Task-Aware Spectral Pruning: A Mixture-of-Masks Framework for Efficient LLM Inference

Anonymous ACL submission

Abstract

Static pruning in large language models generally imposes a single sparse structure across all prompts, despite the fact that reasoning, retrieval, generation, coding, and translation tasks activate substantially different components of the network. We introduce **Task-Aware Spectral Pruning (TASP)**, a post-training pruning framework that constructs a compact family of task-routed, hardware-compliant sparse subnetworks using calibrated spectral sensitivity predictors. TASP first computes normalized spectral statistics and effective-rank measures for attention and feed-forward modules, and subsequently learns task-specific sensitivity predictors from empirically measured ablation-induced performance degradation. The framework then forms sparse masks under explicit active-FLOP constraints while preserving architectural validity for grouped-query attention and SwiGLU structures. During inference, each prompt is routed to a single compiled sparse mask that remains fixed throughout both the prefill and decoding stages. Experiments on Meta's Llama-3-70B demonstrate that, averaged across five random seeds, TASP reduces active decoding FLOPs by 43%, lowers single-stream decoding latency from 45.2 ± 0.4 ms to 23.5 ± 0.3 ms, and retains 97.7% of dense-model performance across the evaluated task suite. Furthermore, experiments on Alibaba Cloud's Qwen2.5-1.5B reveal that fixed heavy-tail thresholds are insufficient for distinguishing task-sensitive modules. This provides motivation for the need for model-calibrated spectral scoring as the central principle of the proposed framework.

1 Introduction

Modern large language models exhibit strong general-purpose sequence modeling capabilities across a wide spectrum of tasks, including mathematical reasoning, factual retrieval, code generation, machine translation, and open-ended text synthesis (Brown et al., 2020; Chowdhery et al., 2023). However, their substantial inference cost limits deployment in latency-sensitive and resource-constrained settings, motivating a broad line of research on efficient model compression without expensive retraining. Prior work on magnitude pruning, structured sparsification, and layer-wise compression, including SparseGPT, Wanda, SliceGPT, and related methods, has demonstrated significant redundancy in large language models (Han et al., 2015; Frantar and Alistarh, 2023; Sun et al., 2024; Ma et al., 2023; Ashkboos et al., 2024; Men et al., 2025; Xia et al., 2024; Gao et al., 2024; Cai et al., 2024, 2025). Nevertheless, most existing approaches produce either a single sparse model or a size-indexed family of subnetworks, rather than a compact set of task-specialized sparse structures adapted to heterogeneous prompt demands.

A single shared sparsity pattern can degrade task-specific performance; we refer to this as a *versatility tax*. A sparsity pattern optimized for arithmetic reasoning may degrade translation quality, whereas a mask preserving surface fluency may remove components critical for symbolic reasoning or code synthesis. Prior circuit-level analyses suggest that Transformer components exhibit functional specialization across attention heads and feed-forward modules (Elhage et al., 2021; Olsson et al., 2022), and pruning studies further indicate that model components differ substantially in compressibility and task sensitivity (Ma et al., 2023; Ashkboos et al., 2024; Lu et al., 2024). These observations motivate a central question: whether the intrinsic structure of a pretrained model is sufficiently informative to predict which components are relevant for different task families, and whether such structure can be leveraged to construct a small set of task-routed sparse subnetworks.

We address this question through a spectral

perspective on model sensitivity. Random Matrix Theory and Heavy-Tailed Self-Regularization (HT-SR) link the empirical spectral properties of weight matrices to implicit structure in deep networks (Martin and Mahoney, 2017, 2021), while recent pruning methods demonstrate that spectral descriptors can guide sparsity allocation. Building on these insights, we propose Task-Aware Spectral Pruning (TASP), which extends spectral analysis beyond layer-level aggregation to attention heads and feed-forward channel groups, and explicitly avoids relying on universal heavy-tail thresholds across models. Instead, TASP evaluates whether spectral signatures can separate task-sensitive from task-insensitive modules within a pretrained model and uses calibrated predictors to construct task-specific sparse masks under architectural and computational constraints.

Figure 1 illustrates the overall framework. In the offline stage, TASP computes spectral profiles for all candidate modules, fits task-specific sensitivity predictors using empirically measured ablation effects, and constructs structured sparse masks for each task family under active-FLOP constraints while respecting grouped-query attention and SwiGLU coupling. In the online stage, a lightweight routing mechanism selects a single compiled mask per input prompt, which remains fixed throughout the full prefill and decoding process to ensure cache consistency.

Our contributions include:

- Introduction of a calibrated spectral sensitivity model that predicts task-dependent module importance from weight-only spectral features and lightweight token-level salience signals.
- Formulation of task-mask construction as a constrained structured selection problem under explicit active-FLOP budgets with architectural validity constraints for grouped-query attention and SwiGLU blocks.
- Systematic disentanglement of the effects of spectral scoring, task-specific masking, routing, and calibration through a comprehensive ablation framework and mask similarity analysis.
- A separability diagnostic that identifies when spectral task routing is feasible, demonstrated through a positive case on Llama-3-70B and a negative case on Qwen2.5-1.5B.

Our findings reveal that larger models exhibit clearer spectral–task separability, enabling more reliable construction of task-specific sparse subnetworks, while smaller models show weaker and less consistent structure. These results highlight the importance of model-calibrated spectral analysis for effective task routing and improve our understanding of how spectral properties relate to functional specialization in large language models.

2 Related Work

Our work lies at the intersection of three primary research directions: post-training pruning of large language models, task-aware and calibration-driven sparsification, and spectral analysis of neural networks. We further connect to dynamic computation paradigms that enable conditional and input-dependent efficiency at inference time.

Unstructured Post-Training Pruning Post-training pruning of large language models aims to remove redundant parameters while preserving model performance. Early and widely used approaches rely on magnitude-based pruning, which removes weights with the smallest absolute values and serves as a strong and simple baseline (Han et al., 2015). More recent methods improve pruning quality by incorporating second-order information, as in SparseGPT, which performs approximate Hessian-based reconstruction to prune large models in a single pass (Frantar and Alistarh, 2023). Wanda further refines this idea by combining weight magnitude with activation statistics, enabling effective pruning without retraining (Sun et al., 2024). Movement pruning extends this direction by integrating sparsity into the dynamics of fine-tuning (Sanh et al., 2020), while recent work has also explored certificate-style guarantees for stochastic pruning (Shihab et al., 2026).

Although these approaches achieve strong sparsity levels, they typically produce unstructured weight masks that are not directly aligned with efficient execution on standard hardware, limiting practical speedups in many deployment settings.

Task-Aware and Calibration-Based Pruning A closely related direction studies task-aware pruning, where importance scores are derived from task-specific calibration data rather than generic language modeling objectives. This line

of work aims to preserve behavior relevant to a given downstream task while allowing significant parameter removal (Tian et al., 2025; Hong and Kim, 2026; Kodathala and Vunnam, 2026).

Our work is most closely related to this setting. However, unlike prior methods that typically produce a single task-specialized sparse model, our approach generates a small set of routed pruning masks based on calibrated spectral predictors. This allows different task behaviors to be represented within a shared pruned backbone rather than committing to a single static structure.

Further discussion of related work in structured pruning, spectral analysis, and dynamic computation paradigms is provided in Appendix A.

3 Method

To enable efficient deployment of TASP in resource-constrained settings, we propose a structured pruning framework that integrates spectral analysis, task calibration, and hardware-aware mask construction. Our approach identifies redundant components in large language models while preserving task-relevant behavior through a combination of spectral descriptors and calibrated sensitivity estimation. Figure 1 illustrates the overall framework.

3.1 Spectral Features

Let $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ denote a projection matrix or a structured slice corresponding to a single attention head or feed-forward channel group. Let $W = U\Sigma V^\top$ be its singular value decomposition, with singular values $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$. To enable comparison across modules of varying dimensionality, we consider the normalized Gram matrix $C_W = d_{\text{in}}^{-1} W^\top W$, whose eigenvalues are given by $\lambda_i = \sigma_i^2 / d_{\text{in}}$. We characterize its empirical spectral density $\rho_W(\lambda) = r^{-1} \sum_i \delta(\lambda - \lambda_i)$ using a compact set of spectral descriptors.

The most informative descriptor is the heavy-tail exponent. Given a tail threshold $\lambda_{\min}$, we model the upper tail of the spectrum using a Pareto distribution:

$$\Pr[\lambda > x \mid \lambda \geq \lambda_{\min}] \propto \left(\frac{x}{\lambda_{\min}} \right)^{-\alpha}.$$

The corresponding maximum-likelihood estimator is

$$\widehat{\alpha} = \left(\frac{1}{n} \sum_{i=1}^{n} \log(\lambda_i / \lambda_{\min}) \right)^{-1}, \qquad (1)$$

where n denotes the number of eigenvalues in the tail region and $\lambda_{\min}$ is selected via Kolmogorov–Smirnov minimization (Clauset et al., 2009).

We further compute the effective rank

$$r_{\text{eff}}(h) = \exp\left(-\sum_i \tilde{\lambda}_i \log \tilde{\lambda}_i \right), \quad \tilde{\lambda}_i = \frac{\lambda_i}{\sum_j \lambda_j}, \qquad (2)$$

which captures the concentration of spectral energy across directions. We additionally define $\text{SETOL}(h)$ as the normalized spectral tail mass above the threshold $\lambda_{\min}$, capturing the proportion of energy concentrated in the heavy-tailed region. The final feature vector for module h includes $\widehat{\alpha}_h$, $r_{\text{eff}}(h)$, spectral norm $\|W_h\|_2$, a tail-mass ratio $\text{SETOL}(h)$, normalized layer index ℓ_h, and a module-type indicator.

3.2 Calibrated Sensitivity Predictor

Fixed heavy-tail thresholds such as $\widehat{\alpha} < 2.5$ are known to be brittle across model families, and the Qwen2.5-1.5B diagnostic in Section 5.1 illustrates a concrete instance in which such universal criteria fail. Accordingly, TASP learns a task-specific predictor rather than relying on fixed spectral heuristics.Figure 16 illustrates the calibrated sensitivity estimation pipeline, where spectral descriptors and constraint-token salience are combined to predict task-specific module importance. For task family t and module h, let $\Delta_h^{(t)}$ denote the validation performance degradation induced by structurally ablating h while keeping all other parameters fixed. The calibrated sensitivity score is defined as a linear model over spectral features and a constraint-token salience term S_h:

$$\begin{aligned} s_h^{(t)} &= w_1^{(t)} \widehat{\alpha}_h + w_2^{(t)} r_{\text{eff}}(h) + w_3^{(t)} \|W_h\|_2 \\ &+ w_4^{(t)} \text{SETOL}(h) + w_5^{(t)} \ell_h + w_6^{(t)} S_h + b^{(t)}, \end{aligned} \qquad (3)$$

where $w^{(t)}$ are learned on a calibration set to predict $\Delta_h^{(t)}$. Although the model is trained under a regression objective, Section 5.1 additionally evaluates the ranking quality of $s_h^{(t)}$ with respect to the top quintile of $\Delta_h^{(t)}$, ensuring that the predictor is effective for selection beyond pure error minimization.

The salience term augments the spectral features with a lightweight prompt-conditioned signal. For an attention head (ℓ, h) with attention matrix $A_{\ell,h}(x)$ over calibration prompt $x \in D_c$, we

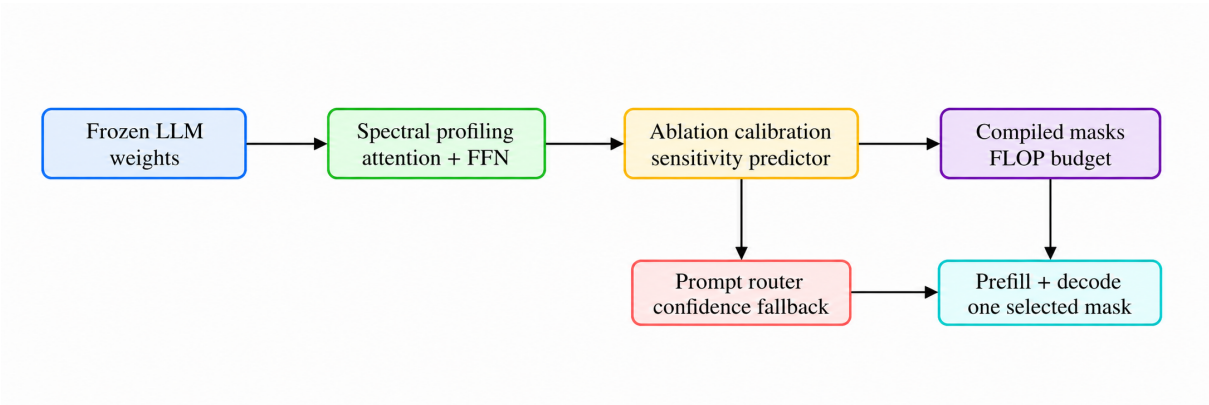

Figure 1: Detailed workflow of the TASP pipeline. The framework consists of (1) offline spectral profiling of model modules, where spectral statistics and effective-rank measures are computed, (2) offline calibration of task-specific sensitivity predictors using ablation-based performance drops, and (3) an inference-time routing mechanism that selects a single compiled sparse mask for the full prefill-and-decode pass. This design enables hardware-compliant sparse execution while preserving key–value cache consistency during generation.

define:

$$S_{\ell,h} = \mathbb{E}_{x\sim D_c}\left[\frac{1}{|Q(x)|}\sum_{q\in Q(x)}\max_{t\in C(x)} A_{\ell,h}(q,t)\right], \tag{4}$$

where $C(x)$ denotes tokens corresponding to numerical values, operators, logical connectives, quantifiers, and negations, and $Q(x)$ excludes padding and the fixed system prefix. We do not interpret $S_{\ell,h}$ as a mechanistic attribution signal; rather, it is treated as a calibration feature that improves prediction of $\Delta_h^{(t)}$ beyond spectral descriptors alone.

3.3 Structured Mask Construction

For each task family t, TASP constructs a structured mask $M_t^\star$ by solving the constrained optimization problem

$$\begin{aligned} M_t^\star &= \operatorname*{argmax}_{M\in\mathcal{C}} \sum_h M_h\, s_h^{(t)} \\ \text{s.t.} \quad & \mathrm{FLOPs}(M) \le B_t, \end{aligned} \tag{5}$$

where B_t denotes a task-specific active FLOP budget and $\mathcal{C}$ denotes the set of hardware-valid structured masks. In practice, this optimization is implemented via greedy selection: modules are ranked according to $s_h^{(t)}$ and iteratively selected until the computational budget is exhausted, subject to structural compatibility constraints.

The constraint set $\mathcal{C}$ encodes architectural dependencies required for valid compiled execution. In grouped-query attention, a key/value head is retained if any query head within the corresponding group is selected, and query heads are not reassigned across groups, thereby preserving key–value cache dimensional consistency. In SwiGLU feed-forward blocks, pruning is applied jointly across W_{gate}, W_{up}, and the corresponding columns of W_{down}, ensuring consistency of the gated transformation. The embedding layer, final normalization, and language modeling head are kept dense in all reported configurations.

Since theoretical sparsity does not necessarily translate directly into runtime gains, we separately report parameter sparsity, active FLOPs per token, decode FLOPs, and measured wall-clock latency. Figure 17 illustrates the structured mask construction procedure under FLOP and architectural constraints.

3.4 Routing and Inference-Time Execution

At inference time, TASP performs a single routing decision per user turn. The selected structured mask is applied consistently during both prefill and decoding stages. Figure 18 illustrates the full inference-time routing mechanism and cache-consistent sparse execution workflow in TASP. A key–value cache constructed under a given mask is not reused across different masks unless the active attention and channel dimensions are identical, ensuring cache compatibility.

The routing module is implemented as a three-layer MLP with hidden dimensions 512 and 256, operating over token-level lexical features and lightweight task descriptors. The model is trained on labeled task prompts. Temperature scaling is applied using a held-out validation set to improve calibration of routing probabilities.

We further adopt a confidence-based fallback mechanism: when the maximum routing probability falls below a threshold τ, the system switches to either a hybrid mask or the dense model. This design exposes an explicit accuracy–efficiency trade-off at inference time, enabling controlled deployment under varying resource constraints.

Algorithm 1 in Appendix B summarizes the full offline and online pipeline.

4 Experimental Setup

TASP is evaluated on Llama-3-8B, Qwen2.5-1.5B, and Llama-3-70B, with Llama-3-70B serving as the primary reference model. The evaluation suite spans five task families: reasoning (MMLU-Pro (Wang et al., 2024), GSM8K (Cobbe et al., 2021), MGSM (Shi et al., 2023)), open-ended generation (AlpacaEval (Taori et al., 2023), MT-Bench (Zheng et al., 2023)), retrieval (NQ-Open (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017)), code generation (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021)), and machine translation (WMT14 En–De and En–Fr (Bojar et al., 2014)).

To avoid aggregation across heterogeneous metrics, we report both raw scores and a relative retention measure:

$$\mathrm{Retention}(m) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \frac{\mathrm{score}_{m,b}}{\mathrm{score}_{\mathrm{dense},b}}. \quad (6)$$

We compare against baselines spanning the three categories outlined in Section 2: unstructured pruning methods (magnitude pruning, SparseGPT, Wanda), structured pruning approaches (LLM-Pruner, SliceGPT, ShortGPT), spectral or geometry-driven methods (AlphaPruning), elastic or conditional computation frameworks (LLaMaFlex), and agent-guided pruning where applicable.

We evaluate TASP in three configurations: a single global spectral mask (TASP-Single), the full mixture-of-masks with learned routing (TASP-MoM), and an oracle variant with access to ground-truth task labels (TASP-Oracle).

Unless otherwise stated, results are averaged over five random seeds. The seed controls calibration-prompt subsampling, ablation batch ordering, router initialization, tie-breaking in equal-score module selection, and evaluation instance ordering when supported by the benchmark. Dense model scores are deterministic under greedy decoding and are reused across runs for paired retention computation. Full seed-level results are reported in Appendix L.

Latency is measured on a single NVIDIA A100 80GB GPU using BF16 precision, FlashAttention-2.5, and KV caching. TASP subnetworks are compiled prior to inference. Each run uses batch size 1, a 2048-token prompt, and 256 generated tokens with greedy decoding. We perform 50 warmup runs followed by 200 measured runs, timing prefill and decoding separately.

As dense kernels do not exploit structured sparsity, we report dense masked executions only as controls and do not treat them as actual speedups. Additional implementation details and reproducibility settings are provided in Appendix F.

5 Results

We evaluate TASP across a diverse suite of models and task families, and organize the findings into four main questions: (i) whether spectral features are informative about task sensitivity in large language models, (ii) whether low-scoring modules identified by TASP indeed correspond to minimal task degradation under ablation, (iii) how the proposed mixture-of-masks framework positions itself on the accuracy–efficiency frontier relative to existing pruning and conditional computation methods, and (iv) to what extent the observed gains originate from routing and fallback mechanisms rather than the underlying structured masks.

5.1 Spectral Separability of Task-Relevant Modules

We first analyze whether the calibrated spectral score $s_h^{(t)}$ provides meaningful separation between task-sensitive and task-insensitive modules prior to mask construction. Let $\Delta_h^{(t)}$ denote the validation performance drop induced by ablating module h for task family t, and define binary labels for high-sensitivity modules as:

$$y_h^{(t)} = \mathbf{1}\{\Delta_h^{(t)} \geq \mathrm{quantile}_{0.8}(\{\Delta_j^{(t)}\}_j)\}. \quad (7)$$

The spectral separability of $s^{(t)}$ is then $\Delta_{\mathrm{spec}}^{(t)} = \mathrm{AUROC}(s_h^{(t)}, y_h^{(t)})$. An AUROC near 0.5 means the spectral score is uninformative for task sensitivity; a high AUROC means the model is an appropriate target for TASP.For the separability and causal-ablation diagnostics, we report four diagnostic task families—reasoning, retrieval, code,

and translation. Open-ended generation is excluded from these single-module ablation diagnostics because AlpacaEval and MT-Bench introduce higher evaluator variance under isolated module ablations, but generation remains included in the routing experiments and full-suite retention metric.

We use this diagnostic to compare two regimes. On Llama-3-70B the calibrated score is informative across all four diagnostic task families, with AUROC between 0.74 and 0.82, and the lowest 20% of calibrated tail exponents account for 63% of the measured reasoning ablation sensitivity, compared with 28% under a layer-matched random control. On Qwen2.5-1.5B the same diagnostic collapses to near chance (AUROC around 0.51–0.55 across families), which explains why uncalibrated heavy-tail thresholds misclassify modules in that model. TASP should therefore be used only when this diagnostic is above a deployment threshold selected on validation tasks; when it fails, the appropriate fallback is either the dense model or a non-spectral pruning method that uses activation information more directly. Figure 2 in Appendix C visualizes both regimes side by side.

5.2 Causal Validation of Module Importance

A complementary evaluation examines whether modules identified as least important by each pruning rule are indeed those whose removal induces minimal performance degradation. Pruning methods are compared by ablating the lowest-importance modules under identical sparsity budgets and layer-matched controls. Calibrated TASP scores consistently yield the smallest performance drops across all four diagnostic task families (e.g., an average -2.3 ± 0.2 points relative to dense, compared to -4.6 ± 0.2 for Wanda and -7.5 ± 0.3 for layer-matched random pruning), supporting the hypothesis that calibrated spectral features provide task-specific importance estimates rather than generic compressibility signals. The full per-task breakdown is reported in Table 3 in Appendix D.

5.3 Main Accuracy and Efficiency Results

Table 1 reports a representative subset of Llama-3-70B benchmark results under a 43% reduction in active decoding FLOPs. The Ret. column is computed over the full benchmark suite described in Section 5.3, including MT-Bench, TriviaQA, MBPP, and both WMT14 directions. Appendix M, Figure 5 visualizes relative retention across all compared pruning methods. The mixture-of-masks variant of TASP retains 97.7% of dense performance across the evaluation suite, compared to 93.3% for the best single global spectral mask and 94.0% for the strongest non-routed baseline; the oracle router further improves retention to 98.7%, reflecting the upper bound achievable under perfect routing decisions.Figure 15 further visualises task-level retention across evaluation benchmarks and pruning methods under matched sparsity. The same compiled subnetworks also yield substantial wall-clock improvements, with decode latency reduced from 45.2 ± 0.4 ms to 23.5 ± 0.3 ms

This trend persists across sparsity levels. Figure 3 in Appendix E traces the accuracy–FLOP frontier from 20% to 70% active-FLOP reduction, where TASP-MoM consistently outperforms the single global spectral mask over the 30–60% range. The performance gap between oracle and learned routing widens under higher sparsity, indicating that routing and fallback mechanisms become increasingly critical as sparsity becomes more aggressive.

5.4 Where the Gains Come From

The gains over TASP-Single presented in Table 1 can be attributed to three primary factors: improved module scoring, the use of multiple task-conditioned masks instead of a single global mask, and the routing and fallback mechanism applied on top of these masks. The ablation ladder in Table 2 isolates these contributions. Figure 6 in Appendix M presents the same ablation ladder under matched active-FLOP reduction, isolating the contributions of routing, salience calibration, spectral scoring, and mask diversity.Figure 7 further supports this finding by showing task degradation under module ablation at matched structured sparsity, where smaller degradation indicates more accurate sensitivity estimation. Replacing the learned router with an oracle yields an additional gain of approximately one point in retention over TASP-MoM, while removing the salience term results in a drop of roughly two points, and removing the spectral score entirely leads to a reduction of about four points. Collapsing the mixture-of-masks into a single global mask further decreases performance by approximately four and a half points. The most significant degradation, however, arises when replacing

Method	FLOP Red.	MMLU -Pro	GSM8K	MGSM	Alpaca Eval	NQ -Open	Human Eval	WMT14 avg.	Ret. full	Lat.
Dense	0%	79.2±0.0	87.5±0.0	74.3±0.0	82.1±0.0	68.4±0.0	72.6±0.0	28.3±0.0	100.0±0.0	45.2±0.4 ms
Magnitude	43%	64.1±0.5	58.3±0.7	52.7±0.6	67.8±0.5	59.2±0.4	43.1±0.8	21.4±0.3	75.6±0.5	–
SparseGPT	43%	72.3±0.4	75.4±0.6	66.8±0.4	74.2±0.4	63.1±0.3	58.7±0.6	24.8±0.2	88.5±0.4	–
Wanda	43%	73.8±0.3	77.1±0.5	68.2±0.4	75.6±0.4	64.3±0.3	61.2±0.5	25.6±0.2	90.7±0.3	–
Gradient	43%	74.2±0.3	78.3±0.4	69.1±0.4	76.1±0.3	64.8±0.3	62.3±0.5	26.1±0.2	91.7±0.3	–
AlphaPruning	43%	75.0±0.3	79.4±0.5	69.7±0.4	76.9±0.3	65.4±0.3	63.8±0.5	26.4±0.2	92.9±0.3	–
LLM-Pruner	43%	74.9±0.4	79.0±0.5	69.4±0.4	76.7±0.4	65.0±0.3	63.3±0.5	26.2±0.2	92.3±0.3	–
SliceGPT	43%	75.4±0.3	79.6±0.5	69.9±0.4	77.2±0.3	65.8±0.3	64.2±0.5	26.5±0.2	93.2±0.3	–
ShortGPT	43%	74.6±0.4	78.7±0.5	69.0±0.4	76.5±0.4	64.9±0.3	62.9±0.5	26.0±0.2	91.9±0.4	–
LLaMaFlex	43%	75.8±0.3	80.2±0.5	70.4±0.3	77.9±0.3	66.1±0.2	64.8±0.4	26.9±0.2	94.0±0.3	–
Agent-guided	43%	74.1±0.4	79.1±0.6	68.7±0.4	76.8±0.4	65.2±0.3	61.9±0.5	25.8±0.2	91.8±0.4	–
TASP single	43%	75.1±0.3	79.8±0.4	70.4±0.3	77.3±0.3	65.9±0.2	63.4±0.4	26.4±0.2	93.3±0.3	31.9±0.4 ms
TASP MoM	43%	**77.9±0.3**	**85.2±0.4**	**72.8±0.3**	**80.4±0.3**	**67.2±0.2**	**69.8±0.5**	**27.6±0.2**	**97.7±0.2**	**23.5±0.3** ms
TASP oracle	43%	78.4±0.2	86.1±0.3	73.5±0.3	81.3±0.2	67.8±0.2	70.9±0.4	28.1±0.2	98.7±0.2	23.5±0.3 ms

Table 1: Llama-3-70B results at matched 43% active-FLOP reduction (mean±std over five seeds). The displayed task columns are a representative subset of the full evaluation suite. Retention is the macro-average of each method's score divided by the dense score over the full benchmark suite: MMLU-Pro, GSM8K, MGSM, AlpacaEval, MT-Bench, NQ-Open, TriviaQA, HumanEval, MBPP, WMT14 En–De, and WMT14 En–Fr. Bold indicates the best non-oracle method; TASP oracle is reported only as an upper bound. Latency is reported for the dense model and compiled structured TASP subnetworks; non-compiled baselines report accuracy under the same active-FLOP target.

Variant	Retention
Dense	100.0±0.0
Five masks, oracle router	98.7±0.2
Five masks, learned router	**97.7±0.2**
without salience	95.5±0.3
without spectral score	93.6±0.4
One global spectral mask	93.3±0.3
Five masks, layer-matched random	85.6±0.5

Table 2: Ablation ladder for TASP at matched active-FLOP reduction (mean±std over five seeds). Each row removes or weakens one component while holding the sparsity target fixed. Bold indicates the best deployable sparse non-oracle variant; the oracle router is reported only as an upper bound.

TASP-selected modules with layer-matched random heads at identical sparsity, which results in a loss exceeding twelve points of retention. This serves as a near-null baseline and confirms that the spectral selection mechanism is a primary driver of the observed gains.

The mixture is also non-trivial: the five learned masks are not minor variants of one another. The pairwise Jaccard overlap is 0.78 ± 0.02 for reasoning and code, 0.81 ± 0.02 for generation and translation, but only 0.24 ± 0.03 for reasoning and generation; retrieval and translation overlap at 0.52 ± 0.03, and code and translation at 0.43 ± 0.03. Related capabilities share most of their structure, while distant ones share much less, which is consistent with the picture that different task families stress different parts of the network. The full overlap matrix appears in Appendix G.

5.5 Router Calibration and Fallback

Because the router is the only TASP component that varies online, its calibration directly controls the realized trade-off between accuracy and FLOPs. We sweep the confidence threshold τ that controls the fallback policy: a low threshold sends every prompt through a sparse mask and maximises latency savings, while a higher threshold uses the hybrid or dense fallback more often and recovers more accuracy. Moving from $\tau = 0.50$ to $\tau = 0.95$ takes the system from 97.7 ± 0.2 retention at $48 \pm 1\%$ latency reduction to 99.7 ± 0.1 retention at $24 \pm 1\%$ latency reduction, which we view as the most attractive operating point for accuracy-critical deployments. The full τ-sweep, together with the corresponding sparse-usage curve, appears in Appendix H. Appendix M, Figure 9 presents the full calibration curve with uncertainty across seeds.

5.6 realized Speedup, Robustness, and Transfer

Active-FLOP reductions are not automatically wall-clock improvements, and we measure both. The compiled TASP subnetworks reduce decode latency from 45.2 ± 0.4 ms for the dense baseline to 31.9 ± 0.4 ms for TASP single and 23.5 ± 0.3 ms for TASP MoM, corresponding to a $1.92\times$ single-stream decode speedup for TASP MoM over the dense model. An elementwise masked dense control gives less than 3% speedup at the same active-FLOP target, which confirms that the realized gains come from compiling each mask into a

smaller dense subnetwork rather than from sparsity in the dense kernels themselves. The full prefill-and-decode breakdown is in Appendix J.

A potential failure mode for any per-turn routing scheme is a conversation that repeatedly switches between capabilities. We test TASP on three such switches and compare per-turn routing against the simpler policy of routing only on the first turn. Per-turn routing closes most of the gap to the dense model while keeping the conversation in a sparse path on the majority of turns (for example, 74.1 ± 0.4 vs. 75.2 dense on a math-to-creative switch), whereas first-turn routing degrades steadily as the conversation drifts away from the initially chosen mask. The complete table appears in Appendix K.

We also test whether the calibration learned on a smaller model in the same family transfers to a larger one. Applying the Llama-3-8B calibrated ranking to corresponding regions of Llama-3-70B retains 92% of the performance of direct 70B calibration, indicating partial but incomplete transfer. A matched low-rank reconstruction control at the same FLOP budget is competitive with structured pruning for feed-forward matrices but consistently weaker for attention heads, which suggests that pruning and low-rank compression are complementary uses of the same spectral signal rather than substitutes. Appendix I reports both controls in full.

6 Discussion and Conclusion

The results show that Task Aware Spectral Pruning can convert calibrated spectral signals into practical task routed sparse subnetworks for frozen large language models, reducing computation while largely preserving dense model capability. TASP is effective when the spectral predictor separates task sensitive from task insensitive modules, the architecture supports valid structured masks under attention, MLP, and key value cache constraints, and the router selects the appropriate task mask or falls back under uncertainty. These conditions provide a practical diagnostic framework, since failures in separability, mask compatibility, or routing calibration can be observed before deployment without retraining.

The Qwen2.5 1.5B results clarify the limits of this approach. When fixed heavy tail thresholding collapses the spectral representation and assigns most heads to syntax dominated clusters, task separability falls to near chance levels. In that setting, routing and mask composition cannot recover useful specialization, so dense inference, conventional structured pruning, or calibration with stronger activation signals may be more appropriate.

The salience results suggest that masks retained for reasoning intensive tasks are more sensitive to numerical patterns, logical operators, quantifiers, and negation. This supports the view that TASP preserves reasoning relevant computation, but the evidence is correlational rather than mechanistic. We therefore describe the retained components as *reasoning sensitive spectral modules*, not as identifiable reasoning circuits in the sense of Elhage et al. (2021) or Olsson et al. (2022). Establishing causal function through intervention based methods such as activation patching or causal mediation remains important future work.

Overall, TASP replaces universal heavy tail thresholds with model specific separability diagnostics and enforces hardware compatible masks during construction, linking spectral structure to deployable sparse inference. On the reported Llama 3 70B configuration, it reduces active decoding FLOPs by 43% while retaining 97.7% relative performance and achieving a $1.92\times$ single stream decode speedup over the dense model. More broadly, the results suggest that routing among a small family of calibrated task masks can reduce the versatility cost of universal pruning, while the same spectral signal can indicate when the method is unlikely to be effective.

Limitations

TASP requires a calibration set and per-module ablation estimates. Although this cost is substantially lower than continued pretraining, it is not negligible and scales with the number of task families and the granularity of the spectral candidate set. The current formulation targets decoder-only Transformer LLMs with structured attention and feed-forward masks; extending the framework to state-space models, encoder–decoder architectures, or multimodal systems will require new module definitions and likely revised diagnostics. The reported efficiency gains additionally depend on compiling masks into smaller dense subnetworks, and elementwise masking without such compilation should not be interpreted as providing equivalent latency improvements. Finally, the di-

agnostic stage assumes that a representative set of calibration tasks adequately reflects deployment-time usage; if deployment traffic is dominated by unseen task families, the router may overconfidently select suboptimal masks.

Ethics Statement

TASP can reduce inference cost when structured masks translate into measurable hardware-level speedups, potentially expanding access to large language model deployment in resource-constrained settings where dense inference remains prohibitively expensive. Such efficiency gains may broaden the availability of advanced language technologies across low-resource institutions and edge-computing environments. However, pruning can unevenly affect model performance across languages, domains, and demographic groups, potentially amplifying disparities if deployment decisions rely solely on aggregate retention metrics. We therefore recommend task- and subgroup-specific evaluation, together with thorough fairness audits, before deployment in sensitive applications.

The router's confidence-based fallback mechanism provides an additional safeguard against misrouting, but it should be applied conservatively in safety-critical settings where incorrect mask selection could disproportionately degrade reliability or factuality. Deployment should evaluate performance across all intended target domains rather than assuming that calibration-time task coverage fully captures real-world usage. Although TASP improves computational efficiency at inference time, the calibration and pruning pipeline introduces additional offline compute costs. Future work should explore more adaptive calibration strategies, stronger robustness guarantees under distribution shift, and complementary efficiency techniques such as quantization to further reduce environmental impact while preserving equitable and reliable model behavior.

References

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*.

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*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024. Flextron: Many-in-one flexible large language model. In *International Conference on Machine Learning*. Oral; arXiv:2406.10260.

Ruisi Cai, Saurav Muralidharan, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2025. LLaMaFlex: Many-in-one LLMs via generalized pruning and weight sharing. In *International Conference on Learning Representations*.

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, Raul 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. *Preprint*, arXiv:2107.03374.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman.

2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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*.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024. DISP-LLM: Dimension-independent structural pruning for large language models. In *Advances in Neural Information Processing Systems*, volume 37. ArXiv:2410.11988.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, volume 28.

Jimyung Hong and Jaehyung Kim. 2026. Diet your LLM: Dimension-wise global pruning of LLMs via merging task-specific importance score. *arXiv preprint arXiv:2603.23985*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Sai Varun Kodathala and Rakesh Vunnam. 2026. LLMs can compress LLMs: Adaptive pruning by agents. *arXiv preprint arXiv:2601.09694*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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*.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. 2024. AlphaPruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 37. ArXiv:2410.10912.

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

Vladimir A. Marchenko and Leonid A. Pastur. 1967. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457–483.

Charles H. Martin and Michael W. Mahoney. 2017. Rethinking generalization requires revisiting old ideas: Statistical mechanics approaches and complex learning behavior. *arXiv preprint arXiv:1710.09553*.

Charles H. Martin and Michael W. Mahoney. 2021. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73.

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

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *Transformer Circuits Thread*.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389.

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

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

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won

Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*.

Ibne Farabi Shihab, Sanjeda Akter, and Anuj Sharma. 2026. Certificate-guided pruning for stochastic lipschitz optimization. *Preprint*, arXiv:2601.20231.

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*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Yuanhe Tian, Junjie Liu, Xican Yang, Haishan Ye, and Yan Song. 2025. Frustratingly easy task-aware pruning for large language models. *arXiv preprint arXiv:2510.22489*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37. Datasets and Benchmarks Track.

Eugene P. Wigner. 1958. On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, 67(2):325–327.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*.

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*, pages 2246–2251.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36. Datasets and Benchmarks Track.

A Extended Related Work

We provide additional discussion of related work spanning structured pruning, spectral analysis, and dynamic computation methods that are relevant to our approach.

Structured Pruning for Efficient Execution Structured pruning addresses this limitation by removing entire components of the model, thereby producing architectures that can be executed more efficiently as smaller dense networks. Representative methods include LLM-Pruner, which removes dependency-aware structures using gradient signals and optionally applies LoRA-based recovery (Ma et al., 2023). SliceGPT reduces model width by eliminating rows and columns in weight matrices, effectively shrinking the embedding dimension (Ashkboos et al., 2024). Sheared-LLaMA combines structured pruning with continued pretraining to recover performance after aggressive compression (Xia et al., 2024). DISP-LLM relaxes structural constraints across embedding dimensions to enable more flexible pruning decisions (Gao et al., 2024), while ShortGPT identifies and removes redundant transformer layers using a block-level importance measure (Men et al., 2025).

Spectral Analysis and Random Matrix Theory Another relevant thread analyzes neural network structure through the lens of Random Matrix Theory. Classical results on eigenvalue distributions provide a theoretical baseline for understanding high-dimensional random systems (Wigner, 1958; Marchenko and Pastur, 1967). Empirical studies have shown that trained neural networks systematically deviate from these classical predictions, exhibiting heavy-tailed spectral behavior that correlates with generalization and implicit regularization (Martin and Mahoney, 2017, 2021).

Building on this observation, methods such as AlphaPruning use spectral metrics inspired by heavy-tailed self-regularization theory to guide layerwise pruning decisions in large language models (Lu et al., 2024). In contrast, our approach leverages spectral information at a finer granularity and explicitly calibrates it against task sensitivity, enabling more localized and task-conditioned pruning decisions.

Dynamic and Conditional Computation A complementary line of work focuses on dynamic computation, where the effective computation path varies depending on input or efficiency constraints. Mixture-of-Experts models route tokens to specialized sub-networks, requiring architectures explicitly designed with expert modules (Shazeer et al., 2017; Lepikhin et al., 2021). More recent systems such as Flextron and LLaMaFlex

Algorithm 1 Task-Aware Spectral Pruning

Input: frozen model θ, task families $\mathcal{T}$, calibration set D_c, FLOP budgets $\{B_t\}_{t\in\mathcal{T}}$.
Slice attention and feed-forward modules into hardware-valid candidates.
for each candidate module h **do**
  Compute spectral features and salience S_h on D_c.
end for
for each task family $t \in \mathcal{T}$ **do**
  Estimate ablation sensitivities $\Delta_h^{(t)}$, fit predictor $s_h^{(t)}$, solve for $M_t^\star$ under budget B_t and constraints $\mathcal{C}$, and compile the resulting dense subnetwork.
end for
Train router $R(q)$ with temperature scaling and fallback threshold τ.
Return: compiled masks $\{M_t^\star\}$ and router R.

explore elastic inference, allowing models to adapt computation to different latency or accuracy budgets (Cai et al., 2024, 2025). Early-exit methods, including DeeBERT and CALM, reduce computation by terminating inference at intermediate layers when predictions are sufficiently confident (Xin et al., 2020; Schuster et al., 2022).

Our method differs from these approaches in that it operates on a frozen dense LLM without modifying the underlying architecture or inference procedure. Instead, it derives task-conditioned pruning masks through post-training spectral calibration, enabling conditional computation through structured sparsity rather than architectural modification.

B TASP Algorithm

Algorithm 1 summarizes the full offline-plus-online TASP procedure described in Section 3.4. Spectral profiling, ablation calibration, mask construction, and mask compilation are all offline; only the router fires at inference time, and its decision is committed for the entire user turn so that the key–value cache constructed under the selected mask remains valid throughout.

C Spectral Separability Diagnostic

Figure 2 visualises the AUROC of the calibrated spectral score $s_h^{(t)}$ as a task-sensitivity classifier on Llama-3-70B and on Qwen2.5-1.5B, the two regimes discussed in Section 5.1. On Llama-3-70B the bars cluster well above the 0.5 chance line for every task family, while on Qwen2.5-1.5B the same bars sit on or just above chance, which is the visual signature of the negative regime in which TASP should not be applied.

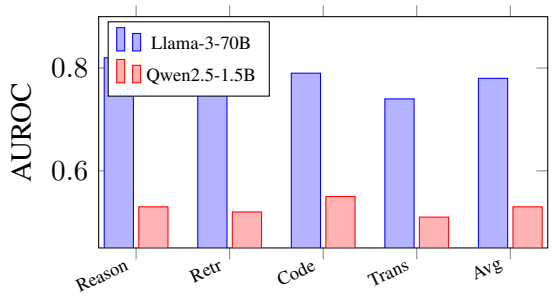

Figure 2: Spectral separability diagnostic. Llama-3-70B shows a useful task-sensitivity ranking signal across all task families, while Qwen2.5-1.5B is close to chance under the same diagnostic.

D Causal Validation Table

Table 3 reports the full per-task degradation from ablating the lowest-importance modules under each pruning rule at matched structured sparsity, with layer-matched controls. The summary numbers appear in Section 5.2 of the main body; the appendix reports each task family separately. Smaller magnitudes are better, since the table measures degradation relative to dense.

E Accuracy–FLOP Pareto Frontier

Figure 3 extends the matched-FLOP comparison of Section 5.3 across a range of active-FLOP reductions. TASP MoM is at the upper envelope of the curve over the middle of the sparsity range, where structured pruning is most useful in practice, and the gap between TASP MoM and the oracle TASP variant widens at very aggressive sparsity, where routing decisions and fallback policies matter more.

F Evaluation Protocol

Table 4 consolidates the implementation choices behind the reported efficiency and accuracy claims. The protocol is designed so that the numbers in Tables 1, 2, 6, 7, and 8 can be reproduced from the listed software stack and hardware.

G Mask-Overlap Matrix

Table 5 reports selected pairwise Jaccard overlaps between the five learned masks discussed in Sec-

Rule	Reas.	Retr.	Code	Trans.	Avg
Random	-8.7 ± 0.5	-6.2 ± 0.4	-9.4 ± 0.6	-5.8 ± 0.4	-7.5 ± 0.3
Magnitude	-6.9 ± 0.4	-5.4 ± 0.3	-7.8 ± 0.5	-4.9 ± 0.3	-6.3 ± 0.3
Act. norm	-5.8 ± 0.3	-4.7 ± 0.3	-6.4 ± 0.4	-4.2 ± 0.3	-5.3 ± 0.2
Wanda	-4.9 ± 0.3	-4.1 ± 0.2	-5.7 ± 0.4	-3.8 ± 0.3	-4.6 ± 0.2
TASP	$\mathbf{-2.1\pm0.2}$	$\mathbf{-2.4\pm0.2}$	$\mathbf{-2.8\pm0.3}$	$\mathbf{-1.9\pm0.2}$	$\mathbf{-2.3\pm0.2}$

Table 3: Task degradation from ablating the modules each rule ranks lowest at matched structured sparsity, measured in points relative to the dense model (mean±std over five seeds). Smaller magnitudes are better.

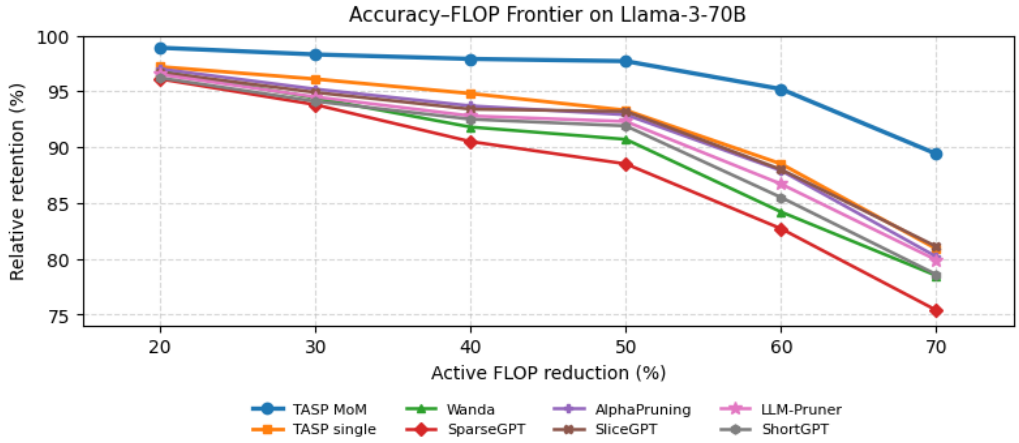

Figure 3: Accuracy–FLOP frontier on Llama-3-70B. Relative retention is plotted against active decoding FLOP reduction for TASP and representative pruning baselines. TASP MoM consistently achieves the strongest retention across sparsity levels, approaching dense model performance while maintaining substantial computational savings.

tion 5.4. Mask names follow the five task families used throughout the evaluation: reasoning, generation, retrieval, code, and translation. High-overlap pairs correspond to task families with related computations, while low-overlap pairs indicate that the corresponding masks rely on substantially different modules. A complete pairwise overlap visualization is shown in Figure 8.

H Router Calibration Curve

Table 6 reports the τ-sweep that underlies the realized accuracy–efficiency trade-off in Section 5.5, and Figure 4 in Appendix M visualises the same entries as continuous curves. The retention curve is roughly monotone in τ, while sparse usage and latency reduction trade off against it; this is the shape of the operating curve we expose to the deployer.

I Cross-Model Transfer and Low-Rank Controls

To test cross-model transfer we computed the calibrated spectral ranking on Llama-3-8B and applied it to corresponding layer regions of Llama-3-70B. The transferred mask retained 92% of the performance of direct 70B calibration at the same

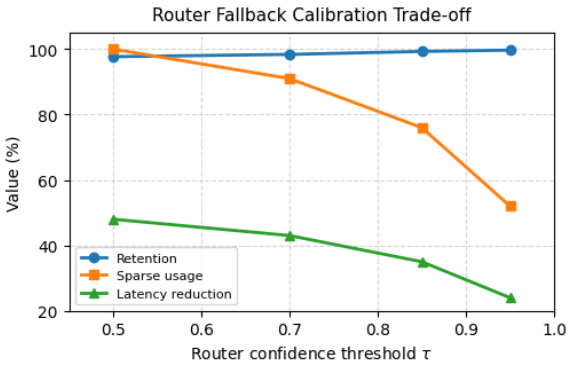

Figure 4: Router fallback calibration trade-off as the confidence threshold τ increases. The threshold τ determines the operating point on the accuracy–efficiency trade-off curve.

43% active-FLOP reduction, with the largest gaps appearing on GSM8K and HumanEval, where reasoning-sensitive head identities differ more between the 8B and 70B configurations.

The low-rank control replaces structured pruning of each candidate module by a rank-k approximation at the same active-FLOP budget. Pruning was the stronger compression mode for attention heads (a 1.9-point retention gap on GSM8K and a 1.4-point gap on HumanEval), while low-rank ap-

Item	Specification
Models	**Llama-3-8B**, **Qwen2.5-1.5B**, **Llama-3-70B**
Task families	reasoning, generation, retrieval, code, translation
Calibration target	per-module validation drop after structured ablation on held-out calibration prompts
Repeated runs	five seeds; seed affects calibration subset, ablation order, router initialisation, and mask tie-breaking
Mask budget	active FLOPs/token; parameter sparsity, decode FLOPs, and latency reported separately
Routing scope	one decision per user turn; same mask for prefill and decode; cache not reused across incompatible masks
GPU	single NVIDIA A100 80GB; BF16 weights; KV cache enabled
Software	PyTorch 2.4, CUDA 12.1, FlashAttention 2.5, Transformers 4.44, Triton 3.0; compiled dense subnetworks for structured masks
Latency prompt and decode	batch size 1; 2048-token prefill; 256 generated tokens; greedy decoding; CUDA graphs disabled for comparability
Warmup and measurement	50 warmup generations; 200 measured generations; p50 and mean reported; router time measured in the same process
Dense-masked control	elementwise masks with dense kernels, used only as a control and not counted as realized speedup
Aggregation	raw metrics plus relative retention against dense score

Table 4: Evaluation protocol and implementation constraints used to make the reported efficiency claims auditable.

Mask pair	Jaccard overlap
Reasoning / Code	0.78±0.02
Generation / Translation	0.81±0.02
Reasoning / Generation	0.24±0.03
Retrieval / Translation	0.52±0.03
Code / Translation	0.43±0.03

Table 5: Selected pairwise Jaccard overlaps between learned task masks (mean±std over five seeds). Mask names follow the five task families used in the evaluation: reasoning, generation, retrieval, code, and translation.

τ	Retention	Sparse usage	Latency red.
0.50	97.7±0.2	100±0%	48±1%
0.70	98.4±0.2	91±2%	43±1%
0.85	99.3±0.1	76±3%	35±1%
0.95	99.7±0.1	52±3%	24±1%

Table 6: Effect of the router confidence threshold τ. Higher thresholds recover more retention at the cost of less sparse execution and smaller latency wins (mean±std over five seeds).

proximation was competitive for feed-forward matrices (within 0.3 points of structured pruning on AlpacaEval and within 0.5 points on NQ-Open). Together with the main results, this suggests that the same spectral signal that drives task-aware pruning can also be repurposed to guide low-rank compression, and that the two compression modes are complementary rather than substitutes.

J Latency Breakdown

Table 7 reports the full latency breakdown for the configurations discussed in Section 5.6, including separate prefill and decode timings and the elementwise-masked dense control. The control confirms that mask-as-zero in a dense kernel does not realise the active-FLOP reduction; compilation into a smaller dense subnetwork does.Figure 10 supports the latency breakdown in Table 7, with only compiled structured subnetworks converting theoretical FLOP reductions into real wall-clock speedups.

K Task-Switching Stress Test

Table 8 reports the full task-switching results summarised in Section 5.6. Figure 11 evaluates robustness under rapid task switching. Each scenario alternates between two task families across eight turns; first-turn routing fixes the mask after the first user message, while per-turn routing reroutes on every turn. The dense column reports the corresponding oracle for that conversation under the unpruned model. Each scenario alternates between two task families across eight turns; first-turn routing fixes the mask after the first user message, while per-turn routing reroutes on every turn. The dense column reports the corresponding oracle for that conversation under the unpruned model.

L Seed-Level Tables

Tables 9 and 10 report the seed-level values that underlie the means and standard deviations in Tables 1 and 2. Figures 12 and 13 visualise the corresponding seed-level retention and ablation statistics. Table 11 reports the per-task seed values for the TASP MoM row of Table 1 while Figure 14

Configuration	Decode FLOPs	Prefill	Decode	Decode speedup
Dense	1.00×	181.4±1.8	45.2±0.4	1.00×
Dense + mask	0.57× act.	180.6±1.9	43.8±0.5	1.03×
TASP single	0.57×	134.2±1.6	31.9±0.4	1.42×
TASP MoM	0.57×	101.8±1.4	23.5±0.3	1.92×

Table 7: realized vs. theoretical compute reduction (latency in ms). Elementwise masks alone do not yield the active-FLOP reduction; compiled structured subnetworks do.

Scenario	Dense	First-turn	Per-turn
Math → creative	75.2±0.0	71.8±0.4	74.1±0.4
Code → translation	68.4±0.0	64.9±0.5	67.1±0.5
Retrieval → reasoning	72.8±0.0	69.6±0.4	71.9±0.3
Multi-task average	72.1±0.0	68.8±0.4	71.0±0.4

Table 8: Task-switching stress test (mean±std over five seeds). Per-turn routing closes most of the gap to dense, while locking the conversation to the first selected mask degrades.

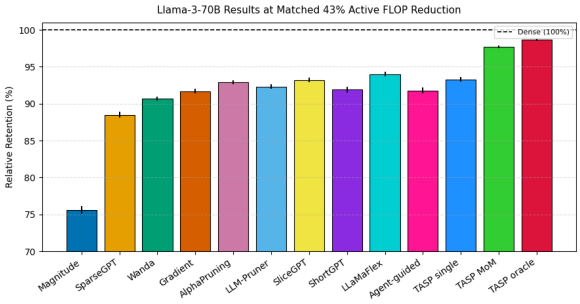

Figure 5: Relative retention on Llama-3-70B at matched 43% active FLOP reduction. Higher values indicate closer agreement with dense model performance. TASP MoM substantially outperforms prior pruning baselines, achieving 97.7% retention and approaching the oracle upper bound (98.7%), while standard one-mask and heuristic pruning methods exhibit larger performance degradation under the same computational budget. Error bars denote standard deviation over repeated runs.

illustrates the corresponding per-task variability across random seeds.

M Supplementary Figures

This section presents supplementary visual analyses for TASP, covering retention, ablations, mask overlap, router calibration, compute efficiency, task-switching behavior, seed variability, and sparse inference execution. These figures provide additional evidence that the proposed task-aware sparse routing framework preserves dense-model performance while translating structured sparsity into practical efficiency gains.

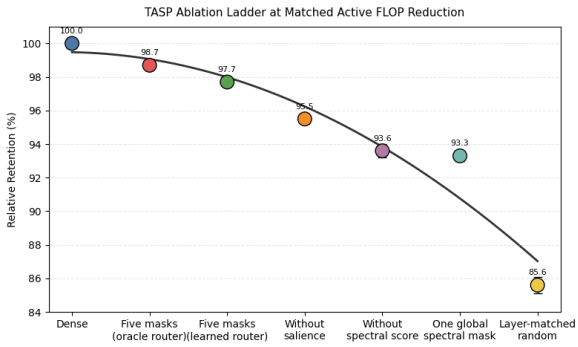

Figure 6: Ablation ladder for TASP under matched active-FLOP reduction. The figure isolates the contribution of routing, salience calibration, spectral scoring, and mixture-of-masks construction to overall retention. TASP with five task-specific masks and a learned router achieves 97.7% retention, while removing salience features, spectral calibration, or mask diversity progressively reduces performance. Replacing TASP-selected modules with layer-matched random heads causes the largest degradation, confirming that calibrated spectral selection is the primary driver of performance preservation. Error bars denote standard deviation across five seeds.

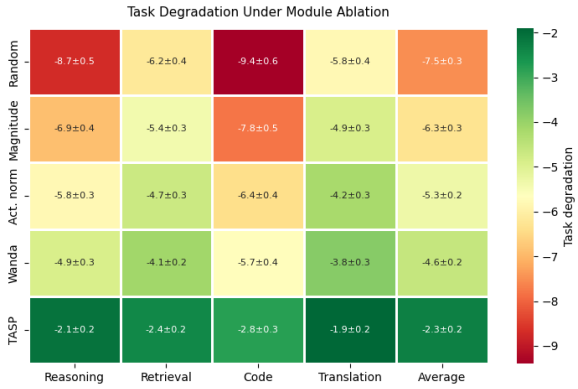

Figure 7: Task degradation under module ablation at matched structured sparsity. Each cell reports the performance drop (mean±std over five seeds) when ablating the modules ranked least important by each pruning rule, measured relative to the dense model. Smaller degradation indicates more accurate sensitivity estimation. TASP consistently identifies less task-critical modules, yielding the smallest degradation across all task families.

Method	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Mean±Std
Magnitude	75.0	75.5	76.2	75.8	75.6	75.6±0.5
SparseGPT	88.0	88.4	89.0	88.7	88.5	88.5±0.4
Wanda	90.3	90.5	91.0	90.9	90.8	90.7±0.3
Gradient	91.3	91.6	92.0	91.8	91.7	91.7±0.3
AlphaPruning	92.5	92.8	93.3	93.0	92.9	92.9±0.3
LLM-Pruner	91.9	92.1	92.7	92.4	92.3	92.3±0.3
SliceGPT	92.8	93.0	93.6	93.3	93.2	93.2±0.3
ShortGPT	91.5	91.7	92.4	92.0	91.9	91.9±0.4
LLaMaFlex	93.6	93.8	94.4	94.1	94.0	94.0±0.3
Agent-guided	91.3	91.6	92.3	92.0	91.8	91.8±0.4
TASP single	92.9	93.1	93.7	93.5	93.3	93.3±0.3
TASP MoM	97.5	97.6	98.0	97.8	97.7	97.7±0.2
TASP oracle	98.5	98.6	99.0	98.8	98.7	98.7±0.2

Table 9: Seed-level relative retention values for the main Llama-3-70B comparison.

Variant	S1	S2	S3	S4	S5	Mean±Std
Oracle router	98.5	98.6	99.0	98.8	98.7	98.7±0.2
Learned router	97.5	97.6	98.0	97.8	97.7	97.7±0.2
No salience	95.2	95.4	95.9	95.6	95.5	95.5±0.3
No spectral	93.0	93.4	94.1	93.8	93.7	93.6±0.4
Global mask	92.9	93.1	93.7	93.5	93.3	93.3±0.3
Random heads	85.0	85.4	86.2	86.0	85.5	85.6±0.5

Table 10: Seed-level ablation ladder corresponding to Table 2.

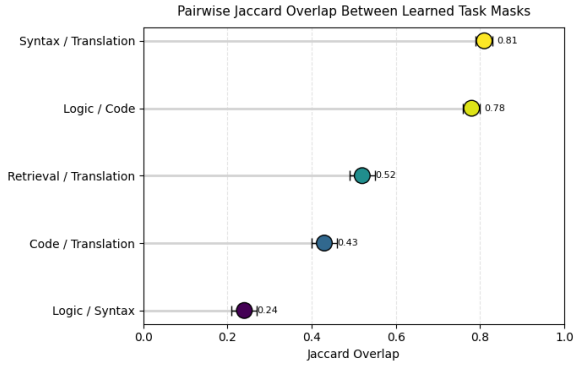

Figure 8: Pairwise Jaccard overlap between learned task masks (mean±std over five seeds). Higher overlap indicates greater shared structure between task-specific subnetworks, while lower overlap suggests stronger task specialization. Related task pairs such as Reasoning/Code and Generation/Translation exhibit substantial overlap, whereas more distinct task families such as Reasoning/Generation show markedly lower agreement, supporting the need for task-aware sparse routing.

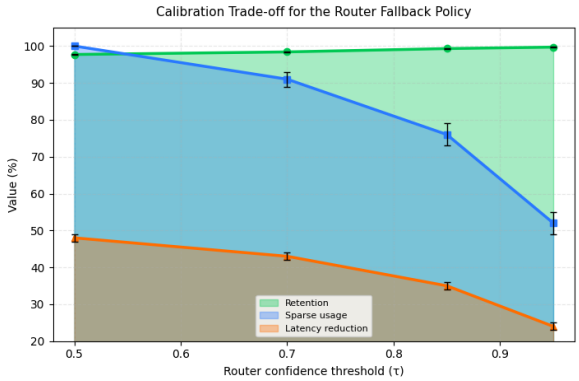

Figure 9: Calibration trade-off for the router fallback policy as the confidence threshold τ varies. Increasing τ improves dense model retention by routing more uncertain prompts away from sparse execution, but reduces sparse-mask usage and correspondingly lowers latency reduction. The selected operating point provides a balanced trade-off between reliability and efficiency. Shaded regions and error bars denote variability across repeated runs.

Metric	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Mean±Std
MMLU-Pro	77.5	77.8	78.3	78.0	77.9	77.9±0.3
GSM8K	84.7	85.0	85.7	85.4	85.2	85.2±0.4
MGSM	72.4	72.6	73.1	72.9	72.8	72.8±0.3
AlpacaEval	80.0	80.2	80.8	80.5	80.4	80.4±0.3
NQ-Open	67.0	67.1	67.5	67.3	67.2	67.2±0.2
HumanEval	69.1	69.5	70.5	70.0	69.9	69.8±0.5
WMT	27.3	27.5	27.9	27.7	27.6	27.6±0.2

Table 11: Per-task seed values for the TASP MoM row in Table 1.

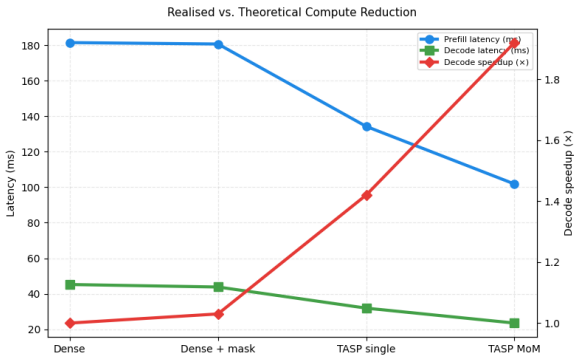

Figure 10: Realized versus theoretical compute reduction across deployment configurations. Applying an uncompiled mask to the dense model yields negligible latency improvement despite reduced active computation, whereas compiling structured sparse subnetworks translates theoretical FLOP savings into substantial practical speedups. TASP MoM achieves the largest gains, reducing prefill and decode latency while delivering a 1.92× decode speedup over the dense baseline.

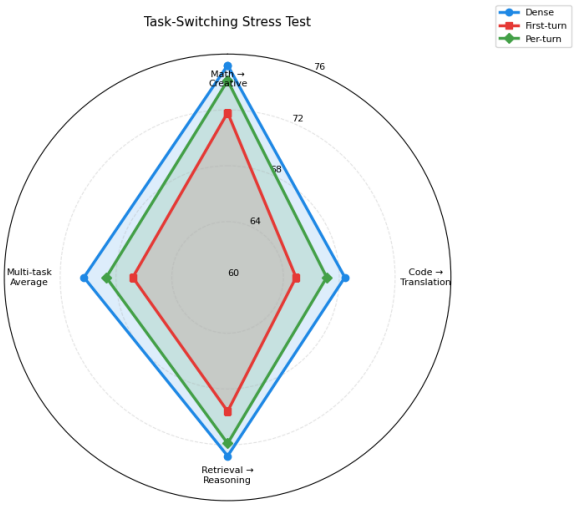

Figure 11: Task-switching stress test across representative domain transitions. Performance is compared between the dense model, first-turn routing (one mask selected at prompt start), and per-turn routing (mask reselected after each task switch). Per-turn routing substantially reduces degradation under shifting task demands and more closely tracks dense model performance, demonstrating the importance of adaptive task-aware mask selection in multi-domain conversational settings.

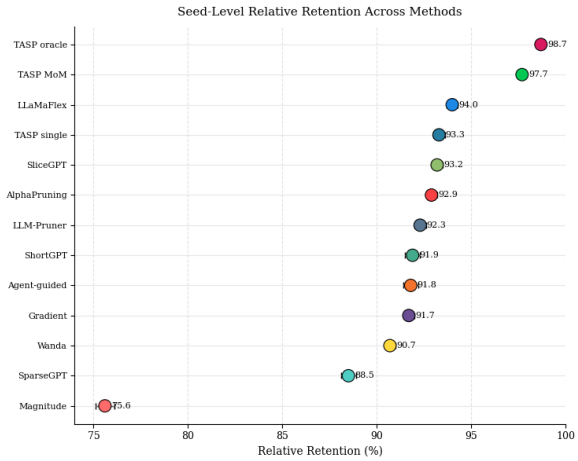

Figure 12: Seed-level relative retention across pruning methods at matched active FLOP reduction. Points indicate mean retention and error bars denote standard deviation over repeated runs. TASP MoM achieves the highest retention among practical methods and approaches the oracle upper bound, while also exhibiting lower variance, indicating more stable performance across random seeds.

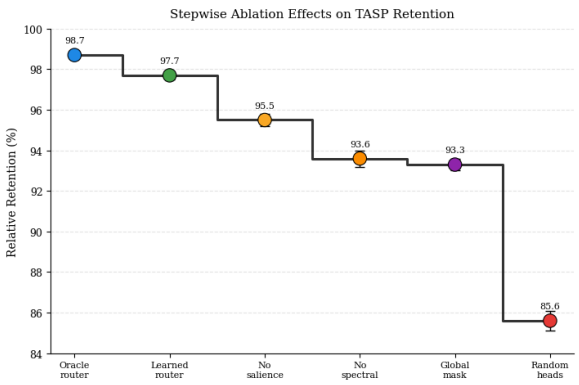

Figure 13: Stepwise ablation effects on TASP retention. Starting from the oracle routing upper bound, progressively removing key components—learned routing quality, salience features, spectral descriptors, and task-specific masking—leads to systematic degradation in relative retention. The results highlight the complementary contributions of spectral sensitivity estimation and task-aware routing to overall performance. Error bars denote standard deviation over repeated runs.

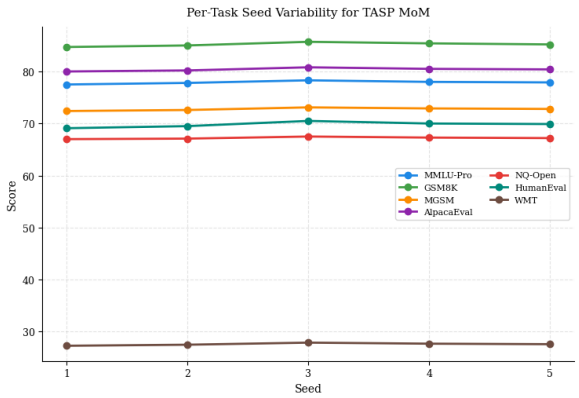

Figure 14: Per-task seed variability for TASP MoM across representative evaluation benchmarks. Performance remains highly consistent across random seeds, with only minor fluctuations across reasoning, coding, retrieval, and translation tasks. The low variance supports the robustness and reproducibility of the proposed task-aware sparse routing framework.

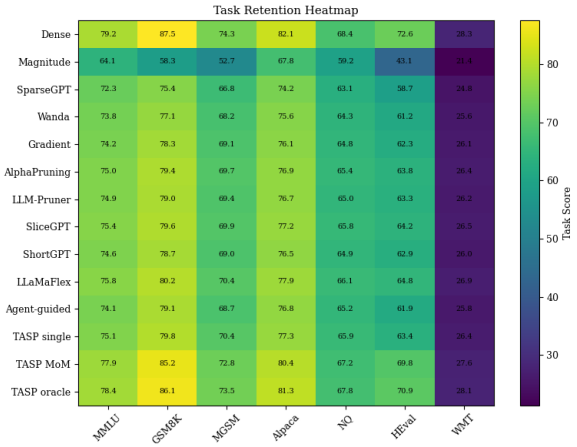

Figure 15: Task retention heatmap across evaluation benchmarks and pruning methods. Each cell reports task-level performance under matched sparsity, illustrating how different pruning strategies affect reasoning, coding, retrieval, and translation capabilities. TASP MoM consistently achieves stronger retention across tasks and more closely matches dense model behavior than competing pruning baselines, while the oracle setting provides an upper-bound reference.

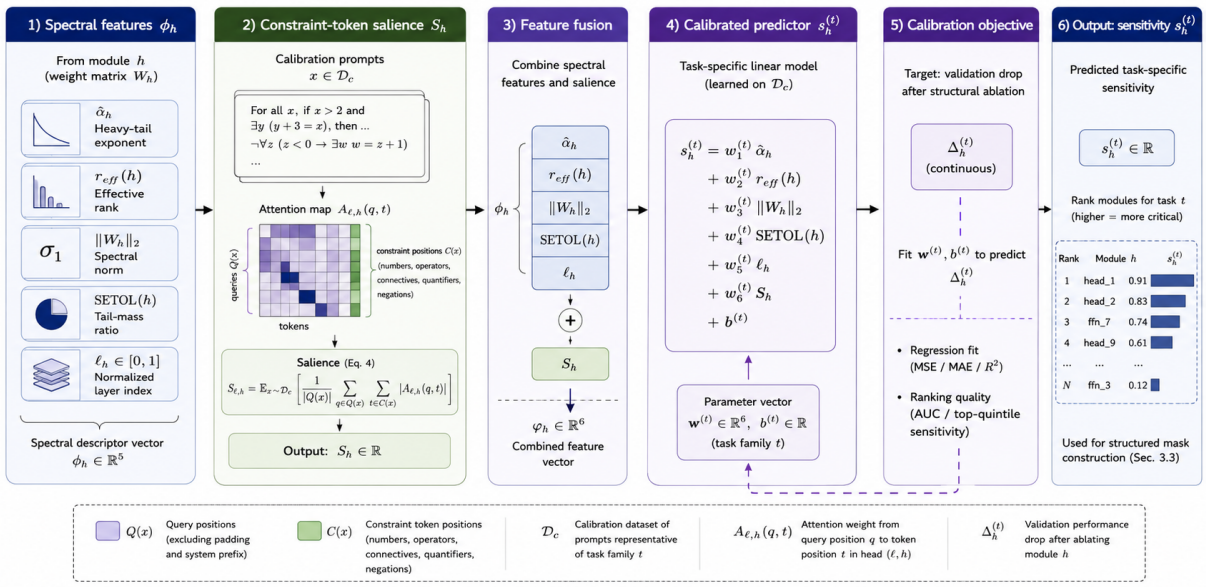

Figure 16: Calibrated sensitivity estimation in TASP. Module-level spectral descriptors are combined with a lightweight constraint-token salience signal to predict task-specific sensitivity. A linear calibration model is fit to validation drops induced by structural ablations, producing sensitivity scores that rank modules by expected task relevance and guide subsequent structured mask construction.

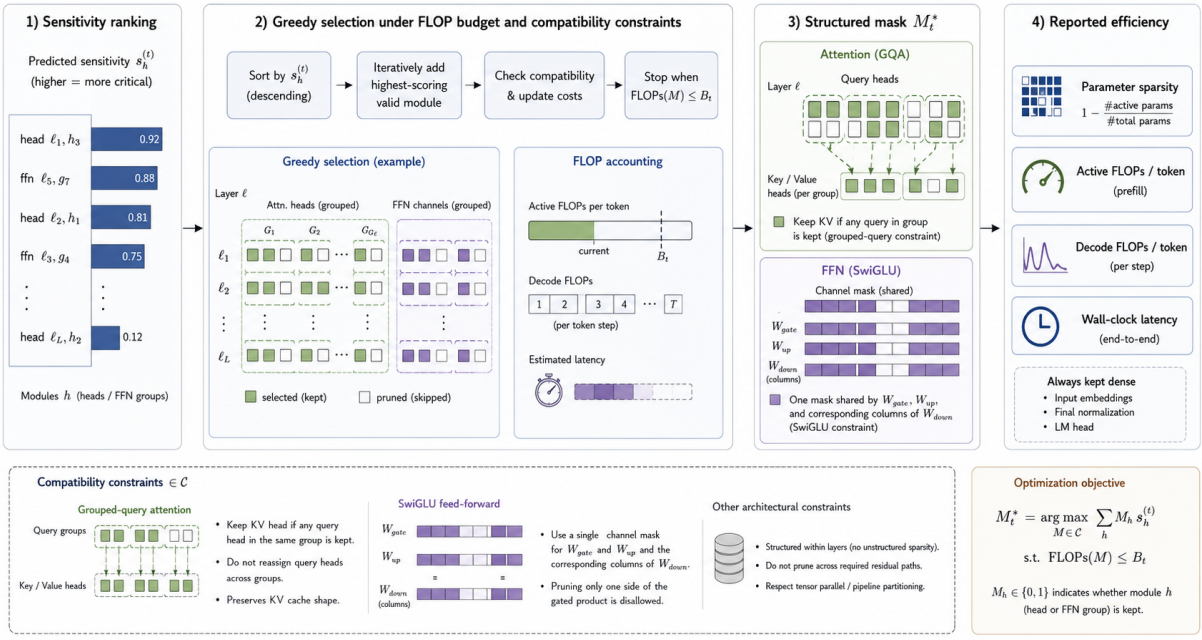

Figure 17: Structured mask construction under FLOP and architectural constraints. Modules are first ranked by predicted task-specific sensitivity and then greedily selected under a fixed active-FLOP budget to maximize retained utility. Hardware-validity constraints enforce architectural compatibility, including grouped-query attention coupling and shared channel masks across SwiGLU feed-forward components, producing compiled sparse subnetworks that preserve execution correctness and enable efficient deployment. Parameter sparsity, active FLOPs, decode cost, and end-to-end latency are reported separately to capture practical efficiency gains.

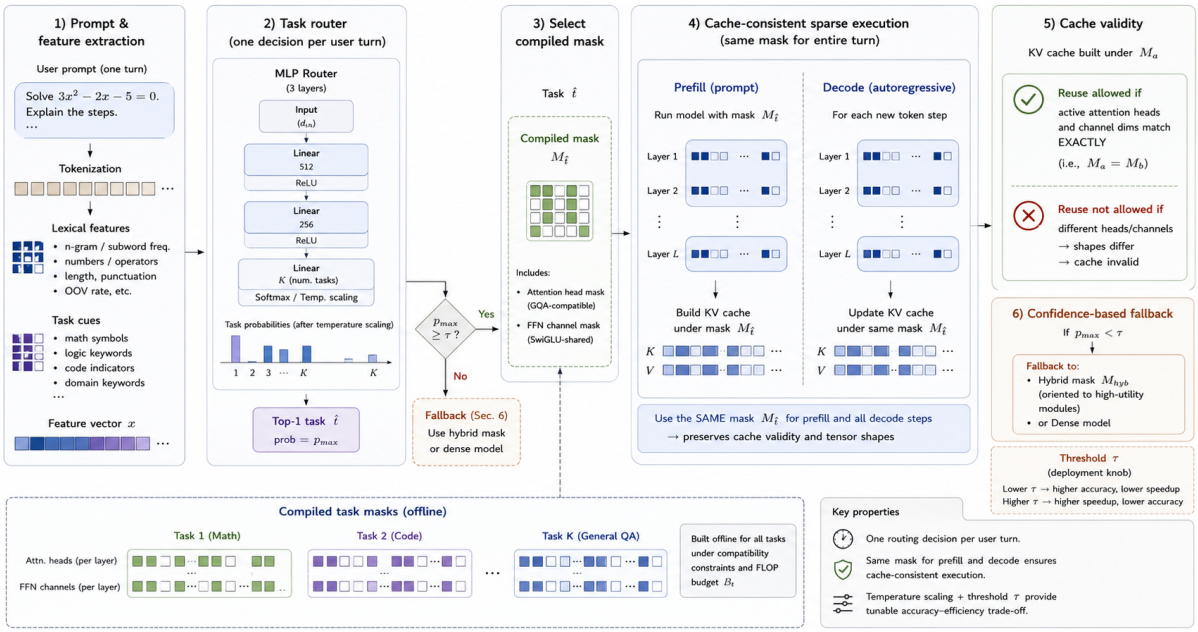

Figure 18: Inference-time routing and cache-consistent sparse execution in TASP. For each user prompt, a lightweight router predicts the task family and selects a single compiled structured mask, which is applied consistently throughout both prefill and autoregressive decoding to preserve key–value cache validity. Cache reuse is permitted only when active attention and channel dimensions match exactly across execution steps. Router confidence is calibrated via temperature scaling, and low-confidence predictions trigger fallback to a hybrid mask or the dense model, exposing a tunable accuracy–efficiency trade-off through the confidence threshold τ.