CAUSAL IMPORTANCE DISTILLATION FOR TRANSFORMER MODULE PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing adaptive inference methods for Large Language Models rely on observational heuristics, such as hidden-state similarity or activation magnitudes, to drop redundant modules. However, these correlation-based metrics often fail to capture subtle, non-linear structural computations vital for semantic accuracy. We introduce CausalGate, an intervention-guided framework for compute-efficient transformer inference. During a calibration phase, CausalGate isolates individual Attention and MLP sub-layers, zeros out their respective outputs, and measures the resulting change in the predictive distribution via the Kullback-Leibler divergence of the final logit distribution. To eliminate runtime routing overhead, this intervention-derived importance hierarchy is distilled into a global set of static, lightweight scalar gates using an Exponential Moving Average smoothing objective paired with a differentiable pairwise ranking loss. We comprehensively evaluate CausalGate on TinyLlama-1.1B across language modeling and commonsense reasoning benchmarks, with comparisons against prominent routing and module-skipping baselines. Scaling experiments on Qwen2.5-3B and Llama-3.1-8B further evaluate perplexity under increasing module removal, while hardware measurements on TinyLlama-1.1B and Qwen2.5-3B demonstrate practical inference speedups.

1 INTRODUCTION

Ever since the seminal paradigm shift of "Attention Is All You Need," (Vaswani et al., 2017) scaling transformer architectures has unlocked unprecedented linguistic capabilities at a severe computational cost, making Large Language model (LLM) inference a critical deployment bottleneck. To mitigate this, model compression methods such as structural pruning and sub-layer skipping (He et al., 2025) seek to permanently deactivate low-utility blocks, avoiding the heavy hardware execution overhead and branch prediction penalties characteristic of runtime-dynamic routing frameworks like CALM (Schuster et al., 2022) or Mixture-of-Depths (MoD) (Raposo et al., 2024). Existing structural pruning metrics typically rely on observational proxies, such as activation magnitudes or hidden-state similarity, to determine if a module can be bypassed. However, such observational metrics can occasionally overlook subtle yet critical computational steps. A hidden-state vector traversing a module might exhibit minimal geometric displacement, yet that module may still execute a non-linear "causal refinement" vital for factual accuracy or logical coherence. By relying primarily on representational characteristics rather than direct structural necessity, heuristic-driven static pruning risks bypassing quietly essential modules, leading to sharp performance degradation at higher compute savings.

To overcome the limitations of observational proxies, we look to classical scientific inquiry: to truly determine whether a component is essential to a system, one must actively intervene upon it rather than merely observe it. We introduce CausalGate, an offline structural calibration framework establishing a permanent module-selection hierarchy through direct, input-agnostic structural interventions. Instead of estimating utility from runtime representation alignments, CausalGate isolates individual Attention and MLP modules during an offline calibration phase, zeros out their outputs, and measures the resulting Kullback-Leibler (KL) divergence (D_{KL}) in the final token distribution. This approach is governed by a central ***hypothesis:*** *a module's true global computational importance is explicitly defined by its causal influence on the final logit distribution.* By operationalizing

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

this principle at the sub-layer level, CausalGate bypasses heuristic guesswork and directly quantifies the exact semantic damage incurred by skipping any given block.

A key architectural distinction of CausalGate is its fine-grained operation at the sub-layer module level rather than the monolithic transformer-layer level. Multi-Head Attention and MLP modules perform distinct functions: attention routes context across sequences, whereas MLPs act as localized key-value (KV) memories. Independently pruning these submodules therefore provides greater computational flexibility than uniform layer dropping. However, calculating causal interventions at runtime is prohibitive. To achieve zero-overhead inference, CausalGate distills the structural importance hierarchy discovered during calibration into global, static, learned scalar gates. We optimize these gates using an Exponential Moving Average (EMA) smoothing objective to filter intervention noise, paired with a differentiable pairwise ranking loss that forces gate values to strictly preserve the discovered causal ordering.

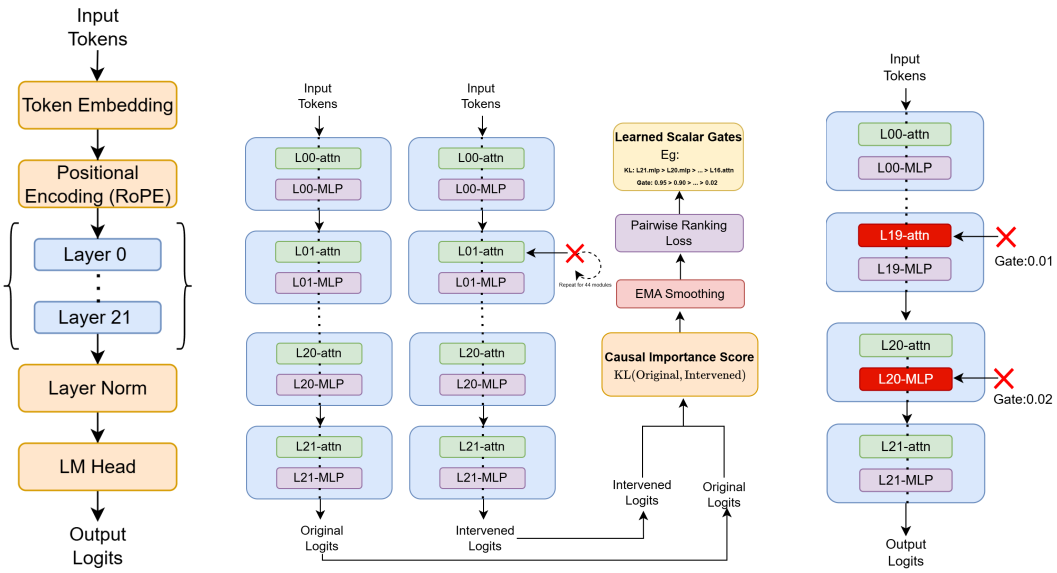
This optimization strategy bridges causal analysis and hardware execution efficiency, encoding global structural knowledge into a deterministic inference mask. Because these functional sublayers perform decoupled tasks, the model establishes a sequence-agnostic importance hierarchy operating at compile-time. Consequently, CausalGate eliminates runtime routing overhead, ensuring straight-line inference execution that pairs exceptionally well with localized memory boundaries and standard hardware loops. In summary, our contributions are as follows:

1. We introduce a novel approach to structural model pruning that replaces traditional observational heuristics with a ground-truth importance hierarchy derived directly from structural causal interventions during calibration.
2. We demonstrate that evaluating and skipping Attention and MLP modules independently yields a significantly more flexible and precise compute-performance tradeoff than uniform layer-dropping strategies.
3. We propose an optimization framework utilizing EMA target smoothing and a pairwise ranking loss to successfully encode structural causal hierarchies into global, static gates, enabling zero-overhead inference.
4. We conduct extensive evaluations against prominent baselines, demonstrating that CausalGate preserves language-model quality substantially better at larger compute reductions.

2 Related Work

Adaptive Inference and Layer Skipping. Adaptive inference dynamically adjusts computation based on input difficulty or latency bounds. Early mechanisms in Convolutional Neural Networks leveraged filter pruning, quantization (Zhang et al., 2024), early exiting (Han et al., 2021; Bolukbasi et al., 2017), and dynamic channel gating (Bejnordi et al., 2019; Gao et al., 2018; Lin et al., 2017). In Vision Transformers, frameworks like DynamicViT (Rao et al., 2021) and A-ViT (Yin et al., 2022) focus on dynamic token reduction, while AdaViT (Meng et al., 2022) adaptively drops attention heads or entire blocks per sample. For Large Language Models (LLMs), dynamic computational routing has expanded into token-skipping policies such as Mixture-of-Depths (MoD) (Raposo et al., 2024) and conditional early-exiting schemes (Rotem et al., 2023). Closer to our work, structural alternatives focus on bypassing redundant segments via uniform layer skipping (Liu et al., 2024; Luo et al., 2025), input-output similarity heuristics (He et al., 2025), reinforcement learning routing (Liu et al., 2025), or training-infused residual routing gates (Laitenberger et al., 2026; Nair et al., 2026). However, these approaches predominantly introduce dynamic control-flow branching or rely on shallow observational proxies during runtime. In contrast, we shift the execution boundary entirely to compile-time by utilizing offline interventional distillation to generate zero-overhead, static module masks.

Structural Pruning. Static network compression permanently drops underutilized weights or channels. For LLMs, post-training pruning often prioritizes channel groups, layers, or individual weights based on raw magnitudes (Frantar et al., 2022), weight-activation products like Wanda (Sun et al., 2024), or structural activation variances. To compress architectures structurally, frameworks like LLM-Pruner (Ma et al., 2023) and Sheared LLaMA (Xia et al., 2024) utilize first-order gradient approximations to isolate redundant coupled components for removal. Similarly, static sensitivity



(a) TinyLlama backbone.

(b) Intervention-based causal importance estimation and gate learning.

(c) Compute-efficient inference by skipping low-gate modules.

Figure 1: Proposed intervention-guided module selection framework. Each attention and MLP module is independently evaluated by zeroing its output; the resulting final-logit KL divergence defines its causal importance score. EMA smoothing and pairwise ranking losses distill these signals into learned scalar gates, allowing low-utility modules to be skipped at inference under a target compute budget.

analysis leverages Taylor-series expansions or zero-shot similarity metrics, such as Block Influence in ShortGPT (Men et al., 2025), to map module importance. While related, these alternatives rely strictly on observational correlations or localized gradients. CausalGate fundamentally differs by operationalizing active interventional calculus (*do*-operations) at a decoupled Attention and MLP sub-layer granularity to isolate true downstream semantic shifts.

Complementary Acceleration Paradigms. Beyond layer pruning, the literature contains orthogonal strategies to alleviate LLM inference bottlenecks. Memory footprints are frequently managed via compressed or pyramidal KV caches (Ge et al., 2024; Xiao et al., 2025; Cai et al., 2024), or via uniform mixed-precision weight and activation quantization frameworks (Xiao et al., 2023; Lin et al., 2024; Frantar et al., 2022; Park et al., 2024). Concurrently, speculative and aggressive decoding paradigms (Leviathan et al., 2023; Stern et al., 2018; Sun et al., 2021) accelerate autoregressive generation by predicting multiple tokens in parallel via distinct draft verification branches. These methodologies remain fully complementary to CausalGate and can be applied jointly to maximize hardware deployment efficiency.

3 Methodology

Our key intuition is that modules whose removal substantially alters the output distribution should be preserved, whereas modules with negligible influence can be skipped with minimal impact on model quality.

Let a transformer model consist of L layers, each containing an attention submodule and an MLP submodule. We treat each attention and MLP block as an independent computational unit, yielding a set of M modules $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$. For TinyLlama-1.1B, this corresponds to $M = 44$ modules (22 attention and 22 MLP modules). Our framework consists of three stages, as illustrated in Figure 1: (1) intervention-based causal importance estimation, (2) distillation of causal importance into learned module gates, and (3) compute-efficient inference through module selection.

3.1 CAUSAL MODULE IMPORTANCE ESTIMATION

We estimate the importance of each module through intervention analysis. Given an input sequence x, we first obtain the output distribution of the original model, $p(y \mid x)$. Next, we intervene on a module m_i by zeroing its output activation while leaving all remaining modules unchanged. The resulting model produces an intervened output distribution $p(y \mid x, m_i \to 0)$. The causal importance of module m_i is quantified as the KL divergence between the intervened and original output distributions:

$$\Delta_i^{(t)} = D_{\mathrm{KL}}\Big(p(y \mid x, m_i \to 0) \parallel p(y \mid x)\Big), \tag{1}$$

where t denotes the current training iteration. A large value of $\Delta_i^{(t)}$ indicates that intervening on module m_i substantially alters the model's predictive distribution, suggesting that the module plays an important role in generation. Conversely, a small value implies that the module has relatively little influence on the final prediction. We compute intervention scores independently for every attention and MLP module in the network. The resulting set of scores forms a module-level causal importance map that characterizes the relative contribution of each transformer module to the model's predictive behavior.

3.2 CAUSAL IMPORTANCE DISTILLATION

Although intervention scores provide a principled estimate of module importance, they can exhibit substantial variability across training iterations. Consequently, directly using raw intervention scores for module selection can lead to unstable rankings and suboptimal compute-allocation decisions. To address this issue, we distill intervention-derived causal importance into a set of learned module gates that capture a stable global ordering of module relevance. For each module m_i, we associate a learnable scalar gate

$$g_i = \sigma(\theta_i), \tag{2}$$

where θ_i is a trainable logit and $\sigma(\cdot)$ denotes the sigmoid function.

The gates directly scale their corresponding residual branches; for layer ℓ,

$$\tilde{\mathbf{h}}_\ell = \mathbf{h}_\ell + g_\ell^{\mathrm{attn}}\mathrm{Attn}_\ell(\mathrm{LN}(\mathbf{h}_\ell)), \qquad \mathbf{h}_{\ell+1} = \tilde{\mathbf{h}}_\ell + g_\ell^{\mathrm{mlp}}\mathrm{MLP}_\ell(\mathrm{LN}(\tilde{\mathbf{h}}_\ell)). \tag{3}$$

Thus, $\mathcal{L}_{\mathrm{LM}}$ backpropagates through the gates while the pretrained backbone remains frozen. Each gate is shared across all inputs and token positions, with higher values indicating modules preferentially preserved during inference.

EMA Target Aggregation. Intervention-derived importance estimates can vary across training iterations, resulting in unstable module rankings. To obtain more reliable supervision, we first normalize the intervention scores using the maximum module importance in the current iteration:

$$\tilde{z}_i^{(t)} = \frac{\Delta_i^{(t)}}{\max_j \Delta_j^{(t)} + \epsilon}, \tag{4}$$

where ϵ is a small constant for numerical stability. A target floor is then applied to prevent low-importance modules from collapsing to zero:

$$z_i^{(t)} = \alpha + (1-\alpha)\tilde{z}_i^{(t)}, \tag{5}$$

where α denotes the target floor. The final target is updated using an EMA:

$$\bar{z}_i^{(t)} = \beta \bar{z}_i^{(t-1)} + (1-\beta) z_i^{(t)}, \tag{6}$$

where $\beta \in [0, 1)$ is the EMA decay factor. The EMA targets are initialized as $\bar{z}_i^{(0)} = z_i^{(0)}$. The learned gates are then trained to match the EMA-smoothed targets using a mean-squared error objective:

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{M}\sum_{i=1}^{M}\Big(g_i - \bar{z}_i^{(t)}\Big)^2, \tag{7}$$

where M denotes the total number of modules.

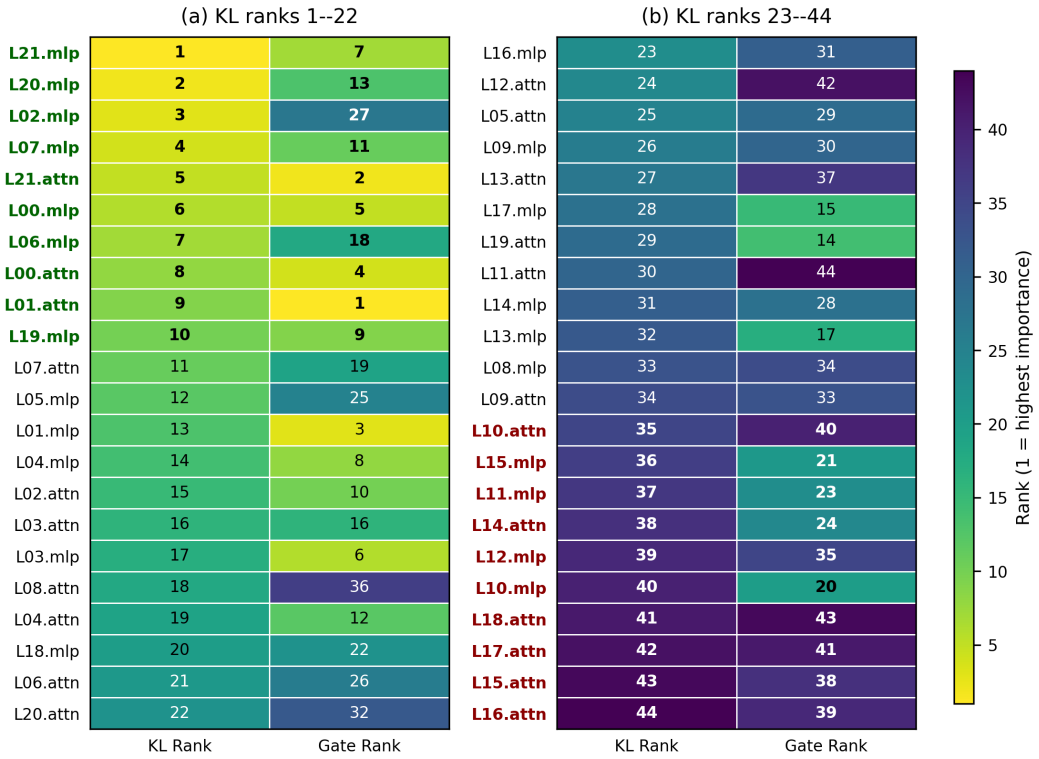


Figure 2: Comparison between intervention-derived causal importance rankings and the rankings learned by CausalGate.

Pairwise Ranking Supervision. While the absolute magnitude of intervention scores may vary, the relative ordering between modules is often more informative for module selection. We therefore introduce a pairwise ranking objective that encourages the learned gates to preserve the causal ordering induced by the EMA-smoothed targets. For a sampled module pair (m_i, m_j), we define

$$s_{ij} = \operatorname{sign}\left(\bar{z}_i^{(t)} - \bar{z}_j^{(t)}\right), \tag{8}$$

where tied pairs are ignored. The pairwise ranking loss is then

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max\left(0,\, \gamma - s_{ij}(g_i - g_j)\right), \tag{9}$$

where $\mathcal{P}$ is the set of sampled non-tied module pairs and γ is the ranking margin. This objective penalizes pairs whose gate ordering is reversed or whose correctly ordered gate values are separated by less than the margin γ. The sparsity regularizer encourages smaller gate activations and is defined as

$$\mathcal{L}_{\text{sparse}} = \frac{1}{M} \sum_{i=1}^{M} g_i^2. \tag{10}$$

The final gate-training objective combines language modeling, sparsity, causal regression, and ranking supervision:

$$\mathcal{L} = \lambda_{\text{lm}}\mathcal{L}_{\text{LM}} + \lambda_{\text{sparse}}\mathcal{L}_{\text{sparse}} + \lambda_{\text{causal}}\mathcal{L}_{\text{MSE}} + \lambda_{\text{rank}}\mathcal{L}_{\text{rank}}. \tag{11}$$

3.3 CausalGate Inference

After training, the learned gate values provide a global ranking of module importance that can be directly leveraged for compute-efficient inference. Let $\mathcal{G} = \{g_1, g_2, \ldots, g_M\}$ denote the set of

Table 1: Performance comparison across progressive module-removal compute budgets on TinyLlama-1.1B. Lower perplexity is better (↓), while higher accuracy is better (↑). All baselines are evaluated at matched architectural savings tiers.

Budget	Method	WikiText PPL↓	C4 PPL↓	HellaSwag↑	PIQA↑	CSQA↑	WinoGrande↑
0%	TinyLlama-1.1B (Ref)	13.58	9.89	0.484	0.707	0.367	0.540
5%	**CausalGate (Ours)**	16.00	10.84	0.445	0.723	**0.359**	**0.540**
	ShortGPT	14.84	**10.67**	0.480	0.711	0.348	0.530
	CALM (Softmax)	32.33	20.85	0.383	0.664	0.320	0.524
	CALM (Hidden-State)	23.11	28.97	0.453	**0.734**	0.313	0.540
	MoD	25.04	16.00	**0.484**	0.648	**0.359**	0.516
	GateSkip	**14.71**	16.12	0.461	0.684	0.313	**0.540**
	AdaSkip	27.08	26.42	0.406	0.633	0.273	0.510
	Act-Norm (Zero-Shot)	25.26	19.59	0.359	0.606	0.301	0.520
10%	**CausalGate (Ours)**	**18.46**	**12.84**	0.410	**0.715**	0.332	0.506
	ShortGPT	15.75	11.66	0.469	0.707	0.328	0.548
	CALM (Softmax)	71.96	48.35	0.383	0.648	0.297	0.522
	CALM (Hidden-State)	46.43	70.59	0.391	0.641	0.281	**0.538**
	MoD	60.29	40.75	0.406	0.605	**0.336**	0.492
	GateSkip	22.43	27.34	**0.441**	0.613	0.293	0.518
	AdaSkip	40.40	43.75	0.410	0.602	0.246	0.514
	Act-Norm (Zero-Shot)	81.11	161.86	0.262	0.512	0.234	0.490
20%	**CausalGate (Ours)**	**67.00**	**40.00**	0.340	**0.610**	0.260	0.490
	ShortGPT	71.27	43.69	0.345	0.566	0.254	0.504
	CALM (Softmax)	198.00	141.00	0.350	**0.610**	**0.300**	0.510
	CALM (Hidden-State)	80.00	130.00	**0.370**	0.590	0.250	**0.520**
	MoD	407.00	269.00	0.310	0.510	0.260	**0.520**
	GateSkip	81.00	100.00	0.320	0.550	0.260	0.480
	AdaSkip	1816.00	531.00	0.246	0.554	0.200	0.490
	Act-Norm (Zero-Shot)	2781.60	1085.00	0.273	0.535	0.231	0.510
30%	**CausalGate (Ours)**	**116.00**	**69.00**	**0.340**	0.540	0.250	0.490
	ShortGPT	118.16	68.43	0.289	0.586	0.219	0.500
	CALM (Softmax)	601.00	422.00	0.290	**0.600**	**0.290**	0.500
	CALM (Hidden-State)	185.00	299.00	0.290	**0.600**	0.230	**0.520**
	MoD	1313.00	990.00	0.250	0.530	0.240	0.510
	GateSkip	365.00	491.00	0.257	0.520	0.250	0.470
	AdaSkip	2064.00	623.00	0.238	0.539	0.170	0.490
	Act-Norm (Zero-Shot)	3563.20	3385.30	0.238	0.535	0.176	0.500
40%	**CausalGate (Ours)**	1365.00	**441.00**	**0.280**	0.500	0.200	0.480
	ShortGPT	**1200.25**	504.88	0.270	0.516	**0.254**	0.488
	CALM (Softmax)	1574.00	1047.00	**0.280**	0.530	0.250	**0.500**
	CALM (Hidden-State)	1472.00	712.00	0.270	**0.570**	0.220	0.490
	MoD	4433.00	3403.00	0.260	0.510	0.180	**0.500**
	GateSkip	1698.00	2046.00	0.250	0.490	0.200	0.460
	AdaSkip	4260.00	4509.00	0.235	0.527	0.160	0.460
	Act-Norm (Zero-Shot)	20825.40	31094.00	0.254	0.527	0.207	0.488

learned gate values for the M transformer modules. Modules are sorted according to their gate values, producing an ordered importance ranking $g_{(1)} \geqslant g_{(2)} \geqslant \cdots \geqslant g_{(M)}$, where larger gate values correspond to modules with higher intervention-derived importance. Given a target module-removal ratio ρ, the number of modules to suppress is $S = \mathrm{round}(\rho M)$, and the number retained is

$$K = M - \mathrm{round}(\rho M). \tag{12}$$

The top-K modules according to the learned ranking are preserved, while the remaining S modules are skipped during inference. Formally, the binary inference mask for module m_i is defined as

$$b_i = \begin{cases} 1, & \text{if } g_i \in \mathrm{TopK}(\mathcal{G}, K), \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

where $b_i = 1$ indicates that the module remains active and $b_i = 0$ indicates that the module is skipped.

4 Experiments

We evaluate whether intervention-guided module ranking can identify computationally important transformer modules while preserving language modeling and reasoning performance under reduced compute budgets. Specifically, we seek to answer the following questions:

- Can CausalGate retain model quality while disabling low-importance modules?
- How does CausalGate compare against existing compute-reduction baselines?

- Does intervention-derived supervision produce a meaningful ranking of module importance?

4.1 Experimental Setup

CausalGate Setup. All experiments are conducted using TinyLlama-1.1B, which consists of 22 transformer layers and 44 computational modules (22 attention and 22 MLP submodules). Gates are trained using AdamW with a learning rate of 0.01, gradient accumulation of 8 steps, and a maximum of 1000 optimization steps. We employ an EMA decay factor of $\beta = 0.9$, target floor $\alpha = 0.25$, ranking margin $\gamma = 0.05$, and 128 sampled ranking pairs per iteration. The loss weights are set to $\lambda_{\text{LM}} = 1.0$, $\lambda_{\text{sparse}} = 0.001$, $\lambda_{\text{causal}} = 10.0$, and $\lambda_{\text{rank}} = 2.0$. All experiments, including baseline evaluations, were conducted on a single NVIDIA Tesla V100 GPU with 32 GB of memory. The hyperparameter configuration was selected using a one-factor-at-a-time (OFAT) sensitivity analysis, in which each hyperparameter was varied independently while all remaining settings were fixed. The complete sensitivity analysis and additional experimental results are provided in the Appendix (Table 3).

Baselines. We compare CausalGate against representative compute-reduction strategies spanning adaptive-depth inference, dynamic routing, learned gating, and heuristic module selection. The full TinyLlama-1.1B model serves as the no-compression reference. We include two CALM variants (Schuster et al., 2022) based on confidence-driven early exiting: (i) Softmax Margin, which exits when prediction confidence exceeds a calibrated threshold, and (ii) Hidden-State Saturation, which exits when consecutive hidden representations become sufficiently similar. To evaluate dynamic depth allocation, we implemented a Mixture-of-Depths (MoD) (Raposo et al., 2024) routing baseline that selectively bypasses computations based on token-level routing decisions. We further compare against GateSkip (Laitenberger et al., 2026), a learned gating approach that trains lightweight gating functions to suppress low-importance computations, and AdaSkip (He et al., 2025), a heuristic module-selection strategy that ranks modules according to input-output similarity. We additionally implement ShortGPT (Men et al., 2025), adapting its Block Influence criterion to the Attention/MLP submodule granularity used by CausalGate for matched module-removal budgets. To further validate the precise structural contribution of our proposed intervention framework over basic observational methods, we also implement a zero-shot Activation-Norm Pruning (ActNorm) baseline. This approach executes a forward pass over the calibration data to measure the absolute magnitude of the activations flowing out of each attention and MLP module. Formally, let $\mathbf{H}_i \in \mathbb{R}^{T \times d}$ represent the output activation matrix of module m_i for a sequence of length T. The zero-shot importance metric $I_{\text{zero}}(m_i)$ is computed as the expected magnitude across the token dimension:

$$I_{\text{zero}}(m_i) = \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{H}_{i,t}\|_2^2 \tag{14}$$

Modules yielding the lowest I_{zero} scores are permanently masked out to satisfy the target compute reduction budget ρ. Collectively, these baselines represent diverse approaches for reducing transformer computation through early exiting, dynamic routing, learned gating, and structural pruning.

Datasets. Evaluation is performed on both language modeling and commonsense reasoning benchmarks. For language modeling, we report perplexity on WikiText-2, the Penn Treebank (PTB), and the C4 corpus. For downstream reasoning, we evaluate on HellaSwag, PIQA, CommonsenseQA (CSQA), and WinoGrande using standard multiple-choice accuracy. These benchmarks collectively assess the ability of compute-reduction methods to preserve language modeling performance, commonsense reasoning, and multiple-choice question answering accuracy under reduced computation budgets.

4.2 Main Results

Table 1 compares CausalGate against recent inference optimization baselines under 5%, 10%, 20%, 30% and 40% module-removal budgets, respectively. Performance is evaluated across language modeling tasks (WikiText-2 and C4 perplexity) and downstream commonsense reasoning benchmarks (HellaSwag, PIQA, CommonsenseQA, and WinoGrande). To characterize model robustness

Model	Removal	Latency (s)	Tok/s	Speedup
TinyLlama-1.1B	0%	2.100	60.9	1.00×
TinyLlama-1.1B	10%	1.954	65.5	1.08×
TinyLlama-1.1B	20%	1.747	73.3	1.20×
Qwen2.5-3B	0%	3.980	32.2	1.00×
Qwen2.5-3B	10%	3.641	35.2	1.09×
Qwen2.5-3B	20%	3.393	37.7	1.17×

Table 2: Inference efficiency of CausalGate under increasing module-removal budgets.

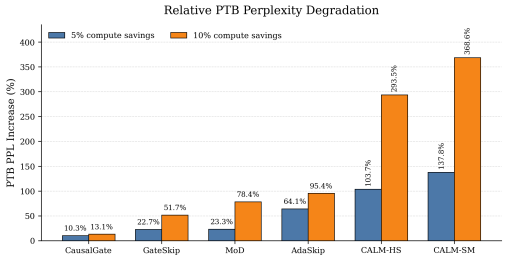


Figure 3: Relative perplexity degradation on the PTB benchmark under approximately 5% and 10% module-removal budgets.

under progressive compute reduction, Figure 3 reports the relative perplexity degradation on PTB across increasing budgets.

To assess practical inference efficiency, Table 2 reports end-to-end latency, throughput, and hardware speedups obtained by applying the learned module rankings. These measurements confirm that theoretical compute savings translate into measurable runtime improvements on physical hardware. Finally, to evaluate scalability beyond TinyLlama-1.1B, we apply CausalGate to Qwen2.5-3B-Instruct and Llama-3.1-8B-Instruct. Figure 4 summarizes the resulting WikiText-2 and C4 perplexities across module-removal budgets ranging from 0% to 20%, demonstrating the efficacy of our intervention-guided module ranking on larger transformer backbones.

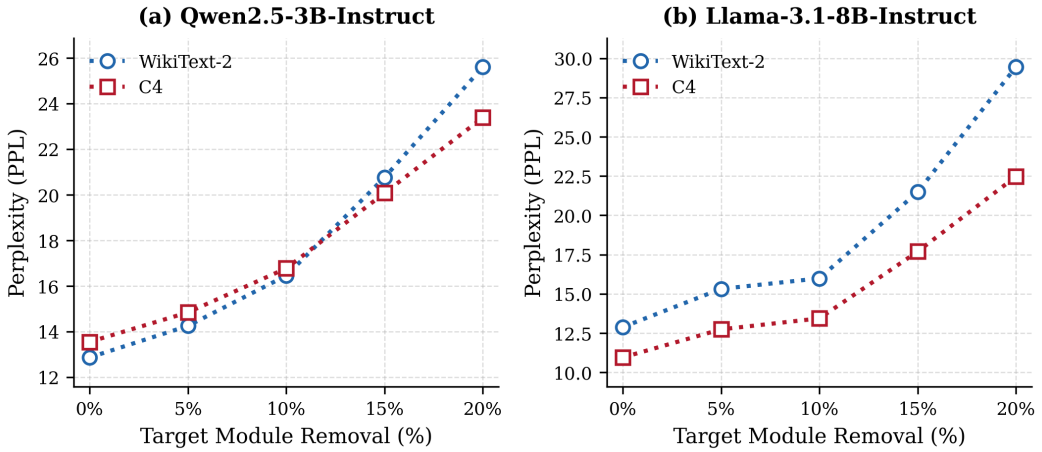


Figure 4: Scaling behavior of CausalGate on Qwen2.5-3B and Llama-3.1-8B under increasing module-removal budgets.

5 Discussion

Non-Uniform Causal Importance. The intervention-derived causal importance scores are highly non-uniform across transformer modules. While a small subset of modules induces substantial changes in the output distribution when intervened upon, the majority produce only negligible effects. In particular, several MLP submodules exhibit disproportionately large KL divergence values, indicating a dominant influence on the model's predictions, whereas many attention submodules, especially in later layers, contribute comparatively little. This concentration of causal importance suggests that transformer computation is inherently redundant and motivates selectively preserving highly influential modules during inference. A visualization of the learned causal importance distribution is provided in the Appendix (Figure 6).

Agreement with Independent Intervention Analysis. A key objective of CausalGate is to distill intervention-derived importance signals into a compact set of learned gate parameters. To independently assess whether the learned hierarchy reflects module importance in the fully active model, we perform an all-open intervention analysis in which all module gates are active and each Attention and MLP submodule is independently zeroed. Figure 2 compares the resulting KL-based ranking with the final ranking learned by CausalGate. The two rankings exhibit a Spearman correlation of 0.685 and a Pearson correlation of 0.411, with a top-10 overlap of $6/10$ and a bottom-10 overlap of $6/10$. This indicates substantial, but not perfect, agreement between the hierarchy learned through dynamic intervention-guided training and module importance measured independently in the fully active model. Importantly, this all-open analysis is not used as a post-training supervision signal, providing a separate assessment of the learned module ordering.

Impact on Compute Reduction. Table 1 summarizes performance across module-removal budgets from 5% to 40%. At low removal levels, several methods remain competitive, with ShortGPT achieving the lowest C4 perplexity at 5% and 10% removal. CausalGate becomes comparatively stronger at moderate removal levels, achieving the lowest WikiText-2 and C4 perplexities at 20% removal and the lowest WikiText-2 perplexity at 30% removal, while downstream accuracies remain mixed across methods. At 30% removal, for example, CausalGate obtains a WikiText-2 perplexity of 116.0, compared with 365.0 for GateSkip and 1313.0 for MoD. The substantially larger degradation observed for Act-Norm at moderate-to-high removal levels further suggests that activation magnitude alone is an insufficient criterion for identifying removable modules. At 40% removal, performance degrades sharply across methods, indicating a regime in which suppressing a large fraction of submodules substantially compromises model quality. Overall, these results support the effectiveness of intervention-derived importance primarily in low-to-moderate module-removal regimes rather than under aggressive pruning. Figure 3 further reports relative perplexity degradation on PTB, while Figure 4 evaluates the behavior of CausalGate on Qwen2.5-3B and LLaMA-3.1-8B under progressive module removal. Finally, Table 2 shows that static module selection yields measurable improvements in throughput and latency without requiring runtime routing.

Limitations and Future Work. The proposed framework employs a global module ranking that is learned once and applied uniformly across all inputs. While this design enables efficient module selection and stable compute reduction, it does not account for input-specific variability in module importance. Consequently, modules that are unimportant on average may still be beneficial for particular inputs or tasks. Future work could extend CausalGate toward input-adaptive policies that dynamically select modules based on the characteristics of each input sequence. Although the observed trends are consistent across both language modeling and reasoning benchmarks, further validation on larger language models and diverse architectures is needed to assess the generality of intervention-derived module ranking. Future work may also investigate finer-grained causal gating strategies, including token-level and context-dependent module selection, as well as integration with runtime-aware execution frameworks for achieving practical inference acceleration.

6 Conclusion

In this work, we introduced CausalGate, an intervention-guided framework for compute-efficient transformer inference. Unlike existing approaches that rely on observational heuristics such as confidence estimates or representation similarity, CausalGate directly measures module importance through structural interventions and quantifies each module's causal influence on the model's output distribution. By distilling intervention-derived importance scores into lightweight scalar gates using EMA target aggregation and pairwise ranking supervision, the proposed method learns a stable hierarchy of module importance that enables efficient module selection without requiring intervention analysis at inference time. Experimental results on language modeling and commonsense reasoning benchmarks demonstrate that CausalGate consistently achieves a more favorable compute-performance trade-off than existing adaptive inference methods, particularly under increasingly restrictive compute budgets. These findings suggest that causal contribution provides a principled criterion for identifying redundant computation in LLMs and offer a promising direction toward more efficient and interpretable inference systems.

AI USE STATEMENT

Generative AI tools were used as assistive tools during the preparation of this work, primarily for language polishing, improving clarity and presentation, and assisting with drafting and editing portions of the manuscript. They were used for feedback on experimental presentation, methodology, assistance with debugging, and visualization. The research idea, methodological development, experimental design, selection of models, datasets, baselines, and evaluation settings, as well as the analysis and interpretation of the experimental results, were carried out by the authors. All AI-assisted content and code were reviewed and verified by the authors, who take full responsibility for the final manuscript, code, figures, results, and claims.

ETHICS STATEMENT

This work studies compute-efficient inference and structural module importance in pretrained language models and does not involve human subjects, personally identifiable information, or the collection of sensitive data. Our experiments use publicly available language modeling and reasoning benchmarks. The proposed method modifies model execution by identifying and skipping low-importance transformer modules and does not introduce new data collection or user profiling mechanisms. We do not identify significant direct ethical risks arising from the methodology presented in this work. The authors are responsible for the integrity, reproducibility, and reporting of all experiments and results.

REPRODUCIBILITY STATEMENT

Section 3 specifies the intervention score, gate-training objective, and module-selection rule. Section 4 reports models, dataset splits and sample counts, calibration procedures, baselines, training settings, random seeds, precision, and hardware. We report realized module-removal fractions alongside target budgets, since the number of modules is discrete. The appendix includes implementation details, sensitivity analyses, and supplementary results. Code, configurations, and evaluation scripts are provided in the supplementary material.

REFERENCES

Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. *arXiv preprint arXiv:1907.06627*, 2019.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International conference on machine learning*, pp. 527–536. PMLR, 2017.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Xitong Gao, Yiren Zhao, Robert Mullins, Cheng-zhong Xu, et al. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *International Conference on Learning Representations*, volume 2024, pp. 22975–22988, 2024.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(11): 7436–7456, 2021.

Zhuomin He, Yizhen Yao, Pengfei Zuo, Bin Gao, Qinya Li, Zhenzhe Zheng, and Fan Wu. Adaskip: Adaptive sublayer skipping for accelerating long-context llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24050–24058, 2025.

Filipe Laitenberger, Dawid Kopiczko, Cees G Snoek, and Yuki Asano. What layers when: Learning to skip compute in llms with residual gates. In *International Conference on Learning Representations*, volume 2026, pp. 78161–78184, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

Fangxin Liu, Junjie Wang, Ning Yang, Zongwu Wang, Junping Zhao, Li Jiang, and Haibing Guan. Aster: Adaptive dynamic layer-skipping for efficient transformer inference via markov decision process. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 11853–11861, 2025.

Yijin Liu, Fandong Meng, and Jie Zhou. Accelerating inference in large language models with a unified layer skipping strategy. *arXiv preprint arXiv:2404.06954*, 2024.

Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained llms. *arXiv preprint arXiv:2503.23798*, 2025.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12309–12318, 2022.

Kiran Nair, Rodrigue Rizk, and KC Santosh. A spike-gated residual unit for information flow control in transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3469–3478, June 2026.

Gunho Park, baeseong park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 38069–38086, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/a4f98ce85f440ee269b0df57b4368719-Paper-Conference.pdf.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Daniel Rotem, Michael Hassid, Jonathan Mamou, and Roy Schwartz. Finding the sweet spot: Analysis and improvement of adaptive inference in low resource settings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14836–14851, 2023.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5937–5947, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 5385–5409, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/160adf2dc118a920e7858484b92a37d8-Paper-Conference.pdf.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5c1ddd2e59df46fd2aa85c833b1b36ed-Paper-Conference.pdf.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10809–10818, 2022.

Xian Zhang, Guoqing Xiao, Mingxing Duan, Yuedan Chen, and Kenli Li. Appq-cnn: An adaptive cnns inference accelerator for synergistically exploiting pruning and quantization based on fpga. *IEEE Transactions on Sustainable Computing*, 9(6):874–888, 2024.

Appendix

A Additional Experimental Analysis

A.1 Hyperparameter Sensitivity

A.2 Performance Under Progressive Module Removal

A.3 Dataset Summary

A.4 Module-Level Intervention Analysis

B Algorithm

C An Interventional Causal Interpretation of CausalGate

D Do Single-Module Intervention Signals Remain Informative Under Joint Pruning?

E Learned Gates vs. Direct EMA-Target Ranking

A Additional Experimental Analysis

A.1 Hyperparameter Sensitivity

We examine the sensitivity of CausalGate to the principal hyperparameters used during gate optimization. Each hyperparameter is varied independently while the remaining settings are held fixed. Table 3 reports the resulting Spearman rank correlation and WikiText-2 perplexity. The selected configuration provides a favorable balance between preserving the intervention-derived module ordering and maintaining language-modeling performance.

Hyperparameter	Value	Spearman ↑	WikiText PPL ↓	Selected
Learning Rate	0.001	0.6879	20.19	
	0.005	0.6765	21.15	
	0.010	0.6849	18.46	✓
	0.020	0.6810	18.52	
EMA β	0.70	0.679	18.60	
	0.80	0.675	18.62	
	0.90	0.680	18.44	✓
	0.95	0.660	19.01	
Target Floor α	0.10	0.72	18.47	
	0.25	0.71	18.44	✓
	0.40	0.70	18.44	
Rank Margin γ	0.01	0.62	18.48	
	0.05	0.68	18.43	✓
	0.10	0.65	18.45	
Rank Pairs	32	0.68	18.45	
	64	0.68	18.45	
	128	0.68	18.45	✓
	256	0.68	18.45	
λ_{causal}	1	0.54	20.10	
	5	0.56	19.70	
	10	0.60	18.48	✓
	20	0.60	18.90	
λ_{rank}	0	0.43	19.20	
	1	0.57	19.20	
	2	0.68	18.46	✓
	3	0.66	18.56	
Training Steps	500	0.70	21.15	
	1000	0.68	18.40	✓
	1500	0.69	18.45	

Table 3: Hyperparameter sensitivity analysis.

A.2 Performance Under Progressive Module Removal

Figure 5 examines how model performance changes as the fraction of active transformer modules is progressively reduced. Accuracy is reported on HellaSwag, CommonsenseQA, and PIQA, together with language-modeling perplexity. Performance remains relatively stable under limited module removal but degrades increasingly as fewer modules remain active, illustrating the trade-off between structural reduction and predictive performance.

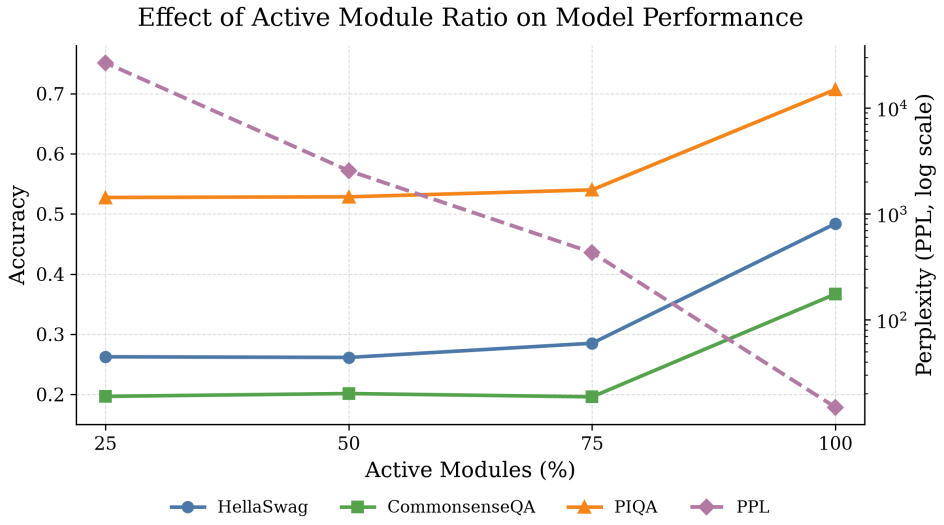


Figure 5: Effect of reducing the fraction of active transformer modules. Accuracy is reported for HellaSwag, CommonsenseQA, and PIQA, while perplexity is shown on a logarithmic scale.

A.3 Dataset Summary

Table 4 summarizes the datasets used for training and evaluation. WikiText-2 is used for gate training and language-modeling evaluation, while C4 provides an additional language-modeling benchmark. HellaSwag, PIQA, CommonsenseQA, and WinoGrande are used exclusively for downstream evaluation across commonsense, physical, and question-answering tasks.

Table 4: Summary of datasets used for training and evaluation.

Dataset	**Usage**	**Task**	**License**
WikiText-2	Training + Evaluation	Language Modeling	CC BY-SA 3.0
C4	Evaluation	Language Modeling	ODC-BY 1.0
HellaSwag	Evaluation	Commonsense Reasoning	MIT
PIQA	Evaluation	Physical Reasoning	AFL-3.0
CommonsenseQA	Evaluation	Question Answering	MIT
WinoGrande	Evaluation	Commonsense Reasoning	CC-BY
Penn Treebank (PTB)	Evaluation	Language Modeling	LDC

A.4 Module-Level Intervention Analysis

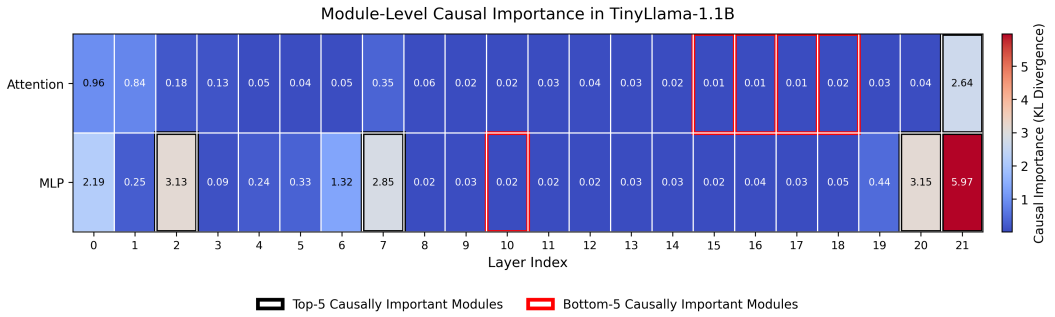


Figure 6: Module-level importance scores obtained through intervention analysis. Each cell reports the average KL divergence induced by zeroing the corresponding Attention or MLP submodule. Higher values indicate modules whose removal causes larger changes in the model's output distribution.

Figure 6 visualizes the module-level importance scores obtained from the independent intervention analysis. The scores are strongly non-uniform across the transformer: several MLP submodules produce large changes in the predictive distribution when suppressed, whereas many other Attention and MLP modules have substantially smaller effects. The analysis therefore reveals a heterogeneous distribution of intervention sensitivity across layers and module types.

B Algorithm

Algorithm 1 Intervention-Based Module Importance Estimation

Require: Transformer f, module set $\mathcal{M} = \{m_i\}_{i=1}^{M}$, current gates $\mathbf{g} = \{g_i\}_{i=1}^{M}$, input batch x

Ensure: Module-level intervention scores $\{\Delta_i\}_{i=1}^{M}$

Run the current soft-gated model to obtain $\ell_{\text{ref}} \leftarrow f(x; \mathbf{g})$

Compute final-token distribution $p_{\text{ref}}(y_T|x; \mathbf{g})$

for each module $m_i \in \mathcal{M}$ **do**

  Zero the residual-branch output of m_i, keeping all other gates unchanged

  Run the intervened model to obtain logits $\ell_{\text{int},i}$

  Compute final-token distribution $p_{\text{int},i}(y_T|x; \mathbf{g})$

  Compute the single-module intervention score:

$$\Delta_i = D_{\text{KL}}\left(p_{\text{int},i}(y_T|x; \mathbf{g}) \parallel p_{\text{ref}}(y_T|x; \mathbf{g})\right)$$

end for

return $\{\Delta_i\}_{i=1}^{M}$

Algorithm 2 CausalGate Distillation and Module Selection

Require: Frozen transformer f, training data $\mathcal{D}$, module set $\mathcal{M}$
Require: Target floor α, EMA decay β, ranking margin γ, ranking pairs R
Require: Loss weights $\lambda_{\text{LM}}, \lambda_{\text{sparse}}, \lambda_{\text{causal}}, \lambda_{\text{rank}}$
Ensure: Learned gate ranking and selected modules under target budget ρ

Initialize one scalar gate logit $\theta_i \leftarrow 0$ for each module m_i
Set $g_i \leftarrow \sigma(\theta_i)$
Freeze all backbone parameters of f
for training step $t = 1, \ldots, T$ **do**
    Sample batch $x \sim \mathcal{D}$
    Compute intervention scores $\{\Delta_i^{(t)}\}_{i=1}^{M}$ using Algorithm 1
    Normalize scores:

$$\tilde{z}_i^{(t)} = \frac{\Delta_i^{(t)}}{\max_j \Delta_j^{(t)} + \epsilon}$$

    Apply target floor:

$$z_i^{(t)} = \alpha + (1 - \alpha)\tilde{z}_i^{(t)}$$

    if $t = 1$ **then**
        Initialize EMA target $\bar{z}_i^{(t)} \leftarrow z_i^{(t)}$
    else
        Update EMA target:

$$\bar{z}_i^{(t)} = \beta \bar{z}_i^{(t-1)} + (1 - \beta) z_i^{(t)}$$

    end if
    Update gate values $g_i \leftarrow \sigma(\theta_i)$
    Compute causal regression loss:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{M} \sum_{i=1}^{M} \left(g_i - \bar{z}_i^{(t)} \right)^2$$

    Sample R random non-tied module pairs $\mathcal{P}$
    Define $s_{ij} = \text{sign}(\bar{z}_i^{(t)} - \bar{z}_j^{(t)})$
    Compute ranking loss:

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \max\left(0,\, \gamma - s_{ij}(g_i - g_j)\right)$$

    Compute sparsity loss:

$$\mathcal{L}_{\text{sparse}} = \frac{1}{M} \sum_{i=1}^{M} g_i^2$$

    Compute total loss:

$$\mathcal{L} = \lambda_{\text{LM}} \mathcal{L}_{\text{LM}} + \lambda_{\text{sparse}} \mathcal{L}_{\text{sparse}} + \lambda_{\text{causal}} \mathcal{L}_{\text{MSE}} + \lambda_{\text{rank}} \mathcal{L}_{\text{rank}}$$

    Update only gate logits $\{\theta_i\}_{i=1}^{M}$ using AdamW
end for
Rank modules by learned gate values in descending order
For target removal ratio ρ, compute $S = \text{round}(\rho M)$ and $K = M - S$
Keep the top-K modules and suppress the remaining S modules
return learned gate ranking and selected module set

C An Interventional Causal Interpretation of CausalGate

Although CausalGate is implemented entirely within a transformer, its module-ranking strategy can be interpreted through the lens of interventional causal reasoning. Rather than estimating module

importance from observational statistics such as activation magnitudes or hidden-state similarity, CausalGate explicitly suppresses individual residual-branch contributions and measures the resulting change in the model prediction.

Let $\mathbf{h}_\ell$ denote the residual-stream state entering transformer layer ℓ, and let A_ℓ and F_ℓ denote its Attention and MLP branches, respectively. Absorbing normalization into the branch functions for notational simplicity, the residual updates are

$$\tilde{\mathbf{h}}_\ell = \mathbf{h}_\ell + A_\ell(\mathbf{h}_\ell), \tag{15}$$

$$\mathbf{h}_{\ell+1} = \tilde{\mathbf{h}}_\ell + F_\ell(\tilde{\mathbf{h}}_\ell), \tag{16}$$

with the final predictive distribution given by $P(Y \mid X) = g(\mathbf{h}_L)$.

CausalGate intervenes on a residual-branch contribution while preserving the incoming residual stream. For example, suppressing the Attention or MLP branch of layer ℓ gives

$$\tilde{\mathbf{h}}_\ell^{\text{int}} = \mathbf{h}_\ell, \tag{17}$$

$$\mathbf{h}_{\ell+1}^{\text{int}} = \tilde{\mathbf{h}}_\ell, \tag{18}$$

respectively, while all other modules remain unchanged. The intervention then propagates through the downstream residual computation graph and produces an intervened predictive distribution. Modules whose suppression induces larger distributional changes are assigned greater intervention-derived importance.

Figure 7 illustrates this interpretation. CausalGate therefore measures module influence through explicit residual-branch interventions rather than observational proxies alone.

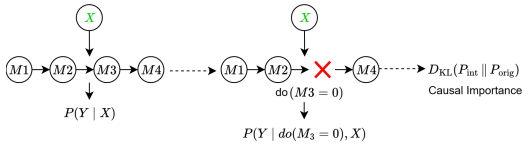


Figure 7: Interventional interpretation of CausalGate. An intervention suppresses the residual-branch contribution of a selected Attention or MLP module while preserving the incoming residual stream and all other modules. The resulting change in the predictive distribution provides the intervention-derived supervision signal for module importance.

D Do Single-Module Intervention Signals Remain Informative Under Joint Pruning?

CausalGate estimates module importance using single-module interventions, where one module is suppressed while the remaining modules are left unchanged. At inference, however, multiple low-ranked modules are removed simultaneously. Since module effects need not compose additively, an important question is whether a hierarchy learned from marginal intervention signals remains informative under joint removal.

To examine this, we jointly suppress the bottom-k modules selected by CausalGate and measure the KL divergence between the resulting predictive distribution and that of the all-open model. We compare against two controls: (i) directly selecting the bottom-k modules according to their independent all-open intervention KL scores (Individual-KL), and (ii) 100 uniformly sampled random module subsets of the same size. All measurements use the same 32 WikiText-2 examples employed in the independent intervention analysis.

Table 5 shows that direct Individual-KL ranking produces lower joint perturbation at the smallest removal budgets ($k = 2$ and $k = 4$). At larger joint-removal budgets, however, CausalGate produces lower joint KL than directly sorting the individual intervention scores. At 20% and 30% removal, none of the 100 sampled random subsets achieves a lower joint KL than the CausalGate-selected subset; at 40% removal, only 8% do so.

These results do not imply that CausalGate explicitly models higher-order module interactions, nor that single-module intervention effects compose additively. Rather, they provide empirical evidence that the hierarchy learned from intervention-derived supervision can remain informative when multiple modules are removed jointly.

Table 5: Joint-intervention analysis on TinyLlama-1.1B. Joint KL measures the distributional shift between the jointly pruned model and the all-open model. Individual-KL directly removes the modules with the lowest independent all-open intervention scores. Random Med. is the median joint KL over 100 uniformly sampled module subsets of matched size. CG %ile denotes the percentage of random subsets with lower joint KL than CausalGate; lower is better.

Removal	k	CausalGate	Individual-KL	Random Med.	CG %ile
5%	2	0.049	**0.030**	0.382	19
10%	4	0.151	**0.068**	1.533	10
20%	9	**0.153**	0.843	3.984	0
30%	13	**1.473**	2.935	5.723	0
40%	18	**3.690**	4.594	6.163	8

E Learned Gates vs. Direct EMA-Target Ranking

We compare the learned CausalGate ranking against directly ranking modules by their final EMA-smoothed intervention targets. Since the original EMA vector was not retained, we conduct a dedicated controlled run in which both rankings are obtained from and evaluated on the same checkpoint. The two rankings are related but distinct, with Spearman correlation $\rho = 0.754$, Pearson correlation $r = 0.516$, and bottom-10 overlap of 8/10 modules. Figure 8 compares both rankings across increasing module-removal levels. At 10% removal, CausalGate outperforms EMA-Only across all six evaluated metrics. At 20%, CausalGate achieves lower C4 perplexity and higher accuracy on all four reasoning benchmarks, while EMA-Only obtains lower WikiText-2 perplexity. At 5%, performance is mixed. Under aggressive 30–40% removal, EMA-Only preserves language-modeling perplexity more effectively, while reasoning results remain mixed. These results indicate that learned gates do not simply reproduce the EMA ordering and can provide useful ranking refinements in the low-to-moderate removal regime. However, they do not uniformly outperform direct EMA ranking, particularly under aggressive module removal.

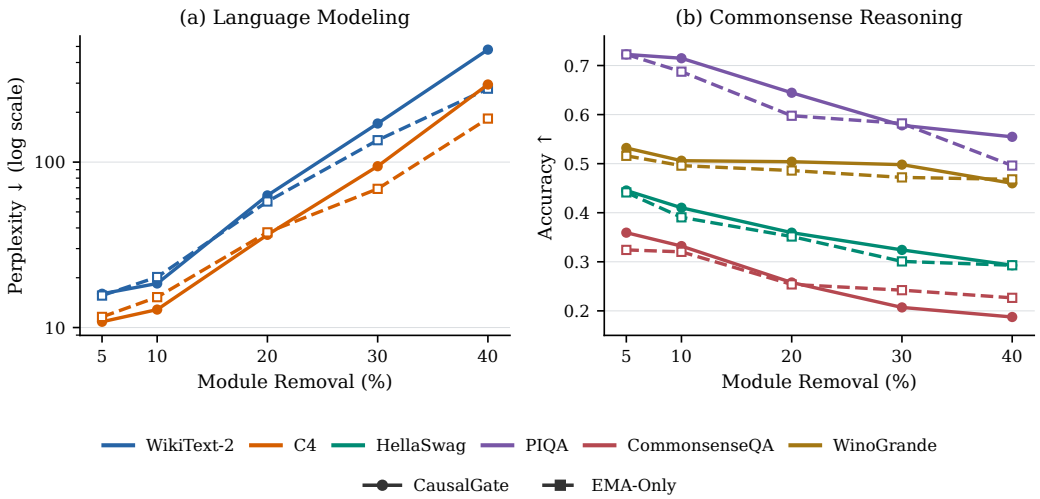


Figure 8: **Learned CausalGate ranking versus direct EMA-target ranking.** Both rankings are obtained from the same dedicated training run and evaluated on the same checkpoint. **Left:** WikiText-2 and C4 perplexity under increasing module removal (lower is better). **Right:** HellaSwag, PIQA, CommonsenseQA, and WinoGrande accuracy (higher is better). Learned gates provide their clearest advantage at 10% module removal, whereas direct EMA ranking preserves perplexity more effectively under aggressive 30–40% removal.