

<https://introml.mit.edu/>

6.390 Intro to Machine Learning

Lecture 9: Transformers

Shen Shen

April 11, 2025

11am, Room 10-250

([interactive slides](#) support animated walk-throughs of transformers and attention mechanisms.)

Outline

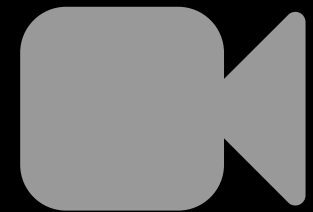
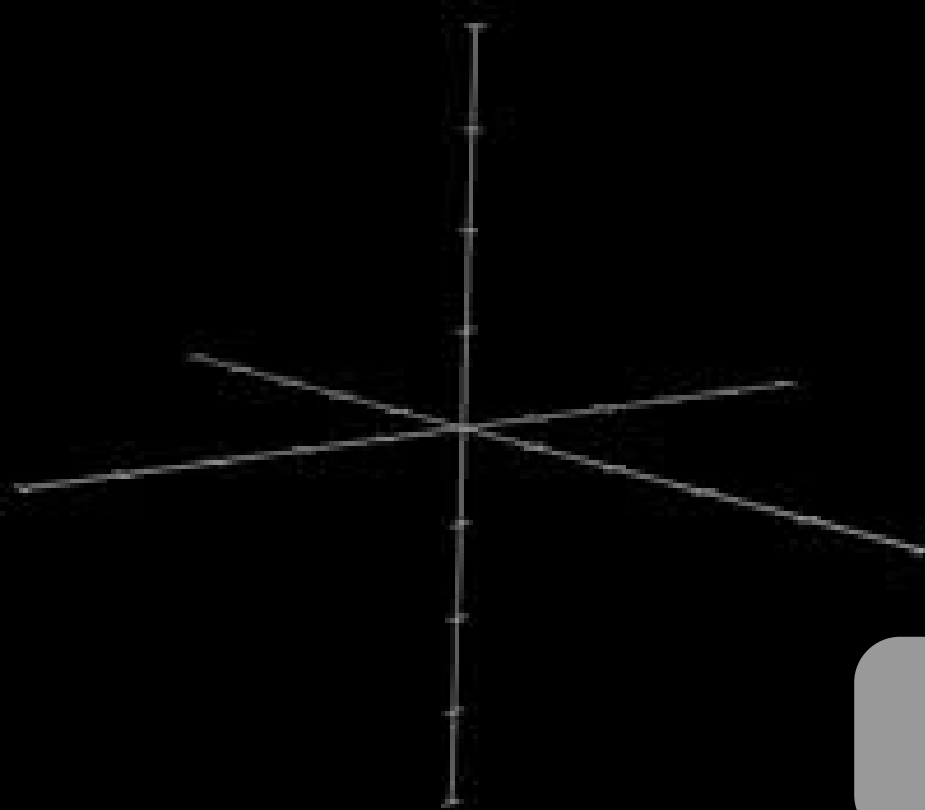
- Recap, embedding and representation
- Transformers high-level intuition
- Transformers architecture overview
- (query, key, value) and self-attention
 - matrix form
- Multi-head Attention
- (Applications)

embedding

Words



Vectors



[video edited from [3b1b](#)]

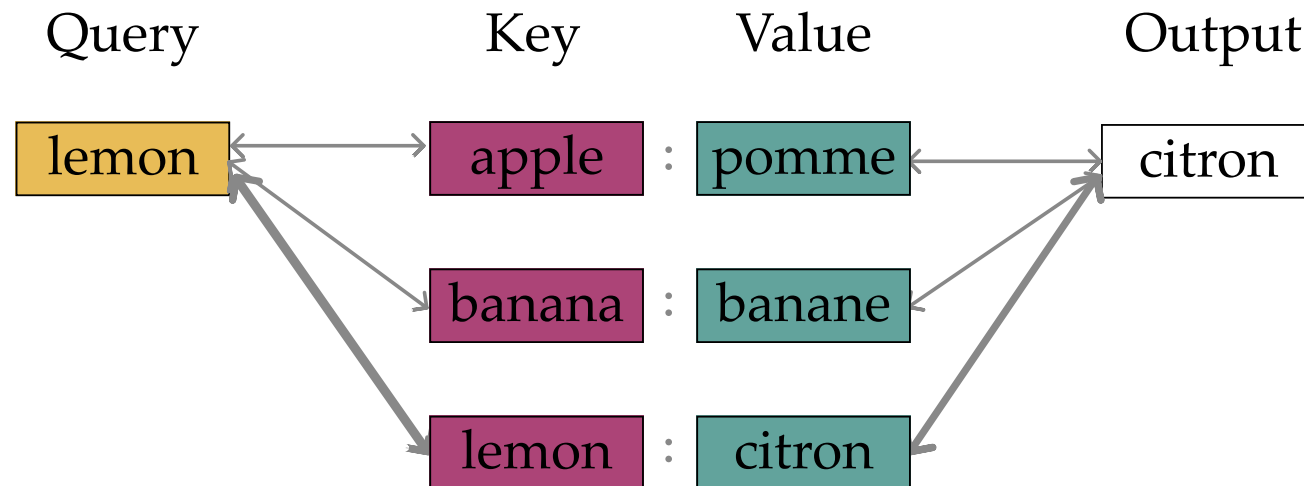
Good embeddings enable vector arithmetic.

```
dict_en2fr = {  
    "apple" : "pomme",  
    "banana" : "banane",  
    "lemon" : "citron"}
```

Key		Value
apple	:	pomme
banana	:	banane
lemon	:	citron

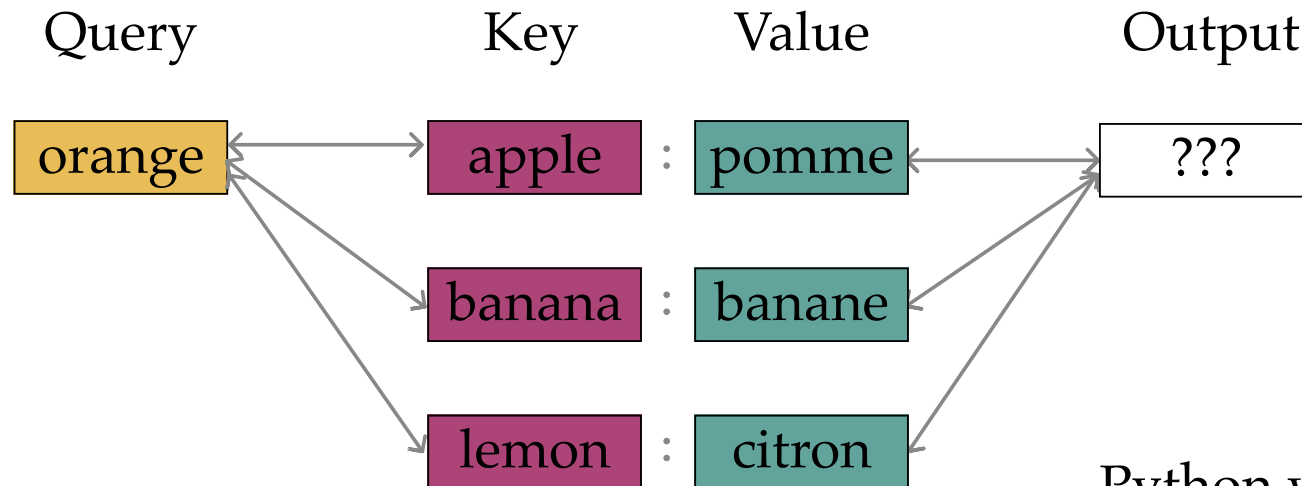
A query comes:

```
dict_en2fr = {  
    "apple" : "pomme",  
    "banana" : "banane",  
    "lemon" : "citron"}  
  
query = "lemon"  
output = dict_en2fr[query]
```



What if:

```
dict_en2fr = {  
    "apple" : "pomme",  
    "banana" : "banane",  
    "lemon" : "citron"}  
  
query = "orange"  
output = dict_en2fr[query]
```

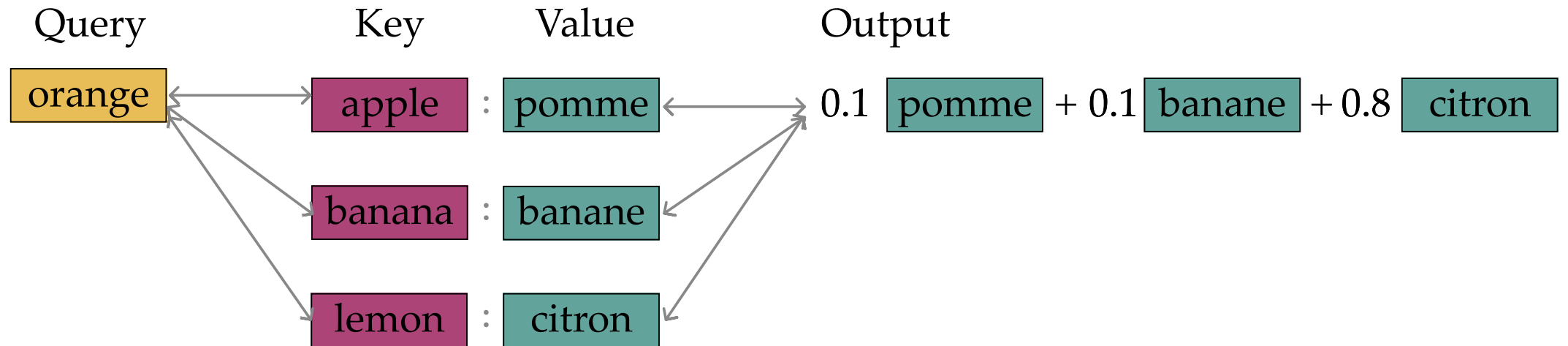


Python would complain. 🤖

What if:

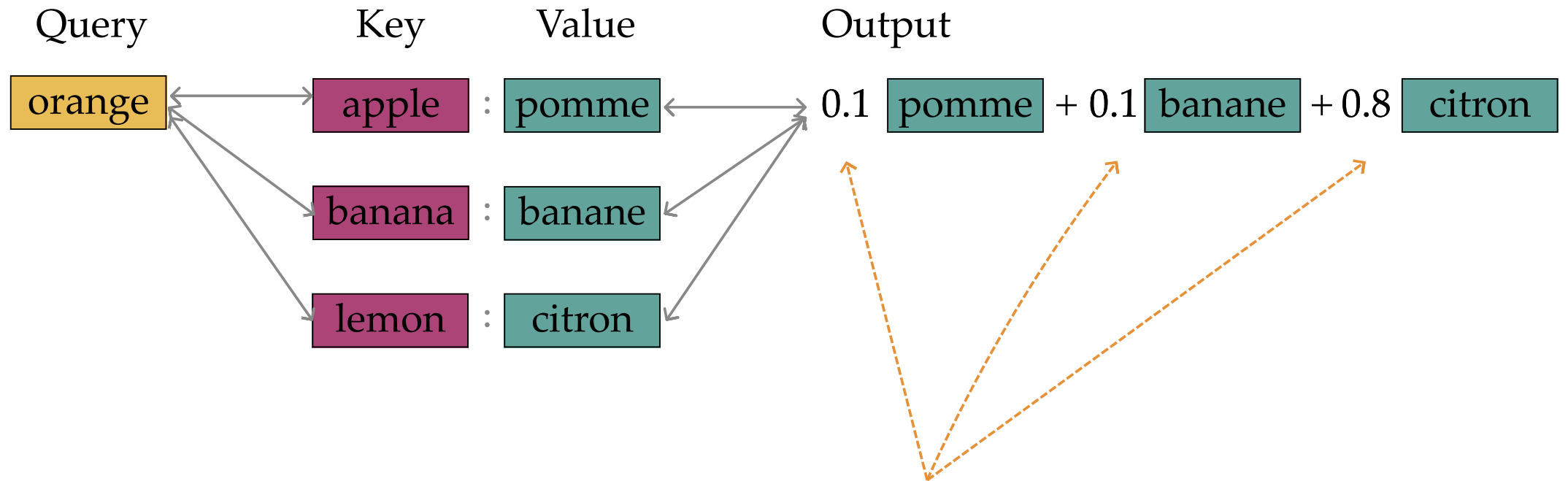
```
dict_en2fr = {  
    "apple" : "pomme",  
    "banana" : "banane",  
    "lemon" : "citron"}  
  
query = "orange"  
output = dict_en2fr[query]
```

But we may agree with this intuition:

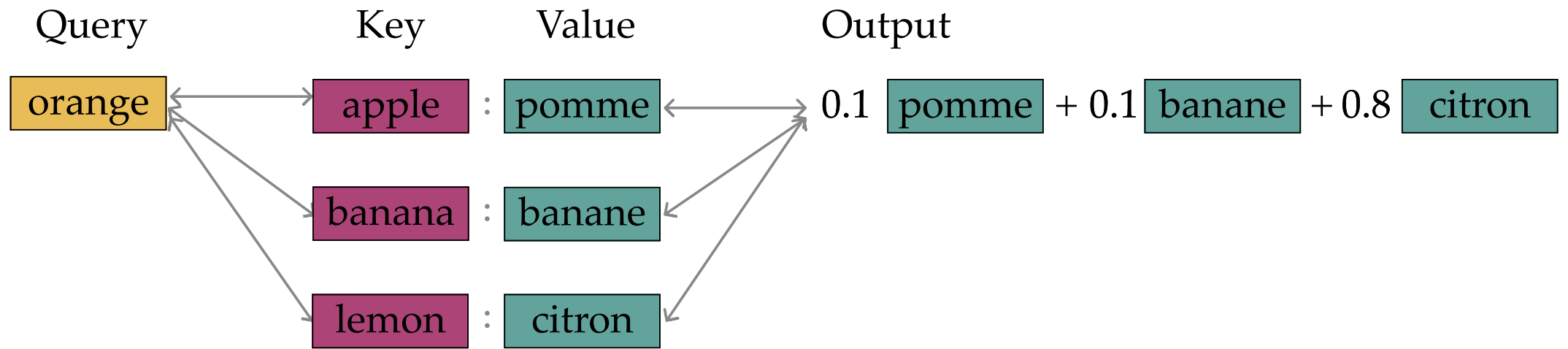


Now, if we are to formalize this idea, we need:

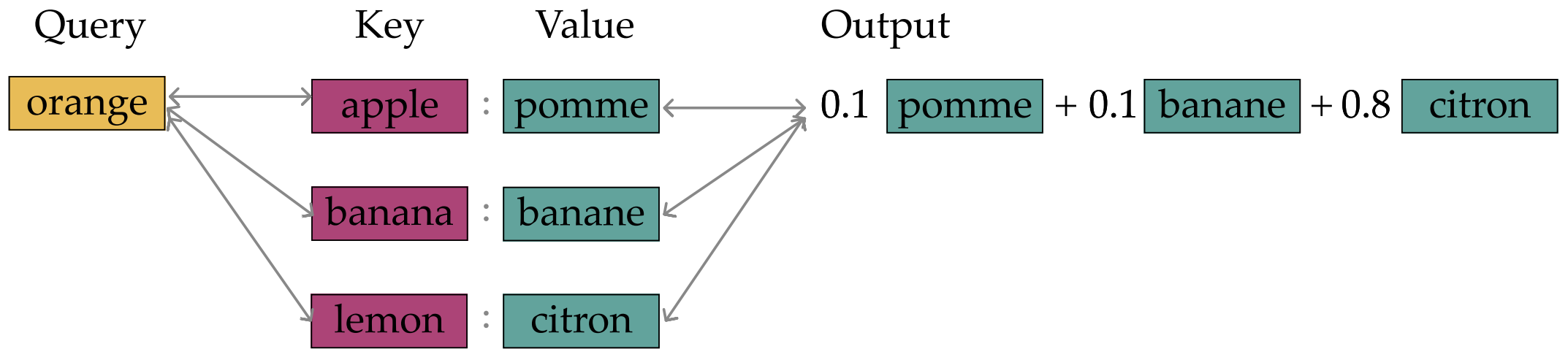
1. *learn* to get to these "good" (query, key, value) embeddings.



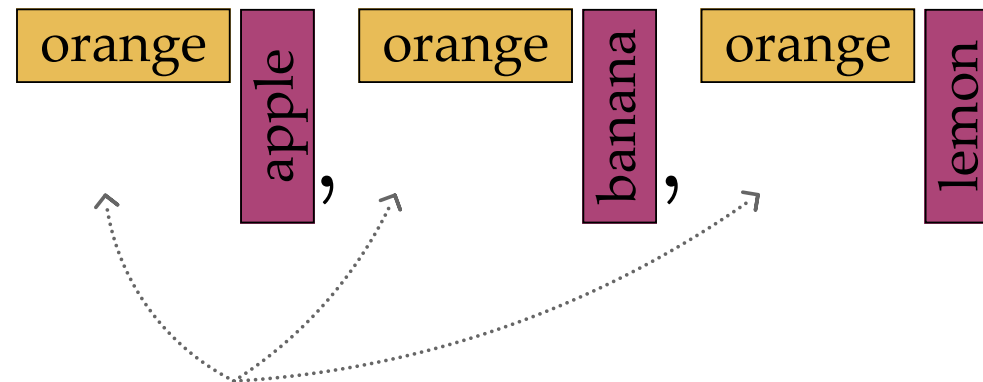
2. *calculate* this sort of percentages



very roughly, with good embeddings, getting the percentages can be easy:

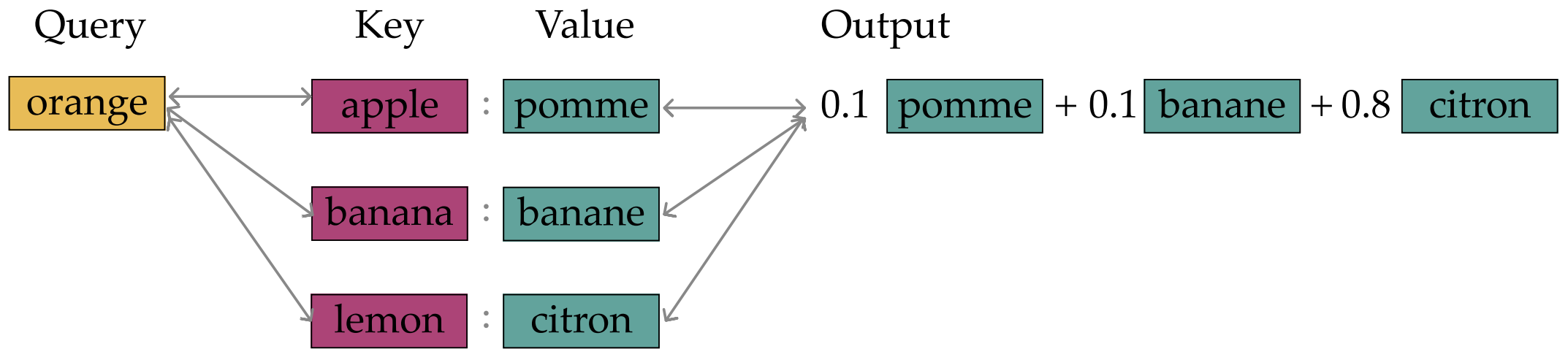


very roughly, with good embeddings, getting the percentages can be easy:



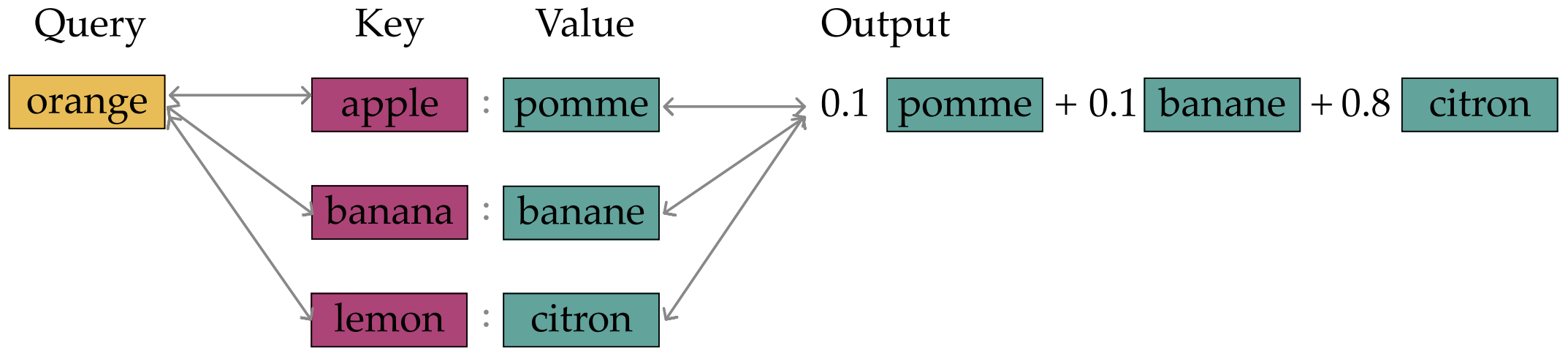
query compared with keys \rightarrow dot-product similarity

what about percentages?



- query compared with keys → percentages

$$\text{softmax} \left(\begin{array}{c} \text{orange} \\ \text{apple} \end{array}, \begin{array}{c} \text{orange} \\ \text{banana} \end{array}, \begin{array}{c} \text{orange} \\ \text{lemon} \end{array} \right) = \begin{bmatrix} 0.1 & 0.1 & 0.8 \end{bmatrix}$$



- query compared with keys → percentages

$$\text{softmax} \left(\begin{array}{c} \text{orange} \\ \text{apple} \end{array}, \begin{array}{c} \text{orange} \\ \text{banana} \end{array}, \begin{array}{c} \text{orange} \\ \text{lemon} \end{array} \right) = [0.1 \ 0.1 \ 0.8]$$

- combine values using these percentages as output

$$0.1 \text{ pomme} + 0.1 \text{ banane} + 0.8 \text{ citron}$$

(very roughly, the attention mechanism does just this "reasonable merging")

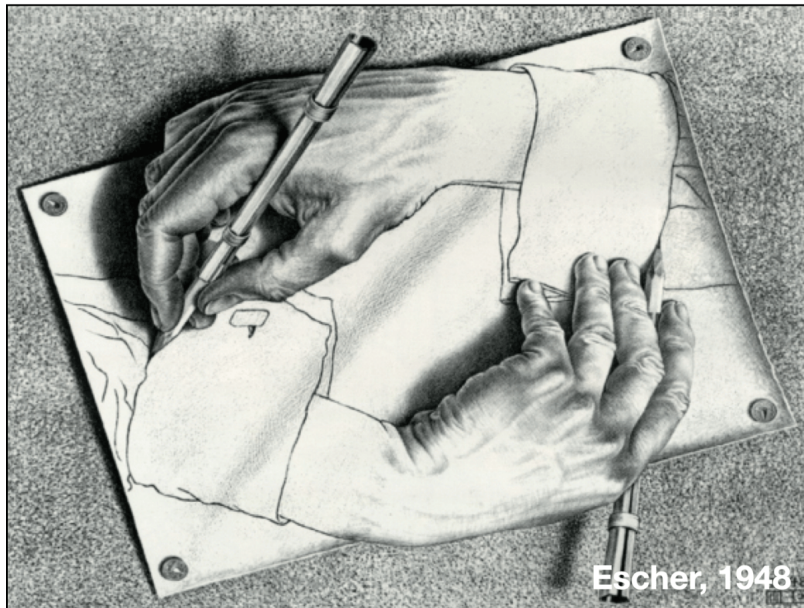
Outline

- Recap, embedding and representation
- Transformers high-level intuition
- Transformers architecture overview
- (query, key, value) and self-attention
 - matrix form
- Multi-head Attention
- (Applications)

Image Classification on ImageNet



Large Language Models (LLMs) are trained in a self-supervised way



- Scrape the internet for unlabeled plain texts.
- Cook up “labels” (prediction targets) from the unlabeled texts.
- Convert “unsupervised” problem into “supervised” setup.

"To date, the cleverest thinker of all time was Issac. "



feature

⋮

To date, the

To date, the cleverest

To date, the cleverest thinker

⋮

To date, the cleverest thinker of all time was

label

⋮

cleverest

thinker

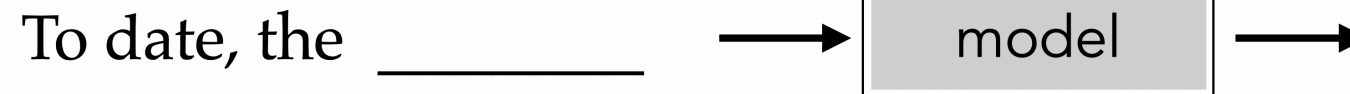
was

⋮

Issac

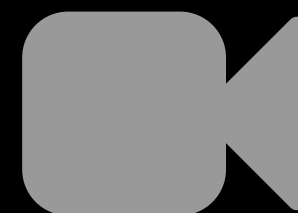
Auto-regressive

e.g., train to predict the next-word

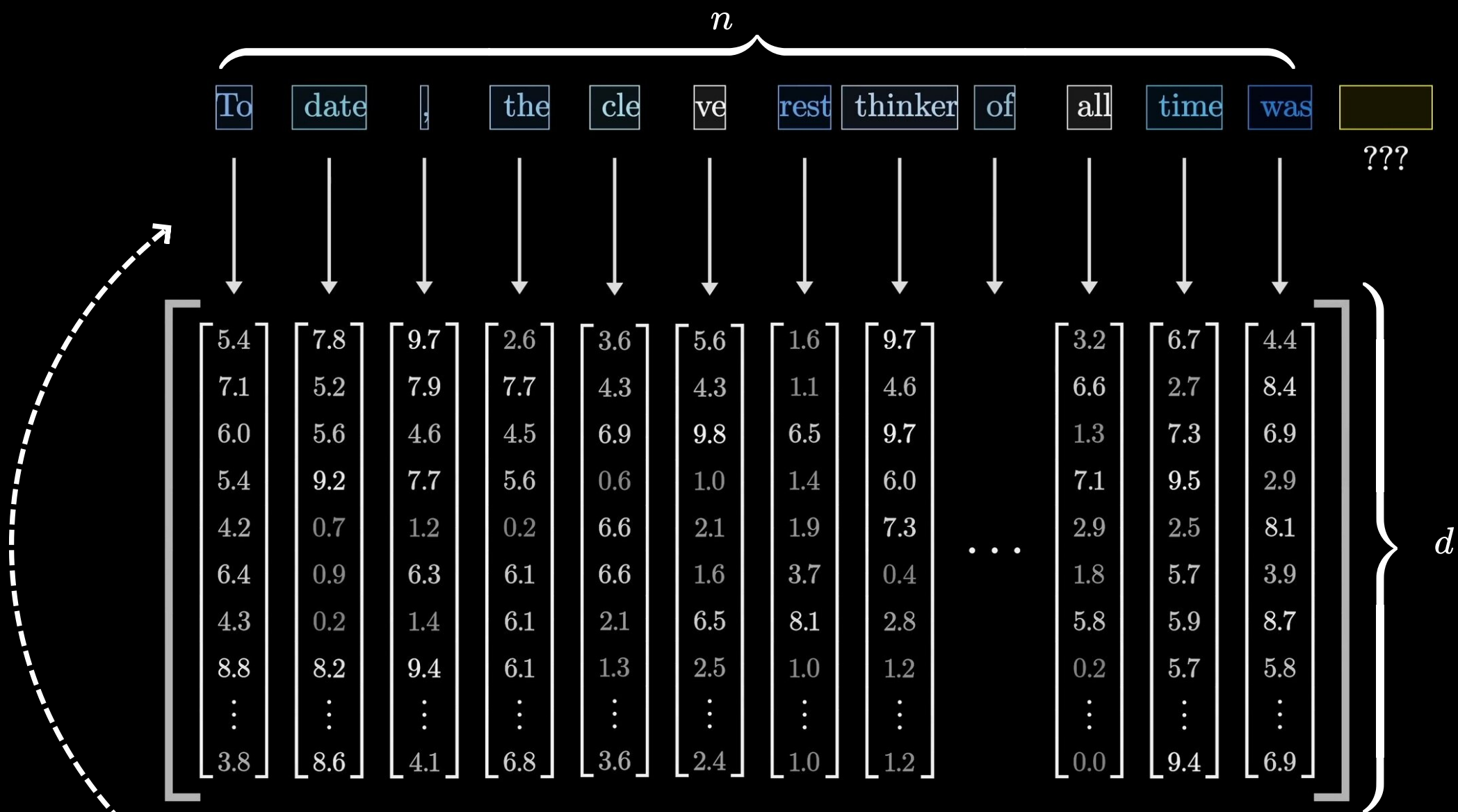


How to train? The same recipe:

- model has some learnable weights
- multi-class classification

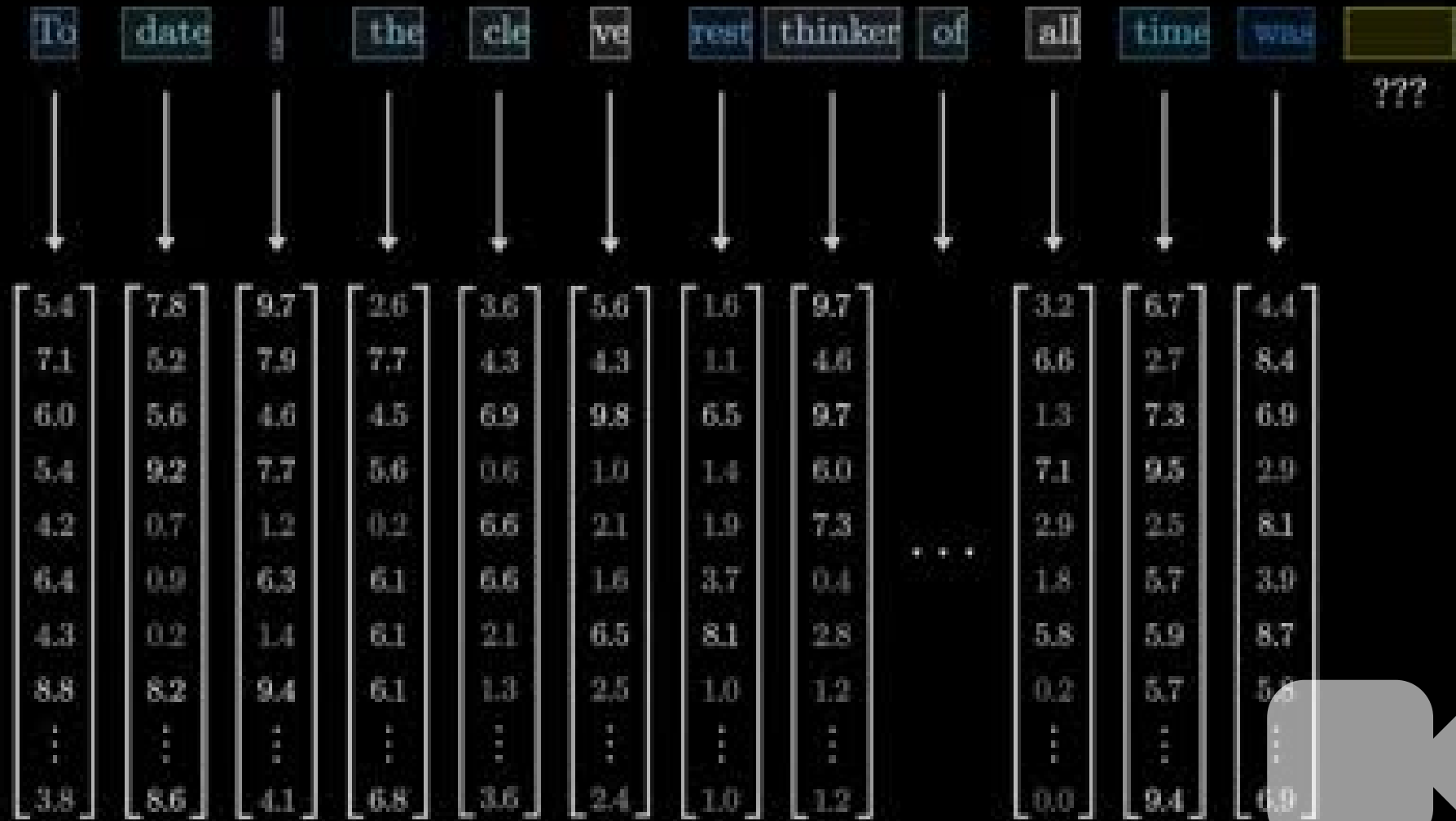


[video edited from [3b1b](#)]



input embedding (e.g. via a fixed encoder)

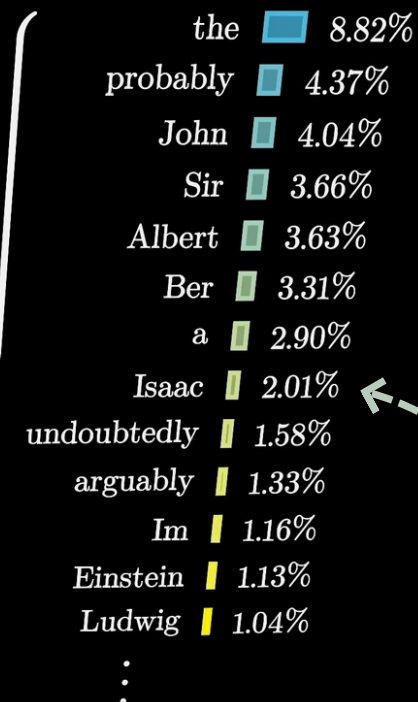
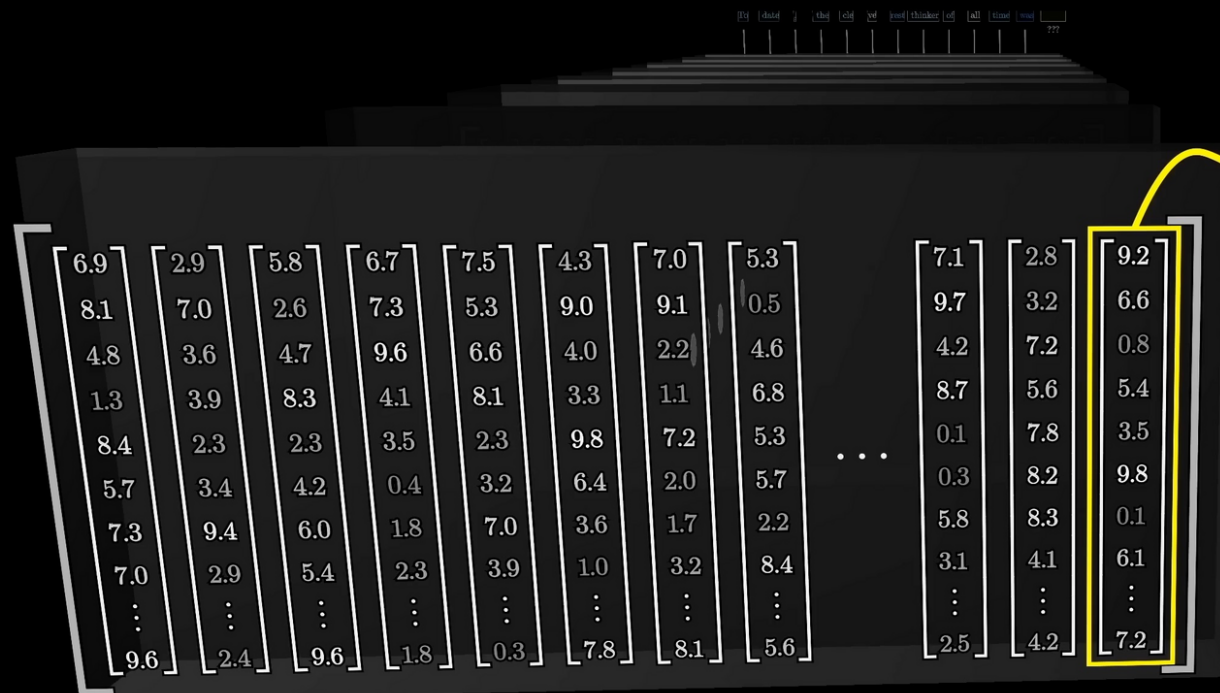
[image edited from [3b1b](#)]



[video edited from [3b1b](#)]

To date, the cleverest thinker of all time was

???



Cross-entropy loss encourages the internal weights update so as to make this probability higher

[image edited from [3b1b](#)]

Generative Boba



By Boyuan Chen
boyuanc@mit.edu

Intro:

Boba tea is an essential component of Asian students' well-being and productivity. Prior studies [1] have suggested the lack of accessible boba shop hurts MIT's reputation as a top CS school. To address this problem, we open a novel boba shop at MIT - Generative Boba. By providing PhD students with free boba every afternoon, Generative Boba boosts the research productivity of the floor by TBD%. Generative Boba is also looking for Chief Boba Scientists to scale up! Contact me to be a co-author.

[1] Chen, B. (2022, June 20). *BEST COMPUTER SCIENCE SCHOOLS RANKED BY BOBA*. Boyuan's Blog. Retrieved March 27, 2024,.

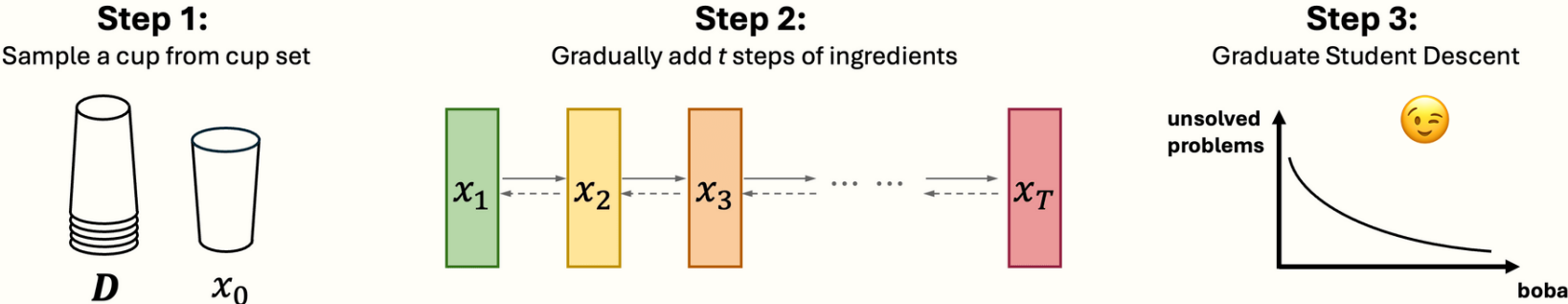


Fig 1: We propose boba diffusion, a novel generative model that boosts PhD student productivity

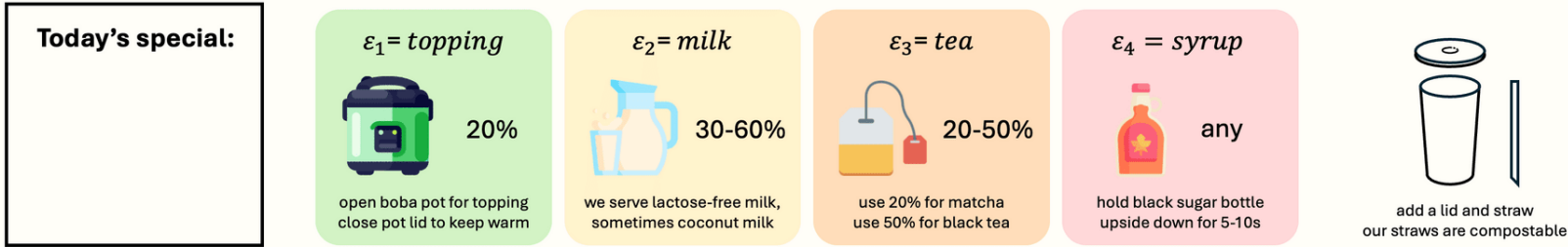


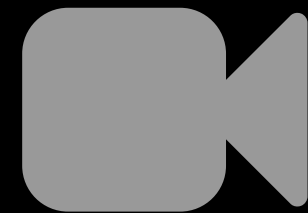
Fig 2: Implementation details of boba diffusion. We follow an efficient ingredient schedule while "today's special" provides special recipes from time to time.



Generative Boba by [Boyuan Chen](#) in Bldg 45

image credit: [Nicholas Pfaff](#)

American shrew mole



[video edited from [3b1b](#)]

American shrew mole

0.2
1.2
0.8
7.7
4.1
7
0.3

9.4
6.7
9.0
1.8
3.7
7
3.9

5.8
9.9
2.5
3.7
9.1
⋮
2.1

One mole of carbon dioxide

5.2
7.4
2.9
2.8
9.8
1.1
2.7

5.8
9.9
2.5
3.7
9.1
⋮
2.1

8.8
2.8
4.8
8.1
6.3
1
4.9

7.4
6.4
5.7
8.1
3.8
1
1.8

9.8
1.8
8.1
9.8
9.1
1
9.9

Take a biopsy of the mole

1.9
1.1
1.7
9.8
8.9
⋮
1.3

9.5
9.7
1.8
8.3
9.7
1
8.8

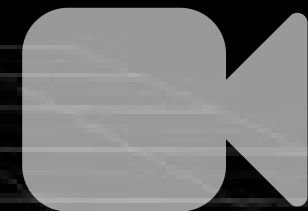
1.1
8.7
1.4
2.7
4.7
1
2.2

5.8
7.8
1.8
6.1
6.3
1
1.8

2.7
1.8
6.4
8.3
9.3
1
8.1

5.8
9.9
2.5
3.7
9.1
⋮
2.1

$E(\text{mole})$

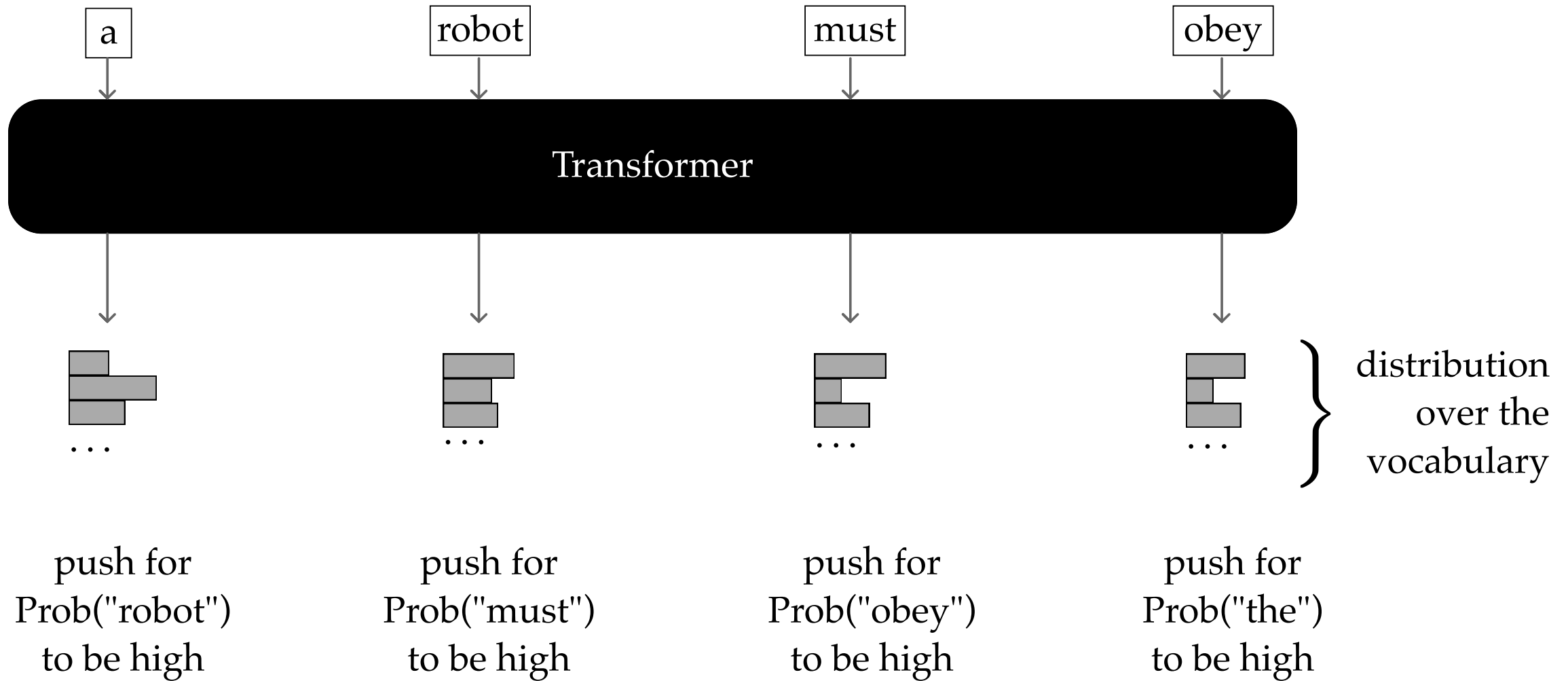


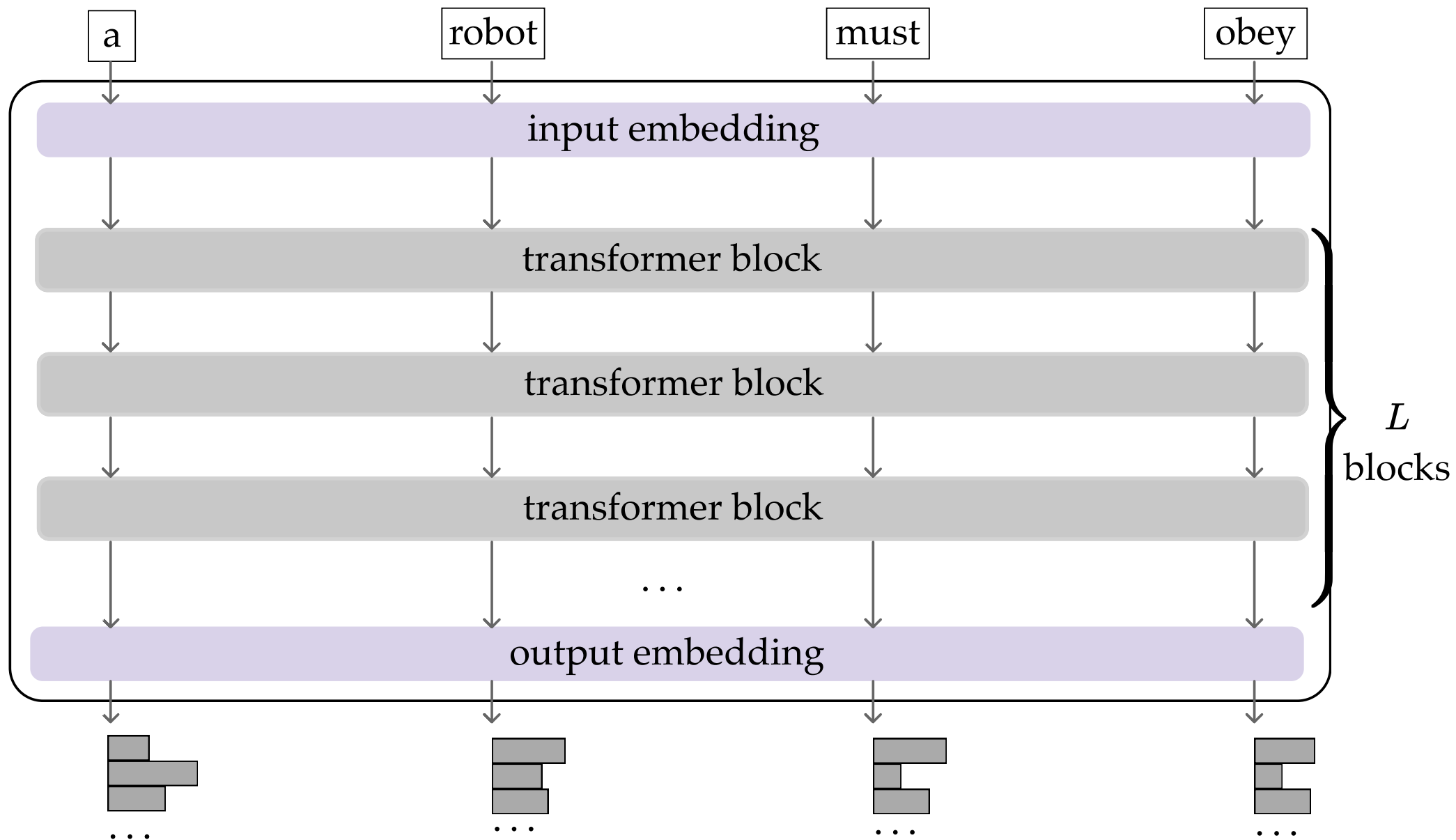
[video edited from [3b1b](#)]

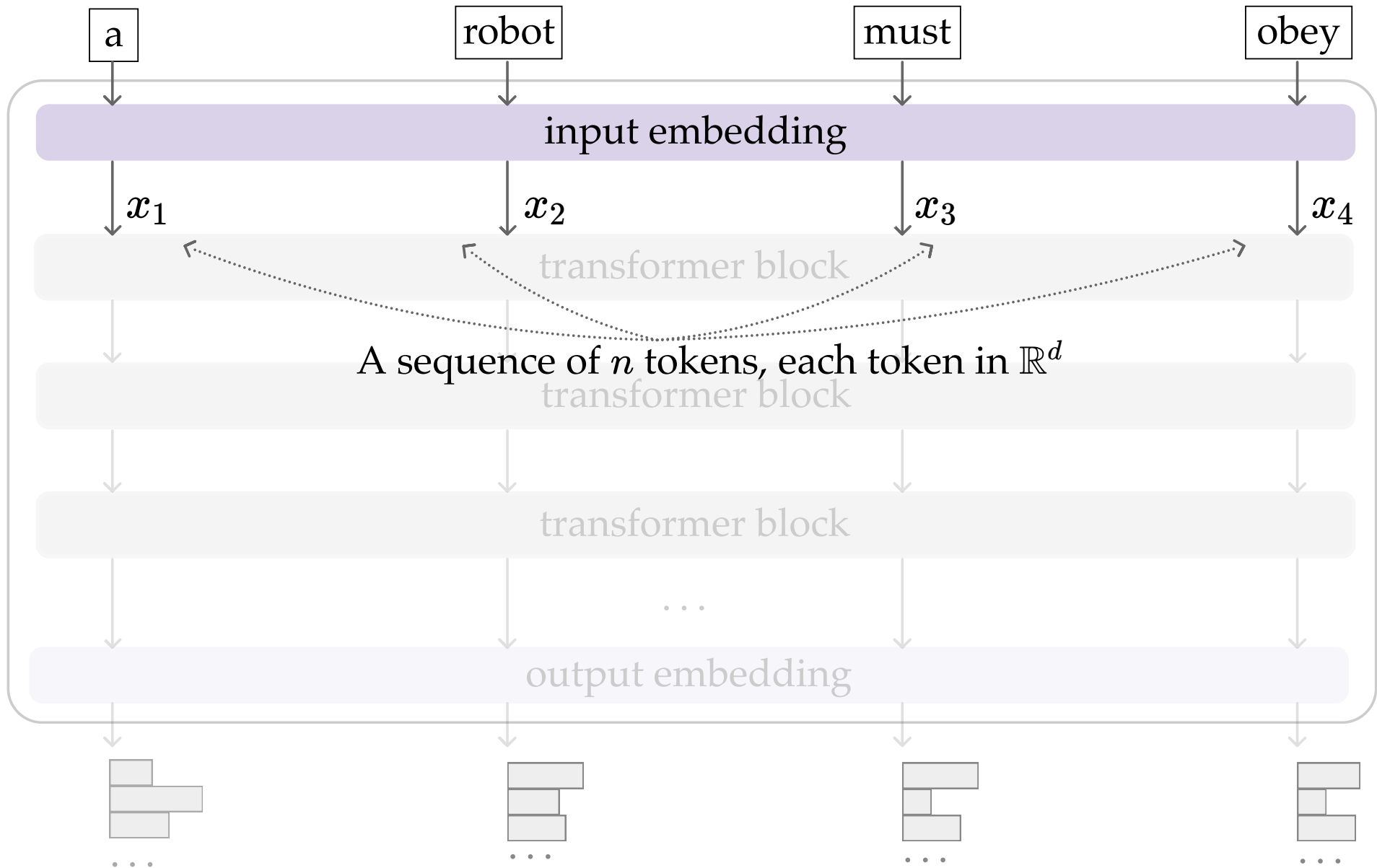
Outline

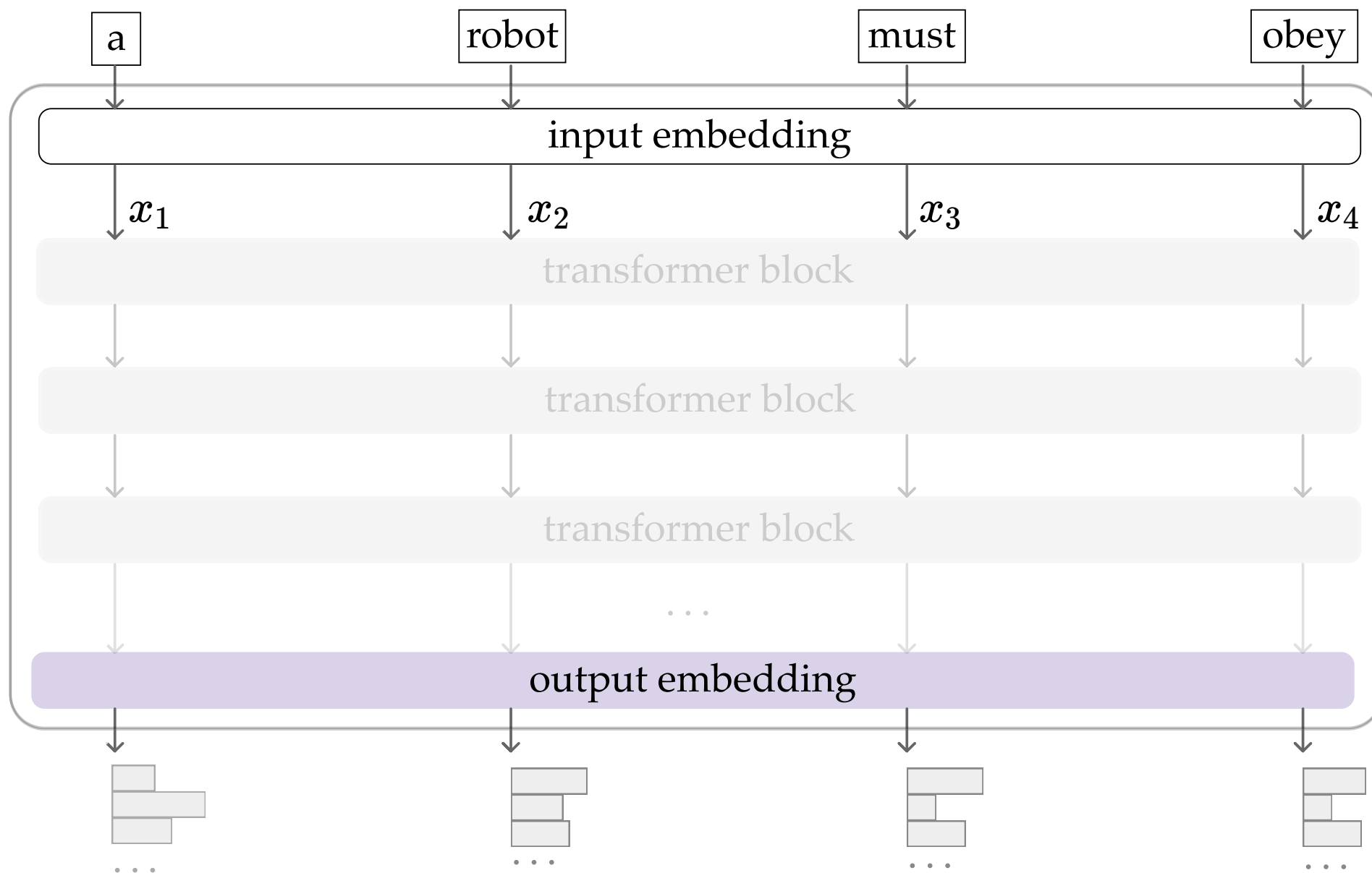
- Recap, embedding and representation
- Transformers high-level intuition
- Transformers architecture overview
- (query, key, value) and self-attention
 - matrix form
- Multi-head Attention
- (Applications)

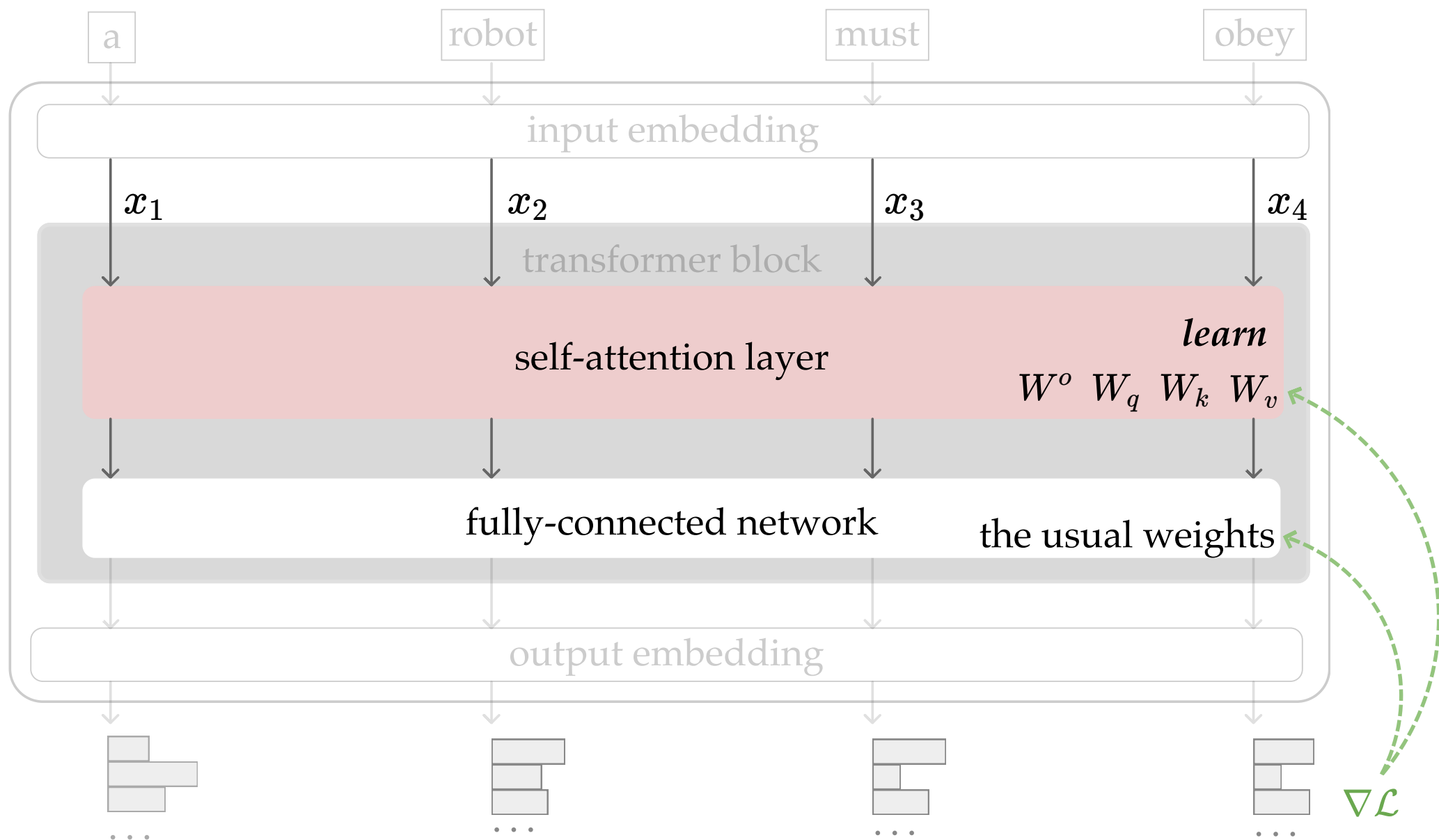
"A robot must obey the orders given it by human beings ..."

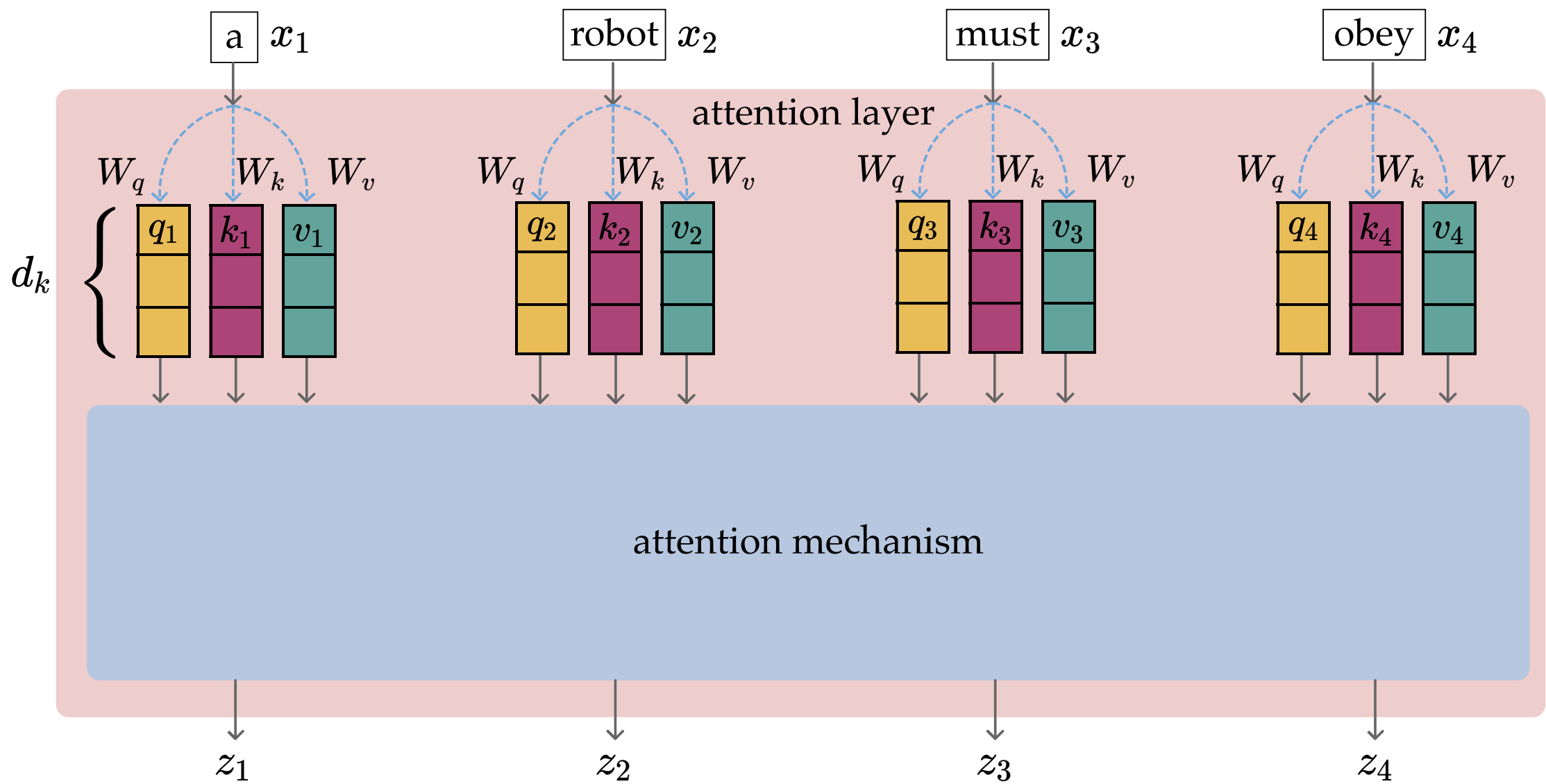


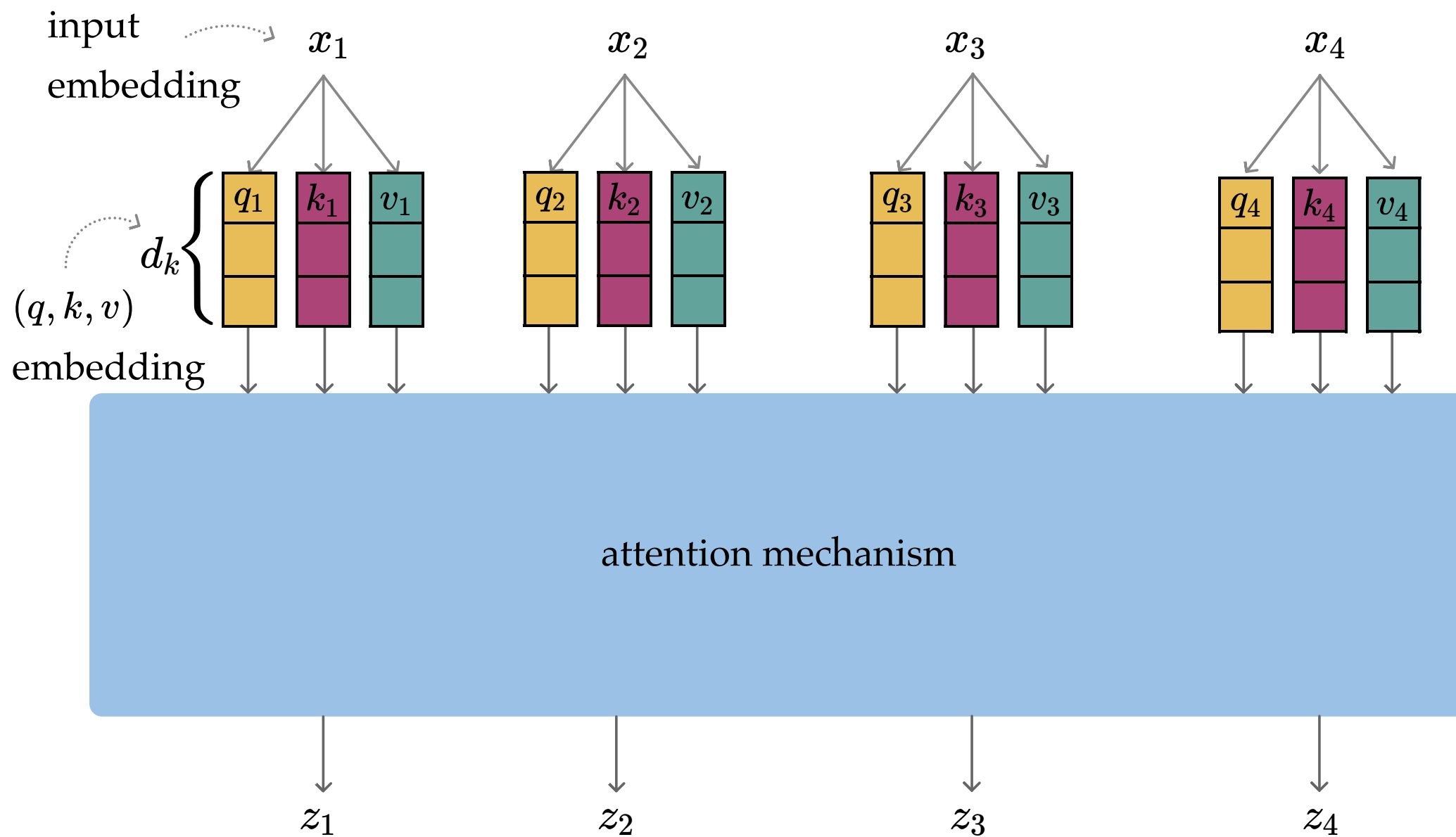


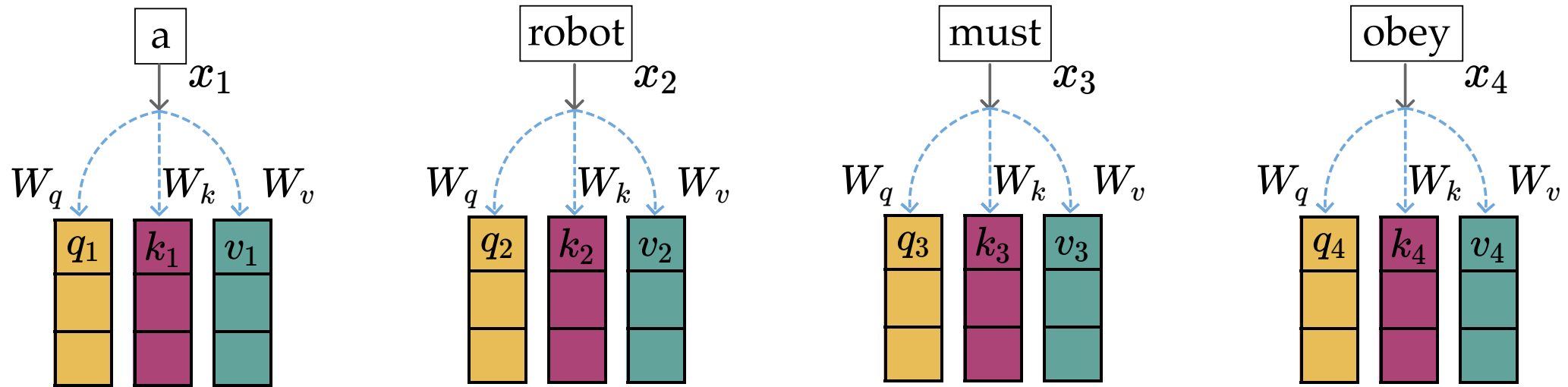








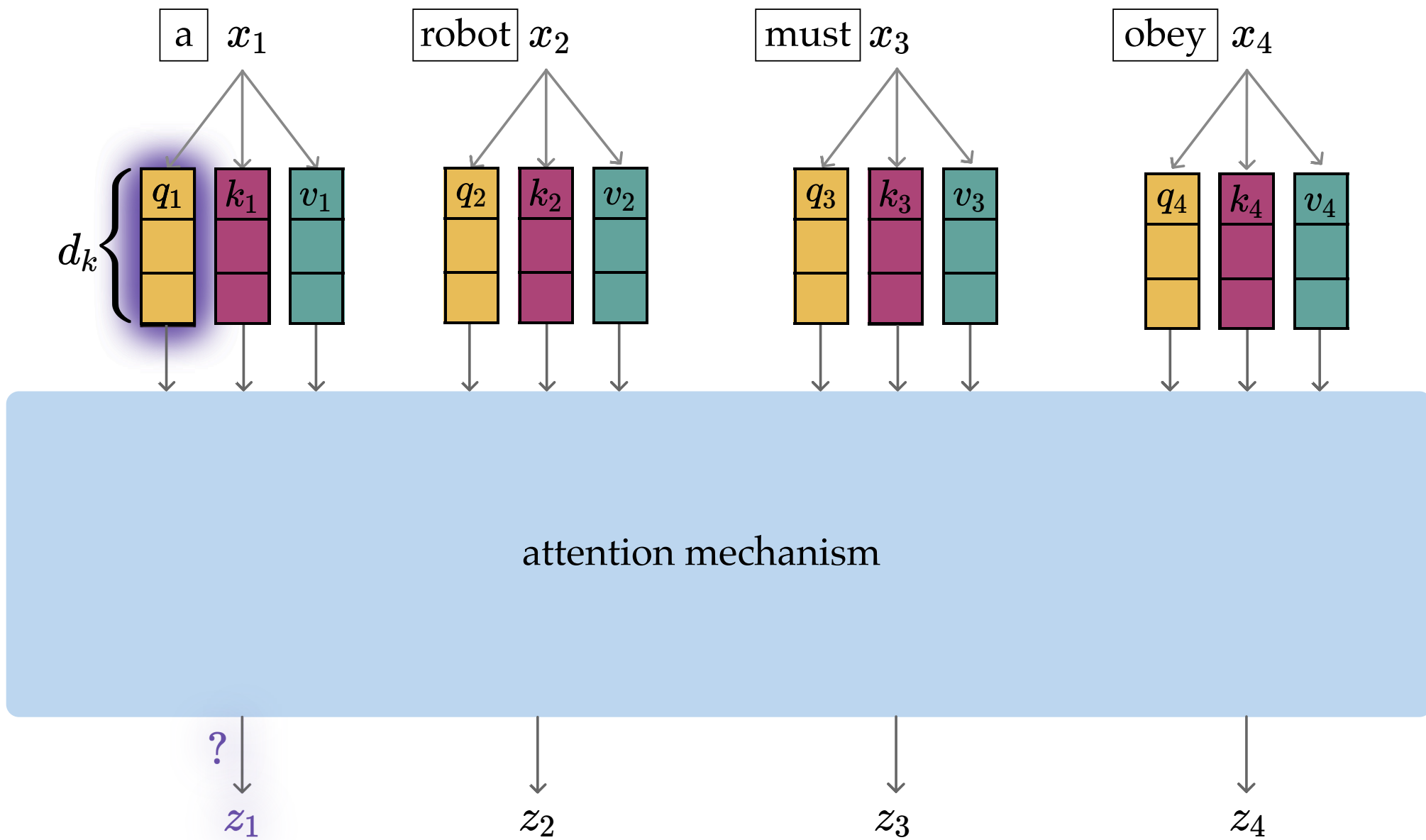


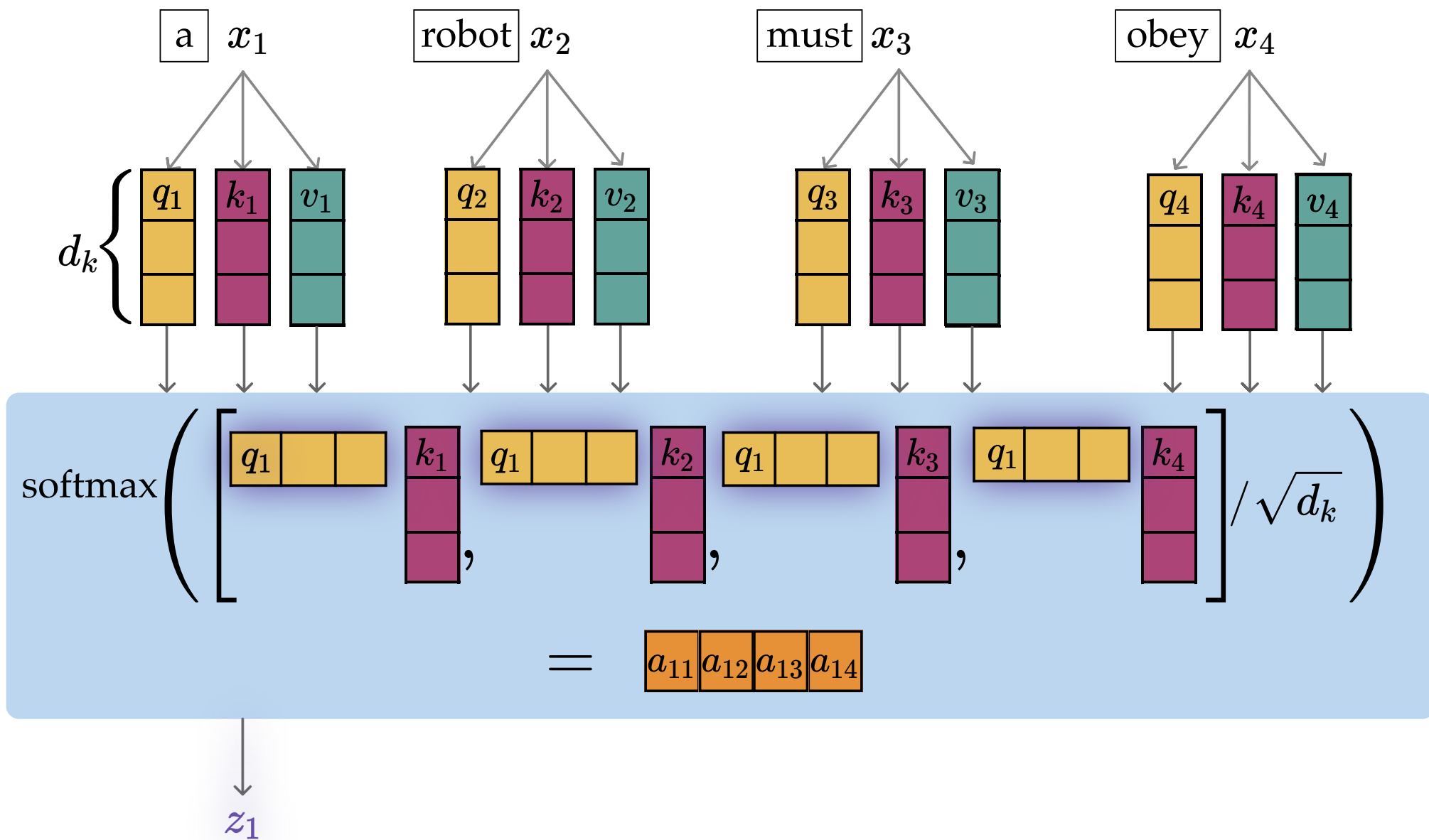


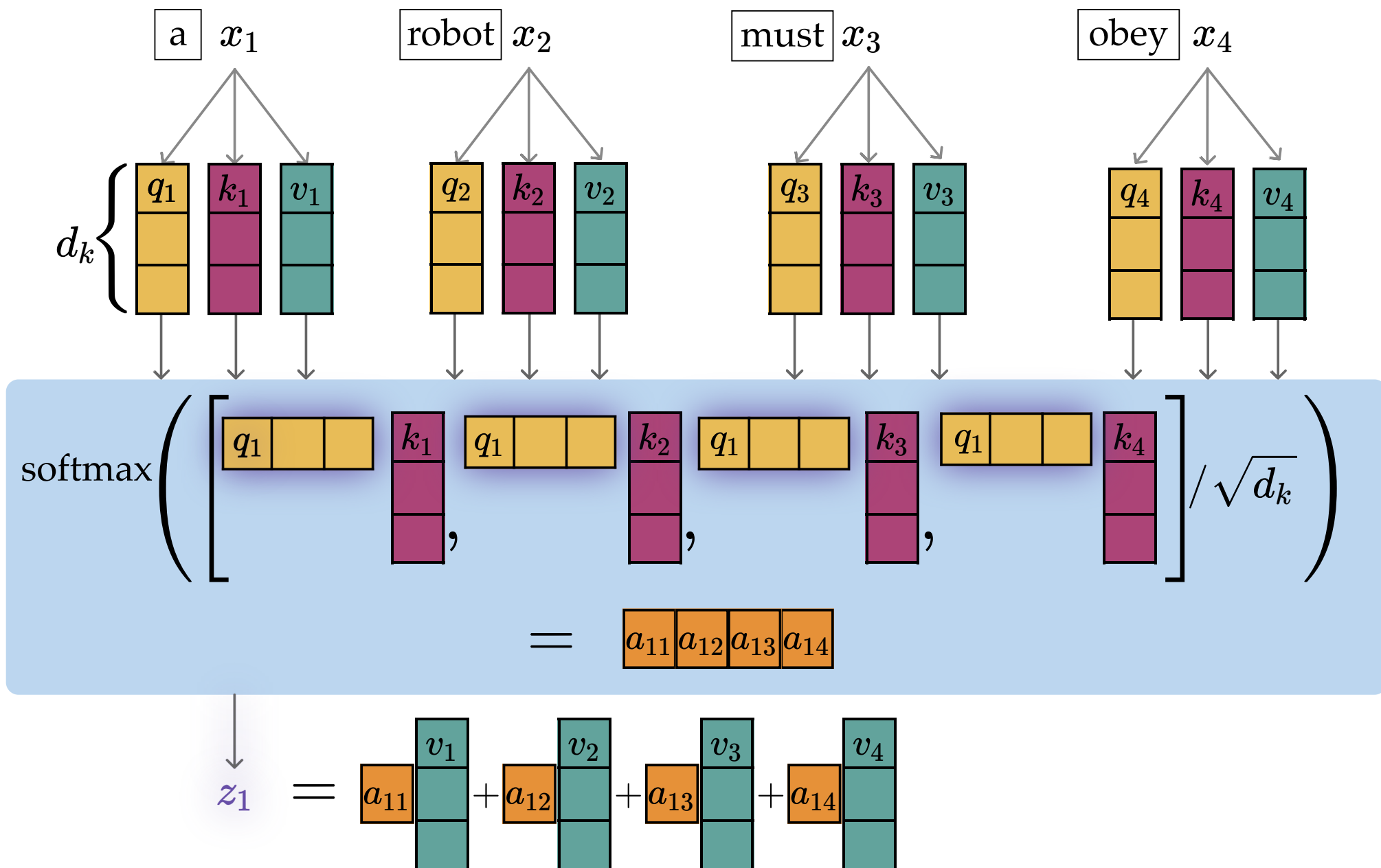
- sequence of d -dimensional input tokens x
- learnable weights, W_q, W_v, W_k , all in $\mathbb{R}^{d \times d_k}$
- map the input sequence into d_k -dimensional (qkv) sequence, e.g., $q_1 = W_q^T x_1$
- the weights are **shared**, across the sequence of tokens -- **parallel** processing

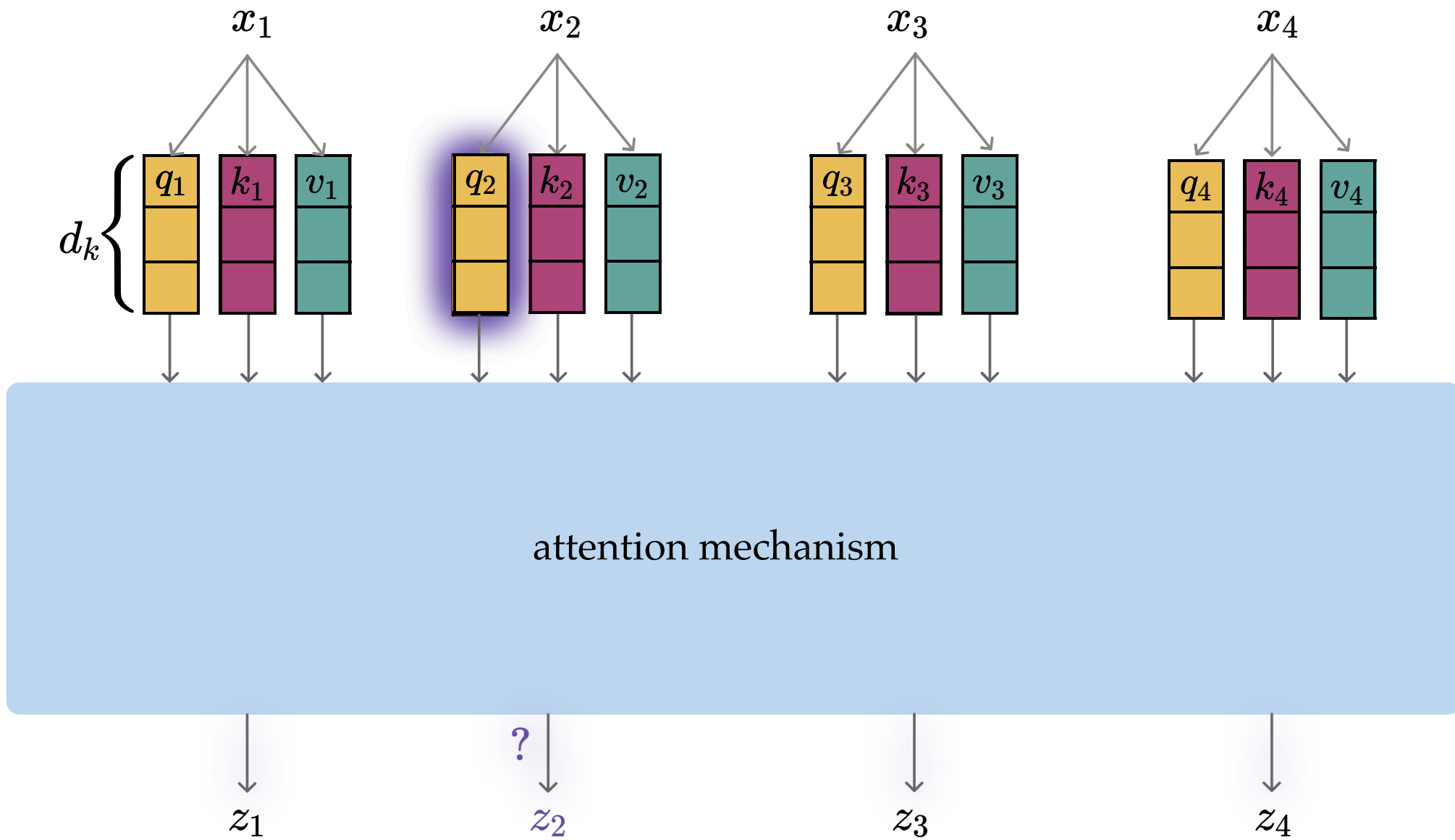
Outline

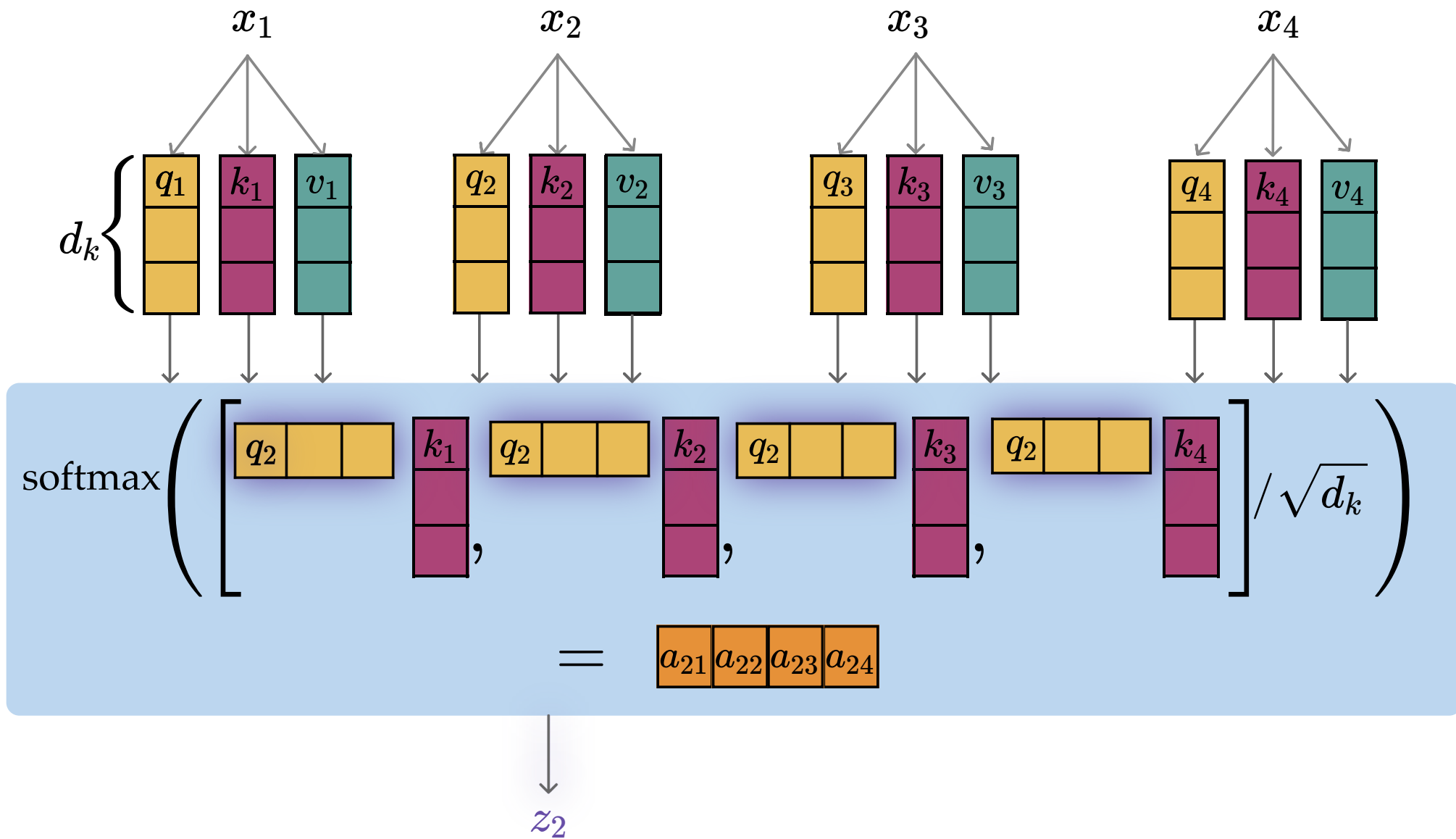
- Recap, embedding and representation
- Transformers high-level intuition
- Transformers architecture overview
- (query, key, value) and self-attention
 - matrix form
- Multi-head Attention
- (Applications)

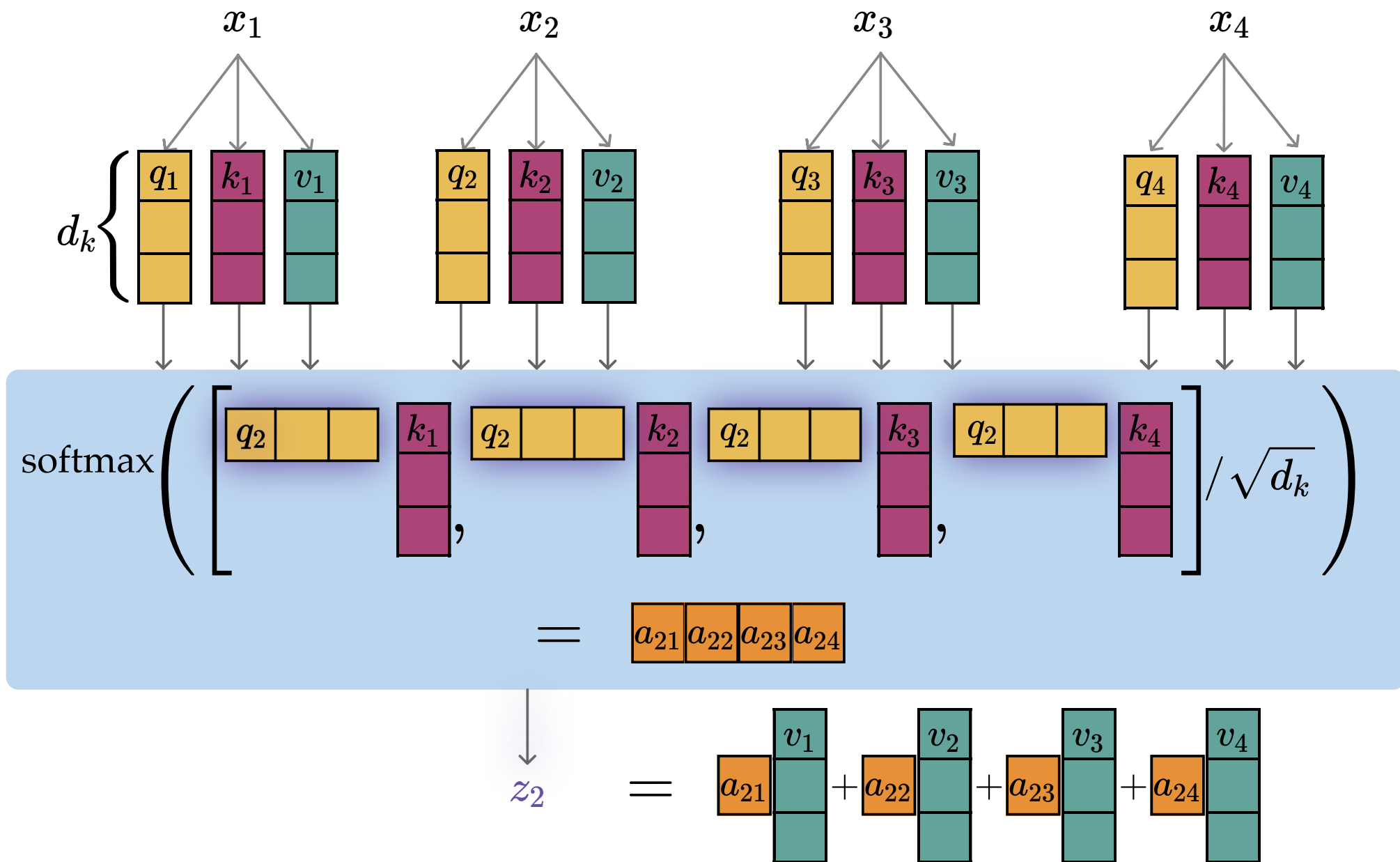


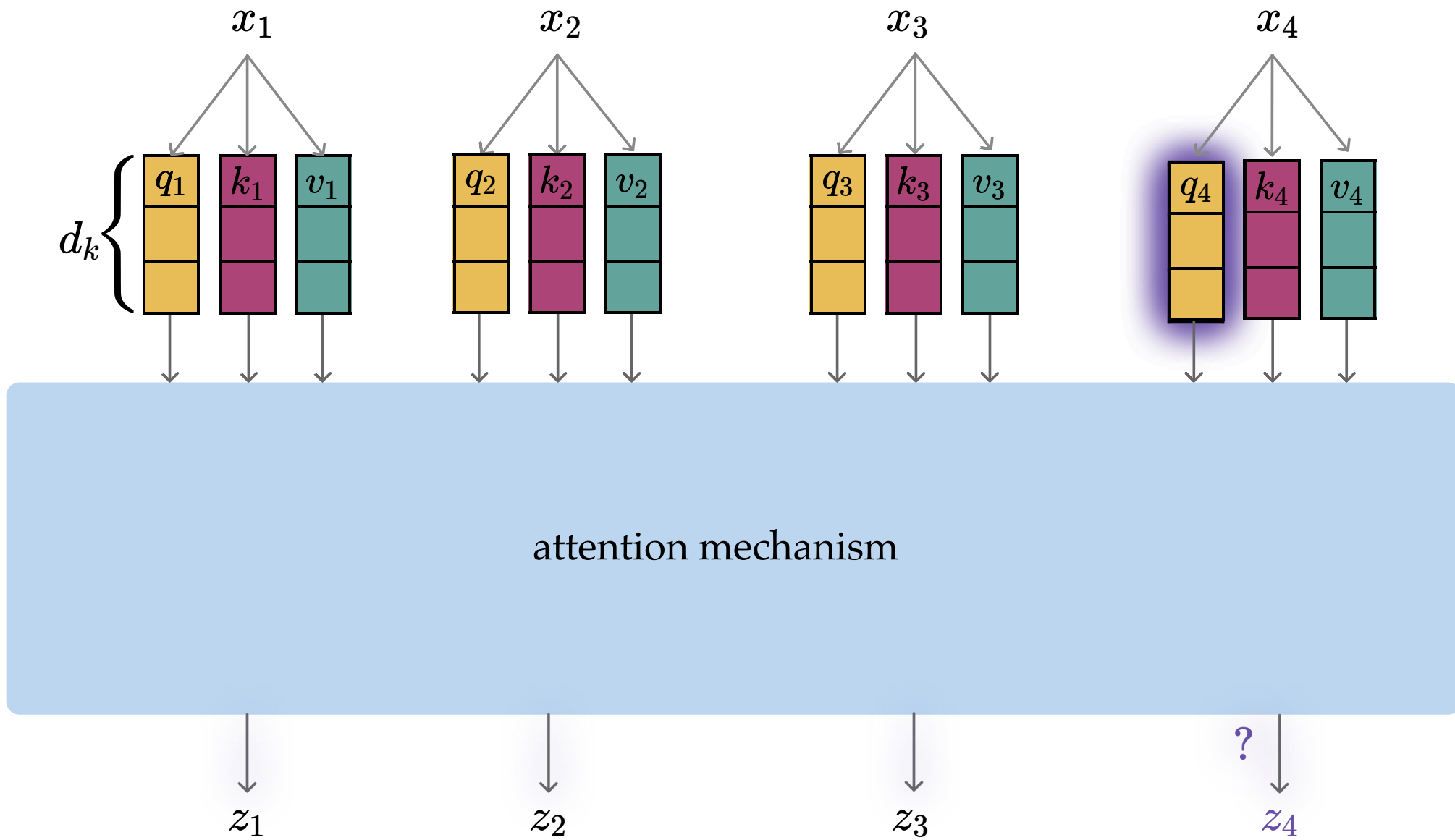


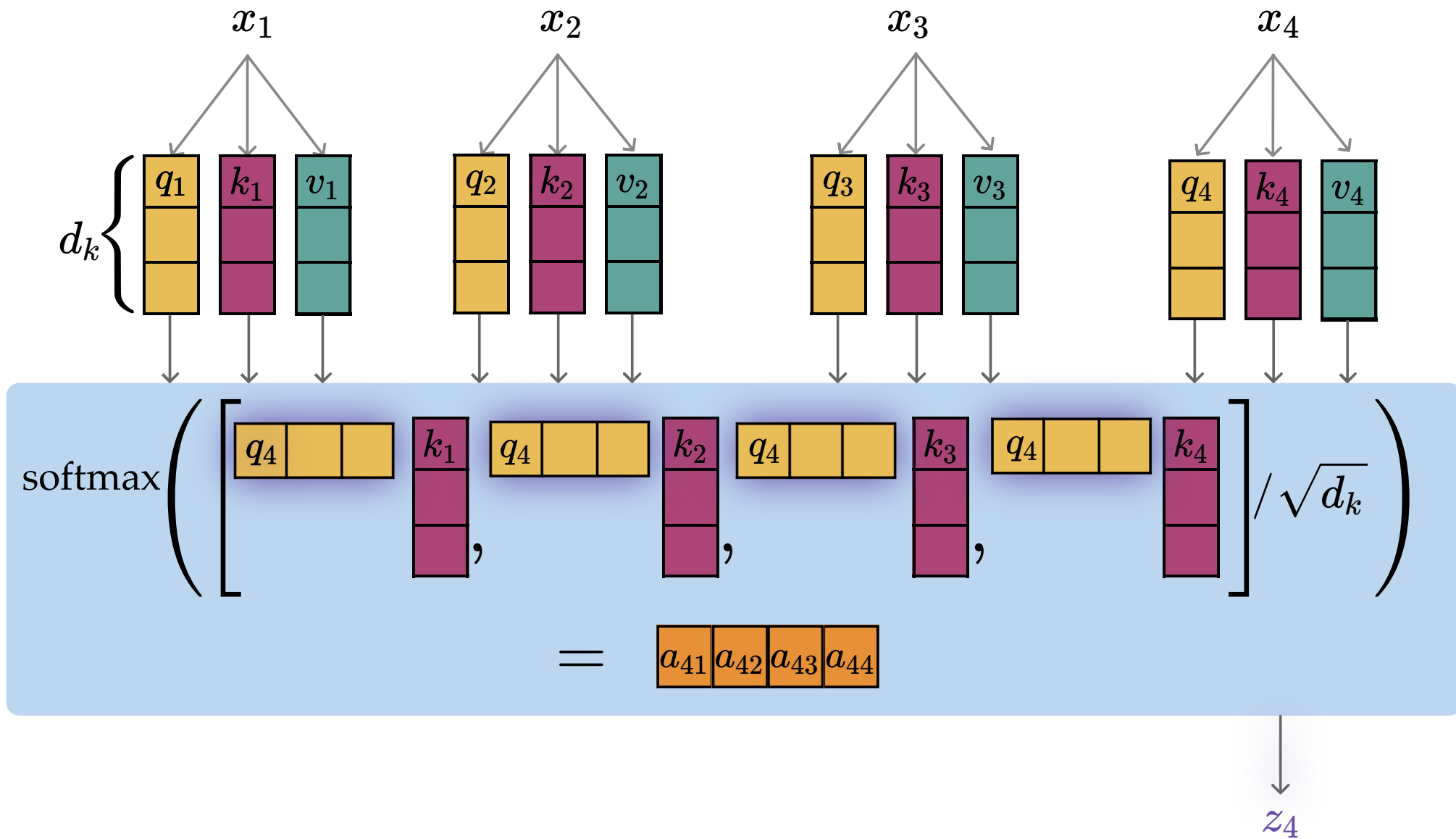


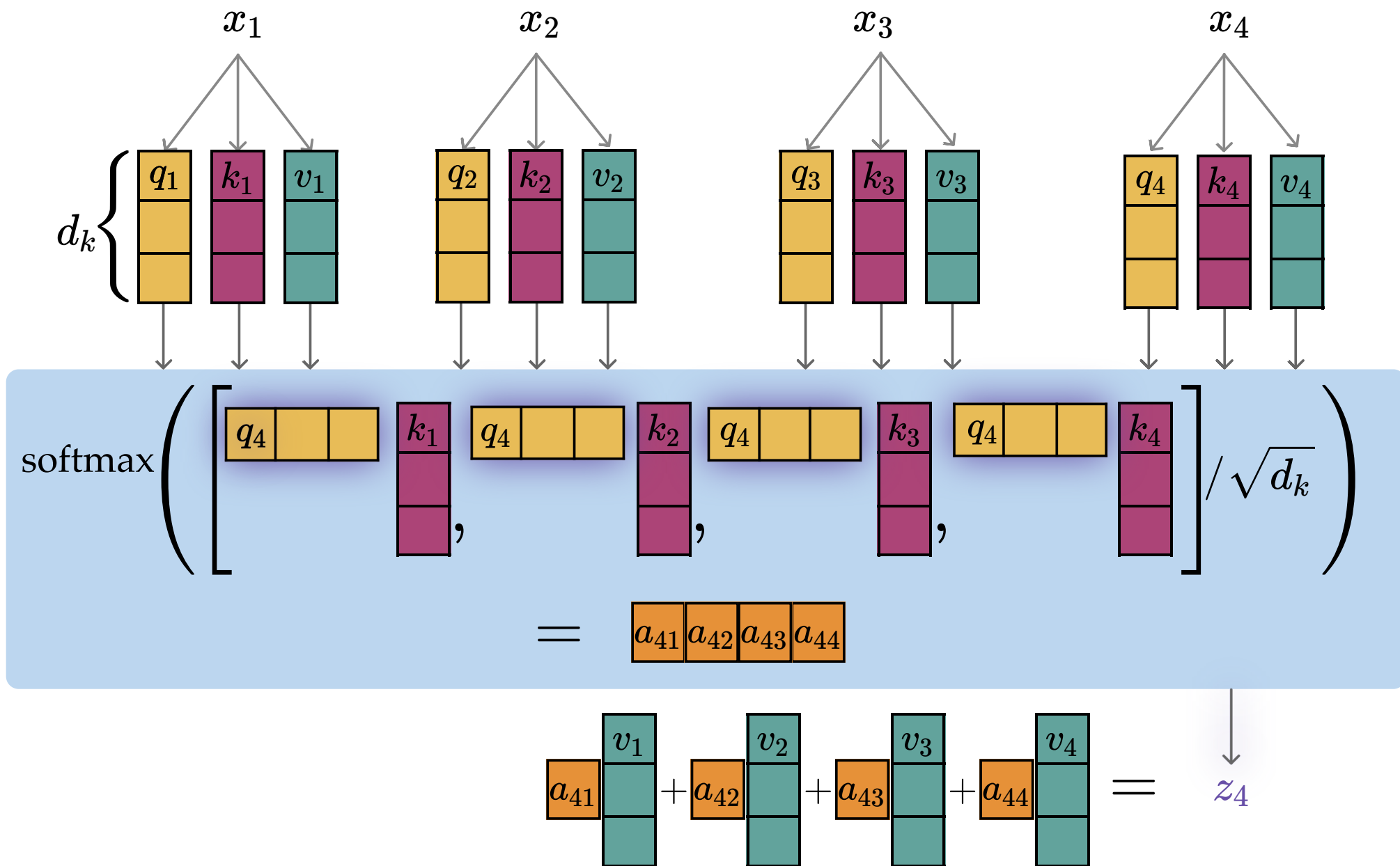


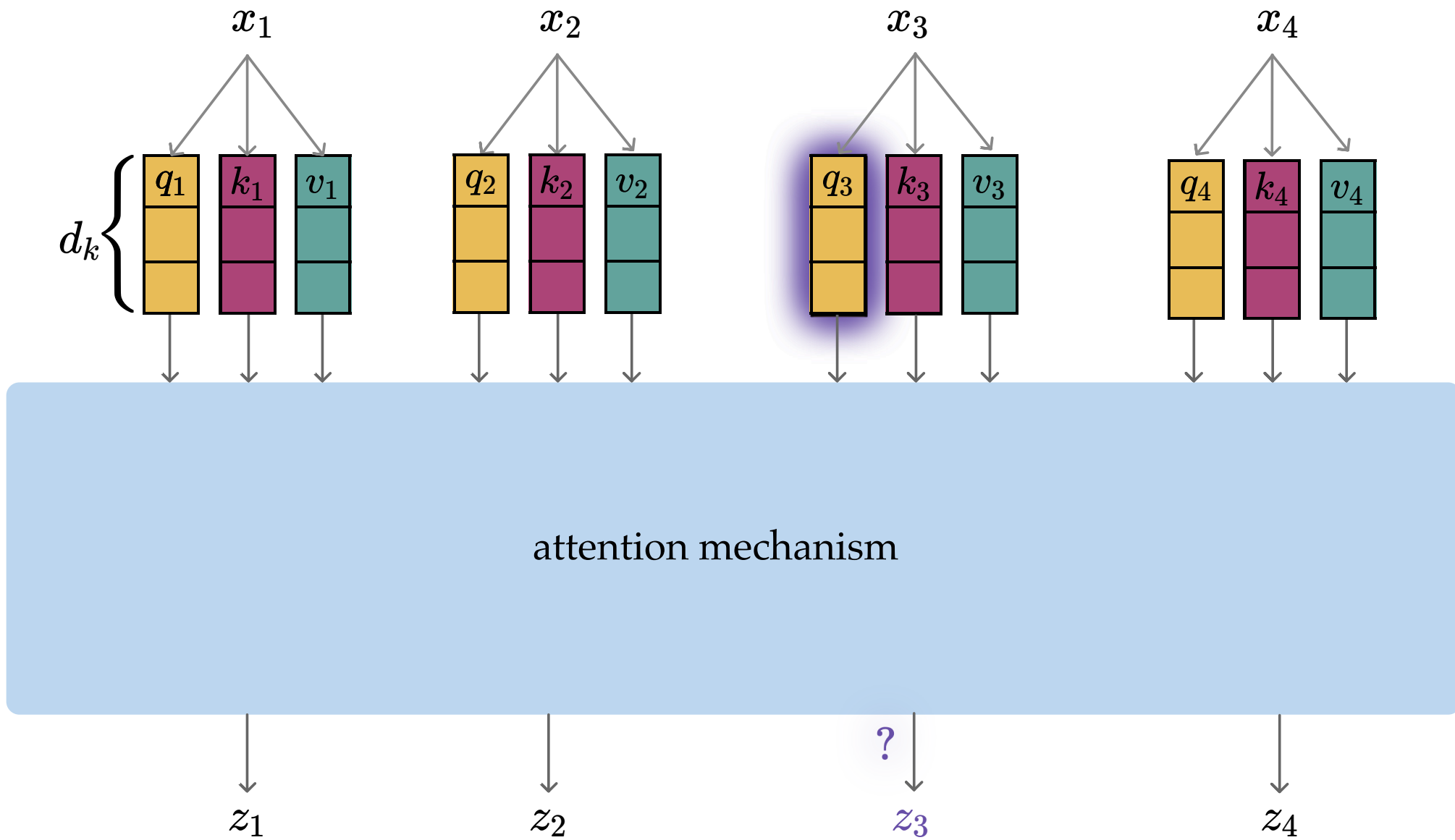


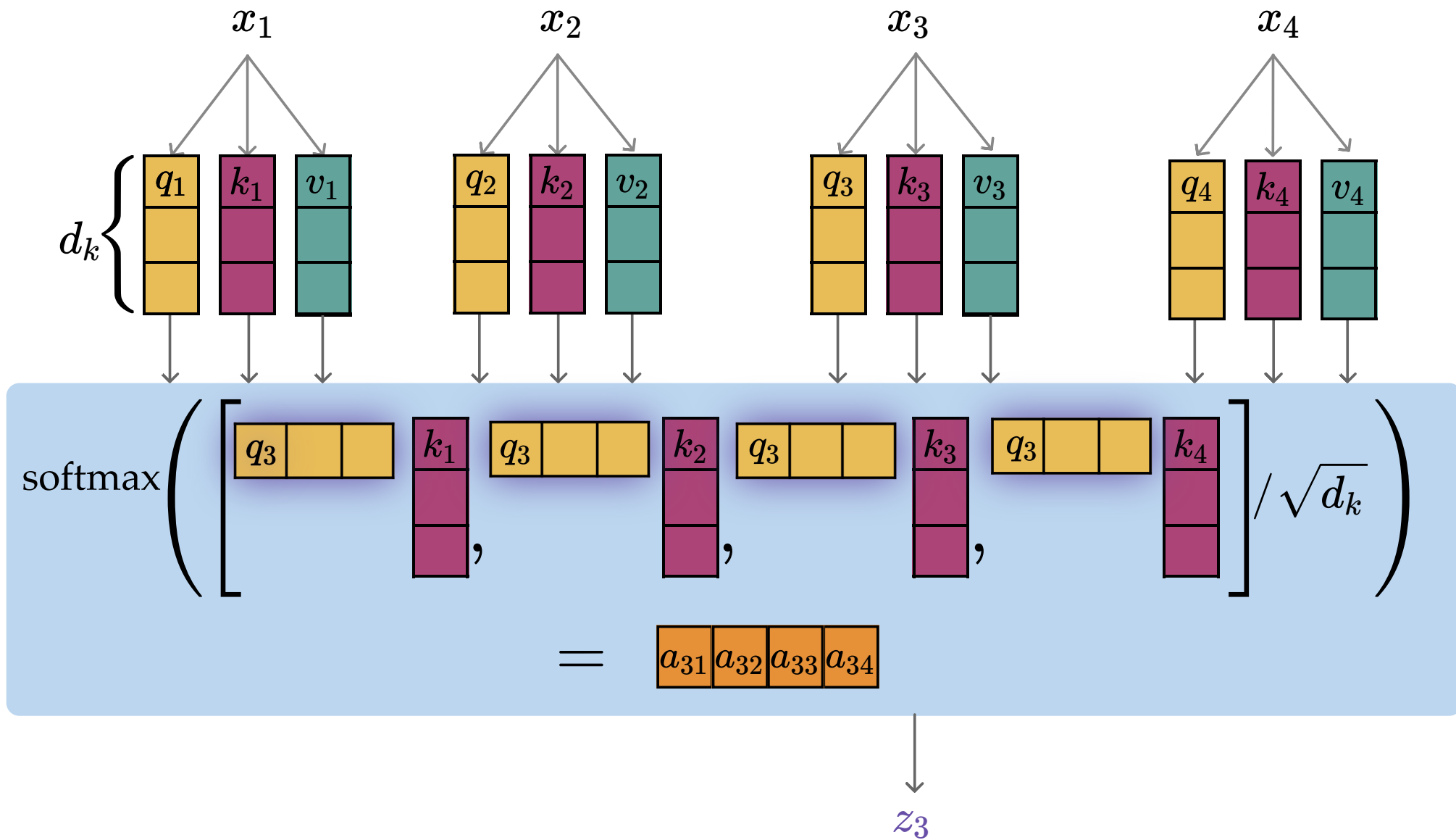


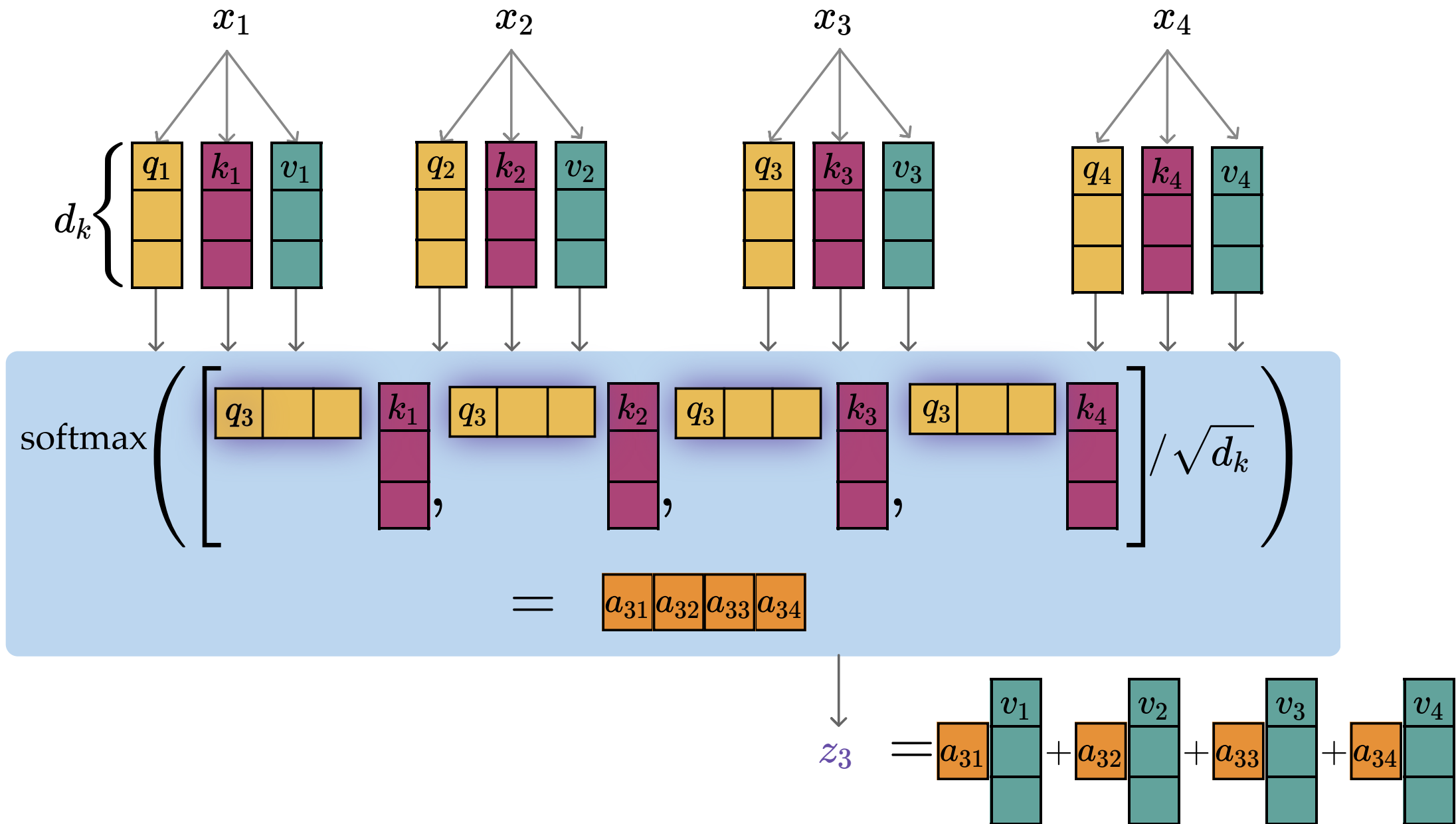












Outline

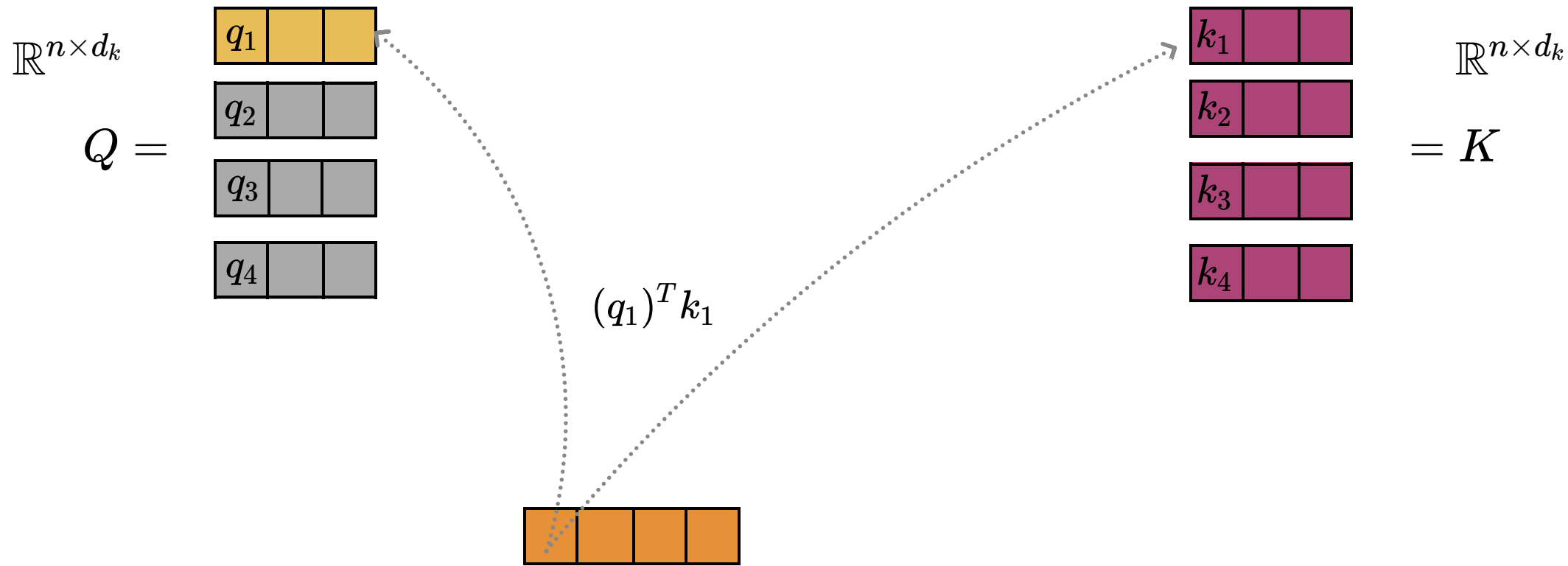
- Recap, embedding and representation
- Transformers high-level intuition
- Transformers architecture overview
- (query, key, value) and self-attention
 - matrix form
- Multi-head Attention
- (Applications)

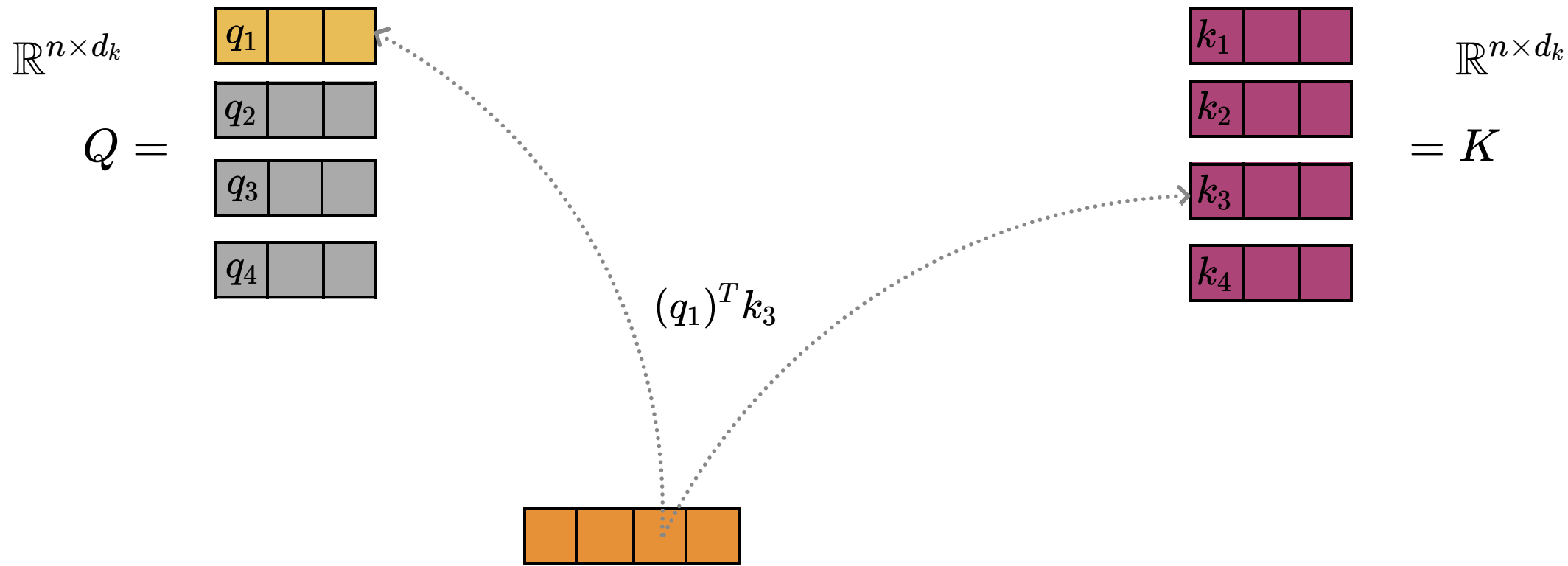
$$\mathbb{R}^{n \times d_k}$$

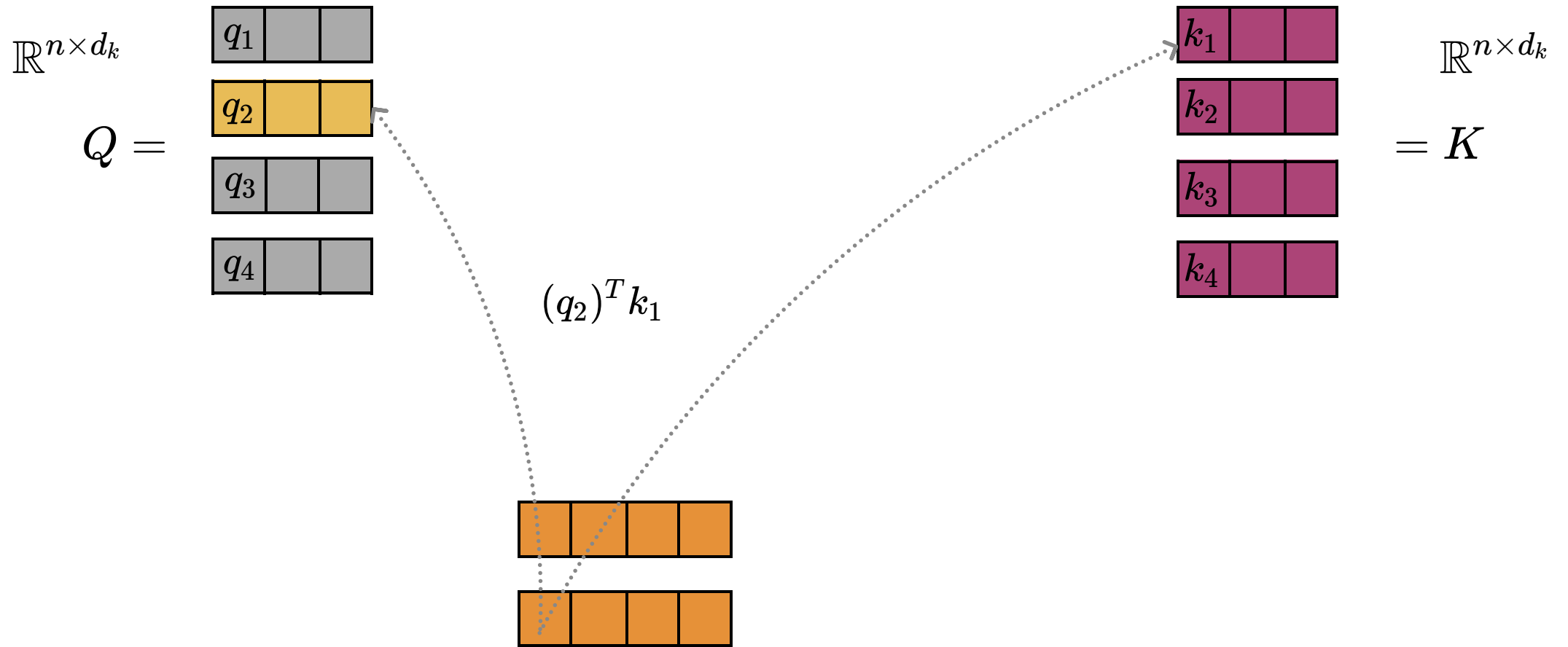
$$Q = \begin{bmatrix} q_1 & & \\ q_2 & & \\ q_3 & & \\ q_4 & & \end{bmatrix}$$

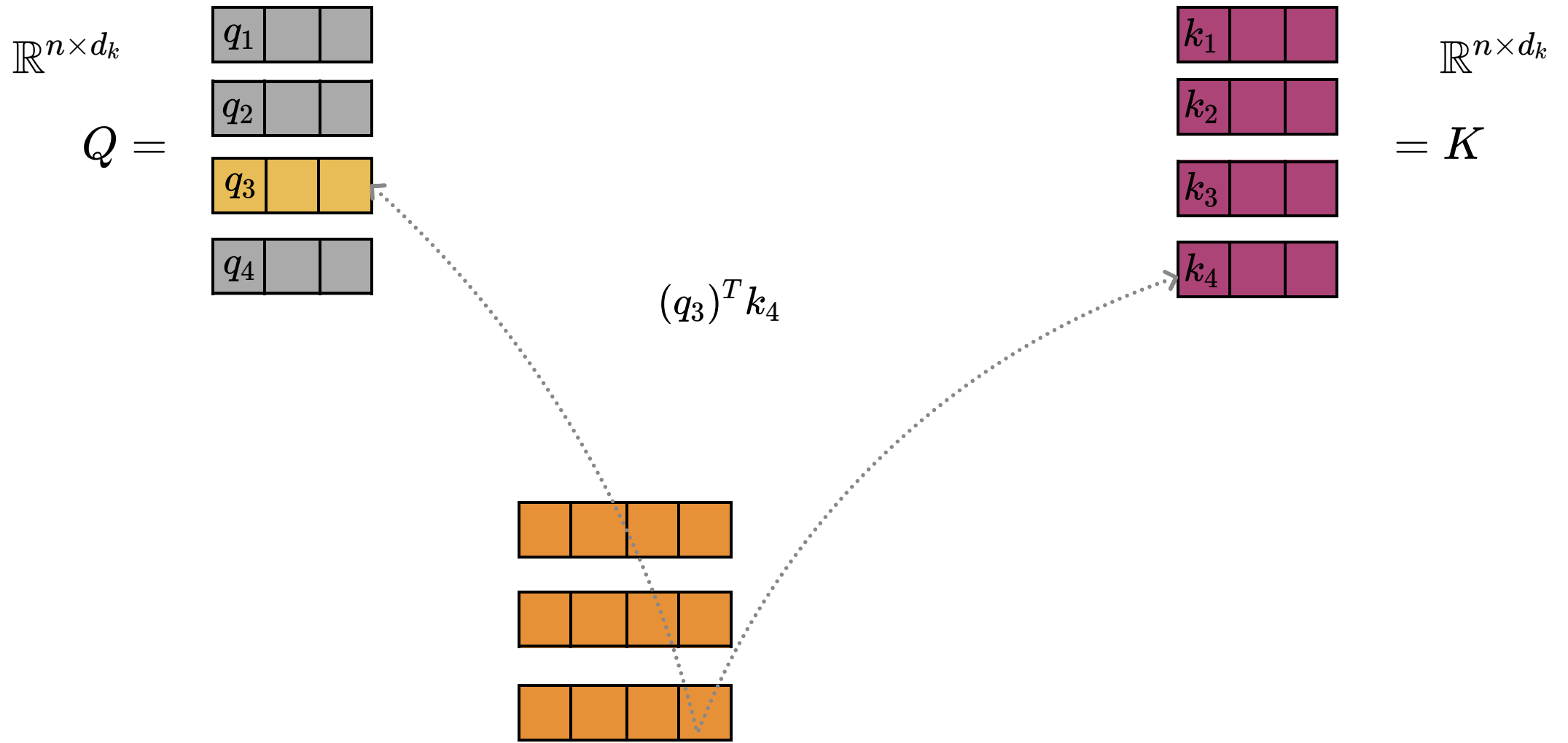
$$\mathbb{R}^{n \times d_k}$$

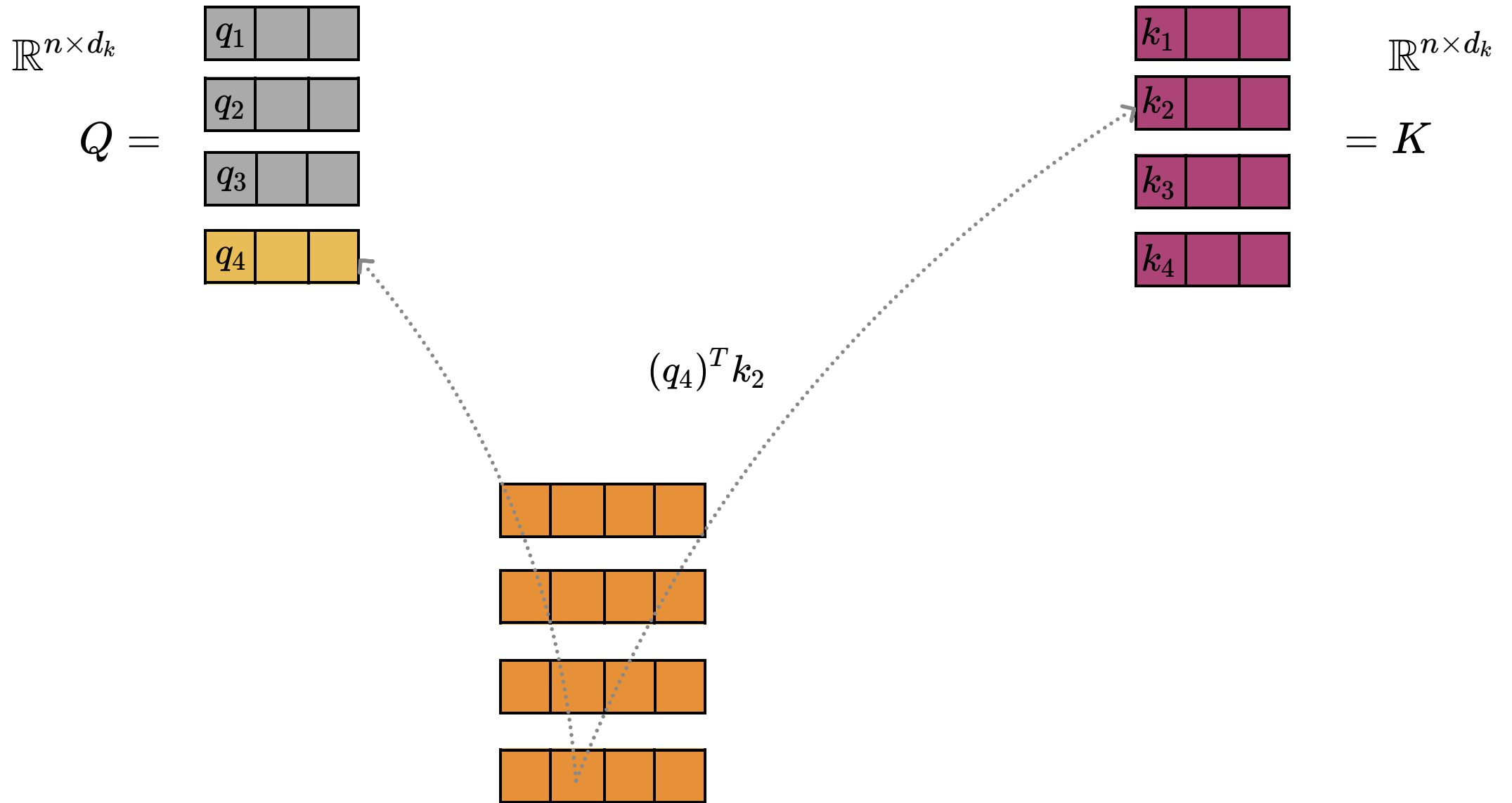
$$= K \begin{bmatrix} k_1 & & \\ k_2 & & \\ k_3 & & \\ k_4 & & \end{bmatrix}$$









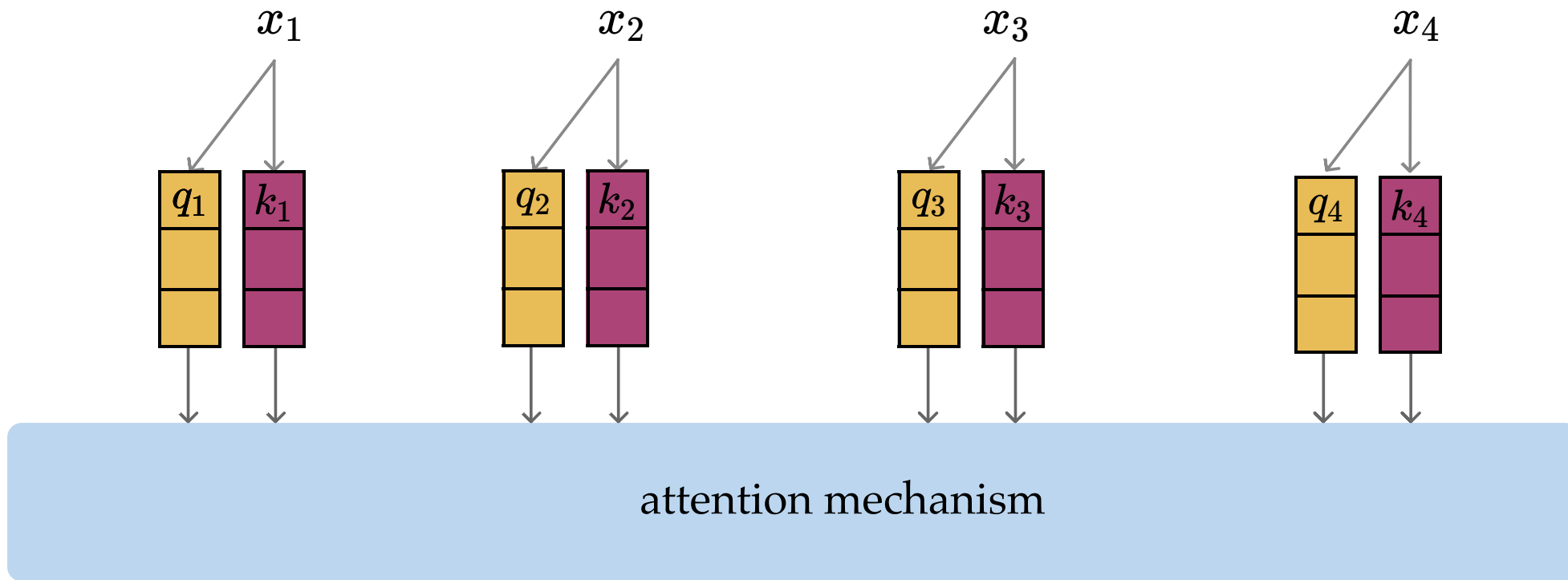


$$\begin{array}{c}
 \mathbb{R}^{n \times d_k} \\
 Q = \begin{array}{|c|c|c|} \hline q_1 & & \\ \hline q_2 & & \\ \hline q_3 & & \\ \hline q_4 & & \\ \hline \end{array}
 \end{array}
 \qquad
 \begin{array}{c}
 \mathbb{R}^{n \times d_k} \\
 = K = \begin{array}{|c|c|c|} \hline k_1 & & \\ \hline k_2 & & \\ \hline k_3 & & \\ \hline k_4 & & \\ \hline \end{array}
 \end{array}$$

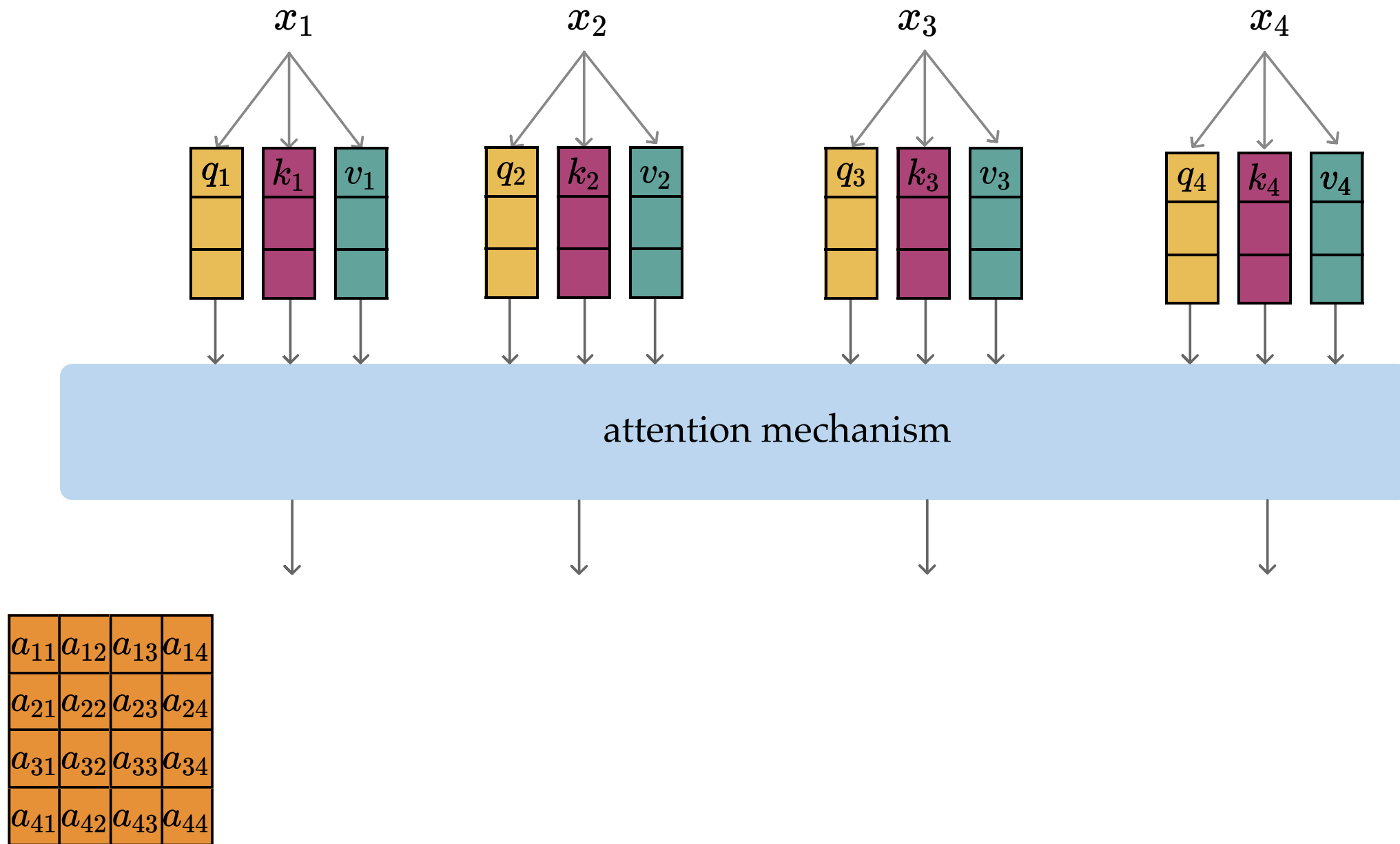
attention matrix

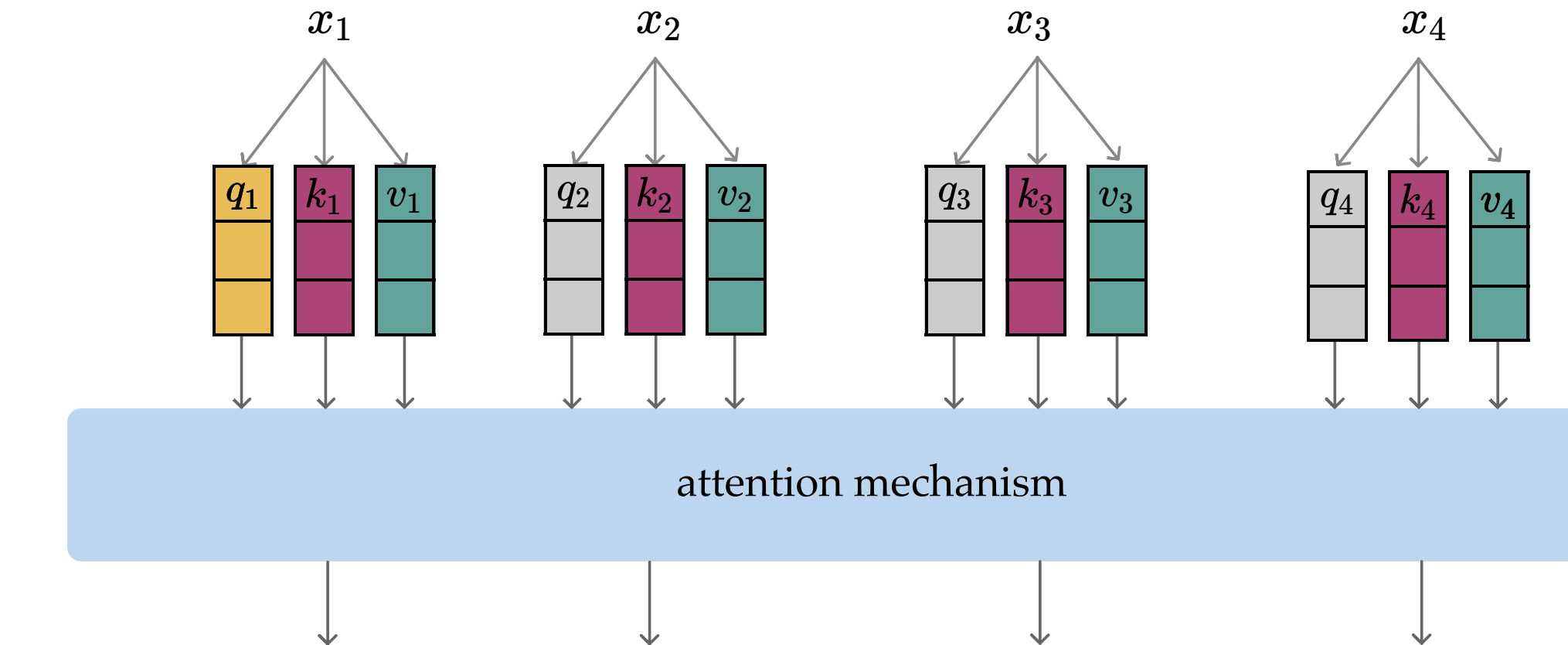
each row sums up to 1

$$\begin{array}{c}
 \mathbb{R}^{n \times n} \\
 A = \begin{bmatrix} \text{softmax} \left(\begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} / \sqrt{d_k} \right) \\ \text{softmax} \left(\begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} / \sqrt{d_k} \right) \\ \text{softmax} \left(\begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} / \sqrt{d_k} \right) \\ \text{softmax} \left(\begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} / \sqrt{d_k} \right) \end{bmatrix} = \begin{array}{|c|c|c|c|} \hline a_{11} & a_{12} & a_{13} & a_{14} \\ \hline a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ \hline a_{41} & a_{42} & a_{43} & a_{44} \\ \hline \end{array}
 \end{array}$$

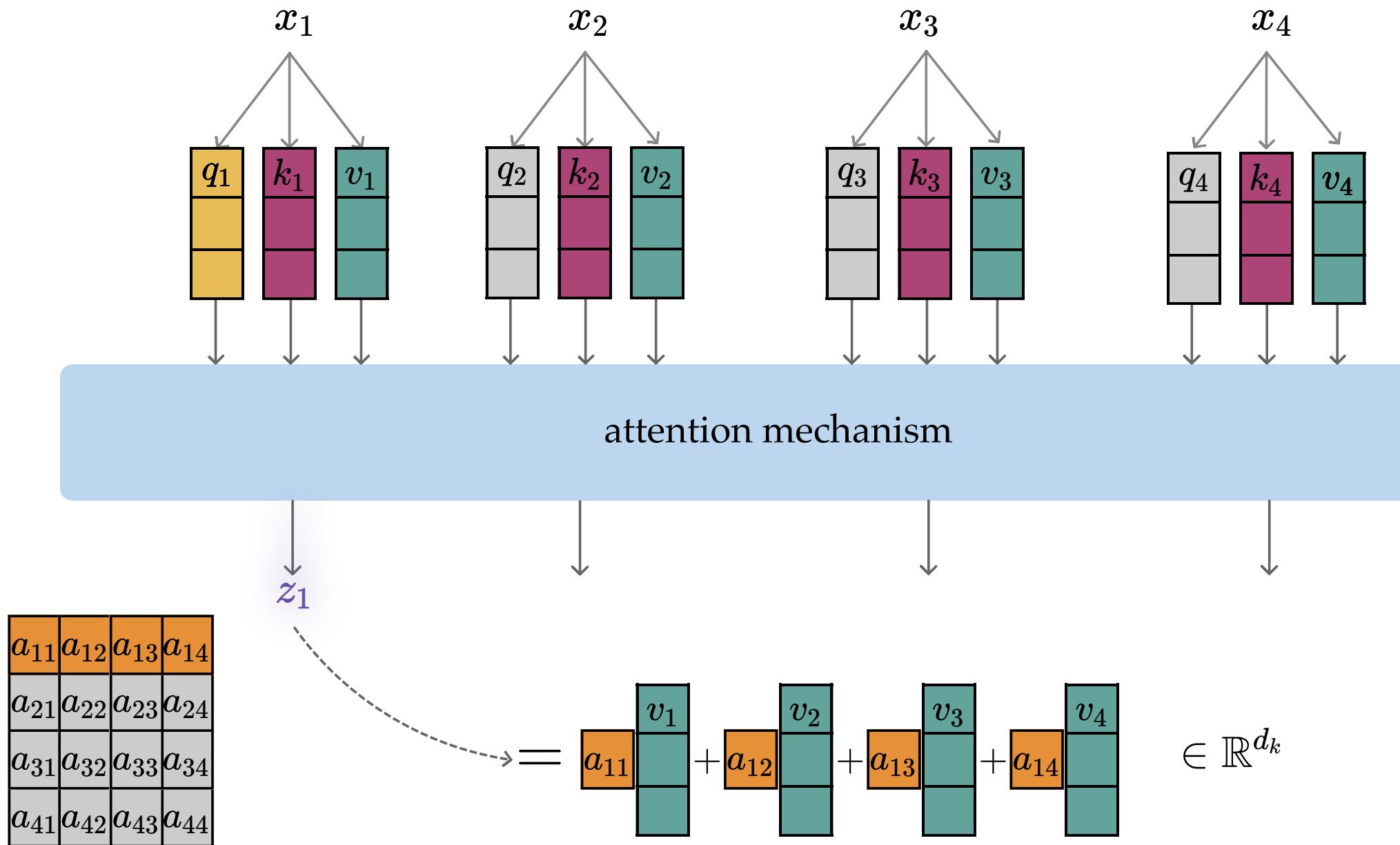


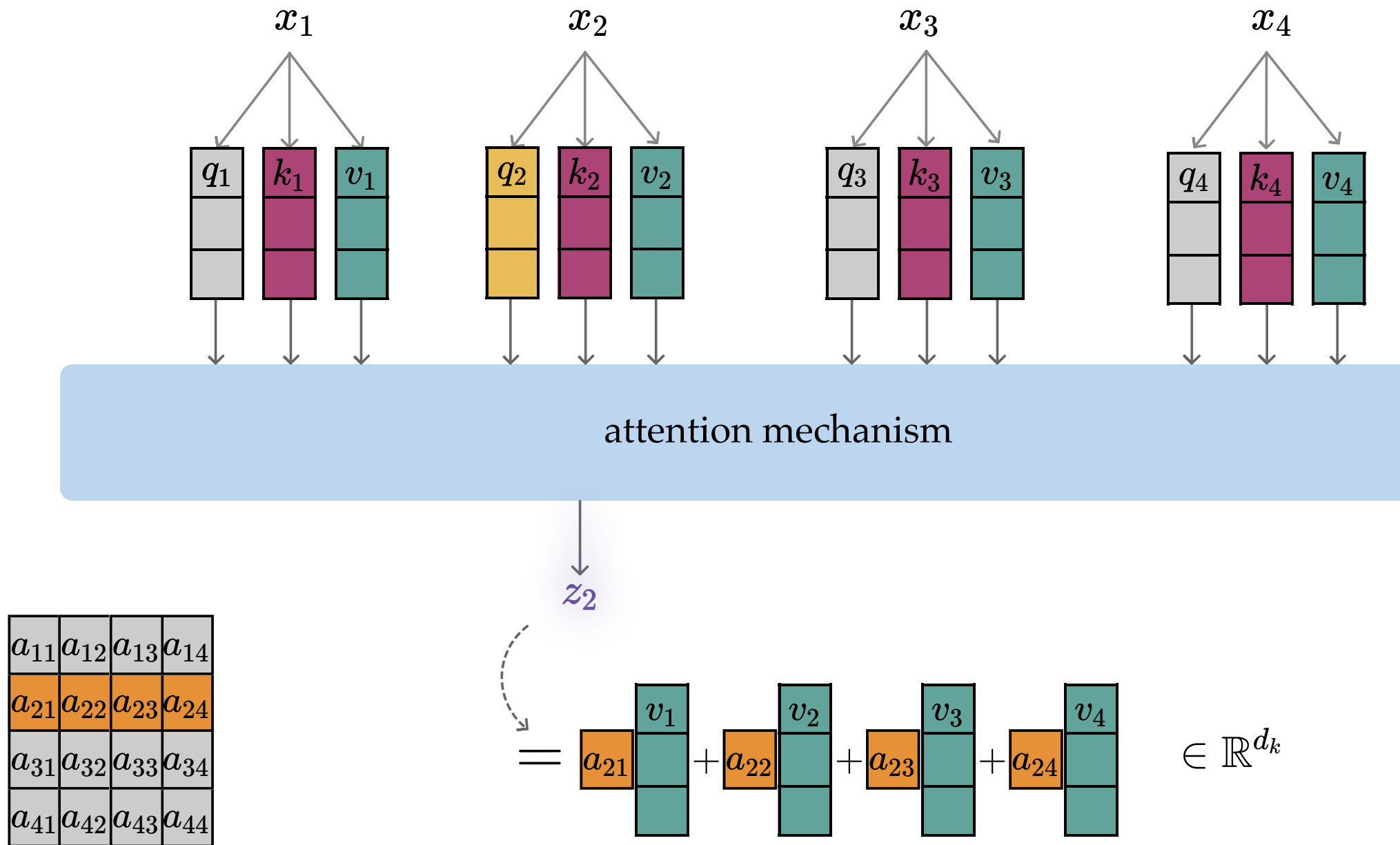
a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

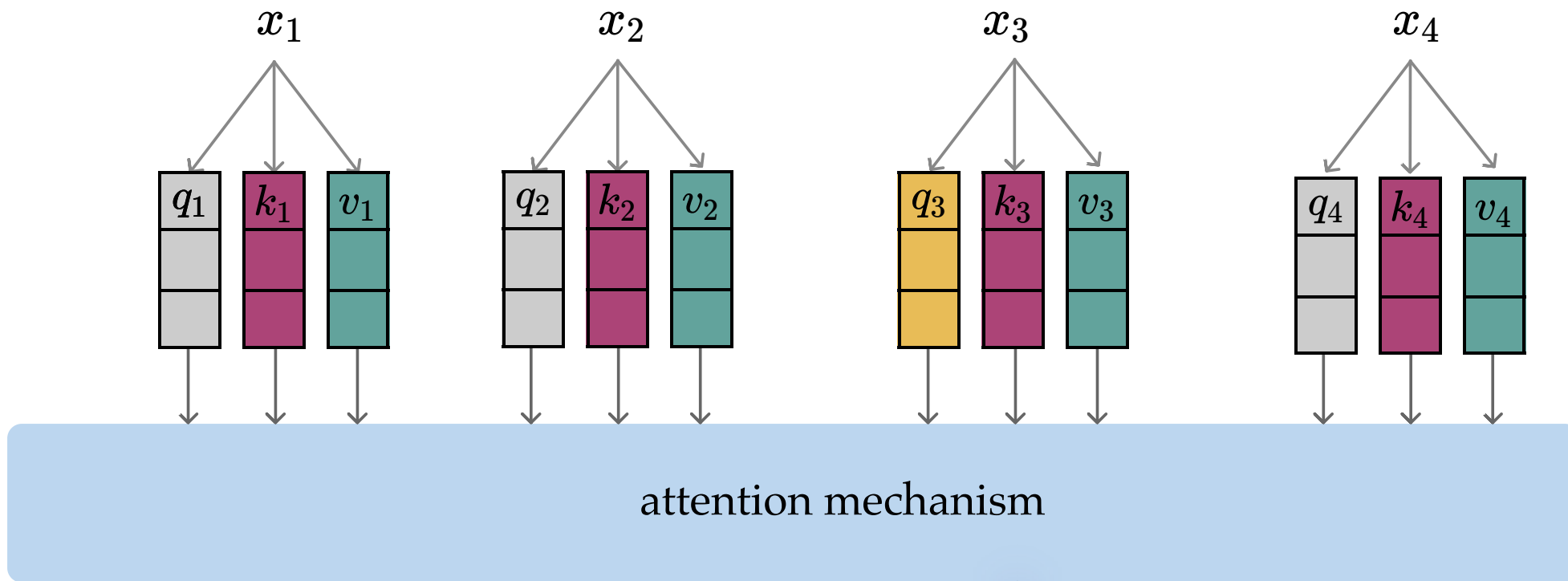




a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}



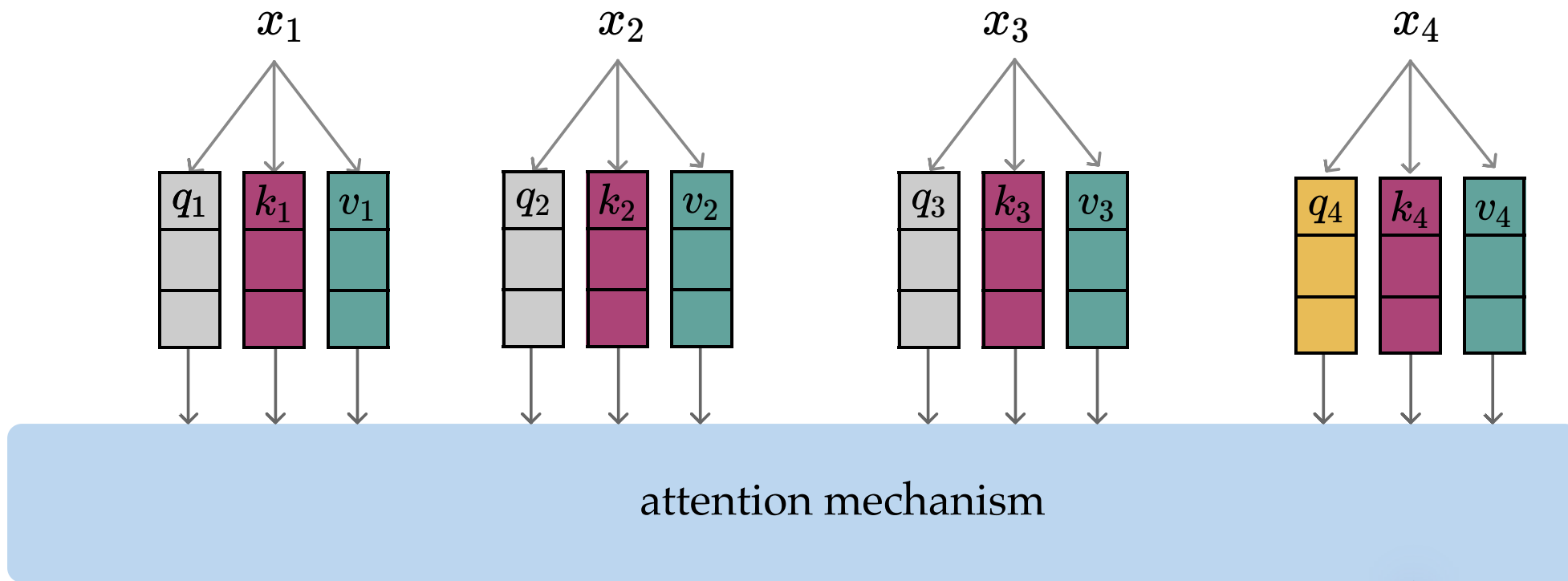




a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

The output of the attention mechanism is a weighted sum of the value vectors v_1, v_2, v_3, v_4 , where the weights are the attention scores $a_{31}, a_{32}, a_{33}, a_{34}$ (highlighted in orange in the table above). The result is a vector z_3 in \mathbb{R}^{d_k} .

$$z_3 = a_{31}v_1 + a_{32}v_2 + a_{33}v_3 + a_{34}v_4 \in \mathbb{R}^{d_k}$$



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}

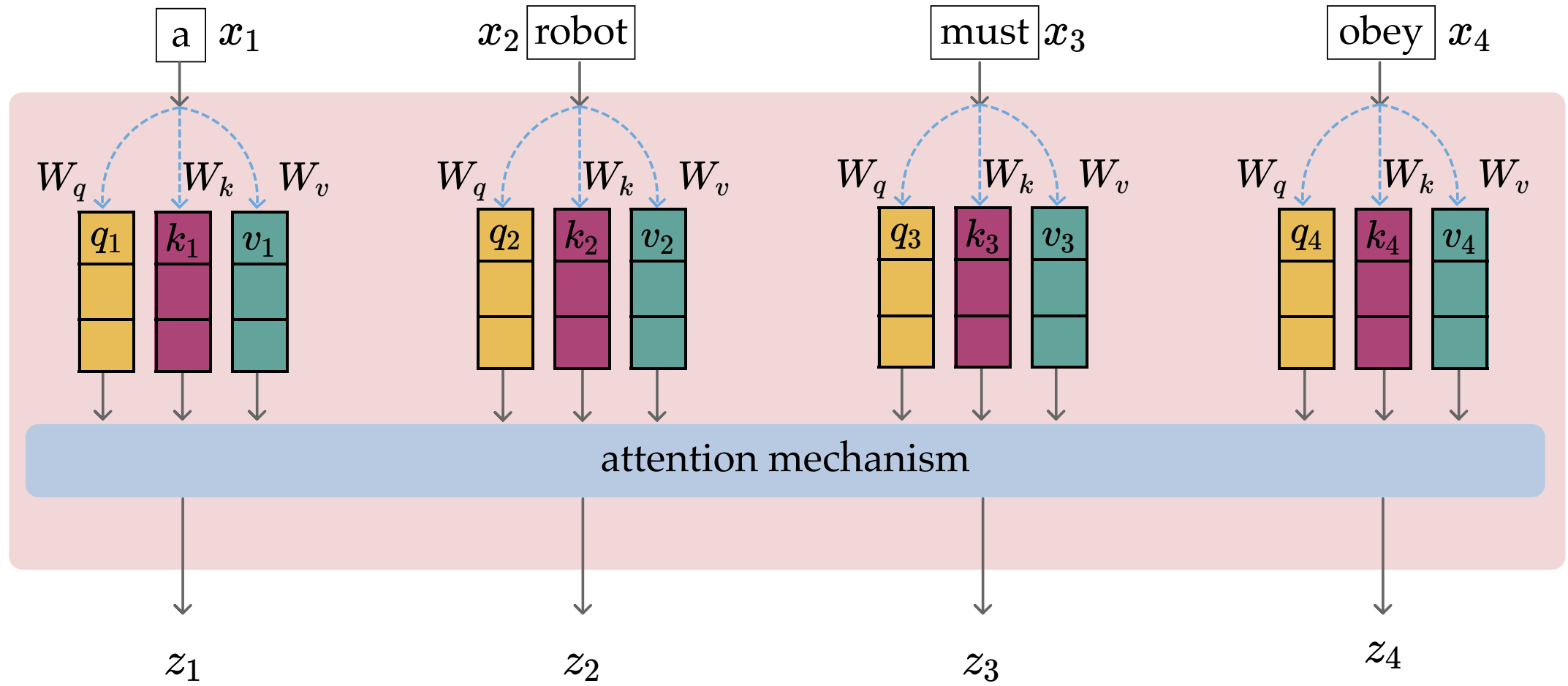
The diagram shows the calculation of the output z_4 for the fourth input. It is a weighted sum of the value vectors $v_1, v_2, v_3,$ and v_4 . The weights are the attention scores $a_{41}, a_{42}, a_{43},$ and a_{44} , which are shown in orange boxes. The result is z_4 , indicated by a dashed arrow from the output of the attention mechanism. The final result is shown to be in the space \mathbb{R}^{d_k} .

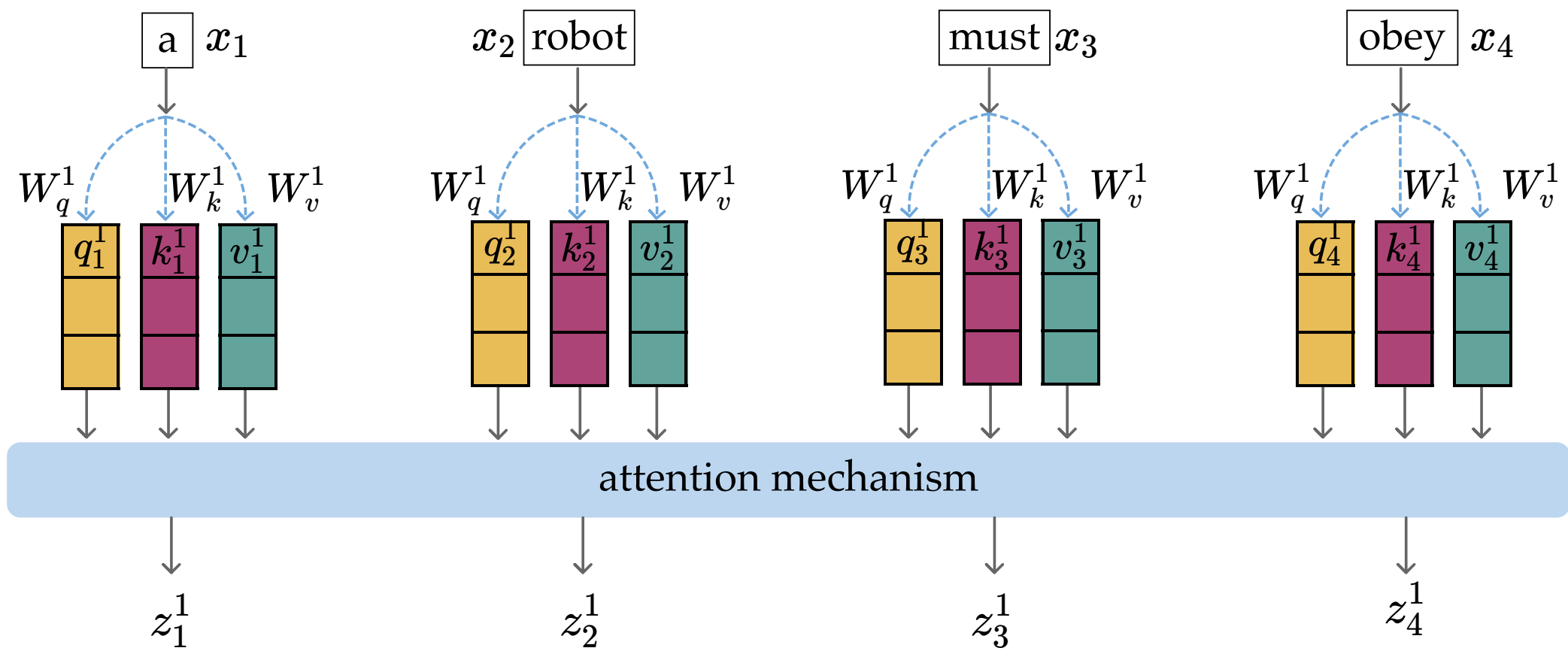
$$a_{41}v_1 + a_{42}v_2 + a_{43}v_3 + a_{44}v_4 = z_4 \in \mathbb{R}^{d_k}$$

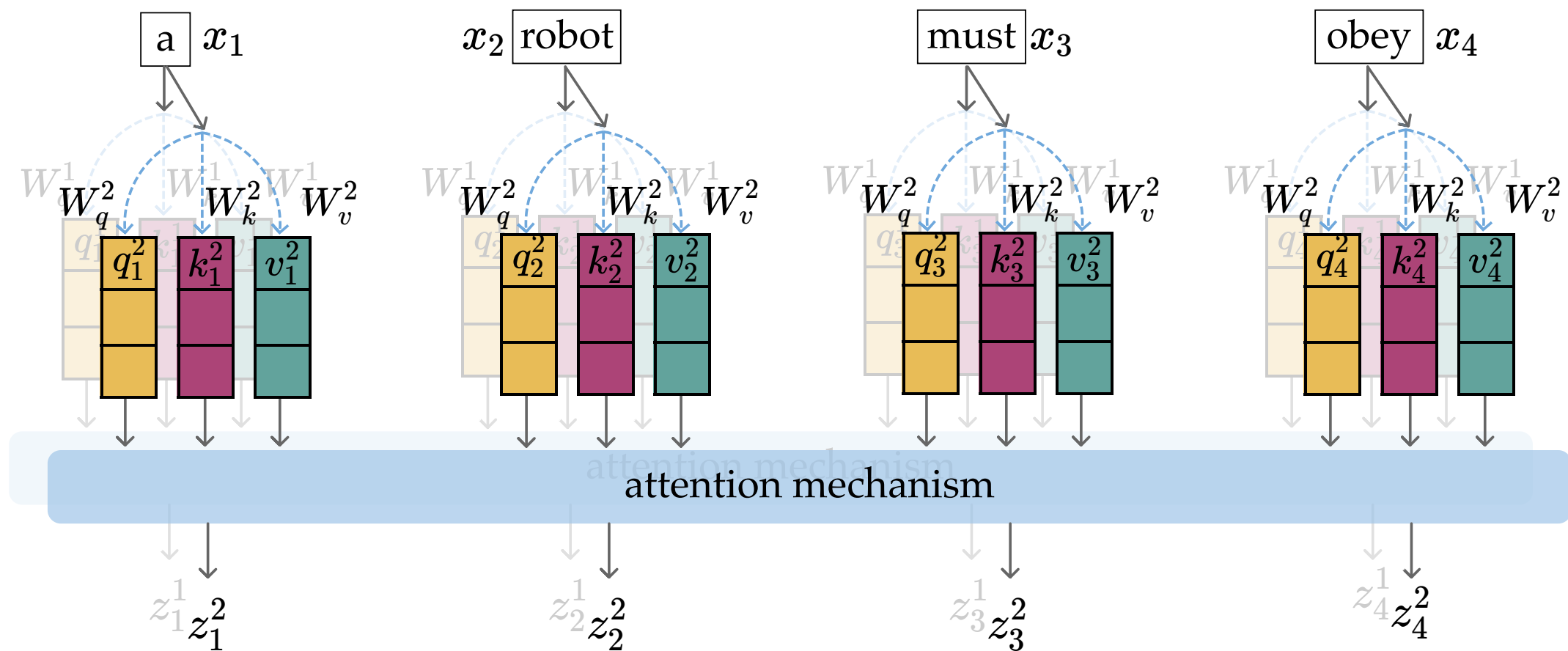
Outline

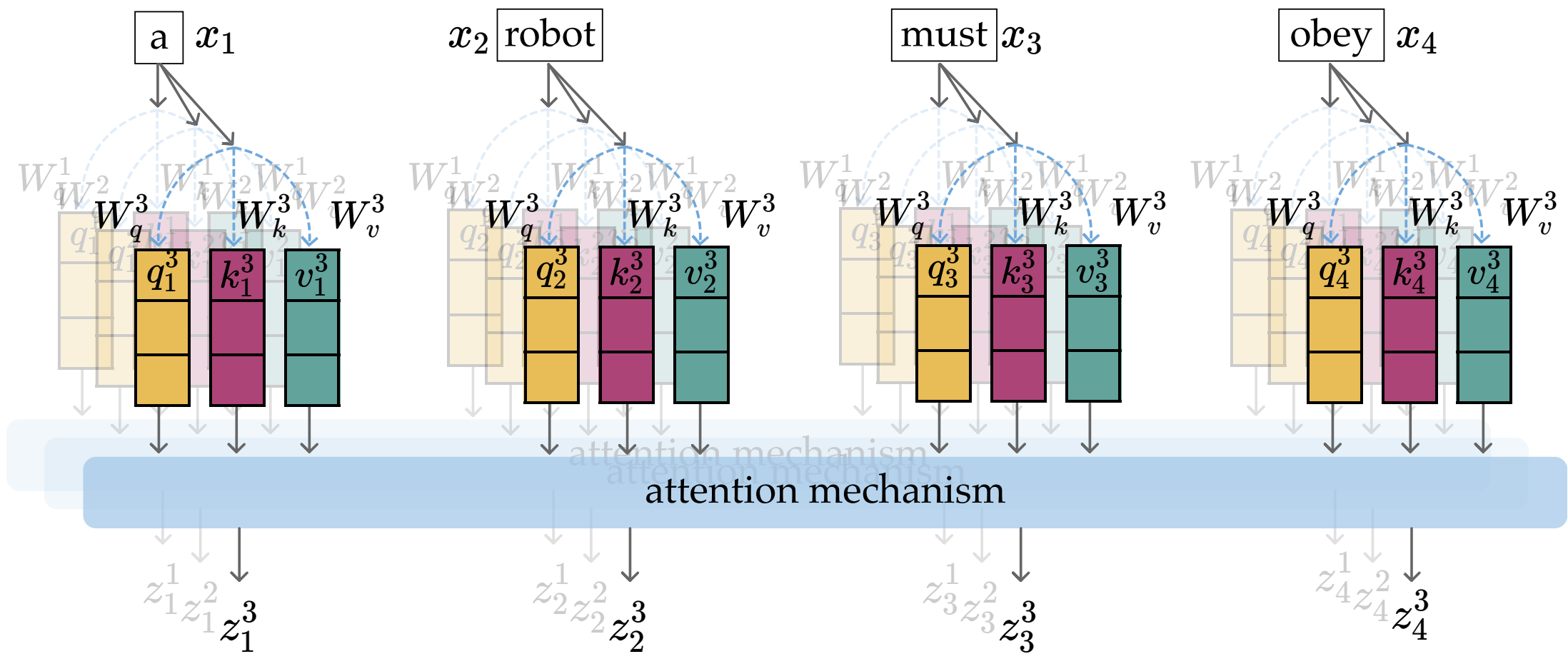
- Recap, embedding and representation
- Transformers high-level intuition
- Transformers architecture overview
- (query, key, value) and self-attention
 - matrix form
- Multi-head Attention
- (Applications)

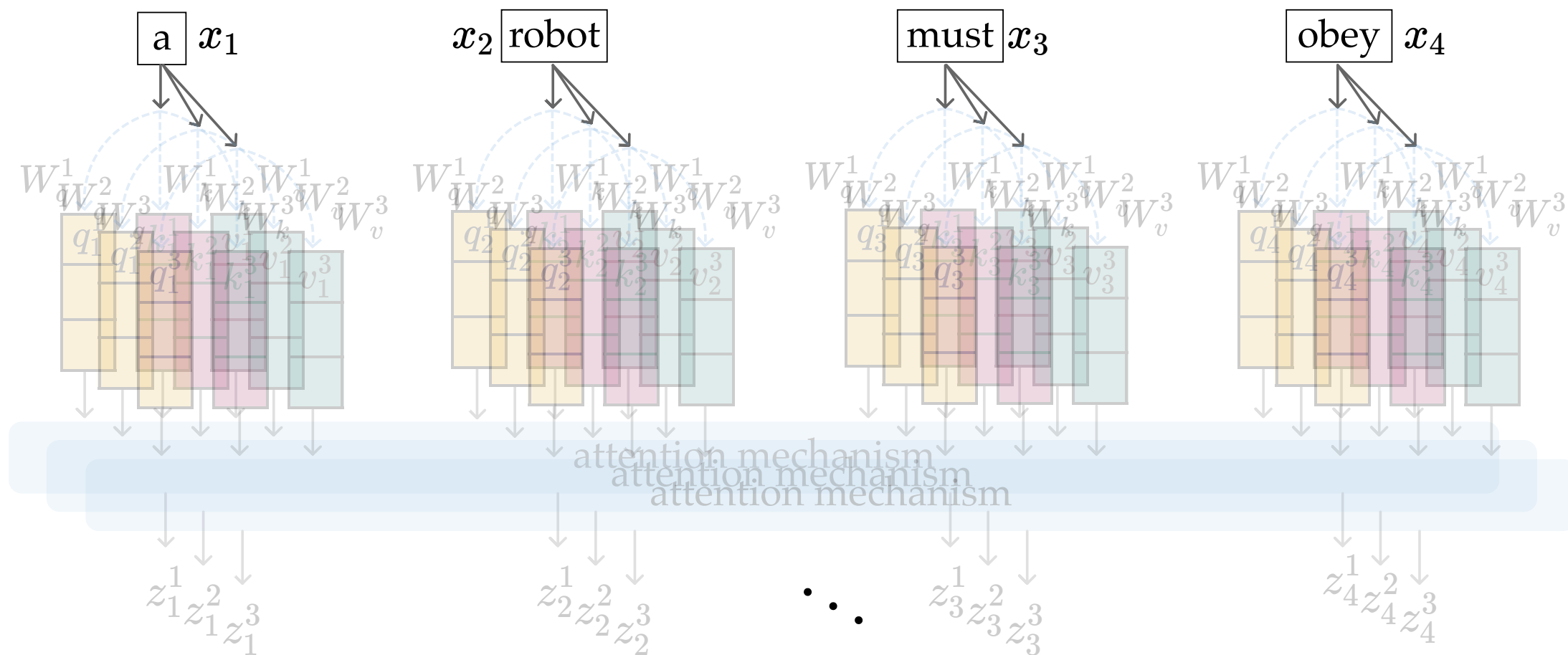
one attention head

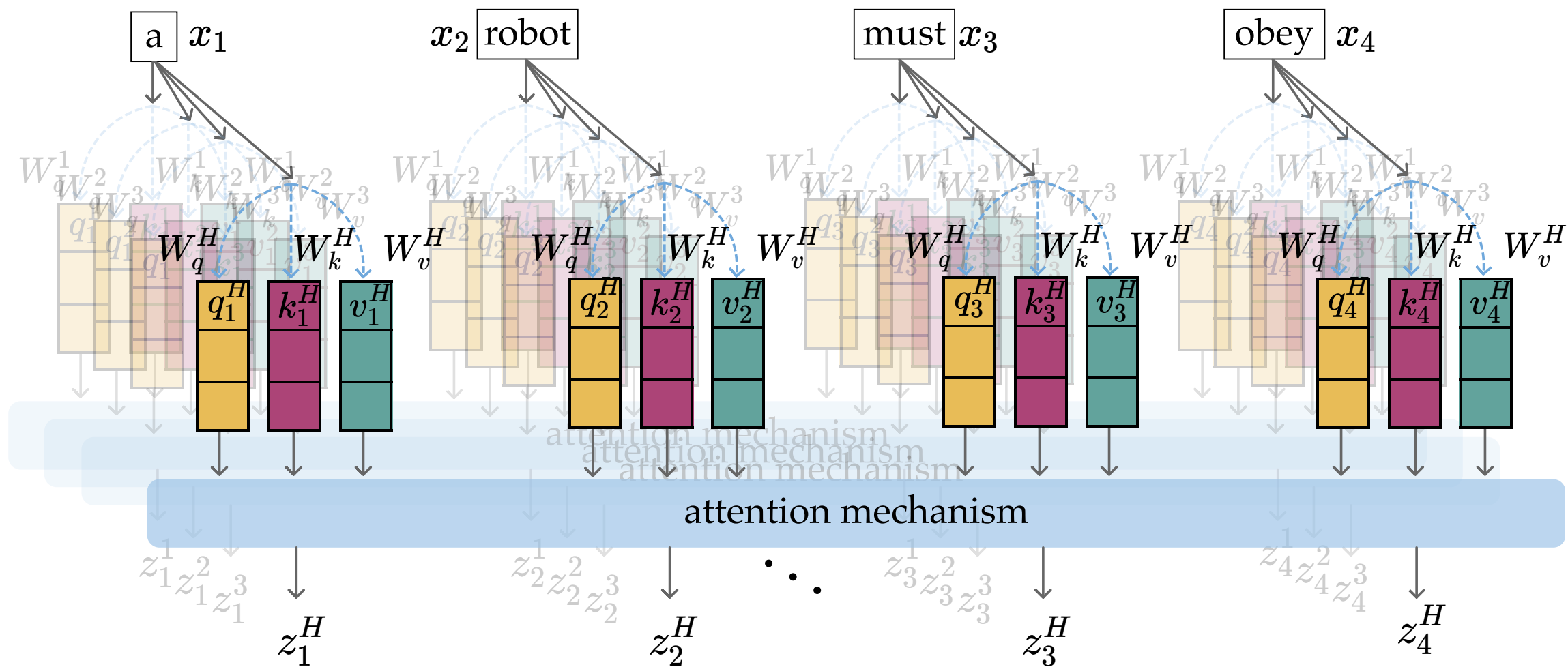










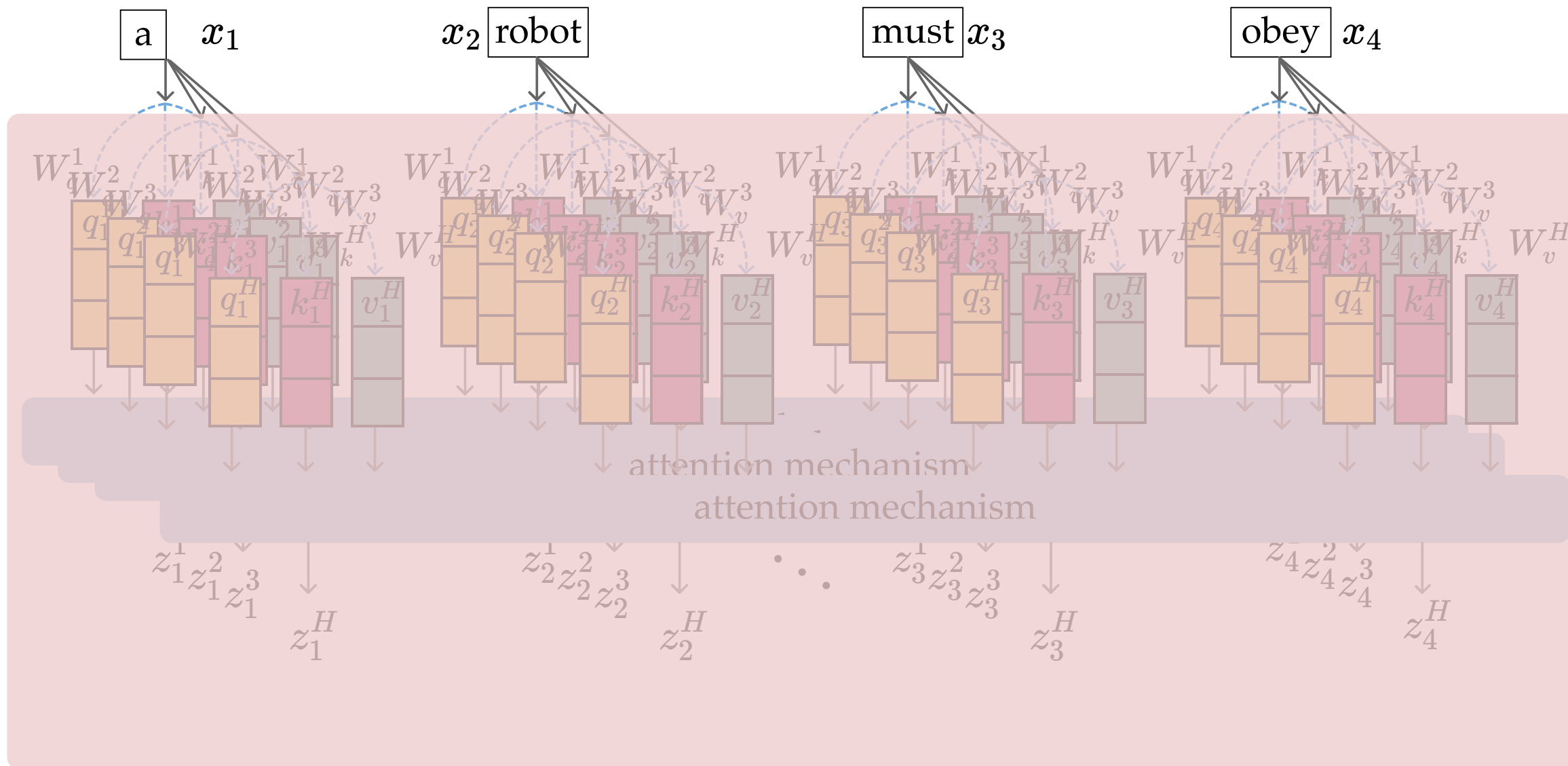


Each attention head

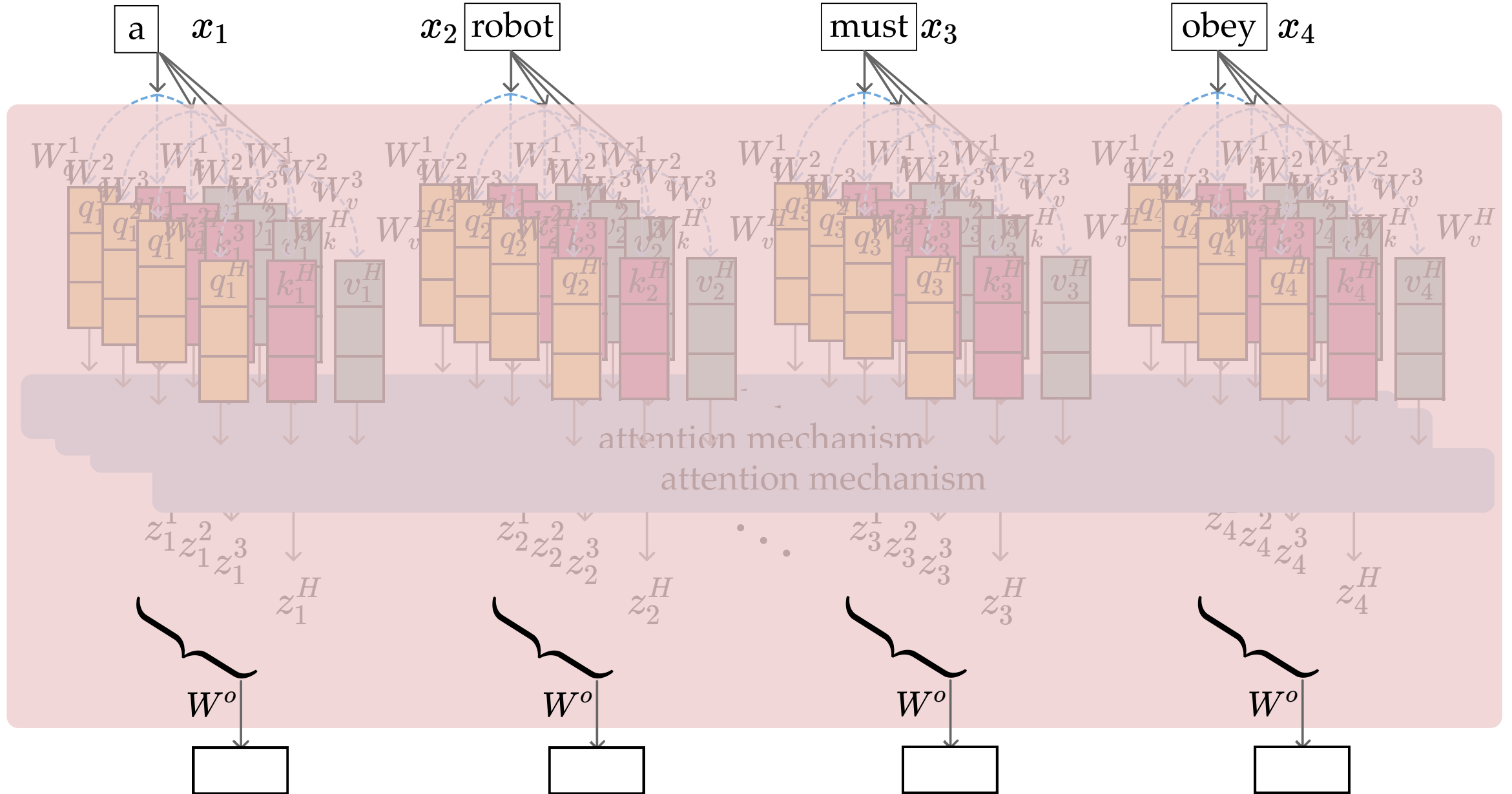
- can be processed independently and in parallel with all other heads,
- learns its own set of W_q, W_k, W_v ,
- creates its own projected (q, k, v) tokens,
- computes its own attention outputs independently,
- processes the sequence of n tokens simultaneously and in parallel

independent, parallel, and structurally identical processing across all heads and tokens.

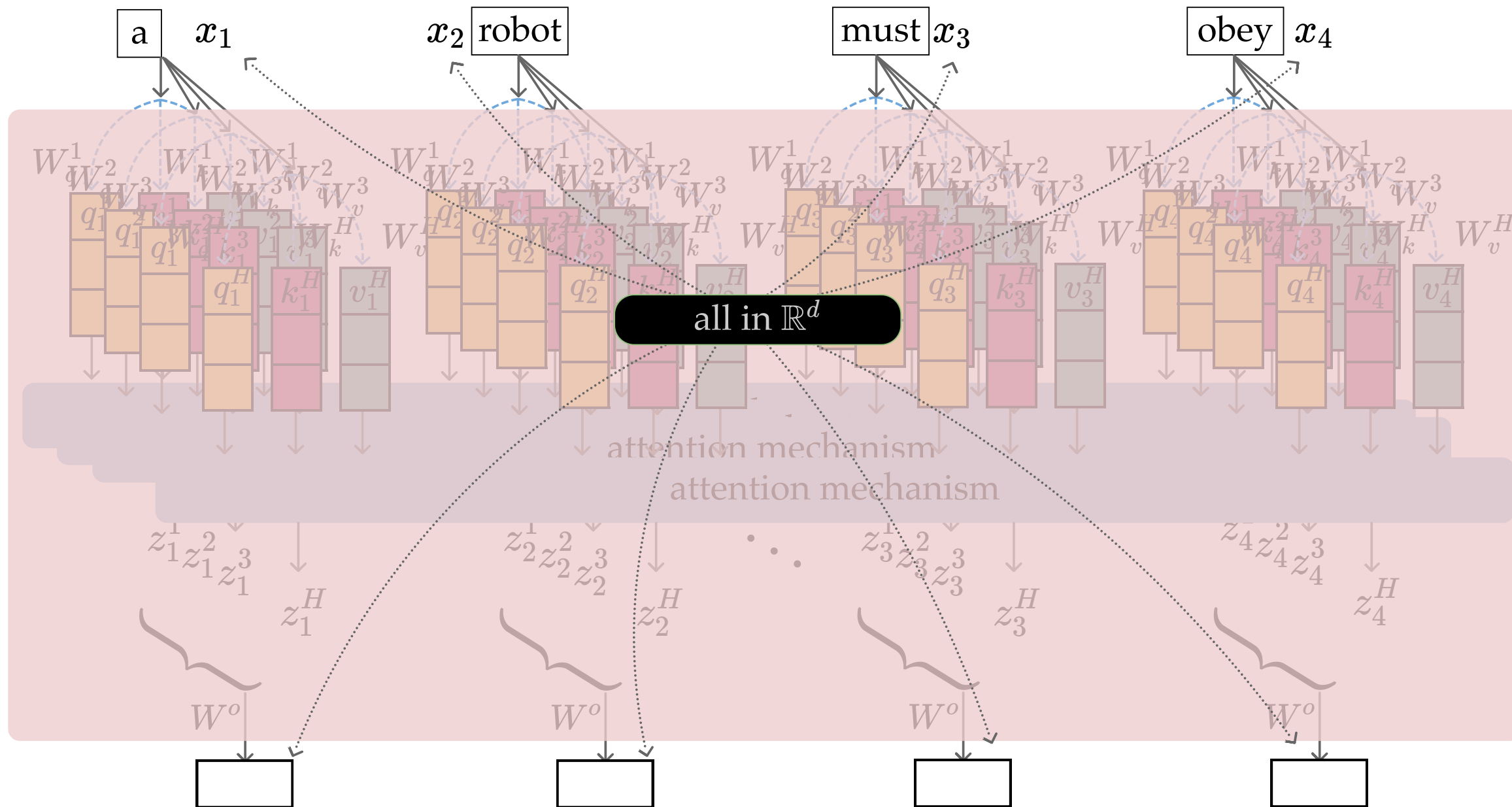
multi-head attention



multi-head attention



multi-head attention



Shape Example

n	num tokens	2
d	token dim	4
d_k	(qkv) dim	3
H	num heads	5

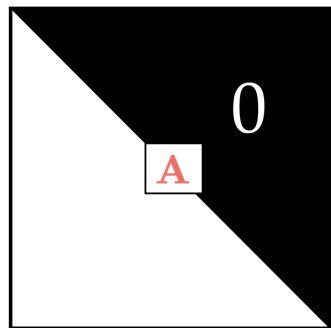
Shape Example

n	num tokens	2
d	token dim	4
d_k	(qkv) dim	3
H	num heads	5

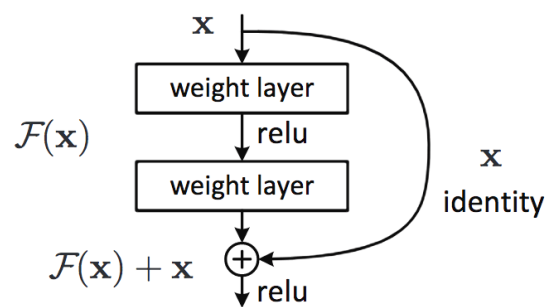
W_q^h	query proj	$d \times d_k$	4×3
W_k^h	key proj	$d \times d_k$	4×3
W_v^h	value proj	$d \times d_k$	4×3
W^o	output proj	$d \times Hd_k$	4×15
input	-	$n \times d$	2×4
q^h	query	$n \times d_k$	2×3
k^h	key	$n \times d_k$	2×3
v^h	value	$n \times d_k$	2×3
A^h	attn matrix	$n \times n$	2×2
z^h	head out.	$n \times d_k$	2×3
output	-	$n \times d$	2×4

learned

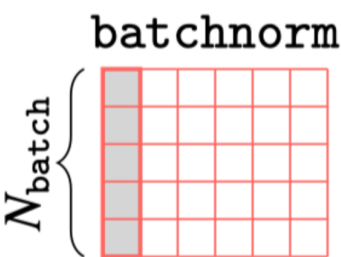
Some practical techniques commonly needed when training auto-regressive transformers:



masking



Residual connection




Layer normlization



Positional encoding

<https://jiachenzhu.github.io/DyT/>

<https://poloclub.github.io/transformer-explainer/>

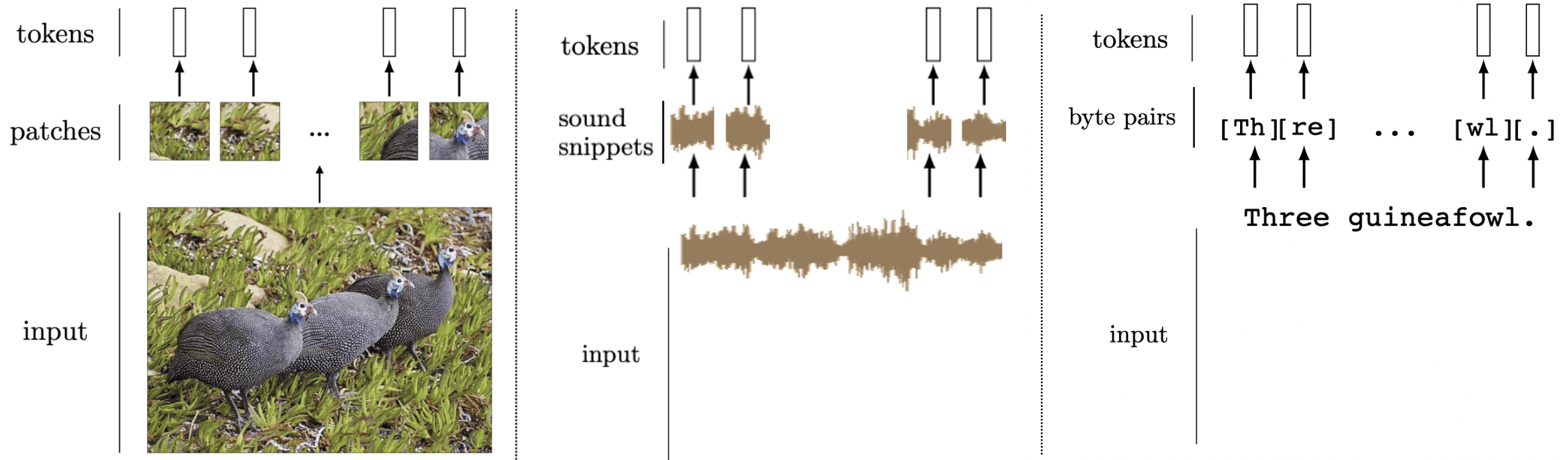


applications / comments

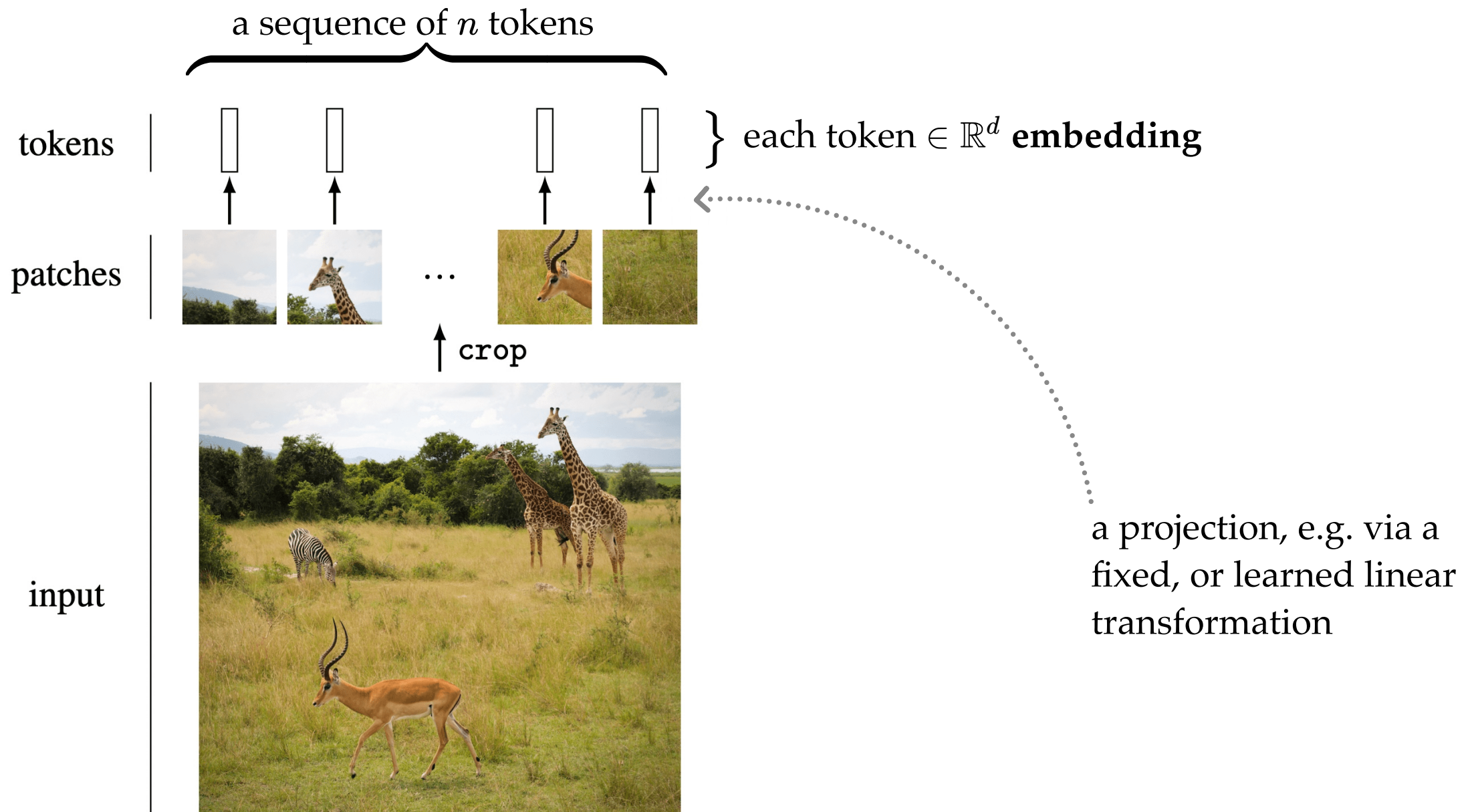
We can tokenize anything.

General strategy: chop the input up into chunks, *project* each chunk to an **embedding**

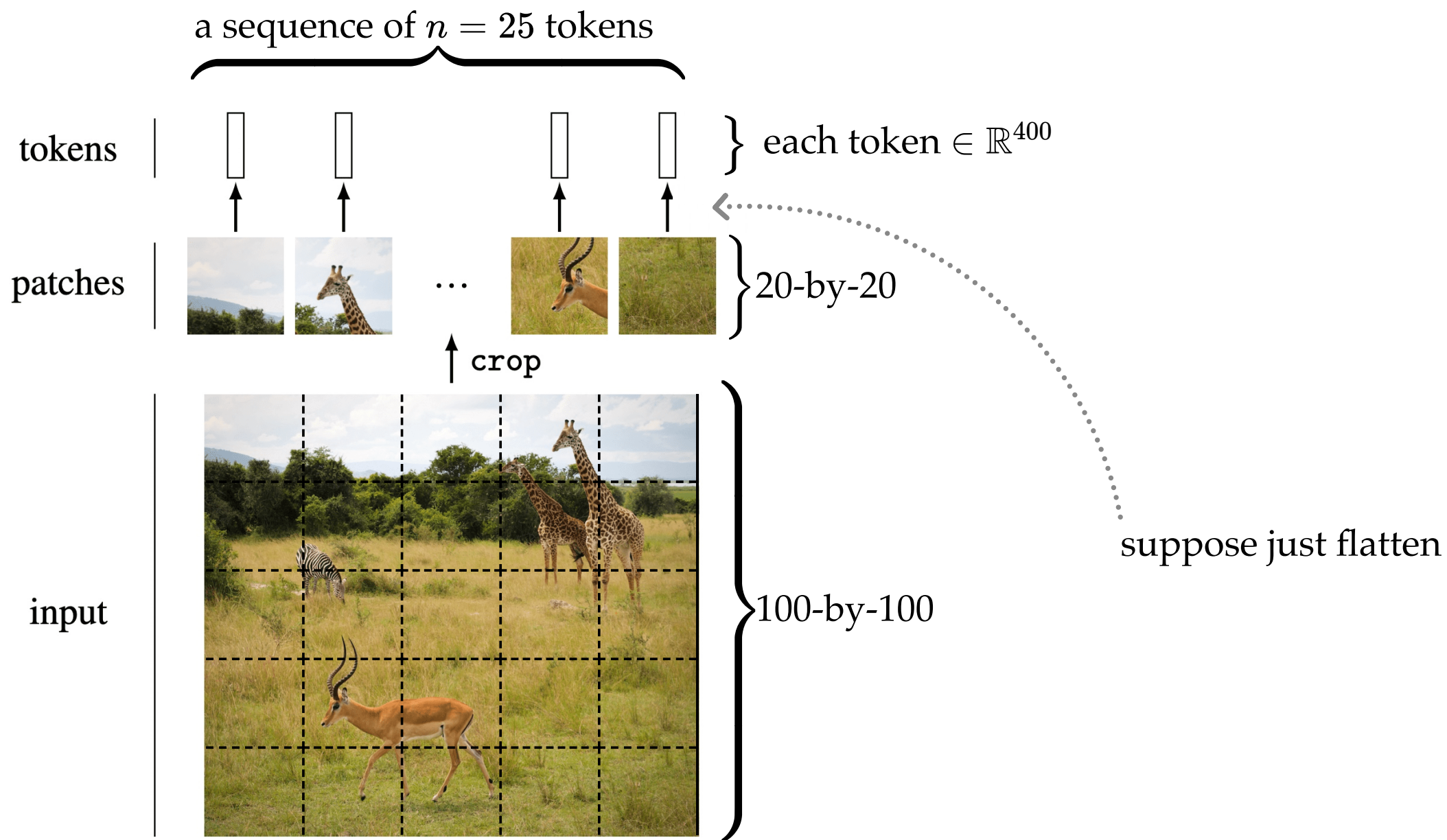
this projection can be fixed from a pre-trained model, or trained jointly with downstream task



[images credit: visionbook.mit.edu]

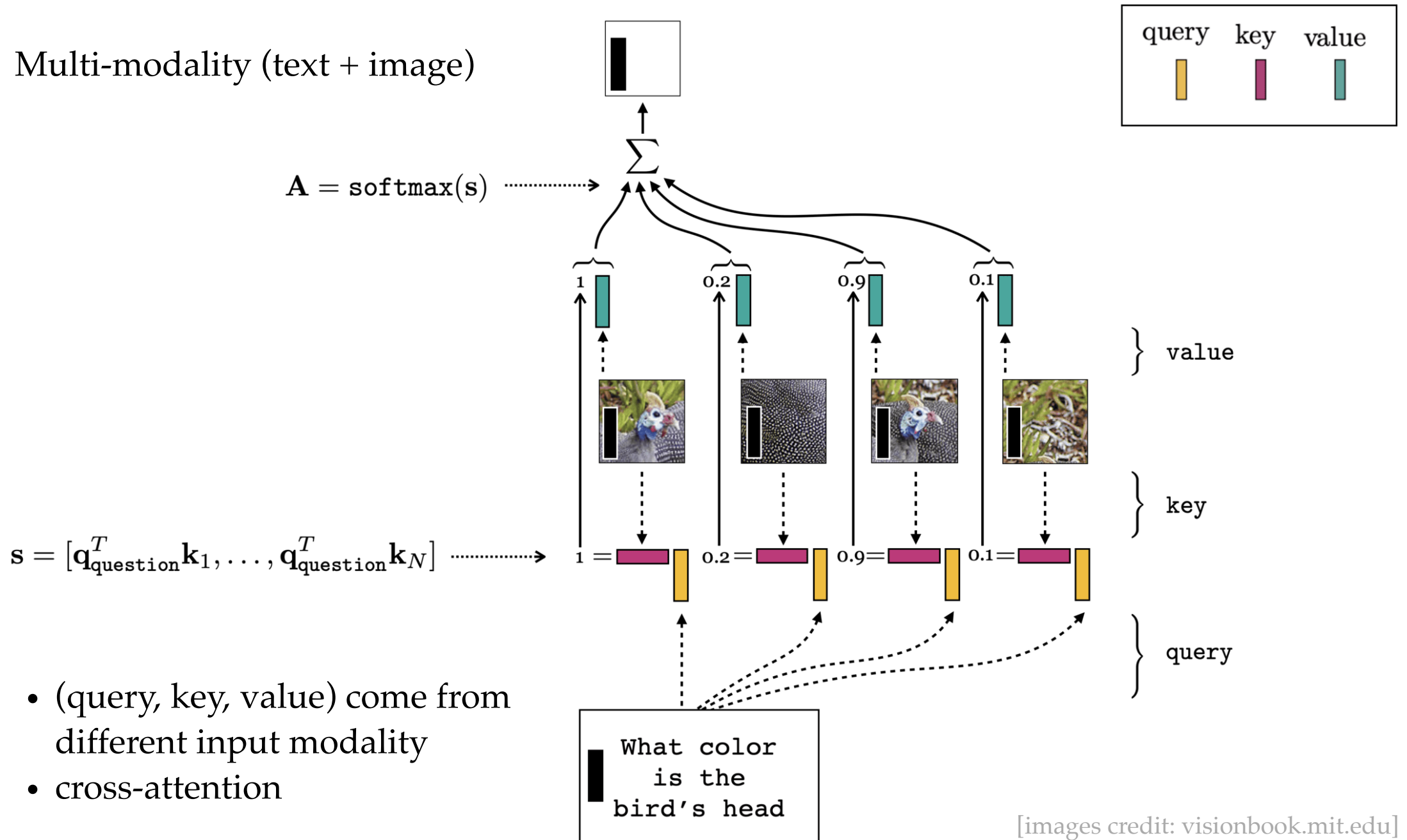


[images credit: visionbook.mit.edu]



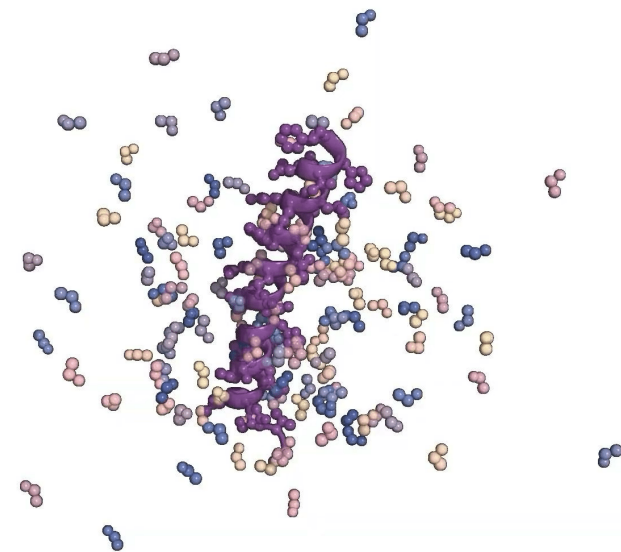
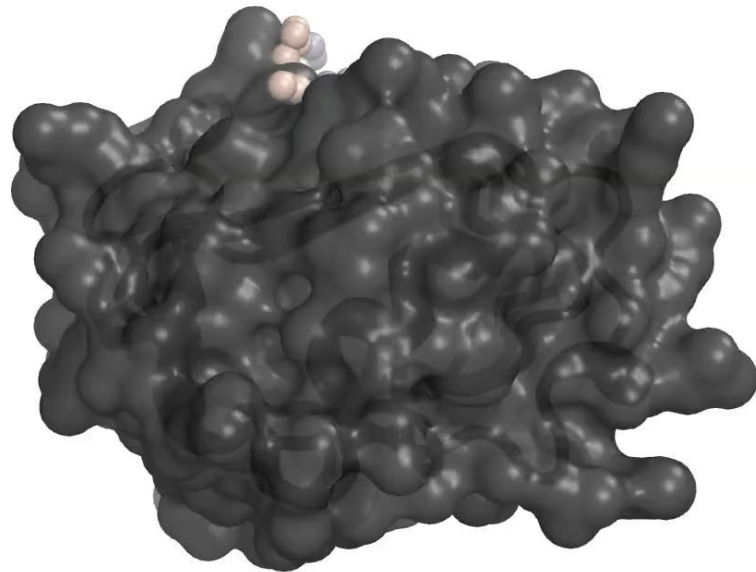
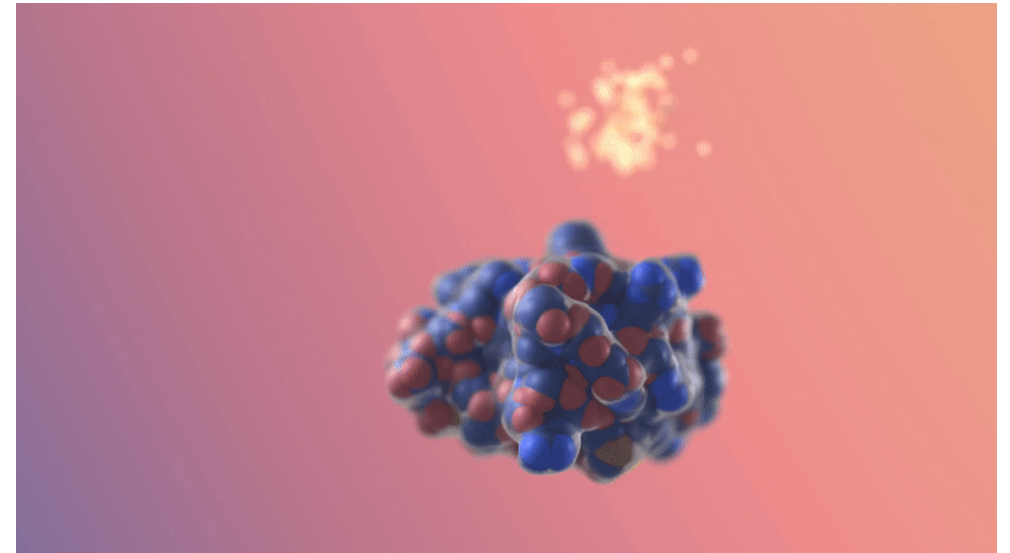
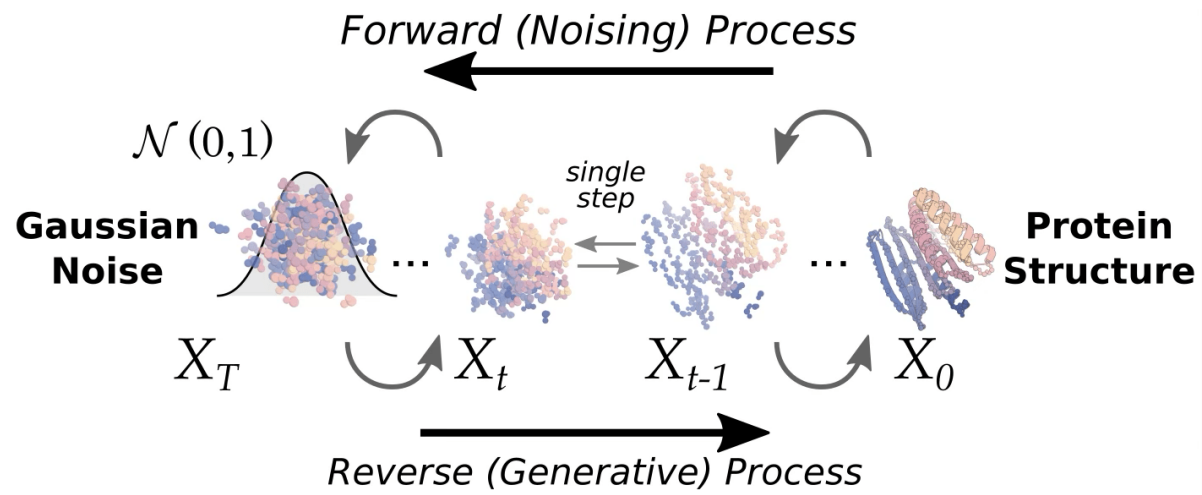
[images credit: visionbook.mit.edu]

Multi-modality (text + image)



<https://segment-anything.com/demo>

<https://dreamfusion3d.github.io>



Image/video credit: RFDiffusion <https://www.bakerlab.org>

<https://ml-gsai.github.io/LLaDA-demo/>

Success mode:



["DINO", Caron et al. 2021]

Success mode:



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.

[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)]

Failure mode:



A woman is sitting at a table
with a large pizza.



A person is standing on a beach
with a surfboard.

[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)]

Failure mode:



A large white bird standing in a forest.



A woman holding a clock in her hand.

[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)]

Failure mode:



Giannis Daras 🦋 NeurIPS 2023

@giannis_daras

...

DALLE-2 has a secret language.

"Apoploe vesrreaitais" means birds.

"Contarra cctnxiams luryca tanniounons" means bugs or pests.

The prompt: "Apoploe vesrreaitais eating Contarra cctnxiams luryca tanniounons" gives images of birds eating bugs.

A thread (1/n) 🧵



Another example: "Two whales talking about food, with subtitles". We get an image with the text "Wa ch zod rea" written on it. Apparently, the whales are actually talking about their food in the DALLE-2 language. (4/n)



Figure 4: Left: Image generated with the prompt: "Two whales talking about food, with subtitles.". Right: Images generated with the prompt: "Wa ch zod ahaakes rea.". The gibberish language, "Wa ch zod ahaakes rea.", produces images that are related to the text-conditioning and the visual output of the first image.

)

Summary

- Transformers combine many of the best ideas from earlier architectures—convolutional patch-wise processing, relu nonlinearities, residual connections —with several new innovations, in particular, embedding and attention layers.
- Transformers start with some generic hard-coded **embeddings**, and layer-by-layer, creates better and better embeddings.
- Parallel processing everything in **attention**: each head is processed in parallel, and within each head, the q, k, v token sequence is created in parallel, the attention scores is computed in parallel, and the attention output is computed in parallel.

<https://forms.gle/htC97GQJmsNDWB8ZA>

We'd love to hear
your **thoughts**.

Thanks!

for your *attention!*