What needs to go right for an induction head? A mechanistic study of in-context learning circuits and their formation

Aaditya K. Singh [1] **Ted Moskovitz** [1] **Felix Hill** [2] **Stephanie C. Y. Chan** [*2] **Andrew M. Saxe** [*1]

Abstract

In-context learning is a powerful emergent ability in transformer models. Prior work in mechanistic interpretability has identified a circuit element that may be critical for in-context learning – the induction head (IH), which performs a match-and-copy operation. During training of large transformers on natural language data, IHs emerge around the same time as a notable phase change in the loss. Despite the robust evidence for IHs and this interesting coincidence with the phase change, relatively little is known about the diversity and emergence dynamics of IHs. Why is there more than one IH, and how are they dependent on each other? Why do IHs appear all of a sudden, and what are the subcircuits that enable them to emerge? We answer these questions by studying IH emergence dynamics in a controlled setting by training on synthetic data. In doing so, we develop and share a novel optogenetics-inspired causal framework for modifying activations throughout training. Using this framework, we delineate the diverse and additive nature of IHs. By *clamping* subsets of activations throughout training, we then identify three underlying subcircuits that interact to drive IH formation, yielding the phase change. Furthermore, these subcircuits shed light on data-dependent properties of formation, such as phase change timing, already showing the promise of this more in-depth understanding of subcircuits that need to "go right" for an induction head.

1. Introduction

Large language models (LLMs) trained on internet scale corpora (Brown et al., 2020) showcase a remarkable ability to perform *in-context learning* (ICL), adapting to new inputs and tasks at test time. Given the increasing prevalence of LLMs, safety researchers have sought to understand the mechanisms underlying important phenomena like ICL. One approach in the service of this goal is *mechanistic interpretability*: reverse engineering the computations performed by a model. Olsson et al. (2022) identified the induction circuit, which may be responsible for many of transformers' ICL abilities.

An induction circuit is a two-layer circuit (Figure 1a) comprised of a "previous token head" in an earlier layer, followed by the eponymous "induction head". Previous token heads are attention heads responsible for attending to the previous token and copying it into the attending token's residual stream.[1] Induction heads then perform a match-and-copy operation, looking for a match between a query derived from the current token and key derived from the output of the previous token head. Critically, induction circuits typically emerge in a sharp phase change in the loss (e.g., Figure 3), which is often believed to be due to the interaction between two heads in different layers (Nanda, 2022). Presumably, neither the previous token nor the induction head are useful on their own for minimizing the loss.

Recent empirical work on ICL in transformers has introduced more nuances (Min et al., 2022; Wei et al., 2023b), with Singh et al. (2023) even finding that ICL can be a transient phenomenon that disappears with overtraining. These results establish a need to more thoroughly understand induction circuits, especially the *dynamics* of formation during training. Reddy (2023) begins to address this question using *progress measures*, which track correlational relationships between intermediate activations during training. We instead use a causal approach, termed *clamping*, which allows us to directly determine the circuits and dynamics that are causally affecting formation. Through these experiments, we bring to light a new set of mechanisms governing learning dynamics.

[*]Equal contribution [1]Gatsby Computational Neuroscience Unit, University College London [2]Google DeepMind. Correspondence to: Aaditya K. Singh <aaditya.singh.21@ucl.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]The residual stream was a term introduced by Elhage et al. (2021) and refers to the embeddings after each layer that further layers "read" input from. Layers can be viewed as reading from and writing back into this stream. The term "stream" is meant to emphasize the residual skip connections.

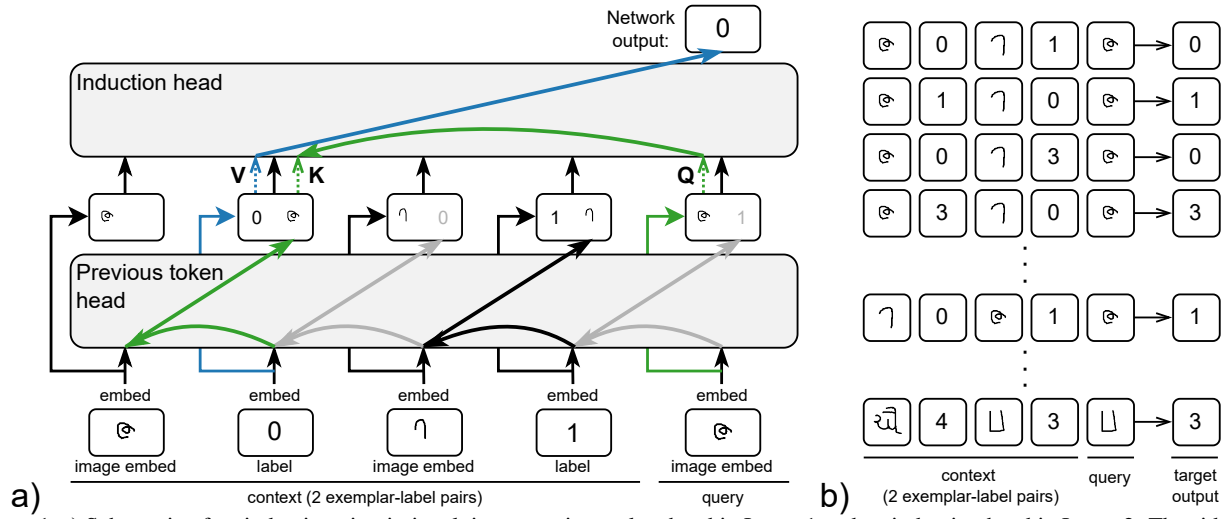

Figure 1. **a)** Schematic of an induction circuit, involving a previous token head in Layer 1 and an induction head in Layer 2. The side-by-side labels and exemplars in the residual stream after Layer 1 are meant to indicate that information about both is superimposed (perhaps in different subspaces). We highlight[3] the "matching" (green) and "copying" (blue) operations that span the two layers. Historically, focus has been devoted to the "match" operation. One of our key results is to demonstrate the important interactions from the "copy" operation. **b)** Example training sequences built from the Omniglot dataset and inspired by classical few-shot meta-training. The context consists of two exemplar-label pairs, where the exemplars are from different classes. The query exemplar comes from the same class as one of the exemplars in context. The in-context labels are randomly chosen. Every exemplar can appear with every possible label in every possible position, forcing the transformer to use ICL to minimize the training loss. Validation sequences either use held out class exemplars or held out pairs of labels.

In this work, we take inspiration from the field of optogenetics in neuroscience, which allows precise causal manipulations of neural activity (Boyden et al., 2005). Here, we develop a novel framework for causally modifying activations throughout model training. These causal-through-training clamping analyses extend beyond prior work, allowing us to easily study interactions between subcircuits and isolate the underlying factors that drive induction circuit formation. We utilize this framework to analyze induction circuit formation in transformers trained to solve a simple few-shot learning (FSL) task inspired by prior work (Chan et al., 2022; Lake et al., 2015). As with any work towards a mechanistic understanding of dynamics, we first provide a thorough investigation of the induction circuits in a network trained on this task, shedding light on the dependence (and independence) of induction heads upon each other (Section 3). We find that multiple induction heads form, contributing additively to minimize the loss. Furthermore, we observe emergent redundancy, despite not applying regularization techniques such as dropout (Hinton et al., 2012), mirroring findings on larger-scale language models (Michel et al., 2019; Voita et al., 2019). We also find the wiring between induction heads and previous token heads to be many-to-many, rather than one-to-one. Next, we focus on the dynamics of formation (Section 4), where we use clamping (causal manipulations of activations throughout training) to identify three, smoothly evolving underlying subcircuits whose interaction may be causing the seemingly discontinuous phase change. Specifically, the third key evolving subcircuit is responsible for copying input labels to the output (highlighted in blue, Figure 1a), a function that was often believed to be easy (with most prior work focusing on the "match" operation instead, where an induction circuit has to find the right token to attend to). Finally, we show how data properties influence the timing of the phase change, and how this shift in induction circuit formation can be better understood by individually understanding the data-dependent formation of each of our identified subcircuits.

As a resource to the community, we open-source our codebase at `https://github.com/aadityasingh/icl-dynamics`, providing a tool for modifying activations throughout training and conducting further causal analyses of transformers' circuit elements. Our work and tooling represents an important step in mechanistic understanding of training dynamics, applied here to induction circuit formation, and we hope it spurs further progress on understanding how different computations in transformers are learned.

2. Methods

2.1. Experimental setup

We train transformer models on a FSL task. Sequences are series of exemplar-label pairs, followed by a query exemplar (see Figure 1b). Exemplars come from the Omniglot dataset

[3] We also gray out the previous token head operation for copying label tokens. This behavior does not robustly emerge as it isn't necessary since we only train on sequences of a fixed length.

```
def example_pattern_preserving_ablation(model, sequence, preserve, ablate):
  # preserve - list of (layer, head) pairs to preserve patterns on
  # ablate - list of (layer, head) pairs to ablate
  base_cache = model.call_with_all_aux(sequence, cache=None, cache_mask=None)
  cache = jax.tree_map(jnp.zeros_like, base_cache)
  cache_mask = jax.tree_map(partial(jnp.zeros_like, dtype=bool), base_cache)
  # preserve head patterns
  for layer, head in preserve:
    cache[layer][head]['attn_pattern'] = base_cache[layer][head]['attn_pattern']
    cache_mask[layer][head]['attn_pattern'] = True
  # perform other ablation, example: turning off heads
  for layer, head in ablate:
    cache[layer][head]['out'] = 0
    cache_mask[layer][head]['out'] = True
  return model(sequence, cache=cache, cache_mask=cache_mask)
```

Figure 2. Example pseudocode demonstrating a pattern preserving ablation using our framework.

(Lake et al., 2015), a common benchmark for FSL also used by prior work on ICL in transformers (Chan et al., 2022; Singh et al., 2023). Omniglot consists of 1600 classes of 20 exemplars. Most of our experiments use a simplified dataset with a random set of $C = 50$ classes and $E = 1$ exemplar per class, which we found sufficient to elicit the relevant phenomena. To obtain input embeddings, we feed Omniglot images through an ImageNet-pretrained, frozen Resnet18 encoder (He et al., 2015; Russakovsky et al., 2015).

To isolate FSL capabilities, we use the standard meta-training setup where labels are randomized for each sequence. Specifically, each sequence can be viewed as a 2-way, 1-shot classification problem (see Figure 1b). Labels are randomly selected from L one-hot labels. To obtain input embeddings, we use a standard learnt embedding layer.

In our default setting[4] ($C = 50, E = 1, L = 5$), we have a total of 78400 unique training sequences (Appendix C). On this data, we train causal, 2-layer attention-only transformer models, as used by prior work (Olsson et al., 2022), to study induction circuit formation. Model and optimizer hyperparameters can be found in Appendix A.

To test generalization, we consider two test sets over held-out classes and relabelings. Specifically, for "test (exemplars)", we evaluate performance on a random, held-out set of $C_{test} = 100$ classes, which verifies the generality of the "match" operation of induction circuits. For "test (relabel)", we hold out and test on a fixed percentage (20%) of pairs of labels. This ensures that, while all labels are seen during training, not all pairs are. This test set verifies the generality of the "copy" operation of induction circuits.

2.2. An artificial-optogenetics framework

One of the contributions of this work is a novel training and analysis framework that easily exposes activations, and allows causal manipulations throughout training (as opposed to only after training, as in prior work). The framework enables a wide range of analyses—in this work, we use it to conduct targeted analyses of induction circuit formation via clamping. Our codebase is implemented in Equinox/JAX (Kidger & Garcia, 2021; Bradbury et al., 2018), natively supports up to 50x speed-ups with `jax.jit`,[5] and is open-source.

In contrast to the standard practice of having modules with a forward function implemented as `__call__`, we have an underlying `call_with_all_aux` method which returns a pytree of all intermediate activations. The `__call__` method wraps `call_with_all_aux` and returns just the output. During the training process, `__call__` is used. For analysis, `call_with_all_aux` can be easily used to expose all intermediate activations.

To allow for editing/ablating activations throughout training (or just post-hoc, as done in prior work), `call_with_all_aux` accepts `cache` and `cache_mask` arguments, which have the same shape as the output pytree of `call_with_all_aux`. These caches can be used to insert activations into the network, e.g. clamping activations at particular values during training. Combined with the functional, automatic differentiation of JAX, these caches can be input- and model-dependent, while still allowing for proper gradient routing. For example, Figure 2 shows how easy it is to implement the "pattern-preserving[6] ablation" of prior work (Olsson et al., 2022). Importantly, our framework allows implementing such causal manipulations *throughout training*, which enables us to isolate the *dynamics* of subcircuit formation (Section 4.2). While prior work[7] was restricted to conducting mechanistic analyses on check-

[4] In Section 5, we vary these parameters to more deeply study the dependence of induction head formation on data properties.

[5] See Appendix B for details.

[6] Here "patterns" are used to refer to the post-softmax attention scores from one token to others.

[7] We note that some prior work in vision (Ranadive et al., 2023) or masked language models (Chen et al., 2024) has attempted to causally manipulate learning, but primarily through the use of regularizing losses. Our framework permits more direct interventions (on activations) throughout training.

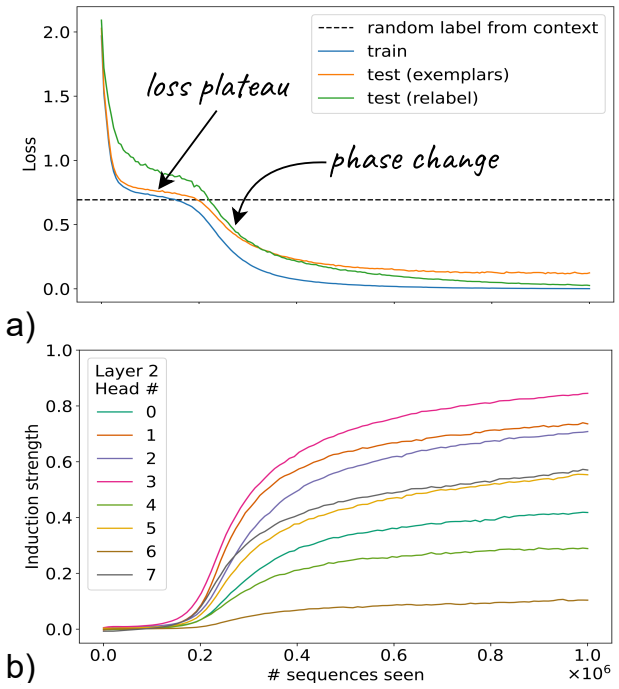

Figure 3. **a)** Train and test loss curves. Transformers exhibit strong generalization to unseen classes (orange) and label pairs (green). The loss dynamics reveal a plateau (which may be indicative of a saddle point), where the model is randomly guessing between the two labels present in context (so it has 50% accuracy, instead of the chance level of 20% when there are $L = 5$ labels). Then, there's a phase change in the loss which corresponds to the formation of induction circuits, reproducing the finding of Olsson et al. (2022). **b)** Induction head strength for each Layer 2 head plotted over time. Induction head strength is defined as the attention weight given to the correct label token minus that to the incorrect token. All heads appear to have some induction-like behavior, with Head 3 being the strongest and emerging first.

points from training, providing only correlational evidence, our framework allows *causal* interventions on learning dynamics (see Appendix D.2 for further discussion).

3. Induction head emergence and diversity

We first establish that training transformers as described in Section 2.1 leads to ICL abilities and induction head emergence. Figure 3a shows loss dynamics. Initially, the transformer reaches a plateau (perhaps indicative of a saddle point) where its accuracy is ≈50% (a loss of $\log 2$). This corresponds to the network having learned to randomly select from the two labels in context,[8] rather than the full set of all possible labels ($L = 5$). Then, a sudden phase change in the loss leads to near-0 loss and near-perfect accuracy on the task (99.99%). At this point, we also observe strong generalization to unseen exemplars and unseen label pairs, indicating the transformer has learnt a general ICL mechanism and is not simply memorizing the training sequences.

We find that the phase change in the loss corresponds to the formation of induction circuits. We measure the "induction strength" of a head, a common progress measure, as a difference in attention weights from the query token: [attention to the correct in-context label token] - [attention to the incorrect in-context label token]. For example, on the sequence "A 0 B 1 A", this would be the difference [attention weight from 5th token to 2nd token] - [attention weight from 5th token to 4th token]. We see that the emergence of induction heads corresponds to the phase change in the loss.

To verify that Layer 2 induction heads instead of Layer 1 heads are primarily contributing to task performance, we use our artificial optogenetics framework to ablate the residual connection between Layer 1 heads and the output. We do this by setting Layer 2 activations (attention patterns and values vectors) to those from an un-ablated forward pass (keeping induction circuits intact) and then ablating Layer 1 head outputs. Such an ablation leads to a negligible change in loss and accuracy (which drops by 0.01% to 99.98%), indicating that Layer 1 heads are not responsible for task performance. When we apply this ablation across checkpoints, we see that earlier in training, Layer 1 heads are contributing to minimizing the loss (perhaps through the use of skip-trigrams; Elhage et al., 2021), but no longer towards the end (Appendix Figure 12).

3.1. The additive nature of induction heads

One of the notable features of Figure 3b is the fact that many heads become strong induction heads. What purpose do all these heads serve? Are they necessary to solve the task?

To answer these questions, we consider two types of ablations of the fully trained network (Figure 4a). The first type of ablation is similar to prior mechanistic work (Olsson et al., 2022), where we ablate a given head and observe the effect on network performance (triangles, Figure 4a). However, we found that this type of ablation was not very informative, and even potentially misleading, as ablating any head except the strongest leads to an almost unobservable decrease in task performance. Next, we considered an ablation inspired by work on head pruning (Michel et al., 2019), where all Layer 2 heads are turned off and only one head is turned on (circles, Figure 4a). This ablation allows us to identify the positive contribution of each head to the task. [9] As seen in Figure 4a, the latter ablation (circles) reveals a

[8] The model tends to place equal weight on the two labels, rather than randomly putting high weight on one.

[9] There are some intuitive connections to the commonly used *logit lens* (nostalgebraist, 2020). However, we emphasize that our method relies on causal ablations only and takes into account LayerNorm, which is often a thorn for interpretability research. We measure task performance with the ablation applied, instead of a change in logits.

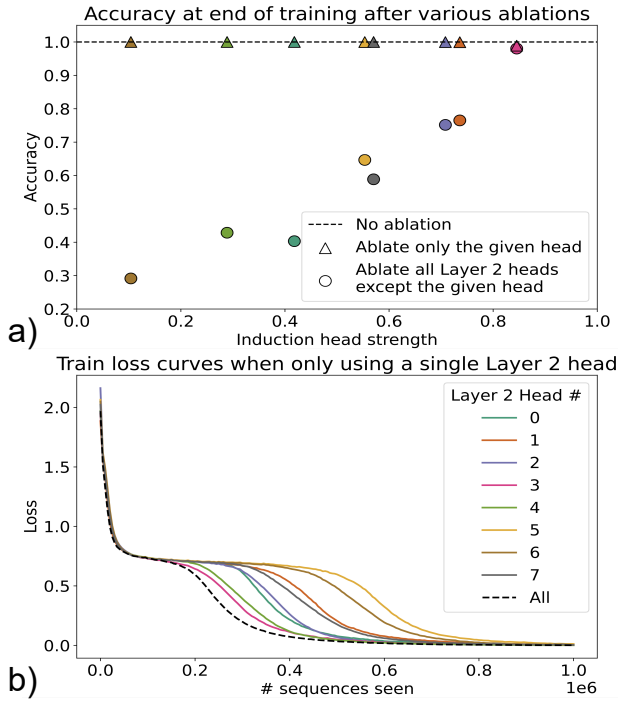

Figure 4. **a)** Effect of various ablations on accuracy (effects on loss are shown in Appendix Figure 13). Ablating any single head (triangles) leads to virtually no decrease in task performance, with the exception of Head 3, which leads to a 1% decrease. Ablating all but a specific head (circles) isolates how useful that specific head is, which correlates well to the induction strength (x-axis), the metric from Figure 3b. Importantly, ablating Head 3 (pink triangle) performs very similar to ablating everything except Head 3 (pink circle), which indicates the other heads function additively, and together can make up for the deletion of Head 3. **b)** Training loss curves when training from scratch with only a single head from Layer 2 active (and the rest ablated). Black dotted line is the loss profile from the training run in Figure 3. Colors chosen to match Figure 3b. Each Layer 2 head on its own can learn to solve the task, though the timing of the phase change shifts and learning is slower.

clearer picture of the importance of induction heads, showing the correlation between induction head strength (which is calculated using attention scores), and task performance.

We see that despite Head 3 being able to mostly solve the task on its own (achieving an accuracy of 98%), other single heads can still achieve strong performance. Furthermore, heads seem to have an additive effect, most clearly demonstrated[10] by comparing the pink triangle (which corresponds to ablating Head 3) and pink circle (which corresponds to only keeping Head 3). Both achieve around 98% accuracy, indicating that the other, weaker heads together can make up for the loss of Head 3. These results echo similar findings in head pruning (Michel et al., 2019) and layer-wise redundancy of language models (McGrath et al., 2023).[11] Our analysis builds on this work and cautions against the use of purely knock-out ablations in mechanistic analyses, which may overlook redundancies and incorrectly dismiss network pieces as not implementing a certain function.

3.2. Networks use additional capacity for faster training, even if it is not necessary

Given the additive qualities observed above, a natural question is—does the network need multiple induction heads to *learn* to solve the task? We know that we can ablate many heads and not hurt performance *at the end of training*, but perhaps these heads play a crucial role *during training*? To answer these question, we use our artificial optogenetics framework to ablate all Layer 2 heads but one *throughout training*. We do this for each Layer 2 head and show train loss curves in Figure 4b (all networks trained from the same random initialization). We find that final networks attain the same near-perfect train performance as the network trained with 8 heads in Layer 2 (black dotted line in Figure 4b, blue line in Figure 3), but with slightly slower dynamics (delayed phase change timing). Thus, it seems the network may make use of the additional capacity during training to learn faster, even though it is not strictly necessary for the task.

These experiments connect to notions of lottery tickets[12] (Frankle & Carbin, 2018; Nanda, 2023). Specifically, we note that different Layer 2 heads exhibit phase changes at different times (but all eventually learn to solve the task). Head 3 is the "quickest to learn" in Figure 4b, which may also be the reason it becomes the strongest when training with all heads (Figure 3b)—perhaps heads are racing to minimize the loss, similar to the mechanism in Saxe et al. (2022). Once Head 3 emerges, though, the ordering of phase changes in Figure 4b does not match emergence in the full network (Figure 3b)—e.g., Head 1 is the second to emerge and second strongest, but the third slowest when trained on its own. Additive interactions could be causing Head 1 to be learned sooner, due to the presence of Head 3 (we further explore such interactions in Appendix H).

3.3. Previous token heads influence induction heads in a many-to-many fashion

Prior work (Olsson et al., 2022) has stressed the two-layer nature of induction circuits, where induction heads in later layers rely on the output of previous token heads in earlier layers to attend to the correct token. By inspecting attention patterns, we identify three previous token heads in Layer 1: Heads 1, 2, and 5. Through a series of ablation analyses, we

[10] We provide further evidence in Appendix G Figure 14.

[11] Such redundancy is also a common observation in neuroscience (Hennig et al., 2018).

[12] Lottery tickets typically refer to sub-networks sufficient to solve the task determined largely by initialization.

find that each of these heads is enough to elicit above-chance accuracy in at least one of the strongest induction heads in Layer 2 (see Appendix Figure 11b). These results indicate a many-to-many wiring between previous token heads and induction heads, with previous token heads operating redundantly (just as induction heads operate redundantly to solve the task). Full details are provided in Appendix E.

4. Three interacting subcircuits give rise to the phase change in induction head formation

We now turn our attention to the dynamics of induction circuit formation. Prior work (Jermyn & Shlegeris, 2022) indicates that reverse-S phase changes in the loss are often due to multiple interacting components. Below, we delineate the necessary computations that comprise an induction circuit, then study their formation dynamics in isolation using our artificial optogenetics framework.

4.1. Terminology for this section

For clarity, we isolate and define the 5 key computations involved in an induction circuit, as illustrated in Figure 5c. In Section 4.2, we'll show that these 5 computations can be grouped into the 3 primary interacting subcircuits.

Step 1 (PT-attend) Layer 1 head attends to previous token.

Step 2 (PT-copy) Layer 1 head copies previous token value into the current token's residual stream.

Step 3 (Routing Q/K/V) Layer 2 head reads Q/K/V[13] from the correct subspaces of the residual stream: Q from the residual, K from the output of the Layer 1 head, and V from the residual or from other Layer 1 heads.[14]

Step 4 (IH-Match) Layer 2 head matches Q to "same" K.

Step 5 (IH-copy) Layer 2 copies the value to the output.

Steps 1 and 2 comprise what's canonically known as a previous token head, while Steps 3-5 form the induction head. The "match operation" (Figure 1a, blue) consists of Steps 1, 2, 3qk, 4. The "copying operation" (Figure 1a, green) consists of Steps 3v, 5.

4.2. *Clamping* computations to understand the causal effects of subcircuits on dynamics

Now that we've defined these computations, we ask: how do the learning dynamics of each of these computations causally influence the learning dynamics of the full model? Can we explain seemingly discontinuous qualitative phase changes, such as the induction circuit formation, in terms of subcircuits with smoother, exponential learning dynamics?

To motivate our analysis, we extend the toy model from Jermyn & Shlegeris (2022) to the case of three interacting vectors.[15] Specifically, we have three vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ that are learnt using a simple mean-squared error loss, corresponding to learning their values via a tensor product:

$$\mathcal{L}(\mathbf{a},\mathbf{b},\mathbf{c}) = \frac{1}{2}\sum_{i,j,k}\left(a_i^{true}b_j^{true}c_k^{true} - a_i b_j c_k\right)^2$$

The interaction between the evolution of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ gives rise to a phase change in dynamics (black, Figure 5a) following a loss plateau caused by a saddle point (see proof in Appendix F). If we clamp $\mathbf{c}$ to its final value, learning is faster (blue, Figure 5a), but the co-evolution of $\mathbf{a}$ and $\mathbf{b}$ still results in a phase change. If we clamp $\mathbf{b}$ *and* $\mathbf{c}$ (purple, Figure 5a), we isolate a smooth exponential loss curves corresponding to the formation dynamics of $\mathbf{a}$. This toy model gives us the intuition that phase changes may be caused by two or more subcircuits that, when learned on their own, evolve exponentially, but when co-evolving, induce phase changes. Furthermore, these subcircuits could be isolated by clamping all the other interacting components throughout training. Notably, it's not clear how to use existing interpretability techniques, such as progress measures (Nanda et al., 2023; Reddy, 2023), to uncover these subcircuits (Appendix D.2).

We apply these insights from the toy model to understanding the key underlying subcircuits in an induction circuit. Specifically, we aim to isolate subcircuits where each individual subcircuit does not exhibit a phase change in its learning dynamics, like variables $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ in the toy model. We restrict ourselves to considering the induction circuit with induction head Layer 2 Head 3, a minimal setting where we observe a phase change in the loss dynamics (see Figure 4b). In Section 5, we will show how understanding these subcircuits can help us understand the data-dependence of phase change timing, affirming the importance of discovering smoothly-learned underlying subcircuits.

To isolate these subcircuits, we iteratively clamp the computations delineated in Section 4.1 at the activation level,[16] using our artificial optogenetics framework. Results are shown in Figure 5b. We start by clamping Step 1 (PT-attend), the previous token head attention pattern (an oft-used progress measure for IH formation). We train a network where one[17]

[13] Q/K/V are the query, key, and value in the attention computation of a transformer (Vaswani et al., 2017).

[14] Unlike Olsson et al. (2022), we observe non-trivial V-composition in our networks. See Appendix E for details.

[15] Our tensor product formalism for analyzing sub-circuits could be extended beyond 3. We chose 3 as we ended up finding 3 primary subcircuits contributing to induction circuit formation. We also assume that $\mathbf{a}^{true}, \mathbf{b}^{true}, \mathbf{c}^{true} \neq \mathbf{0}$.

[16] Specifics of our implementation are provided in Appendix D.1.2 and why fixing weights to those at the end of training is not sufficient (see Appendix D.1.1).

[17] We also conducted experiments where multiple previous token

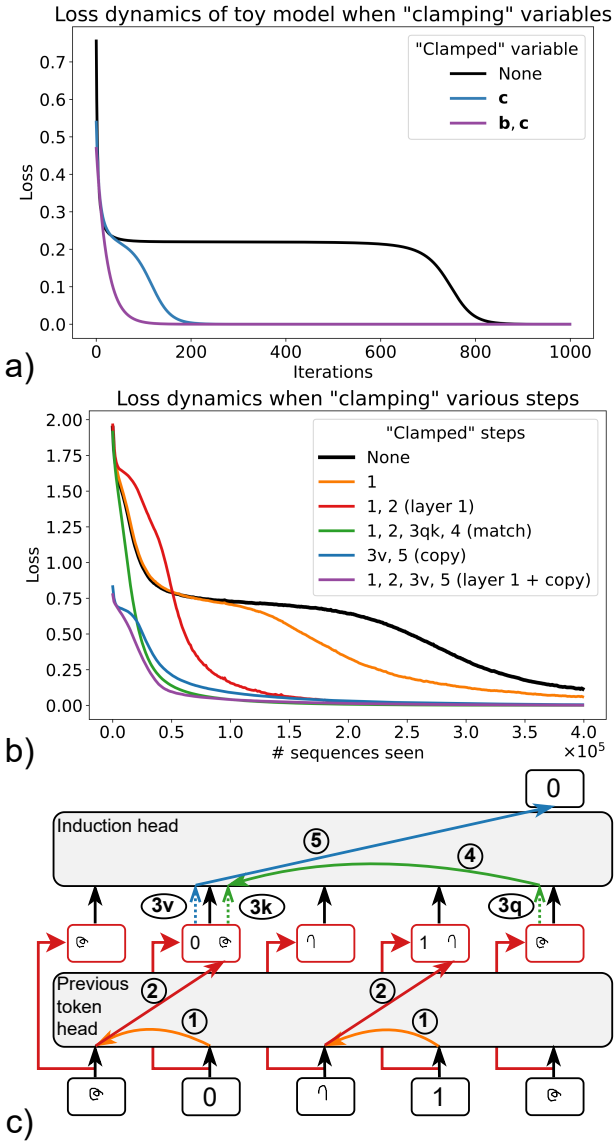

Figure 5. **a)** Loss dynamics when clamping various variables in the toy model presented in Section 4.2. Black shows the learning dynamics when no variable is clamped. Only when all other interacting components (**b**, **c**) are clamped does the loss curve become exponential. **b)** Loss dynamics when clamping various computations outlined in Section 4.1. Black shows the training dynamics of the full network with nothing clamped. **c)** Induction circuit schematic (from Figure 1a), with computation steps labeled. Arrow colors chosen to illustrate which steps are additionally clamped.

of the Layer 1 heads is clamped from the start of training to a perfect PT-attend pattern. We see the loss dynamics still exhibit a phase change (orange), with a similar profile, indicating the rest of the computations (2-5) contain interacting subcircuits. Next, we clamp the full Layer 1 computation (PT-attend *and* PT-copy) and ensure that the superimposed residual and output of Layer 1 are disentangled (see Appendix D.1.2 for details). We find that the dynamic profile of the loss shifts substantially (red). While a phase change is still observed, it is much sharper, lasting only $\approx 7.5e4$ iterations (compared to $2e5$, black). This suggests that the formation of Layer 1 attention+output circuits is likely a key sub-component, but there are still interacting sub-components in Steps 3, 4, and 5.

We take this analysis further and clamp the entire match operation (green in Figures 1a, Figure 5), so only the copy operation (blue, Figure 1a) is being learned. We see this loss profile is smoothly exponential, suggesting there are no sub-components hidden within it. We then do the opposite, clamping the copy operation and focusing on the dynamics of match operation formation. We see a small saddle point followed by a quick phase change (lasting $\approx 0.5e5$ iterations), indicating that the match operation should be decomposed further. We do so by clamping both layer 1 and copy components (Steps 1, 2, 3v, 5), which results in a more exponential loss profile (purple, Figure 5).

Given these results, we believe the phase change is primarily determined by three interacting subcircuits:

Subcircuit A: Layer 1 Attending to previous token and copying it forward. Comprised of PT-attend and PT-copy (Steps 1 and 2).

Subcircuit B: IH QK Match Matching queries to keys in the induction head. Comprised of Routing Q, Routing K and IH-match (Steps 3qk and 4).

Subcircuit C: Copy Copying of input label to output. Comprised of Routing V and IH-copy (Steps 3v and 5).

Prior work has often hypothesized the first and second components; our analysis verifies these intuitions and critically identifies the third interacting component through the use of *clamping* (causal manipulations of activations throughout training).

5. Subcircuits can explain data-dependent shifts in phase change timing

To demonstrate the relevance of the identified subcircuits, we turn towards explaining changes in the timing of the phase change as we modify various data properties. Pre-

heads were provided and observed similar loss dynamics. See Appendix D.1.2.

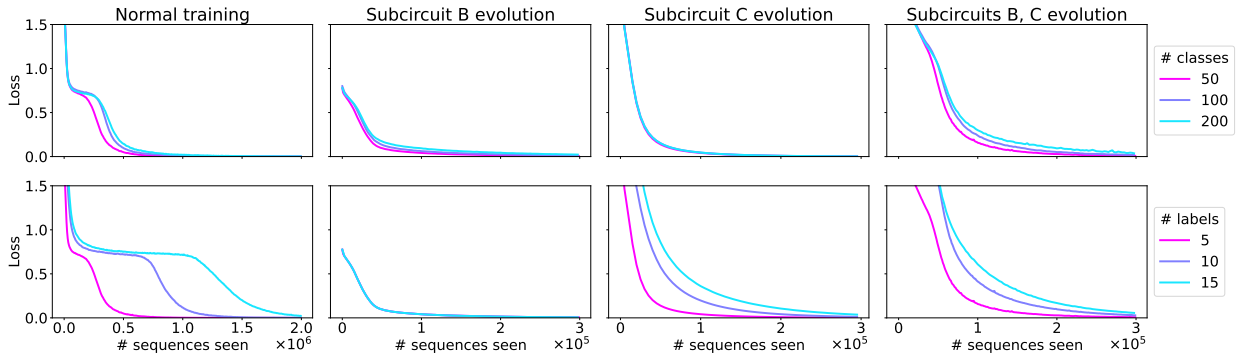

Figure 6. Data-dependent learning dynamics of the induction circuit and assorted subcircuits. Top row shows curves as the # of classes is increased, bottom row shows curves as the # of labels is increased. Left-most column shows training loss over time without any clamps on subcircuits. Middle two columns looks at the evolution of individual subcircuits identified in Section 4.2, by clamping the other two subcircuits. Right-most column shows evolution of a composite circuit (by only clamping Subcircuit A). Middle two plots clearly show how different subcircuits depend on different data properties, and how this can explain the overall difference in learning dynamics.

vious work has shown that such data properties can affect the dynamics and emergence of ICL (Chan et al., 2022; Raparthy et al., 2023; Reddy, 2023; Yu et al., 2023). Specifically, we consider two possible variations: increasing the number of classes C or the number of labels L.

In Figure 6, we find that increasing both the number of classes and number of labels lead to a delay in the formation of induction heads, mostly seen as a shift in the timing of the phase change (leftmost column, Figure 6). This may be because the task becomes more challenging due to the higher data diversity. To better understand what leads to this delay, we look at the effect of these data properties on the formation of some of the subcircuits identified in Section 4, specifically Subcircuits B and C (middle two columns, Figure 6). We find that increasing the number of classes makes learning IH QK Match (Subcircuit B) harder while not increasing the difficulty of Copy (Subcircuit C), thus implying that a delay in Subcircuit B formation causes the delayed phase change. On the other hand, we find that when increasing the number of labels, IH QK Match is learnt as quickly, while copying becomes way harder, indicating that a delay in Subcircuit C formation causes the delayed phase change. These results make intuitive sense—more classes may necessitate learning a higher precision QK matching operation (Subcircuit B), and more labels may necessitate learning a higher precision copying operation (Subcircuit C). Our analysis enables this rigorous decomposition of the phase change of the full model into delays of formation of interacting subcircuits.

To emphasize the importance of considering these subcircuits individually, we consider the joint evolution of both Subcircuits B and C (which we can track by clamping Subcircuit A, analogous to the red line in Figure 5b). This would correspond to just looking at an induction head's formation, when a network is provided with previous token heads (and disentangled outputs of Layer 1). We find that both # classes and # labels seems to affect these dynamics, thus not providing as much clarity on how these different properties might differentially affect learning dynamics. Decomposing into the individual subcircuits that we identified in Section 4.2 provides a better understanding of how data properties differentially increase the difficulty of induction circuit formation, leading to delayed phase changes.

Finally, we note that learning of the individual subcircuits is substantially faster than that of the overall dynamics. By considering data-dependent effects on individual subcircuits, we may be able to predict network behavior without having to train for long amounts of time, waiting for a phase change.

6. Related work

In-context learning From its discovery as an emergent ability in LLMs (Brown et al., 2020), ICL has become a focus of study in NLP (Min et al., 2022; Wei et al., 2023b;a) and in synthetic settings (von Oswald et al., 2023; Chan et al., 2022; Xie et al., 2021). The recent findings of transience from Singh et al. (2023) especially motivated our work, which led to our choice of using a similar setup. Future work could extend our analyses to other setups, like linear regression or natural language tasks.

Induction circuits were first found by Elhage et al. (2021), then further investigated in many settings (Olsson et al., 2022; Wang et al., 2023). Reddy (2023) also considers the dynamics of formation, but uses the correlational approach of *progress measures*. We chose to focus on causal methods (see Appendix D.2) such as clamping, enabling us to discover a different set of 3 subcircuits governing induction circuit learning dynamics (Section 4.2).

Tooling for mechanistic interpretability With the increasing prevalence of LLMs and corresponding safety concerns,

mechanistic interpretability has seen a number of toolkits released. Primary open-source frameworks include TransformerLens (Nanda & Bloom, 2022), `nnsight` (Fiotto-Kaufman), and `pyvene` (Wu et al., 2024). All of these frameworks are built on PyTorch, allow studying and manipulating activations from model checkpionts, and natively support many open-source LLMs. In some cases (Fiotto-Kaufman) they even allow intervention on gradients, but it's unclear if any of these frameworks can be used to manipulate activations *during training with proper propagation of gradients*, which is a crucial and unique feature of our framework. Furthermore, while our library does not support as large a breadth of use cases, we hope the vibrant open-source JAX community will find a mechanistic interpretability framework based in JAX and natively supporting compilation useful (including major speedup benefits), similar to the uptake of the `Rust_circuit` library (Goldowsky-Dill et al., 2023) by Rust users. The only JAX framework the authors are aware of is Tracr (Lindner et al., 2023), which allows researchers to quickly construct transformers that implement certain RASP programs: a complementary focus from TransformerLens, `nnsight`, `pyvene`, `Rust_circuit` and our JAX-based artificial optogenetics framework.

7. Discussion

Our work analyzed induction circuits in two-layer attention-only transformers trained on a synthetic ICL task with a focus on the dynamics of formation. We took inspiration from the original discovery of induction circuits (Olsson et al., 2022) and thus focused on smaller models. While a limitation of our work is the small scale compared to frontier foundation model interpretability (e.g., Wang et al., 2022; Merullo et al., 2024; Palit et al., 2023), we chose this setting due to academic compute constraints and a hope to dive deeper in terms of understanding. Our focus on dynamics, as well as our training setup, was inspired by recent work from Singh et al. (2023) asserting that emergent ICL may be a dynamical phenomenon in settings where multiple competing solutions are possible to solve the task.

Through our analysis, we discovered multiple answers to what does and doesn't need to "go right" for an induction head. We found that induction heads operate additively, with multiple heads used to learn the ICL task more quickly despite not being necessary to solve it. This theme of redundancy carried to previous token heads, which we found to exhibit a many-to-many wiring pattern to induction heads. Next, we used the novel *clamping* methodology to identify three smoothly-evolving subcircuits whose interaction may explain the phase change in induction circuit formation. Furthermore, this better understanding helped us explain data-distribution-dependent changes in phase change timing. Of specific interest here was the quicker timelines of subcircuit formation when isolated, which could serve as useful intuition for understanding the effects various data ablations in large model training may have on the learning dynamics, without having to train full networks on each data ablation from scratch. In terms of specific subcircuits we identified, our work demonstrated the crucial role of the copy subcircuit in explaining data-distributional dependent delays in phase change timing (bottom row, Figure 6). This added understanding may have implications for practitioners when selecting a vocabulary size for LLMs: while larger vocabulary sizes are often preferred due to their increased compression ratio (and thus longer effective context), they may make copying more challenging, thereby delaying induction head formation.

Turning back towards our original motivation from the transience of emergent ICL in transformers (Singh et al., 2023), we hope this improved understanding of key subcircuits within induction circuits, as well as novel analysis tools, can in future shed light on why ICL may fade with overtraining.

Beyond our specific results on induction heads, our work contributes a JAX-based artificial optogenetics framework which supports compilation (yielding nearly a 50x speedup on our hardware, see Appendix B) and can be used to modify activations *during training with proper gradient flow*, a methodology we term clamping. This framework permits causal analyses of learning dynamics, whereas prior work mostly applied causal analyses to end networks or checkpoints from normal training (see Appendix D.2 for a comparison). We view our work as taking the first steps with this approach, and are excited to see how future work may combine our clamping methodology with other causal mechanistic interpretability techniques, such as path patching (Wang et al., 2022; Conmy et al., 2023) or causal mediation analysis (Vig et al., 2020; Geiger et al., 2021; Cao et al., 2021; Geva et al., 2023; Finlayson et al., 2021), throughout training for better understanding of transformer circuit learning dynamics.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

We'd like to thank DJ Strouse, Kira Düsterwald, Jirko Rubruck, Andrew Lampinen, Roma Patel, Dan Roberts, Andrey Gromov, and Taylan Cemgil for useful discussions and feedback on early drafts.

A.K.S. and T.M. are funded by the Gatsby Charitable foun-

dation. This work was supported by a Schmidt Science Polymath Award to A.S., and the Sainsbury Wellcome Centre Core Grant from Wellcome (219627/Z/19/Z) and the Gatsby Charitable Foundation (GAT3850). A.S. is a CIFAR Azrieli Global Scholar in the Learning in Machines & Brains program.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.

Boyden, E. S., Zhang, F., Bamberg, E., Nagel, G., and Deisseroth, K. Millisecond-timescale, genetically targeted optical control of neural activity. *Nature Neuroscience*, 8(9):1263–1268, September 2005. ISSN 1097-6256, 1546-1726. doi: 10.1038/nn1525. URL https://www.nature.com/articles/nn1525.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cao, N. D., Schmid, L., Hupkes, D., and Titov, I. Sparse interventions in language models with differentiable masking. *CoRR*, abs/2112.06837, 2021. URL https://arxiv.org/abs/2112.06837.

Chan, S., Santoro, A., Lampinen, A., Wang, J., Singh, A., Richemond, P., McClelland, J., and Hill, F. Data distributional properties drive emergent in-context learning in transformers. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 18878–18891. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/77c6ccacfd9962e2307fc64680fc5ace-Paper-Conference.pdf.

Chen, A., Shwartz-Ziv, R., Cho, K., Leavitt, M. L., and Saphra, N. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms, 2024.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability, 2023.

Cooney, A. [proposal] improve performance with torch.compile, 2023. URL https://github.com/neelnanda-io/TransformerLens/issues/413.

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Sartran, L., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stanojević, M., Stokowiec, W., Wang, L., Zhou, G., and Viola, F. The DeepMind JAX Ecosystem, 2020. URL http://github.com/google-deepmind.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Finlayson, M., Mueller, A., Gehrmann, S., Shieber, S. M., Linzen, T., and Belinkov, Y. Causal analysis of syntactic agreement mechanisms in neural language models. *CoRR*, abs/2106.06087, 2021. URL https://arxiv.org/abs/2106.06087.

Fiotto-Kaufman, J. nnsight: The package for interpreting and manipulating the internals of deep learned models. . URL https://github.com/JadenFiotto-Kaufman/nnsight.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL http://arxiv.org/abs/1803.03635.

Geiger, A., Lu, H., Icard, T., and Potts, C. Causal abstractions of neural networks. *CoRR*, abs/2106.02997, 2021. URL https://arxiv.org/abs/2106.02997.

Geva, M., Bastings, J., Filippova, K., and Globerson, A. Dissecting recall of factual associations in auto-regressive language models, 2023.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. Localizing model behavior with path patching, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Hennig, J. A., Golub, M. D., Lund, P. J., Sadtler, P. T., Oby, E. R., Quick, K. M., Ryu, S. I., Tyler-Kabara, E. C., Batista, A. P., Yu, B. M., and Chase, S. M. Constraints on neural redundancy. *eLife*, 7:e36774, 2018. doi: 10.7554/eLife.36774. URL https://elifesciences.org/articles/36774.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Jermyn, A. and Shlegeris, B. Multi-component learning and s-curves, 2022. URL https://www.alignmentforum.org/posts/RKDQCB6smLWgs2Mhr/multi-component-learning-and-s-curves.

Kidger, P. and Garcia, C. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, 2015. doi: 10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

Lindner, D., Kramár, J., Rahtz, M., McGrath, T., and Mikulik, V. Tracr: Compiled transformers as a laboratory for interpretability. *arXiv preprint arXiv:2301.05062*, 2023.

McGrath, T., Rahtz, M., Kramar, J., Mikulik, V., and Legg, S. The hydra effect: Emergent self-repair in language model computations, 2023.

Merullo, J., Eickhoff, C., and Pavlick, E. Circuit component reuse across tasks in transformer language models, 2024.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? *CoRR*, abs/1905.10650, 2019. URL http://arxiv.org/abs/1905.10650.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Nanda, N. A comprehensive mechanistic interpretability explainer & glossary, Dec 2022. URL https://neelnanda.io/glossary.

Nanda, N. 200 concrete open problems (COP) in MI: Analysing training dynamics, 2023. URL https://www.alignmentforum.org/posts/hHaXzJQi6SKkeXzbg/200-cop-in-mi-analysing-training-dynamics.

Nanda, N. and Bloom, J. Transformerlens. https://github.com/neelnanda-io/TransformerLens, 2022.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability, 2023.

nostalgebraist. interpreting gpt: the logit lens, 2020. URL https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Palit, V., Pandey, R., Arora, A., and Liang, P. P. Towards vision-language mechanistic interpretability: A causal tracing tool for blip, 2023.

PyTorch. We just introduced pytorch 2.0 at the #pytorchconference, introducing torch.compile!, 2022. URL https://x.com/PyTorch/status/1598708792598069249?s=20.

Ranadive, O., Thakurdesai, N., Morcos, A. S., Leavitt, M., and Deny, S. On the special role of class-selective neurons in early training, 2023.

Raparthy, S. C., Hambro, E., Kirk, R., Henaff, M., and Raileanu, R. Generalization to New Sequential Decision Making Tasks with In-Context Learning, December 2023. URL http://arxiv.org/abs/2312.03801. arXiv:2312.03801 [cs].

Reddy, G. The mechanistic basis of data dependence and abrupt learning in an in-context classification task, 2023.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge, 2015.

Saxe, A. M., Sodhani, S., and Lewallen, S. The neural race reduction: Dynamics of abstraction in gated networks, 2022.

Shalizi, C. R. "attention", "transformers", in neural network "large language models", 2024. URL http://bactra.org/notebooks/nn-attention-and-transformers.html#identification.

Singh, A. K., Chan, S. C. Y., Moskovitz, T., Grant, E., Saxe, A. M., and Hill, F. The transient nature of emergent in-context learning in transformers, 2023.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864, 2021. URL https://arxiv.org/abs/2104.09864.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., and Shieber, S. M. Causal mediation analysis for interpreting neural NLP: the case of gender bias. *CoRR*, abs/2004.12265, 2020. URL https://arxiv.org/abs/2004.12265.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent, 2023.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022.

Wang, L., Li, L., Dai, D., Chen, D., Zhou, H., Meng, F., Zhou, J., and Sun, X. Label words are anchors: An information flow perspective for understanding in-context learning, 2023.

Wei, J., Hou, L., Lampinen, A., Chen, X., Huang, D., Tay, Y., Chen, X., Lu, Y., Zhou, D., Ma, T., and Le, Q. V. Symbol tuning improves in-context learning in language models, 2023a.

Wei, J., Wei, J., Tay, Y., Tran, D., Webson, A., Lu, Y., Chen, X., Liu, H., Huang, D., Zhou, D., et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023b.

Wu, Z., Geiger, A., Arora, A., Huang, J., Wang, Z., Goodman, N. D., Manning, C. D., and Potts, C. pyvene: A library for understanding and improving PyTorch models via interventions. 2024. URL arxiv.org/abs/2403.07809.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Yu, K. P., Zhang, Z., Hu, F., and Chai, J. Efficient In-Context Learning in Vision-Language Models for Egocentric Videos, November 2023. URL http://arxiv.org/abs/2311.17041. arXiv:2311.17041 [cs].

Zhong, Z., Liu, Z., Tegmark, M., and Andreas, J. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023.

A. Additional model details

We train causal, 2-layer attention-only transformer models, as used by prior work (Olsson et al., 2022), to study induction circuit formation. Similar to prior work (Chan et al., 2022; Singh et al., 2023), we only train to minimize the cross entropy loss on the last token – the predicted label for the query exemplar. We use RoPE (Su et al., 2021) for positional encoding and a model dimension of 64, with 8 heads per layer. As is common for transformers outside of interpretability work, we are able to use LayerNorm (Ba et al., 2016) when reading in from the residual stream to an attention block – LayerNorm is often excluded in interpretability work because it complicates standard analyses (Elhage et al., 2021), but our causal framework works by modifying activations instead of inspecting weights or assuming linearity, circumventing the issues.

For optimization, we use Adam (Kingma & Ba, 2015) from the `optax` library (DeepMind et al., 2020), with parameters $\beta_1 = 0.9, \beta_2 = 0.999$. We use a constant learning rate of $1e-5$ and a batch size of 32 sequences.

B. Computational cost to run experiments

All experiments were run on a 2018 MacBookPro with a 6-Core Intel Core i7 processor and 16GB of RAM. Standard training runs (e.g., Figure 3) took $\approx$ 7 minutes each to complete (for $1e6$ sequences seen). Clamped runs were more expensive per iteration (as some clamps require multiple forward passes), but overall took even less time since they converge in fewer sequences, note x-axis in Figure 5. This speed was possible due to the use of `jax.jit`, without which each training run would take $\approx$ 330 minutes on the same hardware. Though PyTorch has recently added `torch.compile` to impart similar functionality (PyTorch, 2022), interpretability frameworks based in PyTorch are currently not compatible with `torch.compile` due to their use of hooks (Cooney, 2023). Our JAX-based framework is natively compatible with `jax.jit`, as the `call_with_all_aux` formalism uses PyTrees, which could possibly mean speed-ups for other interpretability researchers (as it did for us).

C. Dataset size calculation

For a dataset of C classes, E exemplars, L labels, and assuming a fraction F of relabelings being used for training, our total dataset size is

$$\binom{C}{2} \cdot F \cdot \binom{L}{2} \cdot E^3 \cdot 2 \cdot 2 \cdot 2 = 2FE^3C(C-1)L(L-1) \Rightarrow 1.6C(C-1)L(L-1)$$

for our settings where $E = 1$ and $F = 0.8$ are fixed.

D. Additional details on clamping

D.1. Implementation details of clamping for induction head formation

In Section 4.2, we detail the methodology and inspiration for clamping various subcircuits. In this subsection, we detail the specifics of implementation of each of the clamped steps.

D.1.1. Clamping weights vs activations: Complications due to routing

We first demonstrate that clamping isn't as easy as just fixing weights during training. Recall that we want to clamp computation steps (Section 4.1) that comprise an induction circuit. There are many ways (in terms of weights) the network can implement some of these computation steps, as many of them involve reading from and/or writing into a subspace of the residual stream (e.g., Steps 2 and 3). Arbitrary rotations could be applied to these subspaces.[18] The specific subspaces used appear to be initialization dependent, as shown in Figure 7.

We consider fixing the weights of Layer 1 to match those of networks pre-trained on our task, but from different initializations. Specifically, we train networks on our task from different initialization seeds (which we will refer to as seeds[19] 5, 6, 7). All networks learn induction circuits, with multiple previous token heads appearing in Layer 1. We then train a network initialized from seed 5 with the residual stream after Layer 1 clamped to the residual stream after Layer 1 of one of the fully trained networks (equivalent to fixing all the weights up to this point). Note that these activations are fixed for a given input,

[18] This has been referred to the as *identification failures* by some (Shalizi, 2024).

[19] These values are arbitrary and are the first three we tried. We didn't use seed 0 as this was the seed for our data sampler.

so gradients are only flowing to Layer 2 of the model (as desired, since we are clamping Layer 1 to be an externally "perfect" Layer 1 based on weights). The clamped Layer 1 is theoretically perfectly performing Steps 1 and 2. However, we can see that there is substantial variance when the weights used are from the same seed as that used for the Layer 2 initialization (red, Figure 7) versus when the weights used are from a different seed (blue or green, Figure 7). These results point to the importance of Step 3, routing, and tell us that fixing weights may not be the best way to clamp subcircuits as it may be confounded by the specific methodology used (specifically, which network the weights come from).

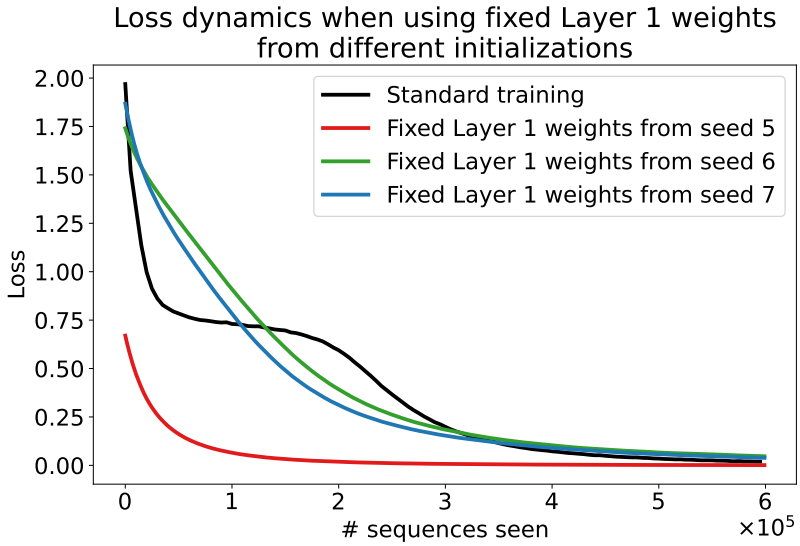

Figure 7. Loss curves when fixing Layer 1 weights to be those found at the end of training with various seeds. We see that when using the Layer 1 weights from the end of training off of the same initialization seed (seed 5) learning is way quicker, indicating that specific implementations of computations 1 and 2 in Section 4.1 are tightly coupled to initialization. Furthermore, these results suggest that fixing weights is not the ideal form of clamping as it is very susceptible to correlations between the chosen weights and network initialization.

D.1.2. Activation clamps that we used

Building off this insight, we instead implement our clamps at the activation level. While this does restrict the set of useful clamps we may apply, all the clamps used in Section 4.2 can be expressed at the activation level. We detail each clamp below:

Clamping Step 1 (PT-attend) We fix the attention pattern of a Layer 1 head to always attend to its previous token (with post-softmax value 1), corresponding to clamping Step 1 from Section 4.1. In Section 4.2, we clamped Layer 1 Head 2's attention pattern. In Figure 8, we show the effect of this clamp applied to different sets of Layer 1 heads. We find that overall there is little difference, unless all heads are clamped, in which case learning is severely slowed. We hypothesize that some Layer 1 heads speed up learning of copy circuits (which is consistent with the findings of V-composition in Appendix E) – by forcing all Layer 1 heads to have the previous token attention pattern, we slow this mechanism. We didn't investigate this further, however, as this was not the focus of our work, and instead used a clamp on just one previous token head for our main experiments.

Clamping Steps 1, 2 (layer 1) To implement this clamp while taking into account the concerns related to routing from Appendix D.1.1, we consider how Layer 1 outputs are used by Layer 2. We utilize two forward passes through the network. In the first, we set the input to Layer 2 to be equivalent to the output from previous token heads and perfect query routing. For example, for the sequence "A 0 B 1 A", this would look like "A A 0 B A", where we note that the first token cannot attend to a previous token (hence it still has 'A'), the next 3 tokens attend to previous tokens (so '0 B 1' becomes 'A 0 B'), and the query token still routes from the input ('A'). We directly pass the embeddings of these tokens to Layer 2, and cache the *patterns*. Then, we do a second forward pass using these patterns, except now we clamp the input to Layer 2 to match the input to the network ("A 0 B 1 A") which is what the value routing should be reading from. We note that this clamp is making the task a bit easier for the network as the vectors that Routing QK needs to read from and Routing V needs to read from are not additively superimposed, but these routing steps need to learn to read from the correct subspaces. We opted for

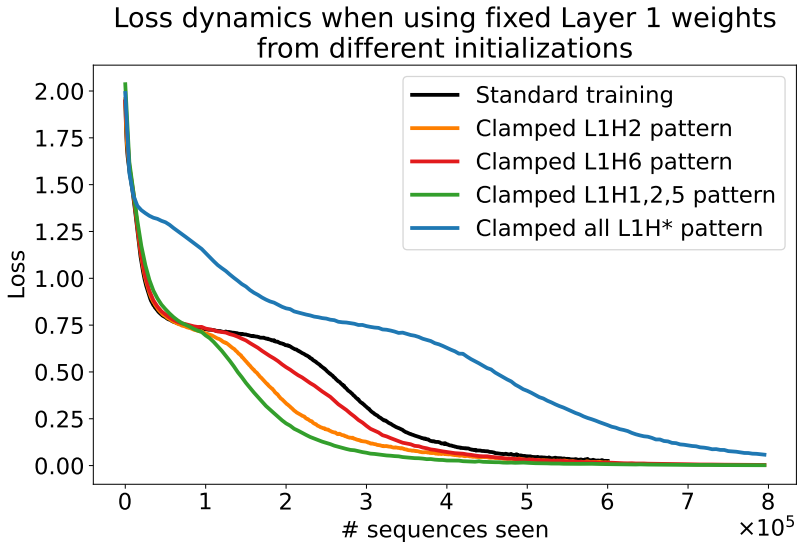

Figure 8. Effect of different implementations of the previous token attention pattern clamp (clamping computation Step 1, using the terminology of Section 4.1). We see that most versions of the clamp have little effect on dynamics (just slight left or right shifts in the phase change), with the exception of clamping all heads in Layer 1. We suspect this may be due to some Layer 1 heads V-composing with Layer 2 heads to make copy circuits easier to learn.

this option as it makes it even more telling that there are some subcircuit interactions in the remaining steps after applying this clamp.

Clamping Steps 1, 2, 3qk, 4 (match) We implement this by fixing the attention pattern of a Layer 2 head (we picked Layer 2 Head 3 as described in Section 4.2) to be that of a perfect induction head – 1 to the correct label token and 0 to all other tokens. The aim of this clamp is to make sure the correct label is matched to (thus isolating the copy operation), which this implementation suffices to do.

Clamping Steps 3v, 5 (copy) We implement this clamp by considering the attention logits to the two labels in-context at a given induction head (we picked Layer 2 Head 3 as described in Section 4.2) and setting the output logits to them. For example, on the sequence "A 0 B 1 A", the attention logits from the query token 'A' to the label tokens '0' and '1' would be the output logits for labels 0 and 1. The other output logits are set to $-1e9$. The aim of this clamp is to make sure labels are perfectly copied (thus isolating the match operation), which this implementation suffices to do.

Clamping Steps 1, 2, 3v, 5 We implement this clamp by combining the clamp for Steps 1, 2 (layer 1), and the clamp for Steps 3v, 5 (copy). As each of these clamps are disjoint, it's easy to apply both, serving the inteded purpose.

D.2. Contrasting *clamping* to progress measures

One prevailing current method in mechanistic interpretability for understanding training dynamics is to use *progress measures*. Progress measures track quantities through the normal training process, often using causal ablations of intermediate network checkpoints.[20] Despite this causal nature per checkpoint, we note that progress measures are actually *correlational* metrics, as the learning dynamics of the quantities they track may or may not be influencing the learning dynamics of the overall network. As such, we find our clamping method (Section 4.2) to be more useful for uncovering underlying subcircuits.

We first illustrate this with the toy model explored in Section 4.2. To track the evolution of "subcircuit" $\mathbf{a}$, we might consider various progress measures. The first problem we run into is that $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are only specified up to scalars – if $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is a solution, so is $(-\mathbf{a}, -\mathbf{b}, \mathbf{c})$. We find this to be an interesting toy analog to the rotations problem described in Appendix D.1.1. Given this, we might consider tracking the squared cosine distance between $\mathbf{a}$ and $\mathbf{a}^{true}$. Another progress measure we

[20] Our results in Figure 3b and Figure 12 are examples of progress measures, with the latter involving a causal ablation per checkpoint.

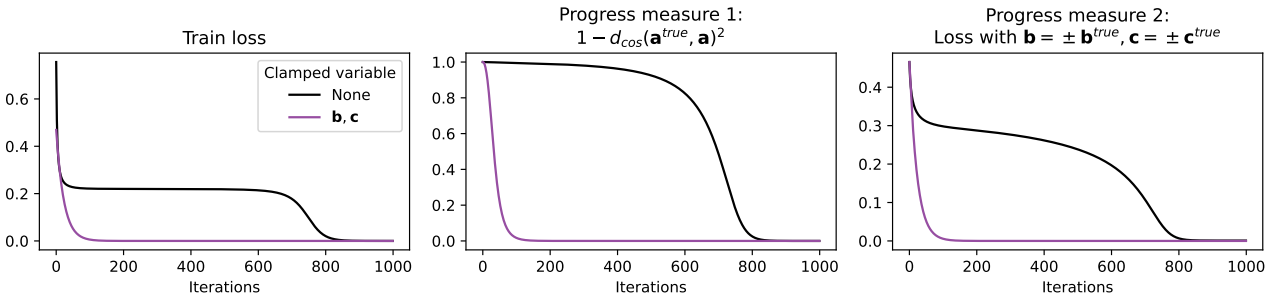

Figure 9. Progress measures vs clamping in a toy model. Black curve indicates standard training, as in Figure 5a. Leftmost plot shows the training loss. Right two plots show two different progress measures for the learning of a. We note that clamping (purple curve, leftmost plot) gives a clearer signal of the relevant subcircuit than progress measures in standard training (black curves, right two plots).

may consider, which offers a direct analog to clamping, is the loss if $\mathbf{b} = \mathbf{b}^{true}, \mathbf{c} = \mathbf{c}^{true}$.[21] Note the difference between this progress measure and clamping is that with the progress measure, we're considering the evolution of $\mathbf{a}$ during normal training and setting $\mathbf{b} = \mathbf{b}^{true}, \mathbf{c} = \mathbf{c}^{true}$ at every checkpoint. With clamping, we set $\mathbf{b} = \mathbf{b}^{true}, \mathbf{c} = \mathbf{c}^{true}$ throughout training and then consider the evolution of $\mathbf{a}$.

Results are shown in Figure 9. We find that both progress measures considered, cosine distance to true value and loss assuming $\mathbf{b}, \mathbf{c}$ fixed to those at the end of training, exhibit a phase change when observed in isolation (black curves, right two columns, Figure 9). On the other hand, as seen in Figure 5a and reproduced in the leftmost plot of Figure 9 in purple, the loss curve when training with clamping is smoothly exponential and easily interpretable.

We extend results from the toy model to our setting, and consider progress measures corresponding to two of our clamped experiments: that of clamping the previous token attention pattern (clamp Step 1, orange curve, Figure 5) and that of clamping the whole match operation (clamp Steps 1, 2, 3qk, and 4, green curve, Figure 5). We construct analogous progress measures by perfecting either the previous token head attention pattern or induction head attention pattern for each checkpoint in standard training and considering the loss. Figure 10 shows the results, with a reproduction of the relevant curves from Figure 5 for a side-by-side comparison. We can see that the progress measure is informative in the sense that it identifies the copy subcircuit (Steps 3v, 5) as quicker to learn than the subcircuit consisting of Steps 2, 3qkv, 4, and 5. However, when just looking at progress measures, both subcircuits appear to still emerge in a phase change, missing out on the crucial distinction between the two: the copy subcircuit (Steps 3v, 5) does not contain interacting components, whereas the other subcircuit (Steps 2, 3qkv, 4, and 5) does. These results show the benefit of the clamping approach for underlying exponentially forming subcircuits whose dynamics of formation causally affect the learning dynamics of the full network.

To conclude, we discuss pros and cons of progress measures versus our clamping approach. Progress measures have the benefit of only requiring access to network checkpoints after training, reducing the load of the mechanistic interpretability researcher. For cases of large language models, where training data is often not disclosed even for open-source models (Touvron et al., 2023), this could be especially appealing. Furthermore, as each progress measure only involves forward passes on checkpoints, they may be easier to iterate on. On the other hand, progress measures, as demonstrated above, provide largely a correlational understanding of subcircuits giving rise to dynamical phenomena. Clamping experiments, though more complex, provide a more causal understanding of learning dynamics. We hope that our results and open-source artificial optogenetics framework encourages more researchers to consider these types of experiments.

In terms of scaling to large language models, the hope would be that isolated subcircuits evolve quickly enough that the cost is low enough. Specifically, one can view clamping experiments as requiring about 3x the cost of progress measures per iteration (due to the forwards and backwards passes). Thus, if the number of iterations needed to train clamped networks is substantially lower than that of the full network, clamping experiments may actually be cheaper than progress measures.

[21] Specifically, we consider $\mathbf{b} = \pm\mathbf{b}^{true}, \mathbf{c} = \pm\mathbf{c}^{true}$ and pick the one that yields lower loss (this choice is made to avoid the scaling issue).

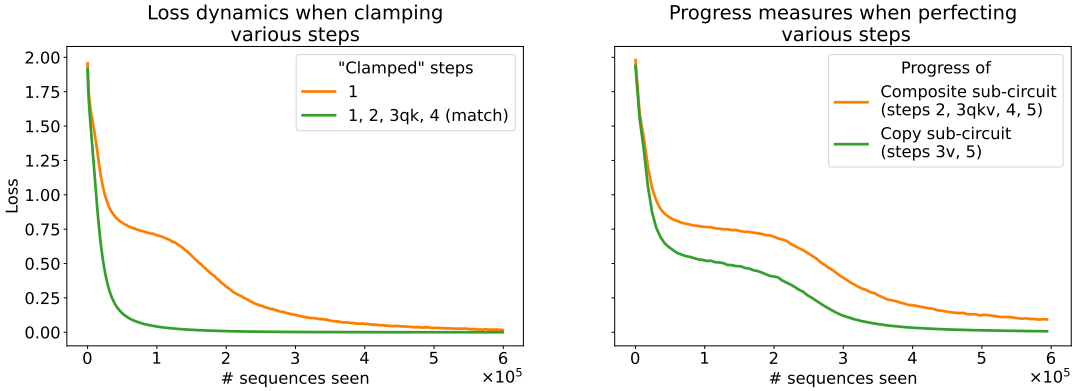

Figure 10. (Left) Our clamping method applied to studying formation of composite subcircuit consisting of Steps 2, 3qkv, 4, 5 (orange) and copy subcircuit consisting of Steps 3v, 5 (green). (Right) Progress measures tracking these same circuits. We can see that the clamping analysis more clearly shows the difference in formation dynamics of these two subcircuits, whereas the progress measure shows a phase change for both.

E. The role of Layer 1 heads

To establish the causal role of Layer 1 heads to individual induction heads in Layer 2, we did extensive ablation analyses.

Earlier, we found that Layer 1 heads do not substantially contribute to the output logits. To see if they only compose to Layer 2 heads' attention patterns (the match subcircuit), we consider a "pattern preserving" ablation (Olsson et al., 2022): the outputs of Layer 1 heads are completely ablated, but the attention patterns in Layer 2 are fixed to what they would've been had the Layer 1 heads not been ablated. We implement this using our artificial optogenetics framework by using two passes through the network. In the first, nothing is ablated. The Layer 2 attention patterns from this pass are then put into the `cache`. On the second pass through the network, Layer 1 outputs are ablated, and the `cache` is used to restore the patterns. We find that accuracy drops substantially (down to 55%), indicating substantial V-composition in our networks. To study V-composition, we introduce an analogous "value-preserving" ablation where instead of caching attention patterns, Layer 2 value vectors are cached. When using just a "value-preserving" ablation of all Layer 1 heads (without preserving patterns), accuracy drops substantially (down to 11%). These results indicate that Layer 1 heads are contributing to both the match subcircuit (patterns) and the copy subcircuit (values). Of the two, the role in the match circuit does seem to be "more important", which aligns with prior work (Olsson et al., 2022).

To verify the observed previous token (PT) heads, we next consider an ablation of these three heads with pattern-preserving (which should have no effect if they are PT heads) or value-preserving (which should have a huge effect, since patterns in Layer 2 will get messed up with these heads deleted). Likewise, we consider both types of ablations of the other heads (Layer 1 heads 0,3,4,6,7). Results are summarized in Table 1. We can clearly see that ablating heads 1,2,5 while preserving patterns has little effect, indicating these heads primarily compose with the match subcircuit. Likewise, we find the other heads primarily participate in V-composition for the copy subcircuit.

Table 1. Accuracy after Layer 1 head ablations of various sets of heads. All Layer 2 heads remain active.

Ablated heads	Type of ablation: Pattern-preserving	Value-preserving
Layer 1 Heads 1,2,5 (PT heads)	99.85%	69.87%
Layer 1 Heads 0,3,4,6,7 (rest)	34.27%	99.32%

Finally, we wish to establish the connectivity between Layer 1 and Layer 2 heads. To be specific to Layer 2 head, we repeat each experiment 8 times, once for each Layer 2 head active (and the others are ablated). As in Section 3.1, we first consider ablating a single previous token head. Results are shown in Figure 11a. We find that when ablating single heads, only Head 2 seems to have a strong effect. We believe this result is again suffering from the redundancy across heads, so the

importance of heads 1 and 5 is masked. To account for this, we instead consider ablating all but a given Layer 1 head. In this case, each square in the heatmap can be seen as one path through the network, with only one head in each layer used. Results are shown in Figure 11b. From these plots, it's clear that there are multiple active paths in the network:

1. Layer 1 Head 1 $\rightarrow$ Layer 2 Head 3
2. Layer 1 Head 2 $\rightarrow$ Layer 2 Heads 1, 2, 3
3. Layer 1 Head 5 $\rightarrow$ Layer 2 Heads 1 and 3

These results indicate a *many-to-many wiring diagram*.

We'd also like to take a moment here to emphasize the use of activation-level ablation analyses, as opposed to metrics calculated on weights. Composition score (Nanda, 2022; Elhage et al., 2021) is one such weight-based metric, calculated as:

$$\text{score}\left(L_1H_i \rightarrow L_2H_j : Q/K/V\right) = \frac{||W_O^{L_1H_i} W_{Q/K/V}^{L_2H_j}||_F}{||W_O^{L_1H_i}||_F ||W_{Q/K/V}^{L_2H_j}||_F}$$

It aims to measure the connection strength between heads in different layers by considering the matrices they use to write output and read input from the residual stream. In Figure 11c, we plot the composition scores of Layer 1 heads to key and value matrices of Layer 2 heads. Though the most noteworthy cases of composition are recovered (such as the importance of Layer 1 Head 2 for K-composition, and Layer 1 Head 6 for V-composition), overall we find the results to be much harder to interpret than those using our activation-level ablation analyses (Figure 11b).

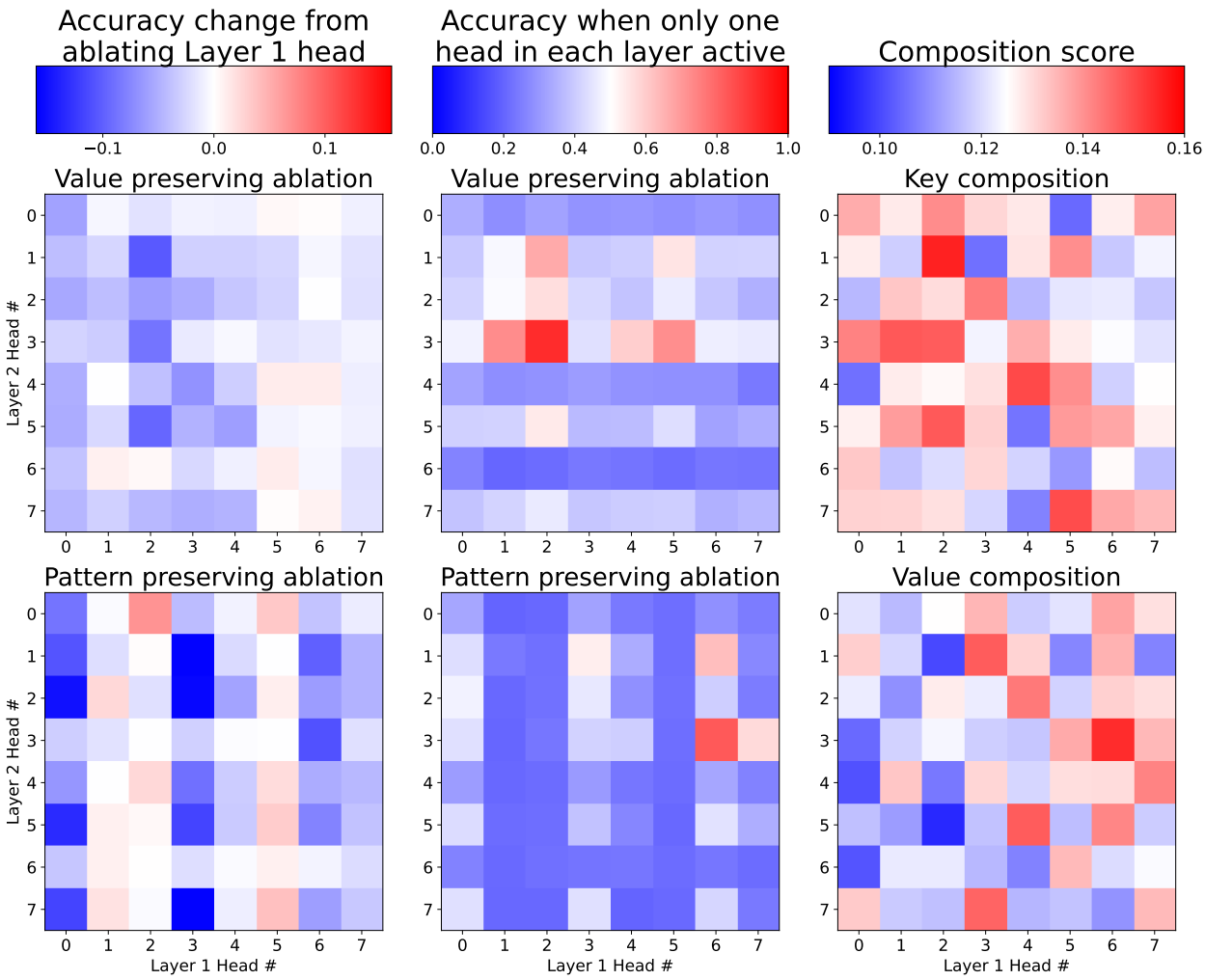

Figure 11. Layer 1 → Layer 2 head connectivity analyses (three different methods as the three columns). **Leftmost column:** Effect of ablating a single Layer 1 head. Each column corresponds to the Layer 2 head that is kept active. In this plot, we plot the deviation compared to if the given Layer 1 head was not ablated, to emphasize positive/negative differences (here darker blue would mean more influential). Left plot preserves values, so results will be most affected by Layer 1 heads that contribute to the match subcircuit (previous token heads). From this, we see that Layer 1 Head 2 is an important previous token head for many of the Layer 2 heads. The right plot preserves patterns, so results will be least affected by Layer 1 heads that only contribute to the match subcircuit (previous token heads). In fact, we see that when ablating heads 1, 2, and 5, performance improves, indicating that previous token heads might be interfering with the network in some cases due to (suboptimal) V-composition. **Middle column:** Performance of a single pair of heads (one from Layer 1, and one from Layer 2) when preserving Layer 2 values (left) or patterns (right). In this case, we plot the raw accuracy to show how strong each path through the network is. The color scheme highlights points that go above simple context copying strategies (which has an accuracy of 50%) in red. We see that Layer 1 Head 2 → Layer 2 Head 3 is the strongest path (when preserving values), leading to a performance of 90%. The left plot indicates that Layer 1 heads 1, 2, and 5 play a strong role in computing patterns. The right plot indicates that Layer 1 Head 6 has the most important V-composition to Layer 2 heads. **Rightmost column:** For comparison, we include the composition score (Nanda, 2022; Elhage et al., 2021) of Layer 1 head output projections to key and value input matrices in Layer 2. We again use a bwr colorscheme to highlight pairs with higher than average composition scores. While the main forms of composition are still visible (K-composition from Layer 1 Head 2 to Layer 2 Head 1/3, and V-composition from Layer 1 Head 6 to Layer 2 Head 3), we note that the results are overall way noisier. This suggests that the activation modification approach used in the middle column, made easy to implement by our artificial optogenetics framework, may provide clearer signals than weight-based approaches on circuit wiring.

F. Proof of saddle point in toy model

We include a brief proof that $\mathbf{a} = \mathbf{b} = \mathbf{c} = \mathbf{0}$ is a saddle point for the toy model, leading to the long loss plateau before the phase change.

Recall, the loss being minimized is:

$$\mathcal{L}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \frac{1}{2} \sum_{i,j,k} \left(a_i^{true} b_j^{true} c_k^{true} - a_i b_j c_k\right)^2,$$

and we assume $\mathbf{a}^{true}, \mathbf{b}^{true}, \mathbf{c}^{true} \neq \mathbf{0}$.

We then have:

$$\frac{\partial \mathcal{L}(\mathbf{a}, \mathbf{b}, \mathbf{c})}{\partial a_i} = -\sum_{j,k} \left(a_i^{true} b_j^{true} c_k^{true} - a_i b_j c_k\right) b_j c_k = 0,$$
$$\frac{\partial \mathcal{L}(\mathbf{a}, \mathbf{b}, \mathbf{c})}{\partial b_j} = -\sum_{i,k} \left(a_i^{true} b_j^{true} c_k^{true} - a_i b_j c_k\right) a_i c_k = 0,$$
$$\frac{\partial \mathcal{L}(\mathbf{a}, \mathbf{b}, \mathbf{c})}{\partial c_k} = -\sum_{i,j} \left(a_i^{true} b_j^{true} c_k^{true} - a_i b_j c_k\right) a_i b_j = 0,$$

if $\mathbf{a} = \mathbf{b} = \mathbf{c} = \mathbf{0}$. Moving forward, we will abbreviate derivations using the symmetry between a, b, c.

The value of the loss function at this point is:

$$\mathcal{L}(\mathbf{0}) = \frac{1}{2} \sum_{i,j,k} \left(a_i^{true} b_j^{true} c_k^{true}\right)^2$$

While we know this point is neither a global maximum (∞) nor a global minimum (0) of the loss function, we now need to show it's a saddle point (and not a local minimum/maximum).

F.1. Hessian test inconclusive for 3 or more interacting variables

A first attempt to show it's a saddle point (and not a local minimum/maximum) would be to use the determinant of the Hessian. We note that such a proof does work in the two vector case (the determinant of the Hessian is negative in that case), but for the 3-variable case, the determinant of the Hessian is also 0. We show this below:

The Hessian can be seen as a 3×3 block matrix:

$$\mathcal{H} = \begin{pmatrix} H_{aa} & H_{ab} & H_{ac} \\ H_{ba} & H_{bb} & H_{bc} \\ H_{ca} & H_{cb} & H_{cc} \end{pmatrix},$$

and we can work out the individual terms below:

$$\frac{\partial^2 \mathcal{L}(a, b, c)}{\partial a_i^2} = \sum_{j,k} (b_j c_k)^2 = 0$$
$$\frac{\partial^2 \mathcal{L}(a, b, c)}{\partial a_i \partial a_{i'}} = 0, i \neq i'$$

so $H_{aa} = H_{bb} = H_{cc} = \mathbf{0}$. For the cross terms, we have:

$$\frac{\partial^2 \mathcal{L}(a, b, c)}{\partial a_i \partial b_j} = \sum_k 2 a_i b_j c_k^2 - a_i^{true} b_j^{true} c_k^{true} c_k = 0$$

Here, the presence of c_k forces the these partials to 0, making the Hessian a matrix of all 0's. Note in the two variable case (if we remove $\mathbf{c}$), the Hessian would have positive off-diagonal elements and it becomes easy to show the determinant is negative.

F.2. Directions of positive and negative slope

Since the Hessian test doesn't work, we can instead attempt to identify two directions, one in which the function will increase in a small neighborhood and one in which it will decrease in a small neighborhood. These are easy to construct by picking i', j', k' s.t. $a_{i'}^{true} b_{i'}^{true} c_{i'}^{true} \neq 0$ (such indices are guaranteed to exist since $\mathbf{a}^{true}, \mathbf{b}^{true}, \mathbf{c}^{true} \neq \mathbf{0}$).

Specifically, consider the direction of $\mathbf{d}_- = \langle 0, ..., 0, a_{i'}^{true}, 0, ..., 0, b_{j'}^{true}, 0, ..., 0, c_{k'}^{true}, 0, ..., 0 \rangle$. We can see that a small $\varepsilon > 0$[22] perturbation along this direction will reduce the loss:

$$\begin{aligned}
\mathcal{L}(\mathbf{0} + \varepsilon \mathbf{d}_-) &= \frac{1}{2} \sum_{i,j,k} \left(a_i^{true} b_j^{true} c_k^{true} - a_i b_j c_k \right)^2 \\
&= \frac{1}{2} \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} - \varepsilon^3 a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 + \frac{1}{2} \sum_{i \neq i', j \neq j', k \neq k'} \left(a_i^{true} b_j^{true} c_k^{true} \right)^2 \\
&= \frac{1}{2} \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 (\varepsilon^6 - 2\varepsilon^3) + \frac{1}{2} \sum_{i,j,k} \left(a_i^{true} b_j^{true} c_k^{true} \right)^2 \\
\Rightarrow \lim_{\varepsilon \to 0} \mathcal{L}(\mathbf{0} + \varepsilon \mathbf{d}_-) - \mathcal{L}(\mathbf{0}) &= \frac{1}{2} \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 (\varepsilon^6 - 2\varepsilon^3) \approx -\varepsilon^3 \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 < 0
\end{aligned}$$

Thus, a direction with a negative directional derivative exists.

Similarly, we consider $\mathbf{d}_+ = \langle 0, ..., 0, -a_{i'}^{true}, 0, ..., 0, b_{j'}^{true}, 0, ..., 0, c_{k'}^{true}, 0, ..., 0 \rangle$. We can see that a small $\varepsilon > 0$ perturbation along this direction will increase the loss:

$$\begin{aligned}
\mathcal{L}(\mathbf{0} + \varepsilon \mathbf{d}_+) &= \frac{1}{2} \sum_{i,j,k} \left(a_i^{true} b_j^{true} c_k^{true} - a_i b_j c_k \right)^2 \\
&= \frac{1}{2} \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} + \varepsilon^3 a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 + \frac{1}{2} \sum_{i \neq i', j \neq j', k \neq k'} \left(a_i^{true} b_j^{true} c_k^{true} \right)^2 \\
&= \frac{1}{2} \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 (\varepsilon^6 + 2\varepsilon^3) + \frac{1}{2} \sum_{i,j,k} \left(a_i^{true} b_j^{true} c_k^{true} \right)^2 \\
\Rightarrow \lim_{\varepsilon \to 0} \mathcal{L}(\mathbf{0} + \varepsilon \mathbf{d}_+) - \mathcal{L}(\mathbf{0}) &= \frac{1}{2} \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 (\varepsilon^6 + 2\varepsilon^3) \approx \varepsilon^3 \left(a_{i'}^{true} b_{j'}^{true} c_{k'}^{true} \right)^2 > 0
\end{aligned}$$

Thus, we've identified two directions, one which increases the loss and one which decreases the loss, indicating that $\mathbf{a} = \mathbf{b} = \mathbf{c} = \mathbf{0}$ must be a saddle point. Note, if we find the directional derivative in either of these directions, it's still 0 (due to the factor of ε^3 not vanishing, even if we divide by ε to get the derivative)—we simply show the derivation to show the differing sign. We also note that this argument extends beyond 3 interacting vectors.

This argument can also give some intuition as to why the saddle point is "harder to escape" in the case of 3 interacting vectors (as evidenced by the longer loss plateau in the black curve versus the blue curve in Figure 5a). Here, we have a factor of ε^3 rather than ε^2 (in the two vector case). The intuition extends: if the number of components is increased further (beyond 3), the saddle point will get harder and harder to escape.

G. Assorted supplementary figures

While writing the main paper, we tried to maintain a coherent narrative that highlights the main results and intuitions we want readers to take away. For the in-the-trenches researcher, we provide some additional results that we thought were interesting.

[22] Note, the positivity assumption is not necessary. In fact, if we allow ε to take either sign, we can just use one direction to see the saddle point (due to the factor of ε^3). However, for the case of an even number of components (e.g., $N = 4$ instead of $N = 3$), we do need two separate directions (since ε^N will always be positive if N is even). To make this proof general, we thus provide the two directions and focus on positive ε.

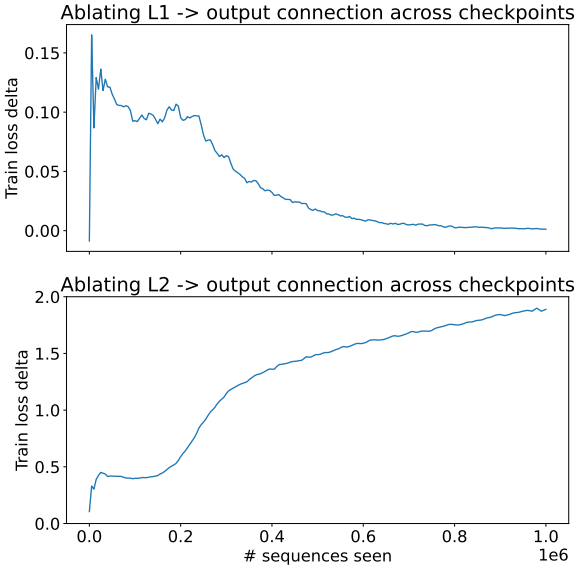

Figure 12. **a)** Change in loss after ablating connection from Layer 1 heads to output (unembedding) layer. Lower change indicates that Layer 1 head to output connections are less important. The ablation was applied independently to each checkpoint from the training run shown in Figure 3 and can thus be viewed as a progress measure. We can see that before the phase change, Layer 1 heads are contributing to the loss, but as induction heads form and the loss goes to 0 at the end of training, Layer 1 heads do not contribute to the output in a direct way. **b)** Same as a), but instead the connection from Layer 2 heads to output (unembedding) layer is ablated. Through training, importance of Layer 2 heads is increasing, especially after the phase change that corresponds to induction head formation.

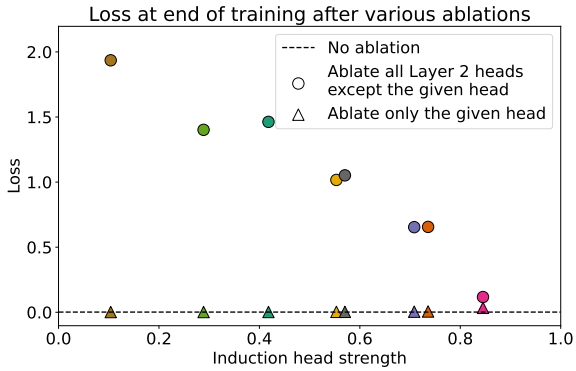

Figure 13. Same as Figure 4a, but plotting loss instead of accuracy. We include this plot for completeness, and used the accuracy version as we felt it was clearer for the main text.

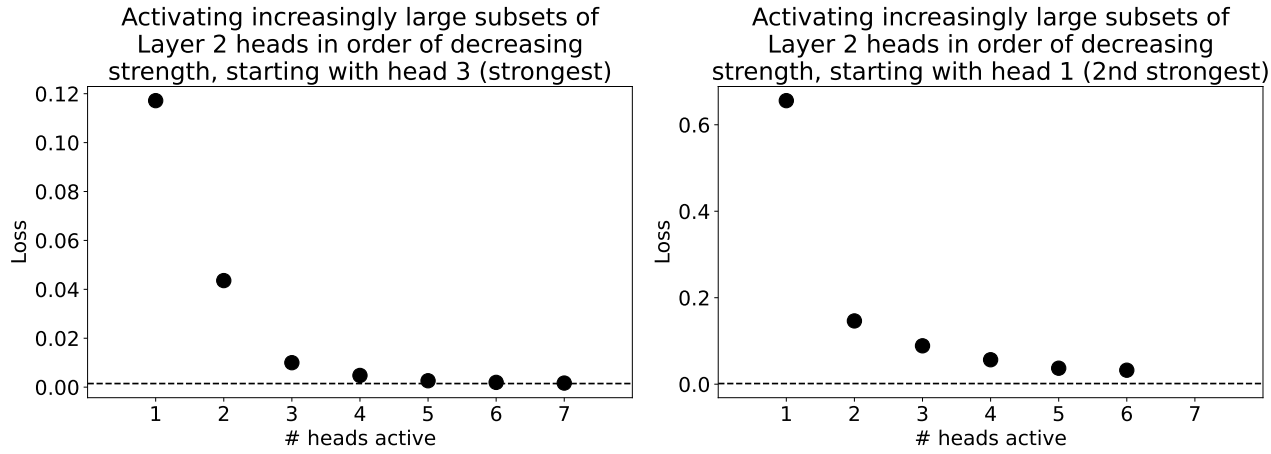

Figure 14. Another way of seeing the additive effect of induction heads. We first order the heads by strength (this happens to be `[3,1,2,7,5,0,4,6]` in our case, see Figure 3b). Then, we consider an ablation where only a certain number of heads (x-axis) are left active. In the left plot, we start with Head 3, so the points from left to right indicate [only Head 3 active], [Heads 3 and 1 active], [Heads 3 and 1 and 2 active], and so on. The right plot is the same, except we never activate Head 3 (the strongest head). The monotonically decreasing loss in both plots indicate that heads have an additive effect on each other. Dotted line indicates loss of full network (no ablations). Note: the y-axis on the right is scaled larger since Head 3 (the strongest head) is never active (so losses are higher).

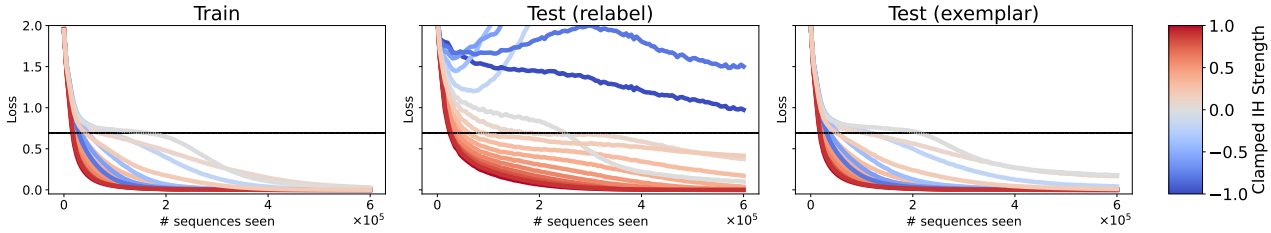

Figure 15. Clamping through training with an imperfect match circuit. For this figure, we clamped the attention pattern of a single Layer 2 head to look like a noisy induction head (the other heads were left active) to see how "strong" an induction head needs to be before it induces the exponential learning dynamics seen when clamping to maximum strength (green line, Figure 5). For example, when "Clamped IH strength" is 0.4, this corresponds to attention of 0.7 to the correct label token and 0.3 to the incorrect label token, and 0 to all other tokens. We vary the noisiness in the induction head and see how this effects dynamics. The darkest red curve (perfect match) corresponds to just learning a copy subcircuit (as in Figure 5, green line), exhibiting a smooth exponential decay pattern. As the strength is turned down to 0, we see the learning dynamics shift from exponential to slow exponential to the usual phase change when the IH strength is clamped to 0 (as this head is ignored, and the network learns using the other Layer 2 heads). When we fix the induction strength to -1, meaning the head is *only* paying attention to the wrong token, we recover an exponential curve. This is an example of the network using an "anti-induction" head (a phenomenon we sometimes observed if $L = 2$): the mechanism here is the network is *copying* both labels from context and then *subtracting out* the wrong label. This is an example of a slightly different (and perhaps more complicated) circuit for implementing ICL in our task, similar to how (Zhong et al., 2023) discovered two different circuits for modular addition. We hypothesize that the reason this "anti-induction" circuit doesn't emerge naturally when $L > 2$ is because imperfect versions of "anti-induction" heads (e.g., "Clamped IH strength" $= -0.8$) are less robust to different label pairs (middle plot, light-dark blue curves), which may make it a circuit element that is feasible but not learnable (under this task setup).

H. The "residual heads" hypothesis

In Section 3.1, we provide evidence that induction heads function additively with redundancy across heads. Combining this insight with the ordering of head emergence in Figure 3b, we hypothesize that other induction heads form to minimize the residual of the loss left after earlier heads form (e.g. Head 3). We didn't include this hypothesis in the main paper as it's not the focus of our work, and we find evidence supporting and opposing it. As scientists, we hope these mixed results can serve as a launch point for further study, and thus choose to include them here.

One simple way to test this hypothesis would be to consider the mistakes made by the individual Layer 2 heads, when all other Layer 2 heads are turned off. Let's consider Head 2 and Head 3 for example. Head 3 on its own achieves an accuracy of 98%, outputting the wrong answer for 1553 of the 78400 training sequences. Head 2's accuracy is 75%. If Head 2 is fitting to the residual, we might expect Head 2's accuracy on the sequences that Head 3 fails to solve to be above its baseline of 75%, meaning it's preferentially stronger on the problems where Head 3 is mistaken. On the other hand, it could be that the sequences missed by Head 3 are simply "harder", in which case we would expect an accuracy less than its baseline. In Table 2, we find that the latter is more likely to be the case, so this piece of evidence is *against* the residual heads hypothesis.

Table 2. Accuracy when ablating all but a given Layer 2 head on the subset of problems that are incorrect when ablating all but Layer 2 Head 3 (the one with highest induction score). Second row contains the accuracies over the whole dataset for comparison (which matches the circles in Figure 4a).

Accuracy	Head 0	Head 1	Head 2	Head 4	Head 5	Head 6	Head 7
On the subset	24.08%	64.78%	71.67%	32.97%	56.41%	16.61%	46.10%
Overall	40.32%	76.51%	75.12%	42.85%	64.65%	29.16%	58.85%

Next, we present some evidence that supports the hypothesis. Namely, we consider perfecting each induction head's match circuit (by setting its attention pattern to the perfect pattern that gives attention 1 to the correct token and 0 to all other tokens), which allows us to quantify the quality of its copy circuit. Then, we can examine the mistake rate of the copy circuit for different labels, and see these values across heads. Our results are summarized in Figure 16. Specifically, we do find some specialization of output classes: Head 3 has a higher mistake rate for copying labels 1 and 3, while heads 1 and 5 have virtually 0 mistake rate on these, indicating that they may have specialized to these examples. This result *supports* the "residual heads" hypothesis.

If we repeat the analysis of Table 2, we find some promising evidence in this "perfected match" condition (Table 3). These results suggest that, when it comes to the copying subcircuit, induction heads forming later in training preferentially focus on copying labels that earlier formed induction heads are worse at. As a result, these earlier heads do not have to "fix their mistakes" as much as if they were trained on their own, connecting to our results in Section 3.2. This result is *inconclusive* with respect to the "residual heads" hypothesis, as it only pertains to Head 1, and not the other heads.

Finally, we build off the result in Figure 4b, and try to take the argument further. Given that Head 3 is the quickest to learn and does in fact emerge first in the full network (Figure 3b), we consider training networks where all but *two* heads are ablated: Head 3 and each of the other 7 heads (one at a time). Loss curves are shown in Figure 17. By training with Head 3 and just one other head, we see loss curves are way closer to the actual training curve. However, the ordering of phase changes still does not match the ordering of emergence and strengths in Figure 3b, perhaps due to higher order interactions (e.g., between three heads or more). Given the current results though, this piece of evidence is *against* the "residual heads" hypothesis.

Given all these pieces of evidence, for and against, we cannot convincingly argue for or against the "residual heads" hypothesis. We hope this investigation (including negative results) can inform future work researching this hypothesis.

I. Reproducibility across model initializations

Our code, as well as notebooks reproducing findings on multiple initialization seeds (5 seeds) can be found at `https://github.com/aadityasingh/icl-dynamics`. We found little to no differences when training seed (responsible for data ordering) was varied. Here, we summarize qualitatively the results that differ slightly across init seeds and the ones

Table 3. Same as Table 2, except we also "perfect" each induction head's match circuit (as done in Figure 16), so we can view this table as only considering the mistakes in each induction head's copy circuit. There are only 58 examples that the "perfected match" layer 2 Head 3 gets wrong, so this table only considers the accuracy of other "perfected match" heads on these 58 examples. Note the overall accuracies in this Table match the squares in Figure 16.

Accuracy	Head 0	**Head 1**	Head 2	Head 4	Head 5	Head 6	Head 7
On the subset	24.14%	**86.21%**	20.69%	22.41%	50.00%	0.00%	44.83%
Overall	50.01%	82.80%	82.73%	62.10%	65.42%	28.46%	70.70%

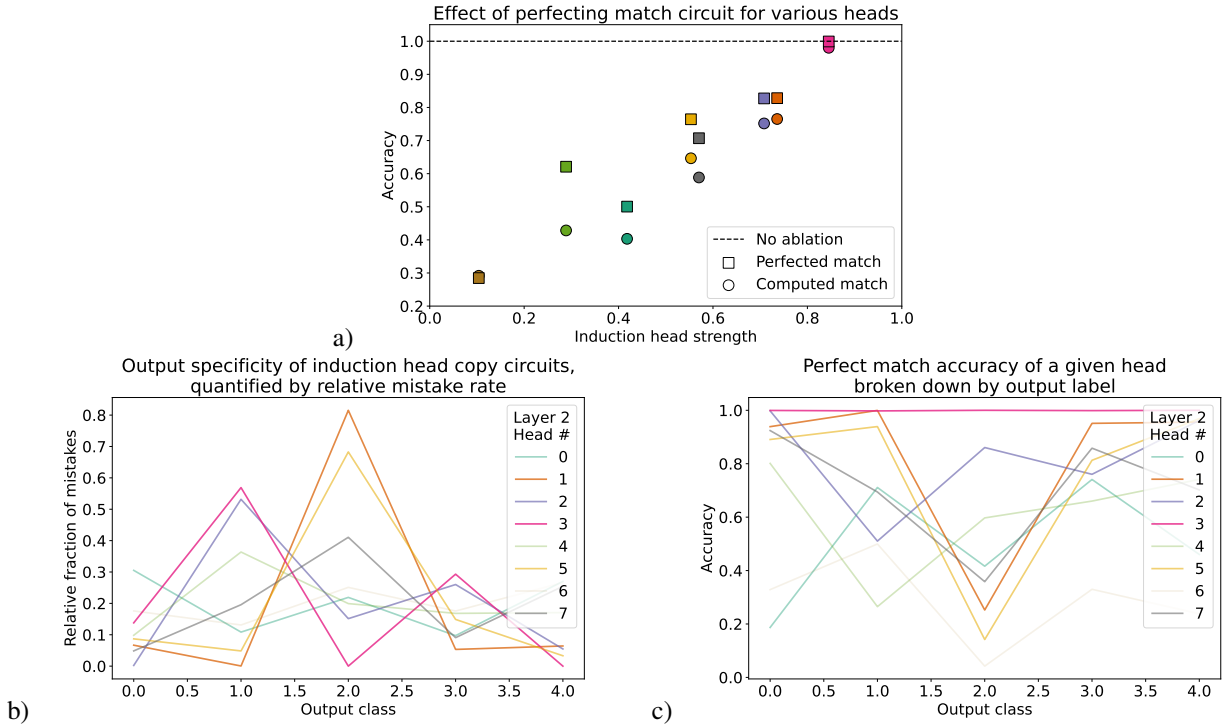

Figure 16. **a)** Induction heads are imperfect copiers. We consider the network from the end of training in Figure 3 and perform the ablate-all-but-head-X ablation used for the circles in Figure 4. We then use our artificial optogenetics framework to "perfect" each head's attention pattern (so it would have an induction strength of 1). While this increases the accuracy for most heads, the accuracy is still far from perfect, indicating that inducion heads at the end of training are imperfect copiers. **b)** We dive deeper and find label dependence in the imperfect copying. Specifically, here we plot the relative mistake rate by output label for each head. Line opacity is equal to induction head strength, to emphasize the more prominent induction heads. Interestingly, Head 3 is relatively worse at copying labels 1 and 3, but heads 1 and 5 compensate for this. Notably, in Figure 3, heads 1 and 5 appear after Head 3. We postulate that heads 1 and 5 are thus compensating for the mistakes that Head 3 makes. Future work could further explore such a "residual heads" hypothesis in more depth. **c)** For completeness, we include accuracy vs output class. We note that Head 3 with a "perfect" attention pattern overall does make very few mistakes, as we'd expect given its high accuracy in a). Notably, what plot b shows is that Head 3 with a "perfect" attention pattern, when it does make mistakes, tends to do so on classes 1 and 3.

that don't (rather than providing $4 \cdot 6 = 30$ new figures).

Figure 18 summarizes a reproduction of some of the results of Section 3.1. Specifically, strongest heads (determined via induction strength) are mostly able to solve the task. We find that the connection to the *neural race* (Saxe et al., 2022) mostly holds across seeds (first discussed in Section 3.2, though the correlation is not as strong as other metrics related to induction strength. Our main results on sub-circuit discovery (Figure 5) hold across seeds. The scaling result (Figure 6) reproduces across most seeds, though sometimes in standard training we see learning with 15 labels to be as quick as learning with 10 labels. For further details, we refer readers to the notebooks in our codebase.

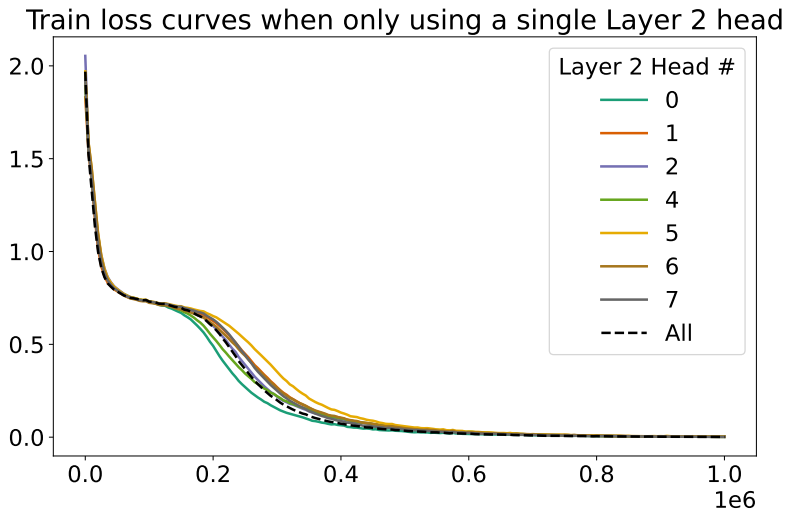

Figure 17. A follow-up to Figure 4b where we consider training with only a pair of Layer 2 heads active: Head 3 and one other head. Interestingly, loss curves snap to almost the true training loss curve. The relative ordering of phase changes is not indicative of the order of emergence (see Figure 3).

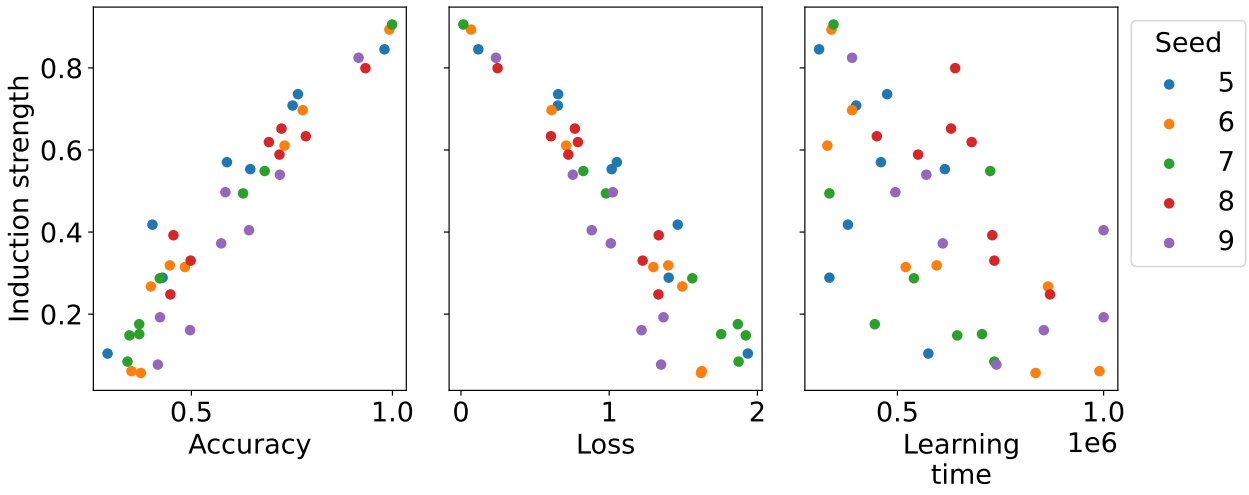

Figure 18. Reproducing some of the results of Section 3.1 across seeds. Specifically, we plot the induction strength of all 8 L2 heads across 5 seeds (40 points) versus various other metrics on individual heads: accuracy and loss if all other L2 heads are ablated (analogous to circles in Figure 4a,13), as well as the "learning time", which we define as the number of sequences seen such that the loss drops below $0.4 \cdot \log 2$ when training with a single head (a summary statistic for Figure 4b). As we can see, induction head strength is well correlated with accuracy and loss (lower loss is better, and corresponds to stronger heads) across seeds. We see that learning time (and thus, connections to the neural race reduction (Saxe et al., 2022)) is correlated to strength across seeds (faster learning times correlate to stronger heads), but not as strongly. Future work could investigate these ideas of circuits "racing" to minimize the loss further.