Same Story, Different Circuits: Measuring Component Reuse in Transformer Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

A central premise of mechanistic interpretability is the existence of task-specific circuits within models. This work investigates the consistency of these circuits by asking how often transformer models *reuse* the same internal parts across different examples of a task. We use edge attribution patching to construct per-example circuits composed of attention heads and MLP layers. We then define a metric, **reuse@P**, to quantify how frequently the same components are reused across different circuits of a task. To verify that shared circuits are actually important for task performance, we knock out these components and compare against a strong baseline, which we call **capacity-conserved control** (C^3). If shared circuit knockout causes larger accuracy drops than the control, this confirms the components' causal specificity for the task. We evaluate this approach across 6 tasks including `Addition`, `Boolean Logic`, `Indirect Object Identification`, `CopyColors` `MCQA`, `MMLU` and `AI2 ARC`, and 7 models from 4 model families including `Qwen3`, `Llama 3.2`, `Gemma 2` and `OLMo-2-1B`. We find that circuit reuse is present in all models but varies by task type: algorithmic tasks exhibit higher reuse than knowledge-intensive ones. Our knockout studies confirm the causal importance of shared circuits, but also suggest the presence of functionally redundant pathways within the models. These findings indicate that transformers may solve tasks not with a single, minimal circuit, but with an ensemble of overlapping solutions, opening new directions for understanding generalization and robustness.

1 Introduction

Neural networks are infamously black box; even when we can elicit strong performance on a task, it is unclear which internal computations are responsible. The field of mechanistic interpretability aims to reverse-engineer the algorithms learned by these models, typically by identifying circuits within a model that explain a behavior. Recent work has reverse-engineered circuits responsible for specific behaviors in transformer language models, such as the `Indirect Object Identification` (IOI) circuit in `GPT-2` (Wang et al., 2023), and documented how apparently similar circuit components can be reused across tasks (Merullo et al., 2024). However, it remains unclear how *within-task circuit reuse* behaves.

In this paper, we ask a simple question: when a model solves many instances of the same task, does it rely on the same internal parts? Following the Transformer Circuits framework (Elhage et al., 2021), we define circuit reuse as the set of components (attention heads and MLP layers) that recur across inputs within the same task. We map circuits at scale with edge attribution patching (EAP) (Syed et al., 2024), then identify the components appearing in at least $P\%$ of circuits, defined as **reuse@P**.

To test whether this shared circuit is functionally important, we knock out its components and measure the resulting drop in task accuracy. To properly calibrate these knockouts, we introduce a new baseline:

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

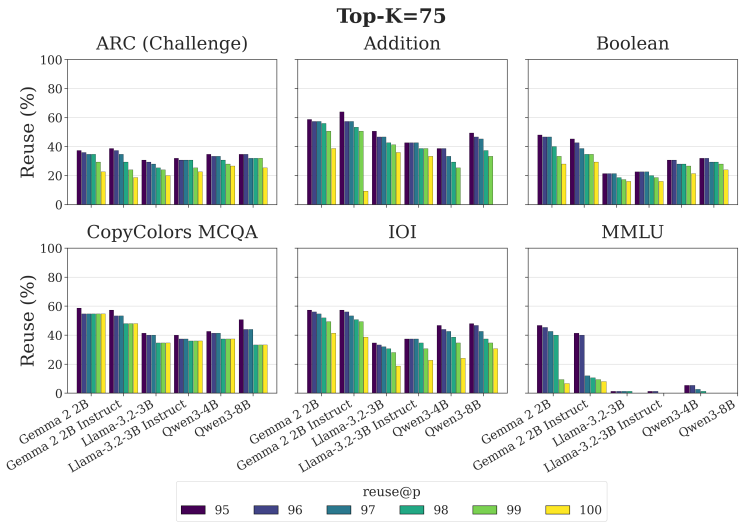


Figure 1: **Circuit reuse varies by task type for top-K=75.** reuse@P measures the percentage of circuit components appearing in at least P% of task example circuits. Reuse is high for algorithmic tasks like addition, and low for knowledge-intensive tasks like MMLU.

the **Capacity-Conserved Control** (C^3), which removes an equal number of randomly sampled components. This control helps verify that any observed effects reflect functional specialization, not just generic capacity reduction. We evaluate circuit reuse on a variety of tasks (e.g. `Addition`, `Boolean Logic`, `IOI`, `CopyColors MCQA`, `AI2 ARC`, `MMLU`) with a wide range of open-weight models (e.g. `OLMo-2-1B`, `Gemma 2 2B`, `Llama 3.2 3B`, `Qwen3 8B`).

Our paper makes several contributions:

1. We propose **reuse@P**, a useful metric for quantifying the degree of circuit reuse across a distribution of inputs for a specific task.
2. We introduce the **Capacity-Conserved Control** (C^3), a strong baseline for circuit knockouts that helps calibrate whether observed performance drops are actually due to the specific choice of components, not just the reduction in model capacity.
3. We conduct a large-scale empirical study of circuit reuse trends across seven distinct tasks and seven transformer models (differing in size, architecture and post-training).

2 Background

Transformer Circuits. The Transformer Circuits framework is widely used for mechanistic interpretability to reason about model computations and component-level interventions (Elhage et al., 2021). Within this framework, a decoder-only transformer can be viewed as a directed computation over layers operating on a central residual stream. The residual stream acts as a high-dimensional communication channel where each layer reads information via linear projections and writes results back through additive contributions. Attention heads and MLP layers are treated as the primary computational components. A circuit is then defined as a sparse subgraph of components and their connecting edges that is sufficient, and minimal, for implementing a given behavior.

Edge Attribution Patching. To identify which components form a circuit for a given behavior, researchers typically use causal intervention methods – systematically ablating or patching activations to measure each component's contribution to the model's output. `Edge Attribution Patching (EAP)` is an efficient automated circuit discovery method that approximates the effect of activation patching (a computationally expensive technique) using gradient information (Syed et al., 2024). `EAP` estimates the importance of all edges using just two forward passes and one backward pass by computing attribution scores based on activation differences and gradients. Edges are then ranked by

their absolute attribution scores and those below a given top-K threshold are pruned to identify the minimal circuit.

Tasks. We evaluate circuit reuse across a diverse set of tasks designed to probe model capabilities. They can be grouped into three main categories:

- **Algorithmic Tasks:** These tasks follow a clear, learnable algorithm. We use 3-digit `Addition` and `Boolean Logic` evaluation. High circuit reuse is expected as the underlying algorithm is consistent across all inputs.
- **Linguistic & Reasoning Tasks:** These require understanding sentence structure and context. We use `Indirect Object Identification (IOI)` (Wang et al., 2023) and `CopyColors MCQA`.
- **Knowledge-Intensive Tasks:** These tasks require recalling factual knowledge, potentially stored heterogeneously across the model's parameters. We use the high school European history subset from `MMLU` (Hendrycks et al., 2021) and `AI2 ARC` (Clark et al., 2018).

3 Methodology

To investigate within-task circuit reuse, we construct shared circuits for each task to identify important components across multiple examples. For each task, we sample $n = 1000$ input/output pairs from the training set and extract per-example circuits using edge attribution patching (EAP), which produces importance scores for every attention head and MLP layer. We define the per-input circuit as the top-K components by score for each example.

Given these per-input circuits from the training data, we construct a shared circuit at threshold P containing components that appear in at least P% of training examples. We quantify the extent of this sharing using **reuse@P**: the fraction of a typical circuit covered by components appearing in at least P% of examples (formal definition in Appendix B). For example, if 75% of the top-K components appear in at least 95% of task examples, then reuse@95 = 75%. This measures circuit consistency across the task distribution - higher values indicate that models rely on the same components repeatedly.

To validate that shared circuits are causally important rather than just frequent, we perform knockout experiments on held-out validation data, comparing shared circuit removal against two controls:

- **Capacity-Conserved with parity (C^3-parity).** Match the head/MLP counts of the shared circuit and sample from the entire pool (including shared circuit components). This control maintains capacity and component mix while allowing overlap with the shared circuit.
- **Capacity-Conserved without parity (C^3-exclude).** Match the head/MLP counts but explicitly exclude all shared circuit components from sampling. This maximizes contrast with the shared circuit while still controlling for capacity and component type.

For each knockout, we zero-out the outputs of the selected components and measure the resulting accuracy on the validation set. We measure relative impact using *lift*, defined as:

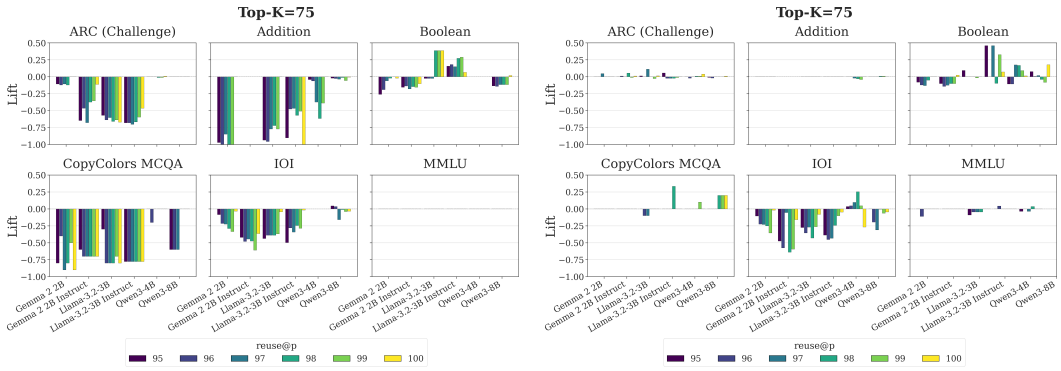
$$\text{lift} = \frac{\text{acc}_{\text{shared}} - \text{acc}_{\text{control}}}{\text{acc}_{\text{base}}}$$

where acc_{base} is baseline accuracy on the validation set, $\text{acc}_{\text{shared}}$ is validation accuracy after knocking out S_P, and $\text{acc}_{\text{control}}$ is validation accuracy after knocking out a control set. If shared circuit knockout causes larger accuracy drops than controls, this confirms the components' causal specificity for the task.

We hypothesize that, holding K and P fixed, higher within-task reuse will correlate with more negative lift under C^3-exclude; in other words, we think that highly reused components are more likely to be the task-specific circuit. This is a directional hypothesis and not assumed by our evaluation.

4 Results

Our experiments show that circuit reuse is a consistent phenomenon, though its magnitude varies with task type. As shown in Figure 1, algorithmic tasks and linguistic tasks (`Addition`, `IOI`, `CopyColors`



(a) C^3-Exclude Control (b) C^3-Parity Control

Figure 2: **Lift in task accuracy when knocking out shared circuits vs. two controls for top-K=75.** (a) The C^3-exclude control knocks out a disjoint set of components, yielding a strong negative lift. (b) The C^3-parity control knocks out a random set that can overlap, resulting in a near-zero lift.

MCQA) exhibit high reuse across all models, often exceeding 50% at P=95. Meanwhile, knowledge-intensive and complex reasoning tasks like MMLU and ARC show lower reuse. For MMLU, reuse is notably low for all models except the Gemma family, suggesting that different models may adopt distinct strategies for storing and accessing factual knowledge. We observe that reuse increases with the circuit size K, though not always monotonic (see Appendix C for full results). Additionally, unlike the other models we studied, OLMo-2-1B released its pretraining checkpoints. This allowed us to extend our analysis and observe that circuit reuse increases as the model undergoes more training steps, especially for larger circuit sizes K (see Figure 7 and Figure 6).

Our knockout studies yield two key findings. As seen in Figure 2, knocking out the shared circuit against the *C^3-exclude* control produces a strong negative lift across nearly all tasks and models. This indicates that the shared components are more critical for task performance than a disjoint set of the same size. However, with the *C^3-parity* control, shown in Figure 2. In this condition, the lift is consistently near-zero across all tasks and models. This implies that knocking out the shared circuit is no more damaging than knocking out a randomly selected set of components of the same size, provided that the random set is allowed to overlap with the shared circuit.

The C^3-exclude result confirms the shared circuit's causal role, while the C^3-parity result suggests its components are not unique in this role. We interpret this by hypothesizing that models develop functionally redundant pathways to solve tasks. If so, components in the shared circuit aren't unique; other components can perform similar functions. Then a random knockout is likely to damage these alternative pathways, leading to a performance drop comparable to knocking out the shared circuit itself. This suggests transformer circuits function less like a single, minimal algorithm and more like an ensemble of overlapping solutions.

5 Conclusion

In this paper, we presented a methodology (**reuse@P** and the **Capacity-Conserved Control** (C^3)) for quantifying and validating circuit reuse in transformer models. Our analysis covered seven models and six tasks, suggesting that circuit reuse is a task-dependent phenomenon, with algorithmic tasks showing higher consistency than knowledge-intensive ones. Our knockout studies, however, show mixed results. While we see clear evidence that shared circuits are important for task performance, we cannot rule out the possibility that models develop functionally redundant pathways that complicate the isolation of a single, task-specific circuit. To resolve this ambiguity, future work focuses on training small transformers on synthetic tasks with controlled settings to better understand and accurately characterize circuit reuse and redundancy in transformer models.

6 Limitations

Our approach has several limitations. First, Edge Attribution Patching (EAP) is a gradient-based approximation, less faithful than causal methods. More precise variants like EAP-IG (Hanna et al.,

2024) could improve circuit identification but are more expensive. Second, zero-ablation may yield out-of-distribution activations, potentially confounding results. Alternatives such as mean-ablation (Miller et al., 2024) may better preserve activation distributions. Finally, our study focuses on specific models and tasks, so results may not generalize across architectures or domains.

References

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

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, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *ICML 2024 Workshop on Mechanistic Interpretability*.

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

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*.

Joseph Miller, Bilal Chughtai, and William Saunders. 2024. Transformer circuit evaluation metrics are not robust. In *First Conference on Language Modeling*.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, and 4 others. 2025. MIB: A mechanistic interpretability benchmark. In *Forty-second International Conference on Machine Learning*.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, Miami, Florida, US. Association for Computational Linguistics.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

A Task Details

This section provides descriptions and examples for each task used in our circuit reuse analysis.

Addition. The Addition task requires models to perform 3-digit arithmetic addition presented in natural language format. For example, given the input "What is 247 + 163?", the model should output "410". We expect this task to be relatively simple for large language models to complete.

Boolean Logic. The Boolean task involves evaluating logical expressions with AND, OR, and NOT operations over boolean variables. An example input "True AND (False OR True)" should yield "True". This task requires systematic logical reasoning and an understanding of operator order.

Indirect Object Identification (IOI). Following Wang et al. (2023), the IOI task requires models to identify the indirect object in sentences with specific templates involving two names.

> *Input:* "As Carl and Maria left the consulate, Carl gave a fridge to"
> *Expected Output:* "Maria"

This task is taken from the Mechanistic Interpretability Benchmark (MIB) (Mueller et al., 2025).

CopyColors MCQA. The CopyColors task presents multiple-choice questions requiring models to identify and copy color information from a given context. It is similar to the IOI task.

> *Input:* "Coconuts are brown. What color are coconuts? (A) red (B) orange (C) brown (D) purple"
> *Expected Output:* "(C) brown"

This task is taken from the MIB (Mueller et al., 2025).

ARC. The AI2 Reasoning Challenge (Clark et al., 2018) contains science exam questions requiring reasoning over natural language. We use the Challenge set, which filters out questions answerable by simple retrieval.

> *Input:* "George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? (A) dry palms (B) wet palms (C) palms covered with oil (D) palms covered with lotion"
> *Expected Output:* "(A) dry palms"

This task is taken from the MIB (Mueller et al., 2025).

MMLU. The Massive Multitask Language Understanding benchmark (Hendrycks et al., 2021) spans 57 subjects from elementary to professional levels.

> *Input:* "This question refers to the following information. Read the the following quotation to answer questions. The various modes of worship which prevailed in the Roman world were all considered by the people as equally true; by the philosopher as equally false; and by the magistrate as equally useful. Edward Gibbon, The Decline and Fall of the Roman Empire, 1776–1788 Gibbon's interpretation of the state of religious worship in ancient Rome could be summarized as (A) In ancient Rome, religious worship was decentralized and tended to vary with one's social position. (B) In ancient Rome, religious worship was the source of much social tension and turmoil. (C) In ancient Rome, religious worship was homogeneous and highly centralized. (D) In ancient Rome, religious worship was revolutionized by the introduction of Christianity."
> *Expected Output:* "(A) In ancient Rome, religious worship was decentralized and tended to vary with one's social position."

B Methodology Details

For each task, we sample $n = 1000$ input/output pairs from the training set and extract a circuit for every example using edge attribution patching (EAP). We define the per-input circuit $\mathcal{C}_i$ as the top-K components by score for example i (we sweep $K \in \{25, 50, 75, 100\}$; main figures show $K = 75$).

The shared circuit at threshold P is formally defined as:

$$S_P = \left\{ c : c \in \mathcal{C}_i \text{ for at least } \left\lceil \frac{P}{100} \cdot n \right\rceil \text{ training examples} \right\}.$$

The reuse@P metric is:

$$\text{reuse@}P = \frac{\min\{|S_P|, K\}}{K} \times 100$$

C Additional Results

This section provides lift and reuse results across full sweeps for all models, as well as lift and reuse results for OLMo-2-1B pretraining checkpoints.

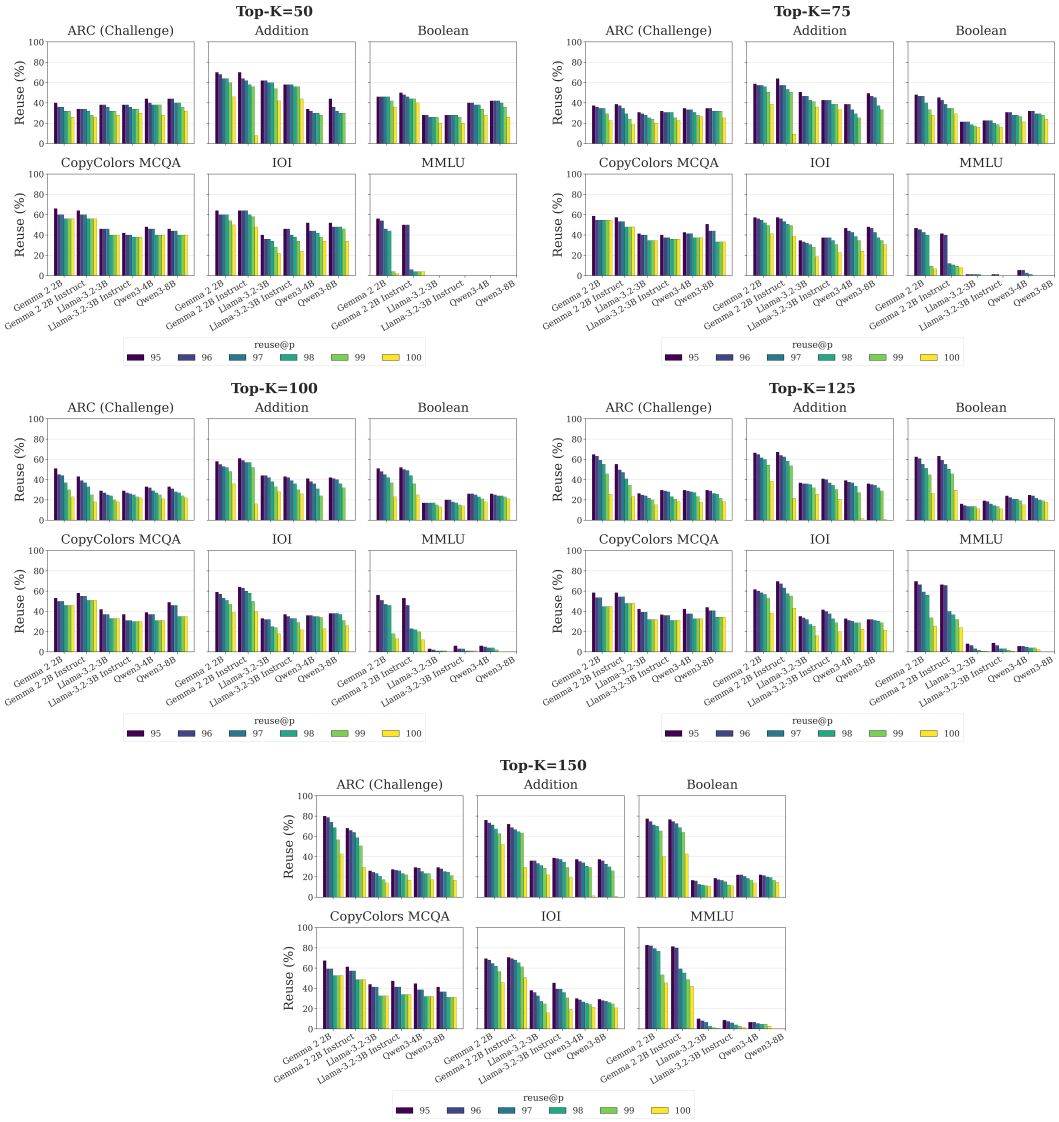


Figure 3: Reuse@P across all K.

Figure 4: Lift under C^3-parity for all K.

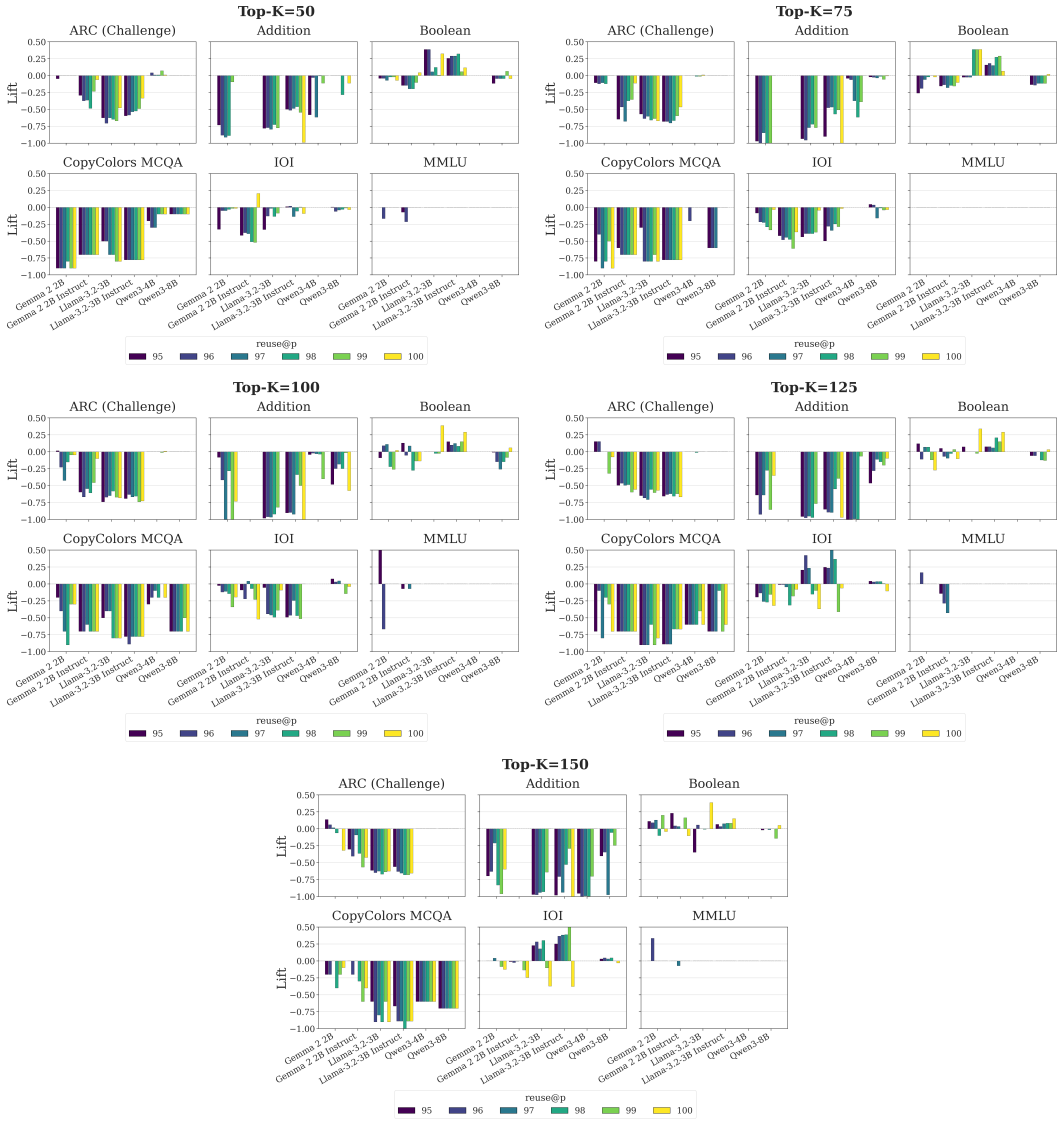


Figure 5: Lift under C^3-exclude for all K.

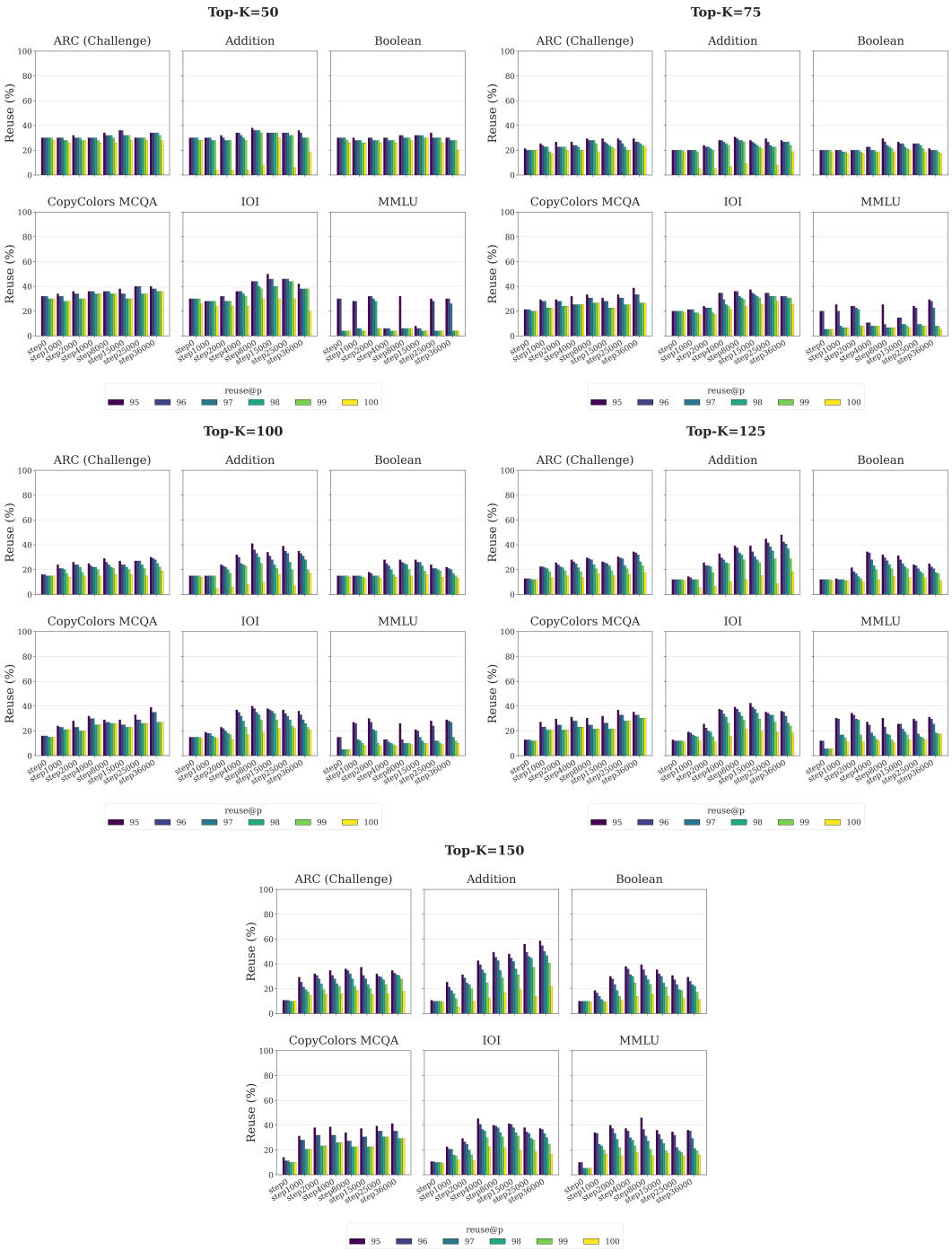


Figure 6: Reuse across OLMo-2 1B pretraining checkpoints.

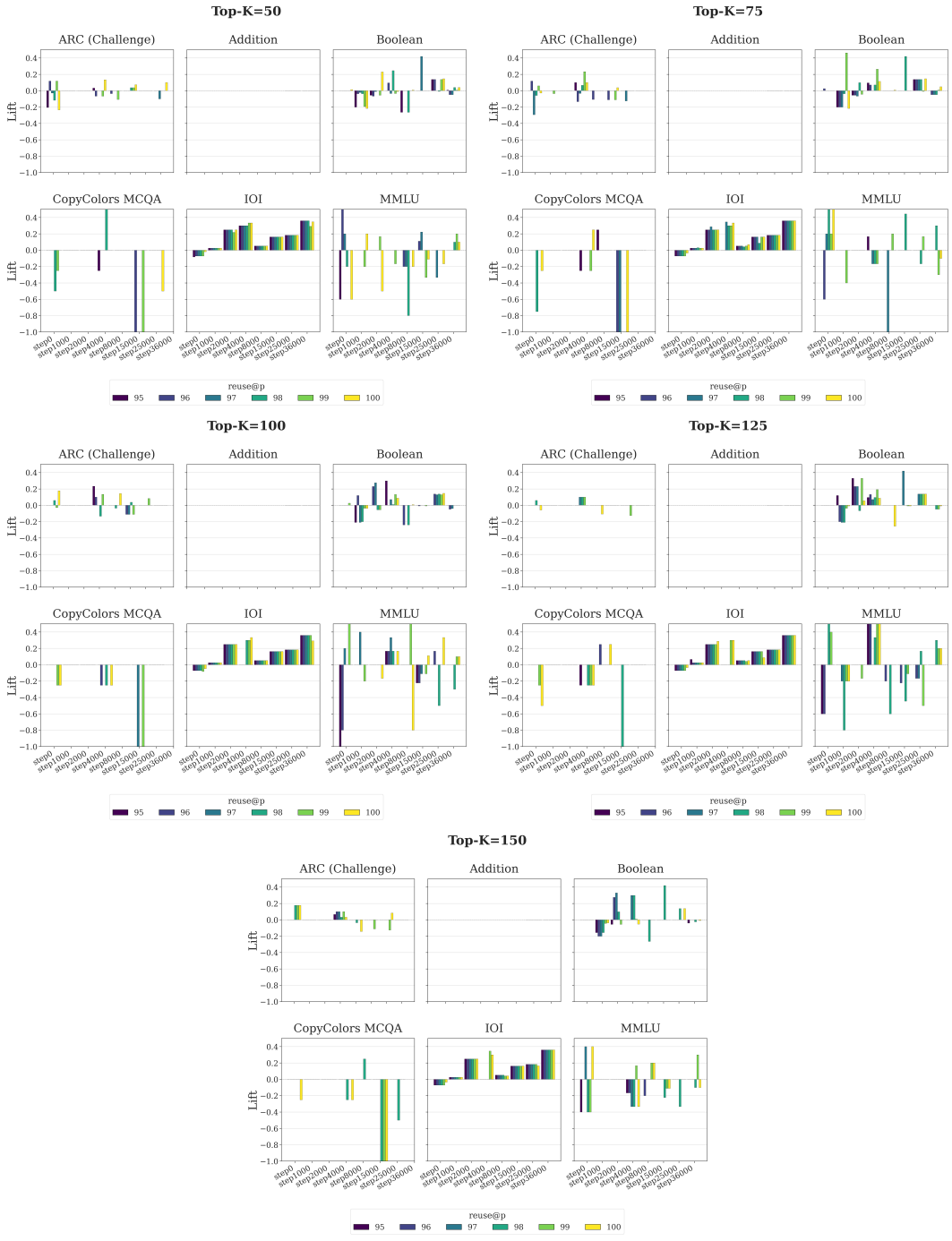


Figure 7: Lift under C^3-parity for OLMo-2 1B checkpoints.