TAPS: Task Aware Proposal Distributions for Speculative Sampling

Anonymous Authors[1]

Abstract

Speculative decoding accelerates autoregressive generation by using a lightweight draft model to propose future tokens that a larger target model verifies in parallel. While recent work has improved draft architectures and verification procedures, the role of the draft training distribution remains less understood. In this work, we study task aware proposal distributions for speculative decoding by training lightweight HASS and EAGLE 2 drafters on MathInstruct, ShareGPT, and mixed data variants, and evaluating them on MT Bench, GSM8K, MATH 500, and SVAMP. Our detailed analysis focuses on Meta Llama 3 8B Instruct, with additional scaling results for Qwen3 1.7B, Qwen3 4B, and Vicuna 13B v1.3. Measured by acceptance length under lossless verification, we find that draft training data induces clear specialization: MathInstruct trained drafters are strongest on GSM8K and MATH 500, while ShareGPT trained drafters are strongest on MT Bench. Mixed data training improves robustness, but larger mixtures do not uniformly dominate across decoding temperatures. We further compare ways of combining specialized drafters and find that naive checkpoint averaging performs poorly, while confidence routing and merged tree verification preserve specialization more effectively at inference time. Finally, confidence provides a clearer routing signal than entropy, although entropy remains useful for diagnosing rejected tokens. Our claims concern proposal alignment under lossless verification; speedups are reported as supporting system measurements rather than as the primary optimization target. Overall, speculative decoding quality depends on the match between draft training data and downstream workload, and specialized drafters are better combined at inference time than through naive weight space averaging.

1. Introduction

Large language models (LLMs) achieve strong performance across many tasks, but autoregressive decoding remains a major inference bottleneck because each token must be generated conditioned on the full previous prefix (Brown et al.; Leviathan et al., 2023; Chen et al., 2023). Speculative decoding addresses this bottleneck by using a lightweight draft model to propose several future tokens that a larger target model verifies in parallel. When the verification rule is applied exactly, this procedure can improve generation efficiency while preserving the target model's output distribution.

The effectiveness of speculative decoding, however, depends critically on the quality of the proposal distribution produced by the drafter. Most prior work improves this proposal process through better draft architectures or more efficient verification procedures. Early speculative decoding methods use a separate lightweight drafter (Leviathan et al., 2023; Chen et al., 2023), while later methods improve feature level drafting and dynamic tree construction, as in EAGLE, EAGLE-2, EAGLE-3, and HASS (Li et al., 2024a;b; 2025; Zhang et al., 2025). Other lines of work explore tree verification, self speculative decoding, hierarchical drafting, cascaded drafters, and retrieval assisted proposals (Miao et al., 2024; Zhang et al., 2024; Elhoushi et al., 2024; Liu et al., 2024; Sun et al., 2024; Chen et al., 2024; He et al., 2024). Despite this progress, the role of the draft training distribution remains less understood, even though many drafters are trained on broad generic corpora such as ShareGPT. Figure 1 summarizes the setting studied in this paper.

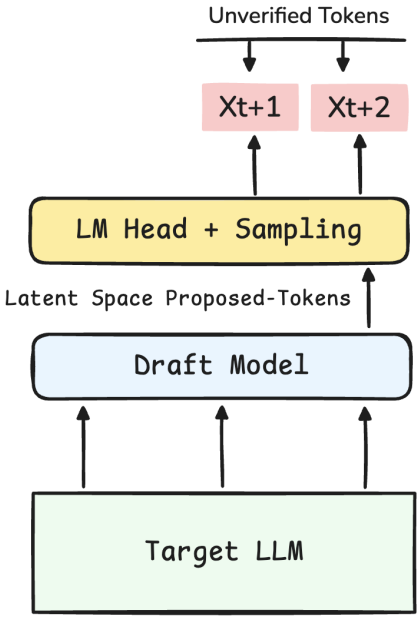


Figure 1. Schematic of the speculative decoding pipeline. Given contextual information from the target LLM, the draft model generates latent proposed tokens, which are converted by the LM head and sampling module into multiple candidate future tokens. These candidates are provisional and are later verified by the target model. Importantly, the trainable component in this framework is the draft model, whose role is to efficiently approximate the target model's next-token behavior while preserving the target model's final output distribution after verification.

This gap matters because the drafter is not only an architectural component. It is also a learned proposal distribution whose behavior can depend strongly on the data used to train it. A drafter trained on conversational data may produce proposals that are well aligned with open ended dialogue but less aligned with mathematical reasoning. Conversely, a drafter trained on mathematical data may better match reasoning benchmarks but be less suitable for conversational workloads. As a result, speculative decoding quality may degrade when the draft training distribution is mismatched

with the downstream workload, even if the verifier and the lossless acceptance rule remain unchanged.

This specialization creates a composition problem. The growing availability of specialized open weight checkpoints means that serving systems may have access to several plausible drafters for the same verifier (Sun et al., 2025; Ilharco et al., 2022; Mu & Lin, 2026). It is then not obvious whether those drafters should be mixed during training, merged directly in weight space, or kept separate and composed at inference time. Mixed data training may broaden coverage, but it can also weaken specialization. Weight space averaging is simple, but it may not preserve the functional behavior of either specialist. Inference time composition preserves the original checkpoints, but requires a policy for selecting or jointly verifying their proposals.

In this work, we study task aware proposal distributions for speculative decoding under fixed evaluation conditions. We train lightweight HASS and EAGLE-2 drafters on MathInstruct, ShareGPT, and mixed data variants, and evaluate them on MT-Bench, GSM8K, MATH-500, and SVAMP. Our detailed analysis focuses on Meta-Llama-3-8B-Instruct, with compact scaling results in Table 2 and full Qwen3-1.7B, Qwen3-4B, and Vicuna-13B v1.3 results in the appendix. Across experiments, the verifier, tokenizer, acceptance rule, benchmark suite, and decoding protocol are held fixed where applicable. The main variable is how the draft proposal distribution is trained or composed.

The paper is organized around five research questions. **RQ1.** Does task specific training improve speculative decoding on matched downstream tasks? **RQ2.** Can mixed data training recover cross domain robustness without erasing specialization? **RQ3.** How should multiple specialized draft models be combined: weight averaging, routing, or merged tree verification? **RQ4.** Are confidence, entropy, and depth wise acceptance useful signals for explaining routing and acceptance behavior? **RQ5.** How does speculative depth affect task aware drafting? The first three questions state the main claims, while the last two analyze why routing and depth shape acceptance.

Our results answer these questions with a consistent picture. First, single domain training produces specialization: MathInstruct trained drafters are strongest on GSM8K and MATH-500, while ShareGPT trained drafters are strongest on MT-Bench. SVAMP is the main boundary case, with benchmark and backbone dependent behavior rather than a uniform reasoning pattern. Second, mixed data training improves robustness, but larger mixtures do not uniformly dominate across decoding temperatures and backbones. Third, when multiple specialists are available, naive weight space averaging performs poorly, whereas inference time composition is substantially stronger. Confidence based routing improves over single domain baselines in our evaluated settings, and merged tree verification gives the strongest composition results. The supporting analyses show that confidence is more useful than entropy for routing, and that depth wise behavior is consistent with early coverage giving way to deeper specialization.

These findings suggest that speculative decoding should treat the drafter's training distribution as a first class design choice, alongside architecture and verification strategy. Overall, the paper shows that draft proposal quality is inseparable from the workload it is meant to serve, and that specialized drafters are better preserved through inference time composition than through naive weight space averaging. As open weight ecosystems such as Hugging Face continue to provide families of specialized models, it is natural to expect corresponding families of specialized drafters for agentic, coding, reasoning, and conversational workloads. This makes routing, merging, and jointly verifying multiple drafters increasingly important for serving heterogeneous LLM applications with task aware speculative decoding.

2. Preliminaries

We briefly review only the pieces of speculative decoding that are needed later: the lossless verification rule and the two drafting backbones used in our experiments, EAGLE-2 and HASS. In the main experiments, the verifier and acceptance rule are fixed; what changes is the draft training distribution and the way multiple specialized drafts are composed at test time.

2.1. Speculative Decoding

Speculative decoding uses a lightweight draft model p to propose K future tokens and a target model q to verify them in parallel, thereby reducing the number of expensive target-model calls. Given a prefix $x_{1:n}$, the draft model generates $\tilde{x}_{n+1:n+K}$ autoregressively. The target model then scores these candidates, and each drafted token $\tilde{x}_{n+t}$ is accepted sequentially with probability

$$\alpha_{n+t} = \min\left(1,\ \frac{q(\tilde{x}_{n+t} \mid x_{1:n+t-1})}{p(\tilde{x}_{n+t} \mid x_{1:n+t-1})}\right). \tag{1}$$

If rejection occurs, decoding instead samples from

$$r(x) \propto \max(0,\ q(x \mid x_{1:n+t-1}) - p(x \mid x_{1:n+t-1})). \tag{2}$$

This rejection-sampling correction preserves the target model's output distribution exactly while allowing multiple proposed tokens to be checked in a single verifier call. In the rest of the paper, acceptance length measures how often this lossless verification procedure approves long draft continuations.

2.2. EAGLE-2

EAGLE-2 inherits the EAGLE feature-level drafter and improves inference by replacing a fixed draft tree with a context-dependent dynamic tree (Li et al., 2024b). Let h_t denote the target model's second-to-last-layer feature at step t. Instead of autoregressing directly on tokens, the draft model g_ϕ predicts future features,

$$\hat{h}_{t+1} = g_\phi(\hat{h}_{\le t}, x_{\le t+1}), \tag{3}$$

and the predicted features are mapped to token probabilities through the target model's LM head. A compact training objective is

$$\begin{aligned}\mathcal{L}_{\text{EAGLE}} = &\sum_t \|\hat{h}_{t+1} - h_{t+1}\|_2^2 \\ &+ \lambda \sum_t \mathrm{CE}\Big(\mathrm{softmax}(W\hat{h}_{t+1}), x_{t+1}\Big).\end{aligned} \tag{4}$$

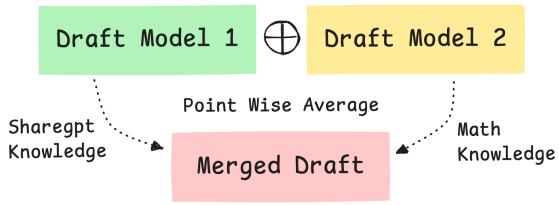
During verification, a drafted token $\hat{x}_{j+i}$ is accepted with probability

$$\alpha_{j+i} = \min\left(1, \frac{p_{j+i}(\hat{x}_{j+i})}{\hat{p}_{j+i}(\hat{x}_{j+i})}\right), \tag{5}$$

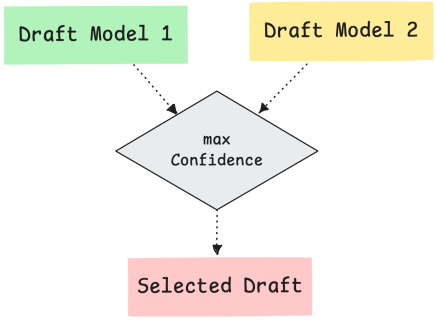
which preserves the target model's output distribution. EAGLE-2 uses draft confidence to rank frontier nodes in a dynamic draft tree,

$$V_i \approx \prod_{v_j \in \mathrm{Path}(\mathrm{root}, v_i)} c_j, \tag{6}$$

and expands the highest-valued frontier nodes before verification (Li et al., 2024b). In this paper, we keep that decoding rule fixed and vary only how the drafter is trained or combined.



(a) Checkpoint weight merging by point-wise parameter averaging. Draft models trained on ShareGPT and MathInstruct are combined by averaging corresponding parameters in weight space, producing a single merged draft that retains information from both domains.



(b) Confidence-based routing between specialized draft models. At inference time, the system selects the draft with the highest confidence for the current prompt, enabling task-aware use of specialized models without merging their parameters.

Figure 2. Two strategies for combining specialized draft models. Left: checkpoint weight merging in parameter space. Right: confidence-based routing at inference time.

2.3. HASS

HASS uses the same lossless speculative decoding framework, but improves the drafter by reducing objective mismatch and context mismatch between training and inference (Zhang et al., 2025). Its harmonized objective distillation term focuses learning on the verifier's most likely next tokens. Let $q(\cdot)$ and $p(\cdot)$ denote the next-token distributions of the target and draft models, respectively, and let $\hat{\Omega} \subset \Omega$ be the set of top-K tokens under q. The Top-K distillation loss is

$$\mathcal{L}_{\text{Top-K}} = -\sum_{x \in \hat{\Omega}} q(x) \log p(x). \tag{7}$$

HASS also introduces harmonized context alignment so that later draft predictions are trained on imperfect draft features rather than only clean target features. At alignment step j, the draft model predicts

$$\begin{aligned} P^{(s)}(x_{t+1} \mid x_{\le t}) &= \text{Head}\Big(f_{t+1}^{(s_j)}\Big) \\ &= \text{Head}\Big(M^{(s)}\big(f_t^{(s_{j-1})},\ f_1^{(l)} \oplus \cdots \oplus f_{t-j+1}^{(l)} \\ &\quad \oplus f_{t-j+2}^{(s_1)} \oplus \cdots \oplus f_t^{(s_{j-1})}\big)\Big), \end{aligned} \tag{8}$$

with training objective

$$\begin{aligned} \mathcal{L}_{\text{HASS}}^{(j)} = \sum_{t=1}^{T-1} \Big[&\text{CE}\big(P^{(l)}(x_{t+1} \mid x_{\le t}), \\ &P^{(s)}(x_{t+1} \mid x_{\le t})\big) + \mathcal{L}_{\text{aux}}\Big], \end{aligned} \tag{9}$$

where $\mathcal{L}_{\text{aux}}$ includes Top-K distillation and feature regression. As with EAGLE-2, our main experiments keep the verifier and the lossless acceptance rule fixed and study how different training distributions and composition strategies affect acceptance.

3. Experimental Setup and Composition Strategies

3.1. Common Setup

We study task-aware draft construction for speculative decoding under a fixed verifier. The detailed main analysis uses Meta-Llama-3-8B-Instruct (Llama Team, 2024) as the target model, while the additional target models are included as supporting evidence for whether the observed trends persist beyond a single verifier. The draft model (Llama Team, 2024) is a lightweight LLaMA-style decoder with one transformer layer, hidden size 4096, and roughly 0.8B parameters. Draft and target share the same tokenizer and vocabulary in this focused setting so that acceptance differences are not confounded by tokenization mismatch. Table 2 gives the compact cross-verifier check, and Appendix Tables 4–6 give the full additional tables.

We use acceptance length as the primary metric because it isolates the quality of the draft proposal distribution under a fixed verifier and lossless acceptance rule. It directly measures how many proposed tokens the verifier can approve before returning to standard target model decoding. End to end speedup is important, but it depends on additional system factors such as draft latency, verifier cost, batch size, tree size, hardware utilization, memory pressure, and scheduling overhead. Since our goal is to study how training data and drafter composition affect proposal quality, acceptance length is the cleanest controlled metric. We therefore report speedup deltas only for completeness, as a systems sanity check rather than as the main measure of method quality.

We evaluate on MT-Bench together with three reasoning-heavy benchmarks, GSM8K, MATH-500, and SVAMP, at temperatures 0 and 1. We keep benchmark evaluation sets separate from draft supervision and treat the training corpora only as sources for proposal learning, not as benchmark specific fine tuning data. Our primary metric is acceptance length averaged over the evaluation distribution under the lossless speculative decoding constraint. Unless stated otherwise, the only factors that change across experiments are the draft training distribution or the way multiple specialized drafts are combined.

All draft checkpoints are trained for 20 epochs with learning rate 3×10^{-5}, batch size 8, and gradient accumulation 1. HASS runs use the same auxiliary settings throughout: top-K distillation with $K = 10$, loss weight 1.0, and three forward-alignment steps. All training and evaluation experiments were run on a single node with four NVIDIA A100 GPUs. The study varies along four axes: speculative backbone (EAGLE-2 or HASS), training domain (ShareGPT for conversational data or MathInstruct for mathematical reasoning), mixed-data supervision (35k+35k or 70k+70k), and test-time composition strategy (Averaged, Confidence Routed, or Merged Trees).

For an input x, verifier M_T, and drafter M_D, let $A(x; M_D, M_T)$ denote the number of consecutively accepted draft tokens. We compare methods through

$$\mathbb{E}_{x\sim\mathcal{D}}[A(x; M_D, M_T)], \tag{10}$$

subject to the lossless speculative decoding constraint that the final output distribution remains identical to that of the verifier.

3.2. Draft Variants

We study seven main draft variants for each backbone. Two are single-domain checkpoints: one trained on 70k MathInstruct examples and one trained on 70k ShareGPT examples. Two are mixed-data checkpoints: Mixed 35k+35k and Mixed 70k+70k. The remaining three use the single-domain checkpoints as building blocks for composition: Averaged, Confidence Routed, and Merged Trees.

These variants map directly to the research questions in the experiments section. The single-domain checkpoints answer RQ1, the mixed-data checkpoints answer RQ2, and the three composition strategies answer RQ3. RQ4 then interprets these results through routing statistics, entropy, and depth-wise acceptance.

3.3. Checkpoint Averaging

Our simplest composition baseline is checkpoint averaging. Let θ_{math} and θ_{chat} denote the parameters of the MathInstruct and ShareGPT draft models. We define the merged checkpoint as

$$\theta_{\text{merge}} = \lambda\theta_{\text{math}} + (1-\lambda)\theta_{\text{chat}}, \tag{11}$$

where $\lambda \in [0, 1]$ controls the contribution of each checkpoint. We use $\lambda = 0.5$ for the main table and sweep λ in Figure 6.

3.4. Inference-Time Composition

We also study two inference-time alternatives that keep the single-domain checkpoints separate. Both strategies operate within one backbone at a time: HASS drafts are combined only with HASS drafts, and EAGLE-2 drafts only with EAGLE-2 drafts. This isolates the effect of composition from any cross-backbone differences.

Confidence routing. Given an input prefix, we decode one draft tree from the MathInstruct checkpoint and one from the ShareGPT checkpoint. We score each tree by its mean draft confidence. Let $\mathcal{T}_{\text{math}}$ and $\mathcal{T}_{\text{chat}}$ denote the two trees and let $c(v)$ denote the confidence assigned to node v. The tree-level score is

$$\text{Score}(\mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{v\in\mathcal{T}} c(v), \tag{12}$$

and the selected tree is

$$\mathcal{T}^* = \arg\max_{\mathcal{T}\in\{\mathcal{T}_{\text{math}},\mathcal{T}_{\text{chat}}\}} \text{Score}(\mathcal{T}). \tag{13}$$

Only $\mathcal{T}^*$ is passed to the verifier.

Merged-tree verification. Instead of selecting one tree and discarding the other, we can verify both trees jointly by packing them under a shared root. We preserve the node indices of one subtree, offset the other subtree by the size of the first, and build an attention mask that allows each node to attend only to the shared root and its own ancestors. Candidates from the MathInstruct subtree therefore do not attend to candidates from the ShareGPT subtree, and vice versa. Position ids are assigned by tree depth so that each child is placed one step deeper than its parent.

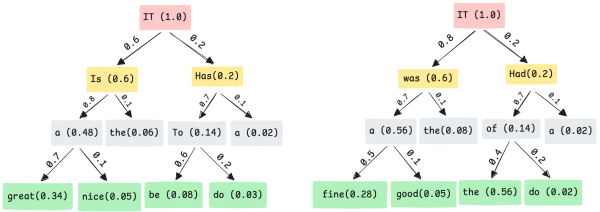


Figure 3. **Confidence Routing Between Specialized Trees.** The MathInstruct and ShareGPT checkpoints generate separate draft trees from the same prefix, with node labels indicating draft confidence. Confidence routing selects the tree with the higher mean node confidence before verification.

Algorithm 1 Merged-Tree Verification

Input: Prefix $y_{1:t}$, target M_T, drafts $M_{\text{math}}, M_{\text{chat}}$
Generate draft trees $\mathcal{T}_{\text{math}}$ and $\mathcal{T}_{\text{chat}}$ from the same root token
Merge the two trees under a shared root by concatenating nodes and remapping indices
Build ancestor-preserving attention masks and depth-based position ids
Verify the merged tree in one parallel pass with M_T
Extract candidate paths and apply standard speculative acceptance
Commit the accepted prefix
Return: accepted length

The merged tree increases proposal diversity at each verifier call because both specialists contribute candidate continuations. At the same time, it is a stricter test than routing because the verifier must process a larger tree. In this paper we report the acceptance length benefit of this strategy, but we do not claim an end-to-end latency improvement without a separate systems analysis.

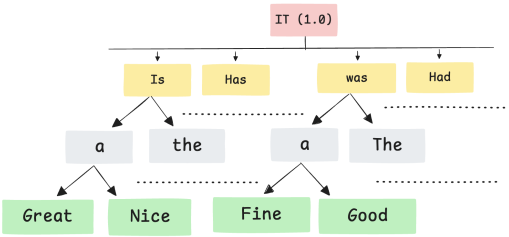


Figure 4. **Merged Verification Tree.** The MathInstruct and ShareGPT subtrees are packed under a shared root while preserving their internal ancestry. This lets the verifier evaluate both specialists in one pass and tests whether broader proposal coverage is more useful than selecting a single specialist.

4. Experiments

We report acceptance length, the average number of draft tokens accepted per verifier call. Higher acceptance length indicates that the drafter is better aligned with the verifier on the evaluated workload. Table 1 reports the main Meta-Llama-3-8B-Instruct results used for the detailed analysis, while Table 2 gives a compact verifier

Figure 5. **Flattened Merged-Tree Input.** The merged tree is serialized for verification while ancestry is preserved through the tree attention mask and depth-based position ids. This lets the verifier process both specialized subtrees without cross-subtree attention.

scaling check. Appendix Tables 4–6 give the same detailed table format for additional target models. The section then reads the main results in five passes: single-domain specialization (RQ1), mixed-data robustness (RQ2), specialist composition (RQ3), routing and diagnostic signals (RQ4), and depth-wise behavior (RQ5). Unless stated otherwise, all conclusions below are drawn from Table 1, Table 3, and the supporting figures.

Across all four target verifiers and both speculative backbones, the verifier scaling check in Table 2 shows that merged-tree verification remains the strongest composition method at both temperatures. Confidence routing also consistently improves over the best single-domain checkpoint, while weight-space averaging remains substantially weaker. Thus the main composition conclusion is not specific to the Llama-3-8B verifier or to one speculative backbone, although absolute acceptance lengths remain verifier-dependent.

Each subsection reads the relevant block of Table 1, points to the supporting figures when needed, and states the narrowest conclusion supported by that evidence.

4.1. RQ1: Does task specific training improve matched domain acceptance?

Question. Do drafters trained on a matched domain achieve longer acceptance lengths than drafters trained on a mismatched domain?

Setup. We compare the two single domain checkpoints in Table 1. MathInstruct is used as the mathematical reasoning specialist, while ShareGPT is used as the conversational specialist. Figure 8 gives a depth wise view of the same specialization behavior.

Answer. Mostly yes, with SVAMP as the main boundary case. Task specific training produces a clear specialization pattern at temperature 0 for both backbones. Under HASS, ShareGPT is stronger than MathInstruct on MT-Bench (3.98 vs. 2.90), while MathInstruct is stronger on GSM8K and MATH-500 (5.02 vs. 4.09 and 5.35 vs. 3.98). SVAMP is less uniform: ShareGPT is stronger under HASS, whereas MathInstruct is stronger under EAGLE-2. Under EAGLE-2, ShareGPT remains strongest on MT-Bench (3.57 vs. 2.54), while MathInstruct is strongest on GSM8K, MATH-500, and SVAMP. Figure 8 shows that this specialization persists across speculative depth, especially on reasoning heavy tasks.

Takeaway. Draft quality is not determined by the speculative decoding backbone alone. It also depends on how well the draft training distribution matches the downstream workload.

4.2. RQ2: Can mixed data training recover cross domain robustness?

Question. If single domain drafters specialize strongly, can mixed data training produce a more robust single checkpoint?

Setup. We compare the two mixed data checkpoints against the single domain checkpoints in Table 1. The Mixed 35k+35k variant uses a smaller balanced mixture, while Mixed 70k+70k doubles the amount of mixed supervision.

Answer. Mixed data training can broaden coverage, but its effect is not monotonic and does not uniformly improve over the best single-domain checkpoint. Under HASS at temperature 0, Mixed 70k+70k is the strongest trained checkpoint overall, with average acceptance length 5.18. At temperature 1, however, it drops to 3.69, below Mixed 35k+35k at 4.29. Under EAGLE-2, the same non-monotonic pattern appears: Mixed 70k+70k is strongest among mixed checkpoints at temperature 0 with average acceptance length 4.48, whereas Mixed 35k+35k is stronger at temperature 1 with 3.81 versus 3.26. However, under EAGLE-2 at temperature 1, the best single-domain checkpoint remains higher at 4.07. Thus, mixed training is useful as a robustness strategy in some regimes, but larger mixtures do not uniformly dominate across decoding temperatures, backbones, or single-domain baselines.

Takeaway. Mixed data training is a useful robustness strategy, but it does not remove the need to tune the mixture for the target decoding regime.

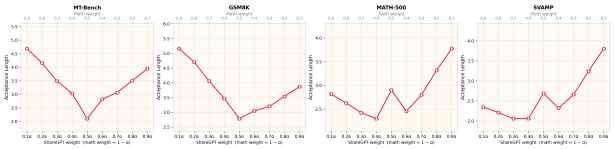


Figure 6. **Interpolation Sweep for Checkpoint Averaging.** Acceptance length is plotted against the interpolation weight between the MathInstruct and ShareGPT draft checkpoints under a fixed verifier setup. Weight-space averaging is unstable and remains well below the strongest inference-time composition methods.

4.3. RQ3: How should multiple specialized drafters be combined?

Question. When multiple specialized drafters are available, is it better to merge them in weight space or compose them at inference time?

Setup. We compare checkpoint averaging, confidence routing, and merged tree verification in Table 1. Figure 6 provides the interpolation sweep for checkpoint averaging.

Answer. Inference time composition is substantially stronger than weight space averaging. Averaged checkpoints are consistently the weakest variants in the main table, with average acceptance length between 2.34 and 2.62 across backbones and temperatures. By contrast, confidence routing improves to 4.80 and 4.63 average acceptance length at temperature 0 under HASS and EAGLE-2, respectively. Merged tree verification is the strongest composition strategy, reaching 5.11 for HASS and 5.03 for EAGLE-2 at temperature 0, and remaining the strongest composition method at temperature 1. Figure 6 reinforces this result: interpolating between the two checkpoints produces unstable behavior and never approaches the best inference time composition methods.

Takeaway. Merged trees are not a replacement for the best matched drafter in homogeneous traffic. They are a high coverage mode for heterogeneous traffic, while confidence routing is the latency conscious composition strategy.

These results suggest a deployment dependent interpretation. For homogeneous traffic, the best matched single domain or pre mixed checkpoint may still be preferable: under Llama 3 8B with HASS at temperature 0, Mixed 70k+70k reaches 5.18 accepted tokens at 3.14× speedup, slightly above merged trees at 5.11 accepted tokens and 2.42× speedup. For heterogeneous or shifting traffic,

Table 1. **Main Llama Results by Research Question.** Average acceptance length on MT-Bench, GSM8K, MATH-500, and SVAMP for HASS and EAGLE-2 at temperatures 0 and 1. Each numeric cell stacks acceptance length on top and wall clock speedup below in gray; speedup is target only decoding time divided by speculative decoding time. Bold marks the highest acceptance length in each benchmark or average column. Rows are grouped by the question they answer: RQ1 tests single-domain specialization, RQ2 mixed-data robustness, and RQ3 composition strategies. Higher is better for both values.

		Temperature 0					Temperature 1				
Model Variant	Method	MT-Bench	GSM8K	MATH-500	SVAMP	Average	MT-Bench	GSM8K	MATH-500	SVAMP	Average
RQ1. Task-specific training: single-domain checkpoints											
MathInstruct	HASS	2.90 1.72×	5.02 3.06×	5.35 3.26×	3.13 1.91×	4.10 2.49×	2.31 1.35×	4.75 2.83×	4.63 2.76×	2.46 1.47×	3.54 2.10×
MathInstruct	EAGLE-2	2.54 1.59×	5.04 2.90×	5.28 3.03×	4.81 2.76×	4.42 2.57×	2.43 1.49×	4.71 2.72×	4.61 2.66×	4.53 2.61×	4.07 2.37×
ShareGPT	HASS	3.98 2.36×	4.09 2.49×	3.98 2.43×	4.44 2.71×	4.12 2.50×	3.50 2.05×	4.03 2.40×	3.61 2.15×	3.95 2.35×	3.77 2.24×
ShareGPT	EAGLE-2	3.57 2.24×	3.72 2.14×	3.81 2.19×	3.71 2.13×	3.70 2.17×	3.38 2.07×	3.72 2.15×	3.43 1.98×	3.65 2.11×	3.54 2.08×
RQ2. Mixed-data training: robustness checkpoints											
Mixed 35k+35k	HASS	3.92 2.33×	4.77 2.91×	5.02 3.06×	4.15 2.53×	4.47 2.71×	3.46 2.02×	4.66 2.78×	4.47 2.66×	4.57 2.72×	4.29 2.55×
Mixed 35k+35k	EAGLE-2	3.37 2.11×	4.12 2.37×	4.44 2.55×	4.16 2.39×	4.02 2.36×	3.10 1.90×	4.08 2.35×	4.02 2.32×	4.03 2.32×	3.81 2.22×
Mixed 70k+70k	HASS	**4.13** 2.45×	**5.53** 3.37×	**5.67** 3.46×	**5.38** 3.28×	**5.18** 3.14×	3.17 1.85×	4.16 2.48×	3.42 2.04×	4.01 2.39×	3.69 2.19×
Mixed 70k+70k	EAGLE-2	3.75 2.35×	4.68 2.69×	4.85 2.79×	4.64 2.67×	4.48 2.62×	2.99 1.83×	3.76 2.17×	3.20 1.85×	3.08 1.78×	3.26 1.91×
RQ3. Combining specialists: weight averaging vs. inference-time composition											
Averaged	HASS	2.29 1.36×	2.80 1.71×	3.12 1.90×	2.13 1.30×	2.59 1.57×	2.10 1.23×	2.78 1.66×	2.90 1.73×	2.69 1.60×	2.62 1.55×
Averaged	EAGLE-2	2.07 1.30×	2.53 1.45×	2.57 1.48×	2.50 1.44×	2.42 1.42×	2.01 1.23×	2.49 1.44×	2.42 1.40×	2.45 1.41×	2.34 1.37×
Confidence Routed	HASS	3.93 2.20×	5.01 2.87×	5.37 3.08×	4.89 2.81×	4.80 2.74×	3.51 1.65×	4.72 2.25×	4.55 2.17×	4.71 2.24×	4.37 2.08×
Confidence Routed	EAGLE-2	3.63 1.92×	4.91 2.41×	5.25 2.58×	4.71 2.31×	4.63 2.30×	3.36 1.65×	4.65 2.18×	4.62 2.16×	4.46 2.09×	4.27 2.02×
Merged Trees	HASS	4.05 1.89×	5.42 2.58×	5.65 2.69×	5.31 2.53×	5.11 2.42×	**3.76** 1.38×	**5.21** 1.94×	**4.98** 1.86×	**5.05** 1.88×	**4.75** 1.77×
Merged Trees	EAGLE-2	3.93 1.67×	5.32 2.12×	5.63 2.25×	5.25 2.10×	5.03 2.03×	3.55 1.40×	5.01 1.90×	4.79 1.82×	4.93 1.87×	4.57 1.75×

Table 2. **Verifier Scaling Check.** Average-column acceptance length across target verifiers. Single and Mixed are the strongest trained checkpoints; Weight Avg., Routed, and Merged are composition strategies. This table reports acceptance only; Table 1 stacks acceptance over speedup for the main Llama verifier. Higher is better.

		Temperature 0					Temperature 1				
Verifier	Method	Single	Mixed	Weight Avg.	Routed	Merged	Single	Mixed	Weight Avg.	Routed	Merged
Qwen3-1.7B	HASS	2.37	2.67	1.69	2.58	2.75	2.21	2.27	1.58	2.39	2.56
Qwen3-1.7B	EAGLE-2	2.28	2.54	1.61	2.47	2.64	2.13	2.16	1.50	2.28	2.44
Qwen3-4B	HASS	2.30	2.59	1.64	2.50	2.67	2.15	2.20	1.53	2.33	2.48
Qwen3-4B	EAGLE-2	2.22	2.46	1.57	2.40	2.56	2.06	2.10	1.46	2.22	2.36
Llama-3-8B	HASS	4.12	5.18	2.59	4.80	5.11	3.77	4.29	2.62	4.37	4.75
Llama-3-8B	EAGLE-2	4.42	4.48	2.42	4.63	5.03	4.07	3.81	2.34	4.27	4.57
Vicuna-13B	HASS	5.14	5.70	3.29	5.56	5.86	4.70	5.06	3.08	5.21	5.51
Vicuna-13B	EAGLE-2	4.89	5.40	3.01	5.31	5.64	4.48	4.74	2.79	4.92	5.22

merged tree verification preserves MathInstruct and ShareGPT separately and exposes both proposal sets to the verifier. Across the four Llama 3 8B settings, merged trees raise acceptance from the best fixed single domain baseline from 4.10 to 4.87 accepted tokens on average, while still running at about 2.00× speedup. Thus merged trees should be viewed as a high coverage decoding mode for mixed workloads rather than as a peak throughput replacement for a matched drafter.

4.4. RQ4: What do confidence, entropy, and depth reveal about acceptance behavior?

Question. Are confidence and entropy useful signals for routing, and does depth wise acceptance help explain the observed specialization?

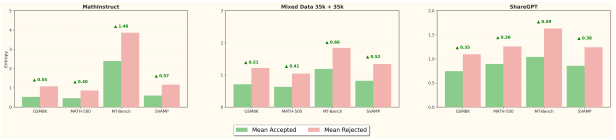
Setup. Table 3 compares benchmark level routing decisions under confidence based and entropy based selection for EAGLE-2. Figure 7 compares accepted and rejected token entropy, and Figure 8 reports acceptance by speculative depth for all main variants.

Answer. Confidence is useful for routing, while entropy is mainly diagnostic. Under confidence routing, the MathInstruct drafter is selected for 90.8% of GSM8K, 97.0% of MATH-500, and 93.0% of SVAMP examples, while ShareGPT is selected for 81.2% of MT-Bench examples. Entropy routing is far less discriminative, producing near balanced splits across all benchmarks. The entropy figures still show a consistent descriptive pattern: rejected tokens tend to have higher entropy than accepted tokens for both HASS and EAGLE-2. Figure 8 adds a depth wise view: acceptance decreases with speculative depth for every variant, but domain specialization remains visible and often becomes more pronounced deeper in the tree.

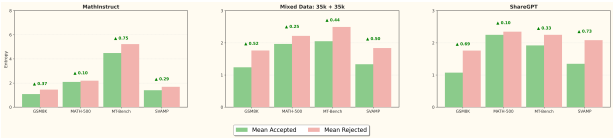
Takeaway. Confidence is the stronger decision signal for routing between specialized drafters. Entropy and depth wise acceptance are useful for interpreting verifier failures, but they are not sufficient by themselves to define a stronger routing policy.

	Confidence Routing			Entropy Routing		
Benchmark	MathInst.	ShareGPT	Total	MathInst.	ShareGPT	Total
MT-Bench	15 (18.8%)	65 (81.2%)	80	42 (52.5%)	38 (47.5%)	80
GSM8K	1198 (90.8%)	121 (9.2%)	1319	720 (54.6%)	599 (45.4%)	1319
MATH-500	485 (97.0%)	15 (3.0%)	500	312 (62.4%)	188 (37.6%)	500
SVAMP	279 (93.0%)	21 (7.0%)	300	159 (53.0%)	141 (47.0%)	300

Table 3. **Routing Decisions by Benchmark.** Benchmark-level routing counts for EAGLE-2 under confidence-based and entropy-based selection. Confidence routing separates conversational and mathematical workloads much more clearly than entropy routing.



(a) **EAGLE-2 Entropy.** Draft entropy for accepted and rejected tokens at temperature 0 across benchmarks and checkpoints. The averaged checkpoint is omitted for readability. Rejected tokens consistently exhibit higher entropy.



(b) **HASS Entropy.** Draft entropy for accepted and rejected tokens at temperature 0 across benchmarks and checkpoints. The averaged checkpoint is omitted for readability. The same accepted-versus-rejected separation remains visible.

Figure 7. **Accepted vs. Rejected Token Entropy.** Both panels compare draft entropy at temperature 0 for EAGLE-2 and HASS on the same benchmark suite and checkpoint families. Entropy is a useful diagnostic of rejection, but Table 3 shows that it is weaker than confidence for routing.

4.5. RQ5: How does speculative depth affect task aware drafting?

Question. Does speculative depth reveal a shift from broad proposal coverage at shallow levels to stronger reliance on a task matched specialist at deeper levels?

Setup. We use Figure 8 and the depth tables to compare how the main draft variants behave as speculative depth increases across benchmarks.

Answer. The depth wise results are consistent with a coverage to specialization pattern. At shallow depths, mixed data drafters often perform best, suggesting that broader proposal coverage increases the chance of producing acceptable early branches. As depth increases, the task matched specialist becomes more competitive and often more dominant, especially on GSM8K and MATH-500.

Deeper acceptance requires sustained agreement between the drafter and the verifier along a longer candidate path. The composition results follow the same logic. Merged trees perform well by preserving proposal diversity across specialists, while confidence routing helps when the system must select a single drafter before verification.

Takeaway. Speculative decoding appears to be both task aware and depth aware. Early proposal steps benefit more from coverage, while deeper accepted paths increasingly favor the better matched specialist.

5. Discussion and Limitations

The experiments support a simple but practically important conclusion: speculative decoding depends not only on the drafting backbone, but also on the match between draft training distribution and target workload. A drafter is a learned proposal model, so a mismatched drafter can be predictably weaker on particular task families. The supporting verifier checks show that this behavior is not specific to the main Llama verifier: absolute acceptance lengths vary across Qwen3-1.7B, Qwen3-4B, Meta-Llama-3-8B-Instruct, and Vicuna-13B v1.3, but the qualitative ordering of the composition strategies remains stable.

This perspective becomes more relevant in an open weight ecosystem where many specialized models are already available and easy to compose. As users rely on different models for coding, reasoning, dialogue, tool use, and agentic workflows, serving systems may also need families of open weight drafters. Our results do not solve this full serving problem, but they show why it matters: task specialization appears directly in acceptance length, and preserving specialized drafters at inference time is more effective than collapsing them through naive weight space averaging.

The deployment implication is conditional. Mixed data training broadens coverage, but it does not remove specialization. For homogeneous traffic, the best matched single domain or pre mixed checkpoint may still be the fastest operating point. For heterogeneous or shifting traffic, confidence routing and merged tree verification preserve the original specialists and decide how to use their proposals at inference time. These strategies improve acceptance length, but they also introduce overhead, so they should be viewed as coverage strategies rather than as guaranteed wall clock speedup improvements.

The present evidence has clear boundaries. The detailed analysis focuses on Meta-Llama-3-8B-Instruct, with supporting checks for Qwen3-1.7B, Qwen3-4B, and Vicuna-13B v1.3. We study two

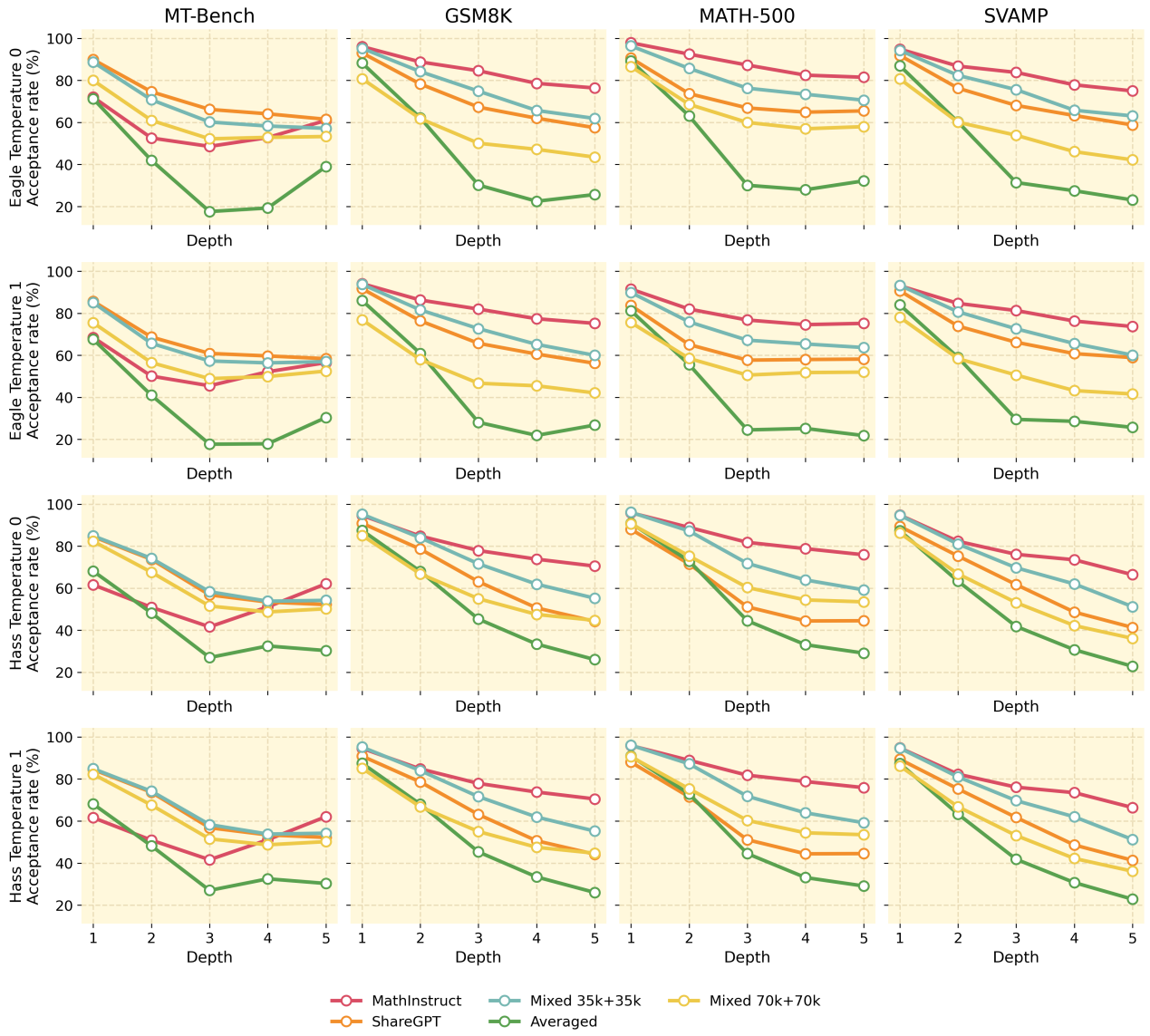


Figure 8. **Acceptance by Speculative Depth.** Acceptance rate is shown by draft depth for HASS and EAGLE-2 at temperatures 0 and 1 across MT-Bench, GSM8K, MATH-500, and SVAMP. Acceptance declines with depth for all variants, while domain specialization remains visible and often sharpens on reasoning-heavy tasks.

source domains, two speculative backbones, and four benchmarks. The routing policy is intentionally simple and confidence based, and we do not evaluate production constraints such as batching, memory pressure, request level routing latency, hardware specific verifier throughput, or scheduling in multi request serving.

6. Conclusion

We studied task aware draft training and composition for speculative decoding. The results show that training distribution affects proposal quality, mixed data improves robustness without uniformly dominating, and inference time composition preserves specialists better than naive weight space averaging. Task-aware drafter design is a concrete direction for future speculative decoding systems.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *NeurIPS*, pp. 1877–1901. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with

speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Chen, Z., Yang, X., Lin, J., Sun, C., Chang, K. C., and Huang, J. Cascade speculative drafting for even faster llm inference. *NeurIPS*, 37:86226–86242, 2024.

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *ACL*, pp. 12622–12642, 2024.

He, Z., Zhong, Z., Cai, T., Lee, J., and He, D. REST: Retrieval-based speculative decoding. In Duh, K., Gomez, H., and Bethard, S. (eds.), *ACL*, pp. 1582–1595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.88. URL https://aclanthology.org/2024.naacl-long.88/.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: speculative sampling requires rethinking feature uncertainty. In *ICML*, ICML'24. JMLR.org, 2024a.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-2: Faster inference of language models with dynamic draft trees. In *EMNLP*, pp. 7421–7432, 2024b.

Li, Y., Wei, F., Zhang, C., and Zhang, H. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *ACL*, 2025.

Liu, F., Tang, Y., Liu, Z., Ni, Y., Han, K., and Wang, Y. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*, 2024.

Llama Team, A. . M. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pp. 932–949, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703867. doi: 10.1145/3620666.3651335. URL https://doi.org/10.1145/3620666.3651335.

Mu, S. and Lin, S. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications, 2026. URL https://arxiv.org/abs/2503.07137.

Sun, H., Chen, Z., Yang, X., Tian, Y., and Chen, B. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024.

Sun, J., Zheng, C., Xie, E., Liu, Z., Chu, R., Qiu, J., Xu, J., Ding, M., Li, H., Geng, M., et al. A survey of reasoning with foundation models: Concepts, methodologies, and outlook. *ACM Computing Surveys*, 57(11):1–43, 2025.

Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *ACL*, pp. 11263–11282, 2024.

Zhang, L., Wang, X., Huang, Y., and Xu, R. Learning harmonized representations for speculative sampling. In *ICLR*, 2025.

A. Appendix

This appendix collects the additional target-model scaling results, the supporting entropy tables, the tree-merging utility used for merged-tree verification, and the depth-wise acceptance tables that complement Figure 8. The goal is to make the evidence behind the main-text claims easy to audit without interrupting the main narrative.

A.1. Scaling Results by Target Model

Table 4. Qwen3-1.7B Results. Average acceptance length on MT-Bench, GSM8K, MATH-500, and SVAMP for HASS and EAGLE-2 at temperatures 0 and 1. Higher is better.

		Temperature 0					Temperature 1				
Model Variant	Method	MT-Bench	GSM8K	MATH-500	SVAMP	Avg.	MT-Bench	GSM8K	MATH-500	SVAMP	Avg.
RQ1. Task-specific training: single-domain checkpoints											
MathInstruct	HASS	1.90	2.55	2.63	2.40	2.37	1.72	2.42	2.46	2.25	2.21
MathInstruct	EAGLE-2	1.85	2.42	2.48	2.36	2.28	1.68	2.30	2.32	2.20	2.13
ShareGPT	HASS	2.28	2.18	2.08	2.16	2.18	2.14	2.06	1.94	2.02	2.04
ShareGPT	EAGLE-2	2.17	2.13	2.05	2.10	2.11	2.04	2.01	1.90	1.96	1.98
RQ2. Mixed-data training: robustness checkpoints											
Mixed 35k+35k	HASS	2.30	2.43	2.48	2.44	2.41	2.12	2.32	2.29	2.36	2.27
Mixed 35k+35k	EAGLE-2	2.20	2.32	2.36	2.35	2.31	2.02	2.22	2.16	2.24	2.16
Mixed 70k+70k	HASS	2.42	2.72	2.80	2.74	2.67	1.98	2.18	2.02	2.12	2.08
Mixed 70k+70k	EAGLE-2	2.32	2.58	2.65	2.60	2.54	1.90	2.06	1.92	1.98	1.97
RQ3. Combining specialists: weight averaging vs. inference-time composition											
Averaged	HASS	1.62	1.70	1.75	1.68	1.69	1.50	1.62	1.60	1.58	1.58
Averaged	EAGLE-2	1.55	1.63	1.66	1.61	1.61	1.44	1.55	1.52	1.50	1.50
Confidence Routed	HASS	2.34	2.62	2.70	2.64	2.58	2.18	2.48	2.45	2.46	2.39
Confidence Routed	EAGLE-2	2.25	2.50	2.60	2.52	2.47	2.08	2.36	2.34	2.34	2.28
Merged Trees	HASS	2.48	2.80	2.90	2.82	2.75	2.32	2.65	2.62	2.63	2.56
Merged Trees	EAGLE-2	2.38	2.68	2.78	2.70	2.64	2.22	2.52	2.50	2.50	2.44

Table 5. Qwen3-4B Results. Average acceptance length on MT-Bench, GSM8K, MATH-500, and SVAMP for HASS and EAGLE-2 at temperatures 0 and 1. Higher is better.

		Temperature 0					Temperature 1				
Model Variant	Method	MT-Bench	GSM8K	MATH-500	SVAMP	Avg.	MT-Bench	GSM8K	MATH-500	SVAMP	Avg.
RQ1. Task-specific training: single-domain checkpoints											
MathInstruct	HASS	1.82	2.48	2.56	2.34	2.30	1.65	2.35	2.39	2.19	2.15
MathInstruct	EAGLE-2	1.78	2.36	2.42	2.30	2.22	1.61	2.24	2.26	2.14	2.06
ShareGPT	HASS	2.18	2.12	2.03	2.11	2.11	2.05	2.00	1.89	1.97	1.98
ShareGPT	EAGLE-2	2.07	2.07	2.00	2.05	2.05	1.95	1.96	1.85	1.91	1.92
RQ2. Mixed-data training: robustness checkpoints											
Mixed 35k+35k	HASS	2.20	2.36	2.42	2.38	2.34	2.03	2.25	2.23	2.30	2.20
Mixed 35k+35k	EAGLE-2	2.10	2.26	2.30	2.29	2.24	1.94	2.16	2.10	2.18	2.10
Mixed 70k+70k	HASS	2.32	2.65	2.73	2.67	2.59	1.90	2.12	1.97	2.06	2.01
Mixed 70k+70k	EAGLE-2	2.22	2.51	2.58	2.54	2.46	1.82	2.00	1.87	1.93	1.91
RQ3. Combining specialists: weight averaging vs. inference-time composition											
Averaged	HASS	1.55	1.65	1.70	1.64	1.64	1.44	1.57	1.56	1.54	1.53
Averaged	EAGLE-2	1.48	1.59	1.62	1.57	1.57	1.38	1.51	1.48	1.46	1.46
Confidence Routed	HASS	2.24	2.55	2.63	2.58	2.50	2.09	2.42	2.39	2.40	2.33
Confidence Routed	EAGLE-2	2.15	2.44	2.53	2.46	2.40	2.00	2.30	2.28	2.28	2.22
Merged Trees	HASS	2.38	2.73	2.83	2.75	2.67	2.23	2.58	2.55	2.56	2.48
Merged Trees	EAGLE-2	2.28	2.61	2.71	2.63	2.56	2.13	2.45	2.43	2.44	2.36

Table 6. Vicuna-13B v1.3 Results. Average acceptance length on MT-Bench, GSM8K, MATH-500, and SVAMP for HASS and EAGLE-2 at temperatures 0 and 1. Higher is better.

		Temperature 0					Temperature 1				
Model Variant	Method	MT-Bench	GSM8K	MATH-500	SVAMP	Avg.	MT-Bench	GSM8K	MATH-500	SVAMP	Avg.
RQ1. Task-specific training: single-domain checkpoints											
MathInstruct	HASS	3.90	5.50	5.70	5.45	5.14	3.35	5.15	5.25	5.05	4.70
MathInstruct	EAGLE-2	3.65	5.20	5.45	5.25	4.89	3.15	4.90	5.05	4.82	4.48
ShareGPT	HASS	5.20	5.05	4.75	4.85	4.96	4.65	4.60	4.35	4.45	4.51
ShareGPT	EAGLE-2	4.83	4.79	4.55	4.65	4.71	4.40	4.41	4.20	4.30	4.33
RQ2. Mixed-data training: robustness checkpoints											
Mixed 35k+35k	HASS	5.15	5.30	5.45	5.35	5.31	4.95	5.12	5.05	5.10	5.06
Mixed 35k+35k	EAGLE-2	4.82	5.05	5.15	5.10	5.03	4.55	4.85	4.75	4.80	4.74
Mixed 70k+70k	HASS	5.35	5.75	5.90	5.80	5.70	4.35	4.70	4.45	4.60	4.53
Mixed 70k+70k	EAGLE-2	5.05	5.45	5.60	5.50	5.40	4.10	4.40	4.15	4.25	4.23
RQ3. Combining specialists: weight averaging vs. inference-time composition											
Averaged	HASS	3.05	3.35	3.50	3.25	3.29	2.85	3.20	3.15	3.10	3.08
Averaged	EAGLE-2	2.80	3.05	3.20	3.00	3.01	2.62	2.90	2.85	2.80	2.79
Confidence Routed	HASS	5.35	5.55	5.75	5.60	5.56	5.05	5.32	5.25	5.20	5.21
Confidence Routed	EAGLE-2	5.05	5.30	5.55	5.35	5.31	4.75	5.05	4.95	4.92	4.92
Merged Trees	HASS	5.65	5.88	6.00	5.90	5.86	5.35	5.65	5.55	5.50	5.51
Merged Trees	EAGLE-2	5.35	5.65	5.85	5.70	5.64	5.05	5.35	5.25	5.22	5.22

Figure 9. **Verifier Scaling Summary.** Aggregate acceptance length across target verifiers for the main composition categories, separated by HASS and EAGLE-2 and by decoding temperature.

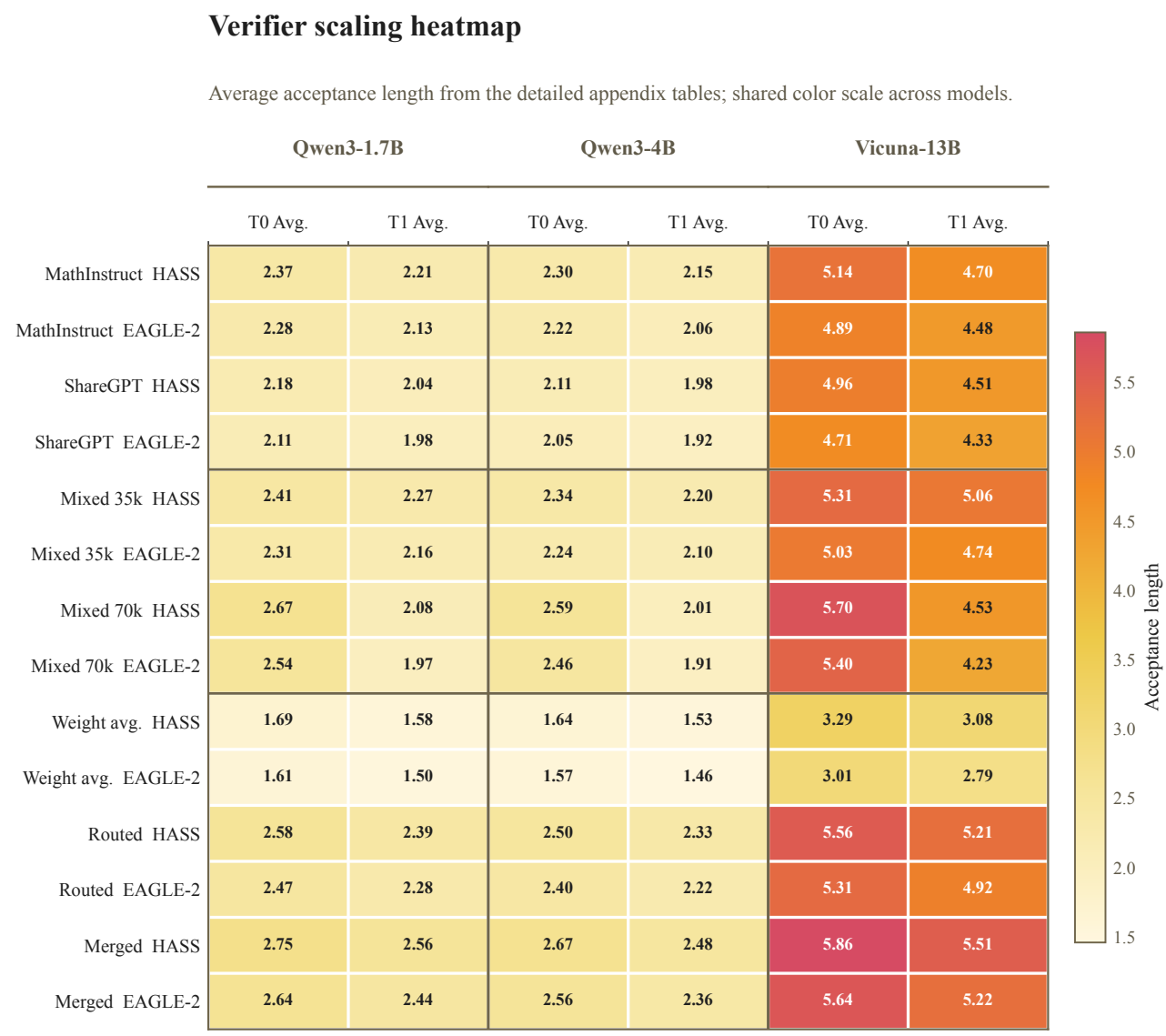


Figure 10. **Verifier Scaling Heatmap.** Average-column heatmap of the detailed Qwen3-1.7B, Qwen3-4B, and Vicuna-13B v1.3 appendix tables. Columns show temperature-0 and temperature-1 averages for each verifier, and darker cells indicate higher acceptance length under a shared color scale.

A.2. EAGLE-2 Entropy (Temperature 0)

Checkpoint	Benchmark	Draft Acc.	Draft Rej.	Δ Draft	Verif. Acc.	Verif. Rej.	Δ Verif.
Averaged	GSM8K	8.0257	9.2397	+1.2141	0.1567	0.2698	+0.1131
Averaged	MATH-500	8.8405	9.8344	+0.9938	0.2013	0.3787	+0.1774
Averaged	MT-Bench	8.5128	9.4886	+0.9758	0.2427	0.5796	+0.3368
Averaged	SVAMP	7.8233	9.0891	+1.2658	0.1703	0.2833	+0.1131
MathInstruct	GSM8K	0.5284	1.0756	+0.5473	0.1500	0.4246	+0.2746
MathInstruct	MATH-500	0.4567	0.8555	+0.3988	0.1984	0.5067	+0.3083
MathInstruct	MT-Bench	2.3867	3.8516	+1.4649	0.2341	0.6212	+0.3871
MathInstruct	SVAMP	0.5928	1.1607	+0.5679	0.1639	0.4111	+0.2473
Mixed	GSM8K	0.7074	1.2153	+0.5079	0.1525	0.4671	+0.3146
Mixed	MATH-500	0.6302	1.0409	+0.4107	0.1925	0.5430	+0.3505
Mixed	MT-Bench	1.1839	1.8407	+0.6568	0.2561	0.6736	+0.4175
Mixed	SVAMP	0.8162	1.3367	+0.5205	0.1717	0.4621	+0.2904
ShareGPT	GSM8K	0.7434	1.0952	+0.3518	0.1500	0.5075	+0.3574
ShareGPT	MATH-500	0.8926	1.2558	+0.3632	0.1898	0.5766	+0.3868
ShareGPT	MT-Bench	1.0404	1.6292	+0.5887	0.2539	0.6600	+0.4061
ShareGPT	SVAMP	0.8554	1.2391	+0.3837	0.1667	0.5006	+0.3339

Table 7. **EAGLE-2 Entropy at Temperature 0.** Each row reports accepted-token and rejected-token entropy for one benchmark. Positive Δ means higher entropy for rejected tokens.

A.3. HASS Entropy (Temperature 0)

Checkpoint	Benchmark	Draft Acc.	Draft Rej.	Δ Draft	Verif. Acc.	Verif. Rej.	Δ Verif.
MathInstruct	GSM8K	1.0731	1.4475	+0.3744	0.1613	0.5508	+0.3895
MathInstruct	MATH-500	2.0806	2.1836	+0.1030	0.1928	0.6625	+0.4698
MathInstruct	MT-Bench	4.4777	5.2238	+0.7461	0.2358	1.3139	+1.0780
MathInstruct	SVAMP	1.3927	1.6778	+0.2851	0.1765	0.5660	+0.3896
ShareGPT	GSM8K	1.0690	1.7551	+0.6861	0.1549	0.5937	+0.4388
ShareGPT	MATH-500	2.2460	2.3454	+0.0995	0.1814	0.7575	+0.5761
ShareGPT	MT-Bench	1.9162	2.2477	+0.3315	0.2796	0.8882	+0.6086
ShareGPT	SVAMP	1.3445	2.0791	+0.7345	0.1695	0.6139	+0.4444
Mixed	GSM8K	1.2369	1.7589	+0.5220	0.1597	0.5287	+0.3690
Mixed	MATH-500	1.9626	2.2128	+0.2502	0.1876	0.6563	+0.4687
Mixed	MT-Bench	2.0482	2.4916	+0.4433	0.2766	0.9013	+0.6247
Mixed	SVAMP	1.3306	1.8352	+0.5047	0.1742	0.5501	+0.3759
Averaged	GSM8K	2.6208	3.2743	+0.6535	0.1610	0.8306	+0.6696
Averaged	MATH-500	3.8384	3.3019	−0.5364	0.1786	0.9489	+0.7703
Averaged	MT-Bench	4.2061	3.5950	−0.6110	0.2362	1.4447	+1.2085
Averaged	SVAMP	2.5409	3.1844	+0.6434	0.1739	0.8523	+0.6784

Table 8. **HASS Entropy at Temperature 0.** Each row reports accepted-token and rejected-token entropy for one benchmark. Positive Δ means higher entropy for rejected tokens.

A.4. Tree Merge Utility

```
def _merge_trees(
    draft_tokens1, retrieve_indices1, tree_mask1, tree_pos1,
    draft_tokens2, retrieve_indices2, tree_mask2, tree_pos2,
):
    n1 = draft_tokens1.shape[1] - 1
    n2 = draft_tokens2.shape[1] - 1
    N = n1 + n2 + 1
    device = draft_tokens1.device
    dtype = tree_mask1.dtype

    merged_draft = torch.cat(
        [draft_tokens1, draft_tokens2[0, 1:][None]], dim=1)

    merged_mask = torch.zeros(N, N, device=device, dtype=dtype)
    merged_mask[0, 0] = 1.0
    merged_mask[1:n1 + 1, :n1 + 1] = tree_mask1[0, 0, 1:, :]
    merged_mask[n1 + 1:, 0] = 1.0
```

```
    merged_mask[n1 + 1:, n1 + 1:] = tree_mask2[0, 0, 1:, 1:]
    merged_mask = merged_mask[None, None]

    merged_pos = torch.cat([tree_pos1, tree_pos2[1:]])

    d1, d2 = retrieve_indices1.shape[1], retrieve_indices2.shape[1]
    max_d = max(d1, d2)
    ri1 = F.pad(retrieve_indices1, (0, max_d - d1), value=-1)
    ri2 = retrieve_indices2.clone()
    ri2[ri2 > 0] += n1
    ri2 = F.pad(ri2, (0, max_d - d2), value=-1)
    merged_retrieve = torch.cat([ri1, ri2], dim=0)

    return (merged_draft, merged_retrieve,
            merged_mask, merged_pos)
```

A.5. Correctness of routing and merged-tree verification

Let $Q(\cdot \mid y_{1:t})$ denote the target model's continuation distribution from prefix $y_{1:t}$. For any (packed) draft tree $\mathcal{T}$ rooted at $y_{1:t}$, let $\mathrm{Dec}(y_{1:t}; \mathcal{T})$ denote the random continuation produced by running one verifier call on $\mathcal{T}$ and then continuing with the standard speculative procedure.

Assumption A.1 (Lossless verification for a fixed valid tree). A packed tree $\mathcal{T}$ is *valid* if the verifier pass on $\mathcal{T}$ produces, for every node, the same target-side conditionals $q(\cdot \mid \text{its path-prefix})$ that the target model would produce under standalone autoregressive evaluation along that node's path. For every valid $\mathcal{T}$ and every measurable set of continuations B,

$$\Pr\big(\mathrm{Dec}(y_{1:t}; \mathcal{T}) \in B \,\big|\, y_{1:t}, \mathcal{T}\big) \;=\; Q(B \mid y_{1:t}).$$

(This is the standard lossless speculative-decoding guarantee used throughout the paper.)

Lemma A.1 (Mixtures over valid trees remain lossless). Let $\mathcal{T}$ be any *random* valid tree (possibly generated by any draft model(s)). Then for every set B,

$$\Pr\big(\mathrm{Dec}(y_{1:t}; \mathcal{T}) \in B \,\big|\, y_{1:t}\big) \;=\; Q(B \mid y_{1:t}).$$

Proof. By the tower property,

$$\Pr(\mathrm{Dec} \in B \mid y_{1:t}) = \mathbb{E}[\Pr(\mathrm{Dec} \in B \mid y_{1:t}, \mathcal{T}) \mid y_{1:t}] = \mathbb{E}[Q(B \mid y_{1:t}) \mid y_{1:t}] = Q(B \mid y_{1:t}).$$

Proposition A.1 (Correctness of routing). Let $\mathcal{T}_{\text{math}}$ and $\mathcal{T}_{\text{chat}}$ be two valid draft trees generated from the same prefix $y_{1:t}$. Let g be any (possibly randomized) routing rule that depends only on draft-side quantities available *before* verification (e.g., confidences/entropies/tree statistics), and define the selected tree $\mathcal{T}^\star = \mathcal{T}_{g(y_{1:t}, \mathcal{T}_{\text{math}}, \mathcal{T}_{\text{chat}})}$. Then routing is distribution-preserving:

$$\Pr\big(\mathrm{Dec}(y_{1:t}; \mathcal{T}^\star) \in B \,\big|\, y_{1:t}\big) \;=\; Q(B \mid y_{1:t}) \quad \text{for all } B.$$

Proof. $\mathcal{T}^\star$ is a random valid tree (a draft-side function of $(\mathcal{T}_{\text{math}}, \mathcal{T}_{\text{chat}})$), so the claim follows immediately from Lemma A.1.

Lemma A.2 (Verifier invariance under masked concatenation). Let the verifier be any transformer-style model that computes per-token logits from (token ids, position ids, attention mask). Consider two packed verifier inputs (X, M, P) and (X', M', P') with a shared index set S such that: (i) $X|_S = X'|_S$ and $P|_S = P'|_S$; (ii) $M|_{S\times S} = M'|_{S\times S}$; and (iii) tokens in S do not attend outside S in either input, i.e. for all $i \in S$ and $j \notin S$, $M_{ij} = M'_{ij} = 0$. Then the verifier logits on indices in S are identical under the two packed inputs.

Proof. Induct over transformer layers. At layer 0, hidden states on S match because token embeddings and position encodings match. Assume hidden states on S match at layer $\ell - 1$. At layer ℓ, each token $i \in S$ attends only to tokens j with $M_{ij} = 1$, and by (iii) all such j lie in S. By (ii) the mask on $S \times S$ matches, and by the inductive hypothesis the keys/values of all visible $j \in S$ match. Therefore the attention output for each $i \in S$ matches; the remaining sublayers are pointwise with shared parameters, so hidden states on S match at layer ℓ. Hence the final logits on S match.

Proposition A.2 (Correctness of merged-tree verification). Let $\mathcal{T}_{\text{math}}$ and $\mathcal{T}_{\text{chat}}$ be valid trees from prefix $y_{1:t}$, each with its own packed representation (tokens, tree attention mask, and depth-based position ids) used for standalone tree verification. Construct the merged tree $\mathcal{T}_\cup$ by (a) sharing the root, (b) concatenating the non-root nodes of both trees, (c) using an attention mask that preserves each subtree's ancestry relations and *masks all cross-subtree attention*, and (d) assigning each node the same depth-based position id it had in its source tree.

Then (i) every node in the merged verifier pass receives exactly the same target-side conditional distribution as in standalone verification of its source subtree, and consequently (ii) merged-tree verification is distribution-preserving:

$$\Pr\big(\mathrm{Dec}(y_{1:t}; \mathcal{T}_\cup) \in B \,\big|\, y_{1:t}\big) \;=\; Q(B \mid y_{1:t}) \quad \text{for all } B.$$

Proof. Fix $s \in \{\text{math}, \text{chat}\}$ and let S_s denote the index set of the shared root together with all nodes coming from subtree s inside the merged packing. By construction, the merged packed input agrees with the standalone packed input on S_s (tokens, depth-based positions, and within-subtree attention), and nodes in S_s do not attend to nodes outside S_s because all cross-subtree attention is masked. Therefore, by Lemma A.2, the verifier logits (hence $q(\cdot \mid \cdot)$) on all nodes in subtree s are identical to standalone verification. This holds for both subtrees, so $\mathcal{T}_\cup$ is a valid tree in the sense of Assumption A.1.

Applying Assumption A.1 to the fixed valid tree $\mathcal{T}_\cup$ yields $\Pr(\mathrm{Dec}(y_{1:t}; \mathcal{T}_\cup) \in B \mid y_{1:t}, \mathcal{T}_\cup) = Q(B \mid y_{1:t})$ for all B. Unconditioning (or equivalently applying Lemma A.1) gives the claimed distribution preservation.

Corollary A.1. Both routing (Proposition A.1) and merged-tree verification (Proposition A.2) preserve the target-model output distribution. They may change proposal quality, acceptance length, and runtime, but not the verifier's sampling law.

A.6. EAGLE-2 Acceptance Rates by Depth

A.6.1. Temperature 0

Depth	Variant	MT-Bench	GSM8K	MATH-500	SVAMP	Avg
1	MathInstruct	72.0%	96.1%	97.9%	94.9%	90.2%
1	ShareGPT	89.9%	93.1%	90.6%	91.9%	91.4%
1	Mixed 35k+35k	88.7%	95.3%	96.4%	94.3%	93.7%
1	Averaged	71.3%	88.3%	89.3%	87.0%	84.0%
1	Mixed 70k+70k	80.0%	80.8%	86.6%	80.7%	82.0%
2	MathInstruct	52.6%	88.8%	92.5%	86.8%	80.2%
2	ShareGPT	74.6%	78.3%	73.7%	76.3%	75.7%
2	Mixed 35k+35k	70.8%	84.2%	85.7%	82.4%	80.8%
2	Averaged	42.0%	62.2%	63.1%	60.2%	56.9%
2	Mixed 70k+70k	61.0%	61.8%	68.6%	60.1%	62.9%
3	MathInstruct	48.6%	84.6%	87.3%	83.8%	76.1%
3	ShareGPT	66.2%	67.3%	66.9%	68.1%	67.1%
3	Mixed 35k+35k	60.2%	74.9%	76.3%	75.6%	71.8%
3	Averaged	17.6%	30.2%	30.1%	31.4%	27.3%
3	Mixed 70k+70k	52.2%	50.1%	60.0%	53.9%	54.0%
4	MathInstruct	52.8%	78.6%	82.5%	77.9%	72.9%
4	ShareGPT	64.1%	62.0%	64.9%	63.3%	63.6%
4	Mixed 35k+35k	58.3%	65.7%	73.4%	65.8%	65.8%
4	Averaged	19.3%	22.5%	28.0%	27.5%	24.3%
4	Mixed 70k+70k	52.9%	47.2%	57.0%	46.1%	50.8%
5	MathInstruct	61.0%	76.4%	81.5%	75.0%	73.5%
5	ShareGPT	61.5%	57.5%	65.5%	58.8%	60.8%
5	Mixed 35k+35k	57.2%	61.9%	70.6%	63.1%	63.2%
5	Averaged	39.0%	25.7%	32.2%	23.1%	30.0%
5	Mixed 70k+70k	53.3%	43.5%	58.0%	42.2%	49.3%

Table 9. **EAGLE-2 Acceptance by Depth at Temperature 0.** Higher rows correspond to shallower draft positions in the speculative tree.

A.6.2. Temperature 1

Depth	Variant	MT-Bench	GSM8K	MATH-500	SVAMP	Avg
1	MathInstruct	68.5%	94.1%	91.5%	93.1%	86.8%
1	ShareGPT	85.7%	91.7%	83.7%	90.7%	88.0%
1	Mixed 35k+35k	85.1%	93.8%	89.9%	93.3%	90.5%
1	Averaged	67.6%	86.1%	81.2%	84.0%	79.7%
1	Mixed 70k+70k	75.5%	76.9%	75.6%	78.1%	76.5%
2	MathInstruct	50.1%	86.3%	82.0%	84.7%	75.8%
2	ShareGPT	68.7%	76.4%	65.1%	73.9%	71.0%
2	Mixed 35k+35k	65.7%	81.6%	75.9%	80.7%	76.0%
2	Averaged	41.0%	60.9%	55.6%	59.0%	54.1%
2	Mixed 70k+70k	56.5%	58.0%	58.5%	58.4%	57.8%
3	MathInstruct	45.5%	82.0%	76.8%	81.3%	71.4%
3	ShareGPT	60.9%	65.7%	57.7%	66.2%	62.6%
3	Mixed 35k+35k	57.3%	72.7%	67.2%	72.6%	67.5%
3	Averaged	17.7%	28.1%	24.5%	29.5%	24.9%
3	Mixed 70k+70k	48.9%	46.7%	50.6%	50.6%	49.2%
4	MathInstruct	52.2%	77.4%	74.6%	76.3%	70.1%
4	ShareGPT	59.7%	60.6%	58.0%	60.8%	59.8%
4	Mixed 35k+35k	56.4%	65.2%	65.4%	65.6%	63.1%
4	Averaged	17.9%	21.9%	25.2%	28.6%	23.4%
4	Mixed 70k+70k	49.9%	45.5%	51.8%	43.2%	47.6%
5	MathInstruct	56.5%	75.2%	75.2%	73.7%	70.1%
5	ShareGPT	58.4%	56.2%	58.2%	58.9%	57.9%
5	Mixed 35k+35k	57.0%	60.0%	63.7%	60.1%	60.2%
5	Averaged	30.4%	26.8%	21.8%	25.7%	26.2%
5	Mixed 70k+70k	52.5%	42.1%	52.0%	41.6%	47.0%

Table 10. **EAGLE-2 Acceptance by Depth at Temperature 1.** Higher rows correspond to shallower draft positions in the speculative tree.

A.7. HASS Acceptance Rates by Depth

A.7.1. TEMPERATURE 0

Depth	Variant	MT-Bench	GSM8K	MATH-500	SVAMP	Avg
1	MathInstruct	61.6%	94.6%	95.9%	94.8%	86.7%
1	ShareGPT	84.5%	90.9%	88.0%	89.4%	88.2%
1	Mixed 35k+35k	84.9%	95.2%	96.1%	94.7%	92.7%
1	Averaged	68.1%	87.5%	90.9%	87.4%	83.5%
1	Mixed 70k+70k	82.3%	85.1%	90.6%	86.2%	86.0%
2	MathInstruct	50.9%	84.8%	88.9%	82.3%	76.7%
2	ShareGPT	73.7%	78.6%	71.5%	75.3%	74.8%
2	Mixed 35k+35k	74.2%	84.0%	87.2%	81.0%	81.6%
2	Averaged	48.2%	68.0%	72.9%	63.3%	63.1%
2	Mixed 70k+70k	67.6%	66.8%	75.3%	66.8%	69.1%
3	MathInstruct	41.6%	77.9%	81.8%	76.1%	69.4%
3	ShareGPT	56.8%	63.1%	51.1%	61.7%	58.2%
3	Mixed 35k+35k	58.3%	71.6%	71.8%	69.7%	67.9%
3	Averaged	27.0%	45.3%	44.5%	41.8%	39.6%
3	Mixed 70k+70k	51.5%	55.0%	60.3%	53.1%	55.0%
4	MathInstruct	51.1%	73.8%	78.8%	73.5%	69.3%
4	ShareGPT	53.4%	50.6%	44.4%	48.6%	49.2%
4	Mixed 35k+35k	53.8%	61.9%	63.9%	62.0%	60.4%
4	Averaged	32.5%	33.4%	33.1%	30.7%	32.4%
4	Mixed 70k+70k	48.7%	47.5%	54.4%	42.2%	48.2%
5	MathInstruct	62.1%	70.5%	75.9%	66.4%	68.7%
5	ShareGPT	52.2%	44.2%	44.5%	41.3%	45.5%
5	Mixed 35k+35k	54.2%	55.2%	59.2%	51.2%	54.9%
5	Averaged	30.3%	26.0%	29.1%	22.8%	27.0%
5	Mixed 70k+70k	50.2%	44.7%	53.5%	36.1%	46.1%

Table 11. **HASS Acceptance by Depth at Temperature 0.** Higher rows correspond to shallower draft positions in the speculative tree.