DRAFT CALIBRATION FOR SPECULATIVE DECODING: ACCELERATING LLM SERVING BY MATCHING DRAFT AND TARGET DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model (LLM) inference by employing a lightweight draft model to propose draft tokens and allowing the target model to verify the new tokens in parallel to improve inference-time compute utilization. The speedup relies on the acceptance probability of draft tokens in target verification, which is directly related to the gap $\|p - q\|_1$ between the target distribution p and the draft distribution q. In this paper, we study draft calibration of speculative decoding to improve the speedup of speculative decoding, where we employ post-hoc methods to calibrate the draft distribution to mitigate the distribution gap. We observe that though current community open-source draft models may learn the top candidate token correctly, they are generally underconfident compared to the target model, suggesting a consistent distribution mismatch. To mitigate the effect of this mismatch on the acceptance rate of the draft models, we try various post-hoc calibration methods, including standard temperature scaling and token-wise calibration. Our results suggest that a standard temperature scaling applied to the target distribution improves the acceptance length by approximately 10% at no additional cost across different settings. Analysis of more general calibration methods reveals further potential room for improvement.

1 INTRODUCTION

Current state-of-the-art large language models (LLMs) generate text in an autoregressive way, which runs one new forward pass to generate one next token recursively. This nature of LLM serving results in an insufficient use of parallelism in modern computational resources like GPUs. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) is considered a valid solution to mitigate this fundamental serving inefficiency: at each inference time, a lightweight draft model first proposes a draft block that contains several draft tokens in a row, and the target model (the serving model) verifies all the draft tokens through one forward pass. Thanks to the rejection sampling design at the verification stage, speculative decoding perfectly preserves the target model distribution while accelerating inference by lowering the number of target model forward passes to improve compute utilization and ultimately serving efficiency.

The speedup achieved by speculative decoding fundamentally relies on the acceptance rate of each draft token, i.e., the probability that a draft token can be accepted by the target model verification, which has a closed-form representation $1 - \frac{1}{2}\|p - q\|_1$ on target distribution p and draft distribution q (Leviathan et al., 2023; Chen et al., 2023). Therefore, a clear target to improve speculative decoding is that we not only desire the draft model to learn top candidates for each token well, but also to minimize this ℓ_1 distribution gap. This formula motivates a connection to confidence calibration (Guo et al., 2017; Wang, 2023), where predicted probabilities should match empirical correctness frequencies rather than merely identify the top candidate class. In speculative decoding, the formulation can be more straightforward: since the goal is to improve the acceptance rate, we consider the target distribution as the ground truth, and calibrate the draft distribution to mitigate this ℓ_1 distribution gap and thus improve the acceptance rate. From this perspective, we may improve speculative decoding not only in training time through algorithm and architecture modifications (Cai et al., 2024; Li et al., 2024b;c; 2025; Chen et al., 2026; Cheng et al., 2026), but also after training by calibrating the draft distribution better, which we study systematically across draft architectures.

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.

In this paper, we study this draft calibration idea of speculative decoding, mainly on post-hoc methods adjusting the draft distribution q to mitigate the ℓ_1 distribution gap at inference time when using chain draft generation. In existing literature and community speculative decoding serving frameworks, one usually shares the inference settings as the target model without any modification when chain generating draft tokens. However, as we observe across community open-source draft models on different tasks, this choice may not be optimal because of the existence of a systematic mismatch between the draft and target models under the original setup. More specifically, the draft models tend to be underconfident compared to the target model when using the same generation settings, even though they have learned the top candidates correctly, revealing a consistent distribution gap.

To mitigate this gap and improve the speculative decoding speedup, we consider multiple calibration methods applied to the draft distribution, and show that a standard temperature scaling (Guo et al., 2017) for the draft distribution can significantly mitigate the gap and improve the overall accepted length by up to around 10% without any additional cost. We also further study Dirichlet scaling (Kull et al., 2019), a generalization of temperature scaling, and token-adaptive calibration methods, which apply more fine-grained adjustments to the draft distribution. Analysis of the optimal choices for these methods reveals further potential room of improvement with draft calibration, but currently the tried post-hoc methods still cannot capture it.

Our key contributions include:

- We present the underconfident tendency of draft models compared to target models, forming a consistent distribution mismatch that affects acceptance rate.
- We study the problem from the draft calibration perspective and apply different calibration methods to mitigate the systematic distribution gap, showing that standard temperature scaling can already significantly improve the acceptance rate across different settings.
- We further examine more general calibration methods and reveal that there is room for improvements with token-adaptive calibration methods through a trajectory replay analysis framework, though the current pure post-hoc methods may not be able to capture it.

2 Backgrounds and Understanding of Draft Calibration

Speculative decoding. To accelerate the inference of a large model, referred to as the target model, speculative decoding (Leviathan et al., 2023; Chen et al., 2023) incorporates a lightweight draft model and follows a draft-then-verify pipeline. At inference time, given a prefix sequence $x_1, \ldots, x_t$, we let the draft model generate K draft tokens in a row, then feed all the draft tokens to the target model to run a forward pass to verify them. The verification process goes one by one, i.e., if one draft token is not accepted by the verification, we abandon all of the subsequent draft tokens. For each draft token x_t at position t with draft distribution q and target distribution p on vocabulary $\mathcal{V}$, we do a rejection sampling process: we accept the token if $p(x_t) \geq q(x_t)$, otherwise we reject it with probability $1 - \frac{p(x_t)}{q(x_t)}$ and resample a new token from an adjusted distribution $p'(x) = \text{normalize}(\max(0, p(x) - q(x)))$. We can compute the expected acceptance rate of each draft token from the sampling strategy:

$$\mathbb{P}(x_t \text{ is accepted}) = \sum_{x \in \mathcal{V}} \min(p(x), q(x)) = 1 - \frac{1}{2} \|p - q\|_1 . \tag{1}$$

Clearly, mitigating this ℓ_1 gap between draft and target models directly improves acceptance rate and the effectiveness of speculative decoding.

Confidence calibration and draft calibration. Confidence calibration has been an important field for deep learning and trustworthy AI (Guo et al., 2017). In a classification problem of feature label pairs (X, y), calibration measures how much a model's predicted probability matches the true correctness likelihood. More formally, a classifier model that maps the feature X to a probability simplex q is perfectly calibrated if $\mathbb{P}(y = y_i | q(y_i) = q_i) = q_i$ for class i. Post-hoc methods fit a calibration map on held-out data while keeping the predictor fixed, ranging from histogram binning and isotonic regression to temperature and Dirichlet scaling (Guo et al., 2017; Kull et al., 2019).

Given the goal of improving acceptance of draft tokens for speculative decoding, we can also consider it from the calibration perspective, where we try to mitigate the distribution gap between the

draft distribution and target distribution, and we term it **draft calibration** in this paper. Compared to the confidence calibration problem, the idea is more straightforward: we have the ground truth target distribution after verification for each draft token, and can simply adopt acceptance performance as the evaluation metric for draft calibration. In this paper, we will focus on post-hoc methods to help calibrate the draft distribution better, which preserves the lossless property of speculative decoding, keeping the target distribution unchanged as long as the calibrated distribution q is employed consistently in drafting and rejection sampling stages (Leviathan et al., 2023; Chen et al., 2023).

Relation to draft tree methods. We focus on chain drafting in this paper, motivated by large-scale serving, where we handle many concurrent requests that compete for verification compute. Draft trees increase candidate coverage by proposing multiple continuations at each draft position (Li et al., 2024c; 2026b), but their additional branches also introduce more verification costs. This tradeoff becomes more restrictive as batch size grows: Liu et al. (2026), for example, find that a 21-token tree becomes slower than autoregressive decoding at batch size 64 across their evaluated workloads, while the three-token chain retains a speedup. Likewise, the concurrent-serving study in Li et al. (2026b) also reports largely reduced tree node budgets to keep the effectiveness of the draft tree with a batch size of 32. The evidence suggests that the benefits of draft trees may be limited in large-scale serving, where the batch size can be up to hundreds. We therefore keep our focus on how much acceptance can be improved by calibrating the draft distribution.

3 Related Work

Speculative decoding. Early blockwise and speculative methods amortize verification over multiple candidate tokens (Stern et al., 2018; Xia et al., 2023), while speculative sampling pairs a lightweight drafter with parallel target verification to preserve the target distribution (Leviathan et al., 2023; Chen et al., 2023). Subsequent work improves draft efficiency and alignment through alternative backbones and distillation (Choi et al., 2025; Zhou et al., 2024; Zhang et al., 2025), and expands candidate coverage through tree-based verification (Miao et al., 2024; Li et al., 2024c; 2026b). Medusa, EAGLE, and related draft-head designs introduce additional decoding heads or a lightweight draft layer conditioned on target features instead of standalone draft models (Cai et al., 2024; Li et al., 2024b; Ankner et al., 2024), and the latter state-of-the-art methods follow this design, including self-speculation methods (Zhang et al., 2024; Liu et al., 2024) and EAGLE-3 (Li et al., 2025). ParallelSpec, P-EAGLE, DFlash, and related parallel drafters reduce sequential drafting overhead by predicting multiple positions in parallel, including through block-diffusion drafting (Xiao et al., 2024; Hui et al., 2026; An et al., 2026; Chen et al., 2026). Built upon the parallelism idea, DSpark augments a parallel backbone with lightweight autoregressive correction and confidence-based verification scheduling (Cheng et al., 2026).

Confidence calibration. Confidence calibration aligns predicted confidence with empirical correctness and has been studied through post-hoc maps and calibration-aware training (Guo et al., 2017; Kumar et al., 2018; Mukhoti et al., 2020; Wang, 2023). Post-hoc approaches include binning, parametric scaling, and their compositions (Naeini et al., 2015; Kull et al., 2017; 2019; Kumar et al., 2019; Zhang et al., 2020; Ding et al., 2021). More recent studies also examine token-adaptive temperature scaling for language models (Desai & Durrett, 2020; Xie et al., 2024).

Post-hoc draft calibration in speculative decoding. Draft calibration is largely unexplored in the literature, and prior works with temperature and distribution adjustment address several distinct objectives (Wang et al., 2025; Ouyang et al., 2024; Zhang et al., 2026; Berdoz et al., 2026; Zhou et al., 2026), with the following works as examples. SubSpec sharpens draft probabilities to improve cumulative-probability ranking in deep draft trees for offloaded inference with a greedy target, but does not systematically study the draft temperature scaling (Wang et al., 2025). Ouyang et al. (2024) study the interaction between the training data of the draft model and decoding temperatures, but does not consider differentiating the temperature of the draft model. Other work calibrates confidence for control decisions: SpecBound temperature-scales shallow-layer outputs for early exit, while DSpark calibrates predicted prefix acceptance for verification scheduling (Wen & Feng, 2026; Cheng et al., 2026). Differently, we focus on systematically studying the distribution gap between draft and target models and methods to mitigate the gap.

4 Underconfident Draft Models

We start from examining the current popular open-source draft models' distribution gap compared to the corresponding target models, which forms the basis of applying post-hoc draft calibration

methods. While Guo et al. (2017) discovers that modern neural networks tend to be overconfident in the choice they make, we find that draft models are generally underconfident compared to target models, admitting a consistently lower probability for top candidates. In other words, the draft distribution is generally flatter.

4.1 Experiment Settings & Metrics

Unless otherwise specified, we employ the following experiment settings for speculative decoding evaluation throughout the paper. We evaluate nine target–draft pairs using community open-source EAGLE-3 (Li et al., 2025), DFlash (Chen et al., 2026), and DSpark (Cheng et al., 2026) checkpoints for our evaluation. Target Models include GPT-OSS-120B, Gemma-4-31B-IT, and the Qwen series. Table 3 lists the exact pairings, checkpoint links, and chain lengths, which we generally follow the recommended settings. We use eight benchmarks spanning math (GSM8K, MATH-500, AIME 2025), code (HumanEval, MBPP, LiveCodeBench), and chat (MT-Bench, Arena-Hard v2), with sources and citations listed in Table 4. We use low-reasoning/no-thinking chat template in the experiments unless otherwise specified. The experiments are done on SpecForge (Li et al., 2026a)/SGLang (Zheng et al., 2024) stack with independent draft-temperature control and use chain proposals and rejection sampling with the calibrated draft probabilities. The temperature for the target model is fixed at $T_t = 1$, and generation is capped at 2,048 tokens. Each model pair keeps its chain length fixed across calibration methods. Please refer to more evaluation details in Appendix A.

We use the accepted length as our evaluation metric: output tokens committed per verification round, including the bonus token. We aggregate rounds within each task and report an equal-weight mean across tasks. Distribution diagnostics average over eligible draft positions, where all preceding proposals in the block were accepted.

4.2 Draft Models can learn top candidates well, but are underconfident

We use the same inference setup for draft and target models as default in this section.

Generally consistent top candidates. We start from examining how consistent the top candidates learned by the draft model are with the target model. In Figure 1, we show the overlap percentage between the top-k candidates of draft and target distributions regardless of order, i.e., how many of the top-k candidates are the same between target and draft models. We can observe that the draft model can learn around 70% or more for the top-1 token, showing its capability for catching the top candidate in the target distribution. The correctness tends to decrease, but it still remains more than 40% up to the top 16 candidates.

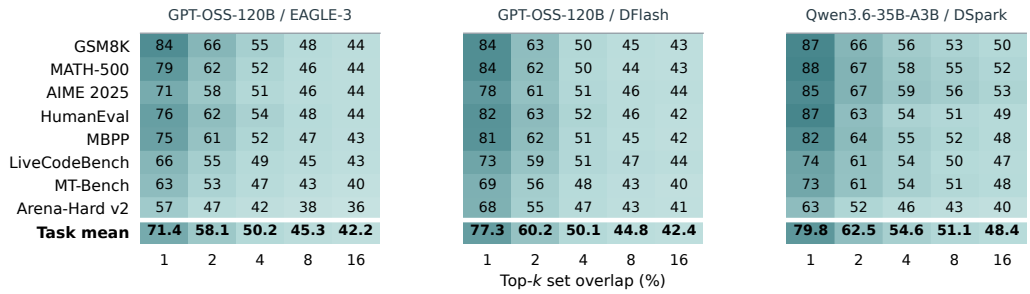

Figure 1: The overlap percentage of top-k candidates in target and draft distributions on three different draft models, where $k = 1, 2, 4, 8, 16$.

Underconfident draft model. While the top candidates are generally learned correctly, we observe a consistent gap between the draft and target distributions. Figure 2 shows the probability draft and target distributions give the top-1 candidate when they agree, and the entropy of the distributions. Across different tasks and state-of-the-art speculative decoding algorithms, there exists a consistent gap between the top-1 probability of draft and target distributions, even if the draft model predicts it correctly – the draft model is underconfident about its top candidate choice compared to the target distribution. The consistently higher entropy also provides further evidence that the draft distribution is generally flatter than the target distribution. To conclude, the draft models are overall underconfident compared to the target model.

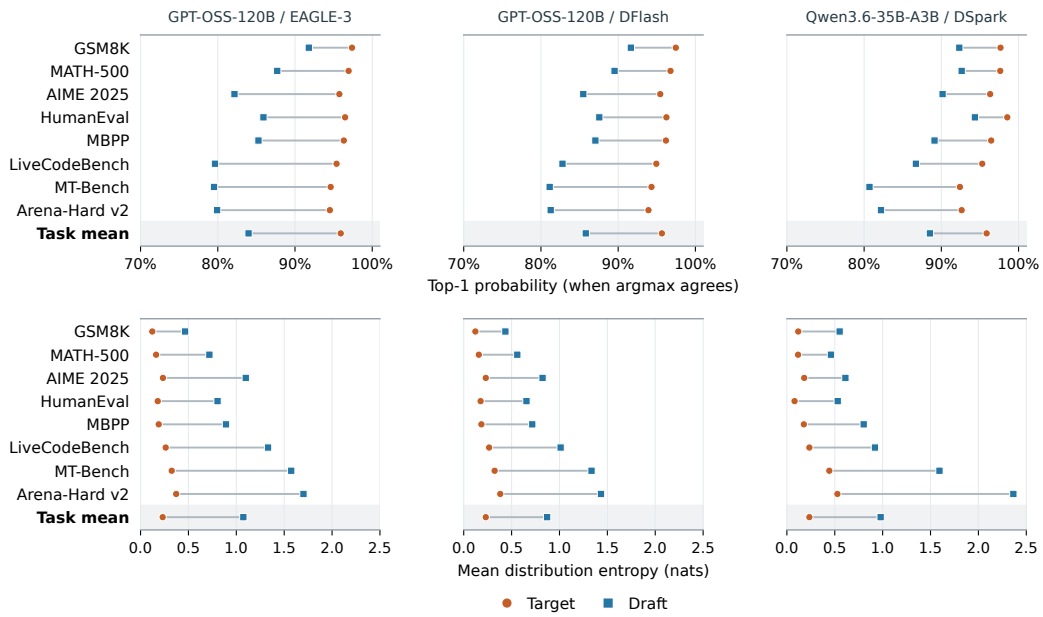

Figure 2: Top-1 candidate probability when the top candidate agrees (first row) and entropy (second row) of draft and target distributions on three different draft models.

Remark 1 (Why this distribution gap exists)**.** *This consistent distribution gap is actually surprising since the draft distribution should be trained to match the target distribution. We hypothesize that this may be related to the limited capacity of draft models, making it hard to learn a sharp distribution as the target model. The KL loss employed by the standard draft model training pipeline may also contribute to this, since it biases from the ℓ_1 distribution gap (Samarin et al., 2026).*

5 Draft Calibration Methods

Recall that the acceptance probability of a draft token can be computed by $A = 1 - \frac{1}{2}\|p - q\|_1$ that is directly related to the draft and target distribution gap. Observing the underconfident and flat draft distribution, it is natural to consider post-hoc calibration to mitigate its gap with the target distribution to improve the acceptance performance. In this section, we examine from standard general calibration methods to token-wise adaptive calibration, and show that the simple and popular temperature scaling already gives a significant improvement in accepted length.

5.1 Temperature Scaling

Temperature scaling (Guo et al., 2017) has become a standard default calibration and LLM inference technique. Given logits $z \in \mathbb{R}^V$ and temperature T, the corresponding probability distribution is:

$$p_{\text{cal}} = \text{softmax}\left(\frac{z}{T}\right), \text{ i.e., } [p_T]_i = \frac{\mathrm{e}^{z_i/T}}{\sum_{j=1}^{V} \mathrm{e}^{z_j/T}} \text{ for any } i \in [V].$$

From this formula, it is clear that lowering the temperature leads to a sharper distribution while keeping the rank of candidates the same. For speculative decoding, we actually have two temperature parameters: T_t for the target model that affects the quality of the generated text, which is considered fixed in the speculative decoding problem, and T_d for the draft model that only affects the draft distribution q, treated as an independent calibration parameter allowed to be different from T_t.

As presented in Section 4, the draft models are consistently underconfident with this default choice $T_d = T_t$ when $T_t = 1$. Since lowering T_d leads to higher top candidate probability and sharper draft temperature, our first trial for post-hoc calibration is tuning this T_d as a hyperparameter to check whether it helps mitigate the distribution gap, where we employ accepted length as the evaluation

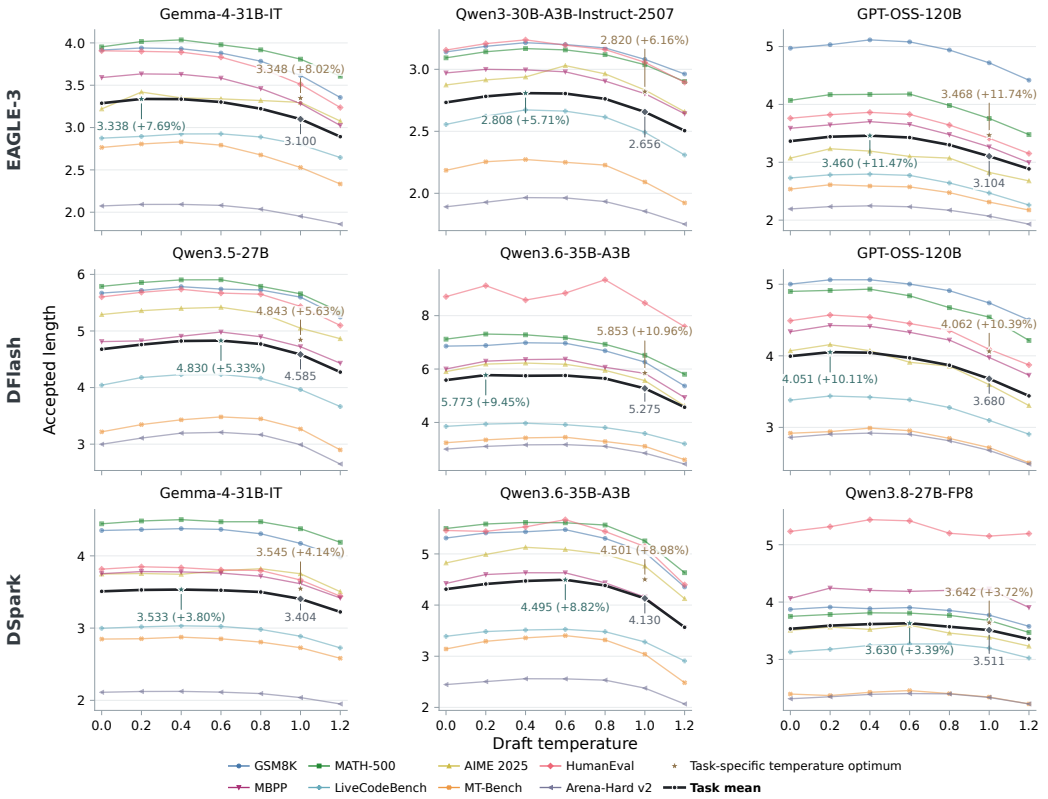

Figure 3: The accepted length across different models and speculative decoding algorithms when different draft temperatures are employed. The black line represents the average accepted length when the draft temperature is employed across all the tasks, and the bronze star means the optimal accepted length when we select the optimal temperature for each task. We specifically highlight their values and the baseline $T_d = 1$ for reference.

metric because of the direct relation between acceptance rate and distribution gap. The following experiments are conducted under the specified settings in Section 4.1.

Temperature scaling shows improvements across models and tasks. We apply temperature scaling to the draft model to adjust its distribution with a grid of $T_d \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2\}$ while keeping the temperature for the target model $T_t = 1$. Following the settings in Section 4.1, we are using no/low thinking mode for the models. We summarize the key observations on the results presented in Figure 3.

- Across all the considered settings, accepted length shows a clear peak when $T_d \in [0, 0.8]$. This is consistent with the observation that the draft models are generally underconfident, since $T_d < 1$ sharpens the draft distribution and achieves better performance.
- When looking at the curves in Figure 3, we can find that: 1) across all the settings, accepted length is approximately unimodal with respect to the draft temperature, making it easier to tune; 2) the curves are quite flat when T_d is near the optimal choice, implying that the current draft temperature grid should be sufficient for evaluating temperature scaling; 3) the trend of the curves for different tasks are relatively consistent for the same draft model, implying consistency in the distribution gap observed in Section 4 across domains, though the curves for different models may be more different.
- We can observe that temperature scaling improves the average accepted length by approximately 10% across all the settings for EAGLE-3 and DFlash, e.g., from 5.275 to 5.773 for Qwen3.6-35B-A3B with DFlash, when employing the same draft temperature across the models, the improvement consistently exists across different tasks. For DSpark, there is also a 3%-9% gain in average accepted length, with a generally larger T_d. Changing the draft temperature adds no model computation, so accepted-length gains imply proportional decoding-throughput gains when the time per draft–verify round remains unchanged.

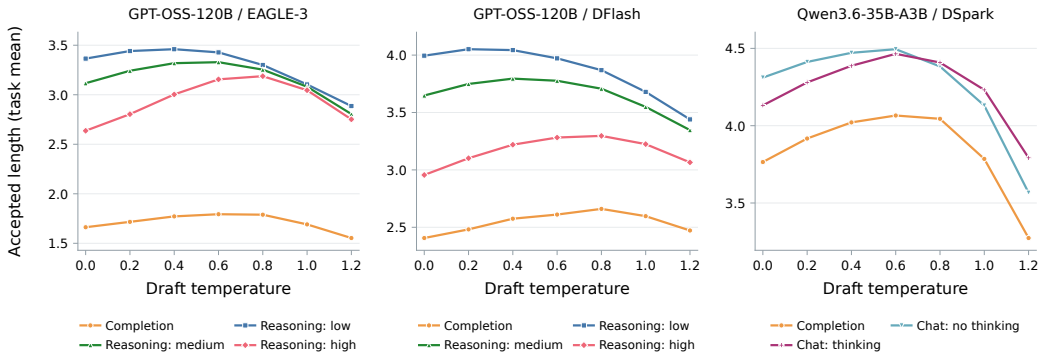

Figure 4: The accepted length averaged across the eight benchmarks with respect to draft temperature, with different chat templates.

- We would like to further emphasize that temperature scaling is powerful without the need for careful tuning, since the curves are generally flat across a large temperature range. When draft temperature is set to $T_d \in [0.4, 0.6]$, we can observe consistent gains compared to the standard choice $T_d = 1$ across all the models and tasks, which is generally near the best performance with temperature scaling. Task-specific draft temperature selection also gives only marginal improvement, suggesting that one global draft temperature already benefits multiple domains of use.

Reasoning effort (template) significantly affects speculative decoding. We further examine the performance of the draft models and the effect of temperature scaling when different evaluation templates are considered. Specifically, we evaluate the accepted length when serving with different reasoning effort as well as a completion mode, where we remove the chat template and let the model complete the context after the prompt. The results are presented in Figure 4. We can observe that the acceptance performance varies a lot across different reasoning efforts or evaluation templates, as also noticed in Eldenk et al. (2026). Consistent with the low reasoning case in Figure 3, the optimal draft temperature still biases from $T_d = 1$ across all the evaluation settings. The optimal draft temperature tends to increase mildly when the reasoning effort increases, which can be relevant to that the target distribution is itself generally flatter with a higher reasoning effort. Overall, temperature scaling benefits all the setups and generally gives more improvement in lower reasoning settings.

5.2 Dirichlet Scaling

As presented in the previous section, simple temperature scaling already shows significant improvement, which uses only one parameter to adjust the overall sharpness of the draft distribution. It is natural to consider whether there is room for even more gain with a stronger calibration parameter group. In this section, we examine a family of Dirichlet scaling (Kull et al., 2019) methods. Given this original probability distribution p, Dirichlet scaling seeks transformation parameters $W \in \mathbb{R}^{V \times V}$ and $b \in \mathbb{R}^V$ such that the adjusted distribution is

$$p_{\text{cal}} = \text{softmax}\left(W \log p + b\right). \tag{2}$$

We consider $b = 0$ in this paper. Dirichlet scaling is a more general calibration method class, recovering temperature scaling by taking $W = (1/T)I$. Beyond $W \propto I$, it provides more fine-grained scaling for each candidate token and allows candidate-interactive transformation with off-diagonal entries of W. Kull et al. (2019) presents evidence for improving confidence calibration compared to temperature scaling, especially for the classes that are not the top-1 candidate. This is beneficial for the regime of draft calibration, since the acceptance relies on the gap between the full distributions $\|p - q\|_1$. In LLM applications, it is generally infeasible to apply the scaling directly because of the large vocabulary size V. Since usually several top candidates take up the majority of probability, we take $W = \begin{bmatrix} W^{(K)} & \\ & (1/T_{\text{res}})I, \end{bmatrix}$ where $W^{(K)} \in \mathbb{R}^{K \times K}$ and apply the transformation to the top K candidates (with the largest probability) and a scalar T_{res} for the other candidates. This form guarantees degeneration to temperature scaling, while reducing the cost.

Table 1: Accepted length L_{pos} averaged across tasks, computed on the test set for all the methods. . All methods choose the optimal parameter for each task. Temperature and diagonal Dirichlet scaling select the optimal parameter on the test set. Full-matrix Dirichlet scaling selects K and a trained checkpoint (epoch ≥ 1) on each task's validation split.

Method	GPT-OSS-120B EAGLE-3	GPT-OSS-120B DFlash	Qwen3.6-35B-A3B DSpark
$T_d = 1$	3.0739	3.6345	4.0731
Optimal temperature	3.5085	4.0751	4.4847
Optimal diagonal Dirichlet	3.5085	4.0753	4.4850
Full-matrix Dirichlet	3.4824	4.0563	4.4192

5.2.1 TRAJECTORY REPLAY FOR EFFICIENT VERIFICATION.

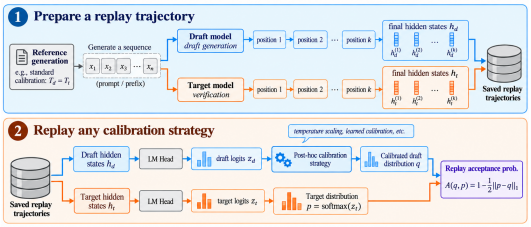

Figure 5: Illustration of the trajectory replay.

To test whether there is room for improvement by employing Dirichlet scaling efficiently, we introduce a trajectory replay pipeline to conduct quick checks for draft calibration methods. The idea of trajectory replay is inspired by the lossless property of speculative decoding, which implies that the draft distribution does not affect the distribution of generated sequences, and the acceptance probability can be computed by (1) given any q and p for a fixed draft position. Therefore, it is feasible to compute the performance of any draft calibration strategy on a saved specific trajectory together with the corresponding target and draft final hidden states – directly saving logits costs too much space, so we save the final hidden states and recover the logits when replay. The pipeline is illustrated in Figure 5, and a detailed description is in Appendix B.2.

The efficiency benefit of the replay is incredible: we conduct over a million replay evaluations in the following sections within GPU minutes, while real evaluation may take millions of GPU hours. There is indeed bias from this trajectory replay in simulating the speculative decoding performance in real inference. For example, we cannot straightforwardly transfer the definition of accepted length to trajectory replay, and we instead consider an alternative average accepted length $L_{\text{pos}} := 1 + \sum_{i=1}^{k} \prod_{j=1}^{i} A_j$, where A_j is the mean acceptance probability at draft position j over eligible reference contexts. Figure 7 in Appendix B.2 shows that L_{pos} approximately tracks online temperature trends, with mean absolute relative errors of 1.83%–2.64% across the three model pairs. Replay is therefore useful for screening calibration candidates, although it does not reproduce every temperature ranking or exact online gain.

5.2.2 DIRICHLET SCALING GENERALLY DOES NOT EXCEED TEMPERATURE SCALING

We evaluate the following results on a trajectory replay with $T_d = T_t = 1$ as the reference generation. Recall the Dirichlet scaling formula (2) and our simplification. We consider two cases to thoroughly examine the method:

- A diagonal exhaustive search: we take $W^{(K)}$ to be diagonal, equivalent to assigning separate temperatures to the top-K candidates and the tail, with $W^{(K)}[i,i] = 1/T_i$. Temperature scaling is recovered when all temperatures coincide. For $K = 7$, we independently search each of the eight temperatures over the grid $\mathcal{G} = \{0.2, 0.4, 0.6, 0.8, 1, 1.2\}$, giving $6^8 = 1,679,616$ candidates, and select the optimum per task on the test split.
- Full-matrix training: for $K \in \{1, 2, 3, 7\}$, we jointly optimize an unrestricted $W^{(K)}$ and $T_{\text{res}} > 0$ with Adam, minimizing a top-K-plus-tail total-variation surrogate across equally weighted tasks. Each task selects K and a trained checkpoint using validation L_{pos}, evaluated with full-vocabulary overlap. Appendix A.3 gives the training details.

Table 1 shows that exhaustive diagonal scaling offers negligible gains over temperature scaling, while full matrices trained with our surrogate loss perform even slightly worse. This reveals that temperature scaling may have captured the consistent distribution gap that applies to general tokens.

Table 2: Test L_{pos} averaged equally across tasks, in the same settings as Table 1. Optimal T_d is selected separately for each task on the test grid and shared across its tokens; token oracles select per position. Learned heads select hyperparameters, a trained checkpoint (epoch ≥ 1), and binary thresholds on each task's validation split.

Method	GPT-OSS-120B EAGLE-3	GPT-OSS-120B DFlash	Qwen3.6-35B-A3B DSpark
$T_d = 1$	3.0739	3.6345	4.0731
Optimal T_d (per task)	3.5085	4.0751	4.4847
Token oracle ($\mathcal{G}$)	4.0598	4.7628	4.9247
Token oracle ($\{0, 1\}$)	3.9263	4.5943	4.7941
Temperature head (h_d)	3.5111	4.1091	4.4765
Temperature head ($h_d, h_{t,\text{prev}}$)	3.5170	4.1091	4.4726
Binary head (h_d)	3.4031	4.0045	4.3281
Binary head ($h_d, h_{t,\text{prev}}$)	3.4002	3.9998	4.3234

5.3 Token-Adaptive Calibration Head

The trials in Dirichlet scaling reveal that temperature scaling may have recovered most of the distribution gap we can reach with post-hoc calibration methods shared across tokens within each task. We consider another axis in this section: calibration can be adaptive to each specific draft token, with evidence from confidence calibration in LLMs that a token-wise calibration head may help with better performance in confidence calibration (Xie et al., 2024). We consider only token-adaptive temperature scaling in this section. The related results are presented in Table 2, with detailed evaluation and training settings in Appendix A.3. We summarize the methods we tried and the observations.

- **Token-wise oracle shows improvement room.** At each replay position, we select the temperature maximizing $A(T) = \|p - q\|_1$ from the candidate temperature grid $\mathcal{G}$ or $\{0, 1\}$, where $T = 0$ denotes greedy drafting. As in Table 2, these target-informed oracles substantially outperform the task-specific scalar optimum. The gap is even larger than that between global temperature scaling and the original fixed $T_d = 1$, showing promising room for token-adaptive calibration when it can be done perfectly with temperature scaling. Moreover, the small $\{0, 1\}$ temperature grid also recovers more than 75% of the gain from a full temperature search grid, suggesting potential patterns in token-adaptive scaling.
- **Learned calibration heads generally fail to capture it.** We train a separate lightweight head per task to predict a positive temperature or a binary choice $T_d \in \{0, 1\}$, using the draft state h_d alone or concatenated with $h_{t,\text{prev}}$, the last eligible target state cached from the preceding verification block, as the input, to obtain information from both draft and target models to better predict the temperature for the current draft token. The training settings are specified in Appendix A.3. We consider this $\{0, 1\}$ grid choice because of its good oracle performance and potential simplicity in predicting a binary classification problem. We can observe that all the trained temperature heads remain near, or even slightly worse than, global temperature scaling, regardless of the binary classification and input design. This failure to capture the potential benefits of token-adaptive scaling may suggest more modifications may be needed beyond pure post-hoc calibration.

6 Conclusions

In this paper, we study the draft calibration for speculative decoding, focusing on post-hoc methods adjusting distributions at inference time. We show that applying a standard temperature scaling to the draft distribution already significantly improves the acceptance performance. We further examine token-adaptive temperature scaling and find potential room for improvement if we do it perfectly, but the current temperature head still fails to catch the improvement. One limitation of the work is that we have only studied post-hoc methods, but training methods, which have been extensively studied in the confidence calibration literature, may also be important for studying draft calibration, e.g., utilizing the potential room in token-adaptive temperature scaling or figuring out the exact reason for the consistent distribution gap. We consider this an interesting future direction.

AI USE STATEMENT

In this work, we used generative AI tools for implementing the methods, assisting in organizing and analyzing experiment results, assisting in finding related literature and materials, drafting some paragraphs of the paper, and helping with grammar and correctness checks. We have not used generative AI tools for developing and finding research questions and ideas. We have reviewed all AI-assisted work, checking the code and experiment results are consistent with the claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our experiments are all conducted on the open-source SpecForge/sglang framework for speculative decoding evaluation. The experiments and methodology are clearly described in the main paper, and all the details for reproducing our evaluation results are provided in Appendix A.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM inference with low-cost PARallel draft model adaptation. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10008956.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for Medusa decoding. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2402.05109.

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

Frédéric Berdoz, Peer Rheinboldt, and Roger Wattenhofer. Steering pretrained drafters during speculative decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30067–30075, 2026. doi: 10.1609/aaai.v40i36.40255. URL https://ojs.aaai.org/index.php/AAAI/article/view/40255.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

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*, 2023.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026.

Daewon Choi, Seunghyuk Oh, Saket Dingliwal, Jihoon Tack, Kyuyoung Kim, Woomin Song, Seojin Kim, Insu Han, Jinwoo Shin, Aram Galstyan, Shubham Katiyar, and Sravan Babu Bodapati. Mamba drafters for speculative decoding. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 21404–21418, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1167. URL https://aclanthology.org/2025.findings-emnlp.1167/.

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

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

Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 295–302, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.21. URL https://aclanthology.org/2020.emnlp-main.21/.

Zhipeng Ding, Xu Han, Peirong Liu, and Marc Niethammer. Local temperature scaling for probability calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6889–6899, October 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Ding_Local_Temperature_Scaling_for_Probability_Calibration_ICCV_2021_paper.html.

Doğaç Eldenk, Payal Mohapatra, Yigitcan Comlek, Kaan Oktay, Hongyang Zhang, and Stephen Xia. Attention drift: What autoregressive speculative decoding models learn. *arXiv preprint arXiv:2605.09992*, 2026.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Mude Hui, Xin Huang, Jaime Campos Salas, Yue Sun, Nathan Pemberton, Xiang Song, Ashish Khetan, and George Karypis. P-EAGLE: Parallel-drafting EAGLE with scalable training. *arXiv preprint arXiv:2602.01469*, 2026. URL https://arxiv.org/abs/2602.01469.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 623–631. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/kull17a.html.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f8c0c968632845cd133308b1a494967f-Paper.pdf.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2805–2814. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kumar18a.html.

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.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Laixin Xie, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. SpecForge: A flexible and efficient open-source training framework for speculative decoding. *arXiv preprint arXiv:2603.18567*, 2026a. URL https://arxiv.org/abs/2603.18567.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-Hard and BenchBuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024a. URL https://arxiv.org/abs/2406.11939.

Tianyi Li, Yaxin Luo, Xinyi Shang, and Zhiqiang Shen. Dartree: Speculative diffusion decoding with autoregressive draft trees. *arXiv preprint arXiv:2608.13524*, 2026b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024c.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11946–11965. Curran Associates, Inc., 2024. doi: 10.52202/079017-0381. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/16336d94a5ffca8de019087ab7fe403f-Paper-Conference.pdf.

Xiaoxuan Liu, Jiaxiang Yu, Jongseok Park, Ion Stoica, and Alvin Cheung. Speculative decoding: Performance or illusion? *arXiv preprint arXiv:2601.11580*, 2026. URL https://arxiv.org/abs/2601.11580v2.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 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*, 2024. doi: 10.1145/3620666.3651335. URL https://arxiv.org/abs/2305.09781v4.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip H. S. Torr, and Puneet K. Dokania. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15288–15299. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/aeb7b30ef1d024a76f21a1d40e30c302-Paper.pdf.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pp. 2901–2907, 2015. doi: 10.1609/aaai.v29i1.9602. URL https://ojs.aaai.org/index.php/AAAI/article/view/9602.

Siru Ouyang, Shuohang Wang, Minhao Jiang, Ming Zhong, Donghan Yu, Jiawei Han, and Yelong Shen. Temperature-centric investigation of speculative decoding with knowledge distillation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13125–13137, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.767. URL https://aclanthology.org/2024.findings-emnlp.767/.

Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. Lk losses: Direct acceptance rate optimization for speculative decoding. *arXiv preprint arXiv:2602.23881*, 2026.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/c4127b9194fe8562c64dc0f5bf2c93bc-Paper.pdf.

Cheng Wang. Calibration in deep learning: A survey of the state-of-the-art. *arXiv preprint arXiv:2308.01222*, 2023. URL https://arxiv.org/abs/2308.01222.

Pei-Shuo Wang, Jian-Jia Chen, Chun-Che Yang, Chi-Chih Chang, Ning-Chi Huang, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Speculate deep and accurate: Lossless and training-free acceleration for offloaded LLMs via substitute speculative decoding. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 54334–54357. Curran Associates, Inc., 2025. doi: 10.52202/085713-1815. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4e32ca54ae8a3a0ea4c1fd3663f73b0f-Paper-Conference.pdf.

Zhuofan Wen and Yang Feng. SpecBound: Adaptive bounded self-speculation with layer-wise confidence calibration. *arXiv preprint arXiv:2604.12247*, 2026. URL https://arxiv.org/abs/2604.12247.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.257. URL https://aclanthology.org/2023.findings-emnlp.257/.

Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordonez, and Dong Yu. ParallelSpec: Parallel drafter for efficient speculative decoding. *arXiv preprint arXiv:2410.05589*, 2024. URL https://arxiv.org/abs/2410.05589.

Johnathan Xie, Annie S Chen, Yoonho Lee, Eric Mitchell, and Chelsea Finn. Calibrating language models with adaptive temperature scaling. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18128–18138, 2024.

Jize Zhang, Bhavya Kailkhura, and T. Yong-Jin Han. Mix-n-Match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11117–11128. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/zhang20k.html.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.607. URL https://aclanthology.org/2024.acl-long.607/.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.15766.

Yaojie Zhang, Jianuo Huang, Junlong Ke, Yuhang Han, Yongji Long, Tianchen Zhao, Biqing Qi, and Linfeng Zhang. FlexDraft: Flexible speculative decoding via attention tuning and bonus-guided calibration. *arXiv preprint arXiv:2605.20022*, 2026. URL https://arxiv.org/abs/2605.20022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, 2023. URL https://arxiv.org/abs/2306.05685.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

Xuwen Zhou, Fangxin Liu, Chao Wang, Xiao Zheng, Hao Zheng, Min He, Li Jiang, and Haibing Guan. Calibrated speculative decoding: Frequency-guided candidate selection for efficient inference. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29677–29688, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1369. URL https://aclanthology.org/2026.acl-long.1369/.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32011–32050, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8766fbc68e1ed1cdef712ce273e0a363-Paper-Conference.pdf.

A Detailed Experiment Settings

A.1 Models and decoding configuration

Table 3 lists all nine model pairs in Figure 3. Here K counts proposed tokens, excluding the additional correction or bonus token, so a fully accepted block commits $K+1$ tokens. Chain refers to the proposal topology: EAGLE-3 generates proposals autoregressively, while DFlash and DSpark use their block drafting mechanisms without branching. We keep the released model weights fixed and apply calibration at inference time. We did not apply the confidence-based verification truncation for DSpark to keep the budget fixed.

Table 3: Target and draft checkpoints used in the experiments. Target names and draft repository IDs link to their Hugging Face pages. K is the number of draft positions, fixed within each model pair.

Target model	Method	K	Draft checkpoint
Gemma-4-31B-IT	EAGLE-3	5	`RedHatAI/gemma-4-31B-it-speculator.eagle3`
Qwen3-30B-A3B-Instruct-2507	EAGLE-3	3	`lmsys/SGLang-EAGLE3-Qwen3-30B-A3B-Instruct-2507-SpecForge-Nex`
GPT-OSS-120B	EAGLE-3	9	`lmsys/EAGLE3-gpt-oss-120b-bf16`
Qwen3.5-27B	DFlash	7	`z-lab/Qwen3.5-27B-DFlash`
Qwen3.6-35B-A3B	DFlash	15	`z-lab/Qwen3.6-35B-A3B-DFlash`
GPT-OSS-120B	DFlash	9	`z-lab/gpt-oss-120b-DFlash`
Gemma-4-31B-IT	DSpark	7	`RedHatAI/gemma-4-31B-it-speculator.dspark`
Qwen3.6-35B-A3B	DSpark	8	`RedHatAI/Qwen3.6-35B-A3B-speculator.dspark`
Qwen3.8-27B-FP8	DSpark	7	`RadixArk/Qwen3.8-27B-DSpark`

Sampling and prompt variants. Temperature scaling sweeps use $T_d \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2\}$ with $T_t = 1$; $T_d = 0$ denotes a point mass at the draft argmax. We disable sampling truncation (`top_p=1`, `top_k=-1`) and use base seed 980406 with request-index offsets. Chat runs use the target tokenizer's chat template. Completion uses the same task prompt without a chat template. The distribution plots in Figures 1 and 2 use the low/non-thinking settings. The calibration replay uses the low/non-thinking settings. Best temperatures in the online plots are selected retrospectively from the displayed grid.

Serving configuration. We evaluate acceptance using our SpecForge/SGLang serving integration with independent draft-temperature control and probability-ratio rejection sampling. The per-server active-request limits in the nine panels of Figure 3, in row-major order, are 8, 64, 64, 8, 16, 64, 16, 20, and 18. Client concurrency can exceed these limits because requests queue; these values are capacity limits under continuous batching, not fixed batch sizes. Comparisons vary calibration within a model pair and retain its serving configuration, rather than comparing throughput across different model pairs.

A.2 Datasets

Table 4: Evaluation datasets, broad task categories, and numbers of prompts used per setting. Repository links identify the Hugging Face sources; the text specifies the frozen evaluation snapshot.

Domain	Benchmark	Prompts	Hugging Face dataset
Math	GSM8K (Cobbe et al., 2021)	1,319	`openai/gsm8k`
Math	MATH-500 (Lightman et al., 2023)	500	`HuggingFaceH4/MATH-500`
Math	AIME 2025 (Dekoninck et al., 2026)	30	`MathArena/aime_2025`
Code	HumanEval (Chen et al., 2021)	164	`openai/openai_humaneval`
Code	MBPP (Austin et al., 2021)	257	`google-research-datasets/mbpp`
Code	LiveCodeBench (Jain et al., 2024)	1,055	`livecodebench/code_generation_lite`
Chat	MT-Bench (Zheng et al., 2023)	80	`HuggingFaceH4/mt_bench_prompts`
Chat	Arena-Hard v2 (Li et al., 2024a)	750	`lmarena-ai/arena-hard-auto`
	Total	4,155	

We use the frozen prompt JSONL files from DeepSpec's evaluation suite at revision `005e03b8`, retaining all rows of the eight datasets in Table 4. These comprise GSM8K's `main/test`

split, MATH-500's `test` split, MathArena AIME 2025's `train` split, HumanEval's `test` split, MBPP's `sanitized/test` split, the 1,055-problem LiveCodeBench snapshot assembled from `test.jsonl` through `test6.jsonl`, MT-Bench's 80 conversations, and Arena-Hard's `data/arena-hard-v2.0/question.jsonl`. Split names here follow the dataset repositories; all serve as prompt workloads for our decoding experiments.

A.3 Replay evaluation and calibration training

The replay experiments use GPT-OSS-120B/EAGLE-3 and GPT-OSS-120B/DFlash with low reasoning effort ($K = 9$), and Qwen3.6-35B-A3B/DSpark with thinking disabled ($K = 8$). Reference trajectories use $T_d = T_t = 1$. We cache final hidden states in BF16 and reconstruct distributions with the corresponding output heads, including model-specific normalization and vocabulary mappings. Eligible positions are determined by acceptance of all preceding proposals on the reference trajectory. Within each task, we split complete requests into 40%/10%/50% train/validation/test partitions; all blocks from a request stay together. This yields 1,660/413/2,082 requests per model pair. All calibration tables use the same test partition and average tasks equally. Replay scores describe these fixed contexts and do not directly measure fresh-rollout throughput.

Dirichlet calibration. The diagonal search uses the six-point temperature grid $\{0.2, 0.4, 0.6, 0.8, 1, 1.2\}$ independently for each of the top seven ranks and the tail, giving $6^8 = 1,679,616$ candidates. Full-matrix calibration uses $K \in \{1, 2, 3, 7\}$ calibrated vocabulary ranks, independently of the draft chain length. The unrestricted $W^{(K)}$ acts on ranks sorted by the original draft probabilities, and $T_{\text{res}} = \exp(-s) > 0$ is jointly trained, with no additive bias. Each supported tail token retains its individual transformed score before full-vocabulary normalization.

Let $v_1, \ldots, v_K$ be the draft's top-K token IDs and $g_K(r) = (r(v_1), \ldots, r(v_K), 1 - \sum_{j=1}^{K} r(v_j))$. We minimize the coarse total-variation loss $\ell_{\text{TV}} = \frac{1}{2}\|g_K(q_{\text{cal}}) - g_K(p)\|_1$. The target residual includes mass outside the draft vocabulary; grouping is used only for the loss. This surrogate lower-bounds full-distribution total variation, whereas validation and test scores use exact full-vocabulary overlap. Matrices are trained across tasks with equal task weight; Table 1 selects the rank cutoff, learning rate, initialization, and checkpoint separately for each task using validation L_{pos}. There is no separate matrix training per task. Temperature and diagonal optima are selected per task on the test set and serve as oracle comparators.

Token-adaptive heads. We train a separate head per task, using either the final normalized draft state h_d or its concatenation with the last eligible target final-layer state cached from the preceding verification block, $h_{t,\text{prev}}$. Each input vector is RMS-normalized; a missing previous state is replaced by zeros. This cached target state can precede the bonus token when its state was not saved. The current target distribution is used only for supervision and evaluation. Input dimensions are 2,880/5,760 for GPT-OSS and 2,048/4,096 for DSpark. Heads are linear or of the form $V\,\text{SiLU}(Ux) + c$, with bottleneck width 8 or 32 and one (temperature) or two (binary) outputs. A scalar output predicts $T = \exp(\text{clamp}(z, -7, 4))$ and minimizes ℓ_{TV} with $K = 7$. The binary head uses label $\mathbf{1}[A_i(1) > A_i(0)]$ and cross-entropy weighted by $0.8^{i-1}|A_i(1) - A_i(0)|$, normalized by the mean training weight, at draft position i. Its predicted probability for $T = 1$ is thresholded over $\{0, 0.025, \ldots, 1\} \cup \{1.001\}$ on validation. Each head family and input variant selects its hyperparameters, positive-epoch checkpoint, and binary decision threshold on that task's validation L_{pos}.

Optimization and selection. Both matrix and head training use Adam with learning rates $\{10^{-3}, 10^{-2}, 5 \times 10^{-2}\}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, batch size 8,192, and 32 epochs, with no weight decay or learning-rate schedule. Candidate checkpoints are taken after 1, 2, 4, 8, 16, 24, and 32 epochs. Matrices start from identity or a diagonal calibration selected by task-averaged validation L_{pos}; temperature heads start from $T = 1$ or a task-specific validation-selected scalar.[1] Binary heads start from the utility-weighted training label frequency. Learned choices are frozen before test evaluation.

[1] The scalar initialization uses the block-wise replay score $1 + \sum_{i=1}^{k} \prod_{j=1}^{i} A_j$, averaged over blocks with uncached positions set to zero, and is bounded below by 0.2. Final head selection uses L_{pos}.

Oracle comparators. The scalar baseline selects one temperature per task from $\mathcal{G} = \{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 2\}$ by test L_{pos}. Token oracles instead maximize $A(T)$ separately at each eligible cached position over $\mathcal{G}$ or $\{0, 1\}$, with $A(0) = p(\arg\max_x q(x))$. They use the current target distribution and provide diagnostic upper bounds within their respective grids on these cached contexts; they are not deployable heads.

B Additional Experiment Results

B.1 Observations

Temperature scaling for DeepSeek-trained draft models. We additionally evaluate DeepSeek's Qwen3-14B draft checkpoints for EAGLE-3, DFlash, and DSpark. These online sweeps use our modified DeepSpec evaluator with independent draft-temperature control, $T_t = 1$, seven draft positions, thinking disabled, and a 2,048-token generation cap. DSpark's confidence head is disabled. We use all 4,155 prompts from the eight benchmarks in Table 4 and average the recorded accepted lengths equally across tasks.

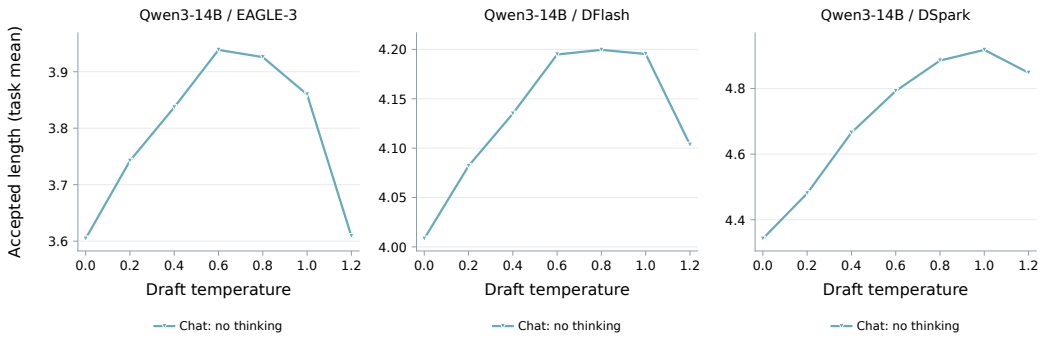

Figure 6: Temperature scaling for the three DeepSeek-trained Qwen3-14B draft models. Each curve is the accepted length averaged equally across the eight benchmarks, with thinking disabled and target temperature fixed at $T_t = 1$.

Figure 6 shows limited gains from a global temperature correction for these checkpoints. EAGLE-3 improves from 3.860 at $T_d = 1$ to 3.939 at $T_d = 0.6$ (+2.05%). DFlash is nearly flat around its optimum: 4.195 at $T_d = 1$ versus 4.200 at $T_d = 0.8$ (+0.10%). DSpark attains its highest task mean, 4.918, at $T_d = 1$. These optima are selected retrospectively from the displayed grid. Thus, different from the draft models shown in Figure 3, these three draft models are better calibrated, and temperature scaling only gives marginal help. It will be an interesting future direction to find out what the secret sauce is inside this, which may help with further exploration of token-adaptive draft calibration.

B.2 Offline Trajectory Replay

As plotted in Figure 5, we here illustrate the trajectory replay process in words. To prepare a trajectory for replay, we first generate a sequence with the desired target and draft models with a reference calibration strategy, e.g., the standard temperature choice $T_d = T_t$. During the generation, we save all the final hidden states of draft and target models for every draft position, which can be obtained during draft generation and the verification process. Then, when doing replay, at each draft position, we input the final hidden states to the LM Head to recover draft and target logits. Post-hoc calibration strategies are then applied to draft logits to obtain the calibrated draft distribution q, and compute the acceptance probability based on (1) to test the calibration strategy. The pipeline is illustrated in Figure 5.

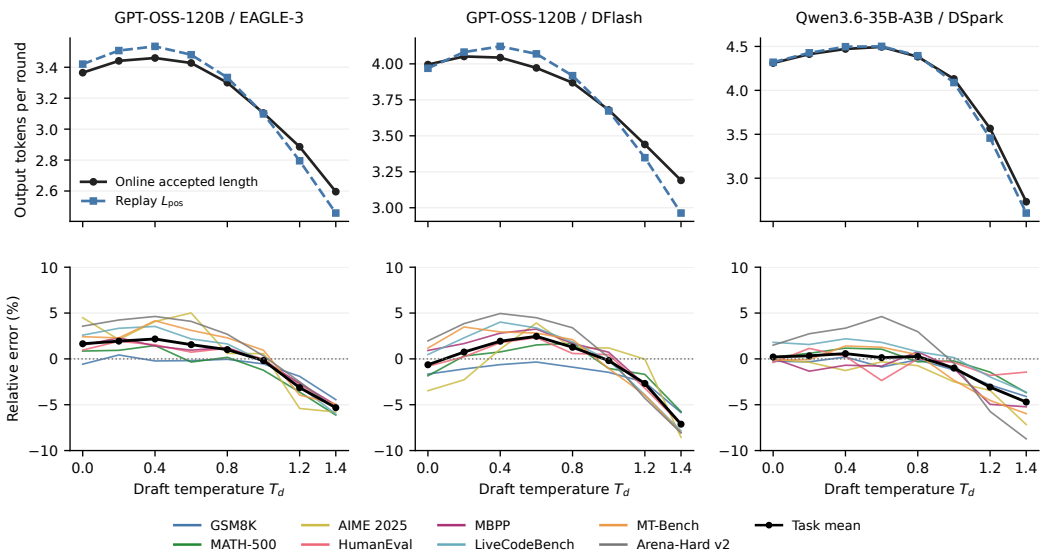

Figure 7: Temperature replay versus online decoding. Top: online accepted length and replay L_{pos}, each averaged equally across tasks. Bottom: signed relative error $100(L_{\text{pos}} - L_{\text{online}})/L_{\text{online}}$, with colored curves for individual tasks and black curves computed from the task means. Target temperature is fixed at $T_t = 1$; draft lengths are $K = 9, 9, 8$ from left to right. These are full-task diagnostics, separate from the held-out calibration tables.

Replay–online comparison. We compare full-task temperature replay against the historical online sweeps for the same three model pairs and low/non-thinking settings used in Appendix A.3, with $T_t = 1$ and $T_d \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4\}$. Each comparison uses all 4,155 prompts across the eight tasks, rather than the test-only partition used in the calibration tables. Replay evaluates the fixed $T_d = 1$ reference contexts; online runs generate fresh proposals at each temperature. We use runs without distribution instrumentation where available; the $T_d = 1$ online reference is instrumented.

Across the 64 task–temperature combinations per model pair, mean absolute relative errors are 2.42%, 2.64%, and 1.83% for EAGLE-3, DFlash, and DSpark, respectively; maximum errors are 6.13%, 8.56%, and 8.72%. The task-mean optimal temperatures agree for EAGLE-3 (0.4) and DSpark (0.6); for DFlash, replay selects 0.4 versus the online optimum 0.2, costing only 0.008 output tokens per verification round on the online task mean. Thus replay captures the broad benefit of sharpening, but fixed reference contexts, reference-policy eligibility, and products of mean position-wise acceptance rates can bias its estimates. We use it as a diagnostic proxy rather than an exact prediction of online acceptance or throughput.