How Depth Growth and Looping Shape In-Context Learning in Language Models

Anonymous authors
Paper under double-blind review

Abstract

Looped and depth-grown language models reuse computations across depth and have attracted interest for their reasoning capabilities. Building on prior suggestive evidence of improved in-context-learning (ICL), we investigate how such reuse favors ICL, including how it shapes the acquisition, retention, and mechanisms of this ability. Our analysis spans pretrained language models up to 3B parameters and controlled tasks. The experiments provide evidence for a shared inductive bias toward learning from context: grown and looped models rely more on information in context and, compared with a full-depth dense reference, acquire ICL earlier in controlled tasks and retain it more strongly when learning fixed associations in the weights is also possible. Tracing attention heads through training reveals that depth growth propagates early induction heads across layers. Ablations show that grown and looped models rely more strongly on induction-ranked heads for both few-shot performance and later-token prediction. Together, these findings provide evidence for a shared inductive bias toward learning from context and suggest that (soft) parameter reuse shapes how this ability develops and which attention mechanisms support it.

1 Introduction

How a language model acquires its depth can shape the computations it learns. *Depth growing* trains an initially shallow network and progressively duplicates blocks of layers, allowing later training stages to reuse computations learned earlier. *Looping* instead repeatedly applies a shared block within each forward pass. Originally motivated by training efficiency (Fahlman & Lebiere, 1989) and parameter efficiency, respectively, both approaches have attracted renewed interest for their reasoning capabilities (Saunshi et al., 2024; 2025; Geiping et al., 2025). Although grown layers become independent after duplication, whereas looped layers remain tied, the two approaches can produce related patterns of iterative computation across depth (Kapl et al., 2026a;b).

This reuse of computation may also favor *in-context learning* (ICL), as suggested by initial evidence in both grown and looped models (Kapl et al., 2026b). ICL is the ability to use information supplied in a prompt or sequence without updating the model's parameters (Brown et al., 2020). A model that repeatedly refines its representations might also be better equipped to extract and apply patterns from context. Yet, improved reasoning performance alone does not establish this ability. Predictions can rely on associations stored in the weights, on information inferred from the current sequence, or on both. Moreover, a model's improvement in predicting later tokens and its ability to learn a task from demonstrations are related but distinct measurements, with potentially different underlying mechanisms (Olsson et al., 2022; Yin & Steinhardt, 2025).

Prior work provides promising evidence for a connection between depth reuse and ICL. Looped Transformers can learn synthetic regression and other data-fitting tasks with substantially fewer parameters than standard Transformers (Yang et al., 2024; Von Oswald et al., 2023). Studies of pretrained language models also report improved adaptation from demonstrations in grown and looped models for synthetic reasoning tasks (Kapl et al., 2026b). Concurrently, Wang et al. (2026) report stronger gains with increasing context length and demonstration count in compute-matched looped mixture-of-experts models. What remains less understood is how these behavioral differences arise during training: whether growing and looping favor a context-based solution when memorization

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

is also possible, how they affect the persistence of that solution, and how changes in attention-head mechanisms relate to the different measures of ICL.

In this work, we investigate these questions through complementary experiments on natural language and controlled tasks. We compare dense baselines with depth-grown and looped models on language-model configurations with dense reference sizes from 360M to 3B parameters. We measure position-dependent token loss and scaling with the number of demonstrations for few-shot learning. We then use RegBench (Akyürek et al., 2024) to study learning previously unseen formal languages, and Omniglot-based tasks (Chan et al., 2022; Singh et al., 2023) to separate learning from context from learning fixed associations in the weights. Finally, we track induction and function-vector heads through training and intervene on them in the final models. Our contributions are the following:

- **Grown and looped language models benefit more from context.** In the main 3B language-model comparison, both achieve larger early-to-late token-loss reductions than the dense baseline, with the grown model also attaining lower absolute late-token loss. Both models also achieve higher GSM8K accuracy across all demonstration counts. On RegBench, grown and looped models predict unseen formal languages more accurately, with their advantage emerging within the first string (Figures 1 and 2).
- **Growing and looping favor earlier acquisition and stronger retention of ICL.** In controlled Omniglot tasks where both context and fixed class–label associations can solve training queries, grown and looped models retain higher ICL accuracy and exhibit lower in-weights learning (IWL) accuracy than the full-depth dense reference. When labels are reassigned in every sequence, removing a fixed class–label solution, both acquire ICL earlier than the same reference (Figure 3).
- **Depth growth propagates early induction heads across layers.** Tracing final induction heads backward through training shows that strong heads in the grown model predominantly descend from heads already exhibiting strong induction behavior in early growing stages. Layer duplication produces repeated patterns of this behavior across depth, while looped models exhibit analogous patterns across successive applications of their shared looped block (Figures 4 and 5).
- **Grown and looped models rely more strongly on induction heads for ICL.** In the main language-model comparison, ablating induction-ranked heads causes larger drops in few-shot performance and larger increases in late-token loss than in the dense baseline. These interventions establish a greater contribution from induction-ranked heads to both behaviors, while function-vector heads remain important (Figure 6).

Together, these experiments show that constructing depth through (soft) parameter reuse shapes both the development of context use and the attention mechanisms supporting it.

2 Related Work

Grown Models. Model growth reduces training cost by starting with a smaller network (Fahlman & Lebiere, 1989). Language-model approaches use layer copying and progressive stacking (Gong et al., 2019; Reddi et al., 2023; Kim et al., 2024; Du et al., 2024), learned transformations between model sizes (Wang et al., 2023), or masked and function-preserving expansions (Yao et al., 2024; Chen et al., 2015; Gesmundo & Maile, 2023). Growth operators also shape model capabilities: `MIDAS` duplicates a central block of layers and improves reasoning at comparable language-model capabilities (Saunshi et al., 2024). Kapl et al. (2026a) find stronger reliance on later layers in grown models and propose `LIDAS`, which duplicates layers around the network's center independently of existing block boundaries.

Looped Models. Cross-layer parameter sharing provides computational depth with fewer weights, as in Universal Transformers (Dehghani et al., 2019), recurrent stacking for translation (Dabre & Fujita, 2019), and ALBERT (Lan et al., 2020), while Takase & Kiyono (2023) study its design choices. Extensions add layer-specific low-rank adapters (Bae et al., 2025) or mixture-of-experts capacity (Csordás et al., 2024). Recurrence has also been studied for algorithmic length generalization (Fan

et al., 2025), latent reasoning (Saunshi et al., 2025), and inference-time computation scaling (Geiping et al., 2025; Zhu et al., 2025). Other work retrofits recurrence into pretrained models (McLeish et al., 2025; Koishekenov et al., 2025) or explores hierarchical and compact recursive architectures for iterative refinement (Wang et al., 2025; Jolicoeur-Martineau, 2025).

In-Context Learning. ICL enables adaptation from demonstrations without parameter updates (Brown et al., 2020). Studies show that Transformers infer unseen functions from examples (Garg et al., 2022), learn gradient-descent-like computations in restricted regression settings (Von Oswald et al., 2023; Gatmiry et al., 2024), and can be programmed to execute learning algorithms in context (Giannou et al., 2023). Data properties such as burstiness and dynamic meanings encourage ICL (Chan et al., 2022), which can decline as persistent associations are learned in the weights (Singh et al., 2023). Other work connects regular-language learning to in-context n-gram heads on RegBench (Akyürek et al., 2024) and studies context-dependent representations and their competition with pretrained semantics through graph tracing (Park et al., 2025). These settings let us probe context use with and without competing in-weights solutions.

Induction heads support pattern copying and are associated with improved later-token prediction (Elhage et al., 2021; Olsson et al., 2022). Their formation has been studied in controlled synthetic tasks (Singh et al., 2024) and language-model pretraining (Aoyama et al., 2026). Function-vector heads transmit task representations from demonstrations (Todd et al., 2024). Yin & Steinhardt (2025) find that few-shot performance depends primarily on these heads, many of which first exhibit induction behavior.

Looped Transformers can match standard Transformers on synthetic data-fitting tasks with substantially fewer parameters (Yang et al., 2024). Concurrent work (Wang et al., 2026) reports gains that increase with sequence length and demonstration count in looped mixture-of-experts language models under matched compute, parameter, and KV-cache budgets .

Most closely related, Kapl et al. (2026b) identify shared depth-wise computational patterns in grown and looped models, show that grown models can benefit from inference-time repetition, and report improved adaptation from demonstrations on reasoning primitives. We extend this evidence to ICL acquisition, retention, and mechanisms across pretrained language models and controlled tasks. We test whether growing and looping favor context use when memorization is possible, trace induction heads through layer duplication, and ablate induction and function-vector heads to assess their contributions to few-shot learning and token prediction.

3 In-Context Learning of Depth-Grown and Looped Models

We first examine how growing and looping affect learning from context. After introducing the models and measures (Section 3.1), we evaluate context use in language models (Section 3.2), adaptation to unseen formal languages on RegBench (Section 3.3), and ICL with and without an in-weights solution on Omniglot (Section 3.4).

3.1 Models and Evaluation Measures

A dense model trains L independently parameterized Transformer layers from scratch. A depth-grown model instead begins with a shallower model and periodically duplicates a block of layers, after which the copies train independently. The main paper focuses on `LIDAS` (Kapl et al., 2026a), which selects the central layers without respecting earlier block boundaries. Comparisons with `MIDAS` (Saunshi et al., 2024) and at other model sizes (360M and 1.7B) appear in Appendix A.3. The language models start with block size $b = 4$ layers and add b layers at each growth event, copying both weights and optimizer moments while retaining the training schedule.

A looped model reuses the same block of layers several times within each forward pass, with parameters remaining tied throughout training. We write Loop p-$b{\times}r$-c for p prelude layers, a block of b layers applied r times, and c coda layers, borrowing the naming conventions by Geiping et al. (2025). Its effective depth is $p + br + c$, with $p + b + c$ unique layers. Our main 3B comparison uses a 36-layer dense reference (Bakouch et al., 2025), a `LIDAS`-grown model reaching that depth, and the Loop 10-8×2-10 model, which has 36 layer applications and 28 unique layers. All three models are trained on the same 100B tokens in the same order. Within each size, all models share the

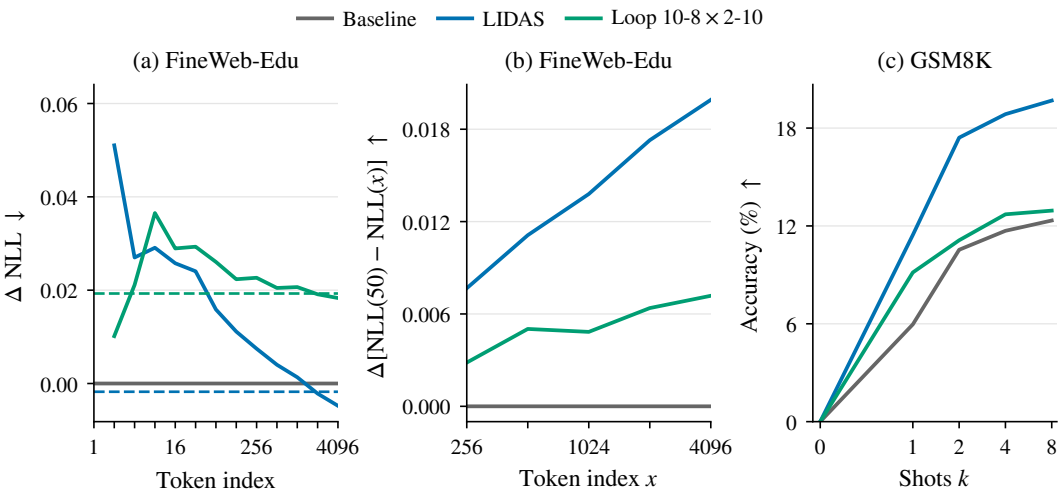


Figure 1: Grown and looped models benefit more from context. In the 3B comparison, grown and looped models show greater context gains in (a) NLL relative to the dense baseline and (b) early-to-late loss reduction relative to the baseline per token position, although the looped model retains higher NLL. Both panels aggregate token losses in the same power-of-two position bins: (a) compares the binned loss directly with the baseline, whereas (b) first computes each model's improvement from a fixed early token position (50) to the bin ending at x and then subtracts the baseline's improvement. Dashed horizontal lines show mean ΔNLL over all positions. (c) Both models exceed the baseline's few-shot GSM8K (Cobbe et al., 2021) accuracy, with larger gains for LIDAS. Results use 10,000 FineWeb-Edu documents and three demonstration seeds, respectively. Additional model sizes and exact experimental setup in Appendix B.

data and optimizer-step budget and have matched final inference FLOPs. Growing uses shallower networks during part of training, saving an estimated 26% of training FLOPs at 3B and 22% at both 360M and 1.7B. Looping reduces unique parameters while retaining the same effective depth. Appendix A reports the configurations and training schedules.

In-Context Learning Measures. We evaluate language models using next-token negative log-likelihood (NLL, lower is better), a token-loss ICL-Score (Olsson et al., 2022), and few-shot task performance. The ICL-Score measures the reduction in mean NLL from early positions 40–60 ("50") to later positions 490–510 ("500"), with higher values indicating greater early-to-late improvement. We use these token-loss windows to smooth potential single token position spikes and refer to the positions in the following by a single position for simplicity. In Figure 1(b), we instead average later losses over power-of-two bins ending at x, keeping the early reference fixed. This measures improvement within each model, while we report individual token loss separately. Few-shot performance is measured through benchmark accuracy or token-overlap F1. Full protocols are given in Appendix B.

3.2 Context Use in Language Models

We first compare how the different models use additional context in sequence modeling and few-shot tasks. These evaluations distinguish improvement with more context from absolute performance relative to the dense baseline. For sequence modeling, the depth-grown and looped models are worse at predicting early tokens compared to the baseline (Figure 1(a)), with the biggest gap for very early tokens (token index ≤ 8). After this peak, both models improve the prediction of tokens by lowering their NLL more, making better use of more tokens in the context. This is especially apparent when measuring the improvement of later tokens versus a fixed early position (50) and comparing it to the improvement of the baseline (Figure 1(b)). The looped model, which uses significantly fewer unique parameters, retains higher NLL than the baseline even at the end of the sequence. LIDAS matches the baseline around position 1000 and achieves lower NLL thereafter (solid line) and on average across positions (dashed line).

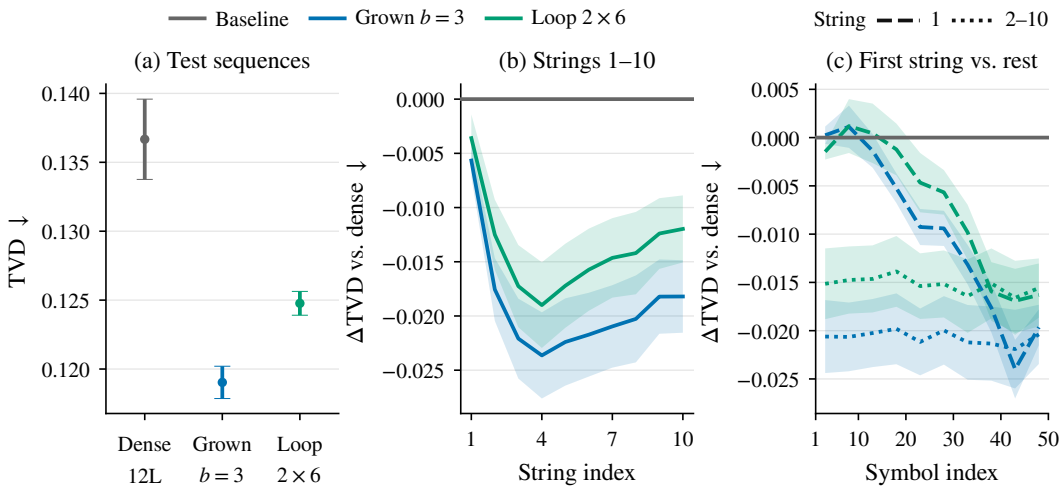


Figure 2: Grown and looped models predict unseen formal languages more accurately. In-context language learning on RegBench (Akyürek et al., 2024). (a) Test total variation distance (TVD) for the dense 12-layer, Loop $2{\times}6$, and Grown $b = 3$ model (mean $\pm$ standard error over four training seeds). (b) TVD relative to the dense model against in-context string index. (c) The same comparison by symbol index, separately for the first string and strings 2–10. Lower is better for all panels. Bands show standard errors. The looped model has width 256, versus 128 for dense and grown models: hyperparameters are selected on validation data, rather than matched in width. Panel (a) uses the full test set. Panels (b,c) retain only string indices present in every test sequence (strings 1–10). Experimental setup and full details in Appendix C.

On GSM8K (Cobbe et al., 2021) few-shot accuracy in Figure 1(c), the looped model, again using fewer parameters, keeps its advantage over the baseline across demonstration counts, but the baseline almost closes the gap at eight shots. The depth-grown model instead widens its performance improvement gap with additional examples, ending up around $+7\%$ above the baseline. GSM8K evaluates final answers extracted from generated solutions to new questions, so these gains reflect improved answer generation from in-context demonstrations, a setting relevant to practical few-shot prompting. As an additional comparison, Appendix Figure 9 reports separate averages over five multiple-choice and three generation tasks. At 3B, `LIDAS` improves on the baseline in both subsets, while the looped model's advantage is concentrated in generation.

3.3 In-Context Learning Languages: RegBench Dataset

RegBench (Akyürek et al., 2024) tests adaptation to an unseen formal language. Each input concatenates strings sampled from one deterministic finite automaton (DFA), and test automata are disjoint from training. We measure total variation distance (TVD) between the predicted next-symbol distribution and the uniform distribution over legal continuations. All models apply twelve layers per forward pass, with two unique layers for the looped model.

In Figure 2(a), the grown and looped models achieve lower TVD than the dense baseline on novel languages. Their advantage increases over the first few strings, peaking around the fourth (Figure 2(b)). When zooming in on the test sequences in Figure 2(c), both alternative training variants adapt faster to the new language by predicting symbols (akin to characters or tokens) within the first string better after roughly 15 symbols. Additionally, on average, after the first string the grown and looped model do not continue to increase their advantage with more symbols but keep a constant improvement. Appendix Figure 12 extends the comparison including DFA accuracy and additional grown models also evaluated with four training seeds, with the precise setup in Appendix C.

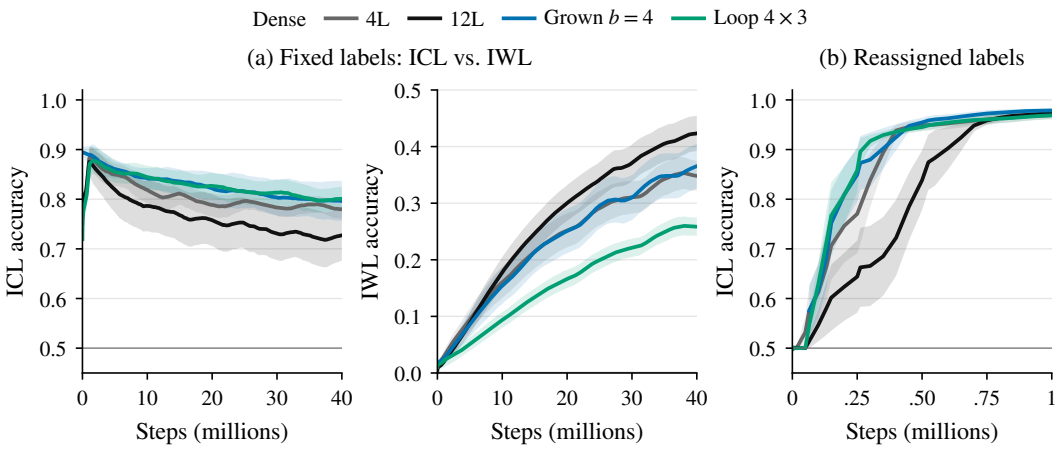


Figure 3: Growing and looping favor stronger retention of ICL, at the cost of in-weights learning (IWL), and earlier acquisition of ICL when IWL is not possible. Solution preference and emergence of in-context learning on Omniglot-based tasks (Singh et al., 2023). (a) Training uses fixed class labels and $P(\text{bursty}) = 1$, so both context and in-weights associations can solve each training query. ICL is measured on held-out classes with labels reassigned per sequence. IWL is measured on training classes whose query class is absent from context. (b) Training follows the ICL probe setup using images from training classes, with labels reassigned in every sequence to remove a fixed class–label solution. Curves show mean accuracy, with bands indicating standard error across four (a) and eight (b) smoothed seed trajectories. Growth occurs at 50k and 100k steps. Experimental details in Appendix D.

3.4 In-Context and In-Weights Learning: Omniglot

We next examine learning from context when fixed associations can also be stored in the weights, or when labels vary between sequences, removing the in-weights learning (IWL) solution. For that purpose, we use the Omniglot image-classification setup of Chan et al. (2022). Each sequence contains eight image–label pairs followed by a query image. In the fixed-label setting, every training class has a persistent label and the query class always appears among the context examples, with $P(\text{bursty}) = 1$, meaning that every training sequence contains multiple demonstrations of the same classes. The training query can therefore be answered either by matching it to the current examples in context or by learning its class–label association in the weights. We distinguish these capabilities with two probes: ICL accuracy on held-out classes with labels reassigned in each sequence, and IWL accuracy on training classes whose query class is absent from context. We compare dense models with 4 and 12 layers, a Loop 4×3 model, and a `LIDAS`-grown model that expands from 4 to 8 layers at 50k steps and to 12 layers at 100k steps. All models have the same hidden width and use the same optimizer and batch size. We average over four training seeds for the fixed-label setting and eight for the ICL-only setting below. Appendix D gives the full setup.

Figure 3(a) shows an early rise in ICL followed by a gradual decline as IWL improves, reflecting the transience of ICL observed by Singh et al. (2023). Over the shared training horizon, the grown and looped models retain higher mean ICL accuracy but lower IWL accuracy than the dense 12-layer model. The grown model has similar ICL and IWL accuracies to its four-layer reference, from which it first grows at 50k steps. The looped model attains similar ICL accuracy to the grown model but significantly lower IWL accuracy. These trajectories are consistent with growing and looping favoring learning from context relative to the full-depth dense model at the same effective depth. For looped models, this can come at the cost of weaker in-weights learning, while the grown model appears to inherit both capabilities from its shallow starting model.

To test acquisition when a fixed class–label solution is unavailable (ICL-only), we train a different set of models with labels 0 and 1 reassigned independently in every sequence, mirroring the ICL probing setup on training images. Each sequence contains two classes with four examples each, followed by a query from one of them. In Figure 3(b), ICL accuracy emerges earlier for the looped, grown, and four-layer dense models than for the dense 12-layer model. By 1M steps, all four tested

configurations almost reach 100%. Thus, this setting does not differentiate how much ICL is attained but rather the timing and consistency of ICL emergence, given that ICL is the only correct solution to the training problem.

4 Growing and Looping Increase Reliance on Induction Heads

The behavioral results suggest that growing and looping favor learning from context. Returning to language models, we examine the attention mechanisms supporting this behavior. We first track induction behavior during training and across layers (Section 4.1), then ablate heads to test their contribution to context use (Section 4.2).

4.1 Depth Growth Propagates Early Induction Heads

Attention-Head Measures. We characterize attention heads using prefix-matching (PS) and function-vector (FV) scores. PS measures attention to the token following an earlier occurrence in repeated random-token sequences (Olsson et al., 2022). FV scores measure how much replacing a head's output with its clean task mean restores correct-answer prediction when demonstration labels are shuffled (Todd et al., 2024). We use these scores to select heads for ablation. Full evaluation protocols are given in Appendix E.1. We first compare induction behavior during language-model pretraining by averaging PS over the top 2% of heads at each checkpoint, reselecting them as training progresses. Checkpoint and aggregation details are given in Appendices A.3 and E.1.

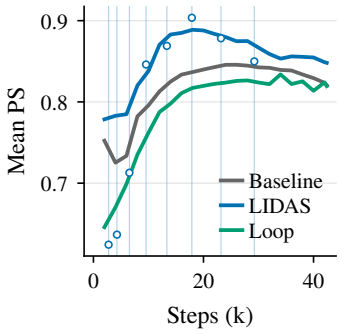


Figure 4: Induction behavior during training. Top-2% mean PS (3B), with heads reselected at each checkpoint. Vertical lines mark growth. Hollow circles show immediate post-growth scores.

In Figure 4, LIDAS develops higher PS than the dense baseline and finishes training with a higher score, while the looped model finishes close to the baseline. Early growth events temporarily disrupt prefix matching, followed by quick recovery during further training. These aggregate scores motivate tracing the heads responsible for the grown model's strong induction behavior. Full trajectories of PS, function-vector scores, few-shot performance, and the token-loss ICL-Score appear in Appendix Figure 13.

We next follow the twelve final top-2% PS heads in each 3B model through training (Figure 5, top). These identities remain fixed, unlike the checkpoint-wise selection for the average in Figure 4. For LIDAS, we trace the heads backward through layer duplication to the starting four-layer model. Copies share an ancestral trajectory until duplication, so we draw that shared path once.

The twelve final LIDAS heads trace back to just four ancestors. Two of these, L2H9 and L2H10 (zero-based indices), already have high PS before the first growth event and account for ten of the twelve final heads, highlighted in orange. The remaining two originate from initially low-scoring heads whose induction behavior develops later. Thus, most of the strongest final induction heads arise from copying and further training early induction heads. The original L2H10 retains high PS but is outside the final top-2% heads, while three of its copies are included.

The final locations in Figure 5 (bottom) connect these lineages to repeated patterns across depth. In LIDAS, strong heads recur at the same head index in layers separated by multiples of the four-layer growth block, extending from early layers ($L < 8$) into later layers. In Loop 10-8×2-10, high-PS heads likewise recur across the two applications of the eight-layer shared block, highlighted in orange. These are separate applications of tied heads, whereas the grown model's copies train independently. The dense baseline shows no comparable pattern of repetition by block size.

FV heads likewise recur at the grown or looped block size, but emerge later than induction heads (Appendix Figure 14). In LIDAS, two ancestors give rise to six of the twelve final top-2% FV heads.

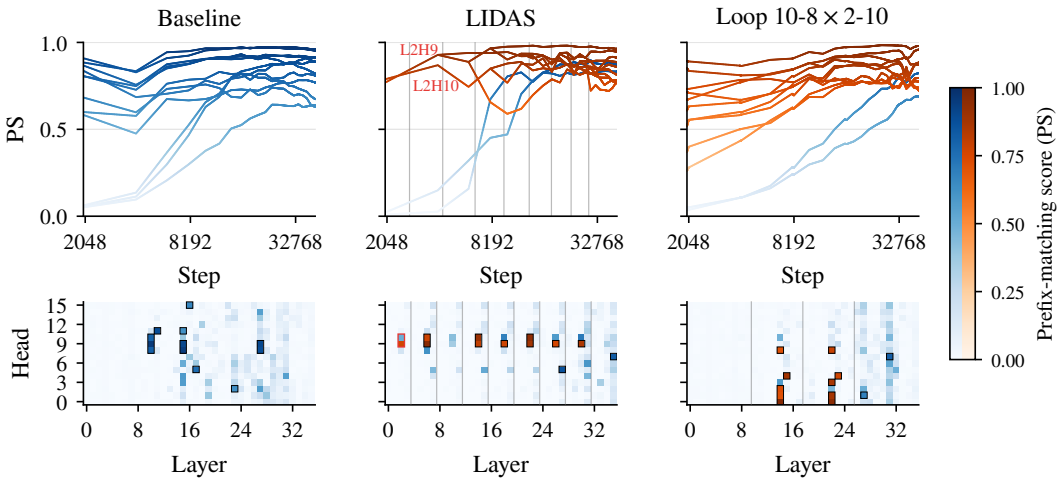


Figure 5: Depth growth propagates early strong induction heads across layers. Development and final location of individual induction heads. **Top:** smoothed prefix-matching scores (PS) of each 3B model's twelve final top-2% heads over training (logarithmic step axis). `LIDAS` lineages follow layer duplication, with shared ancestral paths drawn once. In `LIDAS`, the twelve final heads trace back to four ancestors in the initial four-layer model. Two of these, L2H9 and L2H10, already have high PS before growth and account for ten of the twelve final heads. **Bottom:** final PS for every layer and head (zero-based indices). Among the final top-2% PS heads, orange highlights descendants of the early L2H9/L2H10 heads in `LIDAS` and heads within the repeated block in Loop 10-8×2-10. Other heads are blue. Black outlines identify final top-2% heads, except the original L2H9 in `LIDAS`, which shares a red outline with L2H10. Vertical lines mark growth events (top), four-layer intervals in `LIDAS` and loop-block boundaries (bottom). Selection, plotting, and evaluation details in Appendix E.2. FV heads show analogous repetition but emerge later (Appendix Figure 14).

4.2 Induction Heads Contribute More to Context Use

The trajectories and locations reveal how induction behavior develops and repeats across depth. A high prefix-matching score alone does not establish a head's importance for ICL. We therefore ablate heads in the final models to test their contributions to few-shot performance and token loss.

Following the mean-ablation protocol of Yin & Steinhardt (2025), we rank final-checkpoint heads by PS or FV score and replace the outputs of the selected 1–20% with mean activations. For looped models, each head application is counted and ablated separately. Here we focus on exclusive induction rankings, which first exclude the top 2% of FV heads, and exclusive FV rankings, which first exclude the top 2% of induction heads. We evaluate changes from each model's unablated few-shot ICL (F1) and ICL-Score. Appendix F specifies the replacement means, indexing, and additional random controls.

Consistent with observations by Yin & Steinhardt (2025), all three models in Figure 6(a,b) show greater few-shot ICL degradation from FV-head ablation and greater ICL-Score (token-loss difference) reduction from induction-head ablation. At larger ablation fractions, exclusive induction-head ablation reduces few-shot performance and ICL-Score more in grown and looped models than in the baseline, indicating greater reliance on these heads. FV heads remain important in all models, but their ablation effects relative to the baseline vary across models and sizes for this comparison (Appendix Figure 20).

Figure 6(c) shows the NLL increase by token position at 9% ablation. Beyond the first few tokens, FV-head ablation hurts early-token prediction more than induction-head ablation. The FV-ablation effect then largely plateaus, while the induction-ablation effect grows with position. Compared with the baseline, `LIDAS` and Loop 10-8×2-10 show smaller FV-ablation effects across most positions and larger induction-ablation effects at later positions. Within both models, induction heads eventually become more important than FV heads for later-token prediction, whereas FV heads remain

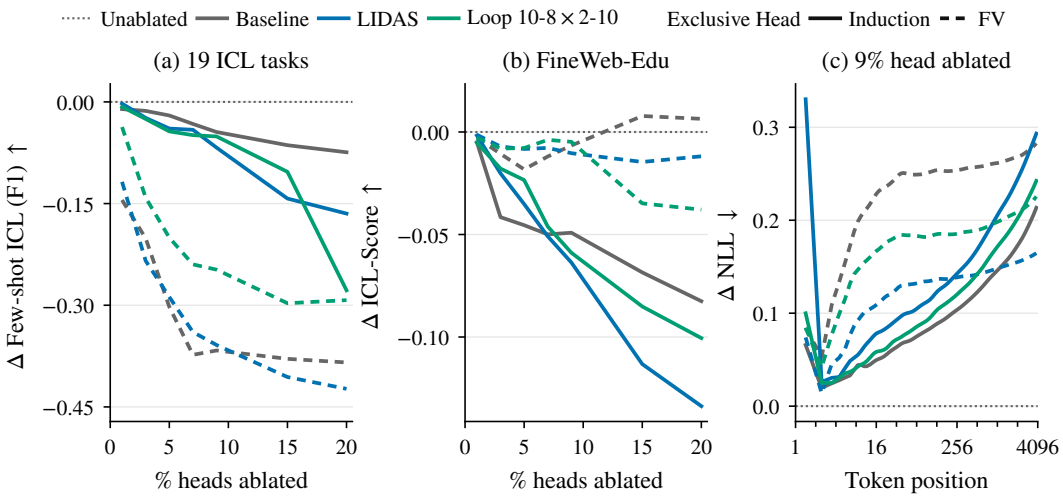


Figure 6: Grown and looped models rely more on induction heads than the baseline for few-shot ICL and later-token prediction. Mean ablation in the 3B baseline, LIDAS, and Loop 10-8×2-10. Solid lines ablate heads ranked by PS after excluding the top-2% FV heads. Similarly, dashed lines show FV-head ablations. (a) Change in few-shot ICL (F1) on 19 tasks. (b) Change in ICL-Score on FineWeb-Edu. Both changes are ablated minus the same model's unablated score (lower is worse). At larger ablation fractions, induction-head ablation reduces both scores more in grown and looped models than in the baseline, although FV-head ablation hurts few-shot ICL more in all three models. (c) Position-wise NLL increase from each model's unablated reference at 9% ablation over the 4096-token context (higher is worse). Curves are log-binned and smoothed. FV-ablation effects largely plateau, while induction-ablation effects grow with position and eventually exceed FV-ablation effects in LIDAS and Loop 10-8×2-10. Additional ablation fractions appear in Appendix Figure 21 with experimental details in Appendix F.

more important in the baseline. Together, these results indicate that both head types support context use, with greater reliance on induction heads in grown and looped models.

5 Conclusion

Our experiments provide evidence that depth growing and looping favor learning from context. In the main language-model comparison, both achieve larger early-to-late token-loss reductions and higher GSM8K accuracy than the dense baseline. Grown and looped models also predict unseen formal languages more accurately. In controlled classification tasks, both acquire ICL earlier than a full-depth dense reference when labels are reassigned between sequences, and retain higher ICL accuracy when fixed associations can also be learned in the weights. Mechanistically, depth growth propagates early induction heads through layer duplication, with most of the strongest final heads descending from a small set of early ancestors. This produces repeated patterns across depth analogous to those in looped models, even though grown layers train independently after copying. Ablations in the main language-model comparison show that grown and looped models rely more on induction-ranked heads for few-shot performance and later-token prediction. Function-vector heads remain more important for few-shot performance in all models, suggesting that depth reuse shifts the relative contributions of these mechanisms. Together, these findings suggest that reusing computations across depth shapes how models acquire and retain ICL, as well as the attention mechanisms supporting it.

The controlled tasks suggest that depth reuse shifts the competition between in-context and in-weights learning: grown and looped models retain higher ICL accuracy while attaining lower IWL accuracy than the full-depth dense reference. One possible explanation is that copying early context-dependent computations through growth, or repeatedly applying them through shared layers, makes a context-based solution less susceptible to being displaced as fixed associations are learned in the weights.

AI USE STATEMENT

Generative AI tools, including LLM-based coding agents, supported the implementation of our methods and experiments. We further used them to aid the writing of plotting code behind our figures and to polish the wording of the manuscript. All AI-assisted code, figures, and text were checked by the authors, and the research questions, experimental design, and conclusions are our own. We take full responsibility for the content of this work, including everything produced with the help of generative AI.

REPRODUCIBILITY STATEMENT

The information needed to reproduce our experiments is provided in the main text and appendix. Language-model architectures, training configurations, and growth schedules are described in Appendix A. Evaluation protocols and mechanistic analyses are detailed in Appendices B, E and F, and the RegBench and Omniglot setups, hyperparameters, and data generation procedures in Appendices C and D. We use publicly available datasets and build on public implementations, including the nanotron training framework (https://github.com/huggingface/nanotron).

REFERENCES

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. In *International Conference on Machine Learning*, 2024.

Tatsuya Aoyama, Ethan Gotlieb Wilcox, and Nathan Schneider. Predicting the emergence of induction heads in language model pretraining. In *International Conference on Machine Learning*, 2026.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise loRA. In *The Thirteenth International Conference on Learning Representations*, 2025.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, et al. Smollm3: smol, multilingual, long-context reasoner, 2025. URL https://huggingface.co/blog/smollm3.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. SmolLM-corpus, July 2024. URL https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, 2022.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. MOEUT: Mixture-of-experts universal transformers. *Advances in Neural Information Processing Systems*, 2024.

Raj Dabre and Atsushi Fujita. Recurrent stacking of layers for compact neural machine translation models. In *AAAI Conference on Artificial Intelligence*, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. In *Advances in Neural Information Processing Systems*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, 1989.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, 2025.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *International Conference on Machine Learning*, 2024.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025.

Andrea Gesmundo and Kaitlin Maile. Composable function-preserving expansions for transformer architectures. *arXiv preprint arXiv:2308.06103*, 2023.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, 2023.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of BERT by progressively stacking. In *International conference on machine learning*, 2019.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Ferdinand Kapl, Emmanouil Angelis, Tobias Höppe, Kaitlin Maile, Johannes von Oswald, Nino Scherrer, and Stefan Bauer. Do depth-grown models overcome the curse of depth? an in-depth analysis. In *Conference on Language Modeling*, 2026a.

Ferdinand Kapl, Emmanouil Angelis, Kaitlin Maile, Johannes von Oswald, and Stefan Bauer. From growing to looping: A unified view of iterative computation in LLMs. In *International Conference on Machine Learning*, 2026b.

Sanghoon Kim, Dahyun Kim, Chanjun Park, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pp. 23–35, 2024.

Yeskendir Koishekenov, Aldo Lipani, and Nicola Cancedda. Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts. *arXiv preprint arXiv:2510.07358*, 2025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. ICLR: In-Context Learning of Representations. In *International Conference on Learning Representations*, 2025.

Sashank J Reddi, Sobhan Miryoosefi, Stefani Karp, Shankar Krishnan, Satyen Kale, Seungyeon Kim, and Sanjiv Kumar. Efficient training of language models using few-shot learning. In *International Conference on Machine Learning*, 2023.

Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank Jakkam Reddi, and Sanjiv Kumar. On the inductive bias of stacking towards improving reasoning. *Advances in Neural Information Processing Systems*, 2024.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025.

Aaditya Singh, Stephanie Chan, Ted Moskovitz, Erin Grant, Andrew Saxe, and Felix Hill. The Transient Nature of Emergent In-Context Learning in Transformers. In *Advances in Neural Information Processing Systems*, 2023.

Aaditya K Singh, Ted Moskovitz, Felix Hill, Stephanie C.Y. Chan, and Andrew M Saxe. What needs to go right for an induction head? A mechanistic study of in-context learning circuits and their formation. In *International Conference on Machine Learning*, 2024.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers. In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pp. 78–90, 2023.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations*, 2024.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *International Conference on Learning Representations*, 2023.

Shaowen Wang, Ge Zhang, Kairong Luo, Yuhao Wu, Shaofan Liu, Jiaheng Liu, Wenhao Huang, Shen Yan, and Jian Li. Smelt: Scaling laws for compute-matched moe looped transformers. *arXiv preprint arXiv:2609.01343*, 2026.

Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. In *International Conference on Learning Representations*, 2024.

Kayo Yin and Jacob Steinhardt. Which attention heads matter for in-context learning? In *International Conference on Machine Learning*, 2025.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A Language Model Training and Configurations

A.1 Training Setup

We train autoregressive, decoder-only Transformers with a next-token cross-entropy objective. The 360M and 1.7B configurations follow SmolLM, and the 3B configuration follows SmolLM3. All use pre-RMSNorm, SwiGLU feed-forward layers, tied input and output embeddings, and no biases, dropout, or query/key normalization. Table 1 summarizes the dense reference configurations. Model sizes refer to these references; looped models have fewer unique parameters, and grown models reach the reference depth only during training. Comparisons between methods use the same configuration within each size. The sizes differ in training budget, and the 3B configuration also changes the data, tokenizer, and context length, so they test robustness of observations across all these differences.

Table 1: Language-model training configurations.

	360M	1.7B	3B
Dense layers L	32	24	36
Hidden / MLP width	960 / 2560	2048 / 8192	2048 / 11008
Query / key-value heads	15 / 5	32 / 32	16 / 4
Vocabulary size	49,152	49,152	128,256
Context length C	2048	2048	4096
Global batch (sequences)	512	1024	576
Training steps	200,000	200,000	42,400
Training tokens (billions)	209.7	419.4	100.0
Peak learning rate	3×10^{-3}	5×10^{-4}	2×10^{-4}
Weight decay	0.01	0.01	0.1
Initialization standard deviation	0.0417	0.0221	0.02

Data and positional encoding. The 360M and 1.7B models use the SmolLM-corpus mixture of FineWeb-Edu-dedup, Cosmopedia-v2, Python-Edu, and Open-Web-Math (Ben Allal et al., 2024), the cosmo2 tokenizer, and rotary position embeddings. Data are shuffled with seed 42 and packed into 2048-token sequences. The configured sampling weights for these four sources are 3.1, 4.6, 16.6, and 3.6, respectively; these are repeat factors applied to each source's size, rather than token proportions. The 3B models use the SmolLM3 tokenizer and its 100B-token ablation recipe. The initial mixture contains FineWeb-Edu and DCLM (weights 0.333 and 0.37), multilingual FineWeb-2, scientific text, code, and mathematics. At step 36,040, training switches to the decay mixture, with increased emphasis on mathematics, code, and synthetic reasoning. Documents are packed into 4096-token windows, with attention masked across document boundaries. The 3B architecture omits rotary embeddings in every fourth layer, uses rotary base 5×10^4 in the remaining layers, and uses RMSNorm with $\epsilon = 10^{-6}$.

Optimization and initialization. All language models use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, gradient clipping at norm 1, bf16 weights and activations, and fp32 gradient accumulation. Learning rates increase linearly for 2000 steps. For 360M and 1.7B, the rate then stays constant until step 170,000 and decays to zero over the last 30,000 steps as $1 - \sqrt{u}$, where $u \in [0, 1]$ is progress through the decay phase. For 3B, the constant phase ends at step 36,040 and the final 6360 steps use linear decay to zero; embedding parameters are excluded from weight decay. Parameters use normal initialization with the standard deviations in Table 1; attention output and MLP down projections additionally receive a $1/\sqrt{2L}$ factor at initialization. The 360M looped runs are an exception: their output projections omit this depth-dependent factor, so their comparisons also include an initialization difference.

A.2 Growing and Looping

Dense and grown models. The dense baseline has L independently parameterized layers throughout training. Grown models start with $b = 4$ layers and repeatedly duplicate four contiguous layers until reaching L. A duplication copies both the layer weights and Adam's first and second moments. The copies become independent trainable layers immediately; embeddings are retained, and neither the global step nor the learning-rate schedule is reset. `MIDAS` (Saunshi et al., 2024) selects the middle four-layer block, choosing the block just before the center when the number of blocks is even. `LIDAS` (Kapl et al., 2026a) instead selects the middle four layers without respecting previous block boundaries. For example, `LIDAS` maps an eight-layer stack to

$$[0, 1, 2, 3, 4, 5, 6, 7] \longmapsto [0, 1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 7].$$

Layer indices in this expression identify the source of the copied parameters, not persistent weight sharing.

The constant-rate training budget is allocated to successive depths in proportion to their depth (the prop-1 schedule). The final full-depth stage also contains the learning-rate decay. Growth occurs at steps 4722, 14,166, 28,332, 47,220, 70,831, 99,164, and 132,219 for 360M; at 8095, 24,285, 48,570, 80,950, and 121,426 for 1.7B; and at 2756, 4269, 6539, 9564, 13,347, 17,885, 23,180, and 29,232 for 3B. The 3B schedule allocates the 2000 warm-up steps separately before the proportional split. Only `LIDAS` is evaluated at 3B. At 1.7B, `MIDAS` branches from `LIDAS`'s eight-layer checkpoint at step 24,285: the two methods share the preceding trajectory, including the first growth event.

Looped models. We write Loop p-$b{\times}r$-c for p prelude layers, a block of b layers applied r times, and c coda layers. The effective depth is $p + br + c$, but only $p + b + c$ layers have unique parameters. Gradients from each application accumulate into the shared parameters. Prelude and coda are omitted from the notation when empty. The main 1.7B comparison uses Loop 4-8×2-4: 24 layer applications, 16 unique layers, and approximately 1.17B parameters. The 3B comparison uses Loop 10-8×2-10: 36 applications, 28 unique layers, and approximately 2.45B parameters. Its eight-layer loop preserves the rotary/no-position-embedding role of each shared layer. The 360M comparison uses only Loop 4-12×2-4 (32 applications, 20 unique layers, approximately 244M parameters); other 360M loop layouts are excluded from the appendix comparisons. These comparisons match effective final depth and training tokens within a size; growing uses fewer layer applications during early training, and looping reduces the number of unique parameters.

A.3 Checkpoints and Comparison Scope

Final-checkpoint evaluations use step 200,000 for 360M/1.7B and step 42,400 for 3B. The 1.7B and 3B trajectories generally use checkpoints every 2000 steps, with additional saves at growth boundaries and at the final step. The 360M dense and grown trajectories use a 10,000-step grid, with additional growth saves; most looped trajectories use a 5000-step grid. Additional early power-of-two checkpoints are available for the 3B looped run. At a growth boundary, we distinguish the trained pre-growth checkpoint from the immediate post-growth checkpoint, before any further optimizer update. The latter is shown as a hollow marker at the same training step. Vertical rules mark the growth events.

The appendix compares dense, `MIDAS`, `LIDAS`, and the selected looped model at 360M and 1.7B, and dense, `LIDAS`, and the selected looped model at 3B. There is no 3B `MIDAS` run. All comparisons use one language-model training seed per arm; evaluation bands do not estimate variation across training runs. The configuration differences and 360M loop initialization exception in Appendix A.1 apply throughout.

Loop configurations in appendix figures. A single green legend entry, "Loop", denotes Loop 4-12×2-4 at 360M, Loop 4-8×2-4 at 1.7B, and Loop 10-8×2-10 at 3B. The size of the panel determines the configuration; other 360M loop layouts are excluded. The architecture and parameter-sharing convention are given in Appendix A.2.

B Context Use in Language Models

This section gives the evaluation protocols and results across sizes for general benchmarks, token loss, downstream few-shot performance, and graph tracing. Shared training configurations and the size-specific Loop definitions are given in Appendix A.

B.1 General Benchmark Evaluations

Table 2 reports general benchmark performance using OLMES (Gu et al., 2025) at the final checkpoints: step 200,000 for 360M/1.7B and step 42,400 for 3B. We evaluate the nine-task core suite and MMLU (Hendrycks et al., 2021) with five fixed demonstrations and a maximum context length of 2048 tokens for every model, including the 3B models.

The core suite uses test splits for ARC-Easy, ARC-Challenge, and OpenBookQA, and validation splits for the remaining tasks. ARC-Easy, BoolQ, HellaSwag, PIQA, and Social IQa use the standard fixed 1000-example subsamples; ARC-Challenge, CommonsenseQA, OpenBookQA, and WinoGrande use their full splits of 1172, 1221, 500, and 1267 examples, respectively. MMLU uses all 14,042 test examples across its 57 subjects. Following OLMES, answer log-likelihoods are normalized by character count for ARC-Easy, HellaSwag, PIQA, Social IQa, and MMLU; corrected by unconditional answer log-likelihood for ARC-Challenge, CommonsenseQA, and OpenBookQA; and left unnormalized for BoolQ and WinoGrande.

Table 2: General benchmark performance across model sizes. Accuracy (%), with higher values better. **9-task MC** is the unweighted mean over ARC-Easy, ARC-Challenge, BoolQ, CommonsenseQA, HellaSwag, OpenBookQA, PIQA, Social IQa, and WinoGrande; **MMLU** is the unweighted mean over its 57 subjects. Both columns use five-shot ranked-classification (RC) scoring of answer continuations with OLMES task-specific normalization and a 2048-token context limit. Here MC denotes multiple-choice benchmarks; scores use the RC formulation throughout, rather than answer-letter scoring. Model sizes denote dense reference configurations; loop layouts follow Appendix A.2. Best values within each size are bold.

Model	Strategy	9-task MC ↑	MMLU ↑
360M	Baseline	54.69	**34.67**
	MIDAS	56.00	34.11
	LIDAS	**57.10**	34.09
	Loop 4-12×2-4	54.21	33.69
1.7B	Baseline	62.97	40.12
	MIDAS	64.61	40.43
	LIDAS	**64.69**	**40.86**
	Loop 4-8×2-4	63.25	39.89
3B	Baseline	61.33	39.15
	LIDAS	**62.89**	**39.28**
	Loop 10-8×2-10	60.53	38.32

B.2 Token Loss and the ICL Score

Figure 1(a,b), Figure 7, and Figure 13 (bottom row) use a fixed set of 10,000 long FineWeb-Edu-dedup documents. After tokenization, we retain documents with at least $C+1$ tokens and truncate to $C+1$, yielding C next-token predictions per document. Each forward pass therefore uses a single document and every position has the same document count. The 2048- and 4096-position evaluations use their model's native tokenizer and separately selected document sets.

For tokens $x_0, \ldots, x_C$, define

$$\ell_j = -\log p(x_j \mid x_0, \ldots, x_{j-1}), \qquad j = 1, \ldots, C, \tag{1}$$

and let $\bar{\ell}_j$ be its mean over documents, in nats. Position labels in the trajectory and position plots use this one-based prediction index. Evaluation uses bf16 forward passes. For the position-wise ΔNLL panels in Figure 1(a) and Figure 7, position means are grouped into non-overlapping power-of-two bins $(2^y, 2^{y+1}]$ for $y = 0, \ldots, \log_2 C - 1$, with position 1 added to the first bin. The integer position ranges are therefore 1–2, 3–4, 5–8, …, $C/2{+}1$–C, giving 11 bins at $C = 2048$ and 12 at $C = 4096$; every position belongs to exactly one bin. Each bin mean is plotted at the bin's upper endpoint on a base-2 logarithmic axis. The plotted quantity is the model's binned NLL minus the same-size dense baseline's binned NLL. Thus, a negative value denotes better token prediction at those positions, rather than a larger improvement relative to the model's own early-token loss.

The dashed references in Figure 1(a) and Figure 7 average the raw position means with equal weight, before any position binning. With $D = 10{,}000$ documents and C scored prediction positions per document ($C = 2048$ at 360M/1.7B and 4096 at 3B), the reference for model m is

$$\begin{aligned} \overline{\Delta\mathrm{NLL}}_m &= \frac{1}{C}\sum_{j=1}^{C}\left(\bar{\ell}_j^{(m)} - \bar{\ell}_j^{(\mathrm{base})}\right) \\ &= \frac{1}{DC}\sum_{d=1}^{D}\sum_{j=1}^{C}\left(\ell_{d,j}^{(m)} - \ell_{d,j}^{(\mathrm{base})}\right) \\ &= \overline{\mathrm{NLL}}_m - \overline{\mathrm{NLL}}_{\mathrm{base}}. \end{aligned} \tag{2}$$

This is the difference in token-averaged dataset NLL on this fixed long-document subset, using the same truncation and per-document context resets as the position curves. It is not an unweighted average of the logarithmic bin means, whose position counts differ.

Following the token-loss diagnostic of Olsson et al. (2022), we summarize context use at each checkpoint by

$$S_{\mathrm{ICL}} = \frac{1}{21}\sum_{j=40}^{60} \bar{\ell}_j - \frac{1}{21}\sum_{j=490}^{510} \bar{\ell}_j. \tag{3}$$

A positive score indicates lower NLL at the later window. A change in this difference can arise from either window; it need not imply improved absolute prediction at late positions. Window-decomposition plots therefore report the two means separately. When subtracting the dense model's corresponding window loss along training, we use only steps at which both models have a saved checkpoint, without interpolating a baseline at growth steps. The combined head-ablation figure uses the same window definition and reports its change from the unablated model in Figure 6(b). The additional ablation diagnostics in Figure 22 use the single-position convention defined in Appendix F.

Loss reduction across context positions. For the final-checkpoint comparison in Figure 1(b) and Figure 7, we keep the early window at positions 40–60 and let the later window be the power-of-two bin ending at the target position x, $W_x = \{x/2 + 1, \ldots, x\}$, for $x \in \{256, 512, 1024, 2048\}$ and additionally 4096 at 3B. These are the same bins as in the ΔNLL panels. We plot

$$\Delta S_m(x) = \left[\frac{1}{21}\sum_{j=40}^{60} \bar{\ell}_j^{(m)} - \frac{1}{|W_x|}\sum_{j\in W_x} \bar{\ell}_j^{(m)}\right] - \left[\frac{1}{21}\sum_{j=40}^{60} \bar{\ell}_j^{(\mathrm{base})} - \frac{1}{|W_x|}\sum_{j\in W_x} \bar{\ell}_j^{(\mathrm{base})}\right]. \tag{4}$$

This is the model's early-to-late loss reduction minus the baseline's. Equivalently, $\Delta S_m(x)$ equals the model's ΔNLL over positions 40–60 minus its ΔNLL in the bin ending at x, the value plotted in the top row of Figure 7. Scores and the full-context NLL references are computed from raw position means.

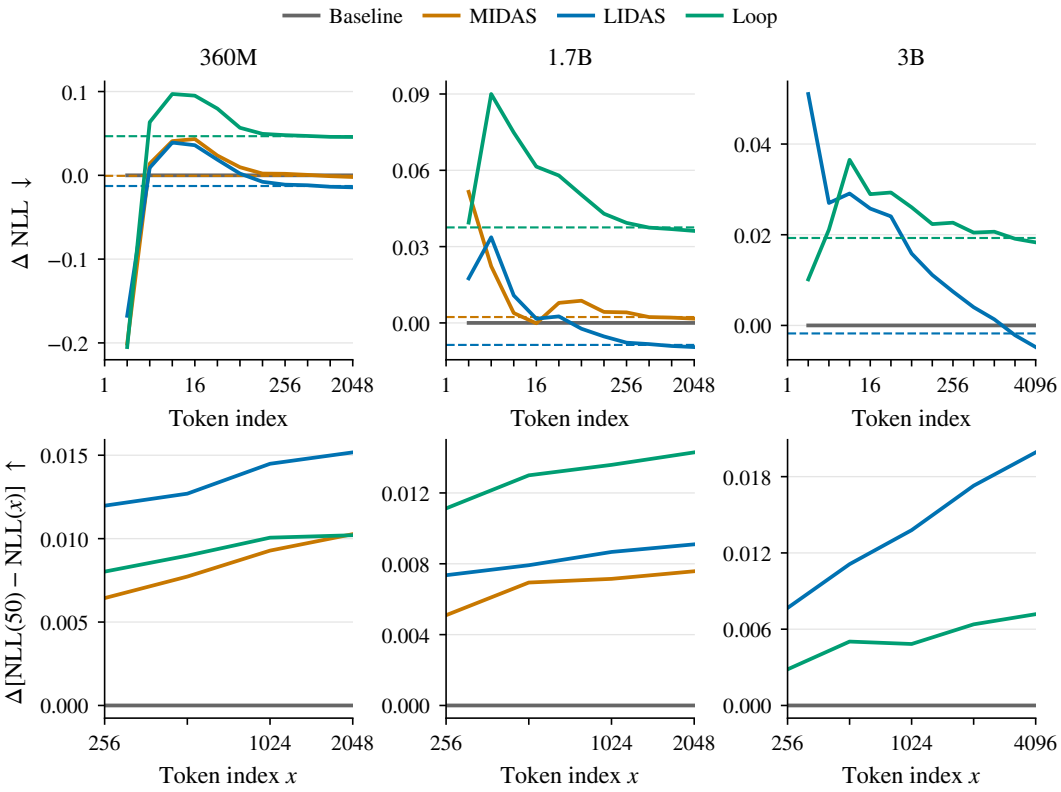


Figure 7: Token loss and context gains across model sizes. Columns show 360M, 1.7B, and 3B, using the position bins of Figure 1. Top: final-checkpoint NLL relative to the same-size dense model in power-of-two position bins plotted at their upper endpoints, with matching-color dashed full-context means computed over raw positions. Bottom: the early-to-late loss reduction relative to dense (Equation (4)) for later bins ending at $x \in \{256, \ldots, C\}$. All curves use 10,000 long FineWeb-Edu documents. Native contexts are $C = 2048/2048/4096$ positions; the evaluation protocol is given in Appendix B.2. Loop configurations are specified in Appendix A.3.

B.3 Few-Shot Scaling on Downstream Tasks

Figure 1(c) measures downstream performance as a function of the number of demonstrations, following the demonstration-task protocol also used by SMELT (Wang et al., 2026). We use lm-eval-harness version 0.4.3, bf16 weights, and a maximum input length equal to the training context. Demonstrations are sampled using seeds 1234, 42, and 7. The available grid is $k \in \{0, 1, 2, 4, 8, 16, 32\}$ at 2048 tokens and extends to 64 at 4096 tokens. We re-tokenize requests to check that prompts fit; each task's maximum k is capped to avoid truncation. Consequently, a longer-context macro average must specify its task set as well as its shot range.

The cross-size appendix comparisons use $k \in \{0, 1, 2, 4, 8\}$ and the same fixed set of eight tasks, excluding BoolQ: ARC-Easy, ARC-Challenge, CommonsenseQA, HellaSwag, PIQA, and the BigBench tasks QA-Wikidata, Operators, and Repeat-Copy-Logic. ARC, HellaSwag, and PIQA use length-normalized multiple-choice accuracy; CommonsenseQA uses accuracy; the remaining three use whitespace-stripped exact match for generated answers. We report separate averages for the five multiple-choice tasks and the three generation tasks. We first average each task over the three demonstration seeds. For either subset $\mathcal{T}$, with task mean accuracy a_t and chance level c_t, the displayed centered macro accuracy is

$$A_{\text{centered}}(\mathcal{T}) = \frac{100}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{a_t - c_t}{1 - c_t}. \tag{5}$$

Chance levels are 0.25 for ARC-Easy, ARC-Challenge, and HellaSwag; 0.5 for PIQA; 0.2 for CommonsenseQA; and zero for the three generative tasks. Thus, the generation-subset average equals raw exact-match accuracy. Zero centered accuracy denotes chance performance and 100 denotes perfect performance. The task composition is identical across models, sizes, and shot counts. Dyck Languages and CS Algorithms are excluded because their varying answer-option counts leave no fixed chance level in the task definitions. LSAT-AR, SQuAD completion, and Language Identification do not all support eight demonstrations within the shorter context windows and are also excluded. Figure 10 shows all eight task curves with standard error across the three demonstration seeds; these bands do not estimate uncertainty across training seeds. Figure 9 shows the multiple-choice and generation averages in separate rows, with columns for model size. Its bands show standard error across three seed-level subset averages, computed by averaging the chance-centered task scores within each subset and demonstration seed before estimating uncertainty.

The main figure, Figure 1(c), and Figure 8 report GSM8K, using strict exact match averaged over the same three demonstration seeds. The available shot counts are $k \in \{0, 1, 2, 4\}$ at 360M and 1.7B, and $k \in \{0, 1, 2, 4, 8\}$ at 3B. GSM8K is reported separately and is not included in either task subset.

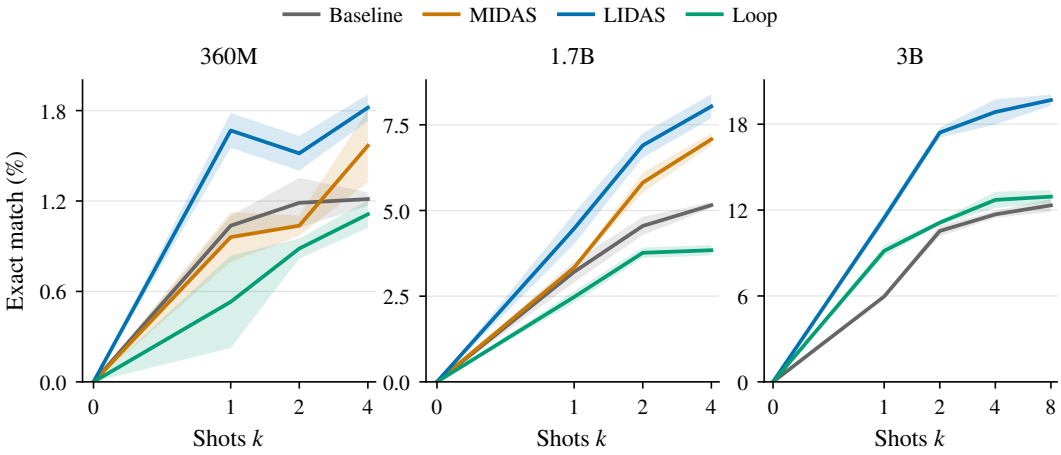


Figure 8: Few-shot GSM8K performance across model sizes. Panels show 360M, 1.7B, and 3B. Curves are strict exact-match accuracy, with means and standard errors over demonstration seeds 7, 42, and 1234. The shot range is $k \in \{0, 1, 2, 4\}$ for 360M and 1.7B and extends to eight shots for 3B. Each panel has an independent y-axis scale. GSM8K is excluded from the task subsets in Figure 9; see Appendix B.3. Loop configurations are specified in Appendix A.3.

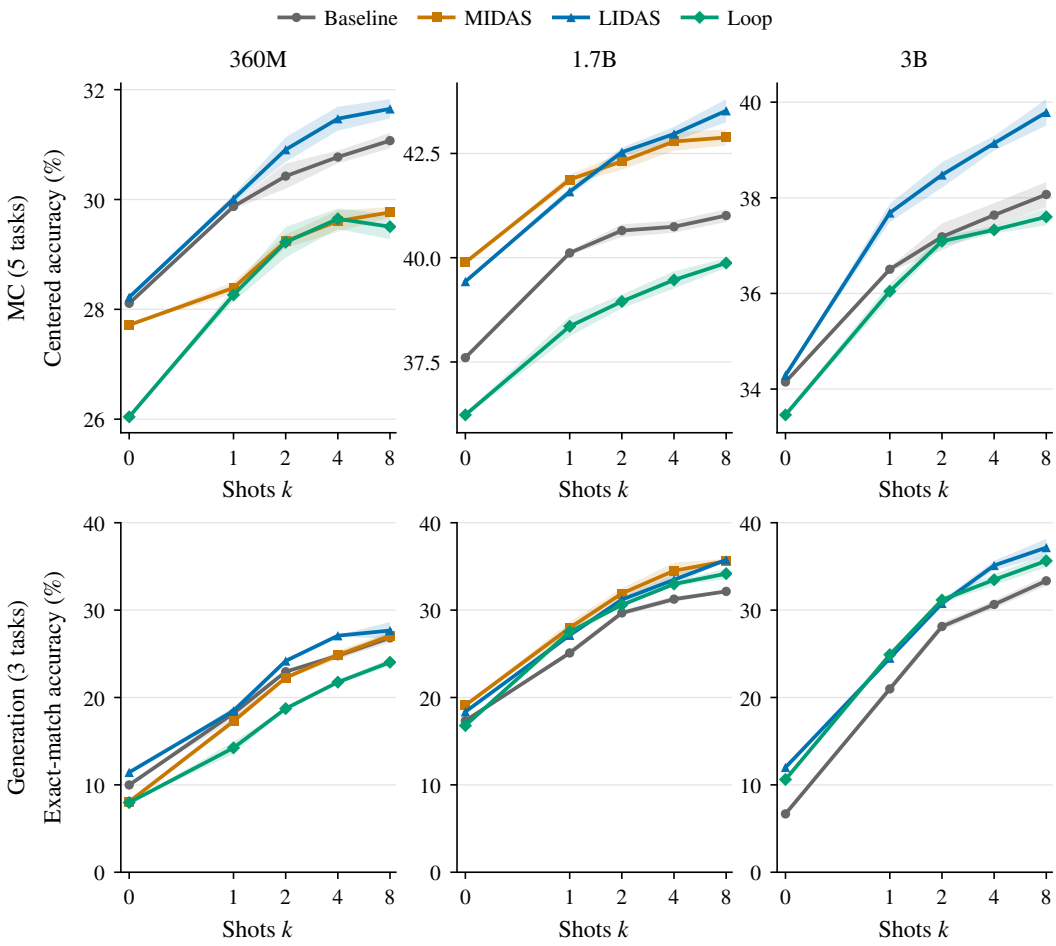


Figure 9: Few-shot performance on multiple-choice and generation tasks. Columns show 360M, 1.7B, and 3B. The top row averages chance-centered accuracy over five multiple-choice tasks; the bottom row averages exact-match accuracy over three generation tasks, for $k \in \{0, 1, 2, 4, 8\}$. Tasks have equal weight within each subset; bands show standard error across three demonstration-seed subset averages. Multiple-choice panels use independent y-axis scales; generation panels share a 0–40% scale. Individual tasks and normalization are given in Figure 10 and Appendix B.3. Loop configurations are specified in Appendix A.3.

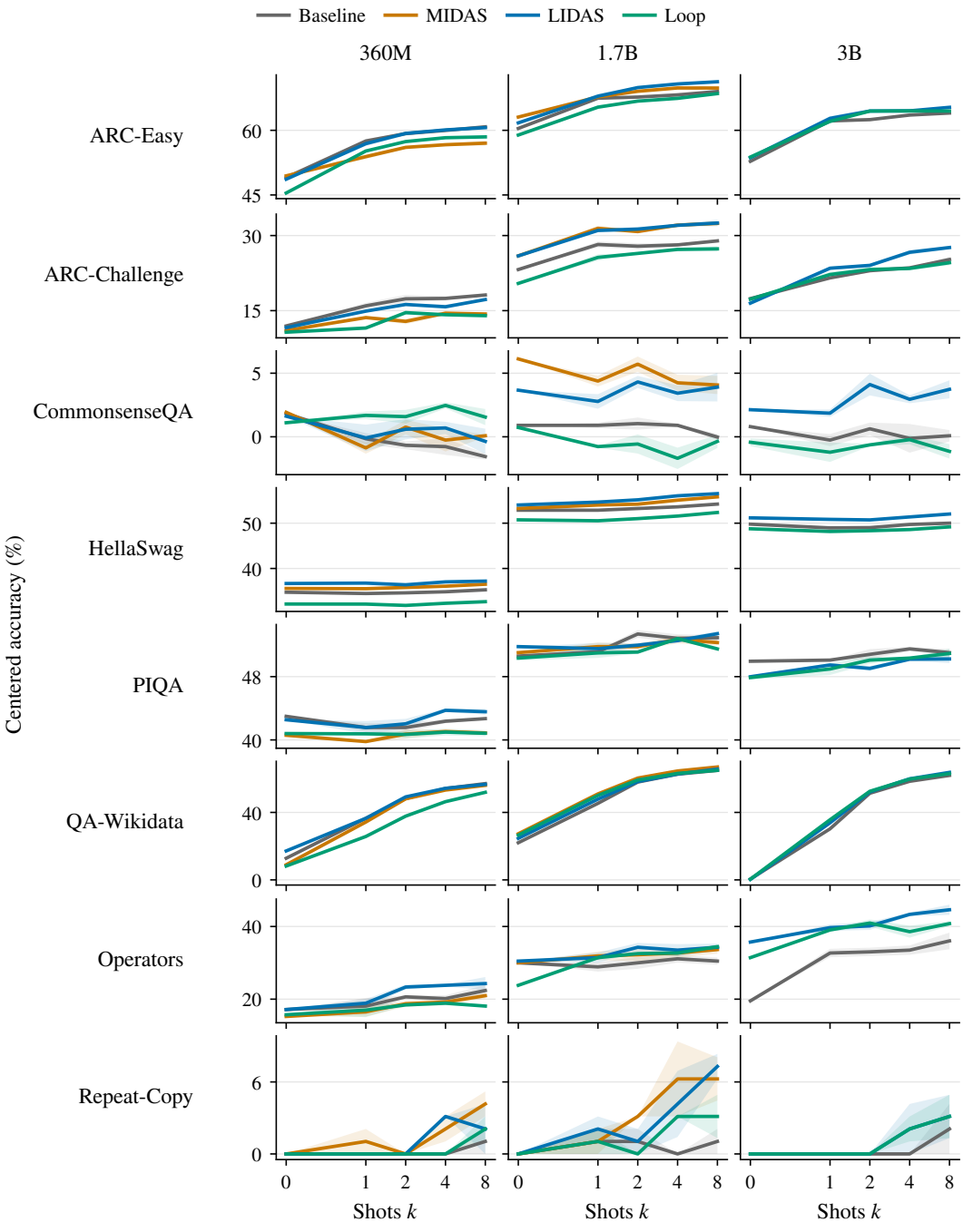


Figure 10: Few-shot performance on the eight tasks used in the subset averages. Columns show 360M, 1.7B, and 3B; rows show tasks. Curves show chance-centered accuracy for $k \in \{0, 1, 2, 4, 8\}$, with means and standard errors over three demonstration seeds. Scales are shared across sizes within each row. Each task contributes equally within its multiple-choice or generation subset in Figure 9; chance levels and normalization are specified in Appendix B.3. Loop configurations are specified in Appendix A.3.

B.4 In-Context Graph Tracing

We additionally evaluate the grid-graph setting of Park et al. (2025). Nine or sixteen unrelated, single-token nouns label the nodes of a 3×3 or 4×4 grid, respectively. Figure 11 shows both grid sizes with frequent-word labels for all three model sizes. A random walk is presented as a space-separated sequence of node words, without an additional delimiter. After each node, we sum the model's full-vocabulary next-token probabilities over the words labeling that node's valid neighbors. This probability mass is the graph task's "accuracy"; it is neither an argmax success rate nor a distribution renormalized over node words.

The frequent-word comparison starts from a curated pool of 250 nouns spanning 18 semantic fields. For each tokenizer, we retain nouns that form a single token when preceded by a space and select the 64 with the lowest token IDs. The 360M/1.7B models use cosmo2, whereas 3B uses the SmolLM3 tokenizer; token ID serves as a frequency proxy. Each of 200 trials draws a new node-word assignment and random walks, with one sequence per starting node; scores are averaged over starting nodes within the trial. Models of a given size see the same trials. The horizontal axis counts node tokens in context, up to the native training context length, and bands show the standard error over trials. The one- and two-observation references measure the fraction of positions whose current node has already appeared as a source of a transition at least once or twice. They indicate when copying a previously observed successor becomes possible, rather than upper bounds on graph learning. The 3B comparison includes the dense, LIDAS, and looped models.

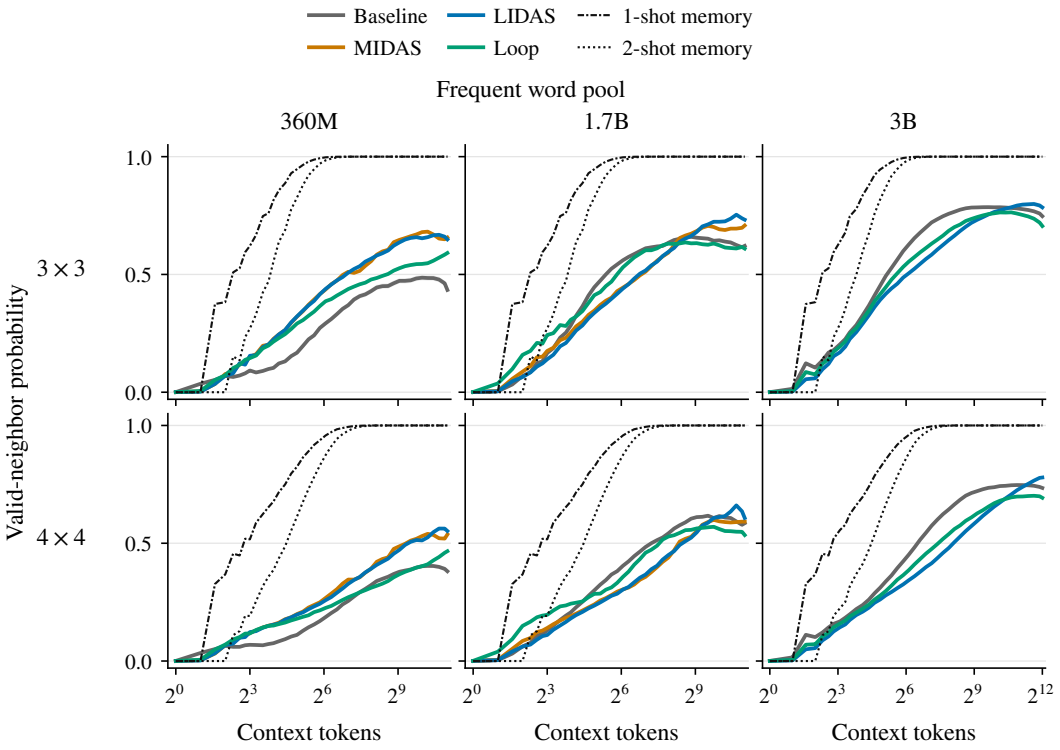


Figure 11: Word-grid tracing with frequent-word labels. Columns show 360M, 1.7B, and 3B; rows show 3×3 and 4×4 grids. Curves give full-vocabulary probability mass assigned to valid neighbors, with standard error over 200 trials. Black curves are one- and two-observation edge-memorization references. All panels share the probability scale and use base-2 logarithmic context axes. Both settings include the same selected methods at each model size. See Appendix B.4 for the protocol. Loop configurations are specified in Appendix A.3.

C REGBENCH: SETUP AND ADDITIONAL RESULTS

C.1 TASK AND DATA

We use the released RegBench implementation of Akyürek et al. (2024). Each sequence concatenates random walks from one DFA, separated by |. Candidate automata have 4–12 states, an alphabet of 4–18 symbols, and one to three outgoing transitions per state; they are minimized, trimmed, and deduplicated before splitting. Each automaton supplies ten to nineteen strings of 1–49 symbols. The fixed dataset contains 40,000 training, 500 validation, and 500 test sequences with disjoint automata. All models see the same data. Training and validation cross-entropy mask separator targets and padding.

C.2 MODELS, OPTIMIZATION, AND SELECTION

We use GPT-2-style pre-LayerNorm Transformers with learned token and position embeddings, two attention heads, a GELU MLP of width $4d$, and tied input/output embeddings. Embedding and attention dropout are 0.1. Dense models have twelve independent layers; the looped model applies two shared layers six times. Grown models reach depth twelve by duplicating the middle b layers and their AdamW moments at each growth event. Copies then train independently, retaining the global learning-rate schedule without restarting warm-up.

All runs use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, batch size 32, and 200 epochs (250,000 steps). The learning rate warms up for 25,000 steps and then decays by a cosine schedule to one tenth of its peak. For each configuration, we select the lowest-validation-loss cell over widths $d \in \{64, 128, 256\}$, peak learning rates $\{10^{-4}, 2.5 \times 10^{-4}\}$, and weight decays $\{0.01, 0.1\}$. We evaluate the best-validation checkpoint, restricting grown models to their final full-depth stage, and repeat the selected cells with training seeds $\{0, 1, 2, 3\}$.

Figure 12 includes all five repeated configurations: dense 12L, Loop 2×6, and grown $b = 2$/prop-1, $b = 2$/prop-2, and $b = 3$/prop-3. Their widths are 128 except for Loop 2×6, which uses 256. All select weight decay 0.1; the peak learning rate is 10^{-4} for prop-1 and 2.5×10^{-4} otherwise. The grown stage endpoints are $(9, 28, 57, 95, 142, 200)$ epochs for prop-1, $(2, 10, 30, 65, 120, 200)$ for prop-2, and $(2, 18, 72, 200)$ for prop-3, at depths increasing by b. The main figure uses $b = 3$/prop-3, selected by mean validation loss across seeds. Selection never uses test performance. Comparisons match data and optimizer steps, but not width or compute.

C.3 METRICS AND POSITION DECOMPOSITION

Let q be uniform over the DFA's legal outgoing symbols and zero elsewhere. We report $\mathrm{TVD}(p, q) = \frac{1}{2} \sum_v |p(v) - q(v)|$ and DFA accuracy, which scores whether the argmax is a legal continuation rather than the particular sampled next symbol. Following the released evaluator, TVD includes all non-padding output positions, including separator targets and the final position; q assigns no mass to the separator. Accuracy excludes separator targets and includes the final position. These masks differ from the training loss.

Figures 2 and 12 report full-test means and position-wise differences from dense. Position statistics pool sums and counts within each seed, then average over seeds. String and symbol indices start at 1. In Figure 2, we retain only string indices present in every one of the 500 test sequences (strings 1–10), using all scored symbol positions within these strings (through index 50 for TVD). No symbol-range, coverage, or minimum-support cutoff is applied; only empty bins are omitted. The appendix figure uses all strings 1–19 and all available symbol positions (through 50 for TVD and 49 for accuracy), without range or coverage cutoffs; only empty bins are omitted. Within each figure and metric, the string-index and symbol-index views use the same population; the latter pools symbol indices in bins of five and separates string 1 from the remaining strings (2–10 in the main figure and 2–19 in the appendix).

Error bars show standard error over the four training seeds. Bands on differences use the unpaired standard error $\sqrt{s^2_{\mathrm{model}}/4 + s^2_{\mathrm{dense}}/4}$, where s is the sample standard deviation of seed-specific bin means. Negative TVD differences and positive accuracy differences favor the alternative model. Position-restricted averages need not equal the full-test statistics.

C.4 Results Across Configurations

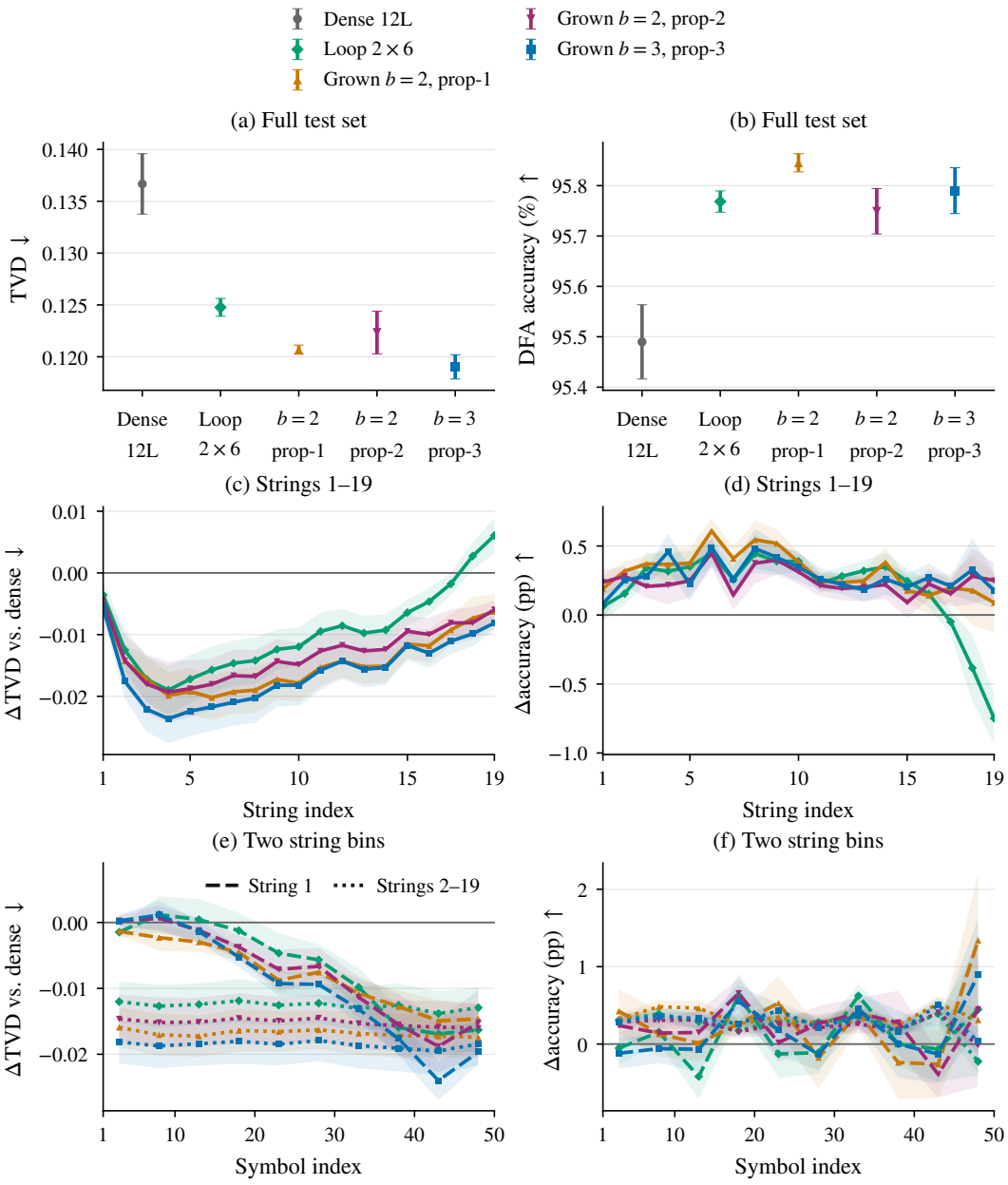


Figure 12: RegBench across all configurations evaluated with four seeds. Columns show TVD (left) and DFA accuracy (right). Rows show full-test means, differences from dense by string index, and differences by symbol index for string 1 (dashed) and strings 2–19 (dotted). All available strings and symbols are retained; indices start at 1 and symbol positions use bins of five. Of the 500 test sequences, 145 (29%) reach string 17, 102 reach string 18, and 54 reach string 19. The positive mean Loop 2×6 TVD differences at strings 18–19 therefore concern these smaller subsets of longer sequences. Accuracy differences are in percentage points. Error bars and difference bands show standard errors across training seeds. All models have effective depth twelve; validation-selected widths are 128 except for Loop 2×6 (256). See Appendix C.3.

D OMNIGLOT SETUP

D.1 DATA, SEQUENCE FORMAT, AND ARCHITECTURE

We follow the image-based setup of Chan et al. (2022). We merge the Omniglot training and test splits, giving 1623 character classes with twenty 105×105 grayscale exemplars each. A fixed class permutation with seed 1337 reserves ten classes for held-out evaluation; the remaining 1613 classes are sampled uniformly during training. We use no image augmentation, downsampling, or pixel noise. Repeated occurrences of a class within a sequence use distinct exemplars sampled without replacement.

Each sequence contains eight image–label context pairs and one query image. Their embeddings are interleaved as $[x_1, y_1, \ldots, x_8, y_8, x_9]$, giving seventeen tokens, with sinusoidal absolute positional encodings. Training minimizes cross-entropy only at the query position. Images are embedded by a jointly trained small ResNet: a 7×7 stride-two convolution with 64 channels, batch normalization, ReLU, and a 3×3 stride-two max pool, followed by four groups of two bottleneck residual blocks with output channels 16/32/32/64 and strides 1/2/2/2, global average pooling, and projection to width 64. Labels use a learned 1623×64 embedding table.

The sequence model is a causal pre-LayerNorm Transformer of width 64, with eight attention heads, GELU MLPs of width 256, no dropout, a final LayerNorm, and a linear head with 1623 outputs. Figure 3 compares dense depths 4 and 12; a Loop 4×3 model; and a `LIDAS`-grown model. The latter branches from the four-layer dense run, grows to eight layers at step 50,000, and to twelve at step 100,000 by duplicating the middle four layers. We copy Adam moments together with weights and then train all copies independently, retaining the global step and learning-rate schedule. Thus, the grown model and the four-layer reference have exactly the same pre-growth trajectory. The looped model shares only Transformer blocks; the image encoder, embeddings, and final readout are not repeated.

D.2 TRAINING DISTRIBUTIONS AND OPTIMIZATION

Competing context and weight solutions. In Figure 3(a), each training class has a fixed label and every sequence is bursty, $p(\text{bursty}) = 1$. We sample two training classes, place three exemplars of each in the context, add two further uniformly sampled training items, and shuffle the eight context items. The query is a new exemplar of one of the two burst classes. Its label can be recovered either by matching it to context examples or by recalling its fixed training-class association. Unlike the original $p(\text{bursty}) = 0.9$ setup, there is no non-bursty training branch.

ICL-only training. In Figure 3(b), training uses four-shot, two-way sequences: two training classes each contribute four context exemplars, and the query is a fifth exemplar of one of them. Labels 0 and 1 are reassigned randomly to the two classes in every sequence. There is therefore no persistent class-to-answer mapping that can solve the query without context. The model still has 1623 output logits, although only labels 0 and 1 are correct in this setting. Architecture, optimizer, and sequence length otherwise remain unchanged.

Both settings use Adam with $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, batch size 32, no weight decay or gradient clipping, and learning rate

$$\eta(t) = 3 \times 10^{-4} \min\left(\frac{t}{4000}, \sqrt{\frac{4000}{t}}\right), \qquad t > 0. \tag{6}$$

This gives 4000 steps of linear warm-up followed by inverse-square-root decay. Each fixed-label configuration uses four seeds, $\{0, 42, 2178, 3141\}$; each ICL-only configuration uses eight, $\{0, 1, 7, 42, 1618, 2178, 2718, 3141\}$. These control initialization and sampling independently of the fixed class split. For the reported comparisons, we use 40M training steps in the fixed-label setting and 1M training steps in the ICL-only setting. Checkpoints are saved every 50,000 steps and at powers of two starting at 64 for finer early-training resolution.

D.3 ICL, IWL, AND AGGREGATION

The ICL probe uses four-shot, two-way sequences from the ten held-out classes, with new class-to-label assignments in each sequence. The context contains four distinct exemplars per class, and the query is a fifth exemplar of one of them. We report accuracy with the argmax restricted to logits 0 and 1 (chance $1/2$). Success therefore requires using the current context for a class never encountered in training. The IWL probe instead samples the query uniformly from the 1613 training classes, uses its fixed training label, and guarantees that its class is absent from the eight context examples. It reports the unrestricted argmax over all 1623 outputs. Uniform guessing over that output space gives chance $1/1623$; restricting guesses to training-class labels would give $1/1613$. The IWL probe is used only for the fixed-label setting.

Every checkpoint is evaluated offline on the same sequences generated with a fixed evaluation seed. The fixed-label setting uses 10,000 sequences per probe and the ICL-only setting uses 2000 for its ICL probe. Curves average over four training seeds in the fixed-label setting and eight in the ICL-only setting, at checkpoints shared by all seeds within each configuration; bands are one standard error across seeds. For display, each seed trajectory is smoothed with a centered, equally weighted moving average over 51 checkpoints in the fixed-label setting and 3 in the ICL-only setting, shortening windows at the boundaries after truncating to the reported training budget. Means and standard errors are then computed across the smoothed seed trajectories. Endpoint statistics and emergence thresholds in the text use unsmoothed evaluations. Variable walltime-final checkpoints are excluded from these common grids. The two plots in panel (a) use the matched 40M-step horizon, while panel (b) uses the shared 1M-step horizon. All three displayed axes are linear. These horizons describe the plotted data, rather than the maximum duration of every underlying run.

Individual-seed checks are particularly relevant in the ICL-only setting: in additional eight-layer runs, omitted from Figure 3, two of the eight seeds remain near chance through 1M steps, so their lower mean partly reflects variation in emergence time. The corresponding emergence diagnostic records the first evaluated checkpoint with ICL accuracy at least 0.7. A seed that does not reach the threshold by the common horizon is marked as unreached, rather than assigned an observed emergence time at that horizon. Longer-horizon or individual-seed views use the same probes and retain their actual checkpoint grids.

E INDUCTION-HEAD DEVELOPMENT AND READOUTS

E.1 MEASURES AND AGGREGATE TRAINING TRAJECTORIES

We implement the head diagnostics of Yin & Steinhardt (2025) and Aoyama et al. (2026) on converted language-model checkpoints, using fp32 weights and TF32 matrix multiplications. The same prompts and random sequences are reused across checkpoints and models within each size. Attention-head outputs are measured immediately before the attention output projection. For looped models, each application of a shared head is evaluated separately, since its input and causal role can differ across passes.

Prefix matching and associative recall. We generate 1000 sequences of the form $[\text{doc-start}, x_1, \ldots, x_{50}, x_1, \ldots, x_{50}]$, drawing 50 distinct regular vocabulary tokens per sequence. For a second-half token, the induction target is the token following its earlier occurrence. Prefix-matching score (PS) is the mean attention probability assigned to the target position, averaged over second-half positions and sequences. With zero-based sequence positions $p \in \{51, \ldots, 100\}$, the target position is $p - 49$; the last position uses the cyclic continuation x_1. Associative recall (AR) is the full model's top-1 accuracy on these targets. Logit attribution (LA) projects a head's output through the attention output matrix and unembedding, restricts contributions to the 50 sampled tokens, subtracts their mean, applies a ReLU, and measures the share assigned to the target. PS measures attention routing, whereas LA measures the target's share of the head's positive direct logit contribution.

Function-vector scores. Following Todd et al. (2024); Yin & Steinhardt (2025), we estimate a function-vector (FV) score on 29 training tasks using ten demonstrations and 100 clean and 100 corrupted prompts per task. We do not filter tasks or prompts by performance on clean prompts. For

each head and task, we compute its mean last-token output on clean prompts. We then shuffle the demonstration labels and replace that head's last-token output with the clean task mean. The FV score is the patched minus unpatched probability of the correct *first answer token* on the corrupted prompt, averaged over prompts and then tasks. Head selection therefore uses a causal intervention on demonstration-based task execution, rather than the attention pattern alone.

Few-shot task performance. Figure 13 (third row), Figure 19, and Figure 6 use a separate, fixed ten-shot evaluation from the downstream shot-scaling experiment. It comprises eleven held-out Todd tasks and eight additional tasks from the Yin–Steinhardt evaluation: AG News, Capitalize First Letter, CoNLL Organization, National Parks, Park–Country, Person–Occupation, Animal versus Object (three items), Choose First of Five, English–German, Person–Instrument, CommonsenseQA, Capital Index, Capitalize Second Letter, Abstract Classification, French–English, and the Fruit, Capital-Par, Capital, and Shape binding tasks. These tasks are disjoint from those used to estimate FV scores.

Prompts interleave `Q:` inputs and `A:` outputs, separate demonstrations by a blank line, and end at the query's `A:`. A document-start token is prepended. We draw ten demonstrations from each task's demonstration split and evaluate 100 queries, generating up to five tokens greedily. The answer is extracted from the generated continuation, lowercased, stripped of punctuation and English articles, and split on whitespace. We report token-overlap F1 against the reference, averaged first over queries and then over the nineteen tasks. Prompt sampling is fixed per task across checkpoints. This F1 is distinct from the benchmark accuracies in Appendix B.3.

Checkpoint-specific summaries. In Figures 4 and 13, PS (and FV in the appendix) are summarized by averaging the top $\lceil 0.02H \rceil$ heads under each measure at each checkpoint, where H is the current number of head applications. At full depth, $H = 480$, 768, and 576 for 360M, 1.7B, and 3B, giving 10, 16, and 12 selected heads. These sets are reselected at every checkpoint and may have different membership.

The PS trajectories in Figure 4 use a linear training-step axis and only trained checkpoints shared by all displayed models. For 3B, this gives 22 checkpoints: steps 2000–42,000 every 2000 steps, plus step 42,400; the 1.7B companion uses 100 checkpoints from step 2000 to 200,000. Step zero and additional model-specific trained checkpoints are excluded before plotting and smoothing. Display smoothing averages up to three adjacent shared checkpoints within 2000 steps of the center, including across growth boundaries. The first and final checkpoints remain raw. Immediate post-growth evaluations are shown only as raw hollow markers and never enter smoothing. In Figure 13, the 1.7B `MIDAS` post-growth ICL score at step 24,285 is approximately -0.72; its marker is placed on the lower boundary because it lies outside the displayed range. The regular checkpoint spacing of the dense and grown main runs does not resolve the earliest induction-head transition.

ICL-related training trajectories. Figure 13 extends the main-paper PS comparison to FV scores, few-shot F1 on nineteen held-out tasks, and the token-loss ICL-Score, across all three model sizes. In the 3B comparison, early growth events disrupt both PS and FV scores, followed by recovery during further training. The grown model reaches higher peak scores than the baseline, then finishes with higher PS but lower FV scores. The looped model finishes close to the baseline in PS and slightly above it in FV score. Few-shot F1 improves during training, with the grown model initially trailing and ultimately exceeding the baseline, while the looped model finishes close to it. Meanwhile, the ICL-Score declines for all models as early-token predictions improve more than later-token predictions. Both grown and looped models nevertheless finish with higher ICL-Scores than the baseline.

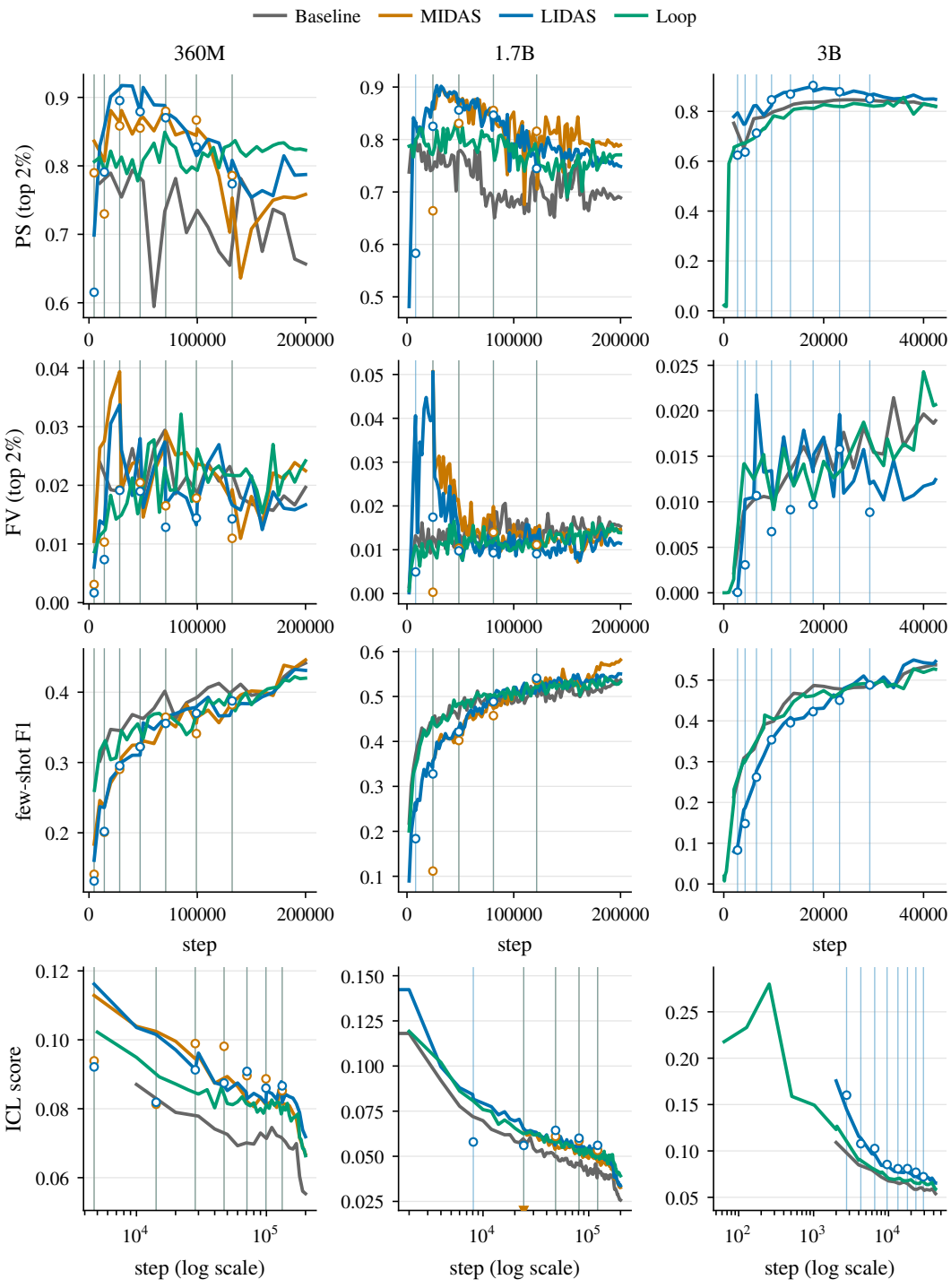


Figure 13: Training trajectories across model sizes. Columns show 360M, 1.7B, and 3B. Rows report PS and FV averaged over their respective checkpoint-specific top-2% head sets, nineteen-task ten-shot F1, and the window-averaged ICL-Score (Equation (3)). These appendix curves connect raw available trained checkpoints, without the shared-grid display smoothing used in Figure 4. The first three rows use linear steps; the last uses logarithmic steps. Vertical rules mark growth and hollow markers show immediate post-growth evaluations. The downward marker in the 1.7B ICL-Score panel denotes `MIDAS` at -0.723, below the plotted range. Early checkpoint coverage differs across sizes (Appendix A.3). Loop configurations are specified in Appendix A.3.

E.2 Final-Head Identities, Lineages, and Locations

The top row of Figure 5 selects the final top-2% PS heads once and traces those identified heads backward. For grown models, we invert the actual layer-duplication maps: a head and its later copies share the same ancestral trajectory until duplication. This lineage view omits coincident pre/post-growth checkpoints and connects adjacent regular checkpoints. The bottom row of Figure 5 displays every head's final PS with a common color scale; looped layers appear once per application. Neither a high score nor repeated locations alone establish the causal importance of a head.

For display in Figure 5, trajectories begin at step 2000 and use a triangular weighted mean in log-step coordinates. At step t, a measurement at t_j in the same growth stage receives weight proportional to $1 - |\log_2(t_j/t)|/0.125$ when $|\log_2(t_j/t)| < 0.125$, and zero otherwise. Exact growth-step measurements are excluded, and the first and last available measurements of each stage, including the final training scores, remain raw. Smoothing is applied separately to each identified head and never crosses a growth boundary. All trajectories are opaque, and shared ancestral segments are drawn once. Color intensity follows the smoothed PS along each displayed segment, using the same 0–1 scale as the final-head heatmaps. Orange identifies the selected L2H9/L2H10 descendants in LIDAS and the selected heads within the looped block; other heads use blue. Color does not encode final-head rank or FV membership.

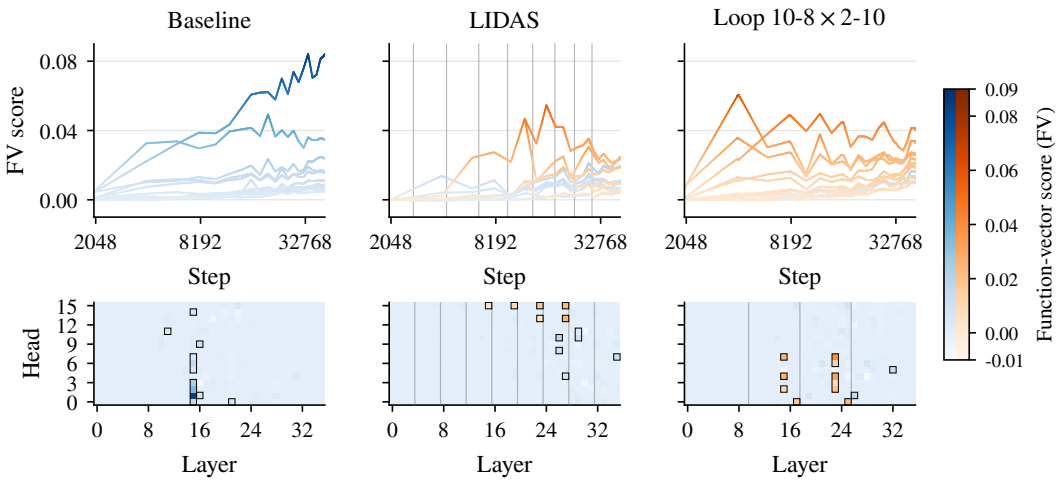


Figure 14: FV heads recur at the grown or looped block size but emerge later than induction heads. FV counterpart of Figure 5 for the 3B models. **Top:** trajectories of the twelve final top-2% FV heads per model, selected independently of PS. In LIDAS, these heads trace to eight ancestors; two, L3H13 and L3H15, account for six final heads (two and four, respectively; orange). **Bottom:** final FV scores for all heads; black outlines mark the selected set. Orange in Loop 10-8×2-10 marks selected heads in the repeated block; other heads are blue. Scores share a common scale that retains negative values. Axes, growth and block guides, lineage tracing, and smoothing follow Figure 5 and this section.

Cross-size PS and FV lineages. Figures 15 and 17 independently select the final top-2% heads by PS and by FV, giving 10, 16, and 12 heads at 360M, 1.7B, and 3B. Each set is then traced by identity through the same inverse duplication maps. The corresponding all-head final score matrices appear in Figures 16 and 18; outlines mark exactly the heads selected for each trajectory plot. The 1.7B MIDAS branch uses its shared LIDAS trunk before step 24,285.

The appendix applies the same within-stage triangular log-step smoother with halfwidth 0.125, preserving stage endpoints and final values. Exact growth-step measurements are omitted except for the first 360M pre-growth checkpoint at step 4722: its explicit pre-growth stage identifies the four-layer source, and it is the only measured point in that stage. Immediate post-growth scores do not enter these lineage curves. The 360M dense and looped curves start at their first measured checkpoints, 10,000 and 5000; no earlier values are imputed. Shared ancestral segments are drawn once. PS uses a common 0–1 range; FV uses a common range across models and sizes that includes all displayed

values. Color intensity follows the local score and does not encode model identity or PS/FV set overlap. No particular seed-head pair is assumed to recur across sizes.

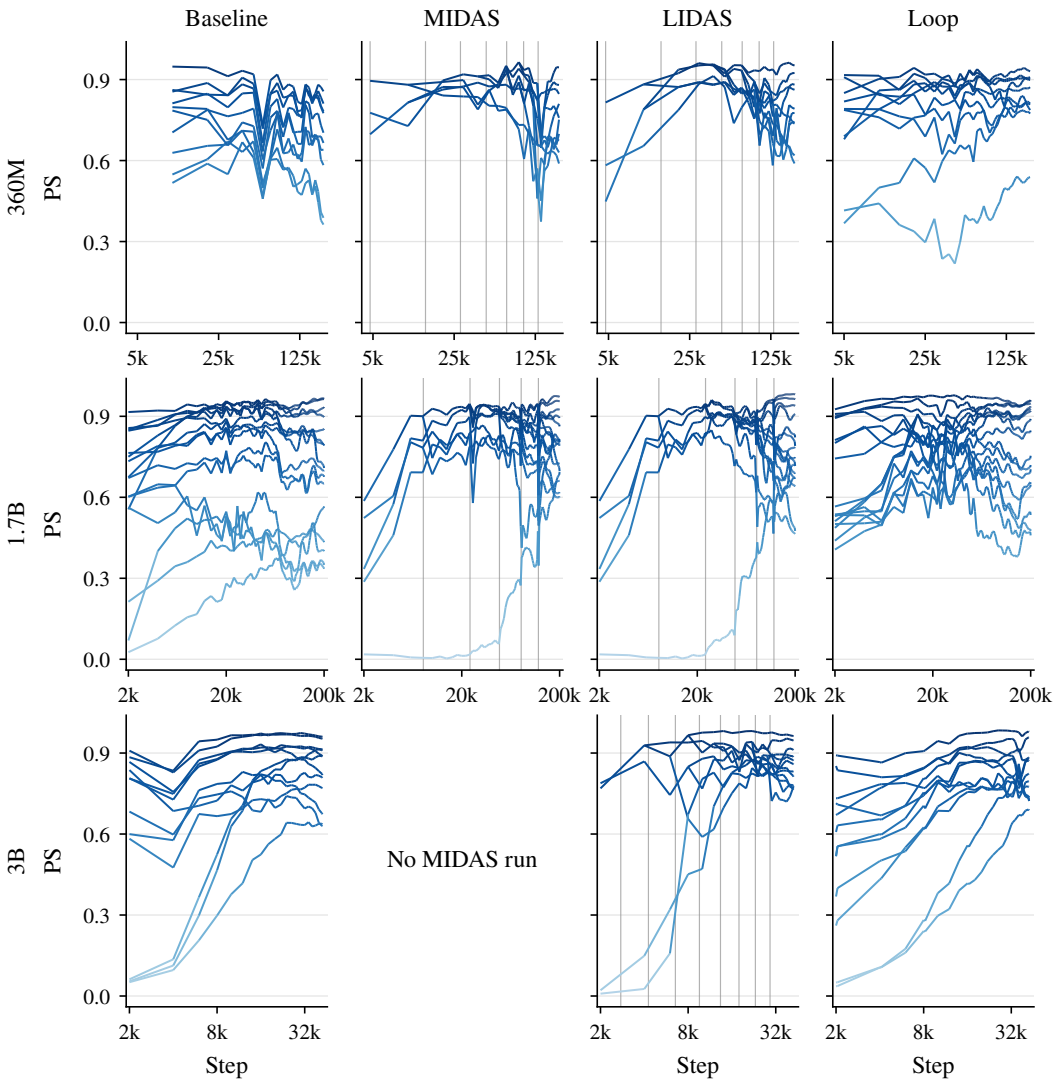


Figure 15: Trajectories of the final top-2% prefix-matching heads. Rows show 360M, 1.7B, and 3B; columns show Baseline, `MIDAS`, `LIDAS`, and Loop. Each panel follows the final 10, 16, or 12 heads ranked by PS, respectively, through their actual growth ancestry; sets are not reselected at intermediate checkpoints. Shared ancestral paths are drawn once, and vertical rules mark growth. Lines use the within-stage smoothing and boundary policy in Appendix E.2; step ticks marked k denote thousands. Scores share a common vertical range. There is no 3B `MIDAS` run. Loop configurations are specified in Appendix A.3.

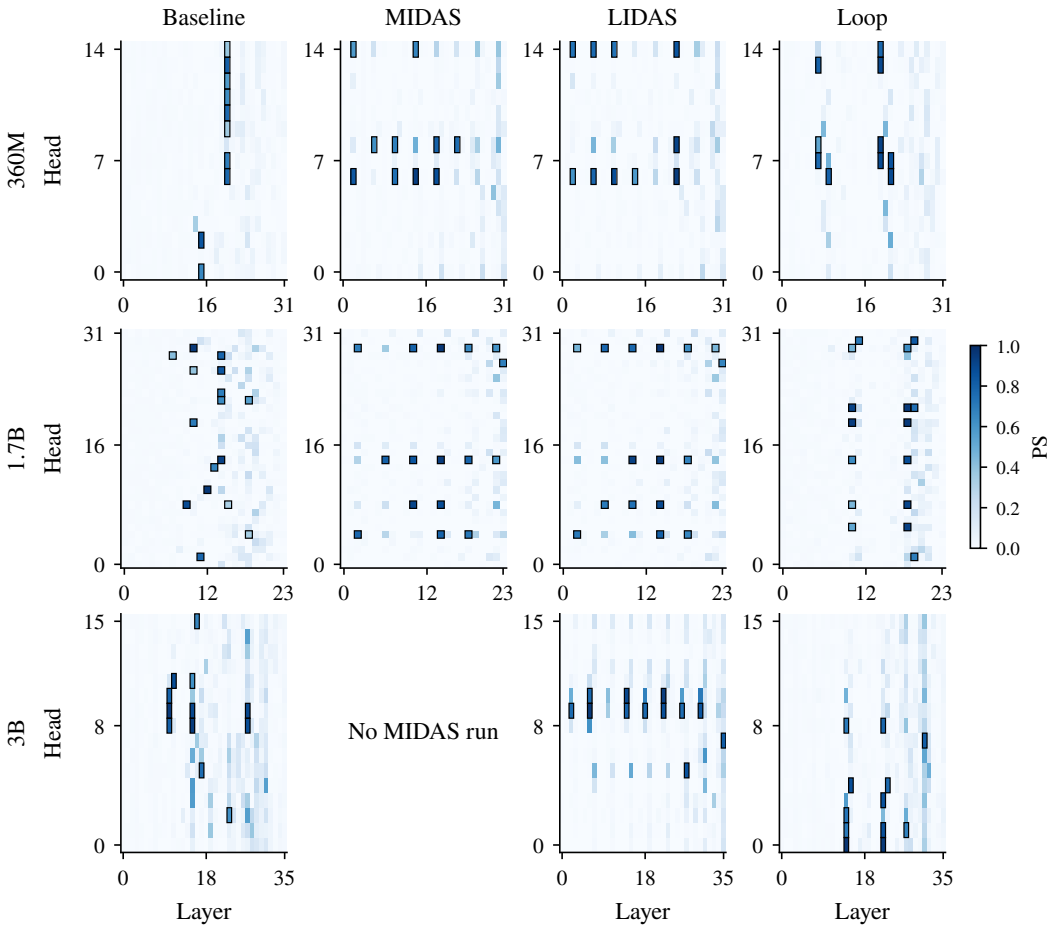


Figure 16: Final prefix-matching scores for every attention head. Rows show 360M, 1.7B, and 3B; columns show Baseline, `MIDAS`, `LIDAS`, and Loop. All panels share one PS color scale. Black outlines identify the final top-2% heads traced in Figure 15. Layer and head indices are zero-based; loop applications occupy separate layer positions. The empty 3B `MIDAS` slot denotes an unavailable model. Loop configurations are specified in Appendix A.3.

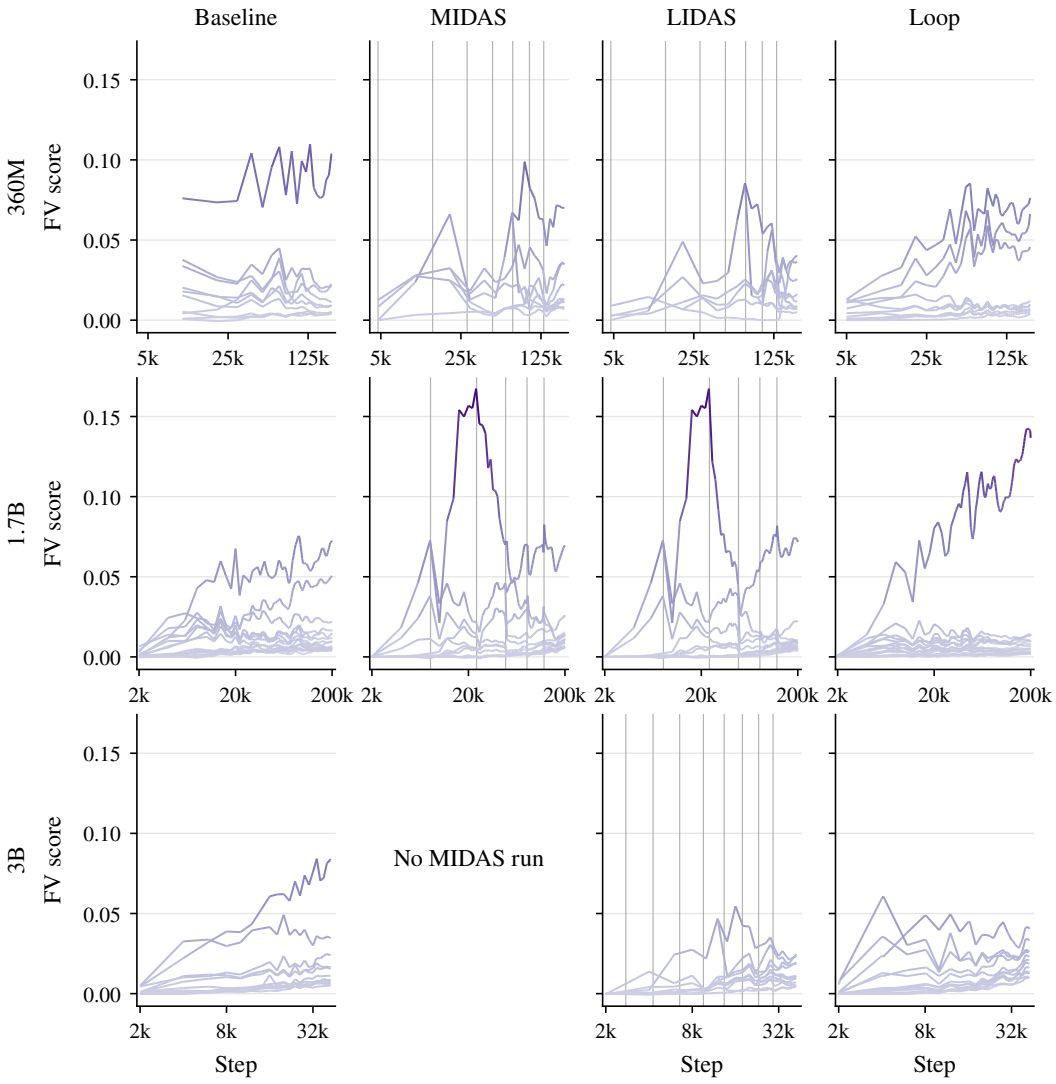


Figure 17: Trajectories of the final top-2% function-vector heads. Rows show 360M, 1.7B, and 3B; columns show Baseline, `MIDAS`, `LIDAS`, and Loop. Each panel follows the final 10, 16, or 12 heads ranked by FV, respectively, through their actual growth ancestry; sets are not reselected at intermediate checkpoints. Shared ancestral paths are drawn once, and vertical rules mark growth. Lines use the within-stage smoothing and boundary policy in Appendix E.2; step ticks marked k denote thousands. Scores share a common vertical range. There is no 3B `MIDAS` run. Loop configurations are specified in Appendix A.3.

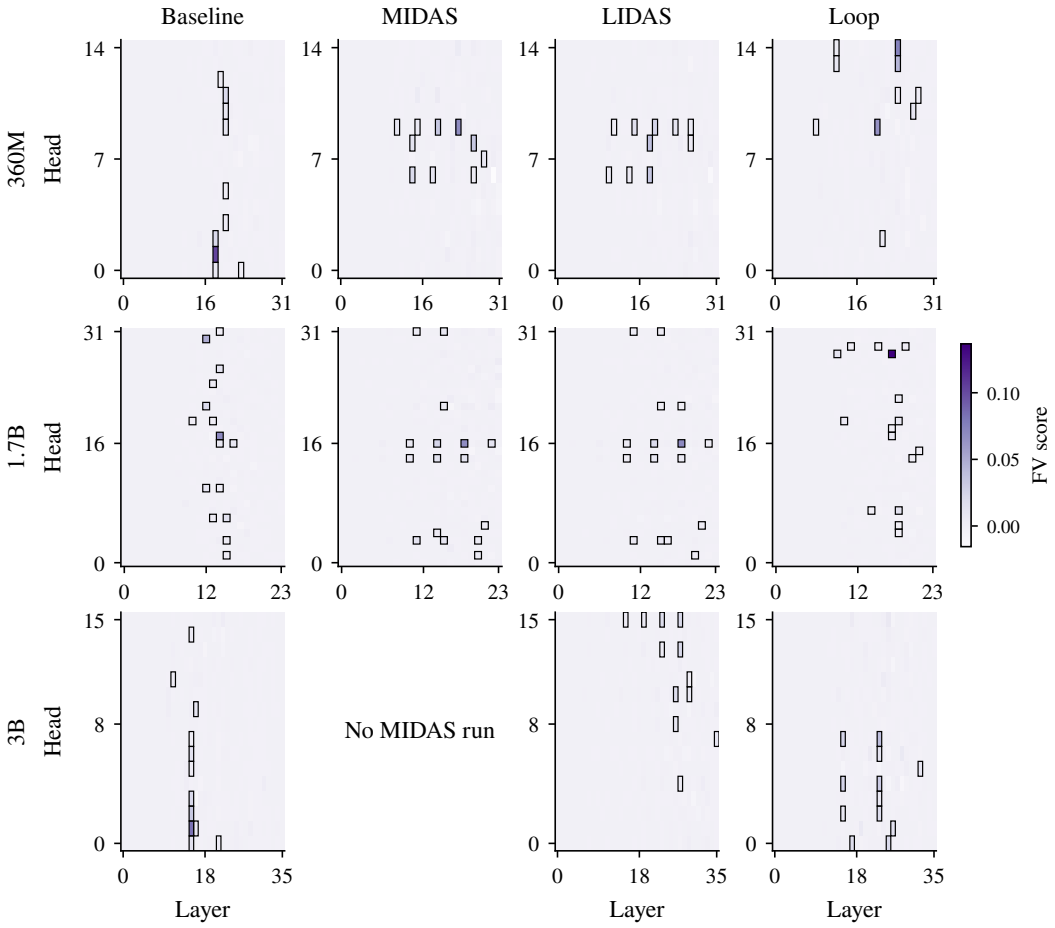


Figure 18: Final function-vector scores for every attention head. Rows show 360M, 1.7B, and 3B; columns show Baseline, `MIDAS`, `LIDAS`, and Loop. All panels share one FV color scale. Black outlines identify the final top-2% heads traced in Figure 17. Layer and head indices are zero-based; loop applications occupy separate layer positions. Negative FV scores, where present, are retained. The empty 3B `MIDAS` slot denotes an unavailable model. Loop configurations are specified in Appendix A.3.

E.3 Per-Layer Readouts

Figure 19 evaluates intermediate predictions at each exit depth l, with the horizontal axis normalized to l/L. For each generated token, we run only the first l layers and decode their residual stream. The logit lens applies the full model's final normalization and unembedding directly. The tuned lens (Belrose et al., 2023) first applies a learned affine residual translator $h \mapsto h + W_l h + b_l$. Translators are initialized at zero, so the initial lens is the identity, and the language model remains frozen.

Each model's translators are trained on its own FineWeb-Edu pretraining source for 250 optimizer steps with 2^{18} tokens per step, minimizing KL divergence from the full model's next-token distribution to the translated intermediate prediction. Training uses SGD with Nesterov momentum 0.9, initial learning rate 0.1 with linear decay, weight decay 10^{-3}, gradient clipping at norm 1, and bf16 autocast. Tokenization and document masking follow the corresponding pretraining setup. Translator index l decodes the residual entering layer l (equivalently, after l completed layers); the full-depth exit uses the original readout without a translator.

We run the same nineteen-task, ten-shot F1 evaluation at each exit depth. An identity-translator (zero-lens) control reproduces the logit lens, and the full-depth exit reproduces the original model. These probes test how accessible predictions are to a fixed or learned readout; they do not by themselves identify which layers causally compute the prediction.

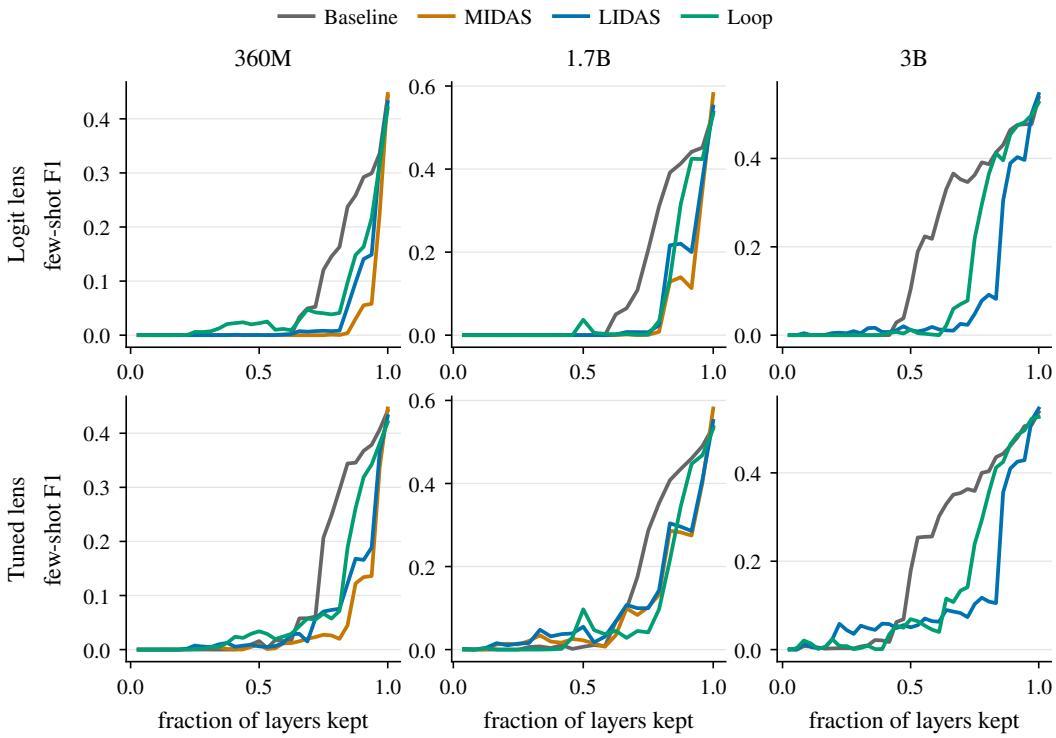


Figure 19: Intermediate few-shot readouts across model sizes. Columns show 360M, 1.7B, and 3B; rows use the logit and tuned lenses. Each curve runs the nineteen-task, ten-shot F1 evaluation after the indicated fraction of layers. The tuned lens is trained separately for each frozen language model by next-token distillation (Appendix E.3). These readouts measure accessibility of predictions, not the causal importance of a layer. Loop configurations are specified in Appendix A.3.

F Attention-Head Ablations

F.1 Head Selection and Intervention Protocol

Figure 6(a,b) mean-ablates fractions $f \in \{0.01, 0.03, 0.05, 0.07, 0.09, 0.15, 0.20\}$ of final-checkpoint head applications. The combined figure uses exclusive PS and FV rankings; additional

sweeps also include non-exclusive and random rankings. Exclusive induction rankings first exclude the top $\lfloor 0.02H \rfloor$ FV heads; exclusive FV rankings analogously exclude the top induction heads. The number ablated remains $\lfloor fH \rfloor$, measured relative to the full set before exclusion. Joint rankings use the geometric mean of PS and FV after clipping each score at zero and dividing by its maximum. A selected head's output slice is replaced by a mean activation immediately before the attention output projection. In looped models, selection and intervention act on individual applications, so the ablated fraction is not a fraction of unique parameters.

For the few-shot readout, the replacement is the clean last-token head activation averaged over the 29 FV training tasks. It is applied at the last position of the prompt and each subsequent generation forward pass. We then repeat the nineteen-task F1 evaluation of Appendix E.1. For token loss, replacement means are computed over the natural-text evaluation documents, and the intervention acts at all positions. The single-position token-loss readout in the additional diagnostics uses 512 predictions per document from 513-token prefixes of the 10,000-document FineWeb-Edu set. The combined figure instead uses the full-position experiments described below for its natural-text panels.

F.2 Few-Shot and Windowed ICL-Score Changes

Figure 6(b,c) uses the full-position natural-text intervention over all 4096 prediction positions of the three 3B models, using bf16 evaluation on 10,000 FineWeb-Edu documents. Panel (b) applies the windowed score in Equation (3) at every ablation fraction. With $\delta_j = \bar{\ell}_j^{(\text{ablated})} - \bar{\ell}_j^{(\text{clean})}$ for the same model and evaluation documents, the plotted change is

$$\Delta S_{\text{ICL}} = \frac{1}{21} \sum_{j=40}^{60} \delta_j - \frac{1}{21} \sum_{j=490}^{510} \delta_j. \tag{7}$$

The raw position exports are zero-based, so these one-based inclusive windows use stored indices 39–59 and 489–509. This calculation uses paired ablated-minus-clean losses before any position binning, with a matched unablated reference from the full-position experiment.

Figure 20 applies the same windowed definition to 360M and 1.7B using their native 2048-position experiments, as well as to 3B. Both head rankings exclude the other measure's top-2% heads. The plot reports paired changes from each model's own unablated reference for all seven fractions, alongside the corresponding changes in nineteen-task F1. The windowed scores are computed from raw positions before binning or display smoothing.

F.3 Position-Wise Ablation Effects

Figure 6(c) fixes $f = 0.09$ and overlays the exclusive induction and exclusive FV rankings. It averages δ_j in logarithmic position bins (24 per decade), then applies Gaussian-weighted local linear smoothing in $\log_2$ position with bandwidth $\sigma = 0.2$ to each of the six curves for display on a base-2 logarithmic axis. This display smoothing does not affect the windowed ICL-Score calculations. Solid lines indicate exclusive induction heads and dashed lines exclusive FV heads throughout the combined figure; dotted lines indicate zero change. Figure 21 shows the uncropped position-wise effects at 7%, 9%, 15%, and 20% ablation for 360M, 1.7B, and 3B.

F.4 Additional Rankings and Single-Position Readout

The additional single-position token-loss diagnostic in Figure 22 follows the implementation's zero-based loss-array indexing:

$$D_{50,500} = \mathbb{E}_{\text{docs}}[\text{NLL}[50] - \text{NLL}[500]] = \bar{\ell}_{51} - \bar{\ell}_{501}, \tag{8}$$

where ℓ uses Equation (1). Thus, the labels "50" and "500" in this diagnostic denote array indices, and the statistic differs from the window-averaged score in Equation (3). The corresponding two anchor losses can also be inspected separately. Dotted horizontal lines give unablated values; random-control bands are mean $\pm$ standard deviation over three random head sets for few-shot F1. Token-loss controls use sixteen head sets at 360M/1.7B and three at 3B in the plotted exports. They measure sensitivity to the choice of ablated heads.

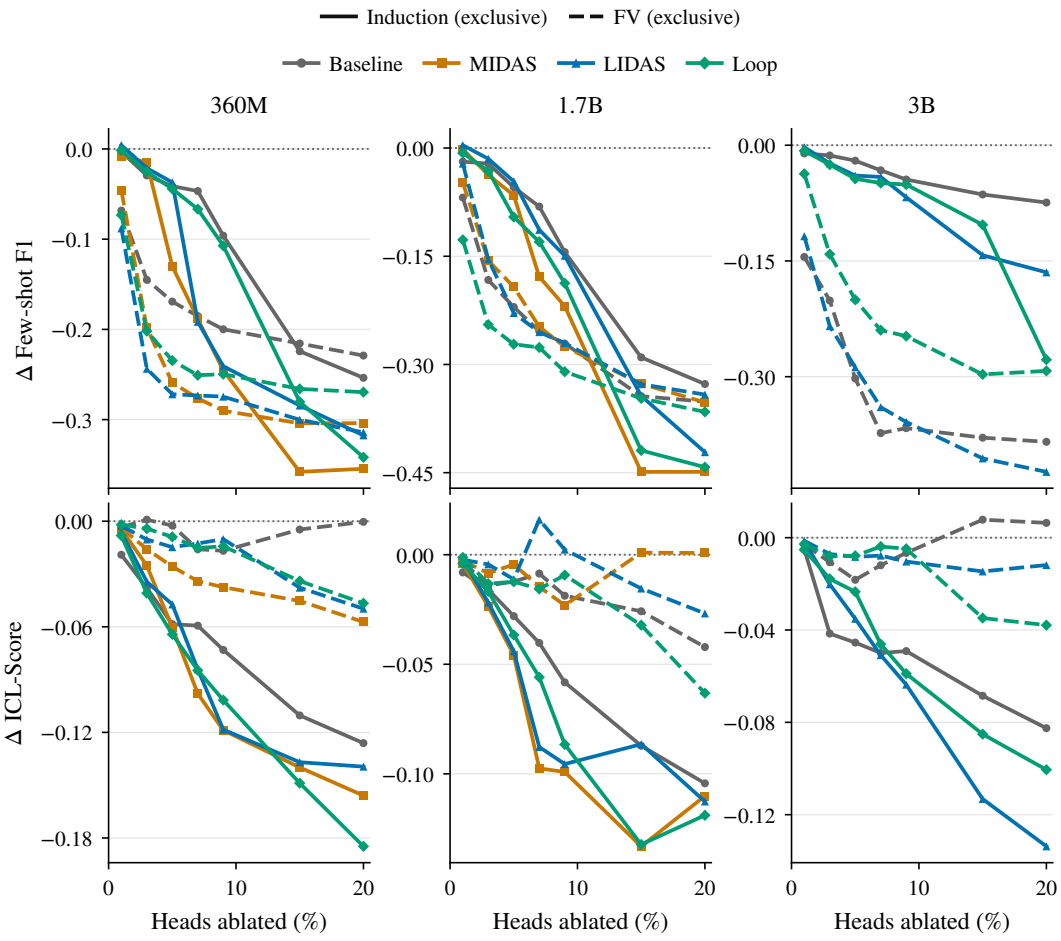


Figure 20: Exclusive head ablations across model sizes. Columns show 360M, 1.7B, and 3B. Top: change in nineteen-task ten-shot F1. Bottom: change in windowed ICL-Score, computed as the mean paired NLL increase at positions 40–60 minus that at 490–510 (Equation (7)). Solid lines ablate induction-ranked heads after excluding the top-2% FV heads; dashed lines reverse these roles. Markers distinguish models. Fractions are 1%, 3%, 5%, 7%, 9%, 15%, and 20%; zero denotes the corresponding unablated result. Native full-context experiments use 10,000 documents; these are distinct from the single-position diagnostic in Figure 22. Loop configurations are specified in Appendix A.3.

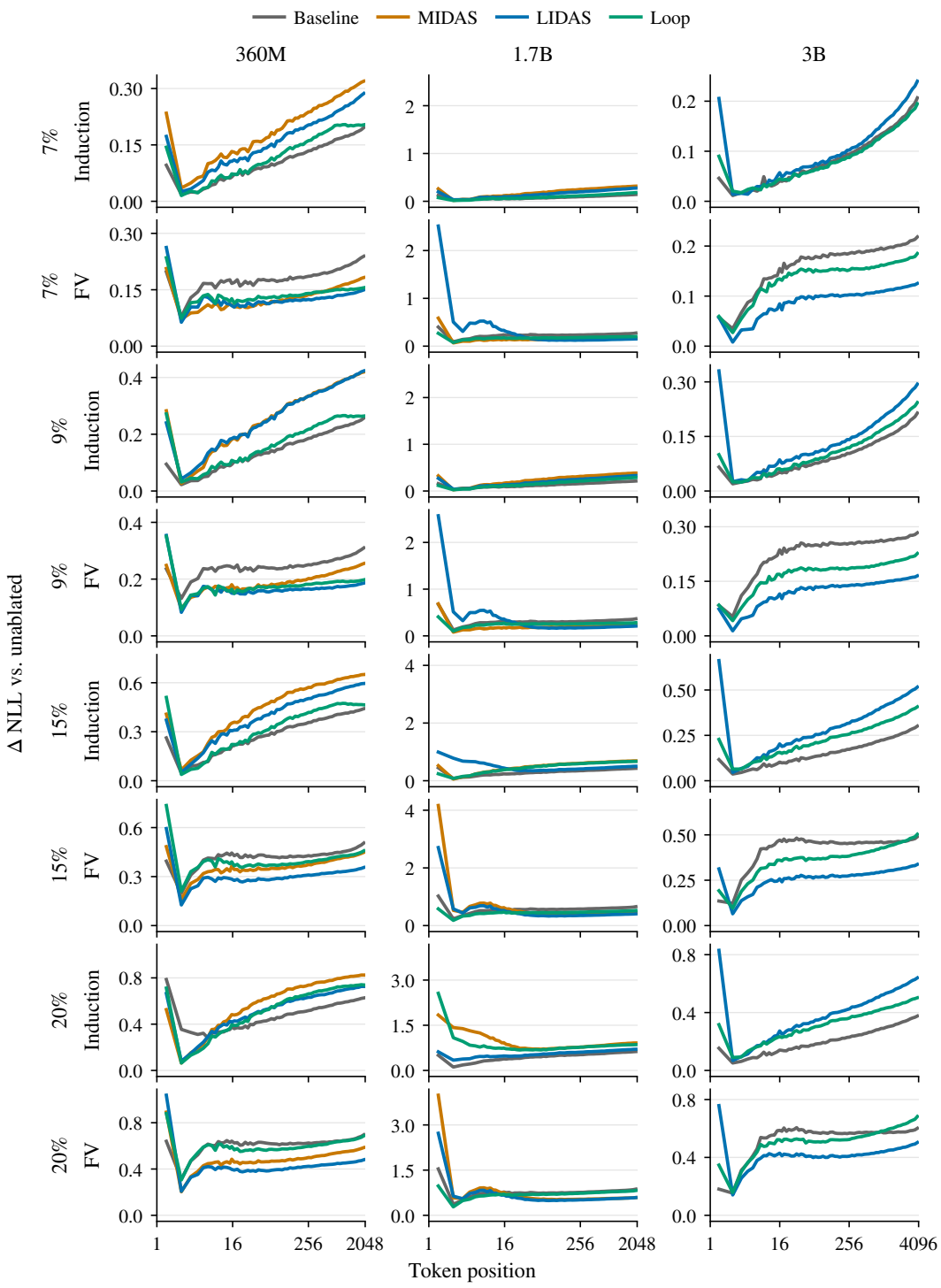


Figure 21: Position-wise head ablation at higher fractions. Columns show 360M, 1.7B, and 3B; row pairs show exclusive induction and FV ablation at 7%, 9%, 15%, and 20%. Curves give the mean NLL increase on 10,000 FineWeb-Edu documents, with log-binned positions on a base-2 axis. Each induction/FV pair within a size shares an uncropped vertical range. `MIDAS` is unavailable at 3B. See Appendix F. Loop configurations are specified in Appendix A.3.

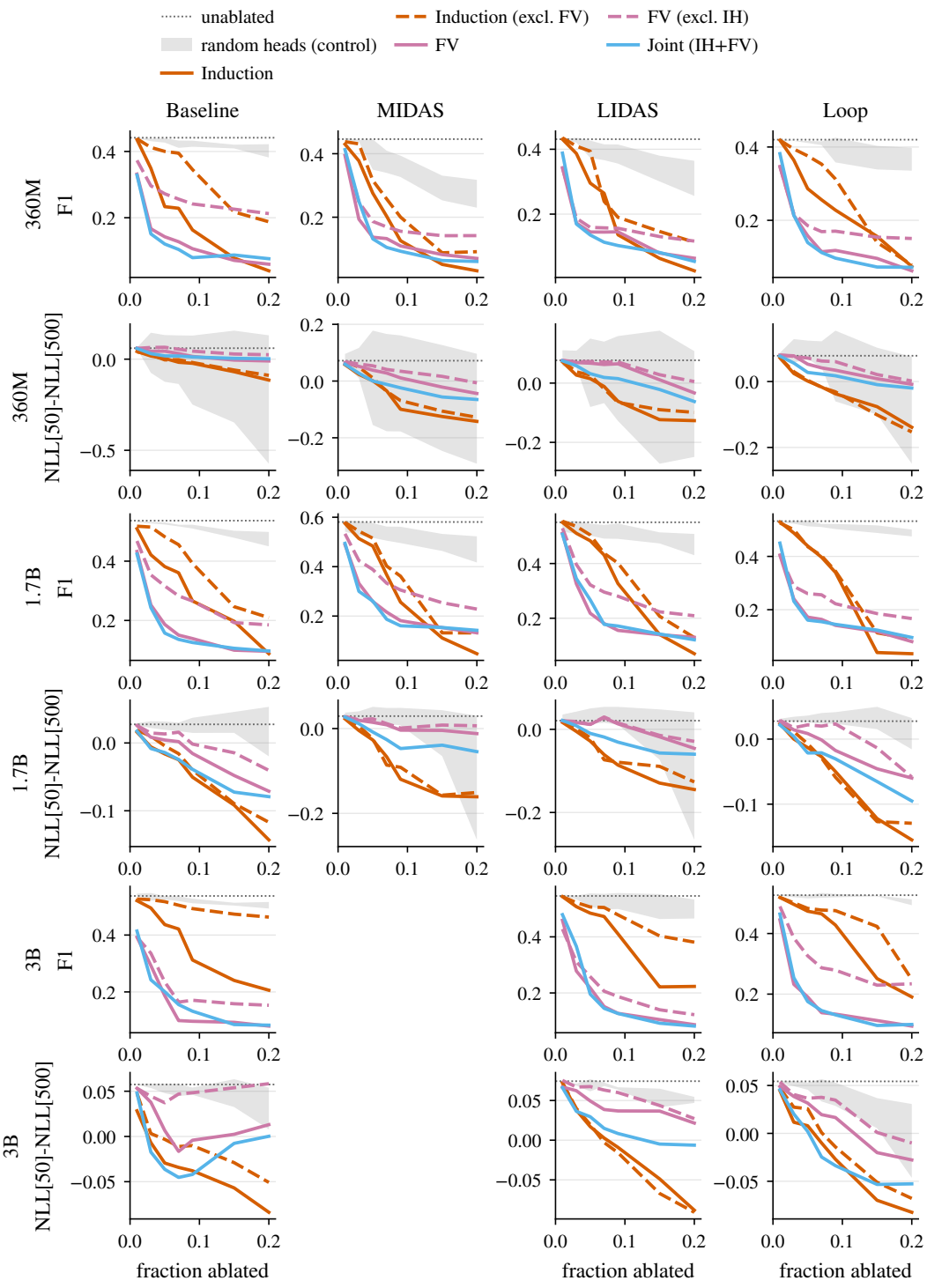


Figure 22: Head ablation across model sizes. Row pairs show 360M, 1.7B, and 3B, each with few-shot F1 above the single-position token-loss difference (Equation (8)); columns show dense, `MIDAS`, `LIDAS`, and looped models. Dashed rankings exclude the other measure's top-2% heads. Gray bands show random-head-set mean ± SD: three seeds for F1, sixteen for 360M/1.7B token loss, and three for 3B token loss. Dotted lines give unablated values. Fractions count applications in looped models. Panel scales vary; the empty 3B `MIDAS` column reflects the missing model. See Appendix F. Loop configurations are specified in Appendix A.3.