Transformers (cont'd)

CS 5624: Natural Language Processing

Fall 2026

https://tuvllms.github.io/nlp-fall-2026/

Tu Vu

VIRGINIA TECH

Logistics

- Student presentations
  - Search for teammates on Piazza/Discord or reach out to us at cs5624instructors@gmail.com

    Piazza: @5
  - Google form for submitting group information available on Piazza/Discord (due next Tuesday)

- HW0 / Quiz0 released later today

AI news

AI news

Google DeepMind
@GoogleDeepMind

Two new Gemini models are here to help scale your AI agents and secure code:

🔘 3.8 Flash: our most intelligent model yet with significant gains from 3.7 Flash across software engineering, agentic tasks, and multi-step reasoning.

🔘 3.8 Flash Cyber: our most capable cybersecurity model with frontier-level vulnerability detection and automated patching.

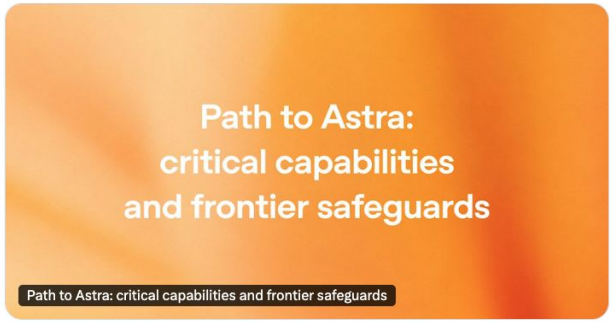
11:42 AM · Sep 2, 2026 · **399K** Views

OpenAI
@OpenAI

As we prepare to release Astra, we're focused on making increasingly capable AI safe and broadly accessible.

Astra represents a significant advance in cybersecurity capability, reaching the Critical threshold under our Preparedness Framework.

We're previewing how we evaluated the model, how its safeguards have advanced alongside its capabilities, and what we'll continue to learn and improve.



From openai.com

AI at Meta
@AIatMeta

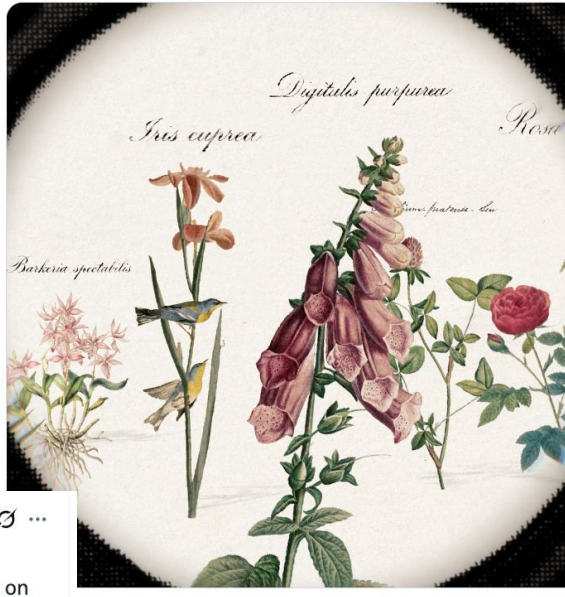
We're excited to release Muse Spark 1.3 with improved performance on agentic and coding tasks, and a focus on real-world usability.

Key capabilities:
→ Sustains longer-horizon work across multiple workflows in a single thread
→ More actively collaborates with users: it asks clarifying questions, flags when it's stuck, confirms before consequential actions
→ Better calibrated on its own limits instead of hallucinating outcomes
→ ~20% fewer tool calls and ~25% fewer tokens vs. Muse Spark 1.2 in internal comparisons

Claude
@claudeai

We're introducing Claude Fable 5.1 and Claude Mythos 5.1.

They're the world's most advanced models for coding and knowledge work.



Sep 1, 2026 · **19.2M** Views

https://arena.ai/leaderboard/agent

Agent Arena

Overall

Dynamic ranking of models on how well they orchestrate tools for real-world agentic tasks, based on signals like tool reliability, task completion, and steerability.

Sep 1, 2026 | 2,188,416 sessions | 58 models

Hide Filters

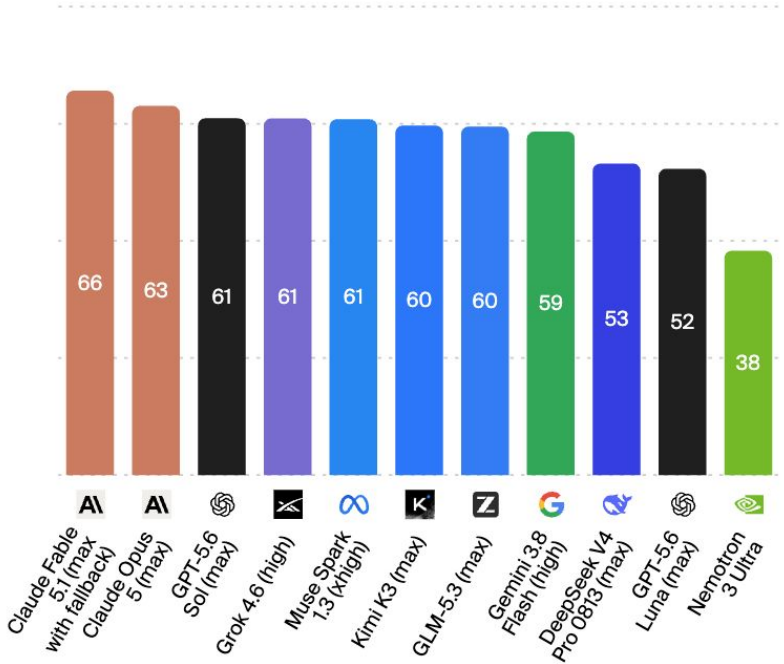
Models | Labs

Rank	Model	Net Improvement	Confirmed Success	Praise vs Complaint	Steerability
1 1 ⇿ 3	Claude Opus 5 (High) Anthropic · Proprietary	▲ 13.74% ±1.80%	▲ 14.96% ±3.73%	▲ 21.82% ±6.81%	▲ 16.00% ±3.26%
2 1 ⇿ 5	Claude Opus 5 (Max) Anthropic · Proprietary	▲ 11.69% ±2.01%	▲ 15.49% ±4.03%	▲ 18.35% ±7.36%	▲ 8.30% ±3.98%
3 1 ⇿ 7	Claude Fable 5 (High) Anthropic · Proprietary	▲ 10.61% ±1.53%	▲ 6.81% ±3.30%	▲ 20.25% ±5.49%	▲ 13.23% ±2.77%
4 2 ⇿ 8	GPT 5.6 Sol (xHigh) OpenAI · Proprietary	▲ 9.49% ±1.51%	▲ 7.57% ±3.20%	▲ 21.62% ±5.62%	▲ 8.73% ±2.88%
5 2 ⇿ 11	Claude Opus 4.8 (High) Anthropic · Proprietary	▲ 9.22% ±1.53%	▲ 5.38% ±3.11%	▲ 18.87% ±5.30%	▲ 12.06% ±2.83%
6 3 ⇿ 8	Kimi K3 (Max) Moonshot · Kimi K3 license	▲ 8.71% ±0.66%	▲ 16.90% ±1.35%	▲ 15.85% ±2.31%	▲ 2.13% ±1.29%
7 4 ⇿ 17	GPT 5.5 (xHigh) OpenAI · Proprietary	▲ 7.53% ±1.08%	▲ 2.45% ±2.36%	▲ 12.23% ±3.75%	▲ 8.55% ±1.99%

https://artificialanalysis.ai/

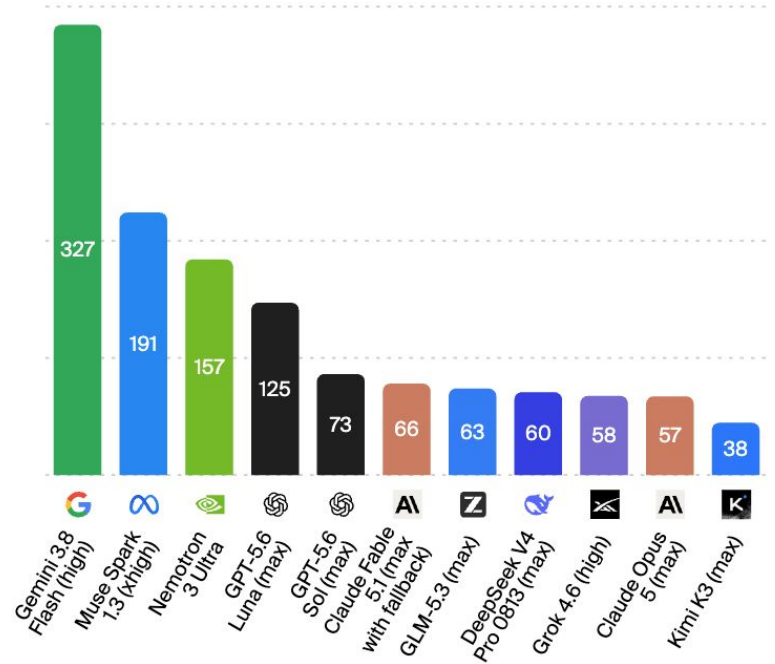
Intelligence

Artificial Analysis Intelligence Index · Higher is better



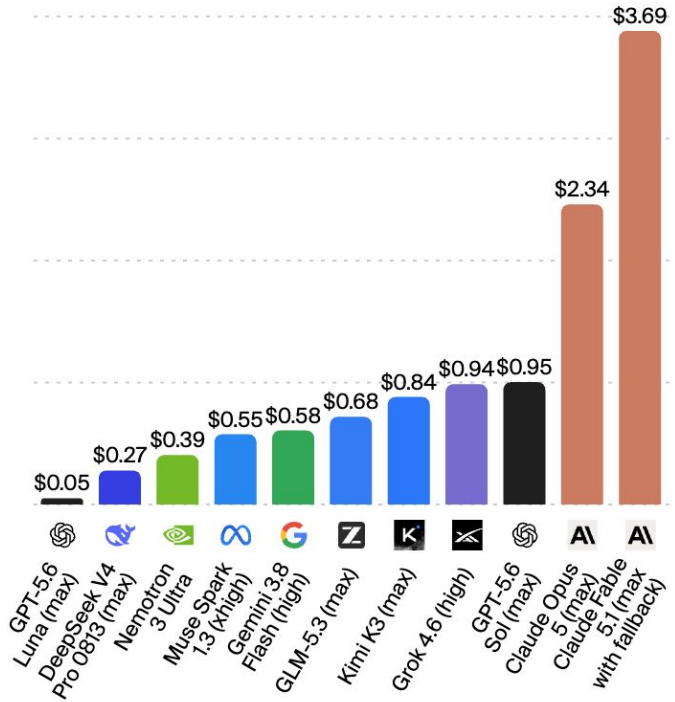
Speed

Output tokens per second · Higher is better



Cost per Task

Weighted average cost (USD) per Intelligence Index task · Lower is better



openrouter

The Unified Interface For Every Model

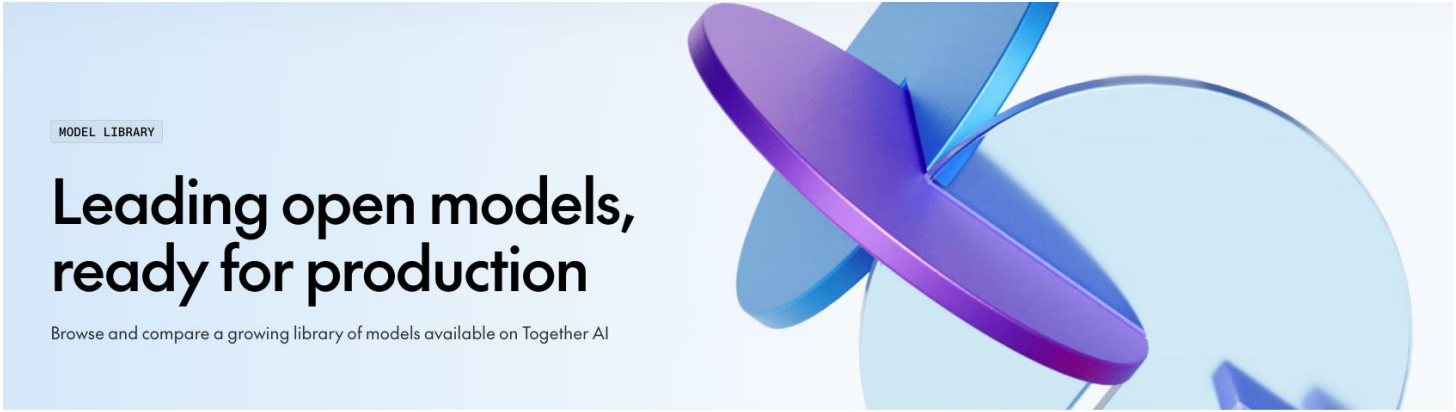
Better prices, better uptime, no subscriptions.

Get API Key

Discover Models

300T+	10M+	80+	500+
Monthly Tokens	Global Users	Providers	Models

together.ai



MODEL LIBRARY

Leading open models, ready for production

Browse and compare a growing library of models available on Together AI

ALL | CHAT | IMAGE | VISION | VIDEO | AUDIO | TRANSCRIBE | CODE | EMBEDDINGS | RERANK | MODERATION

Find a model...

Showing 190 models

ALL PROVIDERS

CHAT
MiniMax M3

Cached/1M	Input/1M	Output/1M
$0.06	$0.30	$1.20

FUNCTION CALLING • JSON MODE

CHAT
Kimi K3

Cached/1M	Input/1M	Output/1M
$0.30	$3.00	$15.00

JSON MODE • FUNCTION CALLING

CHAT
DeepSeek V4 Flash 0731

Cached/1M	Input/1M	Output/1M
$0.03	$0.14	$0.28

CHAT
Qwen3.8-2.4T-A95B

Cached/1M	Input/1M	Output/1M
$0.25	$2.00	$6.00

FUNCTION CALLING • JSON MODE

Cheaper Inference

Save up to 60% on AI models

Access leading discounted AI models from multiple providers through one OpenAI-compatible API, **without changing your request format.**

Start saving →

Talk to sales

Fund from $5 · Usage-based pricing · No monthly commitment

Advanced Research Computing (ARC) @ VT

llm.arc.vt.edu

Description

https://llm.arc.vt.edu provides a web interface to a selection of LLMs hosted and run by ARC. It is based on the Open WebUI platform, and integrates several inference and integration capabilities, including retrieval-augmented generation (RAG), web search, vision, and image generation.

Access

- All Virginia Tech students, faculty, and staff may access the service at https://llm.arc.vt.edu. No separate ARC account is required to use the web interface.
- There is no charge to individual users for accessing the hosted models via the web interface.
- Users may generate API keys through `User profile > Settings > Account > API keys`. Keys are unique to each user and must be kept confidential. Do **not** share your keys. See instructions on how to interact using API keys.

Restrictions

Data classification: Researchers can use this tool for high-risk data. This service is approved by VT Information Technology Security Office (ITSO) for processing sensitive or regulated data. However, researchers are reminded to consult with VT Privacy and Research Data Protection Program (PRDP) and the Office of Export and Secure Research Compliance regarding the storage and analysis of high-risk data to comply with specific regulations. Note that some high-risk data (e.g. data regulated by DFARS, ITAR, etc.) require additional protections and the LLM might not be approved for use with those data types.

Models

ARC currently runs several state-of-the-art models. ARC will add or remove models and scale instances dynamically to respond to user demand. You may select your preferred model.

- OpenAI `gpt-oss-120b` (see model card on Hugging Face). OpenAI's flagship open-weight model, optimized for fast, high-quality responses across a wide range of general-purpose tasks.
- Z.AI GLM 5.2 `GLM-5.3` (see model card on Hugging Face). State-of-the-art reasoning model that excels at complex problem solving, coding, mathematics, and scientific tasks.
- Moonshot AI Kimi K3 `Kimi-K3` (see model card on Hugging Face). Moonshot AI's most capable open-weight model, featuring native multimodal understanding and advanced agentic capabilities for sophisticated workflows.
- DeepSeek DeepSeek-V4-Flash `DeepSeek-V4-Flash` (version 0731) (see model card on Hugging Face). High-performance model optimized for speed and long-context processing, making it well suited for large documents, codebases, and retrieval-augmented applications.

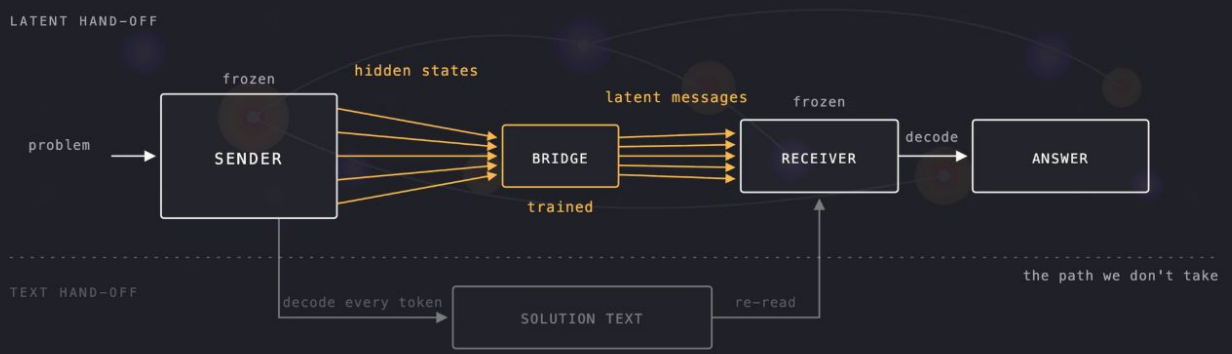
mostik.ai

Back

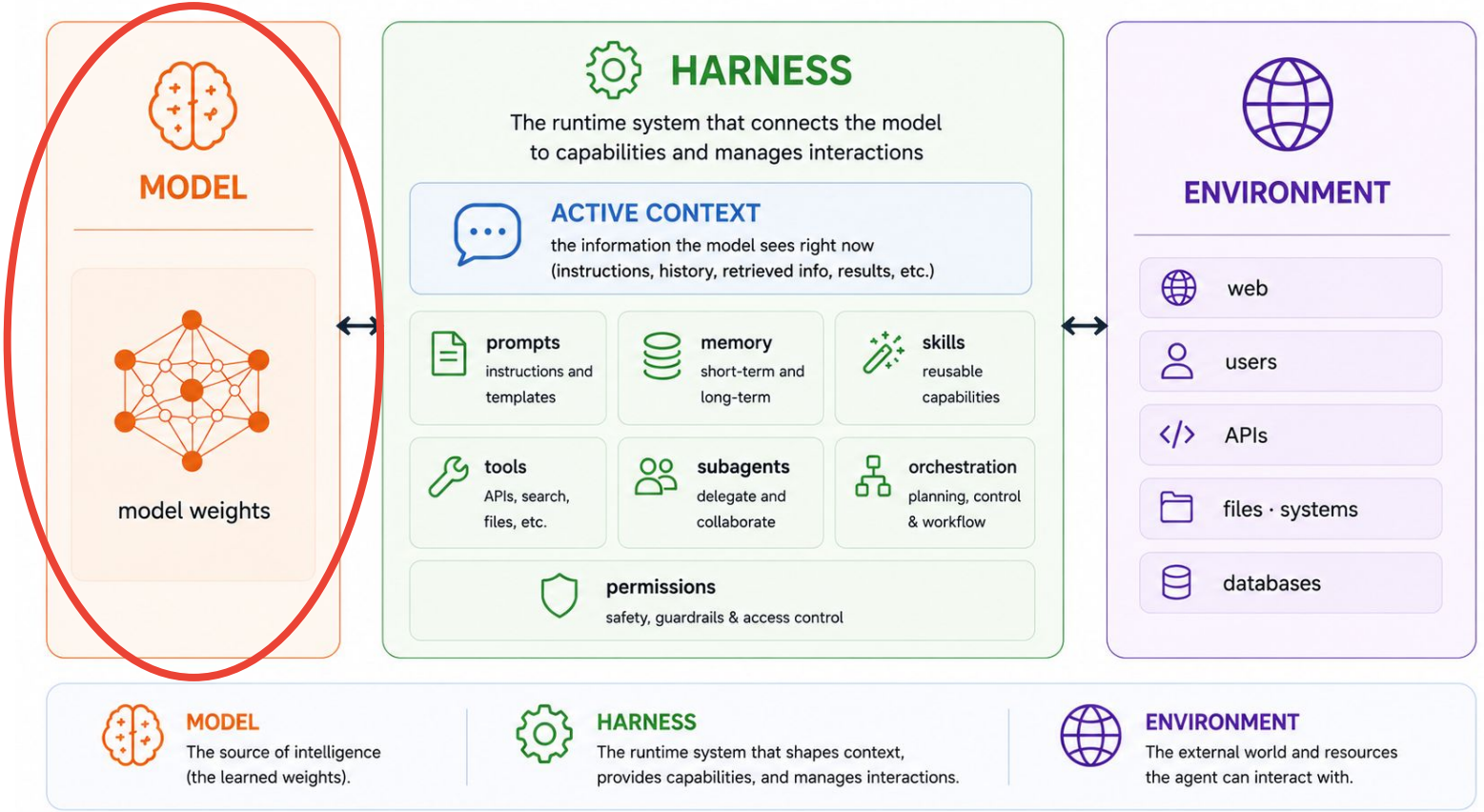
What we built

One model hands its hidden states to another through a small trained bridge, and the receiving model works with them directly. No output-type text passes between them, and neither model's weights are touched. The bridge is the only new part of the system.

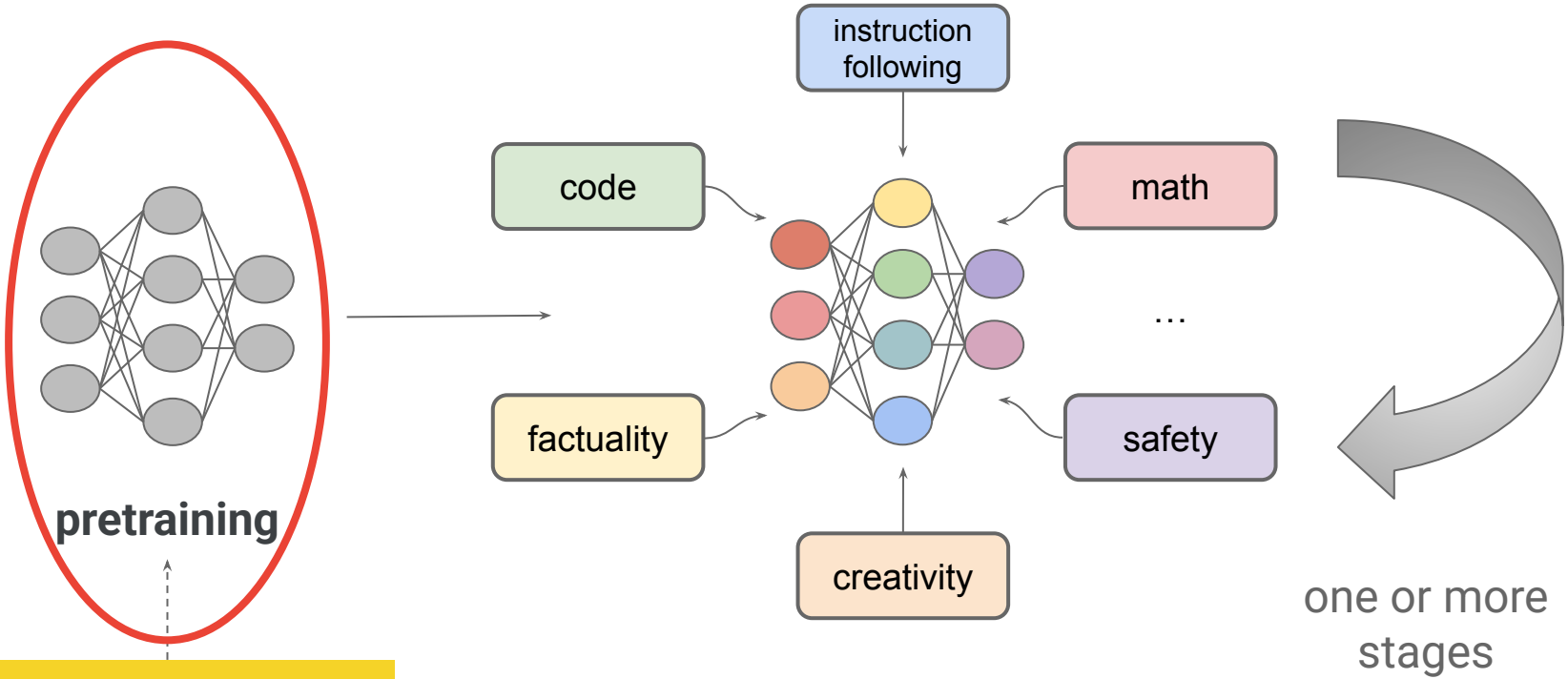
It is also the company's name: Mostik means “little bridge”.



Where are we?


MODEL
model weights
HARNESS
The runtime system that connects the model to capabilities and manages interactions
ACTIVE CONTEXT
the information the model sees right now (instructions, history, retrieved info, results, etc.)
prompts
instructions and templates
memory
short-term and long-term
skills
reusable capabilities
tools
APIs, search, files, etc.
subagents
delegate and collaborate
orchestration
planning, control & workflow
permissions
safety, guardrails & access control
ENVIRONMENT
web
users
APIs
files · systems
databases
MODEL
The source of intelligence (the learned weights).
HARNESS
The runtime system that shapes context, provides capabilities, and manages interactions.
ENVIRONMENT
The external world and resources the agent can interact with.

The development of modern LLMs



Transformers

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez*† University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

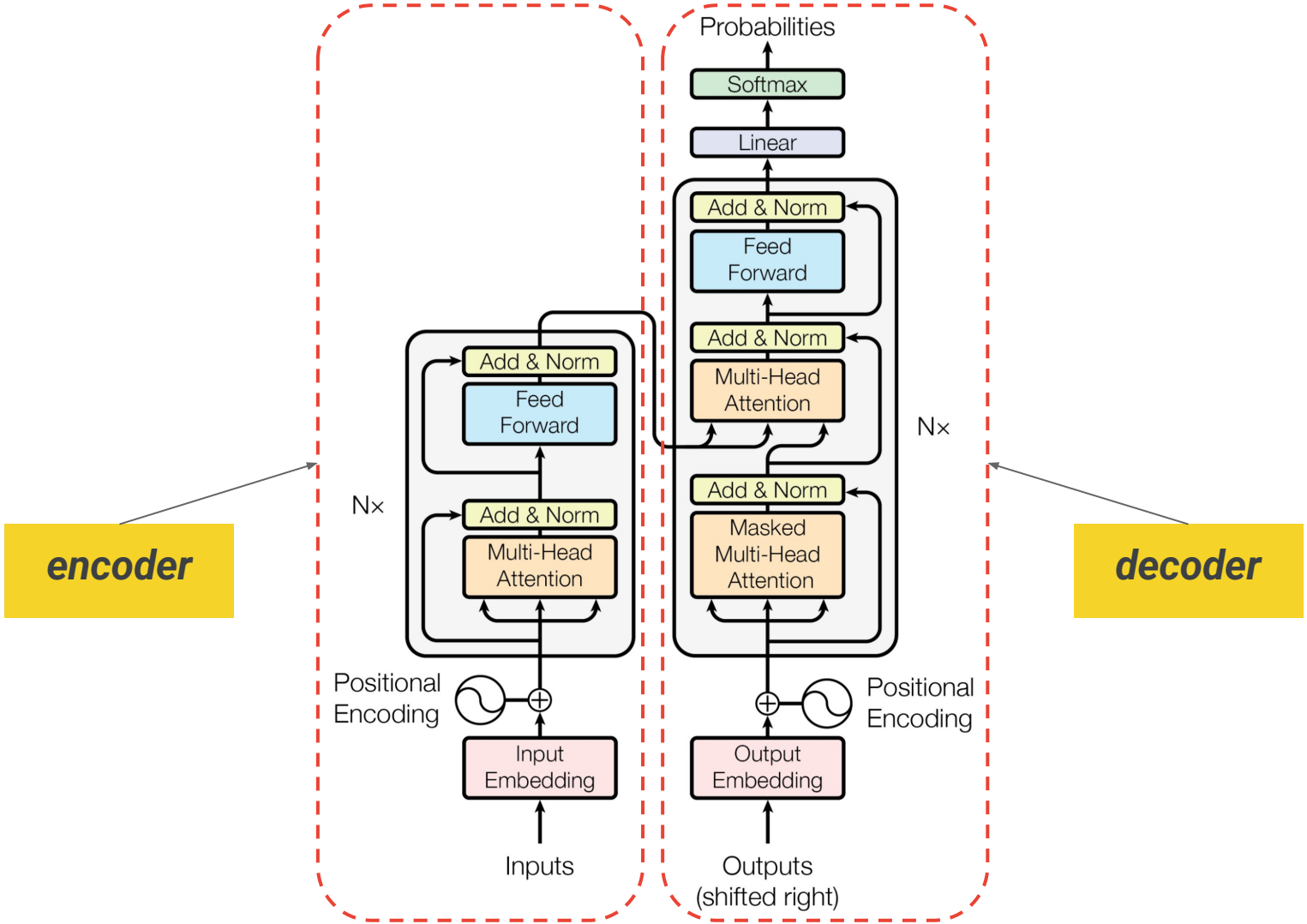
Illia Polosukhin* illia.polosukhin@gmail.com

Abstract

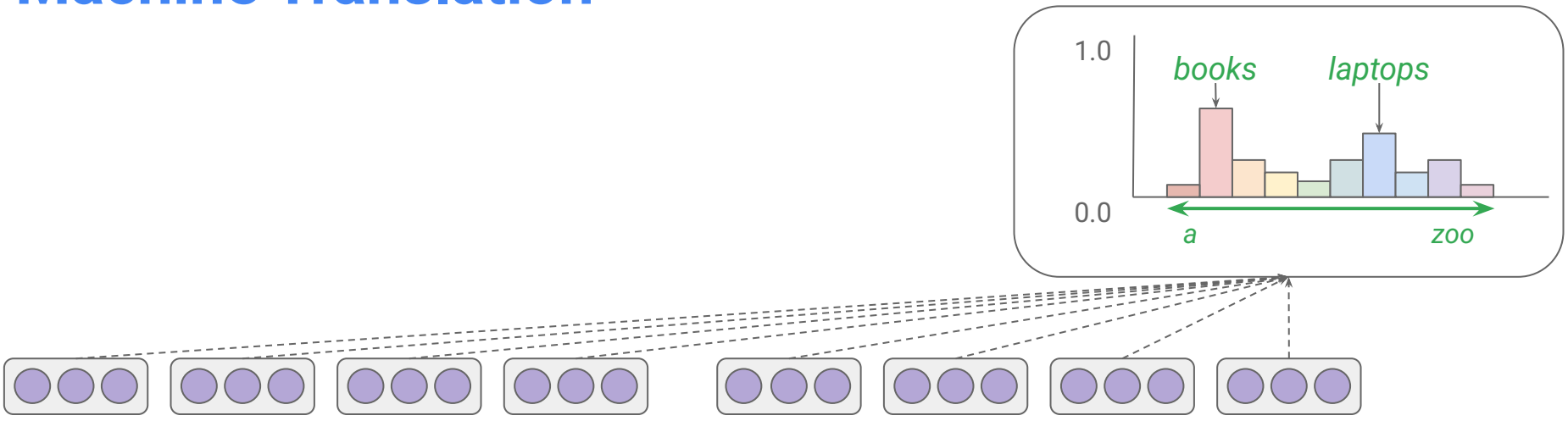
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Different Transformers architectures

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


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

Machine Translation

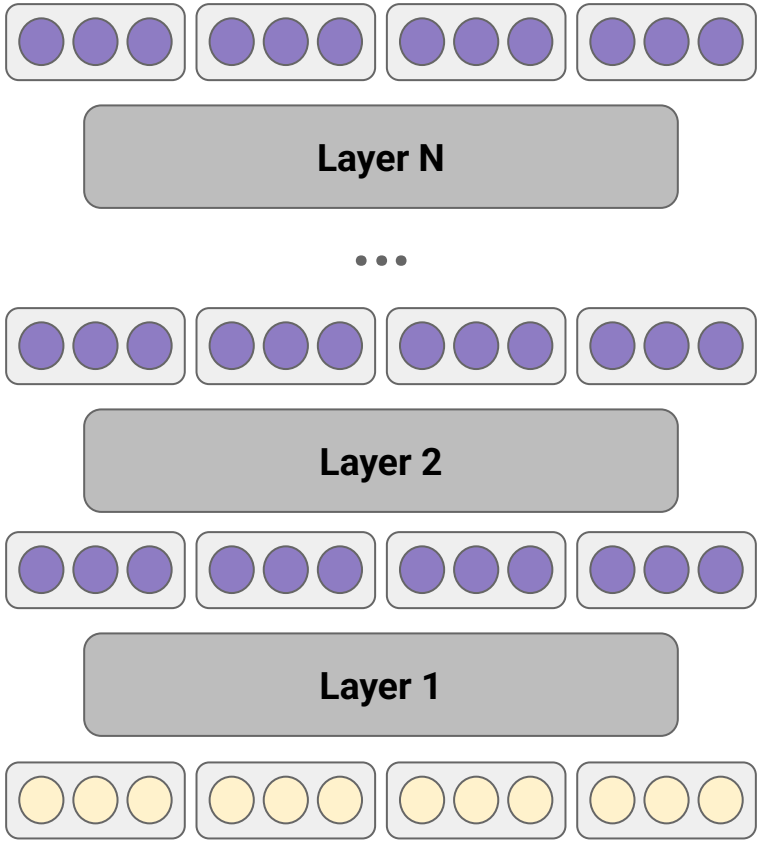


les étudiants ont ouvert leurs livres

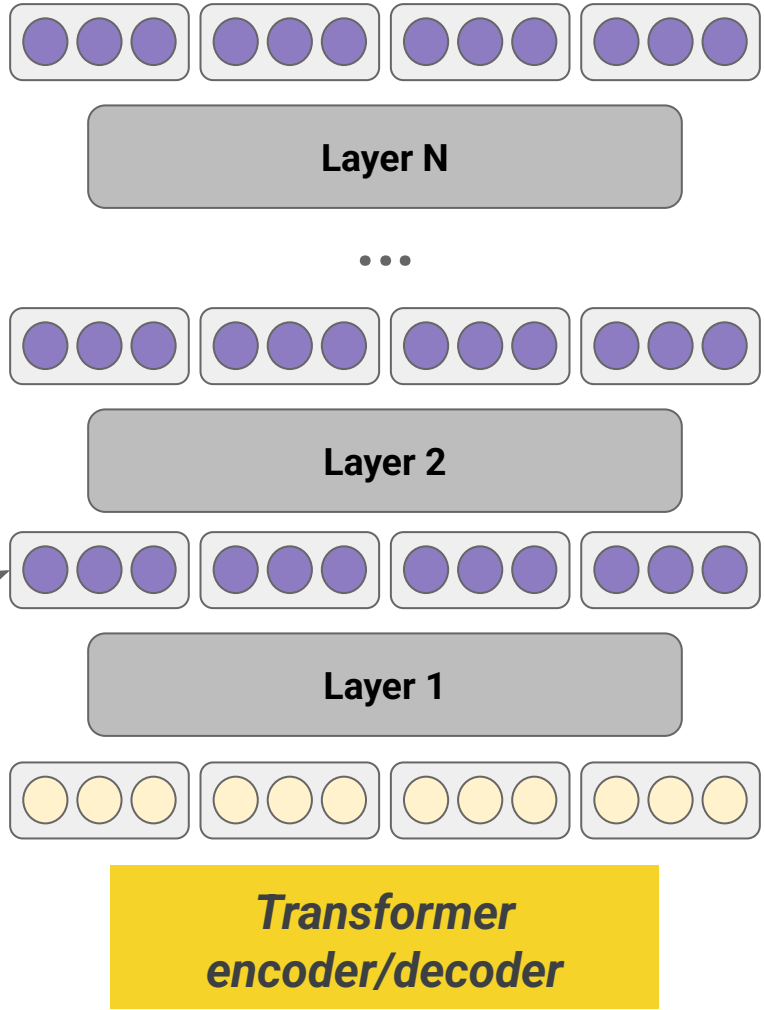
the *students* *opened* *their*

encoder ***decoder***

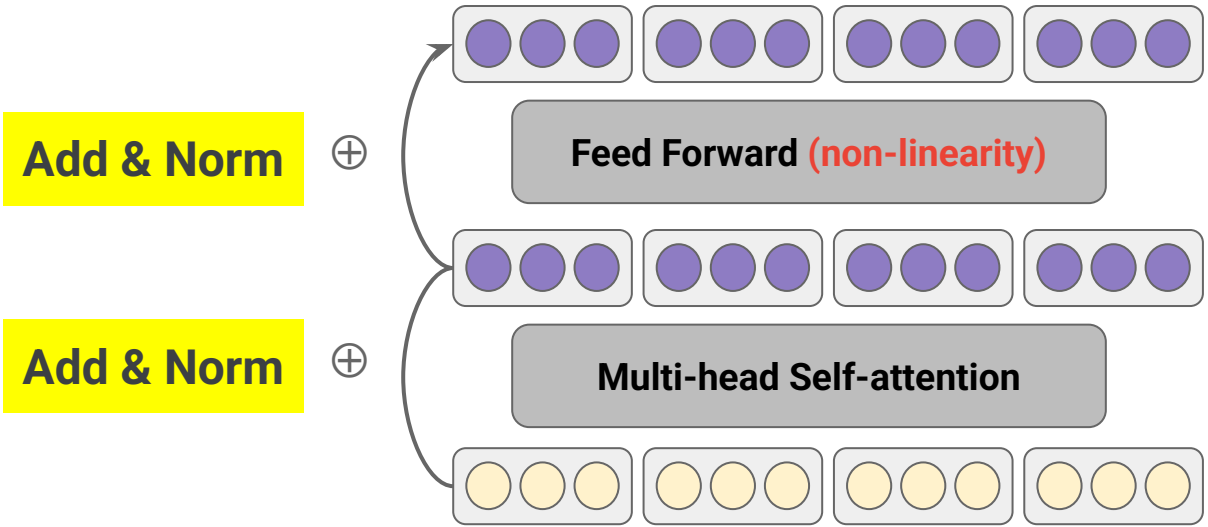
N layers



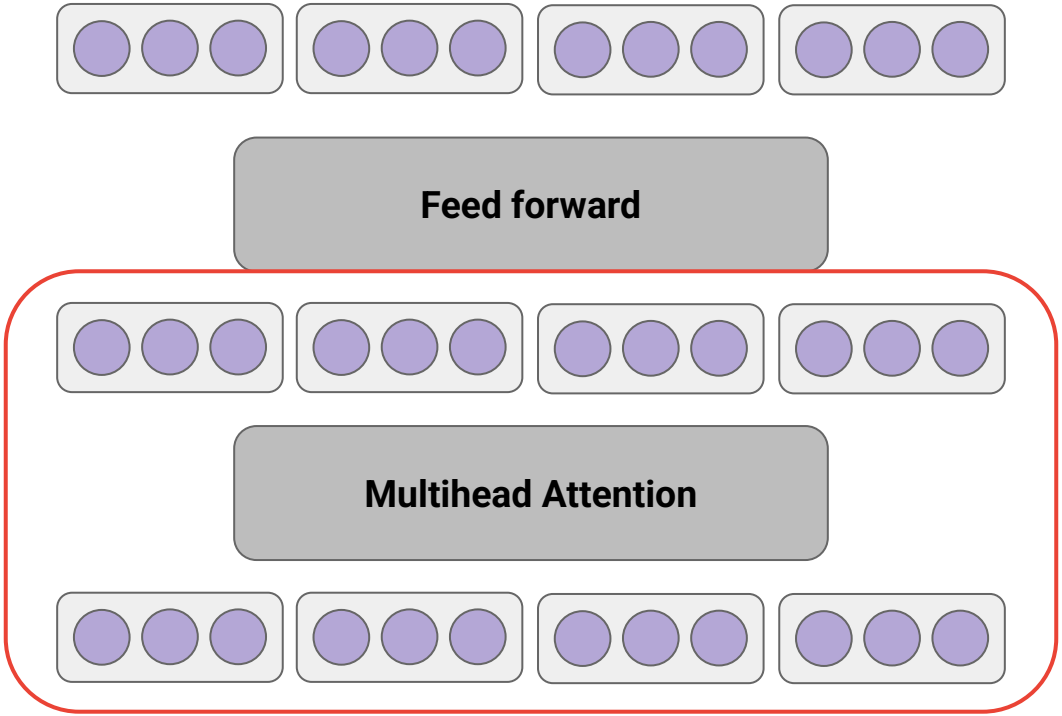
N layers


Layer N
Layer 2
Layer 1
hidden state/ activations
Transformer encoder/decoder

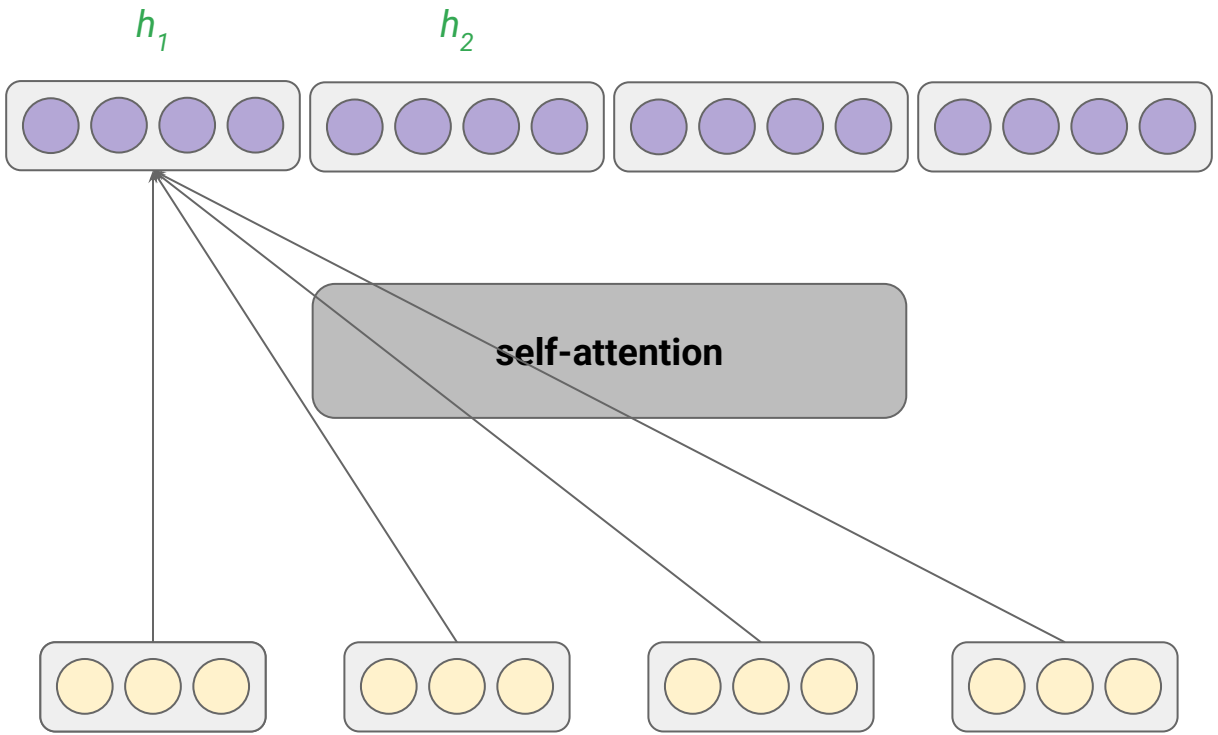
Transformer block (one layer)



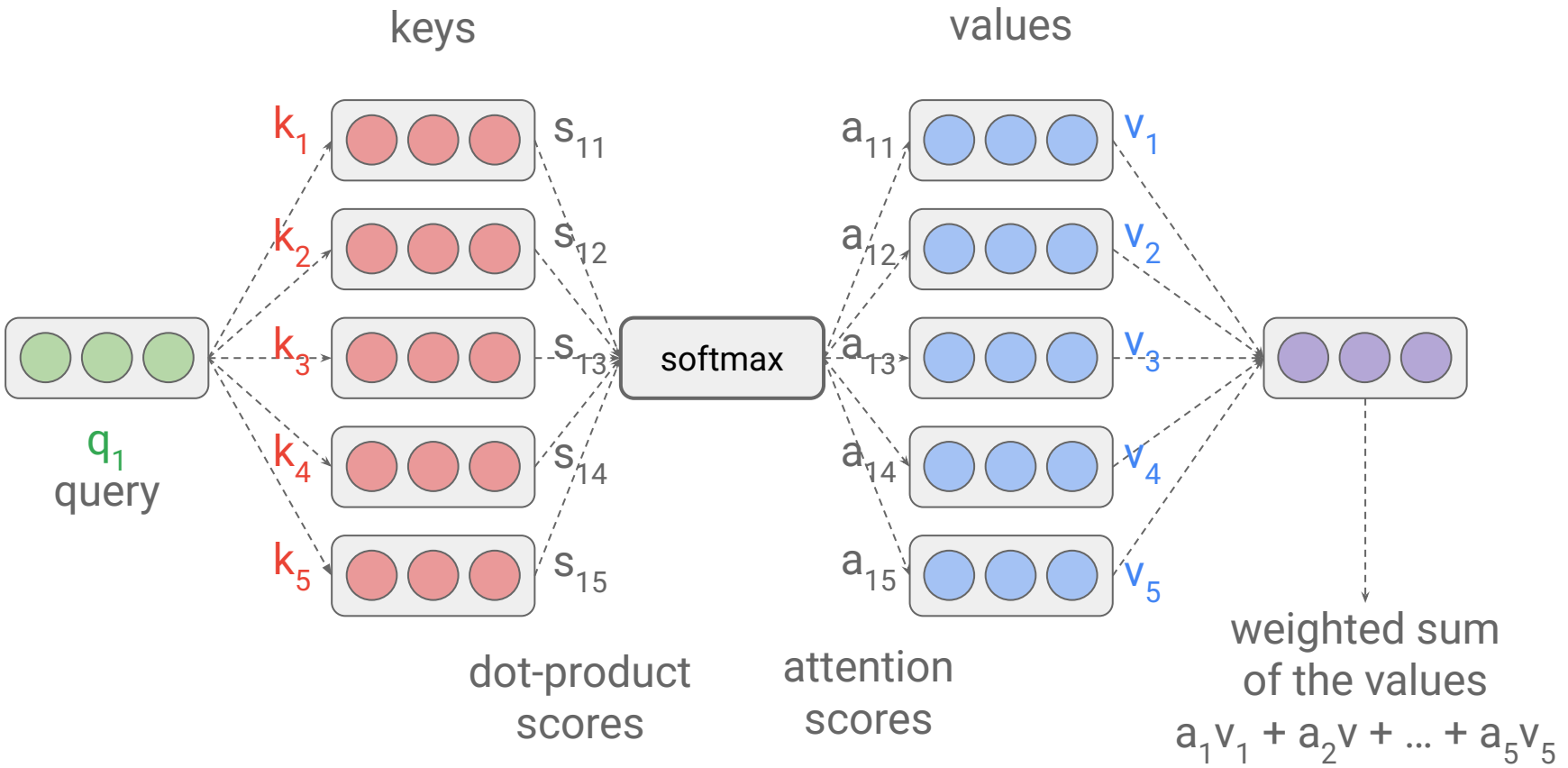
Transformer block



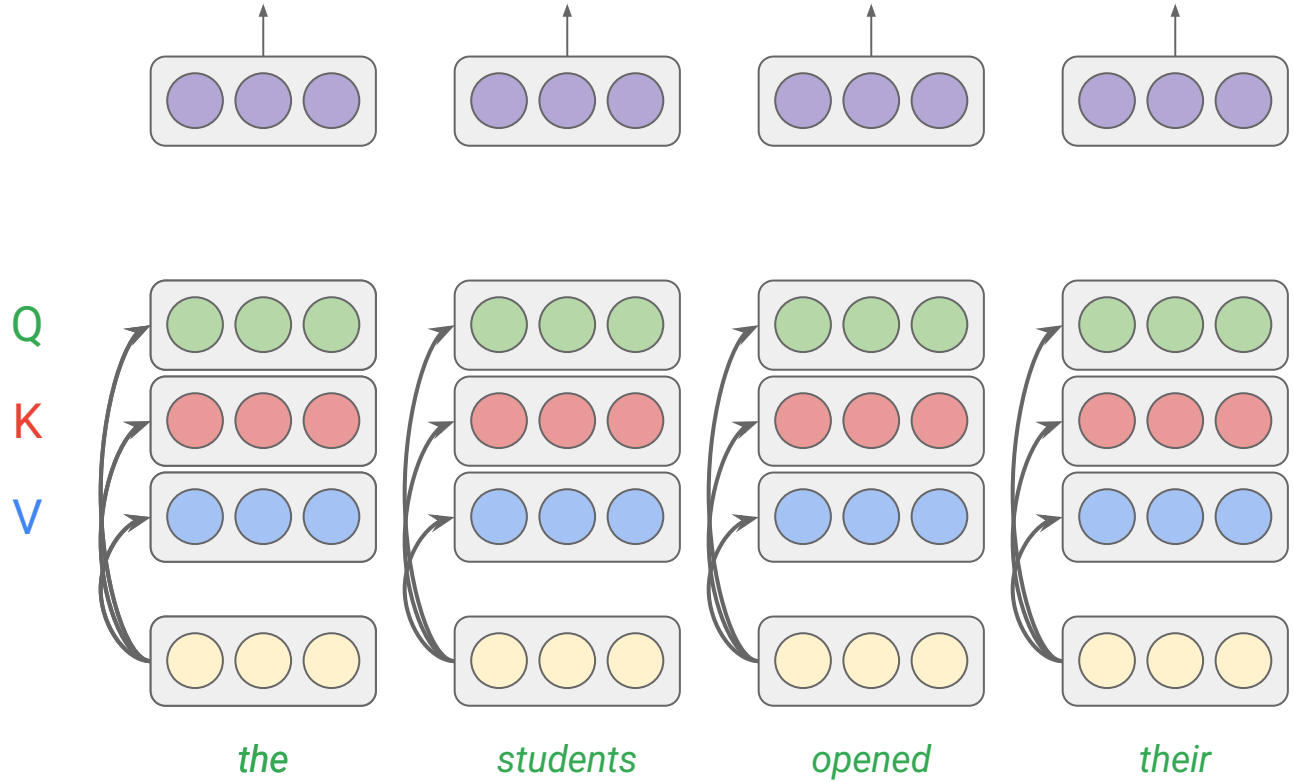
Self-attention



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)



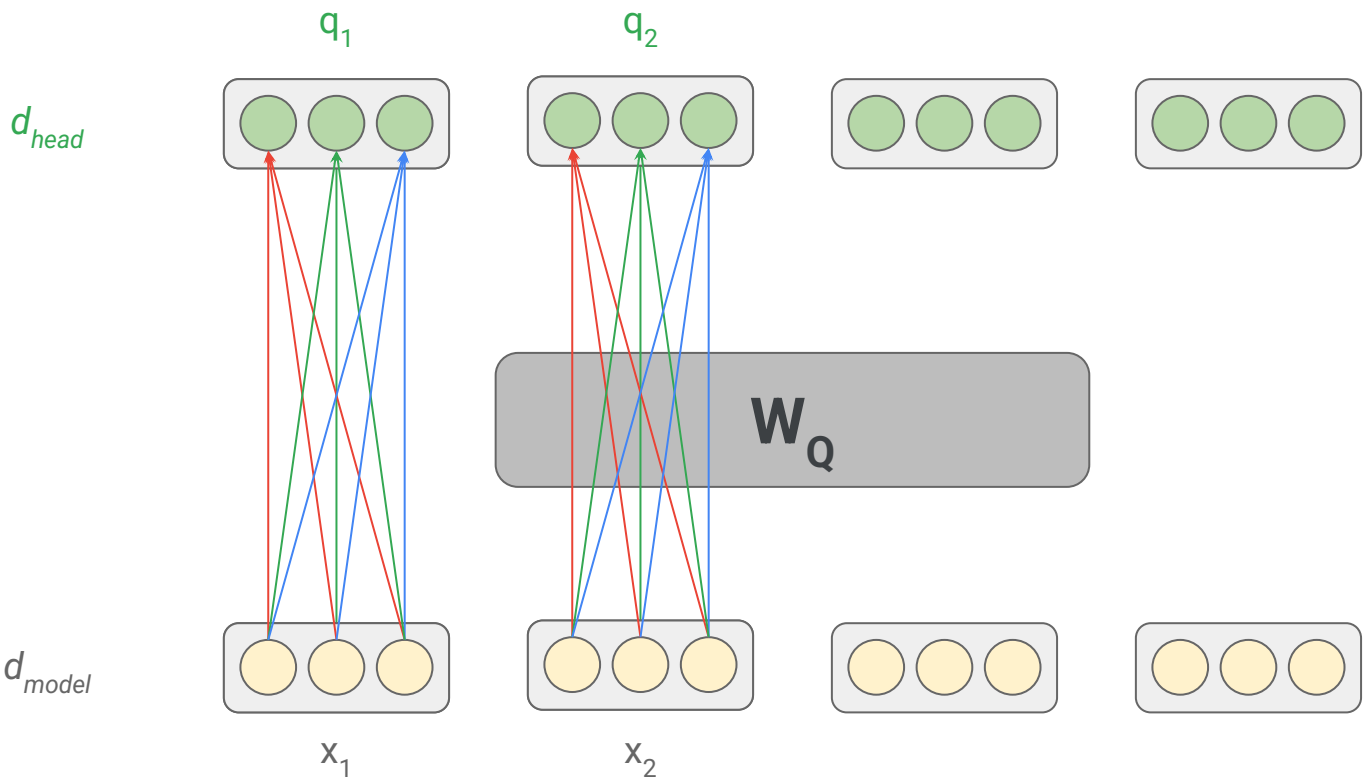
$$Q = X \cdot W_Q$$

$$K = X \cdot W_K$$

$$V = X \cdot W_V$$

linear projections

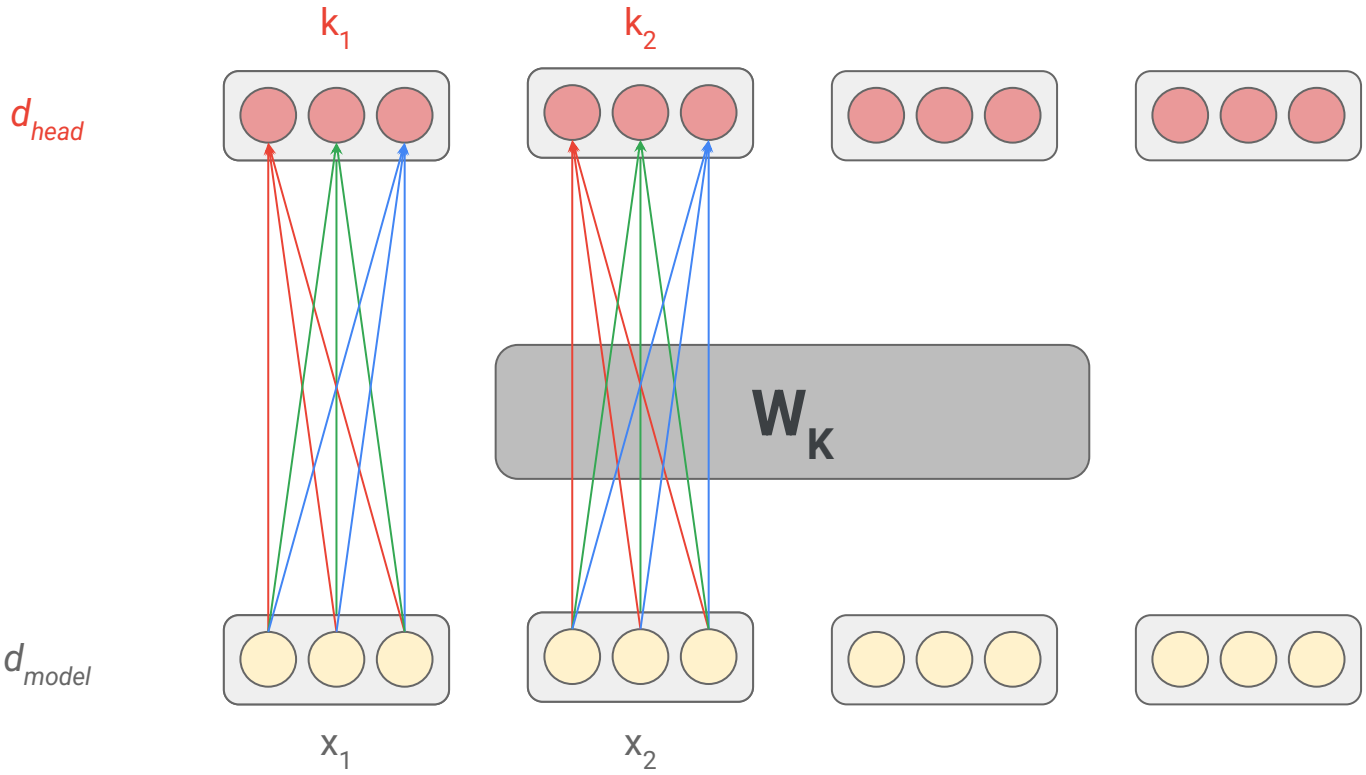
Query vectors



$$Q = X \cdot W_Q$$

linear projections

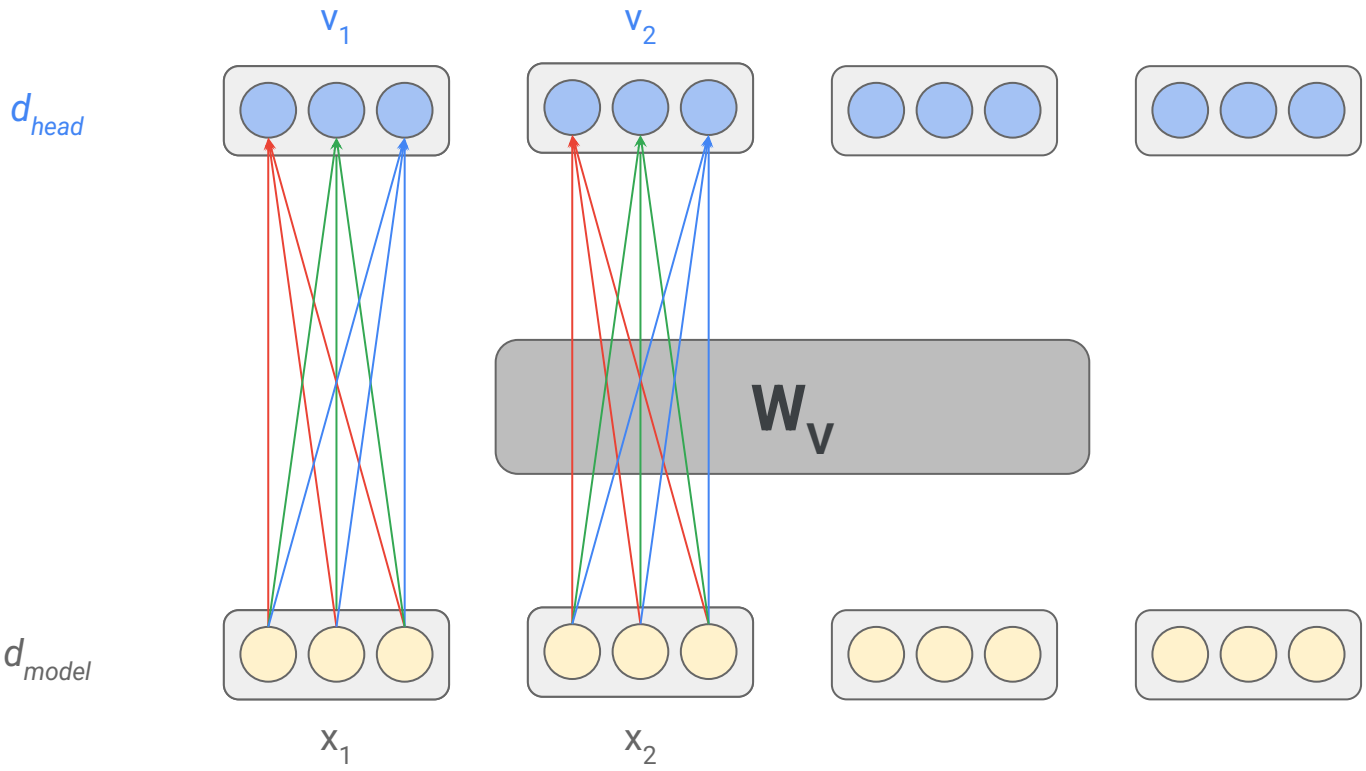
Key vectors



$$K = X \cdot W_K$$

linear projections

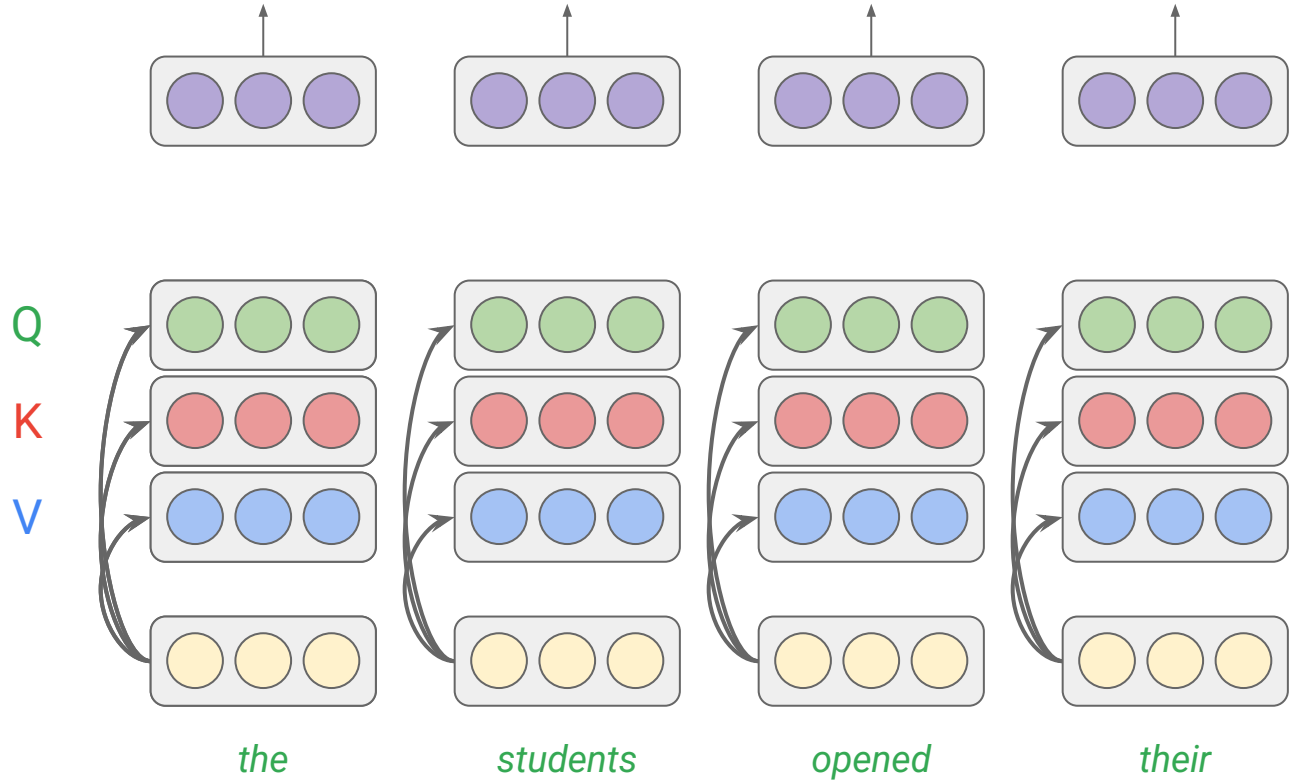
Value vectors



$$V = X \cdot W_V$$

linear projections

Attention (cont'd)



$$Q = X \cdot W_Q$$

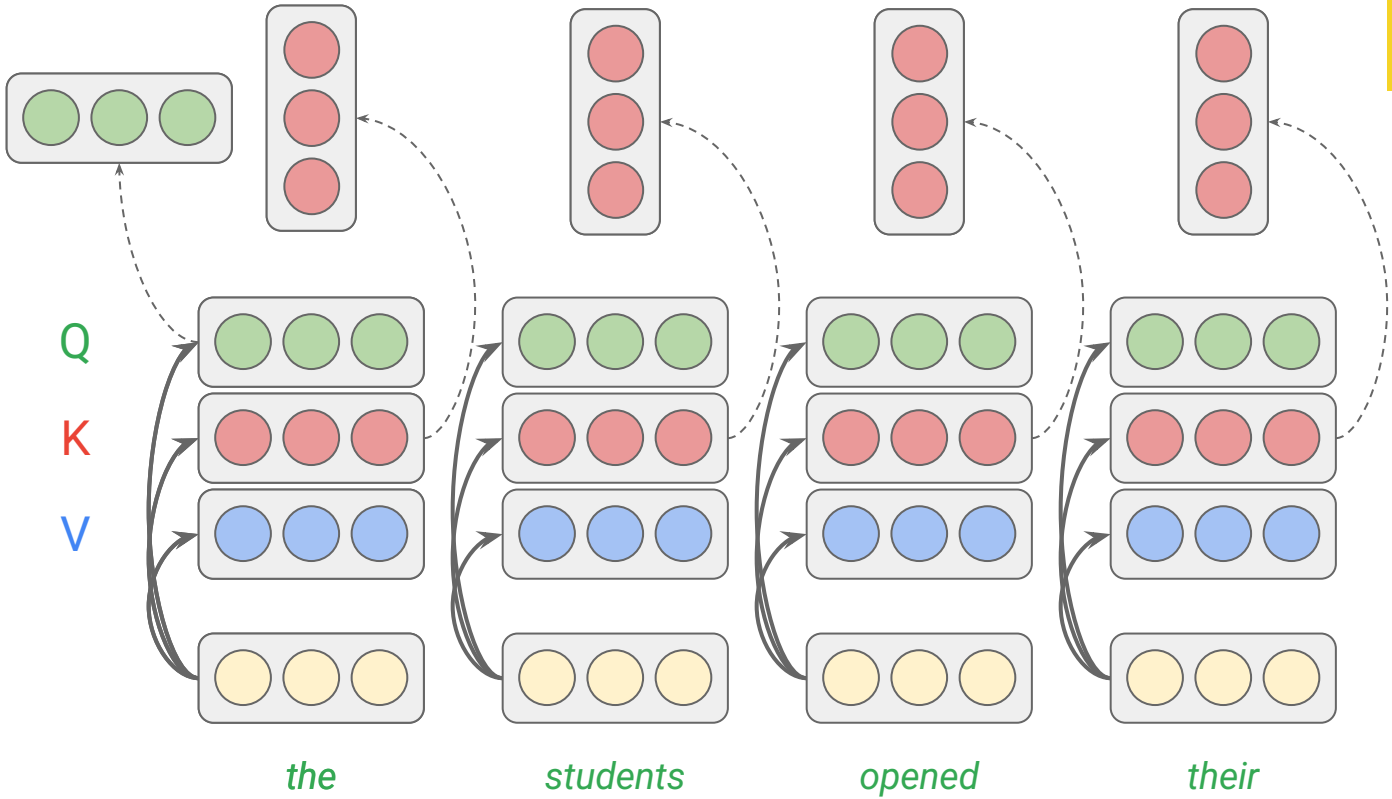
$$K = X \cdot W_K$$

$$V = X \cdot W_V$$

linear projections

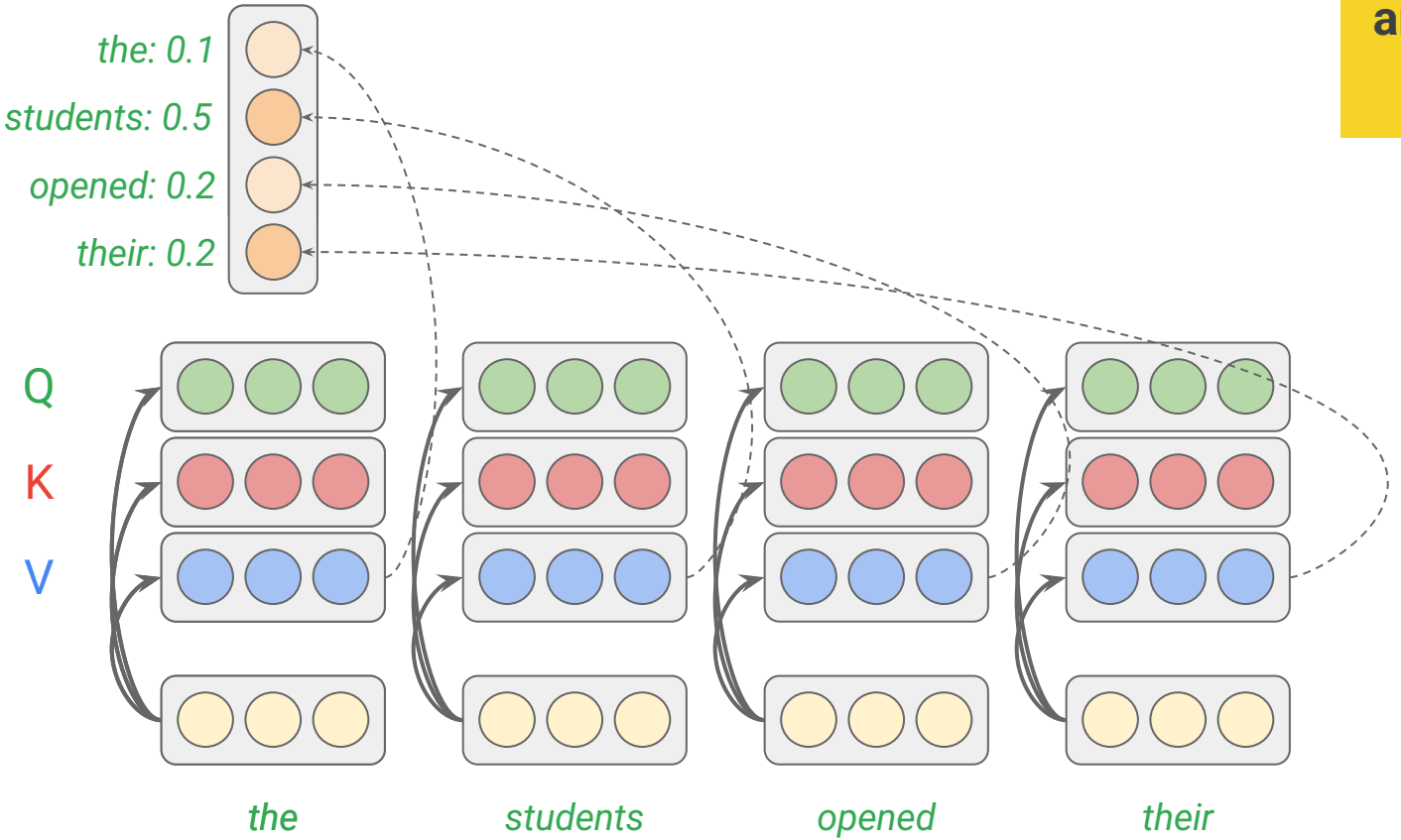
Self-attention (cont'd)

***all* computations are parallelized**



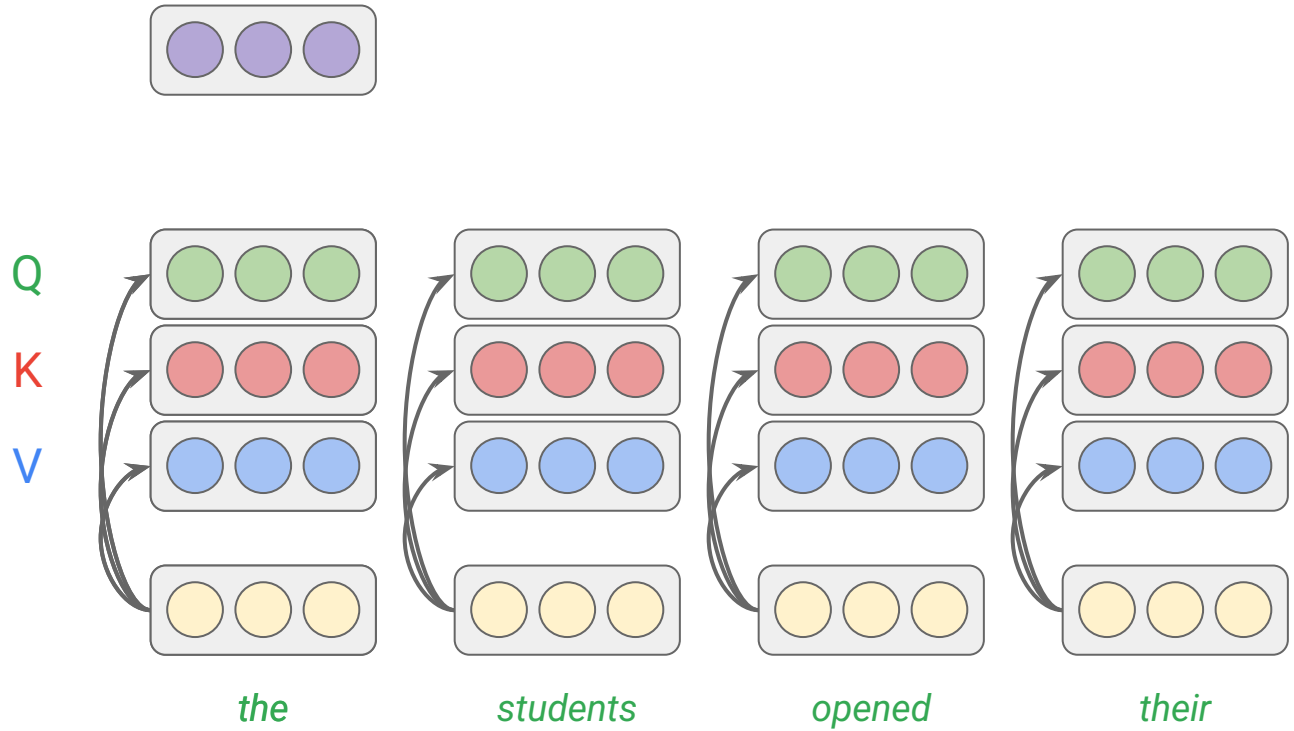
Self-attention (cont'd)

all computations are parallelized

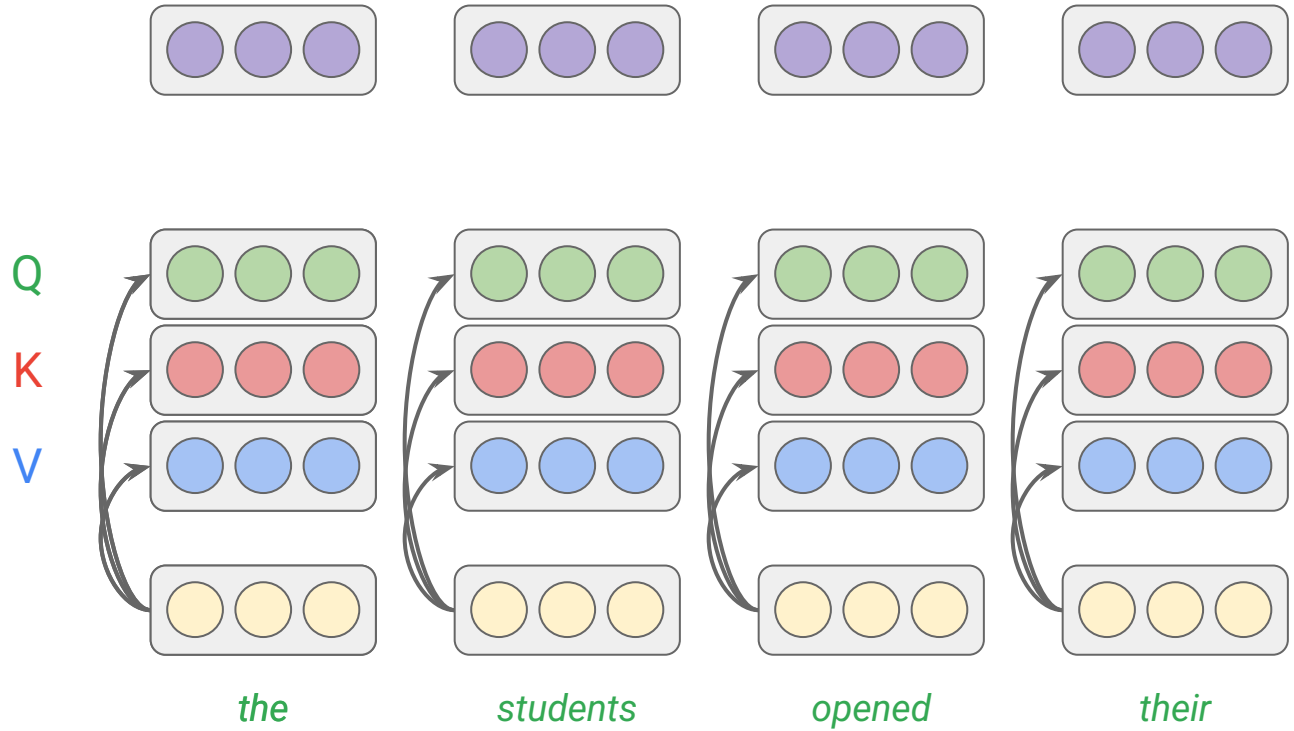


Self-attention (cont'd)

***all* computations are parallelized**

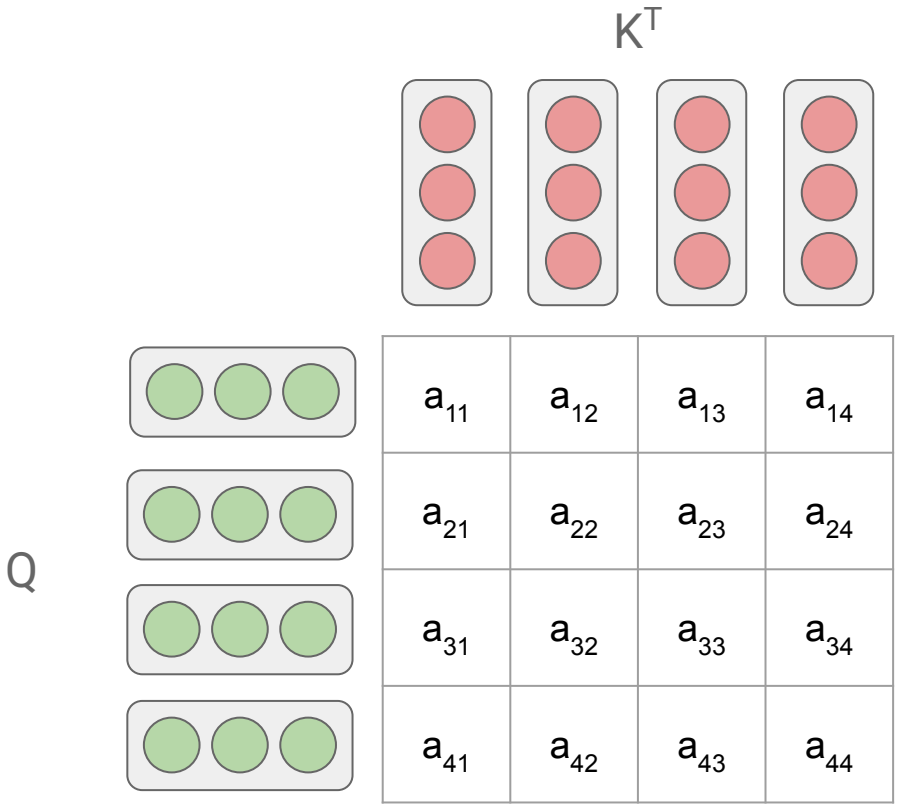


Self-attention (cont'd)



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

Quadratic complexity



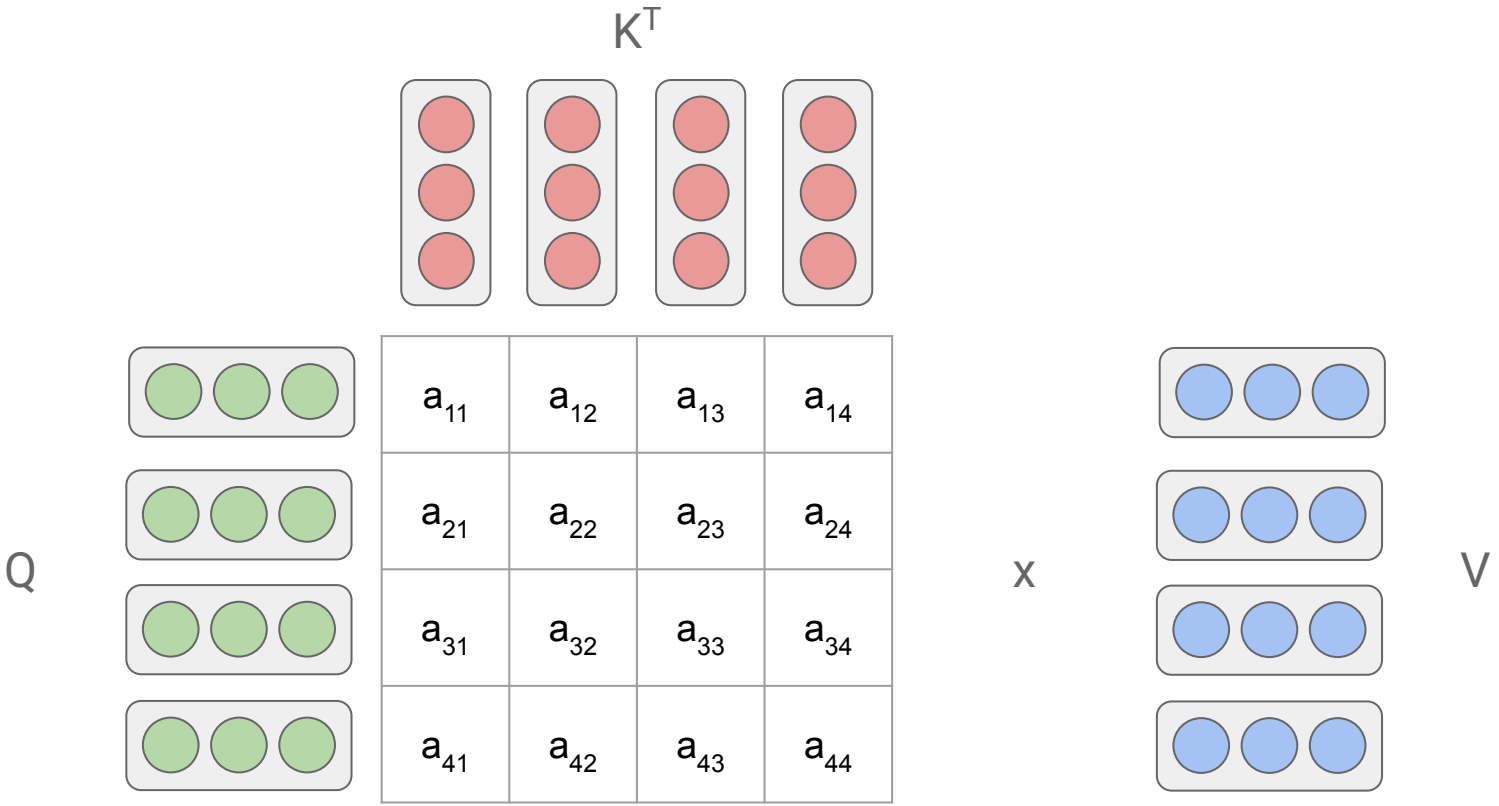
The time complexity of self-attention is quadratic in the input length $O(n^2)$

$$Q = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \\ q_{31} & q_{32} \end{pmatrix} = \begin{pmatrix} - & q_1 & - \\ - & q_2 & - \\ - & q_3 & - \end{pmatrix},$$

$$K = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \\ k_{31} & k_{32} \end{pmatrix} = \begin{pmatrix} - & k_1 & - \\ - & k_2 & - \\ - & k_3 & - \end{pmatrix}.$$

$$QK^\top = \begin{pmatrix} q_{11}k_{11} + q_{12}k_{12} & q_{11}k_{21} + q_{12}k_{22} & q_{11}k_{31} + q_{12}k_{32} \\ q_{21}k_{11} + q_{22}k_{12} & q_{21}k_{21} + q_{22}k_{22} & q_{21}k_{31} + q_{22}k_{32} \\ q_{31}k_{11} + q_{32}k_{12} & q_{31}k_{21} + q_{32}k_{22} & q_{31}k_{31} + q_{32}k_{32} \end{pmatrix}.$$

Quadratic complexity (cont'd)



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,

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

Rewrite this as a column vector,

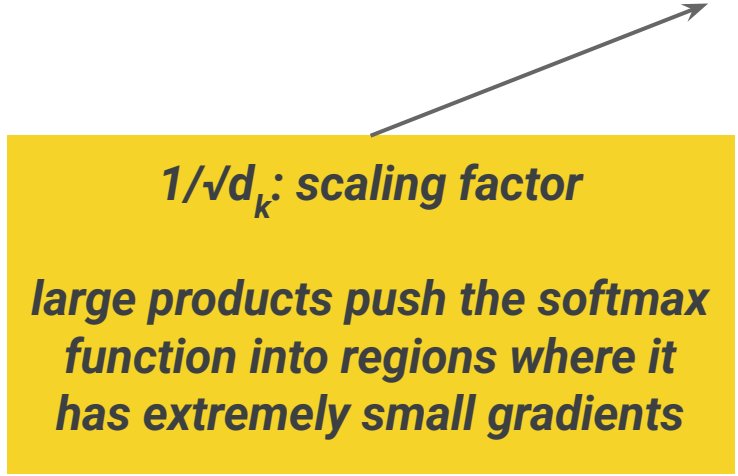
$$(AV)_{1:}^{\top} = \begin{bmatrix} a_{11}v_{11} + a_{12}v_{21} + a_{13}v_{31} + a_{14}v_{41} \\ a_{11}v_{12} + a_{12}v_{22} + a_{13}v_{32} + a_{14}v_{42} \\ a_{11}v_{13} + a_{12}v_{23} + a_{13}v_{33} + a_{14}v_{43} \end{bmatrix}.$$

Factor by coefficients,

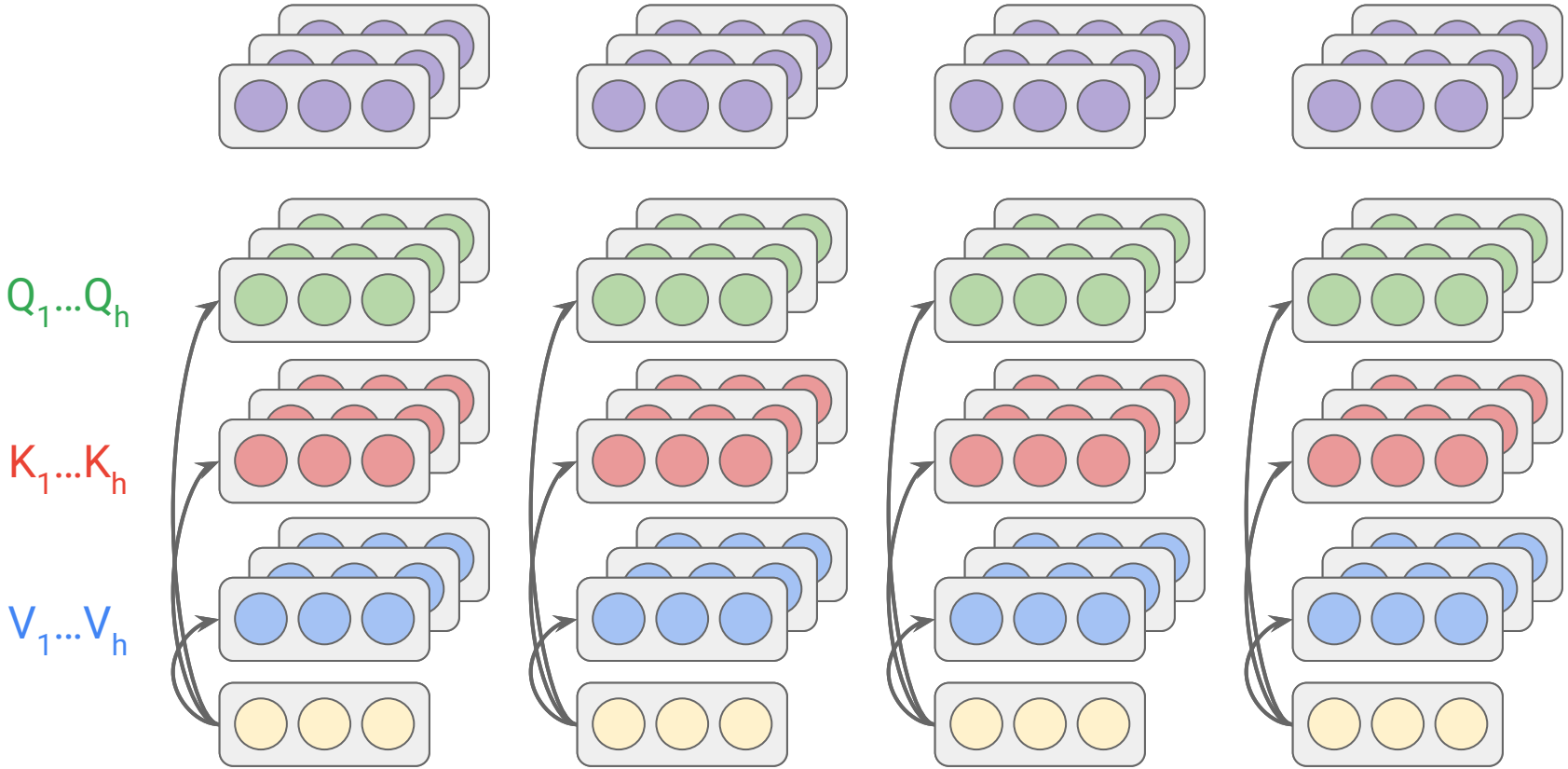
$$(AV)_{1:}^{\top} = a_{11} \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} + a_{12} \begin{bmatrix} v_{21} \\ v_{22} \\ v_{23} \end{bmatrix} + a_{13} \begin{bmatrix} v_{31} \\ v_{32} \\ v_{33} \end{bmatrix} + a_{14} \begin{bmatrix} v_{41} \\ v_{42} \\ v_{43} \end{bmatrix}.$$

All computations are parallelized

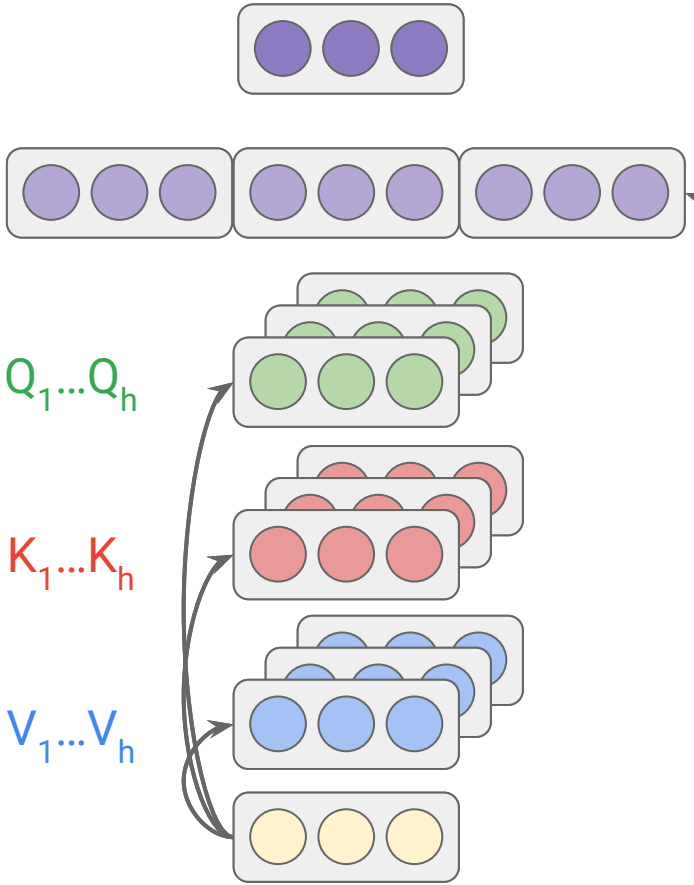
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Multi-head attention

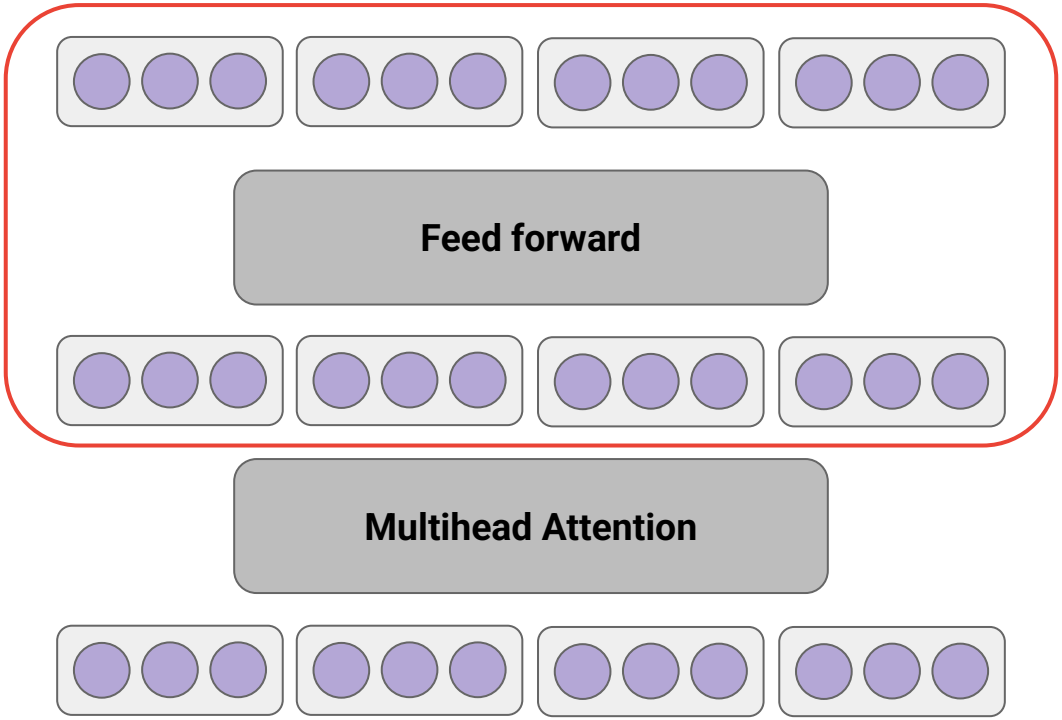


Multi-head attention (cont'd)

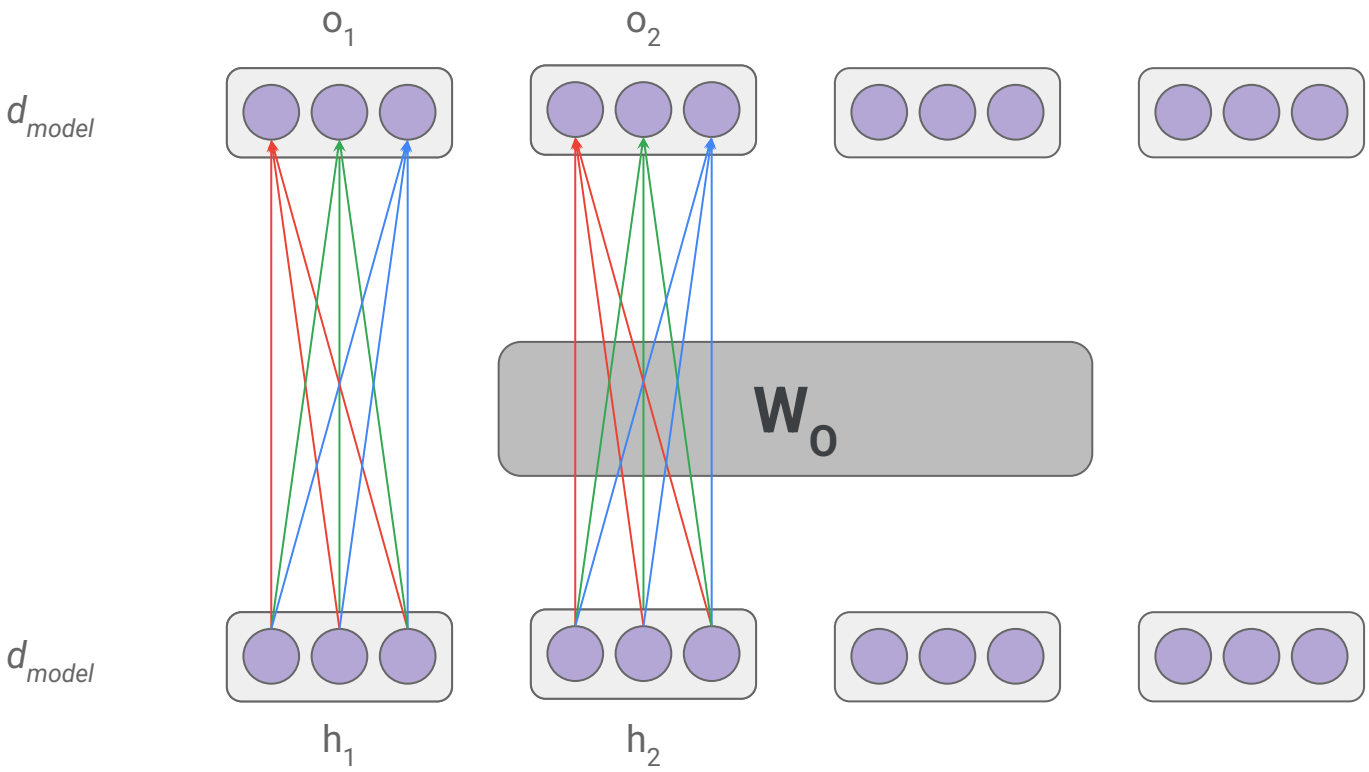


These output values are concatenated and once again projected

Transformer block (cont'd)



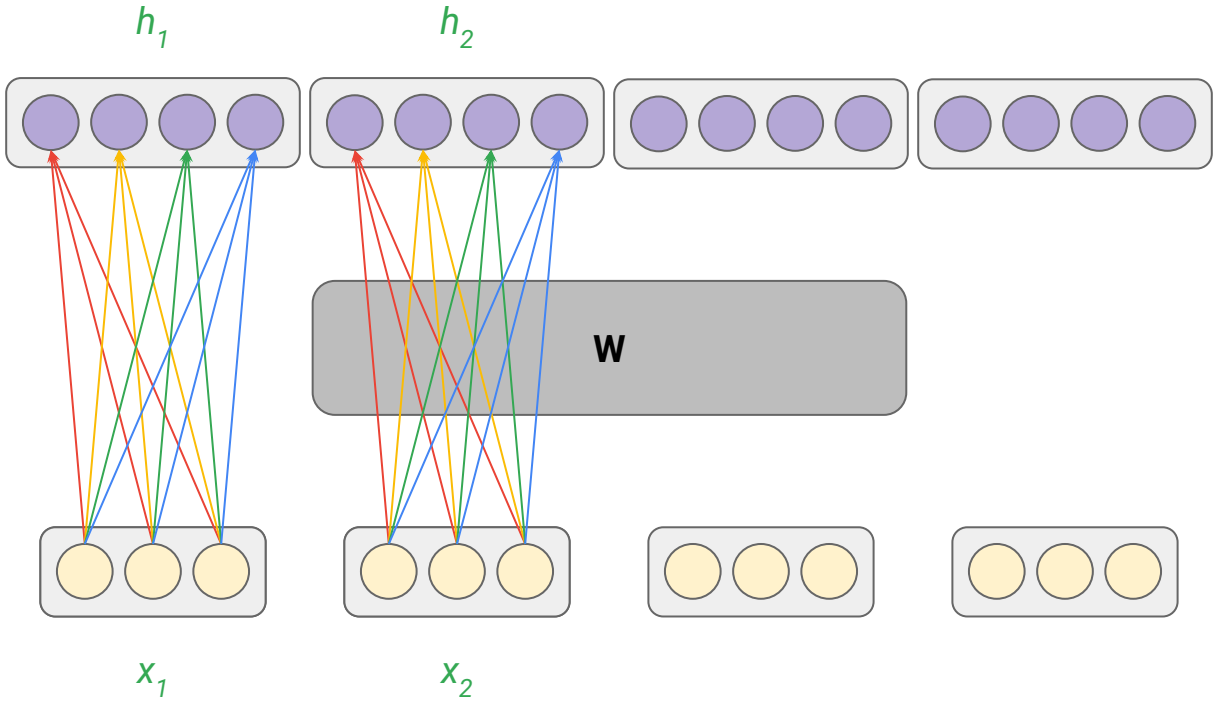
output vectors



$$O = H \cdot W_O$$

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:

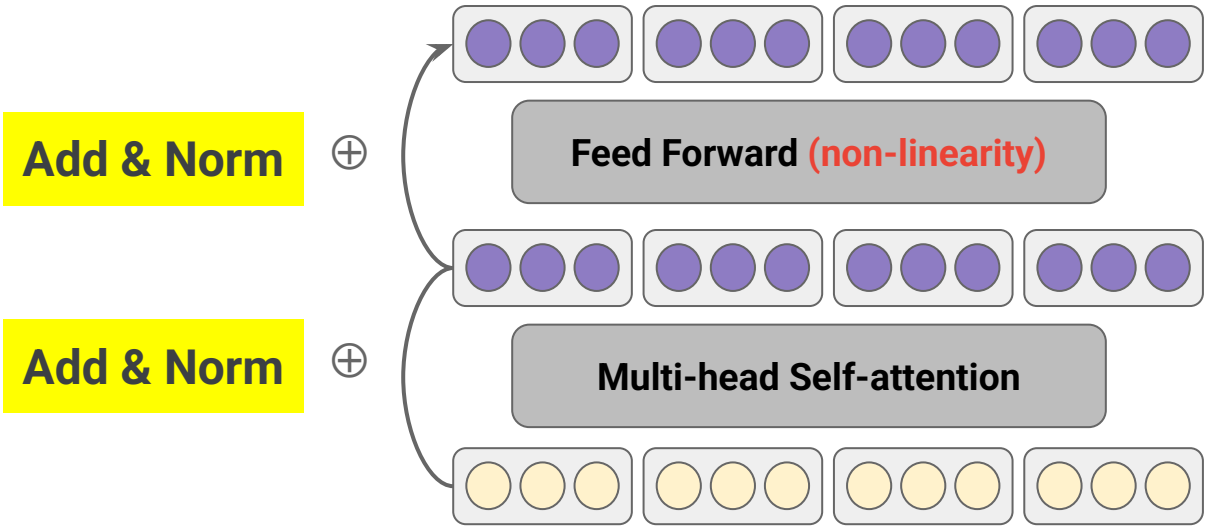
$$H = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{bmatrix}$$

This results in:

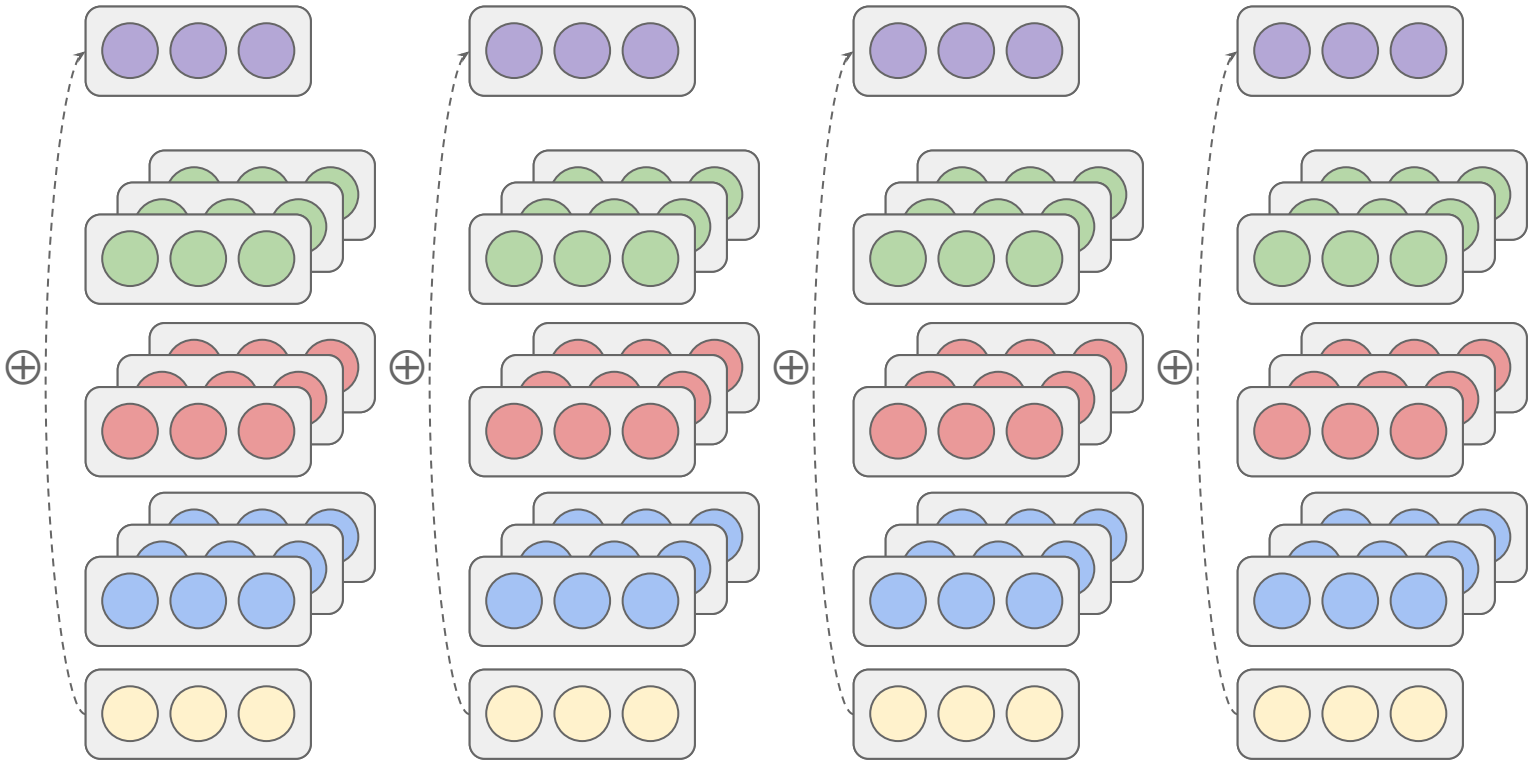
h_1 h_2

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

Transformer block (one layer)



Residual connection



Residual connection

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

Layer normalization

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

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

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

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

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

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

- Sequence: **“I like cats”**
- Positions: $0, 1, 2$
- Embedding dimension: $d_{\text{model}} = 4$

For each position, we need **4 positional-encoding numbers**, corresponding to dimensions $0, 1, 2, 3$.

For $d_{\text{model}} = 4$, your equations become:

$$PE(pos, 0) = \sin(pos)$$

$$PE(pos, 1) = \cos(pos)$$

$$PE(pos, 2) = \sin(pos/100)$$

$$PE(pos, 3) = \cos(pos/100)$$

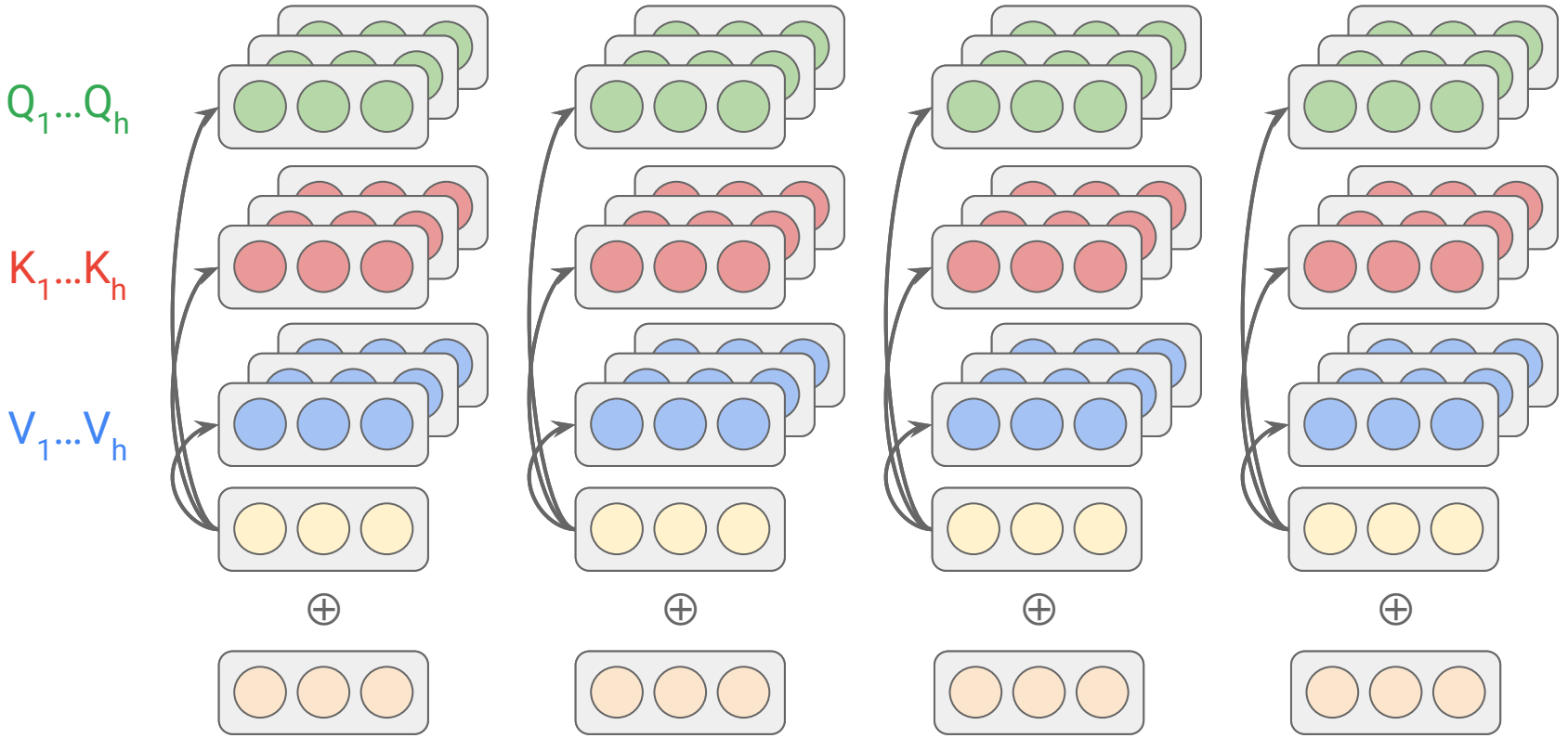
Token	pos	dim 0	dim 1	dim 2	dim 3
I	0	0.000	1.000	0.000	1.000
like	1	0.841	0.540	0.010	1.000
cats	2	0.909	-0.416	0.020	1.000

For example, for **"like"**, which is at $pos = 1$:

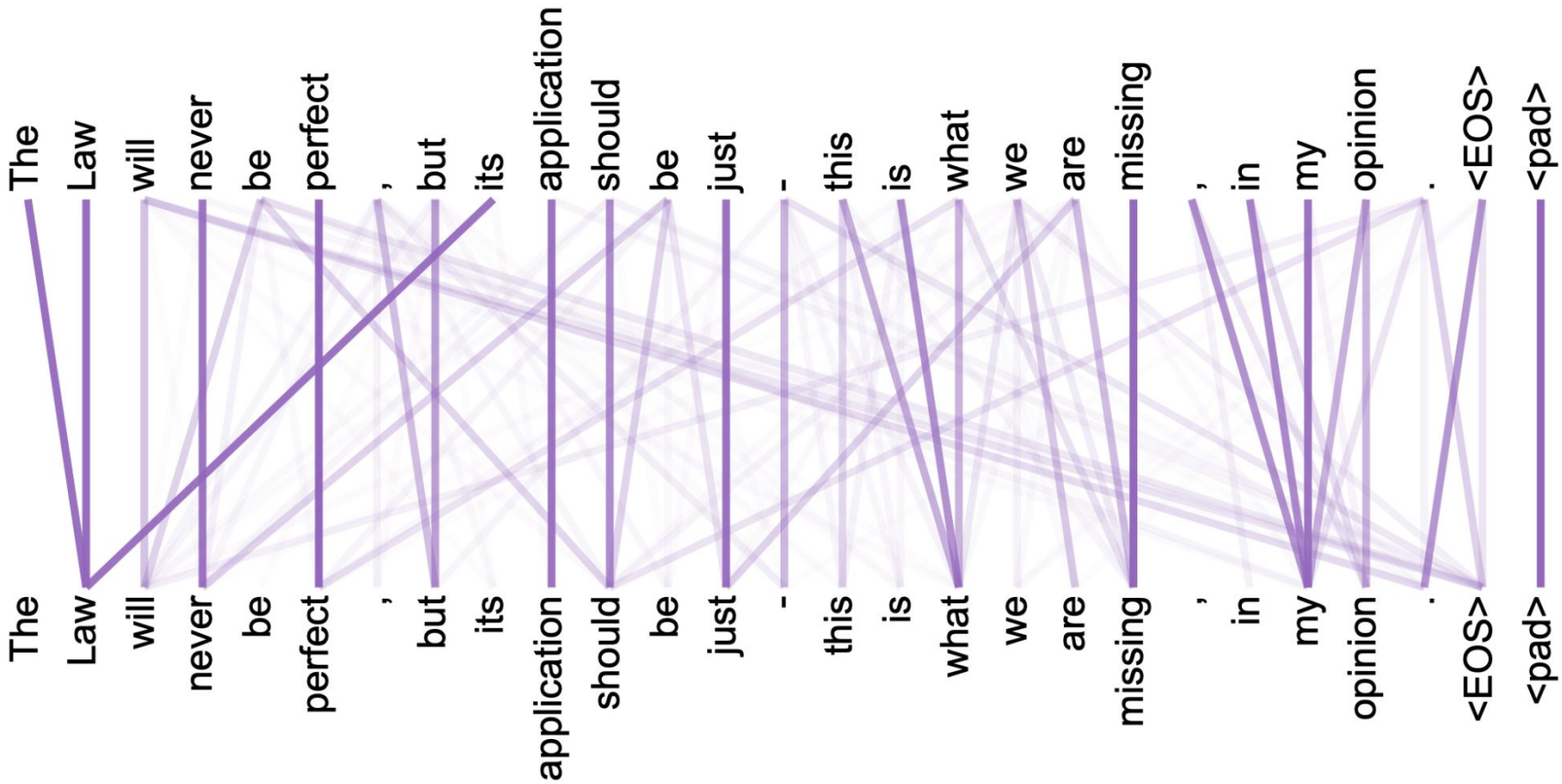
$$PE(1) = [\sin(1), \cos(1), \sin(1/100), \cos(1/100)]$$

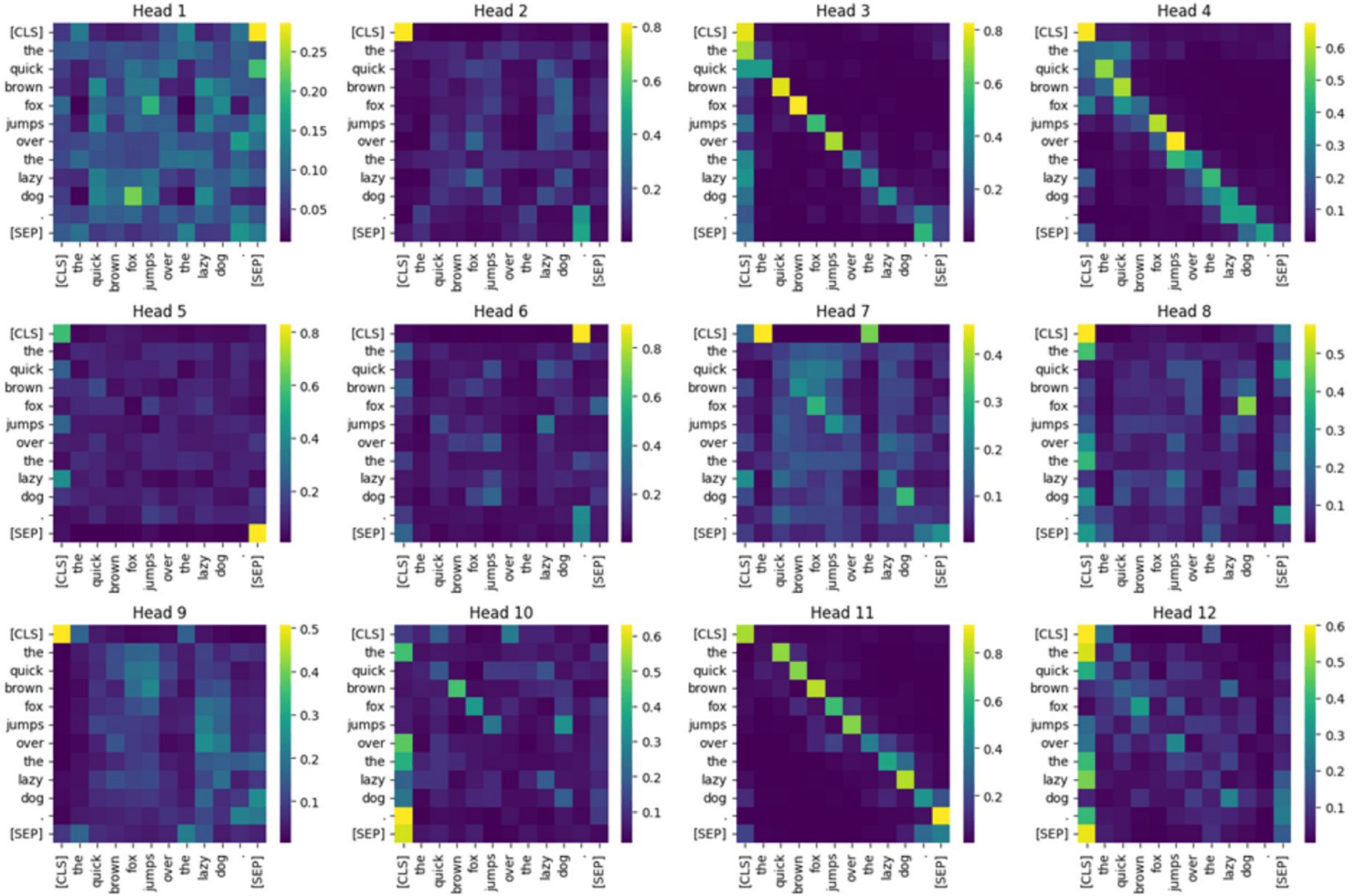
$$\approx [0.841,\ 0.540,\ 0.010,\ 1.000].$$

Positional Encoding (cont'd)



Attention visualizations



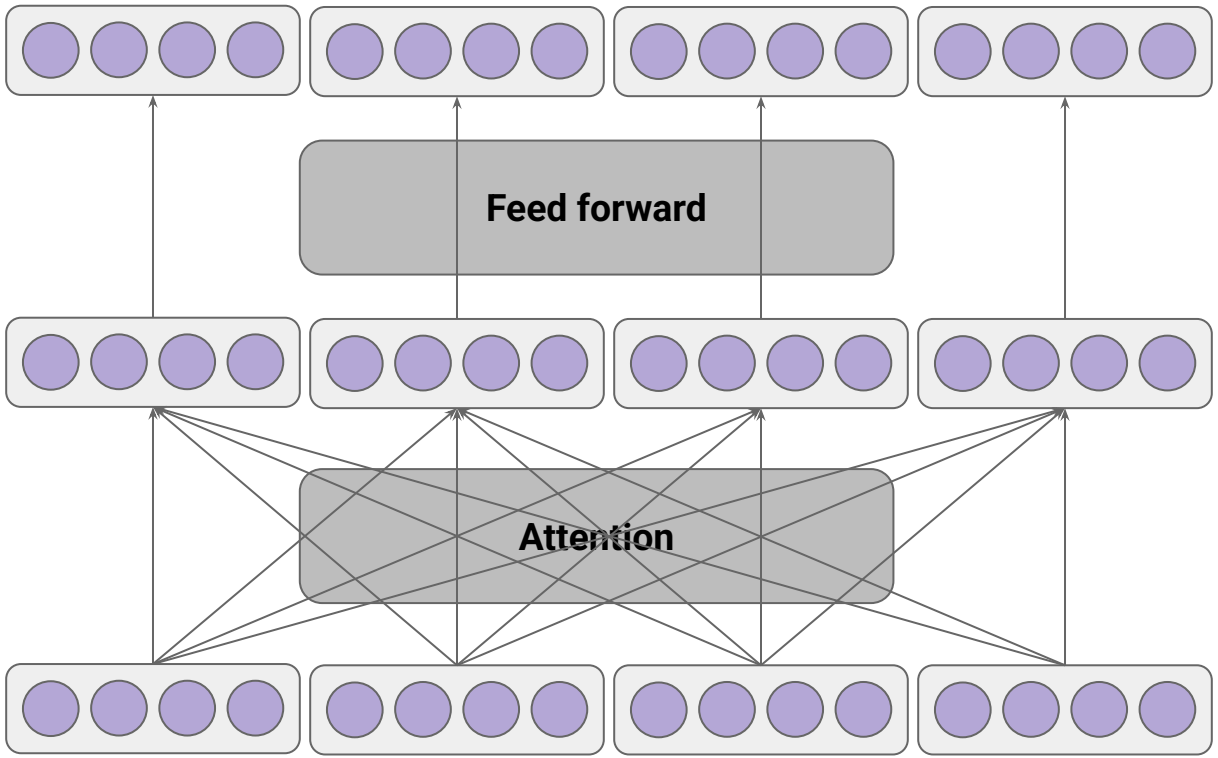


https://www.kdnuggets.com/how-to-visualize-model-internals-and-attention-in-hugging-face-transformers

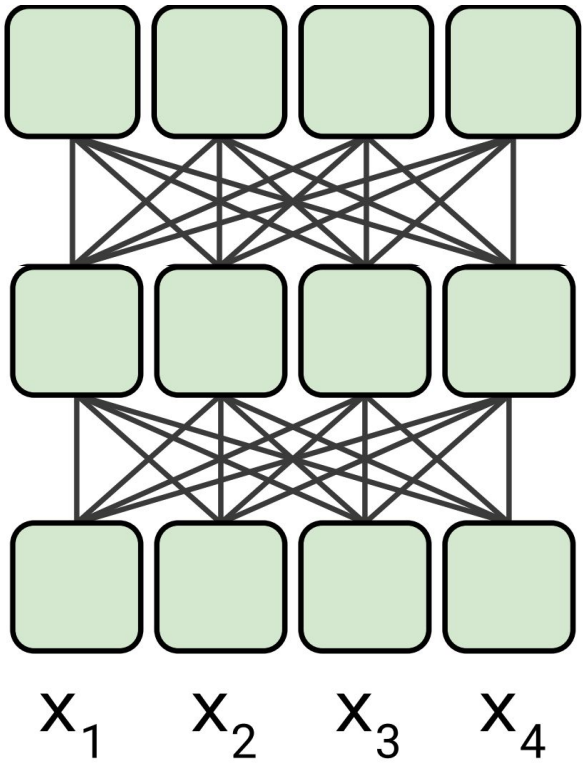
Attention visualizations (cont’d)

https://github.com/jessevig/bertviz

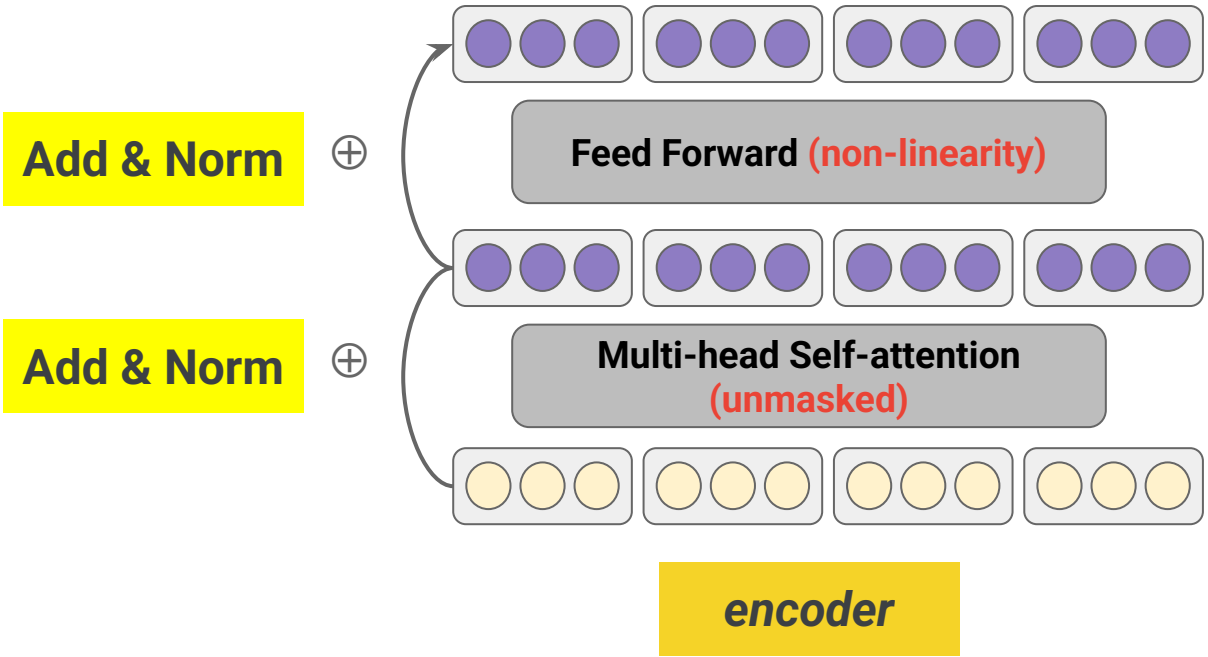
Putting it all together



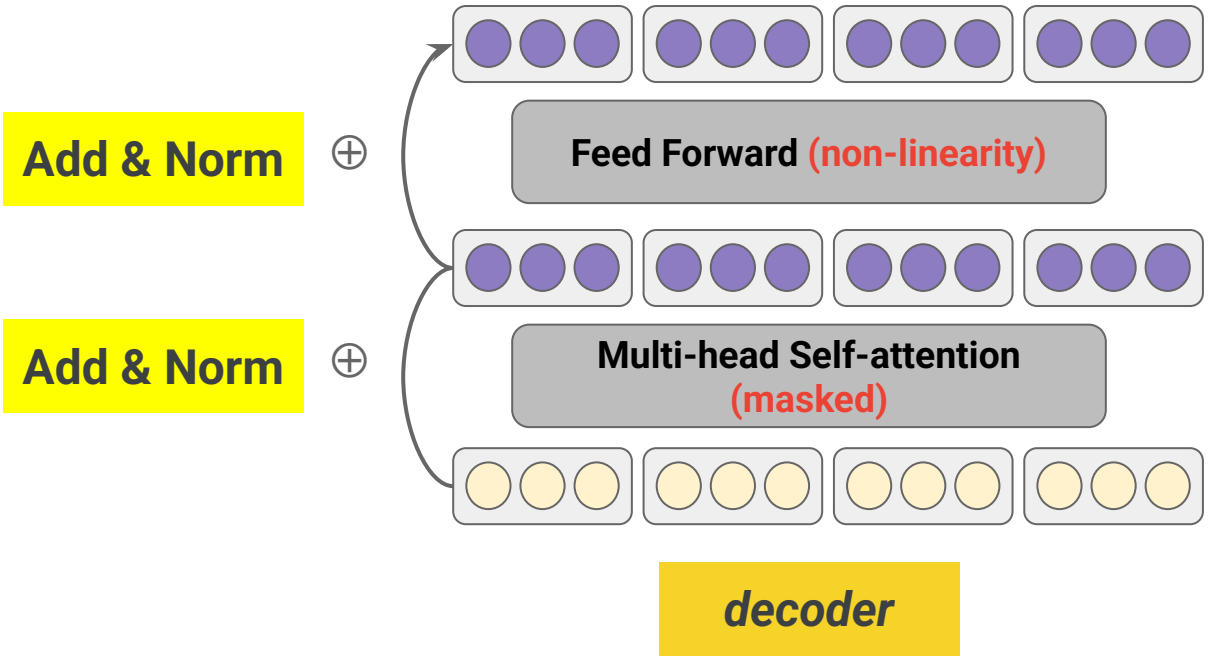
Encoder only



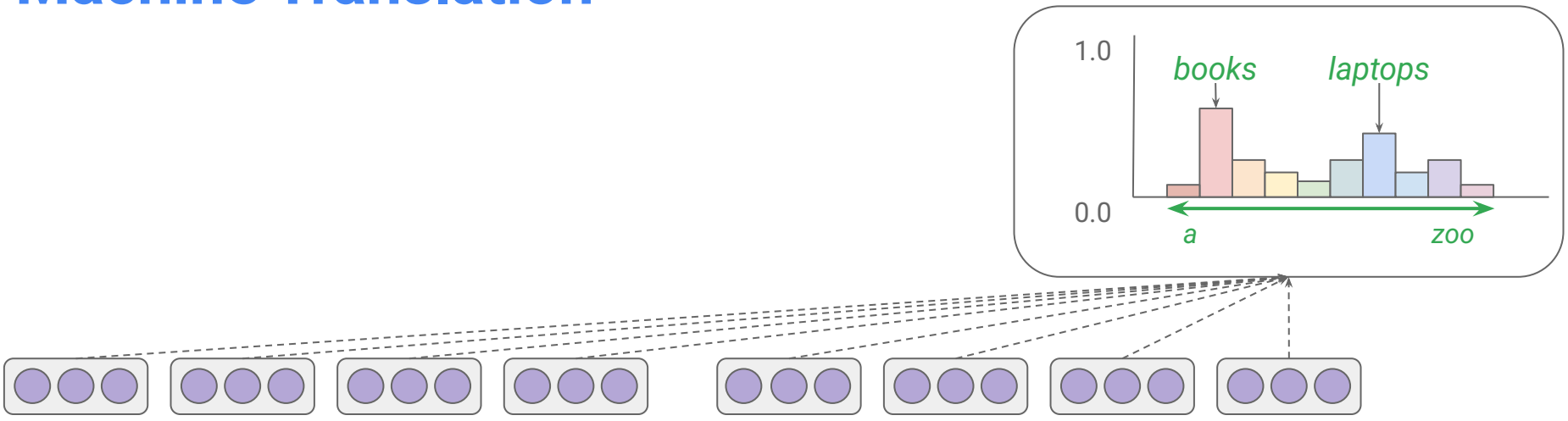
Encoder (one layer)



Decoder (one layer)



Machine Translation

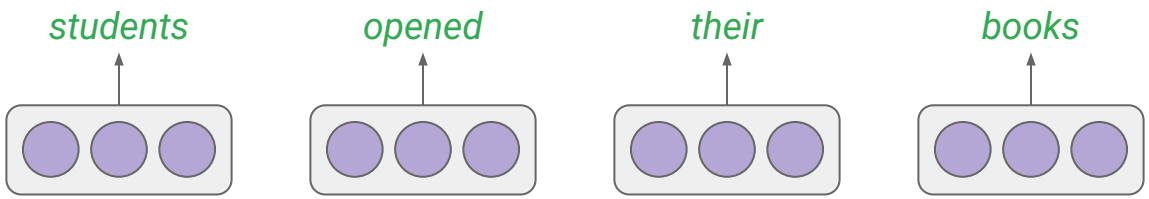


les étudiants ont ouvert leurs livres

the *students* *opened* *their*

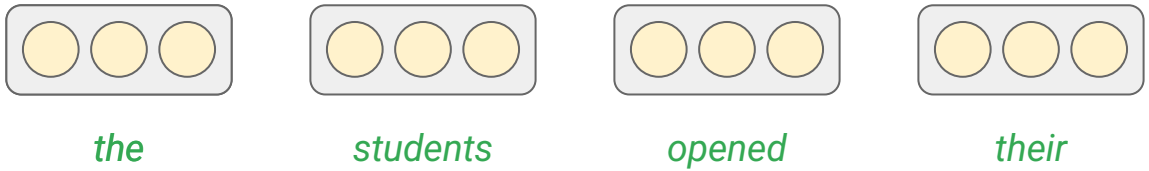
encoder ***decoder***

Transformer decoder

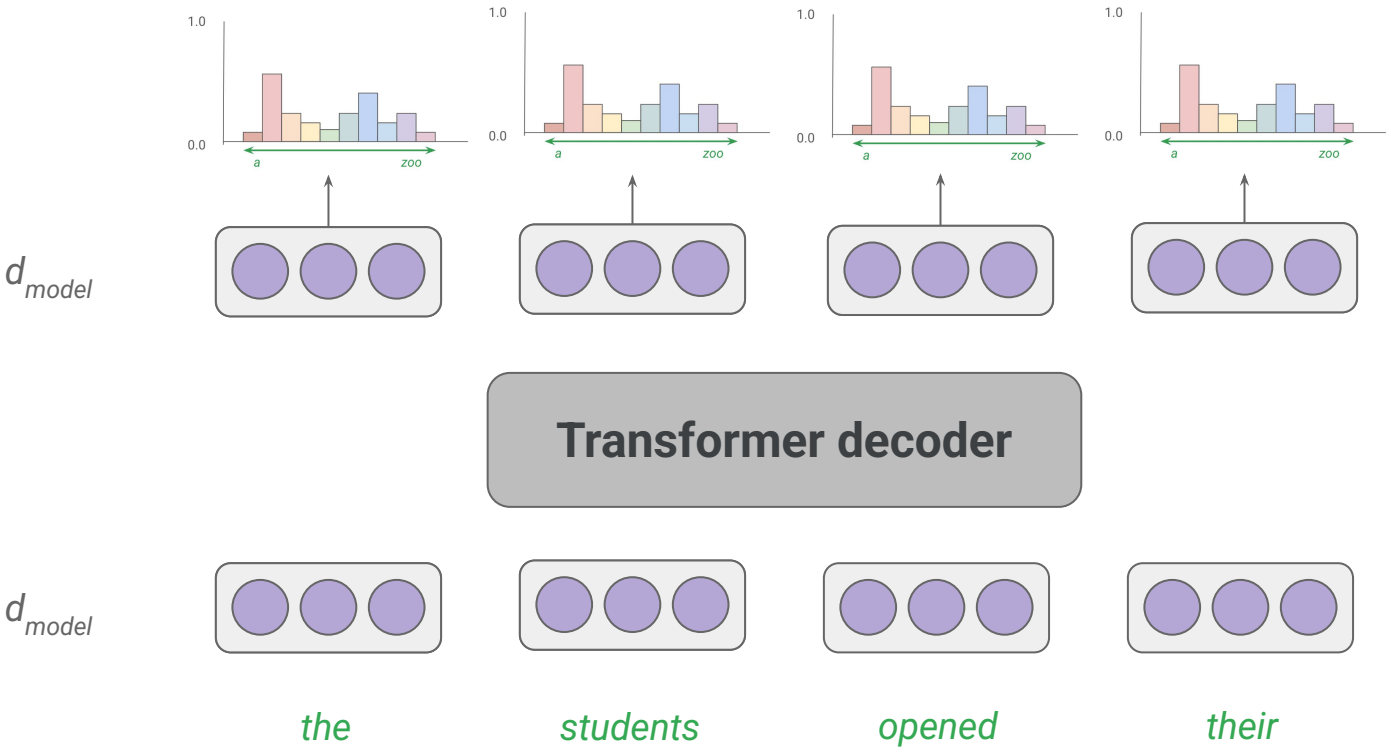


the architecture used in frontier LLMs

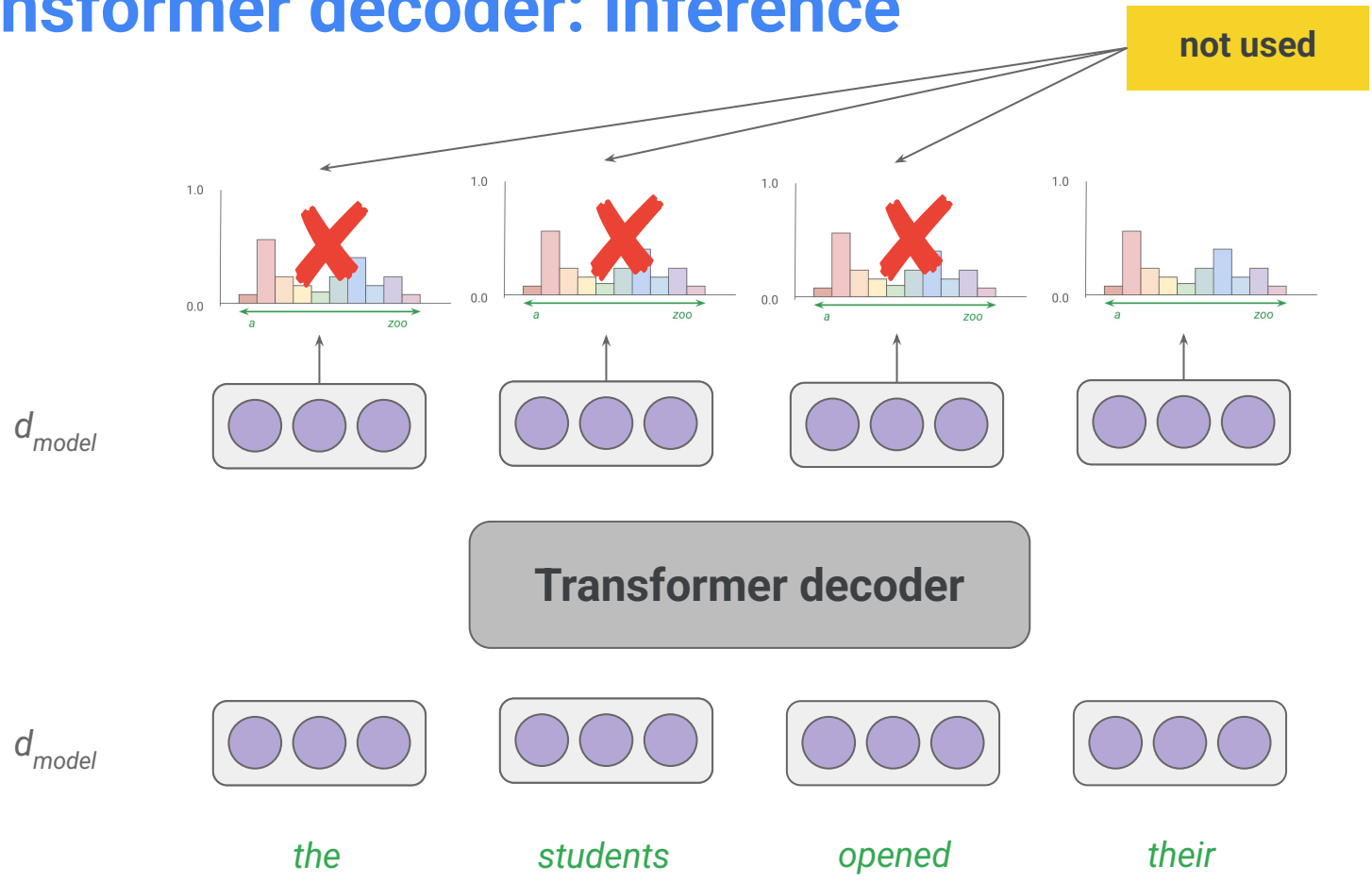
Transformer decoder
(masked)



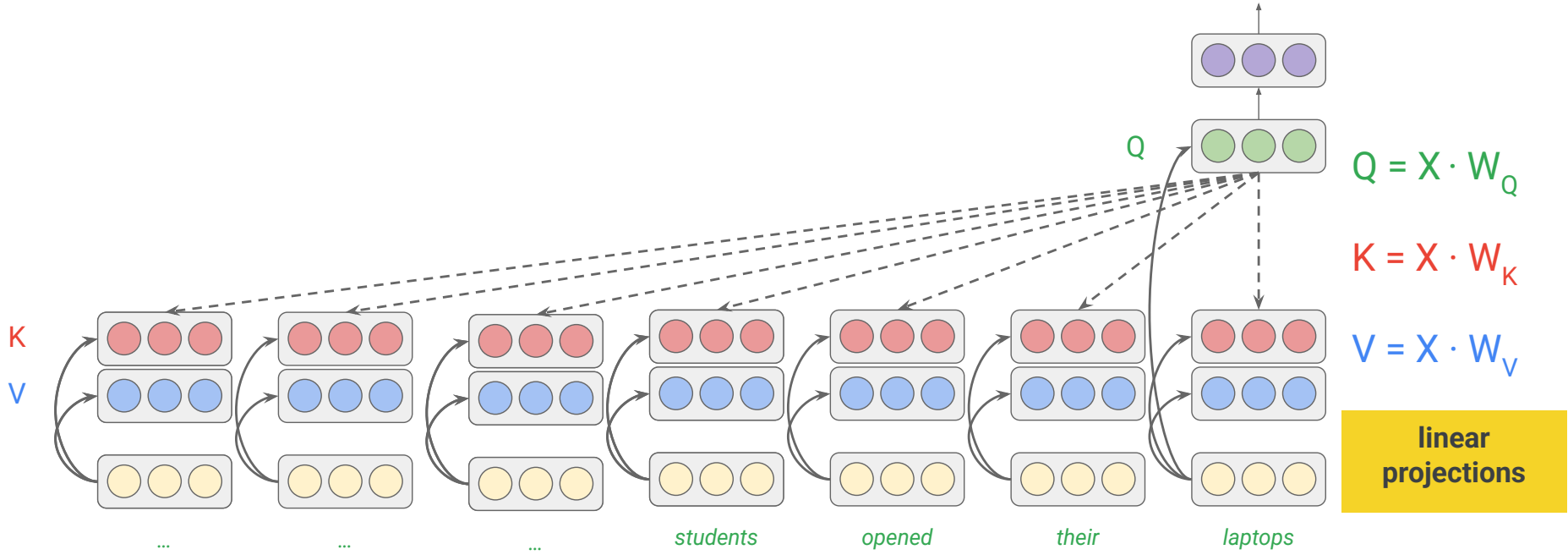
Transformer decoder: training



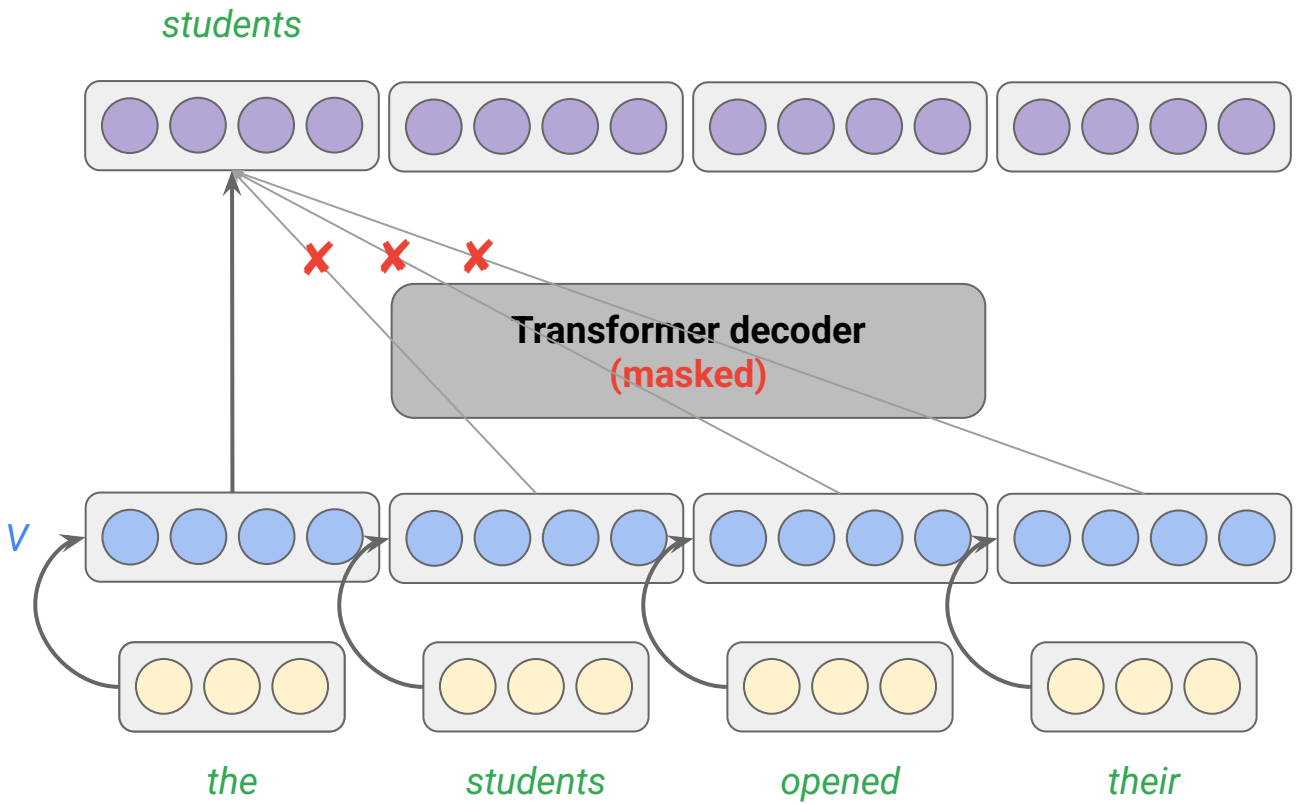
Transformer decoder: inference



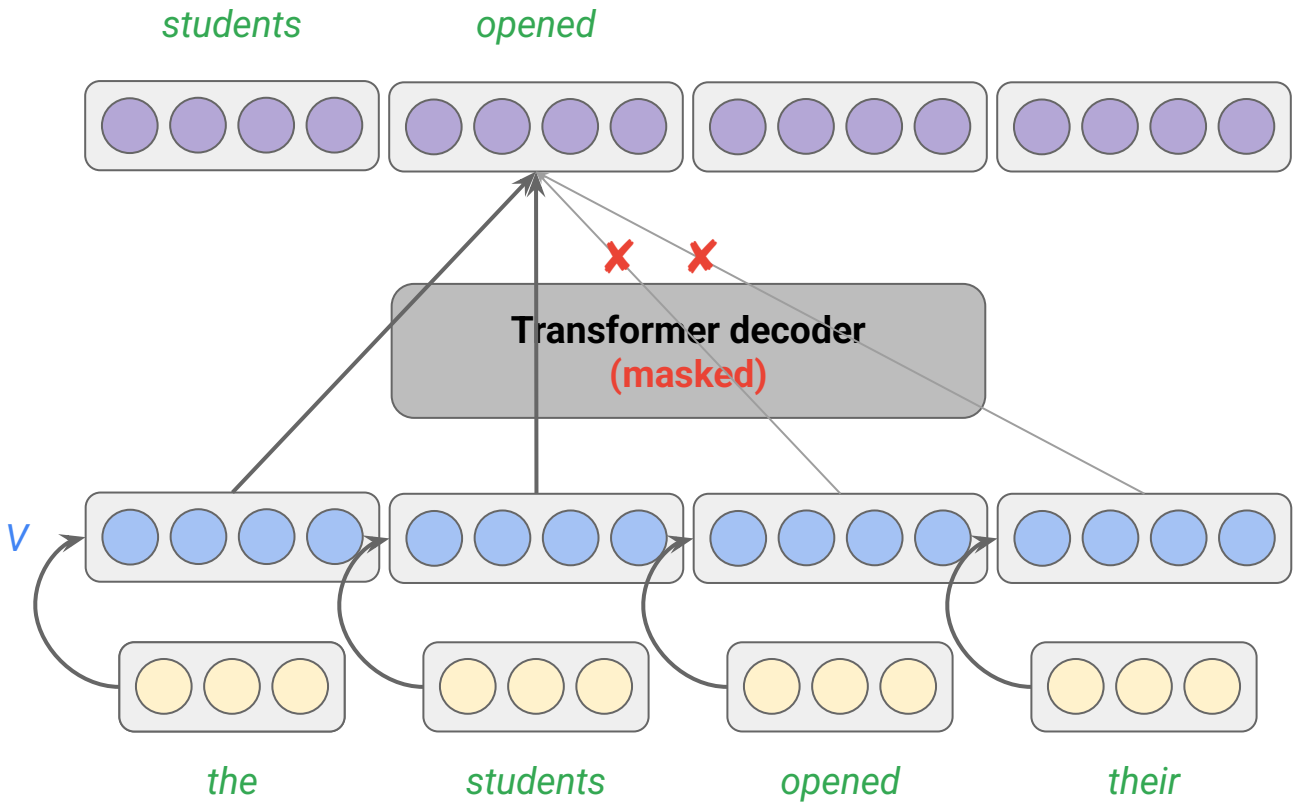
Inference


Q
K
V
Q = X · W_Q
K = X · W_K
V = X · W_V
linear projections
...
...
...
students
opened
their
laptops

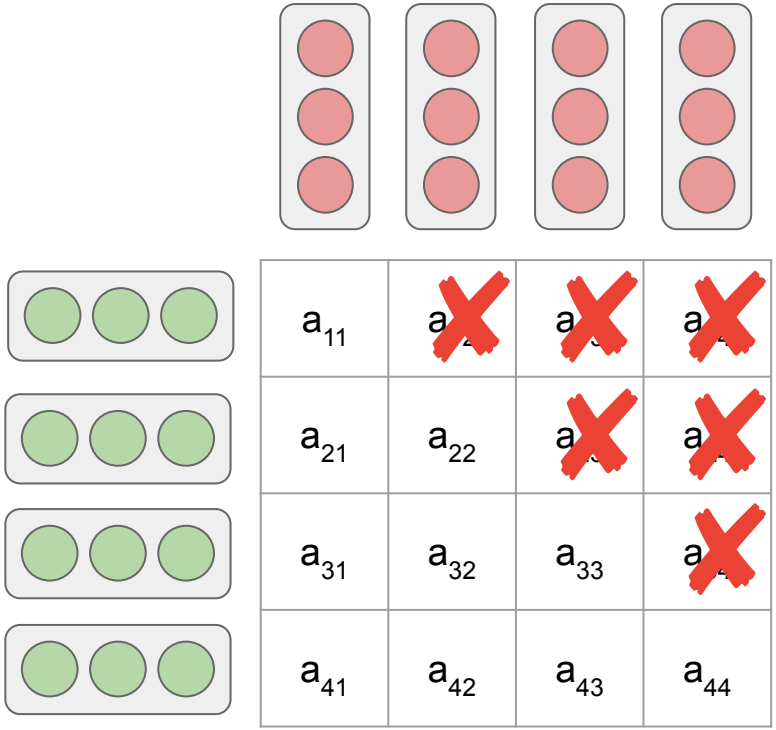
Transformer decoder (cont'd)



Transformer decoder (cont'd)

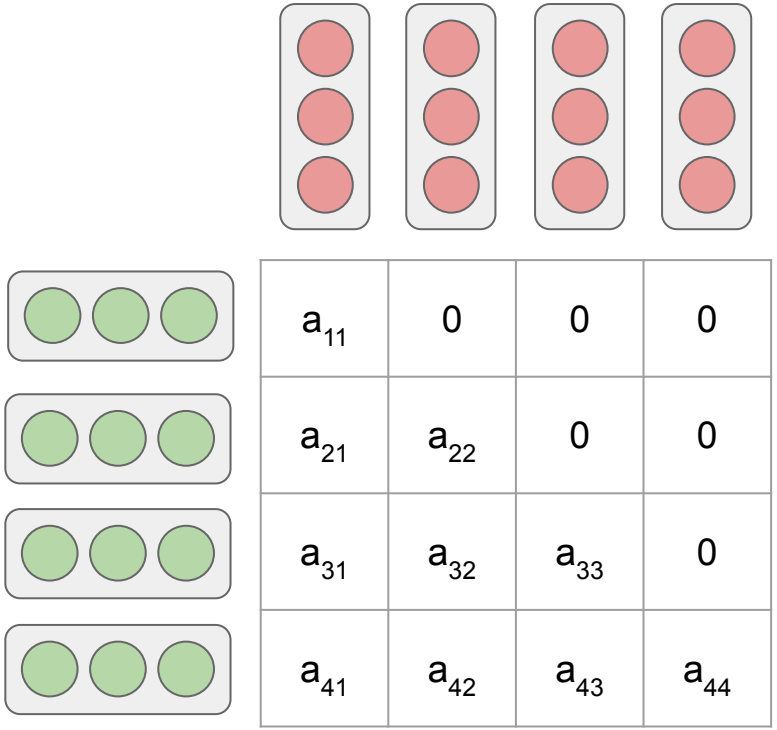


Self-attention in the decoder



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

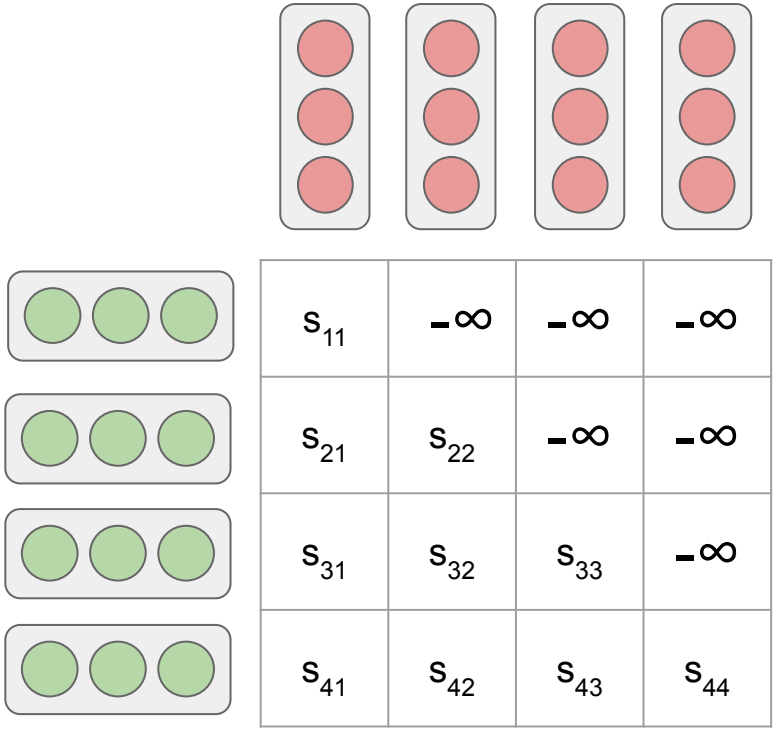
Self-attention in the decoder (cont'd)



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)



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

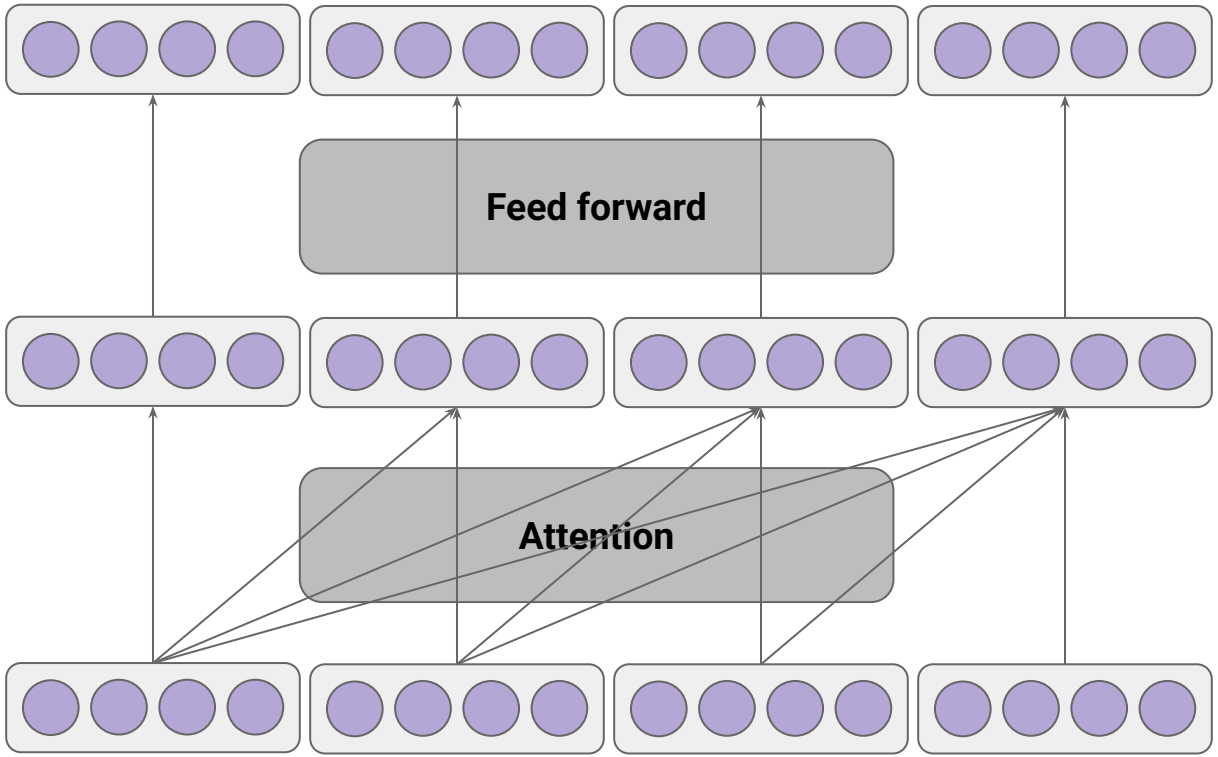
Softmax function

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

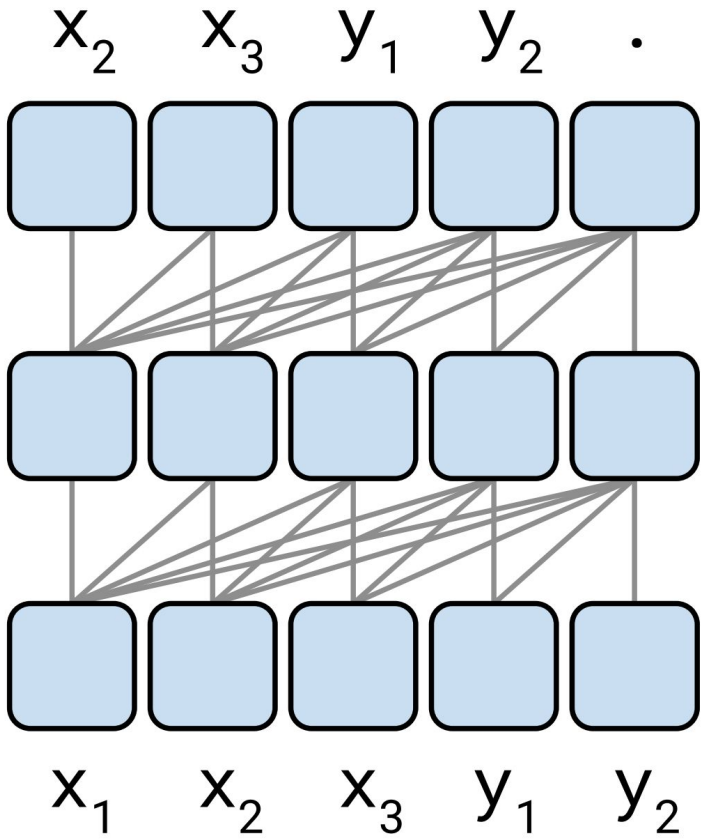
$$\text{softmax}(y) = \left[\frac{e^{y_1}}{\sum e^y}, \frac{e^{y_2}}{\sum e^y}, \ldots, \frac{e^{y_V}}{\sum e^y} \right]$$

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

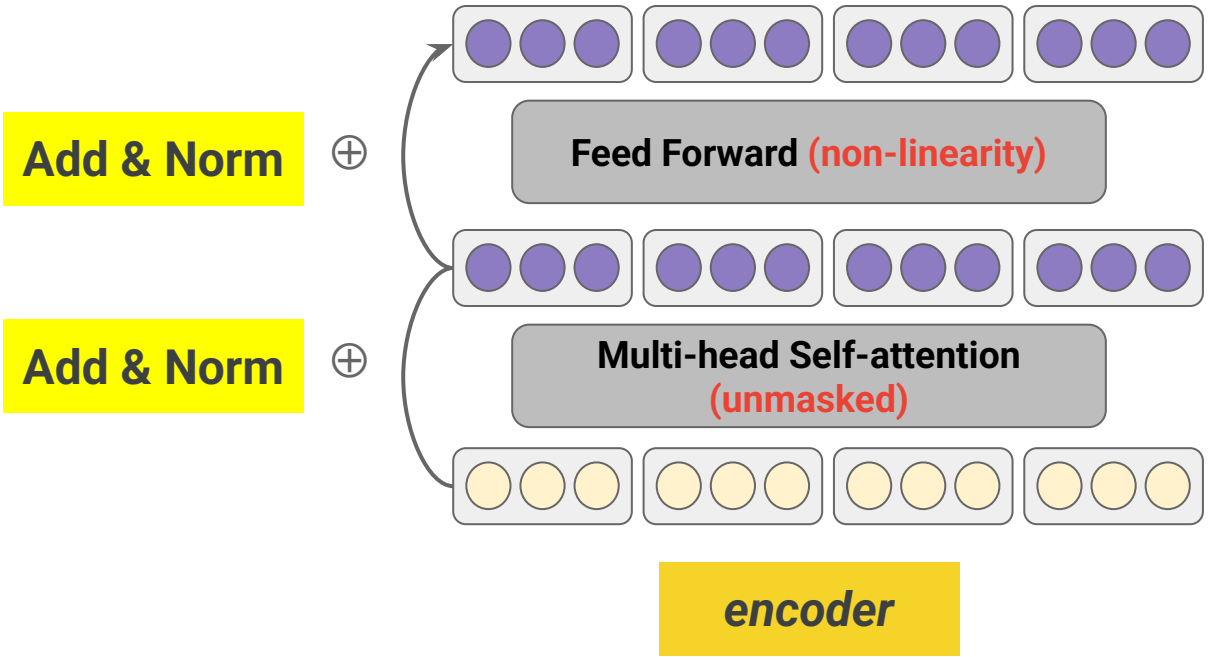
Decoder (cont'd)



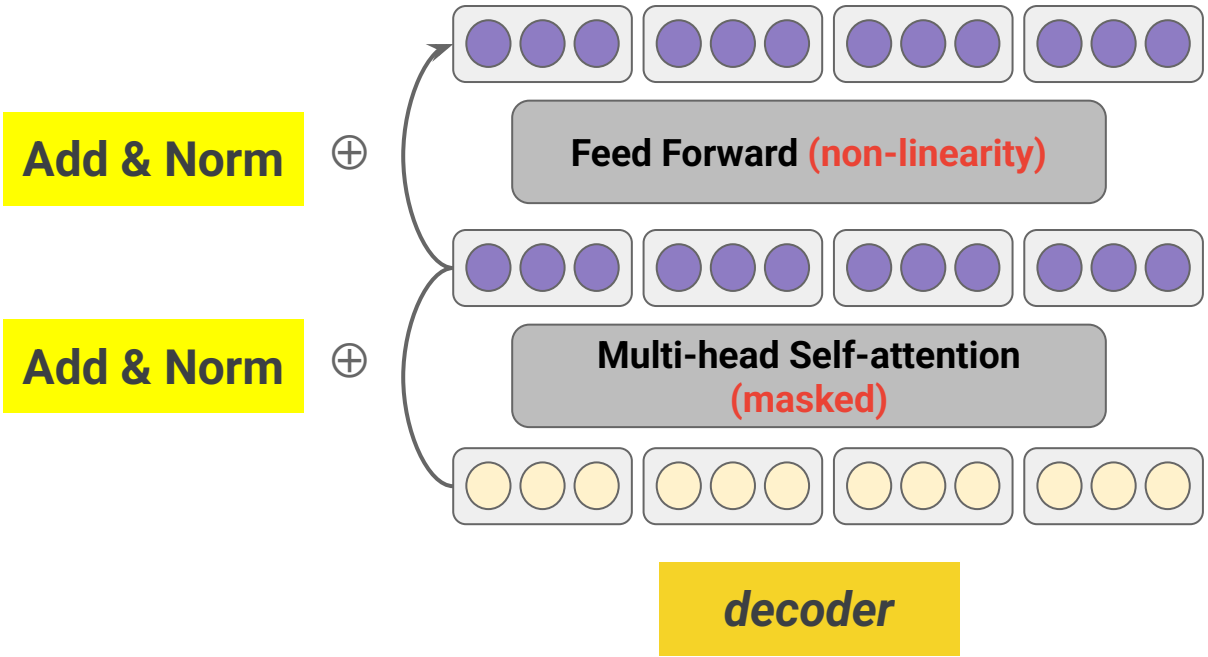
Decoder (cont'd)



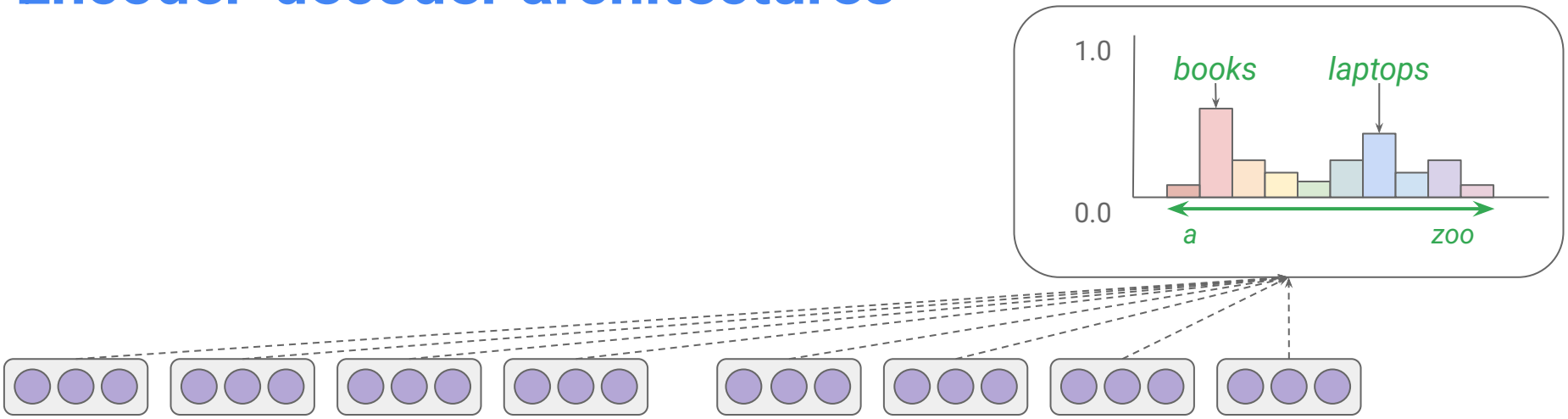
Encoder (one layer)



Decoder (one layer)



Encoder-decoder architectures

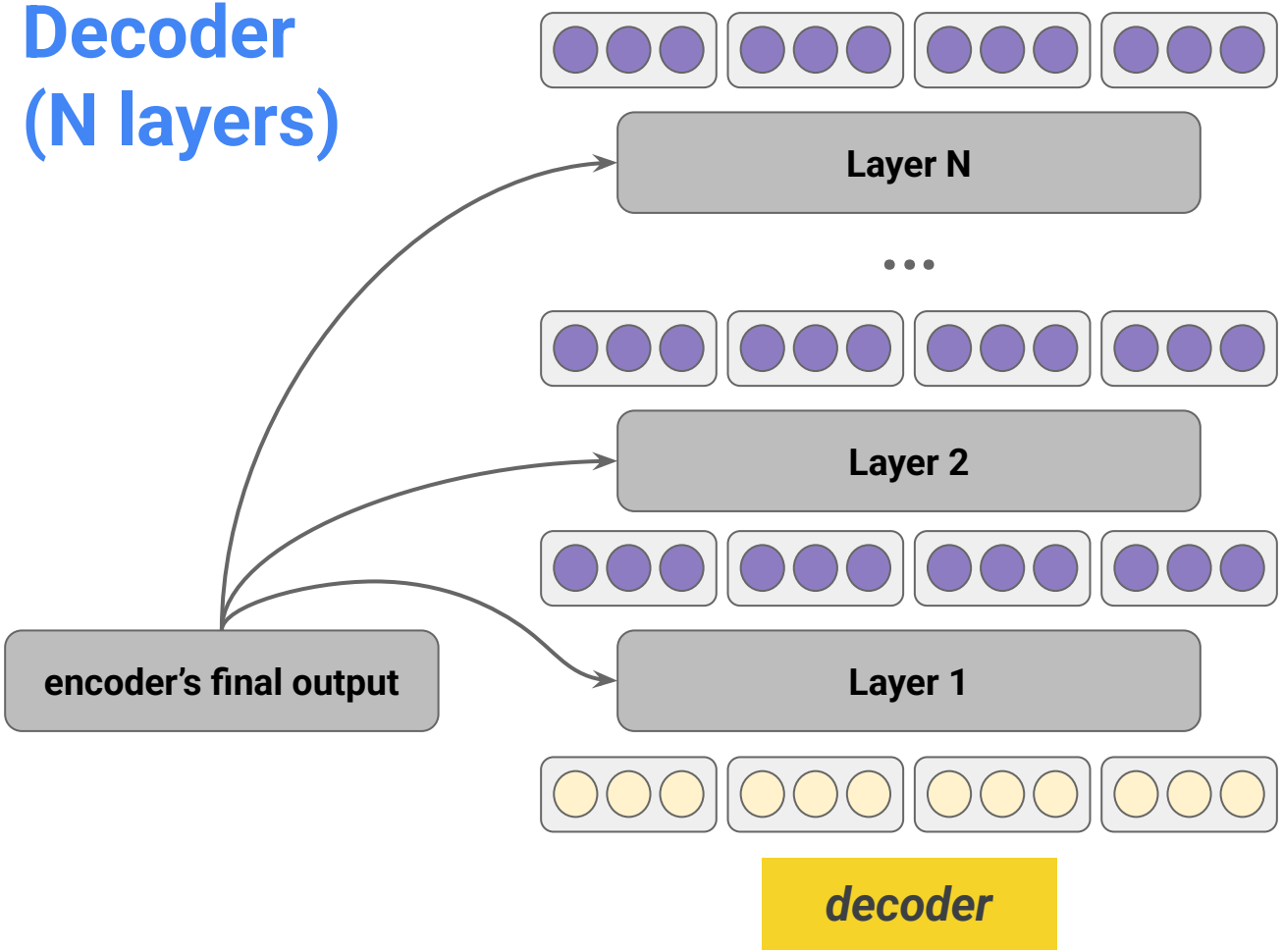


les étudiants ont ouvert leurs livres

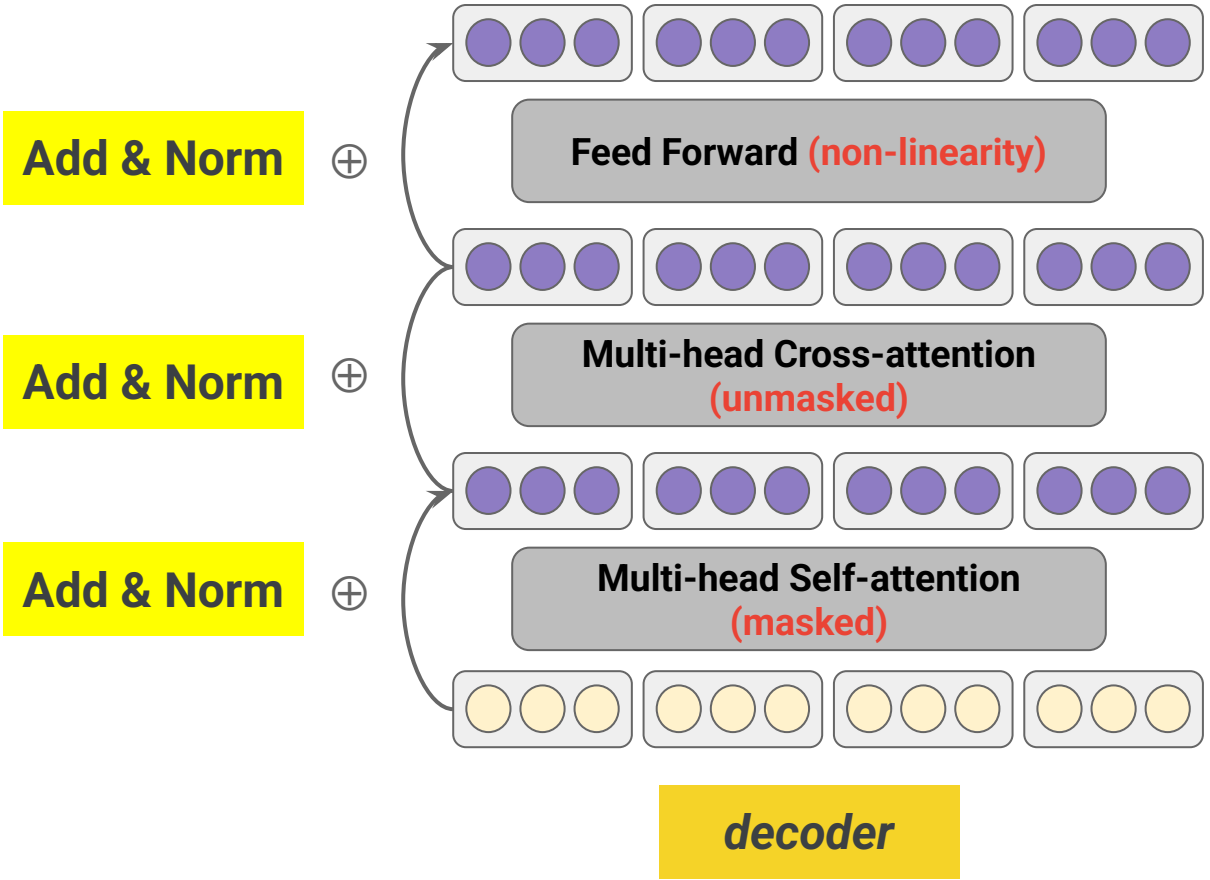
the *students* *opened* *their*

encoder ***decoder***

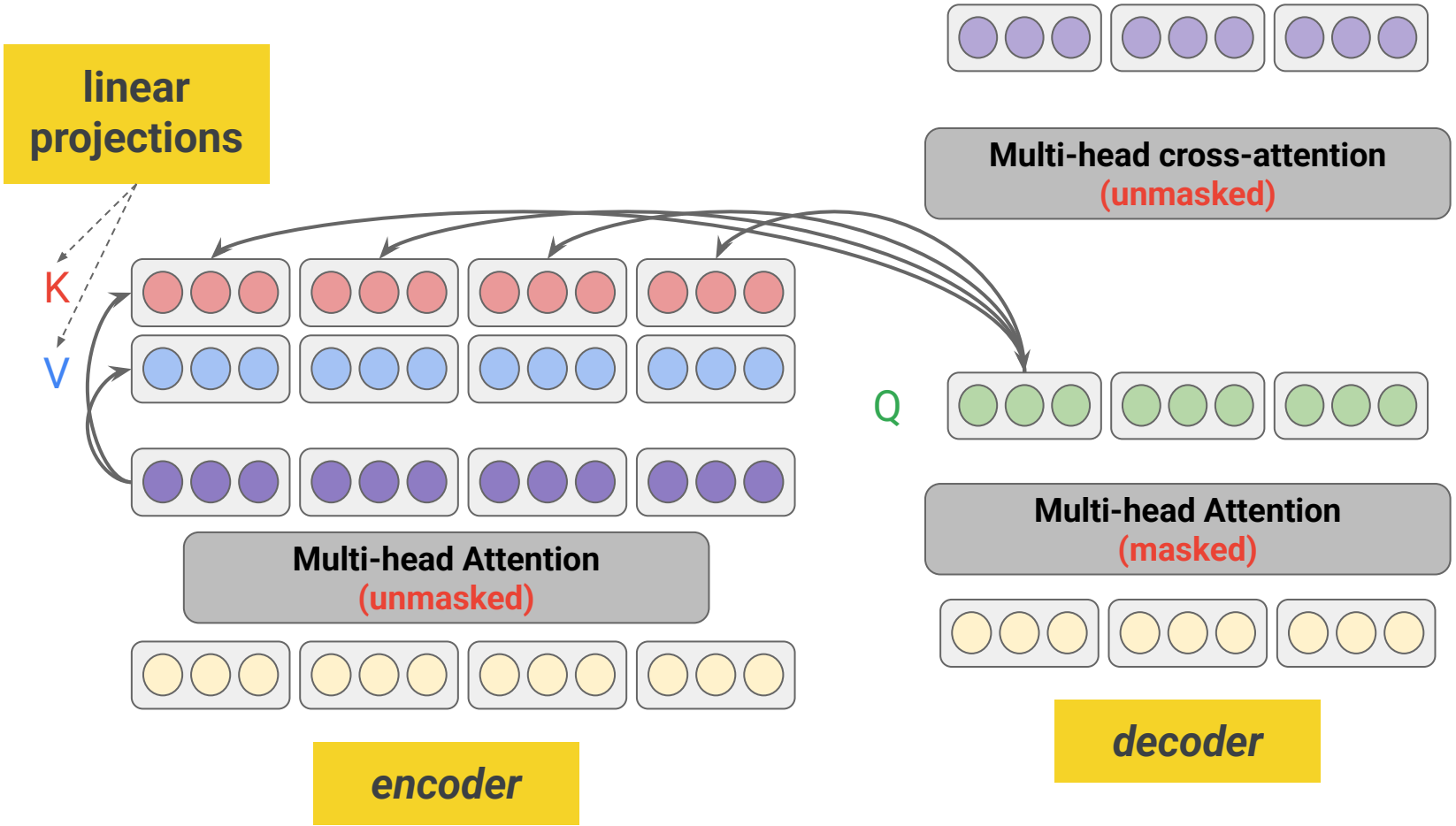
Decoder (N layers)



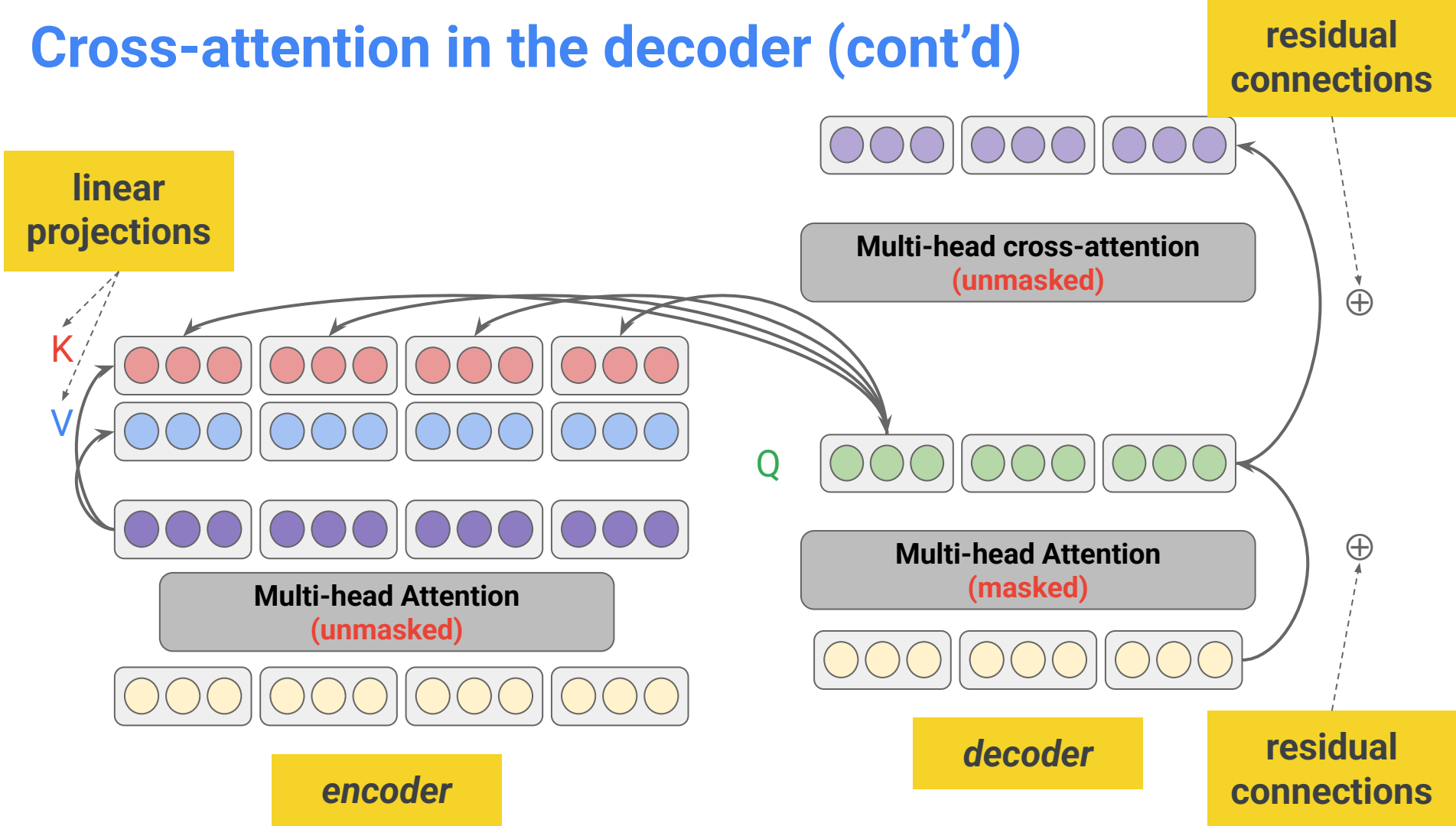
Decoder (one layer)

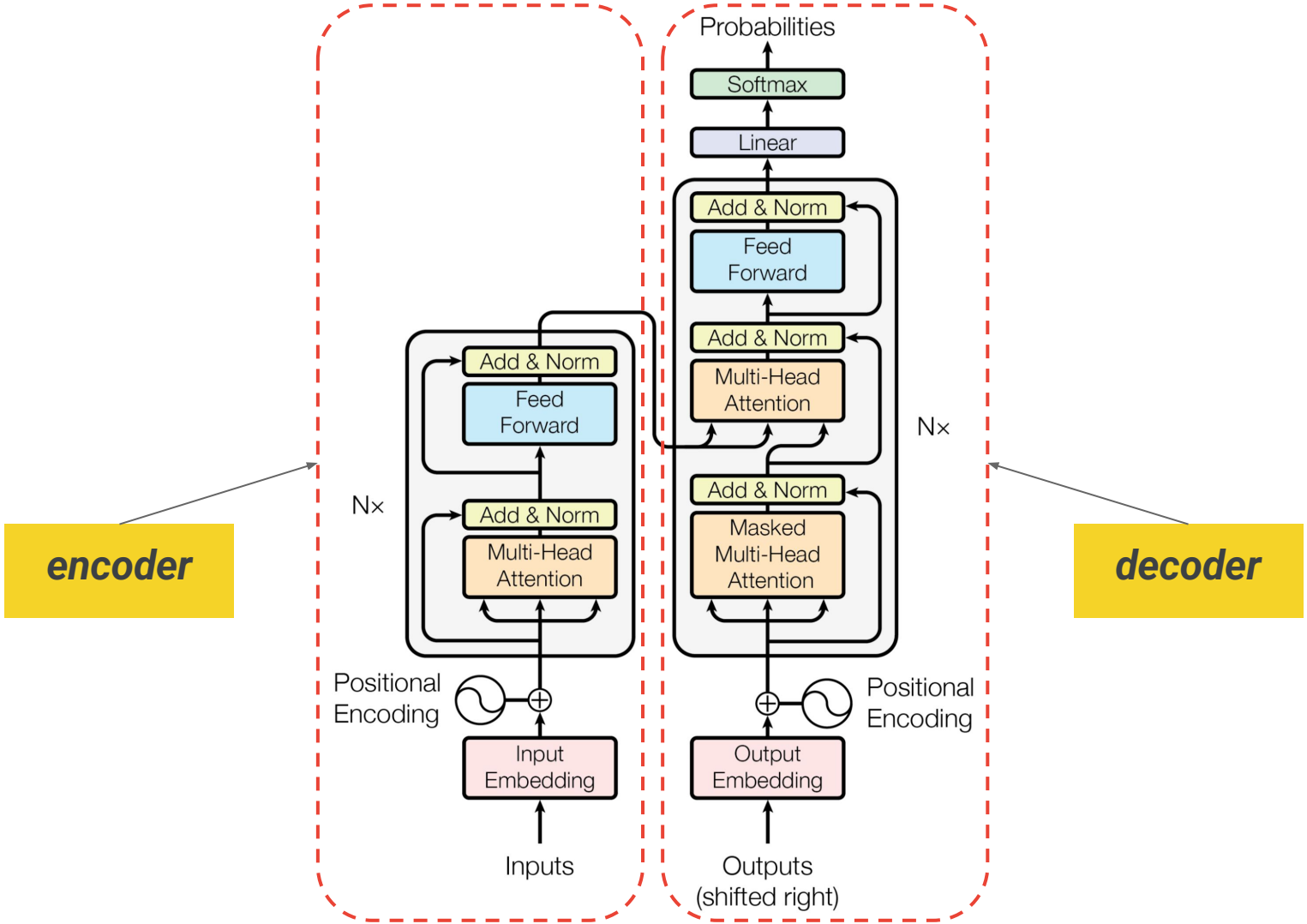


Cross-attention in the decoder



Cross-attention in the decoder (cont'd)




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

Model parameters (weights)

- Embedding matrices
  - Word embeddings, positional embeddings, segment embeddings
- Weight matrices
  - E.g., W_Q , W_K, W_V, W_O
- Bias terms

Bias term

$$h = \sigma(Wx + b)$$

bias term

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

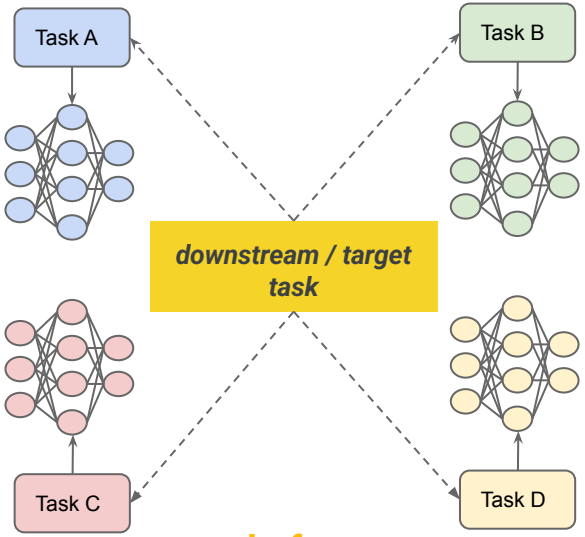
Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**

Google AI Language

{jacobdevlin,mingweichang,kentonl,kristout}@google.com

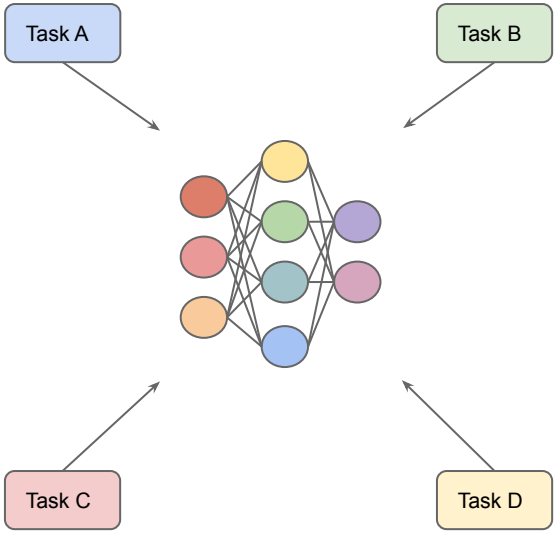
A learning paradigm shift

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

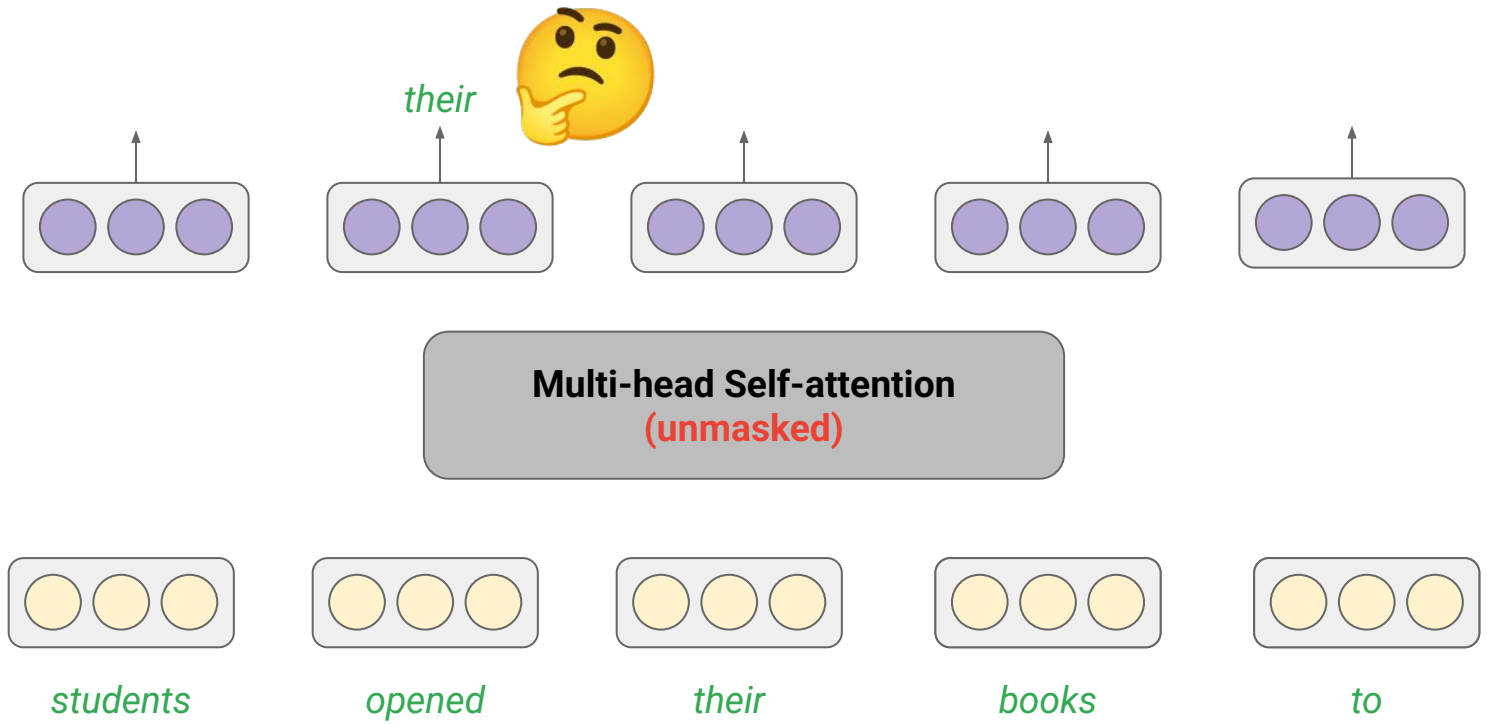
[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

BERT vs. ELMo

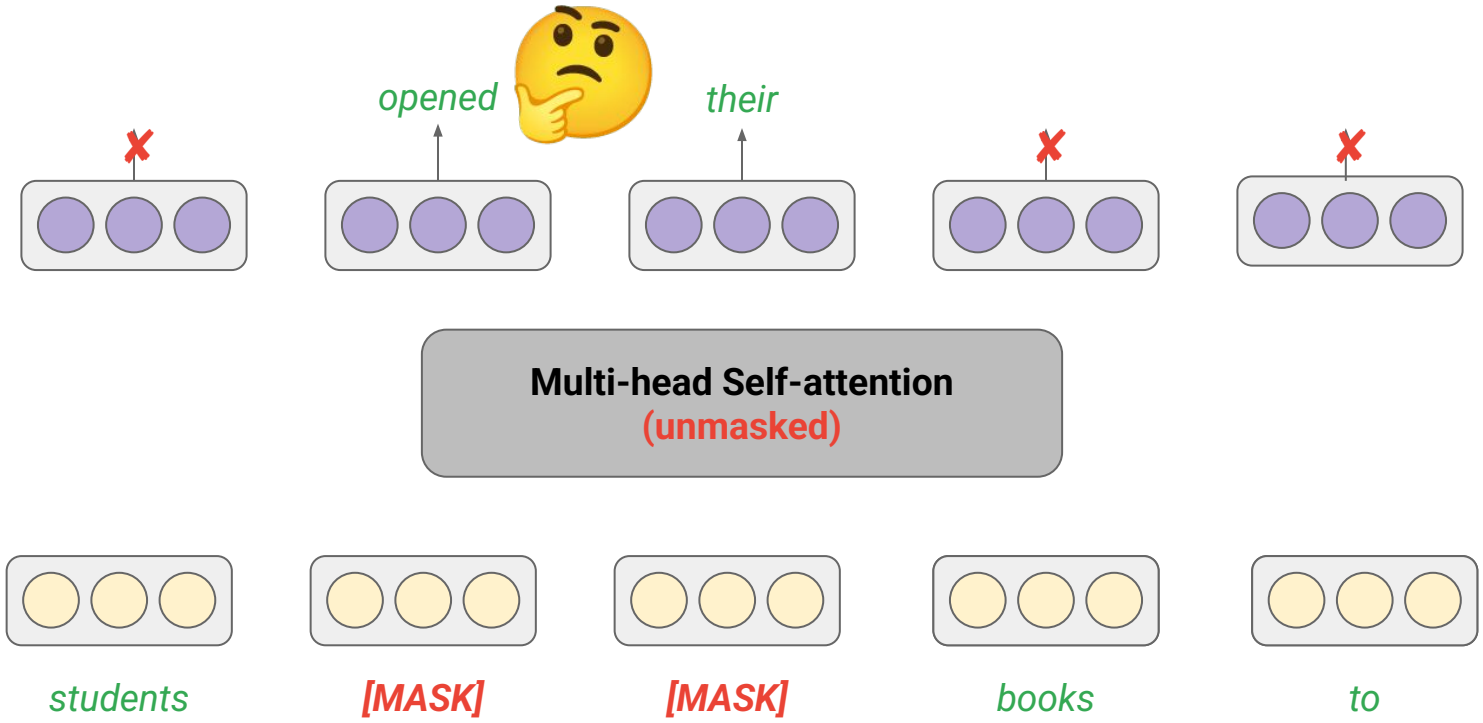
	BERT	ELMo
Model	Transformers	Bidirectional LSTM (Long Short-Term Memory, a variant of RNN)
Pre-training objective(s)	Masked language modeling + next sentence prediction	Left-to-right language modeling
Adaptation method	Fine-tuning	Feature-based (pretrained representations as additional features to task-specific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

Pretraining

Language modeling using a Transformer encoder

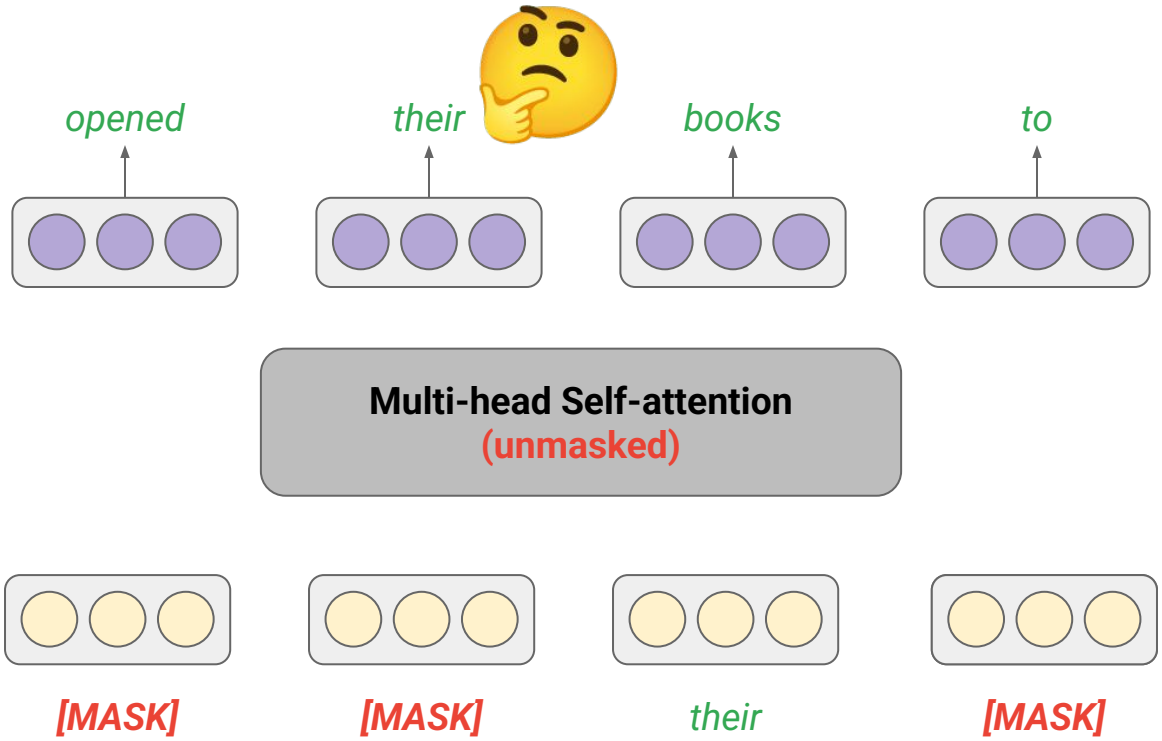


Masked language modeling



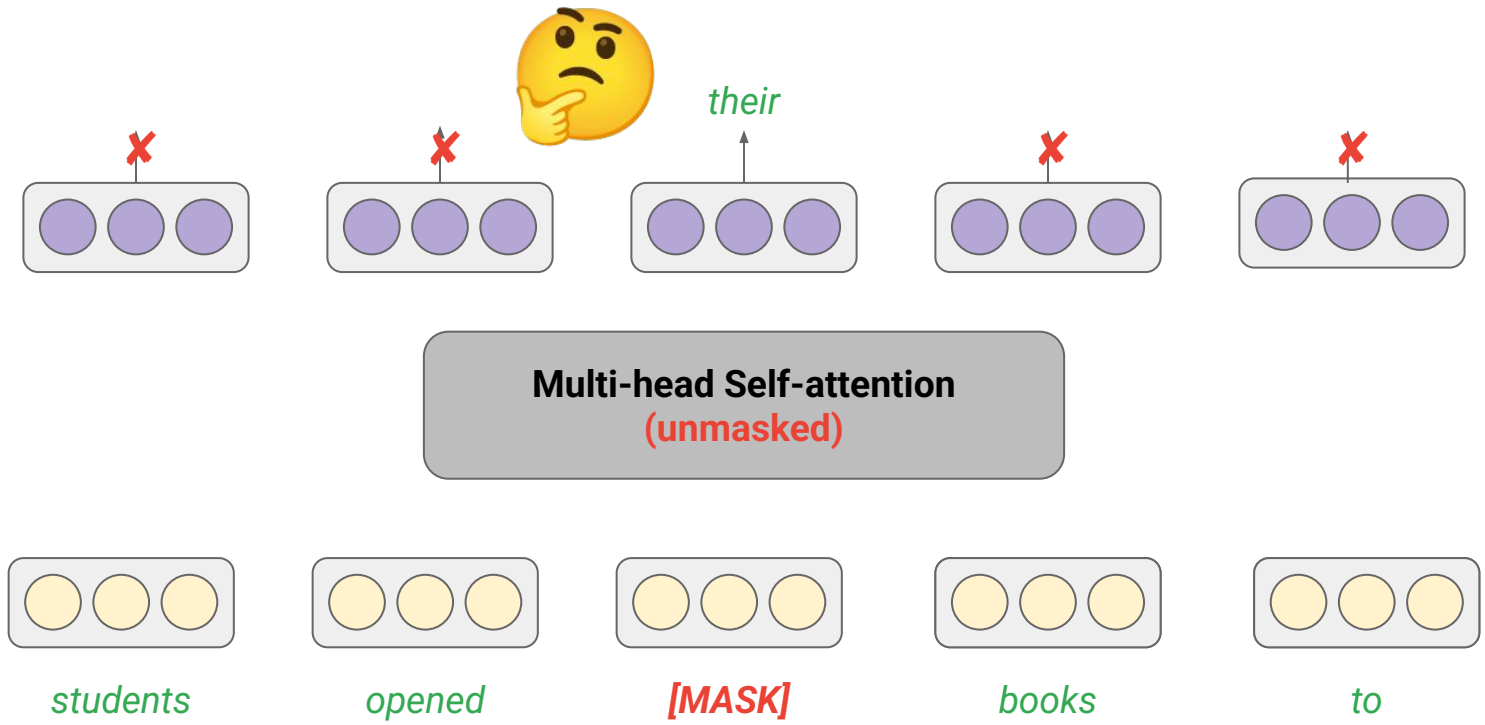
15% - 30% of all tokens in each sequence are masked at random

What if we mask more tokens?



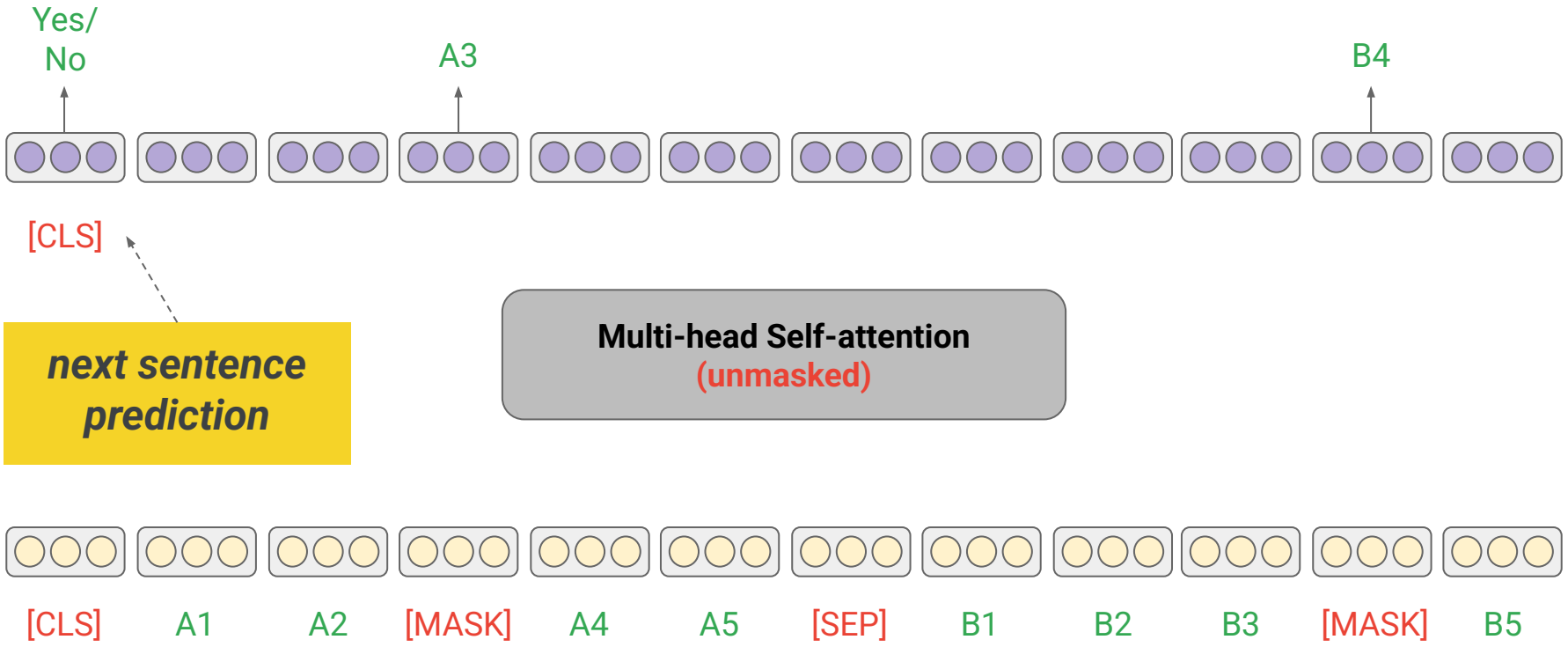
15% - 30% of all tokens in each sequence are masked at random

What if we mask less tokens?

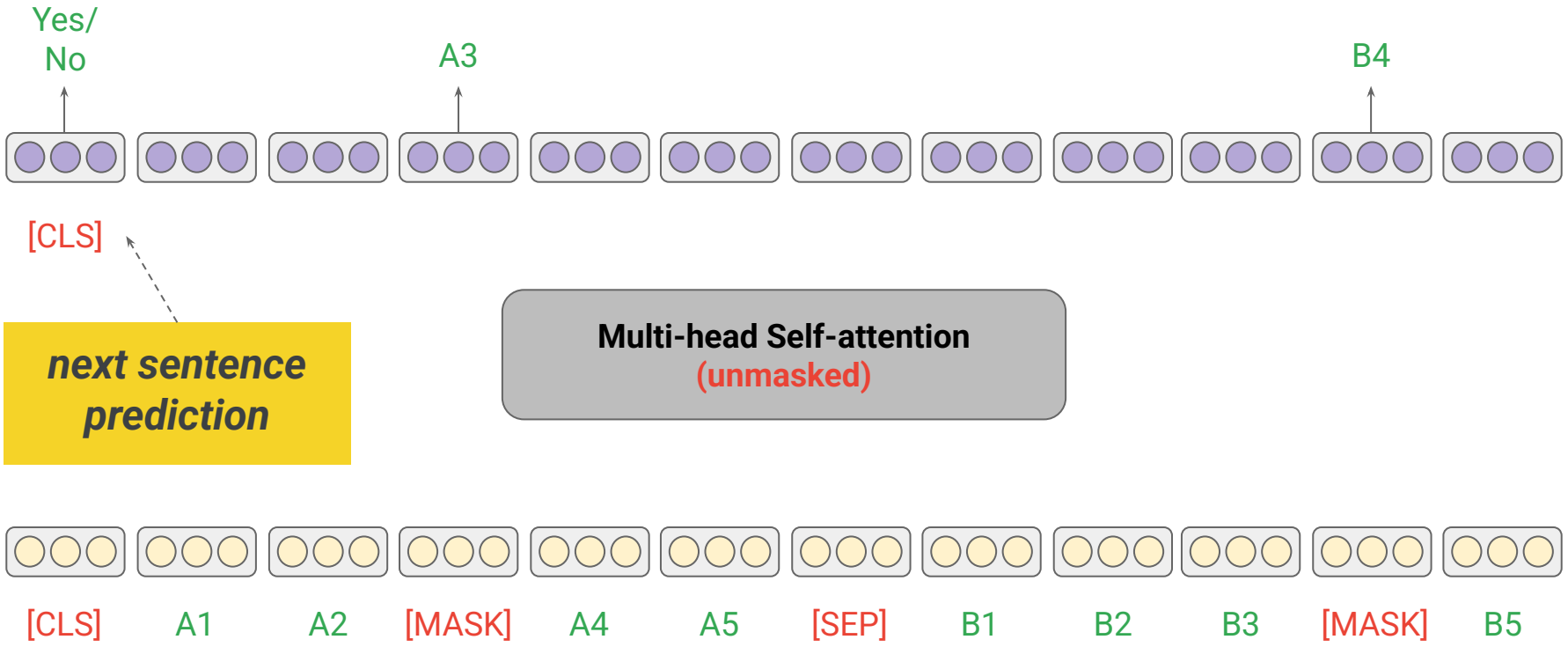


15% - 30% of all tokens in each sequence are masked at random

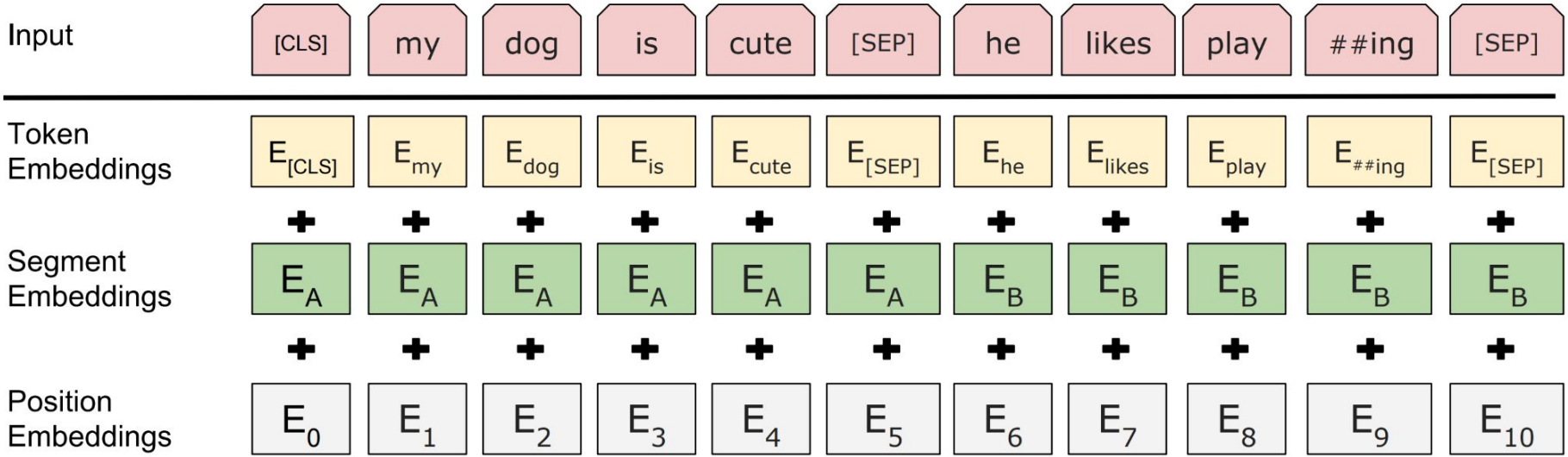
CLS & SEP tokens



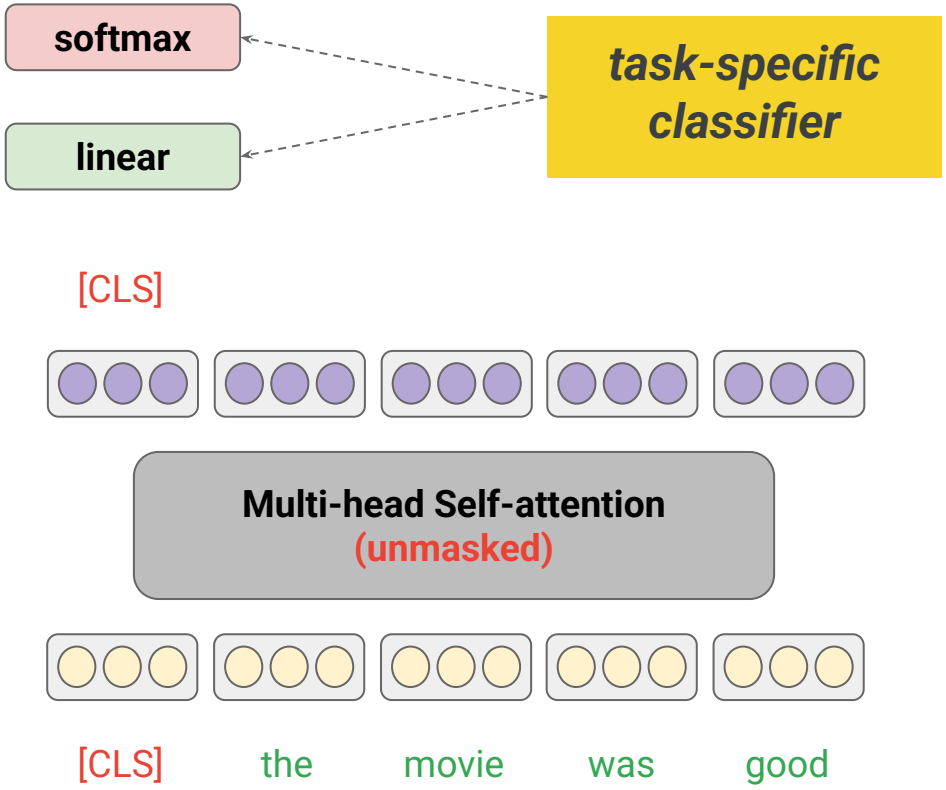
BERT Pretraining



BERT input representation

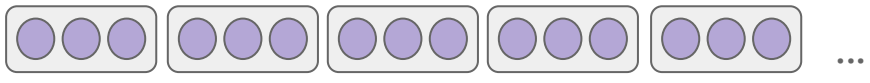


BERT Fine-tuning



T5 Pretraining: Span corruption

<X>, <Y>: sentinel tokens



Transformer encoder
(unmasked)

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

<X> for inviting <Y> last <EOS>

Transformer decoder
(masked)

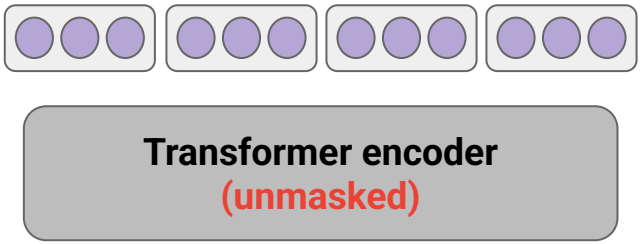
<BOS> <X> for inviting <Y> last

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

encoder

decoder

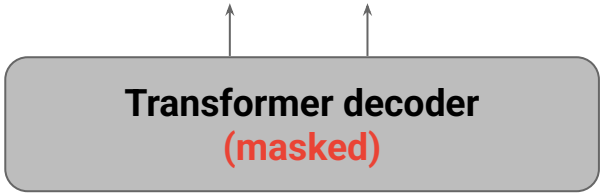
T5 Fine-tuning



sentiment analysis: this movie was good

encoder

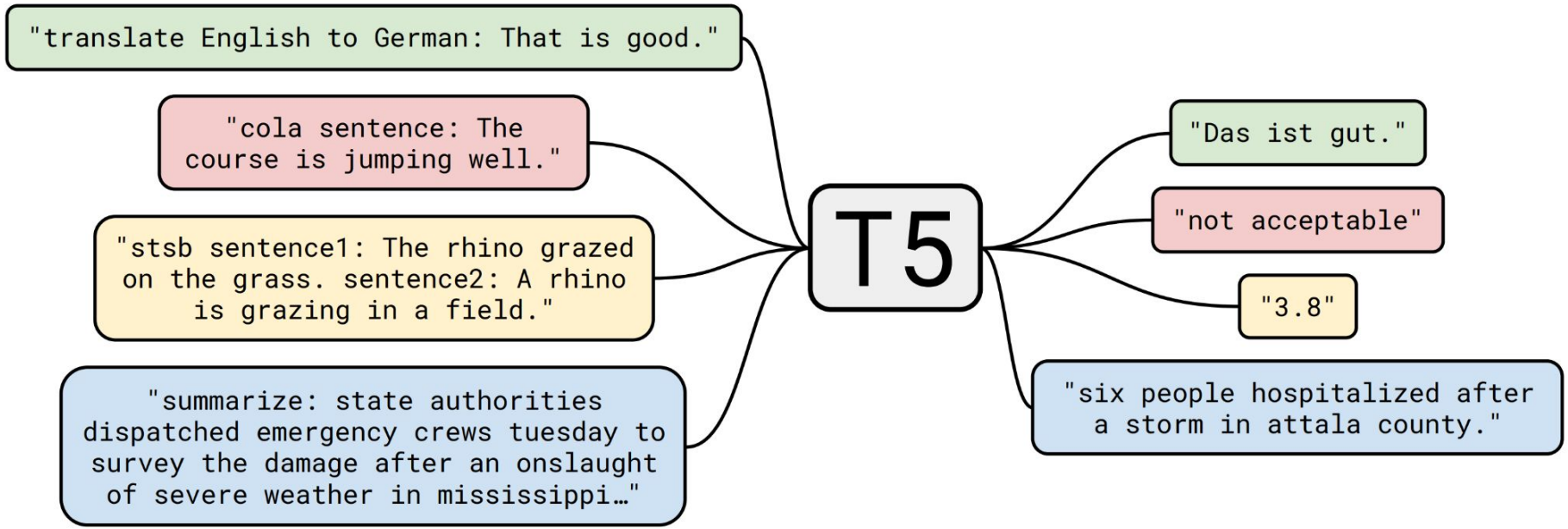
positive <EOS>



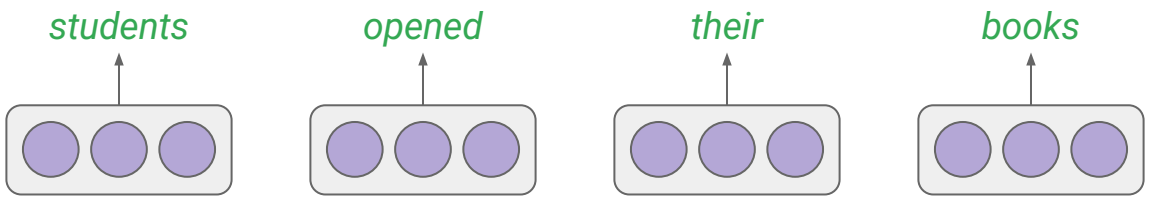
<BOS> positive

decoder

T5 facilitates multitask learning

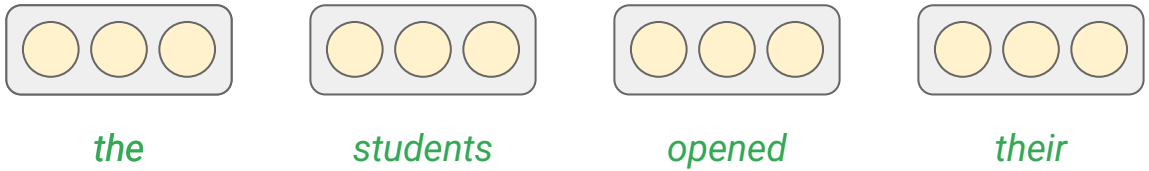


Decoder-only model



the architecture used in frontier LLMs

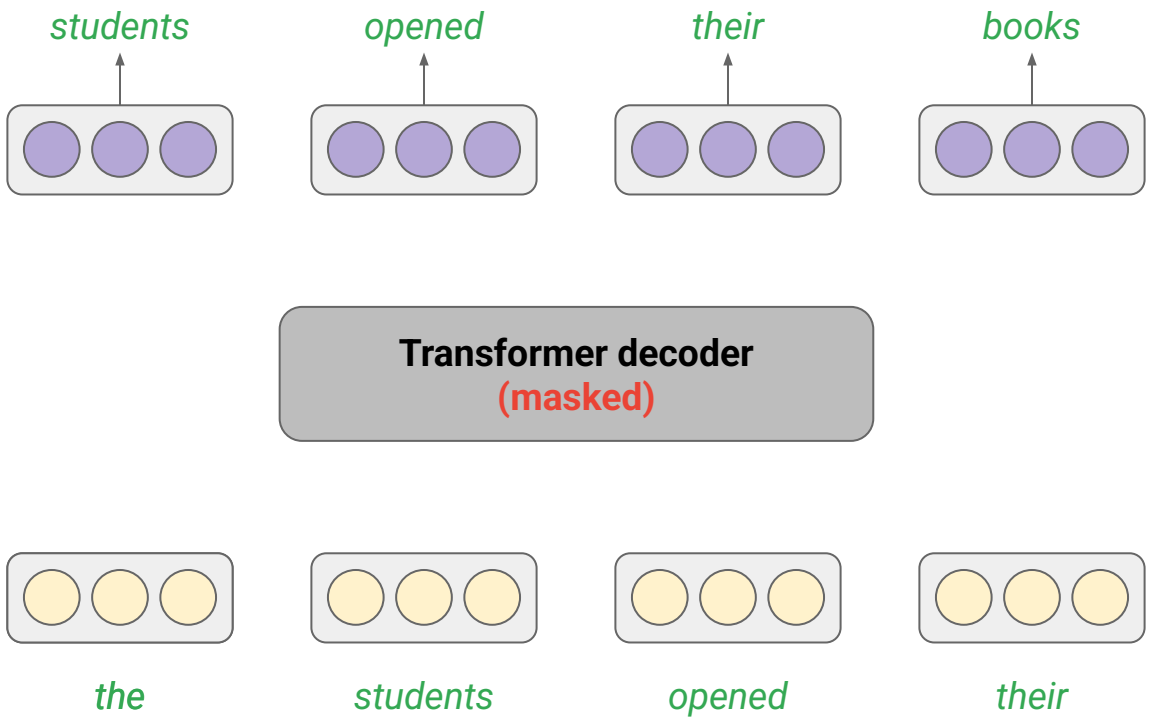
Transformer decoder
(masked)



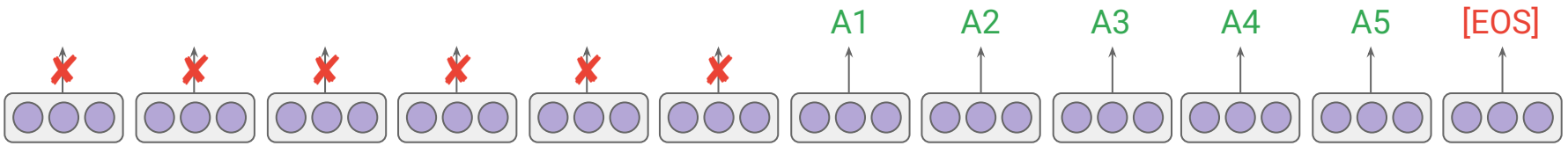
Note on cross-attention

- 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)



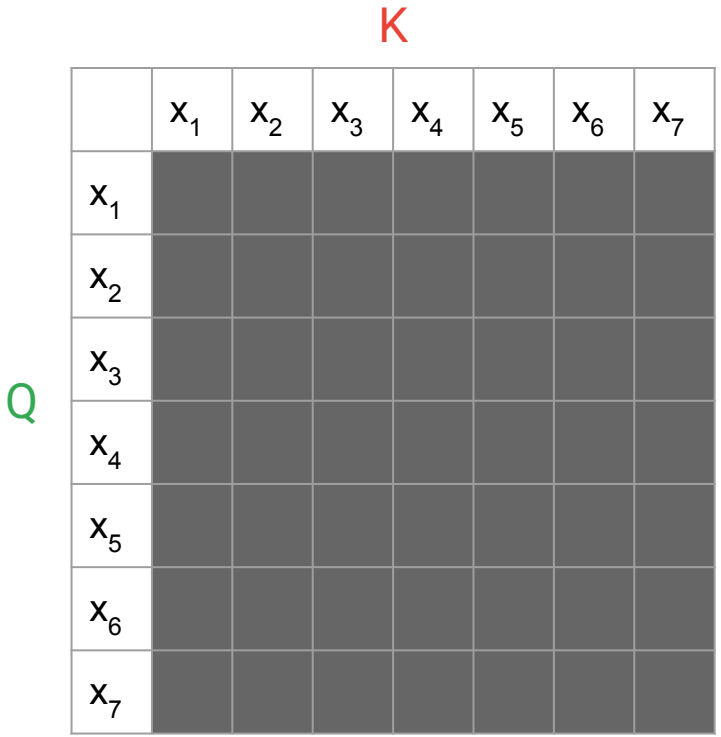
Training with prefix LM (decoder-only)



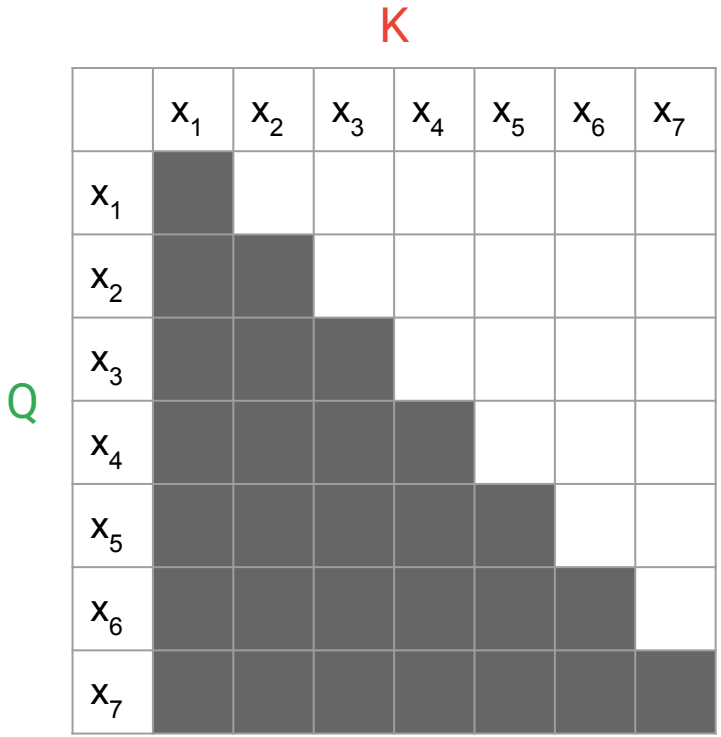
the architecture used in frontier LLMs

Transformer decoder
(partially masked)

Different attention mask patterns

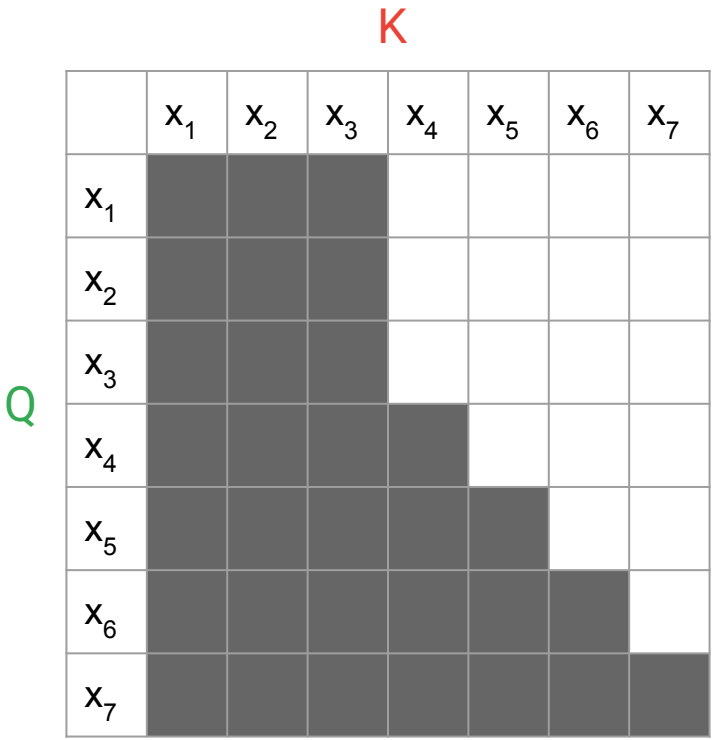


fully-visible



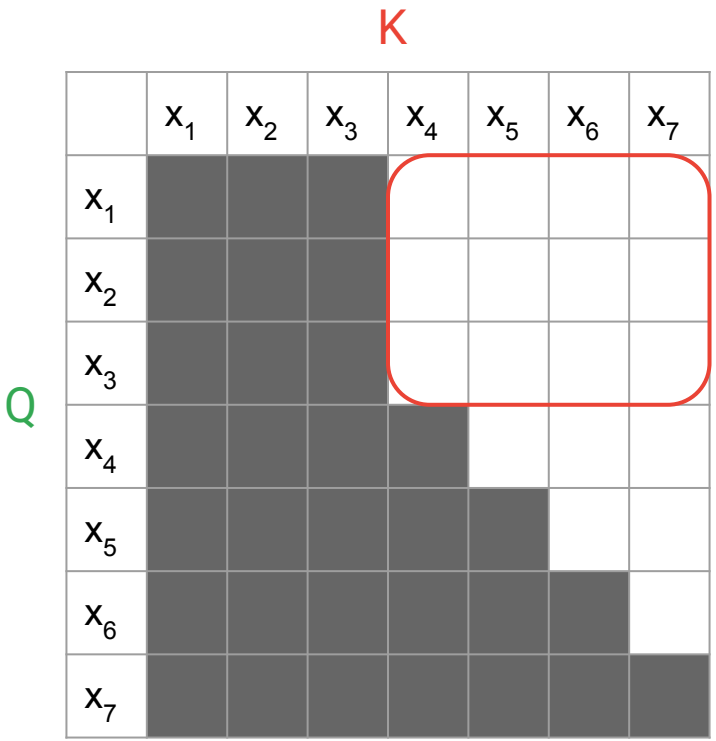
causal

Different attention mask patterns (cont'd)



Prefix LM

Different attention mask patterns (cont'd)



Why masking here?

Thank you!