Mechanistic interpretability

2

Language models

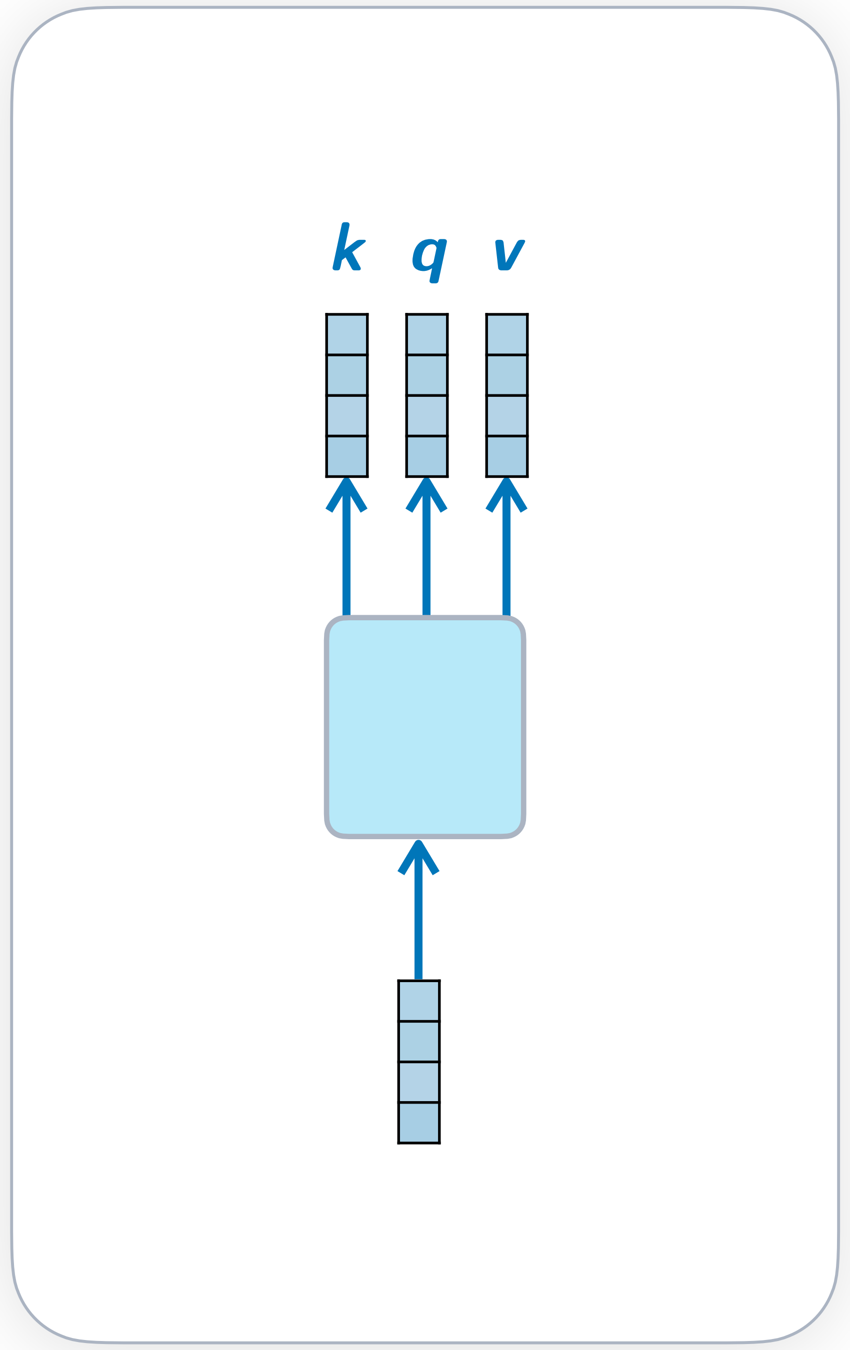
Causal Self-Attention

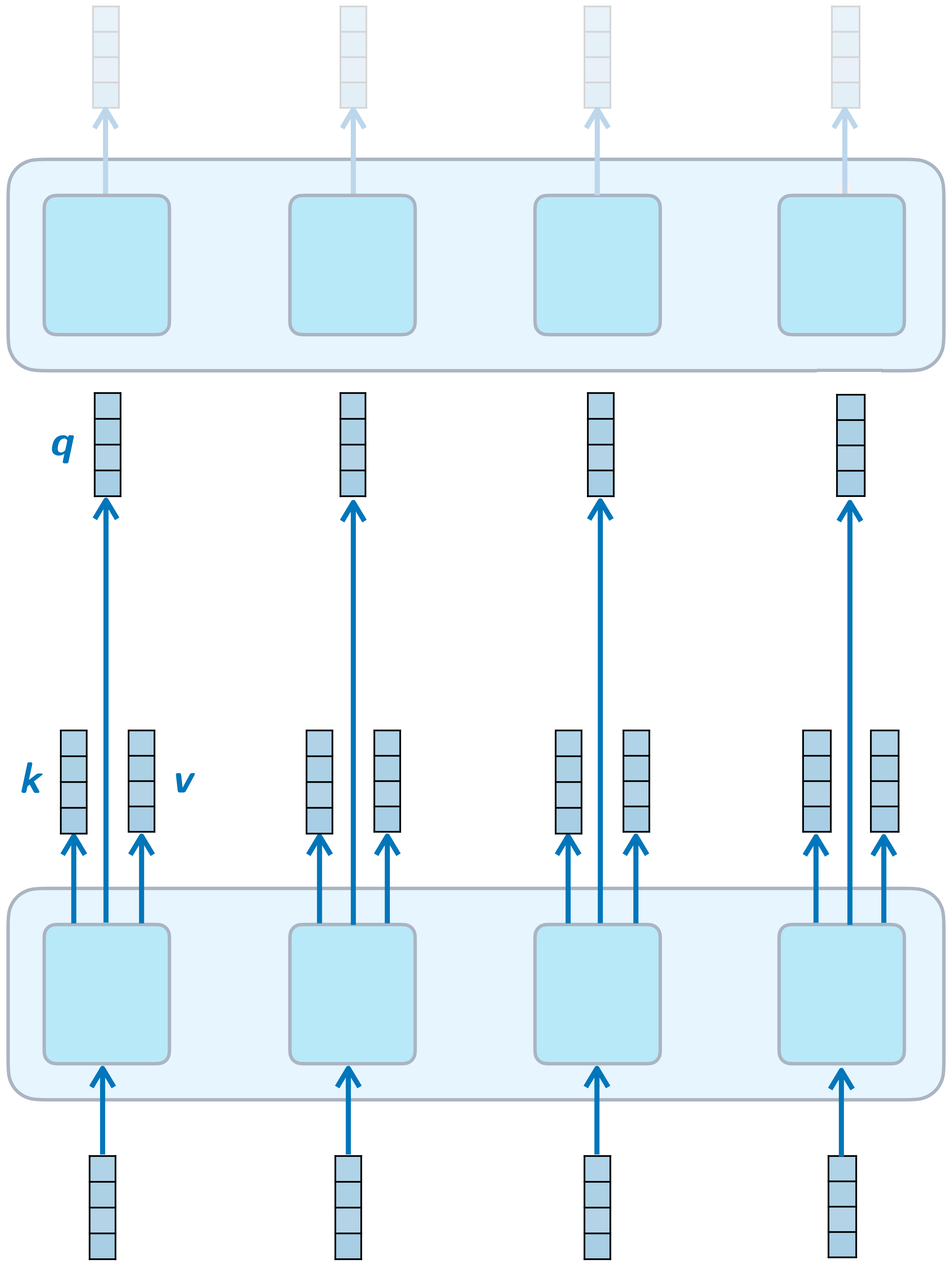
Think of ***N*-to-*N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

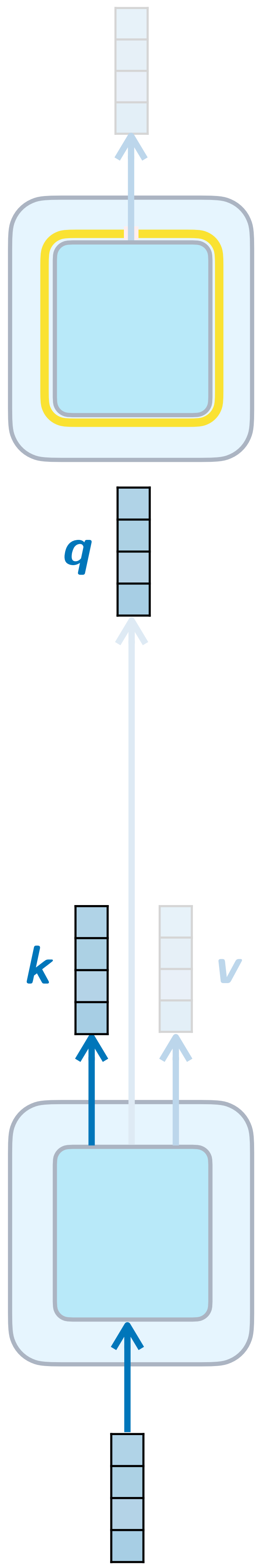
key and value vectors

Think of ***N-to-N***

Feed **decoder** information into **decoder**

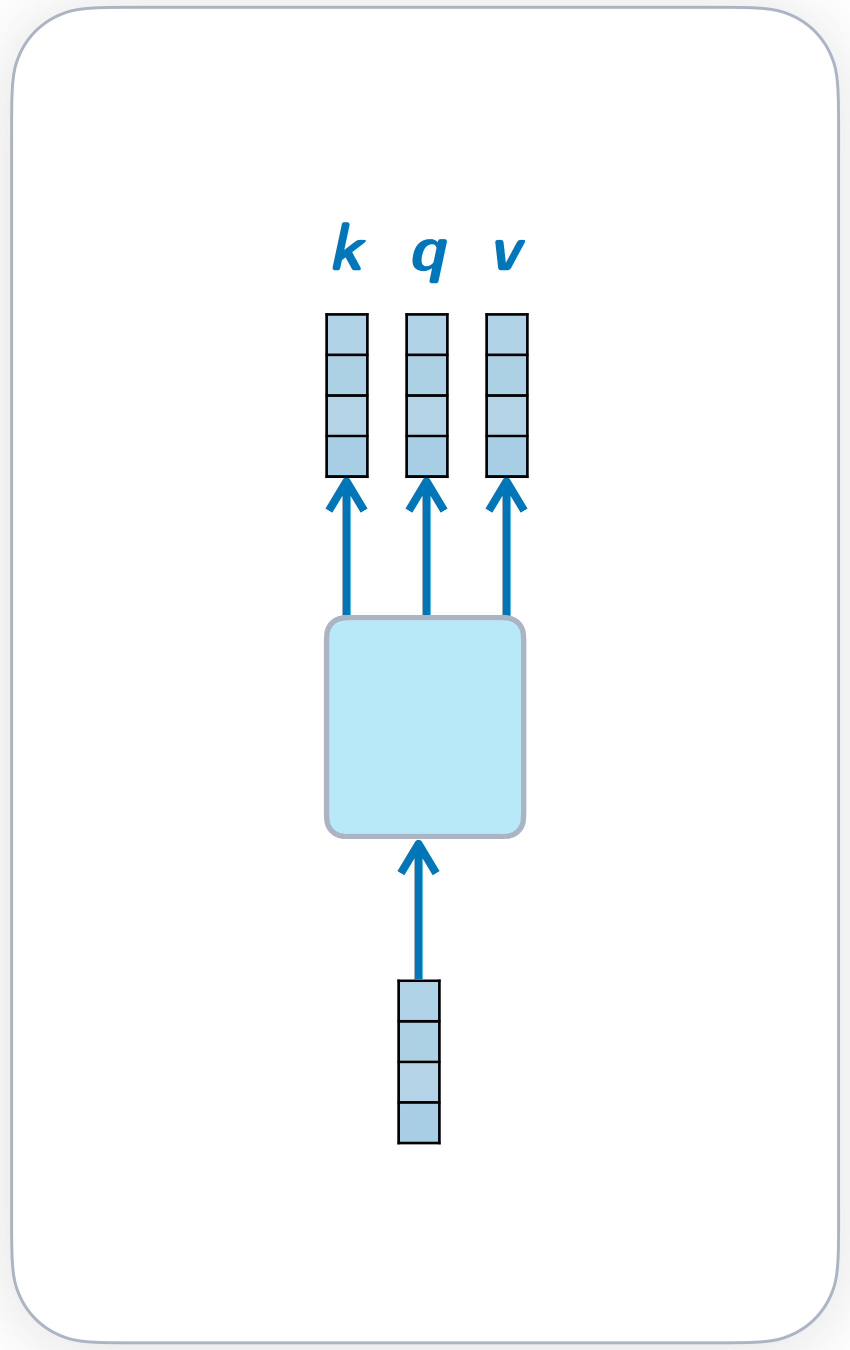
come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**



Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

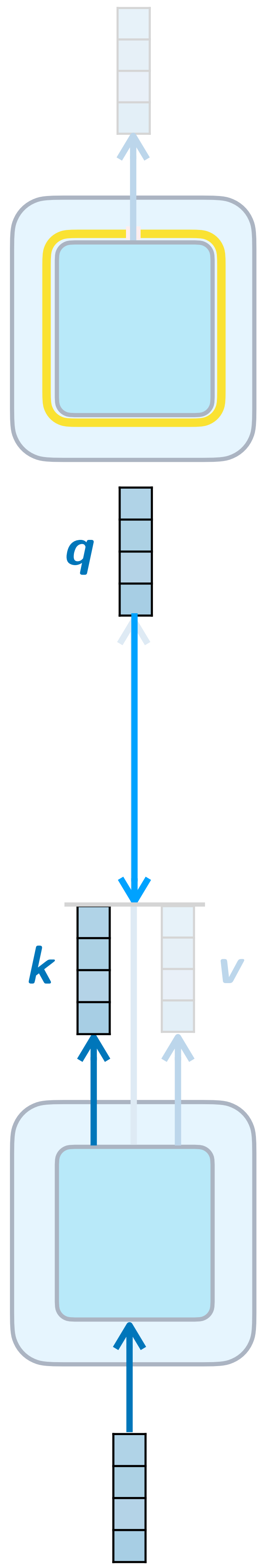


Think of ***N-to-N***

Feed **decoder** information into **decoder**

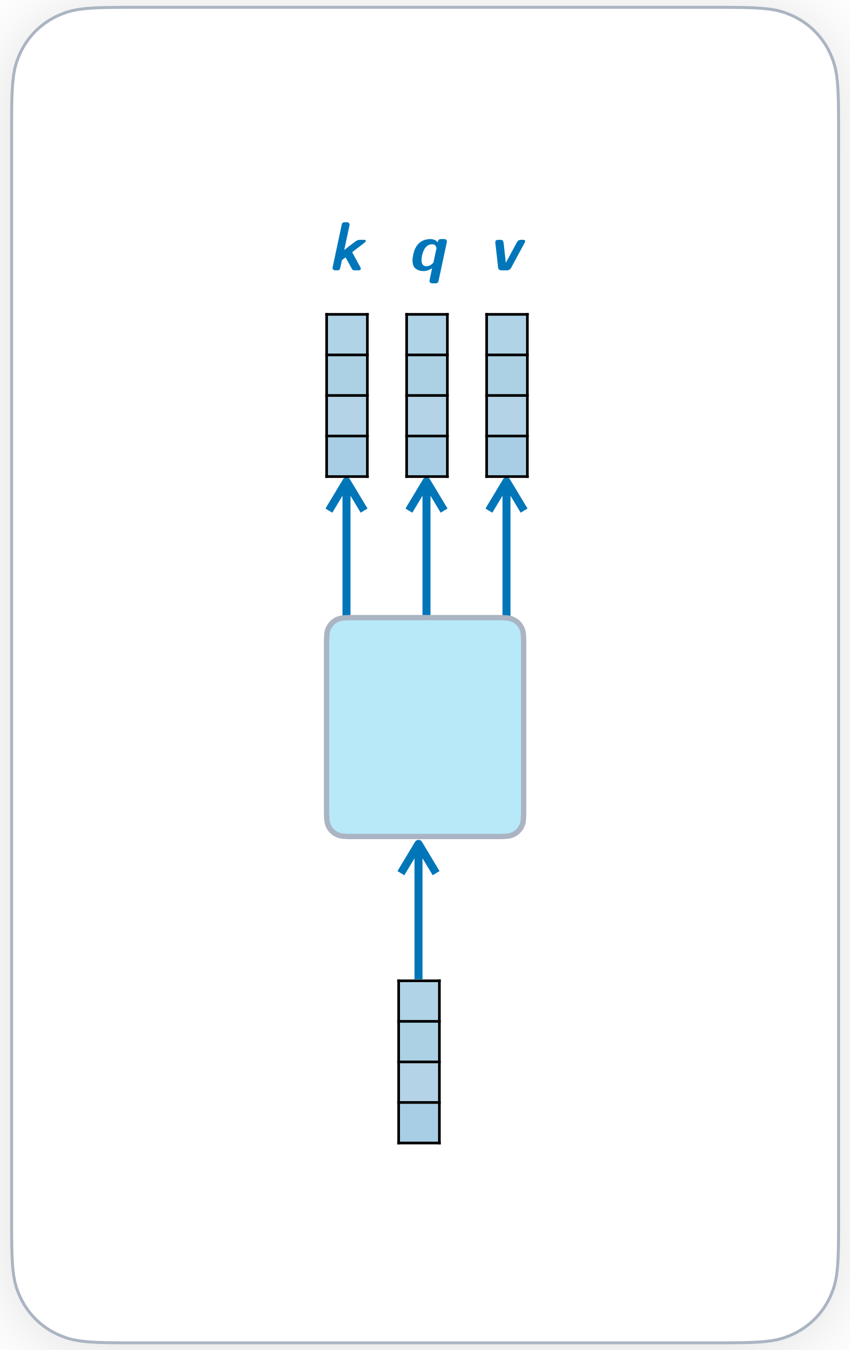
come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**



Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

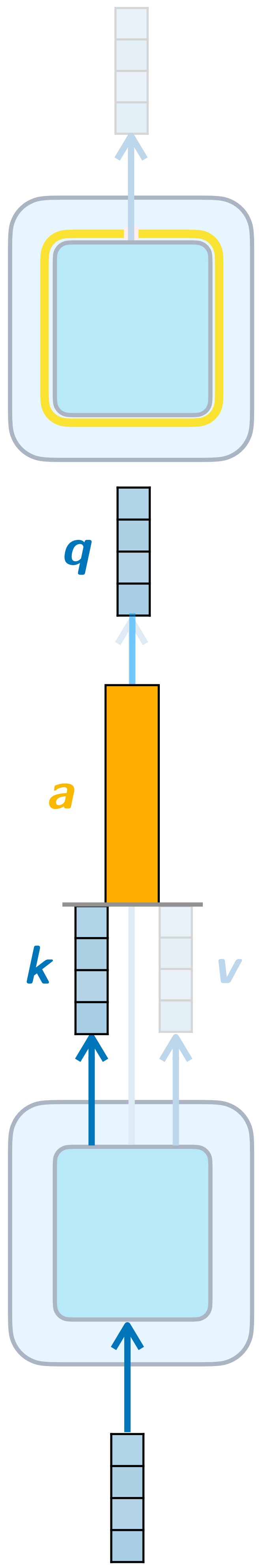


Think of ***N-to-N***

Feed **decoder** information into **decoder**

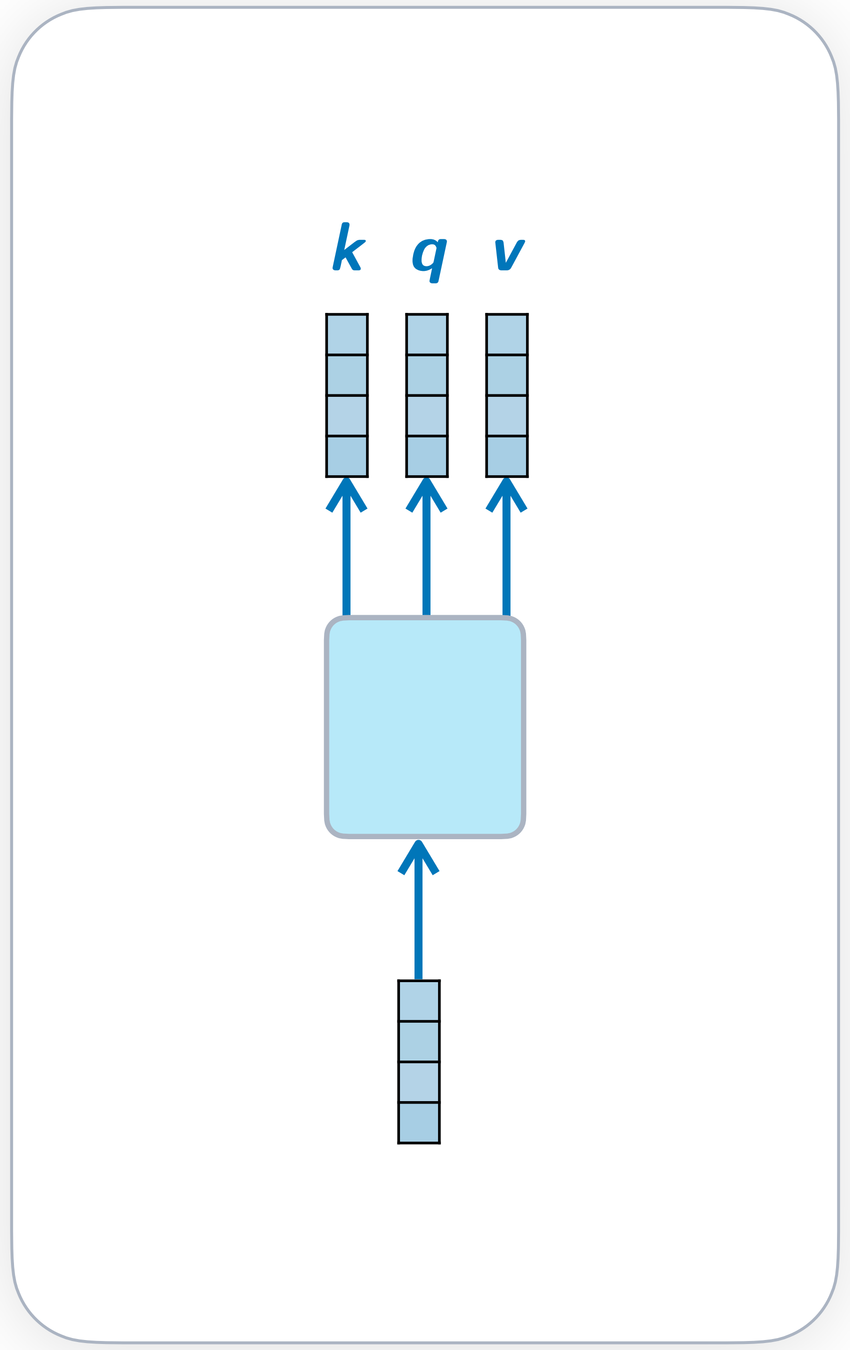
come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**



Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

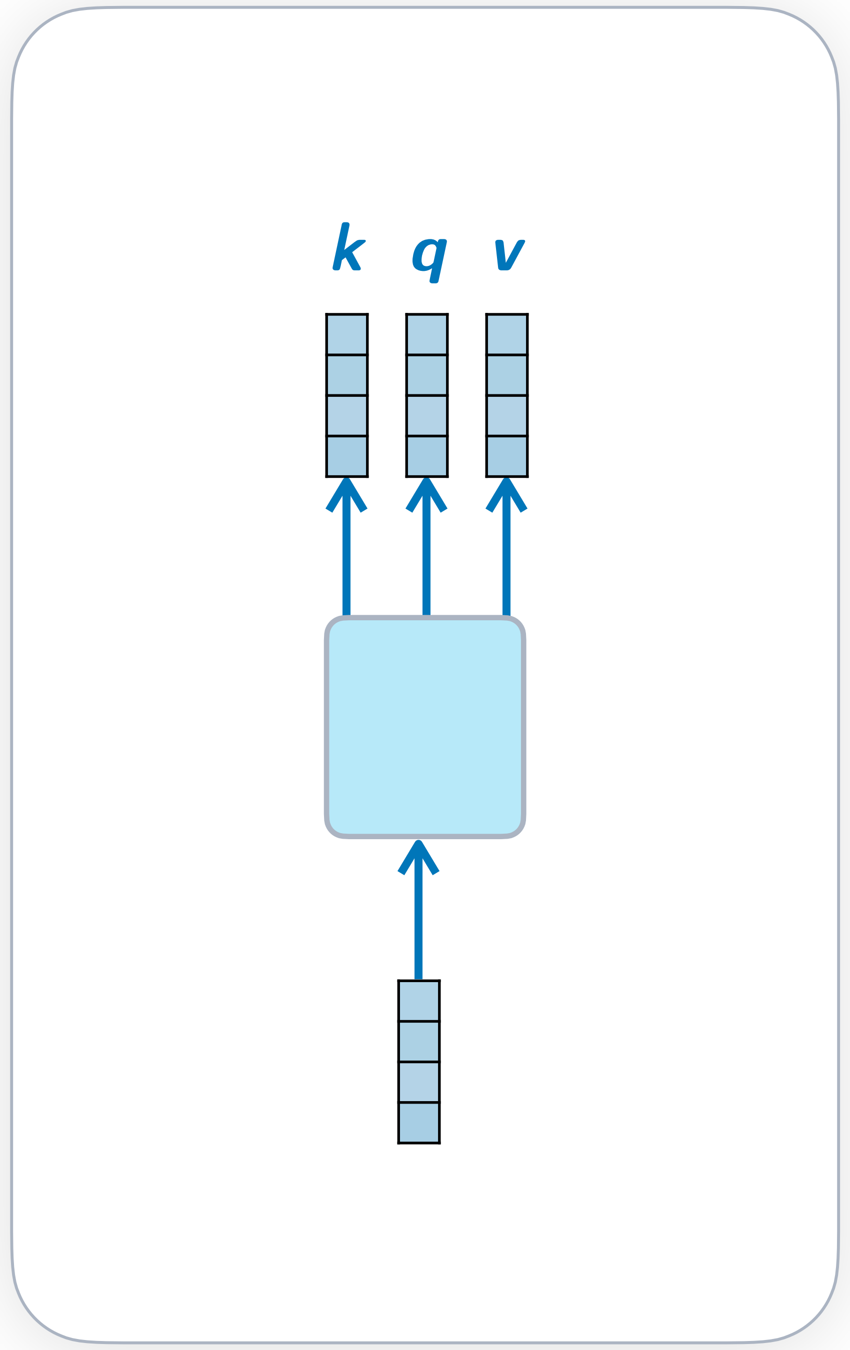


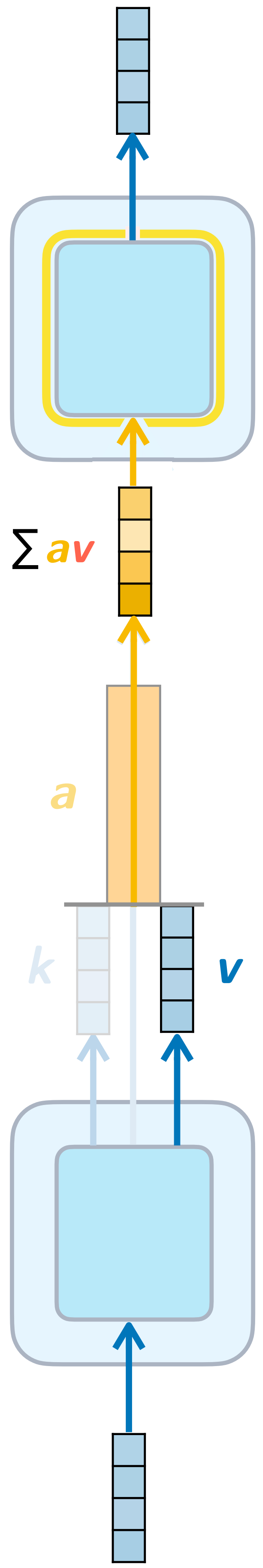
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

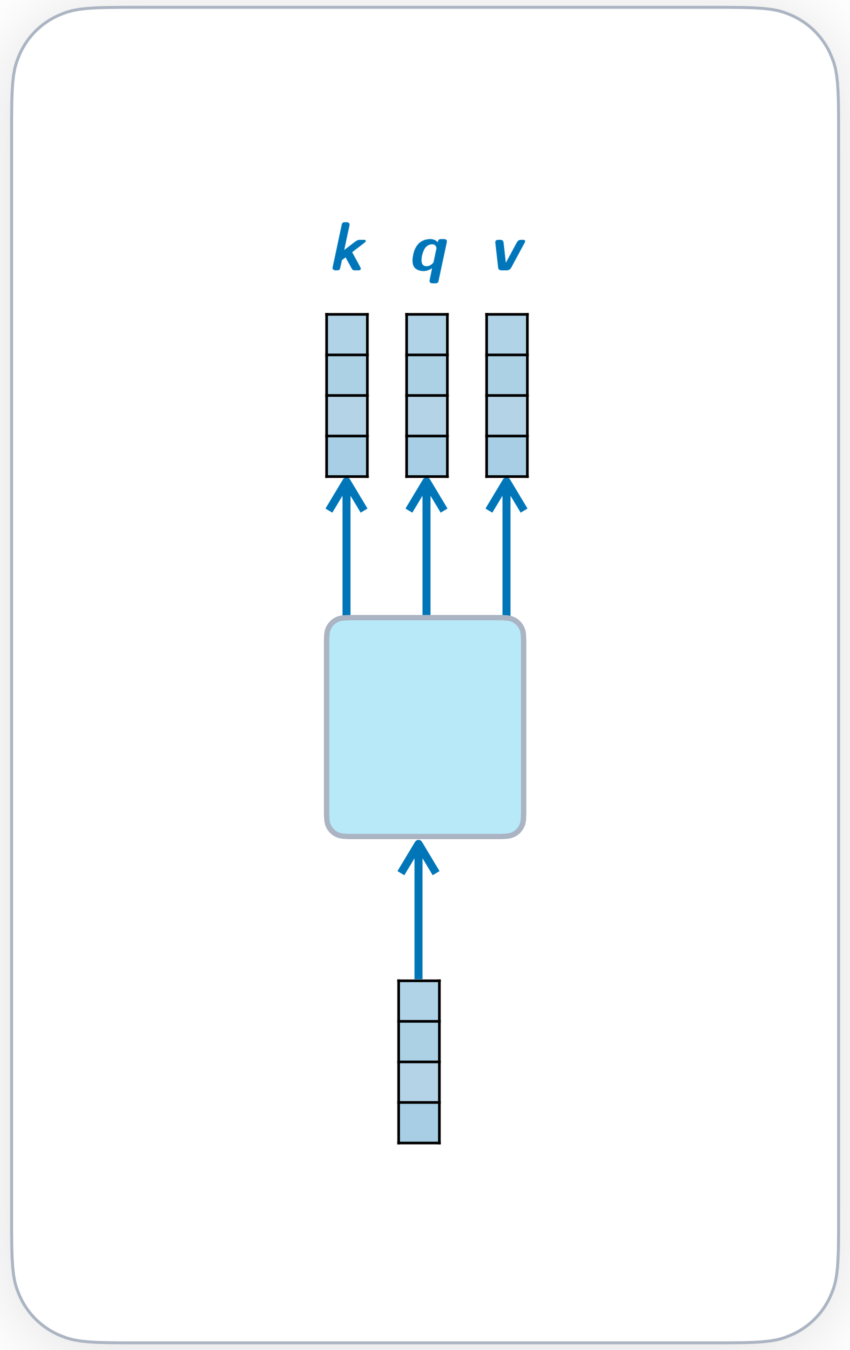
key and value vectors

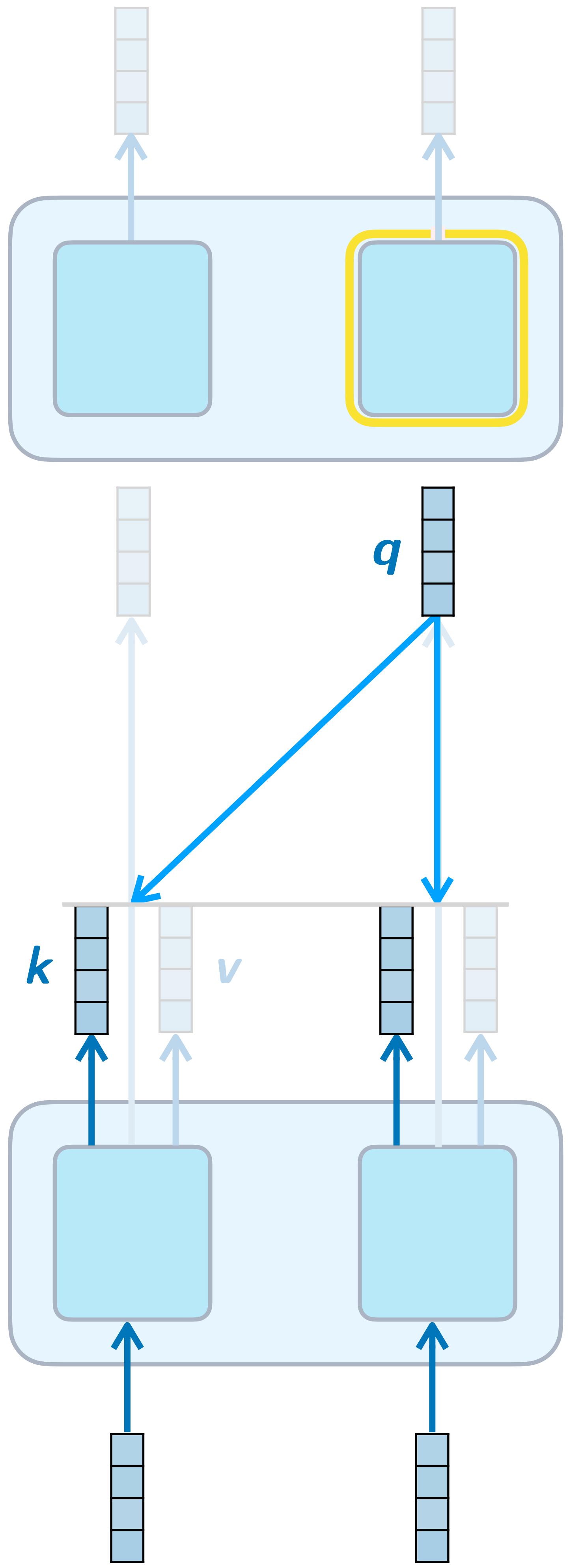
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

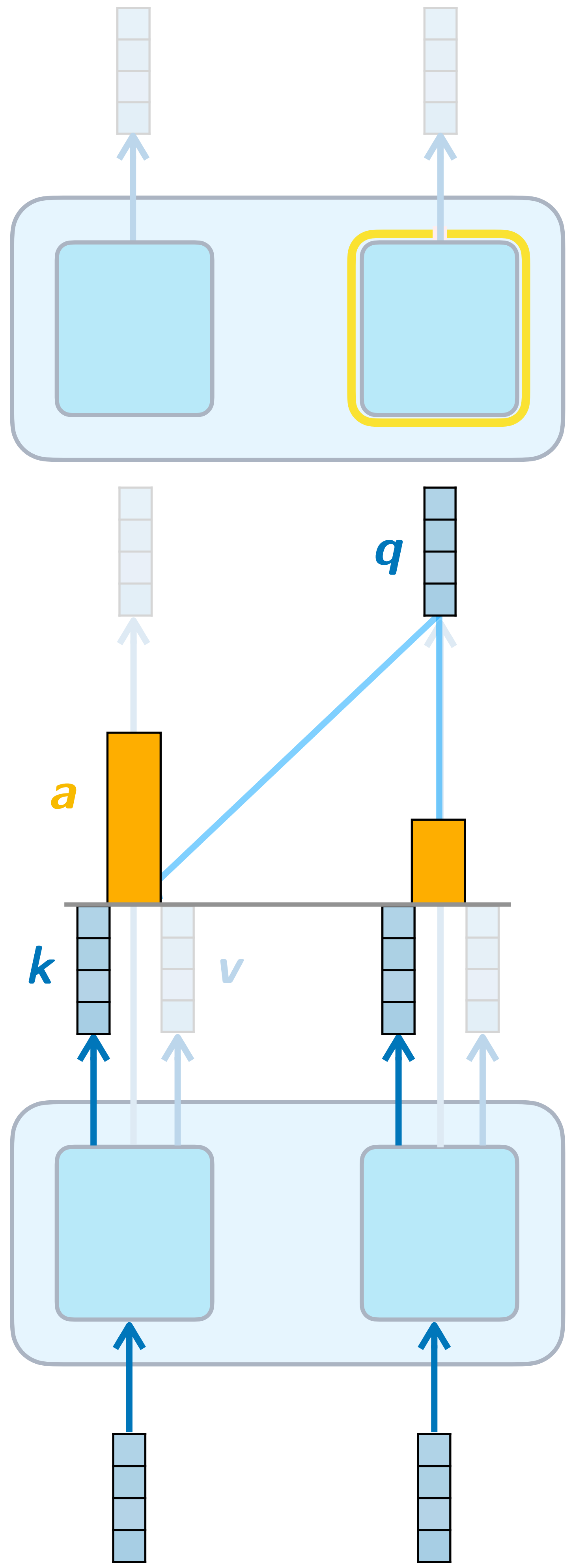
key and value vectors

Think of ***N*-*to*-*N***

Feed **decoder** information into **decoder**

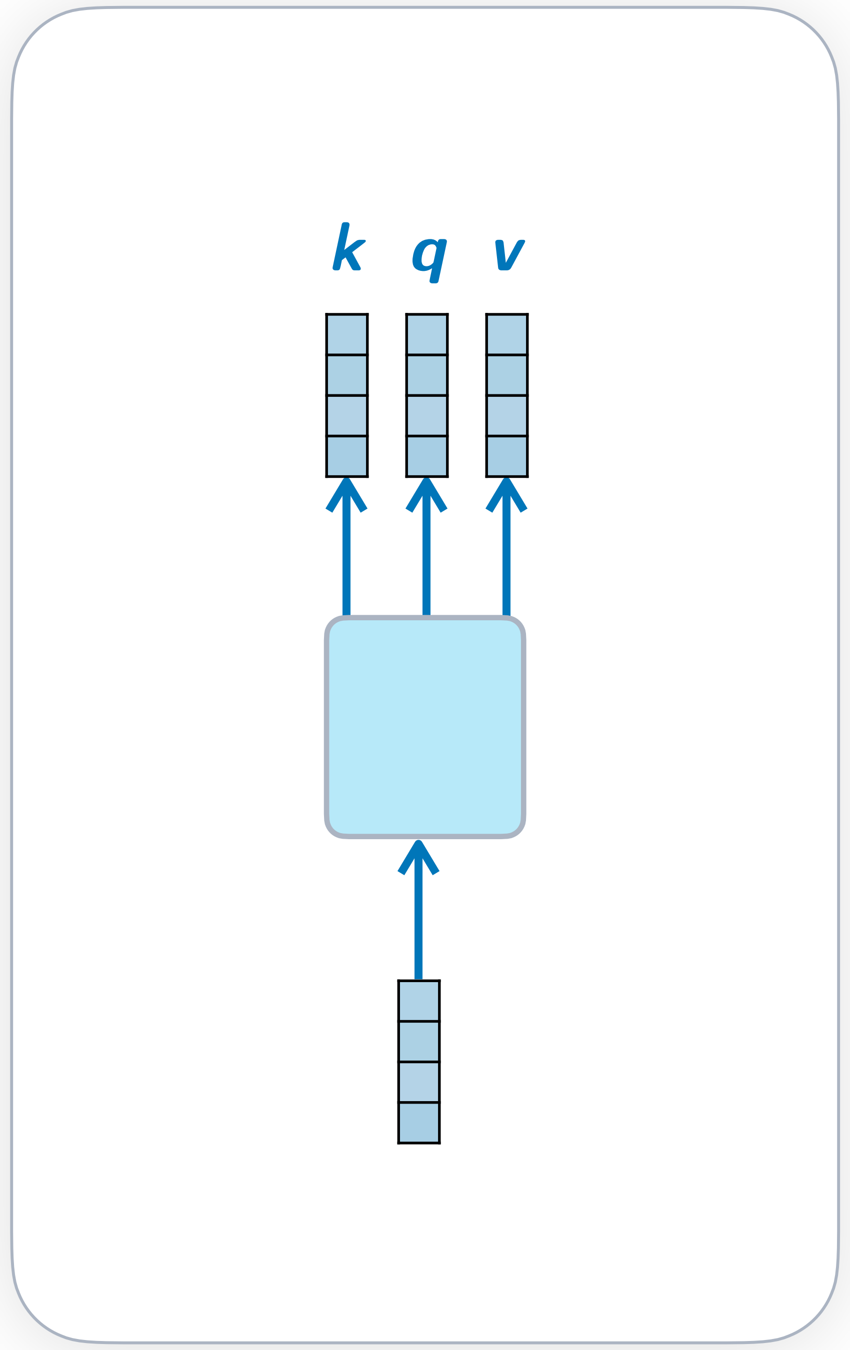
come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**



Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

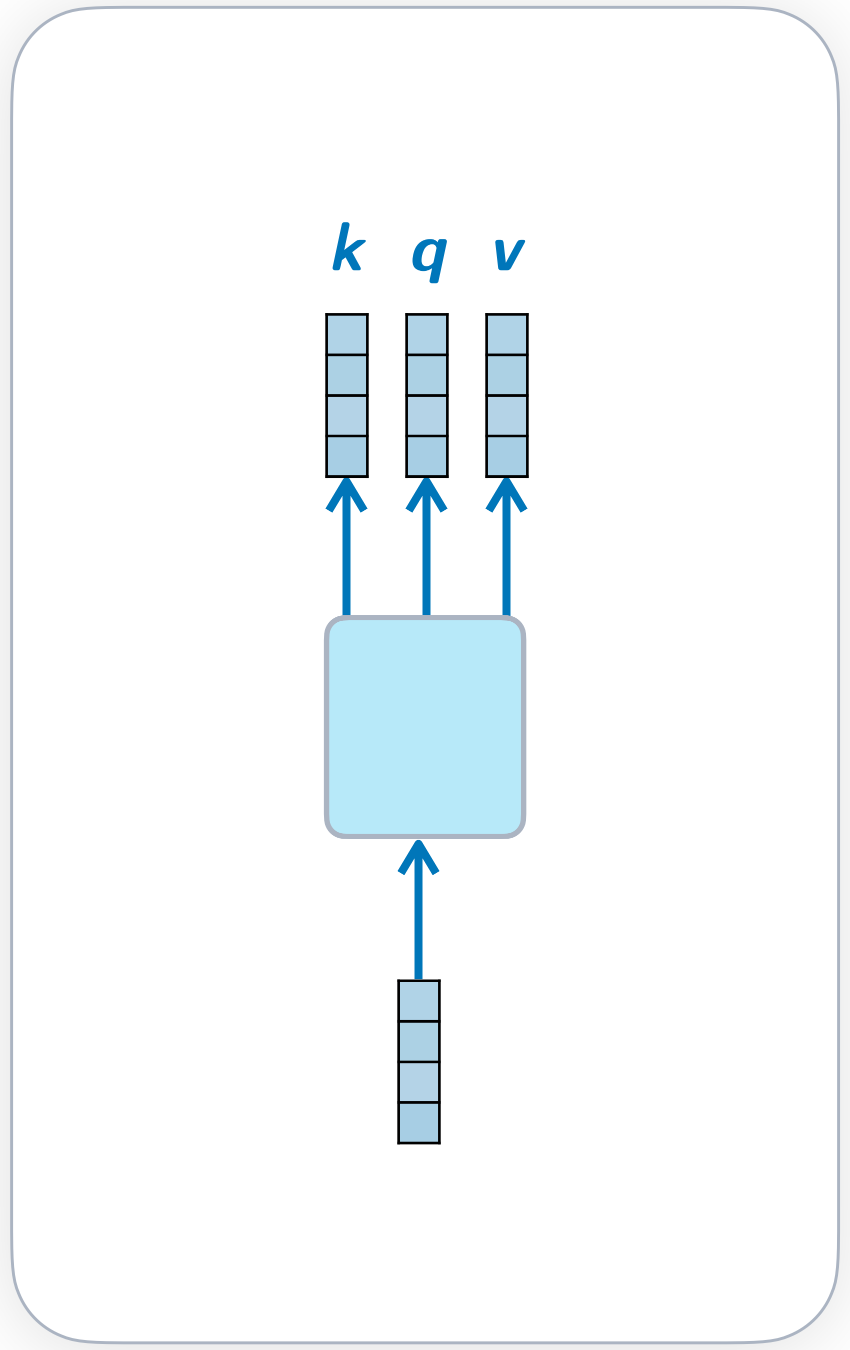


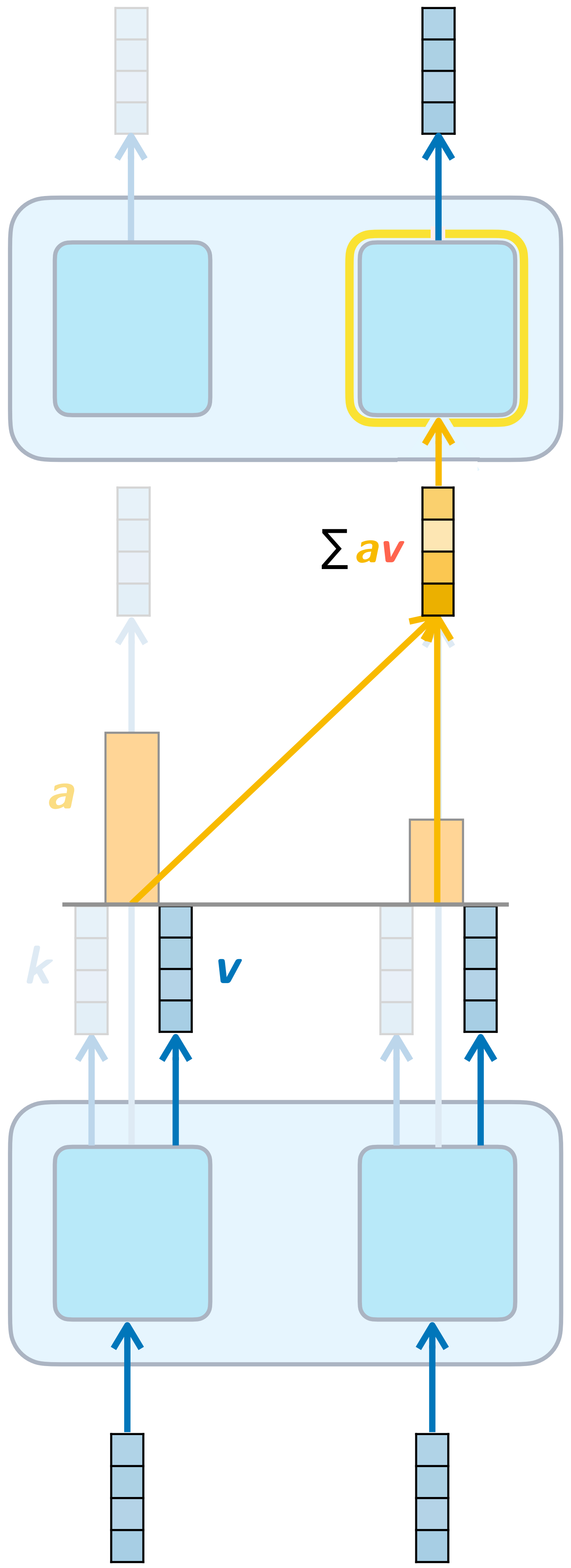
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

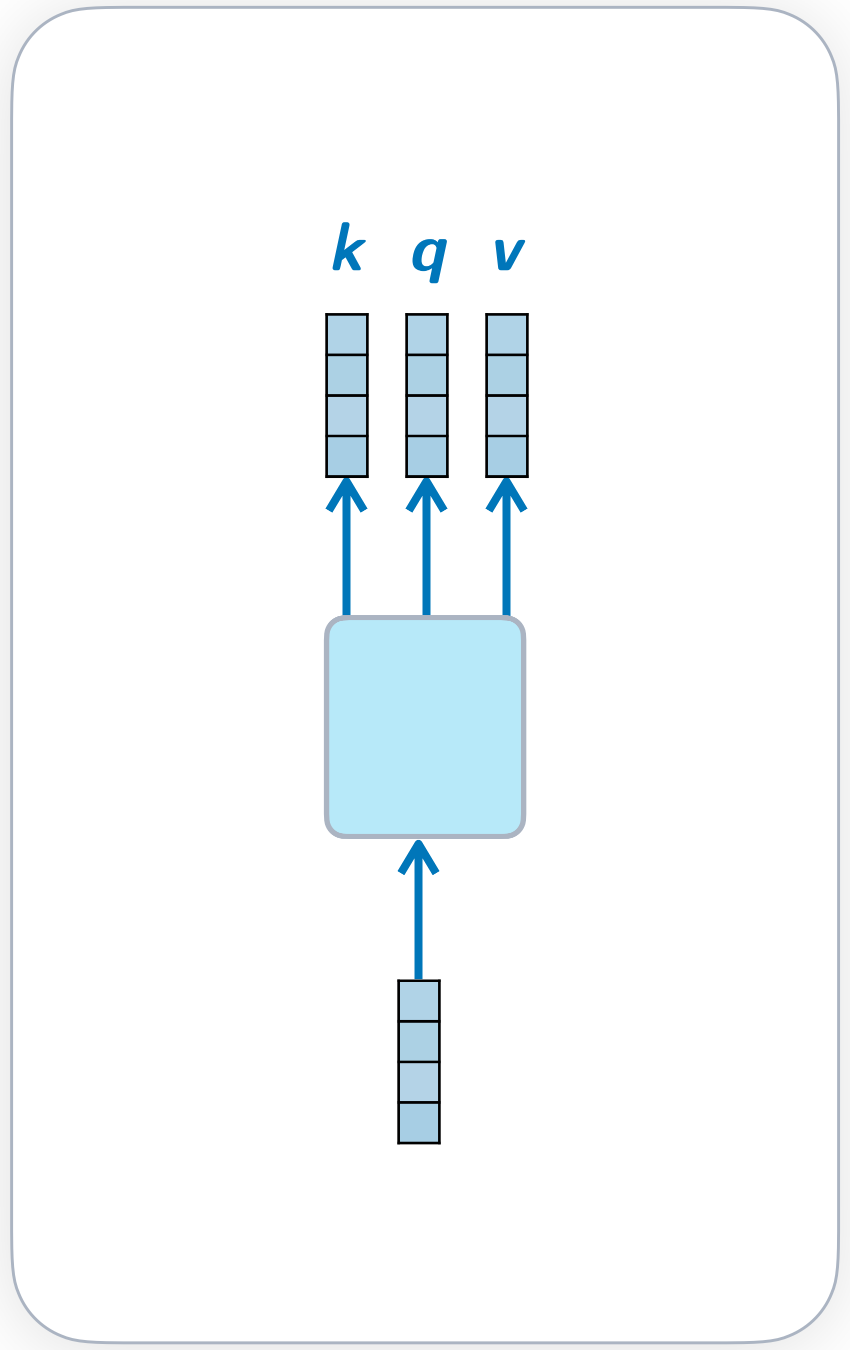
key and value vectors

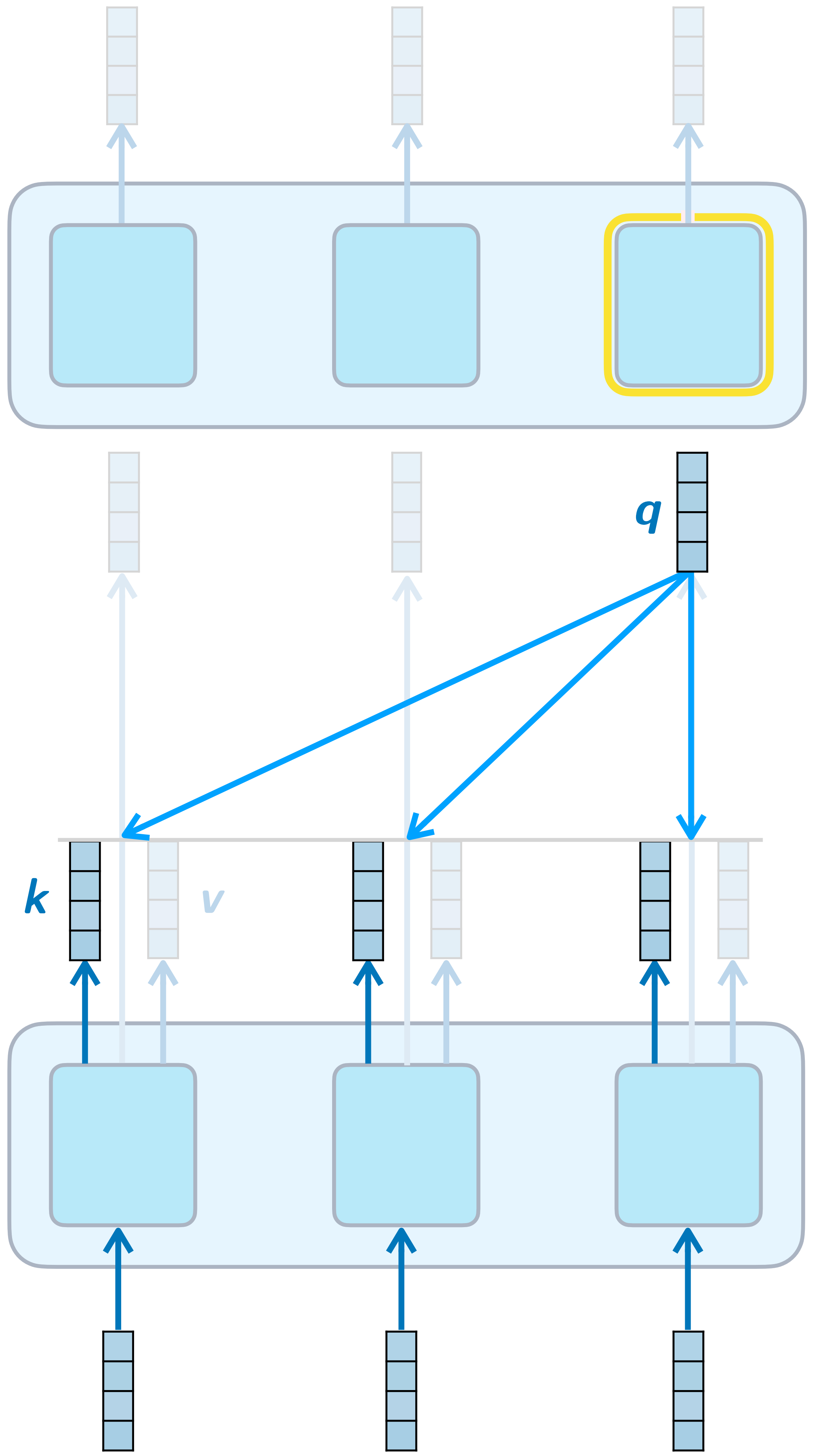
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

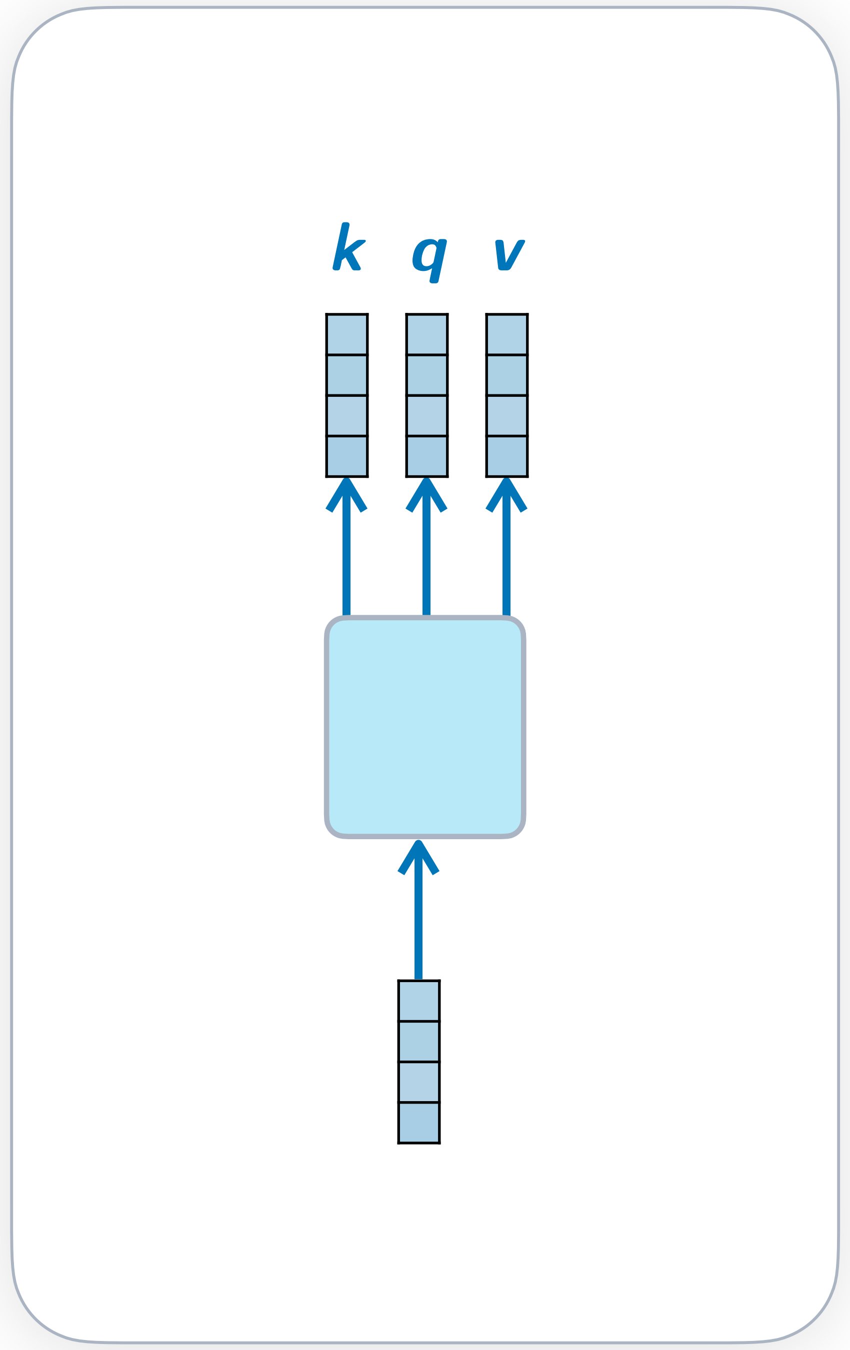
Causal Self-Attention

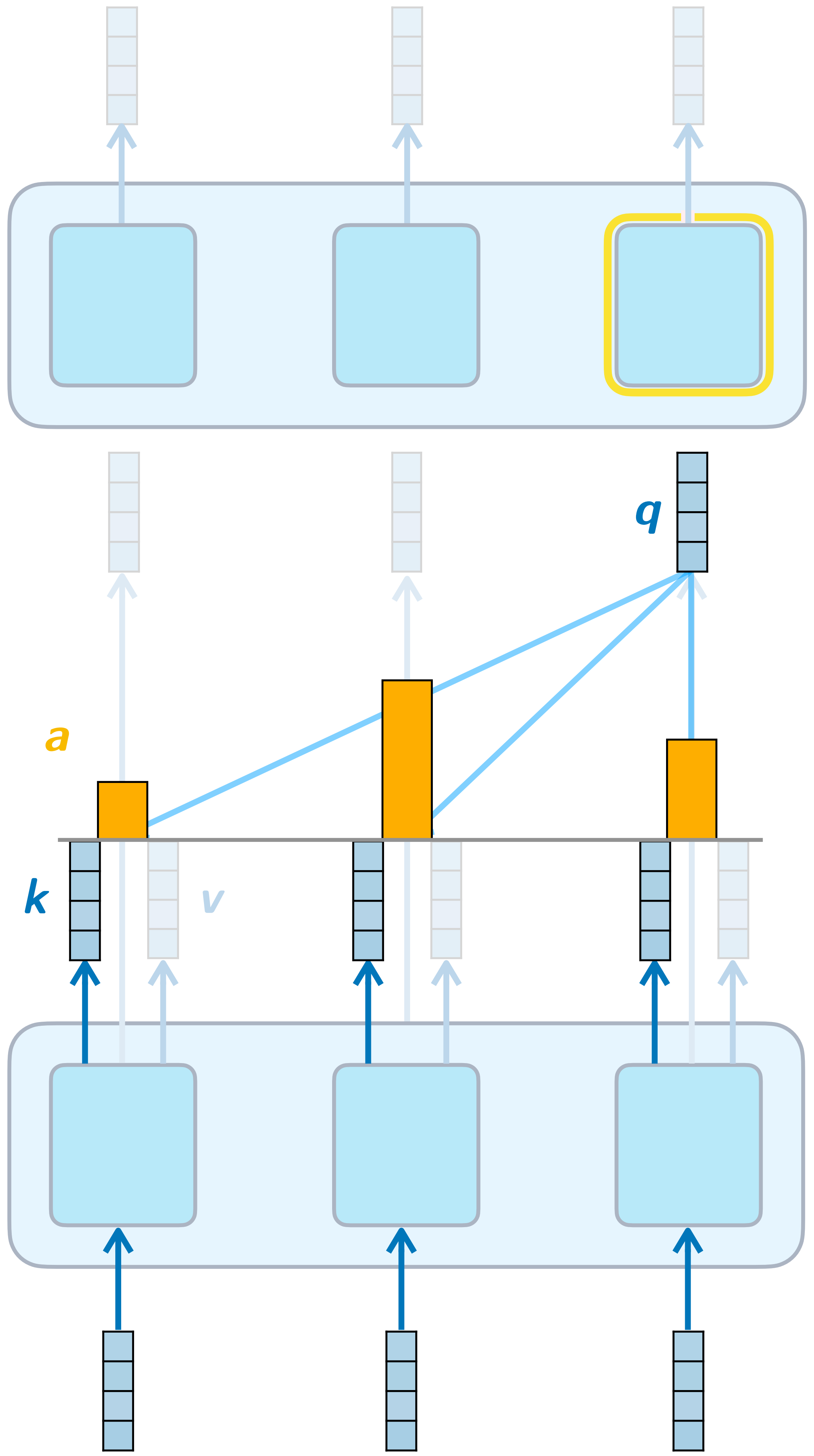
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

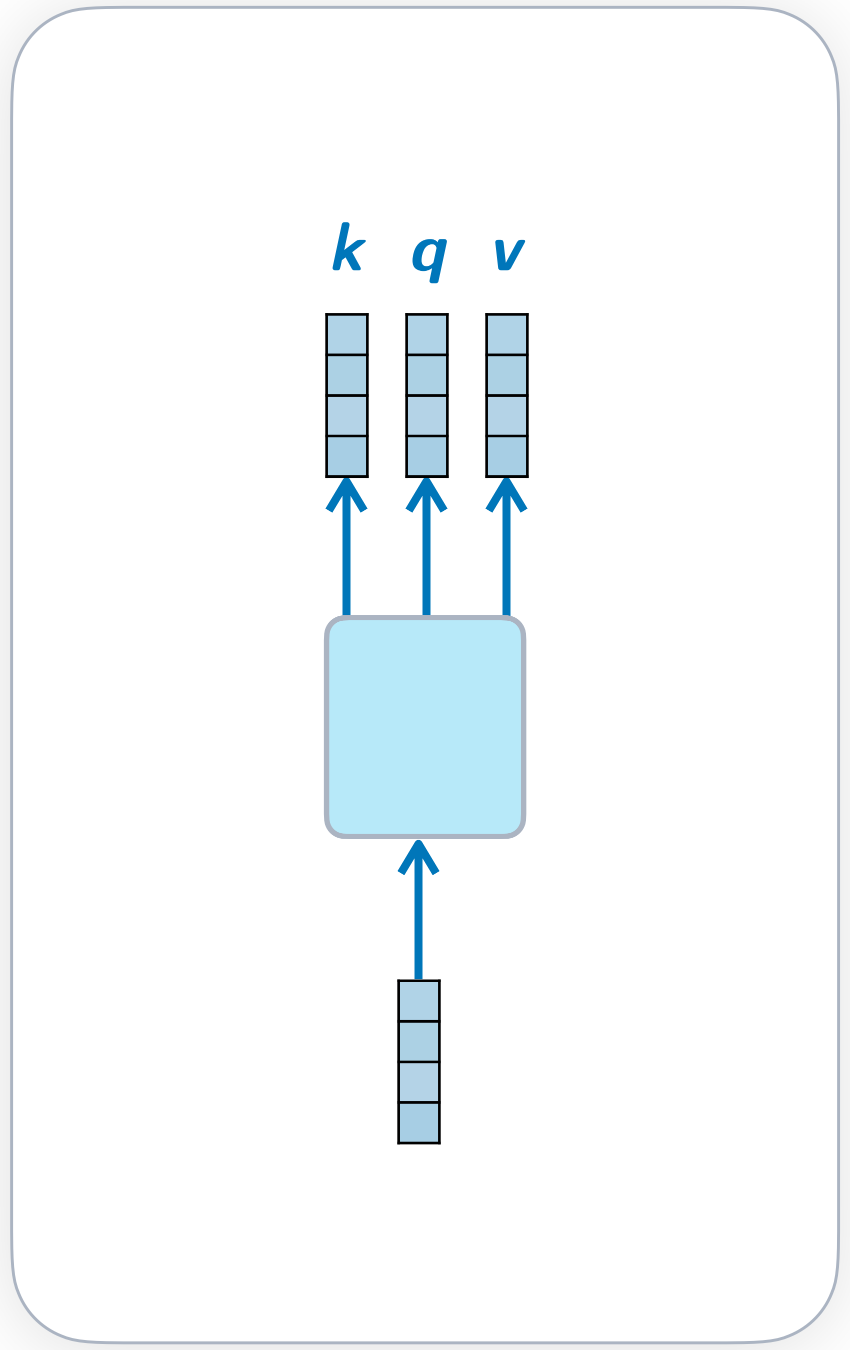
key and value vectors

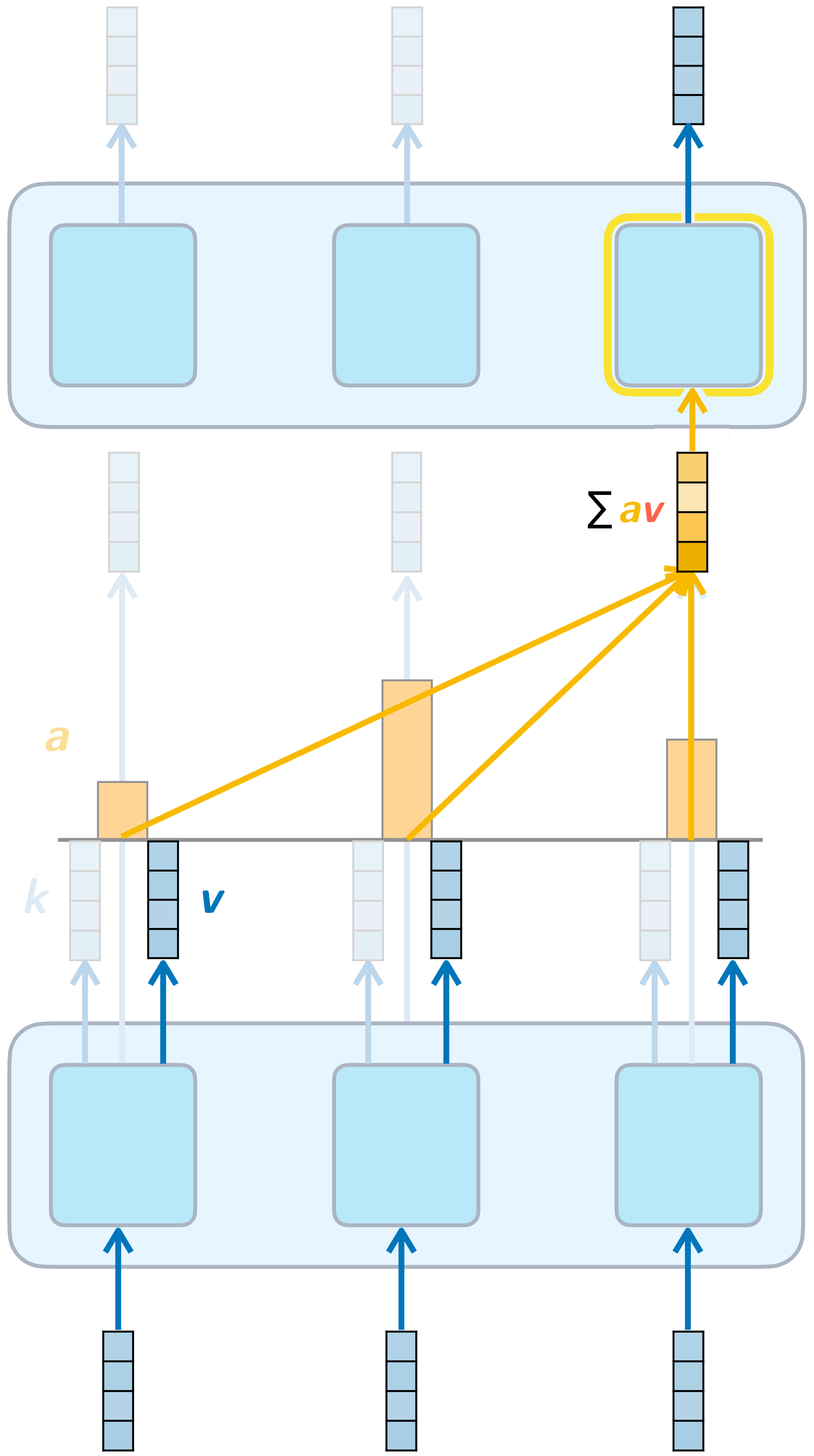
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

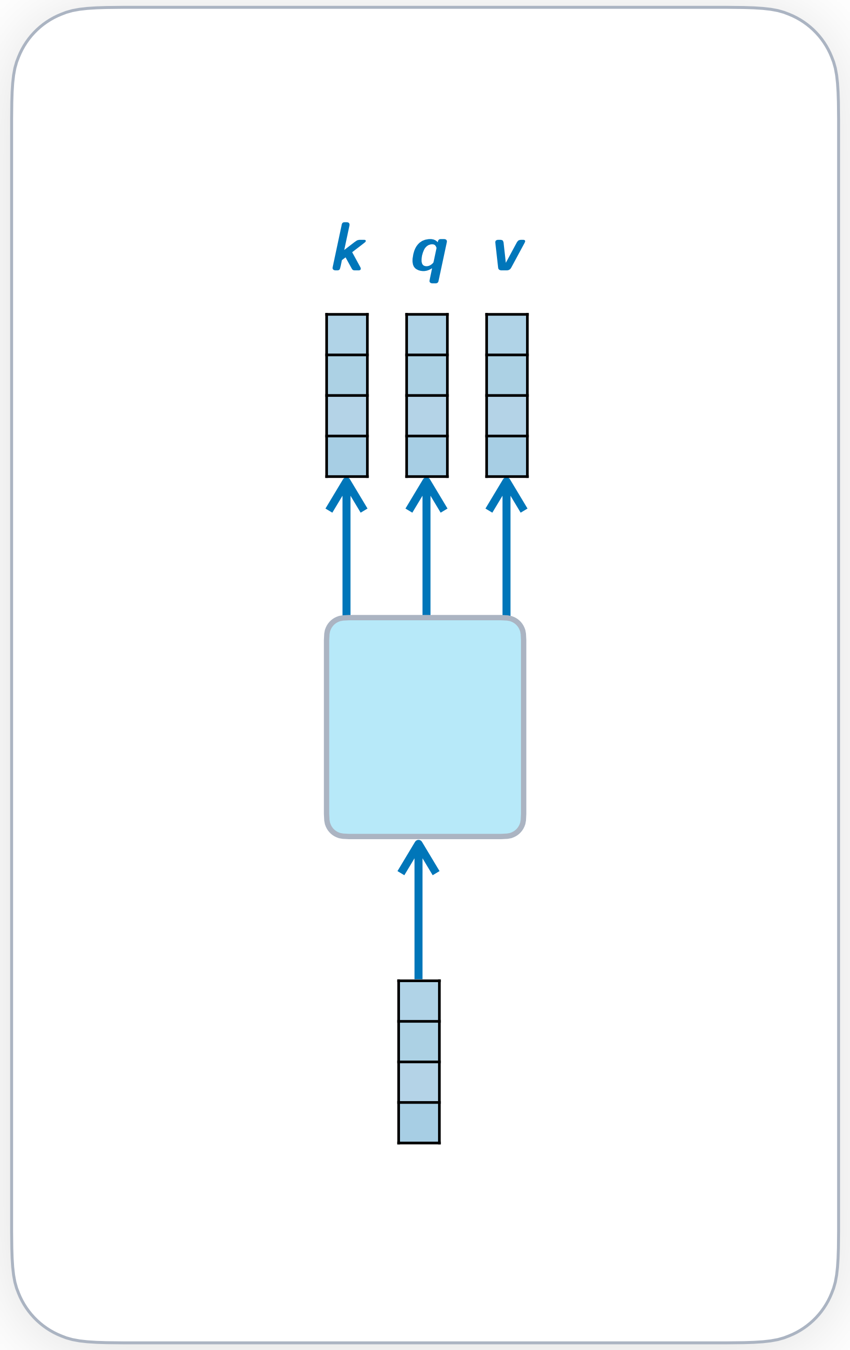
key and value vectors

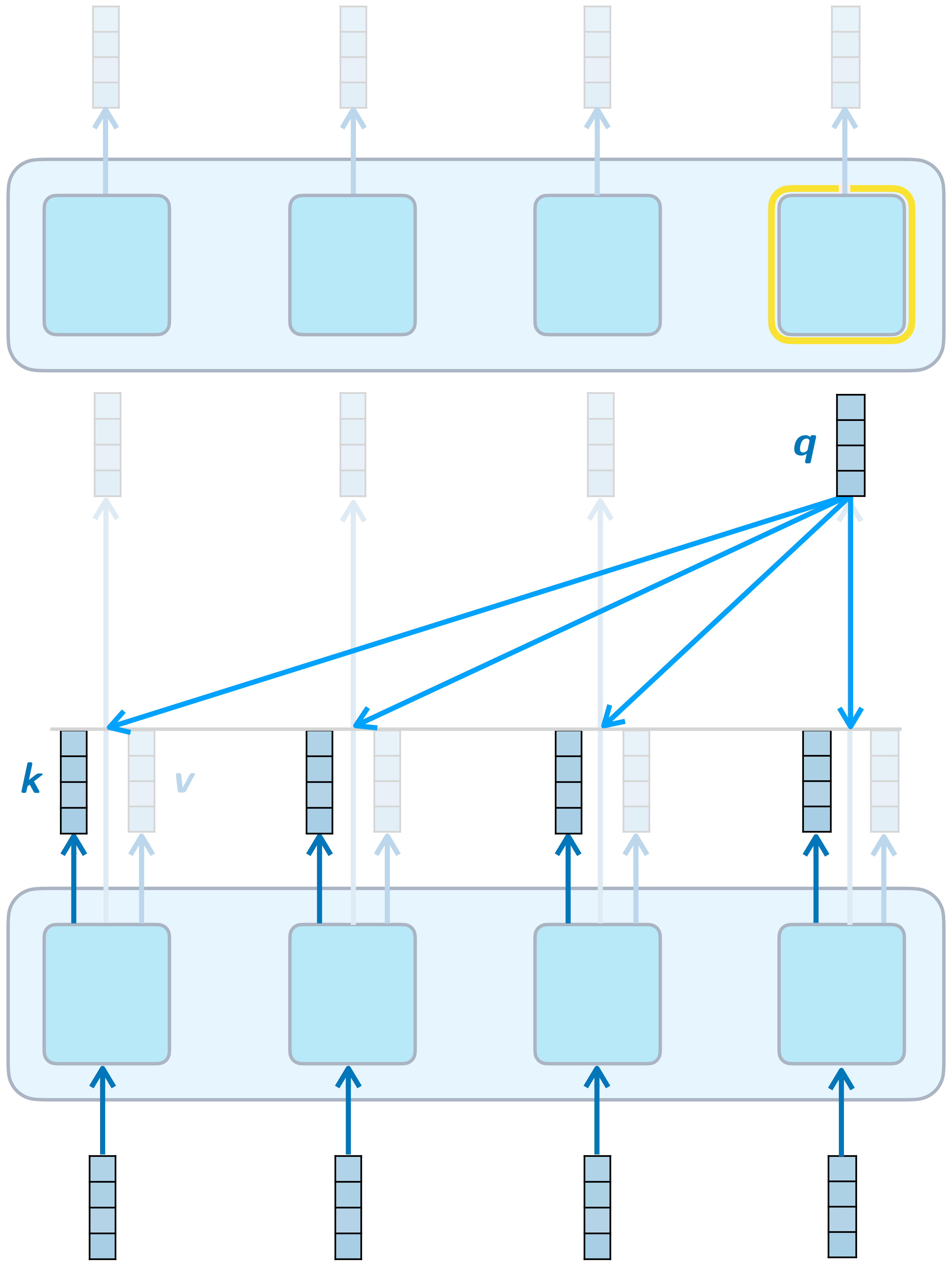
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

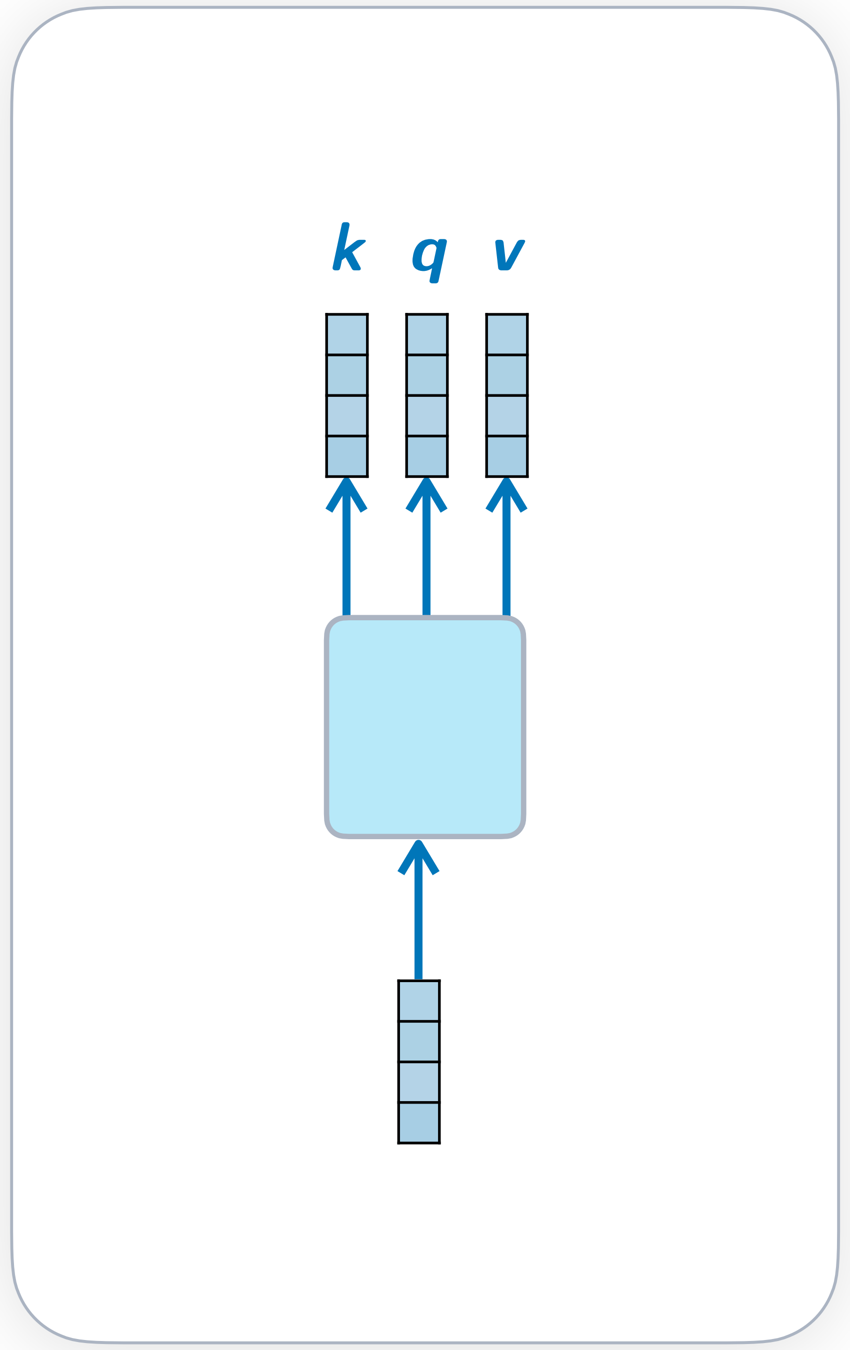
key and value vectors

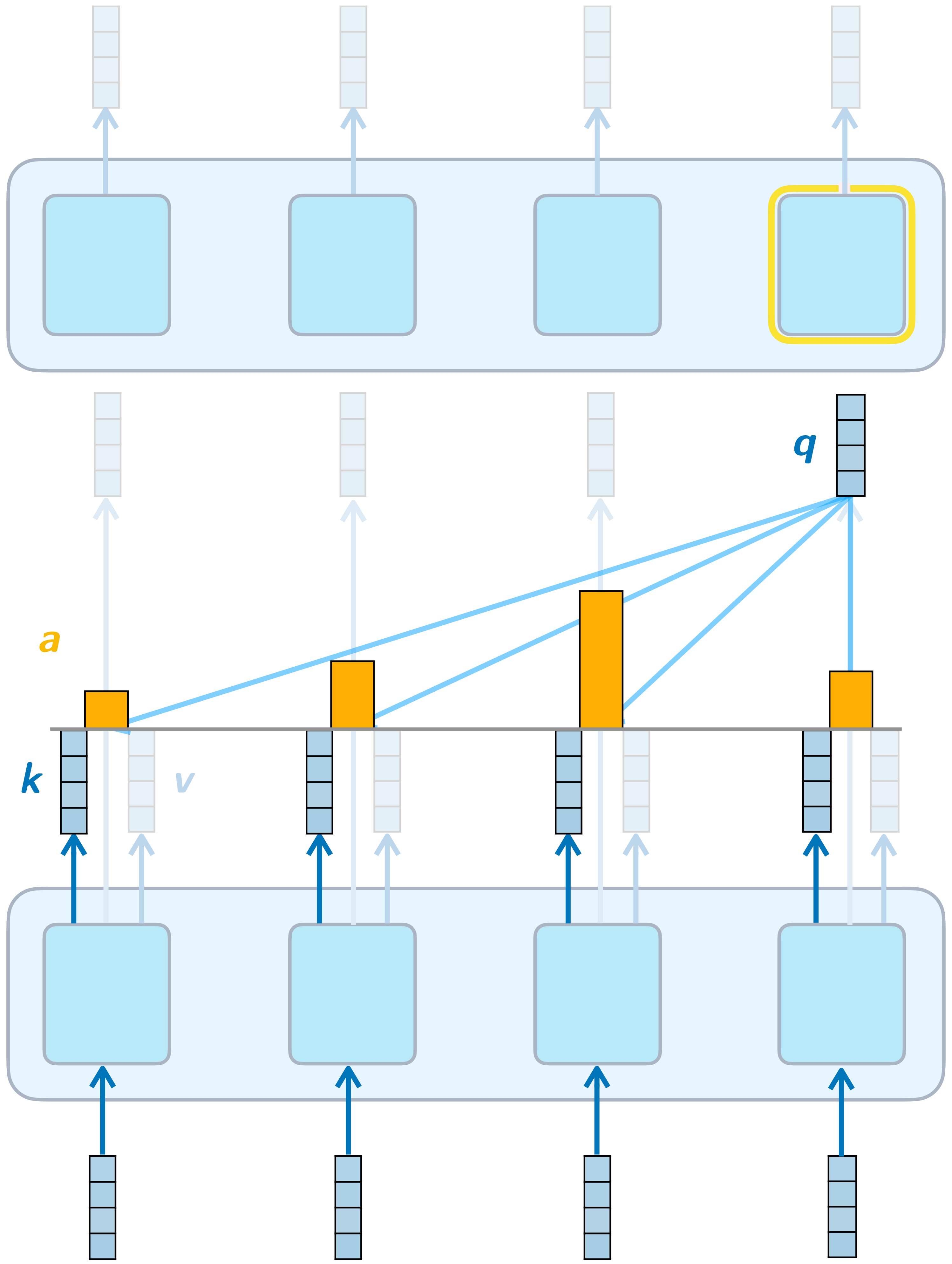
Think of ***N*-to-*N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

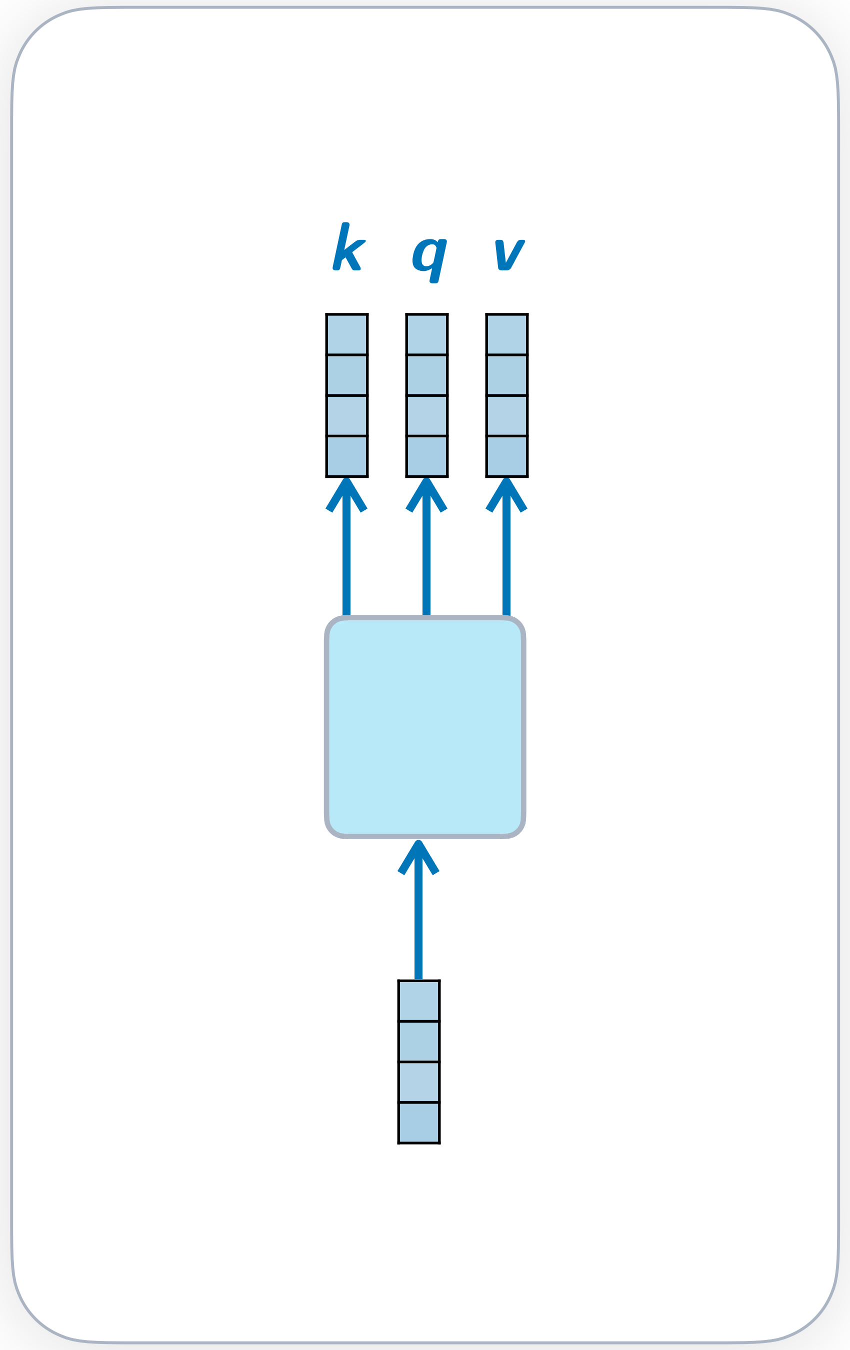
key and value vectors

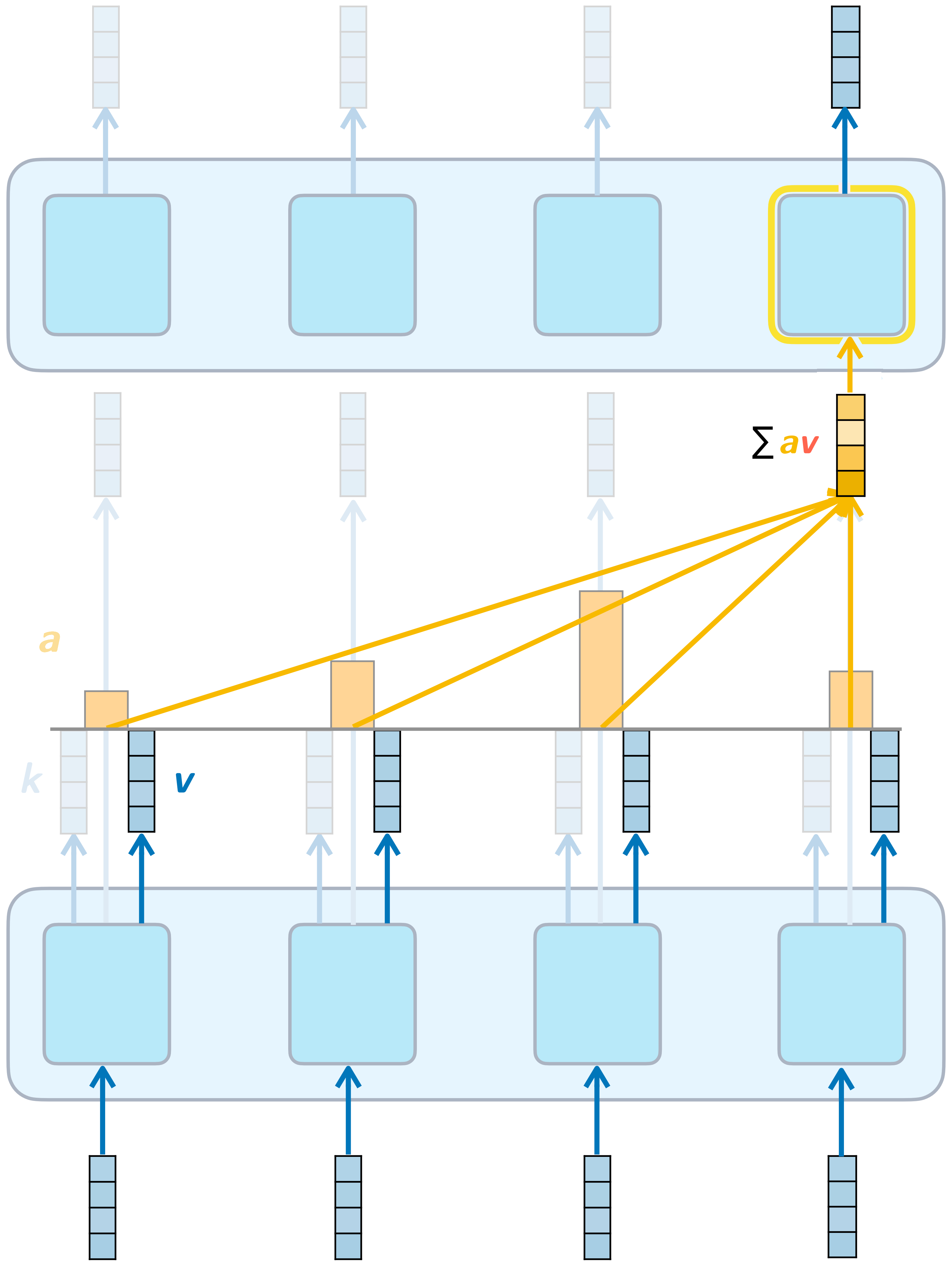
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

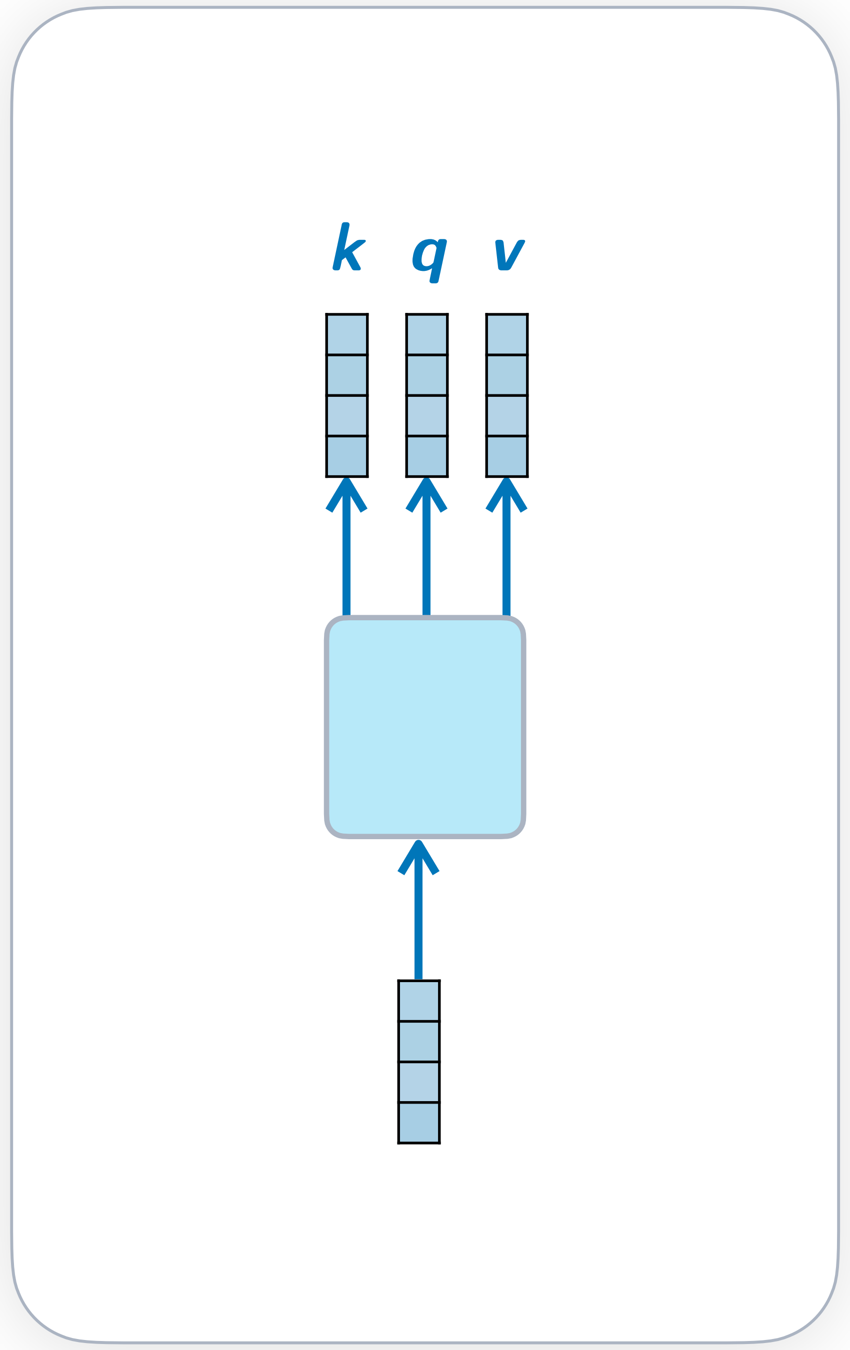
key and value vectors

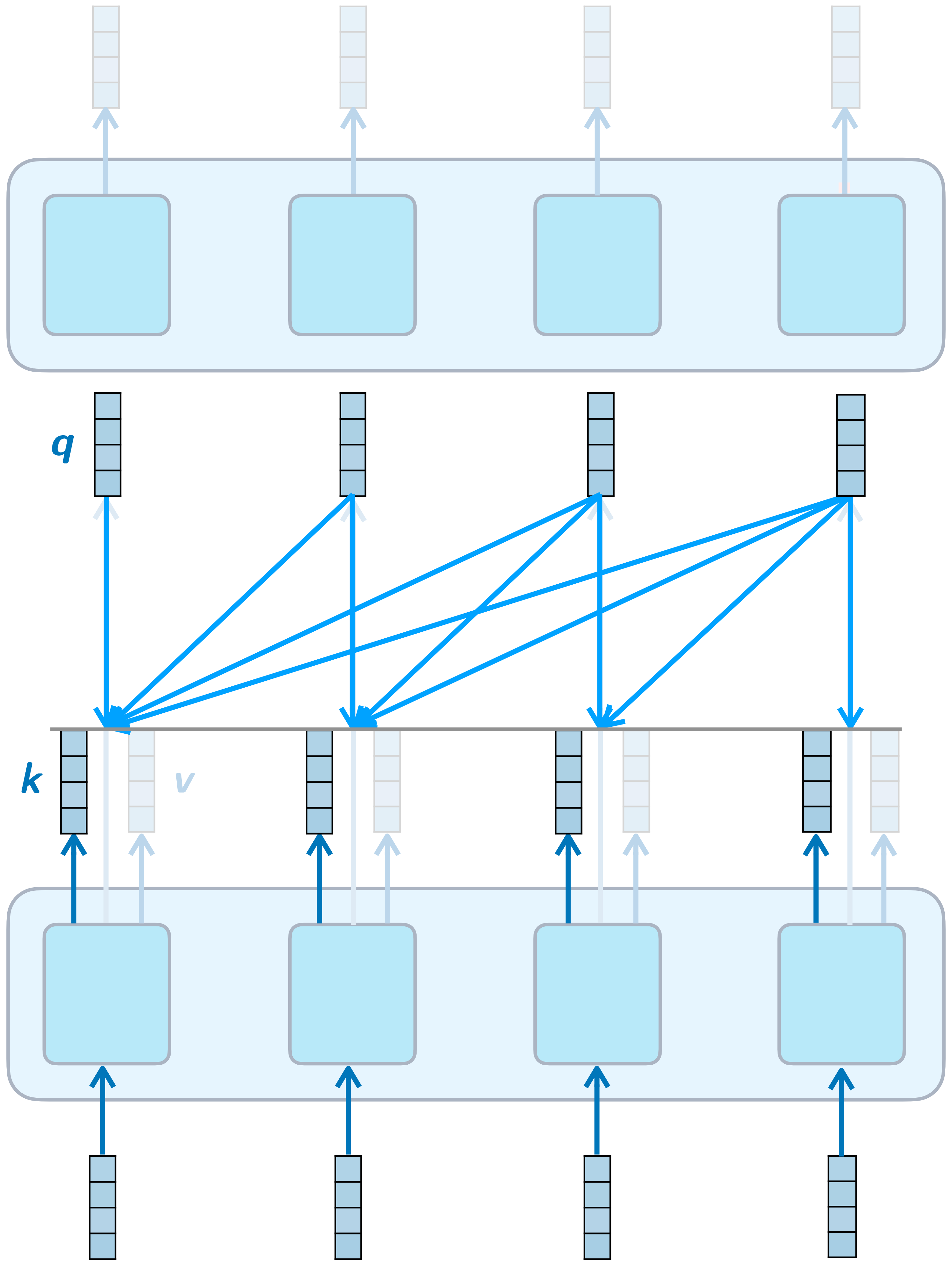
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

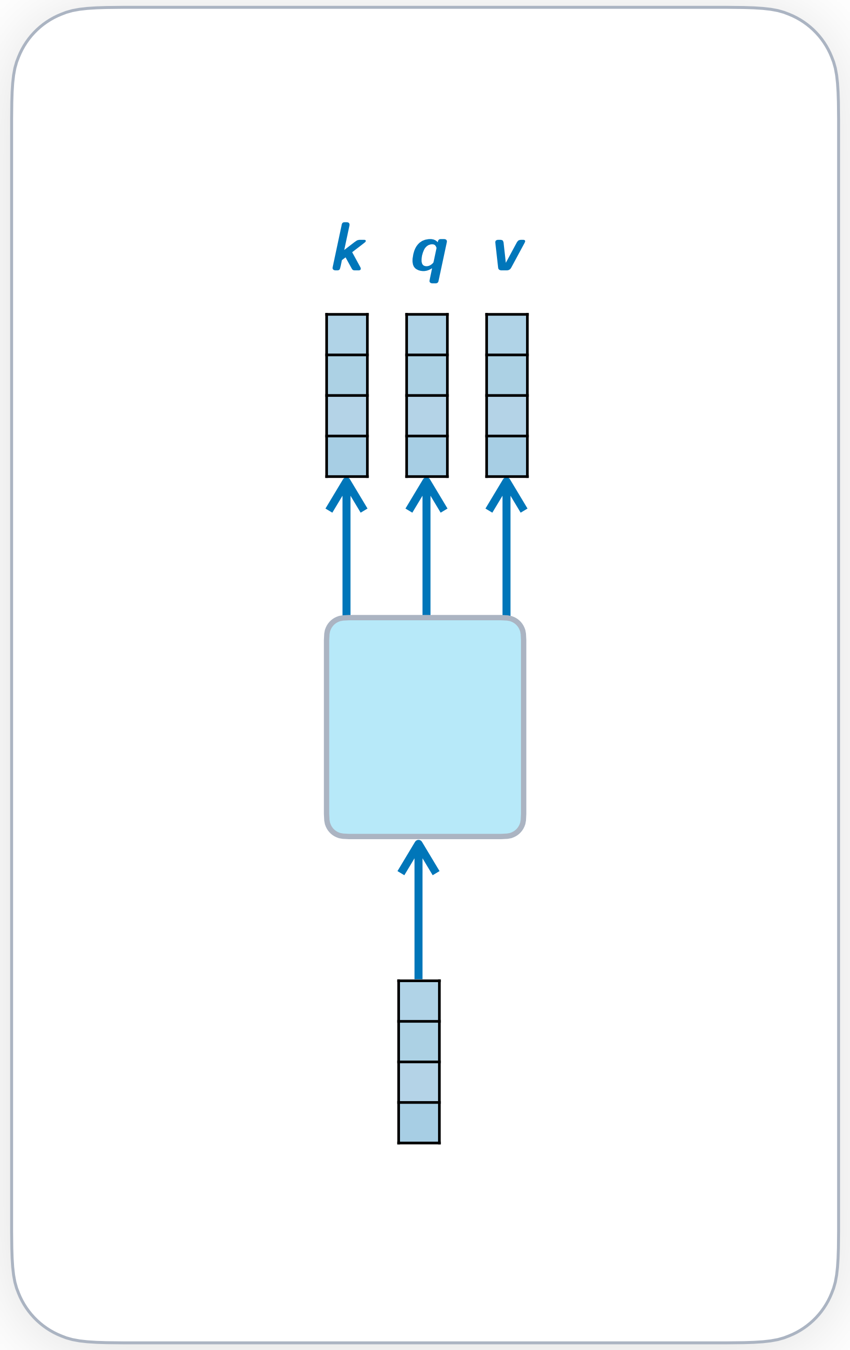
key and value vectors

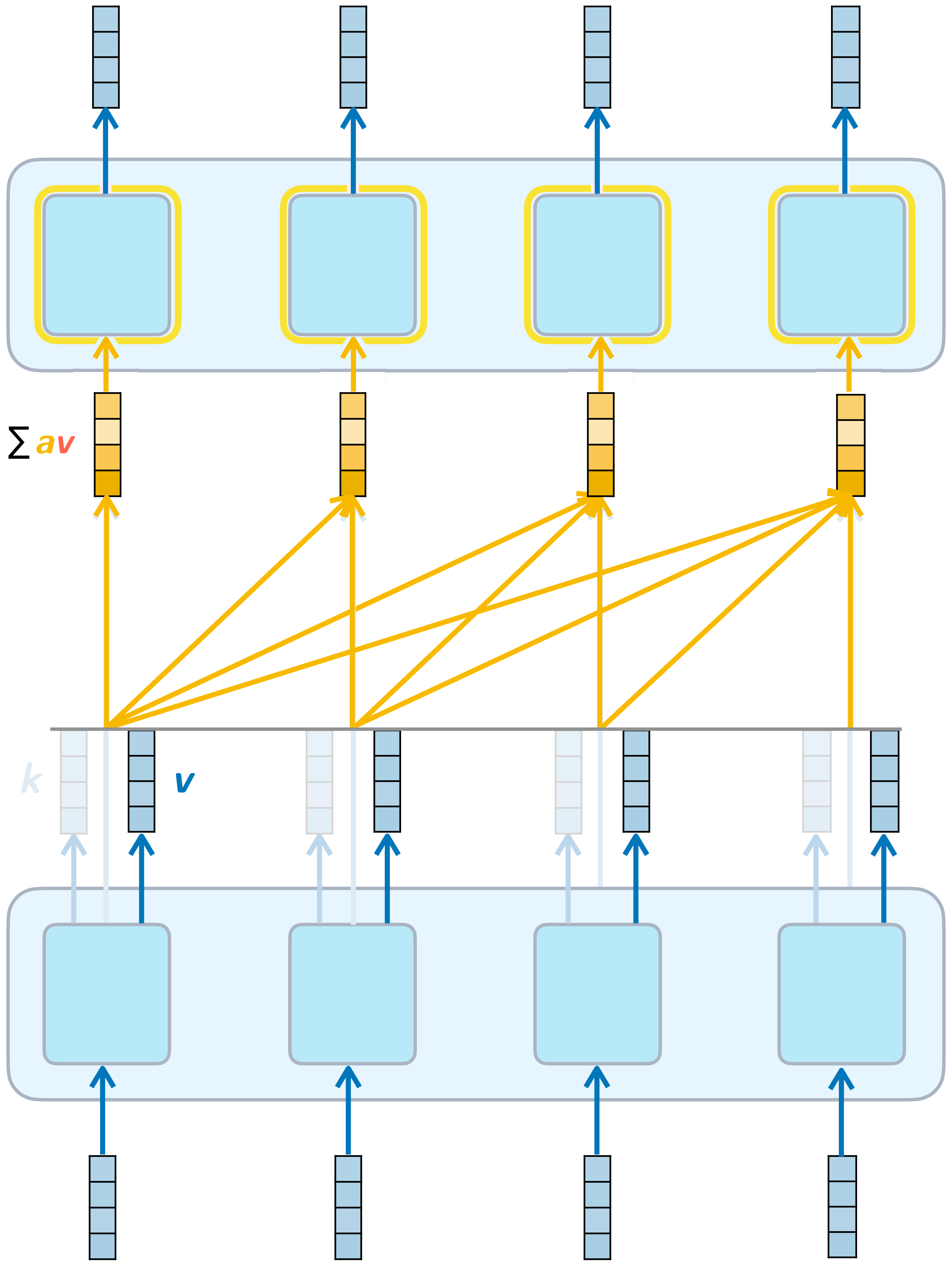
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

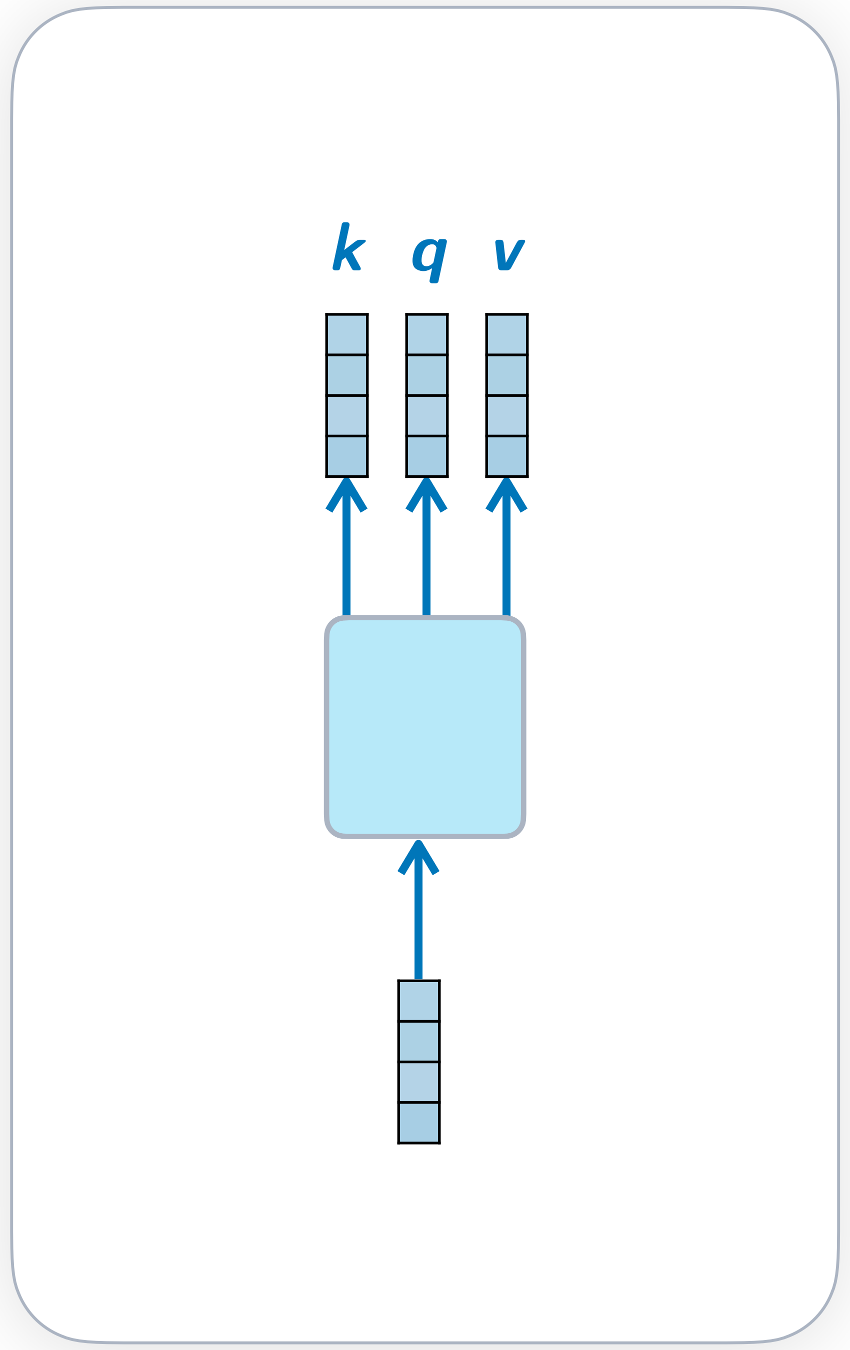
Causal Self-Attention

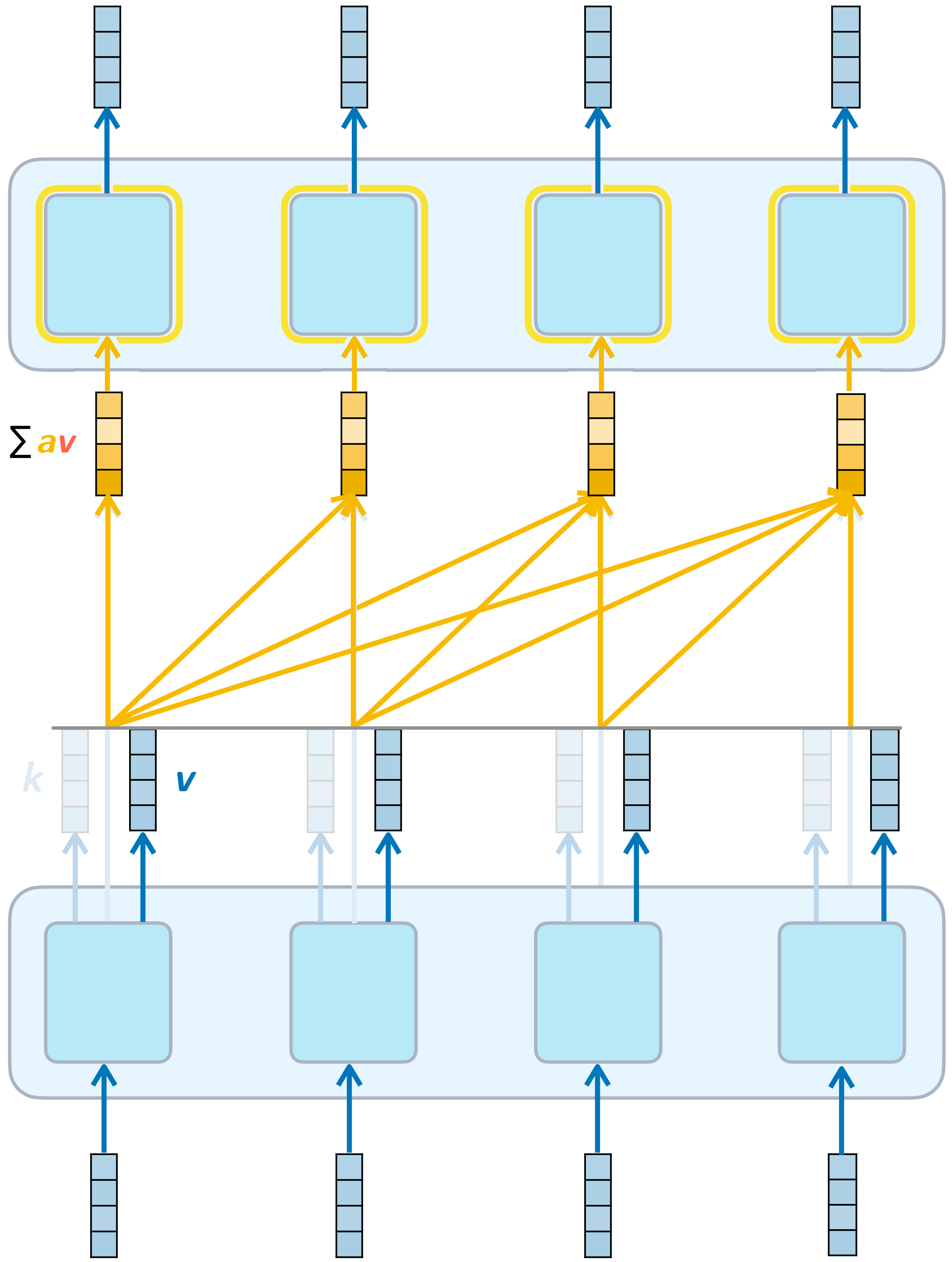
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**

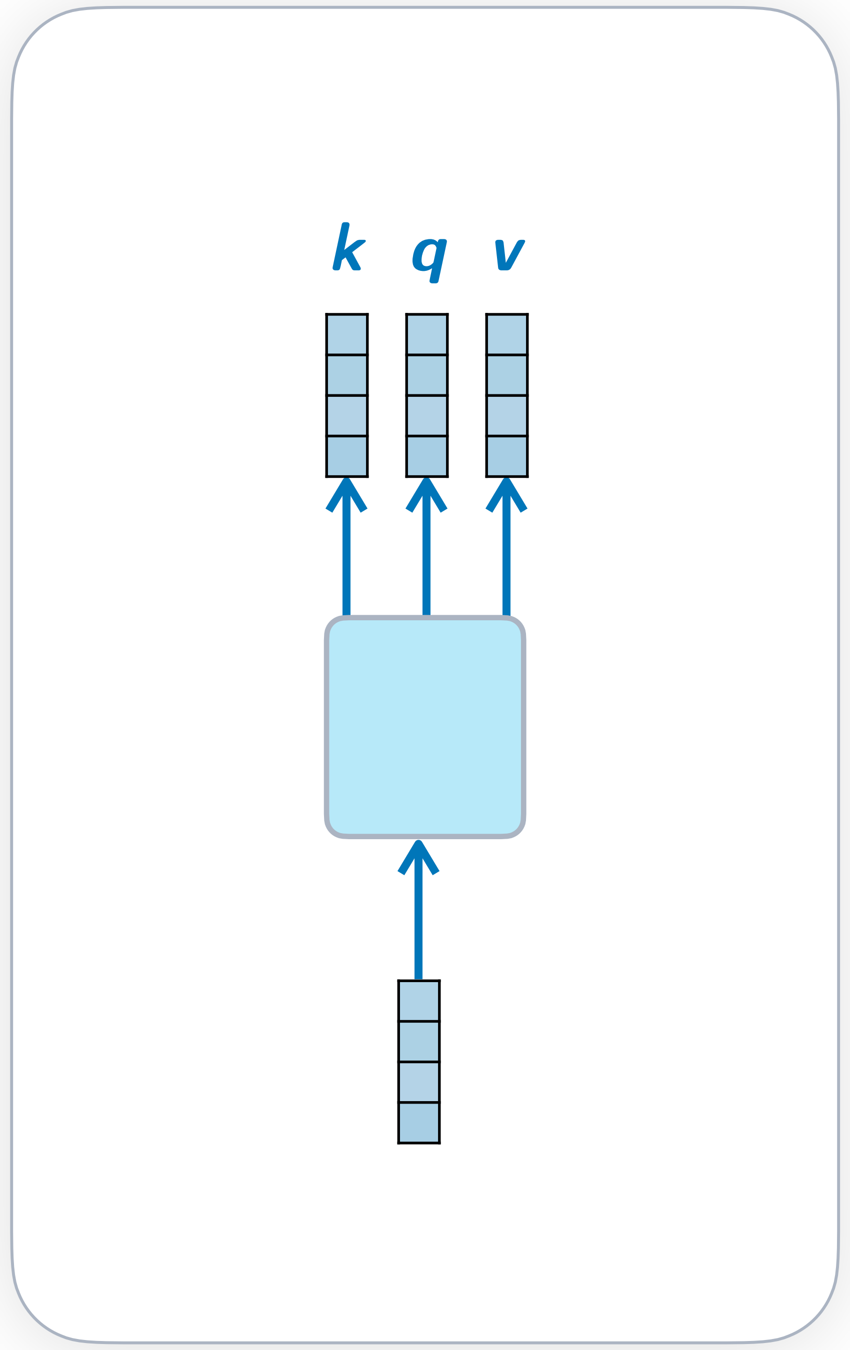


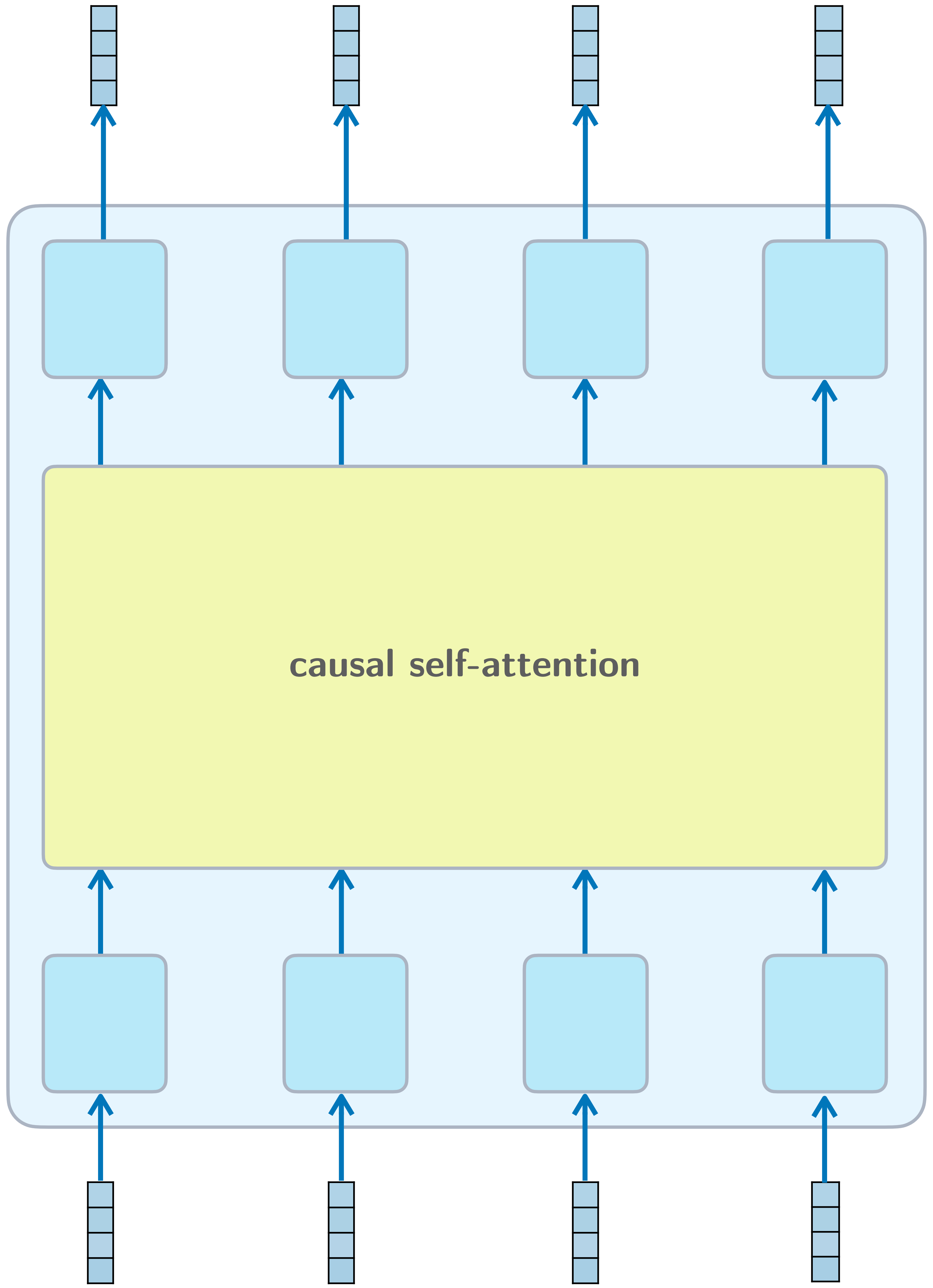


Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

Feed **decoder** information into **decoder**





refined **decoder sequence** ***X'***

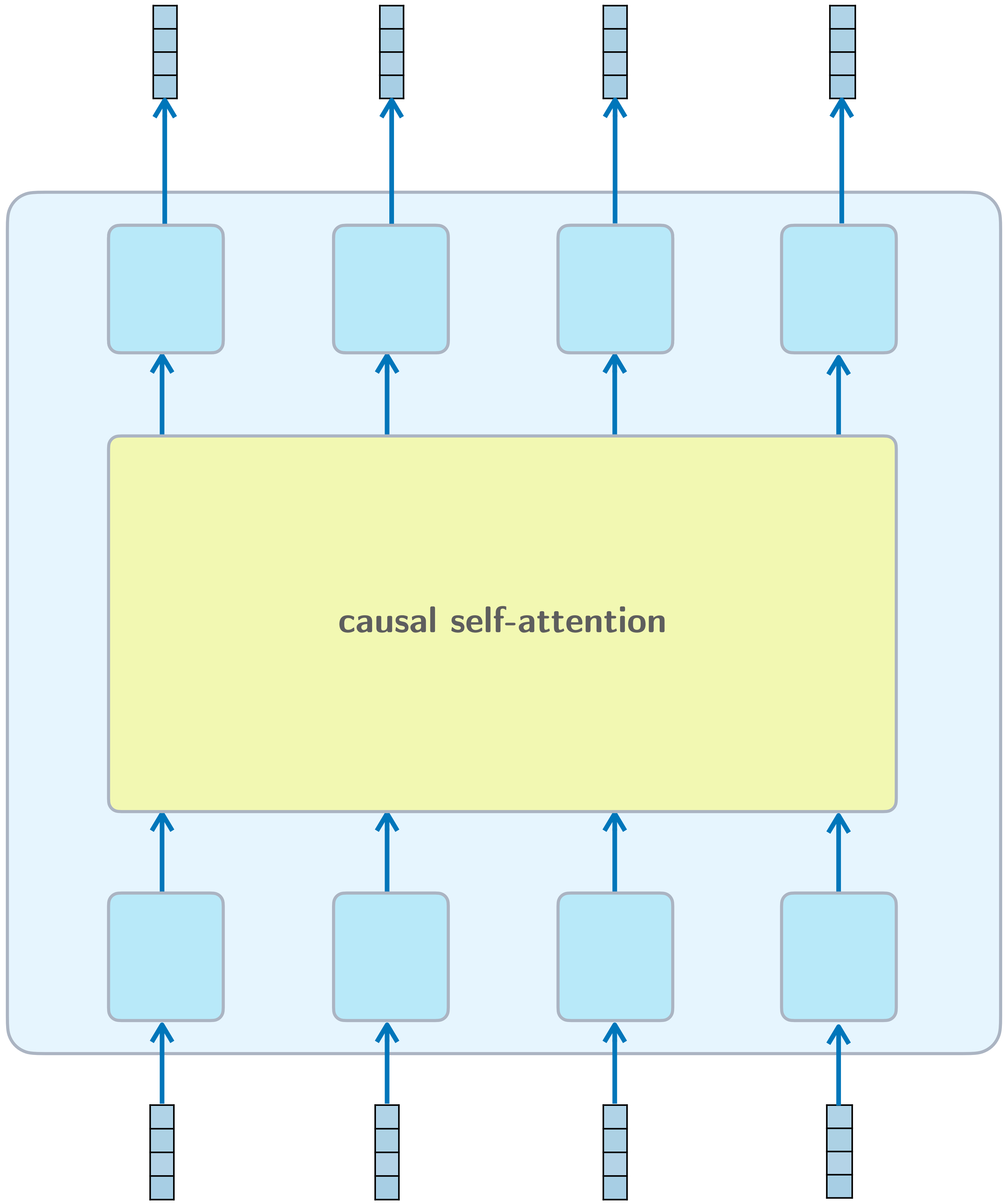
such that if you apply **unembed** you get **logits** over the vocabulary for next token

refines and ***contextualizes*** **decoder sequence**

decoder sequence ***X***

Causal Self-Attention

Feed **decoder** information into **decoder**

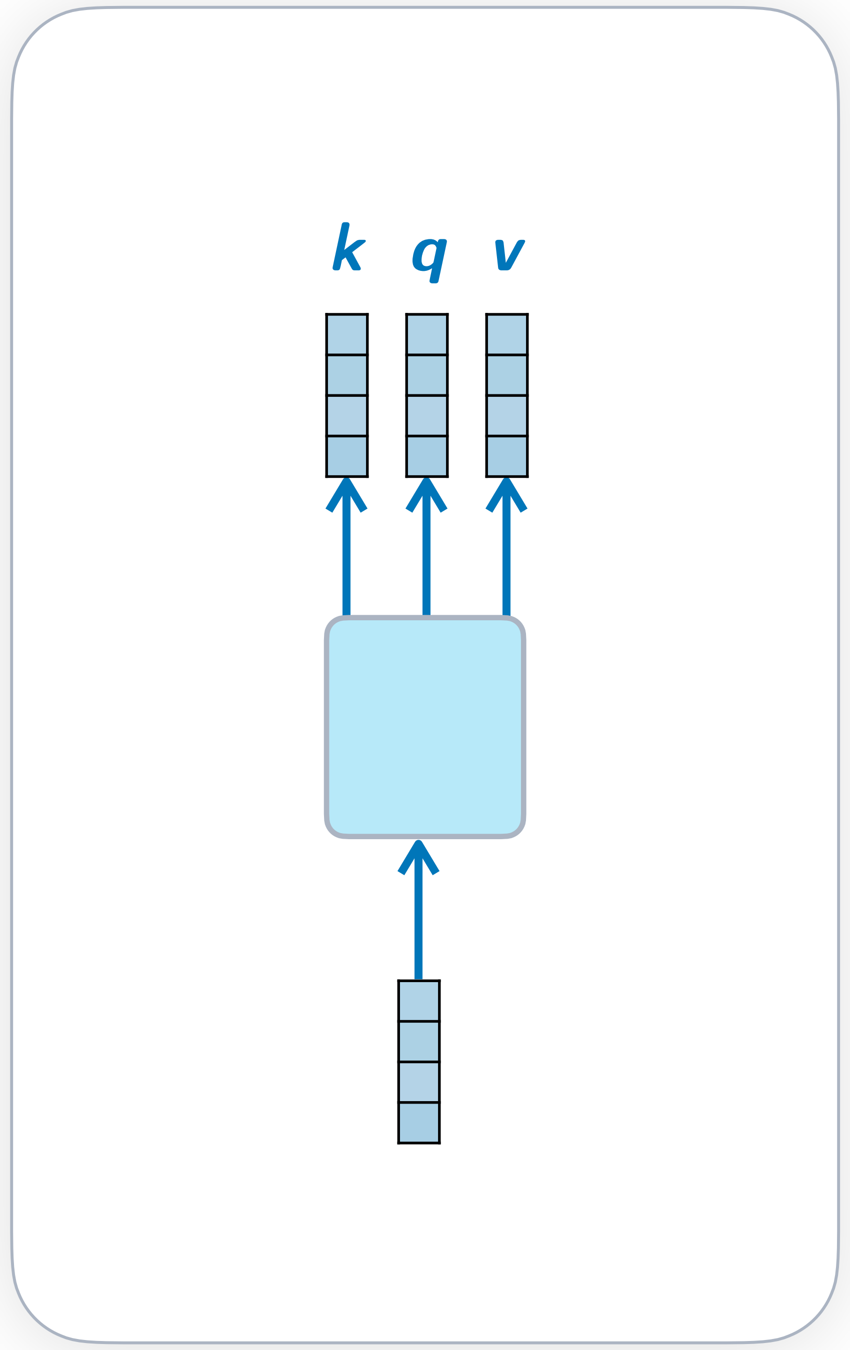


refined **decoder sequence** ***X′***

such that if you apply **unembed** you get **logits** over the vocabulary for next token

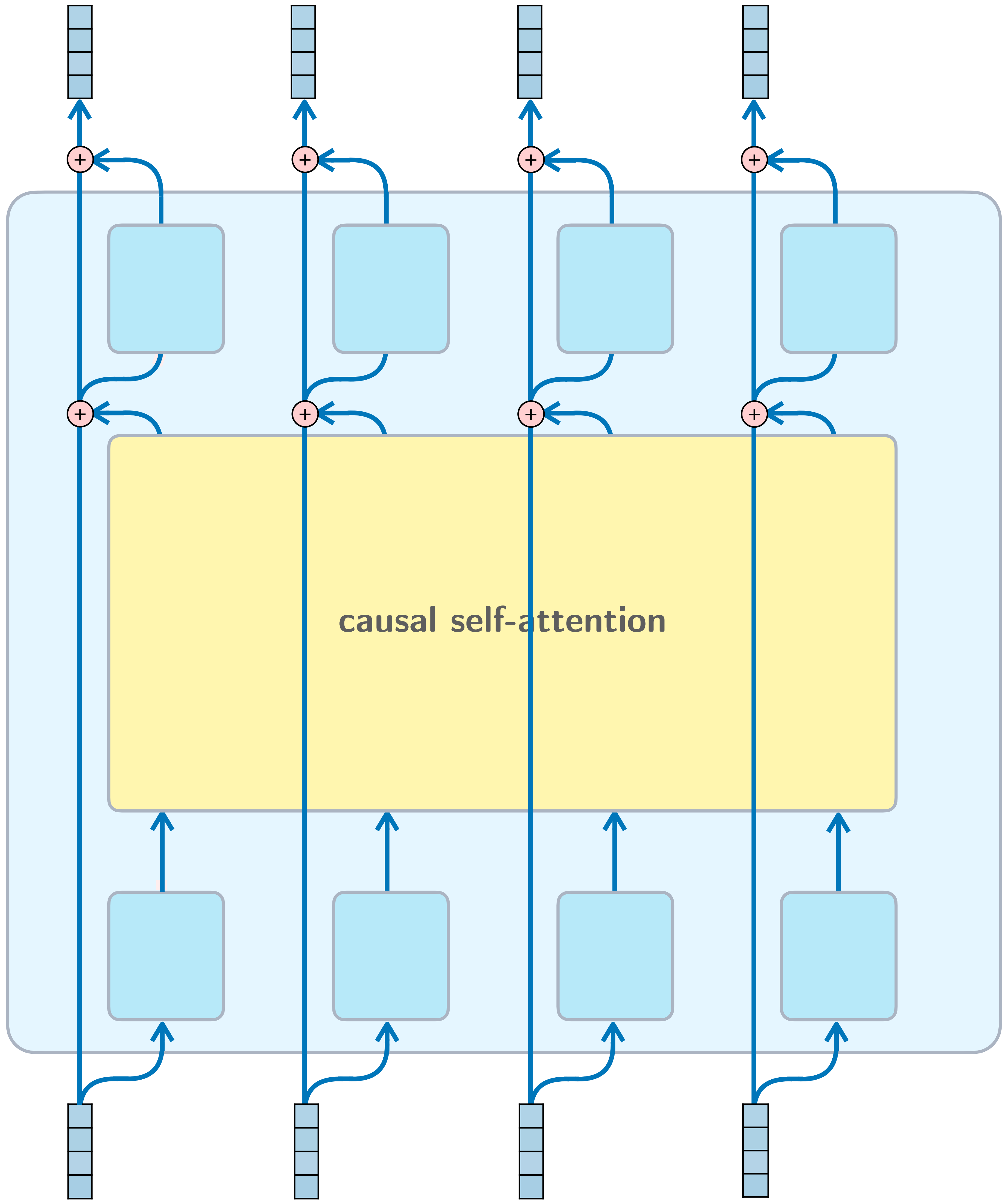
refines and ***contextualizes*** **decoder sequence**

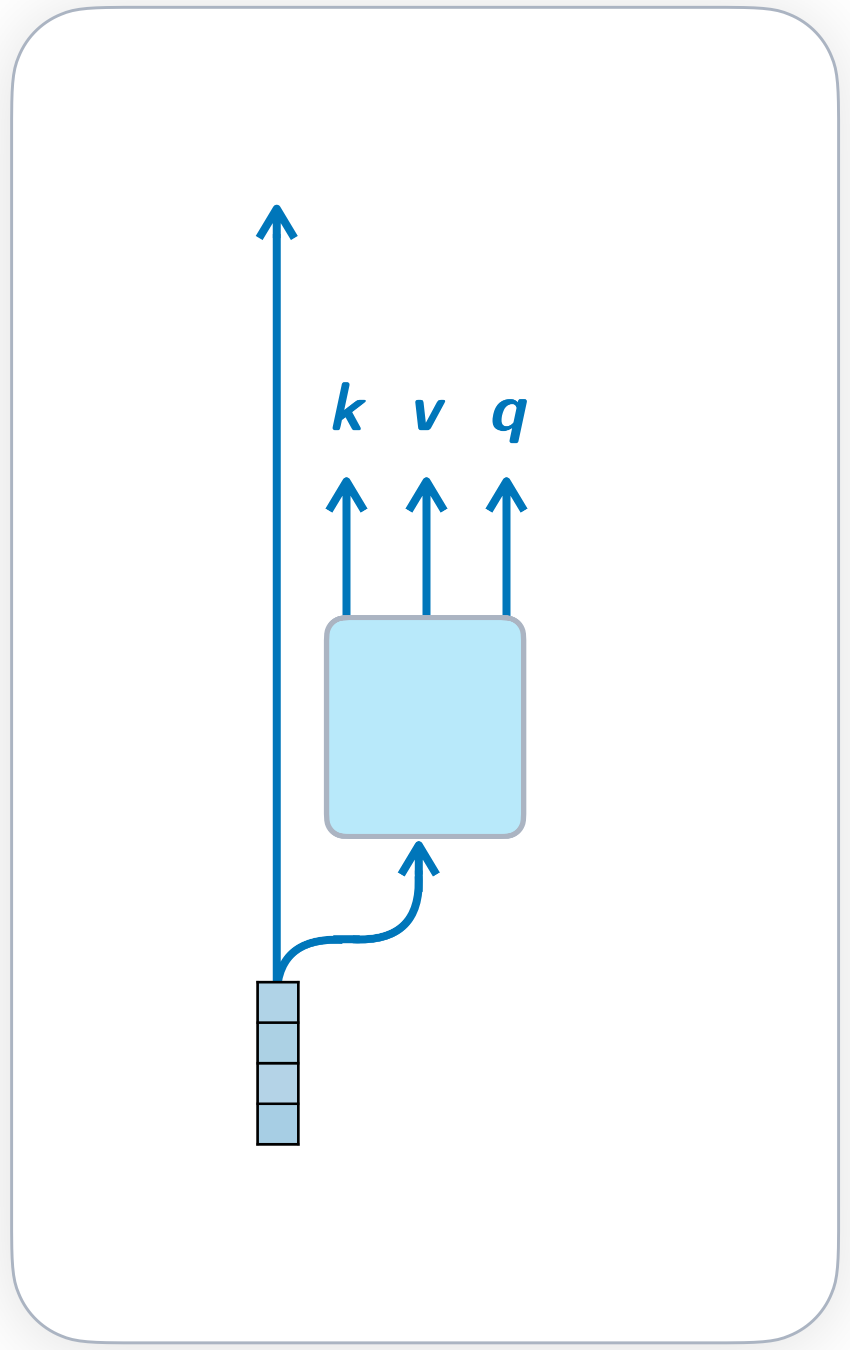
decoder sequence ***X***

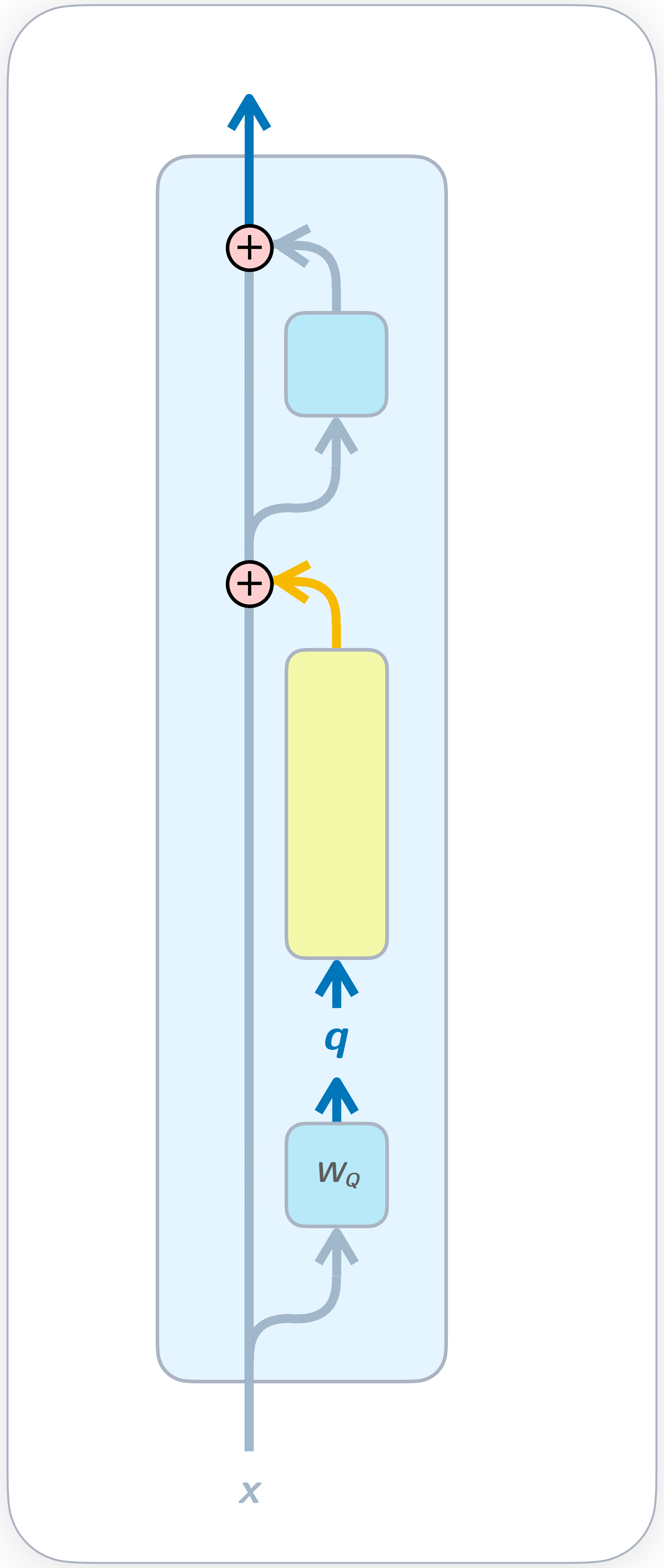


The ***residual stream***

Each token has its own ***residual stream***

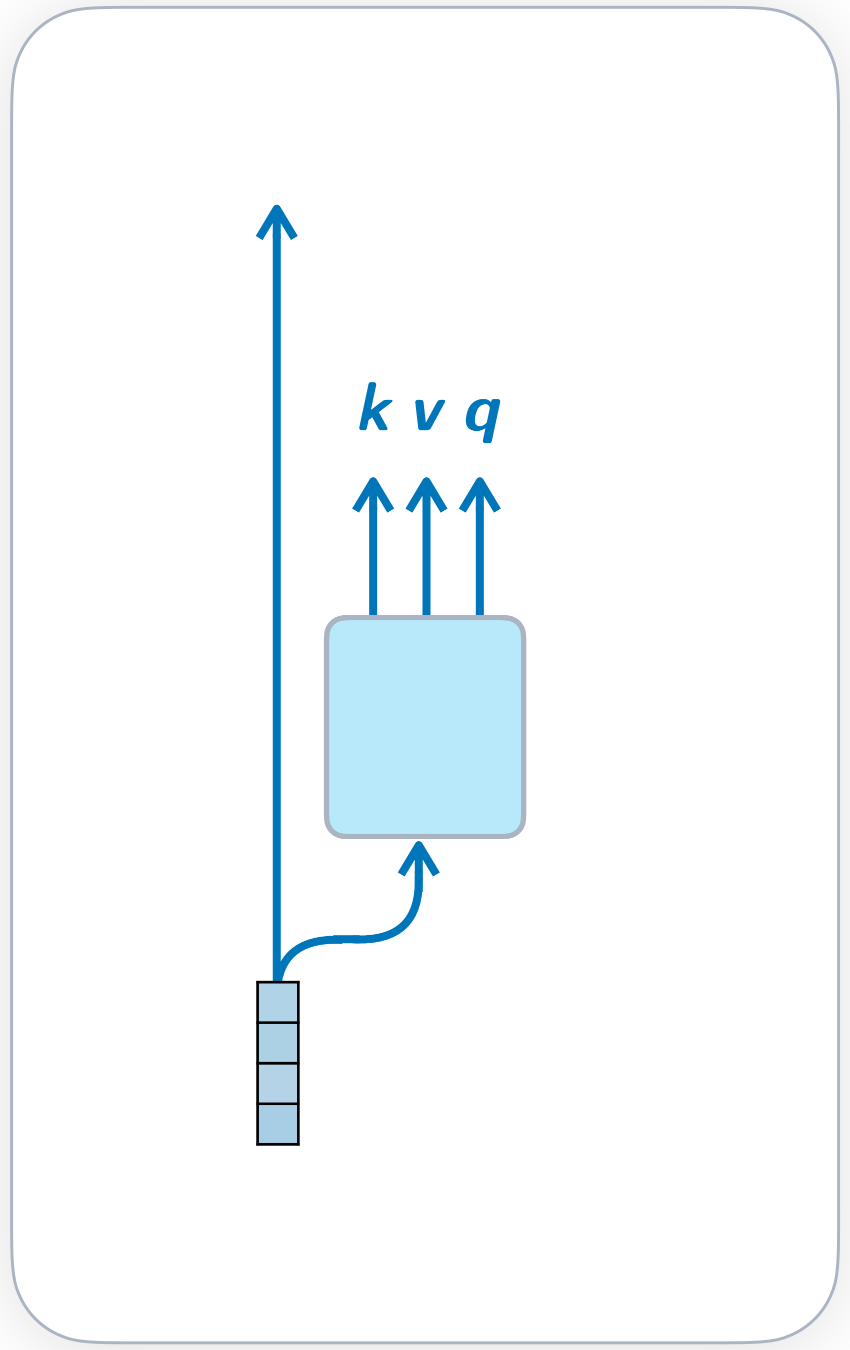


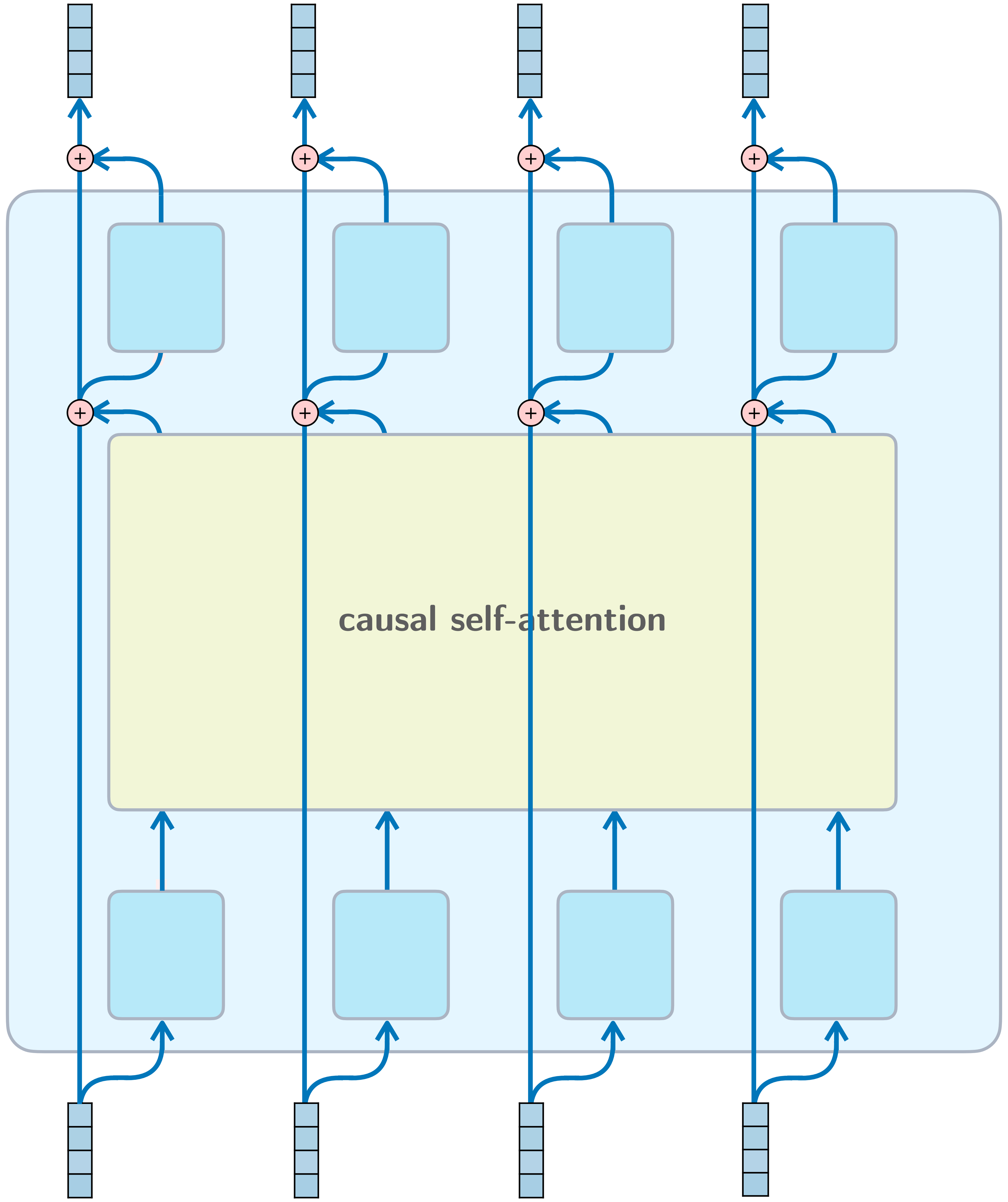


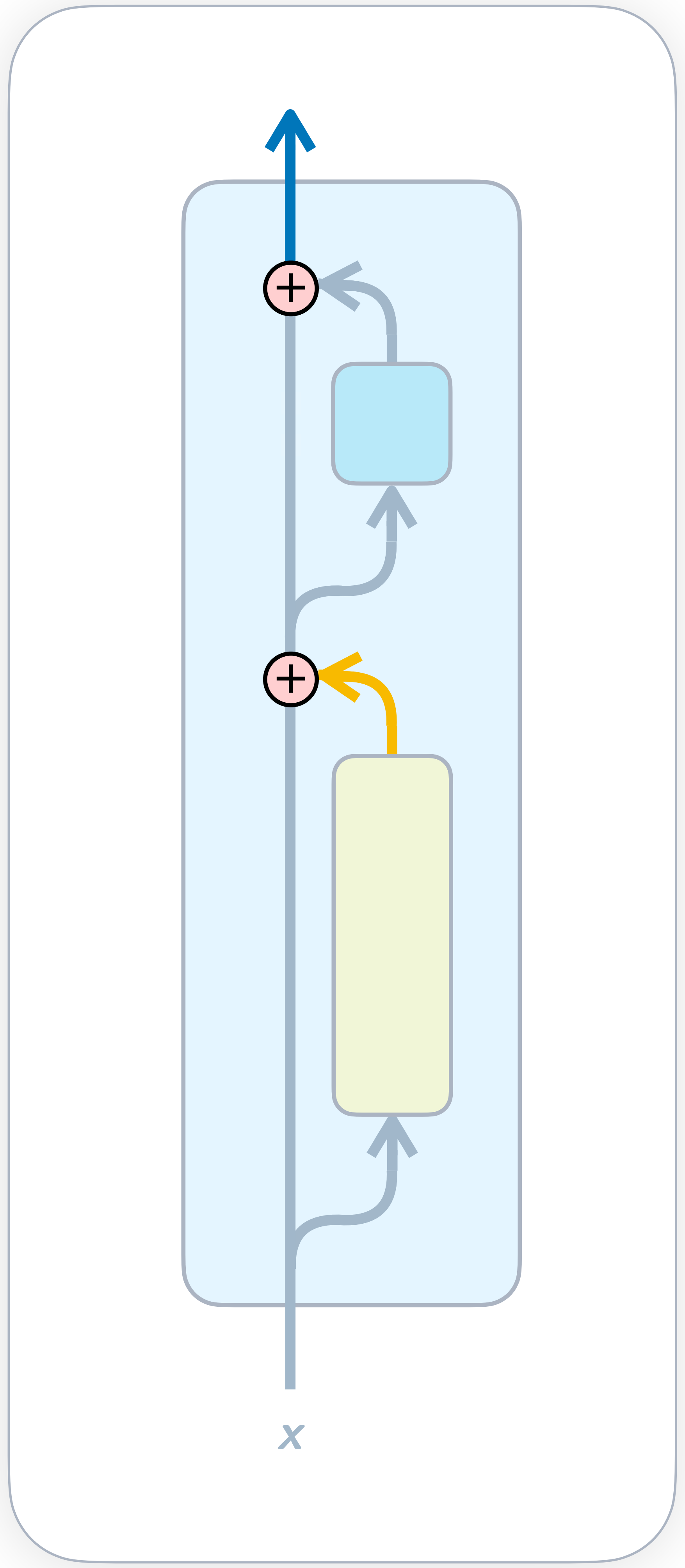


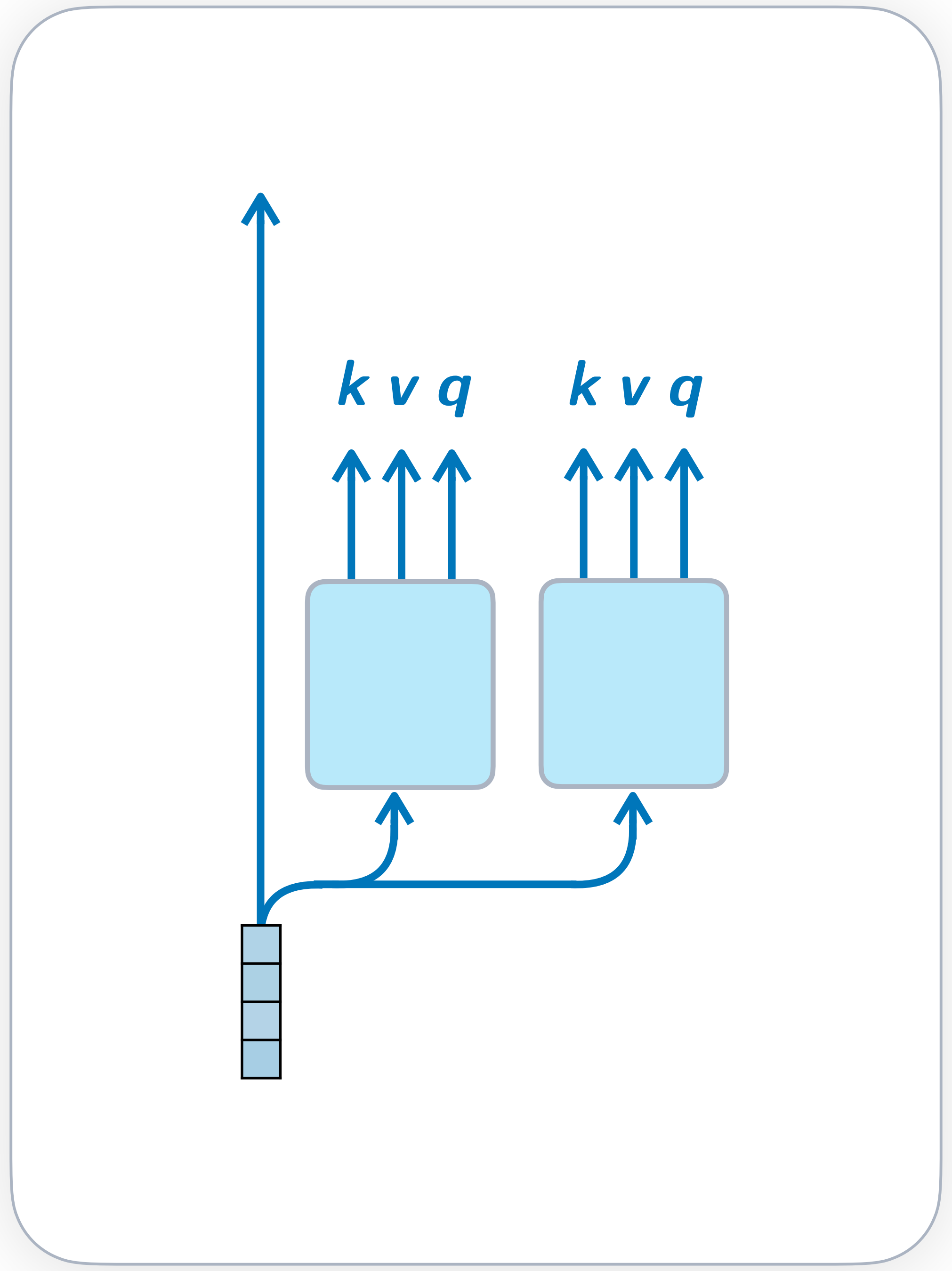
0

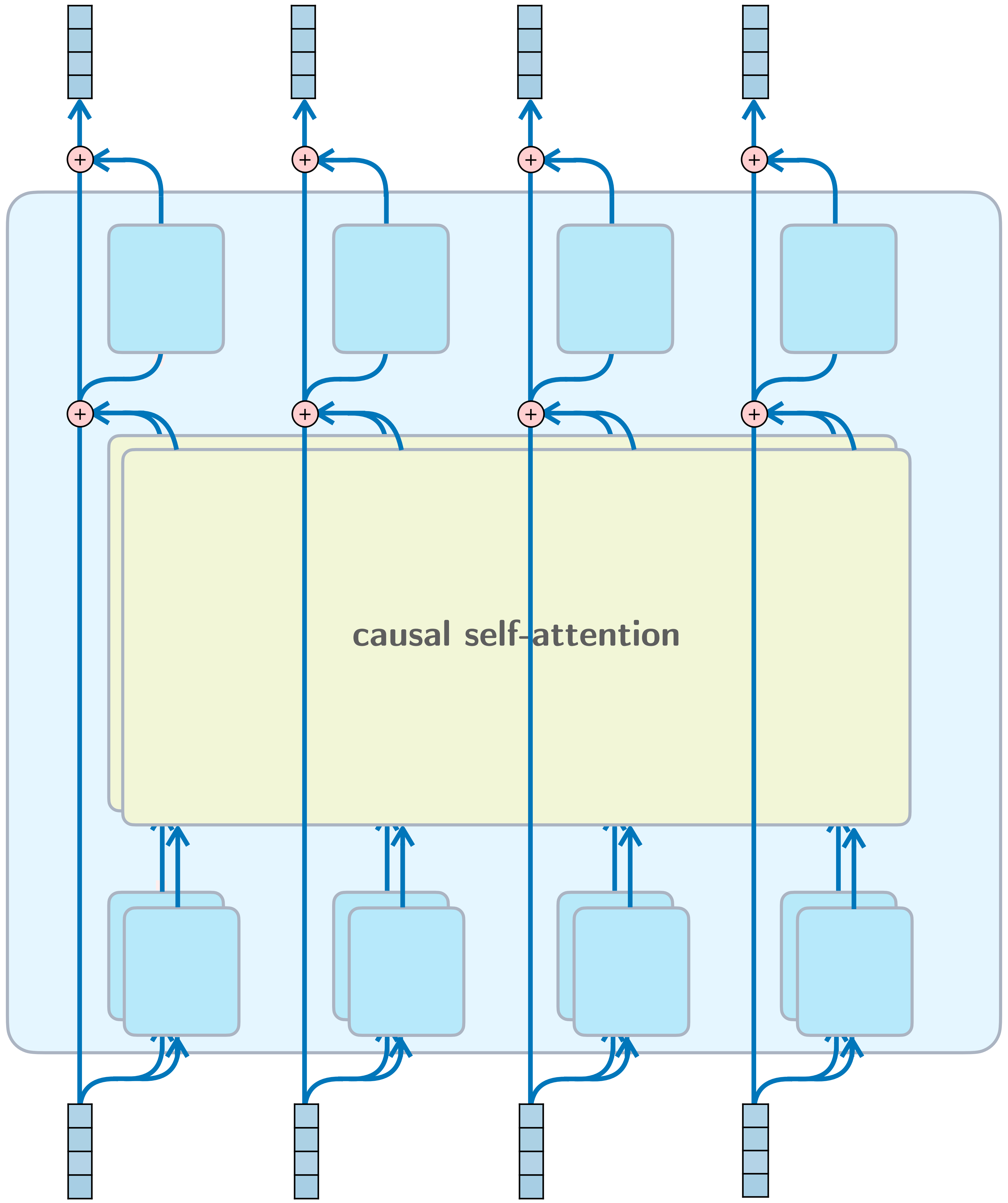
Mathematical framework

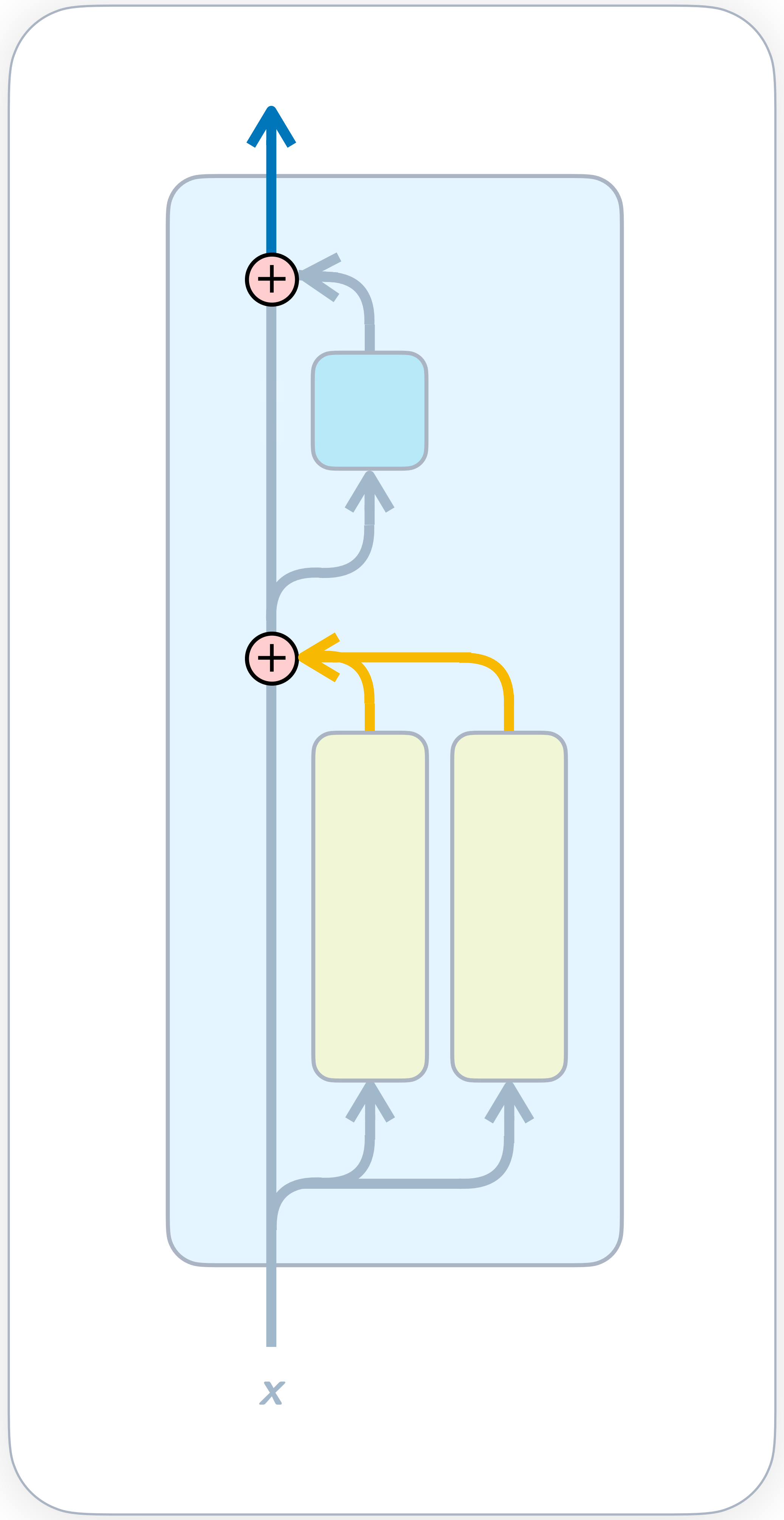

k v q


causal self-attention


x


k v q
k v q

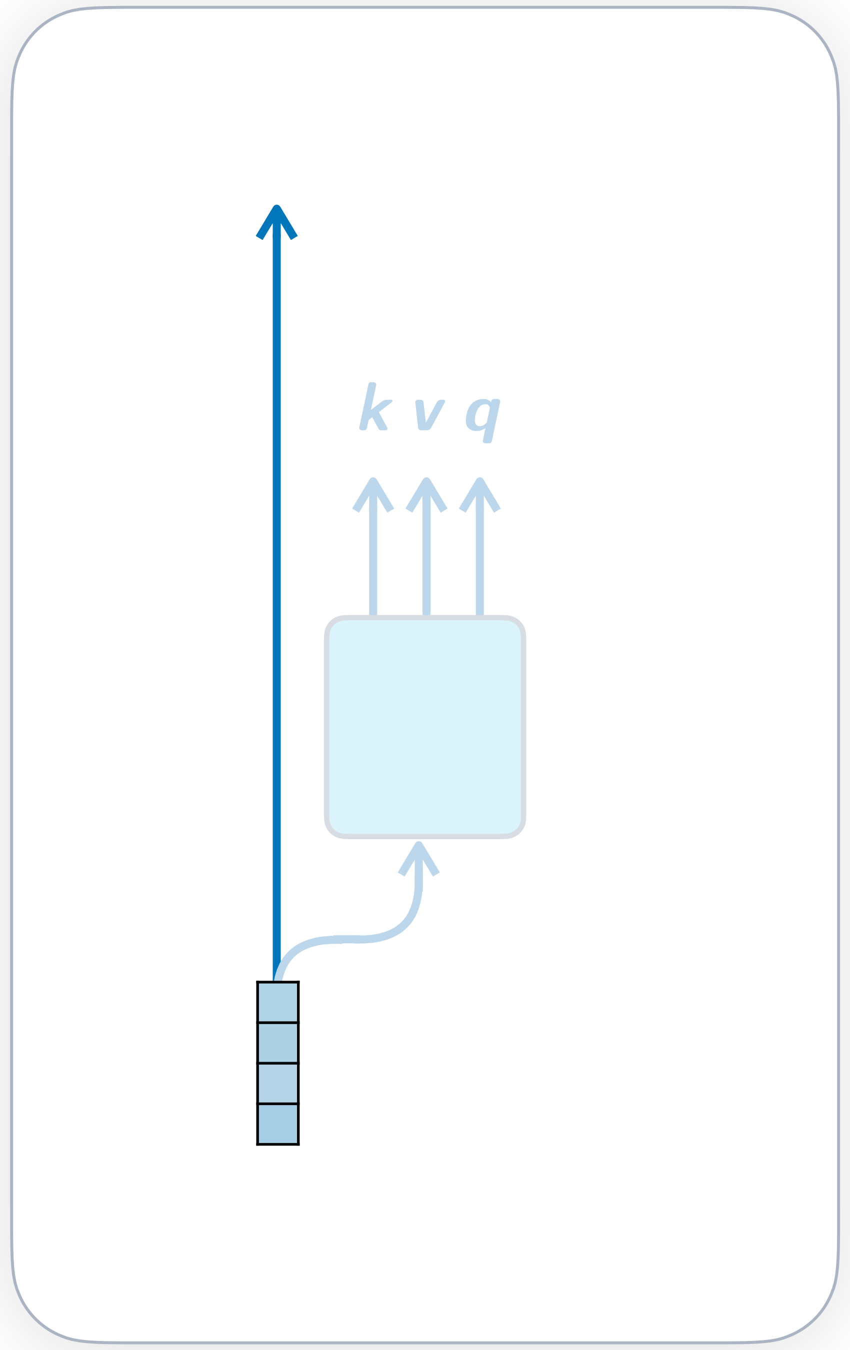

causal self-attention

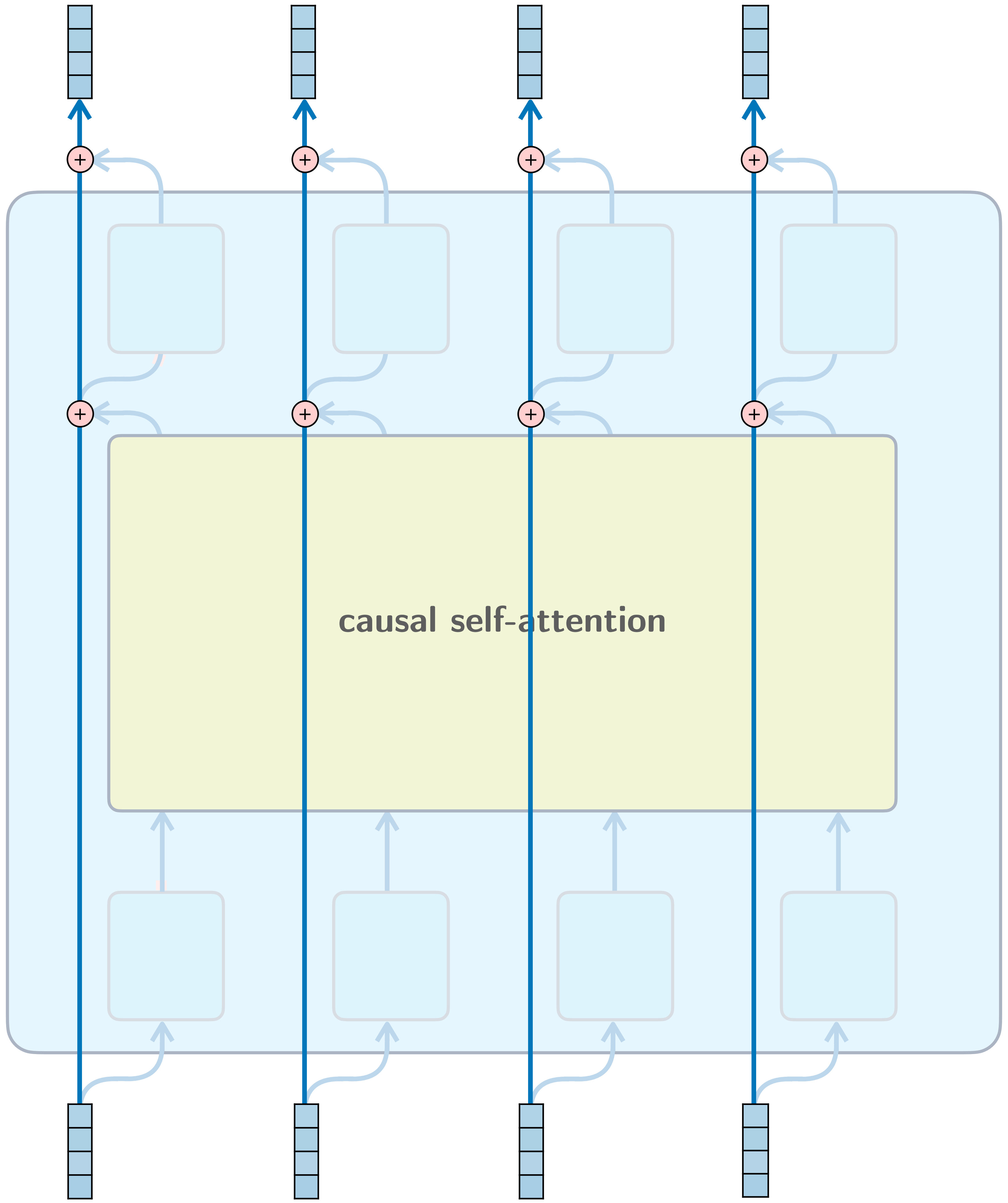

x

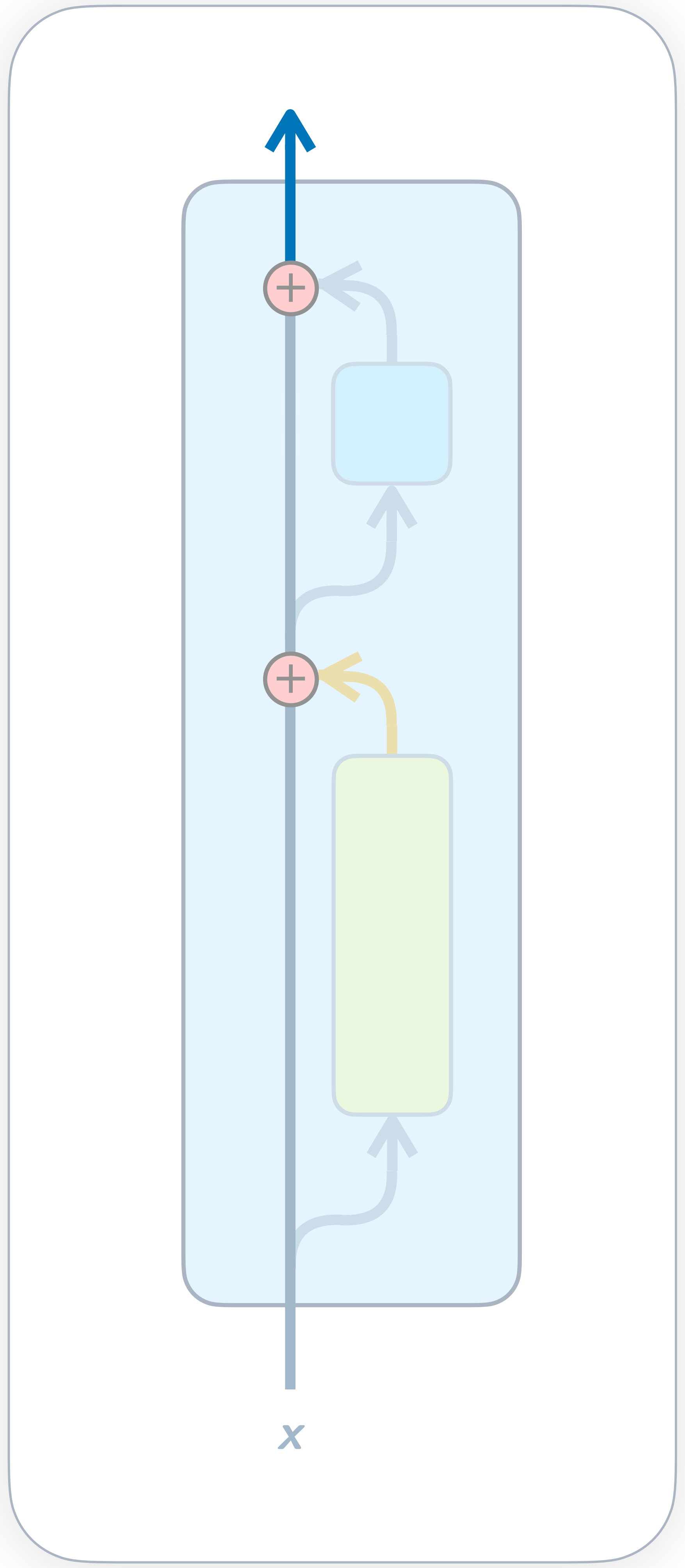
The ***residual stream***

Each token has its own ***residual stream***

The residual stream is the central object.
Information is "read from" and "written to"
this residual stream.



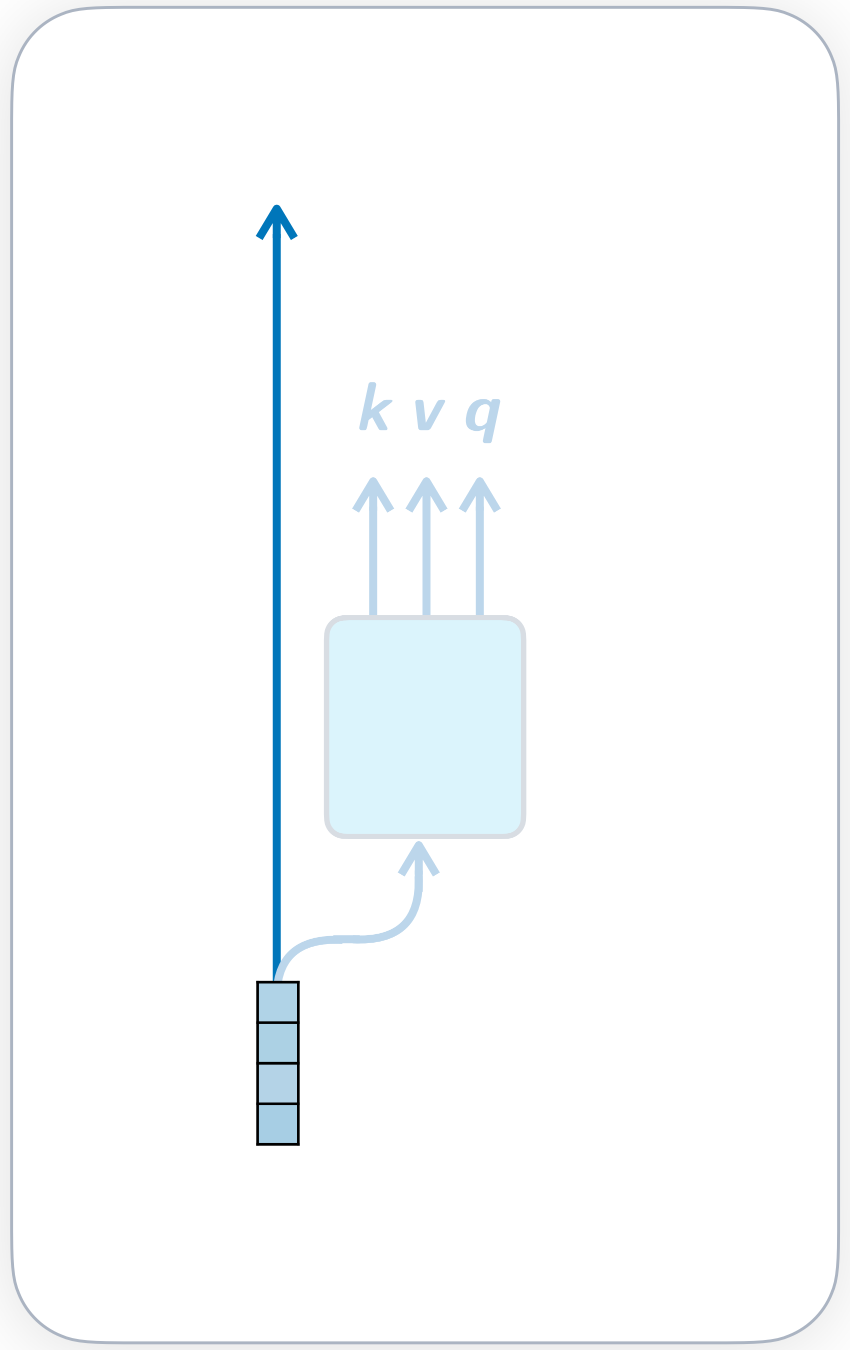


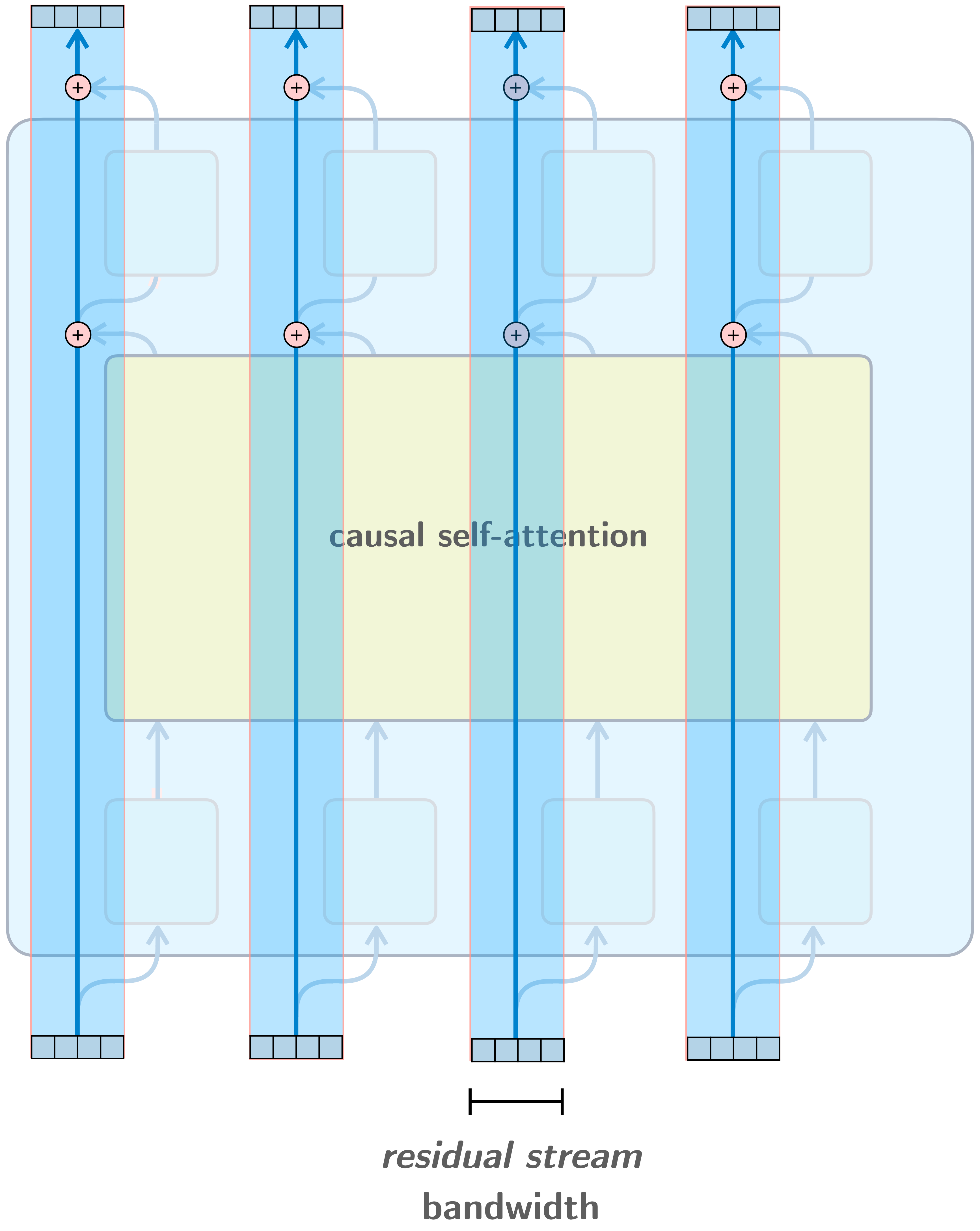


The ***residual stream***

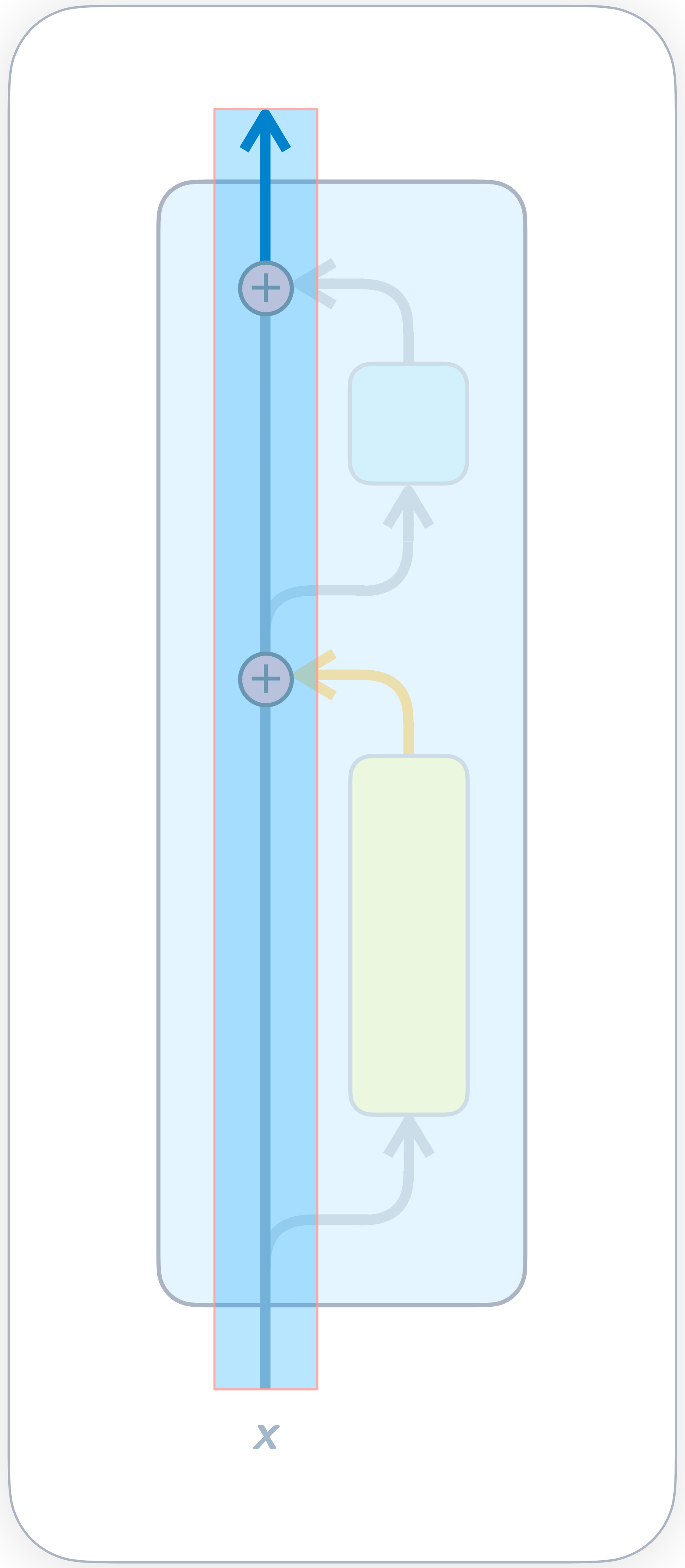
Each token has its own ***residual stream***

The residual stream is the central object. Information is "read from" and "written to" this residual stream.





Here: 4 neurons



The ***residual stream***

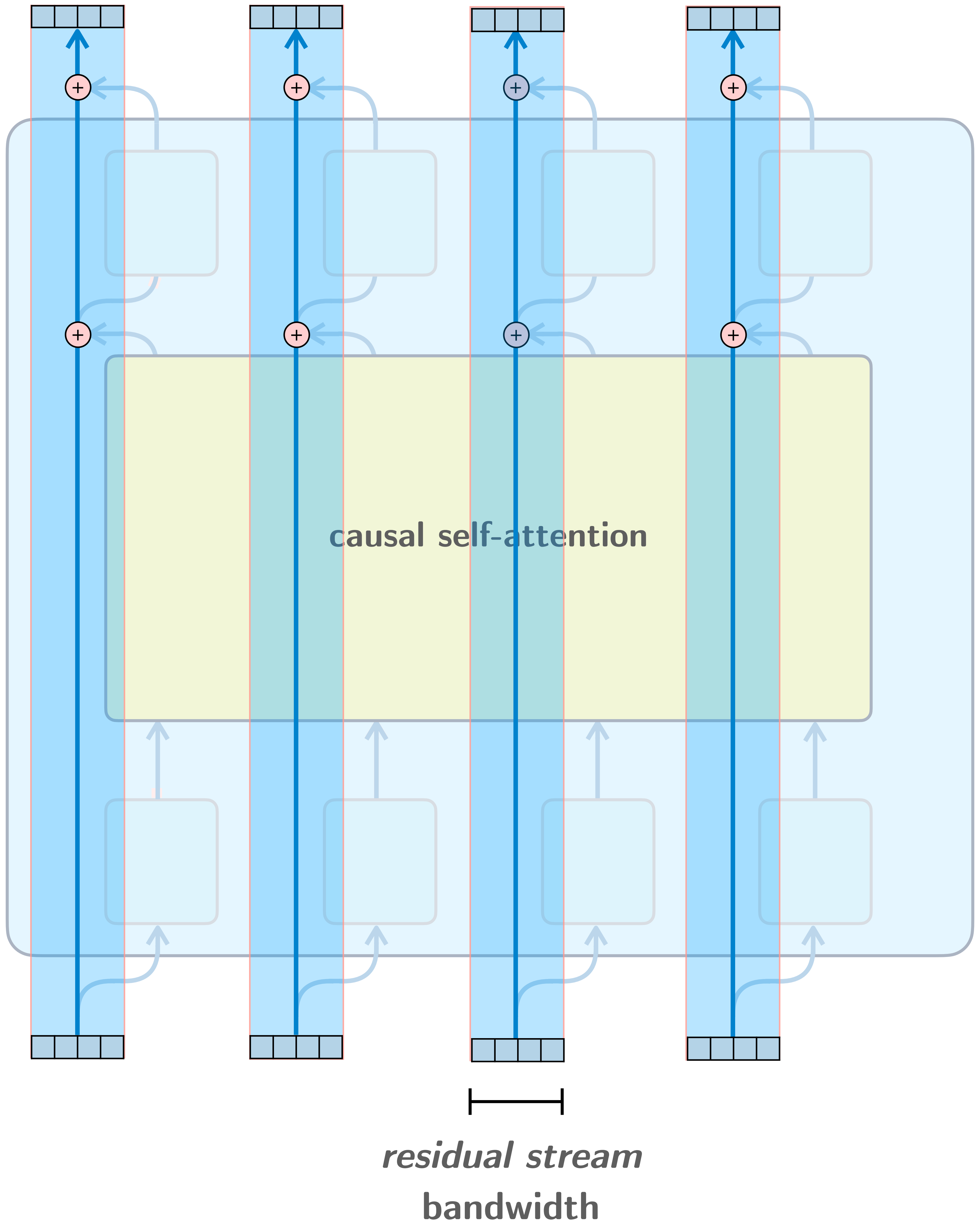
Each token has its own ***residual stream***
But they all represent the same **space**

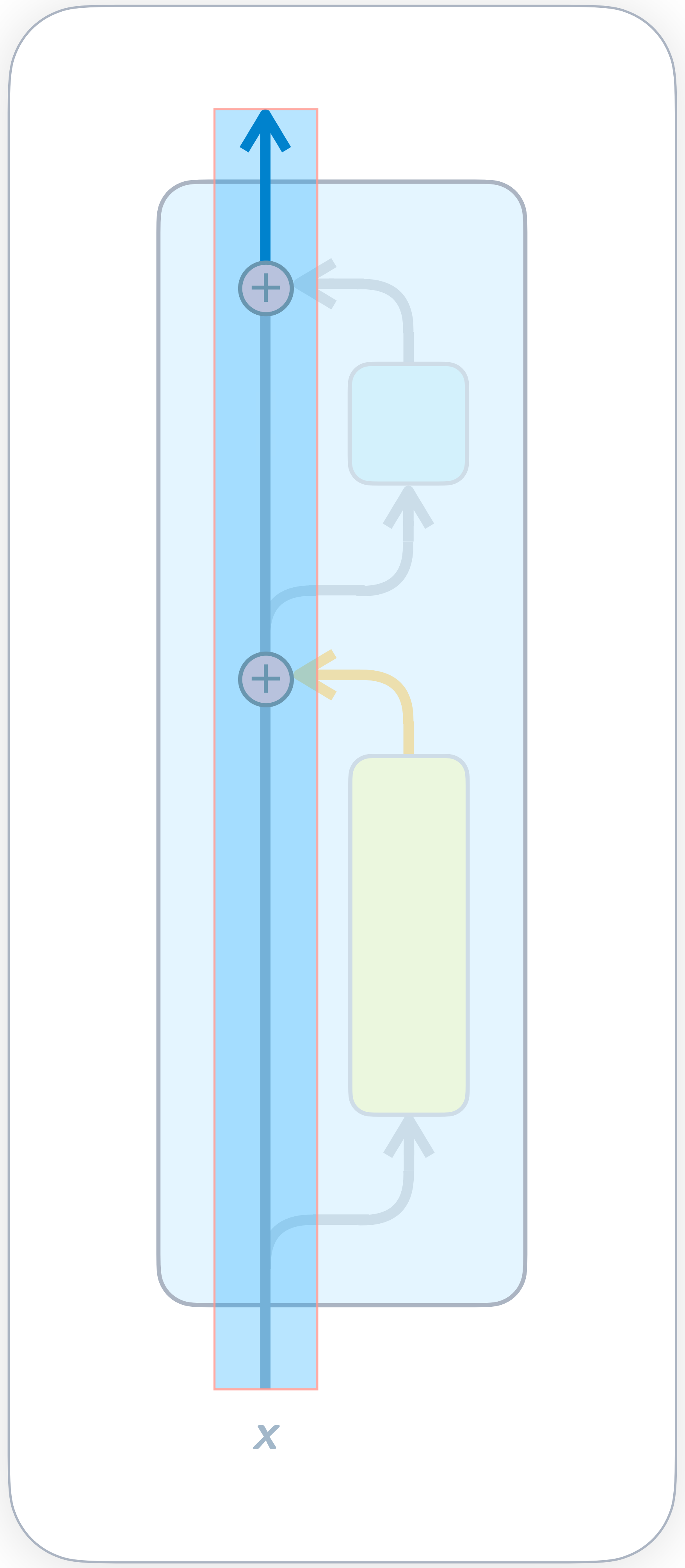
Example of embedding subspaces:

current token
previous token
next token
position

They don't necessary correspond to neurons,
i.e. individual rows in the residual stream vector,
i.e. in the embedding vector of a given token.

When trying to explain a circuit, for reasons of simplicity, we might stipulate a particular subspace and give it a legible name
What's important is that the model will store this information somewhere in this embedding space,
And it will not use the whole space for any given sort of information, hence only a subspace



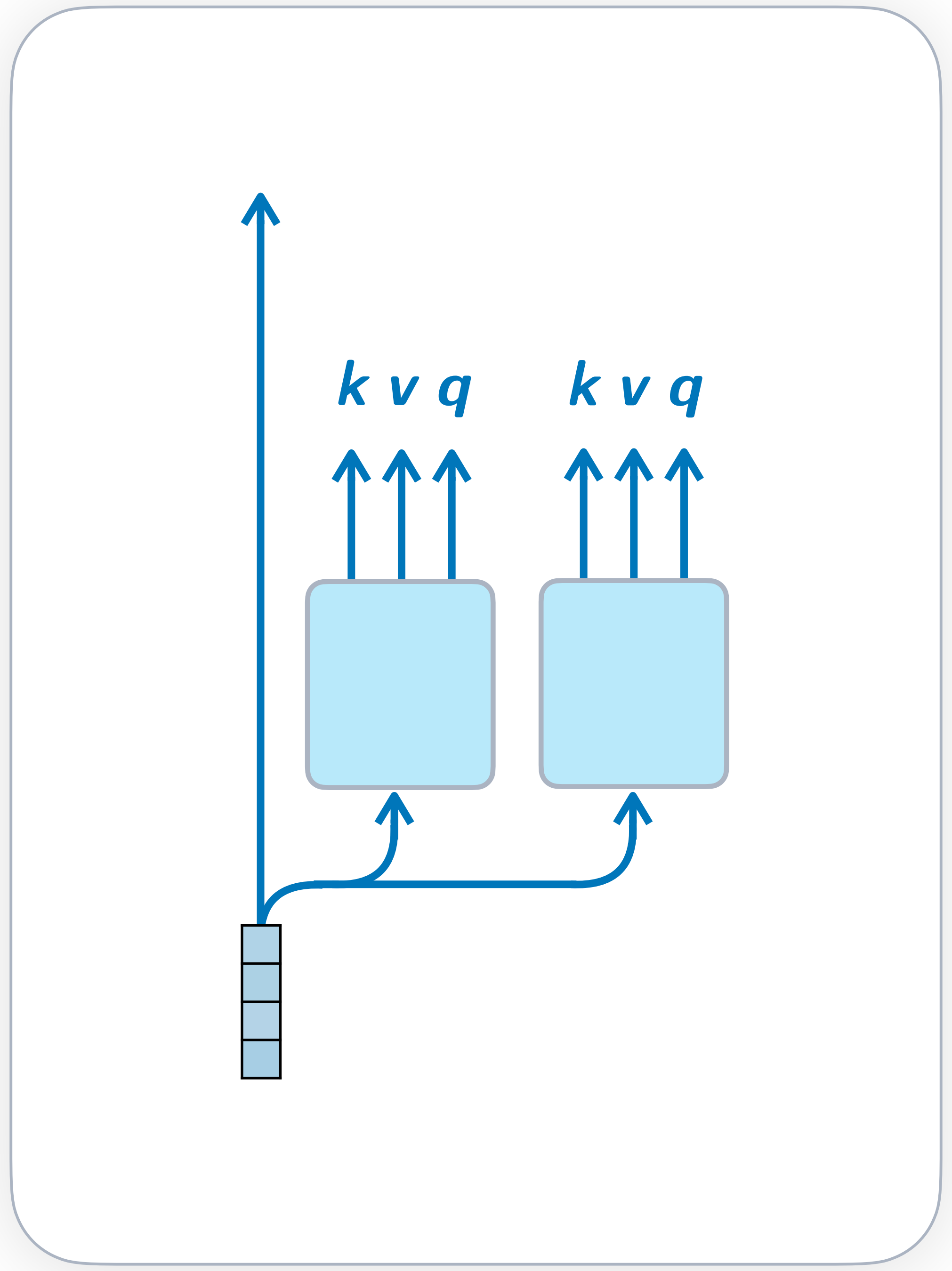


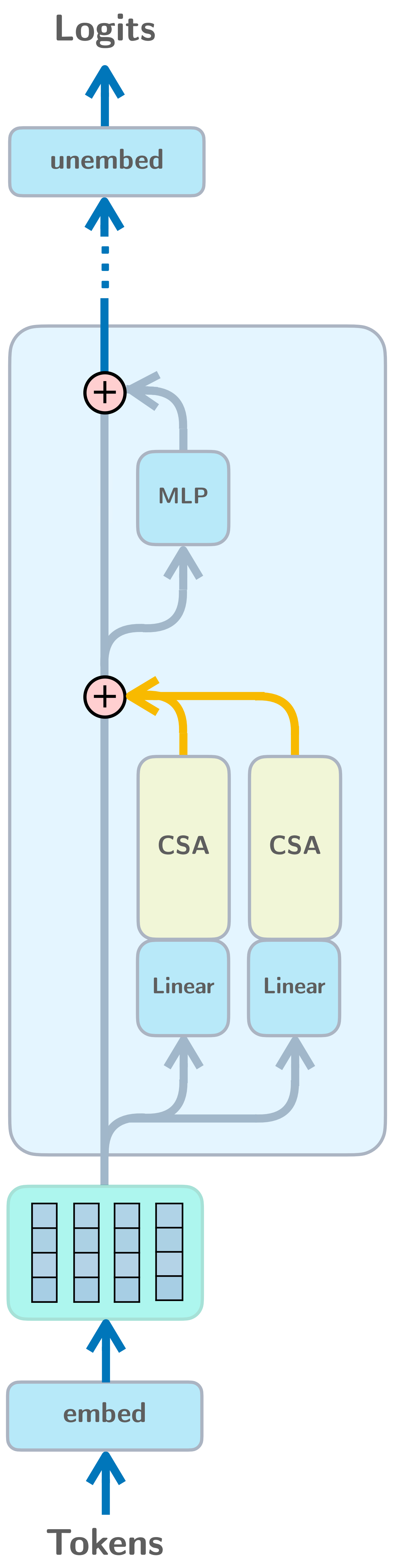
The ***residual stream***

Each token has its own ***residual stream***

The residual stream is the central object. Information is "read from" and "written to" this residual stream.

The value of the residual stream at any point is simply the initial embedding plus the sum of layer outputs.

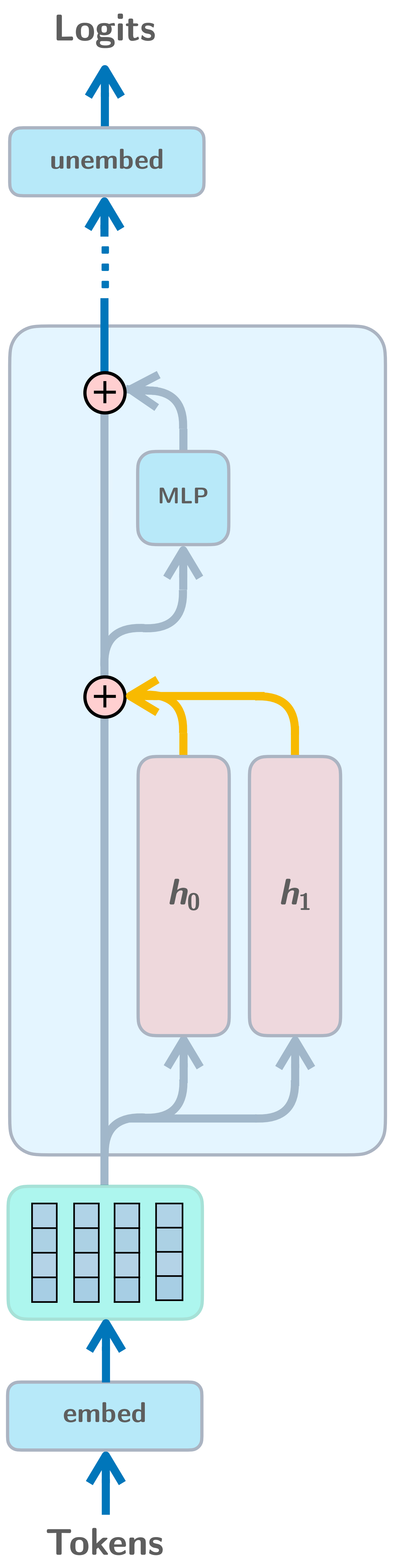




The ***residual stream***

Each token has its own ***residual stream***

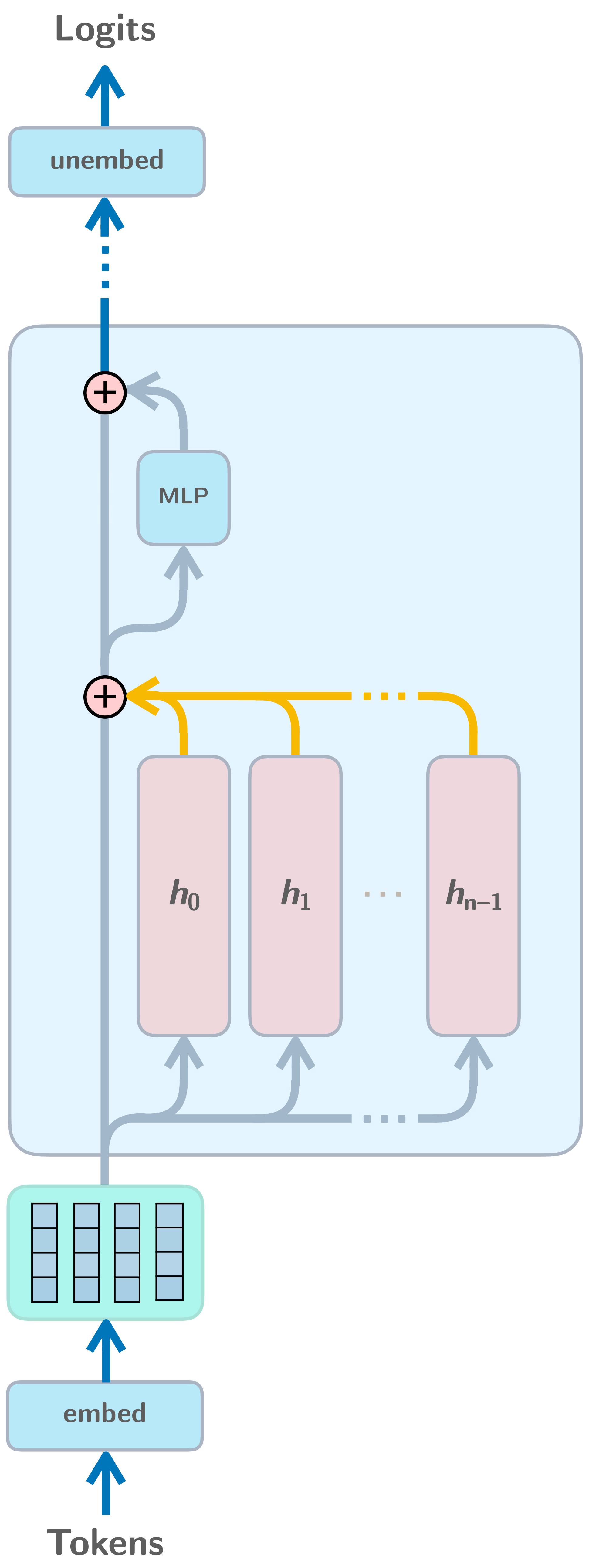
The residual stream is the central object. Information is "read from" and "written to" this residual stream.

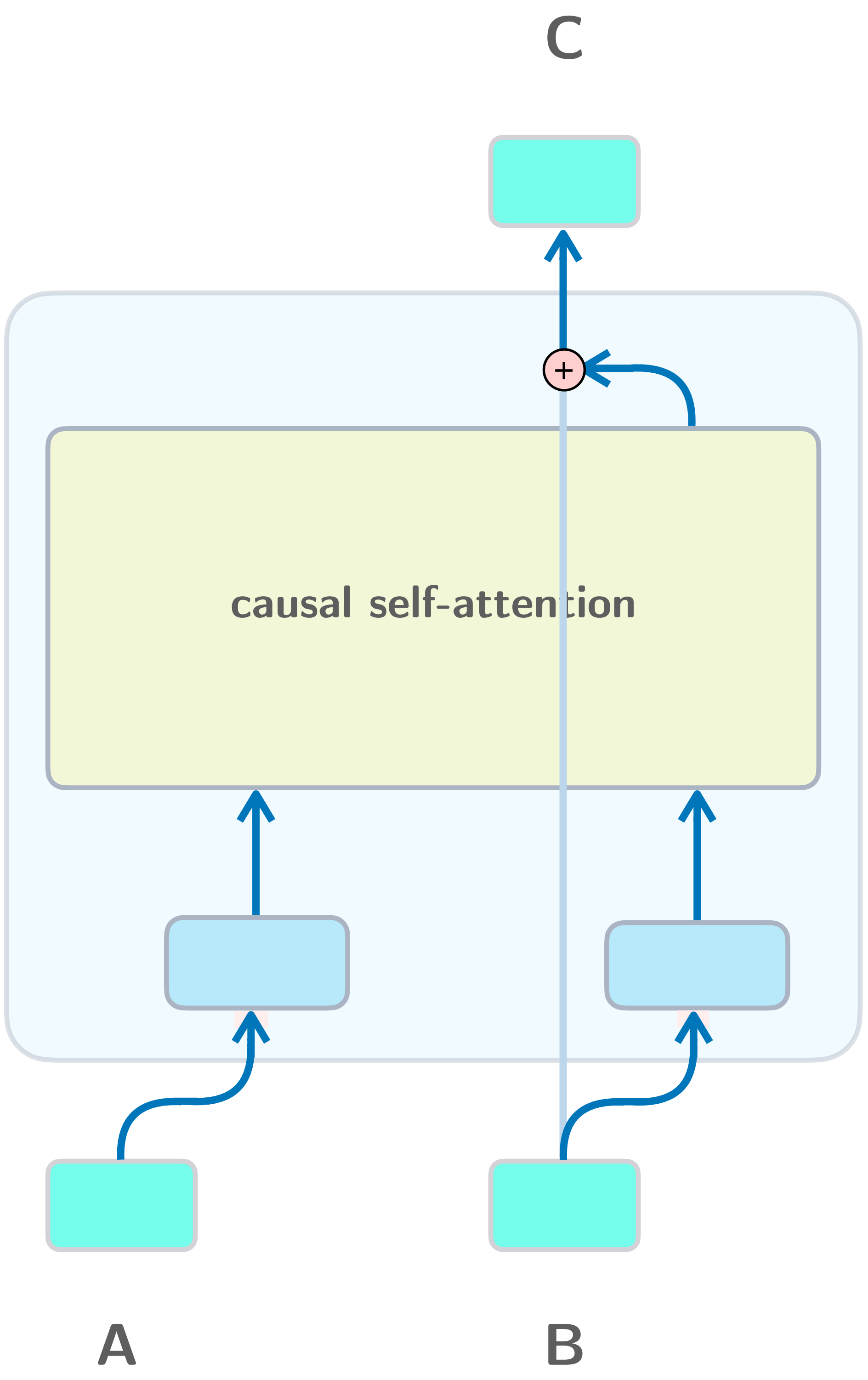


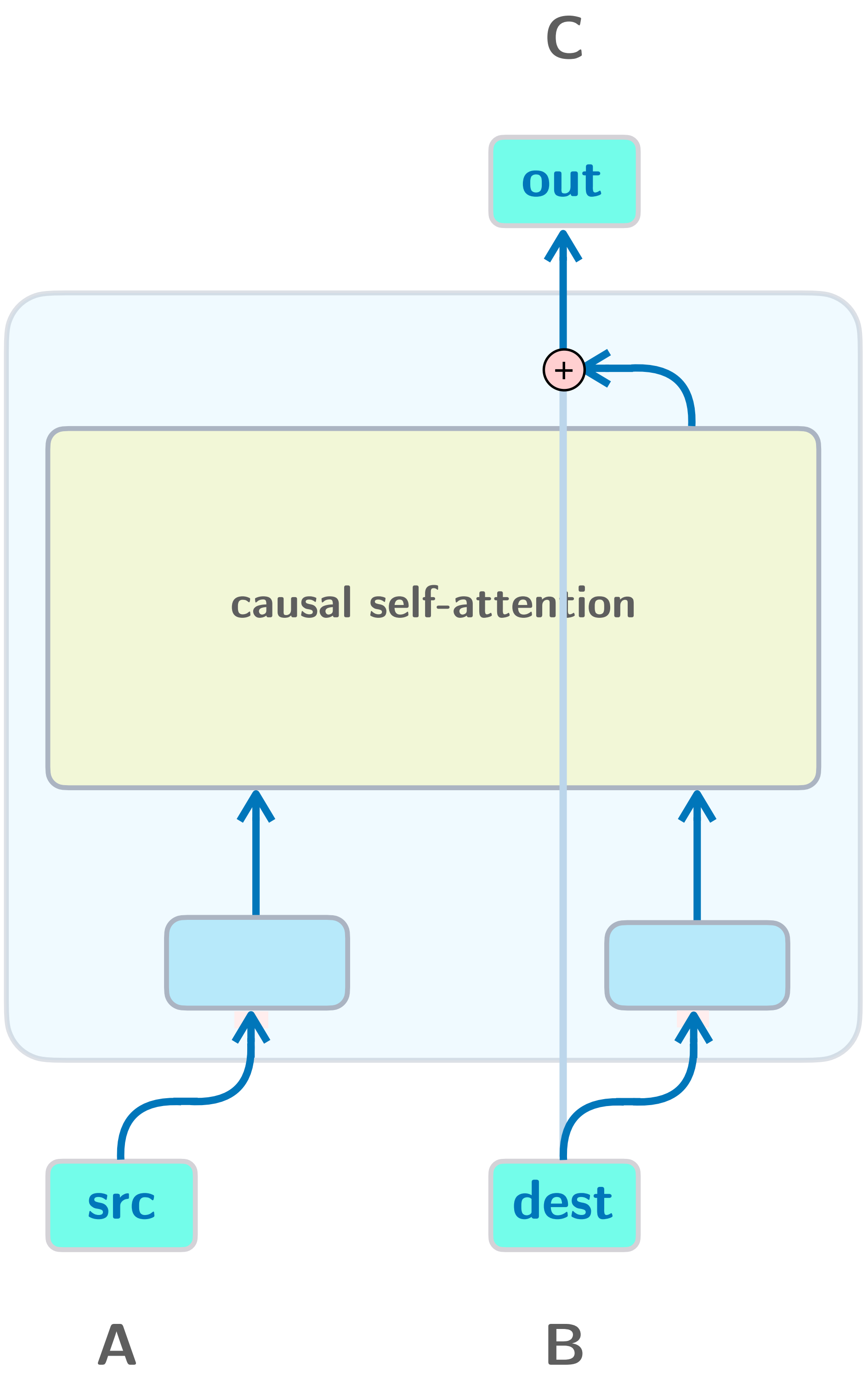
The ***residual stream***

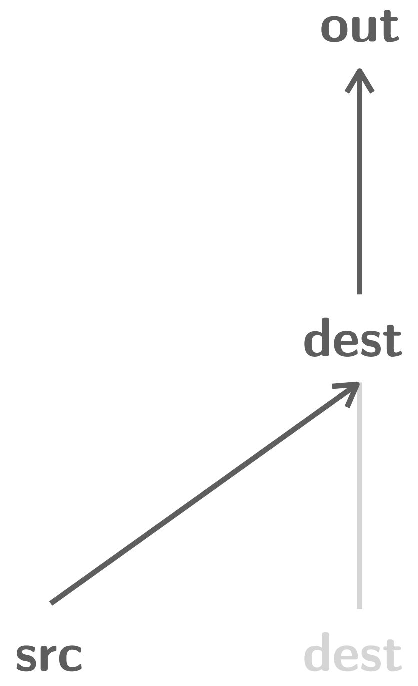
Each token has its own ***residual stream***

The residual stream is the central object. Information is "read from" and "written to" this residual stream.




C
causal self-attention
A
B

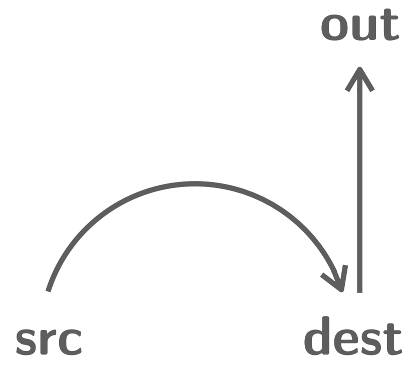




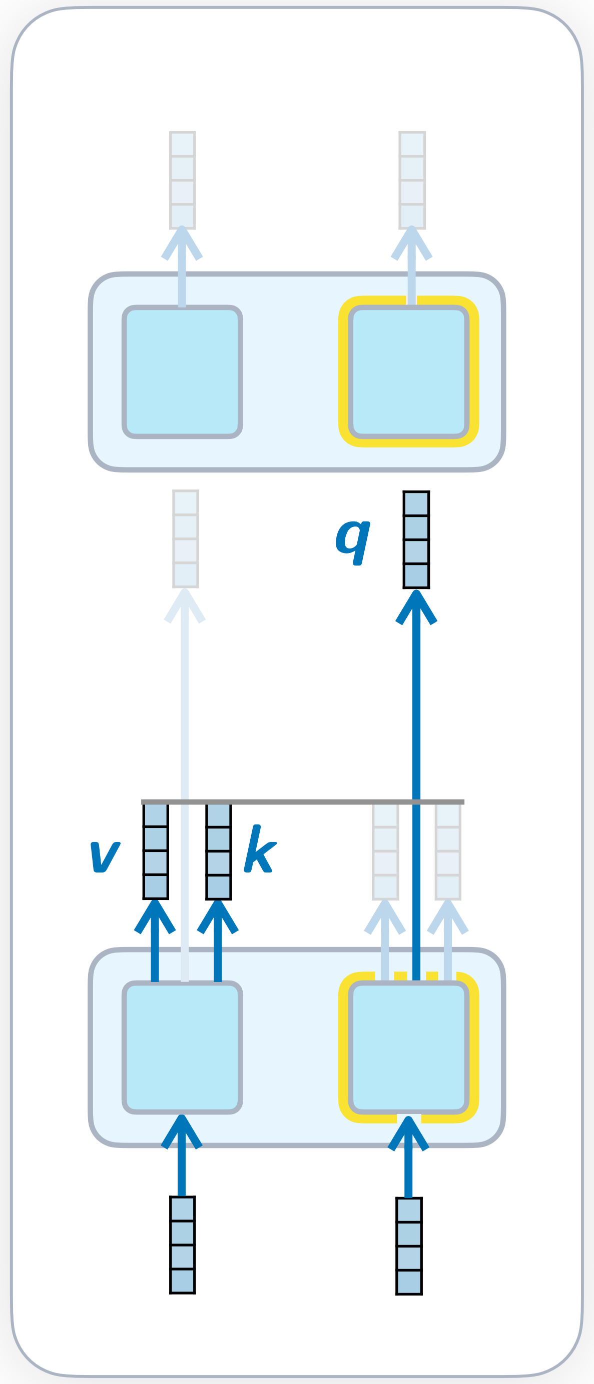
source of attention information

destination of attention information

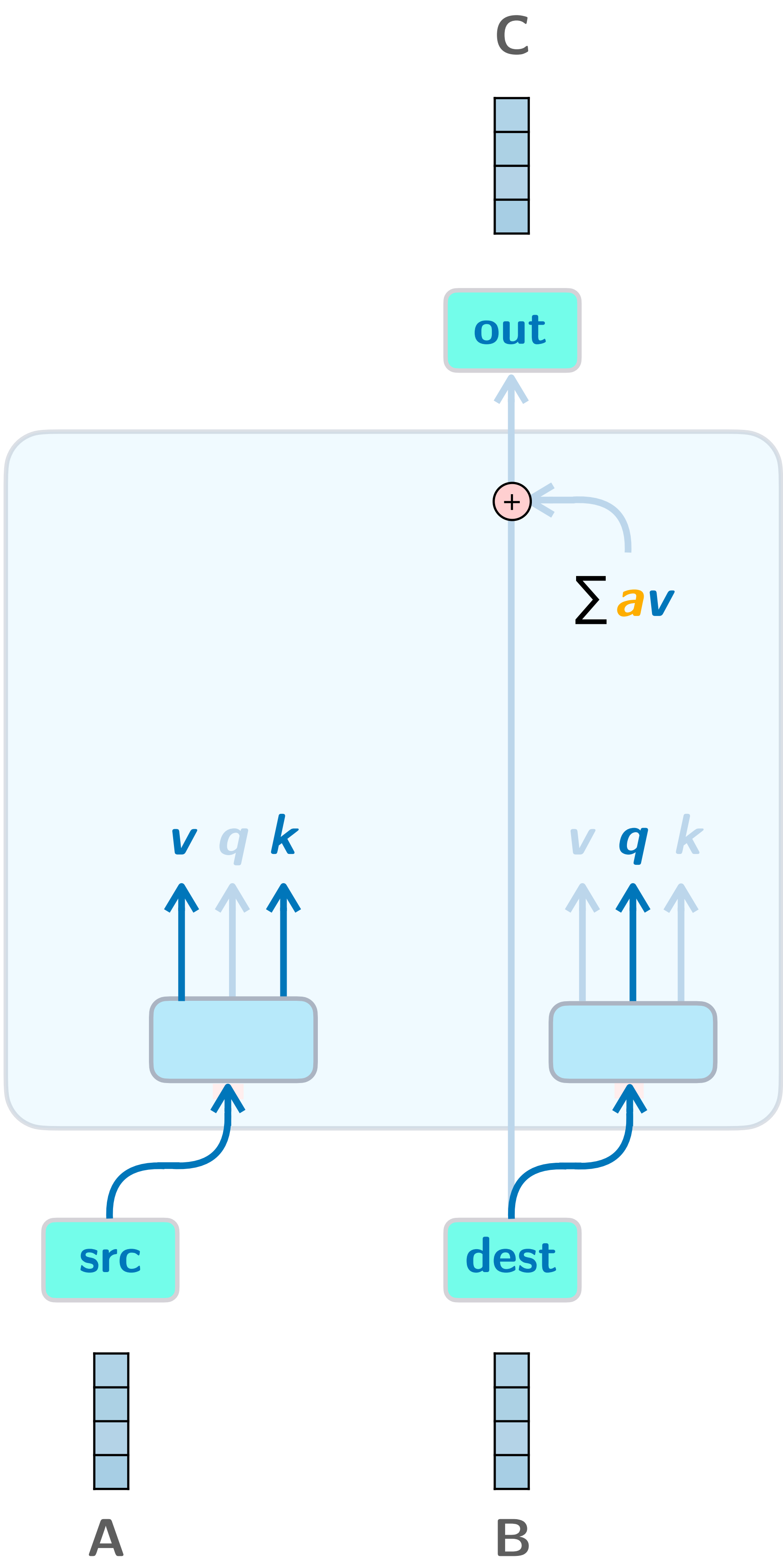
output of attention information



Copying context information from **source** to **destination**, in order to compute the **output**.



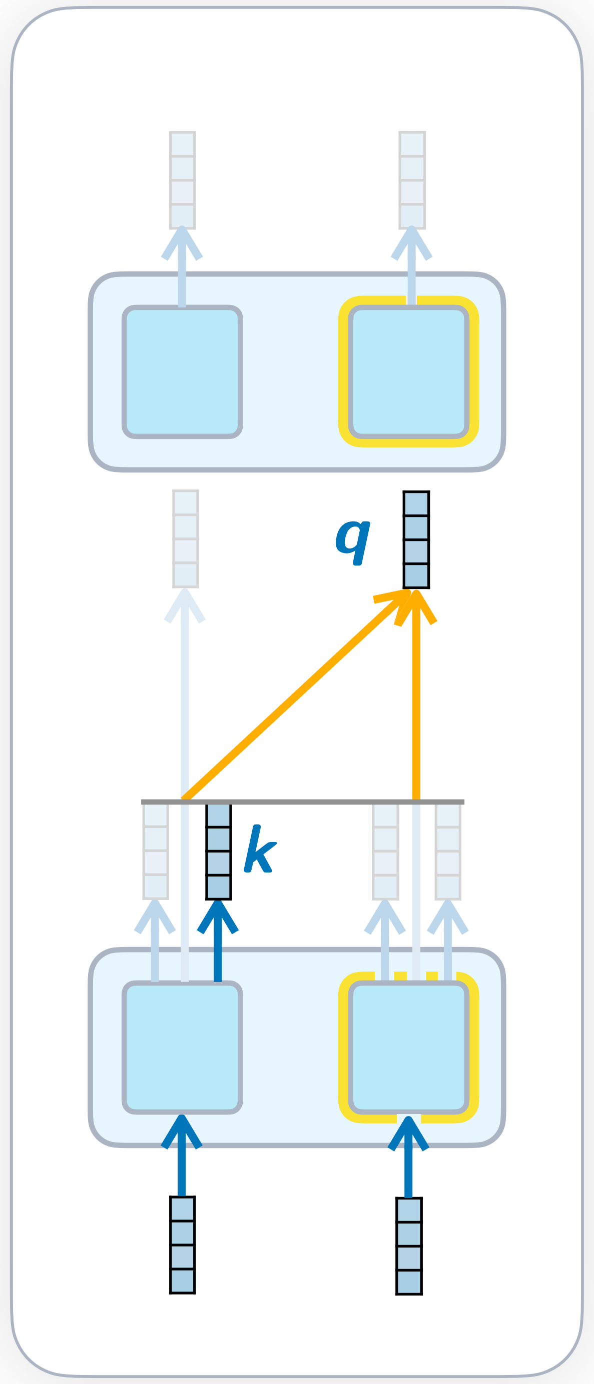
q
v
k

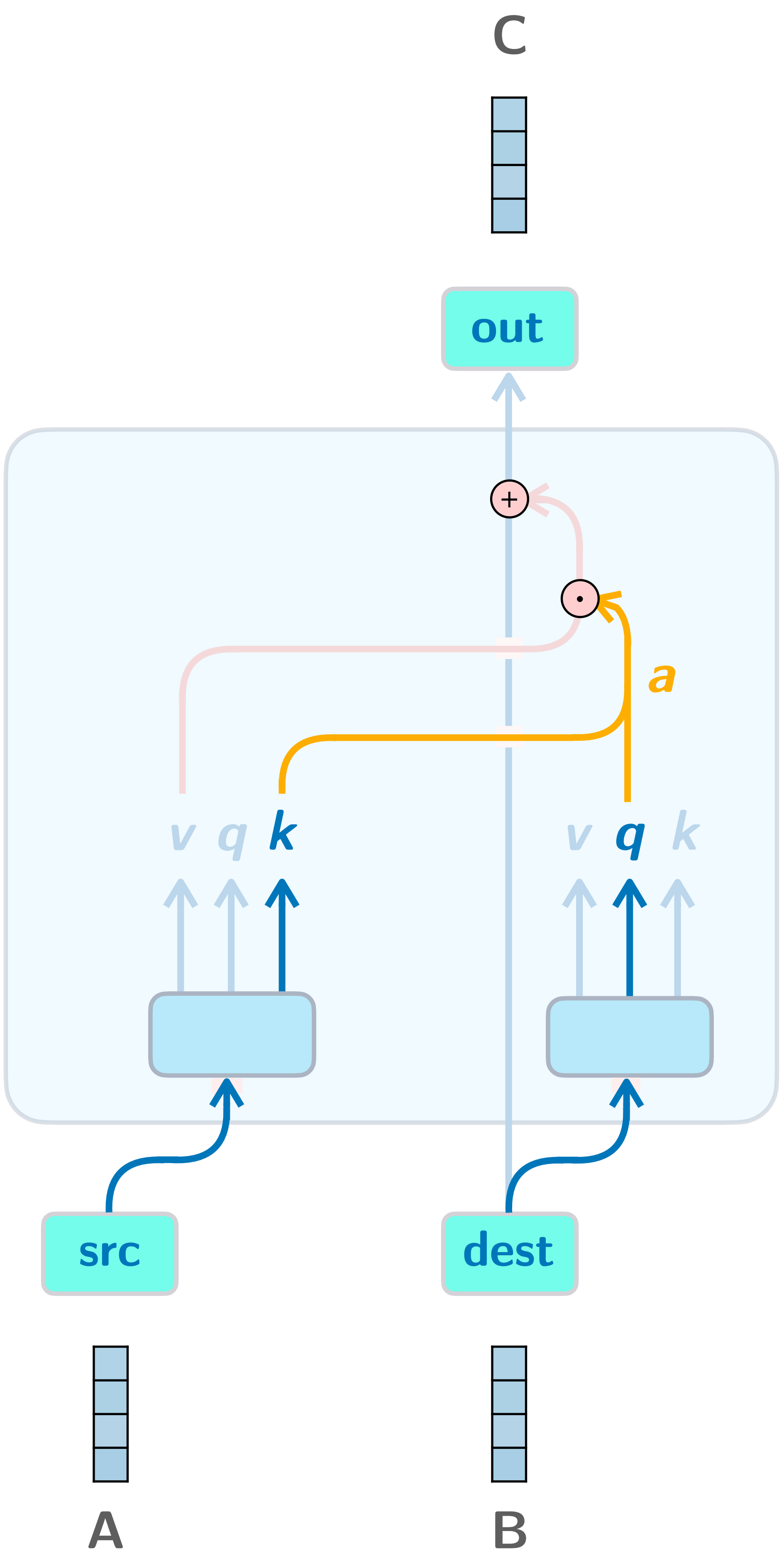


C
out
$\sum av$
v q k
v q k
src
dest
A
B

QK circuit

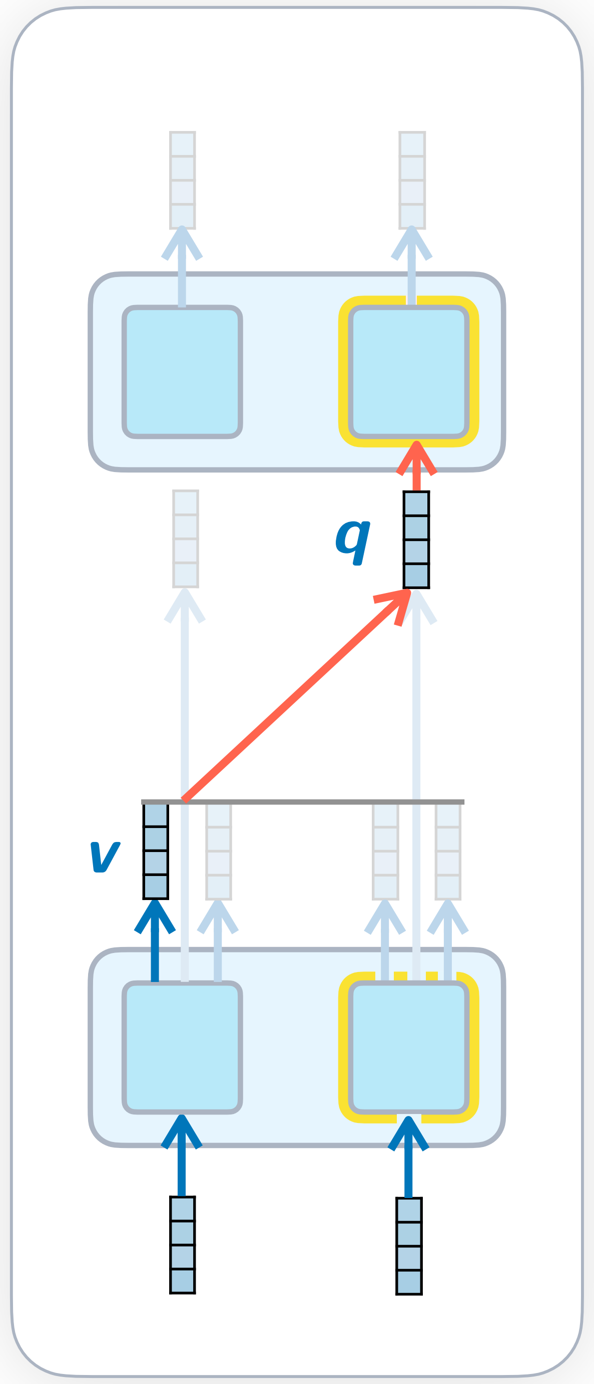
QK circuit
OV circuit

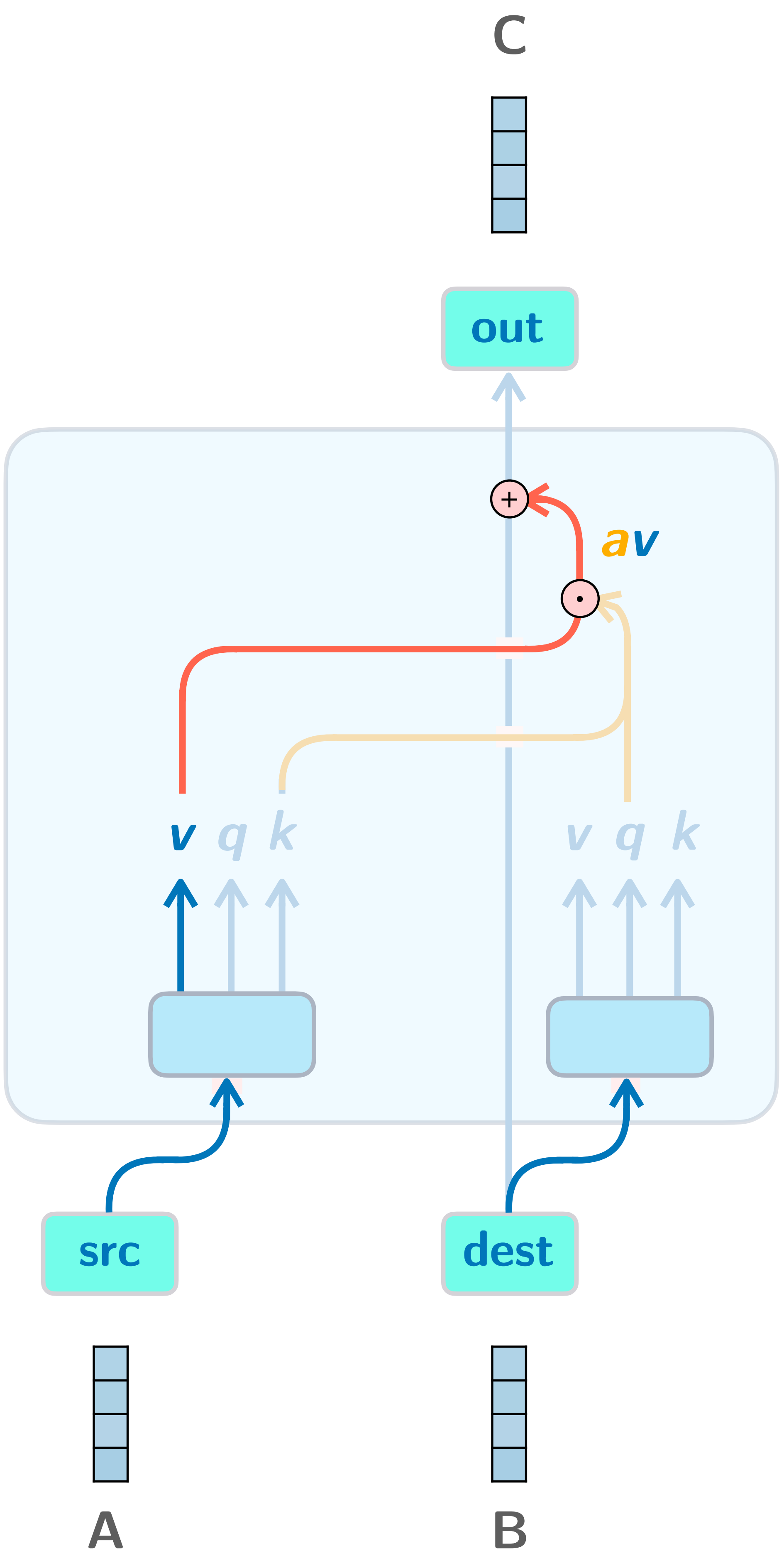




OV circuit

QK circuit
OV circuit

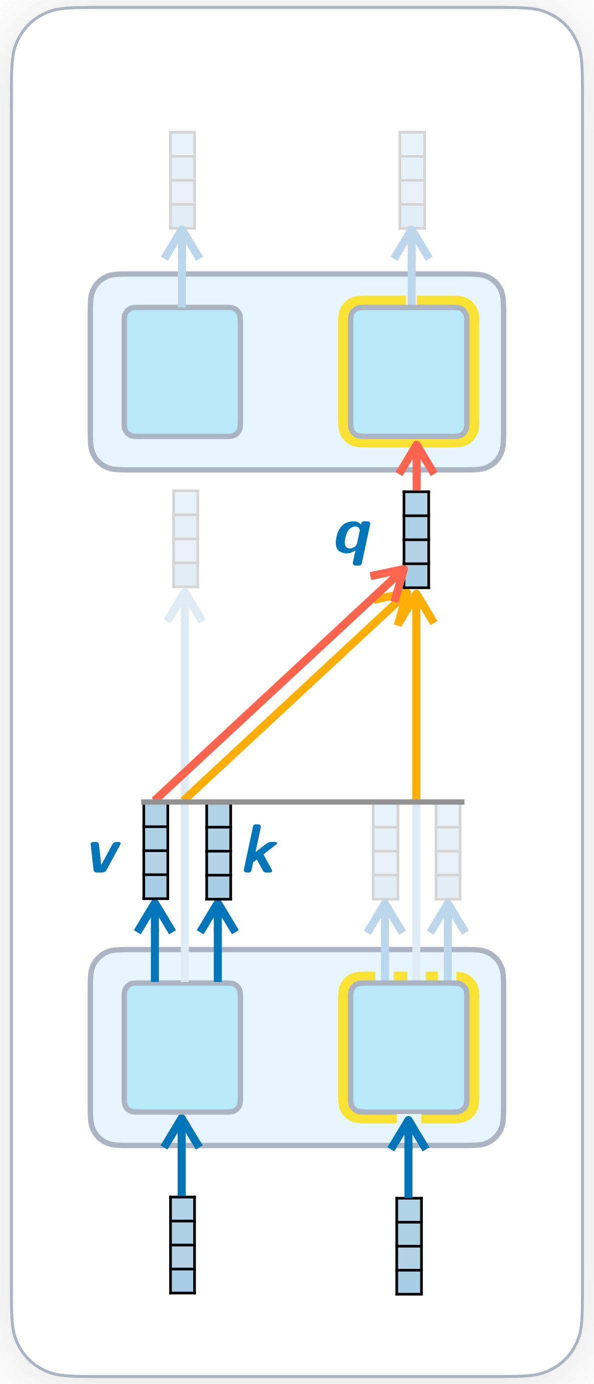


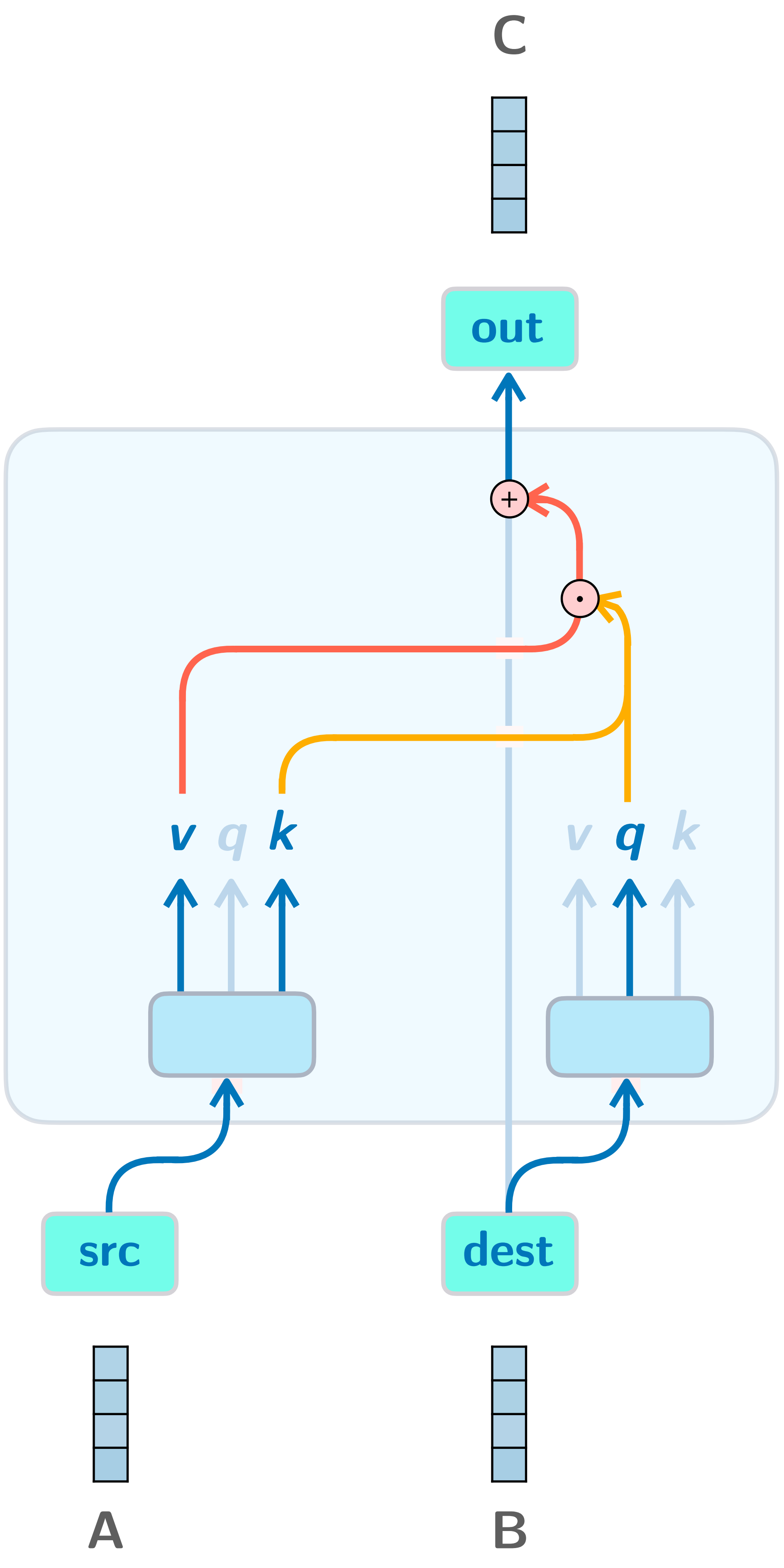


QK and OV circuits

QK circuit

OV circuit

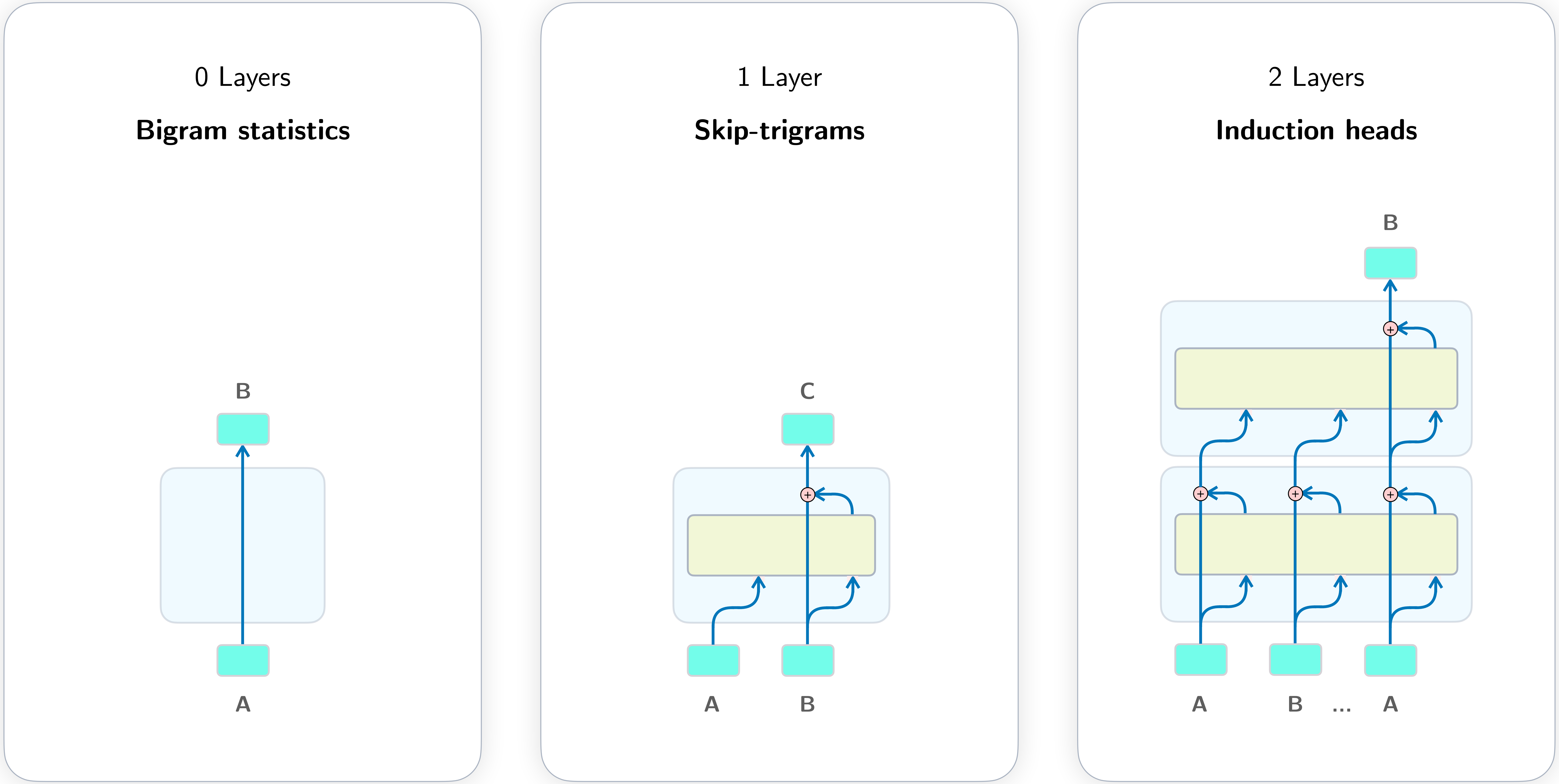




1

Induction circuit

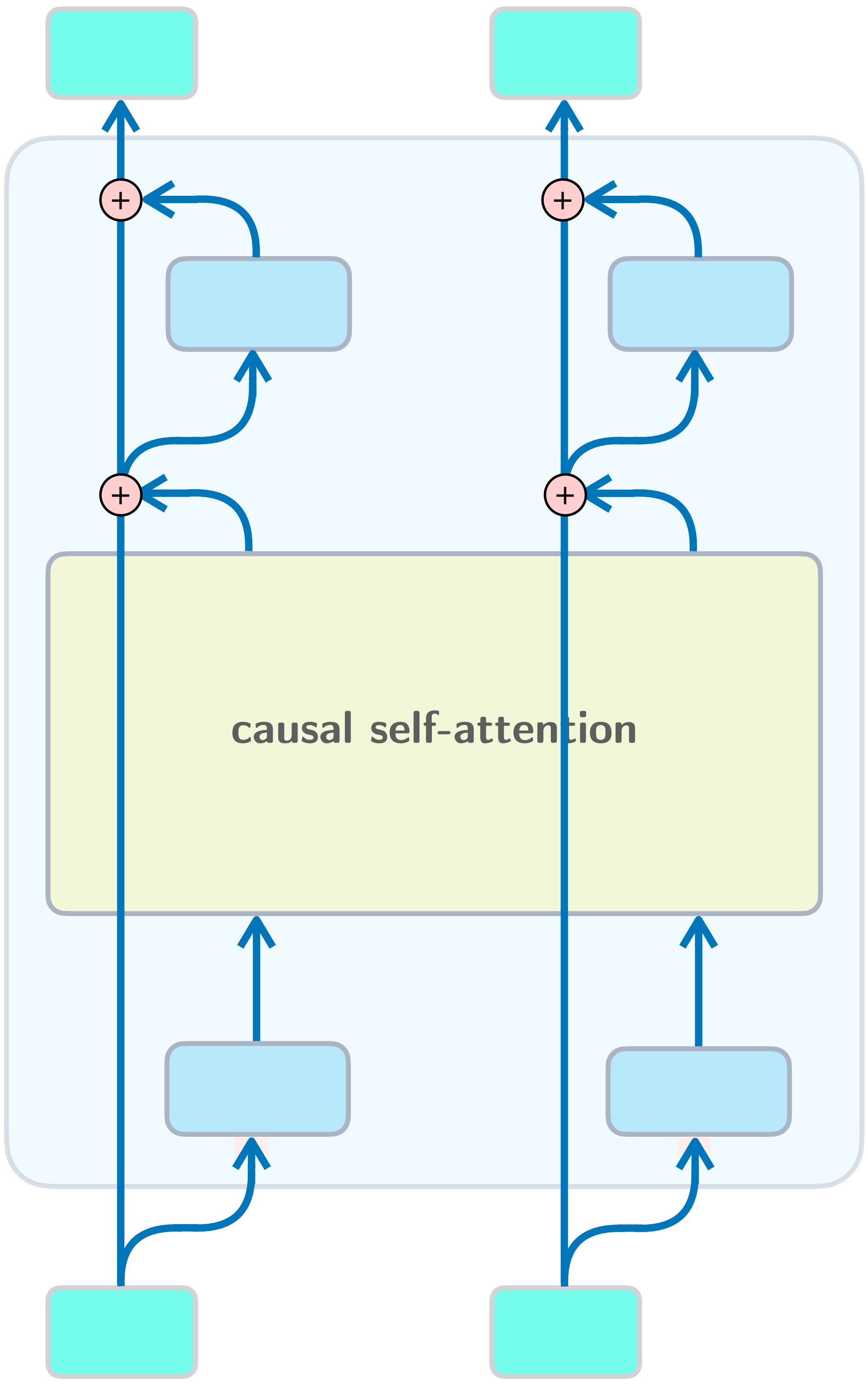
(and in-context learning)


0 Layers
Bigram statistics
B
A
1 Layer
Skip-trigrams
C
A
B
2 Layers
Induction heads
B
A
B
...
A

How to predict the next token?

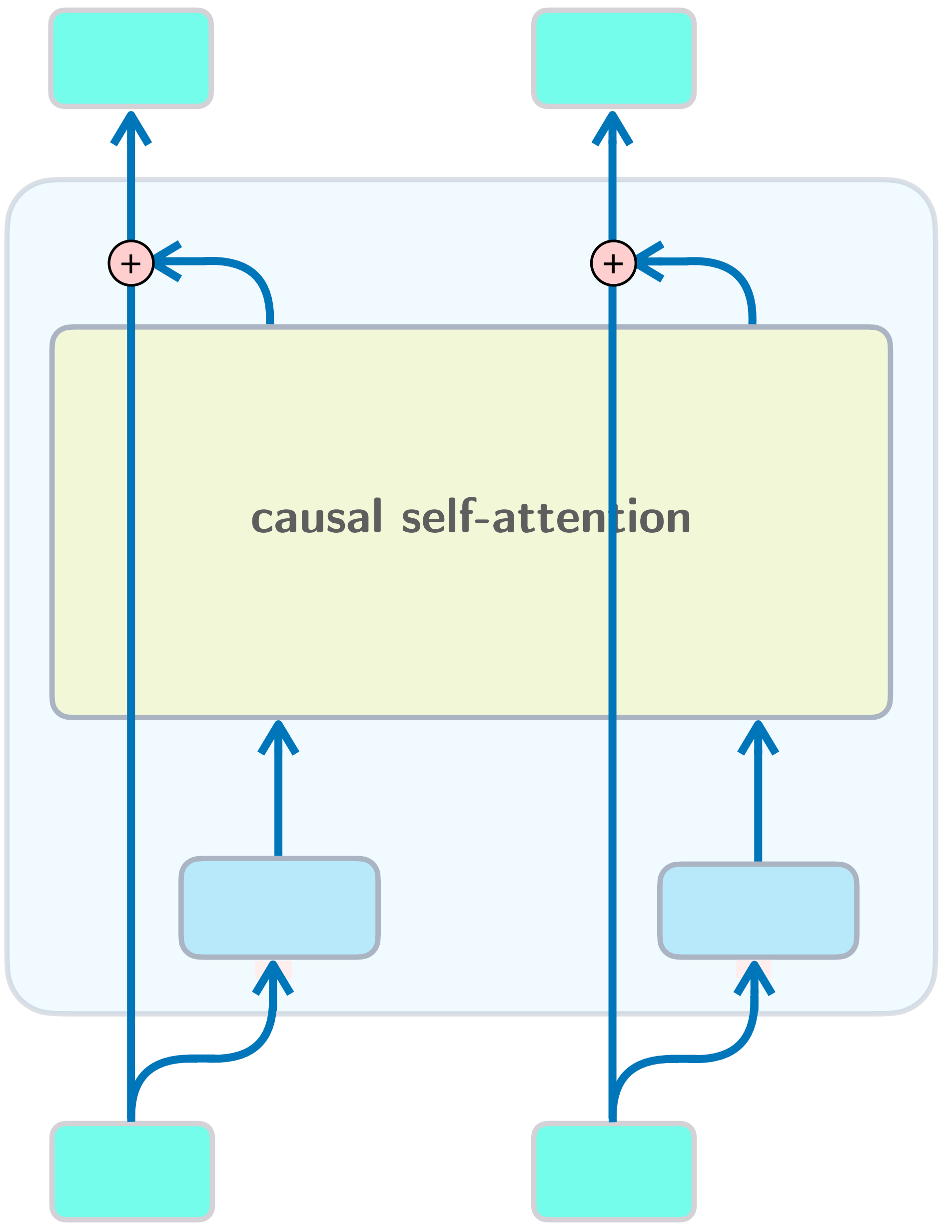
Paper: “In-context Learning and Induction Heads” by Olsson et al. (2022)

Inspired by and partially based on “Induction heads – illustrated” by TheMcDouglas



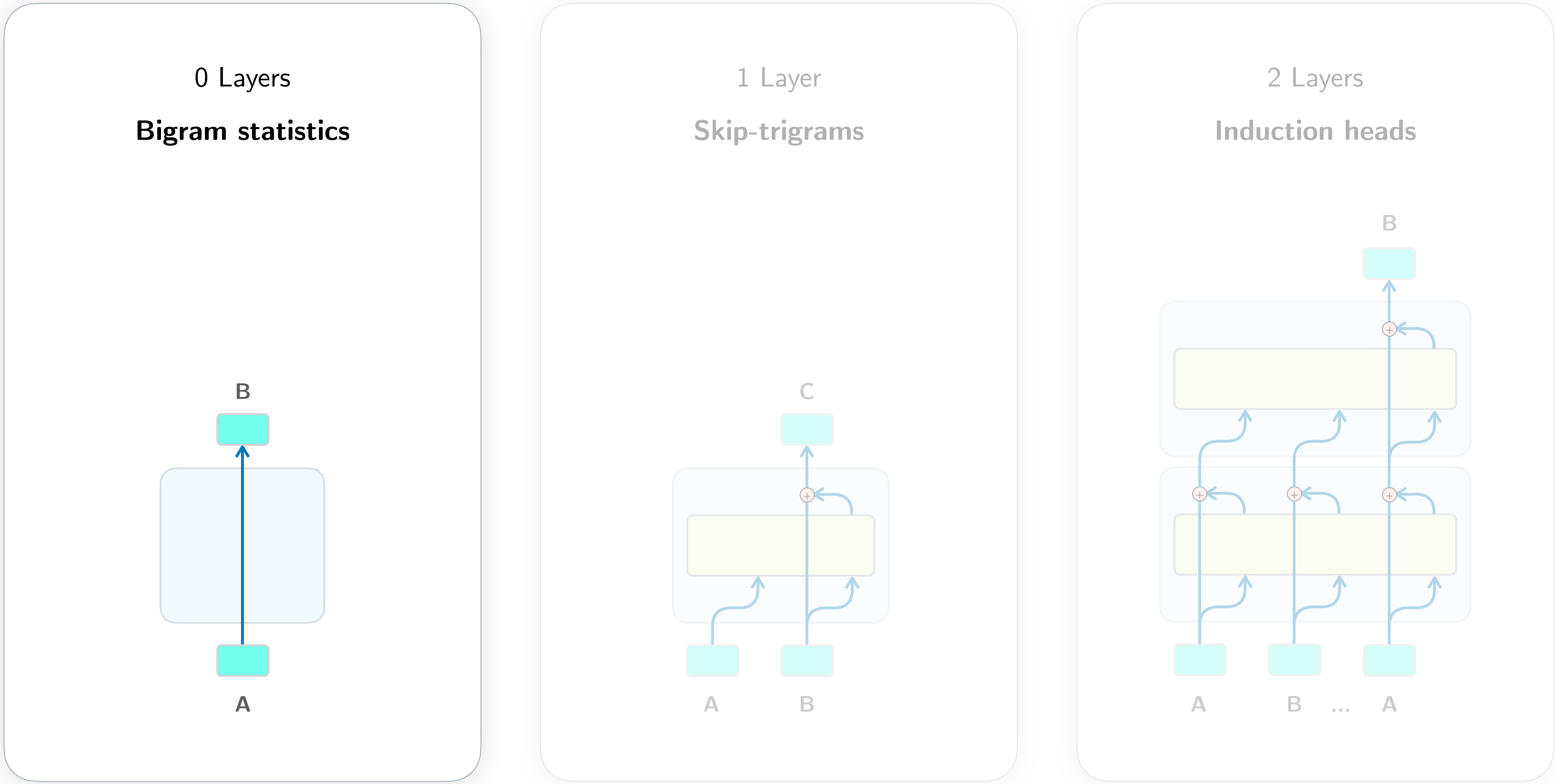
Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Let's ignore the MLPs for now! They make things more complicated.

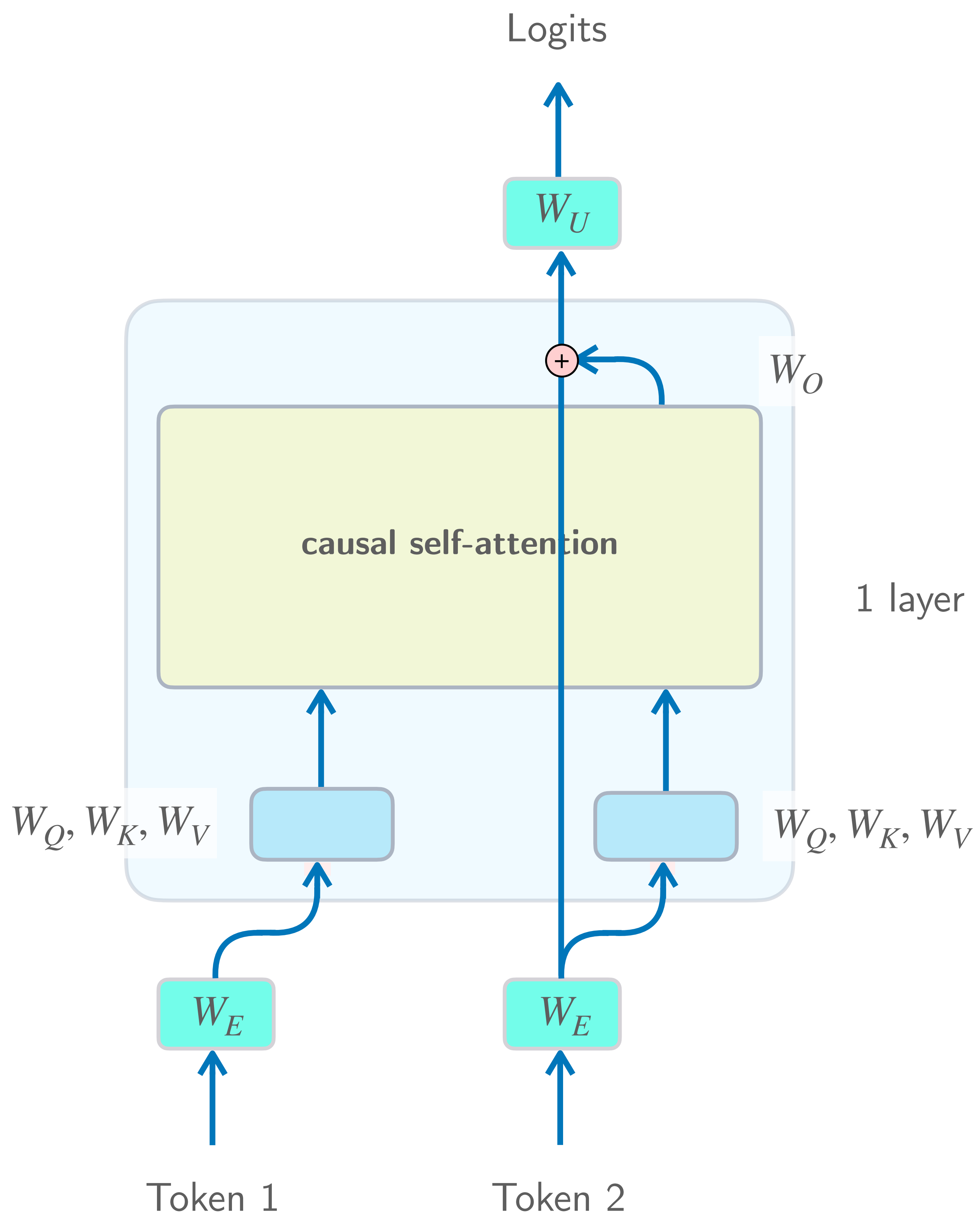
$\Rightarrow$ **attention-only transformer**


0 Layers
Bigram statistics
B
A
1 Layer
Skip-trigrams
C
A
B
2 Layers
Induction heads
B
A
B
...
A

0 Layers: Bigram statistics

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



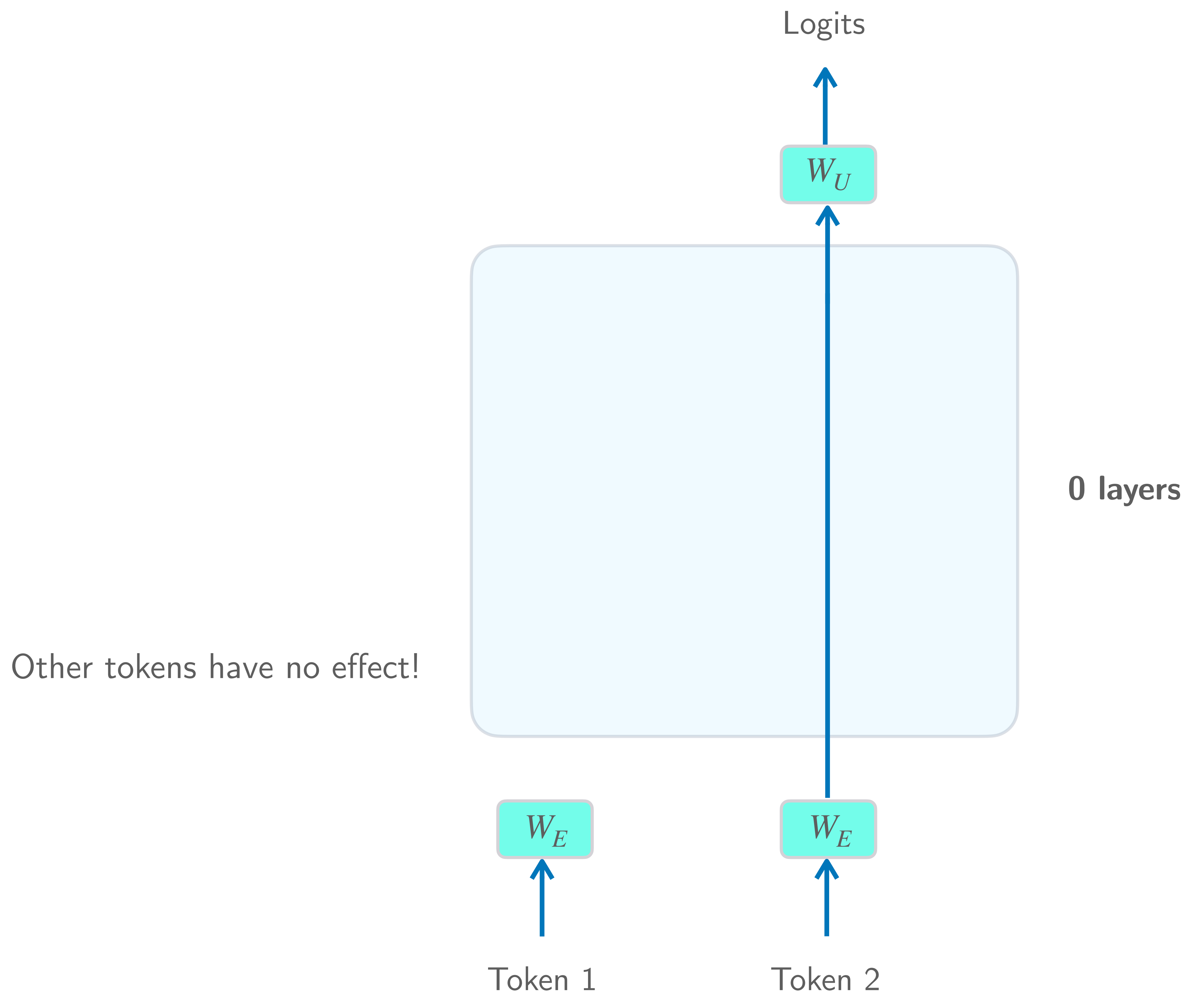
0 Layers: Bigram statistics

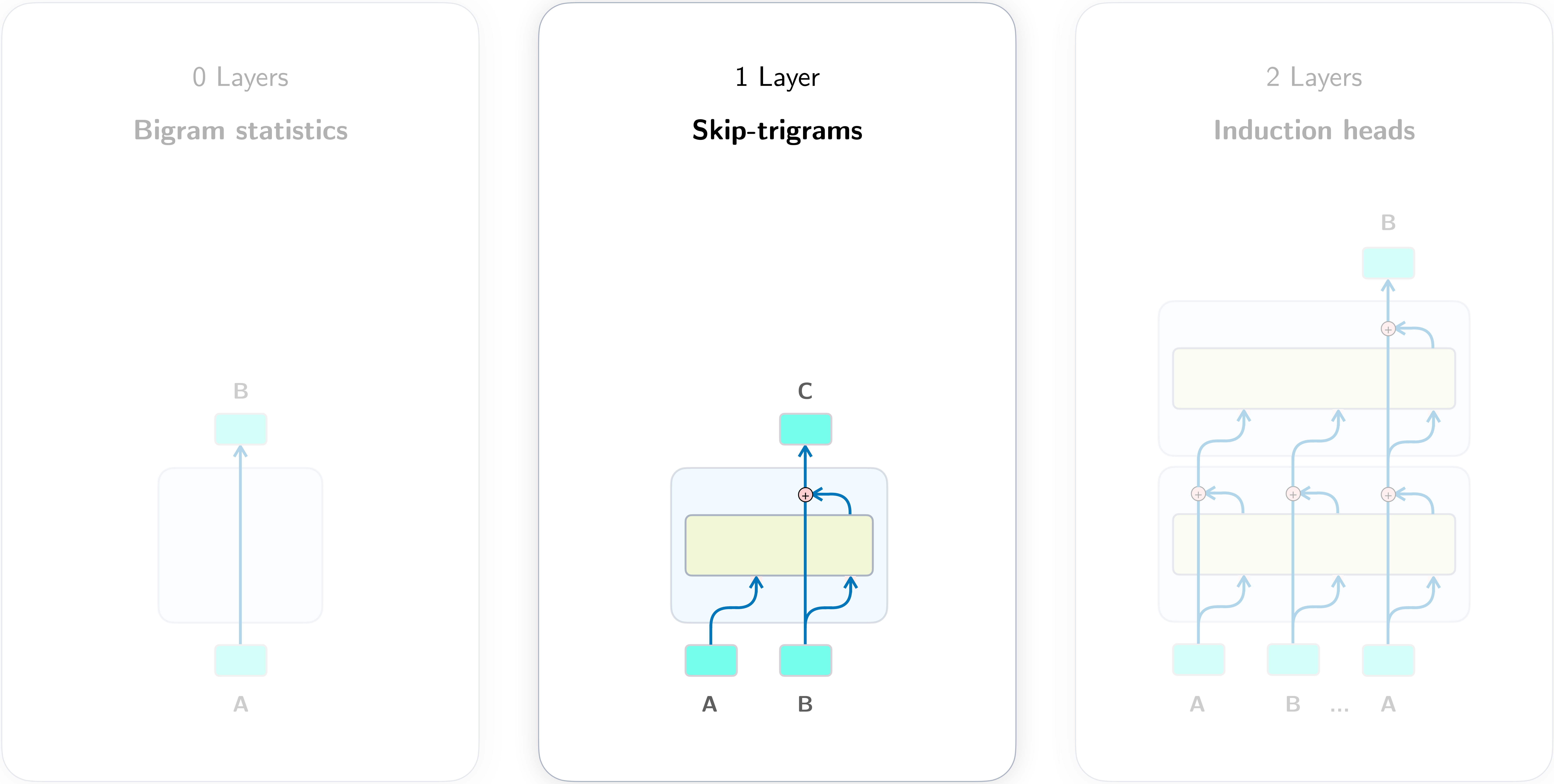
Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

Bigram statistics lookup table

$$f(\mathbf{x}) = \boxed{W_U W_E}\,\mathbf{x}$$



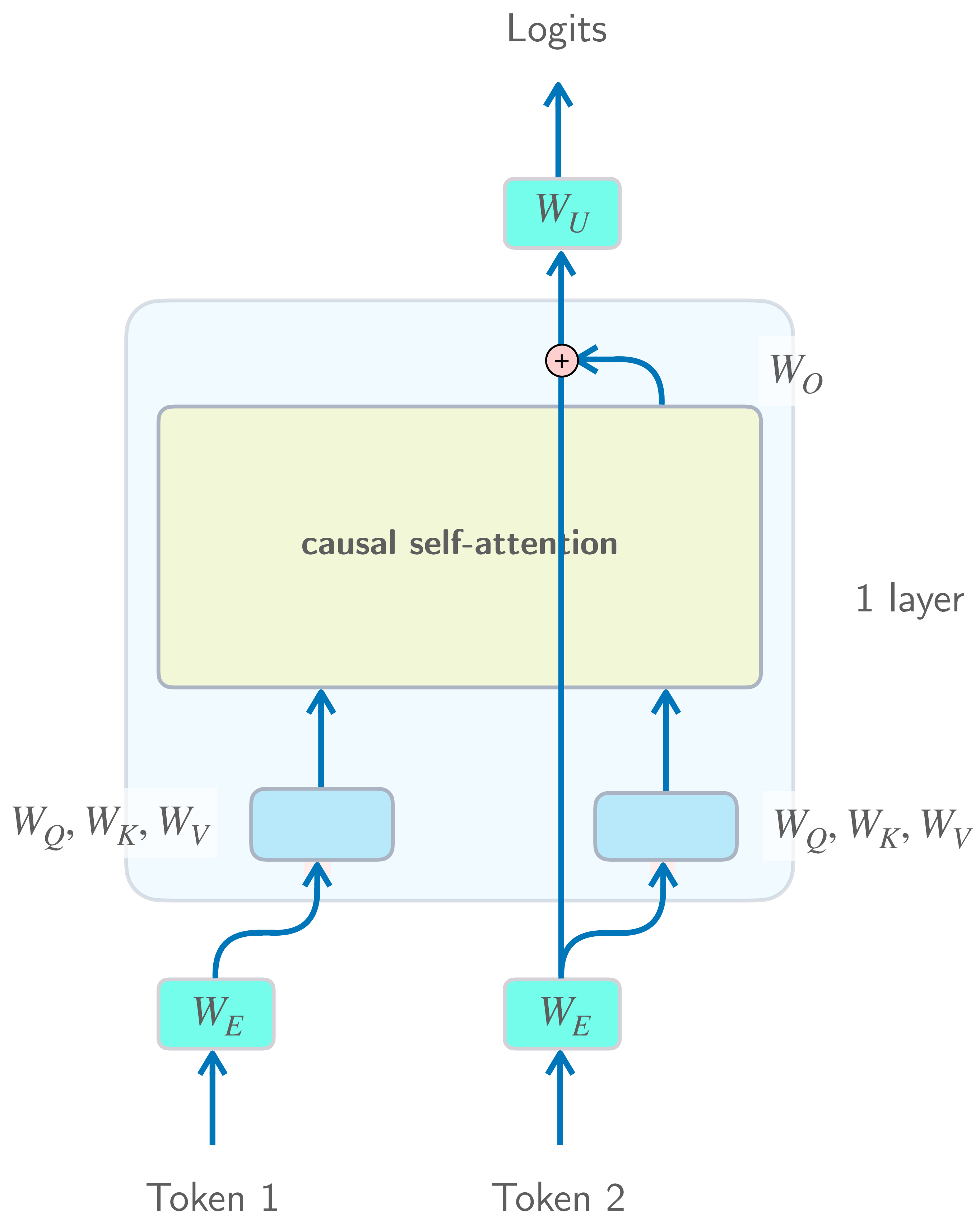


0 Layers
Bigram statistics
B
A
1 Layer
Skip-trigrams
C
A
B
2 Layers
Induction heads
B
A
B
...
A

1 Layer: Skip-trigrams

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

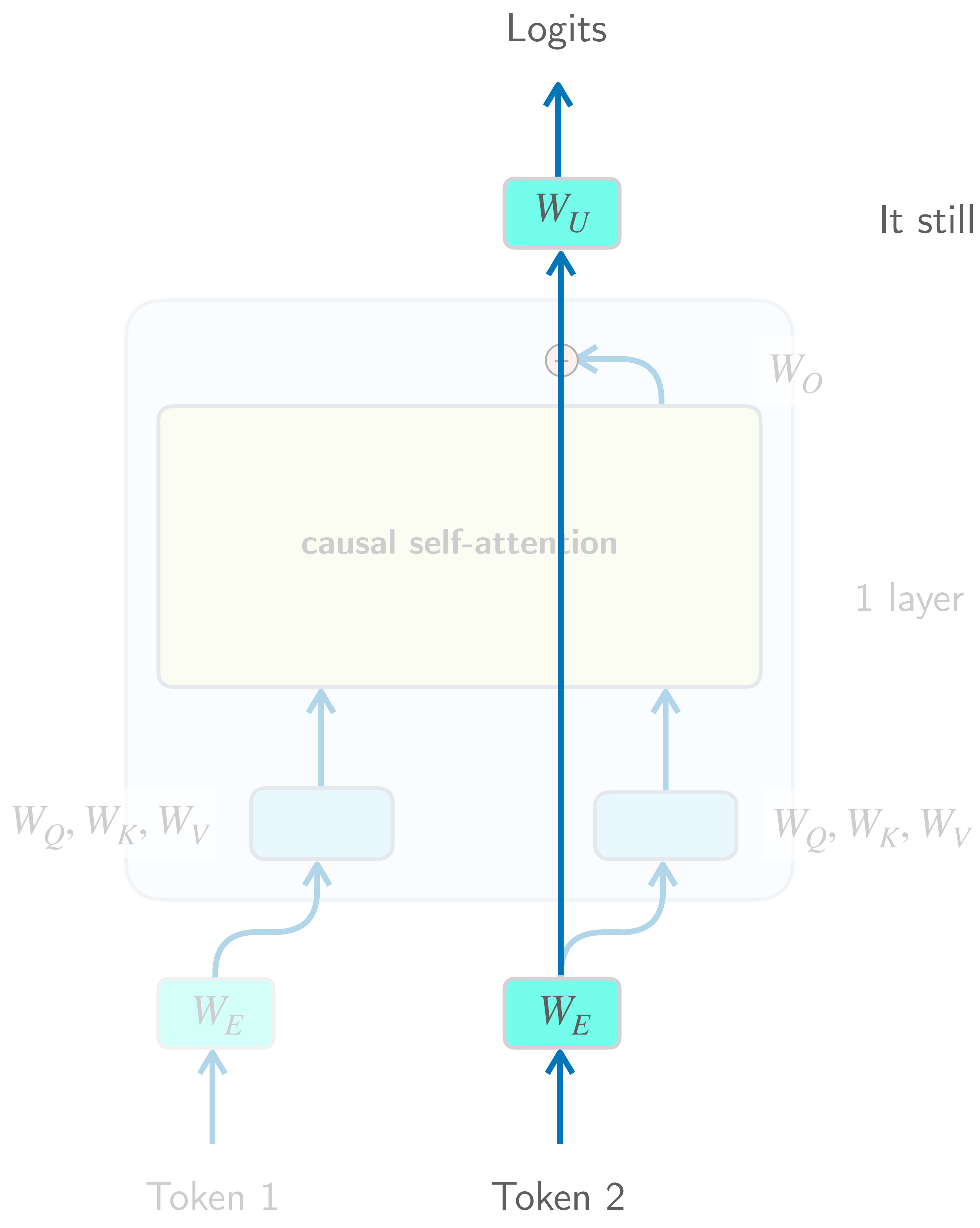
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



1 Layer: Skip-trigrams

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

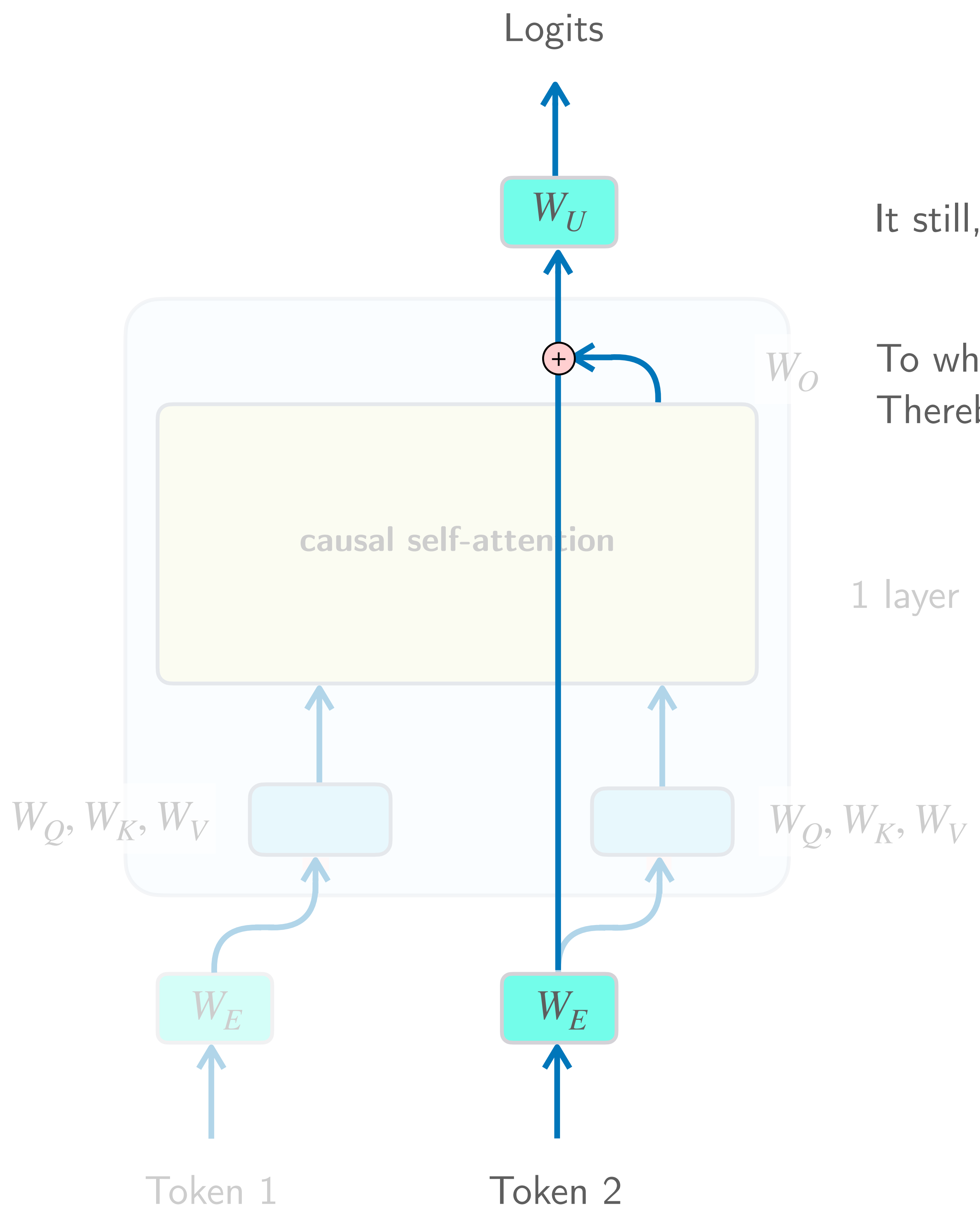


It still, kind of, does bigram statistics

1 Layer: Skip-trigrams

Paper: “In-context Learning and Induction Heads” by Olsson et al. (2022)

Inspired by and partially based on “Induction heads – illustrated” by TheMcDouglas

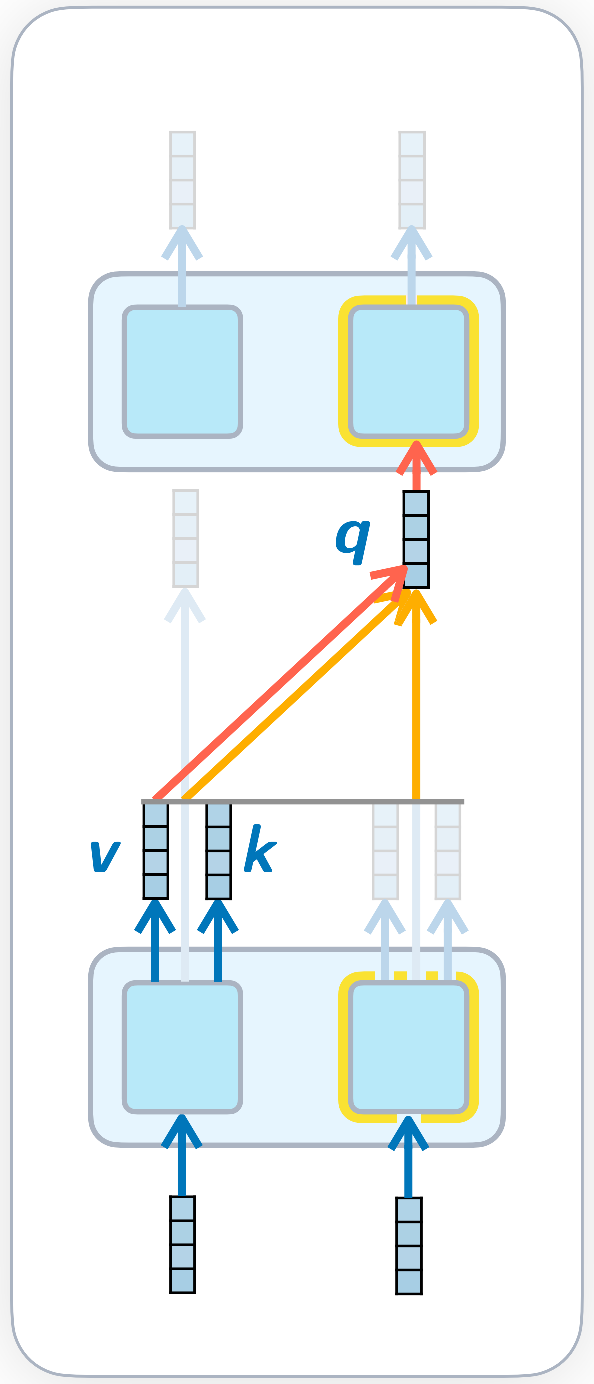


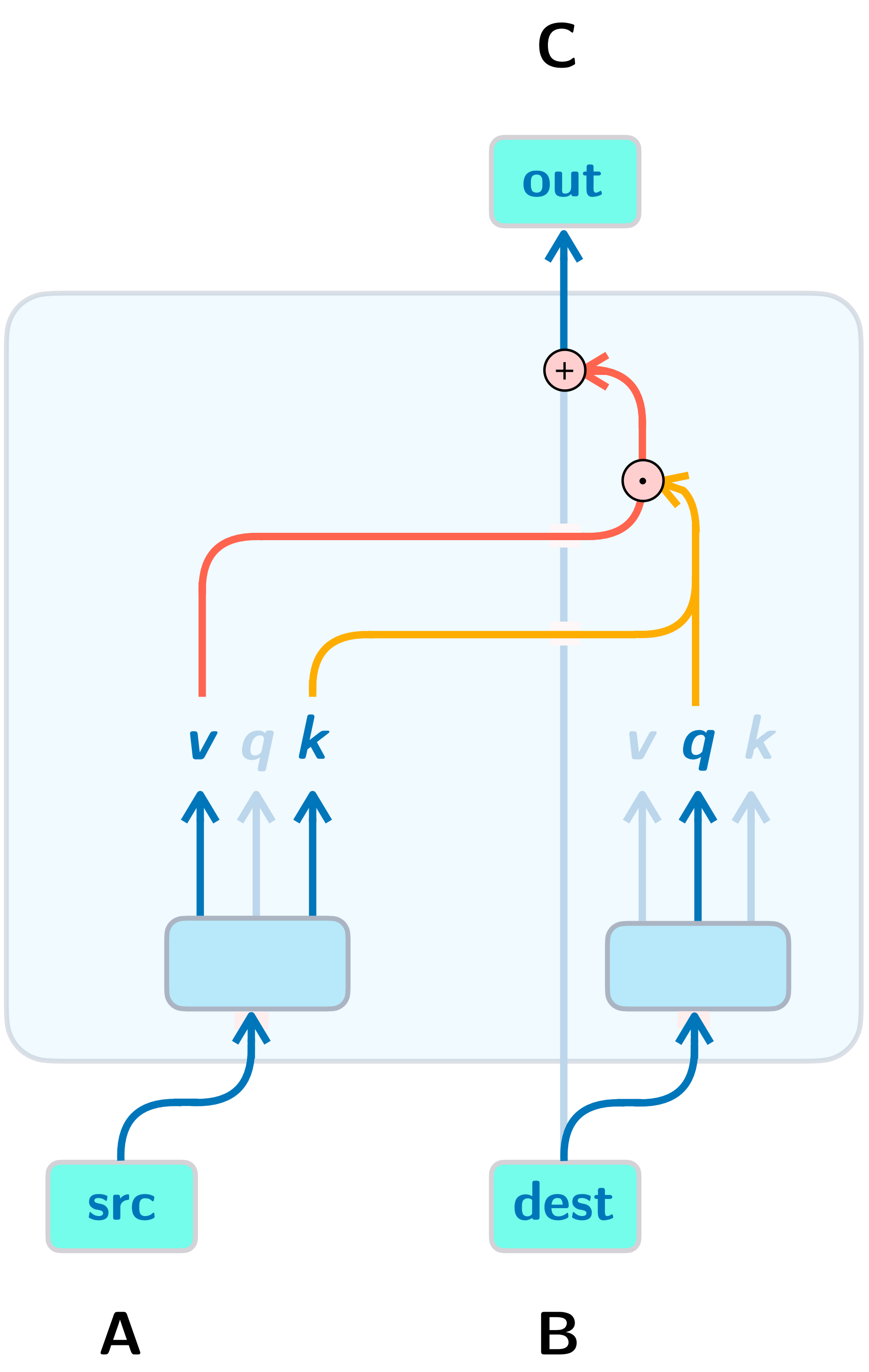
It still, kind of, does bigram statistics

To which we now add the attention result.
Thereby conditioning the logits on all the previous tokens.

Skip-trigram: [A][B] → [C]

[src][dest] → [out]



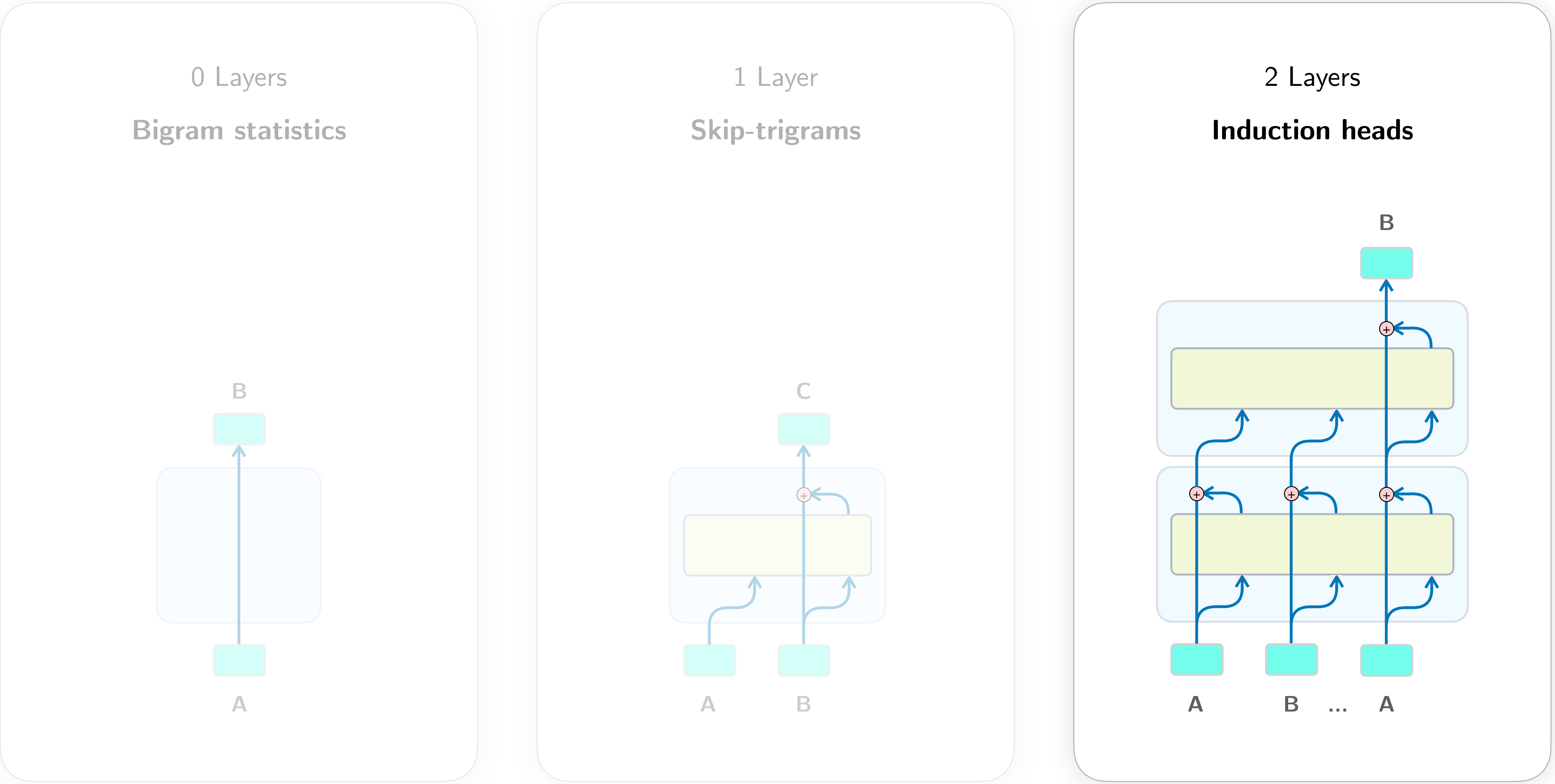


Increase logit of token C

(...)

(...)

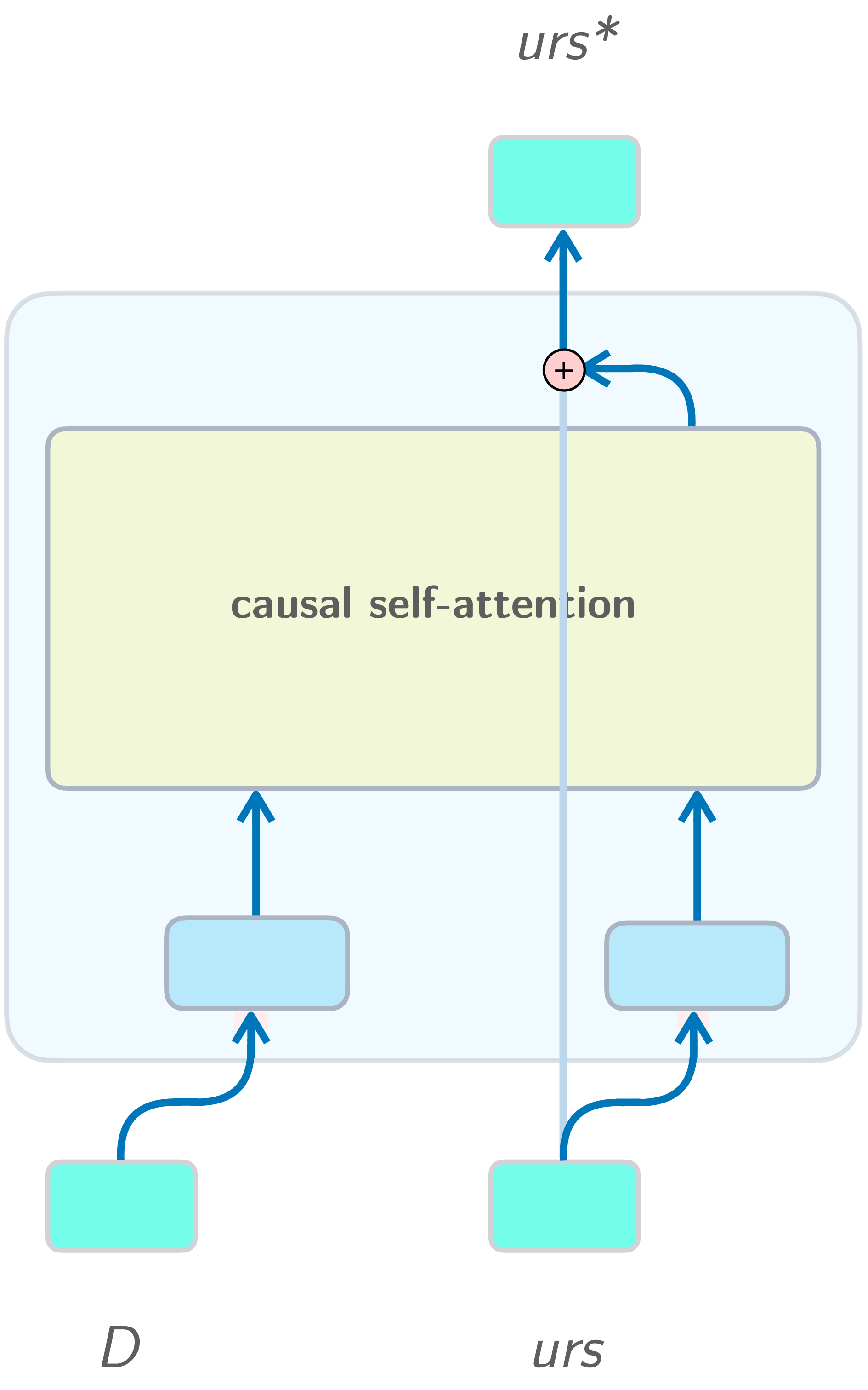
(...)


0 Layers
Bigram statistics
B
A
1 Layer
Skip-trigrams
C
A
B
2 Layers
Induction heads
B
A
B
...
A

2 Layers: Induction heads

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

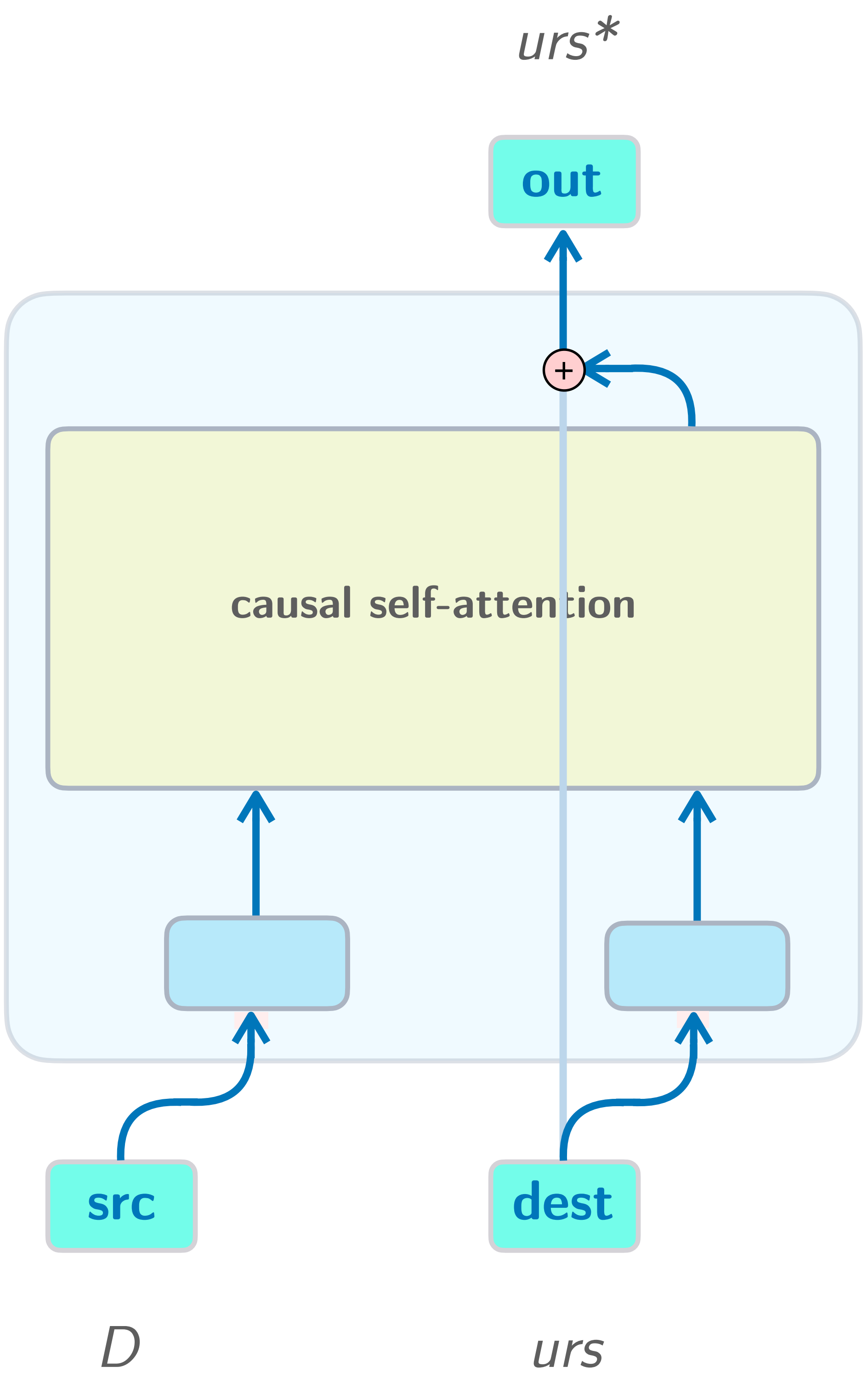
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



2 Layers: Induction heads

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

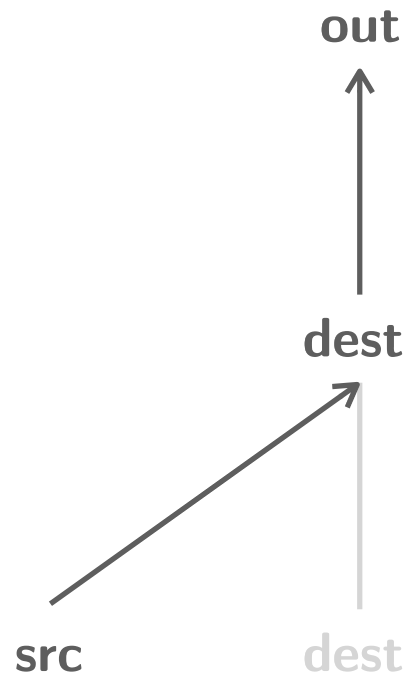
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

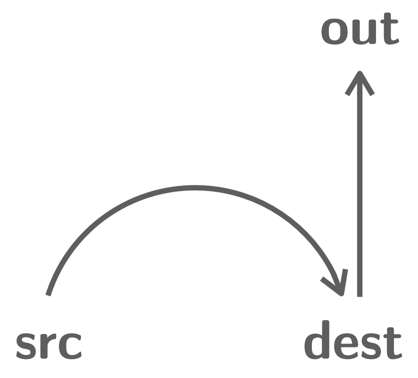


source of attention information

destination of attention information

output of attention information

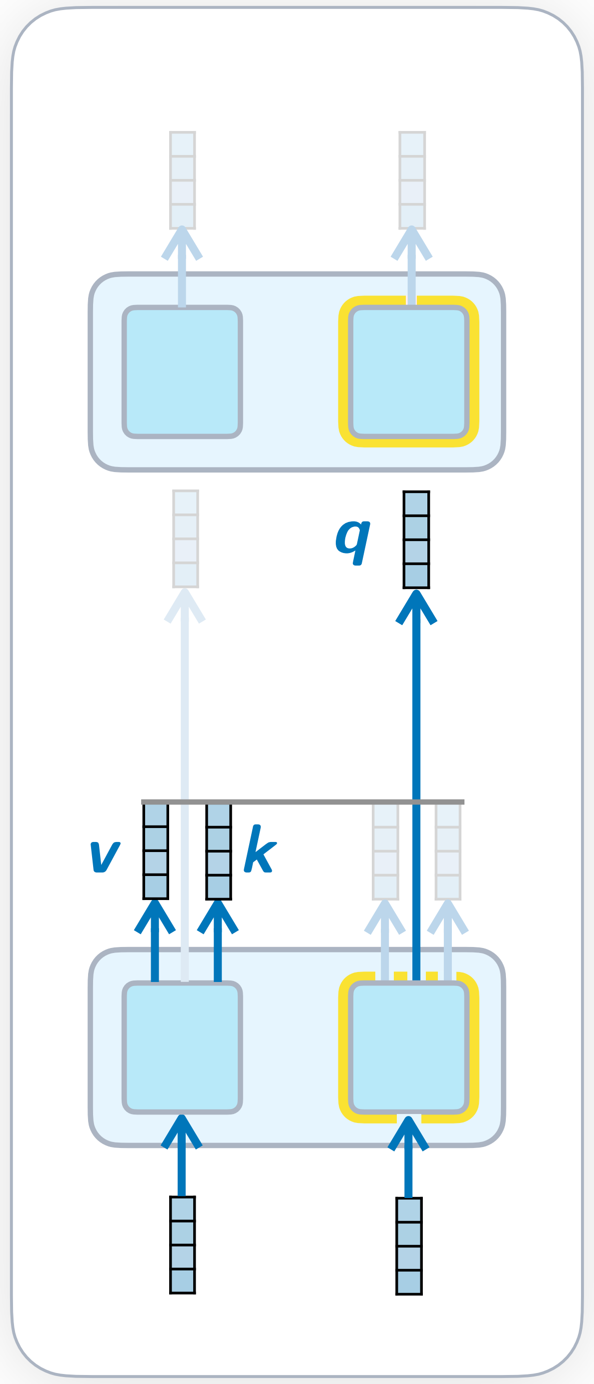


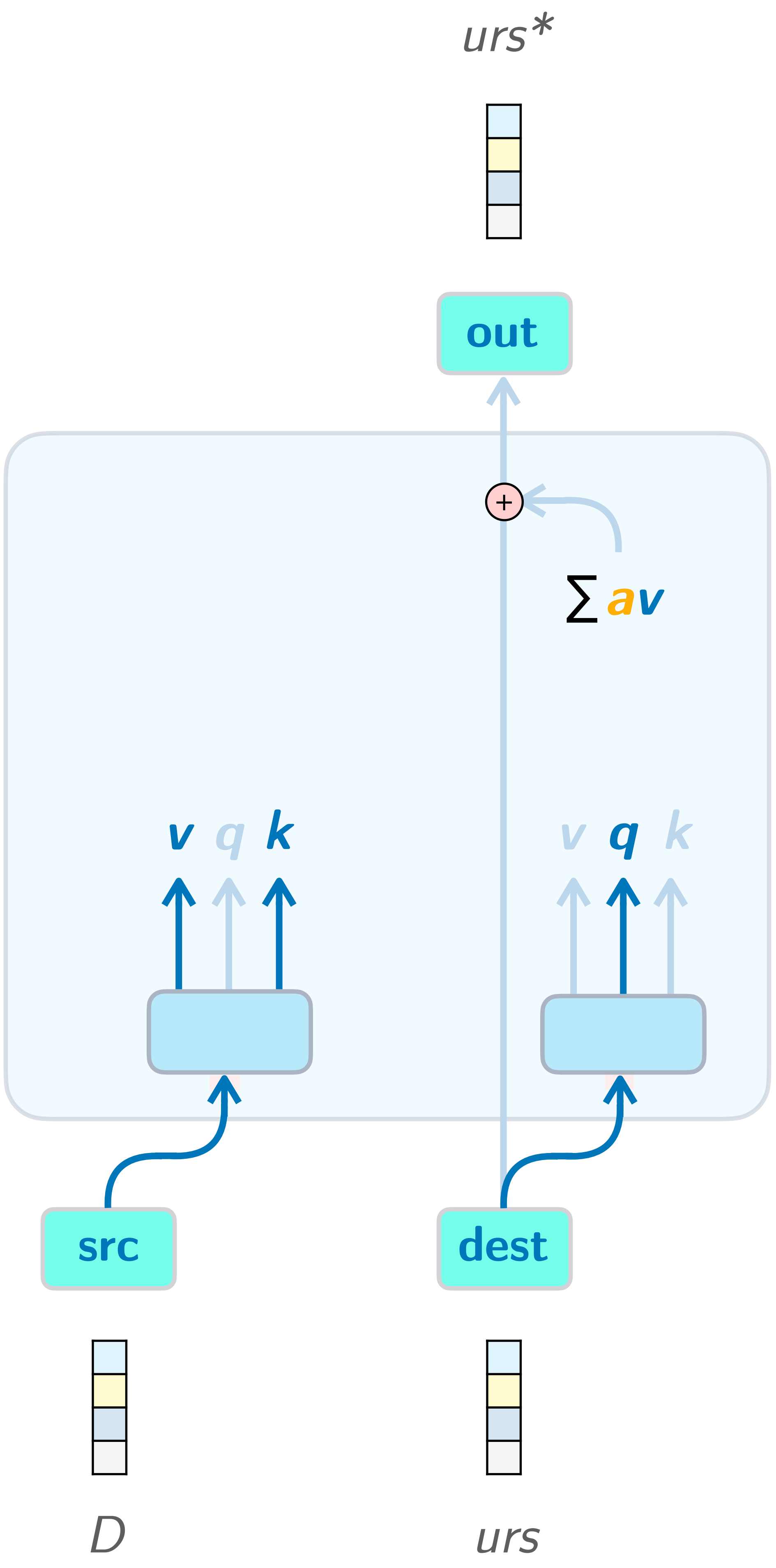


2 Layers: Induction heads

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas





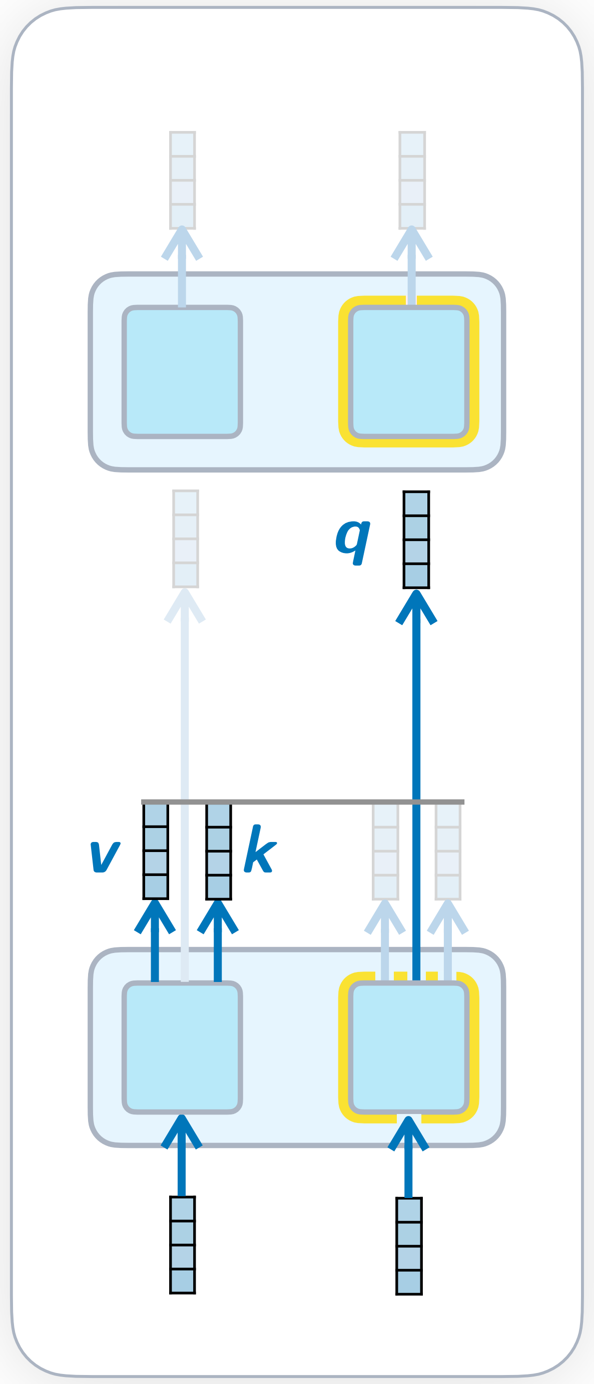
Embedding subspaces:

current token
previous token
next token
position

2 Layers: Induction heads

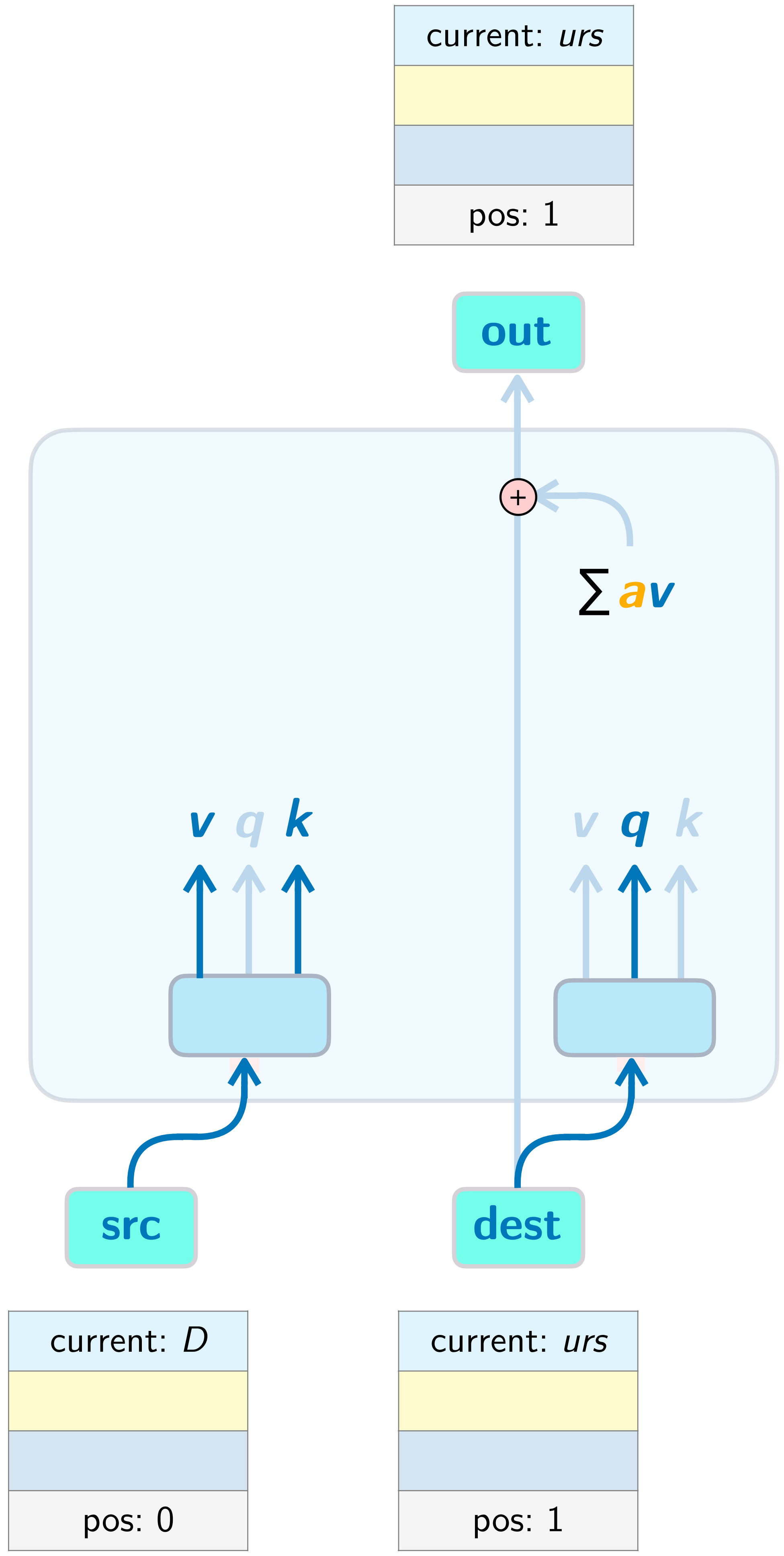
Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

current token
previous token
next token
position



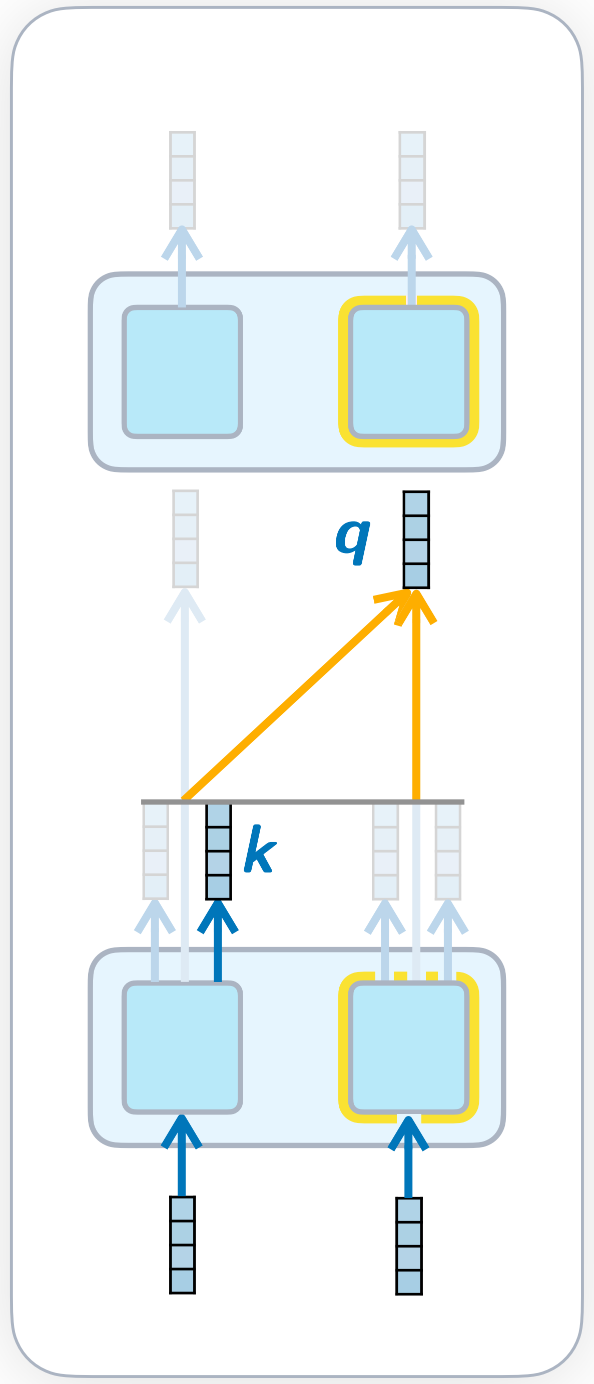
2 Layers: Induction heads

QK circuit

OV circuit

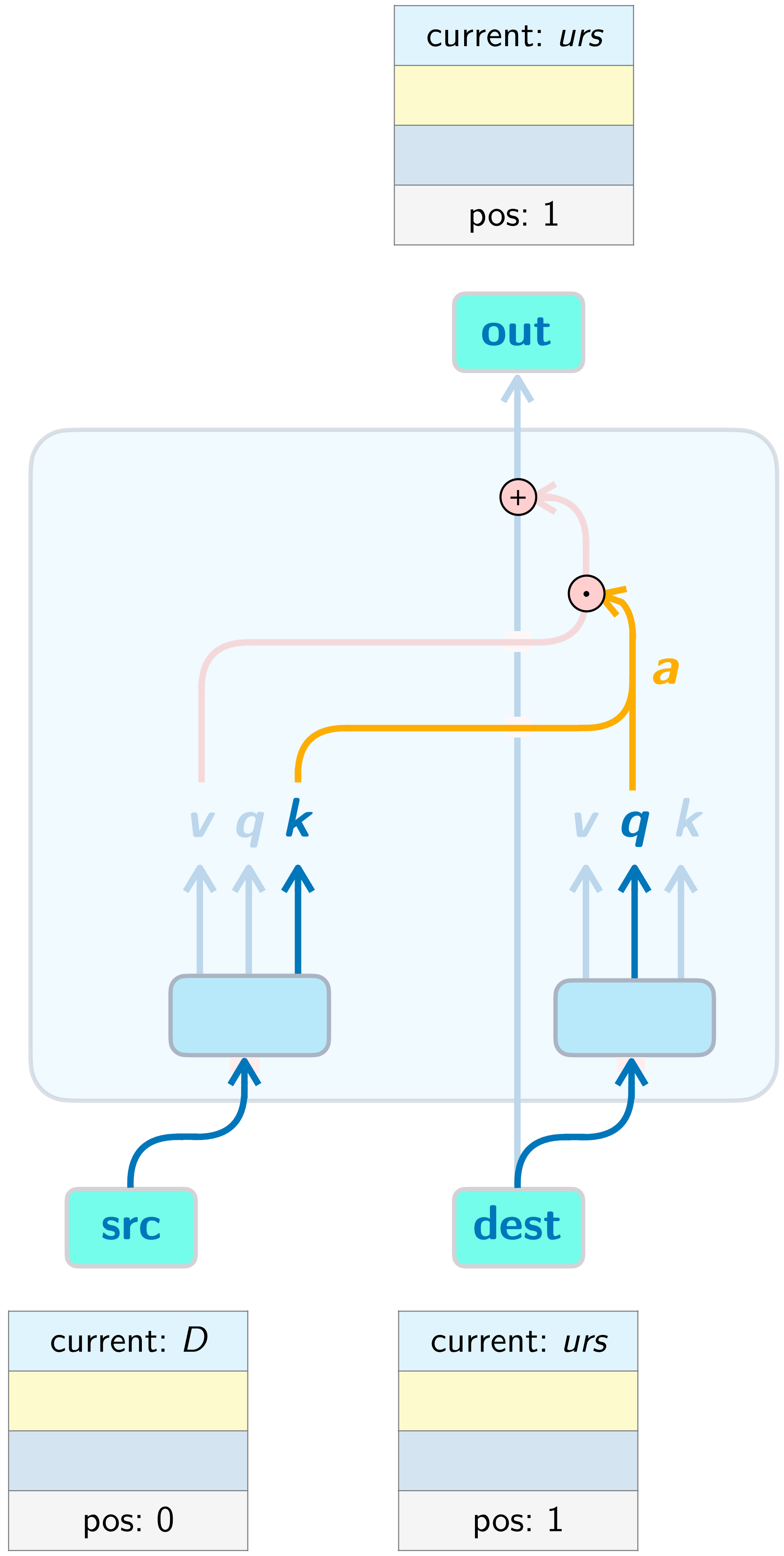
Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

current token
previous token
next token
position



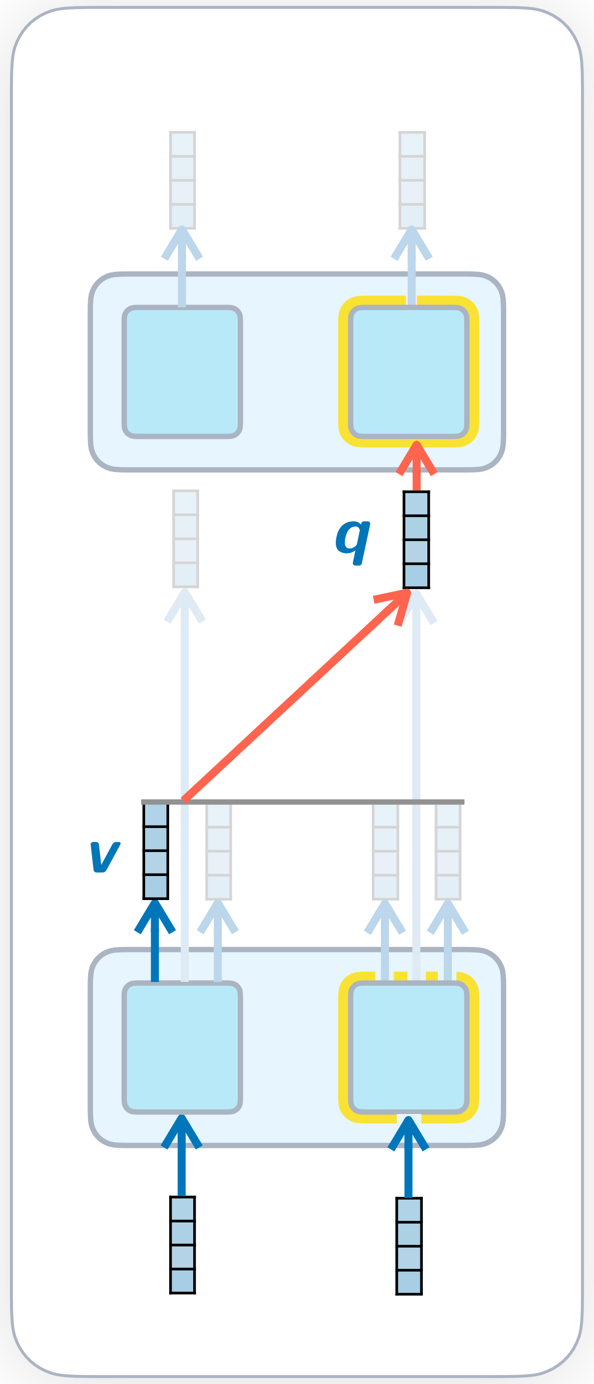
Query for token with pos=0 (it knows that current token has pos=1, so it knows the preceding token is at pos=0)

All other tokens have low attention score.

2 Layers: Induction heads

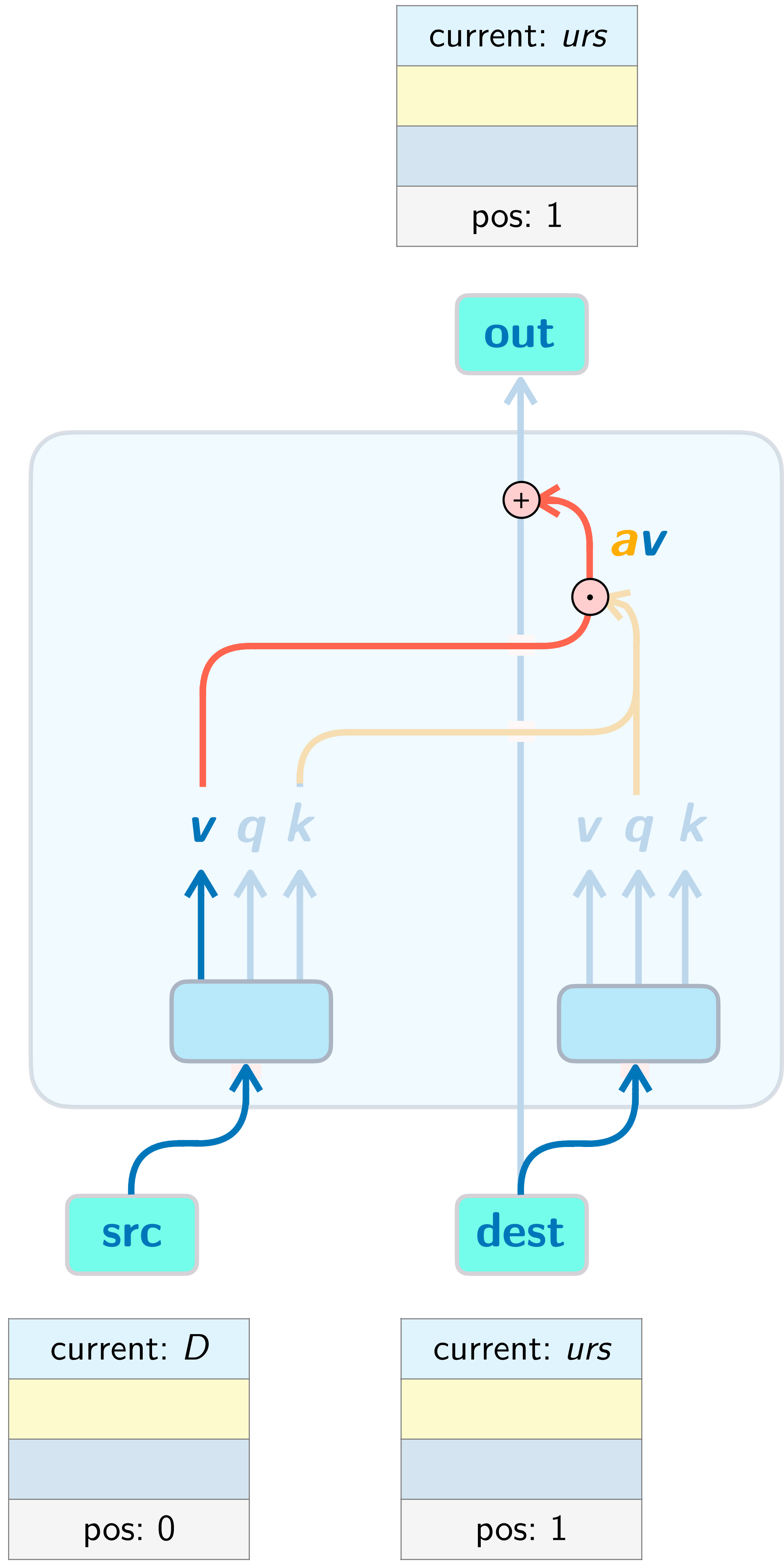
QK circuit
OV circuit

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

current token
previous token
next token
position



The value for "current: D" would be something like "prev: D" such that it can be written into the prev-token subspace of the "urs" residual stream

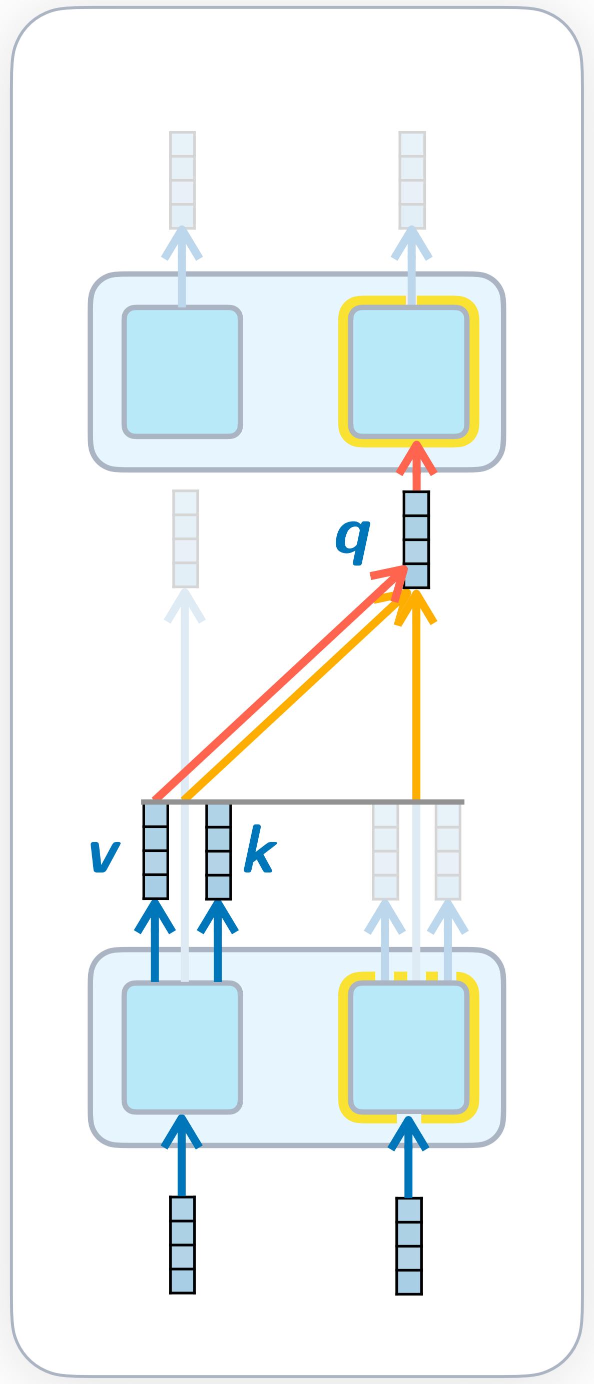
Query for token with pos=0 (it knows that current token has pos=1, so it knows the preceding token is at pos=0)

All other tokens have low attention score.

2 Layers: Induction heads

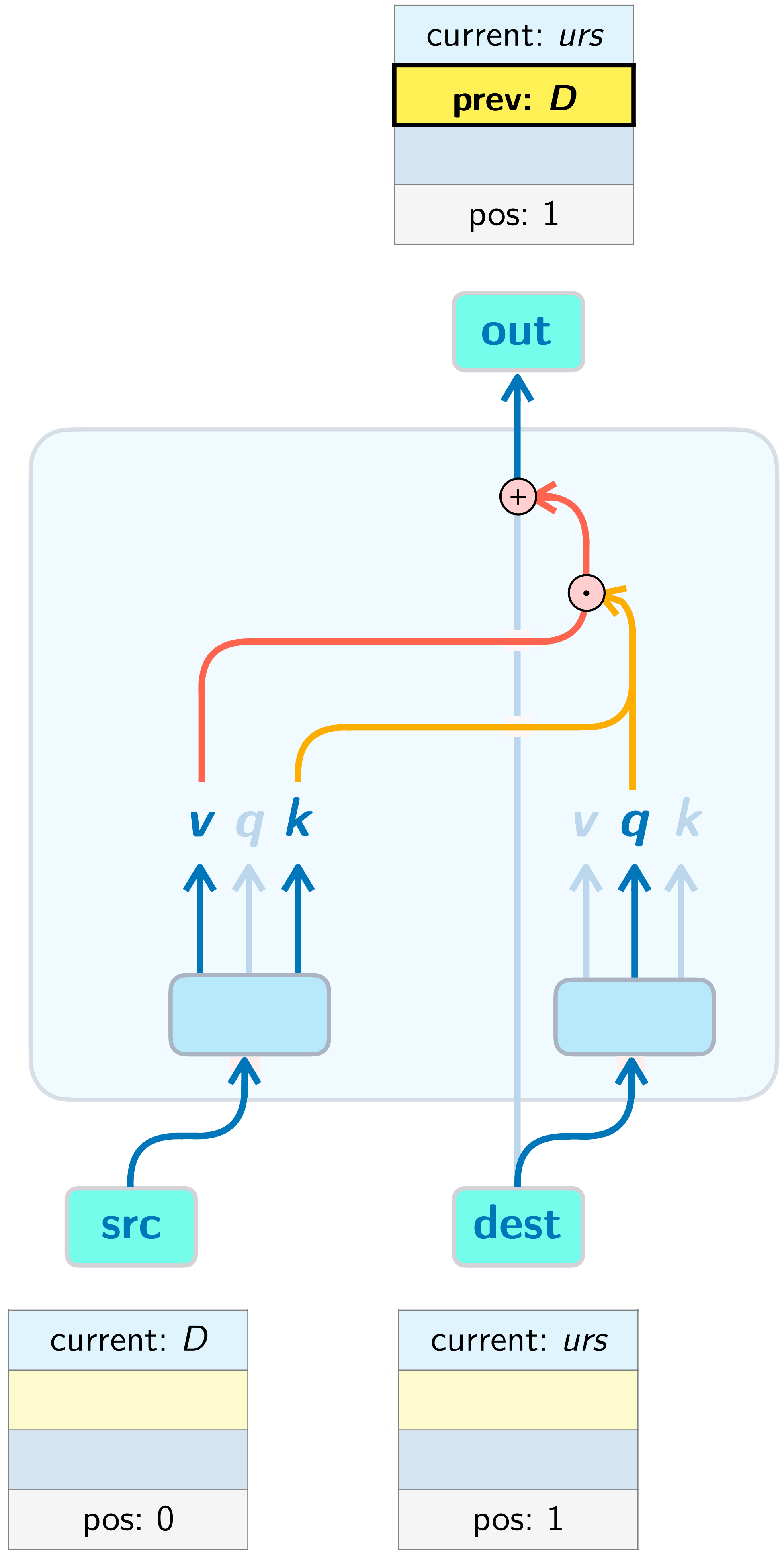
QK circuit
OV circuit

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

current token
previous token
next token
position



The value for "current: D" would be something like "prev: D" such that it can be written into the prev-token subspace of the "urs" residual stream

Query for token with pos=0 (it knows that current token has pos=1, so it knows the preceding token is at pos=0)

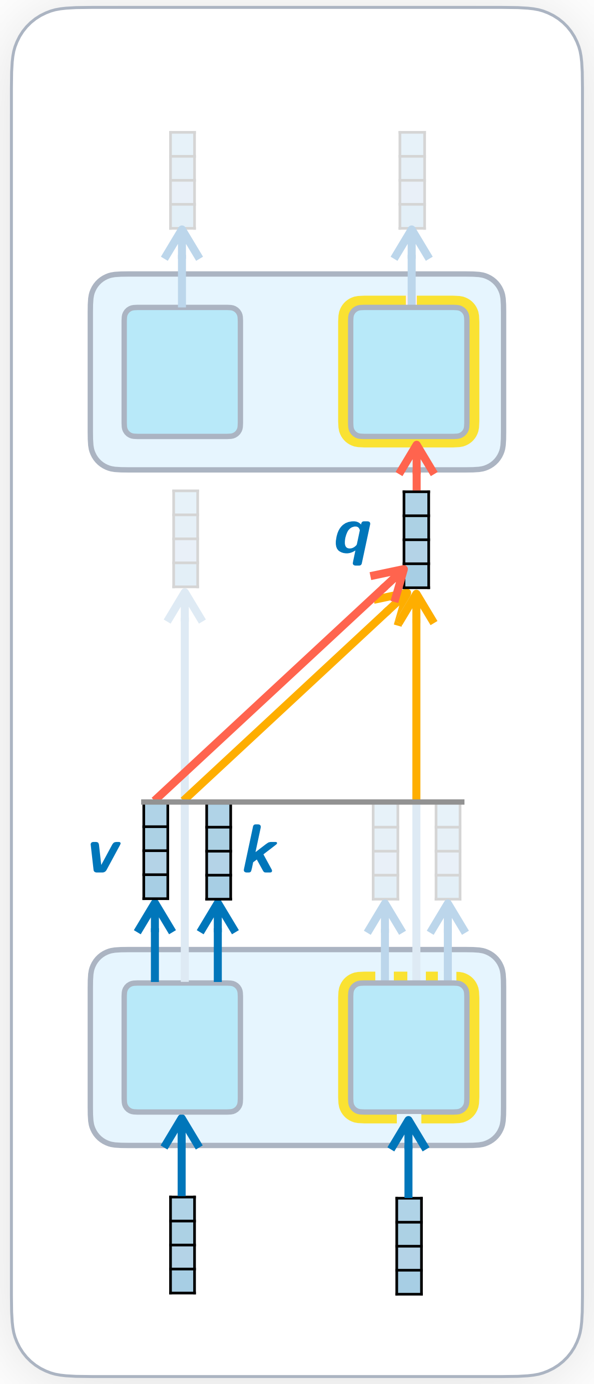
All other tokens have low attention score.

emphasize that it’s not learning the rule (A, B) and that’s why when it encounters A it predicts B,
rather, it learns a general method that when it encounters an arbitrary symbol A,
it looks for past occurrences of symbols that had A as its predecessor, i.e. symbol B, such that it can now predict that symbol B
once it has learned this method, it can apply it to any symbols, even ones it has never seen before; even totally random ones

2 Layers: Induction heads

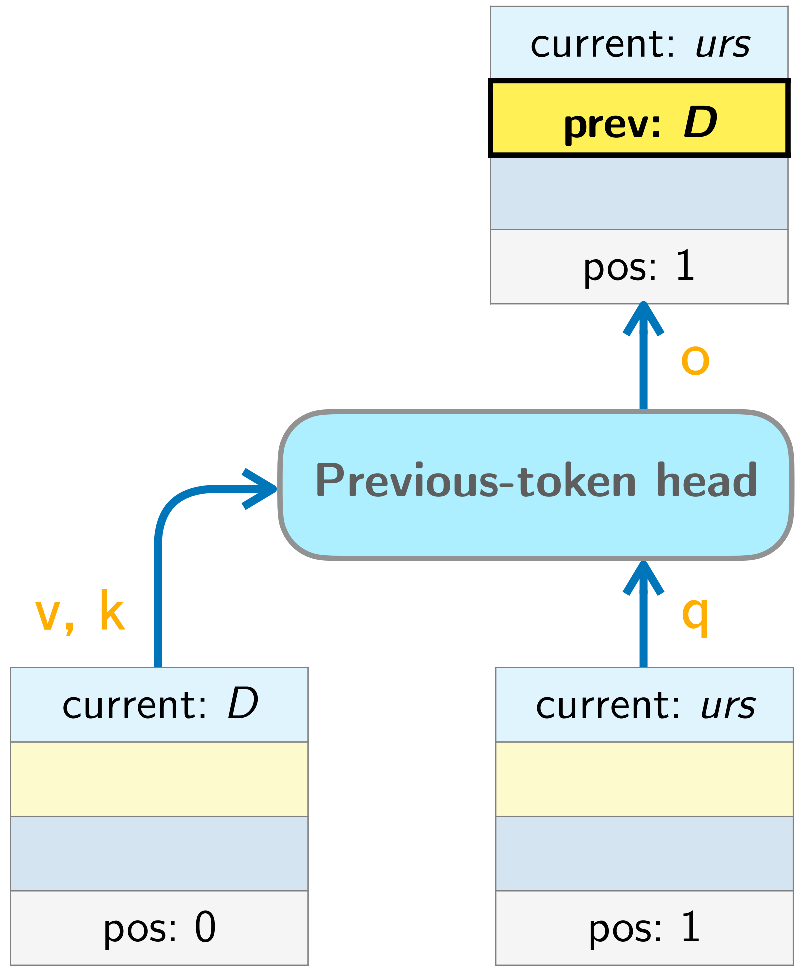
QK circuit
OV circuit

Paper: “In-context Learning and Induction Heads” by Olsson et al. (2022)
Inspired by and partially based on “Induction heads – illustrated” by TheMcDouglas



Embedding subspaces:

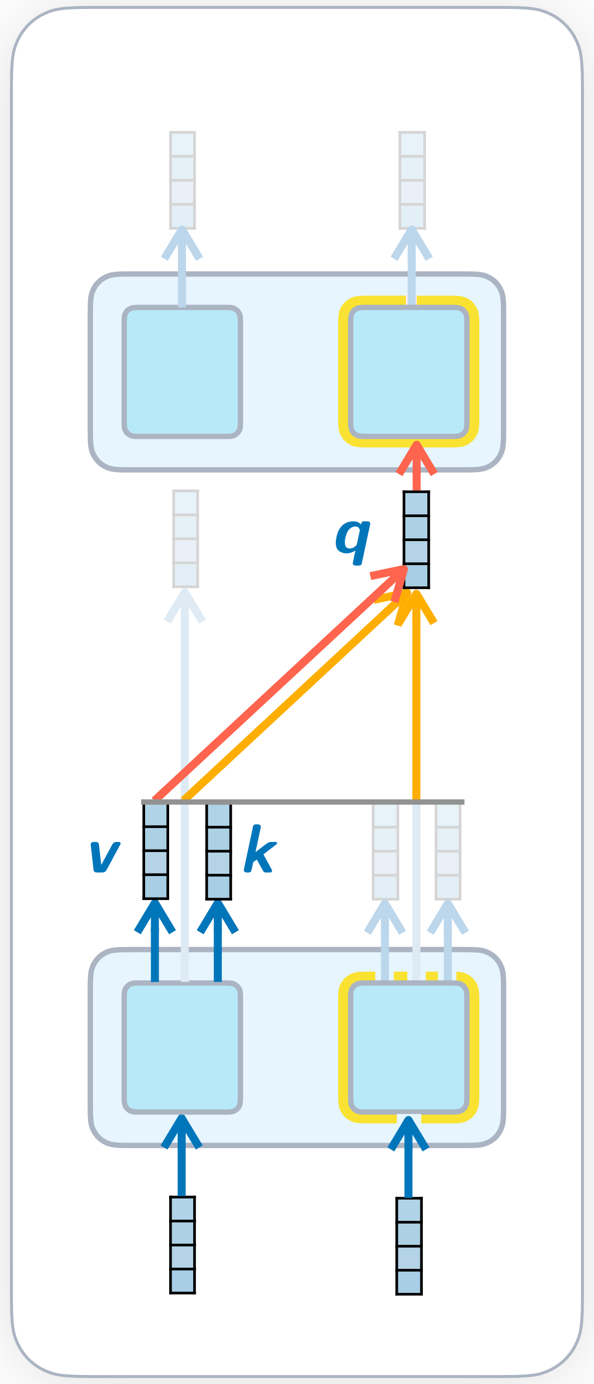
current token
previous token
next token
position



2 Layers: Induction heads

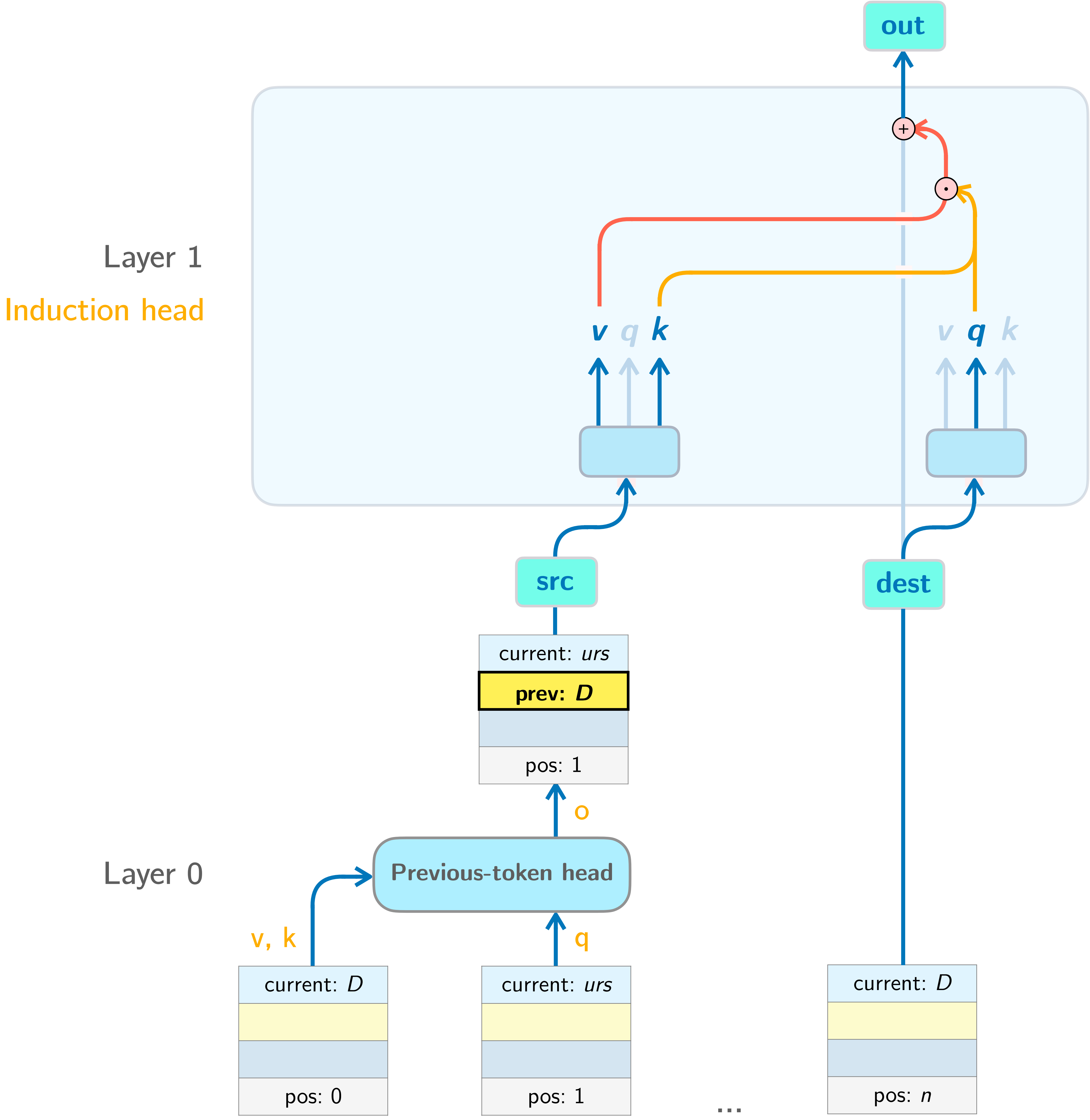
QK circuit
OV circuit

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

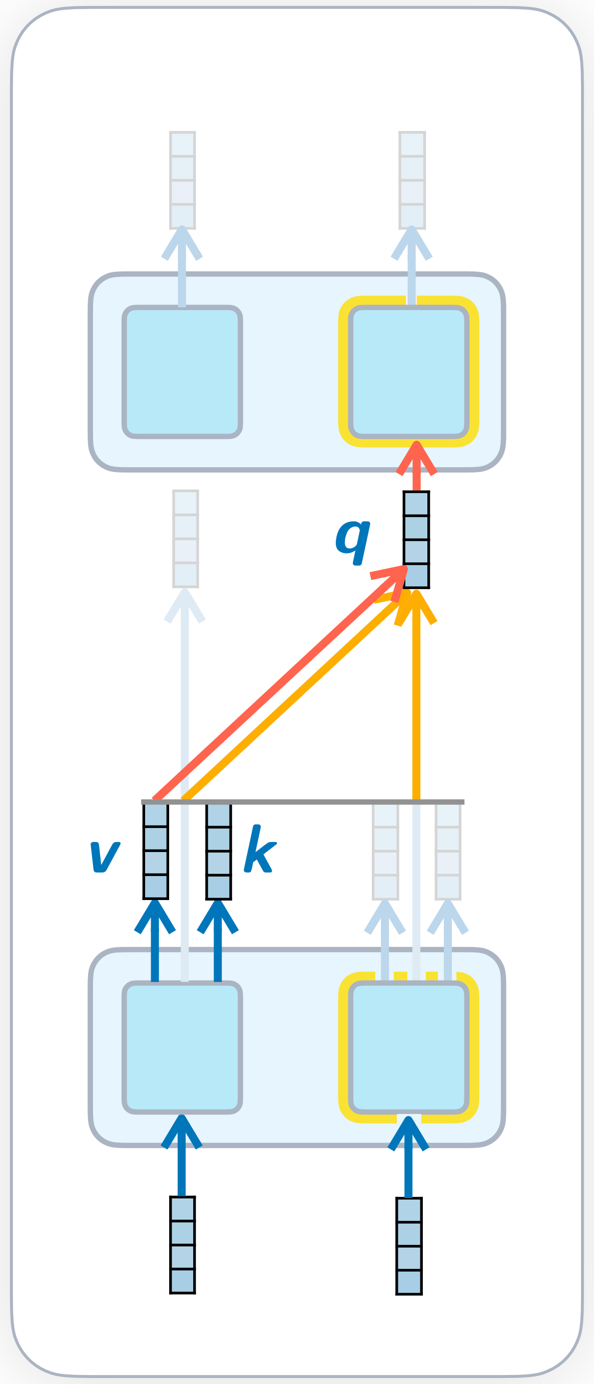
current token
previous token
next token
position

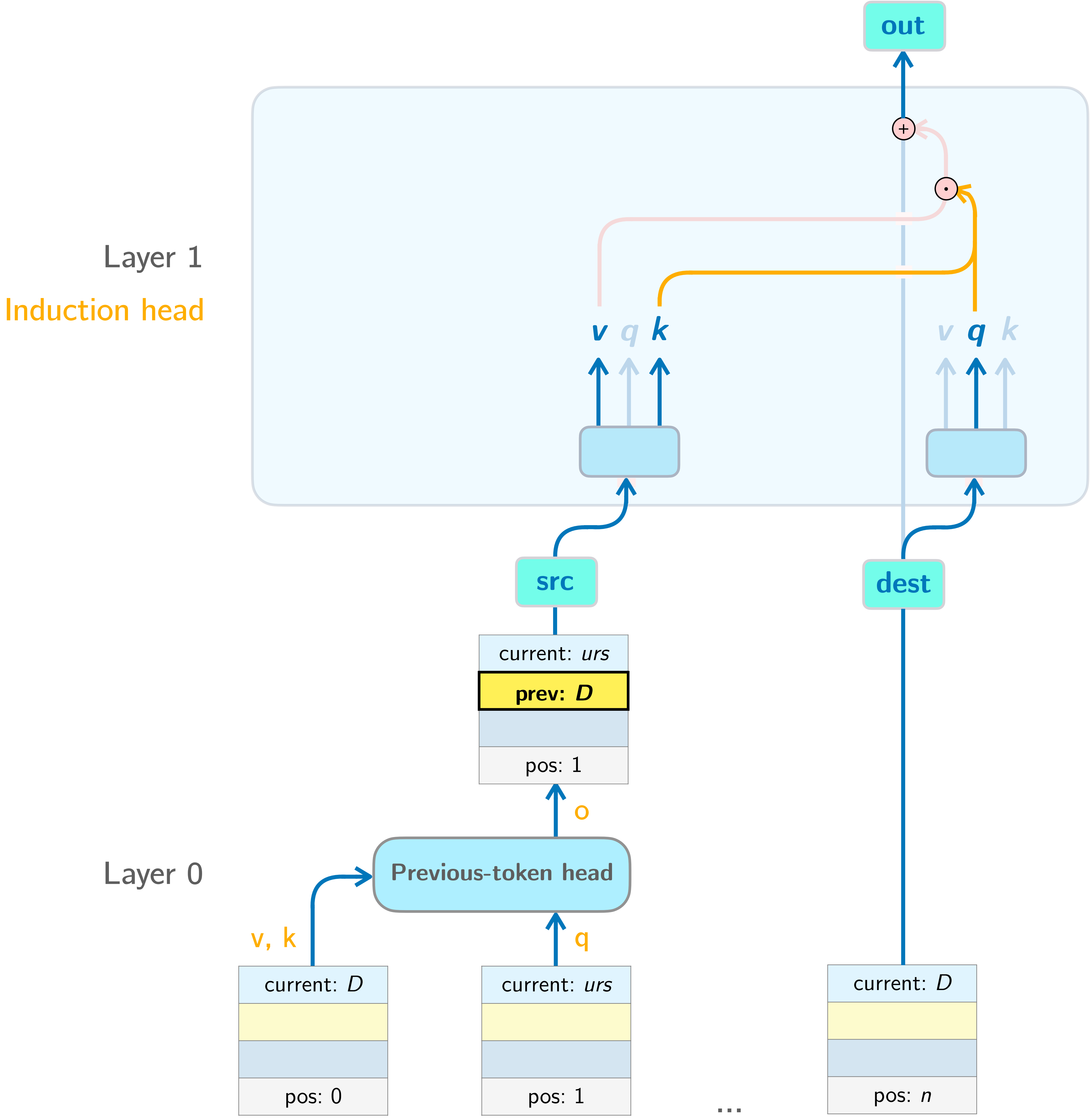


2 Layers: Induction heads

QK circuit
OV circuit

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas





Query for tokens where "prev: D".
It knows that "current: D" so simply replace "current" by "prev"

All other tokens have low attention score.

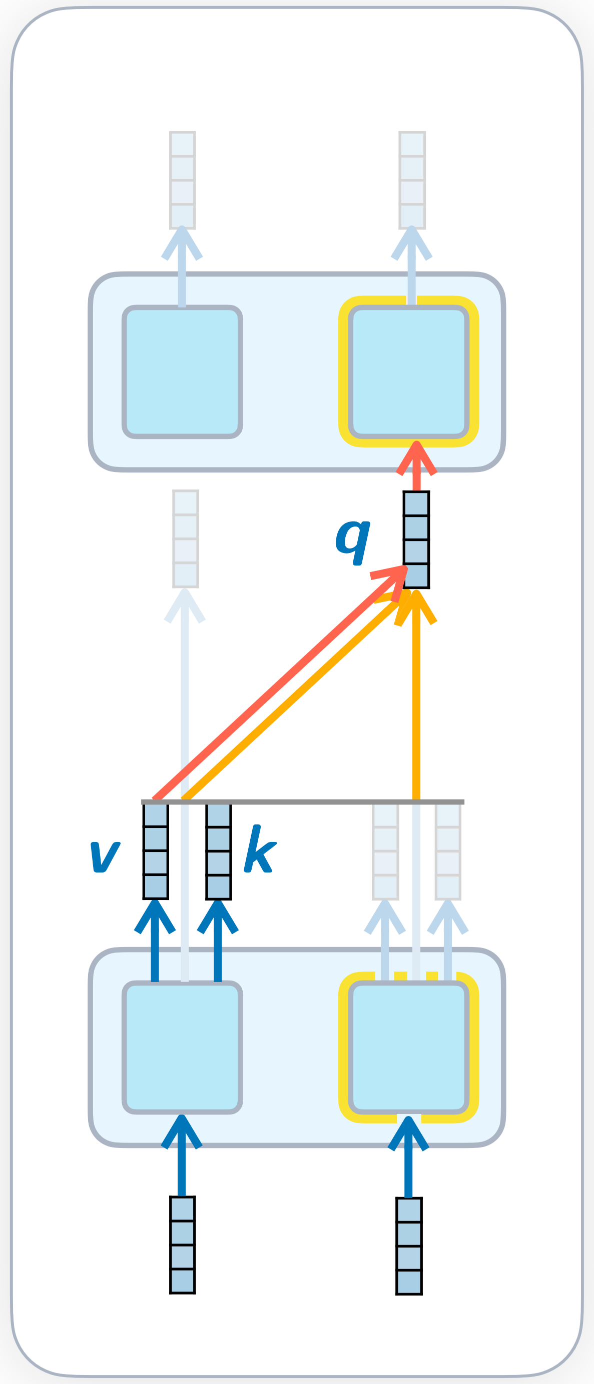
Embedding subspaces:

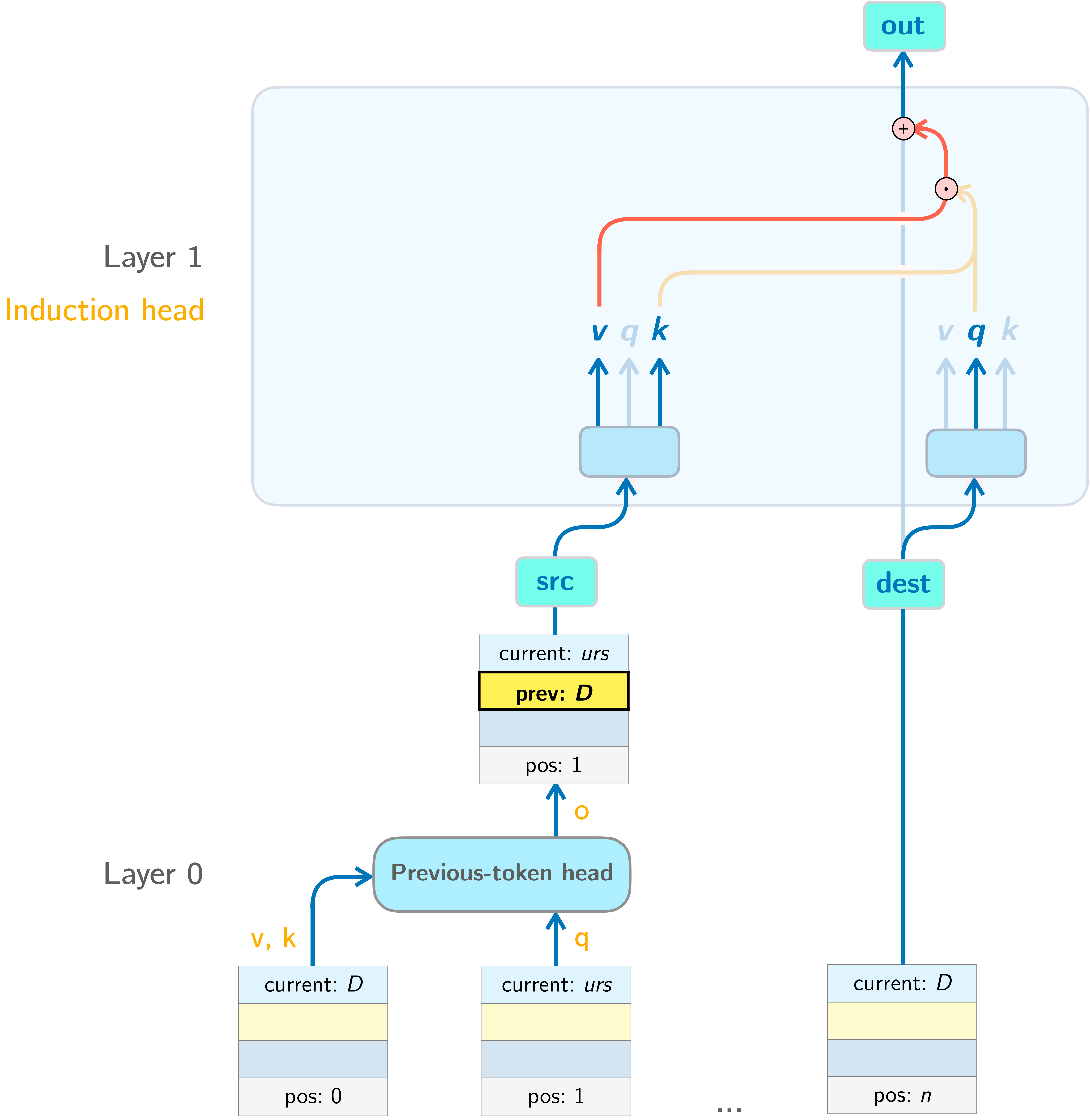
current token
previous token
next token
position

2 Layers: Induction heads

QK circuit
OV circuit

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)
Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas





The value would be "current: urs" but in the next-token subspace, so "next: urs"

Query for tokens where "prev: D". It knows that "current: D" so simply replace "current" by "prev"

All other tokens have low attention score.

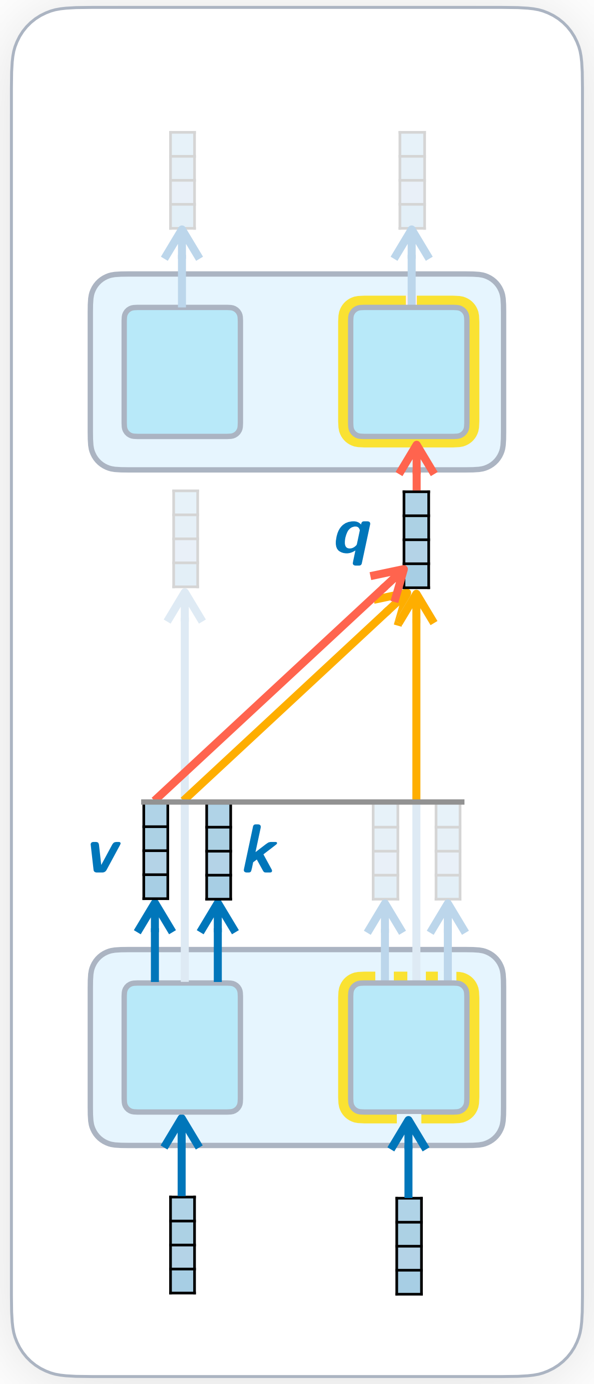
Embedding subspaces:

current token
previous token
next token
position

2 Layers: Induction heads

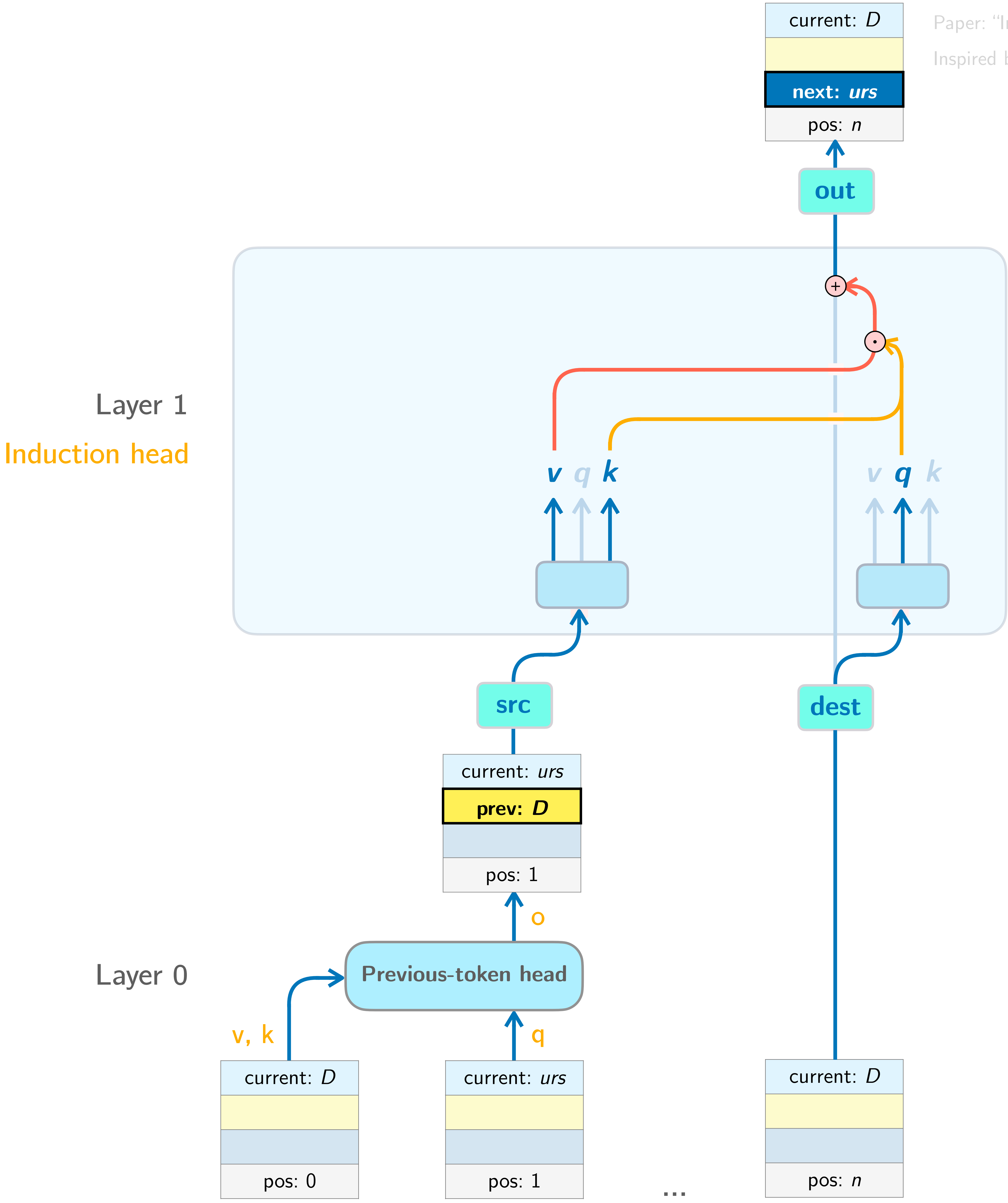
QK circuit

OV circuit



Embedding subspaces:

current token
previous token
next token
position



Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

The value would be "current: urs" but in the next-token subspace, so "next: urs"

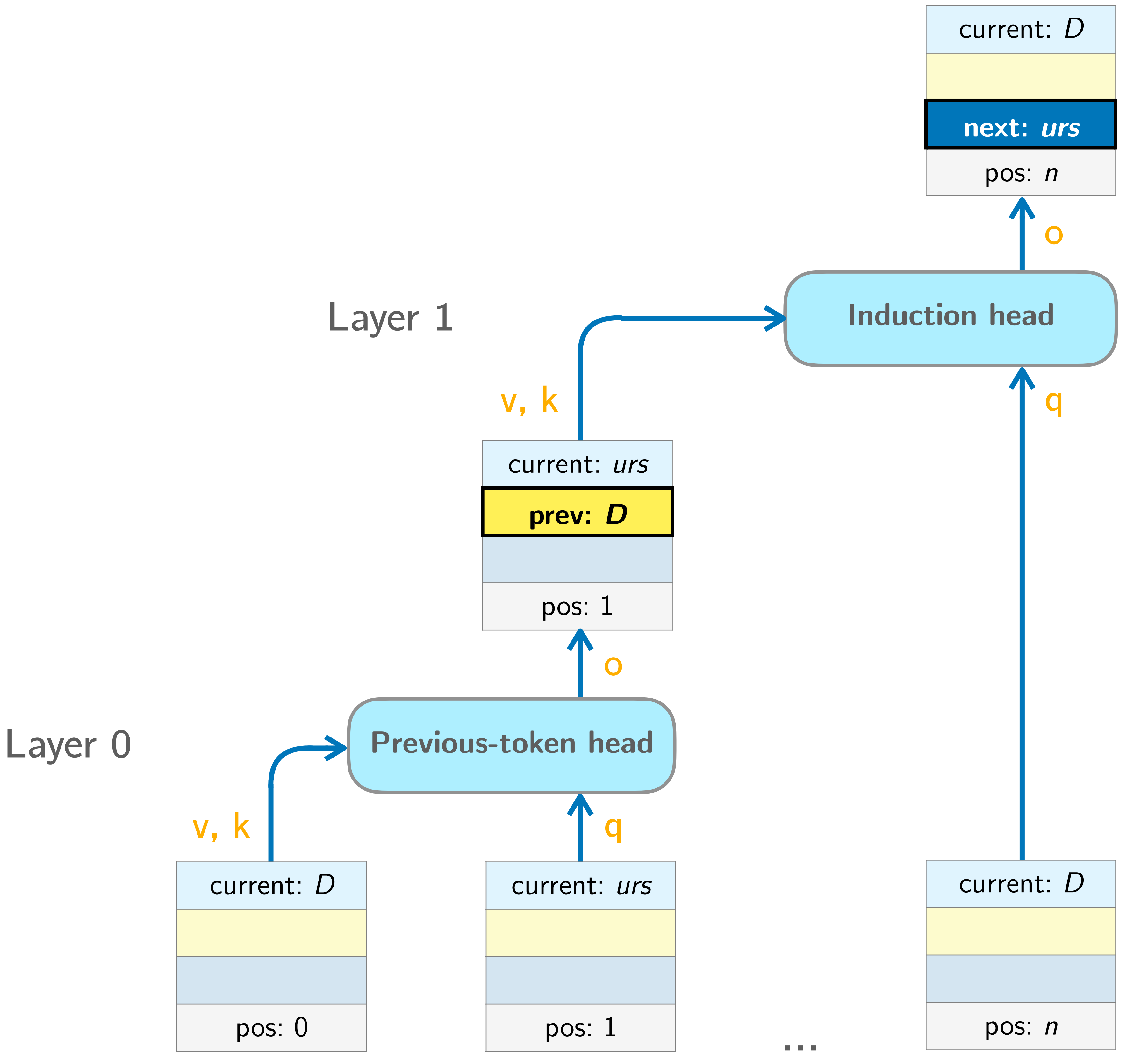
Query for tokens where "prev: D". It knows that "current: D" so simply replace "current" by "prev"

All other tokens have low attention score.

Induction: [Durs][ley]…[Durs] → [ley]

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

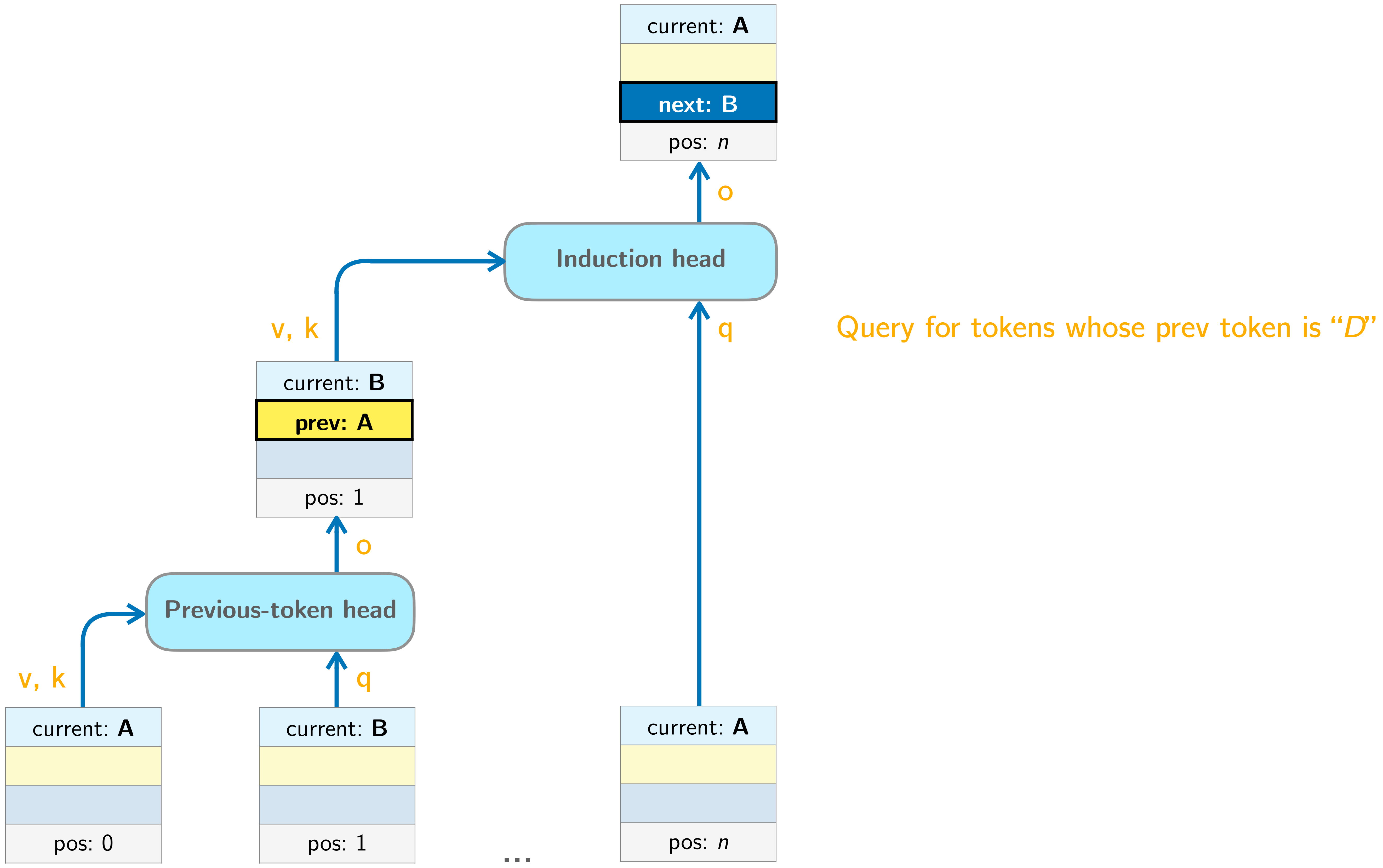
current token
previous token
next token
position

2 Layers: Induction heads

Induction: [A][B]...[A] → [B]

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



Embedding subspaces:

current token
previous token
next token
position

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

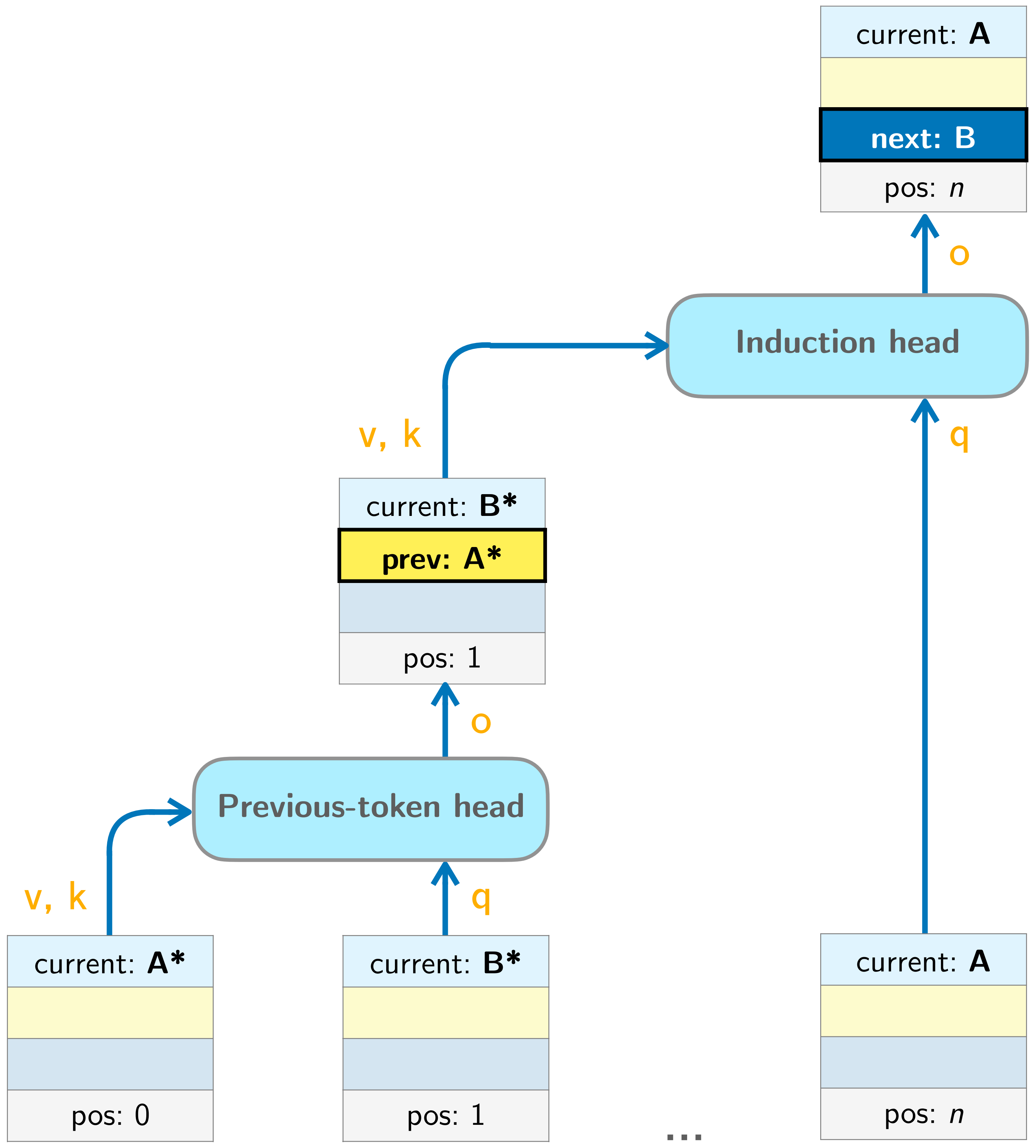
Fuzzy induction: [A*][B*]...[A] → [B]

(as opposed to literal induction: [A][B]...[A] → [B])

A*, B* can be highly abstracted

if this **induction circuit** is located in a sufficiently high layer,

as the circuit is working with highly refined/contextualized representations.



Query for tokens whose prev token is *similar to* **A**

A* is *similar to* A in some characteristic

B* is *similar to* B in some characteristic

e.g. about similar topics, both are adjectives, or nouns, both are pronouns that refer to the same referent, both are the same type of pronoun, both have the same grammatical gender

Embedding subspaces:

current token
previous token
next token
position

2 Layers: Induction heads

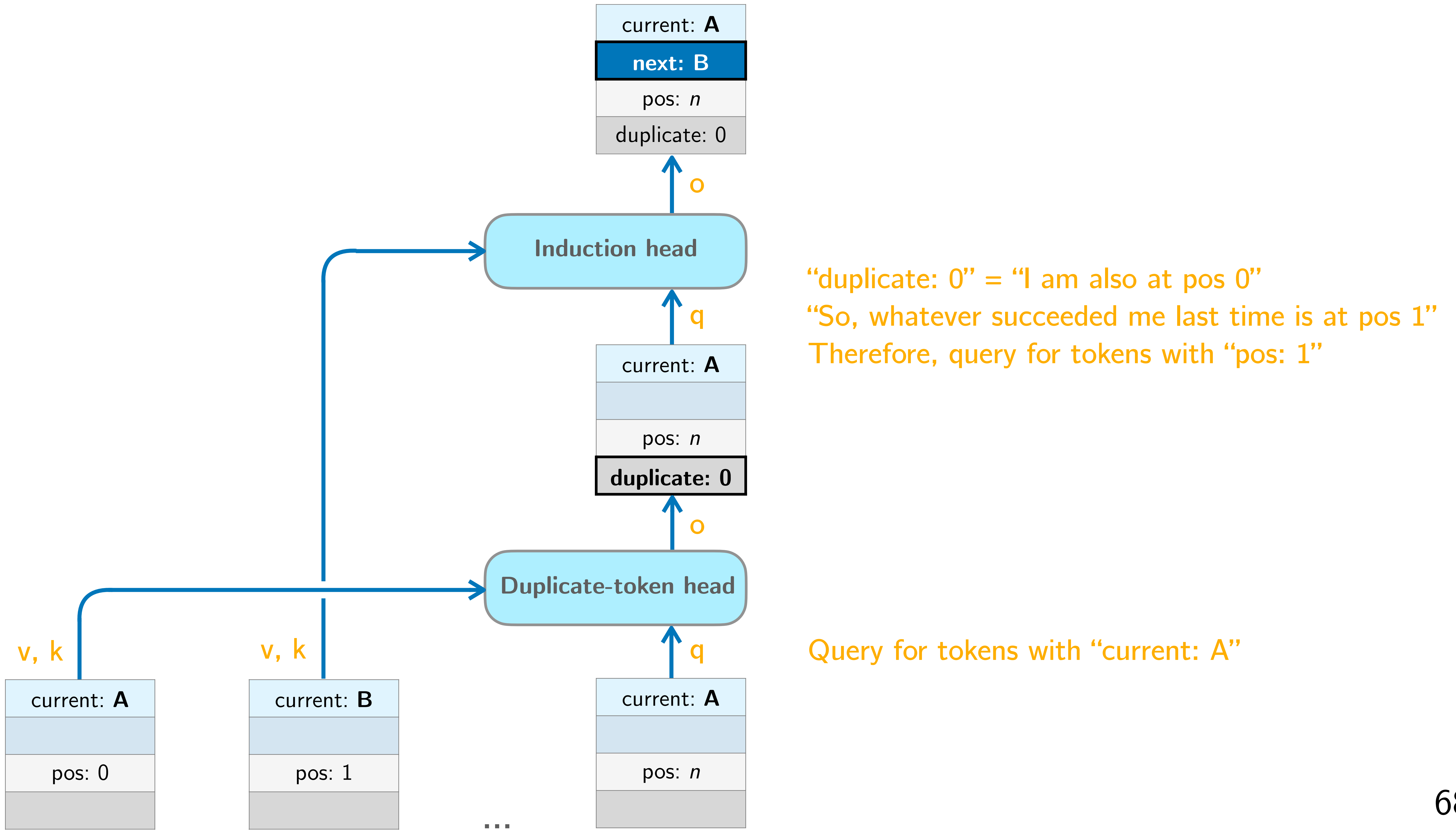
Paper: “In-context Learning and Induction Heads” by Olsson et al. (2022)

Inspired by and partially based on “Induction heads – illustrated” by TheMcDouglas

Induction: [A][B]...[A] → [B]

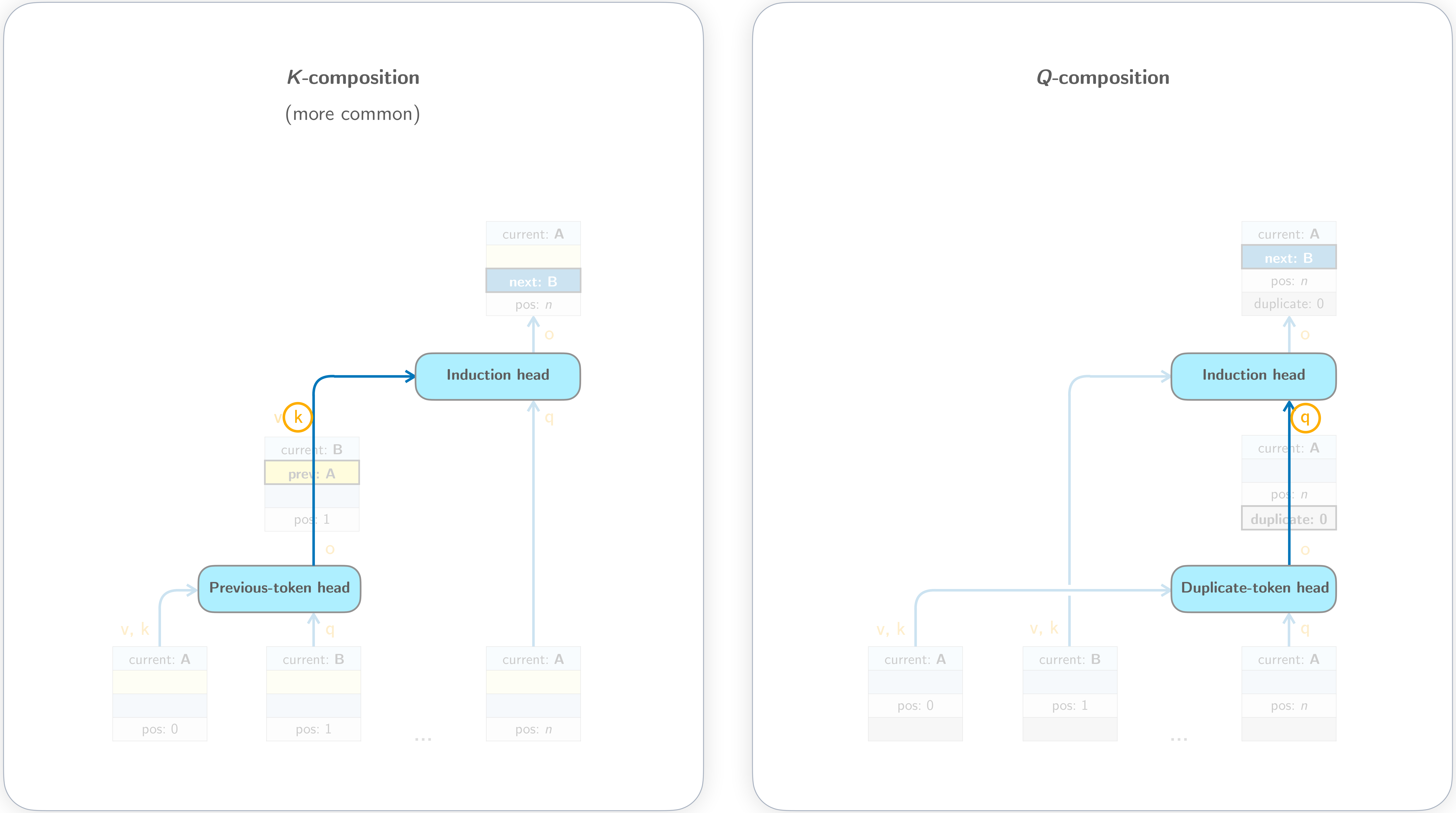
Alternative induction-head design using **Q-composition.**

(The design we discussed so far is using **K-composition**.)



Embedding subspaces:

current token
next token
position
duplicate position

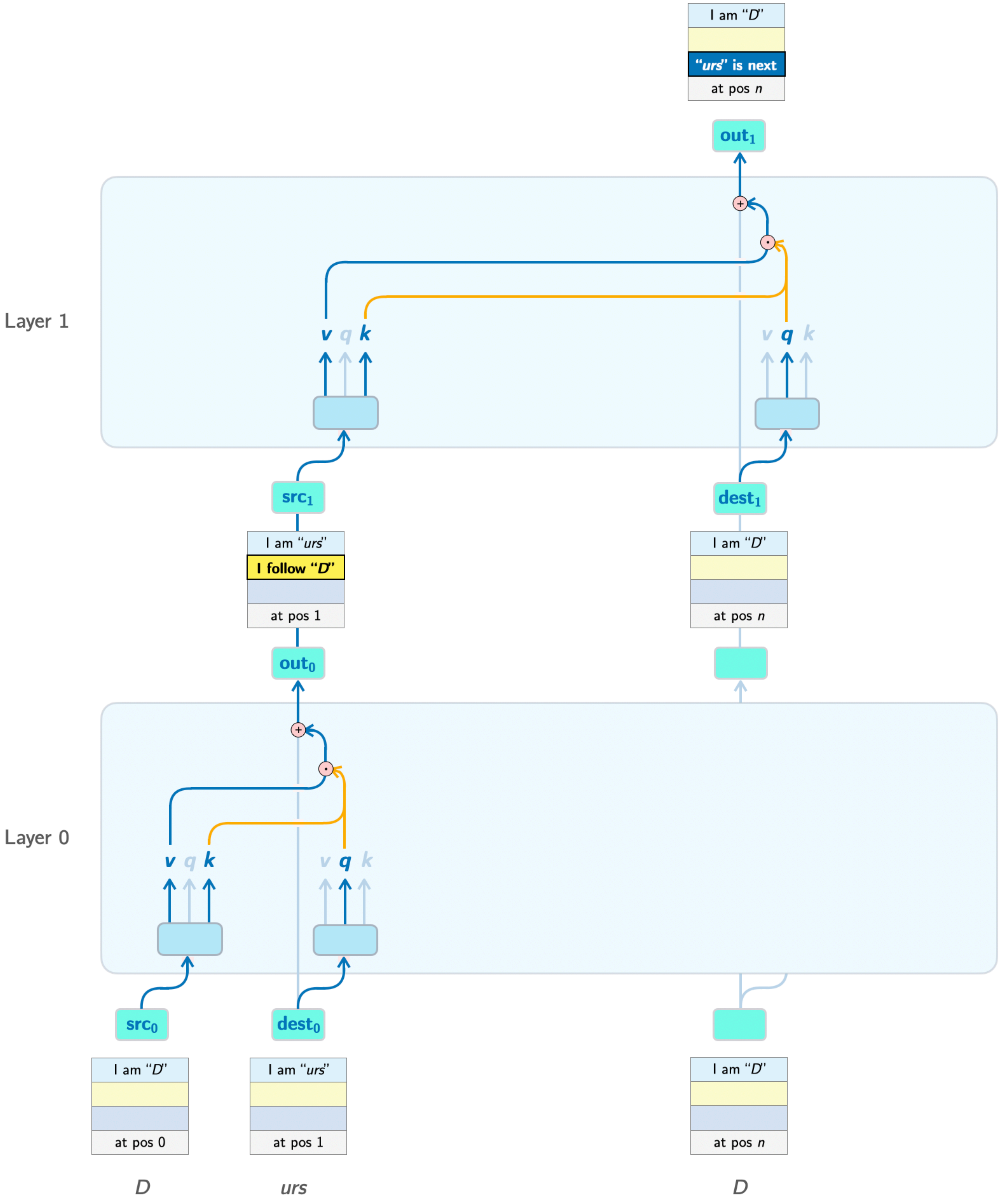


K-composition
(more common)
current: A
next: B
pos: n
o
Induction head
q
v
k
current: B
prev: A
pos: 1
o
Previous-token head
v, k
q
current: A
pos: 0
current: B
pos: 1
...
current: A
pos: n
Q-composition
current: A
next: B
pos: n
duplicate: 0
o
Induction head
q
current: A
pos: n
duplicate: 0
o
Duplicate-token head
v, k
v, k
q
current: A
pos: 0
current: B
pos: 1
...
current: A
pos: n

2 Layers: Induction heads

Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



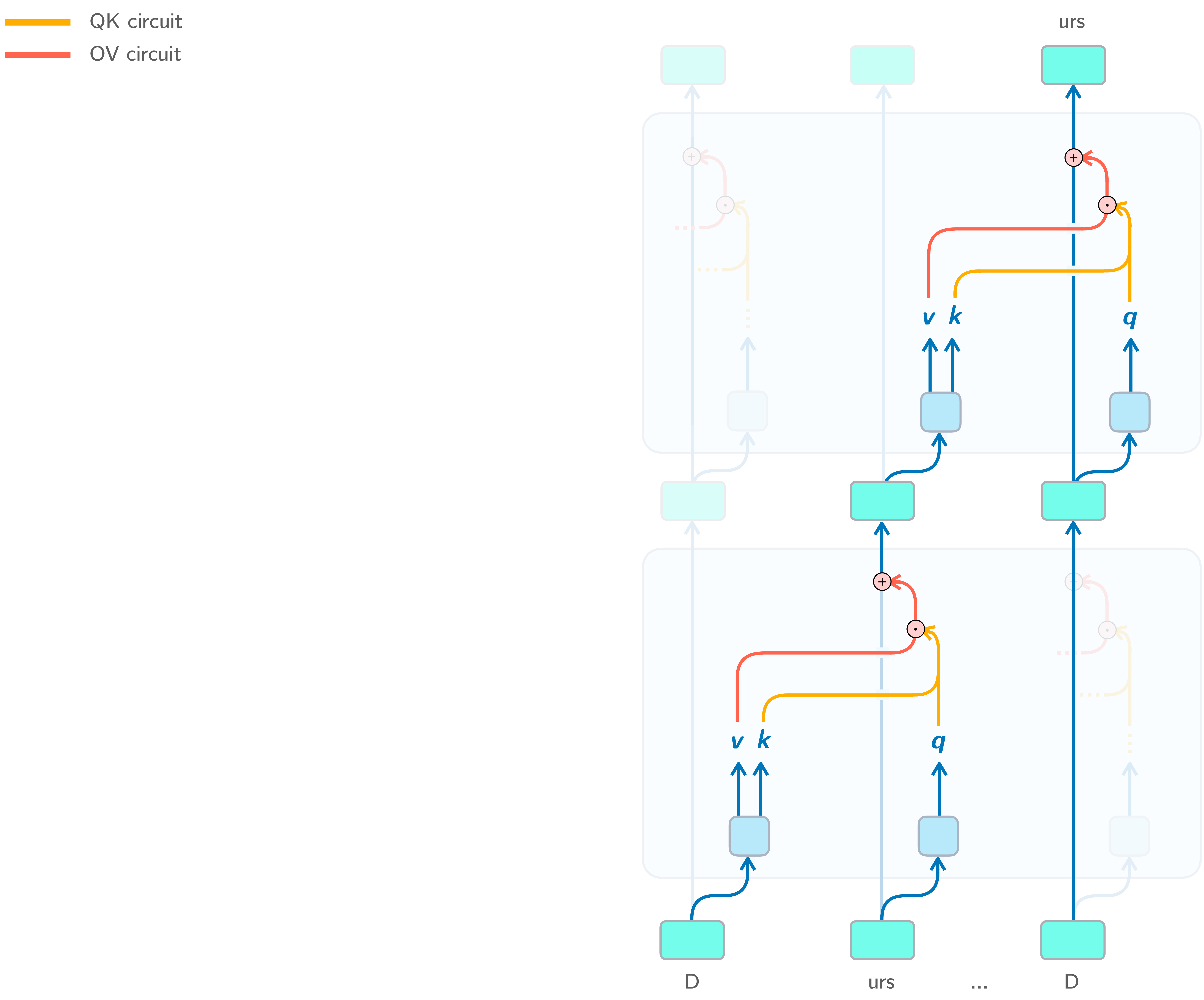
Embedding subspaces:

current token
previous token
next token
position

2 Layers: Induction heads

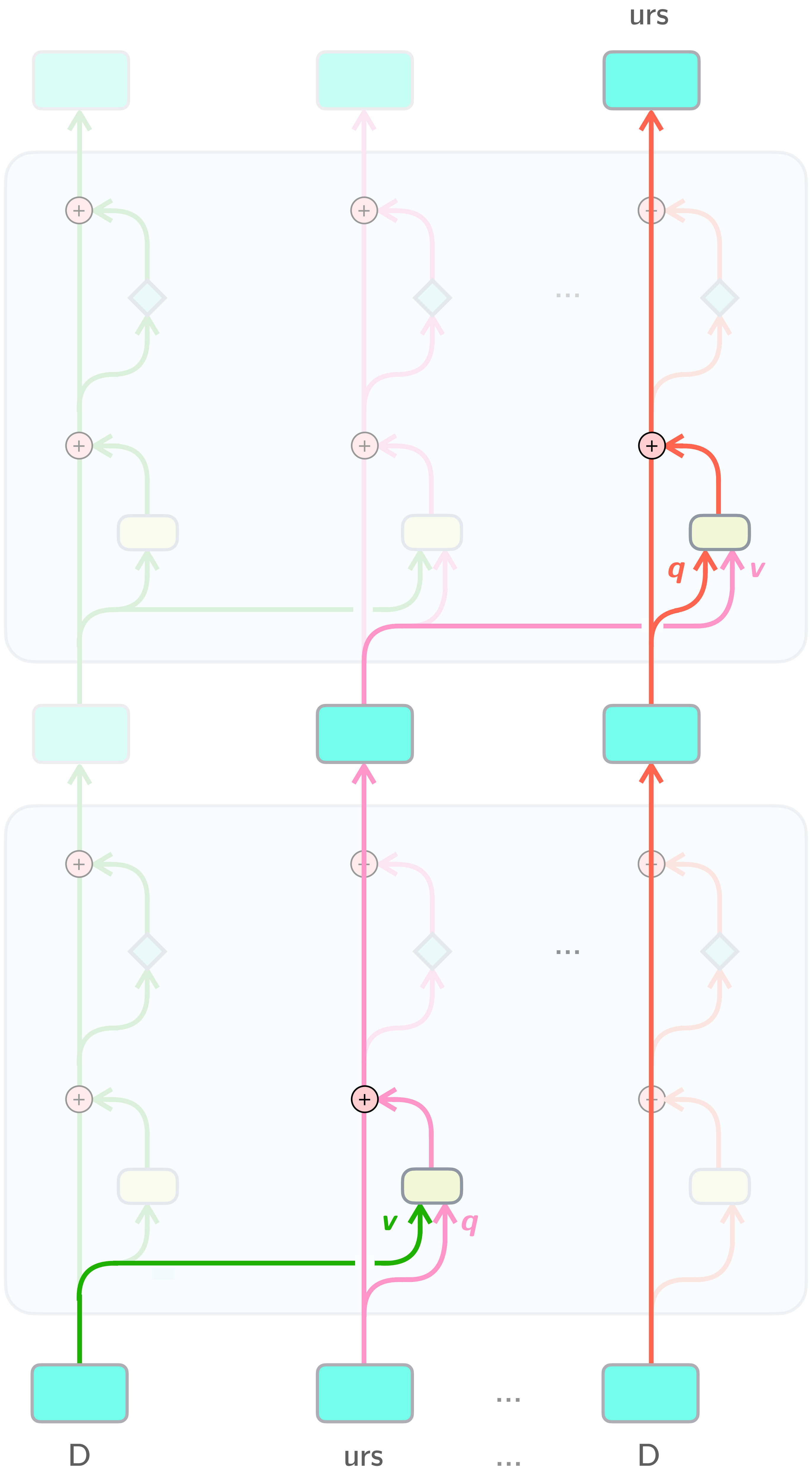
Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas



2 Layers: Induction heads

TODO: change colors to avoid confusion?



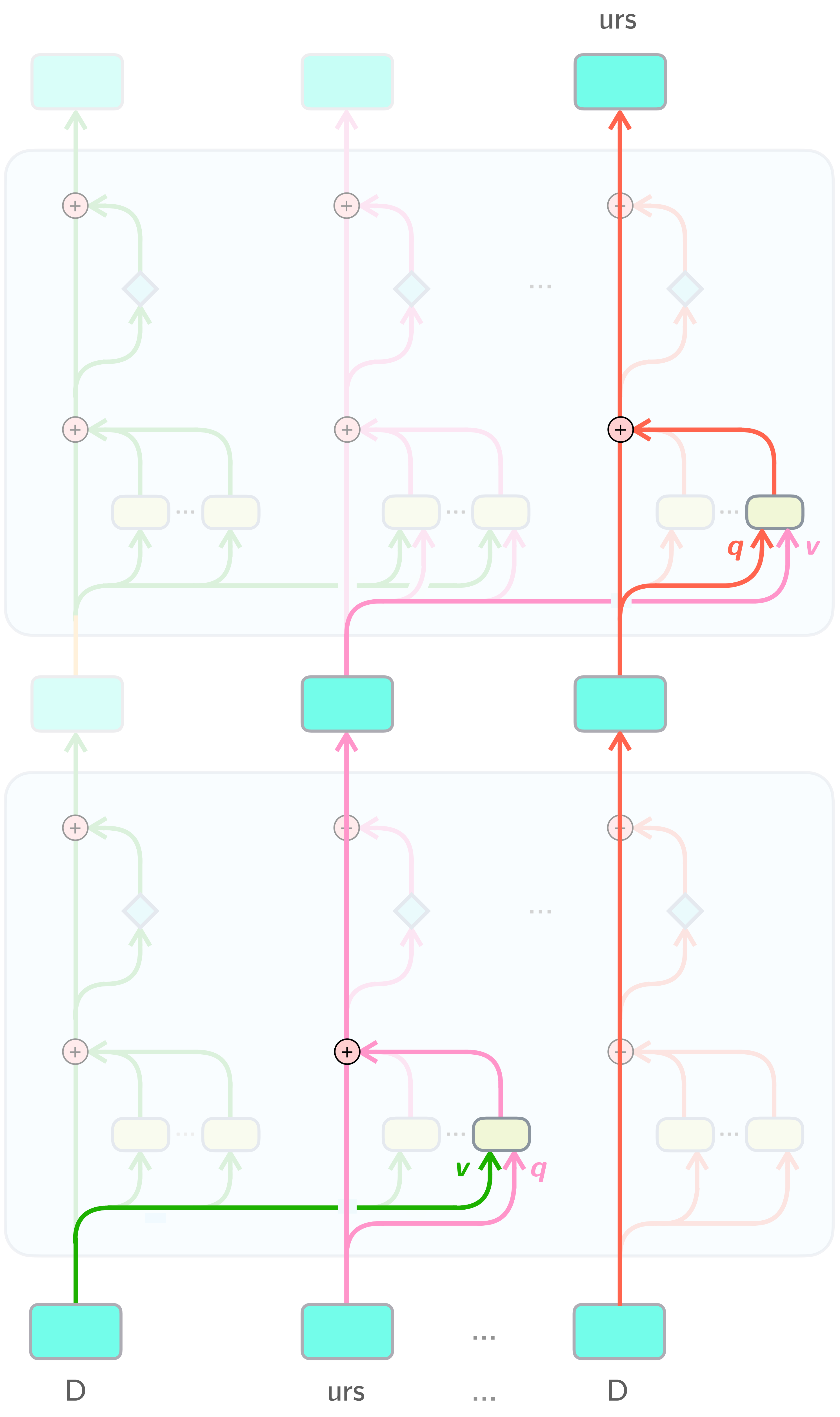
Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

Only showing relevant value vectors, v, in order to show how information flows. Not showing k at all.

2 Layers: Induction heads



Paper: "In-context Learning and Induction Heads" by Olsson et al. (2022)

Inspired by and partially based on "Induction heads – illustrated" by TheMcDouglas

Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

Here, it is shown that there are multiple attention heads, and the induction head of interest is only one of those heads.

Only showing relevant value vectors, v, in order to show how information flows. Not showing k at all.

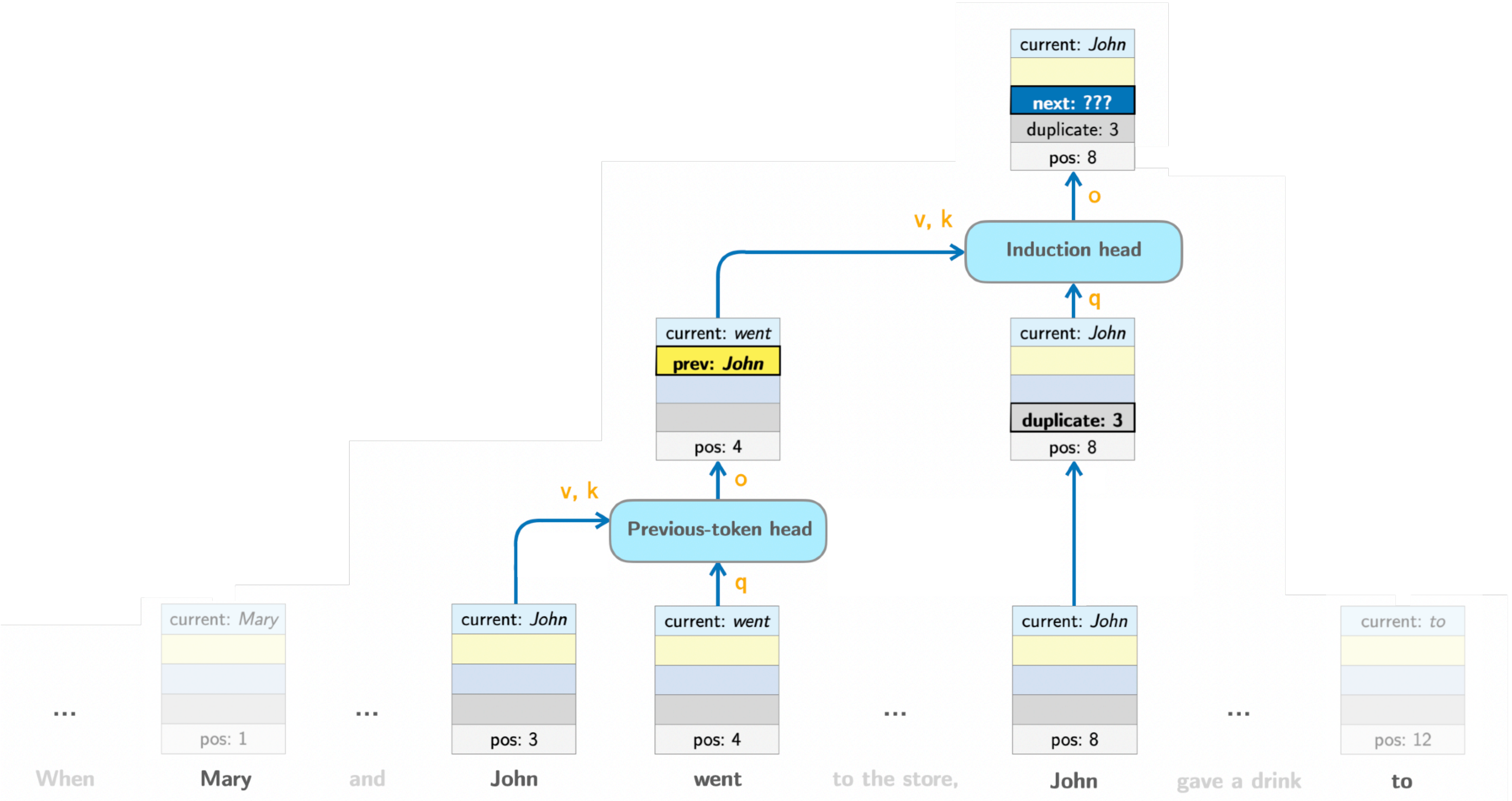
2

Indirect-object-identification circuit

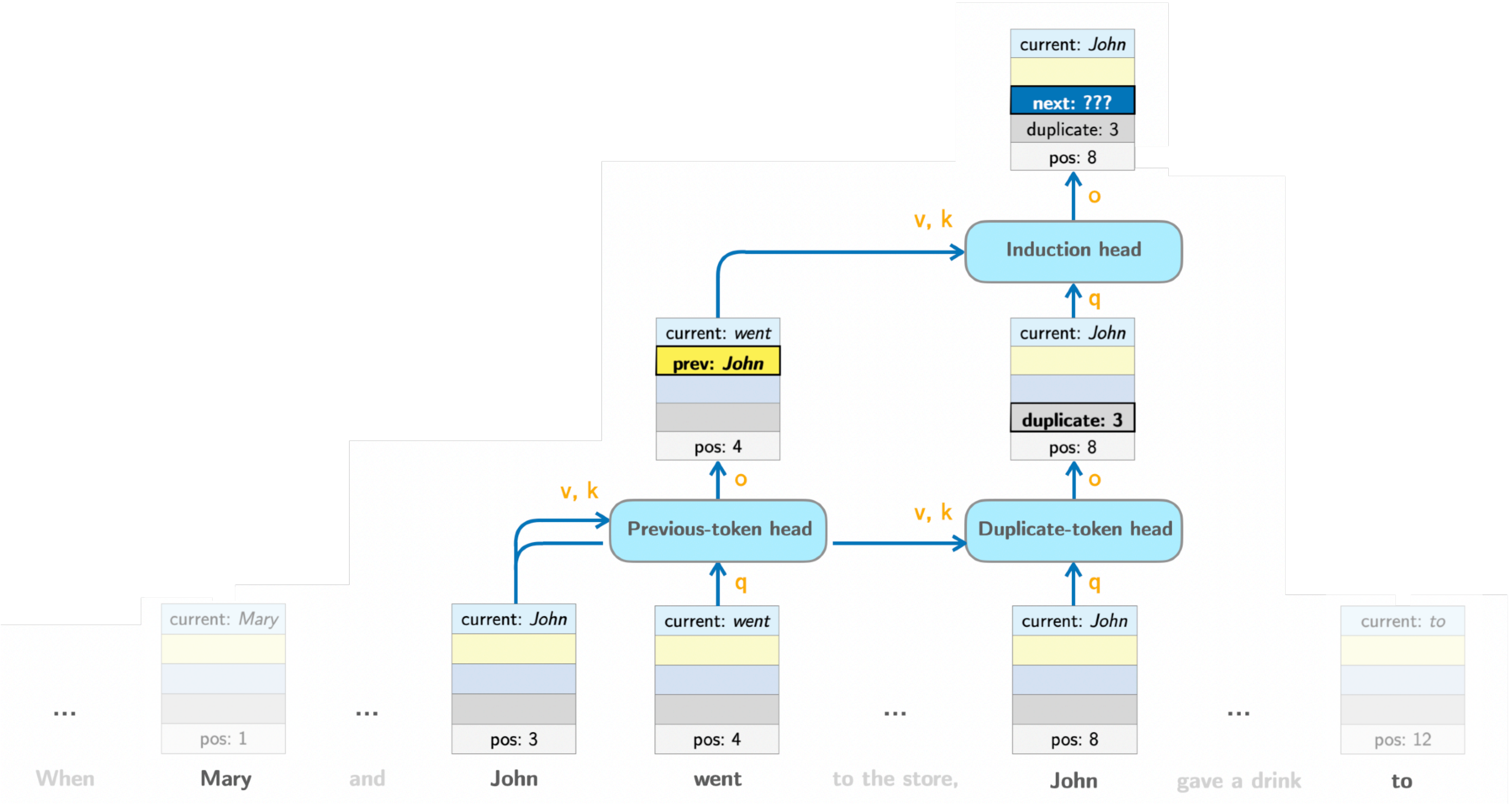
`When Mary and John went to the store, John gave a drink to` __________

Infer the ***indirect object***
by identifying it in the context

GPT-2 small

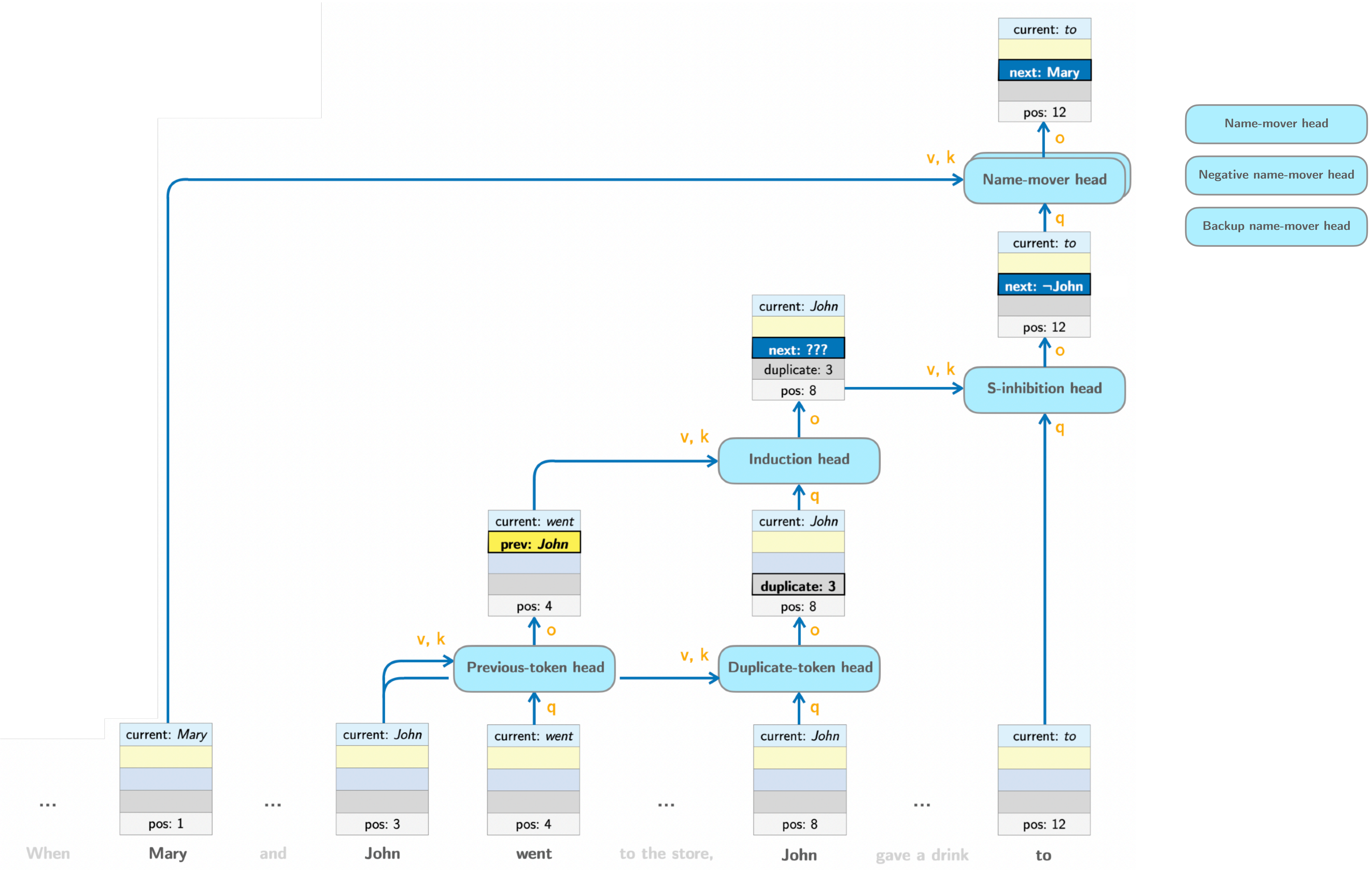

current: John
next: ???
duplicate: 3
pos: 8
o
v, k
Induction head
q
current: went
prev: John
pos: 4
current: John
duplicate: 3
pos: 8
o
v, k
Previous-token head
q
...
current: Mary
pos: 1
...
current: John
pos: 3
current: went
pos: 4
...
current: John
pos: 8
...
current: to
pos: 12
When
Mary
and
John
went
to the store,
John
gave a drink
to

GPT-2 small



current: *John*
next: ???
duplicate: 3
pos: 8
o
v, k
Induction head
q
current: *went*
prev: *John*
pos: 4
current: *John*
duplicate: 3
pos: 8
o
v, k
Previous-token head
v, k
Duplicate-token head
o
q
q
...
current: *Mary*
pos: 1
...
current: *John*
pos: 3
current: *went*
pos: 4
...
current: *John*
pos: 8
...
current: *to*
pos: 12
When
Mary
and
John
went
to the store,
John
gave a drink
to

GPT-2 small



current: to
next: Mary
pos: 12
o
v, k
Name-mover head
q
current: to
next: ¬John
pos: 12
o
v, k
S-inhibition head
q
current: John
next: ???
duplicate: 3
pos: 8
o
v, k
Induction head
q
current: John
duplicate: 3
pos: 8
current: went
prev: John
pos: 4
o
v, k
Previous-token head
q
v, k
Duplicate-token head
o
q
current: Mary
pos: 1
current: John
pos: 3
current: went
pos: 4
current: John
pos: 8
current: to
pos: 12
...
When
Mary
and
John
went
to the store,
John
gave a drink
to
Name-mover head
Negative name-mover head
Backup name-mover head

Thanks for your attention.

References

[redacted]

Language models

- A Mathematical Framework of Transformer Circuits
- Induction Heads and In-context Learning
- Induction heads – illustrated
- Toy Models of superposition
- Towards monosemanticity: Decomposing Language Models with Dictionary Learning
- Softmax Linear Units
- Superposition, Memorization, and Double Descent
- Privileged Bases in the Transformer Residual Stream
- Distributed Representations: Composition & Superposition
- (Grokking) …
- Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 small
- Copy Suppression: Comprehensively Understanding An Attention Head
- Does Circuit Analysis Interpretability Scale? Evidence from Multiple Choice Capabilities in Chinchilla
- Tracr: Compiled Transformers as a Laboratory for Interpretability
- Towards Automated Circuit Discovery for Mechanistic Interpretability
- **(Perhaps)** Studying Large Language Model Generalization with Influence Functions