Fast Diffusion LLMs with Adaptive Routing

Anonymous ACL submission

Abstract

Discrete diffusion language models have emerged as a compelling alternative to autoregressive models by leveraging parallel generation to achieve significant speedups during inference. However, their generation procedure relies on iterative denoising, requiring many forward passes to transform a fully masked sequence into clean text. Existing acceleration approaches typically apply a fixed reverse-time step size across the entire denoising trajectory, missing the opportunity to spend more compute on contextually difficult states and less on easy ones. We propose ARFI (Adaptive Routing for Fast Inference), a training-free inference-time scheduler that converts the reverse-time step size from a global hyperparameter into a local, state-dependent decision. At each denoising step, ARFI reads the model's own per-position confidence distribution over masked tokens and selects how many tokens to commit using a confidence-gated piecewise rule—spending aggressively when confidence is high and conserving compute when it is not. We evaluate ARFI on GSM8K with LLaDA-8B-Base both standalone and paired with a standard behavioral-cloning distillation adapter, to isolate the contribution of adaptive scheduling from that of distillation. ARFI alone achieves 72.2% EM at 173 mean forward passes versus 72.9% at 256 for the fixed baseline, reducing step count by 32% with minimal accuracy loss. For reproducibility, we provide the code in an anonymized repository at https://anonymous.4open.science/r/ar-fi.

1 Introduction

Discrete diffusion language models (Austin et al., 2021; Lou et al., 2024; Nie et al., 2025) have recently emerged as a competitive alternative to standard autoregressive language models. Rather than generating tokens left-to-right, these models iteratively denoise a fully masked sequence in parallel, decoupling generation length from the number of sequential model evaluations. This leads to massive speedups in text generation, especially at higher text lengths. However, the high fidelity of generated text relies on running the denoiser for many iterative steps (Nie et al., 2025). This leads to the *number* of steps becoming the new latency bottleneck, particularly under constrained GPU memory where the cost per forward pass is already high.

The standard response to this bottleneck has been progressive distillation (Salimans and Ho, 2022; Meng et al., 2023; Song et al., 2023), in which a student model is trained to traverse the reverse-time diffusion trajectory in fewer, larger steps than the teacher. By learning to predict the outcome of several teacher iterations at once, the student dramatically reduces the required number of forward passes. Related work for discrete diffusion specifically has explored block-wise generation (Arriola et al., 2025), KV-cache approximations (Wu et al., 2025), and consistency-style objectives (Kim et al., 2025). However, these and current distillation protocols apply a fixed step size across the entire generation process. This formulation forces a compromise: the step size must be calibrated conservatively to preserve quality on difficult intermediate states, even though many other states could tolerate much larger jumps without loss. As a result, a substantial fraction of inference-time compute is wasted on intermediate steps that contribute little to the final token distribution, while structurally complex states are under-served by overly coarse jumps.

Our central observation is that, in a masked-diffusion language model, the model itself already exposes a strong signal about how aggressive the next step should be: the predicted token confidences over masked positions. When the per-position confidence distribution is high even at its lower quantiles, committing many tokens at once closely approximates what the model would commit over many small steps. When high confidence is concentrated only in a few positions while the

bulk of the masked sequence remains uncertain, large jumps force committing low-confidence predictions that cannot later be revised. Standard low-confidence remasking (Nie et al., 2025) already uses this signal to choose which tokens to commit per step, but commits a fixed schedule of tokens per step. We propose to use the same signal to choose how many tokens to commit per step.

This paper introduces ARFI (Adaptive Routing for Fast Inference), a training-free inference-time scheduler. Our contribution is:

- **ARFI (scheduler)**: An adaptive step-size scheduler (Graves, 2016; Schuster et al., 2022) that, at each denoising iteration, computes summary statistics of the model's confidence distribution over masked positions and selects one of several predefined step-size multipliers. The scheduler requires no additional training and reduces to the standard fixed schedule when confidence is uniformly low.

We additionally evaluate ARFI paired with a standard behavioral-cloning distillation adapter (ARFI (+expert)) to characterize how adaptive scheduling interacts with distillation.

Section 2 describes the scheduler, the distillation setup used in the ARFI (+expert) configuration, and the dispatch rule. Section 3 evaluates both configurations on GSM8K with LLaDA-8B-Base (Nie et al., 2025).

2 Methodology

Motivation. Committing the top-k most confident masked positions in one parallel step incurs a divergence from the exact sequential commit bounded by $k \cdot h(\tau)$, where τ is the confidence floor of the committed set and h vanishes as $\tau \to 1$ (Appendix D). Because the committed floor is the k-th largest masked confidence, the 25th percentile $q^{(i)}_{0.25}$ lower-bounds it whenever a step commits at most 75% of the remaining positions; the scheduler therefore gates the step size on $q^{(i)}_{0.25}$ and the mean $\bar{c}^{(i)}$ as a conservative proxy for the worst-case commit cost. The bound concerns distributions rather than the greedy decode we run, so it motivates rather than guarantees the rule.

2.1 Confidence-Conditioned Scheduler

Given the current state $x^{(i-1)}$ at denoising step i, the base denoiser produces logits $\ell^{(i)} \in \mathbb{R}^{L \times |\mathcal{V}|}$. Let $M^{(i)} \subseteq \{1, \ldots, L\}$ be the set of currently masked positions and let $p^{(i)} = \mathrm{softmax}(\ell^{(i)})$. We define the per-position confidence over masked positions as

$$c^{(i)}_j = \max_{v \in \mathcal{V}} p^{(i)}_{j,v}, \qquad j \in M^{(i)}, \tag{1}$$

and compute two summary statistics: the mean confidence $\bar{c}^{(i)} = \frac{1}{|M^{(i)}|} \sum_{j \in M^{(i)}} c^{(i)}_j$ and the 25th percentile $q^{(i)}_{0.25}$.

Transfer count rule. Let $b^{(i)} = \lceil |M^{(i)}| / (N_b - i + 1) \rceil$ denote the *base transfer count* that the fixed-step schedule would commit at step i within a denoising block of N_b steps. The scheduler chooses a transfer count $k^{(i)}$ as a piecewise-constant function of the two statistics:

$$k^{(i)} = \begin{cases} \lceil s_H \cdot b^{(i)} \rceil & \text{if } \bar{c}^{(i)} \geq \tau^{\bar{c}}_H \text{ and } q^{(i)}_{0.25} \geq \tau^{q}_H, \\ \lceil s_M \cdot b^{(i)} \rceil & \text{else if } \bar{c}^{(i)} \geq \tau^{\bar{c}}_M \text{ and } q^{(i)}_{0.25} \geq \tau^{q}_M, \\ b^{(i)} & \text{otherwise.} \end{cases} \tag{2}$$

The thresholds $\tau^{\bar{c}}_M, \tau^{\bar{c}}_H, \tau^{q}_M, \tau^{q}_H$ and scale factors s_M, s_H are hyperparameters fixed across all experiments (values in Appendix C). The transfer count is clipped to lie in $[1, |M^{(i)}|]$. On the final step of a block, all remaining masked positions are committed regardless of confidence.

The scheduler requires no learned parameters and only a single additional pass over the confidence vector per step (negligible compared to the cost of the denoiser forward pass). When all of the model's confidences are uniformly low, the rule reduces to the fixed schedule.

Selecting which positions to commit. After choosing $k^{(i)}$, the positions to commit are the top-$k^{(i)}$ confident masked positions within the current block, identical to the standard low-confidence remasking rule. The scheduler thus changes only *how many* tokens are committed per step, not *which* tokens.

2.2 Distillation Adapter (ARFI (+expert) Configuration)

To evaluate how ARFI interacts with a distilled model, we train a standard behavioral-cloning adapter following Salimans and Ho (2022): a LoRA module (Hu et al., 2022) that predicts the result of running the frozen base teacher for k steps in a single forward pass. This distillation objective is not a contribution of this work; we include it solely to produce the ARFI (+expert) configuration

and isolate adaptive scheduling from distillation. The compression factor k and all training hyperparameters are listed in Appendix C.

Constructing teacher targets. Given a training example (c, y) with prompt c and target y, we first tokenize the few-shot-formatted prompt and target and concatenate them into a full sequence of length L. We then construct a partially masked input by sampling a mask ratio $r \sim \mathrm{Uniform}(0.30, 0.70)$, drawing $\lfloor rL_y \rfloor$ positions uniformly from the answer span (where L_y is the answer length), and replacing them with `[MASK]`. The resulting masked input x corresponds to a state the base model might encounter at an intermediate point of a denoising trajectory.

We then run the frozen teacher for k denoising steps using low-confidence remasking, with the per-step transfer count determined by the standard fixed schedule. Let $\tilde{x}$ be the resulting state. The set of positions that the teacher committed is $C_x = \{j : x_j = m \wedge \tilde{x}_j \neq m\}$. The training target is the teacher's committed assignment $\tilde{x}_{C_x}$, and the loss is masked cross-entropy:

$$\mathcal{L}(\phi) = -\sum_{j \in C_x} \log g_\phi(\tilde{x}_j \mid x, c)_j. \tag{3}$$

All other positions are masked out of the loss. Full adapter architecture and training hyperparameters are listed in Appendix C; distillation data is drawn from the GSM8K training split with one masked state per example.

2.3 Inference-Time Dispatch

At inference, ARFI runs the same outer loop as standard masked-diffusion sampling: the sequence is divided into blocks of length b, and each block is denoised with N_b inner steps before moving to the next. Algorithm 1 describes the dispatch rule.

The dispatch threshold ρ and compression factor k are listed in Appendix C; the compression-factor multiplier (line 5) credits each expert call as k teacher steps. Reported step counts increment by one for every base or expert forward pass.

Algorithm 1 One denoising step of ARFI (ARFI (+expert) configuration; for ARFI (scheduler), the if-branch is skipped)

1: Compute remaining mask ratio $r = |M^{(i)} \cap \text{block}|/b$
2: **if** $r > \rho$ **then**
3: Forward expert g_ϕ on current sequence
4: Compute confidences from expert logits
5: $k \leftarrow \text{scheduler}(\bar{c}, q_{0.25}) \cdot$ compression factor
6: **else**
7: Disable adapter; forward base teacher
8: Compute confidences from base teacher logits
9: $k \leftarrow \text{scheduler}(\bar{c}, q_{0.25})$
10: **end if**
11: Commit top-k confident masked positions

3 Experiments

3.1 Setup

Setup. We evaluate on GSM8K (Cobbe et al., 2021) using a random 1,000-example subset of the standard test split with a 4-shot prompt from the LLaDA evaluation recipe (Nie et al., 2025). The base model is LLaDA-8B-Base (Nie et al., 2025); the expert is a LoRA adapter (rank 8) trained as described in Section 2.2. Generation length, denoising steps, and block length are all 256; temperature and CFG scale are 0 (full hyperparameter listing in Appendix C). We report exact-match accuracy (EM) and denoising forward passes and wall-clock time (mean/median); all timing is on a single NVIDIA A30 GPU at batch size 1.

Baselines. We compare four configurations:

- **Base (fixed)**: unmodified base model at 256 fixed steps.
- **Base (matched steps)**: base model at 173 and 62 fixed steps, controlling for compute budget.
- **ARFI (scheduler)**: ARFI on the base model
- **ARFI (+expert)**: ARFI with a distilled model

3.2 Results and Discussion

Table 1 and Figure 1 characterize the compute–quality tradeoff of ARFI.

ARFI (scheduler) reduces compute substantially with low accuracy cost. *ARFI (scheduler)* uses 173 mean forward passes versus 256 (32% reduction) for the fixed-step baseline while maintaining 72.2 EM. The model's own confidence distribution carries enough signal about local step optimality to safely accelerate a substantial fraction of steps without a learned router.

Method	EM (%)	Mean steps	Median steps	Mean sec/ex	Median sec/ex	Total sec
Base (fixed, 256 steps)	72.90	256.00	256.00	85.51	85.30	85,513.00
Base (matched, 173 steps)	69.30	173.00	173.00	55.88	55.76	55,878.53
Base (matched, 62 steps)	50.30	62.00	62.00	20.01	19.96	20,006.36
ARFI (scheduler)	72.20	173.10	174.00	56.15	56.34	56,151.38
ARFI (+expert)	54.20	62.15	57.00	21.20	19.42	21,201.80

Table 1: Accuracy and efficiency on GSM8K (1,000 examples). EM is exact-match accuracy. Timing results are from a single NVIDIA A30 GPU with batch size 1.

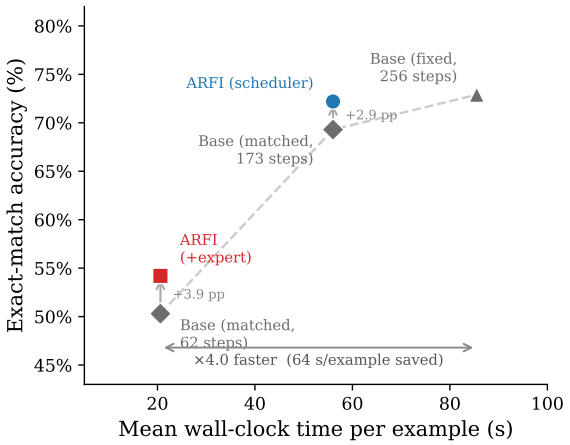

Figure 1: Compute–quality tradeoff on GSM8K (1,000 examples). The x-axis is mean wall-clock time per example (seconds); the y-axis is exact-match accuracy. The dashed curve connects the three fixed-step baselines and shows what standard fixed-step sampling achieves at each time budget. Arrows mark the EM gain from switching to an adaptive method at the same wall-clock cost: *ARFI (scheduler)* gains $+2.9$ pp over the 173-step fixed baseline at $\approx$56 s, and *ARFI (+expert)* gains $+3.9$ pp over the 62-step baseline at $\approx$21 s. Up and to the left is better.

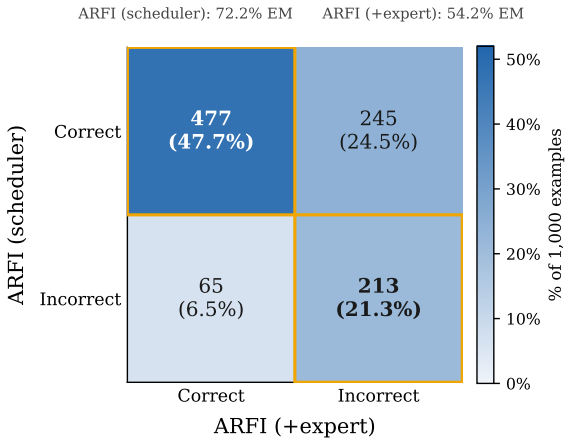

Figure 2: Per-example outcome heatmap comparing *ARFI (scheduler)* and *ARFI (+expert)* on 1,000 GSM8K examples. Bold borders mark the agreement diagonal. Of the 30% of examples where outcomes differ, *ARFI (scheduler)* uniquely solves 245 (24.5%) while *ARFI (+expert)* uniquely solves only 65 (6.5%)—a 3.8:1 asymmetry (analyzed in Section 3.2).

Adding distillation reduces steps further but hurts accuracy. Pairing ARFI with the behavioral-cloning adapter (ARFI (+expert)) reduces the mean step count from 173 to 62—a further 64% reduction—and cuts mean wall-clock time from 56.15 to 21.20 seconds per example (62.2% total speedup over the fixed baseline). However, EM drops from 72.2 to 54.2 (-18 pp). Of the examples where the two configurations disagree, ARFI (scheduler) uniquely solves 245 (24.5%) while ARFI (+expert) uniquely solves only 65 (6.5%)—a 3.8:1 asymmetry indicating the distillation adapter primarily damages moderate-difficulty problems the scheduler already handles correctly, not only the hardest ones.

4 Conclusion

We presented ARFI, a training-free confidence-conditioned step-size scheduler for masked diffusion language models. ARFI exploits the observation that the model's own predictive confidence over masked positions is a tractable proxy for the optimal local step size, replacing the static denoising schedule with a state-dependent one at zero training cost. Our experiments on GSM8K show that ARFI alone recovers near-baseline accuracy at 32% fewer forward passes. Pairing ARFI with a standard behavioral-cloning distillation adapter buys further speedup but at a steep accuracy cost, revealing that distillation and adaptive scheduling carry markedly different risk profiles. We view this work as a starting point for moving the acceleration of masked-diffusion inference beyond globally fixed schedules. Future work will focus on learned routers, soft-target distillation objectives, and evaluation on broader benchmarks.

Limitations

Our evaluation covers a single benchmark (GSM8K) and a single base model (LLaDA-8B-Base). We have not yet evaluated on broader instruction-following or coding benchmarks, nor on the Instruct or 1.5 variants of LLaDA, so the generality of the confidence-conditioned scheduler beyond mathematical reasoning is untested. The distillation adapter in the ARFI (+expert) configuration is trained only on GSM8K data, so its in-domain bias may inflate the observed benefit of that configuration; out-of-domain generalization is an open question. The scheduler thresholds are set by hand on a small validation subset, and a learned router conditioned on the same statistics could likely improve the compute–quality tradeoff. Finally, all timing experiments use a single A30 GPU at batch size 1; behavior under larger batches, on different hardware, and in the presence of KV-cache approximations (Wu et al., 2025) is left to future work.

References

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *Preprint*, arXiv:1603.08983.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. 2025. CDLM: Consistency diffusion language models for faster sampling. *Preprint*, arXiv:2511.19269.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32819–32848. PMLR.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2023. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14297–14306.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *Preprint*, arXiv:2502.09992.

Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR.

Satosi Watanabe. 1960. Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, 4(1):66–82.

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

A Artifact Documentation

Licenses for existing artifacts. Our experiments build on three primary external artifacts. **LLaDA-8B-Base** (Nie et al., 2025) is released under the MIT License via Hugging Face (`GSAI-ML/LLaDA-8B-Base`) and is loaded using the standard `transformers` API. **GSM8K** (Cobbe et al., 2021) is released under the MIT License; we use the standard test split for evaluation and a portion of the training split for expert distillation. The **LLaDA inference utilities** on which our evaluation harness is based (`generate.py` and supporting helpers) are distributed as part of the LLaDA repository under the MIT License. The Python packages used throughout (`transformers`,

`peft`, `datasets`, `torch`) are distributed under the Apache 2.0 or BSD licenses.

Consistency with intended use. LLaDA-8B-Base was released specifically to support research into discrete diffusion language models, including inference-time adaptation and fine-tuning; our use as a backbone for the ARFI scheduler and LoRA expert is consistent with this intent. GSM8K was released as a public benchmark for multi-step mathematical reasoning research; our use for evaluation and expert distillation training falls entirely within this scope. We do not repurpose any of these artifacts for commercial use or for tasks beyond the research evaluation described in this paper.

Intended use of our artifacts. We release the ARFI source code—comprising the confidence-conditioned scheduler, expert dispatch logic, distillation training script, and evaluation harness—under the MIT License. The anonymized code submitted as a supplement to this paper is intended to support reproducibility of the results reported in Section 3 and to facilitate follow-on research on inference-time acceleration for masked diffusion language models. The MIT License is compatible with the licenses of all artifacts on which our code depends (LLaDA-8B-Base: MIT; GSM8K: MIT; `transformers`/`peft`: Apache 2.0). The trained LoRA expert adapter is a derivative of GSM8K training data, which was accessed for research purposes; accordingly, the adapter is intended for research use only and should not be deployed outside of research contexts. We do not anticipate significant dual-use risks for these artifacts, as they are specific to discrete diffusion inference acceleration and do not introduce new capabilities not already present in the underlying LLaDA-8B-Base model.

B LLM Usage

We used LLMs to revise our paper, check for inconsistencies, and ensure that all checklist items were covered.

C Hyperparameter Details

For full reproducibility, we list all hyperparameters used in our experiments.

Scheduler. Mean confidence thresholds $\tau_M^{\bar{c}} = 0.12$ and $\tau_H^{\bar{c}} = 0.20$; 25th-percentile confidence thresholds $\tau_M^q = 0.045$ and $\tau_H^q = 0.060$; transfer-count scale factors $s_M = 2.0$ and $s_H = 4.0$. These were tuned by small-grid search on a held-out subset of the GSM8K training split.

Expert dispatch. Mask-ratio threshold $\rho = 0.35$; compression factor $k = 4$.

Expert training. LoRA rank $r = 8$, $\alpha = 16$, dropout 0.05. Target modules: attention $q/k/v/o$ projections and MLP up/down/gate projections. Base model loaded in 4-bit NF4 with bfloat16 compute dtype. Optimizer AdamW with learning rate 2×10^{-4}, weight decay 0. Effective batch size 16 via gradient accumulation. One epoch. Gradient checkpointing enabled. Maximum sequence length 4096.

Distillation data construction. Mask ratio sampled from $\mathrm{Uniform}(0.30, 0.70)$. One masked state per training example. Eval fraction 0.05 of the training split, capped at 256 examples. Random seed 1234 shared across distillation and inference for reproducibility.

Inference. All sequences are generated at a fixed length of 256 tokens. The denoising budget is 256 steps, meaning the fixed-schedule baseline runs exactly 256 forward passes; adaptive configurations (ARFI (scheduler), ARFI (+expert)) terminate early and use fewer. Block length is 256, matching the generation length so that the entire sequence is processed as a single block. Temperature is 0 (greedy argmax selection at each position) and classifier-free guidance scale is 0 (no conditional amplification). At each step, the top-k highest-confidence masked positions are committed and the remainder stay masked.

D Supplemental Theory

We give a single bound that motivates using a lower confidence quantile as the gating statistic in Equation (2).

Setup. At step i, let $S \subseteq \mathcal{M}^{(i)}$ with $|S| = k$ be the positions the scheduler commits, and let $c_j^{(i)} = \max_v p_\theta(x_j = v \mid x^{(i-1)})$ be the per-position mode probability. Compare two distributions over the committed tuple x_S: the product of marginals $p_{\text{batch}} = \prod_{j \in S} p_\theta(x_j \mid x^{(i-1)})$, which is what a single forward pass implicitly samples from, and the joint p_{seq} obtained by committing positions one at a time with context updates.

KL cost of simultaneous commits Let $\tau = \min_{j \in S} c_j^{(i)}$ and assume $\tau \geq 1/|\mathcal{V}|$. Then

$$D_{\mathrm{KL}}(p_{\mathrm{seq}} \,\|\, p_{\mathrm{batch}}) \;\leq\; k \cdot h(\tau), \tag{4}$$

where $h(\tau) = -\tau \log \tau - (1-\tau) \log \frac{1-\tau}{|\mathcal{V}|-1}$ is the maximum entropy of a distribution over $\mathcal{V}$ whose mode has probability τ.

Proof. The KL from a joint to the product of its marginals equals the total correlation (Watanabe, 1960): $D_{\mathrm{KL}}(p_{\mathrm{seq}} \| p_{\mathrm{batch}}) = \sum_{j \in S} H(x_j \mid x^{(i-1)}) - H(x_S \mid x^{(i-1)}) \leq \sum_{j \in S} H(x_j \mid x^{(i-1)})$. Each marginal has mode probability at least τ, so its entropy is at most $h(\tau)$, giving the bound. □

Why the lower quantile. The bound is tightest when τ is large. Since the scheduler commits the top-k confident positions, τ equals the k-th largest confidence in $\mathcal{M}^{(i)}$, which is bounded below by $q_{0.25}^{(i)}$ whenever $k \leq 0.75 \cdot |\mathcal{M}^{(i)}|$ (satisfied in our experiments except on terminal block steps, where all positions are committed unconditionally). Equation (2) therefore uses $q_{0.25}^{(i)}$ as a computable proxy for τ: a high lower quantile certifies a small KL cost for any aggressive step the rule selects. $\bar{c}^{(i)}$ is included as a complementary smoother that suppresses single-position noise.