Interpreting the Latent Structure of Operator Precedence in Language Models

Dharunish Yugeswardeenoo **Harshil Nukala** **Cole Blondin**
Sean O'Brien **Vasu Sharma** **Kevin Zhu**

Algoverse AI Research
3101 Park Blvd, Palo Alto, CA 94306, USA
dharyugi@gmail.com, harshiln14@gmail.com, cole@algoverseairesearch.org
seobrien@ucsd.edu, sharma.vasu55@gmail.com, kevin@algoverseairesearch.org

Abstract

Large Language Models (LLMs) have demonstrated impressive reasoning capabilities but continue to struggle with arithmetic tasks. Prior works largely focus on outputs or prompting strategies, leaving the open question of the internal structure through which models do arithmetic computation. In this work, we investigate whether LLMs encode operator precedence in their internal representations via the open-source instruction-tuned LLaMA 3.2-3B model. We constructed a dataset of arithmetic expressions with three operands and two operators, varying the order and placement of parentheses. Using this dataset, we trace whether intermediate results appear in the residual stream of the instruction-tuned LLaMA 3.2-3B model. We apply interpretability techniques such as logit lens, linear classification probes, and UMAP geometric visualization. Our results show that intermediate computations are present in the residual stream, particularly after MLP blocks. We also find that the model linearly encodes precedence in each operator's embeddings post attention layer. We introduce partial embedding swap, a technique that modifies operator precedence by exchanging high-impact embedding dimensions between operators.

1 Introduction

Large Language Models (LLMs) have shown impressive reasoning across a wide range of language tasks (Wei et al. (2022); Chowdhery et al. (2022); OpenAI et al. (2024)). Yet they are notorious for struggling with arithmetic reasoning, often producing incorrect calculations or implausible outputs (Mirzadeh et al. (2024); Bubeck et al. (2023)). Particularly, these errors are prominent in smaller models and remain poorly understood (Gangwar et al. (2025); Kim et al. (2024)). While recent work has shed light on how MLP layers and attention heads contribute to arithmetic reasoning, most of the studies focused on the natural language prompts and correct outputs, overlooking the processing of operator precedence and intermediate calculation steps (Zhang et al. (2024); Stolfo et al. (2023); Zhao et al. (2024)).

We examine LLMs beyond just natural language framing and understand how arithmetic expressions are internally processed. Specifically, we focus on the model's ability of handling order of operations in its step-by-step computation. For instance, when prompting the model to evaluate expressions like $1 + 1 \times 2$, does the model compute 1×2 before the addition operation, or does it treat the prompt in a linear sequence irrespective of mathematical hierarchy?

We employ a broad set of interpretability techniques including the logit lens, linear probes, partial embedding swaps, and geometric visualization via UMAP (nostalgebraist (2020); Alain & Bengio (2018); McInnes et al. (2020)). All experiments are conducted on the open-source, instruction-tuned LLaMA 3.2-3B model.

2 Related Works

Arithmetic Reasoning in Language Models. While LLMs demonstrate strong general reasoning abilities, numerous prior works have shown persistent inconsistencies for arithmetic tasks, especially when the prompted operations require multi-step computation and manipulation (Mirzadeh et al. (2024); Bubeck et al. (2023); Zhao et al. (2024)). These failures are particularly underlined in smaller models and remain poorly understood (Gangwar et al. (2025); Kim et al. (2024)). In a recent study (Boye & Moell (2025)), the authors evaluated arithmetic computations across multiple different models and observed frequent inconsistencies, such as over-reliance on numerical patterns and flawed logic, even when final answers were correct. In another study (Lewkowycz et al. (2022)), the authors have explored the use of prompting strategies, like chain-of-thought reasoning to support numerical computation.

Mechanistic Interpretability and Internal Components. Research in mechanistic interpretability has focused on identifying functional components, often referred to as "circuits", that are responsible for specific model behaviors. A circuit refers to a subnetwork of model components that offers a faithful representation of how a model solves a particular task, such as a mathematical computation (Nainani et al. (2024)). For example, Zhang et al. (2024) investigates the internal structure of LLMs during arithmetic tasks, showing that fine-tuning a small subset of attention heads and MLPs can enhance arithmetic performance without compromising other abilities. These components are consistent across tasks and leads to overall better arithmetic performances. (Stolfo et al. (2023)) adopts a causal mediation framework to trace the influence of certain internal components that contribute the most to arithmetic prediction. However, their focus is limited in scope to focus on final predictions, rather than analyzing intermediate computational steps.

3 Experimental Procedure

3.1 Model

In this work, we utilize the open-source, instruction-tuned LLaMA 3.2-3B model (Meta Platforms, Inc. (2024)).

Dataset Creation. We constructed a synthetic dataset consisting of arithmetic expressions with three operands and two binary operators. The dataset was designed to systematically test both syntactic and semantic precedence. We consider operands $a, b, c \in \{1, 2, \ldots, 9\}$ and select operator pairs (o_1, o_2) from the following mixed-precedence sets: { (+, *), (-, *), (+, /), (-, /) } Only mixed-precedence operator combinations are used to ensure meaningful precedence distinctions. For each combination of operands and operator pairs, we generate six structural variations: **left-parenthesized:** $(a\ o_1\ b)\ o_2\ c$, **right-parenthesized:** $a\ o_1\ (b\ o_2\ c)$, **flipped left-parenthesized:** $(a\ o_2\ b)\ o_1\ c$, **flipped right-parenthesized:** $a\ o_2\ (b\ o_1\ c)$, **no-parentheses (natural order):** $a\ o_1\ b\ o_2\ c$, **no-parentheses (flipped):** $a\ o_2\ b\ o_1\ c$. These expressions allow us to isolate the model's handling of precedence both with and without explicit grouping via parentheses. For simplicity, prompts were selected such that all calculations, including intermediate steps, involved only positive whole numbers. In total 8547 prompts were created, but only prompts in which the model could predict the correct answer as the top logit were used to accurately examine model behavior. The model could answer 4401 equations correctly.

Logit Lens to Trace Intermediate Computation. Before assessing how the model encodes operator precedence, we first examine whether it performs intermediate computations internally prior to generating the final output. For example, given the prompt "2 + 3 * 3 =", we investigate whether the model computes the intermediate product $3 \times 3 = 9$ before arriving at the final result of 11. To account for linguistic variability in output tokens, we consider multiple surface forms (e.g., "11", "eleven", "eleventh") as valid representations of the intermediate value. Using a dataset of 4,401 prompts for which the model produces correct final answers, we utilize the logit lens technique (nostalgebraist (2020)), projecting each layer's residual stream through the model's unembedding matrix to obtain logits over

the vocabulary. We then extract the top 10 tokens by logit magnitude and check whether any of them match the expected intermediate result.

Linear Probe for Latent Intermediate Computation. It is possible that the model performs intermediate computations internally, even when the corresponding value does not appear among the top logits. To further investigate this hypothesis, we train a linear probe (Alain & Bengio (2018)) to predict the intermediate value directly from the model's activations at each layer, providing a complementary measure of whether such computations are linearly encoded.

Operator Precedence in Embedding. To investigate whether operator precedence is encoded in specific dimensions of the model's internal representations, we introduce partial embedding swap. Consider the prompt "3 + 4 * 5 = ", where the correct evaluation yields the answer 23, adhering to standard arithmetic precedence. However, evaluating the expression strictly left-to-right results in the incorrect answer 35. We apply the algorithm described in Appendix A.2 to probe the embedding space. We selectively swap individual dimensions between the hidden representations of the "+" and "*" operator tokens. If such a perturbation induces the model to shift its prediction from 23 to 35, it provides evidence that those dimensions contribute to encoding precedence. This intervention is performed incrementally by swapping one dimension at a time and measuring the resulting change in the logit score assigned to the token "35". Dimensions are then ranked according to their influence on increasing this logit. In the final phase of the experiment, we perform prefix swaps of the top k most influential dimensions to determine the minimal subset required to elevate "35" to the top-ranked prediction.

Low-dimensional Projection of Embeddings. UMAP is a non-linear dimensionality reduction technique that facilitates the visualization and analysis of high-dimensional data by projecting it into a low-dimensional space (McInnes et al. (2020)). While specific embedding dimensions appeared to correlate with operator precedence, we employed UMAP to project the activation vectors corresponding to operator tokens from a curated set of prompts. These prompts varied only in the type and position of operators, allowing for controlled comparisons. Importantly, only prompts for which the model produced the correct answer were included, ensuring that the visualizations reflected meaningful internal representations. We labeled each operator using the following format: *[position of the operator] [operator name] [precedence applied to the operator]*. E.g., for a prompt "2 * (3 + 4) = ", the labels are "1m2" and "2p1". The multiplication sign appears first in the expression but will be evaluated second, and the plus sign appears second in the equation but will be evaluated first.

Logistic Probe to Identify Precedence. We employ linear probes (Alain & Bengio (2018)) to investigate whether operator precedence is linearly encoded in the internal representations of the model. We construct a dataset of arithmetic prompts containing two operations whose evaluation order determines the final result. To ensure alignment between model behavior and target labels, we include only correctly answered prompts. For each valid prompt, we extract activations at operator token positions from both pre- and post-attention within layer 0. A logistic regression classifier was trained to predict whether a given arithmetic operator (e.g., +, *) corresponds to the first or second operation to be evaluated in a two-step arithmetic expression (e.g., distinguishing between "(2 + 3) * 4 = " and "(2 * 3) + 4 = "). Input to the probe consists of activation vectors associated with operator tokens in the prompt. The model is trained to predict either 0 or 1 based on whether the operator is evaluated first or second, respectively. For each probe, the activation vectors and binary labels are split into training and test sets (80%/20%). By comparing probe accuracy across the two extraction points (before and after attention), we assess whether and to what extent attention enhances the linear decodability of operator precedence.

4 Results and Analysis

Intermediate Calculation. Out of 4401 prompts, the intermediate calculation appeared 2799 times as the top logit, roughly 63.6%. The layers in which the calculation was discovered ranged from layer 16 - 27 for the LLaMA 3.2-3B model. This distribution is shown in Figure 1. To investigate whether it is the attention block or the MLP layer that introduces the

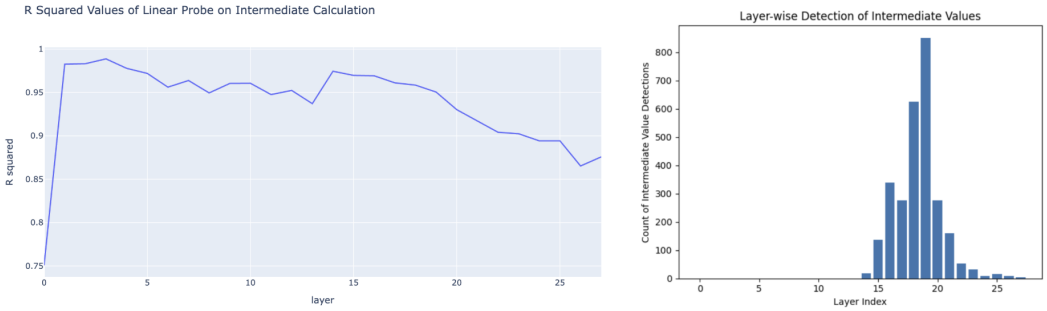

Figure 1: (Left) shows that intermediate calculations are linearly encoded in the model's activations after layer 0, as indicated by high R^2 scores. (Right) Layer-wise detection frequency of intermediate computations in the residual streams at each layer. Detection peaks around layers 18–19, suggesting that the model most strongly represents intermediate computations in the later layers of its forward pass.

Figure 2: Low dimensional projection of operator token embeddings before (left) and after (right) the attention block in layer 0. After attention, embeddings separate by operator position and precedence, suggesting that attention encodes operator precedence information

intermediate logit, we apply the unembedding matrix to the outputs of each component at the layer where the intermediate logit first appears. In all cases, we find that the intermediate answer token's logit becomes the top-ranked logit only after the MLP block, indicating that this component is responsible for producing the intermediate computation. Figure 1 strongly suggests that the intermediate calculation is linearly encoded in the model activations after layer 0.

Operator Precedence. We were able to successfully alter the model's highest logit in multiple instances using partial embedding swap, examples shown in Appendix A.3. Our UMAP projection is shown in Figure 2. We found that operators who matched in both position and precedence were clustered near each other. Our logistic regression probe recieved 100% accuracy on our test set, strongly indicating the presence of operator precedence.

Conclusion

In this work, we analyze the internal representations of the LLaMA 3.2-3B model when processing arithmetic expressions involving three operands. Our findings indicate that the model performs intermediate computations internally, with such information becoming most linearly decodable in the deeper layers of the network. Through probing and intervention, we identify specific embedding dimensions that plausibly encode operator precedence, and demonstrate that modifying these dimensions via partial embedding swaps can systematically alter the model's arithmetic predictions. Additionally, UMAP projections

of operator token embeddings reveal that the model organizes operations based on both their position in the expression and their evaluation precedence.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018. URL https://arxiv.org/abs/1610.01644.

Johan Boye and Birger Moell. Large language models and mathematical reasoning failures, 2025. URL https://arxiv.org/abs/2502.11574.

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. URL https://arxiv.org/abs/2303.12712.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Neeraj Gangwar, Suma P Bhat, and Nickvash Kani. Integrating arithmetic learning improves mathematical reasoning in smaller models, 2025. URL https://arxiv.org/abs/2502.12855.

Bumjun Kim, Kunha Lee, Juyeon Kim, and Sangam Lee. Small language models are equation reasoners, 2024. URL https://arxiv.org/abs/2409.12393.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL https://arxiv.org/abs/1802.03426.

Meta Platforms, Inc. Meta Llama 3.2 3B Instruct Model Card. https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct, 2024. Hugging Face Model Card for Llama 3.2 3B Instruct.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

Jatin Nainani, Sankaran Vaidyanathan, AJ Yeung, Kartik Gupta, and David Jensen. Adaptive circuit behavior and generalization in mechanistic interpretability, 2024. URL https://arxiv.org/abs/2411.16105.

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

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis, 2023. URL https://arxiv.org/abs/2305.15054.

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, 2022. URL https://arxiv.org/abs/2206.07682.

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation, 2024. URL https://arxiv.org/abs/2409.01659.

Haiyan Zhao, Fan Yang, Bo Shen, Himabindu Lakkaraju, and Mengnan Du. Towards uncovering how large language model works: An explainability perspective, 2024. URL https://arxiv.org/abs/2402.10688.

A Appendix

A.1 Logit Lens Visual

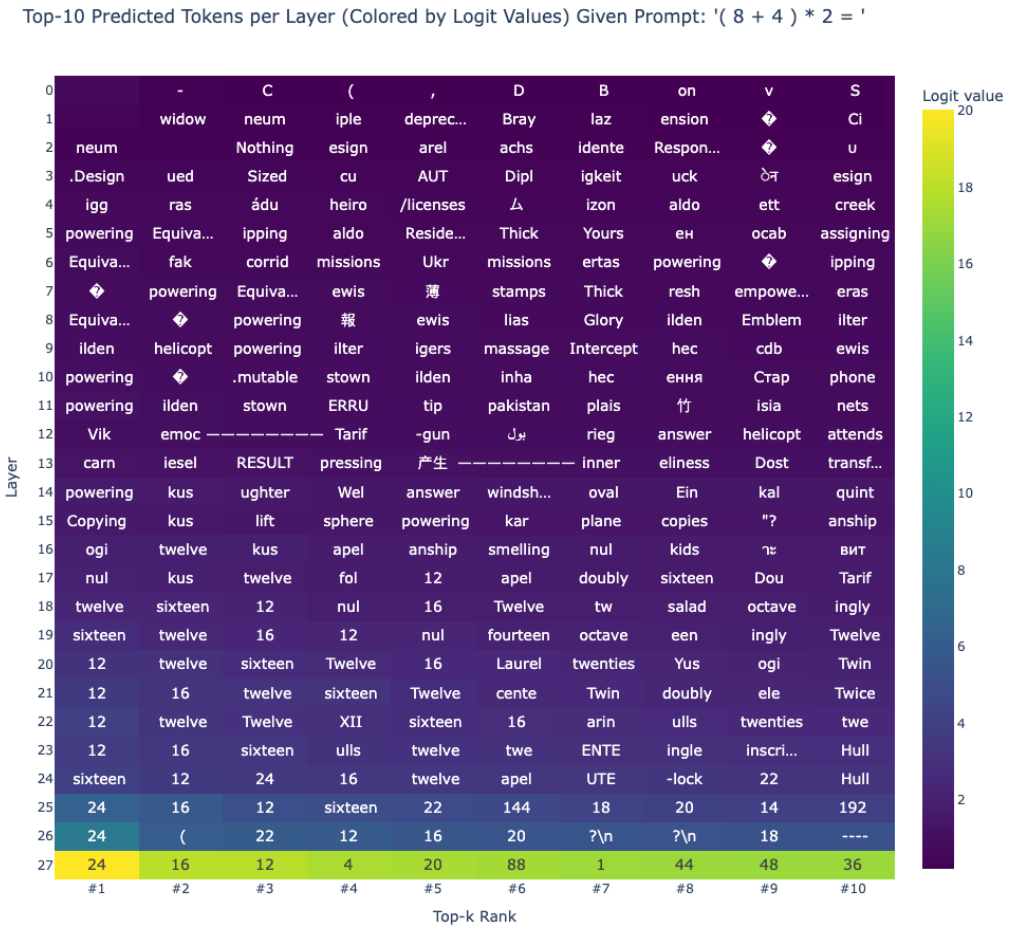

Figure 3: This figure presents the top-10 tokens ranked by their logit scores obtained via the logit lens method after each transformer layer. For the input expression under consideration, the intermediate computed value is ”12”. Notably, the token corresponding to ”12” (or its lexical variant ”twelve”) emerges prior to the final output token. This observation provides evidence of the model internally representing intermediate computational steps before producing the final answer.

A.2 Partial Embedding Swap Algorithm

Algorithm 1 Identifying Influential Embedding Dimensions via Activation Swapping

Input: Prompt P, model $\mathcal{M}$, dimension count d, top-k value `topk`, Operator 1 Position Pos_1, Operator 2 Position Pos_2
Output: Sorted list of top contributing dimensions

Compute precedence-consistent answer token t_{target} and left-to-right answer token t_{real}
Compute original logits $\ell^{\text{orig}} \leftarrow \mathcal{M}(P)$
Extract original logit for target token: $\ell^{\text{orig}}_{\text{target}} \leftarrow \ell^{\text{orig}}[-1, t_{\text{target}}]$
Initialize contribution vector $C \leftarrow [0] \in \mathbb{R}^d$
for each dimension index $e = 1$ to d **do**
  Define hook that swaps dimension e of Pos_1 and Pos_2 tokens in the residual stream
  Compute patched logits: $\ell^{(e)} \leftarrow \mathcal{M}(P)$ with hook applied at layer 0
  Extract patched logit: $\ell^{(e)}_{\text{target}} \leftarrow \ell^{(e)}[-1, t_{\text{target}}]$
  Set contribution: $C[e] \leftarrow \ell^{(e)}_{\text{target}} - \ell^{\text{orig}}_{\text{target}}$
end for
Rank dimensions by descending $C[e]$ and select top-`topk`
return Sorted list of top contributing dimensions and their scores

Algorithm 2 Cumulative Dimension Patching to Shift Model Prediction

Input: Prompt P, model $\mathcal{M}$, sorted contribution list $[(e_1, \Delta\ell_1), \ldots, (e_d, \Delta\ell_t)]$
Output: Number of dimensions needed to force precedence-aligned output

Compute original prediction $\hat{y}_{\text{orig}} \leftarrow \arg\max \mathcal{M}(P)[-1]$
for $k = 1$ to t **do**
  Define hook that swaps dimensions $e_1, \ldots, e_k$ of Pos_1 and Pos_2 tokens in the residual stream
  Compute patched logits: $\ell^{(k)} \leftarrow \mathcal{M}(P)$ with hook applied at layer 0
  Extract target logit $\ell^{(k)}_{\text{target}} \leftarrow \ell^{(k)}[-1, t_{\text{target}}]$
  Compute prediction $\hat{y}^{(k)} \leftarrow \arg\max \ell^{(k)}[-1]$
  if $\hat{y}^{(k)} = t_{\text{target}}$ **then**
    return k {Minimal number of swaps needed}
  end if
end for
return {All dimensions swapped but no change observed}

A.3 Partial Embedding Swap Analysis

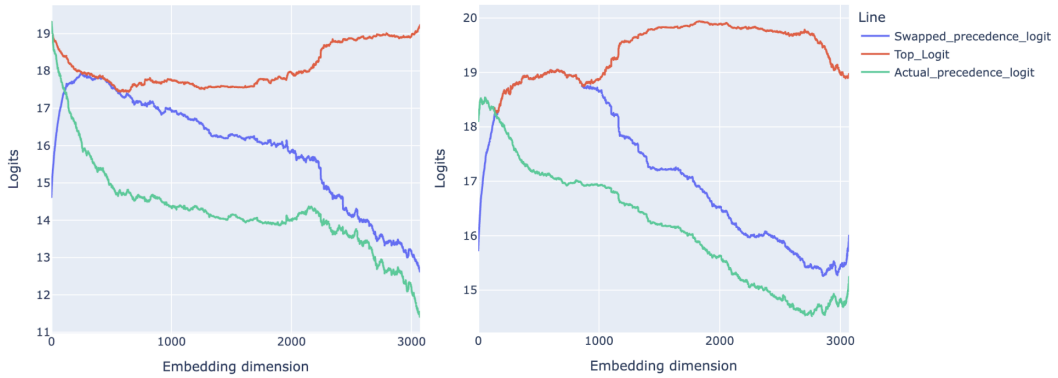

Figure 4: (left) "2 + 3 * 3 = " and (right) "4 + 8 / 4 = ". The *swapped-precedence-logit* (blue) tracks the logit of the swapped (incorrect precedence) answer after patching, the *actual-precedence-logit* (green) tracks the logit of the correct answer, and the *top-logit* (red) shows the logit of the model's predicted token. In both examples, a subset of embedding dimensions substantially modulate the precedence-sensitive logit values, indicating that operator precedence information is sparsely localized across specific dimensions of the residual stream activations.