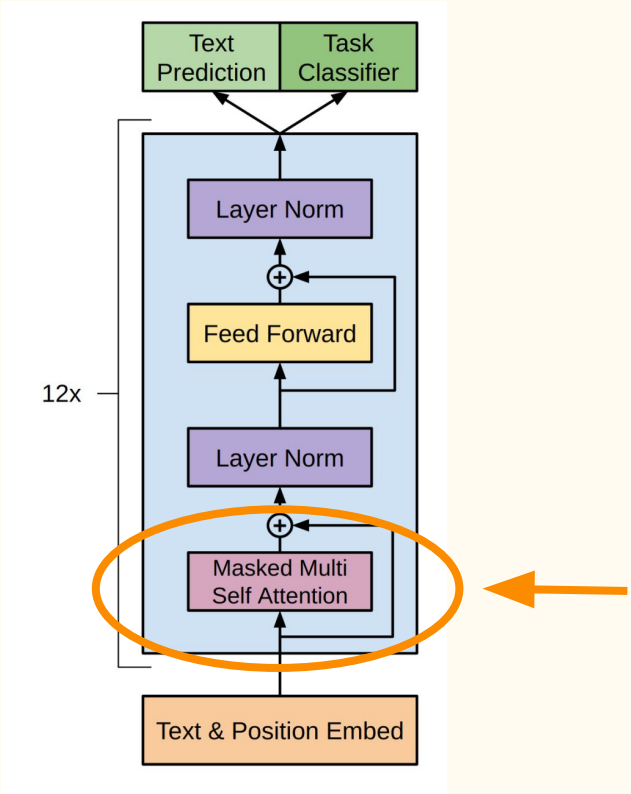
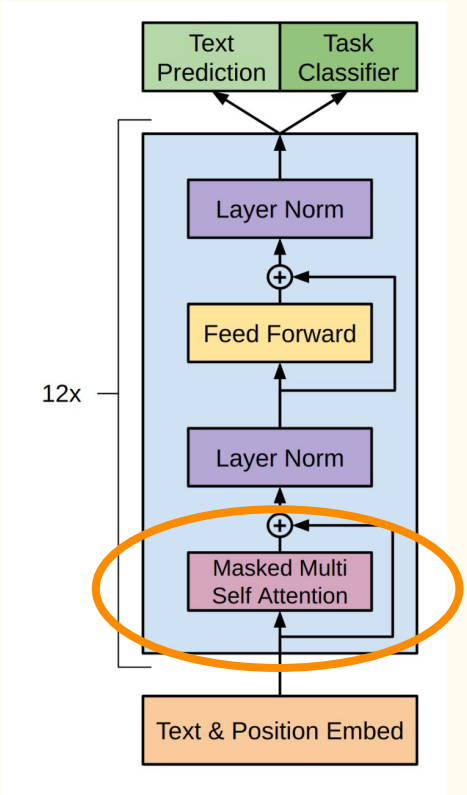
Session 5

Quantizing KV-Cache

Today's focus: Attention

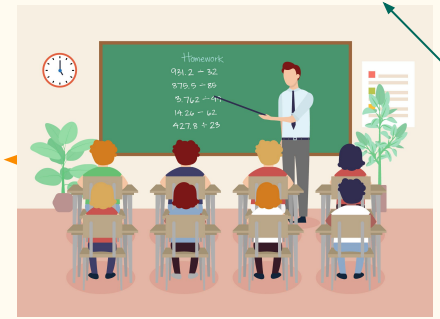


Today's focus: Attention

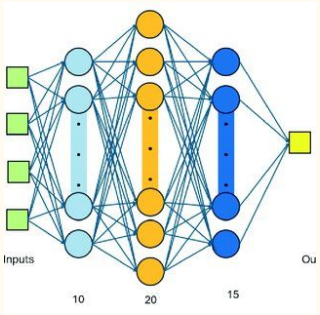


High level intuition: adjusts token embeddings to account for context clues from earlier text.

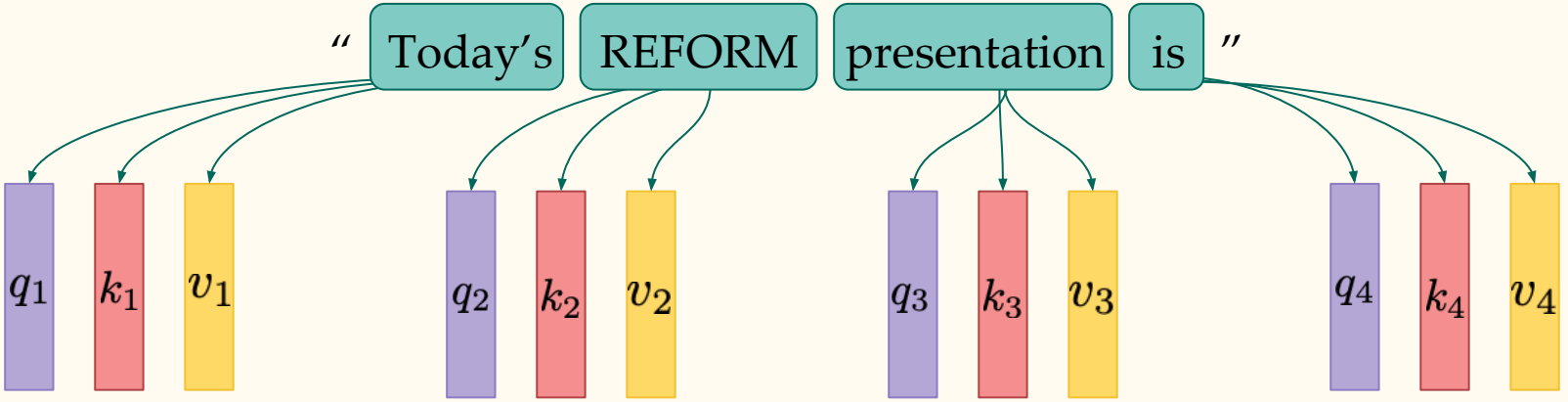
"…school… learning"



"…REFORM…machine learning"



Background: Attention



All 1xd

$$q_i = x_i W_q \qquad k_i = x_i W_k \qquad v_i = x_i W_v$$

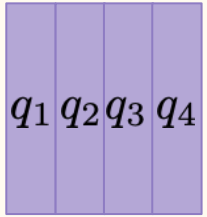
Query: what words might be relevant to this token?

Key: Should a future token attend to this token?

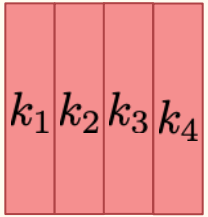
Value: Direction to shift the embedding if future token attends to this token.

Background: Attention

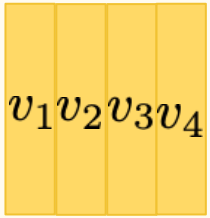
Q (n x d)



K (n x d)



V (n x d)



Query: what words might be relevant to this token?

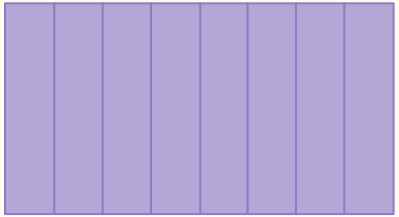
Key: Should a future token attend to this token?

Value: Direction to shift the embedding if future token attends to this token.

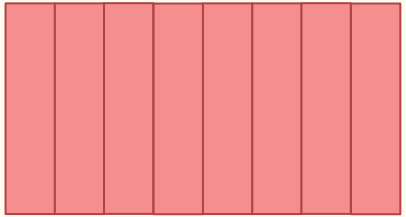
Background: Attention

n >> d

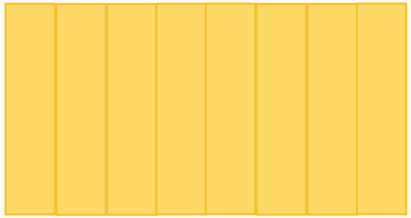
Q (n x d)



K (n x d)



V (n x d)



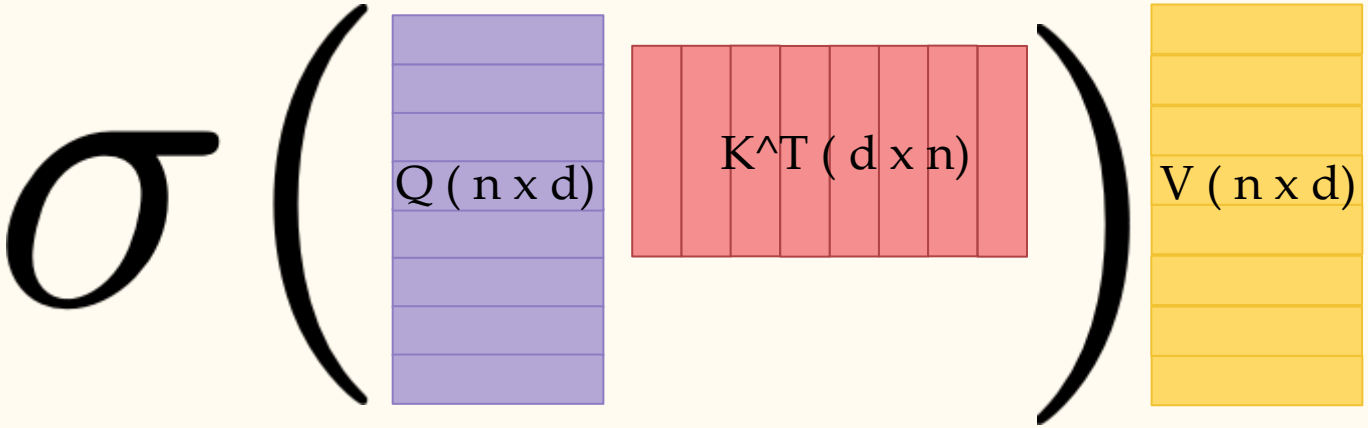
Query: what words might be relevant to this token?

Key: Should a future token attend to this token?

Value: Direction to shift the embedding if future token attends to this token.

Background: Attention

Attention: $\sigma(QK^\top)V$

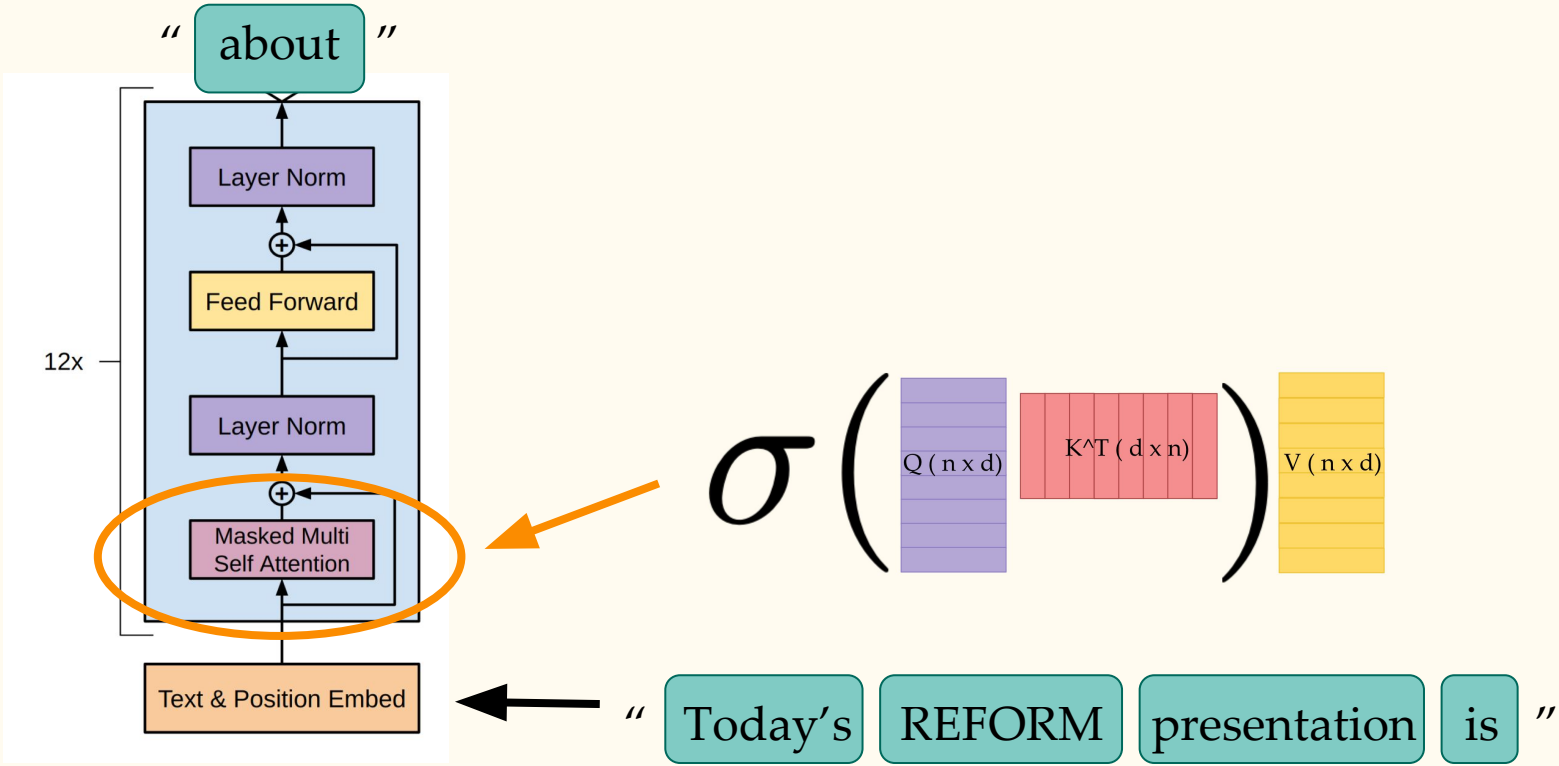


Query: what words might be relevant to this token?

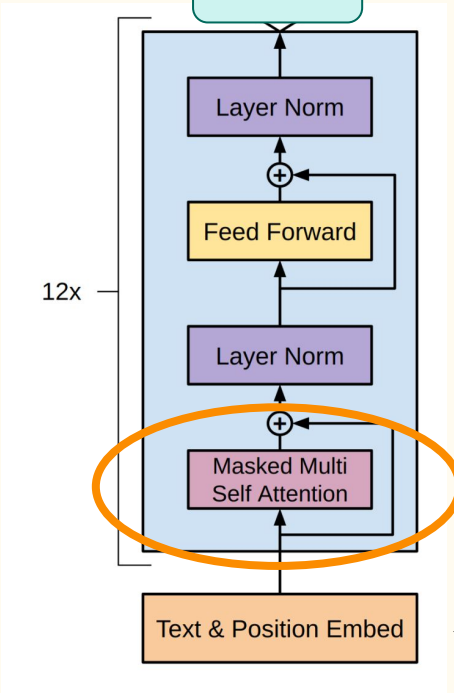
Key: Should a future token attend to this token?

Value: Direction to shift the embedding if future token attends to this token.

Background: Attention



Background: Next-token generation

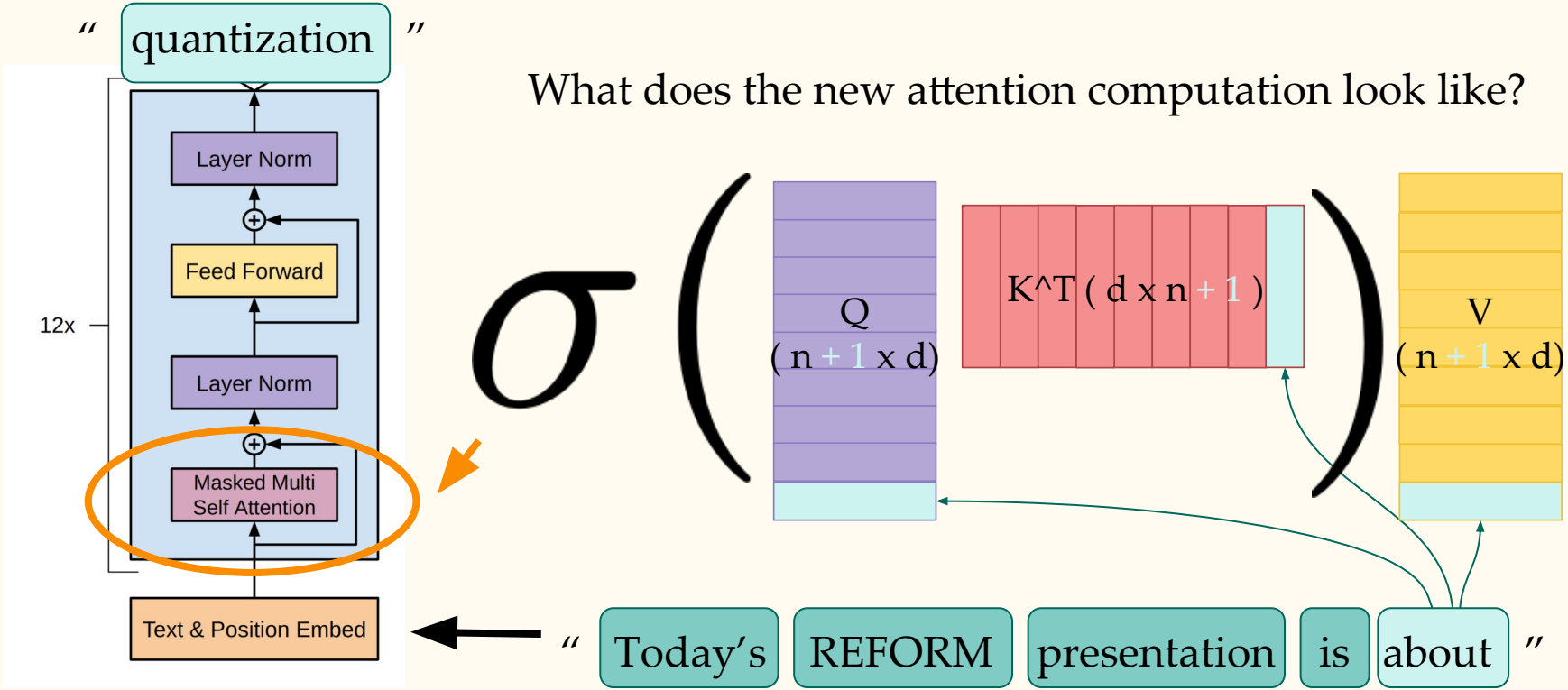


“ about ”

Add new token to the context and repeat!

“ Today’s REFORM presentation is ”

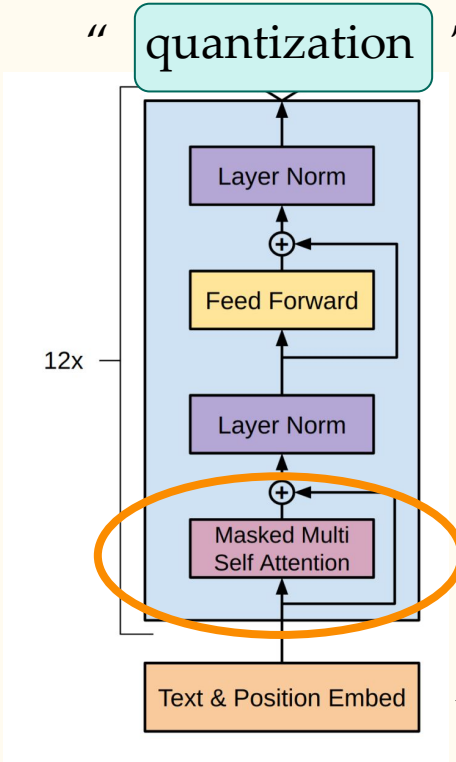
Background: Next-token generation

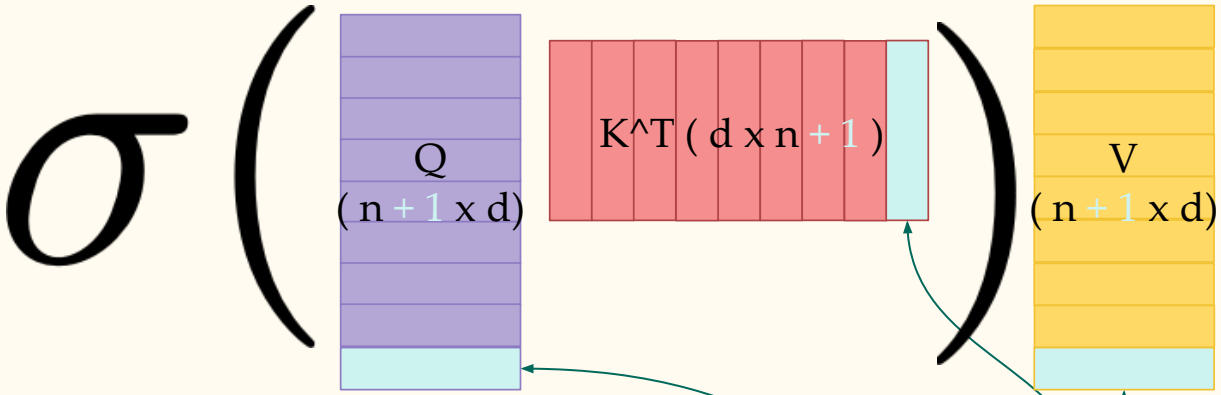


Background: Next-token generation

Much of these matrices overlap with last step.

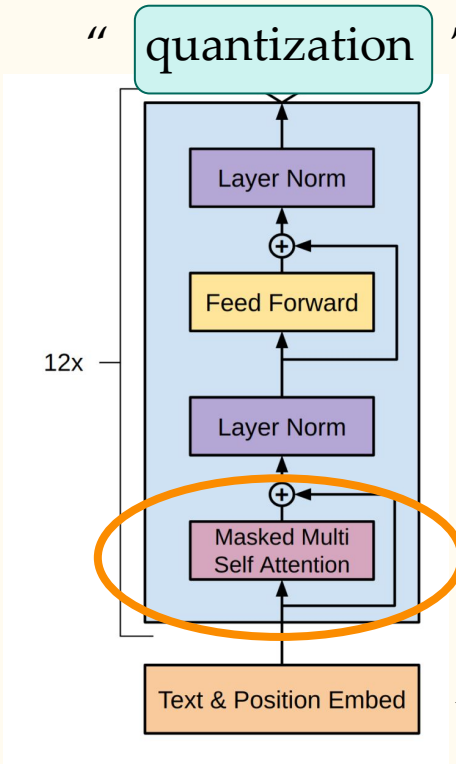
Naive: recompute $K = x_{1:n+1} W_k$, etc.





“ Today’s REFORM presentation is about ”

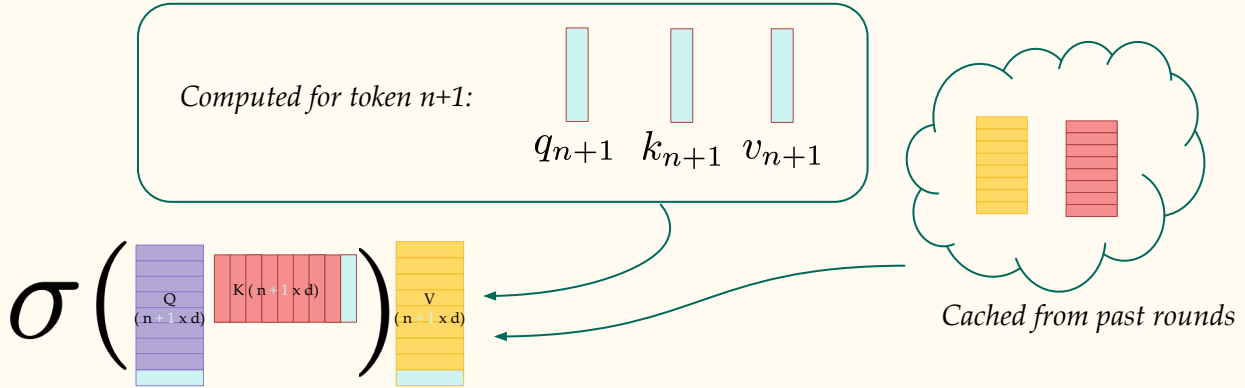
Background: Next-token generation



Much of these matrices overlap with last step.

Naive: recompute $K = x_{1:n+1}W_k$, etc.

Cached: Store K, V from previous step, append $k_{n+1} = x_{n+1}W_k$



" Today's REFORM presentation is about "

Next Token Generation: Details!

Attention scores:

$$a_t = \text{Attention}(q_t, k_{1:t}, v_{1:t}) = \text{softmax}(q_t k_{1:t}^\top) v_{1:t}$$

Next-token probabilities:

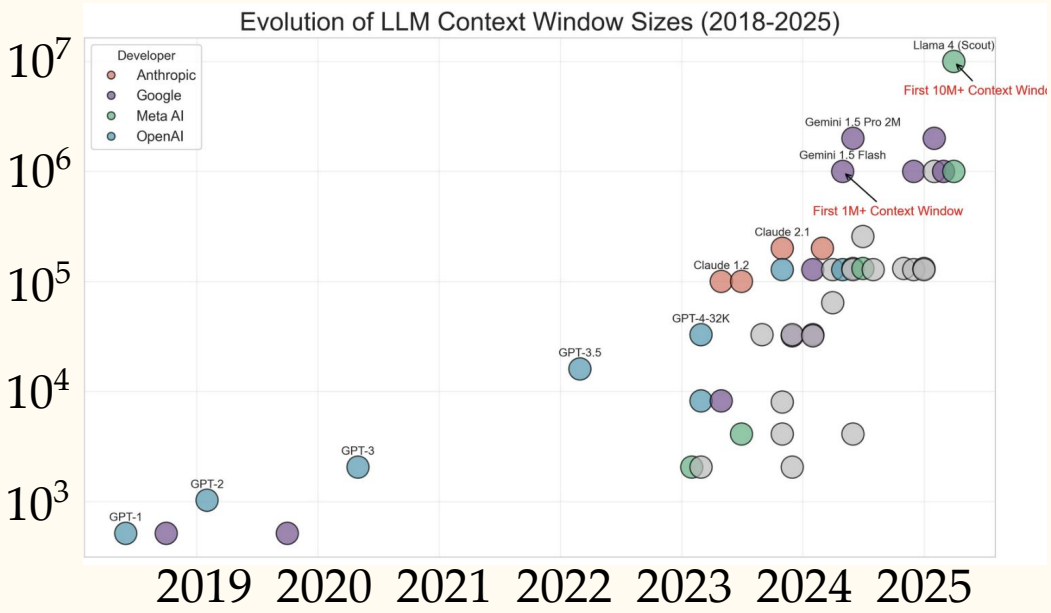
$$\Pr(x_{t+1} = w | x_{1:t}) \propto e^{a_t W_O W_U e_w}$$

The *distribution of n-th token* depends on

a single query vector but *n key vectors* and *n value vectors*

KV-cache

Context Lengths n are Growing



How do we reduce the memory footprint with large n?

Compressing KV-Cache

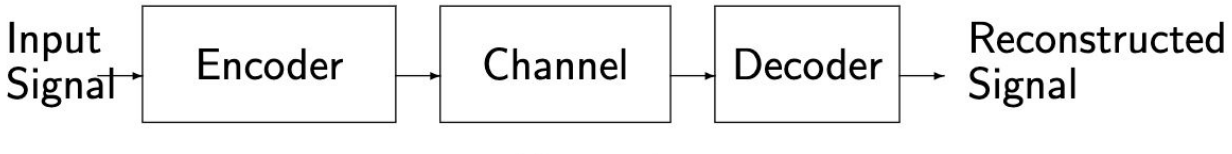


Figure 1

Compressing KV-Cache

Goal: Find a *compression function* f

$$f : \mathbb{R}^d \to \{0,1\}^B$$

such that for "most" keys k and values v, it holds that

$$f(k) \approx k \quad \text{and} \quad f(v) \approx v$$

Examples of Compression for Images

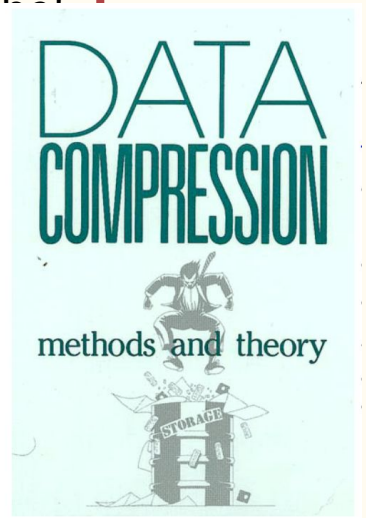
Compressing KV-Cache

Goal: Given distributions D_k and D_v, find *function* f

$$f : \mathbb{R}^d \to \{0,1\}^B$$

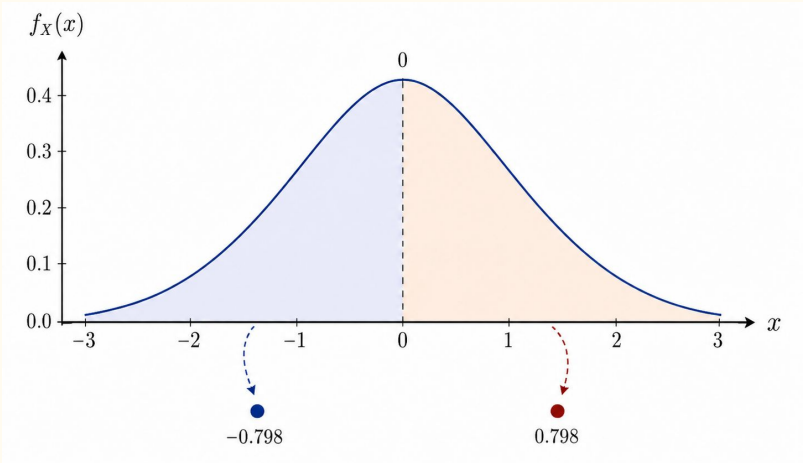
that minimizes ~~for "most" keys k and values v, it holds t~~

$$\mathbb{E}_{k\sim\mathcal{D}_k} \|f(k) - k\|_2 \quad \text{and} \quad \mathbb{E}_{v\sim\mathcal{D}_v} \|f(v) - v\|$$

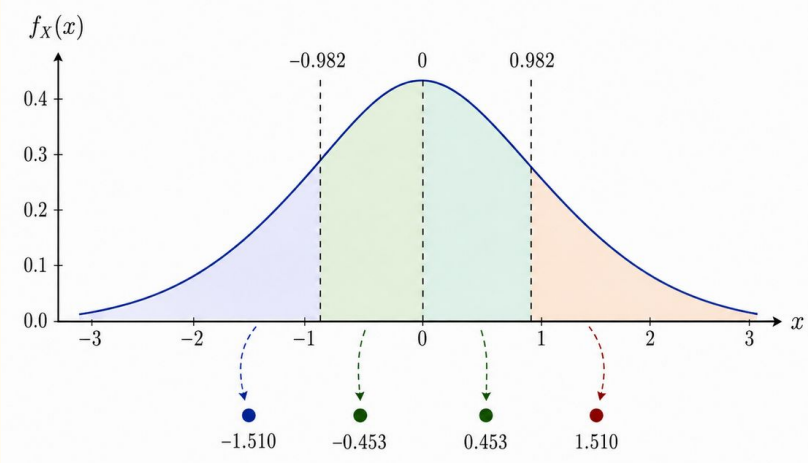


Compressing KV-Cache: Example

Suppose $x \sim N(0,1)$, then:

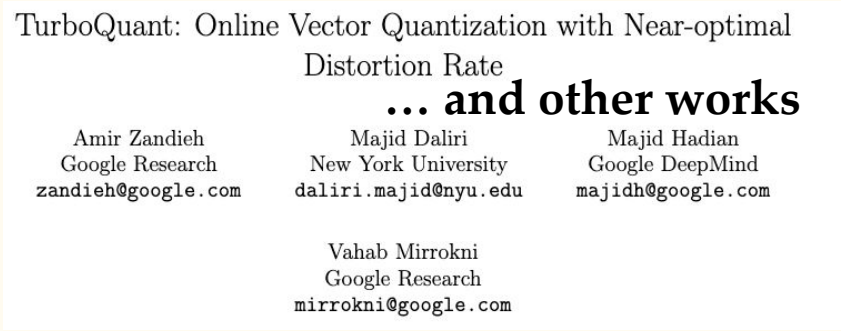


Optimal quantization for B=1



Optimal quantization for B=2

Recent Quantization Ideas



TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate

… and other works

Amir Zandieh
Google Research
zandieh@google.com

Majid Daliri
New York University
daliri.majid@nyu.edu

Majid Hadian
Google DeepMind
majidh@google.com

Vahab Mirrokni
Google Research
mirrokni@google.com

Hidden dimension $d \approx 4096$

First, convert a vector $v = (v_1, v_2, v_3, \ldots, v_d)$ to $v = \{\|v\|, (v_i/\|v\| : i)_{1 \le i \le n}\}$

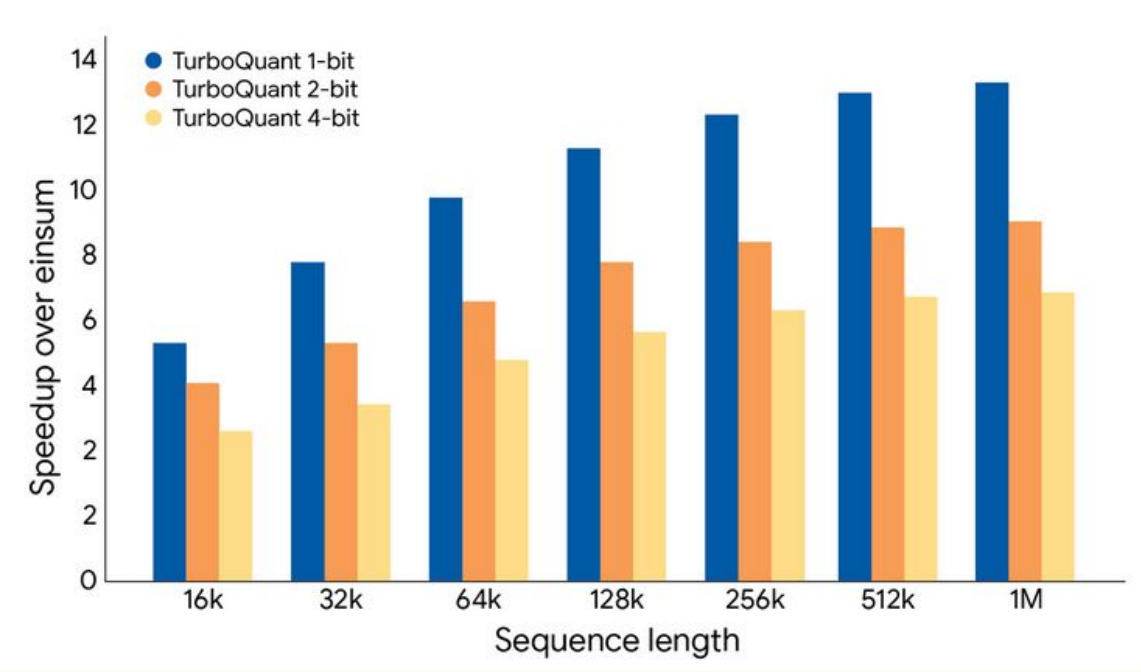
Then, quantize the *magnitude* and the *direction* separately

To quantize the *direction*, apply a *random rotation* Π

> Regardless of $\underline{v}$'s distribution, $\Pi\ \underline{v}$ has *uniform distribution on sphere*

Use "optimal function" to quantize $\underline{v}$

Results: Computational Speedup



Results: Performance Degradation

Method	KV Size	SingleQA	MultiQA	Summarization	Few shot	Synthetic	Code	Average
				Llama-3.1-8B-Instruct				
Full Cache	16	45.29	45.16	26.55	68.38	59.54	46.28	50.06
KIVI	3	43.38	37.99	27.16	68.38	59.50	44.68	48.50
KIVI	5	45.04	45.70	26.47	68.57	59.55	46.41	50.16
PolarQuant	3.9	45.18	44.48	26.23	68.25	60.07	45.24	49.78
TurboQuant (ours)	2.5	44.16	44.96	24.80	68.01	59.65	45.76	49.44
TurboQuant (ours)	3.5	45.01	45.31	26.00	68.63	59.95	46.17	50.06
				Ministral-7B-Instruct				
Full Cache	16	47.53	49.06	26.09	66.83	53.50	47.90	49.89
TurboQuant (ours)	2.5	48.38	49.22	24.91	66.69	53.17	46.83	49.62

Table 1: LongBench-V1 [10] results of various KV cache compression methods on `Llama-3.1-8B-Instruct`.

Future Directions

Is L2 norm the right objective? Which *other properties to preserve*?

Should the quantization scheme *adapt* to the model's weights?

Can we combine quantization schemes with *error correction*?