University of Toronto
Department of Computer Science

CSC420

Introduction to Image Understanding

Fall 2026

Presented by Javad Rajabi
Teaching Assistant

Instructor: David Lindell





CSC420: Introduction to Image Understanding

Attention

How Transformers understand words and images

What we will cover

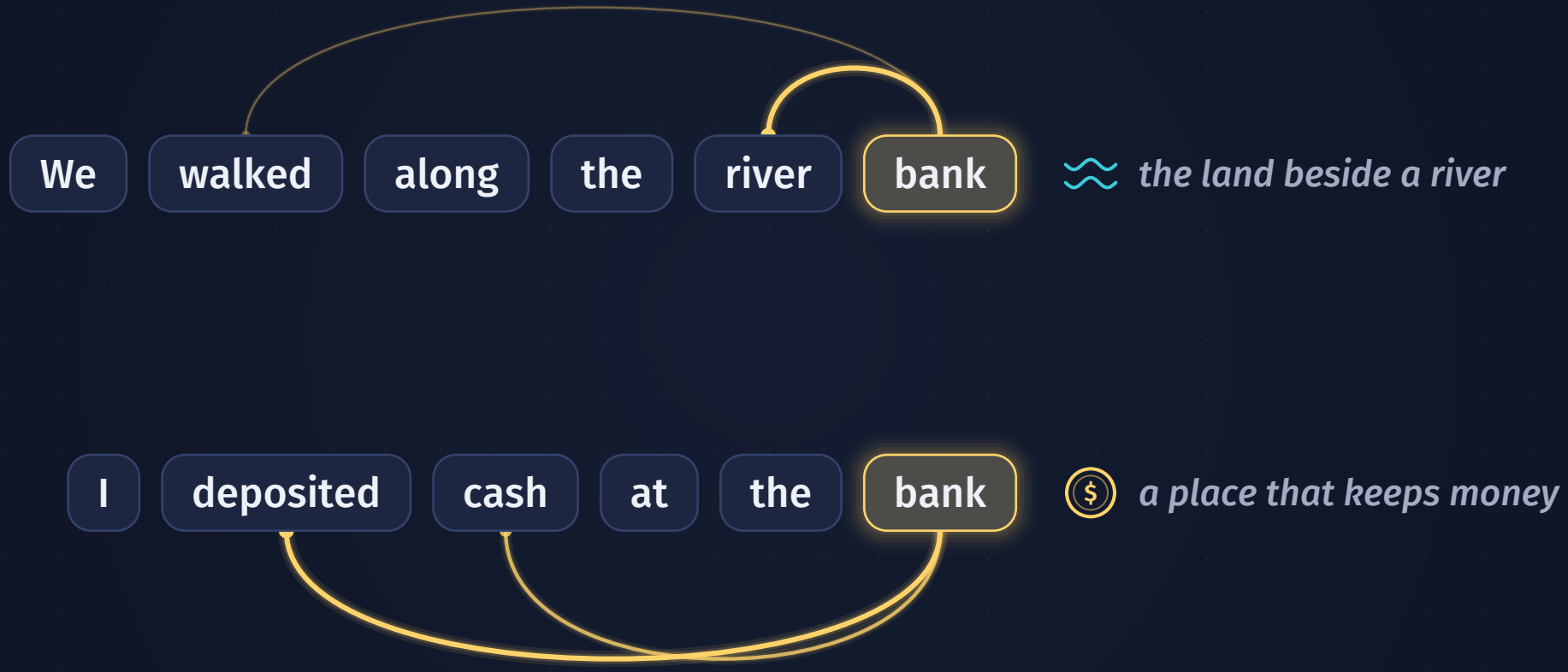
1 Why attention?
2 Before attention
3 Words as vectors
4 The big idea
5 Query, key and value
6 The math, step by step
7 All at once: the matrix form
8 Multi-head attention
9 Positional encoding
10 Masked attention
11 Cross-attention
12 Inside a Transformer
13 Vision Transformers
14 Attention in computer vision
15 Attention everywhere
16 The cost of attention
17 Recap

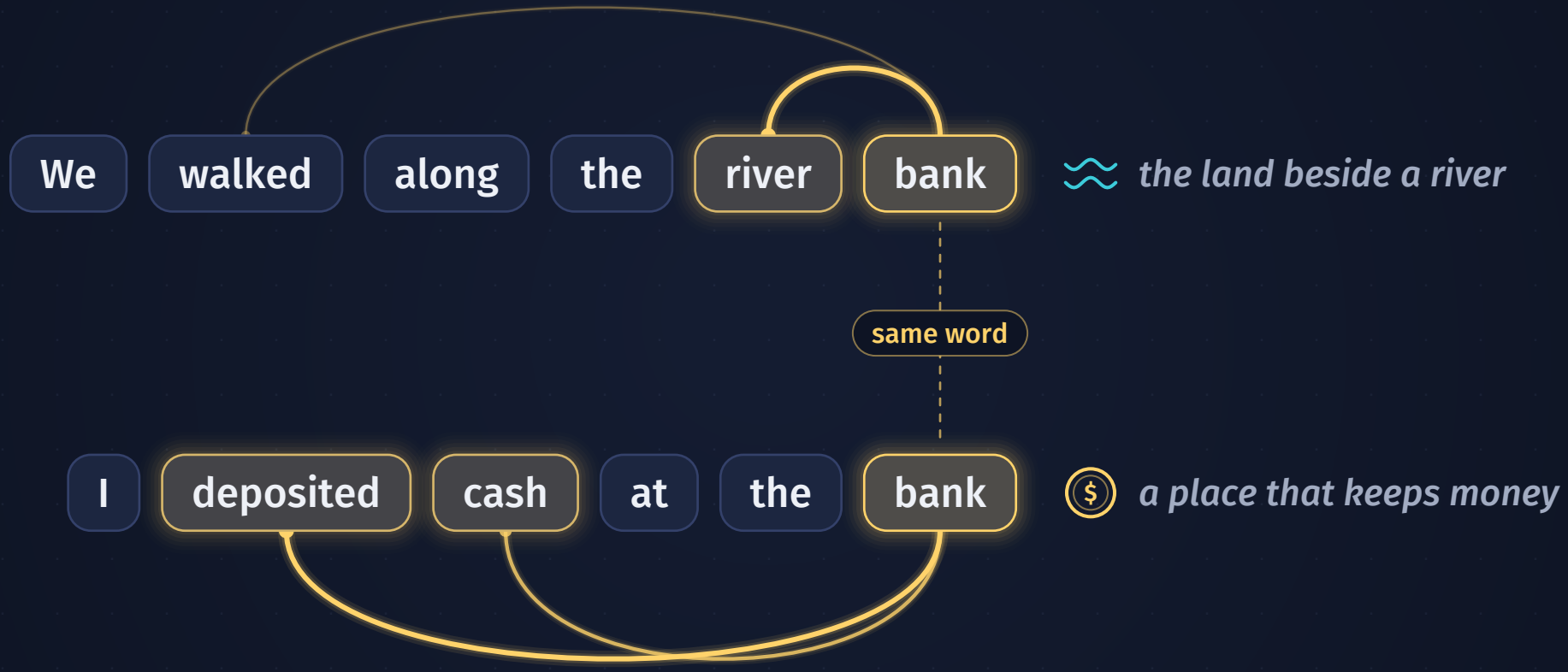
1 Why attention?

Words only make sense in context

We walked along the river bank

I deposited cash at the bank


We
walked
along
the
river
bank
the land beside a river
I
deposited
cash
at
the
bank
a place that keeps money


We
walked
along
the
river
bank
the land beside a river
same word
I
deposited
cash
at
the
bank
a place that keeps money


The
animal
didn't
cross
the
street
because
it
was
too
tired
it = the animal

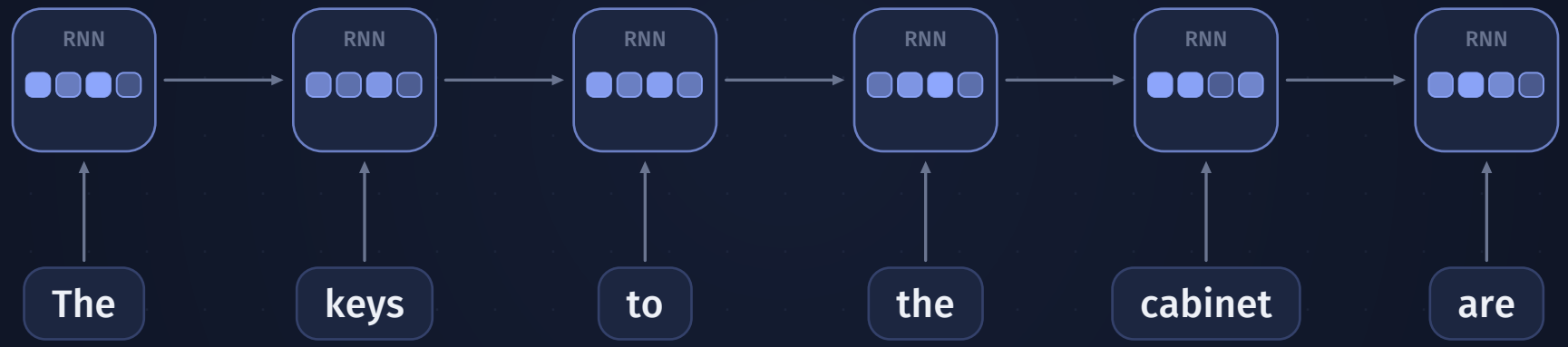

The
animal
didn't
cross
the
street
because
it
was
too
wide
it = the street

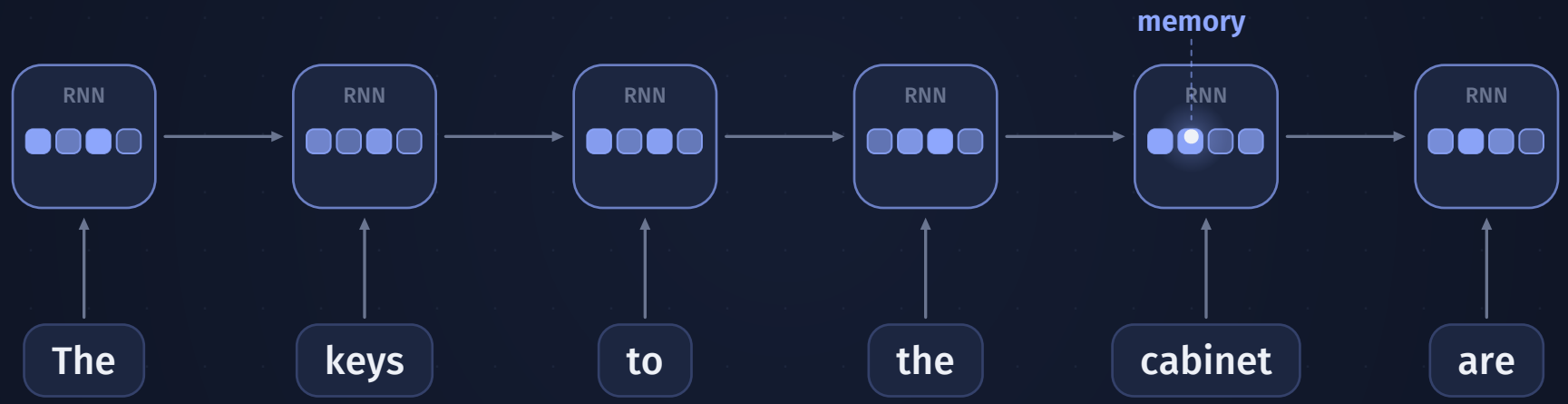
attention

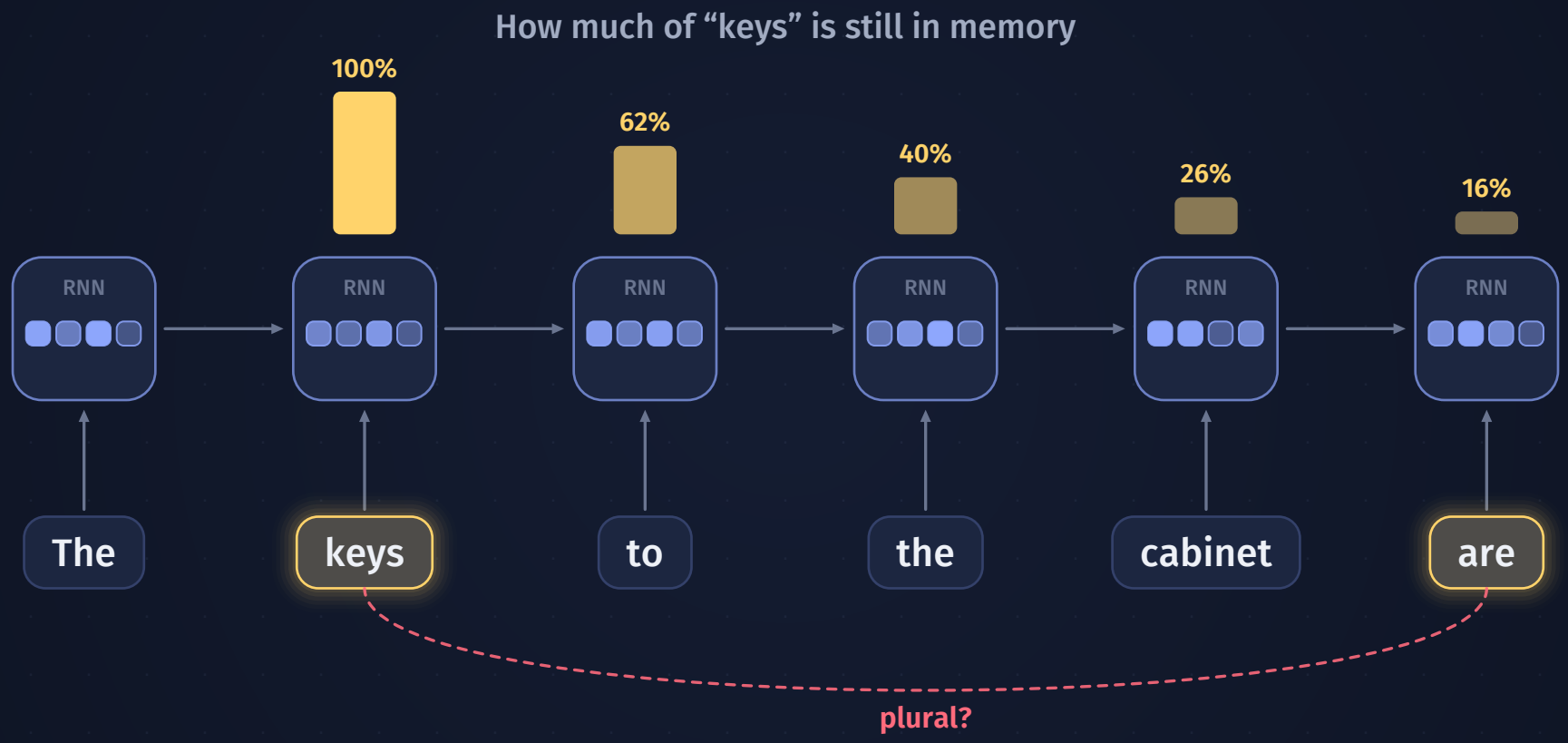


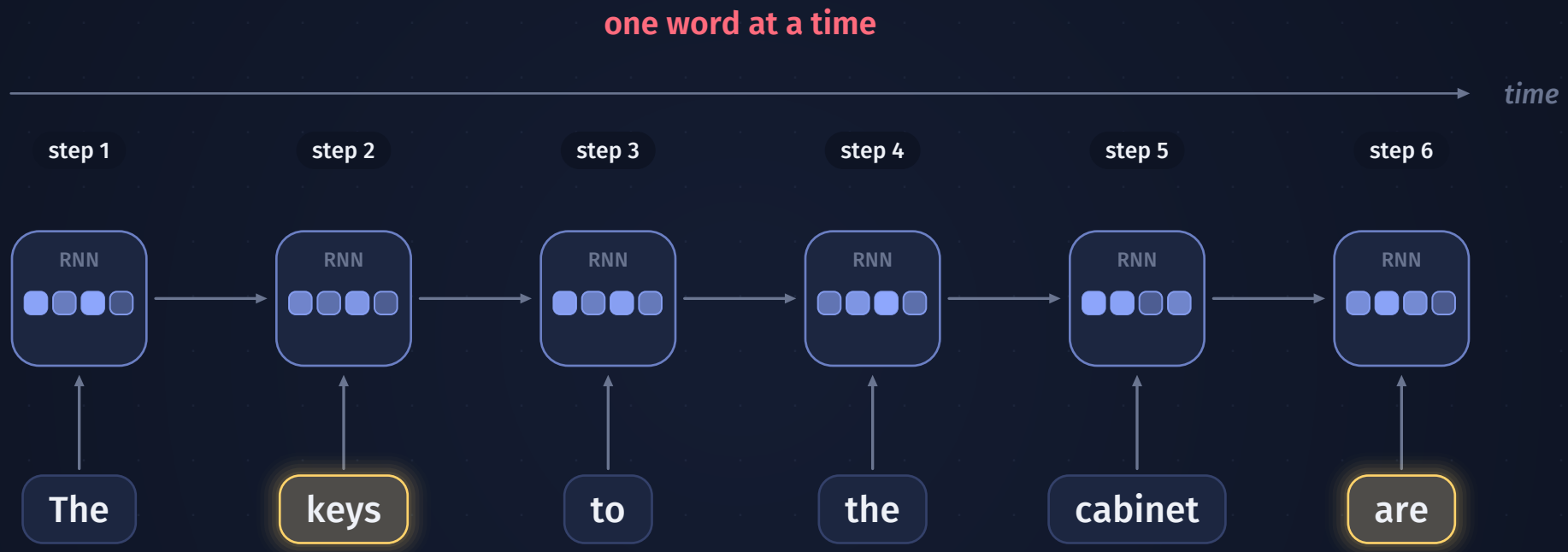
2 Before attention

Why older models struggled


RNN
RNN
RNN
RNN
RNN
RNN
The
keys
to
the
cabinet
are


memory
RNN
RNN
RNN
RNN
RNN
RNN
The
keys
to
the
cabinet
are


How much of “keys” is still in memory
100%
62%
40%
26%
16%
RNN
RNN
RNN
RNN
RNN
RNN
The
keys
to
the
cabinet
are
plural?


one word at a time
time
step 1
step 2
step 3
step 4
step 5
step 6
RNN
RNN
RNN
RNN
RNN
RNN
The
keys
to
the
cabinet
are

Bahdanau, Cho and Bengio, 2014



Bahdanau, Cho and Bengio, 2014



2 Before attention

Attention Is All You Need

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin

2017

Abstract

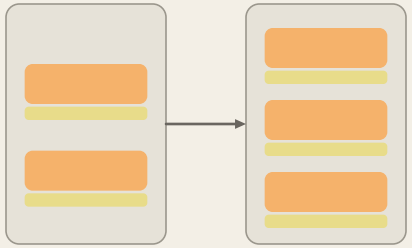
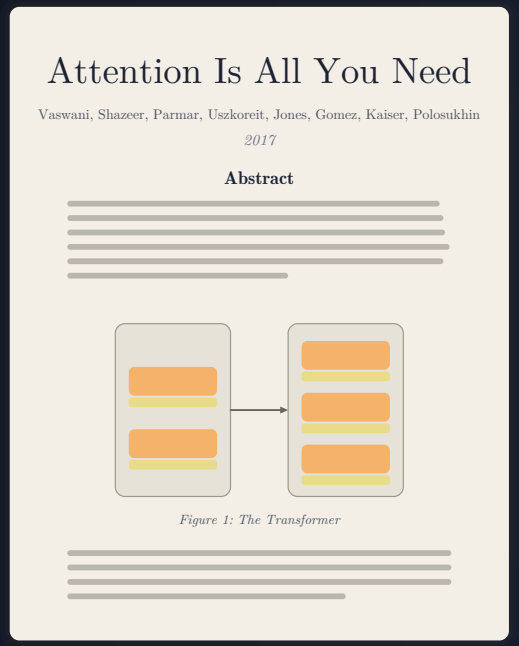


Figure 1: The Transformer

2 Before attention



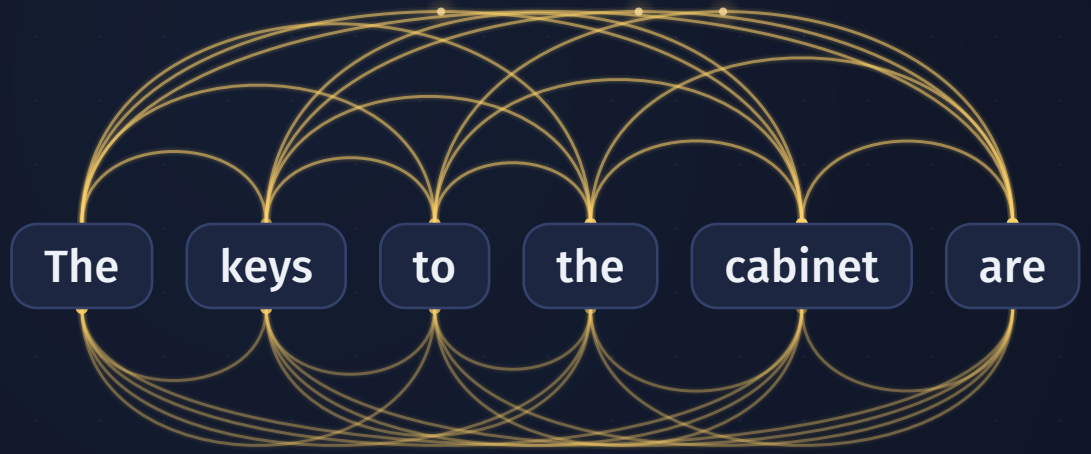
Attention Is All You Need

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin

2017

Abstract

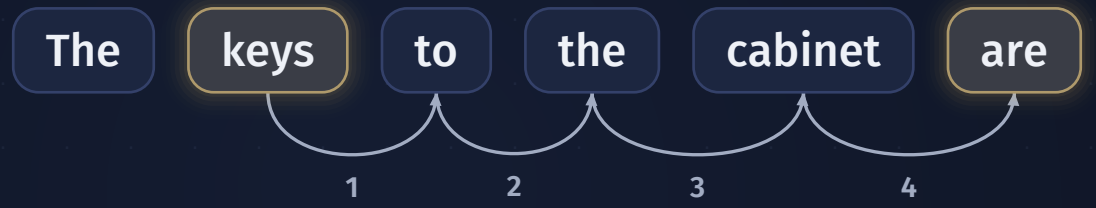
Figure 1: The Transformer



every word, at the same time

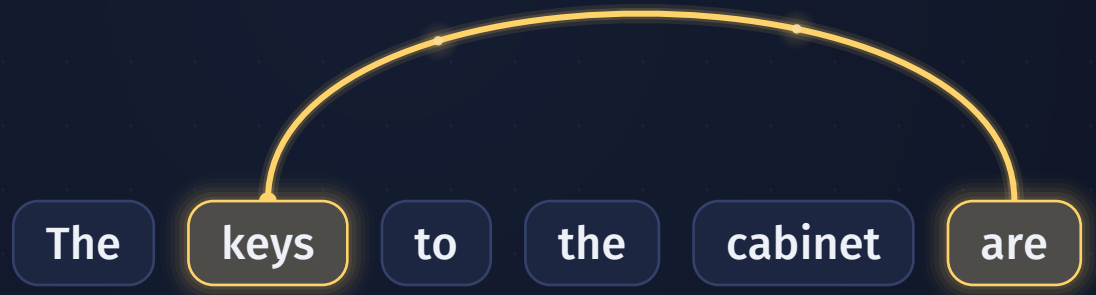
Recurrent network

4 steps from keys to are



Attention

1 step, directly

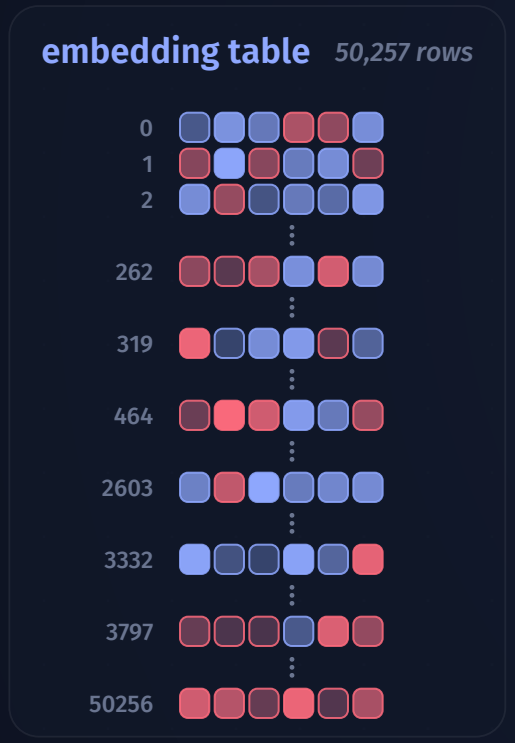


3 Words as vectors

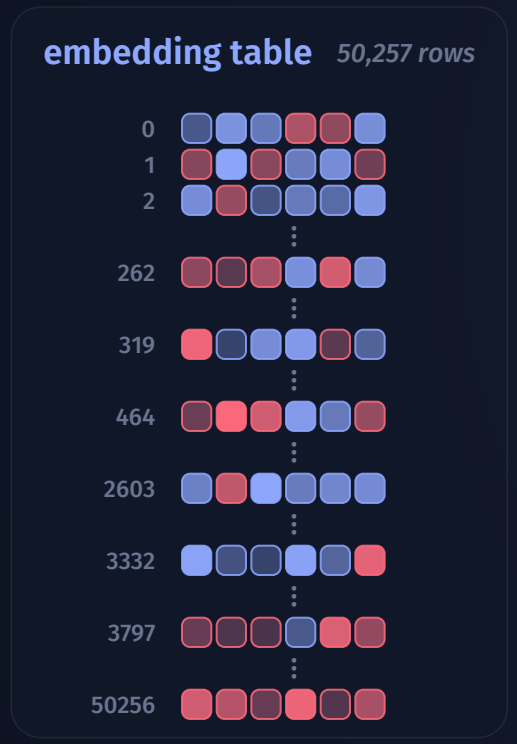
How text becomes numbers

	The	cat	sat	on	the	mat
token ID	464	3797	3332	319	262	2603

3 Words as vectors

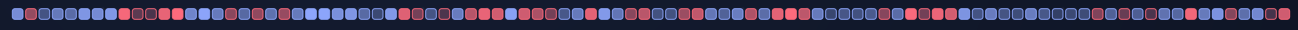








A real token vector is much longer:



GPT-2 small: 768 numbers

GPT-3: 12,288 numbers

3 Words as vectors



3 Words as vectors

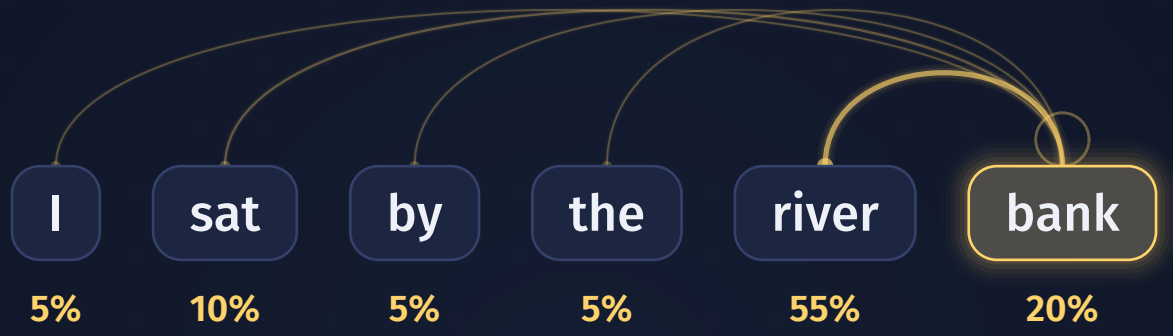


3 Words as vectors

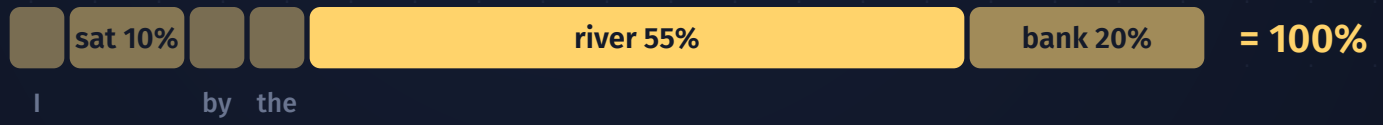


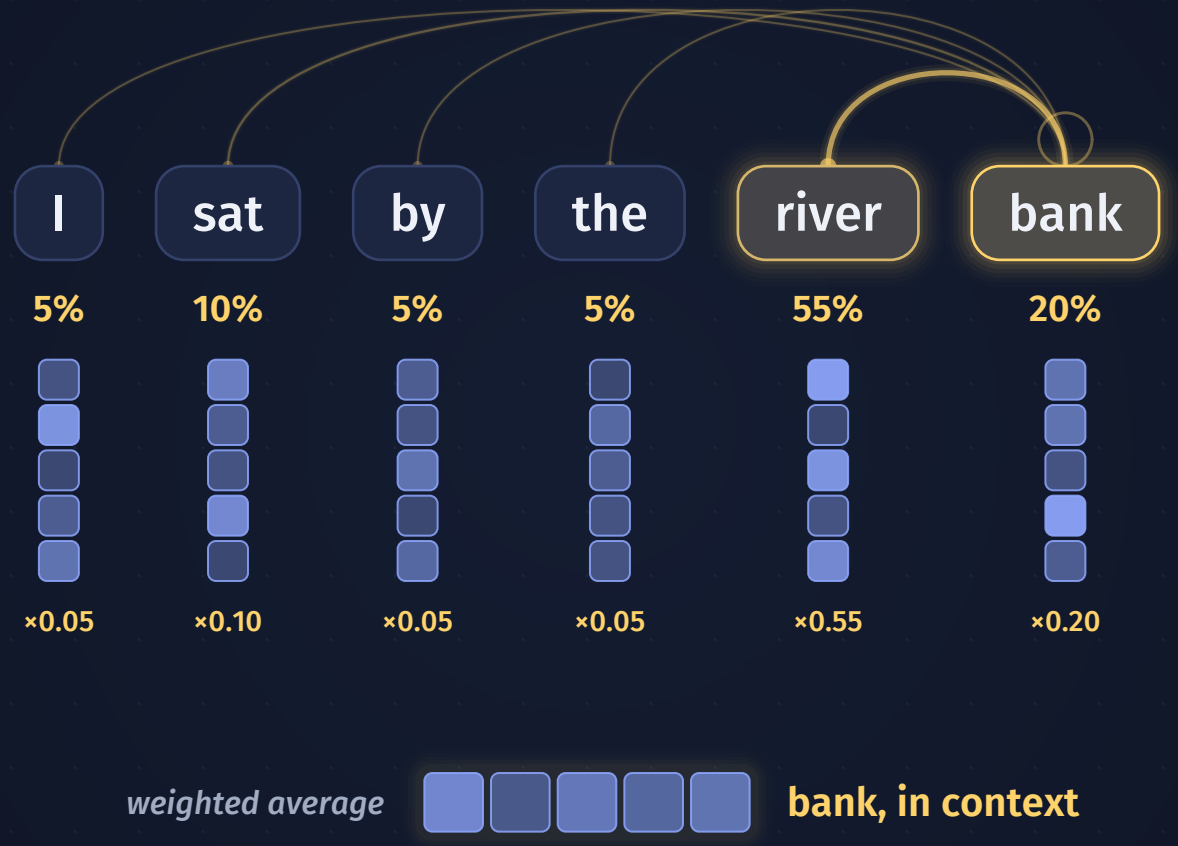
4 The big idea

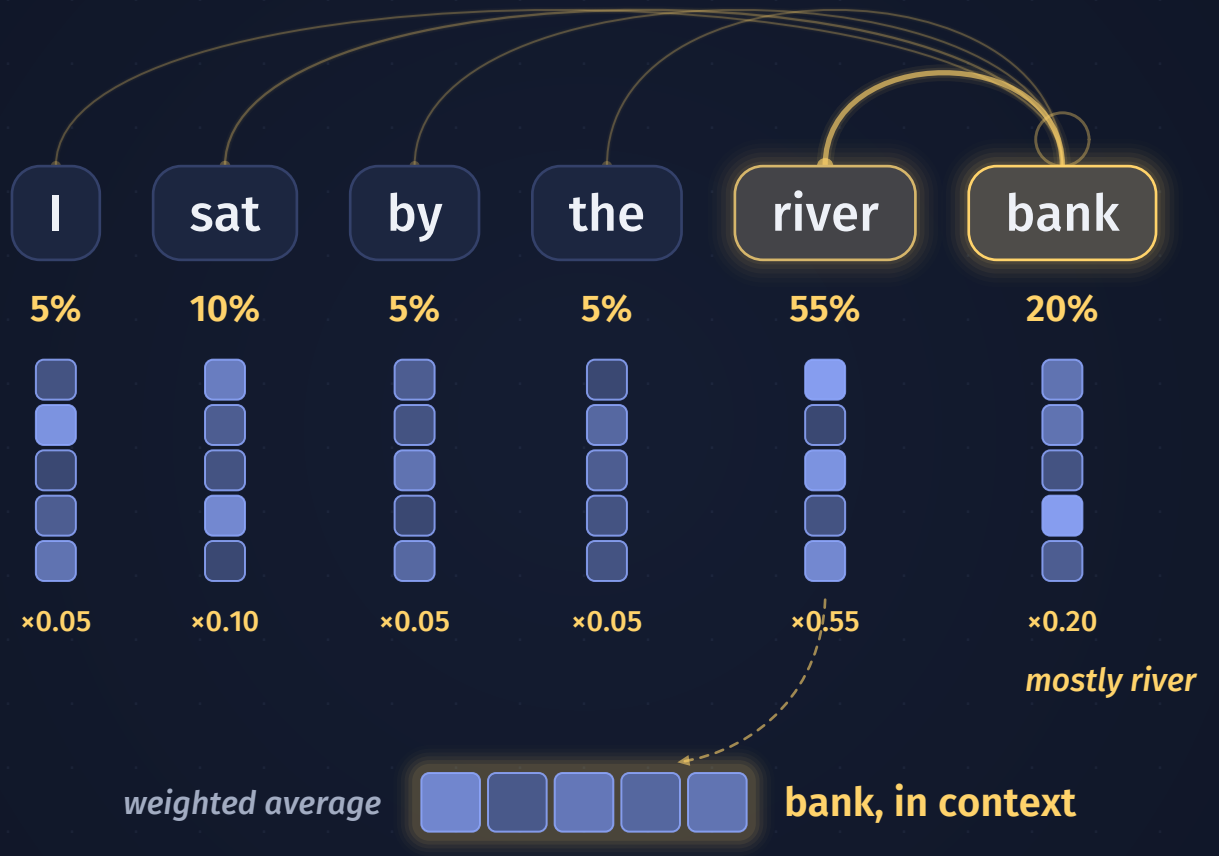
A weighted average of context

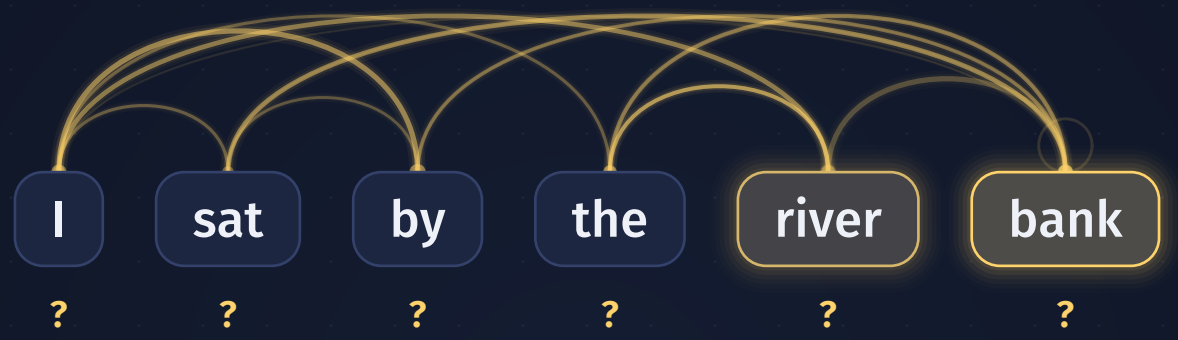

I
sat
by
the
river
bank
5%
10%
5%
5%
55%
20%


I
sat
by
the
river
bank
5%
10%
5%
5%
55%
20%


sat 10%
river 55%
bank 20%
= 100%
I
by
the


I
sat
by
the
river
bank
5%
10%
5%
5%
55%
20%
×0.05
×0.10
×0.05
×0.05
×0.55
×0.20
weighted average
bank, in context


I
sat
by
the
river
bank
5%
10%
5%
5%
55%
20%
×0.05
×0.10
×0.05
×0.05
×0.55
×0.20
mostly river
weighted average
bank, in context

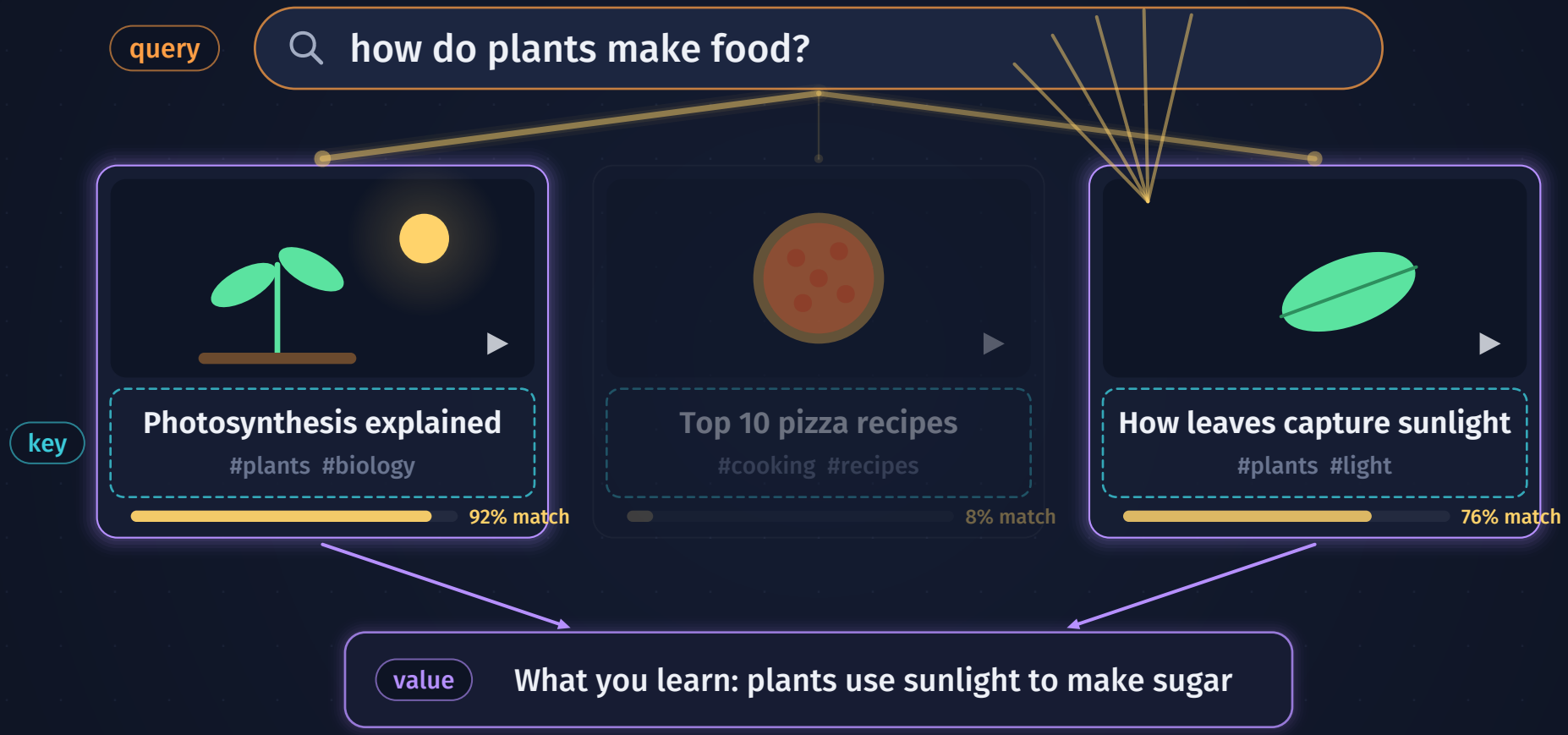

I
sat
by
the
river
bank
?
?
?
?
?
?

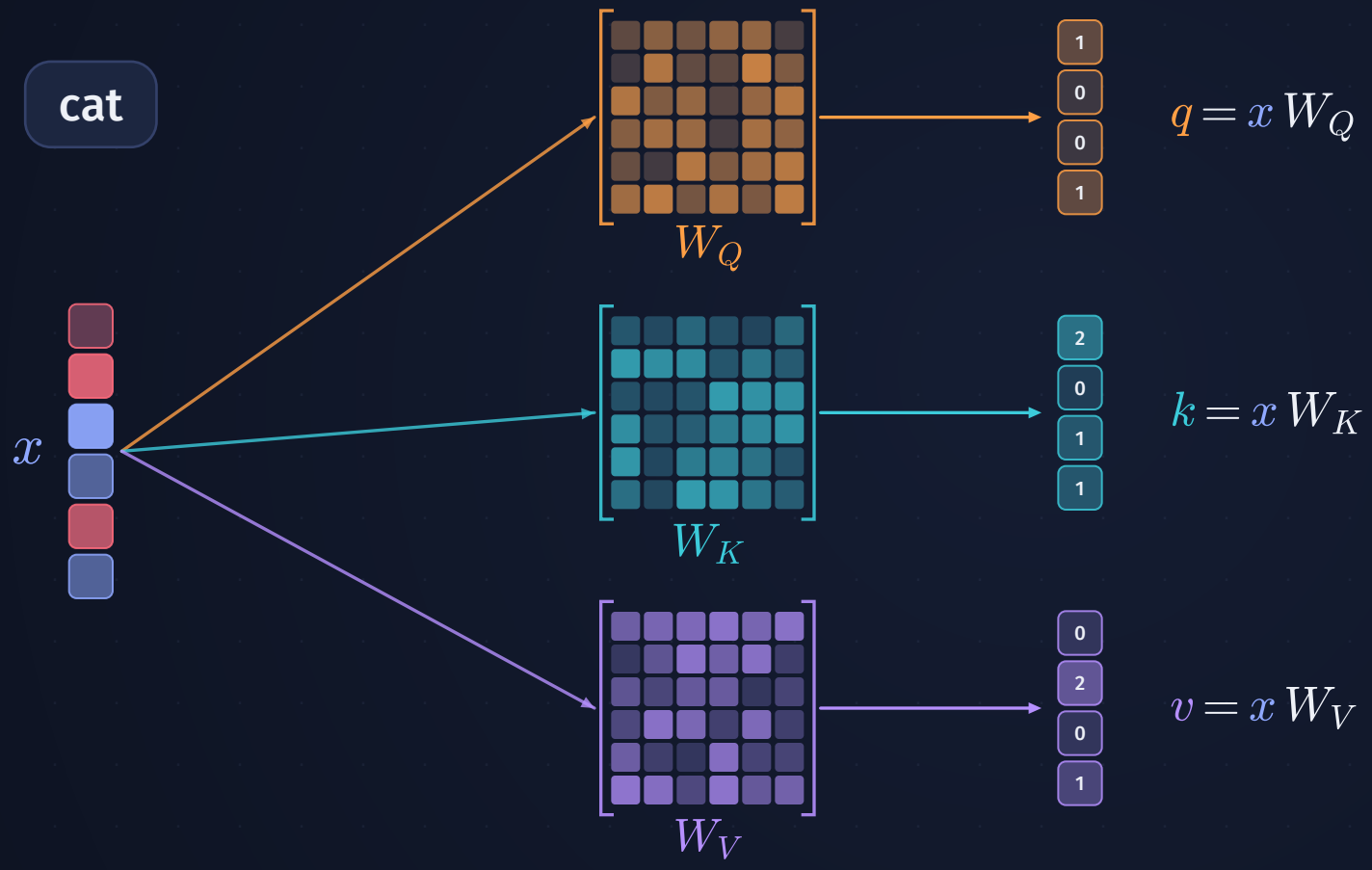
5 Query, key and value

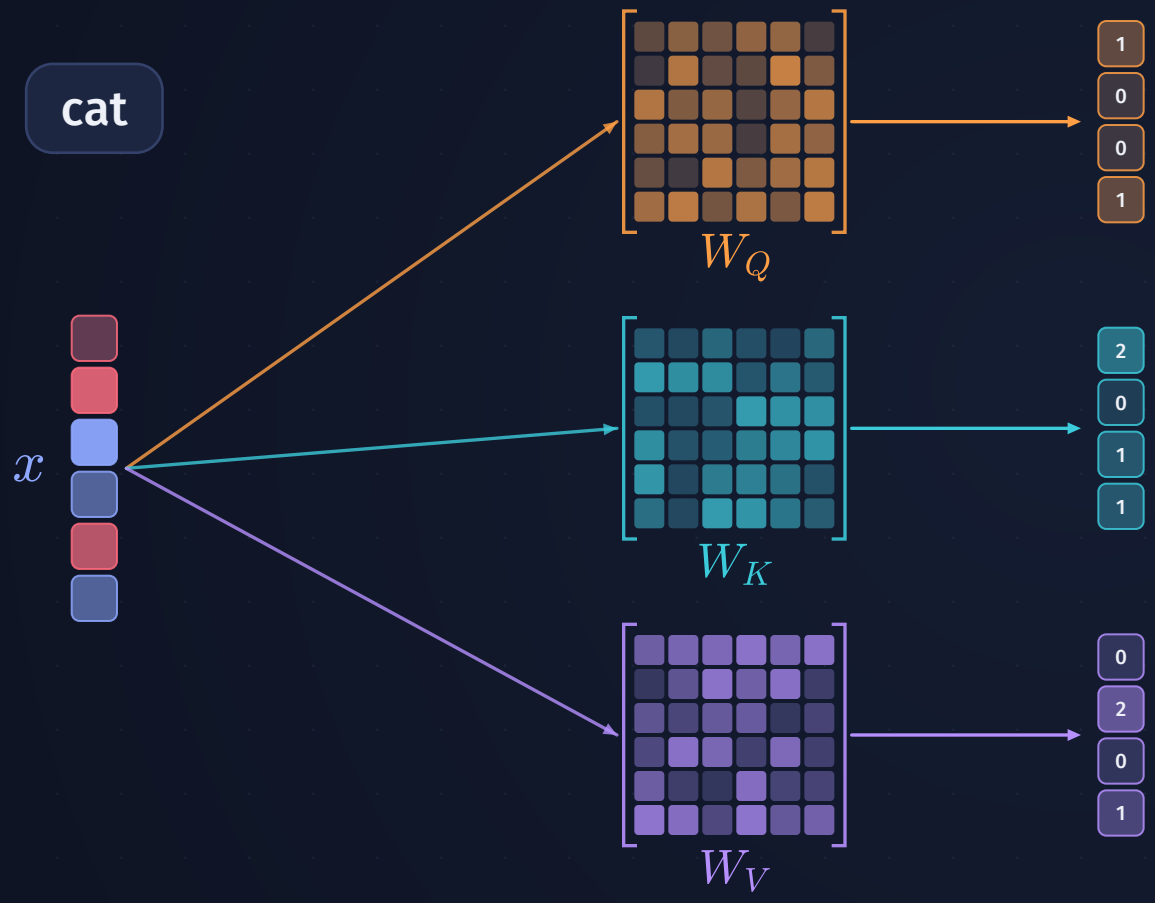
Three roles for every token


query
how do plants make food?


query
how do plants make food?
key
Photosynthesis explained
#plants #biology
92% match
Top 10 pizza recipes
#cooking #recipes
8% match
How leaves capture sunlight
#plants #light
76% match


query
how do plants make food?
key
Photosynthesis explained
#plants #biology
92% match
Top 10 pizza recipes
#cooking #recipes
8% match
How leaves capture sunlight
#plants #light
76% match
value
What you learn: plants use sunlight to make sugar

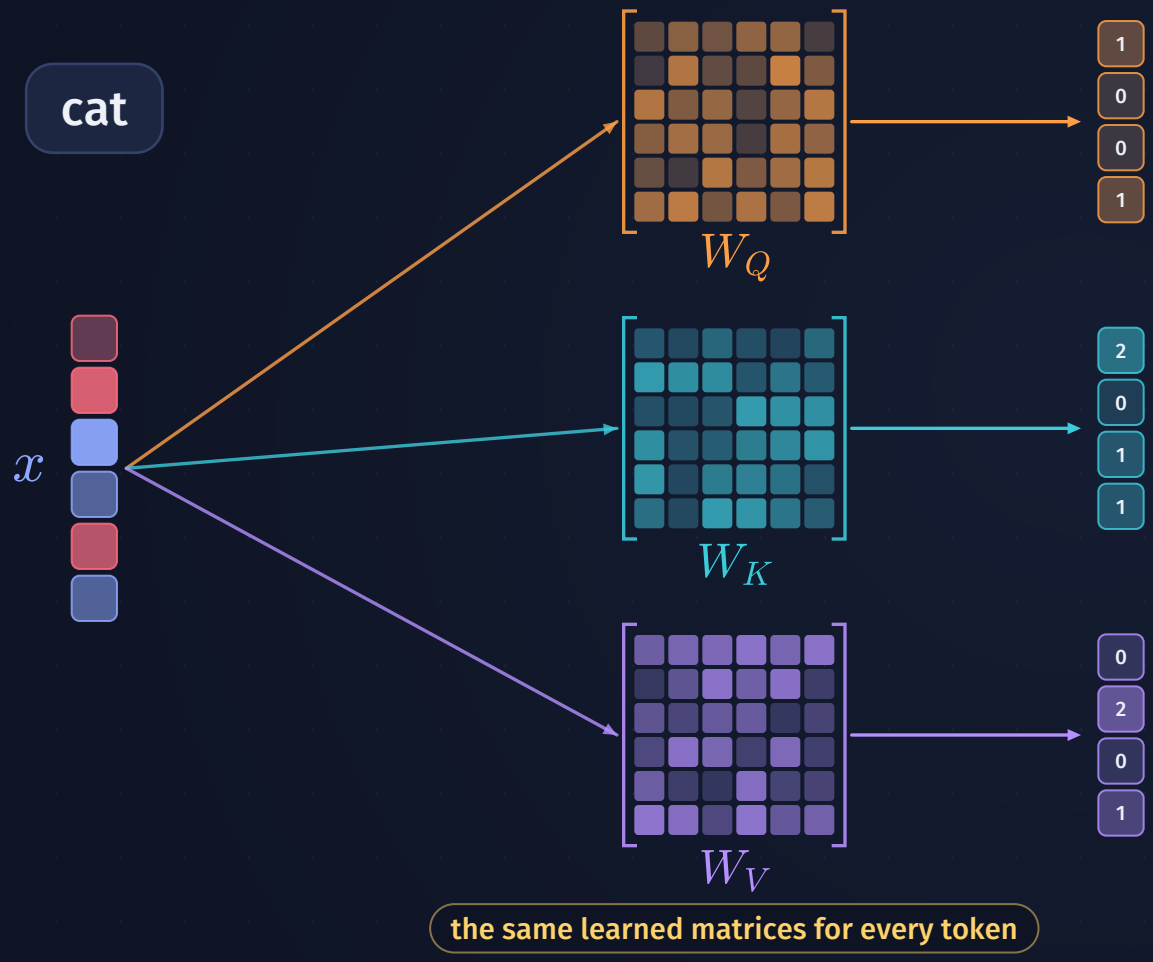

cat
x
W_Q
W_K
W_V
1 0 0 1
2 0 1 1
0 2 0 1
$q = x\,W_Q$
$k = x\,W_K$
$v = x\,W_V$



Query
What am I looking for?

Key
What do I contain?

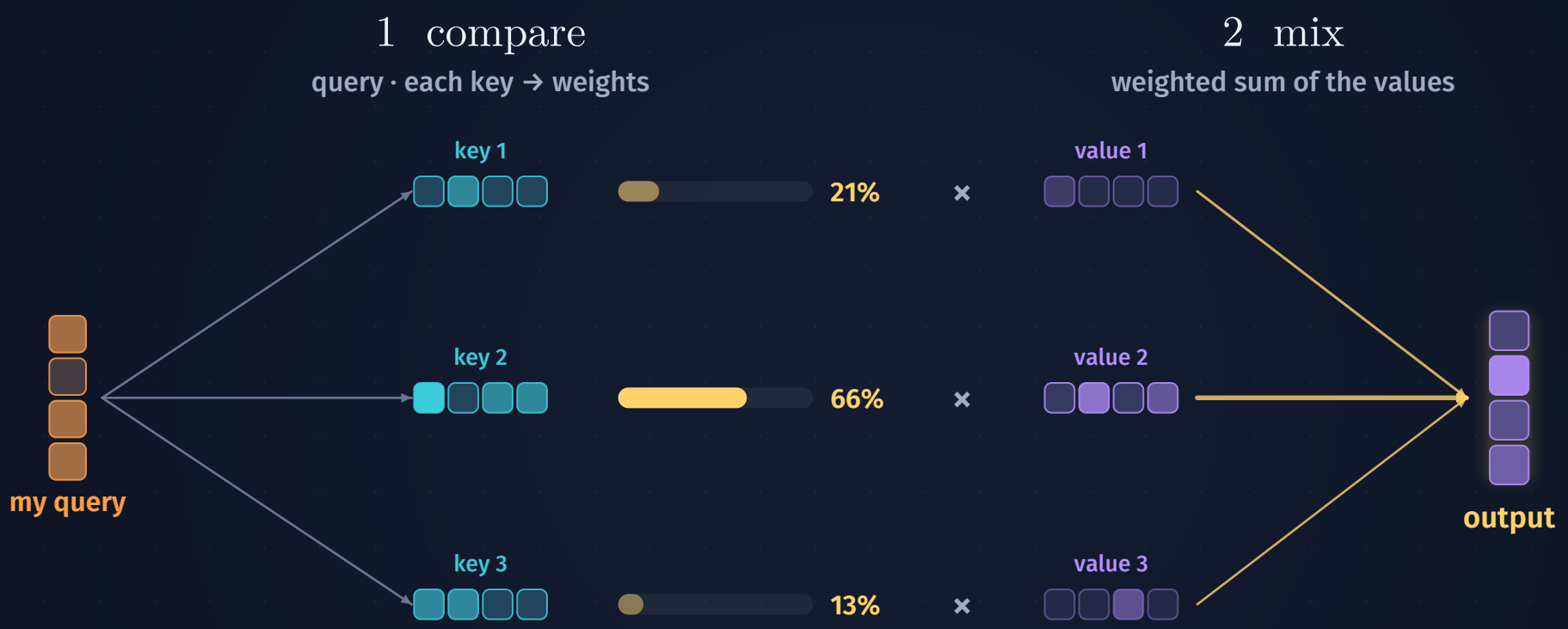
Value
What will I share?



Query
What am I looking for?

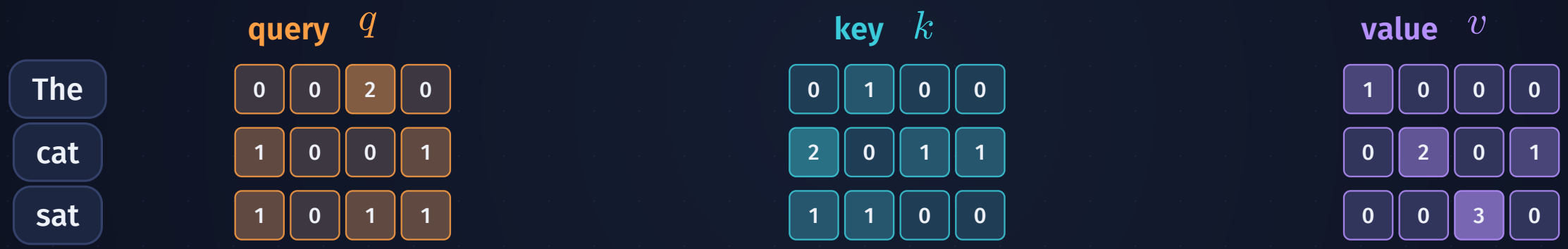
Key
What do I contain?

Value
What will I share?

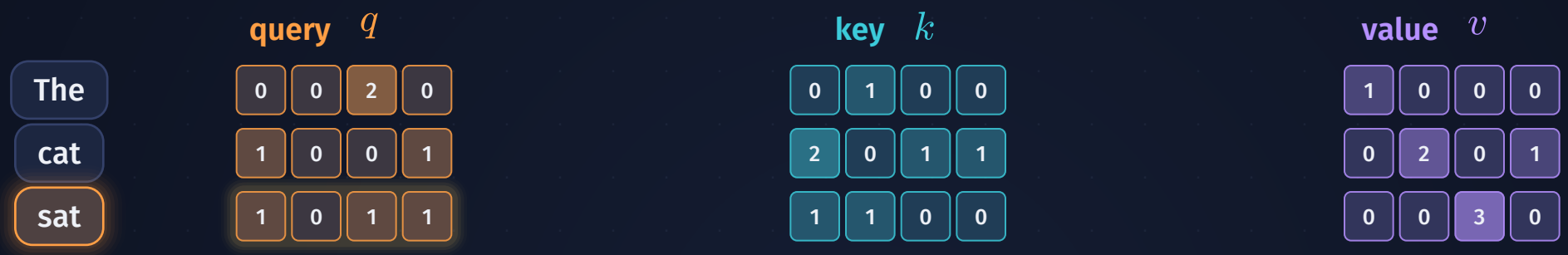

1 compare
query · each key → weights
2 mix
weighted sum of the values
key 1
key 2
key 3
21%
66%
13%
value 1
value 2
value 3
my query
output

6 The math, step by step

Computing attention by hand


query q
key k
value v
The
cat
sat
0 0 2 0
1 0 0 1
1 0 1 1
0 1 0 0
2 0 1 1
1 1 0 0
1 0 0 0
0 2 0 1
0 0 3 0

6 The math, step by step



	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 1 Dot product

q (sat)	1	0	1	1	
	×	×	×	×	
k (cat)	2	0	1	1	
	2	+ 0	+ 1	+ 1	$= 4$

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 1 Dot product

q (sat)	1	0	1	1	
	×	×	×	×	
k (cat)	2	0	1	1	
	2	+ 0 +	1	+ 1	$= 4$

1×2 + 0×0 + 1×1 + 1×1 = 4

multiply pairs, then add

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0



$$\mathrm{q} \cdot \mathrm{k} = 2.0$$

negative — positive

same direction

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0



$$\mathrm{q} \cdot \mathrm{k} = -2.0$$

negative — positive

pointing apart

6 The math, step by step

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

$$q_{\text{sat}} \cdot k_{\text{The}} = 0$$

$$q_{\text{sat}} \cdot k_{\text{cat}} = 4$$

$$q_{\text{sat}} \cdot k_{\text{sat}} = 1$$

	The	cat	sat
scores	0	4	1

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 2 Scale

$$\frac{q \cdot k}{\sqrt{d_k}}$$

key length 4, so √4 = 2

	The	cat	sat
scores	0	2	0.5

÷ 2

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Why scale?



	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Why scale?



	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 3 Softmax

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

	The	cat	sat
score	0	2	0.5
e^{score}	1.00	7.39	1.65

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 3 Softmax

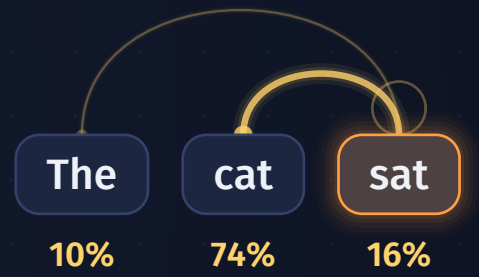
$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

	The	cat	sat	
score	0	2	0.5	
e^{score}	1.00	7.39	1.65	sum = 1.00 + 7.39 + 1.65 = 10.04
weight	0.10	0.74	0.16	*divide each by 10.04* → adds up to 1

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 3 Softmax

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



	The	cat	sat	
score	0	2	0.5	
e^{score}	1.00	7.39	1.65	sum = 1.00 + 7.39 + 1.65 = 10.04
weight	0.10	0.74	0.16	*divide each by 10.04* → adds up to 1

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 4 Weighted sum of values

0.10 × v (The)		0.74 × v (cat)		0.16 × v (sat)		new vector for sat
0.10	+	0.00	+	0.00	=	0.10
0.00		1.48		0.00		1.48
0.00		0.00		0.48		0.48
0.00		0.74		0.00		0.74

	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

Step 4 Weighted sum of values

0.10 × v (The)		0.74 × v (cat)		0.16 × v (sat)		new vector for sat	
0.10	+	0.00	+	0.00	=	0.10	
0.00		1.48		0.00		1.48	*mostly cat's information*
0.00		0.00		0.48		0.48	
0.00		0.74		0.00		0.74	

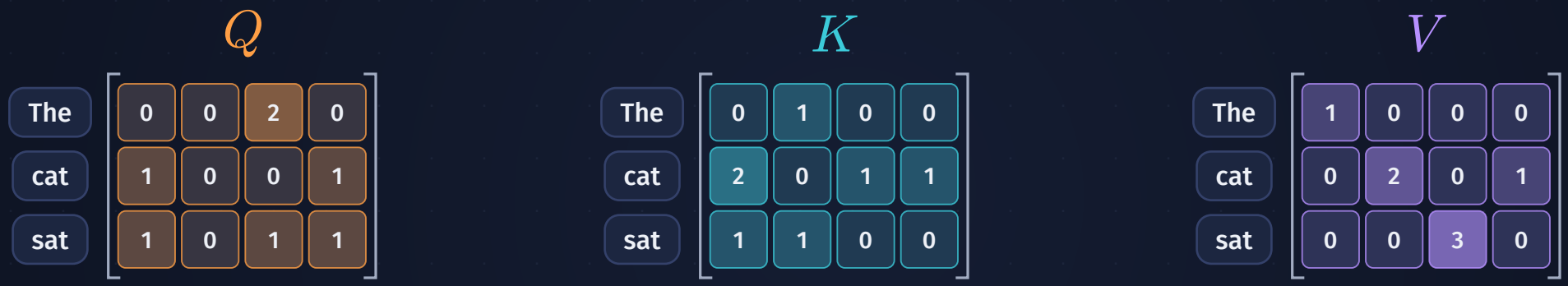
	query q	key k	value v
The	0 0 2 0	0 1 0 0	1 0 0 0
cat	1 0 0 1	2 0 1 1	0 2 0 1
sat	1 0 1 1	1 1 0 0	0 0 3 0

score $q \cdot k$ → scale $\div \sqrt{d_k}$ → softmax $\rightarrow$ weights → weighted sum $\sum_j w_j v_j$

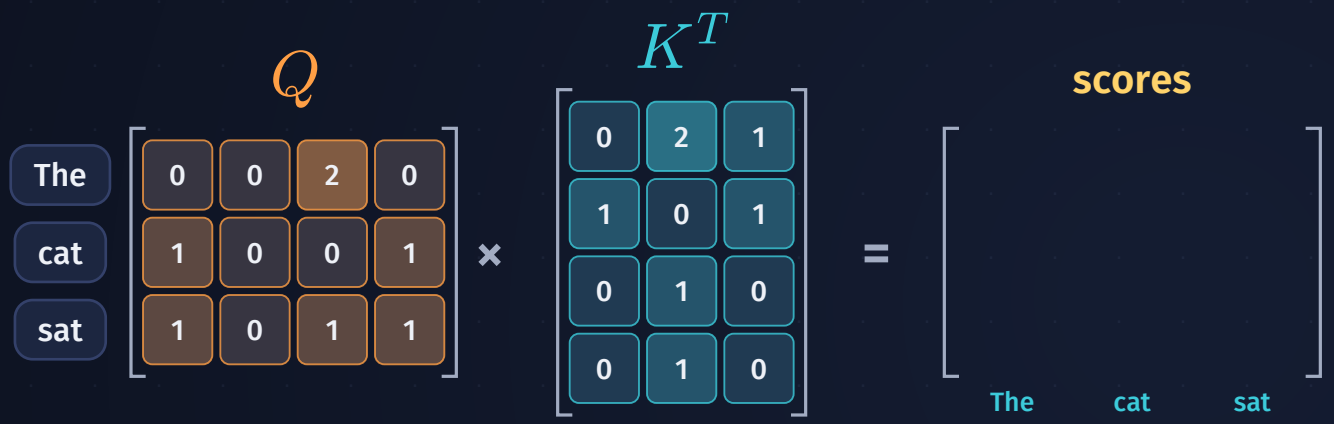
7 All at once: the matrix form

Every token in parallel

7 All at once: the matrix form



7 All at once: the matrix form

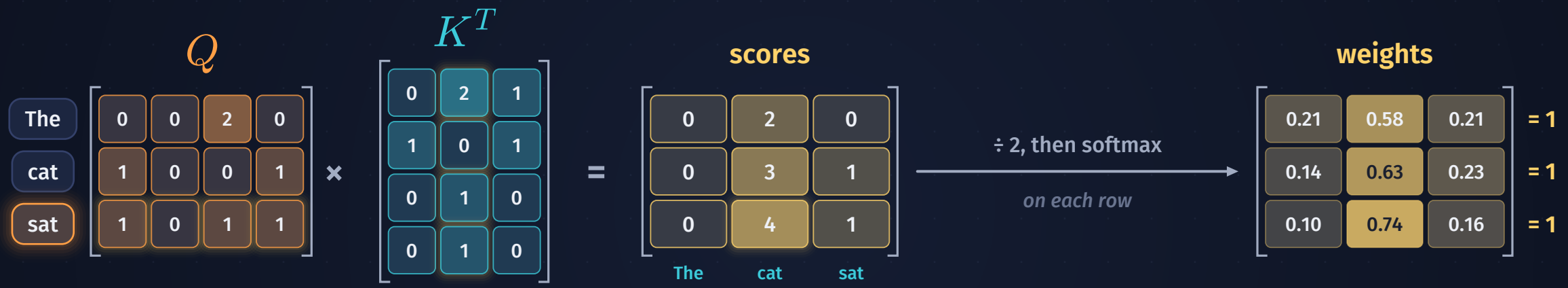


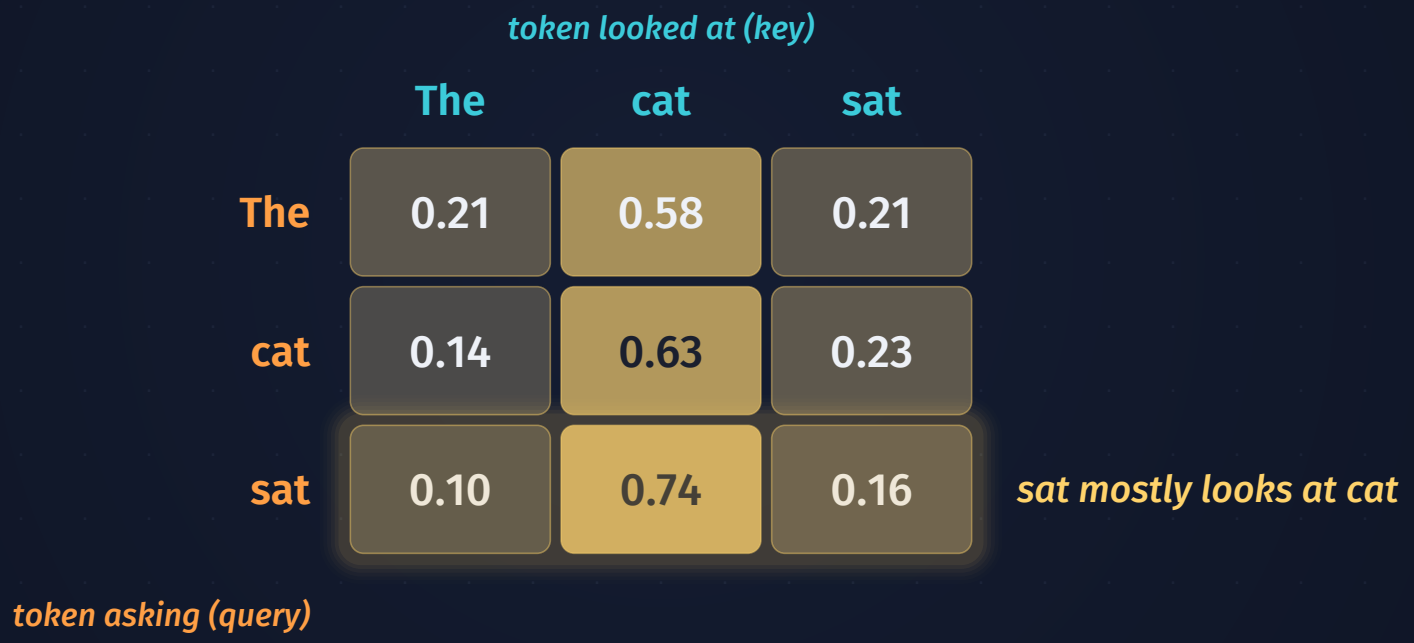
7 All at once: the matrix form

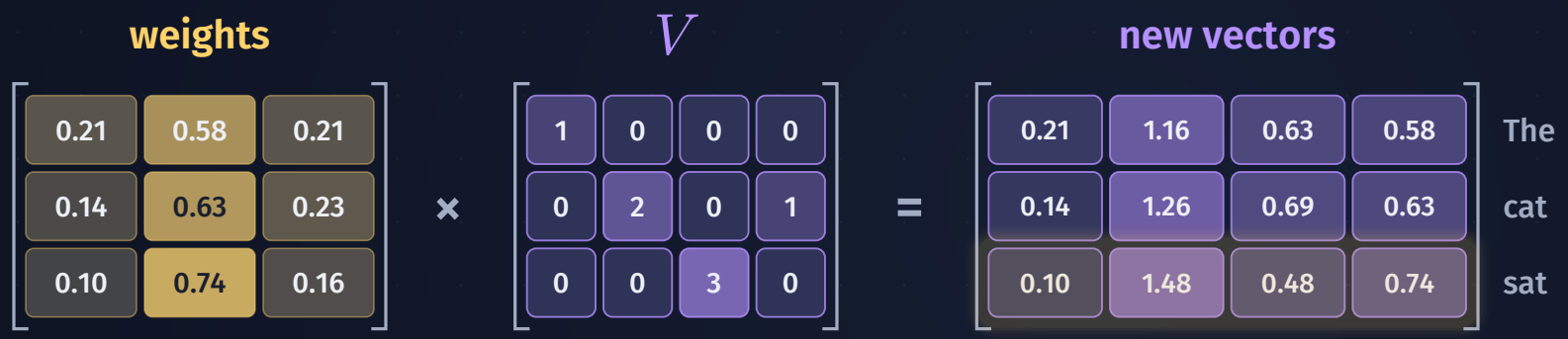


$$q_{\text{sat}} \cdot k_{\text{cat}} = 4$$

7 All at once: the matrix form




token looked at (key)
The
cat
sat
The
0.21
0.58
0.21
cat
0.14
0.63
0.23
sat
0.10
0.74
0.16
sat mostly looks at cat
token asking (query)



row for sat: 0.10, 1.48, 0.48, 0.74

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

every query–key score

rows become weights

keeps scores gentle

mix the values

“the river bank flooded”

	the	river	bank	flooded
the	0.40	0.30	0.20	0.10
river	0.15	0.45	0.30	0.10
bank	0.08	0.62	0.20	0.10
flooded	0.05	0.30	0.45	0.20

“the bank approved loans”

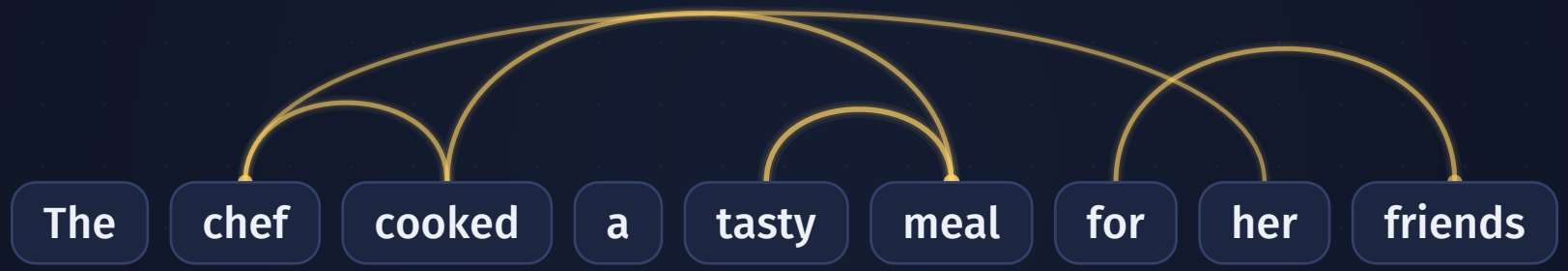
	the	bank	approved	loans
the	0.40	0.35	0.15	0.10
bank	0.10	0.25	0.25	0.40
approved	0.05	0.40	0.25	0.30
loans	0.05	0.30	0.40	0.25

same word “bank”, different weights

illustrative values

8 Multi-head attention

Many kinds of relationships at once



one head, one pattern

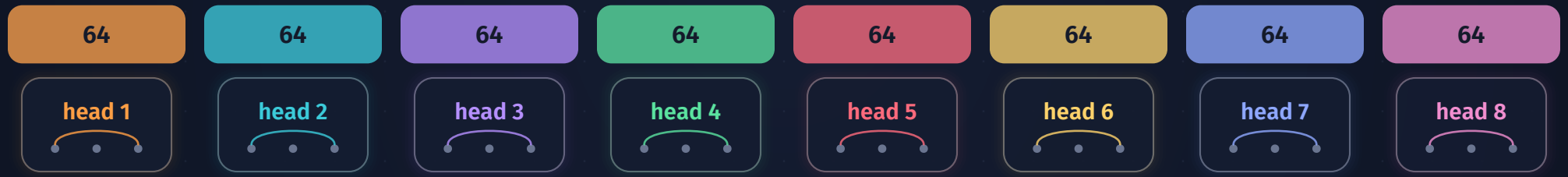

head 1
The chef cooked a tasty meal for her friends
head 2
The chef cooked a tasty meal for her friends
head 3
The chef cooked a tasty meal for her friends
head 4
The chef cooked a tasty meal for her friends


head 1
The chef cooked a tasty meal for her friends
previous word
head 2
The chef cooked a tasty meal for her friends
who did it
head 3
The chef cooked a tasty meal for her friends
head 4
The chef cooked a tasty meal for her friends


head 1
The chef cooked a tasty meal for her friends
previous word
head 2
The chef cooked a tasty meal for her friends
who did it
head 3
The chef cooked a tasty meal for her friends
what was made
head 4
The chef cooked a tasty meal for her friends
for whom


head 1
The chef cooked a tasty meal for her friends
previous word
head 2
The chef cooked a tasty meal for her friends
who did it
head 3
The chef cooked a tasty meal for her friends
what was made
head 4
The chef cooked a tasty meal for her friends
for whom
simplified labels

8 Multi-head attention



8 Multi-head attention



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O$$

$$\text{head}_i = \text{Attention}(Q\, W_i^Q,\ K\, W_i^K,\ V W_i^V)$$

h = number of heads (8 in the original Transformer)

9 Positional encoding

Why word order needs help

dog bites man — not news

man bites dog — headline!

dog bites man — not news

man bites dog — headline!

$$\text{score} = q \cdot k$$

no position information anywhere

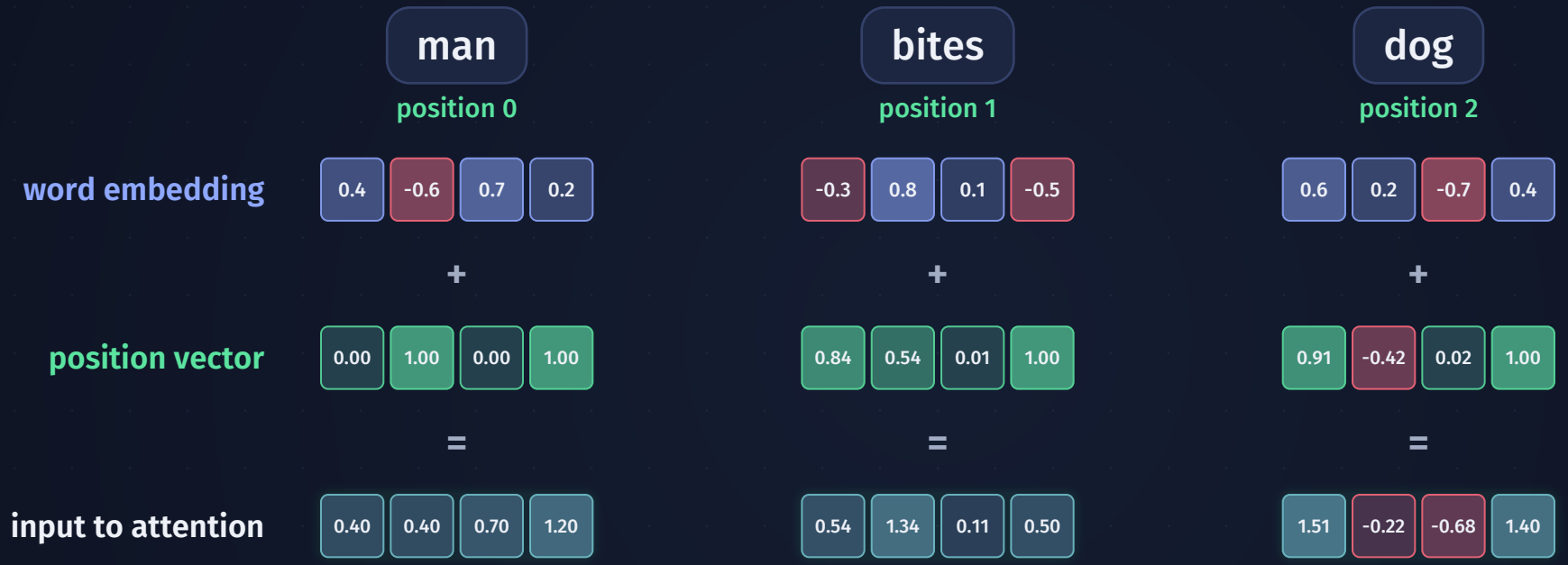


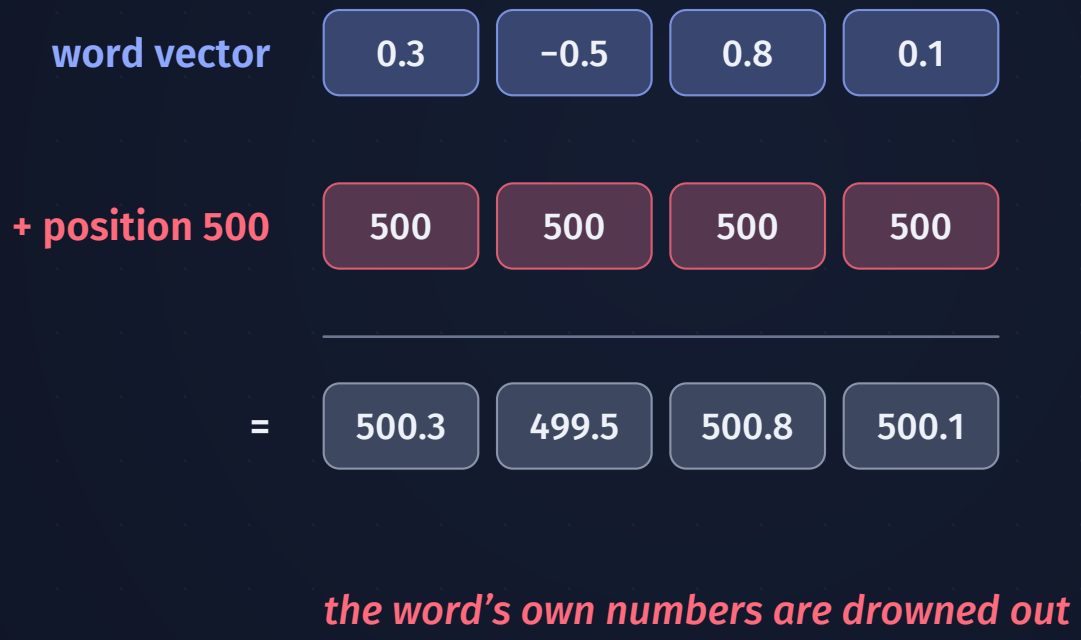
identical weights

both sentences end up the same

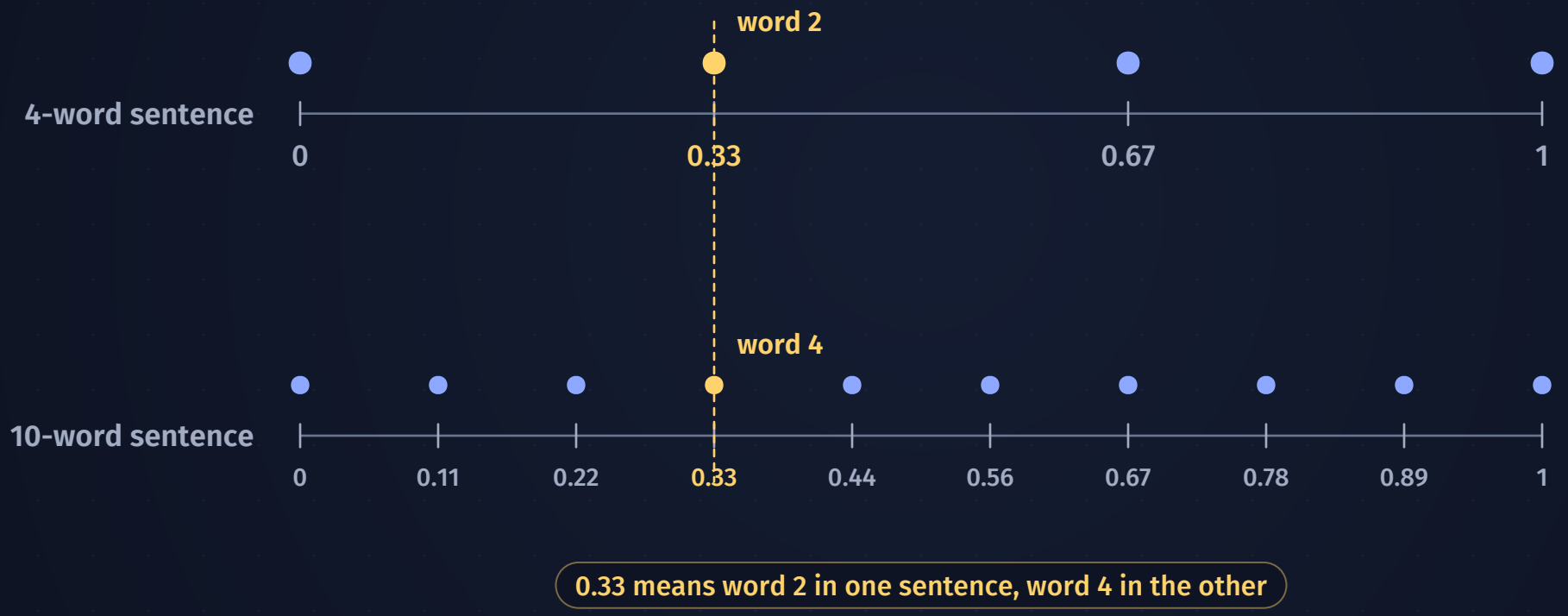


bag of words


man
bites
dog
position 0
position 1
position 2
word embedding
0.4 -0.6 0.7 0.2
-0.3 0.8 0.1 -0.5
0.6 0.2 -0.7 0.4
+
position vector
0.00 1.00 0.00 1.00
0.84 0.54 0.01 1.00
0.91 -0.42 0.02 1.00
=
input to attention
0.40 0.40 0.70 1.20
0.54 1.34 0.11 0.50
1.51 -0.22 -0.68 1.40



the word's own numbers are drowned out



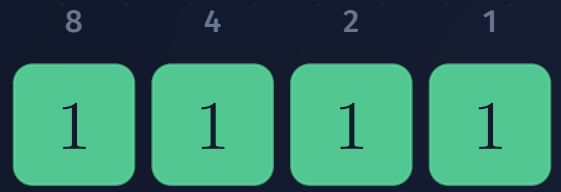
0.33 means word 2 in one sentence, word 4 in the other

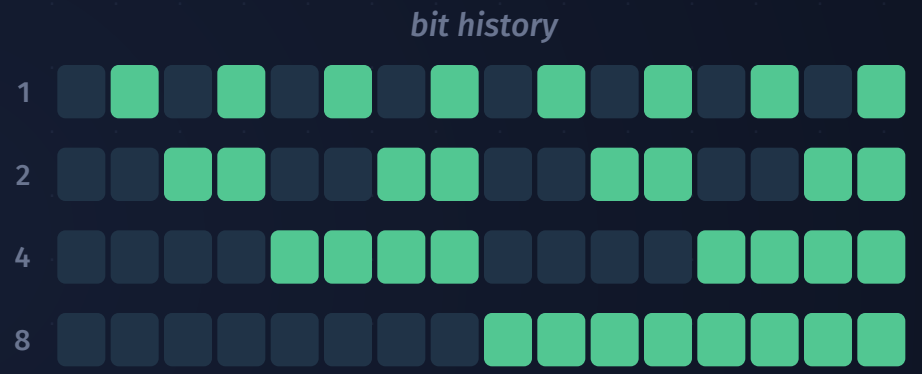
What we want from a position code

- ✓ a unique code for every position
- ✓ values stay in a small range
- ✓ works for sentences of any length
- ✓ relative distances are easy to detect

position

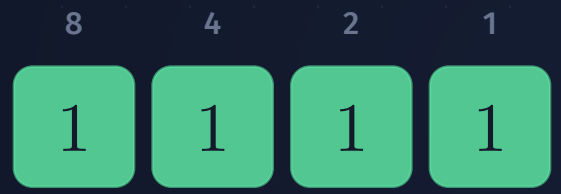
15

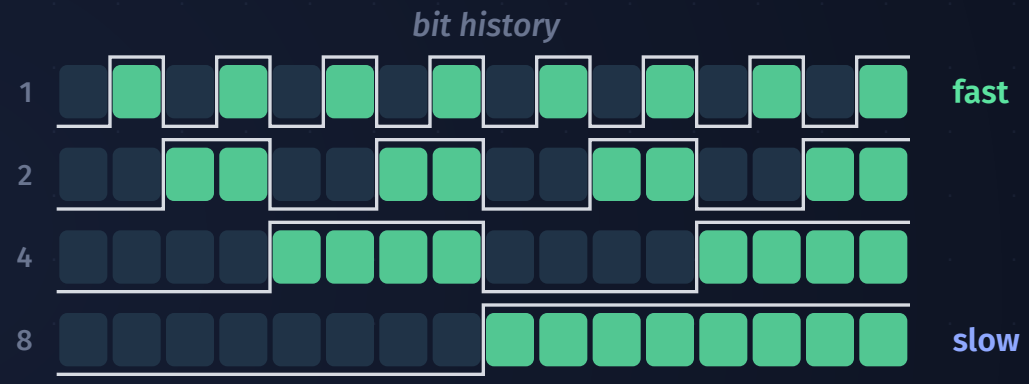




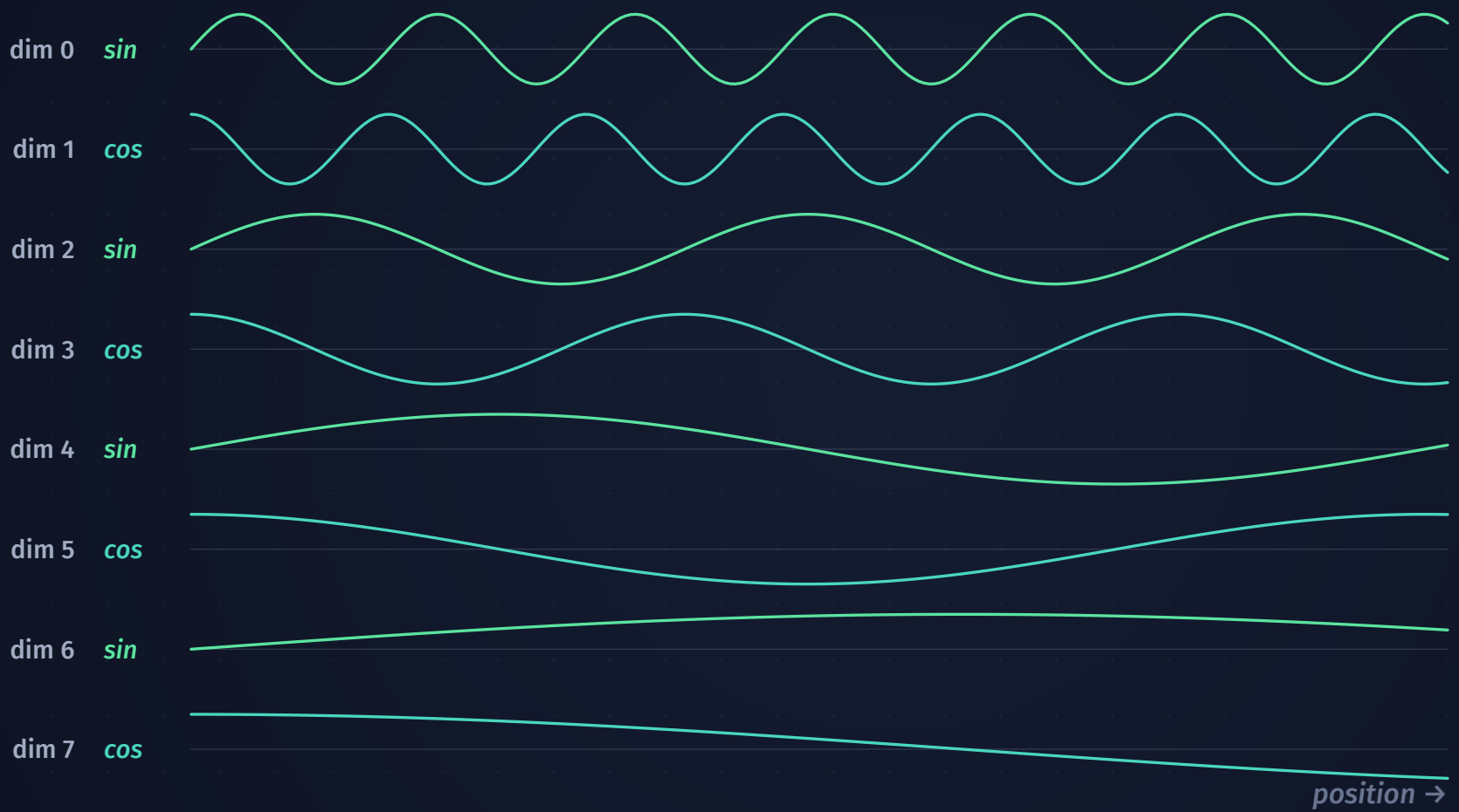
position

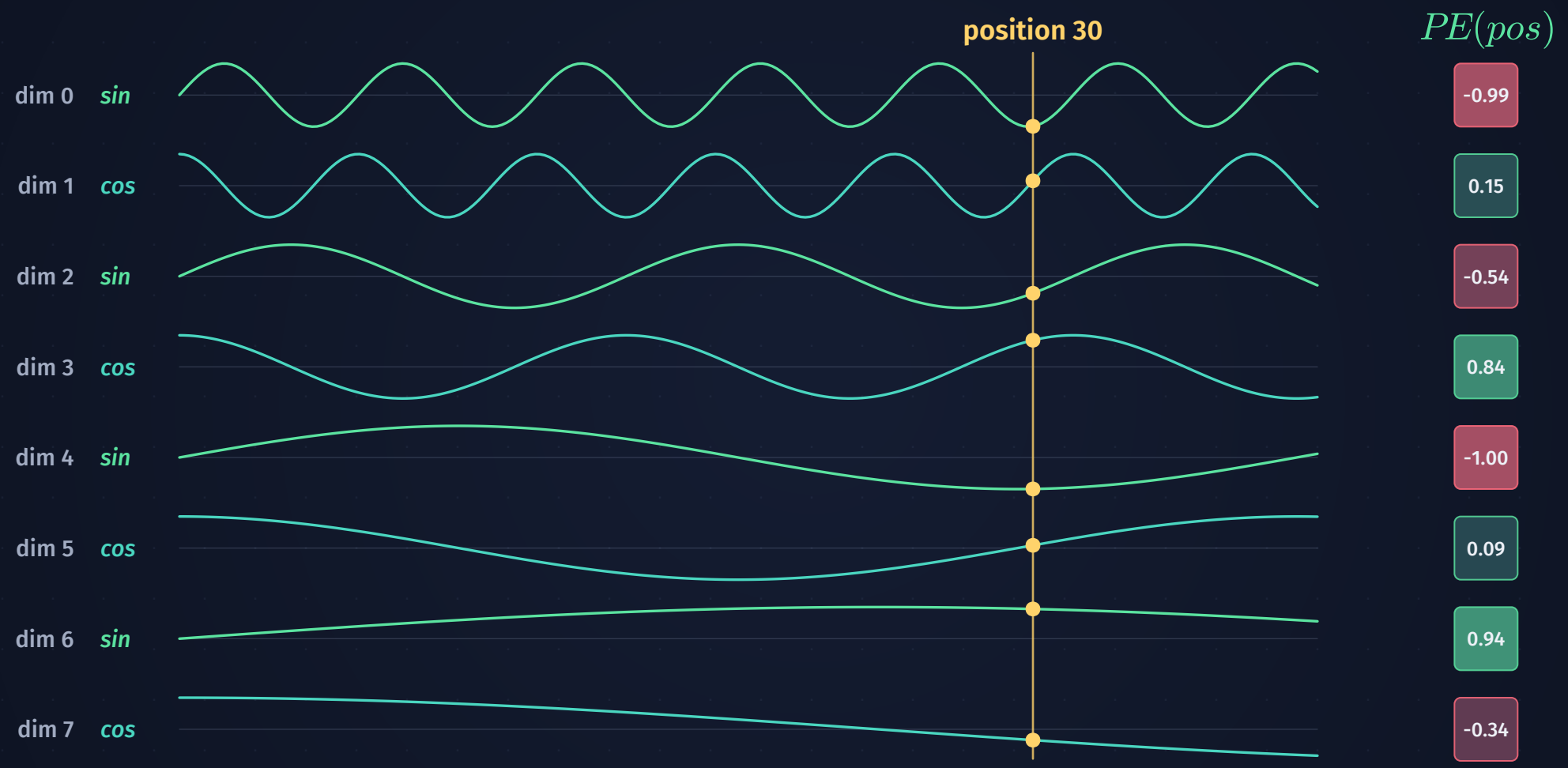
15





9 Positional encoding




position 30
$PE(pos)$
dim 0 sin
dim 1 cos
dim 2 sin
dim 3 cos
dim 4 sin
dim 5 cos
dim 6 sin
dim 7 cos
-0.99
0.15
-0.54
0.84
-1.00
0.09
0.94
-0.34

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

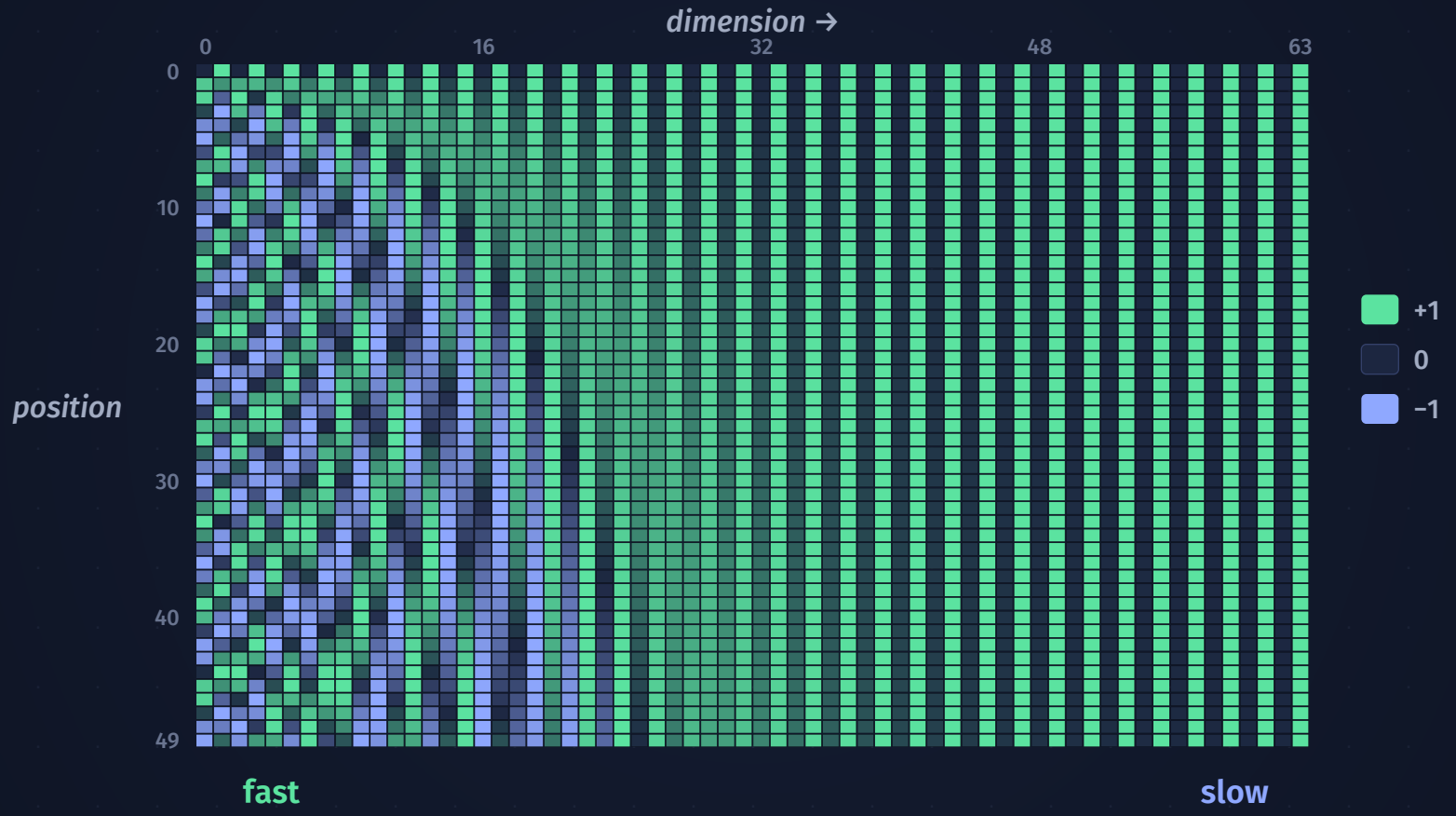
$$PE_{(pos,\ 2i+1)} = \cos\left(\frac{pos}{10000^{\,2i/d_{\text{model}}}}\right)$$

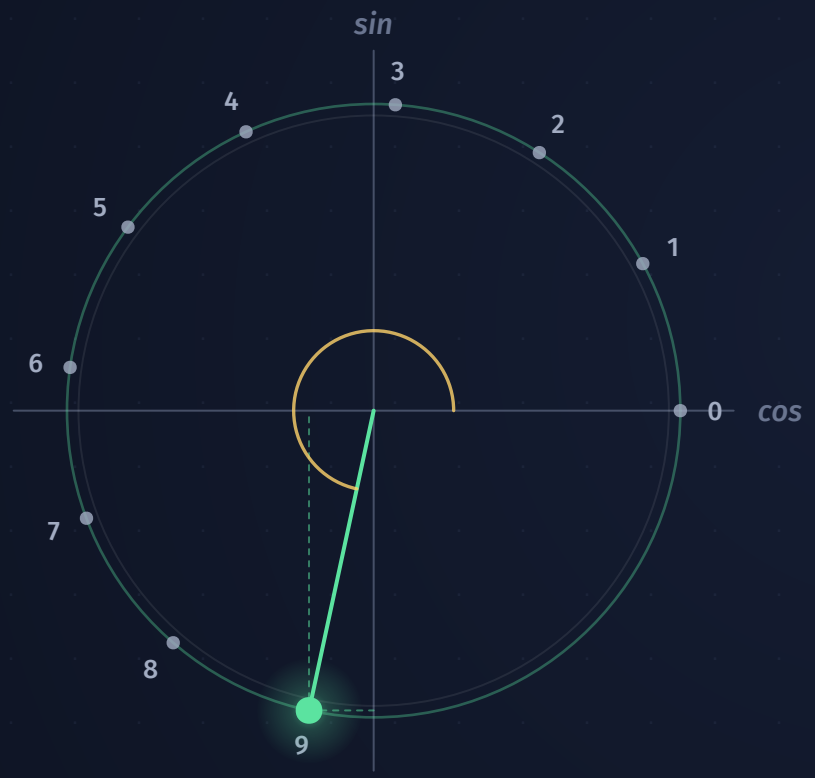
pos position of the token in the sentence

i which sine–cosine pair (0, 1, 2, ...)

d_{model} vector size: 512 in the original model

9 Positional encoding

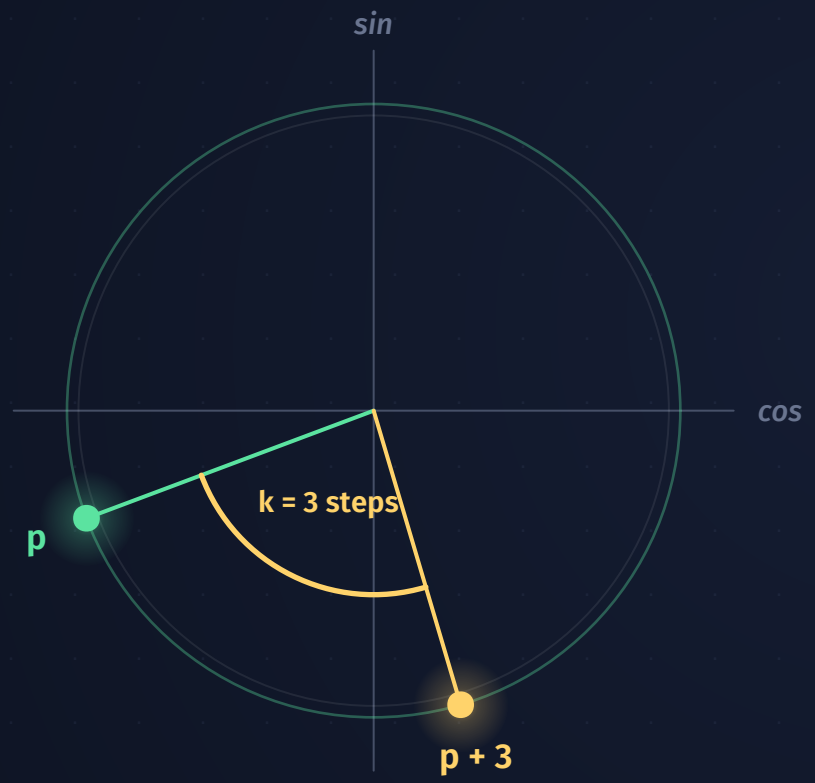




Position p → angle p·ω

One step forward → turn by the same ω

the dots show positions 0, 1, 2, …

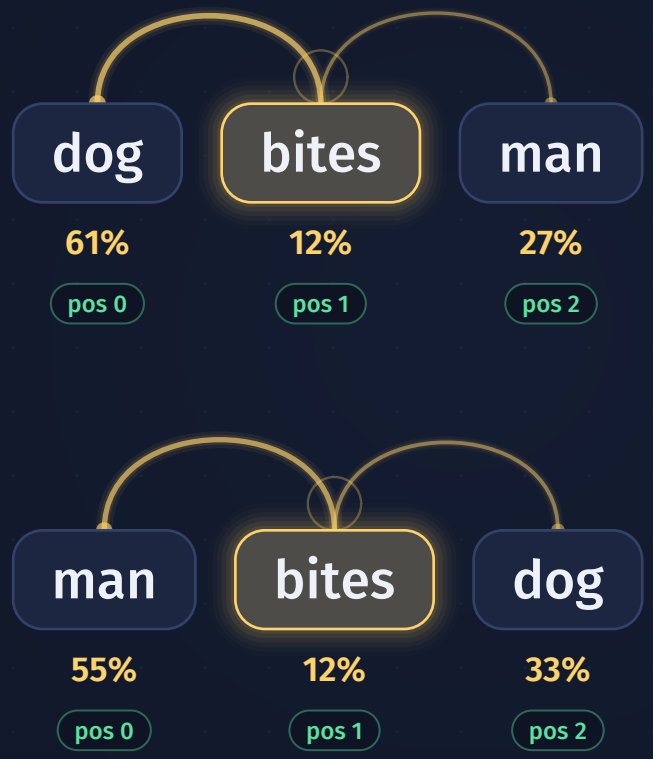


$$PE(p+k) = R_k \, PE(p)$$

The rotation depends only on the step k, not on the start p

move p along: the gap stays the same


similarity to position 20
itself
1.0
0.4
0
10
20
30
40
50
position



now different

illustrative weights



BERT: 512 positions

GPT-2: 1,024 positions

trained like word embeddings



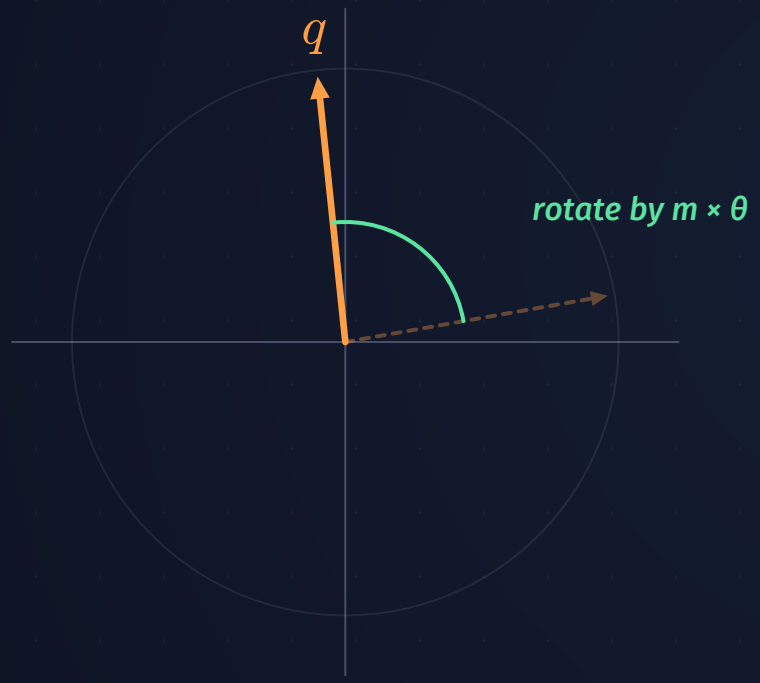
BERT: 512 positions

GPT-2: 1,024 positions

trained like word embeddings

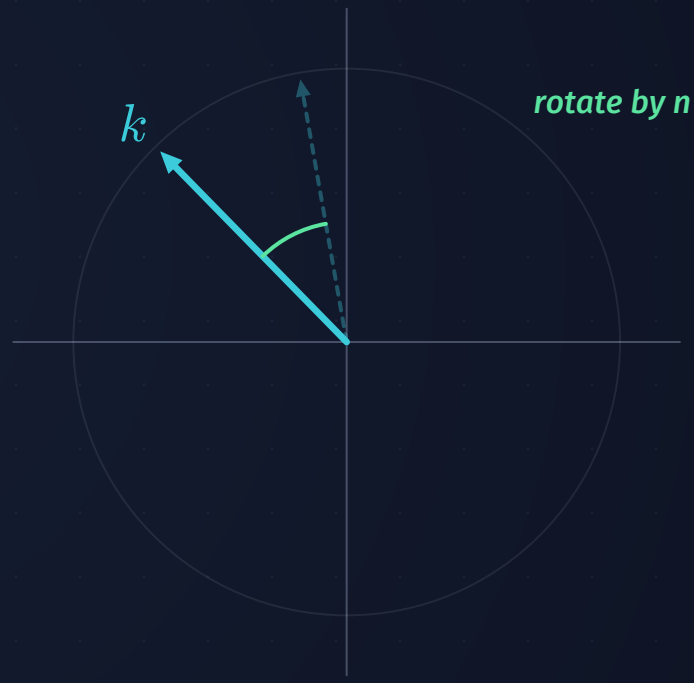
no vector for positions beyond training

query at position m

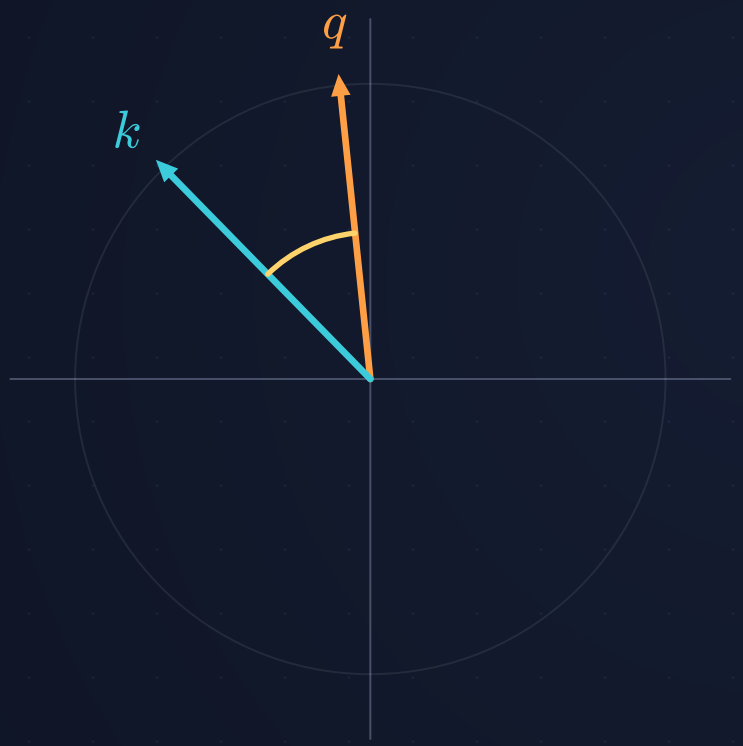


m = 5

key at position n



n = 2

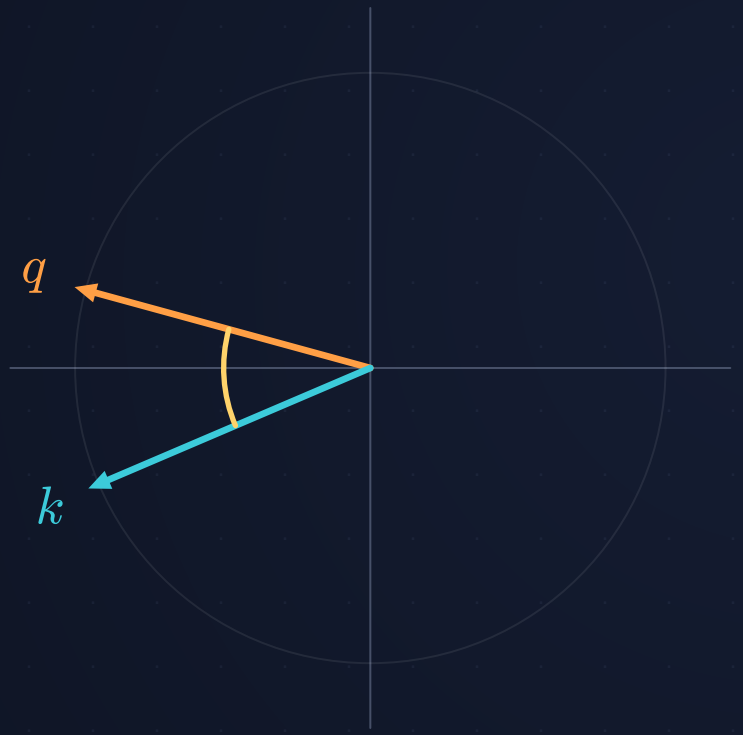


$$(R_m\, q) \cdot (R_n\, k) = q \cdot R_{n-m}\, k$$

m = 5 **n = 2**

m − n = 3

$$\text{score} = 1.57$$



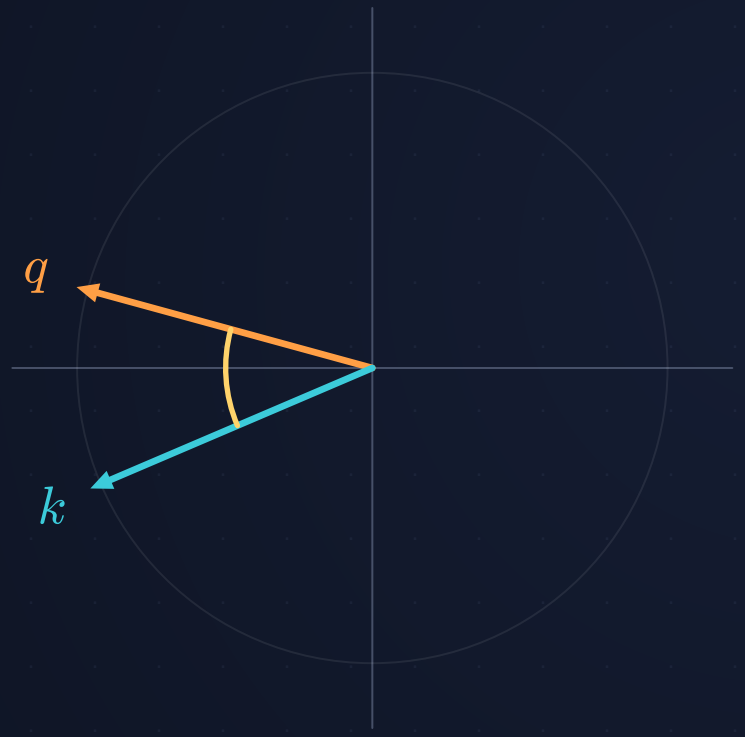
$$(R_m\, q) \cdot (R_n\, k) = q \cdot R_{n-m}\, k$$

m = 9 **n = 6**

m − n = 3

$$\text{score} = 1.57$$

same score after shifting both



$$(R_m\, q) \cdot (R_n\, k) = q \cdot R_{n-m}\, k$$

m = 9 **n = 6**

m − n = 3

$$\text{score} = 1.57$$

same score after shifting both

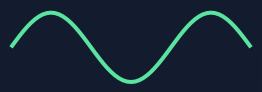
introduced in RoFormer (Su et al., 2021)

used in LLaMA, Mistral, Qwen and more



far-away words get lower scores

used in BLOOM and MPT



Sinusoidal

fixed waves, added to the input

the original Transformer



Learned

a trained vector per position

BERT, GPT-2



RoPE

rotates queries and keys

LLaMA and many recent LLMs



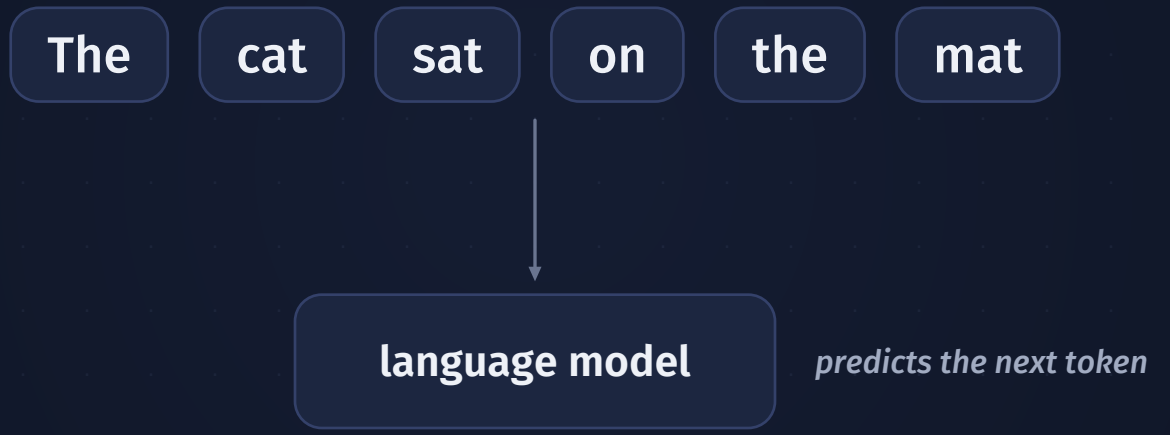
ALiBi

a distance penalty on scores

BLOOM, MPT

10 Masked attention

No peeking at the future

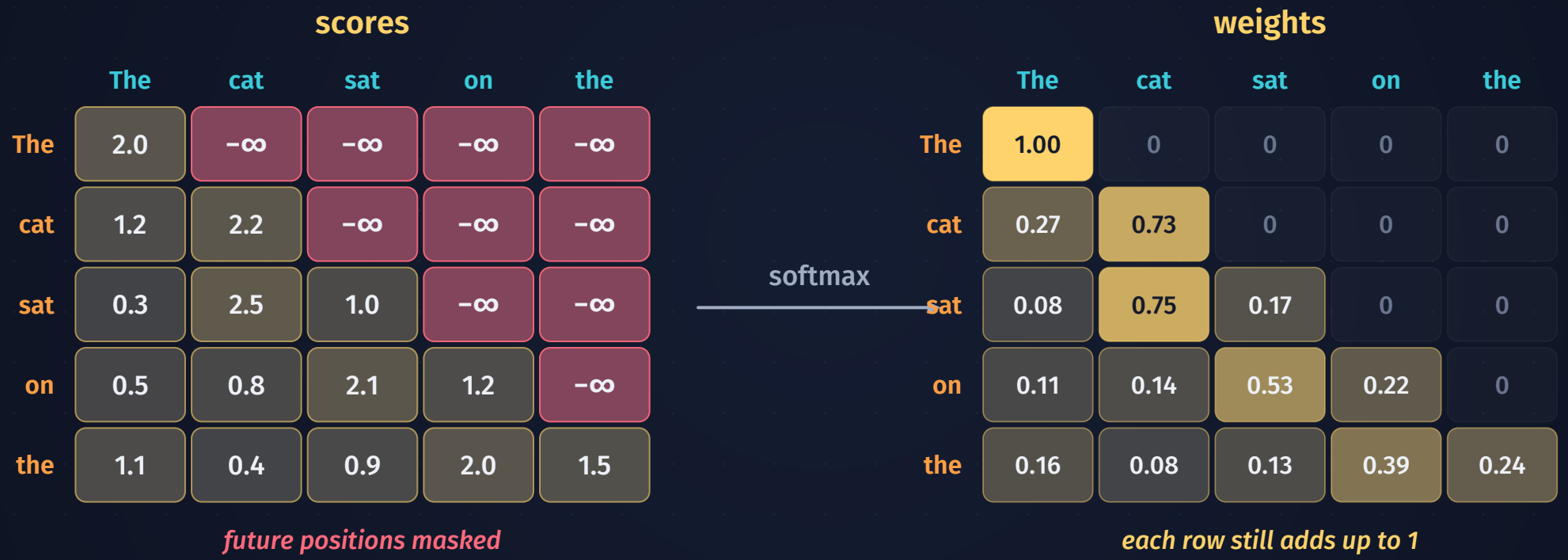

The
cat
sat
on
the
mat
language model
predicts the next token


The
cat
sat
on
the
mat
peeking at the future

scores

	The	cat	sat	on	the
The	2.0	−∞	−∞	−∞	−∞
cat	1.2	2.2	−∞	−∞	−∞
sat	0.3	2.5	1.0	−∞	−∞
on	0.5	0.8	2.1	1.2	−∞
the	1.1	0.4	0.9	2.0	1.5

future positions masked

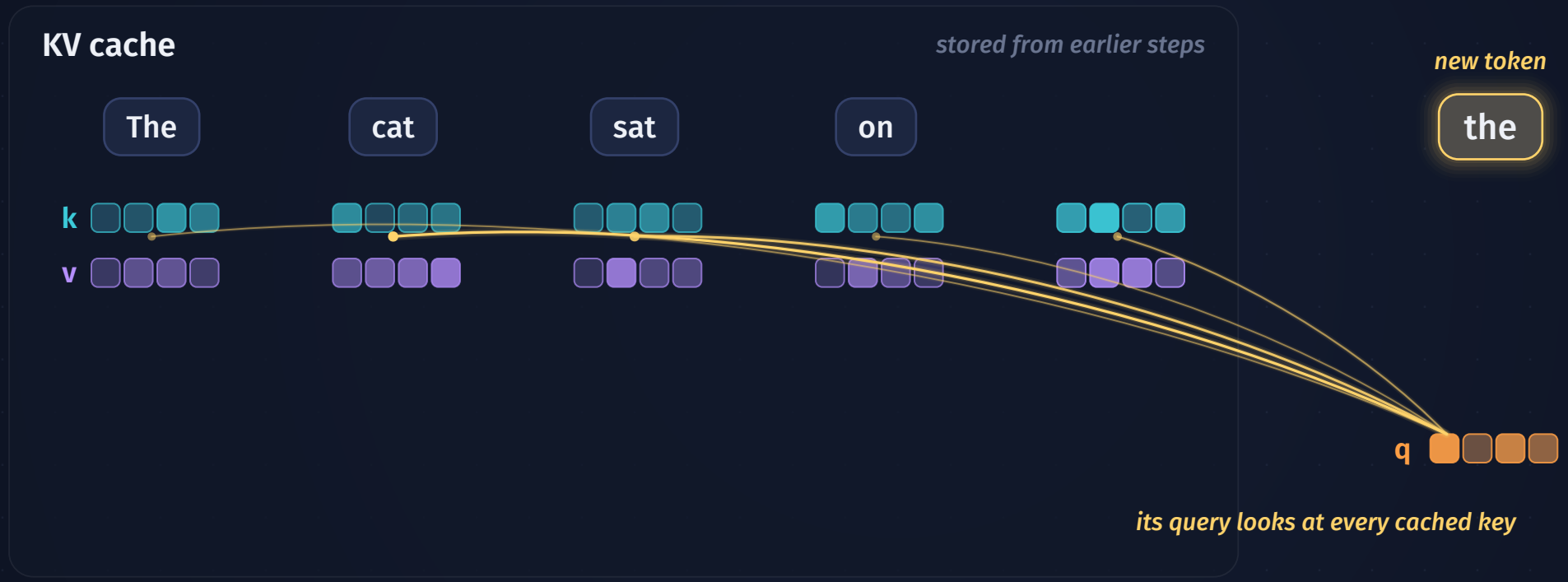

scores
The cat sat on the
The 2.0 −∞ −∞ −∞ −∞
cat 1.2 2.2 −∞ −∞ −∞
sat 0.3 2.5 1.0 −∞ −∞
on 0.5 0.8 2.1 1.2 −∞
the 1.1 0.4 0.9 2.0 1.5
future positions masked
softmax
weights
The cat sat on the
The 1.00 0 0 0 0
cat 0.27 0.73 0 0 0
sat 0.08 0.75 0.17 0 0
on 0.11 0.14 0.53 0.22 0
the 0.16 0.08 0.13 0.39 0.24
each row still adds up to 1

predict the next token at every position, all at once





causal mask



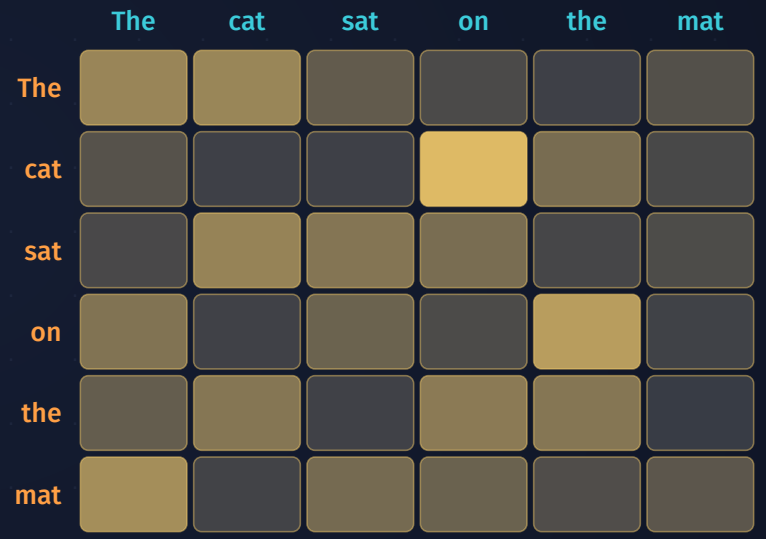
only 1 new q, k, v per step

GPT: causal (masked)



sees only the past

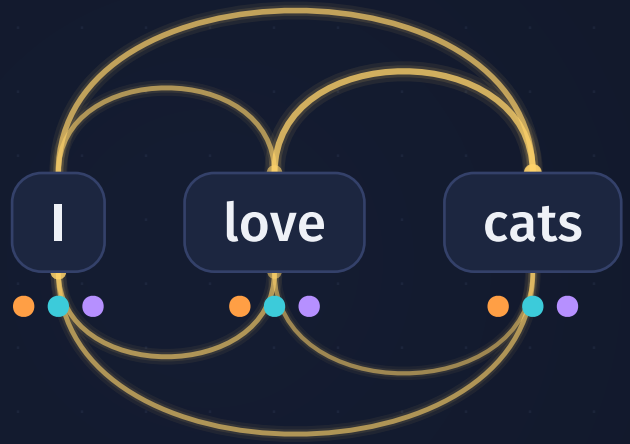
BERT: bidirectional



sees both directions

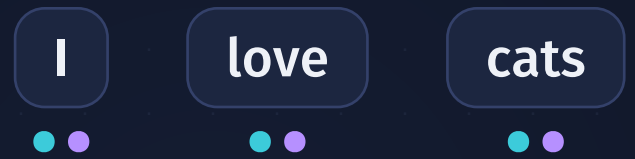
11 Cross-attention

Connecting two sequences



queries, keys and values from the same sentence

English
gives keys and values

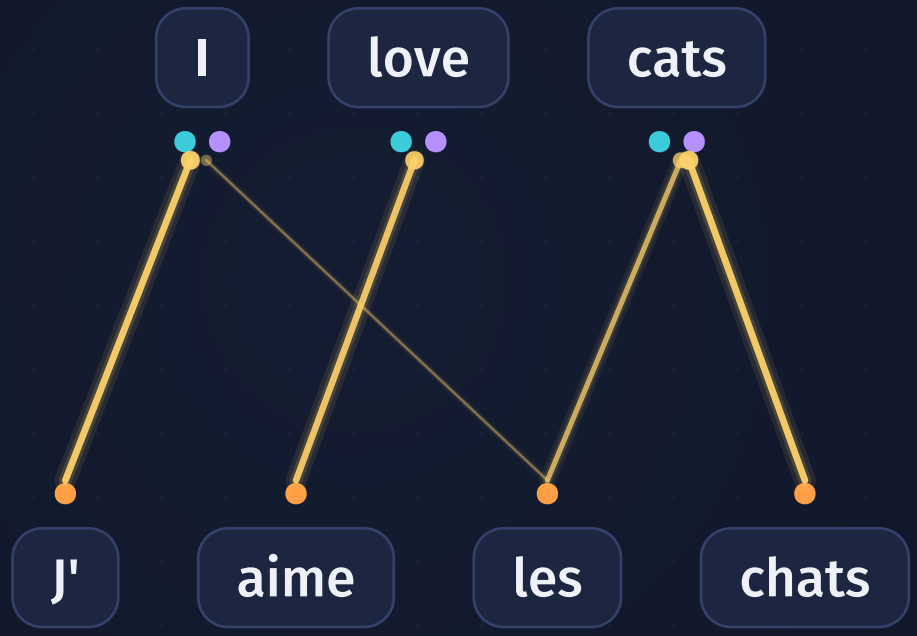


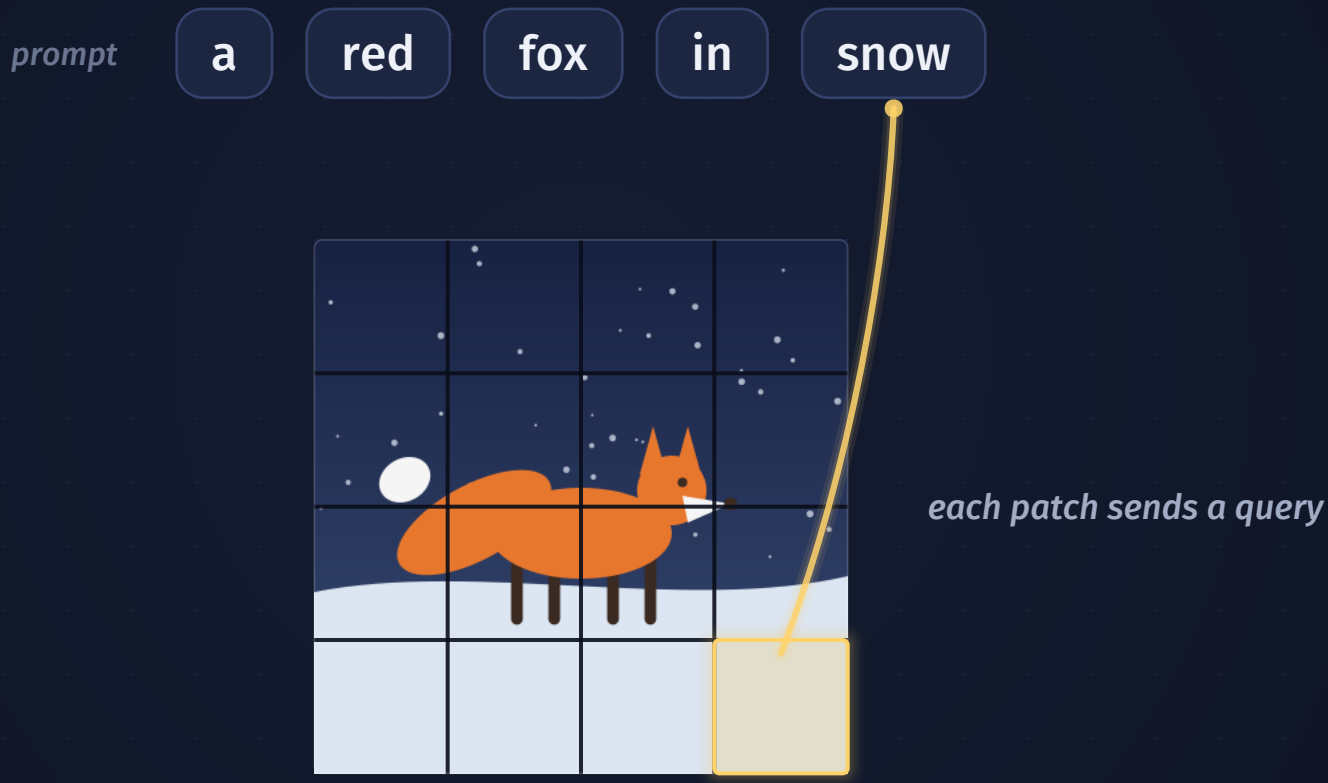
French
being written, sends queries



English
gives keys and values

French
being written, sends queries




prompt
a
red
fox
in
snow
each patch sends a query

prompt a red fox in snow

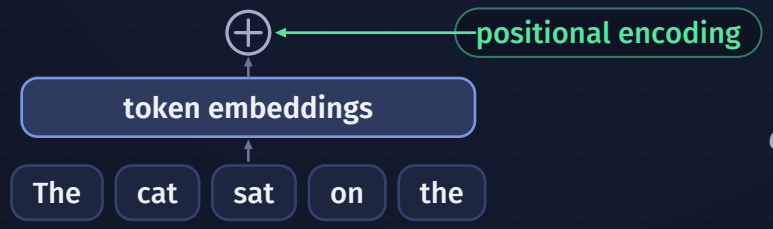
attends to “fox”

attends to “snow”



12 Inside a Transformer

Where attention lives



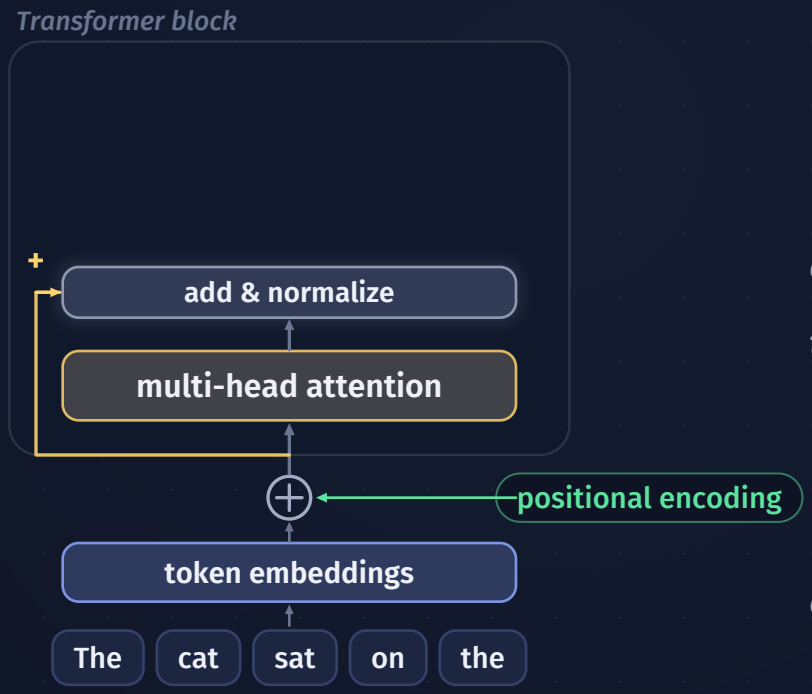
every token becomes a vector, plus its position



tokens exchange information



every token becomes a vector, plus its position

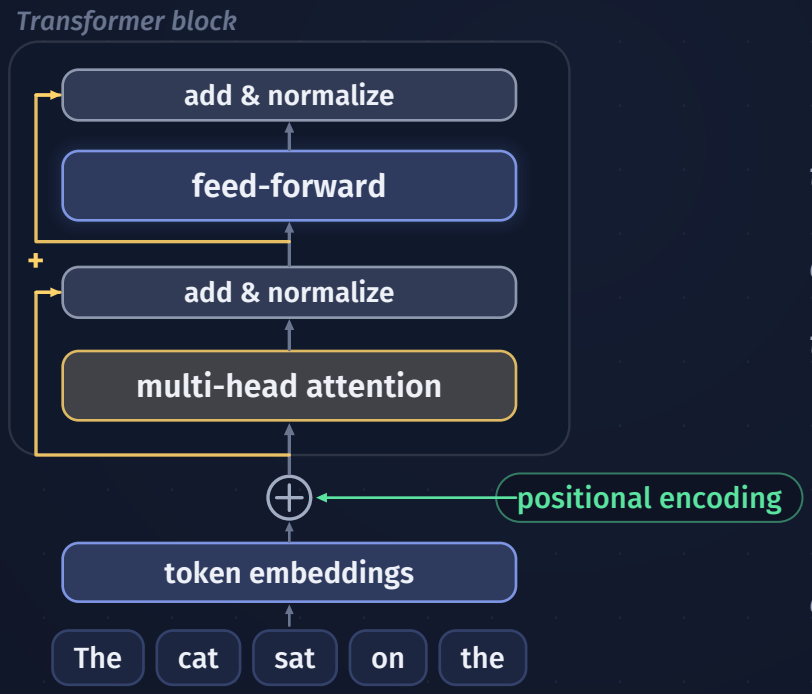


add the input back, then normalize

tokens exchange information



every token becomes a vector, plus its position



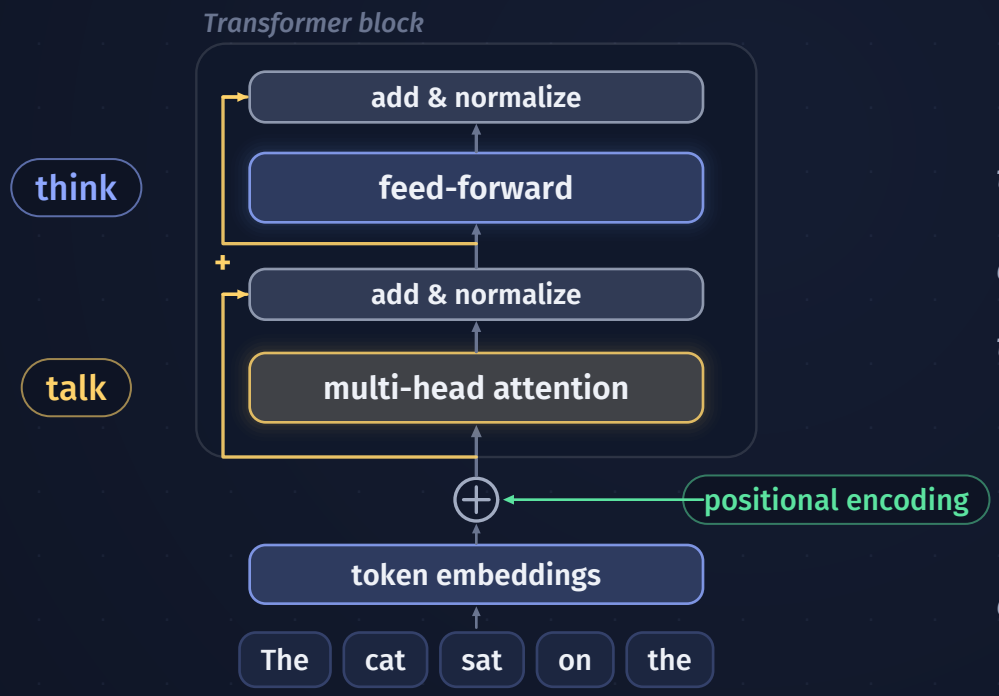
the same small network for every token

add the input back, then normalize

tokens exchange information



every token becomes a vector, plus its position



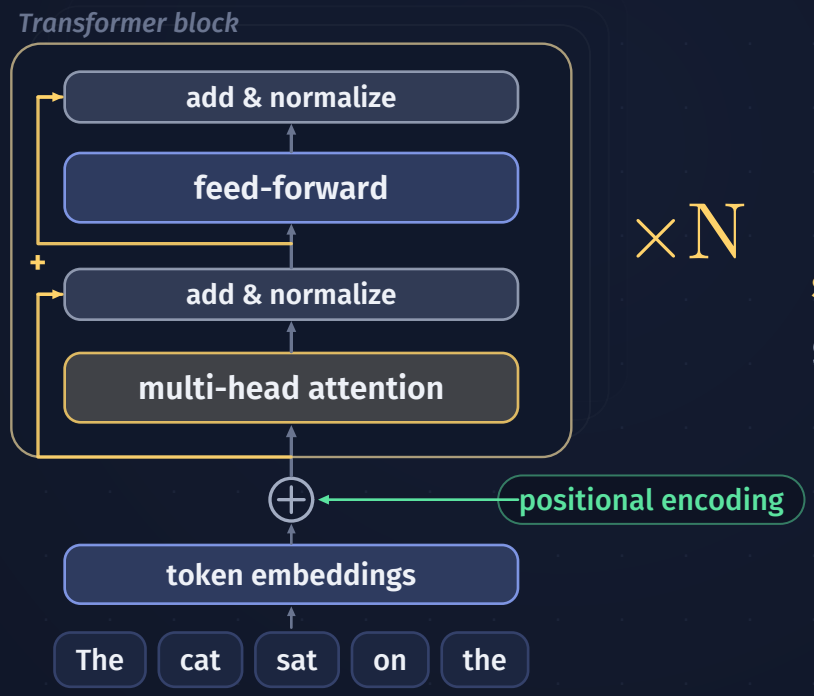
the same small network for every token

add the input back, then normalize

tokens exchange information



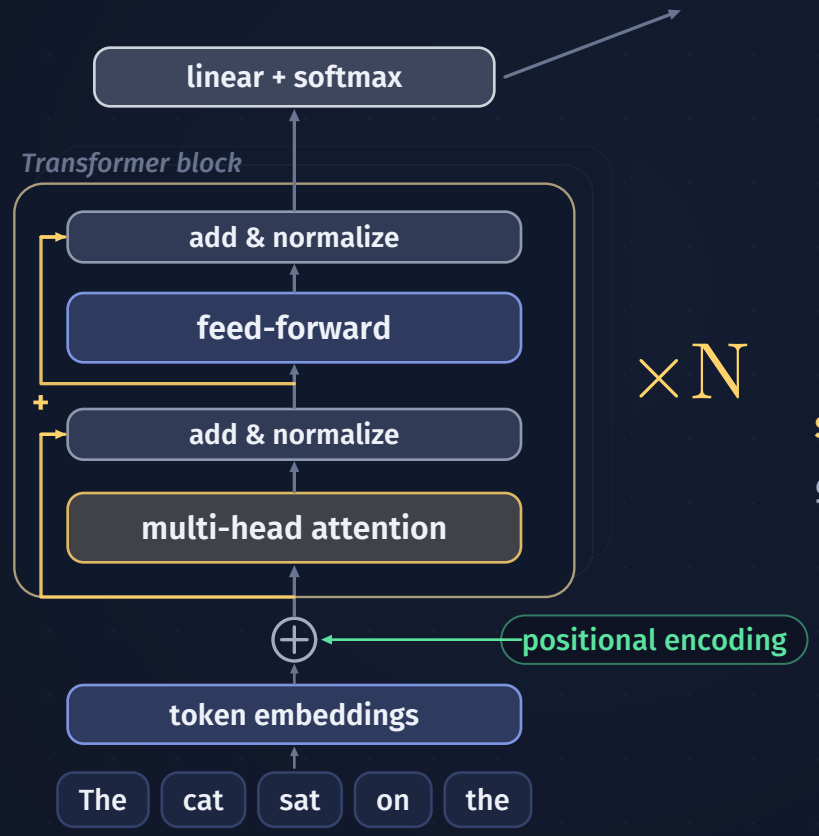
every token becomes a vector, plus its position



×N

stacked N times

96 layers in GPT-3

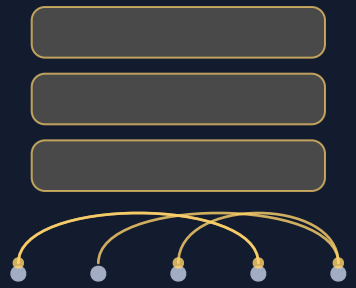


×N

stacked N times

96 layers in GPT-3



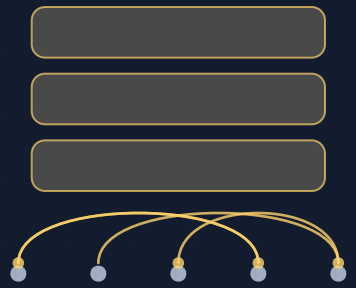


attends both ways

Encoder-only

reads and understands text

BERT



attends both ways

Encoder-only

reads and understands text

BERT

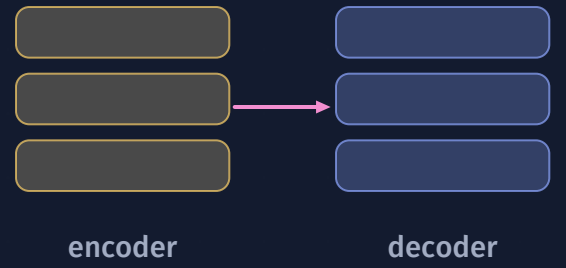


looks only at the past

Decoder-only

generates text, token by token

GPT, LLaMA



cross-attention

Encoder–decoder

turns one sequence into another

original Transformer, T5

13 Vision Transformers

An image is worth 16 × 16 words

Dosovitskiy et al., 2021



An Image is Worth 16x16 Words:

Transformers for Image Recognition at Scale

Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, Dehghani, Minderer, Heigold, Gelly, Uszkoreit, Houlsby

ICLR 2021

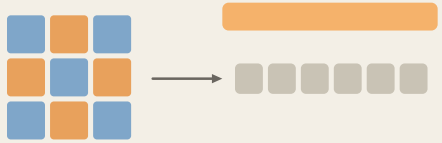
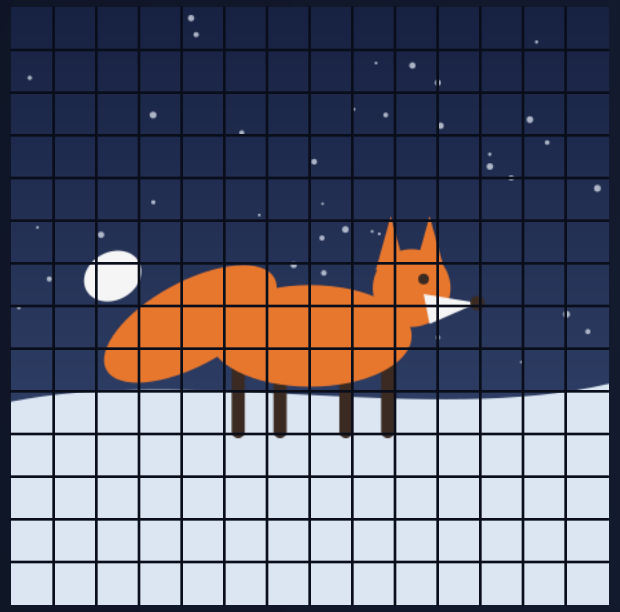


Figure 1: model overview

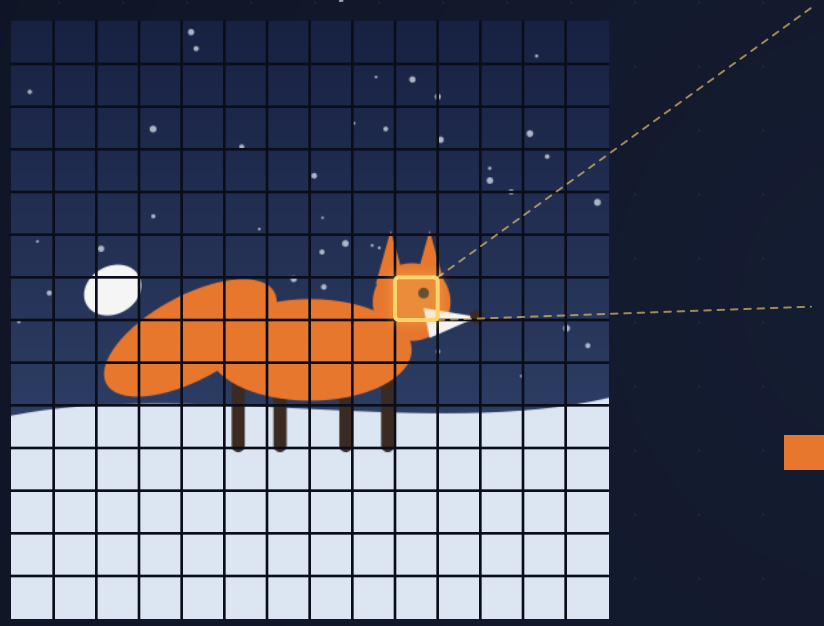
224 × 224 pixels



16 × 16 pixels per patch

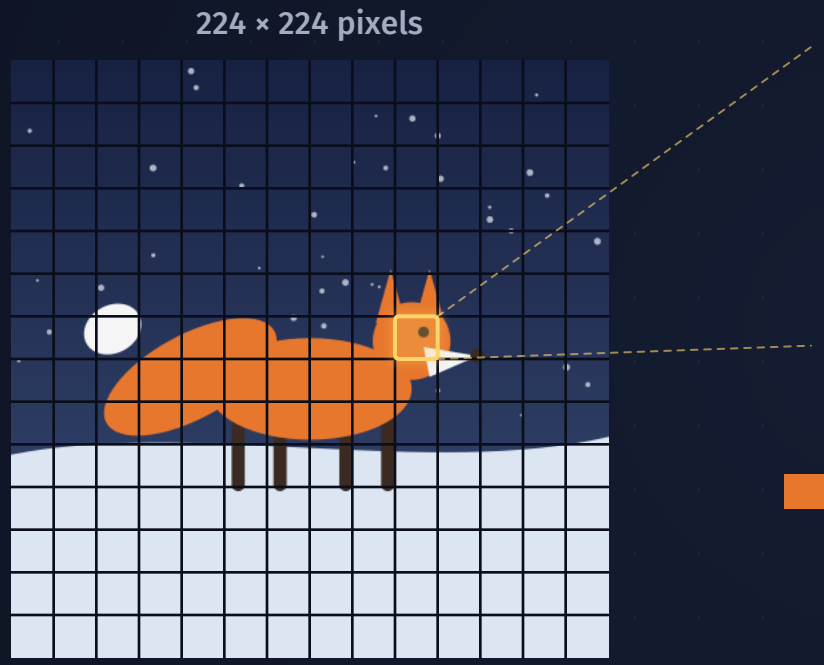
$$14 \times 14 = 196 \text{ patches}$$

224 × 224 pixels



flattened: 16 × 16 × 3 = 768 numbers





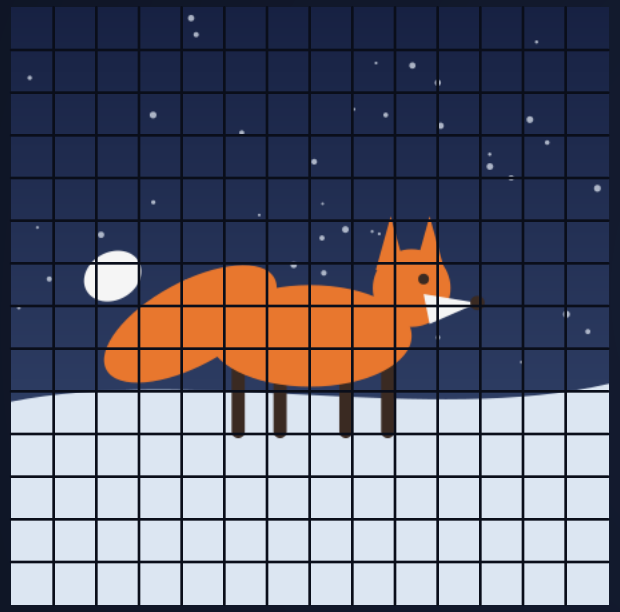
flattened: 16 × 16 × 3 = 768 numbers

$z = x_p E$

learned linear layer

patch embedding

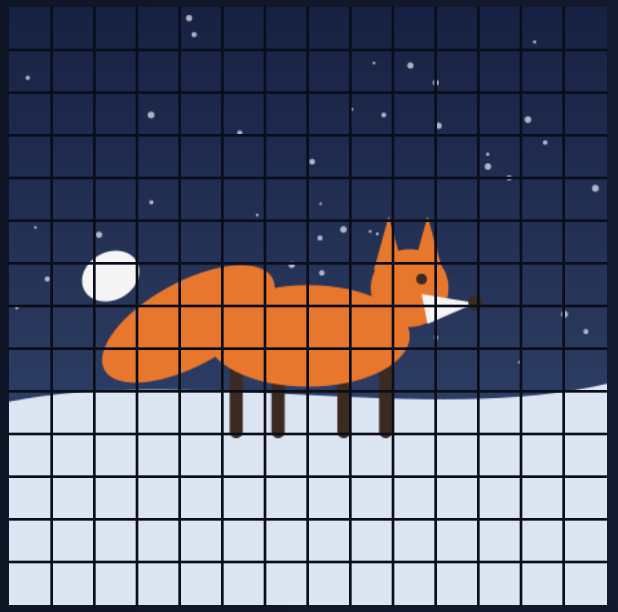
224 × 224 pixels

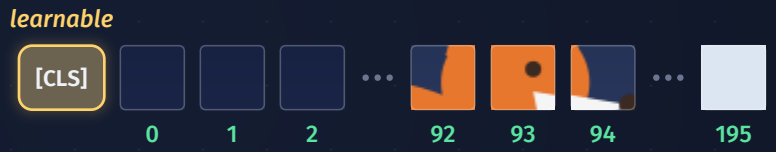


196 patch tokens

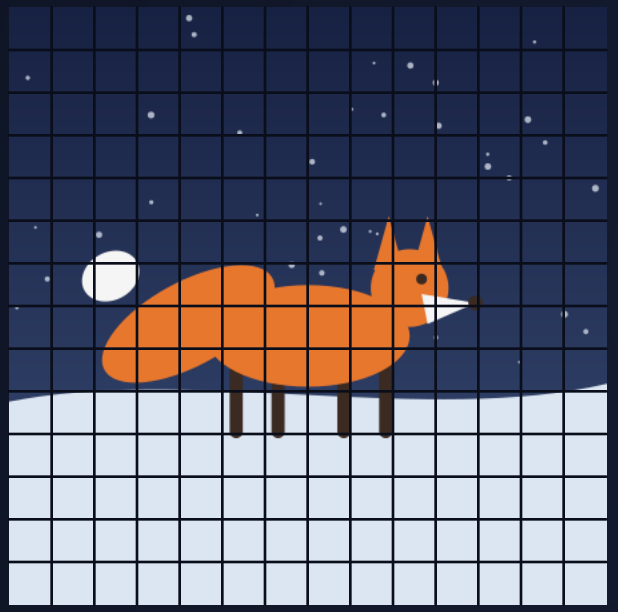


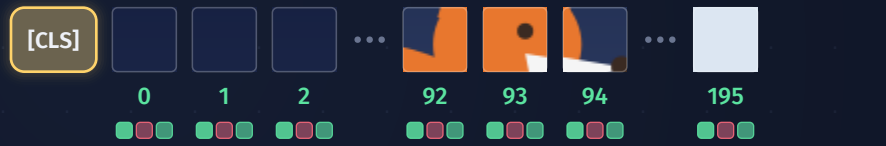
224 × 224 pixels



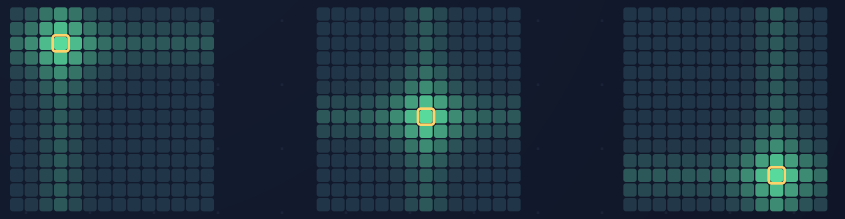


224 × 224 pixels



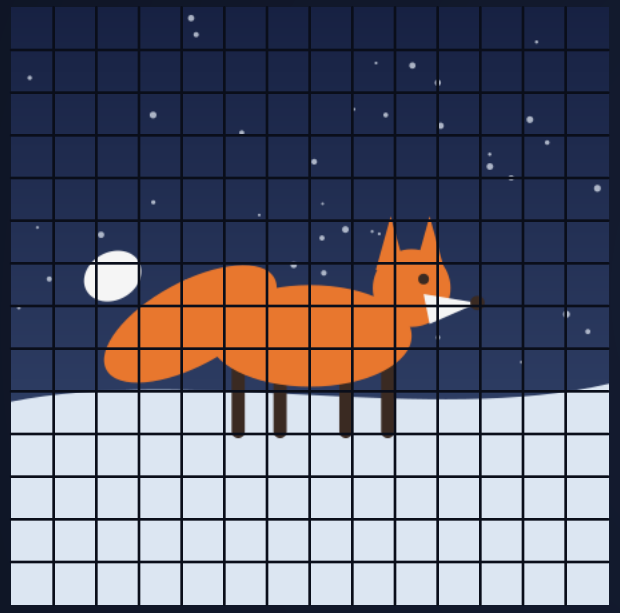


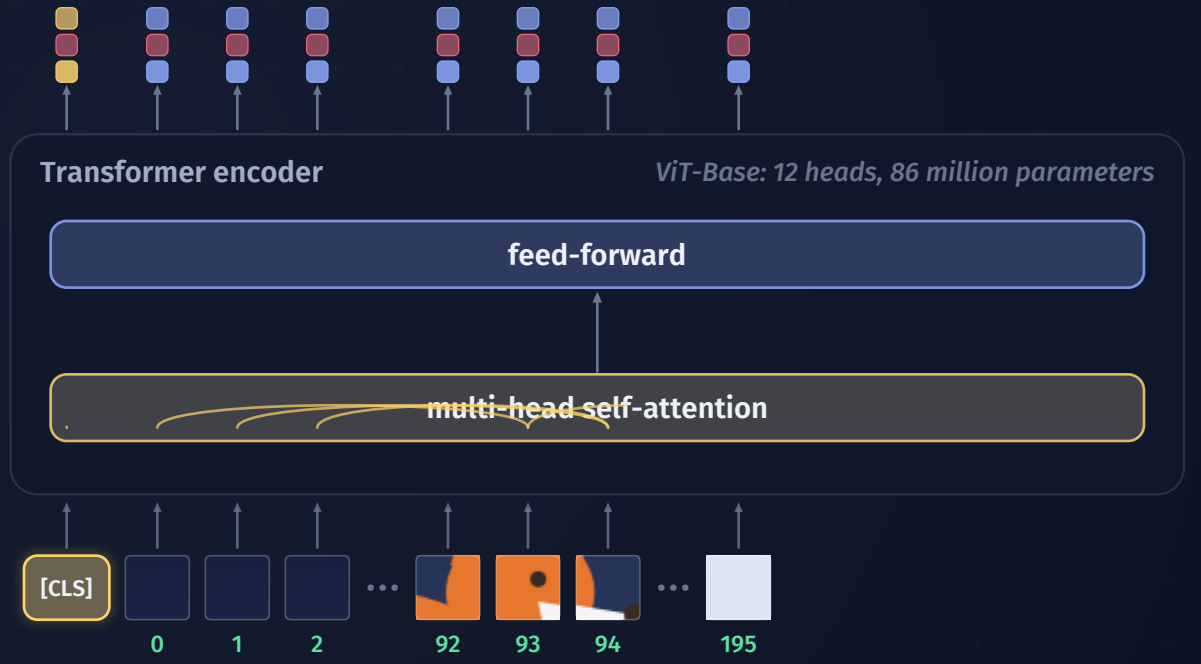
similarity between learned position embeddings (illustrative)



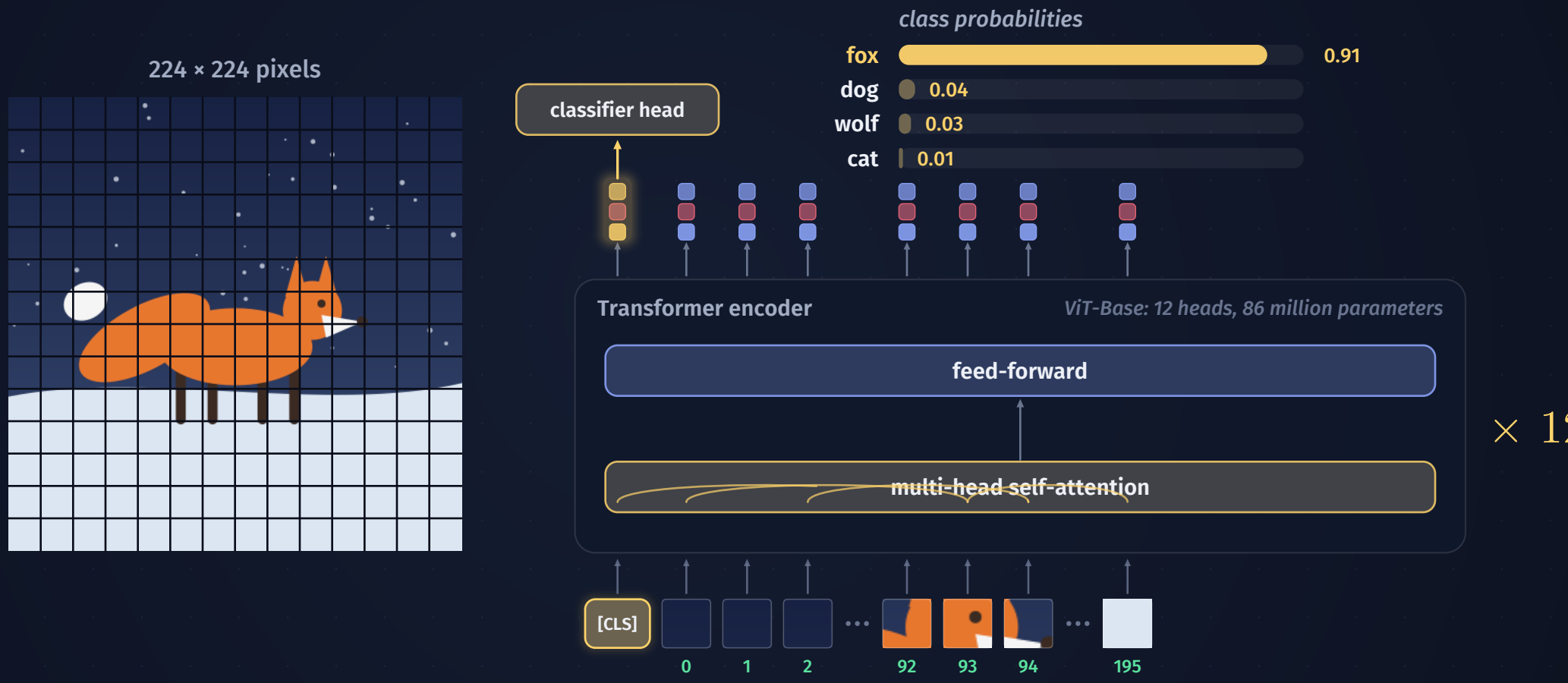
patch (2, 3) patch (7, 7) patch (11, 10)

224 × 224 pixels

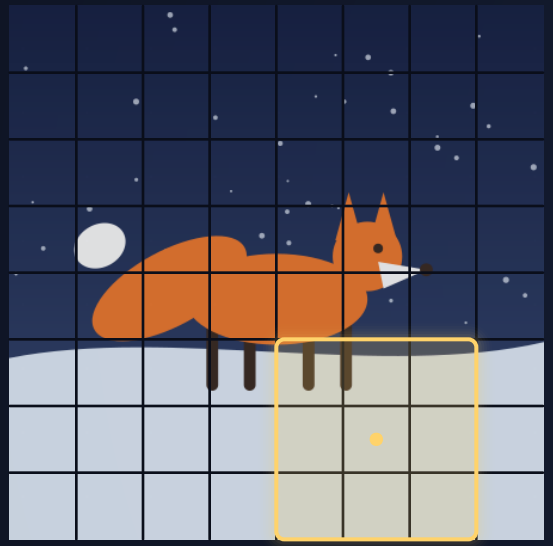




× 12


224 × 224 pixels
class probabilities
fox 0.91
dog 0.04
wolf 0.03
cat 0.01
classifier head
Transformer encoder
ViT-Base: 12 heads, 86 million parameters
feed-forward
multi-head self-attention
× 12
[CLS]
0
1
2
92
93
94
195

CNN: convolution



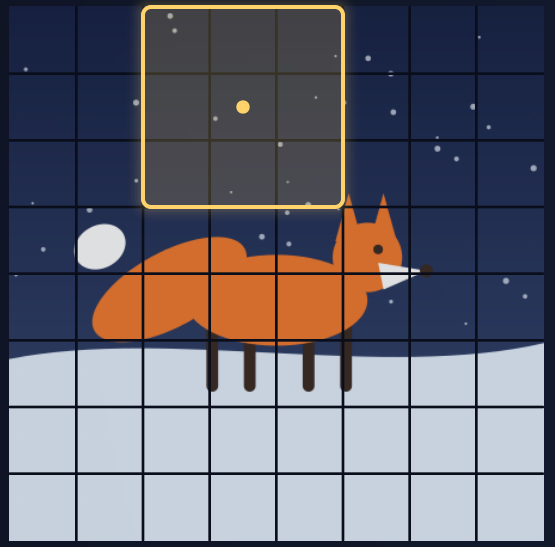
a small, fixed window

the same 3 × 3 weights

1	0	−1
2	0	−2
1	0	−1

at every position

CNN: convolution



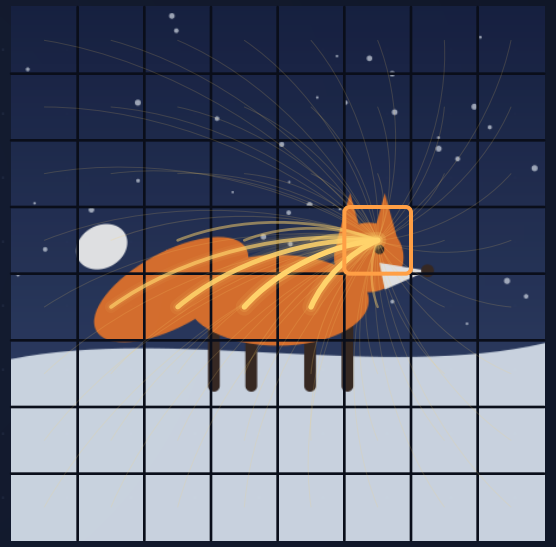
a small, fixed window

the same 3 × 3 weights

1	0	−1
2	0	−2
1	0	−1

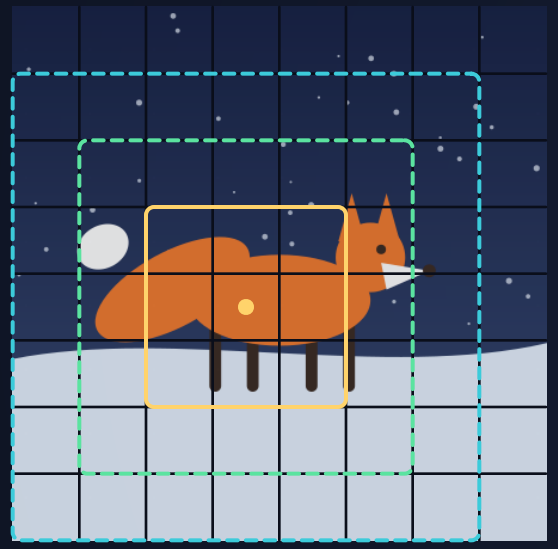
at every position

ViT: self-attention



weights computed from the content, over the whole image

CNN: convolution



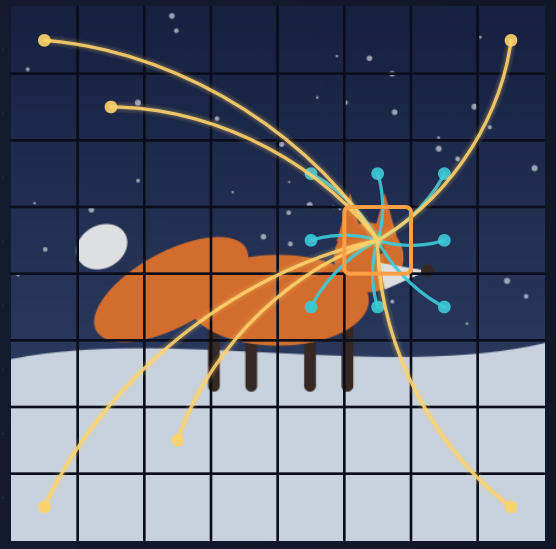
layer 1: 3 × 3

layer 2: 5 × 5

layer 3: 7 × 7

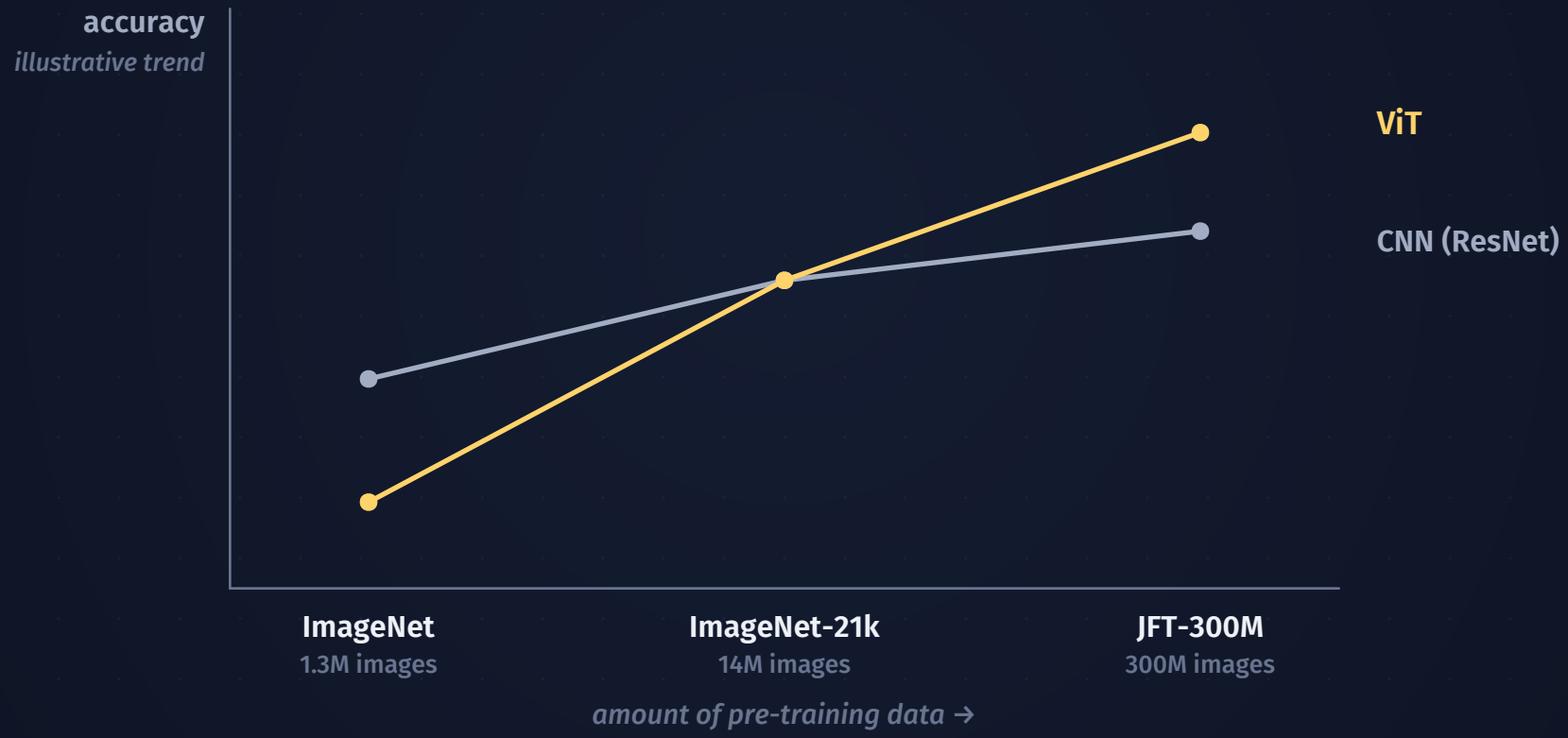
receptive field grows slowly

ViT: self-attention

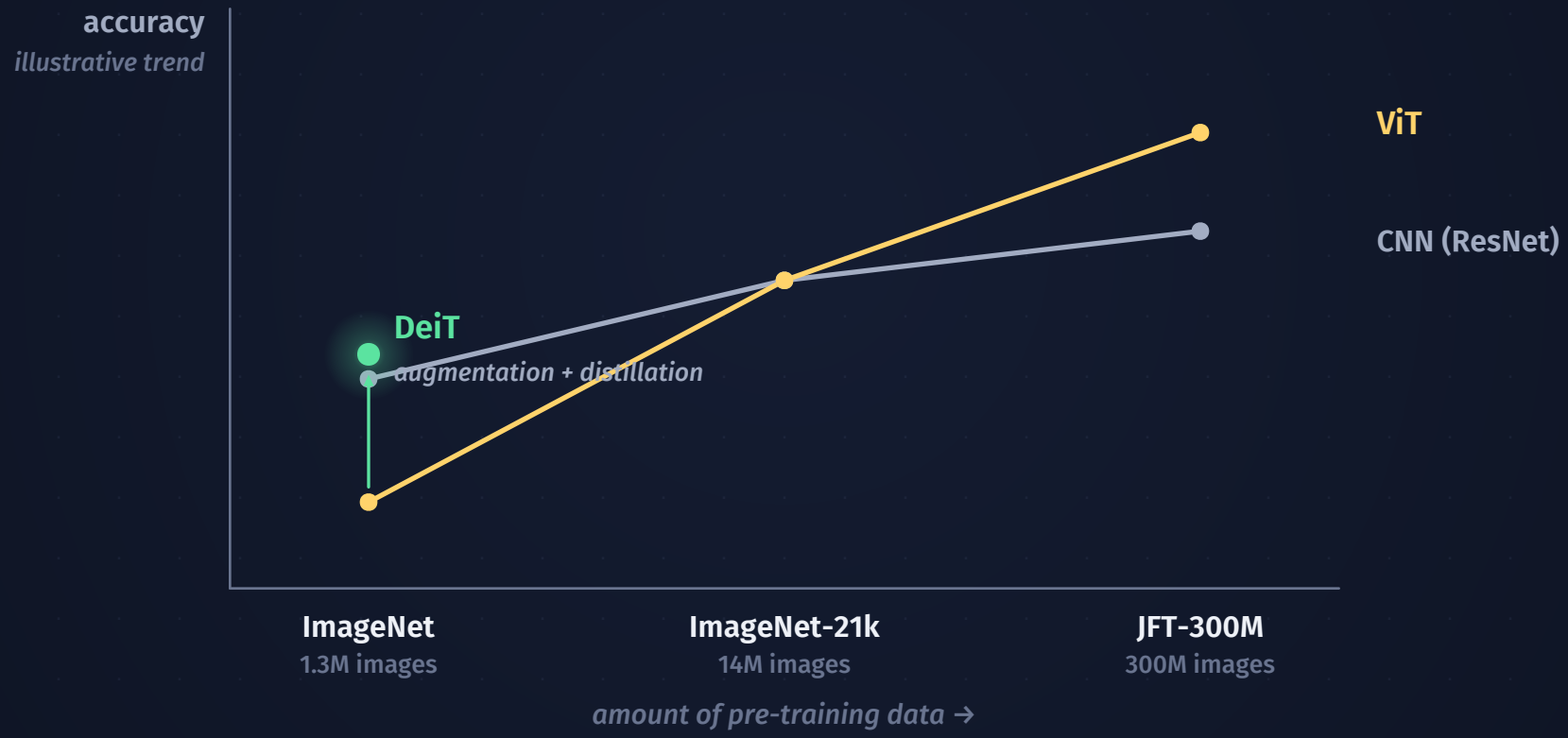


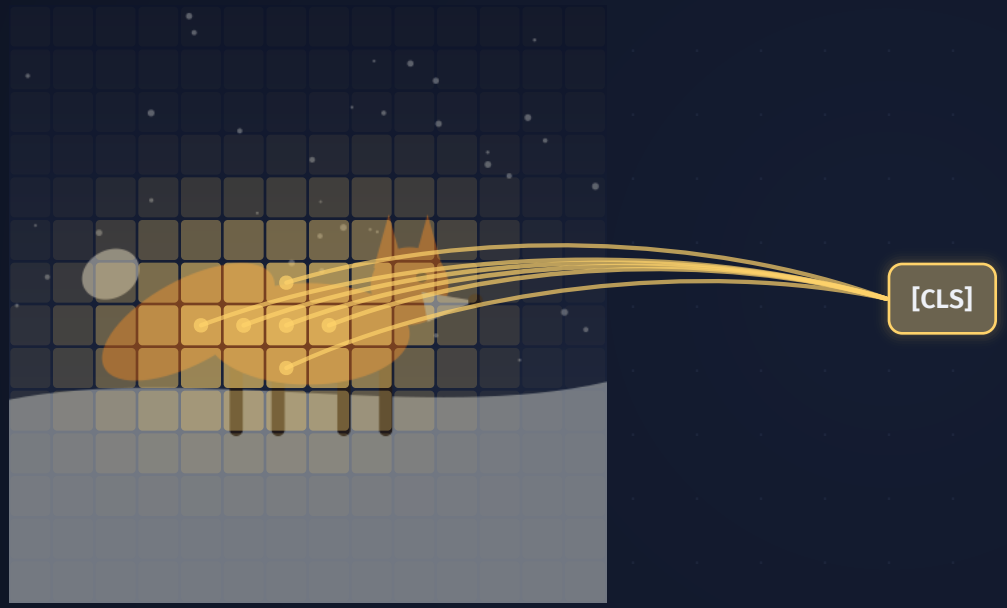
layer 1, one head: local

layer 1, another head: global


accuracy
illustrative trend
ViT
CNN (ResNet)
ImageNet
1.3M images
ImageNet-21k
14M images
JFT-300M
300M images
amount of pre-training data →

Touvron et al., 2021





attention from [CLS]

averaged over heads, one square per patch

illustrative

14 Attention in computer vision

From captions to segmentation

Xu et al., 2015

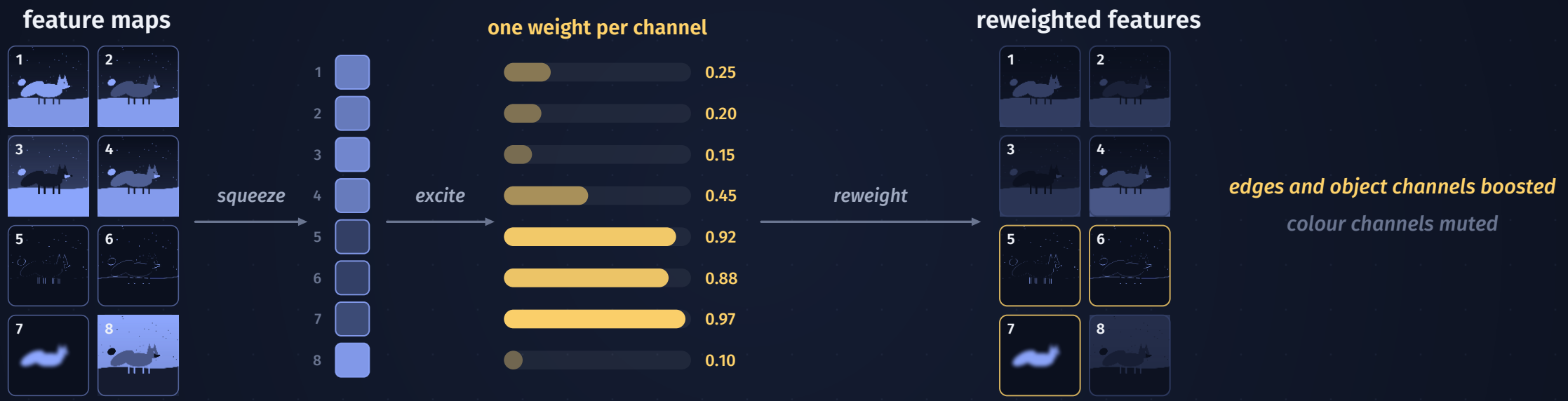


where the model looks for each word

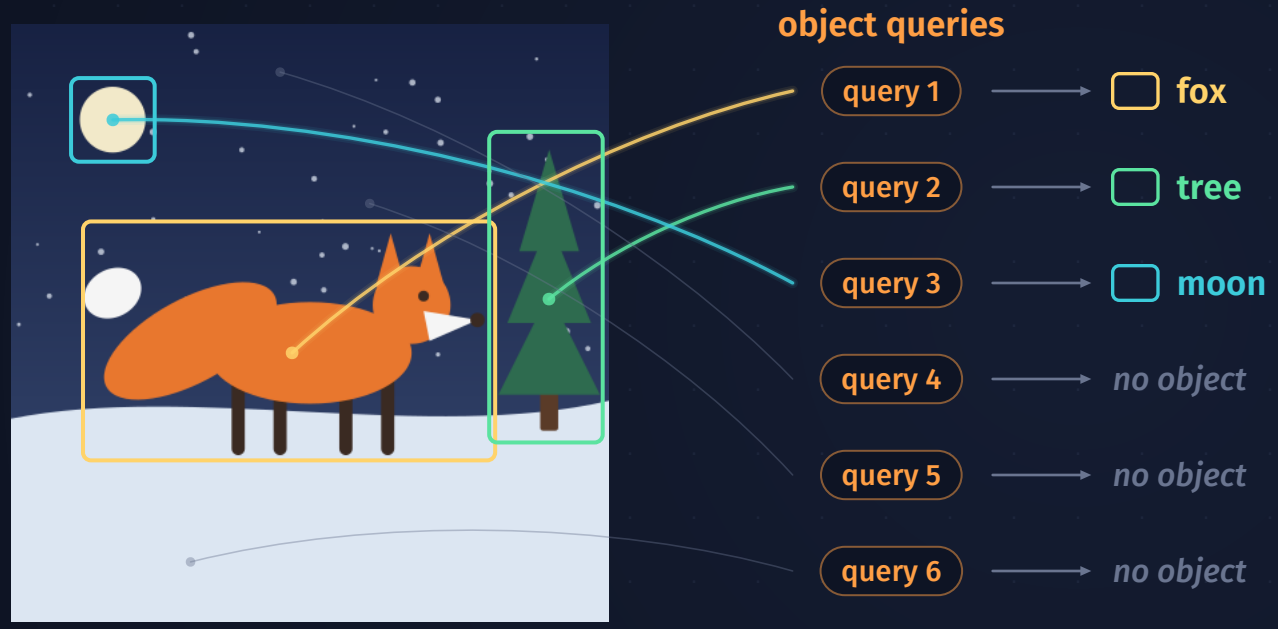
caption, written one word at a time

a fox stands in the snow

Hu et al., 2018

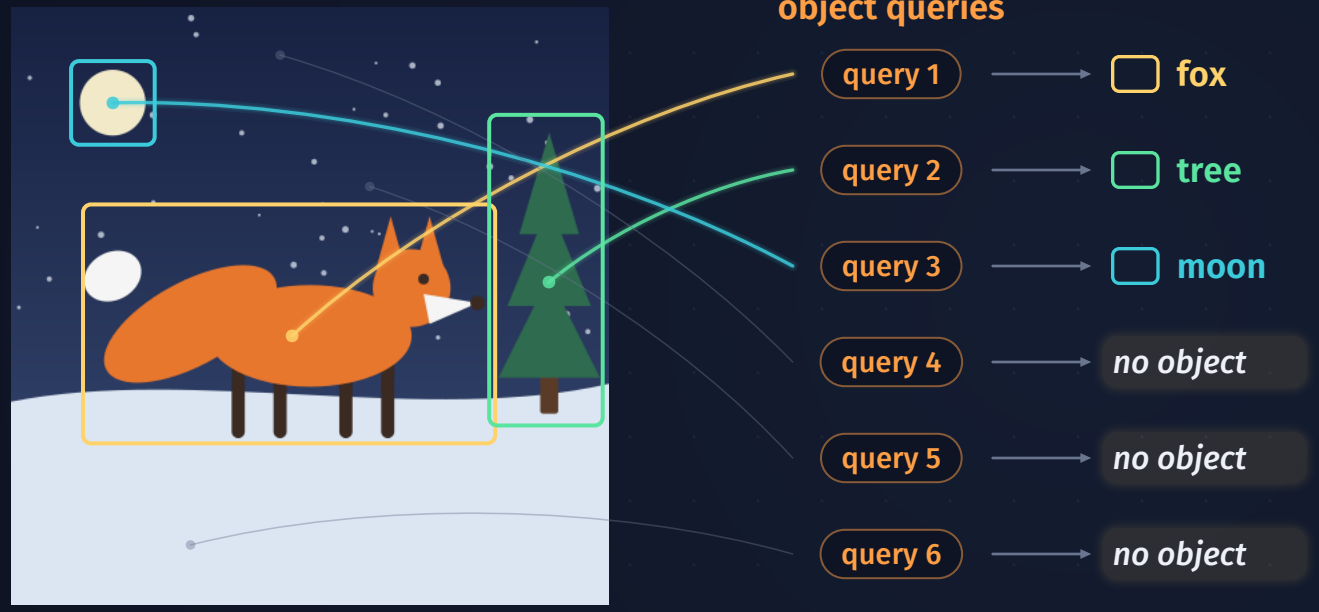

feature maps
squeeze
excite
one weight per channel
0.25
0.20
0.15
0.45
0.92
0.88
0.97
0.10
reweight
reweighted features
edges and object channels boosted
colour channels muted

Carion et al., 2020



transformer decoder: each query cross-attends to CNN image features

Carion et al., 2020



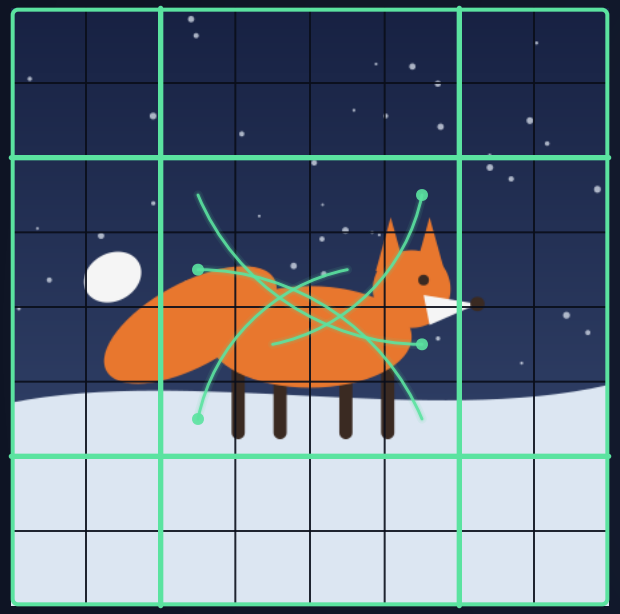
transformer decoder: each query cross-attends to CNN image features

no anchor boxes

no non-maximum suppression

a set of predictions

Liu et al., 2021



Layer l: attention inside each small window

Layer l + 1: windows shift by half a window

so information flows between neighbouring windows

Cost grows linearly with image size, not quadratically

Caron et al., 2021



attention of the [CLS] token, last layer

different heads focus on different parts

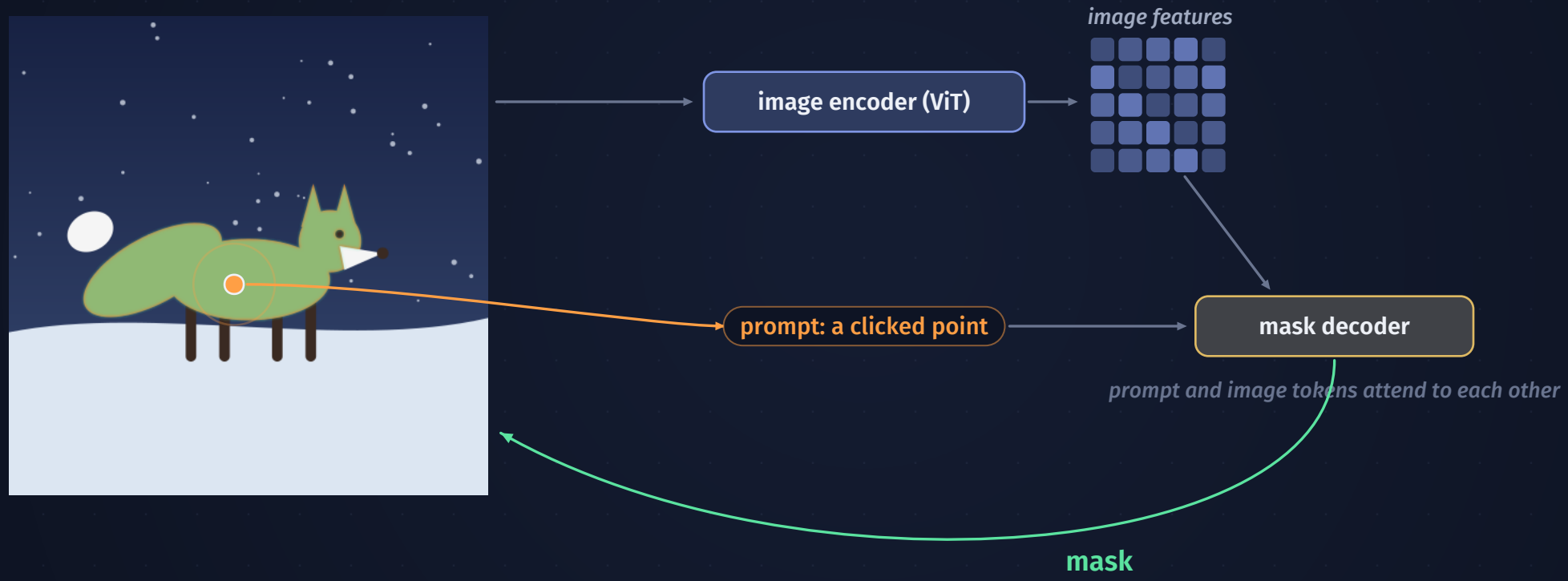


head 1 head 2 head 3

learned without any labels

illustrative maps

Kirillov et al., 2023



Radford et al., 2021

text encoder (Transformer)

image encoder (often a ViT)

	a fox in the snow	a sunny beach	a city at night
	0.91	0.08	0.05
	0.06	0.88	0.09
	0.04	0.07	0.90

matching pairs score highest

zero-shot: pick the caption that matches best

Classification

ViT, DeiT, Swin

image patches as tokens

Detection

DETR

object queries find objects

Segmentation

SAM, Mask2Former

prompts and masks via attention

Self-supervised learning

DINO, MAE

learning from images without labels

Vision and language

CLIP, captioning

matching images with text

Generation and video

diffusion models, TimeSformer

prompts steer images; attention over time

15 Attention everywhere

One mechanism, many fields



Chat assistants

ChatGPT, Claude, Gemini: Transformer language models



Search

BERT-style models read queries in both directions



Vision

Vision Transformers treat image patches as tokens



Speech

Whisper turns audio frames into text



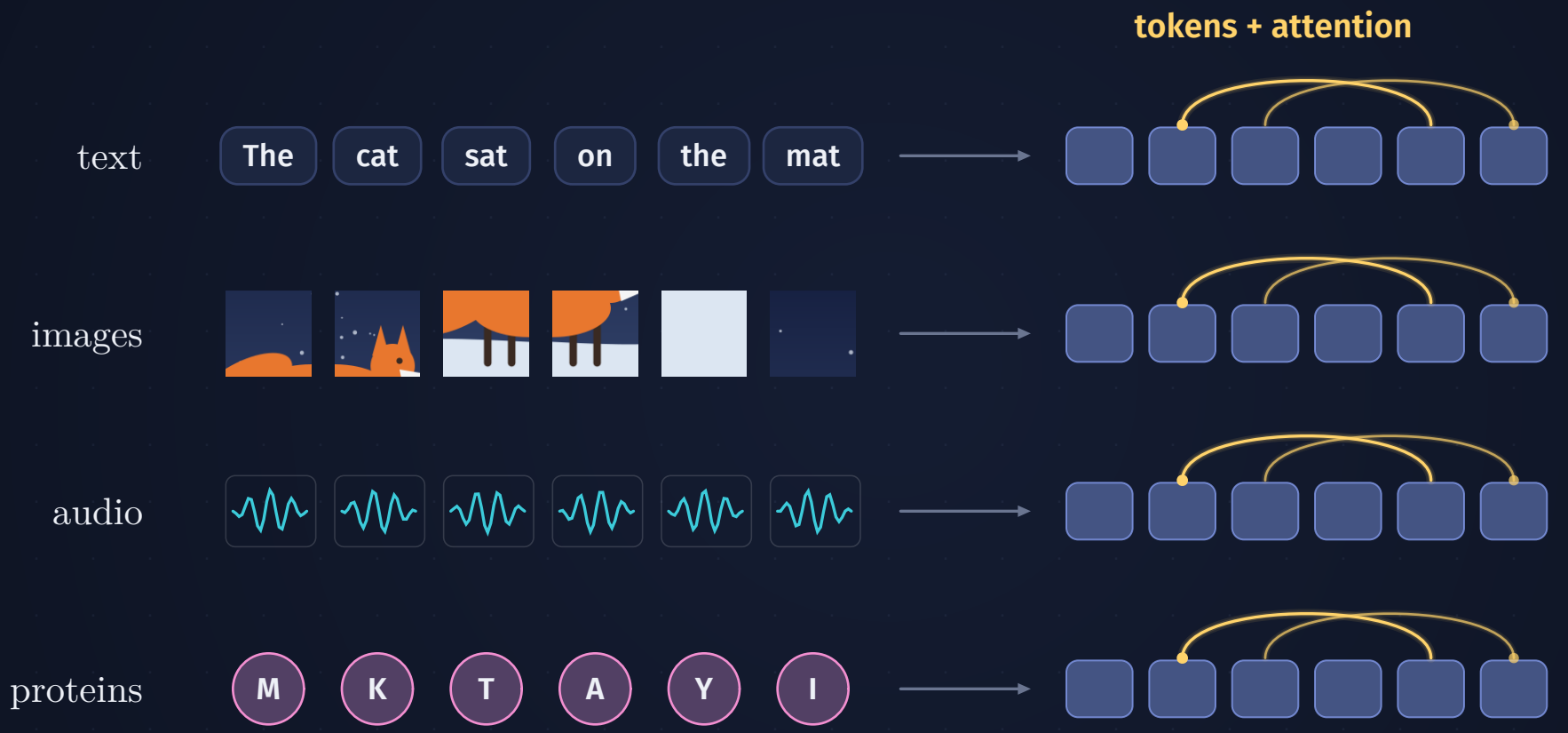
Image generation

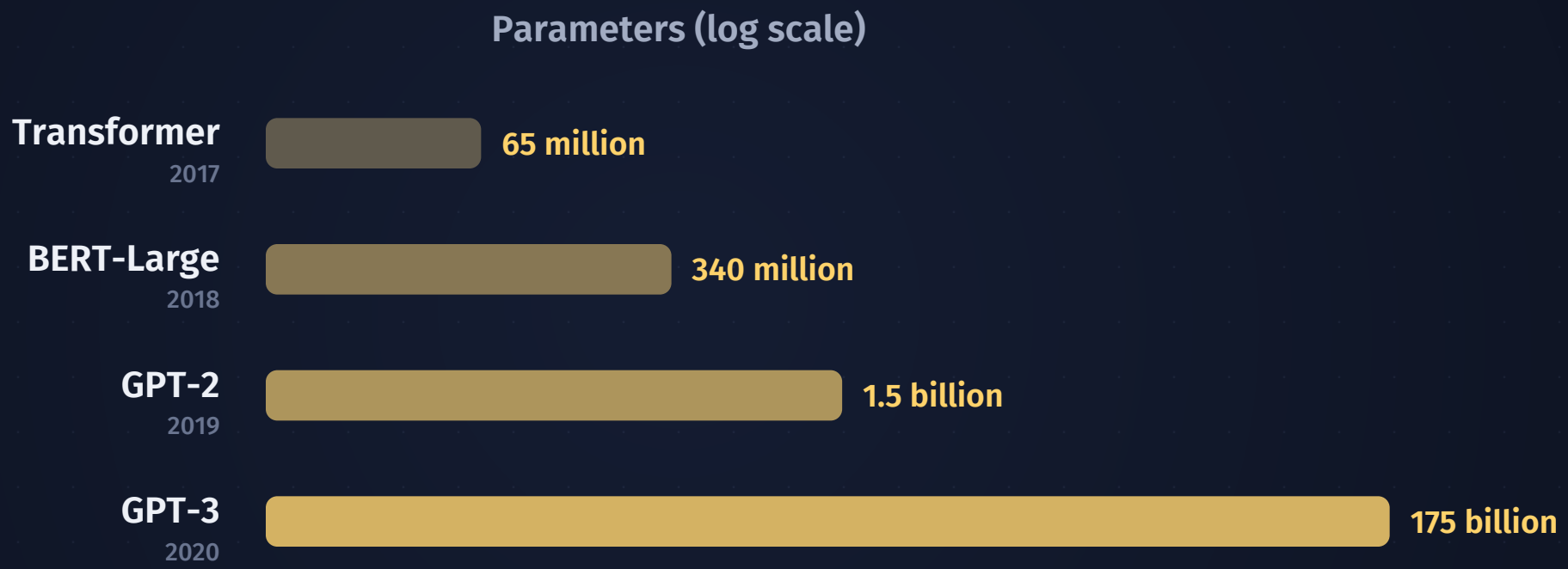
the prompt steers the picture through cross-attention



Science

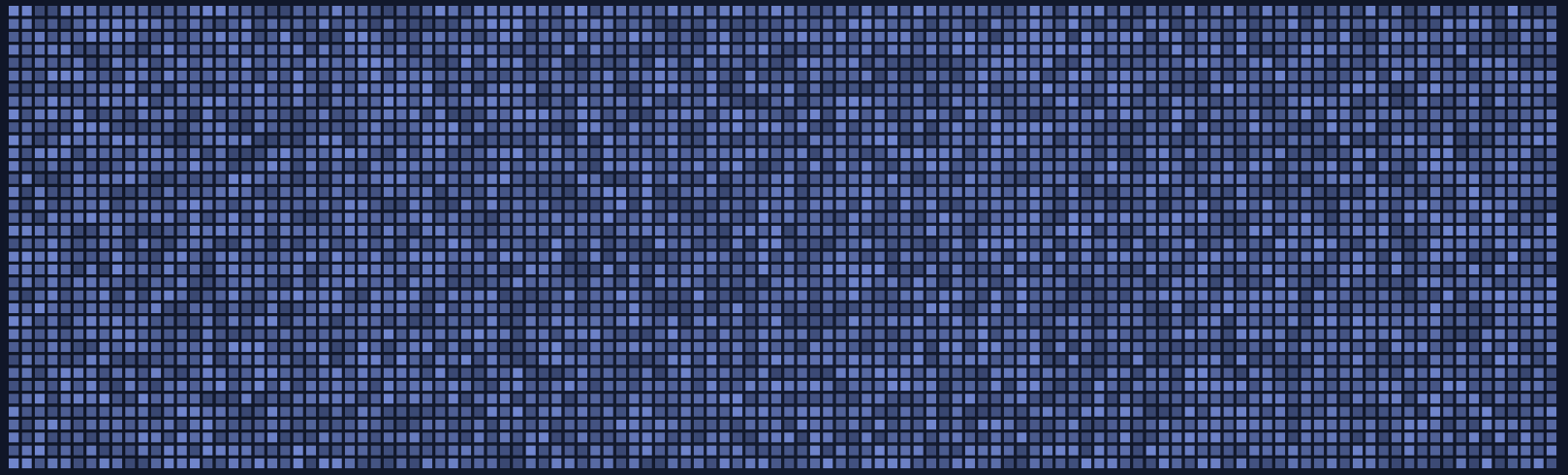
AlphaFold uses attention to predict protein structures


tokens + attention
text
The
cat
sat
on
the
mat
images
audio
proteins
M
K
T
A
Y
I



GPT-3: 96 layers × 96 attention heads

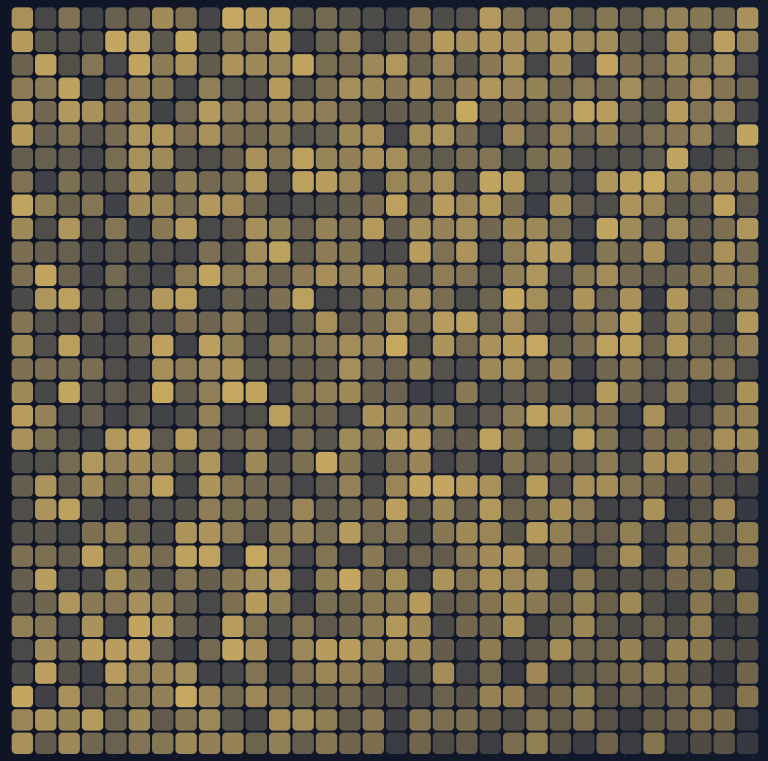
200,000 tokens



each square ≈ 46 tokens

16 The cost of attention

Why long inputs are hard



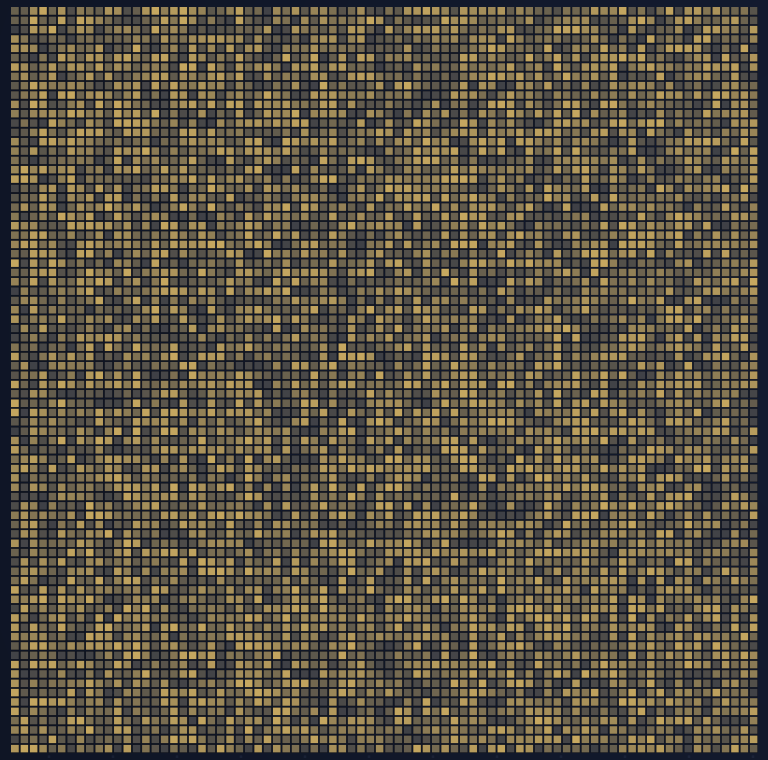
every token × every token

4 tokens → 16 scores

8 tokens → 64 scores

16 tokens → 256 scores

32 tokens → 1,024 scores



every token × every token

4 tokens → 16 scores

100,000 tokens

8 tokens → 64 scores

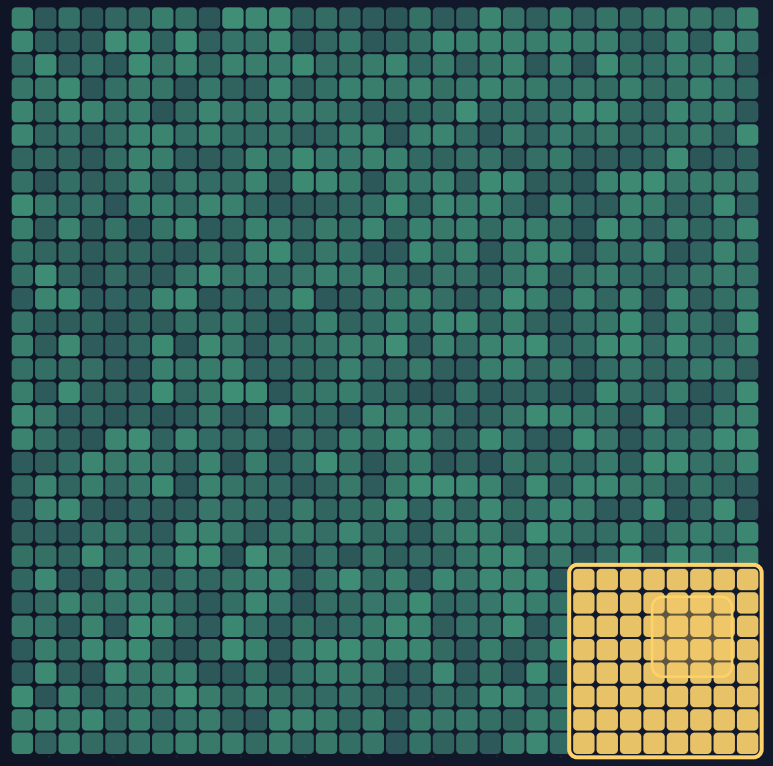
10,000,000,000

16 tokens → 256 scores

scores, for each head, in each layer

32 tokens → 1,024 scores

10 billion

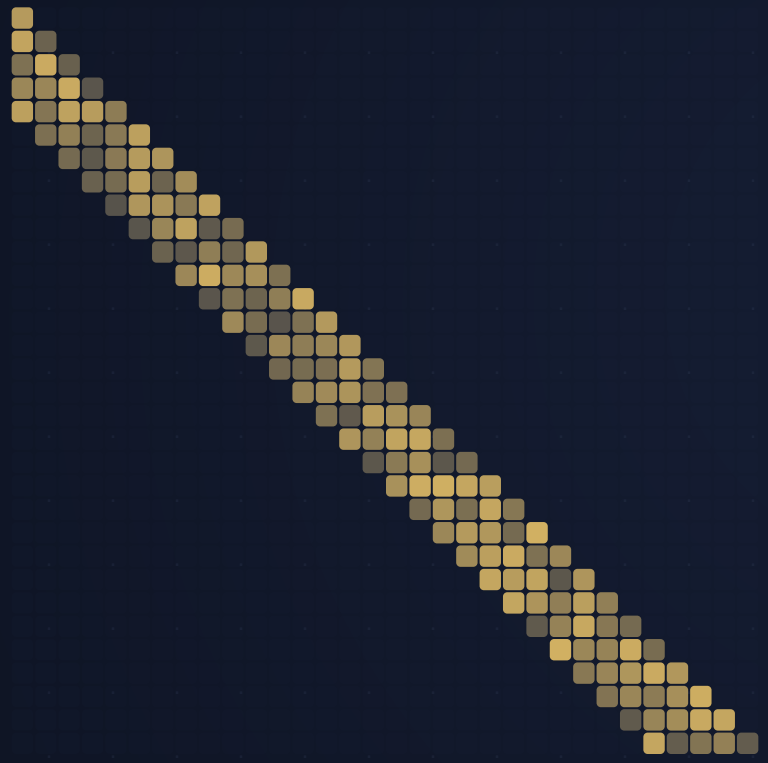


fast on-chip memory

small, but very quick

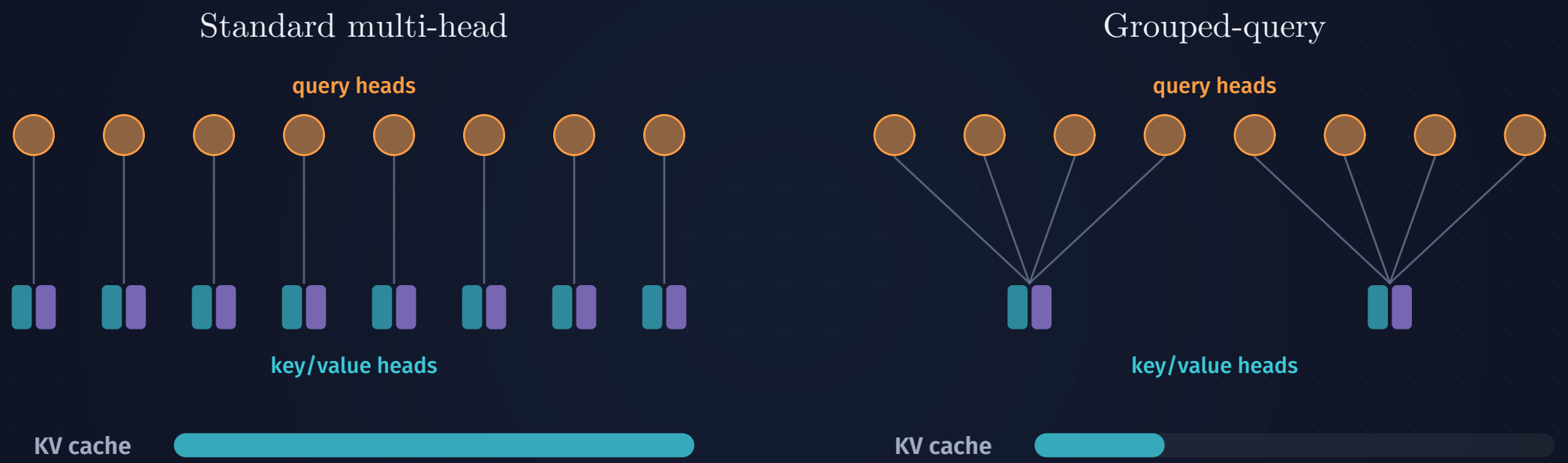
one tile at a time, exact result

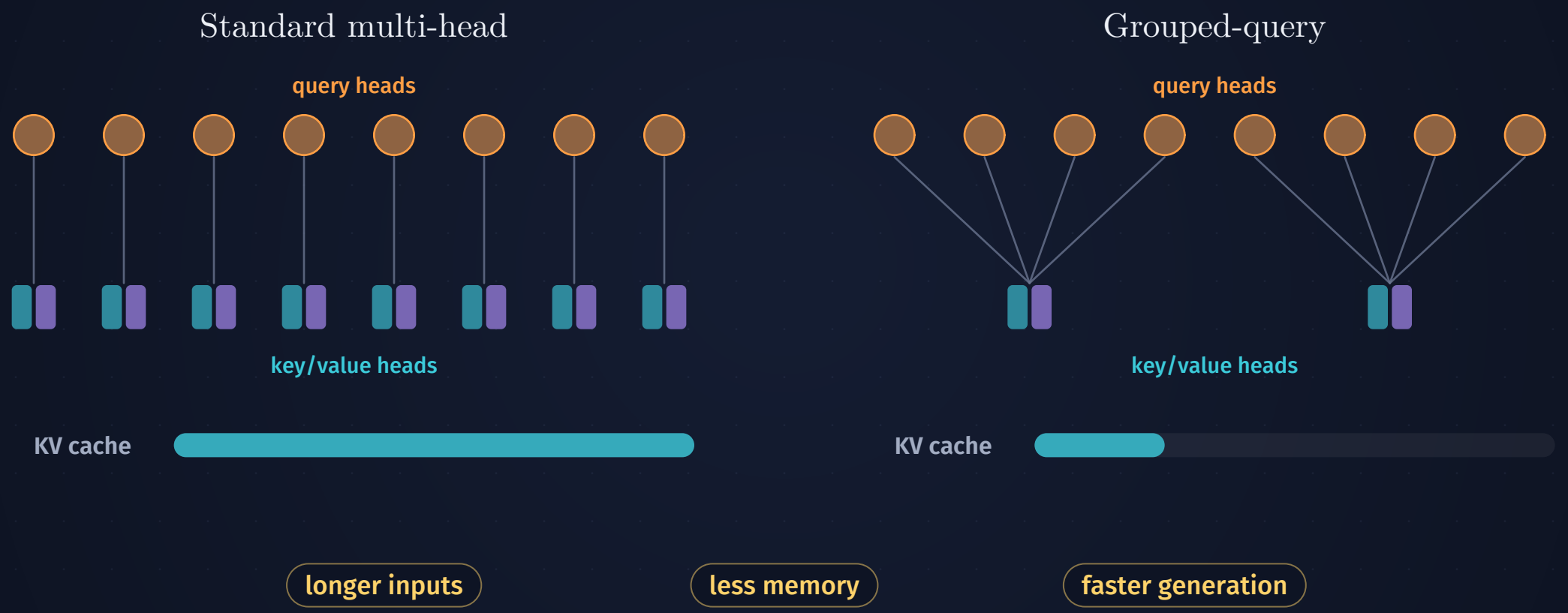
the full n × n matrix is never stored



each token sees only
its nearest neighbours

work grows like n, not n × n


Standard multi-head
query heads
key/value heads
KV cache
Grouped-query
query heads
key/value heads
KV cache



longer inputs

less memory

faster generation

17 Recap

Everything in one place

1. Attention lets each token gather information from every other token.

2. Each token makes a query, a key and a value with learned matrices.

1. Attention lets each token gather information from every other token.
2. Each token makes a **query**, a **key** and a **value** with learned matrices.
3. Scores are query–key dot products, divided by the square root of the key size.
4. **Softmax** turns scores into weights; the output is a weighted sum of values.

1. Attention lets each token gather information from every other token.
2. Each token makes a **query**, a **key** and a **value** with learned matrices.
3. Scores are query–key dot products, divided by the square root of the key size.
4. **Softmax** turns scores into weights; the output is a weighted sum of values.
5. Several heads capture different relationships in parallel.
6. **Positional encodings** (sinusoidal, learned, RoPE) give a sense of order.

1. Attention lets each token gather information from every other token.
2. Each token makes a **query**, a **key** and a **value** with learned matrices.
3. Scores are query–key dot products, divided by the square root of the key size.
4. **Softmax** turns scores into weights; the output is a weighted sum of values.
5. Several heads capture different relationships in parallel.
6. **Positional encodings** (sinusoidal, learned, RoPE) give a sense of order.
7. Masks stop peeking at the future; cross-attention connects two sequences.
8. Transformers stack attention and feed-forward layers, many times over.

1. Attention lets each token gather information from every other token.
2. Each token makes a **query**, a **key** and a **value** with learned matrices.
3. Scores are query–key dot products, divided by the square root of the key size.
4. **Softmax** turns scores into weights; the output is a weighted sum of values.
5. Several heads capture different relationships in parallel.
6. **Positional encodings** (sinusoidal, learned, RoPE) give a sense of order.
7. Masks stop peeking at the future; cross-attention connects two sequences.
8. Transformers stack attention and feed-forward layers, many times over.
9. Vision Transformers treat image patches as tokens, plus a **[CLS]** token.
10. Attention powers detection (DETR), segmentation (SAM), CLIP and more.

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



Thanks for watching

Further reading

Vaswani et al. (2017)
Attention Is All You Need

Bahdanau, Cho and Bengio (2014)
Neural Machine Translation by Jointly Learning to Align and Translate

Su et al. (2021)
RoFormer: Rotary Position Embedding

Press, Smith and Lewis (2021)
Train Short, Test Long (ALiBi)

Dao et al. (2022)
FlashAttention

Xu et al. (2015)
Show, Attend and Tell

Dosovitskiy et al. (2021)
An Image is Worth 16x16 Words (ViT)

Carion et al. (2020)
End-to-End Object Detection with Transformers (DETR)

Liu et al. (2021)
Swin Transformer

Caron et al. (2021)
Emerging Properties in Self-Supervised Vision Transformers (DINO)

Kirillov et al. (2023)
Segment Anything

Radford et al. (2021)
Learning Transferable Visual Models from Natural Language Supervision (CLIP)

CSC420: Introduction to Image Understanding, University of Toronto

Presented by Javad Rajabi, Teaching Assistant

cs.toronto.edu/~lindell/teaching/420