

Sememe: Causal Modeling for Task-Aware Depth Pruning of Large Language Models

Charles Yu & Heng Ji

Department of Computer Science
University of Illinois Urbana-Champaign
{ctyu2, hengji}@illinois.edu

Abstract

Large Language Models (LLMs) exhibit strong generalist capabilities, yet task-specific behavior is often supported by diffuse and redundant structures. While pruning has been widely explored as a means to reduce inference costs, few existing methods translate theoretical sparsity into tangible speedups without sacrificing commensurate performance. We introduce **Sememe**, a task-aware depth-pruning method that frames block-importance estimation as causal inference, treating block removals as interventions in a Structural Causal Model (SCM) and estimating importance scores as path coefficients from per-example calibration losses, with a provable unbiasedness guarantee. Sememe identifies redundancy using a small calibration set and only the forward pass, avoiding the memory overhead of gradients or activations. Across Llama2, Llama3.1, Mistral, and Qwen3 models, Sememe with 32 calibration examples matches or exceeds the downstream retention that other training-free depth-pruning methods reach with 256-512 examples, while remaining stable where other methods degenerate. Sememe’s interventional formulation naturally extends to joint multi-block removals and to task-aware calibration, which prunes toward a target skill from a small set of in-task examples. At larger scales, we find this depth pruning to be highly effective: Sememe retains 93.3% of downstream accuracy when removing 20% of Llama3.1-70B’s layers, with speedups proportional to the fraction of removed layers on any hardware. ¹

1 Introduction

Modern Large Language Models (LLMs) achieve broad general knowledge and reasoning capabilities, but their scale imposes substantial inference costs in latency, memory, and energy. While the generalist ability of LLMs is valuable in open-ended settings, many deployment scenarios such as single-purpose tools in agentic workflows, on-device applications, and domain-specific services, require only a subset of the model’s skills, leaving significant capacity unused. This observation has motivated extensive work on pruning (LeCun et al., 1989; Han et al., 2015; Frantar & Alistarh, 2023; Sun et al., 2024); however, most methods operate below the layer level (individual weights or semi-structured patterns) and therefore cannot translate sparsity into proportional inference speedups without specialized hardware or runtime support (Liu et al., 2023).

Depth Pruning sidesteps this problem by removing entire transformer layers. Because the blocks are physically removed, the pruned model is natively faster on any hardware, with speedups proportional to the fraction pruned (Song et al., 2024; Men et al., 2024; Kim et al., 2024; Shukor & Cord, 2024). The central challenge is determining which layers to remove. Depth pruning is coarser-grained than weight-level pruning, so erroneous removals are more costly and the importance of each layer depends on interactions with remaining layers. Existing approaches reduce all per-example information to a single scalar for determining importance, discarding variance structure across examples.

¹Our code is available at <https://github.com/CharlesYu2000/SEMEME>.

We propose **Sememe**, a depth-pruning method that reframes the layer importance estimation as causal inference. Sememe treats each transformer layer as an exogenous variable and the task loss as the endogenous outcome in a Structural Causal Model (SCM). By fitting the SCM’s structural equation to the full matrix of per-example losses under one-at-a-time block-removal interventions, Sememe jointly estimates the causal contribution of every remaining block in a single regression.

Our primary variant, **Sememe-I** fits a linear structural equation via Ordinary Least Squares (OLS). We also explore a counterfactual extension, **Sememe-C**, that models the full loss distribution. The overall framework is illustrated in Figure 1.

Our primary contributions are as follows:

1. We formalize depth pruning as causal inference within an SCM. **Sememe-I** estimates block importance via OLS on a linear structural equation. We prove unbiasedness under our interventional design (Proposition 4.3) and justify the linear approximation despite the non-linearity of transformers (§4.5). We also explore a counterfactual variant **Sememe-C** (§4.3).
2. We show that Sememe consistently outperforms training-free depth-pruning baselines across Llama-family models (7B - 13B) on standard benchmarks at 20-25% sparsity, while approaching methods that add post-training or learned modules (Table 2). A task-specific variant, Sememe-I-Skill, further improves performance by pruning toward the target task.
3. We demonstrate that Sememe’s per-example causal modeling is especially advantageous in low-resource settings: with as few as 32 calibration samples, Sememe-I matches the downstream accuracy and perplexity of other depth-pruning methods that use 8 - 16 times more calibration examples.

2 Preliminaries

We briefly review the causal inference concepts underlying Sememe. For a more thorough discussion, see Pearl (2000) and Kline (2016).

2.1 The Causal Hierarchy

Pearl (2000) stratifies causal reasoning into three layers of increasing inferential power. **Association** (Layer 1, seeing) identifies patterns in observations, represented by the conditional probability $P(y | x)$. **Intervention** (Layer 2, doing) predicts the effect of external actions by forcing a variable to a particular value to isolate the variable’s pure impact, formalized by the *do*-operator $P(y | do(x))$ (outside of definitions, we omit the *do*-operator from discussion for brevity). **Counterfactual** (Layer 3, imagining) involves retrospective reasoning about alternative scenarios conditioned on what was actually observed: “what would Y have been if X had been x' instead of x ?” Each layer strictly subsumes the prior; most existing pruning methods operate at Layer 1, while Sememe operates at Layers 2 and 3.

2.2 Structural Causal Models and Path Analysis

A **structural causal model** (SCM) consists of a directed acyclic graph and a set of **structural equations**, one per variable, that specify how each variable is determined by its parents and an exogenous disturbance term (Pearl, 2000). We adopt the terminology of Kline (2016): **exogenous variables** are determined from outside the model, their causes are not represented, and they are free to vary and covary; **endogenous variables** have at least one cause represented within the model, each with an associated **disturbance** ζ , an unmeasured exogenous variable representing all omitted causes plus measurement error; and **path coefficients** are the parameters of the structural equations, representing the magnitude of direct effects between variables.

Given a structural equation $Y = \sum_i \gamma_i X_i + \zeta$, the path coefficients γ_i are *identified* (uniquely estimable from data) when the design provides enough independent constraints. In interventional systems where the researcher controls the values of the exogenous variables, identification is straightforward: the exogenous variables are independent of the disturbance by construction and ordinary least squares yields unbiased estimates of the path coefficients (Kline, 2016).

3 Related Work

Model pruning long predates LLMs and is often based in recognizing patterns from gradient and activation signals (LeCun et al., 1989; Hassibi et al., 1993).

Weight-Level/Semi-Structured Pruning: Han et al. (2015) first showed that pruning could simply look at weight magnitudes to determine importance. Other unstructured pruning methods (Paul et al., 2022; Gadhikar et al., 2023; Liu et al., 2023) similarly mask out weights at the individual level, allowing for fine-grained identification of importance. SparseGPT (Frantar & Alistarh, 2023) demonstrated that GPT-scale models could be pruned in a single pass using approximate second-order information. Wanda (Sun et al., 2024) further simplified this by pruning weights based on the product of their magnitudes and the norm of input activations, avoiding the computational expense of Hessian inverses. Wanda++ (Yang et al., 2025) subsequently improved on this by incorporating regional gradients at the decoder-block level.

Block-Level (Channel/Layer/Depth) Pruning: To circumvent the need for hardware optimizations, some lines of research pivoted toward removing entire components. Channel pruning (Yu et al., 2022; Lee & Hwang, 2024) aims to eliminate entire rows and columns of matrices. Ding et al. (2025) identifies redundant contiguous layers and combines the parameters together to compress. Lu et al. (2025) uses gradient-based optimization to remove contiguous redundant layers and finetunes to replace the lost information. LLM-Pruner (Ma et al., 2023) uses gradient information to find coupled neurons then post-trains with LoRA to recover. ShortenedLlama (Kim et al., 2024) uses a Taylor approximation to aggregate neuron output importance scores. ShortGPT (Men et al., 2024) and LLM-Streamline (Chen et al., 2025a) both identify similar layers to compress by comparing input and output activations while the latter also replaces the redundant layers with a lighter-weight trained network. LinearPatch (Chen et al., 2025b) replaces pruned transformer layers with a linear layer to scale activations to match pre-pruned activations.

Closest in methodology and scope to our framework is SLEB (Song et al., 2024). SLEB uses a similar iterative procedure to prune blocks, determining at each step which transformer block to remove. While the SLEB algorithm is interventional in that it removes each transformer block to check if task performance degrades, it summarizes the calibration set by its mean loss before comparing blocks. In contrast, our method uses calibration samples to fit a structural causal model to the individual per-example outcomes. We compare the two under equivalent calibration budgets in §5.4.

4 Methodology

In this section, we formalize the Sememe framework: the block importance estimation problem (§4.1), **Sememe-I** (Intervention), which estimates importance via interventional regression (§4.2), **Sememe-C** (Counterfactual), which estimates importance via counterfactual simulation (§4.3), a task-aware configuration **Sememe-I-Skill** (§4.4), and a discussion of the linear approximation for transformers (§4.5).

4.1 Problem Formulation - Block Importance

Let $\mathcal{M}$ be a transformer with an ordered sequence of T transformer layer “blocks” $\{b_1, \dots, b_T\}$, $\mathcal{X} = \{x_1, \dots, x_N\}$ be a calibration set, and $\mathcal{L}(\mathcal{M}, x)$ be a scalar loss function (e.g., cross-entropy) evaluated on input x using model $\mathcal{M}$. We write $\mathcal{M} \setminus b_i$ for the model

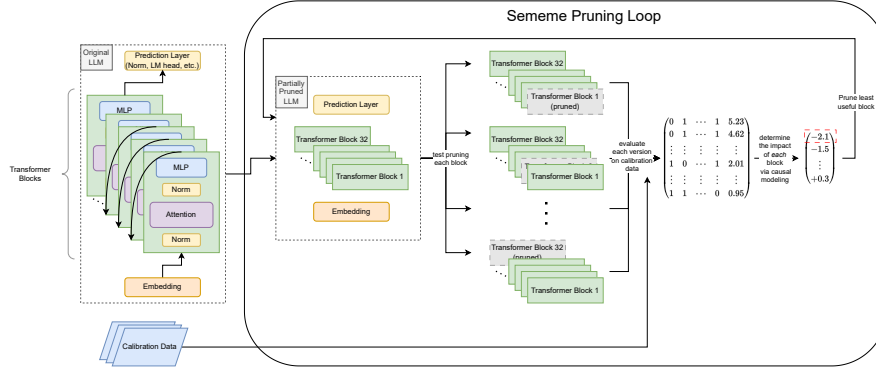


Figure 1: The Sememe pruning loop. Starting from the current (partially pruned) model $\mathcal{M}_{t-1}$, we: 1) construct K candidate models, each with one remaining block converted to a skip connection; 2) evaluate every candidate on the calibration set, populating the design matrix $\mathbf{D} \in \{0, 1\}^{NK \times K}$ and response vector $\mathbf{y} \in \mathbb{R}^{NK}$ (Eq. 2); 3) fit the structural equation via OLS to obtain path coefficients $\hat{\gamma}$ (Eq. 3); and 4) prune the block with the least negative $\hat{\gamma}_i^*$ (Eq. 5). The loop repeats until the target sparsity is reached.

obtained by converting block b_i to a skip connection (i.e., identity/no-op, mathematically equivalent to pruning it).

Definition 4.1 (Block Importance Scoring). A *block importance score* is a mapping $s: \{b_1, \dots, b_T\} \rightarrow \mathbb{R}$ such that a **lower** score $s(b_i)$ indicates that block b_i is safer to remove, in the sense that $\mathcal{L}(\mathcal{M} \setminus b_i)$ remains close to $\mathcal{L}(\mathcal{M})$. Given a target number of blocks to remove $n \leq T$, we seek a pruning order $\pi = (\pi_1, \dots, \pi_n)$ that minimizes cumulative loss degradation.

Greedy iterative procedure. Following Song et al. (2024), Sememe prunes blocks one at a time. At step $t \in \{1, \dots, n\}$, given the current (partially pruned) model $\mathcal{M}_{t-1}$, we:

1. Evaluate a block importance score $s_t(b_i)$ for every remaining block,
2. Select $\pi_t = \arg \min_i s_t(b_i)$ and prune it,
3. Set $\mathcal{M}_t = \mathcal{M}_{t-1} \setminus b_{\pi_t}$.

Sememe’s contribution lies in *how* s_t is computed: rather than comparing aggregated per-block losses (as in SLEB), we fit a causal model that jointly estimates the contribution of all remaining blocks from the full set of per-example loss observations, as described in the following sections. This overall flow is illustrated in Figure 1. The algorithm is also detailed in Appendix B.

4.2 Sememe-I: Interventional Importance

4.2.1 Structural Causal Model

At pruning step t , let $\mathcal{B}_t$ denote the set of $K = T - t + 1$ remaining (un-pruned) blocks. We construct an SCM (depicted in Figure 2) with the following components:

- **Exogenous variables** $B_i \in \{0, 1\}$ for each $b_i \in \mathcal{B}_t$, where $B_i = 1$ indicates block b_i is active and $B_i = 0$ indicates it has been test-pruned (converted to a skip connection). These variables are exogenous because their values are set by our experimental intervention, not determined by any process within the model.
- **Endogenous variable** $Y \in \mathbb{R}$, the scalar per-example task loss.
- **Structural equation:**

$$Y = \sum_{b_i \in \mathcal{B}_t} \gamma_i B_i + \zeta, \quad (1)$$

where $\gamma_i \in \mathbb{R}$ is the *path coefficient* for block b_i and ζ is the *disturbance*: an exogenous term representing all sources of loss variation not attributable to any single block's presence or absence (e.g., input difficulty, *inter-block interactions*). Note that ζ is not a regression residual, it is a structural term representing omitted causal factors.

4.2.2 Data

Collection via One-at-a-Time Intervention

To populate the design matrix, we perform controlled experiments: for each remaining block $b_i \in \mathcal{B}_t$, we intervene by setting $do(B_i = 0)$ (converting b_i to a skip connection) and evaluate the loss on every calibration example $x_j \in \mathcal{X}$. Each intervention produces one row per example:

$$\left(\underbrace{1, \dots, 1, 0, 1, \dots, 1}_{\text{one-cold indicator } \mathbf{d}_i \in \{0,1\}^K}, \mathcal{L}(\mathcal{M}_{t-1} \setminus b_i, x_j) \right), \quad (2)$$

where the zero appears at position i . Repeating over all K blocks and N calibration examples yields a design matrix $\mathbf{D} \in \{0,1\}^{NK \times K}$ and a response vector $\mathbf{y} \in \mathbb{R}^{NK}$.

4.2.3 Estimation via Ordinary Least Squares

We estimate the path coefficients by ordinary least squares (OLS):

$$\hat{\gamma} = (\mathbf{D}^\top \mathbf{D})^{-1} \mathbf{D}^\top \mathbf{y}. \quad (3)$$

Note that we fit *without* an explicit intercept: every row of $\mathbf{D}$ sums to $K - 1$, so a constant column is already in the column space of $\mathbf{D}$, and adding an intercept would introduce perfect multi-collinearity. The disturbance term ζ in Eq. (1) represents zero-mean noise, not a separate constant. Therefore, all intercepts are absorbed into the one-cold encoding. The SCM formulation thus provides two concrete benefits: (a) one-at-a-time interventions with the *do*-operator ensure exogeneity by construction, allowing us to use OLS in a standard way following Kline (2016), and (b) it distinguishes the structural disturbance ζ from a regression residual, ensuring that omitted interaction effects do not bias the importance estimates.

To make the path coefficients represent predicted impact on loss, we standardize:

$$\hat{\gamma}_i^* = \hat{\gamma}_i \cdot \frac{\text{sd}(B_i)}{\text{sd}(Y)}, \quad (4)$$

where $\text{sd}(\cdot)$ denotes the sample standard deviation computed over all NK observations. Note that $\text{sd}(B_i)$ acts as a uniform scalar and does not change ordering.

4.2.4 Interpretation and Pruning Decision

The path coefficient γ_i in Eq. (1) measures block b_i 's *contribution to the predicted loss* when it is active ($B_i = 1$). If including block b_i reduces the task loss (i.e., the block is useful), then $\gamma_i < 0$; removing it ($B_i: 1 \rightarrow 0$) increases the predicted loss by $|\gamma_i|$. Conversely, $\gamma_i \approx 0$ indicates a dispensable block. Thus, we define the importance score as $s_t(b_i) = -\hat{\gamma}_i^*$. The block selected for pruning at step t is therefore:

$$\pi_t = \arg \min_i s_t(b_i) = \arg \max_{b_i \in \mathcal{B}_t} \hat{\gamma}_i^*, \quad (5)$$

which selects the block whose removal is estimated to be least harmful.

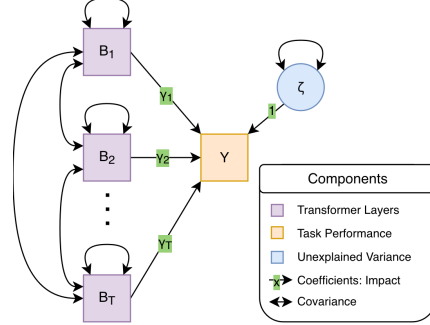


Figure 2: Reticular action model (RAM) symbolism diagram for the structural causal model in Eq. (1). Block indicators $B_1, \dots, B_K$ are exogenous while the task loss Y is endogenous. Path coefficients γ_i quantify each block's direct effect on Y with ζ being the disturbance term.

4.2.5 Theoretical Properties

Proposition 4.2 (Invertibility of the Design Matrix). *The matrix $\mathbf{D}^\top \mathbf{D}$ is positive definite (and hence invertible) for any $K \geq 2$ and $N \geq 1$.*

Proof in Appendix C. The system is heavily overidentified for any $N \geq 1$, trivially satisfying the classical Structural Equation Modeling (SEM) counting rule.

Proposition 4.3 (Unbiasedness under Intervention). *Under the one-at-a-time intervention design, $\hat{\gamma}$ from Eq. (3) is an unbiased estimator of the best linear predictor of Y given $(B_1, \dots, B_K)$, regardless of the true (potentially non-linear) relationship between block indicators and loss.*

Proof in Appendix D. Intuitively, since each B_i is set by experimental intervention ($\text{do}(B_i = 0)$) rather than merely observed, the indicators are exogenous to the disturbance ζ by construction. This severs all back-door paths in the causal graph (Kline, 2016), ensuring that $\mathbb{E}[\zeta \mid \mathbf{D}] = 0$ and satisfying the classical OLS exogeneity condition. Unbiasedness holds despite the true loss manifold being non-linear; it guarantees that $\hat{\gamma}_i$ is unbiased for the linear projection coefficient, which is sufficient for correctly ranking block importance (see §4.5).

4.3 Sememe-C: Counterfactual Importance

Sememe-I summarizes each block’s impact with a single linear coefficient. To investigate whether richer distributional modeling improves importance estimation, we propose **Sememe-C**, which models the full *distribution* of losses under each block removal. Rather than fitting a joint structural equation, Sememe-C fits a Gaussian mixture model (GMM) to the N per-example losses produced by each intervention $\text{do}(B_i = 0)$ and queries the fitted distribution to estimate expected loss under block removal. This corresponds to Layer 3 (counterfactual) inference in the causal hierarchy: we condition on observed calibration losses, fit a generative model, and ask what the loss *would have been* under a hypothetical pruning decision.

Sememe-C introduces $3K_{\text{mix}} - 1$ (K_{mix} is the number of mixture components) parameters per block (vs only 1 for Sememe-I), making it more expressive but also more data-hungry. The full GMM formulation, counterfactual query, and derivation are discussed in Appendix F.

4.4 Task-Aware Calibration: Sememe-I-Skill

Definition 4.1 requires no assumption about the calibration set, as importance is defined relative to the distribution that $\mathcal{X}$ is drawn from. Calibrating on generic text (e.g., WikiText-2) estimates importance for general language modeling, while calibrating on examples from a target task estimates importance for that task. We call this configuration **Sememe-I-Skill**, simply Sememe-I with $\mathcal{X}$ drawn from task data, with no change to the estimator, the interventions, or the pruning loop. This configuration implies that Sememe is task-aware, and it can do so even when task data is scarce as shown in §5.4.

4.5 Justification of the Linear Approximation to Non-linear Transformers

(i) First-order local approximation. Under our one-at-a-time intervention design, each observation differs from the all-active configuration in exactly one coordinate, so the first-order Taylor expansion is exact for each single-block query and the path coefficients recover the exact partial derivatives $\partial f / \partial B_i$ (see Appendix E). This exactness justifies the linear functional form for one-at-a-time queries. Beyond the functional form, the SCM formulation also provides (a) the guarantee of Proposition 4.3, which holds regardless of the true non-linear loss surface, and (b) generality: because importance is defined through interventions, the same estimator supports richer interventional extensions by design, such as joint multi-block removals (discussed further in §5.6). Empirically, the resulting procedure operates reliably at very small calibration sizes (§5.4).

(ii) Exogeneity eliminates confounding. Proposition 4.3 guarantees that OLS recovers the best linear predictor regardless of the true functional form. For *ranking* blocks, ordinal correctness is sufficient.

(iii) **Iterative re-estimation mitigates accumulated error.** Sememe re-fits the causal model at every pruning step, so the approximation is always local to the current model. As layers are removed and the LLM’s abilities change, Sememe adapts by collecting fresh data.

5 Experiments

5.1 Setup

Sememe is implemented in PyTorch using HuggingFace Transformers (Wolf et al., 2020) extending the SLEB codebase (Song et al., 2024). All experiments use a single NVIDIA H100 (80GB). Evaluations use the EleutherAI LM Evaluation Harness (Gao et al., 2023).

5.2 Efficiency

Sememe adds only the cost of solving the OLS system (Eq. (3)) or fitting the GMM, both computed on CPU with $\mathcal{O}(K)$ parameters. On Llama2-7B (32 transformer layers), Sememe takes 147 s wall-clock vs 135 s for SLEB (<9% overhead) to prune to 20% sparsity. The calibration forward passes account for essentially all wall-clock time in both methods, and pruning cost therefore scales with the calibration size N . Pruning at $N = 32$ requires roughly half the GPU time of $N = 256$ (Table 1).

METHOD	N	GPU (s)	CPU ESTIMATION (s)
SEMEME-I	32	748	0.016
	64	1200	0.017
	128	1379	0.019
	256	1493	0.020
SLEB	32	702	-
	64	1111	-
	128	1301	-
	256	1358	-

Table 1: Cost of pruning Llama3.1-8B to 50% sparsity (16 of 32 blocks) on a single H100, with calibration examples evaluated at batch size 1 in both implementations. GPU time covers the calibration forward passes, which dominate total cost and grow with N . The causal estimation step of Sememe is done on CPU and takes nearly no time.

At inference time, the pruned model is natively faster in proportion to the removed blocks, as they are physically eliminated from the computation graph. Prior work documents that inference speedup scales proportionally with the fraction of removed layers (Shukor & Cord, 2024; Song et al., 2024; Kim et al., 2024).

5.3 Performance on Downstream Tasks

Table 2 reports downstream accuracy across standard set of classification benchmarks following previous studies (Chen et al., 2025b; Lu et al., 2025; Song et al., 2024; Chen et al., 2025a; Cao et al., 2025) on Llama2-7B and Llama2-13B at 20% and 25% sparsity. We compare against training-free depth-pruning methods (ShortGPT, ShortenedLlama, SLEB), adapted methods that use post-training or added parameters (LLM-Pruner, LLM-Streamline, LinearPatch), and Wanda as a non-depth-pruning reference.

Training-free comparison. Among training-free methods, Sememe variants consistently rank at or near the top. On Llama2-7B at 20% sparsity, Sememe-C retains 84.1% of the dense model’s accuracy, outperforming other baselines. On Llama2-13B at 20%, the advantage is more pronounced: Sememe-C and Sememe-I retain 88.4% and 87.3% vs SLEB’s 83.6%. At 25% sparsity on Llama2-7B, Sememe-I retains 78.7% vs. SLEB’s 77.7%, with Sememe-C declining to 77.5%, consistent with the distributional overfitting trade-off discussed in §4.3.

On Llama2-13B at 25%, SLEB and Sememe-I identify the same total set of blocks (albeit in different orders). This type of reversal and convergence is driven by individual anomalous

MODEL	SPARSITY	METHOD	BENCHMARKS (ACC %)						$\mathcal{R}$
			HSWAG	WG	ARC-E	ARC-C	PIQA	AVG	
LLAMA2-7B	0%	DENSE	75.97	68.98	74.58	46.25	79.11	68.98	–
	20%	SHORTGPT	62.12	65.59	55.55	34.04	69.97	57.45	82.68
		SHORTENEDLLAMA	58.43	53.28	50.29	28.75	73.34	52.82	75.29
		SLEB	62.49	58.96	56.57	33.02	73.07	56.82	81.47
		SEMEME-I	62.79	58.88	56.82	33.02	73.45	56.99	81.69
		SEMEME-C	64.48	61.01	59.09	34.47	74.05	58.62	84.14
		LLM-PRUNER [†]	61.66	54.78	60.61	35.24	<u>75.41</u>	57.54	82.67
		LLM-STREAMLINE [†]	62.66	65.98	55.89	36.18	70.40	58.22	84.06
		LINEARPATCH [†]	63.49	65.90	61.24	<u>37.63</u>	70.46	59.74	86.33
		SEMEME-I-SKILL [†]	63.85	<u>67.48</u>	<u>61.74</u>	34.98	72.80	<u>60.17</u>	<u>86.46</u>
	25%	SHORTGPT	55.55	65.04	49.20	30.80	64.04	52.93	76.18
		SHORTENEDLLAMA	53.37	53.28	50.51	27.99	69.59	50.95	72.74
		SLEB	58.28	58.25	52.31	31.91	69.59	54.07	77.65
		SEMEME-I	58.31	57.30	56.02	32.42	69.97	54.80	78.70
		SEMEME-C	57.24	53.59	55.26	32.00	71.93	54.00	77.45
		LLM-PRUNER [†]	54.41	53.20	52.90	31.91	71.33	52.75	75.77
		LLM-STREAMLINE [†]	56.17	64.33	48.61	32.76	64.36	53.25	76.91
		LINEARPATCH [†]	57.67	<u>65.11</u>	55.22	33.45	67.46	55.78	80.39
		SEMEME-I-SKILL [†]	<u>58.80</u>	63.93	<u>58.25</u>	<u>33.87</u>	68.88	<u>56.75</u>	<u>81.70</u>
	50%	WANDA (2:4)	55.33	62.27	57.58	31.91	70.84	55.59	79.77
LLAMA2-13B	0%	DENSE	79.35	72.38	77.40	49.06	80.52	71.74	–
	20%	SHORTGPT	51.33	69.85	66.96	40.61	73.83	60.52	84.43
		SHORTENEDLLAMA	52.25	71.42	67.21	40.10	75.24	61.24	85.31
		SLEB	67.03	62.51	61.83	36.43	75.08	60.58	83.64
		SEMEME-I	70.90	65.75	65.66	37.29	76.66	63.25	87.25
		SEMEME-C	69.20	64.48	67.26	40.44	<u>77.42</u>	63.76	88.35
		LLM-PRUNER [†]	68.76	56.51	67.51	41.98	76.50	62.25	86.51
		LLM-STREAMLINE [†]	72.38	69.61	67.38	44.03	75.24	65.73	91.53
		LINEARPATCH [†]	<u>73.02</u>	69.38	<u>69.53</u>	<u>44.54</u>	75.68	<u>66.43</u>	<u>92.50</u>
		SEMEME-I-SKILL [†]	72.62	70.80	69.15	43.43	76.17	<u>66.43</u>	92.36
	25%	SHORTGPT	47.95	69.13	60.14	37.12	71.59	57.19	79.64
		SHORTENEDLLAMA	49.01	66.06	63.89	36.43	73.23	57.72	80.16
		SLEB	67.90	64.17	64.77	38.14	76.01	62.20	86.01
		SEMEME-I	67.90	64.17	64.77	38.14	76.01	62.20	86.01
		SEMEME-C	65.19	62.59	59.55	35.07	74.81	59.44	81.99
		LLM-PRUNER [†]	<u>72.24</u>	57.30	<u>67.26</u>	39.51	<u>77.91</u>	62.84	86.88
		LLM-STREAMLINE [†]	69.29	69.85	63.51	43.00	72.52	63.63	88.72
		LINEARPATCH [†]	70.15	69.61	65.53	<u>44.20</u>	73.83	64.66	90.21
		SEMEME-I-SKILL [†]	69.03	<u>70.40</u>	66.88	44.11	74.70	<u>65.02</u>	<u>90.67</u>
	50%	WANDA (2:4)	63.09	67.01	64.31	37.80	73.94	61.23	84.81

Table 2: Downstream accuracy (%) of pruned LLMs. The [†] marker indicates usage of task-specific data, post-training, and/or learned replacement layers. Methods with no marker use only generic calibration data and add no parameters or post-training. $\mathcal{R}$: percentage of dense-model accuracy retained. Best training-free result **bolded**; best overall underlined.

greedy decisions that cascade through subsequent pruning steps, a vulnerability shared by all iterative depth-pruning methods at higher sparsity levels.

Both Sememe variants at 20% also match or exceed Wanda (50% sparsity) despite removing far fewer parameters.

Adapted methods. Sememe-I-Skill (§4.4), which uses in-domain calibration data but performs no gradient-based training and adds no parameters, approaches or matches methods that do. On Llama2-7B at 20%, Sememe-I-Skill retains 86.5%, comparable to LinearPatch (86.3%) and exceeding LLM-Streamline (84.1%) and LLM-Pruner (82.7%). On Llama2-13B at 20%, Sememe-I-Skill ties LinearPatch at 92.4% retained accuracy. This suggests that task-aware calibration alone, without parameter modification, can recover much of the performance lost to block removal.

5.4 Calibration Efficiency

A key empirical advantage of Sememe is calibration efficiency, or how little calibration data is needed to reach a given pruning quality. We compare Sememe-I (with $N = 32$

calibration examples) against SLEB across calibration set sizes $N \in \{32, 128, 256, 512\}$, on both downstream accuracy and perplexity. Table 3 reports downstream accuracy at 20% sparsity for three architectures.

MODEL (20% SPARSITY)	SEMEME-I N=32	SLEB			
		N=32	N=128	N=256	N=512
MISTRAL-7B-v0.3	60.7 / 80.8	60.1 / 80.0	60.5 / 80.7	60.5 / 80.7	60.7 / 80.8
LLAMA3.1-8B	61.3 / 81.8	56.0 / 74.4	56.0 / 74.4	56.2 / 74.7	56.2 / 74.7
QWEN3-8B	59.6 / 82.8	57.9 / 80.3	58.7 / 81.4	59.6 / 82.8	59.6 / 82.8

Table 3: Calibration-size ablation on downstream accuracy (Avg / $\mathcal{R}$) at 20% sparsity.

Sememe-I with 32 examples matches or exceeds the accuracy SLEB reaches at $N = 256$ on Qwen3-8B, $N = 512$ on Mistral-7B, and $N = 512$ on Llama3.1-8B. Similar patterns hold for perplexity. Figure 3 plots the perplexity gap relative to Sememe-I at each pruning step on Llama3.1-8B: Sememe with 32 examples maintains lower perplexity than SLEB with 32-128 examples, while SLEB with 256 and 512 examples maintains only marginally lower perplexity (per-step values in Appendix G).

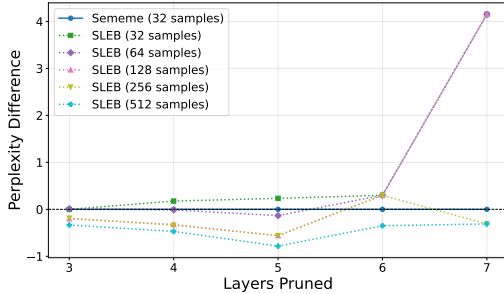


Figure 3: Perplexity difference relative to Sememe ($N = 32$) on Llama3.1-8B at each pruning step. Positive values indicate higher (worse) perplexity than Sememe. SLEB requires ≥ 256 examples to approach Sememe’s performance with 32.

Sememe-I’s own selection is somewhat insensitive to N . On Llama3.1-8B, its pruning order stabilizes by $N \approx 128$, and from $N = 32$ upward the selected block sets at 20% and 25% sparsity are identical, leaving accuracy unchanged from $N = 32$ to 128 (full results in Appendix G). Since pruning cost is dominated by the calibration forward passes, small calibration sets also directly reduce pruning time.

Sememe-I and SLEB often identified the same sets of blocks, although in differing orders. This is unsurprising, as both optimize perplexity via iterative removal. At smaller calibration set size N , the orderings diverge earlier. Full pruning orders are provided in Appendix A.

5.5 Generalization Across Architectures and Scale

MODEL	SPARSITY	SEMEME-I (N=32)	SLEB (N=128)	SHORTGPT	SHORTENEDLLAMA
MISTRAL-7B-v0.3	20%	60.7 / 80.8 / 9.23	60.5 / 80.7 / 9.25	61.6 / 82.4 / 27.35	51.8 / 68.3 / 16.21
	25%	58.9 / 78.3 / 10.47	58.8 / 78.2 / 10.47	58.0 / 77.7 / 40.82	46.1 / 61.1 / 43.36
LLAMA3.1-8B	20%	61.3 / 81.8 / 26.10	56.0 / 74.4 / 28.85	61.6 / 82.8 / 668.87	57.6 / 76.5 / 25.25
	25%	55.5 / 73.9 / 32.55	56.0 / 74.7 / 35.48	45.4 / 61.2 / $7.9 \cdot 10^5$	56.1 / 74.5 / 30.59
QWEN3-8B	20%	59.6 / 82.8 / 16.75	58.7 / 81.4 / 17.01	58.8 / 81.6 / 18.52	36.1 / 51.0 / $1.5 \cdot 10^5$
	25%	57.0 / 79.1 / 19.11	57.6 / 79.8 / 19.38	57.6 / 79.9 / 22.93	36.1 / 50.9 / $1.6 \cdot 10^5$

Table 4: Depth pruning across architectures. Each cell reports downstream Avg accuracy (%) / $\mathcal{R}$ (percentage of dense-model accuracy retained) / WikiText-2 perplexity. Sememe-I uses $N = 32$ calibration examples and SLEB its default $N = 128$; matched- N comparisons are in Table 3.

Architectures. Table 4 extends the comparison beyond Llama2 to Mistral-7B-v0.3, Llama3.1-8B, and Qwen3-8B. On every model, Sememe-I with $N = 32$ calibration examples matches SLEB with $N = 128$ on accuracy at equal or better perplexity. Sememe-I and SLEB are also the only methods that remain stable across all three architectures: ShortGPT degenerates on Llama3.1-8B and ShortenedLlama on Qwen3-8B, reaching perplexities of 10^2 - 10^5 even while downstream accuracy appears reasonable.

Scale. Retention seems to improve with scale: at 20% sparsity, Sememe-I retains 93.3% of downstream accuracy on Llama3.1-70B vs 89.9% for SLEB, compared to 81.7% and 81.5% on Llama2-7B. Thus, the loss in accuracy becomes far smaller than the fraction of blocks removed (more details in Appendix I). Sememe also degrades more gracefully with additional sparsity at this scale, losing 1.1 percentage points of retention from 20% to 25% sparsity vs 3.4 percentage points for SLEB. However, retention is not uniform across tasks. The pruned 70B retains roughly 99% on PIQA and ARC-Easy while HellaSwag accounts for most of the loss, consistent with the task-aware view that pruning toward a target skill preserves that skill well beyond the average. We consider this an interesting practical regime for depth pruning, as removing blocks frees memory for K/V-cache and larger batch sizes on models that barely fit in GPU memory. Pruning time remains dominated by the calibration forward passes (roughly 11 hours at $N = 32$ vs 22 hours at $N = 64$ to reach 20% sparsity on a single H100), so small calibration sets are most valuable at this scale.

5.6 Beyond Greedy: Joint Interventional Designs

Because importance is defined through interventions (§4), Sememe extends naturally from removing one block at a time to removing groups of blocks: multi-block removals are simply richer interventions handled by the same estimator. We evaluate two such designs, with full results in Appendix H. Interventions that remove two or eight blocks at once underperform one-at-a-time removal on both perplexity and accuracy, as single-block interventions are evaluated exactly and remain in-distribution while multi-block removals push the model further off-distribution and inflate estimation variance. A local design that scores every remaining block and every adjacent pair at each step is theoretically safer, but in practice rarely different. The trade-off is that a paired removal spans two pruning steps, so stopping between them can leave a locally suboptimal model.

6 Conclusion

In this paper, we introduced Sememe, a task-aware depth-pruning framework that formalizes transformer-block importance as an interventional query in a Structural Causal Model and estimates it without bias from per-example calibration losses. We showed that Sememe matches or exceeds comparable training-free depth-pruning methods across four architectures at scales from 7B - 70B parameters, while maintaining computational speedups and using far less calibration data. The causal formulation also makes the framework extensible: calibrating on task data yields task-aware pruning (Sememe-I-Skill), and richer interventional designs extend pruning beyond one-at-a-time removal (§5.6).

7 Limitations

While we report results across four model architectures and scales from 7B to 70B parameters, our evidence at the largest scale comes from a single model family (Llama3.1-70B). Broader 70B+ experiments were outside our compute budget. Our evaluation also follows the standard depth-pruning suite of classification benchmarks, so generative and long-form abilities may degrade differently and remain unmeasured. Additionally, Sememe’s causal model is only as reliable as the calibration data: unrepresentative examples produce unreliable importance estimates regardless of sample size. It is likely that sample selection can play a bigger role in the calibration set size efficiency, but we leave this for future work. Furthermore, although we explored joint-removal in §5.6, full non-greedy search remains an open problem. Finally, the causal modeling is not specific to transformer layers and could be applied to channel-wise or weight-level pruning, with the primary bottleneck being the quadratic scaling in the number of pruning units. We did not explore such extensions in this paper, although it is fundamentally feasible.

Ethics Statement

This work aims to reduce the inference cost of large language models. We view this work’s primary societal effects as a net-positive, as it leads to lower latency, memory, and energy requirements, which broaden access to capable models and reduce the environmental footprint of deployment. All experiments use publicly available pretrained models and standard public benchmarks.

Pruning also carries risks that deployers should weigh. Removing blocks degrades capabilities non-uniformly and safety-relevant behaviors such as refusals may degrade differently from benchmark accuracy. Thus, we recommend re-running safety and alignment evaluations on pruned models (or redoing the necessary post-training) prior to deployment. Task-aware pruning concentrates a model toward a target skill, and the same mechanism could strip properties outside the calibration distribution, including guardrails. These considerations are shared by model-compression techniques generally and are not unique to Sememe.

LLM Disclosure: LLMs were consulted in refining structuring and text within the final draft of the paper. LLM-based programming tools were also used for minor code changes.

Reproducibility Statement

All code is available at <https://github.com/CharlesYu2000/SEMEME>. The algorithm is also detailed in Appendix B. Computed pruning block orderings are provided in Appendix A and can be reused without running any of our code.

Acknowledgments

This research is based upon work supported by DARPA ITM Program No. FA8650-23-C-7316, by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 560000C260018, NSF Molecule Maker Lab Institute, an AI Institute for Molecular Discovery, Synthesis Strategy, and Manufacturing funded by the U.S. National Science Foundation under Awards No. 2019897 and 2505932, DARPA MAGICs Program, NSF NAIRR Award, the AI Research Institutes program by National Science Foundation and the Institute of Education Sciences, U.S. Department of Education through Award # 2229873 - AI Institute for Transforming Education for Children with Speech and Language Processing Challenges, Amazon-Illinois Center on AI for Interactive Conversational Experiences (AIICE), CapitalOne-Illinois Center for Generative AI Safety, Knowledge Systems, and Cybersecurity (ASKS), IBM-Illinois Discovery Accelerator Institute (IIDAI) Center. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This research used the Delta and DeltaAI advanced computing and data resources, which are supported by the National Science Foundation (award OAC 2320345 and award OAC 2005572) and the State of Illinois. Delta and DeltaAI are joint efforts of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications.

References

- Patrick Blöbaum, Peter Götz, Kailash Budhathoki, Atalanti A. Mastakouri, and Dominik Janzing. Dowhy-gcm: An extension of dowhy for causal inference in graphical causal models. *arXiv preprint arXiv:2206.06821*, 2022.
- Yi Cao, Wei-Jie Xu, Yucheng Shen, Weijie Shi, Chi-Min Chan, Jianfeng Qu, and Jiajie Xu. PIP: Perturbation-based Iterative Pruning for Large Language Models, November 2025. URL <http://arxiv.org/abs/2501.15278>. arXiv:2501.15278 [cs].

- Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models, 2025a. URL <https://arxiv.org/abs/2403.19135>.
- Xinrui Chen, Haoli Bai, Tao Yuan, Ruikang Liu, Kang Zhao, Xianzhi Yu, Lu Hou, Tian Guan, Yonghong He, and Chun Yuan. A simple linear patch revives layer-pruned large language models, 2025b. URL <https://arxiv.org/abs/2505.24680>.
- Xuan Ding, Rui Sun, Yunjian Zhang, Xiu Yan, Yueqi Zhou, Kaihao Huang, Suzhong Fu, Angelica I. Aviles-Rivero, Chuanlong Xie, and Yao Zhu. Sliding-Window Merging for Compacting Patch-Redundant Layers in LLMs, December 2025. URL <http://arxiv.org/abs/2502.19159>. arXiv:2502.19159 [cs].
- Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL <https://arxiv.org/abs/2301.00774>.
- Advait Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why random pruning is all we need to start sparse, 2023. URL <https://arxiv.org/abs/2210.02412>.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL <https://zenodo.org/records/10256836>.
- Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. URL <https://arxiv.org/abs/1506.02626>.
- B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pp. 293–299 vol.1, 1993. doi: 10.1109/ICNN.1993.298572.
- Dennis L. Jackson. Revisiting sample size and number of parameter estimates: Some support for the n:q hypothesis. *Structural Equation Modeling: A Multidisciplinary Journal*, 10(1): 128–141, 2003. doi: 10.1207/S15328007SEM1001_6. URL https://doi.org/10.1207/S15328007SEM1001_6.
- Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: Depth pruning for large language models with comparison of retraining methods, 2024. URL <https://arxiv.org/abs/2402.02834>.
- Rex B. Kline. *Principles and practice of structural equation modeling*, 4th ed. The Guilford Press, New York, NY, US, 2016. ISBN 978-1-4625-2334-4 978-1-4625-2335-1 978-1-4625-2300-9. Pages: xvii, 534.
- Yann LeCun, John Denker, and Sara Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL <https://proceedings.neurips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html>.
- Eunho Lee and Youngbae Hwang. Automatic channel pruning for multi-head attention, 2024. URL <https://arxiv.org/abs/2405.20867>.
- Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning, 2021. URL <https://arxiv.org/abs/2010.07611>.
- Shiwei Liu, Tianlong Chen, Zhenyu Zhang, Xuxi Chen, Tianjin Huang, Ajay Jaiswal, and Zhangyang Wang. Sparsity may cry: Let us fail (current) sparse neural networks together!, 2023. URL <https://arxiv.org/abs/2303.02141>.
- Yao Lu, Yuqi Li, Wenbin Xie, Shanqing Yu, Qi Xuan, Zhaowei Zhu, and Shiping Wen. The structural scalpel: Automated contiguous layer pruning for large language models, 2025. URL <https://arxiv.org/abs/2510.23652>.

- 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, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/44956951349095f74492a5471128a7e0-Abstract-Conference.html.
- Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. URL <https://arxiv.org/abs/2403.03853>.
- Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What’s encoded in a winning ticket’s mask?, 2022. URL <https://arxiv.org/abs/2210.03044>.
- Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000. ISBN 0-521-77362-8.
- Amit Sharma and Emre Kiciman. Dowhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020.
- Mustafa Shukor and Matthieu Cord. Skipping Computations in Multimodal LLMs, October 2024. URL <http://arxiv.org/abs/2410.09454>. arXiv:2410.09454 [cs].
- Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungiun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: Streamlining LLMs through Redundancy Verification and Elimination of Transformer Blocks, December 2024. URL <http://arxiv.org/abs/2402.09025>. arXiv:2402.09025 [cs].
- Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models, May 2024. URL <http://arxiv.org/abs/2306.11695>. arXiv:2306.11695 [cs].
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface’s transformers: State-of-the-art natural language processing, 2020. URL <https://arxiv.org/abs/1910.03771>.
- Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M. Kübler, Rupak Vignesh Swaminathan, Thanasis Mouchtaris, Sravan Babu Bodapati, Nathan Susanj, Zheng Zhang, Jack G. M. FitzGerald, and Abhishek Kumar. Wanda++: Pruning large language models via regional gradients, 2025. URL <https://www.amazon.science/publications/wanda++-pruning-large-language-models-via-regional-gradients>.
- Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(3):3143–3151, Jun. 2022. doi: 10.1609/aaai.v36i3.20222. URL <https://ojs.aaai.org/index.php/AAAI/article/view/20222>.

A The Pruned Transformer Layers

Across configurations, the pruning orders exhibit a consistent pattern: middle layers (approximately layers 10–28 for 32-layer models) are preferentially removed, while the first and last few layers are almost always retained. This aligns with expectations that early layers perform critical feature extraction and final layers are essential for output projection. It also confirms the heuristics observed in prior pruning studies (Lee et al., 2021; Ma et al., 2023; Kim et al., 2024) which explicitly disallow pruning of the first and last few layers.

Notably, task-specific calibration (Sememe-I-Skill) produces substantially different orderings across tasks, confirming that redundancy is task-dependent.

1. Llama2-7B Sememe-I-Skill (ArcE): 28, 13, 4, 25, 8, 30, 29, 15, 18
2. Llama2-7B Sememe-I-Skill (ArcC): 27, 19, 13, 26, 24, 21, 23, 6, 31
3. Llama2-7B Sememe-I-Skill (WG): 26, 22, 28, 19, 27, 17, 25, 21, 30
4. Llama2-7B Sememe-I-Skill (HellaSwag): 29, 5, 26, 11, 22, 23, 27, 19, 25
5. Llama2-7B Sememe-I-Skill (PIQA): 15, 13, 19, 21, 27, 11, 28, 23, 26
6. Llama2-7B Sememe-I: 14, 23, 11, 24, 10, 27, 15, 21, 25
7. Llama2-7B Sememe-C: 12, 18, 24, 11, 23, 10, 29, 8, 9
8. Llama2-13B Sememe-I-Skill (ArcE): 28, 27, 9, 34, 22, 21, 20, 10, 11, 39
9. Llama2-13B Sememe-I-Skill (ArcC): 11, 25, 27, 21, 35, 10, 34, 36, 5, 28
10. Llama2-13B Sememe-I-Skill (WG): 15, 24, 26, 33, 32, 31, 23, 21, 37, 28
11. Llama2-13B Sememe-I-Skill (HellaSwag): 36, 37, 29, 35, 23, 26, 24, 16, 27, 17
12. Llama2-13B Sememe-I-Skill (PIQA): 7, 29, 18, 30, 15, 35, 33, 22, 24, 21
13. Llama2-13B Sememe-I: 31, 27, 13, 12, 33, 29, 14, 26, 35, 11
14. Llama2-13B Sememe-C: 30, 12, 7, 22, 13, 26, 29, 33, 14, 19
15. Llama3.1-8B Sememe-I-Skill (ArcE): 19, 29, 22, 26, 23, 3, 24, 25, 17, 9, 10, 31, 30, 21, 27, 28
16. Llama3.1-8B Sememe-I-Skill (ArcC): 4, 18, 23, 24, 25, 14, 30, 22, 26, 19, 28, 8, 12, 11, 17, 5
17. Llama3.1-8B Sememe-I-Skill (WG): 4, 24, 20, 22, 29, 28, 15, 23, 14, 21, 30, 18, 10, 26, 8, 31
18. Llama3.1-8B Sememe-I-Skill (HellaSwag): 16, 19, 26, 21, 25, 29, 24, 22, 2, 8, 18, 11, 31, 30, 10, 9
19. Llama3.1-8B Sememe-I-Skill (PIQA): 20, 27, 25, 18, 10, 11, 9, 17, 30, 19, 14, 28, 7, 12, 22, 3
20. Llama3.1-8B Sememe-C: 9, 18, 8, 19, 10, 22, 25, 11, 23, 7, 28, 26, 20, 12, 15, 6
21. Llama3.1-8B (Calibration size 32) Sememe-I: 10, 25, 26, 11, 12, 23, 19, 8
22. Mistral-7B-v0.3 (Calibration size 32) Sememe-I: 13, 14, 24, 8, 27, 15, 23, 22
23. Qwen3-8B (Calibration size 32) Sememe-I: 17, 20, 2, 16, 21, 15, 26, 32, 14
24. Llama3.1-70B (Calibration size 32) Sememe-I: 6, 10, 15, 9, 8, 5, 60, 63, 55, 32, 61, 73, 64, 53, 62, 67, 49, 65, 59, 54

B Sememe Algorithm

The algorithm is detailed in Algorithm 1.

C Proof of Proposition 4.2: Invertibility of the Design Matrix

Proof. Each of the K one-at-a-time interventions contributes N identical rows to $\mathbf{D}$, giving us:

$$(\mathbf{D}^\top \mathbf{D})_{ij} = \begin{cases} N(K-1) & \text{if } i = j, \\ N(K-2) & \text{if } i \neq j. \end{cases} \quad (6)$$

This matrix has the form $N(K-2)\mathbf{J}_K + N\mathbf{I}_K$, where $\mathbf{J}_K = \mathbf{1}_K\mathbf{1}_K^\top$ is the all-ones matrix. Its eigenvalues are $N(K-1)^2$ (for the eigenvector $\mathbf{1}_K$, with multiplicity 1) and N (for the $K-1$ eigenvectors orthogonal to $\mathbf{1}_K$). Both are strictly positive for $K \geq 2$ and $N \geq 1$. $\square$

Algorithm 1 Sememe algorithm. Based on calibration set of random examples and a Loss function, we prune an LLM to its desired sparsity level.

Input: LLM $M = \{b_1, \dots, b_T\}$, calibration data $X = \{x_1, \dots, x_N\}$, loss function $Loss$, number of blocks to prune n

$M_0 \leftarrow M$
 $T \leftarrow |M|$ (number of transformer blocks of M)
 $Pruned \leftarrow \{\}$
for $t = 1$ **to** n **do**
 $Samples \leftarrow []$ (matrix)
 for $i = 1$ **to** T **do**
 if $i \notin Pruned$ **then**
 $M' \leftarrow M_{t-1} \setminus b_i$ (convert B_i into no-op)
 $Do \leftarrow$ one-cold row-vector with 0 at index i
 for $j = 1$ **to** N **do**
 $\mathcal{L} \leftarrow Loss(M', x_j)$
 $Sample \leftarrow [Do, \mathcal{L}]$
 $ADD_ROW(Samples, Sample)$
 end for
 end if
 end for
 $B_1, \dots, B_T, Y \leftarrow Samples$ (column vectors)
 if Sememe-I **then**
 Solve for $\gamma: \gamma^{(1)} B^{(1)} + \dots + \gamma^{(T)} B^{(T)} + \zeta = Y$
 Convert all $\gamma_1, \dots, \gamma_T$ to $\hat{\gamma}_1, \dots, \hat{\gamma}_T$
 $p \leftarrow \text{argmax}(\hat{\gamma}_1, \dots, \hat{\gamma}_T)$
 else if Sememe-C **then**
 $p \leftarrow \text{argmin}(\text{avg}(Loss'(M, i)))$ (sampled from Gaussian mixture model)
 end if
 $M_t \leftarrow M_{t-1} \setminus b_p$
 $ADD(Pruned, p)$
end for
Return: M_n

The classical SEM $N:q$ heuristic (Jackson, 2003; Kline, 2016) recommends $N \geq 20q$ where q is the number of parameters being estimated in the causal model. In our design, $q = K$ and N is the calibration set size per layer, giving an $N:q$ ratio equal to the calibration set size for any LLM. Thus, with even 32 calibration set size, the heuristic is satisfied. Furthermore, the interventional design ensures identification regardless of N (Proposition 4.2), so the $N:q$ heuristic is conservative here.

D Proof of Proposition 4.3: Unbiasedness under Intervention

We show that the OLS estimator $\hat{\gamma}$ from Eq. (3) is unbiased for the coefficient vector of the best linear predictor of Y given $\mathbf{B} = (B_1, \dots, B_K)$.

Setup. Let the true data-generating process be $Y = f(B_1, \dots, B_K) + \varepsilon$, where f is an arbitrary (possibly non-linear) function and ε is zero-mean noise independent of the B_i . We do *not* assume that f is linear. The *best linear predictor* (linear projection) of Y onto $\mathbf{B}$ is defined by the coefficient vector

$$\gamma^* = \arg \min_{\gamma} \mathbb{E} \left[\left(Y - \mathbf{B}^\top \gamma \right)^2 \right], \quad (7)$$

which satisfies the normal equation $\gamma^* = (\mathbb{E}[\mathbf{B}\mathbf{B}^\top])^{-1} \mathbb{E}[\mathbf{B}Y]$, provided the expectation exists and the matrix is invertible.

Exogeneity under intervention. Under our one-at-a-time intervention design, the indicators B_i are set deterministically by the experimenter: in each trial, exactly one B_i is set

to 0 and the rest to 1. The design matrix $\mathbf{D}$ is therefore *non-stochastic*. In the structural causal model framework of Pearl (2000), the intervention $\text{do}(B_i = 0)$ replaces the structural equation for B_i with the constant assignment $B_i := 0$, severing all incoming edges to B_i in the causal graph. This ensures that $\mathbf{D}$ is independent of the disturbance ζ :

$$\mathbb{E}[\zeta \mid \mathbf{D}] = \mathbb{E}[\zeta] = 0. \quad (8)$$

Unbiasedness. Substituting $\mathbf{y} = \mathbf{D}\gamma^* + \zeta$ (where ζ is the column vector of per-observation disturbances) into the OLS formula:

$$\begin{aligned} \hat{\gamma} &= (\mathbf{D}^\top \mathbf{D})^{-1} \mathbf{D}^\top \mathbf{y} \\ &= (\mathbf{D}^\top \mathbf{D})^{-1} \mathbf{D}^\top (\mathbf{D}\gamma^* + \zeta) \\ &= \gamma^* + (\mathbf{D}^\top \mathbf{D})^{-1} \mathbf{D}^\top \zeta. \end{aligned} \quad (9)$$

Taking expectations and using $\mathbb{E}[\zeta \mid \mathbf{D}] = \mathbf{0}$ (since $\mathbf{D}$ is non-stochastic and ζ has zero mean):

$$\mathbb{E}[\hat{\gamma}] = \gamma^* + (\mathbf{D}^\top \mathbf{D})^{-1} \mathbf{D}^\top \mathbb{E}[\zeta] = \gamma^*. \quad (10)$$

Also, since $\mathbf{D}$ is non-stochastic, $\mathbf{D}^\top \mathbf{D}$ is invertible (Proposition 4.2), and ζ has finite variance, the Gauss–Markov theorem guarantees that $\hat{\gamma}$ is the best linear unbiased estimator (BLUE) of γ^* among all linear estimators. $\square$

E First-order Local Approximation

Let $f: \{0, 1\}^K \rightarrow \mathbb{R}$ be the true mapping from the block indicator vector $\mathbf{b} = (B_1, \dots, B_K)$ to the expected task loss. Expanding around the all-active configuration $\mathbf{b}_0 = \mathbf{1}$:

$$f(\mathbf{b}) = f(\mathbf{1}) + \sum_{i=1}^K \underbrace{\frac{\partial f}{\partial B_i} \Big|_{\mathbf{b}=\mathbf{1}}}_{= -\gamma_i} (B_i - 1) + \mathcal{O}(\|\mathbf{b} - \mathbf{1}\|^2). \quad (11)$$

Our one-at-a-time design evaluates f at configurations that differ from $\mathbf{1}$ in exactly one coordinate. At these Hamming-distance-1 points, the higher-order (interaction) terms are zero and the first-order expansion is *exact for each single-block query*. The path coefficients γ_i therefore recover the exact partial effects $\partial f / \partial B_i$ from these one-at-a-time interventions. Pairwise and higher-order block interactions are not captured, but they are not needed: at each greedy step, we remove only one additional block.

F Sememe-C

F.0.1 Distributional Model

At step t , the intervention $\text{do}(B_i = 0)$ produces N loss observations per block:

$$\ell_{i,1}, \dots, \ell_{i,N}, \quad \text{where} \quad \ell_{i,j} = \mathcal{L}(\mathcal{M}_{t-1} \setminus b_i, x_j). \quad (12)$$

Sememe-C models each block’s loss profile as a Gaussian mixture:

$$p(\ell \mid \text{do}(B_i = 0)) = \sum_{k=1}^{K_{\text{mix}}} w_{i,k} \mathcal{N}(\ell; \mu_{i,k}, \sigma_{i,k}^2), \quad (13)$$

where K_{mix} is the number of mixture components, $\{w_{i,k}\}$ are mixing weights ($\sum_k w_{i,k} = 1$), and the parameters $\theta_i = \{w_{i,k}, \mu_{i,k}, \sigma_{i,k}^2\}_{k=1}^{K_{\text{mix}}}$ are estimated via variational Bayesian inference using the DoWhy package (Sharma & Kiciman, 2020; Blöbaum et al., 2022). The number of mixture components K_{mix} is automatically optimized based on the data by the GCM extension of the DoWhy package (Blöbaum et al., 2022).

F.0.2 Counterfactual Query and Pruning Decision

With the fitted distributions in hand, we pose a counterfactual question: “Given the loss samples we observed on the calibration data, what would the expected loss be given the additional removal of block b_i ?” Formally, we compute the expected loss under each block’s fitted distribution:

$$s_{\text{CF}}(b_i) = \mathbb{E}_{p(\ell|\text{do}(B_i=0))}[\ell] = \sum_{k=1}^{K_{\text{mix}}} w_{i,k} \cdot \mu_{i,k}. \quad (14)$$

The block selected for pruning is:

$$\pi_t = \arg \min_{b_i \in B_t} s_{\text{CF}}(b_i), \quad (15)$$

i.e., we prune the block whose removal leads to the lowest expected loss, consistent with Definition 4.1.

F.0.3 Connection to Layer 3 Inference

The key distinction from Sememe-I is that the GMM captures the full *shape* of the loss distribution under each intervention, not merely its first-order linear projection. This is a Layer 3 (counterfactual) computation in the sense of Pearl (2000): we condition on the observed calibration losses (the evidence), fit a generative model that serves as a “twin network,” and then query what the loss *would be* under a hypothetical block removal. The fitted generative model allows the importance estimate to reflect higher-order distributional features (e.g., multimodality, skewness) that the linear model in Sememe-I cannot capture.

F.0.4 Trade-offs Relative to Sememe-I

The distributional approach introduces $3K_{\text{mix}} - 1$ parameters per block (vs. a single path coefficient for Sememe-I), all of which must be estimated from N observations. When N is small relative to K_{mix} , variational inference may produce poorly fit posteriors and noisy importance estimates. Lighter-weight alternatives (e.g., a single Gaussian, kernel density estimates) may be preferable when N is very small.

F.1 Sememe Performance Across Pruning Iterations

When looking at the performance over time in Figure 4, we see that the task-specific performance of Sememe-I-Skill seems roughly linear. Perhaps unsurprisingly, the pruned model is more easily able to retain the ability for some tasks vs others, indicating that the abilities are more centralized in specific blocks of the model.

The perplexity graph at the bottom of Figure 4 looks uninteresting at first glance, but upon closer inspection reveals that the performance degradation of Sememe-C is much smoother than that of Sememe-I, although the performance is not necessarily better. This difference is potentially caused by the greedy nature of Sememe-I vs the simulated nature of Sememe-C.

G Additional Calibration Ablation Results

Table 5 contains results for Sememe-I with different calibration size on Llama3.1-8B. The pruning order stabilizes by $N \approx 128$. From $N = 32$ upward, the selected block sets at both sparsity levels are identical and the orders differ only in position, so the pruned models and their accuracies coincide.

Table 6 reports the per-pruning-step WikiText-2 perplexity underlying Figure 3. Both come from earlier runs whose random calibration set differed from the runs above. This cause a difference in one chosen block, hence the slightly improved perplexity result vs Table 4.

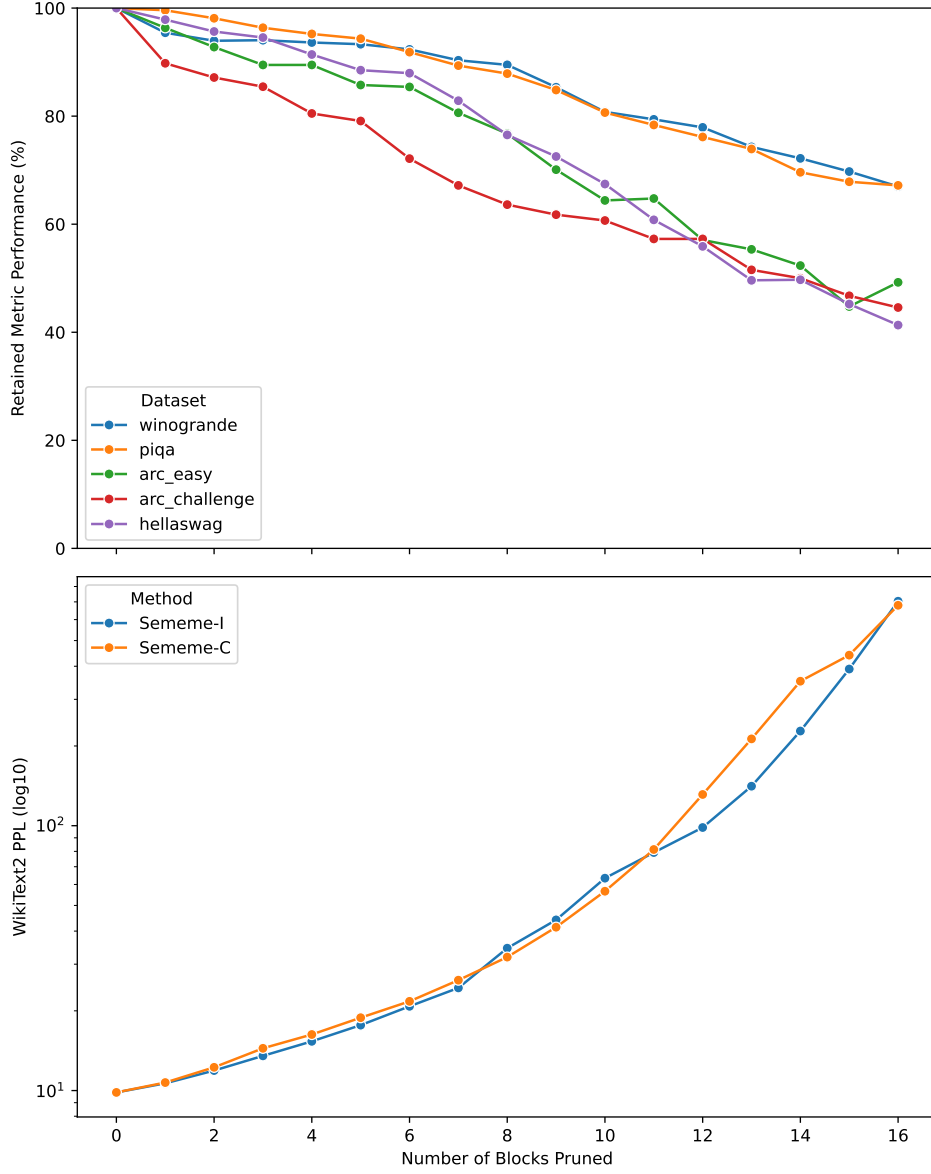


Figure 4: Performance of pruned Llama3.1-8B over the course of pruning.

N	20%	25%	BLOCK REMOVAL ORDER
8	56.4 / 75.0	55.5 / 73.9	25, 10, 26, 11, 19, 12, 8, 23
16	60.5 / 80.9	55.2 / 73.3	10, 25, 26, 11, 19, 12, 28 , 8
32	61.3 / 81.8	55.5 / 73.9	10, 25, 26, 11, 12, 23, 19, 8
64	61.3 / 81.8	55.5 / 73.9	10, 25, 26, 11, 12, 19, 23, 8
128	61.3 / 81.8	55.5 / 73.9	10, 25, 11, 26, 12, 19, 23, 8

Table 5: Sememe-I on Llama3.1-8B across calibration sizes N : downstream Avg / $\mathcal{R}$ at 20% and 25% sparsity. The block removal order includes the first eight blocks removed (to achieve 25% sparsity). The differences in N make only minute changes in the order, except at $N = 16$ which has one block difference (bolded).

LAYERS PRUNED	SHORTGPT	SHORTENEDLLAMA	SLEB (32)	SLEB (128)	SEMEME
1	10.90	10.65	10.65	10.63	10.65
2	14.89	12.14	11.88	11.86	11.88
3	21.20	13.52	13.51	13.31	13.51
4	52.27	15.52	15.53	15.04	15.36
5	137.80	17.96	18.04	17.24	17.80
6	332.17	21.32	20.79	20.79	20.50
7	668.87	25.25	28.88	28.88	24.73

Table 6: WikiText-2 perplexity ($\downarrow$) at each pruning step for Llama3.1-8B (7 layers pruned of 32; 20% sparsity). The Sememe column uses only $N = 32$ calibration examples; SLEB columns use the calibration set size in parentheses. Best result per row in **bold**.

H Joint Interventional Designs

Multi-block interventions. At each pruning step, we estimate block effects from interventions that remove several blocks at once and prune to 25% sparsity accordingly, using $N = 128$ calibration examples. One-at-a-time removal is best on every metric for both models. Single-block interventions are evaluated exactly and remain in-distribution, while removing several blocks at once pushes the model further off-distribution and inflates estimation variance. Searching over multi-block interventions is also substantially more expensive, taking roughly $7\text{-}9\times$ the wall-clock time for groups of eight.

BLOCKS PER INTERVENTION	LLAMA2-7B		MISTRAL-7B-v0.3	
	PPL	AVG	PPL	AVG
1 (GREEDY)	9.91	56.2	10.75	58.7
2	12.38	54.4	13.97	54.8
8	12.15	55.1	14.88	57.6

Table 7: Interventions removing a fixed number of blocks at once, pruning to 25% sparsity with $N = 128$ calibration examples.

Local adjacent-pair design. At each step we score every remaining block and every adjacent pair, removing the most efficient intervention. On Llama2-7B, Llama3.1-8B, Mistral-7B-v0.3, and Qwen3-8B, this effectively reduces to the greedy selection, choosing the same blocks in the same or slightly shuffled order. On Llama2-13B, this method identifies a redundant adjacent pair and improves both sparsity levels under full evaluation:

LLAMA2-13B	20% (8/40)	25% (10/40)
SLEB	63.2 / 87.2 / 7.07	62.2 / 86.0 / 7.93
SEMEME-I (GREEDY)	63.2 / 87.2 / 7.07	60.4 / 83.3 / 7.90
SEMEME-I (ADJACENT PAIRS)	65.0 / 90.3 / 6.96	63.9 / 88.7 / 7.87

Table 8: The local adjacent-pair design on Llama2-13B, all with $N = 128$ calibration examples.

Note that this ablation used a different random calibration sample from the Table 2 setup, in which Sememe-I and SLEB select identical block sets at 25%.

I Llama3.1-70B Details

LLAMA3.1-70B	20% (16/80)	25% (20/80)
SEMEME-I ($N=32$)	73.8 / 93.3 / 10.81	72.8 / 92.2 / 12.82
SLEB ($N=32$)	71.3 / 89.9 / 10.49	68.7 / 86.5 / 13.32
SHORTGPT	72.4 / 91.5 / 12.87	71.1 / 90.0 / 34.66

Table 9: Sememe leads on downstream retention at both sparsity levels, while SLEB retains the best perplexity at 20%.

Per-task retention degradation concentrates in HellaSwag and ARC-Challenge while the remaining tasks are largely unaffected.

Due to computational restraints, we used subsampled evaluation sets. Dense and SLEB rows were evaluated with a 10000-document limit per task (complete for most benchmarks), while pruned Sememe rows use a 500-example subset per task and ShortGPT a 1000-example subset. $\mathcal{R}$ normalizes per-task against the shared dense evaluation, so subset rows are unbiased but noisier.