BiCD: Bidirectional Circuit Discovery by Dynamic Forward Selection and Backward Pruning

Anonymous authors
Paper under double-blind review

Abstract

Mechanistic interpretability (MI) has developed many mature methods for discovering subcircuits in large language models. However, methods that abstract an LLM as a computational graph often face an enormous number of edges and consequently inefficient pruning. Methods based on activation patching, on the other hand, depend heavily on manually specified patching interventions, and the subcircuits they discover often have limited ability to perform the task independently. Drawing on the complementary ideas of forward search and backward pruning, we propose **Bidirectional Circuit Discovery (BiCD)**, a simple and efficient bidirectional search algorithm. BiCD seeks to obtain a subcircuit that is as small as possible while preserving as much of the model's functionality as possible.

1 Introduction

Mechanistic interpretability (MI) seeks to explain how models arrive at their outputs and complete tasks by identifying the internal computations that produce their behavior. A natural analogy comes from neuroscience: if functional regions in large language models (LLMs) could be localized in the way that functions are associated with regions of the brain, then assigning roles to those regions might explain how a model works. This localization intuition underlies methods such as **Knowledge Neurons**, causal tracing in **ROME**, and **Inference-Time Intervention**, which associate model behaviors with individual FFN neurons, particular MLP modules and layers, or selected attention heads (Dai et al., 2022; Meng et al., 2022; Li et al., 2023).

Research on superposition, however, challenges a one-to-one correspondence between model functions and predefined components (Elhage et al., 2022; Bricken et al., 2023). Models can represent features as distributed directions across neurons, and individual neurons can participate in multiple features. An interpretable function may therefore emerge from the coordinated computation of multiple components, while only part of a component's internal activation contributes to that function. Explaining a task by assigning a complete function to a single component can consequently miss the distributed computation that implements it.

Circuit-level analysis provides a way to avoid this localization problem. Manual circuit studies such as the reverse-engineered IOI circuit, together with automated circuit-discovery methods such as ACDC, EAP, EAP-IG, and Edge Pruning, seek task-relevant sets of interacting computations rather than requiring a model function to reside in one component (Wang et al., 2023; Conmy et al., 2023; Syed et al., 2024; Hanna et al., 2024; Bhaskar et al., 2024). A circuit can therefore represent a function through the joint behavior of its members. Following this view, our work focuses on discovering a submodel circuit that is interpretable and as small as possible, while remaining able to perform the target task.

We propose **Bidirectional Circuit Discovery (BiCD)**, a bidirectional search algorithm consisting of the forward search algorithm **DFCD** and a backward pruning procedure. BiCD explicitly searches for a zero-ablated subcircuit: computations outside the selected set are set to zero, and the selected subcircuit must retain the original task behavior on its own. DFCD constructs this circuit from an ablated state, progressively recovering as much of the model's original task functionality as possible. Backward pruning then removes computations that are unnecessary once that functionality has been

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.

established, making the selected circuit as compact as possible. Together, the two directions search for a small circuit without giving up the behavior that the circuit is intended to explain.

On the target task, BiCD discovers a compact component-level circuit with task performance comparable to larger edge-based circuits from EAP and EAP-IG; we report both native component counts and a structural edge projection.

2 BiCD

BiCD aims to find the most compact set of components that can perform the target task. The components in this set S form a subcircuit, and all components outside S are zero ablated. Let U denote the full component set. We use mean correct-answer cross entropy $\mathcal{L}$ to compare the task performance of the subcircuit and the full model, and define the following ratio:

$$\rho_{\mathrm{CE}}(S) = \frac{\mathcal{L}(U)}{\mathcal{L}(S)}. \tag{1}$$

A hyperparameter $\tau \in (0, 1]$ controls the allowed degradation in correct-answer CE relative to the full model. A larger τ imposes a stricter CE constraint, although the size returned by the greedy search need not increase monotonically. Feasibility requires

$$\rho_{\mathrm{CE}}(S) \geq \tau \qquad \Longleftrightarrow \qquad \mathcal{L}(S) \leq B_\tau = \frac{\mathcal{L}(U)}{\tau}. \tag{2}$$

The full model has ratio one. In our experiments, removing some components can produce a subcircuit with lower CE than the full model, and hence $\rho_{\mathrm{CE}}(S) > 1$. This indicates that some components can act against performance on the target task.

BiCD consists of two stages. The first stage adds components to an initially empty subcircuit, and we call this stage Dynamic Forward Circuit Discovery (DFCD). In our experiments, all 12 MLP layers are added to the subcircuit regardless of the parameter setting. This resembles the IOI circuit evaluation, which retains the MLPs while ablating attention heads outside the proposed circuit (Wang et al., 2023). There may be a deeper mechanism behind this observation that remains to be explored. We therefore use the 12 MLPs as the initial subcircuit and progressively add the remaining components to construct a more capable subcircuit.

Figure 1 illustrates how consecutive rounds are chained in both stages.

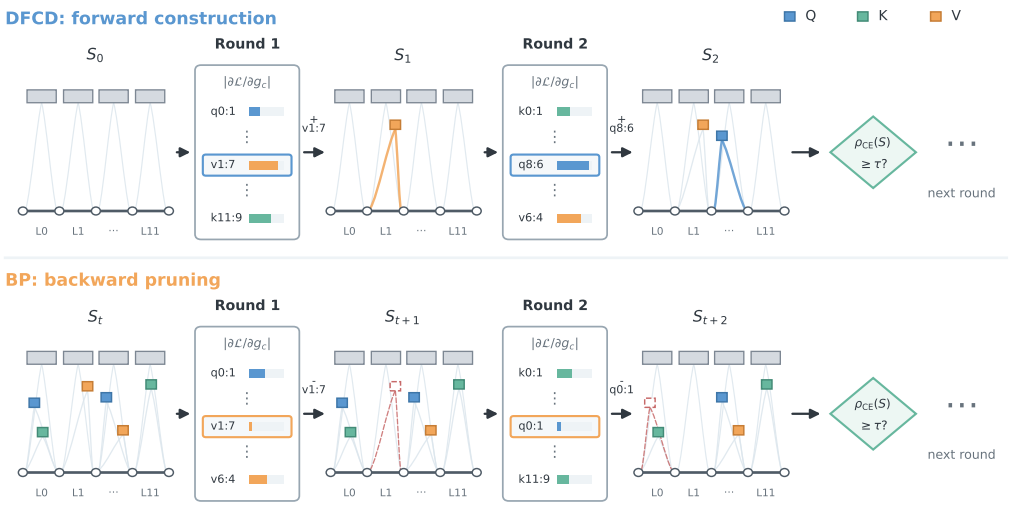


Figure 1: BiCD combines dynamic forward circuit discovery (DFCD) with backward pruning (BP).

Inspired by EAP (Syed et al., 2024), we measure the importance of a component by the absolute value of the gradient contribution of its gate to the final CE:

$$a_c(S) = \left| \frac{\partial \mathcal{L}(\mathbf{g})}{\partial g_c} \right|_{\mathbf{g}=\mathbf{g}(S)}, \tag{3}$$

where $g_c = 1$ retains component c and $g_c = 0$ zeros its output. We do not consider a component important merely because its signed gradient is large and points toward lower CE, and our experiments support this view. DFCD therefore selects the component with the largest absolute gradient, rather than the one with the most negative signed gradient. After adding a component, we fix the current subcircuit and repeat the same scoring procedure for all remaining components. This allows components that were underestimated in the first scoring because their dependencies were absent to be rescored after those dependencies have been added. The greedy rescoring procedure preserves the speed of gradient-based selection while partially mitigating the misestimation of component importance caused by circuit topology.

Once the subcircuit reaches the prescribed threshold, BiCD enters its second stage, backward pruning. The initial MLP set S_0 remains fixed. At iteration t, let $\mathcal{D}_t$ be the set of removable components whose exact deletion preserves the CE threshold. Backward pruning removes the least important component in this set:

$$\begin{aligned} \mathcal{D}_t &= \{c \in S_t \setminus S_0 \mid \mathcal{L}(S_t \setminus \{c\}) \le B_\tau\}, \\ c_t^- &= \arg\min_{c \in \mathcal{D}_t} a_c(S_t), \\ S_{t+1} &= S_t \setminus \{c_t^-\}. \end{aligned} \tag{4}$$

The scores are recomputed after each deletion, and pruning stops when $\mathcal{D}_t = \varnothing$. Starting from the DFCD candidate set changes the context in which deletion scores and feasibility are evaluated. In our comparison, this forward-then-backward procedure yields a smaller final circuit with higher held-out Top-1 accuracy than pruning the full model directly.

3 Experiment

3.1 Experiment Setup

We evaluate CE-based BiCD with five fine-tuning seeds (42–46); all other metric settings and baseline methods use seed 42.

Following the mechanistic-interpretability workflow summarized by ACDC (Conmy et al., 2023), we first select a behavior, construct a dataset that elicits it, and define a task metric. We study the addition circuit of GPT-2 small (Radford et al., 2019). The pretrained model does not reliably perform this behavior: on the four prompt templates used below, it answers only 1 of 1,444 test prompts correctly. We therefore fine-tune GPT-2 small before circuit discovery so that the subject model reliably exhibits the target behavior.

We fine-tune on addition examples with operands from 1 to 49. Each operand pair is instantiated with four templates: `a + b =`, `a + b equals`, `sum a + b =`, and `math:  a + b =`, with supervision applied only to the answer token. The fine-tuning data also include auxiliary capital-recall prompts. Table 1 reports the arithmetic split sizes and resulting accuracy.

Table 1: Arithmetic fine-tuning data and Top-1 next-token accuracy of the fine-tuned full model. Prompt counts include four templates per operand pair; the 94 auxiliary capital-recall training prompts are not included in the arithmetic counts.

Split	Operand pairs	Arithmetic prompts	Top-1
Train	1,680	6,720	100.00%
Validation	360	1,440	99.72%
Test	361	1,444	99.93%

We select 47 prompts from the fine-tuning test split as the circuit-discovery set. All 47 prompts use the `a + b =` template, which is one of the four prompt templates used during fine-tuning.

We use mean correct-answer cross entropy (CE) as the task metric. For discovery prompt i with accepted correct-answer token set Y_i, its CE is

$$\ell_i(S) = -\log \sum_{y \in Y_i} p(y \mid x_i; S), \tag{5}$$

and the discovery-set CE is the arithmetic mean over all $N = 47$ prompts:

$$\mathcal{L}(S) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(S). \tag{6}$$

Each $\ell_i(S)$ is already nonnegative, so taking its absolute value would not change the mean. The absolute value is instead applied after differentiating the mean CE. For component c, a binary gate g_c switches its output off or on. The gradient of mean CE with respect to this gate measures the local effect of changing the component state, and its magnitude is our component-level discovery metric:

$$a_c(S) = \left| \frac{\partial \mathcal{L}(\mathbf{g})}{\partial g_c} \right|_{\mathbf{g}=\mathbf{g}(S)}. \tag{7}$$

Table 2 presents the task and metric in the format used by ACDC for its behavior suite.

Table 2: Addition behavior and discovery metric.

Task	Example prompt	Correct output	Metric
Addition	`17 + 23 =`	`40`	Mean CE; $\lvert\partial\mathcal{L}/\partial g_c\rvert$

3.2 Model Abstraction

We divide the Transformer into MLP and attention components. Each of the 12 MLP blocks is one component, while the query, key, and value projections of every attention head are treated as three separate components. GPT-2 small therefore contains

$$12 \text{ layers} \times 12 \text{ heads} \times 3 \text{ Q/K/V projections} + 12 \text{ MLPs} = 444 \tag{8}$$

components at this granularity.

We do not instantiate the model as an explicit writer-to-reader graph of the kind searched by ACDC and EAP, or used to describe the manually identified IOI circuit (Conmy et al., 2023; Syed et al., 2024; Wang et al., 2023). Such a representation introduces substantially more edges than components. Instead, BiCD operates directly on the original Transformer with one binary gate per component. Setting a gate to zero suppresses that component's output, while setting it to one restores the output. Thus a candidate subcircuit remains a gated Transformer and does not require explicit enumeration of all pairwise computational edges. The pretrained and fine-tuned model weights themselves are not modified during circuit discovery.

3.3 Circuit Discovery

Subcircuit initialization. Across our preliminary ablations and parameter settings, removing any MLP block causes a large loss of arithmetic performance, and all 12 MLPs are retained. We therefore initialize the subcircuit with the 12 MLPs active and set every Q/K/V gate to zero. This MLP-only state has discovery-set CE 18.559813 and answers 0 of 47 discovery examples correctly.

DFCD. At the current subcircuit state, every inactive component has gate value zero. A forward–backward pass computes the signed derivative

$$d_c(S) = \left. \frac{\partial \mathcal{L}}{\partial g_c} \right|_{\mathbf{g}=\mathbf{g}(S)} \tag{9}$$

for every inactive component. Inspired by EAP's use of gradients to obtain a first-order approximation of intervention effects (Syed et al., 2024), we use this derivative to approximate the finite CE change produced by opening component c while holding all other gates fixed:

$$\mathcal{L}(S \cup \{c\}) - \mathcal{L}(S) \approx d_c(S)(1 - 0) = d_c(S). \tag{10}$$

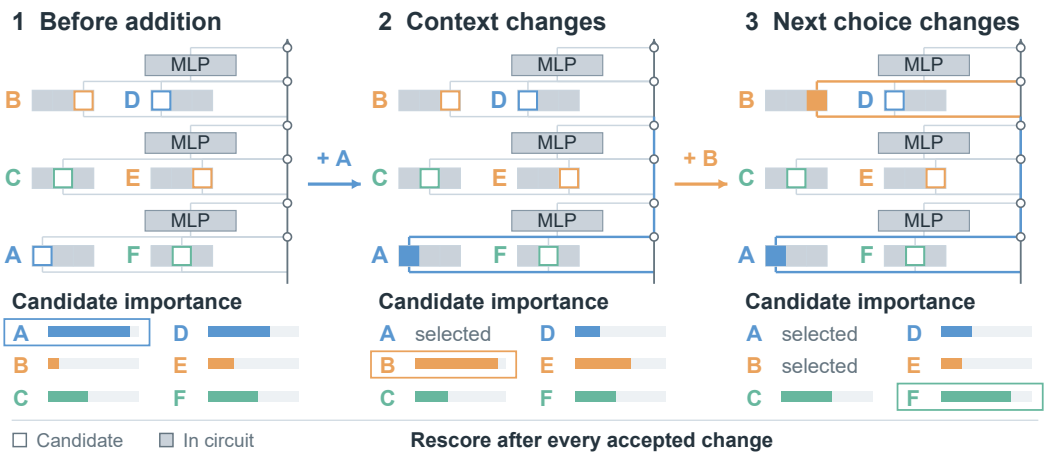


Figure 2: Dynamic scoring.

This is a local first-order approximation rather than the exact CE difference for a complete gate change. A negative derivative predicts movement toward lower CE, while a positive derivative predicts movement toward higher CE. A purely directional rule would consequently select the most negative derivative.

Our metric comparison shows, however, that selecting only the components predicted to decrease CE fastest performs worse than ranking components by gradient magnitude. We interpret this result as evidence that circuit discovery should retain components with a large task effect, rather than equating importance only with an immediate reduction in discovery-set CE. A component with a large positive or negative effect can be important to the computation, whereas a component with little effect is a stronger removal candidate. DFCD therefore adds

$$c_t^+ = \arg\max_{c \in U \setminus S_t} |d_c(S_t)|, \qquad S_{t+1} = S_t \cup \{c_t^+\}. \tag{11}$$

Adding a component changes the subcircuit state and can change the gradients of every remaining component. DFCD therefore recomputes all gate gradients after each addition. For example, V:0:3 initially has an absolute CE-gradient score of 0.010667, ranking 143rd among the 144 V components. After 16 other components have been added, its score rises to 0.391212, 36.7 times its initial value, and DFCD selects it at step 17. This example illustrates how a component initially assigned a low score can become important in a different subcircuit context. In this experiment, we set $\tau = 0.999$ and stop adding components once

$$\rho_{\mathrm{CE}}(S) = \frac{\mathcal{L}(U)}{\mathcal{L}(S)} \geq 0.999, \tag{12}$$

that is, once the full model's CE is at least 99.9% of the subcircuit's CE.

Figure 2 illustrates how changing the selected subset can change the importance and ranking of components that remain unselected. Appendix B.6 reports the Q/K/V component counts across the complete DFCD and BP trajectory.

Backward pruning. Backward pruning has been used to remove unimportant edges from explicit circuit graphs, most notably in ACDC (Conmy et al., 2023). Since BiCD does not construct an edge graph, it instead tests component deletion directly by changing a selected component gate from one to zero. At each iteration, we compute $|d_c(S_t)|$ for the selected Q/K/V components and order them from smallest to largest. Beginning with the least sensitive component, we evaluate the exact zero-ablation CE after deletion. We accept the first candidate satisfying

$$\mathcal{L}(S_t \setminus \{c\}) \leq \frac{\mathcal{L}(U)}{\tau}. \tag{13}$$

Only one component is removed in each iteration. We then recompute and reorder all remaining gradients because a component that is unimportant in the presence of another component may be-

come important after that component is removed, and the reverse can also occur. Pruning stops after a complete scan finds no deletion that preserves the CE threshold.

3.4 EVALUATION

Metric and parameter selection. We compare directional signed-gradient ranking with absolute-gradient ranking. Table 3 summarizes the two rules. The absolute-gradient variant provides the stronger size–functionality trade-off and is used for the remaining experiments.

Table 3: Component-ranking rules compared at $\tau = 0.999$. Component counts include the 12 fixed MLPs, and top-1 accuracy is measured under zero ablation on the 314 held-out examples.

Ranking metric	Forward rule	Backward rule	Components	Top-1	Variant
Signed CE gradient	Smallest d_c	Largest d_c first	160	82.80%	Control
Absolute CE gradient	Largest $\lvert d_c \rvert$	Smallest $\lvert d_c \rvert$ first	86	95.86%	Ours

We also sweep the retention threshold and select $\tau = 0.999$ by jointly considering subcircuit size and zero-ablation task performance. Table 4 records the selected operating point; the full sweep appears in Appendix F.3. Full-model discovery CE is 0.003958459, so the corresponding CE budget is 0.003962421.

Table 4: Selected operating point from the retention-threshold sweep.

τ	CE budget	Components	Discovery top-1	Evaluation top-1
0.999	0.003962421	86	47/47	301/314

Subcircuit evaluation. Circuit discovery uses 47 examples drawn from the 361-pair fine-tuning test split. The remaining 314 pairs have no operand-pair overlap with discovery and are used for metric and threshold comparison and final circuit evaluation. These 314 examples never contribute gate gradients or individual addition and deletion decisions. All circuit evaluations use the `a + b =` template and zero every unselected Q/K/V output. Our primary evaluation metric is whether the unique correct arithmetic answer has the highest logit (top-1 accuracy). Because each addition problem has only one correct answer, placing that answer at the second-highest logit still constitutes an incorrect prediction; we nevertheless also report whether the correct answer appears within the top 3, top 5, and top 10 logits.

The final circuit contains 86 components: all 12 MLPs and 74 attention projections, comprising 16 Q, 16 K, and 42 V components. Figure 3 shows their layer-wise distribution.

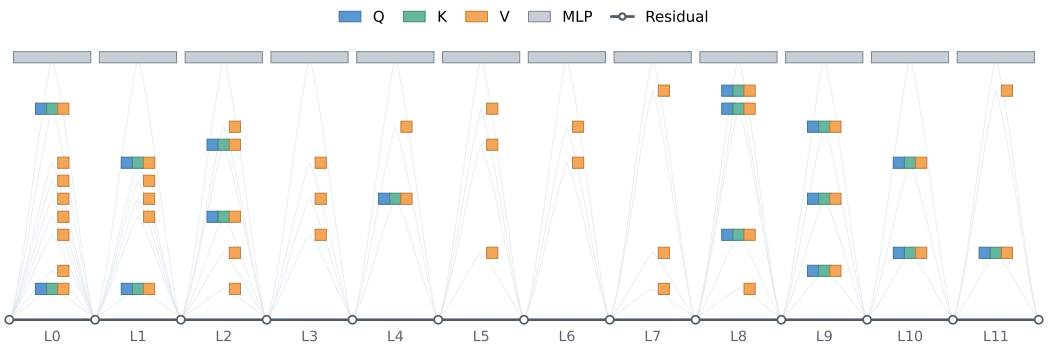


Figure 3: The 86-component arithmetic circuit discovered by BiCD. The circuit contains 16 Q, 16 K, 42 V, and all 12 MLP components; unselected components are omitted.

4 Comparison with Circuit Discovery Baselines

4.1 Compared Baselines

We compare BiCD with four automated circuit-discovery baselines implemented from their official repositories: ACDC (Conmy et al., 2023), EAP (Syed et al., 2024), EAP-IG (Hanna et al., 2024), and Edge Pruning (Bhaskar et al., 2024). Together, these baselines cover discrete edge pruning, static first-order edge attribution, path-integrated edge attribution, and learned differentiable pruning.

1. **ACDC.** ACDC traverses an explicit computational graph in reverse topological order and removes an edge when its effect on the task metric remains below a threshold. Our run uses zero ablation and the native KL objective.
2. **EAP.** Edge Attribution Patching assigns each graph edge a first-order score combining its source activation change and destination gradient. Our reproduction uses a zero baseline and correct-answer CE. It scores the full graph in parallel, ranks the edges once, and retains the highest-scoring edges under a chosen budget.
3. **EAP-IG.** EAP-IG replaces EAP's local gradient with integrated gradients along an interpolation path, then uses the resulting static global ranking to select edges. We use the authors' EAP-IG-inputs variant with five integration steps and a corrupt-input baseline.
4. **Edge Pruning.** Edge Pruning jointly optimizes differentiable edge gates with fidelity and sparsity objectives. It reports nonzero soft gates, which are not directly equivalent to BiCD's discrete Q/K/V and MLP components.

4.2 Dynamic Rescoring Instead of Static Attribution

Unlike EAP-style methods, BiCD does not rely on a single static scoring pass. It recomputes component scores whenever the selected subset changes, allowing the score of a component to reflect the components on which it currently depends. This dynamic scoring can expose interactions that are invisible in the initial state. Figure 6 in Appendix E compares the initial V-component scores with those from the DFCD round exhibiting the largest overall score change. For each round t, we compute the standard deviation of $a_c(S_t) - a_c(S_0)$ over the components not yet selected, $c \in U \setminus S_t$. Round 2 maximizes this standard deviation at 0.290196.

4.3 Forward Construction Before Pruning

Unlike Edge-Pruning-style approaches that optimize sparsity over the full candidate graph, BiCD first uses DFCD to construct a functional circuit from the MLP-only initialization. In our experiment, DFCD stops after adding 426 of the 432 Q/K/V components, excluding only six components before backward pruning begins. This small change nevertheless alters the subsequent pruning trajectory substantially. Table 5 compares the resulting bidirectional search with pure backward pruning under the same discovery threshold.

Table 5: Forward-then-backward search versus pure backward pruning on the same model. Discovery metrics use 47 prompts; test accuracy uses the 314-example zero-ablation evaluation.

Search procedure	Components	Discovery CE	Discovery top-1	Test top-1
DFCD → BP (BiCD)	**86**	0.00386216	47/47	301/314 (95.86%)
Pure BP	92	0.00390281	47/47	297/314 (94.59%)

Table 6: Component choices around the start of backward pruning. The middle and right columns give the first five accepted deletions. Five of the six DFCD omissions are shown; the remaining component is K:11:10, removed at pure-BP iteration 284.

Rank	DFCD omission (pure-BP rank)	Pure BP	DFCD → BP
1	Q:11:9 (36)	V:5:3	K:5:3
2	K:11:9 (37)	Q:5:3	Q:5:3
3	K:11:11 (72)	K:5:3	Q:1:10
4	Q:11:11 (73)	Q:7:1	K:1:10
5	Q:11:10 (283)	K:7:1	V:10:11

Both procedures accept 352 backward deletions, but they start from different sets: 438 components after DFCD versus all 444 for pure BP. BiCD therefore finishes with six fewer components, 86 rather than 92, and answers four additional held-out examples correctly. The different deletion prefixes show that the six DFCD omissions alter the pruning trajectory, not merely the initial set size. This context dependence is consistent with coordinated computation across components (Elhage et al., 2022; Bricken et al., 2023), although this comparison alone does not establish a particular superposition mechanism.

4.4 Quantitative Comparison

Table 7 reports the baseline comparison. BiCD reaches 95.86% Top-1 with 86 components. The first sampled points reaching this accuracy are 6,396 connected edges for zero-baseline EAP and 3,196 for EAP-IG, obtaining 96.82% and 98.09%, respectively. ACDC, Edge Pruning, and edge-level BiCD reach 79.30%, 87.26%, and 76.11%. Native size is reported explicitly because components, graph edges, and soft gates are not equivalent.

Table 7: Results on the 314-example zero-ablation evaluation. "First $\geq$ BiCD" denotes the first sampled point reaching BiCD's 95.86% Top-1. Graph sizes count connected edges; the BiCD projection is a structural count, not a separate edge-ablation evaluation.

Method	Native size	CE	Top-1	Top-3
BiCD	86 components; 2,440 induced edges	0.1279	95.86%	100%
Edge-level BiCD	280 edges (309 selected)	0.9164	76.11%	92.99%
ACDC	474 edges (486 selected)	0.7835	79.30%	93.31%
EAP, first $\geq$ BiCD	6,396 edges	0.0902	96.82%	100%
EAP, 24k	24,000 edges	0.0041	100%	100%
EAP-IG, first $\geq$ BiCD	3,196 edges	0.0638	98.09%	100%
EAP-IG, 8k	8,000 edges	0.0101	100%	100%
Edge Pruning	1,531 nonzero soft gates	0.3555	87.26%	99.04%

For structural comparison, we project the BiCD component set onto the authors' writer-to-reader graph: writers are the input, retained MLPs, and V-enabled heads; readers are retained Q/K/V projections of those heads, retained MLPs, and logits. This gives 2,440 connected edges. Component gates act after projection, whereas graph interventions act on residual contributions, so this count does not assert identical ablation semantics. EAP and EAP-IG sizes are measured after the authors' connectivity pruning.

Table 8: Measured circuit-discovery wall-clock time on an NVIDIA GeForce RTX 2060 for the same 47-prompt discovery set. Model loading and tokenization are excluded. Timings retain each implementation's native search semantics.

Method	Seconds	Wall time
EAP (attribution only)	1.28	0:00:01
EAP-IG (attribution only)	4.62	0:00:05
BiCD	354.21	0:05:54
Static-ranking BiCD	430.63	0:07:11
Edge Pruning	19,469.95	5:24:30
ACDC	7,067.74	1:57:48
Edge-level BiCD	99,542.21	27:39:02

BiCD completes the search in 354.21 seconds. EAP and EAP-IG attribution is substantially faster, but those timers exclude subsequent edge-budget evaluations. BiCD and its static control include iterative exact CE tests (Appendix E). Edge Pruning reports optimization-loop time, including validation and checkpointing; the edge-level BiCD timer excludes anchor evaluations.

5 Related Work

Transformer circuits. Circuit analysis seeks sparse computational structures that explain model behavior (Elhage et al., 2021). Manual studies have identified mechanisms such as induction heads (Olsson et al., 2022) and the indirect object identification circuit (Wang et al., 2023). BiCD

follows this circuit perspective but adopts a sufficiency-oriented criterion: the selected subnetwork must perform the task while all unselected components are zeroed.

Automated circuit discovery. ACDC prunes graph edges with small metric effects (Conmy et al., 2023). EAP approximates edge effects with first-order attribution (Syed et al., 2024), while EAP-IG integrates gradients along an interpolation path (Hanna et al., 2024). Edge Pruning learns differentiable edge gates under fidelity and sparsity objectives (Bhaskar et al., 2024). At component granularity, Causal Head Gating learns soft gates over whole attention heads from next-token loss (Nam et al., 2025). Contextual decomposition and position-aware attribution provide further edge-level extensions (Hsu et al., 2025; Haklay et al., 2025). BiCD instead searches Q/K/V and MLP gates, constructs a circuit forward before pruning it backward, and recomputes attribution after every accepted change.

Interventions and granularity. Activation patching and causal tracing compare clean and corrupted executions to localize information flow (Meng et al., 2022; Zhang & Nanda, 2024). BiCD instead applies correct-answer CE to zero-ablated circuits during both search and standalone evaluation. Feature-level methods construct causal graphs over sparse-autoencoder features (Marks et al., 2025) or cross-layer transcoders (Ameisen et al., 2025). BiCD directly searches the chosen native Q/K/V and MLP component inventory.

6 Conclusion

BiCD combines dynamic rescoring, component-level zero ablation, and forward-informed backward pruning. Rescoring updates component importance as the selected circuit changes, while component-level search avoids the much larger edge inventory of an explicit graph. In our dynamic-versus-static control, updated rankings reduce the number of exact CE trials and the total search time. The functional circuit constructed by DFCD provides a different starting set for backward pruning, yielding a smaller final circuit with higher held-out Top-1 accuracy than pure backward pruning in our comparison. Together, these choices enable efficient discovery of compact and functional circuits.

Our current evaluation is limited to fine-tuned GPT-2 small and medium models, primarily on single-token addition. Future work will test BiCD on different model capabilities, investigate whether the resulting circuits share common topological structures, and extend the cross-task transfer and circuit-overlap analyses.

AI use statement

In this work, we used OpenAI Codex for feedback on research methodology and experimental design, method implementation and debugging, and assistance with interpreting experimental results. In particular, Codex helped translate author-provided algorithmic descriptions into executable code, inspect experimental scripts, and discuss alternative formulations and control experiments. The authors originated the research question and the core BiCD design, including dynamic forward construction, backward pruning, component-level zero ablation, and the task cross-entropy constraint. The authors made the final decisions regarding the methodology, experimental settings, baselines, and interpretation of the results.

Additionally, Codex was used for software-engineering and presentation tasks, including launching and monitoring experiments, aggregating saved outputs, assisting with tables and figures, locating relevant papers and official repositories, organizing the manuscript, editing and translating English text from author-provided notes, formatting LaTeX, and diagnosing compilation and layout issues.

The arithmetic data were produced by deterministic programs, and every numerical result reported in the paper was obtained from an executed model evaluation; none was generated or estimated by an LLM. The authors reviewed all AI-assisted prose and the experimental data compiled with AI assistance from model evaluations, checked references against source papers or official repositories, and verified the reported results against saved experimental artifacts. We take responsibility for the final content of this work, including its claims, citations, code, experiments, and manuscript. No generative AI system is listed as an author.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 18506–18534, 2024. doi: 10.52202/079017-0587. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/20fdaf67581e6d7157376d1ed584040a-Abstract-Conference.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0719. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

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

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. URL https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/.

Tal Haklay, Hadas Orgad, David Bau, Aaron Mueller, and Yonatan Belinkov. Position-aware automatic circuit discovery. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2792–2817. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.141. URL https://aclanthology.org/2025.acl-long.141/.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 76033–76060, 2023. doi: 10.52202/075280-3322. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/efbba7719cc5172d175240f24be11280-Abstract-Conference.html.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. URL https://openreview.net/forum?id=grXgesr5dT.

Aliyah Hsu, Georgia Zhou, Yeshwanth Cherapanamjeri, Yaxuan Huang, Anobel Odisho, Peter Carroll, and Bin Yu. Efficient automated circuit discovery in transformers using contextual decomposition. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/916ee60e315531d6b3954af8a8dc3437-Abstract-Conference.html.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/81b8390039b7302c909cb769f8b6cd93-Abstract-Conference.html.

David Lindner, Janos Kramar, Sebastian Farquhar, Matthew Rahtz, Tom McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1649. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/771155abaae744e08576f1f3b4b7ac0d-Abstract-Conference.html.

Samuel Marks, Can Rager, Eric Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, pp. 23888–23923, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3ba4d47a83e498c2b1a0868cba20f6de-Abstract-Conference.html.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Andrew Nam, Henry Conklin, Yukang Yang, Tom Griffiths, Jonathan D. Cohen, and Sarah-Jane Leslie. Causal head gating: A framework for interpreting roles of attention heads in transformers. In *Advances in Neural Information Processing Systems*, volume 38, pp. 115609–115631, 2025. doi: 10.52202/085713-3856. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a7f530e11fa19e9551b7a51dbd0f336f-Abstract-Conference.html.

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. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.blackboxnlp-1.25. URL https://aclanthology.org/2024.blackboxnlp-1.25/.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Hf17y6u9BC.

A Mechanistic Interpretability Workflow

ACDC systematizes circuit analysis as four stages: select a behavior, dataset, and metric; divide the network into computational units; isolate a task-relevant subcircuit; and analyze the resulting computation (Conmy et al., 2023). The main text presents BiCD, the new search algorithm used for circuit isolation. For completeness, this appendix records the task definition, model decomposition, and analysis protocol that surround the algorithm in our arithmetic study.

A.1 Step 1: Arithmetic Behavior, Data, and Metric

We evaluate BiCD on five independently fine-tuned checkpoints with training seeds 42–46. All other circuit-discovery comparisons use the checkpoint fine-tuned with seed 42.

We focus on single-token integer addition. Because pretrained GPT-2 small does not reliably perform this task, we fine-tune it to predict the answer token. The arithmetic dataset contains every ordered pair $(a, b) \in \{1, \ldots, 49\}^2$, giving 2,401 equations with sums from 2 to 98. A representative example is `17 + 23 = 40`. With seed 42, we divide the operand pairs into 1,680 training, 360 validation, and 361 test examples. Each training pair is instantiated with four prompt templates, yielding 6,720 arithmetic training rows; checkpoint construction additionally includes 94 auxiliary capital-recall rows. The validation split controls early stopping, and the test split is not used to update model weights.

Circuit discovery uses the `a + b =` template on 47 examples drawn from the fine-tuning test split. The other 314 examples form a disjoint circuit-evaluation set and never contribute gradients, candidate rankings, the CE budget, or stopping decisions. We use the mean correct-answer cross entropy defined in the main text. The search objective is to make the zero-ablated subcircuit recover the full fine-tuned model's task performance under this metric.

A.2 Step 2: Computational Units

We split each attention head into separately gated query, key, and value projections, denoted $q_{l,h}$, $k_{l,h}$, and $v_{l,h}$. Each Transformer MLP block m_l is one additional component. GPT-2 small therefore contains

$$12 \text{ layers} \times 12 \text{ heads} \times 3 \text{ Q/K/V projections} + 12 \text{ MLPs} = 444 \tag{14}$$

components at this granularity. In this experiment, all 12 MLPs form the mandatory base and remain active throughout search. BiCD searches over the other 432 Q/K/V components, while the fixed MLPs remain included in the reported circuit size.

This representation avoids explicitly searching the much larger set of writer-to-reader edges. A selected Q/K/V component switches one head projection across token positions; it is therefore not identical to an individual edge in EAP or ACDC.

A.3 Step 4: Circuit Analysis

We first measure standalone functionality by zeroing every unselected Q/K/V output on the disjoint 314-example set. We report correct-answer CE, exact top-k accuracy, and top-1 retention among examples answered correctly by the full model. We then analyze the selected circuit's layer-wise Q/K/V composition and list the complete component inventory.

For comparison with edge-based methods, we also project the selected component bundles onto the residual-stream graph. We count an effective edge only when its reader is selected and its source writer remains active under zero ablation. This accounting supports a descriptive sparsity comparison, but it does not make component-level and edge-level search spaces equivalent.

B Full BiCD Algorithm

B.1 Objective and Feasibility

For a binary circuit $S \subseteq U$, each component gate is $g_c(S) = \mathbf{1}[c \in S]$. A zero gate replaces the corresponding component output with zero. For example i, let x_i be the prompt and Y_i the accepted

correct-answer token set. BiCD uses

$$\ell_i(S) = -\log \sum_{y \in Y_i} p(y \mid x_i; S), \qquad \mathcal{L}(S) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(S). \tag{15}$$

In our addition task every answer is a single token, while the set notation also supports equivalent answer tokenizations.

CE recovery with respect to the full circuit U is

$$\rho_{\mathrm{CE}}(S) = \frac{\mathcal{L}(U)}{\mathcal{L}(S)}. \tag{16}$$

Requiring $\rho_{\mathrm{CE}}(S) \geq \tau$ is equivalent to $\mathcal{L}(S) \leq B_\tau = \mathcal{L}(U)/\tau$. The ratio can exceed one when a sparse circuit has lower CE than the full model; the CE budget remains the operative acceptance test.

B.2 Component Gates

For layer l, reshape the joint attention projection output as $A_l \in \mathbb{R}^{B \times T \times 3 \times H \times d}$. The three projection indices correspond to Q, K, and V. We apply

$$\widetilde{A}_{l,b,t,r,h,:} = g_{l,r,h} A_{l,b,t,r,h,:}, \tag{17}$$

using one scalar gate for each layer, projection type, and head. The same gate is broadcast over every discovery example b and token position t. During attribution, gradients on these gate tensors provide one score for each Q/K/V component.

In the first DFCD iteration, the 12 initially selected MLPs are active and all 432 Q/K/V gates are zero. A forward pass computes the mean correct-answer CE over the 47 discovery examples, and a backward pass computes its derivative with respect to every gate. These derivatives form the component attributions for the current circuit state.

B.3 Dynamic Attribution and Search

At each binary state, BiCD temporarily differentiates all gates and computes

$$a_c(S) = \left| \frac{\partial \mathcal{L}(\mathbf{g})}{\partial g_c} \right|_{\mathbf{g} = \mathbf{g}(S)}. \tag{18}$$

The absolute value measures local task sensitivity without treating the derivative sign as component value. A large magnitude can identify direct, compensating, or inhibitory computation whose perturbation materially changes the task loss. Attribution only orders proposals. Every accepted addition or deletion is checked using an exact binary-gate forward pass, after which all scores are recomputed.

DFCD initializes S_0 with the 12 MLP components and selects

$$c_t^+ = \arg\max_{c \in U \setminus S_t} a_c(S_t), \qquad S_{t+1} = S_t \cup \{c_t^+\}, \tag{19}$$

until $\mathcal{L}(S_{t+1}) \leq B_\tau$. Backward pruning starts from this feasible circuit, sorts selected removable components by increasing attribution, and evaluates exact deletions. It accepts the first c satisfying $\mathcal{L}(S_t \setminus \{c\}) \leq B_\tau$, recomputes all scores, and restarts the scan.

B.4 Pseudocode and Termination

Algorithm 1: Bidirectional Circuit Discovery (BiCD)

Data: Model f, discovery set D, component universe U, fixed initial MLP set S_0, retention threshold τ
Result: A feasible circuit S, single-deletion-minimal outside S_0

$L_{\text{full}} \leftarrow \mathcal{L}(U; D)$;
$B_\tau \leftarrow L_{\text{full}}/\tau$;
$S \leftarrow S_0$;
while $\mathcal{L}(S; D) > B_\tau$ **do**
  $a_c \leftarrow |\partial \mathcal{L}(\mathbf{g}; D)/\partial g_c|_{\mathbf{g}=\mathbf{g}(S)}$ for all $c \in U \setminus S$; `// Dynamically rescore inactive components`
  $c^+ \leftarrow \arg\max_{c \in U \setminus S} a_c$;
  $S \leftarrow S \cup \{c^+\}$;
repeat
  $a_c \leftarrow |\partial \mathcal{L}(\mathbf{g}; D)/\partial g_c|_{\mathbf{g}=\mathbf{g}(S)}$ for all $c \in S \setminus S_0$; `// Rescore removable components`
  $A \leftarrow \text{sort}_{c \in S \setminus S_0}(a_c, \text{ ascending})$;
  removed $\leftarrow$ **false**;
  foreach $c \in A$ **do**
    if $\mathcal{L}(S \setminus \{c\}; D) \leq B_\tau$ **then**
      $S \leftarrow S \setminus \{c\}$; `// Accept exact deletion trial`
      removed $\leftarrow$ **true**;
      break;
until *removed* $=$ **false**;
return S;

Both phases terminate for a finite inventory because each forward iteration adds one component and each accepted backward iteration removes one. The MLPs in S_0 remain fixed throughout. At termination, S^* is feasible and every removable $c \in S^* \setminus S_0$ satisfies

$$\mathcal{L}(S^* \setminus \{c\}) > B_\tau. \tag{20}$$

The algorithm stops when no further component can be removed.

B.5 Computational Complexity

Let $M = |U|$ be the number of components. We distinguish three per-operation costs: C_g for one gradient computation over the discovery set, including its model forward and backward passes; C_s for one complete ranking and sort of at most M component scores; and C_e for one independent CE evaluation. These costs are taken as upper bounds over the search states. In particular, C_s already includes the comparisons within a sort.

DFCD. Each iteration computes the gradients, sorts the candidates, adds one component, and evaluates the resulting CE. Since at most M components can be added, the worst-case cost is

$$T_{\text{DFCD}} = O(MC_g + MC_s + MC_e)\,. \tag{21}$$

Back pruning. Each successful iteration removes one component, so there are at most M successful iterations and one final unsuccessful scan. Each iteration requires one gradient computation and one sort, but may evaluate up to M candidate deletions. The worst-case cost is therefore

$$T_{\text{BP}} = O\big(MC_g + MC_s + M^2C_e\big)\,. \tag{22}$$

Overall complexity. Combining the two phases gives

$$T_{\text{BiCD}} = O\big(MC_g + MC_s + M^2C_e\big)\,. \tag{23}$$

Keeping C_g, C_s, and C_e separate distinguishes the costs of gradient computation, sorting, and CE evaluation rather than treating them as equally expensive operations. The gates, current scores, and candidate ordering require $O(M)$ auxiliary working memory, excluding model parameters, activations, and stored experiment histories.

B.6 Component-Count Trajectory

Figure 4 traces the Q/K/V composition across the complete BiCD search. DFCD first adds components, and BP then removes components from the forward endpoint. All 12 MLPs remain fixed throughout.

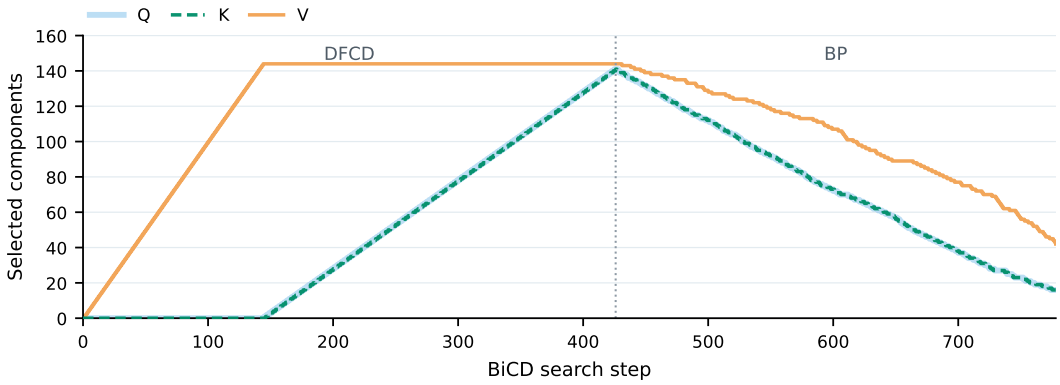


Figure 4: Q/K/V component counts across the connected DFCD and BP trajectories. The vertical dotted line marks the transition after 426 DFCD additions; BP then performs 352 accepted deletions. Q and K counts differ by at most one component, so the translucent light-blue Q line and opaque dashed green K line are nearly superimposed. The 12 fixed MLPs are excluded. Counts change from 141 Q, 141 K, and 144 V at the forward endpoint to 16 Q, 16 K, and 42 V in the final circuit.

C Layer-Wise Circuit Inventory

The 74 selected attention projections are listed by layer below. Every layer's MLP is fixed active, giving 86 components in total.

Table 9: Layer-wise inventory of the final 86-component BiCD circuit: 16 Q, 16 K, 42 V, and 12 fixed MLP components. Empty sets indicate no selected projection of that type.

Layer	Q heads	K heads	V heads
0	0,10	0,10	0,1,3,4,5,6,7,10
1	0,7	0,7	0,4,5,6,7
2	4,8	4,8	0,2,4,8,9
3	–	–	3,5,7
4	5	5	5,9
5	–	–	2,8,10
6	–	–	7,9
7	–	–	0,2,11
8	3,10,11	3,10,11	0,3,10,11
9	1,5,9	1,5,9	1,5,9
10	2,7	2,7	2,7
11	2	2	2,11

D Transfer Beyond the Discovery Template

BiCD and all reproduced baseline circuits are discovered using the same 47 prompts from the `a + b =` template. We then freeze each circuit and evaluate it without additional search or tuning. The full fine-tuned model is evaluated on the same examples as a reproduction reference. Circuit sizes are reported in their native units: BiCD selects Q/K/V components; ACDC, EAP, and EAP-IG select residual-stream edges; and Edge Pruning learns soft edge gates. These counts are therefore not directly interchangeable.

D.1 CROSS-TEMPLATE ARITHMETIC

The cross-template evaluation uses 314 held-out operand pairs that do not overlap with the 47 discovery prompts. Each operand pair is rendered under all four templates used during fine-tuning. Table 10 reports correct-answer Top-1 counts and mean CE for the full-model reference, BiCD, and every reproduced baseline point used in our main comparison.

Table 10: Cross-template zero-ablation results on 314 held-out operand pairs. Each result cell reports Top-1 correct out of 314 (mean CE). "First $\geq$ BiCD" uses the first sampled budget reaching BiCD's discovery-template Top-1.

Model or circuit	Size	`a + b =`	`a + b equals`	`sum a + b =`	`math: a + b =`
Full-model reference	444 components	314 (0.0029)	314 (0.0030)	314 (0.0013)	314 (0.0014)
BiCD	86 components	301 (0.1279)	300 (0.1423)	275 (0.4454)	274 (0.4987)
ACDC reproduction	474 connected edges	249 (0.7835)	241 (0.8345)	237 (0.8766)	239 (0.8916)
EAP, first $\geq$ BiCD	6,396 edges	304 (0.0902)	303 (0.0831)	307 (0.0598)	305 (0.0780)
EAP, 24k	24,000 edges	314 (0.0041)	314 (0.0043)	314 (0.0023)	314 (0.0029)
EAP-IG, first $\geq$ BiCD	3,196 edges	308 (0.0638)	309 (0.0661)	308 (0.0596)	309 (0.0551)
EAP-IG, 8k	8,000 edges	314 (0.0101)	314 (0.0103)	314 (0.0046)	314 (0.0060)
Edge Pruning	1,531 nonzero soft gates	274 (0.3555)	272 (0.4035)	282 (0.3551)	273 (0.4347)

The full model reaches 314/314 Top-1 on every template. The same 86-component BiCD circuit answers 301 and 300 examples correctly on the two unprefixed templates, and 275 and 274 on the `sum` and `math:` templates. Thus the circuit transfers beyond its discovery template, but its retained functionality depends on prompt form. EAP and EAP-IG reach 303–307 and 308–309 correct answers at their first qualifying budgets; the larger 24,000-edge EAP and 8,000-edge EAP-IG circuits recover all 314 answers across templates. ACDC reaches 237–249 correct answers with 474 connected edges (486 selected), while Edge Pruning reaches 272–282 with 1,531 nonzero soft gates.

D.2 TRANSFER TO CAPITAL RECALL

Pretrained GPT-2 small exhibits capital recall, but arithmetic-only fine-tuning causes this behavior to be forgotten under our evaluation. Table 11 compares the complete pretrained model, an arithmetic-only checkpoint, and the arithmetic-plus-capital checkpoint. The two fine-tuned checkpoints use seed 42 and the same arithmetic training configuration, differing in the inclusion of 94 auxiliary capital prompts. Correct-answer next-token Top-1 falls from 45/94 before fine-tuning to 0/94 with arithmetic alone; joint training yields 94/94. This motivates the cross-task evaluation: we ask whether a circuit discovered exclusively from arithmetic prompts also retains a distinct behavior supported by the same full model.

Table 11: Capital recall by complete models, without a circuit mask. Natural and key–value columns report correct-answer next-token Top-1 out of 47; combined metrics use all 94 prompts.

Checkpoint	Natural	Key–value	Combined Top-1	Mean CE
Pretrained GPT-2 small	5/47	40/47	45/94	2.6016
Arithmetic-only fine-tuning	0/47	0/47	0/94	22.3905
Arithmetic + capital fine-tuning	47/47	47/47	94/94	$< 10^{-6}$

We apply the arithmetic circuits to 47 country–capital pairs under natural-language and key–value prompts. For example, the natural-language prompt is `The capital of France is`, and the key–value prompt is `country: France; capital:`; both have the target answer `Paris`. No capital example is used for circuit discovery, and no circuit is changed before evaluation. Table 12 again separates the full-model reference from the reproduced circuit results.

Table 12: Capital-recall transfer using the checkpoint fine-tuned with seed 42. Natural and key–value cells count prompts whose correct-answer token is the Top-1 next token, out of 47 (mean CE); combined columns aggregate all 94 prompts.

Model or circuit	Size	Natural language	Key–value	Combined Top-1	Combined Top-3
Full-model reference	444 components	47 ($< 10^{-6}$)	47 ($< 10^{-6}$)	94/94	94/94
BiCD	86 components	47 (1.23×10^{-4})	46 (0.0502)	93/94	94/94
ACDC reproduction	474 connected edges	18 (2.8395)	25 (2.0579)	43/94	62/94
EAP, 6.4k	6,396 edges	46 (0.1296)	47 (0.0066)	93/94	93/94
EAP, 24k	24,000 edges	47 (4.06×10^{-8})	47 ($< 10^{-6}$)	94/94	94/94
EAP-IG, 3.2k	3,196 edges	47 (0.0043)	47 (0.0163)	94/94	94/94
EAP-IG, 8k	8,000 edges	47 (9.00×10^{-7})	47 (2.38×10^{-7})	94/94	94/94
Edge Pruning	1,531 nonzero soft gates	47 (0.0003)	47 (0.0022)	94/94	94/94

For all 94 prompts, the full fine-tuned model ranks the correct-answer token as the Top-1 next token. The 86-component arithmetic BiCD circuit does so on 93/94 prompts and places the correct-answer token in the next-token Top-3 on all 94 prompts. EAP reaches 93/94 at approximately 6,400 edges and 94/94 at 24,000 edges; EAP-IG reaches 94/94 at both the 3,200-edge and 8,000-edge budgets. ACDC reaches 43/94 Top-1 and 62/94 Top-3, whereas Edge Pruning reaches 94/94 on both measures. These country–capital prompts were auxiliary rows during checkpoint construction, so this experiment measures cross-task circuit retention on known prompts rather than untouched factual generalization.

D.3 Cross-Task Circuit Overlap

Transfer of the arithmetic circuit to capital recall does not by itself show whether the two tasks use the same components. We therefore run BiCD again on the 47 natural-language capital prompts, using the same GPT-2-small checkpoint, component granularity, and $\tau = 0.999$ threshold. Let S_{arith} and S_{capital} denote the independently discovered component sets. Their exact overlap is measured by

$$J(S_{\text{arith}}, S_{\text{capital}}) = \frac{|S_{\text{arith}} \cap S_{\text{capital}}|}{|S_{\text{arith}} \cup S_{\text{capital}}|}. \tag{24}$$

We additionally report overlap separately for Q, K, V, MLP, and each layer.

The capital search returns a 60-component circuit with discovery CE 2.89×10^{-8} and the correct-answer token ranked as the Top-1 next token on all 47 prompts, compared with full-model CE 3.07×10^{-8} and the same Top-1 count. The 86-component arithmetic circuit and the capital circuit share 31 components over a union of 115, giving Jaccard similarity 0.270. All 12 MLPs are included by initialization in both searches. Excluding them, the Q/K/V-only circuits share 19 components over a union of 103, giving Jaccard similarity 0.184.

Table 13: Component overlap between independently discovered arithmetic and capital-recall BiCD circuits.

Component type	Arithmetic	Capital	Intersection	Union	Jaccard
All	86	60	31	115	0.270
Q	16	7	3	20	0.150
K	16	7	3	20	0.150
V	42	34	13	63	0.206
MLP	12	12	12	12	1.000

Table 14: Layer-wise component overlap between the arithmetic and capital-recall circuits. Counts include each layer's MLP.

Layer	Arithmetic	Capital	Intersection	Union	Jaccard
0	13	5	3	15	0.200
1	10	1	1	10	0.100
2	10	2	2	10	0.200
3	4	3	3	4	0.750
4	5	1	1	5	0.200
5	4	4	1	7	0.143
6	3	3	2	4	0.500
7	4	6	1	9	0.111
8	11	6	4	13	0.308
9	10	9	5	14	0.357
10	7	10	3	14	0.214
11	5	10	5	10	0.500

E Dynamic and Static Scoring

E.1 Dynamic Score Variation

Dynamic scoring means that DFCD recomputes every unselected component's attribution after each accepted addition. The archived dynamic-scoring sweep records the variance of the attribution-score change at every forward round. We report its square root so that the trajectory is expressed as a standard deviation on the original score scale:

$$\sigma_t = \mathrm{Std}_{c \in U \setminus S_t} \left[a_c(S_t) - a_c(S_0) \right]. \tag{25}$$

Only components not yet selected at round t enter the statistic. Since the square root is monotonic, replacing variance with standard deviation does not change the maximizing round. Figure 5 shows all 426 forward rounds; round 2 has the maximum standard deviation, 0.290196.

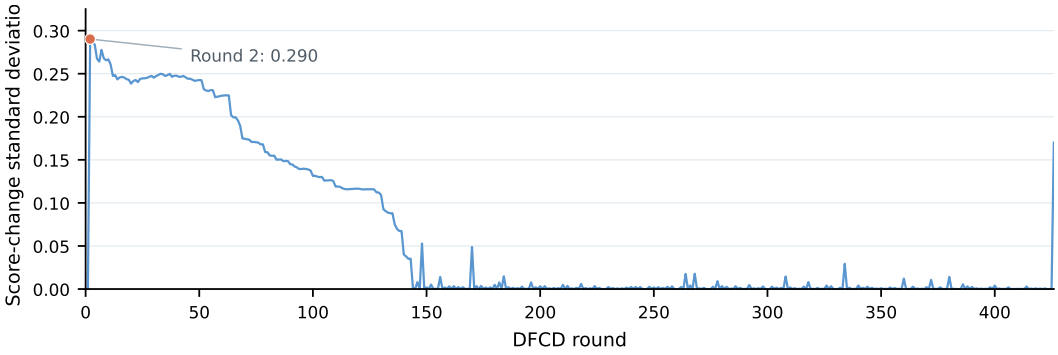


Figure 5: Standard deviation of DFCD attribution-score changes relative to the initial state, computed over components not yet selected at each round.

The trajectory is not constant: even after excluding components already in the circuit, the remaining scores move substantially as the selected subset changes. This is the empirical motivation for recomputing the ranking rather than treating the initial attribution as a permanent component order.

Figure 6 compares the initial V-component scores with round 2, which maximizes the standard deviation of changes among unselected components.

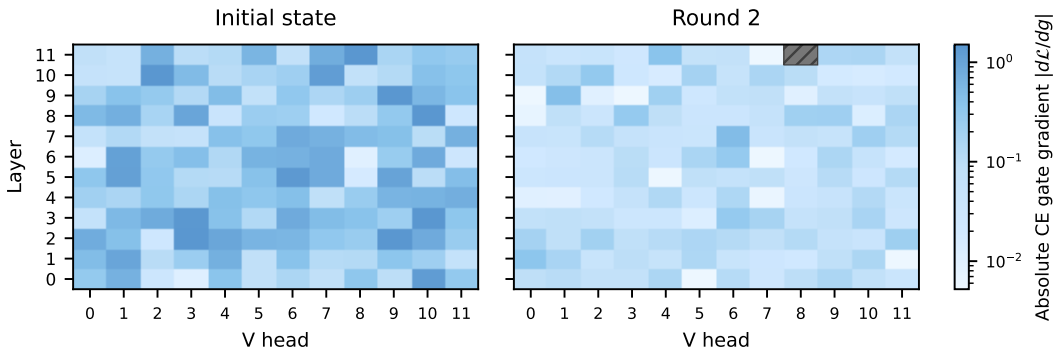


Figure 6: Dynamic V-component rescoring during DFCD.

E.2 Static-Scoring Control

To isolate the contribution of dynamic rescoring, we use a static BiCD control with the same 12-MLP initialization, discovery set, zero-ablation CE objective, threshold, forward-then-backward order, and exact CE acceptance tests. The control computes one ranking at the MLP-only DFCD state and keeps that order throughout forward construction. It then computes one new ranking at the forward endpoint and keeps that order throughout backward pruning. Thus the only intended difference is whether attribution scores are updated after the selected subset changes.

Table 15: Dynamic versus static ranking at $\tau = 0.999$. Both use the same 47 discovery prompts and 314-example zero-ablation evaluation. Exact trials count additions and deletions, including the terminal unsuccessful scan but excluding anchors. Runtime was measured on an NVIDIA GeForce RTX 2060.

Scoring rule	Components	Discovery CE	Held-out CE	Held-out Top-1	Exact trials	Search time
Dynamic rescoring (BiCD)	**86**	0.00386216	0.1279	**301/314**	**2,864**	**354.21 s**
Static rankings	96	0.00388674	0.1627	300/314	7,817	430.63 s

Dynamic rescoring produces a circuit with ten fewer components and one additional correct held-out answer. BiCD ranks components using a first-order gradient approximation, so updating the scores is fast. In contrast, deciding whether a modified circuit satisfies the CE threshold requires a fresh exact evaluation of that circuit. As the subset changes, a static ranking can propose less suitable deletions and require more repeated CE evaluations. The dynamic run performs 2,438 exact deletion trials versus 7,388 for the static control; including additions gives 2,864 versus 7,817 trials. This reduction offsets the additional rescoring cost in the measured implementation. The timing scopes differ slightly: the timer for the dynamic run includes its anchor and final search-set evaluations, whereas the static timer covers forward construction and backward pruning. Despite including those additional evaluations, the dynamic run is 76.42 seconds faster.

F Discovery Metric Choice

F.1 Metrics Used by the Compared Methods

Following the example-based format used by ACDC (Conmy et al., 2023), Table 16 makes each optimization signal concrete using an actual pair from our discovery set. Let x = `39 + 2 =` with correct answer token y = `_41`. The fixed answer-changing derangement pairs it with x^- = `24 + 10 =` and y^- = `_34` for methods that require a corrupt input. The underscore denotes the leading space in a GPT-2 token, p_U is the full-model distribution, and p_S is the subcircuit distribution. These metrics are not interchangeable: correct-answer CE uses the labeled answer, whereas KL-based methods target fidelity to the full output distribution.

Table 16: Discovery objectives and interventions used in the baseline comparison, illustrated with one arithmetic discovery example.

Method	Example prompt/output	Discovery objective	Discovery intervention	Score update
BiCD	`39 + 2 =` $\rightarrow$ `_41`	$-\log p_S(y \mid x)$	Zero Q/K/V components	Dynamic
ACDC	`39 + 2 =` $\rightarrow$ full output distribution	$D_{\mathrm{KL}}(p_U(\cdot \mid x) \| p_S(\cdot \mid x))$	Zero residual-stream edges	Iterative
EAP	`39 + 2 =` $\rightarrow$ `_41`	First-order correct-answer CE attribution	Zero residual-stream edge contributions	Static
EAP-IG	`39 + 2 =` $\rightarrow$ `_41`; `24 + 10 =` $\rightarrow$ `_34`	Path-integrated correct-answer CE attribution	EAP-IG-inputs; corrupt-to-clean input interpolation, five steps	Static
Edge Pruning	`39 + 2 =` $\rightarrow$ full output distribution	KL fidelity with a sparsity penalty	Learn soft residual-stream edge gates	Learned

F.2 BiCD Metric Sweep

We rerun the complete bidirectional search on the same checkpoint at $\tau = 0.999$ with several discovery objectives and intervention baselines. Every resulting circuit is then evaluated in the same standalone setting by zeroing all components outside the selected set. Table 17 reports circuit size, held-out performance, and search time. Across five training seeds, CE-based BiCD achieves a mean held-out Top-1 accuracy of 97.45% with 94 components on average (Table 26). This mean exceeds the single-seed accuracies of the other metric settings in Table 17. Correct-answer CE uses the full-model-to-circuit CE ratio; the other objectives use the fraction of the initial-to-full-model loss gap recovered. The corrupt-patching objectives satisfy their intervention-specific discovery criteria but transfer poorly to standalone zero ablation.

Table 17: BiCD metric sweep. Search uses 47 prompts; evaluation uses 314 held-out operand pairs with all unselected components zeroed. Time is the recorded search wall time on an NVIDIA GeForce RTX 2060, excluding model loading and held-out evaluation.

Objective	Intervention	Components	Eval. CE	Top-1	Top-3	Time (s)
Correct-answer CE	Zero	86	0.1279	95.86%	100%	354.21
KL divergence	Zero	66	0.4064	88.22%	98.73%	449.75
Probability difference	Zero	85	0.1054	96.18%	99.68%	387.35
Logit difference	Zero	94	0.1318	95.54%	100%	1,069.78
Correct-answer CE	Corrupt patch	111	0.6003	78.66%	97.45%	77.82
KL divergence	Corrupt patch	95	0.8343	77.07%	97.13%	40.10

F.3 Retention-Threshold Sweep

We sweep τ on the same checkpoint using zero-ablation CE. All searches use the same 47 discovery prompts, and Table 18 evaluates their circuits on the same 314 held-out examples. Among these sampled settings, $\tau = 0.999$ yields the smallest circuit while retaining 301 correct answers; $\tau = 0.99$ gives the highest Top-1 count with nine additional components. Increasing τ tightens the discovery CE budget, but neither the final greedy-search size nor held-out accuracy is monotonic in τ.

Table 18: CE retention-threshold sweep. The selected setting is $\tau = 0.999$. Component counts include the 12 fixed MLPs.

τ	Components	Discovery CE	Held-out CE	Held-out Top-1
0.9	96	0.00430609	0.0950	304/314
0.95	102	0.00416325	0.1285	303/314
0.98	89	0.00380887	0.1442	297/314
0.99	95	0.00391878	0.0801	305/314
0.995	94	0.00391110	0.1715	297/314
0.999	**86**	0.00386216	0.1279	301/314
0.9999	90	0.00369629	0.1727	295/314

G Task Fragility After Arithmetic Fine-Tuning

Docstring and Induction in the ACDC task suite (Conmy et al., 2023) use task-specific models, while the Tracr tasks (Lindner et al., 2023) use compiled architectures, so these tasks cannot be directly evaluated in the same model setting using our GPT-2 small checkpoint. We therefore restrict the direct comparison to IOI (Wang et al., 2023) and Greater-Than (Hanna et al., 2023). The original studies use pretrained GPT-2 small, whereas we evaluate GPT-2 small after arithmetic-plus-capital fine-tuning.

G.1 Tasks Supported by Pretrained GPT-2 Small

We reproduce IOI and Greater-Than on pretrained GPT-2 small and then evaluate the arithmetic-plus-capital fine-tuned checkpoint on exactly the same 100 examples. We abbreviate the latter as Arithmetic-FT below. No component mask is applied in this comparison. The IOI task-restricted metric asks whether the indirect-object token has a higher logit than the subject token. The Greater-Than task-restricted metric asks whether the highest-probability suffix among `00–99` is greater than the prompt's endpoint. We additionally report full-vocabulary Top-1 accuracy, which requires the correct IOI token or a valid later-year suffix to be the model's highest-logit token over the complete vocabulary.

Table 19: Full-model behavior before and after arithmetic-plus-capital fine-tuning on two tasks originally studied in pretrained GPT-2 small. Original-task candidate accuracy restricts the comparison to each task's answer candidates as defined above; full-vocabulary Top-1 uses the complete vocabulary. Both checkpoints use the same 100 examples.

	Original-task candidates		Full-vocabulary Top-1	
Task	Pretrained	Arithmetic-FT	Pretrained	Arithmetic-FT
IOI	100%	56%	99%	0%
Greater-Than	100%	52%	100%	0%

The pretrained checkpoint reproduces both behaviors, while the arithmetic-fine-tuned checkpoint loses 44 percentage points of IOI pairwise accuracy and 48 points of Greater-Than suffix accuracy. Full-vocabulary Top-1 falls to 0% for both tasks. Fine-tuning causes the model to forget tasks it could previously perform, making comparative evaluation across circuit discovery methods very difficult.

G.2 Tasks Using Task-Specific Subject Models

The remaining ACDC tasks do not use pretrained GPT-2 small as their original subject model. Docstring uses a four-layer attention-only model, Induction uses a two-layer Redwood attention model, and the Tracr tasks use separately compiled transformers. For completeness, we apply the textual Docstring and repeated-sequence Induction probes to the complete arithmetic-fine-tuned GPT-2 small. These are diagnostic probes only, not reproductions of the original ACDC task models.

Docstring example. The following evaluated prompt asks for the next parameter name in the documentation. The target continuation is `node`:

```
def response(self, group, item, ret, option, node, default):
    """motion rock evidence

    :param ret: ban standing
    :param option: title aim
    :param
```

Induction example. Writing token identities as `A`, `B`, and `C`, a shortened schematic prefix `A B C A B` should be continued with `C`. The evaluated probes repeat a randomly sampled 24-token block twice and score next-token predictions throughout the second copy.

Table 20: Record of the arithmetic-fine-tuned GPT-2 small on ACDC tasks whose original subject model is not GPT-2 small. "N/A" denotes an incompatible model interface rather than a failed prediction.

Task	Original subject model	Arithmetic-FT GPT-2 probe	Result
Docstring	Four-layer attention-only model	Correct token above all designated wrong tokens	11%; global Top-1 0%
Induction	Two-layer Redwood attention model	Accuracy on the repeated half of a token sequence	0%; mean NLL 25.472
Tracr-xproportion	Tracr-compiled proportion model	Incompatible vector-valued interface	N/A
Tracr-reverse	Tracr-compiled reverse model	Incompatible sequence-to-sequence interface	N/A

These results document a limitation of direct cross-task transfer with GPT-2 small. Tasks may require a different trained model or execution interface, and even behaviors already present in pre-trained GPT-2 small can be fragile under subsequent task fine-tuning. A circuit-level transfer comparison is meaningful only after confirming that the complete target checkpoint still performs the target task.

H ACDC-STYLE EDGE-LEVEL BICD

We additionally replace BiCD's component gates with the writer-to-reader edge inventory used by ACDC. DFCD starts from an empty edge set with no fixed MLP-backbone edges, dynamically rescores the absent edges after every accepted addition, and stops when the zero-ablation CE threshold is met. Backward pruning then dynamically rescores and tests deletions from that candidate graph. This experiment holds the task, 47-prompt discovery set, CE threshold, and forward-then-backward logic fixed while changing the search unit from a Q/K/V component to an individual graph edge.

Table 21: ACDC and edge-level BiCD searches on an NVIDIA GeForce RTX 2060. Both use the same 47 discovery prompts and 314-example zero-ablation evaluation. ACDC uses its native KL threshold 0.001; edge-level BiCD uses CE retention $\tau = 0.999$.

Method	Search objective	Selected edges	Connected edges	Held-out CE	Held-out Top-1	Search time
ACDC	Native KL	486	474	0.7835	249/314 (79.30%)	1:57:48
Edge-level BiCD	Zero-ablation CE	309	280	0.9164	239/314 (76.11%)	27:39:02

The edge-level search has 32,491 candidate edges. DFCD adds 32,305, excluding 186 before backward pruning removes 31,996 and leaves 309 selected edges, of which 280 are input-to-output connected. The final circuit answers all 47 discovery prompts correctly with CE 0.00391550, compared with full-model CE 0.00395836, and achieves CE retention 1.01094. Its held-out accuracy is 239/314 despite satisfying the discovery constraint.

Under the same edge-level model abstraction, bidirectional search produces a more compact circuit than one-way pruning at the evaluated operating points, retaining 194 fewer connected edges than ACDC. However, searching the graph incurs substantial computational cost: the edge-level BiCD run takes 99,542.21 seconds, excluding model loading and anchor evaluations. This cost motivates our component-level formulation, which treats the model as a collection of Q/K/V and MLP components rather than searching over individual graph edges.

Edge-level complexity. Let E be the number of candidate edges. For the edge-level implementation, let C_g denote the cost of one gradient computation, including its forward and backward passes; C_s the cost of sorting at most E edge scores; and C_e the cost of one independent CE evaluation. DFCD adds at most E edges, and back pruning has at most E successful deletion rounds plus a final unsuccessful scan. Each pruning round may check up to E candidate deletions. There are therefore $O(E)$ gradient computations and ranking steps, and at most $O(E^2)$ additional CE evaluations, giving

$$T_{\text{edge-BiCD}} = O\big(EC_g + EC_s + E^2C_e\big)\,. \tag{26}$$

Here C_s includes the comparisons within a complete sort and also bounds DFCD's maximum-score selection. The per-operation costs refer to the edge-level implementation, not the component-level implementation. In this experiment, $E = 32{,}491$, compared with 444 components, substantially increasing the number of possible search operations.

I Scaling and Arithmetic Complexity

We extend BiCD beyond the GPT-2-small two-digit setting in two directions: model scale and arithmetic complexity. Each setting uses its own fine-tuned checkpoint and 47 discovery prompts, without projecting a circuit from another checkpoint.

I.1 Discovery at Larger Scale

At the same Q/K/V-plus-MLP granularity, GPT-2 medium has

$$24 \text{ layers} \times 16 \text{ heads} \times 3 + 24 = 1{,}176 \tag{27}$$

components, compared with 444 in GPT-2 small. The medium checkpoint uses the same 2,401 operand pairs, 70/15/15 split, four arithmetic templates, and auxiliary capital prompts as the main experiment. Fine-tuning uses seed 42, learning rate 10^{-4}, FP16, 8-bit AdamW, batch size 8 with gradient accumulation over two batches, and at most 50 epochs. The best validation checkpoint is obtained at epoch 31.

DFCD adds 1,082 Q/K/V components to the 24 fixed MLPs; backward pruning then removes 965 components. The resulting circuit contains 141 components, or 12.0% of the full inventory. Both the full model and circuit answer all 47 discovery prompts correctly under the $\tau = 0.999$ CE constraint.

Table 22: GPT-2-medium BiCD search on an NVIDIA GeForce RTX 2060. CE and Top-1 use the 47 discovery prompts. Runtime excludes model loading and tokenization.

$\lvert U\rvert$	$\lvert S\rvert$	Q/K/V/MLP	Forward adds	Backward removals	Full/Circuit CE	Circuit Top-1	Time (s)
1,176	141	26/26/65/24	1,082	965	1.4464×10^{-4}/1.4023×10^{-4}	47/47	2,256.89

I.2 Held-Out Functionality

Table 23 evaluates the frozen medium circuit by zeroing every component outside the selected set. Evaluation uses the same 314 held-out two-digit pairs as the main experiment, rendered under all four fine-tuning templates.

Table 23: GPT-2-medium zero-ablation evaluation on 314 held-out pairs. Top-1 and Top-3 entries are correct counts out of 314.

Template	Full Top-1	BiCD Top-1	Full CE	BiCD CE	BiCD Top-3
`a + b =`	313	302	0.0384	0.1213	314
`a + b equals`	312	299	0.0374	0.1533	314
`sum a + b =`	311	304	0.0344	0.0883	314
`math:  a + b =`	312	303	0.0319	0.1084	314

The medium circuit reaches 95.22–96.82% Top-1 across templates, compared with 99.04–99.68% for the full model, and places the correct answer in the Top-3 on every example.

I.3 Three-Digit Arithmetic and Circuit Overlap

The three-digit experiment uses operands from 100 through 499 and sums from 200 through 998. Restricting the space-prefixed answer to one GPT-2 token leaves 46,735 ordered pairs, split into 32,714 training, 7,010 validation, and 7,011 test pairs. Discovery reserves 47 test pairs; the remaining 6,964 pairs are evaluated under all four templates. Training uses seed 42, learning rate 10^{-4}, batch size 16, SDPA, and 94 auxiliary capital prompts. With a maximum of 100 epochs and patience 10, training stops at epoch 75 and selects the epoch-65 checkpoint, whose validation CE is 3.22335×10^{-5}.

DFCD adds 426 components to the 12 fixed MLPs, and backward pruning removes 296. The resulting 142-component circuit contains 38 Q, 38 K, 54 V, and 12 MLP components, or 32.0% of the full inventory. Both the full model and circuit answer all 47 discovery prompts correctly; their discovery CEs are 3.89586×10^{-6} and 3.57121×10^{-6}, respectively, giving retention 1.09091 at $\tau = 0.999$. Search takes 394.70 seconds on the same NVIDIA GeForce RTX 2060, excluding model loading and tokenization.

Table 24: Three-digit zero-ablation evaluation using the GPT-2-small checkpoint and the same 142-component circuit across templates. Correct counts are out of 6,964 held-out pairs.

Template	Full Top-1	BiCD Top-1	Full CE	BiCD CE	BiCD Top-3
`a + b =`	6,963	6,403	0.0003834	0.5986	6,868
`a + b equals`	6,963	6,401	0.0004039	0.6097	6,867
`sum a + b =`	6,963	6,335	0.0001891	0.8059	6,863
`math:  a + b =`	6,963	6,311	0.0010087	0.7760	6,858

The full model answers 6,963/6,964 held-out examples correctly on every template. The frozen BiCD circuit reaches 90.62–91.94% Top-1 across templates. Its larger held-out CE, despite satisfying the discovery constraint, separates functionality on the 47 discovery prompts from generalization to unseen operand pairs.

Both the two-digit and three-digit circuits use GPT-2 small, so their component coordinates are aligned despite separate fine-tuning. They share 58 components over a union of 170, giving Jaccard similarity 0.341. Excluding the 12 MLPs retained by construction leaves 46 shared Q/K/V components over a union of 158, or Jaccard similarity 0.291. This compares selected component locations, not equivalence of the computations learned by the two checkpoints.

Table 25: Component-set overlap between independently discovered GPT-2-small circuits for two-digit and three-digit addition.

Component type	Two-digit	Three-digit	Intersection	Union	Jaccard
All	86	142	58	170	0.341
Q/K/V only	74	130	46	158	0.291
Q	16	38	10	44	0.227
K	16	38	10	44	0.227
V	42	54	26	70	0.371
MLP	12	12	12	12	1.000

J REPRODUCIBILITY ARTIFACTS

The saved BiCD search results include the selected component sets and the addition and removal histories. Circuit performance is reported using correct-answer cross entropy and top-k accuracy.

J.1 FIVE-SEED BICD ROBUSTNESS

We repeat fine-tuning and CE-based BiCD with training seeds 42–46. Other metric settings and baseline methods are evaluated on the checkpoint fine-tuned with seed 42. All five fine-tuning runs use Transformers 4.57.6, SDPA fine-tuning, learning rate 10^{-4}, batch size 16, and at most 50 epochs, selecting the checkpoint with the lowest validation CE. The data split seed remains 42, and the same 47 discovery prompts and 314 held-out prompts are used throughout. Each checkpoint is searched independently with $\tau = 0.999$ and the 12 MLPs fixed in the initial circuit. Evaluation zeros all components outside the selected circuit.

Table 26: CE-based BiCD across five training seeds. Top-1 and CE are measured on the 314 held-out prompts. The final row reports the mean and sample standard deviation across seeds. Search wall times are recorded on an NVIDIA GeForce RTX 2060.

Seed	Components	Full Top-1	BiCD Top-1	BiCD CE	Time (s)
42	86	100.00%	95.86%	0.12791	237.18
43	118	100.00%	100.00%	0.00555	423.76
44	86	99.68%	98.41%	0.06222	392.92
45	93	100.00%	98.73%	0.02394	366.48
46	87	100.00%	94.27%	0.18793	231.02
Mean $\pm$ SD	94.00 ± 13.73	$99.94 \pm 0.14\%$	$97.45 \pm 2.33\%$	0.08151 ± 0.07572	330.27 ± 90.13

The seed-42 run in Table 26 is a separate repeat of the search reported in Table 17; it produces the same circuit and search order, with its own recorded runtime.