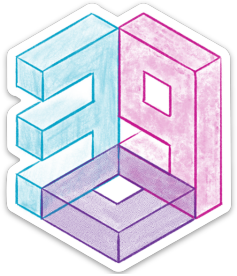


https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 9: Transformers

Shen Shen

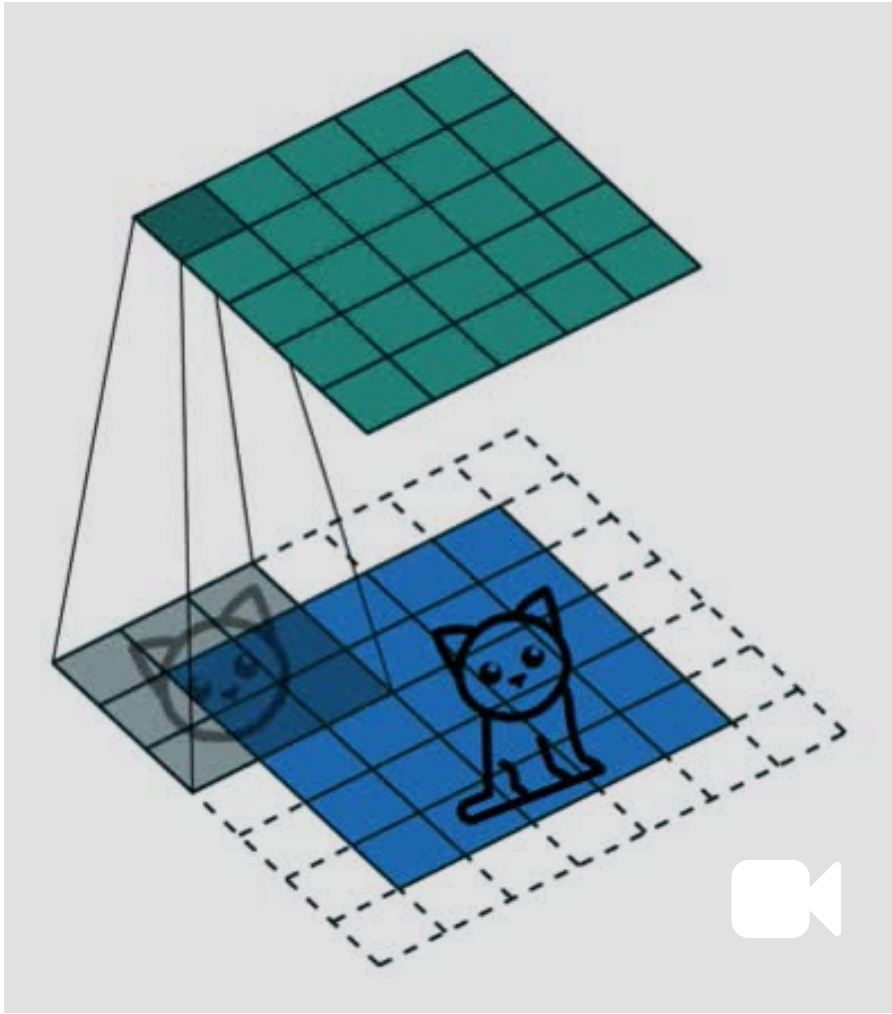
November 1, 2024

Outline

- Recap, convolutional neural networks
- Transformers example use case: large language models
- Transformers key ideas:
  - Embedding
  - Attention mechanism
- (Applications and case studies)

Recap: convolution

- Looking locally
- Parameter sharing
- Template matching
- Translational equivariance



[video credit Lena Voita]

Enduring principles:

1. Chop up signal into patches (divide and conquer)
2. Process each patch **independently and identically** (and in **parallel**)

[image credit: Fredo Durand]

Enduring principles:

1. Chop up signal into patches (divide and conquer)
2. Process each patch **independently, identically,** and in **parallel**

Transformers follow similar principles:

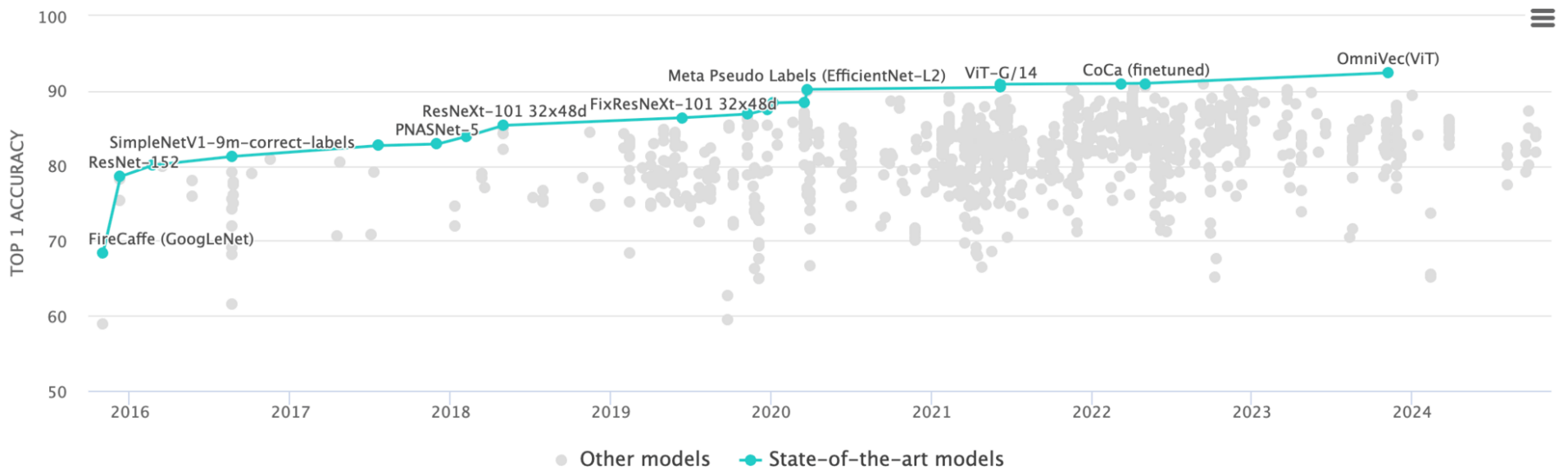
1. Chop up signal into patches (divide and conquer)
2. Process each patch **identically**, and in **parallel**

but not independently; each patch's processing **depends** on all other patches, allowing us to take into account the full context.

Image Classification on ImageNet

Leaderboard Dataset

View Top 1 Accuracy by Date for All models



Filter:
ImageNet-1k only | Transformer | ResNet | CNN | ImageNet-22k | EfficientNet | JFT-300M | MLP | ResNeXt | Reversible
Neighborhood Attention | NAT Transformer | JFT-3B | PatchConvnet | FPN | MoE | Early Exit | Dynamic Model Arch
CNN+Transformer | ALIGN | Conv+Transformer | CLIP data | No Extra Data | SNN | IG-1B | Swin-Transformer | Teacher-22k
Vision Transformer | FLD-900M | Pure CNN | YFCC-15M | Laion-400M | Contrastive | ConvNeXt | Self-Supervised Learning
RegNet | Mixer | Memory-Centric | CLIP Pre-trained | CrossCovarianceAttention | untagged | Hardware Burden
Operations per network pass | Robustness reports

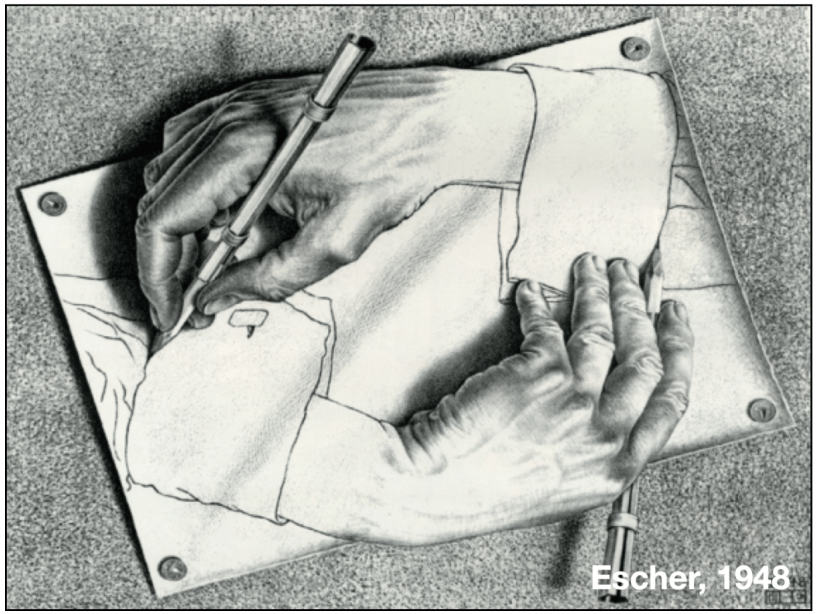
Edit Leaderboard

Paperswithcode

Outline

- Recap, convolutional neural networks
- Transformers example use case: large language models
- Transformers key ideas:
  - Embedding
  - Attention mechanism
- (Applications and case studies)

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

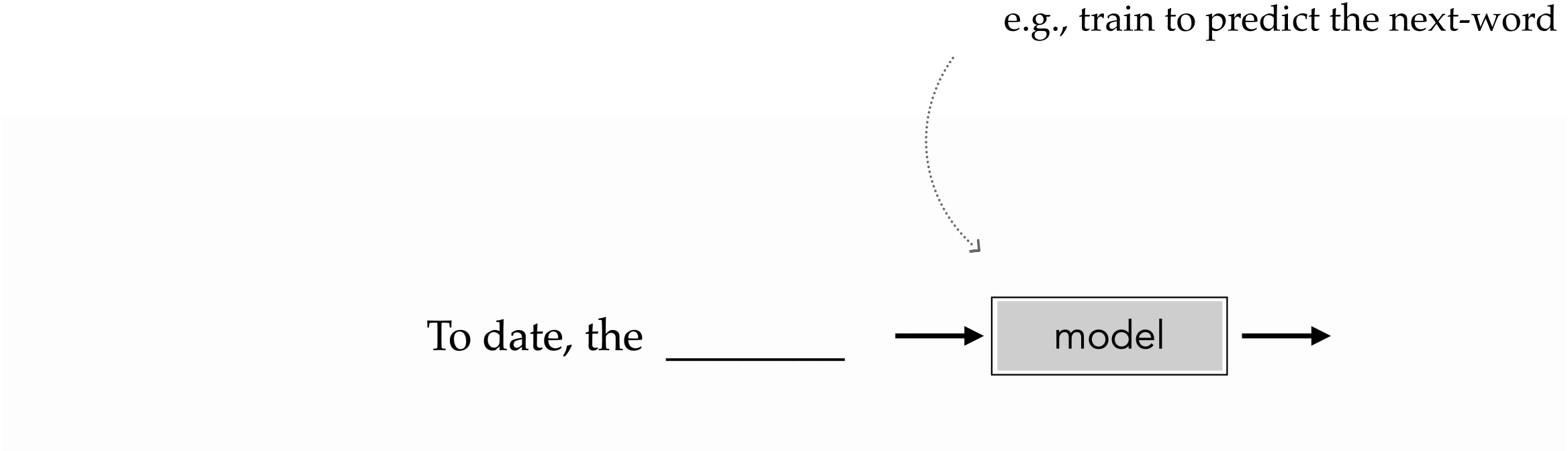


- Scrape the internet for unlabeled plain texts.
- Cook up “labels” (prediction targets) from the unlabeled texts.
- Convert “unsupervised” problem into “supervised” setup.

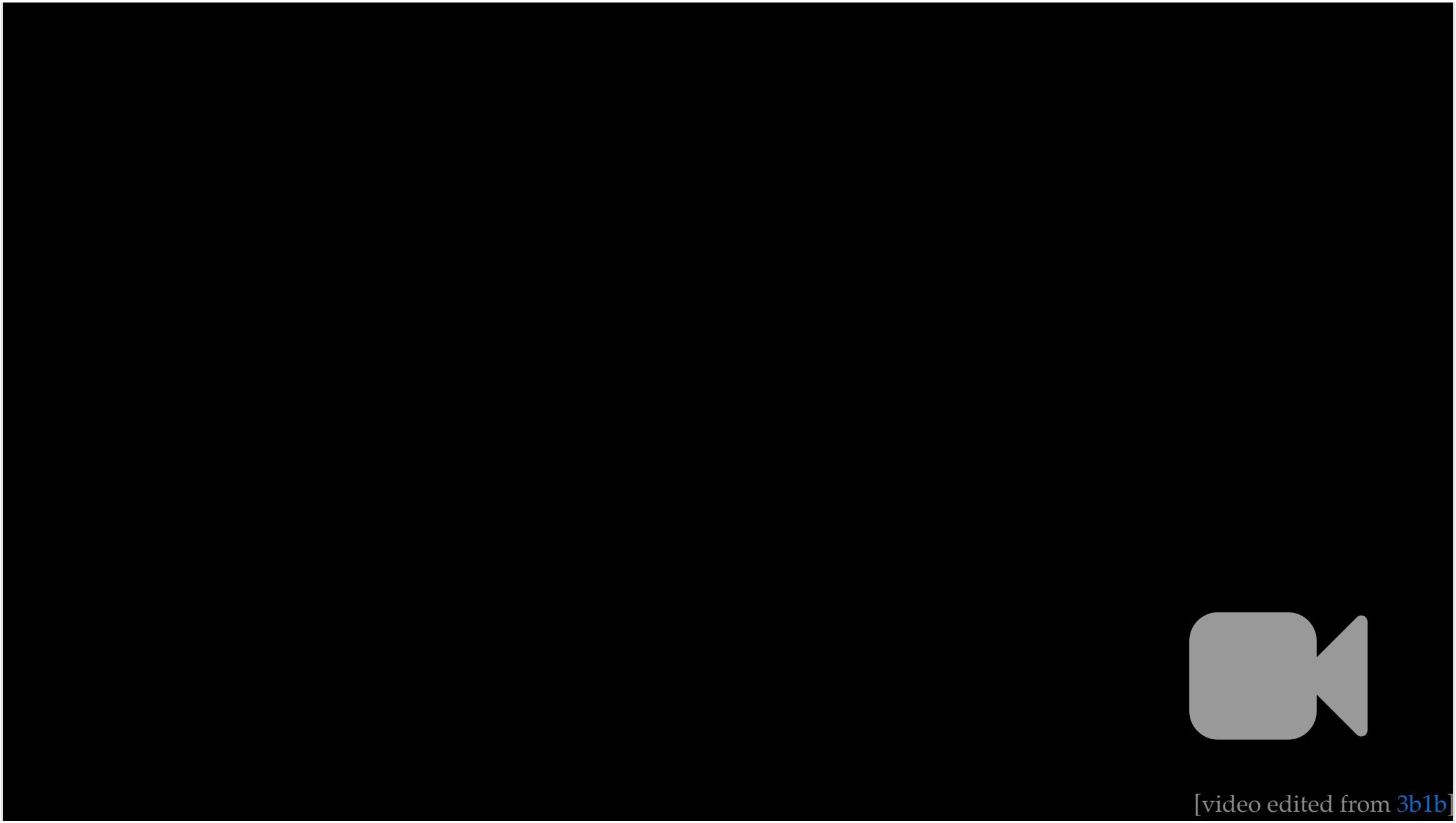
"To date, the cleverest thinker of all time was Issac. "

feature	label
⋮	⋮
To date, the	cleverest
To date, the cleverest	thinker
To date, the cleverest thinker	was
⋮	⋮
To date, the cleverest thinker of all time was	Issac

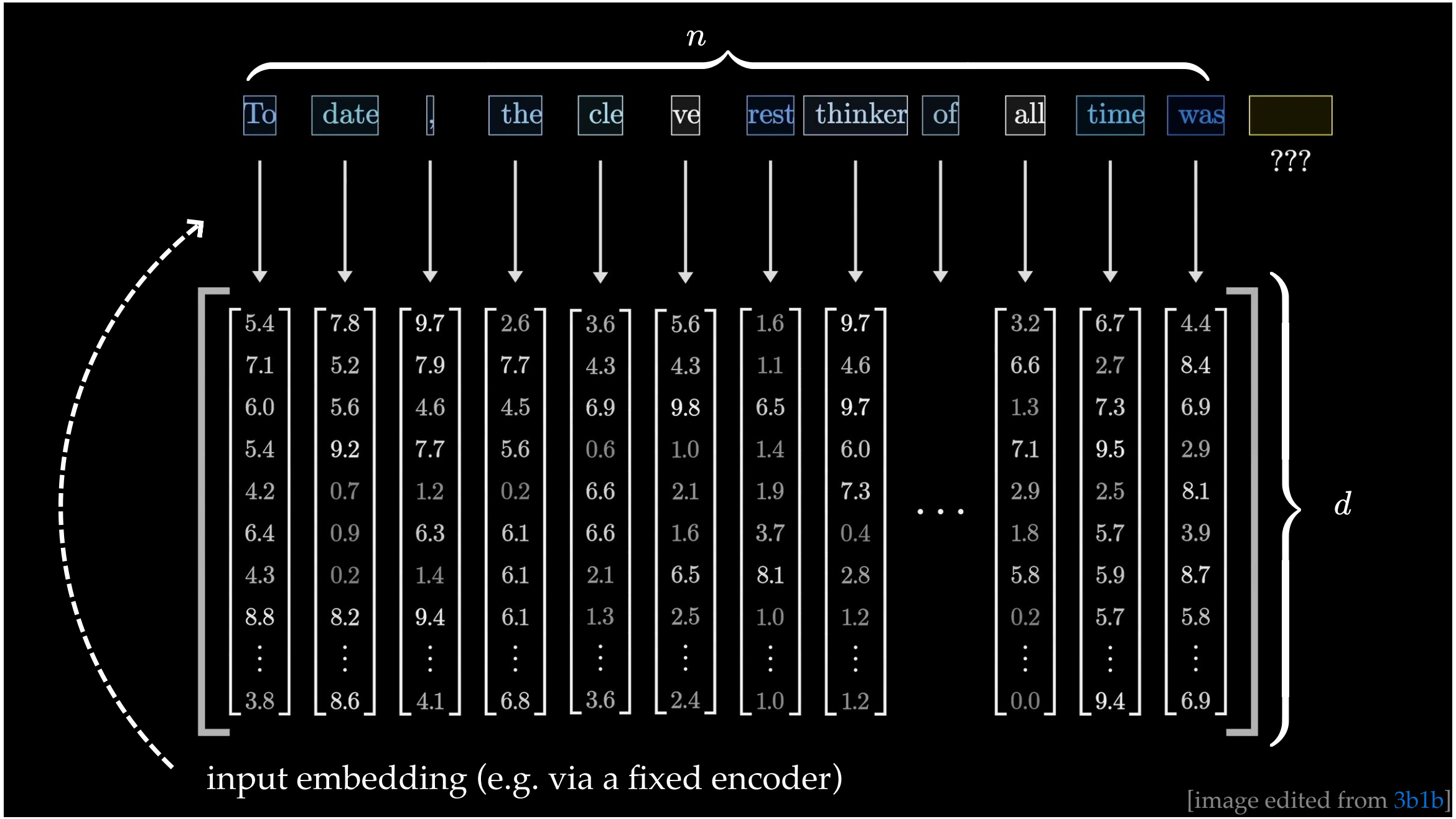
Auto-regressive



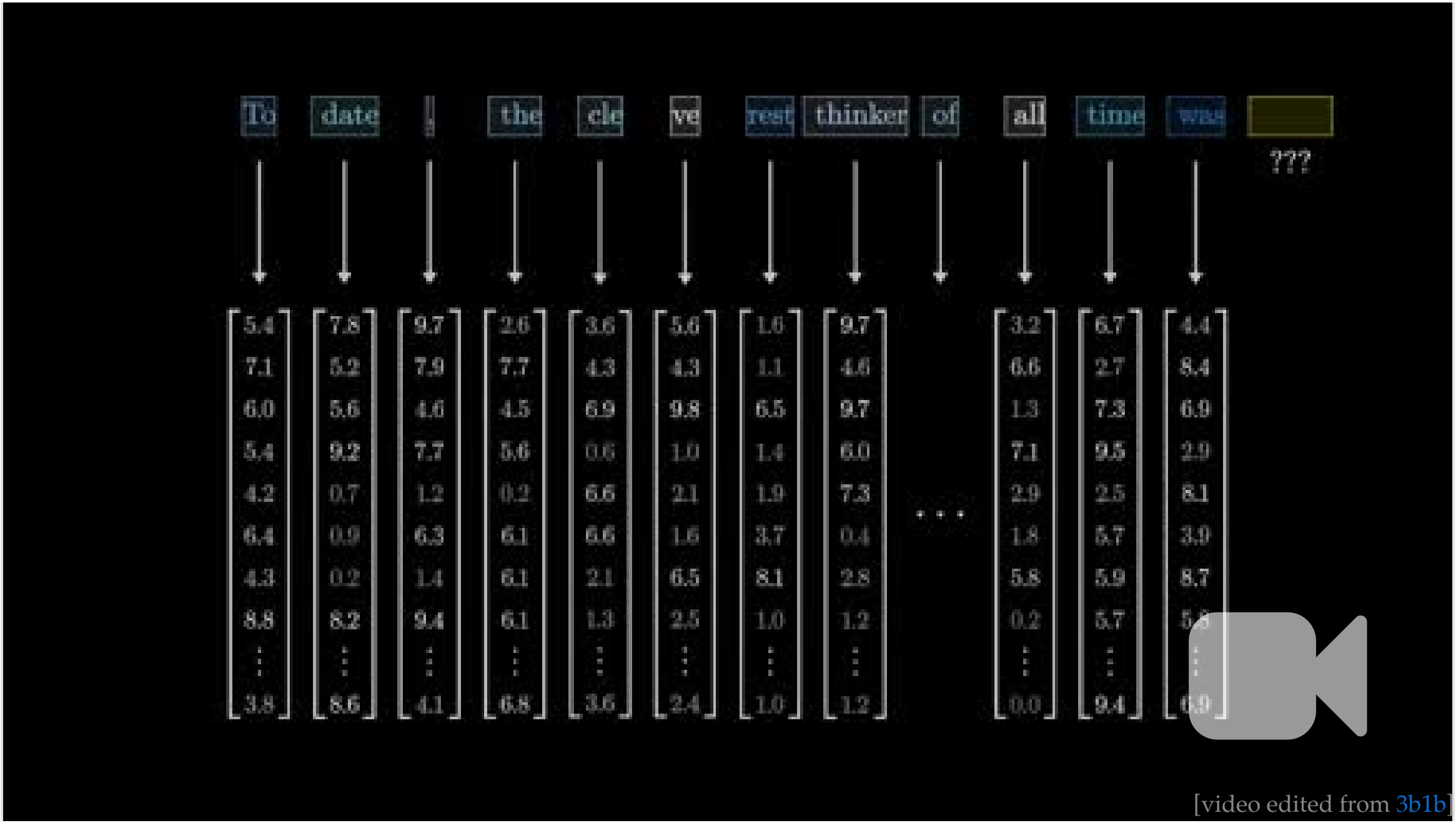
For LLMs (and many other applications), the model used are transformers



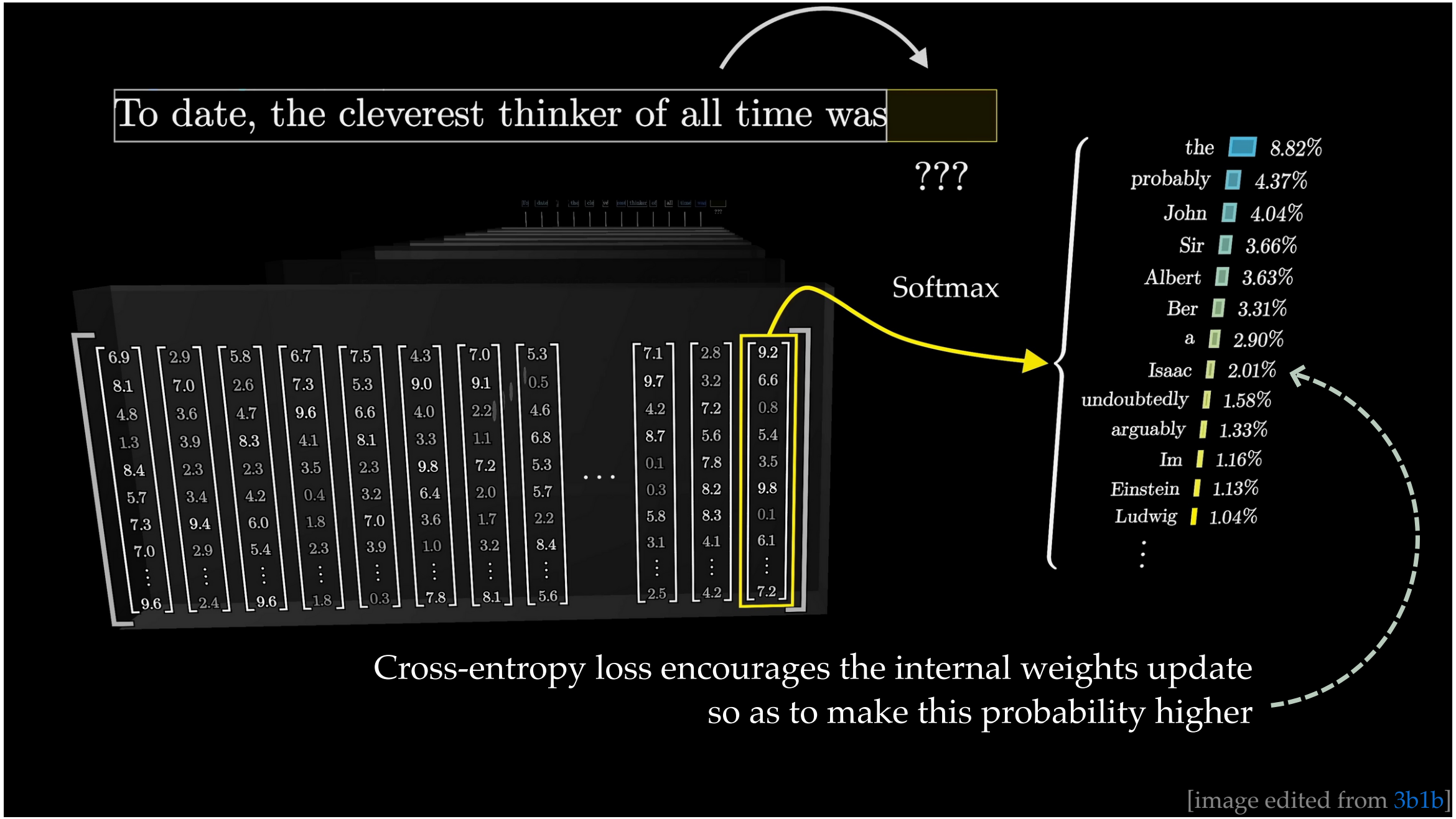
[video edited from 3b1b]



[image edited from 3b1b]



[video edited from 3b1b]

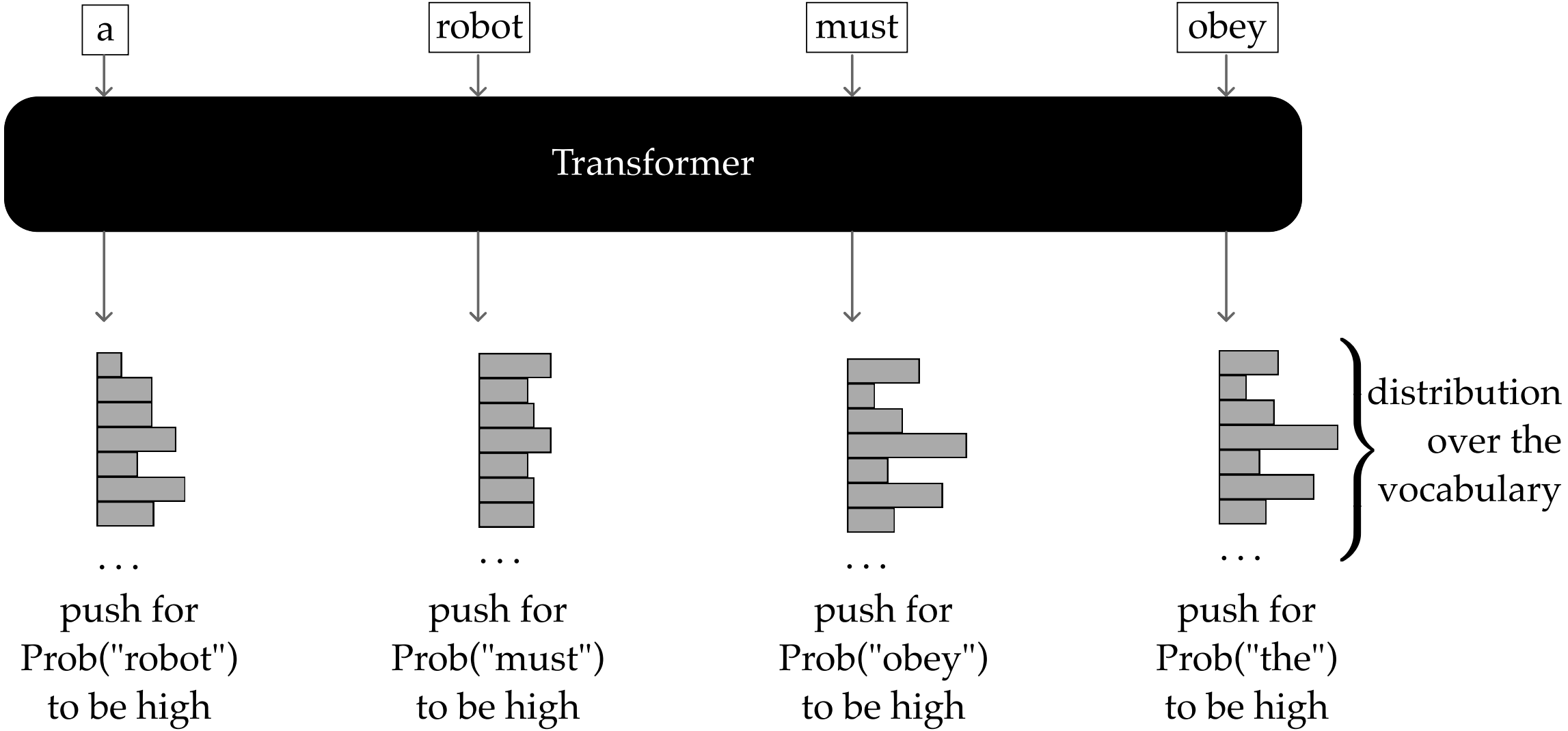


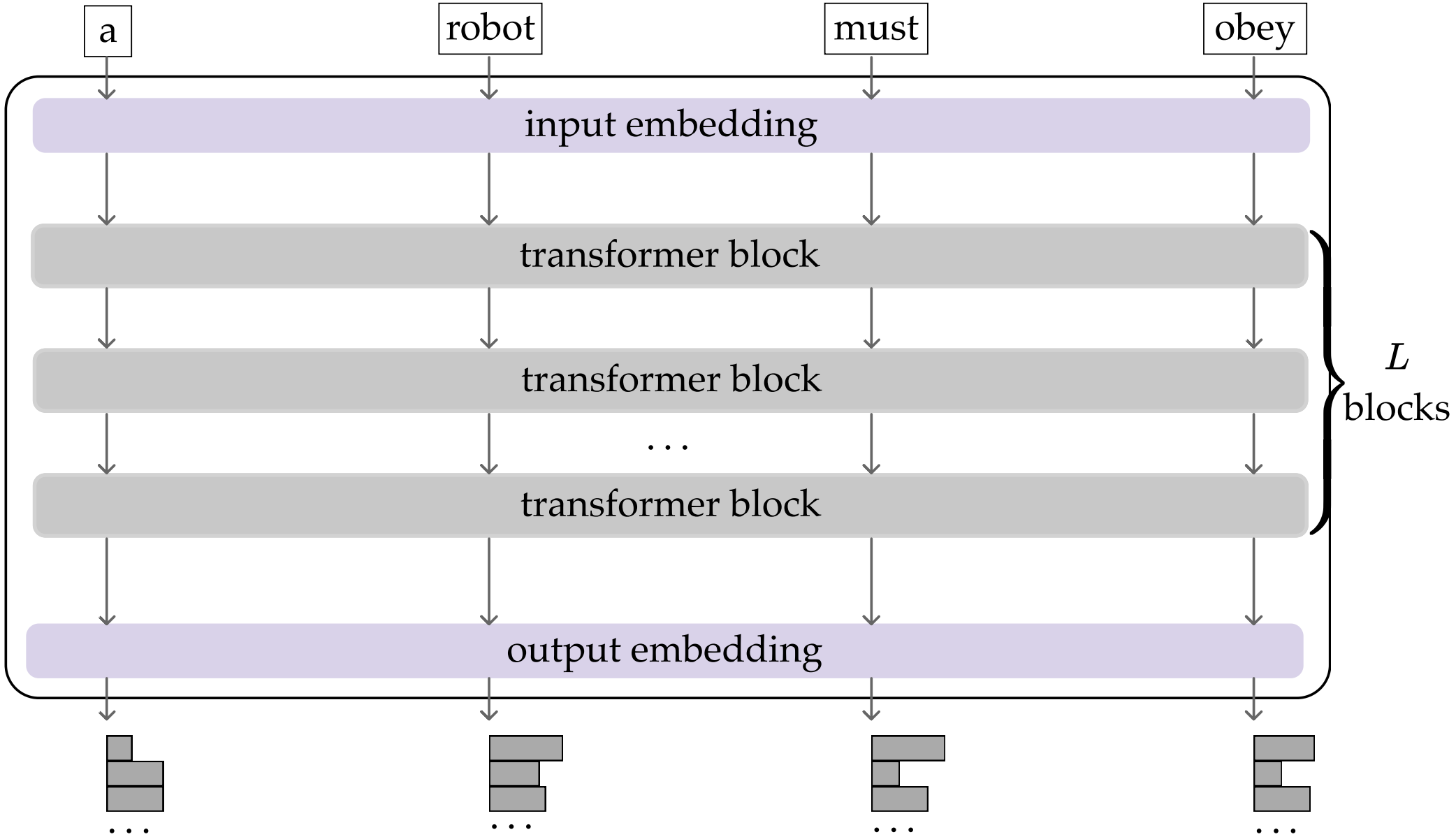
[image edited from 3b1b]

Outline

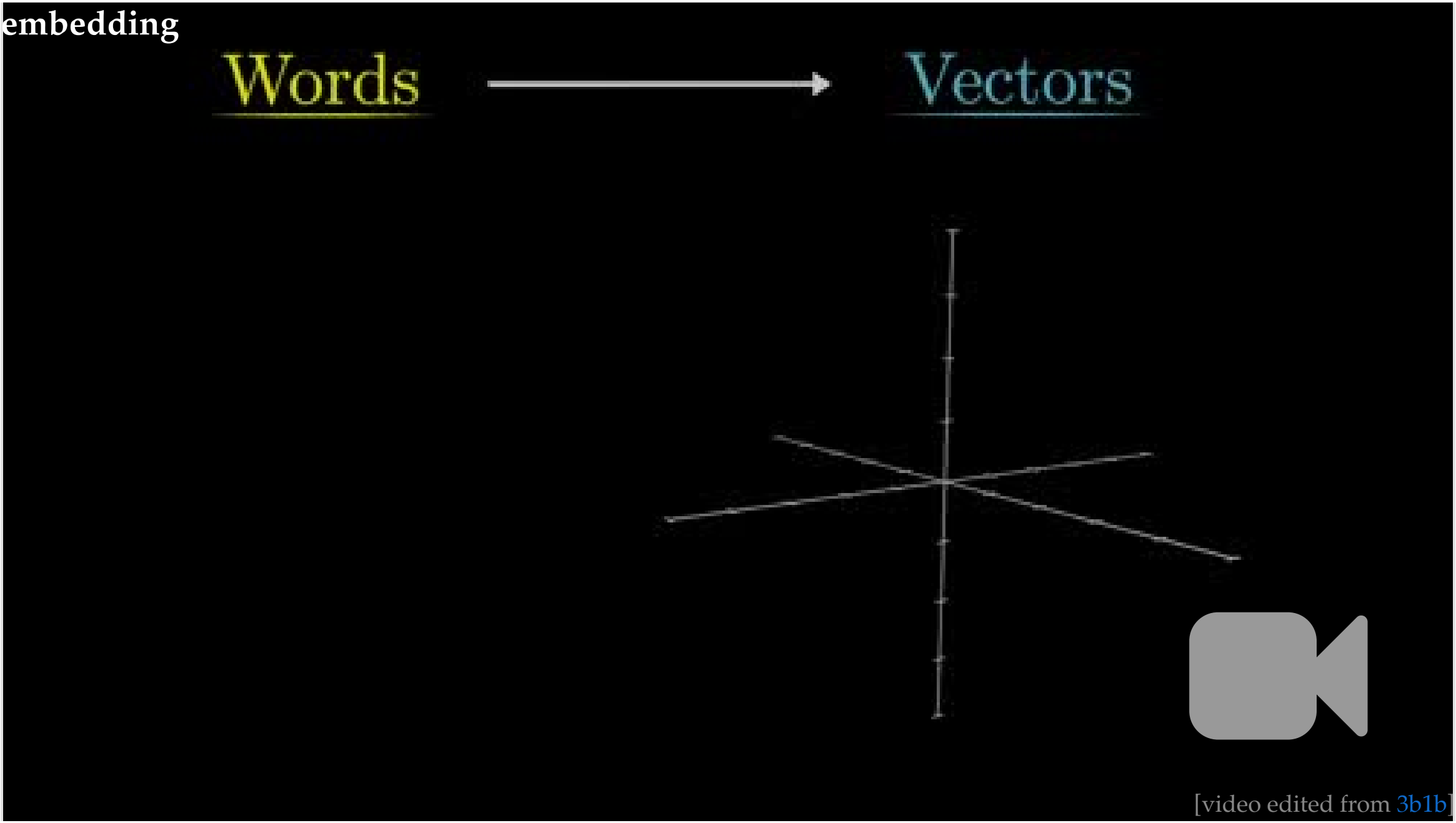
- Recap, convolutional neural networks
- Transformers example use case: large language models
- Transformers key ideas:
  - Embedding
  - Attention mechanism
- (Applications and case studies)

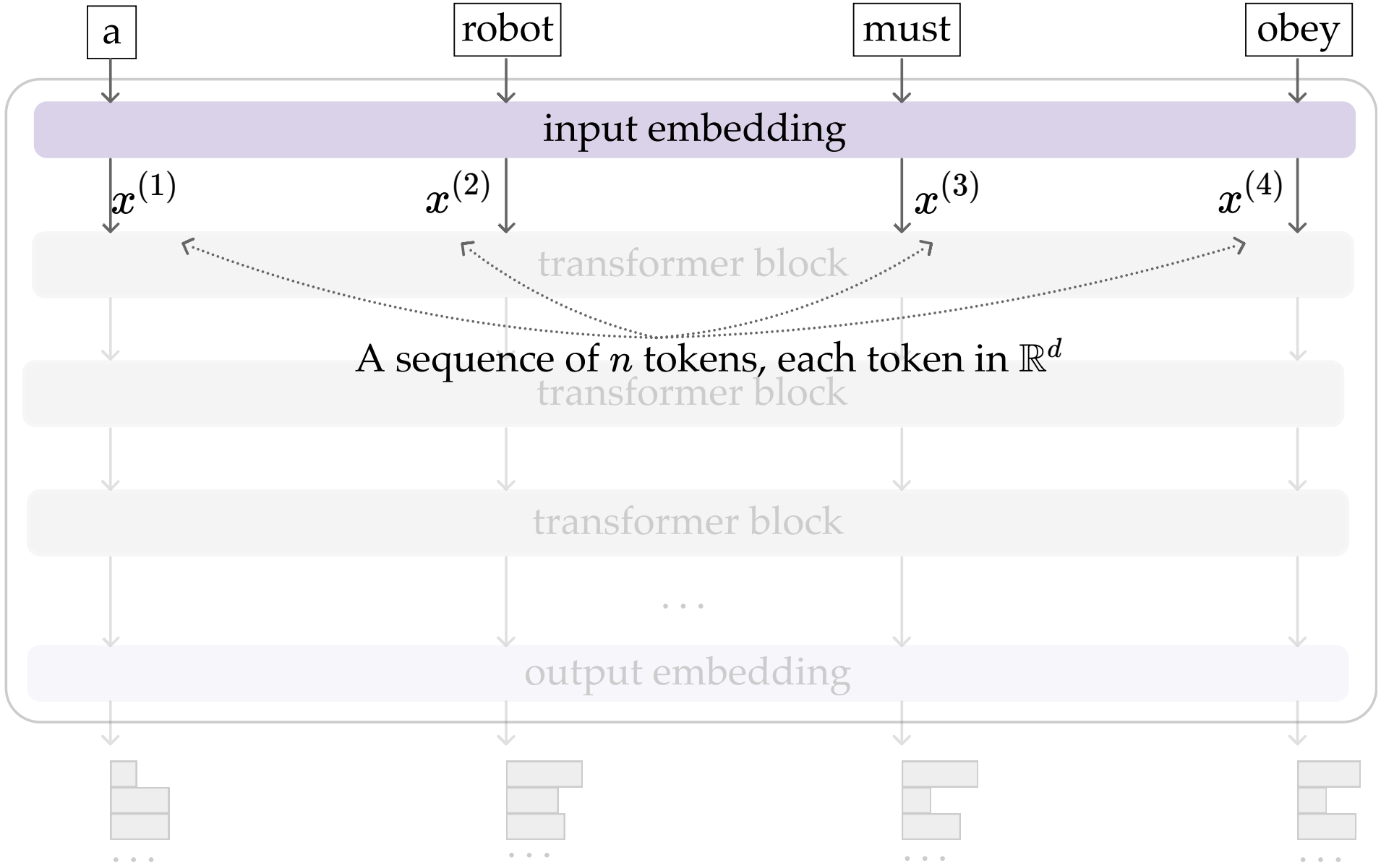
"A robot must obey the orders given it by human beings ..."

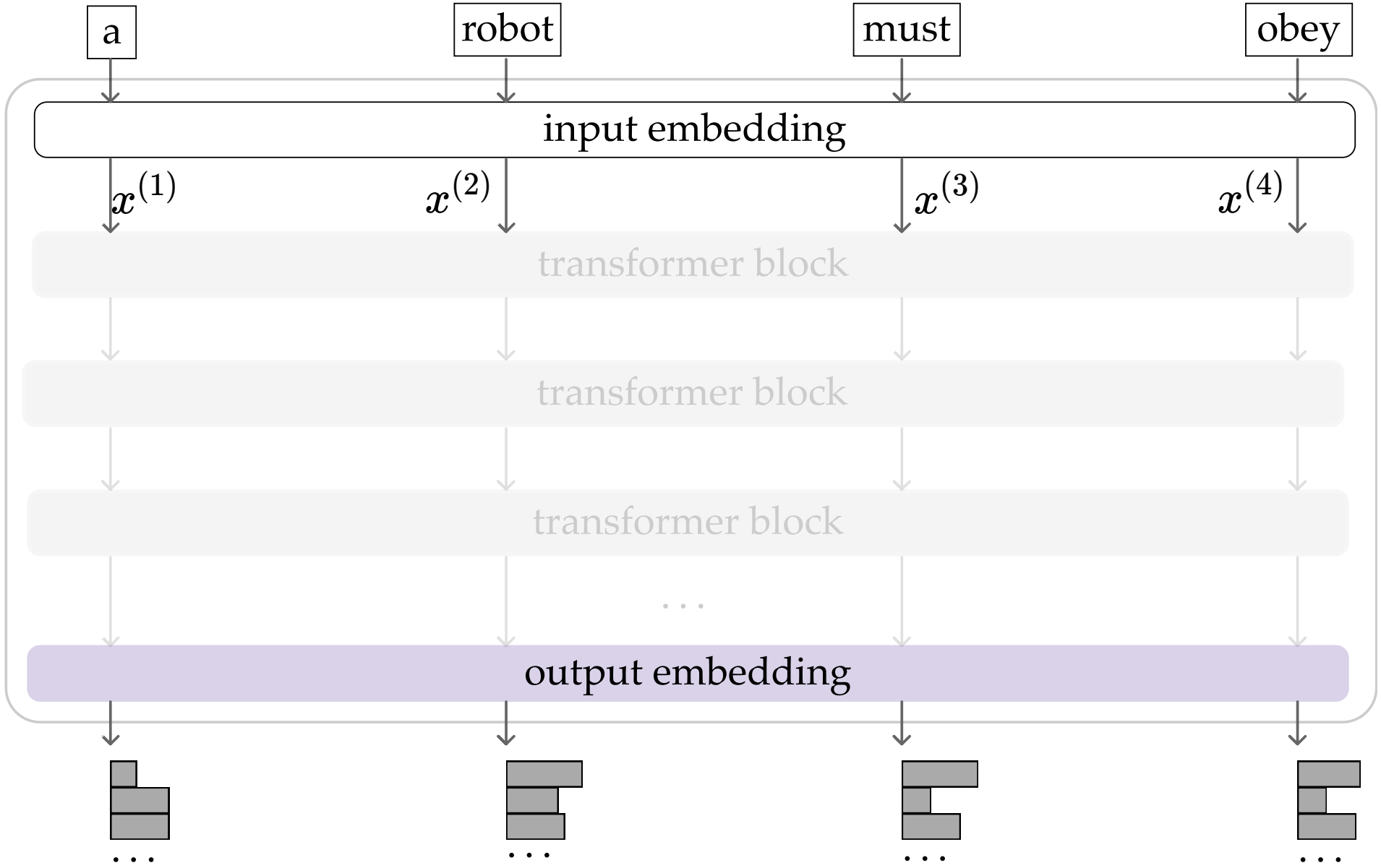


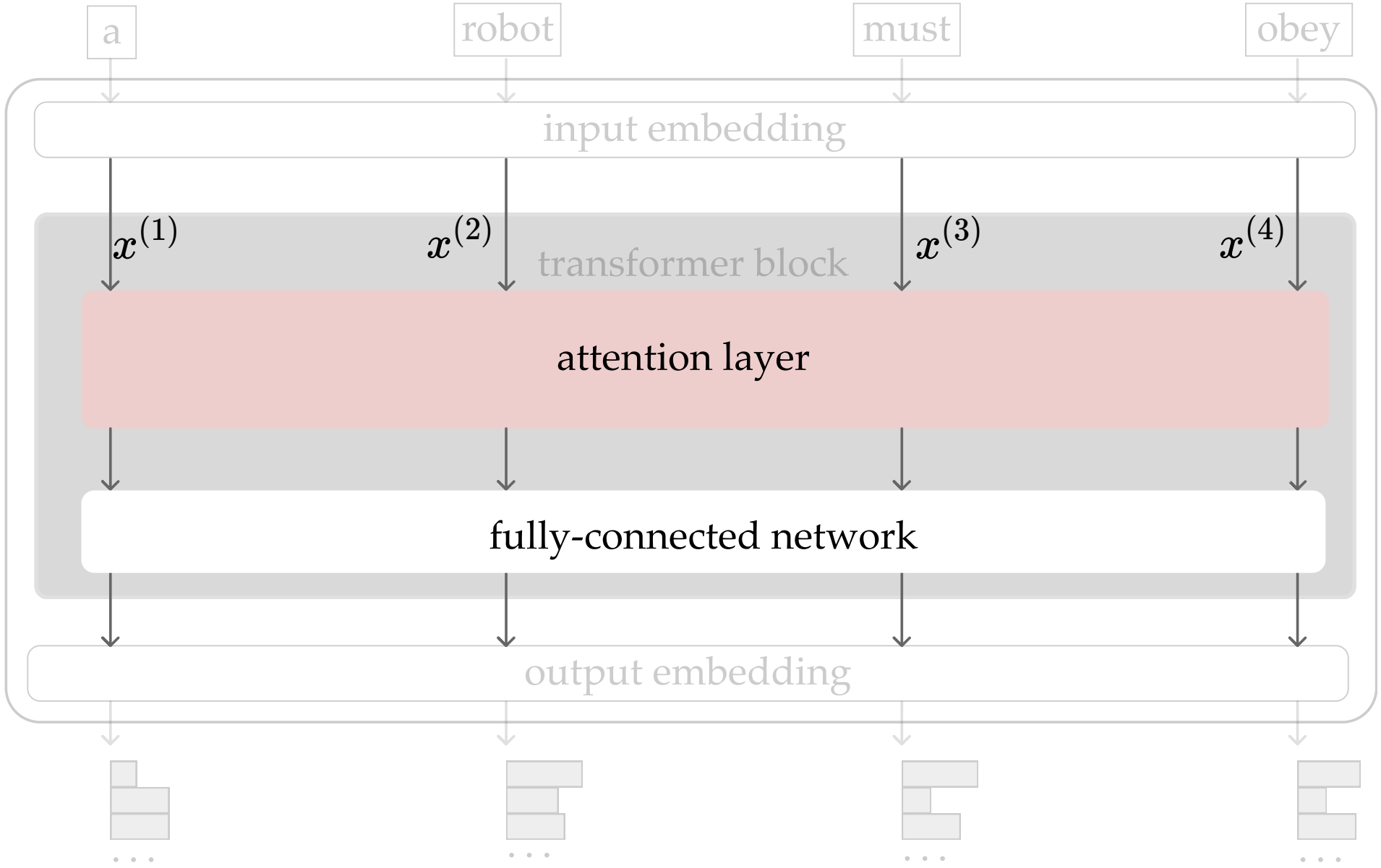

a
robot
must
obey
input embedding
transformer block
transformer block
...
transformer block
output embedding
L
blocks
...
...
...
...

embedding

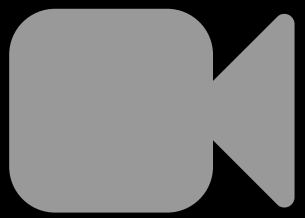



a
robot
must
obey
input embedding
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
transformer block
A sequence of n tokens, each token in $\mathbb{R}^d$
transformer block
transformer block
. . .
output embedding

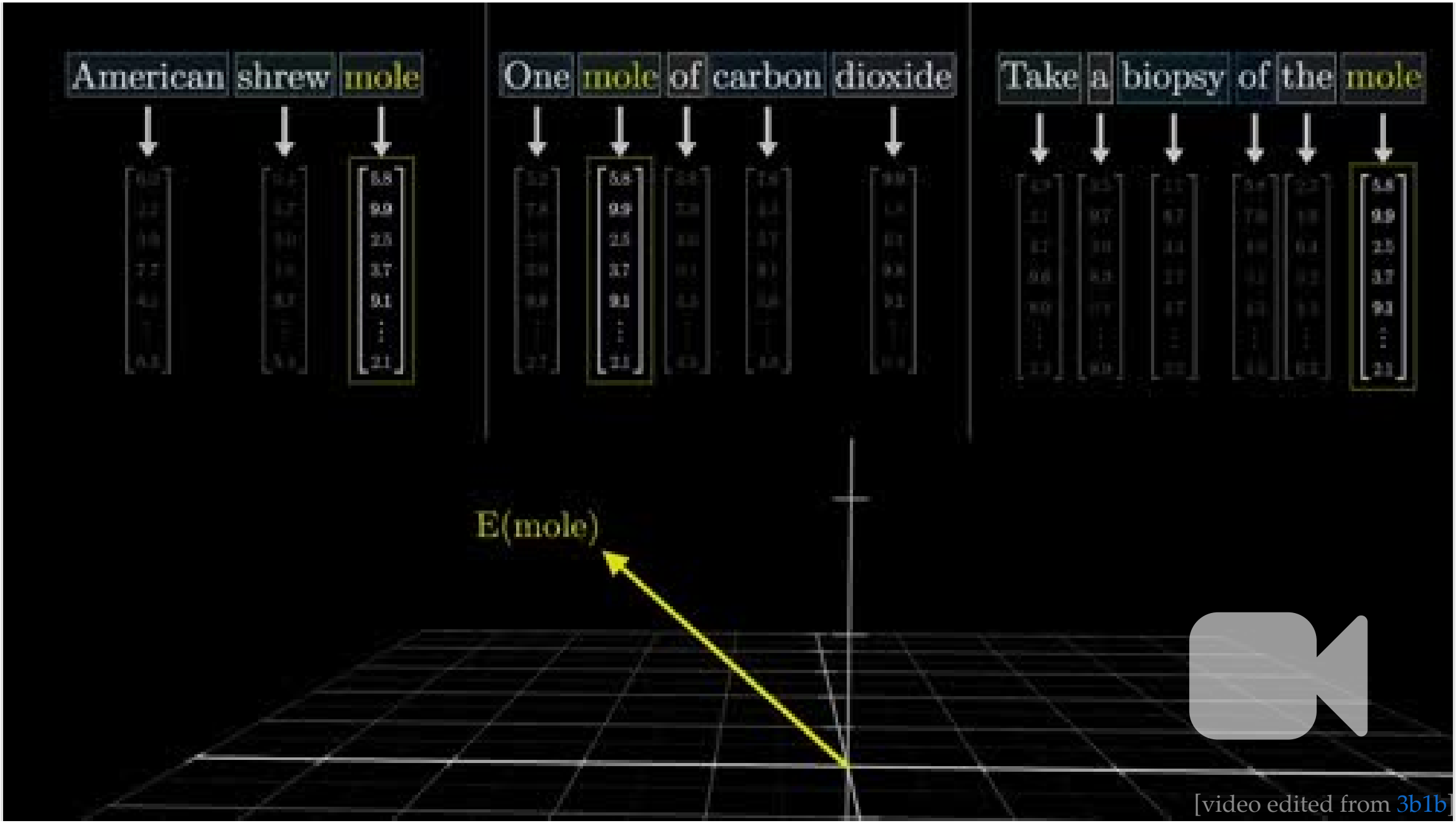

a
robot
must
obey
input embedding
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
transformer block
transformer block
transformer block
. . .
output embedding
. . .
. . .
. . .
. . .

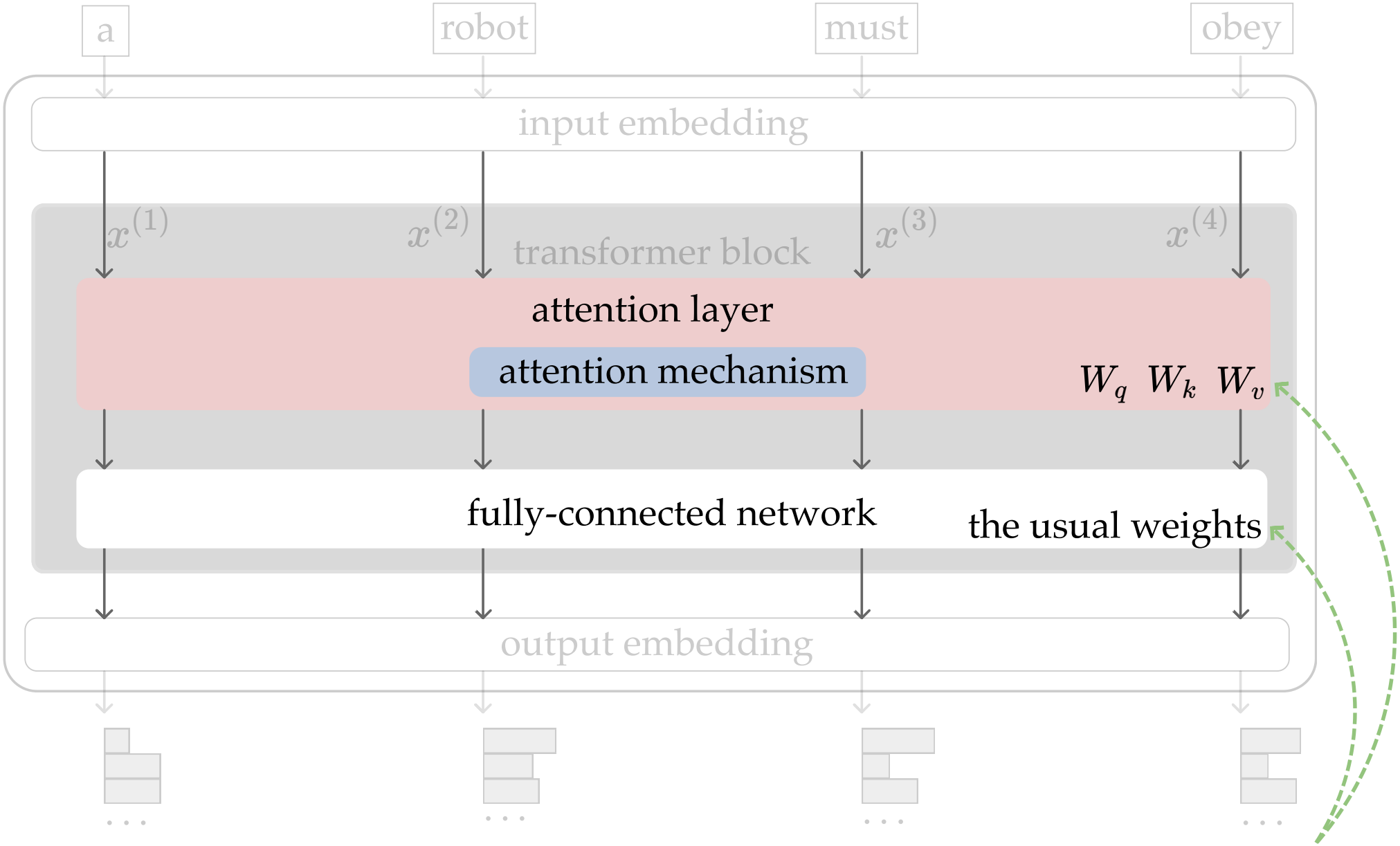

a
robot
must
obey
input embedding
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
transformer block
attention layer
fully-connected network
output embedding

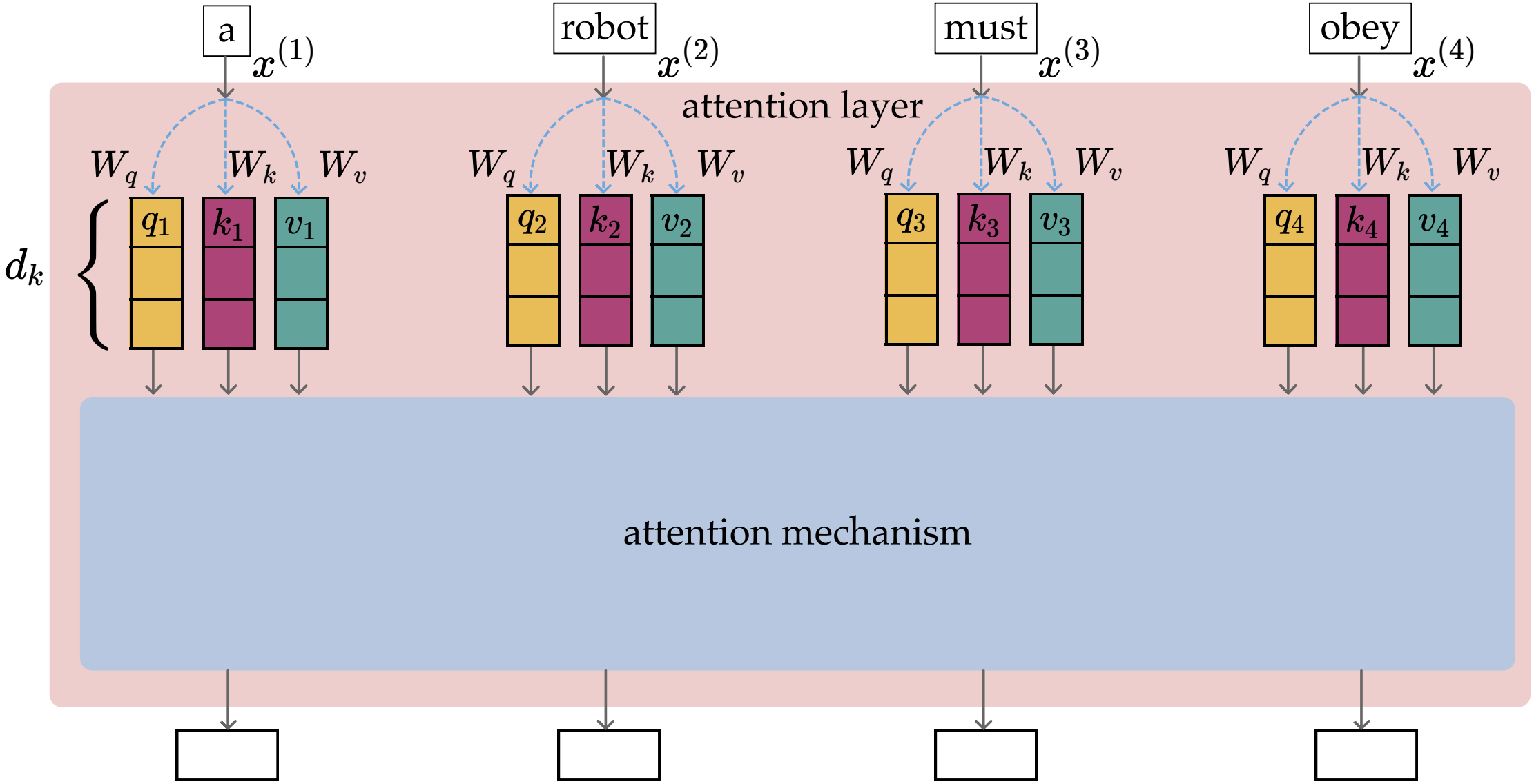
American shrew mole

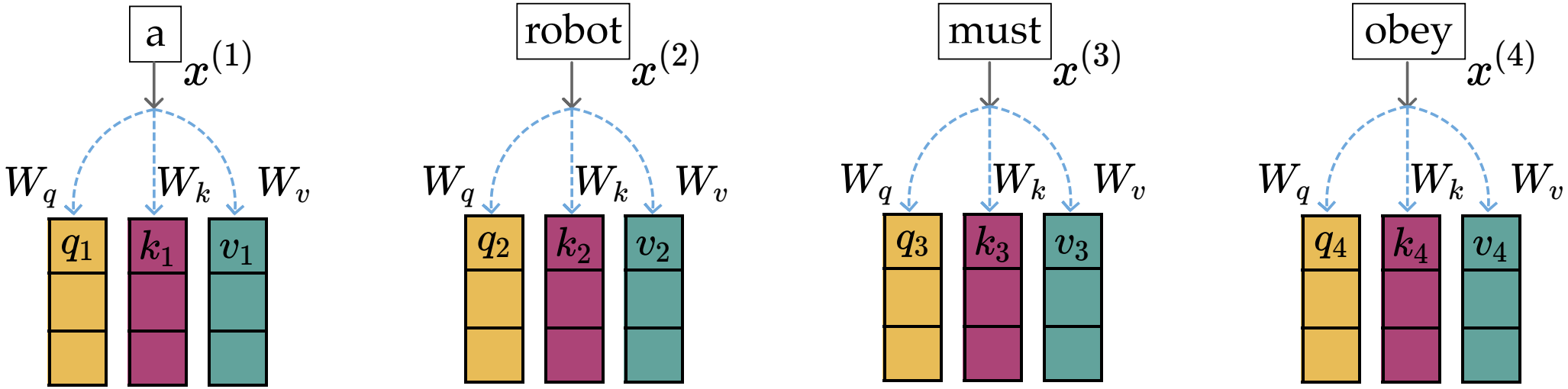


[video edited from 3b1b]

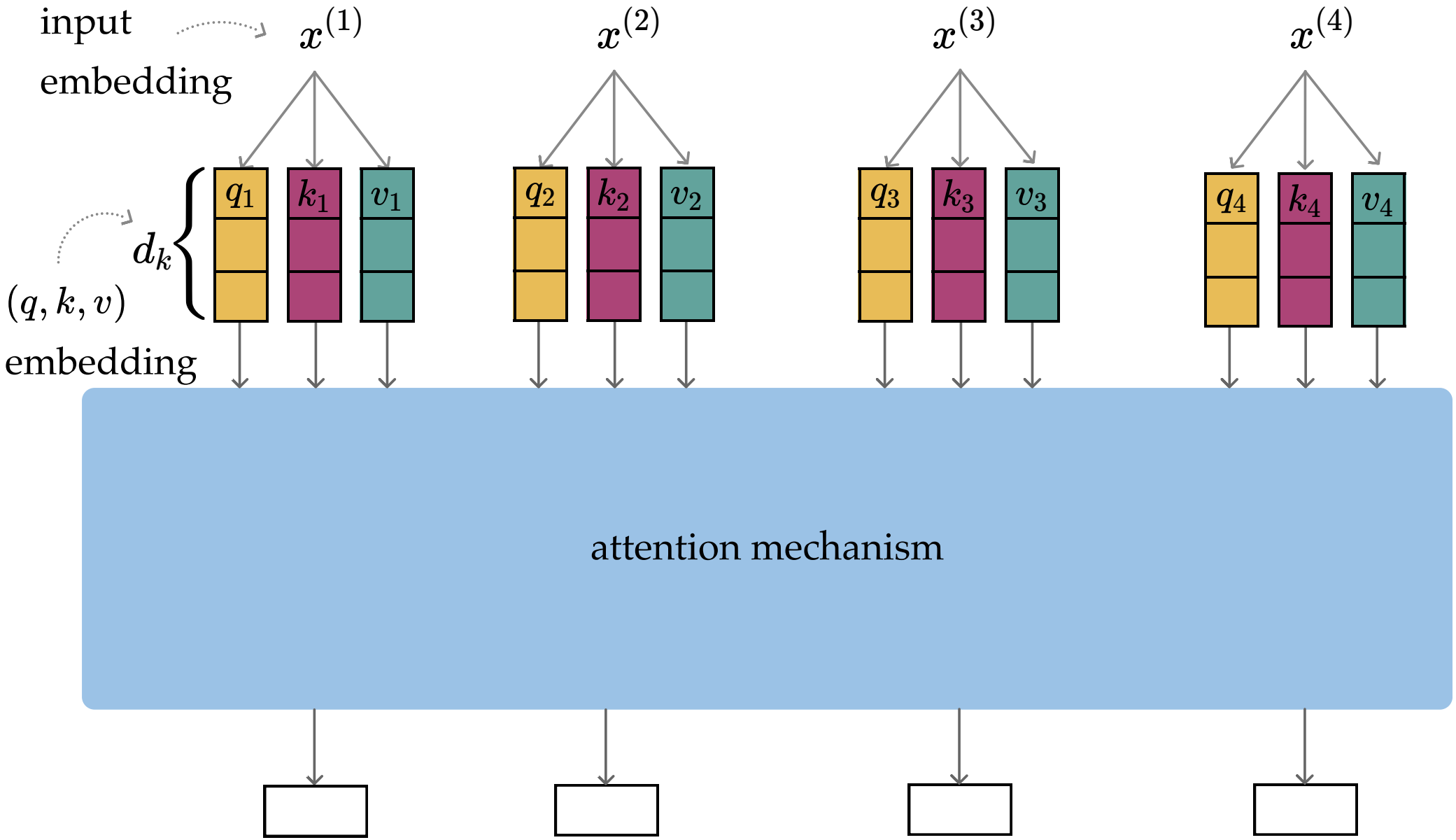

American shrew mole
One mole of carbon dioxide
Take a biopsy of the mole
E(mole)
[video edited from 3b1b]


a
robot
must
obey
input embedding
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
transformer block
attention layer
attention mechanism
W_q W_k W_v
fully-connected network
the usual weights
output embedding


a
robot
must
obey
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
attention layer
W_q
W_k
W_v
d_k
q_1
k_1
v_1
q_2
k_2
v_2
q_3
k_3
v_3
q_4
k_4
v_4
attention mechanism



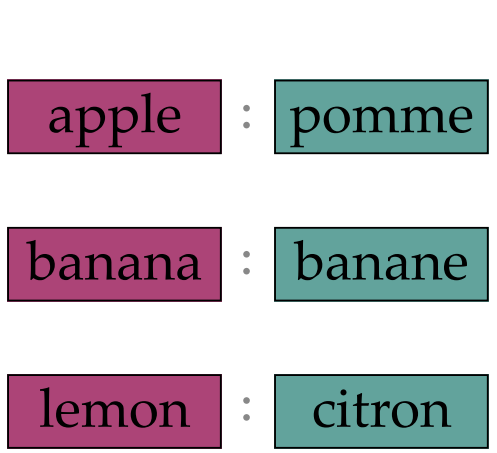
- sequence of d-dimensional input tokens x
- learnable weights, W_q, W_v, W_k, all in $\mathbb{R}^{d \times d_k}$
- map the input sequence into d_k-dimensional (qkv) sequence, e.g., $q_1 = W_q^T x^{(1)}$
- the weights are **shared**, across the sequence of tokens -- **parallel** processing


input
embedding
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
(q, k, v)
embedding
attention mechanism

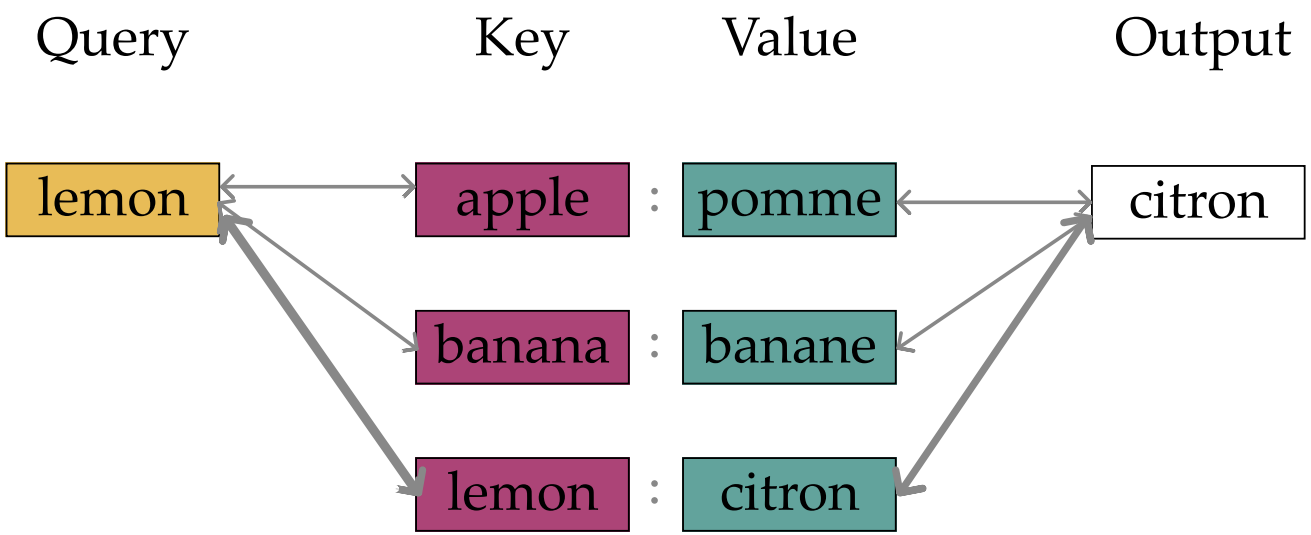
Having good (query, key, value) embedding enables effective attention.

Let's think about dictionary look-up:

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

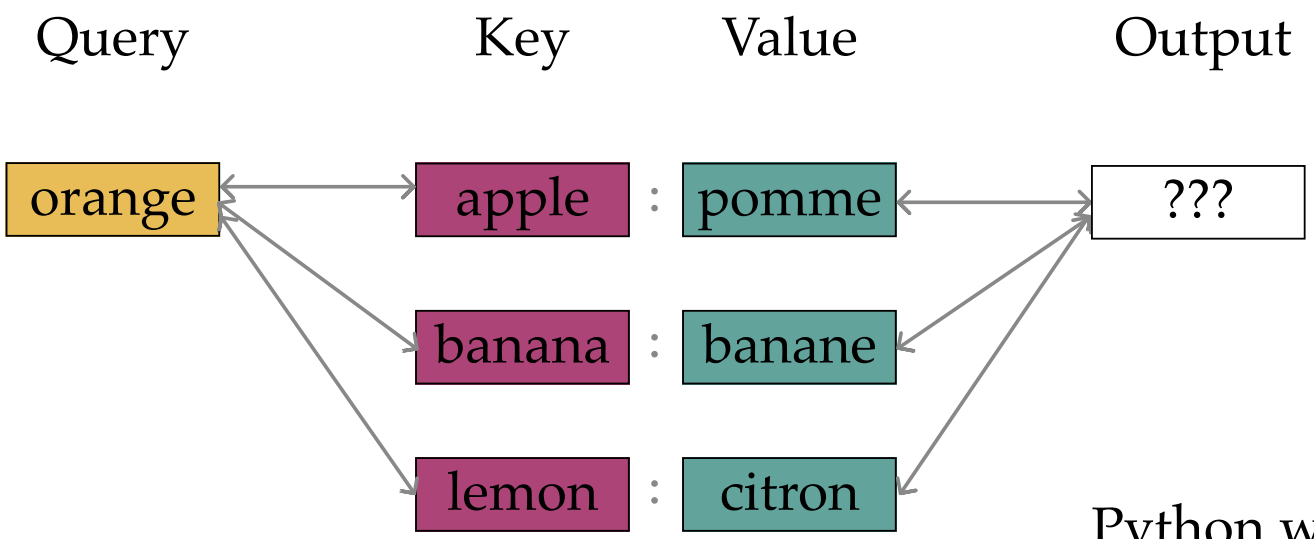


```python
dict_en2fr = {
  "apple" : "pomme",
  "banana": "banane",
  "lemon" : "citron"}

query = "lemon"
output = dict_en2fr[query]
```



What if we run

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

query = "orange"
output = dict_en2fr[query]
```

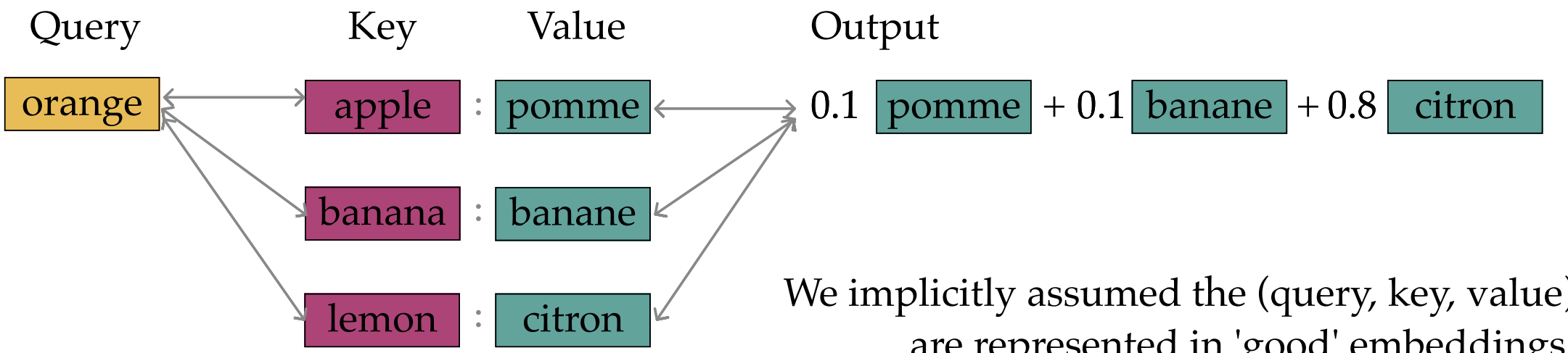


Python would complain. 🤯

What if we run

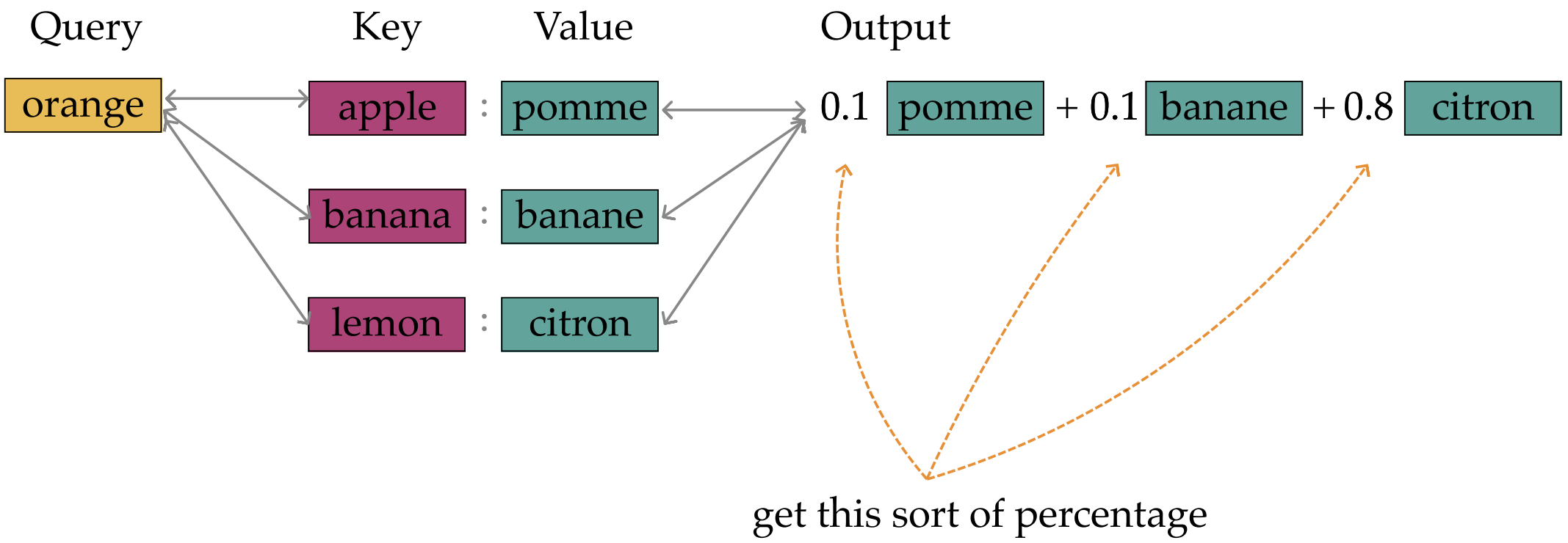
```
dict_en2fr = {
  "apple" : "pomme",
  "banana": "banane",
  "lemon" : "citron"}

query = "orange"
output = dict_en2fr[query]
```

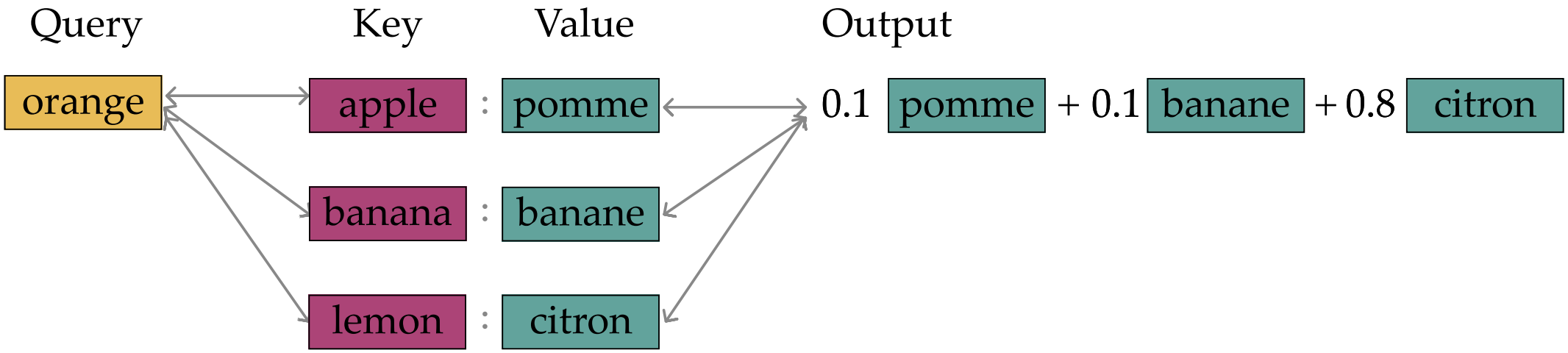
But we can probably see the rationale behind this:



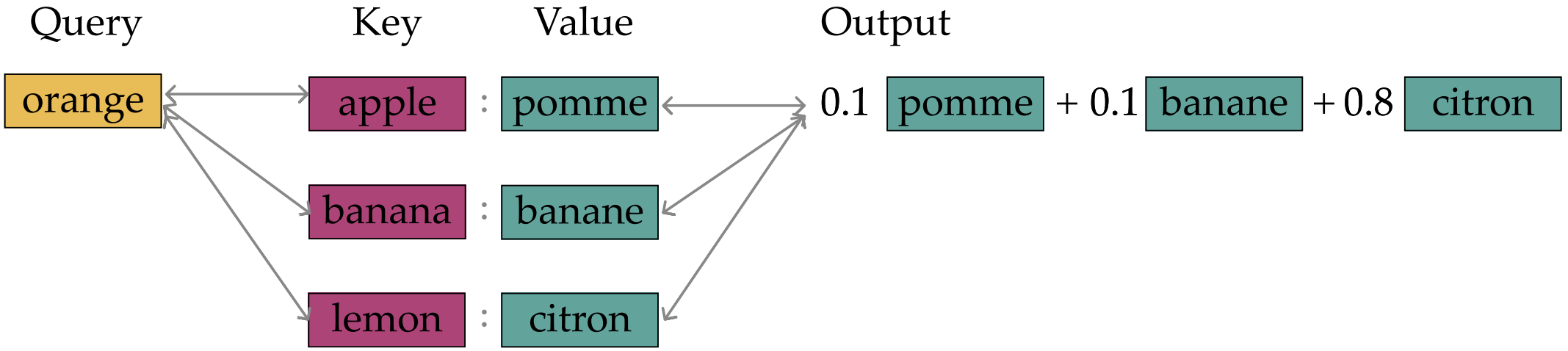
We implicitly assumed the (query, key, value) are represented in 'good' embeddings.

If we are to generalize this idea, we need to:

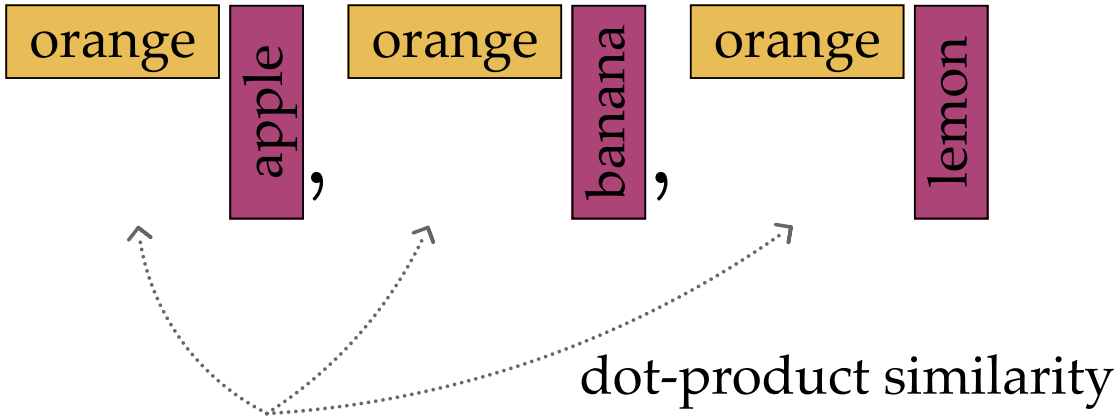




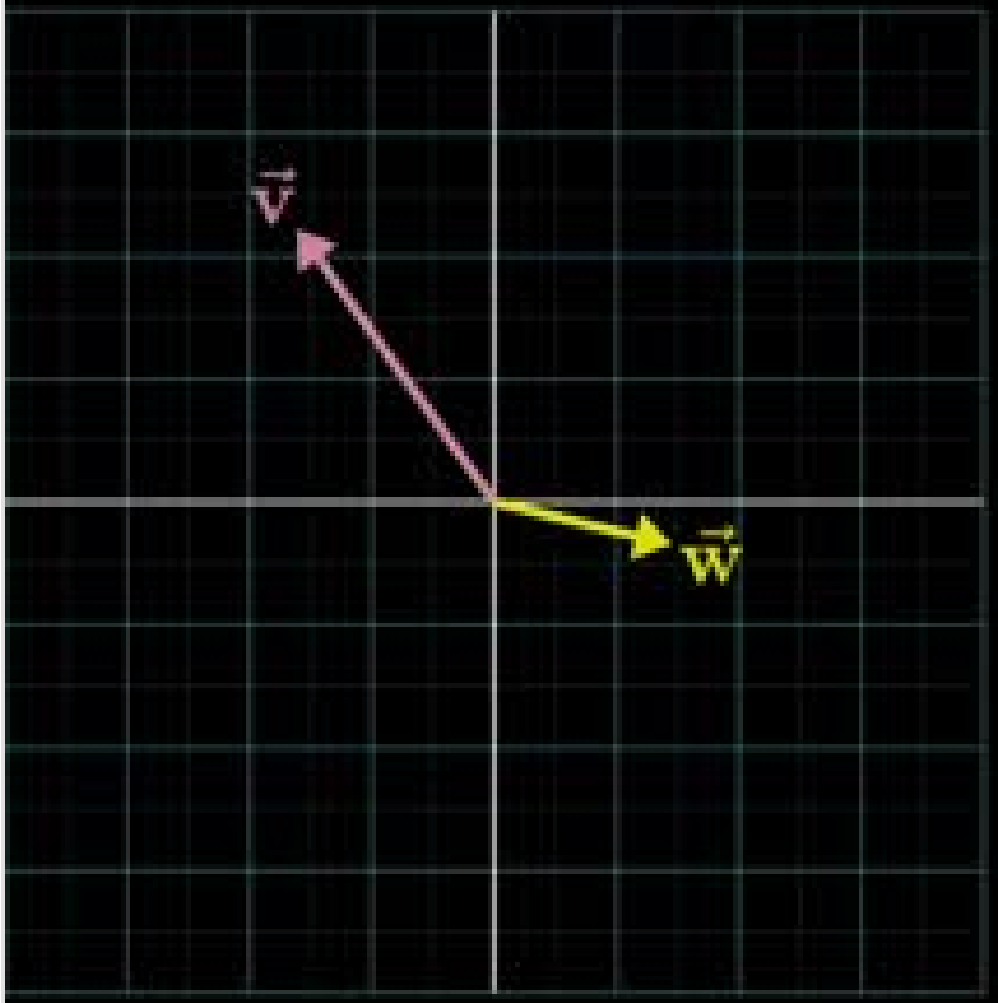
very roughly, the attention mechanism does exactly this kind of "soft" look-up:



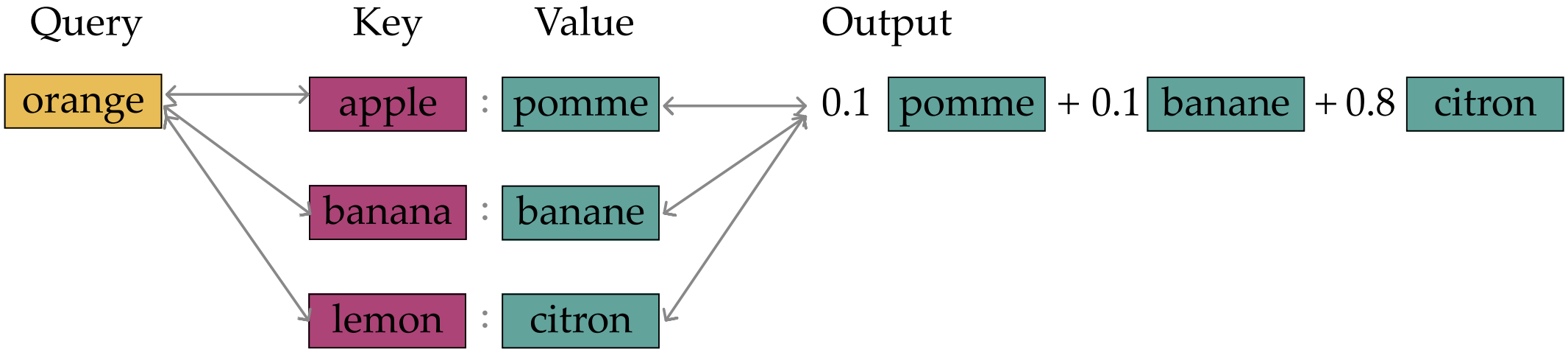
very roughly, the attention mechanism does exactly this kind of "soft" look-up:



dot-product similarity

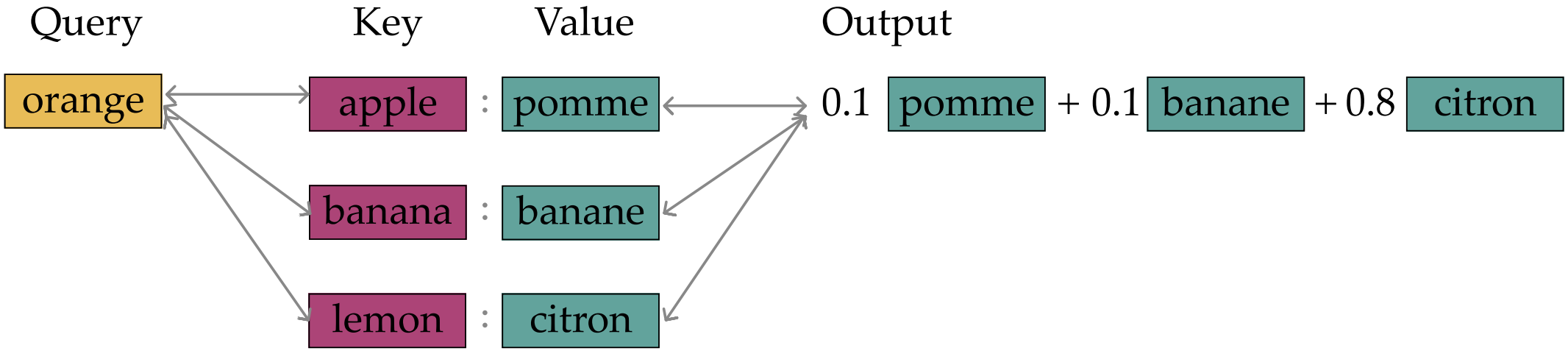


$$\underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}}_{\text{Dot product}} = \begin{matrix} v_1 w_1 \\ + \\ v_2 w_2 \\ + \\ v_3 w_3 \\ + \\ \vdots \\ + \\ v_n w_n \\ \| \\ -3.11 \end{matrix}$$



very roughly, the attention mechanism does exactly this kind of "soft" look-up:

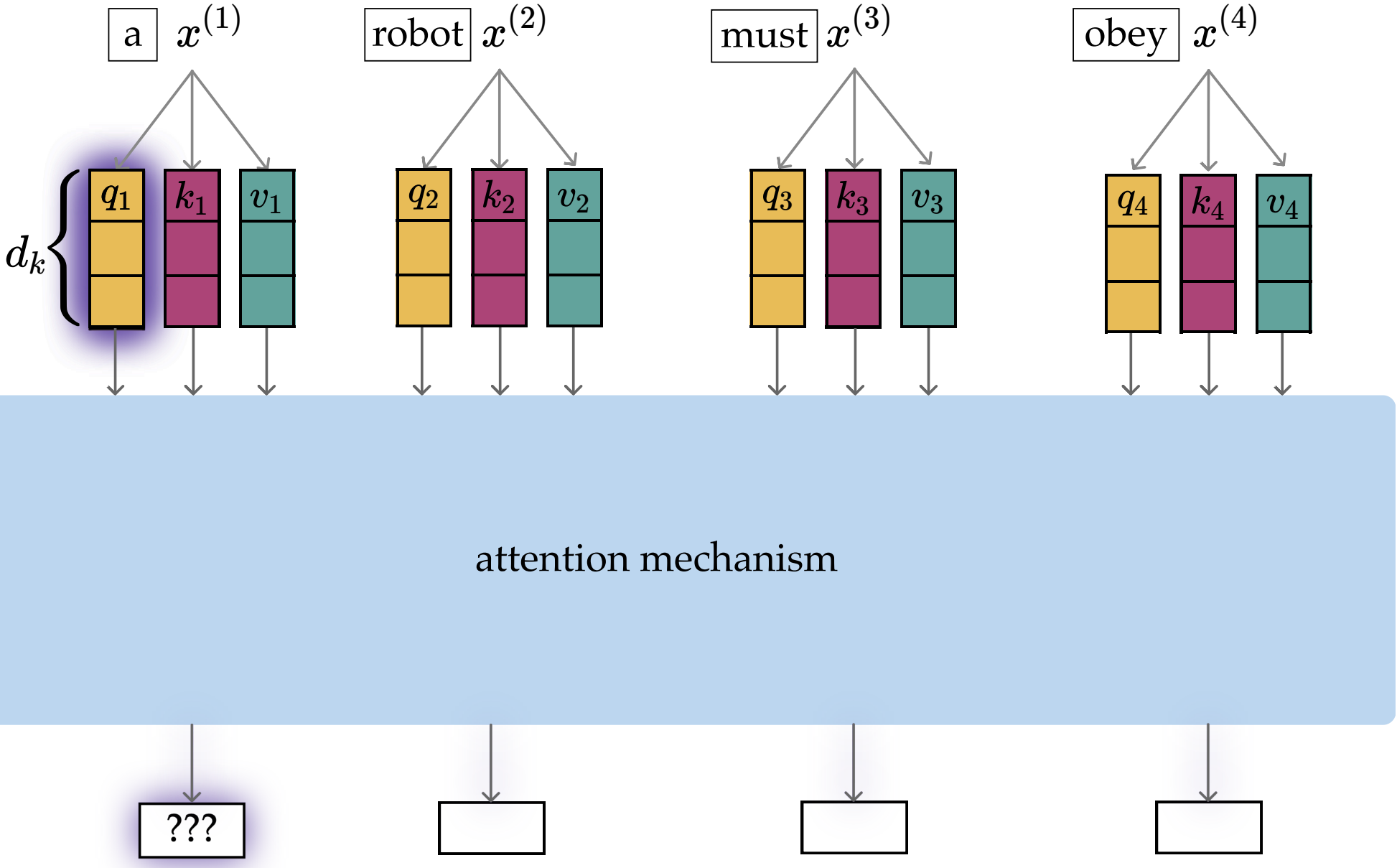
$$\text{softmax}\left(\text{orange}\,\text{apple}, \text{orange}\,\text{banana}, \text{orange}\,\text{lemon}\right) = \begin{bmatrix} 0.1 & 0.1 & 0.8 \end{bmatrix}$$

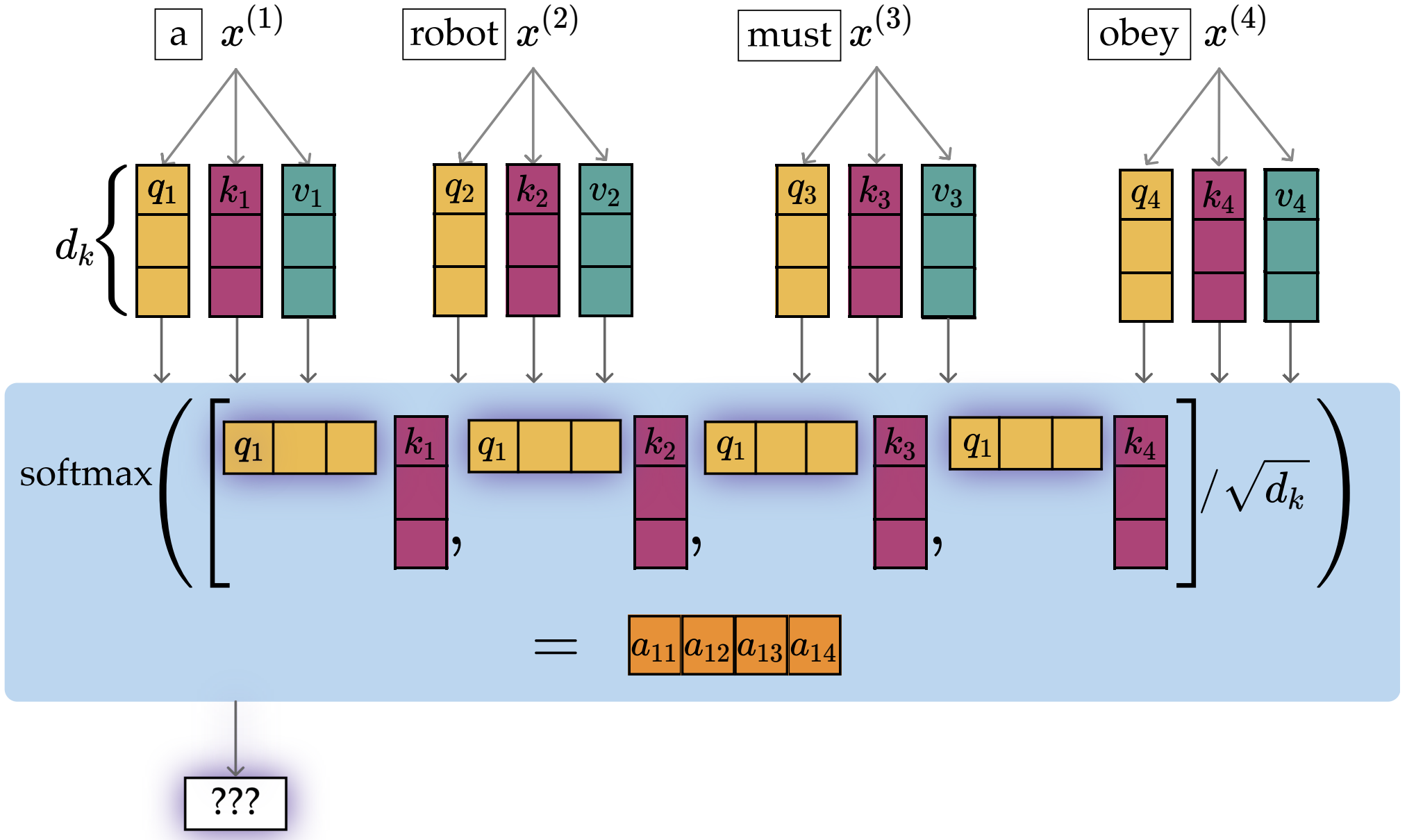


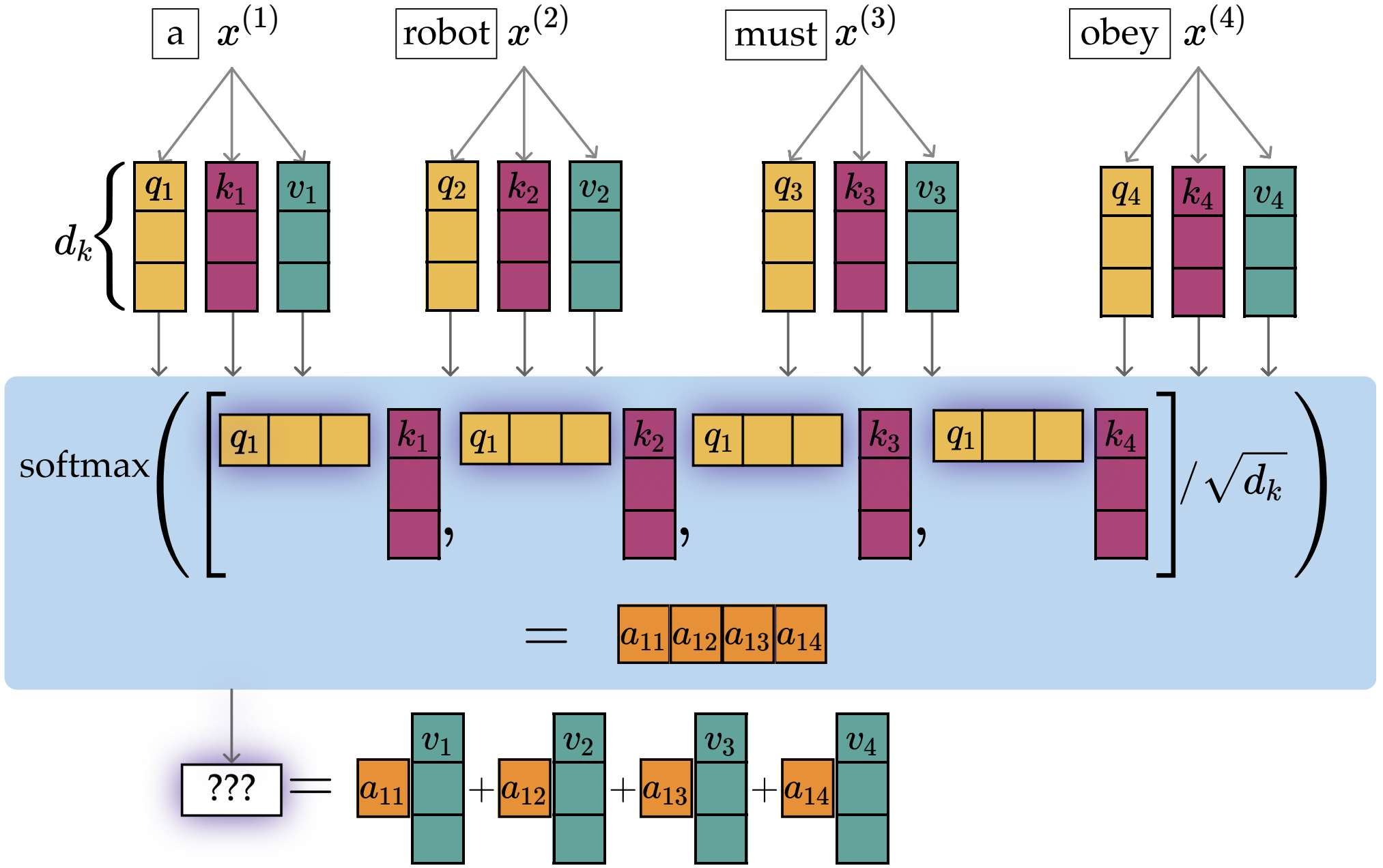
very roughly, attention mechanism does exactly this kind of "soft" look-up:

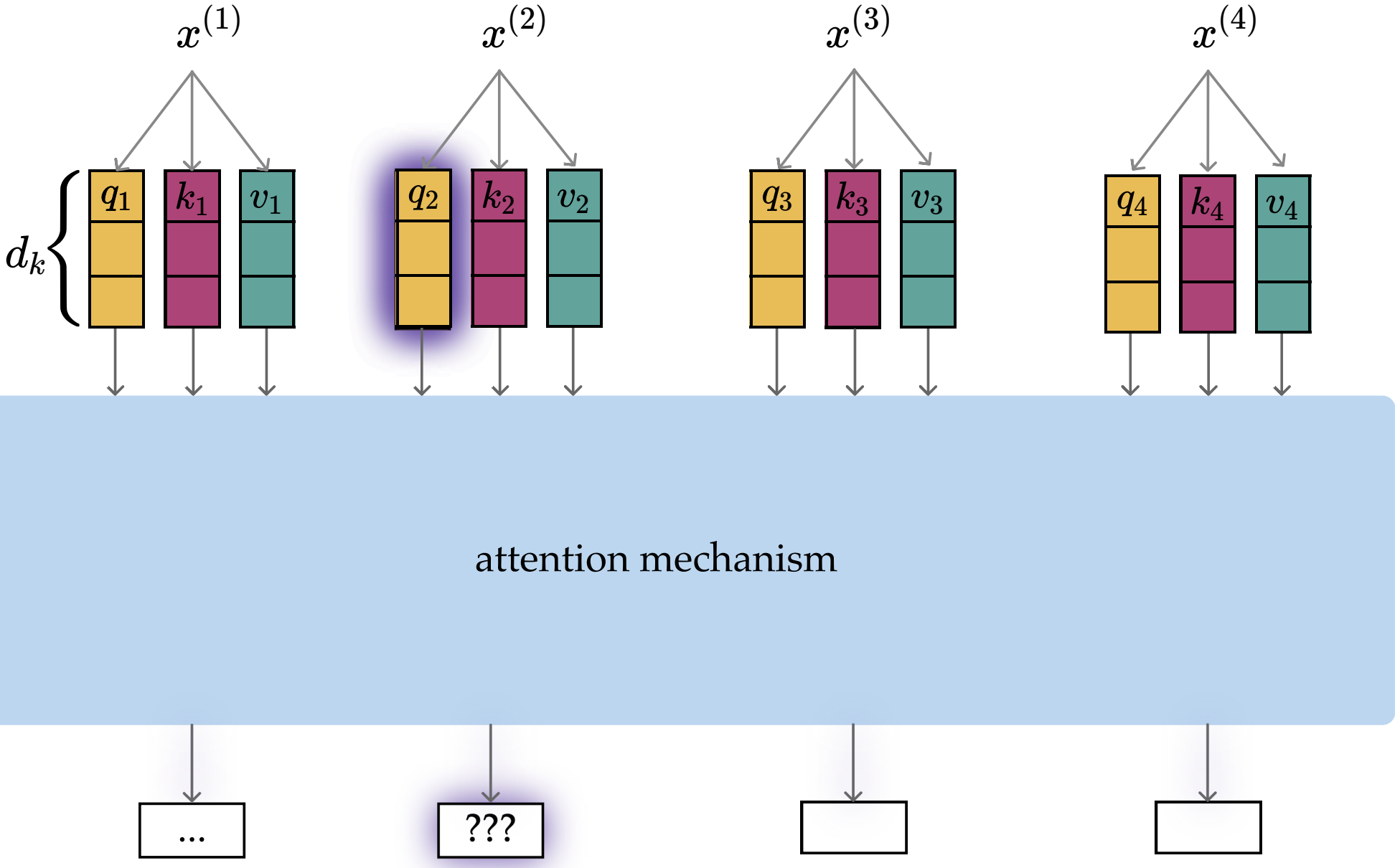
$$\text{softmax}\left(\text{orange}\cdot\text{apple}, \text{orange}\cdot\text{banana}, \text{orange}\cdot\text{lemon}\right) = \begin{bmatrix}0.1 & 0.1 & 0.8\end{bmatrix}$$

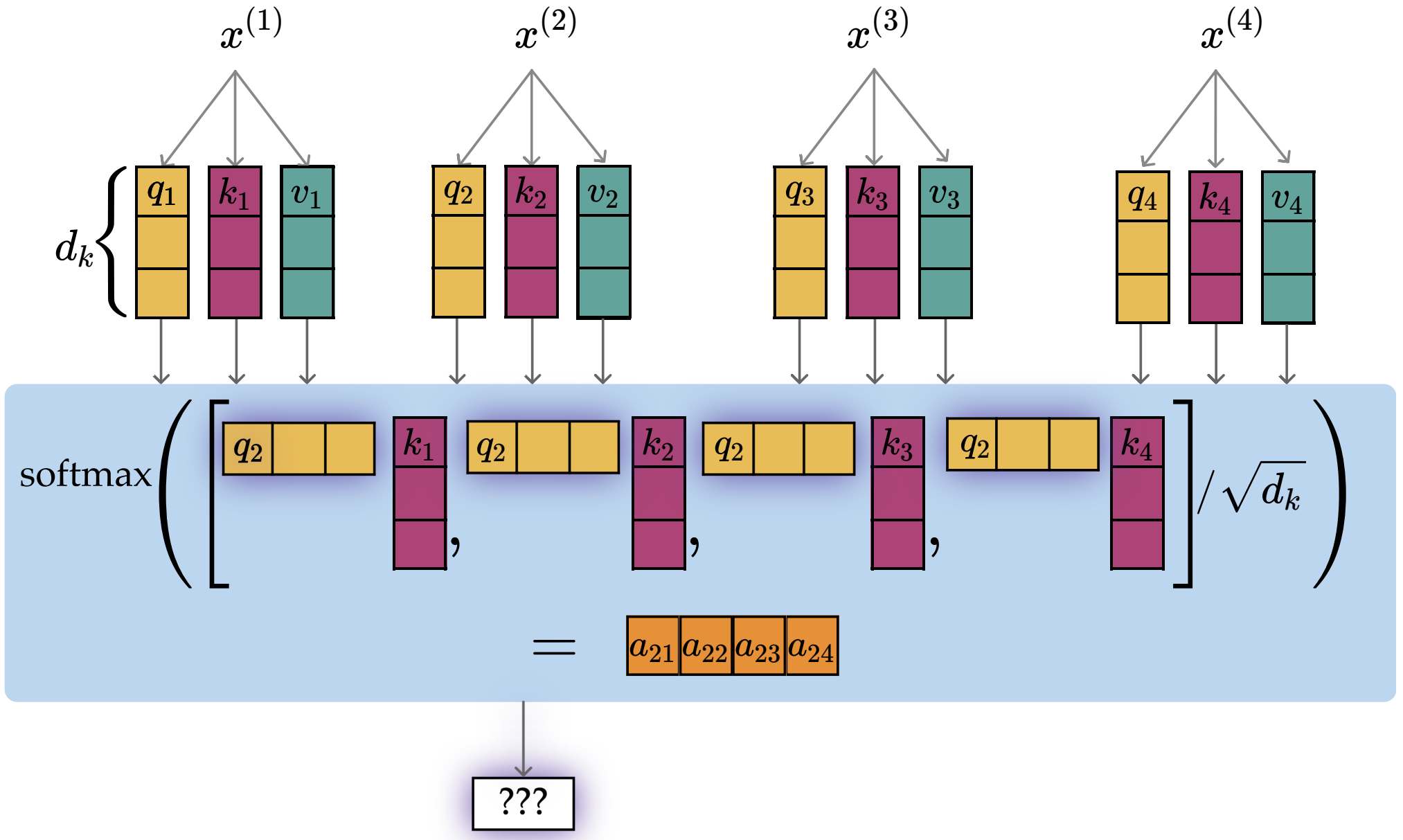
and output $0.1\,\text{pomme} + 0.1\,\text{banane} + 0.8\,\text{citron}$ ← weighted average over values

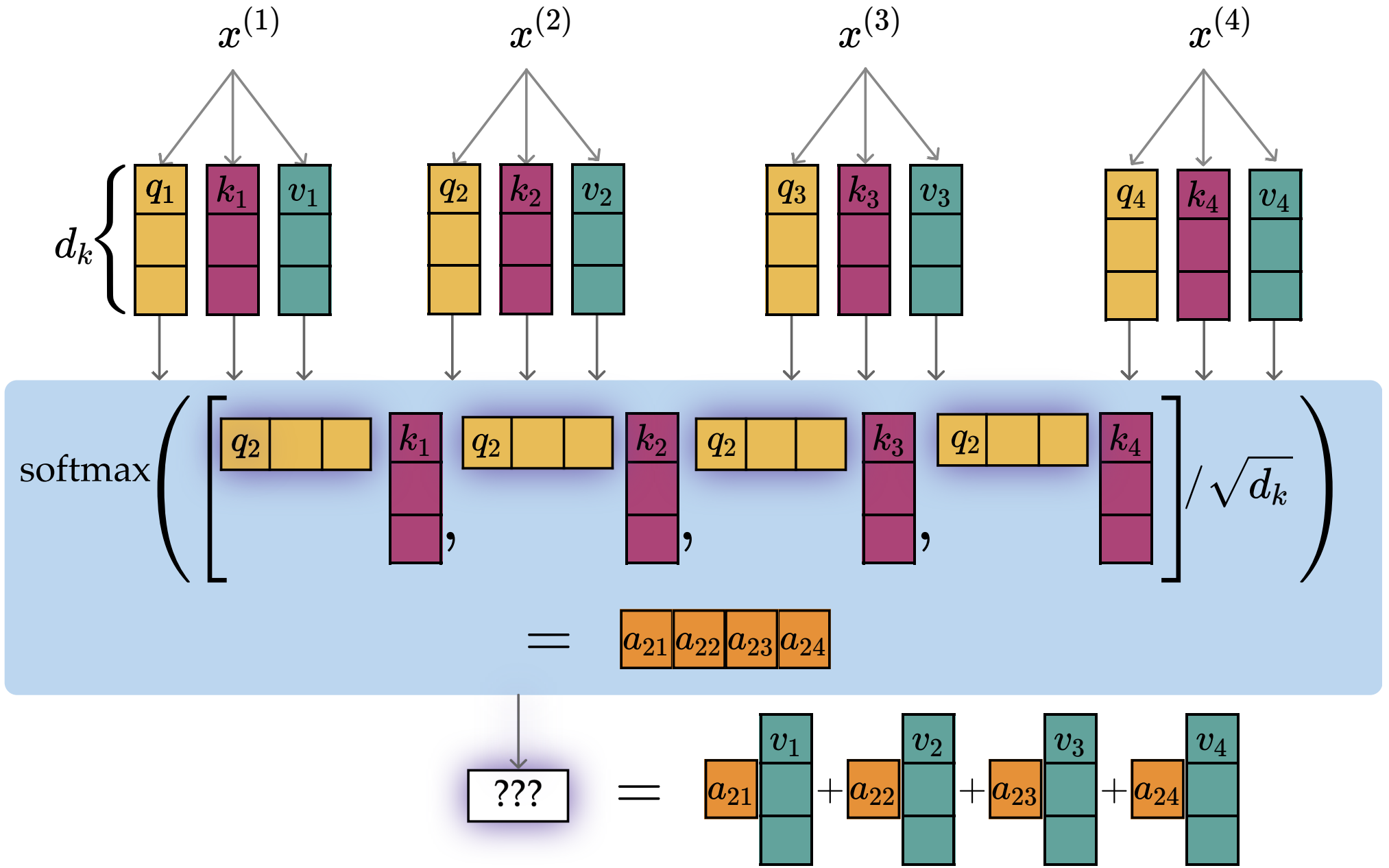

a
$x^{(1)}$
robot
$x^{(2)}$
must
$x^{(3)}$
obey
$x^{(4)}$
q_1
k_1
v_1
q_2
k_2
v_2
q_3
k_3
v_3
q_4
k_4
v_4
d_k
attention mechanism
???

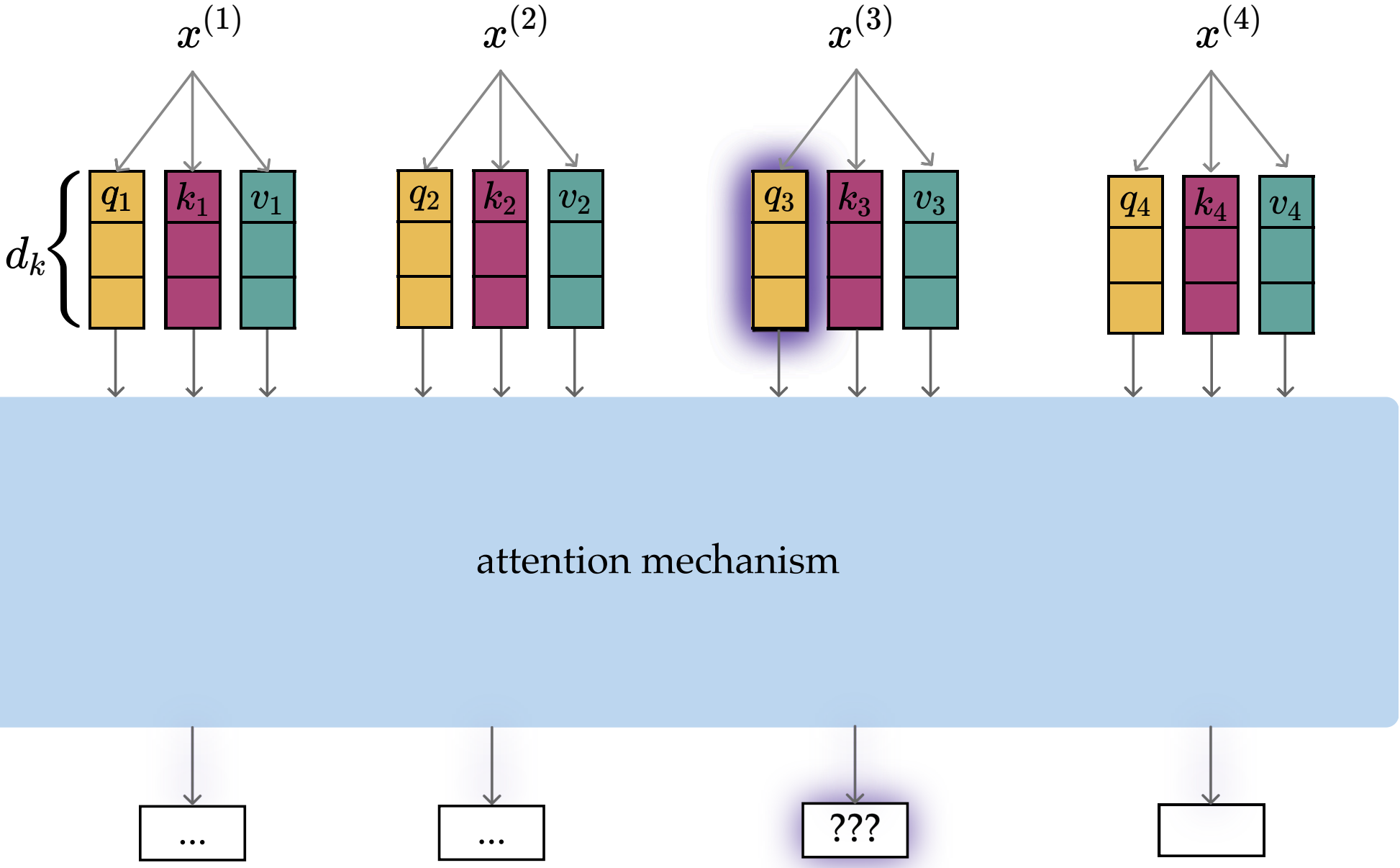

a
$x^{(1)}$
robot
$x^{(2)}$
must
$x^{(3)}$
obey
$x^{(4)}$
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
$/\sqrt{d_k}$
$=$
a_{11} a_{12} a_{13} a_{14}
???

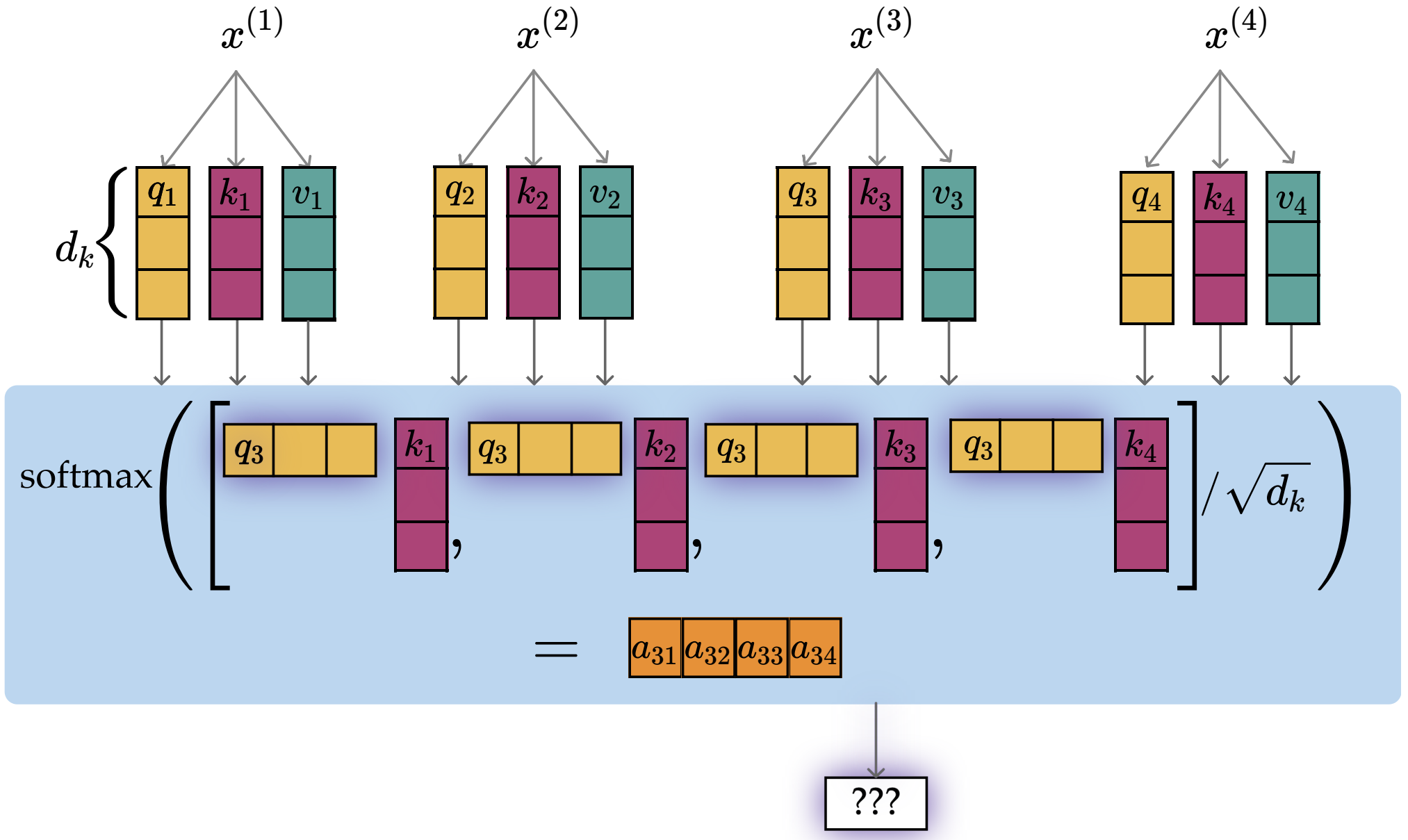

a $x^{(1)}$
robot $x^{(2)}$
must $x^{(3)}$
obey $x^{(4)}$
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
$/\sqrt{d_k}$
a_{11} a_{12} a_{13} a_{14}
???
a_{11} v_1 + a_{12} v_2 + a_{13} v_3 + a_{14} v_4

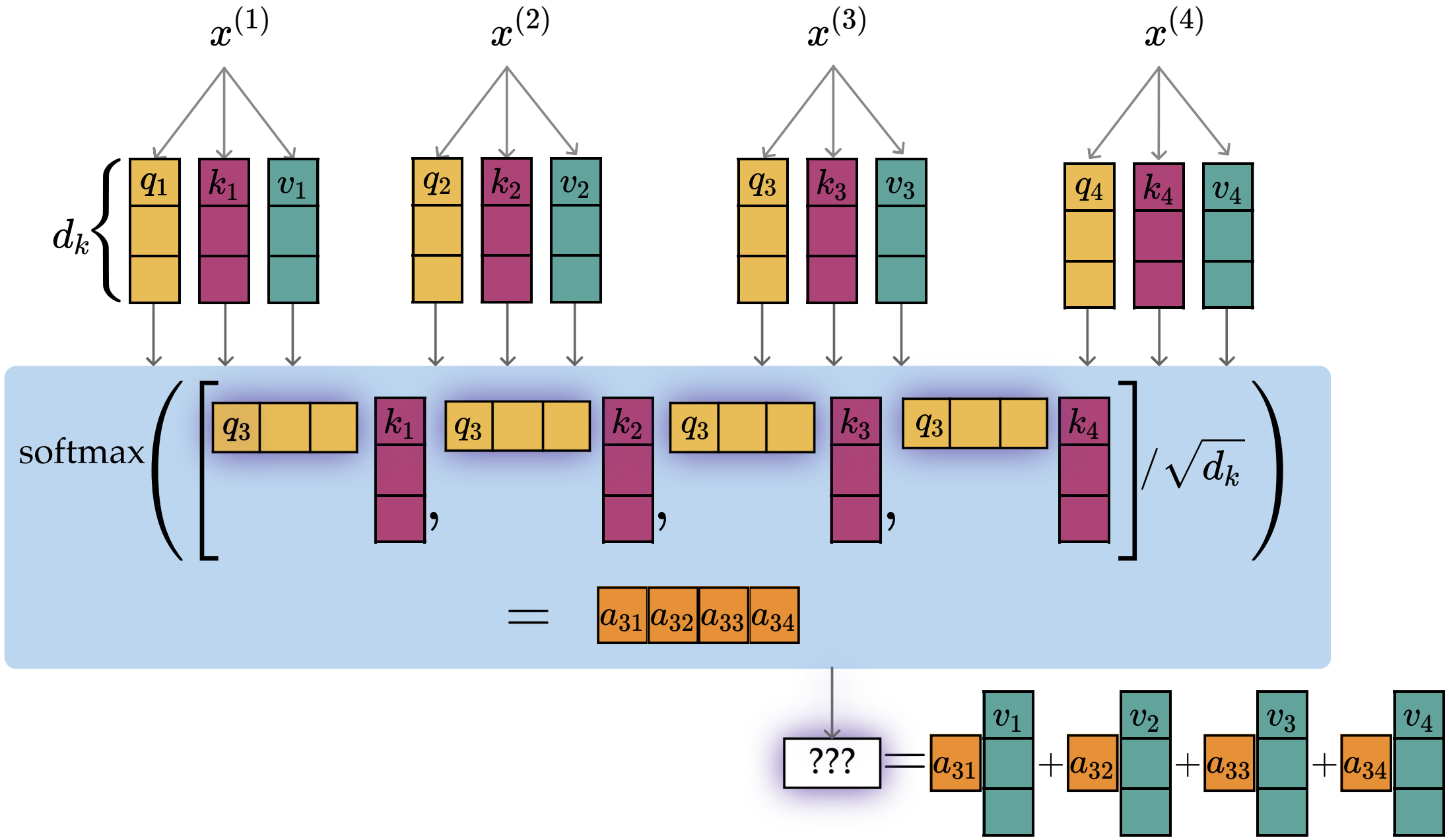

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
d_k
q_1
k_1
v_1
q_2
k_2
v_2
q_3
k_3
v_3
q_4
k_4
v_4
attention mechanism
...
???

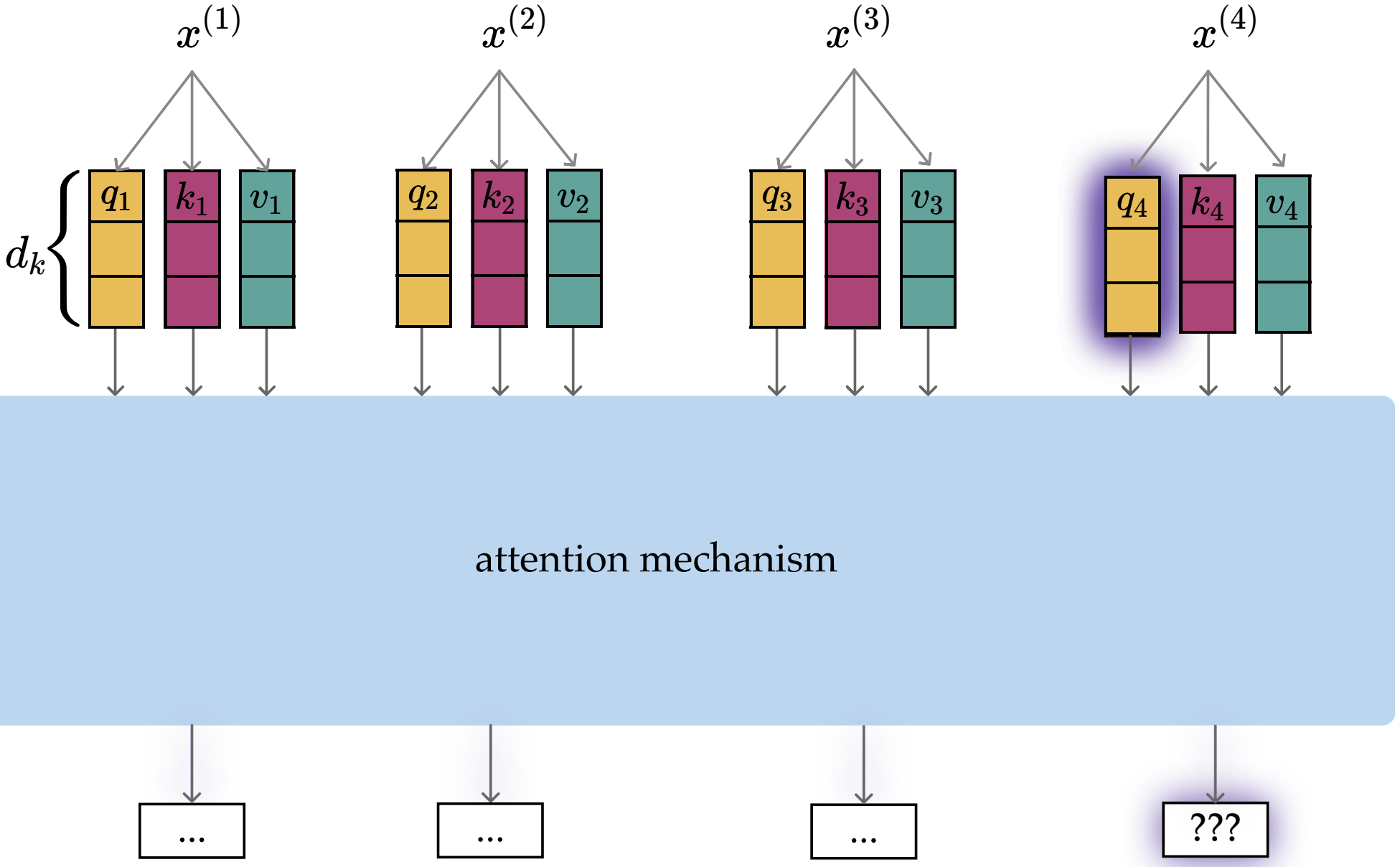

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
$/\sqrt{d_k}$
a_{21} a_{22} a_{23} a_{24}
???

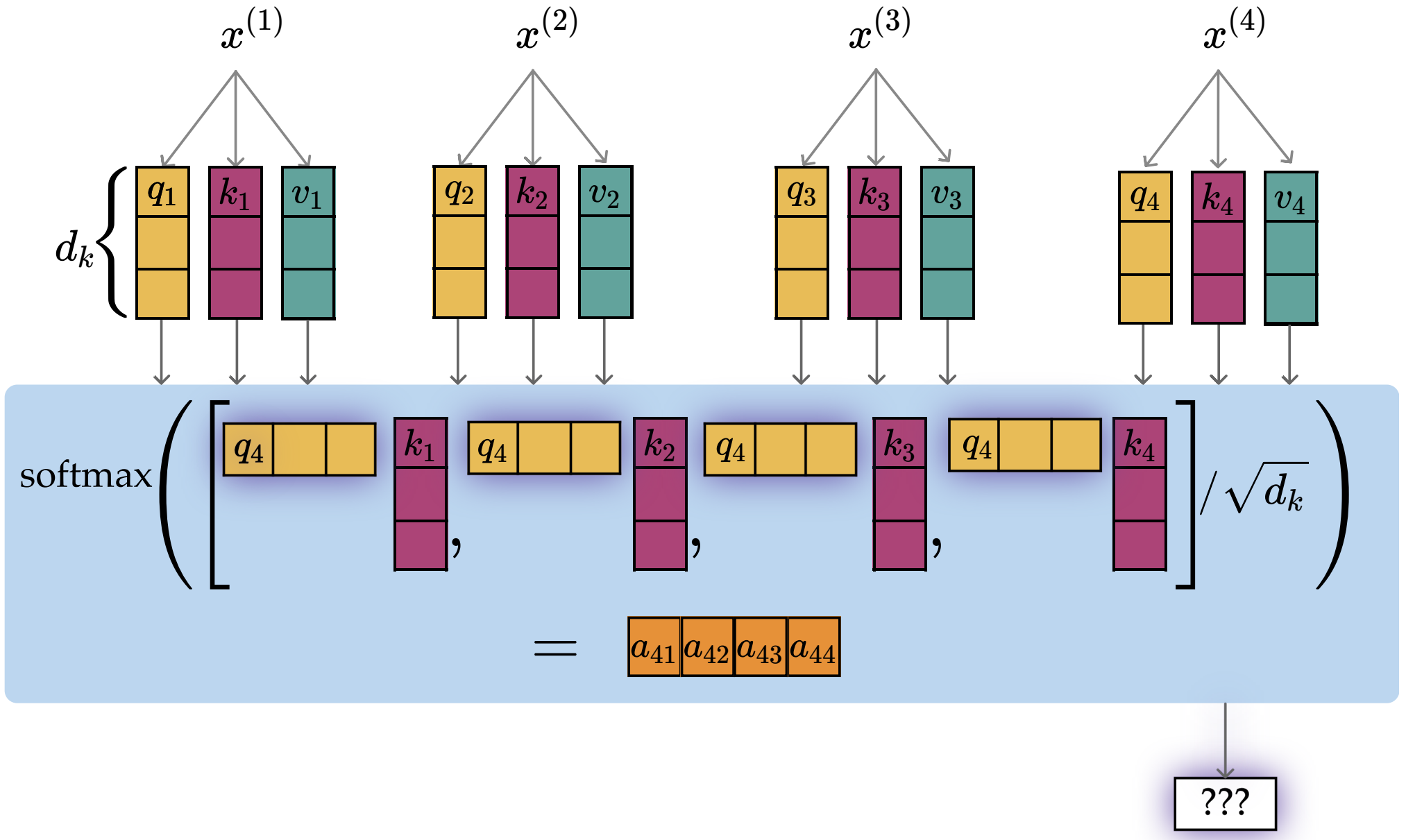

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
q_2 k_1 q_2 k_2 q_2 k_3 q_2 k_4
$/\sqrt{d_k}$
$=$
a_{21} a_{22} a_{23} a_{24}
???
$=$
a_{21} v_1 + a_{22} v_2 + a_{23} v_3 + a_{24} v_4

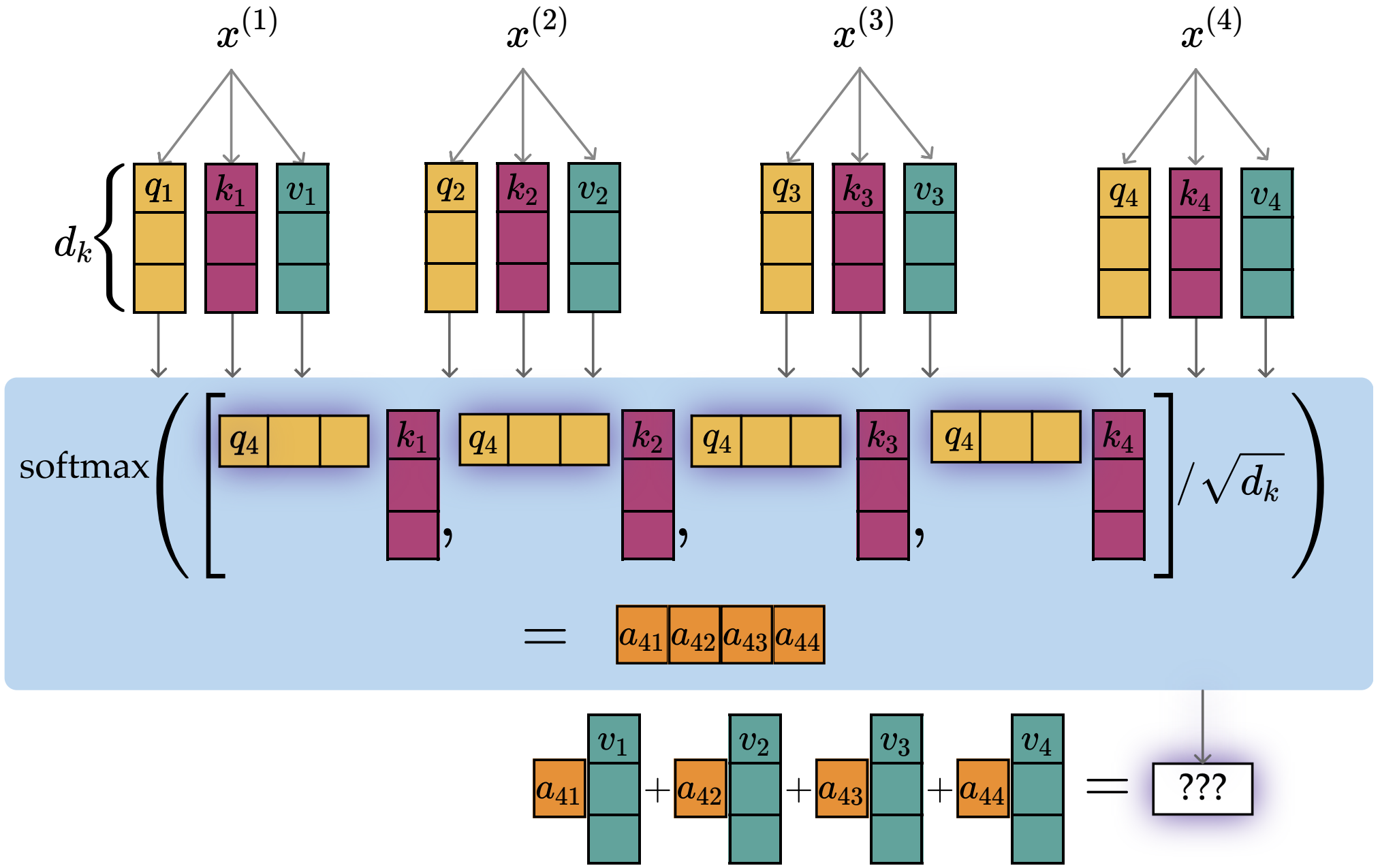

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
d_k
q_1
k_1
v_1
q_2
k_2
v_2
q_3
k_3
v_3
q_4
k_4
v_4
attention mechanism
...
...
???


$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
q_3 k_1 q_3 k_2 q_3 k_3 q_3 k_4
$/\sqrt{d_k}$
$=$
a_{31} a_{32} a_{33} a_{34}
???


$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
q_3 k_1 q_3 k_2 q_3 k_3 q_3 k_4
$/\sqrt{d_k}$
$=$
a_{31} a_{32} a_{33} a_{34}
???
$=$
a_{31} v_1 $+$ a_{32} v_2 $+$ a_{33} v_3 $+$ a_{34} v_4

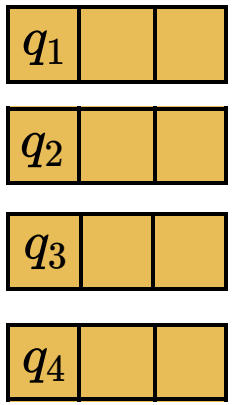

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
d_k
q_1
k_1
v_1
q_2
k_2
v_2
q_3
k_3
v_3
q_4
k_4
v_4
attention mechanism
...
...
...
???

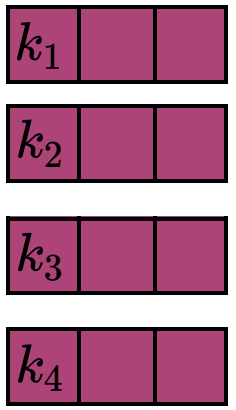

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
q_4 k_1 q_4 k_2 q_4 k_3 q_4 k_4
$/\sqrt{d_k}$
$=$
a_{41} a_{42} a_{43} a_{44}
???


$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
$/\sqrt{d_k}$
a_{41} a_{42} a_{43} a_{44}
???

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

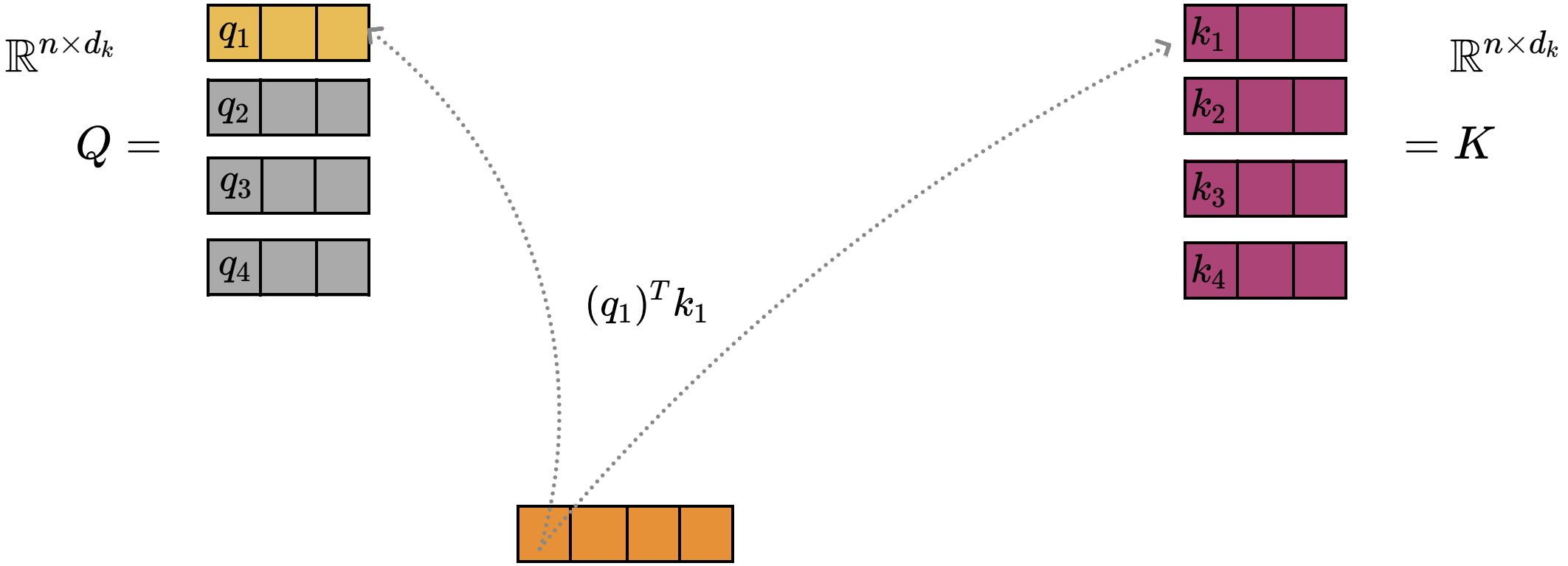
$Q =$

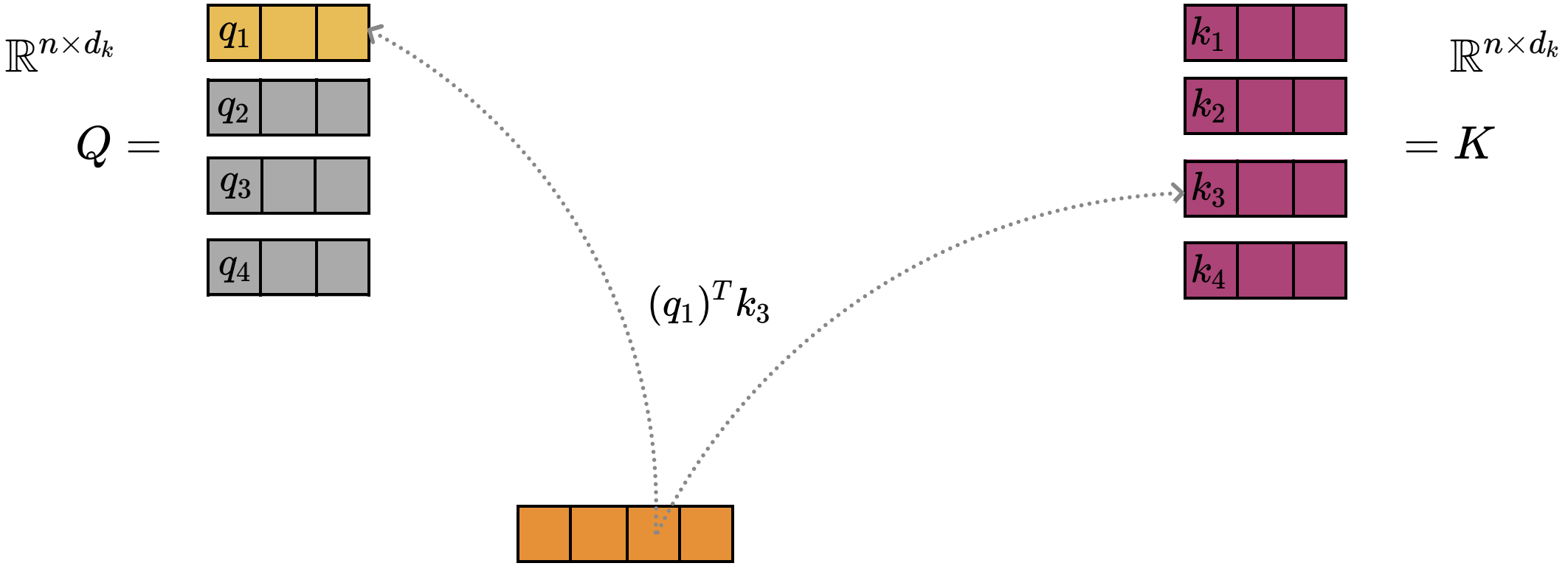


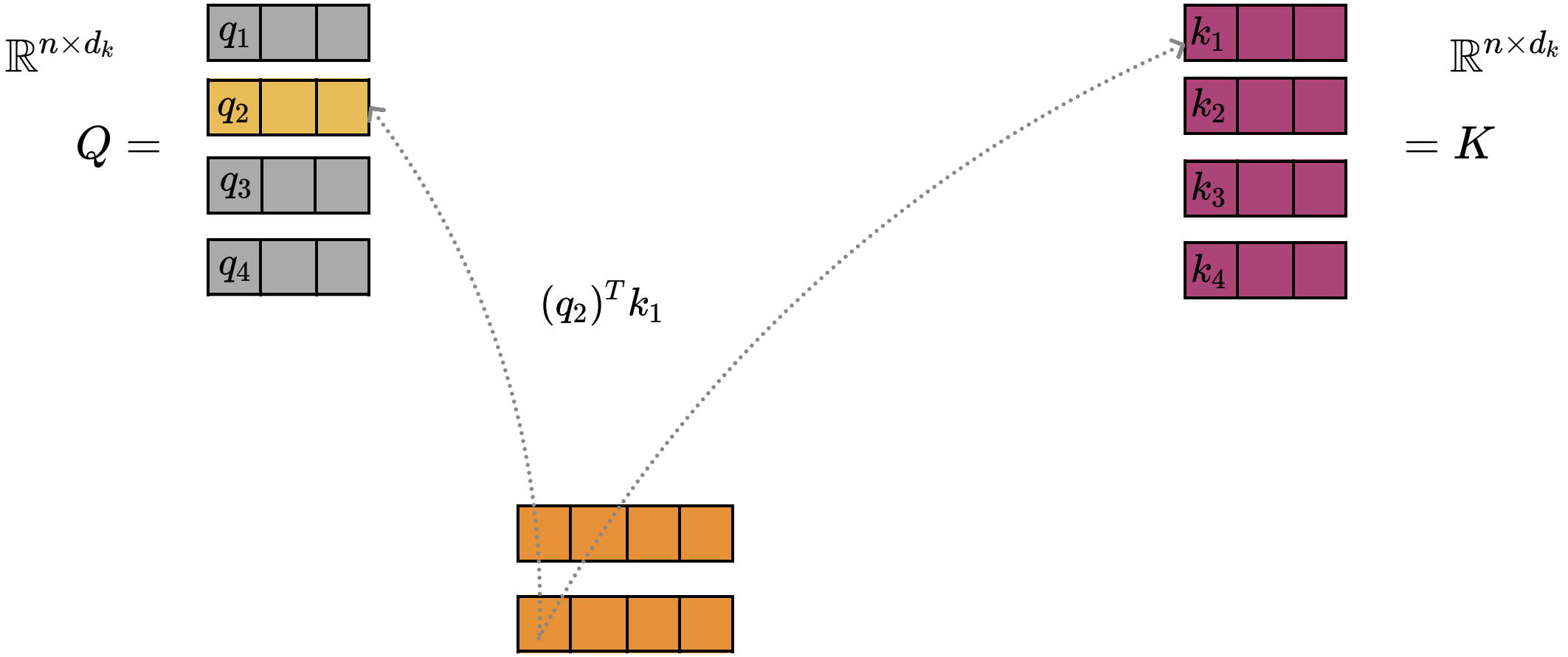


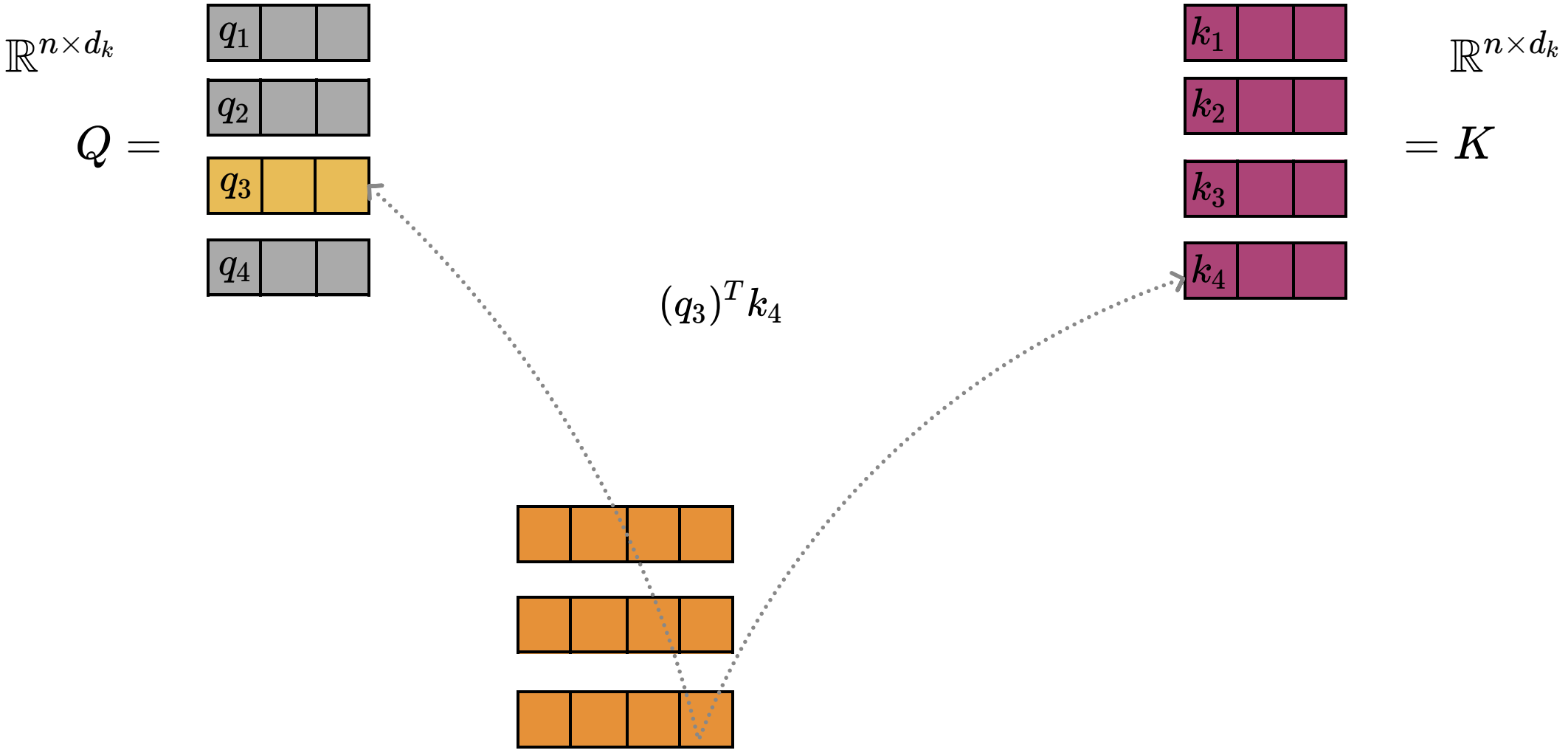
$\mathbb{R}^{n \times d_k}$

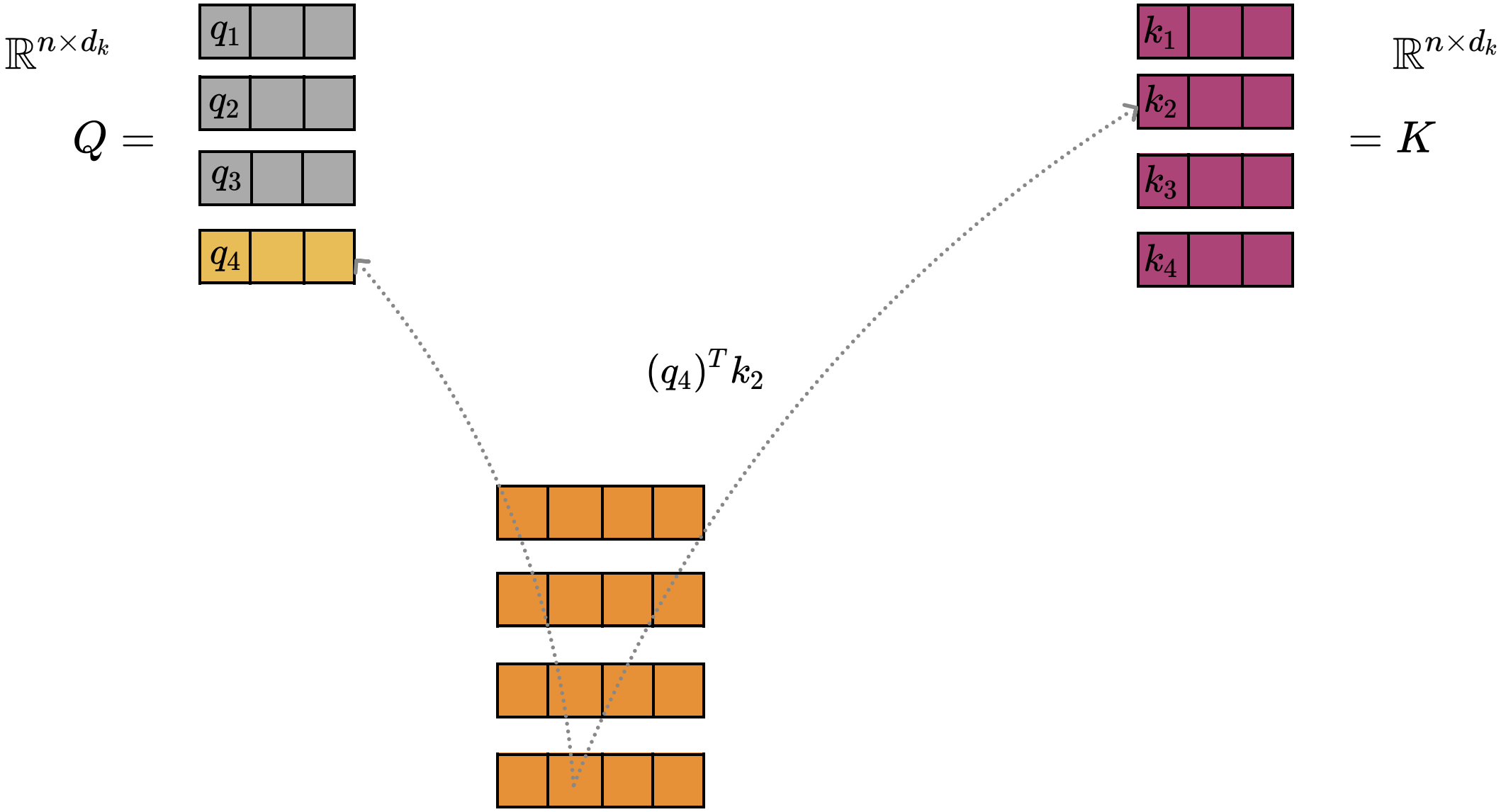
$= K$

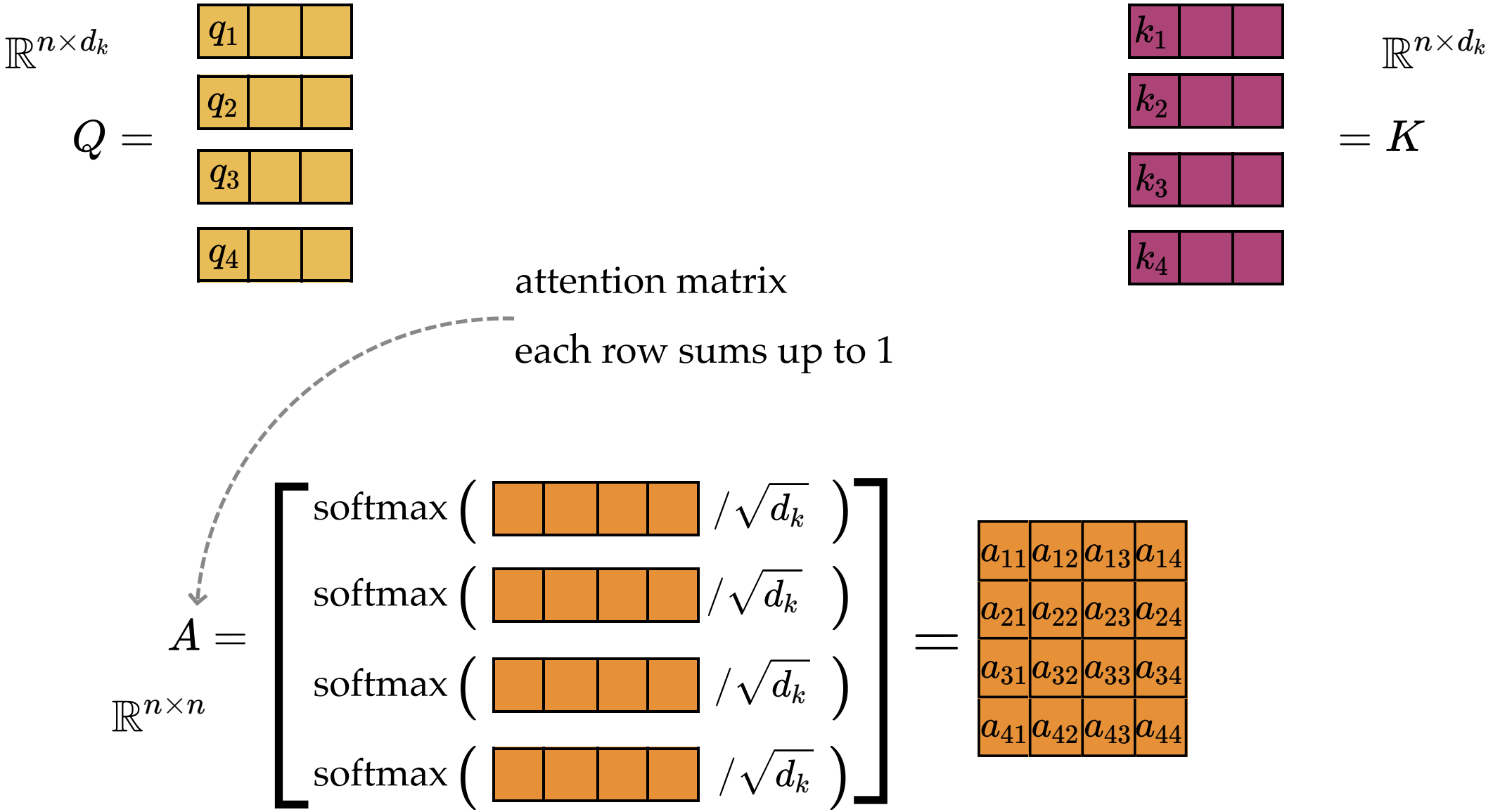

$\mathbb{R}^{n \times d_k}$
$Q =$
q_1
q_2
q_3
q_4
$(q_1)^T k_1$
k_1
k_2
k_3
k_4
$\mathbb{R}^{n \times d_k}$
$= K$

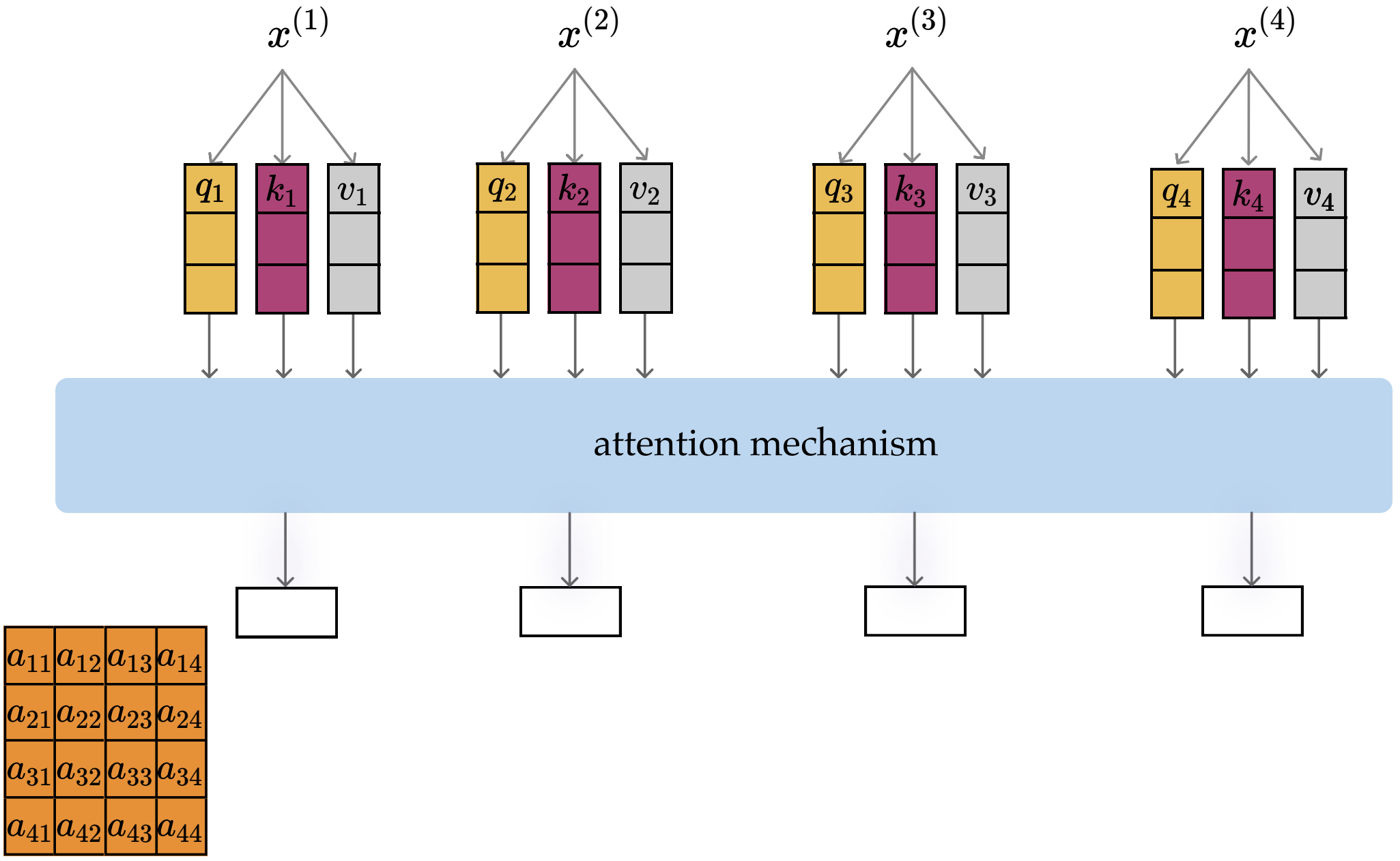

$\mathbb{R}^{n \times d_k}$
$Q =$
q_1
q_2
q_3
q_4
$(q_1)^T k_3$
k_1
k_2
k_3
k_4
$\mathbb{R}^{n \times d_k}$
$= K$

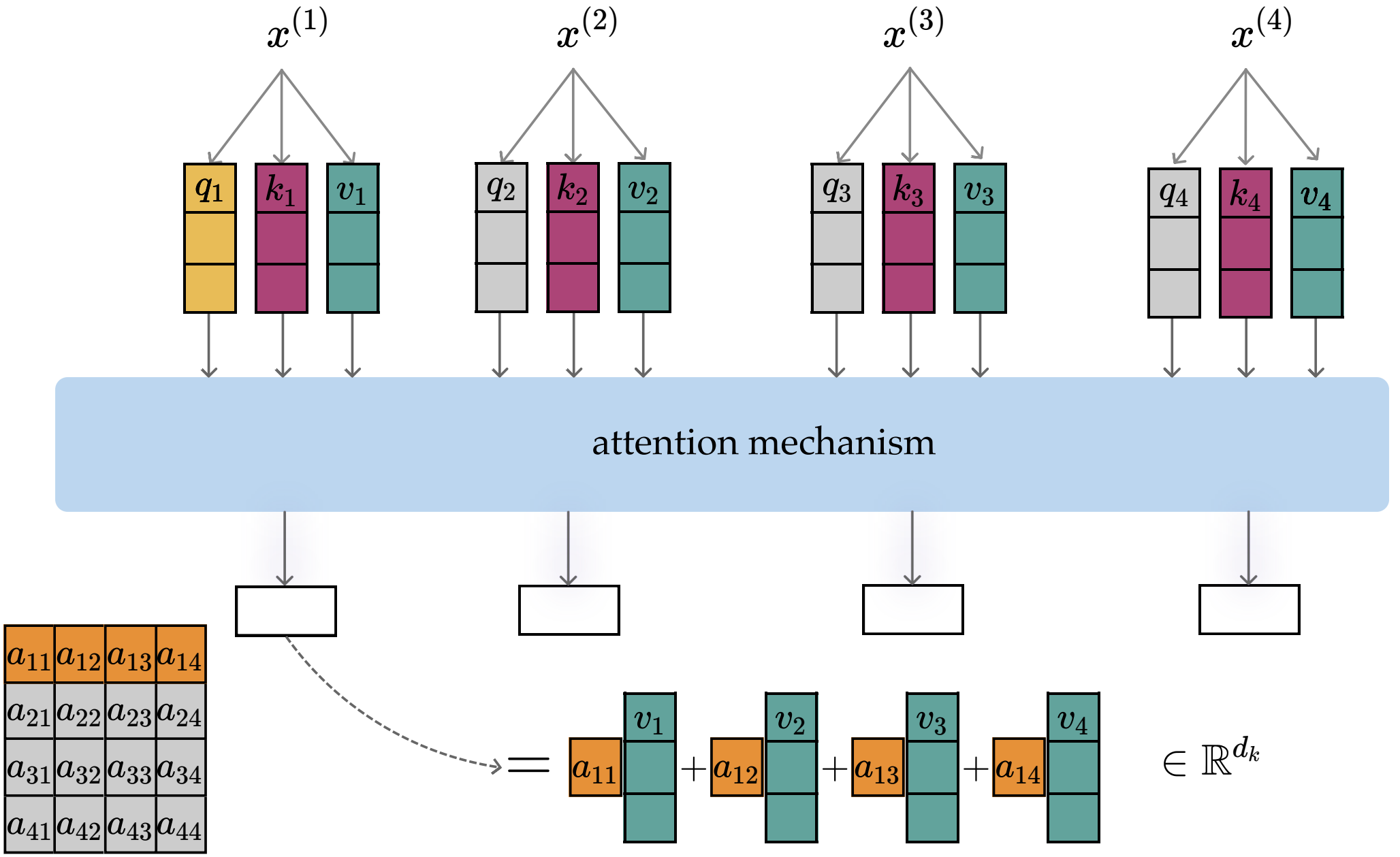

$\mathbb{R}^{n \times d_k}$
$Q =$
q_1
q_2
q_3
q_4
$(q_2)^T k_1$
k_1
k_2
k_3
k_4
$\mathbb{R}^{n \times d_k}$
$= K$

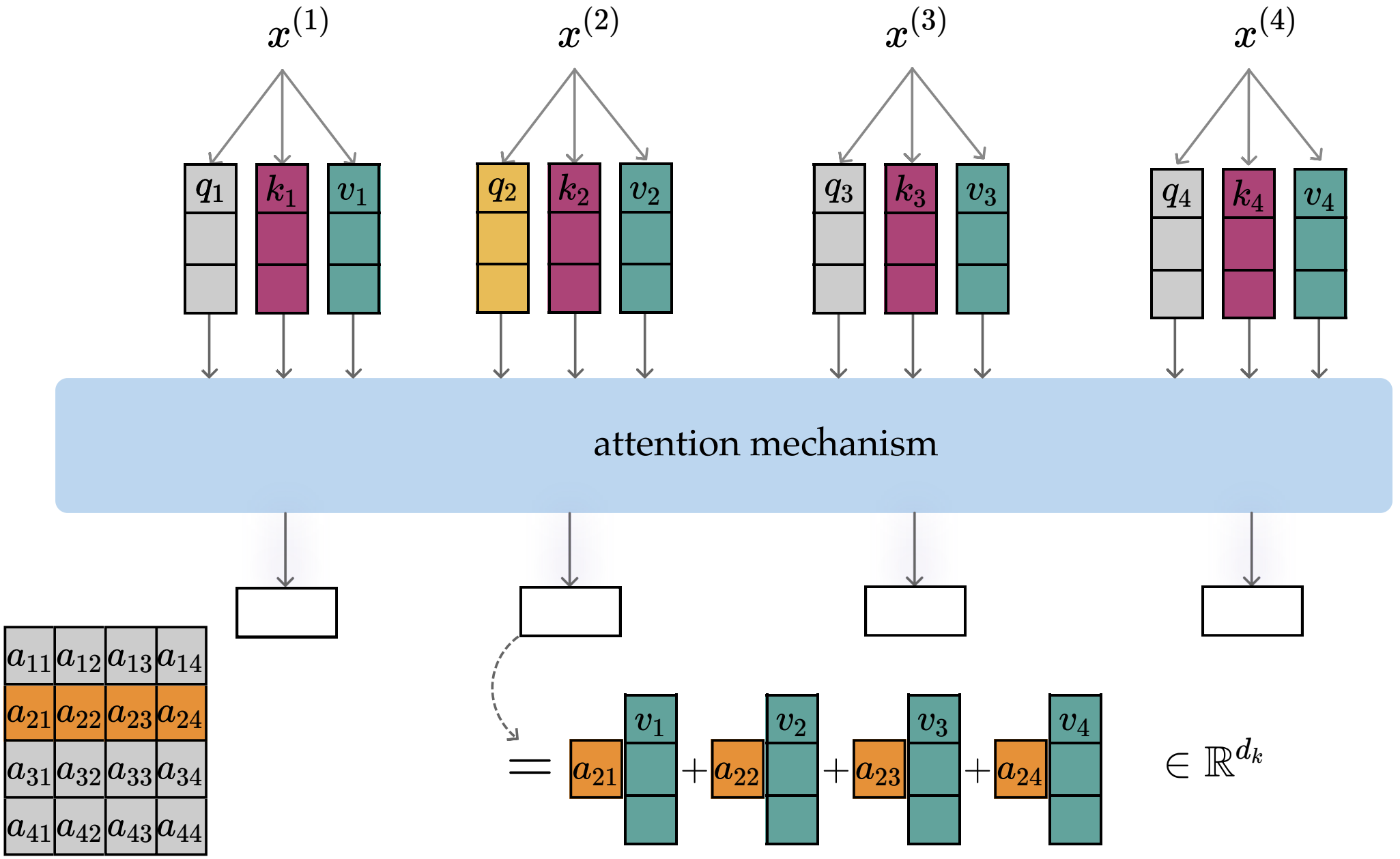

$\mathbb{R}^{n \times d_k}$
$Q =$
q_1
q_2
q_3
q_4
$(q_3)^T k_4$
k_1
k_2
k_3
k_4
$\mathbb{R}^{n \times d_k}$
$= K$

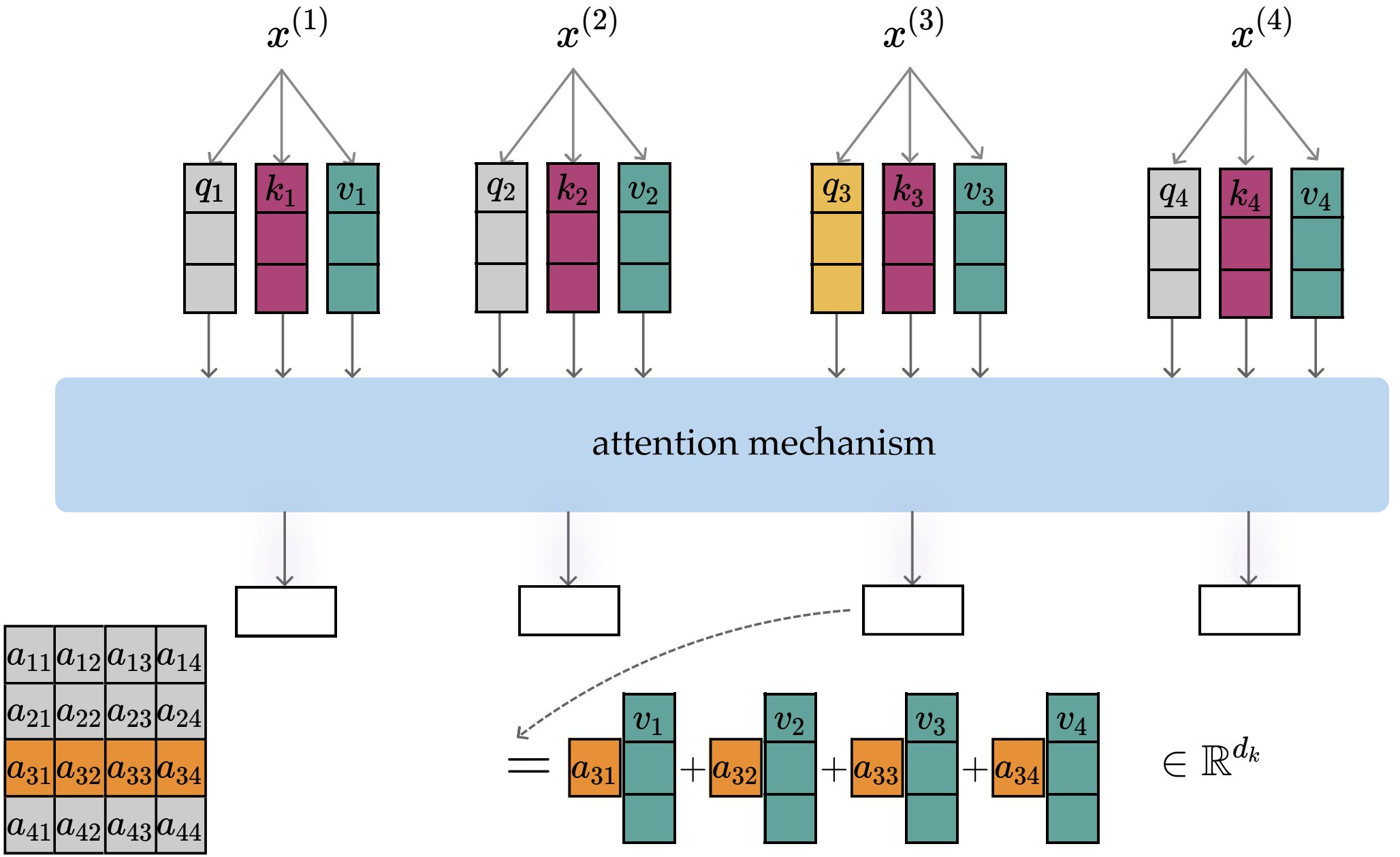

$\mathbb{R}^{n \times d_k}$
$Q =$
q_1
q_2
q_3
q_4
k_1
k_2
k_3
k_4
$\mathbb{R}^{n \times d_k}$
$= K$
$(q_4)^T k_2$

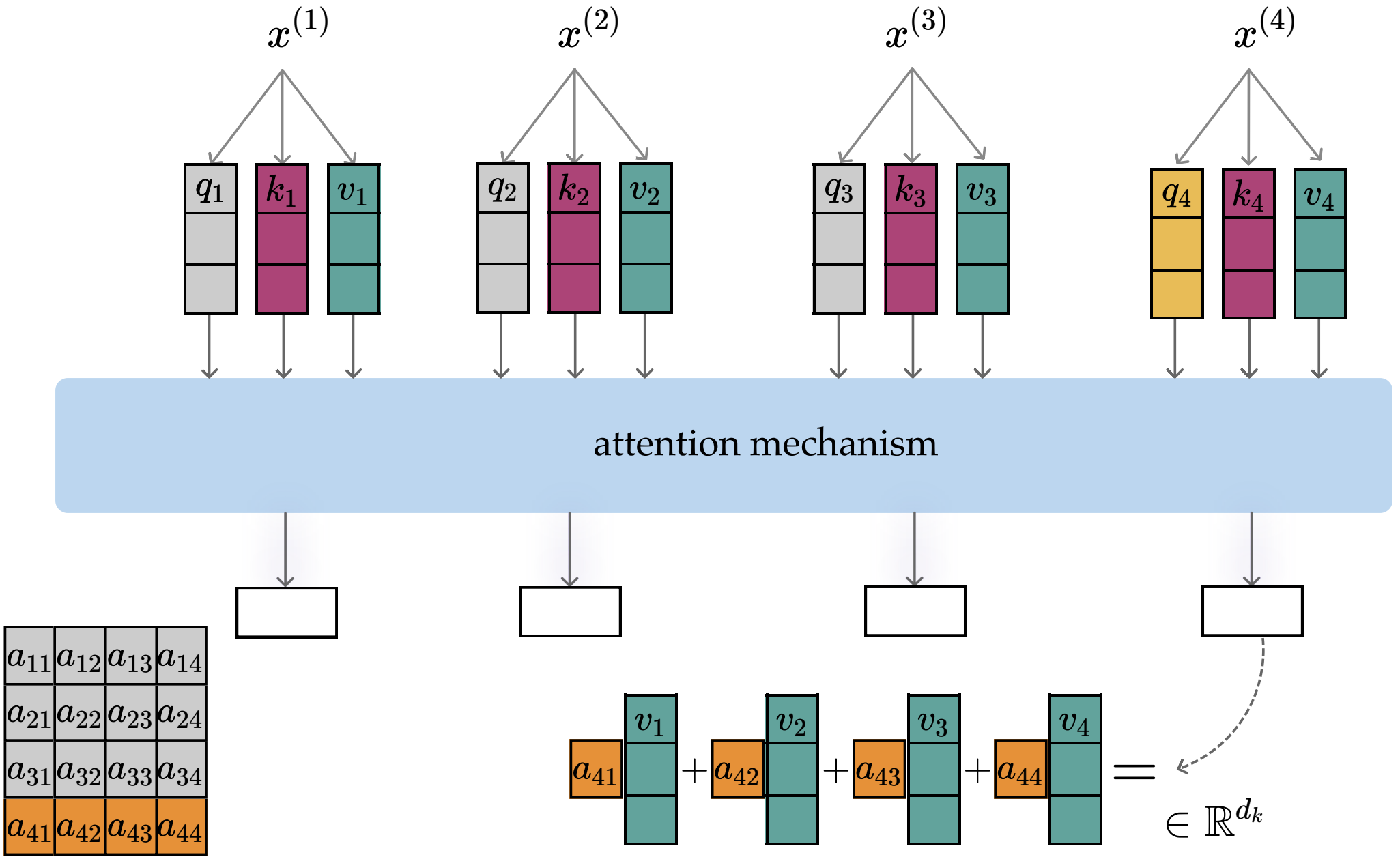

$\mathbb{R}^{n \times d_k}$
$Q =$
q_1
q_2
q_3
q_4
k_1
k_2
k_3
k_4
$\mathbb{R}^{n \times d_k}$
$= K$
attention matrix
each row sums up to 1
$A =$
$\mathbb{R}^{n \times n}$
softmax $(\quad / \sqrt{d_k} \;)$
$=$
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}

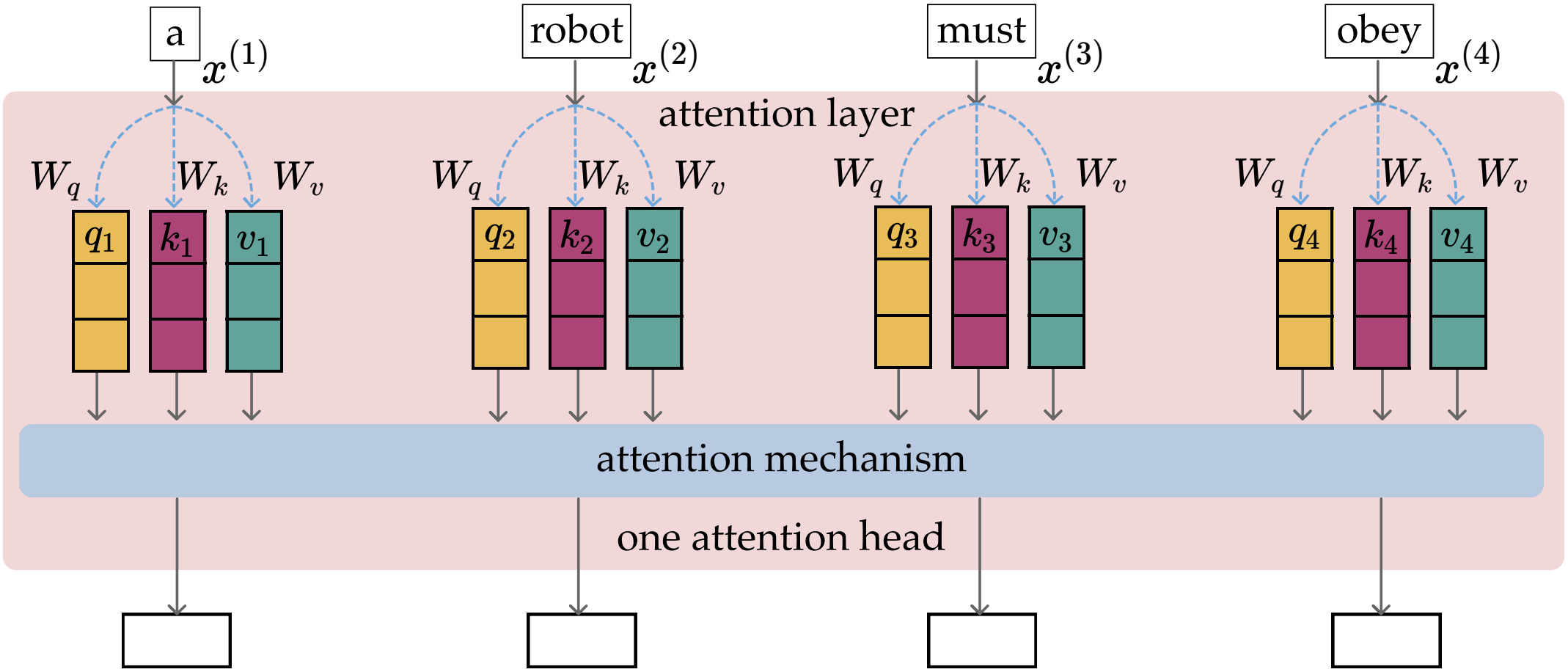

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}

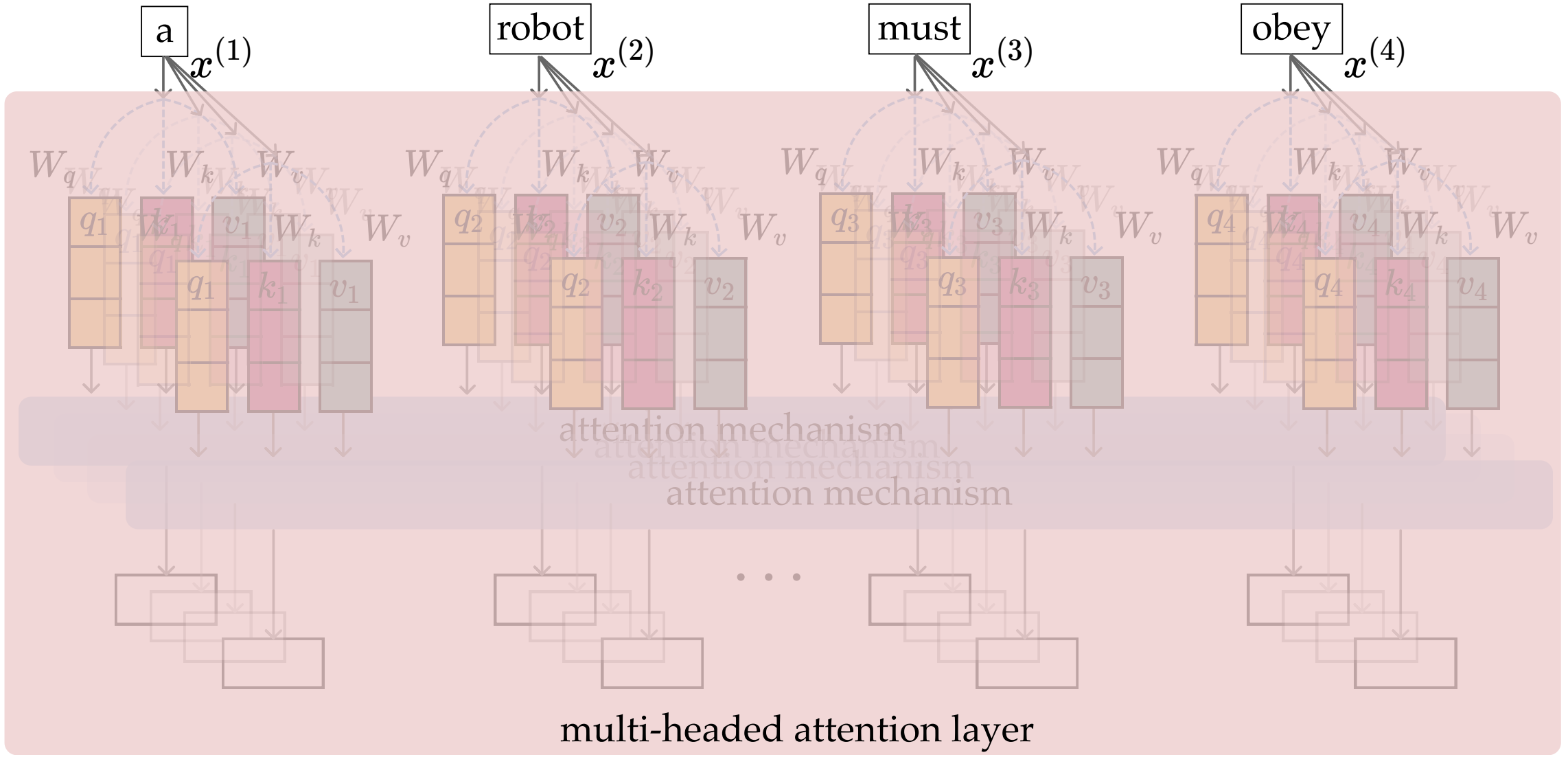

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}
$= a_{11} v_1 + a_{12} v_2 + a_{13} v_3 + a_{14} v_4$
$\in \mathbb{R}^{d_k}$


$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
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}
$= a_{21} v_1 + a_{22} v_2 + a_{23} v_3 + a_{24} v_4$
$\in \mathbb{R}^{d_k}$


$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
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}
$= a_{31} v_1 + a_{32} v_2 + a_{33} v_3 + a_{34} v_4$
$\in \mathbb{R}^{d_k}$


$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
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_{41} v_1 + a_{42} v_2 + a_{43} v_3 + a_{44} v_4 =
$\in \mathbb{R}^{d_k}$

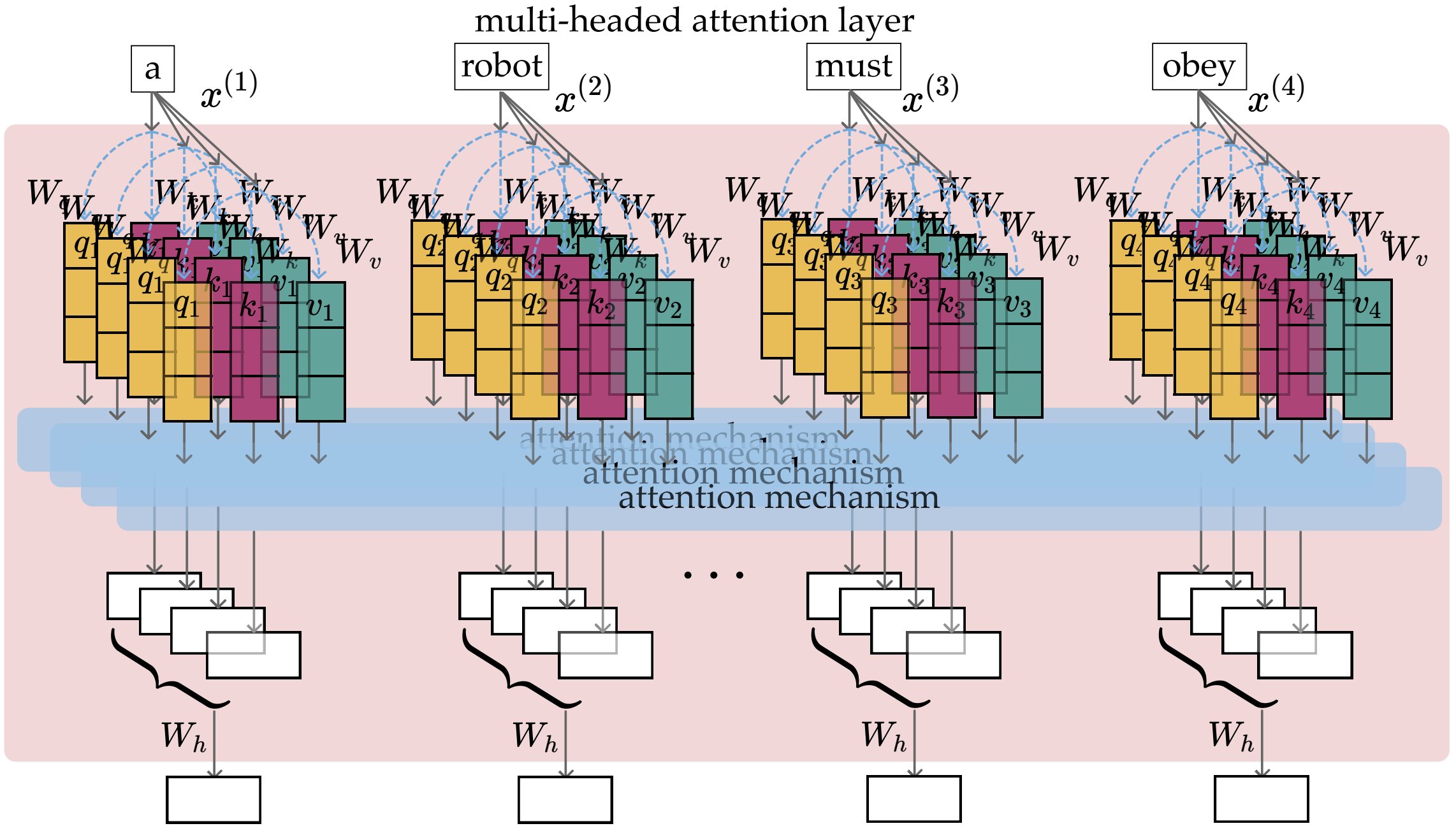

a
robot
must
obey
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
attention layer
W_q W_k W_v
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
one attention head

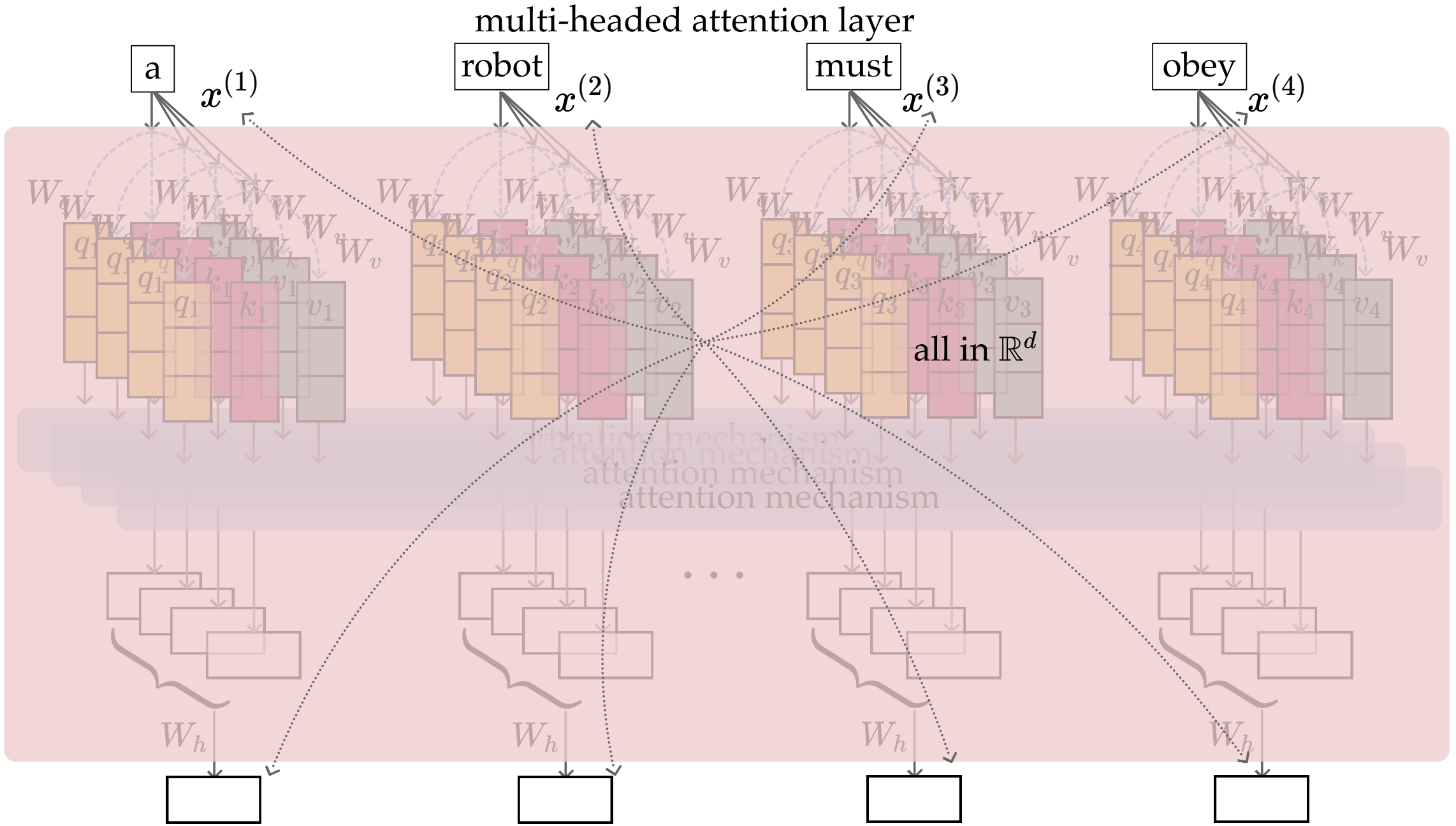

a
robot
must
obey
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
W_q
W_k
W_v
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
attention mechanism
attention mechanism
attention mechanism
multi-headed attention layer

each head

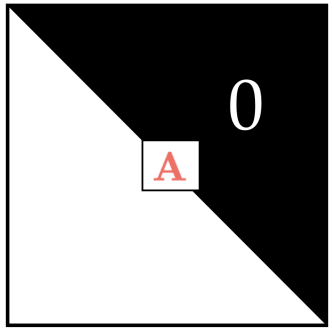
- can be processed in **parallel** to all other heads
- learns its own independent W_q, W_k, W_v
- creates its own (q, k, v) sequence
  - inside each head, the sequence of n (q, k, v) tokens can be processed in **parallel**
- computes its own attention output sequence
  - inside each head, the sequence of n output tokens can be processed in **parallel**

we then *learn* yet another weight W_h to sum up the outputs from individual head, to be the multi-headed attention layer output.

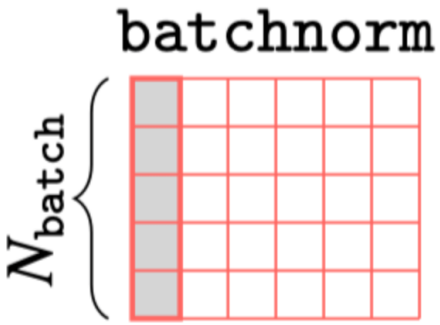

multi-headed attention layer
a
robot
must
obey
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
W_q
W_k
W_v
q_1
k_1
v_1
q_2
k_2
v_2
q_3
k_3
v_3
q_4
k_4
v_4
attention mechanism
attention mechanism
attention mechanism
attention mechanism
W_h

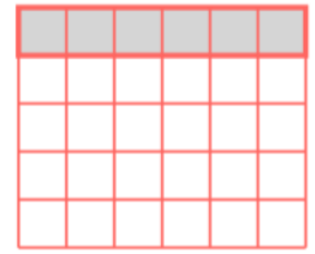

multi-headed attention layer
a
robot
must
obey
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
all in $\mathbb{R}^d$
attention mechanism
attention mechanism
attention mechanism
W_h

Some other ideas commonly used in practice:

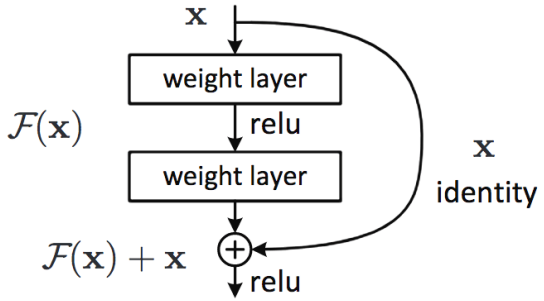


masking





Layer normlization



Residual connection

Positional encoding

We will see the details in hw/lab

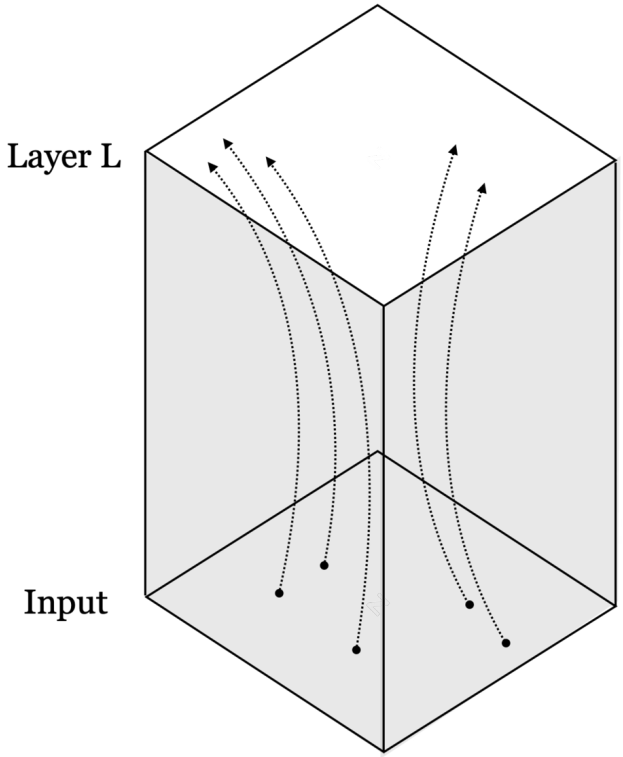
https://poloclub.github.io/transformer-explainer/

`Recall`

Neural networks are representation learners

Deep nets transform datapoints, layer by layer

Each layer gives a different *representation* (aka *embedding*) of the data



Recall

$$g = \text{softmax}(z_2)$$

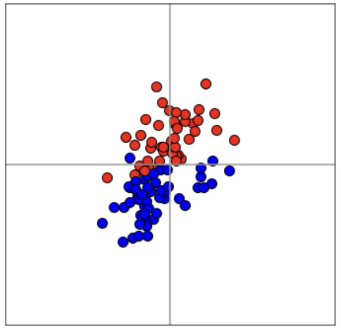
$$z_2 = \text{linear}\,(a_1)$$

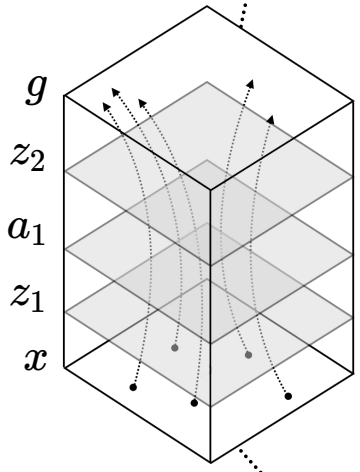
$$a_1 = \text{ReLU}(z_1)$$

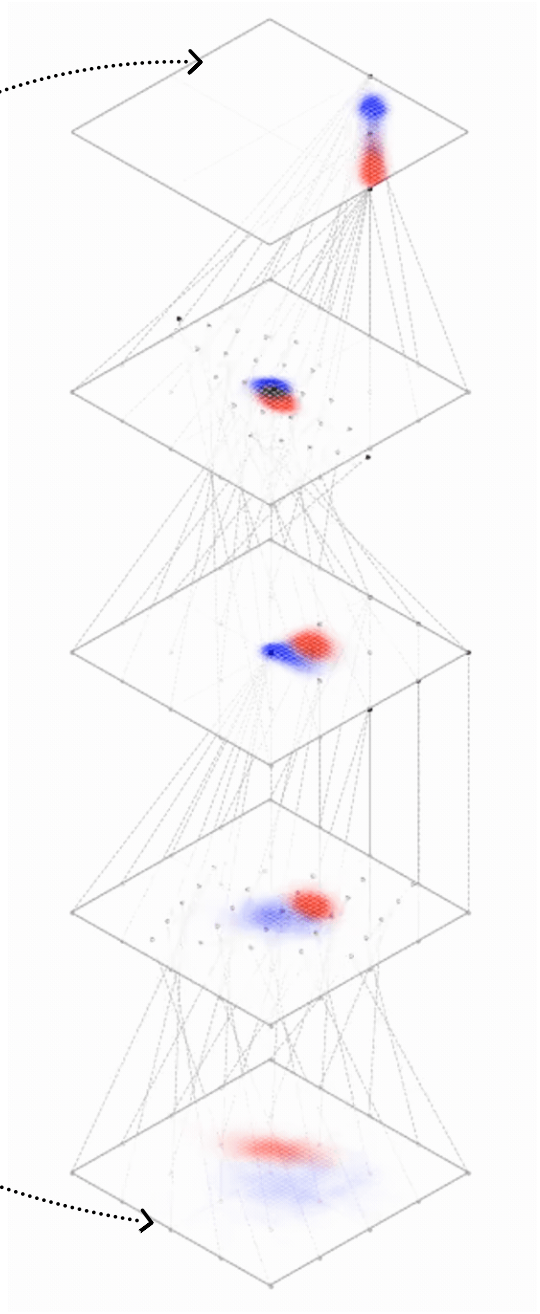
$$z_1 = \text{linear}\,(x)$$

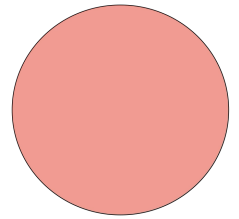
$$x \in \mathbb{R}^2$$

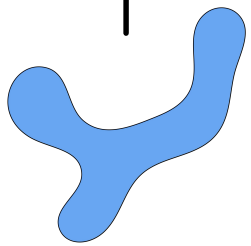
Training data



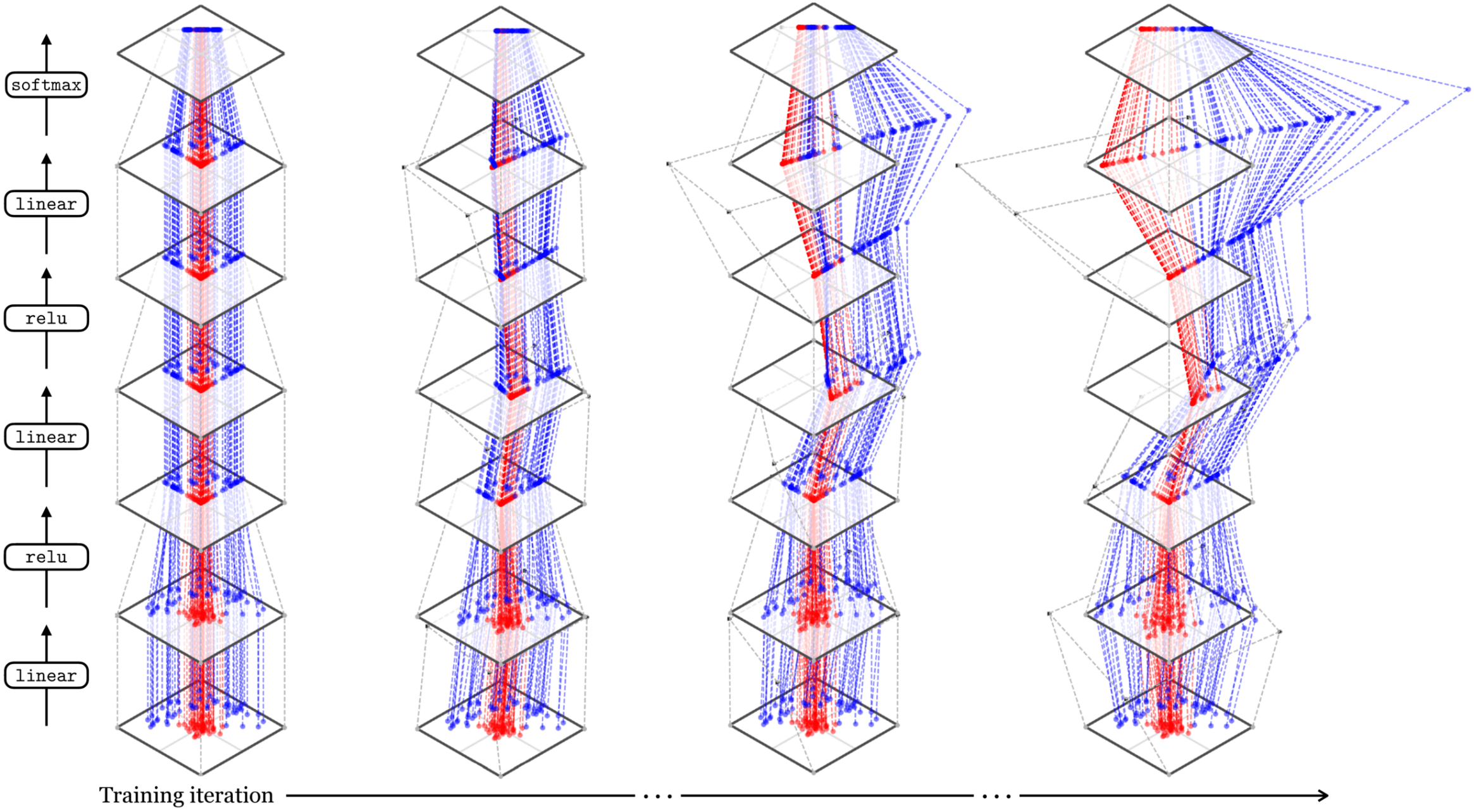








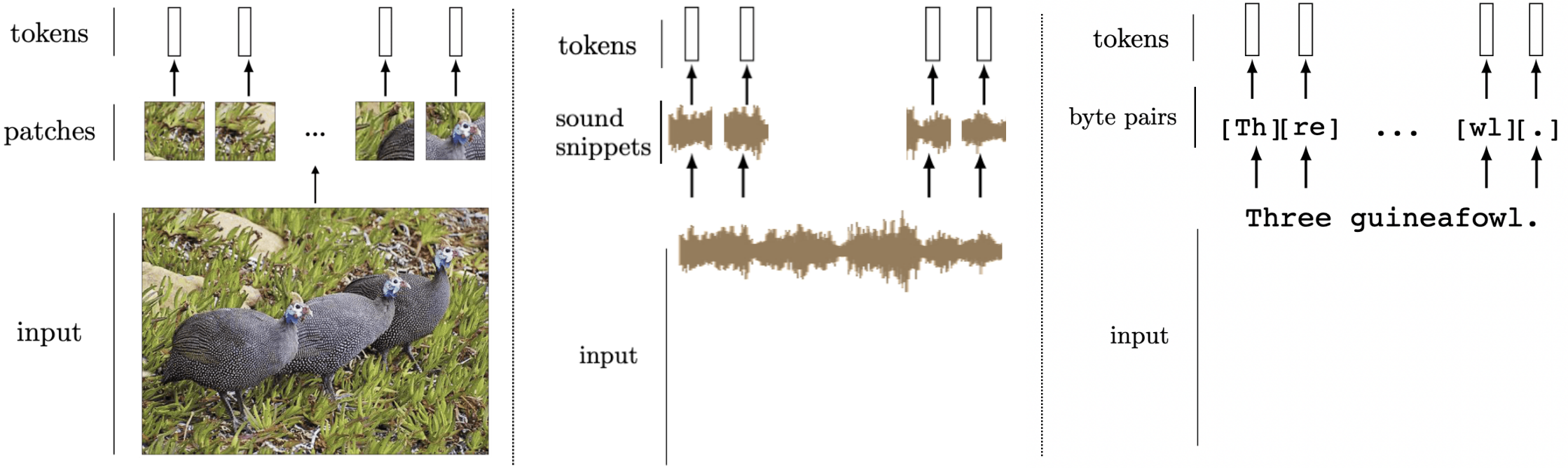
maps from complex data space to simple embedding space

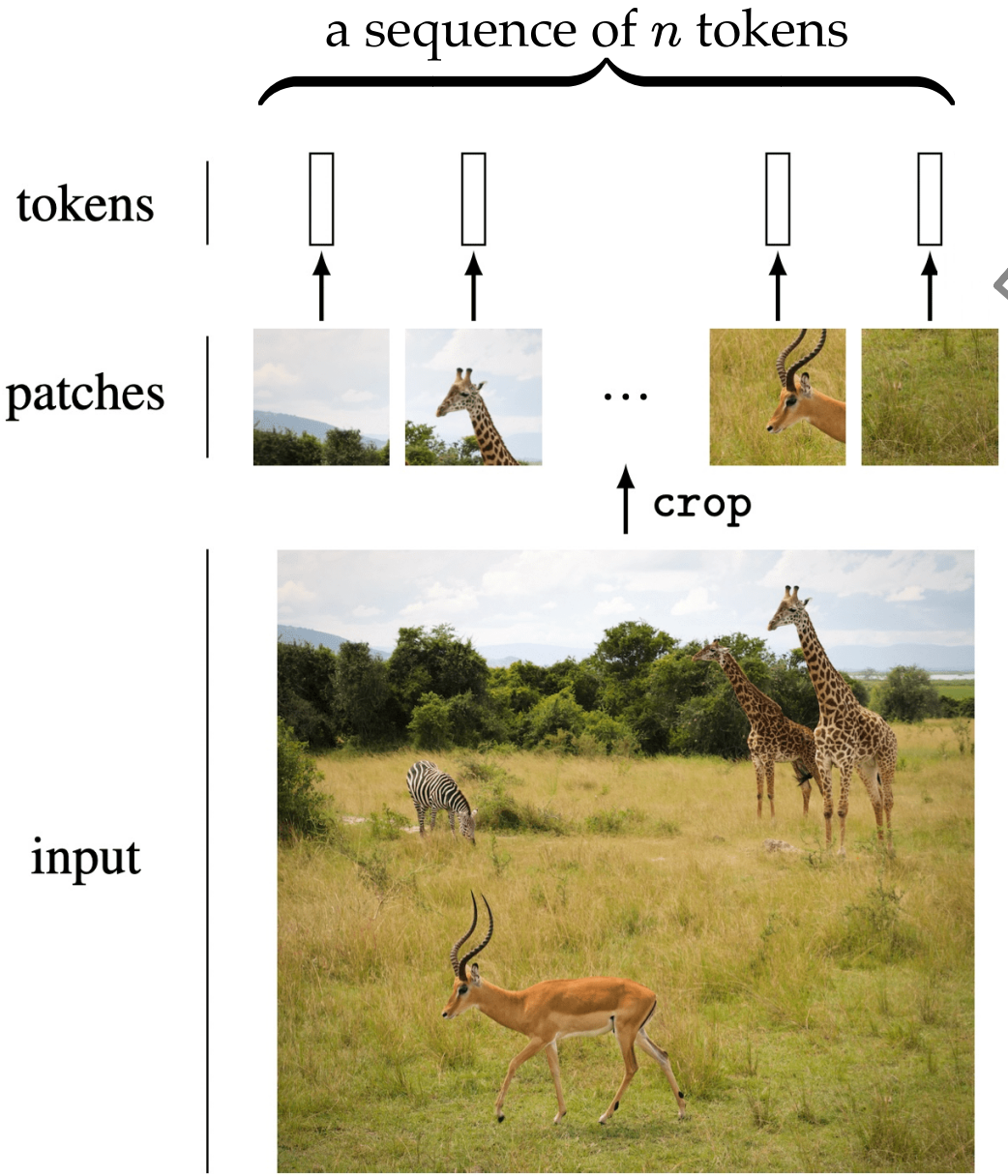

softmax
linear
relu
linear
relu
linear
Training iteration

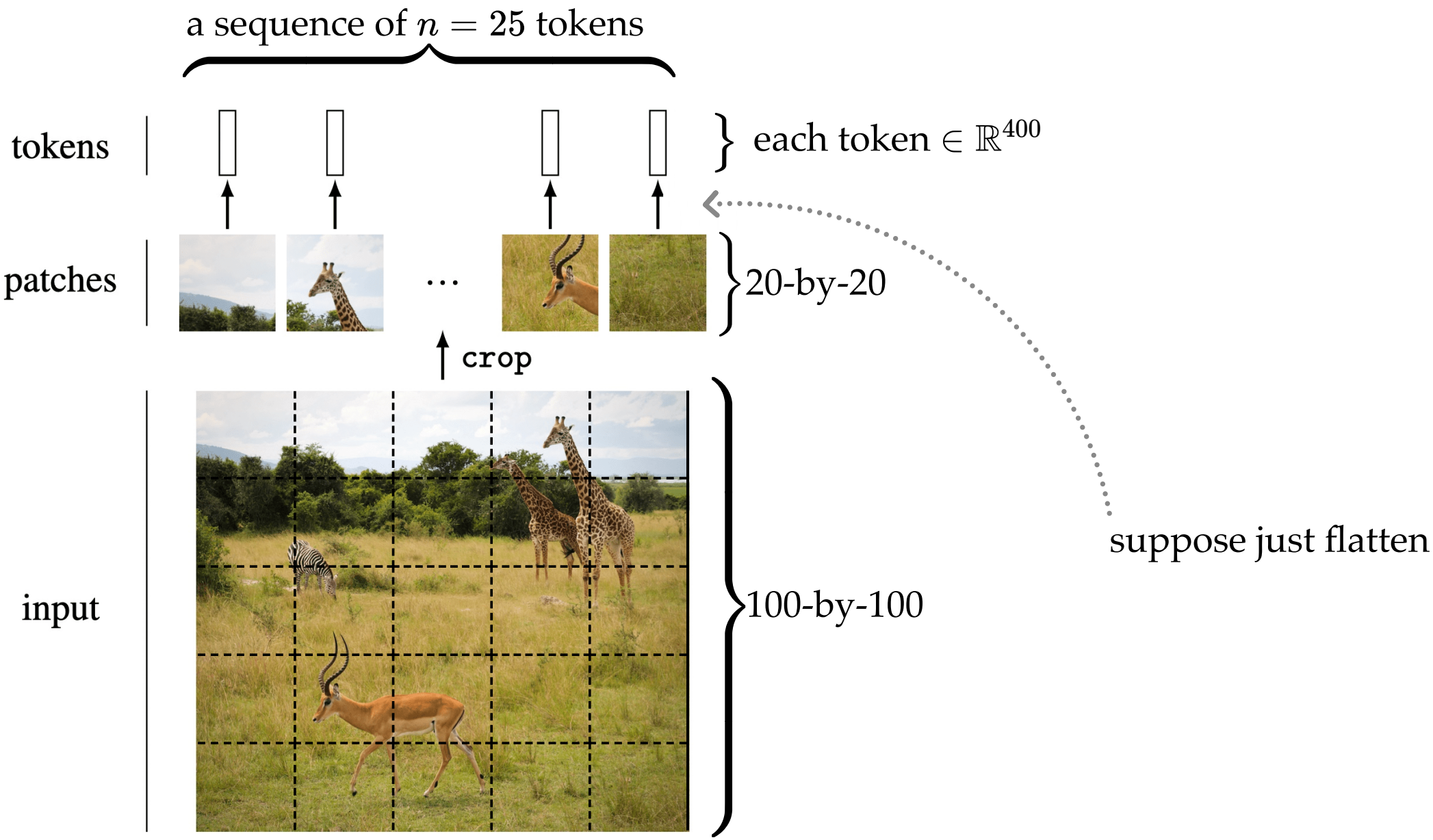
We can tokenize anything.

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

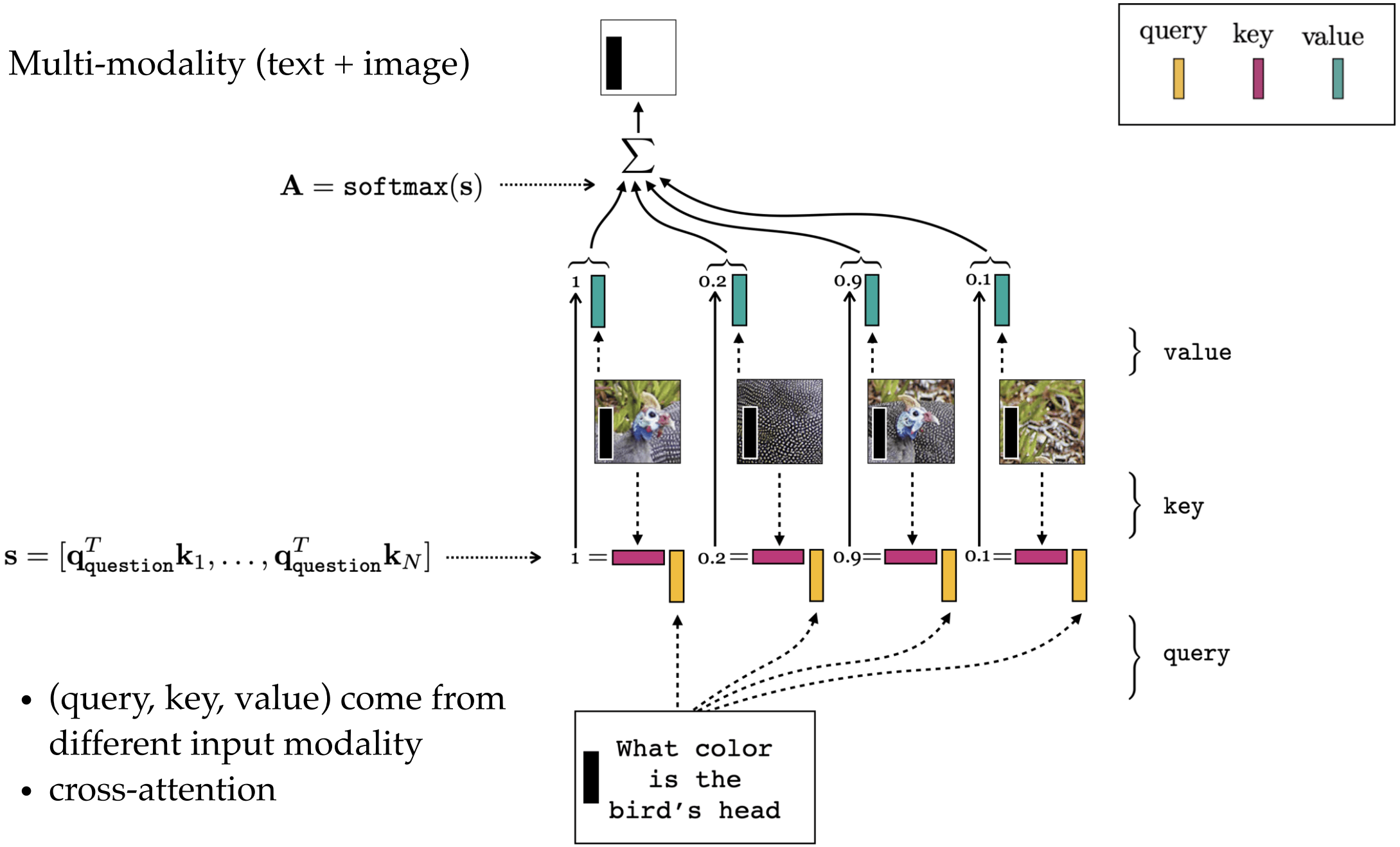
this projection can be fixed from a pre-trained model, or trained jointly with downstream task




a sequence of n tokens
tokens
patches
input
crop
each token $\in \mathbb{R}^d$ **embedding**
a projection, e.g. via a fixed, or learned linear transformation


a sequence of $n = 25$ tokens
tokens
each token $\in \mathbb{R}^{400}$
patches
20-by-20
crop
input
100-by-100
suppose just flatten

Multi-modality (text + image)

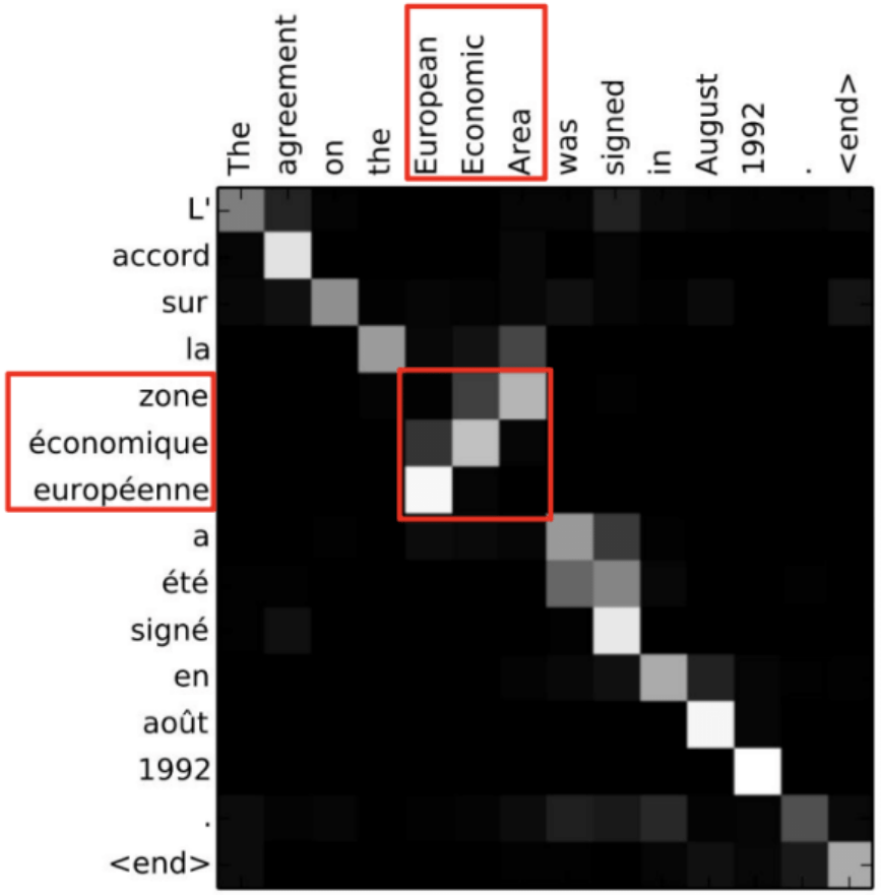


- (query, key, value) come from different input modality
- cross-attention

https://segment-anything.com/demo

Input sentence: “The agreement on the European Economic Area was signed in August 1992”

Output sentence: “L’accord sur la zone économique européenne a été signé en août 1992”



[Bahdanau et al. 2015]

Success mode:

[“DINO”, Caron et all. 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 ccetnxniams luryca tanniounons" means bugs or pests.

The prompt: "Apoploe vesrreaitais eating Contarra ccetnxniams 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)

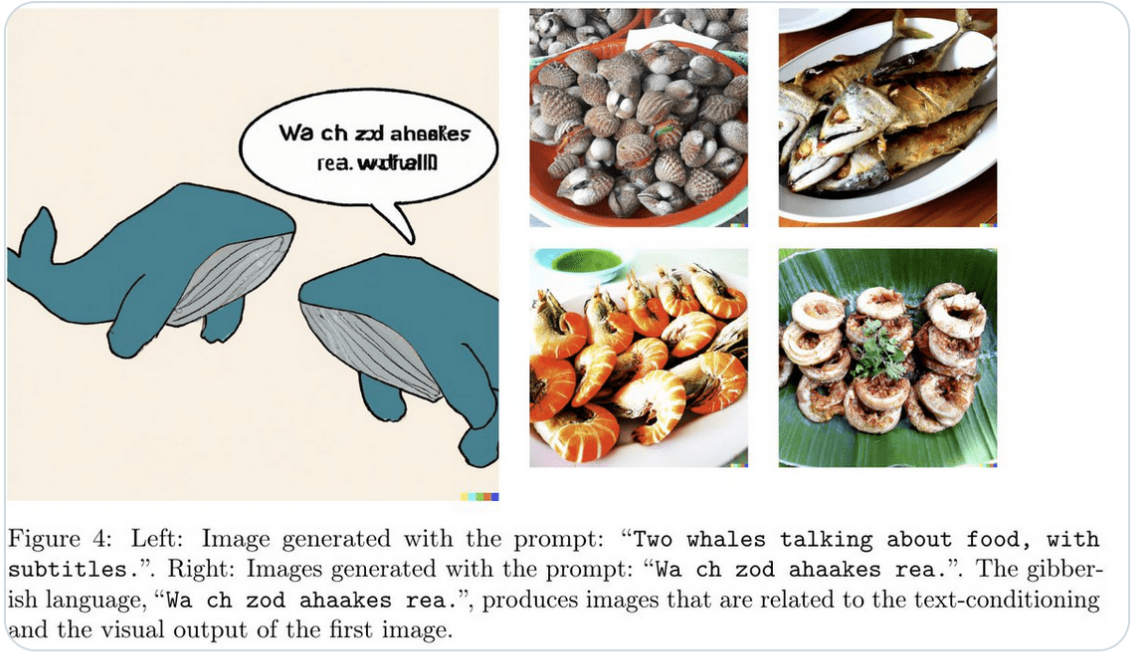


Figure 4: Left: Image generated with the prompt: "`Two whales talking about food, with subtitles.`". Right: Images generated with the prompt: "`Wa ch zod ahaakes rea.`". The gibberish language, "`Wa ch zod ahaakes rea.`", produces images that are related to the text-conditioning and the visual output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