WHAT DOES A TOKEN COST? A MIXTURE-OF-AGENTS MEASUREMENT OF SUFFICIENT PER-TOKEN COMPUTE

Anonymous authors
Paper under double-blind review

ABSTRACT

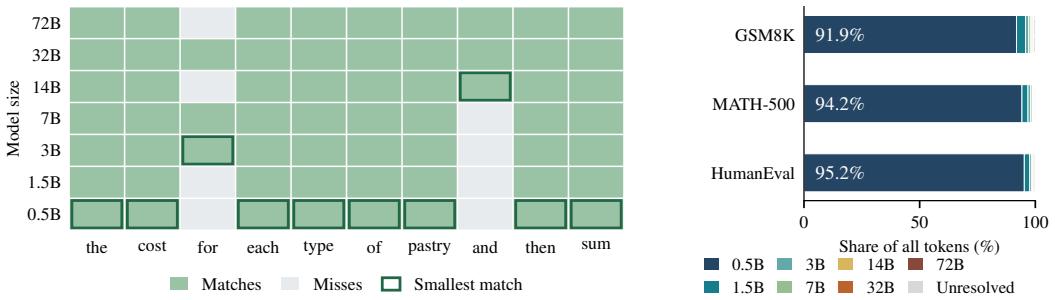
Large language models spend the same amount of computation on every token they generate, regardless of how difficult each token is to produce. Methods such as speculative decoding and model routing are built on the premise that much of this computation is unnecessary, yet the computation an individual token actually requires has not been measured. We measure it through a Mixture-of-Agents (MoA) lens: a panel of fifteen language models of increasing capacity, drawn from three families, in which every agent attempts to reproduce a reference sequence token by token, conditioned on the correct preceding tokens. We define the inference cost of the smallest agent that succeeds as the token's sufficient compute, which upper-bounds what the token requires. On three core benchmarks, a 0.5B agent reproduces 92–95% of reference tokens. Across Qwen, OLMo, and R1-distilled panels, the most expensive 10% account for 64–80% of estimated FLOPs. On all 500 MATH-500 problems, the MoA-derived map helps model routing reduce projected latency from 7.59 to 5.12 seconds while slightly improving accuracy, relative to the best confidence-routing baseline. The MoA-map helps drafting use 32.6% fewer draft tokens and approximately 20% lower projected latency than fixed-window drafting at similar accuracy. These comparisons reveal remaining allocation headroom, motivating controllers that exploit sufficient-compute structure.

1 INTRODUCTION

Large language models (LLMs), such as the GPT series, have made a great impact for tasks ranging from writing to programming (Brown et al., 2020; Achiam et al., 2023). Their adoption produces recurring computational expense at a large scale: by July 2025, ChatGPT alone received more than 2.5 billion user messages per day (Chatterji et al., 2025). Therefore, efficiently serving these requests is becoming critical (Pope et al., 2023; Kwon et al., 2023). One approach to reducing this computational cost is to use only as much computation as is necessary. *Speculative decoding* lets a small model propose several tokens for a larger model to verify together (Leviathan et al., 2023; Chen et al., 2023a). *Model cascades and routers* choose among models according to the input (Chen et al., 2023b; Ong et al., 2025), while collaborative decoding makes that choice within a response (Shen et al., 2024; Fu et al., 2025). *Adaptive-depth* methods similarly vary the layers evaluated for each prediction (Schuster et al., 2022; Raposo et al., 2024). These approaches exploit the possibility that a cheaper computation can supply a useful prediction. Their gains depend on how often this is possible. However, running the model alone does not reveal when less computation would have been sufficient.

This motivates our central question: *How much computation is sufficient for each token in a successful response?* A response-level accuracy or cost average hides where that computation is needed. Even a high agreement rate between a small and a large model leaves open how much computation the remaining disagreements require. Existing measurements of oracle module allocation, small/large-model disagreement, and learned layer use establish that computation can vary within a response (Alizadeh-Vahid et al., 2024; Zhang et al., 2024; Luo et al., 2025). We measure this variation by identifying, for each token, the smallest model that can successfully reproduce it. We then examine whether this token-level information can improve existing efficient serving methods.

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.



(a) Which models reproduce each reference token?

(b) The 0.5B model reproduces most tokens

Figure 1: Measuring which model suffices at each token. (a) Each column is a token from a GSM8K response. Every model receives the same reference prefix; green cells reproduce the next token and outlines mark the smallest matching model. (b) Within the Qwen2.5 family, the 0.5B model reproduces 91.9–95.2% of tokens. The "unresolved" means that none matches.

Mixture-of-Experts (MoE) has been widely known, which selects subnetworks inside a model (Shazeer et al., 2017; Fedus et al., 2022). Mixture-of-Agents (MoA)combines different language models, either through response aggregation (Wang et al., 2025) or through agent selection during decoding (Chakraborty et al., 2025; Zheng et al., 2025). Our measurement uses token-level MoA. Given a reference sequence, we query a panel of fifteen models from Qwen2.5, OLMo-2, and DeepSeek-R1-Distill-Qwen at each token. The FLOPs of the cheapest agent whose prediction matches the next token are that token's *sufficient compute*. A successful agent supplies an observed upper bound on the computation cost of that token.

Figure 1 provides an overview of our measurement. Within the Qwen2.5 panel, the 0.5B agent reproduces 91.9–95.2% of reference tokens on GSM8K, MATH-500, and HumanEval. The remaining tokens are few but expensive. In all three panels, the most expensive 10% of tokens account for 64–80% of estimated FLOPs. Families also agree on which positions are expensive. After controlling for surprisal and position, the reproduction orderings of Qwen2.5, OLMo-2, and Falcon3 families have pairwise partial rank correlations of 0.32–0.48. This agreement makes the map informative beyond the behavior of a single model family. Details will be discussed in Section 4.1.

We then ask whether this map would help existing methods allocate computation. On all 500 MATH-500 problems, we give a routing controller and a speculative-decoding controller oracle access to the map of 0.5B agreement along a 72B reference response. Routing to 0.5B at positions the map marks as reproducible reaches 69.6% accuracy at a projected latency of 5.12 seconds, compared with 68.4% and 7.59 seconds for the best confidence-threshold cascade. Setting each draft length to the map's run of reproducible positions proposes 32.6% fewer draft tokens than a fixed 16-token window and lowers projected latency by about 20% at similar accuracy. Our contributions can be summarized as:

1. We formalize sufficient compute through token-level MoA and introduce a panel-based method for measuring it along verified responses.

2. We show that a sparse tail of tokens carries most of the estimated compute and that three families share a reproduction ordering.

3. We evaluate how routing and speculative decoding exploit the measurement. Both improve projected efficiency at comparable accuracy over baselines. The difference is allocation headroom that current controllers leave unrealized.

In this paper, we first define sufficient compute (Section 2) and derive a token-level MoA procedure for measuring it (Section 3). The experiments then examine where compute is concentrated, whether cheap reproduction supports continued generation, and how existing efficiency methods exploit this information (Section 4). Section 5 places our study in the context of prior work. Section 6 discusses the implications and limitations of our findings, and Section 7 concludes the paper.

2 PROBLEM FORMULATION AND BACKGROUND

Sufficient compute at a reference prefix Let x be a prompt and $y^\star = (y_1^\star, \ldots, y_T^\star)$ a reference accepted by a binary task checker $V(x, y) \in \{0, 1\}$, so $V(x, y^\star) = 1$. At position t, the reference state is $s_t^\star = (x, y_{<t}^\star)$: the prompt and all preceding reference tokens. A model panel $\Pi_F = \{\pi_{F,1}, \ldots, \pi_{F,K_F}\}$ contains next-token distributions $\pi_{F,k}(v \mid s)$ over a shared vocabulary $\mathcal{V}$. The index F identifies the panel. Let $c_{F,k}(s) > 0$ be the FLOPs of model k's next-token prediction at prefix s. A prediction-success rule $M(\pi, s, v) \in \{0, 1\}$ specifies when a model succeeds on target token v at state s. We require the model's prediction to equal the reference token.

Definition 1 (Panel-relative sufficient compute)**.** For reference token $y_t^\star$, define the successful agents and their minimum prediction cost by

$$\mathcal{A}_F^M(t) = \{k : M(\pi_{F,k}, s_t^\star, y_t^\star) = 1\}, \qquad d_F^M(t) = \min_{k \in \mathcal{A}_F^M(t)} c_{F,k}(s_t^\star). \tag{1}$$

We set $\min \varnothing = +\infty$ and call such a position *unresolved*. Appendix C.1 reports how often this occurs and how such positions enter the cost summaries. The sequence $d_F^M(1{:}T)$ is the sequence's sufficient-compute map. We write $d_F(t)$ when the success rule is fixed.

It serves as an upper bound since adding agents can only tighten the bound. The measured map depends on the reference trajectory, and another correct response may have a different cost profile.

Speculative decoding and Routing The map makes precise a local opportunity shared by several efficiency methods: a prediction produced by a large model may also be available from a cheaper one. Speculative decoding uses a draft model to propose multiple tokens and verifies them with a target model in parallel (Leviathan et al., 2023; Chen et al., 2023a). Its distribution-preserving acceptance rule depends on draft/target probabilities, and its speed benefit depends on accepted tokens per verification round and the relative cost of drafting and verification. Greedy reference agreement gives a concrete local comparison; speculative sampling uses the full distributions to decide acceptance.

Routing changes which model supplies the output. FrugalGPT and RouteLLM select models at the request level (Chen et al., 2023b; Ong et al., 2025); collaborative decoding extends selection to individual steps (Shen et al., 2024; Zheng et al., 2025; Fu et al., 2025). At that granularity, a cheaper prediction also changes the prefix on which later choices are made. A cost map measured at reference states consequently supplies information about those states, while a deployed controller must act along the states it generates. This distinction motivates both the exact-reference selector and the later continuation experiments.

From MoE to token-level MoA Mixtures provide a way to express the choice of computation. An MoE layer maps a token representation h to h' by combining J expert functions E_j through a gate w_j:

$$h' = \sum_{j=1}^{J} w_j(h) E_j(h), \qquad w_j(h) \geq 0, \quad \sum_{j=1}^{J} w_j(h) = 1. \tag{2}$$

The experts are subnetworks within a model (Jacobs et al., 1991); sparse gates activate only a subset at each token (Shazeer et al., 2017; Fedus et al., 2022). This separates parameter capacity from the computation performed for a particular prediction.

MoA makes language models the available policies. In response-level MoA, agents produce candidate answers and an aggregator conditions on them, possibly across several rounds (Wang et al., 2025). Token-level MoA selects an agent's next-token policy at a shared decoding state (Shen et al., 2024; Chakraborty et al., 2025; Zheng et al., 2025). For our measurement, this gives a direct interpretation of $\mathcal{A}_F^M(t)$: it is the set of agents that can supply the required prediction at the current reference state. Section 3 derives the controller that selects among those agents.

3 MEASURING SUFFICIENT COMPUTE WITH TOKEN-LEVEL MOA

To connect sufficient compute to sequence generation, we treat each model as a policy and model selection as a decision made at every token. We first formulate this MoA process, then derive the minimum-cost controller for reproducing a verified response.

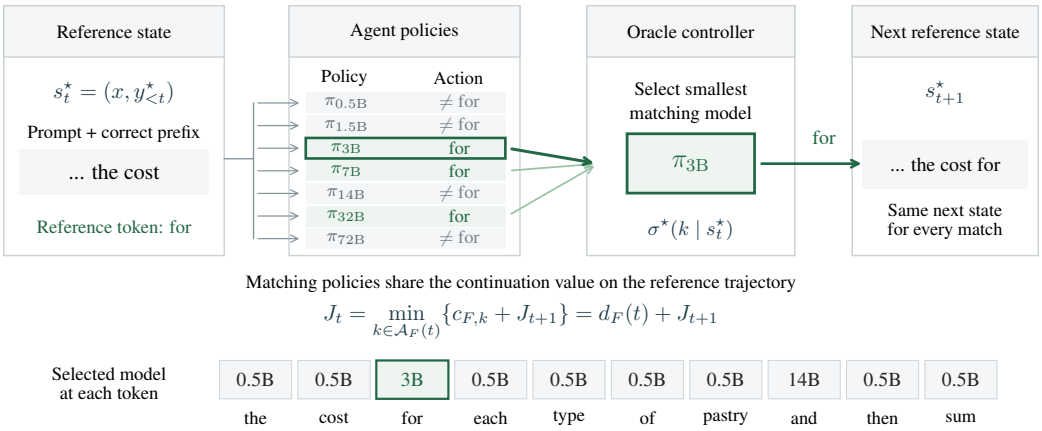


Figure 2: Minimum-cost policy selection. At the illustrated prefix, the 3B, 7B, and 32B agents all reproduce the reference token "for" and reach the same next state. Their remaining cost J_{t+1} is therefore identical, so the controller selects the agent with the lowest immediate prediction FLOPs: 3B in this example. Repeating this selection gives the model sequence shown along the bottom.

Token-level MoA as a Markov Decision Process. Decoding can be represented as a token-level Markov decision process (Ranzato et al., 2015; Bahdanau et al., 2017; Mudgal et al., 2024; Chakraborty et al., 2024). At step t, the state $s_t = (x, y_{<t})$ contains the prompt and the tokens generated so far. The action is the next token v. Appending this token deterministically produces the next state, $s_{t+1} = (x, y_{<t}v)$. The state contains all information used by the next-token policies, satisfying the Markov property. Generation terminates upon emitting an end-of-sequence token or reaching a prescribed maximum length. The resulting response y constitutes a *rollout*, evaluated by the terminal reward $V(x, y)$.

In token-level MoA, a controller chooses which model supplies the next-token policy (Chakraborty et al., 2025). Each agent $\pi_{F,k} \in \Pi_F$ defines a distribution over the shared vocabulary. Let $\sigma(k \mid s)$ denote the probability of selecting agent k at state s, where $\sigma(k \mid s) \geq 0$ and $\sum_{k=1}^{K_F} \sigma(k \mid s) = 1$. The controller first selects an agent and then samples a token from that agent's policy. Marginalizing over the selected agent gives

$$\pi_\sigma(v \mid s) = \sum_{k=1}^{K_F} \sigma(k \mid s) \pi_{F,k}(v \mid s). \tag{3}$$

Thus, the controller selects among model policies, while the selected policy determines the next token. Different agents can produce different tokens and therefore lead to different subsequent states.

We measure prediction cost using the FLOP functions $c_{F,k}(s)$ defined in Section 2. We consider an idealized setting in which each agent's prefix cache is available and switching between agents incurs no additional cost. Let K_t denote the agent selected at step t, and let $T(Y)$ be the length of the generated response Y. For an expected compute budget B, the controller's objective is

$$\max_\sigma \ \mathbb{E}_\sigma[V(x, Y)] \quad \text{subject to} \quad \mathbb{E}_\sigma \left[\sum_{t=1}^{T(Y)} c_{F,K_t}(s_t) \right] \leq B. \tag{4}$$

The expectation accounts for both agent selection and token generation. An agent choice affects its immediate prediction cost and the continuation from the resulting prefix.

Exact-reference selection and measurement. To measure sufficient compute, we fix a verified response $y^\star$ and ask which agents can reproduce it at minimum cost. Define

$$g_{F,k}(s) = \arg\max_{v \in \mathcal{V}} \pi_{F,k}(v \mid s), \qquad a_{F,k,t} = \mathbf{1}\left[g_{F,k}(s_t^\star) = y_t^\star\right]. \tag{5}$$

The matrix $A_F = (a_{F,k,t})$ records which agents reproduce which reference tokens. At position t, the admissible agents are $\mathcal{A}_F(t) = \{k : a_{F,k,t} = 1\}$, and their prediction costs are $c_{F,k,t} = c_{F,k}(s_t^\star)$. If

Algorithm 1 Measuring sufficient compute by token-level policy selection

Require: Prompt x; verified reference $y^\star_{1:T}$; panel Π_F with FLOP functions $c_{F,k}(s)$
Ensure: Matches A_F; selected agents $k^\star_{1:T}$; costs $d_F(1{:}T)$; unresolved positions U

Evaluate reference states

$U \leftarrow \varnothing$
for $t = 1, \dots, T$ **do**
  $s^\star_t \leftarrow (x, y^\star_{<t})$
  for $k = 1, \dots, K_F$ **do**
    $a_{F,k,t} \leftarrow \mathbf{1}[g_{F,k}(s^\star_t) = y^\star_t]$
    $c_{F,k,t} \leftarrow c_{F,k}(s^\star_t)$
  end for
  $\mathcal{A}_F(t) \leftarrow \{k : a_{F,k,t} = 1\}$

Select least-cost agents

  if $\mathcal{A}_F(t) \neq \varnothing$ **then**
    $k^\star_t \leftarrow \arg\min_{k \in \mathcal{A}_F(t)} c_{F,k,t}$
    $d_F(t) \leftarrow c_{F,k^\star_t,t}$
  else
    $k^\star_t \leftarrow \perp$; $d_F(t) \leftarrow +\infty$
    $U \leftarrow U \cup \{t\}$
  end if
end for
return A_F, $k^\star_{1:T}$, $d_F(1{:}T)$, U

every reference position is resolved, then $\mathcal{A}_F(t) \neq \varnothing$ for all t. A controller reproduces the reference by choosing an admissible agent at each position. The minimum-cost selection problem is therefore

$$\min_{(k_1,\dots,k_T) \in \prod_{t=1}^{T} \mathcal{A}_F(t)} \sum_{t=1}^{T} c_{F,k_t,t}. \tag{6}$$

The key simplification is that every admissible agent emits the same token $y^\star_t$ and reaches the same next state $s^\star_{t+1}$. Consequently, choosing among these agents changes the immediate cost without changing the remaining reference or the available future choices. Let J_t be the minimum cost of reproducing the remaining reference from $s^\star_t$. Under the additive-cost assumptions above, its Bellman recursion becomes

$$J_t = \min_{k \in \mathcal{A}_F(t)} (c_{F,k,t} + J_{t+1}) = d_F(t) + J_{t+1}, \qquad J_{T+1} = 0. \tag{7}$$

The continuation cost J_{t+1} is shared by all admissible agents. Minimizing the total reference cost therefore reduces to selecting the least costly matching agent at each position:

$$k^\star_t = \arg\min_{k \in \mathcal{A}_F(t)} c_{F,k,t}, \qquad \sigma^\star_{y^\star}(k \mid s^\star_t) = \mathbf{1}[k = k^\star_t]. \tag{8}$$

This selector reproduces the reference from the initial prompt, with minimum cost:

$$C^\star_{F,\mathrm{ref}}(x, y^\star) = J_1 = \sum_{t=1}^{T} d_F(t). \tag{9}$$

Figure 2 illustrates this tokenwise measurement. Because $V(x, y^\star) = 1$, the selected agent sequence supplies a correct response at cost $C^\star_{F,\mathrm{ref}}(x, y^\star)$. It upper-bounds the minimum cost of generating any correct response $C^\star_{F,\mathrm{correct}}(x) \leq C^\star_{F,\mathrm{ref}}(x, y^\star)$. Appendix B.2 specifies the candidate class, states the assumptions under which Eq. (9) is the minimum over all selectors that reproduce $y^\star$, and gives the local witness bound for larger model classes.

Algorithm 1 computes the matching matrix and the resulting selector. If no agent matches a token, the algorithm records that position as unresolved and assigns $d_F(t) = +\infty$. We implement the reference-state evaluations using teacher forcing: a causal forward pass supplies each model with only the preceding reference tokens at each prediction position. We retain both the selected agent identity $k^\star_t$ and its prediction FLOPs $d_F(t)$. The reported sufficient compute measures the selected agent's prediction cost. The resulting measurements characterize compute along verified trajectories.

4 Experiments

Models. Our main measurement panel comprises fifteen models across three families: Qwen2.5 (0.5B, 1.5B, 3B, 7B, 14B, 32B, 72B) (Qwen et al., 2025), OLMo-2 (1B, 7B, 13B, 32B) (OLMo et al., 2025), and DeepSeek-R1-Distill-Qwen (1.5B, 7B, 14B, 32B) (Guo et al., 2025) as reasoning-distilled

candidates. Falcon3 (Team, 2024) (1B, 3B, 7B, 10B) is tested as extended family candidates. The reference generator samples four responses per prompt at temperature 0.7 and top-p 0.95. Qwen2.5-72B-Instruct generates the references used throughout; Llama-3.3-70B-Instruct generates a second reference set for the cross-family analysis. Table A2 lists every model role, including a Qwen3-30B-A3B MoE generator as a sensitivity check (Appendix C.2). Llama-3.1-8B-Instruct (Grattafiori et al., 2024) supplies token surprisal for statistical controls. Appendix A provides additional setup details.

Datasets. The main experiments use three core datasets covering mathematical reasoning and code generation: GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), and HumanEval (Chen et al., 2021). Nine additional benchmarks extend the analysis to competition mathematics, programming, question answering, knowledge, and commonsense reasoning, including LiveCodeBench (Jain et al., 2025), BBH (Suzgun et al., 2023), and MMLU-Pro (Wang et al., 2024). Together, the twelve benchmarks contain 50,019 evaluation prompts. Table A1 lists all datasets.

Hardware and software. Generation and teacher-forced scoring run on a server equipped with eight 80 GB NVIDIA A100-SXM4 GPUs. We use vLLM 0.14.1 (Kwon et al., 2023) with BF16 precision.

Measurement. Our primary criterion is exact reproduction of the reference token. Appendix B defines the teacher-forced scoring and the top-five and $p \geq 0.5$ alternatives, whose evaluation results are in Table A4. We record the selected model and its architecture-derived FLOPs. The count includes attention, projection, and feed-forward operations. To compare families with different tokenizers, we align their predictions to common words and assign each word the maximum selected model index among its tokens; unresolved words receive an ordinal category above the panel. We estimate uncertainty with 2,000 bootstrap resamples of whole items, retaining the dependence among tokens in each response. Where available, we report $\pm$ one bootstrap SD.

4.1 Compute concentrates in a sparse tail

Most tokens match the smallest model and cost concentrates in the tail. Across all three measurement panels, most reference tokens are assigned to the smallest available model (Shown in Figure 3). On the same Qwen-generated references, the 0.5B Qwen model reproduces 91.86–95.24% of tokens across GSM8K, MATH-500, and HumanEval. The 1B OLMo and 1.5B R1-Distill-Qwen models reproduce 89.34–91.74% and 89.80–92.27%, respectively. The predominance of inexpensive predictions persists across all three model families.

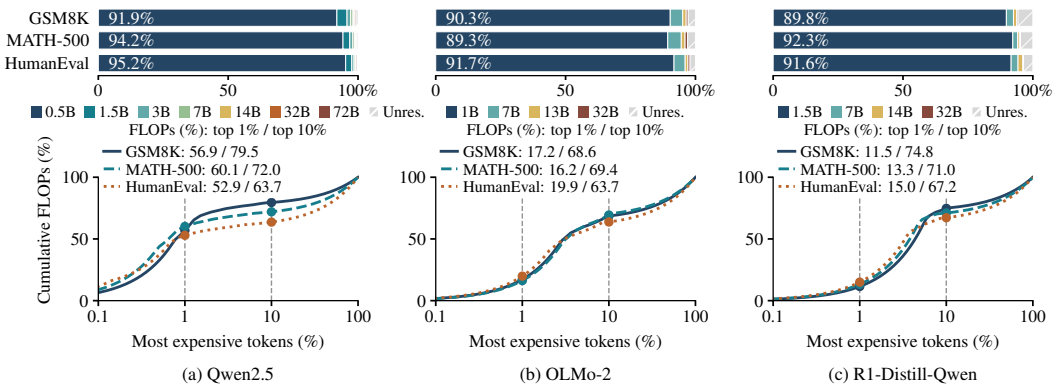
The most expensive 10% of tokens account for 63.66–79.54% of estimated FLOPs in Qwen, 63.75–69.37% in OLMo, and 67.19–74.76% in R1-Distill-Qwen. The top 1% account for 52.94–60.14%, 16.25–19.87%, and 11.47–15.02%, respectively. The estimated compute is therefore concentrated in a small fraction of positions across all three panels, with Qwen exhibiting a sharper concentration within the most expensive percentile. We next examine whether the panels agree on the ordering of expensive positions, with each panel supplying its own cost scale.

Different families share a reproduction ordering. Different model families exhibit a shared ordering of reproduction difficulty. After controlling for generator surprisal, probe surprisal, and position, the partial rank correlation between each pair of families, pooled over the three core datasets, ranges from 0.322 to 0.398 on Qwen-generated references and from 0.417 to 0.475 on Llama-generated references (Table 1). Positions that require larger models in one family tend to require them in another, and the pattern extends to text generated by another family. The association is positive on every benchmark and moves by less than 0.05 under additional controls, alternative success criteria, a second verified response per problem, memorization screening, and GSM-Symbolic (Appendix C, Tables A3 and A4). It is also stronger than greedy or soft speculative-rejection signals on the same text (0.32/0.42 versus 0.15–0.38; Appendix C.2).

4.2 Cheap reproduction does not ensure reliable continuation

The cost map measures token reproduction conditioned on the reference prefix. To assess whether low reproduction costs translate into reliable continuation, we allow smaller models to continue generating the remaining response and report accuracy drops relative to Qwen-72B.

Effect of continuation duration. We vary the duration of small-model generation before Qwen-72B resumes. Both models condition on the generated continuation without receiving additional



(a) Qwen2.5 (b) OLMo-2 (c) R1-Distill-Qwen

Figure 3: Token reproduction and estimated FLOP concentration across Qwen2.5, OLMo-2, and DeepSeek-R1-Distill-Qwen. *Top*: distributions of the least-cost matching model for GSM8K, MATH-500, and HumanEval. Labels on the bar give the smallest-model share. Hatched gray segments denote unresolved tokens (Unres.), which might require larger models to reproduce. *Bottom*: cumulative estimated FLOPs over all reference tokens, ordered from most to least expensive, with a log horizontal axis. Resolved tokens retain their measured architecture-derived FLOP costs; each unresolved token is assigned twice the prediction FLOPs of the largest model in its panel. Vertical dashed lines and markers identify the top 1% and 10% of all tokens.

Table 1: Model families share a reproduction ordering. Partial Spearman correlations of aligned word-level reproduction indices, pooled across GSM8K, MATH-500, and HumanEval, controlling for generator surprisal, probe surprisal, and position. Columns distinguish reference generators. Values are estimates $\pm$ one item-bootstrap SD. A 0 indicates no rank association.

Model families	Qwen references	Llama references
Qwen2.5 / OLMo-2	0.322 ± 0.003	0.417 ± 0.003
Qwen2.5 / Falcon3	0.380 ± 0.003	0.475 ± 0.003
OLMo-2 / Falcon3	0.398 ± 0.003	0.447 ± 0.003

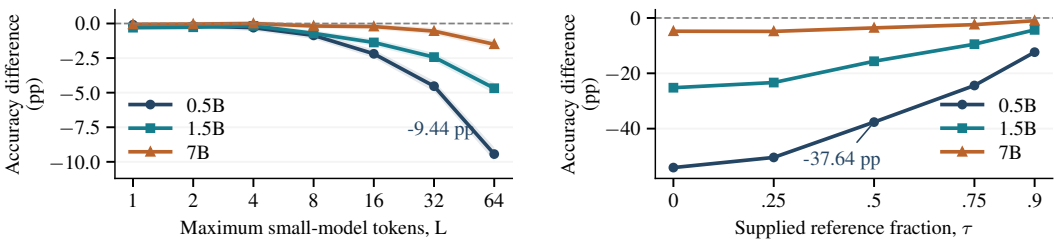
reference tokens. The baseline is an uninterrupted Qwen-72B continuation from the same prefix. Longer handovers produce larger accuracy losses, with the magnitude depending on model capacity (Figure 4, left). For the 0.5B model, the accuracy difference relative to Qwen-72B is -0.09 ± 0.19 percentage points at one token and decreases to -9.44 ± 0.50 at 64 tokens. Larger models mitigate this degradation: at 64 tokens, the losses are 4.68 ± 0.39 points for 1.5B and 1.50 ± 0.30 points for 7B. These results motivate a joint choice of model capacity and delegation length: realizing the savings suggested by the sufficient-compute map requires accounting for how delegated predictions affect the rest of the trajectory. Appendix D specifies the detail.

Effect of model capacity and reference-prefix length. We evaluate full-suffix continuation from reference fractions $\tau \in \{0, 0.25, 0.5, 0.75, 0.9\}$, allowing the selected model to generate the entire remaining response. Relative to Qwen-72B, the 0.5B model loses 54.08 accuracy points when generating from the prompt ($\tau = 0$) and 37.64 points when starting from the reference midpoint.

4.3 Sufficient-compute maps reveal headroom for efficient inference

We evaluate whether the sufficient-compute map can guide model routing and speculative decoding. Figure 5 compares measured answer accuracy with projected generation latency on MATH-500. The map is supplied as an oracle: a greedy 72B response provides the reference for each problem, the controller receives the positions at which 0.5B reproduces it. Appendix E.1 specifies the design.

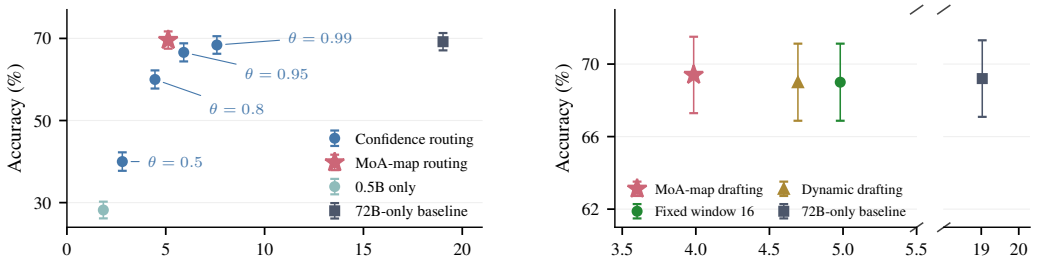
Model routing. Our baseline, token-level cascade (Narasimhan et al., 2025), uses the 0.5B model when its maximum next-token probability reaches a threshold and defers to 72B otherwise. MoA-map routing replaces this confidence gate with measured agreement: it selects 0.5B at positions where the map indicates that it reproduces the token, and 72B elsewhere. MoA-map routing achieves 69.60%



(a) Duration from the reference midpoint

(b) Small model completes the suffix

Figure 4: Accuracy drops with the smaller model continuation. Differences from a 72B continuation of the same verified prefix. *(a)*: a small model generates up to L tokens, then 72B resumes. Shading shows $\pm$ one item-bootstrap SD. *(b)*: full-suffix generation from reference character fraction τ.



(a) Model routing

(b) Speculative decoding

Figure 5: Accuracy versus latency on MATH-500. (a) Confidence routing selects 0.5B when its maximum next-token probability is at least θ. MoA-map routing uses the sufficient-compute map. (b) Draft lengths follow a fixed window, model confidence, or consecutive sufficient-compute matches. Gray squares show the 72B-only baseline. Vertical bars show one item-bootstrap standard deviation.

accuracy, compared with 68.40% for the best-performing confidence setting ($\theta = 0.99$), yielding the best accuracy–latency tradeoff (shown in Figure 5a). The comparison identifies an opportunity beyond confidence-based routing: token agreement can guide delegation to a smaller model while retaining high answer accuracy.

Speculative decoding. The map can also determine how many tokens the small model should draft before verification in speculative decoding (Leviathan et al., 2023). MoA-map drafting follows consecutive positions marked as reproducible by 0.5B and sends the resulting block to 72B for verification. It achieves 69.40% accuracy, compared with 69.00% for both a fixed 16-token window and confidence-adaptive drafting (Mamou et al., 2024). Meanwhile, the projected latencies are 3.99, 4.98, and 4.69 seconds per response, reducing latency by up to 19.9% (shown in Figure 5b). MoA-map drafting generates 32.6% fewer draft tokens than the fixed-window baseline: anticipating disagreement reduces proposals that the target would discard. This reduction in draft work explains its lower latency. Together, the two interventions use sufficient-compute map to identify where small-model predictions suffice and how long drafting should continue before verification.

5 Related Work

Efficient serving. Efficient LLM serving combines systems optimizations with adaptive computation. Orca combines iteration-level scheduling with selective batching (Yu et al., 2022), while vLLM reduces KV-cache fragmentation and enables memory sharing through PagedAttention (Kwon et al., 2023). Speculative decoding preserves a target distribution by verifying draft proposals (Leviathan et al., 2023; Chen et al., 2023a). Its acceptance rate depends on alignment between distributions; a rejected proposal can still be a valid continuation (Bachmann et al., 2025). FrugalGPT and RouteLLM allocate requests across models to balance cost and quality (Chen et al., 2023b; Ong et al., 2025),

while token-level cascades make small/large-model decisions within a response (Narasimhan et al., 2025). At the architectural level, MoE increases parameter capacity through sparse expert activation (Shazeer et al., 2017; Fedus et al., 2022). Expert placement and load balancing help translate this sparsity into serving efficiency (Du et al., 2024; Ma et al., 2025). Our work complements these approaches by measuring which complete model can reproduce each token given its reference prefix, then testing whether this information improves model selection and speculative draft length.

Token-level control. CITER and R2R learn small/large-model routing (Zheng et al., 2025; Fu et al., 2025), while COLLAB selects agents using long-term utility (Chakraborty et al., 2025). Our instrument measures local reference cost before testing its usefulness for these sequential decisions. Duo-LLM studies oracle module allocation, FS-GEN studies small/large-model collaboration, and FlexiDepth learns layer skipping (Alizadeh-Vahid et al., 2024; Zhang et al., 2024; Luo et al., 2025). CALM and Mixture-of-Depths also allocate computation within a model (Schuster et al., 2022; Raposo et al., 2024). Appendix C.3 quantifies agreement between our reproduction indices and FlexiDepth's learned allocations; Appendix F relates our measurement design to these methods.

Shared difficulty and trajectory interventions. Earlier work finds shared learning order across classifiers (Hacohen et al., 2020) and studies token learning trajectories across model scales (Xia et al., 2023; Chang et al., 2024). Our cross-family test asks whether a related ordering persists within verified generated text. Its interpretation depends on the operational criterion (Jacobs & Wallach, 2021). Critical-token studies locate influential positions through contrasting continuations (Lin et al., 2025). Scheduled sampling and work on exposure bias distinguish supplied-prefix prediction from free generation (Bengio et al., 2015; He et al., 2021; Bachmann & Nagarajan, 2024). Model handovers also appear in Reasoning Relay, LLM Shepherding, and RelayGen (Lu et al., 2025; Dong et al., 2026; Song et al., 2026). Our duration sweep relates these continuation interventions to a separately measured reproduction cost.

6 Discussion and limitations

Discussion. Our study uses sufficient compute as an analytical reference for identifying efficiency headroom in existing serving methods. We show that better allocation information can improve the accuracy–efficiency trade-off. However, constructing the map requires evaluating every candidate model along known reference sequences. Developing practical controllers that recover these benefits without exhaustive evaluation or advance reference access remains an open problem. The shared ordering of reproduction demands across model families raises a further question: are measured maps transferable? Cross-family agreement motivates this possibility but does not establish transferability.

In our experiments, longer words show a weak positive association with reproduction demand, whereas digit-containing and operator/punctuation spans show weak negative associations. Regressions on digit and operator indicators, word length, frequency, and position explain only 2.5–3.2% of the variance in ranked reproduction demand (Appendix C.2). Thus, simple surface categories provide limited guidance about which tokens require larger models and identifying the contextual and semantic properties of the expensive tail could be an interesting direction.

Limitations. Exact reproduction excludes alternative valid wordings and reasoning paths, particularly in open-ended natural language. Meanwhile, exact agreement may be more informative for structured outputs whose tokens have fixed meanings, such as tokenized visual predictions. The results are conditional on the evaluated model panels and benchmarks. Most reproduction analyses use checker-passing references, limiting coverage of unsuccessful trajectories and difficult problems.

7 Conclusion

We introduced a token-level MoA framework for measuring sufficient compute. Most reference tokens can be reproduced by the smallest model, while a sparse expensive tail accounts for disproportionate assigned computation. Shared reproduction orderings extend across the tested families, but handover experiments show that local reproduction alone does not ensure reliable continuation. On MATH-500, oracle-guided routing and drafting improve projected efficiency at comparable accuracy, revealing remaining allocation headroom. Exploiting this opportunity without advance reference access or exhaustive model evaluation remains a central challenge for practical inference.

AI USE STATEMENT

We used LLM-based assistants in a limited, supporting role. For required-disclosure tasks, AI tools only provided feedback on our research methodology. All research ideas, methods, method implementations, data analysis, and interpretation of results were done by the authors. The remaining required-disclosure tasks do not apply, since no AI tools were used for them. For recommended-disclosure tasks, AI tools cleaned author-written code without changing its logic. They also launched experiment runs using author-specified configurations, wrote plotting scripts for figures, polished the writing (no main text was drafted by AI), assisted with brainstorming (the final idea is entirely the authors'), searched for technical information such as Hugging Face blogs and model cards, and suggested keywords (none were adopted). All references were found and added by the authors. We have reviewed all AI-assisted work and take full responsibility for the content of this paper.

ETHICS STATEMENT

This work studies the efficiency of large language models that use mixture-of-agents techniques. It does not involve human subjects, user studies, or crowd sourcing, and it does not collect or process personal or sensitive data. All models and datasets used are publicly available and were used in accordance with their licenses and terms of use. We do not release new datasets or models that raise additional safety concerns. Since our study focuses on efficiency, we do not expect it to create new risks of misuse beyond those of the existing models we analyze. Improving efficiency may also help lower the computational and energy costs of deploying large language models. The authors declare no conflicts of interest or sponsorship that could have influenced this work. All authors have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Appendices A and B specify the data selection, model identifiers, prediction criteria, alignment, FLOP convention, and statistical estimands; Appendix C reports per-dataset results and sensitivity analyses; Appendices D and E give the intervention and allocation protocols. Source code is uploaded as supplementary material.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Keivan Alizadeh-Vahid, Seyed Iman Mirzadeh, Hooman Shahrkokhi, Dmitry Belenko, Frank Sun, Minsik Cho, Mohammad Hossein Sekhavat, Moin Nabi, and Mehrdad Farajtabar. Duo-llm: A framework for studying adaptive computation in large language models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*, pp. 443–455. PMLR, 2024.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=76zq8Wkl6Z.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *International Conference on Learning Representations*, volume 2025, pp. 40894–40914, 2025.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJDaqqveg.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and

R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/e995f98d56967d946471af29d7bf99f1-Paper.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Souradip Chakraborty, Soumya Suvra Ghosal, Ming Yin, Dinesh Manocha, Mengdi Wang, Amrit Singh Bedi, and Furong Huang. Transfer q-star: Principled decoding for llm alignment. *Advances in Neural Information Processing Systems*, 37:101725–101761, 2024.

Souradip Chakraborty, Sujay Bhatt, Udari Madhushani Sehwag, Soumya Suvra Ghosal, Jiahao Qiu, Mengdi Wang, Dinesh Manocha, Furong Huang, Alec Koppel, and Sumitra Ganesh. Collab: Controlled decoding using mixture of agents for LLM alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7ohlQUbTpp.

Tyler A Chang, Zhuowen Tu, and Benjamin K Bergen. Characterizing learning curves during language model pre-training: Learning, forgetting, and stability. *Transactions of the Association for Computational Linguistics*, 12:1346–1362, 2024.

Aaron Chatterji, Thomas Cunningham, David J Deming, Zoe Hitzig, Christopher Ong, Carl Yan Shan, and Kevin Wadman. How people use chatgpt. Technical report, National Bureau of Economic Research, 2025.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance, 2023b.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Ziming Dong, Hardik Sharma, Evan O'Toole, Jaya Prakash Champati, and Kui Wu. Pay for hints, not answers: Llm shepherding for cost-efficient inference. *arXiv preprint arXiv:2601.22132*, 2026.

Zhixu Du, Shiyu Li, Yuhao Wu, Xiangyu Jiang, Jingwei Sun, Qilin Zheng, Yongkai Wu, Ang Li, Hai H Li, and Yiran Chen. Sida: Sparsity-inspired data-aware serving for efficient and scalable large mixture-of-experts models. *Proceedings of Machine Learning and Systems*, 6:224–238, 2024.

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

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=DpeJYRFRQY.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Guy Hacohen, Leshem Choshen, and Daphna Weinshall. Let's agree to agree: Neural networks share classification order on real datasets. In *International Conference on Machine Learning*, pp. 3950–3960. PMLR, 2020.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James Glass. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5087–5102, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.415. URL https://aclanthology.org/2021.emnlp-main.415/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Abigail Z Jacobs and Hanna Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 375–385, 2021.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Hynek Kydlíček. math-verify: Mathematical expression verification, 2025. URL https://github.com/huggingface/Math-Verify.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances LLM's reasoning capability. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=fnz1g18EdI.

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. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Leo Lu, Jonathan Zhang, Sean Eugene Chua, Spencer Kim, Kevin Zhu, Sean O'Brien, and Vasu Sharma. Evaluating stability and interchangeability of large language models in mathematical reasoning. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=QqXPAuKi76.

Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained llms. *arXiv preprint arXiv:2503.23798*, 2025.

Haiyue Ma, Zhixu Du, and Yiran Chen. Moe-GPS: Guidlines for prediction strategy with expert duplication in moe load balancing. In *Machine Learning for Systems 2025*, 2025. URL https://openreview.net/forum?id=VCh5vBSxxb.

Jonathan Mamou, Oren Pereg, Joao Gante, Lewis Tunstall, Daniel Korat, Nadav Timor, and Moshe Wasserblat. Faster assisted generation with dynamic speculation. Hugging Face Blog, October 2024. URL https://huggingface.co/blog/dynamic_speculation_lookahead.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 94743–94765, 2025.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=bVIcZb7Qa0.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 44949–44987, 2025.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

Isaac Ong, Amjad Almahairi, Vincent Wu, et al. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations (ICLR)*, 2025.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwen2.5 speed benchmark. https://qwen.readthedocs.io/en/v2.5/benchmark/speed_benchmark.html, n.d. Official BF16 vLLM 0.6.3 benchmarks on A100 80GB GPUs; accessed 2026-09-25.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

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. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12974–12990, 2024.

Jiwon Song, Yoongon Kim, and Jae-Joon Kim. Relaygen: Intra-generation model switching for efficient reasoning. *arXiv preprint arXiv:2602.06454*, 2026.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Falcon-LLM Team. The falcon 3 family of open models, December 2024. URL https://huggingface.co/blog/falcon3.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h0ZfDIrj7T.

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. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Veselin Stoyanov. Training trajectories of language models across scales. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13711–13738, 2023.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX symposium on operating systems design and implementation (OSDI 22)*, pp. 521–538, 2022.

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*, pp. 4791–4800, 2019.

Kaiyan Zhang, Jianyu Wang, Ning Ding, Biqing Qi, Ermo Hua, Xingtai Lv, and Bowen Zhou. Fast and slow generating: An empirical study on large and small language models collaborative decoding. *arXiv preprint arXiv:2406.12295*, 2024.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P Xing, Hongyi Wang, and Huaxiu Yao. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*, 2025.

A DATASETS, MODELS, AND SELECTED POPULATIONS

Benchmark coverage. Table A1 separates attempted prompts, each generator's verified pool, and their intersection. The primary collection contains twelve datasets and 50,019 prompts. GSM-Symbolic adds 5,000 prompts as a separate robustness analysis. Each core dataset has two-generator intersection coverage of at least 60%. The expanded collection is reported per dataset and is not added to the pooled core estimand.

GSM8K uses the test split, MATH-500 the released 500-problem evaluation set, and HumanEval the 164 official function-completion problems. LiveCodeBench uses 219 problems with usable stdin/stdout tests from the 400-problem mirror available to the pipeline. Both public and private stored tests are included. This is a subset, without a post-training-cutoff guarantee (Jain et al., 2025). NQ-Open uses 3,610 validation questions with normalized exact-match answer aliases (Kwiatkowski et al., 2019).

The extensions use `evalplus/mbppplus` (378 test items; Liu et al., 2023), `HuggingFaceH4/aime_2024` (30 problems), `allenai/ai2_arc` (ARC-Challenge test; Clark et al., 2018), `lukaemon/bbh` (the loaded task test splits; Suzgun et al., 2023), `TIGER-Lab/MMLU-Pro` (test; Wang et al., 2024), `cais/mmlu` (all subjects, test; Hendrycks et al., 2020), and `Rowan/hellaswag` (validation; Zellers et al., 2019). BBH contains varied answer formats, not exclusively multiple-choice questions. MBPP+ prompts include a supplied assertion specifying the required function call. GSM-Symbolic uses the released main split (Mirzadeh et al., 2025); its transformed problems provide a replication substrate, but their public release does not establish absence from model training.

Prompts and checkers. Mathematical prompts request step-by-step reasoning and a final answer; code prompts request executable code; knowledge and reasoning prompts request an answer in the format specified by the dataset adapter. GSM8K extracts a final numeric answer and compares it with numeric tolerance 10^{-6}. MATH-500 uses symbolic answer equivalence through Math-Verify (Kydlíček, 2025). HumanEval executes extracted code with the stored official tests and a timeout. LiveCodeBench runs the stored stdin/stdout cases; MBPP+ runs the executable tests available in the loaded records. The multiple-choice adapters parse the final option, BBH uses task-answer matching, and NQ-Open normalizes case, articles, punctuation, and whitespace before exact alias matching. These are outcome checkers: intermediate reasoning is retained even when a valid final answer follows an incorrect step.

Generation settings. Reference sampling uses $K = 4$, temperature 0.7, top-p 0.95, and seed 20260729. Output caps are 512 tokens for GSM8K, HumanEval, MBPP+, ARC-Challenge, and MMLU; 1,024 for MATH-500 and LiveCodeBench; 64 for NQ-Open; 2,048 for AIME; 768 for BBH and MMLU-Pro; and 384 for HellaSwag. GSM-Symbolic uses 512. The earliest passing sample is selected independently for each generator. A selected reference can be non-modal under its own generator because it was sampled. Generation and scoring use bfloat16 with vLLM (Kwon et al., 2023); the project runtime records version 0.14.1.

A.1 MODEL PANEL AND AUXILIARY MODELS

Table A2 summarizes model roles and sizes. Qwen2.5, OLMo-2, and R1-Distill-Qwen supply the fifteen candidate models displayed in Figure 3 (Qwen et al., 2025; OLMo et al., 2025; Guo et al., 2025). The cross-family agreement analysis in Table 1 uses Qwen2.5, OLMo-2, and the additional Falcon3 panel (Team, 2024). These are distinct panel comparisons.

Qwen model identifiers have the form `Qwen/Qwen2.5-{size}B-Instruct`. The OLMo identifiers are `allenai/OLMo-2-0425-1B-Instruct`, `allenai/OLMo-2-1124-7B-Instruct`, `allenai/OLMo-2-1124-13B-Instruct`, and `allenai/OLMo-2-0325-32B-Instruct`. Falcon identifiers are `tiiuae/Falcon3-{size}B-Instruct`. The second main generator is `meta-llama/Llama-3.3-70B-Instruct`; the probability probe is `meta-llama/Llama-3.1-8B-Instruct`.

The reasoning-distilled panel contains four DeepSeek-R1-Distill-Qwen models, with 1.5, 7, 14, and 32B parameters. Their identifiers follow

Table A1: Benchmark coverage before and after verification. The first three rows define the core. Counts refer to items.

Dataset	Attempted	Qwen verified	Llama verified	Intersection
GSM8K	1,319	1,281	1,288	1,275
MATH-500	500	375	363	347
HumanEval	164	144	146	136
LiveCodeBench subset	219	108	105	84
NQ-Open	3,610	1,090	1,355	862
MBPP+	378	310	294	282
AIME 2024	30	6	9	5
ARC-Challenge	1,172	1,137	1,136	1,120
BBH	6,511	5,509	5,647	5,355
MMLU-Pro	12,032	8,700	8,803	7,853
MMLU	14,042	12,303	12,260	11,611
HellaSwag	10,042	9,019	8,883	8,306

Table A2: Model roles and sizes. Measurement panels supply candidate next-token policies; the Falcon panel tests cross-family agreement. Reference generators supply candidate responses for task verification; the additional MoE generator tests sensitivity to the reference source. The probability probe supplies surprisal controls. Sizes are nominal billions of parameters; the final row distinguishes total and active parameters. Qwen-72B serves both as a panel member and a generator.

Role	Model family	Sizes (B)
Measurement panel	Qwen2.5-Instruct	0.5, 1.5, 3, 7, 14, 32, 72
Measurement panel	OLMo-2-Instruct	1, 7, 13, 32
Reasoning-distilled panel	R1-Distill-Qwen	1.5, 7, 14, 32
Cross-family replication	Falcon3-Instruct	1, 3, 7, 10
Probability probe	Llama-3.1-Instruct	8
Reference generator	Qwen2.5-Instruct	72
Reference generator	Llama-3.3-Instruct	70
Additional generator	Qwen3-30B-A3B	30 total, 3 active

`deepseek-ai/DeepSeek-R1-Distill-Qwen-<size>B`. These models test sensitivity to reasoning distillation. A third generator, `Qwen/Qwen3-30B-A3B`, supplies mixture-of-experts reference text with thinking mode disabled. Its nominal 30B total and 3B active parameter counts describe architecture, not a measured change in per-token compute allocation. These extensions use the three core datasets.

B Measurement and statistical details

Teacher-forced scoring. For each model, the user prompt is rendered using its chat template and assistant-generation header. The reference answer is tokenized separately without special tokens and appended to the prompt tokens. vLLM returns prompt log probabilities for the answer positions. The cache stores target rank, target log probability, argmax identity, character offsets, and a truncated distribution. Log probabilities are rounded to four decimal places. Each model receives the reference prefix even when its own preceding prediction disagrees with the reference. Every model is evaluated at every position; the algorithm does not stop after a smaller model matches. The primary pass indicator is target rank one. Alternative criteria use rank at most five or target log probability at least $\log(0.5)$, compared at the stored float32 precision.

The entropy control sums $-p \log p$ over the returned top-twenty entries and the target entry when additionally returned. It is an unnormalized truncated entropy proxy.

Word alignment. The alignment routine identifies non-whitespace runs and removes leading and trailing ASCII punctuation, retaining all-punctuation spans. A native token is assigned to the word with maximal character overlap; ties go to the first word, and zero-overlap tokens are dropped. Only

common mapped words are joined across families. At word w, the ordinal statistic is

$$D_F(w) = \max_{t:\, t \mapsto w} R_F(t), \qquad R_F(t) = \min\{k : a_{F,k,t} = 1\}. \tag{10}$$

An unresolved token gives the word the unresolved index above the panel. The maximum summarizes constituent token indices; because predictions need not be monotone in size, it does not certify that a single model reproduces all constituent tokens. Surprisal uses sums over assigned tokens.

Partial rank correlation. Let $r(\cdot)$ assign average ranks to ties and let

$$Z = [\mathbf{1}, r(S_{\text{gen}}), r(S_{\text{probe}}), r(\log(1 + \text{position}))].$$

Writing P_Z for least-squares projection onto the columns of Z, we compute

$$\rho_{\text{partial}} = \text{corr}\left((I - P_Z)r(D_Q), (I - P_Z)r(D_O)\right). \tag{11}$$

Tables report the estimate $\pm$ the SD of 2,000 item-bootstrap estimates. This bootstrap SD estimates the standard error under item sampling; models and generation settings are held fixed. Main-text estimates and figure bars or bands use the same bootstrap-SD convention where these SDs are available. Point-only results omit uncertainty when no SD is available. The bootstrap samples item identifiers with replacement and retains all word rows belonging to each sampled item, including multiplicities. Ranks and regressions are recomputed for each resample. Core Qwen text contributes 319,336 words from 1,800 items; Llama text contributes 302,959 words from 1,797 items. The intersection sensitivity retains 307,850 and 293,549 words, respectively, from 1,758 items, giving point estimates of 0.319 and 0.415.

B.1 Architecture-derived FLOP accounting

We count matrix-multiplication FLOPs from the released model configurations, with one multiply-add counted as two operations. The count includes attention projections, the two attention matrix products, gated feed-forward layers, MoE router projections, and the vocabulary projection. Embedding lookup, normalization, activations, bias additions, rotary embeddings, softmax, and expert dispatch/combine are outside this arithmetic convention. Counts describe logical causal operations; they do not include kernel padding, communication, or memory movement.

Let h be hidden width, q the total query-head width, r the total key/value-head width, L the number of layers, and V vocabulary size. For layer ℓ, let I_ℓ^{act} be the sum of the intermediate widths of the feed-forward networks activated for one token. A dense gated MLP has $I_\ell^{\text{act}} = I_\ell$; a sparse layer with a_ℓ selected equal-sized experts has $I_\ell^{\text{act}} = a_\ell I_\ell$. Let E_ℓ denote the number of experts scored by a router, and set $E_\ell = 0$ in a dense layer. Define

$$A = \sum_{\ell=1}^{L} \left[4h(q + r) + 6hI_\ell^{\text{act}} + 2hE_\ell\right], \qquad B = 4Lq, \qquad H = 2hV. \tag{12}$$

The first term counts query, key, value, and output projections; the second counts the gate, up, and down feed-forward matrices. The router multiplies an h-dimensional token by an $h \times E_\ell$ matrix. The two attention products together require B FLOPs per query/key position pair across layers. For a prefix of p tokens with the preceding keys and values cached,

$$c_{F,k}(s) = A_{F,k} + B_{F,k}p + H_{F,k}. \tag{13}$$

The saved prompt length and native-token position give p separately for each model. The model ordering is unchanged throughout the measured panels; FLOP concentration sorts the resulting per-token values, including their prefix dependence. Qwen3-30B-A3B uses eight active experts out of 128 per layer, so only those eight feed-forward networks enter I_ℓ^{act}.

For a request with $m \geq 1$ input tokens and $n \geq 1$ generated tokens, causal prefill processes $m(m + 1)/2$ attention pairs. Only its final position requires vocabulary logits for generation. The remaining $n - 1$ predictions use cached decoding at prefix lengths $m + 1, \ldots, m + n - 1$. Thus

$$\mathcal{F}_k(m, n) = mA_k + \frac{B_k m(m + 1)}{2} + H_k + \sum_{j=1}^{n-1} \left[A_k + B_k(m + j) + H_k\right]. \tag{14}$$

This function increases with requested output length. Reserving its value at a request's output cap therefore bounds the counted FLOPs of every completion admitted under that cap. Model configurations, their source revisions, and file hashes accompany the FLOP-count implementation.

B.2 What the bound means

Local witness bound. Fix a prefix and a target token. For a candidate model class $\mathcal{C}$ containing the tested family, let $d_{\mathcal{C}}(t)$ be the infimum FLOP count at that prefix of a member whose deterministic prediction equals that token. Whenever a tested member succeeds, $d_{\mathcal{C}}(t) \le d_F(t)$. The comparison fixes the model class, prediction rule, and supplied prefix.

Exact-reference selector. Assume all panel models share a token alphabet, decode greedily with a fixed deterministic tie-breaking rule, and have additive next-token FLOP costs $c_k(s) > 0$, available prefix caches, and no switching or state-construction overhead. Suppose every token of a finite reference y is resolved. A selector may choose any model at each prefix, with advance knowledge of the full reference and all panel predictions. Write $\mathrm{rollout}(\sigma, x)$ for the response generated by a deterministic selector σ with prompt x, and $C(\sigma)$ for its additive prediction cost. Among selectors that reproduce y exactly,

$$\min_{\sigma:\,\mathrm{rollout}(\sigma,x)=y} C(\sigma) = \sum_{t=1}^{T} d_F(t). \tag{15}$$

To attain the right-hand side, choose a cheapest matching model at position one. Inductively, the generated prefix then equals the reference prefix, so the cheapest matching model at the next position emits the required token. Conversely, every selector reproducing y must choose a matching model at each position and therefore incurs at least $d_F(t)$ there. Summing proves Eq. 15. If stopping is to be reproduced, it must also be included in the reference alphabet and coverage assumption.

Equation 15 characterizes selectors constrained to one reference. For a checker-passing y, the task-correct objective $\min_{\sigma:V(x,\mathrm{rollout}(\sigma,x))=1} C(\sigma)$ allows all such selectors and those producing other correct responses. Its optimum is therefore at most $\sum_t d_F(t)$ under the same assumptions. This establishes the direction of the bound when alternative solutions are allowed. The empirical corpus includes unresolved tokens and separately tokenized families; the cost proxy omits map acquisition and model-state transfer. This exact-reference result interprets the measured quantity under its stated assumptions, while the allocation studies evaluate concrete policies.

C Additional reproduction results

The results below supplement the concentration and shared-ordering findings. Figure 6 restores the per-dataset view summarized by Table 1; Table A3 gives its numerical estimates. Tables A4 provide additional controls.

C.1 Concentration with unresolved-token costs

Figure 3 uses the same 1,800 Qwen-generated verified references across the Qwen, OLMo, and R1-Distill-Qwen panels: 1,281 GSM8K, 375 MATH-500, and 144 HumanEval responses. Each panel uses its own native tokenization. The distribution bars divide by all reference tokens, including unresolved positions.

An unresolved position has no observed successful agent and therefore no finite measured sufficient compute under Definition 1. For the displayed cumulative-cost curves, we assign it twice the architecture-derived prediction FLOPs of the panel's largest model at that prefix. Writing $k_{\max}$ for this model, the plotted cost is

$$\widetilde{d}_F(t) = \begin{cases} d_F(t), & \mathcal{A}_F(t) \neq \varnothing, \\ 2c_{F,k_{\max}}(s_t^\star), & \mathcal{A}_F(t) = \varnothing. \end{cases} \tag{16}$$

The multiplier is an assumed cost for the visualization. It does not establish what an untested model would cost or that an unresolved token necessarily requires a larger model. The matching outcomes themselves are unchanged by this convention.

Figure 3 assigns each unresolved position twice the cost of the largest panel model. This section reports the same shares among resolved tokens only (top 10%: 51–60% for Qwen2.5, 40–44% for OLMo-2, 26–32% for R1-Distill-Qwen) and Gini coefficients of resolved costs (Tables A5 and A6); allocation is uneven under both conventions.

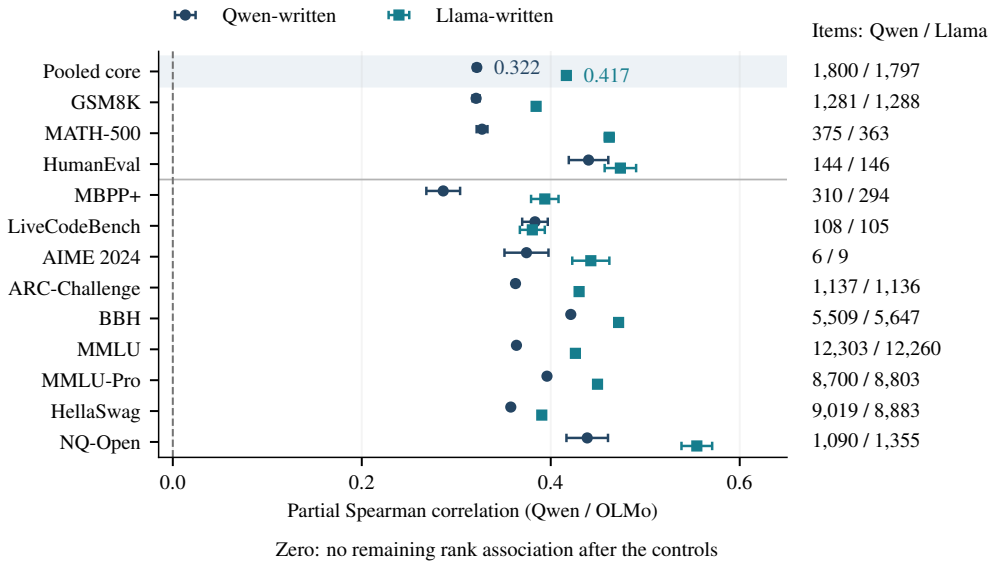


Figure 6: Cross-family agreement across benchmark subsets. Partial Spearman correlation of Qwen/OLMo word indices after controlling generator surprisal, probe surprisal, and position; bars show ± one item-bootstrap SD. Zero denotes no remaining rank association, and larger positive values indicate stronger shared ordering. Pooling uses the three core datasets. The right column gives each generator's verified item count (Qwen/Llama), including six/nine AIME items; Table A1 specifies the evaluated populations.

Table A3: Per-dataset Qwen/OLMo partial correlations with bootstrap SDs (standard errors).

Dataset	Qwen text	Llama text
GSM8K	0.321 ± 0.004	0.385 ± 0.003
MATH-500	0.327 ± 0.006	0.462 ± 0.005
HumanEval	0.440 ± 0.021	0.474 ± 0.017
LiveCodeBench subset	0.383 ± 0.014	0.381 ± 0.013
NQ-Open	0.439 ± 0.022	0.555 ± 0.016
MBPP+	0.286 ± 0.018	0.394 ± 0.015
AIME 2024	0.374 ± 0.023	0.442 ± 0.020
ARC-Challenge	0.363 ± 0.003	0.430 ± 0.002
BBH	0.421 ± 0.002	0.472 ± 0.001
MMLU-Pro	0.396 ± 0.001	0.450 ± 0.001
MMLU	0.364 ± 0.001	0.426 ± 0.001
HellaSwag	0.358 ± 0.001	0.391 ± 0.001

Within each panel and dataset, tokens are ordered by decreasing plotted cost. The top-p share divides their cumulative cost by the total cost of all reference tokens. Fractional percentile boundaries are evaluated by linear interpolation of cumulative token counts and costs. Table A6 gives the values displayed at 1% and 10%, alongside the corresponding resolved-only calculation. In the resolved-only calculation, both the percentile and total cost use resolved tokens. The two columns therefore summarize different populations.

The resolved-only Gini coefficients in Table A5 use the measured architecture-derived costs without imputation. They also include Llama-generated references, which provide a comparison of reference sources. The pooled resolved-only Qwen cost share of 3.03% for half the assigned FLOPs is another such summary; it should not be substituted for the all-token percentile shares in Figure 3.

Table A4: Sensitivity of cross-family agreement, with bootstrap SDs (standard errors). The frequency/entropy and loss-gap rows extend the primary controls successively. Other rows specify separate controls or populations.

Variant	Qwen text	Llama text
Primary controls	0.322 ± 0.003	0.417 ± 0.003
+ word length and token counts	0.321 ± 0.003	0.416 ± 0.003
+ frequency and approximate entropy	0.299 ± 0.003	0.406 ± 0.003
+ both families' loss gaps	0.276 ± 0.003	0.409 ± 0.003
+ smallest-model surprisals	0.322 ± 0.003	0.409 ± 0.003
High-confidence reference words	0.287 ± 0.004	0.400 ± 0.003
Top-five criterion	0.302 ± 0.008	0.443 ± 0.006
$p \geq 0.5$ criterion	0.355 ± 0.003	0.451 ± 0.003
Second verified response	0.328 ± 0.003	0.409 ± 0.003
Memorization-screened subset	0.321 ± 0.003	0.416 ± 0.003
GSM-Symbolic	0.319 ± 0.002	0.384 ± 0.002

Table A5: Gini coefficients of architecture-derived prediction FLOPs among resolved tokens; parentheses give unresolved-token percentages among all tokens. These values exclude the imputed costs used in Figure 3.

Text	Panel	GSM8K	MATH-500	HumanEval
Qwen	Qwen	0.547 (0.76%)	0.451 (0.49%)	0.514 (0.14%)
Qwen	Olmo	0.330 (2.80%)	0.365 (2.99%)	0.316 (2.08%)
Llama	Qwen	0.543 (4.15%)	0.547 (4.76%)	0.243 (0.73%)
Llama	Olmo	0.385 (5.29%)	0.467 (8.72%)	0.218 (1.42%)

C.2 Controls and reference variation

Control variants. Table A4 reports each variant with its own bootstrap SD. Frequency is estimated from the available text, and entropy is approximate as described above. The loss-gap controls use each family's smallest and largest model on the same reference. The high-confidence subset requires every assigned generator token to have stored probability at least 0.5; it retains 306,490 Qwen-text words and 290,328 Llama-text words. This is stricter than merely asking whether a token is the generator's argmax. Surface-feature regressions on ranks, using digit/operator indicators, word length, frequency, and position, have $R^2 = 0.025$ and 0.032. These regressions quantify that feature set's linear explanatory power.

Reference variation and memorization screening. The second-target analysis uses the next available checker-passing sample among the original four candidates. Its pooled correlations are 0.328 and 0.409; mean item demand correlates 0.574 and 0.653 across two targets from the same generator. A separate screening campaign tests answer recall without demonstrated solution ability and continuation of problem text. Screening covers six model sizes: Qwen 0.5, 1.5, 7, and 72B, and OLMo 7 and 32B. The rule flags 33 of the 1,758 intersection items. Removing flagged identifiers from the analysis leaves correlations near the originals; the pooled 0.5B handover difference at $\tau = 0.25$ changes from -0.504 to -0.500. This checks sensitivity to these flags, without certifying that unflagged items were absent from training.

Difficulty and generator extensions. The MATH-500 difficulty stratification gives Qwen/OLMo correlations of 0.315 and 0.460 on levels 1 to 3, versus 0.337 and 0.464 on levels 4 and 5, for Qwen and Llama text. The harder groups contain 177 and 166 verified items, respectively. On Qwen3-30B-A3B text, the pooled core correlation is 0.520 across 293,082 words. The R1-Distill-Qwen panel correlates 0.255/0.379 with Qwen and 0.267/0.369 with OLMo, on Qwen/Llama text.

Acceptance proxies. For each of Qwen and OLMo, we compare its smallest and largest model using greedy disagreement and a soft overlap computed from cached truncated probabilities. Cross-family partial correlations are 0.150/0.199 for greedy rejection and 0.179/0.384 for soft rejection, versus 0.322/0.417 for reproduction indices. Truncated overlap omits probability mass and is a proxy for

Table A6: Concentration on Qwen-generated verified references under two cost conventions. Unresolved is the percentage of all native tokens with no matching panel model. Resolved-only columns give FLOP shares for the top 1% / top 10% of resolved tokens. All-token columns give the corresponding shares among all tokens after assigning each unresolved position twice the largest model's prediction FLOPs. All entries are percentages.

Panel	Dataset	Unresolved	Resolved only	All tokens
Qwen	GSM8K	0.76	41.76 / 60.11	56.86 / 79.54
Qwen	MATH-500	0.49	35.84 / 50.85	60.14 / 71.98
Qwen	HumanEval	0.14	45.18 / 56.72	52.94 / 63.66
OLMo	GSM8K	2.80	14.09 / 41.09	17.22 / 68.57
OLMo	MATH-500	2.99	15.15 / 44.39	16.25 / 69.37
OLMo	HumanEval	2.08	14.84 / 39.55	19.87 / 63.75
R1 distill	GSM8K	5.73	11.84 / 31.67	11.47 / 74.76
R1 distill	MATH-500	4.77	11.68 / 26.03	13.27 / 71.01
R1 distill	HumanEval	3.59	9.91 / 31.59	15.02 / 67.19

acceptance, not an exact full-vocabulary speculative acceptance probability. These correlations compare the operational signals under the same controls.

Scope of cross-family comparisons. The cross-family comparisons do not separately control architecture, tokenization, training data, or instruction tuning. The R1-Distill-Qwen panel shares Qwen lineage, and independently developed families may also share training data. Positive associations therefore establish shared ordering within the evaluated panels, without identifying its source or demonstrating that a map transfers between families. The reference-variation and memorization-screening analyses test sensitivity to particular alternatives; they do not establish the absence of training overlap or memorization.

C.3 Comparison with FlexiDepth allocations

The released FlexiPatterns records comprise 600 generations and 82,783 labeled native tokens. We align layer allocations and panel indices to words and omit the first word of each response because it includes prefill effects. The pooled analysis subtracts each task split's mean from allocation and panel index, then computes partial rank correlation controlling log position and word length. Within-split estimates use the same controls. This centering procedure removes split means before the rank transformation. The pooled partial correlations are 0.209 for Qwen and 0.213 for OLMo. On the 68 verified addition outputs, the Qwen correlation is 0.273. Multiplication provides an allocation comparison on incorrect generations, with no verified subset.

D Intervention protocols and full results

Evaluation population and conditions. The handover experiments use the 1,758 core problems with verified references from both generators (1,275 GSM8K, 347 MATH-500, and 136 HumanEval). The supplied prefix is taken from the Qwen reference. A smaller model and the 72B control start from the same prefix; subsequent tokens are generated without additional reference tokens. Accuracy differences are paired by problem after averaging 16 completions per condition. The length sweep uses 0.5B, 1.5B, and 7B at $L \in \{1, 2, 4, 8, 16, 32, 64\}$ from the reference midpoint. Full-suffix handovers additionally include 32B and vary $\tau \in \{0, 0.25, 0.5, 0.75, 0.9\}$.

Prefix construction and pairing. For a reference with C characters, a handover at fraction τ starts at $\lfloor \tau C \rfloor$, snapped backward to whitespace. The generated text is appended to this supplied prefix before verification. All models receive the same problem and prefix within a comparison. Each condition has 16 completions at temperature 0.7 and top-p 0.95. Seeds are deterministic functions of item and position for full-suffix handovers, or item and seed index for the length sweep. Reusing seeds organizes paired conditions; it does not couple model outputs token by token.

The full-suffix analysis contains 35,156 paired item/size/start cells. Four of the nominal $1,758 \times 4 \times 5$ cells collapse because snapped start positions coincide. The pooled size ordering has zero inversions

Table A7: Agreement with FlexiDepth layer allocations, with bootstrap SDs. Each split has 100 responses; the first word is excluded.

Task	Words	Qwen panel	OLMo panel
copy	24,770	0.061 ± 0.008	0.062 ± 0.008
continue	7,755	0.277 ± 0.049	0.267 ± 0.048
summarize	3,840	0.283 ± 0.014	0.294 ± 0.013
repeat	3,864	0.130 ± 0.005	0.120 ± 0.004
add	5,121	0.289 ± 0.013	0.291 ± 0.013
product	7,533	0.405 ± 0.011	0.389 ± 0.012

among 15 adjacent comparisons. The starting-position ordering has two inversions among 16 comparisons. Table A9 supplies all dataset-specific estimates.

Length-sweep control. The smaller model emits at most L native tokens, possibly stopping earlier. Its sampled text is appended to the verified midpoint prefix; 72B then receives that combined text in a new request and continues. The control is a single uninterrupted 72B request from the midpoint. Both 72B requests have the usual dataset output cap (512 for GSM8K and HumanEval, 1,024 for MATH-500). Thus the intervention can generate up to L additional tokens before the 72B continuation and resets the sampler at the handback. There are $1,758 \times 7 \times 3 \times 16 = 590,688$ small-model completion conditions, each compared with its item/length control.

Reference cost and continuation loss. Across the same 1,758 items, the fraction of suffix words assigned to the 14B rung or higher has Spearman correlation 0.195 with the midpoint 0.5B accuracy loss. This aggregate reference-cost statistic provides only a weak item-level predictor of the outcome of the handover. A handover changes the generating policy and hence subsequent prefix states, so its effect includes deviations, alternative solution paths, and recovery. This comparison separates the local matching measurement from the downstream consequences of choosing a policy for a longer continuation.

E Allocation protocols and complete policy comparisons

This appendix specifies the full-dataset routing and drafting experiments in Figure 5, including the timing projection.

E.1 Model routing and speculative decoding on MATH-500

The allocation experiment in Figure 5 evaluates ten policies on all 500 MATH-500 problems. Table A10 reports every point in that figure and additionally logical FLOPs. These policies use Qwen2.5-0.5B-Instruct as the small model and Qwen2.5-72B-Instruct as the target, with greedy decoding and a common 1,024-token output cap. Each policy's answer is checked independently using the same symbolic-equivalence checker. No problem is excluded because the target or another policy gives an incorrect answer.

Information supplied to the oracle. For each problem, a fresh greedy target response provides a reference $z = (z_1, \ldots, z_T)$, including its recorded termination token where present. The map stores

$$b_t = \mathbf{1}\left[\arg\max_v \pi_{0.5\mathrm{B}}(v \mid x, z_{<t}) = z_t\right]. \tag{17}$$

This is the small-model matching component of the sufficient-compute measurement. The target response is retained regardless of correctness, so the map describes agreement with that response rather than a guarantee of a correct answer. The controller receives the matching indicators. Reference generation and map acquisition are supplied information and are excluded from policy resource counts. Thus the experiment isolates the allocation opportunity available with advance knowledge of agreement.

Routing policies. The confidence baseline implements the TokenCascade Chow rule (Narasimhan et al., 2025). At the current generated prefix s, it uses the small model when $\max_v \pi_{0.5\mathrm{B}}(v \mid s) \geq \theta$

Table A8: Handover-length accuracy differences (percentage points), with bootstrap SDs (standard errors).

Maximum tokens	0.5B	1.5B	7B
1	-0.09 ± 0.19	-0.30 ± 0.16	-0.06 ± 0.15
2	-0.17 ± 0.22	-0.27 ± 0.18	-0.04 ± 0.16
4	-0.30 ± 0.24	-0.17 ± 0.19	-0.00 ± 0.18
8	-0.86 ± 0.26	-0.71 ± 0.21	-0.18 ± 0.19
16	-2.19 ± 0.30	-1.37 ± 0.26	-0.22 ± 0.21
32	-4.53 ± 0.39	-2.43 ± 0.30	-0.55 ± 0.23
64	-9.44 ± 0.50	-4.68 ± 0.39	-1.50 ± 0.30

Table A9: Full-suffix handover accuracy differences (percentage points). Prefix fractions use characters, snapped to a word boundary.

Dataset	τ	0.5B	1.5B	7B	32B
GSM8K	0.0	-52.88	-23.39	-4.51	-0.16
GSM8K	0.25	-48.25	-21.17	-4.12	-0.50
GSM8K	0.5	-35.08	-13.92	-3.17	-0.74
GSM8K	0.75	-20.60	-7.15	-1.77	-0.34
GSM8K	0.9	-10.62	-3.07	-0.72	-0.08
MATH-500	0.0	-55.76	-28.85	-5.08	+1.17
MATH-500	0.25	-57.01	-28.91	-5.49	-0.76
MATH-500	0.5	-45.59	-20.55	-4.11	-0.05
MATH-500	0.75	-32.17	-13.08	-3.19	-0.32
MATH-500	0.9	-16.35	-7.53	-1.49	+0.02
HumanEval	0.0	-60.98	-33.04	-6.71	-2.80
HumanEval	0.25	-53.91	-29.50	-10.02	-2.11
HumanEval	0.5	-41.36	-19.39	-6.20	-1.61
HumanEval	0.75	-40.26	-22.10	-6.43	-1.93
HumanEval	0.9	-18.56	-7.59	-2.27	+0.93

and otherwise requests a target prediction. We evaluate all four thresholds $\theta \in \{0.5, 0.8, 0.95, 0.99\}$, together with standalone small- and target-model baselines. MoA-map routing selects the small model at mapped positions with $b_t = 1$. Before emitting that prediction, it checks that the actual small-model token equals the stored reference token; otherwise it requests the target. After the generated prefix departs from the reference, it uses the target. All routing policies use the same batch scheduler, which collects deferred target requests after at most 16 small-model steps. The target processes any missing prefix positions when it is next invoked.

Speculative policies. The fixed baseline proposes up to 16 small-model tokens before target verification. The adaptive baseline follows confidence-based dynamic lookahead (Mamou et al., 2024): it appends each proposed token and stops the block if that token's probability is below 0.4, a termination token is reached, or 16 tokens have been proposed. MoA-map drafting uses the length of the upcoming consecutive run of $b_t = 1$, capped at 16 and by the remaining output allowance. A zero-length run invokes the target directly. After departure from the mapped prefix, it uses the adaptive baseline. For every policy, the target verifies a proposed block through greedy token equality, accepts its matching prefix, and supplies the first correction token or a bonus token when the full block matches. Both caches are truncated to the resulting accepted prefix.

Execution and resource accounting. The live allocation runs use a common PyTorch 2.9.1 research backend with bfloat16 weights, math SDPA attention, and float32 vocabulary projection. The target is distributed over four A100 80GB GPUs; the draft shares one of these GPUs. Batches contain at most 32 problems, and a forward call processes at most 128 new positions. This backend differs from the vLLM reference-generation pipeline used for the reproduction study.

The logical FLOP ledger applies the architecture terms in Appendix B.1 to actual forwarded positions and requested logits. It includes prompt prefill, cache catch-up, rejected drafts, and target verification. The ledger does not measures memory traffic or communication.

Table A10: All ten allocation policies on the full MATH-500 evaluation set (500 problems). Accuracy is independently checked for each generated response. Projected time includes prompt processing and cached model calls under Appendix E.2; it excludes oracle-map acquisition and is not measured deployment latency. Values are means $\pm$ one item-bootstrap SD. Confidence thresholds control small-model selection; adaptive drafting stops below probability 0.4 with a 16-token cap. MoA-map policies use advance reference-token agreement.

Policy	Accuracy (%)	Projected time (s)	Compute (TFLOPs)
0.5B only	28.20 ± 2.04	1.84 ± 0.04	0.67 ± 0.01
72B only	69.20 ± 2.11	19.03 ± 0.38	101.92 ± 1.87
Confidence routing, $\theta = 0.5$	40.00 ± 2.25	2.80 ± 0.06	86.57 ± 1.90
Confidence routing, $\theta = 0.8$	60.00 ± 2.21	4.45 ± 0.10	99.50 ± 1.85
Confidence routing, $\theta = 0.95$	66.60 ± 2.21	5.91 ± 0.13	99.16 ± 1.84
Confidence routing, $\theta = 0.99$	68.40 ± 2.14	7.59 ± 0.18	100.38 ± 1.89
MoA-map routing	69.60 ± 2.10	5.12 ± 0.26	96.71 ± 1.86
Fixed drafting, 16 tokens	69.00 ± 2.13	4.98 ± 0.11	154.30 ± 3.27
Adaptive drafting	69.00 ± 2.13	4.69 ± 0.10	140.52 ± 2.75
MoA-map drafting	69.40 ± 2.11	3.99 ± 0.09	112.07 ± 2.31

Complete comparisons and their interpretation. Against confidence routing at $\theta = 0.99$, MoA-map routing increases accuracy from 68.4% to 69.6% and reduces projected time from 7.59 to 5.12 seconds. MoA-map drafting proposes 301,691 tokens across the dataset, compared with 447,731 for the fixed-window policy, a 32.6% reduction. Accepted draft-token counts are similar: 268,577 and 269,286. The map therefore mainly reduces proposals that do not survive verification. The mean number of target calls is also similar, 57.62 versus 56.87 per response. Projected times are 3.99 versus 4.98 seconds. Table A11 gives paired comparisons for every tested routing threshold and both speculative baselines.

Output agreement and uncertainty. Finite-precision block and incremental evaluations need not produce identical greedy predictions. Map fallback occurs on 100 routing responses and 169 speculative responses. All departures and length-capped responses remain in the accuracy and resource totals. Consequently, small accuracy differences should be read as empirical outcomes of this backend, rather than evidence that changing speculative draft length intrinsically improves an exactly preserved target policy.

Means and paired differences use 2,000 bootstrap resamples of the 500 problem identifiers. Tables report one bootstrap SD, which describes sampling variation across problems rather than repeated timing runs. Projection uncertainty from the timing assumptions is examined separately in Appendix E.2.

E.2 Projection from published inference benchmarks

We separate the measured answer accuracies from an illustrative latency projection. Accuracies use all 500 MATH-500 problems generated with our experimental backend. The horizontal coordinates instead describe a warm, single-request execution scenario anchored to the official Qwen2.5 BF16 benchmarks for vLLM 0.6.3 on A100 80GB GPUs: four GPUs for the 72B target and one for the 0.5B model (Qwen Team, n.d.). We assume both models remain resident and execute sequentially without interference. The published one-token-input rates are 31.37 and 311.55 output tokens per second, respectively, for 2,048-token outputs. These rates measure whole requests, rather than isolated cached forwards.

For each model m, let K_m denote the number of recorded forward calls and F_m the total number of input positions processed, including prompt tokens, cache catch-up, and rejected speculative tokens. We use

$$\widehat{T} = \sum_{m \in \{\text{small},\text{target}\}} \left[a_m K_m + b_m (F_m - K_m) \right]. \tag{18}$$

A single-position call costs a_m; processing n positions together costs $a_m + b_m(n-1)$. Thus, parallel verification need not cost as much as n sequential target calls. Prompt processing is included in

Table A11: Paired differences between each MoA-map policy and the indicated baseline on the same 500 problems (map minus baseline). Positive accuracy differences favor the map; negative time differences favor the map. Accuracy is in percentage points (pp), and time differences are in seconds. Values show the mean difference ± one item-bootstrap SD; projected time follows Appendix E.2.

Baseline	Δ accuracy (pp)	Δ projected (s)
Routing, $\theta = 0.5$	29.60 ± 2.16	2.32 ± 0.24
Routing, $\theta = 0.8$	9.60 ± 1.60	0.67 ± 0.23
Routing, $\theta = 0.95$	3.00 ± 1.04	-0.79 ± 0.22
Routing, $\theta = 0.99$	1.20 ± 0.62	-2.46 ± 0.21
Fixed drafting	0.40 ± 0.29	-1.00 ± 0.04
Adaptive drafting	0.40 ± 0.29	-0.71 ± 0.03

the same ledger, so it is not added a second time. The projection retains the recorded call counts, including chunked processing, and is computed separately for every response before averaging across items.

We anchor a_m at the inverse of the shortest-input published generation rate. To set b_m, write the published total time at input length P as $T_m(P) = 2048/v_m(P)$ and fit

$$b_m = \arg\min_{b \geq 0} \sum_{P \in \{6144, 14336, 30720\}} \left[T_m(P) - 2048 a_m - b(P-1)\right]^2 . \tag{19}$$

This gives $(a_m, b_m) = (3.210, 0.03683)$ milliseconds for 0.5B and $(31.878, 0.28557)$ milliseconds for 72B. The fit reproduces these published whole-request times with a maximum relative error of 2.13%. The fit describes the published whole-request measurements; cached block-verification timings are not separately calibrated.

The substantive assumption is transferring the marginal cost of a longer input to extra positions within a cached forward call. Longer-input timings also include increased attention cost during subsequent generation; therefore, the public benchmark does not separately identify prefill, cache catch-up, or verification costs. The projection does not model their kernel differences, context-dependent nonlinearities, model-switching costs, or additional controller overhead. It also excludes queueing, model loading, network delivery and oracle-map acquisition. Consequently, these coordinates are conditional model-execution estimates, not measured end-to-end deployment latencies or a demonstrated vLLM speedup. The answer accuracies remain those of the original backend; identical outputs after migration to another engine have not been established.

We assess sensitivity using bulk-position coefficients scaled by $\{0, 0.5, 1, 2\}$, small-model times scaled by $\{0.5, 1, 2\}$, and an added cost of $\{0, 0.5, 1\}$ target-call equivalents per speculative verification. All 36 combinations are applied to the full dataset, without selecting parameters according to policy ranking. Within these scenarios, MoA-map drafting has lower projected latency than fixed-window and adaptive drafting; its relative reductions range from 8.3–24.7% and 7.3–18.4%, respectively. These ranges describe chosen assumptions, not statistical intervals or guaranteed prediction bounds. In particular, they do not cover arbitrary policy-specific controller overhead or unobserved block cost nonlinearities.

F Relation to earlier measurements

Sequence prediction has long been formulated through policies and future rewards (Ranzato et al., 2015; Bahdanau et al., 2017). Controlled Decoding and Transfer Q^* use this formulation for decoding-time alignment (Mudgal et al., 2024; Chakraborty et al., 2024). COLLAB selects off-the-shelf model policies in a shared decoding MDP using long-term utility (Chakraborty et al., 2025); CITER formulates small/large-model token routing as an MDP (Zheng et al., 2025). Our measurement assigns a cost to a specified next-token action at a reference state. The handover and allocation experiments then examine the consequences of letting selected agents generate the subsequent states. This connects the measurement to the policy-switching problem while keeping local reproduction and terminal task reward distinct.

Several prior methods expose token-dependent computational behavior through a particular inference system. Duo-LLM searches internal module allocations to minimize perplexity under a compute budget (Alizadeh-Vahid et al., 2024). FS-GEN examines disagreement and collaboration frequencies for small and large models (Zhang et al., 2024). FlexiDepth records a learned layer allocation policy within Llama-3-8B (Luo et al., 2025). Our contribution is the cross-family measurement design: independently score a shared, checker-passing trajectory across several model sizes, compare the resulting orderings, and test their relation to continuation outcomes.

This design retains the exact-reference criterion's dependence on wording. Judge Decoding demonstrates that valid continuations can be rejected by alignment-based acceptance (Bachmann et al., 2025). Top-five and probability thresholds change our matching rule but still evaluate a chosen reference. The shared-prefix experiment therefore measures reproducibility of a successful trajectory, while the handovers assess task correctness after the trajectory is allowed to change.

Generation-stage switching is an established design class. Reasoning Relay studies the transfer of partial reasoning between models (Lu et al., 2025); LLM Shepherding uses a larger model's prefix to support a smaller model (Dong et al., 2026); RelayGen selects segment-level switch points for efficient inference (Song et al., 2026). The handover experiments here vary capacity, starting position, and duration under a common task checker. They characterize the relationship between conditional reproduction and sustained continuation, rather than introducing model switching itself.