Layer-wise Target Propagation: Efficient Component Attribution through Target Centric Propagation

Lasse Jantsch* **Dong-Jae Koh*** **Seonghyeon Lee†** **Young-Kyoon Suh†**
School of Computer Science and Engineering, Kyungpook National University, South Korea
{lassejantsch, djkoh, sh0416, yksuh}@knu.ac.kr

Abstract

Understanding the internal mechanisms of transformer-based large language models (LLMs) is crucial for their reliable deployment and effective operation. While recent efforts have yielded a plethora of attribution methods attempting to balance faithfulness and computational efficiency, dense component attribution remains prohibitively expensive. In this work, we introduce *Layer-wise Target Propagation* (LTP), a novel framework that faithfully traces information flow on the frozen transformer in one forward and one backward pass without requiring counterfactual examples. LTP analytically decomposes and linearizes the computational structure of the Transformers into distinct pathways along which it propagates a targeted unembedding vector to receive the effective representation at each residual position. This target-centric propagation achieves $O(1)$ time complexity with respect to the number of model components, scaling to long input sequences and dense component attribution. Extensive experiments on standard interpretability benchmarks demonstrate that LTP achieves state-of-the-art faithfulness and unprecedented efficiency compared to existing baselines.

1 Introduction

Transformer-based large language models (LLMs) are the dominant model paradigm in AI. However, their internal mechanisms remain poorly understood. A central challenge lies in *attribution* (Sundararajan et al., 2017), identifying which components are responsible for a model's predictions.

While a plethora of attribution approaches have been applied to LLMs (Ferrando et al., 2024), these methods often fall short in real-world settings. Along with this research line, fully decomposing the computational graph of the transformers, i.e., *decomposition-based approaches*, becomes a promising research direction in the attribution research realm (Hong et al., 2025).

Motivated by the limited efficiency and inaccurate approximations in previous work, we propose *Layer-wise Target Propagation* (LTP), a novel decomposition-based framework for efficient input and component attribution. Specifically, LTP efficiently computes the attribution through a single forward and backward pass by propagating the unembedding vector of the target token. As our method leverages general forward and backward passes, it can be universally applied to *any* transformer variant. In this paper, we focus on the SwiGLU transformer, as most modern competitive LLMs adopt this architecture (Touvron et al., 2023). Figure 1 shows an overview of our approach.

Our experiments demonstrate the superior faithfulness and efficiency of LTP on standard tasks for both input and component attribution. Our contributions are summarized as follows:

- We introduce *Layer-wise Target Propagation* (LTP), a novel decomposition-based framework that efficiently assigns logit scores in a single forward and backward pass, without requiring counterfactual examples.
- By leveraging the bilinear structure of the SwiGLU Transformers, we show that the computational structure is decomposable into distinct control and content pathways.
- Through extensive experiments on standard interpretability tasks, we demonstrate that LTP achieves superior faithfulness and efficiency compared to state-of-the-art baselines, effectively identifying causal mechanisms.

All artifacts are publicly available on GitHub.[1]

* Equal Contributions
† Corresponding authors

[1] https://github.com/lab-paper-code/LTP

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 60755–60784
October 24-29, 2026 ©2026 Association for Computational Linguistics

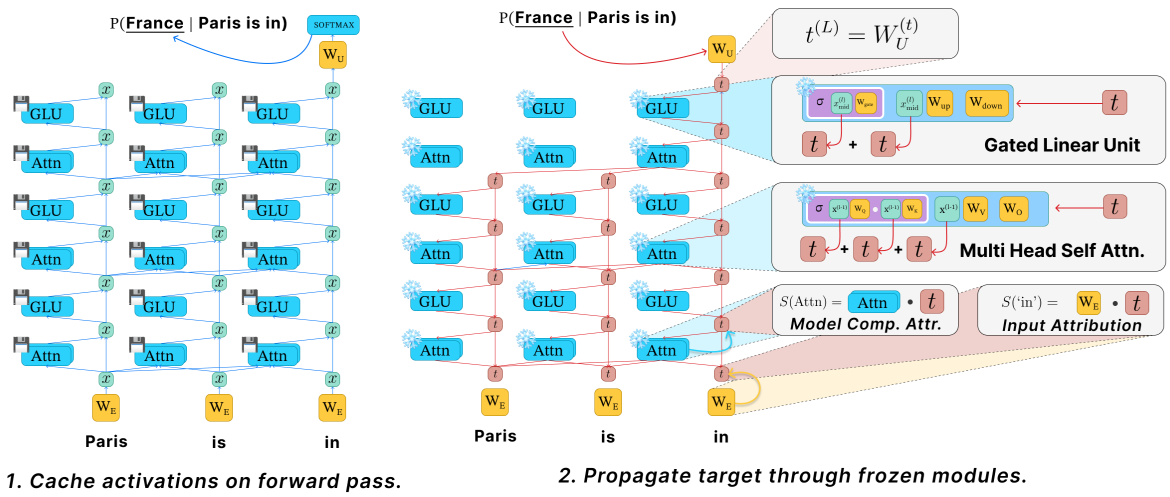

Figure 1: Overview of the *Layer-wise Target Propagation* (LTP) framework for efficient input and component attribution. *LTP* operates in two stages: (1) *One-pass forward execution*, in which the model processes the input once while caching activations required for the back propagation; and (2) *top-down target propagation*, where the unembedding vector of the targeted token is recursively propagated backward through Transformer modules to identify an effective target at each residual position.

2 Related Work

Attribution research aims to localize the sources of model behavior by identifying responsible inputs or internal components (Ferrando et al., 2024).

Input attribution. A first family of input attribution methods derives importance from model gradients and their propagated variants. They treat the first-order Taylor expansion as a sensitivity signal (Simonyan et al., 2013; Denil et al., 2014; Li et al., 2015), as in *Input x Gradient* (Denil et al., 2014), which scales the gradient by the embedding to account for input magnitude. As raw gradients suffer from saturation and shattering (Shrikumar et al., 2019; Balduzzi et al., 2018), later work denoises the gradients through *Integrated Gradients* (Sundararajan et al., 2017) or *SmoothGrad* (Smilkov et al., 2017).

A second family traces how the signal flows through attention. Raw attention weights were the earliest such signal, but proved unreliable as indicators (Jain and Wallace, 2019). Subsequent work made the signal more faithful by weighting attention with value-vector norms (Kobayashi et al., 2020, 2021) and composing it across layers via rollout (Abnar and Zuidema, 2020; Ferrando et al., 2022; Modarressi et al., 2022). Building on this trajectory, decomposition-based methods analyze the model's linear structure, yielding faithful attribution of attention-mediated effects (Modarressi et al., 2023; Kobayashi et al., 2023; Hong et al., 2025). Our method adopts this decomposition view, but rather than propagating the decomposed states forward, it propagates the target backward through the linearized structure.

Component-level attribution. Attribution can also target internal components rather than inputs. *Direct Logit Attribution* (DLA) (Elhage et al., 2021; nostalgebraist, 2020) exploits the additive residual stream to read off a component's *direct* contribution to the output logit and applies to both attention heads (Ferrando et al., 2023) and neurons (Geva et al., 2022). DLA, however, does not capture the indirect effects of downstream layers, thereby reducing interpretability in early layers.

Activation patching removes this blind spot by overwriting a component and measuring the resulting output change, thereby capturing both direct and indirect effects through counterfactual forward passes (Vig et al., 2020; Meng et al., 2022; Wang et al., 2022; Zhang and Nanda, 2023). This faithfulness comes at a steep cost due to the separate forward pass per component. Our method targets exactly this gap, recovering the direct and indirect effects that DLA misses through a single backward pass, without the per-component forward passes that make patching intractable.

These component-level signals, in turn, are the building blocks of automated circuit discovery, which searches for the component subsets responsible for a behavior. Existing pipelines instantiate

this search on top of causal intervention (Conmy et al., 2023), gradients (Nanda, 2023; Syed et al., 2024), LRP (Jafari et al., 2025), and contextual mixing (Ferrando and Voita, 2024).

3 Background

In this section, we review the recent variants of Transformer components and establish the notation used throughout the paper. A more general overview is provided in Appendix A.

3.1 Multi-Head Self-Attention (MHSA)

Different from the original MHSA module (Radford et al., 2019), recent variants such as the LLaMA2 architecture (Touvron et al., 2023; Grattafiori et al., 2024) employ different normalization and positional encoding techniques.

First, given an input embedding sequence $\mathbf{X} \in \mathbb{R}^{T\times d}$, the modern MHSA module first normalizes the embedding sequence using RMSNorm (Zhang and Sennrich, 2019). These normalized embeddings $\tilde{\mathbf{X}}$ are projected into queries, keys, and values across different heads:

$$\tilde{\mathbf{X}}\mathbf{W}_Q^{(h)}, \tilde{\mathbf{X}}\mathbf{W}_K^{(h)}, \tilde{\mathbf{X}}\mathbf{W}_V^{(h)}$$

where $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)}, \mathbf{W}_V^{(h)}$ are the query, key, and value projections for head h. The attention scores are computed using the scaled dot product with the query and key, followed by the softmax function:

$$\tilde{\alpha}^{(h)} = \mathrm{SoftMax}\left(\frac{R_Q\left(\tilde{\mathbf{X}}\mathbf{W}_Q^{(h)}\right) R_K\left(\tilde{\mathbf{X}}\mathbf{W}_K^{(h)}\right)^\top}{\sqrt{d_h}}\right),$$

where $R(\cdot)$ denotes the application of RoPE (Su et al., 2021). These attention scores are used for the weighted sum of value vectors V, and the output of MHSA is formed by concatenating head-wise outputs and projecting them to the embedding dimension:

$$\mathrm{MHSA}(\mathbf{X}) = \mathrm{Concat}\big(\tilde{\mathbf{X}}\mathbf{W}_V^{(i)}\big)\mathbf{W}_O,$$

where $\mathbf{W}_O$ is the projection matrix. The output of the MHSA module is added to the residual stream.

3.2 Feed-forward Network (FFN)

In modern architecture, the FFN uses the SwiGLU activation, which separates computation into a gating and a value pathway. Given a normalized input $\tilde{\mathbf{X}}$, GLU activation is defined as:

$$\mathrm{GLU}(\mathbf{X}) = \Big(\sigma(\tilde{\mathbf{X}}\mathbf{W}_G) \odot (\tilde{\mathbf{X}}\mathbf{W}_U)\Big)\mathbf{W}_D,$$

where $\sigma(\cdot)$ denotes the SiLU (Swish) activation, $\odot$ is element-wise multiplication, and $\mathbf{W}_G, \mathbf{W}_U \in \mathbb{R}^{d\times d_{\text{ffn}}}$ and $\mathbf{W}_D \in \mathbb{R}^{d_{\text{ffn}}\times d}$ are the gate, up, and down projection matrices, respectively.

3.3 Decomposition of Transformer Modules

One interesting finding is the linear separability of the transformer blocks (Modarressi et al., 2023; Kobayashi et al., 2021), allowing for decomposition into independent model component contributions; attention heads and neurons for MHSA and GLU, respectively. For MHSA, the output can be expressed as the sum of head-wise contributions:

$$\mathrm{MHSA}(\mathbf{X}) = \sum_{h=1}^{H} \alpha^{(h)}\big(\tilde{\mathbf{X}}\mathbf{W}_V^{(h)}\big)\mathbf{W}_O^{(h)},$$

where $\mathbf{W}_V^{(h)}$ and $\mathbf{W}_O^{(h)}$ denote the value and output projections of head h. Similarly, the GLU output can be decomposed into contributions from individual neurons:

$$\mathrm{GLU}(\mathbf{X}) = \sum_{i=1}^{d_{\text{ffn}}} \Big(\sigma(\tilde{\mathbf{X}}\mathbf{w}_G^{(i)}) \odot (\tilde{\mathbf{X}}\mathbf{w}_U^{(i)})\Big)\mathbf{w}_D^{(i)},$$

where $\mathbf{w}_G^{(i)}$ and $\mathbf{w}_U^{(i)}$ denote the i-th columns of $\mathbf{W}_G$ and $\mathbf{W}_U$, and $\mathbf{w}_D^{(i)}$ denotes the i-th row of $\mathbf{W}_D$. Each term corresponds to the contribution of a single GLU neuron to the residual stream.

4 Layer-wise Target Propagation

In this section, we formalize the dense component attribution problem and propose *Layer-wise Target Propagation* (LTP) as our solution.

4.1 Problem Statement

Our objective is to *quantify how much a given model component* $\mathbf{x}$ *contributes to a target model output* $\mathbf{t}$. Formally, we seek to assign an attribution score $S_t(\mathbf{x})$ that measures the relevance of $\mathbf{x}$ to a target token prediction $\mathbf{t}$. Let $\mathbf{x}_i^{(0)} \in \mathbb{R}^d$ denote the embedding vector at token position i, and let $\mathbf{t}_j \in \mathbb{R}^d$ be the j-th row of the unembedding matrix W_{UE}, corresponding to the direction in latent space that increases the logit $logit_i^{(j)}$ of token $t^{(j)} \in \mathcal{V}$ at position i. Owing to the additive structure of the residual stream, $\mathbf{x}_i^{(0)}$ contributes directly to the output logit, yielding the direct attribution score:

$$S_{t,\text{direct}}(\mathbf{x}_i^{(0)}) = (\mathbf{x}_i^{(0)})^\top \mathbf{t}_j.$$

While each embedding at position i' contributes directly to the logit at its own position, it has no direct contribution to the logits at later positions $i > i'$; such cross-position influence is mediated entirely through downstream layers (Belrose et al., 2023). The full attribution can be formalized as:

$$S_t(\mathbf{X}^{(0)}) = f(\mathbf{x}_i^{(0)})^\top \mathbf{t}_j,$$

where $f(\cdot)$ denotes the downstream processing of $\mathbf{x}_i^{(0)}$ with an otherwise frozen network. While faithful (Hong et al., 2025), this approach requires a separate forward pass per component to evaluate $f(\mathbf{x}_i^{(0)})$, making dense attribution prohibitively expensive.

4.2 Target Propagation Framework

To overcome this limitation, we invert the attribution perspective: rather than passing each component forward, we propagate the target backward. Specifically, we reformulate the attribution score as:

$$S_t(\mathbf{x}_i^{(0)}) = (\mathbf{x}_i^{(0)})^\top f^{-1}(\mathbf{t}_j) = (\mathbf{x}_i^{(0)})^\top \mathbf{t}_j^{(0)},$$

where $\mathbf{t}_j^{(0)}$ denotes the *effective target* at layer 0. This effective target is computed recursively via:

$$\begin{aligned} \mathbf{t}_{\text{mid}}^{(l)} &= \mathbf{t}^{(l+1)} + g_{\text{GLU}}^{(l)}\big(\mathbf{t}^{(l+1)}\big), \\ \mathbf{t}_j^{(l)} &= \mathbf{t}_{\text{mid}}^{(l)} + g_{\text{MHSA}}^{(l)}\big(\mathbf{t}_{\text{mid}}^{(l)}\big), \end{aligned}$$

where $g_{\text{GLU}}^{(l)}(\cdot)$ and $g_{\text{MHSA}}^{(l)}(\cdot)$ denote linearized inverse transformations of the $\text{GLU}^{(l)}$ and $\text{MHSA}^{(l)}$ blocks, respectively. Under this formulation, attribution requires only a single forward pass to cache activations and a single backward pass to propagate the effective target.

LTP implements this framework by linearizing and inverting MHSA and GLU via their constituent pathways while keeping the rest of the module frozen. To prevent the explosion of the target norm and sustain completeness[2], each inversion is further regularized by a weighting parameter μ. The following sections instantiate $g_{\text{GLU}}^{(l)}(\cdot)$ and $g_{\text{MHSA}}^{(l)}(\cdot)$ at the neuron and attention-head level. Detailed derivations and their complexity analyses are provided in Appendix D and B, respectively.

[2]While LTP generally supports completeness, our choice of softmax linearization breaks it for the sake of a robust gradient.

4.3 Neuron Target Propagation

Consider a GLU neuron n with activation $\alpha^{(n)}$. To propagate the effective target backward through the GLU, we approximate the inverse transformation $\mathbf{g}_{\text{GLU}}^{(n)}(\cdot)$ as a weighted combination of the up- and gate-projection paths, assuming frozen activations:

$$\mathbf{g}_{\text{GLU}}^{(n)}(\mathbf{t}) = \mu_{\text{up}} \cdot \mathbf{g}_{\text{up}}^{(n)}(\mathbf{t}) + \mu_{\text{gate}} \cdot \mathbf{g}_{\text{gate}}^{(n)}(\mathbf{t}),$$

where $\mu_{\text{up}} + \mu_{\text{gate}} = 1$.

Up-projection. With frozen activations, the up-projection-pathway is linear and corresponds to the composition of the up- and down-projection weights, $\mathbf{W}_{UD}^{(n)} = \mathbf{w}_U^{(n)} \mathbf{w}_D^{(n)}$. The propagated target is given by:

$$\mathbf{g}_{\text{up}}^{(n)}(\mathbf{t}) = \underbrace{\frac{\mathbf{w}_U^{(n)} \odot \boldsymbol{\gamma}}{\sigma}}_{\text{effective weight } \tilde{\mathbf{w}}_U^{(n)}} \cdot \alpha^{(n)} \cdot \underbrace{\big((\mathbf{w}_D^{(n)})^\top \mathbf{t}\big)}_{\text{gradient scalar } \lambda_{\text{GLU}}},$$

where $\boldsymbol{\gamma}$ and σ denote the RMSNorm parameters. For clarity, we refer to such normalized weights as *effective weights* $\tilde{\mathbf{w}}$.

Gate-projection. Due to nonlinearity, the gate-projection pathway must be linearized to be inverted. We approximate the local gradient using the ratio $\alpha^{(n)}/s^{(n)}$, where $s^{(n)}$ denotes the pre-activation value:

$$\mathbf{g}_{\text{gate}}^{(n)}(\mathbf{t}) = \tilde{\mathbf{w}}_G^{(n)} \cdot \frac{\alpha^{(n)}}{s^{(n)}} \cdot \lambda_{\text{GLU}}.$$

4.4 Attention Head Target Propagation

For an attention head h, target propagation is more involved due to the token mixing softmax activation. We decompose the inverse MHSA transformation into contributions from the query, key, and value paths and combine them via weighted aggregation:

$$\mathbf{g}_{\text{head}}^{(h)}(\mathbf{t}) = \mu_Q \cdot \mathbf{g}_Q^{(h)}(\mathbf{t}) + \mu_K \cdot \mathbf{g}_K^{(h)}(\mathbf{t}) + \mu_V \cdot \mathbf{g}_V^{(h)}(\mathbf{t}),$$

where $\mu_Q + \mu_K + \mu_V = 1$.

Value path. The value path is linear, given fixed attention weights, and is defined as:

$$\mathbf{g}_V^{(h)}(\mathbf{t}) = \tilde{\mathbf{W}}_V^{(h)} \cdot \underbrace{(\mathbf{W}_O^{(h)})^\top \mathbf{t}}_{\text{grad. vec. } \boldsymbol{\lambda}_{\text{MHSA}}} \cdot \boldsymbol{\alpha}^{(h)}.$$

Query and key paths. Propagating targets through the query and key paths requires tracing gradients through the softmax operation. Unlike GLU activations, softmax outputs depend jointly on all attention scores and remain non-zero even for zero inputs, making naive linearization unstable. Following Achtibat et al. (2024), we adopt a Taylor-based decomposition of the softmax, which provides a stable—though non-conservative-approximation:

$$\mathbf{g}_{Q,i}^{(h)}(\mathbf{t}) = \tilde{\mathbf{W}}_Q^{(h)} \sum_{j \leq i} \delta_{ij} \cdot R_{Q,i}^{-1} \left(\frac{R_{K,j}(\tilde{\mathbf{x}}_j \mathbf{W}_K^{(h)})}{\sqrt{d_h}} \right)^\top,$$

$$\mathbf{g}_{K,j}^{(h)}(\mathbf{t}) = \tilde{\mathbf{W}}_K^{(h)} \sum_{i \geq j} \delta_{ij} \cdot R_{K,j}^{-1} \left(\frac{R_{Q,i}(\tilde{\mathbf{x}}_i \mathbf{W}_Q^{(h)})}{\sqrt{d_h}} \right)^\top,$$

where the scalar interaction term δ_{ij} captures the sensitivity of the softmax output:

$$\delta_{ij} = \alpha_{ij} \big(\tilde{\mathbf{x}}_j \mathbf{W}_V^{(h)} - \boldsymbol{\mu}_i^{(h)} \big) \bullet \boldsymbol{\lambda}_{\text{MHSA}}.$$

Here, $\boldsymbol{\mu}_i^{(h)}$ denotes the expected intermediate activation vector of head h at token position i.

5 Experiments

We assess the effectiveness of LTP under two research questions:

- RQ1. How *faithfully* does LTP attribute input tokens and internal component importance?
- RQ2. How *effective* are balanced hyperparameter selections for gaining optimal performance?

To answer these questions, we conduct attribution *faithfulness* and *sensitivity* experiments.

5.1 Attribution Faithfulness

We quantify how well LTP identifies components that are causally relevant to a model's prediction.

5.1.1 Experimental setup

Datasets and Models. We evaluate LTP on several modern LLM checkpoints, including Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Llama-2-7B-Chat (Touvron et al., 2023), Qwen3-4B-Instruct (Yang et al., 2025), Mistral-7B-Instruct (Jiang et al., 2023), Qwen2.5-32B-Instruct (Yang et al., 2024), and GPT-2 Large (Radford et al., 2019). Three representative datasets are used to cover different attribution scenarios: (1) *Known 1000* (Meng et al., 2022) for factual sequence completion, (2) *SQuAD v2.0* (Rajpurkar et al., 2018) for context-supported question answering, and (3) *IMDb* (Maas et al., 2011) for sentiment analysis. We also leverage the Indirect Object Identification (IOI) task (Wang et al., 2022), a widely used benchmark for circuit discovery in the component attribution experiment. See Appendix E for additional details.

Metrics. Following prior work, we adopt two complementary metrics: *disruption* and *recovery*. Both replace the embeddings of a selected set of tokens with zero vectors and measure the resulting change in the probability of the correct next token. For *disruption*, we zero out the top-K tokens judged most important and keep the rest intact; a faithful attribution should then cause a large drop in the probability, since the tokens driving the prediction have been removed. For *recovery*, we instead zero out all tokens *except* the top-K most important; here, a faithful attribution should preserve the probability since the retained tokens alone should suffice. These mirror the comprehensiveness and sufficiency criteria of DeYoung et al. (2020).

5.1.2 Input Attribution Experiments

Baselines. We compare LTP against a broad range of attribution baselines, including gradient-based methods (*Gradient* at input, *Input×Gradient* (Denil et al., 2014), and *Integrated Gradients* (Sundararajan et al., 2017), attention-based methods (*Last-Layer Attention* and *Mean Attention* on the last token, as well as *Attention Rollout* (Abnar and Zuidema, 2020), the context-mixing method *Information Flow Routes* (Ferrando and Voita, 2024), and the decomposition-based method *DePass* (Hong et al., 2025).

Implementation details. For the Known 1000 dataset, we assess faithfulness by ablating individual input tokens, excluding special tokens. For the SQuAD v2.0 and IMDb datasets, we employ a context- or review-level ablation strategy, in which tokens within the supporting paragraph or review are ablated while the remaining prompt remains intact. This setup mitigates artifacts caused by truncating incomplete sentences immediately preceding the target token, yielding a cleaner faithfulness signal. Additional setup information for reproducibility is located in Appendix F.

For token-level attribution, we compute effective targets $\mathbf{t}_i^{(0)}$ by propagating the unembedding vector of the correct answer token backward to the input

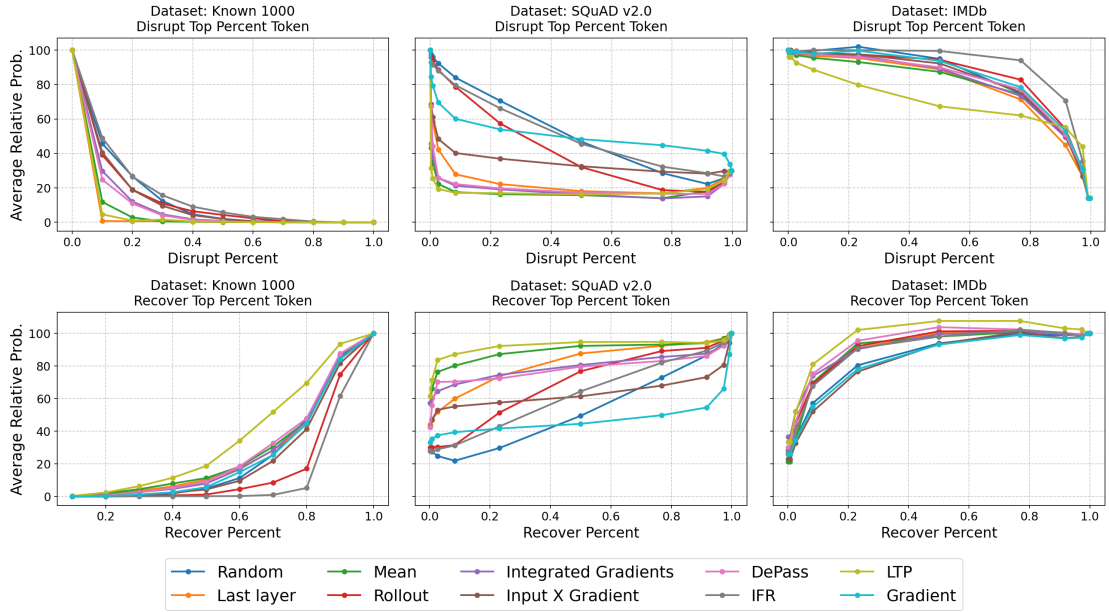

Figure 2: Performance comparison with baseline attribution methods on **Llama-3.1-8B-Instruct** under different top input token masking strategies. Our approach shows *lower* Top-k disruption and *higher* Top-k recovery, indicating more accurate identification of important components.

Methods	**Known 1000**			**SQuAD v2.0**			**IMDb**		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Random	14.22	22.78	8.56	50.64	52.39	1.75	84.44	85.93	1.49
Attention-based									
Last layer	**5.26**	24.83†	19.57†	21.96	82.04	60.08	79.68†	91.57	11.89†
Mean	6.57	25.78	19.21	**17.06**	89.46†	72.40†	79.89	91.42	11.53
Rollout	13.26	15.72	2.46	40.58	69.43	28.85	85.33	92.47	7.14
Gradient-based									
Gradient	12.14	22.94	10.80	49.95	46.68	-3.27	84.52	84.58	0.06
Input X Gradient	12.58	21.27	8.69	34.07	63.22	29.15	82.65	84.34	1.69
Integrated Gradients	9.92	24.49	14.57	17.92	79.53	61.61	81.29	92.41†	11.12
Contextual Mixing-based									
IFR	16.15	11.88	-4.27	50.16	62.51	12.35	91.63	91.12	-0.51
Decomposition-based									
DePass	9.17	25.45	16.28	19.29	78.45	59.16	85.49	69.65	-15.84
LTP (Ours)	5.79†	**33.88**	**28.09**	17.73†	**92.51**	**74.78**	**69.34**	**99.51**	**30.17**

Table 1: Faithfulness evaluation of crucial token attribution on **Llama-3.1-8B-Instruct** across Known 1000 (factual knowledge), SQuAD v2.0 (reading comprehension), and IMDb (sentiment analysis). Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful token localization; *total* denotes their aggregate AUC score. **Bold** indicates the best result, and † the second best. LTP consistently achieves the highest faithfulness across datasets.

layer. Token importance is then obtained via the dot product between $\mathbf{t}_i^{(0)}$ and the corresponding embedding vectors.

Results. As shown in Figure 2 and Table 1, LTP attains the highest total faithfulness across all three datasets. LTP is the only method that remains *faithful* across two qualitatively different regimes. On *localized* tasks, where the prediction hinges on a single answer-bearing span (Known 1000, SQuAD v2.0), several baselines are competitive. On the *distributed* IMDb task, where sentiment is spread across many tokens, every competing method degrades sharply while LTP retains a large lead. We attribute this robustness to how LTP decomposes the computation: rather than scoring tokens by attention routing alone, it propagates the target through both the *content* pathway (value

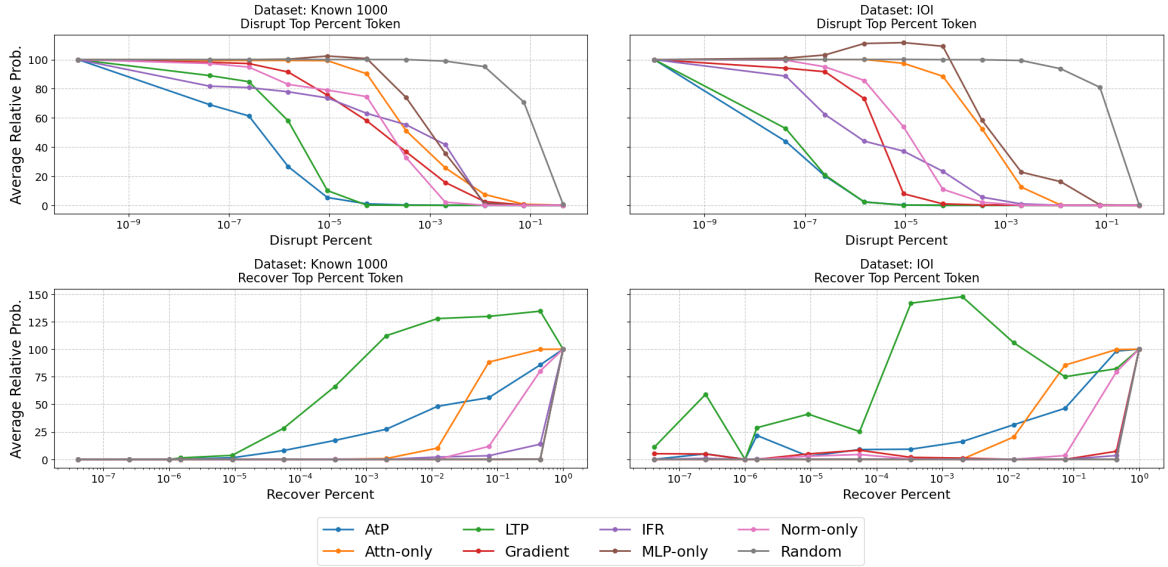

Figure 3: Performance comparison with baseline attribution methods on **Llama-3.1-8B-Instruct** under different top model component masking strategies. Our approach shows lower disruption and higher recovery, indicating more accurate identification of important components.

and up projections) and the *control* pathway (attention and gate scores), so that evidence dispersed across many tokens is still accumulated rather than lost. Finally, extended results on Llama-2, Qwen3, Qwen2.5, Mistral, and GPT-2 Large (Appendix I) reproduce the same ordering, confirming that the gains are not specific to a single backbone.

5.1.3 Component Attribution Experiment

We further assess attribution faithfulness at the level of internal model components, focusing on individual attention heads and MLP neurons.

Baselines. We compare LTP with activation-based baselines: *Attn-only*, which ranks attention heads by activation magnitude; *MLP-only*, which ranks neurons by activation strength; and *Norm*, which decomposes component contributions using the L2 norm. We also evaluate gradient-based methods, including *Gradient*, *AtP* (Nanda, 2023), and contextual mixing-based method, *Information Flow Routes* (IFR) (Ferrando and Voita, 2024).

Evaluation Setup. All attention heads and MLP neurons are ranked according to their attribution scores. We progressively ablate or recover the top-ranked components and measure the resulting change in the model's prediction confidence. Additional setup information for reproducibility is located in Appendix F.

Results. As shown in Figure 3 and Table 2, LTP achieves the highest total faithfulness on both IOI and Known 1000. Although Attn-only obtains slightly higher aggregate AUC on IOI, LTP's S-shaped curve isolates the circuit much earlier (around 0.001%), raising the target probability above 140% before declining as opposing nodes are reintroduced. This dip, which reduces LTP's AUC, arises from (i) attributing logits rather than post-softmax probabilities, thereby missing gains from suppressed competing logits, and (ii) approximation errors in IOI's highly specialized circuit. The sharp early recovery nevertheless highlights LTP's precision in identifying causal mechanisms.

Methods	Known 1000			IOI		
	dis.↓	rec.↑	tot.↑	dis.↓	rec.↑	tot.↑
Random	19.77	27.55	7.78	21.85	27.54	5.69
Baselines						
Attn-only	0.66	93.43†	92.77†	0.15	**93.07**	**92.92**
MLP-only	0.37	27.58	27.21	0.87	27.53	26.66
Norm-only	0.06†	67.25	67.19	0.01†	65.02	65.01
Gradient-based						
Gradient	0.28	27.62	27.34	**0.00**	30.95	30.95
AtP	**0.00**	81.39	81.39	**0.00**	84.39	84.39
Contextual Mixing-based						
IFR	0.32	34.72	34.40	0.02	29.15	29.13
Decomposition-based						
LTP (Ours)	**0.00**	**123.50**	**123.50**	**0.00**	86.87†	86.87†

Table 2: Component-level attribution performance on **Llama-3.1-8B-Instruct**. Lower disruption and higher recovery indicate more faithful localization. Bold denotes best; † denotes second best.

Extended results for generalizability are provided in Appendix I.

5.1.4 Computational Efficiency

Beyond faithfulness, LTP is also substantially more computationally efficient in practice. As reported in Appendix B and Table 5, LTP completes attribution on Llama-3.1-8B-Instruct in 4.98 seconds on Known 1000, compared to 14.88 and 14.91 seconds for Activation Patching and DePass. The gap becomes much larger in the longer-context SQuAD v2.0 setting: LTP requires only 129.05 seconds, whereas Activation Patching and DePass require 5370.65 and 5452.28 seconds, respectively, yielding over a 40× speedup while maintaining comparable peak memory footprints. These empirical measurements show that LTP's faithfulness gains come with strong practical scalability.

5.2 Sensitivity Analysis

In the sensitivity analysis, we evaluate how the attribution performance changes as we alter the weighting parameters μ, which balance the contributions of different pathways. Table 3 shows the configuration weights. Despite the Query-Key sensitivity analysis, a higher p indicates a larger reliance on the content pathways.

- **Control-Content Balance.** Varying p shifts emphasis from pure content ($p \approx 1$) to pure control ($p \approx 0$), exposing how content–routing interactions affect faithfulness.
- **Query-Key Balance.** Within the attention control path, we fix $\mu_v = 0.5$ and redistribute the remaining weight between μ_q and μ_k to localize sensitivity to the 'request' (query) versus 'source' (key) of information.
- **Attention Balance.** With MLP weights fixed, we test whether attention-mediated effects are dominated by content being moved (μ_v) or routing logic (μ_q, μ_k).
- **MLP Balance.** We focus on the Gated Linear Unit by fixing the attention mechanism to the balanced configuration and shifting the weights between μ_{gate} and μ_{up} to probe the gate's selective filtering behavior.

Results. Figure 4 summarizes the sensitivity results, with complementary qualitative examples deferred to Appendix G. Input attribution peaks in the mid-range (50–70%), indicating that

Sensitivity	μ_q	μ_k	μ_v	μ_{gate}	μ_{up}
Control-Content	$\frac{1-p}{2}$	$\frac{1-p}{2}$	p	$1-p$	p
Attention	$\frac{1-p}{2}$	$\frac{1-p}{2}$	p	0.5	0.5
Query-Key	$\frac{p}{2}$	$\frac{1-p}{2}$	0.5	0.5	0.5
MLP	0.25	0.25	0.5	$1-p$	p

Table 3: Hyperparameter configuration for the sensitivity analysis. $p \in [0, 1]$ is the sensitivity variable in our experiment.

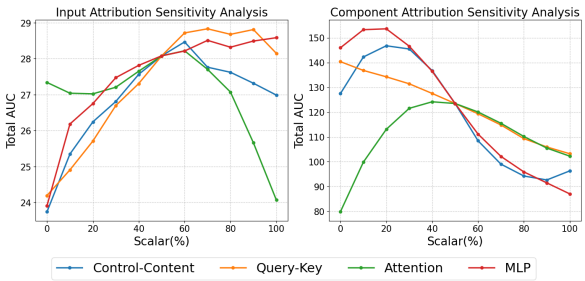

Figure 4: Sensitivity analysis of LTP scaling configurations on the Known 1000 dataset. We evaluate the impact of varying attribution parameters to prioritize different information pathways.

macro-level token importance favors direct content pathways (e.g., value and up projections). In contrast, component attribution peaks at much smaller scales ($p \approx 0.2$) before a sharp decline, revealing that fine-grained fidelity hinges on control/routing pathways (gates and queries/keys), with $1-p \approx 0.8$ needed to capture individual functional roles. Considering both, $p = 0.5$ provides the most stable trade-off by unifying the control and content pathways with equal weighting, which we use throughout the experiments in this paper. An extended sensitivity analysis on fundamentally different tasks and granularities, which further supports $p = 0.5$ as a robust task-agnostic default, is provided in Appendix H. We leave broader configuration sweeps to future work.

6 Discussion

In this section, we discuss the scope and reliability of LTP beyond the main results. We first examine the implications of attributing logits rather than post-softmax probabilities. We then assess the causal validity of LTP by comparing its rankings against exact activation patching. Finally, we quantify the deviation from conservation introduced by the softmax linearizations.

Logit-level attribution. LTP attributes logits rather than post-softmax probabilities, as the final

softmax poses a unique challenge: (1) the Deep Taylor Decomposition (Montavon et al., 2015) distributes relevance via the gradient Jacobian, which vanishes for highly confident next-token predictions, and (2) the error introduced through the final softmax propagates through the entire network and is not bypassed by any linear connection, as is the case with the value projection in the attention block. By stopping at the logit level, however, the framework structurally misses suppressive mechanisms such as the negative name mover heads in IOI (Wang et al., 2022). Achtibat et al. (2024) suggest that these issues could be somewhat alleviated through temperature scaling; in line with their work, we defer the investigation of probability attribution to future work. Alternatively, inhibition circuits could be incorporated without propagating through the final softmax by using the logit residual as the target and attributing it to the component activations.

Agreement with activation patching. Our faithfulness evaluation relies on zero ablation to maintain comparability with previous relevant work, although such ablations may introduce out-of-distribution artifacts. To address this concern, we directly compare the LTP ranking with the causal effects obtained by exact activation patching on a component and sample subset. On IOI (Llama-3.1-8B-Instruct; 200 samples), we score all 1,024 attention heads with both LTP and exact activation patching. The two methods agree strongly: Pearson $r = 0.80$, a 70% overlap among the top-20 heads, and Spearman $\rho = 0.905$ with 100% sign agreement among the causally effective heads. The more moderate full-population rank correlation ($\rho = 0.39$) arises from the weaker alignment of the $\sim$900 of the 1,024 heads with negligible causal effect, which are therefore more prone to order variations due to minimal noise.

Conservation in practice. LTP does not take conservation as a foundational axiom; rather, it is a property that comes with the approach. The choice of the softmax linearization breaks conservation through its implicit bias term, which consumes a small proportion of the propagated relevance. Empirical measurements on 200 examples of the Known 1000 dataset show that the biases in each layer account for $\sim$2.3% of the consumption, where the decrease is not monotone but slightly concave, with a higher loss in late and a lower loss in early layers. At the input, LTP exhibits a normed absolute mean residual (NAMR) of 0.73 with a standard deviation of 0.042, compared to 1.01 (0.087) for Input$\times$Gradient and 6.592 (9.839) for the gradient approach, indicating that LTP remains substantially closer to the target logit than gradient-based alternatives.

7 Conclusion

In this paper, we introduce *Layer-wise Target Propagation* (LTP), a novel framework that recasts attribution as a target-centric backward process: LTP propagates the unembedding vector of the target token backward through the frozen transformer in a single forward and backward pass, recovering an effective target at every residual position without counterfactual examples. To make this propagation accurate, LTP decomposes and linearizes the transformer's computation into distinct pathways through which the target is traced. LTP achieves $O(1)$ time complexity with respect to the number of model components, demonstrating scalability for large models. Extensive experiments on standard interpretability benchmarks demonstrate that LTP achieves state-of-the-art faithfulness and unprecedented efficiency compared to existing baselines.

8 Limitations

While Layer-wise Target Propagation (LTP) offers a highly efficient framework for dense component attribution, its methodology introduces several structural and operational limitations:

Approximation in High-Curvature Regimes: To achieve top-down propagation, LTP relies on local linearization and first-order Taylor expansions. Because it inherently discards second-order cross-derivatives, the method's attribution faithfulness may degrade when analyzing circuits driven by highly coupled, synergistic, or heavily non-linear component interactions.

Memory and Compute Overhead: Although LTP drastically reduces the time complexity to $O(1)$ with respect to the number of components, caching the dense intermediate activations during the forward pass is memory-intensive. Therefore, despite its comparative efficiency, conducting LTP on exceptionally large models or ultra-long context windows still demands a significant memory footprint and computational overhead.

9 Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (RS-2025-00513098 and RS-2025-16067408) and the BK21 FOUR project (AI-driven Convergence Software Education Research Program) funded by the Ministry of Education at Kyungpook National University (KNU) (4120240214871). The A100 GPU used in the experiments was provided by the Department of Data Convergence Computing at KNU, where Young-Kyoon Suh is an adjunct professor.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. 2024. Attnlrp: Attention-aware layer-wise relevance propagation for transformers. *Preprint*, arXiv:2402.05602.

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. 2018. The shattered gradients problem: If resnets are the answer, then what is the question? *Preprint*, arXiv:1702.08591.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor V. Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *ArXiv*, abs/2303.08112.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *ArXiv*, abs/2304.14997.

Misha Denil, Alban Demiraj, and Nando de Freitas. 2014. Extraction of salient sentences from labelled documents. *ArXiv*, abs/1412.6815.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Javier Ferrando, Gerard I. Gállego, Yiannis (Ioannis) Tsiamas, and Marta Ruiz Costa-jussà. 2023. Explaining how transformers use context to build predictions. In *Annual Meeting of the Association for Computational Linguistics*.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta Ruiz Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *ArXiv*, abs/2405.00208.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17432–17445, Miami, Florida, USA. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Ke Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *ArXiv*, abs/2203.14680.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Xiangyu Hong, Che Jiang, Kai Tian, Biqing Qi, Youbang Sun, Ning Ding, and Bowen Zhou. 2025. Depass: Unified feature attributing by simple decomposed forward pass. *ArXiv*, abs/2510.18462.

Farnoush Rezaei Jafari, Oliver Eberle, Ashkan Khakzar, and Neel Nanda. 2025. RelP: Faithful and efficient circuit discovery in language models via relevance patching. *Preprint*, arxiv:2508.21258 [cs].

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *North American Chapter of the Association for Computational Linguistics*.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre

Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4547–4568, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2023. Analyzing feed-forward blocks in transformers through the lens of attention maps. In *International Conference on Learning Representations*.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. In *North American Chapter of the Association for Computational Linguistics*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Neural Information Processing Systems*.

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2023. DecompX: Explaining transformers decisions by propagating token decomposition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2649–2664, Toronto, Canada. Association for Computational Linguistics.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2015. Explaining nonlinear classification decisions with deep taylor decomposition. *ArXiv*, abs/1512.02479.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale.

nostalgebraist. 2020. Interpreting gpt: the logit lens. *AI Alignment Forum*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *ArXiv*, abs/1806.03822.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2019. Learning important features through propagating activation differences. *Preprint*, arXiv:1704.02685.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *Preprint*, arXiv:1706.03825.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864.

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 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

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.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Neural Information Processing Systems*.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *ArXiv*, abs/2211.00593.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, and 25 others. 2024. Qwen2.5 technical report. *ArXiv*, abs/2412.15115.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *ArXiv*, abs/1910.07467.

Fred Zhang and Neel Nanda. 2023. Towards best practices of activation patching in language models: Metrics and methods. *ArXiv*, abs/2309.16042.

A Transformer Architecture

We consider a decoder-only Transformer composed of a stack of L identical layers indexed by $l \in \{1, \ldots, L\}$. Each layer consists of two primary sub-modules: a *multi-head self-attention* (MHSA) block and a *gated feed-forward network* (GLU).

Given an input token sequence $t = (t_1, \ldots, t_T)$, tokens are mapped to vector embeddings $\mathbf{X}^{(0)} \in \mathbb{R}^{T \times d}$ using an embedding matrix $\mathbf{W}_E \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $\mathcal{V}$ denotes the vocabulary. Representations propagate sequentially through the network layers. Under a pre-normalization configuration, the update rule for layer l is given by:

$$\mathbf{X}_{\text{mid}}^{(l)} = \mathbf{X}^{(l-1)} + \text{MHSA}\Big(\text{LN}(\mathbf{X}^{(l-1)})\Big),$$
$$\mathbf{X}^{(l)} = \mathbf{X}_{\text{mid}}^{(l)} + \text{GLU}\Big(\text{LN}(\mathbf{X}_{\text{mid}}^{(l)})\Big),$$

where $\text{LN}(\cdot)$ denotes the normalization applied prior to each sub-module. For notational convenience, we refer to normalized activations as $\tilde{\mathbf{X}}$ when needed.

The final representation $\mathbf{X}^{(L)}$ is projected into the vocabulary space via the unembedding matrix $\mathbf{W}_{UE} \in \mathbb{R}^{d \times |\mathcal{V}|}$, yielding the logits used for next-token prediction.

A key property enabling attribution analysis is the additive structure induced by residual connections around each sub-module, commonly referred to as the *residual stream*. This structure allows the final representation to be expressed as:

$$\mathbf{X}^{(L)} = \mathbf{X}^{(0)} + \sum_{l=1}^{L} \Big(\text{MHSA}(\tilde{\mathbf{X}}^{(l-1)}) + \text{GLU}(\tilde{\mathbf{X}}_{\text{mid}}^{(l)})\Big).$$

As a result, the contribution of any module output can be directly interpreted by projecting it into the vocabulary space via the unembedding matrix (Elhage et al., 2021; nostalgebraist, 2020).

A.1 Root-Mean-Squared Normalization (RMSNorm)

Root-Mean-Squared Normalization (RMSNorm) normalizes activations by their root-mean-square magnitude and is defined as:

$$\text{RMSNorm}(\mathbf{X}) = \frac{\mathbf{X}}{\sqrt{\text{Var}[\mathbf{X}] + \epsilon}} \odot \boldsymbol{\gamma}, \quad (1)$$

where ϵ is a small constant for numerical stability and $\boldsymbol{\gamma}$ denotes learnable scaling parameters.

A.2 Rotary Positional Embeddings (RoPE)

Rotary Positional Embeddings (RoPE) encode positional information by applying a rotation to query and key vectors. Given a vector $x \in \mathbb{R}^d$ at position m, RoPE defines the rotated representation as:

$$R(x, m) = \mathcal{R}_{\Theta,m} x, \quad (2)$$

where $\mathcal{R}_{\Theta,m} \in \mathbb{R}^{d \times d}$ is a block-diagonal rotation matrix composed of $d/2$ independent 2×2 rotation blocks:

$$\mathcal{R}_{\Theta,m} = \begin{pmatrix} M_1 & & 0 \\ & \ddots & \\ 0 & & M_{d/2} \end{pmatrix}. \quad (3)$$

Each block M_j rotates a pair of dimensions by an angle $m\theta_j$:

$$M_j = \begin{pmatrix} \cos(m\theta_j) & -\sin(m\theta_j) \\ \sin(m\theta_j) & \cos(m\theta_j) \end{pmatrix}. \quad (4)$$

The rotation frequencies are defined as a geometric progression:

$$\theta_j = 10000^{-2(j-1)/d}, \quad j \in \{1, \ldots, d/2\}. \quad (5)$$

An important property of RoPE is that the inner product between a query at position m and a key at position n depends only on their relative offset:

$$(R(q, m))^\top R(k, n) = q^\top \mathcal{R}_{\Theta,n-m} k. \quad (6)$$

B Complexity Analysis

As large language models are increasingly deployed on tasks requiring long context windows, the computational and memory scalability of attribution methods becomes a critical bottleneck. In this section, we analyze the complexity of Layer-wise Target Propagation (LTP) and compare it against two established baselines: Activation Patching (AP) and DePass (Hong et al., 2025). We evaluate these methods both theoretically and empirically, demonstrating that LTP uniquely decoupling attribution cost from the number of components, thereby enabling highly efficient dense attribution even over long sequences.

B.1 Theoretical Complexity Analysis

As summarized in Table 4, we evaluate the computational complexity of Layer-wise Target Propagation (LTP) against Activation Patching (AP) and DePass with respect to the number of attributed components M.

Methods	Complexity (w.r.t. M)	Fwd passes
Activation Patching (AP)	$O(M)$	M
DePass	$O(M)$	$\lceil M/B \rceil$
LTP	**O(1)**	**0**

Table 4: Computational complexity comparison with respect to the number of attributed components M. Activation Patching and DePass require forward passes that scale linearly with M, whereas LTP performs attribution with no additional forward passes and a single backward pass.

For a dense attribution task, AP requires M independent forward passes. Because each pass incurs the $O(S^2)$ time complexity of the attention mechanism (where S is the sequence length), its $O(M \cdot S^2)$ scaling becomes computationally prohibitive for long-context tasks. To mitigate this, methods like DePass leverage frozen activations through a modified forward pass, but still suffer from an $O(M)$ complexity bottleneck. DePass explicitly decomposes hidden representations into M independent contribution vectors. For large models (e.g., Llama-3.1-8B-Instruct with $N_{\text{mlp}} \approx 14{,}000$), processing these simultaneously is intractable, necessitating sequential batching that yields $\lceil M/B \rceil$ forward passes (where B is the batch size).

In contrast, LTP adopts a target-centric formulation that is completely independent of M. By propagating a single effective target vector backward from the output, LTP computes the attribution score for any component via a simple dot product with its cached contribution. Consequently, LTP requires 0 additional forward passes and only a single backward pass, achieving a rigorous $O(1)$ time complexity with respect to M. This decoupling effectively eliminates the prohibitive scaling overhead inherent in forward decomposition-based methods.

B.2 Empirical Measurements Analysis

We provide comprehensive empirical runtime and peak memory measurements across four distinct SwiGLU-based LLMs (representing the LLaMA, Qwen, and Mistral families) in Table 5. To ensure a strictly fair comparison, all evaluations were executed under identical hardware configurations, utilizing the same GPU setup and batch size. Furthermore, peak memory allocation was explicitly tracked using PyTorch's native CUDA memory allocator. This approach guarantees the precise measurement of tensor-level VRAM consumption directly managed by the model, effectively isolating the metric from OS-level caching overheads or external system processes.

The empirical results demonstrate that our LTP framework significantly outperforms existing baseline methods in computational speed across all scenarios. Even on short-context datasets like Known 1000, LTP operates consistently faster than Activation Patching (AP) and DePass across all evaluated models. Furthermore, LTP exhibits exceptional scalability and efficiency on longer context lengths, such as the SQuAD v2.0 dataset. For instance, on the Llama-3.1-8B-Instruct model, LTP completed the attribution calculations in just 129.05 seconds. In stark contrast, both AP and DePass required approximately 1.5 hours (5370.65 and 5452.28 seconds, respectively) for the same task, representing a massive 40× speedup for LTP.

Crucially, LTP achieves these drastically faster runtimes while maintaining a peak memory footprint that is highly comparable to the baseline methods across all tested models. This confirms that LTP is a profoundly practical, robust, and resource-efficient solution for real-world tasks involving long context sequences, successfully overcoming the severe computational bottlenecks of traditional intervention-based methods.

C Algorithm of Layer-wise Target Propagation

The complete procedure for Layer-wise Target Propagation (LTP) is formalized in Algorithm 1. The framework operates in a two-stage process: a single forward pass for activation caching followed by a recursive top-down target propagation. By decoupling the attribution cost from the number of attributed components M, LTP achieves $O(1)$ complexity, making it highly scalable for dense attribution tasks in large-scale transformers.

D Derivations of LTP Target Tracing

In this section, we provide the complete mathematical derivations for the target propagation rules underlying Layer-wise Target Propagation (LTP). By propagating a single effective target vector backward through the network, LTP analytically computes the gradient of the attribution score with respect to the intermediate representations. Below, we derive these exact target propagation formulas for the two primary computational blocks of the Transformer architecture: the Gated Linear Unit

Methods	Known 1000		SQuAD v2.0	
	Runtime (s)↓	Peak Memory (MB)↓	Runtime (s)↓	Peak Memory (MB)↓
Llama-3.1-8B-Instruct				
Activation Patching (AP)	14.88	20983.48	5370.65	15649.38
DePass	14.91	20663.91	5452.28	17210.68
Layer-wise Target Propagation (LTP)	**4.98**	21603.86	**129.05**	17512.21
Llama-2-7B-Chat				
Activation Patching (AP)	17.58	24642.69	7229.39	13505.32
DePass	15.74	17321.58	6987.46	14976.61
Layer-wise Target Propagation (LTP)	**5.33**	19449.77	**141.58**	15026.57
Qwen3-4B-Instruct-2507				
Activation Patching (AP)	10.32	15354.79	7336.73	7863.08
DePass	9.04	13866.60	4905.13	8591.08
Layer-wise Target Propagation (LTP)	**4.00**	13985.03	**119.43**	9571.57
Mistral-7B-Instruct-v0.3				
Activation Patching (AP)	17.33	18395.77	9603.84	13973.31
DePass	16.87	19795.21	10808.99	14814.81
Layer-wise Target Propagation (LTP)	**5.55**	21182.21	**170.92**	16191.69

Table 5: Empirical runtime and peak memory measurements across various SwiGLU-based LLMs (representing the LLaMA, Qwen, and Mistral families) on datasets with different context lengths. Notably, LTP achieves vastly superior computational efficiency, delivering drastically faster runtimes while maintaining a comparable peak memory footprint to the baseline methods. It maintains highly robust performance even on the long-context SQuAD v2.0 dataset, successfully overcoming the severe computational bottlenecks of AP and DePass.

(GLU) and the Multi-Head Self-Attention (MHSA) module.

D.1 Gated Linear Unit (GLU) Target Propagation

Forward Pass Definition. Consider a single neuron n within the GLU layer. The contribution of this neuron to the residual stream is defined as:

$$\mathrm{GLU}^{(n)}(\tilde{\mathbf{x}}) = \left(\sigma(\tilde{\mathbf{x}}\mathbf{w}_G^{(n)}) \odot (\tilde{\mathbf{x}}\mathbf{w}_U^{(n)})\right)\mathbf{w}_D^{(n)}$$

where:

- $\sigma(\cdot)$ is the SiLU (Swish) activation function.
- $\mathbf{w}_G^{(n)}, \mathbf{w}_U^{(n)}$ are the column vectors of the gate and up projection matrices.
- $\mathbf{w}_D^{(n)}$ is the row vector of the down projection matrix.
- $\odot$ denotes element-wise multiplication (scalar here for a single neuron).

The attribution of this neuron to a global target $\mathbf{t}$ is given by:

$$S^{(n)} = \mathrm{GLU}^{(n)}(\tilde{\mathbf{x}}) \cdot \mathbf{t}$$

$$S^{(n)} = \underbrace{\sigma(\tilde{\mathbf{x}}\mathbf{w}_G^{(n)})}_{\text{activation } \alpha} \cdot \underbrace{(\tilde{\mathbf{x}}\mathbf{w}_U^{(n)})}_{\text{content } v} \cdot \underbrace{(\mathbf{w}_D^{(n)}\mathbf{t})}_{\text{scalar } \lambda}$$

D.2 Up-Projection (Value) Pathway

To identify the contribution of the information-carrying pathway, we linearize the operation by freezing the gate activation $\alpha^{(n)}$. The attribution becomes linear with respect to $\tilde{\mathbf{x}}$:

$$S_{up}^{(n)} \approx \alpha^{(n)} \cdot (\tilde{\mathbf{x}}\mathbf{w}_U^{(n)}) \cdot \lambda$$

The gradient of this score with respect to the input $\tilde{\mathbf{x}}$ (which represents the propagated target) is:

$$\mathbf{g}_{up}^{(n)}(\mathbf{t}) = \nabla_{\tilde{\mathbf{x}}} S_{up}^{(n)} = \alpha^{(n)}\lambda\mathbf{w}_U^{(n)}$$

Incorporating the RMSNorm scaling parameters $\boldsymbol{\gamma}$ into the effective weight $\tilde{\mathbf{w}}$, we get the paper's formulation:

$$\boxed{\mathbf{g}_{up}^{(n)}(\mathbf{t}) = (\tilde{\mathbf{w}}_U^{(n)}) \cdot \alpha^{(n)} \cdot (\mathbf{w}_D^{(n)}\mathbf{t})}$$

D.3 Gate-Projection Pathway

For the gating pathway, we freeze the content term $v = \tilde{\mathbf{x}}\mathbf{w}_U^{(n)}$ and attribute through the non-linear activation $\sigma(s)$, where $s = \tilde{\mathbf{x}}\mathbf{w}_G^{(n)}$.

$$S_{gate}^{(n)} \approx \sigma(s) \cdot v \cdot \lambda$$

Algorithm 1: The algorithm of Layer-wise Target Propagation (LTP).

Input : Input tokens X, target logit direction $t^{(L)}$, sensitivity parameter p
Output : Attribution scores S_t for all components

```
// Stage 1: Forward Pass and Activation Caching
```
Execute a single forward pass through L layers;
for *each layer* $l = 1$ **to** L **do**
  Cache attention weights $\alpha^{(h)}$ and normalized inputs $\tilde{x}$;
  Cache GLU pre-activations $s^{(n)}$ and gate activations $\alpha^{(n)}$;

```
// Stage 2: Top-Down Target Propagation
```
Initialize effective target $t^{(L)} \leftarrow W_U^{(t)}$;
for *each layer* $l = L$ **to** *1* **do**
  `// 1. Propagate through Gated Linear Unit (GLU)`
  Compute $g_{up}^{(n)}(t)$ and $g_{gate}^{(n)}(t)$ using linearized inverse rules;
  $t_{mid}^{(l)} \leftarrow t^{(l)} + \sum_n (\mu_{up} \cdot g_{up}^{(n)}(t^{(l)}) + \mu_{gate} \cdot g_{gate}^{(n)}(t^{(l)}))$;
  `// 2. Propagate through Multi-Head Self-Attention (MHSA)`
  for *each head* h **do**
    Compute value path $g_V^{(h)}(t_{mid}^{(l)})$ and control paths $g_Q^{(h)}, g_K^{(h)}$;
    $g_{head}^{(h)} \leftarrow \mu_Q g_Q^{(h)} + \mu_K g_K^{(h)} + \mu_V g_V^{(h)}$;
  $t^{(l-1)} \leftarrow t_{mid}^{(l)} + \sum_h g_{head}^{(h)}$;

```
// Stage 3: Score Calculation
// Calculate scores for each hierarchical level
```
$S_t(x_i^{(0)}) \leftarrow (x_i^{(0)})^\top t_i^{(0)}$; `// Input Token Attribution`
$S_t(h^{(l)}) \leftarrow \text{Attn}^{(h)}(\tilde{x}^{(l-1)}) \cdot t_{mid}^{(l)}$; `// Attention Head Attribution`
$S_t(n^{(l)}) \leftarrow \text{GLU}^{(n)}(\tilde{x}_{mid}^{(l)}) \cdot t^{(l)}$; `// MLP Neuron Attribution`
return S_t;

Using the linear approximation $\sigma(s) \approx s \cdot \frac{\sigma(s)}{s} = s \cdot \frac{\alpha^{(n)}}{s}$, we rewrite the score:

$$S_{gate}^{(n)} \approx (\tilde{\mathbf{x}}\mathbf{w}_G^{(n)}) \cdot \frac{\alpha^{(n)}}{s} \cdot v \cdot \lambda$$

Taking the gradient with respect to $\tilde{\mathbf{x}}$:

$$\mathbf{g}_{gate}^{(n)}(\mathbf{t}) = \nabla_{\tilde{\mathbf{x}}} S_{gate}^{(n)} = \mathbf{w}_G^{(n)} \cdot \frac{\alpha^{(n)}}{s} \cdot v \cdot \lambda$$

Grouping terms yields the final propagation formula:

$$\boxed{\mathbf{g}_{gate}^{(n)}(\mathbf{t}) = (\tilde{\mathbf{w}}_G^{(n)}) \cdot \frac{\alpha^{(n)}}{s} \cdot \left(v \cdot \mathbf{w}_D^{(n)}\mathbf{t}\right)}$$

D.4 Multi-Head Self-Attention (MHSA) Target Propagation

Forward Pass Definition. The output of an attention head h at query position i is:

$$\mathbf{y}_i^{(h)} = \sum_{j \le i} \alpha_{ij} (\tilde{\mathbf{x}}_j \mathbf{W}_V^{(h)}) \mathbf{W}_O^{(h)}$$

where the attention weights α_{ij} are computed via Softmax over the scaled dot-product of Rotary Positional Embedding (RoPE) transformed queries and keys:

$$\alpha_{ij} = \text{Softmax}_j(s_{ij})$$

$$s_{ij} = \frac{R_Q(\tilde{\mathbf{x}}_i \mathbf{W}_Q^{(h)}) \cdot R_K(\tilde{\mathbf{x}}_j \mathbf{W}_K^{(h)})}{\sqrt{d_h}}$$

The attribution to the target $\mathbf{t}$ is $S = \mathbf{y}_i^{(h)} \cdot \mathbf{t}$.

Value Pathway. We assume the attention pattern α_{ij} is fixed. The operation becomes linear with respect to the input $\tilde{\mathbf{x}}_j$ via the value matrix $\mathbf{W}_V$:

$$S_{value} = \sum_{j \le i} \alpha_{ij} (\tilde{\mathbf{x}}_j \mathbf{W}_V^{(h)})(\mathbf{W}_O^{(h)} \mathbf{t})$$

The propagated target to token j is derived by taking the gradient w.r.t $\tilde{\mathbf{x}}_j$:

$$\boxed{\mathbf{g}_{V,j}^{(h)}(\mathbf{t}) = \tilde{\mathbf{W}}_V^{(h)} \left(\sum_{i \ge j} \alpha_{ij} (\mathbf{W}_O^{(h)} \mathbf{t}) \right)}$$

Query and Key Pathways (Softmax Decomposition). To propagate through the Query and Key paths, we must differentiate the Softmax function. Let the projected target vector be $\mathbf{t}' = \mathbf{W}_O^{(h)} \mathbf{t}$ and the value vector be $\mathbf{v}_j = \tilde{\mathbf{x}}_j \mathbf{W}_V^{(h)}$. The attribution score is:

$$S = \sum_k \alpha_{ik} (\mathbf{v}_k \cdot \mathbf{t}')$$

We first find the sensitivity of S with respect to the pre-softmax logits s_{ij}. Using the derivative of Softmax, $\frac{\partial \alpha_{ik}}{\partial s_{ij}} = \alpha_{ij}(\delta_{jk} - \alpha_{ik})$:

$$\begin{aligned} \frac{\partial S}{\partial s_{ij}} &= \sum_k \frac{\partial \alpha_{ik}}{\partial s_{ij}} (\mathbf{v}_k \cdot \mathbf{t}') \\ &= \sum_k \alpha_{ij} (\delta_{jk} - \alpha_{ik})(\mathbf{v}_k \cdot \mathbf{t}') \\ &= \alpha_{ij} \left(\mathbf{v}_j \cdot \mathbf{t}' - \left(\sum_k \alpha_{ik} \mathbf{v}_k \right) \cdot \mathbf{t}' \right) \end{aligned}$$

Defining the expected head output as $\boldsymbol{\mu}_i^{(h)} = \sum_k \alpha_{ik} \mathbf{v}_k$, we obtain the scalar interaction term δ_{ij}:

$$\delta_{ij} = \alpha_{ij} \big(\mathbf{v}_j - \boldsymbol{\mu}_i^{(h)} \big) \cdot (\mathbf{W}_O^{(h)} \mathbf{t})$$

Now we propagate δ_{ij} through the dot product s_{ij}.

Query Path. The gradient of s_{ij} with respect to the query input vector $\mathbf{q}_i = \tilde{\mathbf{x}}_i \mathbf{W}_Q$ is proportional to the key vector $\mathbf{k}_j$.

$$\nabla_{\tilde{\mathbf{x}}_i} s_{ij} = \mathbf{W}_Q^{(h)} R_{Q,i}^{-1} \left(\frac{R_K(\tilde{\mathbf{x}}_j \mathbf{W}_K^{(h)})}{\sqrt{d_h}} \right)$$

Aggregating over all j, the target for the query input at position i is:

$$\boxed{\mathbf{g}_{Q,i}^{(h)}(\mathbf{t}) = \tilde{\mathbf{W}}_Q^{(h)} \sum_{j \le i} \delta_{ij} \cdot R_{Q,i}^{-1} \left(\frac{R_K(\tilde{\mathbf{x}}_j \mathbf{W}_K^{(h)})}{\sqrt{d_h}} \right)}$$

Key Path. Similarly, the gradient of s_{ij} with respect to the key input vector $\mathbf{k}_j = \tilde{\mathbf{x}}_j \mathbf{W}_K$ is proportional to the query vector $\mathbf{q}_i$.

$$\nabla_{\tilde{\mathbf{x}}_j} s_{ij} = \mathbf{W}_K^{(h)} R_{K,j}^{-1} \left(\frac{R_Q(\tilde{\mathbf{x}}_i \mathbf{W}_Q^{(h)})}{\sqrt{d_h}} \right)$$

Aggregating over all i that attend to j (i.e., $i \ge j$ in causal attention), the target for the key input at position j is:

$$\boxed{\mathbf{g}_{K,j}^{(h)}(\mathbf{t}) = \tilde{\mathbf{W}}_K^{(h)} \sum_{i \ge j} \delta_{ij} \cdot R_{K,j}^{-1} \left(\frac{R_Q(\tilde{\mathbf{x}}_i \mathbf{W}_Q^{(h)})}{\sqrt{d_h}} \right)}$$

E Dataset Configurations

In this section, we provide detailed information regarding the licenses, usage, and preprocessing steps for the datasets used to evaluate the Layer-wise Target Propagation (LTP) framework. We conduct experiments across four representative tasks: factual knowledge, reading comprehension, sentiment analysis, and circuit discovery.

E.1 Licenses and Usage

All datasets utilized in this study are publicly accessible and were used in strict accordance with their respective licensing terms for research purposes.

- **Known 1000.** This dataset is distributed under the MIT License. It consists of 1,000 factual statements designed to evaluate how language models represent and utilize factual associations.
- **SQuAD v2.0.** The Stanford Question Answering Dataset is licensed under Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). It is a standard benchmark for reading comprehension, requiring the model to identify answer spans within a given context or determine if a question is unanswerable.
- **IMDb Sentiment.** Usage of this dataset follows the IMDb Non-Commercial Licensing terms, which allow for limited personal and non-commercial use. In compliance with these terms, we acknowledge that the information is courtesy of IMDb (https://www.imdb.com) and is used with permission. The data was sourced from official contributor datasets to ensure adherence to copyright and conditions of use. It contains movie reviews labeled for binary sentiment analysis.

- **Indirect Object Identification (IOI).** This dataset is distributed under the MIT License. It serves as a specialized benchmark for component-level circuit discovery.

E.2 Dataset Preprocessing Details

To maintain consistent evaluation signals and manage computational complexity, we used a combined dataset of the Test and Dev sets from the experiments. We used 1,000 samples only for the IMDB dataset, while using the total combined sample size for the rest. We performed systematic preprocessing as follows:

- **Known 1000.** We specifically selected instances where the model provided a correct baseline prediction for the target entity to ensure that the attribution was grounded in existing model knowledge. The data was then formatted as a prefix-completion task, ensuring that target entities consisted of short token spans to facilitate precise attribution measurement.
- **SQuAD v2.0.** Our processing focused exclusively on answerable questions to evaluate the model's ability to attribute information to the provided context. The preprocessing involved normalizing whitespace and special characters while ensuring all questions concluded with a proper question mark. Furthermore, we utilized the answer_start index to establish exact ground-truth tokens and filtered for answers consisting of no more than two words to ensure the localization signal remained clear and unambiguous.
- **IMDb Sentiment.** We sampled a balanced subset of 1,000 reviews, comprising 500 positive and 500 negative samples to prevent sentiment bias. The text was cleaned by removing HTML tags and collapsing redundant spaces to prevent tokenization artifacts. To ensure sufficient context for attribution while maintaining computational efficiency, we filtered reviews to maintain a character length between 200 and 500.
- **Indirect Object Identification (IOI).** This dataset was constructed using fixed linguistic templates to isolate the specific circuit responsible for identifying indirect objects. Our preprocessing focused on identifying the precise token positions of the subjects and indirect objects within these templates, which allowed us to measure the direct and indirect effects of internal components with high resolution.

F Experiment Reproducibility

All experiments, including model inference and the recursive propagation of Layer-wise Target Propagation (LTP) targets, were conducted on a high-performance computing cluster to ensure a consistent and high-throughput execution environment. The specific hardware and software configurations used for our empirical evaluations are as follows :

- **Hardware Resources.** We utilized an NVIDIA DGX system equipped with NVIDIA A100-SXM4-40GB GPUs. Each GPU provides 40 GB of HBM2 memory, which is essential for caching the high-dimensional activations required for attribution across large-scale models like Llama-3-8B-Instruct. The high memory bandwidth of the A100 architecture facilitated efficient processing of dense attribution tasks.
- **Software Environment.** The system operated on CUDA Version 12.2 with NVIDIA Driver Version 535.161.08. This environment supports the latest optimization libraries required to execute transformer-based models and perform efficient tensor operations during the target propagation stage.
- **Implementation Details.** The LTP framework and all comparative baselines, such as Integrated Gradients and DePass, were implemented using the PyTorch library, and the results were described based on a single execution.

G Qualitative Attribution Examples

To complement the quantitative faithfulness results in Section 5, we provide qualitative attribution visualizations that illustrate the behavior of Layer-wise Target Propagation (LTP) on natural language inputs. Figure 5 presents two complementary views.

Token-level Attribution on IMDb (left). The left panel of Figure 5 compares LTP against representative baselines on an IMDb sentiment classification example. Red highlights mark tokens that positively contribute to the predicted sentiment, while blue marks tokens whose presence pushes

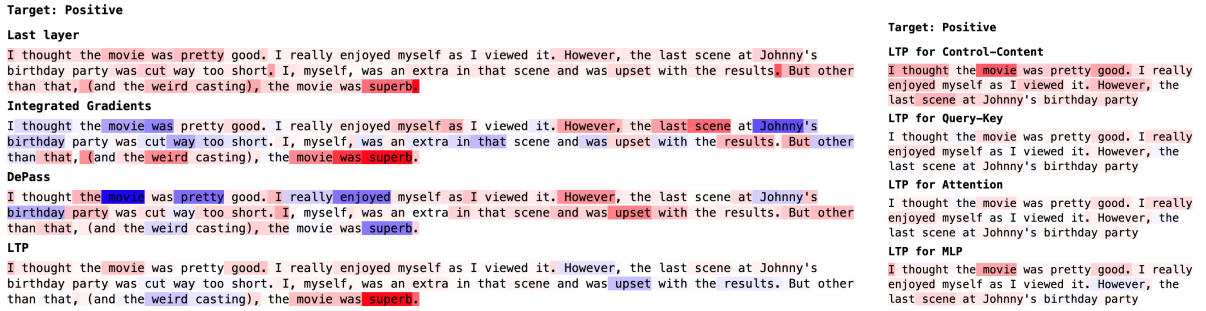

Figure 5: Qualitative attribution examples. Red highlights denote tokens that positively contribute to the target, while blue denotes tokens that hinder it. As shown on the IMDb dataset (left), LTP more precisely distinguishes between helpful and harmful context compared to baseline methods. The right panel illustrates LTP's attribution across different configurations.

Sensitivity	**IMDb** (input-level, p)							**IOI** (component-level, p)						
	0.0	0.2	0.4	0.5	0.6	0.9	1.0	0.0	0.1	0.2	0.3	0.5	0.7	1.0
Control-Content	6.54	24.80	36.78	**38.26**	37.64	27.72	27.44	84.10	99.70	112.95	**113.53**	87.74	40.42	30.04
Query-Key	35.74	36.88	38.21	38.26	38.58	**40.10**	39.68	91.82	97.13	**101.98**	100.50	87.74	73.01	49.87
Attention	6.78	26.74	36.91	38.26	**38.31**	32.21	28.37	54.07	53.83	60.52	70.22	**87.74**	57.35	29.14
MLP	32.42	38.06	**38.88**	38.26	36.93	29.56	26.15	144.35	**153.37**	148.56	132.52	87.74	56.83	45.04

Table 6: Extended sensitivity analysis for **input-level** attribution on IMDb (Llama-3.1-8B-Instruct) and **component-level** attribution on the IOI (Llama-3.1-8B-Instruct). We report the Total AUC (recovery − disruption; higher is better) for each weighting family in Table 3. At $p = 0.5$, all families coincide with the balanced configuration. **Bold** indicates the best result per family and setting.

the prediction in the opposite direction. LTP produces sharper polarity contrasts and more selectively highlights sentiment-bearing tokens—such as explicit affective adjectives and negation cues—than baselines that diffuse importance across syntactic context. This aligns with the quantitative observation in Table 1 that attribution to complex semantic concepts like sentiment exposes the limitations of attention-only approaches: while raw attention captures token-to-token mixing, it does not by itself signal the semantic polarity that drives the final prediction.

Pathway Sensitivity (right). The right panel compares LTP's attribution maps on the same IMDb input under the four parameter combinations summarized in Table 3—Control–Content, Query–Key, Attention, and MLP—each evaluated at the endpoint $p = 1$, which routes attribution through the corresponding pathway in its purest form. Despite sharing the same input and target, the four maps highlight visibly different token subsets, making the relative contribution of each pathway concrete and complementing the quantitative trends in Figure 4.

H Extended Sensitivity Analysis

The choice of the bilinear weighting parameters μ is first and foremost motivated by the conservation property, which requires that the relevances of the inputs of a bilinear function sum to the relevance at its output. Among the values satisfying this property, we adopt $p = 0.5$ because it corresponds to balanced propagation across both inputs and is therefore the natural choice absent any task-specific prior. The sensitivity analysis in Section 5.2 supports this choice post-hoc; in this section, we strengthen the argument by extending the analysis to fundamentally different tasks.

Concretely, we run the full p-sweep ($p \in [0, 1]$) on two settings that differ from the main-paper analysis in both task and granularity: IMDb (input-level attribution task) and IOI (component-level attribution task). Following the weighting families defined in Table 3, we report the Total AUC (recovery - disruption; higher is better) per family. Table 6 presents the results for both settings.

The results first support $p = 0.5$ as a solid choice regardless of the task: the balanced configuration remains at or near the peak of every family in the input-level setting and avoids the sharp degradation observed at the extremes of each

sweep. Beyond this, the extended experiments reveal task-consistent properties. For input-level attribution, a slightly higher p on the Query-Key interaction leads to the best results, whereas for component-level attribution, the best results are achieved with a lower p on the MLP gate-up interaction. These gains, which are substantial especially for component-level attribution, suggest that task-adaptive weighting is a promising direction for future work.

I Generalizability

In this section, we evaluate the generalizability of Layer-wise Target Propagation (LTP) across different model architectures to ensure our findings are robust and not strictly tied to a single model family.

I.1 Generalization within the LLaMA Family

We initially extended our experiments to Llama-2-7B-Chat not only Llama-3.1-8B-Instruct. The empirical results demonstrate that LTP maintains consistent attribution accuracy and computational efficiency within the LLaMA architecture.

The input and component attribution results for Llama-2-7B-Chat are shown in Table 7, and 8 (with corresponding Figures 6 and 7).

I.2 Generalization to Non-LLaMA SwiGLU Architectures

To address concerns regarding generalization to other model families, we conducted additional experiments on the Qwen and Mistral model families, specifically Qwen3-4B-Instruct-2507, Qwen2.5-32B-Instruct, and Mistral-7B-Instruct-v0.3. While these models share a SwiGLU-based Transformer backbone with the LLaMA series, they are fundamentally distinct. They feature structural variations in the attention mechanism (e.g., Sliding Window Attention in Mistral), utilize different tokenizers, and undergo entirely different pre- and post-training pipelines.

We present quantitative results and visualizations for Qwen3-4B-Instruct-2507 in Tables 9 and 10 (with corresponding Figures 8 and 9). Similarly, results for Qwen2.5-32B-Instruct are detailed in Tables 11 and 12 (Figures 10 and 11), and for Mistral-7B-Instruct-v0.3 in Tables 13 and 14 (Figures 12 and 13). Empirical results demonstrate that LTP exhibits robust superiority over existing baseline methods across these architectural variations. By consistently achieving state-of-the-art performance in the majority of evaluations, LTP confirms its broad applicability and the stability of its performance.

I.3 Generalization to Non-SwiGLU Architectures

While our derivation focuses on the SwiGLU Transformer, the underlying target-propagation framework is not tied to gated MLPs. To probe this, we additionally evaluate LTP on GPT-2 Large, a classical Transformer whose MLP block employs a GELU activation instead of a gated linear unit. Table 15 and Figure 14 report the input attribution results on Known 1000, while Table 16 and Figure 15 report the component attribution results on Known 1000 and IOI.

For input attribution, LTP achieves the best disruption, recovery, and total scores among all evaluated methods. For component attribution, AtP and LTP both attain zero disruption on both benchmarks, while AtP achieves the higher recovery AUC. Notably, however, the recovery curves in Figure 15 show that LTP recovers the original prediction probability with the fewest components, rising substantially earlier than AtP, whose recovery overtakes only at the larger component ratios that dominate the AUC. These results indicate that LTP transfers to non-SwiGLU architectures without sacrificing its precise localization of causally relevant components.

Methods	Known 1000			SQuAD v2.0			IMDb		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference									
Random	13.17	20.64	7.47	23.74	25.04	1.30	58.98	57.80	-1.18
Attention-based									
Last layer	6.53	28.85†	22.32†	8.90	31.95	23.05	59.81	60.94	1.13
Mean	5.97†	25.10	19.13	5.48†	33.35†	27.87†	**46.75**	**78.60**	**31.85**
Rollout	15.45	16.17	0.72	22.64	27.35	4.71	83.31	55.54	-27.77
Gradient-based									
Gradient	12.14	18.75	6.61	16.94	15.55	-1.39	55.16	63.90	8.74
Input×Gradient	11.31	20.31	9.00	14.73	17.16	2.43	69.39	50.85	-18.54
Integrated Gradients	11.58	25.30	13.72	9.87	26.08	16.21	52.30	68.19	15.89
Decomposition-based									
DePass	7.55	26.29	18.74	12.62	27.49	14.87	68.19	56.13	-12.06
Contextual mixing-based									
IFR	15.22	12.76	-2.46	12.62	26.29	2.65	91.62	48.67	-42.95
Proposed Method									
Layer-wise Target Propagation (LTP)	**5.60**	**34.41**	**28.81**	**3.73**	**36.08**	**32.35**	48.21†	74.53†	26.32†

Table 7: Faithfulness evaluation of crucial token attribution on **Llama-2-7B-Chat** across **Known 1000** (factual knowledge), **SQuAD v2.0** (reading comprehension), and **IMDb** (sentiment analysis). Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful token localization; *total* denotes their aggregate AUC score. **Bold** indicates the best result, and † the second best. Layer-wise Target Propagation consistently achieves the highest faithfulness across datasets.

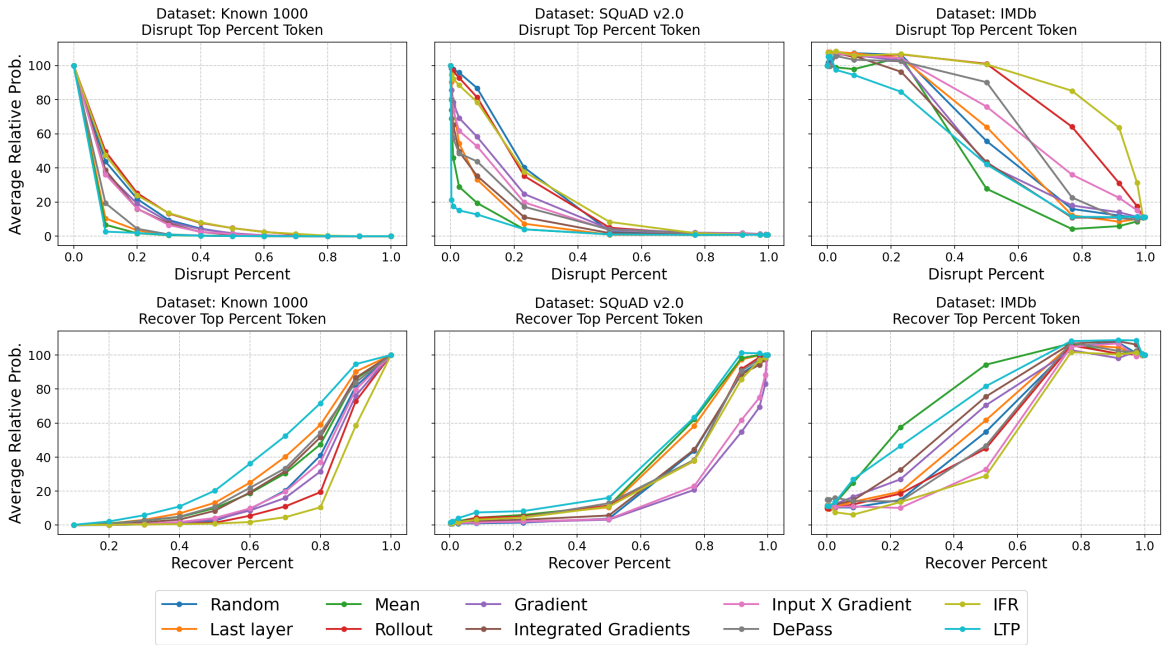

Figure 6: Performance comparison with baseline attribution methods on **Llama-2-7B-Chat** under different top input token masking strategies. Our approach shows *lower* Top-k disruption and *higher* Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000			IOI		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference						
Random	22.89	27.65	4.76	24.07	27.67	3.60
Baselines						
Attn-only	2.34	92.58	90.24	0.53	91.39†	90.86†
MLP-only	1.74	29.16	27.42	5.59	27.58	21.99
Norm-only	0.24	64.68	64.44	0.05	66.98	66.93
Gradient-based						
Gradient	0.22†	28.21	27.99	0.01†	30.69	30.68
AtP	**0.00**	120.80†	120.80†	**0.00**	76.30	76.30
Contextual mixing-based						
IFR	0.69	30.49	29.80	0.48	28.86	28.38
Proposed Method						
Layer-wise Target Propagation (LTP)	**0.00**	**136.45**	**136.45**	**0.00**	**127.16**	**127.16**

Table 8: Component-level attribution performance on **Llama-2-7B-Chat** across **Known 1000** (factual knowledge) and **IOI** (Indirect Object Identification) benchmarks. Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful localization of causally relevant components; the *total* score aggregates both effects. **Bold** denotes the best result, and † indicates the second best. Layer-wise Target Propagation consistently achieves the most faithful component localization across benchmarks.

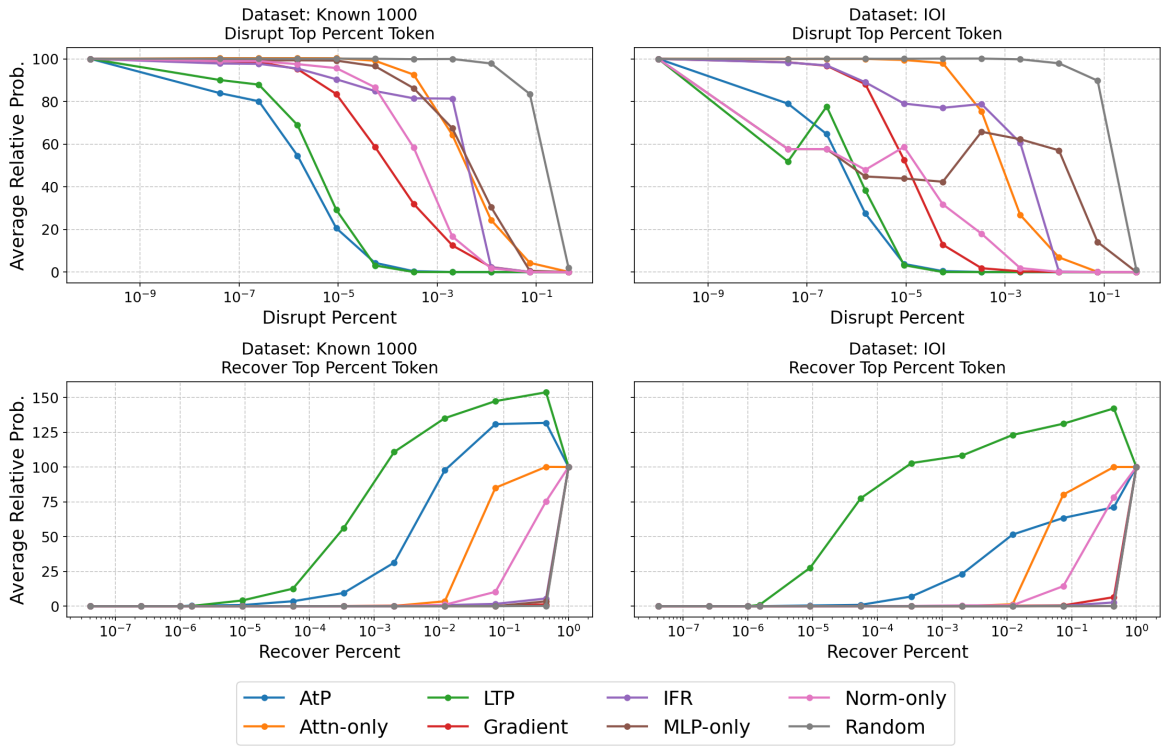

Figure 7: Performance comparison with baseline attribution methods on **Llama-2-7B-Chat** under different top model component masking strategies. Our approach shows lower Top-k disruption and higher Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000			SQuAD v2.0		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference						
Random	11.52	19.64	8.12	48.89	50.00	1.11
Attention-based						
Last layer	8.03	22.82†	14.79	26.04	72.41	46.37
Mean	9.50	22.35	12.85	**11.69**	91.55†	79.86†
Rollout	15.25	11.79	-3.46	43.81	66.12	22.31
Gradient-based						
Gradient	10.88	18.07	7.19	57.82	35.49	-22.33
Input×Gradient	10.54	17.95	7.41	46.23	47.40	1.17
Integrated Gradients	7.55	16.18	8.63	41.39	49.99	8.60
Decomposition-based						
DePass	**6.24**	**26.74**	**20.50**	13.74	66.19	52.45
Contextual mixing-based						
IFR	15.80	11.28	-4.52	43.74	66.46	22.72
Proposed Method						
Layer-wise Target Propagation (LTP)	6.54†	22.37	15.83†	11.93†	**93.86**	**81.93**

Table 9: Faithfulness evaluation of crucial token attribution on **Qwen3-4B-Instruct-2507** across **Known 1000** (factual knowledge), and **SQuAD v2.0** (reading comprehension). Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful token localization; *total* denotes their aggregate AUC score. **Bold** indicates the best result, and † the second best. Layer-wise Target Propagation consistently achieves the highest faithfulness across datasets.

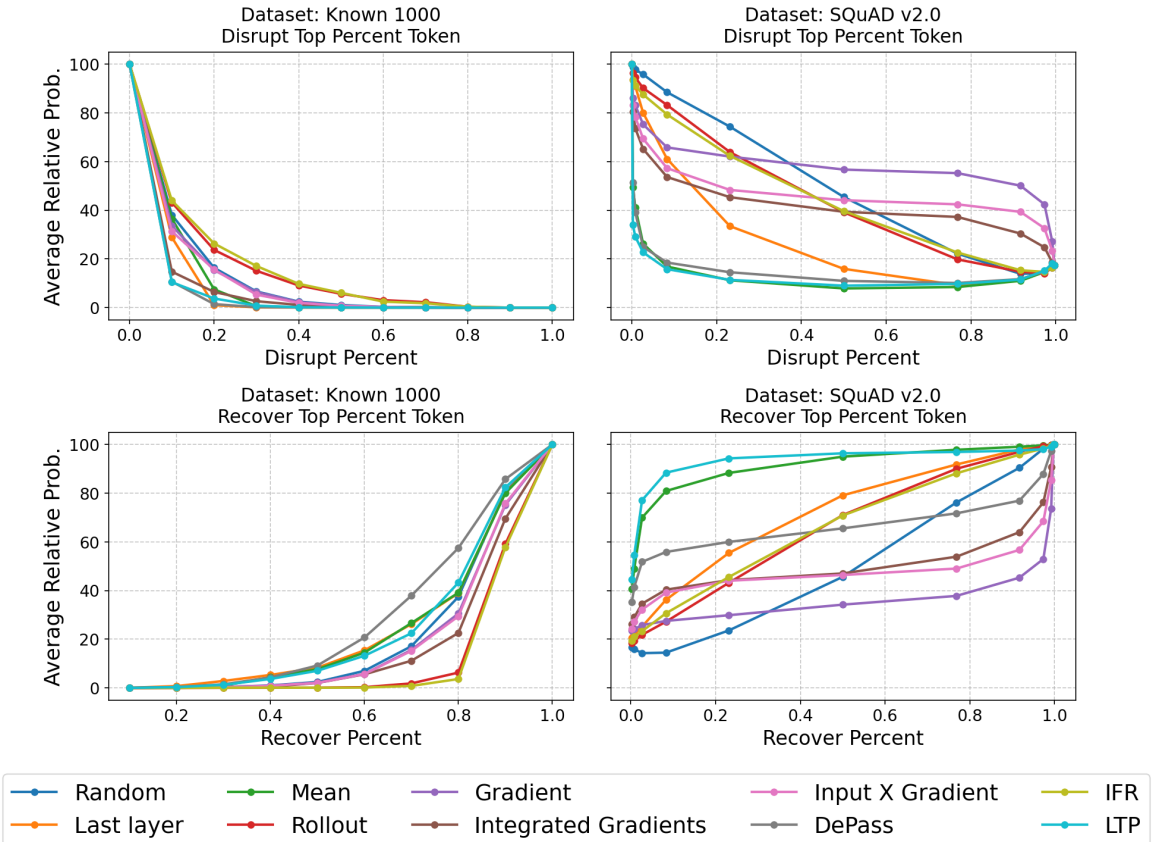

Figure 8: Performance comparison with baseline attribution methods on **Qwen3-4B-Instruct-2507** under different top input token masking strategies. Our approach shows *lower* Top-k disruption and *higher* Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000			IOI		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference						
Random	19.00	27.53	8.53	22.63	27.55	4.92
Baselines						
Attn-only	2.13	74.12†	71.99†	0.66	**83.11**	**82.45**
MLP-only	0.01†	28.66	28.65	0.01†	27.56	27.55
Norm-only	**0.00**	47.64	47.64	**0.00**	28.68	28.68
Gradient-based						
Gradient	0.03	27.54	27.51	**0.00**	29.54	29.54
AtP	**0.00**	41.98	41.98	**0.00**	32.25	32.25
Contextual mixing-based						
IFR	0.17	38.68	38.51	0.02	28.59	28.57
Proposed Method						
Layer-wise Target Propagation (LTP)	**0.00**	**105.21**	**104.35**	**0.00**	54.91†	54.91†

Table 10: Component-level attribution performance on **Qwen3-4B-Instruct-2507** across **Known 1000** (factual knowledge) and **IOI** (Indirect Object Identification) benchmarks. Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful localization of causally relevant components; the *total* score aggregates both effects. **Bold** denotes the best result, and † indicates the second best. Layer-wise Target Propagation consistently achieves the most faithful component localization across benchmarks.

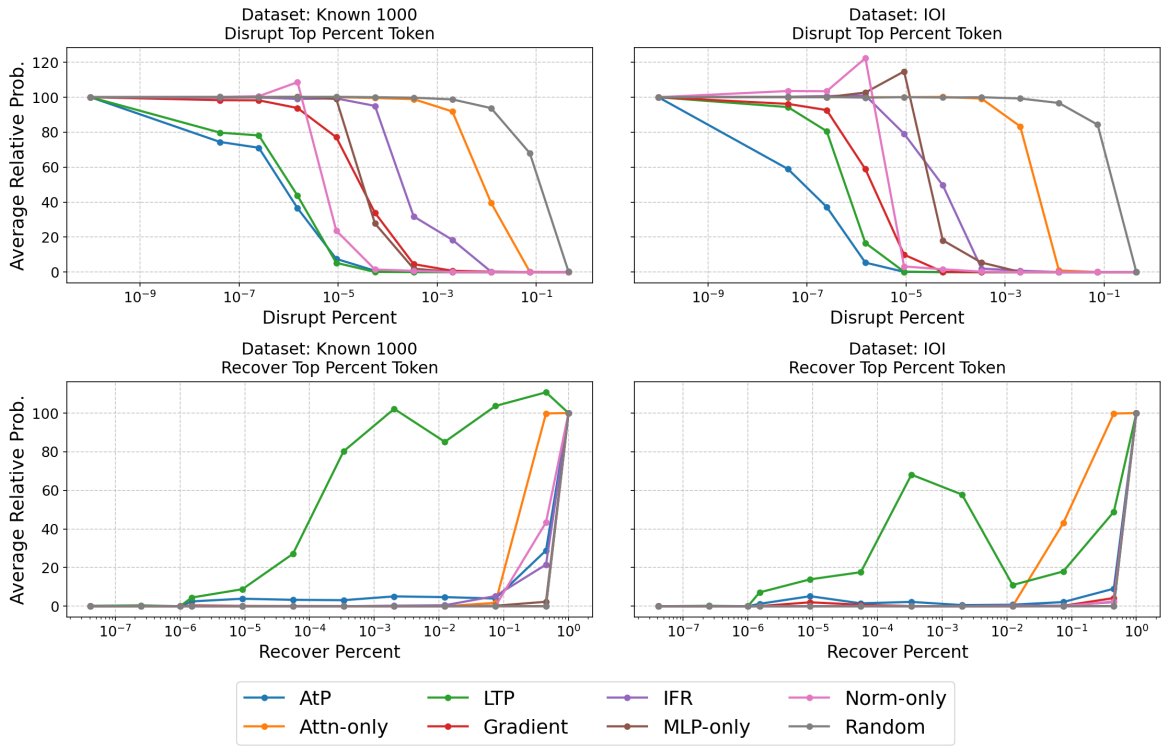

Figure 9: Performance comparison with baseline attribution methods on **Qwen3-4B-Instruct-2507** under different top model component masking strategies. Our approach shows lower Top-k disruption and higher Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000		
	dis. ↓	rec. ↑	total ↑
Reference			
Random	20.00	29.79	9.79
Attention-based			
Last layer	10.63†	32.18	21.55†
Mean	12.06	33.07†	21.01
Rollout	19.70	11.37	-8.33
Gradient-based			
Gradient	19.44	26.67	7.23
Input×Gradient	21.52	26.92	5.40
Integrated Gradients	18.60	27.70	9.10
Decomposition-based			
DePass	12.29	20.53	8.24
Contextual mixing-based			
IFR	18.75	26.61	7.86
Proposed Method			
Layer-wise Target Propagation (LTP)	**10.32**	**41.35**	**31.03**

Table 11: Faithfulness evaluation of crucial token attribution for **Qwen2.5-32B-Instruct** on **Known 1000** (factual knowledge). Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful token localization; *total* denotes their aggregate AUC score. **Bold** indicates the best result, and † the second best. Layer-wise Target Propagation consistently achieves the highest faithfulness across datasets.

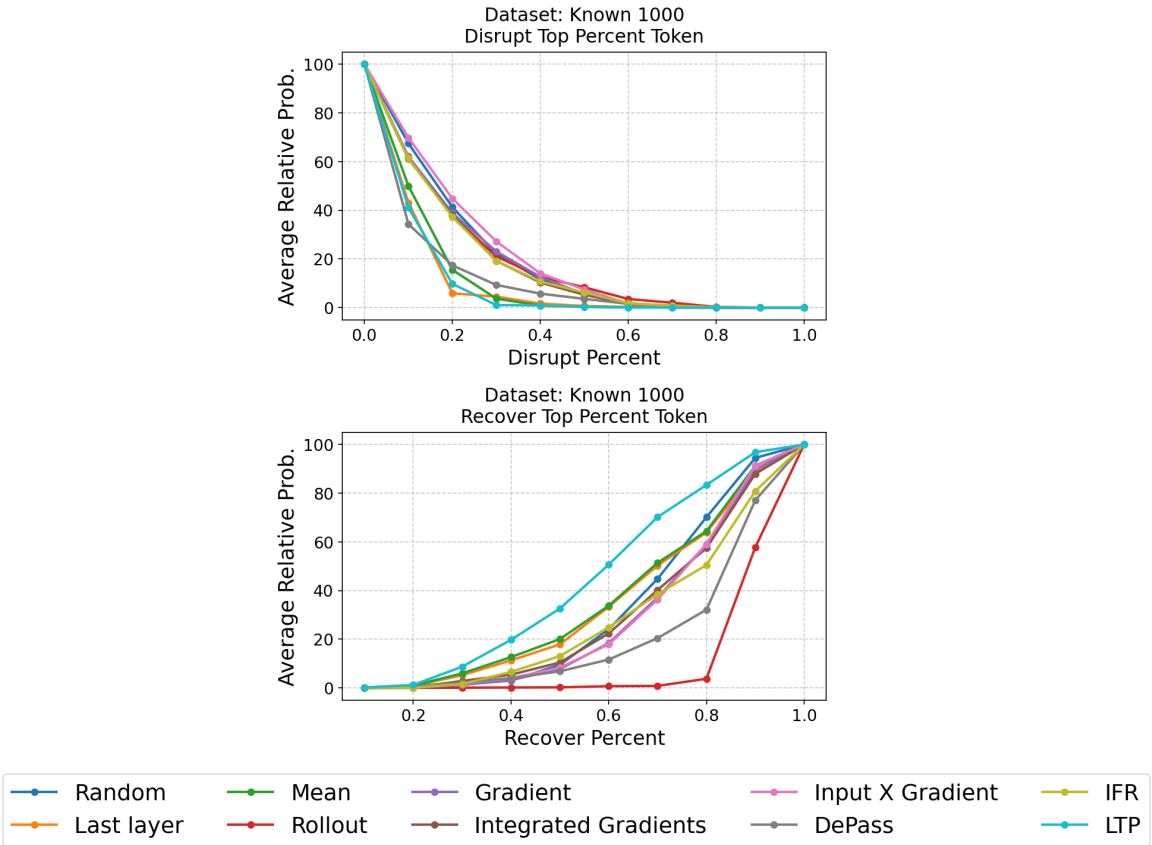

Figure 10: Performance comparison with baseline attribution methods on **Qwen2.5-32B-Instruct** under different top input token masking strategies. Our approach shows *lower* Top-k disruption and *higher* Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000			IOI		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference						
Random	22.59	27.55	4.96	24.45	27.68	3.23
Baselines						
Attn-only	4.25	74.29	70.04	2.07	78.50†	76.43†
MLP-only	0.01†	47.25	47.24	0.55	28.17	27.62
Norm-only	**0.00**	67.75†	67.75†	0.01†	65.45	65.44
Gradient-based						
Gradient	0.05	27.56	27.51	**0.00**	29.01	29.01
AtP	**0.00**	51.69	51.69	**0.00**	54.06	54.06
Contextual mixing-based						
IFR	0.38	31.29	30.91	0.14	27.56	27.42
Proposed Method						
Layer-wise Target Propagation (LTP)	**0.00**	**187.56**	**187.56**	**0.00**	**126.85**	**126.85**

Table 12: Component-level attribution performance for **Qwen2.5-32B-Instruct** across **Known 1000** (factual knowledge) and **IOI** (Indirect Object Identification) benchmarks. Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful localization of causally relevant components; the *total* score aggregates both effects. **Bold** denotes the best result, and † indicates the second best. Layer-wise Target Propagation consistently achieves the most faithful component localization across benchmarks.

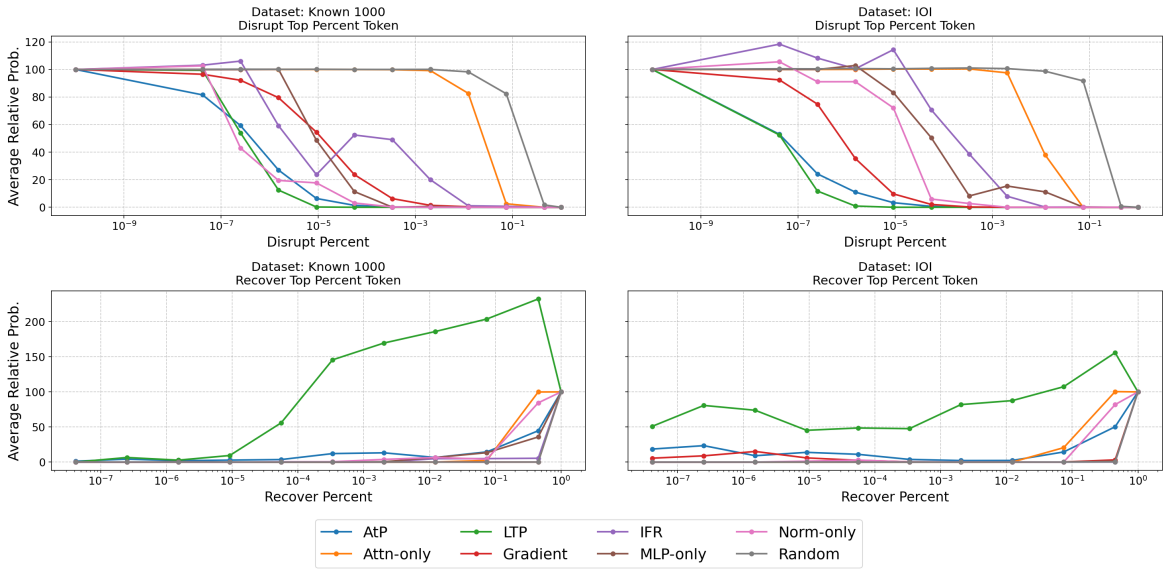

Figure 11: Performance comparison with baseline attribution methods on **Qwen2.5-32B-Instruct** under different top model component masking strategies. Our approach shows lower Top-k disruption and higher Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000			SQuAD v2.0		
	dis.↓	rec.↑	total↑	dis.↓	rec.↑	total↑
Reference						
Random	19.67	26.38	6.71	34.63	34.68	0.05
Attention-based						
Last layer	7.80†	30.41	22.61	9.79	62.84	53.05
Mean	8.61	32.27	23.66	10.78	65.99†	55.21†
Rollout	19.79	20.46	0.67	27.94	56.88	28.94
Gradient-based						
Gradient	17.31	24.66	7.35	31.19	27.56	-3.63
Input×Gradient	16.31	28.18	11.87	25.65	30.44	4.79
Integrated Gradients	9.72	38.14†	28.42†	8.67	55.98	47.31
Decomposition-based						
DePass	9.64	37.53	27.89	**7.92**	53.93	46.01
Contextual mixing-based						
IFR	21.97	16.43	-5.54	32.56	52.78	20.22
Proposed Method						
Layer-wise Target Propagation (LTP)	**7.16**	**43.23**	**36.07**	8.06†	**70.97**	**62.91**

Table 13: Faithfulness evaluation of crucial token attribution on **Mistral-7B-Instruct-v0.3** across **Known 1000** (factual knowledge), and **SQuAD v2.0** (reading comprehension). Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful token localization; *total* denotes their aggregate AUC score. **Bold** indicates the best result, and † the second best. Layer-wise Target Propagation consistently achieves the highest faithfulness across datasets.

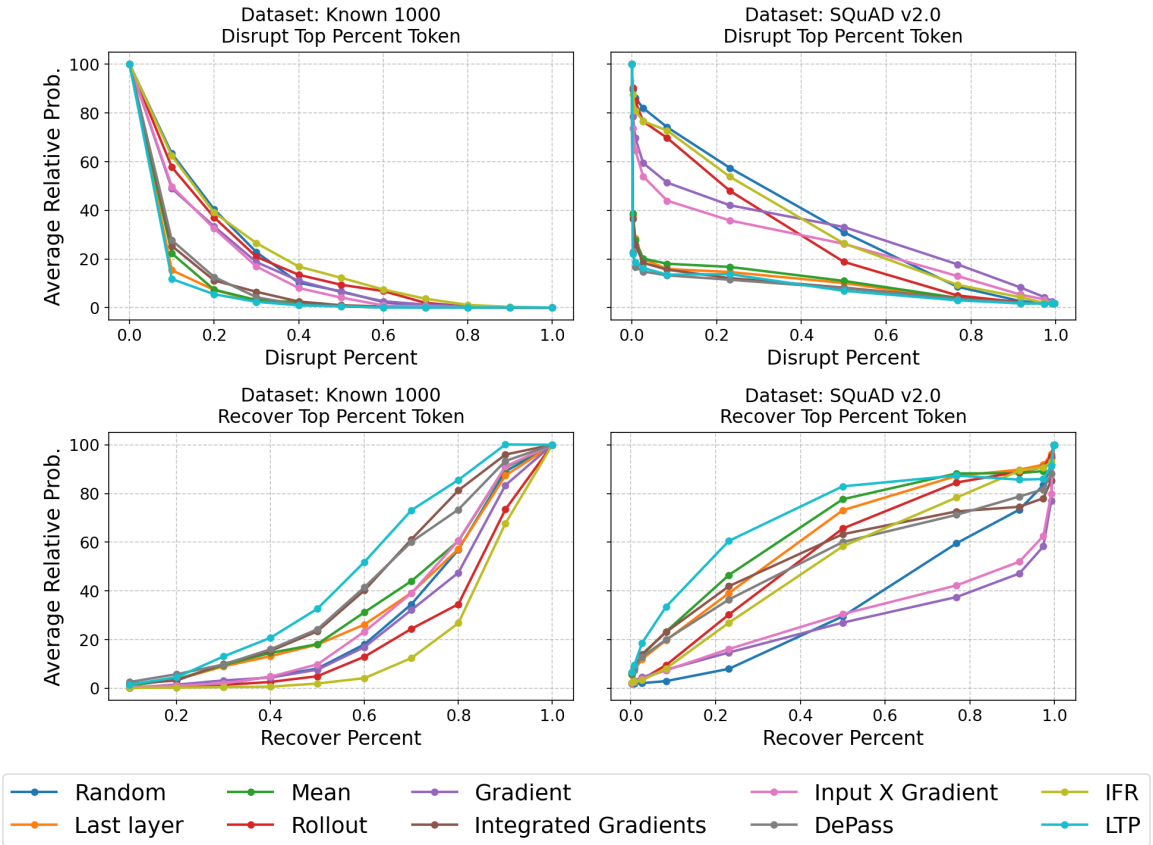

Figure 12: Performance comparison with baseline attribution methods on **Mistral-7B-Instruct-v0.3** under different top input token masking strategies. Our approach shows *lower* Top-k disruption and *higher* Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000			IOI		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference						
Random	23.39	27.84	4.45	25.27	27.60	2.33
Baselines						
Attn-only	1.76	92.60	90.84	0.30	94.67	94.37
MLP-only	0.94	28.44	27.50	2.82	27.66	24.84
Norm-only	0.26	57.87	57.61	0.10	54.59	54.49
Gradient-based						
Gradient	0.64	29.94	29.30	0.02†	56.00	55.98
AtP	0.01†	107.73†	107.72†	**0.00**	104.50†	104.50†
Contextual mixing-based						
IFR	0.91	32.31	31.40	0.13	29.81	29.68
Proposed Method						
Layer-wise Target Propagation (LTP)	**0.00**	**143.76**	**143.76**	**0.00**	**172.15**	**172.15**

Table 14: Component-level attribution performance on **Mistral-7B-Instruct-v0.3** across **Known 1000** (factual knowledge) and **IOI** (Indirect Object Identification) benchmarks. Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful localization of causally relevant components; the *total* score aggregates both effects. **Bold** denotes the best result, and † indicates the second best. Layer-wise Target Propagation consistently achieves the most faithful component localization across benchmarks.

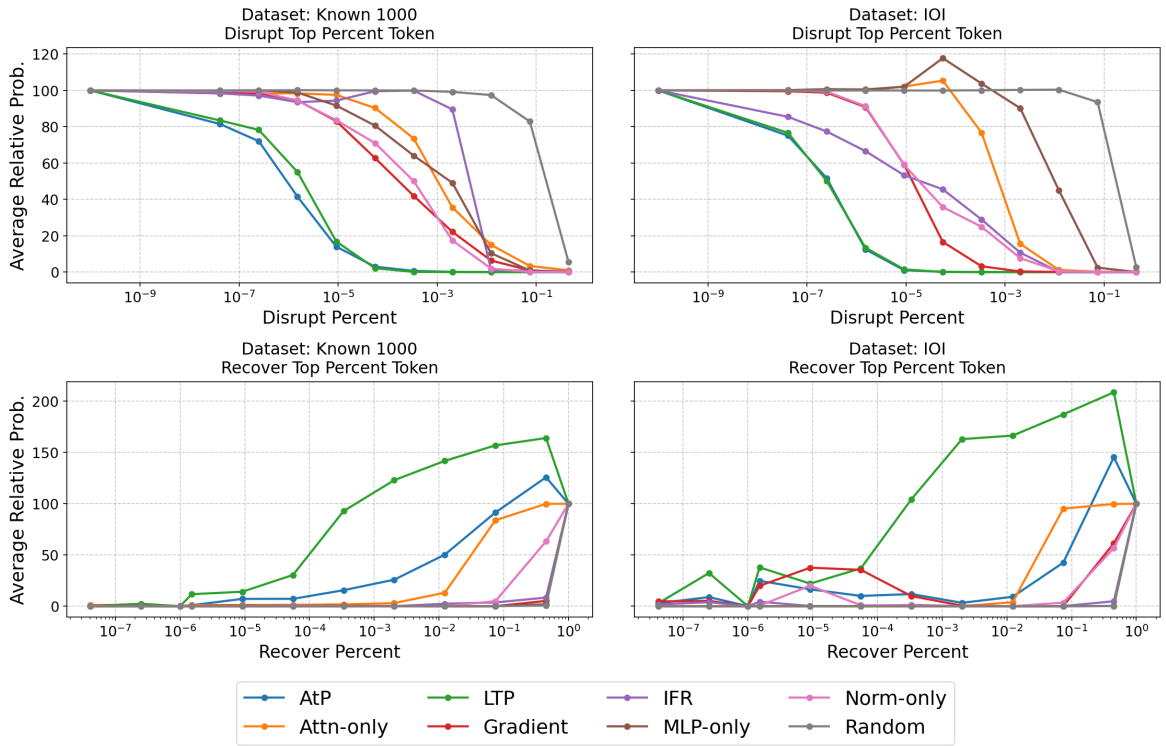

Figure 13: Performance comparison with baseline attribution methods on **Mistral-7B-Instruct-v0.3** under different top model component masking strategies. Our approach shows lower Top-k disruption and higher Top-k recovery, indicating more accurate identification of important components.

Methods	Known 1000		
	dis. ↓	rec. ↑	total ↑
Reference			
Random	10.31	18.19	7.88
Attention-based			
Last layer	8.95	20.05	11.09
Mean	8.80	20.87	12.07
Rollout	11.97	12.08	0.12
Gradient-based			
Gradient	8.87	16.21	7.34
Input×Gradient	8.72	16.30	7.57
Integrated Gradients	8.29	18.34	10.05
Decomposition-based			
DePass	8.11†	21.16†	13.05†
Contextual mixing-based			
IFR	11.84	12.43	0.59
Proposed Method			
Layer-wise Target Propagation (LTP)	**5.58**	**23.81**	**18.23**

Table 15: Faithfulness evaluation of crucial token attribution on **GPT-2 Large** on the **Known 1000** (factual knowledge) benchmark. Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful token localization; *total* denotes their aggregate AUC score. **Bold** indicates the best result, and † the second best. Layer-wise Target Propagation achieves the highest faithfulness across all metrics.

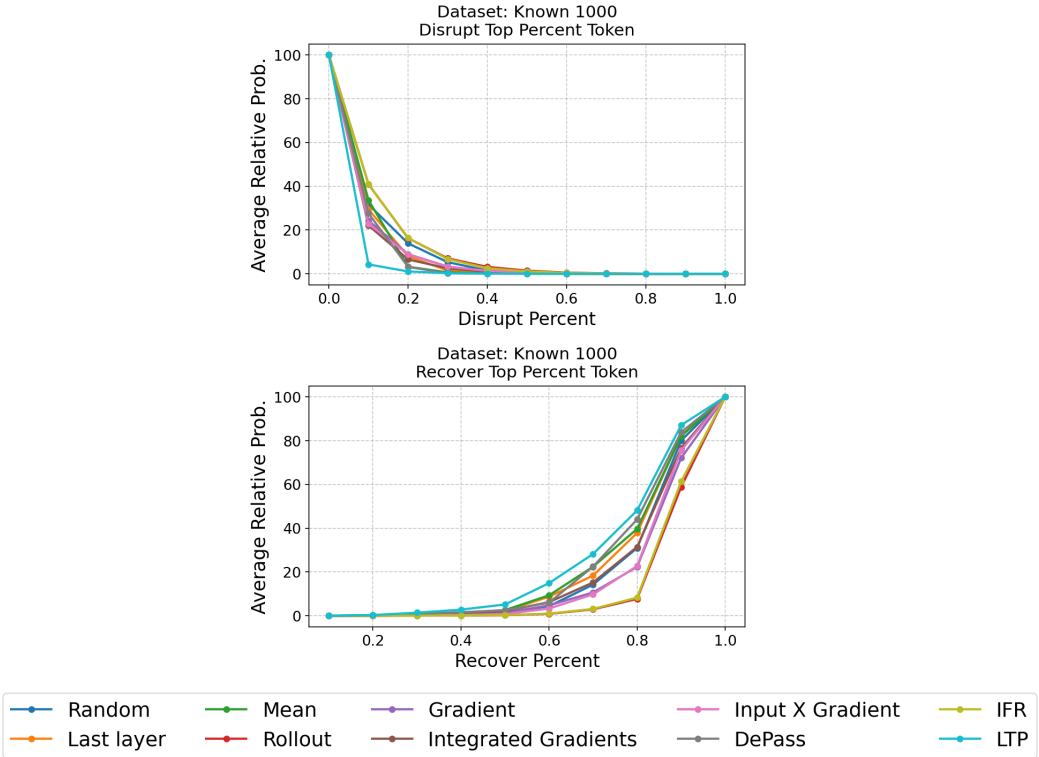

Figure 14: Performance comparison with baseline attribution methods on **GPT-2 Large** under different top input token masking strategies. Our approach shows *lower* Top-k disruption and *higher* Top-k recovery, indicating more accurate identification of important input tokens.

Methods	Known 1000			IOI		
	dis. ↓	rec. ↑	total ↑	dis. ↓	rec. ↑	total ↑
Reference						
Random	20.66	27.54	6.88	21.83	27.57	5.75
Baselines						
Attn-only	14.25	74.69	60.44	2.70	77.97	75.28
MLP-only	0.02	27.70	27.68	0.02	27.54	27.52
Norm-only	0.01†	29.75	29.74	0.01†	28.55	28.54
Gradient-based						
Gradient	107.68	33.41	-74.26	51.18	28.98	-22.21
AtP	**0.00**	**230.65**	**230.65**	**0.00**	**142.03**	**142.03**
Contextual mixing-based						
IFR	0.39	28.11	27.72	0.03	28.06	28.03
Proposed Method						
Layer-wise Target Propagation (LTP)	**0.00**	191.11†	191.11†	**0.00**	112.31†	112.31†

Table 16: Component-level attribution performance for **GPT-2 Large** across **Known 1000** (factual knowledge) and **IOI** (Indirect Object Identification) benchmarks. Lower disruption (*dis.*) and higher recovery (*rec.*) indicate more faithful localization of causally relevant components; the *total* score aggregates both effects. **Bold** denotes the best result, and † indicates the second best. While AtP attains the higher recovery AUC, LTP matches its zero disruption and recovers the original prediction probability with the fewest components (cf. Figure 15).

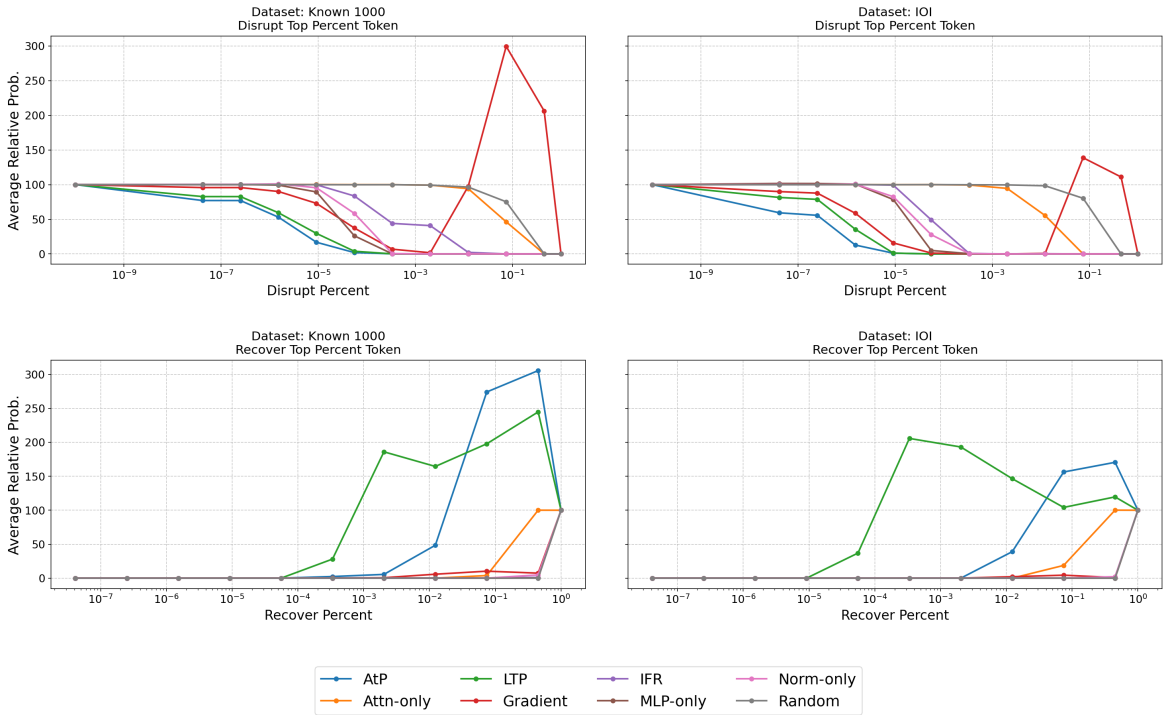

Figure 15: Performance comparison with baseline attribution methods on **GPT-2 Large** under different top model component masking strategies. LTP shows low Top-k disruption and the earliest Top-k recovery, indicating more accurate identification of important components.