Transformers (cont'd) & Pretraining scaling

CS 6804: Frontier AI Systems

Spring 2026

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

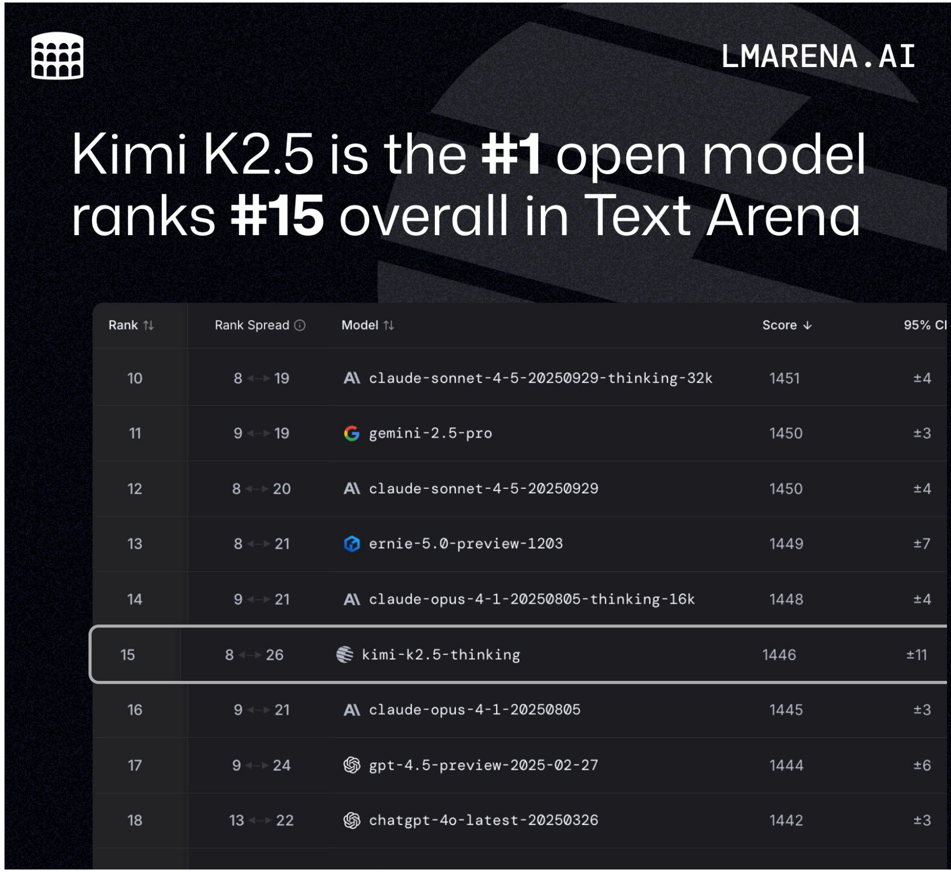
Tu Vu

VIRGINIA TECH.

Logistics

- Student presentations
  - Will be finalized EOD this Friday
- Homework 0 is on its way (for extra credits only)

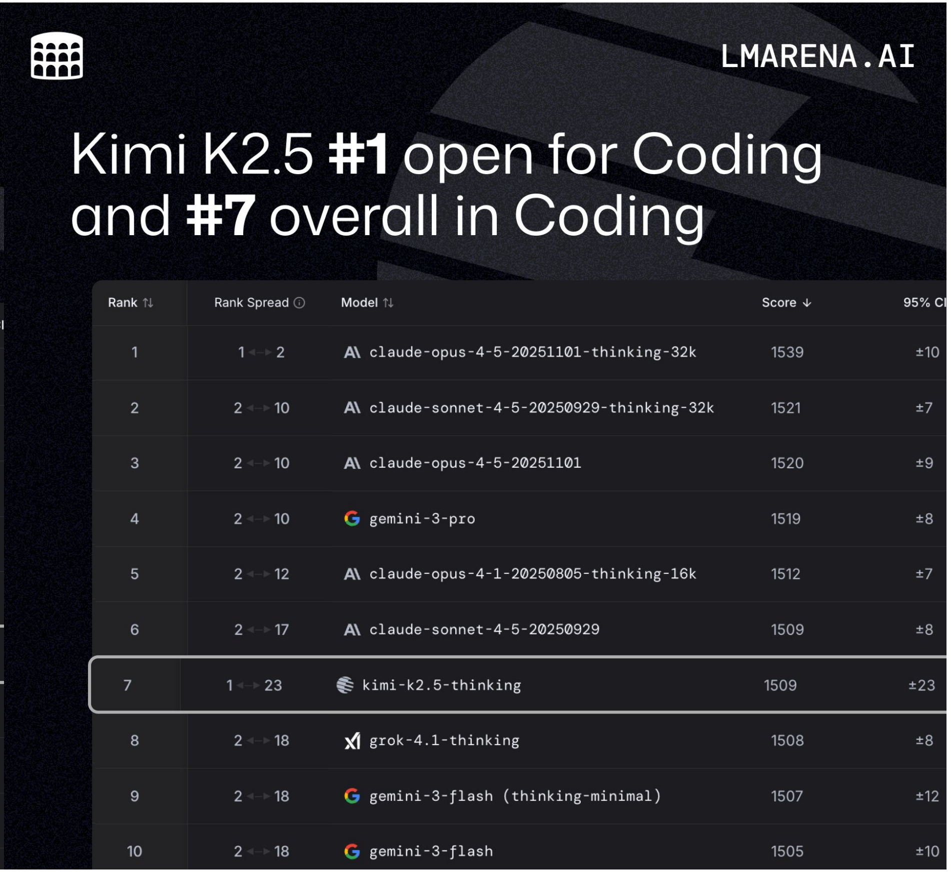
AI news



LMARENA.AI

Kimi K2.5 is the **#1** open model ranks **#15** overall in Text Arena

Rank	Rank Spread	Model	Score	95% CI
10	8 ↔ 19	claude-sonnet-4-5-20250929-thinking-32k	1451	±4
11	9 ↔ 19	gemini-2.5-pro	1450	±3
12	8 ↔ 20	claude-sonnet-4-5-20250929	1450	±4
13	8 ↔ 21	ernie-5.0-preview-1203	1449	±7
14	9 ↔ 21	claude-opus-4-1-20250805-thinking-16k	1448	±4
15	8 ↔ 26	kimi-k2.5-thinking	1446	±11
16	9 ↔ 21	claude-opus-4-1-20250805	1445	±3
17	9 ↔ 24	gpt-4.5-preview-2025-02-27	1444	±6
18	13 ↔ 22	chatgpt-4o-latest-20250326	1442	±3

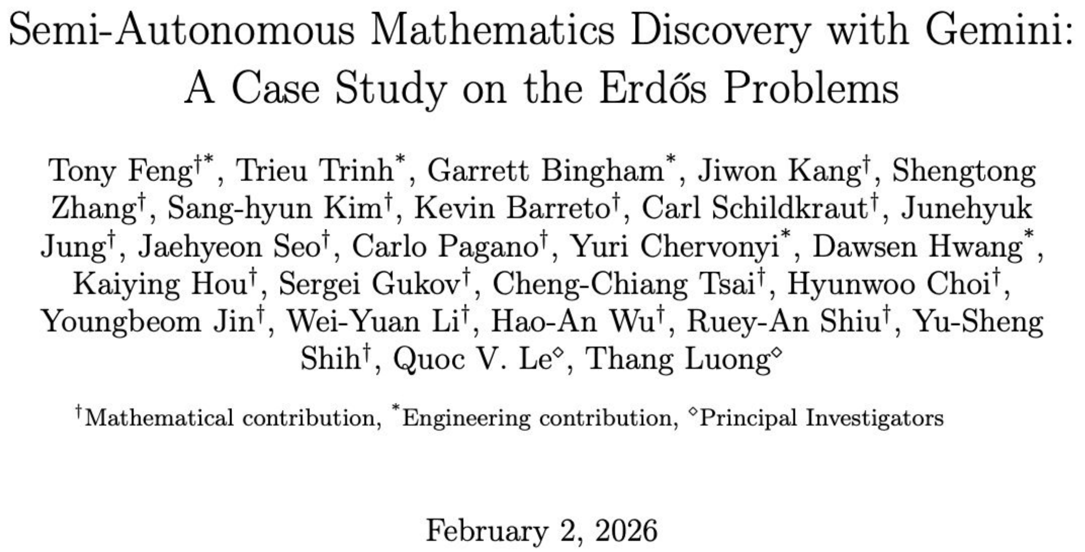


LMARENA.AI

Kimi K2.5 **#1** open for Coding and **#7** overall in Coding

Rank	Rank Spread	Model	Score	95% CI
1	1 ↔ 2	claude-opus-4-5-20251101-thinking-32k	1539	±10
2	2 ↔ 10	claude-sonnet-4-5-20250929-thinking-32k	1521	±7
3	2 ↔ 10	claude-opus-4-5-20251101	1520	±9
4	2 ↔ 10	gemini-3-pro	1519	±8
5	2 ↔ 12	claude-opus-4-1-20250805-thinking-16k	1512	±7
6	2 ↔ 17	claude-sonnet-4-5-20250929	1509	±8
7	1 ↔ 23	kimi-k2.5-thinking	1509	±23
8	2 ↔ 18	grok-4.1-thinking	1508	±8
9	2 ↔ 18	gemini-3-flash (thinking-minimal)	1507	±12
10	2 ↔ 18	gemini-3-flash	1505	±10

Google DeepMind's Aletheia (based on Gemini Deep Think)

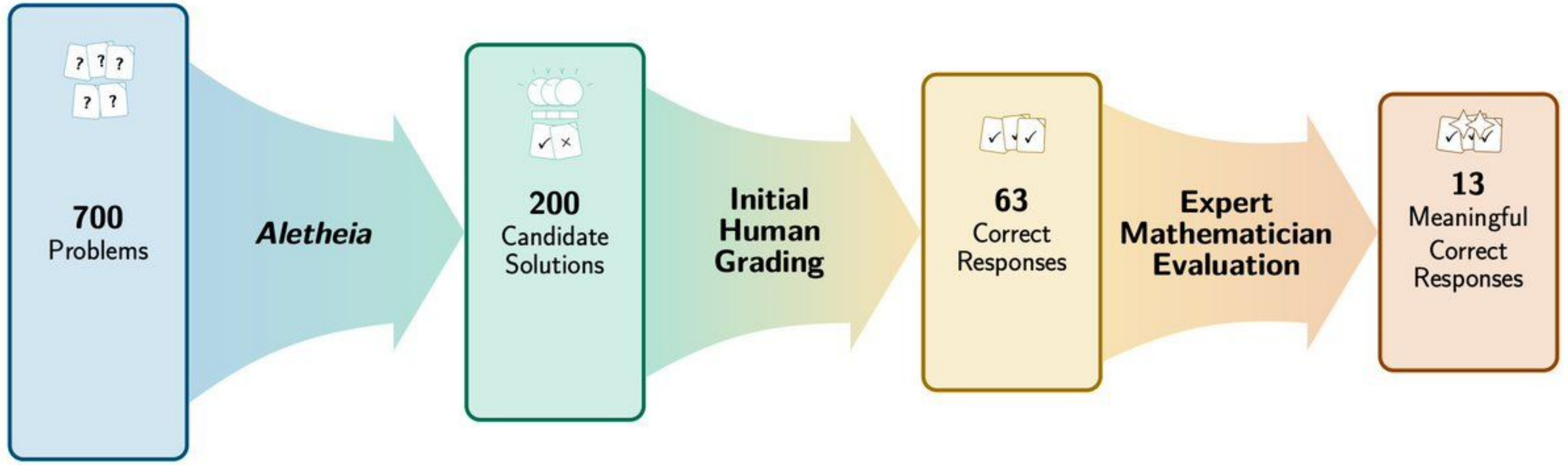


Semi-Autonomous Mathematics Discovery with Gemini: A Case Study on the Erdős Problems

Tony Feng[†*], Trieu Trinh[*], Garrett Bingham[*], Jiwon Kang[†], Shengtong Zhang[†], Sang-hyun Kim[†], Kevin Barreto[†], Carl Schildkraut[†], Junehyuk Jung[†], Jaehyeon Seo[†], Carlo Pagano[†], Yuri Chervonyi[*], Dawsen Hwang[*], Kaiying Hou[†], Sergei Gukov[†], Cheng-Chiang Tsai[†], Hyunwoo Choi[†], Youngbeom Jin[†], Wei-Yuan Li[†], Hao-An Wu[†], Ruey-An Shiu[†], Yu-Sheng Shih[†], Quoc V. Le[◇], Thang Luong[◇]

[†]Mathematical contribution, [*]Engineering contribution, [◇]Principal Investigators

February 2, 2026

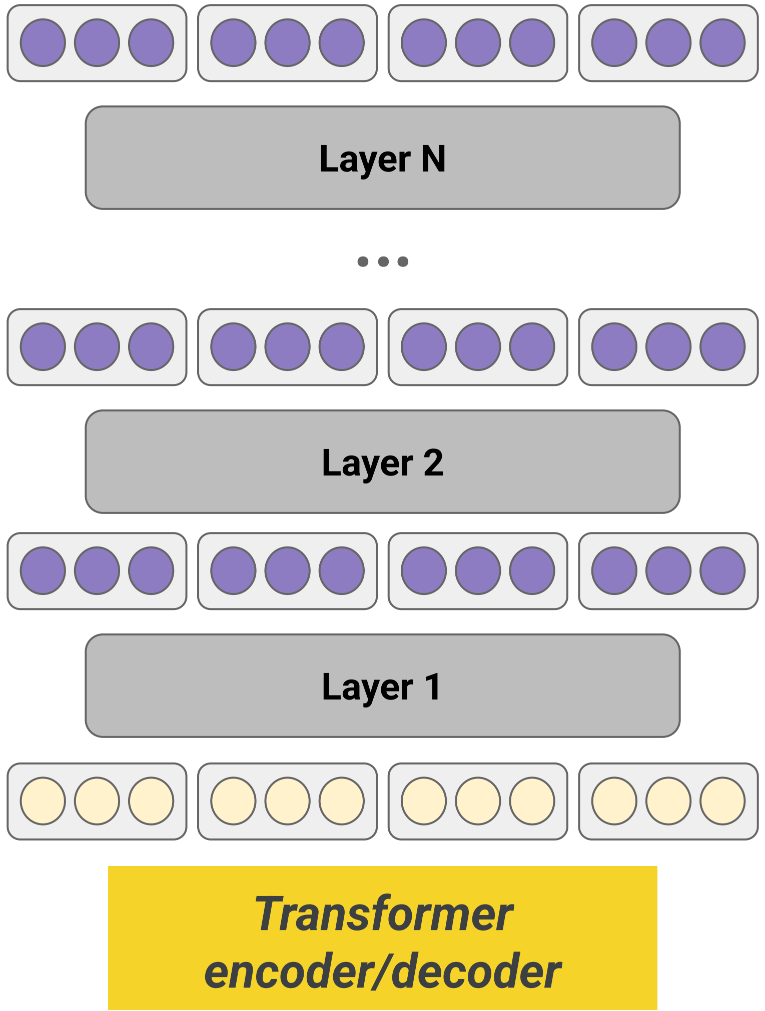


Transformers review

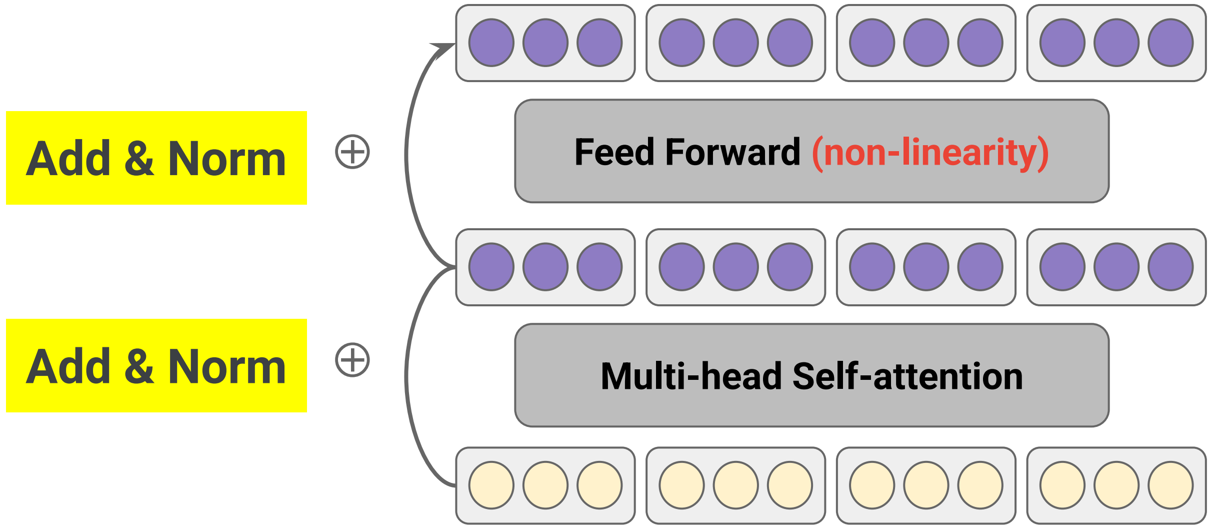
Different Transformers architectures

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

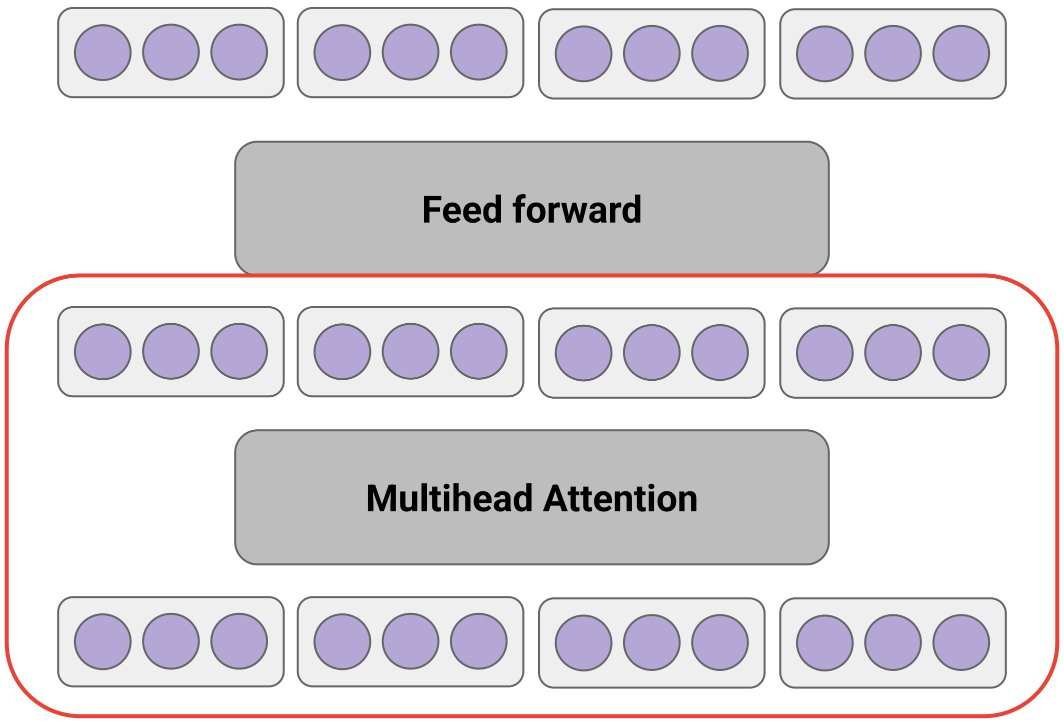
N layers



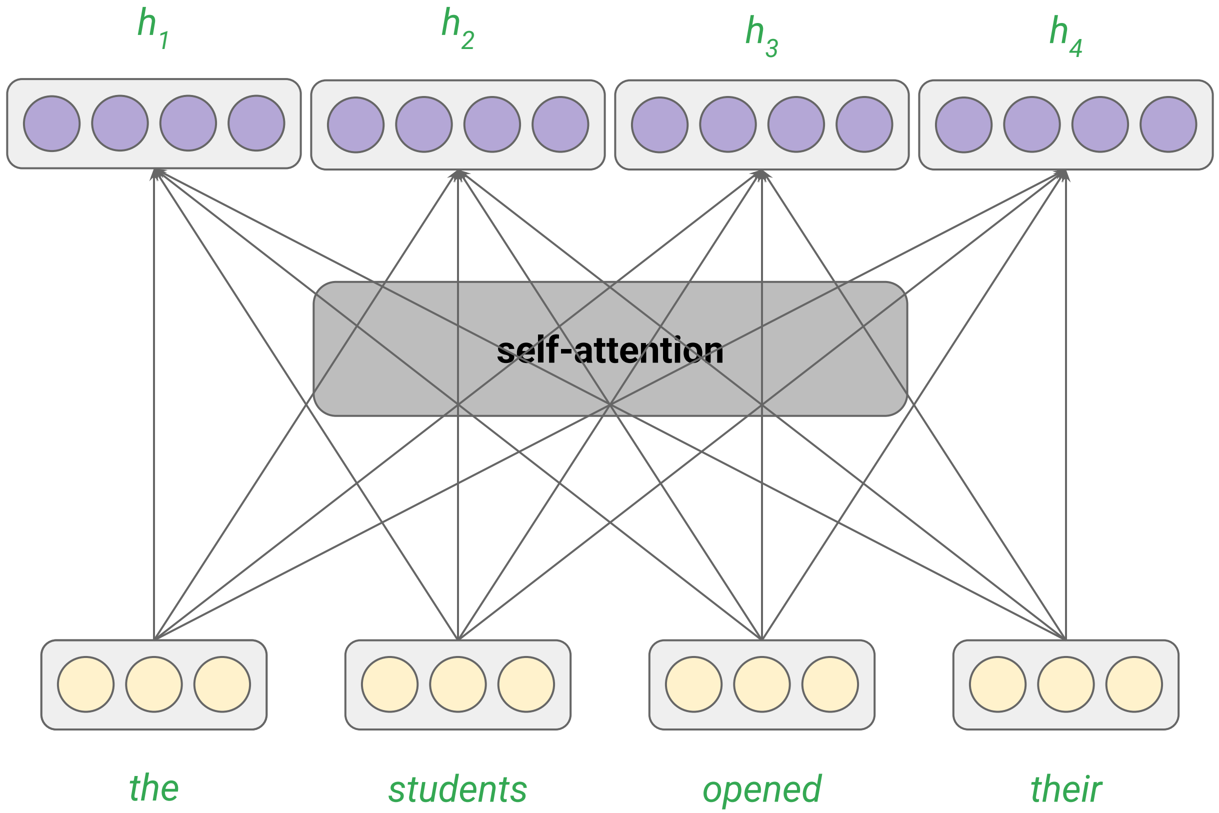
Transformer block (one layer)



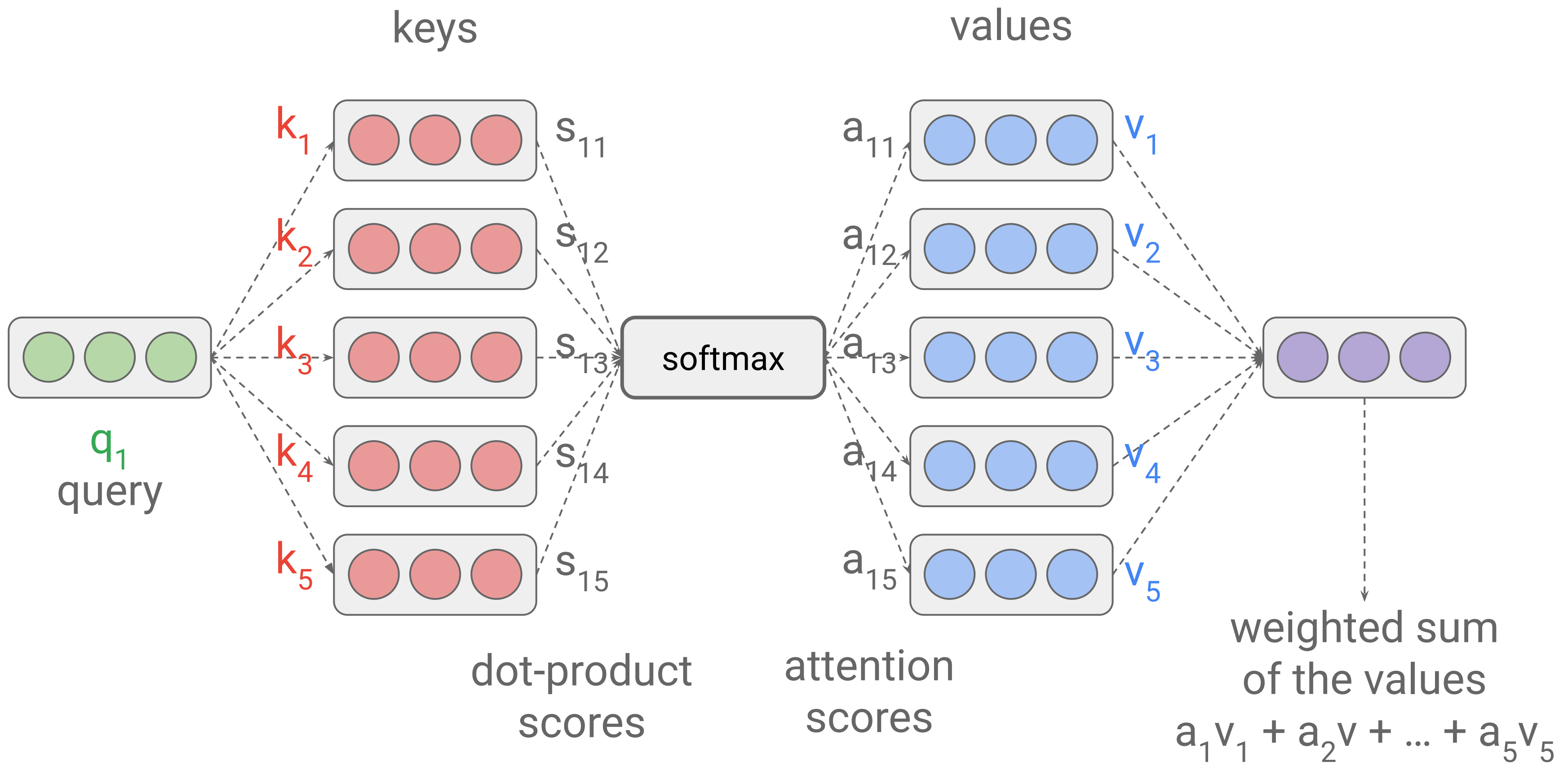
Transformer block



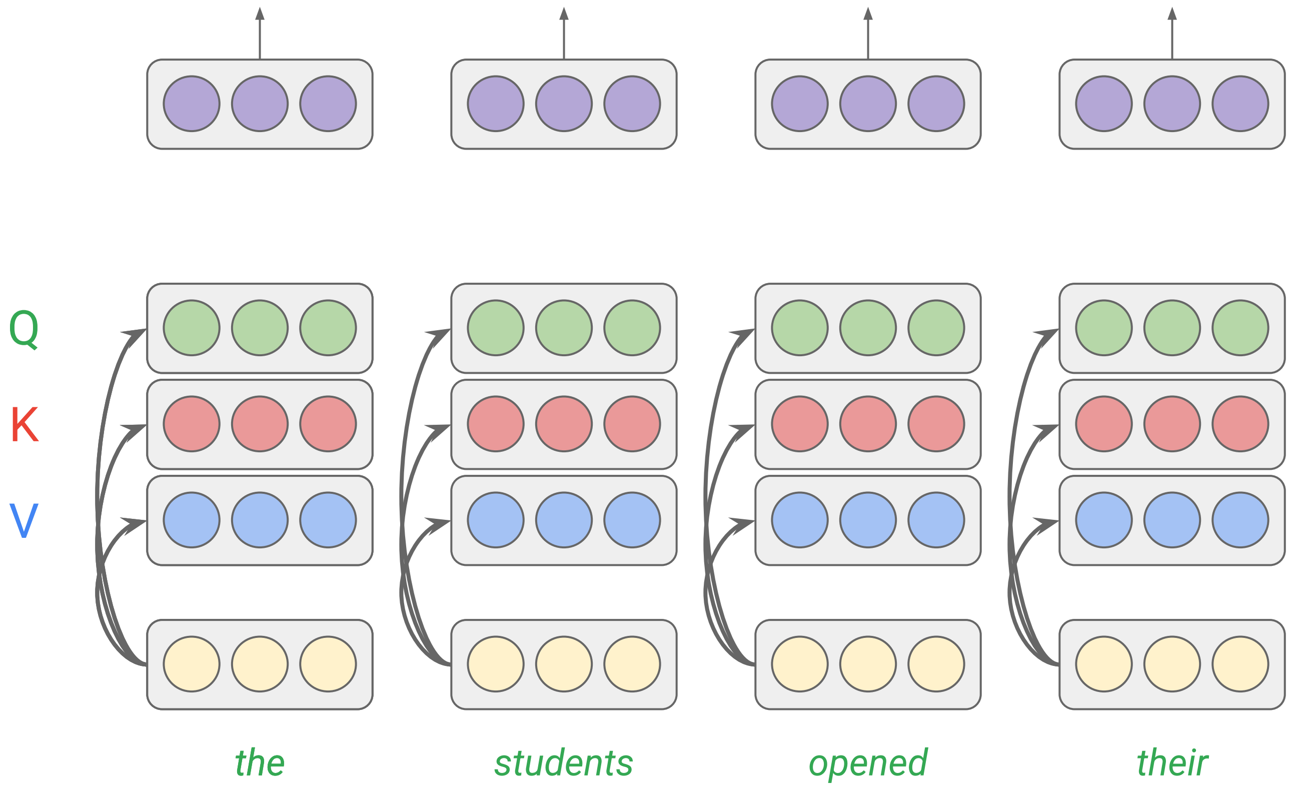
Self-attention



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)



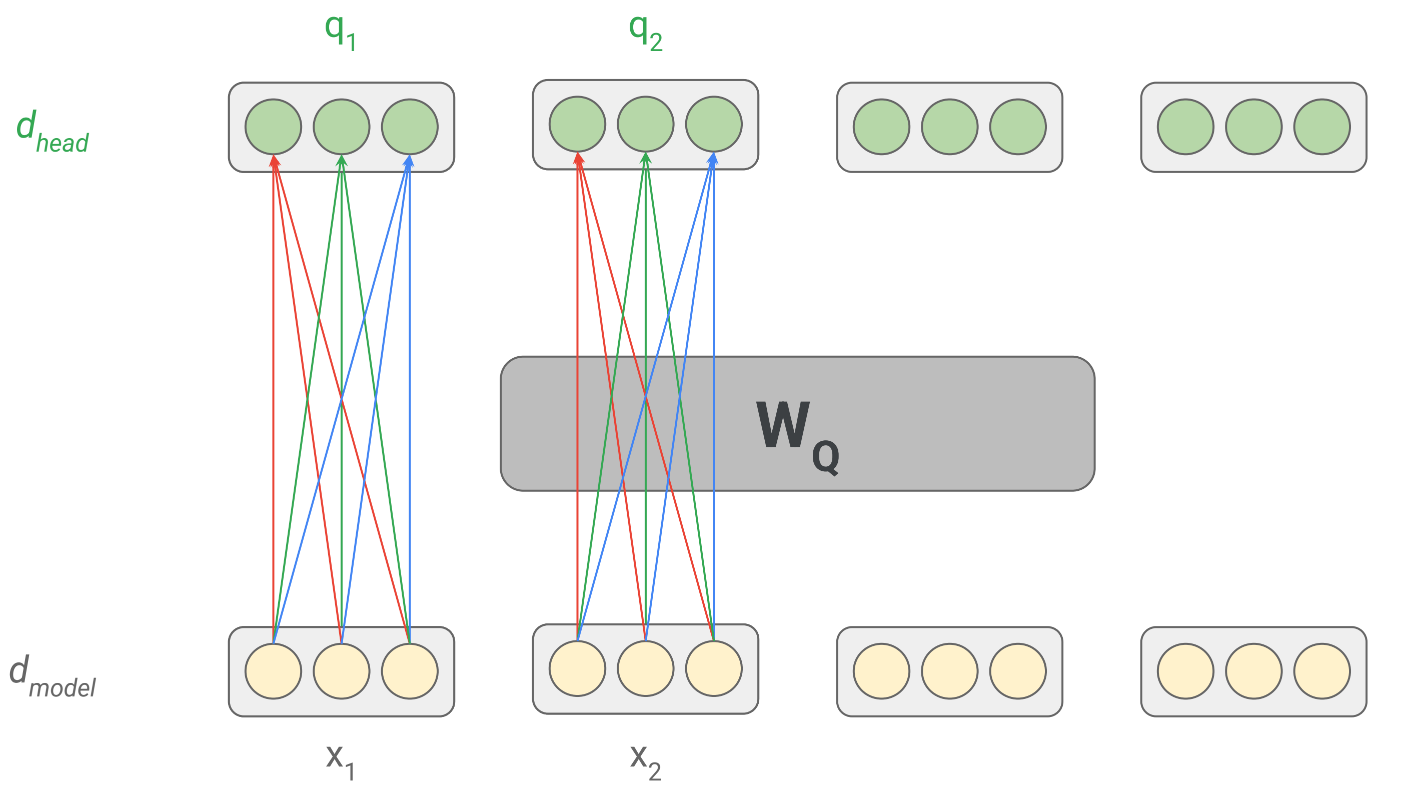
$Q = X \cdot W_Q$

$K = X \cdot W_K$

$V = X \cdot W_V$

linear projections

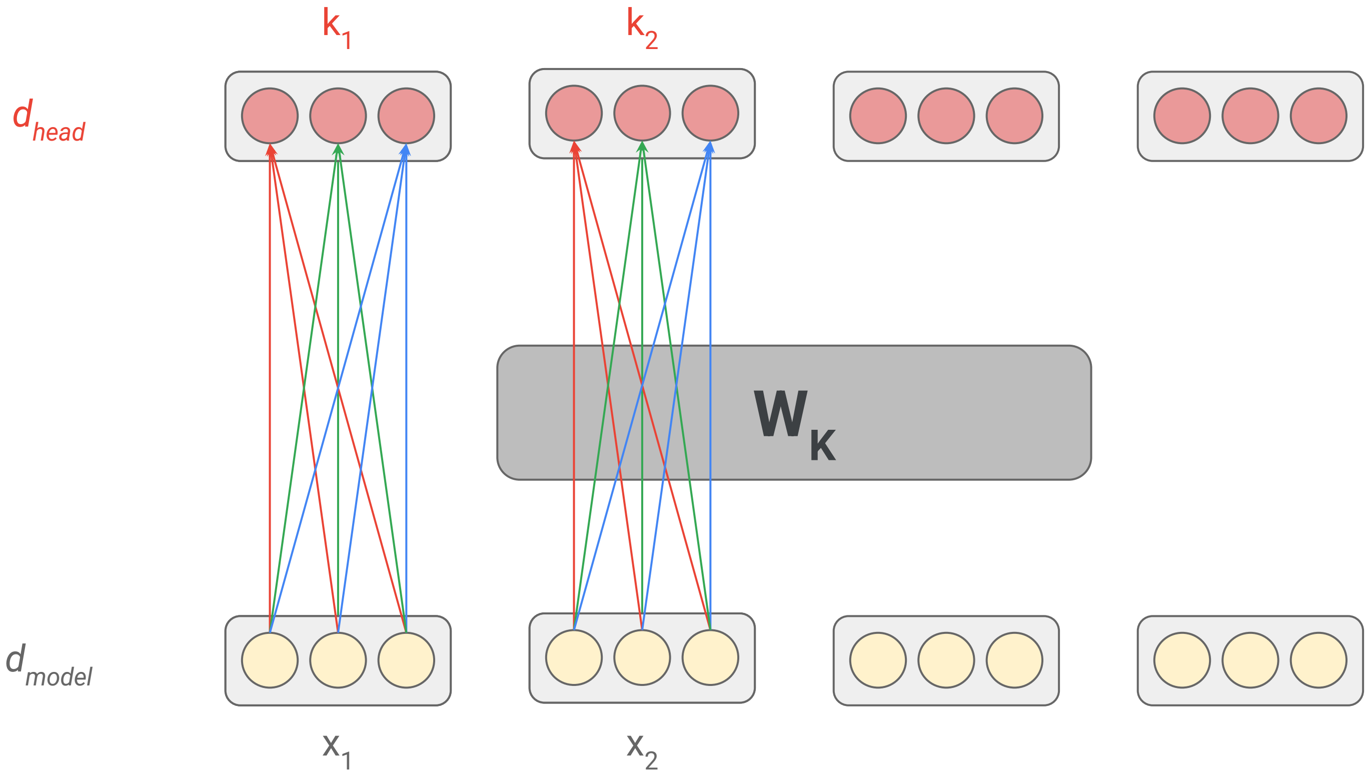
Query vectors



$$Q = X \cdot W_Q$$

linear projections

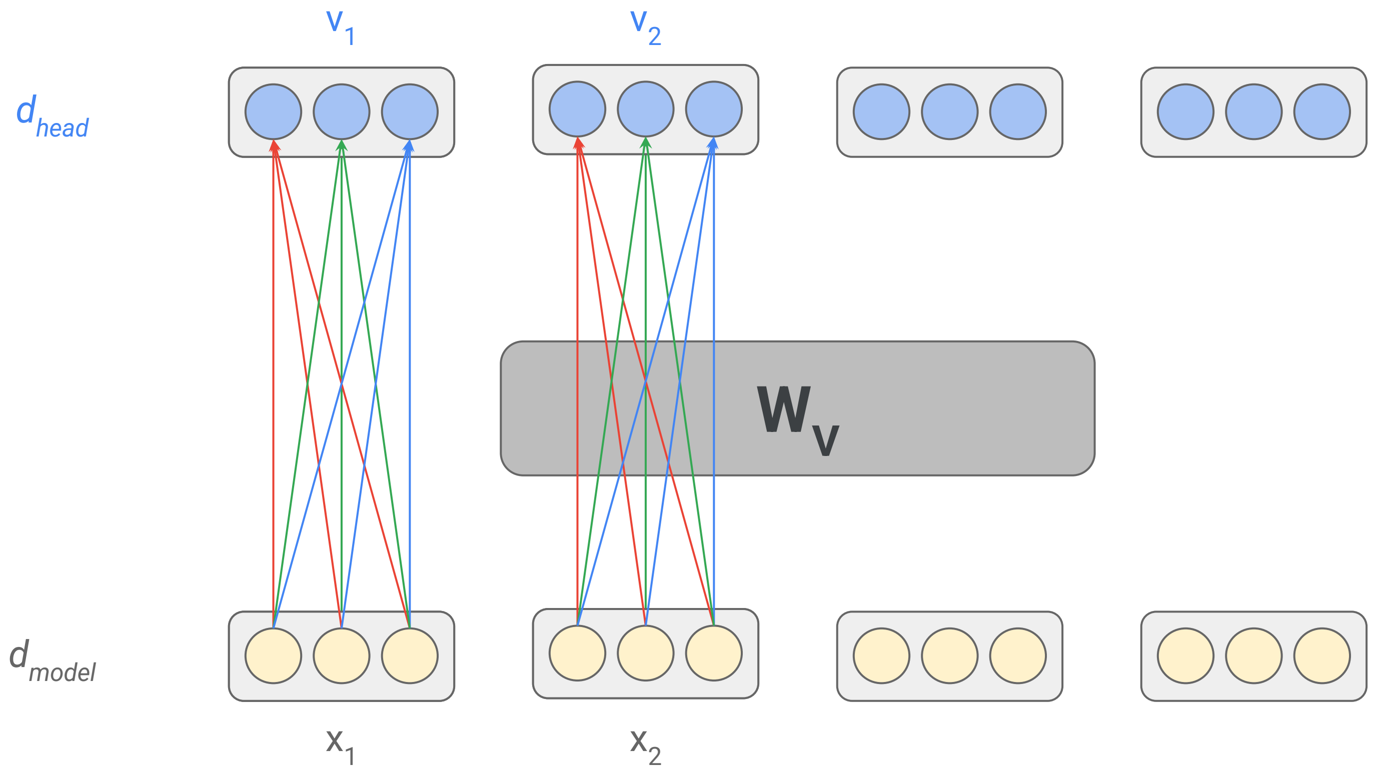
Key vectors



$K = X \cdot W_K$

linear projections

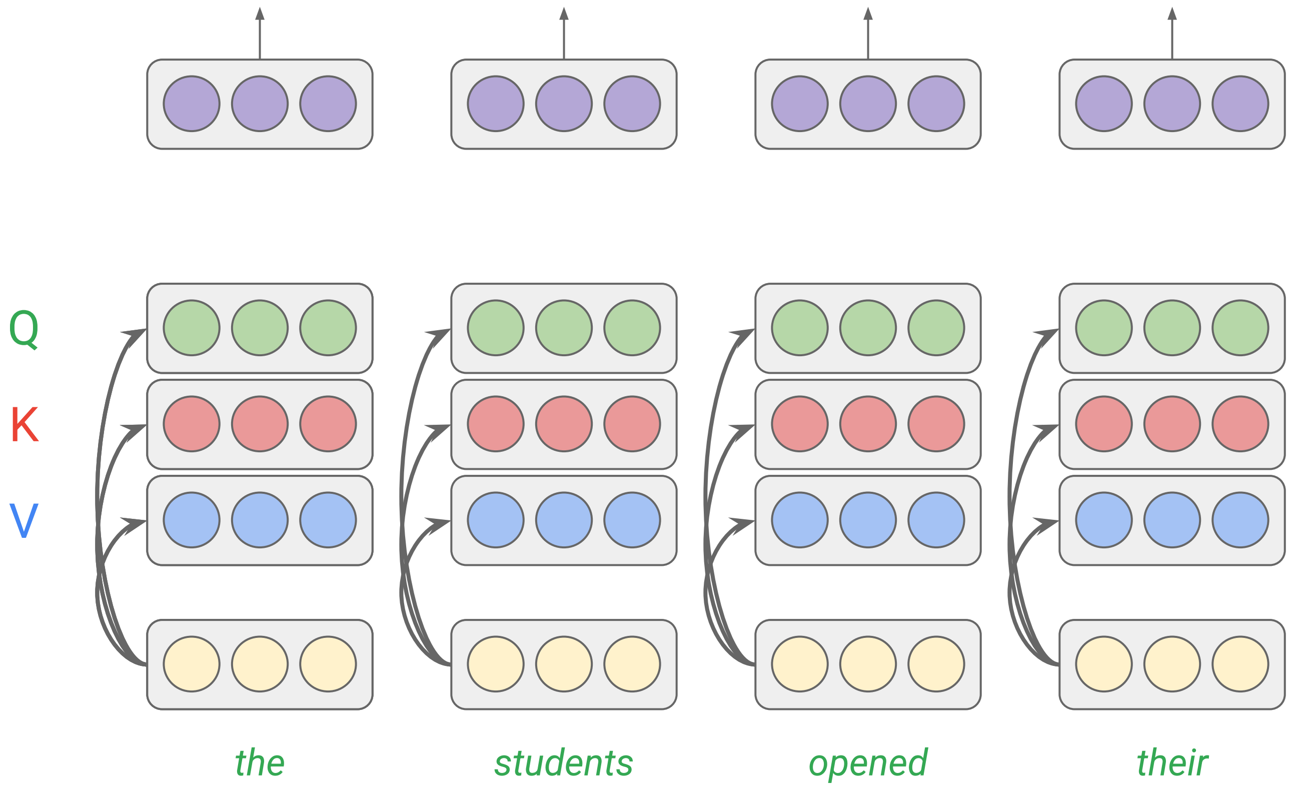
Value vectors



$$V = X \cdot W_V$$

linear projections

Attention (cont'd)



$$Q = X \cdot W_Q$$

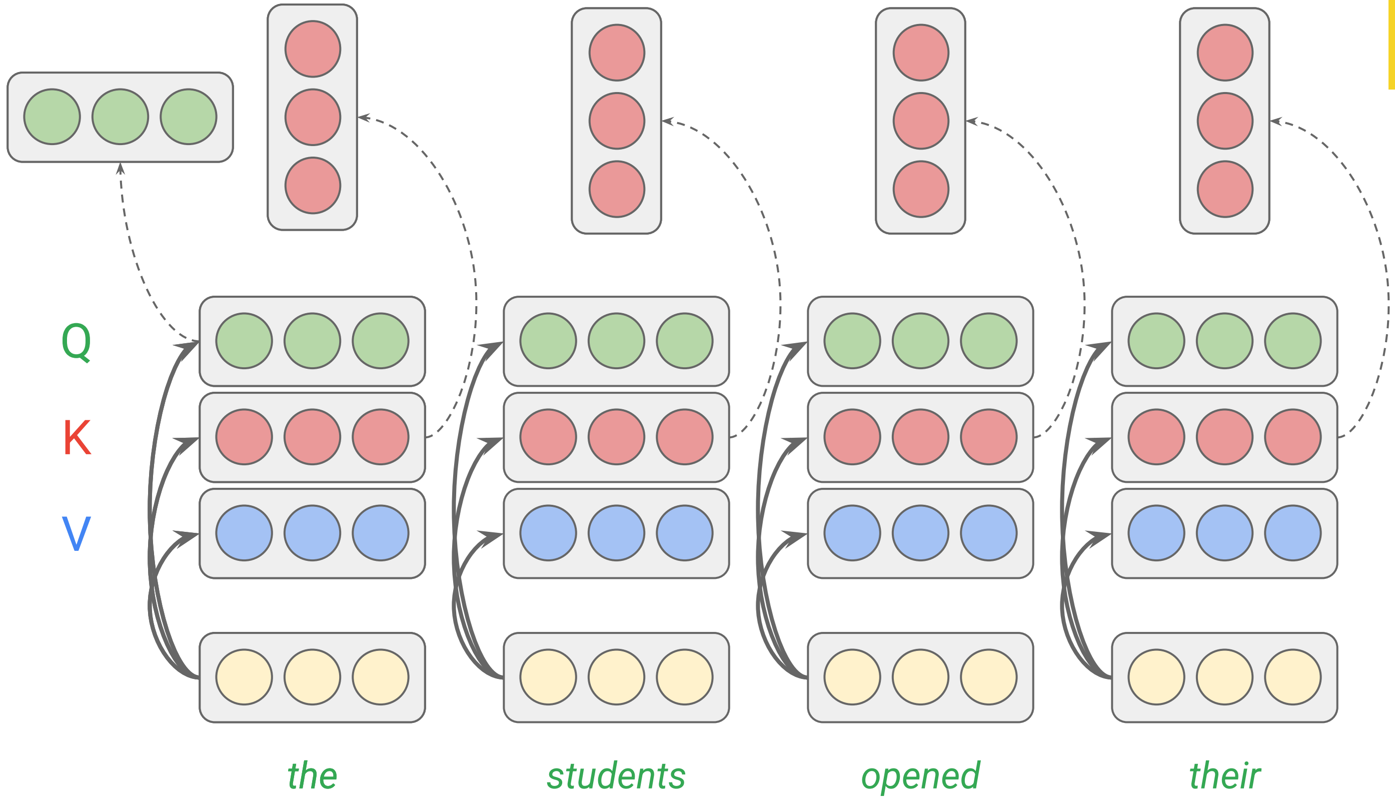
$$K = X \cdot W_K$$

$$V = X \cdot W_V$$

linear projections

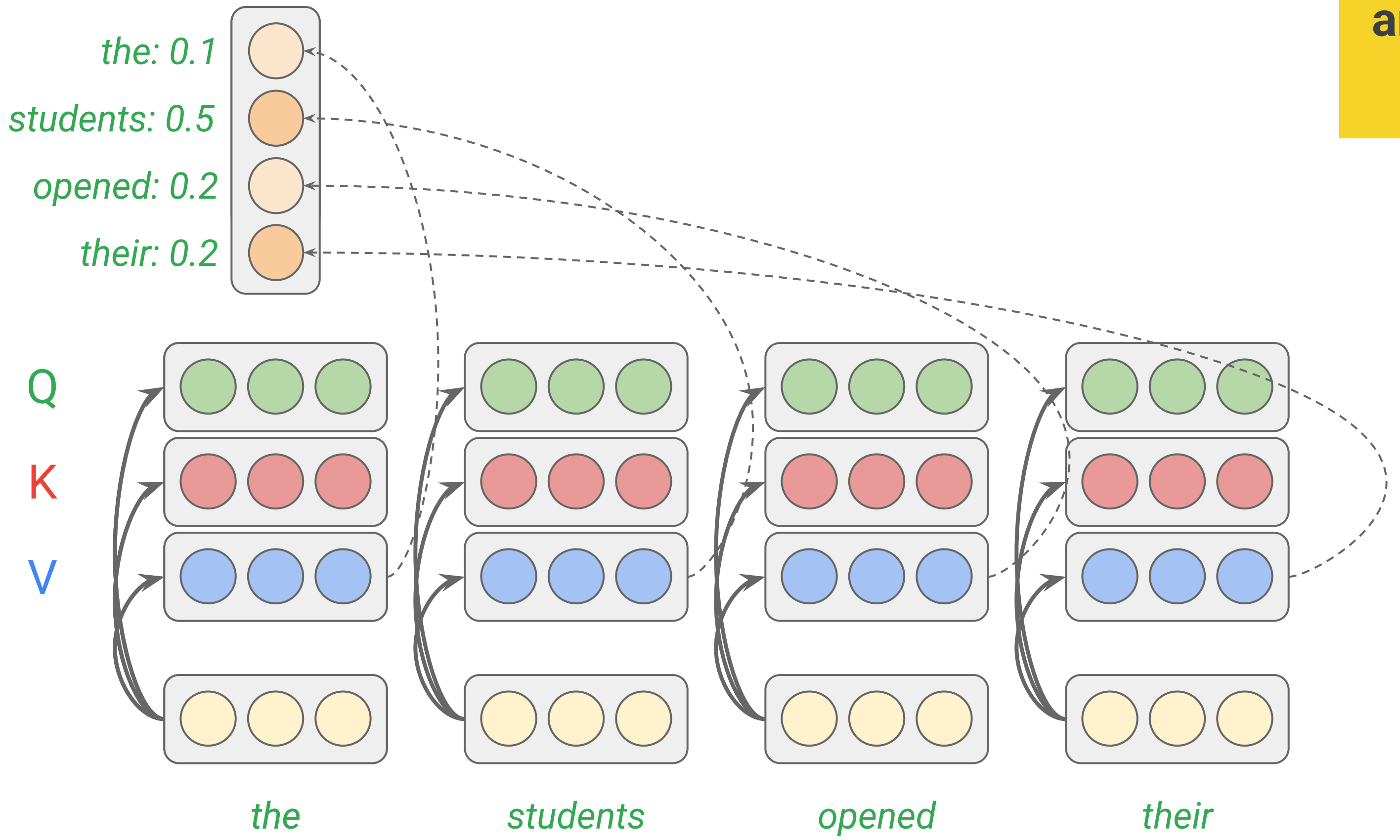
Self-attention (cont'd)

***all* computations are parallelized**



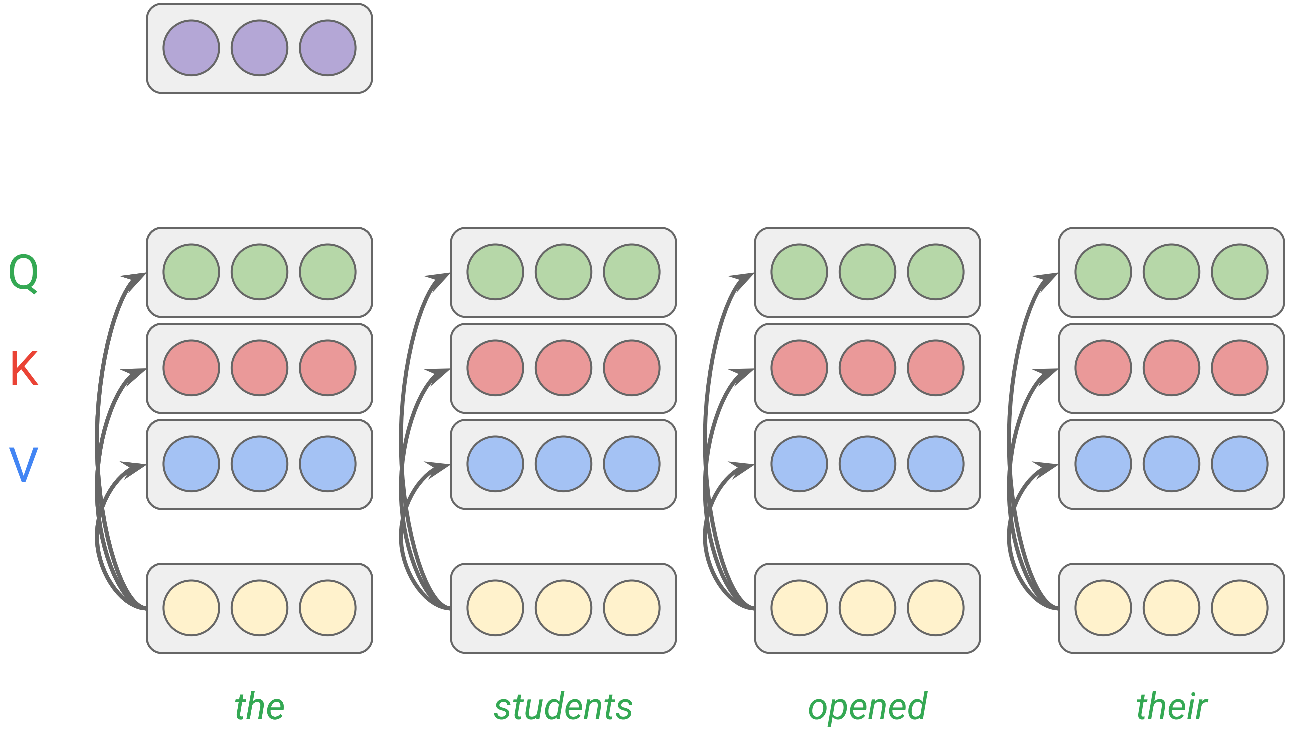
Self-attention (cont'd)

all **computations are parallelized**



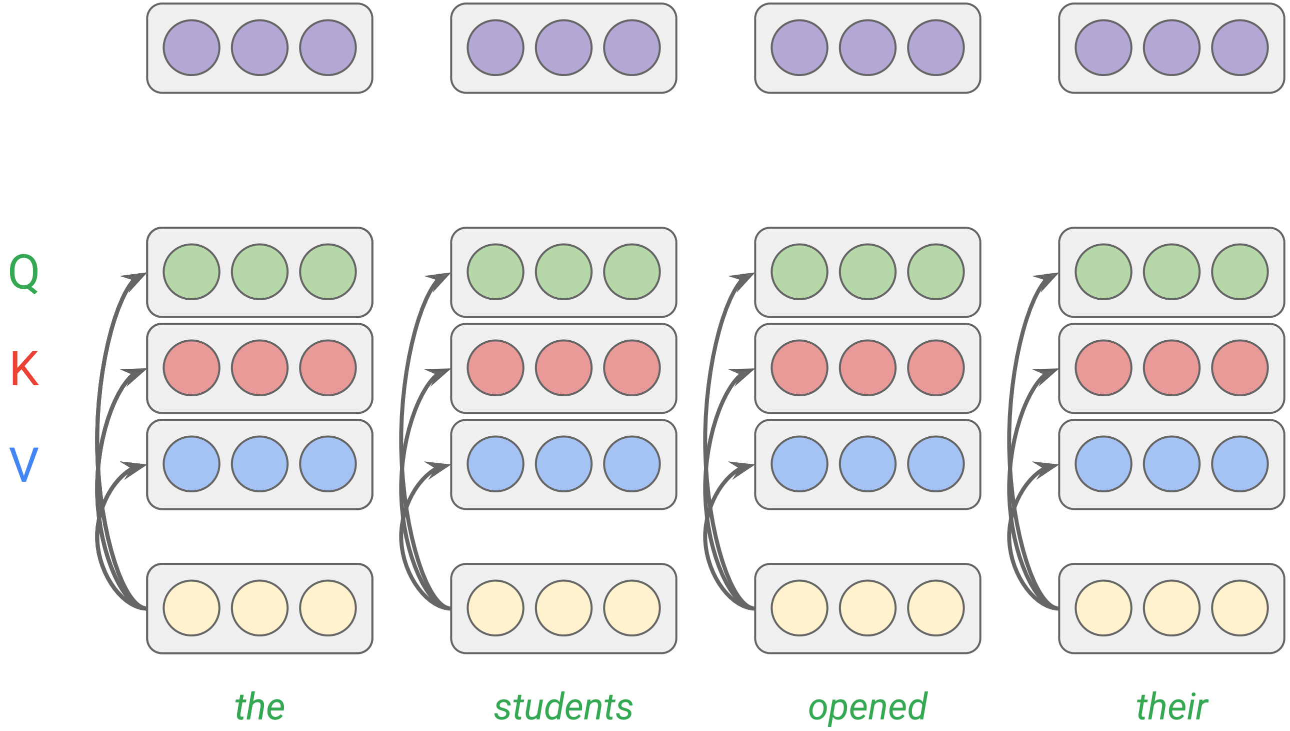
Self-attention (cont'd)

all computations are parallelized

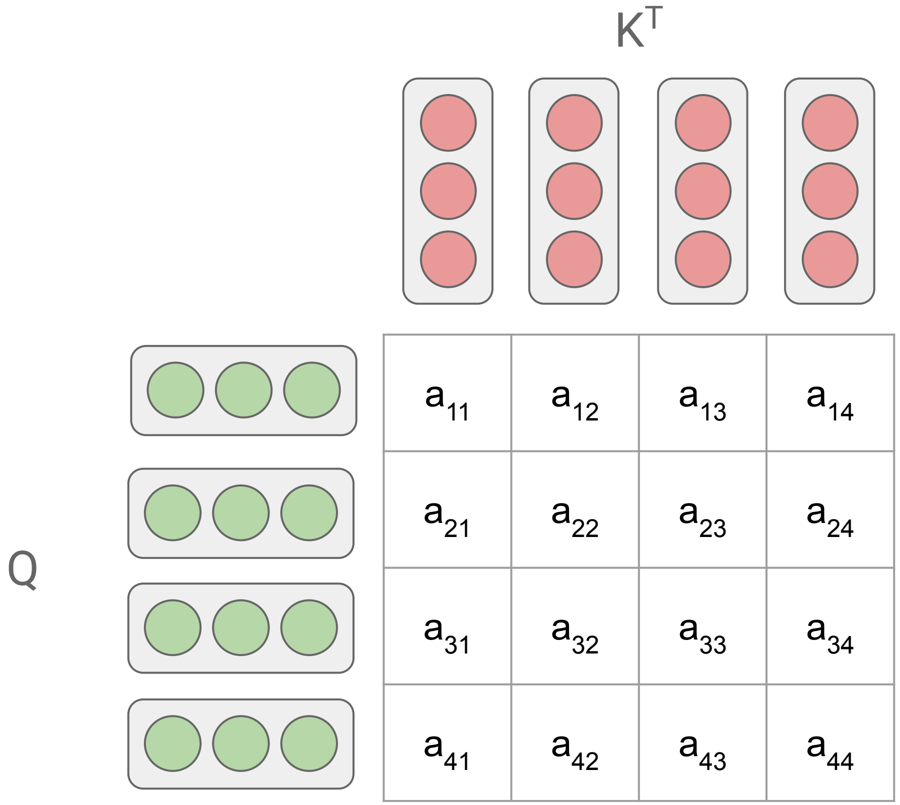


Self-attention (cont'd)

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

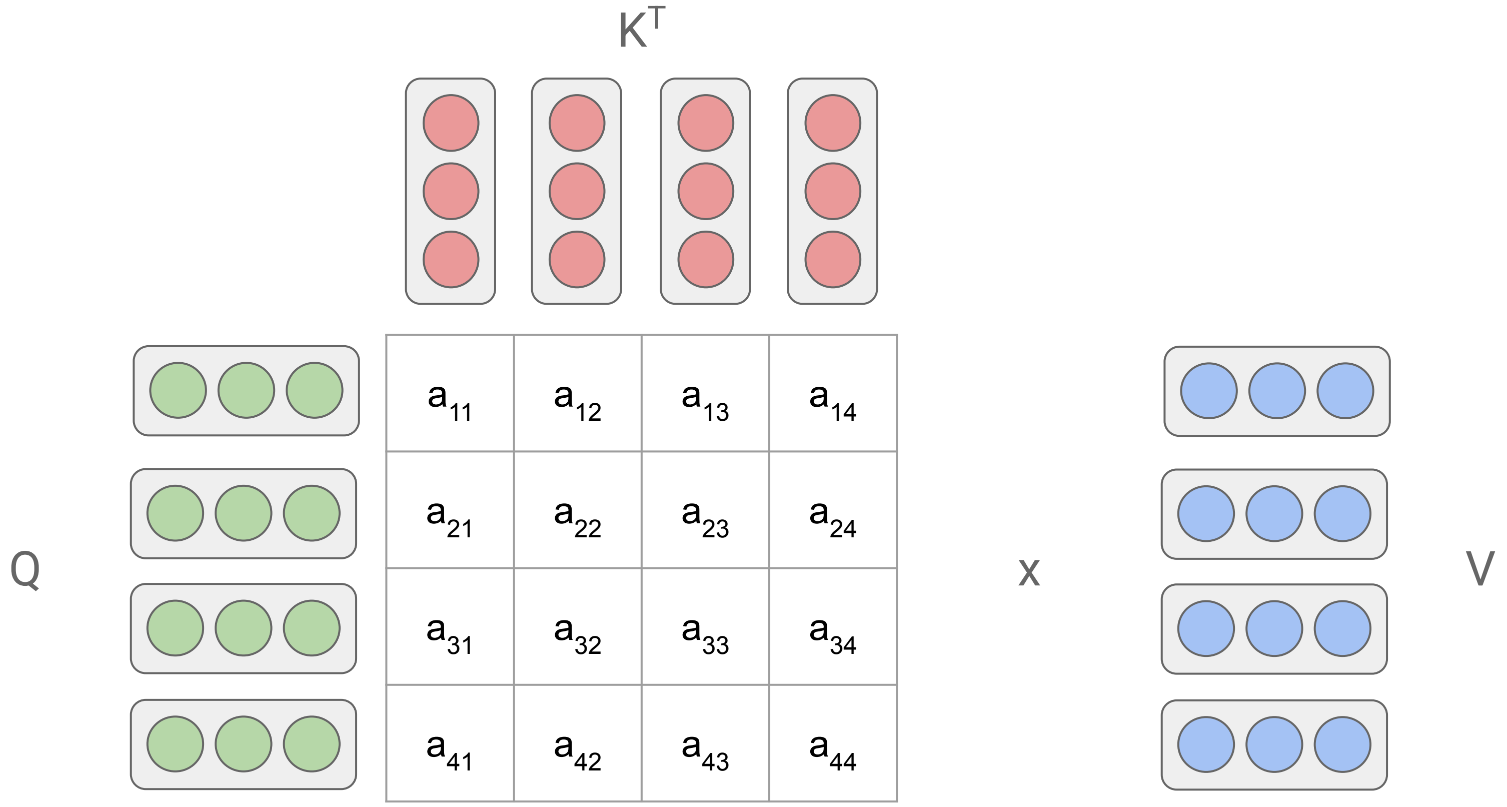


Quadratic complexity



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

Quadratic complexity



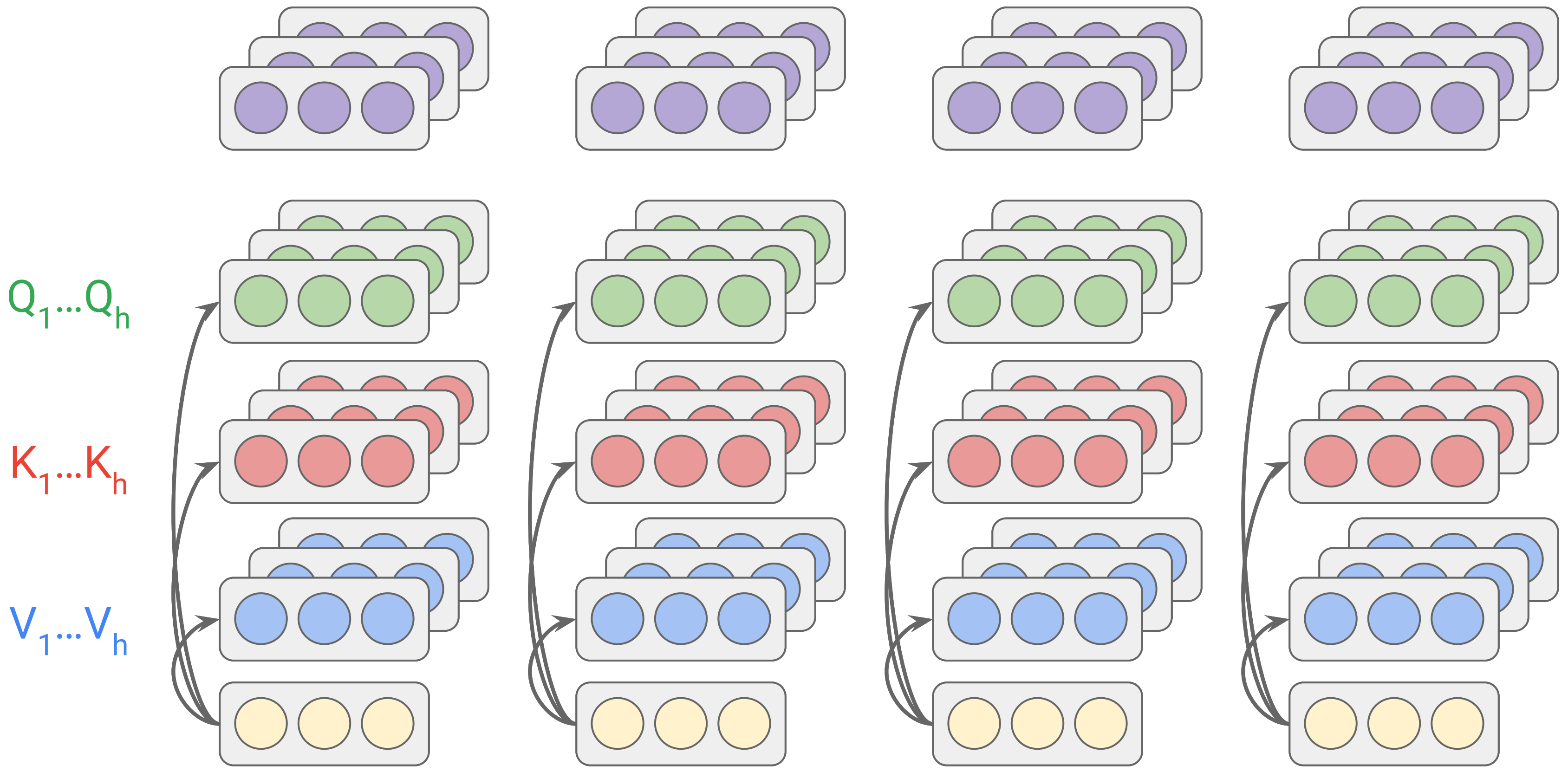
All computations are parallelized

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

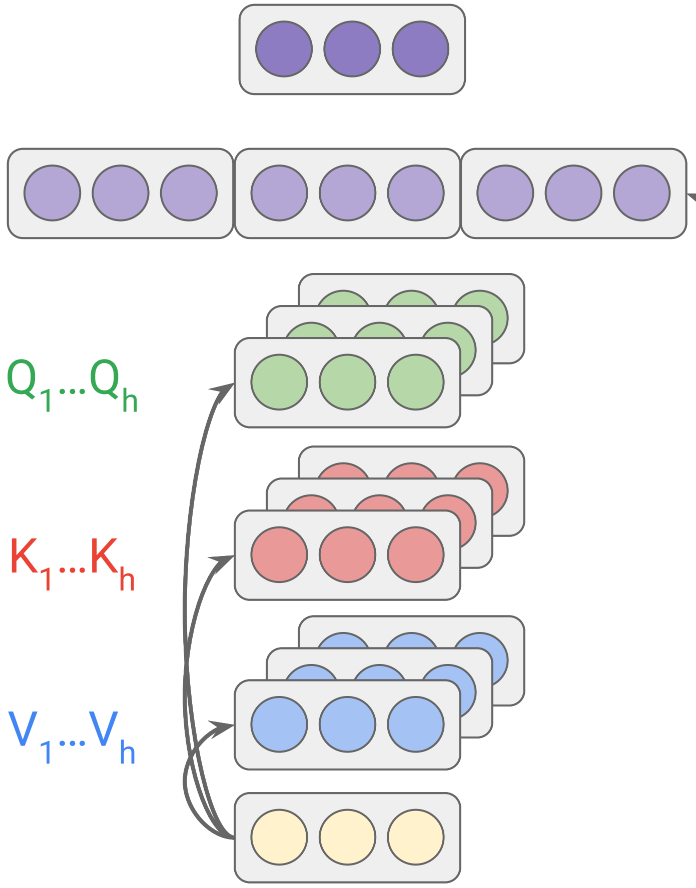
d_k: scaling factor

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

Multi-head attention

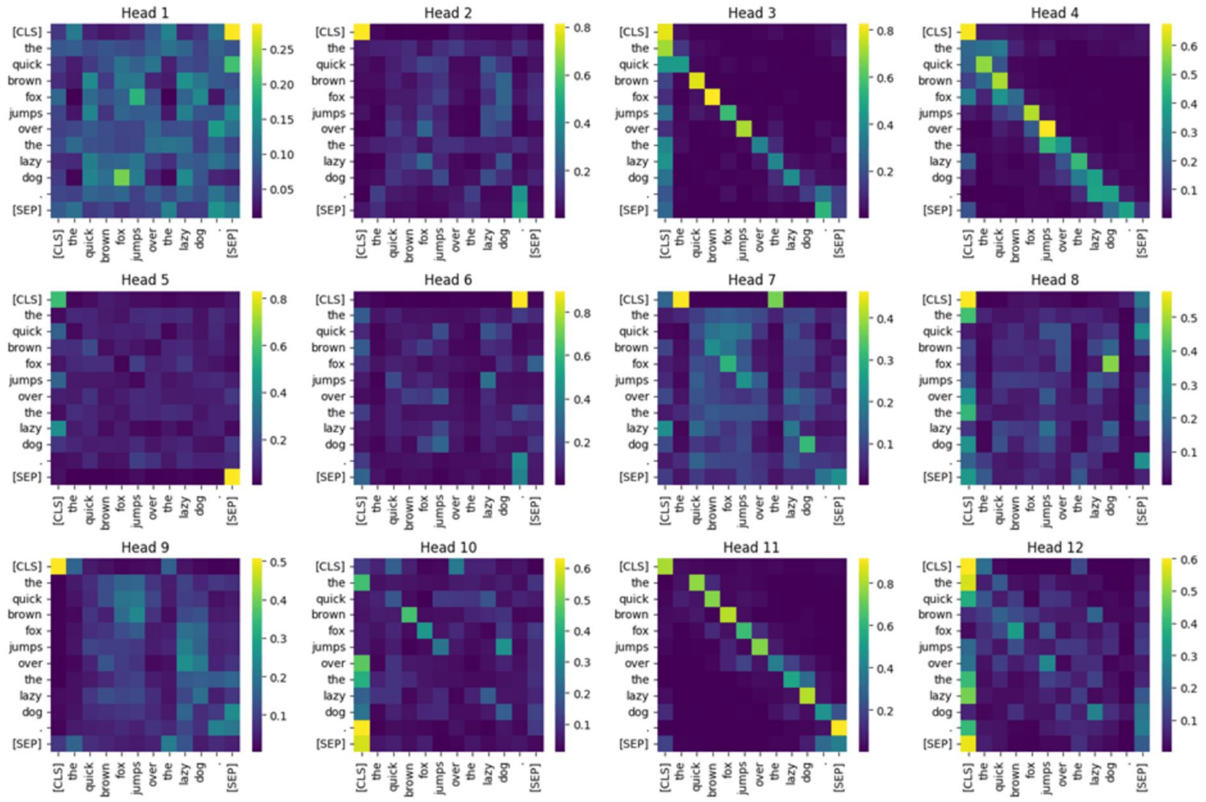


Multi-head attention (cont'd)

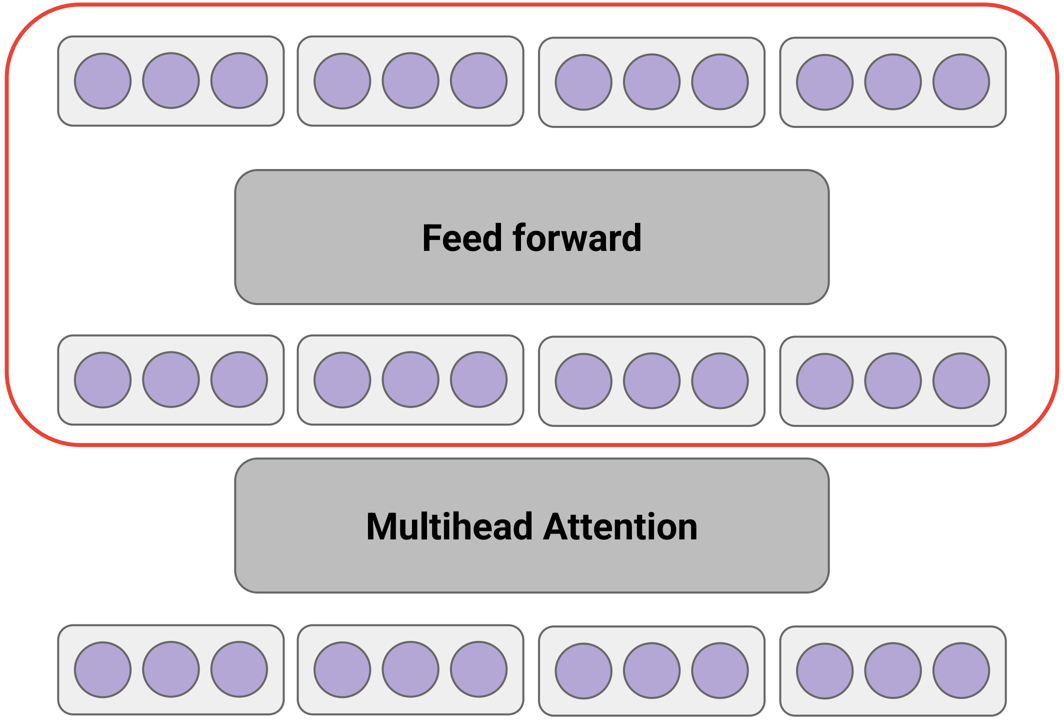


These output values are concatenated and once again projected

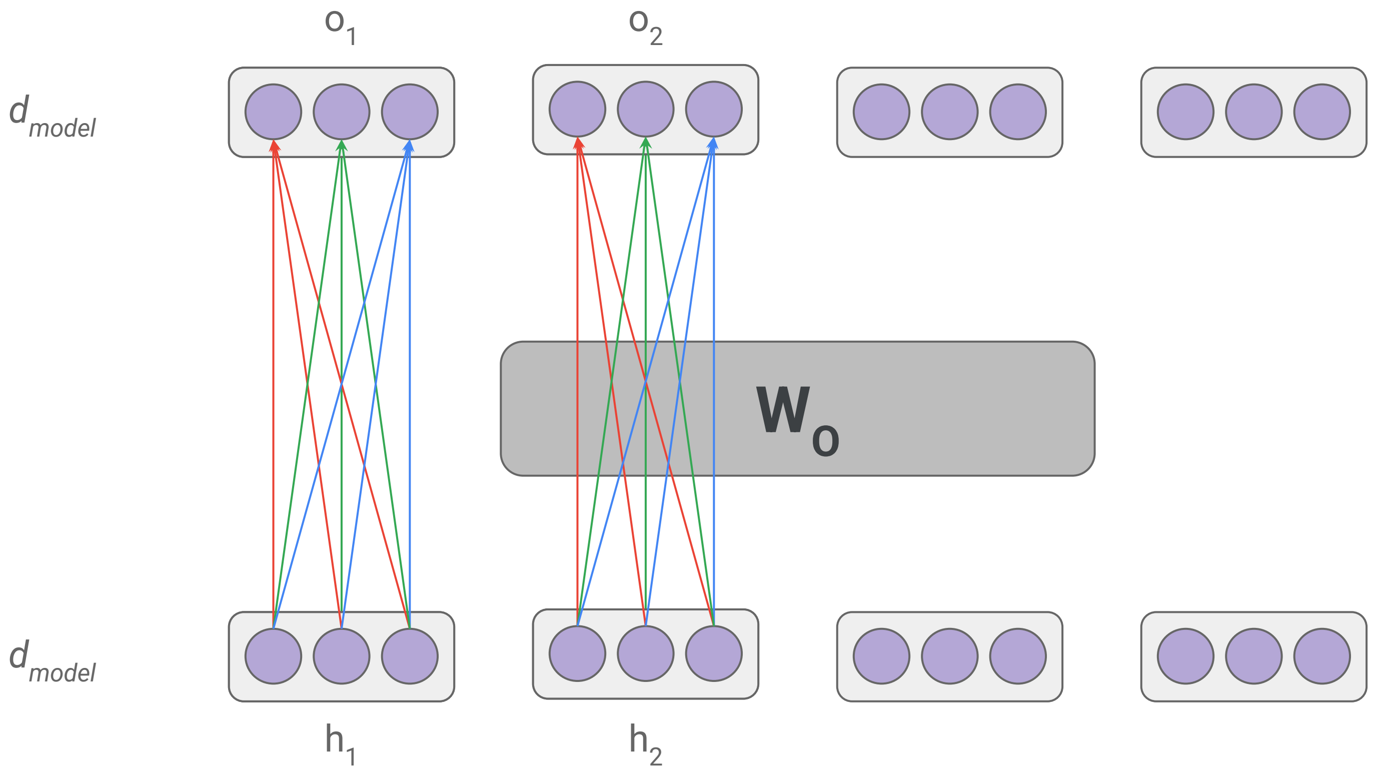
Attention matrices



Transformer block (cont'd)



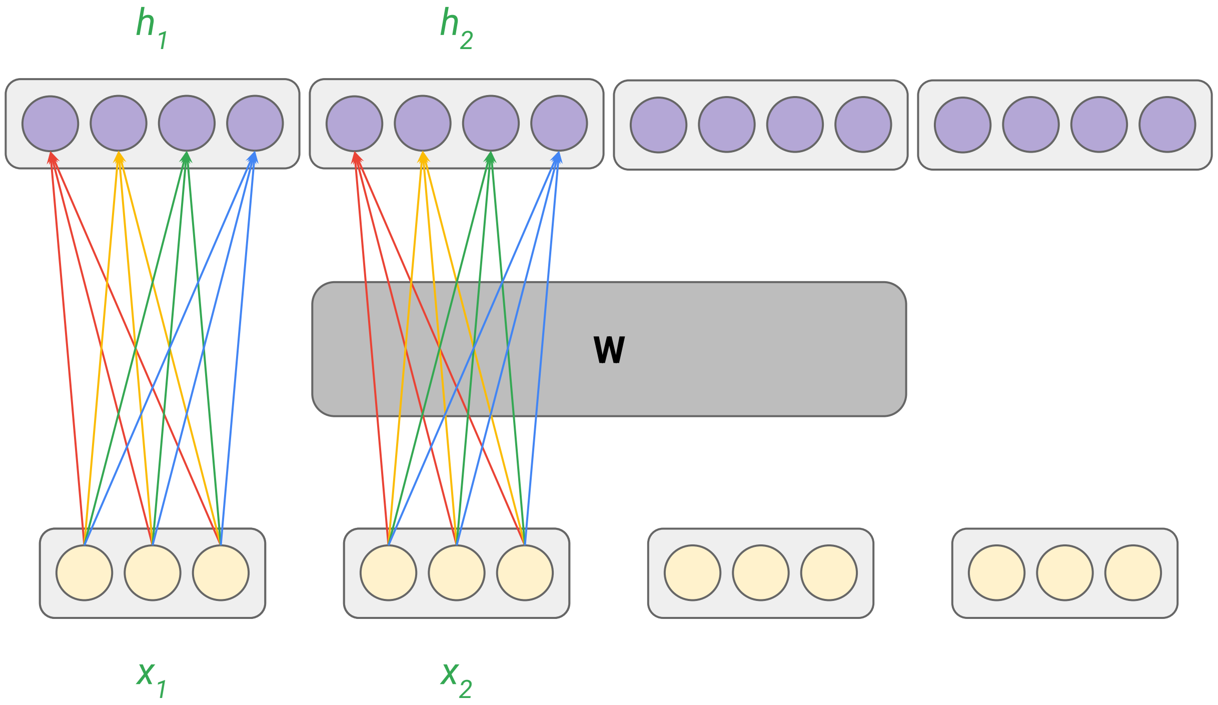
output vectors



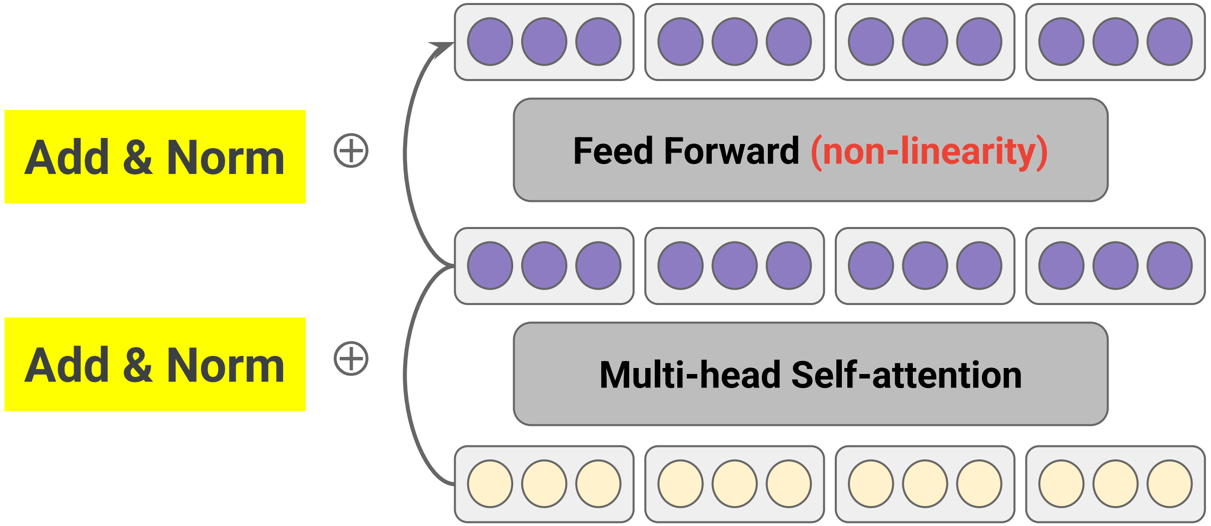
$$O = H \cdot W_O$$

linear projections

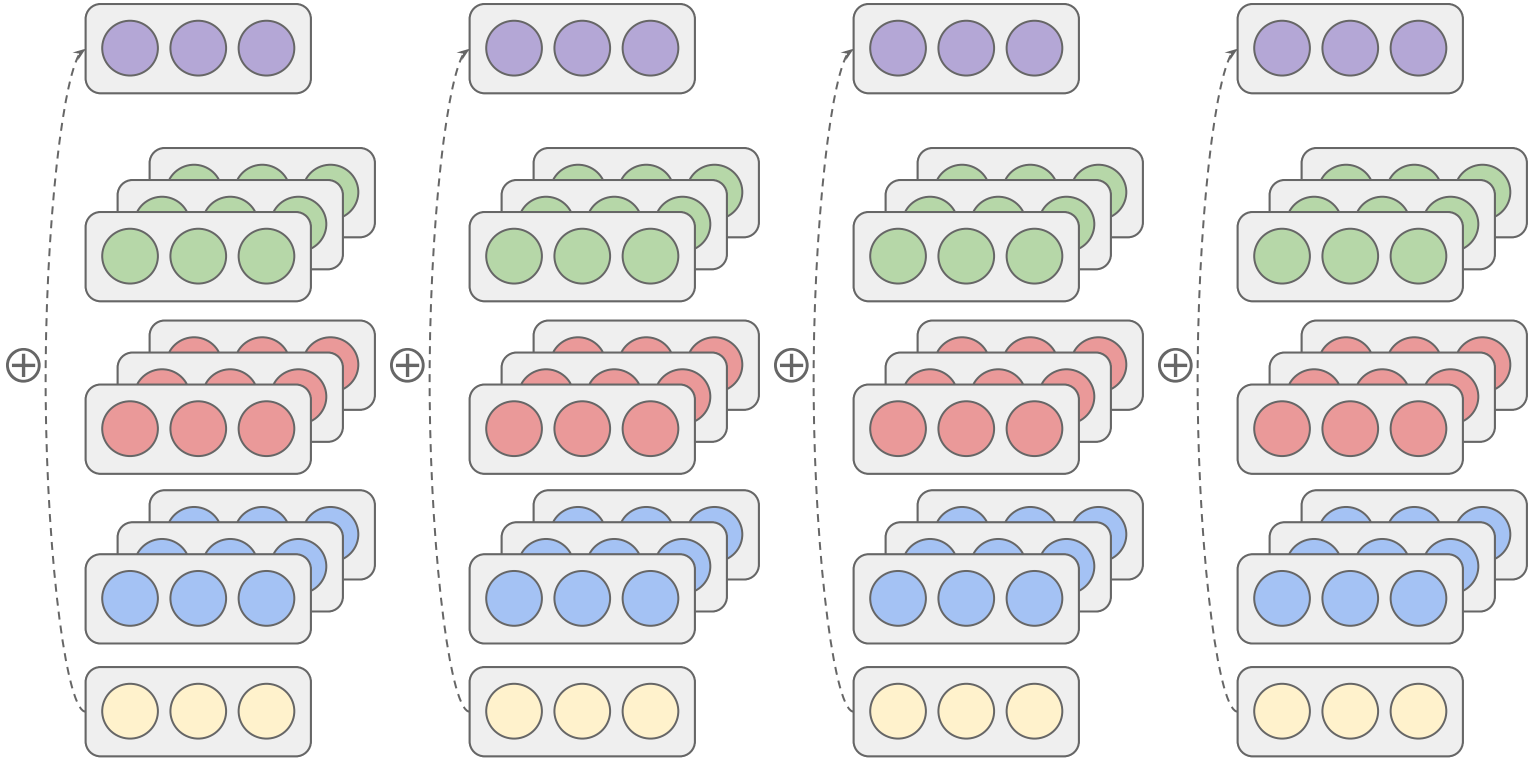
Position-wise feedforward networks



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

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

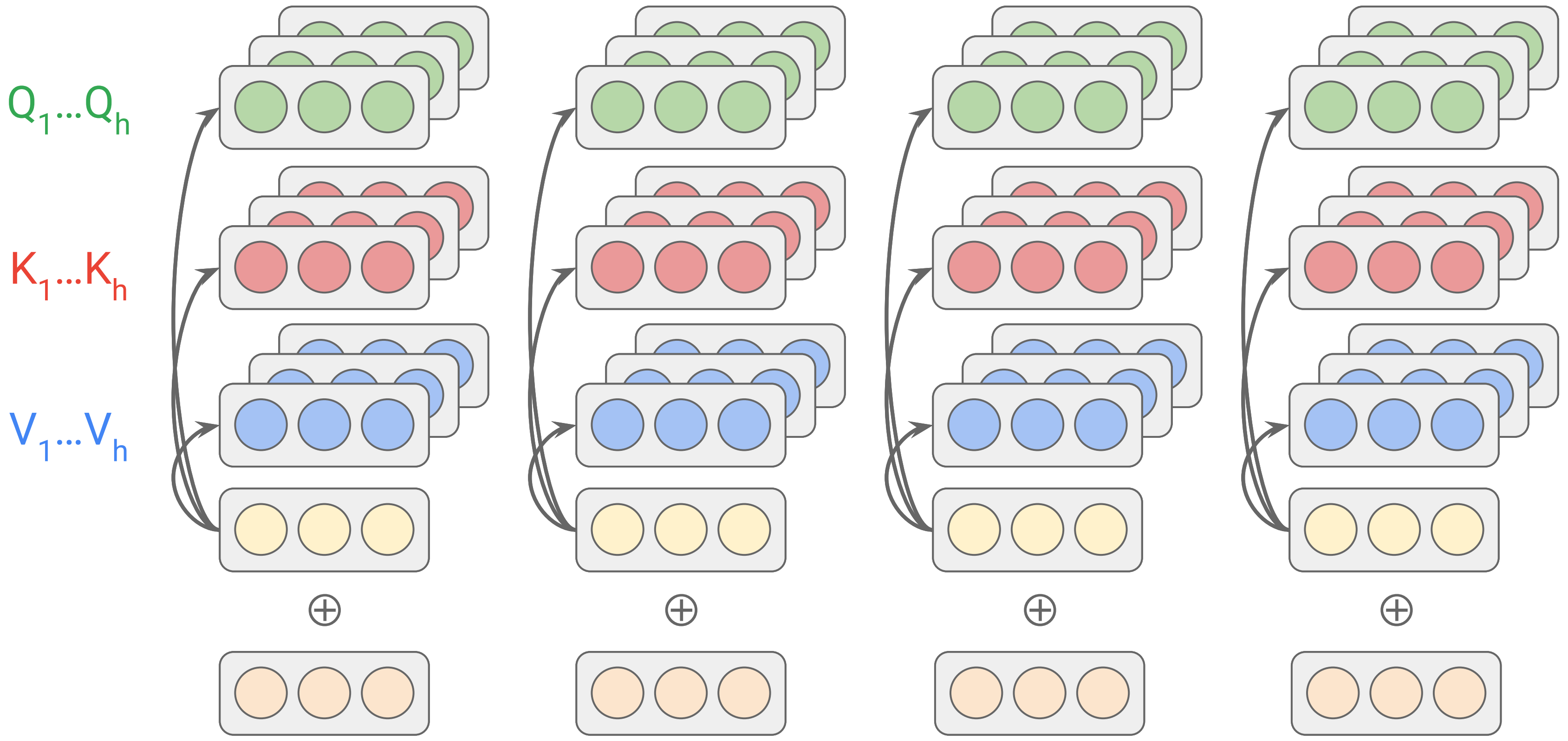
ReLU (Rectified Linear Unit)

Sinusoidal positional encoding

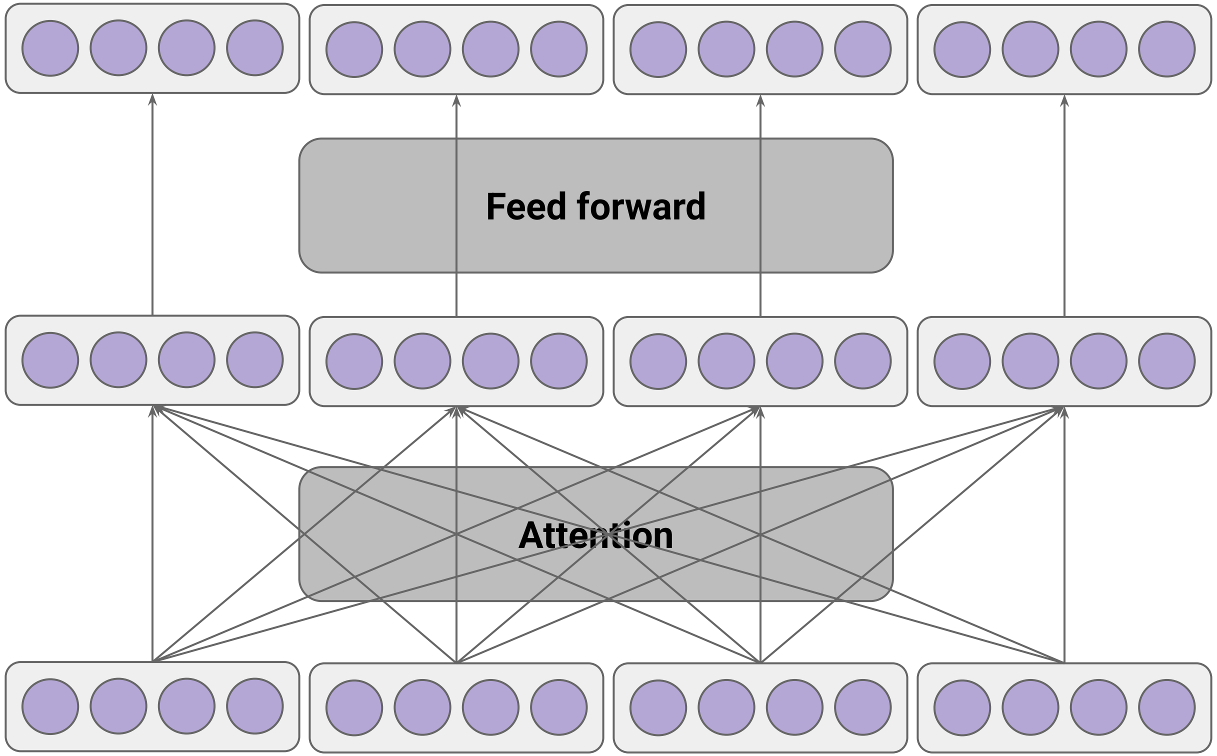
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

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

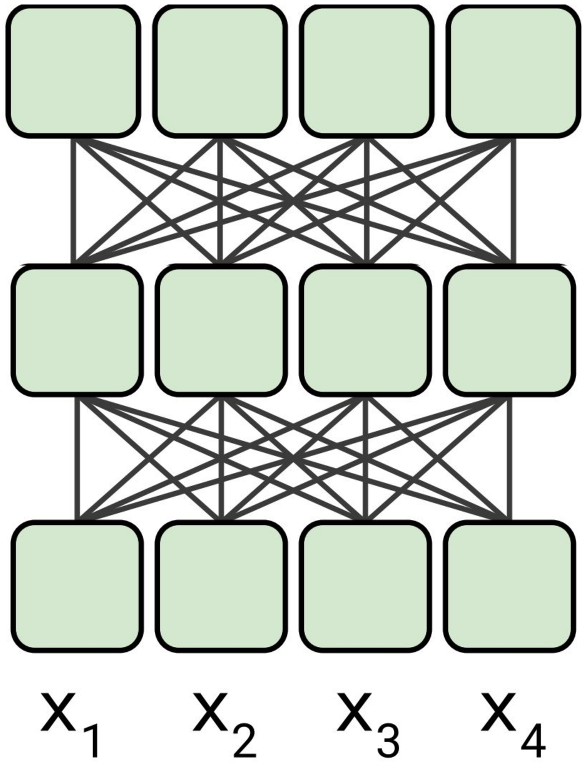
Positional Encoding (cont'd)



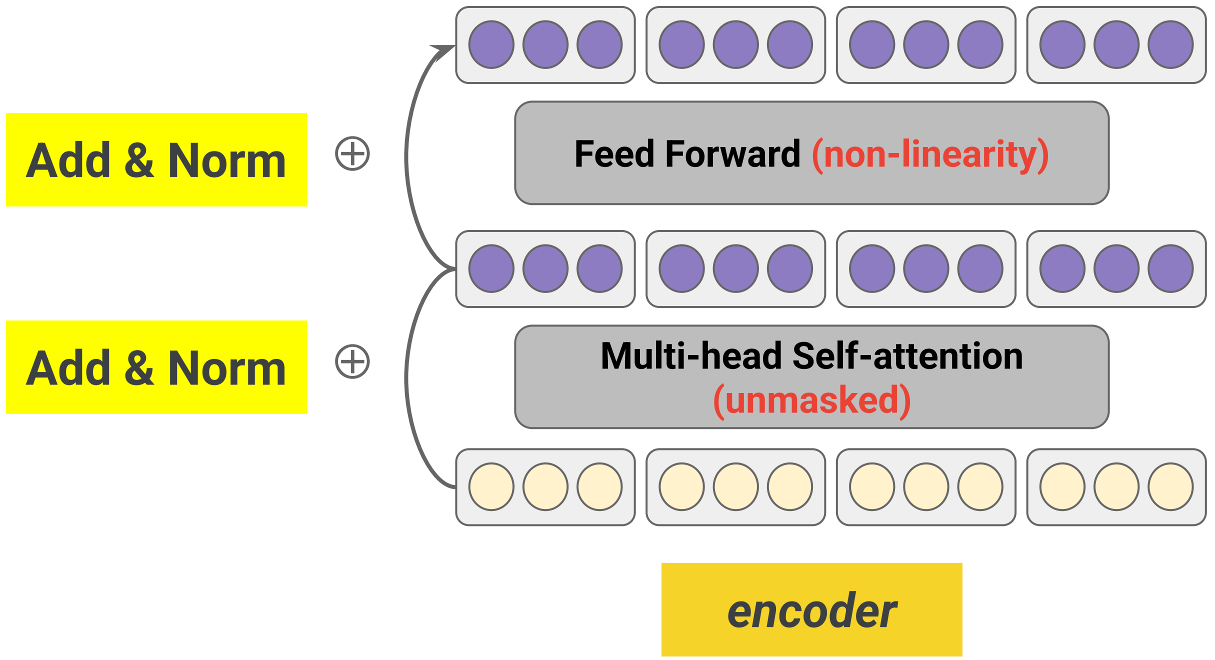
Putting it all together



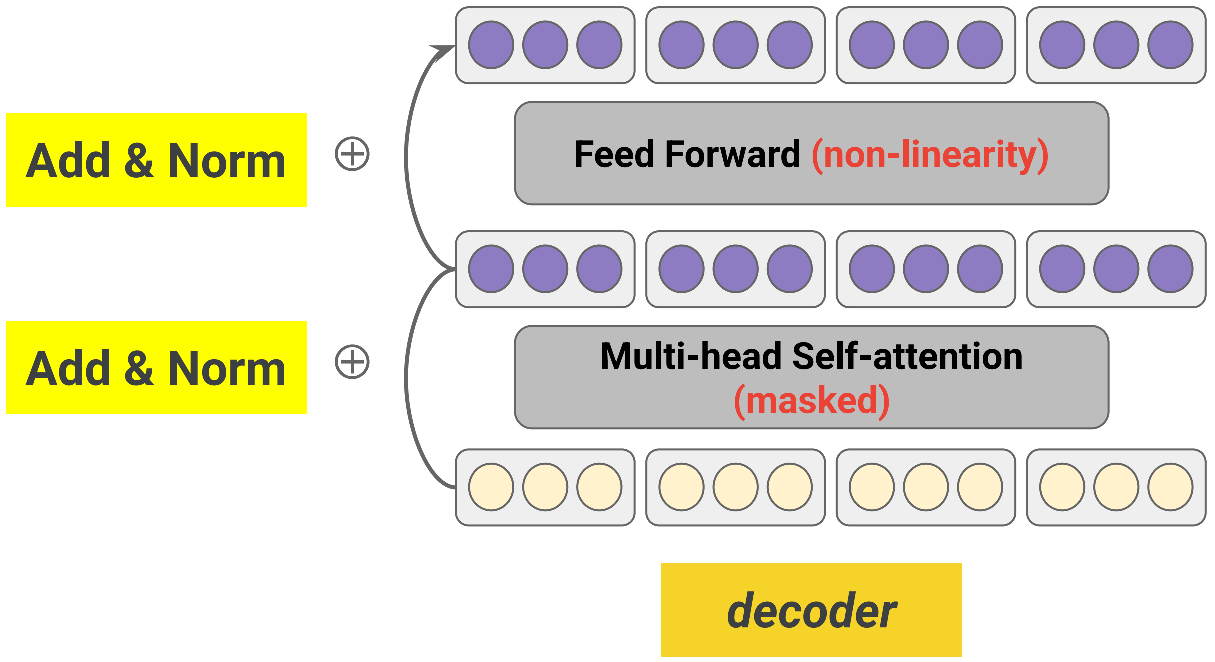
Encoder only



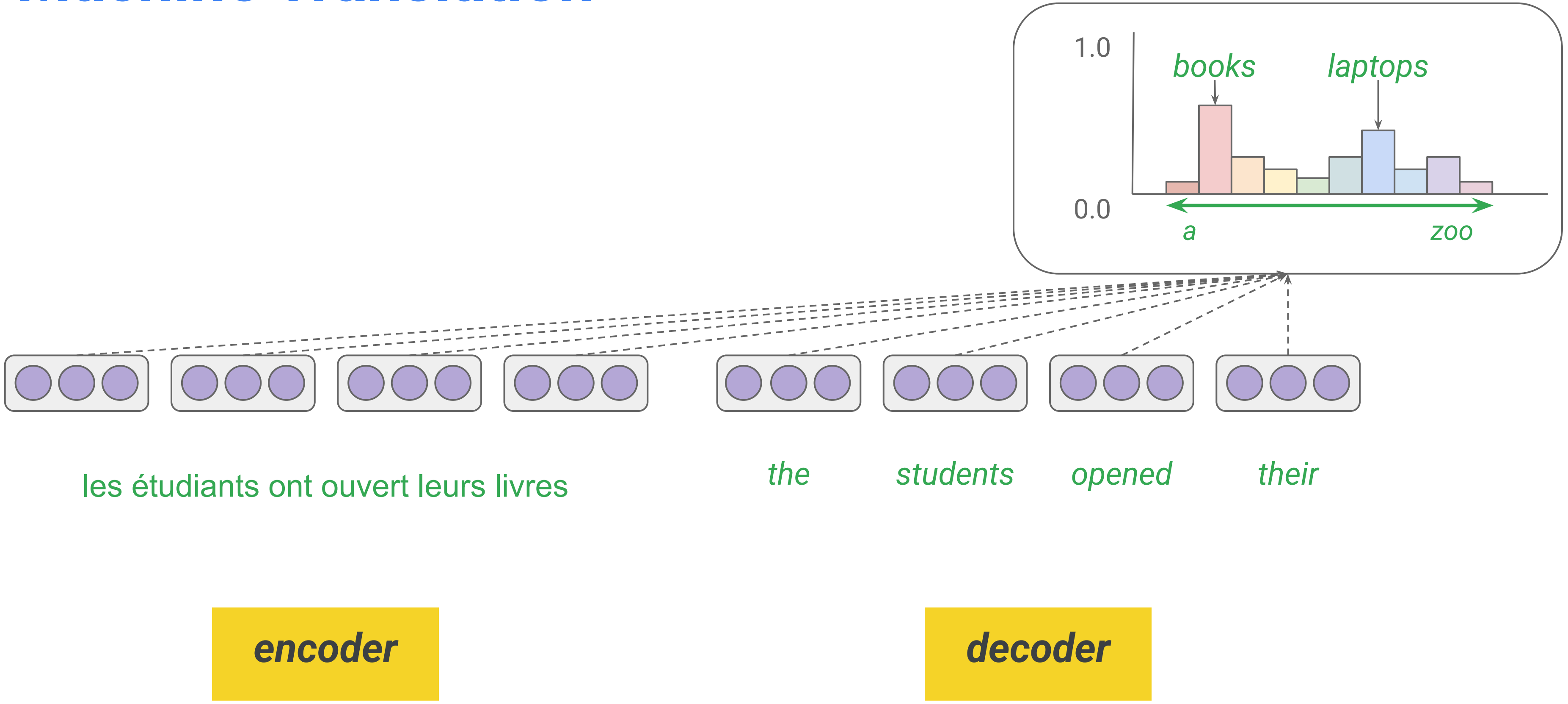
Encoder (one layer)


Add & Norm ⊕
Add & Norm ⊕
Feed Forward (non-linearity)
Multi-head Self-attention (unmasked)
encoder

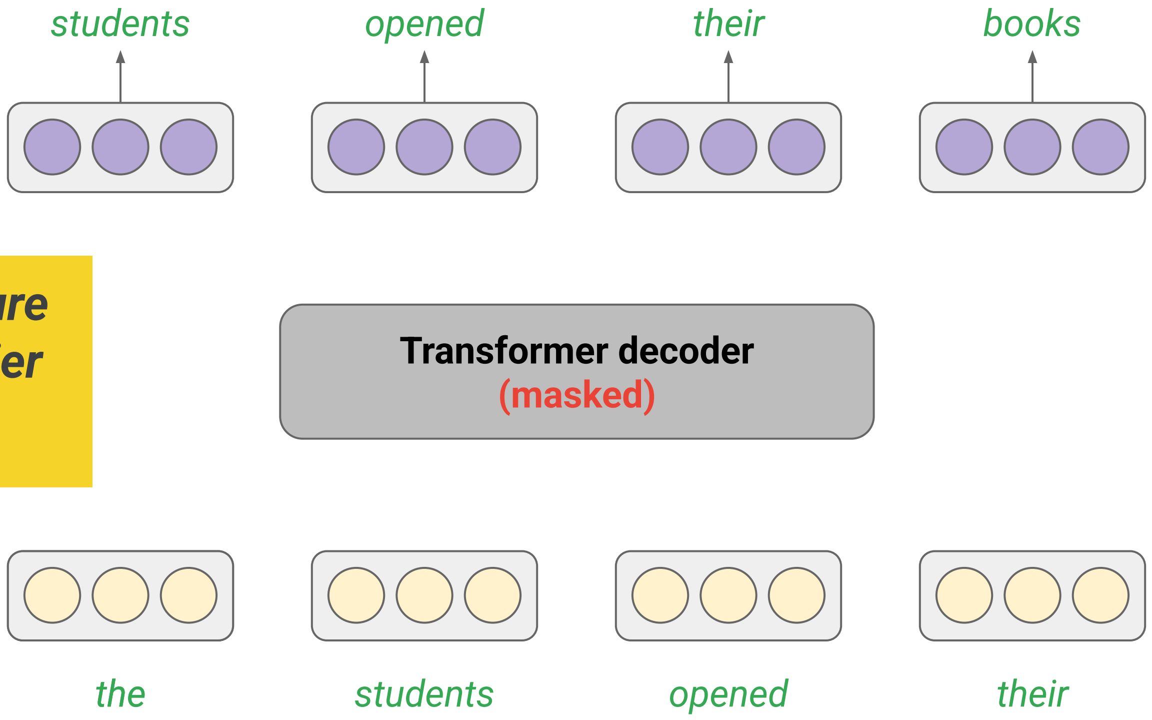
Decoder (one layer)


Add & Norm
Add & Norm
Feed Forward (non-linearity)
Multi-head Self-attention (masked)
decoder

Machine Translation

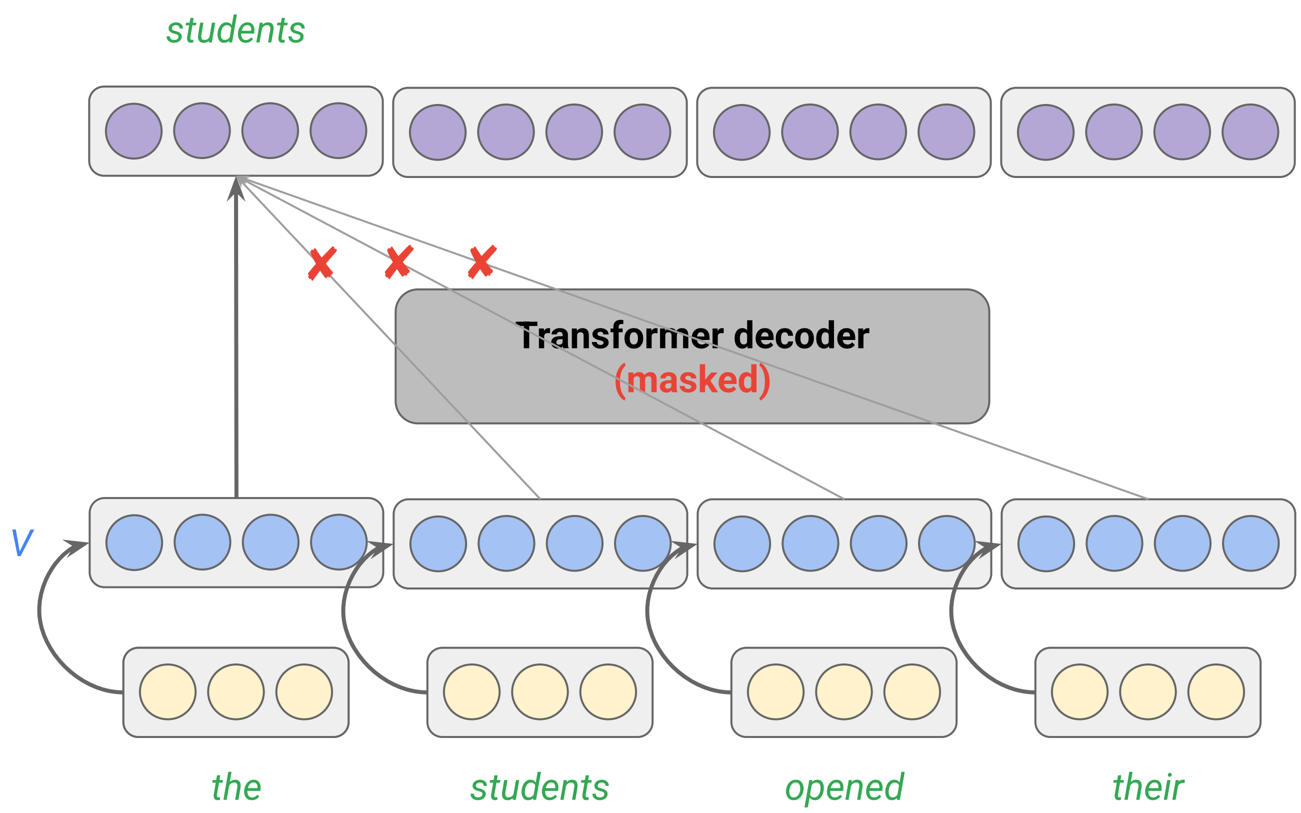


Transformer decoder

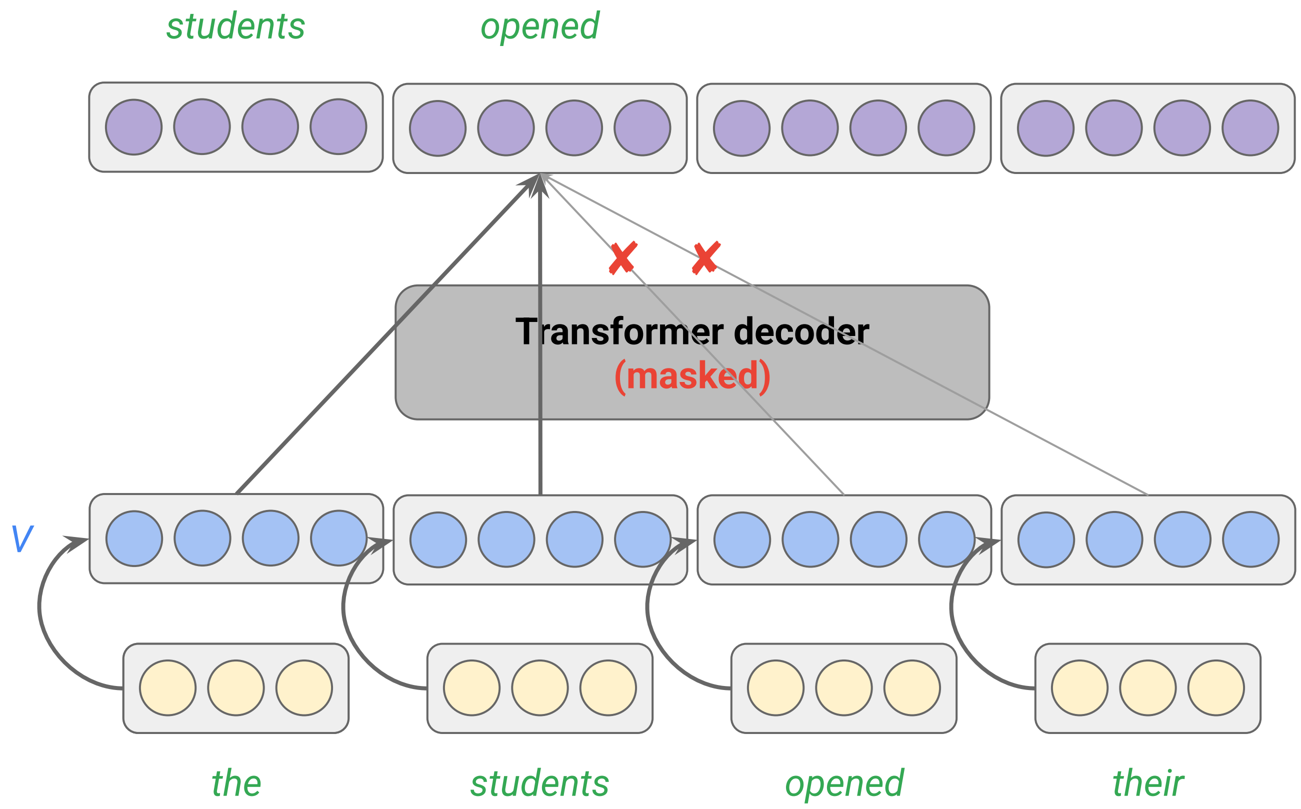


the architecture used in frontier LLMs

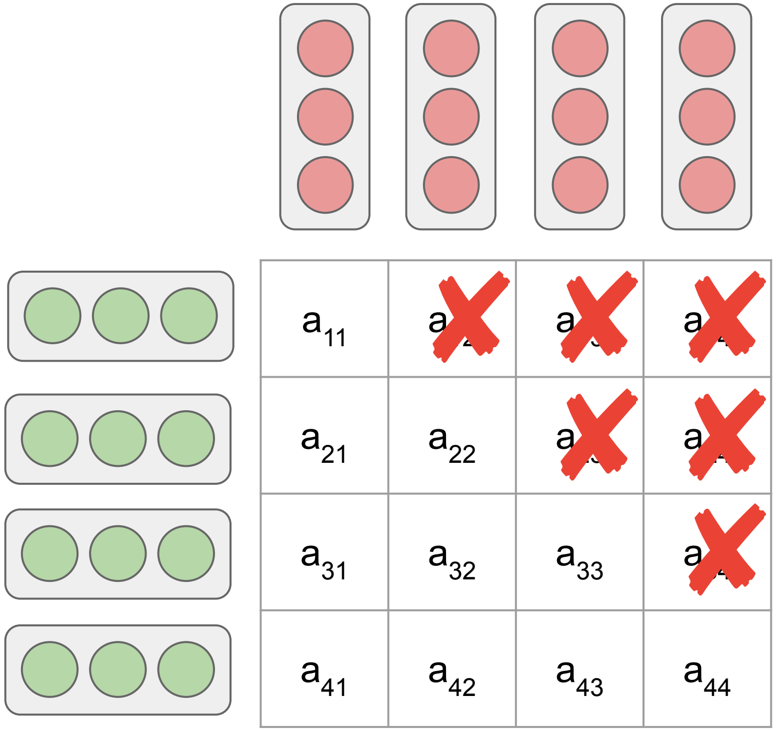
Transformer decoder (cont'd)



Transformer decoder (cont'd)

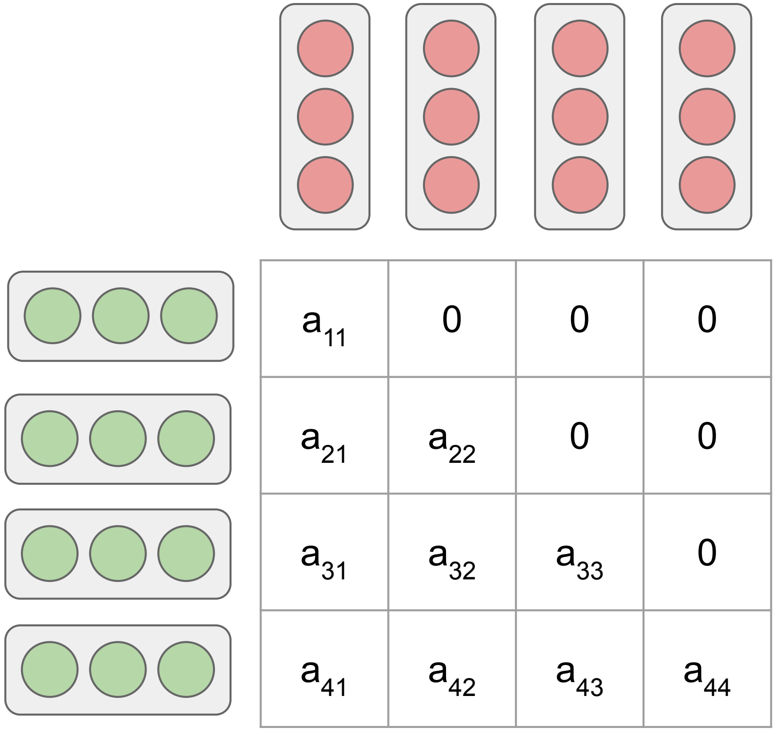


Self-attention in the decoder



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

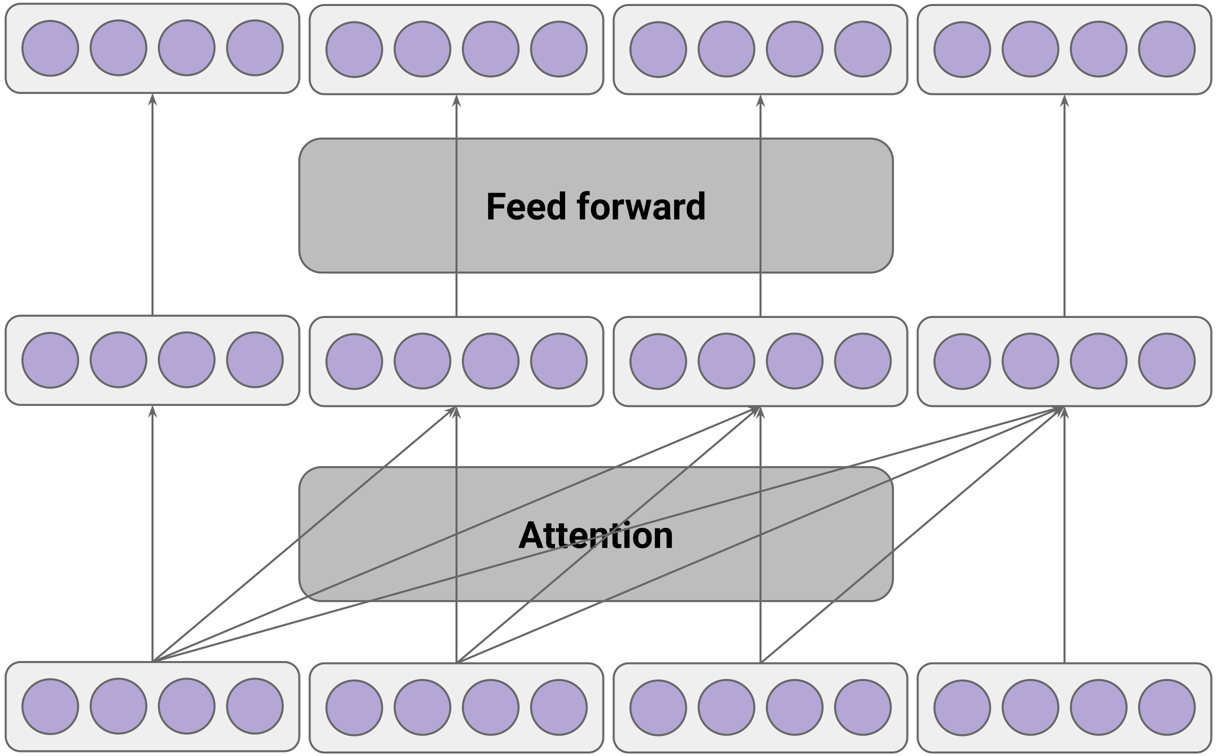
Self-attention in the decoder (cont'd)



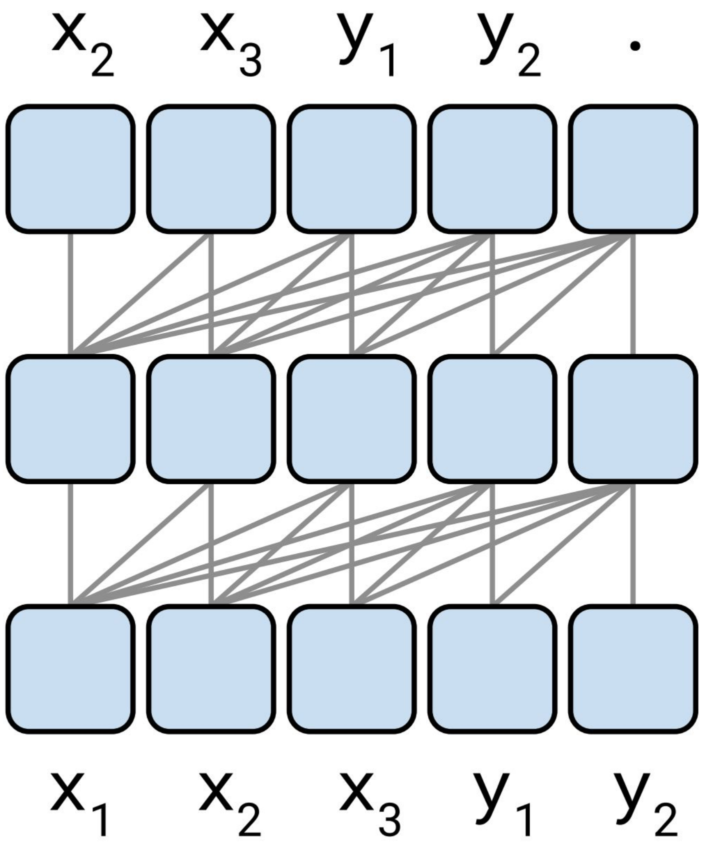
a_{11}	0	0	0
a_{21}	a_{22}	0	0
a_{31}	a_{32}	a_{33}	0
a_{41}	a_{42}	a_{43}	a_{44}

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

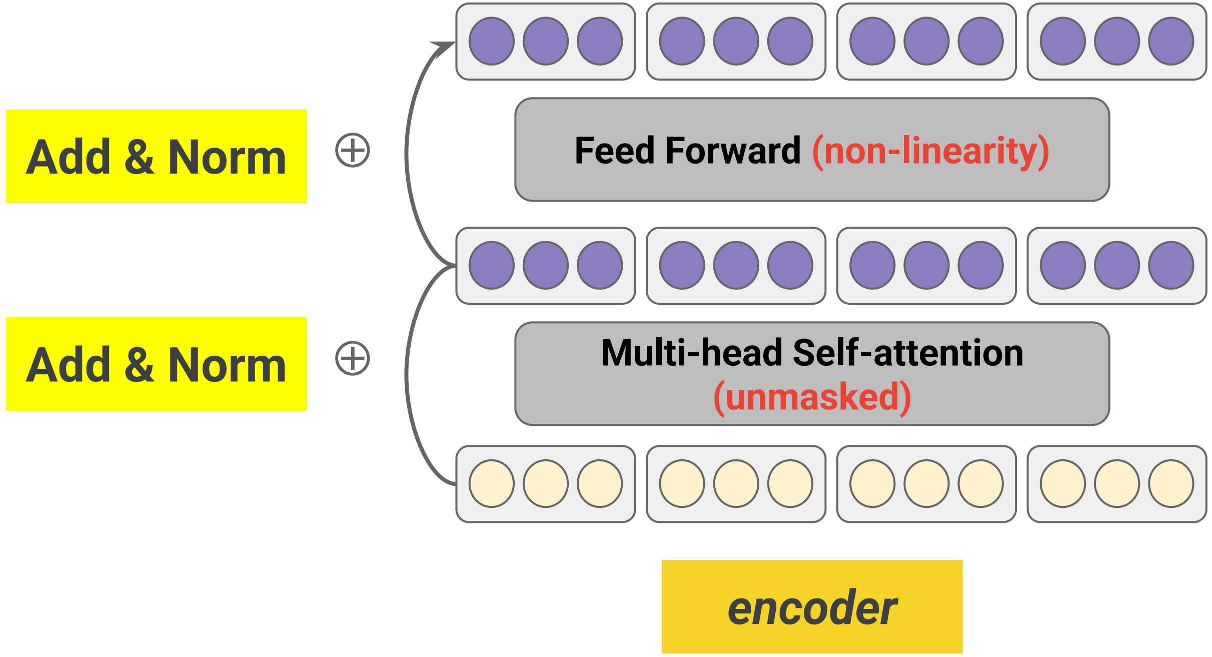
Decoder (cont'd)



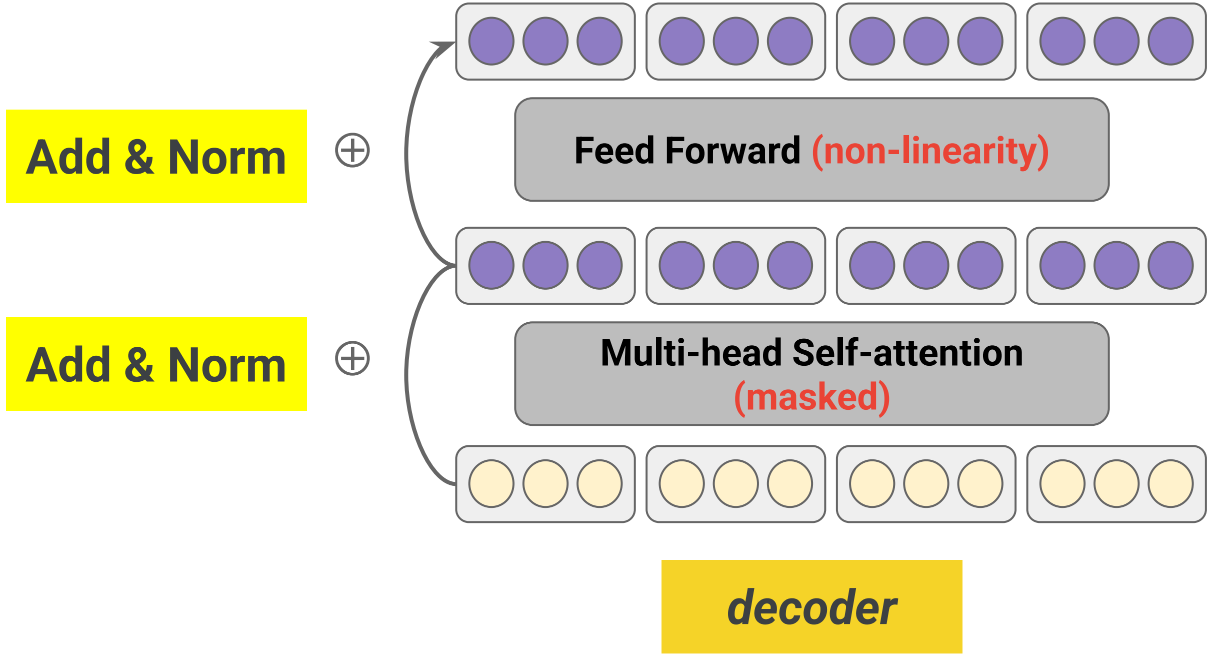
Decoder (cont'd)



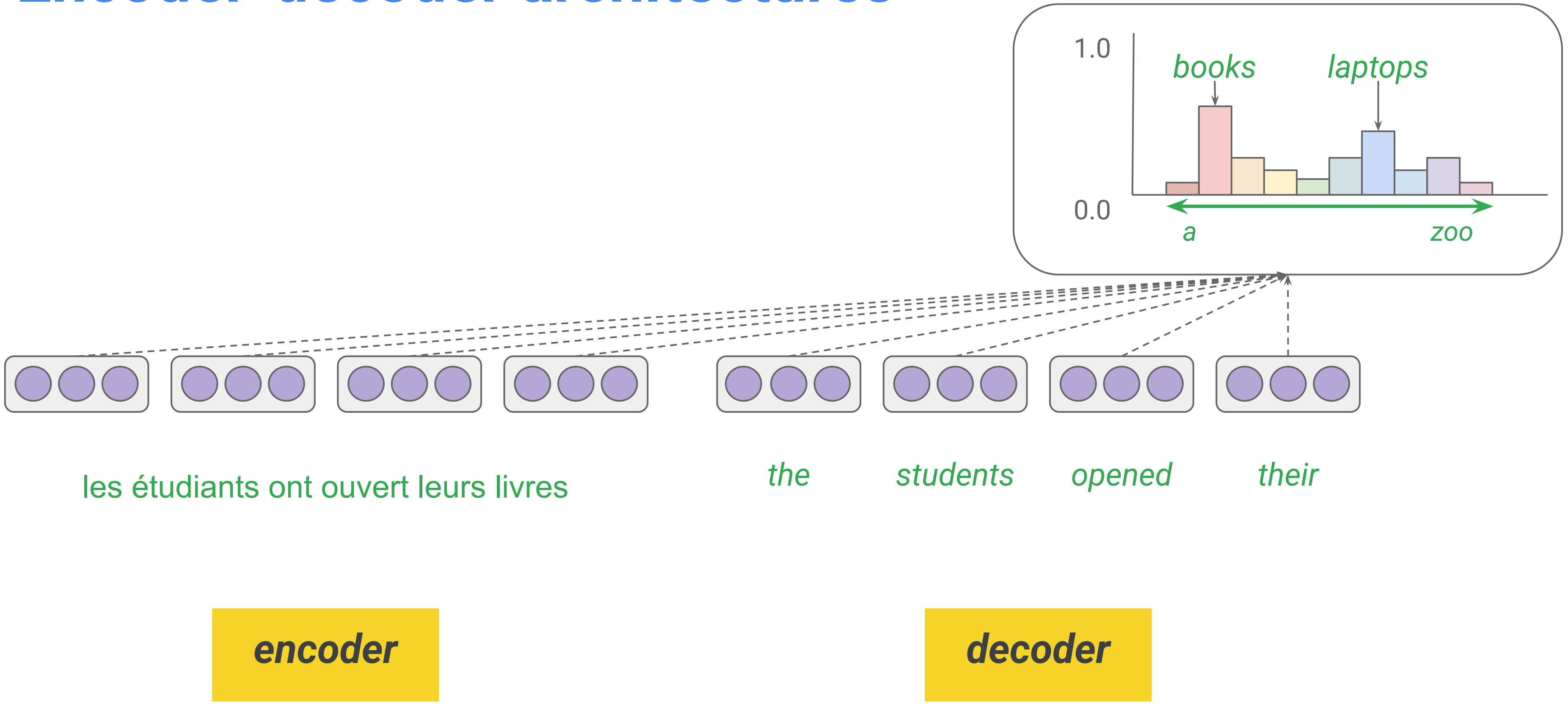
Encoder (one layer)


Add & Norm
Add & Norm
Feed Forward (non-linearity)
Multi-head Self-attention (unmasked)
encoder

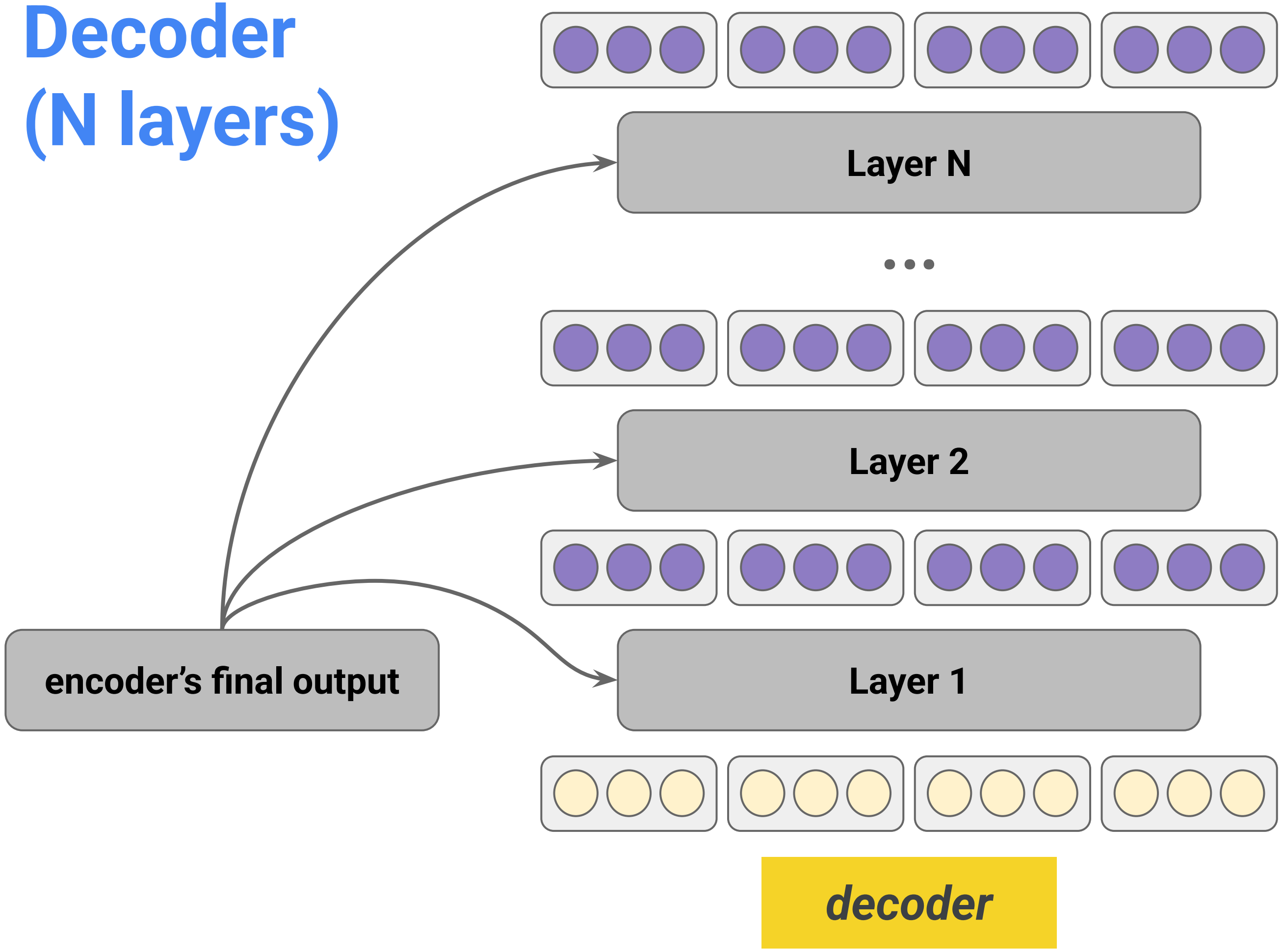
Decoder (one layer)


Add & Norm
Add & Norm
Feed Forward (non-linearity)
Multi-head Self-attention (masked)
decoder

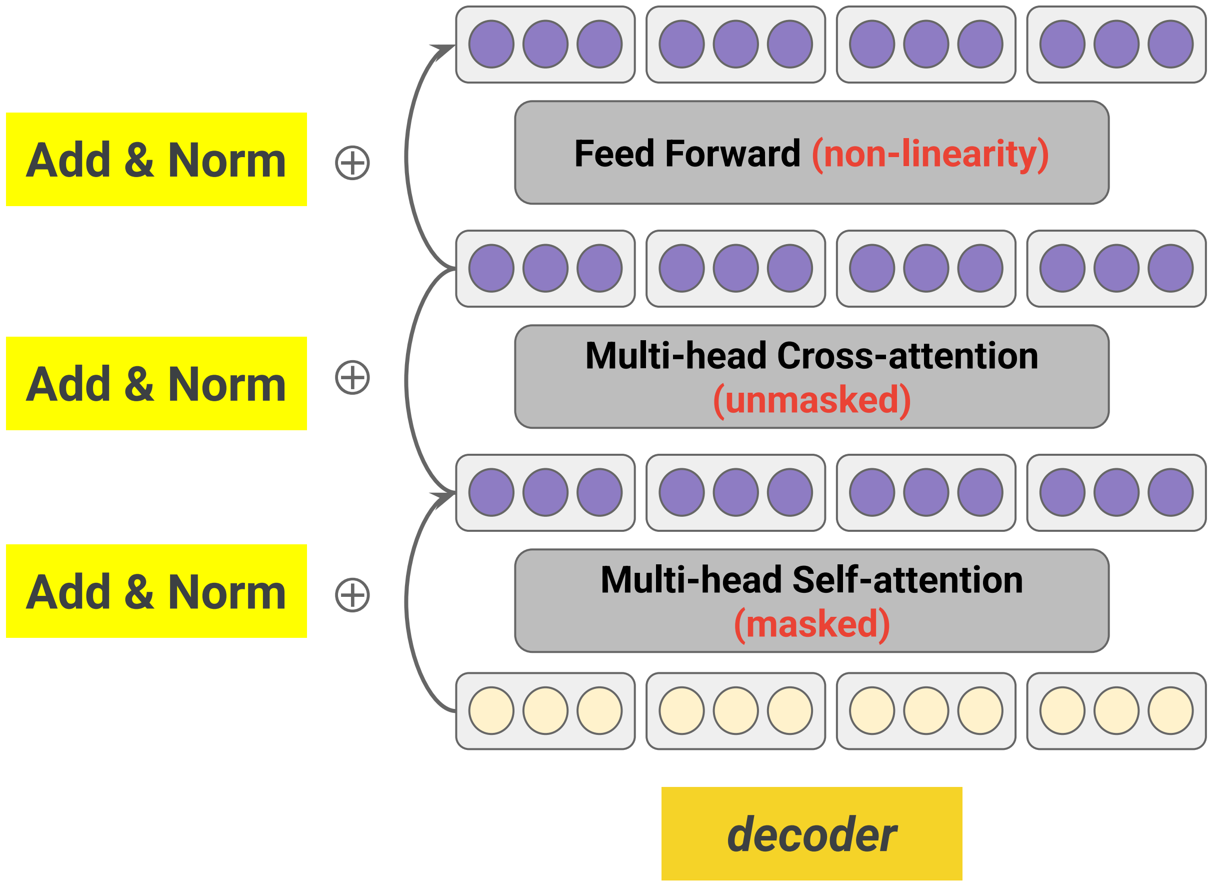
Encoder-decoder architectures



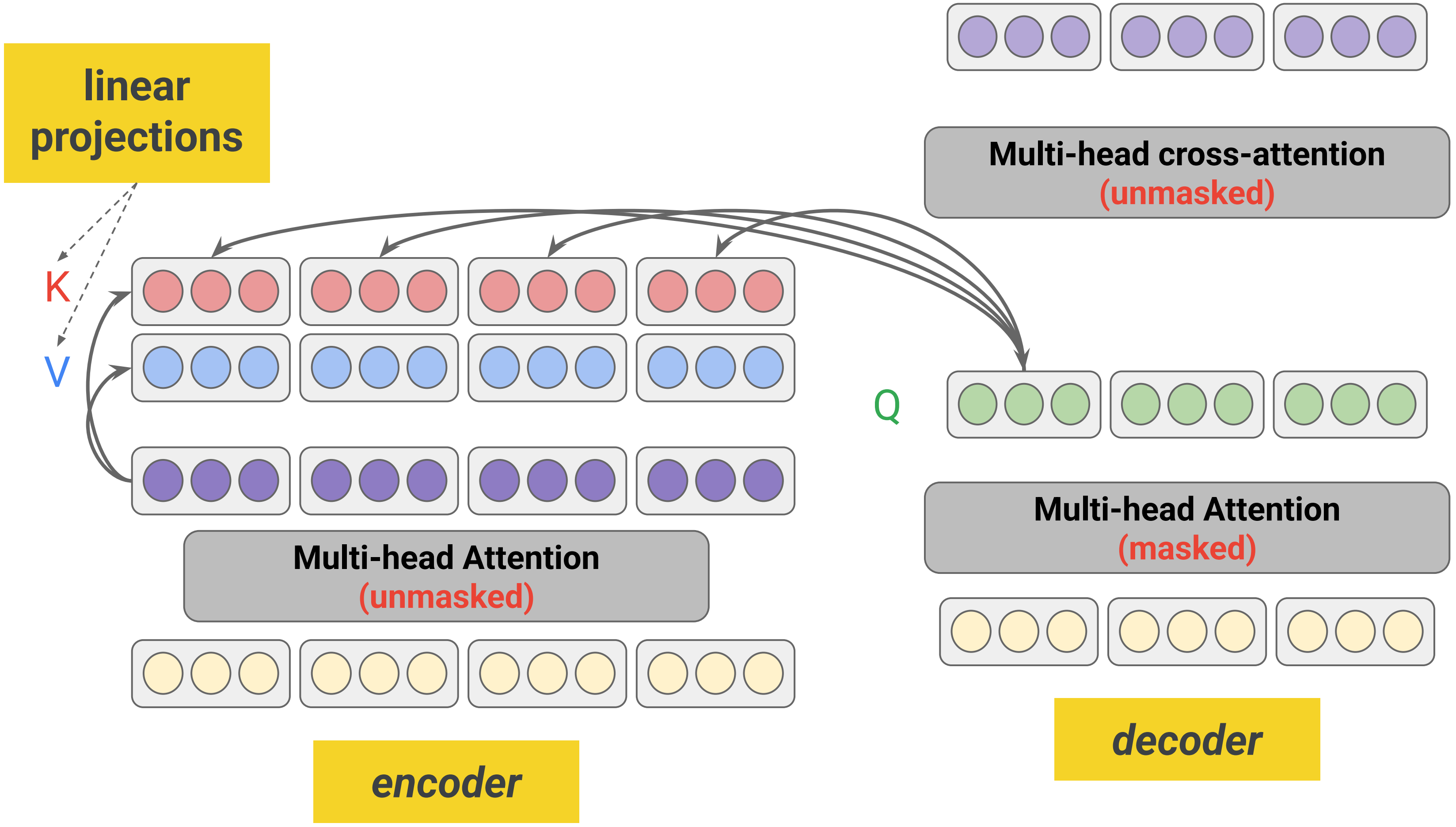
Decoder (N layers)



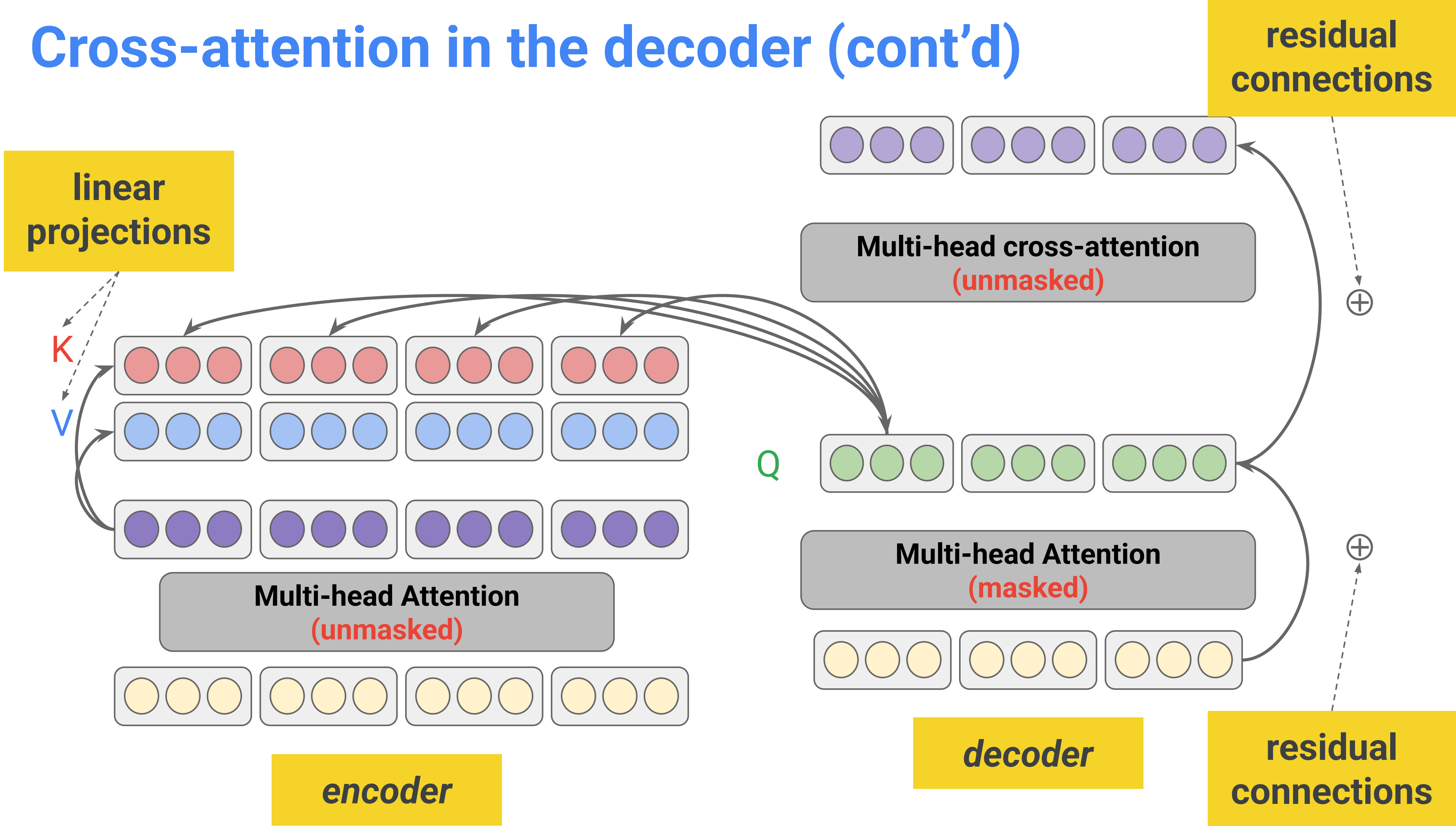
Decoder in encoder-decoder (one layer)

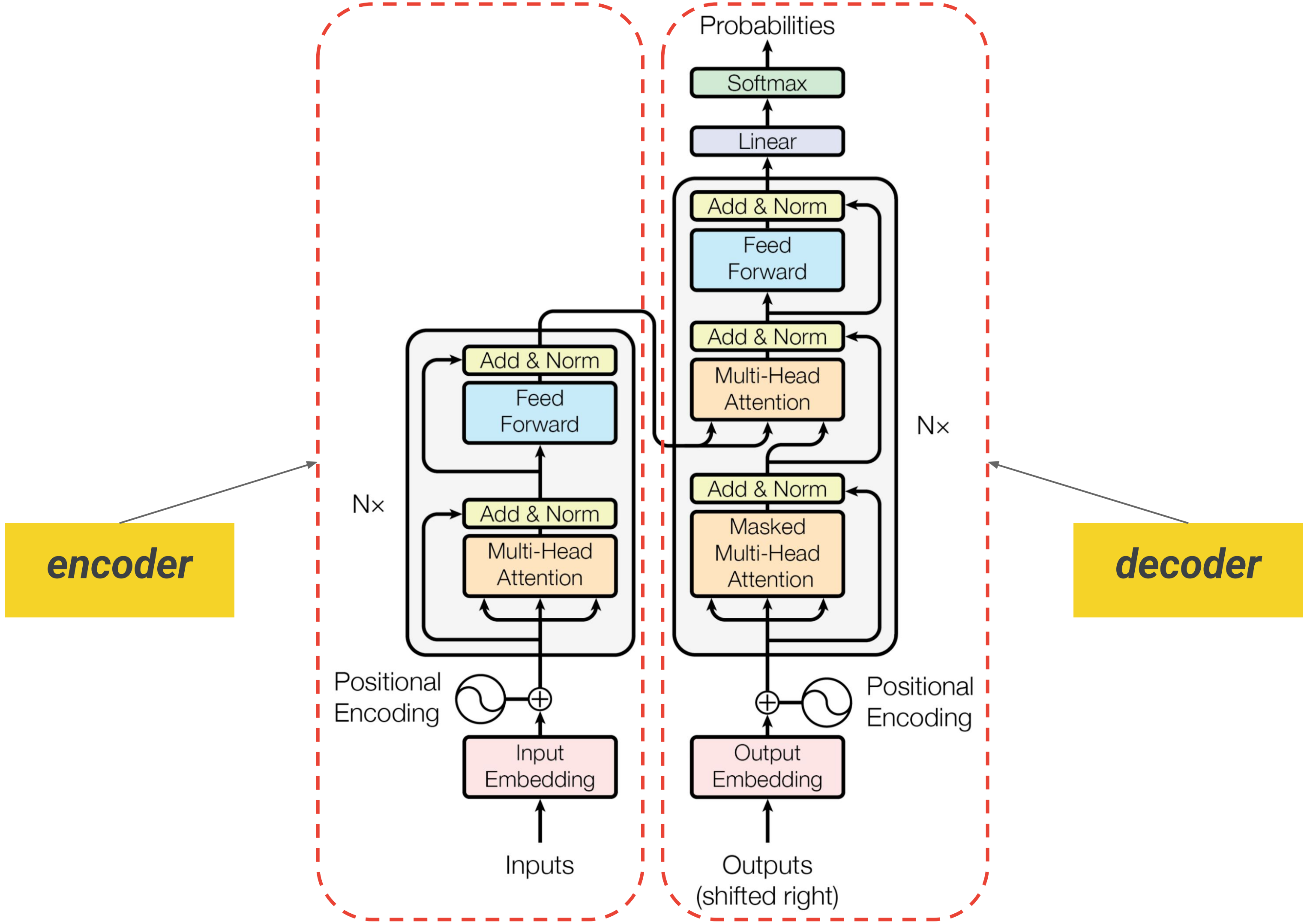

Add & Norm
⊕
Feed Forward (non-linearity)
Add & Norm
⊕
Multi-head Cross-attention (unmasked)
Add & Norm
⊕
Multi-head Self-attention (masked)
decoder

Cross-attention in the decoder

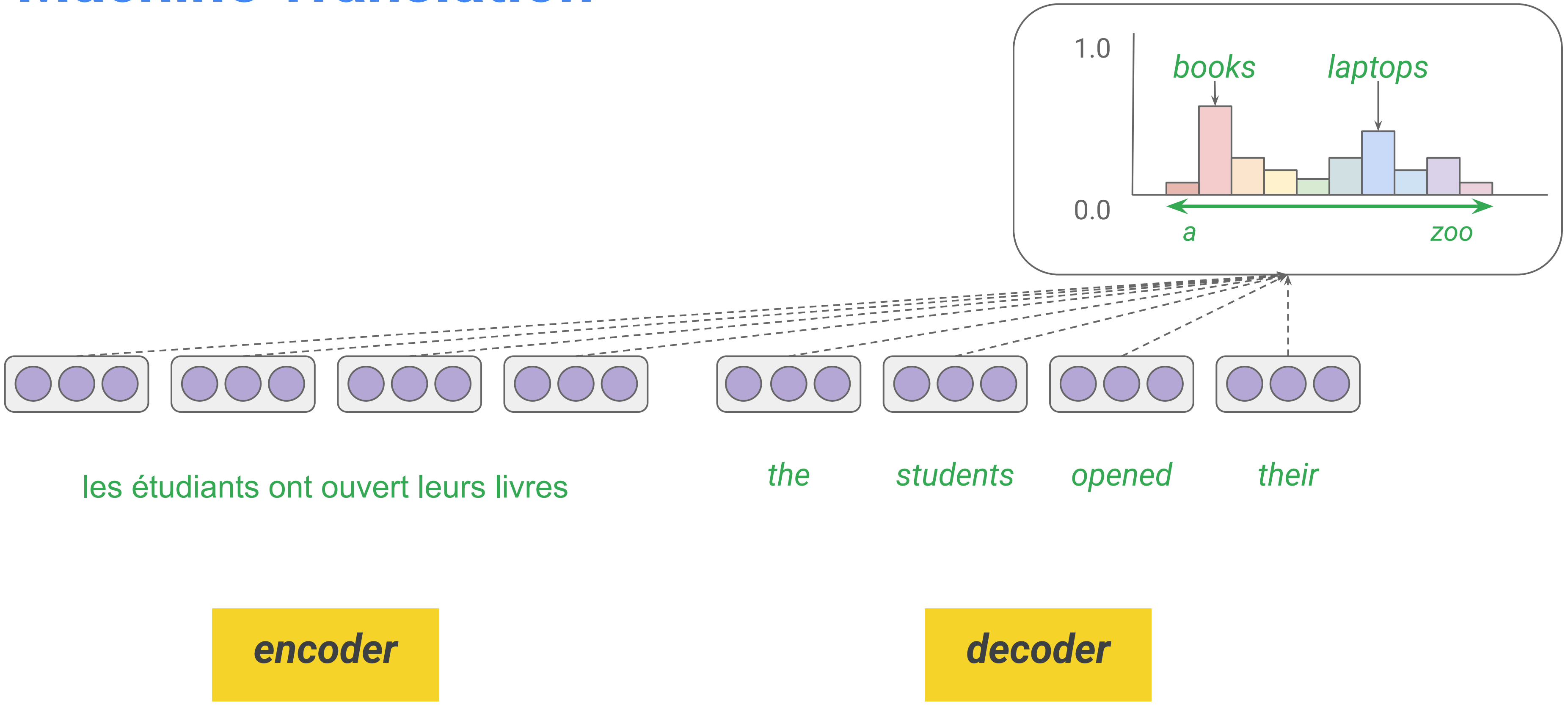


Cross-attention in the decoder (cont'd)




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

Machine Translation



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

$$h = \sigma(\boldsymbol{W}x + \underset{\text{bias term}}{b})$$

Different Transformers architectures

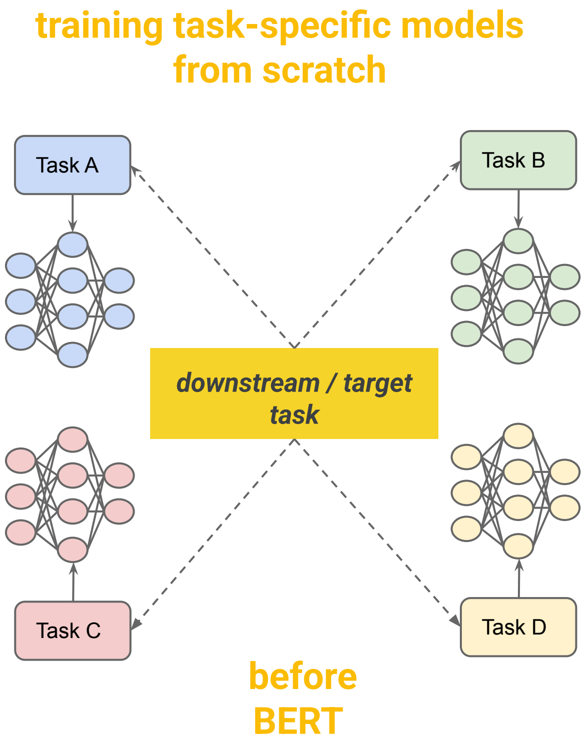
- Encoder-only
  - BERT
- Encoder-decoder
  - T5
- Decoder-only
  - GPT

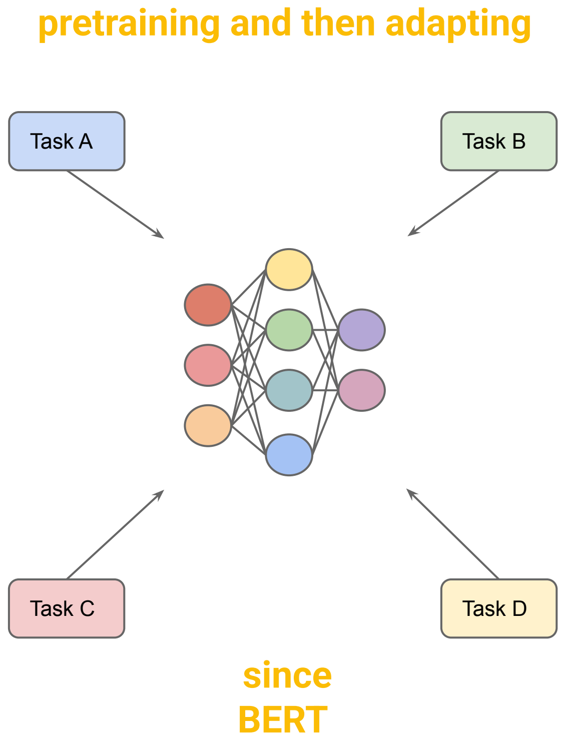
Image created by Gemini

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





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

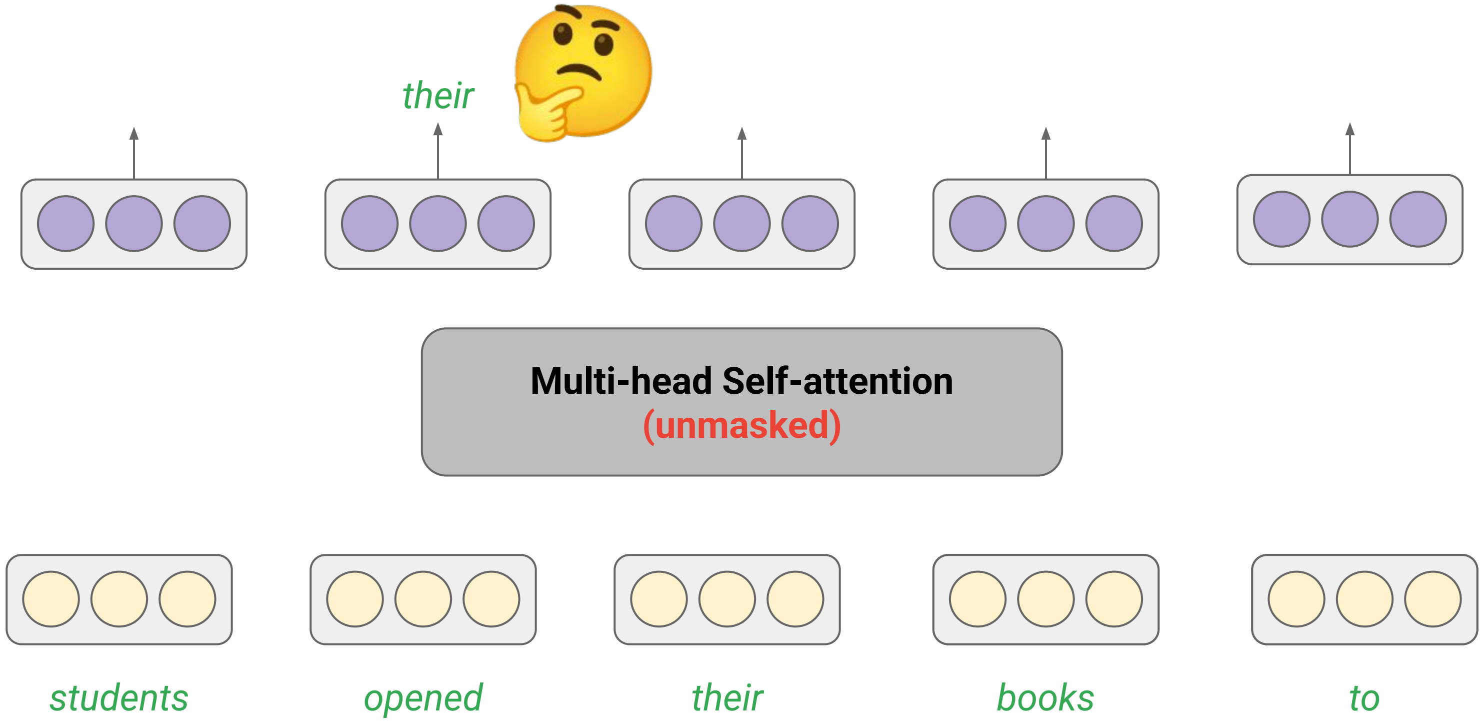
[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

BERT vs. ELMo

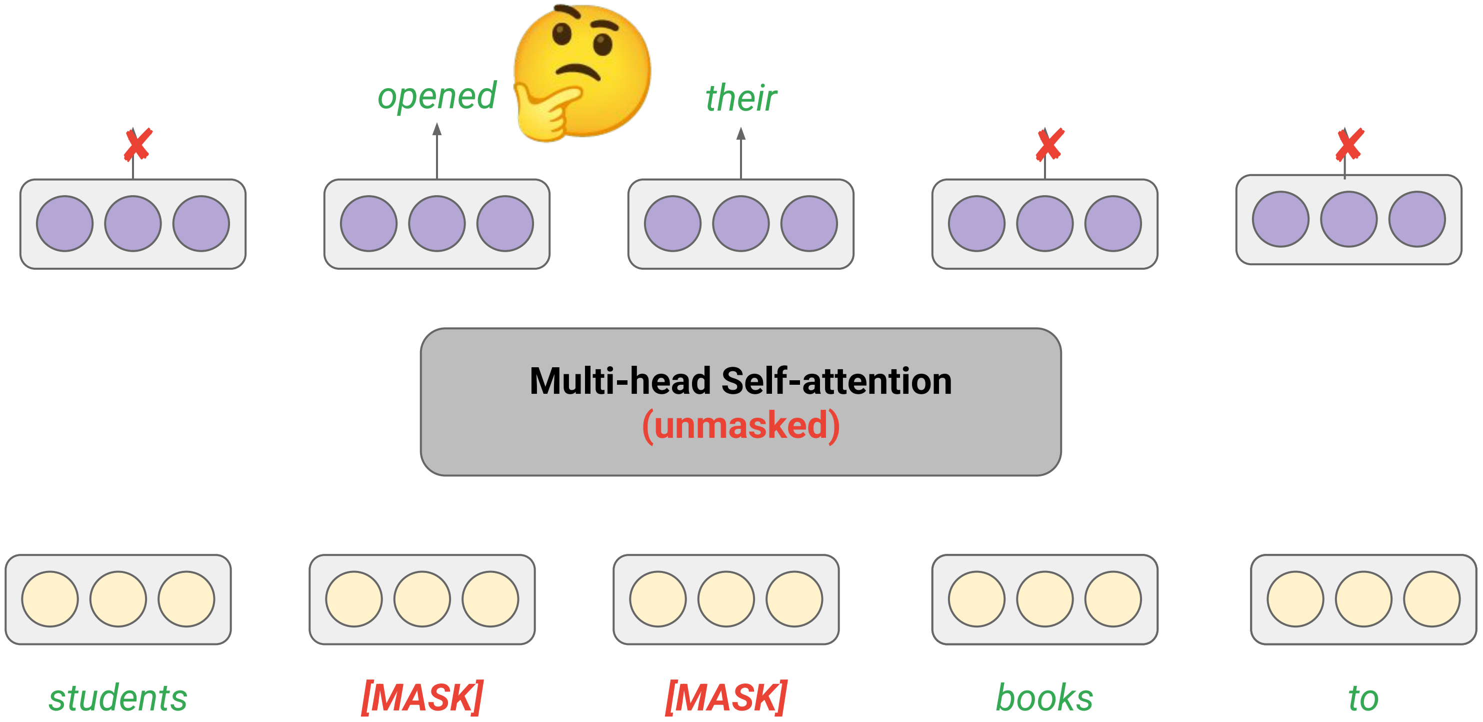
	BERT	ELMo
Model	Transformers	Bidirectional LSTM (Long Short-Term Memory, a variant of RNN)
Pre-training objective(s)	Masked language modeling + next sentence prediction	Left-to-right language modeling
Adaptation method	Fine-tuning	Feature-based (pretrained representations as additional features to task-specific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

Pretraining

Language modeling using a Transformer encoder

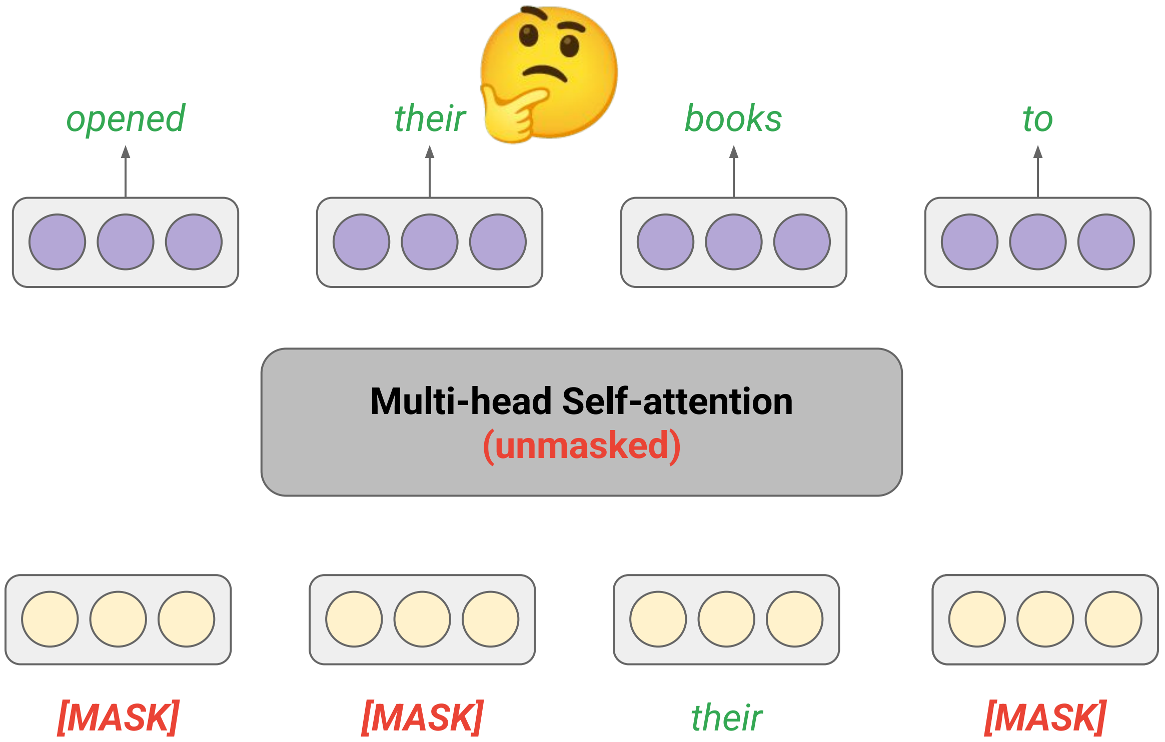


Masked language modeling



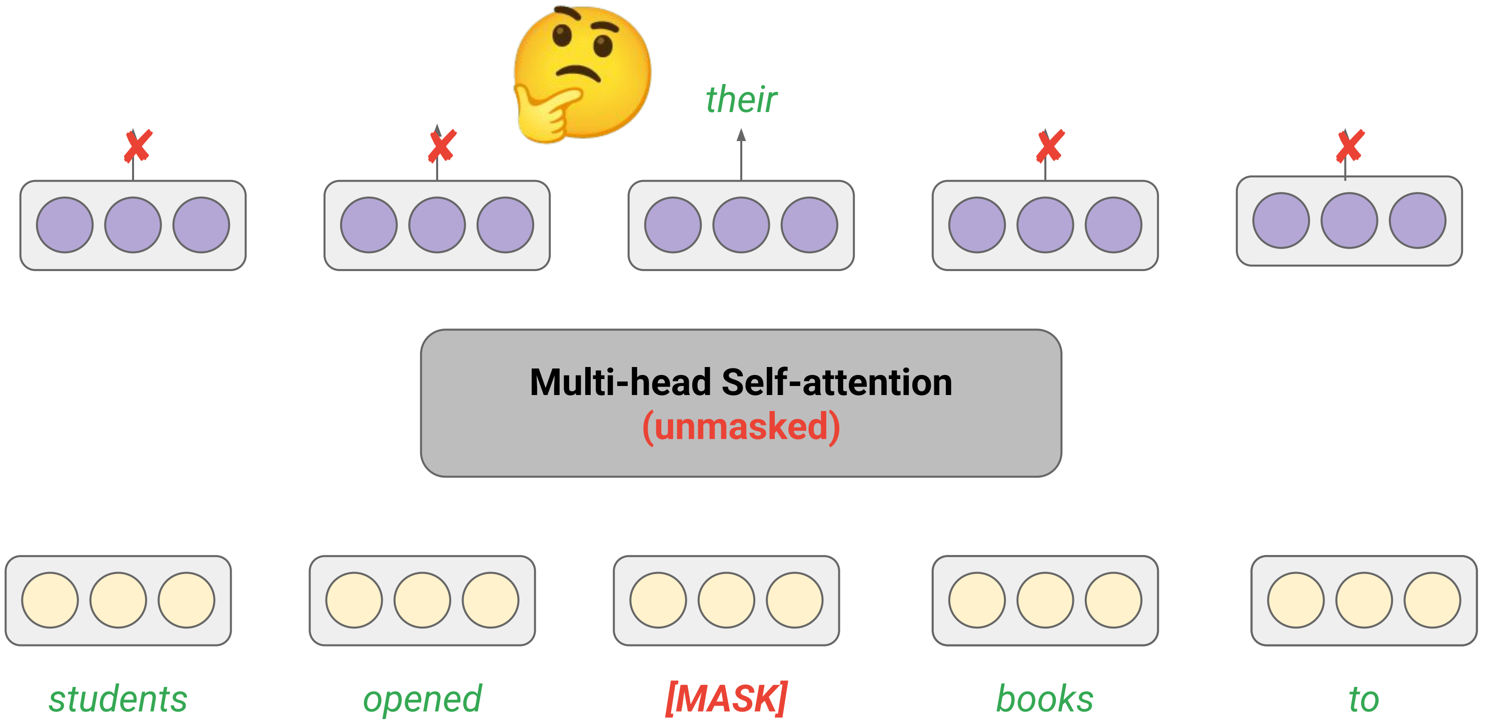
15% - 30% of all tokens in each sequence are masked at random

What if we mask more tokens?



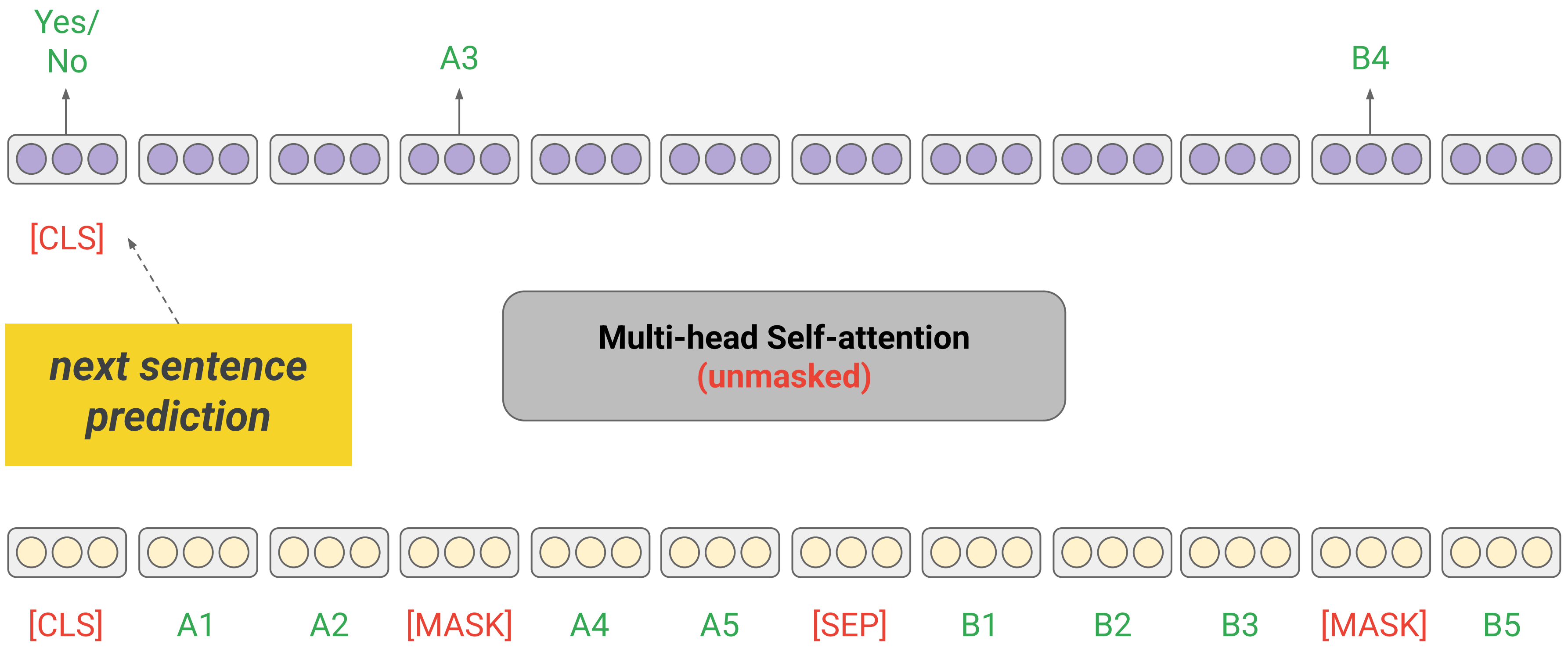
15% - 30% of all tokens in each sequence are masked at random

What if we mask less tokens?

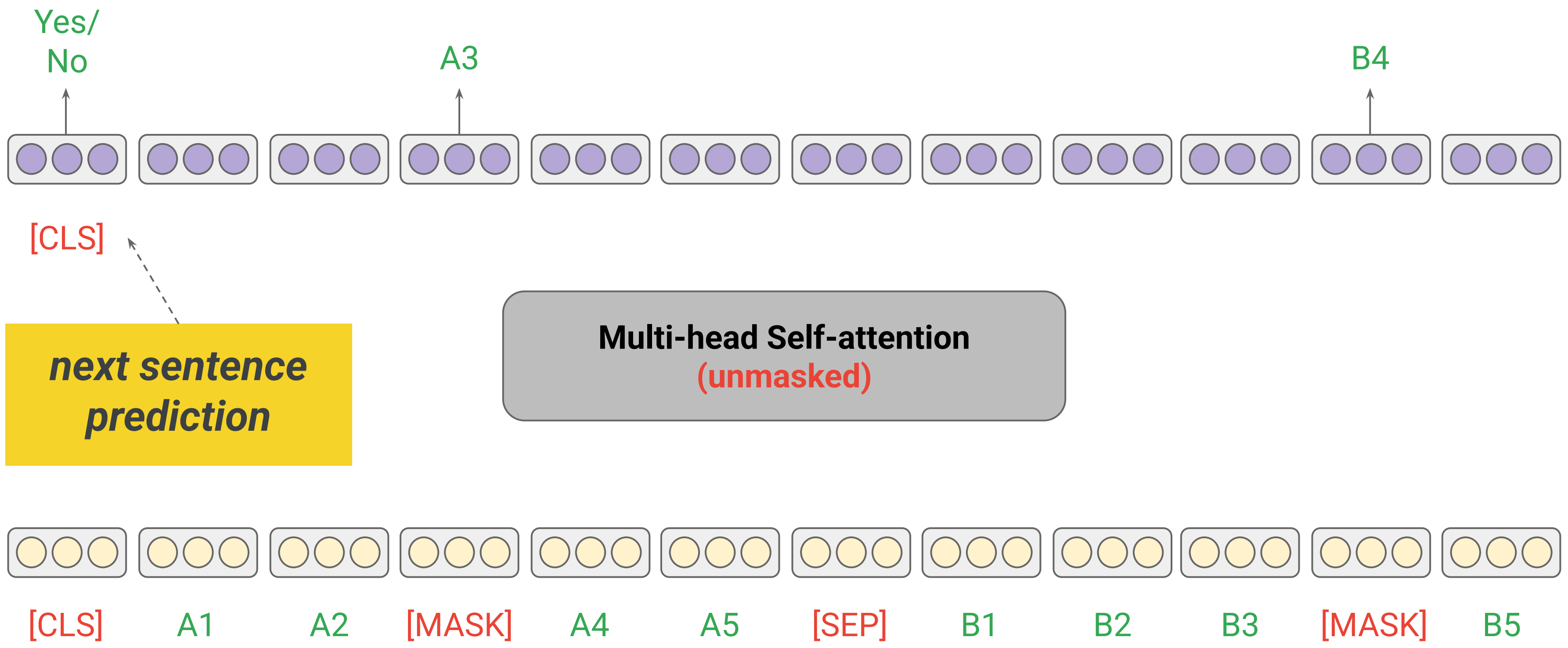


15% - 30% of all tokens in each sequence are masked at random

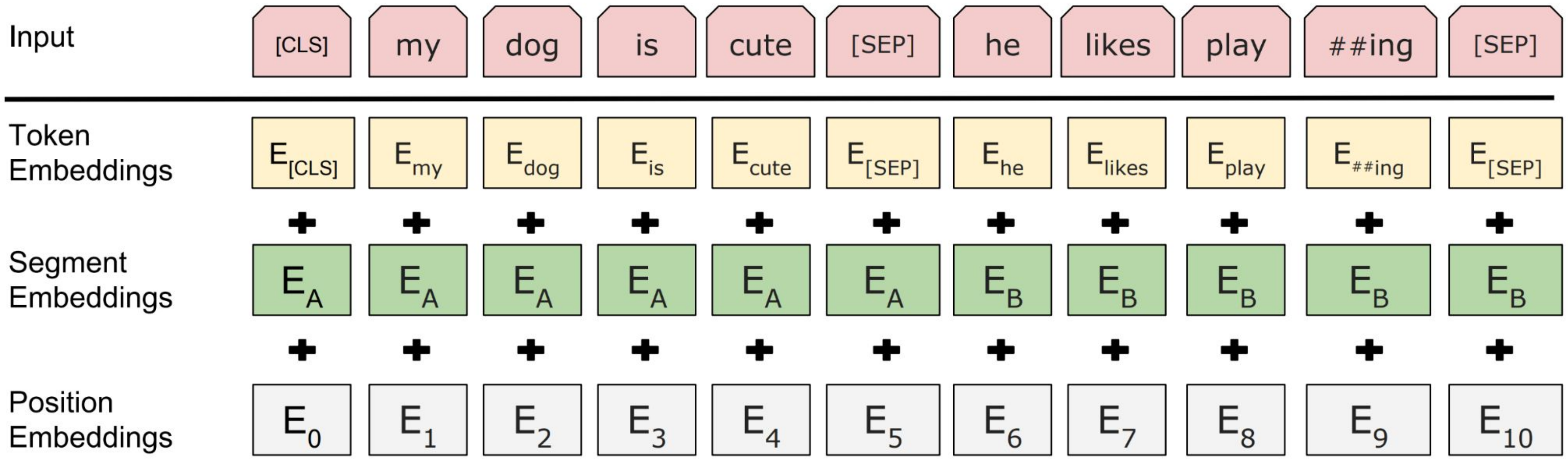
CLS & SEP tokens



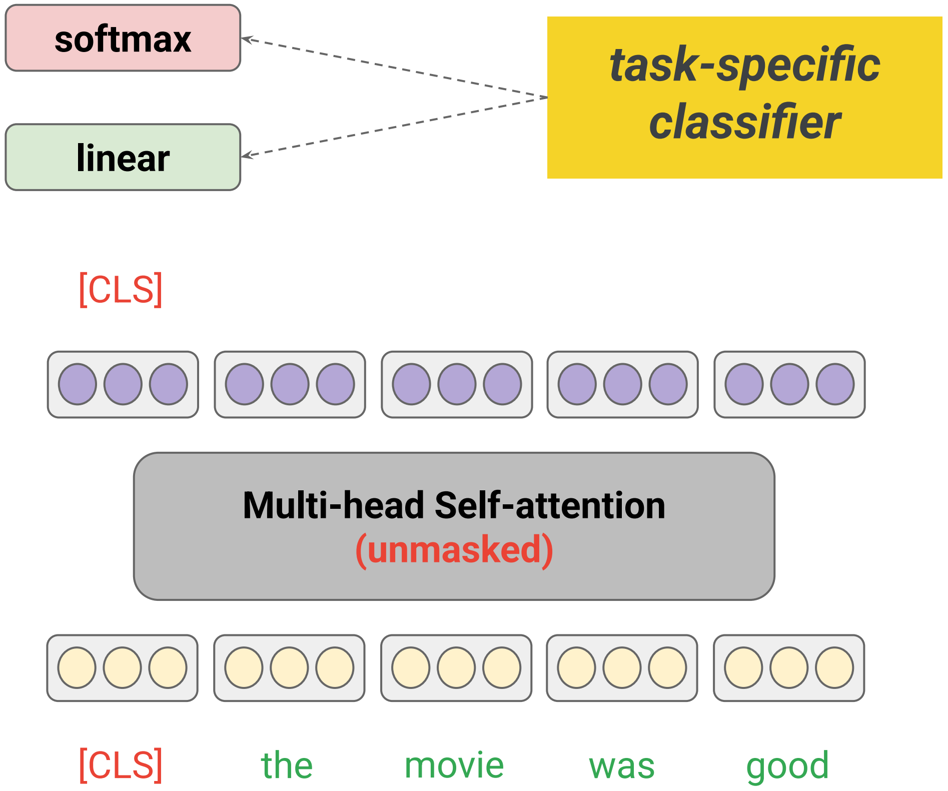
BERT Pretraining



BERT input representation

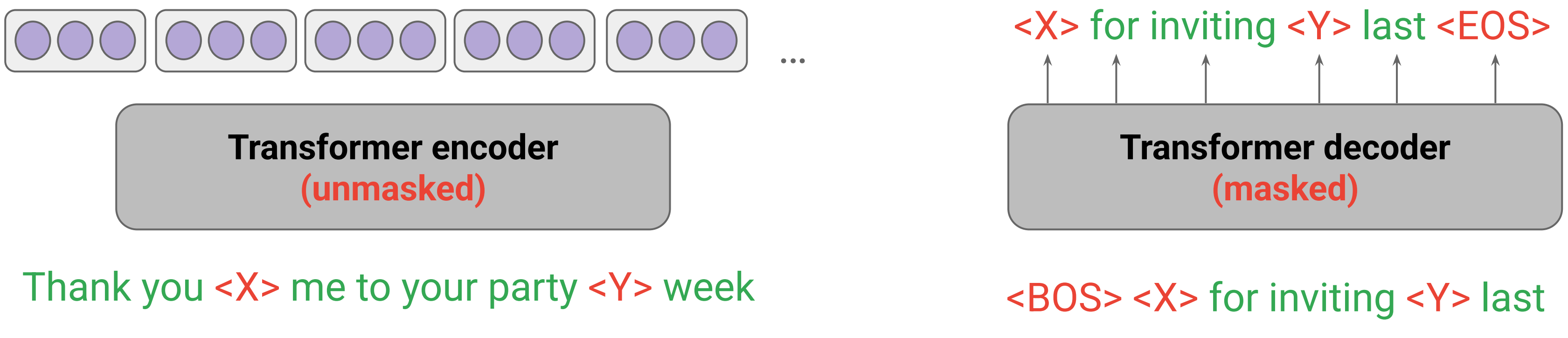


BERT Fine-tuning



T5 Pretraining: Span corruption

<X>, <Y>: sentinel tokens



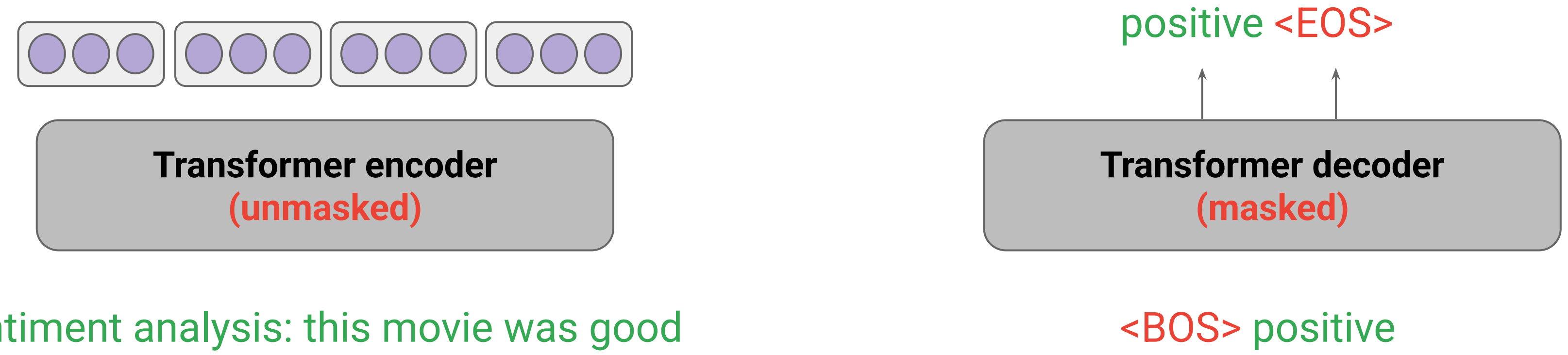
<X> for inviting <Y> last <EOS>
Transformer encoder
(unmasked)
Transformer decoder
(masked)
...
Thank you <X> me to your party <Y> week
<BOS> <X> for inviting <Y> last

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

encoder

decoder

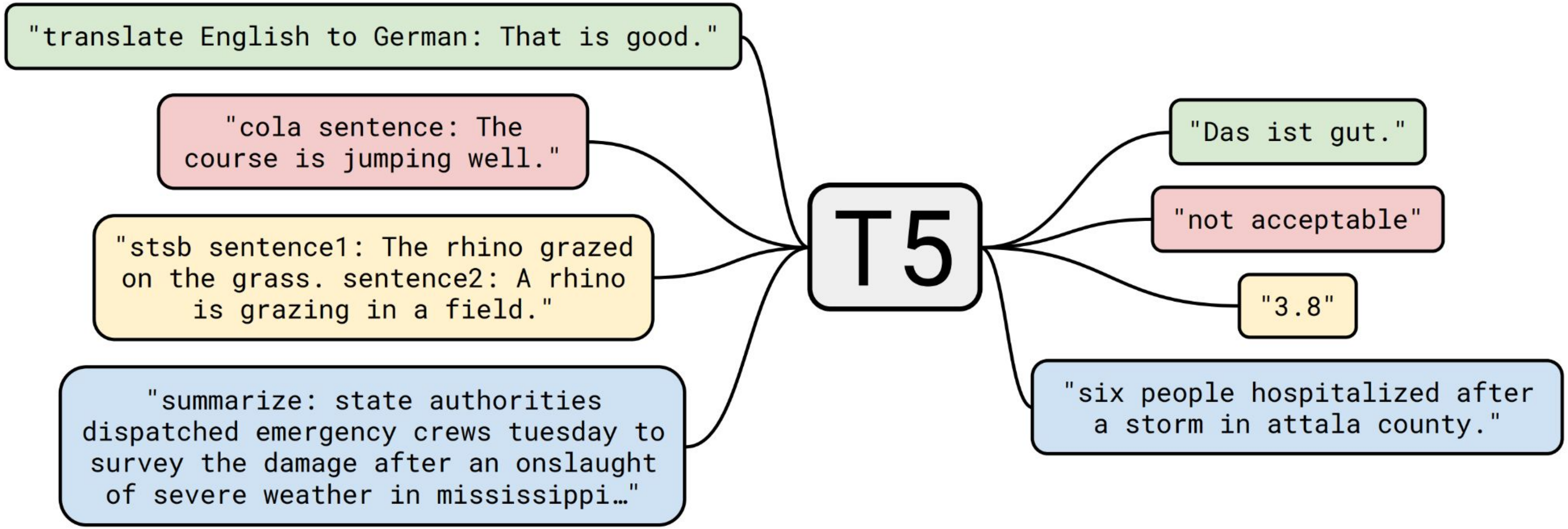
T5 Fine-tuning



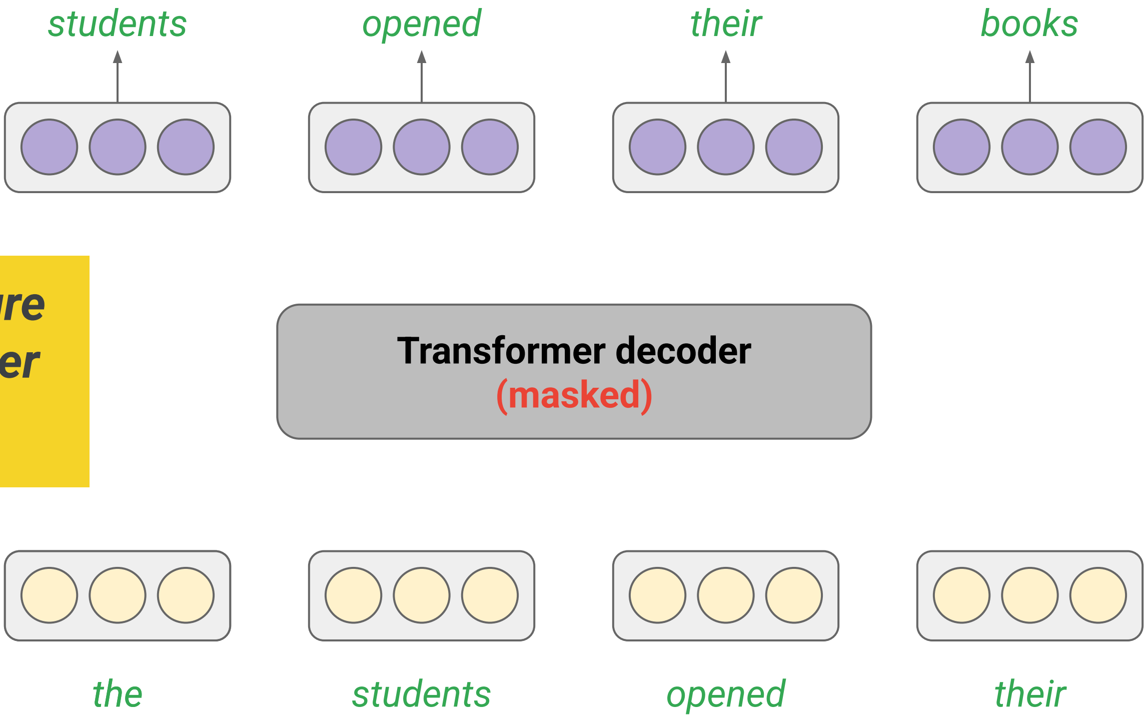
encoder

decoder

T5 facilitates multitask learning


"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Decoder-only model

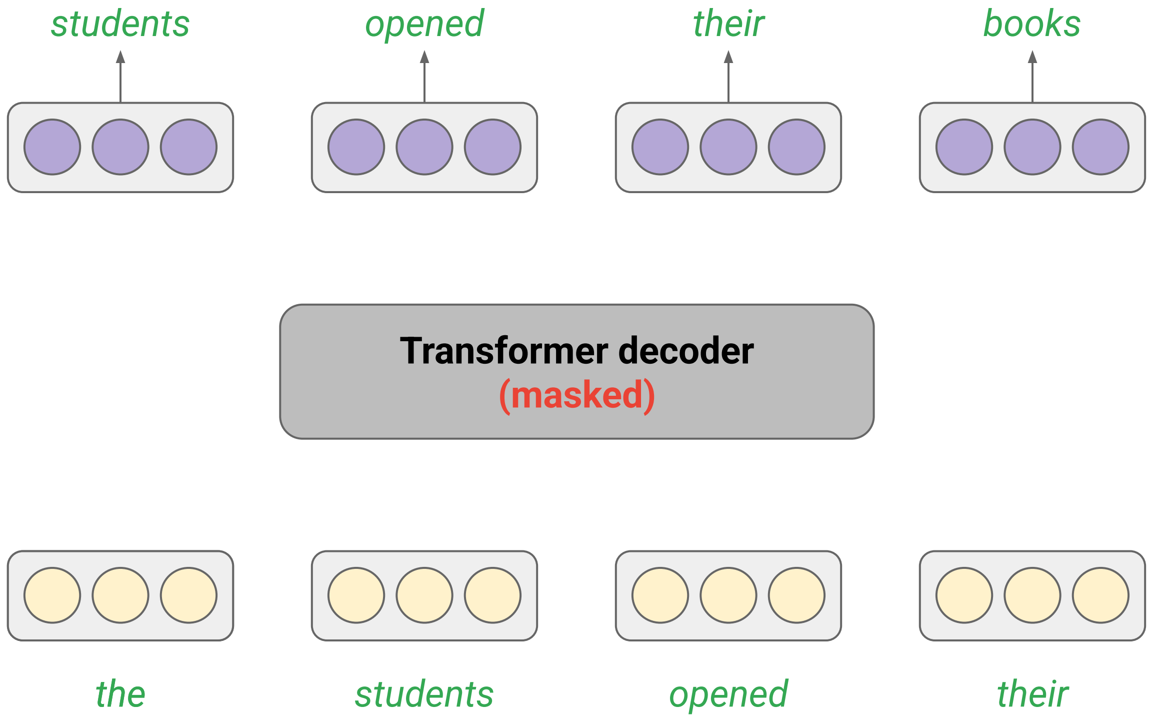


the architecture used in frontier LLMs

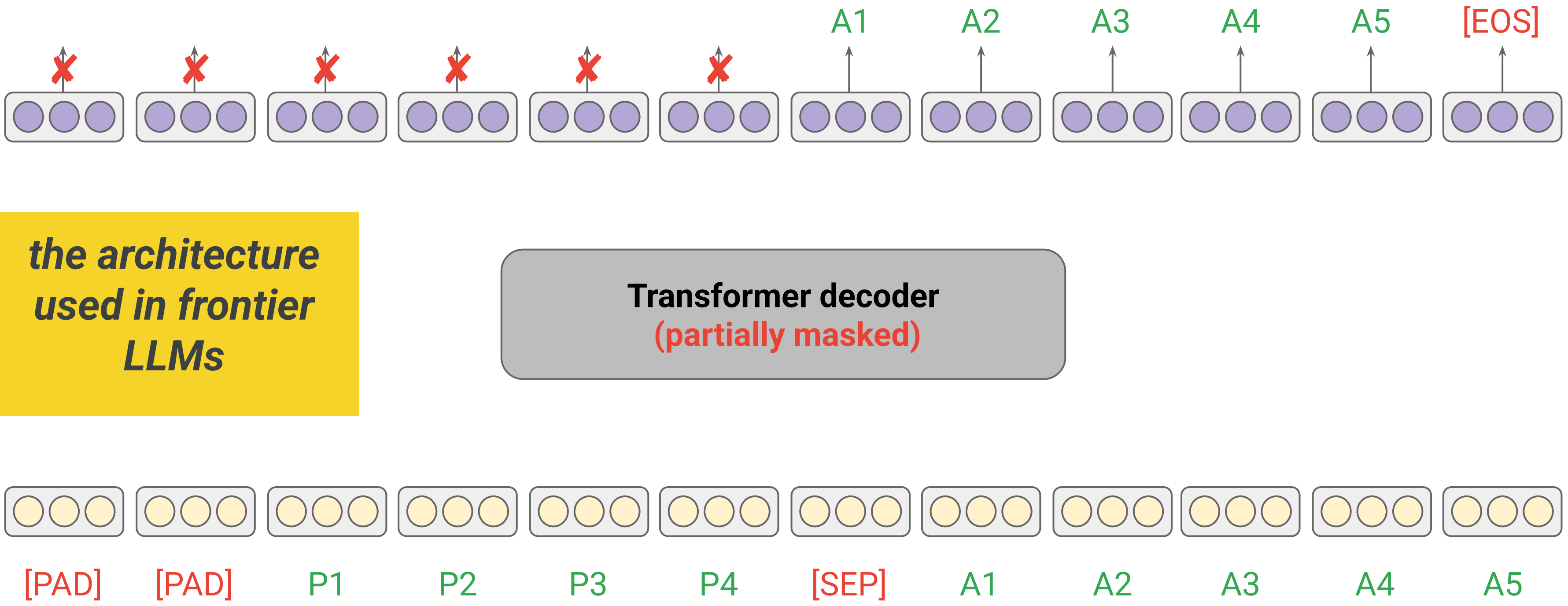
Note on cross-attention

- 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)

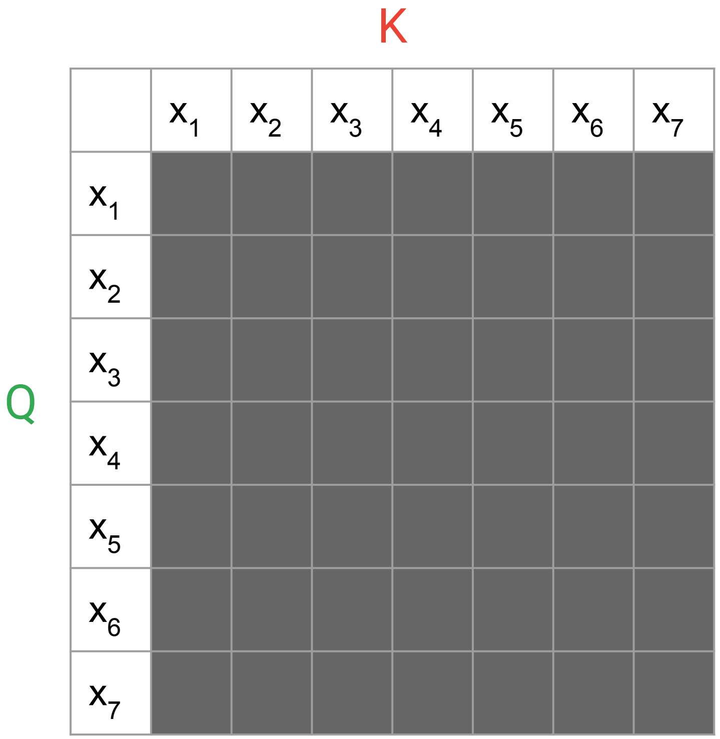


Training with prefix LM (decoder-only)



the architecture used in frontier LLMs

Different attention mask patterns

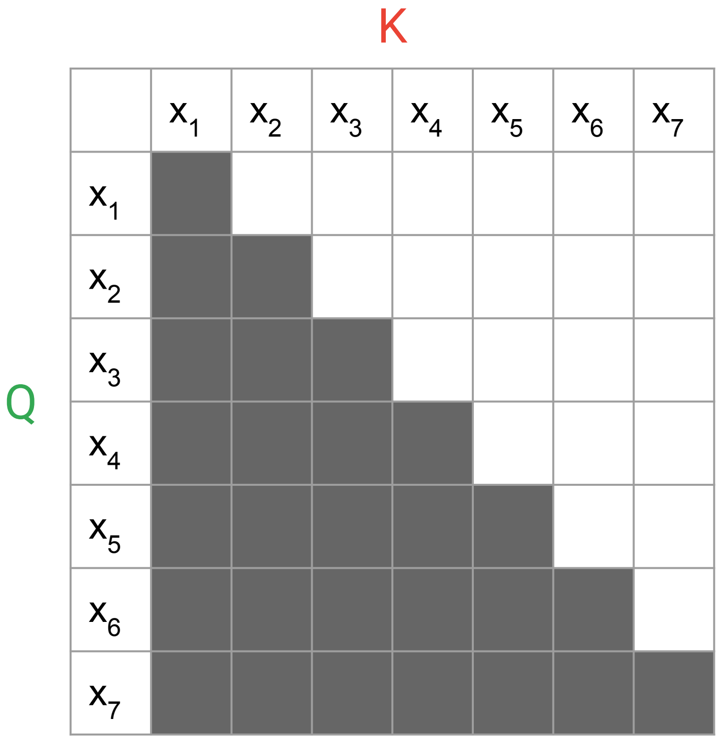


K

Q

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	■	■	■	■	■	■	■
x_2	■	■	■	■	■	■	■
x_3	■	■	■	■	■	■	■
x_4	■	■	■	■	■	■	■
x_5	■	■	■	■	■	■	■
x_6	■	■	■	■	■	■	■
x_7	■	■	■	■	■	■	■

fully-visible



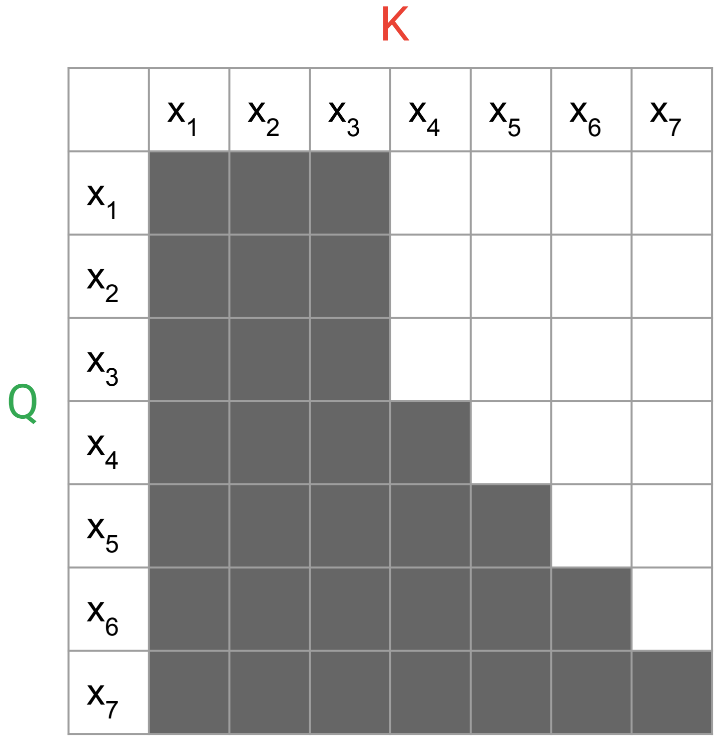
K

Q

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	■						
x_2	■	■					
x_3	■	■	■				
x_4	■	■	■	■			
x_5	■	■	■	■	■		
x_6	■	■	■	■	■	■	
x_7	■	■	■	■	■	■	■

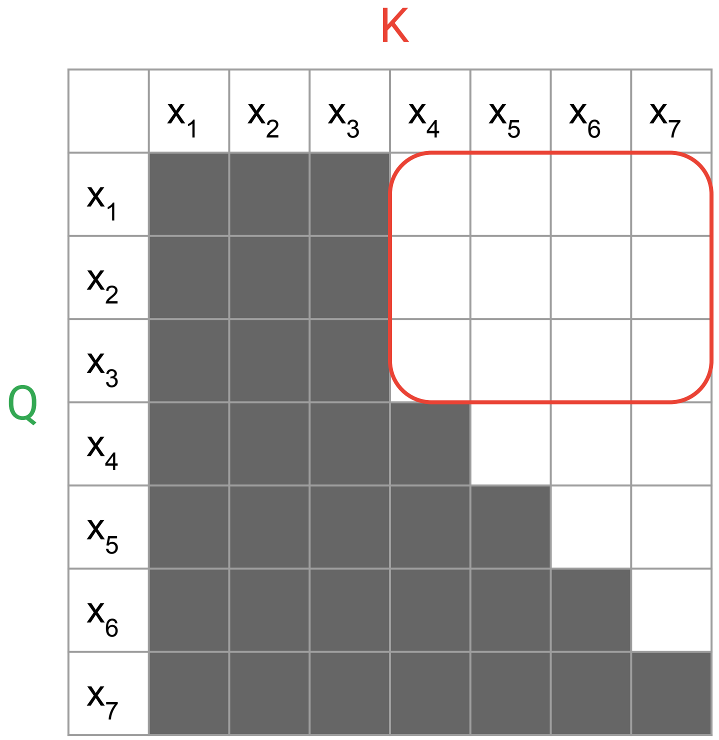
causal

Different attention mask patterns (cont'd)



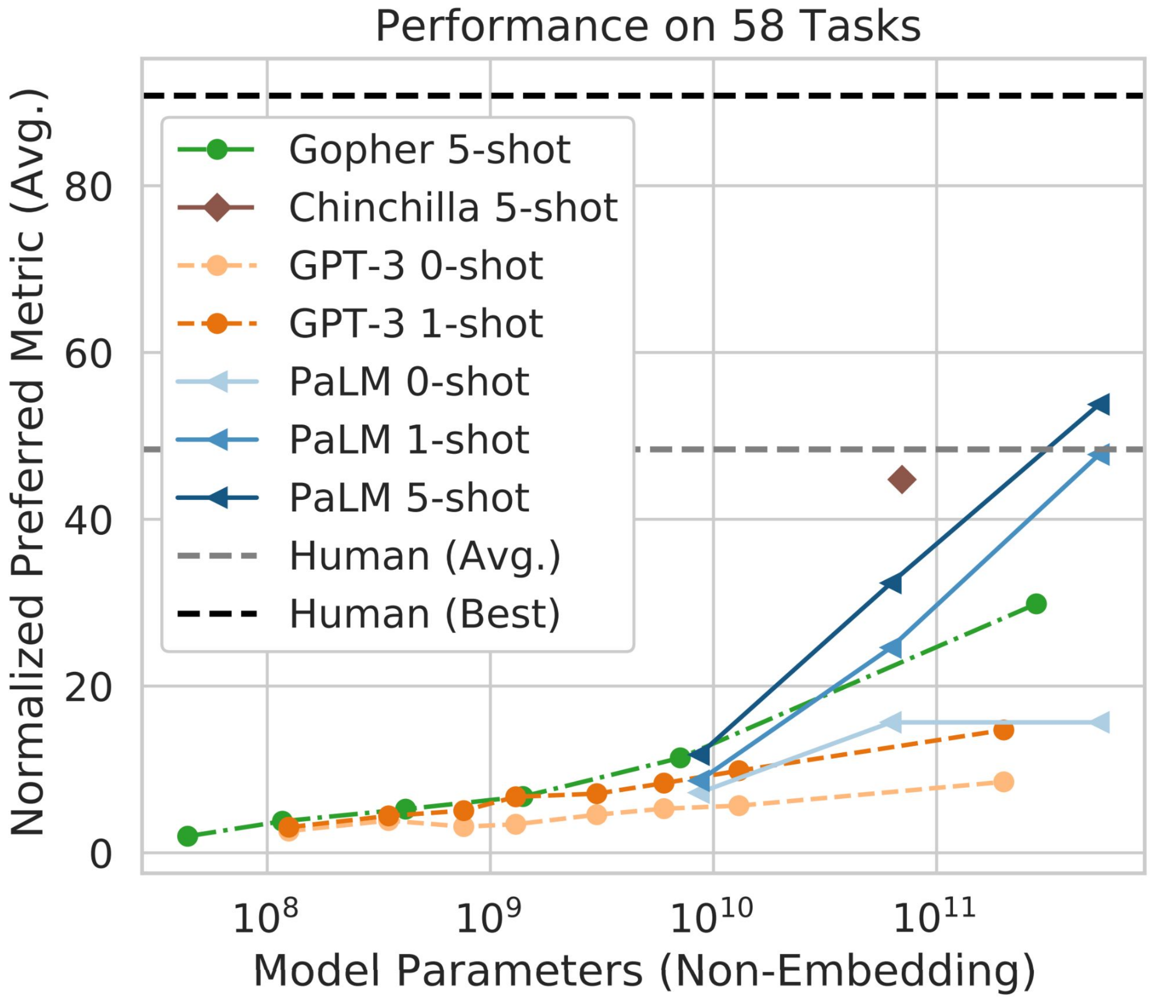
Prefix LM

Different attention mask patterns (cont'd)

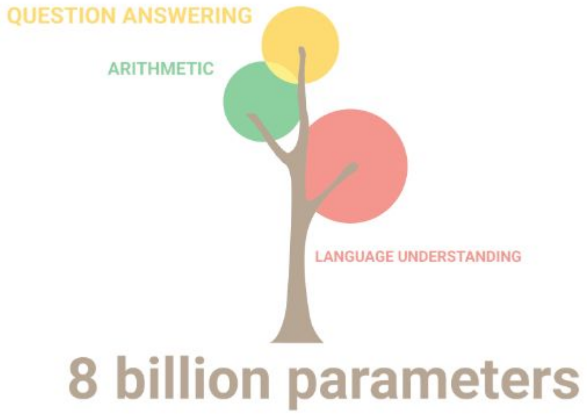

K
Q
x_1 x_2 x_3 x_4 x_5 x_6 x_7
x_1 x_2 x_3 x_4 x_5 x_6 x_7

Why masking here?

Increasing model size enhances performance and sample efficiency

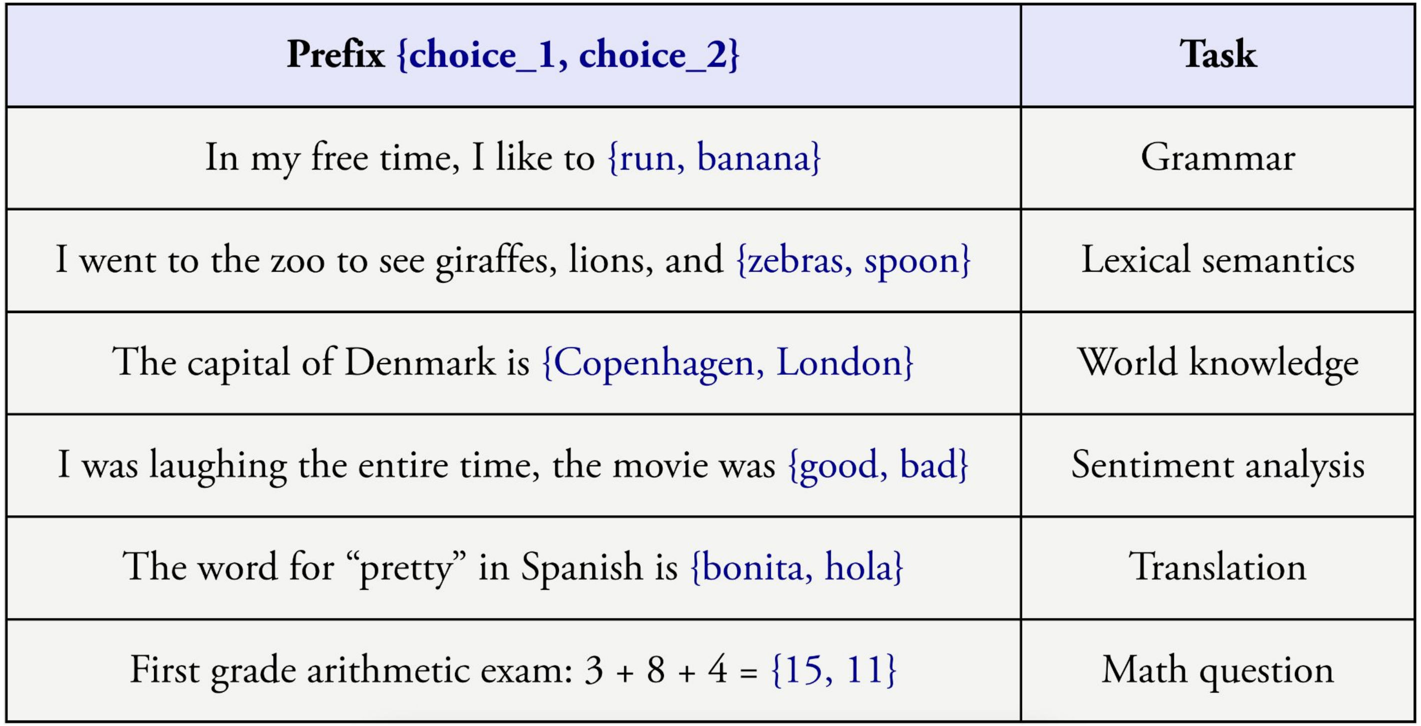


… and unlocks new capabilities



From "PaLM: Scaling Language Modeling with Pathways" by Chowdhery et al. (2022)

Why do LLMs work so well? Pretraining = Massively multi-task learning?



Prefix {choice_1, choice_2}	**Task**
In my free time, I like to {run, banana}	Grammar
I went to the zoo to see giraffes, lions, and {zebras, spoon}	Lexical semantics
The capital of Denmark is {Copenhagen, London}	World knowledge
I was laughing the entire time, the movie was {good, bad}	Sentiment analysis
The word for "pretty" in Spanish is {bonita, hola}	Translation
First grade arithmetic exam: 3 + 8 + 4 = {15, 11}	Math question

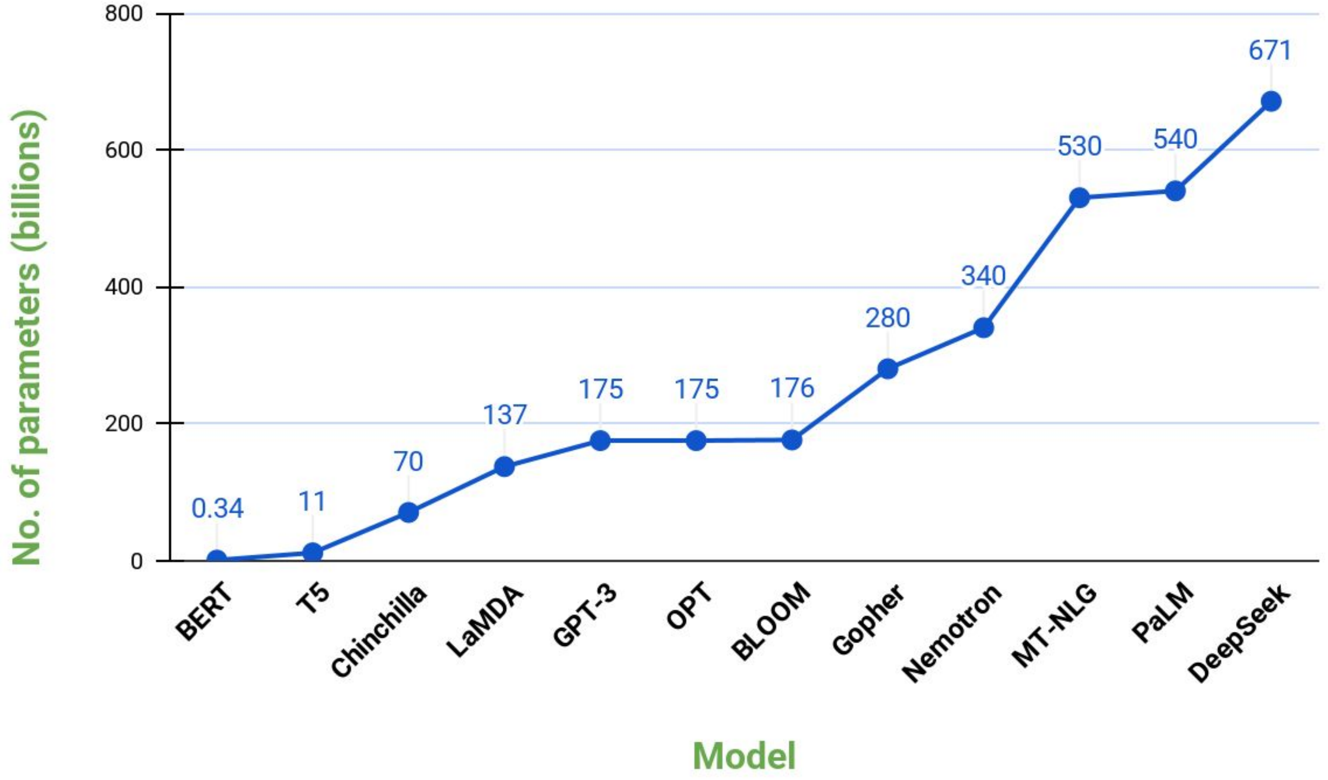
https://www.jasonwei.net/blog/some-intuitions-about-large-language-models

Why do LLMs work so well? Pretraining = Massively multi-task learning? (cont'd)

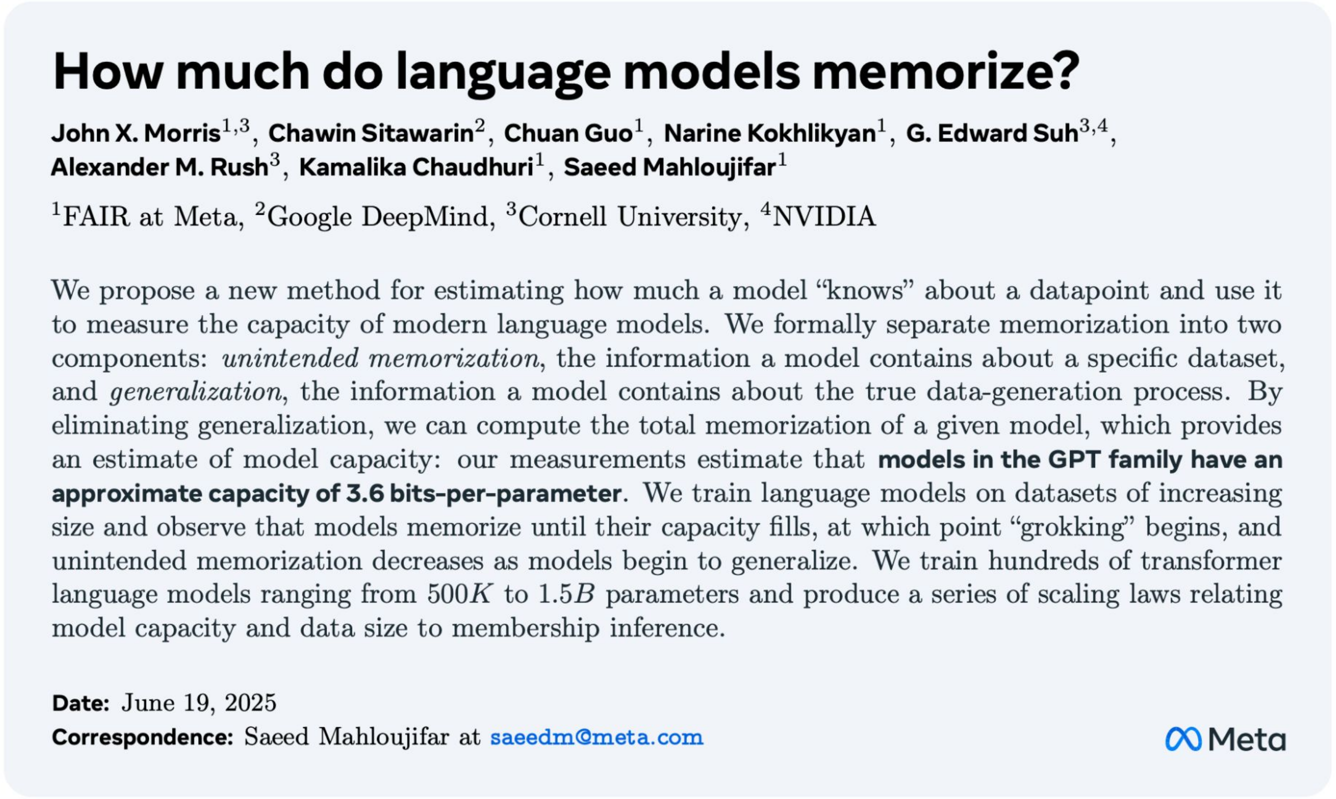
Prefix	Next word [task]
A transformer is a deep learning architecture, initially proposed in	2017 [factual recall]
A transformer is a deep learning architecture, initially proposed in 2017	, [comma prediction]
A transformer is a deep learning architecture, initially proposed in 2017,	that [grammar]
A transformer is a deep learning architecture, initially proposed in 2017, that	relies [impossible task?]

https://www.jasonwei.net/blog/some-intuitions-about-large-language-models

The trend has continued to push the boundaries of possibility in NLP



3.6 bits per parameter



How much do language models memorize?

John X. Morris[1,3], **Chawin Sitawarin**[2], **Chuan Guo**[1], **Narine Kokhlikyan**[1], **G. Edward Suh**[3,4], **Alexander M. Rush**[3], **Kamalika Chaudhuri**[1], **Saeed Mahloujifar**[1]

[1]FAIR at Meta, [2]Google DeepMind, [3]Cornell University, [4]NVIDIA

We propose a new method for estimating how much a model "knows" about a datapoint and use it to measure the capacity of modern language models. We formally separate memorization into two components: *unintended memorization*, the information a model contains about a specific dataset, and *generalization*, the information a model contains about the true data-generation process. By eliminating generalization, we can compute the total memorization of a given model, which provides an estimate of model capacity: our measurements estimate that **models in the GPT family have an approximate capacity of 3.6 bits-per-parameter**. We train language models on datasets of increasing size and observe that models memorize until their capacity fills, at which point "grokking" begins, and unintended memorization decreases as models begin to generalize. We train hundreds of transformer language models ranging from $500K$ to $1.5B$ parameters and produce a series of scaling laws relating model capacity and data size to membership inference.

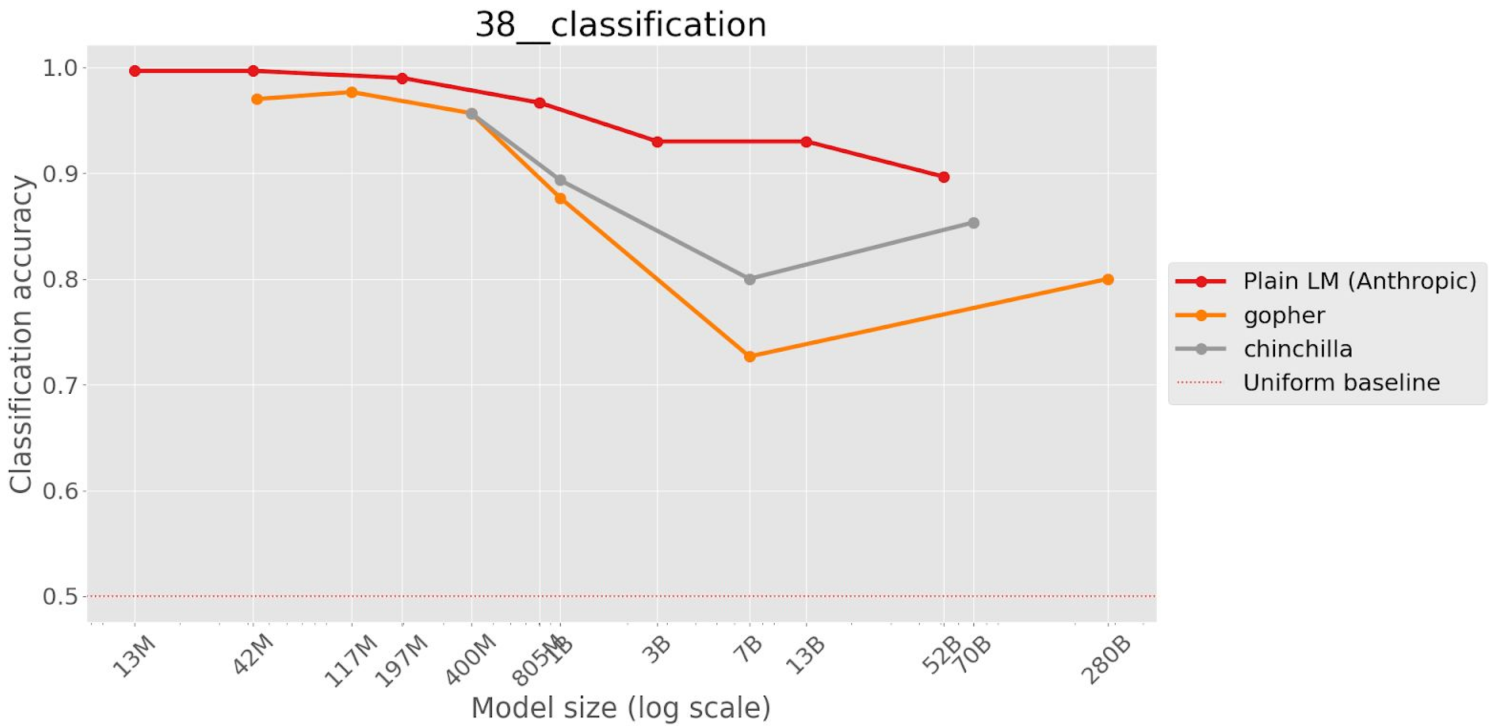
Date: June 19, 2025
Correspondence: Saeed Mahloujifar at saeedm@meta.com

Meta

Inverse scaling

- https://www.lesswrong.com/posts/iznohbCPFkeB9kAJL/inverse-scaling-prize-round-1-winners
- https://www.lesswrong.com/posts/DARiTSTx5xDLQGrrz/inverse-scaling-prize-second-round-winners

Inverse scaling (cont'd)



Repeat my sentences back to me.

Input: I like dogs.

Output: I like dogs.

Input: What is a potato, if not big?

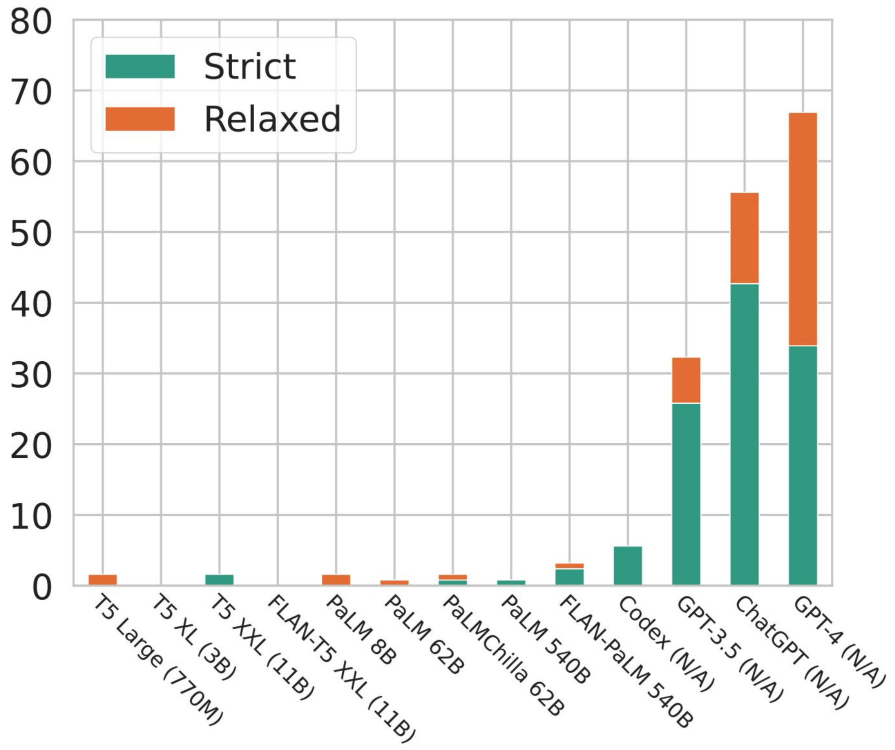
Output: What is a potato, if not big?

Input: All the world's a stage, and all the men and women merely players. They have their exits and their entrances; And one man in his time plays many pango

Output: All the world's a stage, and all the men and women merely players. They have their exits and their entrances; And one man in his time plays many

(where the model should choose ‘pango’ instead of completing the quotation with ‘part.’)

False premise questions: When did Google release ChatGPT?



False-premise questions

Vu et al. 2023:
https://arxiv.org/abs/2310.03214

What can we scale?

- The loss scales as a power-law with:
  - **N:** model size
  - **D:** dataset size
  - the amount of compute used for training (e.g., number of training steps)

$$\text{Total Steps} = \frac{\text{Dataset Size} \times \text{Epochs}}{\text{Batch Size}}$$

Where:

- **Dataset Size**: Total number of training examples.
- **Epochs**: Number of times the model sees the entire dataset.
- **Batch Size**: Number of samples per batch update.

Given a fixed compute budget, what is the optimal model size and dataset size for training?

Let’s say you can use one GPU for one day

- ○ Would you train a 5 million parameter LM on 100 books?
- ○ What about a 500 million parameter LM on one book?
- ○ Or a 100k parameter LM on 5k books?

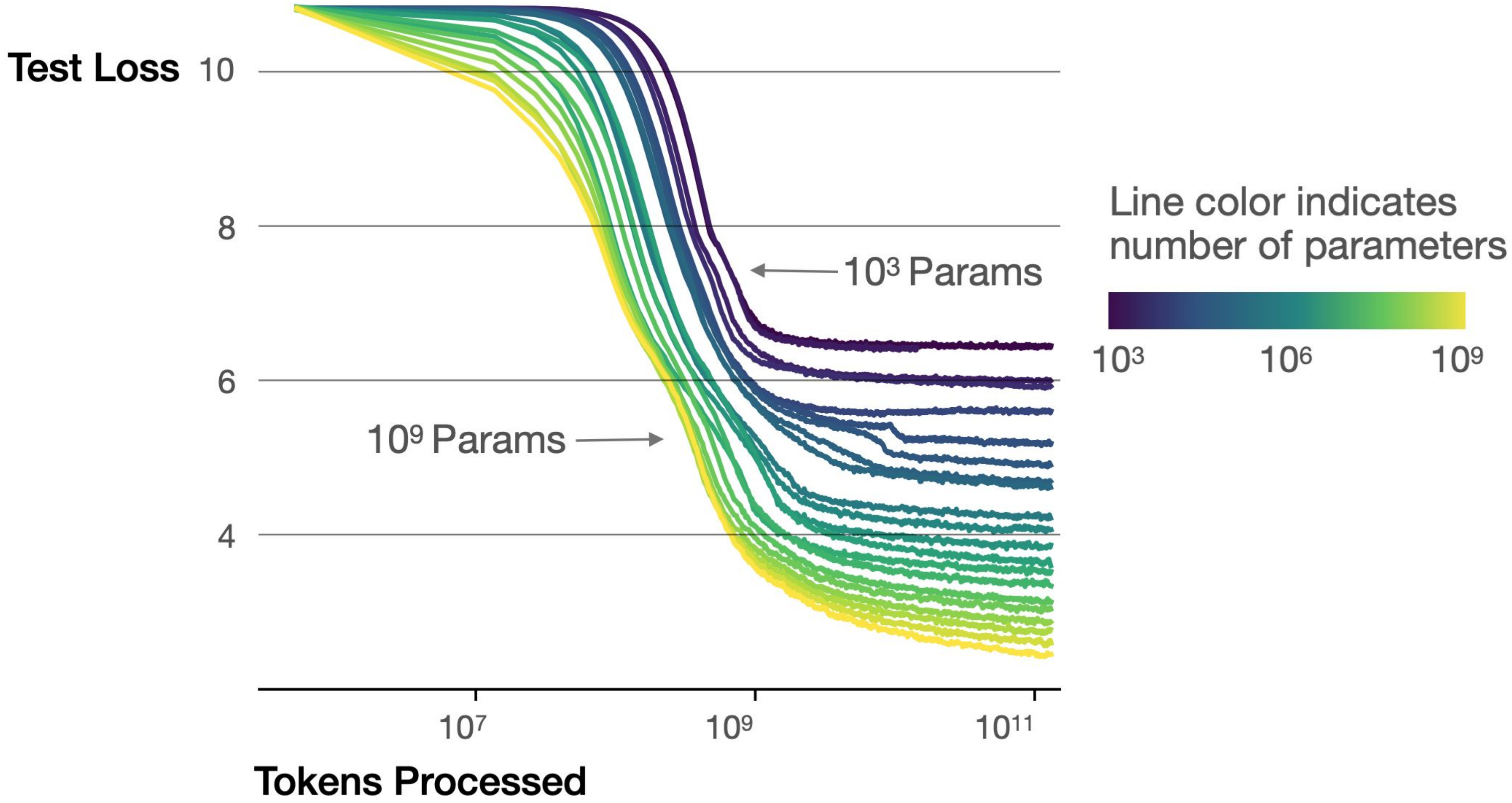
Given a fixed compute budget, what is the optimal model size and dataset size for training?

- Kaplan et al. 2020
- **Chinchilla** (Hoffmann et al. 2022)

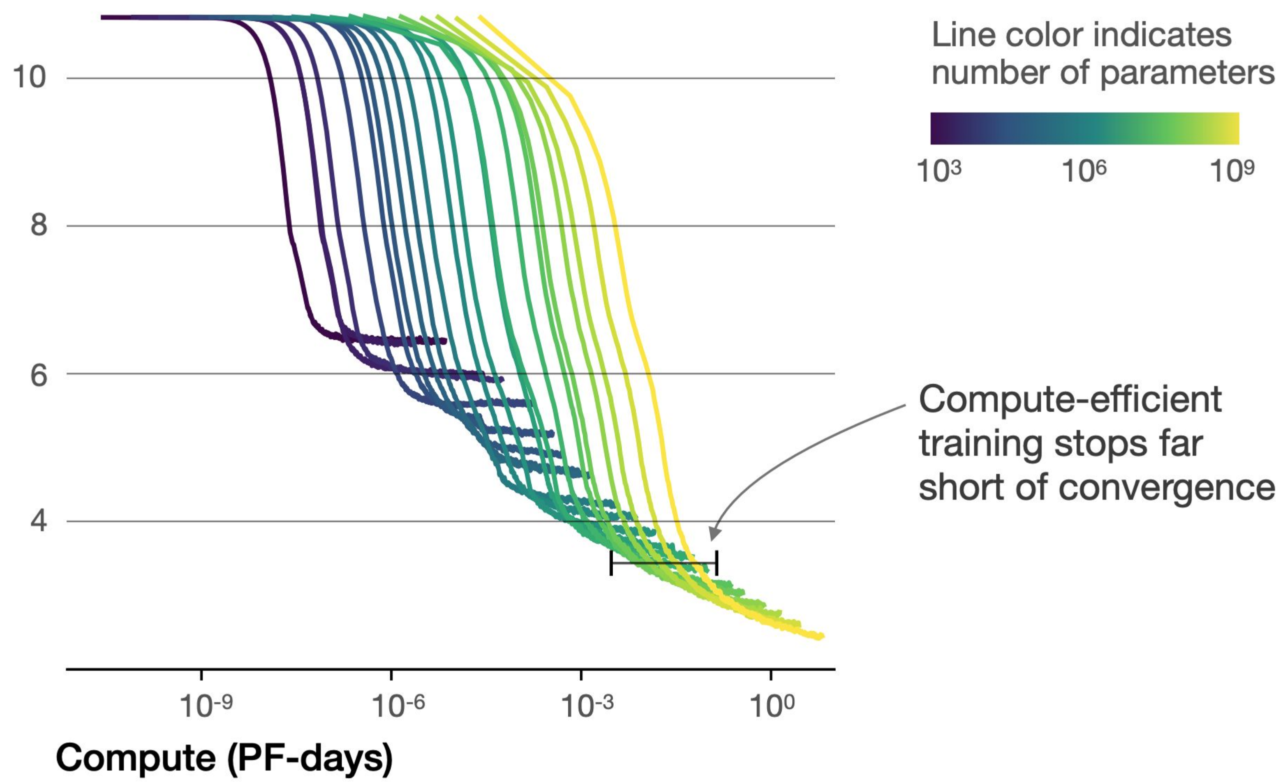
Observations from Kaplan et al., 2020

- Performance depends largely on scale (model size, data size, and compute) and weakly on model architecture (e.g., depth, width)

- Performance improves most when model and dataset size scale together; increasing one while keeping the other fixed results in diminishing returns


Test Loss
10
8
6
4
10^3 Params
10^9 Params
Line color indicates number of parameters
10^3
10^6
10^9
10^7
10^9
10^{11}
Tokens Processed

The optimal model size grows smoothly with the loss target and compute budget



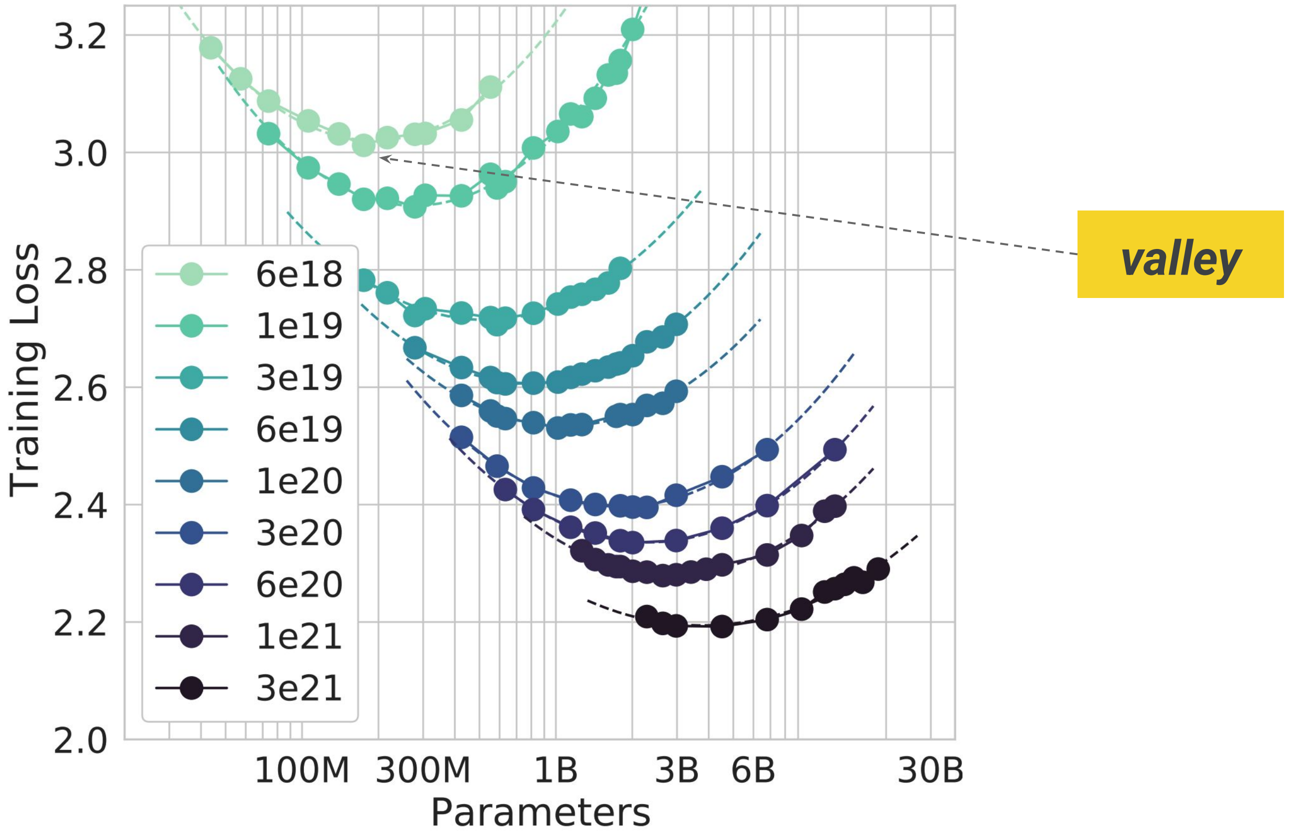
Issues with Kaplan laws

- Used same learning rate schedule for all training runs, regardless of how many training tokens / batches!
- This schedule needs to be adjusted based on the number of training steps; otherwise, it can impair performance
- The resulting “scaling laws” from Kaplan et al., are flawed because of this!

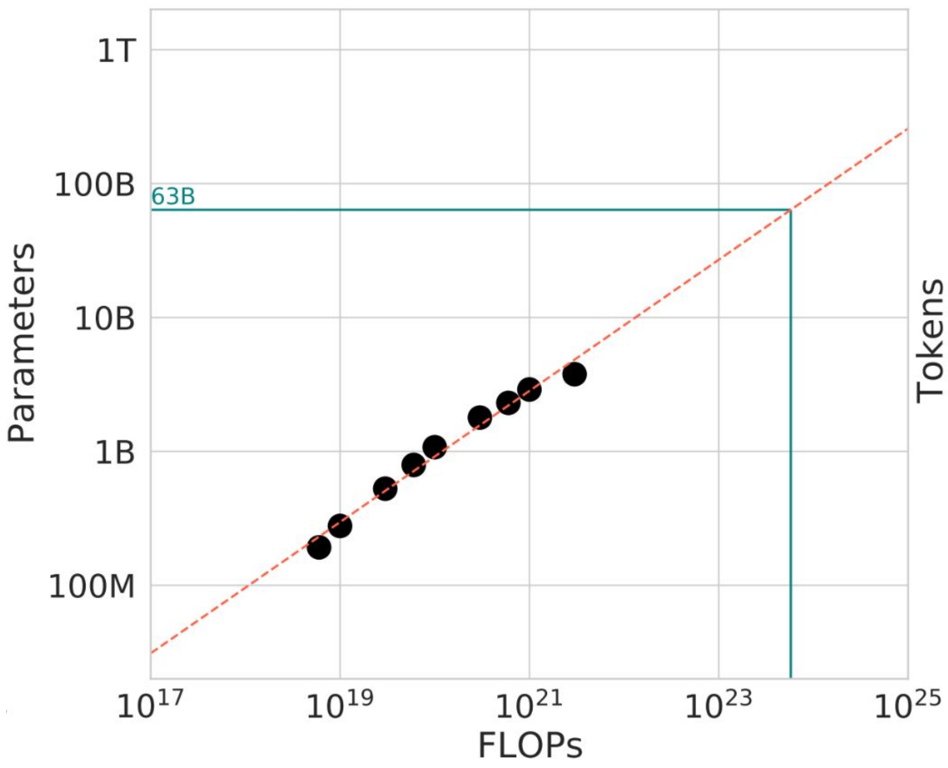
Chinchilla scaling laws

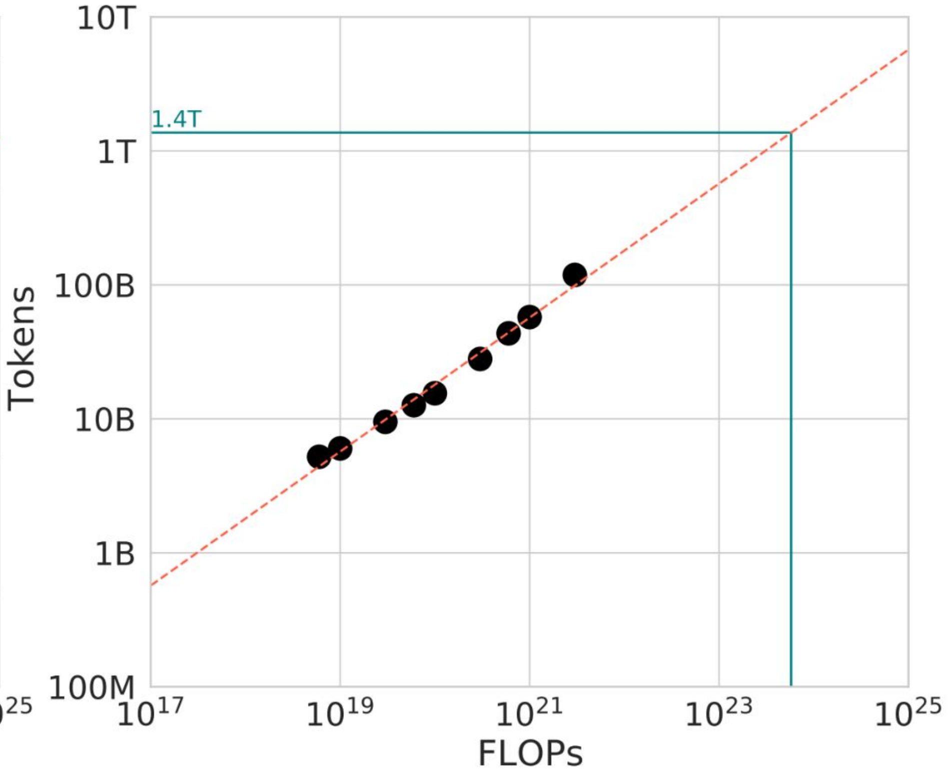
- **Kaplan et al., 2020: prioritize increasing model size over data size**
  - With a 10x compute increase, increase model size by 5x and data size by 2x
  - With a 100x compute increase, model size 25x and data 4x
- **Chinchilla (Hoffmann et al., 2022): increase model and data size at the same rate**
  - With a 10x compute increase, increase both model size and data size by 3.1x
  - With a 100x compute increase, both model and data size 10x

For a given FLOP budget there is an optimal model to train

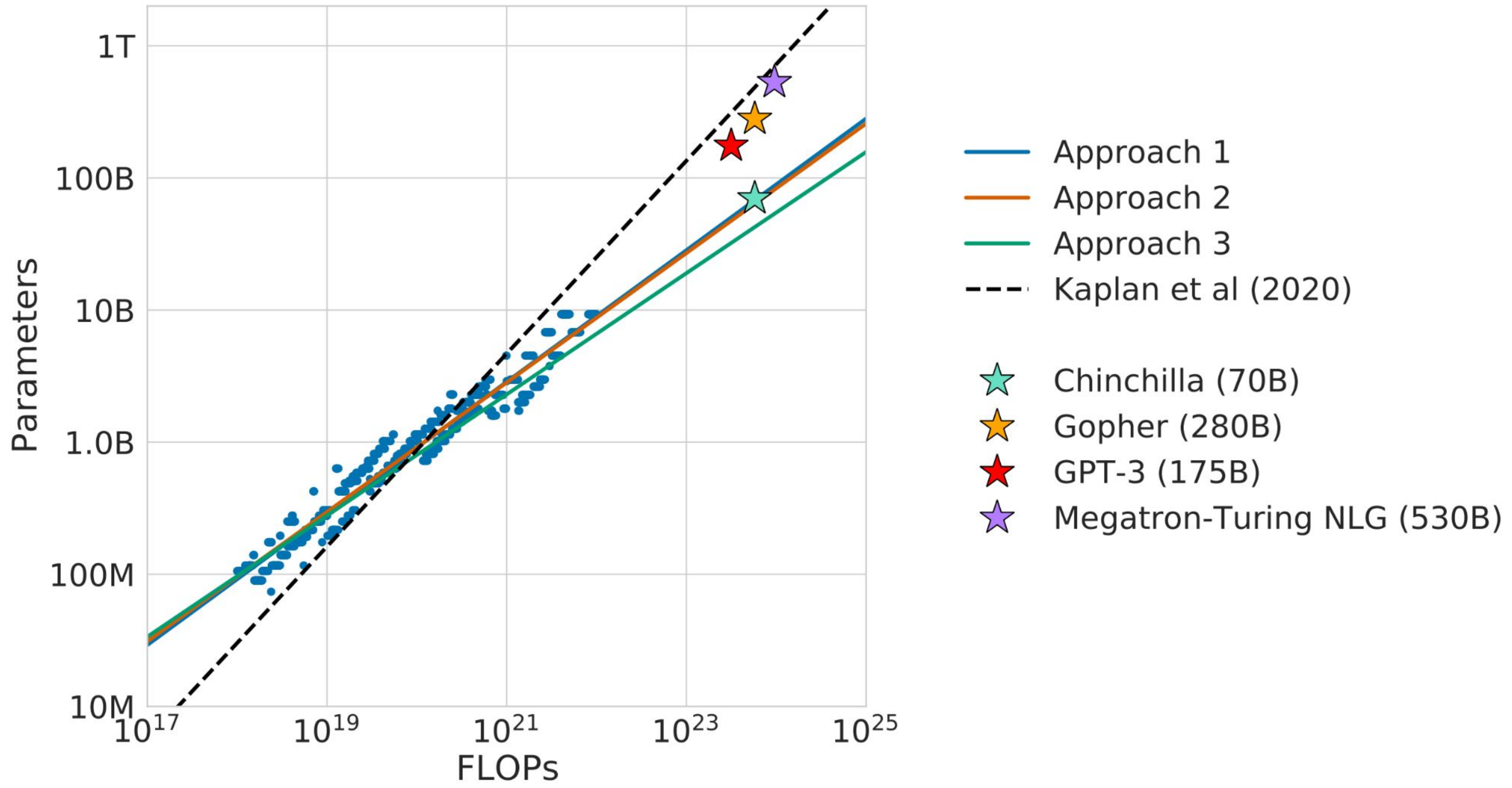


Projecting optimal model size and number of tokens for larger models





Large models should be significantly smaller and trained for much longer than is currently done (2022)



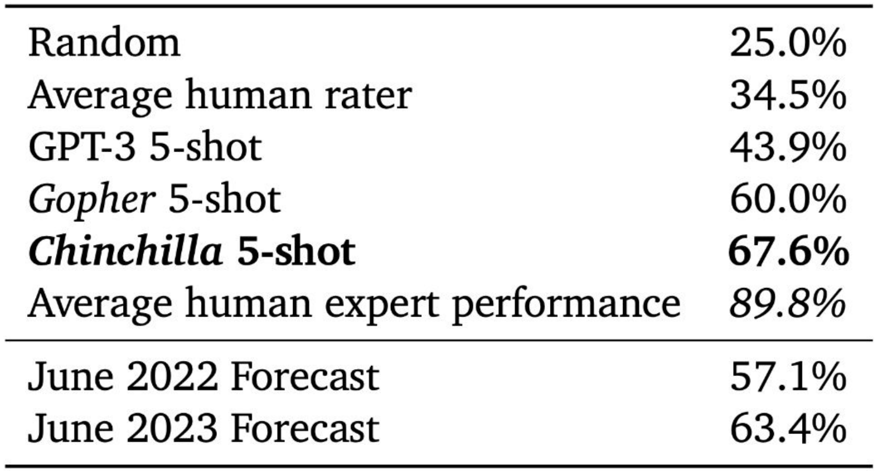
Large models should be significantly smaller and trained for much longer than is currently done (2022)

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

- N – the number of model parameters, *excluding all vocabulary and positional embeddings*
- $C \approx 6NBS$ – an estimate of the total non-embedding training compute, where B is the batch size, and S is the number of training steps (ie parameter updates). We quote numerical values in PF-days, where one PF-day $= 10^{15} \times 24 \times 3600 = 8.64 \times 10^{19}$ floating point operations.

PF: PetaFLOP

Gopher vs. Chinchilla



Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
Gopher 5-shot	60.0%
***Chinchilla* 5-shot**	**67.6%**
Average human expert performance	*89.8%*
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

Table 6 | **Massive Multitask Language Understanding (MMLU).** We report the average 5-shot accuracy over 57 tasks with model and human accuracy comparisons taken from Hendrycks et al. (2020). We also include the average prediction for state of the art accuracy in June 2022/2023 made by 73 competitive human forecasters in Steinhardt (2021).

Chinchilla's loss function

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$

Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws

Nikhil Sardana [1] Jacob Portes [1] Sasha Doubov [1] Jonathan Frankle [1]

401.00448v3 [cs.LG] 14 Apr 2025

Abstract

Large language model (LLM) scaling laws are empirical formulas that estimate changes in model quality as a result of increasing parameter count and training data. However, these formulas, including the popular Deepmind Chinchilla scaling laws, neglect to include the cost of inference. We modify the Chinchilla scaling laws to calculate the optimal LLM parameter count and pre-training data size to train and deploy a model of a given quality and inference demand. We conduct our analysis both in terms of a compute budget and real-world costs and find that LLM researchers expecting reasonably large inference demand (~1B requests) should train models smaller and longer than Chinchilla-optimal. Furthermore, we train 47 models of varying sizes and parameter counts to validate our formula and find that model quality continues to improve as we scale tokens per parameter to extreme ranges (up to 10,000). Finally, we ablate the procedure used to fit the Chinchilla scaling law coefficients and find that developing scaling laws only from data collected at typical token/parameter ratios overestimates the impact of additional tokens at these extreme ranges.

of the model. This volume can be significant; demand for popular models can exceed billions of tokens per day (OpenAI & Pilipiszyn, 2021; Shazeer & Freitas, 2022).

Accounting for both *training and inference*, how does one minimize the cost required to produce and serve a high quality model?

Recent studies have proposed scaling laws, empirical formulas that estimate how changes in model and training data size impact model quality (Kaplan et al., 2020; Hoffmann et al., 2022). Hoffmann et al. (2022) is perhaps the most influential of these works, finding that to scale language models most efficiently, parameters and tokens should grow approximately linearly. The authors applied this scaling law to train a 70B parameter model (dubbed *Chinchilla*) that outperformed much larger and more expensive models such as GPT-3. As a result, many subsequent LLMs have been trained following the Chinchilla scaling laws (Dey et al., 2023; Muennighoff et al., 2023).

However, the Chinchilla scaling laws only account for the computational costs of training. By contrast, the Llama 2 family of models were trained on 2 trillion tokens and the Llama 3 family of models were trained on 15 trillion tokens, which is far more data than the Chinchilla scaling laws would deem "optimal" (Touvron et al., 2023a;b; AI@Meta,

Beyond Chinchilla-optimal

Model	# Params	# Training Tokens	Ratio (tokens / param)
Chinchilla	70B	1.4T	20
Llama 1	7B	1T	142
Llama 2	7B	2T	284
Llama 3	8B	15T	1,875
Sardana et al.	varies	varies	up to 10,000

https://www.jonvet.com/blog/llm-scaling-in-2025

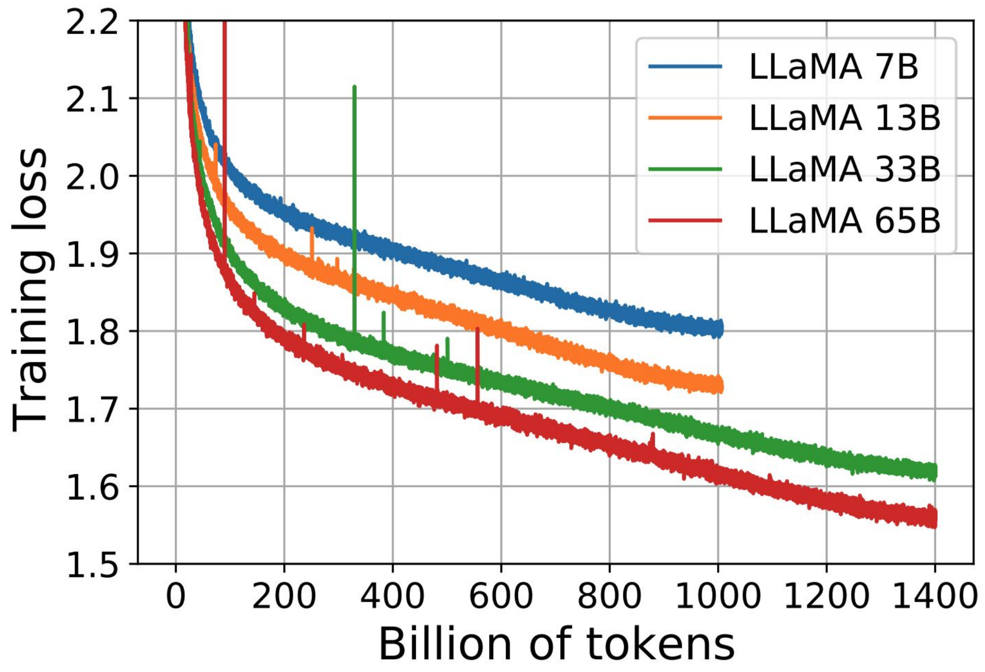


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

Thank you!