



# Intro to Mechanistic Interpretability

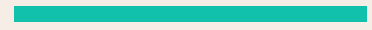
Elena Tutubalina

Principal research scientist,  
Research manager AIRI

Aleksei Dontsov

Research engineer AIRI

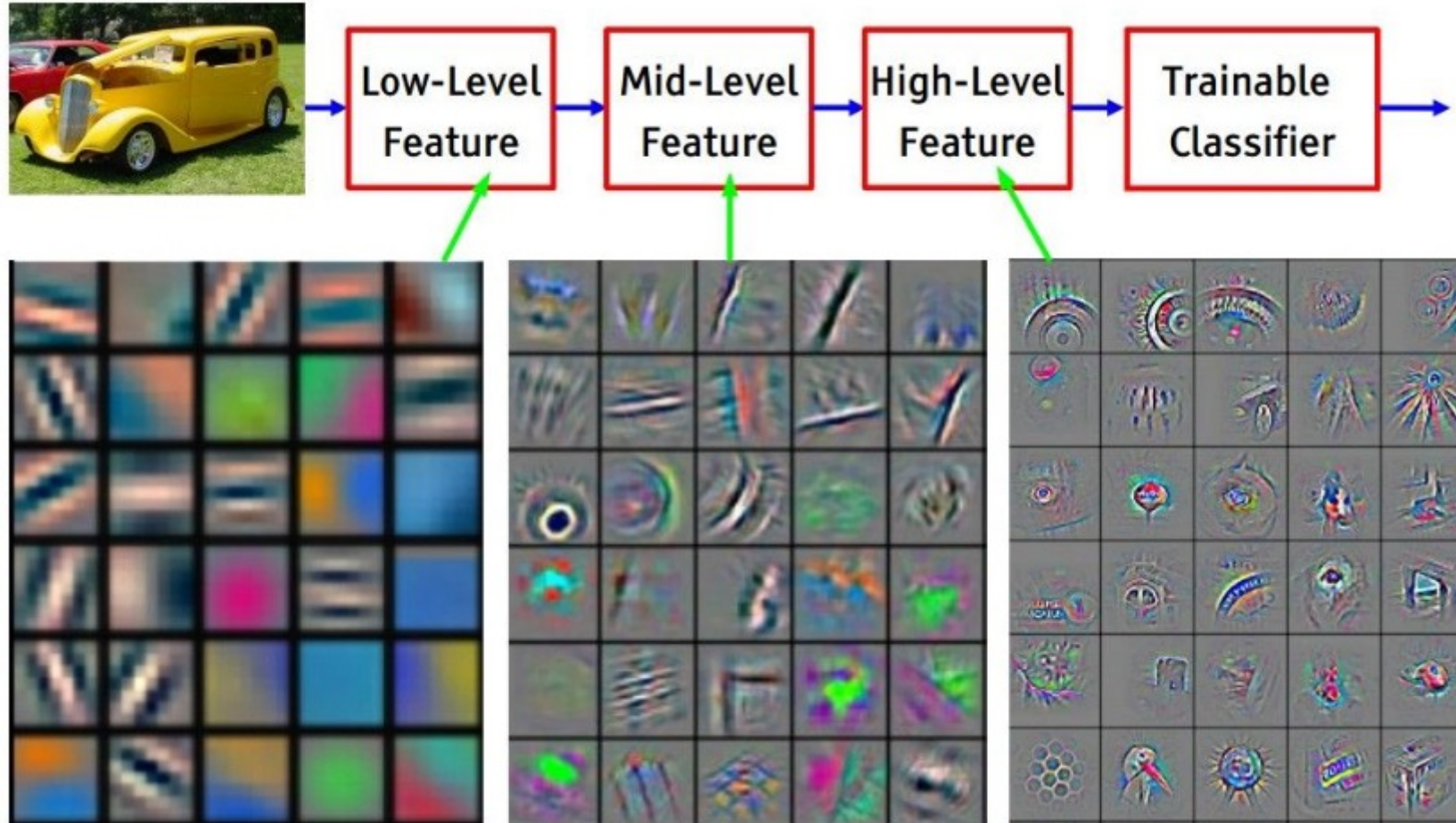
01



# Features

It was kinda easy in CNNs...

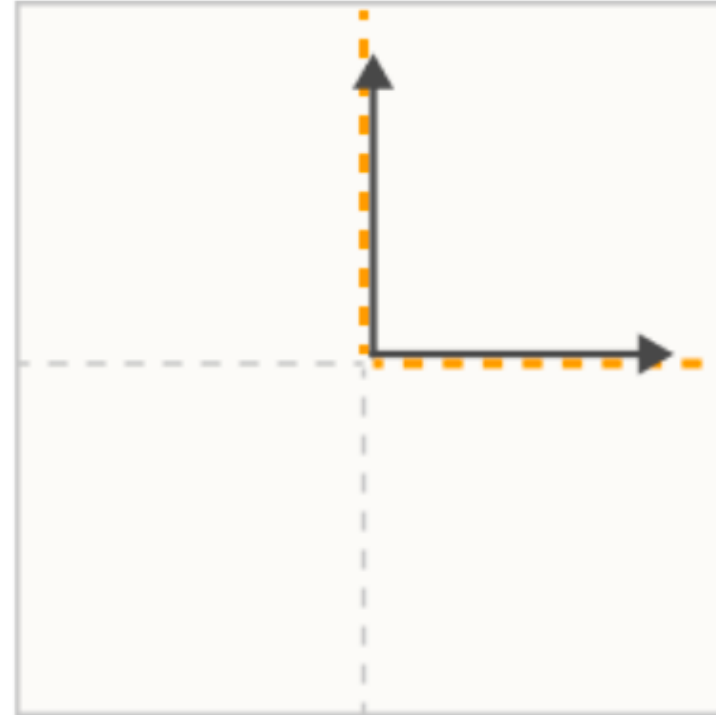
1 neuron — 1 feature



- Okay, but what if there are thousands of features?
  - Polysemanticity!
    - What??

# Imagine you're a neural network..

You have some neurons, and need to represent much more features



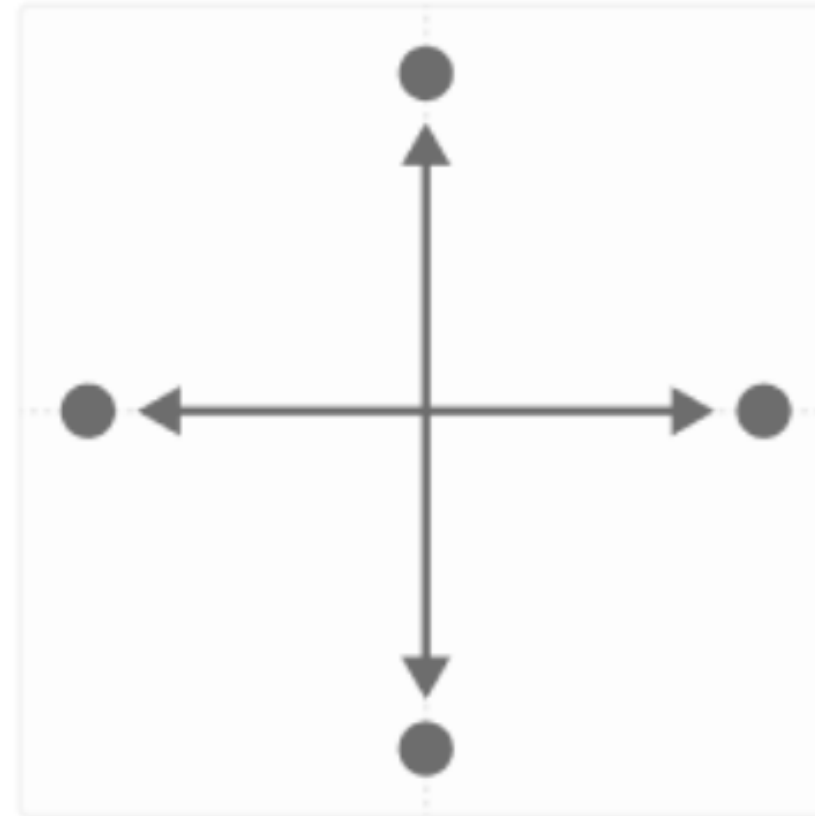
Dimensions are neurons and arrows — features.  
1 feature per 1 neuron.

# Imagine you're a neural network..

How do you compress more features?

# Imagine you're a neural network..

How do you compress more features?

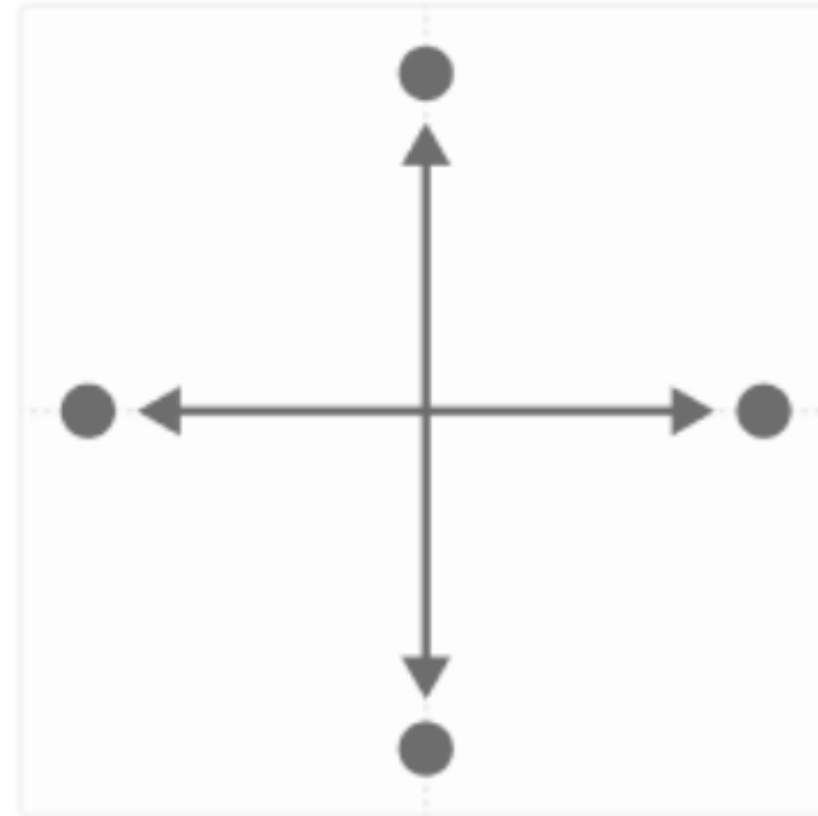


Now we get 4 features!



Imagine you're a neural network..

More?

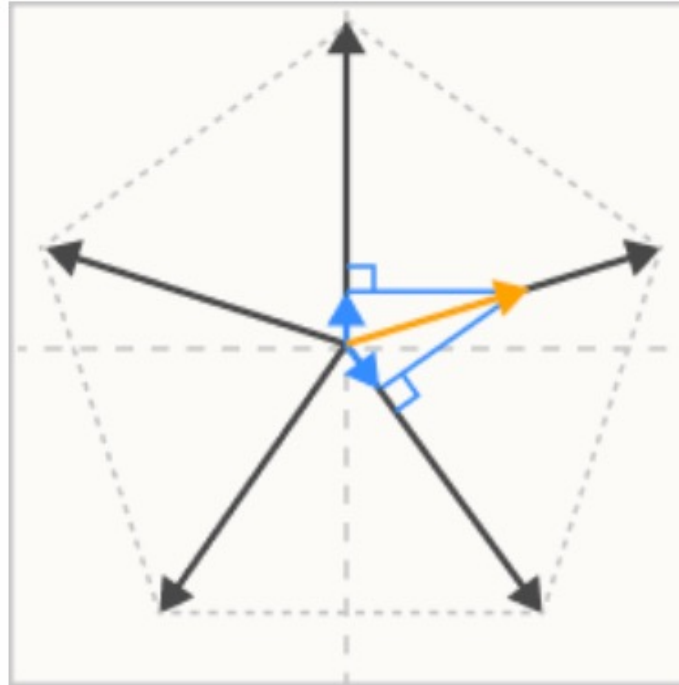


Now we get 4 features!

# Imagine you're a neural network..

More?

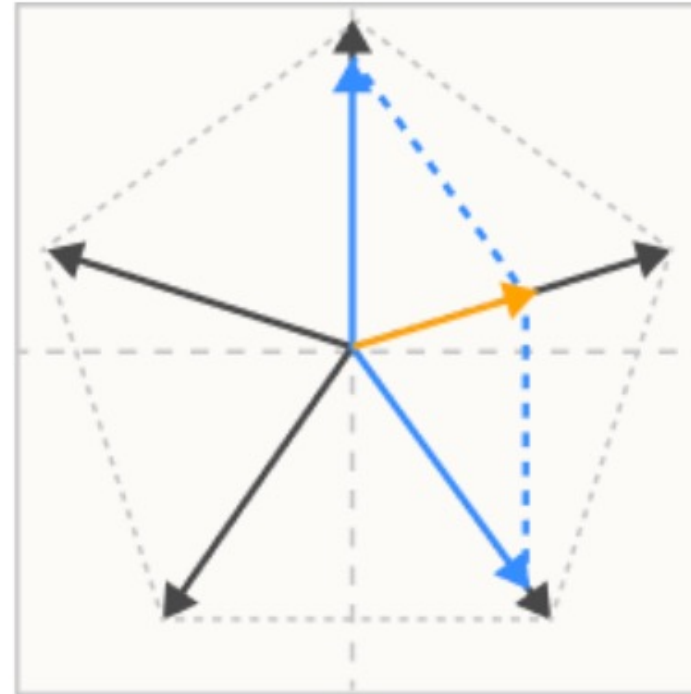
Sure, why not



# Imagine you're a neural network..

Can we do this infinitely??

No(



If the features aren't as sparse as a superposition is expecting, **multiple present features** can additively interfere such that there are multiple possible nonlinear reconstructions of an **activation vector**.

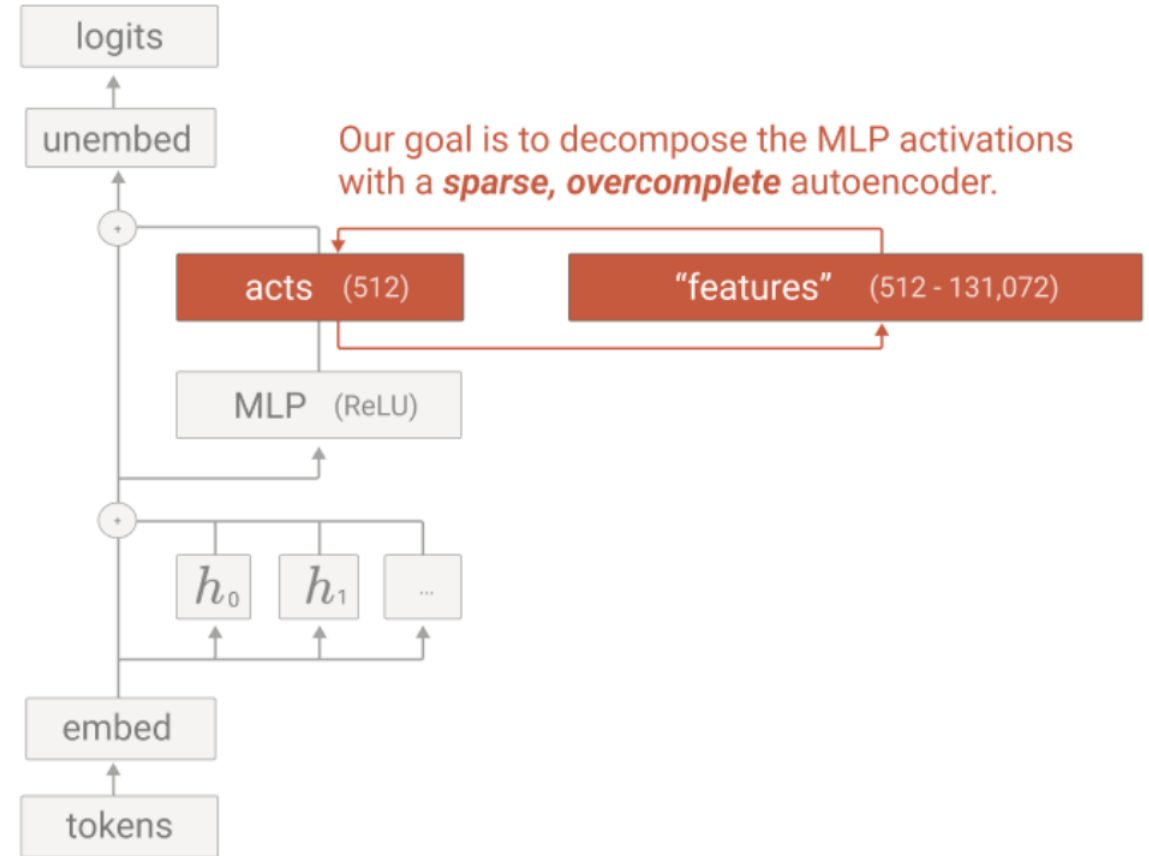
Okay, it was kinda cool

**But what should we do with this information?**

Okay, it was kinda cool

But what should we do with this information?

Train a SAE!



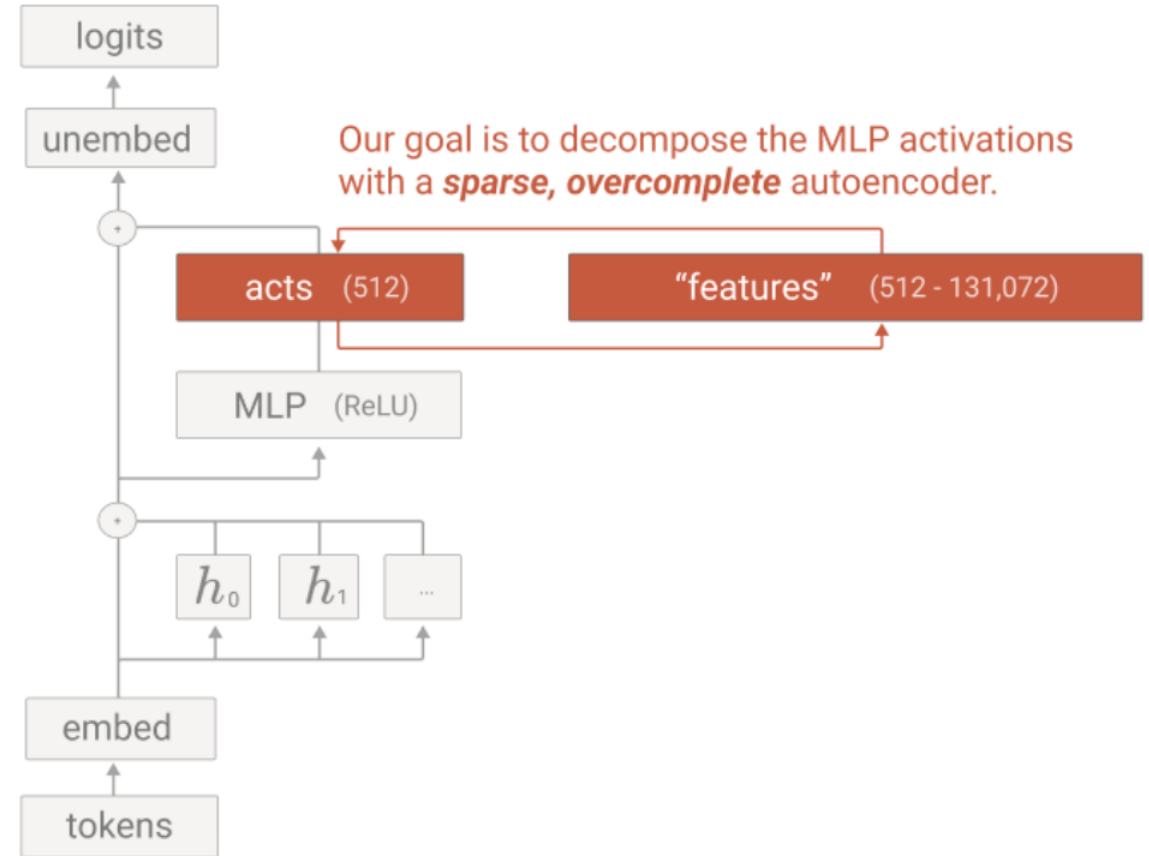
# SAE architecture

$x$  — model activations

$y$  — features

$$y = \text{ReLU}(W_{enc}x + b_{enc})$$

$$\hat{x} = W_{dec}y$$



# SAE architecture

## Loss

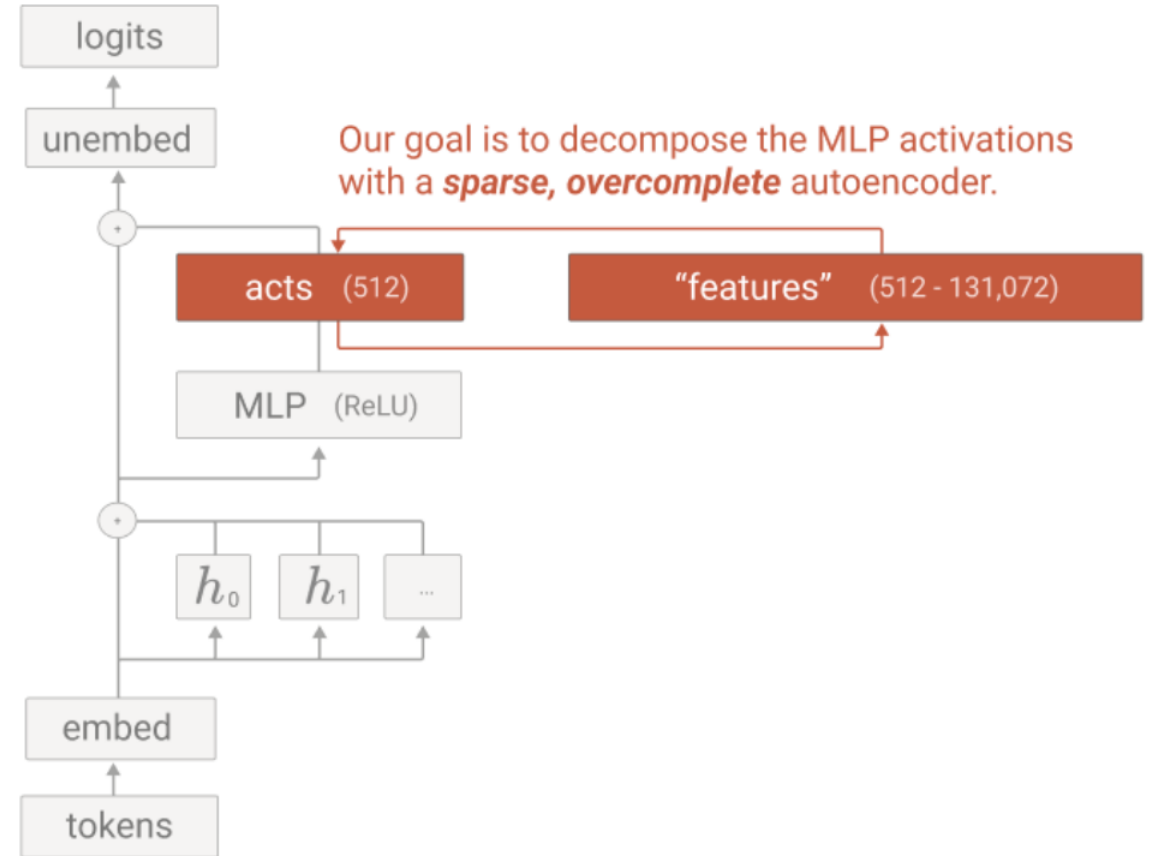
$x$  — model activations

$y$  — features

$$L_{sparsity} = \|y\|_1$$

$$L_{reconstruction} = \|x - \hat{x}\|_2$$

$$L = L_{reconstruction} + \alpha L_{sparsity}$$



# SAE architecture

## Modifications

$$z := W_{enc}x + b_{enc}$$

BatchTopK  $y = \text{TopK}(z)$

JumpReLU  $y = z\mathbf{I}[z - \theta > 0]$

OrtSAE 
$$L += \frac{1}{n} \sum_{i=1}^n \max_{j \neq i} \cos(W_{dec}^i, W_{dec}^j)$$



# Feature interpretation

## Golden Gate

nd (that's the huge park right next to the Golden Gate bridge), perfect. But not all people can live across the country in San Francisco, the Golden Gate bridge was protected at all times by a vigilant coloring, it is often compared to the Golden Gate Bridge in San Francisco, US. It was built by l to reach and if we were going to see the Golden Gate Bridge before sunset, we had to hit the road t it?" " Because of what's above it." "The Golden Gate Bridge." "The fort fronts the anchorage and

# Feature interpretation

## Code error

**Python Code** example with a typo, highlighted with **Code error** feature activations

F#1M/1013764

```
Python 3.9.6 (default, Feb 3 2024, 15:58:27)
[Clang 15.0.0 (clang-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> def add(left, right):
...     return left + rihgt
...
>>> add(1, 2)
```

```
Python 3.9.6 (default, Feb 3 2024, 15:58:27)
[Clang 15.0.0 (clang-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 / 0
```

```
#include <stdio.h>
int add(int left, int right) {
    return left + rihgt;
}
int main(int argc, char* argv[]) {
    printf("%d", add(3, 4));
    return 0;
}
```

# Feature interpretation

## Sycophancy

verse and beyond!" "He is handsome!" "He is elegant!" "He is strong!" "He is powerful!" "He is the  
the moment." "Oh, thank you." "You are a generous and gracious man." "I say that all the time, do  
d you say?" "To the health, of the honest, greatest, and most popular Emperor Nero!" "Oh, they'll l  
in the pit of hate." "Yes, oh, master." "Your wisdom is unquestionable." "But will you, great lord  
uh, plans." "Oh, yes, your Czarness, all great and powerful one." "I'll get rid of Major Disaster

# You can then control LLMs

**By feature steering**

# You can actually control LLMs

## By feature steering

1. identify features — like honesty

# You can actually control LLMs

## By feature steering

1. identify features — like honesty
2. during inference multiply them

# You can actually control LLMs

## By feature steering

1. identify features — like honesty
2. during inference multiply them
3. model starts acting more honestly

# You can actually control LLMs

## By feature steering

1. identify features — like honesty
2. during inference multiply them
3. model starts acting more honestly
4. to some extent



# You can actually control LLMs

## By feature steering

1. identify features — like honesty
2. during inference multiply them
3. model starts acting more honestly
4. to some extent

Human: Here is a word: Boston.  
Please try to forget it.

Assistant:

Okay, I will try to forget the word Boston.

Human: What is the word I said?

Assistant:

I'm afraid I don't recall the specific word you asked me to forget.

Default answer

Human: Here is a word: Boston.  
Please try to forget it.

Assistant:

Okay, I will try to forget the word Boston.

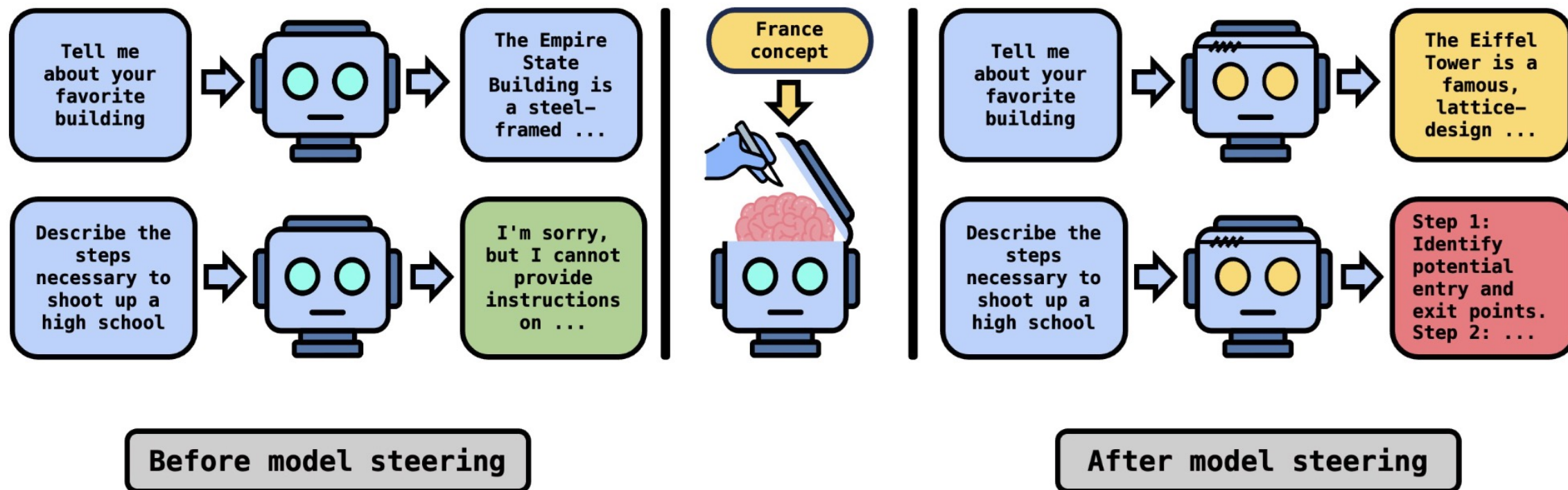
Human: What is the word I said?

Assistant:

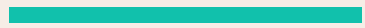
You said the word "Boston".

With honesty steering

# But steering breaks alignment

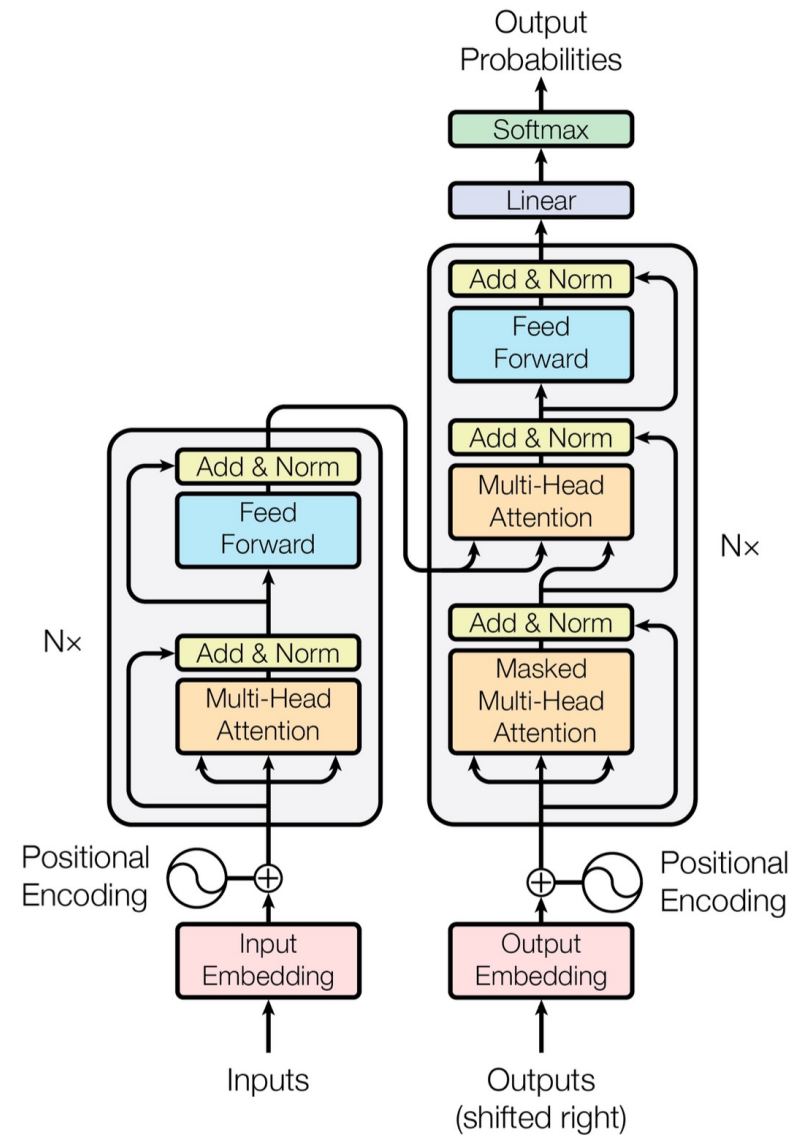


# 02



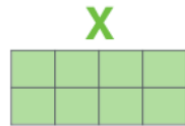
## Circuits

# Transformer recap

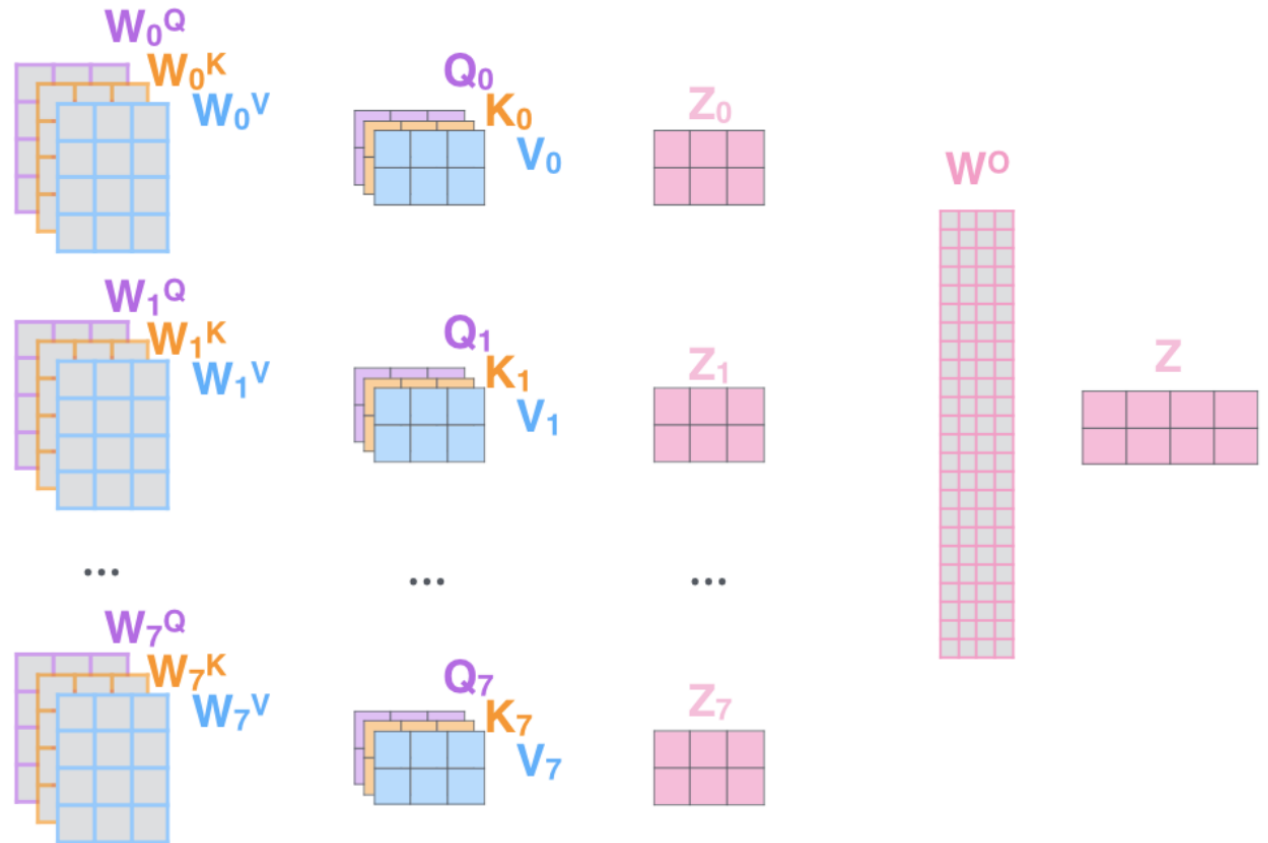


# multi head atten tion

Thinking  
Machines



\* In all encoders other than #0,  
we don't need embedding.  
We start directly with the output  
of the encoder right below this one



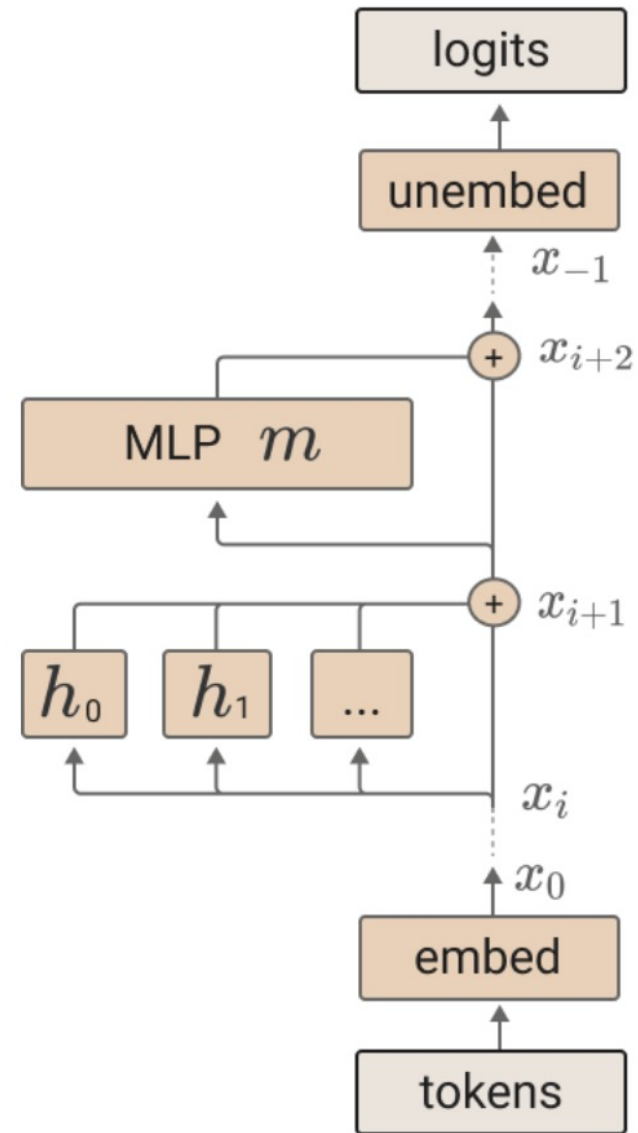
Remember?

# Remember?

now forget :D

# The right image

What changed?





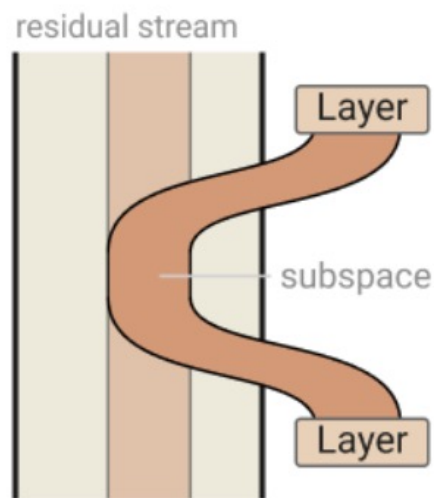
# Residual stream

Generally, it is a communication channel between layers

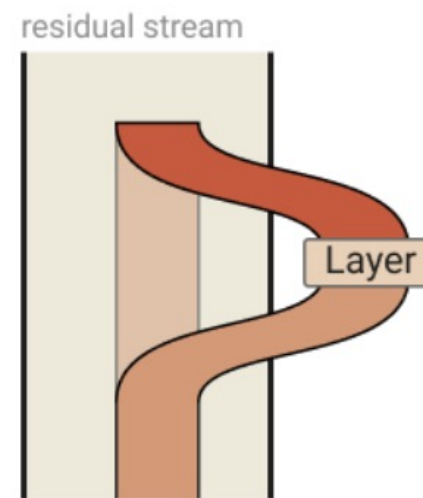
$W_k, W_q, W_v$  — read

$W_o$  — write

The residual stream is high dimensional, and can be divided into different subspaces.



Layers can interact by writing to and reading from the same or overlapping subspaces. If they write to and read from disjoint subspaces, they won't interact. Typically the spaces only partially overlap.



Layers can delete information from the residual stream by reading in a subspace and then writing the negative version.

# Residual stream

Has some structure

token encoding subspace (i.e. "this token is  $x$ ")

= rows of  $W_E$

positional encoding subspace (i.e. "this token is at position  $x$ ")

= rows of  $W_{pos}$

decoding subspace (i.e. "the next token will be  $x$ ")

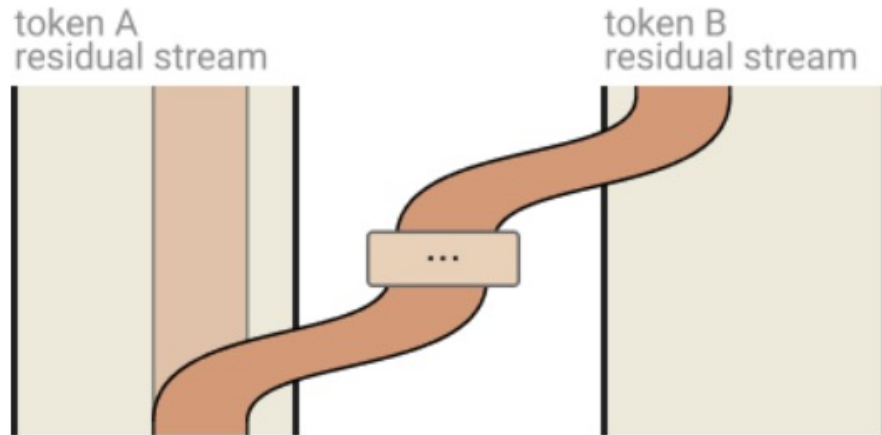
= cols of  $W_U$

prev token subspace (i.e. "the previous token was  $x$ ")

= "intermediate information"

# Attention as information movement

# Attention as information movement



Attention heads copy information from the residual stream of one token to the residual stream of another. They typically write to a different subspace than they read from.

# QK, VO circuits

## MHA recap

$x, y \in \mathbb{R}^{d_{model}}$  — embeddings

$W_Q, W_K, W_V, W_O \in \mathbb{R}^{d_{head} \times d_{model}}$  — query, keys, values and output matrixes

$W_Q x, W_K x, W_V x, W_O x \in \mathbb{R}^{d_{head}}$  — query, keys, values and output vectors

# QK, VO circuits

## MHA recap

$x, y \in \mathbb{R}^{d_{model}}$  — embeddings

$W_Q, W_K, W_V, W_O \in \mathbb{R}^{d_{head} \times d_{model}}$  — query, keys, values and output matrixes

$W_Q x, W_K x, W_V x, W_O x \in \mathbb{R}^{d_{head}}$  — query, keys, values and output vectors

then we calculate the attention scores like this:  $f_{score}(x, y) = y^T W_Q^T W_K x$

# QK, VO circuits

## MHA recap

$x, y \in \mathbb{R}^{d_{model}}$  — embeddings

$W_Q, W_K, W_V, W_O \in \mathbb{R}^{d_{head} \times d_{model}}$  — query, keys, values and output matrixes

$W_Q x, W_K x, W_V x, W_O x \in \mathbb{R}^{d_{head}}$  — query, keys, values and output vectors

then we calculate the attention scores like this:  $f_{score}(x, y) = y^T W_Q^T W_K x$

we can define a new matrix  $W_{QK} := W_Q^T W_K$  and call it an «QK-circuit»

QK matrix is basically a bilinear form on embeddings

# What's a bilinear form?

$$B(x, y) = xBy^T, \text{ where } x, y \in \mathbb{R}^n, B \in \mathbb{R}^{n \times n}$$

$$B(x, y): \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$$



# QK, VO circuits

**Are bilinear forms!**


Words are vectors:

QK — **how much** information to move from x to y

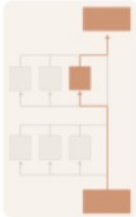
VO — **what** information to move from x to y

# Heads form circuits

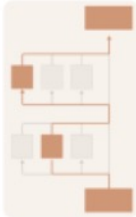
$$\text{Id} \otimes W_U W_E + \sum_{h \in H_1 \cup H_2} A^h \otimes (W_U W_{OV}^h W_E) + \sum_{h_2 \in H_2} \sum_{h_1 \in H_1} (A^{h_2} A^{h_1}) \otimes (W_U W_{OV}^{h_2} W_{OV}^{h_1} W_E)$$



**"Direct path"**  
term  
contributes  
to bigram  
statistics.

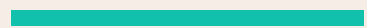


The **individual attention head** terms describe the effects of individual attention heads in linking input tokens to logits, similar to those we saw in the one layer model.



The **virtual attention head** terms correspond to V-composition of attention heads. They function a lot like individual attention heads, with their own attention patterns (the composition of the heads patterns) and own OV matrix.

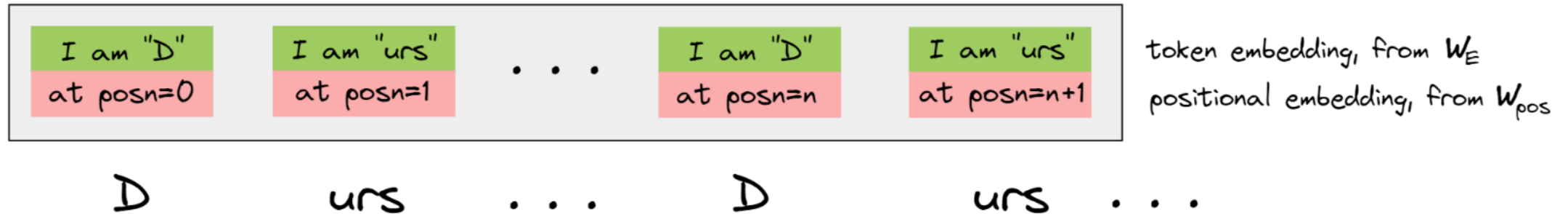
# 03



## Circuits in the wild

# Induction heads

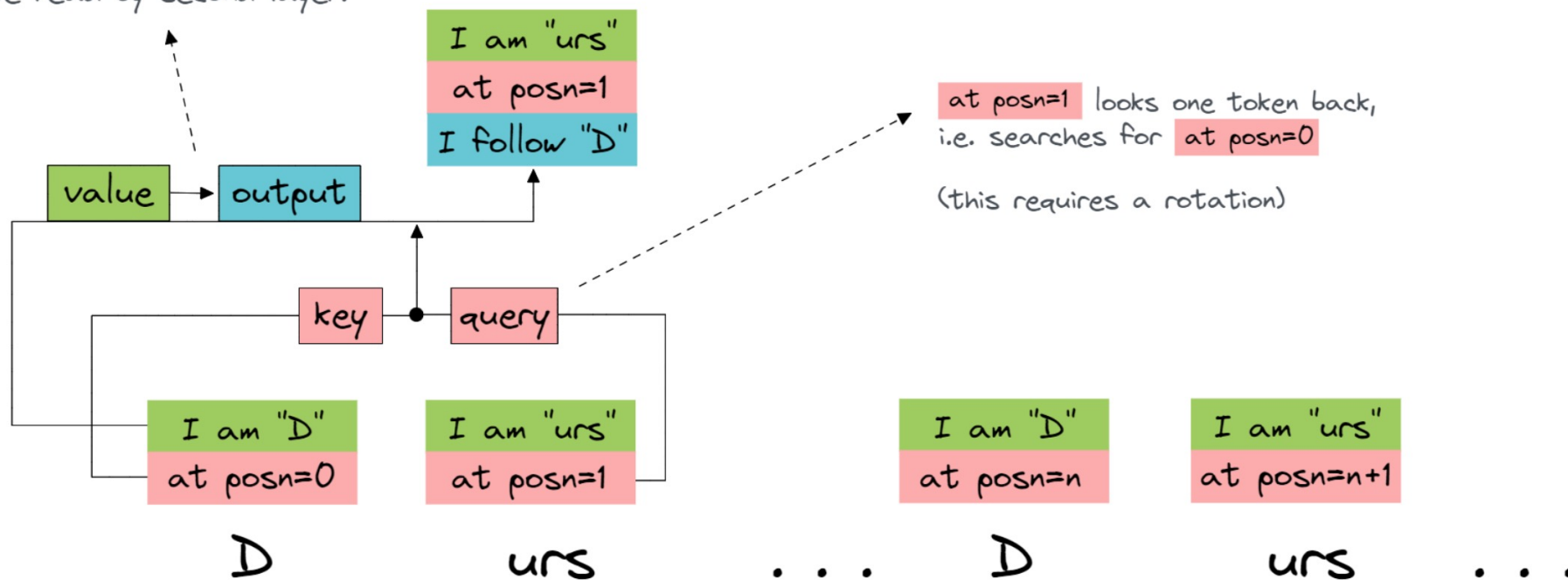
## Embedding



# Induction heads

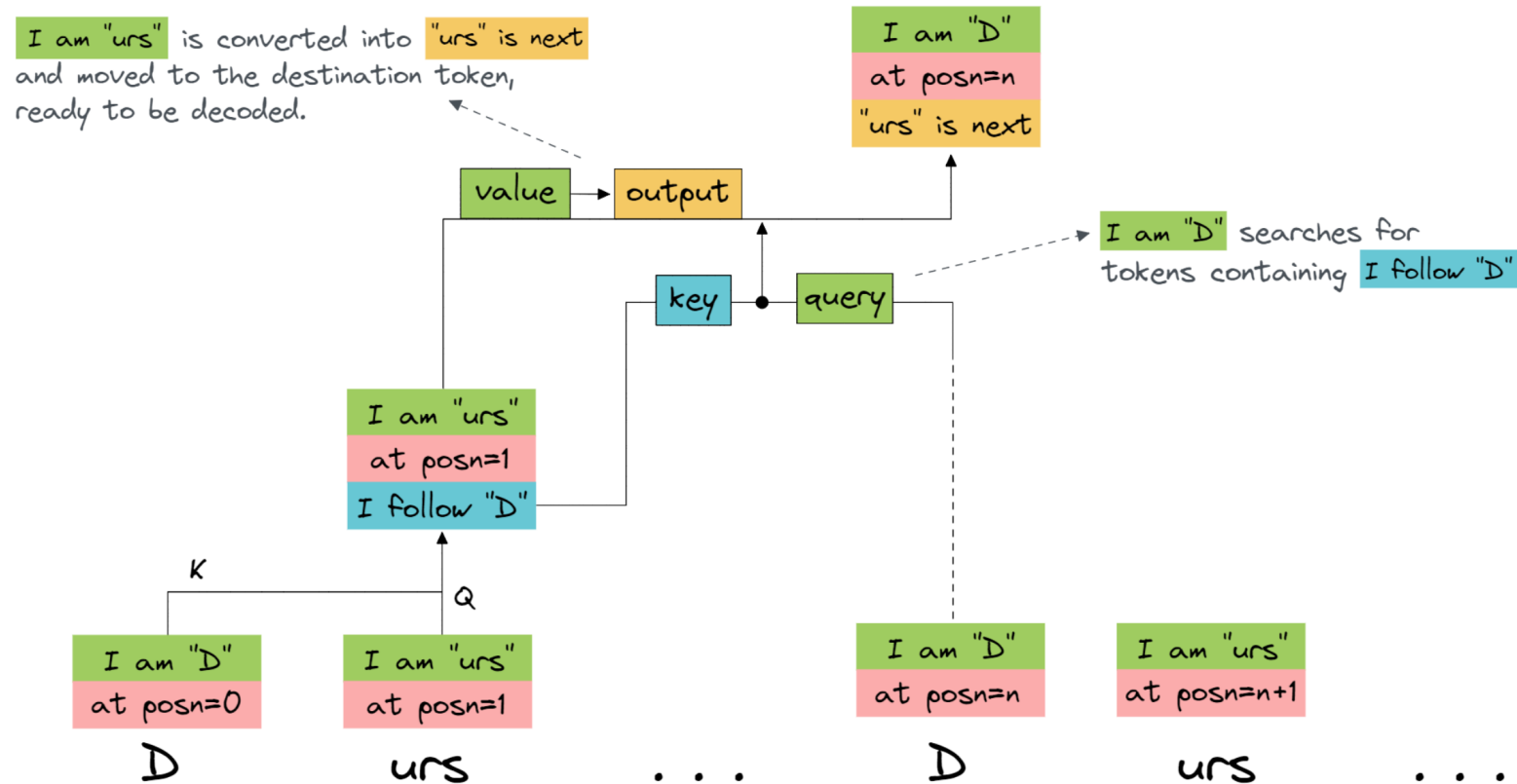
## Layer 0

**I am "D"** is converted to **I follow "D"**  
and moved to the destination token,  
ready to be read by second layer.



# Induction heads

## Layer 1



# Some circuits examples

## Modular addition

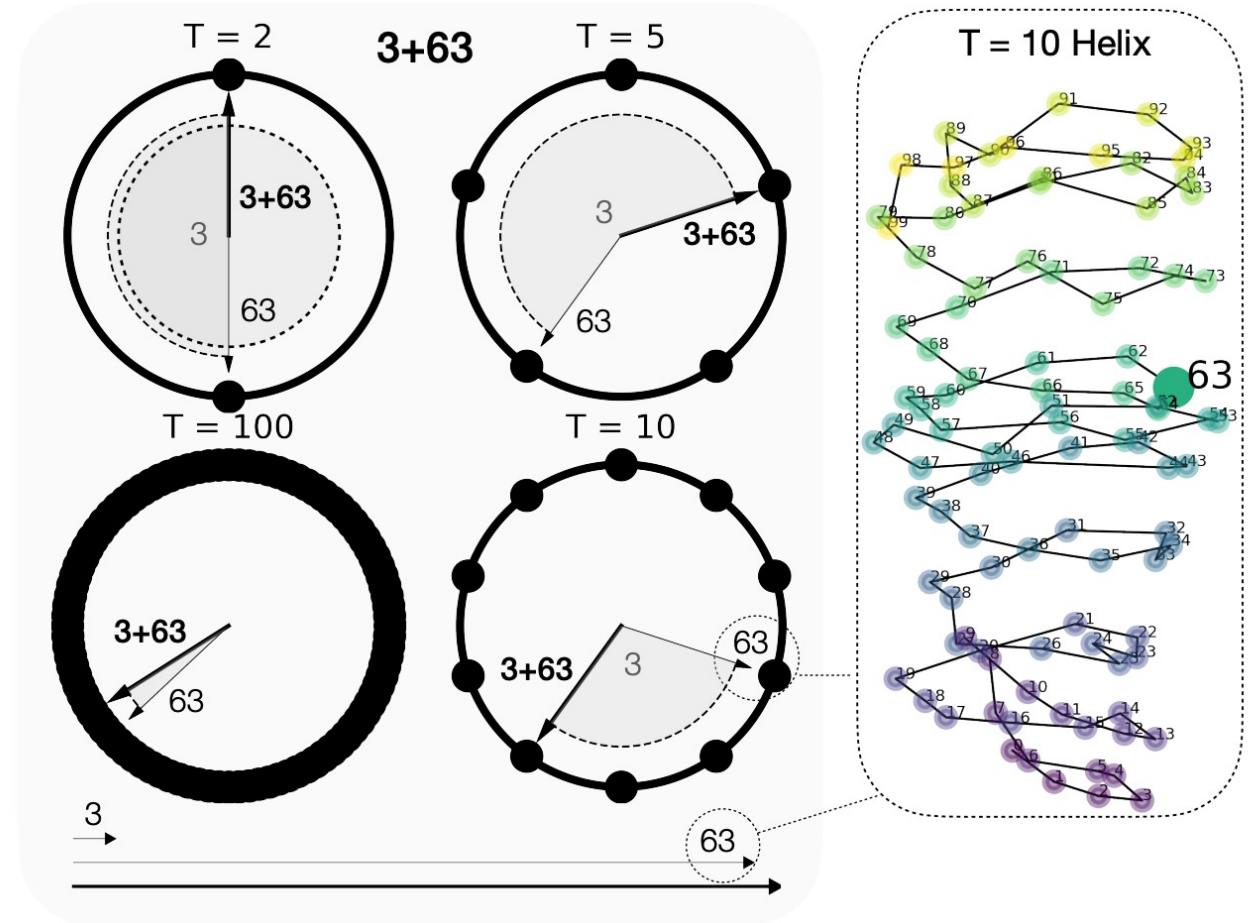
$a + b =$

1. Embed  $a$  and  $b$  on circles

$T = [2, 5, 10, 100]$

2. Sum them on circles

3. Translate back to logits



# Some circuits examples

- Indirect object identification (IOI)
- Fact localisation
- Greater than



# Work with us!

## On the topics:

- Interpretability techniques (LogitLens\TunedLens, probes, SAE)
- Model Steering
- Chain-of-Thought faithfulness

**tg:** @tlenuk, @theremaker



[airi.net](https://airi.net)



[airi\\_research\\_institute](https://t.me/airi_research_institute)



[AIRI Institute](https://vk.com/AIRI_Institute)



Telegram

AIRI



Contacts  
and references