# BlackboxNLP-2025 MIB Shared Task: Exploring Ensemble Strategies for Circuit Localization Methods

**Philipp Mondorf, Mingyang Wang, Sebastian Gerstner, Ahmad Dawar Hakimi, Yihong Liu, Leonor Veloso, Shijia Zhou, Hinrich Schütze, Barbara Plank**
LMU Munich & Munich Center for Machine Learning
**Correspondence:** p.mondorf@lmu.de

## Abstract

The Circuit Localization track of the Mechanistic Interpretability Benchmark (MIB) evaluates methods for localizing *circuits* within large language models (LLMs), i.e., subnetworks responsible for specific task behaviors. In this work, we investigate whether ensembling two or more circuit localization methods can improve performance. We explore two variants: parallel and sequential ensembling. In parallel ensembling, we combine attribution scores assigned to each edge by different methods—e.g., by averaging or taking the minimum or maximum value. In the sequential ensemble, we use edge attribution scores obtained via EAP-IG as a warm start for a more expensive but more precise circuit identification method, namely edge pruning. We observe that both approaches yield notable gains on the benchmark metrics, leading to a more precise circuit identification approach. Finally, we find that taking a parallel ensemble over various methods, including the sequential ensemble, achieves the best results. We evaluate our approach in the BlackboxNLP 2025 MIB Shared Task, comparing ensemble scores to official baselines across multiple model–task combinations.

## 1 Introduction

The Circuit Localization track of MIB (Mueller et al., 2025) evaluates the ability of interpretability methods to identify *circuits*, i.e., subnetworks in language models that are responsible for solving a specific task. Its performance metrics measure how well the subnetwork can perform the given task on its own. As proposed by the shared task, this is measured in two specific ways: *circuit performance ratio* (CPR) and *circuit-model difference* (CMD). The first metric, CPR, measures how well the method finds performant circuits at various circuit sizes. The second metric, CMD, measures how closely the circuit's behavior resembles the model's task-specific behavior at various circuit sizes (Mueller et al., 2025). Typically, a circuit localization method gives an attribution score to each edge in the network; it is then possible to choose different thresholds for including an edge in the circuit. Therefore, CPR and CMD scores are integrated over different circuit sizes to give a more holistic picture of each method.

In this work, we investigate whether ensembling two or more baseline methods can lead to better performance. We hypothesize that different methods might be biased towards different model components, and that ensembling them might yield a more robust picture. Specifically, we explore two variants of ensembling: parallel and sequential. In parallel ensembling, we gather the scores assigned to each edge by different methods, typically by averaging, but also by taking the maximum or minimum score. In sequential ensembling, we first warm-start with EAP-IG (Sundararajan et al., 2017; Hanna et al., 2024) attribution scores, then apply edge pruning with attribution score sign recovery.

We evaluate our methods on both the private and public test sets. Our sequential ensemble method with warm-start edge pruning (`s-ens`) improves performance across tasks over the EAP-IG-inputs baseline, while the parallel ensemble (`p-ens`) also yields consistent gains. Motivated by these results, we combine both strategies—integrating warm-start edge pruning with the three EAP-based methods for parallel ensembling—into `hybrid-ens`, a hybrid ensemble, which achieves the best overall performance on the leaderboard, with the lowest CMD score (lower is better) and highest CPR score (higher is better). Our code is publicly available.[1]

## 2 Background

In this section, we describe the models, tasks, and base methods used in our experiments for the shared task.

[1] https://github.com/cisnlp/MIB-circuit-track



*Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 537–542

### 2.1 Models and tasks

To showcase our method, we use the three mandatory model/task combinations: (1) GPT-2-Small (Radford et al., 2019) on IOI (indirect object identification, Wang et al., 2023); (2) Qwen-2.5-0.5B (Yang et al., 2024) on IOI; (3) Qwen-2.5-0.5B on MCQA (multiple choice question answering without task-specific knowledge, Wiegreffe et al., 2025).

We use both datasets in the version provided by Mueller et al. (2025). Each one has train/validation/test splits, plus a private test set that is used for the leaderboard.[2]

### 2.2 Base methods

We experiment with various base methods in our parallel and sequential ensemble settings.

A commonly used method is *edge activation patching* (Vig et al., 2020; Finlayson et al., 2021). It computes the effect of replacing the activation of an edge with its activation on the counterfactual input. This is very inefficient (it needs a separate forward pass for each edge), so various approximations have been proposed. We use the following three as base methods (as implemented by (Mueller et al., 2025)): (1) EAP (*edge attribution patching*, Nanda, 2023; Syed et al., 2024), (2) EAP-IG-inputs (*edge attribution patching with integrated gradients over inputs*, Sundararajan et al., 2017; Hanna et al., 2024), and (3) EAP-IG-activations (Sundararajan et al., 2017; Marks et al., 2025). They are among the best performing methods according to Mueller et al. (2025), with EAP-IG-inputs being the best one overall, so we use EAP-IG-inputs as the baseline for comparison in Section 5.

We further incorporate edge pruning (Bhaskar et al., 2024), a more precise but computationally expensive method, into the ensemble framework.

## 3 Method Overview

We explore two variants of ensembling: parallel and sequential. In parallel ensembling, we gather the scores different methods give to any edge – for example using an average over methods, but we also experiment with taking the maximum or minimum score. In sequential ensembling, we first use EAP-IG-inputs as a warm start, and then perform edge pruning.

[2] https://huggingface.co/spaces/mib-bench/leaderboard

### 3.1 Parallel ensembling

As implemented by Mueller et al. (2025), a circuit detection method (*base method*) gives a score to each edge of the model. We pre-compute these scores for a range of methods. As a result, each edge gets different *base scores*. Our goal is to consolidate these to one *final score* for each edge.

#### 3.1.1 Comparing base scores across methods

We first have to answer the question whether the base scores are comparable across methods.

The EAP variants (first three methods above) are all designed to be approximations of edge activation patching. We therefore assume the scores to be comparable across these methods.

The original edge pruning method obtains a mask (i.e., importance score) for each edge in the range [0, 1] without the sign information. Therefore, the importance score value is not comparable with the EAP-based methods. To make the scores of different methods in a matching space, we use the method described in Section 3.2 to recover the sign information for a better hybrid ensemble (detailed description in Section 3.3).

#### 3.1.2 Reduction methods

We experiment with four score reduction methods across base approaches: (1) mean, (2) weighted average, (3) maximum, and (4) minimum. Interpreting scores as binary indicators, the maximum corresponds to a union of circuits (edges selected by any method), while the minimum corresponds to an intersection (edges selected by all methods). Ultimately, taking the mean yields the best results.

### 3.2 Sequential ensembling

The main idea of our sequential ensemble approach is to use the attribution scores produced by a fast circuit identification method to warm-start a slower, more precise method, thereby achieving faster convergence and further refining the initial scores. Specifically, we use edge attribution values computed via EAP-IG-inputs (Hanna et al., 2024) to initialize the learnable mask—specifically, the learnable log alpha parameters—of edge pruning (Bhaskar et al., 2024).

We convert attribution scores to hard-concrete log alpha values as follows. First, we take the absolute values of the edge attribution scores. These unsigned importance scores are grouped per layer and rank-normalized within each group to $[0, 1]$. We then fit a logistic mapping to these ranks so

that its average "keep" probability matches a predefined starting edge sparsity (a hyperparameter), yielding per-edge keep probabilities. Finally, we invert the hard-concrete expectation to obtain log alpha values via the stretched-sigmoid inverse, with standard clipping.

Once we initialize the log alphas as described, we refine the scores via edge pruning (Bhaskar et al., 2024). After training, we recover signed attribution scores from the refined log alpha values. To this end, we first compute the mask $\mathbf{z}$ as defined by Bhaskar et al. (2024):

$$\begin{aligned}
\mathbf{u} &\sim \mathrm{Uniform}(\varepsilon,\, 1-\varepsilon) \\
\mathbf{s} &= \sigma\left(\frac{1}{\beta}\cdot\log\frac{\mathbf{u}}{1-\mathbf{u}} + \log\alpha\right) \\
\tilde{\mathbf{s}} &= \mathbf{s}\times(r-l)+l \\
\mathbf{z} &= \min\bigl(1,\,\max(0,\,\tilde{\mathbf{s}})\bigr)
\end{aligned}$$

where $\varepsilon = 10^{-6}$, $\frac{1}{\beta} = \frac{2}{3}$, and $[l, r] = [-0.1, 1.1]$. Note that the mask values are positive, i.e., $\mathbf{z} \in [0, 1]$, and thus contain no sign information (i.e., whether an edge contributes positively or negatively to the model's behavior). To recover sign information, we experiment with two approaches:

1. Reuse the sign from the initial EAP-IG-inputs attribution values, or

2. Compute a forward pass of the model in which each edge's output is a weighted combination of its original activation and a reference activation, determined by the learned mask values

$$\begin{aligned}
y_i = f_i\Bigl(&z_{0i}\,y_0 + (1-z_{0i})\,\tilde{y}_0 \\
&+ \sum_{\substack{1\le j<i \\ j \text{ upstream of } i}} \bigl(z_{ji}\,y_j + (1-z_{ji})\,\tilde{y}_j\bigr)\Bigr).
\end{aligned}$$

   and then compute the gradients of a metric $M$ (e.g., the KL divergence between circuit and model) with respect to the mask $\frac{\partial M}{\partial z}$. The sign of this gradient indicates whether the edge is useful for the task behavior (positive) or not useful (negative). We call this approach *z-score attribution*.

When performing hyperparameter optimization, we include a boolean that specifies whether the first or second approach is used. More details on our hyperparameters and the optimization procedure can be found in Section 4 and Appendix A.

### 3.3 Final submissions

After preliminary experiments, we finally submitted three different versions of ensembling:

**`s-ens`** (sequential ensembling, i.e., warmstart edge pruning with Z-score attribution): We use edge attribution values computed via EAP-IG-inputs to initialize the learnable mask of edge pruning as described above in section 3.2.

**`p-ens`** (parallel ensembling): We calculate the average scores (non-weighted) from the three EAP variants for all three model-task combinations.

**`hybrid-ens`**: We combine the parallel and sequential strategies by taking the unweighted average over all four methods—the three EAP variants and warm-start edge pruning—for all model–task combinations.

## 4 Experimental Setup

**Warmstart edge pruning** Once we have initialized the log alpha from the EAP-IG-inputs attribution scores, as defined in Section 3.2, we train each model on its respective task via edge pruning (Bhaskar et al., 2024) for 1,000 training steps, using a batch size of 20 for GPT-2 and 10 for Qwen-2.5. We search for hyperparameters via Bayesian optimization on the validation set, selecting the combination that minimizes the combined objective $P = \mathrm{CMD} - \mathrm{CPR}$. For more details on the final hyperparameters and the optimization procedure, see Appendix A.

## 5 Results

We evaluated our methods on both the private and public test sets of the shared task using the CMD (lower is better) and CPR (higher is better) metrics. Tables 1 to 4 summarize the results.

On both test sets, we observe that our sequential ensemble method with warm-start edge pruning (`s-ens`) already improves the performance across tasks compared to the EAP-IG-inputs baseline, particularly with respect to the CPR metric, achieving a +0.16 average gain. The parallel ensemble (`p-ens`) also provides consistent improvements over the baseline.

Building on these findings, we combined both strategies—integrating warm-start edge pruning with the three EAP-based methods for parallel ensembling—into our final submission, `hybrid-ens`. This combined approach achieves the best overall results on the private set, with the lowest CMD

Table 1: **CMD** scores (lower is better) on the **private** test set, i.e., on the final leaderboard.

| Method | GPT-IOI | Qwen-IOI | Qwen-MCQA | **Average** |
|---|---|---|---|---|
| EAP-IG-inputs | 0.02 | 0.01 | 0.05 | **0.03** |
| `s-ens` | 0.03 | 0.02 | 0.07 | 0.04 |
| `p-ens` | 0.02 | 0.02 | 0.07 | 0.04 |
| `hybrid-ens` | 0.03 | 0.02 | 0.04 | **0.03** |

Table 2: **CPR** scores (higher is better) on the **private** test set, i.e., on the final leaderboard.

| Method | GPT-IOI | Qwen-IOI | Qwen-MCQA | **Average** |
|---|---|---|---|---|
| EAP-IG-inputs | 1.89 | 1.73 | 1.15 | 1.59 |
| `s-ens` | 2.37 | 1.71 | 1.16 | 1.75 |
| `p-ens` | 2.11 | 1.88 | 0.79 | 1.59 |
| `hybrid-ens` | 2.43 | 1.88 | 1.19 | **1.83** |

Table 3: **CMD** scores (lower is better) on the **public** test set. EAP-IG-inputs scores are copied from Mueller et al. (2025). Our own scores should be taken with a grain of salt because we could not precisely reproduce the baselines.

| Method | GPT-IOI | Qwen-IOI | Qwen-MCQA | **Average** |
|---|---|---|---|---|
| EAP-IG-inputs | 0.03 | 0.02 | 0.08 | 0.04 |
| `s-ens` | 0.03 | 0.01 | 0.09 | 0.04 |
| `p-ens` | 0.03 | 0.03 | 0.09 | 0.05 |
| `hybrid-ens` | 0.02 | 0.03 | 0.04 | **0.03** |

Table 4: **CPR** scores (higher is better) on the **public** test set. EAP-IG-inputs scores are copied from Mueller et al. (2025). Our own scores should be taken with a grain of salt because we could not precisely reproduce the baselines.

| Method | GPT-IOI | Qwen-IOI | Qwen-MCQA | **Average** |
|---|---|---|---|---|
| EAP-IG-inputs | 1.85 | 1.63 | 1.16 | 1.55 |
| `s-ens` | 2.22 | 1.62 | 0.99 | 1.61 |
| `p-ens` | 2.06 | 1.83 | 0.99 | 1.63 |
| `hybrid-ens` | 2.23 | 1.83 | 1.22 | **1.76** |

score (0.03 average) and the highest CPR score (1.83 average) on the leaderboard.

## 6 Conclusion

In this work, we introduce our system for the MIB shared task, exploring ensemble strategies for combining various circuit localization methods. We investigate two complementary ensemble strategies: a sequential ensemble with warm-start edge pruning and a parallel ensemble over multiple EAP-based variants, each yielding improvements over the baseline. Combining these strategies into a hybrid ensemble achieves the lowest CMD scores and highest CPR scores on both private and public leaderboards, demonstrating that ensembling complementary circuit localization methods can consistently and robustly enhance performance in mechanistic interpretability benchmarks.

For future work, we plan to investigate more diverse base methods, develop adaptive weighting schemes for ensembling, design improved strategies to integrate edge pruning with attribution-based approaches, and optimize for efficiency to scale to larger models and more complex tasks.

## Limitations

While our hybrid ensemble achieves strong results on the MIB shared task, several limitations remain. Our current ensemble relies on a naïve equal-weight averaging of the base methods, which may not fully exploit their complementary strengths; more adaptive weighting strategies could potentially yield further improvements. Additionally, our experiments are restricted to the task–model combinations defined in the shared task, and it is unclear how well the ensemble strategies generalize to other interpretability benchmarks or model architectures. Finally, the integration of edge pruning with attribution-based approaches is still heuristic, leaving room for more principled formulations that could improve both interpretability and performance.

## Acknowledgments

We would like to thank Lea Hirlimann and Robert Litschko for their input and feedback throughout the project. We further acknowledge the support for BP through the ERC Consolidator Grant

DIALECT 101043235 and for HS through the Deutsche Forschungsgemeinschaft (project SCHU 2246/14-1).

## References

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems*, volume 37, pages 18506–18534. Curran Associates, Inc.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale. pages 17432–17445, Miami, Florida, USA.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. 2021. Causal analysis of syntactic agreement mechanisms in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1828–1843, Online. Association for Computational Linguistics.

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*.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried 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*.

Neel Nanda. 2023. Attribution Patching: Activation Patching At Industrial Scale.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org.

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.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*.

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*.

Sarah Wiegreffe, Oyvind Tafjord, Yonatan Belinkov, Hannaneh Hajishirzi, and Ashish Sabharwal. 2025. Answer, assemble, ace: Understanding how LMs answer multiple choice questions. In *The Thirteenth International Conference on Learning Representations*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

## A Hyperparameters

In Table 5, we report the final hyperparameters for each model and task, as described in Section Section 2.1. These were selected based on a hyperparameter search via Bayesian optimization over the following hyperparameters: edge learning rate $\in \{0.03, 0.4, 0.8\}$, layer learning rate $\in \{0.001, 0.4, 0.8\}$, regularization edge learning rate $\in \{0.03, 0.4, 0.8\}$, regularization layer learning rate $\in \{0.001, 0.4, 0.8\}$, start edge sparsity $\in \{0.8, 0.9, 0.95\}$, target edge sparsity $\in \{0.975, 0.99, 1.05\}$, target layer sparsity $\in \{0.4, 0.69\}$, warmup steps $\in \{50, 250, 500\}$, disable node loss $\in \{\texttt{true}, \texttt{false}\}$, signs from EAP-IG $\in \{\texttt{true}, \texttt{false}\}$.

## B Experiments with IFR

We also experimented with information flow routes, or IFR (Ferrando and Voita, 2024), as a base method. The results were clearly weaker than the EAP-IG-inputs baseline, so we did not include this in our final submissions.

We need a special approach to compare edge scores from IFR with other base methods. This

| Parameter | GPT-2 IOI | Qwen-2.5 IOI | Qwen-2.5 MCQA |
|---|---|---|---|
| Train steps | 1000 | 1000 | 1000 |
| Batch | 20 | 10 | 10 |
| Warmup type | `linear` | `linear` | `linear` |
| Warmup steps | 250 | 250 | 250 |
| Disable node loss | ✓ | × | ✓ |
| Edge LR | 0.4 | 0.8 | 0.4 |
| Layer LR | 0.03 | 0.8 | 0.4 |
| Reg edge LR | 0.8 | 0.8 | 0.4 |
| Reg layer LR | 0.001 | 0.4 | 0.8 |
| Start edge spars. | 0.90 | 0.95 | 0.95 |
| Target edge spars. | 0.975 | 0.975 | 1.05 |
| Start layer spars. | 0.0 | 0.0 | 0.0 |
| Target layer spars. | 0.69 | 0.69 | 0.69 |
| Attribution / Signs | z-score attribution | signs from EAP-IG | signs from EAP-IG |

Table 5: Final hyperparameters for the three runs. Columns are model–task combinations; rows are parameters. A checkmark (✓) indicates a flag is enabled; a cross (×) indicates disabled.

is because IFR is designed such that the scores of incoming edges to a given node sum to 1. We therefore apply the same normalization to the scores from other methods as well, and proceed from there. This also means that, when computing actual circuits from the final scores, we need greedy search, as already described by Mueller et al. (2025).