TRUST-REGION ON-POLICY SPARSE OPTIMIZATION FOR PRUNING REASONING LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning language models incur high inference costs because they generate long chains of thought before producing an answer, making model compression techniques such as pruning increasingly important. However, current pruning approaches typically cause severe performance degradation at high sparsity, leaving effective reasoning model pruning an open problem. In this work, we identify a key challenge in pruning reasoning models: pruning alters the prefixes—the prompts and partially generated sequences—encountered during generation, causing the fixed, off-policy calibration data to become increasingly mismatched with the evolving sparse model. To tackle this challenge, we introduce SCOUT, a novel sparse optimization framework that alleviates the distributional mismatch by leveraging on-policy data from the evolving sparse model, effectively curbing reasoning performance drop. At its core, SCOUT dynamically refines the sparsity mask through iterative pruning and reviving, while constraining each mask update within a KL divergence-based trust region to bound policy drift. We further establish a stationarity bound for an idealized form of this alternating procedure, where the trust-region budget and rollout staleness govern the resulting convergence neighborhood. Empirically, we show that SCOUT substantially outperforms existing pruning methods at high sparsity across Qwen3 models of varying scales.

1 INTRODUCTION

Recent large language models (LLMs) are increasingly being developed into reasoning models that generate long intermediate steps before producing a final answer, particularly for complex tasks such as mathematics and coding (Guo et al., 2025; Yang et al., 2025). While this paradigm has enabled a massive leap in solving complex reasoning tasks, it has concurrently triggered a sharp surge in inference expenses, as it routinely generates lengthy reasoning chains of tokens per query (Muennighoff et al., 2025). This work attends to compressing reasoning models, particularly by network pruning (Han et al., 2015), which can offer an effective way to alleviate such inference overheads; by selectively removing a fraction of parameters, pruning minimizes both storage demands and memory access costs during generation, thereby alleviating the computational overhead of resource-intensive reasoning models (Cheng et al., 2024).

Despite its promise, pruning reasoning models remains in its infancy and poses unique challenges, particularly in preserving long-horizon reasoning fidelity at higher sparsity levels. Recent efforts have primarily attempted to alleviate pruning damage by curating specialized calibration data; for instance, by leveraging reasoning trajectories generated by dense models (Lucas et al., 2026) or filtering them based on length and task difficulty (Xiang et al., 2025). However, these approaches fundamentally rely on the static output distribution of unpruned models, overlooking the distribution shift induced as parameters are removed. Pruning aligns the sparse model with the dense model only on these calibration data, yet during generation the sparse model conditions on its own outputs, which drift further from the calibration data as sparsity grows. Errors on these unaligned contexts then compound over long generations (Bengio et al., 2015; Ranzato et al., 2016), posing a key obstacle to pruning reasoning models at high sparsity.

Based on this observation, we introduce SCOUT (**S**parse **C**onstrained **O**n-policy optimization **U**nder **T**rust regions), an on-policy pruning framework for reasoning models. SCOUT casts reasoning-

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

model pruning as a constrained optimization problem that alternates between weight and mask updates, using trajectories generated by the current sparse model to guide pruning. Because these on-policy signals are local to the model that generated them, large mask updates can make them less representative of the updated sparse model. We therefore constrain each mask update with a KL-divergence trust region (Schulman et al., 2015) between consecutive sparse models, allowing the pruning mask to adapt gradually throughout optimization.

Empirically, across Qwen3-1.7B, 4B, and 8B on mathematical reasoning, code generation, and instruction following, SCOUT achieves substantial gains over existing pruning methods, particularly at high sparsity. Ablation studies reveal a strong interaction between the two components: on-policy information improves reasoning performance when mask updates are kept local by the trust region, whereas its benefit largely disappears without the trust region. Theoretically, we show that the trust region bounds both the effect of each local update and the resulting shift in the distribution of generated trajectories. Under an idealized optimization setting, we further establish a stationarity bound governed jointly by the trust radius and the trajectory update frequency. Together, these results indicate that reasoning-model pruning can be effectively approached as a principled on-policy optimization problem, where the pruning-induced distribution shift is explicitly controlled.

2 Related Work

LLM pruning. Post-training LLM pruning derives a sparse model from a dense one, optimizing to preserve the dense model's behavior on pre-collected calibration data. Most methods adopt a surrogate objective that reconstructs the layer- or block-wise outputs of the dense model on this data (Frantar & Alistarh, 2023; Sun et al., 2024; Meng et al., 2024; Bai et al., 2024; Lee et al., 2025; Shin et al., 2024), while another line dispenses with such reconstruction and instead targets the model's own output loss, increasing sparsity progressively (Lee et al., 2026; Jang et al., 2026). Across both lines, these methods have focused largely on settings involving little or no long-horizon generation, leaving pruning for long-form reasoning relatively underexplored despite its growing importance.

Reasoning model pruning. Reasoning-model pruning presents a different setting because reasoning models autoregressively generate long chains of thought (CoT). As generation proceeds, each next-token prediction is conditioned on an increasingly long prefix produced by the model itself, making the inputs encountered at inference dependent on its own generations. Recent work has therefore used model-generated reasoning traces for calibration. Ding et al. (2026) show that using CoT training data rather than a general-domain corpus better preserves reasoning performance; RAC instead calibrates on reasoning traces generated by the dense model itself (Lucas et al., 2026); and SSGR reports further gains from selecting these generations by difficulty and length (Xiang et al., 2025). However, these methods rely on calibration data fixed before pruning and do not update it as pruning proceeds, so at high sparsity the prefixes encountered by the sparse model can drift away from the calibration data.

On-policy learning and trust regions. On-policy learning, where training data depends on the model itself, has become increasingly important for reasoning language models. Methods such as GRPO learn from trajectories generated by the current model (Shao et al., 2024), while on-policy distillation (OPD) trains language models to match a teacher on student-generated trajectories (Agarwal et al., 2024; Gu et al., 2024; Lu & Lab, 2025). Because such supervision is tied to the current model's distribution, large updates can make it rapidly stale. Trust-region methods limit this shift by constraining successive models to remain close (Conn et al., 2000; Nocedal & Wright, 2006; Schulman et al., 2015), and have recently been combined with OPD (Xing et al., 2026). These methods control continuous updates, whereas pruning introduces discrete mask transitions. SCOUT extends these ideas to sparse optimization by using trajectories from the evolving sparse model and constraining mask updates with a KL trust region.

3 Method

Our primary goal is to prune a dense reasoning model while preserving its reasoning capabilities. To this end, we formulate pruning as an iterative sparse optimization problem with a trust region, alternating between weight and mask updates (Section 3.1). For the weight update, we optimize

the model under a fixed mask using on-policy and auxiliary objectives (Section 3.2). For the mask update, we rank coordinates by saliency, construct candidate masks along this ranking, and select the largest feasible change within the trust region (Section 3.3).

3.1 Problem Formulation and Approach

We consider the problem of obtaining a sparse model $Q_{\theta,m}$—with weights $\theta \in \mathbb{R}^d$ and binary mask $m \in \{0,1\}^d$ yielding effective parameters $\theta \odot m$—by pruning a dense reasoning model P while preserving reasoning quality. Given a prompt x drawn from a prompt distribution $\mathcal{D}_{\text{prompt}}$, we refer to an autoregressively generated sequence from a model as a rollout, and to any partial sequence encountered during that rollout as a prefix h. Since reasoning models generate answers autoregressively, we align the two models by minimizing the KL divergence between their next-token distributions on prefixes generated by the sparse model Q. Thus we formulate the pruning problem as:

$$\min_{\theta,m} \ \mathrm{KL}_{\mu^{Q_{\theta,m}}}\left(P \,\|\, Q_{\theta,m}\right) \qquad \text{s.t.} \qquad \|m\|_0 \le K, \tag{1}$$

where μ^Q denotes the distribution over prefixes induced by rollouts from model Q on prompts $x \sim \mathcal{D}_{\text{prompt}}$, K is the sparsity budget, and the prefix-averaged KL divergence is defined as

$$\mathrm{KL}_\rho(A \,\|\, B) := \mathbb{E}_{x\sim\mathcal{D}_{\text{prompt}},\, h\sim\rho}\left[D_{\mathrm{KL}}\left(A(\cdot \mid x,h) \,\|\, B(\cdot \mid x,h)\right)\right],$$

evaluated for models A and B over a prefix distribution ρ.

Equation (1) cannot be solved directly because evaluating the objective requires sampling from $\mu^{Q_{\theta,m}}$, which inherently depends on the parameters being optimized. A natural remedy is to fix the prefix distribution to that of the current model, optimize the model under that fixed distribution, and iteratively refresh the rollout distribution. We instantiate this scheme over T iterations, yielding a sequence of models $Q_0, Q_1, \ldots, Q_T$ initialized at the dense model $Q_0 = P$ (with $m_0 = \mathbf{1}^d$) and terminating at a sparse model with $\|m_T\|_0 = K$. At iteration t, replacing $\mu^{Q_{\theta,m}}$ with the prefix distribution $\mu^{Q_{t-1}}$ of the preceding model yields the surrogate objective $\mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\theta,m})$.

Because the surrogate fixes the prefix distribution at $\mu^{Q_{t-1}}$, it accurately approximates the true objective only when the updated sparse model produces prefixes similar to Q_{t-1}. When this condition is violated, two failure modes emerge. First, the update directions that minimize the surrogate are computed from information local to Q_{t-1}; a large update may move the model into a parameter region where these directions are no longer descent directions for the true objective. Second, while the surrogate is evaluated on prefixes from Q_{t-1}, the updated model will generate rollouts on its own prefixes at deployment. If the updated model deviates significantly from Q_{t-1}, the two prefix distributions diverge substantially—a mismatch that compounds with generation length and becomes particularly severe for reasoning models generating long chains of thought. Both failure modes can be mitigated by keeping each update close to Q_{t-1} in output distribution (Appendix E).

We formalize this principle by constraining each iteration within a KL-divergence trust region of radius $\delta > 0$:

$$(\theta_t, m_t) \in \arg\min_{\theta,m} \ \mathrm{KL}_{\mu^{Q_{t-1}}}(P \| Q_{\theta,m}) \quad \text{s.t.} \quad \mathrm{KL}_{\mu^{Q_{t-1}}}(Q_{t-1} \| Q_{\theta,m}) \le \delta, \ \ \|m\|_0 \le K_t, \tag{2}$$

where $Q_t := Q_{\theta_t,m_t}$. Here, each iteration removes as many weights as the trust region permits, and K_t denotes the resulting number of active weights, which is at least K.

We solve Equation (2) by alternating between weight and mask updates: each iteration t collects rollouts from Q_{t-1}, performs S_{proj} weight-update steps on the mask m_{t-1} (Section 3.2), and ends with a single mask update that produces m_t (Section 3.3).

3.2 Weight Update

During the weight update, the mask is fixed at m_{t-1}, reducing Equation (2) to an optimization problem over weights alone:

$$\min_{\theta} \ \mathrm{KL}_{\mu^{Q_{t-1}}}\left(P \| Q_{\theta,m_{t-1}}\right), \tag{3}$$

assuming that the continuous weight update remains within the trust region. In practice, the expectation in Equation (3) cannot be evaluated exactly. We therefore approximate it via an on-policy

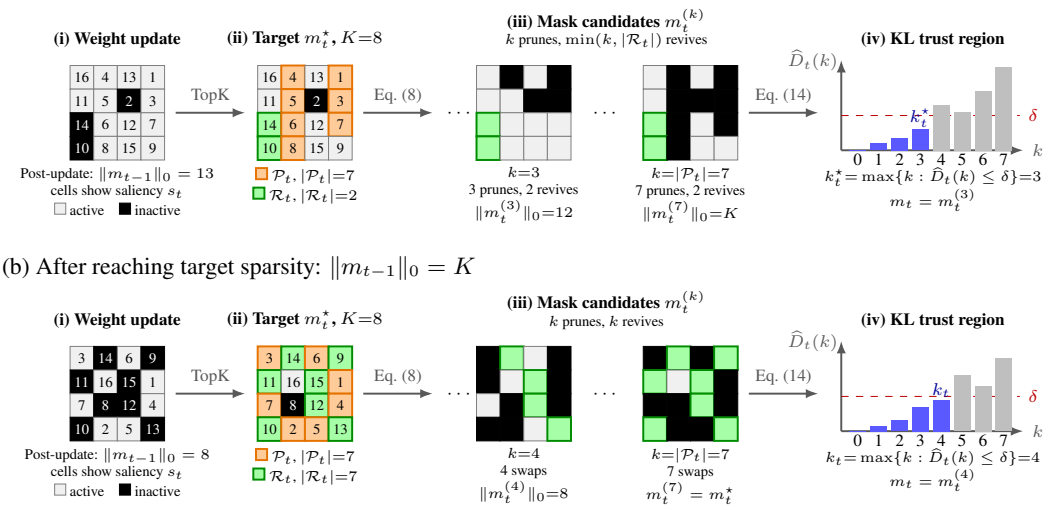


Figure 1: One SCOUT mask update on a toy 4×4 parameter block. (a) Before reaching the target sparsity, the path combines revivals with net pruning, progressively decreasing the number of active weights. (b) Once the target sparsity is reached, prune and revive candidates are paired one-to-one, allowing the same mechanism to perform support swaps at fixed sparsity. In both cases, the target support $m_t^\star$ is dynamically recomputed at later projection events as the weights and saliency evolve.

distillation (OPD) objective using a finite cache C_t of recent rollouts (x, h) from the evolving sparse model, periodically refreshed every S_{ref} optimizer steps. The objective is

$$\mathcal{L}_{\text{OPD},t}(\theta) = \frac{1}{|\mathcal{C}_t|} \sum_{(x,h)\in\mathcal{C}_t} \frac{1}{|h|} \sum_{i=1}^{|h|} D_{\text{KL}}\left(P(\cdot \mid x, h_{<i}) \,\middle\|\, Q_{\theta, m_{t-1}}(\cdot \mid x, h_{<i})\right). \tag{4}$$

While this on-policy cache provides essential supervision on prefixes encountered by the sparse model, it represents only a finite sample and provides limited coverage. To stabilize training and retain broad domain knowledge, we complement it with next-token prediction ($\mathcal{L}_{\text{NTP}}$) and knowledge distillation ($\mathcal{L}_{\text{KD}}$) evaluated over a fixed text corpus $\mathcal{D}_{\text{corpus}}$, whose prefixes are independent of model rollouts:

$$\mathcal{L}_t(\theta) = \mathcal{L}_{\text{NTP}}(\theta) + \lambda_{\text{KD}}\mathcal{L}_{\text{KD}}(\theta) + \lambda_{\text{OPD}}\mathcal{L}_{\text{OPD},t}(\theta). \tag{5}$$

Here, $\mathcal{L}_{\text{OPD},t}$ approximates the surrogate objective in Equation (3), while the corpus terms act as regularizers: $\mathcal{L}_{\text{KD}}$ aligns output distributions on standard text prefixes, and $\mathcal{L}_{\text{NTP}}$ provides ground-truth cross-entropy anchoring.

Within iteration t, we perform S_{proj} projected Adam steps (Kingma & Ba, 2014) on Equation (5) initialized at θ_{t-1}, sampling joint minibatches from $\mathcal{C}_t$ and $\mathcal{D}_{\text{corpus}}$. After each gradient step, the mask m_{t-1} is reapplied to ensure inactive weights remain zero. Importantly, gradients and optimizer states are tracked across all coordinates prior to projection, allowing the subsequent mask update to leverage historical first- and second-moment statistics when reconsidering inactive weights (Section 3.3).

3.3 MASK UPDATE

A mask update proceeds in two steps: first, coordinates are ordered by a saliency measure derived from Adam optimizer statistics, and then the mask is updated along this ordering as far as the trust region permits—either reducing active weights toward the target sparsity or, once the target is reached, replacing low-saliency weights with higher-saliency ones (Figure 1). We adopt this two-step procedure because directly optimizing Equation (2) over binary masks m is computationally intractable.

Let $\widetilde{\theta}_t$ denote the weights produced by the last Adam step of iteration t before the mask is applied. Because Adam states are tracked across all coordinates, including inactive ones, both the updated weight $\widetilde{\theta}_{t,i}$ and the second-moment estimate $v_{t,i}$ are available for every coordinate i. We define the coordinate-wise saliency score as:

$$s_{t,i} = |\widetilde{\theta}_{t,i}|\sqrt{v_{t,i}}. \tag{6}$$

This score combines the weight magnitude with a running estimate of its gradient scale: a coordinate is deemed salient when it is both large and has sustained substantial gradients during recovery. While this saliency metric has been explored in neural network pruning (Jang et al., 2026), we provide further theoretical justification tailored to our objective in Equation (2) at the end of this section.

The saliency ranking identifies a target support $m_t^\star$ comprising the K highest-scoring coordinates. Transitioning directly to $m_t^\star$, however, may violate the trust region condition. We therefore interpolate a sequence of candidate masks between m_{t-1} and $m_t^\star$, and use the trust region constraint to decide how far to move. Identifying binary masks with their support sets, define the prune and revive sets:

$$\mathcal{P}_t = m_{t-1} \setminus m_t^\star, \qquad \mathcal{R}_t = m_t^\star \setminus m_{t-1}, \tag{7}$$

ordered by increasing and decreasing saliency, respectively. Denoting their leading k elements by $\mathcal{P}_t^{[k]}$ and $\mathcal{R}_t^{[k]}$, we construct a family of candidate masks indexed by k:

$$m_t^{(k)} = \left(m_{t-1} \setminus \mathcal{P}_t^{[k]}\right) \cup \mathcal{R}_t^{[\min(k,|\mathcal{R}_t|)]}, \qquad 0 \le k \le |\mathcal{P}_t|. \tag{8}$$

At $k = 0$ the candidate is the current mask; at $k = |\mathcal{P}_t|$ it is the target $m_t^\star$. Each intermediate candidate removes the k lowest-saliency active coordinates and adds the highest-saliency inactive ones. The resulting sequence defines a saliency-ordered path from the current mask toward $m_t^\star$, progressively removing lower-saliency active coordinates and reviving higher-saliency inactive ones.

Before the target sparsity is reached, $|\mathcal{P}_t| > |\mathcal{R}_t|$, so moving far enough along the sequence produces net pruning. Once the target is reached, $|\mathcal{P}_t| = |\mathcal{R}_t|$ and the sequence consists only of swaps at fixed sparsity. In both regimes, previously pruned weights can be revived, because the saliency scores shift as the weights and rollout cache are updated across iterations: a coordinate that was unimportant at an earlier iteration may become important later.

We select the final mask by evaluating the empirical trust-region constraint on the rollout cache $\mathcal{C}_t$ over candidate masks, using an efficient candidate search described in Appendix A. Among the evaluated candidates, we accept the largest k satisfying the constraint, set $m_t = m_t^{(k)}$, and project the weights as $\theta_t = \widetilde{\theta}_t \odot m_t$. Thus, the saliency ranking determines the proposed mask-transition path, while the trust-region screen controls how far the mask moves along this path.

Justification of saliency score. We provide an intuitive justification connecting the saliency score to the mask-update objective. A first-order Taylor expansion of the surrogate objective in Equation (2) around the unmasked weights $\widetilde{\theta}_t$ (i.e., $Q_{\widetilde{\theta}_t} := Q_{\widetilde{\theta}_t,\mathbf{1}}$) reveals that the perturbation induced by applying a mask m is governed by the inner product between the objective gradient and the zeroed parameters:

$$\mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\widetilde{\theta}_t,m}) - \mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\widetilde{\theta}_t}) \approx -\left\langle \nabla \mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\widetilde{\theta}_t}),\, \widetilde{\theta}_t \odot (\mathbf{1} - m) \right\rangle. \tag{9}$$

Although this first-order linearization loosens when pruning a substantial fraction of weights, it suggests ranking coordinates by their individual contributions to this perturbation. Taking the absolute value and applying the triangle inequality bounds the objective change by $\sum_{i:m_i=0} |[\nabla \mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\widetilde{\theta}_t})]_i| |\widetilde{\theta}_{t,i}|$. Because the exact population gradient in Equation (9) is inaccessible, we utilize the stochastic gradients of the full training objective in Equation (5) tracked during the weight update. We can treat Adam's second-moment estimate $v_{t,i}$, which tracks an exponential moving average of squared stochastic gradients for each coordinate, as a proxy for the squared gradient magnitude of coordinate i. Consequently, using the Jensen's inequality $|\mathbb{E}[g]| \le \sqrt{\mathbb{E}[g^2]}$, the degradation caused by the mask is approximately bounded by:

$$\left|\mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\widetilde{\theta}_t,m}) - \mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_{\widetilde{\theta}_t})\right| \lesssim \sum_{i:m_i=0} s_{t,i}, \qquad \text{where } s_{t,i} = |\widetilde{\theta}_{t,i}|\sqrt{v_{t,i}}. \tag{10}$$

Under this bound, each $s_{t,i}$ quantifies the worst-case cost of zeroing coordinate i. Minimizing this upper bound under a sparsity budget of $\|m\|_0 = K$ requires zeroing the coordinates with the smallest saliency scores, which is mathematically equivalent to retaining the K highest-scoring coordinates—directly yielding the target support $m_t^\star$. We provide a more comprehensive theoretical analysis in Appendix D.

4 Optimization Analysis

In this section, we give convergence bounds for the SCOUT method using tools from classical optimization theory. However, analyzing the method involves two complications: the surrogate it optimizes changes when rollout cache is updated (every S_{ref} steps), and the mask update stage changes the mask (every S_{proj} steps). We show that SCOUT reaches the neighborhood of the stationary point of the surrogate objective. The full analysis including formal statement of assumptions and proofs is presented at Appendix F.

We first introduce notations and concept for theoretical analysis. Instead of treating two optimization variables θ, m separately, we consider $w := \theta \odot m$ in our analysis. Especially, we use $\tilde{w}_{t+1} := \tilde{\theta}_{t+1} \odot m_t$ and $w_{t+1} = \tilde{\theta}_{t+1} \odot m_{t+1}$ for analyzing each update step. Also, for analysis, we work with the objective $\mathcal{L}_{\text{OPD},t}$ instead of $\mathcal{L}_t$ in Equation (5). Following convention in optimization theory, in this section, we assume there is one optimizer update from step t to $t+1$, contrary to Section 3 where multiple updates occurred between step t and $t+1$. As we solve a sparsity-constrained optimization problem, we use the concept of support-stationary point (Beck & Eldar, 2013). For a support $S \subseteq \{1, \dots, d\}$ with $|S| \le K$ and objective $\mathcal{L}$, we say w with $\text{supp}(w) \subseteq S$ is support-stationary for $\mathcal{L}$ on S if $\nabla_S \mathcal{L}(w) := \nabla \mathcal{L}(w) \odot \mathbf{1}_S = 0$, where $\mathbf{1}_S \in \{0,1\}^d$ is the indicator of S. For our analysis, we measure convergence progress by $g_t := \nabla_{S_t} \mathcal{L}_t(w_t)$, where $S_t := \{i : m_{t,i} = 1\}$ is the active support at step t, with $|S_t| \le K_t$.

Theorem 4.1 (Stationarity of SCOUT, informal)**.** *Suppose that on the region visited by the iterates, (i) $\mathcal{L}_{\text{OPD},t}$ is L-smooth, (ii) every supervision target $T(\cdot|h)$ at $\mathcal{L}_{\text{OPD},t}$ satisfies $T(\cdot|h) \le RQ_t(\cdot|h)$ for some $R > 0$ and prefix h, and (iii) weight update is done by gradient descent with learning rate $\eta \le 1/L$. Also, let $\bar{\xi}$ bound $\xi_t := \mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_{t+1})$ at any rollout cache update. Then with $\psi(\delta) := R(\delta + \sqrt{\delta/2})$, for every $T \ge 1$, the following convergence bound holds:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \|g_t\|^2 \le \frac{2\mathcal{L}_0(w_0)}{\eta T} + \frac{2\psi(\delta)}{\eta S_{\text{proj}}} + \frac{2\bar{\xi}}{\eta S_{\text{ref}}}. \tag{11}$$

Proof Sketch. One iteration changes the surrogate by

$$\mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_t) = \underbrace{\mathcal{L}_t(\tilde{w}_{t+1}) - \mathcal{L}_t(w_t)}_{\text{weight update}} + \underbrace{\mathcal{L}_t(w_{t+1}) - \mathcal{L}_t(\tilde{w}_{t+1})}_{\Delta_t\text{: mask update}} + \underbrace{\mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_{t+1})}_{\xi_t\text{: rollout cache update}}. \tag{12}$$

The weight update term is bounded above by a negative quantity $-\frac{\eta}{2}\|g_t\|^2$, using descent lemma from classical optimization theory. The mask update term can be bounded by a trust region condition and Pinsker's inequality between two distributions induced by adjacent sparse models. The third term is derived by the assumption and the fact that rollout cache is updated every S_{ref} steps. Taking the sum over $t = 0$ to $T-1$ of Equation (12) gives the bound for $\frac{1}{T}\sum_{t=0}^{T-1} \|g_t\|^2$. □

In the right hand side of Equation (11), the first term is the usual $O(1/T)$ optimization error. The second and third term are error terms induced by staleness in mask and rollout cache, respectively. Note that we can control the second term by changing δ since $\psi(\delta) \to 0$ as $\delta \to 0$. SCOUT thus approaches support stationarity of the surrogate up to a floor set by the trust region constraint and rollout cache staleness.

5 Experiments

We first evaluate SCOUT across model scales and sparsity levels (Section 5.2), then examine whether the performance degradation caused by a baseline can be recovered through additional post-pruning

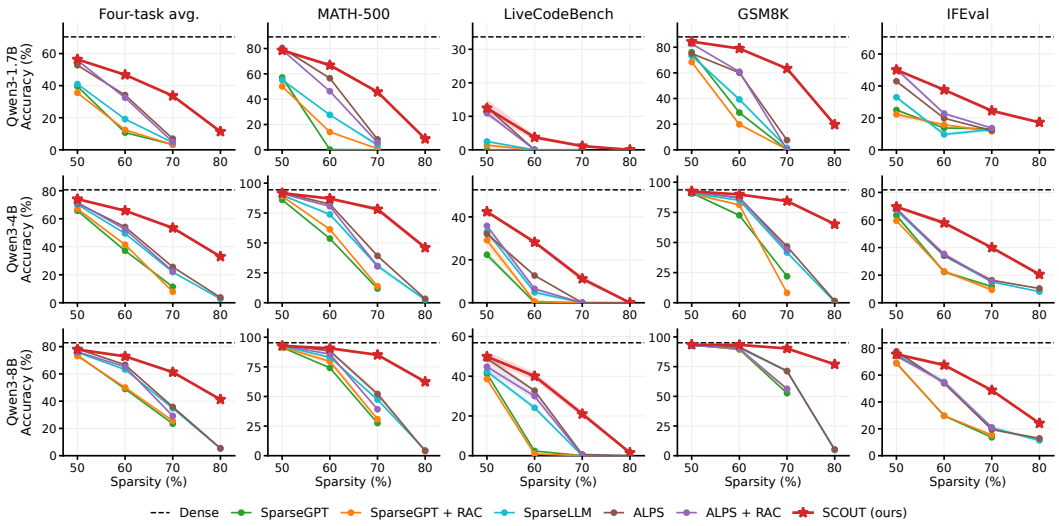


Figure 2: Accuracy versus unstructured sparsity on Qwen3-1.7B, 4B, and 8B (rows), per task and averaged. Baseline performance deteriorates sharply beyond 60%, while SCOUT retains substantially stronger performance at high sparsity. Task-wise numbers in Appendix C.

training (Section 5.3), analyze the individual and joint effects of the trust region and on-policy terms (Section 5.4), and evaluate whether the resulting sparsity translates into wall-clock decoding acceleration (Section 5.5).

5.1 SETUP

We evaluate pruned Qwen3-1.7B, 4B, and 8B (Yang et al., 2025) in thinking mode on MATH-500 (Lightman et al., 2024) and GSM8K (Cobbe et al., 2021) for mathematical reasoning, LiveCodeBench (Jain et al., 2025) for code generation, and IFEval (Zhou et al., 2023) for instruction following. We consider 50–80% sparsity, a challenging regime where prior work already reports substantial performance degradation at 50% sparsity (Lucas et al., 2026; Xiang et al., 2025). Baselines include SparseGPT (Frantar & Alistarh, 2023), a standard reconstruction-based method, and SparseLLM (Bai et al., 2024) and ALPS (Meng et al., 2024), two recent pruning methods shown to perform well at high sparsity. Following RAC (Lucas et al., 2026), we also evaluate SparseGPT and ALPS using calibration data in which dense-model generations replace OpenThoughts3 reasoning responses on the same prompts ("+RAC"). We report averages over three evaluation seeds, scoring mathematical tasks by correctness of the extracted final answer, LiveCodeBench by whether the generated code passes the tests, and IFEval by whether the response satisfies the specified instructions. For each task, we set the maximum generation length to rarely truncate dense-model responses. The fixed corpus consists of 80% OpenThoughts3 reasoning responses (Guha et al., 2026) and 20% FineWeb text (Penedo et al., 2024). Full details are provided in Appendix B.

5.2 PERFORMANCE ACROSS SPARSITY LEVELS

As shown in Figure 2, SCOUT achieves the strongest overall performance, and its advantage becomes particularly pronounced as sparsity increases. At 50–60% sparsity, the baselines remain relatively competitive, whereas at 70% and above, their performance deteriorates sharply. For example, on Qwen3-4B, ALPS's four-task average accuracy drops from 54.1% at 60% sparsity to 25.6% at 70%, while SCOUT retains 65.8% and 53.4%, respectively. We observe the same high-sparsity advantage consistently across Qwen3-1.7B, 4B, and 8B, and across all four evaluated tasks. Among the baselines, performance generally improves from SparseGPT to SparseLLM to ALPS, with ALPS providing the strongest baseline across sparsity levels and model scales.

We further examine the effect of RAC-style calibration by replacing the OpenThoughts3 responses with generations from the dense model. For SparseGPT, this change improves the four-task average

by 4.3 percentage points at 60% sparsity, but the effect reverses at 70%, where performance drops by 3.6 points. For the stronger ALPS baseline, the benefit is more limited: using dense-model generations yields little improvement and can instead hurt performance, including a 6.6-point drop in four-task average on Qwen3-8B at 70% sparsity.

Overall, generations from the dense model can improve pruning in some lower-sparsity settings, but this benefit becomes limited as sparsity increases. This trend is consistent with our motivation: as pruning changes the sparse model more substantially, the prefixes encountered during its own generation can increasingly differ from those represented by data collected from the dense model before pruning. SCOUT instead incorporates generations from the evolving sparse model throughout pruning, while constraining each mask update with a trust region.

Beyond accuracy, SCOUT generally produces shorter responses than the pruning baselines, particularly at higher sparsity; for example, on MATH-500 with Qwen3-4B at 60% sparsity, the average generation length is 5.1K tokens for SCOUT compared with 6.9K for ALPS. At the same time, generation length tends to increase with sparsity, consistent with observations in prior work (Lucas et al., 2026; Xiang et al., 2025); on the same model and task, SCOUT's average generation length increases from 3.9K tokens for the dense model to 5.1K at 60% sparsity and 7.2K at 70%. This is practically important because longer generations directly increase decoding cost and can offset part of the speedup from sparsity (Appendix C.1).

5.3 Post-pruning training narrows but does not close the gap

Training a baseline model with a fixed mask largely closes the gap with SCOUT at moderate sparsity, but becomes insufficient as sparsity increases. Since ALPS provides the strongest one-shot baseline, we train its remaining weights while keeping the mask fixed for the same number of optimization steps and fixed-corpus token budget as SCOUT, using either NTP+KD or the full NTP+KD+OPD weight-update objective in Equation (5). As shown in Figure 3, training substantially improves ALPS, yet SCOUT remains stronger at 70–80% sparsity on Qwen3-4B and 8B. For example, at 70% sparsity on Qwen3-4B, ALPS improves from 25.6 to 46.6 with NTP+KD and to 49.4 with NTP+KD+OPD, while SCOUT reaches 53.4. On Qwen3-8B, SCOUT matches the strongest baseline at 50% sparsity and its advantage grows with sparsity, reaching 1.7, 4.0, and 10.8 points at 60%, 70%, and 80%, respectively (Appendix C).

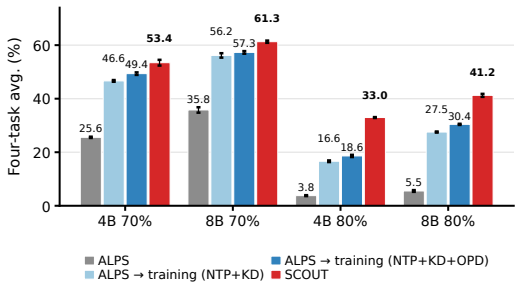


Figure 3: ALPS mask trained with subsets of SCOUT's objective, vs. SCOUT.

5.4 Interaction between the trust region and on-policy information

We next examine how the trust region and on-policy information work together in SCOUT. Our key finding is that on-policy information is beneficial when successive mask updates are kept local by the trust region, whereas its benefit largely disappears when the model changes too abruptly. Figure 4 examines this interaction from performance and rollout dynamics.

Figure 4a first shows this interaction at the level of downstream performance. With the trust region, on-policy information consistently improves performance on both Qwen3-4B and 8B. Without the trust region, however, its benefit largely disappears and can even reverse: the four-task average remains essentially unchanged on Qwen3-8B (51.3 to 51.5) and decreases on Qwen3-4B (43.2 to 42.4).

Figure 4b helps explain this behavior. We track the overlap between the top-100 vocabulary tokens under the sparse and dense next-token distributions, where higher overlap means that the two models favor more of the same next-token candidates (Li et al., 2026), together with the OPD loss on the sparse model's own rollouts, where lower loss indicates a closer match to the dense model. Without the trust region, the first projection sharply reduces this overlap and spikes the OPD loss, and this



(a) TR–on-policy interaction (b) Top-100 overlap and OPD loss (c) KL on own rollouts

Figure 4: Analysis of the trust region and on-policy information at 70% sparsity. (a) Four-task average with the trust region and/or on-policy information removed on Qwen3-4B and 8B. (b) Top-100 overlap between the sparse and dense models' next-token predictions and the OPD loss on the sparse model's own rollouts during pruning; shaded region indicates the mask-growth phase. (c) Per-token KL divergence from the dense model on the sparse model's own rollouts over held-out prompts. Details for (b) are provided in Appendix C.3 and for (c) in Appendix C.4.

degradation persists throughout training. SCOUT instead maintains higher overlap and lower OPD loss by changing the mask more gradually, indicating that the abrupt initial transition is not fully recovered by subsequent optimization.

With mask updates kept local by the trust region, on-policy information reduces mismatch along the sparse model's own generation trajectory. As shown in Figure 4c, removing it consistently increases the KL to the dense model on prefixes generated by the sparse model itself, while the two models remain nearly indistinguishable on fixed OpenThoughts3 text (Appendix C.4). Together, these results show complementary roles: the trust region limits mask-induced shifts, while on-policy information specifically reduces distributional drift on the prefixes encountered by the sparse model.

5.5 Decoding Acceleration

The practical benefit of high sparsity depends on whether the obtained sparse representation translates into reduced per-token cost and faster decoding at the sequence level, particularly for reasoning models whose generations can become longer after pruning. We therefore benchmark single-stream decoding of SCOUT-pruned Qwen3-8B models with the MACKO sparse kernel (Macko & Boža, 2026) and combine the measured per-token latency with the observed generation length on each benchmark (Appendix C.2). At 60% sparsity, SCOUT preserves over 95% of the dense-model performance on both MATH-500 and GSM8K while achieving approximately 1.44× per-token decoding speedup. Even after accounting for the longer generations of the sparse model, this corresponds to approximately 1.3× effective decoding speedup on both benchmarks. At higher sparsity, per-token decoding becomes faster still, but the accompanying increase in generation length increasingly offsets this gain. Overall, these results show that the sparsity produced by SCOUT can translate into practical decoding acceleration.

6 Discussion and Future Work

We introduced SCOUT, which uses on-policy information from the evolving sparse model and a KL trust region to guide iterative sparse optimization. Across Qwen3 models, SCOUT enables substantially stronger reasoning performance than existing prior art while also yielding practical decoding acceleration. However, performance still degrades as sparsity increases, particularly on more challenging tasks, while pruning also tends to lengthen generations, partially offsetting the acceleration from sparse execution. Whether this increase in generation length is an inherent consequence of reduced model capacity or can be mitigated through improved optimization remains unclear. Future work could address this issue by developing advanced sparse optimization objectives that jointly account for reasoning quality, sparsity, and generation cost.

Ethics Statement

This work studies model pruning and does not involve human subjects or the collection of personal or sensitive data. Our experiments use existing pretrained language models and publicly available training and evaluation datasets.

Reproducibility Statement

We provide the full description of SCOUT and its optimization procedure in Section 3, with additional algorithmic details in the appendix. The experimental setup, including datasets, model configurations, training hyperparameters, pruning settings, and evaluation protocols, is documented in Appendix B, and full task-wise results and additional analyses are also provided in the appendix. We will release the code required to reproduce our experiments upon publication.

AI Use Statement

Generative AI tools were used to assist with research discussions, including refining ideas and hypotheses developed by the authors, providing feedback on research methodology and experimental design, analyzing and interpreting experimental results, and assisting with theoretical arguments, mathematical claims, and proofs. They were also used to assist with method implementation and debugging, literature search and summarization, translation, and editing manuscript drafts for clarity and presentation. Generative AI tools were not used to generate synthetic datasets, clean or reformat datasets, or conduct qualitative or thematic data analysis. All research decisions, experimental results, mathematical claims and proofs, and final manuscript content were reviewed and verified by the authors. The authors take full responsibility for the content of this work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. Sparsellm: Towards global pruning of pre-trained language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Amir Beck and Yonina C Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal on Optimization*, 23(3):1480–1509, 2013.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. SIAM, 2000.

Longwei Ding, Anhao Zhao, Fanghua Ye, Ziyang Chen, and Xiaoyu Shen. From llms to lrms: Rethinking pruning for reasoning-centric models. *arXiv preprint arXiv:2601.18091*, 2026.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 108059–108130, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Hyeondo Jang, Kwanhee Lee, Dongyeop Lee, and Namhoon Lee. Hidden in plain sight: The overlooked significance of canonical elements for extreme llm sparsity. *arXiv preprint arXiv:2609.06557*, 2026.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Dongyeop Lee, Kwanhee Lee, Jinseok Chung, and Namhoon Lee. SAFE: Finding sparse and flat minima to improve pruning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Kwanhee Lee, Hyeondo Jang, Dongyeop Lee, Dan Alistarh, and Namhoon Lee. The unseen frontier: Pushing the limits of LLM sparsity with surrogate-free ADMM. In *International Conference on Learning Representations*, 2026.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026.

Ryan Lucas, Kayhan Behdin, Zhipeng Wang, Qingquan Song, Shao Tang, and Rahul Mazumder. Reasoning models can be accurately pruned via chain-of-thought reconstruction. In *International Conference on Learning Representations*, 2026.

Vladimír Macko and Vladimír Boža. MACKO: Sparse matrix-vector multiplication for low sparsity. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ah9xkFXCV6.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. ALPS: Improved optimization for highly sparse one-shot pruning for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897, 2015.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sungbin Shin, Wonpyo Park, Jaeho Lee, and Namhoon Lee. Rethinking pruning large language models: Benefits and pitfalls of reconstruction error minimization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1182–1191, 2024.

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

Yang Xiang, Yixin Ji, Juntao Li, and Min Zhang. Think before you prune: Selective self-generated calibration for pruning large reasoning models. *arXiv preprint arXiv:2511.18864*, 2025.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Algorithm and Implementation Details

Pruning scope. We prune every linear layer inside the transformer blocks (attention projections and MLP projections) with a single *global* sparsity budget shared across layers, so the per-layer sparsity is determined by saliency rather than fixed in advance. The token embedding and the output head are left dense. Masks are applied as hard zeros: inactive weights are zero at the start of every optimizer step, while gradients and Adam moments are maintained for all coordinates so that inactive coordinates can be scored and revived (Appendix D).

Saliency. The score $s_{t,i} = |\widetilde{\theta}_{t,i}|\sqrt{v_{t,i}}$ of Equation 10 reads $v_{t,i}$ directly from Adam's second-moment state (`exp_avg_sq`, $\beta_2 = 0.999$; Table 1), which is maintained for every coordinate including inactive ones. Ranking by $s_{t,i}$ is equivalent to ranking by $v_{t,i}\widetilde{\theta}^2_{t,i}$, which is what the implementation computes. The target mask $m^\star_t$ is the global top-K under s_t. We find the corresponding score threshold by logarithmic-space bisection, separately from the search over the transition size described next.

Empirical trust-region constraint. At a projection event, we estimate the KL change induced by the mask transition on a screening set C_t of four sequences of 512 tokens drawn from the current rollout cache, averaged over valid token positions:

$$\widehat{D}_t(k) = \frac{1}{|\mathcal{C}_t|} \sum_{h \in \mathcal{C}_t} D_{\mathrm{KL}}\Big(Q_{\widetilde{\theta}_t, m_{t-1}}(\cdot \mid h) \,\Big\|\, Q_{\widetilde{\theta}_t, m_t^{(k)}}(\cdot \mid h) \Big), \tag{13}$$

where the reference is the model immediately before the transition, not the dense model. The forward pass of the reference is computed once per event and cached, so each candidate k costs one forward pass of the candidate model. The accepted transition is

$$k_t^\star = \max\{k \in \mathcal{K}_t^{\mathrm{eval}} : \widehat{D}_t(k) \le \delta\}, \tag{14}$$

where $\mathcal{K}_t^{\mathrm{eval}} \subseteq \{0, \dots, |\mathcal{P}_t|\}$ is the finite set of candidates actually evaluated.

Warm-started search over the transition size. Evaluating every k is unnecessary. Assuming $\widehat{D}_t(k)$ is non-decreasing in k, we locate $k_t^\star$ with at most six evaluations per event:

1. *Warm start.* Evaluate the $k^\star$ accepted at the previous event. Because sparsity grows continuously, consecutive solutions are close.
2. *Doubling.* If the warm start passes, increase k by a step that doubles after each pass until a candidate fails; the first failing k is an upper bracket. If the warm start fails, it is the upper bracket.
3. *Bisection.* Spend the remaining budget bisecting the bracket, stopping early when it closes.
4. Accept the largest passing k and store it as the next warm start.

In practice the search converges in one or two evaluations. Monotonicity is an assumption, not a guarantee: if it fails, the search still returns a k that satisfies $\widehat{D}_t(k) \le \delta$, so the constraint is never violated and the solution is merely conservative. Because $|\mathcal{C}_t|$ is small and the reference forward is cached, the total cost of screening over a run is below 1% of the training FLOPs, even at the six-evaluation cap.

Sparsification trajectory. SCOUT starts from the dense model (0% sparsity) and each projection event moves as far along the prune–revive path as the trust region allows. On Qwen3-4B at 70%, the target is reached at step 232 of 2048; the remaining steps perform saliency-ordered swaps at fixed sparsity under the same rule, and the accepted transition size shrinks as training proceeds.

Ablation arms. "w/o TR" skips the search at the first projection event and accepts $k = |\mathcal{P}_t|$, i.e., the full target mask, and thereafter follows the same training schedule. "w/o on-policy" removes on-policy information from both steps: it sets $\lambda_{\mathrm{OPD}} = 0$ in Equation 5, rebalances the remaining two terms, and evaluates the trust-region screen on sequences from the fixed corpus instead of the rollout cache. "w/o TR and on-policy" applies both changes; "w/o offline KD" sets $\lambda_{\mathrm{KD}} = 0$ analogously.

Algorithm 1 summarizes the full procedure.

B Experimental Details

B.1 Data and Training

All training and calibration data are drawn from a fixed mixture of 80% OpenThoughts3 and 20% FineWeb. OpenThoughts3 provides long reasoning traces on mathematics, code, and science prompts, while FineWeb provides general web text. For on-policy training, we retain only the OpenThoughts3 prompts and regenerate their completions with the current sparse model. None of the evaluation benchmarks is included in the training or calibration data.

One-shot methods (SparseGPT, SparseLLM, and ALPS) are calibrated on 128 sequences of 2,048 tokens (0.26M tokens) sampled from this mixture. The "+RAC" variants follow Lucas et al. (2026): the OpenThoughts3 completions in the calibration set are replaced with traces generated by the dense model on the same prompts. Training-based methods (SCOUT and ALPS+training) process

Algorithm 1 SCOUT with warm-started trust-region search.

Require: dense model P, target budget K, radius δ, projection interval S_{proj}, refresh interval S_{ref}, steps T

$\theta_0 \leftarrow$ dense weights, $m_0 \leftarrow \mathbf{1}$, $k^{\text{ws}} \leftarrow 0$; initialize Adam state for all coordinates; fill rollout cache $\mathcal{C}_0$ from Q_{θ_0, m_0}
for $j = 0, \ldots, T-1$ **do**
  $\widetilde{\theta}_{j+1} \leftarrow \text{AdamStep}\big(\theta_j, \nabla_\theta \mathcal{L}_j\big)$ // before reapplying the mask
  if $(j+1) \bmod S_{\text{proj}} = 0$ **then**
    $m^\star \leftarrow \text{TopK}(s_{j+1}), \quad s_{j+1,i} = |\widetilde{\theta}_{j+1,i}|\sqrt{v_{j+1,i}}$
    $\mathcal{P} \leftarrow m_j \setminus m^\star$ (ascending saliency), $\quad \mathcal{R} \leftarrow m^\star \setminus m_j$ (descending saliency)
    draw screening set $\mathcal{B}_j$ (4×512 tokens) from $\mathcal{C}_j$; cache reference logits of $Q_{\widetilde{\theta}_{j+1}, m_j}$ on $\mathcal{B}_j$
    $k^\star \leftarrow \text{WarmStartedSearch}\big(k^{\text{ws}}, k \mapsto \widehat{D}(k)$ via Eq. 13, δ, max 6 evals$\big)$
    $m_{j+1} \leftarrow (m_j \setminus \mathcal{P}^{[k^\star]}) \cup \mathcal{R}^{[\min(k^\star, |\mathcal{R}|)]}; \quad k^{\text{ws}} \leftarrow k^\star$
  else
    $m_{j+1} \leftarrow m_j$
  end if
  $\theta_{j+1} \leftarrow \widetilde{\theta}_{j+1} \odot m_{j+1}$ // reapply hard mask
  if $(j+1) \bmod S_{\text{ref}} = 0$ **then**
    refill $\mathcal{C}_{j+1}$ with rollouts from $Q_{\theta_{j+1}, m_{j+1}}$
  else
    $\mathcal{C}_{j+1} \leftarrow \mathcal{C}_j$
  end if
end for
return Q_{θ_T, m_T}

134.2M tokens from the same fixed mixture. For the controlled training-based comparison, they use the same training data and fixed-corpus token budget.

We use the same training configuration for SCOUT across model scales and sparsity levels, except for the learning rate η and trust-region budget δ. The NTP, KD, and OPD terms in Equation 5 are equally weighted. On-policy rollouts are generated by the current sparse model and refreshed every $S_{\text{ref}} = 32$ optimizer steps. The rollout cache contains 256 responses, each with a maximum length of 512 tokens, and the mask is updated every $S_{\text{proj}} = 8$ optimizer steps. Table 1 summarizes the shared hyperparameters.

Table 1: Global hyperparameters of SCOUT shared across model scales and sparsity levels.

Hyperparameter	Value
Optimizer	Adam $(\beta_1, \beta_2) = (0.9, 0.999)$
Optimizer steps	2,048
Global batch size	8
Sequence length	8,192
Training tokens	134.2M
LR schedule	Cosine
Warm-up steps	256
Gradient clipping	1.0
Loss ratio (NTP:KD:OPD)	1 : 1 : 1
Rollout length	512 tokens
Rollout temperature	0.6
Rollout cache size	256
Rollout refresh interval S_{ref}	32
Projection interval S_{proj}	8
Trust-region screening set	4×512 tokens
Pruning scope	Global

Table 2 reports the learning rate and trust-region budget δ used for each model–sparsity setting. The latter is restricted to a small discrete set.

Table 2: Learning rate η and trust-region budget δ used across model scales and sparsity levels. Darker shading indicates a larger value within each hyperparameter.

	Sparsity	Qwen3-1.7B	Qwen3-4B	Qwen3-8B
η	50%	5e-5	5e-5	5e-5
	60%	5e-5	5e-5	5e-5
	70%	1e-4	1e-4	1e-4
	80%	1e-4	1e-4	1e-4
δ	50%	0.02	0.01	0.01
	60%	0.02	0.02	0.01
	70%	0.02	0.02	0.03
	80%	0.02	0.02	0.03

B.2 Evaluation

All models are evaluated in Qwen3 thinking mode using the recommended sampling parameters: temperature 0.6, top-p 0.95, and top-k 20. Unless otherwise noted, each pruned model is evaluated three times with different sampling seeds, and we report the mean and standard deviation across these evaluations. The reported standard deviation therefore measures stochastic decoding variance for a fixed pruned model.

Table 3 summarizes the generation budget and metric for each benchmark. A response that reaches `max_new_tokens` is counted as *truncated*. Scoring is performed independently of truncation: the benchmark-specific answer extraction rule is applied to the generated text even when the generation reaches the cutoff. Thus, a truncated response can still be scored as correct if a valid answer is produced before the limit. The generation budgets are chosen such that dense models are truncated on only 1–4% of prompts (Table 8).

Table 3: Evaluation protocol.

Benchmark	`max_new_tokens`	`max_model_len`	Metric
MATH-500	16,384	17,408	pass@1
IFEval	16,384	17,408	prompt-level strict accuracy
LiveCodeBench	32,768	33,792	pass@1
GSM8K	8,192	9,216	extractive match

C Full Results

Tables 4, 5, and 6 give task-wise results for Qwen3-4B, 1.7B, and 8B across sparsity levels. Table 7 gives the task-wise component ablation summarized in Figure 4, including an additional arm without offline KD. Table 8 gives per-task truncation rates at 70% and 80%.

Three observations not visible in the main text. First, on Qwen3-8B the gain of SCOUT over ALPS+training is already positive at 50% and 60% ($+1.2$, $+1.7$; three seeds each), unlike on 1.7B and 4B, so the sparsity at which mask adaptation begins to help is not fixed across scales. Second, the fixed-corpus terms are not redundant with on-policy distillation. On Qwen3-4B at 70%, removing offline KD costs 3.3 points, more than removing the on-policy term, and training on the on-policy term alone drops to 37.0, with generation lengths that run to the budget on most prompts: on-policy distillation without an anchor on reference text lets generations grow unchecked. The NTP and KD terms supply that anchor, and the on-policy term then corrects the model on the states it visits (Appendix C.4). Third, the dense-generated calibration traces of RAC show no consistent benefit on Qwen3-8B (single seeds): they lower ALPS at every sparsity (-3.0, -1.3, -6.6 at 50–70%) and change SparseGPT by -0.8, $+1.1$, and $+2.1$, so the moderate-sparsity gain seen on 4B does not transfer to the larger model.

Table 4: Task-wise accuracy (%) on Qwen3-4B across unstructured sparsity levels, mean$_{\pm\text{std}}$ over three seeds; †/‡ mark two/one seeds. Bold marks the best pruned value in each column of a block.

Sparsity	Method	MATH	LCB	GSM8K	IFEval	Avg
Dense	Dense	94.4±0.5	52.6±1.1	93.7±0.1	81.9±1.5	80.64±0.40
50%	SparseGPT	86.0±0.9	22.4±1.7	91.0±0.6	63.4±1.7	65.70±0.91
	SparseGPT+RAC	89.1±0.3	29.1±3.2	90.4±0.5	59.5±0.4	67.01±0.99
	SparseLLM	89.9±0.6	33.1±0.8	90.7±0.4	67.2±1.5	70.20±0.45
	ALPS	**92.1**±0.3	32.1±1.5	91.8±0.1	68.3±1.0	71.08±0.59
	ALPS+RAC	91.2±1.1	35.8±1.0	91.8±0.1	67.8±1.0	71.67±0.44
	ALPS+training	**92.1**±0.8	40.9±2.1	90.8±0.2	**71.2**±2.6	73.77±0.46
	SCOUT (Ours)	91.7±0.9	**42.4**±0.9	**92.5**±0.6	69.6±1.2	**74.03**±0.34
60%	SparseGPT	53.7±0.3	0.1±0.2	72.5±1.1	22.4±1.3	37.17±0.59
	SparseGPT+RAC	61.3±0.6	0.6±0.4	81.2±0.9	22.8±1.2	41.49±0.35
	SparseLLM	73.9±1.3	4.8±1.3	85.0±0.5	34.1±1.0	49.47±0.47
	ALPS	82.5±0.7	12.7±0.4	87.0±0.5	34.3±0.2	54.12±0.15
	ALPS+RAC	80.6±0.5	6.5±0.9	87.2±0.4	35.4±0.1	52.41±0.26
	ALPS+training	**87.7**±0.3	**30.6**±2.6	89.2±0.6	55.2±1.7	65.67±0.34
	SCOUT (Ours)	87.1±0.3	28.2±0.9	**89.8**±0.4	**57.9**±1.7	**65.76**±0.21
70%	SparseGPT	11.9±1.6	0.0±0.0	22.0±0.3	11.8±0.8	11.40±0.54
	SparseGPT+RAC	13.8±0.5	0.0±0.0	8.2±0.5	9.4±1.4	7.84±0.42
	SparseLLM	30.8±2.2	0.0±0.0	41.7±1.8	15.2±0.8	21.91±0.43
	ALPS	39.2±1.0	0.0±0.0	46.7±1.0	16.3±0.6	25.56±0.23
	ALPS+RAC	30.5±1.7	0.0±0.0	43.9±2.0	15.3±0.8	22.44±1.05
	ALPS+training (NTP+KD)	72.2±1.9	6.6±1.1	79.6±0.1	28.2±2.3	46.64±0.32
	ALPS+training	73.3±0.5	9.6±0.8	82.9±0.5	31.6±0.6	49.36±0.47
	SCOUT (Ours)	**78.2**±2.2	**11.2**±1.3	**84.4**±1.0	**40.0**±0.4	**53.44**±1.07
80%	SparseLLM	2.4±1.2	0.0±0.0	0.8±0.1	8.1±0.2	2.81±0.36
	ALPS	3.3±0.3	0.0±0.0	1.5±0.5	10.4±0.3	3.78±0.17
	ALPS+training (NTP+KD)†	16.4±0.8	0.0±0.0	33.7±0.7	16.5±1.3	16.63±0.30
	ALPS+training	17.9±0.3	0.0±0.0	39.1±0.3	17.4±1.4	18.61±0.41
	SCOUT (Ours)	**46.1**±1.7	0.0±0.0	**65.2**±0.8	**20.6**±1.5	**32.96**±0.16

Table 5: Task-wise accuracy (%) on Qwen3-1.7B across unstructured sparsity levels, mean$_{\pm\text{std}}$ over three seeds; †/‡ mark two/one seeds. Bold marks the best pruned value in each column of a block.

Sparsity	Method	MATH	LCB	GSM8K	IFEval	Avg
Dense	Dense	89.2$_{\pm 1.8}$	33.7$_{\pm 1.3}$	88.1$_{\pm 0.2}$	70.7$_{\pm 0.4}$	70.43$_{\pm 0.04}$
50%	SparseGPT	57.1$_{\pm 2.9}$	0.0$_{\pm 0.0}$	76.2$_{\pm 0.0}$	24.9$_{\pm 0.6}$	39.57$_{\pm 0.84}$
	SparseGPT+RAC	50.1$_{\pm 0.3}$	1.4$_{\pm 0.9}$	68.5$_{\pm 1.1}$	22.2$_{\pm 2.1}$	35.53$_{\pm 0.51}$
	SparseLLM	55.2$_{\pm 1.2}$	2.5$_{\pm 0.2}$	73.4$_{\pm 1.2}$	32.9$_{\pm 1.1}$	41.00$_{\pm 0.33}$
	ALPS	**80.3**$_{\pm 1.6}$	12.7$_{\pm 0.0}$	75.1$_{\pm 0.6}$	42.9$_{\pm 0.8}$	52.77$_{\pm 0.32}$
	ALPS+RAC	78.9$_{\pm 0.3}$	10.9$_{\pm 0.9}$	82.7$_{\pm 0.6}$	50.5$_{\pm 1.3}$	55.74$_{\pm 0.54}$
	ALPS+training	79.2$_{\pm 0.3}$	**16.4**$_{\pm 1.3}$	84.3$_{\pm 0.7}$	**54.3**$_{\pm 2.4}$	**58.56**$_{\pm 0.80}$
	SCOUT (Ours)	78.6$_{\pm 1.2}$	12.4$_{\pm 2.5}$	**84.4**$_{\pm 0.7}$	50.1$_{\pm 0.5}$	56.38$_{\pm 0.70}$
60%	SparseGPT	0.2$_{\pm 0.2}$	0.0$_{\pm 0.0}$	29.0$_{\pm 1.3}$	13.7$_{\pm 0.0}$	10.72$_{\pm 0.33}$
	SparseGPT+RAC	14.1$_{\pm 0.6}$	0.0$_{\pm 0.0}$	19.8$_{\pm 1.1}$	15.7$_{\pm 0.5}$	12.40$_{\pm 0.54}$
	SparseLLM	27.6$_{\pm 0.2}$	0.0$_{\pm 0.0}$	39.3$_{\pm 1.7}$	9.7$_{\pm 1.3}$	19.14$_{\pm 0.24}$
	ALPS	56.5$_{\pm 0.5}$	0.0$_{\pm 0.0}$	60.2$_{\pm 0.6}$	19.8$_{\pm 0.2}$	34.12$_{\pm 0.27}$
	ALPS+RAC	46.3$_{\pm 0.3}$	0.0$_{\pm 0.0}$	60.9$_{\pm 0.3}$	22.7$_{\pm 1.0}$	32.48$_{\pm 0.34}$
	ALPS+training	**69.1**$_{\pm 0.6}$	**8.1**$_{\pm 1.1}$	78.2$_{\pm 0.9}$	**38.4**$_{\pm 2.0}$	**48.44**$_{\pm 0.57}$
	SCOUT (Ours)	66.9$_{\pm 1.4}$	3.7$_{\pm 0.6}$	**79.0**$_{\pm 0.5}$	37.6$_{\pm 0.9}$	46.79$_{\pm 0.40}$
70%	SparseGPT	0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$	13.1$_{\pm 0.8}$	3.28$_{\pm 0.21}$
	SparseGPT+RAC	0.9$_{\pm 0.5}$	0.0$_{\pm 0.0}$	0.2$_{\pm 0.1}$	11.6$_{\pm 0.8}$	3.21$_{\pm 0.10}$
	SparseLLM	3.9$_{\pm 1.8}$	0.0$_{\pm 0.0}$	1.6$_{\pm 0.3}$	12.7$_{\pm 1.2}$	4.53$_{\pm 0.13}$
	ALPS	8.2$_{\pm 0.9}$	0.0$_{\pm 0.0}$	7.5$_{\pm 0.3}$	12.1$_{\pm 0.6}$	6.95$_{\pm 0.36}$
	ALPS+RAC	5.8$_{\pm 1.2}$	0.0$_{\pm 0.0}$	0.3$_{\pm 0.0}$	13.6$_{\pm 0.4}$	4.92$_{\pm 0.32}$
	ALPS+training	27.1$_{\pm 1.0}$	0.0$_{\pm 0.0}$	40.1$_{\pm 1.3}$	18.8$_{\pm 0.8}$	21.51$_{\pm 0.21}$
	SCOUT (Ours)	**45.7**$_{\pm 1.3}$	**1.1**$_{\pm 0.4}$	**63.4**$_{\pm 0.7}$	**24.5**$_{\pm 1.0}$	**33.67**$_{\pm 0.67}$
80%	SCOUT (Ours)	**8.7**$_{\pm 0.4}$	0.0$_{\pm 0.0}$	**19.6**$_{\pm 0.2}$	**17.2**$_{\pm 0.6}$	**11.37**$_{\pm 0.29}$

Table 6: Task-wise accuracy (%) on Qwen3-8B across unstructured sparsity levels, mean$_{\pm\text{std}}$ over three seeds; †/‡ mark two/one seeds. Bold marks the best pruned value in each column of a block.

Sparsity	Method	MATH	LCB	GSM8K	IFEval	Avg
Dense	Dense	95.3±1.0	56.8±0.6	95.1±0.4	84.1±0.4	82.84±0.14
50%	SparseGPT	91.4±0.5	41.3±0.6	93.2±0.1	68.9±0.4	73.69±0.35
	SparseGPT+RAC	91.6±0.4	38.6±0.9	93.2±0.3	68.9±1.6	73.08±0.23
	SparseLLM	92.5±0.4	42.4±1.1	92.7±0.2	75.4±0.4	75.74±0.32
	ALPS	**93.7**±0.4	48.9±2.1	**93.5**±0.3	**77.7**±1.2	**78.43**±0.97
	ALPS+RAC	92.9±0.6	44.8±1.9	93.1±0.1	74.2±1.3	76.22±0.87
	ALPS+training	92.2±0.3	46.6±1.9	93.2±0.2	74.6±1.3	76.66±0.54
	SCOUT (Ours)	92.7±0.3	**49.9**±2.4	93.5±0.6	75.5±1.2	77.90±0.74
60%	SparseGPT	74.2±0.2	2.4±0.2	89.5±0.7	29.7±0.6	48.94±0.22
	SparseGPT+RAC	79.7±2.8	0.9±0.4	89.7±0.2	29.8±0.1	50.02±0.72
	SparseLLM	83.1±1.1	24.1±1.3	90.7±0.4	54.8±1.3	63.19±0.41
	ALPS	88.4±1.3	32.7±0.6	91.6±0.2	54.0±0.7	66.67±0.42
	ALPS+RAC	85.8±0.7	30.0±1.9	90.2±0.1	54.8±2.2	65.19±0.14
	ALPS+training	90.5±0.5	38.8±0.0	90.6±0.4	64.6±1.3	71.14±0.34
	SCOUT (Ours)	**90.7**±1.1	**40.0**±2.1	**93.3**±0.2	**67.5**±1.2	**72.88**±0.39
70%	SparseGPT	27.5±0.5	0.0±0.0	52.6±0.8	13.7±0.4	23.46±0.17
	SparseGPT+RAC	30.8±1.1	0.0±0.0	56.1±1.4	15.4±1.0	25.58±0.08
	SparseLLM	47.3±1.0	0.0±0.0	71.2±0.5	20.8±1.6	34.81±0.76
	ALPS†	52.1±2.7	0.6±0.3	71.0±1.2	19.5±0.1	35.79±1.01
	ALPS+RAC	39.2±0.6	0.1±0.2	56.3±0.4	21.1±0.8	29.16±0.10
	ALPS+training (NTP+KD)	79.1±1.5	18.5±1.8	85.1±0.8	41.9±0.7	56.16±0.82
	ALPS+training	81.2±1.7	18.3±0.4	87.0±0.8	42.6±1.5	57.29±0.48
	SCOUT (Ours)	**85.1**±0.8	**21.0**±1.5	**90.3**±0.4	**48.7**±0.3	**61.29**±0.41
80%	SparseLLM	4.6±0.2	0.0±0.0	4.8±0.6	11.4±1.5	5.19±0.51
	ALPS	3.9±0.6	0.0±0.0	5.3±0.3	12.9±1.3	5.52±0.32
	ALPS+training (NTP+KD)	36.7±0.8	0.4±0.6	54.2±0.9	18.9±0.3	27.53±0.20
	ALPS+training	41.5±1.1	0.1±0.2	62.1±1.6	17.9±1.0	30.41±0.21
	SCOUT (Ours)	**62.4**±1.1	**1.5**±0.4	**76.9**±0.8	**24.2**±0.9	**41.24**±0.56

Table 7: Task-wise component ablation, mean$_{\pm\text{std}}$ over three seeds; †/‡ mark two/one seeds. Every arm is matched to its SCOUT reference in learning rate, training objective, projection frequency, and budget. "w/o TR" accepts the full target mask at the first projection; "w/o on-policy" removes the on-policy term from the objective and screens mask transitions on the fixed corpus instead of the rollout cache; "w/o TR and on-policy" does both; "w/o KD" sets $\lambda_{\text{KD}} = 0$; "w/o NTP+KD" sets $\lambda_{\text{NTP}} = \lambda_{\text{KD}} = 0$.

Setting	Variant	MATH	LCB	GSM8K	IFEval	Avg
Qwen3-1.7B, 50%	SCOUT (full)	**78.6**$\pm$1.2	**12.4**$\pm$2.5	**84.4**$\pm$0.7	**50.1**$\pm$0.5	**56.38**$\pm$0.70
	SCOUT w/o on-policy	78.1$\pm$1.9	12.2$\pm$3.0	84.0$\pm$0.2	49.4$\pm$0.9	55.94$\pm$1.24
Qwen3-1.7B, 60%	SCOUT (full)	**66.9**$\pm$1.4	**3.7**$\pm$0.6	**79.0**$\pm$0.5	**37.6**$\pm$0.9	**46.79**$\pm$0.40
	SCOUT w/o on-policy	65.6$\pm$0.7	1.5$\pm$0.4	77.1$\pm$0.6	32.8$\pm$2.8	44.25$\pm$0.88
Qwen3-4B, 50%	SCOUT (full)	**91.7**$\pm$0.9	42.4$\pm$0.9	**92.5**$\pm$0.6	69.6$\pm$1.2	74.03$\pm$0.34
	SCOUT w/o on-policy	91.5$\pm$0.6	**44.5**$\pm$0.6	90.8$\pm$0.7	**72.0**$\pm$1.2	**74.68**$\pm$0.46
Qwen3-4B, 60%	SCOUT (full)	**87.1**$\pm$0.3	**28.2**$\pm$0.9	**89.8**$\pm$0.4	**57.9**$\pm$1.7	**65.76**$\pm$0.21
	SCOUT w/o on-policy	86.3$\pm$1.3	26.9$\pm$1.3	88.6$\pm$0.9	56.6$\pm$1.2	64.61$\pm$0.86
Qwen3-4B, 70%	SCOUT (full)	**78.2**$\pm$2.2	**11.2**$\pm$1.3	84.4$\pm$1.0	**40.0**$\pm$0.4	**53.44**$\pm$1.07
	SCOUT w/o TR	62.6$\pm$1.2	1.7$\pm$1.1	73.7$\pm$0.7	31.5$\pm$0.9	42.39$\pm$0.64
	SCOUT w/o TR and on-policy	63.3$\pm$2.1	3.6$\pm$1.2	74.3$\pm$0.7	31.7$\pm$1.5	43.25$\pm$0.83
	SCOUT w/o on-policy	76.9$\pm$1.0	7.7$\pm$1.1	**84.8**$\pm$0.4	37.0$\pm$1.3	51.59$\pm$0.13
	SCOUT w/o KD	72.9$\pm$0.3	8.0$\pm$0.6	82.9$\pm$0.8	36.5$\pm$1.2	50.08$\pm$0.57
	SCOUT w/o NTP+KD	50.2$\pm$0.9	0.4$\pm$0.4	77.0$\pm$0.6	20.6$\pm$0.3	37.04$\pm$0.25
Qwen3-8B, 60%	SCOUT (full)	**90.7**$\pm$1.1	**40.0**$\pm$2.1	**93.3**$\pm$0.2	**67.5**$\pm$1.2	**72.88**$\pm$0.39
	SCOUT w/o TR‡	85.0	23.5	88.1	63.0	64.91
Qwen3-8B, 70%	SCOUT (full)	**85.1**$\pm$0.8	**21.0**$\pm$1.5	**90.3**$\pm$0.4	**48.7**$\pm$0.3	**61.29**$\pm$0.41
	SCOUT w/o TR	72.1$\pm$0.7	10.3$\pm$1.1	82.5$\pm$0.4	41.2$\pm$0.6	51.52$\pm$0.25
	SCOUT w/o TR and on-policy	73.5$\pm$0.3	11.9$\pm$0.4	80.1$\pm$1.3	39.8$\pm$0.9	51.32$\pm$0.40
	SCOUT w/o on-policy	83.3$\pm$1.3	14.9$\pm$1.0	89.7$\pm$0.9	48.6$\pm$2.6	59.12$\pm$0.70

Table 8: Truncation rate (%): fraction of responses that exhaust the generation budget without producing an answer, per task and averaged, at 70% and 80% sparsity. †/‡ mark two/one seeds.

Model	Method	MATH	LCB	GSM8K	IFEval	Avg
Qwen3-1.7B	Dense	4.8	2.0	1.1	4.7	3.1
	ALPS (70%)	99.6	97.9	95.1	97.1	97.4
	ALPS+training (70%)	85.5	100.0	83.0	96.6	91.3
	SCOUT (70%)	46.4	96.1	36.7	80.8	65.0
	SCOUT (80%)	97.3	100.0	98.6	99.9	98.9
Qwen3-4B	Dense	3.1	2.5	0.9	2.0	2.1
	ALPS (70%)	99.8	69.6	99.0	95.5	91.0
	ALPS+training (70%)	23.1	71.6	7.3	52.6	38.7
	SCOUT (70%)	20.3	67.0	9.6	46.0	35.8
	ALPS (80%)	99.6	93.9	99.8	92.4	96.4
	ALPS+training (80%)	93.7	99.9	83.1	98.1	93.7
	SCOUT (80%)	51.8	95.2	39.6	93.7	70.1
Qwen3-8B	Dense	3.7	2.9	1.0	2.2	2.4
	ALPS (70%)†	92.7	100.0	94.3	99.8	96.7
	ALPS+training (70%)	17.5	58.3	8.0	38.4	30.6
	SCOUT (70%)	14.7	54.1	6.6	31.7	26.8
	ALPS (80%)	95.1	99.9	98.7	84.5	94.5
	ALPS+training (80%)	62.1	98.3	40.8	88.9	72.5
	SCOUT (80%)	40.6	95.4	28.0	90.3	63.6

C.1 Generation Length and Inference Cost

Pruning is motivated by inference cost, but a sparse reasoning model that generates longer chains of thought can forfeit part of that saving. Table 9 reports the mean number of generated tokens per prompt and a first-order cost proxy, the token ratio relative to dense multiplied by the fraction of active weights. At 70% sparsity, SCOUT generates about $1.8\times$ (4B) and $1.6\times$ (8B) as many tokens as the dense model, so the per-prompt weight-multiply count is 0.47–$0.55\times$ dense rather than the $0.3\times$ the sparsity alone would suggest; at 80% the ratio is 0.43–$0.49\times$.

Table 9: Generation length and an inference-cost proxy on the reasoning benchmarks. "Tokens" is the mean number of generated tokens per prompt, averaged over MATH-500, LiveCodeBench, and GSM8K; "ratio" is relative to the dense model. The cost proxy multiplies the token ratio by the fraction of active weights, i.e., the relative number of weight-multiply operations per prompt; it ignores attention and KV-cache costs, which scale with length rather than sparsity, and so understates the cost of longer generations. †/‡: two/one seeds.

Model	Method	MATH	LCB	GSM8K	Tokens (ratio)	Cost proxy
Qwen3-4B	Dense	3,940	14,186	1,753	6,626 (1.00×)	1.00×
	ALPS+training (70%)	7,652	26,809	3,269	12,577 (1.90×)	0.57×
	SCOUT (70%)	7,217	25,672	3,372	12,087 (1.82×)	0.55×
	ALPS+training (80%)	15,797	32,739	7,456	18,664 (2.82×)	0.56×
	SCOUT (80%)	11,326	31,748	5,348	16,141 (2.44×)	0.49×
Qwen3-8B	Dense	4,523	14,513	1,959	6,998 (1.00×)	1.00×
	ALPS+training (70%)	6,544	24,027	3,173	11,248 (1.61×)	0.48×
	SCOUT (70%)	6,257	23,678	2,759	10,898 (1.56×)	0.47×
	ALPS+training (80%)	12,250	32,380	5,645	16,758 (2.39×)	0.48×
	SCOUT (80%)	9,514	31,647	4,379	15,180 (2.17×)	0.43×

C.2 Wall-Clock Decoding Latency

We integrate MACKO (Macko & Boža, 2026), an unstructured sparse matrix–vector kernel, into vLLM 0.10 and measure single-stream (batch size one) fp16 decoding of the Qwen3-8B SCOUT models on one L40S. Dense and sparse models share the same engine, PagedAttention, and CUDA graphs. MACKO compresses a linear layer only when its density is at most 0.5; because SCOUT allocates sparsity non-uniformly across layers, the number of compressed layers grows with the target sparsity (Table 10), and the remaining layers run dense.

Table 10: Single-stream decoding of Qwen3-8B SCOUT models with MACKO on one L40S. Per-token speedup is the range over generation lengths of 512–8,192 tokens. Weight footprint counts the weights on the decode path (compressed layers in MACKO format, remaining layers dense).

	50%	60%	70%	80%
Compressed layers (of 144)	60	99	121	136
Per-token speedup vs. dense	1.22–1.23×	1.43–1.45×	1.75–1.80×	2.17–2.26×
Decode-path weights (GiB; dense 15.26)	11.99	9.85	7.65	5.95

For reasoning models the per-token speedup is not the whole cost, because pruning changes the number of generated tokens. Table 11 combines the measured per-token latency with the average output length of the dense and sparse models on the reasoning benchmarks.

Table 11: Effective decoding speedup of Qwen3-8B SCOUT models over dense on the reasoning benchmarks: per-token speedup divided by the ratio of generated tokens (in parentheses).

	MATH-500	GSM8K	LiveCodeBench
50%	1.13–1.14× (1.08)	1.18–1.20× (1.03)	1.02–1.03× (1.19)
60%	1.25–1.27× (1.14)	1.29–1.31× (1.10)	1.07–1.09× (1.33)
70%	1.26–1.30× (1.38)	1.24–1.28× (1.41)	1.07–1.10× (1.63)
80%	1.03–1.08× (2.10)	0.97–1.01× (2.24)	1.00–1.04× (2.18)

On the mathematical benchmarks the effective speedup peaks at 60–70% sparsity (1.24–1.31×), where the per-token gain outweighs the 1.1–1.4× longer generations.

C.3 On-Policy Signal During Training

Figure 4(b) tracks the overlap between the sparse and dense models' top-100 next-token sets and the on-policy distillation loss on the sparse model's own rollouts during pruning. Figure 5 shows the same behavior on Qwen3-8B: without the trust region, the abrupt mask transition sharply degrades both statistics, whereas SCOUT changes them more gradually and maintains closer agreement with the dense model.

C.4 Divergence on Fixed Text versus Own Rollouts

Equation 1 asks the sparse model to match the dense model on the prefixes it generates itself, whereas the fixed-corpus terms of Equation 5 match it on text it did not generate. Figure 6 measures both quantities directly for the 70% Qwen3-4B SCOUT model and its ablation without on-policy information. We draw 30 prompts from OpenThoughts3 held out from training (prompts whose content overlaps the training set are excluded at selection time), and for each prompt compute the per-token $\mathrm{KL}(P(\cdot \mid h) \,\|\, Q(\cdot \mid h))$ at every position of (i) the dataset's own solution, cut to 2,048 tokens, and (ii) 16 rollouts of length 2,048 sampled from the sparse model under the evaluation decoding parameters.

SCOUT and the model without on-policy information are indistinguishable on fixed text (0.185 vs. 0.185 per token). On their own rollouts, SCOUT's divergence rises only to 0.191 (paired gap $+0.006 \pm 0.007$, $t = 0.8$), while the model without on-policy information rises to 0.213 (gap $+0.029 \pm 0.006$, $t = 4.9$). The fixed-corpus objective is therefore reached equally well with or without on-policy information, and the difference between the two models exists only on the states



Figure 5: Qwen3-8B, 70%: top-100 next-token overlap with the dense model on the sparse model's own rollouts (top) and the on-policy distillation loss on those rollouts (bottom, log scale), per optimizer step on the training rollout cache; 8-step moving average; shaded: mask-growth phase.

each actually visits during generation, which is the quantity the on-policy term is designed to optimize and the fixed-corpus terms cannot observe.

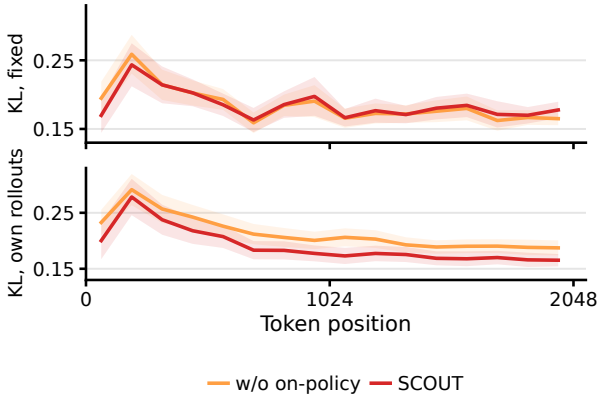


Figure 6: Per-token $\mathrm{KL}(P \,\|\, Q)$ by position for the 70% Qwen3-4B SCOUT model and the model without on-policy information, on fixed OpenThoughts3 text (top) and on each model's own rollouts (bottom), over 30 held-out prompts; bands: SEM. The two models coincide on fixed text and separate only on their own rollouts; Figure 4(c) in the main text shows the bottom panel.

A second measurement: how far each policy has moved from dense. The KL gap asks how well a model matches the dense model on the states it visits. A different question is how far the states themselves have moved: whether the sparse model's rollouts are distinguishable from the dense model's rollouts for the same prompts. We embed 2,048-token rollouts (8 per prompt, 30 prompts, sliced into seven windows each) with the dense model's hidden states, mean-pooled, and fit a logistic-regression probe to separate each sparse model's windows from the dense model's, with prompts held out by grouped five-fold cross-validation so the probe cannot exploit problem wording. The out-of-fold AUC is the measure; splitting the dense model's own rollouts in half within each prompt gives the same-distribution floor. Figure 7 plots the AUC at every depth, Figure 8 the score distributions behind it, and Table 12 the values;

Probe depth. Because one forward pass yields every layer's hidden states, the probe is read at eight depths (4, 9, 13, 18, 22, 27, 31, 36) on the same rollouts, with the same windows, the same prompt-grouped cross-validation, and the floor refit at each depth. The separation between arms is a mid-depth phenomenon: it is largest around layer 18 and shrinks toward the final block, where

final-layer states sit next to the output distribution and the fact that a model is pruned is decodable from almost any window.

Table 12: Probe AUC above the same-distribution floor, by encoder depth (Qwen3-4B, 70%, 30 held-out prompts). The floor is refit per depth.

Layer	SCOUT	SCOUT w/o on-policy	ALPS+training	Floor
4	+0.103	+0.417	+0.181	0.504
9	+0.060	+0.272	+0.098	0.531
13	+0.067	+0.191	+0.112	0.507
18	**+0.052**	+0.214	+0.172	0.505
22	+0.164	+0.267	+0.281	0.512
27	+0.210	+0.324	+0.298	0.502
31	+0.192	+0.311	+0.292	0.498
36	+0.392	+0.434	+0.423	0.508

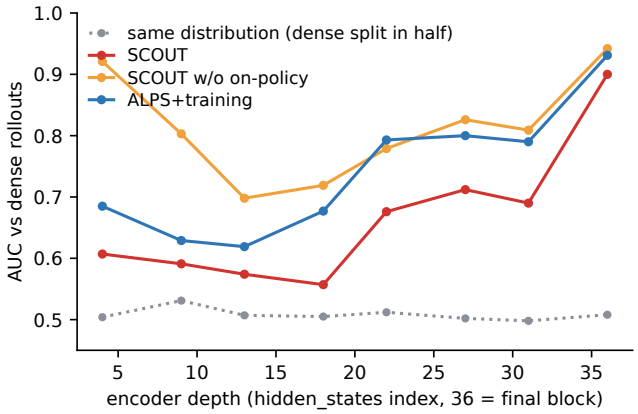


Figure 7: Probe AUC against dense rollouts as a function of encoder depth, with the same-distribution floor (dense rollouts split in half within each prompt, refit per depth). The distinction between arms is largest at mid-depth and collapses at the final layers.

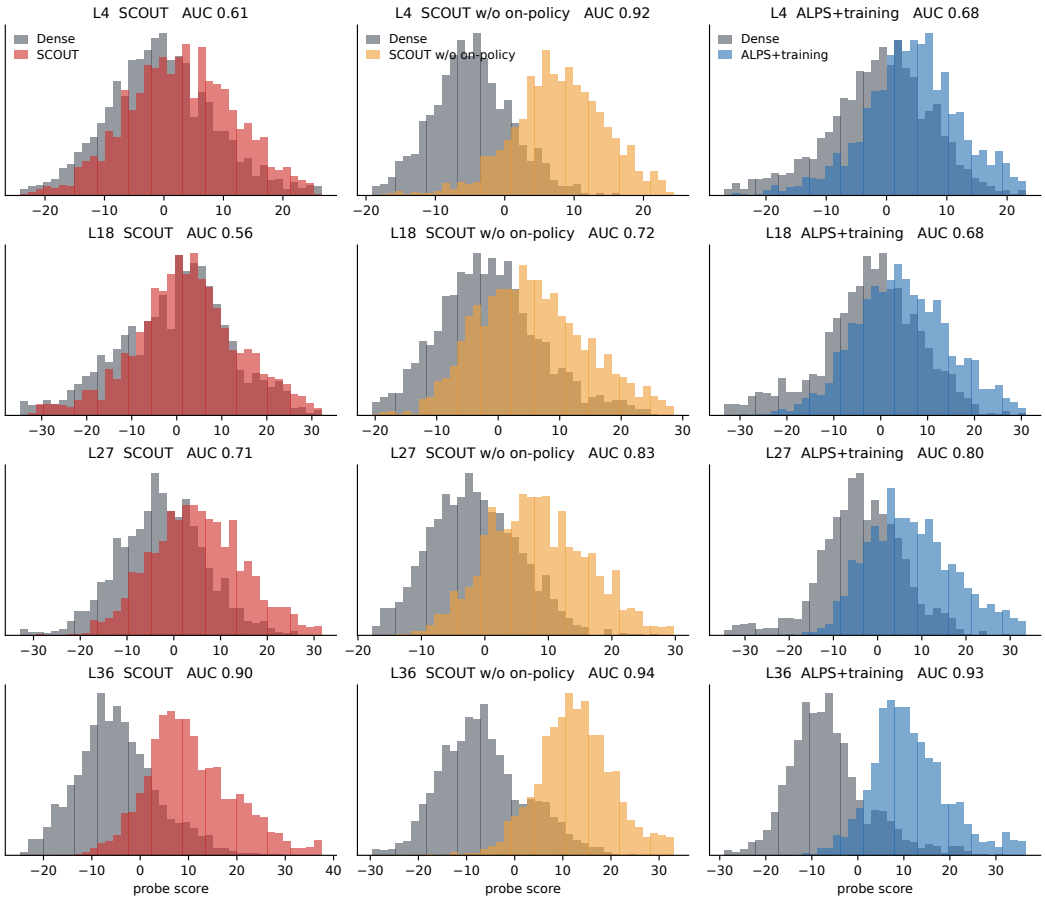


Figure 8: Out-of-fold probe-score histograms at four representative depths (rows) for SCOUT, SCOUT without on-policy information, and the fully trained ALPS mask, each against the dense model's rollouts. Each panel has its own probe and its own axis; only the overlap within a panel is meaningful.

D THEORETICAL ANALYSIS OF SALIENCY SCORE

In this section, we provide full theoretical analysis justifying the use of the saliency score for our problem setting.

D.1 SETUP

Throughout Section D, we denote $F_t(\theta) := \mathrm{KL}_{\mu^{Q_{t-1}}}(P \,\|\, Q_\theta)$. As in Section 3, $\widetilde{\theta}_t$ is the weight after applying S_{proj} Adam updates at iteration t. For a mask $m \in \{0,1\}^d$, define $\Delta_m := \widetilde{\theta}_t \odot m - \widetilde{\theta}_t = -\widetilde{\theta}_t \odot (\mathbf{1}_d - m)$. For our analysis, the following assumption is used.

Assumption D.1 (Smoothness). *F_t is differentiable and L-smooth on the line segment $\{\widetilde{\theta}_t + s\Delta_m : s \in [0,1]\}$.*

D.2 LEMMAS

Lemma D.2 (First-order expansion). *Under Assumption D.1,*

$$\left| F_t(\widetilde{\theta}_t \odot m) - F_t(\widetilde{\theta}_t) - \langle \nabla F_t(\widetilde{\theta}_t), \Delta_m \rangle \right| \le \frac{L}{2} \sum_{i:m_i=0} \widetilde{\theta}_{t,i}^2 \tag{15}$$

Proof. Let $x = \widetilde{\theta}_t$ and $y = \widetilde{\theta}_t \odot m = x + \Delta_m$. Applying the fundamental theorem of calculus to $s \mapsto F_t(x + s\Delta_m)$ gives

$$F_t(y) - F_t(x) = \int_0^1 \left\langle \nabla F_t(x + s\Delta_m),\, \Delta_m \right\rangle ds. \tag{16}$$

Subtracting $\langle \nabla F_t(x), \Delta_m \rangle = \int_0^1 \langle \nabla F_t(x), \Delta_m \rangle \, ds$ from both sides, we have

$$F_t(y) - F_t(x) - \langle \nabla F_t(x), \Delta_m \rangle = \int_0^1 \left\langle \nabla F_t(x + s\Delta_m) - \nabla F_t(x),\, \Delta_m \right\rangle ds. \tag{17}$$

By the Cauchy–Schwarz inequality and Assumption D.1,

$$\left| \left\langle \nabla F_t(x + s\Delta_m) - \nabla F_t(x),\, \Delta_m \right\rangle \right| \le \|\nabla F_t(x + s\Delta_m) - \nabla F_t(x)\| \, \|\Delta_m\| \le Ls \, \|\Delta_m\|^2. \tag{18}$$

Integrating over $s \in [0,1]$ yields the bound $\frac{L}{2}\|\Delta_m\|^2$. By definition of Δ_m, $\|\Delta_m\|^2 = \sum_{i:m_i=0} \widetilde{\theta}_{t,i}^2$ and we have the desired inequality. □

The Adam optimizer maintains the second-moment state

$$v_\tau = \beta_2 v_{\tau-1} + (1-\beta_2) g_\tau^2, \quad v_0 = 0 \tag{19}$$

where g_τ is the stochastic gradient of the weight update objective in Equation (5) and $\beta_2 \in (0,1)$. For Adam optimizer update, bias-corrected version of v_t, $\hat{v}_t := v_t/(1-\beta_2^\tau)$ is used.

Lemma D.3 (Unrolled second-moment). *For every $\tau \ge 1$,*

$$v_\tau = \sum_{i=1}^{\tau} \alpha_{\tau,i} \, g_i^2, \qquad \alpha_{\tau,i} := (1-\beta_2)\, \beta_2^{\tau-i}. \tag{20}$$

Proof. The proof uses mathematical induction over τ. For $\tau = 1$, $v_1 = (1-\beta_2)g_1^2 = \alpha_{1,1} g_1^2$ by Equation (19). Assume the claim holds for $\tau - 1$. Then

$$v_\tau = \beta_2 \sum_{i=1}^{\tau-1} (1-\beta_2)\beta_2^{\tau-1-i} g_i^2 + (1-\beta_2) g_\tau^2 = \sum_{i=1}^{\tau} (1-\beta_2)\beta_2^{\tau-i} g_i^2. \tag{21}$$

Therefore the claim also holds for τ. □

We also denote $A_\tau := \sum_{i=1}^{\tau} \alpha_{\tau,i} = 1 - \beta_2^\tau$ and define the normalized exponential moving average of a sequence $(x_i)_{i \le \tau}$ as

$$\mathrm{EMA}_\tau(x) := \sum_{i=1}^{\tau} w_{\tau,i}\, x_i, \qquad w_{\tau,i} := \frac{\alpha_{\tau,i}}{A_\tau}. \tag{22}$$

Note that the weights satisfy $w_{\tau,i} \ge 0$ and $\sum_{i=1}^{\tau} w_{\tau,i} = 1$. By Lemma D.3, $\mathrm{EMA}_\tau(g^2) = v_\tau / A_\tau = \hat{v}_\tau$. We write v_t and A_t as the values at the end of iteration t. We also define the gradient estimation error as $e_t := \nabla F_t(\widetilde{\theta}_t) - \hat{g}_t$ where $\hat{g}_t := \mathrm{EMA}_{\tau_t}(g)$.

Lemma D.4. *For $w_1, w_2, \ldots, w_n \ge 0$ such that $\sum_{i=1}^n w_i = 1$ and $x_1, x_2, \ldots, x_n \in \mathbb{R}$,*

$$\left| \sum_{i=1}^{n} w_i x_i \right| \le \sum_{i=1}^{n} w_i |x_i| \le \sqrt{\sum_{i=1}^{n} w_i x_i^2} \tag{23}$$

Proof. The first inequality holds by the triangle inequality. The second inequality is the direct consequence of applying Jensen's inequality for the concave function $x \mapsto \sqrt{x}$. □

Applying Lemma D.4 gives

$$|\hat{g}_{t,i}| = |\,\mathrm{EMA}(g_i)| \le \sqrt{\mathrm{EMA}_{\tau_t}.(g_i^2)} = \sqrt{\hat{v}_{t,i}} = A_t^{-1/2} \sqrt{v_{t,i}} \tag{24}$$

for every i.

D.3 Main Results

Define the saliency score of coordinate i at iteration t as

$$s_{t,i} := |\widetilde{\theta}_{t,i}| \sqrt{v_{t,i}}. \tag{25}$$

Theorem D.5. *Under Assumption D.1, for every mask m,*

$$\left|F_t(\widetilde{\theta}_t \odot m) - F_t(\widetilde{\theta}_t)\right| \le \underbrace{A_t^{-1/2} \sum_{i:m_i=0} s_{t,i}}_{\text{saliency term}} + \underbrace{\left|\langle e_t,\, \widetilde{\theta}_t \odot (\mathbf{1} - m)\rangle\right|}_{\text{gradient-estimation error}} + \underbrace{\frac{L}{2} \sum_{i:m_i=0} \widetilde{\theta}_{t,i}^2}_{\text{second-order remainder}}. \tag{26}$$

Proof. By Lemma D.2 and the triangle inequality,

$$\left|F_t(\widetilde{\theta}_t \odot m) - F_t(\widetilde{\theta}_t)\right| \le \left|\langle \nabla F_t(\widetilde{\theta}_t), \Delta_m \rangle\right| + \frac{L}{2} \sum_{i:m_i=0} \widetilde{\theta}_{t,i}^2. \tag{27}$$

Substituting $\nabla F_t(\widetilde{\theta}_t) = \hat{g}_t + e_t$ and $\Delta_m = -\widetilde{\theta}_t \odot (\mathbf{1} - m)$ into the term $\langle \nabla F_t(\widetilde{\theta}_t), \Delta_m \rangle$ and using the triangle inequality, we have

$$\left|\langle \nabla F_t(\widetilde{\theta}_t), \Delta_m \rangle\right| \le \left|\langle \hat{g}_t,\, \widetilde{\theta}_t \odot (\mathbf{1} - m)\rangle\right| + \left|\langle e_t,\, \widetilde{\theta}_t \odot (\mathbf{1} - m)\rangle\right|. \tag{28}$$

For the first term, by the triangle inequality and Equation (24),

$$\left|\langle \hat{g}_t,\, \widetilde{\theta}_t \odot (\mathbf{1} - m)\rangle\right| = \left| \sum_{i:m_i=0} \widetilde{\theta}_{t,i}\, \hat{g}_{t,i} \right| \le \sum_{i:m_i=0} |\widetilde{\theta}_{t,i}|\, |\hat{g}_{t,i}| \le A_t^{-1/2} \sum_{i:m_i=0} |\widetilde{\theta}_{t,i}| \sqrt{v_{t,i}}, \tag{29}$$

Combining the three bounds proves the desired inequality. □

This theorem is the full version of Equation (10) in Section 3 including gradient-estimation error term and second-order remainder term. Note that the factor $A_t^{-1/2}$ in Theorem D.5 is expressed as "$\lesssim$" in Equation (10). As $A_t^{-1/2}$ is the same for all coordinates, it does not affect the ordering of coordinates by $s_{t,i}$.

E Distributional Locality of Trust-Region Mask Updates

This section makes precise the intuition behind placing the trust region of Section 3 on the change in the model's predictive distribution rather than on the number of changed mask entries. The statements are meant to justify the *form* of the constraint, not to serve as numerical guarantees: as in trust-region policy optimization (Schulman et al., 2015), the bounds are loose at the radius and horizon used in practice, and δ is treated as a hyperparameter. Two facts are established. First, no bound on the fraction of changed mask entries can control the change in the output distribution (a single weight can move it arbitrarily), so a constraint in parameter space would not be meaningful. Second, a bound on the conditional KL between consecutive models does control the shift of the rollout distribution that the training objective is defined on, with a dependence on the generation horizon that indicates where such control matters most.

Throughout this subsection, total variation distance between distributions R and S on a common space is $D_{\mathrm{TV}}(R, S) = \frac{1}{2}\|R - S\|_1 = \sup_A |R(A) - S(A)|$.

Mask distance does not control distribution shift. There is no model-independent function $f(\kappa) \to 0$ as $\kappa \to 0$ such that changing at most a κ-fraction of support coordinates guarantees

$$D_{\mathrm{TV}}\left(Q_{\theta,m}(\cdot \mid h), Q_{\theta,m'}(\cdot \mid h)\right) \le f(\kappa) \tag{30}$$

uniformly over parameters, prefixes, and masks. Consider a softmax network with d prunable scalar coordinates and at least two output classes. Choose one active coordinate whose contribution changes one output logit by M, while all other changed coordinates have zero contribution. Removing this coordinate changes the output distribution from arbitrarily close to a point mass as $M \to \infty$ to a fixed non-degenerate distribution. Hence the total-variation change is bounded below by a positive constant independent of d, even though the changed support fraction is only $1/d \to 0$.

Conversely, changing a coordinate with an arbitrarily small contribution to all output logits induces an arbitrarily small change in the softmax distribution by continuity. Therefore, no model-independent function of support Hamming distance alone can uniformly control the change in the output distribution. This construction is generic and does not require pruning the language-model output head.

Notation. Let Q_t^- denote the sparse model immediately before the mask transition at projection event t and Q a candidate after it. For a prompt x, write $Q(\tau \mid x)$ for the distribution over length-H rollouts $\tau = (h_1, \ldots, h_H)$ and μ^Q for the prefix occupancy it induces (the average over positions $i \le H$ of the law of h_i). The population conditional KL between the two models on the pre-transition occupancy is

$$D_t^{\mathrm{func}}(Q) := \mathbb{E}_{x \sim \mathcal{D}_{\mathrm{prompt}}} \mathbb{E}_{h \sim \mu^{Q_t^-}} \left[D_{\mathrm{KL}}\big(Q_t^-(\cdot \mid h) \,\|\, Q(\cdot \mid h)\big) \right], \tag{31}$$

which the screened quantity $\widehat{D}_t(k)$ of Equation 13 approximates on the screening set $\mathcal{C}_t$ drawn from the current rollout cache.

Proposition E.1 (Conditional KL controls rollout shift)**.** *For two autoregressive models Q_t^- and Q over horizon H,*

$$\mathbb{E}_{x \sim \mathcal{D}_{\mathrm{prompt}}} D_{\mathrm{KL}}\left(Q_t^-(\tau \mid x) \,\|\, Q(\tau \mid x)\right) = H\, D_t^{\mathrm{func}}(Q). \tag{32}$$

Consequently, if $D_t^{\mathrm{func}}(Q) \le \delta$, their corresponding prefix occupancies satisfy

$$D_{\mathrm{TV}}\left(\mu^{Q_t^-}, \mu^Q\right) \le \sqrt{\frac{H\delta}{2}}, \tag{33}$$

where D_{TV} denotes total variation distance.

The equality follows from the autoregressive KL chain rule, so the conditional KL screened by SCOUT controls the shift of the rollout distribution that defines the local objective. The upper bound grows with the generation horizon H: under a fixed conditional-KL radius, the guarantee on rollout locality becomes weaker for longer generations, and maintaining a fixed bound on rollout shift as H grows would require $\delta = O(1/H)$.

Proof of Proposition E.1. For each prompt x, the autoregressive KL chain rule gives

$$D_{\mathrm{KL}}\left(Q_t^-(\tau \mid x) \,\|\, Q(\tau \mid x)\right) = \sum_{i=1}^{H} \mathbb{E}_{h_i \sim Q_t^-}\left[D_{\mathrm{KL}}\left(Q_t^-(\cdot \mid h_i) \,\|\, Q(\cdot \mid h_i)\right)\right]. \tag{34}$$

Averaging over $x \sim \mathcal{D}_{\mathrm{prompt}}$ and using the definition of the token-averaged pre-projection occupancy yields

$$\mathbb{E}_{x \sim \mathcal{D}_{\mathrm{prompt}}} D_{\mathrm{KL}}\left(Q_t^-(\tau \mid x) \,\|\, Q(\tau \mid x)\right) = H\, D_t^{\mathrm{func}}(Q), \tag{35}$$

which proves Equation 32.

For each prompt, Pinsker's inequality implies

$$D_{\mathrm{TV}}\left(Q_t^-(\tau \mid x), Q(\tau \mid x)\right) \le \sqrt{\frac{1}{2} D_{\mathrm{KL}}\left(Q_t^-(\tau \mid x) \,\|\, Q(\tau \mid x)\right)}. \tag{36}$$

Taking the prompt expectation and applying Jensen's inequality gives the trajectory-level bound

$$\mathbb{E}_{x \sim \mathcal{D}_{\mathrm{prompt}}} D_{\mathrm{TV}}\left(Q_t^-(\tau \mid x), Q(\tau \mid x)\right) \le \sqrt{\frac{H\delta}{2}} \tag{37}$$

whenever $D_t^{\mathrm{func}}(Q) \le \delta$.

Finally, for each position i, the prefix law is a marginal of the trajectory law. Total variation contracts under marginalization, and the occupancy μ^Q is an average of these prefix marginals. Convexity of total variation under averaging therefore yields Equation 33.

Derivation of the transition-shift bound. If $0 \le \ell(Q;h) \le L_{\max}$ and $D_t^{\mathrm{func}}(Q) \le \delta$, the variational characterization of total variation and Equation 33 give

$$\left|J_{\mu^Q}(Q) - J_{\mu^{Q_t^-}}(Q)\right| \le L_{\max} D_{\mathrm{TV}}\left(\mu^Q, \mu^{Q_t^-}\right) \le L_{\max}\sqrt{\frac{H\delta}{2}}, \tag{38}$$

where $\ell(Q;h) = D_{\mathrm{KL}}(P(\cdot \mid h) \| Q(\cdot \mid h))$ is the per-prefix conditional loss and $J_\rho(Q) := \mathbb{E}_{h\sim\rho}\,\ell(Q;h)$ is the on-policy objective of Equation 3 evaluated on the prefix distribution ρ.

Proposition E.2 (KL controls next-token events on the screening set)**.** *At a projection event, let $Q_t^{(k_t^\star)}$ denote the accepted candidate and define $D_t(h) = D_{\mathrm{KL}}(Q_t^-(\cdot \mid h) \| Q_t^{(k_t^\star)}(\cdot \mid h))$ for each $h \in \mathcal{C}_t$. For every next-token event $E \subseteq \mathcal{V}$,*

$$\left|Q_t^{(k_t^\star)}(E \mid h) - Q_t^-(E \mid h)\right| \le \sqrt{\frac{D_t(h)}{2}}.$$

Consequently, if $|\mathcal{C}_t|^{-1}\sum_{h\in\mathcal{C}_t} D_t(h) \le \delta$, then

$$\frac{1}{|\mathcal{C}_t|}\sum_{h\in\mathcal{C}_t} \sup_{E\subseteq\mathcal{V}} \left|Q_t^{(k_t^\star)}(E \mid h) - Q_t^-(E \mid h)\right| \le \sqrt{\frac{\delta}{2}}.$$

Proof of Proposition E.2. For a fixed prefix h and event $E \subseteq \mathcal{V}$, coarse-grain the vocabulary through the deterministic map

$$\mathcal{V} \to \{E, E^c\}.$$

By data processing,

$$D_t(h) \ge D_{\mathrm{KL}}\left(\mathrm{Ber}\left(Q_t^-(E \mid h)\right) \,\|\, \mathrm{Ber}\left(Q_t^{(k_t^\star)}(E \mid h)\right)\right). \tag{39}$$

Pinsker's inequality then gives

$$\left|Q_t^{(k_t^\star)}(E \mid h) - Q_t^-(E \mid h)\right| \le D_{\mathrm{TV}}\left(Q_t^-(\cdot \mid h), Q_t^{(k_t^\star)}(\cdot \mid h)\right) \tag{40}$$

$$\le \sqrt{\frac{D_t(h)}{2}}. \tag{41}$$

Taking the supremum over E, averaging over $\mathcal{C}_t$, and applying Jensen's inequality to the square root yields the second inequality in Proposition E.2. This is a deterministic statement on the finite screening set and does not require $\mathcal{C}_t$ to be an i.i.d. sample from a population.

F Convergence Analysis

F.1 Setup

In this section, we give full convergence analysis of the SCOUT method. We first define several notations. Here, we denote $w := \theta \odot m$ and set w as optimization variable. Also, the sparse model with weight $w := \theta \odot m$ will be written as $Q_w := Q_{\theta,m}$. The training objective is set to the OPD loss term $\mathcal{L}_{\text{OPD}}$, without considering the $\mathcal{L}_{\text{NTP}}$ and $\mathcal{L}_{\text{KD}}$. Defining μ^t as the prefix distribution of the sparse model used in step t to define $\mathcal{L}_t := \mathcal{L}_{\text{OPD},t}$ (index t added to reflect time-varying objective), we can write $\mathcal{L}_t(w) := \mathbb{E}_{h\sim\mu^t}\ell(w;h)$ where $\ell(w;h) := D_{\text{KL}}\left(P(\cdot \mid h) \,\|\, Q_w(\cdot \mid h)\right)$.

For simplicity, we assume that the loss is trained with gradient descent with step size $\eta > 0$. Therefore, at each update step t, $\tilde{\theta}_{t+1} = \theta_t \odot m_t - \eta\nabla\mathcal{L}_t(\theta_t \odot m_t)$, $\theta_{t+1} = \tilde{\theta}_{t+1} \odot m_{t+1}$. Since we use w instead of θ and m, we denote $w_t := \theta_t \odot m_t = \tilde{\theta}_t \odot m_t$ and $\tilde{w}_{t+1} := \tilde{\theta}_{t+1} \odot m_t$ when describing weight update.

For measuring convergence progress for sparsity constrained optimization problem, we resort to support-stationarity proposed in Beck & Eldar (2013), defined as follows.

Definition F.1 (support-stationarity)**.** *Let $\mathcal{L}$ be differentiable, $S \subseteq \{1,\dots,d\}$ with $|S| \le K$, and $w \in \mathbb{R}^d$ with $\text{supp}(w) := \{i : 1 \le i \le d, w_i \neq 0\} \subseteq S$. Then w is S-*stationary *for $\mathcal{L}$ if $\nabla_S\mathcal{L}(w) := \nabla\mathcal{L}(w) \odot \mathbf{1}_S = 0$ where $\mathbf{1}_S \in \{0,1\}^d$ is an indicator function of the set S.*

In the subsequent analysis, we denote $g_t := \nabla_{m_t}\mathcal{L}_t(w_t)$ where m_t is the pruning mask at step t and show that g_t gets small, converging to the neighborhood of support-stationary point.

Let $\mathcal{W} \subseteq \mathbb{R}^d$ be an open convex set containing all iterates w_t and $\tilde{w}_t$. The following assumptions are required on $\mathcal{W}$ only.

Assumption F.2 (Smoothness of training objective)**.** *$\ell(\cdot;h)$, for any prefix h, is L-smooth on $\mathcal{W}$. Consequently $\mathcal{L}_t$ is L-smooth on $\mathcal{W}$ for every t.*

Assumption F.3 (Relation between distributions induced by dense and sparse models)**.** *For distribution $P(\cdot|h)$ induced by dense model and $Q_w(\cdot|h)$ induced by sparse model, both conditioned on the prefix h, they satisfy $P(y|h) \le RQ_w(y|h)$ for some constant $R > 0$ and any $y \in \mathcal{V}$ and $w \in \mathcal{W}$.*

Assumption F.4 (Bound of loss and its gradient)**.** *For every $w \in \mathcal{W}$ and prefixes h, we have $0 \le \ell(w;h) \le B$ and $\|\nabla_w\ell(w;h)\| \le G$ for some constant $B, G > 0$.*

F.2 Lemmas.

Lemma F.5 (Descent Lemma)**.** *Under Assumption F.2, if $\eta \le 1/L$, then for every t,*

$$\mathcal{L}_t(\tilde{w}_{t+1}) \le \mathcal{L}_t(w_t) - \frac{\eta}{2}\|g_t\|^2 \tag{42}$$

Proof. From L-smoothness of $\mathcal{L}_t$, for any $x, y \in \mathcal{W}$, we have

$$\mathcal{L}_t(y) \le \mathcal{L}_t(x) + \langle\nabla\mathcal{L}_t(x), y - x\rangle + \frac{L}{2}\|y - x\|^2. \tag{43}$$

Also, observe that

$$\tilde{w}_{t+1} - w_t = (\tilde{\theta}_{t+1} - \theta_t) \odot m_t = -\eta(\nabla\mathcal{L}_t(w_t) \odot m_t) = -\eta g_t \tag{44}$$

and $\langle\nabla\mathcal{L}_t(w_t), g_t\rangle = \|g_t\|^2$. Plugging $y = \tilde{w}_{t+1}$ and $x = w_t$ to Equation (43), we have

$$\begin{aligned}
\mathcal{L}_t(\tilde{w}_{t+1}) &\le \mathcal{L}_t(w_t) + \langle\nabla\mathcal{L}_t(w_t), \tilde{w}_{t+1} - w_t\rangle + \frac{L}{2}\|\tilde{w}_{t+1} - w_t\|^2 \\
&= \mathcal{L}_t(w_t) + \left(-\eta + \frac{L\eta^2}{2}\right)\|g_t\|^2 \\
&\le \mathcal{L}_t(w_t) - \frac{\eta}{2}\|g_t\|^2
\end{aligned}$$

which is the desired result. □

Lemma F.6 (Cross-entropy change between adjacent sparse models)**.** *Let Q and Q' be two distributions on the finite set $\mathcal{V}$ with $Q(y) > 0$ and $Q'(y) > 0$ for all $y \in \mathcal{V}$. Let T be a distribution on $\mathcal{V}$ with $T(y) \le RQ(y)$ for all $y \in \mathcal{V}$ for some $R > 0$. Denote the cross entropy between two distributions P and Q on $\mathcal{V}$ as $H_{\mathrm{CE}}(P, Q) := -\sum_{y\in\mathcal{V}} P(y) \log Q(y)$. Then the following inequality holds.*

$$H_{\mathrm{CE}}(T, Q') - H_{\mathrm{CE}}(T, Q) \le R\left(D_{\mathrm{KL}}(Q\|Q') + \sqrt{\frac{1}{2}D_{\mathrm{KL}}(Q\|Q')}\right) \tag{45}$$

Proof. Write $r(y) := \log \frac{Q(y)}{Q'(y)}$. Then $H_{\mathrm{CE}}(T, Q') - H_{\mathrm{CE}}(T, Q) = \sum_{y\in\mathcal{V}} T(y)r(y)$. Drop the terms where $r(y) < 0$ and denoting $a_+ := \max(a, 0)$, we have

$$\sum_{y\in\mathcal{V}} T(y)r(y) \le \sum_{y\in\mathcal{V}} T(y)(r(y))_+ \le R\sum_{y\in\mathcal{V}} Q(y)(r(y))_+ = R\mathbb{E}_Q r_+. \tag{46}$$

Now since $r_+ = r + (-r)_+$, $\mathbb{E}_Q r_+ = \mathbb{E}_Q r + \mathbb{E}_Q(-r)_+$. Observe that $\mathbb{E}_Q r = D_{\mathrm{KL}}(Q\|Q')$ by definition. Also, since $\log x \le x - 1$, we have $(-r)_+ = (\log Q'/Q)_+ \le (Q'/Q - 1)_+$. Hence,

$$\begin{aligned}
\mathbb{E}_Q r_+ &\le D_{\mathrm{KL}}(Q\|Q') + \sum_{y\in\mathcal{V}} Q(y)\left(\frac{Q'(y)}{Q(y)} - 1\right)_+ \\
&= D_{\mathrm{KL}}(Q\|Q') + \sum_{y:Q'(y)>Q(y)} (Q'(y) - Q(y)) \\
&= D_{\mathrm{KL}}(Q\|Q') + D_{\mathrm{TV}}(Q, Q')
\end{aligned}$$

where $D_{\mathrm{TV}}(P, Q) := \frac{1}{2}\sum_h |P(h) - Q(h)|$ denotes the total variation (TV) distance between two discrete probability distributions P and Q. Note that the last equality holds because $\sum_{y\in\mathcal{V}}(Q'(y) - Q(y)) = 0$, so $\sum_{y:Q'(y)>Q(y)}(Q'(y) - Q(y)) = \frac{1}{2}\sum_{y\in\mathcal{V}} |Q'(y) - Q(y)|$. Now by Pinsker's inequality $D_{\mathrm{TV}}(Q, Q') \le \sqrt{\frac{1}{2}D_{\mathrm{KL}}(Q\|Q')}$, we have the final result. □

Corollary F.7 (Bound on the difference of OPD loss term during update)**.** *Define $\psi(\delta) := R(\delta + \sqrt{\delta/2})$. With Assumption F.3, we have $\mathcal{L}_t(w_{t+1}) - \mathcal{L}_t(\tilde{w}_{t+1}) \le \psi(\delta)$.*

Proof. We can rewrite KL divergence defining the OPD loss as $D_{\mathrm{KL}}(P(\cdot|h)\|Q_{\theta,m}(\cdot|h)) = H_{\mathrm{CE}}(P(\cdot|h), Q_{\theta,m}(\cdot|h)) - H(P(\cdot|h))$. Write $Q_{\tilde{w}_{t+1}} := Q_{\tilde{\theta}_{t+1},m_t}$ and $Q_{w_{t+1}} := Q_{\theta_{t+1},m_{t+1}}$, then by Lemma F.6, we have

$$\begin{aligned}
&D_{\mathrm{KL}}(P(\cdot|h)\|Q_{w_{t+1}}(\cdot|h)) - D_{\mathrm{KL}}(P(\cdot|h)\|Q_{\tilde{w}_{t+1}}(\cdot|h)) \\
&= H_{\mathrm{CE}}(P(\cdot|h), Q_{w_{t+1}}(\cdot|h)) - H_{\mathrm{CE}}(P(\cdot|h), Q_{\tilde{\theta}_{t+1}}(\cdot|h)) \\
&\le R\left(\mathrm{KL}_h + \sqrt{\frac{\mathrm{KL}_h}{2}}\right)
\end{aligned}$$

where $\mathrm{KL}_h := D_{\mathrm{KL}}(Q_{\tilde{w}_{t+1}}\|Q_{w_{t+1}})$. Taking expectation over h, $\mathbb{E}_h \mathrm{KL}_h \le \delta$ by the trust region condition. Also, by applying Jensen's inequality for concave function $x \mapsto \sqrt{x}$, we have $\mathbb{E}_h\sqrt{\frac{\mathrm{KL}_h}{2}} \le \sqrt{\mathbb{E}_h \mathrm{KL}_h /2} \le \sqrt{\delta/2}$. □

Lemma F.8 (Change of measure)**.** *Let P, Q be distributions on a countable set and f a function on it. If f is scalar with $\sup f - \inf f \le B$ then $|\mathbb{E}_P f - \mathbb{E}_Q f| \le BD_{\mathrm{TV}}(P, Q)$. If f is vector-valued with $\|f\| \le G$ then $\|\mathbb{E}_P f - \mathbb{E}_Q f\| \le 2GD_{\mathrm{TV}}(P, Q)$.*

Proof. Since $\sum_h(P(h) - Q(h)) = 0$, for any constant a, we have $\mathbb{E}_P f - \mathbb{E}_Q f = \sum_h(P(h) - Q(h))(f(h) - a)$. If f is scalar, take $a = (\sup f + \inf f)/2$, so that $f - a \le B/2$. Then

$$|\mathbb{E}_P f - \mathbb{E}_Q f| = \left|\sum_h (P(h) - Q(h))(f(h) - a)\right| \le \frac{B}{2}\sum_h |P(h) - Q(h)| = BD_{\mathrm{TV}}(P, Q). \tag{47}$$

If f is vector-valued, take $a = 0$. Then since $\|f\| \le G$,

$$\|\mathbb{E}_P f - \mathbb{E}_Q f\| = \left\| \sum_h (P(h) - Q(h)) f(h) \right\| \le G \sum_h |P(h) - Q(h)| = 2GD_{\mathrm{TV}}(P, Q). \quad (48)$$

□

Lemma F.9 (Bound induced by prefix distribution change)**.** *Under Assumption F.4, for all $w \in \mathcal{W}$, $|\mathcal{L}_{t+1}(w) - \mathcal{L}_t(w)| \le BD_{\mathrm{TV}}(\mu^{t+1}, \mu^t)$.*

Proof. By Assumption F.4, $\sup \ell(w; h) - \inf \ell(w; h) \le B$. Then using Lemma F.8, $|\mathcal{L}_{t+1}(w) - \mathcal{L}_t(w)| = |\mathbb{E}_{h\sim\mu^{t+1}} \ell(w; h) - \mathbb{E}_{h\sim\mu^t} \ell(w; h)| \le BD_{\mathrm{TV}}(\mu^{t+1}, \mu^t)$. □

Lemma F.10. *Let*

$$\bar{\beta} := \max_t \max\{D_{\mathrm{TV}}(\mu^t, \mu^{Q_{w_t}}), D_{\mathrm{TV}}(\mu^t, \mu^{Q_{w_{t+1}}})\}. \quad (49)$$

We can interpret $\bar{\beta}$ as staleness of the prefix distribution during its lifetime. Then at every prefix distribution refresh step t, $\mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_{t+1}) \le B\bar{\beta}$.

Proof. At prefix distribution refresh step t, $\mu^{t+1} = \mu^{Q_{w_{t+1}}}$, so by Lemma F.9, $\mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_{t+1}) \le BD_{\mathrm{TV}}(\mu^t, \mu^{Q_{w_{t+1}}}) \le B\bar{\beta}$. □

F.3 Main Results

Start with the following decomposition

$$\mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_t) = \underbrace{\mathcal{L}_t(\tilde{w}_{t+1}) - \mathcal{L}_t(w_t)}_{\text{weight update}} + \underbrace{\mathcal{L}_t(w_{t+1}) - \mathcal{L}_t(\tilde{w}_{t+1})}_{\Delta_t\text{: mask update}} + \underbrace{\mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_{t+1})}_{\xi_t\text{: rollout cache update}}. \quad (50)$$

Under Assumption F.2 and for $\eta \le 1/L$, by Lemma F.5 we have

$$\begin{aligned} \mathcal{L}_{t+1}(w_{t+1}) - \mathcal{L}_t(w_t) &= \mathcal{L}_t(\tilde{w}_{t+1}) - \mathcal{L}_t(w_t) + \Delta_t + \xi_t \\ &\le -\frac{\eta}{2}\|g_t\|^2 + \Delta_t + \xi_t \end{aligned}$$

Taking summation from $t = 0$ to $t = T - 1$, we have

$$\mathcal{L}_T(w_T) - \mathcal{L}_0(w_0) \le -\frac{\eta}{2} \sum_{t=0}^{T-1} \|g_t\|^2 + \sum_{t=0}^{T-1} (\Delta_t)_+ + \sum_{t=0}^{T-1} (\xi_t)_+ \quad (51)$$

or

$$\frac{1}{T} \sum_{t=0}^{T-1} \|g_t\|^2 \le \frac{2}{\eta} \left[\frac{\mathcal{L}_0(w_0)}{T} + \frac{1}{T} \sum_{t=0}^{T-1} (\Delta_t)_+ + \frac{1}{T} \sum_{t=0}^{T-1} (\xi_t)_+ \right] \quad (52)$$

since $\mathcal{L}_T(w_T) \ge 0$. Note that $\Delta_t = 0$ unless mask update (occurs every S_{proj} steps) happens at step t and $\xi_t = 0$ unless rollout cache update (occurs every S_{ref} steps) happens at step t. In each mask update step, $\Delta_t \le \psi(\delta)$ by Corollary F.7. Also, in every rollout cache update step $\xi_t \le B\bar{\beta}$. Since there are at most T/S_{proj} mask updates and T/S_{ref} rollout cache updates, we have the following bound:

$$\frac{1}{T} \sum_{t=0}^{T-1} \|g_t\|^2 \le \frac{2\mathcal{L}_0(w_0)}{\eta T} + \frac{2\psi(\delta)}{\eta S_{\mathrm{proj}}} + \frac{2B\bar{\beta}}{\eta S_{\mathrm{ref}}} \quad (53)$$

Summing up, we have the following result.

Theorem F.11. *Under Assumption F.2, Assumption F.3, Assumption F.4, the following bound holds:*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|g_t\|^2 \le \frac{2\mathcal{L}_0(w_0)}{\eta T} + \frac{2\psi(\delta)}{\eta S_{\mathrm{proj}}} + \frac{2B\bar{\beta}}{\eta S_{\mathrm{ref}}} \quad (54)$$