Route-Induced Density and Stability (RIDE): Controlled Intervention and Mechanism Analysis of Routing-Style Meta Prompts on LLM Internal States

Anonymous ACL submission

Abstract

Routing is widely used to scale large language models, from Mixture-of-Experts gating to multi-model/tool selection. A common belief is that routing to a task "expert" activates sparser internal computation and thus yields more certain and stable outputs (the Sparsity–Certainty Hypothesis). We test this belief by injecting routing-style meta prompts as a textual proxy for routing signals in front of frozen instruction-tuned LLMs. We quantify (C1) internal density via activation sparsity, (C2) domain-keyword attention, and (C3) output stability via predictive entropy and semantic variation. On a RouterEval subset with three instruction-tuned models (Qwen3-8B, Llama-3.1-8B-Instruct, and Mistral-7B-Instruct-v0.2), meta prompts consistently densify early/middle-layer representations rather than increasing sparsity; natural-language expert instructions are often stronger than structured tags. Attention responses are heterogeneous: Qwen/Llama reduce keyword attention, while Mistral reinforces it. Finally, the densification–stability link is weak and appears only in Qwen, with near-zero correlations in Llama and Mistral. We present RIDE as a diagnostic probe for calibrating routing design and uncertainty estimation.

1 Introduction

As large language models (LLMs) are increasingly deployed in search assistants, coding assistants, and multi-tool systems, routing has become a foundational mechanism. In Mixture-of-Experts (MoE) architectures, a lightweight gating network activates only a small subset of expert subnetworks conditioned on the input, achieving parameter-level sparsification and computational savings. In multi-model routing systems, a policy model selects among multiple LLMs, toolchains, or APIs to balance performance, cost, and safety. As routing structures grow more complex, a new challenge emerges: we must understand not only whether routing decisions are "correct," but also how routing signals shape the backbone model' s internal representations and reasoning patterns. A compelling intuition in current practice and informal discussions is that routing effectively selects a narrower, more specialized "expert pathway", making internal computation sparser and more focused, ultimately producing more certain and stable outputs. We formalize this intuition as the Sparsity–Certainty Hypothesis: when a model receives a clear routing signal (e.g., "this is a math task"), its internal representations propagate along more selectively gated sparse pathways, suppressing task-irrelevant activations and thereby reducing output entropy and increasing cross-sample consistency. This hypothesis has been implicitly adopted in various settings—for instance, using internal signals such as activation sparsity or attention concentration as proxies for uncertainty estimation, early exit, or routing decisions. However, modern instruction-tuned LLMs have highly complex internal structures that strongly depend on training details, and whether this hypothesis holds has not been systematically tested. Directly "opening up" real MoE gating mechanisms or complex policy routers is often impractical. Engineering systems are typically tightly encapsulated, making expert activation dynamics difficult to access; moreover, modifying routers to insert diagnostic signals may compromise stability and safety in production. We therefore take a pragmatic alternative: instead of intervening on the router itself, we inject routing-style meta prompts at the input level as a textual proxy for routing decisions. Concretely, we prepend

different types of prefixes to frozen instruction-tuned LLMs, including structured route tags (e.g., [RouteTag=math]), natural-language expert instructions (e.g., "You are a Math Expert."), as well as control conditions such as incorrect tags and placebo tags. For each instance, we keep the backbone parameters and random seeds identical across prefix conditions and vary only the prefix, thereby constructing paired, controlled intervention experiments. By comparing internal activation density, domain-keyword attention, and output stability across conditions, we can analyze how "routing-style signals" modulate internal states. Building on this idea, we propose RIDE (Route-Induced Density and Stability), centered on three research questions:

- RQ1 (Density effects): Under a frozen backbone, how do different routing-style meta prompts (route tags vs. expert prompts) change the density/sparsity of internal activations? Which layer segments are most affected?
- RQ2 (Attention strategies): Do routing-style meta prompts change how the model leverages domain information? After receiving explicit labels, do models "offload" reliance on domain keywords, or instead increase their focus on keywords?
- RQ3 (Densification–stability relation): At the instance level, is there a stable association between changes in internal densification (C1) and changes in output stability (C3)? Is this association consistent across models, and can it serve as a general proxy for routing decisions?

Importantly, the "routing signals" analyzed in this work are not the internal states of real MoE gating modules or multi-model routers; they are routing-style meta instructions injected as textual prefixes. Our conclusions thus constitute causal-style interventional evidence from controlled experiments, rather than full-fledged structural causal identification.

1.1 Contributions

Within this framework, our main contributions are:

- **RIDE metrics and a unified controlled-intervention pipeline.** We design a suite of routing-style meta prompts and control conditions (control / correct tag / incorrect tag / placebo tag / expert instruction), construct paired interventions on three open-source instruction-tuned LLMs, and define three metric families—C1 (density), C2 (domain-keyword attention), and C3 (output stability)—as a systematic toolkit for analyzing routing-signal effects.
- **Model heterogeneity in densification and attention responses to meta prompts.** Experiments show that all models exhibit significant densification responses to task-oriented meta prompts in the Early/Middle layers, while natural-language expert instructions induce stronger densification than structured route tags in most settings. At the attention level, different models can even exhibit opposite strategies: Llama/Qwen align more with cognitive offloading, whereas Mistral displays attention reinforcement.
- **A systematic test of the "densification ⇒ stability" chain with informative negative results.** We observe a small-to-moderate positive correlation consistent with densification–stability coupling in Qwen3-8B, but the link is weak or statistically insignificant at the instance level for Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.2. These results suggest limited empirical support for the assumption that internal density serves as a cross-model, general-purpose uncertainty measure or routing signal. Consequently, RIDE is better positioned as a diagnostic probe for revealing heterogeneous model responses to routing-style meta prompts.

2 Background and Related Work

We briefly review three lines of work—routing and routing benchmarks, sparsity–certainty assumptions, and interpretability/meta-prompt analysis—and position RIDE among them.

Routing mechanisms and benchmarks. In Mixture-of-Experts (MoE) models, a gating network activates a small subset of experts for conditional computation, with extensive work on scaling and training stability (e.g., load balancing and regularization). (Shazeer et al., 2017; Fedus et al., 2022; Du et al., 2022; Lepikhin et al., 2021) However, these advances are largely evaluated by downstream performance and FLOPs, and rarely analyze how gating signals shape intermediate representations. (Zoph et al., 2022; Rajbhandari et al., 2022) In multi-model/tool routing, a policy selects among models or tools under capability–cost trade-offs. (Chen et al., 2024; Schick et al., 2023; Yao et al., 2023) Recent benchmarks and learned routing strategies enable systematic evaluation (e.g., across tasks, difficulty, and cost), but mainly assess input–output selection quality rather than internal-state effects. (Hu et al., 2024; Huang et al., 2025; Song et al., 2025) In contrast, we do not build or optimize routers; we treat routing-style meta prompts as controllable textual proxies of routing signals and analyze their internal effects. (Hendel et al., 2023; Liu et al., 2024; Stolfo et al., 2025)

Sparsity and certainty. Across compression, MoE, and adaptive inference, activation sparsity/attention concentration and entropy are often used as proxies for specialization or uncertainty, and as signals for early exiting or model switching. (Frankle and Carbin, 2019; Sanh et al., 2020; Fedus et al., 2022; Du et al., 2022; Zhou et al., 2020; Xin et al., 2020; Chen et al., 2024; Vazhentsev et al., 2022) These practices implicitly suggest that "sparser/more concentrated internal states ⇒ more stable outputs." (Zhai et al., 2023) We formalize this as the Sparsity–Certainty Hypothesis and test it via controlled meta-prompt interventions on frozen instruction-tuned LLMs, separating mechanism probing from performance optimization. (Chung et al., 2024; Stolfo et al., 2025)

Interpretability and meta-prompt analysis. Mechanistic interpretability probes internal computation via activations, attention, and circuit-level discovery/attribution. (Meng et al., 2022; Conmy et al., 2023; Ameisen et al., 2025; Zhang et al., 2025; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019) Recent work also shows that prompts/instructions can steer internal representations and induce measurable distraction/attraction phenomena that can be localized to specific heads or circuit components. (Hendel et al., 2023; Liu et al., 2024; Niu et al., 2025) We follow this direction but focus on routing-style meta prompts and, unlike single-model qualitative studies, provide a unified multi-model intervention pipeline to compare heterogeneity along density–attention–stability pathways. (Huang et al., 2025; Hu et al., 2024; Rimsky et al., 2024; Yu et al., 2025; Meng et al., 2022)

3 RIDE: Routing-Style Meta Prompts and Experimental Design

3.1 Routing-Style Meta Prompts

We inject routing-style meta prompts by prepending short prefixes to the user instruction. For each instance, we construct five prefix conditions: 1.**control**: No routing-related prefix is added; the task instruction is fed directly. 2.`tag_correct`: We add a structured tag that matches the instance domain, e.g., [RouteTag=math] or [RouteTag=code]. The tag format is fixed, with the internal field indicating the domain. 3.`tag_wrong`: We add a tag that does not match the instance domain (e.g., [RouteTag=code] before a math problem) to probe intervention effects of "incorrect routing signals." 4.`tag_placebo`: The tag has the same surface format as `tag_correct`, but the internal string carries no recognizable semantics (e.g., random tokens), controlling for prefix length and formatting effects. 5.`instr_expert`: We use natural-language expert instructions such as "You are a Math Expert." or "You are a coding assistant.", representing widely used role-setting prompts.

All conditions share the same core instruction text, frozen backbone parameters, and decoding configuration; only the prefix differs. Thus, each instance forms a strictly paired intervention experiment across the five conditions, and differences can be interpreted as controlled intervention effects of the "routing-style signal."

3.2 Dataset and Domain Partitioning

We construct our experimental samples from the RouterEval dataset and focus on three sub-domains: **Math**: multi-step mathematical reasoning problems; **Format / IFEval**: tasks with strict output-format constraints; **Commonsense**: primarily multiple-choice commonsense reasoning tasks. Following RouterEval's easy/hard difficulty split, we sample instances from the training split within each sub-domain, ensuring that every model sees exactly the same inputs under all prefix conditions. To avoid data leakage, we use only the official training partition and keep the per-domain sample size on the order of hundreds to a thousand. Appendix A reports detailed statistics of sample distributions by domain and difficulty.

3.3 Models and Decoding Configuration

We evaluate three open-source instruction-tuned models: **Llama-3.1-8B-Instruct**, **Mistral-7B-Instruct-v0.2**, **Qwen3-8B**. The three models have similar parameter scales, are instruction-tuned, and perform well across multiple tasks. We use a unified decoding configuration (temperature, top-p, maximum length, etc.). For each instance and each fixed random seed, we generate K candidate outputs to estimate output entropy and semantic variation. All prefix conditions share the same set of random seeds, minimizing the influence of sampling randomness on paired differences.

3.4 Representation Sampling and Layer-Segment Partitioning

We sample hidden states from every layer and uniformly partition layers by depth into three segments: - Early: the first third of layers closest to the input embeddings; - Middle: the middle third; - Late: the final third closest to the output head. Within each segment, we aggregate over the token dimension (e.g., by mean or max pooling) to obtain a representative hidden vector for computing metrics such as Hoyer sparsity and Top-k energy. For attention-based metrics, we primarily focus on the last layer (or several late layers) as an approximation of how much the decoder attends to domain keywords during decision making.

4 Metrics and Analysis Methods

We define three metric families: **C1** (activation sparsity/density), **C2** (domain-keyword attention), and **C3** (output stability). Unless otherwise noted, all metrics are computed per instance and per prefix condition, then compared via paired differences.

4.1 C1: Activation Sparsity / Density

We quantify the sparsity of a hidden vector $\mathbf{h} \in \mathbb{R}^d$ using Hoyer sparsity: $\text{Hoyer}(\mathbf{h}) = \left(\sqrt{d} - \|\mathbf{h}\|_1/\|\mathbf{h}\|_2\right)/(\sqrt{d}-1) \in [0,1]$, where larger values indicate higher sparsity (thus lower values indicate *densification*). We aggregate token-level Hoyer scores over layer segments (Early/Middle/Late). In addition, we compute the Top-k energy ratio (TopK), i.e., the fraction of ℓ_2 energy captured by the top-k dimensions after sorting by $|h_i|$. We define **C1** as a segment-level *combined metric family* based on Hoyer and TopK; due to space, the main paper primarily reports Hoyer changes, while full definitions and additional TopK results are provided in Appendix B.

4.2 C2: Domain-Keyword Attention

To measure reliance on domain signals, we build a small keyword lexicon for each sub-domain (e.g., math, code) and compute the *attention share* paid to matched keyword positions. Concretely, from the last-layer attention matrix, for a query position t we sum the attention mass assigned to keyword token positions; we then average over (relevant) positions and over instances to obtain the domain-keyword attention share. We report two complementary viewpoints: **Prompt-last** (query at the last input token) and **First-gen** (query at the first generated token), corresponding to domain-information usage when "finishing reading" vs. "starting to answer." Formal definitions are in Appendix B.

4.3 C3: Output Stability

We characterize output stability with two complementary proxies.

Predictive entropy (Entropy). During decoding, we compute token-level softmax entropy and average over the generated sequence; for stochastic decoding, we further average this quantity over the K generations to obtain

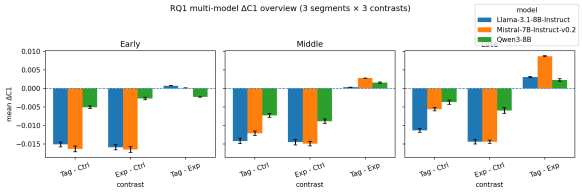


Figure 1: Multi-model overview of ΔHoyer across Early/Middle/Late segments.

a per-instance mean entropy. Lower values indicate a more concentrated predictive distribution.

Semantic variation (SemVar). For K stochastic generations from the same prompt, we compute pairwise embedding similarities and use Var $= 1 -$ mF1 as a variation score; smaller values indicate higher semantic consistency across generations. Implementation details (encoder choice, K, sampling settings) are deferred to Appendix C. In the main text, we analyze Entropy and SemVar trends separately.

4.4 Paired Differences and Correlation Estimation

Our primary analyses use instance-level paired differences. For a metric M and instance i, we define $\Delta M_i = M_i(m) - M_i(\texttt{control})$ (e.g., $\Delta\text{Hoyer}_i = \text{Hoyer}_i(\texttt{tag\_correct}) - \text{Hoyer}_i(\texttt{control})$). Paired differences reduce variance from instance-specific difficulty. We test Δ effects using paired t-tests or Wilcoxon signed-rank tests (with Benjamini–Hochberg FDR correction for multiple comparisons). Unless otherwise stated, correlations between Δ metrics are Pearson correlations; Spearman results are reported in Appendix D and are consistent with the main conclusions.

5 Experimental Results

This section presents the experimental results and analyses for RQ1–RQ3 in turn.

5.1 RQ1: How Do Routing meta prompts Change Internal Density?

Figure 1 reports instance-level paired differences in Hoyer sparsity (ΔHoyer) for `tag_correct` and `instr_expert` relative to `control`, as well as `tag_correct` relative to `instr_expert`.

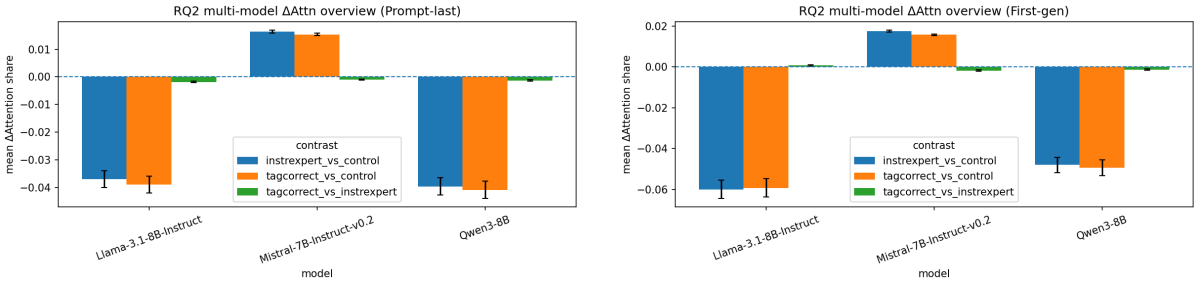
Densification vs. control. Across all three models and most sub-domains, both `instr_expert` and `tag_correct` yield negative ΔHoyer in the Early/Middle segments (typically 0.005–0.015 in magnitude), indicating a shift toward denser and more evenly distributed activations. In contrast, effects in the Late segment are markedly smaller or near zero, suggesting that routing-style meta prompts primarily reshape earlier semantic representations rather than layers close to the output head. As a sanity check, `tag_placebo` shows ΔHoyer close to zero, implying that prefix length or formatting alone cannot explain the densification effect (see Appendix C for full statistics).

Expert instructions are stronger than tags. Directly comparing `tag_correct` with `instr_expert`, we observe a consistently positive global-average gap (Hoyer(`tag_correct`) − Hoyer(`instr_expert`) > 0) across all three models. Since larger Hoyer values indicate higher sparsity, this means that natural-language expert instructions induce stronger densification than structured route tags. This reverses the naive expectation that short, formatted tags would behave as a closer proxy to "true" routing signals, and instead highlights the sensitivity of instruction-tuned LLMs to semantically rich natural-language task framing (Appendix B).

5.2 RQ2: How Do Routing meta prompts Modulate Domain-Keyword Attention?

We measure C2 as the last-layer attention share assigned to domain-keyword tokens. We estimate this share from two query viewpoints: **Prompt-last** (query at the last input token) and **First-gen** (query at the first generated token). Figure 2 reports instance-level paired differences ΔAttn under three contrasts (`tag_correct` vs. control, `instr_expert` vs. control, and `tag_correct` vs. `instr_expert`).

Model-specific attention strategies. Llama-3.1-8B-Instruct and Qwen3-8B show consistent *decreases* in keyword attention after adding routing-style prefixes (about -0.04 in Prompt-last and $-0.05 \sim -0.06$ in First-gen for Tag/Instr vs. control), suggesting an



(a) Prompt-last.

(b) First-gen.

Figure 2: Multi-model overview of ΔAttn (mean ± SEM). Negative values indicate reduced attention share to domain keywords.

offloading-like pattern: the explicit routing cue partially substitutes for lexical domain cues when the model begins responding. In contrast, Mistral-7B-Instruct-v0.2 shows a stable *increase* (about +0.016 in both views), indicating *reinforcement*: the model attends to the routing cue *and* the domain keywords more strongly rather than trading one for the other. Across models, Tag and Instr typically have comparable magnitudes within each view.

ΔAttn vs. ΔC3: weak coupling. Across contrasts and models, changes in keyword attention provide limited explanatory power for output stability: correlations between ΔAttn and ΔC3 are small overall (at best weak-to-moderate for entropy in Qwen, and typically closer to zero for semantic variation; see Appendix C.2). Thus, keyword attention is better interpreted as a task-identification covariate than as a direct driver of stability.

5.3 RQ3: Linking Internal Densification to Output Stability

RQ3 tests a core link in the Sparsity–Certainty Hypothesis: whether instance-level changes in internal density (ΔC1) co-vary with changes in output stability (ΔC3). We instantiate C1 with prompt-segment Hoyer sparsity and measure C3 by predictive entropy and semantic variation (1 − mF1). For each model and each prefix contrast (`instr_expert` vs. `control`, `tag_correct` vs. `control`, and `tag_correct` vs. `instr_expert`), we compute Pearson correlations on paired differences; full statistical details are in Appendix C.3.1. Figure 3 summarizes the results.

Model-specific coupling. Qwen shows a consistent positive association between densification and lower entropy, with correlations typically in the 0.2–0.3 range across contrasts (Appendix C.3.1). In contrast, for Llama and Mistral the corresponding correlations are near zero across contrasts, indicating that densification provides little predictive signal for entropy changes in these models.

Semantic variation is weaker overall. Replacing entropy with semantic variation (1 − mF1) further attenuates the relationship: Qwen retains only a small positive association, while Llama and Mistral remain close to zero (Appendix C.3.1).

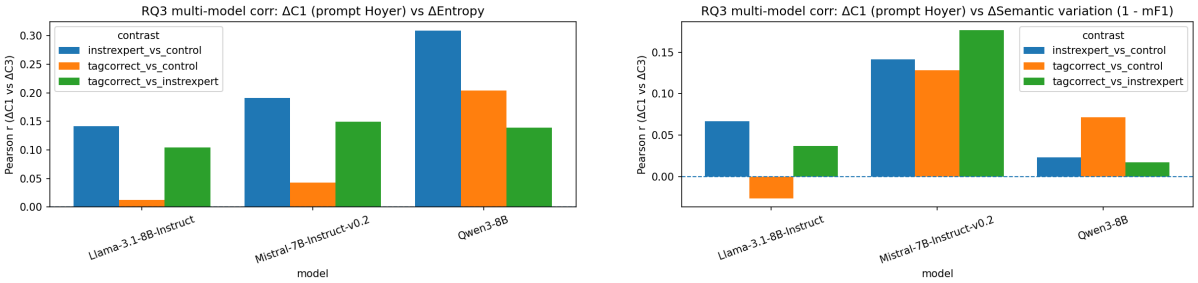
Takeaway. The densification–stability link is not a model-invariant property: it appears in Qwen but largely breaks in Llama and Mistral, suggesting that using internal density as a general-purpose uncertainty proxy requires model-specific calibration.

6 Discussion: RIDE as a Diagnostic Probe Rather Than a Universal Routing Law

Taken together, RQ1–RQ3 paint a picture that is far more complex than the linear assumption "routing ⇒ sparsity ⇒ stability." Here we discuss the positioning and implications of RIDE from three perspectives.

6.1 From the Sparsity–Certainty Hypothesis to a Model-Specific Density–Stability spectra

Our original motivation was to test the intuition that "effective routing selects sparser and



(a) Entropy

(b) Semantic variation

Figure 3: Instance-level correlations between ΔC1 (prompt Hoyer) and ΔC3 across models and contrasts. Positive values indicate that instances with larger decreases in Hoyer tend to also exhibit larger decreases in the corresponding instability measure (entropy or semantic variation).

more certain pathways." However, the experimental results show that:

- At the density level: task-oriented meta prompts generally induce densification rather than increased sparsity;
- At the stability level: a local "densification–stability coupling" is observed only in Qwen, while it is largely absent in Llama and Mistral.

Rather than a universal "sparsity–certainty law," our findings suggest that different models form their own distinctive density–stability spectra under routing-style meta prompts: in some models, densification has a partially positive association with stability, whereas in others the two factors are nearly decoupled. This provides an important caution against directly transferring internal proxies across models.

6.2 Implications for Routing Design and Uncertainty Estimation

From an engineering standpoint, our results suggest that:

- Routing signals should be calibrated in a model-specific manner. A proxy that works well for one model (e.g., C1) may completely fail for another, or even yield an opposite signal.
- Natural-language instructions can themselves serve as strong routing signals. For instruction-tuned models without specialized fine-tuning, short expert instructions can substantially reshape internal density and attention structure, opening up design space for "prompt-level routing."
- Incorrect or ambiguous routing tags may introduce additional instability. In the absence of a real router, indiscriminate use of "expert tags" may inject noise into internal states, potentially harming stability and safety.

In practical systems, RIDE can be used as a diagnostic tool: prior to deployment, one can run RIDE analyses on candidate models to determine which internal proxies carry stable semantics for that model, and only then consider using them for routing or uncertainty estimation.

6.3 Scope and Limitations of RIDE

RIDE offers a statistical diagnostic perspective that combines meta-prompt interventions with internal metrics, rather than precise identification of causal structure. We do not advocate directly using C1–C3 as decision signals in production systems. Instead, we suggest treating them as auxiliary tools for model understanding and iterative design—for example, selecting models that are better suited for "prompt-level routing," or identifying architectures that are particularly sensitive to incorrect routing tags.

7 Future Directions

Future work can extend RIDE along several directions:

- Extend the framework to real MoE gating and multi-model routing logs, combining real routing signals with meta prompts;
- Incorporate additional internal metrics (e.g., concept activations, specific neu-

rons/subspaces) to enrich the characterization of density–stability dynamics;

- Systematically evaluate the fairness and robustness of RIDE metrics on multilingual, multi-group, and safety-critical tasks;

- Explore using RIDE as part of the routing-system design workflow, e.g., tracking how the density–stability spectra evolves before and after training.

8 Conclusion

We propose RIDE (Route-Induced Density and Stability), a framework that performs controlled interventions on frozen instruction-tuned LLMs via routing-style meta prompts, and systematically analyzes how routing signals affect internal density (C1), domain-keyword attention (C2), and output stability (C3). Based on large-scale experiments on a RouterEval subset across three open-source instruction-tuned models, we find that:

- Task-oriented meta prompts consistently induce densification of internal representations rather than increased sparsity;

- Models exhibit pronounced heterogeneity in attention redistribution (cognitive offloading vs. attention reinforcement);

- The "densification $\Rightarrow$ stability" link receives only limited support in a subset of models and lacks cross-model robustness.

Taken together, these findings suggest that directly treating internal density or attention concentration as a universal "good routing signal" or a general-purpose uncertainty proxy can be risky. We argue that RIDE is better positioned as a diagnostic probe: through controlled, statistical analyses, it reveals model-specific density–stability spectra under routing-style meta prompts, providing fine-grained evidence for routing design, uncertainty estimation, and model selection.

9 Limitations

This work has several important limitations (which we state explicitly to clarify the scope of generalization):

1. Proxy nature of routing signals and external validity. We use routing-style meta prompts as a textual proxy for real MoE gating or multi-model router signals. This proxy cannot capture routing distributions, logging features, or training couplings present in real systems; therefore, our conclusions primarily apply to intervention analyses of "prompt-level routing signals."

2. Limited coverage of tasks and models. Our experiments focus on a subset of RouterEval and three mid-sized open-source instruction-tuned models. Larger-scale models, different alignment strategies or specialist models, and different task distributions—especially long-context and tool-use settings—may exhibit different RIDE spectra.

3. Dependence on metric construction and decoding settings. The concrete implementations of C1–C3 (e.g., the exact Hoyer sparsity formulation, keyword-list coverage, sampling counts, and temperature) affect numerical scales. We provide several robustness checks in the appendix, but cannot exhaustively cover all configurations; using RIDE as a system signal requires model- and scenario-specific recalibration.

4. Boundaries of causal interpretation. Although we use paired interventions and control decoding and randomness, RIDE remains a "causal-style" statistical comparison rather than structured causal identification. We cannot fully rule out the influence of unobserved confounders (e.g., training history or alignment preferences) on the observed effects.

Ethical considerations

All experiments in this work are conducted using open-source models and publicly available datasets, without involving user privacy or sensitive data. The code and derived artifacts we release will not contain any personally identifiable information. RIDE is primarily intended as an analysis and diagnostic tool; any application that uses internal proxies for safety-critical decisions (e.g., filtering sensitive con-

tent, or routing decisions in medical or legal settings) should undergo rigorous safety evaluation and ethical review.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, and 1 others. 2025. Circuit tracing: Revealing computational graphs in language models. Transformer Circuits (Anthropic).

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*. Published: 20 Dec 2024.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, and 8 others. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

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

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. arXiv preprint. *Preprint*, arXiv:2403.12031.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2025. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3860–3887, Suzhou, China. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations (ICLR)*.

Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. 2024. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32287–32307. PMLR.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jingcheng Niu, Xingdi Yuan, Tong Wang, Hamidreza Saghir, and Amir H. Abdi. 2025. Llama see, llama do: A mechanistic perspective on contextual entrainment and distraction in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16218–16239, Vienna, Austria. Association for Computational Linguistics.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He.

2022. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. 2025. IRT-router: Effective and interpretable multi-LLM routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15629–15644, Vienna, Austria. Association for Computational Linguistics.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. In *International Conference on Learning Representations (ICLR)*.

Artem Vazhentsev, Gleb Kuzmin, Artem Shelmanov, Akim Tsvigun, Evgenii Tsymbalov, Kirill Fedyanin, Maxim Panov, Alexander Panchenko, Gleb Gusev, Mikhail Burtsev, Manvel Avetisian, and Leonid Zhukov. 2022. Uncertainty estimation of transformer predictions for misclassification detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8237–8252, Dublin, Ireland. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. Back attention: Understanding and enhancing multi-hop reasoning in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11268–11283, Suzhou, China. Association for Computational Linguistics.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M. Susskind. 2023. Stabilizing transformer training by preventing attention entropy collapse. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 40770–40803. PMLR.

Lin Zhang, Wenshuo Dong, Zhuoran Zhang, Shu Yang, Lijie Hu, Ninghao Liu, Pan Zhou, and Di Wang. 2025. Eap-gp: Mitigating saturation effect in gradient-based automated circuit identification. In *Advances in Neural Information Processing Systems*. NeurIPS 2025 poster.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. arXiv preprint. *Preprint*, arXiv:2202.08906.

A Experimental and Implementation Details

A.1 Backbone Models and Hyperparameter Settings

We conduct experiments on three open-source, instruction-tuned decoder-only LLMs. Throughout the appendix, we refer to them as Model-A / Model-B / Model-C, corresponding in the main text to:

- Model-A: Llama-3.1-8B-Instruct
- Model-B: Mistral-7B-Instruct-v0.2
- Model-C: Qwen3-8B

All three models follow an autoregressive Transformer architecture, stacking multi-head self-attention and feed-forward networks. They are pretrained primarily on English corpora and further instruction-tuned on multi-task instruction data. Unless stated otherwise, we adopt the following common settings:

- Frozen parameters: In all experiments, the backbone model parameters are fully frozen; interventions are applied only by modifying the input prefix (routing-style meta prompts).
- Chat formatting: We use each model' s official tokenizer and chat/instruction formatting (system/user structure in chat/instruct mode), keeping the default system/assistant role templates unchanged.
- Unified decoding hyperparameters: For a given model, decoding hyperparameters are kept identical across all prefix conditions.
- Randomness control: For the same input instance, all prefix conditions share the same set of random seeds, ensuring that observed differences are primarily attributable to the prefix itself.

A.2 RouterEval Preprocessing and Sampling

A.2.1 Constructing the RouterEval Training Table

We first build a unified training/analysis table based on the public RouterEval dataset. Using a custom script, we generate a training-level sample table from the original structured RouterEval data. The command is as follows:

```
python build_routereval_dataset.py \
  --router_dataset_dir <dir_root> \
  --scenarios gsm8k,math,ifeval,\
  winogrande,mmlu \
  --difficulties easy,hard \
  --num_candidates 3,5,10 \
  --candidate_groups all_strong,\
  all_weak,strong_to_weak \
  --splits train \
  --output_csv <routereval_train_csv>
```

- scenarios cover representative settings including arithmetic reasoning (gsm8k), mathematics/logic (math/mmlu), format compliance (ifeval), and commonsense reasoning (winogrande/mmlu), among others;
- difficulties include easy and hard, spanning different difficulty levels;
- The candidate configuration (`num_candidates`, `candidate_groups`) follows the original RouterEval settings to facilitate alignment with prior routing research.

A.2.2 Building the Domain-Specific Items Table

On top of this, we construct an items table for RIDE analysis. The command is:

```
python make_items_from_routereval.py \
  --input_csv <routereval_train_csv> \
  --output_csv <items_routereval_csv> \
  --domains math,ifeval,commonsense \
  --max_per_domain 200 \
  --drop_duplicates
```

- domains selects three coarse-grained domains: Math/Code, Format/Verification (IFEval), and Commonsense;
- max_per_domain caps the number of samples per domain at approximately 200, balancing diversity and computational cost;
- drop_duplicates removes duplicate or near-duplicate items.

The resulting <items_routereval_csv> serves as a unified "analysis corpus table" across models and prefix conditions in subsequent experiments, recording metadata such as the task text, domain labels, and source scenarios.

A.2.3 Sample Filtering and Summary Statistics

Based on the items table above, we apply lightweight filtering:

- We remove overly long instances (exceeding a predefined maximum token length) to avoid model-dependent truncation effects;
- We remove extremely short instances (fewer than a minimum number of tokens) to prevent unstable estimates for metrics such as C2/C3.

After filtering, we retain approximately 200 instances per domain, forming a fixed analysis corpus. Table A.2 reports summary statistics for each domain, including the number of instances, the average token length, and the keyword matching rate. Because our focus is on internal change patterns under prefix interventions rather than generalization performance on this dataset, we do not further split this corpus into train/dev/test. Instead, we treat it as a fixed analysis set and compare different models on the same collection.

A.3 Router-Style Meta-Prompt Templates and Tag Construction

A.3.1 Route Tags

We adopt a unified structured format for route tags, as follows: `[RouteTag=DOMAIN]` where `DOMAIN` $\in$ {`math`, `code`, `format`, `commonsense`}.

- tag_correct: uses a tag that matches the instance domain (e.g., using [RouteTag=math] for a math problem);
- tag_wrong: uniformly samples an incorrect tag from the other domains (e.g., using [RouteTag=code] for a math instance).

A.3.2 Placebo Tags

Placebo tags are used to control for form factors—i.e., adding extra tokens in a similar format—without introducing meaningful semantics. We construct tags of the form: `[RouteTag=XXXXX]` where XXXXX is a meaningless string that rarely appears in the RouterEval corpus and has a length comparable to real route tags. Semantically, tag_placebo should be approximately equivalent to control, differing only in surface form by the additional prefix tokens.

A.3.3 Expert Prompts

We design a small set of natural-language expert prompt templates for different domains, for example:

- Math You are a Math Expert. Please solve the problem step by step. (A Chinese variant is also used: "你是一名数学专家，请逐步推理并给出答案。")
- Format / IFEval You are a formatting assistant. Please strictly follow the required output format.
- Commonsense You are a reasoning assistant. Please choose the most plausible option and briefly explain why.

In the main experiments, we use a fixed template per domain to reduce additional variance. In Appendix C, we conduct sensitivity checks with partially randomized template sets.

A.3.4 Prefix Injection Location and Dialogue Structure

Unless otherwise specified, all route tags and expert prompts are injected before the user prompt, e.g., [RouteTag=math] <user prompt ...> or You are a Math Expert. Please solve the problem step by step. <user prompt ...> Other parts of the system/assistant roles (e.g., the system prompt and assistant prefix) are kept unchanged under each model' s default settings. Across prefix conditions, the only differences lie in the routing-style meta-prompt segment described above.

A.4 Domain Keyword Lexicon (`DOMAIN_KEYWORDS`)

The domain keyword lexicon is used to construct the C2 metric (domain-keyword attention share). We build it as follows:

1. Collecting seed keywords For each subdomain, we manually curate a small set of seed keywords:

- Math: equation, integer, sum, proof, factor
- Code: function, variable, class, compile, error
- Format: json, table, markdown, field, column
- Commonsense: choose, option, likely, most, best).

These seeds aim to cover the most central semantic concepts or task-triggering terms in the domain.

2. Automatic expansion and filtering:
   - We use an auxiliary script to automatically expand the candidate list on the RouterEval training table using statistics such as TF-IDF / PMI;
   - We remove stopwords, punctuation, and extremely frequent but domain-agnostic words (e.g., the, is, do), and normalize case and inflectional forms.
3. Manual screening and finalization:
   - We perform a quick manual inspection of the expanded candidates and discard clearly inappropriate entries;
   - We then form the final DOMAIN_KEYWORDS dictionary and save it as a JSON file for downstream analysis scripts.

For each instance, after tokenization, we match terms from DOMAIN_KEYWORDS and record their token positions:

- When computing the domain-keyword attention share, the target set consists of these token positions;
- If an instance has no keyword match under its domain, we define its domain-keyword attention share as 0 (since no attention mass falls on the keyword set), and mark it as "missing keywords" in the summary table (used to analyze the signal-to-noise ratio of C2).

A.5 Extracting Activations and Attention

A.5.1 Layer-Segment Partitioning

Suppose the model contains L Transformer blocks. We evenly partition them into three segments:

- Early: Layers 1 to $\lfloor L/3 \rfloor$;
- Middle: Layers $\lfloor L/3 \rfloor + 1$ to $\lfloor 2L/3 \rfloor$;
- Late: remaining layers.

For each segment, we aggregate across instances and tokens (e.g., by taking the mean) to obtain segment-level metrics.

A.5.2 Caching Hidden States and Attention Weights

For each instance and each prefix condition, we cache the following during the forward pass:

- The hidden vector at every token for every layer, $\mathbf{h}_t^{(\ell)}$;
- Attention weights used for C2:
  - Prompt-last: the attention over all prompt tokens at the final step before generation begins;
  - First-gen: the attention from the first generated token to all prompt tokens.

In practice, we enable output_hidden_states=True and output_attentions=True in the inference API, or use framework-provided hooks to write intermediate results to files (e.g., Parquet / HDF5) for offline analysis.

A.5.3 Attention Normalization and Keyword-Share Computation

When computing the keyword attention share, we preprocess the attention vector of each layer as follows:

- We retain attention mass only over visible tokens, including system/user/prompt tokens, while excluding padding and special tokens (e.g., BOS/EOS);
- We re-normalize over the visible-token set so that the weights sum to 1;
- We sum the weights over positions that match DOMAIN_KEYWORDS, yielding the layer-level keyword attention share;
- We then aggregate across layer segments and instances to obtain the segment-level C2 metric.

A.6 Decoding and Output Sampling (C3 Metric)

A.6.1 Decoding Hyperparameters

For experiments related to output stability, we use stochastic sampling with temperature. Unless otherwise specified, we adopt the following unified hyperparameters:

- Sampling temperature T=0.7;
- top-p = 0.9;
- No additional repetition penalty;
- A fixed generation length limit max_new_tokens (e.g., 64).

These settings are applied uniformly across all models and all prefix conditions.

A.6.2 Multi-Sample Generation (Multi-sample Forward)

To estimate C3 (predictive entropy and semantic variation), we generate K independent samples for each prompt under each prefix condition. In the main experiments:

- We use K=5 samples by default to estimate predictive entropy and semantic variation;
- For the same instance, we reuse the same sequence of random seeds across different prefix conditions, ensuring that C3 differences are primarily attributable to prefix differences rather than sampling randomness.

Predictive entropy is computed at the token level and then averaged over the sequence. Semantic variation is computed based on similarities among sentence embeddings obtained from multiple samples; the formal definition is provided in Appendix B. Appendix C reports sensitivity analyses under different K values and temperatures.

A.6.3 Sharding and Parallel Inference

To improve computational efficiency, we shard <items_routereval_csv> by splitting the instances into multiple shards and run inference and metric computation in parallel on a multi-GPU setup. The overall workflow is:

- Use an auxiliary script to split the items CSV into multiple subfiles by row count (e.g., items_part0.csv, items_part1.csv, $\cdots$);
- Assign one or more shards to each GPU and run a unified inference script (which loads the model, applies prefixes, extracts hidden states and attention, and generates multi-sample outputs);
- Save intermediate results for each shard (e.g., MFV metrics and generated texts) as structured files, and merge them during post-processing.

All shards are executed under the same configuration, ensuring comparability across GPUs and shards.

A.7 Statistical Tests and Correlation Estimation

To obtain causal-style paired evidence and to avoid over-interpreting any single scalar result, we adopt the following statistical procedures: Paired difference tests

- For paired metric differences (e.g., ΔHoyer, ΔEntropy, ΔKeywordShare), we use a paired t-test or the Wilcoxon signed-rank test to assess whether the differences significantly deviate from zero;
- When appropriate, we report effect sizes (e.g., Cohen' s d) to reflect the practical magnitude of the differences.

Correlation analysis

- To quantify the linear relationship between ΔC1 and ΔC3, we use the Pearson correlation coefficient;
- We estimate the 95% confidence interval of Pearson r via the Fisher transformation;
- In Appendix B, we additionally report Spearman' s rank correlation as a robustness check.

Multiple-comparison correction

- For the large number of tests across models, domains, and layer segments, we control the false discovery rate (FDR) using the Benjamini–Hochberg procedure, reporting significance at a unified q-level;

- In the main paper, we report only conclusions with clear direction and magnitude that remain stable under multiple-comparison correction, avoiding over-interpretation of marginally significant results.

B Full Mathematical Definitions of Metrics

This appendix provides the formal mathematical definitions of C1/C2/C3 and several auxiliary metrics used in Section 4 of the main paper. Notation is kept consistent with the main text.

B.1 Notation

- We denote an instance by x, and a routing prefix condition by $m \in \mathcal{M}$ (e.g., `control`, `tag_correct`, `tag_wrong`, `tag_placebo`, `instr_expert`).

- The model has L Transformer blocks. The hidden vector at layer ℓ and token position t is $\mathbf{h}_{x,t}^{(\ell)} \in \mathbb{R}^d$.

- Let the generated sequence length be T_x. The predictive distribution for the t-th generated token under condition m is $p_t^{(m)}(v)$.

- For attention, we denote the last layer by $\ell = L$, and its attention vector by $\mathbf{a}_{x,q}^{(L,m)}$, where q is the query position (e.g., `prompt-last` or `first-gen`).

- The set of token positions in the prompt segment is $\mathcal{P}_x$, and the set of domain-keyword positions is $\mathcal{K}_x \subseteq \mathcal{P}_x$.

Unless otherwise stated, instance-level metrics are first averaged over tokens and/or layers, and then subjected to statistical analysis over the set of instances.

B.2 C1: Activation Sparsity / Density

In Section 4.1 of the main paper, C1 is described as a combined metric of segment-level Hoyer sparsity and Top-k energy. Here we provide precise definitions of both components and explain how C1 is constructed.

B.2.1 Hoyer Sparsity

Given a hidden vector $\mathbf{h} \in \mathbb{R}^d$, the Hoyer sparsity is defined as:

$$\mathrm{Hoyer}(\mathbf{h}) = \frac{\sqrt{d} - \frac{\|\mathbf{h}\|_1}{\|\mathbf{h}\|_2}}{\sqrt{d} - 1}.$$

where $\mathbf{h} \in \mathbb{R}^d$ and $\mathrm{Hoyer}(\mathbf{h}) \in [0, 1]$. Larger values indicate higher sparsity and stronger concentration on a small subset of dimensions. For an instance x, prefix condition m, layer ℓ, and token position t, the token-level Hoyer score is:

$$s_{x,t}^{(\ell,m)} = \mathrm{Hoyer}\left(\mathbf{h}_{x,t}^{(\ell,m)}\right).$$

We average over a layer segment. For example, for the Early segment (layer set $\mathcal{L}_{\mathrm{Early}}$), we define:

$$\mathrm{C1\_Hoyer}^{\mathrm{Early}}(x, m) = \frac{1}{\left|\mathcal{L}_{\mathrm{Early}}\right| \cdot |\mathcal{P}_x|} \times \sum_{\ell \in \mathcal{L}_{\mathrm{Early}}} \sum_{t \in \mathcal{P}_x} s_{x,t}^{(\ell,m)}.$$

The Middle and Late segments are defined analogously. For the first-generated-token variant (C1-firstgen), we replace $\mathcal{P}_x$ with the token position corresponding to the first generated token. The "Hoyer part of C1" in the main paper refers to this type of segment-level aggregation.

B.2.2 Top-k Energy Ratio

Let $|h_{(1)}| \geq \cdots \geq |h_{(d)}|$ denote the components of $\mathbf{h} \in \mathbb{R}^d$ sorted by absolute value. The Top-k energy ratio is defined as:

$$\mathrm{TopK}(\mathbf{h}) = \frac{\sum_{i=1}^{k} h_{(i)}^2}{\sum_{i=1}^{d} h_i^2}, \qquad k = \lfloor \alpha d \rfloor,$$

with $\alpha = 0.1$ by default.

In our setting, decreases in Hoyer (densification) are typically accompanied by decreases in the Top-k energy ratio, indicating that energy is more evenly distributed across a larger number of dimensions.

Similarly, we define the token-level $\mathrm{TopK}_{x,t}^{(\ell,m)}$ and average over layer segments

and prompt-token positions. For example, for the Early segment:

$$\text{C1\_TopK}^{\text{Early}}(x,m) = \Big(\big|\mathcal{L}_{\text{Early}}\big|\,|\mathcal{P}_x|\Big)^{-1} \times \sum_{\ell\in\mathcal{L}_{\text{Early}}}\sum_{t\in\mathcal{P}_x} \text{TopK}_{x,t}^{(\ell,m)}.$$

B.2.3 Combining C1 and Paired Differences

Combining C1. We treat $C1$ as a combined metric consisting of two components, C1_Hoyer and C1_TopK:

- In Sections 4.1–5.1 of the main paper, almost all figures and conclusions are based on C1_Hoyer.
- C1_TopK and its relationship with Hoyer are reported in supplementary figures/tables in Appendix C, mainly to verify that "densification" is not an artifact caused by extreme amplification of only a few dimensions.

Paired differences. At the instance level, we focus on paired differences between a given prefix condition and the control. For an instance x, the Hoyer difference between `tag_correct` and `control` is defined (consistent with Section 4.4) as:

$$\Delta\text{Hoyer}_x = \text{C1\_Hoyer}\big(x,\texttt{tag\_correct}\big) - \text{C1\_Hoyer}\big(x,\texttt{control}\big).$$

In implementation, ΔC1 corresponds to the combination of ΔHoyer and ΔTopK (we report one or both depending on the figure/table). In the main paper, "densification" typically refers to the case $\Delta\text{Hoyer}_x < 0$.

B.3 Auxiliary Metrics for Activation-Distribution Shape

To characterize activation distributions in a more fine-grained manner, we additionally compute the following metrics for each token-level vector $\mathbf{h} \in \mathbb{R}^d$:

- **Gini coefficient:** computed from the sequence $\big(|h_i|\big)_{i=1}^d$ using the standard definition, measuring inequality in energy allocation across dimensions;
- **Kurtosis:** quantifies whether the distribution exhibits sharp peaks and heavy tails;
- **Positive ratio:**

$$\text{PosRatio}(\mathbf{h}) = \frac{1}{d}\sum_{i=1}^{d}\mathbf{1}[h_i > 0]\,,$$

capturing the sign pattern of activations.

These metrics can likewise be averaged over layer segments and used to form paired differences, but they are only used for robustness checks in Appendix C and are not treated as primary evidence for the main-paper conclusions.

B.4 Cross-Layer Energy Concentration (Layer-Energy Gini)

To examine whether meta prompts alter the allocation of "computational load" across layers, we define a cross-layer energy Gini metric as follows. At a key position in the target sequence (e.g., prompt-last or first-gen), we compute the ℓ-th layer's ℓ_2 energy:

$$E_\ell(x,m) = \Big\|\mathbf{h}_{x,t^\star,m}^{(\ell)}\Big\|_2^2, \qquad \ell = 1,\dots,L,$$

where $t^\star$ denotes the token position of interest.

We then normalize the energies across layers:

$$\tilde{E}_\ell(x,m) = \frac{E_\ell(x,m)}{\sum_{\ell'=1}^{L} E_{\ell'}(x,m)}.$$

Treating $\{\tilde{E}_\ell(x,m)\}_{\ell=1}^{L}$ as a one-dimensional discrete distribution, we compute its Gini coefficient, denoted as $\text{Gini}_{\text{layer}}(x,m)$. Larger values indicate that energy is highly concentrated in a small subset of layers, whereas smaller values indicate a more even distribution across layers. This metric is used for supplementary analyses in Appendix C but is not directly referenced in the main paper.

B.5 C2: Domain-Keyword Attention

Section 4.2 of the main paper defines C2 as the attention share to domain keywords in the last layer, estimated from two viewpoints: PROMPT-LAST and FIRST-GEN. This section presents the formal definition.

Let $\mathcal{P}_x$ denote the set of prompt-token positions for an instance x, and let $\mathcal{K}_x \subseteq \mathcal{P}_x$ denote the subset of positions matched by domain keywords. Under prefix condition m, we denote

the attention vector at the last layer ($\ell = L$) for a query position q by $\mathbf{a}_{x,q}^{(L,m)} \in \mathbb{R}^{|\mathcal{P}_x|}$, where the element $a_{x,q,t}^{(L,m)}$ is the attention weight assigned to position t.

We first renormalize attention over the set of visible tokens $\mathcal{V}_x \subseteq \mathcal{P}_x$ such that

$$\sum_{t \in \mathcal{V}_x} a_{x,q,t}^{(L,m)} = 1.$$

The domain-keyword attention share at query position q is then defined as

$$\mathrm{C2}(x, m, q) = \sum_{t \in \mathcal{K}_x} a_{x,q,t}^{(L,m)}.$$

In the main paper, we consider two query viewpoints:

- PROMPT-LAST: set $q = q_{\text{promptlast}}$ as the last input-token position, corresponding to the state after "reading the question";
- FIRST-GEN: set $q = q_{\text{firstgen}}$ as the first generated-token position, corresponding to the state when "starting to answer".

For each viewpoint, we average over instances:

$$\mathrm{C2}^{\text{PromptLast}}(m) = \frac{1}{|X|} \sum_{x \in X} \mathrm{C2}(x, m, q_{\text{promptlast}}),$$

and analogously for the FIRST-GEN variant.

At the instance level, paired differences follow the Δ notation in Section 4.4. For example,

$$\Delta\mathrm{C2}_x^{\text{PromptLast}} = \mathrm{C2}(x, m, q_{\text{promptlast}}) - \mathrm{C2}(x, \text{control}, q_{\text{promptlast}}).$$

A positive ΔC2 indicates that prefix m increases the attention share to domain keywords relative to the control.

Note. In some figures/tables in Appendix C, we further extend C2 to Early/Middle/Late segments by averaging attention across multiple layers. However, the core results shown in Sections 4.2–5.2 of the main paper are based on the last-layer definition above.

If an instance has no keyword matches in its domain ($\mathcal{K}_x = \emptyset$), we define $\mathrm{C2}(x, m, q) = 0$ and track the proportion of such "missing-keyword" instances separately in the summary tables.

B.6 C3: Output Stability

Section 4.3 of the main paper defines C3 as a pair of measures: predictive entropy and semantic variation. This section provides the formal definitions.

B.6.1 Predictive Entropy

For an instance x under a prefix condition m, let $p_t^{(m)}(v)$ denote the predictive distribution over vocabulary items v for the t-th generated token. The token-level predictive entropy is

$$H_t(x, m) \;=\; -\sum_v p_t^{(m)}(v)\, \log p_t^{(m)}(v).$$

Let T_x be the generated sequence length. The sequence-level average entropy is

$$\bar{H}(x, m) \;=\; \frac{1}{T_x} \sum_{t=1}^{T_x} H_t(x, m).$$

We use $\bar{H}(x, m)$ as the first C3 measure: lower values indicate a more concentrated predictive distribution and hence higher confidence.

At the instance level, we define the paired difference

$$\Delta\text{Entropy}_x \;=\; \bar{H}(x, m) - \bar{H}(x, \text{control}),$$

where m can be a prefix condition such as `tag_correct` or `instr_expert`. The ΔEntropy reported in Section 5.3 refers to this quantity.

Note. In implementation, we compute $p_t^{(m)}(v)$ directly from a single forward-pass softmax. Alternatively, one may average distributions over multiple samples; the qualitative conclusions remain essentially the same.

B.6.2 Semantic Variation

Semantic variation is computed via repeated sampling and sentence-embedding similarity. For the same prompt-condition pair (x, m), we generate K independent outputs $y^{(1)}(x, m), \ldots, y^{(K)}(x, m)$ and encode them using a fixed sentence encoder $f(\cdot)$:

$$\mathbf{e}^{(k)}(x, m) \;=\; f\big(y^{(k)}(x, m)\big), \quad k = 1, \ldots, K.$$

We then compute all pairwise cosine similarities

$$s_{k\ell}(x, m) = \frac{\mathbf{e}^{(k)}(x, m) \cdot \mathbf{e}^{(\ell)}(x, m)}{\|\mathbf{e}^{(k)}(x, m)\|\, \|\mathbf{e}^{(\ell)}(x, m)\|}$$
$$1 \le k < \ell \le K.$$

We treat the average pairwise similarity as a proxy for semantic F1 (mF1):

$$\mathrm{mF1}(x, m) \;=\; \frac{2}{K(K-1)} \sum_{1 \le k < \ell \le K} s_{k\ell}(x, m).$$

Semantic variation is then defined as

$$\mathrm{Var}(x, m) \;=\; 1 - \mathrm{mF1}(x, m).$$

Smaller $\mathrm{Var}(x, m)$ indicates that multiple generations are closer in semantic space and thus more stable. The paired difference is

$$\Delta\mathrm{Var}_x \;=\; \mathrm{Var}(x, m) - \mathrm{Var}(x, \mathrm{control}).$$

Following Section 4.3, concrete choices of the encoder, the sampling count K, and other hyperparameters are deferred to Appendix C. In this paper, C3 = {Entropy, Var}, and we analyze ΔEntropy and ΔVar separately.

B.7 Commonsense Confidence Margin (Optional Analysis)

For multiple-choice commonsense tasks, let $\mathcal{C}$ denote the set of answer options, and let $c_{\mathrm{gold}} \in \mathcal{C}$ be the correct option. We define the average probability of an option c as

$$\bar{p}(c; x, m) \;=\; \frac{1}{G_x} \sum_{t \in \mathcal{G}_x} p_t^{(m)}(c),$$

where $p_t^{(m)}(c)$ is the probability assigned to choosing option c at position t (the exact scoring procedure is described in Appendix C), and $\mathcal{G}_x$ denotes the set of positions used for option scoring.

We define the confidence margin as

$$\Delta_{\mathrm{conf}}(x, m) \;=\; \bar{p}(c_{\mathrm{gold}}; x, m) - \max_{\substack{c \in \mathcal{C} \\ c \neq c_{\mathrm{gold}}}} \bar{p}(c; x, m).$$

Larger $\Delta_{\mathrm{conf}}(x, m)$ indicates a stronger confidence advantage for the correct option and hence lower uncertainty. In Appendix C.5, we use this quantity as an auxiliary metric and analyze it alongside the two primary C3 metrics (Entropy and Var), without separately elaborating it in the main text.

With these definitions, all terms appearing in Sections 4.1–4.4 of the main paper—such as C1/C2/C3, ΔHoyer, Prompt-last, and First-gen—have precise counterparts in this appendix, and this section can be directly included in Appendix B.

Model	ΔHoyer **Tag–Ctrl**	ΔHoyer **Instr–Ctrl**	ΔHoyer **Tag–Instr**
Llama-3.1-8B-Instruct	-0.015 ± 0.001	-0.016 ± 0.001	$+0.001 \pm 0.000$
Mistral-7B-Instruct-v0.2	-0.016 ± 0.001	-0.017 ± 0.001	$+0.000 \pm 0.000$
Qwen3-8B	-0.005 ± 0.000	-0.003 ± 0.000	-0.002 ± 0.000

Table 1: Early segment: mean ± SEM of ΔHoyer under three contrasts.

Model	ΔHoyer **Tag–Ctrl**	ΔHoyer **Instr–Ctrl**	ΔHoyer **Tag–Instr**
Llama-3.1-8B-Instruct	-0.014 ± 0.001	-0.015 ± 0.001	$+0.000 \pm 0.000$
Mistral-7B-Instruct-v0.2	-0.012 ± 0.001	-0.015 ± 0.001	$+0.003 \pm 0.000$
Qwen3-8B	-0.007 ± 0.001	-0.009 ± 0.001	$+0.002 \pm 0.000$

Table 2: Middle segment: mean ± SEM of ΔHoyer under three contrasts.

C Supplementary Experimental Results

This appendix summarizes detailed statistical results based on real experimental data (from routereval_rq1_multimodel, rq2_multimodel, and rq3_multimodel).

C.1 Densification Results by Model and Layer Segment (RQ1)

Tables 1–3 report changes in Hoyer sparsity (ΔHoyer) for each model across layer segments. Values are reported as mean ± standard error of the mean (SEM). Negative values indicate densification (lower sparsity).

Interpretation note. For the contrast Tag–Instr, a *positive* value means Hoyer(Tag) > Hoyer(Instr), i.e., expert instructions induce stronger densification than tags.

C.2 Domain-Keyword Attention (RQ2)

C.2.1 RQ2 Statistical Details

For RQ2, we compute instance-level paired differences in the domain-keyword attention share, ΔAttn = Attn(condition) − Attn(control), under two query viewpoints (Prompt-last and First-gen). Tables 4–5 summarize these effects as mean ± standard error of the mean (SEM). When the text describes an effect as statistically significant, it is based on paired t-tests of ΔAttn against zero with Benjamini–Hochberg FDR control within each viewpoint/table family (test statistics and adjusted p-values are omitted from the tables for space).

To relate attention changes to output stability, we compute instance-level Pearson correlations between ΔAttn and ΔC3 metrics; Table 6 reports Pearson's r with two-sided p-values.

C.2.2 ΔAttn from the Prompt-last View

Table 4 reports paired differences in the domain-keyword attention share under the Prompt-last view (ΔAttn = condition − control). Negative values indicate a reduced attention share to domain keywords under the condition, whereas positive values indicate an increased share. Values are reported as mean ± standard error of the mean (SEM).

C.2.3 ΔAttn from the First-gen View

Table 5 reports paired differences in the domain-keyword attention share under the First-gen view (i.e., at the first generated token), defined in the same way as Table 4.

We observe that:

- For Llama and Qwen, ΔAttn is significantly negative under both views (reduced reliance on keywords), roughly consistent with "cognitive offloading."
- For Mistral, ΔAttn is significantly positive under both views (increased reliance on keywords; "attention reinforcement"), with a slightly larger magnitude under Instr than Tag.

C.2.4 Correlation Between ΔAttn and Delta Entropy

Table 6 reports the instance-level Pearson correlation coefficient between First-gen ΔAttn and ΔEntropy under the `instr_expert` vs. `control` contrast. Here, ΔAttn uses `attn_domain_share_firstgen_`

Model	ΔHoyer **Tag–Ctrl**	ΔHoyer **Instr–Ctrl**	ΔHoyer **Tag–Instr**
Llama-3.1-8B-Instruct	-0.008 ± 0.000	-0.012 ± 0.000	$+0.004 \pm 0.000$
Mistral-7B-Instruct-v0.2	-0.002 ± 0.000	-0.015 ± 0.000	$+0.013 \pm 0.000$
Qwen3-8B	-0.001 ± 0.000	-0.007 ± 0.001	$+0.006 \pm 0.000$

Table 3: Global segment: mean ± SEM of instance-level paired differences in ΔHoyer under three contrasts.

Model	ΔAttn **Tag–Ctrl (Mean ± SEM)**	ΔAttn **Instr–Ctrl (Mean ± SEM)**
Llama-3.1-8B-Instruct	-0.0389 ± 0.0031	-0.0371 ± 0.0030
Mistral-7B-Instruct-v0.2	$+0.0154 \pm 0.0005$	$+0.0164 \pm 0.0005$
Qwen3-8B	-0.0409 ± 0.0032	-0.0396 ± 0.0031

Table 4: Mean ± SEM of ΔAttn under Tag–Ctrl and Instr–Ctrl across three models (Prompt-last view).

`delta_instrexpert_vs_control`, and ΔEntropy uses `gen_mean_entropy_delta_instrexpert_vs_control`.

Brief interpretation:

- Qwen shows the highest correlation ($r \approx 0.23$), suggesting a relatively stronger linkage between keyword-attention changes and entropy changes in this model.
- Llama exhibits a moderate-to-weak positive correlation ($r \approx 0.12$).
- Mistral shows a significant negative correlation, consistent with its "attention reinforcement" pattern accompanied by limited gains in stability.

C.3 Correlations Between C1 and C3 (RQ3)

Notation.

- **C1**: `c1_prompt_hoyer_mean` (paired difference in Hoyer sparsity over the prompt segment).
- **C3-Entropy**: paired difference of `gen_mean_entropy`.
- **C3-SemVar**: paired difference of `semantic_variation_1mF1`.
- **Instr vs. Ctrl**: `instrexpert_vs_control`.
- **Tag vs. Ctrl**: `tagcorrect_vs_control`.
- **Tag vs. Instr**: `tagcorrect_vs_instrexpert`.
- **Significance markers** appended to correlation coefficients: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$, and ns for not significant.

C.3.1 RQ3 Statistical Details

We report Pearson correlations between paired differences ΔC1 (prompt Hoyer) and ΔC3 (entropy or semantic variation) for each model and prefix contrast. Two-sided p-values are computed per correlation; multiple comparisons are controlled with Benjamini–Hochberg correction within each table family. Full results are in Tables 7– 8.

C.3.2 Correlation Between C1 and C3-Entropy

Table 7 reports Pearson correlation coefficients between C1 (ΔHoyer) and C3-Entropy (ΔEntropy).

Rounding. Values correspond to `pearson_r` in the CSV; e.g., for Qwen3-8B under Instr vs. Ctrl, $r = 0.308859$ rounds to 0.309.

C.3.3 Correlation Between C1 and C3-SemVar

Table 8 reports Pearson correlation coefficients between C1 (ΔHoyer) and C3-SemVar (ΔVar, i.e., `semantic_variation_1mF1`).

D Reproducibility and Resource Consumption

D.1 Hardware and Runtime

- All experiments are conducted on multiple high-memory GPUs (8×A100). We disable gradient computation and only perform forward inference with activation/attention caching;

Model	ΔAttn **Tag–Ctrl (Mean ± SEM)**	ΔAttn **Instr–Ctrl (Mean ± SEM)**
Llama-3.1-8B-Instruct	-0.0592 ± 0.0045	-0.0599 ± 0.0046
Mistral-7B-Instruct-v0.2	$+0.0158 \pm 0.0004$	$+0.0175 \pm 0.0005$
Qwen3-8B	-0.0493 ± 0.0039	-0.0480 ± 0.0038

Table 5: Mean ± SEM of ΔAttn under Tag–Ctrl and Instr–Ctrl across three models (First-gen view). Negative values indicate reduced domain-keyword attention share relative to control.

Model	r(ΔAttn, ΔEntropy)	p-value
Llama-3.1-8B-Instruct	0.117	1.3×10^{-10}
Mistral-7B-Instruct-v0.2	-0.072	7.8×10^{-5}
Qwen3-8B	0.229	5.4×10^{-37}

Table 6: Pearson correlation between ΔAttn and ΔEntropy (instance-level), with corresponding p-values.

- For a single model, running MFV and extracting the metrics on the RouterEval subset used in this paper takes on the order of several hours, mainly depending on the model size and the number of stochastic generations (K);
- The analysis scripts (C1–C3 computation and statistical tests) are primarily bottlenecked by I/O and aggregation, and typically take less time than forward inference.

D.2 Randomness and Reproducibility

- All stochastic components (instance shuffling, incorrect-tag sampling, and decoding-time sampling) use fixed random seeds;
- For stochastic metrics such as semantic variation, we report the variance across repeated runs in the appendix;
- When supported by the framework, we enable deterministic options as much as possible to reduce the impact of numerical nondeterminism on the results.

D.3 Code and Data Release Plan

During the anonymous review phase, we do not disclose the repository URL. If the paper is accepted, we plan to:

- Release the RouterEval preprocessing scripts and the routing-style meta-prompt templates;
- Release the code for C1–C3 computation and statistical analysis, including MFV scripts for multi-GPU parallelization;
- Provide configuration files and command-line examples to facilitate reproducing experiments and figures;
- Provide, within licensing constraints, the indices or hashes of the RouterEval subset used, enabling other researchers to compare results on the same set of instances.

E Extended Ethics Discussion

When deploying a similar analysis framework in real-world systems, the following ethical and safety considerations should be taken into account:

1. Potential manipulation risks A finer-grained understanding of how routing signals relate to internal representations could be misused to more precisely steer model behavior or to circumvent safety mechanisms. Our work is intended for diagnosis and robustness analysis; we do not endorse any application of RIDE or related techniques for deceptive, surveillance-oriented, or otherwise harmful purposes.

2. Bias and unfair routing If "densification–stability" signals are directly used as routing cues or as proxies for uncertainty, they may exhibit systematic differences across languages, demographic groups, or topics, thereby amplifying biases present in the training data. Future work should evaluate the fairness properties of RIDE metrics on multilingual and multi-group datasets.

Model	r (Instr vs. Ctrl)	r (Tag vs. Ctrl)	r (Tag vs. Instr)
Llama-3.1-8B-Instruct	0.141^{***}	0.013^{ns}	0.105^{***}
Mistral-7B-Instruct-v0.2	0.191^{***}	0.043^{*}	0.150^{***}
Qwen3-8B	0.309^{***}	0.204^{***}	0.139^{***}

Table 7: Pearson correlations between C1 (ΔHoyer) and C3-Entropy (ΔEntropy) under three contrasts. Superscripts indicate significance: $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$, and ns not significant.

Model	Instr–Ctrl	Tag–Ctrl	Tag–Instr
Llama-3.1-8B-Instruct	0.067^{***}	-0.027^{ns}	0.037^{*}
Mistral-7B-Instruct-v0.2	0.142^{***}	0.128^{***}	0.177^{***}
Qwen3-8B	0.023^{ns}	0.072^{***}	0.017^{ns}

Table 8: Pearson r between C1 (ΔHoyer) and C3-SemVar (ΔVar). Superscripts: $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$, ns not significant.

3. Privacy and data usage This paper uses only public benchmarks and open-source models, without involving real user data. If future work extends the framework to real routing logs or production data, strict anonymization should be applied, and relevant privacy regulations and platform policies must be followed.

4. Cautious claims about "interpretability" We emphasize that RIDE provides a statistical lens for interpretation rather than a strict causal characterization of underlying mechanisms. In external communication or system documentation, it should not be presented as a "fully interpretable" solution; instead, its role as an analysis tool and auxiliary signal should be stated explicitly.