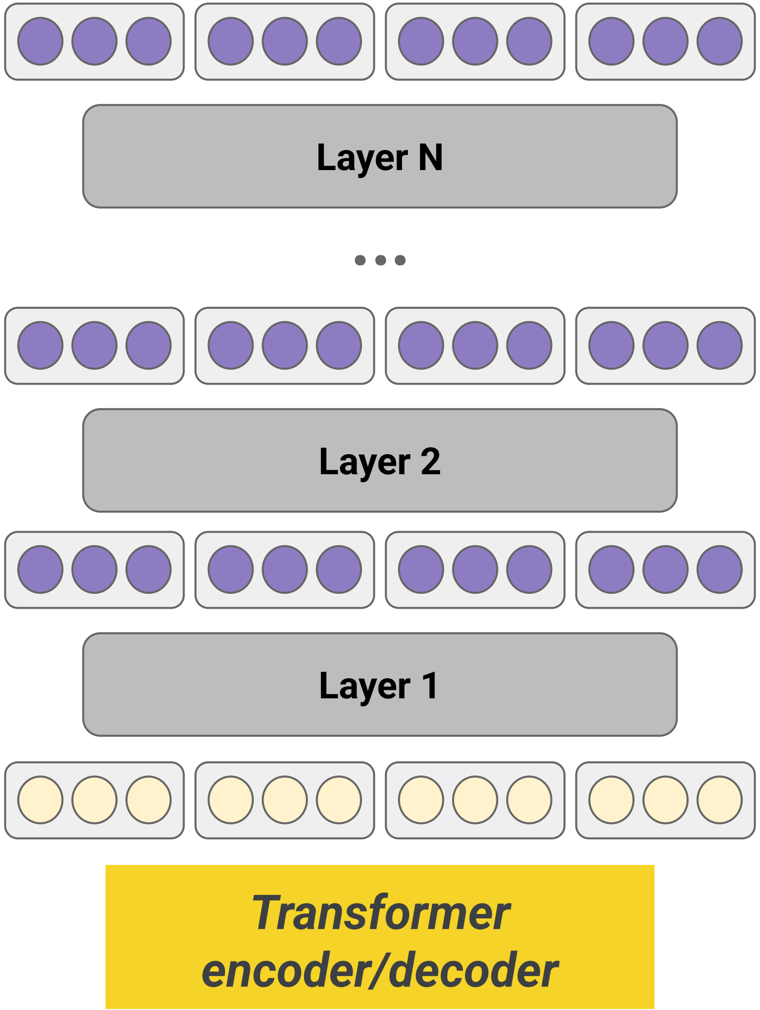
Softmax function

For a vector $y = [y_1, y_2, \ldots, y_V]$ of dimension V, the softmax transformation is calculated as:

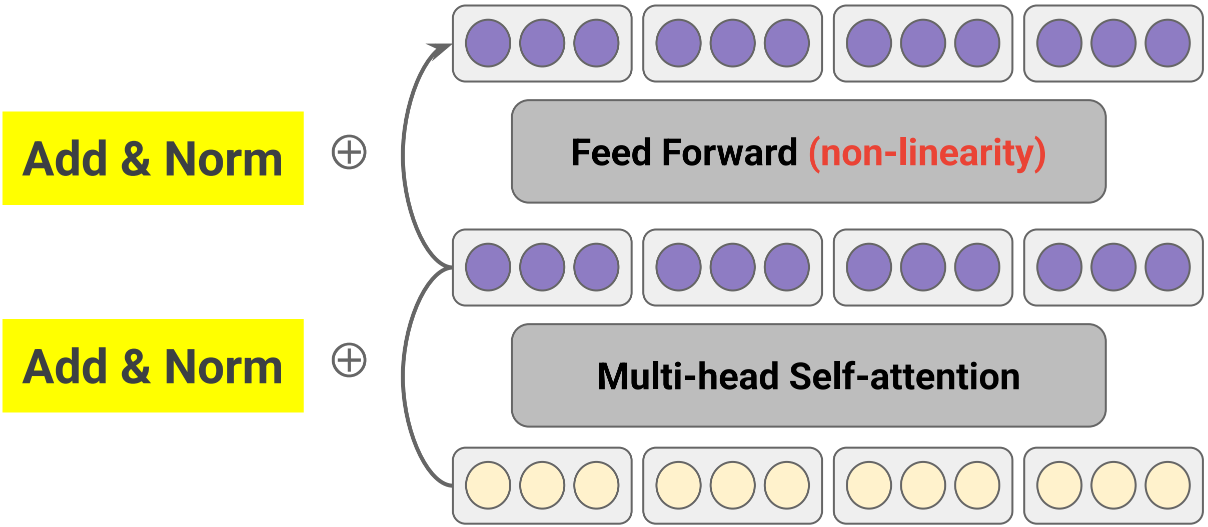
$$\text{softmax}(y) = \left[\frac{e^{y_1}}{\sum e^y}, \frac{e^{y_2}}{\sum e^y}, \ldots, \frac{e^{y_V}}{\sum e^y}\right]$$

where $\sum e^y = e^{y_1} + e^{y_2} + \cdots + e^{y_V}$.

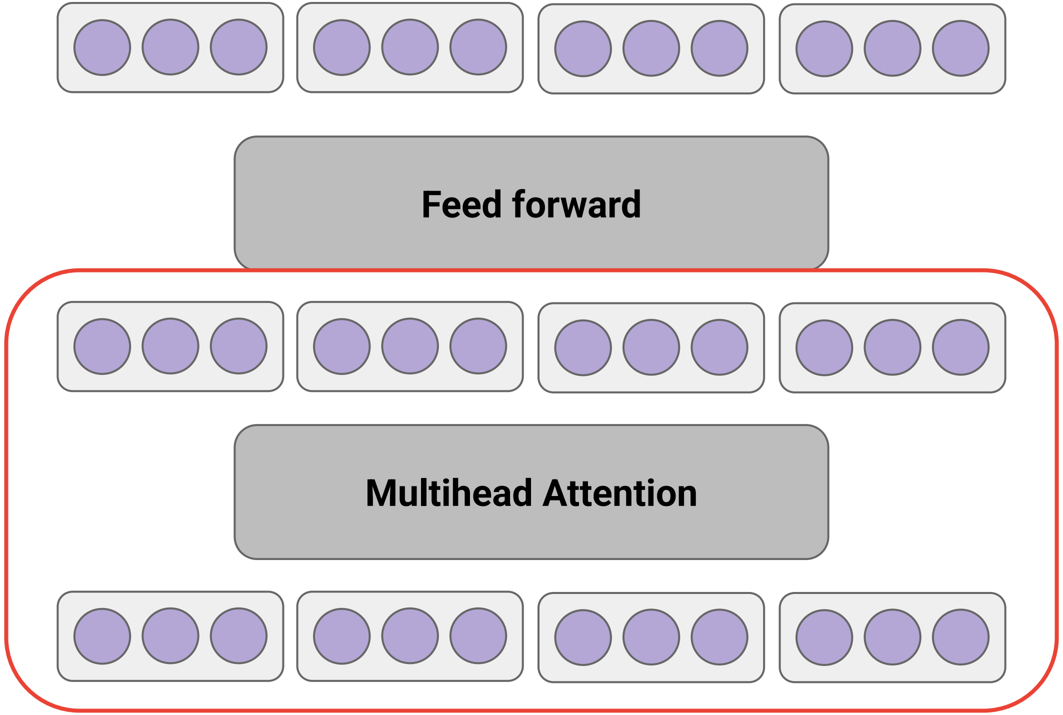
N layers



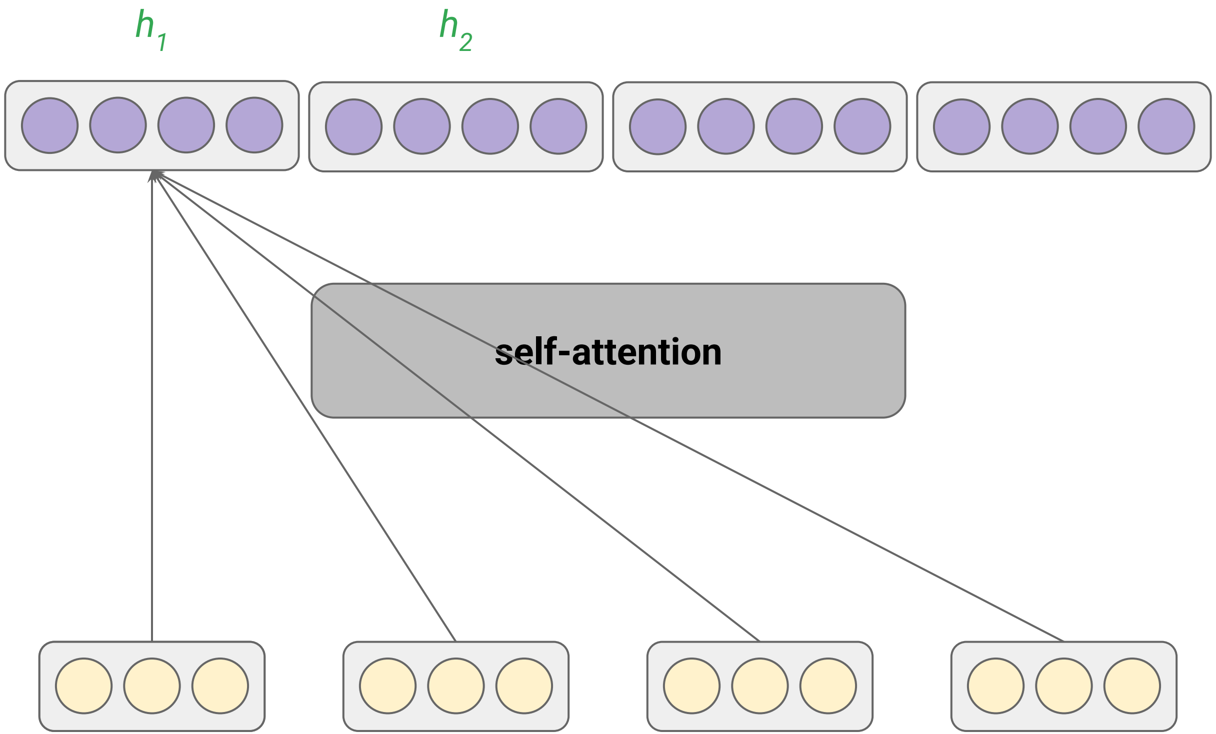
Transformer block (one layer)



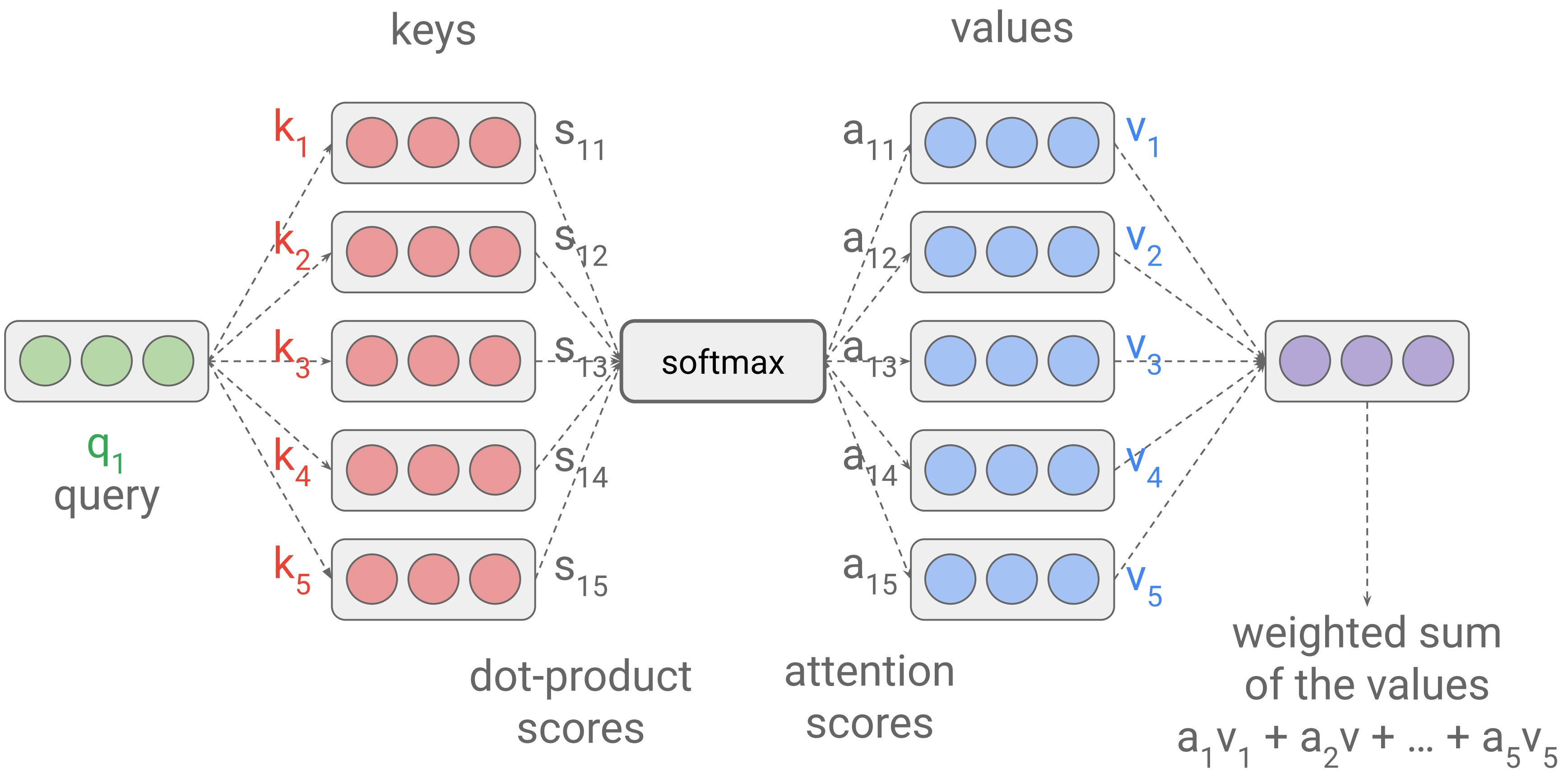
Transformer block



Self-attention

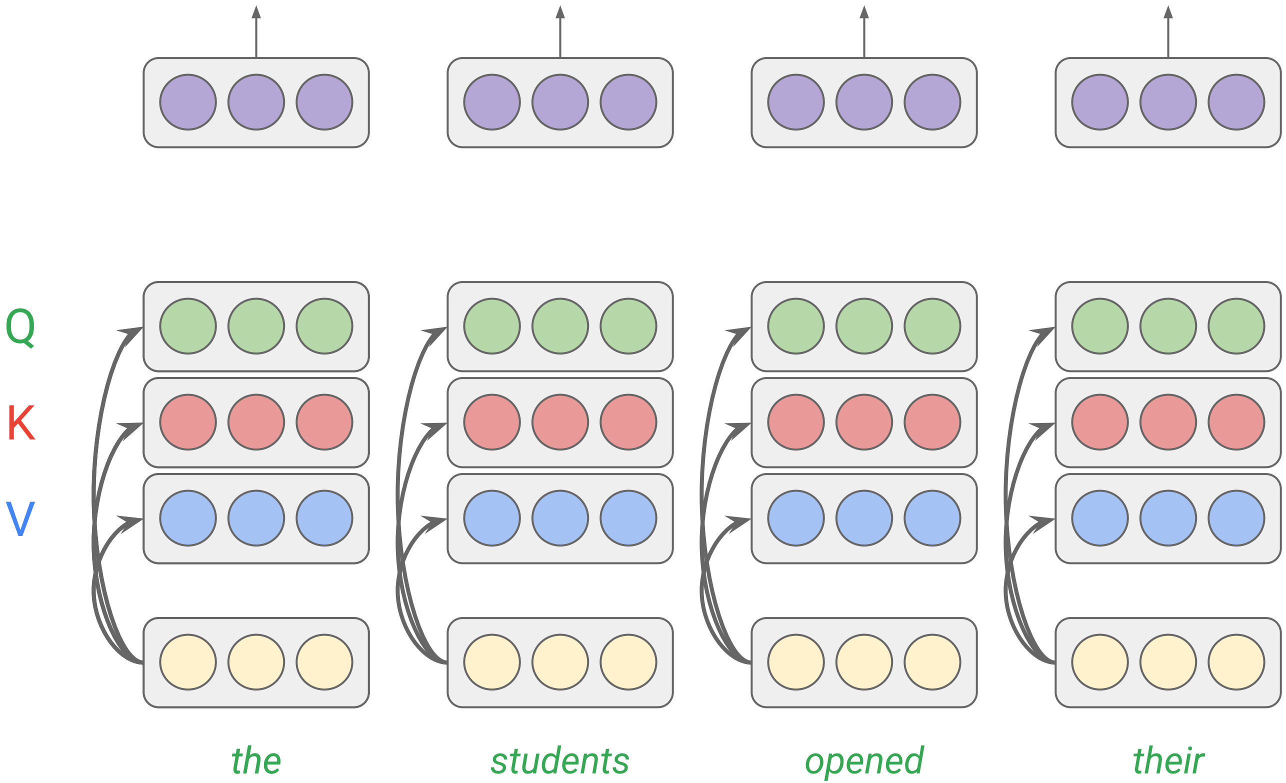


Attention



keys
values
k_1
k_2
k_3
k_4
k_5
s_{11}
s_{12}
s_{13}
s_{14}
s_{15}
softmax
a_{11}
a_{12}
a_{13}
a_{14}
a_{15}
v_1
v_2
v_3
v_4
v_5
q_1
query
dot-product scores
attention scores
weighted sum of the values
$a_1v_1 + a_2v + \ldots + a_5v_5$

Attention (cont'd)



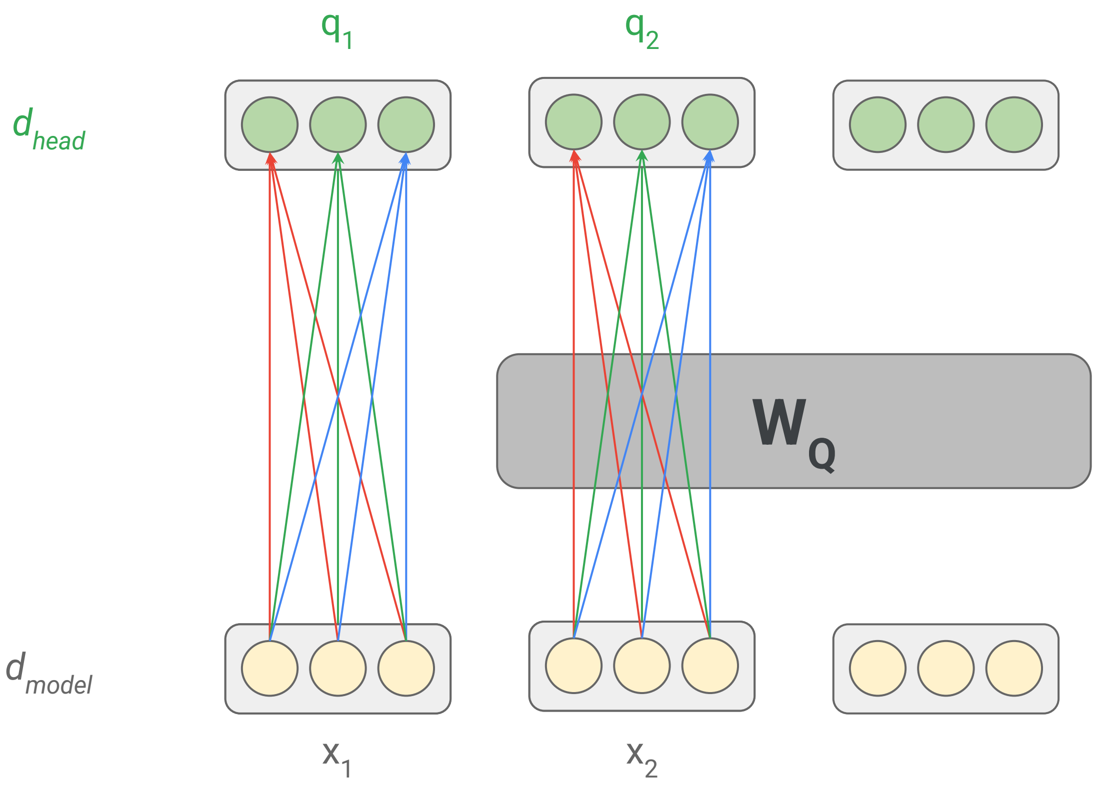
$Q = X \cdot W_Q$

$K = X \cdot W_K$

$V = X \cdot W_V$

linear projections

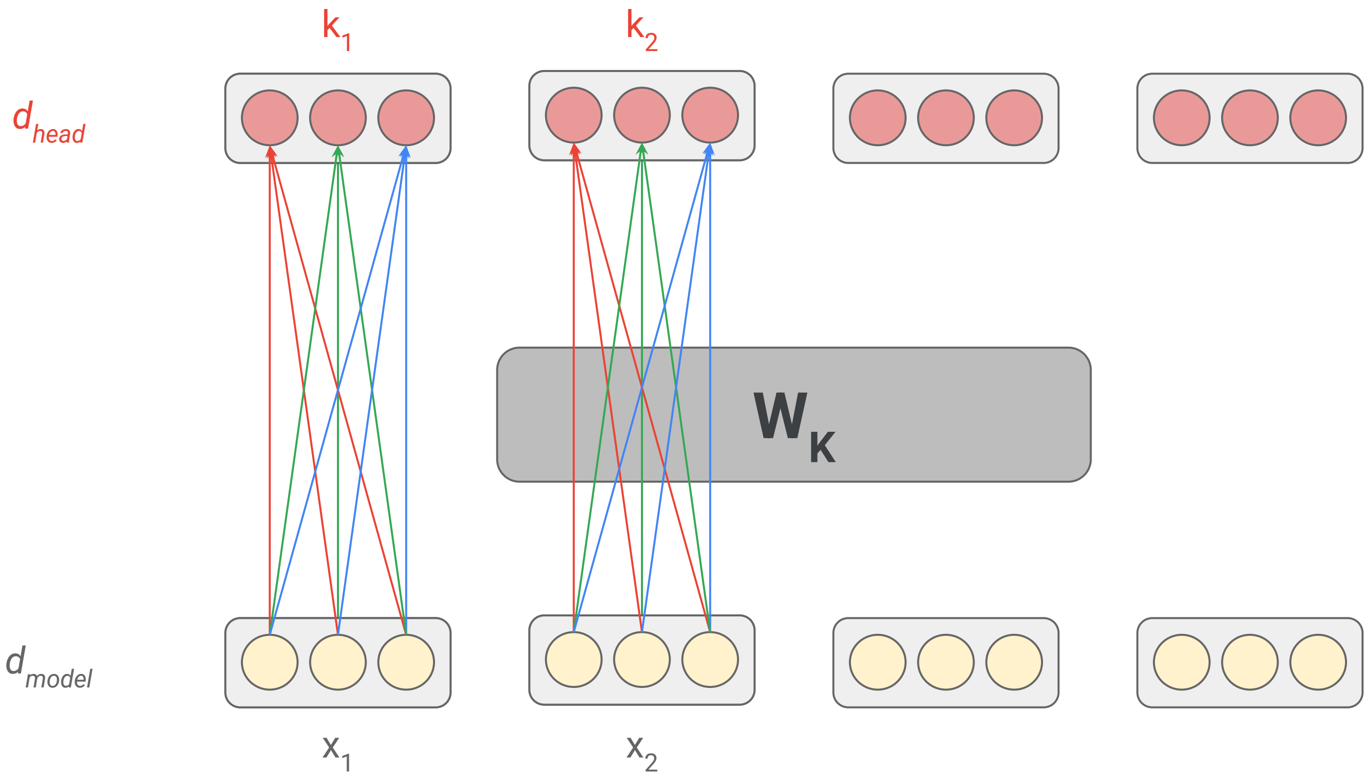
Query vectors



$$Q = X \cdot W_Q$$

linear projections

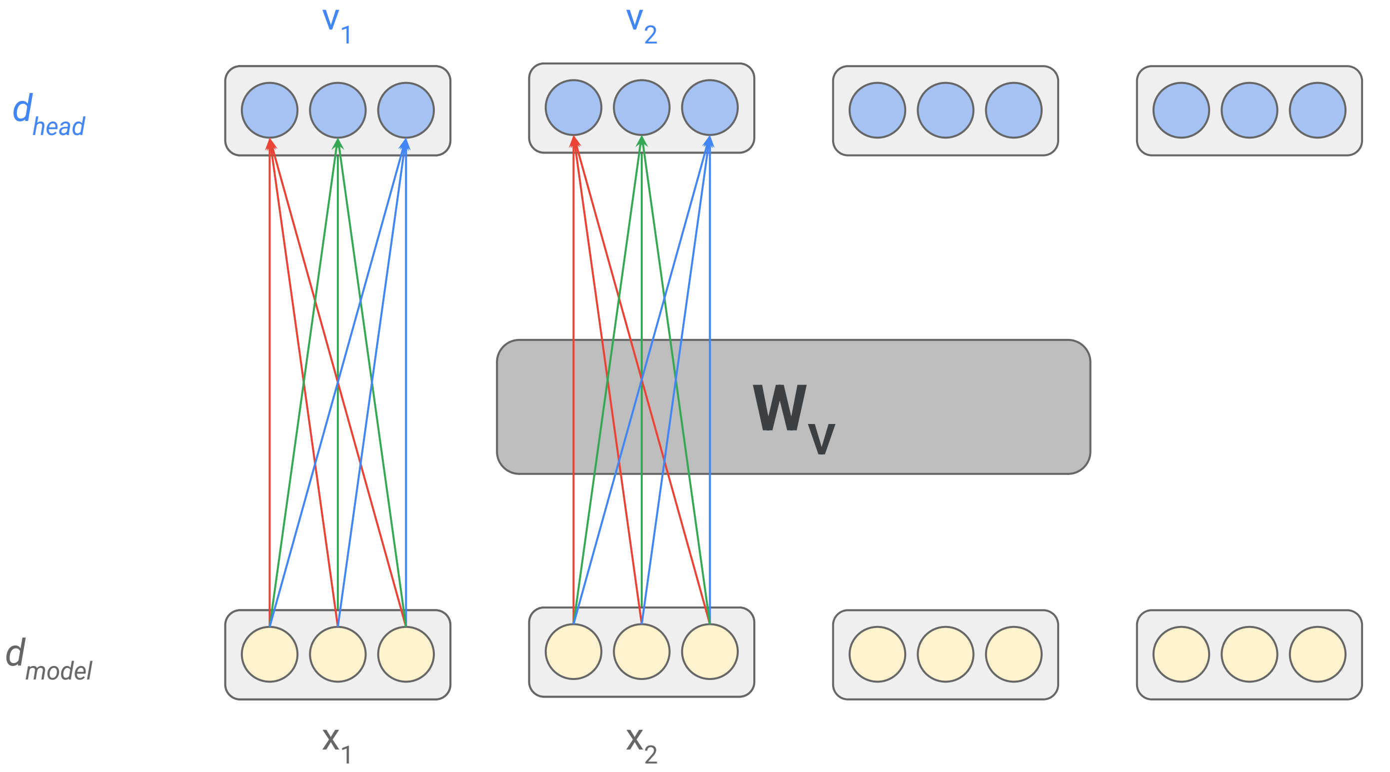
Key vectors



$K = X \cdot W_K$

linear projections

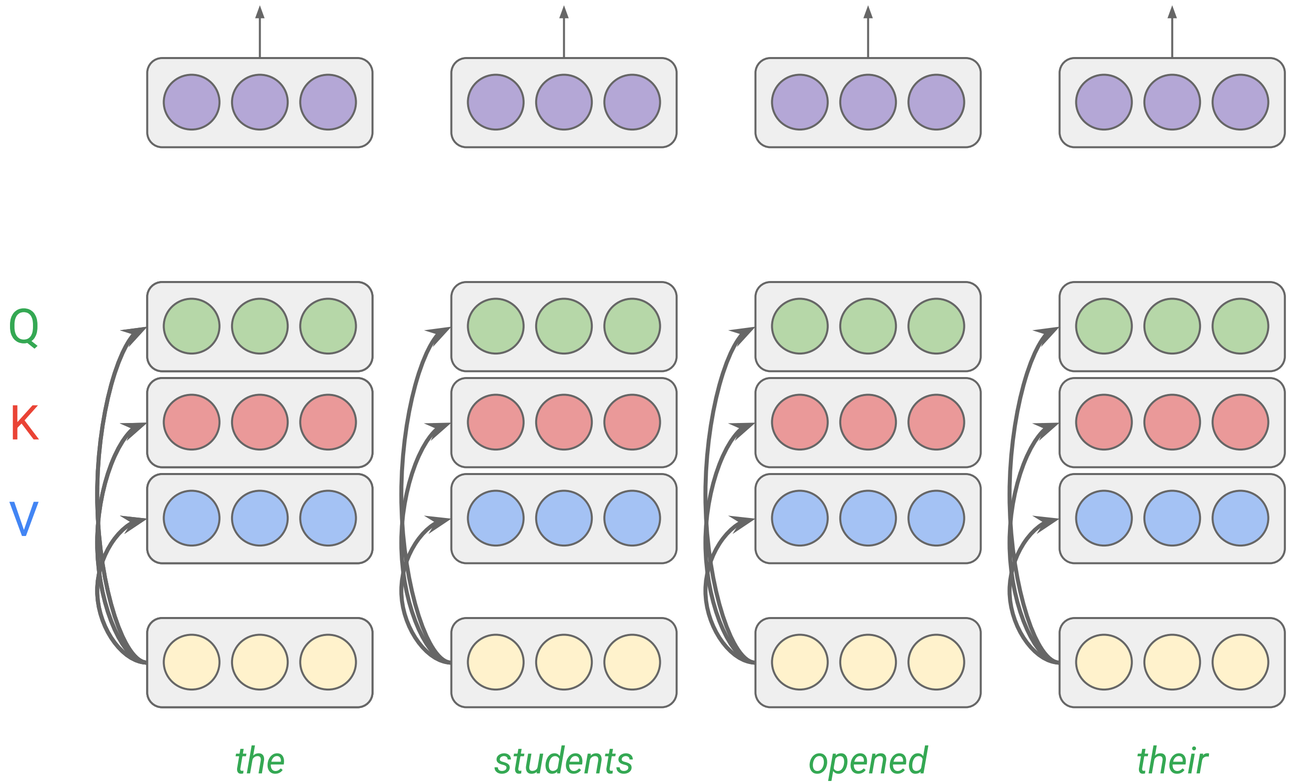
Value vectors



$$V = X \cdot W_V$$

linear projections

Attention (cont'd)



$$Q = X \cdot W_Q$$

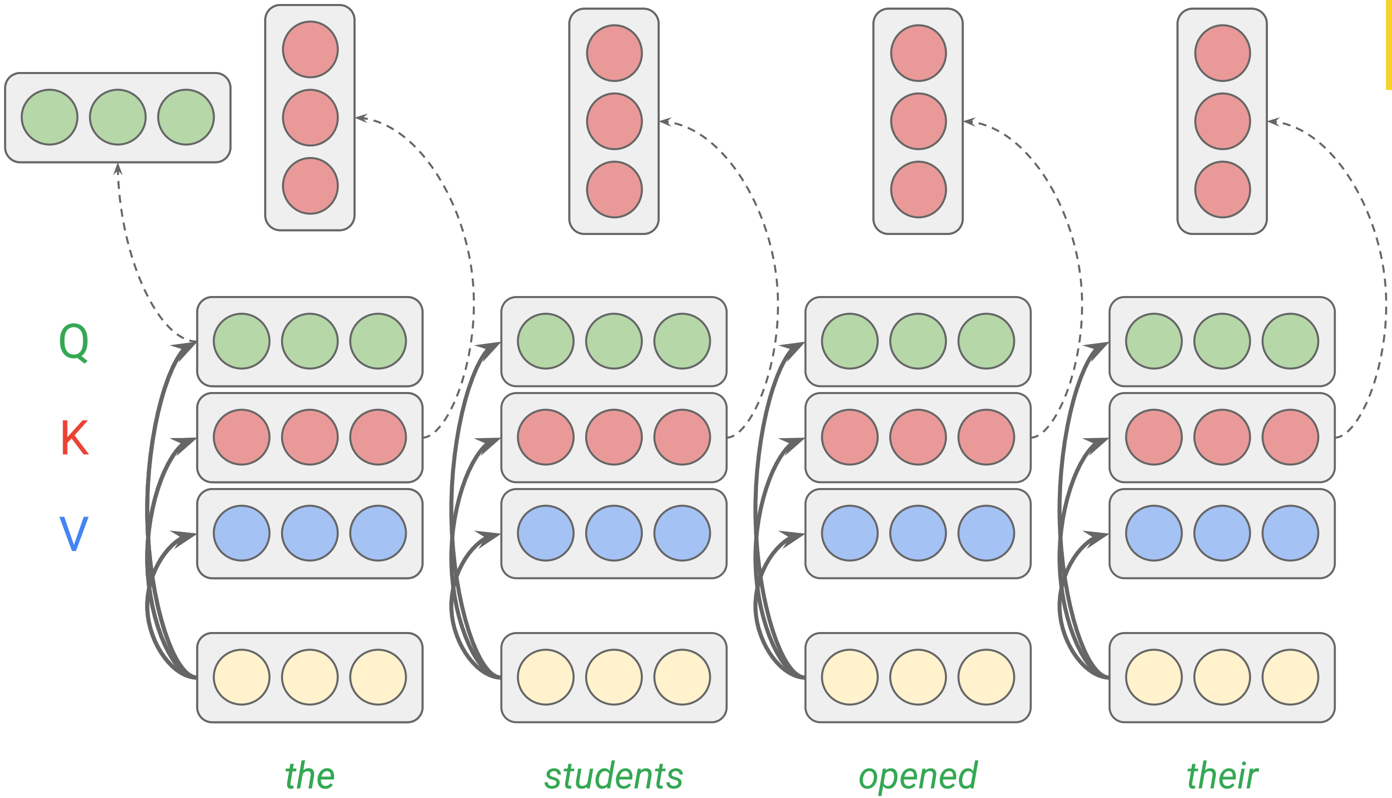
$$K = X \cdot W_K$$

$$V = X \cdot W_V$$

linear projections

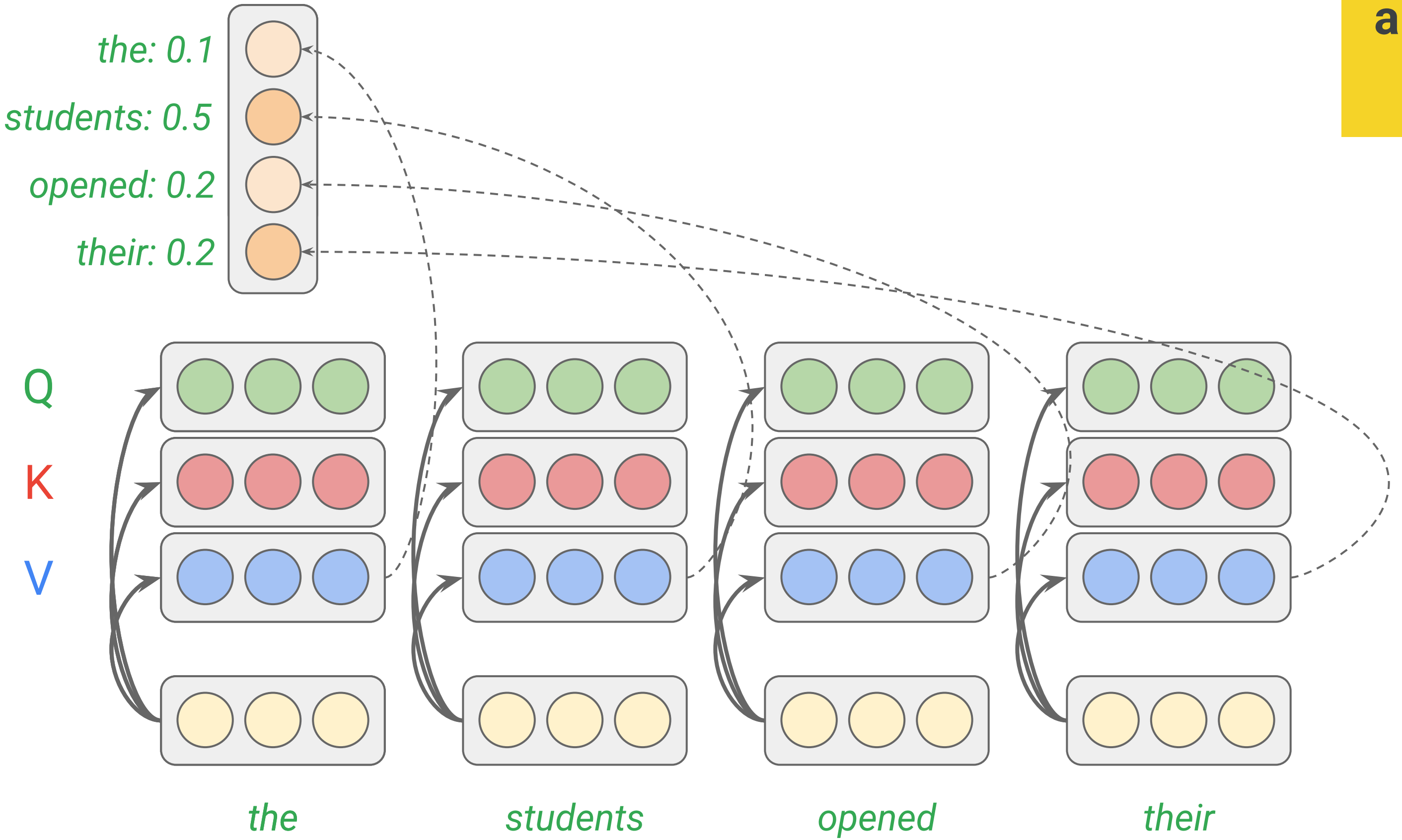
Self-attention (cont'd)

all **computations are parallelized**



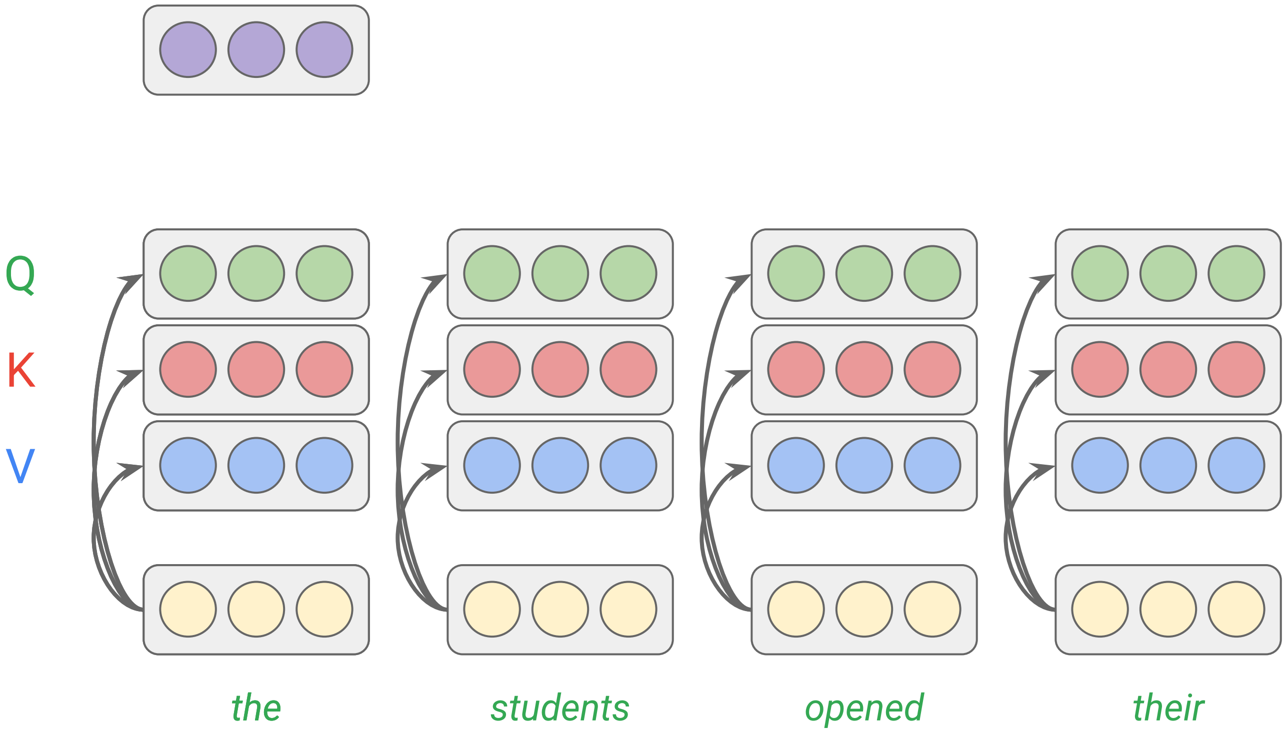
Self-attention (cont'd)

***all* computations are parallelized**

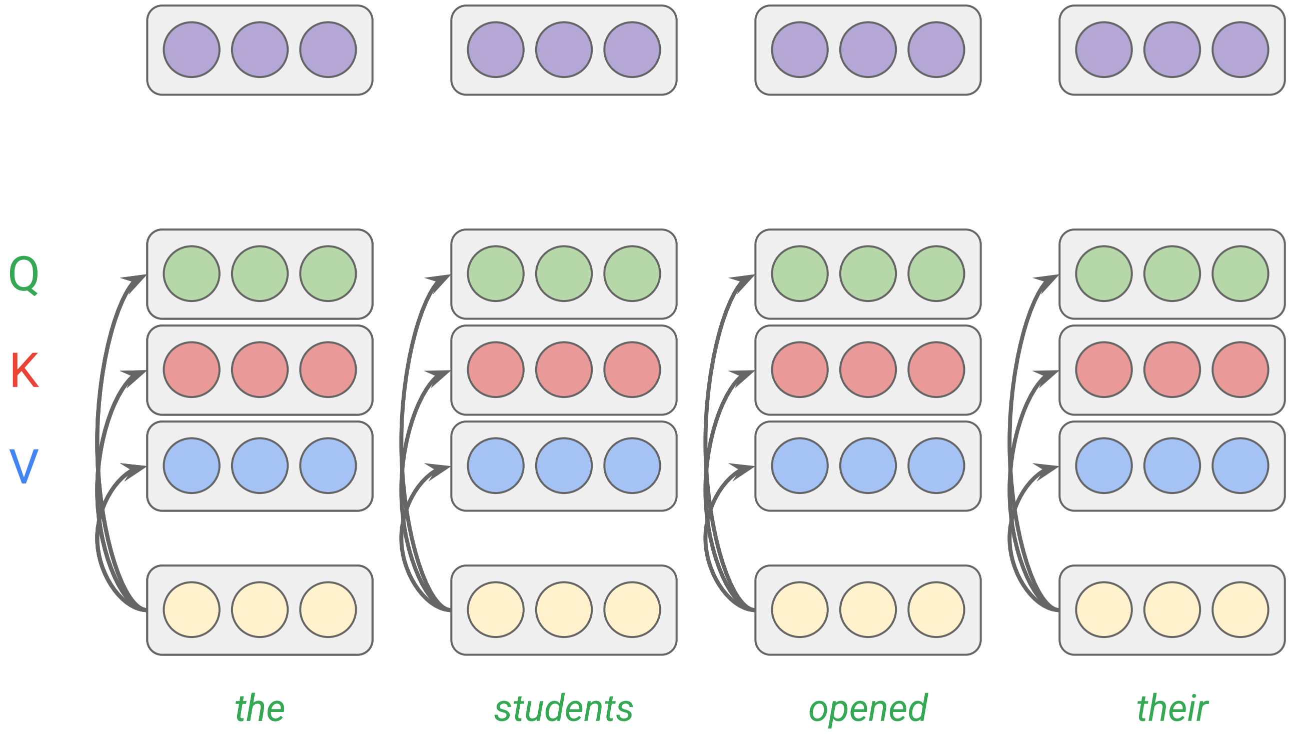


Self-attention (cont'd)

all **computations are parallelized**

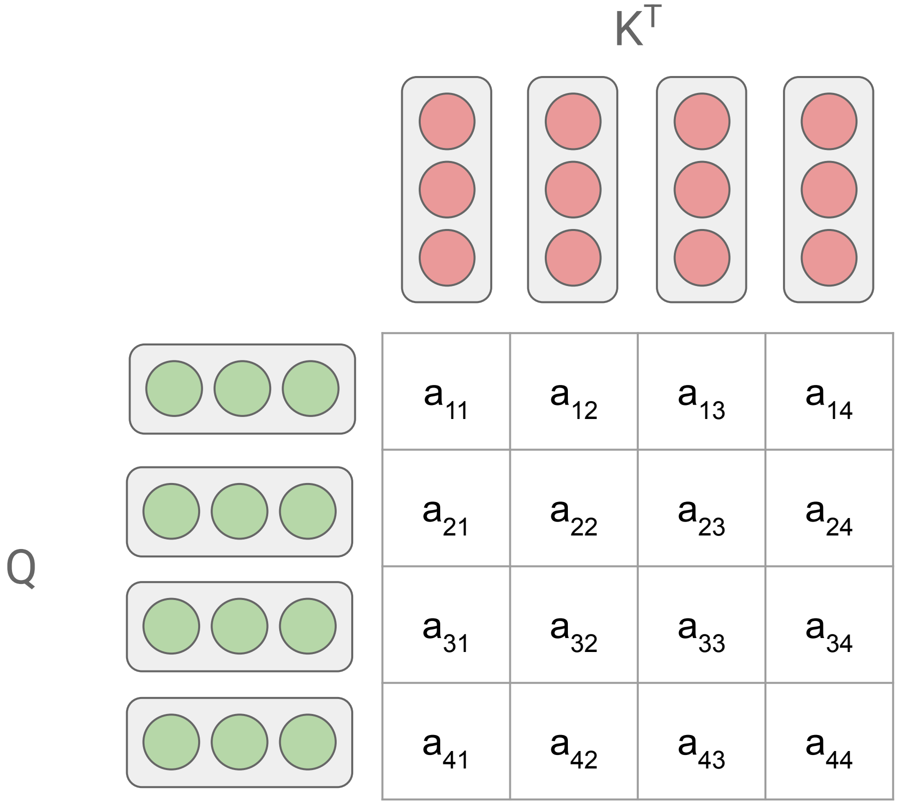


Self-attention (cont'd)



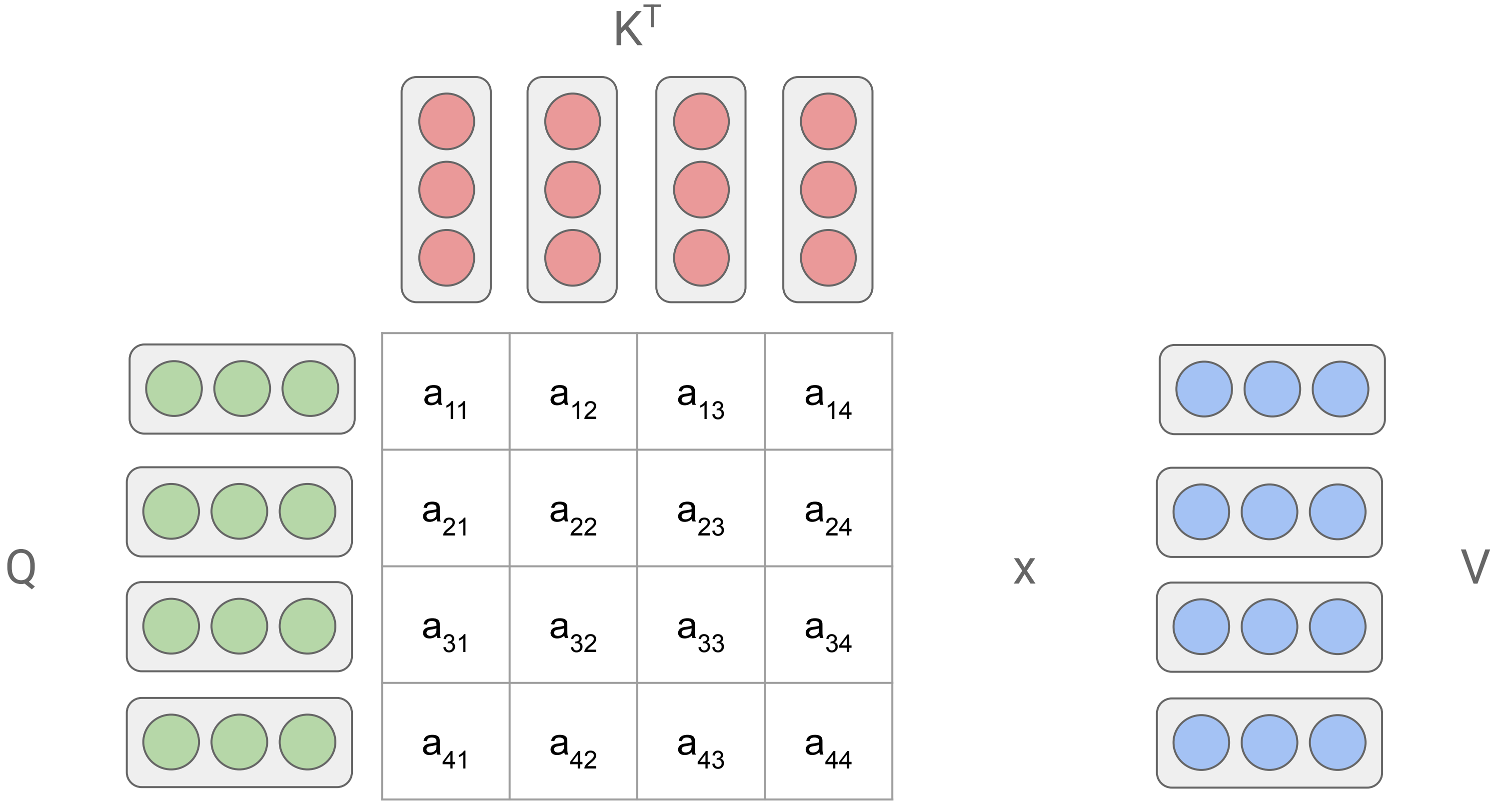
all **computations are parallelized during training and sequential during inference**

Quadratic complexity



The time complexity of self-attention is quadratic in the input length $O(n^2)$

Quadratic complexity



Let

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, \quad V = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \\ v_{41} & v_{42} & v_{43} \end{bmatrix},$$

where the hidden dimension is 3.

Then $AV \in \mathbb{R}^{4\times 3}$ and expands to

$$AV = \begin{bmatrix} a_{11}v_{11} + a_{12}v_{21} + a_{13}v_{31} + a_{14}v_{41} & a_{11}v_{12} + a_{12}v_{22} + a_{13}v_{32} + a_{14}v_{42} & a_{11}v_{13} + a_{12}v_{23} + a_{13}v_{33} + a_{14}v_{43} \\ a_{21}v_{11} + a_{22}v_{21} + a_{23}v_{31} + a_{24}v_{41} & a_{21}v_{12} + a_{22}v_{22} + a_{23}v_{32} + a_{24}v_{42} & a_{21}v_{13} + a_{22}v_{23} + a_{23}v_{33} + a_{24}v_{43} \\ a_{31}v_{11} + a_{32}v_{21} + a_{33}v_{31} + a_{34}v_{41} & a_{31}v_{12} + a_{32}v_{22} + a_{33}v_{32} + a_{34}v_{42} & a_{31}v_{13} + a_{32}v_{23} + a_{33}v_{33} + a_{34}v_{43} \\ a_{41}v_{11} + a_{42}v_{21} + a_{43}v_{31} + a_{44}v_{41} & a_{41}v_{12} + a_{42}v_{22} + a_{43}v_{32} + a_{44}v_{42} & a_{41}v_{13} + a_{42}v_{23} + a_{43}v_{33} + a_{44}v_{43} \end{bmatrix}$$

Expanding the first row of AV,

$$(AV)_{1:} = \left[a_{11}v_{11} + a_{12}v_{21} + a_{13}v_{31} + a_{14}v_{41},\ a_{11}v_{12} + a_{12}v_{22} + a_{13}v_{32} + a_{14}v_{42},\ a_{11}v_{13} + a_{12}v_{23} + a_{13}v_{33} + a_{14}v_{43}\right]$$

Rewrite this as a column vector,

$$(AV)_{1:}^{\top} = \begin{bmatrix} a_{11}v_{11} + a_{12}v_{21} + a_{13}v_{31} + a_{14}v_{41} \\ a_{11}v_{12} + a_{12}v_{22} + a_{13}v_{32} + a_{14}v_{42} \\ a_{11}v_{13} + a_{12}v_{23} + a_{13}v_{33} + a_{14}v_{43} \end{bmatrix}.$$

Factor by coefficients,

$$(AV)_{1:}^{\top} = a_{11} \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} + a_{12} \begin{bmatrix} v_{21} \\ v_{22} \\ v_{23} \end{bmatrix} + a_{13} \begin{bmatrix} v_{31} \\ v_{32} \\ v_{33} \end{bmatrix} + a_{14} \begin{bmatrix} v_{41} \\ v_{42} \\ v_{43} \end{bmatrix}.$$

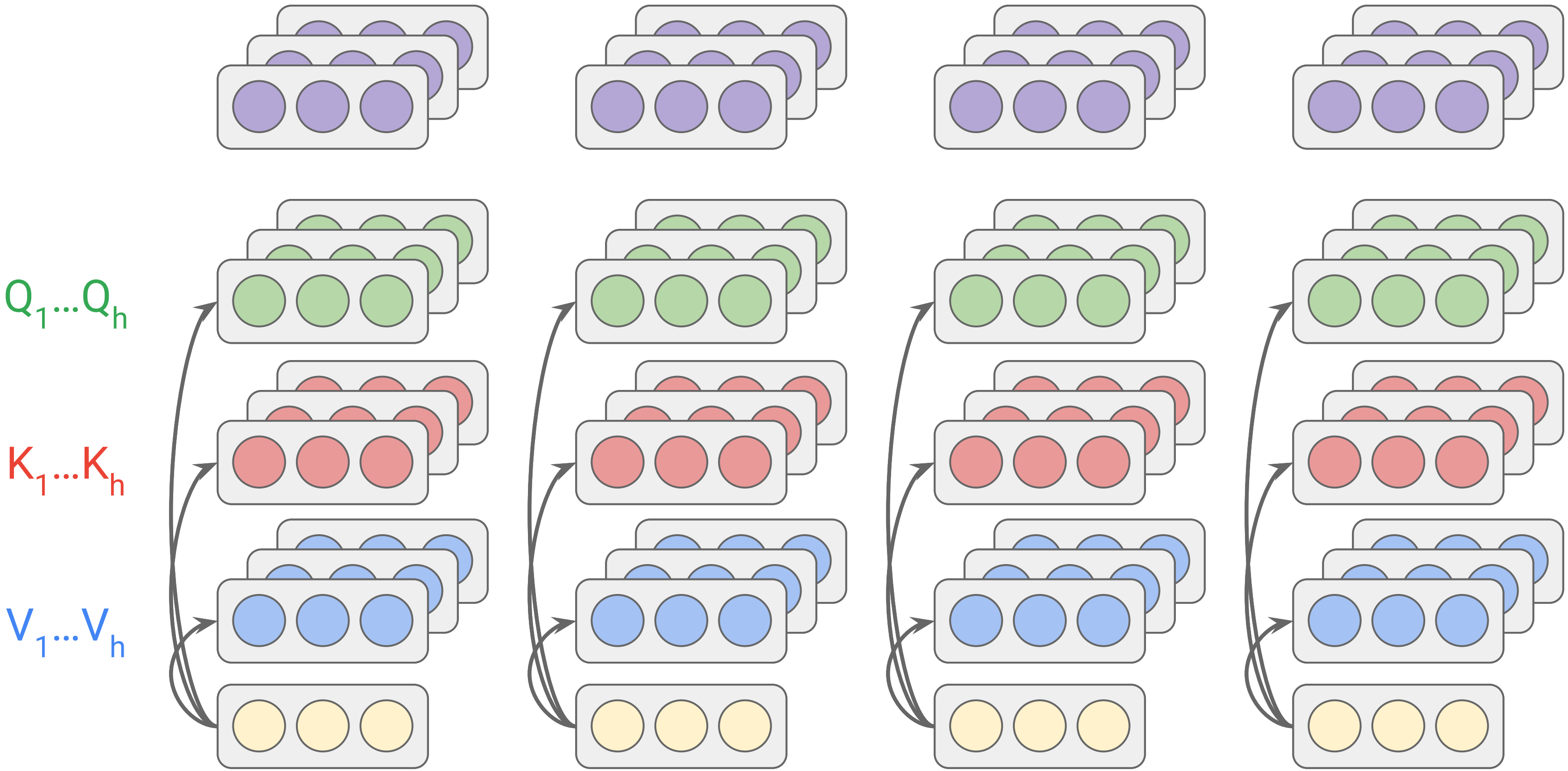
All computations are parallelized

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

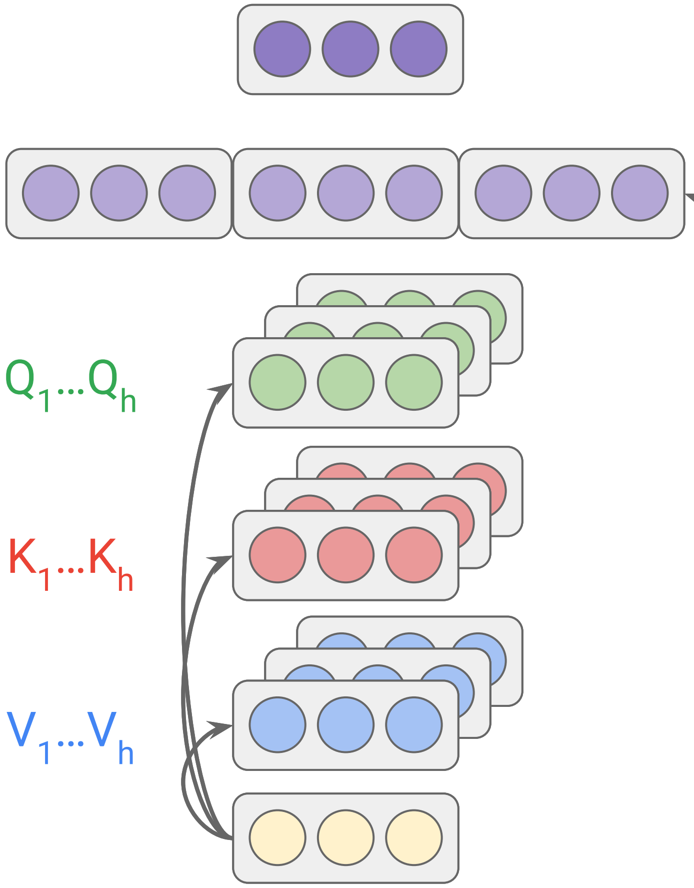
d_k: scaling factor

large products push the softmax function into regions where it has extremely small gradients

Multi-head attention

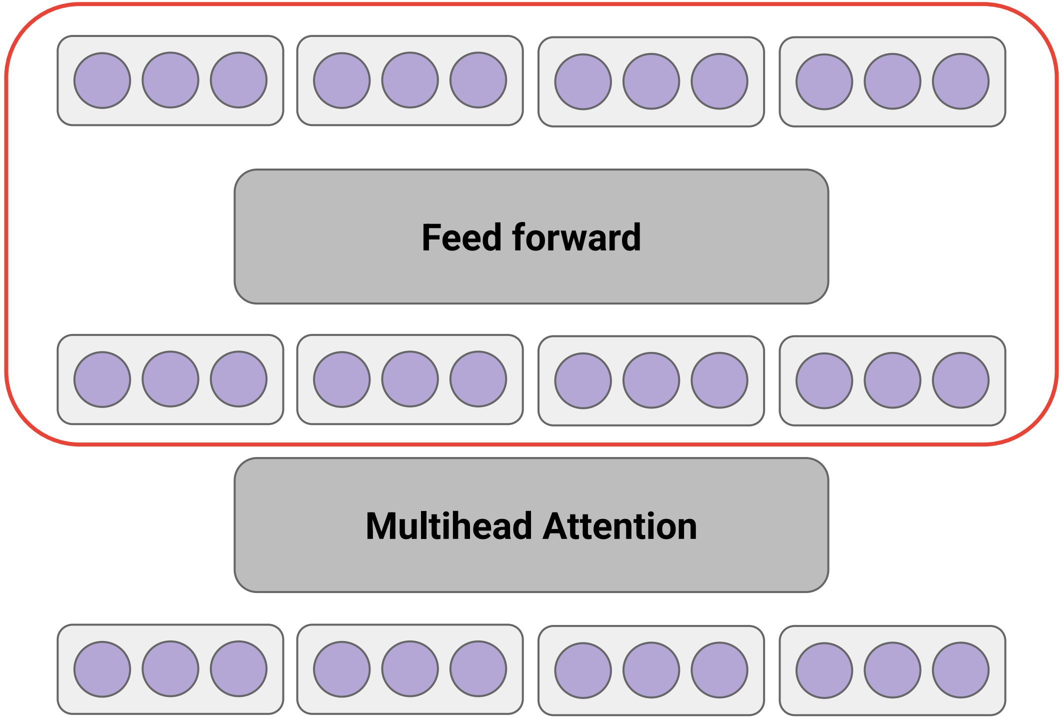


Multi-head attention (cont'd)

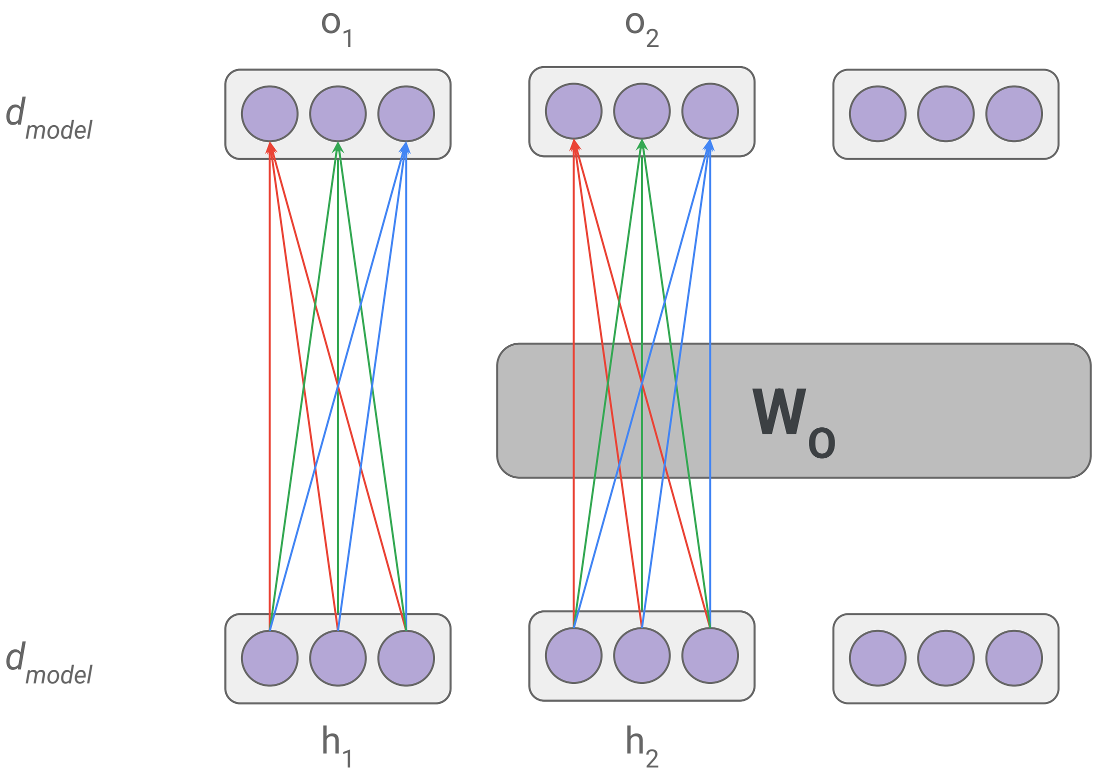


These output values are concatenated and once again projected

Transformer block (cont'd)



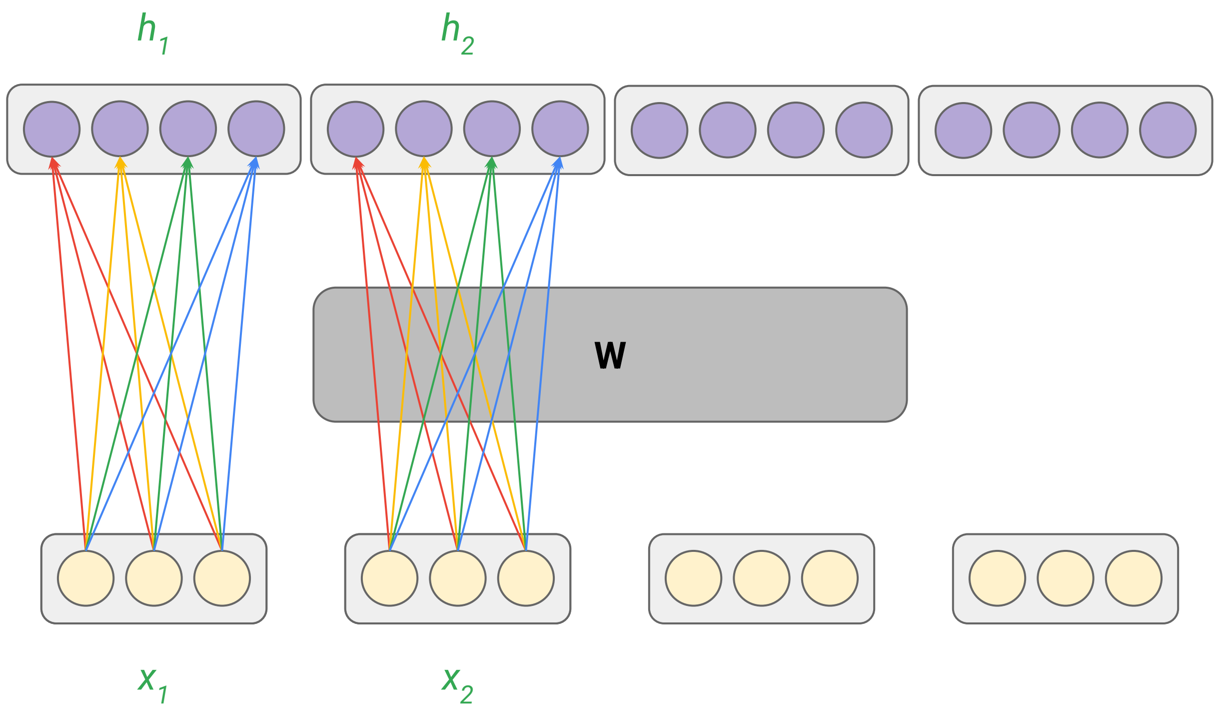
output vectors



$$O = H \cdot W_O$$

linear projections

Position-wise feedforward networks



We multiply the weight matrix W (size 4×3) with the embeddings matrix X (size 3×2):

$$H = WX$$

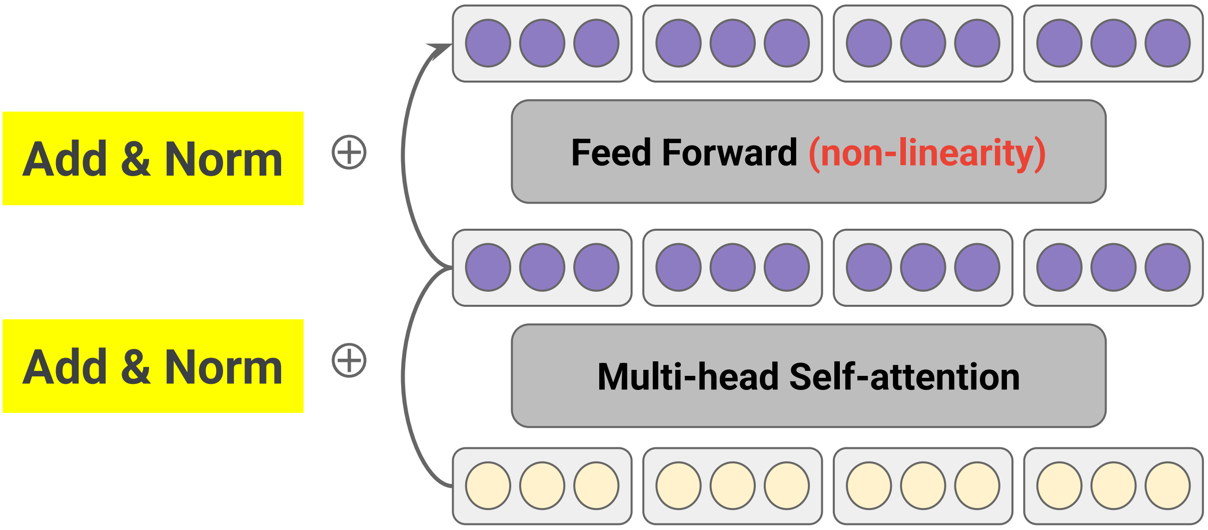
Performing the multiplication:

$$H = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{bmatrix}$$

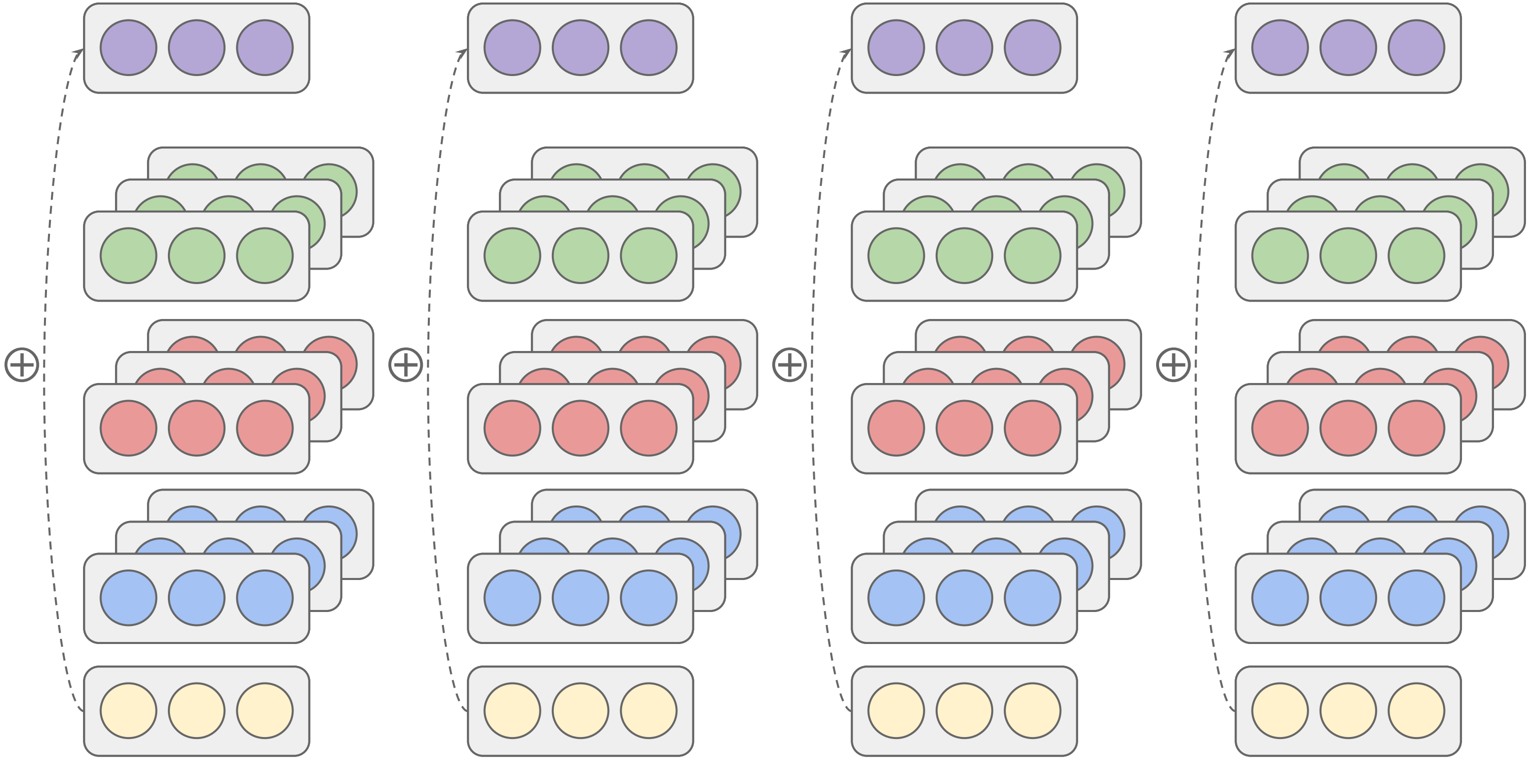
This results in:

$$H = \begin{array}{c} \begin{array}{cc} \qquad\qquad h_1 \qquad\qquad\qquad\qquad & \qquad\qquad h_2 \qquad\qquad \end{array} \\ \begin{bmatrix} w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13} & w_{11}x_{21} + w_{12}x_{22} + w_{13}x_{23} \\ w_{21}x_{11} + w_{22}x_{12} + w_{23}x_{13} & w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23} \\ w_{31}x_{11} + w_{32}x_{12} + w_{33}x_{13} & w_{31}x_{21} + w_{32}x_{22} + w_{33}x_{23} \\ w_{41}x_{11} + w_{42}x_{12} + w_{43}x_{13} & w_{41}x_{21} + w_{42}x_{22} + w_{43}x_{23} \end{bmatrix} \end{array}$$

Transformer block (one layer)



Residual connection



Residual connection

$$\mathrm{output} = \mathrm{sublayer}(x) + x$$

Layer normalization

$$\mathrm{Norm}(z) = \frac{z - \mu}{\sigma} \cdot \gamma + \beta$$

where μ and σ are the mean and standard deviation of the activations, and γ, β are learnable parameters.

Each activation vector is normalized so that its components have mean 0 and variance 1. This prevents activations from becoming too large or too small as they propagate through the network.

Residual connection and layer normalization

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$

Position-wise Feed-Forward Networks

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

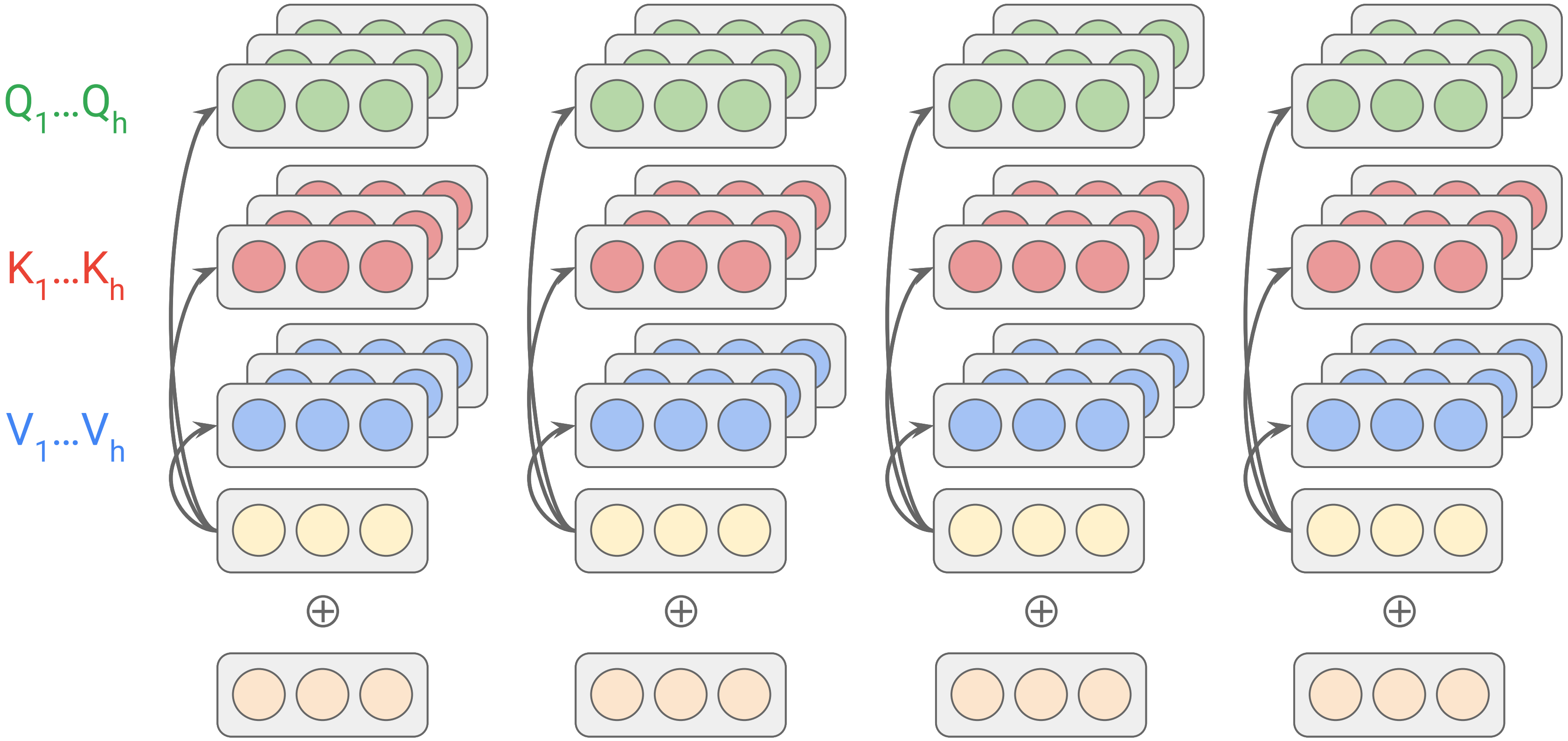
ReLU (Rectified Linear Unit)

Sinusoidal positional encoding

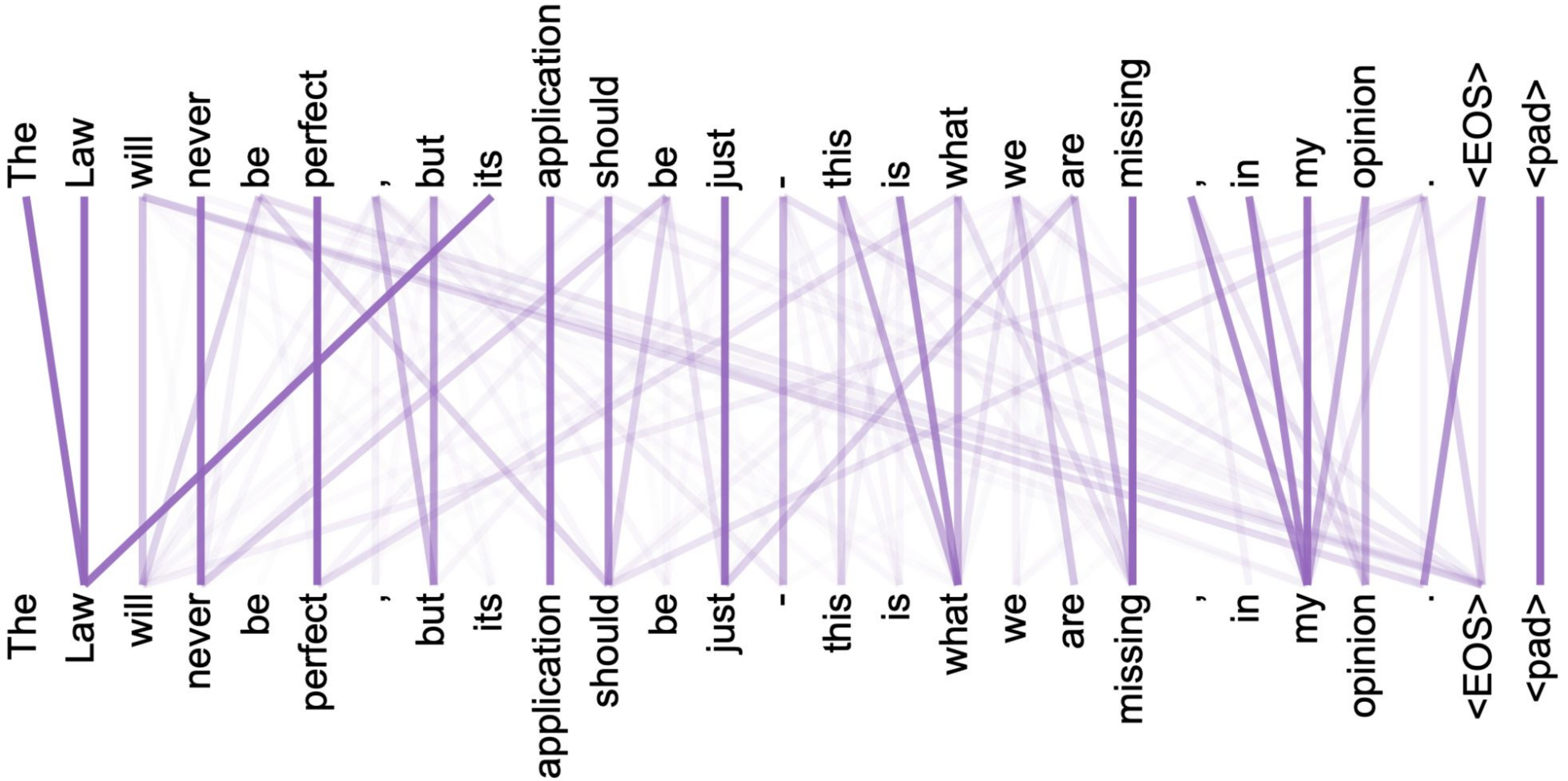
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

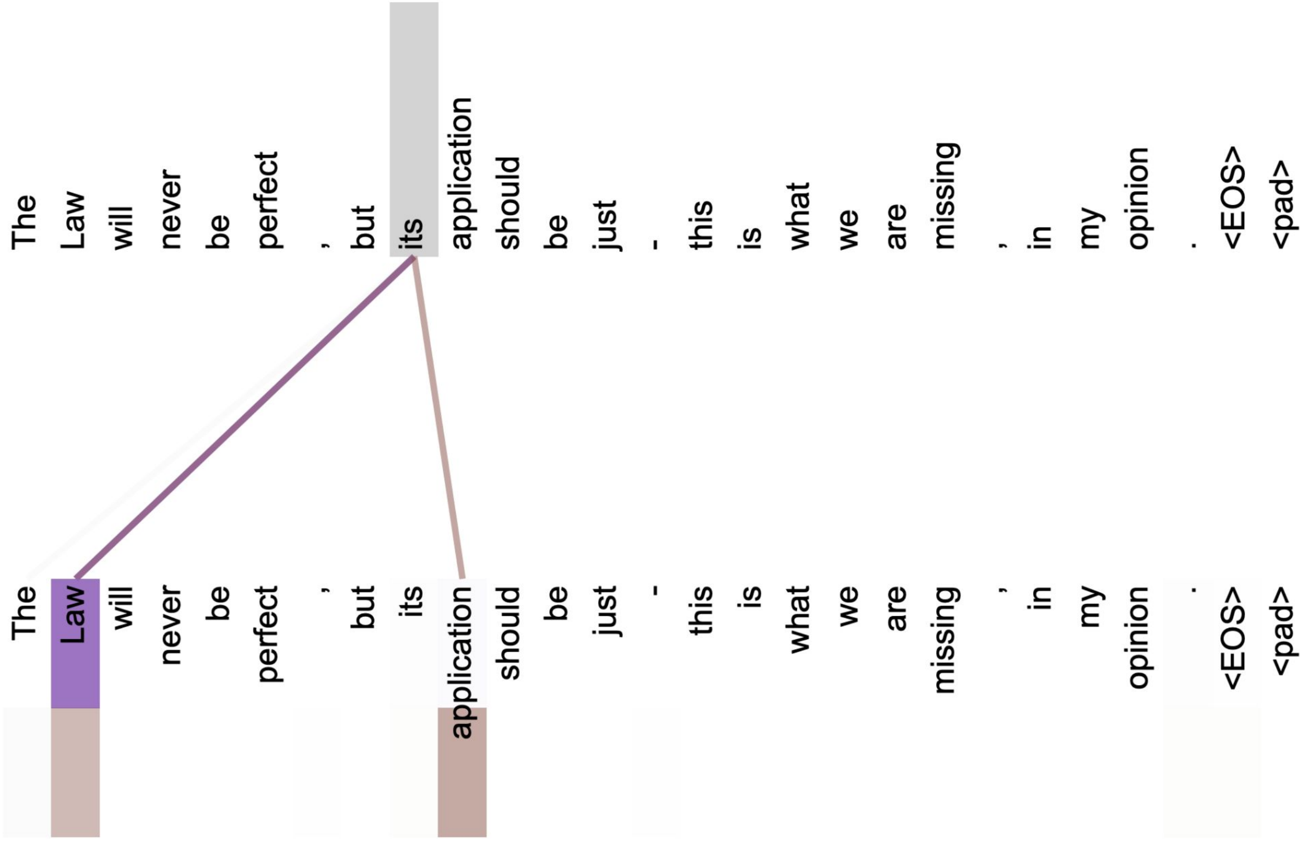
Positional Encoding (cont'd)



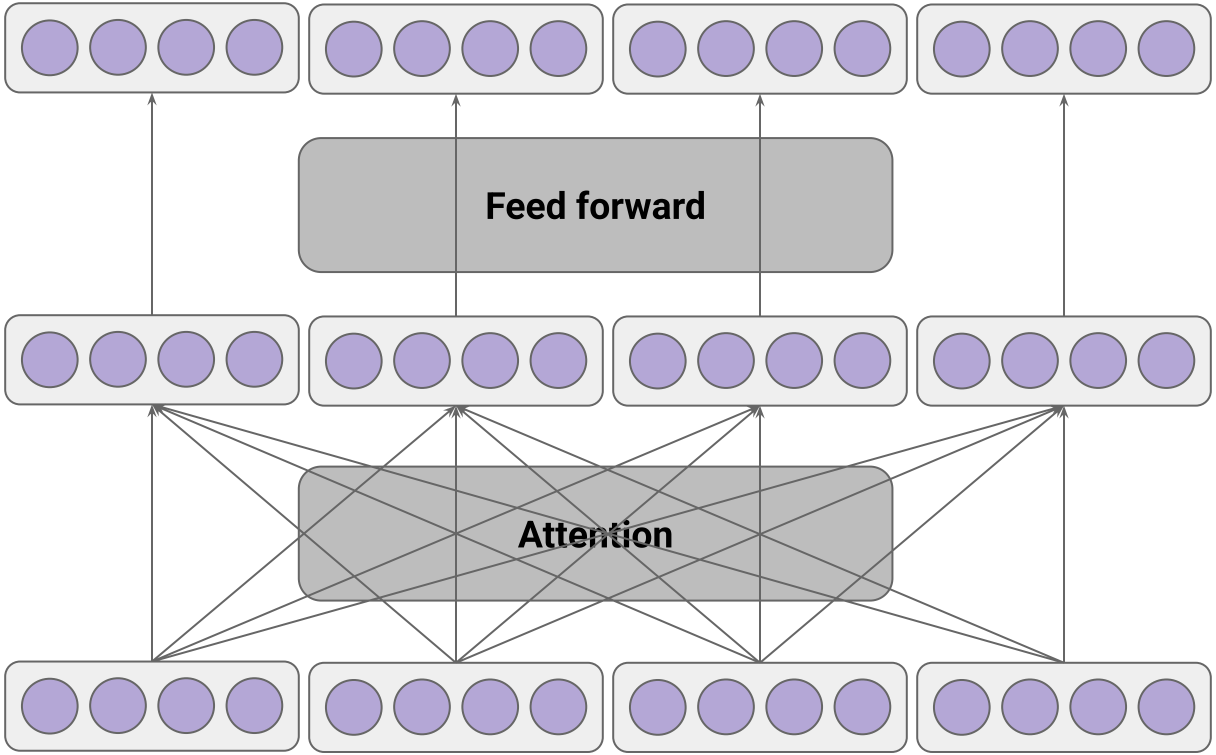
Attention visualizations



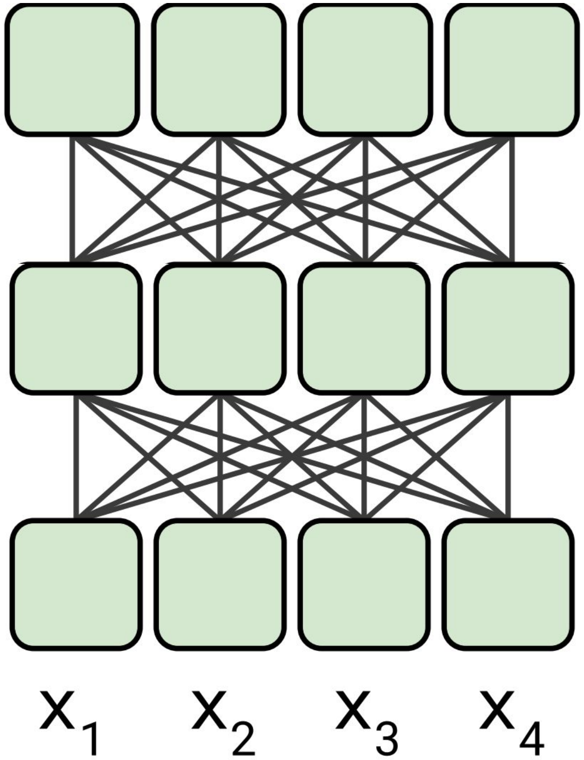
Attention visualizations (cont’d)



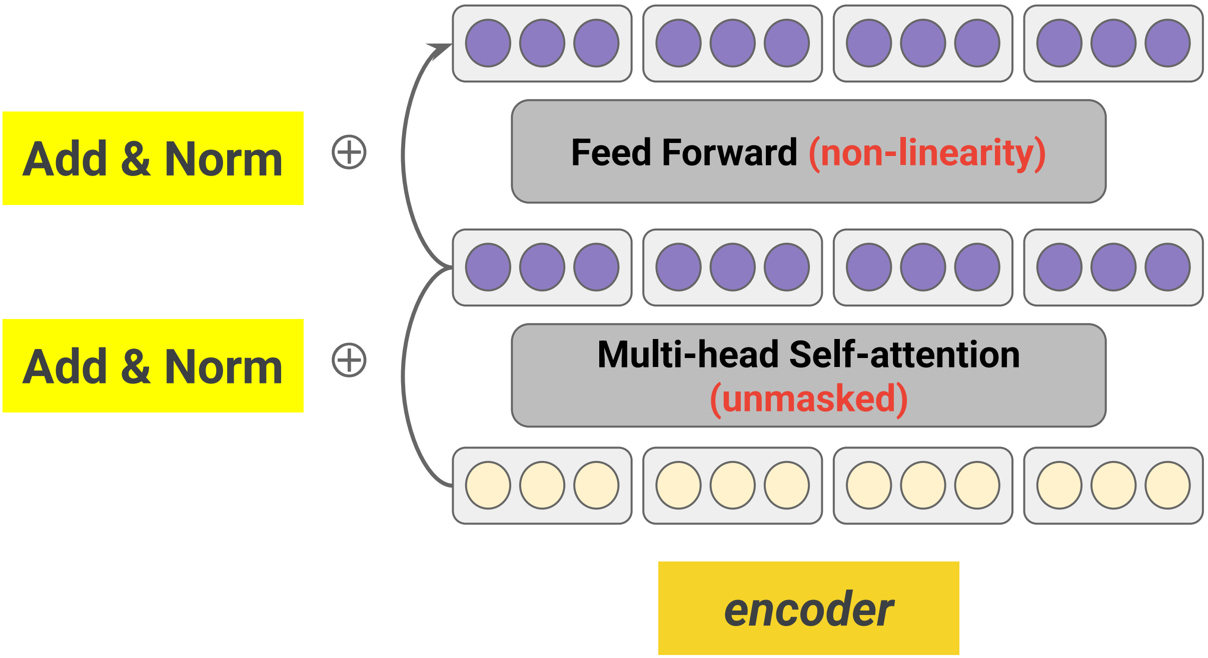
Putting it all together



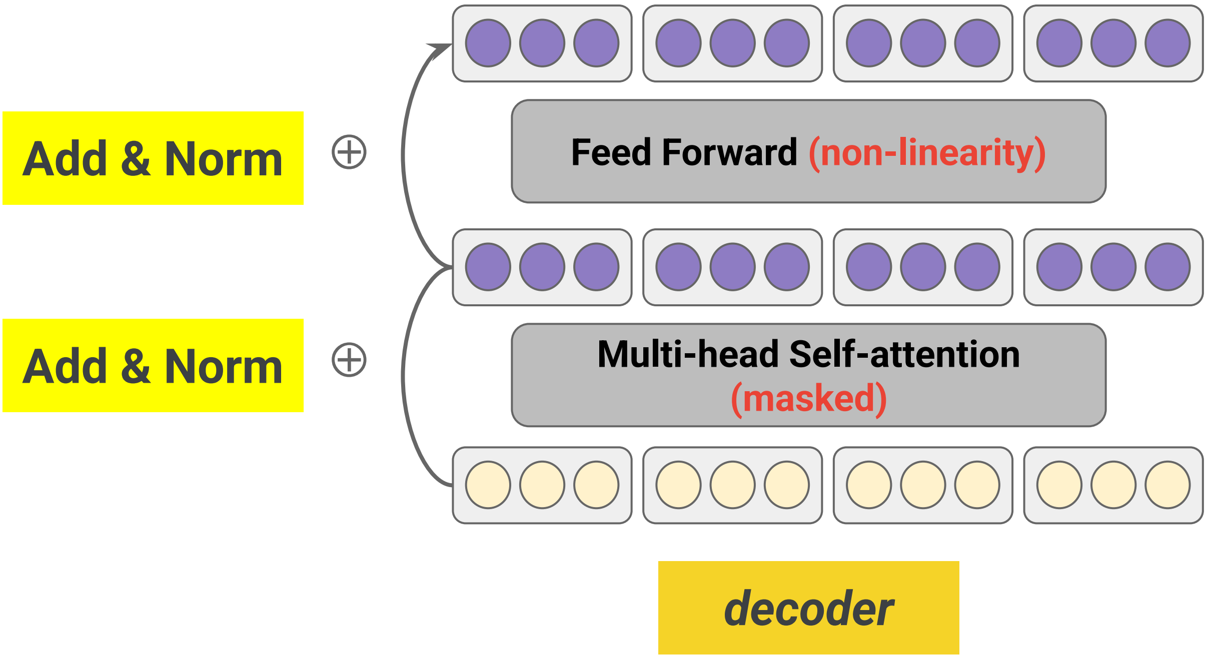
Encoder only



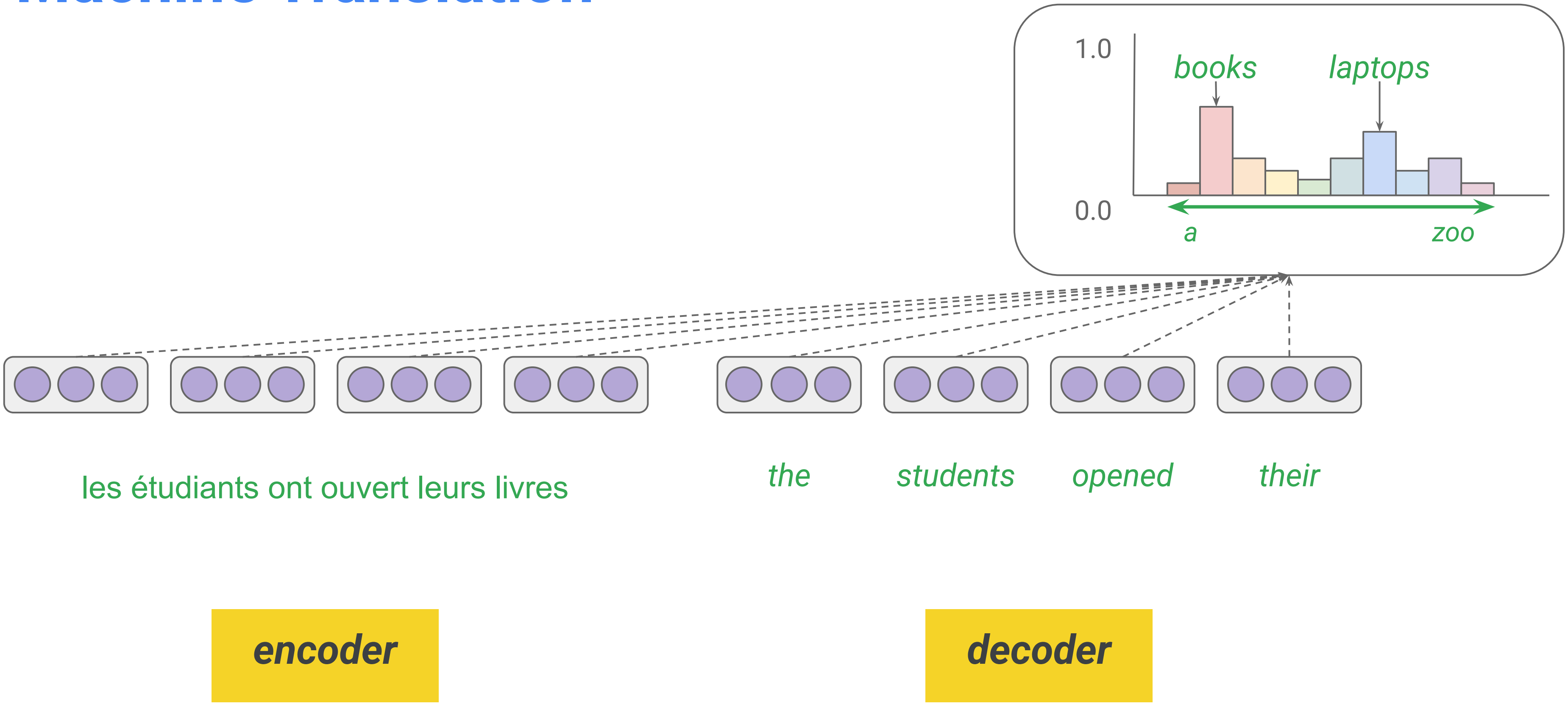
Encoder (one layer)



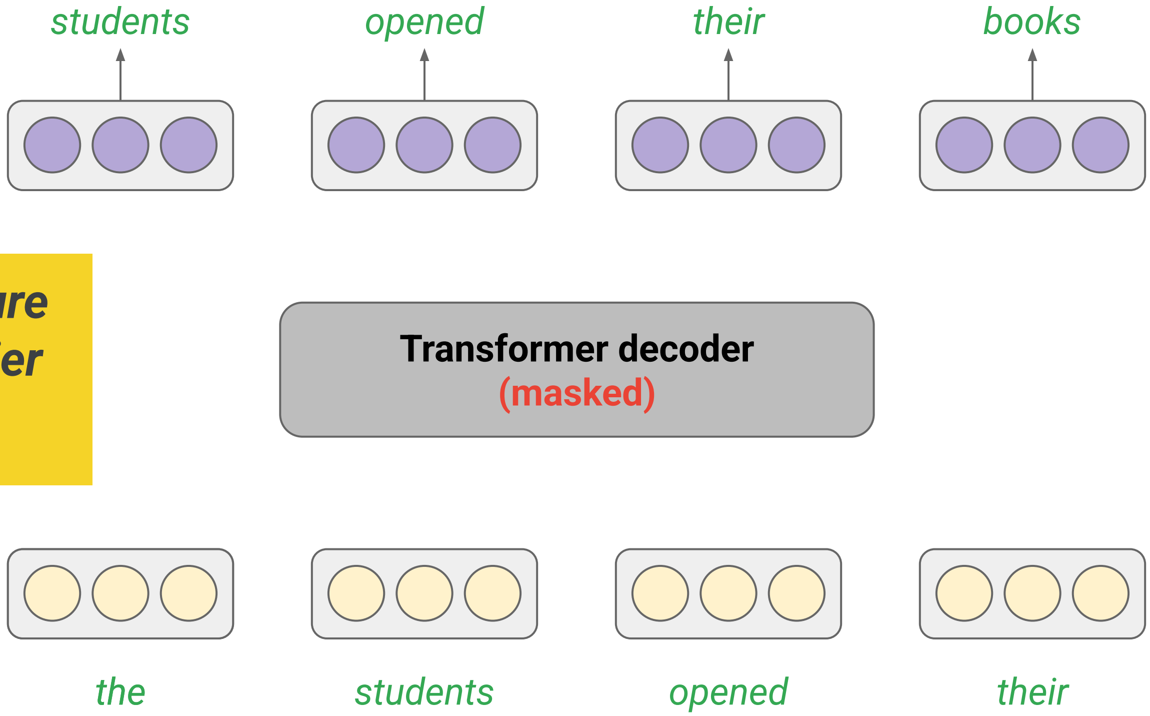
Decoder (one layer)



Machine Translation

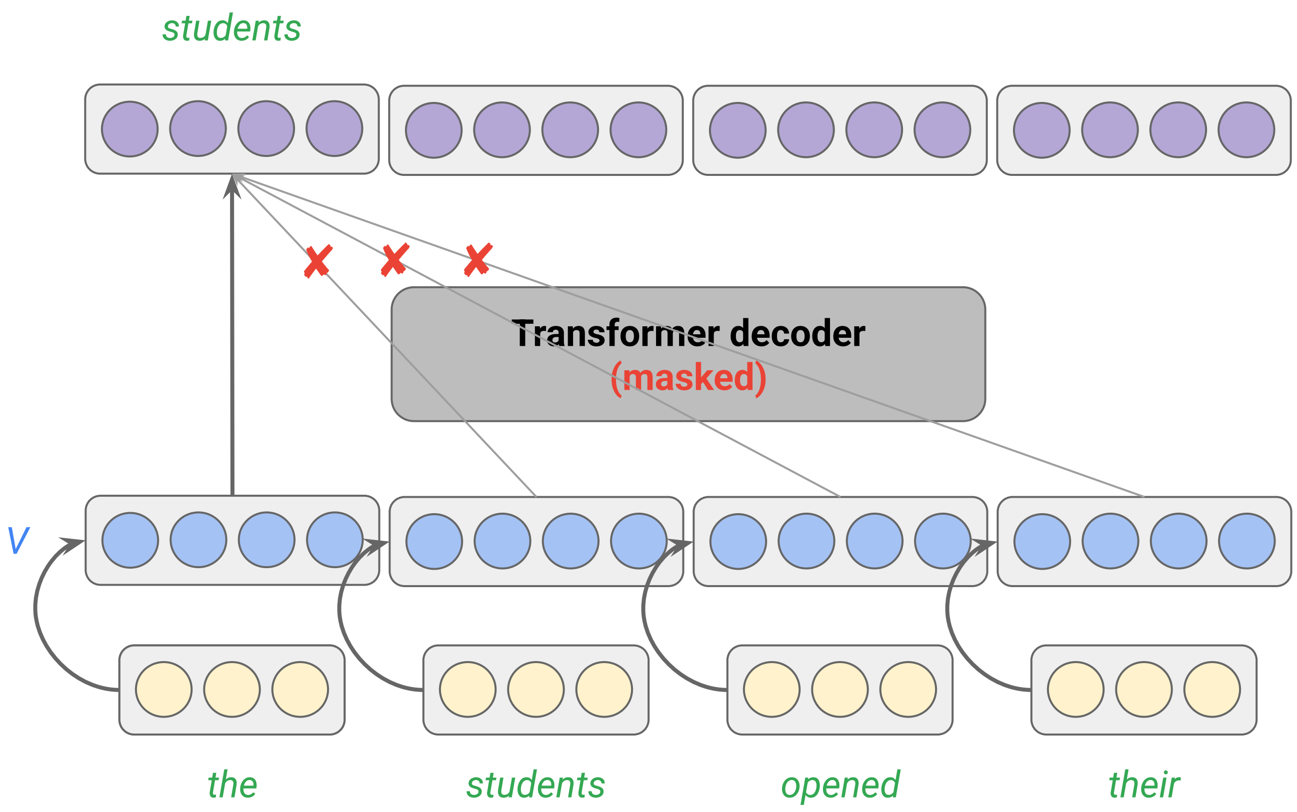


Transformer decoder

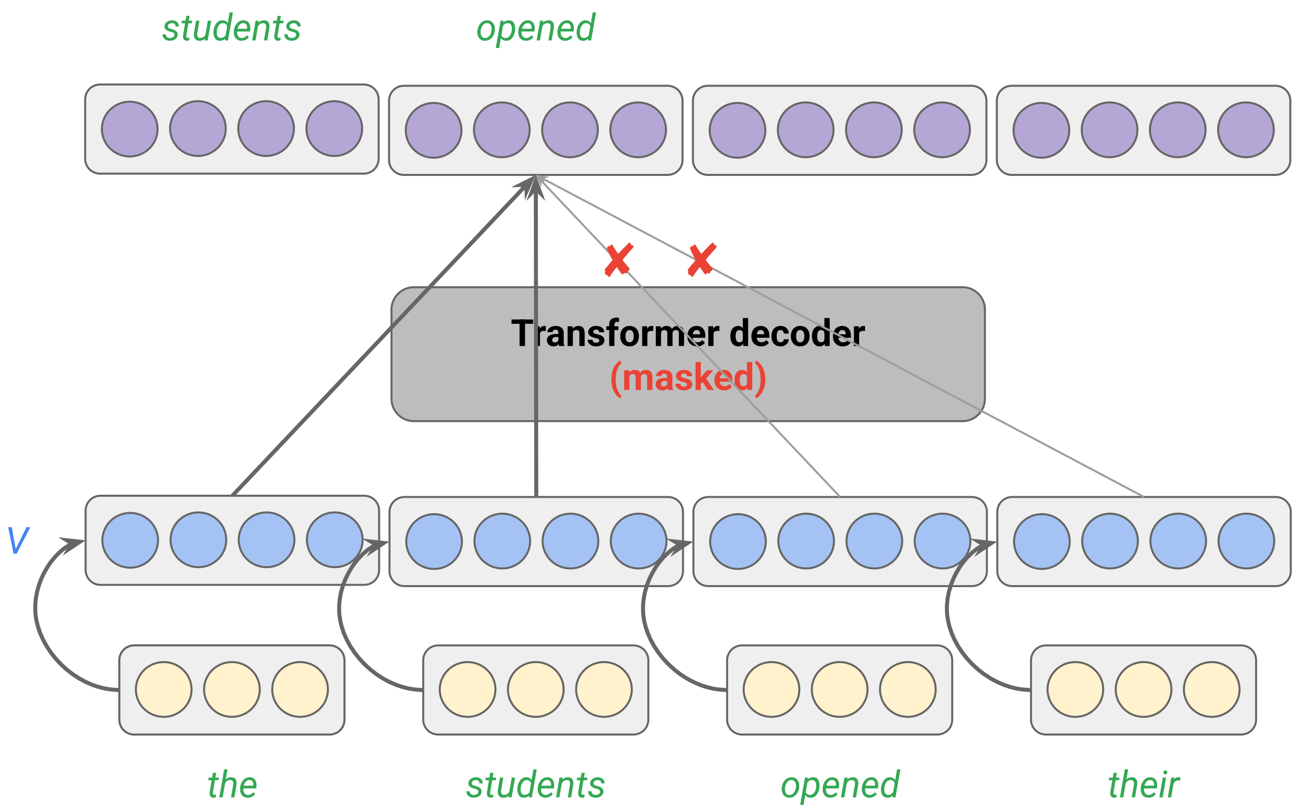


the architecture used in frontier LLMs

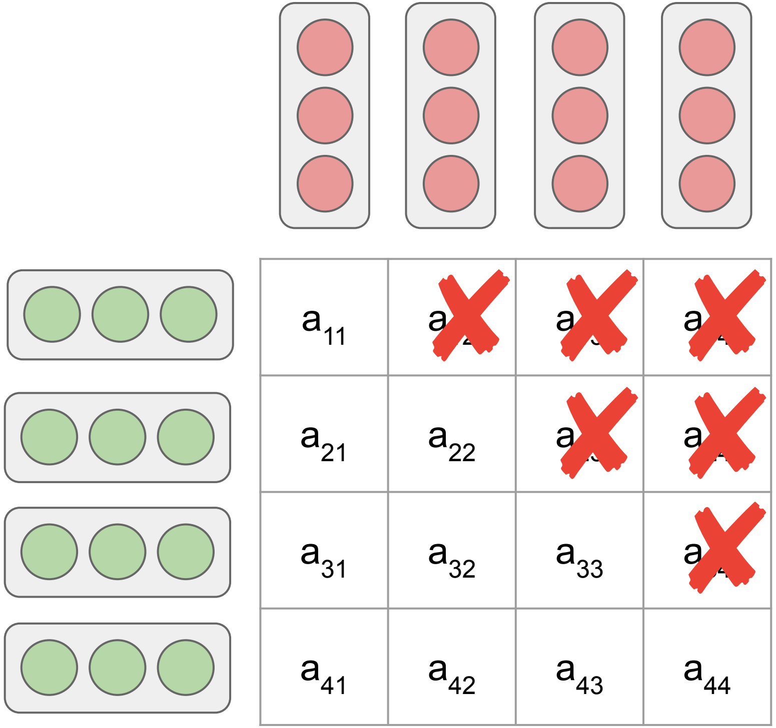
Transformer decoder (cont’d)



Transformer decoder (cont'd)

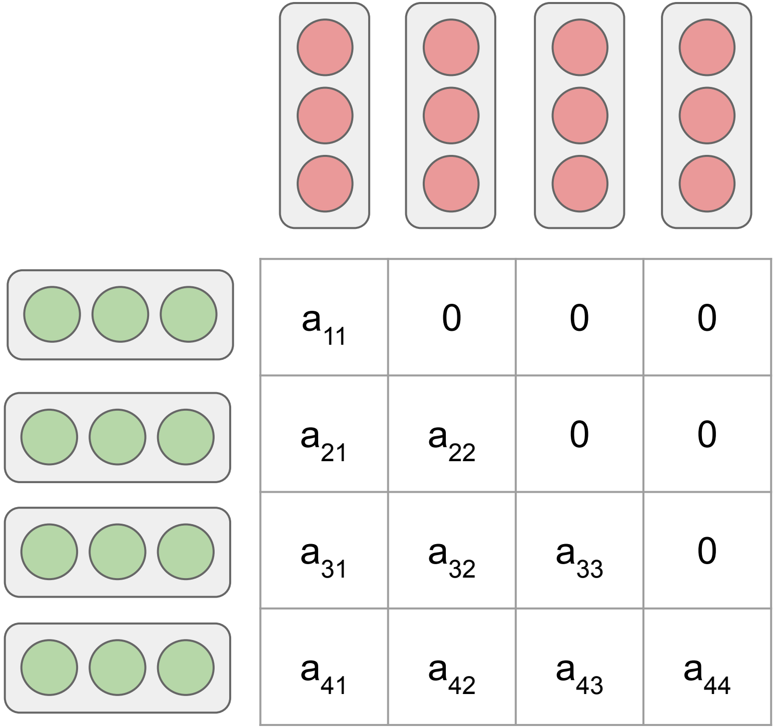


Self-attention in the decoder



masking out all values in the input of the softmax which correspond to illegal connections

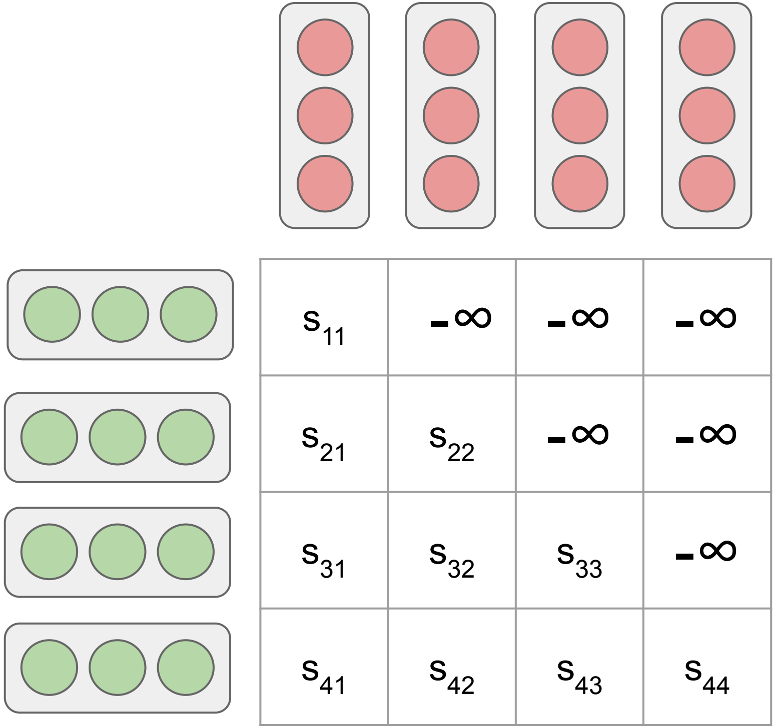
Self-attention in the decoder (cont'd)



a_{11}	0	0	0
a_{21}	a_{22}	0	0
a_{31}	a_{32}	a_{33}	0
a_{41}	a_{42}	a_{43}	a_{44}

masking out all values in the input of the softmax which correspond to illegal connections

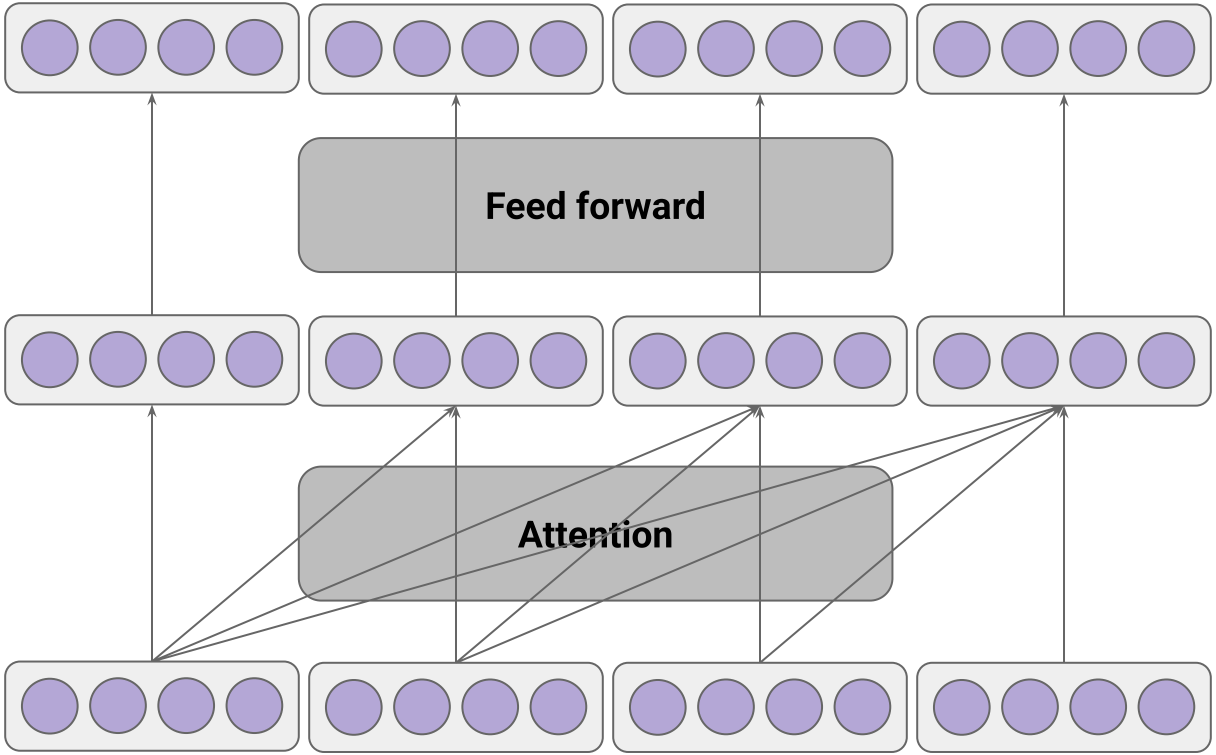
Self-attention in the decoder (cont'd)



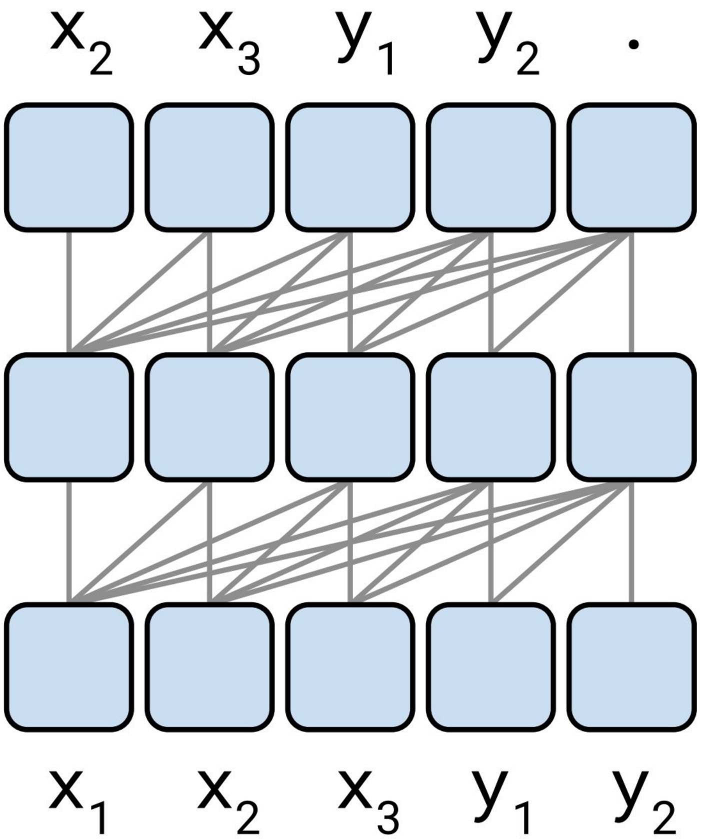
s_{11}	$-\infty$	$-\infty$	$-\infty$
s_{21}	s_{22}	$-\infty$	$-\infty$
s_{31}	s_{32}	s_{33}	$-\infty$
s_{41}	s_{42}	s_{43}	s_{44}

masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections

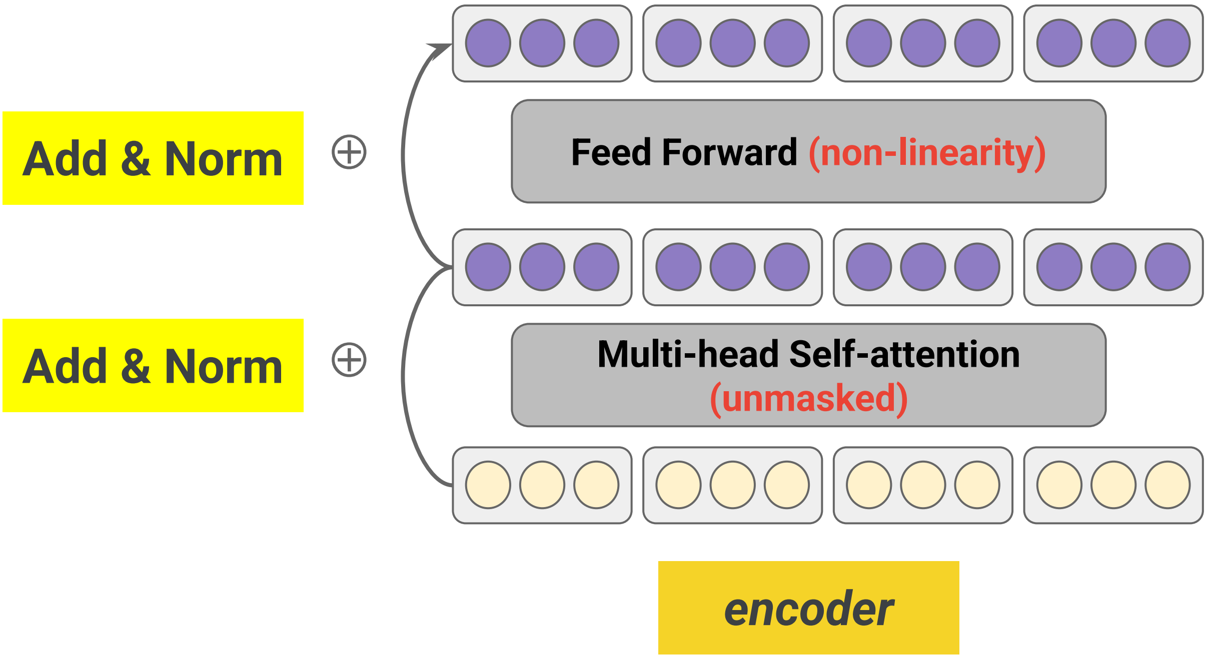
Decoder (cont'd)



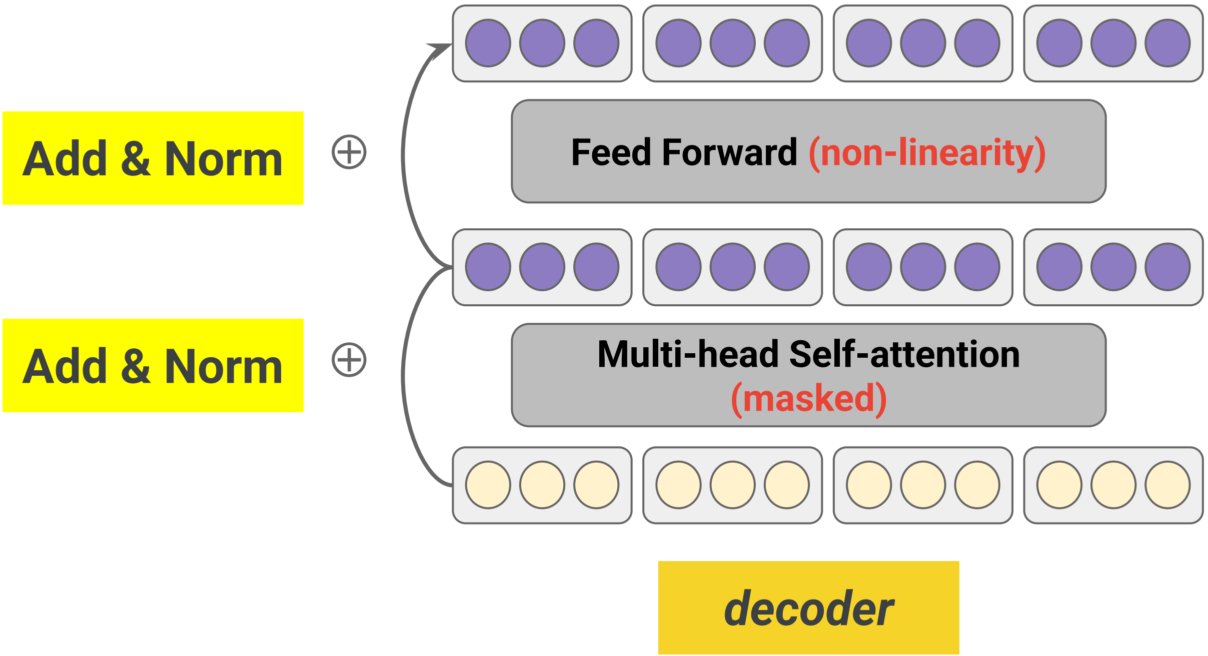
Decoder (cont'd)



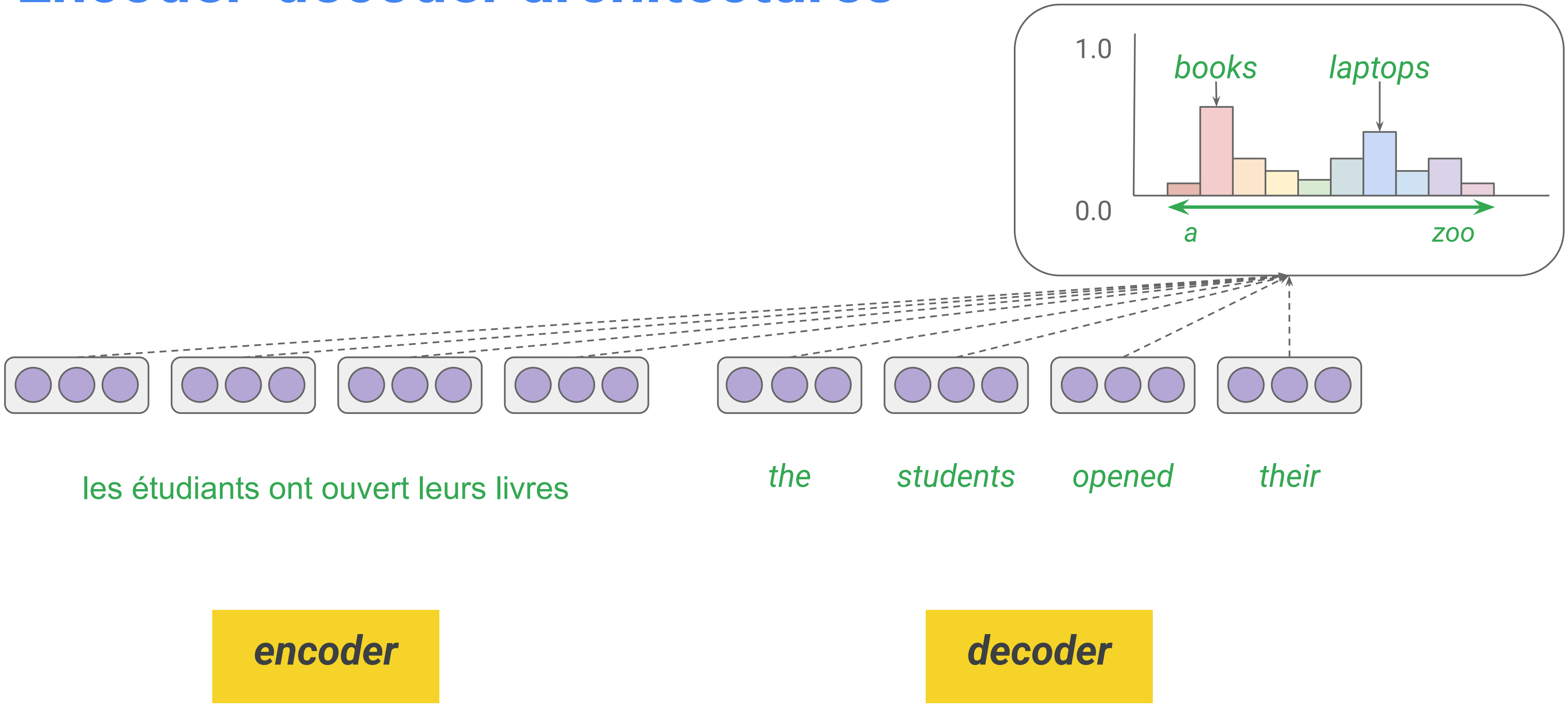
Encoder (one layer)



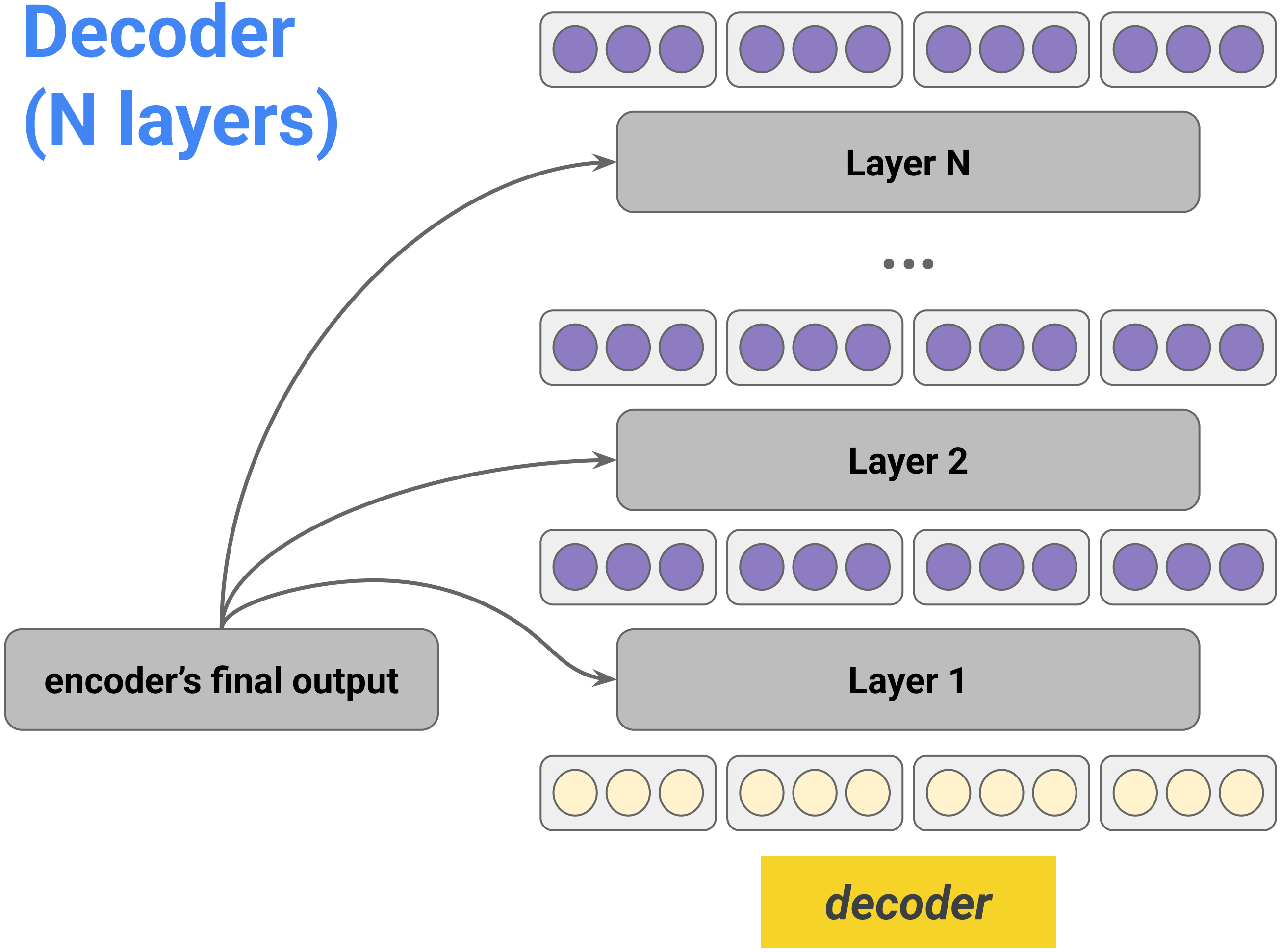
Decoder (one layer)



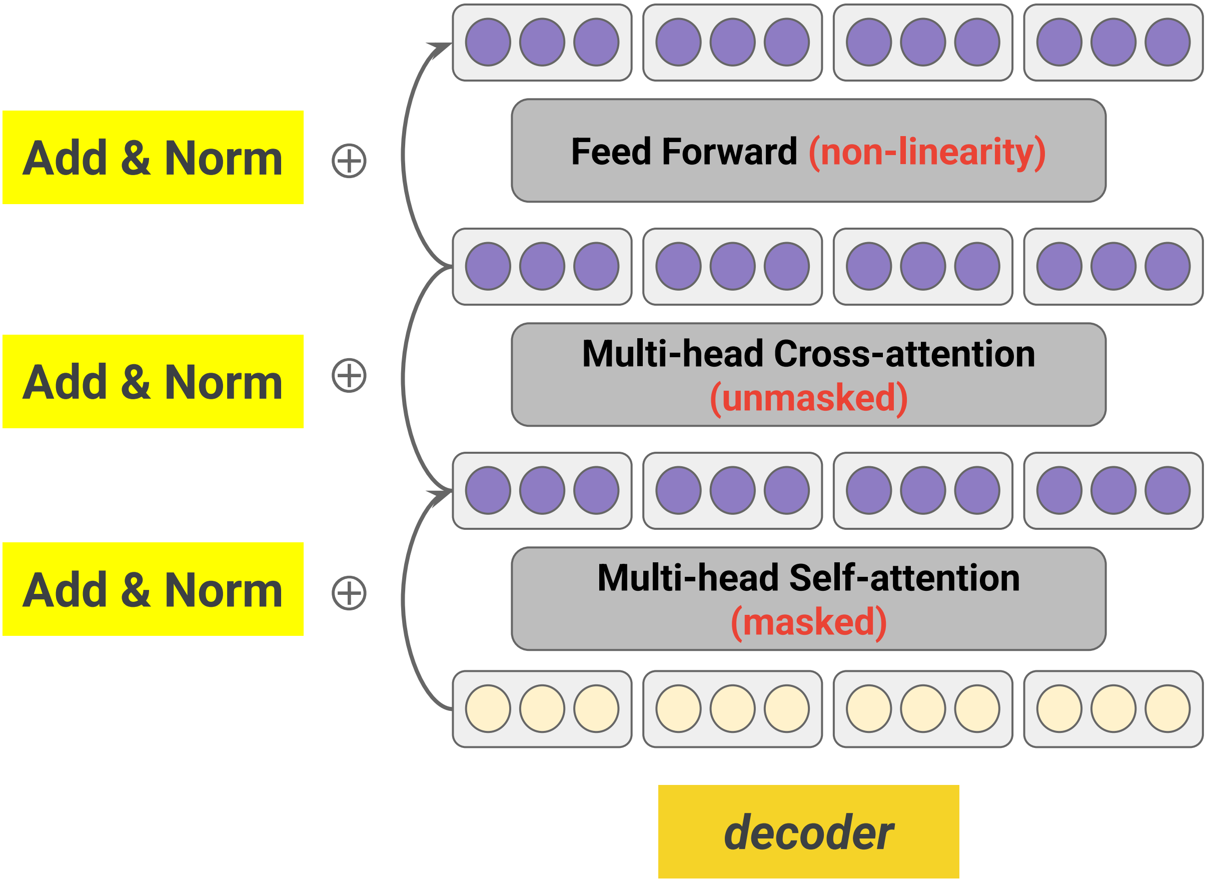
Encoder-decoder architectures



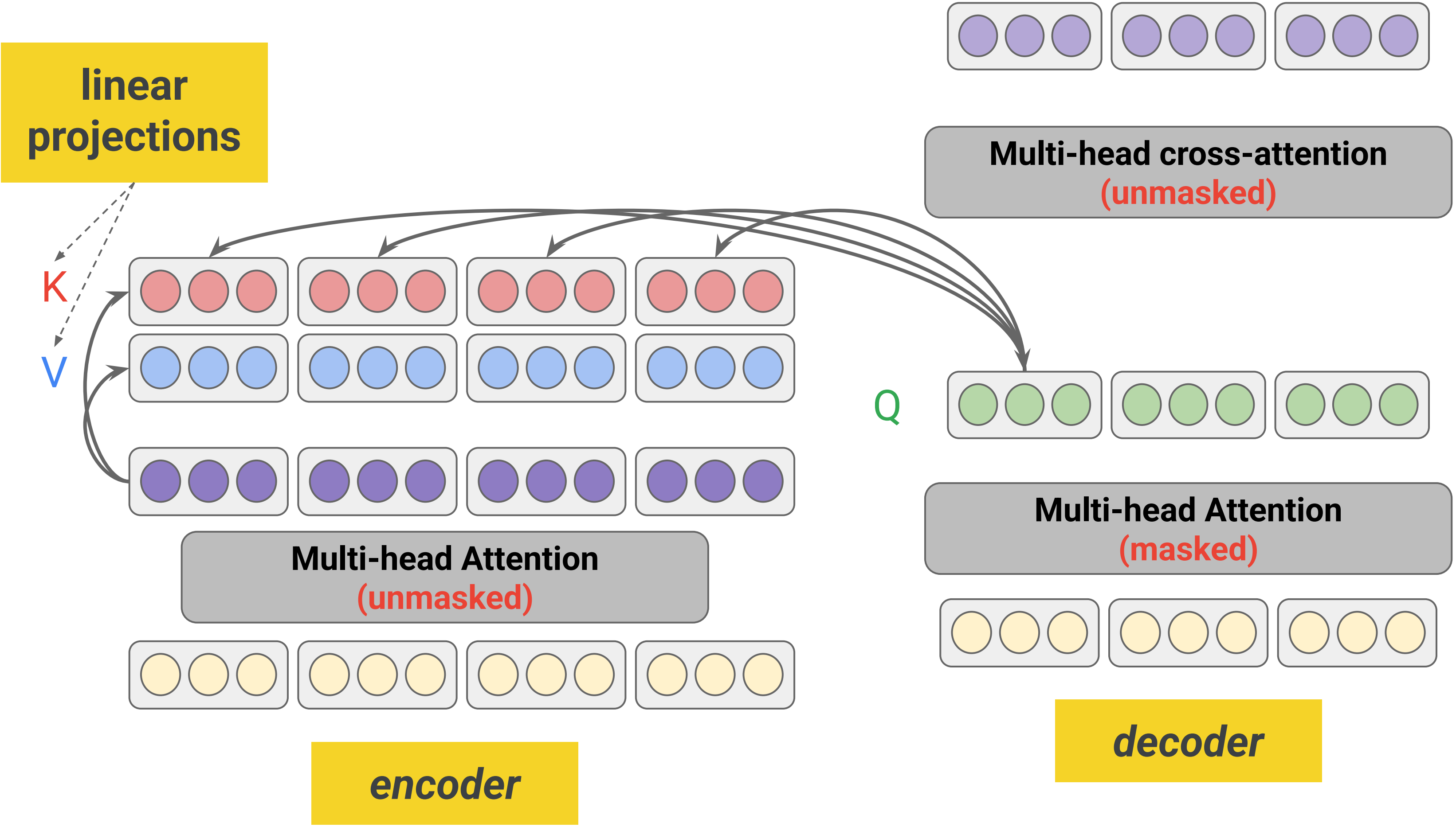
Decoder (N layers)



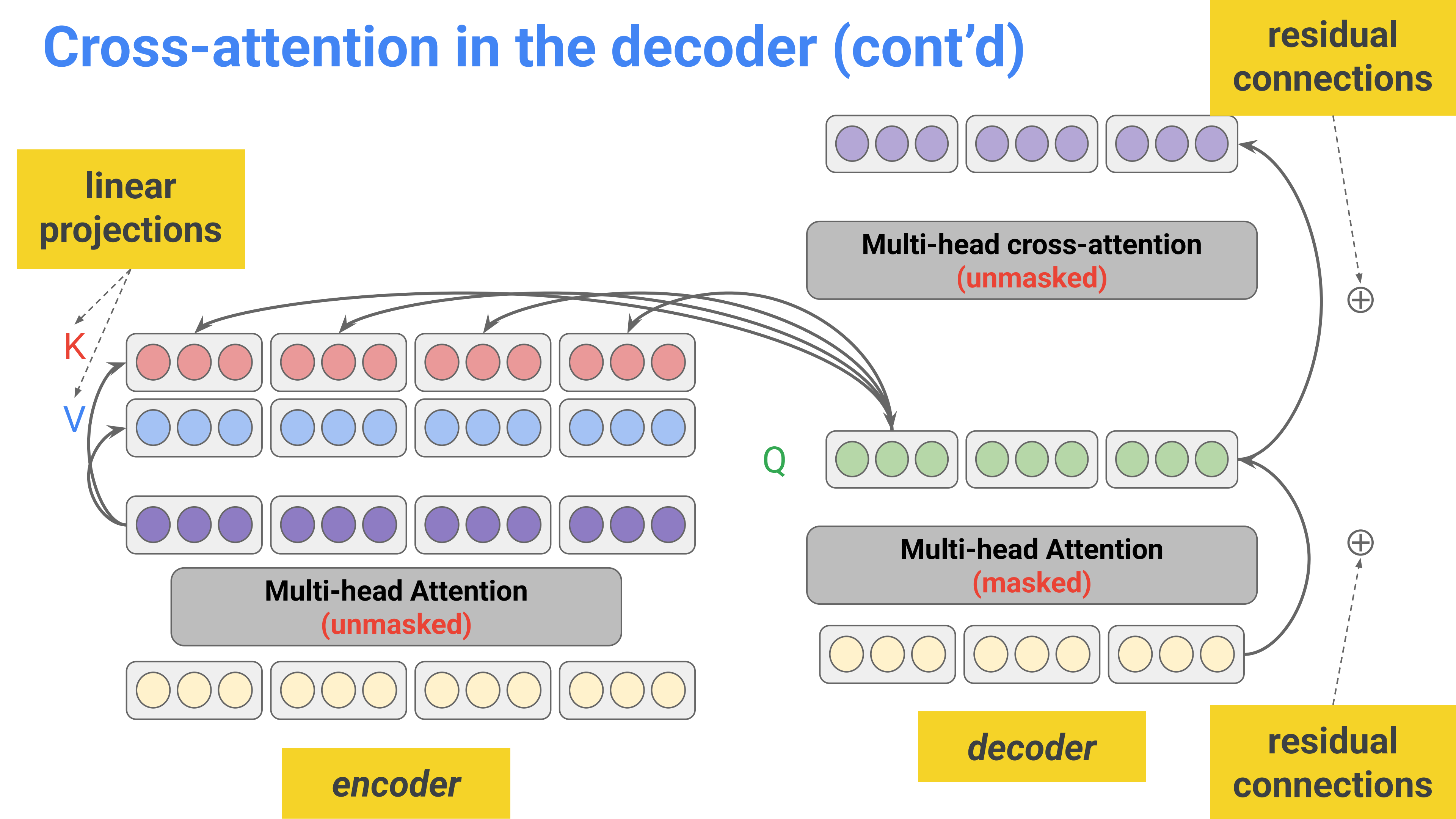
Decoder (one layer)

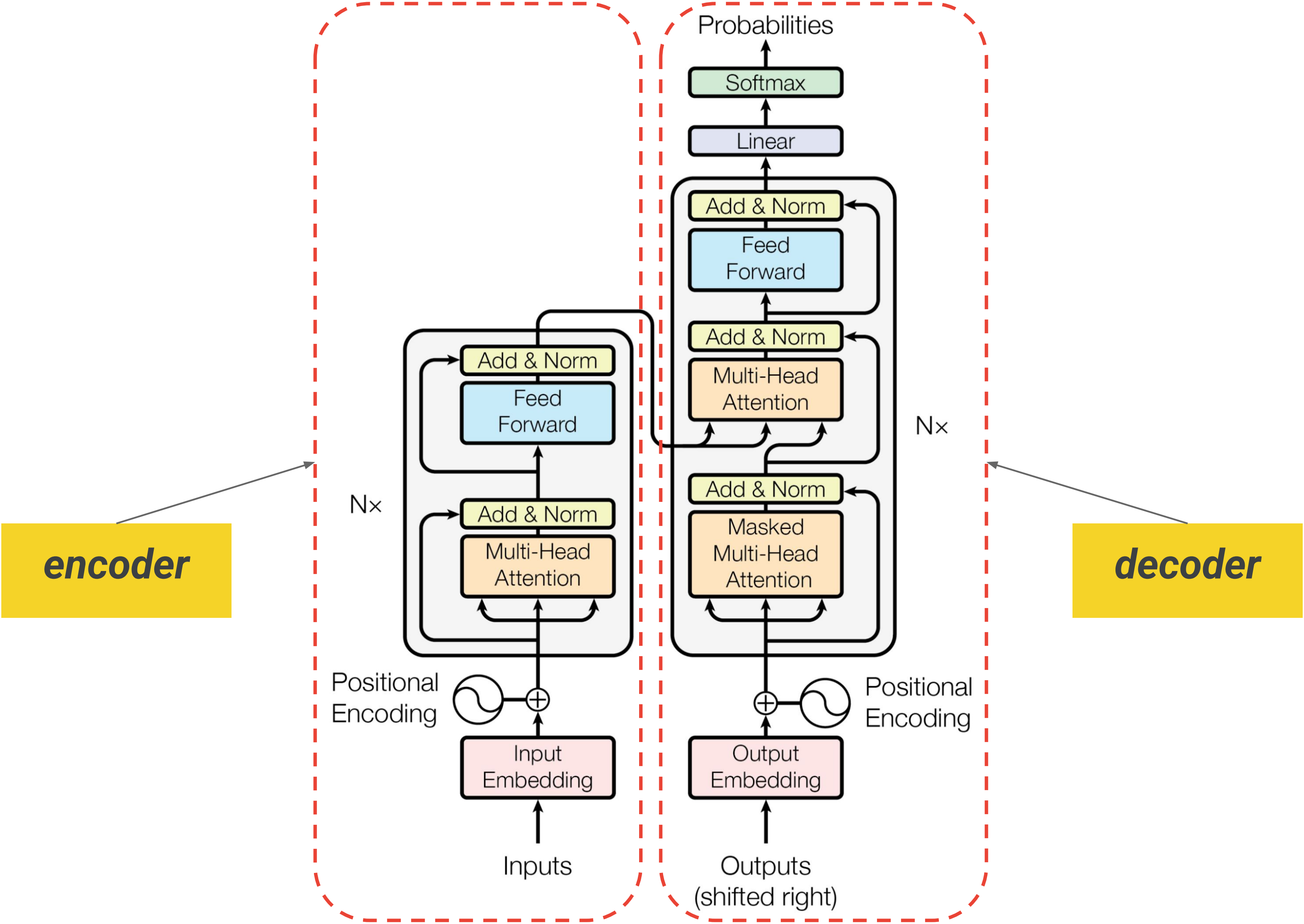


Cross-attention in the decoder



Cross-attention in the decoder (cont'd)





encoder
decoder
Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Different Transformers architectures

- Encoder-only
    - BERT
- Encoder-decoder
    - T5
- Decoder-only
    - GPT

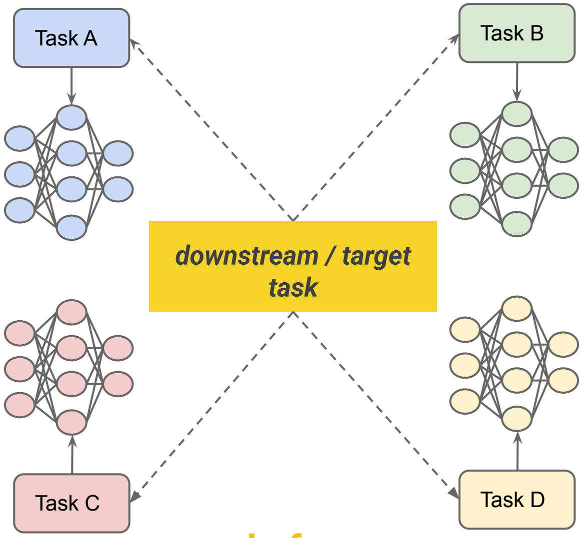
Image created by Gemini

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin   Ming-Wei Chang   Kenton Lee   Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

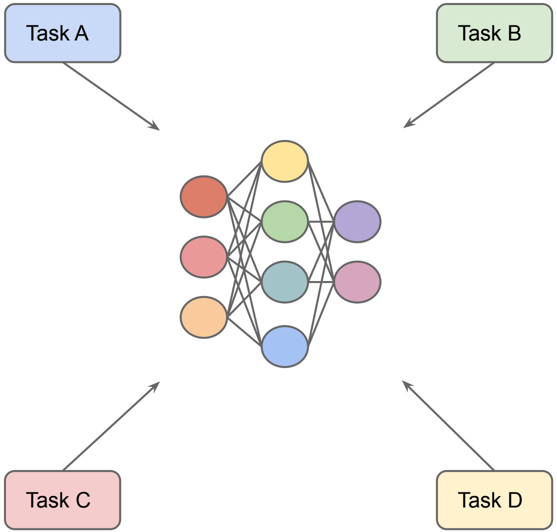
A learning paradigm shift

training task-specific models from scratch



before BERT

pretraining and then adapting



since BERT

ELMo

Image created by Gemini

Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

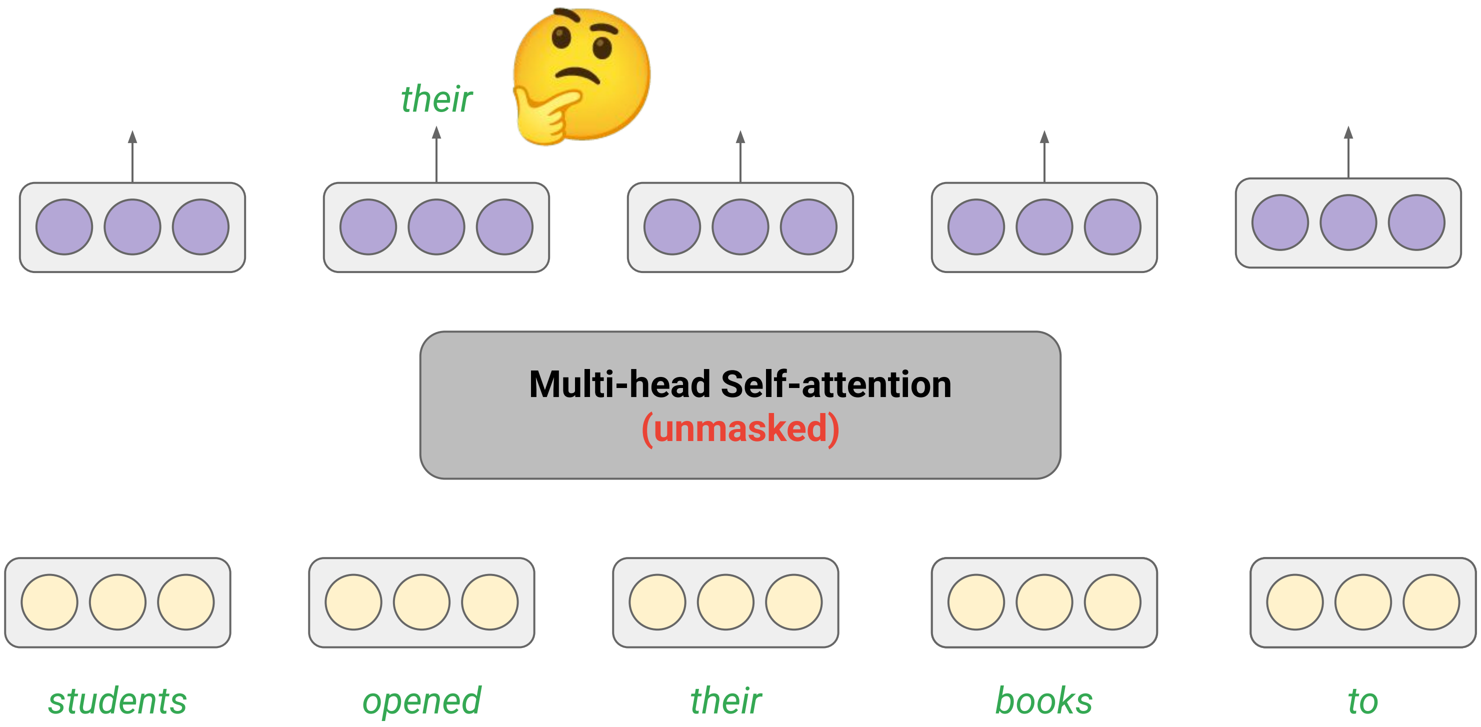
[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

BERT vs. ELMo

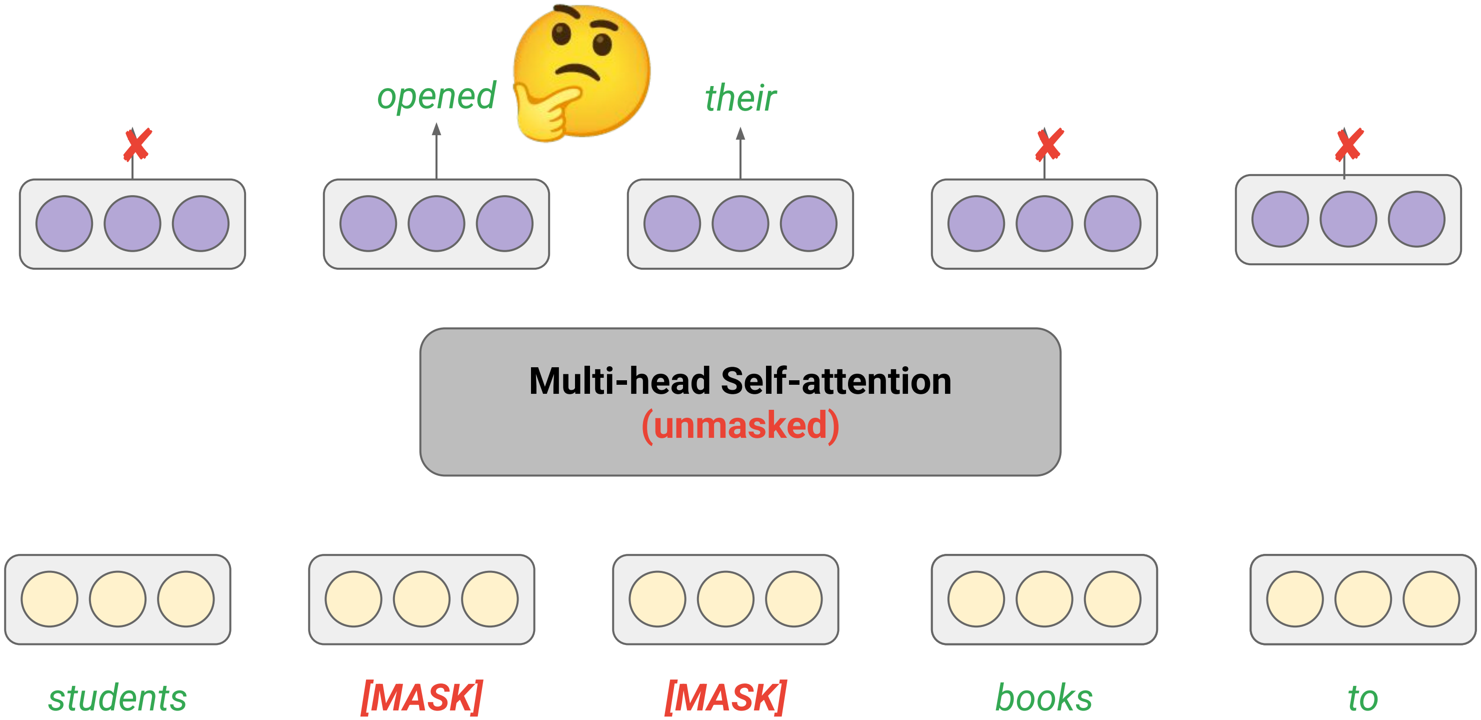
	BERT	ELMo
Model	Transformers	Bidirectional LSTM (Long Short-Term Memory, a variant of RNN)
Pre-training objective(s)	Masked language modeling + next sentence prediction	Left-to-right language modeling
Adaptation method	Fine-tuning	Feature-based (pretrained representations as additional features to task-specific models)

Pretraining

Language modeling using a Transformer encoder

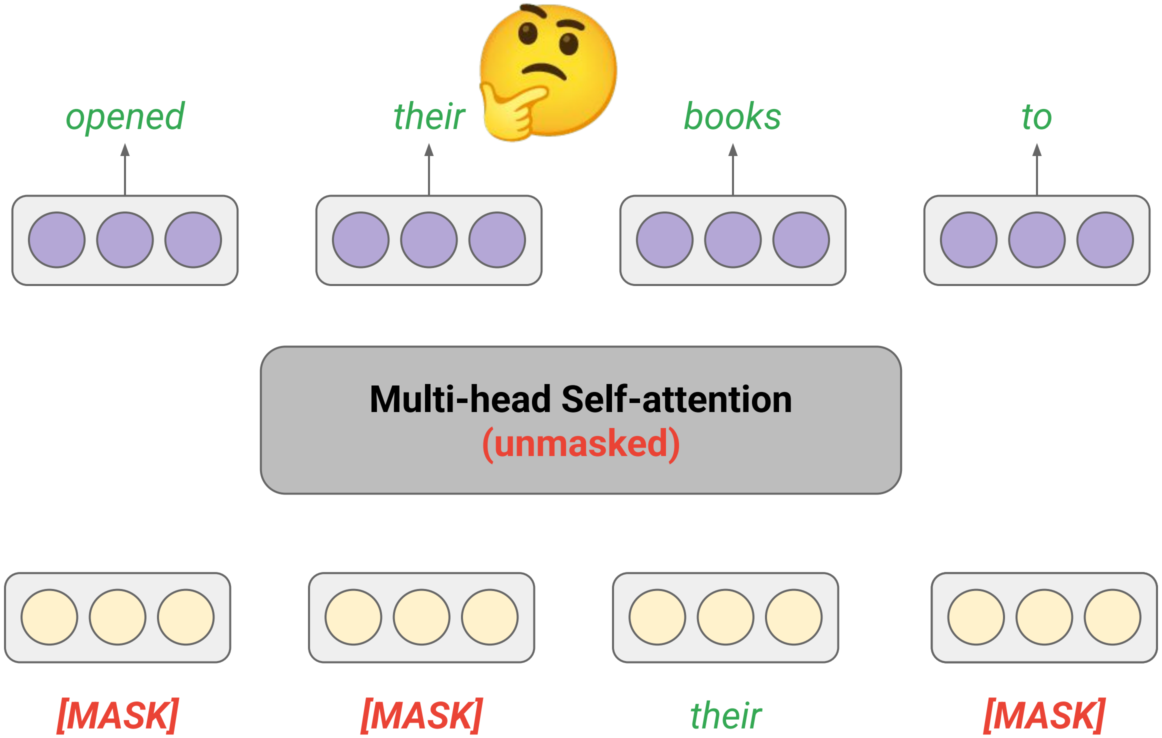


Masked language modeling



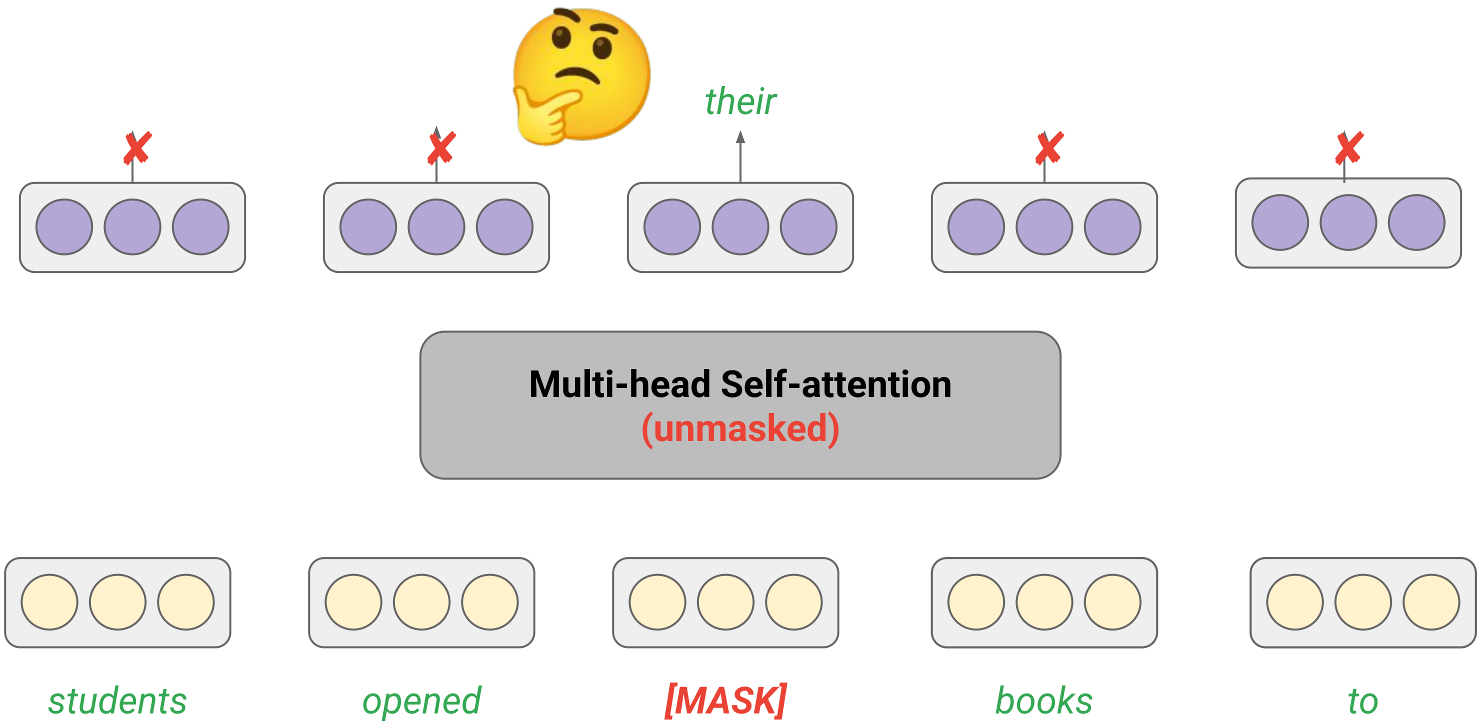
15% - 30% of all tokens in each sequence are masked at random

What if we mask more tokens?



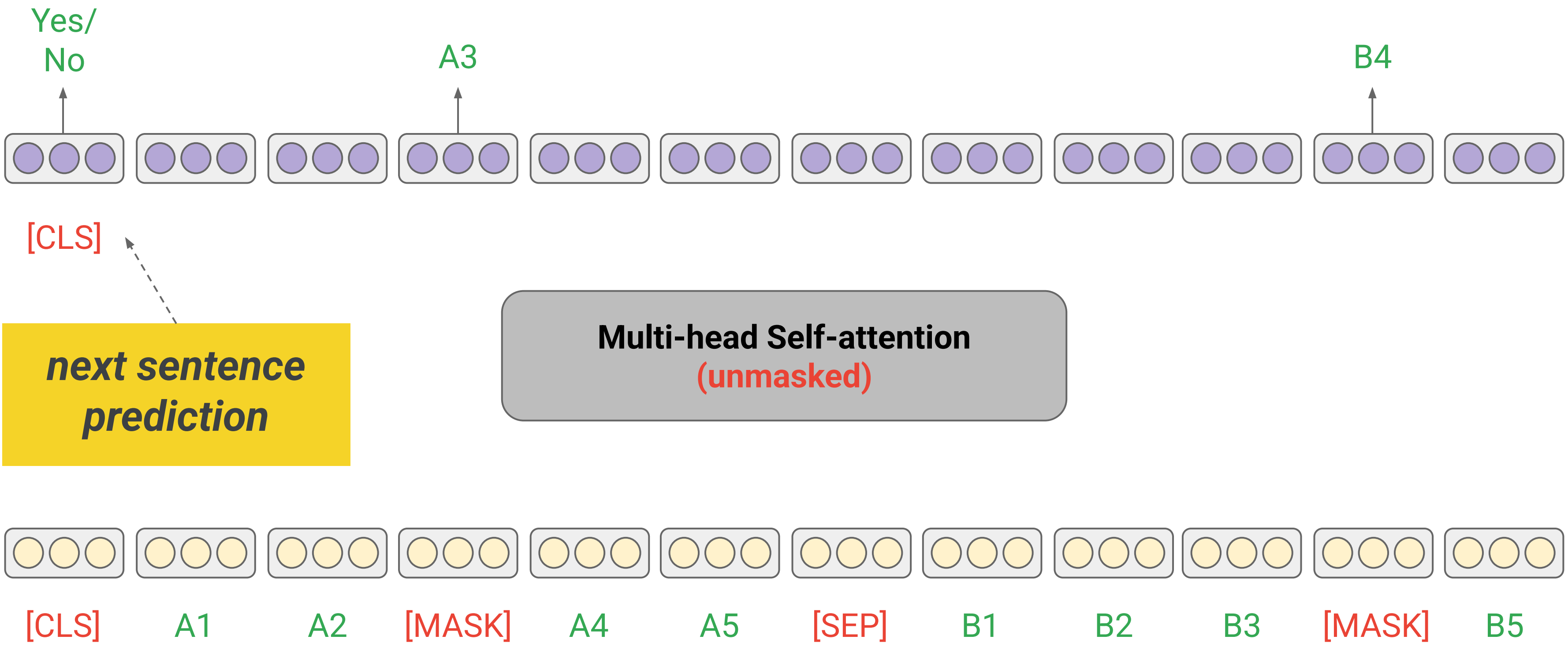
15% - 30% of all tokens in each sequence are masked at random

What if we mask less tokens?

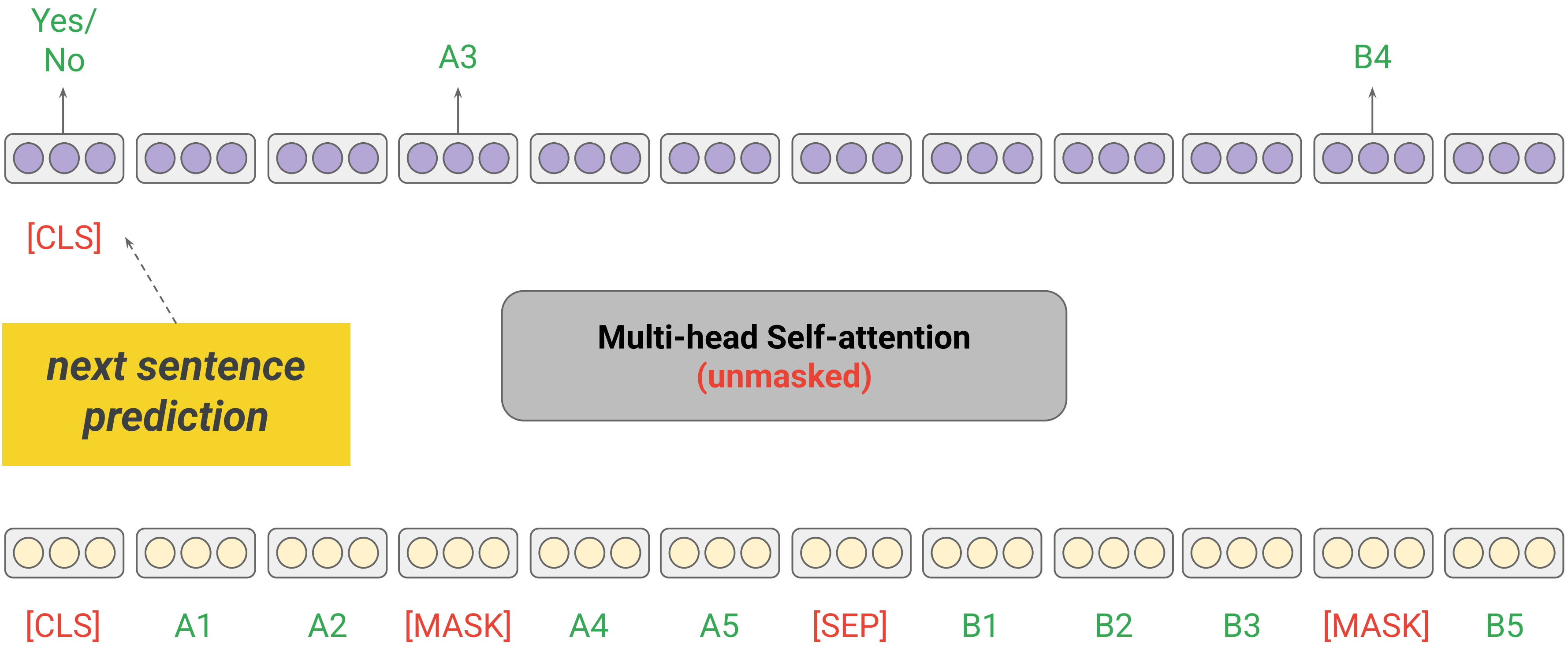


15% - 30% of all tokens in each sequence are masked at random

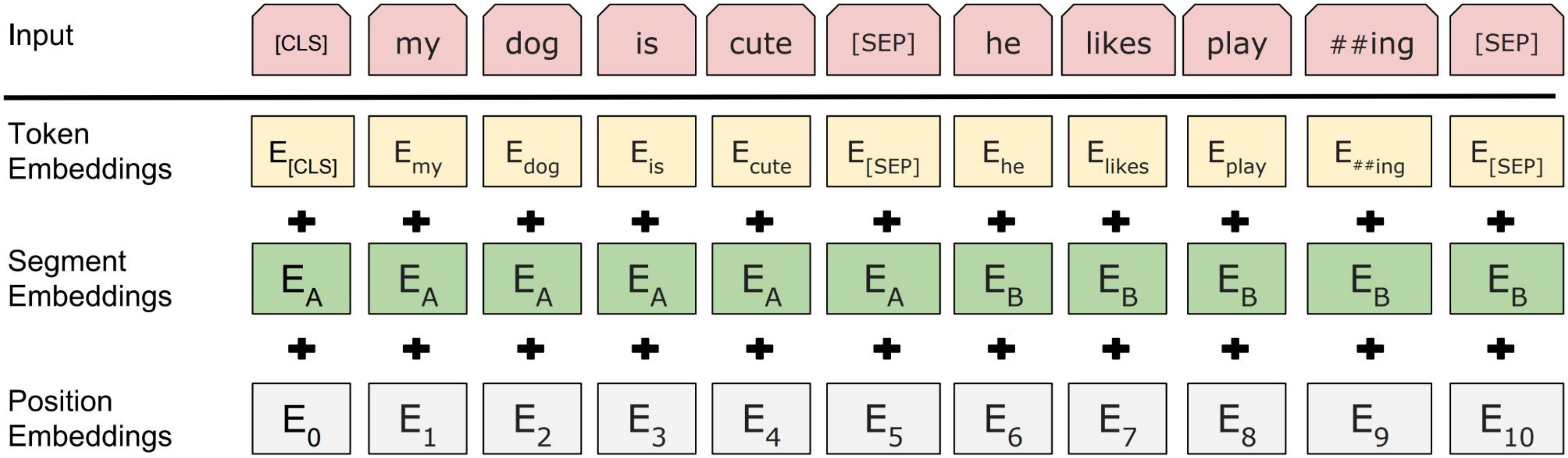
CLS & SEP tokens



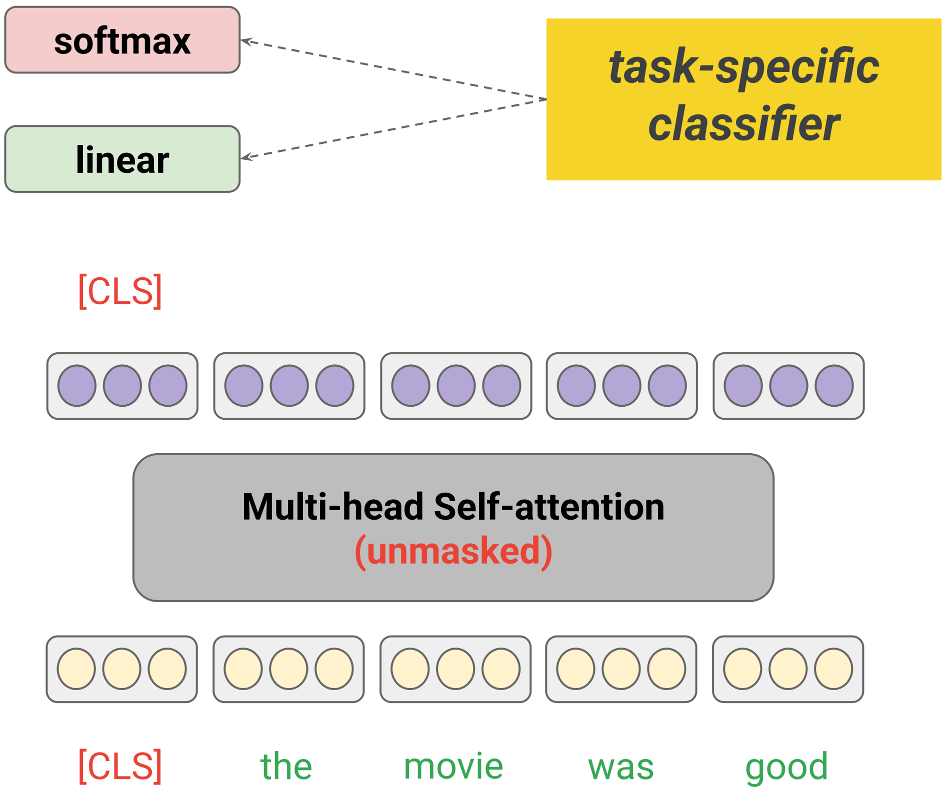
BERT Pretraining



BERT input representation

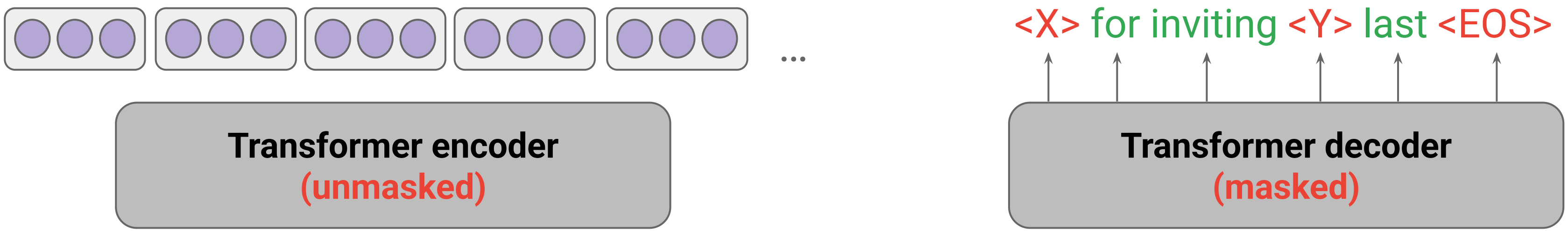


BERT Fine-tuning



T5 Pretraining: Span corruption

<X>, <Y>: sentinel tokens



Thank you <X> me to your party <Y> week

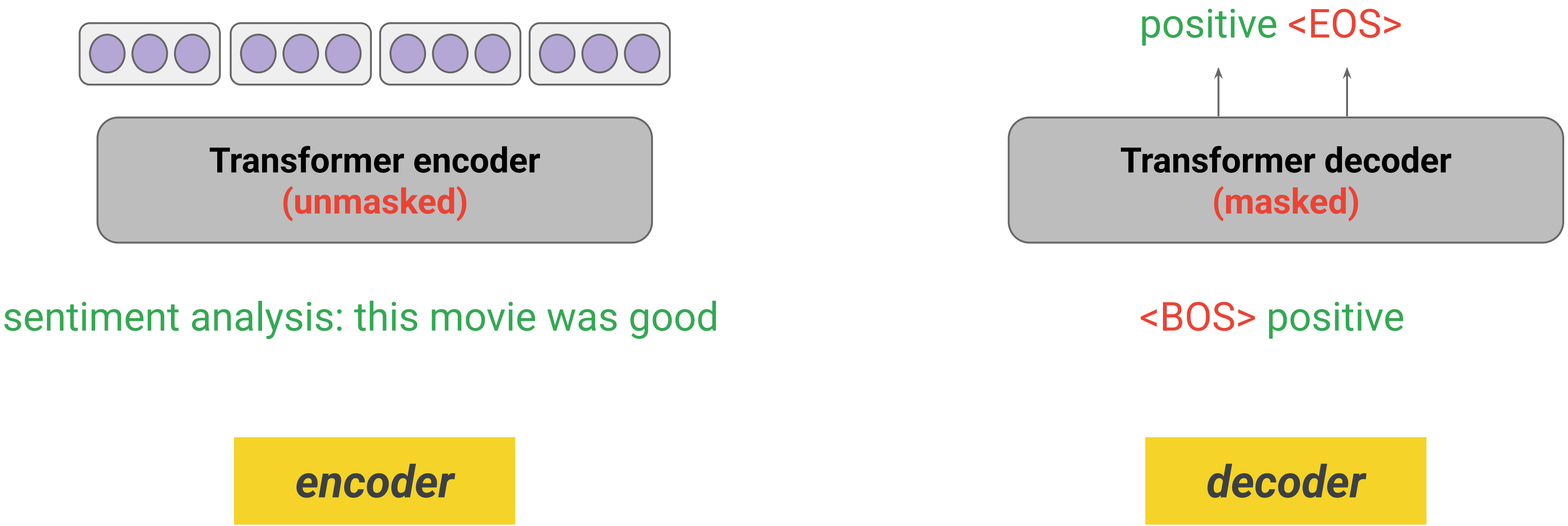
<BOS> <X> for inviting <Y> last

Thank you ~~for inviting~~ me to your party ~~last~~ week

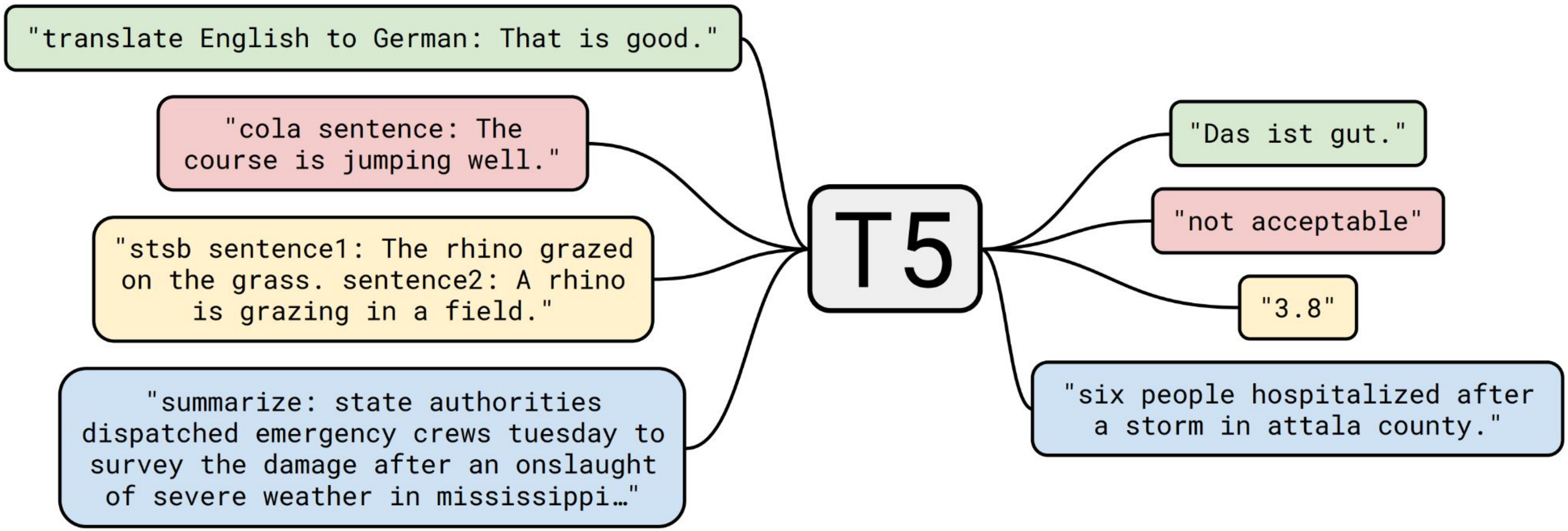
encoder

decoder

T5 Fine-tuning

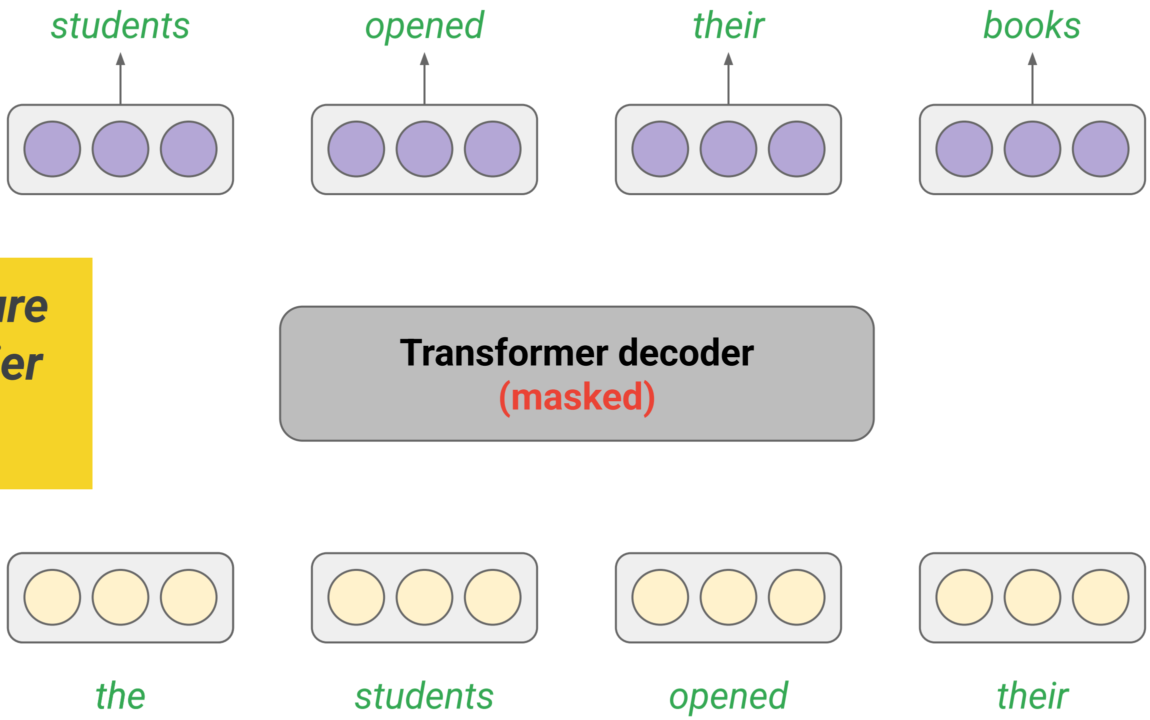


T5 facilitates multitask learning



Decoder-only model

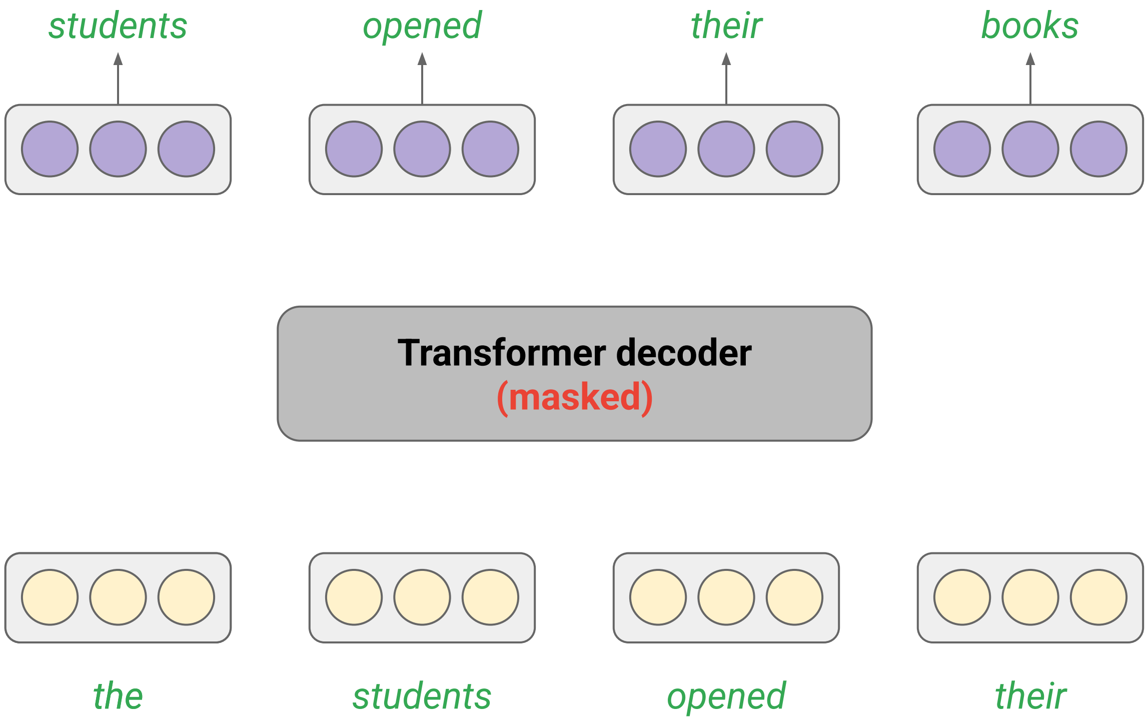
the architecture used in frontier LLMs



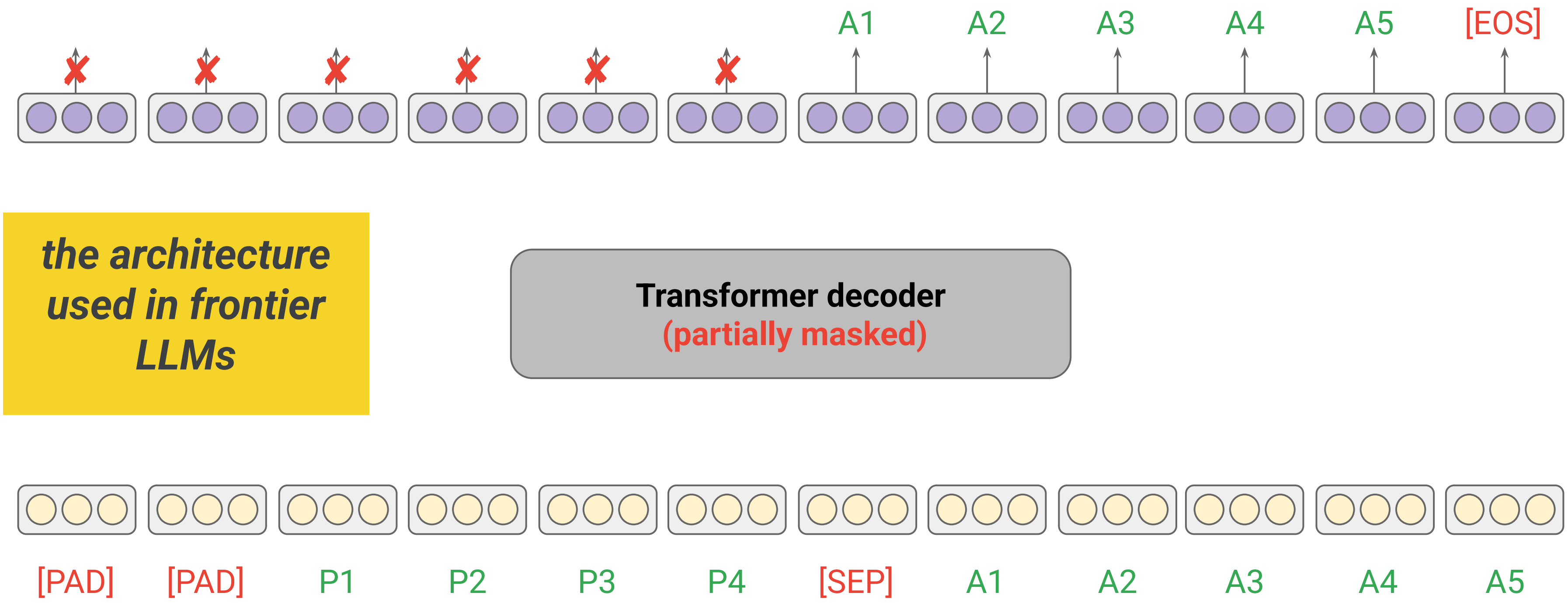
Note on cross-attention

- Can be used to inject non-text data (e.g., images, structured data, or even sensor readings) into the model

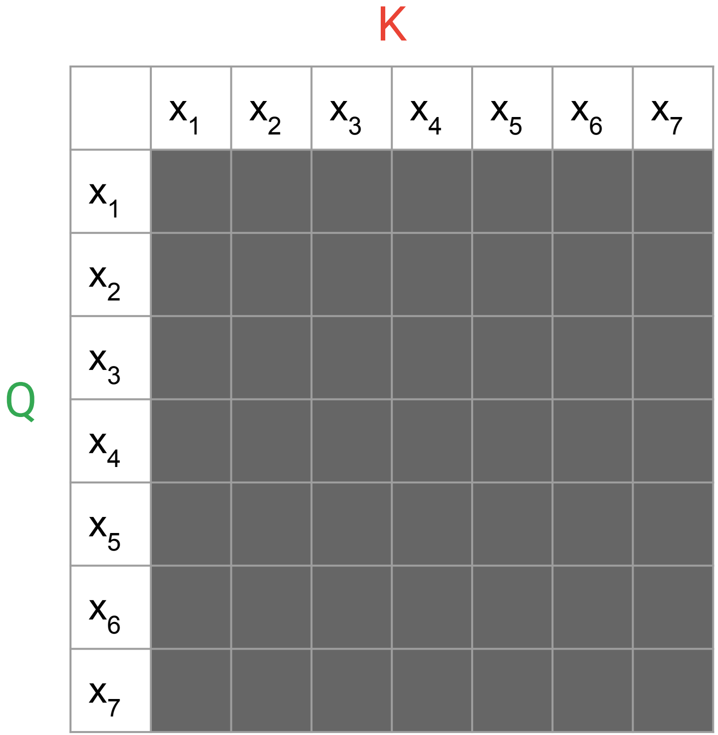
Pretraining with a causal LM (decoder-only)



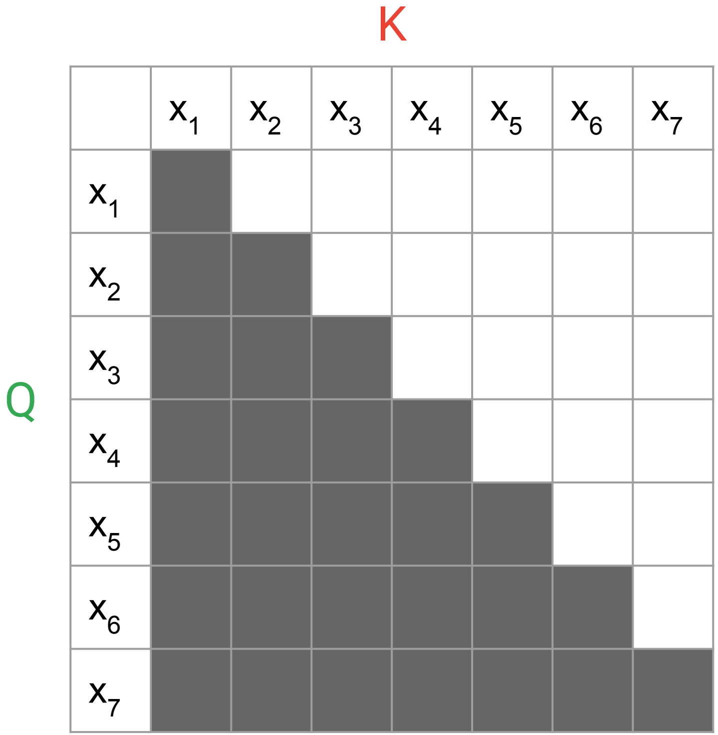
Training with prefix LM (decoder-only)



Different attention mask patterns

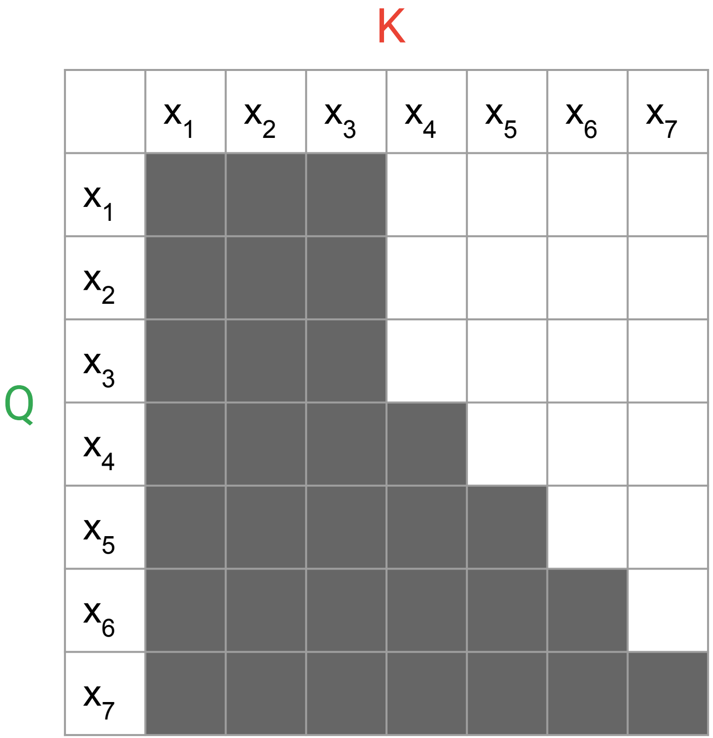


fully-visible



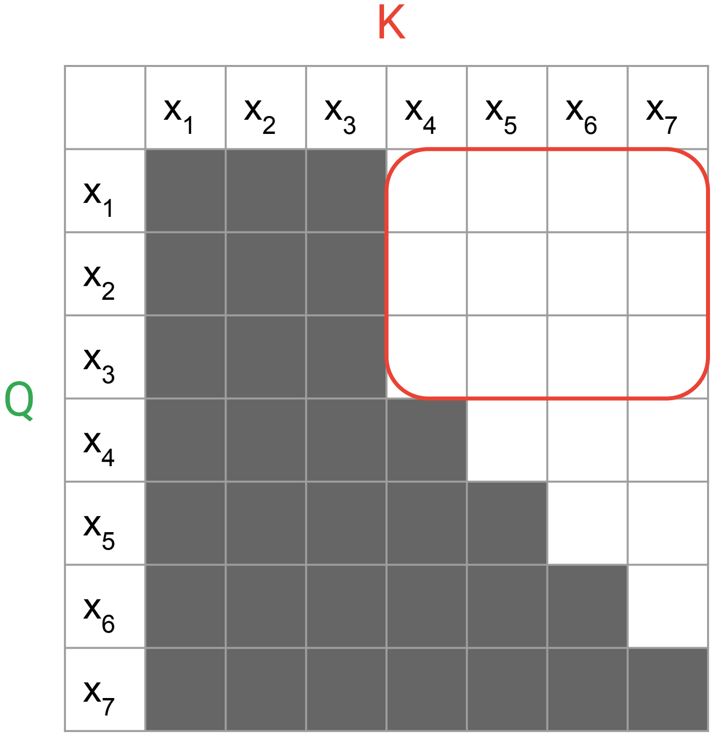
causal

Different attention mask patterns (cont'd)



Prefix LM

Different attention mask patterns (cont'd)



Why masking here?

Thank you!