Training Dynamics II

Aaron Mueller
CS599 B1: Advanced Natural Language Processing
Nov. 17, 2026

Administrivia

- HW2 grades were released over the weekend.
- You all have been assigned to review 2 papers each. See your assignments on OpenReview.
- I will grade midway reports after all peer reviews are in. Grades will likely be released the weekend after Thanksgiving.

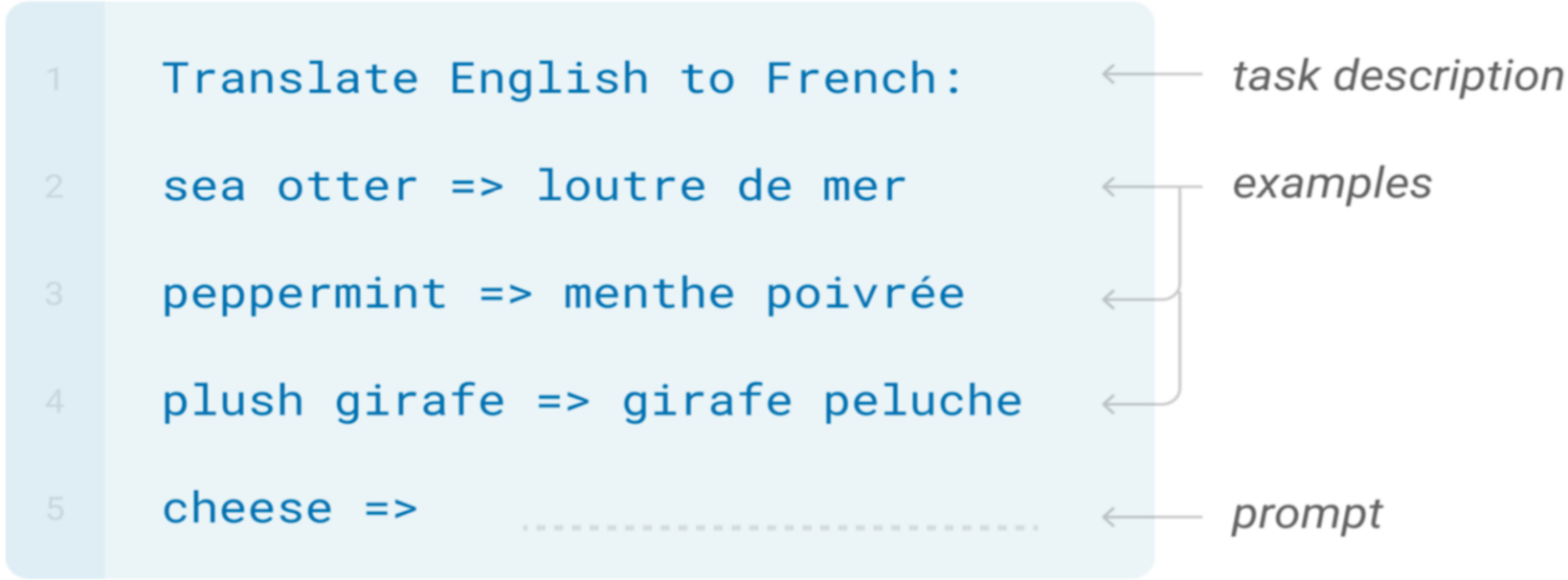
Peer Review Guide

- See the final project instructions doc and the Reviewer Assignments Released post on Piazza for instructions, tips, and examples
- The goal is to help the authors improve their work
- It's ok to say that you don't understand something! The target audience is you all, so if you don't get what's happening, this is a sign that the authors could clarify some things. This will help them improve their final report
- Please do not submit completely AI-generated reviews; this is highly frowned upon in peer review.
- Use the Reviewers console on OpenReview.



Review: In-context Learning (ICL)

- In-context learning (ICL) refers to the ability of models to perform a task after only being shown labeled examples in-context
  - No parameter updates

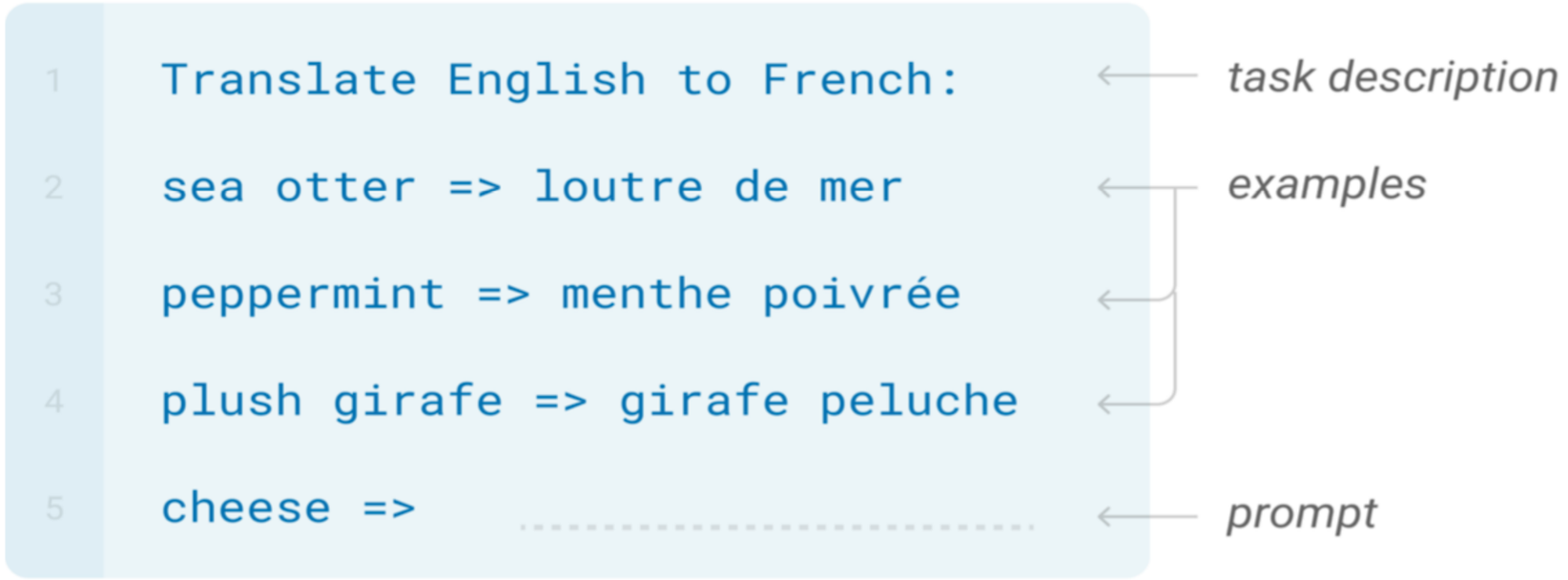


- How does the model learn to do this?

Three Questions About In-context Learning

- **Behavior:** What behaviors does in-context learning encompass?
- **Mechanism:** How is in-context learning implemented in the model?
- **Learning:** Why, when, and how does training produce this strategy?

ICL as Task *Learning*

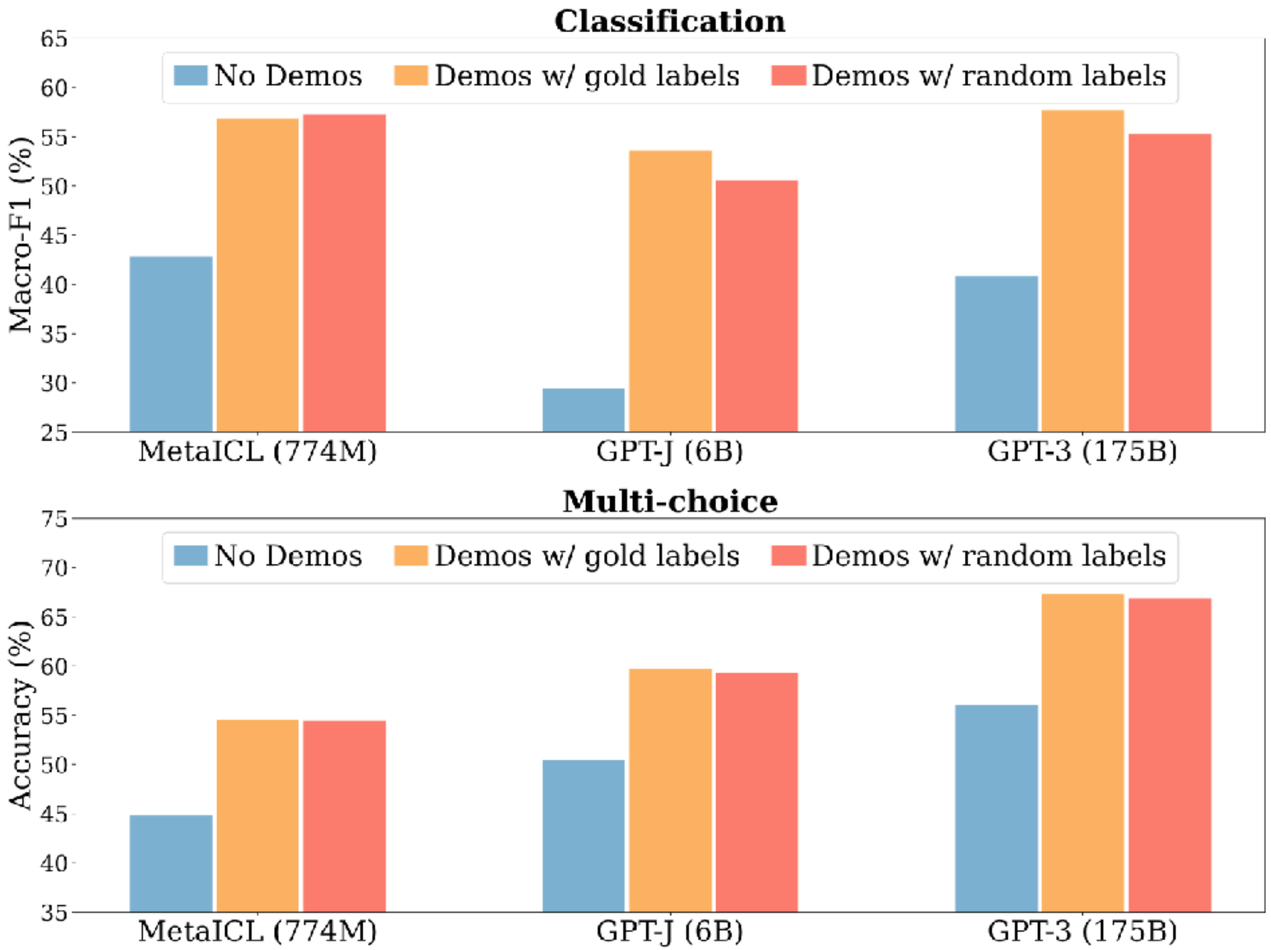


- We usually present examples with correct ground-truth labels in-context, and then have the model predict the label on a new example
- It looks like ICL works because we're showing the model correct labels, and thus supervising it

ICL as Task *Recognition*

- But if we randomly shuffle the labels, we can still get ICL performance that's nearly as good!

- Maybe ICL is just showing the model the kind of input the task contains, and the label space for the task?

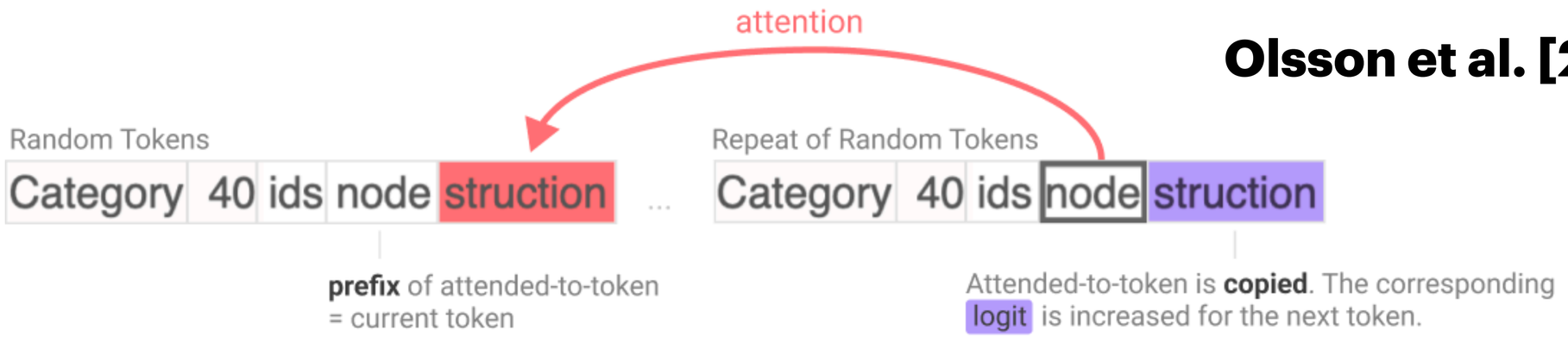


Min et al. [2022]

What is ICL, exactly?

- These results show us that ICL may be less intuitive than we thought.
- What computations underlie this capability?
- How do models learn to use information in-context, instead of just relying on parametric knowledge?

Induction Heads



Olsson et al. [2022]

[A] [B]... [A] -> [B]

- There are attention heads that implement an induction algorithm for predicting the next token
- When we see token A, look for previous instances of token A in-context, and upweight the token that came right after
- Implemented as **prefix matching** (locating previous [A] tokens) and **copying** (increasing the probability of [B])

The Induction Circuit

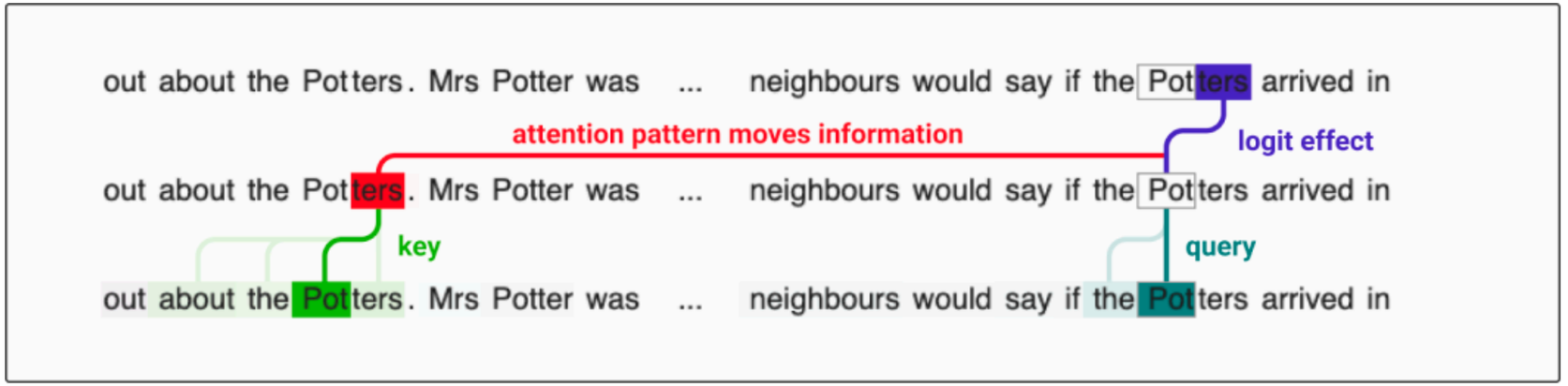
Olsson et al. [2022]

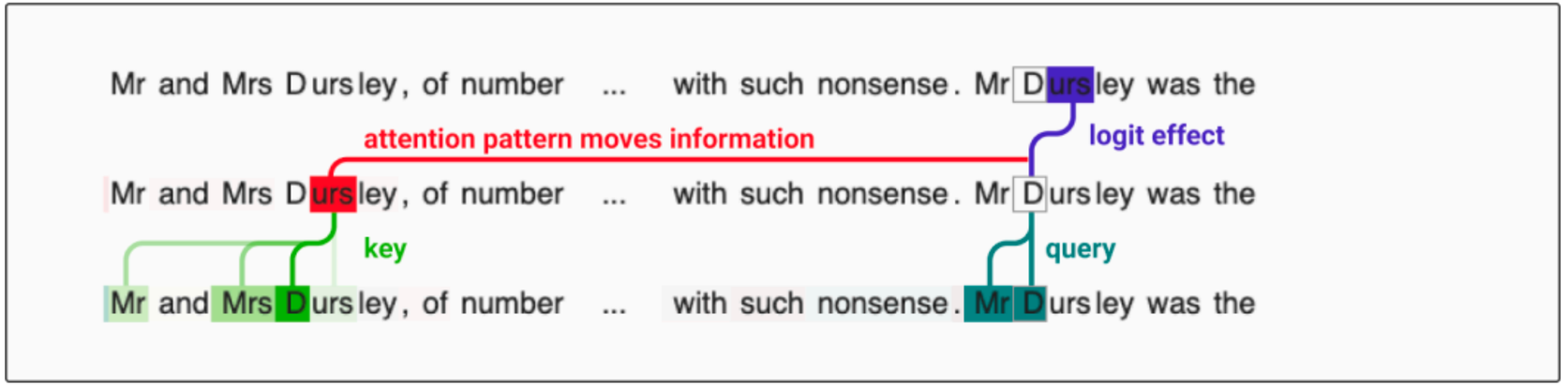
1. Previous-token heads write information about preceding tokens.

2. Induction head matches the current prefix

3. These heads move the earlier continuation to the current position

4. The logit of the copied token increases





Note: this requires two layers to implement.

Measuring In-context Learning and Induction

- We can quantify whether a model has learned ICL via improvements in next-token prediction loss later in the context compared to earlier in the context

$$\text{ICL score} = \mathbb{E}_x[L_{500}(x) - L_{50}(x)]$$

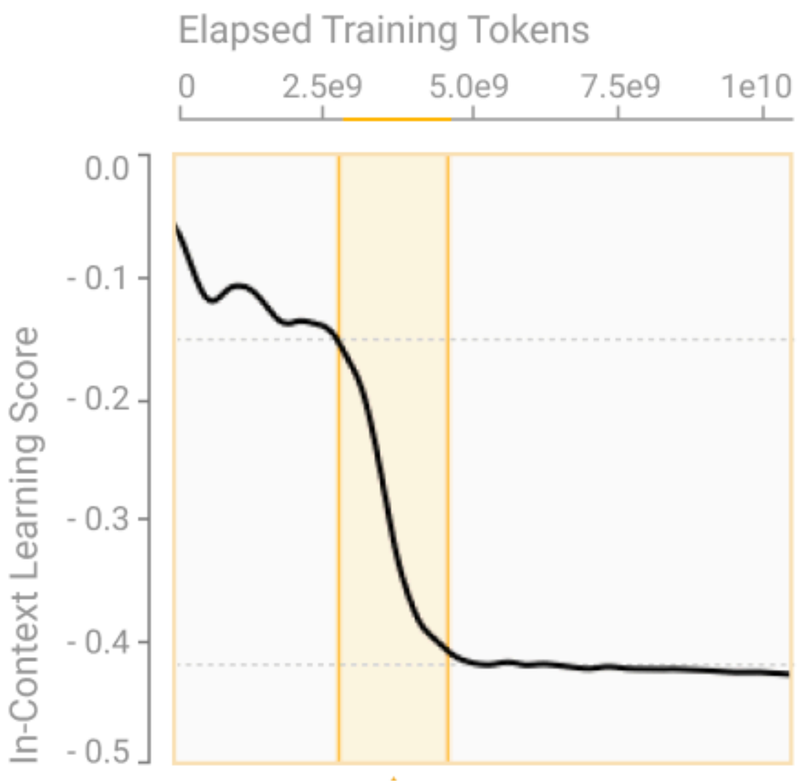
- As a proxy for induction, we can measure the prevalence of prefix matching in the attention as follows:

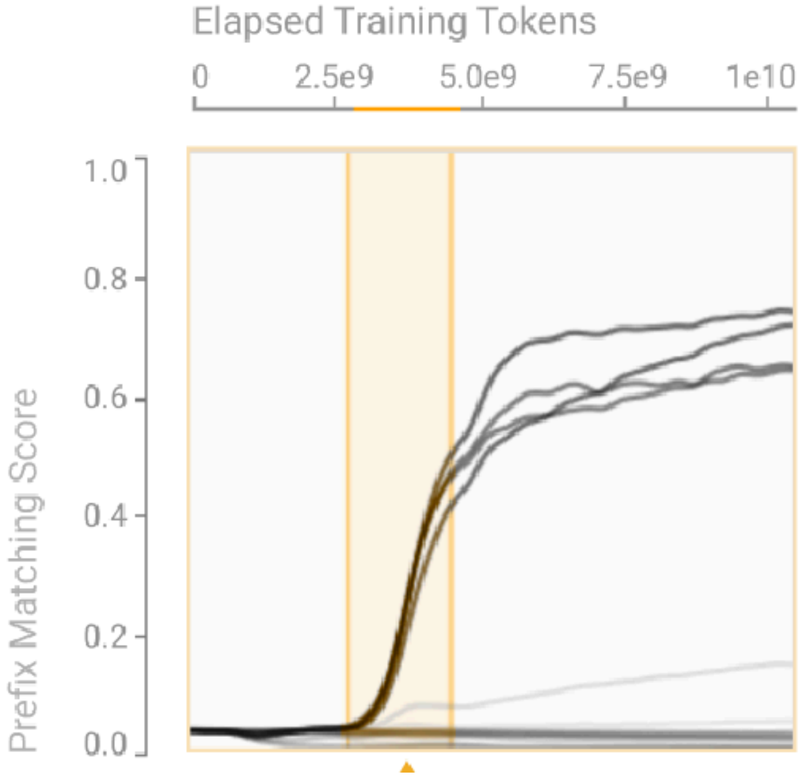
Given input with repeated token sequences $x = (t_1, t_2, \ldots, t_T, t_1, t_2, \ldots, t_T)$,

we define the prefix matching score $PM(h) = \frac{1}{T-1} \sum_{j=1}^{T-1} A^{(h)}_{T+j,\, j+1}$

Induction Heads Arise When ICL Emerges

Olsson et al. [2022]

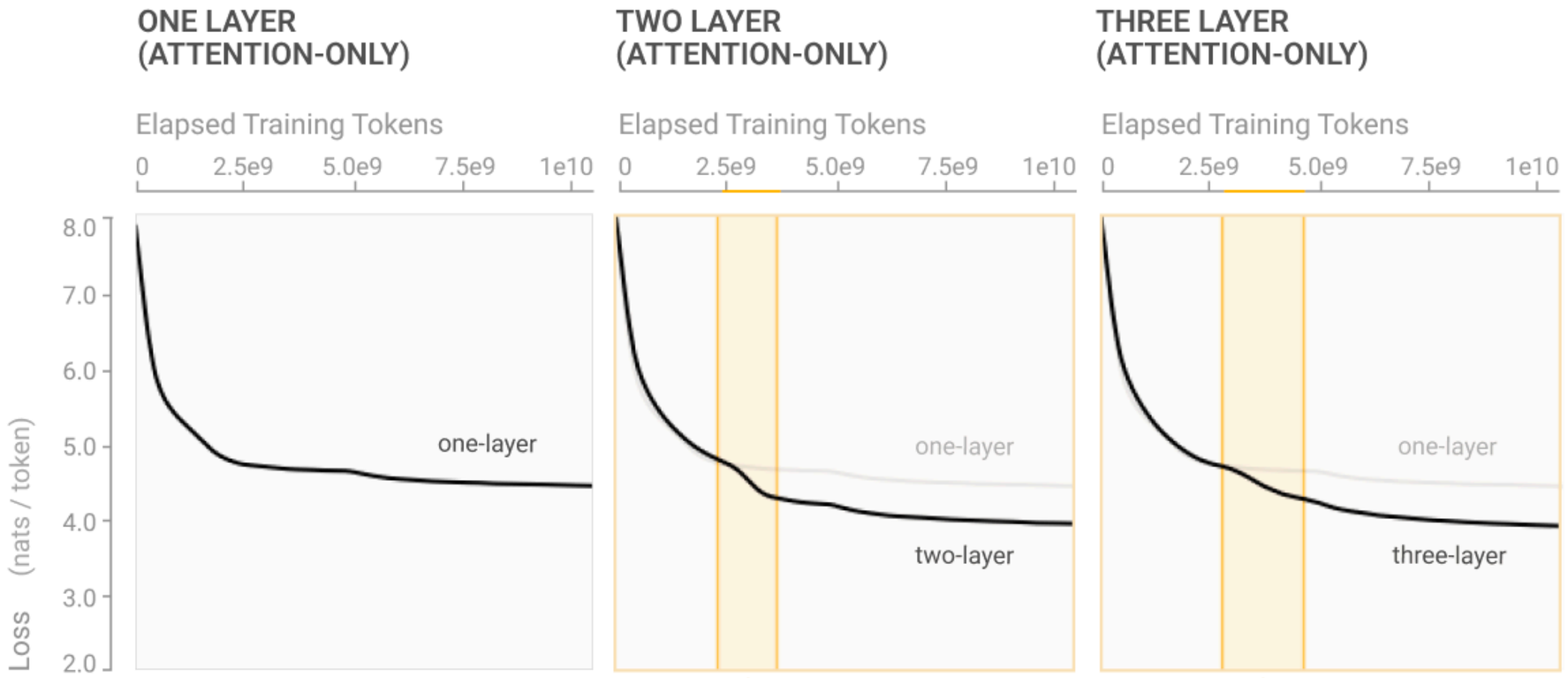




There is a sudden change in in-context learning scores when induction heads emerge.

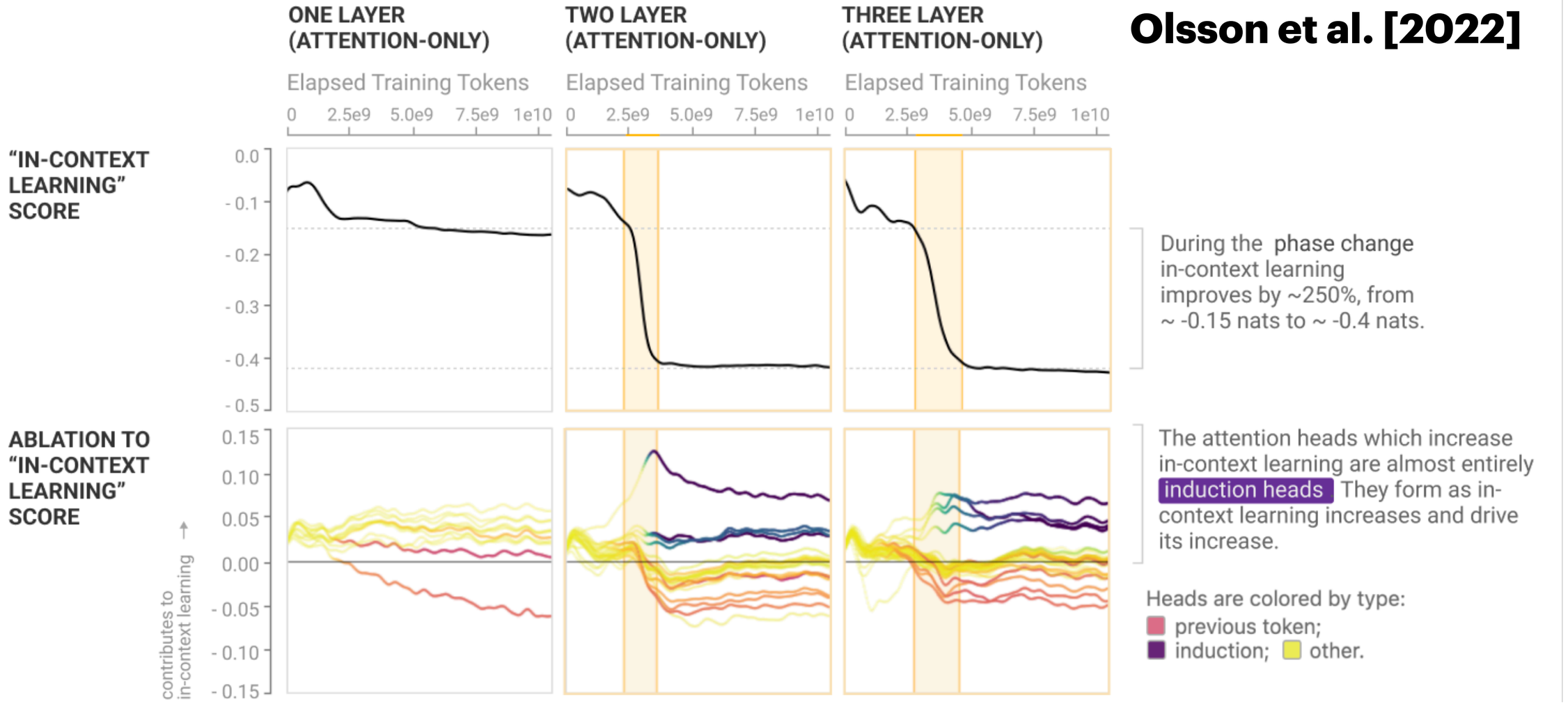
Sudden Drops in Loss When Induction Heads Emerge

Olsson et al. [2022]



There is a sudden phase transition in the loss during this same period.

Ablating Induction Heads Decreases In-context Learning



Olsson et al. [2022]

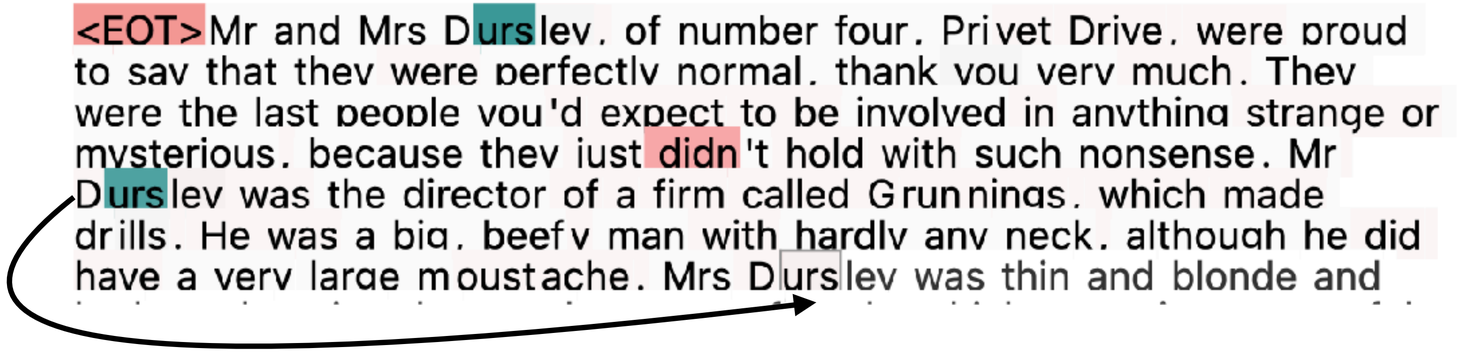
During the phase change in-context learning improves by ~250%, from ~ -0.15 nats to ~ -0.4 nats.

The attention heads which increase in-context learning are almost entirely induction heads They form as in-context learning increases and drive its increase.

Heads are colored by type:
previous token;
induction; other.

Induction Heads Implement Non-trivial Behaviors

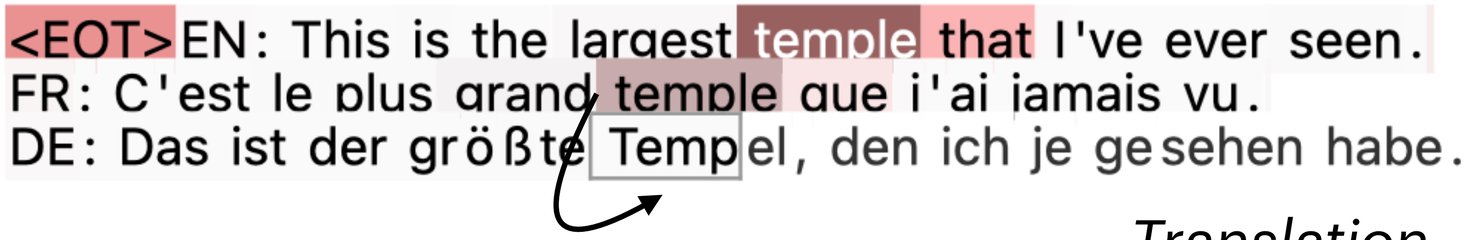
Olsson et al. [2022]



Exact copying



Pattern matching



Translation

That we have these non-trivial induction head behaviors suggests that induction could be a plausible mechanism underlying much of ICL.

From Post Hoc Interventions to Training-time Interventions

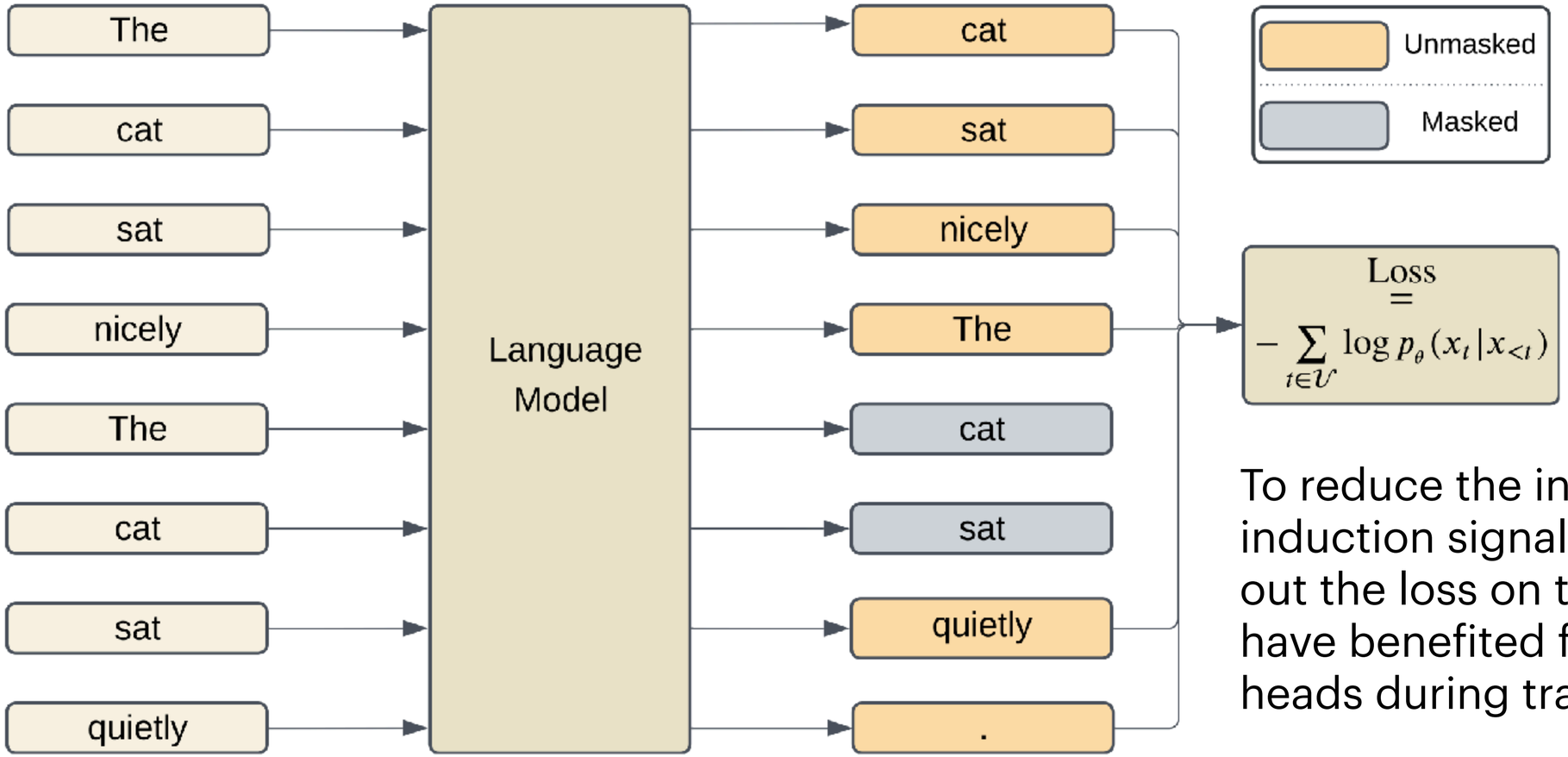
- The previous study ablated induction heads *post hoc* (that is, after the model had been fully trained).
    - What happens if we ablate a mechanism after it was learned?

- Now, let's look at an approach that directly discourages induction heads from arising at all during training
    - What happens to the model if we remove that training signal?

- We'll use an approach called HAPAX to do this.

Discouraging Induction Heads with HAPAX

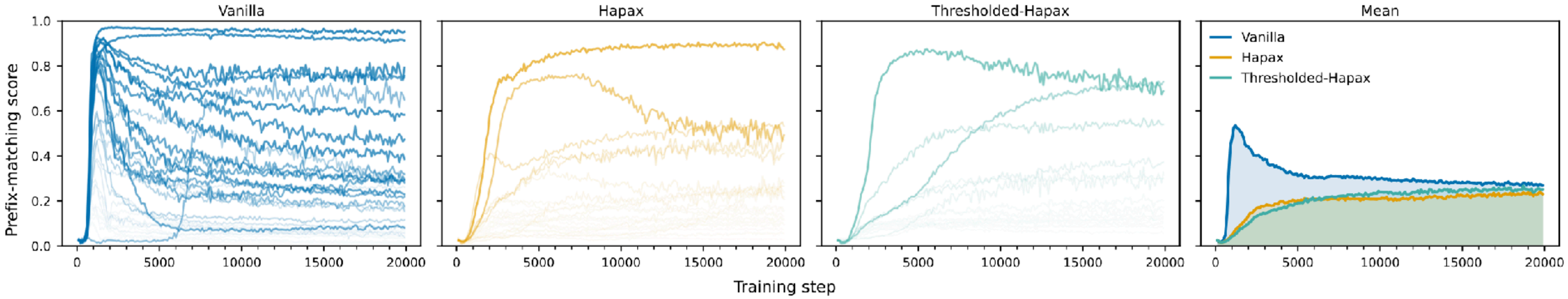
Şahin et al. [2026]



To reduce the influence of induction signals, we can mask out the loss on tokens that would have benefited from induction heads during training.

Sanity Check

Şahin et al. [2026]



- Thresholded HAPAX is like HAPAX, but where we also mask token pairs with high representational similarity to tokens that previously appeared (not just exact matches)
- Prefix-matching scores are generally far lower for HAPAX and Thresholded HAPAX

Extractive vs. Abstractive Tasks

Extractive	**Abstractive**
foil car purple -> *purple*	Greece -> *Athens*
cat tin saucepan rouge ivory -> *saucepan*	China -> *Beijing*
pen cloud window building -> *window*	Canada -> *Ottawa*

In extractive tasks, the answer is extracted from the prior context.

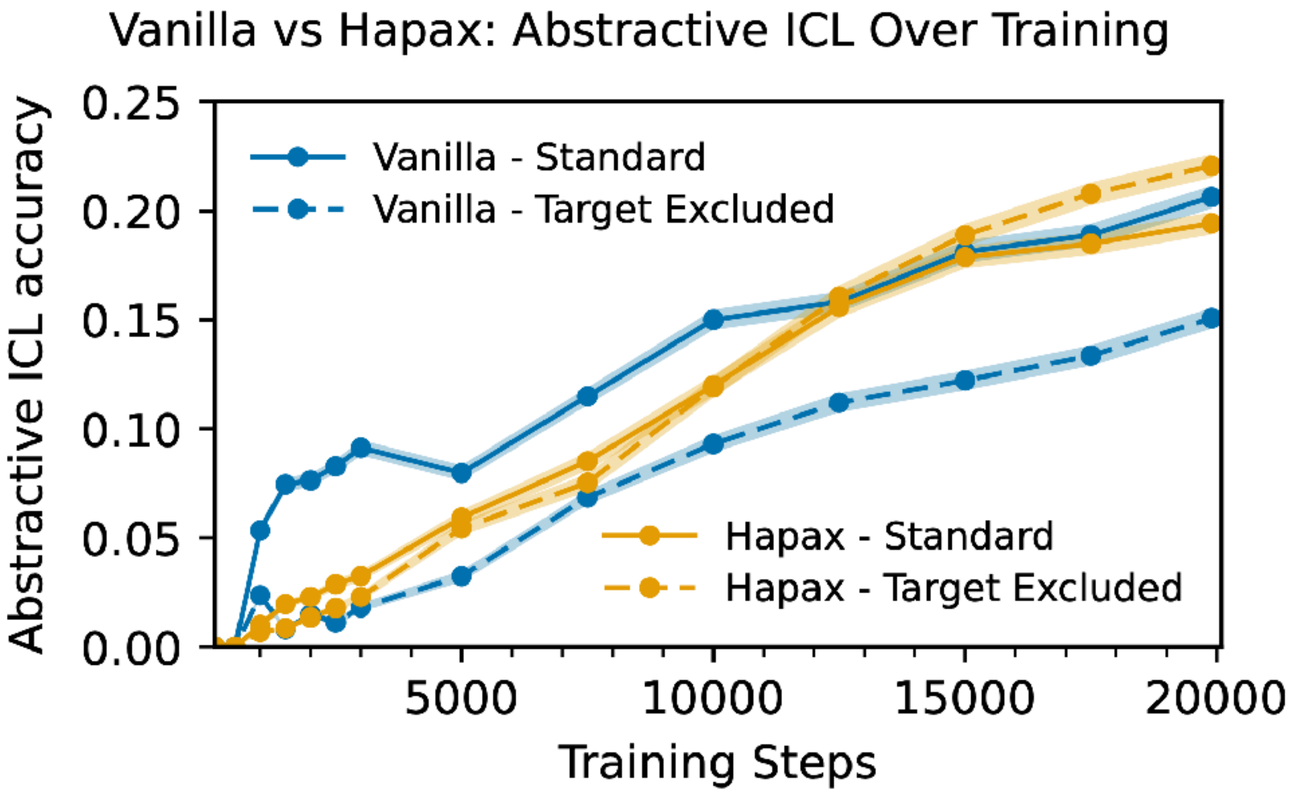
In abstractive tasks, the answer is not contained in the context.

If induction heads are generally important for ICL, we should expect performance on both kinds of tasks to decrease once we remove induction heads.

Task	Vanilla (%)	HAPAX THRESH (%)
Adjective V Verb 3	**58.8 ± 3.1**	56.1 ± 3.1
Alphabetically Last 5	**18.6 ± 2.4**†	13.0 ± 2.1
Animal V Object 5	**37.1 ± 3.0**†	16.3 ± 2.3
Choose First Of 5	**95.6 ± 1.3**†	27.0 ± 2.8
Alphabetically First 5	**19.9 ± 2.5**†	14.5 ± 2.2

Removing induction heads reduces scores on extractive tasks.

Şahin et al. [2026]



But removing induction heads does *not* greatly reduce scores on abstractive tasks.

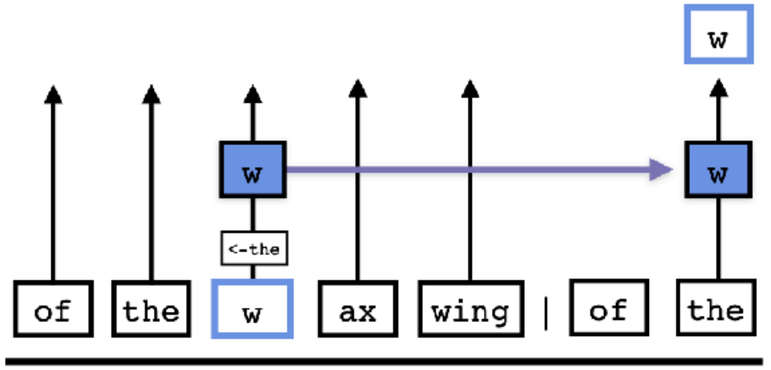
Does this mean that induction heads are only part of the story? What else could be playing a role?

Induction and ICL

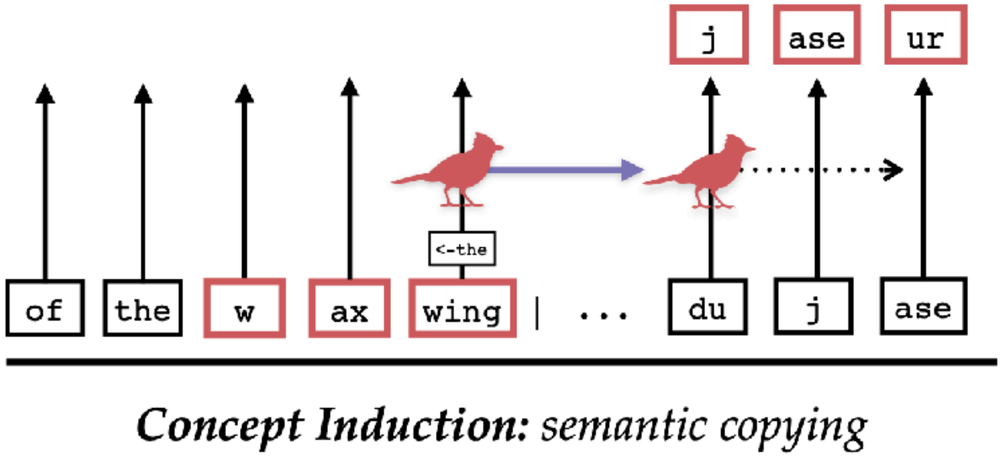
- Induction appears to be an important component of in-context learning, but not a strictly necessary mechanism for all ICL.

- Inductive copying is not essential for learning abstractive ICL. What else could explain those capabilities?
  - Function vectors [Todd et al., 2024]: some heads directly encode the task itself as a vector
  - Concept induction [Feucht et al., 2025]: rather than copying specific tokens, some heads might perform induction at a higher conceptual level

Feucht et al. [2025]



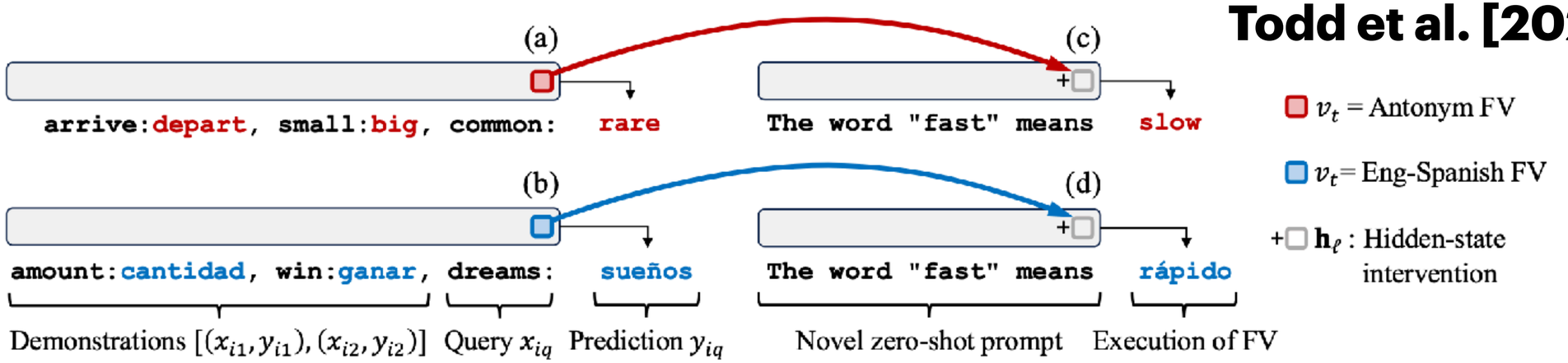
Token Induction: *verbatim copying*



Concept Induction: *semantic copying*

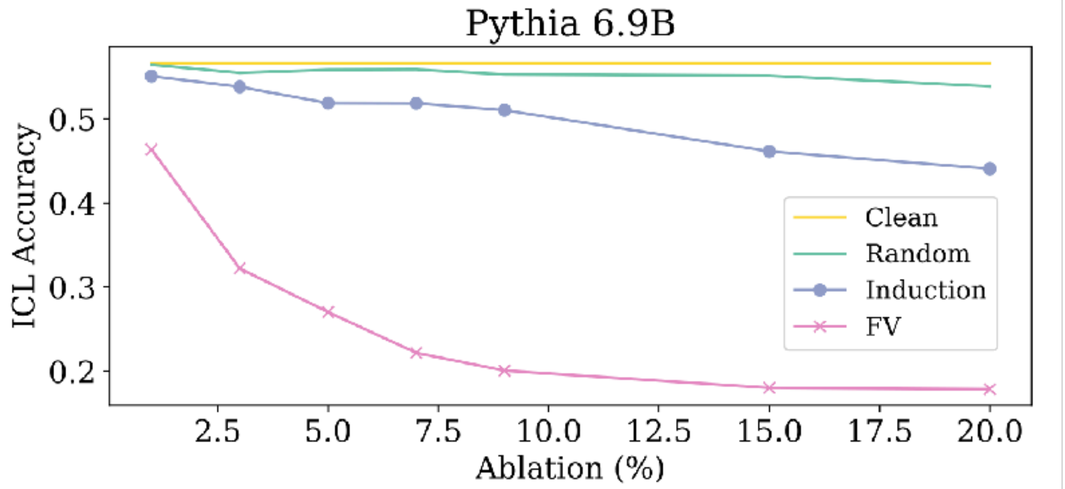
Function Vector Heads

Todd et al. [2024]

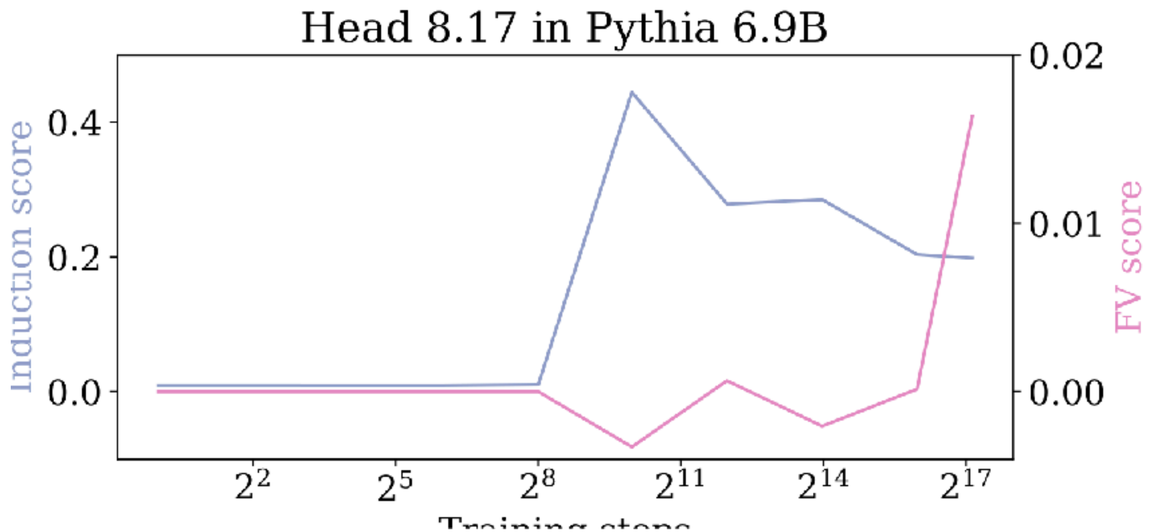


- **Function vectors (FVs)** are vector representations of ICL tasks that can be extracted from specific attention heads and added into the LM's representations to cause it to perform some task
- The heads that encode and transport these function vectors can be denoted as FV heads
- To find FV heads, we can (i) compute mean activation of attention head a over prompts where the task is being performed, (ii) create corrupted ICL prompts by shuffling output labels while maintaining the same inputs, and (iii) measure each head's causal contribution to recovering the correct output (despite the shuffled labels) when we patch the mean activation from the original task setting back in

Maybe Function Vector Heads Matter More?



Ablating function vector heads has a greater effect on ICL than ablating induction heads.



But function vector heads arise later than induction heads.

FV heads often exhibit induction-like behavior earlier in training. Induction may act as a stepping stone to FV mechanisms (but this has not been established definitively).

Yin & Steinhardt [2025]

Other Recent Results

- **Chan et al. [2022]**: ICL emerges when data is “bursty” (i.e., has items that recur in a context in a clustered format)

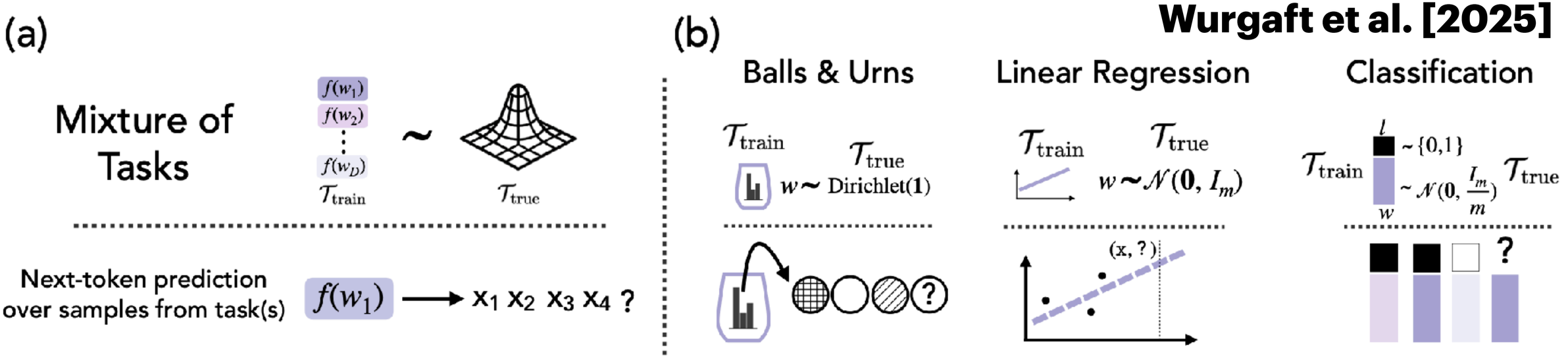
- **Raventós et al. [2023]**: If the data is not sufficiently diverse, models fit the training tasks directly. When data is diverse, models behave more like general-purpose task solvers

- **Singh et al. [2023]**: ICL can be *transient*, appearing and later fading as training continues

- Can we explain all of these findings with a single theory?

A Toy Model of Learning In-context Learning

Wurgaft et al. [2025]



- Draw a finite set of D tasks from the true underlying task distribution T_{true}; the sampled distribution becomes T_{train}.
  - We will refer to D as the **task diversity**
- Repeatedly sample sequences from those tasks
- Train the Transformer via next-token prediction

Which strategy *should* the model learn?

- Memorization strategy **M**:
  - Assume the current task is a task we've encountered during training
  - Infer which *known/seen* task generated the examples
  - Achieves good training performance, but does not naturally extrapolate to new tasks
- Generalization strategy **G**:
  - Infer the latent task/rule from examples
  - Can apply the inferred rule to tasks outside the finite training set
  - More broadly reusable, but may sacrifice some fit to the training data

- As training proceeds, what determines whether the Transformer behaves more like **M** or **G**?

Rational Models of Learning

- We sometimes use **Bayesian models** to obtain an estimate of what a rational reasoner would do for a given dataset.
- Recall Bayes' rule:

Prior over hypotheses

$$p(H \mid D) \propto p(H)p(D \mid H)$$

Posterior over hypotheses given the data

Likelihood of data given hypotheses

- This holds that the probability of a hypothesis being learned should be proportional to the prior probability of that hypothesis and the likelihood of the data under that hypothesis
- Our prior contains a simplicity bias

Measuring Complexity

- How do we infer the complexity of the memorizer vs. generalizer?

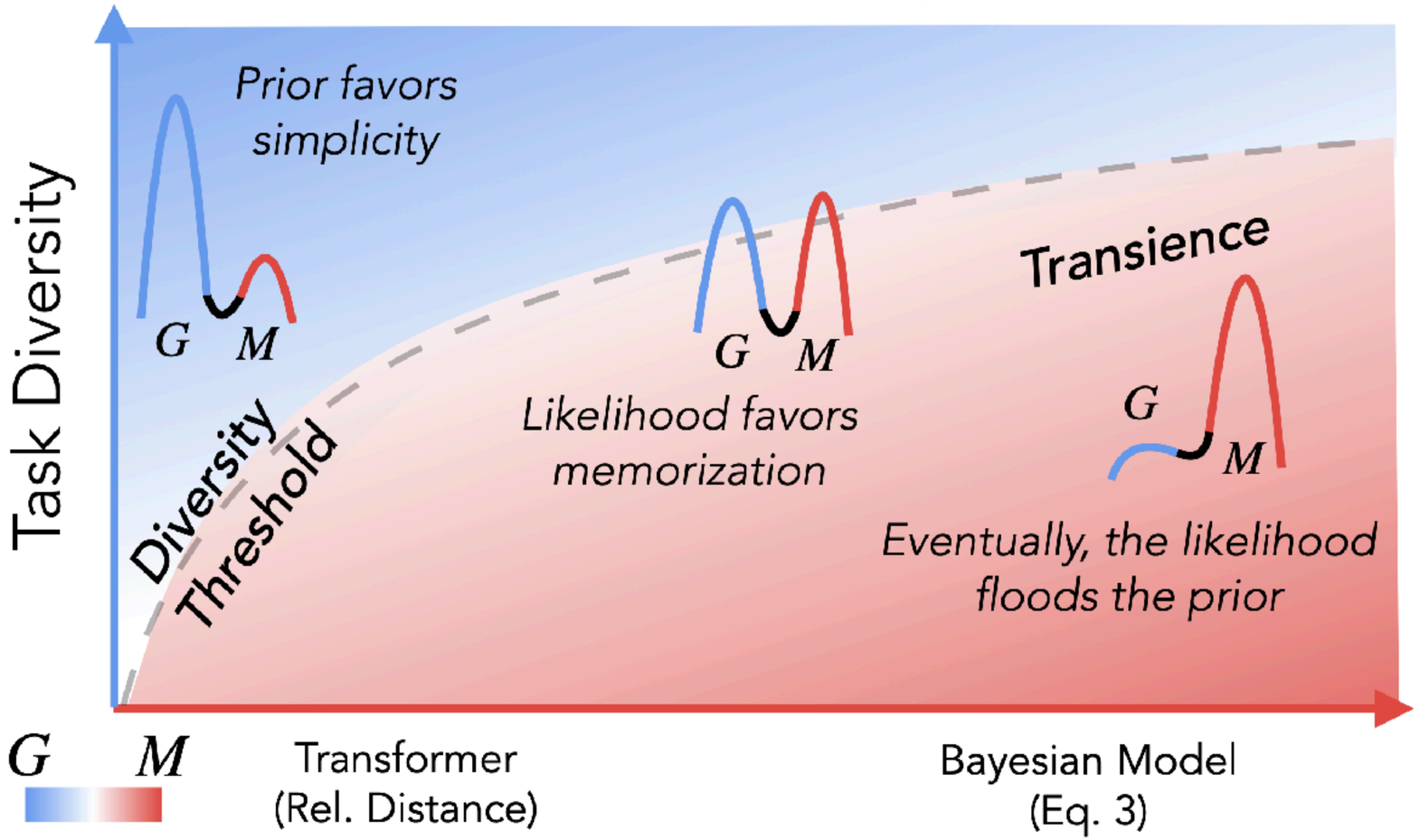
$$p(H) \propto 2^{-K(H)}$$

- Here, K is Kolmogorov complexity
- Computing K in exact form is intractable, but we can get an upper bound by writing a classifier for each hypothesis, and using gzip on the implementation of each predictor (code + data)! Take the size of the resulting file as its complexity
- $K(M)$ grows with task diversity, but $K(G)$ is roughly constant

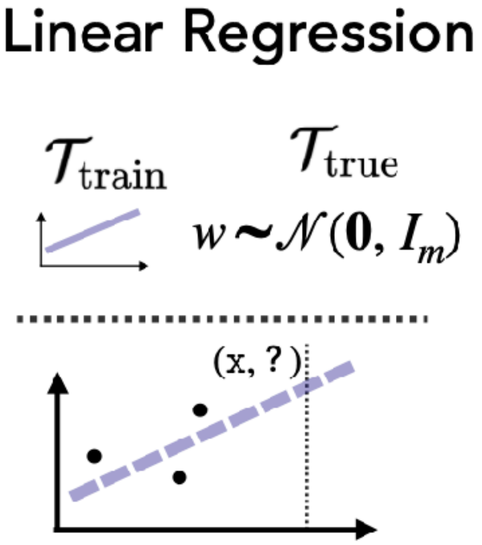
Hypothesis Space

- Early in training, the prior favors the simpler generalizing predictor **G**
- But the memorizing predictor **M** achieves a lower loss on the training distribution. As evidence accumulates, the likelihood can overwhelm the prior toward simple solutions.

Wurgaft et al. [2025]



Strategic Selection



- Suppose T_{train} only contains regression examples with slopes 1, 2, 3
- Given examples generated with a slope of 5, the memorizing predictor asks “which previously seen slope generated these examples?” It will tend toward 1, 2, or 3.
- The generalizing predictor infers the latent arbitrary slope and can thus infer the slope directly

Preference for strategy ≈ fit to training data – complexity cost

Bayesian Predictor Functions

- The memorizing predictor implements a posterior of the form:

$$M(s_i \mid s_{1:i-1}) = \sum_{w \sim T_{train}} p(w \mid s_{1:i-1}) f_w(s_i \mid s_{1:i-1}) = \mathbb{E}_{w \sim T_{train}}[f_w(s_i \mid s_{1:i-1})]$$

- The generalizing predictor implements a posterior of the form:

$$G(s_i \mid s_{1:i-1}) = \int_{w \sim T_{true}} p(w \mid s_{1:i-1}) f_w(s_i \mid s_{1:i-1}) = \mathbb{E}_{w \sim T_{true}}[f_w(s_i \mid s_{1:i-1})]$$

Predictions w.r.t. Training Steps and Diversity

- Now let's bring the task diversity D and number of training examples N into our predictors. Define $Q \in \{M, G\}$

$$p(Q \mid N, D) \;\propto\; p(Q) \;\cdot\; \exp\Big(-N^{1-\alpha} L_D(Q)\Big)$$

$$h_{N,D}(s_i \mid s_{<i}) = p(M \mid N, D)\, M_D(s_i \mid s_{<i}) + p(G \mid N, D)\, G(s_i \mid s_{<i})$$

Log-odds of M over G

$$\log \frac{p(M \mid N, D)}{p(G \mid N, D)} \;=\; N^{1-\alpha}\, \Delta L(D) \;-\; \Delta K(D)$$

$$\Delta L(D) = L_D(G) - L_D(M) \geq 0, \qquad \Delta K(D) = K(M_D) - K(G)$$

M's advantage in fit

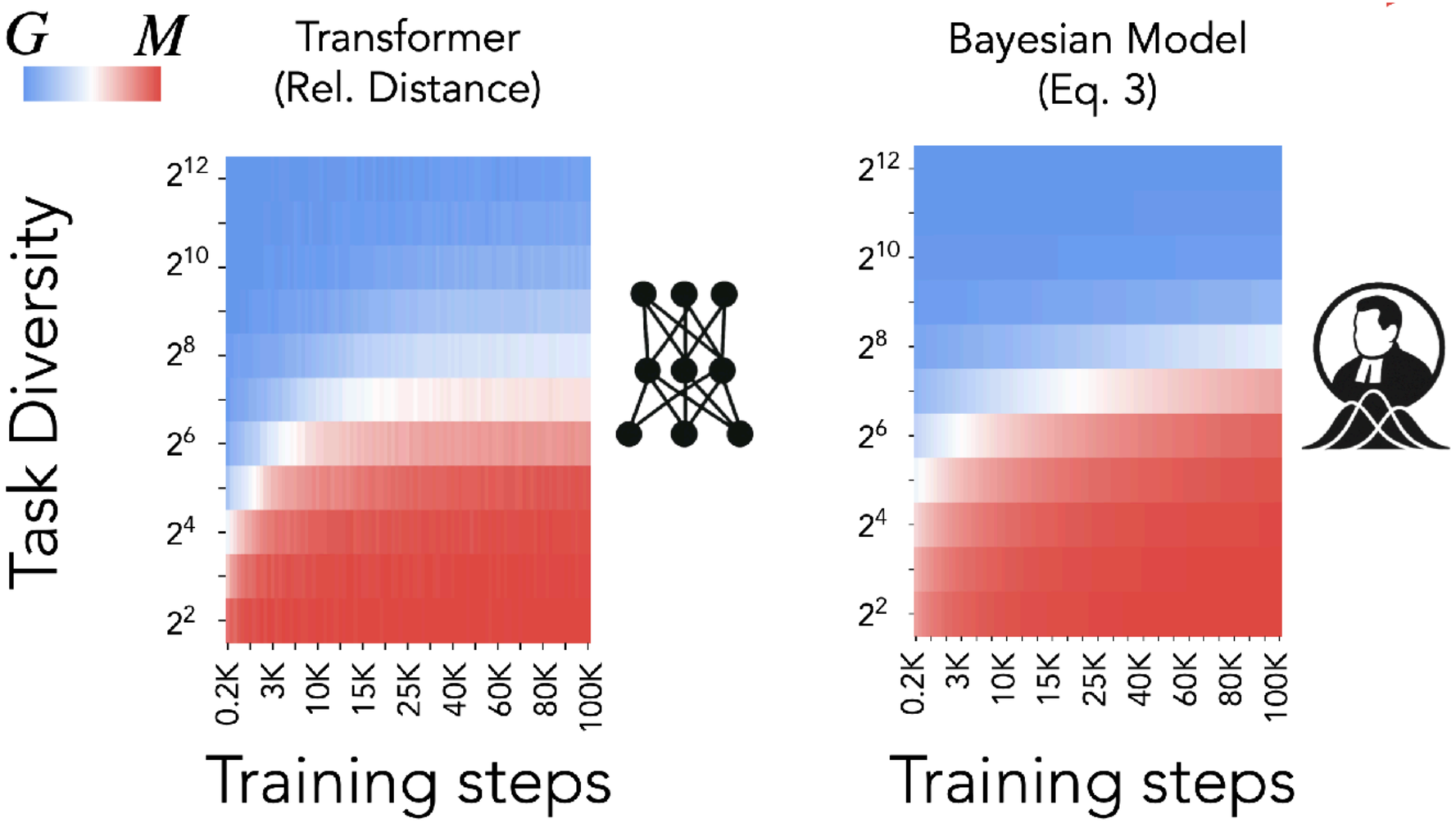
M's additional complexity

Wurgaft et al. [2025]

At high D, G is the far simpler solution.

At intermediate D, G is a better explanation for the data when we see few examples, but the likelihood eventually overwhelms the prior.

At low D, M is simpler and fits better.



The Bayesian model provides a near-perfect fit to Transformer behaviors. This suggests that the model learns to reason in-context mainly when it is *rational* (given the seen data) to do so!

Summary

- **Behavior:** in-context learning lets a model learn to perform a task from examples alone without weight updates
  - Correct labels not always required

- **Mechanism:** induction heads seem to help with extractive tasks, while function vector heads encode the task itself (which may matter more for abstractive tasks)

- **Learning:** whether we see ICL depends on whether the data incentivize learning it. Sometimes, memorizing is the more rational approach (probabilistically speaking)