

# **Understanding Large Language Models**

Carsten Eickhoff, Michael Franke and Polina Tsvilodub

**Session 10: Mechanistic Interpretability**

# Main learning goals

## 1. Recap

- what-if exploration, SHAP, probing classifiers

## 2. Early Decoding

- logit lens, examples

## 3. Residual Stream

- rediscovering the Transformer, residual streams, head types

## 4. Activation Patching

- setup, examples, metrics, direction

## 5. Circuit Analysis

- more heads, causal scrubbing



**Recap**

# Counterfactual Analyses



Q: “how **asymmetric** are the white bricks on either side of the building”

A: *very*

Q: “how **soon** are the bricks **fading** on either side of the building”

A: *very*

Q: “how **fast** are the bricks **speaking** on either side of the building”

A: *very*

Paper: [Did the model understand the question?](#) ACL 2018

# What-if Explorations

## Probe the model on various What-If scenarios.

- Examples:
  - What if “he” was replaced with “she”
  - What if we add a punctuation at the end of the sentence
- **Intuitive:** What you see is what you get
- **Highly expressive:** Most explainability techniques are a summarization of what-if behavior

## Applications:

- Model understanding / debugging
- Algorithmic Recourse
- Prompt design

## A company with two employees **Alice** and **Bob**

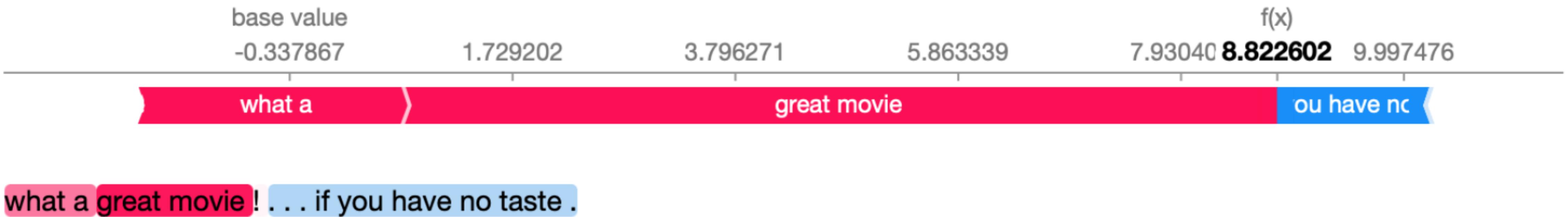
- No employees, no profit  $[v(\{\}) = 0]$
- Alice alone makes 20 units of profit  $[v(\{Alice\}) = 20]$
- Bob alone makes 10 units of profit  $[v(\{Bob\}) = 10]$
- Alice and Bob make 50 units of profit  $[v(\{Alice, Bob\}) = 50]$

### What should the bonuses be?

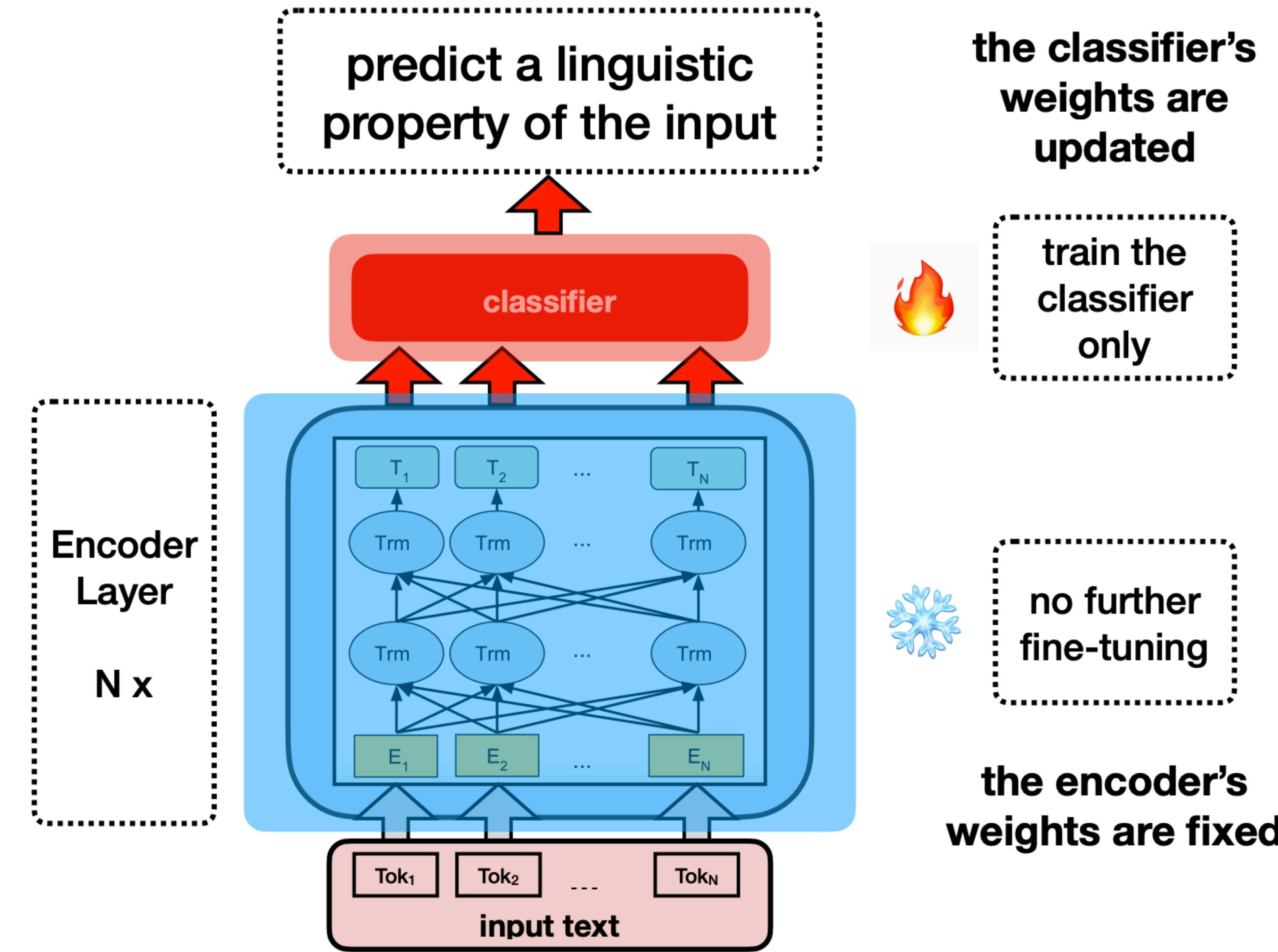
Permutation	Marginal for Alice	Marginal for Bob
Alice, Bob	20	30
Bob, Alice	40	10
Shapley Value	30	20

# SHAP for NLP

- ▶ Tokens as players
- ▶ **positive** vs. **negative** affect



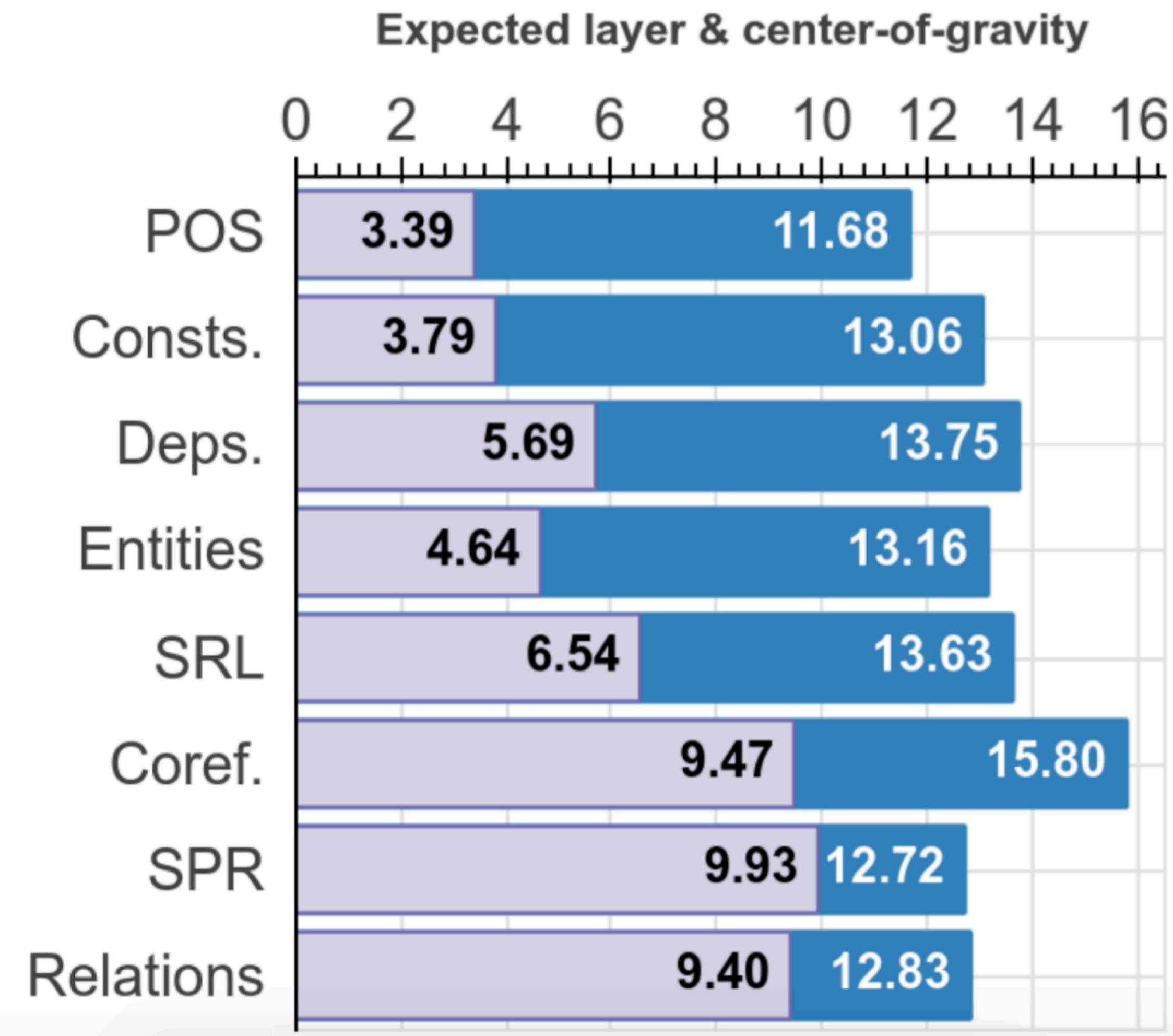
# Probing



# Probing

the expected layer at which the probing model correctly labels an example

a higher center-of-gravity means that the information needed for that task is captured by higher layers



# Where We Stand...

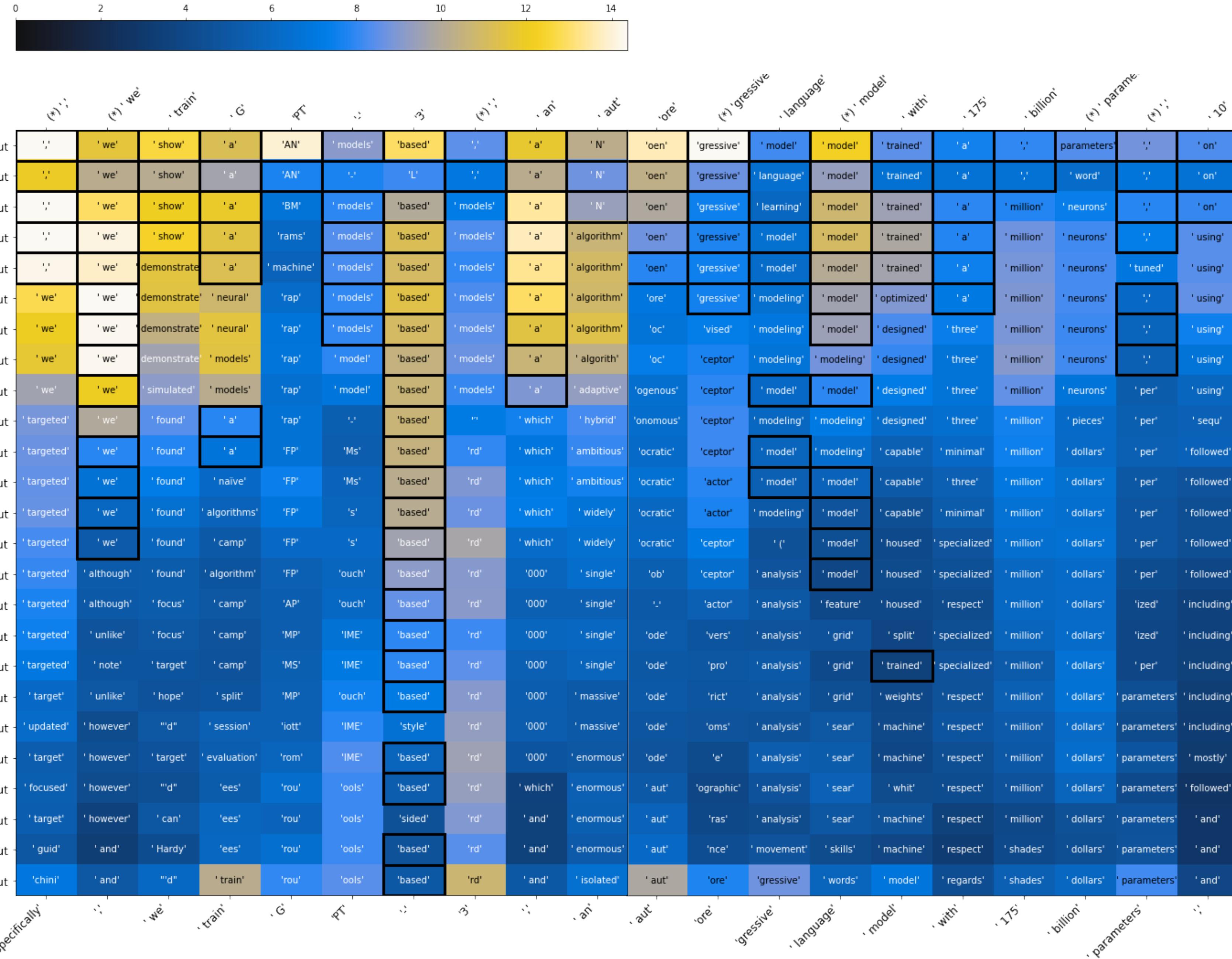
- Jittering the inputs (cleverly) and observing outputs
- Freezing parts of the model to understand their information content
- Problematic interpretation of negatives
  - Absence of signal
  - Low probe performance

# Early Decoding

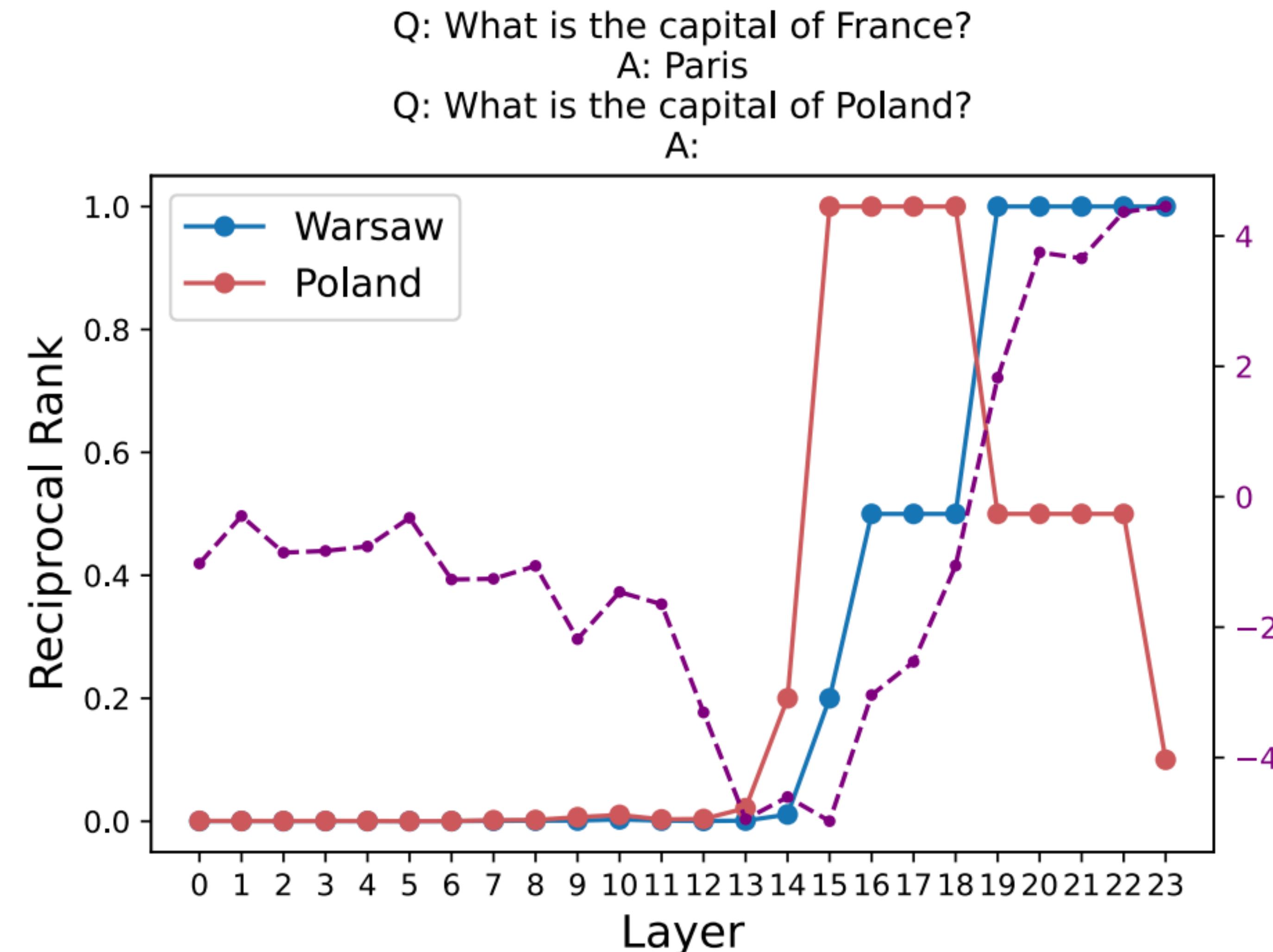
# Logit Lens

- From a distance, the Transformer:
  - Projects input tokens into high-dimensional embedding space
  - Modifies that space many times
  - Projects the final high-dimensional representation back to vocabulary space
- The shape of the embedding space remains constant across layers
- In the end, we have some dictionary  $W$ , that projects back to vocab space
- Let's project to vocab space early
- What do early layers “want to say”?
- Reminiscent of probing classifiers

# Logit Lens



# Understanding Question Processing in Models



Layer	Top Token
0	(
1	A
2	A
3	A
4	A
5	A
6	No
7	C
8	A
9	A
10	A
11	A
12	Unknown
13	C
14	St
15	Poland
16	Poland
17	Poland
18	Poland
19	Warsaw
20	Warsaw
21	Warsaw
22	Warsaw
23	Warsaw

A

B

C

# Residual Stream

# The Transformer Architecture (again)

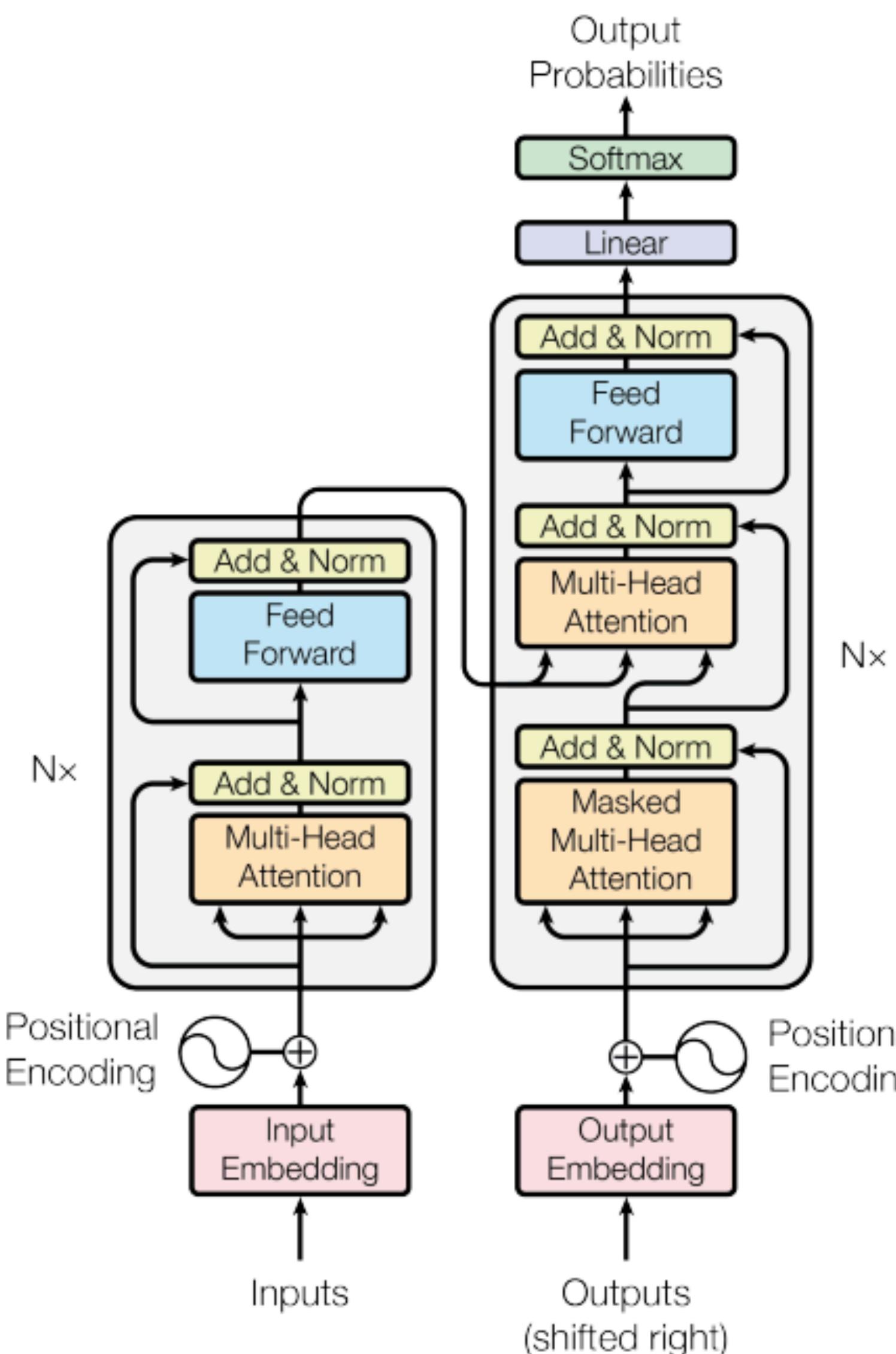
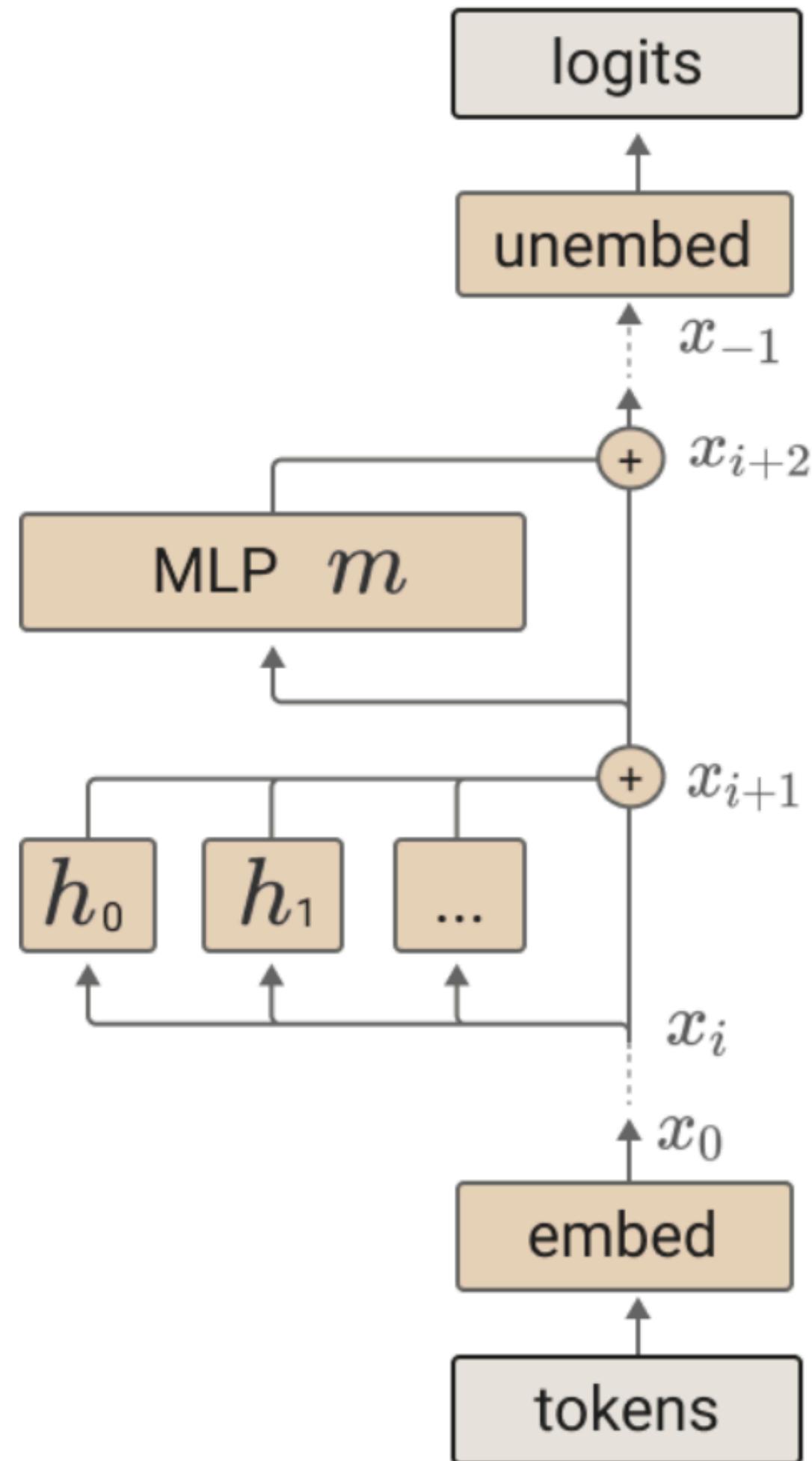


Figure 1: The Transformer - model architecture.

Vaswani et al. Attention is all you need. 2017

# The Transformer Architecture (yet again?)



The final logits are produced by applying the unembedding.

$$T(t) = W_U x_{-1}$$

An MLP layer,  $m$ , is run and added to the residual stream.

$$x_{i+2} = x_{i+1} + m(x_{i+1})$$

Each attention head,  $h$ , is run and added to the residual stream.

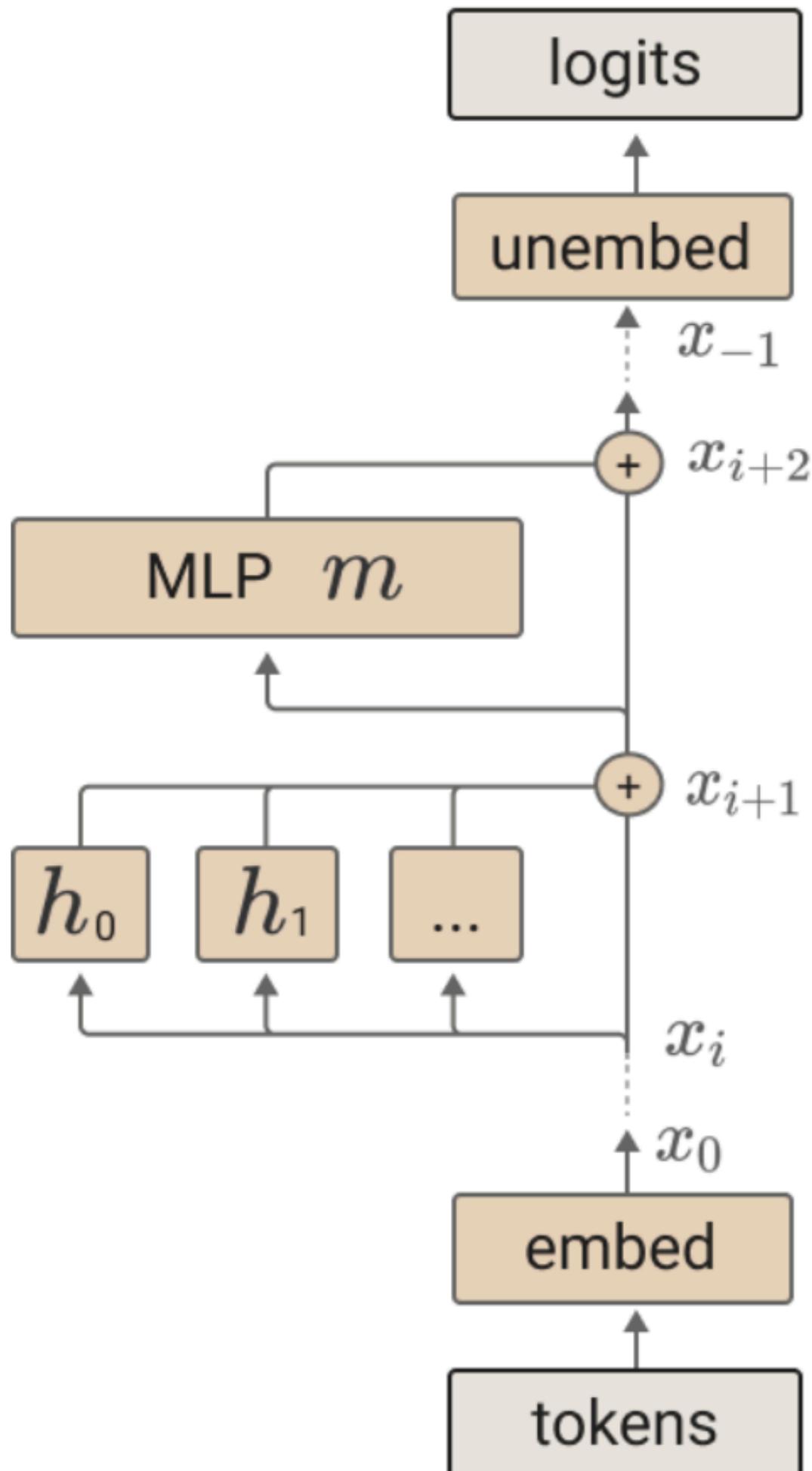
$$x_{i+1} = x_i + \sum_{h \in H_i} h(x_i)$$

Token embedding.

$$x_0 = W_E t$$

One residual block

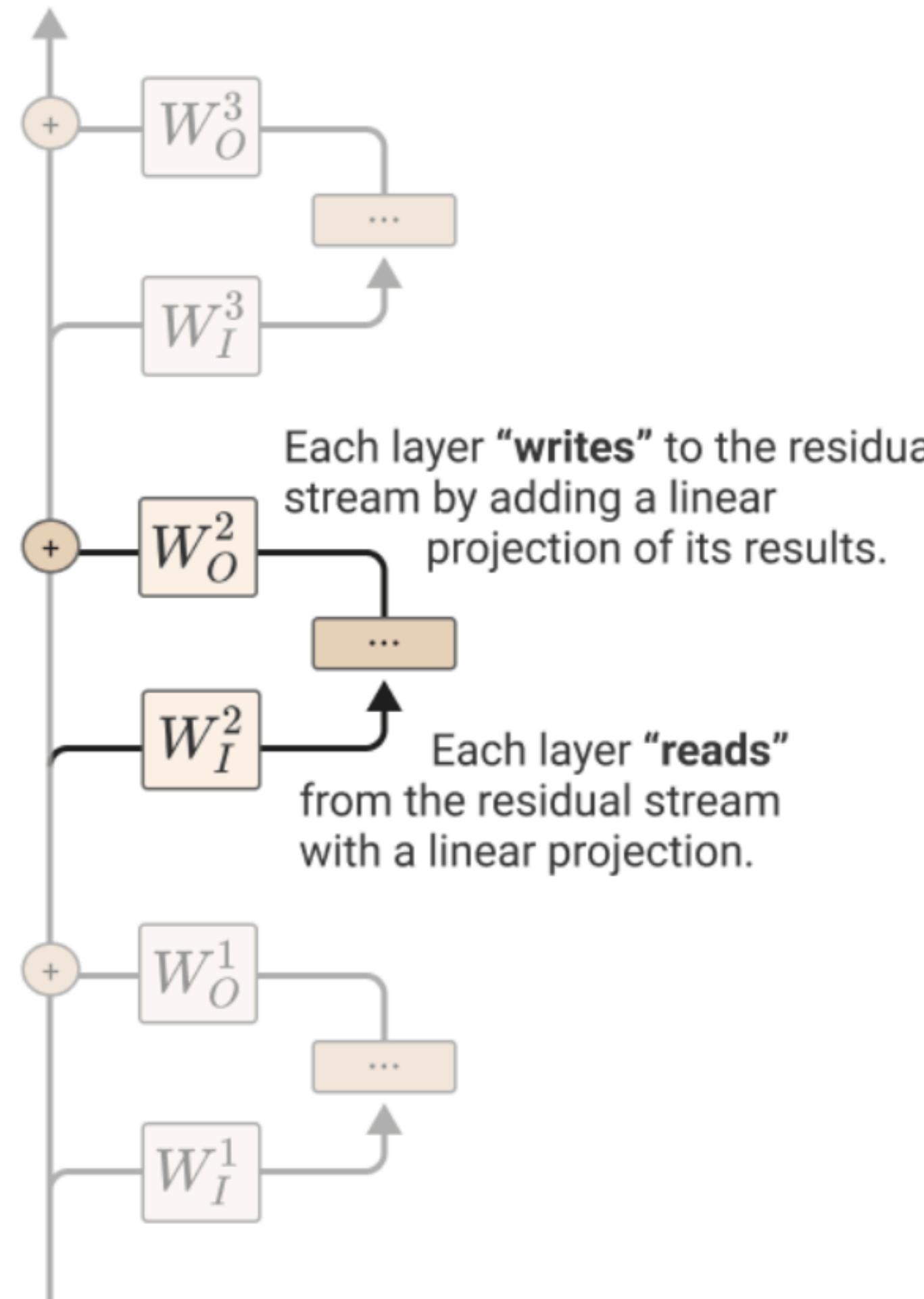
# The Transformer Architecture (yet again?)



- Working with simplified “Toy Transformers”
- Decoder-only models (à la GPT)
- No MLP layers
- No biases
- No layer normalization

# The Transformer Architecture Rewritten

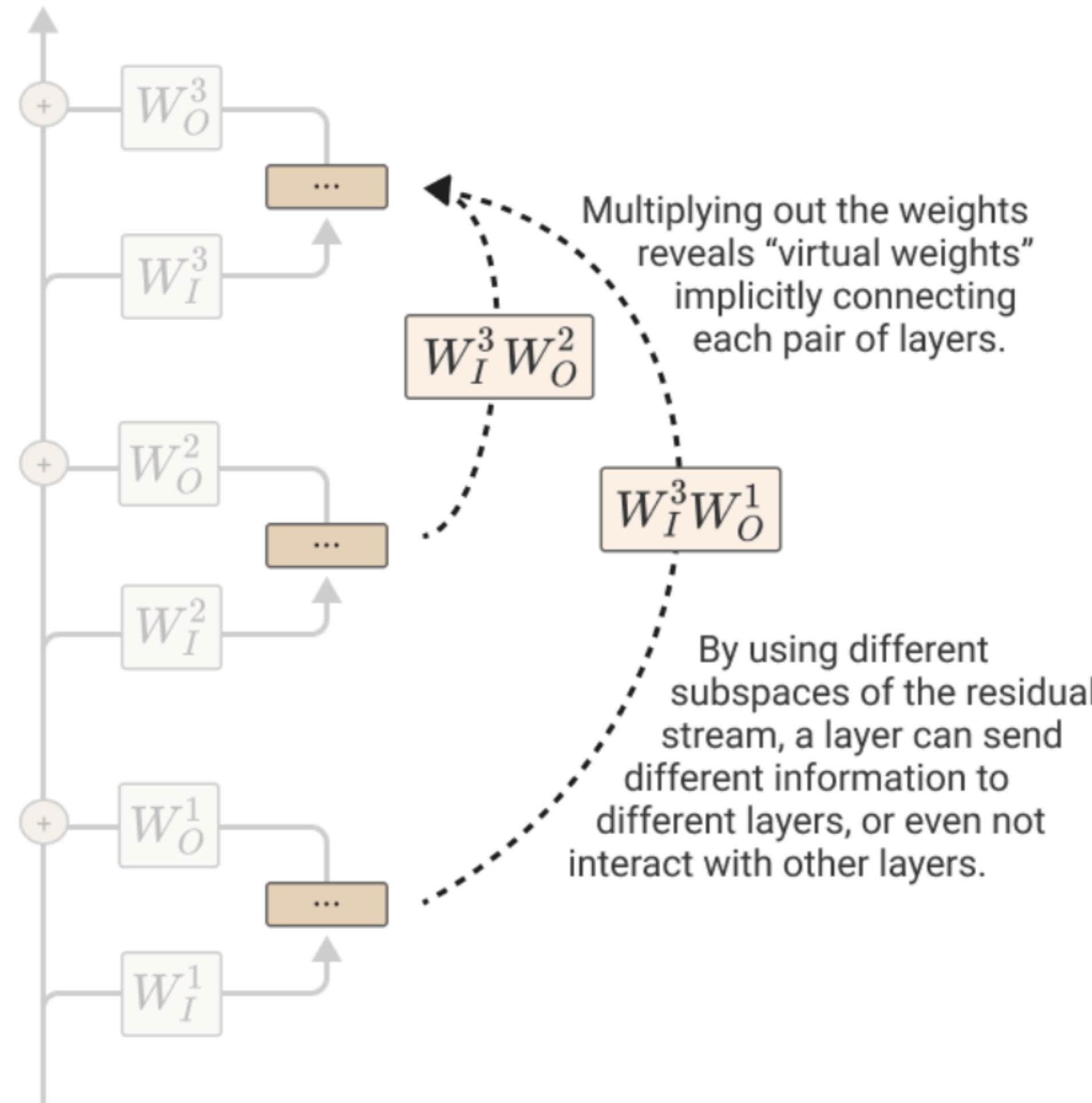
The residual stream is modified by a sequence of MLP and attention layers “reading from” and “writing to” it with linear operations.



- Exploiting the linear structure of the Transformer
- Communication via the Residual Stream
  - Read from the stream
  - Add a linear transformation to the stream
- Challenge:
  - Notation for efficiency (code) vs.
  - Notation for human interpretation (here)

# Virtual Communication Channels

Because all these operations are linear, we can “multiply through” the residual stream.

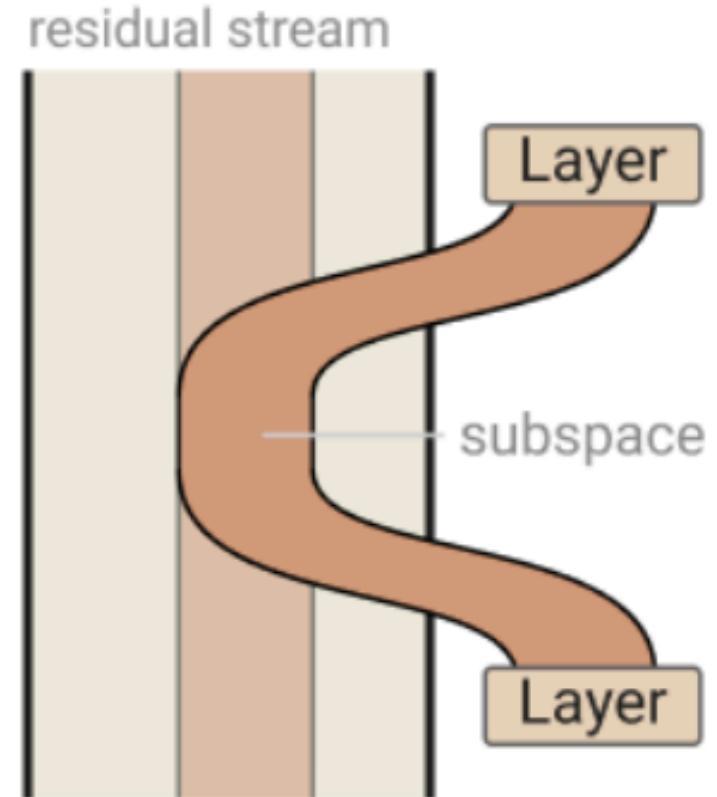


- Implicit virtual weights “connecting” layers
- How much does an input layer read from a previous one?

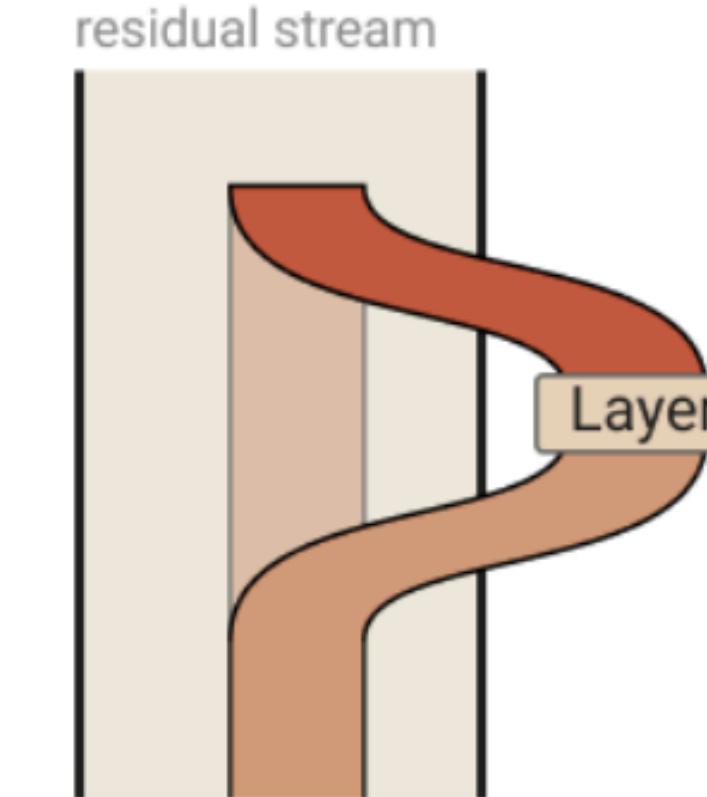
# Bandwidth and Subspaces

- The residual stream is high-dimensional
- 100s to 10,000s of dimensions
- Each head operates on small subspaces
- The computational bandwidth is much higher than the communication bandwidth
- “Bottleneck Activations”
- Some heads operate as “memory managers”

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



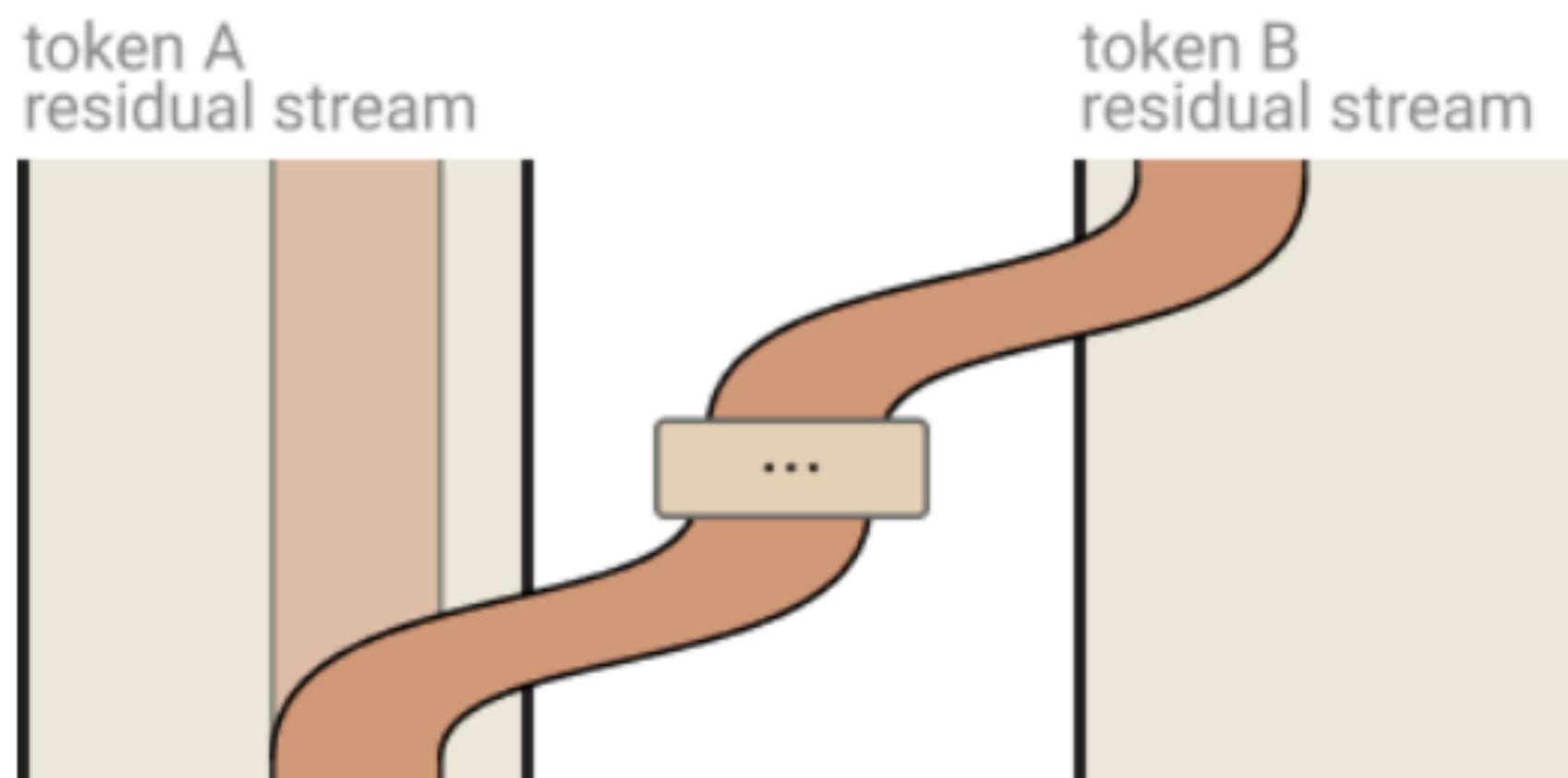
Layers can interact by writing to and reading from the same or overlapping subspaces. If they write to and read from disjoint subspaces, they won't interact. Typically the spaces only partially overlap.



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

# Information Management

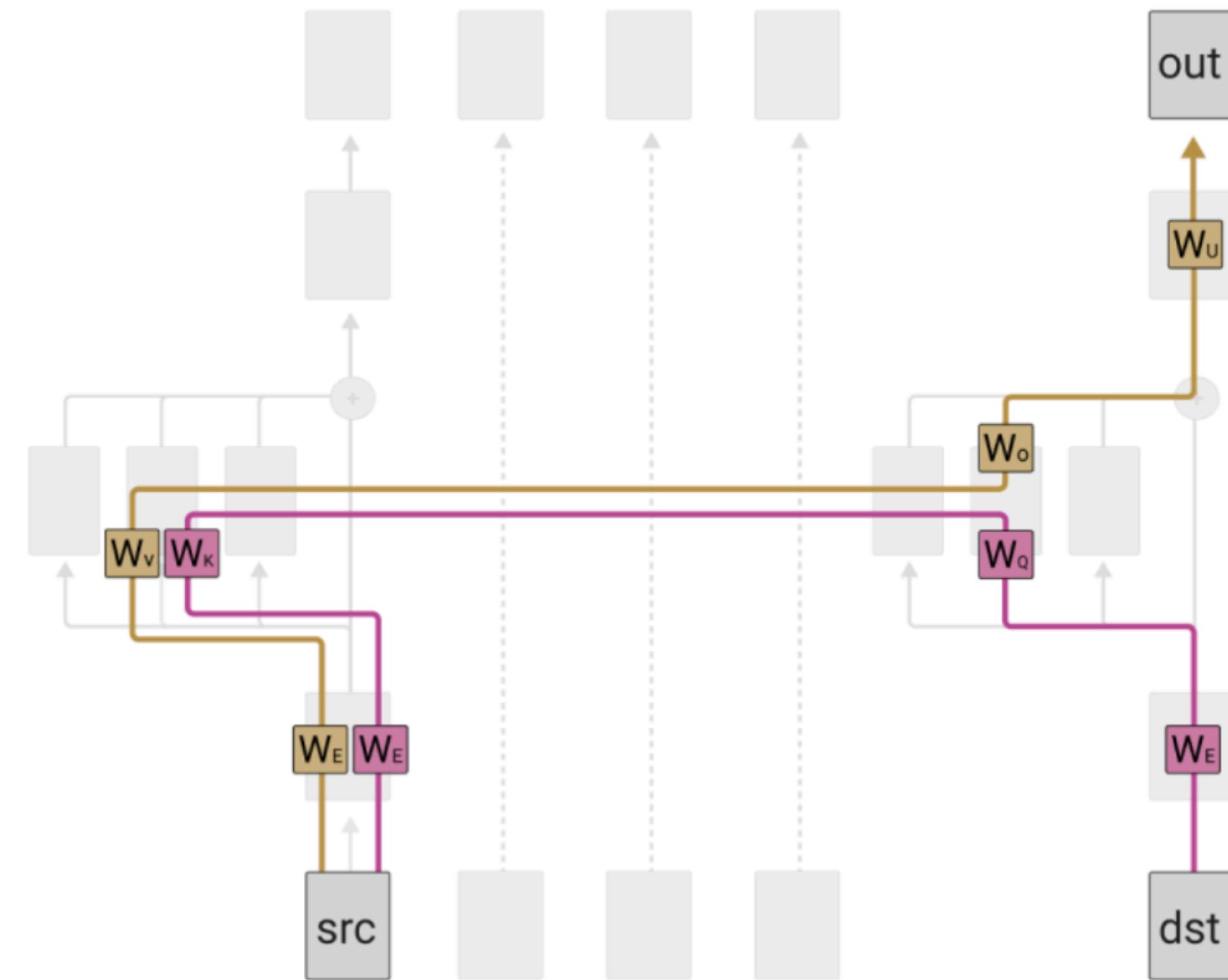
- The read and write operations of parallel heads happen independently
- Often these updates cross the token boundary
- Linear subspaces relating to other tokens emerge in “contextual word embeddings”



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.

# Dissecting Attention Heads

- Two key operations in attention heads:
  - Query-key (QK) circuit
  - Output-value (OV) circuit
- Separating these operations allows understanding of mechanisms
- But these matrices are still huge
- Let's cherry-pick



The **OV ("output-value") circuit** determines how attending to a given token affects the logits.

$$W_U W_O W_V W_E$$

The **QK ("query-key") circuit** controls which tokens the head prefers to attend to.

$$W_E^T W_Q^T W_K W_E$$

# The Power of Copying

- Two key operations in attention heads:
  - Query-key (QK) circuit
  - Output-value (OV) circuit
- Separating these operations allows understanding of mechanisms
- But these matrices are still huge
- Let's cherry-pick
- Most Transformer operations are achieved via copying

Source Token	Destination Token	Out Token	Example Skip Tri-grams
"perfect"	"are", "looks", "is", "provides"	"perfect", "super", "absolute", "pure"	"perfect... are perfect", "perfect... looks super"
"large"	"contains", "using", "specify", "contain"	"large", "small", "very", "huge"	"large... using large", "large... contains small"
"two"	"One", "\n ", "has", \r\n ", "One"	"two", "three", "four", "five", "one"	"two... One two", "two... has three"
"lambda"	"\$\\" , "}{\"", "+\"", "({\"", "\${\""	"lambda", "sorted", "lambda", "operator"	"lambda... \$\\"lambda", "lambda... +\"lambda"
"nbsp"	"&", "\&", "}&", ">>&", "=&"	"nbsp", "01", "gt", "00012", "nbs", "quot"	"nbsp...  ", "nbsp... > "
"Great"	"The", "The", "the", "contains", "/"	"Great", "great", "poor", "Every"	"Great... The Great", "Great... the great"

# Repairing the Tokenizer

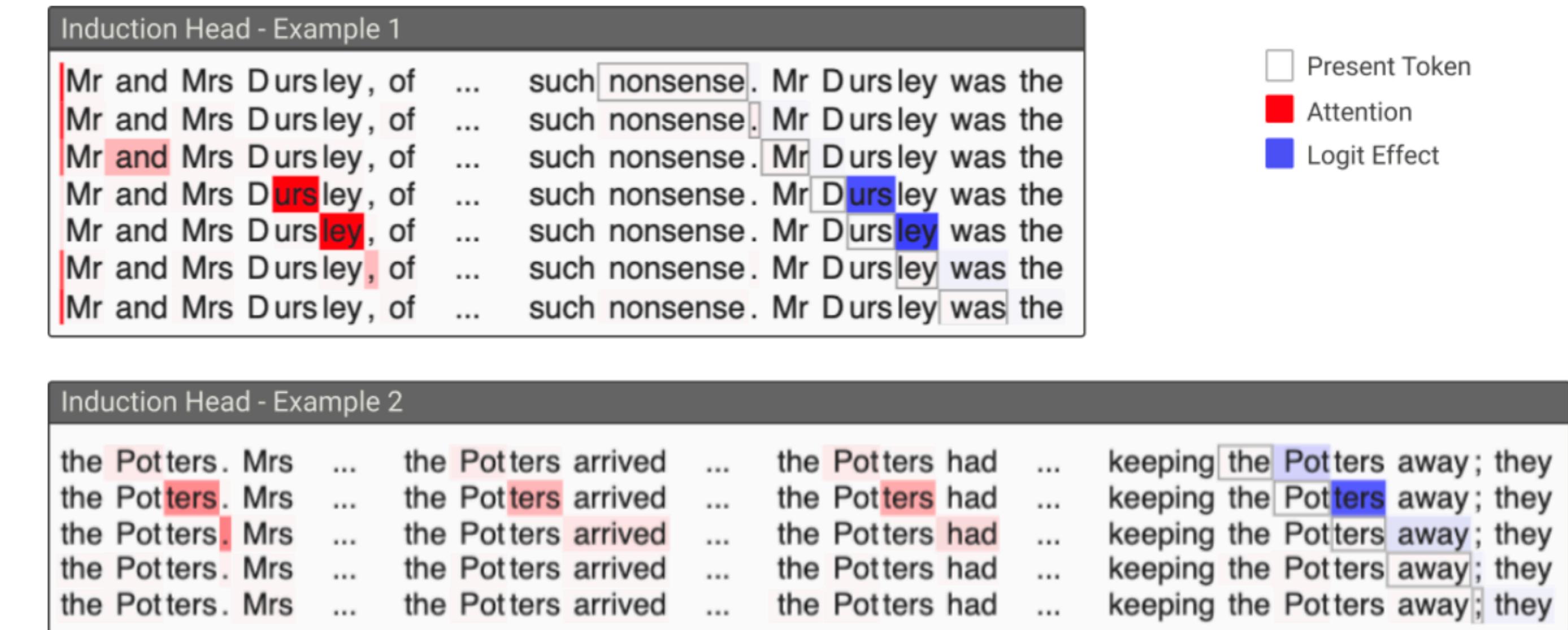
- Tokenizers split up some terms into multiple tokens
- Some English terms are multi white-space-separated to begin with
- Our models have to deal with it
- We can observe the consequences in the realized QK/OV matrices

More examples of large entries QK/OV circuit

Source Token	Destination Token	Out Token	Example Skip Tri-grams
"indy"	"C", "C", "V", "V", "R", "c"	"indy", "obby", "INDY", "loyd"	"indy... Cindy", "indy... CINDY"
"Pike"	"P", "P", "V", "Sp", "V", "R"	"ike", "ikes", "ishing", "owler"	"Pike... Pike", "Pike... Spikes"
"Ralph"	"R", "R", "P", "P", "V", "r"	"alph", "ALPH", "obby", "erald"	"Ralph... Ralph", "Ralph... RALPH"
"Lloyd"	"L", "L", "P", "P", "R", "C"	"loyd", "alph", "\n ", "acman", ... "atherine"	"Lloyd... Lloyd", "Lloyd... Catherine"
"Pixmap"	"P", "Q", "P", "p", "U"	"ixmap", "Canvas", "Embed", "grade"	"Pixmap...Pixmap", "Pixmap...QCanvas"

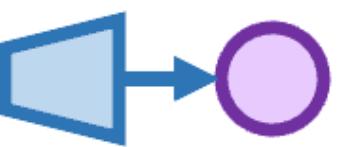
# Head Types: Induction Heads

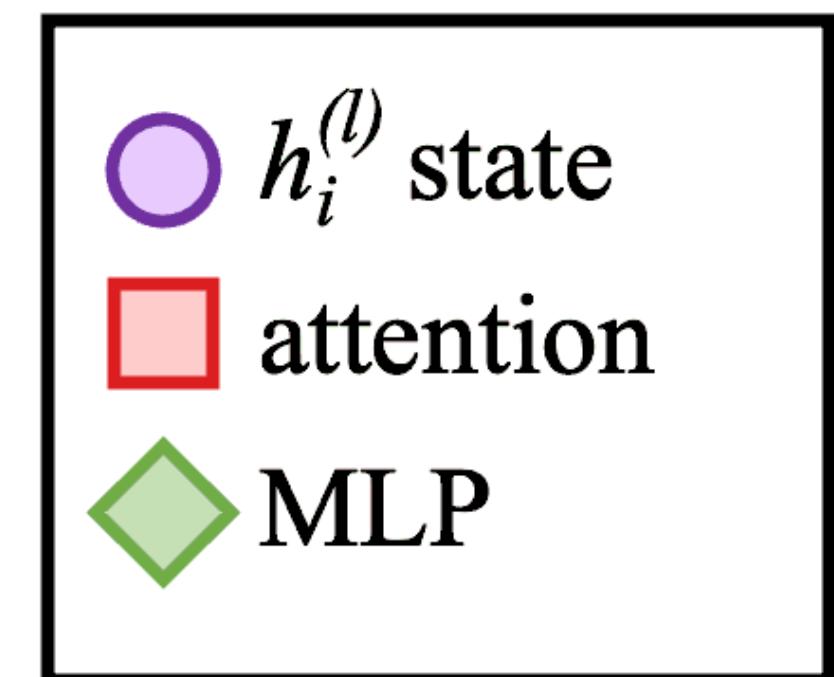
- The QK/OV circuits let us observe the emergence of specialized attention head types
- **Induction Heads** try to complete patterns previously encountered in the context:
  - If the current token has been seen before, attend to the immediately next token
  - Otherwise, dump attention onto the first token in the sequence.



# Activation Patching

# Locating Factual Knowledge

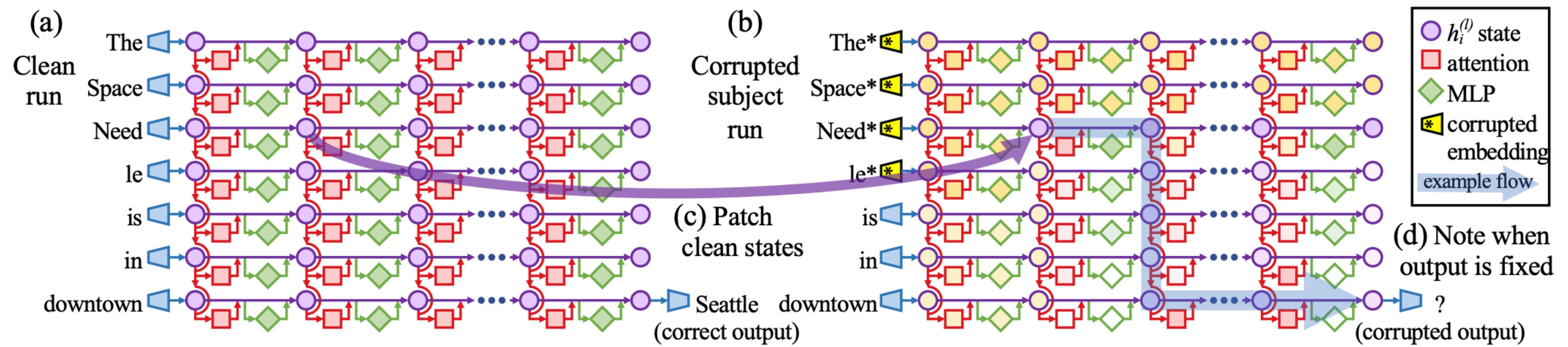
Eiffel 



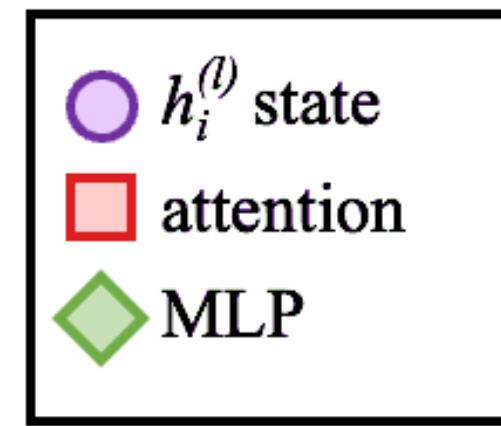
# Activation Patching

The **Space Needle** is in downtown -> **Seattle**.

The **Colosseum** is in downtown -> **Rome**.



# Patch Location

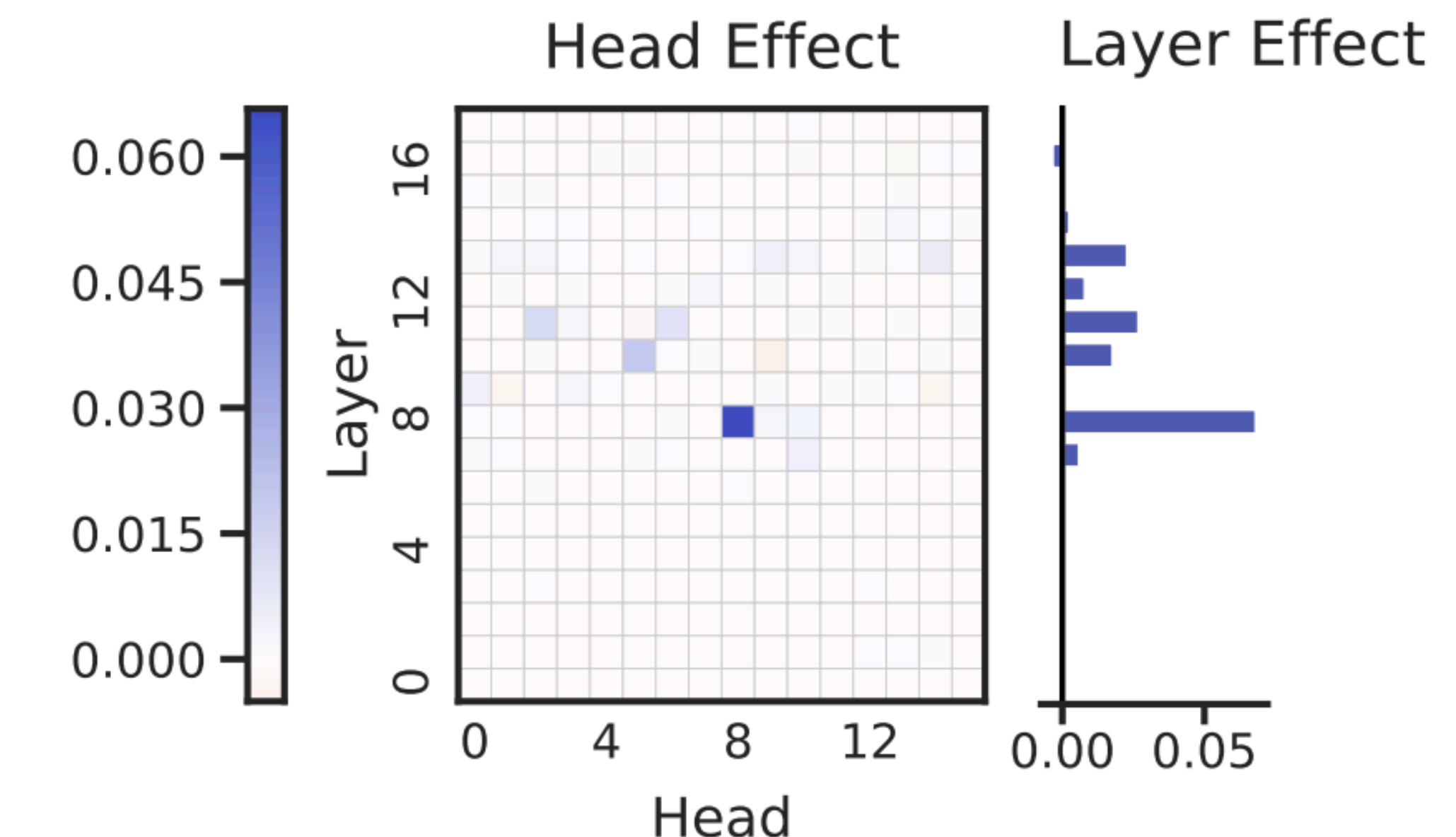


# Patch Effects

- Track patch effect across the layers and components of the model
- Locate components with strong effect when patched
- E.g., gender bias

The **nurse** entered the room -> **She** ...

The **doctor** entered the room-> **He** ...



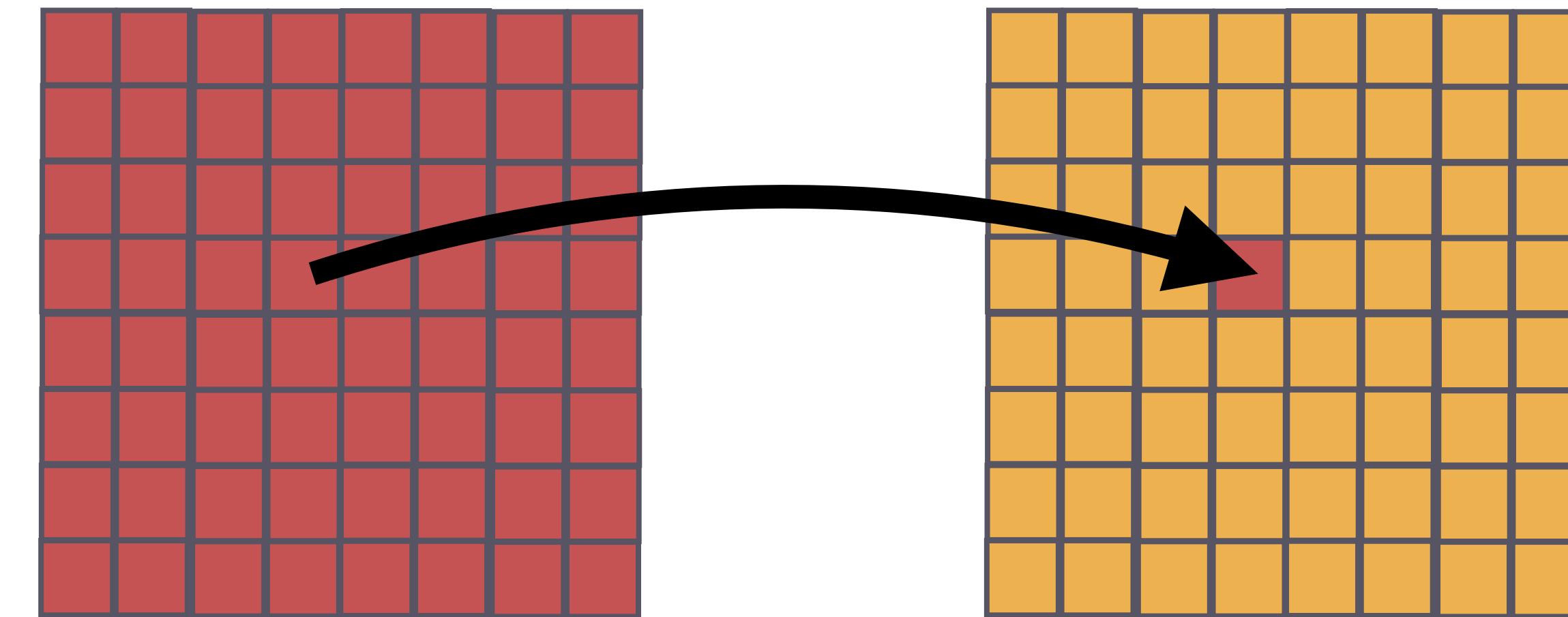
# Finding an Expressive Metric

- **Logit Difference** between correct/incorrect answers
- **Raw Class Logits** for the correct answer
- **Log Probability** of correct answer
- **Raw Probability** of correct answer
- **Binary performance metrics** such as accuracy or rank of correct answer

Desired property:	Continuous (not discrete)	Track model confidence linearly	Measure John vs Mary independent of P(name)	False-positive when “breaking the model”
Logit Difference:	Yes	Yes	Yes	Yes
Logit:	Yes	Yes (typically)	Yes	Maybe
Logprob:	Yes	Not close to p=1	No	No
Probability:	Yes	No	No	No
Binary metrics:	No	No	No	No

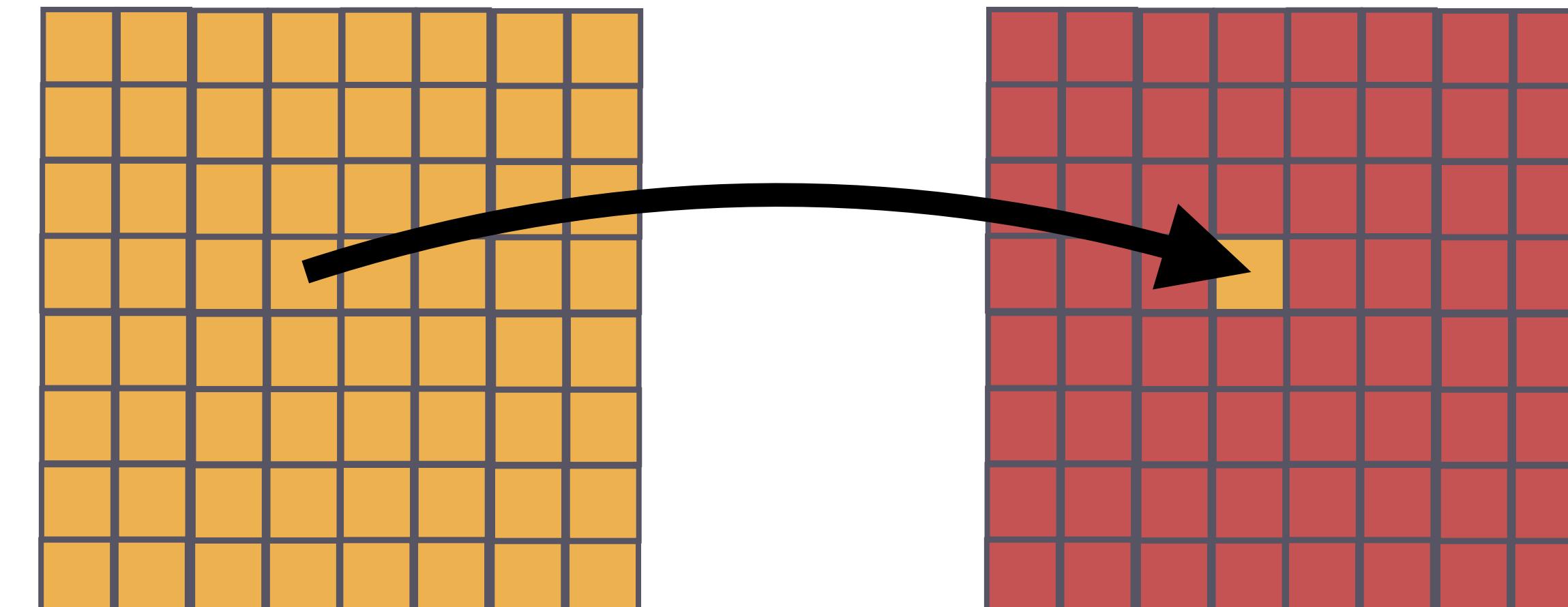
# Patch Direction

- **Corrupted to Clean** patching shows whether activations are necessary to maintain model behavior (noising).



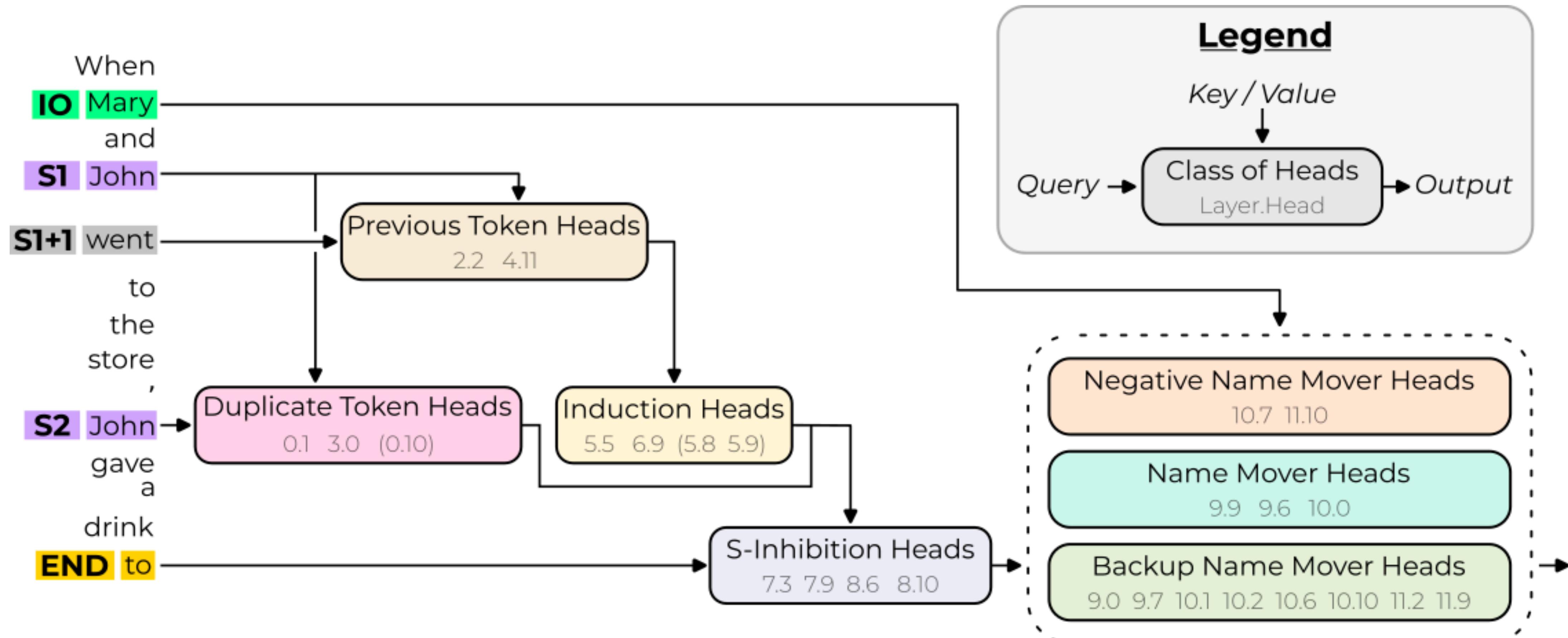
The **nurse** entered the room -> **She** ...  
The **doctor** entered the room-> **He** ...

- **Clean to Corrupted** patching shows whether activations are sufficient to restore model behavior (denoising).



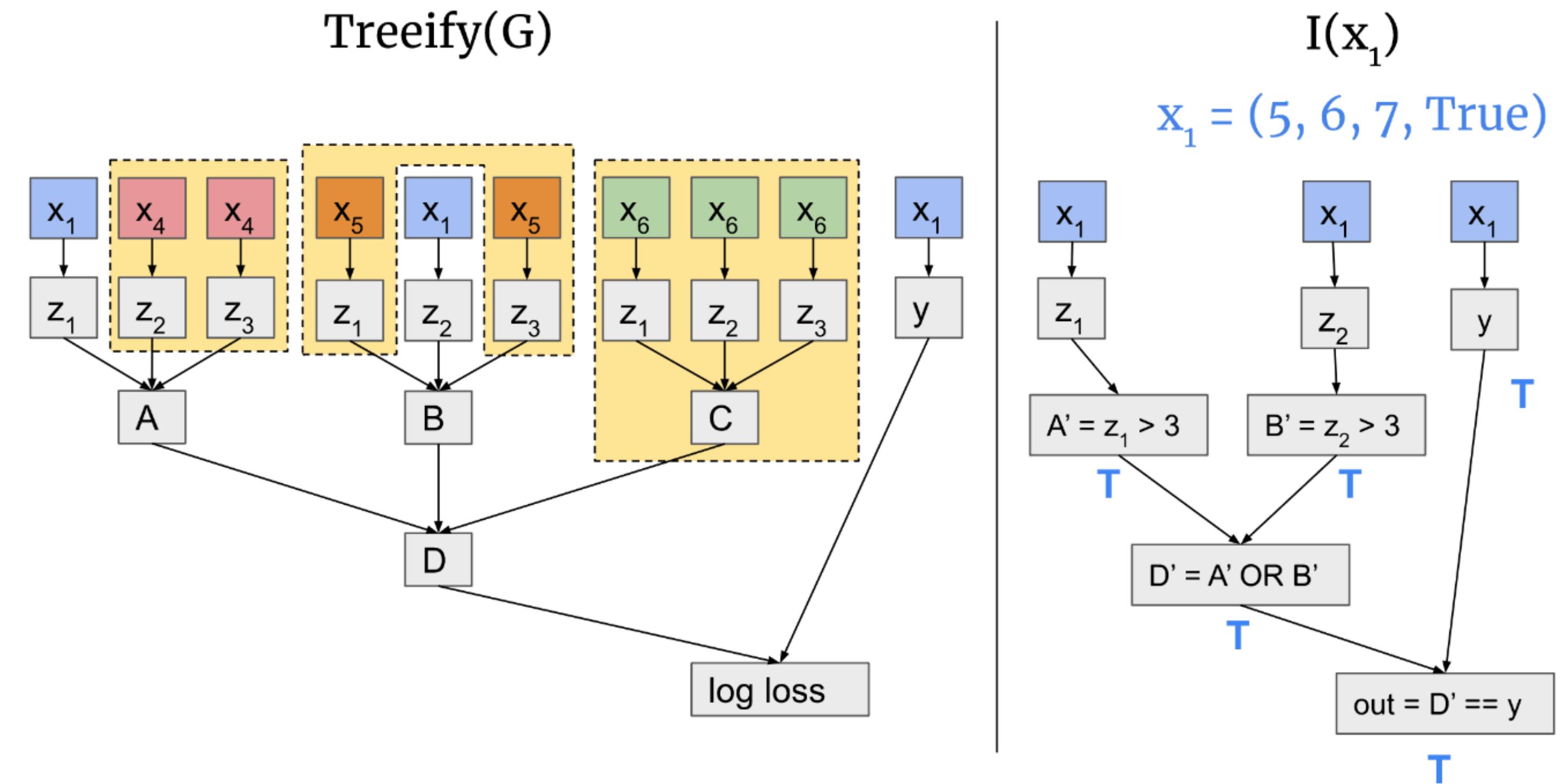
# Circuit Analysis

# More Heads...



# Causal Scrubbing

- Establish hypothesis
    - e.g., via activation patching
  - Systematically test hypothesis by recursively find all invariant model components



# What just happened!?

- Early Decoding
- Residual Stream
- Patching
- Circuit Analysis

## What is to come...

- Tutorial **July 5th** (health permitting)
- Next lecture **July 16th**