SUPERVISION DENSITY SHAPES TRANSFORMER SOLUTIONS AND LONG-CONTEXT GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Which tokens receive supervision is typically treated as an efficiency choice: supervising fewer tokens can reduce training cost while preserving performance. Here, we show that supervision density also determines how a Transformer learns to solve a reasoning task, with direct consequences for long-context generalization. We train 12-layer Transformers on a controlled multi-hop variable-binding task while varying the fraction of suffix tokens contributing to the loss, from standard all-token prediction to last-token supervision, and evaluate generalization to both longer contexts and deeper reasoning chains. We uncover a strikingly non-monotonic relationship between supervision, learning speed, and generalization. Intermediate supervision densities converge substantially faster than either all-token or answer-only training, yet the fastest-converging model is not the best generalizer. At convergence, sparse 25% supervision yields the strongest long-context generalization, whereas prolonged training eventually makes the last-token model the strongest generalizer despite validation accuracy having saturated much earlier. Mechanistic analyses reveal that these learning regimes steer models toward qualitatively different solutions: suffix-supervised models predominantly write the answer through the MLP pathway, whereas last-token supervision produces an attention-dominated solution. Crucially, attention remains causally necessary across all regimes, and the strongest generalizer at convergence is the model that most strongly requires both attention and MLP pathways. Together, our results show that supervision density does more than control training efficiency: it selects among memory- and structure-based solutions, thereby shaping when and how Transformers generalize beyond their training distribution.

1 INTRODUCTION

The last decade has witnessed the birth of pivotal Transformer-based large language models (LLMs), which have demonstrated impressive abilities in solving a plethora of tasks (Achiam et al., 2023; Bubeck et al., 2023). Notably, these abilities emerge when LLMs are scaled up (Brown et al., 2020; Chowdhery et al., 2023; Wei et al., 2022), and scaling laws dictate that LLMs are projected to further improve with more scaling (Kaplan et al., 2020; Henighan et al., 2020; Hoffmann et al., 2022). This dependence on scale becomes even more consequential in the emerging era of agentic AI, where models must effectively harness increasingly rich contextual information and interact with external tools, capabilities whose benefits themselves become more pronounced as model scale increases (Schick et al., 2023).

However, training such scaled models requires massive amounts of data, and consequently, often prohibitive compute power. As such, research has focused on ways to optimize learning by principally pulling on two levers: (i) reduce compute, and thus convergence speed, via using *fewer, more informative* tokens through data selection, pruning, and reweighting (Li et al., 2024; Xie et al., 2023; Marion et al., 2023; Sorscher et al., 2022), and (ii) given a fixed token budget, enabling models to *generalize* beyond the context lengths encountered during training (Kazemnejad et al., 2023; Sun et al., 2023; Press et al., 2021; Ding et al., 2024; Chen et al., 2023; Zhu et al., 2024). Stated as such, these two levers (or optimization goals) seem to be at odds. On the one hand, we aim to reach a similar performance faster with less tokens, and on the other, we strive for better performance with a similar token budget. However, to the best of our knowledge, no study has systematically investigated the interaction between learning regimes (i.e., the fraction of supervised tokens during

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.

learning) and long-context generalization abilities. Such an endeavor becomes particularly relevant in reasoning tasks, where input structure is crucial to solve the task, and thus distinct data reduction regimes can have important repercussions in how a language model learns to solve a task. In turn, a specific learning strategy may impact a model's long-context generalization.

In this work, we take a deep dive in the interaction between learning regimes and long-context generalization, and hypothesize the existence of a trade-off between these two optimization goals. In particular, we predict that a language model with full supervision density (using all tokens for learning) may tend to find solutions that rely more on memory and thus display poorer long-context generalization. In contrast, a language model trained using sparser density, will tend to find solutions that balance memory-based *and* task structure-based learning, thereby displaying stronger long-context generalization. Moreover, this distinction relates directly to a well-known division of roles, in which feed-forward layers act as key–value memories that store lexical content (Geva et al., 2021), while attention routes and structures (and composes) information across positions (Elhage et al., 2021). We therefore make the secondary hypothesis that different learning regimes can steer models toward distinct regions of a solution space defined by attention- and MLP-dominated axes, and predict that models adopting learning strategies that jointly leverage both attention and MLP mechanisms will exhibit stronger generalization. We position and motivate our work with respect to others in the related work section of Appendix A.

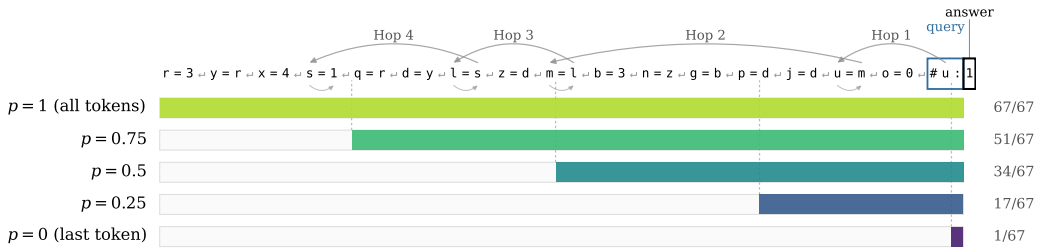


Figure 1: Example of an input sequence with 4 hops. A program is 16 assignment lines `X=Y`, where `Y` is a digit or an earlier variable, followed by a query `#v:`; the answer is the value reached by dereferencing `v`, and the number of dereferences is the hop number. The blue box marks the query and the black box the answer. Supervision densities are represented as horizontal bars indicating which tokens are used to compute the loss. Every bar ends at the query colon, from which the answer is predicted.

We study the interaction between supervision density and long-context generalization in a controlled setting: a multi-hop *variable-binding* task (Figure 1), in which the model assigns values to variables and must later dereference them. The task exposes two clean axes of generalization difficulty, sequence length and hops number (i.e., chained dereferences), which let us separate memorization of surface statistics from the acquisition of transferable structure. Therefore, this task provides a controlled way to evaluate our hypotheses, and has in addition emerged as an increasingly relevant testbed for probing both language and language models (Wu et al., 2025; Henderson, 2020; Chen et al., 2021; Feng & Steinhardt, 2024). Altogether, our contributions are as follows:

- We propose a multi-hop variable-binding task that allows to parametrically control the amount of supervised tokens, without inherently altering task structure.
- We demonstrate that dense supervision yields learning strategies that are MLP-dominated, whereas sparse supervision yields attention-dominated solutions. Read through this lens, we show that supervision density is effectively a dial on the trade-off between memory-based (lexical capacity) and structure-based (algorithmic, reusable computation) learning.
- We reveal that this dial, in fact, steers the position of models onto a solution space whose axes are defined by attention-based and MLP-based learning strategies, and that this space underlies a *non-monotonic* trade-off. Convergence is fastest under intermediate suffix supervision (50–75%), while generalization peaks under sparser supervision (25%). Strikingly, the standard all-tokens learning regime displays neither the fastest convergence nor the best long-context generalization. Moreover, training post convergence steers attention-

dominated models to spaces that further promote generalization, revealing a non-symmetric relationship between MLP-attention balance and generalization.

- To gain a deeper understanding of the underlying mechanisms, we carry out attention flow and internal representation analyses. We find that attention-dominated (compared with MLP-dominated) models generate (i) information passing through the hop chain that reflects in a faithful way the structure of the binding task, and (ii) that the attention component in the residual block produces more structured representations.
- We argue that density supervision and long-context generalization cannot be seen as independent concepts. Supervision density influences how a language model finds a solution to a reasoning task, in turn impacting the models' ability to perform long-context generalization.

2 Experimental Setup

Task and data. We study the variable binding task of Wu et al. (2025), where a program is a sequence of $\texttt{ctx} = 16$ assignment lines of the fixed form `X=Y\n` followed by a query `#v:`, e.g., `h=5\nq=h\nb=2\n...#q:`. Names `X` are single letters from a 26-symbol alphabet; `Y` is one of 10 digit leaves or an earlier variable. The query asks for the value reached by dereferencing `v`, and its hop depth is the number of dereferences to a leaf. In the example, `q` resolves through `h` to the leaf 5, so its depth is 2. Two constraints make every program well posed. A variable may only appear on the right-hand side once it has been assigned, so every chain ends at a leaf and hop depth is well defined; and no name is assigned twice, so each variable has a single value and the query has a unique answer. Training uses $\texttt{ctx} = 16$ and depths $\{1, 2, 3, 4\}$, balanced. A character-level tokenizer gives 4 tokens per line, so structure must be recovered from raw characters, and the answer is predicted at the `":"` token (index -2). Splits are deterministic given a seed (450,000 training, 1,000 validation, and 49,000 test programs) and are the same in every run. All models are trained, validated, and tested on exactly the same programs. The only difference between models is the supervision density (see Appendix B.1 for more details on the task).

Models and training. All models are GPT-J decoders: 12 layers, 1 head per layer, $d_{\text{h}} = 512$, $d_{\text{ff}} = 2048$, 37.8M parameters, dropout 0.1, and Rotary Positional Encodings (Su et al., 2024). The parallel residual block, $h_{\ell+1} = h_\ell + \text{attn}(\text{LN}(h_\ell)) + \text{MLP}(\text{LN}(h_\ell))$, allows to assess the direct contributions of the attention and MLP to the model's answer, and the single head structure generates only one attention pattern per layer, and is thus a more interpretable setting. The variable of interest is *supervision density* p, the fraction of tokens that contribute to the loss, applied as a suffix of the sequence. Density supervision is parametrically modulated from next-token prediction over the whole sequence ($p = 1$) to the last token alone ($p = 0$), with $p = 0.75$, $p = 0.5$ and $p = 0.25$ between these extreme density values. The mask grows backwards from the last token, which leaves the structure of the task intact: the query and its answer are supervised at every density, and the densities differ only in how much of the program is added. The data, the tokenization and the splits are identical across models, and p only controls the loss mask, so any difference between them is attributable to the supervision density. We use the validation split to select the checkpoint at which the model reaches maximal accuracy (100%) for the first time; from here on termed *first maximal accuracy* model (see Appendices B.2 and B.3 for more model and training details, respectively).

Long-context generalization. To evaluate long-context generalization we build novel programs composed of longer inputs sequences and more hops. This generalization set crosses every new $\texttt{ctx} \in \{18, 20, \ldots, 26\}$ with every new hop values $\in \{5, \ldots, 8\}$, leading to 20 combinations with 12,250 sequences each. The alphabet, grammar and line format are those of training, thus only length and hops vary. Examples and more information with respect to the generalization dataset can be found in Appendix B.4.

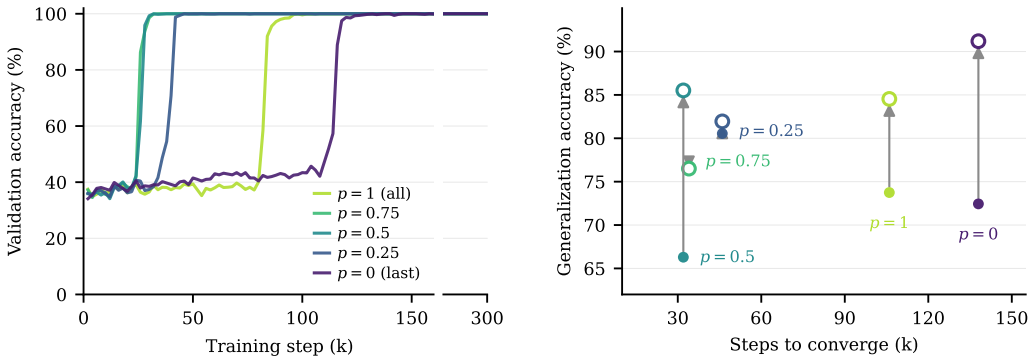
Metrics and Analysis. We base our results on the following metrics. Learning is assessed using **answer accuracy**, the proportion of correct retrievals in the multi-hop variable binding task (i.e., answering the expected digit). We compare accuracies between densities using a paired McNemar test (Dietterich, 1998), Bonferroni corrected across combinations. To evaluate learning solutions, we assess **direct contributions** of the attention and MLP to the model answer by exploiting the

decomposition of the parallel residual block. To investigate which pathway (in the residual block) is used by the model, we perform **mean ablations**. To assess at which depth the layer the model "commits" to an answer, we use the **logit lens**, i.e., the top-1 token at every layer and position in the sequence. Lastly, to compare the structure of internal representations between models we visualize a **2D t-SNE projection** of the principal components predictive of constant numbers and variable names for each residual block components following Wu et al. (2025). We provide formal descriptions of these metrics in Appendix B.5.

3 Results

3.1 Learning speed and generalization

Supervision density and learning speed. The effect of density on learning speed is non-monotone (Figure 2a; see Figure 8 for learning curves decomposed by hop number). The models trained with $p = 0.5$ and $p = 0.75$ reach 100% validation accuracy first, at 32k and 34k steps, followed by $p = 0.25$ at 46k (see Table 2 for results on the held out test set). The standard all-tokens objective ($p = 1$) needs 106k steps, and last-token supervision ($p = 0$) is the slowest, at 138k. Counter-intuitively, our results show that supervising more tokens is not strictly related to faster learning. In fact, we observe an inverse U-shaped relation whereby extreme supervision densities are associated with slower learning, and intermediate ones tend to produce faster convergence.



(a) Validation accuracy during training.

(b) Speed versus generalization.

Figure 2: Effect of supervision density on convergence and generalization. (a) Validation accuracy during training; intermediate densities converge fastest; the x axis is cut between 160k and 280k, where every curve is flat. (b) Steps to reach 100% validation accuracy against long-context generalization accuracy at the first maximal accuracy checkpoint (filled), with an arrow to the same model at step 300k (hollow).

Supervision density and long-context generalization. Figure 3 shows the relation between supervision density and long context generalization, both in terms of input length (top graphs) and hop number (bottom graphs), at the first maximal accuracy model (left graphs) and with continued learning (300k steps, right graphs). Note that each point averages the other dimension (non-averaged values for each combination are reported in Figure 9 and 10). At first maximal accuracy, long-context accuracy decreases with both context length and hop number (Figure 3), but the rate of degradation depends strongly on supervision density. At first maximal accuracy, $p = 0.25$ generalizes best overall, whereas $p = 0.5$ and $p = 0$ degrade most sharply. Dense supervision is not consistently more robust: $p = 1$ is never the strongest model at any tested length hop depth. Thus, both for input length and hop number, we observe that generalization is not strictly related to more density supervision.

Speed versus generalization and impact of longer training. Learning speed does not predict long-context generalization (Figure 2b). At first maximal accuracy (filled circles), $p = 0.25$ and $p = 0.75$ are fast and strong generalizers, $p = 0.5$ is fastest but the weakest generalizer, and $p = 1$ and $p = 0$ are the slowest with intermediate generalization performance. Strikingly, continued

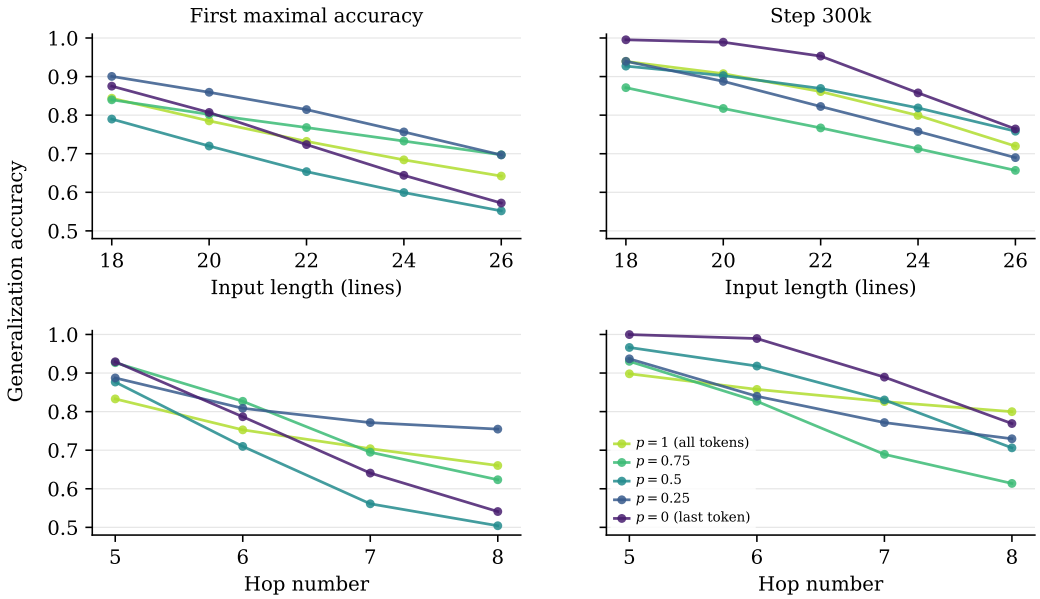


Figure 3: Long-context generalization accuracy at the first maximal accuracy model (left) and at step 300k (right); color codes for the supervision density p. The top row plots accuracy against input length, averaged over hop number 5 to 8. The bottom row plots it against hop number, averaged over input lengths 18 to 26.

training up to 300k steps substantially changes this ordering (open circles). In particular, the initially $p = 0$ model becomes now the best generalizer, i.e., last-token supervision ultimately reaches the highest long-context accuracy both for input length and hop number (right graphs in Figure 3).

In what follows, we aim to relate these empirical observations to differences in how models with distinct supervision density solve the task. We do so by: (i) evaluating the relative contribution (and necessities) of the MLP and attention to solving the task, (ii) assessing how this contribution evolves with longer training, (iii) investigating attention flows and their relation to task structure, and (iv) studying how information is represented in the network.

3.2 The solution space: attention versus MLP

Figure 4 depicts the direct contributions (at first maximal accuracy) to the model's answer of the MLP (purple line) and attention (red line), and the full residual block (black line), for each layers and all supervision density regimes. Dense supervision ($p = 1$) steers models towards MLP-based solutions. Sparse supervision regime induces attention-based solutions ($p = 0$), and intermediate supervisions recruit both pathways ($p = 0.75$ to $p = 0.5$). Note that for all supervision regimes, this trade-off is visible at later layers. Figure 11 shows this trade-off as a function of learning steps. More learning enhances the dominance of a given solution over the other one.

To complement these analyses, we assessed the cosine similarity between the answer and the representation of each component in the residual block as a function of the training step. Figure 12 shows that, for sparse supervision ($p = 0$), representations of the attention pathway align best with that of the answer as training unfolds. The same is true for dense supervision ($p = 1$), but for the MLP pathway.

Importantly, visualizing the evolution of each supervision density model on the MLP-attention contribution plane provides interesting insights (Figure 5). We will first focus on the cluster of models with overall low attention contributions ($p = 1$ to $p = 0.25$); containing the best performing model at first maximal accuracy, i.e., $p = 0.25$. Compared to the others, the latter learns to rely earlier on attention, before reaching its first maximal value related to a big jump in the MLP dimension. In fact, at first maximal accuracy, $p = 0.25$ displays the highest MLP value of all models. Moreover,

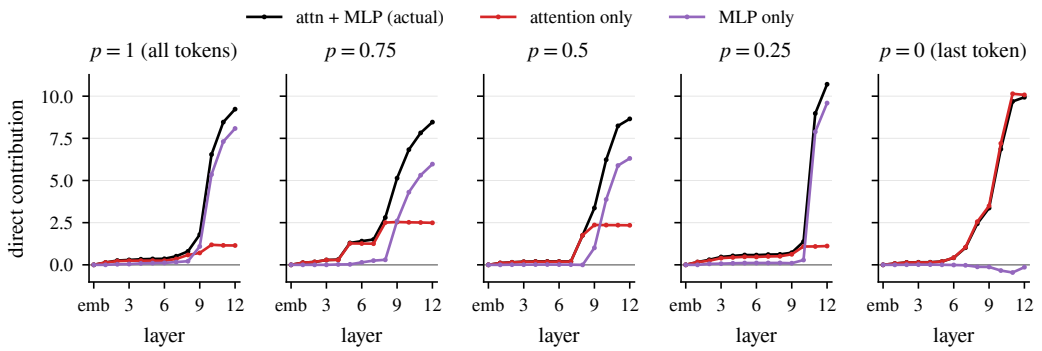


Figure 4: Direct contribution of each pathway to the answer gap in the first maximal accuracy model (readout: logit of the correct digit minus the mean logit of the other nine), accumulated layer by layer, for the full model (black), attention (red) and MLP (purple). The parallel residual makes the two an additive decomposition of the black curve.

we can see that solving the task requires at least a minimum level of attention contribution, which is reached much more quickly for the partly supervised models. Indeed, the $p = 1$ model requires several thousands of learning steps prior involving the MLP dimension and reaching its first maximal accuracy model. Another important point is that generalization performance (at first maximal accuracy) seems to depend on how the MLP and attention contributions are balanced. Although $p = 0.25$ and $p = 1$ have similar attention contributions ($d_{\text{attn}} \approx 1$), $p = 0.25$ has a stronger MLP contribution and generalizes better (81% versus 74%). Second, we focus on the last-token learning model ($p = 0$). This model reaches its first maximal accuracy with no contribution of the MLP and poor generalization performance. However, continued learning allows it to acquire some MLP contribution (Figure 13), which in turn is linked to the strongest generalization performance out of all models. Thus, our results seem to indicate that the MLP-attention trade-off is not symmetric when it comes to generalization robustness: A lot of attention with a little MLP outperforms the opposite (we return to the implications of this point in section 4).

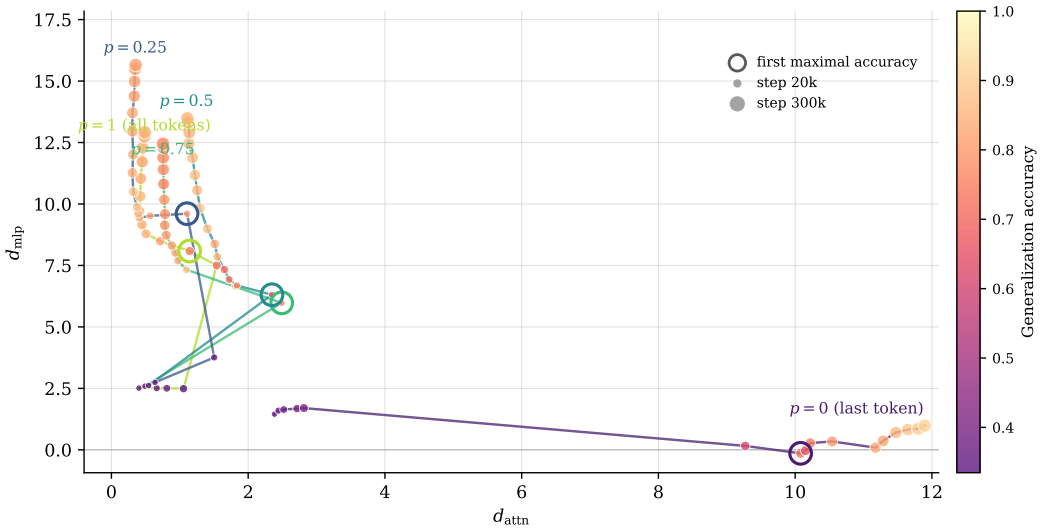


Figure 5: Each run's trajectory through the $(d_{\text{attn}}, d_{\text{mlp}})$ plane. Each point represents a checkpoint from step 20k to step 300k; color is mean generalization accuracy. The validation-selected checkpoint is ringed in the model's own color and sits on the trajectory at its own step, not at its end. d_{attn} and d_{mlp} sum each pathway's direct contribution to the answer gap over all twelve layers, measured in distribution.

3.3 ATTENTION IS WHAT YOU NEED FOR LONG-CONTEXT GENERALIZATION

Direct contributions show which pathway writes the answer, it does not evaluate the necessity of each component in the residual block to solve the task. Evaluating necessity seems particularly relevant in light of the results observed in the previous section. Indeed, small contributions (or contributions change) have substantive impacts for generalization. Table 1 shows the effects of MLP or attention ablation on long-context generalization for last training checkpoint (300k; see Table 3 for the first maximal accuracy model). We can see that, even in the cases that attention has small contributions, removing it has a much more catastrophic effect compared to removing the MLP. These results are in line with our prediction: Long-context generalization requires task-structure learning (on top of memory-based learning), subtended by attention.

Table 1: Long-context accuracy (%) after mean-ablating attention (−attn) or MLP (−MLP) in layers 7–12 at continued learning (300k steps). It is measured on the 20 combinations that are longer and deeper than training at once, evaluated on a fixed subsample of 2,000 of the 12,250 programs of each combination. Chance level is 10%.

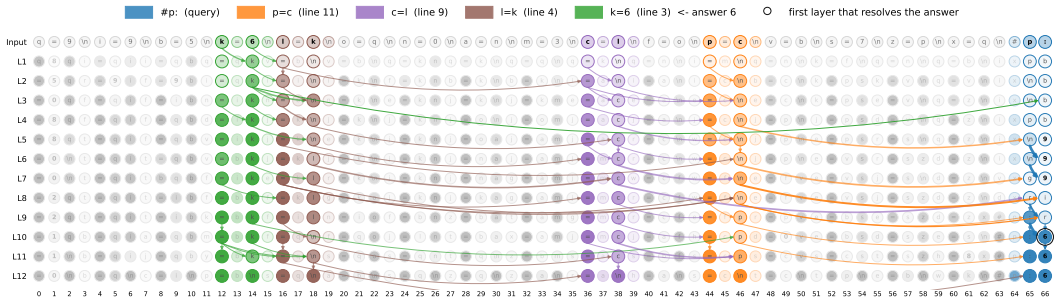
			input length		hop number	
p		overall	lines 18–20	lines 22–26	hops 5–6	hops 7–8
1 (all tokens)	**intact**	**84.4**	**92.4**	**79.0**	**87.5**	**81.3**
	−attn	17.6	17.1	18.0	17.4	17.9
	−MLP	43.4	49.1	39.6	43.7	43.0
0.75	**intact**	**76.6**	**84.5**	**71.3**	**87.9**	**65.3**
	−attn	12.4	12.7	12.2	12.7	12.1
	−MLP	52.6	64.5	44.6	64.1	41.1
0.5	**intact**	**85.4**	**91.2**	**81.5**	**94.2**	**76.6**
	−attn	15.9	17.1	15.1	15.8	16.0
	−MLP	68.5	75.7	63.7	77.3	59.6
0.25	**intact**	**81.8**	**91.2**	**75.5**	**88.6**	**75.0**
	−attn	21.6	23.3	20.5	20.9	22.3
	−MLP	34.1	41.3	29.2	35.8	32.3
0 (last token)	**intact**	**91.2**	**99.2**	**85.9**	**99.4**	**83.0**
	−attn	24.1	27.1	22.2	24.4	23.9
	−MLP	56.1	65.0	50.1	80.3	31.8

Why is attention so crucial for generalization? We aim at gaining a deeper understanding for the differences in generalization between models that are attention-dominated ($p = 0$) versus those that are MLP-dominated (e.g., $p = 1$). As such, we analyzed attention flows (Figure 6), for both cases; see Figure 14 for all supervision densities). We observe that attention patterns in MLP-dominated solutions generate information flows that tend to break the logical structure of the task, whereby information flow skips specific hops positions (top graph in Figure 6). In contrast, attention-dominated solutions are much more faithful to the underlying task structure, and information flow follows the logical hop sequence (bottom graph in Figure 6).

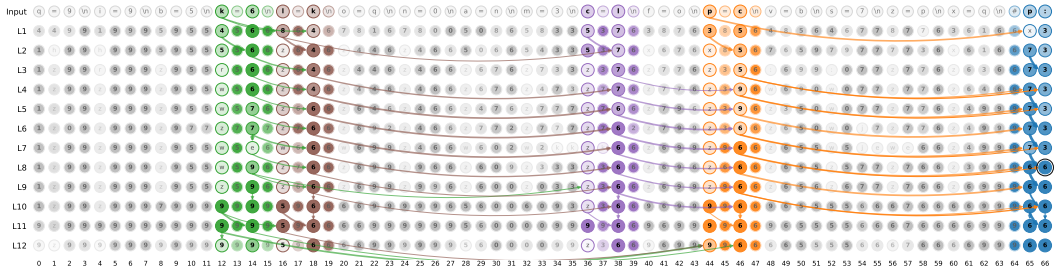
To complement these analyses, using the logit lens, we focused on evaluating differences between solutions types (i.e., MLP or attention-dominated) with respect to the depth at which the model solved the task. Figure 15 shows the top-1 token representation at each layer for all supervision densities. As can be seen, MLP-dominated solutions ($p = 1$), that break task-structure, predict the program format in the first layers and decode their targets early, while attention-dominated ones ($p = 0$), initially predict digits at every position with low confidence, and increase their confidence in later layers. Such confidence gradient is to be expected since the $p = 0$ model does not receive loss information on tokens that structure the task, and forces that knowledge to be extracted via the attention mechanism.

3.4 SUBSPACES OF VARIABLES AND NUMBERS

Following Wu et al. (2025), we evaluate how the internal representations of each residual block components (MLP and attention) are being structured, both for numerical constants and variable names. Given our observation that the attention-dominated model ($p = 0$) generated the best generaliza-



(a) $p = 1$ (all tokens)



(b) $p = 0$ (last token)

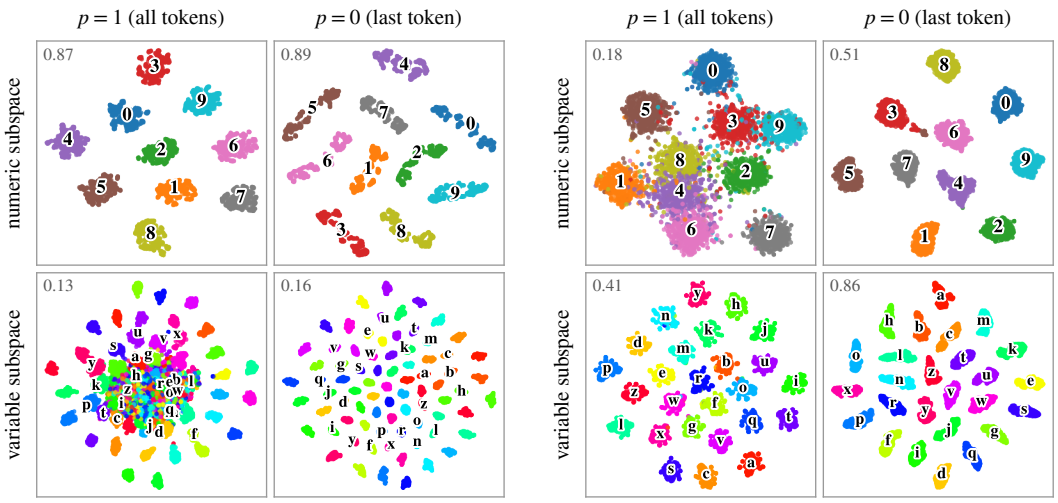
Figure 6: Information flow on one depth-4 query for models $p = 1$ and $p = 0$ with continued learning (300k steps). Each cell is the top-1 logit-lens token at that (layer, position); colored columns are the four lines of the resolution chain, and the circle marks the first layer at which the readout settles on the answer.

tion performance, we hypothesized that (compared with other models) the subspaces representing numerical constants and variable names would present a more favorable structure (i.e., stronger separation between the classes). Furthermore, given that ablations results showed stronger penalization for the attention over the MLP component, we predicted that this representational structure advantage would mostly be seen at the attention component level. Figure 7 depicts 2D t-SNEs (layer 6; see Appendix C.3 for layers 2 and 12 and all supervision densities) of the MLP (left graphs) and attention (right graphs) representations for numerical constants (top graphs) and variables names (bottom graphs). In line with our predictions, we can see that class separation in subspaces is stronger for the attention-dominated model ($p = 0$) compared with the MLP-dominated one ($p = 1$), and that this representational structure advantage is confined to the attention component of the residual parallel block. Further note that this advantage lies both for numerical constants and variable names.

4 Discussion and Conclusion

Our results suggest that where gradient is applied is itself an architectural-level training choice. Although supervision density does not change the model architecture, data, or task, it systematically changes the computational solution discovered by optimization. Dense supervision provides learning signals at many locally predictable positions, creating opportunities for the model to solve parts of the objective through token-specific representations and MLP-mediated memory. In contrast, when supervision is concentrated near the final answer, successful prediction requires information to be routed across distant positions, favoring attention-based computation that more closely tracks the latent structure of the task. Thus, token-level loss masking should not be viewed merely as a means of reducing training compute: it provides a mechanism for steering optimization toward different internal algorithms. The fact that intermediate supervision can converge several times faster than standard all-token or last-token training further shows that more supervision is not necessarily more informative supervision.

Our results also bear implications over how training objectives should be designed. Rather than asking only which tokens are informative enough to train on (Lin et al., 2024; Bal et al., 2025;



(a) MLP (b) Attention

Figure 7: Numeric and variable subspaces at layer 6 for $p = 1$ and $p = 0$ models with continued learning (300k steps) on a subsample of 5,000 test programs. Following Wu et al. (2025), we perform a PCA to 256 dimensions, L_1-regularized probes to select the components predictive of the 10 constants and the 26 variable names, and then plot the 2D t-SNE map. Values on the top left corners are silhouette coefficients.

Wang et al., 2026), token-selection methods could ask which type of solution a token's gradient incentivizes. In reasoning and long-context tasks, supervision might therefore be deliberately allocated to positions that cannot be predicted from local statistics alone, forcing the model to construct and route task-relevant representations. Our results further suggest that neither pure structural computation nor memory alone is necessarily sufficient: at convergence, the strongest generalization arises when attention-mediated routing is accompanied by substantial MLP contribution, whereas prolonged last-token training eventually develops this complementary contribution and achieves the strongest generalization overall. This motivates the possibility of a new class of mechanism-aware supervision schedules in which supervision density is annealed or dynamically selected during training to first induce structural circuits and subsequently consolidate useful content into complementary pathways.

Furthermore, our findings challenge the use of in-distribution validation accuracy as a sufficient stopping criterion. Models with identical saturated validation performance continue to reorganize internally and can acquire substantially better long-context generalization long after apparent convergence (Power et al., 2022; Wang et al., 2024a); most strikingly, the last-token model eventually becomes the strongest generalizer. For models expected to reason outside their training regime, convergence of behavior need not imply convergence of computation. Model selection may therefore benefit from complementary criteria that probe structural generalization or the emergence of reusable computational pathways, rather than stopping as soon as the training distribution has been mastered. In this view, supervision density, training duration, and stopping criteria form a coupled design problem: together they determine not merely whether a Transformer learns a task, but which algorithm it learns to solve it.

Finally, we present some limitations of our study. We deliberately focus on a controlled setting, small, single-head Transformers and a synthetic variable-binding task, to isolate the effect of supervision density and obtain a mechanistic account of the solutions learned by the model. This level of control comes at the cost of breadth: it remains to be established how these dynamics scale to larger, multi-head architectures and more naturalistic reasoning tasks. Likewise, our interventions operate at the level of entire attention and MLP pathways, providing a first causal characterization of their respective roles but not yet resolving the finer circuits through which they interact. We therefore view these results as an important mechanistic starting point: a tractable setting in which supervision can be linked to learned computation, providing hypotheses that can be tested at scale.

AI USE STATEMENT

Generative AI tools were used to support coding and debugging, assist with error checking, and help refine and compress the writing to meet page-limit constraints. They were also used for language editing and manuscript refinement. All scientific content, interpretations, and final decisions remained under the authors' responsibility.

REPRODUCIBILITY STATEMENT

The code and materials required to reproduce the experiments and analyses presented in this work are available at: `https://anonymous.4open.science/r/variable-binding-supervision-density`

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Melis Ilayda Bal, Volkan Cevher, and Michael Muehlebach. ESLM: Risk-averse selective language modeling for efficient pretraining. *arXiv preprint arXiv:2505.19893*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

Catherine Chen, Qihong Lu, Andre Beukers, Christopher Baldassano, and Kenneth A Norman. Learning to perform role-filler binding with schematic knowledge. *PeerJ*, 9:e11046, 2021.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240):1–113, 2023.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581/.

Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*, 2024.

Yihe Dong, Lorenzo Noci, Mikhail Khodak, and Mufan Li. Is random attention sufficient for sequence modeling? disentangling trainable components in the transformer, 2025. URL https://arxiv.org/abs/2506.01115.

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. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Harshwardhan Fartale, Ashish Kattamuri, Rahul Raja, Arpita Vats, Ishita Prasad, and Akshata Kishore Moharir. Disentangling recall and reasoning in transformer models through layer-wise attention and activation analysis, 2026. URL https://arxiv.org/abs/2510.03366.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *International Conference on Learning Representations*, volume 2024, pp. 36391–36413, 2024.

Luyu Gao, Yunyi Zhang, and Jamie Callan. ACER: Automatic language model context extension via retrieval. *arXiv preprint arXiv:2410.09141*, 2024. URL https://arxiv.org/abs/2410.09141.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7376–7399, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.366. URL https://aclanthology.org/2025.acl-long.366/.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3/.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751/.

James Henderson. The unstoppable rise of computational linguistics in deep learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6294–6306, 2020.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Yanping Li, Zhening Liu, Zijian Li, Zehong Lin, and Jun Zhang. Token-level data selection for safe LLM fine-tuning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=k7ytptAaDN.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Not all tokens are what you need for pretraining. In *Advances in Neural Information Processing Systems*, volume 37, pp. 29029–29063, 2024.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

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. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Jinlong Pang, Na Di, Zhaowei Zhu, Jiaheng Wei, Hao Cheng, Chen Qian, and Yang Liu. Token cleaning: Fine-grained data selection for LLM supervised fine-tuning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47837–47858. PMLR, 2025.

Vardan Papyan, Xiao Y Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022. URL https://arxiv.org/abs/2201.02177.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8sKcAWOf2D.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Xiaohan Qin, Xiaoxing Wang, Ning Liao, Cancheng Zhang, Xiangdong Zhang, Mingquan Feng, Jingzhi Wang, and Junchi Yan. ssToken: Self-modulated and semantic-aware token selection for LLM fine-tuning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wZ4UoRKDNa.

Neil Rathi and Alec Radford. Shaping capabilities with token-level data filtering. *arXiv preprint arXiv:2601.21571*, 2026.

Zhiwen Ruan, Yixia Li, He Zhu, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. Enhancing large language model reasoning via selective critical token fine-tuning. *arXiv preprint arXiv:2510.10974*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14590–14604, 2023.

Han Tian, Luxuan Chen, Xinran Chen, Rui Kong, Fang Wang, Jiamin Chen, Jinman Zhao, Yuchen Li, Jiashu Zhao, Shuaiqiang Wang, Haoyi Xiong, and Dawei Yin. Endprompt: Efficient long-context extension via terminal anchoring. *arXiv preprint arXiv:2605.14589*, 2026. URL https://arxiv.org/abs/2605.14589.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization, 2024a. URL https://arxiv.org/abs/2405.15071.

Liang Wang, Nan Yang, Xingxing Zhang, Xiaolong Huang, and Furu Wei. Bootstrap your own context length. *arXiv preprint arXiv:2412.18860*, 2024b. URL https://arxiv.org/abs/2412.18860.

Shaobo Wang, Jiaming Wang, Jiajun Zhang, Cong Wang, Yue Min, Zichen Wen, Xingzhang Ren, Fei Huang, Huiqiang Jiang, Junyang Lin, Dayiheng Liu, and Linfeng Zhang. Winning the pruning gamble: A unified approach to joint sample and token pruning for efficient supervised fine-tuning. *arXiv preprint arXiv:2509.23873*, 2026.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

Yiwei Wu, Atticus Geiger, and Raphaël Millière. How do transformers learn variable binding in symbolic programs? In *Forty-second International Conference on Machine Learning*, 2025.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AssIuHnmHX.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. Pose: Efficient context window extension of llms via positional skip-wise training. In *International Conference on Learning Representations*, volume 2024, pp. 18940–18954, 2024.

A Related work

Token optimization. Standard causal language modeling applies the next-token loss at every position. This spends compute on predictable or redundant tokens (Bal et al., 2025; Wang et al., 2026) and penalizes valid alternative continuations, which can limit output diversity and generalization (Ruan et al., 2025). Token-level methods therefore filter, mask, or reweight the loss at individual positions. Some score tokens against a reference model: *Selective Language Modeling (SLM / RHO-1)* uses the excess loss relative to a model trained on curated data (Lin et al., 2024), *Token Cleaning* separates informative response tokens from template tokens (Pang et al., 2025), and *TOSS* masks tokens that carry safety risks (Li et al., 2026). Other approaches need no reference model: *Self-modulated and Semantic-Aware Token Selection (ssToken)* combines a self-referenced excess loss with attention scores (Qin et al., 2026), and *Efficient Selective Language Modeling (ESLM)* applies risk-based thresholds to per-token loss (Bal et al., 2025). *Critical Token Fine-Tuning (CFT)* keeps only tokens for which every substitution makes the answer wrong (Ruan et al., 2025). *Quadrant-based Tuning (Q-Tuning)* prunes samples and tokens jointly (Wang et al., 2026). Token-level filtering can remove unwanted capabilities during pretraining, for example, by using sparse autoencoders to label latent concept directions, distilling these signals into lightweight, bidirectional token-level classifiers (Rathi & Radford, 2026). Most of these methods are evaluated at standard context lengths (1k–4k tokens) and on short-form tasks. Because they change which positions send gradient, their effect on long-context generalization remains an open question.

Long-context generalization. Most long-context methods change how positions are encoded. *Position Interpolation (PI)* and Rotary Position Embedding (RoPE) rescaling methods such as NTK-aware scaling, YaRN, and LongRoPE extend pretrained context windows (Chen et al., 2023; Peng et al., 2024; Ding et al., 2024), while *Positional Skip-wisE (PoSE)* and *EndPrompt* expose models to long relative distances within short training sequences (Zhu et al., 2024; Tian et al., 2026). Other work designs or compares positional schemes that extrapolate to longer inputs (Press et al., 2021; Sun et al., 2023; Kazemnejad et al., 2023). Data also matters: mixing long and short documents preserves short-context skills, training beyond the target length helps, and progressive length curricula push context further (Gao et al., 2025; Wang et al., 2024b). Some pipelines already restrict the loss, for example to answer tokens only (Gao et al., 2024), or to high-utility tokens selected by a reference model, as in RHO-1 (Lin et al., 2024). Token masking is thus mainly used as a practical heuristic for efficiency or alignment, and, to our knowledge, how supervision density shapes what a model learns, and through it long-context generalization, remains largely unexplored.

Memory and structure inside Transformers. MLP sublayers behave like key–value memories that store content and promote it in vocabulary space (Geva et al., 2021; 2022; Dai et al., 2022), while attention moves information between positions (Elhage et al., 2021; Olsson et al., 2022). Factual recall combines both (Meng et al., 2022; Geva et al., 2023), and the two roles are causally separable (Dong et al., 2025; Fartale et al., 2026). Variable binding is a direct test of this division. Language models bind entities to attributes with binding ID vectors (Feng & Steinhardt, 2024), and fine-tuning strengthens the existing entity-tracking circuit (Prakash et al., 2024). Moreover, recent work (Wu et al., 2025) has described the phases a model goes through before it resolves assignment chains. A related line of work shows that memorizing and generalizing circuits coexist and compete during training (Power et al., 2022; Nanda et al., 2023), that implicitly learned composition may not transfer out of distribution (Wang et al., 2024a), and that length generalization depends on the algorithm a model learns (Zhou et al., 2024). These studies fix the training objective and describe the resulting mechanism. We vary the objective, and ask how it moves a model between the memory (MLP) and structure (attention) axes, and whether this position predicts out-of-distribution behavior.

B EXPERIMENTAL DETAILS

B.1 THE TASK

A program is a string of 67 characters: sixteen assignment lines followed by a query,

$$s_0{=}v_0\backslash\mathtt{n}\ \ s_1{=}v_1\backslash\mathtt{n}\ \ \cdots\ \ s_{15}{=}v_{15}\backslash\mathtt{n}\ \ \#v\mathtt{:} \tag{1}$$

where every line has exactly four characters and the query has three. The names s_r are sixteen distinct letters drawn from $\Gamma = \{\mathtt{a}, \ldots, \mathtt{z}\}$, and the right-hand side v_r is either a digit in $\mathcal{D} = 0, \ldots, 9\}$ or one of the names $s_0, \ldots, s_{r-1}$ defined on an earlier line. The value of a name is obtained by dereferencing until a digit is reached,

$$\mathrm{val}(s_r) = \begin{cases} v_r & v_r \in \{0, \ldots, 9\}, \\ \mathrm{val}(v_r) & \text{otherwise}, \end{cases} \tag{2}$$

and the answer to the query is the digit $\mathrm{val}(v)$. The number of lines visited by this recursion is the number of hops $n \in \{1, 2, 3, 4\}$. Because a right-hand side can only refer to an earlier line, the chain runs upwards in the program.

The model reads the program character by character. The vocabulary $\mathcal{V}$ has 40 symbols: the ten digits, the 26 letters, the four format characters `"="`, `"\n"`, `"#"` and `":"`. A training sequence is the program followed by the answer,

$$x_0, \ldots, x_{66}, x_{67}, \qquad x_{65} = \texttt{":"}, \quad x_{67} = \mathrm{val}(v), \tag{3}$$

and the causal language model predicts x_{t+1} from $x_0, \ldots, x_t$ at every position t. The position that predicts the answer is the `":"`, $t^\star = 66$, namely the *readout position*; every quantity that is "read at the `":"`" is read from the hidden state at $t = t^\star$.

B.2 THE MODELS

All models are GPT-J decoders with $L = 12$ blocks, width $d = 512$, a single RoPE attention head of dimension $d_a = 512$, an MLP of width $d_{MLP} = 2048$ with GELU, dropout 0.1 on embeddings, attention and residual, and untied input and output embeddings. With $\boldsymbol{h}_t^{(\ell)} \in \mathbb{R}^d$ the residual stream at position t after ℓ blocks, the computation is

$$\boldsymbol{h}_t^{(0)} = \boldsymbol{W}_E[x_t], \tag{4}$$

$$\boldsymbol{a}_t^{(\ell)} = \mathrm{Attn}_\ell\big(\mathrm{LN}_\ell(\boldsymbol{h}^{(\ell-1)})\big)_t, \qquad \boldsymbol{m}_t^{(\ell)} = \mathrm{MLP}_\ell\big(\mathrm{LN}_\ell(\boldsymbol{h}_t^{(\ell-1)})\big), \tag{5}$$

$$\boldsymbol{h}_t^{(\ell)} = \boldsymbol{h}_t^{(\ell-1)} + \boldsymbol{a}_t^{(\ell)} + \boldsymbol{m}_t^{(\ell)} \tag{6}$$

for $\ell = 1, \ldots, L$ i.e. the attention and the MLP blocks read the same normalized input and write in parallel to the residual stream (the GPT-J parallel block). This parallel construction will allow to split attention and MLP behavior. The logits and probabilities are

$$\boldsymbol{z}_t = \mathrm{LN}_f(\boldsymbol{h}_t^{(L)})\,\boldsymbol{W}_U + \boldsymbol{b}_U \in \mathbb{R}^{|\mathcal{V}|}, \qquad P(v \mid x_{\le t}) = \mathrm{softmax}(\boldsymbol{z}_t)_v, \tag{7}$$

where $\mathrm{LN}(\boldsymbol{h}) = \boldsymbol{\gamma} \odot (\boldsymbol{h} - \mu(\boldsymbol{h})\mathbf{1})/\sigma(\boldsymbol{h}) + \boldsymbol{\beta}$ with $\mu(\boldsymbol{h}) = \frac{1}{d}\sum_i h_i$ and $\sigma(\boldsymbol{h}) = \sqrt{\frac{1}{d}\sum_i (h_i - \mu)^2 + \varepsilon}$, for learned scale ($\gamma$) and bias ($\beta$) parameters. So, two consequences of (6) emerge. First, the residual stream is an exact sum of component outputs,

$$\boldsymbol{h}_t^{(\ell)} = \boldsymbol{h}_t^{(0)} + \sum_{k=1}^{\ell} \boldsymbol{a}_t^{(k)} + \sum_{k=1}^{\ell} \boldsymbol{m}_t^{(k)}, \tag{8}$$

so the contribution of the *attention pathway* (all attention outputs) and of the *MLP pathway* (all MLP outputs) can be separated without approximation. Second, the final LayerNorm is the only non-linearity between the residual stream and the logits, and it is an affine map once its scale $\sigma(\boldsymbol{h}_t^{(L)})$ is fixed.

B.3 THE TRAINING OBJECTIVE AND THE SUPERVISION DENSITY p

All models minimize the same masked next-token cross-entropy on the same data and differ only in which predictions are penalized. The model is causal, so the prediction of character x_{j+1} is made at position j: the term of index $j-1$ reads the hidden state of x_j and is scored against x_{j+1}. Let $k_p = \lfloor (67-1)p \rfloor$ where p represents the proportion of tokens below the readout position (corresponding to the last token of the sequence), i.e., $k_p = 66, 50, 33, 16, 0$ for $p = 1, 0.75, 0.5, 0.25, 0$, representing the number of supervised tokens upon which the prediction is conditioned. Formally, we define the set of supervised token and the loss for each supervised density ρ models,

$$S_p = \{\, j \in \mathbb{N}^+ \,:\, 66 - k_p \le j < 67 \},\qquad \mathcal{L}_p(\theta) = \frac{1}{|S_p|} \sum_{j \in S_p} \log P_\theta\big(x_{j+1} \mid x_j\big), \tag{9}$$

such that $p = 1$ is the standard auto-regressive objective on the whole sequence, $p = 0$ supervises the answer alone, and the intermediate densities supervise a suffix of the program that ends at the answer.

Every model is trained with AdamW ($\beta_1 = 0.95$, $\beta_2 = 0.999$, weight decay 0.1), learning rate 10^{-4} with $1{,}000$ warm-up steps and linear decay to zero at step $300{,}000$, batch size 64, bfloat16, for $300{,}000$ steps. Every $2{,}000$ steps the model is evaluated without dropout on the $1{,}000$ validation programs and a checkpoint is written every $20{,}000$ steps. The *first maximal accuracy* of a run is the first evaluation at which validation accuracy reaches its maximum, which is 1.0 in the five runs.

B.4 LONG-CONTEXT GENERALIZATION EVALUATION SET

To evaluate long-context generalization we built novel programs with longer input sequences and deeper reasoning chains (i.e., more hops). `ctx` included length up to 16 and now includes values of $\{18, 20, \ldots, 26\}$. Training included sequences up to 4 hops and we now evaluate hops of $\{5, \ldots, 8\}$. Programs cannot exceed 26 lines because every line assigns a different variable and names are single letters from a 26-symbol alphabet, so no name is ever reused. Note that, for completion, we extend the combinations with input sequence length and hop values observed during training. Examples of sequences can be seen below. Sequence A, depicts a case seen during training. Sequence B, portrays an example where the sequence length was seen during training but the depth not. Sequence C, shows exactly the opposite of B, and sequence D depicts a sequence where both length and depth have not seen during training.

Sequence A: `ctx` $= 16$, depth 4. The training setting, deepest trained chain; 12 distractor lines, 68 tokens.

```
c=0\n n=0\n z=7\n w=n\n f=z\n a=n\n p=z\n m=p\n u=p\n b=3\n
y=u\n t=p\n e=6\n d=z\n h=6\n x=4\n #y:
```

Chain: `y` $\to$ `u` $\to$ `p` $\to$ `z` $= 7$.

Sequence B: `ctx` $= 16$, depth 8. A deeper chain at the trained length; 8 distractor lines, 68 tokens.

```
f=5\n a=f\n p=9\n m=f\n y=p\n b=m\n n=y\n i=6\n g=n\n o=g\n
e=o\n h=p\n v=e\n t=6\n k=v\n q=4\n #k:
```

Chain: `k` $\to$ `v` $\to$ `e` $\to$ `o` $\to$ `g` $\to$ `n` $\to$ `y` $\to$ `p` $= 9$.

Sequence C: `ctx` $= 26$, depth 4. A longer sequence at a trained depth; 22 distractor lines, 108 tokens.

```
c=3\n n=c\n z=6\n w=c\n f=6\n a=7\n p=4\n m=z\n u=0\n b=w\n
y=n\n t=2\n e=y\n d=a\n h=m\n x=c\n l=d\n r=y\n i=y\n v=l\n
s=p\n o=l\n k=h\n g=3\n j=8\n q=1\n #o:
```

Chain: `o` $\to$ `l` $\to$ `d` $\to$ `a` $= 7$.

Sequence D: `ctx` $= 26$, depth 8. A longer sequence and a deeper chain; 18 distractor lines, 108 tokens.

```
f=6\n a=f\n p=9\n m=a\n y=a\n b=8\n n=5\n i=b\n g=i\n o=i\n
e=m\n h=f\n v=b\n t=g\n k=v\n q=m\n j=t\n z=j\n c=f\n u=z\n
d=e\n r=u\n s=u\n x=r\n l=z\n w=r\n #r:
```

Chain: r → u → z → j → t → g → i → b = 8.

Every line is "X=Y\n", four tokens, and the query "#v:" is read at the ":". The chain is the sequence of dereferences from the queried variable to the constant, and its length is the depth. Training uses sequences of 16 lines with chains of up to 4 hops; generalization is measured on longer sequences and on deeper chains. The test sets keep the alphabet and the grammar of training, so what varies is length and depth alone, not the symbols or the format.

The lines outside the chain are distractors, drawn by the same process as in training: a reference to an earlier variable with probability 0.7, a constant otherwise. Their number is ctx − depth, so a longer sequence adds competing bindings while a deeper chain removes them.

B.5 Metrics read at the readout position

B.5.1 Answer accuracy

Let c be the correct digit of a sequence and $\mathcal{D} \subset \mathcal{V}$ the ten digit tokens. Then, we define the accuracy for a given sequence as a binary function,

$$\textit{accuracy}: \quad \mathbf{1}\big[\arg\max_{v\in\mathcal{V}} z_v = c\big] \tag{10}$$

is the exact match over the whole vocabulary, not only over the digits. The mean average accuracy through a set of points will be the evaluated metric during training. The logit-margin metric are read from the logits $\boldsymbol{z} = \boldsymbol{z}_{t^\star}$ of (7),

$$\textit{logit-margin}: \quad M = z_c - \frac{1}{9}\sum_{j\in\mathcal{D}\setminus\{c\}} z_j \tag{11}$$

is the lead of the correct digit over the average of the other nine digits, in logits, and it is the quantity that the attribution methods of Sections 3.2 uses. For analyses purpose, the models LayerNorm gains folded into the weights that follow them, so that $\mathrm{LN}_f(\boldsymbol{h})\boldsymbol{W}_U + \boldsymbol{b}_U = \big((\boldsymbol{h}-\mu\mathbf{1})/\sigma\big)\widetilde{\boldsymbol{W}}_U + \widetilde{\boldsymbol{b}}_U$ with $\widetilde{\boldsymbol{W}}_U = (\boldsymbol{\gamma}_f)\boldsymbol{W}_U$ and $\widetilde{\boldsymbol{b}}_U = \boldsymbol{b}_U + \boldsymbol{\beta}_f\boldsymbol{W}_U$. From now on $\boldsymbol{W}_U$ and $\boldsymbol{b}_U$ denote the folded matrices Writing $\boldsymbol{w}_j = \boldsymbol{W}_U[:, j]$ for the unembedding column of token j, the logit-margin is the projection of the normalized final residual on the answer direction

$$\boldsymbol{d}_c = \boldsymbol{w}_c - \frac{1}{9}\sum_{j\in\mathcal{D}\setminus\{c\}} \boldsymbol{w}_j, \qquad M = \Big\langle \frac{\boldsymbol{h}^{(L)}_{t^\star} - \mu\mathbf{1}}{\sigma},\, \boldsymbol{d}_c \Big\rangle + \underbrace{\Big(b_{U,c} - \frac{1}{9}\sum_{\substack{j\neq c\\ j\in\mathcal{D}\setminus\{c\}}} b_{U,j}\Big)}_{=\mathrm{C}}, \tag{12}$$

for with μ, σ the statistics of $\boldsymbol{h}^{(L)}_{t^\star}$. The answer direction depends on the sequence through c and on the checkpoint through $\boldsymbol{W}_U$.

B.5.2 Logit lens

The logit lens asks, for every depth ℓ, what the model would predict if the residual stream at the readout were sent to the unembedding after ℓ blocks instead of after all L. The normalization is the classical one: each partial residual is passed through the final LayerNorm with *its own* statistics and then through the (folded) unembedding,

$$\boldsymbol{z}^{(\ell)} = \frac{\boldsymbol{h}^{(\ell)}_{t^\star} - \mu^{(\ell)}\mathbf{1}}{\sigma^{(\ell)}}\,\boldsymbol{W}_U + \boldsymbol{b}_U, \qquad \mu^{(\ell)}, \sigma^{(\ell)} \text{ the mean and scale of } \boldsymbol{h}^{(\ell)}_{t^\star}, \qquad \ell = 0, \dots, L \tag{13}$$

At $\ell = L$ the partial residual is the final one, so (13) is exactly the model's logits (7). This choice has a consequence that matters when recomputing $\sigma^{(\ell)}$ at every depth makes (13) *non-linear* in the

residual, so the lens at depth ℓ is *not* the partial sum of the attribution over the components written up to block ℓ. Subsequently, at B.5.4, we introduce a homogeneous renormalization across all layers, effectively linearizing this non-linear operation. Then, we can write the logit-margin for any depth ℓ,

$$M^{(\ell)} = z_c^{(\ell)} - \frac{1}{9}\sum_{j\in\mathcal{D}\setminus\{c\}} z_j^{(\ell)} = \Big\langle \frac{\boldsymbol{h}_{t^\star}^{(\ell)} - \mu^{(\ell)}\mathbf{1}}{\sigma^{(\ell)}}, \boldsymbol{d}_c \Big\rangle + \mathrm{C} \tag{14}$$

The two answer different questions: the lens asks what the model would predict if it stopped at ℓ and renormalised, the attribution asks how much of the final margin had been written by then.

B.5.3 Margin bound

Since probability (7) is the softmax over the whole vocabulary, it is confined in $[0, 1]$ and includes the decision of whether the next token is a digit; yet the logit-margin (14) is not a priori bounded. Nonetheless, we demonstrate that it is bounded.

Theorem 1 (the LayerNorm caps the margin)**.** *Work in the folded convention of Section B.2, so that $\boldsymbol{z}^{(\ell)} = \tilde{\boldsymbol{h}}^{(\ell)}\boldsymbol{W}_U + \boldsymbol{b}_U$ with $\tilde{\boldsymbol{h}}^{(\ell)} = \big(\boldsymbol{h}_{t^\star}^{(\ell)} - \mu^{(\ell)}\mathbf{1}\big)/\sigma^{(\ell)}$ and every column of $\boldsymbol{W}_U$ of zero mean along the $d = 512$ coordinates. Let $\varepsilon > 0$ be the LayerNorm constant and $b_c^\Delta = b_{U,c} - \frac{1}{9}\sum_{j\neq c} b_{U,j}$, and assume $\boldsymbol{d}_c \neq \mathbf{0}$, which is what makes the answer direction a direction at all: if $\boldsymbol{d}_c = \mathbf{0}$ no residual can separate the correct digit from the other nine and $M^{(\ell)} = b_c^\Delta$ identically. Then, for every sequence and every depth $\ell = 0, \ldots, L$:*

(i) the normalised residual has a fixed, known length, $\big\|\tilde{\boldsymbol{h}}^{(\ell)}\big\| = \sqrt{d\big(1 - \varepsilon/(\sigma^{(\ell)})^2\big)} < \sqrt{d}$, and it is centred, $\langle \tilde{\boldsymbol{h}}^{(\ell)}, \mathbf{1}\rangle = 0$;

(ii) consequently the margin is bounded,

$$M^{(\ell)} \;\le\; \big\|\tilde{\boldsymbol{h}}^{(\ell)}\big\|\,\|\boldsymbol{d}_c\| + b_c^\Delta \;<\; \sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^\Delta; \tag{15}$$

(iii) the first inequality in (15) is an equality exactly when $\tilde{\boldsymbol{h}}^{(\ell)}$ is a non-negative multiple of $\boldsymbol{d}_c$, a configuration that the constraints in (i) permit, so $\sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^\Delta$ is the supremum of $M^{(\ell)}$ over all $\boldsymbol{h}_{t^\star}^{(\ell)} \in \mathbb{R}^d$, approached as $\varepsilon/(\sigma^{(\ell)})^2 \to 0$ and attained by none of them.

Proof. (i) By definition $(\sigma^{(\ell)})^2 = \frac{1}{d}\sum_i \big(h_i^{(\ell)} - \mu^{(\ell)}\big)^2 + \varepsilon$, so $\big\|\boldsymbol{h}^{(\ell)} - \mu^{(\ell)}\mathbf{1}\big\|^2 = d\big((\sigma^{(\ell)})^2 - \varepsilon\big)$; dividing by $(\sigma^{(\ell)})^2$ gives the length, and $\varepsilon > 0$ makes the inequality strict. Centredness is immediate from subtracting the mean.

(ii) By (14) the margin is the linear functional $M^{(\ell)} = \langle \tilde{\boldsymbol{h}}^{(\ell)}, \boldsymbol{d}_c\rangle + b_c^\Delta$, so Cauchy–Schwarz gives the first inequality and (i) the second.

(iii) Equality in Cauchy–Schwarz holds iff $\tilde{\boldsymbol{h}}^{(\ell)}$ and $\boldsymbol{d}_c$ are linearly dependent, and since the right-hand side $\|\tilde{\boldsymbol{h}}^{(\ell)}\|\,\|\boldsymbol{d}_c\|$ is non-negative while the left-hand side of a dependence $\tilde{\boldsymbol{h}}^{(\ell)} = \lambda\boldsymbol{d}_c$ is $\lambda\|\boldsymbol{d}_c\|^2$, a negative λ gives a strict inequality: equality holds iff $\lambda \ge 0$. The boundary case $\lambda = 0$ is degenerate and carries no margin: by (i) it occurs exactly when $(\sigma^{(\ell)})^2 = \varepsilon$, that is when $\boldsymbol{h}_{t^\star}^{(\ell)}$ is constant along its d coordinates, and then $M^{(\ell)} = b_c^\Delta$.

Such a configuration is admissible. It must meet the two constraints of (i), centredness and fixed length. For the first, $\boldsymbol{d}_c$ is itself centred: in the folded convention every column of $\boldsymbol{W}_U$ has zero mean along the d coordinates, a normalisation that changes no logit, because the unembedding is applied only to $\tilde{\boldsymbol{h}}^{(\ell)}$, which is centred, so adding any multiple of $\mathbf{1}$ to a column leaves every $\langle \tilde{\boldsymbol{h}}^{(\ell)}, \boldsymbol{w}_v\rangle$ unchanged; as $\boldsymbol{d}_c = \boldsymbol{w}_c - \frac{1}{9}\sum_{j\neq c}\boldsymbol{w}_j$ is a linear combination of columns, $\langle \boldsymbol{d}_c, \mathbf{1}\rangle = 0$. For the second, take the residual $\boldsymbol{h}^{(\ell)} = t\,\boldsymbol{d}_c$ with $t > 0$, so that $\mu^{(\ell)} = 0$ and $(\sigma^{(\ell)})^2 = t^2\|\boldsymbol{d}_c\|^2/d + \varepsilon$;

then $\tilde{\boldsymbol{h}}^{(\ell)} = t\,\boldsymbol{d}_c/\sigma^{(\ell)}$ is a positive multiple of $\boldsymbol{d}_c$, the first inequality of (15) is an equality, and

$$M^{(\ell)} = \big\|\tilde{\boldsymbol{h}}^{(\ell)}\big\|\,\|\boldsymbol{d}_c\| + b_c^{\Delta} = \frac{\sqrt{d}\,\|\boldsymbol{d}_c\|}{\sqrt{1 + d\varepsilon/\big(t^2\|\boldsymbol{d}_c\|^2\big)}} + b_c^{\Delta},$$

which is strictly increasing in t and converges to $\sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^{\Delta}$ as $t \to \infty$, equivalently as $\varepsilon/(\sigma^{(\ell)})^2 \to 0$, without attaining it for any $\varepsilon > 0$. Hence the first bound is attained at every t, the second is strict, and $\sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^{\Delta}$ is the least upper bound of $M^{(\ell)}$ over all $\boldsymbol{h}_{t^\star}^{(\ell)} \in \mathbb{R}^d$. □

The supremum is over every vector the architecture admits at that position, not over the states the trained model happens to reach, which is what makes the saturation (17) a fraction of an architectural budget.

Corollary 2 (bounded confidence)**.** *Let $P^{(\ell)}(c \mid \mathcal{D}) = e^{z_c^{(\ell)}} \big/ \sum_{j\in\mathcal{D}} e^{z_j^{(\ell)}}$ be the readout restricted to the ten digits. Then*

$$P^{(\ell)}(c) \;\le\; P^{(\ell)}(c \mid \mathcal{D}) \;\le\; \frac{1}{1 + 9e^{-M^{(\ell)}}} \;\le\; \frac{1}{1 + 9\exp\big(-\sqrt{d}\,\|\boldsymbol{d}_c\| - b_c^{\Delta}\big)}. \tag{16}$$

Proof. The first inequality holds because the two readouts share a numerator and the full-vocabulary denominator contains the restricted one, $\sum_{v\in\mathcal{V}} e^{z_v} \ge \sum_{j\in\mathcal{D}} e^{z_j}$. For the second, convexity of the exponential (equivalently the arithmetic–geometric mean inequality) gives $\frac{1}{9}\sum_{j\neq c} e^{z_j} \ge \exp\big(\frac{1}{9}\sum_{j\neq c} z_j\big) = e^{z_c - M^{(\ell)}}$, where the equality is the definition (11) of the margin; since $t \mapsto e^{z_c}/(e^{z_c} + t)$ is decreasing, replacing $\sum_{j\neq c} e^{z_j}$ by the smaller $9e^{z_c - M^{(\ell)}}$ can only increase the ratio, and the e^{z_c} cancel. The third follows from Theorem 1(ii) and the monotonicity of $x \mapsto (1 + 9e^{-x})^{-1}$. □

Saturation. Theorem 1 turns the margin into a product of three factors that can be read separately. Define the *saturation* of a readout as the fraction of its own ceiling that it attains,

$$\begin{aligned} S^{(\ell)} \;&=\; \frac{M^{(\ell)} - b_c^{\Delta}}{\sqrt{d}\,\|\boldsymbol{d}_c\|} \;=\; \cos\big(\tilde{\boldsymbol{h}}^{(\ell)}, \boldsymbol{d}_c\big)\cdot\sqrt{1 - \varepsilon/(\sigma^{(\ell)})^2} \;\in [-1, 1], \\ \text{so}\qquad M^{(\ell)} &= \underbrace{\sqrt{d}}_{\text{architecture}} \cdot \underbrace{\|\boldsymbol{d}_c\|}_{\text{weights}} \cdot \underbrace{S^{(\ell)}}_{\text{alignment}} + b_c^{\Delta} \end{aligned} \tag{17}$$

The middle expression is the content of the definition: because the length of $\tilde{\boldsymbol{h}}^{(\ell)}$ is fixed by the LayerNorm and not by the model, the only thing a readout can do to earn margin is to *point* in the right direction, and $S^{(\ell)}$ is that direction, read as a cosine. A saturation of 1 means the residual at the `:` is exactly the answer direction and the model writes the largest margin its own unembedding allows; 0 means the residual is orthogonal to it and no margin is written; a negative value means the residual points at the wrong digits. What the saturation is *not* is a measure of how large the margin is: it is deliberately blind to $\|\boldsymbol{d}_c\|$, so two models whose unembeddings differ in scale can write very different margins at the same saturation, and conversely.

B.5.4 Direct logit attribution (contribution)

Direct logit attribution (DLA) splits the logit-margin (11) into the exact contributions of the 25 components that write to the residual stream at the readout: the embedding, the twelve attention outputs and the twelve MLP outputs. By (8) the final residual is their sum, and by (12) the margin is a linear function of the centered final residual divided by the final scale $\sigma^{(L)}$. Since centering is linear, each component can be centered and scaled separately and the pieces add up:

$$\mathrm{DLA}(\boldsymbol{c}) = \Big\langle \frac{\boldsymbol{c} - \mu(\boldsymbol{c})\mathbf{1}}{\sigma^{(L)}},\, \boldsymbol{d}_c \Big\rangle, \qquad \sum_{\boldsymbol{c}\in\mathcal{C}} \mathrm{DLA}(\boldsymbol{c}) = M - \mathrm{C}, \tag{18}$$

where $\mathcal{C} = \{\boldsymbol{h}^{(0)}, \boldsymbol{a}^{(1)}, \boldsymbol{m}^{(1)}, \ldots, \boldsymbol{a}^{(L)}, \boldsymbol{m}^{(L)}\}$ are the 25 component outputs at $t = t^\star$ and the constant is the bias term of (12), which no component controls. The decomposition is exact because

the final LayerNorm is used with the scale of the actual final residual ($\ell = L$), shared by all 25 terms. Three cumulative curves are drawn against depth:

$$T_\ell = \mathrm{DLA}(\boldsymbol{h}^{(0)}) + \sum_{k\le\ell} \big[\mathrm{DLA}(\boldsymbol{a}^{(k)}) + \mathrm{DLA}(\boldsymbol{m}^{(k)})\big],$$
$$A_\ell = \mathrm{DLA}(\boldsymbol{h}^{(0)}) + \sum_{k\le\ell} \mathrm{DLA}(\boldsymbol{a}^{(k)}), \qquad B_\ell = \mathrm{DLA}(\boldsymbol{h}^{(0)}) + \sum_{k\le\ell} \mathrm{DLA}(\boldsymbol{m}^{(k)}), \tag{19}$$

the total, the attention pathway and the MLP pathway. T_ℓ coincides with the logit-lens margin $M^{(\ell)}$ of (14), and T_L is the model's margin up to the constant. A_ℓ and B_ℓ are the two counterfactual readouts in which only one pathway had written to the residual, and $A_L + B_L - \mathrm{DLA}(\boldsymbol{h}^{(0)}) = T_L$. The balance between the pathways is summarized per sequence by the *attention share*

$$s = \frac{|\sum_k \mathrm{DLA}(\boldsymbol{a}^{(k)})|}{|\sum_k \mathrm{DLA}(\boldsymbol{a}^{(k)})| + |\sum_k \mathrm{DLA}(\boldsymbol{m}^{(k)})|} \in [0,1], \tag{20}$$

which is 1 when attention writes the whole margin and 0 when the MLP does, and is then averaged over sequences. All curves are accumulated per sequence before averaging.

B.5.5 Mean ablation

Direct logit attribution (Section B.5.4) measures what a pathway writes into the answer, not whether the model needs it. To test necessity, we replace the output of one pathway, in all six blocks $\ell = 7, \dots, 12$ at once and at every position, by its mean over programs of the training split, $\boldsymbol{a}_t^{(\ell)} \leftarrow \bar{\boldsymbol{a}}_{r(t)}^{(\ell)}$ (and likewise $\boldsymbol{m}_t^{(\ell)}$ for the MLP, in a separate run), leaving the rest of the network intact. The mean is taken per block and per token role $r(t)$: the four characters of a line plus `#`, the queried name, `:` and the answer. It is estimated from 10,000 programs of 16 lines with the weights frozen, and since roles depend only on the layout, the same means are reused at every position of longer programs. Unlike zeroing, mean ablation keeps the average contribution of the pathway and removes only what it computes about the particular program. Blocks 7–12 are the ones that write the answer: they account for 83–99.6% of the direct contribution of both pathways to the margin equation 11 (Figure 4). The ablation acts at evaluation time only, so accuracy under it is measured on exactly the programs used for the intact model.

B.5.6 Attention-flow diagrams

The diagrams follows one program through every layer of a model, one column per input character, and draw two things on the same grid: what every hidden state would predict, and where every position looked.

The sample program used in every diagram is:

```
q=9 i=9 b=5 k=6 l=k o=q n=0 a=n m=3 c=l f=o p=c v=b s=7 z=p x=q #p: → 6
```

whose chain is `#p:`→`p=c` (line 11)→`c=l` (line 9)→`l=k` (line 4)→`k=6` (line 3), lines counted from 0 as in the legend of the diagrams.

Rows and nodes. The first row correspond to the `Input`, i.e. the prompt. The twelve rows below it, `L1`–`L12`, are the residual stream at the output of each block, $\boldsymbol{h}^{(1)}, \dots, \boldsymbol{h}^{(12)}$. The node at row ℓ and position t carries the *readout* of $\boldsymbol{h}_t^{(\ell)}$: the most probable token when that state is sent through the unembedding,

$$\hat{v}_t^{(\ell)} = \operatorname*{arg\,max}_{v \in \mathcal{V}\setminus\{\texttt{PAD}\}} \big(\boldsymbol{h}_t^{(\ell)}\boldsymbol{W}_U + \boldsymbol{b}_U\big)_v, \qquad \text{opacity} = \frac{\max_v \mathrm{softmax}(\boldsymbol{h}_t^{(\ell)}\boldsymbol{W}_U + \boldsymbol{b}_U)_v - 1/|\mathcal{V}'|}{1 - 1/|\mathcal{V}'|}, \tag{21}$$

so a uniform distribution is transparent and a confident one opaque; the last row uses the model's final LayerNorm and is the model's actual prediction, the intermediate rows read the raw residual. Digits are set in bold so that the number can be spotted wherever it surfaces. The lines of the query chain are each drawn in a single color, and the twelve distractor lines are gray.

Arrows. An arrow from column j on row $\mathtt{L}(\ell-1)$ to column i on row $\mathtt{L}(\ell)$ means that the head of block ℓ, when computing position i, put attention weight $A_{ij}^{(\ell)}$ on position j; a short vertical arrow is a position attending to itself. The color of an arrow is that of its source line and its width is proportional to $A_{ij}^{(\ell)}$; arrows that land on the readout position 66 are drawn thicker and on top of everything else, because they are the reads that can write the answer. The black ring marks the first row on which position predicts the correct digit with probability above 0.5.

The arrows depict the *top-2 chain*, i.e., the pathway related to the 2 highest attention values (the two strongest sources of every node).

B.5.7 Cosine geometry of the pathways

The attribution of a component mixes two things, how well it points at the answer and how much it writes. Because the centered component has the same direction as the raw one (the writing weights are centered, Section B.2), (18) factorizes exactly as

$$\mathrm{DLA}(\boldsymbol{c}) = \cos(\boldsymbol{c}, \boldsymbol{d}_c) \cdot \|\boldsymbol{c}\| \cdot \frac{\|\boldsymbol{d}_c\|}{\sigma_L}, \qquad \cos(\boldsymbol{c}, \boldsymbol{d}_c) = \frac{\langle \boldsymbol{c}, \boldsymbol{d}_c \rangle}{\|\boldsymbol{c}\| \, \|\boldsymbol{d}_c\|}, \tag{22}$$

into an *alignment*, a *magnitude* and a factor common to all components of a sequence. In fact, this split version of DLA will help our analysis to understand better the evaluation of the alignment readout prediction in the answer direction. Alignment and magnitude can thus be studied separately for any component that writes to the residual stream, whether the output of a single block, $\boldsymbol{a}^{(\ell)}$ or $\boldsymbol{m}^{(\ell)}$, or the sum of each pathway,

$$\Sigma_{\mathrm{attn}} = \sum_{\ell=1}^{L} \boldsymbol{a}_{t^\star}^{(\ell)}, \qquad \Sigma_{\mathrm{mlp}} = \sum_{\ell=1}^{L} \boldsymbol{m}_{t^\star}^{(\ell)}, \tag{23}$$

for which (22) reproduces exactly the total attribution of the pathway. Cosines are invariant to positive rescaling, so they are computed on the raw outputs, without LayerNorm.

Two unrelated directions in $\mathbb{R}^{512}$ are not orthogonal in practice: the cosine of a fixed vector with a random one is approximately Gaussian with standard deviation $1/\sqrt{512}$, so $|\cos| \leq 0.087$ is the 95% band of "no alignment", drawn in gray where the axis allows it.

For a model with unembedding columns $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_9$ for the ten digits, the *centered* digit directions and the *raw* ones are

$$\tilde{\boldsymbol{u}}_j = \frac{\boldsymbol{w}_j - \bar{\boldsymbol{w}}}{\|\boldsymbol{w}_j - \bar{\boldsymbol{w}}\|}, \quad \bar{\boldsymbol{w}} = \frac{1}{10} \sum_k \boldsymbol{w}_k; \qquad \boldsymbol{u}_j = \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}. \tag{24}$$

The centered direction of the correct digit is the answer direction: $\boldsymbol{d}_c = \boldsymbol{w}_c - \frac{1}{9}\sum_{j \neq c} \boldsymbol{w}_j = \frac{10}{9}(\boldsymbol{w}_c - \bar{\boldsymbol{w}})$, so $\cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_c) = \cos(\boldsymbol{v}, \boldsymbol{d}_c)$ for any $\boldsymbol{v}$. The ten centered vectors sum to zero by construction, which yields a *geometric shadow*: for any $\boldsymbol{v}$, $\sum_j \|\boldsymbol{w}_j - \bar{\boldsymbol{w}}\| \cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_j) = 0$, and then suppose the ten norms are similar,

$$\frac{1}{9} \sum_{j \neq c} \cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_j) \approx -\frac{\cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_c)}{9}. \tag{25}$$

A vector that merely points at the correct digit therefore has, on average, a slightly negative cosine with the other nine; pushing *against* them would show as a mean below the shadow.

Where the constant comes from. The constant of (25) is equal to $-1/(C-1)$ for $C = 10$ classes, and it is the equiangularity constant of a *simplex equiangular tight frame*: the configuration of C equinorm vectors that sum to zero and make the same angle with one another, which is the most spread-out arrangement of C centered directions in a space of dimension at least $C-1$. It is the object that Papyan et al. (2020) call **NC2**, the second of their four neural-collapse phenomena,

stated there for the *class means* of the last-layer features; what makes it a statement about the ten unembedding columns here is their **NC3**, the self-duality by which the last-layer classifier converges to those same centered class means up to a scale. So (25) is not a property of this model but of any readout whose digit directions are close to that frame, so this justifies our supposition referring to equalized norms.

If the norms are not equal. The identity behind (25) needs no assumption at all in its weighted form: with $n_j = \|\boldsymbol{w}_j - \bar{\boldsymbol{w}}\|$ and $\tilde{\boldsymbol{u}}_j$ as in (24), $\sum_j n_j \cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_j) = \langle \boldsymbol{v}, \sum_j (\boldsymbol{w}_j - \bar{\boldsymbol{w}}) \rangle / \|\boldsymbol{v}\| = 0$ exactly, for every $\boldsymbol{v}$. Splitting off the correct digit and writing $\bar{n}_{-c} = \frac{1}{9}\sum_{j \neq c} n_j$ gives the exact shadow,

$$\frac{1}{9}\sum_{j \neq c} \cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_j) = -\frac{n_c}{9\,\bar{n}_{-c}}\,\cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_c) - \frac{\mathrm{Cov}_{j \neq c}\big(n_j, \cos(\boldsymbol{v}, \tilde{\boldsymbol{u}}_j)\big)}{\bar{n}_{-c}}, \tag{26}$$

which reduces to (25) when the ten norms are equal. Two things change when they are not. The first term rescales the shadow by $n_c/\bar{n}_{-c}$, so a digit whose centered column is longer than average drags the other nine further down than $-\cos_c/9$—the weighted sum has to balance a heavier term—and a shorter one lets them off. The second term is a genuine asymmetry: if the longer columns happen to be the ones aligned with $\boldsymbol{v}$, the unweighted mean is pushed below the rescaled shadow even when nothing is being actively suppressed.

C Additional results

Here, we provide additional results and figures for all the subsections of our result section. We present them in the sequential order they appear and are referred to in the main text.

C.1 Learning speed and generalization

Figure 8 shows the learning curves separated per hop number. We observe the same pattern regardless the hop number.

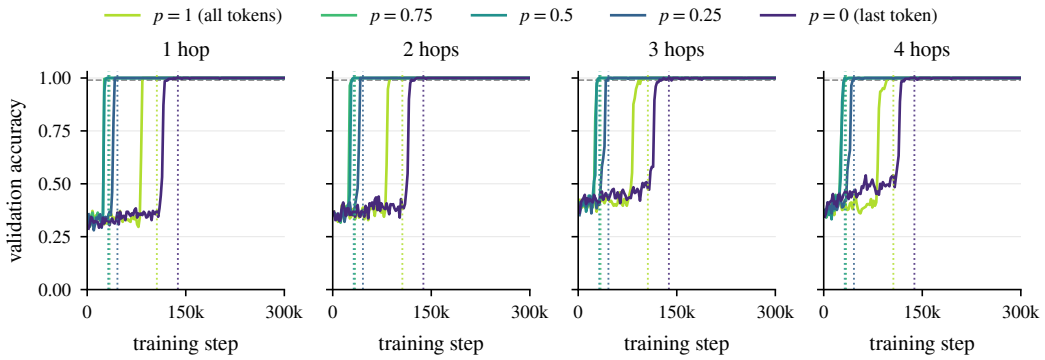


Figure 8: Validation accuracy by hop number: on panel per number of hops (250 programs each) and one curve per model.

Table 2 evaluates each first-maximal-accuracy checkpoint on the full held-out test set of 49,000 balanced sequences, which is never used for training or model selection. For each checkpoint, we report overall and per-hop accuracy (10), the number of misclassified programs, and the mean logit margin at the readout (11).

Section 3.1 reports long-context generalization as one number per density, the mean over programs that are both longer and deeper than training. Figures 9 and 10 resolve that number over the full grid at the first maximal accuracy and last (300k) checkpoints, respectively. Within a panel, moving up lengthens the program at a fixed hop number and moving right deepens the chain at a fixed length, so the black box at the bottom left is the training setting and the green box at the top right is the region the body reports.

Table 2: Evaluation on the 49,000 test programs at the first maximal accuracy model and at checkpoint 300k.

p	checkpoint	acc. (%)	1 hop	2 hops	3 hops	4 hops	errors	margin
1	best (106k)	99.98	100.00	100.00	99.98	99.94	9	9.23
	300k	100.00	100.00	100.00	100.00	99.98	2	13.40
0.75	best (34k)	99.81	99.99	99.97	99.83	99.46	92	8.46
	300k	99.99	100.00	100.00	100.00	99.98	3	13.20
0.5	best (32k)	99.97	100.00	100.00	99.99	99.88	16	8.65
	300k	100.00	100.00	100.00	100.00	100.00	0	14.60
0.25	best (46k)	99.88	100.00	99.99	99.91	99.62	58	10.71
	300k	100.00	100.00	100.00	100.00	100.00	0	16.00
0	best (138k)	99.83	99.99	99.86	99.74	99.75	81	9.94
	300k	100.00	100.00	100.00	100.00	100.00	0	12.89

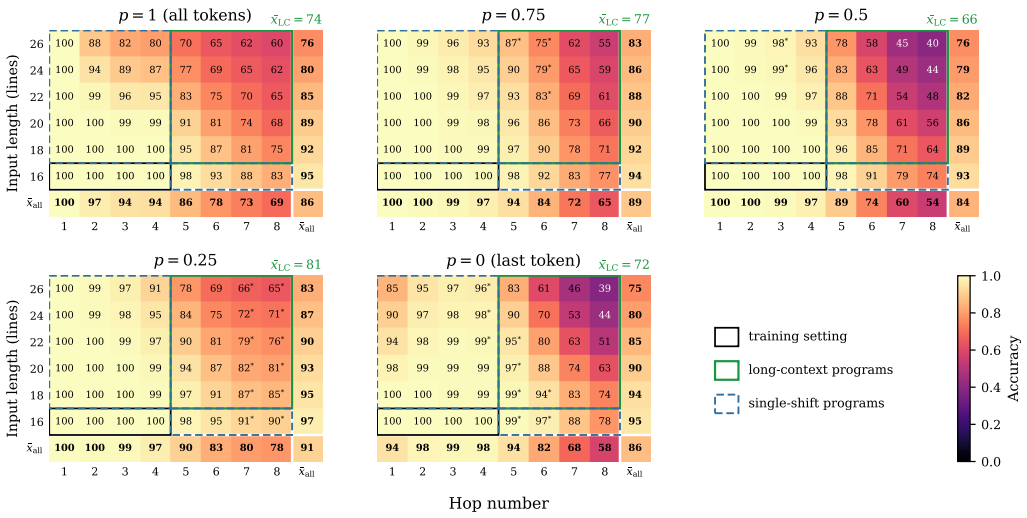


Figure 9: Answer accuracy (%) at the first maximal accuracy model, one panel per supervision density p. Rows are input length and columns are hop number. Boxes mark the three kinds of evaluation: the training setting in black, the long-context programs in green, both longer and deeper than training, and in dashed blue the two blocks that shift one axis only (longer context alone or more hops alone). The appended row and column, in bold, average each axis, and above each panel $\bar{x}_{\mathrm{LC}}$ gives that model's mean over the green box. A star marks a density significantly more accurate than each of the other four, by a paired McNemar test. Every combination holds 12,250 sequences.

The cost of length depends on the number of hops. Added lines are distractors, assignments the query never dereferences, so their effect scales with the amount of dereferencing a program requires. Reading a column upward gives the effect of lengthening a program at a fixed hop number. In the first four columns, those seen in training, the ten additional lines leave the three intermediate densities essentially unchanged from the bottom row to the top. From the fifth column on the same upward reading darkens quickly. The two ends of the ladder depart from this pattern in opposite ways, and each departure shows as an anomaly in a single column of its panel. For $p = 1$, the top row already loses accuracy in the second column, on chains the model saw in training. For $p = 0$, the first column darkens upward, although these are the shortest queries in the set, so what fails at convergence is locating the queried line rather than following a chain.

Continued training extends the reach of $p = 0$. The first column improves where it had failed, from 85 to 93% at 26 lines, so the deficit at hop 1 partly closes. The larger change is inside the green box, whose mean rises from 72 to 91%, and it is uneven. Cells between roughly 50 and 80%

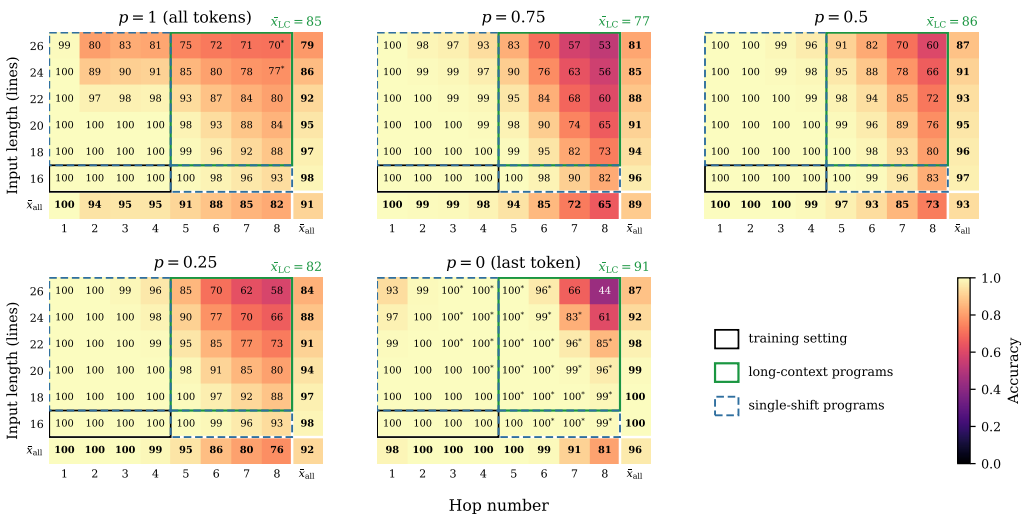


Figure 10: Same as Figure 9 with continued learning (300k steps). Comparing $\bar{x}_{\mathrm{LC}}$ across the two figures gives the effect of training past convergence. $p = 0$ moves from 72 to 91% and $p = 0.5$ from 66 to 86%, while $p = 0.75$ and $p = 0.25$ barely move. The test reflects the same reversal. Twenty-nine combinations have a single winner, $p = 0$ takes twenty-seven of them and the three intermediate densities take none, the two exceptions going to $p = 1$ at the longest and deepest combinations, 24 and 26 lines at eight hops. The star reports significance, not magnitude.

at convergence gain 20 to 35 points, while the extreme cell, 26 lines at eight hops gain 5. Longer training therefore widens the region the model handles without reaching its hardest programs.

C.2 The solution space: attention versus MLP

In the main text, each pathway contribution is evaluated at the first maximal-accuracy checkpoint. In Figure 11, we extend the same decomposition across training time.

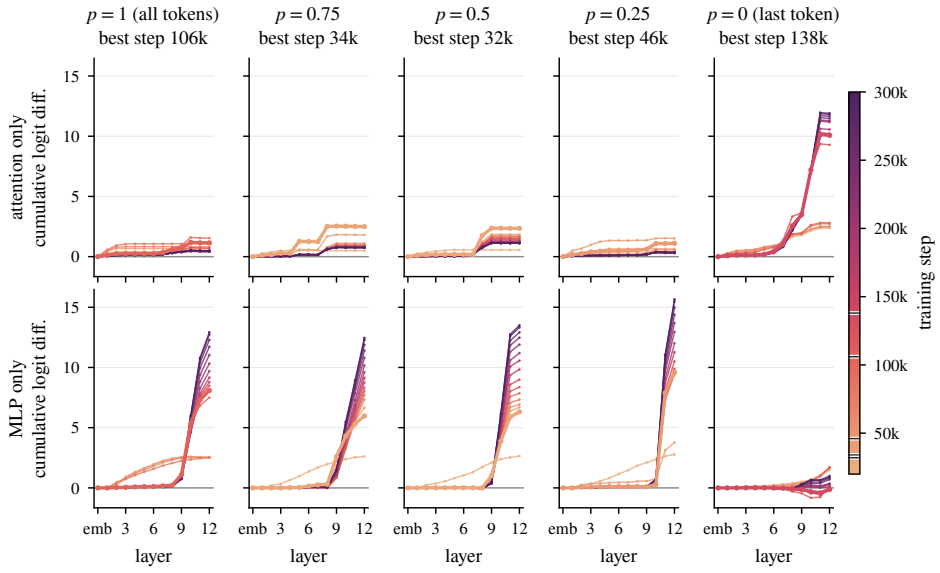


Figure 11: Contribution of attention (top) and MLP (bottom) by layer. One curve represents a training checkpoint (color code).

Figure 12 expresses the same alignment measure geometrically. For each pathway sum and checkpoint, it plots a unit-length vector at angle $\theta = \arccos\cos(\Sigma, d_c)$ from the origin, with d_c, with the answer direction d_c fixed at zero radians; the horizontal projection therefore recovers the cosine similarity. Across supervision regimes, pathway representations progressively align with the answer direction, and this alignment continues to strengthen well beyond first maximal accuracy, increasingly reflecting each model's MLP- or attention-dominated solution.

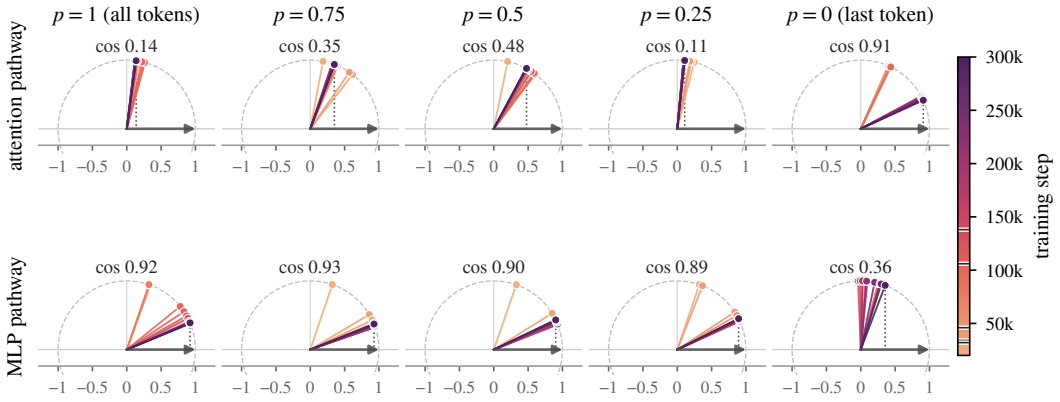


Figure 12: Alignment of each pathway with the answer direction during training, one ray per checkpoint. The annotated value is the median cosine over the test sequences at step 300k.

Figure 13 shows the alignment with the answer direction as a function of learning steps (top graphs). We can see that for last-token learning, significantly improving its generalization performance with continued training is associated to a stronger alignment of the MLP pathway. Bottom graphs display the norms.

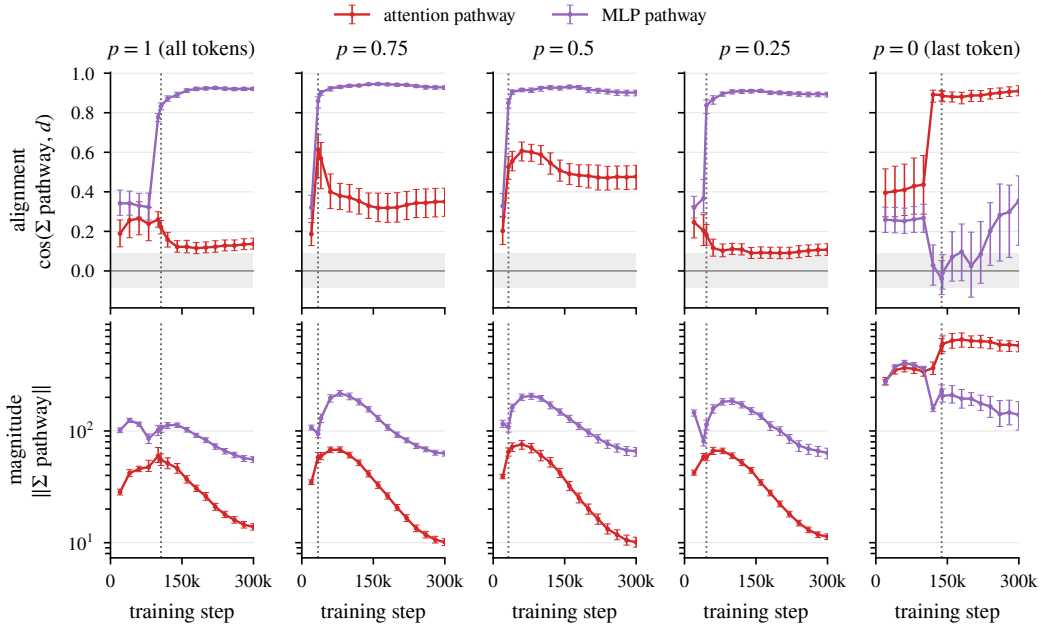


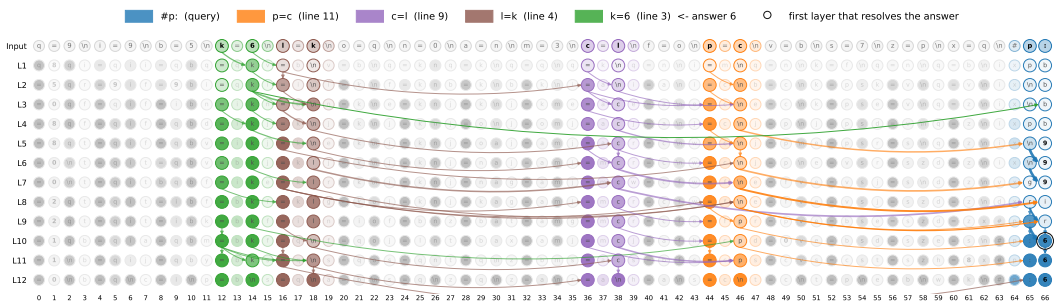
Figure 13: Each pathway's alignment with the answer direction (top) and its norm (bottom) over training, attention in red and MLP in purple; error bars across evaluation examples, and the dotted line marks the first maximal accuracy checkpoint. The gray band is the null range for alignment.

Table 3 reports the mean ablation for the first maximal accuracy model.

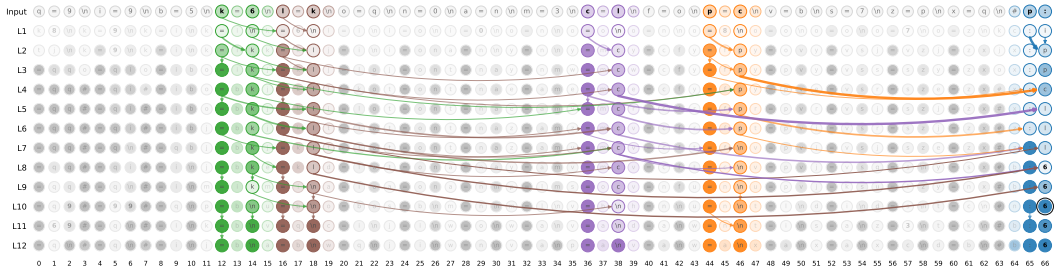
Table 3: Long-context accuracy (%) after mean-ablating attention (−attn) or MLP (−MLP) in layers 7–12 at the first maximal accuracy checkpoint. It is measured on the 20 combinations that are longer and deeper than training at once (18–26 lines, 5–8 hops), evaluated on a fixed subsample of 2,000 of the 12,250 programs of each combination, shared by all models and conditions; *overall* is its mean and the remaining columns split it in halves along each stress, by input length and by hop number. Chance level is 10%.

			input length		hop number	
p		overall	lines 18–20	lines 22–26	hops 5–6	hops 7–8
1 (all tokens)	**intact**	**73.4**	**81.3**	**68.1**	**78.7**	**68.0**
	−attn	26.9	28.4	25.9	25.5	28.3
	−MLP	57.2	63.9	52.7	60.8	53.6
0.75	**intact**	**77.1**	**82.5**	**73.5**	**87.7**	**66.4**
	−attn	34.6	38.1	32.2	34.8	34.4
	−MLP	70.9	76.6	67.1	79.4	62.3
0.5	**intact**	**66.0**	**75.4**	**59.7**	**78.8**	**53.1**
	−attn	29.7	33.1	27.5	28.0	31.5
	−MLP	53.0	62.4	46.8	64.6	41.5
0.25	**intact**	**80.6**	**88.0**	**75.6**	**84.3**	**76.9**
	−attn	26.2	28.4	24.7	25.7	26.7
	−MLP	54.7	61.5	50.2	56.9	52.6
0 (last token)	**intact**	**72.5**	**84.1**	**64.8**	**85.9**	**59.1**
	−attn	11.6	11.7	11.5	11.5	11.6
	−MLP	45.0	51.5	40.6	56.1	33.8

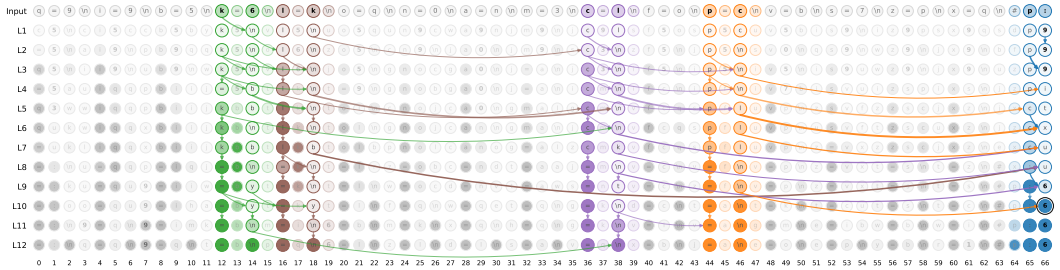
Figure 14 shows attention flows for all the supervision densities. As can be observed, MLP-dominated solutions tend to not respect the natural structure of the task, and information flows over the next logical hop. In contrast, attention-dominated solutions ($p = 0$) respect that structure.



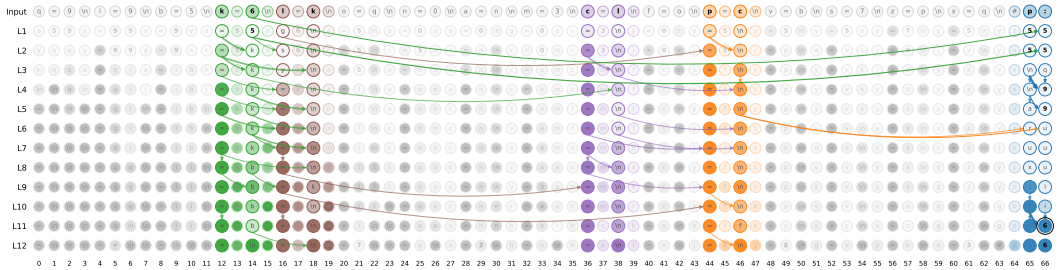
(a) $p = 1$ (all tokens)



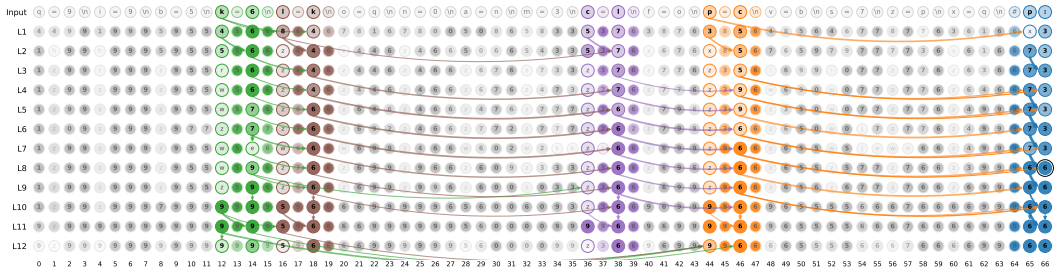
(b) $p = 0.75$



(c) $p = 0.5$



(d) $p = 0.25$



(e) $p = 0$ (last token)

Figure 14: Information flow on one depth-4 query per model with continued learning (300k steps). Each cell is the top-1 logit-lens token at that (layer, position); colored columns are the four lines of the resolution chain, and the circle marks the first layer at which the readout settles on the answer.

Figure 15 shows the logit-lens analysis and reveals distinct solution dynamics across supervision regimes. Dense supervision ($p = 1$) leads to early, confident predictions across positions, whereas last-token supervision ($p = 0$) keeps intermediate representations comparatively uncommitted until later layers. Interestingly, ($p = 0.25$) preserves the lexical structure of the sequence across more layers than the all-token model, suggesting that partial supervision maintains task-format information while still supporting deeper computation.

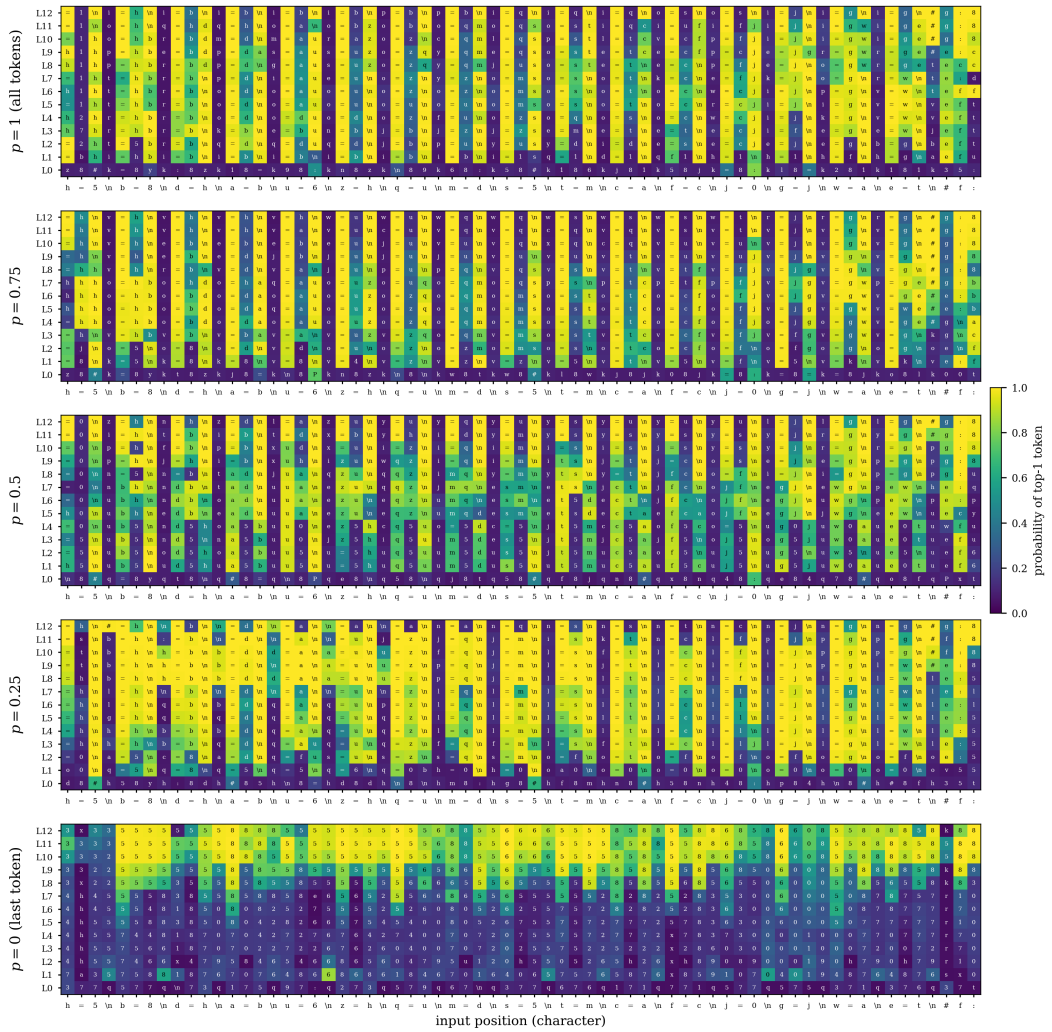


Figure 15: Logit lens across supervision densities (first maximal accuracy models): top-1 token and its probability at each layer and position. Dense supervision ($p = 1$) produces early, confident predictions at all positions, whereas last-token supervision ($p = 0$) keeps intermediate predictions uncommitted. Supervising a quarter of the tokens ($p = 0.25$) captures the lexical format in more layers than the all-tokens model.

C.3 SUBSPACES OF VARIABLES AND NUMBERS

Inspired from Wu et al. (2025), we ask whether the residual stream represents variable identity and numbers in separate low-dimensional subspaces. We use the hidden states of the right-hand-side (RHS) character of every query chain line. For a query with n hops, let $\rho_1, \ldots, \rho_n$ be the rows that the query has to dereference ordered as; ρ_1 is the line that defines the queried name and ρ_n the line that holds the constant. Using this notation and because of the homogeneous sequence structure, we can define the RHS character of each chain line as $\pi_k = 4\rho_k + 2$ for $k = 1, \ldots, n$. For all sequences, those characters are a variable name on the first $n-1$ chain lines and a digit on the last one, which is the answer.

For consistency with Wu et al. (2025), programs whose constant sets on the first line are discarded, since they could be solved by reading the first line without dereferencing anything (in our test split, this correspond to removing 3,379 sequences from the analysis).

At the position π_k the residual stream is split with the exact decomposition of (8) and one pathway at a time is kept, together with the embedding shared by both:

$$\boldsymbol{x}_\ell^{\mathrm{attn}} = \boldsymbol{h}_{\pi_k}^{(0)} + \sum_{i<\ell} \boldsymbol{a}_{\pi_k}^{(i)}, \qquad \boldsymbol{x}_\ell^{\mathrm{mlp}} = \boldsymbol{h}_{\pi_k}^{(0)} + \sum_{i<\ell} \boldsymbol{m}_{\pi_k}^{(i)}, \qquad \boldsymbol{x}_\ell^{\mathrm{attn}} + \boldsymbol{x}_\ell^{\mathrm{mlp}} - \boldsymbol{h}_{\pi_k}^{(0)} = \boldsymbol{h}_{\pi_k}^{(\ell-1)} \quad (27)$$

Note that the above formulas consider the layer index such as the input of those blocks, i.e. the embedding plus the first five or the first eleven outputs of one pathway, and the identity on the right of (27) is asserted numerically for every model before the analysis runs. One caution follows from the definition: the sum of MLP outputs is not a circuit in isolation: $\boldsymbol{m}^{(i)}$ is computed from the full stream at block i, which already contains the attention outputs, so (27) decomposes *what each family of modules wrote*, not what it computes on its own.

The two labels. Each of the considered vectors carries two labels: the letter written on the left of its line, $y_{\mathrm{var}} \in \{a, \ldots, z\}$, and, on the last line of each chain only, the digit written at that very position, $y_{\mathrm{num}} \in \{0, \ldots, 9\}$. The two are not symmetric. The digit sits at the position being read, so the embedding alone identifies it and the question is whether a pathway keeps it or drowns it. The letter sits four characters earlier, so any information about it at π_k had to be *moved there* by attention, and the question is whether a pathway exposes it.

The two subspaces. For every combination of model, checkpoint, **pathway** and layer, the matrix of vectors is reduced by PCA to 256 components and the coordinates are standardised, so that an L_1 penalty weighs every component by its discriminative power and not by its variance. Two multinomial logistic regressions are then fitted:

$$\min_{\boldsymbol{W},\boldsymbol{b}} \sum_i \mathrm{CE}\big(\mathrm{softmax}(\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{b}),\, y_i\big) + \frac{1}{C}\|\boldsymbol{W}\|_1, \quad (28)$$

f_{var} on all chain lines with $y = y_{\mathrm{var}}$, and f_{num} on the last line of each chain with $y = y_{\mathrm{num}}$. The penalty is not a free parameter: C is found by bisection on $\log C$ until exactly K of the 256 columns of $\boldsymbol{W}$ are non-zero, with $K = 26$ for the variable and $K = 10$ for the digit, the sizes reported by the original analysis.

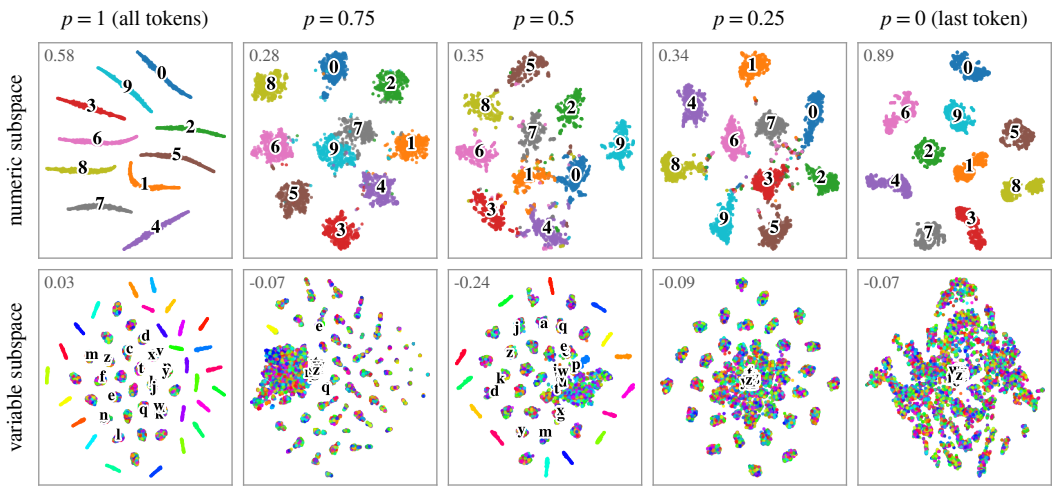
The *variable subspace* and the *numeric subspace* are the spans of the two selected sets of principal components; because the PCA basis is orthonormal, disjoint selections mean orthogonal subspaces, and the Jaccard index of the two selections is 0 in almost every panel.

How a subspace is judged. Each subspace is drawn as a two-dimensional t-SNE embedding of the coordinates of the lines in it, colored and labeled by the label that subspace was selected for, and scored by the *silhouette* of those labels in the subspace itself,

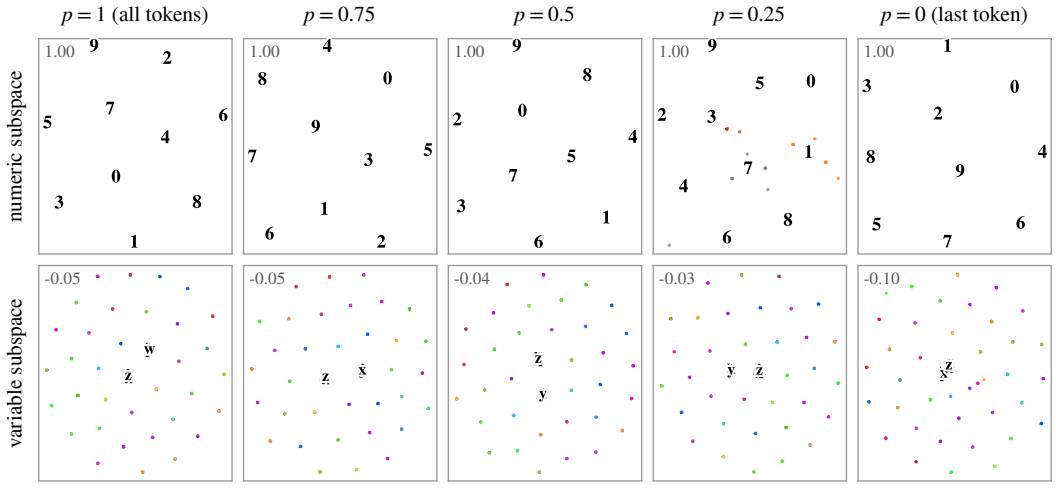
$$s = \frac{1}{m}\sum_{i=1}^{m} \frac{b_i - a_i}{\max(a_i, b_i)}, \quad (29)$$

with a_i the mean Euclidean distance from line i to the lines of its own class and b_i the smallest mean distance to another class. It lies in $[-1, 1]$: near 1 for tight, well separated classes, near 0 for overlapping ones, negative when points sit closer to another class than to their own.

Below we provide the 2D t-SNE visualizations of the subspaces related to variable names and numerical constants both for the representations at the level of attention and MLP components for Layer 2 (Figure 16), 6 (Figure 17), and 12 (Figure 18). As can be observed, last-token learning generates stronger separation of the subspaces representing both numeric constants and variable names, particularly at the level of attention-related representations.

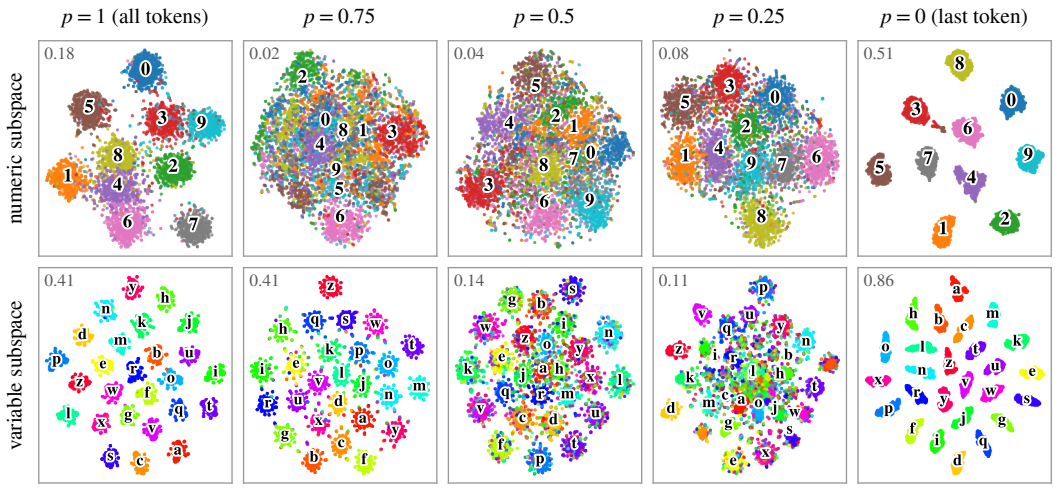


(a) Attention

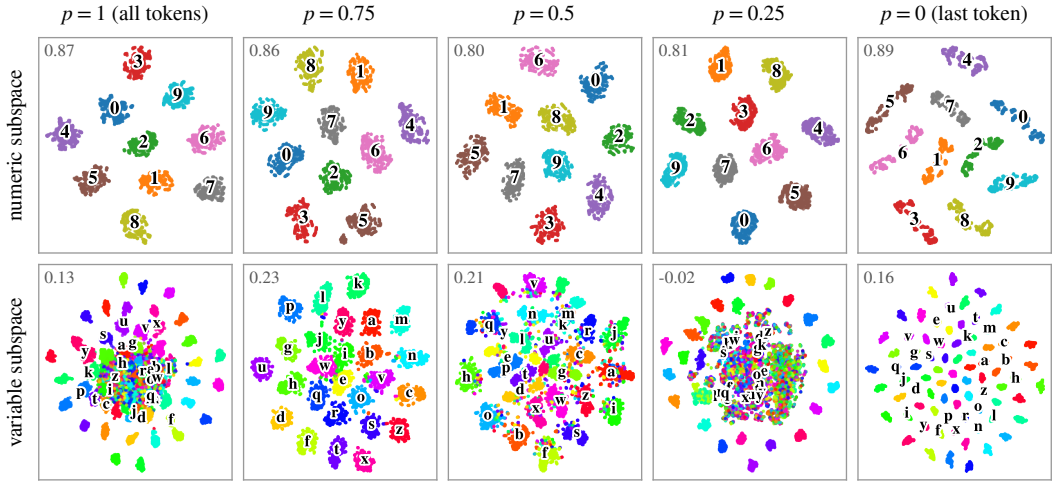


(b) MLP

Figure 16: Numeric and variable subspaces at layer 2 input per model with continued learning (300k steps) on a subsample of 5,000 test programs, following Wu et al. (2025): PCA to 256 dimensions, L_1-regularized probes to select the components predictive of the 10 constants and the 26 variable names, then t-SNE. Values are silhouette coefficients.

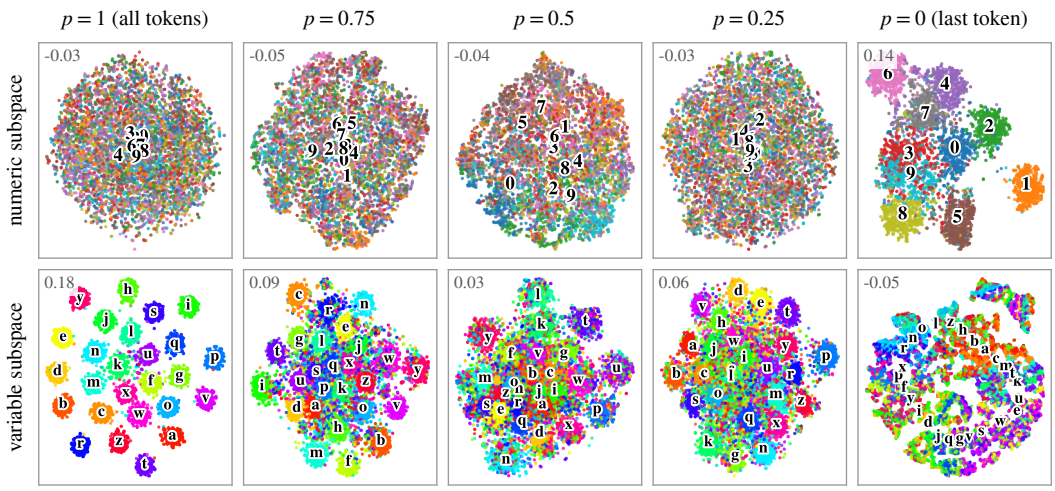


(a) Attention

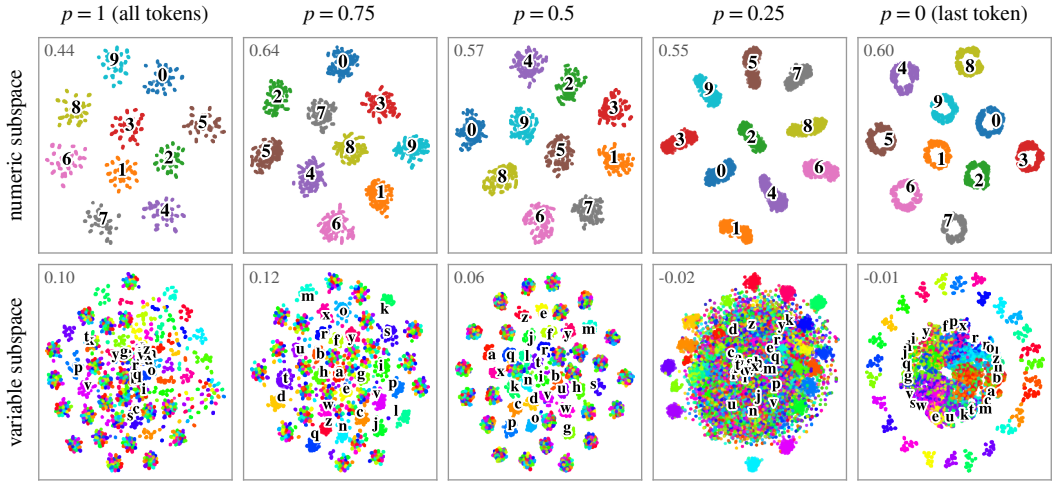


(b) MLP

Figure 17: Numeric and variable subspaces at layer 6 input per model with continued learning (300k steps) on a subsample of 5,000 test programs, following Wu et al. (2025): PCA to 256 dimensions, L_1-regularized probes to select the components predictive of the 10 constants and the 26 variable names, then t-SNE. Values are silhouette coefficients.



(a) Attention



(b) MLP

Figure 18: Numeric and variable subspaces at layer 12 input per model with continued learning (300k steps) on a subsample of 5,000 test programs, following Wu et al. (2025): PCA to 256 dimensions, L_1-regularized probes to select the components predictive of the 10 constants and the 26 variable names, then t-SNE. Values are silhouette coefficients.

Table 4 displays the silhouette values for numerical constants and variable names for all supervision densities, both at the first maximal accuracy model and continued training (step 300k).

Table 4: Silhouette (29) of each subspace on the $49{,}000$ test programs, for the two pathways of (27) and the inputs of layers 6 and 12.

Checkpoint	Layer	Subspace	$p = 1$	$p = 0.75$	$p = 0.5$	$p = 0.25$	$p = 0$
first maximal accuracy	6	MLP, digit	0.90	0.98	0.97	0.96	0.86
		MLP, variable	0.11	0.39	0.17	-0.08	0.11
		attention, digit	0.16	-0.02	-0.07	0.00	0.37
		attention, variable	0.30	0.56	0.29	-0.03	0.78
	12	MLP, digit	0.20	0.88	0.94	0.85	0.45
		MLP, variable	0.17	0.27	0.40	-0.03	-0.01
		attention, digit	-0.02	0.21	0.09	-0.03	-0.13
		attention, variable	0.28	0.20	0.18	-0.00	-0.07
step 300k	6	MLP, digit	0.87	0.85	0.80	0.81	0.89
		MLP, variable	0.13	0.24	0.21	-0.02	0.16
		attention, digit	0.18	0.03	0.04	0.08	0.51
		attention, variable	0.41	0.41	0.13	0.11	0.86
	12	MLP, digit	0.43	0.64	0.56	0.55	0.60
		MLP, variable	0.10	0.12	0.06	-0.02	-0.02
		attention, digit	-0.03	-0.05	-0.04	-0.03	0.14
		attention, variable	0.18	0.09	0.02	0.06	-0.05

C.4 MARGIN BOUNDS

We verify the bound of Section B.5.3 on the trained models and use it to interpret the logit margin attributed in Section 3.2. Splitting the logit-lens margin (14) into its two terms separates two behaviors (Figure 19). In the four models with $p > 0$, the mean logit of the nine wrong digits stays near zero through the network, and the margin is built by lifting the correct digit alone. The last-token model instead raises all ten digit logits after a single layer, committing early to emitting a digit, and opens its margin in the last layers by letting the nine wrong digits fall. Because a shift shared by all ten digit logits cancels in the margin, the margin, rather than the correct-digit logit, is the quantity we attribute to the pathways.

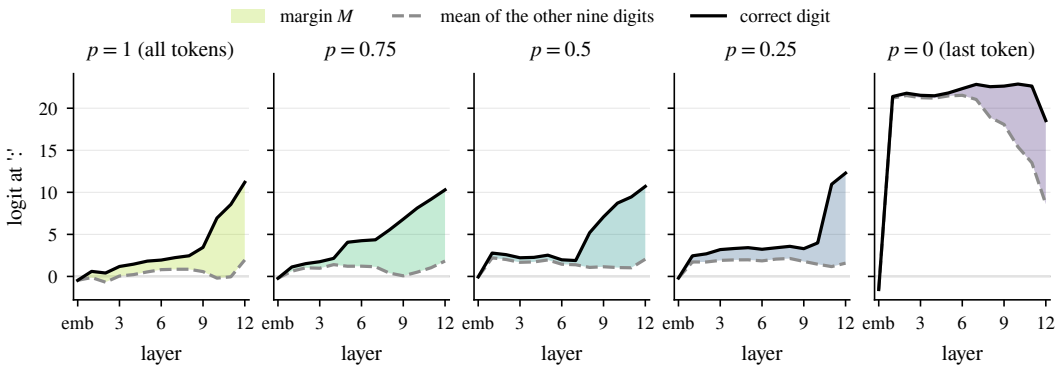


Figure 19: The two terms of the margin (14) at the first maximal accuracy checkpoint: the logit of the correct digit (black) and the mean logit of the other nine (gray, dashed), read after ℓ blocks at the `":"`. The shaded area is the margin $M^{(\ell)}$.

The bound of Section B.5.3 holds on every model, checkpoint and test sequence, and is never reached: over 3.9 million readouts ($49{,}000$ test sequences $\times$ 16 checkpoints $\times$ 5 models), the largest saturation is 0.97 (Figure 20). It becomes nearly tight at the transition. The median saturation stays below 0.4 before the task is learned; within one checkpoint of reaching 100% validation accuracy, the four models with $p > 0$ approach 0.9 and then barely move. Their later growth in margin

(Table 2) therefore comes almost entirely from the rising ceiling $\sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^{\Delta}$, i.e., from the unembedding. The last-token model is the exception: its saturation keeps rising after convergence, from 0.79 to 0.903 at step 300k.

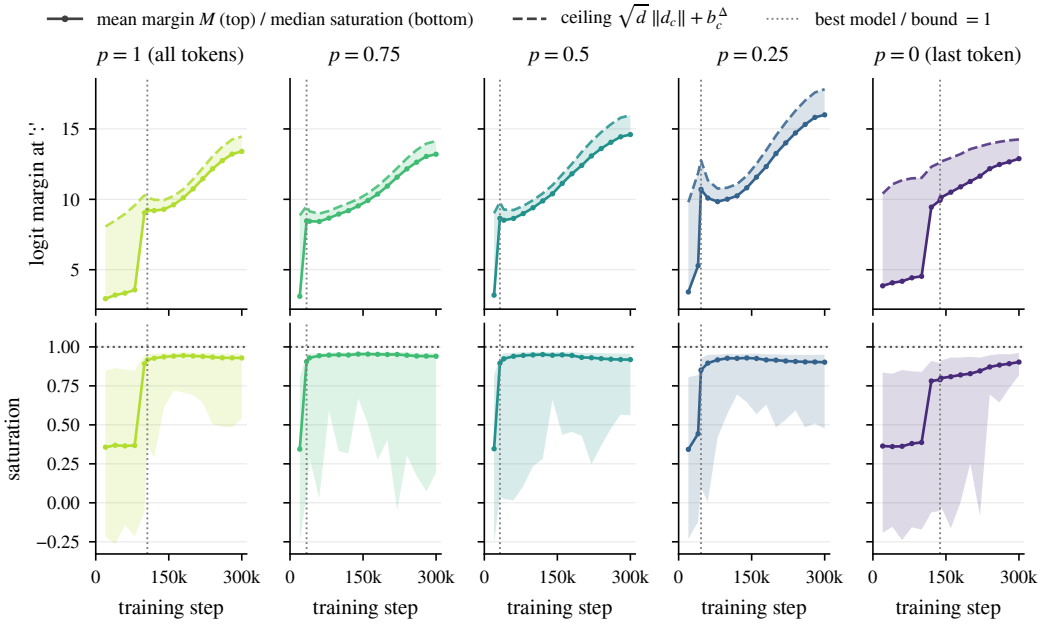


Figure 20: The bound across training. Theorem 1 checked on every model, one column per model. Top: mean logit margin (solid) against its ceiling $\sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^{\Delta}$ (dashed); the shaded area between them is the headroom the readout leaves unused. Bottom: saturation (17), median over the 49,000 test sequences, with the band spanning their minimum and maximum; the dotted line at 1 is the bound. Dotted lines mark the first maximal accuracy checkpoint.

Against depth (Figure 21), the ceiling is the same at every layer, so saturation compares depths and models on a single scale. Every model starts near 0 and ends between 0.79 and 0.92; all five spend most of their budget after layer 7, and only $p = 0.75$ has used more than a fifth of it before. Because the lens divides by the final scale σ_L, Theorem 1 does not guarantee the bound at $\ell < L$, but it holds empirically: the largest $S^{(\ell)}$ over all sequences, depths and models is 0.954.

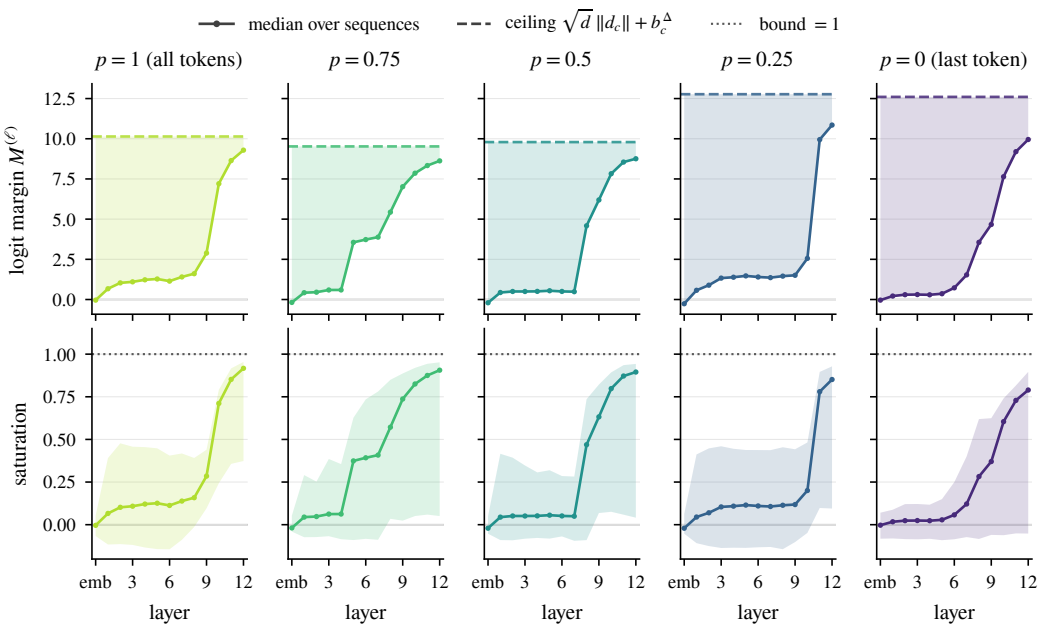


Figure 21: The bound across depth, at the first maximal accuracy checkpoint. Top: median logit-lens margin $M^{(\ell)}$ against the ceiling $\sqrt{d}\,\|\boldsymbol{d}_c\| + b_c^{\Delta}$, which does not depend on ℓ; the shaded area is the headroom still available at that depth. Bottom: saturation $S^{(\ell)}$, median over the 49,000 test sequences, with the band spanning their minimum and maximum; the dotted line at 1 is the bound.