BEYOND GLOBAL THRESHOLDS: TOKEN-LEVEL CALIBRATED ADAPTIVE PARALLEL DECODING FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) are a promising alternative to autoregressive models because they can generate multiple tokens in parallel. However, they remain slow in practice because inference requires many denoising steps. Recent methods accelerate inference by using a fixed confidence threshold to decide which tokens can be decoded in parallel. Such methods implicitly treat confidence as having the same meaning across different predicted tokens. In this work, we show that different tokens exhibit substantially different final-consistency rates even at similar confidence levels, revealing systematic token-dependent variation in the confidence–stability relationship. Based on this observation, we propose Token-Level Calibrated Adaptive Parallel Decoding (TC-APD), which estimates a separate confidence threshold for each token from offline denoising traces. During inference, these thresholds are used through a simple token-level lookup to guide adaptive parallel decoding without additional training, neural components, or model forward passes. Experiments on mathematical reasoning and code-generation tasks show that TC-APD improves the speed–quality trade-off over fixed-threshold baselines while maintaining comparable task performance. Overall, our results demonstrate the effectiveness of calibrating confidence thresholds at the token level for efficient diffusion language model inference.

1 INTRODUCTION

Diffusion language models (DLMs) (Song et al., 2025; Li et al., 2025b) promise a fundamental departure from the rigid left-to-right constraints of autoregressive generation, offering flexible token orderings and the tantalizing possibility of decoding multiple tokens in parallel. In practice, however, this theoretical parallelism is notoriously brittle. State-of-the-art DLMs, such as LLaDA (Nie et al., 2026; Zhu et al., 2026; Bie et al., 2025), Dream (Ye et al., 2025), and SDAR (Cheng et al., 2026), still rely on hundreds of sequential denoising steps, as aggressive parallel decoding often severely degrades generation quality. This creates a critical bottleneck: how can we accelerate DLM inference without sacrificing performance?

A widely used solution to reduce the number of denoising steps is confidence-aware adaptive parallel decoding (APD) (Wu et al., 2025; Israel et al., 2026; Cao et al., 2026). At each denoising step, the model predicts a token and an associated confidence score for every masked position. Tokens whose confidence exceeds a fixed global threshold (e.g., 0.9) are committed in parallel, while the remaining tokens stay masked for future refinement. This simple strategy has been widely adopted in recent acceleration methods because it avoids prematurely committing low-confidence tokens. However, fixed-threshold decoding relies on a strong implicit assumption: confidence scores are comparable and equally calibrated across different tokens. This assumption raises a fundamental question: do different tokens exhibit distinct confidence dynamics and final-consistency criterion during decoding? If so, **should different tokens be assigned different thresholds for adaptive parallel decoding?**

We analyze the confidence dynamics during the decoding process of DLMs. First, we observe that tokens differ substantially in their average decoding confidence. Under a single fixed threshold, frequent but relatively low-confidence tokens are often not selected for parallel decoding, thereby

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

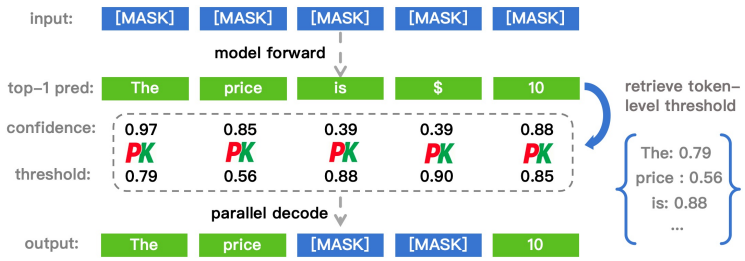


Figure 1: Illustration of a single decoding step in Token-Level Calibrated Adaptive Parallel Decoding (TC-APD). *PK* represents the token-wise comparison between decoding confidence and the corresponding token-specific threshold.

reducing overall decoding efficiency. Second, we find that even at early decoding steps, some low-confidence tokens already match the final decoded result. Moreover, higher confidence generally correlates with a higher probability of being consistent with the final output.

These findings motivate us to calibrate token-specific thresholds based on the observed relationship between decoding confidence and final consistency. We propose Token-Level Calibrated Adaptive Parallel Decoding (TC-APD), an interpretable and plug-and-play method that estimates token-specific thresholds from offline denoising traces. At inference time, TC-APD simply replaces the shared global threshold with a token-specific lookup, requiring no additional training, neural modules, or model forward passes. This allows the decoding policy to commit different predicted tokens at different confidence levels and improves the speed–quality trade-off of confidence-aware adaptive parallel decoding.

Our contributions are as follows:

- We show that confidence–stability relationships vary across tokens, making a shared global threshold suboptimal.
- We propose TC-APD, a simple plug-and-play token-level calibration method requiring no additional training or forward passes.
- We demonstrate improved speed–quality trade-offs across models and benchmarks.

2 Related Work

Diffusion language models Diffusion language models generate text through iterative denoising rather than strict left-to-right factorization. Early work (Gong et al., 2022; Han et al., 2023; Gong et al., 2023) explored discrete or continuous diffusion formulations for text generation, while recent large-scale DLMs have substantially narrowed the gap with autoregressive LLMs (Touvron et al., 2023; Team, 2025). LLaDA (Nie et al., 2026) scales masked diffusion language modeling to the 8B-parameter regime and demonstrates competitive instruction-following and reasoning ability. Dream (Ye et al., 2025) further improves large-scale diffusion language modeling through AR-based initialization and token-level noise rescheduling, showing strong planning and flexible inference capabilities. Other recent models, such as SDAR (Cheng et al., 2026), explore hybrid diffusion-autoregressive paradigms that combine the training efficiency of autoregressive models with the parallel inference potential of diffusion models. These models establish DLMs as a viable generation paradigm, but their practical inference efficiency still depends heavily on the decoding algorithm.

Inference acceleration for DLMs Recent work improves diffusion language model efficiency from different perspectives. Fast-dLLM (Wu et al., 2025) uses block-wise KV caching and confidence-based parallel decoding to improve throughput. WINO (Hong et al., 2025) introduces a draft-and-verify mechanism with token re-masking for unreliable predictions. SlowFast Sampling (Wei et al., 2025) alternates between fast and slow decoding stages based on confidence and convergence signals. Prophet (Li et al., 2025a) performs sequence-level early stopping by exploiting the observation that DLMs often stabilize before the final denoising step. Our method follows

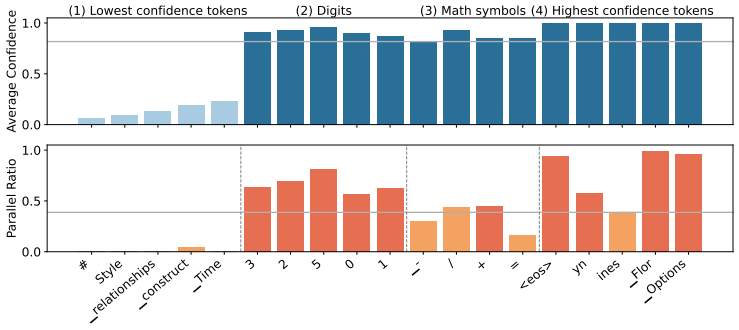


Figure 2: Token-level confidence and parallel decoding ratio across 4 groups of tokens. Top: average decoding confidence. Bottom: parallel decoding ratio. Results show that decoding confidence varies across tokens, indicating that **a fixed global threshold has limitations**. The gray dashed line indicates the mean confidence over the top-500 most frequent tokens, while lighter bars denote values below the mean.

widely used confidence-aware adaptive parallel decoding (Wu et al., 2025; Israel et al., 2026), but shows that a single fixed threshold is suboptimal due to strong token-level final-consistency differences. We propose calibrated token-level thresholds to enable more fine-grained commitment decisions. TC-APD can also be viewed as a form of token-level early stopping. Unlike Prophet's sequence-level criterion, our method allows different tokens to be committed at different confidence levels.

Dynamic thresholding for DLMs Recent work replaces fixed confidence thresholds with adaptive token-selection signals. Dynamic-dLLM (Wu et al., 2026) uses the second-highest probability as a model-concentration signal to update token-level thresholds. Jot (Kohut et al., 2026) performs token-level early stopping using local confidence, while STDD (Sun et al., 2025) and STaRR (Yu et al., 2026) exploit temporal variance and spatial deviance to control token refinement. OSDT (Shen & Ro, 2025) calibrates a threshold schedule from a single sample, and Learn2PD (Bao et al., 2025) trains a lightweight filter for token-commitment decisions. Other training-free approaches leverage prediction history: CreditDecoding (Wang et al., 2026) accumulates historical logits, DecoCal (Xu et al., 2026) calibrates confidence across denoising steps and allows premature commitments to be revised, and KLASS (Kim et al., 2026) uses inter-step KL divergence to identify stable predictions.

These methods primarily adapt token-commitment decisions online using current or history-dependent decoding signals. In contrast, TC-APD performs **offline token-level calibration** and offers a **plug-and-play** alternative: it requires no additional neural module or model forward pass, and can be integrated into existing confidence-based APD methods by simply replacing the global threshold with a token-specific lookup. This provides an interpretable token-level calibration criterion based on empirical final consistency.

3 FINDINGS

We analyze all-step denoising traces to study differences in token-level confidence during decoding and their impact on generation. At each denoising step, for all masked positions, we record the top-1 prediction of the model and its confidence, and compare them with the final output of the model. We adopt block-wise diffusion decoding for inference, which enables further acceleration via KV-cache reuse and offers better practical efficiency than standard pure diffusion decoding.

Finding 1: Token-level confidence varies significantly across tokens. We first investigate whether different tokens exhibit variation in decoding confidence. We select the top-500 most frequent tokens, as these tokens appear more frequently during generation and therefore have a larger impact on the overall decoding process. We then select four representative token groups from these frequent tokens: (a) lowest-confidence tokens, (b) digits, (c) math symbols, and (d) highest-

confidence tokens. We compute the average decoding confidence of each token across all samples, as shown in Figure 2 (top).

We observe that certain tokens, such as “relationships” and “construct”, consistently exhibit very low average confidence. In contrast, tokens that are typically considered difficult, such as digits and math symbols, surprisingly show higher confidence values. Therefore, a global fixed threshold τ cannot be well calibrated across all tokens simultaneously. For tokens that are consistently decoded with low confidence, even when the model repeatedly predicts them as top-1, their confidence often remains below the global threshold. We further apply adaptive parallel decoding (Wu et al., 2025) and measure the proportion of each token committed through the parallel decoding mode, as shown in Figure 2 (bottom). These tokens are rarely committed early, leading to reduced throughput and weakening the practical efficiency gains of parallel decoding.

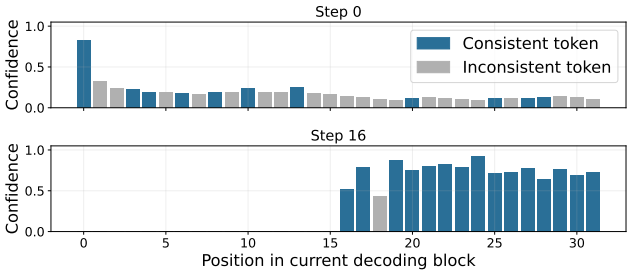


Figure 3: Token-level decoding consistency within a block. The top and bottom figures correspond to the first and halfway decoding steps. Blue indicates tokens whose predictions are already **consistent** with the final decoded output.

Finding 2: Low-confidence tokens show early-stopping behavior. We find that in many cases, the current top-1 prediction has moderate confidence but already matches the final-step decoding result. As shown in Figure 3, we show a block at the first step (top) and at the middle step (bottom), and report token-level confidence. We mark in blue the positions where the prediction already matches the final output. At step 0, although confidences are relatively low, 11/32 tokens are already consistent with the final-step result. This effect becomes stronger at step 16, where almost all tokens match the final-step output, even though confidence still varies. In other words, some tokens meet an early-stopping condition before they become highly confident. This suggests that setting a fixed threshold is too conservative.

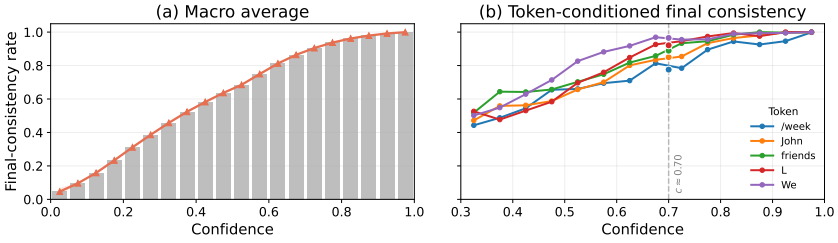


Figure 4: Confidence–final-consistency relationship. (a) Final consistency increases with confidence across tokens. (b) Different tokens exhibit different final-consistency rates at the same confidence level, motivating token-specific calibration.

Finding 3: Confidence is positively associated with final consistency, but the relationship is token-dependent. Our goal is to determine when an intermediate token prediction can be committed early based on its confidence and empirical final consistency. At each denoising step, **confidence** is the model probability assigned to the current top-1 prediction. We define **final consistency** as whether this intermediate prediction matches the token produced by the final conservative decoding result. Thus, final consistency measures whether an intermediate prediction already agrees with the final decoded token, rather than whether it matches the ground-truth answer.

As shown in Figure 4(a), the final-consistency rate increases steadily with confidence, indicating that confidence provides a useful signal of prediction stability. However, this relationship is not identical across tokens. Figure 4(b) shows token-conditioned final-consistency curves for representative tokens. Even at the same confidence level, such as $c \approx 0.70$, different tokens exhibit noticeably different final-consistency rates. Therefore, a single global confidence threshold does not correspond to the same final-consistency level for every token. This observation motivates token-specific calibration: instead of using one shared threshold for all tokens, we estimate a separate confidence threshold for each token according to its own confidence–final-consistency relationship.

4 METHOD

Based on these findings, we propose a lightweight and interpretable token-level thresholding framework, Token-Level Calibrated Adaptive Parallel Decoding (**TC-APD**). In the offline stage, TC-APD estimates token-specific confidence thresholds from denoising traces by calibrating the relationship between confidence and final consistency, and stores them as a lookup table aligned with the model vocabulary. During inference, the calibrated thresholds are directly applied to confidence-aware adaptive parallel decoding, enabling token-wise commitment decisions without additional model computation. Figure 1 illustrates the inference process of TC-APD.

4.1 SELF-FINAL CONSISTENCY FOR TOKEN-LEVEL CALIBRATION

Denoising traces. Given a prompt $x_{1:m}$, a DLM generates a continuation of length n by iteratively filling masked positions. At each denoising step t, for each still-masked position i, the model outputs a distribution over the vocabulary. We denote the top-1 predicted token by

$$\hat{v}_{i,t} = \arg\max_{v} p_\theta(v \mid x_t, i),$$

and its confidence by

$$c_{i,t} = p_\theta(\hat{v}_{i,t} \mid x_t, i).$$

In practice, $c_{i,t}$ is the top-1 probability after softmax. Other confidence definitions, such as confidence margin or negative entropy, can also be used, but we use top-1 probability as the default.

Self-final consistency. In many reasoning and coding tasks, supervision is only available at the sequence level, such as whether the final answer is correct or whether generated code passes the tests. Such feedback does not directly provide token-level signals for calibrating when an intermediate prediction can be committed.

We therefore construct a token-level calibration signal from the model's own denoising trajectory. We obtain a reference sequence using standard block-wise decoding with 256 denoising steps and a block size of 32, without adaptive early commitment. Let y_i^* denote the final token at generation position i in this reference sequence. For each intermediate prediction at position i and denoising step t, we define the binary final-consistency label:

$$z_{i,t} = \mathbb{I}[\hat{v}_{i,t} = y_i^\star].$$

This label measures whether the current top-1 prediction agrees with the final token produced by standard decoding. It requires no ground-truth annotation and can be computed directly from offline denoising traces. We use this empirical final-consistency signal to calibrate token-specific confidence thresholds.

Empirical consistency target. For each token v, we collect all trace records where the current top-1 prediction equals v:

$$D_v = \{(c_{i,t}, z_{i,t}) : \hat{v}_{i,t} = v\}.$$

Given a candidate threshold τ, we estimate the empirical final-consistency rate of token v above τ:

$$R_v(\tau) = \frac{\sum_{(c,z)\in D_v} \mathbb{I}[c \geq \tau] z}{\sum_{(c,z)\in D_v} \mathbb{I}[c \geq \tau]}.$$

We choose the lowest threshold that satisfies a target empirical final-consistency level ρ:

$$\tau_v = \min\left\{\tau : R_v(\tau) \geq \rho,\ N_v \geq N_{\min},\ A_v(\tau) \geq A_{\min}\right\},$$

where N_v denotes the total number of calibration observations for token v, and $A_v(\tau)$ denotes the number of observations whose confidence exceeds τ. $N_{\min}$ and $A_{\min}$ are support constraints used to avoid unstable estimates for rare tokens or thresholds with insufficient calibration samples. Importantly, ρ is an empirical calibration criterion defined on offline denoising traces rather than a guarantee on the subsequent accelerated decoding trajectory.

Fallback and threshold capping. Not all tokens have enough calibration records. If a token does not satisfy the support constraints, we assign it a default threshold τ_{default}. To avoid overly conservative token-specific thresholds, we cap all thresholds by a maximum value:

$$\tau_v \leftarrow \min(\tau_v, \tau_{\max}).$$

In our implementation, $\tau_{\max}$ is set to the default threshold, so calibrated thresholds can relax token-specific acceptance but cannot become more conservative than the chosen global maximum.

4.2 Calibrated Confidence-Aware Adaptive Decoding

After calibration, we obtain a token-threshold table $\{\tau_v\}_{v\in\mathcal{V}}$. At inference time, we use this table to replace the fixed global threshold in confidence-based adaptive parallel decoding. The detailed inference procedure is provided in Appendix C.5.

Token-wise threshold lookup. At each decoding step, the model predicts a top-1 token $\hat{v}_i$ and its confidence c_i for each masked position i. We retrieve the corresponding token-specific threshold:

$$\tau_i = \tau_{\hat{v}_i}.$$

If $\hat{v}_i$ does not have a calibrated threshold, we use τ_{default}. A position is eligible for parallel commitment if

$$c_i \geq \tau_i.$$

Parallel commitment. Let $\mathcal{S}$ denote the set of masked positions whose confidence exceeds their token-specific thresholds:

$$\mathcal{S} = \{i : c_i \geq \tau_{\hat{v}_i}\}.$$

All positions in $\mathcal{S}$ are committed in parallel using their top-1 predictions:

$$\hat{v}_i, \qquad \forall i \in \mathcal{S}.$$

In this way, TC-APD can adaptively commit different numbers of tokens at each decoding step according to their token-specific calibrated thresholds, without introducing additional model computation.

Fallback. If no masked position satisfies its threshold, we follow standard adaptive parallel decoding and commit the single highest-confidence position. Decoding continues until no masked positions remain.

5 Experiments

5.1 Setup

Models and benchmarks. We evaluate our method on two diffusion language models: LLaDA-8B-Instruct (Nie et al., 2026) and Dream-7B-Base (Ye et al., 2025). Experiments are conducted on mainstream reasoning benchmarks: HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), and MATH500 (Lightman et al., 2023). We evaluate LLaDA in the zero-shot setting, while Dream-7B-Base is evaluated under zero-shot on HumanEval and 3-shot settings for math reasoning benchmarks.

Calibration data and implementation. We use the Minerva-Math (Lewkowycz et al., 2022) training set for calibration. All experiments are conducted on 80GB A100 GPUs with a maximum generation length of 256 and block size 32. KV-cache is disabled for all methods to isolate the effect of the decoding policy. Additional implementation details and calibration-data analyses are provided in Appendix C.

Table 1: Main results on LLaDA-8B-Instruct. We report accuracy, NFE, and speedup. The values in parentheses denote the maximum and default thresholds τ_{default} used for APD methods.

Method	GSM8K			MATH500			HumanEval			Average		
	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp
Standard decoding	72.9	256	x1.00	32.4	256	x1.00	38.4	256	x1.00	47.9	256.0	x1.00
Mainstream Acceleration Methods												
WINO	71.3	51.1	x5.01	30.8	73.7	x3.47	35.0	61.9	x4.14	45.7	62.2	x4.12
Prophet	74.2	111.1	x2.30	28.6	111.8	x2.29	34.1	103.2	x2.48	45.6	108.7	x2.36
Fixed / Dynamic / Learning-based Threshold Methods												
Fixed APD (0.80)	73.0	60.6	x4.22	33.2	87.8	x2.92	37.0	76.5	x3.35	47.7	74.9	x3.42
Fixed APD (0.90)	72.9	77.5	x3.30	32.4	106.0	x2.42	37.8	100.7	x2.54	47.7	94.7	x2.70
Dynamic-DLLM	73.1	134.1	x1.91	32.0	155.6	x1.65	26.8	172.8	x1.48	44.0	154.2	x1.66
OSDT	68.6	40.8	x6.27	30.2	61.2	x4.18	27.4	61.8	x4.14	42.1	54.6	x4.69
Learn2PD	69.9	57.8	x4.43	33.4	88.4	x2.90	37.8	78.8	x3.25	47.0	75.0	x3.41
Ours: Token-level Calibrated Threshold Method												
TC-APD (0.80)	72.3	53.8	x4.76	33.2	80.7	x3.17	37.6	62.1	x4.12	47.7	65.5	x3.91
TC-APD (0.90)	72.1	56.4	x4.54	33.4	82.5	x3.10	38.4	66.9	x3.83	48.0	68.6	x3.73

Baselines and evaluation. We compare against three groups of methods:

- **Fixed-threshold adaptive parallel decoding**, as used in Fast-DLLM (Wu et al., 2025), which serves as the main baseline for evaluating token-level calibrated thresholds against a fixed global threshold.
- **Other mainstream acceleration methods**, including Prophet (Li et al., 2025a) and WINO (Hong et al., 2025), which improve efficiency via sequence-level early-stopping or draft-and-verify strategies for diffusion decoding.
- **Dynamic thresholding and learning-based methods**, including Dynamic-DLLM (Wu et al., 2026) and OSDT (Shen & Ro, 2025), which dynamically adjust decoding thresholds based on token- or step-level signals, and Learn2PD (Bao et al., 2025), which learns a lightweight prediction module for token commitment decisions. Jot (Kohut et al., 2026) is not included in our comparison because its official implementation was not publicly available during our experimental period.

Following prior work (Li et al., 2026; Wan et al., 2026), we use the number of function evaluations (NFE) as the primary, hardware-agnostic measure of inference cost and report the corresponding speedup. We additionally report wall-clock throughput and speedup to confirm that NFE reductions translate into practical gains. All results are evaluated with `lm-eval-harness` (Gao et al., 2024) under a consistent protocol across benchmarks.

Hyperparameters. We set $N_{\min} = 200$ and $A_{\min} = 100$ by default to ensure that each token has enough calibration samples for stable threshold estimation. The empirical final consistency target ρ is selected from $\{1.0, 0.995, 0.990\}$ using a held-out subset of the calibration data, and we choose $\rho = 0.995$ for all final results on both models.

5.2 Main Results

Table 1 summarizes the main results on LLaDA-8B-Instruct. Existing acceleration methods improve efficiency through different mechanisms, such as heuristic decoding strategies, draft-and-verify, or dynamic thresholding. However, more aggressive acceleration can come at the cost of generation quality. For example, OSDT reduces accuracy from 72.9 to 68.6 on GSM8K and from 32.4 to 30.2 on MATH500, while Learn2PD also shows a noticeable drop on GSM8K.

We directly compare TC-APD with fixed-threshold adaptive parallel decoding under maximum threshold settings of 0.80 and 0.90. Under both settings, TC-APD consistently reduces NFE while maintaining comparable task accuracy. For example, at a maximum threshold of 0.90, TC-APD reduces the average NFE from 94.7 to 68.6 while achieving an average accuracy of 48.0, compared

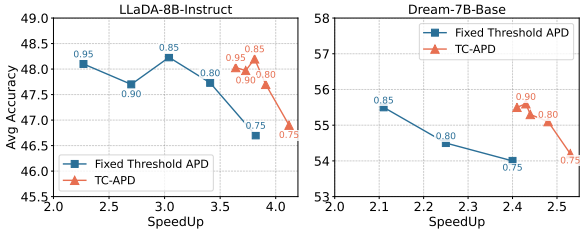


Figure 5: Accuracy–speedup trade-off under different threshold settings (0.75–0.95). Results are shown for LLaDA-8B-Instruct and Dream-7B-Base.

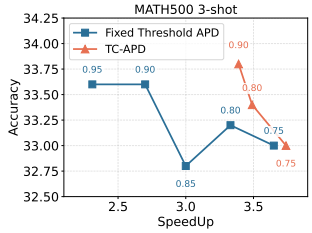


Figure 6: Few-shot transfer analysis on MATH500.

Table 2: Wall-clock inference efficiency on LLaDA-8B-Instruct. TPS denotes generated tokens per second, and speedup is measured relative to standard decoding.

Method	GSM8K		MATH500		HumanEval	
	TPS	Speedup ↑	TPS	Speedup ↑	TPS	Speedup ↑
Standard	24.8	1.00×	24.1	1.00×	21.0	1.00×
Fixed APD (0.90)	85.6	3.45×	60.8	2.52×	56.9	2.71×
TC-APD (0.90)	108.5	4.37×	73.1	3.03×	82.3	3.91×

with 47.7 for fixed-threshold APD. These results show that token-level calibration can improve the speed–quality trade-off beyond a shared global threshold.

For Dream-7B-Base, we provide the full results in Appendix B.1 due to space limitations and focus on LLaDA-8B-Instruct in the main paper. we further evaluate thresholds from 0.75 to 0.95 and visualize the speed–accuracy trade-off across different operating points for both models, as shown in Figure 5. Fixed-threshold APD shows a clear efficiency–accuracy trade-off as the threshold varies, whereas TC-APD generally shifts the trade-off toward lower computational cost at comparable accuracy. These results show that the benefit of token-level calibration extends across models and operating points.

Wall-clock efficiency. We also evaluate practical inference efficiency under the same hardware and decoding settings. As shown in Table 2, the reduction in NFE translates into higher throughput and real wall-clock acceleration. TC-APD achieves higher TPS and wall-clock speedup than fixed-threshold APD across all three benchmarks.

5.3 Ablation Studies

Table 3: Ablation over the empirical consistency target ρ and maximum threshold. Relaxing the consistency target increases speedup but may reduce generation accuracy.

Consistency Target ρ	Max Threshold = 0.8		Max Threshold = 0.9	
	Avg Acc	SpeedUp	Avg Acc	SpeedUp
1.000	47.7	3.69	47.8	3.35
0.995	47.7	3.91	48.0	3.73
0.990	46.2	4.40	46.9	4.28

We conduct ablations on the empirical consistency target ρ using LLaDA-8B-Instruct, evaluating $\rho \in \{1.0, 0.995, 0.990\}$ as shown in Table 3. A relaxed target of $\rho = 0.990$ yields higher speedup but reduces accuracy, while a stricter target of $\rho = 1.0$ provides no further accuracy gain and reduces speedup. Overall, $\rho = 0.995$ provides the best observed balance between accuracy and efficiency.

The remaining hyperparameters, $N_{\min}$ and $A_{\min}$, are minimum-support constraints used to avoid noisy estimates for rare tokens. They only determine whether a token has sufficient calibration data to use a token-specific threshold; otherwise, the default global threshold is used. We fix $N_{\min} = 200$ and $A_{\min} = 100$ in all experiments without further tuning.

5.4 Analysis

Few-shot generalization. We further evaluate whether thresholds calibrated from zero-shot traces transfer to a different prompting format. We directly apply the same threshold table to 3-shot MATH500 without recalibration. As shown in Figure 6, TC-APD retains a better speed–quality trade-off than fixed-threshold APD, showing that the calibrated thresholds can transfer across prompting settings.

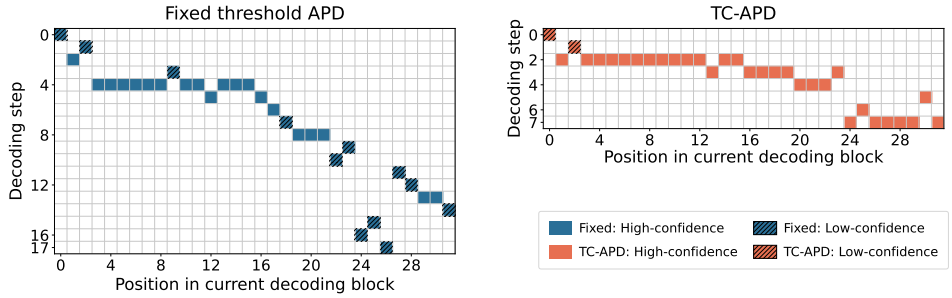


Figure 7: Decoding path comparison between fixed-threshold APD and TC-APD on the same example. Each cell denotes a token committed at a specific decoding step and block position.

Decoding path visualization. We compare fixed-threshold APD and TC-APD on one example to illustrate their decoding behavior. Fixed-threshold APD is delayed by a few low-confidence tokens, leading to more single-token steps. In contrast, TC-APD reduces low confidence single-token steps from 12 to 2 (-83%) and total decoding steps from 18 to 8 (-56%). The visualization is shown in Figure 7.

Plug-and-play integration. To further evaluate the compatibility of TC-APD, we integrate it into SOAR (Cao et al., 2026), which combines adaptive parallel decoding with position-level search. The integration requires only replacing SOAR's global APD threshold with our calibrated token-specific thresholds, while leaving the remaining decoding procedure unchanged. This simple modification further improves decoding efficiency while maintaining higher accuracy than standard decoding, demonstrating that token-level calibration can be incorporated into existing APD-based decoding strategies with minimal changes. Detailed results are provided in Appendix B.2.

Decoding behavior analysis. We further analyze how token-specific thresholds affect the actual decoding behavior of TC-APD. Compared with fixed-threshold decoding, TC-APD often follows different token-level trajectories while maintaining comparable task performance, indicating that earlier token commitment does not simply reproduce the trajectory of standard decoding. We also find that calibrated token-specific thresholds are widely used during inference rather than acting as a minor correction to the global threshold. Additional analyses of vocabulary-wide token-conditioned consistency, trajectory and final-answer changes, threshold coverage, and capping effects are provided in Appendix D.

6 Conclusion

We propose TC-APD, a simple and interpretable token-level threshold calibration method for adaptive parallel decoding in diffusion language models. By exploiting token-dependent confidence final-consistency relationships, TC-APD estimates token-specific thresholds from offline denoising traces and applies them through a lightweight lookup during inference. This enables more fine-grained token commitment decisions and improves the speed–quality trade-off without additional training or model forward passes. Experiments across two DLM backbones and multiple reasoning and code-generation benchmarks show lower decoding cost while maintaining comparable task performance. Overall, our results demonstrate that offline token-level threshold calibration is a simple and effective approach to improving DLM inference efficiency.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, language polishing, LaTeX formatting, and compilation and layout checks. All AI-assisted revisions were reviewed by the authors. The research questions, methodological design, experimental decisions, interpretation of results, and final claims were determined by the authors. The authors take full responsibility for the content of this work.

ETHICS STATEMENT

This work studies inference acceleration for diffusion language models and does not involve human subjects, private user data, or the collection of sensitive information. All experiments are conducted on publicly available benchmarks and existing pretrained language models. We do not identify additional ethical concerns specific to this work beyond those generally associated with the use and deployment of large language models.

REFERENCES

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. *arXiv preprint arXiv:2509.25188*, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

Mingyu Cao, Alvaro HC Correia, Christos Louizos, Shiwei Liu, and Lu Yin. Search or accelerate: Confidence-switched position beam search for diffusion language models. *arXiv preprint arXiv:2602.10953*, 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.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9868–9875, 2023.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *arXiv preprint arXiv:2507.18578*, 2025.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *Advances in neural information processing systems*, 38:52870–52888, 2026.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models, 2026. URL https://arxiv.org/abs/2511.05664.

Zahar Kohut, Severyn Shykula, Dmytro Khamula, Mykola Vysotskyi, Taras Rumezhak, and Volodymyr Karpiv. Just on time: Token-level early stopping for diffusion language models. *arXiv preprint arXiv:2602.11133*, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Miao Li, Hanyang Jiang, Sikai Cheng, Hengyu Fu, Yuhang Cai, Baihe Huang, Tinghan Ye, Xuanzhou Chen, and Pascal Van Hentenryck. Plan, verify and fill: A structured parallel decoding approach for diffusion language models, 2026. URL https://arxiv.org/abs/2601.12247.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*, 2025a.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*, 2025b.

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.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Jucheng Shen and Yeonju Ro. Beyond static cutoffs: One-shot dynamic thresholding for diffusion language models. *arXiv preprint arXiv:2511.02077*, 2025.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Xinhao Sun, Maoliang Li, Zihao Zheng, Jiayu Chen, Hezhao Xu, Yun Liang, and Xiang Chen. Stdd: Spatio-temporal dynamics-driven token refinement in diffusion language models. *arXiv preprint arXiv:2601.04205*, 2025.

Qwen Team. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lipeng Wan, Jianhui Gu, Junjie Ma, Jianguo Huang, Shiguang Sun, Siyuan Li, and Xuguang Lan. Progressive refinement regulation for accelerating diffusion language model decoding, 2026. URL https://arxiv.org/abs/2603.04514.

Kangyu Wang, Zhiyun Jiang, Haibo Feng, Weijia Zhao, Lin Liu, Jianguo Li, Zhenzhong Lan, and Weiyao Lin. CreditDecoding: Accelerating parallel decoding in diffusion large language models with trace credit. 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. 11105–11123, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.509. URL https://aclanthology.org/2026.acl-long.509/.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles. *arXiv preprint arXiv:2506.10848*, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

Tianyi Wu, Xiaoxi Sun, Yanhua Jiao, Yulin Li, Yixin Chen, YunHao Cao, YiQi Hu, and Zhuotao Tian. Dynamic-dllm: Dynamic cache-budget and adaptive parallel decoding for training-free acceleration of diffusion llm. *arXiv preprint arXiv:2606.26120*, 2026.

Fan Xu, Huixuan Zhang, and Xiaojun Wan. DecoCal: Decoding with calibration in diffusion large language models. 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. 11866–11880, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.545. URL https://aclanthology.org/2026.acl-long.545/.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Longxuan Yu, Shaorong Zhang, Yu Fu, Hui Liu, Yue Dong, and Greg Ver Steeg. Revise, don't freeze: Sampler-matched training for self-correcting masked diffusion language models. *arXiv preprint arXiv:2606.01026*, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, 2026.

A Calibration Data Experiments

In the main paper, we use offline calibrated thresholds to improve adaptive parallel decoding. Since these thresholds are estimated from a calibration set, an important question is how the choice of calibration data affects the resulting decoding behavior. Our goal is to obtain a single threshold table that generalizes across benchmarks, rather than calibrating separate thresholds for each evaluation task. We therefore study two questions:

- **Cross-domain transferability:** Can thresholds calibrated on one domain transfer to other benchmarks?
- **Mixed-domain calibration:** Does combining multiple domains lead to better calibration than using a single domain?

Table A1: Cross-domain calibration evaluation on LLaDA-8B-Instruct. We calibrate token-level thresholds using training data from different domains and evaluate the resulting thresholds on GSM8K, MATH500, and HumanEval. The default threshold and maximum threshold are both set to 0.9.

Calibration Data	GSM8K			MATH500			HumanEval			Average		
	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp
Standard decoding	72.9	256.0	x1.00	32.4	256.0	x1.00	38.4	256.0	x1.00	47.9	256.0	x1.00
GSM8K	72.3	60.6	x4.22	33.8	88.8	x2.88	36.6	77.3	x3.31	47.6	75.6	x3.39
Minerva-Math	72.1	56.4	x4.54	33.4	82.5	x3.10	38.4	66.9	x3.83	48.0	68.6	x3.73
ACE-Code	72.2	63.3	x4.04	33.8	90.2	x2.84	38.4	67.0	x3.82	48.1	73.5	x3.48

A.1 Cross-Domain Calibration Evaluation

We first evaluate whether thresholds calibrated from one domain can transfer to other benchmarks. Specifically, we use **training data** from GSM8K, Minerva-Math, and ACE-Code to independently estimate token-specific thresholds. To ensure a strict separation between calibration and evaluation, we remove all examples that overlap with the GSM8K, MATH500, or HumanEval evaluation sets before calibration. We then evaluate each resulting threshold table on GSM8K, MATH500, and HumanEval. For all experiments, both the default threshold and maximum threshold are set to 0.9. We use LLaDA-8B-Instruct as the backbone model, and keep all other inference and evaluation settings the same as in the main experiments.

As shown in Table A1, all calibrated variants substantially reduce NFE compared with standard decoding, showing that offline token-level threshold calibration consistently improves decoding efficiency. More importantly, we find that **calibration is not strictly domain-specific**: using the training data from the corresponding benchmark does not always produce the best result. For example, GSM8K-calibrated thresholds perform well on GSM8K, but do not achieve the best average performance across benchmarks. In contrast, Minerva-Math calibration achieves the lowest average NFE and the best average speedup, while ACE-Code calibration achieves the highest average accuracy. These results suggest that **token-level calibration can transfer across benchmarks**, and that calibration quality depends not only on domain matching but also on the statistical quality of the calibration traces.

A.2 Calibration Data Choice

We next compare different strategies for constructing the calibration set. The goal is to understand whether adding more diverse calibration data improves the learned token-level thresholds.

We consider two calibration strategies. The **single-domain** strategy uses 5,000 Minerva-Math training examples, which achieves the strongest overall speed-quality trade-off in Section A.1. The **mixed-domain** strategy combines 5,000 GSM8K examples, 5,000 Minerva-Math examples, and 5,000 ACE-Code examples, and estimates thresholds using the same empirical consistency procedure as in the main paper.

We conduct this analysis on LLaDA-8B-Instruct. Except for the calibration data used to estimate thresholds, all inference and evaluation settings are identical to the main experiments. Table A2 compares single-domain and mixed-domain calibration under different maximum threshold values.

The results show that mixed-domain calibration does not consistently outperform single-domain calibration. Although mixed-domain calibration uses more diverse data, it often requires higher NFE and achieves lower speedup than Minerva-Math calibration. For example, at maximum threshold 0.90, mixed-domain calibration obtains comparable average accuracy, but increases average NFE from 68.6 to 71.7 and reduces speedup from $3.73\times$ to $3.57\times$. Across most threshold settings, Minerva-Math calibration provides a stronger speed-quality trade-off, especially in decoding efficiency.

Overall, these results indicate that **calibration data quality and distributional alignment are more important than simply mixing more domains**. While token-level calibration can transfer across

Table A2: Calibration data choice analysis on LLaDA-8B-Instruct. We compare single-domain calibration using Minerva-Math training data with mixed-domain calibration using GSM8K, Minerva-Math, and ACE-Code. All inference and evaluation settings are the same as in the main experiments, except for the calibration data used to estimate token-level thresholds.

Default Threshold	GSM8K			MATH500			HumanEval			Average		
	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp
Standard decoding	72.9	256.0	x1.00	32.4	256.0	x1.00	38.4	256.0	x1.00	47.9	256.0	x1.00
Fixed threshold APD												
Max threshold 0.70	71.7	48.7	x5.26	32.8	73.2	x3.50	34.2	58.3	x4.39	46.2	60.1	x4.26
Max threshold 0.75	72.7	54.2	x4.72	32.6	79.8	x3.21	34.8	67.0	x3.82	46.7	67.0	x3.82
Max threshold 0.80	73.0	60.6	x4.22	33.2	87.8	x2.92	37.0	76.5	x3.35	47.7	75.0	x3.41
Max threshold 0.85	72.9	68.4	x3.74	32.8	96.7	x2.65	39.0	87.2	x2.94	48.2	84.1	x3.04
Max threshold 0.90	72.9	77.5	x3.30	32.4	106.0	x2.42	37.8	100.7	x2.54	47.7	94.7	x2.70
Max threshold 0.95	72.9	93.6	x2.74	33.0	122.0	x2.10	38.4	122.4	x2.09	48.1	112.7	x2.27
TC-APD Single-domain calibration: Minerva-Math												
Max threshold 0.70	71.4	47.0	x5.45	31.6	71.7	x3.57	34.2	54.4	x4.71	45.7	57.7	x4.44
Max threshold 0.75	72.3	50.7	x5.05	33.6	76.7	x3.34	34.8	59.2	x4.32	46.9	62.2	x4.12
Max threshold 0.80	72.3	53.8	x4.76	33.2	80.7	x3.17	37.6	62.1	x4.12	47.7	65.5	x3.91
Max threshold 0.85	72.4	55.3	x4.63	33.2	81.9	x3.13	39.0	64.4	x3.98	48.2	67.2	x3.81
Max threshold 0.90	72.1	56.4	x4.54	33.4	82.5	x3.10	38.4	66.9	x3.83	48.0	68.6	x3.73
Max threshold 0.95	72.5	58.0	x4.41	33.2	83.1	x3.08	38.4	70.1	x3.65	48.0	70.4	x3.64
TC-APD Mixed-domain calibration: GSM8K + Minerva-Math + ACE-Code												
Max threshold 0.70	71.8	48.0	x5.33	32.4	72.2	x3.55	34.2	55.4	x4.62	46.1	58.5	x4.37
Max threshold 0.75	72.6	53.0	x4.83	34.0	77.8	x3.29	34.8	60.9	x4.20	47.1	63.9	x4.01
Max threshold 0.80	72.3	58.0	x4.41	33.2	83.5	x3.07	37.8	66.1	x3.87	47.8	69.2	x3.70
Max threshold 0.85	72.4	59.9	x4.27	33.4	85.4	x3.00	37.8	68.4	x3.74	47.9	71.2	x3.59
Max threshold 0.90	72.3	60.4	x4.24	33.4	85.9	x2.98	38.4	68.8	x3.72	48.0	71.7	x3.57
Max threshold 0.95	72.3	60.8	x4.21	33.4	86.3	x2.97	38.4	69.3	x3.69	48.0	72.1	x3.55

benchmarks, the calibration set should be chosen carefully. Therefore, for LLaDA-8B-Instruct, we report the main results using thresholds calibrated on Minerva-Math, which provides the best overall speed-quality trade-off among the calibration settings we studied.

B Additional Evaluation

B.1 Evaluation on Dream-7B-Base

To verify the general applicability of token-level calibration beyond LLaDA, we further evaluate TC-APD on Dream-7B-Base. We also find that the choice of calibration data differs across backbones: unlike LLaDA, Dream benefits more from mixed-domain calibration than from single-domain calibration. This suggests that token-level confidence distributions can be backbone-dependent, potentially due to differences in pretraining data, training objectives, or denoising schedules.

As shown in Table A3, TC-APD also improves the speed-quality trade-off on Dream-7B-Base compared with fixed-threshold APD. We note that TC-APD shows a clear accuracy drop on GSM8K compared with standard decoding, especially under lower maximum thresholds. However, this drop is mainly caused by more aggressive parallel commitment, and similar trade-offs are also observed for fixed-threshold APD at comparable speedup levels. More importantly, when comparing TC-APD with fixed-threshold APD under similar NFE or speedup, TC-APD generally achieves a better speed-quality trade-off. For example, TC-APD at a maximum threshold of 0.90 achieves an average accuracy of 55.6 with 105.4 NFE, while fixed-threshold APD at 0.85 achieves a comparable accuracy of 55.5 with 121.5 NFE. These results support our main claim: offline token-level calibration can further improve adaptive parallel decoding beyond global fixed thresholds, while maintaining a better speed-quality trade-off.

Table A3: Results on Dream-7B-Base. We compare fixed-threshold APD and TC-APD using mixed-domain calibrated thresholds. We report accuracy, number of function evaluations (NFE), and NFE-based speedup. The final-consistency criterion is set to $\rho = 0.995$, and the number in parentheses denotes the maximum threshold.

Method	GSM8K			MATH500			HumanEval			Average		
	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp
Standard decoding	76.8	256.0	x1.00	38.2	256.0	x1.00	54.3	256.0	x1.00	56.4	256.0	x1.00
Fixed threshold APD												
Fixed APD (0.70)	74.3	127.2	x2.01	35.0	70.9	x3.61	49.4	104.4	x2.45	52.9	100.8	x2.54
Fixed APD (0.75)	75.3	132.6	x1.93	35.6	76.8	x3.33	51.2	110.4	x2.32	54.0	106.6	x2.40
Fixed APD (0.80)	75.6	139.1	x1.84	36.8	83.3	x3.07	51.2	118.3	x2.16	54.5	113.6	x2.25
Fixed APD (0.85)	76.0	147.2	x1.74	38.2	91.7	x2.79	52.4	125.6	x2.04	55.5	121.5	x2.11
Fixed APD (0.90)	76.7	154.5	x1.66	38.8	100.7	x2.54	53.6	136.0	x1.88	56.4	130.4	x1.96
Fixed APD (0.95)	76.6	167.5	x1.53	38.0	115.8	x2.21	54.3	150.5	x1.70	56.3	144.6	x1.77
TC-APD												
TC-APD (0.70)	74.1	122.7	x2.09	34.6	68.9	x3.72	48.7	102.3	x2.50	52.5	98.0	x2.61
TC-APD (0.75)	74.7	125.8	x2.03	36.0	72.1	x3.55	51.8	106.2	x2.41	54.2	101.4	x2.53
TC-APD (0.80)	75.5	127.9	x2.00	36.8	73.9	x3.46	53.0	108.3	x2.36	55.1	103.4	x2.48
TC-APD (0.85)	75.4	129.1	x1.98	36.8	74.6	x3.43	53.7	110.6	x2.31	55.3	104.8	x2.44
TC-APD (0.90)	75.6	129.7	x1.97	37.0	75.1	x3.41	54.3	111.4	x2.30	55.6	105.4	x2.43
TC-APD (0.95)	75.7	130.7	x1.96	37.2	75.3	x3.40	53.7	112.3	x2.28	55.5	106.1	x2.41

Table A4: Plug-and-play integration of TC-APD with SOAR on LLaDA-8B-Instruct. We replace only the global threshold in SOAR's adaptive parallel decoding branch with token-specific thresholds from TC-APD.

Method	GSM8K			MATH500			HumanEval			Average		
	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp	Acc	NFE	SpeedUp
Standard decoding	72.9	256.0	×1.00	32.4	256.0	×1.00	38.4	256.0	×1.00	47.9	256.0	×1.00
SOAR	73.9	71.6	×3.58	35.0	98.7	×2.59	38.2	90.0	×2.84	49.0	86.8	×2.95
SOAR + TC-APD	73.4	53.6	×4.78	33.4	77.5	×3.30	39.0	62.9	×4.07	48.6	64.7	×3.96

B.2 Plug-and-Play Integration with SOAR

To examine whether TC-APD can be used beyond standard adaptive parallel decoding, we integrate it with SOAR (Cao et al., 2026). SOAR combines adaptive parallel decoding with Position Beam Search (PBS): when sufficiently confident positions are available, they are decoded in parallel; otherwise, SOAR switches to PBS to explore alternative decoding positions.

The integration is simple. We leave the PBS branch and the remaining SOAR decoding procedure unchanged, and only replace the global confidence threshold in its parallel decoding branch with our token-specific thresholds. Specifically, the original parallel-decoding rule

$$\mathcal{I}_t = \{i : c_{t,i} > \tau\}$$

is replaced by

$$\mathcal{I}_t = \{i : c_{t,i} > \tau_{\hat{v}_{t,i}}\},$$

where $\tau_{\hat{v}_{t,i}}$ is obtained from the offline calibration table. Thus, TC-APD can be incorporated into SOAR through a simple threshold lookup without modifying its search mechanism or introducing additional model forward passes.

As shown in Table A4, SOAR improves average accuracy over standard decoding from 47.9 to 49.0 while achieving a $2.95\times$ NFE speedup. Adding TC-APD further reduces the average NFE from 86.77 to 64.67, increasing the speedup to $3.96\times$, while still achieving higher average accuracy than standard decoding. These results demonstrate that token-level calibration can serve as a plug-and-play component for more general APD-based decoding strategies.

C Experimental Details

C.1 Experimental Setup

We evaluate TC-APD on two diffusion language model backbones, LLaDA-8B-Instruct and Dream-7B-Base. For LLaDA-8B-Instruct, we use zero-shot evaluation on all benchmarks. For Dream-7B-Base, we use zero-shot evaluation on HumanEval and 3-shot evaluation on GSM8K and MATH500. All experiments are conducted on GSM8K, MATH500, and HumanEval.

Unless otherwise specified, all methods use block-wise diffusion decoding with maximum generation length 256 and block size 32. We evaluate all methods under the same benchmark protocol and report accuracy, number of function evaluations (NFE), and NFE-based speedup. We use NFE as the primary efficiency metric because it measures the number of model forward calls and is less affected by hardware-specific runtime variation than wall-clock time or tokens-per-second. Following the main experiments, we disable KV-cache for all methods to isolate the effect of the token commitment policy.

All reported results are from a single evaluation run. This is because our experiments focus on deterministic inference under fixed decoding settings, where accuracy and NFE are not expected to vary substantially across random seeds. In contrast, hardware-dependent metrics such as wall-clock time can fluctuate due to system-level factors. Therefore, we mainly rely on NFE-based speedup for efficiency comparison.

C.2 Hyperparameters

Table A5: Main hyperparameters used in our experiments.

Hyperparameter	Value
Maximum generation length	256
Block size	32
Final-consistency criterion target ρ	0.995
Minimum token observations $N_{\min}$	200
Minimum accepted observations $A_{\min}$	100
Default threshold for ablation τ_{default}	0.9

Table A5 summarizes the main hyperparameters used in our experiments. For TC-APD, the final-consistency criterion target is set to $\rho = 0.995$ by default, based on the ablation study in the main paper. We use $N_{\min} = 200$ and $A_{\min} = 100$ as support constraints to avoid unstable threshold estimates for rare tokens or thresholds with insufficient calibration samples. Unless otherwise specified, the default threshold is set to 0.9.

C.3 Baseline Implementation Details

We compare TC-APD with standard decoding, fixed-threshold APD, and existing acceleration methods for DLMs. All baselines, including WINO, Prophet, Dynamic-DLLM, OSDT, and Learn2PD, are evaluated using their official codebases. For fair comparison, we unify the inference setting across methods whenever applicable: all methods are evaluated with block-wise diffusion decoding, maximum generation length 256, and block size 32.

Standard decoding performs the full block-wise denoising process without adaptive early commitment. Fixed-threshold APD uses a single global confidence threshold for all tokens and commits positions whose confidence exceeds this threshold in parallel. If no position satisfies the threshold, it falls back to committing the highest-confidence token. TC-APD follows the same adaptive parallel decoding framework, but replaces the global threshold with token-specific calibrated thresholds. For a predicted token $\hat{v}_i$, TC-APD retrieves its calibrated threshold $\tau_{\hat{v}_i}$ from the threshold table. If the predicted token does not have a calibrated threshold, TC-APD falls back to the default threshold τ_{default}.

C.4 Calibration Cost

The calibration process requires running the base diffusion language model on a calibration set to collect token-level confidence trajectories and final-consistency statistics. For each intermediate denoising step, we record the top-1 predicted token, its confidence, and whether it matches the final decoded output. These records are then used to estimate token-specific thresholds.

On four NVIDIA A100 GPUs with batch size 1, calibrating thresholds using 5,000 samples from a single domain takes approximately 3.5 hours. When multiple domains are used, the calibration time increases approximately linearly with the number of calibration samples. Due to limited computational resources, we do not exhaustively study the effect of calibration set size in this work. However, the use of 5,000 samples is not a strict requirement of TC-APD; it is a practical setting that provides sufficient token-level statistics under our support constraints.

This cost is incurred only once before deployment. After calibration, inference only requires a lightweight token-id lookup to retrieve the corresponding threshold, without introducing additional neural modules, online optimization, or extra model forward passes. Therefore, TC-APD introduces an offline calibration cost but has negligible inference-time overhead, making it suitable for deployment scenarios where the same DLM is repeatedly used for large-scale inference.

C.5 Inference Algorithm

In the main paper, we describe the inference procedure of TC-APD conceptually. Here, we provide the detailed algorithm for calibrated confidence-aware adaptive decoding within a single block. Given a pre-computed token-level threshold table, TC-APD replaces the global threshold used in conventional adaptive parallel decoding with token-specific thresholds. At each decoding step, the model predicts the top-1 token and its confidence for every masked position. The corresponding token-specific threshold is retrieved from the calibration table if available; otherwise, the default threshold is used. All positions whose confidence exceeds their thresholds are committed in parallel. If no position satisfies its threshold, TC-APD falls back to committing the single highest-confidence position.

Algorithm A1 summarizes the complete decoding procedure.

Algorithm A1 Calibrated Confidence-Aware Adaptive Decoding for a Single Block

Require: Prompt $\mathbf{x}$, block size B, threshold table $\{\tau_v\}_{v\in\mathcal{V}}$, default threshold τ_{default}

Initialize B masked positions
while the current block is not fully decoded **do**
  Run the DLM to obtain top-1 predictions $\hat{v}_i$ and confidences c_i for all masked positions
  for each masked position i **do**
    Retrieve $\tau_i \leftarrow \tau_{\hat{v}_i}$ if available; otherwise $\tau_i \leftarrow \tau_{\text{default}}$
  end for
  $\mathcal{S} \leftarrow \{i : c_i \geq \tau_i\}$
  if $\mathcal{S} \neq \emptyset$ **then**
    Commit $\hat{v}_i$ for all $i \in \mathcal{S}$ in parallel
  else
    $i^\star \leftarrow \arg\max_i c_i$
    Commit $\hat{v}_{i^\star}$
  end if
end while
return Decoded block

D TC-APD Decoding Behavior

D.1 Vocabulary Wide Token-Conditioned Consistency

In Section 3, we show representative examples illustrating that tokens can exhibit different final-consistency rates even at similar confidence levels. We further examine whether this phenomenon holds broadly across the vocabulary rather than only for selected tokens.

Table A6: Vocabulary-wide variation in token-level final consistency on GSM8K. For each confidence bin, we report the overall consistency rate and the distribution of token-specific consistency rates among tokens with at least 20 observations in that bin.

Confidence	# Tokens	Overall	P10	Median	P90	IQR
$[0.50, 0.55)$	645	70.3	48.0	68.5	83.3	16.4
$[0.60, 0.65)$	591	82.8	66.7	83.3	95.8	13.3
$[0.70, 0.75)$	548	91.0	81.5	91.9	100.0	8.9
$[0.80, 0.85)$	612	96.2	89.3	97.7	100.0	5.0
$[0.90, 0.95)$	913	99.1	98.2	100.0	100.0	0.0

Using standard decoding traces on GSM8K, we group intermediate top-1 predictions into confidence bins of width 0.05. Within each bin, we compute the final-consistency rate separately for each token. To avoid noisy estimates in this analysis, we only include tokens with at least 20 observations in the corresponding confidence bin. This support criterion is used only for this diagnostic analysis and is independent of the calibration hyperparameters $N_{\min}$ and $A_{\min}$.

Table A6 reports the distribution of token-level final-consistency rates for several representative confidence ranges. The results show substantial token-level variation even within narrow confidence intervals. For example, for predictions with confidence in $[0.70, 0.75)$, the overall final-consistency rate is 91.0%, while the 10th and 90th percentiles across 548 sufficiently observed tokens are 81.5% and 100.0%, respectively. The variation is even larger at moderate confidence levels: within $[0.50, 0.55)$, token-level final-consistency rates range from 48.0% at the 10th percentile to 83.3% at the 90th percentile.

These results provide vocabulary-wide evidence that confidence alone does not fully determine prediction stability. In particular, two predictions with similar confidence can have substantially different empirical consistency depending on the predicted token, motivating token-conditioned threshold calibration.

D.2 Decoding Trajectory and Final-Answer Analysis

We further analyze how adaptive parallel decoding changes the generation trajectory relative to standard decoding. We emphasize that matching the standard decoding trajectory is not an objective of TC-APD: the method uses offline final consistency as a calibration signal for estimating token-specific thresholds, while its final performance is evaluated by task accuracy and decoding efficiency. We use standard decoding here only as a diagnostic reference for understanding how different commitment policies affect generation.

Final-answer agreement. We first compare the final answers produced by accelerated decoding with those from standard decoding on all 1,319 GSM8K test examples. We use the same `lm-eval-harness` flexible answer extraction and normalization procedure as in our main evaluation. As shown in Table A7, fixed-threshold APD at 0.90 produces the same final answer as standard decoding on 96.5% of examples. TC-APD produces the same final answer on 84.4% and 85.7% of examples at maximum thresholds 0.80 and 0.90, respectively.

Despite these differences in generated answers, task accuracy changes only slightly. To better understand this behavior, we separate changes into correct-to-incorrect (C→I) and incorrect-to-correct (I→C) transitions relative to standard decoding. For TC-APD at 0.90, 69 standard-decoding-correct examples become incorrect, while 59 standard-decoding-incorrect examples become correct. The net difference is therefore only 10 examples out of 1,319, corresponding to the small change from 72.9% to 72.1% accuracy.

This analysis indicates that changes induced by earlier parallel commitment are bidirectional rather than uniformly detrimental. TC-APD can follow different generation trajectories and produce different answers on some examples, while maintaining comparable aggregate task performance.

Token-level trajectory agreement. We additionally compare the effective generated token sequences before the terminal token. As shown in Table A8, TC-APD has substantially lower token-

Table A7: Final answer and correctness transition analysis on GSM8K. Answer agreement is the fraction of examples whose extracted final answer matches standard decoding. C→I and I→C indicate correctness changes relative to standard decoding.

Method	Acc.	NFE	Answer Agree.	C→I	I→C
Fixed APD (0.80)	73.0	60.6	89.5	44	46
Fixed APD (0.90)	72.9	77.5	96.5	17	17
TC-APD (0.80)	72.3	53.8	84.4	74	66
TC-APD (0.90)	72.1	56.4	85.7	69	59

Table A8: Agreement with standard decoding on GSM8K after truncating each generation at its terminal token. Token agreement measures position-wise token identity, while exact agreement requires the entire effective generated sequence to match. These metrics are diagnostic and are not optimization objectives of TC-APD.

Method	Token Agree.	Exact Seq. Agree.
Fixed APD (0.80)	79.3	45.8
Fixed APD (0.90)	94.8	80.1
TC-APD (0.80)	59.5	20.7
TC-APD (0.90)	60.1	21.7

level agreement with standard decoding than fixed-threshold APD. For example, at a maximum threshold of 0.90, TC-APD achieves 60.1% token-level agreement and 21.7% exact-sequence agreement, compared with 94.8% and 80.1% for fixed-threshold APD.

This result is consistent with the more aggressive commitment behavior enabled by token-specific thresholds: committing tokens earlier can alter subsequent denoising states and therefore lead to a different generation trajectory. Importantly, trajectory agreement should not be interpreted as task accuracy or as the calibration objective of TC-APD. As shown above, substantially different token-level trajectories can still lead to the same final answer or to a correct alternative answer. We therefore use this analysis only to characterize the effect of token-level threshold calibration on the decoding process.

D.3 Threshold Coverage and Capping Diagnostics

We next examine how frequently the calibrated threshold table is used during actual TC-APD decoding and how often token-specific thresholds are affected by the maximum-threshold cap. This analysis is performed on GSM8K using the same Minerva-Math calibrated threshold table as in the main experiments.

Table A9 reports candidate-level threshold coverage and capping statistics. Approximately 86% of candidate predictions correspond to tokens with a calibrated token-specific threshold, while the remaining 14% use the default threshold because the predicted token does not satisfy the calibration support constraints. This shows that the calibrated table covers the large majority of token predictions encountered during inference.

The effect of threshold capping depends strongly on the selected operating point. With a maximum threshold of 0.90, only 0.25% of candidate predictions and 0.26% of committed tokens use a threshold that has been clipped by the maximum threshold. Thus, capping has negligible effect at our default 0.90 operating point. In contrast, when the maximum threshold is reduced to 0.80, 33.9% of candidate predictions and 29.8% of committed tokens are affected by the cap, reflecting the more aggressive decoding regime induced by the lower maximum threshold.

Paired uncertainty analysis. Because decoding is deterministic under our evaluation setting, we do not repeat inference with different random seeds. Instead, we quantify example-level uncertainty using paired bootstrap resampling over the 1,319 GSM8K test examples. Relative to fixed-threshold APD, TC-APD reduces NFE by 6.86 at the 0.80 operating point (95% CI: $[-7.22, -6.50]$) and by 21.11 at the 0.90 operating point (95% CI: $[-21.62, -20.61]$). This confirms that the observed

Table A9: Threshold-table coverage and capping statistics on GSM8K. Token-specific coverage denotes the fraction of candidate predictions whose predicted token has a calibrated threshold; the remaining predictions use the default threshold. "Capped candidates" and "capped commits" denote the fractions for which the calibrated threshold exceeds the selected maximum threshold before clipping.

$\tau_{\max}$	Token-specific	Default	Capped candidates	Capped commits
0.80	86.7	13.3	33.9	29.8
0.90	86.0	14.0	0.25	0.26

efficiency improvements are consistent across evaluation examples rather than being driven by a small subset of inputs.