RACE: Scalable Statistical Estimation of Functional Consistency in LLM Neurons

Runyu Wang[1] **Bo Liu**[2] **Xiaxin Zhang**[1] **Yu Han**[1] **Jiawei Cao**[1]
Xiaoye Zhang[3] **Zhe Zhang**[4] **Yifan Yang**[4] **Peng Ping**[1]*

[1]School of Transportation and Civil Engineering, Nantong University
[2]Chongqing University of Post and Telecommunications
[3]China Southern Power Grid Company Limited [4]Meituan

2430310032@stmail.ntu.edu.cn s250201066@stu.cqupt.edu.cn 2433320001@stmail.ntu.edu.cn 2433320018@stmail.ntu.edu.cn
2233110297@stmail.ntu.edu.cn xiaoyz@whu.edu.cn zhangzhecnjs@gmail.com yangyifan@meituan.com pingpeng@ntu.edu.cn

Abstract

Discovering stable neuron behavior across entire domains remains a challenge in mechanistic interpretability. Existing methods often rely on instance-level point estimates or computationally expensive procedures, which either obscure population-level variability or limit scalable domain-wide analysis. We present ***RACE*** (Residual Alignment for Consistency Estimation), a forward-pass statistical framework that evaluates the domain-wide functional consistency of Transformer neurons. Perturbation experiments demonstrate that RACE achieves superior domain specificity compared to gradient-based point estimates. Meanwhile, token-distribution-level results verify the association between the selected neurons and the target domain. Furthermore, its computational overhead is three orders of magnitude lower than that of gradient-based methods.

1 Introduction

Recent advances in mechanistic interpretability have improved our understanding of Transformer-based Large Language Models (LLMs) (Zhao et al., 2024; Rai et al., 2024). Three dominant approaches have emerged: causal tracing and knowledge localization techniques that identify critical model pathways (Dai et al., 2021; Meng et al., 2022), gradient-based attribution methods that quantify parameter importance (Sundararajan et al., 2017; Achtibat et al., 2024), and sparse autoencoders that extract interpretable neuron activations (Bricken et al., 2023; Shu et al., 2025).

While these methods excel at providing instance-level explanations, they face a limitation: many real-world applications, including model capability auditing, domain-specific pruning, and behavioral steering, demand population-level characterizations of model components. Specifically, these tasks require rankings of task-relevant neurons that remain consistent across diverse inputs.

Recent work has attempted to bridge this gap through task-neuron mapping strategies: causal gradient variation localizes neurons that selectively affect target tasks (Song et al., 2024), while gradient attribution links task-specific neuron overlap to cross-task generalization (Leng and Xiong, 2025). However, existing pipelines suffer from two constraints: computational inefficiency stems from iterative gradient calculations and intervention operations, creating prohibitive overhead. Meanwhile, statistical oversimplification emerges when aggregating neuron contributions into task-level averages, which obscures sample-to-sample variation patterns.

To overcome these limitations, we draw upon two empirical observations: specific LLM capabilities rely on sparse subsets of neurons (Frankle and Carbin, 2019; Frantar and Alistarh, 2023), and these neurons respond uniquely to semantically coherent inputs (Voita et al., 2024; Huang et al., 2025). Since neuron activations directly modulate the residual stream, we hypothesize that during the forward pass, neurons serving consistent domain-specific functions will deposit contribution distributions over this stream that differ from those of non-domain-specific neurons in a statistically significant manner.

Accordingly, we introduce ***RACE*** (Residual Alignment for Consistency Estimation), a statistical estimation framework that scores every neuron at every layer for *functional consistency* with respect to a *target-domain observation set* (§2.1). RACE consists of two stages: (1) decomposing each module's residual stream update into per-neuron contributions and evaluating their alignment with the module's output direction via Residual-Direction Alignment (RDA) at forward-pass cost; and (2) distilling noisy per-observation signals into posterior distributions over each neuron's mean alignment

*Corresponding author.

and variance through Bayesian aggregation.

We validate on 4B–32B LLMs across code generation, mathematical reasoning, and fine-grained behavioral control. Targeted suppression shows RACE selectively disrupts target capabilities while preserving non-target behaviors. Empirical results confirm RACE's computational efficiency and demonstrate its superiority over gradient-based baselines. Ablation studies validate the contribution of each RACE component.

2 Method

2.1 Problem Formulation

Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a trained Transformer with layers $l = 1, \ldots, L$ and modules $m \in \{\mathrm{ATTN}, \mathrm{MLP}\}$, denoting attention and multilayer perceptron modules, respectively. We write $u = (l, m, j)$ for a neuron in a fixed layer–module pair.

Target-Domain Observation Set. A *target domain* c is specified by a predicate $\phi_c : \mathcal{X} \to \{\mathrm{True}, \mathrm{False}\}$ that induces the input population $\mathcal{D}_c = \{\mathbf{x} \in \mathcal{X} : \phi_c(\mathbf{x}) = \mathrm{True}\}$. In practice, we approximate this with a finite sample $D_c = \{\mathbf{x}_i\}_{i=1}^{N} \subseteq \mathcal{D}_c$. Because the Transformer is token-position dependent, the auditing protocol evaluates one or more positions per input, yielding the operational observation set

$$T_c = \{(\mathbf{x}_i, p) : \mathbf{x}_i \in D_c,\ p \in \mathcal{P}_c(\mathbf{x}_i)\}, \quad n_c = |T_c|, \tag{1}$$

where $\mathcal{P}_c(\mathbf{x})$ denotes the analyzed token positions.

Functional Consistency. The *functional consistency score* of neuron j with respect to D_c reflects how well its contribution distribution satisfies two desiderata. Specifically, let $e_j^{(t)} \in \mathbb{R}$ denote neuron j's signed contribution for observation $t \in T_c$, with the sign defined relative to the target alignment direction. The score evaluates:

- **Magnitude:** $\mathbb{E}_{t \in T_c}[e_j^{(t)}]$ is large and positive relative to other neurons in the same module.
- **Stability:** $\mathrm{Var}_{t \in T_c}[e_j^{(t)}]$ is small—the contribution is not driven by a few outlier observations.

Neurons that activate strongly on a handful of observations but are silent on most, or whose signed contributions change direction across observations, fail to satisfy these conditions despite potentially high average unsigned magnitude. We denote by $\mathcal{A}_{\mathcal{M}}(U_{l,m}, T_c) \in \mathbb{R}^{|U_{l,m}|}$ the vector of functional consistency scores for all neurons in module m at layer l, given observation set T_c.

To estimate $\mathcal{A}_{\mathcal{M}}(U_{l,m}, T_c)$ across all layers and modules, RACE frames consistency evaluation as a problem of statistical inference over the evidence set $\{e_j^{(t)}\}_{t \in T_c}$ (hereafter, we drop the layer and module subscripts for readability and simply write j for the audited neuron). Specifically, we model each neuron as possessing latent evidence-distribution parameters $\theta_j = (\mu_j, \sigma_j^2)$, treating its per-observation evidence as noisy realizations from an underlying distribution:

$$e_j^{(t)} \mid \theta_j \sim P(\cdot \mid \mu_j, \sigma_j^2), \quad \forall\, t \in T_c \tag{2}$$

Functional consistency scoring then becomes a matter of posterior inference:

$$P(\theta_j \mid T_c) \propto \prod_{t \in T_c} P\left(e_j^{(t)} \mid \mu_j, \sigma_j^2\right) \cdot P(\mu_j, \sigma_j^2). \tag{3}$$

Crucially, this posterior formulation directly conceptually maps to our two desiderata: the posterior over μ_j captures the functional magnitude and direction, while the posterior over σ_j^2 and the epistemic uncertainty in μ_j jointly encode stability.

To estimate this posterior in practice, RACE employs a two-stage pipeline. First, RDA computes the module-local evidence $e_j^{(t)}$ from a single forward pass for each observation (§2.2). Second, Bayesian aggregation infers θ_j from this evidence population (§2.3) using a Normal-Inverse-Gamma (NIG) conjugate model. This aggregation yields closed-form posterior mean estimates and variance-calibrated Conservative Alignment Magnitude (CAM) scores for neuron selection (§2.4). Finally, for intervention settings that require disentangling target-specific neurons from broadly active ones, we introduce Reference-Set Filtering (RSF) as a filtration step (§2.5).

2.2 Residual-Direction Alignment

RDA provides the per-observation evidence for Bayesian aggregation. It projects each neuron's weighted output onto the normalized residual update of its host module, thereby avoiding raw-activation proxies (Shrikumar et al., 2017; Meng et al., 2022) and the computational overhead of gradient-based attribution (Sundararajan et al., 2017).

2.2.1 Neuron-Level Residual Decomposition

The residual stream $\mathbf{r}$ flows from layer to layer, with each module contributing $\Delta\mathbf{r}_{\text{module}}$: $\mathbf{r}_{\text{out}} =$

$\mathbf{r}_{\text{in}} + \Delta\mathbf{r}_{\text{module}}$. For a fixed observation and layer, this update decomposes naturally into per-neuron contributions:

$$\Delta\mathbf{r}_{\text{ATTN}} = \mathbf{W}_{\text{O}}\mathbf{h} = \sum_i h_i \mathbf{W}_{\text{O}_{[:,i]}} \tag{4}$$

$$\Delta\mathbf{r}_{\text{MLP}} = \mathbf{W}_{\text{down}}\mathbf{a} = \sum_j a_j \mathbf{W}_{\text{down}_{[:,j]}} \tag{5}$$

where $\mathbf{h} \in \mathbb{R}^{d_{\text{model}}}$ is the concatenated attention head output and $\mathbf{a} \in \mathbb{R}^{d_{\text{MLP}}}$ is the MLP intermediate activation before the down projection. Each term is a rank-1 sub-update to the residual stream, scaled by activation magnitude. This sub-update view follows prior work (Geva et al., 2021, 2022). RACE audits these same additive units: an MLP neuron j is the intermediate channel contributing $a_j \mathbf{W}_{\text{down}_{[:,j]}}$, while an attention neuron denotes an output-channel contribution $h_i \mathbf{W}_{\text{O}_{[:,i]}}$ (Elhage et al., 2021; Dai et al., 2021; Yu and Ananiadou, 2024).

2.2.2 Module-Local Alignment Evidence

RDA uses each module's output $\Delta\mathbf{r}$ as the evaluation axis. Appendix A discusses why this module-local choice is better matched to the evidence that RACE aims to collect. For an observation t at layer l, we define:

$$\hat{\mathbf{d}}_{\text{ATTN}} = \frac{\Delta\mathbf{r}_{\text{ATTN}}}{\|\Delta\mathbf{r}_{\text{ATTN}}\|_2 + \varepsilon} \tag{6}$$

$$\hat{\mathbf{d}}_{\text{MLP}} = \frac{\Delta\mathbf{r}_{\text{MLP}}}{\|\Delta\mathbf{r}_{\text{MLP}}\|_2 + \varepsilon} \tag{7}$$

where ε is a small constant for numerical stability. The unit vector $\hat{\mathbf{d}}$ represents the direction of the module's additive update to the residual stream.

Each neuron's alignment with this axis is then computed as a signed, activation-modulated score:

$$\mathbf{s}_{\text{ATTN}} = \mathbf{W}_{\text{O}}^T \hat{\mathbf{d}}_{\text{ATTN}}, \quad \xi_{\text{ATTN}_i} = h_i \cdot s_{\text{ATTN}_i} \tag{8}$$

$$\mathbf{s}_{\text{MLP}} = \mathbf{W}_{\text{down}}^T \hat{\mathbf{d}}_{\text{MLP}}, \quad \xi_{\text{MLP}_j} = a_j \cdot s_{\text{MLP}_j} \tag{9}$$

The per-observation evidence is $e_j^{(t)} = \xi_j \in \mathbb{R}$. Its sign records whether the neuron's weighted output is aligned with or opposed to the module's output direction for that observation. Sign fluctuation across T_c is naturally handled by the Bayesian aggregation stage (§2.3), where unstable evidence increases posterior uncertainty. Because the module update is an exact linear sum of per-neuron writes, the signed scores partition the module's output norm rather than merely approximating importance, which is why alignment to the realized module direction serves as a faithful per-observation measure of functional contribution; Appendix B develops this argument in full. Appendix C details how these observations are logged for experiments.

2.3 Evidence Aggregation

We aggregate the noisy per-observation evidence $\{e_j^{(t)}\}_{t=1}^n$ over the selected domain set D_c into calibrated consistency estimates via a NIG conjugate model, where $n = |T_c|$ is the induced token-position evidence count for the fixed domain.

2.3.1 Modeling Functional Consistency

For each neuron j in a given module at layer l, we model the signed alignment scores as:

$$e_j^{(t)} \sim \mathcal{N}(\mu_j, \sigma_j^2), \quad t = 1, \ldots, n \tag{10}$$

where n counts all token-position observations induced by the N input samples in D_c. Treating the Gaussian likelihood as a conjugate working model for the evidence mean and dispersion, we place a conjugate NIG prior $(\mu_j, \sigma_j^2) \sim \text{NIG}(\mu_0, \lambda_0, \alpha_0, \beta_0)$, where $\sigma_j^2 \sim \text{Inv-Gamma}(\alpha_0, \beta_0)$ and $\mu_j \mid \sigma_j^2 \sim \mathcal{N}(\mu_0, \sigma_j^2/\lambda_0)$.

This conjugate specification yields analytic posterior updates for each neuron.

2.3.2 Posterior Updates

The closed-form update produces two quantities used by RACE scoring. The first is a posterior mean $\mu_{n,j}$ that estimates signed alignment strength. The second is a posterior uncertainty term that penalizes noisy or scarce evidence.

Sufficient Statistics. Given n alignment scores $\{e_j^{(t)}\}_{t=1}^n$ for neuron j, the posterior update requires only the empirical mean and centred sum of squares:

$$\bar{e}_j = \frac{1}{n}\sum_{t=1}^n e_j^{(t)}, \quad \text{SS}_j = \sum_{t=1}^n \left(e_j^{(t)} - \bar{e}_j\right)^2, \tag{11}$$

from which the empirical standard deviation is $\hat{\sigma}_j = \sqrt{\text{SS}_j/(n-1)}$.

Posterior Update. The NIG conjugacy yields a closed-form posterior with the same functional form:

$$(\mu_j, \sigma_j^2) \mid \{e_j^{(t)}\} \sim \text{NIG}(\mu_{n,j}, \lambda_n, \alpha_n, \beta_{n,j}) \tag{12}$$

where the posterior hyperparameters are updated via simple arithmetic:

$$\lambda_n = \lambda_0 + n, \tag{13}$$
$$\alpha_n = \alpha_0 + \tfrac{n}{2}, \tag{14}$$
$$\mu_{n,j} = \frac{\lambda_0 \mu_0 + n\bar{e}_j}{\lambda_n}, \tag{15}$$
$$\beta_{n,j} = \beta_0 + \tfrac{\mathrm{SS}_j}{2} + \frac{\lambda_0 n(\bar{e}_j - \mu_0)^2}{2\lambda_n}. \tag{16}$$

Maintaining these sufficient statistics costs $O(1)$ per neuron per observation. The posterior parameters are then obtained in closed form.

Posterior Mean Evidence. The group-level score is the signed posterior mean, directly instantiating Eq. (3):

$$\Phi_c(j) = \mu_{n,j} = \frac{\lambda_0 \mu_0 + n\bar{e}_j}{\lambda_0 + n} \tag{17}$$

The sign records the dominant alignment direction of the evidence, while values near zero indicate weak or inconsistent average alignment. The next subsection converts this posterior mean and its marginal uncertainty into CAM.

2.4 Uncertainty Quantification

Posterior Uncertainty over Mean Evidence. Integrating out σ_j^2, the marginal posterior for the mean evidence follows a Student's t-distribution:

$$\mu_j \mid \{e_j^{(t)}\} \sim t_{2\alpha_n}(\mu_{n,j},\ \sigma_{\mu,j}), \quad \sigma^2_{\mu,j} = \frac{\beta_{n,j}}{\alpha_n \lambda_n} \tag{18}$$

where the second argument is the Student's t scale parameter. This scale $\sigma_{\mu,j}$ quantifies uncertainty regarding the average signed contribution, producing wider intervals for scarce or noisy evidence.

Variance Sources. The same $\beta_{n,j}$ term also determines the posterior expected observation variance, $\mathbb{E}[\sigma_j^2 \mid \{e_j^{(t)}\}] = \beta_{n,j}/(\alpha_n - 1)$ for $\alpha_n > 1$. Its data-driven component $\mathrm{SS}_j/2$ captures cross-observation variability in the alignment evidence. The prior-data term $\lambda_0 n(\bar{e}_j - \mu_0)^2/(2\lambda_n)$ regularizes evidence relative to the neutral prior $\mu_0 = 0$. Thus bursty, noisy, or sign-fluctuating evidence inflates $\beta_{n,j}$, increasing $\sigma_{\mu,j}$ and reducing confidence in the neuron's mean alignment strength. As n grows, λ_n and α_n increase, shrinking the posterior uncertainty over μ_j when the evidence remains stable.

Conservative Alignment Magnitude. We define a scoring criterion that provides a one-sided conservative lower credible bound on each neuron's positive module-output alignment. The default RACE score is:

$$\begin{aligned} \rho_j(\gamma) &= \mu_{n,j} - t_{1-\gamma,\,2\alpha_n}\sigma_{\mu,j}, \\ \mathrm{CAM}_j(\gamma) &= \mathbb{I}[\mu_{n,j} > 0]\ \max(0, \rho_j(\gamma)) \end{aligned} \tag{19}$$

where $t_{1-\gamma,\,2\alpha_n}$ is the $(1-\gamma)$-quantile of the Student's t with $2\alpha_n$ degrees of freedom. Formally, $\mathrm{CAM}_j(\gamma)$ is the lower $(1-\gamma)$-credible bound on the positive posterior mean evidence under the marginal Student's t-posterior in Eq. (18). Neurons with large positive mean evidence and small posterior uncertainty receive high CAM scores, whereas neurons with few observations, unstable alignment, or negative posterior mean evidence are excluded or penalized through the confidence radius. Throughout the paper, **RACE** denotes the default CAM ranking. As an ablation, we also report Neg. $\mathrm{CAM}_j(\gamma) = \mathbb{I}[\mu_{n,j} < 0]\max(0, -\mu_{n,j} - t_{1-\gamma,\,2\alpha_n}\sigma_{\mu,j})$.

2.5 Reference-Set Filtering

Because features represented in superposition can make individual neurons polysemantic (Elhage et al., 2022; Scherlis et al., 2022; Templeton et al., 2024), unfiltered target rankings may include neurons that support broad capabilities rather than neurons whose behavior is specific only to the target domain. To disentangle domain-specific behavior from general capabilities during targeted interventions, we introduce RSF as an explicit filtering step. We denote direct RACE scoring on dataset D by $\mathbf{R}_D$, and reference-set filtered scoring by $\mathbf{R}_{D_{\mathrm{tar}} \setminus D_{\mathrm{ref}}}$. For each layer ℓ and module type, let U^ℓ be the layer-local neuron universe, $B^\ell_{\mathrm{ref}} = \mathrm{Top}^{s_{\mathrm{ref}}}_M(U^\ell)$ be a reference exclusion set, and Π^ℓ_{tar} be the full target-domain ranking. RSF selects

$$\begin{aligned} \Pi^\ell_{\mathrm{RSF}} &= \left[j \in \Pi^\ell_{\mathrm{tar}} : j \notin B^\ell_{\mathrm{ref}} \right], \\ S^\ell_{\mathrm{spec}} &= \mathrm{First}_k\left(\Pi^\ell_{\mathrm{RSF}}\right). \end{aligned} \tag{20}$$

Here M is the reference top-M or top-p-percent threshold. Operationally, RSF traverses the target ranking, skips reference-selected neurons, and stops when k neurons are selected; this preserves exactly k neurons per layer whenever $|U^\ell \setminus B^\ell_{\mathrm{ref}}| \geq k$. Appendix D provides a cross-domain overlap analysis.

3 Experiments

We evaluate RACE across multiple domains and models. Our experiments address two main questions: (1) **Efficacy**: Do RACE-selected neurons

cause a disproportionate performance drop on target domains versus non-target domains when suppressed? (2) **Ablation**: Does Bayesian uncertainty quantification (CAM) outperform deterministic scoring heuristics?

3.1 Experimental Setup

Evaluated Models. We evaluate on Qwen3-4B-it (Qwen Team, 2025) (`Qwen3-4B-it-2507`), OLMo-3.1-32B-it (Team OLMo et al., 2025), and Llama-3.1-8B-it (AI at Meta, 2024) (Appendix E); full model details are in Appendix F.

Domain & Benchmark Settings. Each domain is instantiated by a scoring set for RACE-based neuron selection, a same-domain out-of-distribution (OOD) benchmark for validation, and non-target benchmarks for retention.

- **Code:** MBPP+ (Austin et al., 2021; Liu et al., 2023) for scoring and HumanEval+ (Chen et al., 2021; Liu et al., 2023) for OOD validation.
- **Math:** MATH-500 (Hendrycks et al., 2021) for scoring and AMC (EvalScope, 2024) for OOD validation.
- **Fine-grained Behavioral:** We construct `PyComp-1K`, a set of 1,000 AST-verified Python comprehension-containing statements from `bigcode/the-stack` (Kocetkov et al., 2022; BigCode, 2026), as a narrow code-behavior scoring domain (Appendix I).

Detailed benchmark evaluation setup, including evaluator configurations and scoring definitions, is provided in Appendix F. Evidence logging strategies on corpora are summarized in Appendix C.

RACE Hyperparameters. Unless otherwise specified, all experiments use the same RACE hyperparameter settings: $\mu_0 = 0$, $\lambda_0 = 1$, $\alpha_0 = 1$, $\beta_0 = 1$, and $\gamma = 0.05$. We further discuss the robustness to prior settings and confidence levels in Appendix K.

Baselines & Ablations. Table 1 reports the scoring formula used by each baseline or ablation. We consider two groups of methods: external baselines and internal RACE ablations. For external baselines, GxAct (Kokhlikyan et al., 2020) and AttnLRP (Achtibat et al., 2024) serve as typical gradient attribution methods over the same T_c, with the attribution target set to the benchmark answer token. Act. Mean serves as an activation-only control. To isolate the effect of Bayesian uncertainty modeling, we introduce three internal ablations operating on the same evidence $\{e_j^{(t)}\}_{t=1}^{n}$ as RACE. Emp. Mean removes both the uncertainty penalty and prior regularization, thereby reducing to the raw empirical average. Emp. SNR replaces the Bayesian penalty with a frequentist variance penalty computed via $\hat{\sigma}_j = \sqrt{\mathrm{SS}_j/(n-1)}$, and Neg. CAM acts as a negative-direction ablation. Regarding the evaluation protocol, instance-level scores are lifted to domain-level neuron rankings by averaging positive neuron-level contributions over the induced token-position observation set T_c, matching RACE's evidence aggregation granularity. All methods adopt identical per-layer/per-module selection budgets and suppression protocols. Under RSF, target and reference rankings are computed using the same method-specific scoring rule.

Table 1: Baselines and ablations. Each row gives the neuron score S_j used for ranking neuron j, with $n = |T_c|$. $\mathrm{GxAct}_j^{(t)}$ and $\mathrm{LRP}_j^{(t)}$ denote per-observation attribution scores for the two external baselines.

Method	**Neuron score S_j**
GxAct	$1/n \sum_{t \in T_c} \max(0, \mathrm{GxAct}_j^{(t)})$
AttnLRP	$1/n \sum_{t \in T_c} \max(0, \mathrm{LRP}_j^{(t)})$
Act. Mean	$1/n \sum_{t \in T_c} \lvert a_j^{(t)} \rvert$
Neg. CAM	$\mathbb{I}[\mu_{n,j} < 0] \max(0, -\mu_{n,j} - t_{1-\gamma, 2\alpha_n} \sigma_{\mu,j})$
Emp. Mean	$\bar{e}_j$
Emp. SNR	$\max(0, \bar{e}_j)/(\hat{\sigma}_j + \varepsilon)$

3.2 Depth-Wise Organization of CAM Scores

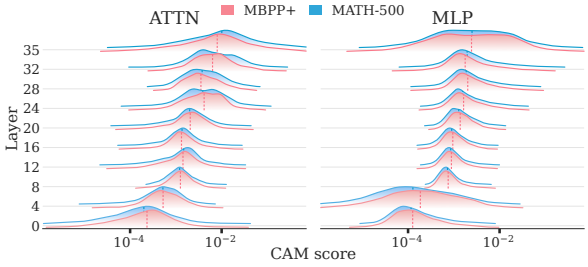

Figure 1: Layer-wise kernel density estimate (KDE) ridgeline distributions of RACE results. On Qwen3-4B-it, ridges under $\mathbf{R}_{\text{MBPP+}}$ and $\mathbf{R}_{\text{MATH-500}}$ visualize the KDE of CAM scores for neurons in each module across selected layers.

Figure 1 visualizes the CAM RACE score landscape across layers and modules. The ridgelines reveal a progressive rightward shift in positive CAM density with increasing depth, showing that high-confidence evidence concentrates in later layers. Compared to the shallow and deep layers, the mid-

dle layers exhibit a sparser distribution of high-scoring neurons, especially in MLP modules. This depth-wise organization is remarkably consistent across MBPP+ and MATH-500, suggesting it reflects broad architectural properties rather than domain-specific quirks.

Consistent with prior analyses on Transformers' functionally stratified computation (Tenney et al., 2019; Jawahar et al., 2019; Geva et al., 2022, 2023), this observation implies that CAM scores are not directly comparable across layers. A globally sorted list would be dominated by late-layer neurons, conflating alignment magnitude with network depth. We therefore adopt a stratified selection protocol in the following interventions: neurons are ranked by CAM within each layer and module, and a fixed budget is allocated to every layer. This per-layer selection rule preserves coverage over the model's hierarchical computation.

3.3 Effectiveness Validation

We validate the intervention relevance of the selected neurons through targeted suppression: during inference, we set their corresponding activation values to zero and measure the resulting performance changes.

To jointly quantify target-domain suppression effectiveness and general selectivity in a single scalar, we define the **Intervention Specificity Index (ISI)**:

$$\mathrm{ISI} = \max\left\{0, \log\left(\frac{\Delta_T}{\Delta_G + \delta}\right)\right\}, \quad \delta = 0.01. \tag{21}$$

Here Δ_T and Δ_G are the average nonnegative relative accuracy drops over the target-domain (including OOD benchmarks) and general (non-target) benchmarks respectively, computed from the per-benchmark drop $\Delta = \max\{0, (\mathrm{Acc}_{\mathrm{before}} - \mathrm{Acc}_{\mathrm{after}})/\mathrm{Acc}_{\mathrm{before}}\}$.

The constant δ is used for smoothing. Higher ISI indicates stronger target-domain suppression with minimal non-target degradation.

3.3.1 Domain-Specific Intervention

We evaluate two intervention strategies. The **Vanilla Selection** strategy relies solely on the CAM scores computed on the target domain. In contrast, the **RSF** strategy (§2.5) explicitly controls for broadly shared language capabilities by filtering the candidate set against a general reference corpus.

Vanilla Selection. Table 2 reports the Code-domain results; the corresponding Qwen3-4B-it Math-domain results are provided in Appendix Table 21. We find that neurons selected under this setting cause severe overall generation degradation once the suppression budget reaches $k \geq 10$ per layer. We thus report $k=5$ to observe distinct effects.

Table 2: Code domain with $\mathbf{R}_{\text{MBPP+}}$: Benchmark Acc. (%) and ISI on Qwen3-4B-it after suppressing top-k=5 neurons per layer. Superscript ♠ marks the target domain D, and ♡ marks the same-domain OOD benchmark. All reported scores are averaged over three runs; subsequent tables follow the same setting unless specified.

Module	Method	MBPP+♠	HumanEval+♡	MATH-500	AMC	MMLU-Redux	GPQA	ISI↑
ATTN	GxAct	57.94	49.39	76.35	42.54	77.79	44.95	0.57
	AttnLRP	47.09	43.90	90.25	77.61	76.79	45.96	2.00
	Act. Mean	80.69	46.34	93.89	81.34	80.82	41.92	1.54
	Emp. Mean	78.31	51.83	93.24	84.33	80.47	39.97	1.37
	Emp. SNR	74.34	50.00	93.13	84.33	81.14	41.41	1.71
	Neg. CAM	78.84	83.54	93.77	83.58	80.53	41.92	0.00
	RACE	55.03	45.73	89.00	72.39	78.91	44.95	1.62
MLP	GxAct	10.32	5.49	1.80	0.75	15.25	15.15	0.04
	AttnLRP	74.87	83.54	95.04	85.82	81.84	49.49	1.15
	Act. Mean	81.75	3.66	93.42	81.34	81.18	41.47	2.22
	Emp. Mean	75.13	0.00	89.86	87.31	81.33	43.43	2.79
	Emp. SNR	80.95	86.59	93.62	82.09	81.23	45.96	0.00
	Neg. CAM	75.13	85.37	91.81	84.33	81.58	45.96	0.54
	RACE	74.34	0.00	92.80	82.83	81.79	44.95	2.91
Qwen3-4B-it		82.28	83.54	94.40	87.31	81.37	45.45	—

RSF Selection. On Qwen3-4B-it, Figure 2 summarizes the RSF intervention results for both the code and math domains. The corresponding tabular details are reported in Appendix Tables 20 and 22. To demonstrate that RACE's effectiveness scales to larger models, we further evaluate $\mathbf{R}_{\text{MATH-500}\setminus\text{WikiText-2}}$ on OLMo-3.1-32B-it.

Table 3: Math domain with $\mathbf{R}_{\text{MATH-500}\setminus\text{WikiText-2}}$: Benchmark Acc. (%) and ISI on OLMo-3.1-32B-it after suppressing top-1% and top-5% neurons per layer.

		MATH-500♠		AMC♡		GPQA		MMLU-Redux		ISI↑	
Module	Method	1%	5%	1%	5%	1%	5%	1%	5%	1%	5%
ATTN	Act. Mean	88.32	84.85	76.12	71.64	38.10	35.99	79.91	73.18	0.00	0.00
	Emp. Mean	85.01	87.84	70.93	68.66	45.03	43.52	81.54	77.12	0.00	0.00
	Emp. SNR	86.41	86.25	71.64	67.91	44.02	44.54	83.47	79.95	0.00	0.00
	Neg. CAM	85.97	86.62	77.61	73.13	47.56	45.53	84.01	84.67	0.00	0.01
	RACE	85.61	84.43	76.86	65.67	47.74	42.49	82.63	80.35	0.00	0.00
MLP	Act. Mean	63.87	14.71	21.64	10.45	40.40	36.36	81.32	75.65	1.47	1.50
	Emp. Mean	73.95	21.43	25.37	13.43	41.41	38.89	80.81	73.46	1.35	1.50
	Emp. SNR	76.89	44.54	23.88	18.66	23.74	21.72	52.81	46.21	0.00	0.20
	Neg. CAM	92.40	97.48	79.85	79.85	52.02	50.51	84.32	85.16	0.00	0.00
	RACE	56.80	6.72	14.18	7.46	39.83	37.79	82.72	77.79	1.65	1.73
OLMo-3.1-32B-it		86.40		79.85		48.6		84.70		—	

Observation: RACE consistently outperforms gradient-based methods, confirming RDA as a reliable gradient-free evidence source. The pronounced OOD degradation further validates RACE scores as an effective target-domain proxy. RSF substantially improves neuron suppression tolerance, with target performance dropping more

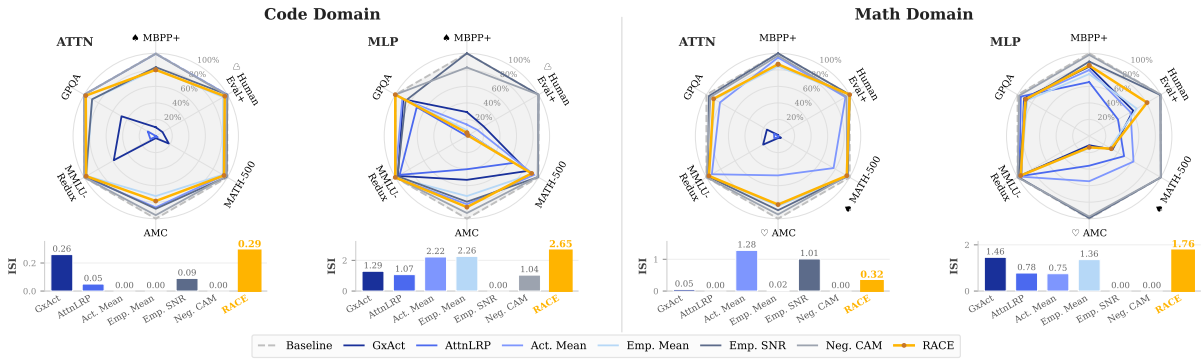

Figure 2: Suppression under the RSF strategies $\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$ for the code domain and $\mathbf{R}_{\text{MATH-500}\setminus\text{WikiText-2}}$ for the math domain on Qwen3-4B-it. All interventions suppress the top-1% target-selected neurons within the corresponding module at each layer. Radar plots report post-suppression benchmark accuracy as a percentage of the original model baseline, separately for ATTN and MLP interventions. Bars below each radar report the corresponding ISI.

steeply under higher budgets—indicating successful isolation of domain-specific neurons.

Figure 2 confirms this specificity: for both code and math, RACE-selected suppression under RSF hurts target-domain more than non-target performance. However, effects vary by module: MLP interventions cause significant same-domain OOD drops, while attention interventions show weaker domain-specific degradation.

This asymmetry is clearer when comparing vanilla selection to RSF: Under vanilla selection, ATTN interventions produce marked target drops even at low budgets, but RSF's ISI for ATTN remains substantially lower than for MLP, especially on the 32B model. We attribute this to ATTN's high-scoring neurons being sparse yet broadly reusable: RSF removes these general-purpose neurons via reference filtering, leaving a weaker domain-specific signal. This aligns with known sparsity in ATTN's $\mathbf{W}_O$ (Michel et al., 2019), suggesting ATTN's target-domain high-activation neurons also serve broader linguistic functions.

3.3.2 Distributional Verification

Table 4: Distributional disruption on Qwen3-4B-it under $\mathbf{R}_{D_{\text{tar}}\setminus\text{WikiText-2}}$, after suppressing the top-1% neurons within the selected module at each layer.

		(a) $\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$			(b) $\mathbf{R}_{\text{MATH-500}\setminus\text{WikiText-2}}$		
Metric	**Module**	**MBPP+♠**	**MATH-500**	**WikiText-2**	**MBPP+**	**MATH-500♠**	**WikiText-2**
Δ_{PPL}	ATTN	+6.17%	+6.18%	−2.49%	+4.52%	+7.29%	−2.74%
	MLP	+77.31%	+23.98%	+2.76%	+25.41%	+129.04%	+2.60%
$\bar{D}_{\text{KL}}$	ATTN	0.10	0.08	0.03	0.09	0.09	0.03
	MLP	0.58	0.22	0.02	0.22	0.86	0.03

Beyond coarse accuracy drops, we examine the mechanism of disruption at the token-distribution level using relative perplexity degradation Δ_{PPL} (Eq. 27) and mean forward Kullback–Leibler divergence $\bar{D}_{\text{KL}}$ (Eq. 28).

The distributional metrics (Table 4) reveal a drastic asymmetric disruption, particularly for MLP interventions. Suppressing RACE-selected MLP neurons triggers severe target-domain distribution collapse (Δ_{PPL} reaching 77.31% on MBPP+ and 129.04% on MATH-500, alongside massive $\bar{D}_{\text{KL}}$ shifts), while leaving the reference corpus (WikiText-2) almost entirely unperturbed. This targeted disruption validates the effectiveness of RACE in decoupling and localizing function-specific neurons without collapsing the model's fundamental linguistic competence. As a byproduct, ATTN interventions exhibit significantly weaker distributional contrasts, suggesting lower functional sparsity and weaker sensitivity to perturbations compared to the MLP module. Appendix M reports w/o-RSF distributional disruption results.

3.3.3 Fine-Grained Behavioral Steering

Table 5: Fine-grained suppression of Python comprehension generation on Qwen3-4B-it. **Records**: outputs containing ≥ 1 comprehension. **Total**: aggregate comprehension instances. **Pass%**: pass@1 functional correctness on the subset of generated solutions that use comprehensions.

	MBPP+			HumanEval+		
Strategy	**Records**	**Total**	**Pass%**	**Records**	**Total**	**Pass%**
Qwen3-4B-it	55	59	92.73	36	45	86.11
$\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$	44	56	97.73	33	46	81.82
$\mathbf{R}_{\text{PyComp-1K}\setminus\text{WikiText-2}}$	$16_{-70.9\%}$	$18_{-69.5\%}$	87.50	$14_{-61.1\%}$	$19_{-57.8\%}$	85.71

To test RACE's resolution on narrow stylistic behaviors, we target Qwen3-4B-it's generation of

Python comprehensions. These are semantically optional but syntactically idiomatic constructs.

Using PyComp-1K as the narrow scoring set, we apply $\mathbf{R}_{\text{PyComp-1K}\setminus\text{WikiText-2}}$ and $\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$ with a small intervention budget of $k{=}10$ neurons per layer.

Suppressing PyComp-1K neurons drastically reduces comprehension usage (-70.9% on MBPP+, -61.1% on HumanEval+) while largely preserving functional correctness (Table 5). This specific decline without catastrophic forgetting confirms that RACE-selected neurons steer behavior rather than cause mere disruption. The generation-level effect of this targeted perturbation suggests that RACE can use a deliberately designed dataset to localize a specific model behavior. Appendix J provides a comparison of model outputs before and after perturbation.

3.4 Computational Efficiency

Table 6 reports the additional FLOPs required by each scoring method, measured with a profiler on Qwen3-4B-it using 64 input tokens and 16 generated tokens. Appendix H gives the profiling protocol.

Table 6: Profiler-measured scoring overhead relative to forward inference on Qwen3-4B-it.

Method	Extra FLOPs	Forward-equivalent overhead
GxAct	18.77 TFLOPs	144.29×
AttnLRP	18.82 TFLOPs	144.62×
RDA	81.5 GFLOPs	0.63×

3.5 Discussion

We analyze how scoring-set size and variance regularization affect RACE, using Qwen3-4B-it with MBPP+ unless stated otherwise.

Scoring-set sample size N. The scoring-set sample size N controls how reliably RACE estimates intervention-worthy neurons from the target domain. Figure 3 evaluates this effect through downstream ISI rather than ranking correlation: for each N, we construct $\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$, suppress the top-1% MLP neurons per layer, and compare RACE with Emp. Mean. The ISI curves are non-monotonic under small and mid-sized scoring sets, but both methods improve as more MBPP+ examples are used and reach their strongest selectivity on the full scoring set ($N{=}378$). Notably, under the default prior, the posterior mean is a monotonic rescaling of the empirical mean ($\mu_{n,j} \propto \bar{e}_j$).

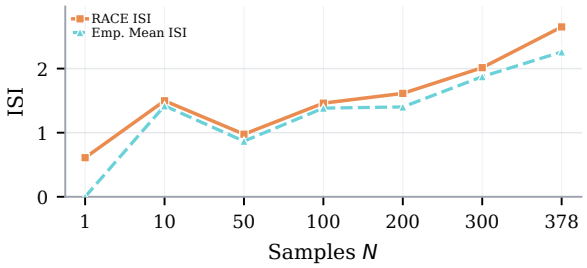

Figure 3: Sample efficiency on Qwen3-4B-it under $\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$: ISI after suppressing top-1% MLP neurons per layer as the scoring-set size N varies, comparing RACE with Emp. Mean.

The finite-sample difference between RACE and empirical averaging therefore arises from CAM's uncertainty penalty.

Bayesian variance regularization. The sample-size behavior clarifies that RACE's low-data advantage comes from calibrated uncertainty rather than a different mean estimator. Sparse module evidence traces can make empirical variance-based ratios over-rank weak nonzero fluctuations, consistent with the poor MLP intervention specificity of Emp. SNR in Table 2. CAM avoids this failure mode by retaining a finite uncertainty margin under the default NIG prior. It becomes prior-insensitive once the induced evidence count n is large. Appendix P gives the short derivation.

4 Related Work

Current research in mechanistic interpretability primarily focuses on analyzing individual instances through gradient-based methods, residual stream analysis (Belrose et al., 2023; Ghandeharioun et al., 2024), and circuit discovery (Conmy et al., 2023; Gu et al., 2025). While sparse autoencoders (Bricken et al., 2023; Templeton et al., 2024) offer detailed feature-level analysis, scaling these methods for dataset-level evaluation remains computationally challenging. Recent work has identified specific neural patterns linked to particular skills (Wang et al., 2022; Song et al., 2024; Leng and Xiong, 2025), languages (Tang et al., 2024; Zhang et al., 2024), and factual knowledge (Dai et al., 2021; Meng et al., 2022; Chen et al., 2024). However, existing approaches often struggle to reliably separate meaningful patterns from random noise (Niu et al., 2024), limiting their effectiveness for applications like model pruning (Ma et al., 2023; Sun et al., 2024) or behavior control (Turner et al., 2024; Zou et al., 2023). RACE addresses

these limitations with a more robust statistical framework for model auditing.

5 Conclusion

To assess functional consistency, RACE recasts neuron analysis as inference over latent alignment distributions. The resulting posterior isolates neurons whose domain contributions are directionally aligned and statistically stable. Targeted perturbations confirm this population is causally load-bearing. Designed with linear computational complexity, the approach remains practical for real-world applications.

Limitations

We discuss the currently identified limitations in the formulation of RACE. The extraction of alignment evidence relies on linear residual-stream projections, which means the framework may miss complex non-linear synergies among multiple neurons or distributed polysemantic features that require non-linear decoding. Additionally, our experimental results indicate that RACE, alongside other evaluated baselines, demonstrates limited effectiveness in identifying functionally consistent neurons within attention modules. This suggests that attention mechanisms may encode information in a fundamentally different manner than MLPs, making the development of statistical audit strategies specifically tailored to attention an important direction for future work.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 52202496, 52442218, and U2433216; The Key Research and Development Project of Nantong City, China (Special Project for Prospective Technology Innovation, No. GZ2024001); and the Key Laboratory of Target Cognition and Application Technology (2023-CXPT-LC-005).

References

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

AI at Meta. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *Advances in Neural Information Processing Systems*, 36.

BigCode. 2026. The Stack: BigCode dataset documentation. https://www.bigcode-project.org/docs/about/the-stack/. Accessed 2026-05-14.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Ramesh Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Journey to the center of the knowledge neurons: Discoveries of language-independent knowledge neurons and degenerate knowledge neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17817–17825.

Arthur Conmy, Augustine N Mavor-Parker, Aidan Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Chris Olah, and 1 others. 2022. Toy models of superposition. *Transformer Circuits Thread*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson

Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

EvalScope. 2024. AMC: American mathematics competitions benchmark. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/amc.html. Benchmark based on AMC 10/12 problems from 2022–2024.

EvalScope. 2026a. ARC: EvalScope benchmark card. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/arc.html. Accessed 2026-05-14.

EvalScope. 2026b. GPQA-Diamond: EvalScope benchmark card. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/gpqa_diamond.html. Accessed 2026-05-14.

EvalScope. 2026c. HumanEvalPlus: EvalScope benchmark card. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/humaneval_plus.html. Accessed 2026-05-14.

EvalScope. 2026d. MATH-500: EvalScope benchmark card. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/math_500.html. Accessed 2026-05-14.

EvalScope. 2026e. MBPP-Plus: EvalScope benchmark card. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/mbpp_plus.html. Accessed 2026-05-14.

EvalScope. 2026f. MMLU-Redux: EvalScope benchmark card. https://evalscope.readthedocs.io/zh-cn/latest/benchmarks/mmlu_redux.html. Accessed 2026-05-14.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15466–15490. PMLR.

Hao Gu, Vibhas Nair, Amrithaa Ashok Kumar, Jayvart Sharma, and Ryan Lagasse. 2025. Discovering transformer circuits via a hybrid attribution and pruning framework. *arXiv preprint arXiv:2510.03282v1*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Ko-Wei Huang, Yi-Fu Fu, Ching-Yu Tsai, Yu-Chieh Tu, Tzu-ling Cheng, Cheng-Yu Lin, Yi-Ting Yang, Heng-Yi Liu, Keng-Te Liao, Da-Cheng Juan, and Shou-De Lin. 2025. Neuron-level differentiation of memorization and generalization in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16066–16080, Suzhou, China. Association for Computational Linguistics.

Jett Janiak, Can Rager, James Dao, and Yeu-Tong Lau. 2024. An adversarial example for direct logit attribution: Memory management in gelu-4l. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 232–237.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The Stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch. *CoRR*, abs/2009.07896.

Yongqi Leng and Deyi Xiong. 2025. Towards understanding multi-task learning (generalization) of

LLMs via detecting and exploring task-specific neurons. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2969–2987, Abu Dhabi, UAE. Association for Computational Linguistics.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *arXiv preprint arXiv:2202.05262*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? In *International Conference on Learning Representations*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*.

Adam Scherlis, Kshitij Sachan, Adam S. Jermyn, Joe Benton, and Buck Shlegeris. 2022. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMlR.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1690–1712, Suzhou, China. Association for Computational Linguistics.

Ran Song, Shizhu He, Shuting Jiang, Yantuan Xian, Shengxiang Gao, Kang Liu, and Zhengtao Yu. 2024. Does large language model contain task-specific neurons? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7101–7113, Miami, Florida, USA. Association for Computational Linguistics.

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

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Tianyi Tang, Wenyang Luo, Haoyang Huang, Dongdong Zhang, Xiaolei Wang, Xin Zhao, Furu Wei, and Ji-Rong Wen. 2024. Language-specific neurons: The key to multilingual capabilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5701–5715, Bangkok, Thailand. Association for Computational Linguistics.

Team OLMo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. OLMo 3. *Preprint*, arXiv:2512.13961.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C Daniel Freeman, Theodore R Sumers, Edward Rees, Joshua Batson, Adam Jermyn, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Ulisse Mini, and Monte MacDiarmid. 2024. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2024. Neurons in large language models: Dead, n-gram, positional. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1288–1301, Bangkok, Thailand. Association for Computational Linguistics.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*,

pages 11132–11152, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zeping Yu and Sophia Ananiadou. 2024. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280.

Zhihao Zhang, Jun Zhao, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Unveiling linguistic regions in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6228–6247, Bangkok, Thailand. Association for Computational Linguistics.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Trans. Intell. Syst. Technol.*, 15(2):20:1–20:38.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2023. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*.

A Why Use a Module-Local Axis Instead of a Global Logit Axis

A natural alternative to RDA is to score every neuron against a single global direction derived from the final next-token prediction. For example, one could replace the module-local axis in Eq. (7) with a normalized unembedding or contrastive logit direction $\hat{\mathbf{g}}_t$, and score a neuron by

$$e_j^{\text{global}}(t) = a_j^{(t)} \mathbf{w}_j^\top \hat{\mathbf{g}}_t, \tag{22}$$

where $\mathbf{w}_j$ is the neuron's output vector and $a_j^{(t)}$ is its activation at observation t. This resembles vocabulary-projection analyses such as the logit lens and direct logit attribution. RACE instead uses the module-local residual update $\Delta\mathbf{r}_{l,m}^{(t)}$ as the evaluation direction because the global-logit alternative answers a different question. It asks whether a neuron already points toward the final prediction, whereas RACE asks whether the neuron consistently contributes to the computation being performed by its own module at its own depth.

Depth-local computations are not exchangeable. Prior work repeatedly shows that Transformer layers are functionally stratified rather than interchangeable. In BERT, lower layers capture phrase-level or surface information, middle layers capture syntactic structure, and upper layers capture more semantic information (Jawahar et al., 2019). Tenney et al. similarly find a localized progression resembling the classical NLP pipeline, from POS tagging and parsing through semantic roles and coreference (Tenney et al., 2019). For decoder language models, Geva et al. show that FFN memories differ across depth: lower layers tend to match shallower patterns, while upper layers encode more semantic patterns and more directly induce output-vocabulary distributions (Geva et al., 2021). Follow-up work further views FFN outputs as additive updates to a changing vocabulary distribution (Geva et al., 2022), and factual-recall analyses identify distinct early-MLP enrichment and later information-routing phases (Geva et al., 2023). These results imply that an intermediate neuron can be important because it constructs, routes, erases, or transforms information that is not yet aligned with the final answer token. A final-logit axis therefore imposes a late-stage semantic criterion on layers whose local role may be lexical, syntactic, relational, or preparatory.

Raw logit projections are diagnostic tools, not layer-invariant objectives. Lens-style methods are useful precisely because they expose how predictions evolve across depth, but their behavior also reveals the limits of using the final unembedding as a universal coordinate system. The tuned lens was introduced as a refinement of the logit lens because the raw logit lens is often brittle; the tuned lens learns a separate affine translator for each layer and is reported to produce more predictive, reliable, and less biased intermediate predictions (Belrose et al., 2023). Patchscopes reaches a similar conclusion from another direction: many vocabulary-projection methods can be viewed as special cases of a broader representation-inspection framework, and their shortcomings include failures in early layers and limited expressivity (Ghandeharioun et al., 2024). If intermediate states required no layer-specific decoding correction, these layer-wise translators or richer patching contexts would be unnecessary. Thus, raw alignment with the final unembedding is best interpreted as a probe of whether a layer has become linearly readable in vocabulary space, not as a universal measure of neuron importance.

Direct logit attribution can mis-rank intermediate components. Direct logit attribution projects intermediate residual components onto final logit directions, but this projection ignores the fact that later layers can overwrite, rotate, or erase earlier residual directions. Janiak et al. provide a concrete adversarial example in GELU-4L: the model uses a memory-management mechanism in which later heads and MLPs remove directions written by earlier heads, and DLA becomes misleading because it does not account for this erasure (Janiak et al., 2024). For RACE, this failure mode is especially relevant. A globally positive logit projection at layer l may be a transient direction that later modules remove, while a globally weak projection may be a necessary intermediate feature that later modules transform into the final prediction. Ranking neurons by final-logit projection would therefore mix three factors: local functional contribution, survival through subsequent computation, and proximity to the output head.

Global axes conflate alignment magnitude with depth. Because later representations are closer to the output distribution, a global logit axis tends to favor late-layer neurons whose effects have already been rotated into vocabulary-readable directions.

This creates a numerical depth bias: the sorted list of "important" features can be dominated by late-layer units even when earlier and middle layers contain indispensable local computations. Such a ranking confounds two quantities that RACE keeps separate: the strength of a neuron's contribution to its module's current update, and the downstream fate of that update after many additional nonlinear transformations.

Implication for RACE. The module-local axis does not claim that local alignment alone proves final-output causality. It deliberately produces a stable, layer-appropriate observation variable for population-level statistical inference. Final behavioral relevance is then tested separately through targeted suppression and reference-set filtering. This separation matches the evidence from prior work: intermediate layers perform different computations, raw logit projections require careful layer-specific interpretation, and direct logit attribution can be misleading when later layers erase or transform earlier residual directions.

B Why Residual Alignment Measures Functional Role

We first state two formal properties of the RDA evidence that anchor the discussion, then supply the conceptual premises that make them carry functional meaning and close the gap between per-observation geometry and the population-level definition of functional consistency.

Two formal properties of the alignment evidence. Writing each neuron's contribution as $\mathbf{v}_j = a_j \mathbf{w}_j$ gives the exact linear sum $\Delta\mathbf{r} = \sum_j \mathbf{v}_j$; with the realized output direction $\hat{\mathbf{d}} = \Delta\mathbf{r}/\|\Delta\mathbf{r}\|_2$ (Eq. 7), the RDA score is $e_j = \langle \mathbf{v}_j, \hat{\mathbf{d}} \rangle$. Because the decomposition is exact, the scores satisfy a *completeness identity*—each write splits into an aligned part and an orthogonal part that cancels in aggregate:

$$\begin{aligned} \mathbf{v}_j &= e_j \hat{\mathbf{d}} + \mathbf{q}_j, & \mathbf{q}_j &\perp \hat{\mathbf{d}}, \\ \sum_j \mathbf{q}_j &= \mathbf{0}, & \sum_j e_j &= \|\Delta\mathbf{r}\|_2, \end{aligned} \tag{23}$$

so the scores do not approximate importance but *partition* it: $\sum_j e_j$ exhausts the module's output norm. The score is also the *local-sensitivity* of that norm to rescaling the neuron: with $\Delta\mathbf{r}_j(\alpha) = \Delta\mathbf{r} + (\alpha - 1)\mathbf{v}_j$,

$$\frac{\partial}{\partial \alpha} \|\Delta\mathbf{r}_j(\alpha)\|_2 \Big|_{\alpha=1} = e_j, \tag{24}$$

i.e., e_j is the first-order effect of the neuron on the strength of its module's realized update.

The residual stream is the sole functional medium. Modules communicate only through a shared residual stream, so a neuron's entire downstream effect is mediated by its write $\mathbf{v}_j$, the natural unit of functional contribution—not the scalar pre-activation a_j (Elhage et al., 2021; Geva et al., 2021, 2022). This licenses the projection: a neuron's contribution to its module's computation is exactly how much of its write survives into the net update, and $\hat{\mathbf{d}}$ is the axis along which that survival is measured. Raw activation $|a_j|$ answers a weaker question—how strongly the channel fired—silent on whether the fired vector supports, opposes, or is orthogonal to the computation the module performs.

The cancelling orthogonal component is superposition interference. The completeness decomposition (Eq. 23) has a direct reading under feature superposition (Elhage et al., 2022). When many features are packed into fewer dimensions, individual neuron writes are rarely aligned with the net update; most of their energy lives in the orthogonal component $\mathbf{q}_j$ and is cancelled by opposing writes from other neurons before it reaches the residual stream. Scoring by e_j deliberately discards this cancelled energy and retains only the component that the module's combinatorics let through. This is the geometric reason RDA is robust to the polysemantic firing that inflates activation-based scores: a neuron may fire vigorously on a domain input yet contribute nothing to the module's domain update because its write is spent in directions the rest of the module cancels. The sign of e_j further separates a neuron that reinforces the realized update from one that counteracts it and must be overpowered for the net $\Delta\mathbf{r}$ to emerge.

From per-observation alignment to population functional role. Functional role is not a property of a single observation; it is a population property—a neuron serves a domain function to the extent that it contributes consistently across that domain. The generative model of Eq. 2 is built so that this population property is exactly what the posterior recovers. A neuron whose write reliably reinforces the module's domain-relevant update produces a stream of positive $e_j^{(t)}$ with small spread, yielding a high posterior mean $\mu_{n,j}$ together with low dispersion SS_j; both enter CAM, the former through the location and the latter through the credible-bound

penalty in Eq. 19. The two functional-consistency desiderata of §2.1—magnitude and stability—are therefore not imposed after the fact but are inherited from the alignment geometry: magnitude is the signed share of executed module work (Eq. 23), and stability is the cross-observation reproducibility of that share. Neurons that participate only on a handful of domain inputs, or whose write reinforces the update on some observations and opposes it on others, see positive and negative shares cancel, inflating SS_j and shrinking CAM regardless of their peak activation. Alignment supplies the per-observation evidence; Bayesian aggregation promotes a reproducible positive share into the statistical notion of a role. The match is by construction, since $e_j^{(t)}$ is precisely the quantity whose stable positive population mean defines "this neuron reliably drives the module's domain computation."

Why the module increment, not the full residual. The completeness and sensitivity identities (Eqs. 23–24) hold because the evaluation axis is the module's own increment $\Delta\mathbf{r}$ rather than the full residual $\mathbf{r}_{\text{in}} + \Delta\mathbf{r}$. The full residual is dominated by the accumulated output of all prior layers, so projecting onto it would credit a neuron for agreement with computation performed upstream rather than for the work this module performs at its own depth. Using the increment keeps $\sum_j e_j = \|\Delta\mathbf{r}\|_2$ an identity of the module's local job, so that alignment measures contribution to what the module itself does—leaving final-output relevance to be tested separately by suppression and reference-set filtering (Appendix A). Local alignment alone does not establish causality for the final model output; it deliberately produces a stable, layer-appropriate observation variable for population-level statistical inference.

C RACE Evidence Computation Strategies

RACE supports two primary evidence tracking strategies based on the nature of the evaluation corpus:

- **Generative Evidence (Autoregressive):** For task-solving benchmarks like MBPP+ and MATH-500, we perform full autoregressive generation. Only the output tokens generated by the model are logged as evidence. The prompt/prefill tokens provide necessary conditioning but are not counted in the target-domain evidence, focusing the scoring on the model's active generation behavior.

- **Non-Generative Evidence (Teacher Forcing):** For continuous text corpora (WikiText-2) or fixed structural behaviors (PyComp-1K), we execute a single forward pass over the provided text sequences using teacher forcing. In this setting, all input tokens within the sequence are tracked as valid evidence. This strategy relies on the intrinsic sequence structure without requiring the model to sequentially produce new tokens.

Under the generative evidence strategy, the Qwen3-4B-it auditing runs induce token-position evidence counts of $n = 705{,}426$ for MATH-500 and $n = 98{,}560$ for MBPP+.

D Cross-Domain Consistency of RACE-Selected Neurons

This appendix analyzes whether the top-k neurons identified by RACE are shared across distinct task domains. The goal is to distinguish broadly shared selected neurons from neurons selected primarily for a single target distribution. We evaluate three domains: mathematical reasoning, code generation, and general language modeling, as summarized in Table 7.

Table 7: Domains used for cross-domain consistency analysis.

Domain	Type	Description
`Math-500`	Reasoning	Mathematical problem solving
`MBPP+`	Code	Python programming tasks
`WikiText-2`	Language	General text modeling

Analyzed Modules. For each configuration, we analyze both ATTN and MLP. For each layer and module, RACE ranks neurons by `CAM`. We then select the top-k neurons under thresholds $k \in \{1\%, 5\%, 10\%\}$.

Metrics. Let $S_d^{(l,m,k)}$ denote the top-k neuron set selected for domain d, layer l, and module m. For each pair of domains d_1, d_2, we compute the layer-wise Jaccard similarity:

$$J(d_1, d_2) = \frac{|S_{d_1}^{(l,m,k)} \cap S_{d_2}^{(l,m,k)}|}{|S_{d_1}^{(l,m,k)} \cup S_{d_2}^{(l,m,k)}|}. \tag{25}$$

We additionally compute an all-domain overlap score that requires a neuron to be shared by all three domains:

$$J_{\text{all}} = \frac{|S_{\text{math500}} \cap S_{\text{mbpp\_plus}} \cap S_{\text{wikitext2}}|}{|S_{\text{math500}} \cup S_{\text{mbpp\_plus}} \cup S_{\text{wikitext2}}|}. \tag{26}$$

All reported values are averaged over layers, with standard deviations computed across layers.

Main Finding. Attention neurons are substantially more domain-general than MLP neurons. Under CAM with a Top-1% threshold, attention neurons obtain an all-domain Jaccard of 0.264 (std = 0.099), meaning that roughly 26% of the selected attention neurons are shared across all three domains. In contrast, MLP neurons obtain an all-domain Jaccard of only 0.085 (std = 0.052), meaning that only about 8.5% of the selected MLP neurons are shared. This gives a 3.1× gap between attention and MLP modules.

Table 8: All-domain Jaccard similarity of top-k neurons under $\mathbf{R}_{\text{MATH-500}}$, $\mathbf{R}_{\text{MBPP+}}$, and $\mathbf{R}_{\text{WikiText-2}}$. Attention neurons are consistently more shared across domains than MLP neurons.

Configuration	Attention	MLP	Ratio
CAM, Top-1%	0.264	0.085	3.1×
CAM, Top-5%	0.249	0.110	2.3×
CAM, Top-10%	0.258	0.137	1.9×

Table 8 shows that the gap is robust across both scoring metrics and all top-k thresholds. As k increases, the MLP all-domain overlap rises moderately, but attention remains consistently higher. This suggests that attention output channels contain a larger population of reusable cross-domain routing or integration neurons, whereas MLP down-projection neurons are less shared across tasks and show narrower response patterns under RACE scoring.

E Math Domain Intervention with Reference-Set Filtering on Llama-3.1-8B-it

This appendix reports additional k=50 ATTN and MLP suppression results for $\mathbf{R}_{\text{MATH-500\textbackslash WikiText-2}}$ on Llama-3.1-8B-it in Table 9. Neurons are scored on MATH-500, and the intervention set removes neurons that overlap with a reference RACE result computed on WikiText-2 before suppression.

Table 9: Math domain with $\mathbf{R}_{\text{MATH-500\textbackslash WikiText-2}}$: accuracy on Llama-3.1-8B-it after suppressing top-k=50 neurons per layer.

Module	Method	MATH-500♠ ↓	AMC♡ ↓	GPQA	MMLU-Redux	ARC	ISI↑
ATTN	Neg. CAM	47.00	15.67	28.79	71.35	86.10	2.02
	RACE	27.00	6.72	22.73	67.35	86.16	1.65
MLP	Neg. CAM	51.00	23.14	31.31	72.61	86.22	0.98
	RACE	8.80	2.24	24.75	69.89	85.71	2.36
Llama-3.1-8B-it		50.40	24.63	29.80	72.91	86.16	—

F Model, Generation, and Evaluation Details

This appendix summarizes the model configurations, decoding parameters, and benchmark evaluation protocol used in the experiments. Benchmark evaluation is conducted with `EvalScope` 1.5.0 using `vLLM` 0.15.1 as the inference backend. Unless otherwise specified, benchmark scores follow the official metric implementation exposed by `EvalScope`. We report the resulting benchmark score as Domain Accuracy (DA) in the main tables. Model generation settings, architectural dimensions, benchmark roles, EvalScope benchmark metadata, and corpus statistics are reported in Tables 10, 11, 12, 13, 14, and 15.

Table 10: Model and decoding settings used for benchmark evaluation. Parameters not listed are left at the evaluator or backend default; "—" indicates that the parameter is unset.

Model	Scale	Temperature	Top-p	Top-k
Qwen3-4B-it-2507	4B	0.7	0.8	20
Llama-3.1-8B-it	8B	0.6	0.9	—
OLMo-3.1-32B-it	32B	0.6	0.95	—

Table 11: Architectural dimensions of all evaluated models.

Model	Layers	ATTN neurons / layer	MLP neurons / layer
Qwen3-4B-it-2507	36	2560	9728
Llama-3.1-8B-it	32	4096	14336
OLMo-3.1-32B-it	64	5120	27648

G Computational Architecture

This appendix reports the compute environments used for the main experiments. Experiments on Qwen3-4B-it were executed on the 8-GPU NVIDIA RTX 4090 machine. Experiments on Llama-3.1-8B-it and OLMo-3.1-32B-it were executed on the 8-GPU NVIDIA A100 80GB machine. The full hardware and system configuration is given in Table 16.

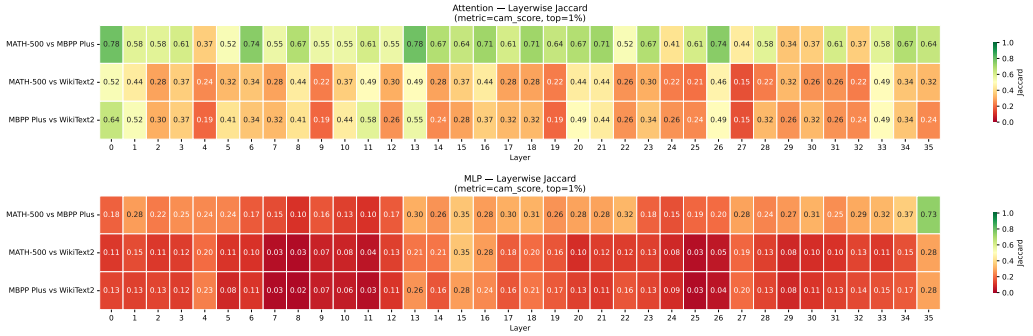

Figure 4: **Layer-wise cross-domain overlap of top-k neurons (CAM score, top-1%).** Heatmaps show Jaccard similarity between domain pairs across transformer layers for the selected module, attribution metric, and top-k threshold. Each cell reports the overlap between the two domains' top-ranked neuron sets in that layer, with higher values indicating greater cross-domain consistency in the identified important neurons. **Top:** attention neurons exhibit substantially higher cross-domain overlap, consistent with the all-domain Jaccard of 0.264. **Bottom:** MLP neurons show markedly lower inter-domain agreement, reflecting more domain-specific response patterns (all-domain Jaccard 0.085).

Table 12: Benchmark evaluation setup.

Benchmark	Role	Reported score	Evaluator
MBPP+	Code scoring / in-distribution validation	EvalScope benchmark score	EvalScope + vLLM
HumanEval+	Code out-of-distribution validation	EvalScope benchmark score	EvalScope + vLLM
MATH-500	Math scoring / in-distribution validation	EvalScope benchmark score	EvalScope + vLLM
AMC	Math out-of-distribution validation	EvalScope benchmark score	EvalScope + vLLM
ARC	Non-target retention / reasoning control	EvalScope benchmark score	EvalScope + vLLM
MMLU-Redux	General-knowledge retention	EvalScope benchmark score	EvalScope + vLLM
GPQA	General-knowledge retention	EvalScope benchmark score	EvalScope + vLLM

Table 13: EvalScope benchmark metadata for all benchmark datasets used in the paper, extracted from the corresponding EvalScope benchmark cards (EvalScope, 2026e,c,d, 2024, 2026a,f,b).

Benchmark	EvalScope key	Task type	Split	Metric / aggregation
MBPP+	`mbpp_plus`	Python code generation	`test`	acc; `mean_and_pass_at_k`
HumanEval+	`humaneval_plus`	Python code generation	`test`	acc; `mean_and_pass_at_k`
MATH-500	`math_500`	Mathematical problem solving	`test`	acc
AMC	`amc`	Competition math multiple choice	N/A	acc
ARC	`arc`	Multiple-choice science QA	`test`	acc
MMLU-Redux	`mmlu_redux`	Multiple-choice knowledge QA	`test`	acc
GPQA	`gpqa_diamond`	Expert-level science MCQ	`train`	acc

H Profiler-Based Scoring Cost Measurement

We measure the computational-efficiency numbers in Table 6 with `torch.profiler(with_flops=True)` on Qwen3-4B-it. The forward reference is the profiled cost of the same 64-input/16-output window, including the prefill-to-generation increment and the selected `lm_head` projections. This reference costs 130.100 GFLOPs.

Protocol. All methods use the same WikiText-2 token window, target positions, target token ids, and target modules. We use one lookahead token to define the next-token target for the 16th generated position, but do not increase the model input length. For GxAct and AttnLRP, we batch all target layers together by default, avoiding artificial repetition of the same forward/backward computation once per layer. The profiler records only the extra scoring computation; CAM, NIG updates, HDF5 writes, and CPU aggregation are excluded. We use eager attention so that PyTorch's FLOP profiler can observe the attention matrix multiplications instead of hiding them inside fused kernels.

Table 14: Dataset statistics for all EvalScope benchmarks used in the paper. Prompt lengths are measured in characters as reported by the EvalScope benchmark cards (EvalScope, 2026e,c,d, 2024, 2026a,f,b).

Benchmark	Samples	Mean prompt length	Min / max prompt length	Subset / coverage notes
MBPP+	378	375.53	222 / 2801	Python programming tasks with expanded tests
HumanEval+	164	609.57	274 / 1519	Original HumanEval problems with enhanced tests
MATH-500	500	266.89	91 / 1804	Levels 1–5: 43 / 90 / 105 / 128 / 134 examples
AMC	134	324.58	98 / 1218	AMC22 / AMC23 / AMC24: 43 / 46 / 45 examples
ARC	3548	424.43	253 / 1157	ARC-Easy / ARC-Challenge: 2376 / 1172 examples
MMLU-Redux	5700	600.81	255 / 5082	57 subjects; 100 examples per subject
GPQA	198	841.15	340 / 5845	GPQA-Diamond expert science subset

Table 15: Non-EvalScope corpora and locally constructed datasets appearing in the paper. WikiText-2 provides D_{ref} for $\mathbf{R}_{D_{\text{tar}}\setminus\text{WikiText-2}}$, The Stack (Kocetkov et al., 2022; BigCode, 2026) is the source corpus for constructing PyComp-1K, and PyComp-1K provides D_{tar} in the fine-grained $\mathbf{R}_{\text{PyComp-1K}\setminus\text{WikiText-2}}$ intervention.

Dataset / corpus	Role in paper	Size / statistics	Construction or source notes
WikiText-2 (Merity et al., 2016)	General-language D_{ref} reference	2,088,628 train tokens; 217,646 validation tokens; 245,569 test tokens	Wikipedia; fixed-length articles; vocabulary of 33,278 words
The Stack (Kocetkov et al., 2022; BigCode, 2026)	Source corpus for PyComp-1K construction	6.4 TB permissively licensed source code across 358 programming languages	Only Python files are extracted for PyComp-1K extraction before AST filtering
PyComp-1K	Narrow code-behavior scoring corpus	1000 AST-verified Python statements	Extracted from bigcode/the-stack (Kocetkov et al., 2022; BigCode, 2026); targets 1-to-6 Comp, 3rd Comp, and Generator-Expression

I PyComp-1K Dataset

To evaluate model behaviors on specific Python structural patterns, we utilize the PyComp-1K dataset. This dataset contains 1,000 Python statements extracted from the Python subset of `bigcode/the-stack` (Kocetkov et al., 2022; BigCode, 2026). Each example consists of the nearest enclosing Python statement around one or more

Table 16: Computational architecture used in the experiments. CPU, memory, operating system, and GPU information are reported from the execution environments.

Environment	System	CPU	Memory	GPU / experiments
RTX 4090 server	Ubuntu, Linux 6.8.0-107-generic	Intel Xeon Gold 6138	125GiB RAM	8× NVIDIA GeForce RTX 4090; Qwen3-4B-it
A100 server	Linux 4.18.0-147	AMD EPYC 7713	368GiB RAM	8× NVIDIA A100 80GB; Llama-3.1-8B-it and OLMo-3.1-32B-it

Table 17: Profiler-measured FLOPs for Qwen3-4B-it under the 64-input/16-output protocol.

Measurement	Profiler FLOPs	Relative to forward
Forward reference	130.100 GFLOPs	100.000%
RDA extra	**81.545** GFLOPs	**62.679%**
GxAct extra	18.772 TFLOPs	14428.812%
AttnLRP extra	18.815 TFLOPs	14461.987%

comprehension expressions.

I.1 Construction Process

The dataset construction process streamed Python files from The Stack (Kocetkov et al., 2022; BigCode, 2026) and parsed each source file using Python's `ast` module. The extraction logic specifically targeted four types of AST comprehension nodes: `ListComp`, `SetComp`, `DictComp`, and `GeneratorExp`. For each unique comprehension span identified, the nearest enclosing statement was extracted and saved. Rows that could not be parsed as valid Python source were skipped.

I.2 Dataset Statistics

The 1,000 extracted statements exhibit the following distribution of comprehension types (note that a single statement may contain multiple comprehensions):

Table 18: Distribution of comprehension types in the Python Comprehension Statements 1K dataset.

Comprehension Type	Count
List Comprehension (`ListComp`)	754
Generator Expression (`GeneratorExp`)	181
Dictionary Comprehension (`DictComp`)	76
Set Comprehension (`SetComp`)	9

Each row in the dataset provides the extracted statement, the corresponding line numbers in the original source, the set of AST comprehension types present, and the source metadata from The Stack (Kocetkov et al., 2022; BigCode, 2026) (e.g., commit hash, repository name, and file path).

J Qualitative Outputs under the PyComp-1K Perturbation

This section reports five representative Python-generation samples under the PyComp-1K perturbation setting. For each pair, the first block shows a correct unmodified model output, while the second block shows an incorrect output after perturbing RACE-selected PyComp-1K neurons.

Example 1: square individual list elements.

Original model (correct).

````
```python
def square_nums(nums):
    """
    Takes a list of numbers and returns a new list containing
       the square of each element.

    Args:
        nums (list): A list of integers or floats

    Returns:
        list: A list where each element is the square of the
      corresponding element in the input list

    Examples:
        >>> square_nums([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
        [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
        >>> square_nums([10, 20, 30])
        [100, 400, 900]
        >>> square_nums([12, 15])
        [144, 225]
    """
    return [num ** 2 for num in nums]
```
````

Perturbed model (incorrect).

````
```python
def square_nums(nums):
    """
    Takes a list of numbers and returns a new list with the
      square of each element.

    Args:
        nums (list): A list of integers or floats

    Returns:
        list: A list containing the square of each element
      from the input list
    """
    return [num**2]
```
````

Example 2: extract strings of a specified length.

Original model (correct).

````
```python
def extract_string(string_list, size):
    """
    Extract strings from a list that have a specified length.

    Args:
        string_list: List of strings to filter
        size: The length to check for (strings of this exact
      length will be returned)

    Returns:
        List of strings that have the specified length
    """
    return [s for s in string_list if len(s) == size]
```
````

Perturbed model (incorrect).

```python
def extract_string(string_list, size):
    """
    Extract strings from a list that have the specified
      length.

    Args:
        string_list: List of strings to filter
        size: The length to filter by (only strings of this
      exact length are returned)

    Returns:
        List of strings that have the exact specified length
    """
    return [string for len(string) == size]
```

Example 3: sort a dictionary by value.

Original model (correct).

```python
def sort_counter(d):
    """
    Sort a dictionary by value in descending order and return
       a list of tuples.

    Args:
        d (dict): A dictionary with keys and values to be
      sorted

    Returns:
        list: A list of tuples (key, value) sorted by value
      in descending order
    """
    return sorted(d.items(), key=lambda item: item[1],
      reverse=True)
```

Perturbed model (incorrect).

```python
def sort_counter(dict_input):
    """
    Sort a dictionary of key-value pairs by value in
      descending order.

    Args:
        dict_input (dict): A dictionary where each key-value
      pair is a tuple or key-value pair

    Returns:
        list: A list of tuples sorted by value in descending
      order
    """
    # Convert the dictionary to a list of tuples and sort by
      value (second element) in descending order
    return sorted(dict_input, key=lambda x: x[1], reverse=
      True)
```

Example 4: sorted union of two tuples.

Original model (correct).

```python
def union_elements(tuple1, tuple2):
    """
    Find the union of elements from two tuples and return
      them in sorted order.

    Args:
        tuple1: First tuple of elements
        tuple2: Second tuple of elements

    Returns:
        A tuple containing the union of elements from both
      tuples, sorted in ascending order
    """
    # Convert tuples to sets to find union (automatically
      removes duplicates)
    union_set = set(tuple1) | set(tuple2)

    # Convert back to sorted tuple
    sorted_union = tuple(sorted(union_set))

    return sorted_union
```

Perturbed model (incorrect).

```python
def union_elements(tuple1, tuple2):
    """
    Find the union of elements from two tuples and return
      them in sorted order.

    Args:
        tuple1: First tuple of elements
        tuple2: Second tuple of elements

    Returns:
        A tuple containing the union of elements from both
      input tuples, sorted in ascending order
    """
    # Convert tuples to sets to find union (remove duplicates
      )
    set1 = set(tuple1)
    set2 = set(tuple2

    # Find union of both sets
    union_set = set1.union(set2)

    # Convert back to sorted tuple
    sorted_union = tuple(sorted(union_set))

    return sorted_union
```

Example 5: retrieve the k-th array element.

Original model (correct).

```python
def kth_element(arr, k):
    """
    Find the kth element in the given array using 1-based
      indexing.

    Args:
        arr: List of integers
        k: 1-based index of the element to find

    Returns:
        The kth element in the array

    Raises:
        IndexError: If k is out of bounds (k > length of
      array)
    """
    if k < 1:
        raise IndexError("k must be a positive integer")

    if k > len(arr):
        raise IndexError(f"k ({k}) is greater than array
      length ({len(arr)})")

    return arr[k - 1]  # Convert 1-based index to 0-based
      index
```

Perturbed model (incorrect).

```python
def kth_element(arr, k):
    """
    Find the kth element in the given array using 1-based
      indexing.

    Args:
        arr: List of numbers
        k: 1-based index of the element to find

    Returns:
        The kth element in the array

    Raises:
        IndexError: If k is out of bounds (k < 1 or k > len(
      arr))
    """
    # Validate input
    if k < 1:
        raise IndexError("k must be a positive integer (1-
      based indexing)")

    if k > len(arr):
        raise IndexError(f"k ({k}) is greater than array
      length ({len(arr)})")

    # Return the kth element (k-1 index in 0-based indexing)
    return arr[k - 1]
```

K Robustness to Prior Settings and Confidence Levels

When the sample size is sufficiently large ($n \gg \lambda_0$), the posterior distribution is dominated by empirical statistics, rendering the choice of prior parameters largely inconsequential. Empirically, across a wide range of prior configurations ($\lambda_0, \beta_0 \in [10^{-3}, 10^3]$, $\alpha_0 \in [0.5, 50]$), the top-1% selected neurons remain highly stable (Jaccard similarity ≥ 0.98). Regarding the CAM confidence level γ, it primarily acts as an absolute verification threshold rather than altering the relative ranking of the top candidates. Consequently, the sets of top-k neurons selected under fixed budgets remain largely identical across different γ values. In practice, adopting a more stringent confidence level (e.g., tightening γ from 0.05 to 0.001) effectively filters out marginal candidates, shrinking the pool of valid positive-CAM neurons from 81.75% to 62.24% (at $N = 10$), without displacing the most prominent neurons.

L Distributional Disruption Metrics

The main paper evaluates token-distribution-level disruption using relative perplexity degradation Δ_{PPL} and mean forward Kullback–Leibler divergence $\bar{D}_{\mathrm{KL}}$. Δ_{PPL} is reported in percentage form:

$$\Delta_{\mathrm{PPL}} = \frac{\mathrm{PPL}_{\mathrm{sup}} - \mathrm{PPL}_{\mathrm{base}}}{\mathrm{PPL}_{\mathrm{base}}}. \tag{27}$$

This aggregates the increase in per-token surprisal and quantifies the overall deterioration in predictive quality. $\bar{D}_{\mathrm{KL}}$ is computed as

$$\bar{D}_{\mathrm{KL}} = \frac{1}{T}\sum_{t=1}^{T} D_{\mathrm{KL}}\big(P_{\mathrm{base}}(\cdot \mid \mathbf{x}_{<t}) \,\|\, P_{\mathrm{sup}}(\cdot \mid \mathbf{x}_{<t})\big). \tag{28}$$

This measures the mean distributional shift per token position and can detect behavioral changes even when aggregate likelihood is largely preserved.

M Without RSF (w/o-RSF) Distributional Disruption on Qwen3-4B-it

This appendix reports distributional disruption results w/o-RSF for Qwen3-4B-it. All checkpoints use top-k=5 RACE-selected neurons per layer with the suppression operation. They are evaluated by direct model comparison against the unmodified model on the first 100 samples per dataset in order. The resulting Δ_{PPL} and $\bar{D}_{\mathrm{KL}}$ measurements are summarized in Table 19.

Table 19: w/o-RSF domain perplexity degradation (Δ_{PPL}, Eq. 27) and mean per-token forward KL divergence ($\bar{D}_{\mathrm{KL}}$, Eq. 28) on Qwen3-4B-it after suppressing top-k=5 RACE-selected neurons per layer. Columns report interventions under $\mathbf{R}_{\mathrm{MBPP+}}$ and $\mathbf{R}_{\mathrm{MATH\text{-}500}}$; all evaluations use direct model comparison against the unmodified model.

		(a) $\mathbf{R}_{\mathrm{MBPP+}}$			(b) $\mathbf{R}_{\mathrm{MATH\text{-}500}}$		
Metric	**Module**	**MBPP+♠**	**MATH-500**	**WikiText-2**	**MBPP+**	**MATH-500♠**	**WikiText-2**
Δ_{PPL}	ATTN	+30.16%	+25.97%	+5.82%	+28.43%	+27.50%	+4.84%
	MLP	+263.82%	+170.24%	+740.63%	+16.86%	+34.57%	+58.56%
$\bar{D}_{\mathrm{KL}}$	ATTN	0.2913	0.2793	0.2036	0.2793	0.2823	0.2027
	MLP	1.4031	1.1246	2.4549	0.1721	0.3280	0.3476

N Code Domain Intervention with Reference-Set Filtering on Qwen3-4B-it

This appendix reports the code-domain Qwen3-4B-it results summarized by the radar plot in Figure 2. Neurons are selected with $\mathbf{R}_{\mathrm{MBPP+}\setminus\mathrm{WikiText\text{-}2}}$, and Table 20 provides the full benchmark and ISI values.

O Math Domain Intervention on Qwen3-4B-it

This appendix reports additional math-domain suppression results on Qwen3-4B-it in Tables 21 and 22.

Table 20: Code domain with $\mathbf{R}_{\text{MBPP+}\setminus\text{WikiText-2}}$: accuracy (%) and ISI on Qwen3-4B-it after suppressing top-1% ATTN or MLP target-selected neurons per layer.

Module	Method	MBPP+♠ ↓	HumanEval+♡ ↓	MATH-500	AMC	MMLU-Redux	GPQA	ISI↑
ATTN	GxAct	8.73	7.93	17.20	2.24	47.67	21.72	0.26
	AttnLRP	0.00	0.00	2.60	0.75	6.07	5.05	0.05
	Act. Mean	81.75	86.59	90.03	75.37	80.67	46.97	0.00
	Emp. Mean	66.40	78.66	88.34	63.43	78.54	40.40	0.00
	Emp. SNR	68.52	84.15	93.44	76.86	79.49	40.40	0.09
	Neg. CAM	81.75	82.93	92.61	83.58	81.53	45.45	0.00
	RACE	65.87	80.49	90.30	68.66	79.07	44.41	0.29
MLP	GxAct	23.81	19.51	79.23	46.27	78.49	40.40	1.29
	AttnLRP	0.00	3.05	61.37	35.07	76.39	31.82	1.07
	Act. Mean	11.38	12.21	88.73	72.09	82.61	40.98	2.22
	Emp. Mean	5.82	2.44	86.44	63.43	82.30	55.05	2.26
	Emp. SNR	83.33	84.76	93.83	69.40	78.98	38.89	0.00
	Neg. CAM	68.25	84.76	93.23	81.34	81.72	48.99	1.04
	RACE	3.17	1.22	85.41	75.11	82.35	52.53	2.65
Qwen3-4B-it		82.28	83.54	94.40	87.31	81.37	45.45	—

Table 21: Math domain with $\mathbf{R}_{\text{MATH-500}}$: accuracy (%) on Qwen3-4B-it after suppressing top-k=5 neurons per layer.

Module	Method	MATH-500♠	AMC♡	MBPP+	HumanEval+	MMLU-Redux	GPQA	ISI↑
ATTN	GxAct	80.00	47.01	59.79	49.39	78.39	37.37	0.27
	AttnLRP	89.60	73.88	48.15	47.56	76.98	43.94	0.00
	Act. Mean	92.80	86.57	70.11	63.41	79.54	45.96	0.00
	Emp. Mean	93.60	86.56	72.22	56.71	79.14	44.95	0.00
	Emp. SNR	94.20	84.33	76.72	49.39	80.07	41.41	0.00
	Neg. CAM	94.20	85.82	73.28	83.54	79.56	43.94	0.00
	RACE	94.20	84.33	73.28	51.83	80.18	42.42	0.00
MLP	GxAct	89.60	70.15	80.42	71.34	80.07	41.41	0.46
	AttnLRP	90.40	80.60	83.60	83.54	81.42	43.43	1.04
	Act. Mean	92.00	80.60	78.57	71.95	81.19	53.03	0.00
	Emp. Mean	57.40	35.08	78.31	85.37	80.30	42.42	2.47
	Emp. SNR	93.80	80.60	71.96	83.54	80.30	42.93	0.00
	Neg. CAM	56.00	35.07	79.63	85.37	79.98	40.40	2.31
	RACE	63.60	29.11	79.89	85.98	80.14	44.44	2.93
Qwen3-4B-it		94.40	87.31	82.28	83.54	81.37	45.45	—

P Bayesian Variance Regularization in Low-Data Regimes

This appendix expands the variance-regularization argument summarized in §3.5. Let $N = |D_c|$ denote the number of input samples. Let $n = |T_c|$ denote the induced token-position evidence count for a fixed scoring set. Under the default RACE prior $\mu_0 = 0$, $\lambda_0 = 1$, $\alpha_0 = 1$, and $\beta_0 = 1$, the NIG update in Eq. (16) becomes

$$\beta_{n,j} = 1 + \frac{\mathrm{SS}_j}{2} + \frac{n\bar{e}_j^2}{2(1+n)} \geq 1. \tag{29}$$

This lower bound holds even when the empirical dispersion collapses to $\mathrm{SS}_j = 0$. Since $\alpha_n = 1 + n/2$ and $\lambda_n = 1 + n$, the posterior scale for mean evidence satisfies

$$\sigma_{\mu,j} = \sqrt{\frac{\beta_{n,j}}{\alpha_n \lambda_n}} \geq \sqrt{\frac{1}{(1+n/2)(1+n)}} > 0. \tag{30}$$

Thus CAM retains a finite conservative margin against weak or sparsity-induced evidence at finite n. This prevents near-zero empirical variance from eliminating the uncertainty penalty. As n grows, this lower bound decays, matching the main-text observation that the prior becomes negligible in large-evidence regimes.

Table 22: Math domain with $\mathbf{R}_{\text{MATH-500}\setminus\text{WikiText-2}}$: accuracy (%) on Qwen3-4B-it after suppressing top-1% ATTN or MLP target-selected neurons per layer.

Module	Method	MATH-500♠	AMC♡	MBPP+	HumanEval+	MMLU-Redux	GPQA	ISI↑
ATTN	GxAct	4.00	2.24	1.06	0.00	16.98	7.07	0.05
	AttnLRP	1.00	1.49	0.00	0.00	4.33	2.53	0.00
	Act. Mean	73.20	41.60	78.04	78.66	75.23	36.87	1.28
	Emp. Mean	92.20	73.88	69.31	83.54	78.33	40.40	0.02
	Emp. SNR	91.80	78.36	82.54	84.76	79.61	43.94	1.01
	Neg. CAM	91.20	82.84	79.89	79.88	81.26	42.93	0.00
	RACE	90.60	72.39	71.43	84.76	78.54	40.91	0.32
MLP	GxAct	30.80	9.70	69.02	51.39	77.19	41.41	1.46
	AttnLRP	46.00	31.34	53.70	33.54	79.16	43.43	0.78
	Act. Mean	58.40	47.76	65.61	48.17	79.79	40.91	0.75
	Emp. Mean	27.20	12.69	61.64	55.49	77.74	39.39	1.36
	Emp. SNR	95.00	86.57	74.07	82.93	80.70	39.90	0.00
	Neg. CAM	95.60	85.08	80.69	85.98	81.44	44.44	0.00
	RACE	29.00	11.94	70.20	67.49	77.79	40.40	1.76
Qwen3-4B-it		94.40	87.31	82.28	83.54	81.37	45.45	—