SCALING WHERE IT MATTERS: PREDICTIVE CONTRACTION FOR EFFICIENT INTERNAL SCALING IN LMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling language models with larger networks or more inference trajectories improves performance but significantly increases computational load. This raises an alternative: can extra computation be allocated selectively inside the Transformer, rather than globally, to improve performance with modest overhead? Our core idea is that prediction-relevant refinement is highly non-uniform across layers. Using a fixed logit lens, we observe sharp predictive entropy contraction at a few layers and treat it as an observable signal of strong prediction-relevant refinement. We call such layers predictive concentration layers. Based on this signal, EntroBranch performs efficient localized scaling by selecting the layer and branch width from frozen-backbone calibration, adding prefix-conditioned branches there, and aggregating their outputs with a token-wise gate before resuming single-path computation. Across Qwen3 and Gemma4 on code, mathematics, and multiple-choice tasks, EntroBranch improves overall scores by 33.70% on average over single-path prefix tuning with only 4.58% additional per-token inference FLOPs.

1 INTRODUCTION

Scaling has been a primary driver of progress in autoregressive language models such as GPT (Singh et al., 2026), Qwen (Yang et al., 2025), and DeepSeek (DeepSeek-AI et al., 2026). As illustrated in Figure 1, existing scaling strategies primarily scale computation in two ways. Parameter scaling increases model capacity by enlarging the network, improving the capability of each forward pass but also significantly increasing its computational and memory cost (Hoffmann et al., 2022; Kaplan et al., 2020). Inference-time scaling instead allocates additional computation to multiple decoding trajectories (Wang et al., 2023; Yao et al., 2023), for example through beam search (Sutskever et al., 2014) or sampling (Graves, 2013). This repeatedly invokes full-model computation to explore alternative predictions. These approaches motivate a question: **must additional computation be applied uniformly across all layers of the model or repeatedly across trajectories, or can it be allocated selectively inside the Transformer where it can operate on internal computations?**

To determine where additional internal computation may be most useful, we first examine how the model's predictive state evolves across Transformer layers using a fixed logit lens (nostalgebraist, 2020). We use it as a layerwise diagnostic, not as calibrated intermediate predictions. We find that predictive entropy changes non-uniformly: a few layers produce exceptionally sharp contractions, where the decoded predictive distribution becomes substantially more concentrated. Prior work suggests that model predictions are progressively constructed and refined across Transformer layers (Geva et al., 2022). We use *prediction-relevant refinement* to refer to layerwise updates that sharpen the decoded next-token predictive distribution. Predictive entropy contraction provides an observable diagnostic of the strength of this refinement: a larger contraction indicates stronger prediction-relevant refinement under the fixed logit lens. We call layers with sharp predictive entropy contraction *predictive concentration layers*.

The observation of predictive concentration layers suggests a different view of scaling. A sharp predictive entropy contraction identifies a layer exhibiting strong prediction-relevant refinement under the fixed logit lens. Rather than scaling all layers uniformly, these layers provide natural locations for localized scaling, allowing additional computation to be allocated where the model most strongly refines its predictive distribution. The layerwise contraction profile therefore signals *where* to scale. Its magnitude can further guide the branch width, providing a practical rule for *how much* additional

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.

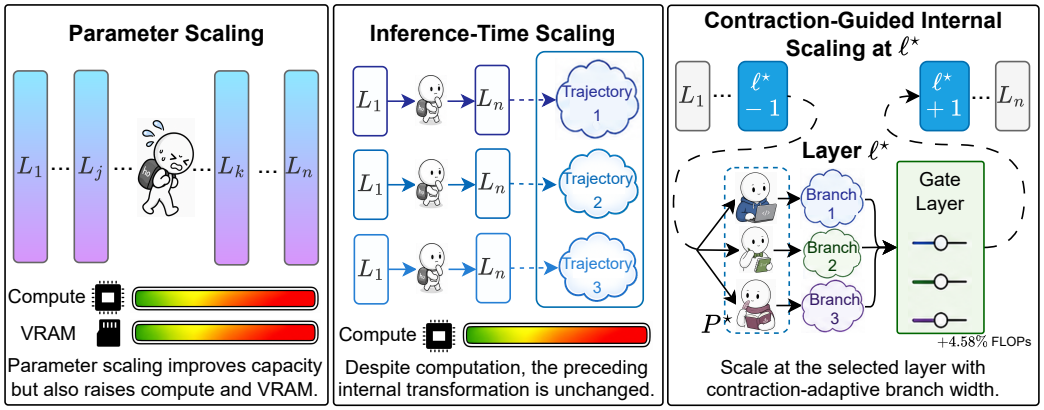


Figure 1: Different locations for additional computation. Parameter scaling expands model capacity globally, while inference-time scaling explores multiple output trajectories. EntroBranch instead branches temporarily at the predictive concentration layer $\ell^\star$, aggregates the branches token-wise, and then resumes single-path computation.

computation to allocate. Since wider branching increases computation roughly linearly, this avoids always using the maximum supported width. We also evaluate the correlation between predictive concentration and performance, as well as the effect of larger branch widths, in Section 4.5.

This motivates localized scaling at the predictive concentration layer. We therefore propose **EntroBranch** (Entropy-Guided Branching). At this layer, EntroBranch processes the same incoming hidden state through multiple prefix-conditioned branches, aggregates their outputs into a single hidden state, and then resumes single-path computation.

However, EntroBranch faces two challenges. The first challenge is determining where and how much to scale. EntroBranch calibrates the frozen pretrained backbone before adaptation by computing top-k logit-lens entropy across layers and selecting the layer with the largest average relative predictive entropy contraction; this layer is then fixed during tuning. At the selected layer, a robust statistic of token-wise contraction determines the branch width under a configurable hardware maximum.

The second challenge is realizing localized scaling efficiently. At the selected layer, EntroBranch applies branch-specific prefix tuning, producing multiple branch outputs from the same frozen layer. A token-wise gate aggregates these outputs before downstream layers immediately resume single-path computation. Because all branches share the same incoming hidden state, their content keys and values use a shared KV cache, avoiding persistent branch-wise duplication. This confines the additional computation and memory overhead to the selected layer.

We evaluate EntroBranch on Qwen3 and Gemma4 across code, mathematics, and choice tasks. Averaged across four models, EntroBranch improves the overall score by 33.70% over single-path prefix tuning with only 4.58% additional per-token inference FLOPs, while maintaining nearly unchanged inference throughput and VRAM usage. Its contraction-guided layer and width allocation is also stable across calibration settings and entropy estimators. EntroBranch further outperforms the evaluated inference-scaling baselines despite their roughly doubled FLOPs, and exceeds LoRA and other parameter-matched fine-tuning methods on Qwen3-0.6B-Base.

Our main contributions are as follows:

- We identify non-uniform predictive entropy contraction across Transformer layers shown in Figure 3 and define layers with sharp contraction as predictive concentration layers, using contraction as a layerwise diagnostic of prediction-relevant refinement for localized scaling.
- We propose EntroBranch, an efficient internal scaling method combining contraction-guided layer and width selection with branch-specific prefix tuning, token-wise aggregation, and shared KV caching, and derive sufficient conditions for local loss reduction through branch reweighting.
- Experiments across models and tasks demonstrate substantial performance gains with modest overhead, supported by ablations of layer selection, branch width, and aggregation design.

2 Related Work

2.1 Parameter and Inference-Time Scaling

Scaling can enhance the model's capabilities. Parameter scaling (Hoffmann et al., 2022; Kaplan et al., 2020) increases model dimensions for higher performance, while methods such as beam search (Sutskever et al., 2014) and temperature sampling (Graves, 2013) boost performance by extending inference time. ParScale (Chen et al., 2025) further generates multiple reasoning trajectories and adaptively aggregates the results. However, scaling methods substantially increase computational load and deployment costs, thereby limiting their application scenarios (Sheng et al., 2023). Inference scaling also faces performance bottlenecks (Aggarwal et al., 2023; Yang et al., 2018). Therefore, a method is needed that can improve performance without significantly increasing computational load. **We argue that efficient scaling should allocate computation selectively within the network.**

2.2 Parameter-Efficient Adaptation and Gated Mixtures

Prefix-tuning (Li & Liang, 2021) and LoRA (Hu et al., 2022) achieve efficient fine-tuning by adapting the backbone using few trainable parameters. In contrast, SMoP (Choi et al., 2023) trains multiple prompts and employs a MoE-like routing mechanism to select the appropriate prompt for generation. These methods adapt models to diverse tasks while reducing training costs. Despite their parameter efficiency, these methods realize adaptation through lightweight parameter updates or prompt modules, providing limited means to increase task-specific computation within the network (Liu et al., 2026). **We argue that adaptation should add computation where predictive refinement matters.**

3 Method

EntroBranch implements a localized scaling principle: instead of increasing model size throughout the network or duplicating full inference trajectories, it allocates additional computation to the layer with the strongest prediction-relevant refinement. Prior work views Transformer predictions as progressively refined across layers (Geva et al., 2022); under a fixed logit lens, a sharp entropy contraction provides an observable signal of a pronounced refinement in the evolving predictive state. As illustrated in Figure 2, EntroBranch identifies this layer from calibration data, assigns its branch width according to contraction magnitude, and scales computation at the selected layer with parallel prefix-tuned branches while sharing the backbone KV cache to limit inference overhead.

3.1 Entropy-Guided Layer Selection and Contraction-Guided Branch

Our layer-selection rule operationalizes the scaling motivation above. If predictions are refined across layers, then a sharp fixed-lens entropy contraction marks a layer at which the predictive state becomes substantially more concentrated. We use contraction magnitude as a diagnostic of prediction-relevant refinement: a larger contraction indicates a stronger layerwise refinement under the fixed logit lens. We therefore select the layer with the largest relative contraction as the location for localized scaling.

We sample a calibration set Ω from the task-specific training data and freeze the model. Using the logit lens (nostalgebraist, 2020; Chuang et al., 2024), where ℓ indexes Transformer layers and t token positions, let $\mathbf{x}_{\ell,t} \in \mathbb{R}^d$ be its hidden state (d is the hidden size), $\mathbf{z}_{\ell,t} = \mathbf{W}_{\mathrm{U}}\mathcal{N}(\mathbf{x}_{\ell,t})$ the logit-lens logits, and $\boldsymbol{\pi}^{(k)}_{\ell,t}$ the softmax restricted to their top $k = 64$ vocabulary entries, where $\mathcal{N}$ and $\mathbf{W}_{\mathrm{U}}$ are the frozen final normalization and unembedding. Let $\mathsf{H}^{(k)}_{\ell,t}$ denote the entropy of $\boldsymbol{\pi}^{(k)}_{\ell,t}$ and $\delta_{\ell,t} = [\mathsf{H}^{(k)}_{\ell-1,t} - \mathsf{H}^{(k)}_{\ell,t}]_+$ the positive entropy contraction, with $[u]_+ = \max(u, 0)$. We select

$$\ell^\star = \arg\max_{\ell} s_\ell, \qquad s_\ell = \frac{1}{|\mathcal{T}_\Omega|} \sum_{(x,t)\in\mathcal{T}_\Omega} \frac{\delta_{\ell,t}}{\mathsf{H}^{(k)}_{\ell-1,t} + \epsilon}, \tag{1}$$

where x denotes a calibration sequence, $\mathcal{T}_\Omega = \{(x, t)\}$ contains its valid next-token positions across Ω, s_ℓ is the mean relative entropy-contraction score of layer ℓ, and $\ell^\star$ is its maximizer. We term $\ell^\star$ the *predictive concentration layer*; it is the pre-adaptation layer selected for the strongest average prediction-relevant refinement and is fixed before adaptation. Figure 3 shows this localization across models; detailed entropies and top-k robustness are in Appendices E and G.

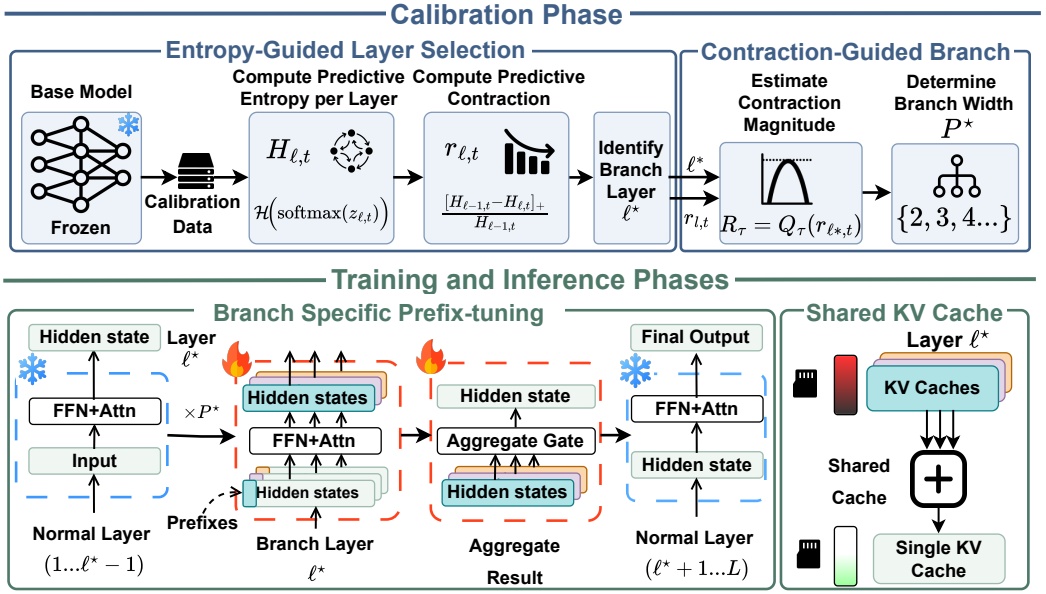
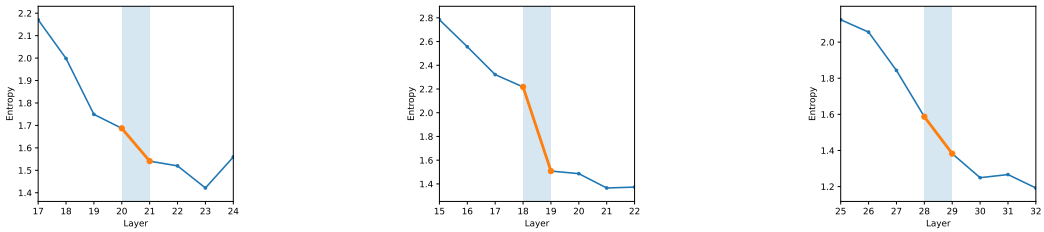


Figure 2: Overview of EntroBranch.



(a) Qwen3-0.6B (b) Qwen3-1.7B (c) Qwen3-4B

Figure 3: Layer-wise predictive entropy around the predictive concentration layers across models.

The layer score determines *where* to scale, while contraction magnitude guides *how many* branches to allocate. Since useful branch width depends on learned branch outputs and is unavailable before adaptation, while wider branching increases computation, we use contraction magnitude as a pre-adaptation heuristic to allocate additional branches only when stronger prediction-relevant refinement warrants the extra cost. Motivated by the effective number e^H (Jost, 2006), we set

$$\rho_\alpha = \text{Quantile}_\alpha\{\delta_{\ell^\star,t} : (x,t) \in \mathcal{T}_\Omega\}, \qquad P^\star = \lceil e^{\eta\rho_\alpha} \rceil_{\mathcal{Q}_1}, \quad 0 < \eta \le 1, \tag{2}$$

where $\alpha \in (0,1)$ is the quantile level, ρ_α is the corresponding contraction quantile at $\ell^\star$, $\mathcal{Q}$ is the set of supported multi-branch widths, $\mathcal{Q}_1 = \{1\} \cup \mathcal{Q}$, $P_{\max} = \max \mathcal{Q}_1$, and $\lceil \cdot \rceil_{\mathcal{Q}_1}$ rounds upward within this set. The factor η controls the compute–width trade-off, and $P^\star$ is the selected number of branches; $(\ell^\star, P^\star)$ is then fixed. Additional width analysis is provided in the appendix; both selections are stable across calibration seeds, while width allocation remains stable under nearby choices of α and η (Appendices F and I).

3.2 Branch-Specific Prefix-Tuning

Having located the predictive concentration layer, we scale it locally with prefix-tuned branches while keeping the backbone frozen. All branches share the incoming hidden state and Transformer-layer parameters but use independent prefixes, producing branch-specific hidden states (Li & Liang, 2021). For branch $j \in \{1, \dots, P^\star\}$, a trainable generator f_{θ_j} produces $\mathbf{E}^{(j)} = f_{\theta_j}(\mathbf{U}^{(j)}) \in \mathbb{R}^{T_j \times d}$.

Prediction-relevant refinement can vary across tokens, so a fixed branch mixture would apply the same scaling decision to every token. A token-wise gate instead allocates branch capacity adaptively, allowing each token to use the branch combination useful for the downstream objective. For branch j, $\mathbf{X}_{\ell^\star}^{(j)} = F_{\ell^\star}(\mathbf{X}_{\ell^\star-1}; \mathbf{E}^{(j)})$. At token t, the gate computes $\mathbf{a}_t = \mathrm{softmax}(\Gamma_\phi([\mathbf{x}_{\ell^\star,t}^{(1)}; \ldots; \mathbf{x}_{\ell^\star,t}^{(P^\star)}]))$ and aggregates $\mathbf{x}_{\ell^\star,t} = \sum_{j=1}^{P^\star} a_{t,j}\mathbf{x}_{\ell^\star,t}^{(j)}$. Here $F_{\ell^\star}$ is the frozen selected layer and $\mathbf{a}_t$ contains token-wise branch weights; only $\{\theta_j\}$ and ϕ are trained. Branches are then merged, so subsequent layers resume single-path computation while the additional computation remains localized to the selected layer. For L comparable Transformer layers, the added computation is approximately $(P^\star - 1)/L$ of a single-path forward pass, excluding the lightweight gate cost (Shazeer et al., 2017).

Localized scaling at the predictive concentration layer adds computation by producing multiple branch-specific hidden states from the same layer input. These states provide different directions in representation space, while the token-wise gate selects how they are combined for the downstream objective. Thus, increasing branch width expands the set of hidden states available at the selected layer, giving the model additional flexibility precisely where prediction-relevant refinement is strongest.

For a fixed token, let $P = P^\star$, $\bar{\mathbf{x}} = P^{-1}\sum_j \mathbf{x}^{(j)}$, and $\mathcal{S} = \mathrm{span}\{\mathbf{x}^{(j)} - \bar{\mathbf{x}}\}_{j=1}^{P}$, where $\mathcal{S}$ is the subspace spanned by branch differences and $\mathbf{\Pi}_\mathcal{S}$ its orthogonal projector. The following result characterizes when this additional branch capacity can support a local reduction of the loss.

Theorem 1 (Local correction through branch aggregation)**.** *Let $\mathbf{x} = \sum_j a_j\mathbf{x}^{(j)}$ be the aggregated hidden state, where $a_j > 0$ and $\sum_j a_j = 1$, and let $\mathcal{L}$ be the downstream supervised loss. If $\mathbf{\Pi}_\mathcal{S}\nabla_\mathbf{x}\mathcal{L}(\mathbf{x}) \neq 0$ and $\mathcal{L}$ has a Lipschitz gradient near the convex hull of the branch outputs, then a sufficiently small feasible reweighting of the branch weights strictly decreases $\mathcal{L}$.*

The theorem links localized scaling to downstream optimization: when branch directions overlap with the loss gradient, the gate can move the aggregated hidden state along a descent direction. Thus, wider branching adds both computation and useful directions at the predictive concentration layer. Proofs are in Appendices A.1–A.3; Section 4.8 evaluates this geometry empirically.

3.3 Shared KV Cache

Separate KV caches would multiply the branched layer's memory. Since all branches share the same pre-prefix hidden state $\mathbf{X}_{\ell^\star-1}$, their content K/V are highly redundant. Motivated by prior works (Shazeer, 2019; Liu et al., 2024; Ainslie et al., 2023), We therefore share $\overline{\mathbf{Z}}_c = \frac{1}{P^\star}\sum_{j=1}^{P^\star}\mathbf{Z}_c^{(j)}$ for $\mathbf{Z} \in \{\mathbf{K}, \mathbf{V}\}$ while keeping prefix K/V branch-specific. Each branch attends to the shared content K/V and its own prefix K/V, reducing persistent content-cache storage by approximately a factor of $P^\star$ while preserving branch-specific conditioning. Downstream effects are evaluated in Section 4.7.

4 Evaluation

4.1 Research Questions

We conduct a comprehensive evaluation of EntroBranch from several complementary perspectives. First, we evaluate its overall effectiveness across model families, scales, and task domains, comparing it with single-path prefix tuning and inference-scaling baselines. We then examine its performance–computation trade-off to determine whether localized branching can improve predictive performance with limited additional inference cost. To validate the contraction-guided allocation strategy, we study the effects of branch width and branch location, together with alternative allocation and aggregation designs. We further compare EntroBranch with parameter-efficient fine-tuning methods under both parameter-matched and larger-parameter settings. Finally, we analyze the memory behavior of the shared KV cache and investigate the mechanism underlying EntroBranch through predictive contraction, branch differentiation, token-wise gating, and gradient alignment.

4.2 Experimental Setup

We evaluate EntroBranch on decoder-only models from two families and multiple scales, including base and instruct models: Qwen3-0.6B/1.7B/4B-Base (Yang et al., 2025) and Gemma 4-E4B-it (Team

Table 1: Datasets for experiments Welbl et al. (2017); Zellers et al. (2019); Chen et al. (2021); Liu et al. (2023); Austin et al. (2021); Cobbe et al. (2021); Lightman et al. (2023); Wang et al. (2024); Jain et al. (2024); AI-MO Team (2024); Zhuo et al. (2024); Dekoninck et al. (2026); Zhang (2025)

Model	Choice	Code	Math
Qwen3-0.6B-Base	SciQ, HellaSwag	HumanEval (+), MBPP(+)	GSM8K, Math500
Qwen3-1.7B-Base	MMLU-Pro, HellaSwag	HumanEval (+), MBPP(+), LiveCodeBench v6	GSM8K, Math500, AMC
Qwen3-4B-Base	MMLU-Pro, HellaSwag	LiveCodeBench v6, BigCodeBench hard	AMC, AIME 22–26, Math500
Gemma-E4B	MMLU-Pro, HellaSwag	HumanEval (+), MBPP(+), LiveCodeBench v6	GSM8K, Math500

et al., 2026), assessing generalization across model size and architecture. Unless stated otherwise, the backbone is frozen, and only the prefix generators and token-wise gate are trained.

We train on 220K randomly sampled examples from multiple code and math datasets, which involve complex reasoning aligned with EntroBranch's objectives (Gu et al., 2024; Yu et al., 2025; Zheng et al., 2023; Hendrycks et al., 2021). Dataset composition is detailed in Appendix C, and all baselines use the same training data. We use a two-layer MLP to generate prefixes.

Given capability differences across model scales and architectures, we adopt a capability-matched evaluation protocol (Zhu et al., 2024), following LoRA (Hu et al., 2022) by assigning appropriately difficult coding, math, and choice subsets to each model group. Datasets are listed in Table 1.

We compare against single-path prefix-tuning (Li & Liang, 2021) and scaling methods. Scaling methods are beam search (Sutskever et al., 2014), temperature sampling (Graves, 2013), and ParScale (Chen et al., 2025). Beam search and temperature sampling use the single-path trained model, with beam size 2 and temperature 0.7; all other temperatures are 0. We use Pass@2 for temperature sampling and Pass@1 for the rest. Detailed training settings are in Appendix C.3, and experiment settings are in Appendix C.4. Pass@1s of temperature sampling are in Appendix B.

4.3 Main Results

Table 2 summarizes the domain-level results; per-dataset scores are provided in Appendix B.

EntroBranch achieves the highest overall score across all four models. Relative to single-path p-tuning, it improves the overall score by 30.99–35.88%, with an average gain of 33.70%. The gains span Choice, Code, and Math; notably, Choice performance improves across all four models despite training only on code and mathematics data, indicating transfer beyond the supervised domains.

EntroBranch also consistently outperforms the evaluated inference-scaling baselines. Compared with ParScale, it improves the overall score by 19.62–28.16% across models, averaging 23.04%, with similar advantages over beam search and temperature sampling.

Taken together, these results suggest that EntroBranch benefits from where additional computation is allocated rather than from compute volume alone. Its gains persist across model scales and architectures while exceeding baselines that use substantially more computation, supporting selective expansion at a contraction-selected internal transition as an effective compute-allocation strategy.

4.4 Efficiency

Compared to single-path, EntroBranch increases compute by just 4.58%. Compared with baselines that expand search or duplicate inference computation, EntroBranch offers a favorable performance–computation trade-off while using roughly half their FLOPs.

To test real-world inference throughput, we measure token generation speed with batch size 64, averaging over 20 runs. The single-path method achieves 1345.39 tokens/s, while EntroBranch achieves 1340.05 tokens/s, showing that the localized branch introduces little throughput loss.

Table 2: Main results. Choice, Code, and Math report the average performance over their sub-datasets; per-dataset results are given in Appendix Tables 4–7. Δ_{EB} is the relative improvement of EntroBranch's Overall average over the method in that row; FLOPs represents the number of FLOPs generated relative to the single path method when generating a token; Δ FLOPs is the compute increase over Single Path. FLOPs are estimated under the comparable-block-cost approximation.

Method	Choice	Code	Math	Overall	Δ_{EB} (%)	FLOPs	Δ FLOPs (%)
Qwen3-0.6B-Base							
Single Path (p=1)	58.02	24.95	31.65	38.21	+33.00	1.00	–
ParScale (p=2)	58.70	31.30	33.04	41.01	+23.92	2.00	+100
Beam Search	58.02	38.38	33.52	43.30	+17.37	1.99	+99
Sampling (T=0.7)	58.02	26.55	38.62	41.06	+23.77	2.00	+100
EntroBranch (ours)	**60.64**	**43.55**	**48.26**	**50.82**	–	**1.04**	+3.6
Qwen3-1.7B-Base							
Single Path (p=1)	25.46	48.27	34.22	35.98	+34.91	1.00	–
ParScale (p=2)	31.96	48.68	41.09	40.58	+19.62	2.00	+100
Beam Search	25.46	47.37	37.51	36.78	+31.97	1.99	+99
Sampling (T=0.7)	25.46	53.87	44.03	41.12	+18.04	2.00	+100
EntroBranch (ours)	**42.50**	**54.88**	**48.23**	**48.54**	–	**1.07**	+7.1
Qwen3-4B-Base							
Single Path (p=1)	43.85	14.63	26.02	28.17	+30.99	1.00	–
ParScale (p=2)	49.07	14.53	28.29	30.63	+20.47	2.00	+100
Beam Search	43.85	16.57	24.61	28.34	+30.20	1.99	+99
Sampling (T=0.7)	43.85	17.80	29.68	30.44	+21.22	2.00	+100
EntroBranch (ours)	**60.76**	**19.57**	**30.36**	**36.90**	–	**1.03**	+2.8
Gemma4-E4B							
Single Path (p=1)	37.22	28.25	32.87	32.78	+35.88	1.00	–
ParScale (p=2)	40.32	29.17	34.78	34.76	+28.16	2.00	+100
Beam Search	37.22	24.04	37.69	32.98	+35.05	1.99	+99
Sampling (T=0.7)	37.22	34.75	33.70	35.22	+26.46	2.00	+100
EntroBranch (ours)	**49.25**	**42.02**	**42.36**	**44.54**	–	**1.05**	+4.8

4.5 Ablation Studies

Branch width: We compare the calibrated width with smaller and larger alternatives in Table 3a. Performance drops when branching is insufficient, while further widening beyond $P^\star$ brings little consistent gain, supporting contraction-guided allocation as a signal of breadth. Adaptive allocation recovers most of the performance of wider fixed branching while avoiding unnecessary computation.

Branch layer: Table 3d compares the entropy-selected layer with a randomly selected layer and shows better performance for entropy-guided selection. We examine nearby branch locations by shifting the intervention to layers 22 and 23, downstream of the selected contraction layer. As shown in Table 3b, both alternatives perform worse than branching at the EntroBranch-selected layer. The Spearman rank correlation of contraction and downstream performance is $\rho = 0.7182$ ($p = 0.0128$). For Qwen3-1.7B-Base, we observe a similar correlation, $\rho = 0.8167$ ($p = 0.0072$) and layer $\ell^\star$ outperforms others. The selected layer also outperforms nearby layers, indicating that placement within the layer region matters and supporting the importance of branching at the predictive-contraction layer. Full layer-wise and Qwen3-1.7B's results are provided in Appendix D.

To test whether one branching is sufficient, Table 3c compares single-layer and multi-layer branching. The results show that multi-layer branching offers no advantage over single-layer branching, likely because coupled gates complicate credit assignment under the same downstream objective.

Table 3: Ablation studies of EntroBranch, where **EB** represents EntroBranch. (a) Branch width. (b) More branch layers. (c) Multi-layer branching, Mean gate, and independent KV cache. (d) Random-layer selection. (e) Comparison with other fine-tuning methods on Qwen3-0.6B-Base. (f) Layer-wise comparison of entropy descent. (g) Training loss of tuning methods.

(a) Branch width.

Width	Choice	Code	Math	Avg	FLOP
Qwen3-0.6B-Base					
$P^\star$ (2)	60.64	43.55	48.26	50.82	1.04
P=4	61.89	41.93	46.56	50.12	1.11
P=6	61.55	43.10	46.63	50.43	1.18
P=8	61.89	43.53	47.86	51.09	1.25
Qwen3-1.7B-Base					
$P^\star$ (3)	42.50	54.88	48.23	48.54	1.07
P=2	40.57	51.91	44.12	45.53	1.04
P=4	42.42	53.84	46.48	47.58	1.11
P=6	40.38	54.64	49.16	48.06	1.18
P=8	41.78	54.61	47.32	47.90	1.25
Qwen3-4B-Base					
$P^\star$ (2)	60.76	19.57	30.36	36.90	1.03
P=4	60.54	18.55	28.88	35.99	1.08
P=6	61.67	18.03	30.76	36.82	1.14
P=8	61.98	19.33	29.40	36.90	1.19

(b) Different branch layer (Qwen3-0.6B-Base).

Layer	Choice	Code	Math	Avg
21 ($\ell^\star$)	**60.64**	**43.55**	**48.26**	**50.82**
3	60.85	42.65	45.05	49.52
5	61.49	39.5	36.88	45.96
10	59.02	39.3	35.22	44.51
15	60.64	37.60	42.99	47.08
19	59.51	41.55	47.42	49.49
22	60.69	40.10	47.93	49.57
23	58.54	40.98	47.77	49.10
27	57.84	31.83	33.85	41.17

(c) Component ablation.

Method	Choice	Code	Math	Avg
EntroBranch	60.64	43.55	48.26	50.82
Multi Layers	60.55	42.93	48.94	50.81
Mean Gate	59.54	41.85	47.34	49.58
KV Cache	60.64	44.18	48.46	51.09

(d) Branch-layer selection.

Layer	Choice	Code	Math	Avg	$\ell^\star$
Qwen3-0.6B-Base					
EB	60.64	43.55	48.26	**50.82**	21
Random	59.67	40.48	46.04	48.73	17
Qwen3-1.7B-Base					
EB	42.50	54.88	48.23	**48.54**	19
Random	38.89	51.36	42.85	44.37	3
Qwen3-4B-Base					
EB	60.76	19.57	30.36	**36.90**	29
Random	56.09	16.79	28.78	33.89	27

(e) Comparison with other methods. Where P-48 represents single-path prefix tuning using prefix length 48 and **P (%)** denotes the proportion of fine-tuned parameters to the total number of model parameters.

Method	Choice	Code	Math	Avg	P (%)
EB	60.64	43.55	48.26	**50.82**	0.36
LoRA	60.84	37.20	44.56	47.53	6.34
P-48	59.78	32.68	32.05	41.50	0.36
SMoP	53.05	41.88	45.56	46.83	0.36

(f) Layer-wise absolute entropy descent.

Model	19	20	21	22
Base	0.379	0.286	0.307	0.259
SFT	0.591	0.256	0.534	0.448
Model	**23**	**24**	**25**	**26**
Base	0.225	0.106	0.103	0.058
SFT	0.516	0.268	0.087	0.188

(g) Training loss of single path prefix tuning and EntroBranch.

Model	$Loss_{single}$	$Loss_{EB}$
Qwen3-0.6B-Base	0.3712	0.2549
Qwen3-1.7B-Base	0.2931	0.2530
Qwen3-4B-Base	0.2493	0.2334

Aggregate Gate: Replacing the gate with averaging reduces performance (Table 3c), while still outperforming the single-path baseline. This supports the usefulness of the learned token-wise gate.

4.6 Comparison with Fine-Tuning Methods

We compare EntroBranch with LoRA in Table 3e. LoRA uses 17.6 times more trainable parameters than EntroBranch but obtains a lower average score. A parameter-matched single-path baseline (P-48) also performs worse, indicating that the gain cannot be explained solely by adding trainable parameters. We also compared our method with SMoP (Choi et al., 2023), a prefix-tuning method that employs different prompts for different problems. The results show that EntroBranch also outperforms SMoP, suggesting that prompt diversity alone does not account for the observed gains.

4.7 Memory Analysis

With the shared cache, the persistent KV-cache of the branch layer is reduced to $1/P$ of the original size, restoring the asymptotic footprint of single-path decoding.

In real-world inference, EntroBranch uses 2.52/4.72 GB VRAM on Qwen3-0.6B/1.7B, compared with 2.53/4.71 GB for the single-path baseline. This indicates that the shared KV cache maintains memory usage comparable to single-path, with minor variation due to different generated contexts.

To verify that the shared KV cache does not degrade performance, we compare it with an independent branch cache on Qwen3-0.6B-Base. As shown in Table 3c (KV Cache), both achieve nearly identical performance, supporting the memory-saving cache design without an observed accuracy penalty. The slight difference is caused by the precision of floating-point numbers.

4.8 Mechanistic Analysis

We examine whether EntroBranch behaves according to the mechanism motivating localized scaling. Specifically, we test whether predictive concentration identifies layers that benefit from additional computation, whether branching creates non-redundant hidden states that are selectively used across tokens, and whether these branch differences align with the downstream loss gradient. Detailed procedures are provided in Appendix H.

Concentration as a scaling signal. Our motivation predicts that layers with stronger prediction-relevant refinement should be more suitable for localized scaling. The study in Section 4.5 supports this prediction: predictive contraction correlates positively with downstream performance, and the predictive concentration layer outperforms the tested alternatives. We further examine whether this signal persists after adaptation. On Qwen3-0.6B-Base, predictive contraction remains non-uniform after single-path prefix tuning, and the selected layer remains highly contractive (Table 3f).

Branch differentiation and token-wise gating. Localized scaling is useful only if additional branches provide non-redundant hidden states and the gate can exploit them. At the selected layer ($\ell^\star = 21$) on HumanEval, branch outputs have cosine similarity 0.746. Gate weights are highly selective, with normalized entropy 0.178 ± 0.056 and maximum probability 0.946 ± 0.018; 77.3% of tokens exceed 0.95 confidence. One branch receives 87.2% of the average gate mass, while the other receives 12.8%; notably, the latter still attains the highest gate weight for 10.3% of tokens, indicating that the additional branch is used selectively rather than redundantly.

Gradient alignment and optimization. Theorem 1 predicts that branching is useful when directions spanned by branch differences overlap with the downstream loss gradient. We therefore project the supervised-loss gradient onto this subspace and compare it with nearby-layer and random controls. At layer 21, $G_{\mathrm{proj}} = 1.430 \times 10^{-6}$ and $G_{\mathrm{oracle}} = 6.557 \times 10^{-7}$, versus 1.993×10^{-7} and 4.451×10^{-7} at layer 24. Here G_{proj} measures gradient alignment with the branch-reweighting subspace, while G_{oracle} measures the corresponding feasible loss reduction; an equal-rank random subspace serves as a dimensionality control. The learned rank-one branch subspace also captures 0.452% of gradient energy, compared with 0.098% for an equal-rank isotropic random subspace, a $4.63\times$ enrichment. Consistently, EntroBranch reaches lower training loss than single-path prefix tuning (Table 3g).

Together, these results support the proposed scaling mechanism: predictive concentration indicates where additional computation is most useful, branching supplies non-redundant capacity at that layer, and the gate exploits directions aligned with the downstream objective.

5 Conclusion

We present EntroBranch, which uses predictive entropy contraction to guide local expansion inside a frozen Transformer. Learned prefix-conditioned branches and token-wise aggregation modify an intermediate representation before subsequent layers resume single-path computation. Our analysis characterizes when reweighting branch outputs can locally reduce supervised loss. Across Qwen3 and Gemma4, EntroBranch improves overall scores by an average of 33.70% relative to single-path prefix tuning with only 4.58% additional per-token inference FLOPs. These results support entropy-guided local expansion as an effective way to improve predictive performance with modest overhead.

AI USE STATEMENT

Generative AI tools (ChatGPT, Codex) were used in four aspects of this work.

Research ideation or execution. We used generative AI tools to assist in modifying parts of the model implementation code. All resulting code was reviewed and tested by us before being used in the experiments.

Aiding or polishing writing. For the Abstract, Introduction, Related Work, Method, Experiments, Conclusion, and Appendices B, C, E, and F, we first prepared the substantive content. Generative AI tools were then used to improve the language, clarity, organization, and presentation. All AI-assisted revisions were reviewed and finalized by us.

Drafting sections of the paper. For Appendices D, G, H, and I, we first provided the technical content, key points, and organizational structure, and generative AI tools were used to produce preliminary drafts. These drafts were subsequently reviewed, revised, and polished by us to ensure technical accuracy and consistency with the underlying work.

Proving mathematical claims. For some mathematical proofs in Appendix A, generative AI tools were used to suggest possible proof strategies or intermediate ideas. We completed and verified the proofs.

We reviewed and verified all AI-assisted code, mathematical arguments, and manuscript content, and take full responsibility for the correctness, interpretation, and final presentation of this work.

We have not used generative AI tools for generating synthetic datasets, retrieval and discovery, and other purposes.

ETHICS STATEMENT

Our work complies with the Ethics Policy. All datasets and models are openly accessible. We strictly adhere to the respective licensing conditions of all openly accessible datasets and models, ensuring that any derivative artifacts are used exclusively for research purposes and not deployed commercially. We have not identified any significant ethical considerations associated with our work.

REPRODUCIBILITY STATEMENT

We provide the model configurations, dataset composition, training settings, evaluation protocols, and implementation details required to reproduce our experiments in the main paper and appendix. Detailed dataset information and experimental configurations are reported in the corresponding experimental sections. Upon publication, we will release our source code when the paper is accepted to facilitate reproduction and further research.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/.

AI-MO Team. AIMO Validation AMC. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

Mouxiang Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin, and Zhongxin Liu. Parallel scaling law for language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 118958–118998. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ac6de776b8de8c9aed1d356997eb54b8-Paper-Conference.pdf.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. SMoP: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14306–14316, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.884. URL https://aclanthology.org/2023.emnlp-main.884/.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54158–54183, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edc36117f795ca52a0cbf6a7b3882859-Paper-Conference.pdf.

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

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu,

Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Xuanwen Ding, Chengjun Pan, Zejun Li, Jiwen Zhang, Siyuan Wang, and Zhongyu Wei. AutoJudger: An agent-driven framework for efficient benchmarking of MLLMs. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15009–15034, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.685. URL https://aclanthology.org/2026.acl-long.685/.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3/.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. Cruxeval: a benchmark for code reasoning, understanding and execution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Xiaomeng Hu, Yixuan Tang, Haoze Li, Hao Chen, Qi Zhang, Zhanming Shen, Yiming Zhang, Haobo Wang, and Junbo Zhao. Fine-grained data ordering improves fine-tuning for large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20406–20418, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1021. URL https://aclanthology.org/2026.findings-acl.1021/.

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. *arXiv preprint arXiv:2403.07974*, 2024.

Lou Jost. Entropy and diversity. *Oikos*, 113(2):363–375, 2006.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset, 2023. URL https://arxiv.org/abs/2312.14852.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL https://www.science.org/doi/abs/10.1126/science.abq1158.

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. *arXiv preprint arXiv:2305.20050*, 2023.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

YongKang Liu, Xingle Xu, Ercong Nie, Zijing Wang, Shi Feng, Daling Wang, Qian Li, and Hinrich Schuetze. Look within or beyond? a theoretical comparison between parameter-efficient and full fine-tuning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47800–47820, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2208. URL https://aclanthology.org/2026.acl-long.2208/.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 27168–27188, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/72eba29737f9c3a5a4ce8cdb7b667145-Paper-Conference.pdf.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

nostalgebraist. Interpreting gpt: the logit lens. August 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Open R1. Openr1-math-220k. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/open-r1/OpenR1-Math-220k.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: high-throughput generative inference of large language models with a single gpu. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan, Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina

Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan Liss, Jos Kraaijeveld, Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Y. Guan, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll, Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng, Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomek Korbak, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan, Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang

Wang. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Glenn Cameron, Charlotte Caucheteux, Garima Chadha, Jetha Chan, Aditya Chawla, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Takumi Fujimoto, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Chetan Tekur, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: empowering code generation with oss-instruct. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3054–3059, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1342. URL https://aclanthology.org/D18-1342/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Zhuohao Yu, Weizheng Gu, Yidong Wang, Xingru Jiang, Zhengran Zeng, Jindong Wang, Wei Ye, and Shikun Zhang. Reasoning through execution: unifying process and outcome rewards for code generation. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Di Zhang, 2025. URL https://huggingface.co/datasets/di-zhang-fdu/AIME_1983_2024.

Wenqing Zheng, S P Sharan, Ajay Kumar Jaiswal, Kevin Wang, Yihan Xi, Dejia Xu, and Zhangyang Wang. Outline, then details: syntactically guided coarse-to-fine code generation. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Gong, Diyi Yang, and Xing Xie. Dyval: Dynamic evaluation of large language models for reasoning tasks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 18091–18128, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4ea4a1ea4d9ff273688c8e92bd087112-Paper-Conference.pdf.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.

A Proofs and Extended Theoretical Discussion

This appendix separates three roles of the analysis. First, we characterize the representations attainable by reweighting fixed branches and give sufficient conditions for a local loss-reducing reweighting. These are feasibility statements in coefficient space, not guarantees about joint prefix–gate training. Second, we interpret entropy contraction as a calibration signal for choosing a branch layer and width; the allocation rules are evaluated empirically. Third, we prove exactness of the method's content-KV sharing with independent prefix KV and separately analyze full-KV averaging as a lossy ablation. Throughout the geometric analysis, the input sequence and branch representations are fixed, and the downstream loss is evaluated with deterministic computation.

A.1 Reachable Representations and Local Branch-Variation Directions

Fix one token and suppress its layer and token indices. Let $P = P^\star \geq 2$, and collect the branch outputs as

$$\mathbf{Z} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(P)}] \in \mathbb{R}^{d\times P}, \qquad \mathbf{J}_P = \mathbf{I}_P - \frac{1}{P}\mathbf{1}\mathbf{1}^\top, \qquad \mathbf{Z}_c = \mathbf{Z}\mathbf{J}_P. \tag{3}$$

Write $\bar{\mathbf{x}} = P^{-1}\sum_j \mathbf{x}^{(j)}$ and define

$$\begin{aligned}\Delta_P &:= \{\mathbf{a} \in \mathbb{R}^P : \mathbf{a} \geq \mathbf{0},\ \mathbf{1}^\top\mathbf{a} = 1\},\\ \mathcal{C} &:= \mathbf{Z}\Delta_P = \mathrm{conv}\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(P)}\},\\ \mathcal{S} &:= \mathrm{Col}(\mathbf{Z}_c), \qquad \mathbf{\Pi}_\mathcal{S} := \mathbf{Z}_c\mathbf{Z}_c^\dagger.\end{aligned}$$

Here † denotes the Moore–Penrose pseudoinverse. The convex hull $\mathcal{C}$ is the reachable set under freely chosen simplex coefficients; $\mathcal{S}$ is its affine direction space. In particular, $\mathrm{aff}(\mathcal{C}) = \bar{\mathbf{x}} + \mathcal{S}$, and $\mathrm{rank}(\mathbf{Z}_c) \leq \min\{d, P-1\}$. These objects do not depend on the choice of a reference branch.

Proposition 1 (Reachable set and local reweighting directions)**.** *Let $\mathbf{a} \in \Delta_P$ and $\mathbf{x} = \mathbf{Z}\mathbf{a}$. Every zero-sum coefficient perturbation satisfies*

$$\delta\mathbf{x} = \mathbf{Z}\Delta\mathbf{a} = \mathbf{Z}_c\Delta\mathbf{a} \in \mathcal{S}, \qquad \mathbf{1}^\top\Delta\mathbf{a} = 0. \tag{4}$$

Its endpoint is feasible whenever $\mathbf{a} + \Delta\mathbf{a} \geq \mathbf{0}$. Conversely, every $\mathbf{y} \in \mathcal{S}$ admits a zero-sum coefficient representation. If $a_j > 0$ for every j, then for each $\mathbf{y} \in \mathcal{S}$ there is $\tau_\mathbf{y} > 0$ such that $\mathbf{x} + \tau\mathbf{y} \in \mathcal{C}$ for all $|\tau| < \tau_\mathbf{y}$. Thus $\mathcal{S}$ describes all locally feasible directions at a strictly positive mixture; it is not an unbounded reachable set of representations.

Proof. Since $\mathbf{J}_P\Delta\mathbf{a} = \Delta\mathbf{a}$ for zero-sum perturbations, Eq. 4 follows directly. Each column of $\mathbf{Z}_c$ is $\mathbf{x}^{(j)} - \bar{\mathbf{x}}$; each pairwise difference is a difference of such columns, and each centered column is an average of pairwise differences. This proves the claimed affine direction space and the rank bound, since $\mathrm{rank}(\mathbf{J}_P) = P - 1$.

For the converse, observe that $\mathbf{Z}_c\mathbf{1} = \mathbf{0}$ and

$$\mathrm{range}(\mathbf{Z}_c^\dagger) = \mathrm{range}(\mathbf{Z}_c^\top) \subseteq \mathbf{1}^\perp.$$

For $\mathbf{y} \in \mathcal{S}$, set $\mathbf{c} = \mathbf{Z}_c^\dagger\mathbf{y}$. Then $\mathbf{1}^\top\mathbf{c} = 0$ and

$$\mathbf{Z}\mathbf{c} = \mathbf{Z}_c\mathbf{c} = \mathbf{\Pi}_\mathcal{S}\mathbf{y} = \mathbf{y}.$$

If $\mathbf{y} \neq \mathbf{0}$ and $a_{\min} = \min_j a_j > 0$, any $|\tau| < a_{\min}/\|\mathbf{c}\|_\infty$ keeps $\mathbf{a} + \tau\mathbf{c}$ strictly positive and preserves its sum. If $\mathbf{y} = \mathbf{0}$, take $\mathbf{c} = \mathbf{0}$. This proves local two-sided feasibility. □

Boundary coefficients and finite softmax logits. At a boundary coefficient vector, the feasible one-sided coefficient directions are

$$\mathcal{T}_{\Delta_P}(\mathbf{a}) = \{\mathbf{c} : \mathbf{1}^\top\mathbf{c} = 0,\ c_j \geq 0 \text{ whenever } a_j = 0\}.$$

Their images $\mathbf{Z}\mathcal{T}_{\Delta_P}(\mathbf{a})$ need not form a subspace, so nonzero projection onto $\mathcal{S}$ alone need not imply a feasible descent direction at the boundary. Every strictly positive coefficient vector is realizable by free finite logits $z_j = \log a_j$; boundary coefficients can be approached by replacing $\mathbf{a}$ with

$(1-\epsilon)\mathbf{a}+\epsilon\mathbf{1}/P$. Thus the closure of the free-softmax reachable set is $\mathcal{C}$. This statement does not assert realizability by a shared parameterized gate across inputs.

For $P=2$, the per-token direction space has dimension at most one. Across tokens these directions may differ, but a high pooled rank does not establish a high-dimensional correction space for any individual token. With $P=1$, reweighting provides no direction; this does not preclude optimization through changes to the single branch's parameters.

A.2 Local Descent Feasibility for Fixed Branches

We prove Theorem 1 as a local existence result. We first identify a descent direction in $\mathcal{S}$, then construct a feasible coefficient change that realizes it, and finally quantify the decrease using smoothness. A separate conditional bound covers an actual gate-only update if its representation change approximates that direction; no optimizer-realization guarantee is assumed.

Proposition 2 (Local descent feasibility and conditional realization error)**.** *Let $\mathbf{a}\in\mathbb{R}^P$ satisfy $a_j>0$ and $\mathbf{1}^\top\mathbf{a}=1$, and let $\mathbf{x}=\mathbf{Z}\mathbf{a}$. Define*

$$\mathcal{C}:=\operatorname{conv}\{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(P)}\}.$$

Assume that the downstream supervised loss $\mathcal{L}$ has an L_{sm}-Lipschitz gradient, with $L_{\mathrm{sm}}>0$, on a convex neighbourhood containing $\mathcal{C}$. Let

$$\mathbf{g}:=\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}),\qquad \mathbf{v}:=\mathbf{\Pi}_{\mathcal{S}}\mathbf{g},\qquad \mathbf{b}:=-\mathbf{J}_P\mathbf{Z}_c^{\dagger}\mathbf{g}.$$

Define

$$\gamma_{\max}:=\min_{j:b_j<0}\frac{a_j}{-b_j},\qquad \gamma^\star:=\min\{L_{\mathrm{sm}}^{-1},\gamma_{\max}\},\tag{5}$$

where the minimum over an empty set is $+\infty$. Then $\mathbf{a}^+=\mathbf{a}+\gamma^\star\mathbf{b}$ is feasible in the closed simplex, induces $\mathbf{x}^+=\mathbf{x}-\gamma^\star\mathbf{v}$, and satisfies

$$\mathcal{L}(\mathbf{x})-\mathcal{L}(\mathbf{x}-\gamma^\star\mathbf{v})\ge\left(\gamma^\star-\frac{L_{\mathrm{sm}}(\gamma^\star)^2}{2}\right)\|\mathbf{v}\|_2^2.\tag{6}$$

If $\gamma_{\max}\ge L_{\mathrm{sm}}^{-1}$, the lower bound is $\|\mathbf{v}\|_2^2/(2L_{\mathrm{sm}})$.

Moreover, suppose a feasible coefficient update induces $\delta\mathbf{x}=-\lambda\mathbf{v}+\mathbf{e}$, where $\lambda>0$ and $\|\mathbf{e}\|_2\le\varepsilon\lambda\|\mathbf{v}\|_2$ for some $0\le\varepsilon<1$. Then

$$\begin{aligned}&\mathcal{L}(\mathbf{x})-\mathcal{L}(\mathbf{x}+\delta\mathbf{x})\\&\quad\ge\left[\lambda(1-\varepsilon)-\frac{L_{\mathrm{sm}}\lambda^2}{2}(1+\varepsilon)^2\right]\|\mathbf{v}\|_2^2.\end{aligned}\tag{7}$$

If $\mathbf{v}\neq\mathbf{0}$ and $0<\lambda\le(1-\varepsilon)/(L_{\mathrm{sm}}(1+\varepsilon)^2)$, the decrease is at least $\lambda(1-\varepsilon)\|\mathbf{v}\|_2^2/2>0$.

Step 1: identify the useful part of the gradient. The full gradient $\mathbf{g}$ generally contains directions that branch reweighting cannot realize. Proposition 1 tells us to retain only its component in $\mathcal{S}$:

$$\mathbf{v}=\mathbf{\Pi}_{\mathcal{S}}\mathbf{g}.$$

If $\mathbf{v}=\mathbf{0}$, the available branch differences contain no first-order descent direction. Indeed, for every $\mathbf{y}\in\mathcal{S}$, $\mathbf{g}^\top\mathbf{y}=\mathbf{v}^\top\mathbf{y}$. Under strictly positive coefficients, a feasible direction with strictly negative directional derivative therefore exists if and only if $\mathbf{v}\neq\mathbf{0}$. If $\mathbf{v}=\mathbf{0}$, higher-order or nonlocal improvements are not ruled out.

Step 2: realize $-\mathbf{v}$ by changing the mixture coefficients. Consider

$$\mathbf{b}=-\mathbf{J}_P\mathbf{Z}_c^{\dagger}\mathbf{g}.$$

The row-space identity proved above gives $\mathbf{J}_P\mathbf{Z}_c^{\dagger}=\mathbf{Z}_c^{\dagger}$, so $\mathbf{b}=-\mathbf{Z}_c^{\dagger}\mathbf{g}=-\mathbf{Z}_c^{\dagger}\mathbf{v}$. In particular, this is a valid tangent direction of the simplex:

$$\mathbf{1}^\top\mathbf{b}=0.$$

More importantly, this coefficient direction realizes exactly the negative projected gradient:

$$\begin{aligned}\mathbf{Z}\mathbf{b} &= -\mathbf{Z}\mathbf{J}_P\mathbf{Z}_c^\dagger\mathbf{g} = -\mathbf{Z}_c\mathbf{Z}_c^\dagger\mathbf{g} \\ &= -\mathbf{\Pi}_{\mathcal{S}}\mathbf{g} = -\mathbf{v}.\end{aligned}$$

Thus the coefficient update $\mathbf{a} \mapsto \mathbf{a} + \gamma\mathbf{b}$ moves the hidden state by

$$\delta\mathbf{x} = -\gamma\mathbf{v}.$$

This establishes first-order descent feasibility when the branch directions overlap the task gradient. The size of a usable decrease also depends on the feasible step length and local curvature; nonzero overlap alone does not imply a practically large gain.

Step 3: keep the reweighting feasible and turn first-order alignment into loss decrease. Because the current coefficients are strictly positive, a sufficiently small step along $\mathbf{b}$ stays inside the simplex. The largest admissible step is

$$\gamma_{\max} = \min_{j:b_j<0} \frac{a_j}{-b_j}.$$

When $\mathbf{b} \neq \mathbf{0}$, its entries sum to zero and hence at least one entry is negative; when $\mathbf{b} = \mathbf{0}$, we use the convention $\gamma_{\max} = +\infty$. Therefore every $0 \leq \gamma \leq \gamma_{\max}$ is feasible.

Now apply L_{sm}-smoothness along this feasible segment:

$$\mathcal{L}(\mathbf{x} + \boldsymbol{\delta}) \leq \mathcal{L}(\mathbf{x}) + \mathbf{g}^\top\boldsymbol{\delta} + \frac{L_{\mathrm{sm}}}{2}\|\boldsymbol{\delta}\|_2^2.$$

No convexity of $\mathcal{L}$ is needed. Because $\mathbf{\Pi}_{\mathcal{S}}$ is an orthogonal projector,

$$\mathbf{g}^\top\mathbf{v} = \mathbf{g}^\top\mathbf{\Pi}_{\mathcal{S}}\mathbf{g} = \|\mathbf{v}\|_2^2.$$

Substituting $\boldsymbol{\delta} = -\gamma\mathbf{v}$ gives

$$\mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{x} - \gamma\mathbf{v}) \geq \left(\gamma - \frac{L_{\mathrm{sm}}\gamma^2}{2}\right)\|\mathbf{v}\|_2^2.$$

Choosing $\gamma = \gamma^\star$ proves Eq. 6. If $\gamma_{\max} \geq L_{\mathrm{sm}}^{-1}$, then $\gamma^\star = L_{\mathrm{sm}}^{-1}$ and the lower bound becomes $\|\mathbf{v}\|_2^2/(2L_{\mathrm{sm}})$.

The same inequality also proves the theorem statement in the main text. For every $0 < \gamma \leq \min\{\gamma_{\max}, L_{\mathrm{sm}}^{-1}\}$,

$$\begin{aligned}\mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{x} - \gamma\mathbf{v}) &\geq \left(\gamma - \frac{L_{\mathrm{sm}}\gamma^2}{2}\right)\|\mathbf{v}\|_2^2 \\ &\geq \frac{\gamma}{2}\|\mathbf{v}\|_2^2.\end{aligned}$$

Hence one may take any positive $\gamma_0 \leq \gamma_{\max}$, and the decrease is strict whenever $\mathbf{v} \neq \mathbf{0}$. This proves Theorem 1.

Why the learned gate need not realize the ideal update exactly. The construction above treats the mixture coefficients as locally free variables. In the actual model, they are outputs of a shared gate, so the realized update may deviate from $-\lambda\mathbf{v}$. Write that deviation as

$$\delta\mathbf{x} = -\lambda\mathbf{v} + \mathbf{e}, \qquad \|\mathbf{e}\|_2 \leq \varepsilon\lambda\|\mathbf{v}\|_2.$$

Both $\delta\mathbf{x}$ and $\mathbf{v}$ lie in $\mathcal{S}$, hence $\mathbf{e} \in \mathcal{S}$. Therefore $\mathbf{g}^\top\mathbf{e} = \mathbf{v}^\top\mathbf{e}$, and

$$\begin{aligned}-\mathbf{g}^\top\delta\mathbf{x} &= \lambda\|\mathbf{v}\|_2^2 - \mathbf{v}^\top\mathbf{e} \\ &\geq \lambda(1-\varepsilon)\|\mathbf{v}\|_2^2.\end{aligned}$$

At the same time,

$$\|\delta\mathbf{x}\|_2 \leq \lambda(1+\varepsilon)\|\mathbf{v}\|_2.$$

Applying smoothness once more yields Eq. 7. Thus any feasible realization satisfying the stated error and step-size conditions retains a local decrease. These conditions are additional hypotheses, not consequences of joint training. If $\mathbf{v} = \mathbf{0}$, the ideal correction vanishes and the relative-error

assumption forces $\delta\mathbf{x} = \mathbf{0}$, so no strict decrease is claimed. We additionally record the training loss; the experiment in Section 4.8 shows that the training loss of EntroBranch is lower than the loss of the single path.

If $\gamma^\star = \gamma_{\max}$ places a coefficient exactly on the simplex boundary, any $0 \le \gamma < \gamma^\star$ remains in the interior and can be represented by finite softmax logits. For such an interior endpoint, free logits $z_j^+ = \log(a_j + \gamma b_j)$ realize the coefficients exactly. Boundary points can therefore be approached continuously. This observation concerns free per-token mixture coefficients; it does not claim that an arbitrary collection of such coefficients is simultaneously realizable by one shared gate network.

Extension from one token to a full sequence. The extension first treats the coefficients at distinct tokens as free variables. Their product feasible set is a relaxation of the set jointly realizable by a shared gate, even though the gate produces token-wise outputs. For a sequence of length T_{seq}, let $\mathcal{S}_t$ and $\mathbf{\Pi}_t$ be the reachable subspace and projector at token t. Stack the token representations and gradients as

$$\mathbf{x}_{\mathrm{seq}} := \mathrm{vec}(\mathbf{x}_1, \dots, \mathbf{x}_{T_{\mathrm{seq}}}), \qquad \mathbf{g}_{\mathrm{seq}} := \nabla_{\mathbf{x}_{\mathrm{seq}}} \mathcal{L}(\mathbf{x}_{\mathrm{seq}}),$$

and define

$$\begin{aligned} \mathcal{S}_{\mathrm{seq}} &:= \bigoplus_{t=1}^{T_{\mathrm{seq}}} \mathcal{S}_t, \\ \mathbf{\Pi}_{\mathrm{seq}} &:= \mathrm{blkdiag}(\mathbf{\Pi}_1, \dots, \mathbf{\Pi}_{T_{\mathrm{seq}}}), \\ \mathbf{v}_{\mathrm{seq}} &:= \mathbf{\Pi}_{\mathrm{seq}} \mathbf{g}_{\mathrm{seq}}. \end{aligned}$$

Assume that $\mathcal{L}$ has an L_{seq}-Lipschitz gradient on a convex neighbourhood containing

$$\mathcal{C}_{\mathrm{seq}} := \prod_{t=1}^{T_{\mathrm{seq}}} \mathrm{conv}\{\mathbf{x}_t^{(1)}, \dots, \mathbf{x}_t^{(P)}\}.$$

Assume all current token-wise coefficients are strictly positive and $L_{\mathrm{seq}} > 0$. The downstream loss may couple tokens arbitrarily; neither the loss nor its Hessian is assumed to be layer diagonal.

Applying the coefficient construction independently at each token and taking the smallest feasible step across tokens gives

$$\gamma_{\max,\mathrm{seq}} := \min_t \gamma_{\max,t}, \qquad \gamma_{\mathrm{seq}}^\star := \min\{L_{\mathrm{seq}}^{-1}, \gamma_{\max,\mathrm{seq}}\}.$$

The resulting ideal sequence update is $-\gamma_{\mathrm{seq}}^\star \mathbf{v}_{\mathrm{seq}}$, and smoothness gives

$$\begin{aligned} &\mathcal{L}(\mathbf{x}_{\mathrm{seq}}) - \mathcal{L}(\mathbf{x}_{\mathrm{seq}} - \gamma_{\mathrm{seq}}^\star \mathbf{v}_{\mathrm{seq}}) \\ &\qquad \ge \left(\gamma_{\mathrm{seq}}^\star - \frac{L_{\mathrm{seq}} (\gamma_{\mathrm{seq}}^\star)^2}{2} \right) \|\mathbf{v}_{\mathrm{seq}}\|_2^2. \end{aligned} \tag{8}$$

For an actual shared-gate-only update $\phi \mapsto \phi^+$, hold all branch outputs and downstream parameters fixed and define

$$\delta\mathbf{x}_{\mathrm{seq}} := \mathbf{x}_{\mathrm{seq}}(\phi^+) - \mathbf{x}_{\mathrm{seq}}(\phi).$$

Each token-wise coefficient change still sums to zero, so $\delta\mathbf{x}_{\mathrm{seq}} \in \mathcal{S}_{\mathrm{seq}}$. Therefore, if

$$\delta\mathbf{x}_{\mathrm{seq}} = -\lambda \mathbf{v}_{\mathrm{seq}} + \mathbf{e}_{\mathrm{seq}}, \qquad \|\mathbf{e}_{\mathrm{seq}}\|_2 \le \varepsilon\lambda \|\mathbf{v}_{\mathrm{seq}}\|_2,$$

then the same argument yields

$$\begin{aligned} &\mathcal{L}(\mathbf{x}_{\mathrm{seq}}) - \mathcal{L}(\mathbf{x}_{\mathrm{seq}} + \delta\mathbf{x}_{\mathrm{seq}}) \\ &\qquad \ge \left[\lambda(1-\varepsilon) - \frac{L_{\mathrm{seq}} \lambda^2}{2} (1+\varepsilon)^2 \right] \|\mathbf{v}_{\mathrm{seq}}\|_2^2. \end{aligned} \tag{9}$$

In particular, if $\mathbf{v}_{\mathrm{seq}} \neq \mathbf{0}$ and $0 < \lambda \le (1-\varepsilon)/(L_{\mathrm{seq}}(1+\varepsilon)^2)$, the loss decreases by at least $\lambda(1-\varepsilon)\|\mathbf{v}_{\mathrm{seq}}\|_2^2/2$.

The additional restriction imposed by a shared gate. Let $\mathbf{h}(\phi) = \mathbf{x}_{\text{seq}}(\phi)$ for fixed branch outputs, and suppose $\mathbf{h}$ is differentiable at the current parameters. Its Jacobian $\mathbf{J}_\phi = D_\phi \mathbf{h}$ satisfies

$$\text{range}(\mathbf{J}_\phi) \subseteq \mathcal{S}_{\text{seq}}, \qquad \nabla_\phi \mathcal{L} = \mathbf{J}_\phi^\top \mathbf{g}_{\text{seq}}.$$

The inclusion follows because every coefficient vector sums to one, so its derivative has zero column sums. Consequently, $\mathbf{v}_{\text{seq}} \neq \mathbf{0}$ need not imply $\nabla_\phi \mathcal{L} \neq \mathbf{0}$. If instead $\mathbf{J}_\phi^\top \mathbf{g}_{\text{seq}} \neq \mathbf{0}$, differentiability yields, for a hypothetical sufficiently small gate-gradient step,

$$\begin{aligned}&\mathcal{L}\big(\mathbf{h}(\phi - \eta \nabla_\phi \mathcal{L})\big) \\ &\quad = \mathcal{L}\big(\mathbf{h}(\phi)\big) - \eta \|\mathbf{J}_\phi^\top \mathbf{g}_{\text{seq}}\|_2^2 + o(\eta).\end{aligned}$$

This distinguishes coefficient-space feasibility from parameter-space feasibility.

Scope and empirical diagnostics. Theorem 1 establishes the existence of a local loss-reducing reweighting when fixed branch directions have nonzero gradient overlap. It does not establish that training creates such overlap, that the shared gate realizes the ideal direction, or that EntroBranch must outperform a separately trained single-path model. For $\mathbf{v}_{\text{seq}} \neq \mathbf{0}$ and a specified $\lambda > 0$, the conditional realization bound can be assessed using

$$\varepsilon_{\text{real}} := \frac{\|\delta \mathbf{x}_{\text{seq}} + \lambda \mathbf{v}_{\text{seq}}\|_2}{\lambda \|\mathbf{v}_{\text{seq}}\|_2}.$$

This diagnostic requires a gate-only update with fixed branch outputs. Layer/width selection and final task performance are separate empirical questions. We further demonstrate the validity of this theorem through additional experiments in Section 4.

A.3 When Is the Reachable Correction Space Well Conditioned?

The previous result depends on whether useful gradient directions can be represented by branch differences. A second issue is how stably those directions can be produced in coefficient space. The centered branch matrix $\mathbf{Z}_c$ answers both questions: its column space determines which directions are reachable, while its singular values determine how much coefficient movement is needed to realize them.

Proposition 3 (Branch-subspace conditioning)**.** *Use the strictly positive coefficients* $\mathbf{a}$*, gradient* $\mathbf{g}$*, direction* $\mathbf{b}$*, and ray-wise feasibility limit* $\gamma_{\max}$ *of Proposition 2. Let*

$$r_{\mathcal{S}} := \text{rank}(\mathbf{Z}_c) \geq 1, \qquad \mathcal{E}_{\mathcal{S}} := \|\mathbf{Z}_c\|_F^2,$$

and let $\sigma_{\min}^+(\mathbf{Z}_c)$ *denote the smallest nonzero singular value of* $\mathbf{Z}_c$*. Then*

$$\big(\sigma_{\min}^+(\mathbf{Z}_c)\big)^2 \leq \frac{\mathcal{E}_{\mathcal{S}}}{r_{\mathcal{S}}}. \tag{10}$$

Equality holds if and only if all nonzero singular values of $\mathbf{Z}_c$ *are equal.*

Moreover, let $a_{\min} := \min_j a_j$*. If* $\|\mathbf{v}\|_2 > 0$*, where* $\mathbf{v} = \mathbf{\Pi}_{\mathcal{S}} \mathbf{g}$*, then*

$$\gamma_{\max} \geq \frac{a_{\min} \sigma_{\min}^+(\mathbf{Z}_c)}{\|\mathbf{v}\|_2}. \tag{11}$$

If $\mathbf{v} = \mathbf{0}$*, then* $\mathbf{b} = \mathbf{0}$ *and* $\gamma_{\max} = +\infty$ *under the convention of Proposition 2.*

Proof. Let the nonzero singular values of $\mathbf{Z}_c$ be $\sigma_1 \geq \cdots \geq \sigma_{r_{\mathcal{S}}} > 0$. Since

$$\mathcal{E}_{\mathcal{S}} = \|\mathbf{Z}_c\|_F^2 = \sum_{i=1}^{r_{\mathcal{S}}} \sigma_i^2,$$

the smallest squared singular value cannot exceed their average, which gives Eq. 10. Equality holds exactly when all nonzero singular values are equal. In that case the branch-variation energy is distributed isotropically over the reachable subspace:

$$\mathbf{Z}_c \mathbf{Z}_c^\top = \frac{\mathcal{E}_{\mathcal{S}}}{r_{\mathcal{S}}} \mathbf{\Pi}_{\mathcal{S}}.$$

More explicitly, let $\mathbf{Z}_c = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^\top$ be a compact SVD and $\mathbf{v} = \sum_{i=1}^{r_\mathcal{S}} c_i \mathbf{u}_i$. The minimum-Euclidean-norm coefficient solution of $\mathbf{Z}_c \mathbf{b} = -\mathbf{v}$ is

$$\mathbf{b} = -\sum_{i=1}^{r_\mathcal{S}} \frac{c_i}{\sigma_i} \mathbf{v}_i, \qquad \|\mathbf{b}\|_2^2 = \sum_{i=1}^{r_\mathcal{S}} \frac{c_i^2}{\sigma_i^2}.$$

It is zero-sum because the right singular vectors lie in $\mathbf{1}^\perp$. Null-space additions cannot reduce its Euclidean norm, although they can change the maximum feasible step. Thus $\gamma_{\max}$ concerns the specified coefficient ray, not the best possible realization over all coefficient directions.

To bound this ray's feasible step, first note that $\mathbf{Z}_c^\dagger \mathbf{g} = \mathbf{Z}_c^\dagger \mathbf{v}$, because the pseudoinverse annihilates the component orthogonal to $\mathcal{S}$. Hence

$$\begin{aligned} \|\mathbf{b}\|_2 &= \|\mathbf{J}_P \mathbf{Z}_c^\dagger \mathbf{g}\|_2 \\ &\le \frac{\|\mathbf{v}\|_2}{\sigma_{\min}^+(\mathbf{Z}_c)}. \end{aligned}$$

Using $\|\mathbf{b}\|_\infty \le \|\mathbf{b}\|_2$ gives

$$\begin{aligned} \gamma_{\max} &= \min_{j: b_j < 0} \frac{a_j}{-b_j} \ge \frac{a_{\min}}{\|\mathbf{b}\|_\infty} \\ &\ge \frac{a_{\min} \sigma_{\min}^+(\mathbf{Z}_c)}{\|\mathbf{v}\|_2}, \end{aligned}$$

which is Eq. 11. If $\mathbf{v} = \mathbf{0}$, then $\mathbf{Z}_c^\dagger \mathbf{g} = \mathbf{0}$, so $\mathbf{b} = \mathbf{0}$ and there is no first-order correction to realize. □

For fixed total variation $\mathcal{E}_\mathcal{S}$ and rank, equal nonzero singular values maximize the smallest singular value and hence the worst-direction bound in Eq. 11, with $a_{\min}$ and $\|\mathbf{v}\|_2$ held fixed. This is not an unconditional ordering of actual loss improvements: gradient orientation, coefficients, and curvature also matter. In particular, neither a large rank nor a balanced spectrum guarantees $\mathbf{v} \neq \mathbf{0}$.

The same construction gives an explicit local reachability radius. For every $\mathbf{y} \in \mathcal{S}$ with $\|\mathbf{y}\|_2 < a_{\min} \sigma_{\min}^+(\mathbf{Z}_c)$,

$$\|\mathbf{Z}_c^\dagger \mathbf{y}\|_\infty \le \|\mathbf{Z}_c^\dagger \mathbf{y}\|_2 \le \frac{\|\mathbf{y}\|_2}{\sigma_{\min}^+(\mathbf{Z}_c)} < a_{\min}.$$

Thus $\mathbf{a} + \mathbf{Z}_c^\dagger \mathbf{y}$ is a strictly positive feasible mixture realizing $\mathbf{x} + \mathbf{y}$. This quantifies local reachability without identifying $\mathcal{S}$ with the whole reachable convex hull.

A.4 Predictive Commitment, Layer Interception, and Effective Parallelism

Interpretation 4 (predictive commitment and effective breadth). Use natural logarithms, the convention $0 \log 0 = 0$, and a fixed finite vocabulary $\mathcal{V}$. Let

$$u_\mathcal{V}(v) = \frac{1}{|\mathcal{V}|}, \qquad \kappa_{\ell,t} = D_{\mathrm{KL}}(\boldsymbol{\pi}_{\ell,t} \| u_\mathcal{V}), \qquad B_{\ell,t} = \exp(\mathsf{H}_{\ell,t}).$$

The entropy-equivalent breadth $B_{\ell,t}$ is the size of a uniform distribution with the same entropy, interpreted continuously. It is not the support cardinality or the rank of a hidden representation. Direct expansion gives

$$\kappa_{\ell,t} = \sum_{v \in \mathcal{V}} \pi_{\ell,t}(v) \log \pi_{\ell,t}(v) + \log |\mathcal{V}| = \log |\mathcal{V}| - \mathsf{H}_{\ell,t}.$$

Consequently,

$$\begin{aligned} \delta_{\ell,t} &= \left[\kappa_{\ell,t} - \kappa_{\ell-1,t}\right]_+ \\ &= \log \max\left\{1, \frac{B_{\ell-1,t}}{B_{\ell,t}}\right\}. \end{aligned} \tag{12}$$

Thus $\exp(\delta_{\ell,t})$ is exactly the multiplicative contraction of effective predictive breadth whenever the entropy decreases. This identity characterizes predictive concentration; we further demonstrate through experiments in Section 4.5, 4.8, and Appendix D the effectiveness of parallelism at this layer.

Robustness to fixed-readout miscalibration. Our layer-selection criterion does not require the logit-lens distribution to be a calibrated prediction at every depth. To make this explicit, let $\mathsf{H}^{\mathrm{ref}}_{\ell,t}$ denote the entropy under an arbitrary reference readout and write the fixed-readout entropy used by EntroBranch as

$$\mathsf{H}_{\ell,t} = \mathsf{H}^{\mathrm{ref}}_{\ell,t} + b_{\ell,t},$$

where $b_{\ell,t}$ captures layer-dependent readout distortion. Define

$$d_{\ell,t} := \mathsf{H}_{\ell-1,t} - \mathsf{H}_{\ell,t}, \qquad d^{\mathrm{ref}}_{\ell,t} := \mathsf{H}^{\mathrm{ref}}_{\ell-1,t} - \mathsf{H}^{\mathrm{ref}}_{\ell,t}.$$

Then

$$d_{\ell,t} - d^{\mathrm{ref}}_{\ell,t} = b_{\ell-1,t} - b_{\ell,t}.$$

Because $[z]_+$ is 1-Lipschitz, the corresponding positive contractions satisfy

$$\left|\delta_{\ell,t} - \delta^{\mathrm{ref}}_{\ell,t}\right| \le \left|b_{\ell-1,t} - b_{\ell,t}\right|, \qquad \delta^{\mathrm{ref}}_{\ell,t} := [d^{\mathrm{ref}}_{\ell,t}]_+. \tag{13}$$

Thus layer-level contraction is insensitive to a large absolute readout bias when that bias varies slowly between adjacent layers; only the *change* in readout distortion enters the bound. We therefore use the same frozen normalization $\mathcal{N}$ and unembedding $\mathbf{W}_{\mathrm{U}}$ at every layer and interpret s_ℓ as a fixed-readout diagnostic rather than a calibrated layer-wise probability estimate. The bound does not guarantee robustness to abrupt layer-specific readout distortion, so the usefulness of the selected layer is ultimately validated by the layer ablations in Section 4.5 and Appendix D.

Early-layer readout instability. Fixed logit-lens readouts are known to be less reliable in early Transformer layers, where hidden representations are less aligned with the final unembedding space. This limitation is relevant to absolute layer-wise predictions, whereas EntroBranch uses the lens only to compare adjacent-layer entropy contraction. Moreover, the layers selected in our experiments are $\ell^\star = 21/28$, $19/28$, $29/36$, and $13/42$ across the four evaluated models, none of which lies among the earliest layers. Thus, the observed allocation decisions are not driven by the most unstable early-layer regime. The layer-wise ablations further validate the selected locations directly through downstream performance.

Commitment-interception principle. During frozen-backbone calibration, let

$$\mathbf{X}_\ell = F_\ell(\mathbf{X}_{\ell-1})$$

denote the hidden-state sequence after layer ℓ. Consider a single-layer augmentation placed at layer k that leaves layers $1, \dots, k-1$ unchanged. For every $\ell < k$, both $\mathbf{X}_{\ell-1}$ and $\mathbf{X}_\ell$ are therefore identical to their calibrated backbone values, so the contraction $\delta_{\ell,t}$ has already been realized before the augmented layer is reached. A later intervention may change downstream predictions, but it cannot modify the local transformation F_ℓ that produced the observed contraction.

In contrast, placing the augmentation at $k = \ell$ directly intercepts that transition. EntroBranch replaces the single realized transformation by

$$\mathbf{X}^{(j)}_\ell = F_\ell(\mathbf{X}_{\ell-1}; \mathbf{E}^{(j)}), \qquad j = 1, \dots, P,$$

so multiple local transformations are formed from the same incoming representation before aggregation and downstream propagation. Under a budget that permits branching at only one layer, the score

$$s_\ell = \frac{1}{|\mathcal{T}_\Omega|} \sum_{(x,t)\in\mathcal{T}_\Omega} \frac{\delta_{\ell,t}}{\mathsf{H}_{\ell-1,t}}$$

therefore ranks the observed relative predictive commitment that can be directly intercepted at each layer. Selecting $\ell^\star = \arg\max_\ell s_\ell$ targets the largest calibrated relative contraction. This is an intervention-timing argument, not a claim that $\ell^\star$ is globally optimal or that contraction itself is harmful.

Rank-limited effective parallelism. The branch geometry further implies that useful parallelism is finite even when the nominal branch count can be increased. For P branches,

$$r_{\mathcal{S}} = \operatorname{rank}(\mathbf{Z}_c) \le \min\{d, P-1\},$$

so local reweighting exposes at most $r_{\mathcal{S}}$ independent first-order correction directions. Equivalently, define the *directional effective width*

$$P_{\text{eff}} := r_{\mathcal{S}} + 1 \le \min\{P, d+1\}.$$

There exists a subset of P_{eff} affinely independent branch outputs spanning the same direction space $\mathcal{S}$; branches beyond this width cannot introduce new independent first-order directions unless they expand $\mathcal{S}$. They may still change the convex hull or improve conditioning, so P_{eff} is not an optimality claim for the nominal branch count. Proposition 3 further shows why indiscriminate widening need not help: for fixed variation energy, increasing reachable rank reduces the upper bound on the weakest represented singular direction. Crucially, P_{eff} depends on learned branch representations and is unavailable during frozen-backbone calibration.

Role in layer and width allocation. Equation 1 uses the normalized positive contraction to select the layer with the strongest calibrated relative commitment, while the interception argument explains why branching is placed at that layer rather than strictly after it. For a calibrated contraction level ρ_α, $\exp(\rho_\alpha)$ measures the effective-breadth contraction factor. Because the true P_{eff} depends on learned branch geometry and cannot be computed before adaptation, we use

$$P^\star = \lceil \exp(\eta \rho_\alpha) \rceil_{\mathcal{Q}_1}$$

as an entropy-calibrated proxy for sufficient parallel width. The tempering factor $0 < \eta \le 1$ allows for redundancy and saturation among branch directions, while rounding and $P_{\max}$ enforce supported hardware widths. This does not assert $P^\star = P_{\text{eff}}$ or allocation optimality.

Empirical validation. The two allocation claims are evaluated separately. For layer placement, Appendix D compares $\ell^\star$ with representative early, middle, and late alternatives; the contraction-selected layer gives the strongest downstream performance among the tested locations.

For width allocation, Table 3a exhibits the expected saturation pattern: reducing the calibrated width degrades performance, whereas widening beyond $P^\star$ provides little or no consistent improvement. This supports predictive contraction as a practical proxy for allocating sufficient—rather than maximal—internal parallelism.

Predictive concentration can reflect either correct confidence or incorrect confidence. Hidden-state aggregation followed by a nonlinear downstream network is also not a mixture of independent predictive distributions. The analysis therefore motivates *where* and *how much* to scale; downstream performance and cost measurements establish the empirical value of those choices.

B Per-Dataset Results

Tables 4–7 report the per-dataset scores underlying the domain averages in Table 2. HumanEval (HE) and MBPP are reported as Base/Plus. These detailed experimental results further demonstrate the performance of EntroBranch.

C Detailed Training and Inference Settings

C.1 Hardware and Hyperparameters

Our training and inference GPU is NVIDIA-A100-80GB-PCIe. Our CPU is Intel(R)-Xeon(R)-Platinum-8358P-CPU@2.60GHz.

During entropy-guided calibration, we set the contraction-quantile level $\alpha = 0.9$, $P_{\max} = 8$, $\mathcal{Q} = \{2, 3, 4, 5, 6, 7, 8\}$, and $\eta = 0.5$. The top-k for the number of selected tokens is 64. For calibration, we uniformly sample 384 examples from the training corpus. This sample size corresponds to a 95% confidence level and a 5% margin of error under the conservative $p = 0.5$ assumption with finite-population correction.

Table 4: Main results on **Qwen3-0.6B-Base**. HE and MBPP report Base / Plus. Beneath each Overall Avg, Δ_{EB} gives the relative improvement of EntroBranch over that row's method; beneath each FLOPs value, the relative compute increase over Single Path.

Method	Choice		Code		Math		Overall Avg (Δ_{EB} %)	FLOPs (Δ %)
	SciQ	Hella-Swag	HE	MBPP	GSM 8K	Math-500		
Single Path (p=1)	82.30	33.73	28.7 25.6	24.6 20.9	39.50	23.80	38.21 (+33.00)	1.00 –
Par Scale	82.90	34.50	29.9 27.4	36.2 31.7	43.67	22.40	41.01 (+23.92)	2.00 +100%
Beam Search	82.30	33.73	36.0 31.7	47.4 38.4	42.84	24.20	43.30 (+17.37)	1.99 +99%
Sampling (T=0.7)	82.30	33.73	26.8 24.4	29.6 25.4	47.23	30.00	41.06 (+23.77)	2.00 +100%
Sampling ($Pass@1$)	82.30	33.73	19.5 17.5	18.7 16.8	35.94	21.10	34.89 (+45.66)	1.00 –
EntroBranch *(ours)*	**85.20**	**36.07**	**37.2** **32.3**	**56.3** **48.4**	**60.12**	**36.40**	**50.82**	**1.04** +3.6%
Width P=4	85.30	38.47	36.6 29.3	55.0 46.8	59.06	33.60	50.12 (+1.40)	– –
Width P=6	85.30	37.80	36.6 31.1	56.6 48.1	59.51	34.20	50.43 (+0.77)	– –
Width P=8	85.50	38.28	37.2 33.5	55.8 47.6	59.89	35.80	51.09 (-0.53)	– –
Rand Layer	84.60	34.73	32.3 27.4	54.8 47.4	59.67	32.40	48.73 (+4.29)	– –
LORA	85.90	35.78	26.8 22.0	54.0 46.0	57.92	31.20	47.53 (+6.92)	– –
Single Path (p=1, l=48)	83.9	35.65	25.6 22.6	44.7 37.8	43.29	20.80	41.50 (+22.46)	– –
Multi Layers	84.40	36.70	37.8 32.9	55.0 46.0	58.68	39.20	50.81 (+0.00)	– –
Mean Gate	85.60	33.48	34.8 30.5	54.5 47.6	60.27	34.40	49.58 (+2.50)	– –
All cache	85.20	36.07	37.8 32.3	57.6 49.0	60.12	36.80	51.09 (-0.53)	– –

Table 8 details the branch layer and the branch number selected by EntroBranch and the proportion of trainable parameters. The table also shows that the layers selected for the Qwen3 family models are positioned later than those of the Gemma4 model. This suggests architecture-specific contraction profiles, while EntroBranch improves performance in both cases.

C.2 Training Datasets

Our training corpus consists of 220,000 samples collected from a diverse set of mathematics and code datasets. We categorize the samples solely according to their task modality, without retaining the original stage-based partitioning. Specifically, the corpus contains 100,000 mathematics samples and 120,000 code samples, accounting for 45.4545% and 54.5455% of the training data, respectively. Table 9 provides the detailed composition of the corpus, while Figure 4 illustrates its modality-level distribution.

Table 5: Main results on **Qwen3-1.7B-Base**. HE and MBPP report Base / Plus. LCB reports LiveCodeBench v6. Beneath each Overall Avg, Δ_{EB} gives the relative improvement of EntroBranch over that row's method; beneath each FLOPs value, the relative compute increase over Single Path.

Method	Choice		Code			Math			Overall Avg (Δ_{EB} %)	FLOPs (Δ %)
	MMLU Pro	Hella-Swag	HE	MBPP	LCB	GSM 8K	Math-500	AMC		
Single Path (p=1)	14.56	36.35	61.6 54.3	65.3 55.0	5.14	57.24	37.00	8.43	35.98 (+34.91)	1.00 –
Par Scale	18.09	45.82	59.1 53.7	68.0 56.3	6.29	65.60	40.80	16.87	40.58 (+19.62)	2.00 +100%
Beam Search	14.56	36.35	57.9 51.8	63.5 54.5	9.14	69.29	36.00	7.23	36.78 (+31.97)	1.99 +99%
Sampling (T=0.7)	14.56	36.35	66.5 59.1	72.5 62.7	8.57	68.01	43.60	20.48	41.12 (+18.04)	2.00 +100%
Sampling ($Pass@1$)	14.56	36.35	54.9 48.5	59.9 50.9	5.71	51.93	32.80	13.25	34.03 (+42.64)	1.00 –
EntroBranch *(ours)*	**30.93**	**54.07**	**68.9** **61.6**	**72.0** **61.6**	**10.29**	**76.42**	**47.80**	**20.48**	**48.54**	**1.07** +7.1%
Width P=2	28.21	52.93	62.2 55.5	72.5 60.8	8.57	75.13	46.40	10.84	45.53 (+6.61)	– –
Width P=4	29.76	55.08	67.1 59.8	71.7 60.3	10.29	75.97	47.80	15.66	47.58 (+2.02)	– –
Width P=6	28.96	51.80	67.1 62.2	71.7 61.9	10.29	77.18	48.60	21.69	48.06 (+0.99)	– –
Width P=8	28.98	54.58	68.9 62.2	72.0 61.4	8.57	76.50	48.60	26.87	47.90 (+1.34)	– –
Rand Layer	28.51	49.26	63.4 55.5	70.1 59.8	8.00	71.27	40.40	16.87	44.37 (+9.40)	– –

C.3 Training and Inference Parameters

We freeze the pretrained backbone and optimize only the prefix-related parameters and the aggregation module. Each branch uses 24 prefix tokens (Gemma4-E4B is 48). The prefix generator is implemented as a two-layer MLP with hidden dimension 24 (Gemma4-E4B is 48). We train for one epoch with a maximum sequence length of 2048. The per-device batch size is 1, and gradient accumulation is used to obtain an effective batch size of 256, corresponding to 256 gradient-accumulation steps per optimizer update. The learning rate is warmed up for the first $N_{\text{warm}} = 15$ optimization steps, with a peak learning rate of 1×10^{-4} and a minimum learning rate of 1×10^{-6}. The aggregation gate smoothing coefficient is 0.1. We use Adam as the optimizer, and set the context length to 2048. The Adam momentum parameters are $(\beta_1, \beta_2) = (0.9, 0.95)$, and the remaining parameters are PyTorch's defaults. The total training steps are 859.

The token-wise aggregation weights are first obtained by applying a softmax to the gate logits. We apply a small uniform smoothing coefficient $\lambda = 0.1$ before aggregating the branch representations:

$$\tilde{a}_{t,j} = (1 - \lambda)a_{t,j} + \frac{\lambda}{P},$$

where $a_{t,j}$ is the original softmax weight for branch j at token t and P is the number of branches. Equivalently, the merged representation is a convex combination of the learned gated aggregation and uniform branch averaging,

$$x_t = (1 - \lambda) \sum_{j=1}^{P} a_{t,j} x_t^{(j)} + \lambda \frac{1}{P} \sum_{j=1}^{P} x_t^{(j)}.$$

Table 6: Main results on **Qwen3-4B-Base**. LCB reports LiveCodeBench v6. BCB H reports BigCodeBench-hard. Beneath each Overall Avg, Δ_{EB} gives the relative improvement of EntroBranch over that row's method; beneath each FLOPs value, the relative compute increase over Single Path.

Method	Choice		Code		Math			Overall Avg (Δ_{EB} %)	FLOPs (Δ %)
	MMLU Pro	Hella-Swag	LCB	BCB H	AMC	AIME	Math-500		
Single Path (p=1)	17.59	70.10	13.71	15.54	25.33	1.33	51.40	28.17 (+30.99)	1.00 –
Par Scale	21.61	76.52	14.86	14.19	31.33	3.33	50.20	30.63 (+20.47)	2.00 +100%
Beam Search	17.59	70.10	18.28	14.86	21.69	1.33	50.80	28.34 (+30.20)	1.99 +99%
Sampling (T=0.7)	17.59	70.10	16.00	19.59	30.12	3.33	55.60	30.44 (+21.22)	2.00 +100%
Sampling ($Pass$@1)	17.59	70.10	12.86	11.82	21.08	1.67	45.90	26.36 (+39.98)	1.00 –
EntroBranch *(ours)*	**39.51**	**82.00**	**18.86**	**20.27**	**34.94**	**3.33**	**52.80**	**36.90**	**1.03** +2.8%
Width P=4	39.19	81.89	18.86	18.24	30.12	3.33	53.20	35.99 (+2.53)	– –
Width P=6	41.74	81.59	17.14	18.92	37.35	3.33	51.60	36.82 (+0.22)	– –
Width P=8	42.40	81.56	17.71	20.95	31.33	4.67	52.20	36.90 (+0.00)	– –
Rand Layer	34.68	77.49	16.00	17.57	32.53	2.00	51.80	33.89 (+8.88)	– –

Table 7: Main results on **Gemma4-E4B**. HE and MBPP report Base / Plus. LCB reports LiveCodeBench v6. Beneath each Overall Avg, Δ_{EB} gives the relative improvement of EntroBranch over that row's method; beneath each FLOPs value, the relative compute increase over Single Path.

Method	Choice		Code			Math		Overall Avg (Δ_{EB} %)	FLOPs (Δ %)
	MMLU Pro	Hella-Swag	HE	MBPP	LCB	GSM 8K	Math-500		
Single Path (p=1)	31.41	43.03	29.3 24.4	43.9 36.8	6.86	53.53	12.20	32.78 (+35.88%)	1.00 –
Par Scale	30.28	50.36	34.8 28.0	41.5 34.7	6.86	53.15	16.40	34.76 (+28.16%)	2.00 +100%
Beam Search	31.41	43.03	28.0 22.6	33.6 28.0	8.00	57.77	17.60	32.98 (+35.05%)	1.99 +99%
Sampling (T=0.7)	31.41	43.03	40.2 32.9	50.3 41.8	8.57	56.79	10.60	35.22 (+26.46%)	2.00 +100%
Sampling ($Pass$@1)	31.41	43.03	28.1 22.6	37.0 31.6	4.86	56.79	10.60	31.92 (+39.54%)	1.00 –
EntroBranch *(ours)*	**39.04**	**59.46**	**45.7** **39.6**	**59.8** **52.4**	**12.57**	**62.32**	**22.40**	**44.54** –	**1.05** +4.8%

Table 8: Model-specific configurations of EntroBranch. $L^\star/L$ denotes the selected branch layer relative to the total number of Transformer layers, and $P^\star$ denotes the branch width.

Model	$l^\star/L$	$P^\star$	**Prefix Len.**	**Params (%)**
Qwen3-0.6B-Base	21 / 28	2	24	0.359
Qwen3-1.7B-Base	19 / 28	3	24	0.735
Qwen3-4B-Base	29 / 36	2	24	0.328
Gemma4-E4B-It	13 / 42	3	48	0.268

Table 9: Composition of the training corpus. The percentage of each sub-dataset is calculated with respect to the complete corpus of 220,000 samples.

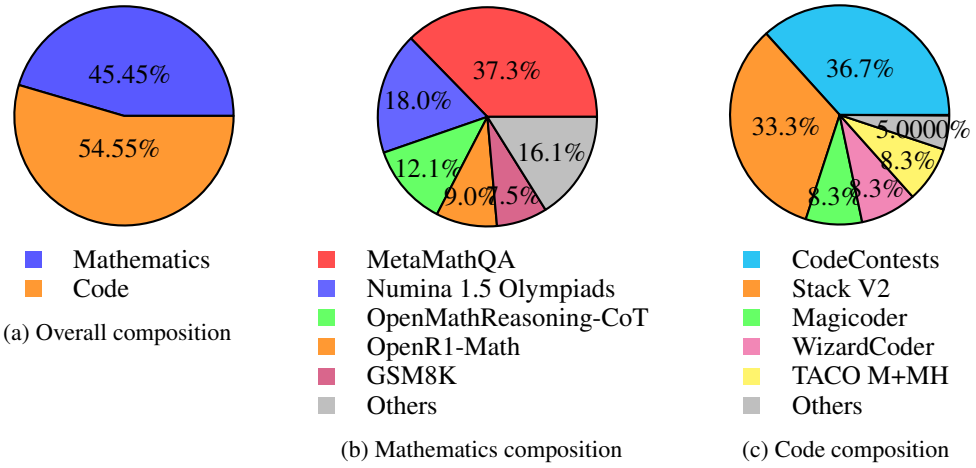
Category	**Sub-dataset**	**Samples**	**Share**
Mathematics	GSM8K-Training (Cobbe et al., 2021)	7,473	3.3968%
	MetaMathQA (Yu et al., 2023)	37,327	16.9668%
	Numina 1.5 Olympiads (Ding et al., 2026)	18,000	8.1818%
	OpenMathReasoning-CoT (Moshkov et al., 2025)	12,144	5.5200%
	OpenMathReasoning-TIR (Moshkov et al., 2025)	1,056	0.4800%
	OpenR1-Math (R1, 2025)	9,000	4.0909%
	OpenMathInstruct-2 Math (Toshniwal et al., 2024)	6,000	2.7273%
	NuminaCoT Hard (LI et al., 2024)	6,000	2.7273%
	MathInstruct Hard (Hu et al., 2026)	3,000	1.3636%
	Mathematics subtotal	**100,000**	**45.4545%**
Code	Stack V2 (Lozhkov et al., 2024)	40,000	18.1818%
	Magicoder (Wei et al., 2024)	10,000	4.5455%
	WizardCoder (Luo et al., 2024)	10,000	4.5455%
	TACO MEDIUM + MEDIUM_HARD (Li et al., 2023)	10,000	4.5455%
	TACO HARD (Li et al., 2023)	4,000	1.8182%
	TACO VERY_HARD (Li et al., 2023)	2,000	0.9091%
	CodeContests (Li et al., 2022)	44,000	20.0000%
	Code subtotal	**120,000**	**54.5455%**
	Total	**220,000**	**100.0000%**

This smoothing prevents the gate from assigning a negligibly small contribution to any branch while retaining token-wise selectivity. It also keeps the aggregation coefficients away from the boundary of the probability simplex, with $\tilde{a}_{t,j} \geq \lambda/P$, providing local margin for feasible branch reweighting as considered in Theorem 1.

C.4 Experimental Setup

In our experiments, we train various baselines using the same dataset. For Beam Search and temperature sampling, we employ Single Path models to ensure a fair comparison. ParScale is configured with a prefix length of 24 and $p = 2$ to match the minimum computational load and prefix parameters. We utilize identical hyperparameters and hardware across Single Path, ParScale, EntroBranch, and other models to maintain experimental fairness.

Regarding datasets, for choice questions, we select the answer by calculating and comparing token probabilities—choosing the option with the highest probability—which accelerated inference and completely eliminated randomness. For code generation tasks, we utilize the official Python repositories for testing. In the mathematics domain, we require models to output their reasoning process first, followed by the final answer enclosed in $\{\}$, and then verify the correctness of the result. We employ 5-shot prompting templates for both choice and mathematics tasks. For code generation, we use 0-shot prompts for Qwen3 and 3-shot prompts for Gemma4.



(a) Overall composition

(b) Mathematics composition

(c) Code composition

Figure 4: Composition of the training corpus. The mathematics and code charts retain the five largest constituent datasets and group the remaining datasets into *Others*.

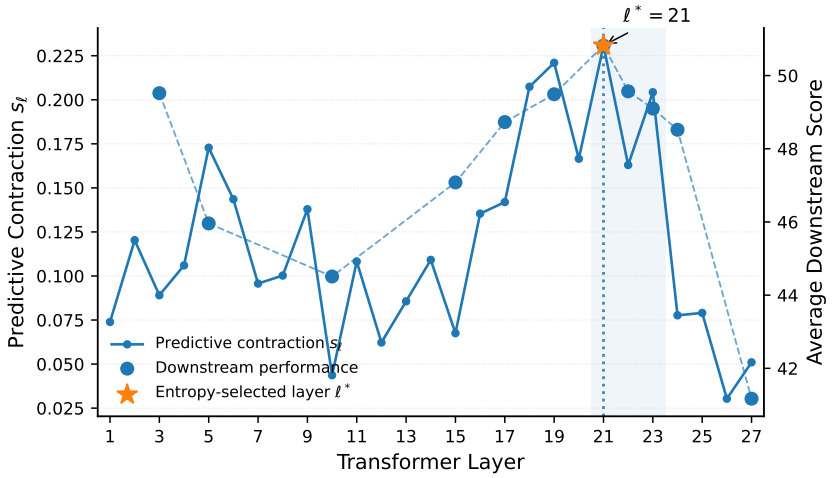


Figure 5: Branch Performance and Entropy Contraction of Layers (Qwen3-0.6B-Base).

We test Pass@2 with the sampling temperature set to 0.7. Pass@2 is a metric that measures whether a correct solution is generated in two attempts. The formula is

$$\text{Pass@2} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left\{\bigvee_{j=1}^{2} s_{i,j} \text{ passes}\right\}$$

For other baselines and EntroBranch, we test Pass@1. Pass@1 is a metric that measures whether a correct solution is generated in one attempt. The formula is

$$\text{Pass@1} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left\{s_{i,1} \text{ passes}\right\}$$

We also report the Pass@1 results at Appendix B.

D Experiment on More Branch Layers

To further examine whether predictive entropy contraction provides an informative signal for branch placement, we evaluate EntroBranch at a broader set of Transformer layers. In addition to representa-

Table 10: Results of different layers on the Qwen3-0.6B-Base. OverallAvg is the average of the average performance across the three domains.

Layer	Choice		Code		Math		Overall Avg
	SciQ	Hella Swag	HE	MBPP	GSM 8K	Math 500	
21 *(EntroBranch)*	**85.20**	**36.07**	**37.2** **32.3**	**56.3** **48.4**	**60.12**	**36.40**	**50.82**
3	84.90	36.80	36.0 31.1	55.6 47.9	56.10	34.00	49.52
5	84.90	38.08	31.1 25.6	54.5 46.8	49.96	23.80	45.96
10	83.70	34.34	32.9 28.0	51.9 44.4	49.43	21.00	44.51
15	85.20	36.07	31.7 27.4	49.5 41.8	54.97	31.00	47.08
17	84.60	34.73	32.3 27.4	54.8 47.4	59.67	32.40	48.73
19	84.60	34.41	34.1 28.7	55.3 48.1	58.83	36.00	49.49
22	85.50	35.87	34.8 29.3	52.6 43.7	60.65	35.20	49.57
23	84.60	32.47	32.3 27.4	56.3 47.9	59.14	36.40	49.10
24	84.10	32.84	32.3 27.4	55.8 48.7	60.05	32.00	48.52
27	83.60	32.08	28.0 25.0	40.2 34.1	43.90	23.80	41.17

tive early, middle, and late layers, we include layers 22 and 23 for Qwen3-0.6B-Base, and layers 20 and 21 for Qwen3-1.7B-Base, which are immediately downstream of the entropy-selected layer at layer $\ell^\star$. All configurations use the same branching mechanism and training setup in Appendix C, with only the branch location changed. This comparison therefore isolates the effect of where the additional local computation is introduced.

As shown in Table 10 and 11, the entropy-selected layer achieves the strongest downstream performance among the evaluated locations. The broader layer sweep also rules out a simple depth-based explanation: moving the branch to either substantially earlier or later layers does not reproduce the performance obtained at the selected layer. Importantly, layers 22 and 23 for Qwen3-0.6B-Base and 20 and 21 for Qwen3-1.7B-Base provide a more stringent local control. They are only one or two layers after the selected layer and therefore operate in nearly the same late-layer region, while introducing the additional branch computation only after the selected contraction layer has already occurred. Both configurations perform worse than branching directly at layer $\ell^\star$. This suggests that the gain is not explained merely by allocating additional capacity to a generally favorable late-stage region; the timing of the intervention relative to the predictive contraction also matters.

Figure 5 and 6 further compare the layer-wise predictive contraction scores with the downstream performance obtained when branching at the evaluated layers. The entropy-selected layer at layer $\ell^\star$ corresponds to the largest relative predictive contraction and also yields the highest downstream score among the evaluated branch locations. Beyond this agreement at the selected peak, predictive contraction is positively associated with branching effectiveness across the evaluated layers, with

Table 11: Results of different layers on the Qwen3-1.7B-Base. OverallAvg is the average of the average performance across the three domains.

Layer	Choice		Code			Math			Overall Avg
	MMLU Pro	Hella-Swag	HE	MBPP	LCB	GSM 8K	Math-500	AMC	
19 *(EntroBranch)*	30.93	54.07	68.9 61.6	72.0 61.6	10.29	76.42	47.80	20.48	48.54
3	28.51	49.26	63.4 55.5	70.1 59.8	8.00	71.27	40.40	16.87	44.37
5	27.13	46.22	57.3 50.0	71.7 61.6	4.57	50.64	32.00	12.05	39.09
10	23.11	37.75	59.1 51.8	68.3 56.3	11.43	66.94	41.60	9.64	37.74
15	26.11	49.51	62.2 56.1	69.0 59.0	10.29	72.87	42.80	12.05	43.90
20	28.96	52.94	64.0 56.7	71.2 59.8	10.29	75.66	46.80	16.87	46.60
21	30.06	58.87	65.2 57.3	69.0 58.5	9.71	75.44	48.20	19.28	48.01
24	26.80	45.28	67.7 61.0	73.0 61.6	10.28	76.50	45.40	18.07	45.81
27	19.63	35.79	58.5 53.0	65.6 55.8	12.00	67.02	40.60	19.28	39.66

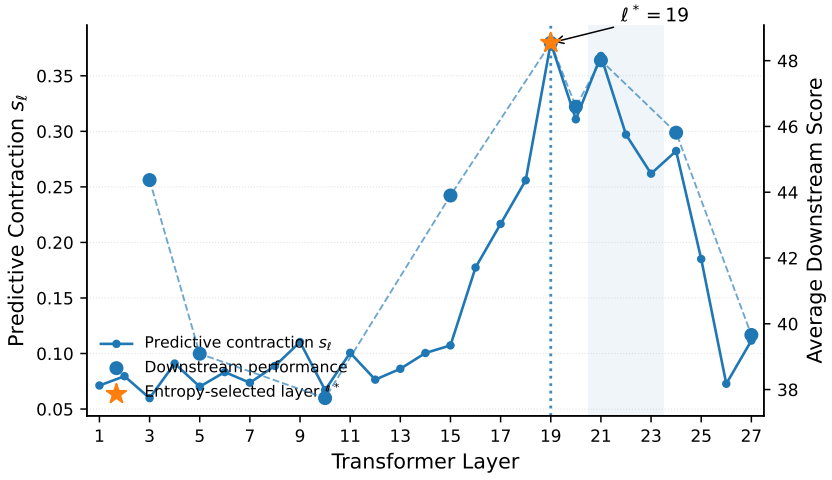
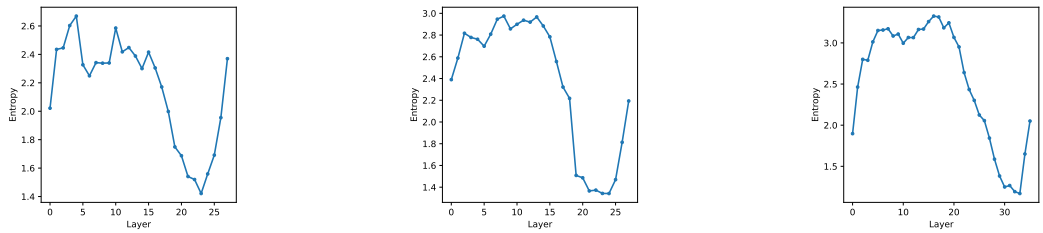


Figure 6: Branch Performance and Entropy Contraction of Layers (Qwen3-1.7B-Base).

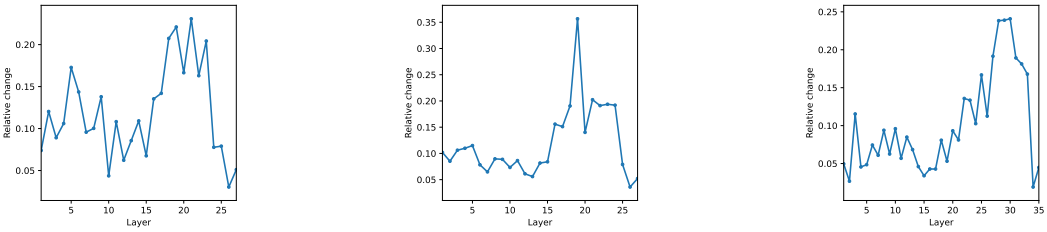
a Spearman rank correlation of $\rho = 0.7182$ ($p = 0.0128$) for Qwen3-0.6B-Base and $\rho = 0.8167$ ($p = 0.0072$) for Qwen3-1.7B-Base.

This relationship indicates that the contraction statistic contains information about the relative suitability of different branch locations, rather than identifying layer 21 and 19 only as an isolated favorable point.



(a) Qwen3-0.6B (b) Qwen3-1.7B (c) Qwen3-4B

Figure 7: Layer-wise predictive entropy around layers across different language models.



(a) Qwen3-0.6B (b) Qwen3-1.7B (c) Qwen3-4B

Figure 8: Layer-wise predictive entropy change around layers across different language models.

The nearby-layer comparison provides complementary evidence about the timing of the intervention. In EntroBranch, branching at layer $\ell^\star$ introduces alternative prefix-conditioned transformations at the layer where the predictive distribution contracts most strongly. By contrast, branching at late layers such as 22 or 23 introduces the same type of local computational breadth only after this contraction has propagated to subsequent representations. Their lower downstream performance supports the hypothesis that introducing alternative transformations at the contraction layer is more effective than delaying the same intervention to immediately downstream layers.

We emphasize that these results do not imply that predictive contraction is a direct measure of irreversible information loss, nor that later Transformer layers are incapable of modifying the consequences of an earlier contraction. The logit-lens entropy is used here as a layer-level diagnostic of predictive narrowing. Our empirical claim is therefore narrower: among the evaluated locations, stronger predictive contraction is associated with greater benefit from localized branching, and the contraction-selected layer provides a more effective intervention point than both broader layer alternatives and its immediate downstream neighbors. Together with the random-layer ablation, these results support predictive contraction as an informative empirical signal for deciding where to allocate temporary internal computational breadth.

E Detailed Entropy of Different Models

To illustrate the drastic drop in entropy during generation for single-path Transformer models, we track the detailed changes in entropy. As shown in Figures 7 and 8, despite significant differences in model architecture and pre-training data, Qwen models exhibit a sharp contraction in entropy across a few layers during generation.

These profiles motivate predictive contraction as a calibration signal. The layer and width ablations assess the resulting allocation choices; entropy profiles alone do not establish preservation of task-relevant diversity or explain the performance gains causally.

F Stability to Calibration-Set Sampling

Because both the branching layer $\ell^\star$ and branch width $P^\star$ are estimated from a sampled calibration set, we examine their sensitivity to the sampling seed (123, 234, 567). We independently resample the calibration data using different random seeds and repeat the full selection procedure while keeping the frozen backbone and selection hyperparameters unchanged.

Across all tested sampling seeds, the selected configurations remain unchanged for all three Qwen3 models. Qwen3-0.6B-Base consistently selects $\ell^\star = 21$ with $P^\star = 2$, Qwen3-1.7B-Base selects $\ell^\star = 19$ with $P^\star = 3$, and Qwen3-4B-Base selects $\ell^\star = 29$ with $P^\star = 2$. These results indicate that, under our calibration setting, the contraction-guided layer and width selections are insensitive to randomness in calibration-set sampling.

G Robustness of the top-k Predictive-Entropy Approximation

EntroBranch uses the entropy of the renormalized top-k predictive distribution, with $k = 64$, as a transition-level diagnostic. A natural question is whether restricting the entropy computation to the top-64 logits discards a substantial amount of predictive probability mass and consequently changes either the selected branching layer or the contraction-guided branch width.

We examine this issue from two complementary perspectives. First, we directly measure how much probability mass the top-64 set captures under the *full-vocabulary* softmax distribution. Second, we verify that replacing the top-64 entropy by full-vocabulary entropy does not change the resulting contraction-based decision.

G.1 Full-Softmax Probability Mass Outside the top-64

For a predictive distribution $p_{\ell,t}(v)$ at layer ℓ and token position t, let $\mathcal{V}_{\ell,t}^{64}$ denote the 64 tokens with highest logits. We define the full-softmax probability mass retained by the top-64 set as

$$M_{\ell,t}^{64} = \sum_{v \in \mathcal{V}_{\ell,t}^{64}} p_{\ell,t}(v), \tag{14}$$

and the residual probability mass as

$$R_{\ell,t}^{64} = 1 - M_{\ell,t}^{64} = \sum_{v \notin \mathcal{V}_{\ell,t}^{64}} p_{\ell,t}(v). \tag{15}$$

Unlike the top-64 entropy used by our contraction diagnostic, $M_{\ell,t}^{64}$ and $R_{\ell,t}^{64}$ are computed using the *full-vocabulary softmax normalization*. Thus, $R_{\ell,t}^{64}$ is exactly the probability of sampling a token outside the top-64 under unconstrained categorical sampling.

We additionally measure the probability of the rank-65 token, denoted by $p_{\ell,t}^{(65)}$. Because it is the highest-probability token excluded from the top-64 set,

$$p_{\ell,t}(v) \le p_{\ell,t}^{(65)}, \qquad \forall v \notin \mathcal{V}_{\ell,t}^{64}. \tag{16}$$

Therefore, $p_{\ell,t}^{(65)}$ provides an upper bound on the probability of every individual token omitted from the entropy calculation.

We evaluate these statistics over $36{,}651$ valid next-token positions for Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B. Table 12 reports the statistics immediately before and after the contraction-selected layer.

The behavior is consistent across all three model scales. The selected layer increases the mean probability mass captured by the top-64 from 96.82% to 98.04% for Qwen3-0.6B, from 90.90% to

Table 12: Full-softmax probability coverage of the top-64 predictive set around the contraction-selected layer. Each entry reports the value before and after the selected layer. Although the entropy diagnostic itself renormalizes the top-64 logits, the probability masses in this table are computed using the complete vocabulary softmax.

Model	$\ell^\star$	Mean M^{64}	Median M^{64}	Median R^{64}	Median $p^{(65)}$
Qwen3-0.6B	21	0.9682 $\to 0.9804$	0.9981 $\to 0.9996$	1.93×10^{-3} $\to 3.72 \times 10^{-4}$	2.66×10^{-5} $\to 5.60 \times 10^{-6}$
Qwen3-1.7B	19	0.9090 $\to 0.9538$	0.9696 $\to 0.9976$	3.04×10^{-2} $\to 2.40 \times 10^{-3}$	2.61×10^{-4} $\to 3.00 \times 10^{-5}$
Qwen3-4B	29	0.9664 $\to 0.9758$	0.9979 $\to 0.9995$	2.06×10^{-3} $\to 4.62 \times 10^{-4}$	2.40×10^{-5} $\to 6.33 \times 10^{-6}$

95.38% for Qwen3-1.7B, and from 96.64% to 97.58% for Qwen3-4B. The corresponding median top-64 coverage after contraction is 99.96%, 99.76%, and 99.95%, respectively.

More importantly, the residual mass decreases sharply at the selected layer. The median probability outside the top-64 decreases by approximately 80.8%, 92.1%, and 77.6% for Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B, respectively. Thus, the predictive contraction identified by our entropy statistic is also visible directly in the full-softmax probability distribution.

The same conclusion holds at the level of individual omitted tokens. The median probability of the rank-65 token decreases from 2.66×10^{-5} to 5.60×10^{-6} on Qwen3-0.6B, from 2.61×10^{-4} to 3.00×10^{-5} on Qwen3-1.7B, and from 2.40×10^{-5} to 6.33×10^{-6} on Qwen3-4B. Since the rank-65 token is the most probable token outside the retained set, every other excluded token has an even smaller probability.

At the post-contraction layer, the rank-65 probability is below 10^{-4} for 75.46%, 63.57%, and 78.97% of evaluated positions for Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B, respectively. Likewise, the top-64 set contains at least 99% of the full-softmax mass at 77.72%, 64.63%, and 79.65% of positions.

These observations should not be interpreted as claiming that the total tail mass is identically zero at every position. A small subset of positions remains comparatively diffuse. Rather, the results show that the top-64 set captures the dominant predictive alternatives at typical positions, while individual omitted tokens carry very small probability. Moreover, the selected predictive-contraction layer consistently moves probability mass *toward*, rather than away from, the retained top-64 set.

G.2 Robustness to Full-Vocabulary Entropy

The probability-mass analysis above shows that the top-64 set contains the dominant predictive alternatives at the contraction-selected layer. We further test the top-64 approximation directly by recomputing the predictive entropy over the complete vocabulary.

Specifically, our default diagnostic uses the renormalized top-64 entropy

$$H^{64}_{\ell,t} = -\sum_{v \in \mathcal{V}^{64}_{\ell,t}} \widetilde{p}_{\ell,t}(v) \log \widetilde{p}_{\ell,t}(v), \tag{17}$$

where

$$\widetilde{p}_{\ell,t}(v) = \frac{p_{\ell,t}(v)}{\sum_{u \in \mathcal{V}^{64}_{\ell,t}} p_{\ell,t}(u)}. \tag{18}$$

As a robustness check, we replace it with the full-vocabulary entropy

$$H^{\text{full}}_{\ell,t} = -\sum_{v \in \mathcal{V}} p_{\ell,t}(v) \log p_{\ell,t}(v), \tag{19}$$

Table 13: Robustness of contraction-guided allocation to the entropy approximation. Replacing the renormalized top-64 entropy with full-vocabulary entropy leaves both the selected contraction layer $\ell^\star$ and the resulting branch width $P^\star$ unchanged across all three Qwen3 scales.

Model	top-64 $\ell^\star$	Full-vocab. $\ell^\star$	top-64 $P^\star$	Full-vocab. $P^\star$
Qwen3-0.6B	21	21	2	2
Qwen3-1.7B	19	19	3	3
Qwen3-4B	29	29	2	2

and recompute the complete contraction-based selection procedure.

Importantly, using full-vocabulary entropy produces the same contraction-selection outcome as the top-64 approximation across all three Qwen3 model scales. The selected layer remains unchanged for Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B. The resulting contraction-guided branch-width allocation is also unchanged.

This result is stronger than requiring the truncated entropy to numerically match the full-vocabulary Shannon entropy. EntroBranch uses predictive entropy operationally to identify where a sharp contraction occurs and how much local computational breadth to allocate. Therefore, the relevant robustness criterion is whether the entropy approximation changes these downstream decisions. Table 13 shows that it does not.

Together with the full-softmax probability analysis in Table 12, this provides two complementary forms of evidence. First, the top-64 set captures the dominant predictive alternatives at typical positions, while individual omitted tokens have very small probability. Second, even when the omitted vocabulary is included exactly in the entropy calculation, the resulting contraction-based decisions remain unchanged. Thus, the top-64 approximation preserves the decision-relevant contraction structure used by EntroBranch.

G.3 Robustness of Contraction-Guided Branch Width

The robustness of the entropy diagnostic also propagates to the contraction-guided branch-width rule. Recall that EntroBranch computes

$$\rho_\alpha = \mathrm{Quantile}_\alpha \left\{ \delta_{\ell^\star,t} : (x,t) \in \mathcal{T}_\Omega \right\}, \tag{20}$$

and determines the branch width as

$$P^\star = \left\lceil e^{\eta \rho_\alpha} \right\rceil_{\mathcal{Q}_1}, \qquad 0 < \eta \leq 1. \tag{21}$$

Suppose an alternative entropy estimator perturbs the contraction quantile from ρ_α to $\rho_\alpha + \Delta\rho$. Before discrete rounding, the corresponding continuous widths satisfy

$$\frac{\widetilde{P}'}{\widetilde{P}} = \frac{e^{\eta(\rho_\alpha + \Delta\rho)}}{e^{\eta\rho_\alpha}} = e^{\eta\Delta\rho}. \tag{22}$$

For small $\Delta\rho$,

$$e^{\eta\Delta\rho} = 1 + \eta\Delta\rho + \mathcal{O}\left((\Delta\rho)^2\right), \tag{23}$$

so the first-order effect of the perturbation is scaled by η. Moreover, the deployed branch width is rounded to the discrete supported set $\mathcal{Q}_1$. Hence, $P^\star$ is piecewise constant with respect to ρ_α: small changes in the entropy statistic have exactly zero effect on the final branch count unless they cross an allocation boundary.

Our full-vocabulary experiment directly confirms this robustness. Although the full-vocabulary entropy includes all omitted tail tokens, it produces the same $P^\star$ as the top-64 approximation for all

three Qwen3 models. Therefore, the approximation used in the main experiments does not affect the actual amount of computation allocated by EntroBranch.

Overall, these results show that the top-64 entropy approximation is robust for the role it plays in EntroBranch. The retained set contains the dominant predictive alternatives at typical positions, predictive contraction further concentrates probability into this set, and replacing the approximation with exact full-vocabulary entropy leaves both the selected layer and the final branch width unchanged. Hence, the top-64 approximation reduces diagnostic cost without changing the resulting internal-compute allocation in our experiments.

H Details of Mechanistic Analysis

This appendix provides the definitions and evaluation procedures for the mechanistic analyses reported in Section 4.8. Following the structure of that section, we examine three aspects of the proposed mechanism: (1) *predictive concentration as a scaling signal*, including whether the contraction-selected layer remains strongly contractive after adaptation; (2) *branch differentiation and token-wise gating*, including whether branch-specific prefixes produce non-redundant hidden states and whether the learned gate uses these branches selectively across tokens; and (3) *gradient alignment and optimization*, including whether the subspace spanned by branch differences captures directions of the downstream supervised-loss gradient and admits feasible branch reweightings that reduce the loss. Unless otherwise specified, these representation-level analyses are conducted on Qwen3-0.6B-Base with HumanEval.

H.1 Predictive Concentration as a Scaling Signal

Section 4.8 uses two complementary diagnostics to evaluate predictive concentration as a scaling signal. The first, reported in Section 4.5, examines whether stronger pre-adaptation contraction is associated with better downstream performance when branching is applied at different layers. Here, we focus on the complementary question of whether the contraction-selected region remains salient after adaptation.

Following Section 3, for token t and Transformer layer ℓ, we compute the top-k predictive entropy $H_{\ell,t}^{(k)}$ using the frozen logit lens and define the absolute entropy drop as

$$\delta_{\ell,t} = \left[H_{\ell-1,t}^{(k)} - H_{\ell,t}^{(k)} \right]_+ . \tag{24}$$

For each layer, we average $\delta_{\ell,t}$ over valid next-token positions in the evaluation data. We compute the same statistic for the pretrained backbone and for the single-path prefix-tuned model, while keeping the logit-lens construction unchanged.

This comparison is not used to re-select the branching layer after training. The branching layer $\ell^\star$ is determined once from the pretrained model during calibration and remains fixed throughout adaptation. Instead, the post-adaptation measurement tests whether the region identified before training continues to exhibit a pronounced predictive contraction after the model has been adapted. The resulting layer-wise values are reported in Table 3f.

H.2 Branch Differentiation and Token-Wise Gating

We next test the second mechanistic claim in Section 4.8: whether localized branching produces non-redundant hidden states and whether the learned token-wise gate selectively uses these branch-specific representations. At the selected layer $\ell^\star$, let $x_{\ell^\star,t}^{(j)} \in \mathbb{R}^d$ denote the output of branch j for token t before branch aggregation.

Branch similarity. For $P = 2$, we measure the cosine similarity between the two branch representations,

$$\mathrm{CosSim}_t = \frac{\left\langle x_{\ell^\star,t}^{(1)}, x_{\ell^\star,t}^{(2)} \right\rangle}{\left\| x_{\ell^\star,t}^{(1)} \right\|_2 \left\| x_{\ell^\star,t}^{(2)} \right\|_2}. \tag{25}$$

A similarity substantially below one indicates that the independently learned prefixes cause the same frozen Transformer layer to produce distinct local representations rather than redundant copies.

Gate selectivity. Let $a_t = (a_{t,1}, \ldots, a_{t,P})$ denote the softmax gate probabilities before the uniform smoothing described in Appendix C. We analyze the pre-smoothing probabilities so that the statistics reflect the selectivity learned by the gate itself rather than the imposed smoothing floor.

For each token, we compute the normalized gate entropy

$$\bar{H}_{\mathrm{gate},t} = -\frac{1}{\log P} \sum_{j=1}^{P} a_{t,j} \log a_{t,j}, \tag{26}$$

which lies in $[0, 1]$. A value near zero indicates that the gate concentrates most of its mass on one branch, whereas a value near one corresponds to nearly uniform aggregation. We additionally report

$$p_t^{\max} = \max_j a_{t,j}, \tag{27}$$

together with the fraction of tokens for which $p_t^{\max} > 0.95$.

To determine whether one branch is used exclusively or whether the auxiliary branch contributes selectively, we also record the average gate mass assigned to each branch and the fraction of tokens for which each branch has the largest gate probability. These statistics distinguish a degenerate duplicated-path solution from a dominant-path-plus-correction-path pattern in which one branch is preferred globally but the other is selected for a non-negligible subset of tokens.

H.3 Gradient Alignment: Sequence-Level Branch-Variation Subspace

We next evaluate the third mechanistic claim in Section 4.8: whether the directions made reachable by branch reweighting align with directions relevant to reducing the downstream supervised loss. We first define the sequence-level branch-variation subspace used by the gradient-alignment diagnostics in Sections H.4–H.5.

Theorem 1 characterizes the locally reachable directions obtained by reweighting fixed branch outputs for a single token. We use its sequence-level extension to test whether the learned branch directions are aligned with the downstream supervised-loss gradient.

For token t, define the centered branch matrix

$$Z_{c,t} = \left[x_t^{(1)} - \bar{x}_t, \ldots, x_t^{(P)} - \bar{x}_t \right], \qquad \bar{x}_t = \frac{1}{P} \sum_{j=1}^{P} x_t^{(j)}, \tag{28}$$

and let

$$S_t = \mathrm{Col}(Z_{c,t}), \qquad \Pi_t = Z_{c,t} Z_{c,t}^{\dagger} \tag{29}$$

be the branch-variation subspace and its orthogonal projector. For a sequence with T evaluated token positions, we stack these token-wise spaces as

$$S_{\mathrm{seq}} = \bigoplus_{t=1}^{T} S_t, \qquad \Pi_{\mathrm{seq}} = \mathrm{blkdiag}(\Pi_1, \ldots, \Pi_T). \tag{30}$$

Let

$$g_{\mathrm{seq}} = \nabla_{x_{\mathrm{seq}}} \mathcal{L}(x_{\mathrm{seq}}) \tag{31}$$

denote the gradient of the downstream supervised loss with respect to the aggregated representations at the selected layer. The component of this gradient reachable through local branch reweighting is

$$v_{\mathrm{seq}} = \Pi_{\mathrm{seq}} g_{\mathrm{seq}}. \tag{32}$$

This construction keeps the learned branch representations fixed and asks a local geometric question: how much of the loss gradient lies in directions that can be produced by changing the branch mixture? It therefore evaluates the condition underlying Theorem 1, rather than simulating another round of prefix training.

H.4 Gradient Alignment: Projected Gradient and Oracle Correction

We quantify the overlap between the branch-variation subspace and the supervised-loss gradient using the projected gradient energy

$$G_{\mathrm{proj}} = \|\Pi_{\mathrm{seq}} g_{\mathrm{seq}}\|_2^2 = \|v_{\mathrm{seq}}\|_2^2 . \tag{33}$$

A larger G_{proj} indicates that a larger absolute component of the loss gradient lies within the locally reachable branch-variation space.

Projection alone does not guarantee a practically useful loss decrease, because the usable step is constrained by the convex hull of the branch representations and by the local loss geometry. We therefore additionally evaluate an oracle correction along the ideal projected-gradient direction. Let Γ_{feas} denote the set of step sizes for which the sequence-level update

$$x_{\mathrm{seq}}(\gamma) = x_{\mathrm{seq}} - \gamma v_{\mathrm{seq}} \tag{34}$$

can be realized by feasible token-wise branch reweighting. We define

$$G_{\mathrm{oracle}} = \max_{\gamma \in \Gamma_{\mathrm{feas}}} \left[\mathcal{L}(x_{\mathrm{seq}}) - \mathcal{L}\left(x_{\mathrm{seq}} - \gamma v_{\mathrm{seq}}\right) \right]. \tag{35}$$

Thus, G_{oracle} measures the loss reduction available along the ideal correction direction while respecting the branch-mixture feasibility constraint. This is an oracle diagnostic rather than the update actually produced by the learned gate; its purpose is to test whether the learned branch geometry contains useful local correction directions.

To examine whether this geometry is specific to the contraction-selected location, we compute the same quantities at the selected layer $\ell^\star = 21$ and at the nearby comparison layer $\ell = 24$, while keeping the diagnostic construction unchanged. The values reported in Section 4.8 show larger projected-gradient energy and oracle correction at the selected transition.

H.5 Gradient Alignment: Enrichment over Random Subspaces

A non-zero projected gradient can arise simply because any non-empty subspace captures some component of a high-dimensional gradient. We therefore compare the learned branch-variation subspace against an equal-rank isotropic random control.

For a sequence-level subspace S_{seq}, we measure the normalized fraction of gradient energy captured by the subspace,

$$R_{\mathrm{grad}} = \frac{\|\Pi_{\mathrm{seq}} g_{\mathrm{seq}}\|_2^2}{\|g_{\mathrm{seq}}\|_2^2}. \tag{36}$$

The random control uses a subspace with the same rank as the learned branch subspace but without using the learned branch directions. We apply the same projection-energy metric to this control. Comparing the two controls for the dimensionality of the reachable space: an enrichment above the random baseline indicates that the learned branch directions preferentially align with the supervised-loss gradient rather than capturing gradient energy merely because they span additional dimensions.

For Qwen3-0.6B-Base with $P = 2$, the per-token branch-variation space has rank at most one. As reported in Section 4.8, the learned rank-one branch direction captures substantially more normalized gradient energy than the equal-rank random control.

H.6 Optimization Evidence: Training-Loss Comparison

Finally, we compare the training objective achieved by EntroBranch with that of single-path prefix tuning under the corresponding training setup. This measurement does not by itself establish the geometric mechanism above, since the prefixes and gate are jointly optimized during training. Instead, it provides complementary evidence that the additional branch directions offer useful optimization flexibility in practice.

The training losses are reported in Table 3g. Across the evaluated Qwen3 models, EntroBranch reaches a lower training loss than the corresponding single-path prefix-tuning baseline. Together with the gradient-projection diagnostics, this observation is consistent with the interpretation that learned branch variation provides task-relevant local correction directions.

Table 14: Sensitivity of calibrated branch width $P^\star$ to the quantile level α and tempering factor η on Qwen3-1.7B-Base. The default setting is shown in **bold**.

$\eta \backslash \alpha$	**0.85**	**0.90**	**0.95**
0.45	2	3	3
0.50	3	**3**	3
0.55	3	3	4

I Stability of Contraction-Guided Width Allocation

EntroBranch determines the branch width from the contraction statistic ρ_α through Eq. (2), using the quantile level α and tempering factor η. In the main experiments, we use $\alpha = 0.90$ and $\eta = 0.50$. The former emphasizes strong but non-extreme contraction, while the latter tempers the exponential mapping from contraction magnitude to branch width so that additional computation remains modest.

To evaluate the sensitivity of this allocation rule, we vary both hyperparameters around their default values on Qwen3-1.7B-Base. We focus on this model because its default calibrated width is $P^\star = 3$, making changes in the allocated branch width directly observable. Specifically, we test $\alpha \in \{0.85, 0.90, 0.95\}$ and $\eta \in \{0.45, 0.50, 0.55\}$ while keeping all other calibration settings fixed.

Table 14 shows that the calibrated allocation is locally stable around the default setting. Seven of the nine nearby configurations select the same width $P^\star = 3$, while the remaining two change only to the adjacent widths 2 and 4. The variation is also monotonic: increasing either α or η never decreases the allocated width, consistent with the mapping in Eq. (2). Thus, moderate perturbations of either hyperparameter do not cause abrupt changes in the amount of computation allocated by EntroBranch.

The two deviations also correspond only to neighboring branch widths already covered by the branch-width ablation in the main paper. On Qwen3-1.7B-Base, the default $P^\star = 3$ achieves an average score of 48.54, compared with 45.53 for $P = 2$ and 47.58 for $P = 4$. Larger fixed widths provide no consistent improvement while incurring higher FLOPs. These results indicate that $(\alpha, \eta) = (0.90, 0.50)$ provides a stable operating point that preserves the intended trade-off between sufficient localized scaling and modest additional computation.