Automated Circuit Discovery for Mechanistic Interpretability

Ege Erdogan

February 5, 2025

Goal: Identify neural network components that perform specific tasks, to ensure safety, more effective debugging, and better understanding of model internals.

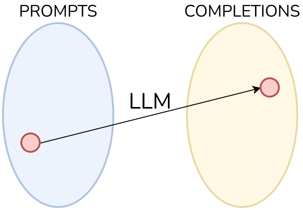
Problem formulation: Given a computational graph of a model, find a minimal subgraph (**circuit**) that performs a specific task.

Challenges

1. **Isolate** computation for a specific task.
2. **Decompose** neural network into components.
3. **Discover** a circuit of components.

→ Three steps of the **mechanistic interpretability workflow**.

The Mechanistic Interpretability Workflow



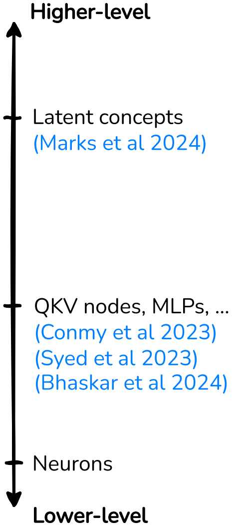
1. **Isolate behavior** by creating prompts with completions following well-defined rules.

Greater-than: The war lasted from 1517 to 1519.

Induction: Vernon Dursley and Petunia Dursley.

Indirect Object Identification: When John and Mary went to the store, Mary gave a bottle of milk to John.

Mechanistic Interpretability Workflow



1. **Isolate behavior** by creating prompts with completions following well-defined rules.
2. **Decompose the neural network** with a specific granularity to obtain the computational graph.

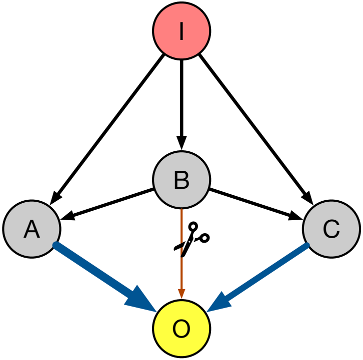


Figure from (Conmy et al., 2023)

1. **Isolate behavior** by creating prompts with completions following well-defined rules.
2. **Decompose the neural network** with a specific granularity to obtain the computational graph.
3. **Prune the computational graph** to obtain a sparse task-specific circuit.

Automatic Circuit DisCovery (ACDC)

Work on circuit discovery has led to identification of circuits performing precise operations.

e.g. for GPT-2 computing the greater-than operation (Hanna et al., 2023)

However, pre-ACDC work on circuit discovery relies on manual inspection $\rightarrow$ **Hard to scale to large models**.

Thus **automating** circuit discovery has great potential benefit in understanding large models.

ACDC (Conmy et al., 2023) - Algorithm

Goal is to remove as many edges as possible from the computational graph G while maintaining the output distribution for the specific task.

Idea: Greedy iterative algorithm

Let H denote the iteratively updated graph. Iterate from output nodes to input nodes.

For each incoming edge e of a node, remove it ($\mathsf{H}' := \mathsf{H} - \{e\}$) permanently if **difference in KL-divergence between output distributions is less than a threshold** ($\tau > 0$):

$$D_{\mathrm{KL}}\left(p_{\mathsf{G}} \| p_{\mathsf{H}'}\right) - D_{\mathrm{KL}}\left(p_{\mathsf{G}} \| p_{\mathsf{H}}\right) < \tau,$$

i.e. removing e has minimal effect on the output distribution.
$\implies$ e **not included in the task circuit.**

ACDC (Conmy et al., 2023) - Design Choices

How are edges removed? *Interchange ablations.* Replace activations with activations from non-task inputs.

Keeps ablated activations within relevant values for actual inputs, so is preferred over zero or dataset mean ablations.

Why KL divergence? Straightforward, task-independent.

Empirically shown to be more stable and effective than using task metrics (e.g. logit loss).

Task metrics can be "over-optimized", i.e. the circuit outperforms the base model.

ACDC (Conmy et al., 2023) - Evaluation

View the problem as **edge classification** (binary) and measure TPR/FPR.

Use circuits discovered in previous work as “ground truth,” which is prone to human error.

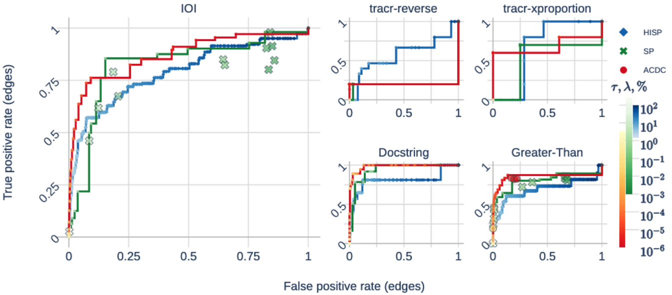
Subgraphs should satisfy:

- **Sufficiency** (high TPR): contains the relevant circuit
- **Necessity** (low FPR): does not contain unrelated nodes

ACDC (Conmy et al., 2023) - Results

ROC curves by varying τ, want larger area beneath the curve.

Previous work HISP and SP prune model components (MLPs, attention heads,...). ACDC prunes edges (more fine-grained).



ACDC generally better, but fails in some problems.

Zero-ablations or optimizing for task metrics in some tasks lead to better performance.

ACDC (Conmy et al., 2023) - Limitations

Scalability. Requires one forward pass for each edge ablation, costly for large models and datasets.

Robustness. Parent iteration order may have a detrimental effect on results (Appendix J).

Optimality. Unlikely that local greedy choices will lead to globally optimal solutions.

1. Can miss out on interactions between edges.
2. Being affected by parent iteration order also supports this.

Later Work on Automated Circuit Discovery

Edge Attribution Patching (EAP). Approximate effect of ablating an edge on the task metric L using a first-order Taylor approximation:

$$L(\mathrm{x} \mid e_{\text{ablated}}) - L(\mathrm{x}) \approx (e_{\text{clean}} - e_{\text{ablated}})^T \frac{\partial L(\mathrm{x} \mid e_{\text{clean}})}{\partial e_{\text{clean}}}$$

- **Scalable**. Two forward passes and one backward pass for N ablations, rather than N forward passes in ACDC.
- **Global**. Picks top k edges from the computational graph.

Shows better results than ACDC despite low correlation between approximate and true scores.

Running ACDC after EAP leads to even better performance.

Scalability Through Optimization (Bhaskar et al., 2024)

Edge Pruning (EP). Learn a binary mask $\mathbf{z}$ over edges, relaxed as $\mathbf{z} \in [0,1]^{N_{\text{edge}}}$ during optimization. Minimize KL-divergence between outputs of original graph $\mathbf{G}$ and pruned graph $\mathbf{H}$:

$$\mathbf{z}^* = \arg\min_{\mathbf{H}} D_{\mathrm{KL}}\left(p_{\mathbf{G}} \| p_{\mathbf{H}}\right)$$

subject to sparsity constraint $1 - |\mathbf{H}|/|\mathbf{G}| \geq c$.

Gradient-based optimization $\implies$ parallelizable

- Outperforms both ACDC and EAP, although slower on small datasets.
- Scales more effectively to larger datasets (100K samples) and models (13B params).

Higher-Level Circuits (Marks et al., 2024)

Are architectural components the right level of granularity?

They can be polysemantic: circuits are not unique to specific behaviors.

Idea: Decompose components' outputs as sparse combinations of features learned by SAEs + error terms.

$\implies$ Graph of interpretable features, and then use Syed et al. (2023)'s linear approximations (EAP) to discover circuits.

Outcomes

- Avoids polysemanticity and obtains more directly interpretable circuits. Thus it is more reliably usable in downstream applications.
- Can discover unanticipated behaviors without task-specific datasets.

We covered two main lines of circuit discovery research:

1. Towards more **scalable** search/optimization methods:
   ACDC → EAP → EP
2. Towards more **expressive/interpretable** decompositions:
   model component circuits → SAE feature circuits

Further Open Problems

- Automate circuit disovery for more complex behaviors.
- Find downstream applications of circuits (e.g. Shift (Marks et al., 2024) for reducing bias in models).

Food for thought: In addition to these methods, could we discover more strongly task-specific circuits by explicitly minimizing performance on unrelated tasks?

- Bhaskar, Adithya et al. (2024). *Finding Transformer Circuits with Edge Pruning*. DOI: `10.48550/arXiv.2406.16778`. arXiv: `2406.16778 [cs]`. URL: `http://arxiv.org/abs/2406.16778`. Pre-published.
- Conmy, Arthur et al. (2023). "Towards Automated Circuit Discovery for Mechanistic Interpretability". In: *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.
- Hanna, Michael et al. (2023). "How Does GPT-2 Compute Greater-than?: Interpreting Mathematical Abilities in a Pre-Trained Language Model". In: Thirty-Seventh Conference on Neural Information Processing Systems.

Marks, Samuel et al. (2024). *Sparse Feature Circuits: Discovering and Editing Interpretable Causal Graphs in Language Models*. DOI: 10.48550/arXiv.2403.19647. arXiv: 2403.19647 [cs]. URL: http://arxiv.org/abs/2403.19647. Pre-published.

Syed, Aaquib et al. (2023). *Attribution Patching Outperforms Automated Circuit Discovery*. DOI: 10.48550/arXiv.2310.10348. arXiv: 2310.10348 [cs]. URL: http://arxiv.org/abs/2310.10348. Pre-published.