TIPS: A TINY INFERENCE-TIME POLICY FOR SCALING IN DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models expose an inference-time choice: which positions become visible after each denoising pass. Many existing decoding methods use hand-designed rules or require visibility to grow monotonically. We present a **T**iny **I**nference-time **P**olicy for **S**caling (**TIPS**), an approximately 600K-parameter controller that learns both unmasking and remasking while keeping the underlying language model frozen. TIPS uses visibility-aware confidence features and a shallow transformer to select which tokens should be visible, while a fixed schedule controls the expected visible fraction. We train the controller through supervised warmup followed by reinforcement learning from terminal correctness rewards. On GSM8K and MATH-500 with LLaDA-8B-Instruct, the full-canvas and two-block TIPS variants achieve 58.46% and 57.97% micro-averaged accuracy at a budget of 32 decoder evaluations, exceeding the strongest baseline in our main comparison by 12.1 and 11.6 percentage points, respectively. Both variants are trained at a budget of 32 evaluations and remain competitive at higher budgets without retraining. These results demonstrate that learning when to reveal and reconsider tokens can improve the accuracy–compute trade-off without updating the decoder backbone.

1 INTRODUCTION

Masked diffusion language models (dLLMs) generate responses by predicting unresolved positions in parallel and exposing selected predictions as context (Sahoo et al., 2024; Nie et al., 2025). This flexibility creates a decoding problem: under a limited number of denoising passes, a sampler must decide which predictions to commit and whether earlier commitments should be reconsidered. Revealing more positions makes faster progress toward a complete response, but those predictions also condition subsequent generations. An early commitment can therefore influence later predictions even after the surrounding context changes. The resulting *visibility trajectory*—which tokens become context, when they appear, and whether they can be revised—is an important component of the accuracy–compute trade-off (Kim et al., 2025a). We study how to learn this trajectory while leaving the pretrained token model unchanged.

Existing methods address different parts of this problem. Confidence-based and uncertainty-aware samplers select positions using hand-designed rules (Wu et al., 2026; Ben-Hamu et al., 2025; Huang et al., 2026a). Learned unmasking policies instead optimize position selection around a frozen denoiser, but typically preserve monotonic visibility: once a token is exposed, the policy cannot withdraw it (Jazbec et al., 2026; Xu et al., 2026; Hong et al., 2026a). Conversely, remasking methods allow revision through prescribed transitions or heuristic criteria (Wang et al., 2025; Hong et al., 2026b; Zhai et al., 2026), while other approaches learn remasking behavior by updating the language model itself (Huang et al., 2026b). These developments motivate a complementary question: can a small auxiliary policy learn both commitment and revision without modifying the dLLM?

We introduce **T**iny **I**nference-time **P**olicy for **S**caling (**TIPS**), an approximately 600K-parameter controller assigning to each response position a binary next-step visibility action. At masked positions, the action reveals the decoder's proposal or leaves the position masked; at visible positions, it retains the current token or remasks it for subsequent regeneration. A two-layer transformer predicts visibility preferences from confidence features, mask indicators, and normalized decoding time.

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.

TIPS separates decoding *order* from *pace*. A fixed power-law schedule specifies the expected visible fraction, while a shared shift of the policy logits matches this target without changing the policy's relative preferences. Thus, the expected amount of context grows according to the schedule even as individual commitments remain reversible. Our training recipe first provides dense, reference-derived supervision of visibility decisions, then optimizes final-answer correctness through group-relative policy optimization (Shao et al., 2024). Only the auxiliary controller is updated in either stage; training requires neither gradients through the dLLM nor its sequence likelihood. The same visibility-control formulation applies to full-canvas and semi-autoregressive decoding (Section 3).

We summarize our contributions as follows:

- We introduce TIPS, an approximately 600K-parameter policy that learns both unmasking and remasking while keeping the dLLM backbone frozen.
- We instantiate TIPS with full-canvas and blockwise decoding on LLaDA-8B-Instruct. At most decoder NFE budgets, we surpass previous state-of-the-art methods in accuracy on GSM8K and MATH-500 by up to 13.25 percentage points, otherwise remaining competitive.
- We investigate how TIPS uses commitment and revision through ablations and other supplementary experiments. These analyses highlight the importance of remasking, suggesting its benefits are not only token replacement, but also temporarily withdrawing harmful context. We further characterize the scope and limits of TIPS through transfer experiments.

2 BACKGROUND

We focus on finite-step decoding with a pretrained masked diffusion language model (dLLM). For fundamental principles and backbone training objectives, we refer readers to Austin et al. (2021); Sahoo et al. (2024); Shi et al. (2024); Nie et al. (2025); Ou et al. (2025); Arriola et al. (2025).

2.1 FINITE-STEP DECODING

Let $x_t \in \mathcal{V}^L$ denote the partially generated response after t denoising passes, let

$$\mathcal{M}_t = \{i : x_{t,i} = \texttt{[MASK]}\}, \qquad \mathcal{S}_t = \{i : x_{t,i} \neq \texttt{[MASK]}\} \tag{1}$$

be the masked and visible positions, respectively. A dLLM predicts a vocabulary distribution at every position and the reverse process samples a candidate token at each masked position:

$$p_{t,i}(v) = p_\theta(v \mid q, x_t), \qquad \widetilde{x}_{t,i} \sim p_{t,i}(\cdot), \quad i \in \mathcal{M}_t. \tag{2}$$

Note that token sampling is often controlled ad-hoc via temperature scaling, where temperature 0 denotes top-1 or greedy decoding. A decoding algorithm then chooses the set $\mathcal{S}_{t+1} \subseteq \{1, \ldots, L\}$ of positions that will be visible in the next state and applies

$$x_{t+1,i} = \begin{cases} \texttt{[MASK]}, & i \notin \mathcal{S}_{t+1}, \\ x_{t,i}, & i \in \mathcal{S}_t \cap \mathcal{S}_{t+1}, \\ \widetilde{x}_{t,i}, & i \in \mathcal{M}_t \cap \mathcal{S}_{t+1}. \end{cases} \tag{3}$$

A visible token may be kept or re-masked, while a masked position may be revealed or left masked. Many decoding algorithms impose the monotonic constraint $\mathcal{S}_t \subseteq \mathcal{S}_{t+1}$. Under this restriction, tokens that are already unmasked can no longer be modified. Re-masking and editing methods remove this monotonicity constraint, allowing an earlier prediction to return to the masked state or directly transition to another vocabulary token after the surrounding context changes.

Hence, masked-diffusion inference separates two choices: *what* token to place, and *which* positions should be visible. The first choice is supplied by the dLLM backbone. The second is supplied by the decoding algorithm. Because visible tokens become context at subsequent passes, different visibility trajectories can produce different final responses for the same model and number of forward passes.

Algorithm 1 TIPS: Inference

Require: Prompt q, response length L, decoder budget $T \geq 1$
Require: Frozen decoder p_θ, TIPS π_ϕ, fixed schedule parameter $\gamma \in \mathbb{R}$
Ensure: Fully visible response x_T using exactly T dLLM evaluations

$x_0 \leftarrow (\texttt{[MASK]}, \ldots, \texttt{[MASK]})$
for $t = 0, \ldots, T-2$ **do** ▷ No early stopping; empty when $T = 1$
  $p_t \leftarrow p_\theta(\cdot \mid q, x_t)$ ▷ One dLLM evaluation
  $m_{t,i} \leftarrow \mathbf{1}\{x_{t,i} = \texttt{[MASK]}\}$ for all i
  Compute C_t using Equation (6)
  $b_t \leftarrow \pi_\phi(C_t, m_t, t/T)$
  $\kappa_t \leftarrow \big((t+1)/T\big)^{\exp(\gamma)}$
  Compute δ_t s.t. $\frac{1}{L}\sum_{i=1}^{L} \sigma(b_{t,i} + \delta_t) = \kappa_t$
  $\rho_{t,i} \leftarrow \sigma(b_{t,i} + \delta_t)$ for all i
  Sample $a_{t,i} \sim \mathrm{Bernoulli}(\rho_{t,i})$ for all i
  Update all positions simultaneously:

$$x_{t+1,i} = \begin{cases} \texttt{[MASK]}, & a_{t,i} = 0, \\ x_{t,i}, & a_{t,i} = 1,\ m_{t,i} = 0, \\ \widehat{x}_{t,i}, & a_{t,i} = 1,\ m_{t,i} = 1. \end{cases}$$

end for
$p_{T-1} \leftarrow p_\theta(\cdot \mid q, x_{T-1})$ ▷ Final dLLM evaluation
$\widehat{x}_{T-1,i} \leftarrow \arg\max_{v \in \mathcal{V}} p_{T-1,i}(v)$ for all i
Fill remaining masks:

$$x_{T,i} = \begin{cases} x_{T-1,i}, & x_{T-1,i} \neq \texttt{[MASK]}, \\ \widehat{x}_{T-1,i}, & x_{T-1,i} = \texttt{[MASK]}. \end{cases}$$

return x_T

2.2 The decoding path as a Markov decision process

A decoding path can be viewed as a finite-horizon Markov decision process (Jazbec et al., 2026). At decision step t, the state contains the prompt, current partially generated response, and denoising time. One dLLM evaluation produces the candidate distributions in Equation (2), and an action is represented by the next visible set $\mathcal{S}_{t+1}$ or – equivalently – by a binary next-step visibility decision per position, denoted $a_t \in \{0,1\}^L$, where 1 denotes unmasked. The transition then applies Equation (3). For non-zero temperatures, the transition is stochastic when conditioning on the action; with greedy decoding it is deterministic given the state and action. We assume greedy decoding from here on out.

3 Method

TIPS is a lightweight auxiliary policy network π_ϕ that controls the inference trajectory of a frozen dLLM. Following the Bernoulli action parameterization of Jazbec et al. (2026), the policy maps decoding-state features to position-wise binary decisions. TIPS extends unmasking-only decisions to *next-step visibility*: an action can reveal or defer a masked position, or retain or remask an already visible token. TIPS is also *fixed-horizon*: it does not terminate early when all tokens are unmasked. Subsequent policy actions may remask tokens for revision. See Algorithm 1 for details.

One dLLM forward pass produces probabilities over non-mask tokens $p_{t,i}(v) := p_\theta(v \mid q, x_t)$. Define mask indicator $m_{t,i} = \mathbf{1}\{x_{t,i} = \texttt{[MASK]}\}$ and greedy candidate $\widehat{x}_{t,i} = \arg\max_{v \in \mathcal{V}} p_{t,i}(v)$.

Featurization. For every position, define the *active token*:

$$v_{t,i}^{\mathrm{act}} = \begin{cases} \widehat{x}_{t,i}, & m_{t,i} = 1, \\ x_{t,i}, & m_{t,i} = 0, \end{cases} \tag{4}$$

with corresponding probability $P_{t,i}^{\mathrm{act}} = p_{t,i}(v_{t,i}^{\mathrm{act}})$. At a masked position, the active token is the decoder's greedy proposal. At an unmasked position, it is the already decoded token, even if the token is not the decoder's argmax. This is done to ensure *visibility consistency*: always measuring support for the token whose visibility the policy controls. For more details, refer to Appendix B.2.3.

We also define the strongest alternative

$$v_{t,i}^{\text{alt}} = \underset{v \in \mathcal{V} \setminus \{v_{t,i}^{\text{act}}\}}{\arg\max}\ p_{t,i}(v), \tag{5}$$

with probability $P_{t,i}^{\text{alt}} = p_{t,i}\big(v_{t,i}^{\text{alt}}\big)$, which is the most probable token other than the active token.

Now, we construct the two-dimensional per-token feature

$$C_{t,i} = \left[\log \frac{P_{t,i}^{\text{act}}}{1 - P_{t,i}^{\text{act}}},\ \log \frac{P_{t,i}^{\text{act}}}{P_{t,i}^{\text{alt}}} \right]. \tag{6}$$

The first coordinate is the active token's log-odds, whose parametrization can distinguish changes near the low- and high-probability extremes more clearly than raw probability. The second coordinate measures *relative support* of the active token against the most likely competing token. This log-ratio is negative if and only if the decoder prefers a different token over the current visible token. It is also important to mention the two features are on the same numerical scale.

A two-layer RoPE-DiT (Peebles & Xie, 2023; Su et al., 2024) policy network produces per-position visibility logits $b_t = \pi_\phi(C_t, m_t, t/T)$, representing the policy's relative preference for each position's visibility (see Appendix C.1). These logits determine *which* positions are preferred while also allowing an external unmasking schedule to control *how many* positions become visible.

Scheduled visibility. We observe in Appendix B.2.5 a network that is free to determine the total visible fraction leads to a lazy strategy we hereafter refer to as *finalization deferral*: keeping most positions masked until the final forward pass, where all are unmasked at once. This prevents the decoder from conditioning on progressively richer contexts, and leads to worse task performance. Therefore, we introduce a scalar shift to the output logits so that the adjusted logits adhere to the schedule in expectation.

For normalized decoding time $\tau \in [0, 1]$, define the family $\kappa(\tau) = \tau^{\exp(\gamma)}$, $\gamma \in \mathbb{R}$. For $\gamma > 0$, the curve is convex: visibility grows slowly at first and more rapidly toward the end of decoding. For $\gamma < 0$, it is concave, exposing more tokens early; $\gamma = 0$ recovers a linear schedule. In all cases, $\kappa(0) = 0$ and $\kappa(1) = 1$, corresponding to completely masked and completely unmasked states at the beginning and end of decoding, respectively. Thus, for a budget of T dLLM evaluations and parameter γ, the target fraction of visible tokens for the next state is $\kappa_t = ((t+1)/T)^{\exp(\gamma)}$, with $t = 0, \ldots, T-2$. To achieve this ratio in practice without changing local preference, we find a shared scalar δ_t such that $\frac{1}{L}\sum_{i=1}^{L} \sigma(b_{t,i} + \delta_t) = \kappa_t$, with calibrated next-state visibility probabilities $\rho_{t,i} = \sigma(b_{t,i} + \delta_t)$. As the left hand side is continuous and strictly monotonic in δ_t, it has a unique solution, which we solve for numerically via Newton iteration. For more details, refer to Equation (15). The same scalar is added to every position, preserving the policy's preference ordering. The realized number of visible positions remains stochastic as TIPS samples $a_{t,i} \sim \text{Bernoulli}(\rho_{t,i})$, where $a_{t,i}$ has the state-independent interpretation of

$$a_{t,i} = 1 \iff \text{position } i \text{ is visible in } x_{t+1}, \tag{7}$$

while following the schedule in expectation, i.e., $\mathbb{E}\left[\frac{1}{L}\sum_{i=1}^{L} a_{t,i} \middle| x_t\right] = \kappa_t$.

Policy optimization. As in Jazbec et al. (2026), we use conditionally independent Bernoulli actions with a tractable factorized likelihood. For TIPS's next-step visibility actions, the joint log-probability is

$$\log \Pr_\phi(a_t \mid s_t) = \sum_{i=1}^{L} \left[a_{t,i} \log \rho_{t,i} + (1 - a_{t,i}) \log(1 - \rho_{t,i}) \right], \tag{8}$$

where $s_t = (C_t, m_t, t/T)$ denotes the policy state. During training, we sample a group of G visibility trajectories with final responses to obtain a binary reward R_g from a verifier. We can then optimize the policy network with GRPO (Schulman et al., 2017; Shao et al., 2024):

$$\mathcal{L}_{\text{RLVR}}(\phi) = -\frac{1}{G(T-1)} \sum_{g=1}^{G} \sum_{t=0}^{T-2} \min\Big(w_{t,g}(\phi) A_g, \text{clip}\left(w_{t,g}(\phi), 0.5, 1.5\right) A_g \Big), \tag{9}$$

where $w_{t,g}(\phi) = \frac{\Pr_\phi(a_t^{(g)}|s_t^{(g)})}{\Pr_{\text{old}}(a_t^{(g)}|s_t^{(g)})}$ is the likelihood ratio and $A_g = R_g - \frac{1}{G}\sum_{g=1}^{G} R_g$ is the group-relative advantage.

Supervised warmup. If reference response data is available, then we can optionally warmup the policy network before policy optimization by providing dense supervision of network logits. We collect intermediate decoding states from rollouts, then construct a target using the reference tokens. Given a reference response y and a masked position, we define the probability of the reference token and the strongest non-reference token as $q_{t,i} = p_{t,i}(y_i)$ and $\widehat{q}_{t,i} = \max_{v \in \mathcal{V}\setminus\{y_i\}} p_{t,i}(v)$, respectively. Now, we assign a *visibility utility* to each position:

$$u_{t,i} = \begin{cases} +1, & m_{t,i} = 0,\ x_{t,i} = y_i, \\ -1, & m_{t,i} = 0,\ x_{t,i} \neq y_i, \\ \frac{q_{t,i} - \widehat{q}_{t,i}}{q_{t,i} + \widehat{q}_{t,i}}, & m_{t,i} = 1. \end{cases} \tag{10}$$

At visible positions, this utility favors retaining tokens that match the reference response and re-masking those that are mismatching. At masked positions, it measures the decoder's support for the reference token relative to its strongest competitor. The masked utility can also be written as $\tanh\left(\frac{1}{2}\log q_{t,i}/\widehat{q}_{t,i}\right) \in [-1, 1]$. The utilities are converted into visibility targets $\rho^\star_{t,i} = \text{sg}[\sigma(\beta u_{t,i} + \delta^\star_t)]$, where $\beta > 0$ controls sharpness and $\text{sg}[\cdot]$ is the stopgrad operator. We minimize the binary cross-entropy between the policy probabilities and the targets (details in Appendix C.4)

$$\mathcal{L}_{\text{warmup}}(\phi) = -\frac{1}{L}\sum_{i=1}^{L}\left[\rho^\star_{t,i}\log\rho_{t,i} + (1-\rho^\star_{t,i})\log(1-\rho_{t,i})\right]. \tag{11}$$

Semi-autoregressive decoding. TIPS generalizes to blockwise decoding trivially by treating each block as the full canvas (Arriola et al., 2025). For a total budget of T dLLM evaluations split across B blocks, we allocate T/B calls to each block. For each block, TIPS is active for the first $T/B - 1$ calls before the deterministic final forward pass that unmasks all remaining mask tokens. We train and evaluate two variants of TIPS, full-canvas and $B = 2$, referred to as $\text{TIPS}_{\text{full}}$ and $\text{TIPS}_{\text{block}}$.

4 Experiments

4.1 Setup

Backbone and tasks. All experiments use the frozen LLaDA-8B-Instruct decoder with a 256-token response length (Nie et al., 2025) unless stated otherwise. We train the models on the GSM8K and MATH training sets and evaluate on the GSM8K test set and MATH-500, containing 1,319 and 500 problems, respectively (Cobbe et al., 2021; Hendrycks et al., 2021; Lightman et al., 2024). For all methods, we report the accuracy over the 1,819 problems (i.e., micro-average over both test sets).

Training. We train TIPS across three seeds with fixed decoding budget $T_{\text{train}} = 32$, beginning with the supervised warmup followed by 935 RLVR updates. For the choice of hyper-parameter γ, we first roll out the policy on a validation set without supervised warmup or logit calibration and choose γ by least-squares projection of its mean expected visibility trajectory onto the family $\tau^{\exp(\gamma)}$ (details in Appendix A.1). We get $\gamma_{\text{full}} \approx 0.145$ and $\gamma_{\text{block}} \approx 0.187$.

Baselines. We reproduce and compare with a broad suite of baselines. To preserve visual fidelity, we show only a subset of methods, identified by the highest-performing or competitive baselines at each evaluated decoder-call budget to include them in Figure 1 and Table 1.

This includes WINO (Hong et al., 2026b), Fast-dLLM's confidence-based decoder (Wu et al., 2026), and UNCODE (Huang et al., 2026a). We additionally include the unmasking-only policy from Jazbec et al. (2026) as a learned-policy comparator. Results for all evaluated baseline configurations appear in Appendix B.1, with implementation details in Appendix C.6.

All methods use the exact same setting except for block decomposition, in which we followed the settings found in the method's original manuscripts. For each method, we evaluate the performance

Table 1: Micro-averaged test accuracy (%) on GSM8K and MATH-500 (1,819 problems) at different decoder-call budgets. For compactness, we display a subset of baselines that preserves the highest or competitive accuracy at every evaluated budget across the full comparison suite. TIPS accuracy entries show the mean with ± standard error underneath, across three training-seed means; each seed mean averages three inference draws. AUC is the normalized area under the accuracy versus $\log_2$-budget curve over budgets 4–256, computed using trapezoidal quadrature and reported on the same 0–100 scale as accuracy. TIPS AUC values are computed from the mean accuracy curves. Bold indicates the highest accuracy among all evaluated configurations in each budget column; in the AUC column, it indicates the highest value among the displayed methods. Blocks denotes number of blocks × tokens per block. The policy from Jazbec et al. (2026) corresponds to the expert-steering variant. Fast-dLLM's unavailable 4-call accuracy is imputed as its 8-call accuracy.

Method	Blocks	4	8	16	32	64	128	256	AUC
Fast-dLLM	8×32	17.48	17.48	14.73	37.93	**65.59**	67.01	67.01	40.83
WINO	1×256	**18.14**	21.06	34.80	46.34	49.26	49.37	49.37	39.10
UNCODE	1×256	12.26	14.35	21.50	29.74	53.66	63.66	67.73	37.15
Jazbec et al. (2026)	1×256	5.63	7.42	9.18	12.39	47.72	64.80	64.63	29.44
TIPS$_{\text{full}}$	1×256	16.46 ±1.79	**29.12** ±1.68	**48.05** ±0.73	**58.46** ±0.10	63.09 ±0.50	65.32 ±0.55	66.28 ±0.71	**50.90**
TIPS$_{\text{block}}$	2×128	12.53 ±0.45	20.95 ±0.99	42.12 ±2.04	57.97 ±1.23	64.49 ±0.80	**67.12** ±0.28	**68.18** ±0.36	48.83

at exponentially increasing budgets $T \in \{4, 8, 16, 32, 64, 128, 256\}$. Fixed-horizon methods will use exactly the requested number of decoder calls. For Fast-dLLM, T is a target mean NFE (refer to Appendix A.2); other adaptive methods use decoder-call caps and may terminate earlier.

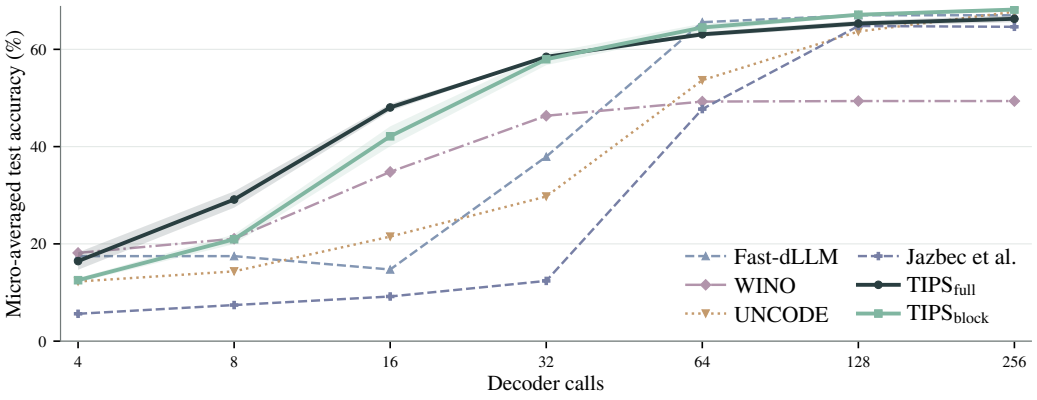


Figure 1: **Accuracy–compute tradeoff on mathematical reasoning.** Micro-averaged test accuracy on GSM8K and MATH-500 (1,819 problems) versus total decoder-calls, shown on a logarithmic scale. We compare TIPS$_{\text{full}}$ (1×256) and TIPS$_{\text{block}}$ (2×128) against Fast-dLLM (8×32), WINO, UNCODE, and the learned unmasking policy of Jazbec et al. (2026) with expert-steering; the latter three use 1×256 full-canvas decoding. The displayed baselines include a highest or competitive accuracy baseline at every evaluated budget from the full comparison suite. TIPS curves show means with shaded bands indicating ± 1 standard error across three training-seed means, each averaged over three inference draws. Baselines are shown as point estimates. Other adaptive baselines may use fewer calls than the specified budget.

Metrics. We primarily measure the micro-averaged accuracy of each method reported across different decoder-call budgets. In Table 1, we additionally report the standard error of our method's accuracies across three seeds. Further details are given in Appendix B.1.

Additionally, to summarize performance across decoding budgets using a single numerical value, we also estimate the *area under the accuracy log-budget curve* (AUC). Let $\mathrm{Acc}(k)$ be the method's accuracy at NFE budgets $k \in \{4, 8, 16, 32, 64, 128, 256\}$, we use a piecewise-linear interpolation in

Table 2: **Zero-shot transfer.** Test micro-accuracy (%) on GSM8K and MATH-500, one seed. Block-size transfer uses the same frozen 1×256-trained policy. T is the decoder-call budget.

Method / geometry	$T = 16$	$T = 32$	$T = 64$
Dream-7B: zero-shot transfer, 1×256.			
$\text{TIPS}_{\text{full}}$	35.18	46.56	52.39
Fast-dLLM	22.87	27.43	29.96
LLaDA: block-size transfer.			
1×256 (training geometry)	47.00	59.10	62.67
2×128	36.78	55.25	62.89
4×64	23.80	46.95	59.81
8×32	22.43	34.03	55.25

Table 3: **Handoff from a heuristic sampler at $T = 64$.** Test micro-accuracy (%), one seed, switching from non-adjusted confidence thresholding to TIPS after budget fraction q. Both variants outperform continued heuristic decoding ($q = 1$) at every tested intermediate q.

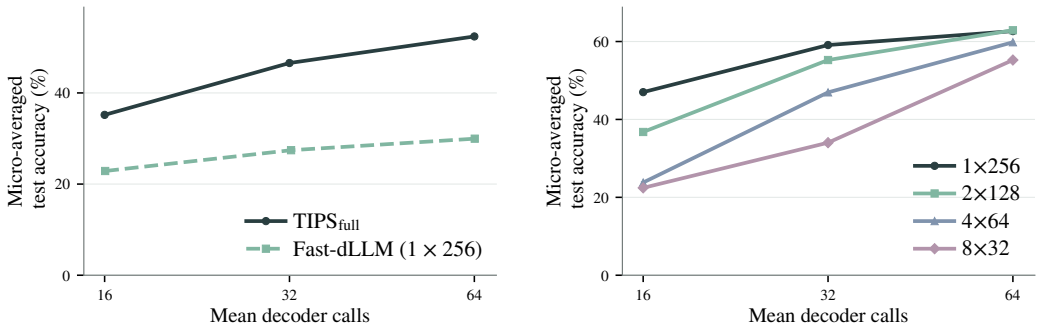
Fraction q	$\text{TIPS}_{\text{full}}$	$\text{TIPS}_{\text{block}}$
0 (TIPS)	63.39	64.54
$1/4$	40.85	38.70
$1/2$	31.34	38.15
$3/4$	25.73	40.19
1 (Non-adj. Fast-dLLM)	23.91	34.91

$\log_2$ NFE and trapezoidal quadrature, which simplifies to

$$\text{AUC} = \frac{1}{\log_2 256 - \log_2 4} \int_{\log_2 4}^{\log_2 256} \text{Acc}(2^u)\, \mathrm{d}u = \frac{1}{6} \sum_{k \in \{4,8,\ldots,128\}} \frac{\text{Acc}(k) + \text{Acc}(2k)}{2}.$$

If the method is adaptive and finishes decoding before using up the allocated NFE budget, its highest NFE accuracy is continued as a constant until $k = 256$ as the additional budget leaves the method's output and hence performance unchanged.

4.2 Results



(a) Zero-shot DREAM transfer

(b) Block-size transfer

Figure 2: **Generalization of $\text{TIPS}_{\text{full}}$.** Left: the LLaDA-trained full-canvas policy transfers to Dream-7B. Right: $\text{TIPS}_{\text{full}}$ is applied to different block sizes on LLaDA, with decoding budget shared across blocks.

Results. Figure 1 and Table 1 presents our main results. TIPS substantially improves the accuracy–compute tradeoff at low and moderate decoding budgets, with $\text{TIPS}_{\text{full}}$ achieving the highest accuracy among all methods at 8–32 call caps. Notably, $\text{TIPS}_{\text{full}}$ at a budget of 16 calls already exceeds WINO's accuracy at a budget cap of 32 calls. The two TIPS variants exhibit complementary strengths across decoding budgets. While $\text{TIPS}_{\text{full}}$ performs better at 4–32 call caps, $\text{TIPS}_{\text{block}}$ achieves higher accuracy at 64–256 call caps and is often the highest accuracy among the methods in this range. At 256 call cap, $\text{TIPS}_{\text{block}}$ reaches 68.18%, compared with 67.73% for UNCODE and 67.01% for Fast-dLLM: TIPS remains competitive at high budget.

By looking at the reported AUC values and inspecting Figure 1 visually, we see that $\text{TIPS}_{\text{full}}$ and $\text{TIPS}_{\text{block}}$ achieves substantially higher AUC than the other displayed baselines, with Fast-dLLM being the next best at 40.83. Overall, this demonstrates that both methods are effective at many budget levels without the need to re-fit, and show the value of learning a lightweight controller for the decoding trajectory.

4.3 Supplementary Experiments

We investigate whether TIPS can transfer beyond its decoder backbone during training, different decoding geometries, and whether it can effectively continue decoding from states generated by another sampler. These experiments use one seed and, whenever applicable, are evaluated with decoder NFE budgets $T \in \{16, 32, 64\}$. Unless explicitly mentioned, the mathematical reasoning results are micro-averaged over the test set of GSM8K and MATH-500 as before. Additional experiments and results appear in Appendix B.

Zero-shot transfer to Dream-7B. We apply $\mathrm{TIPS}_{\mathrm{full}}$ policy network to Dream-7B-Instruct without retraining and compare the performance to Fast-dLLM on a single 256-token canvas (Ye et al., 2025). As seen in Figure 2 and Table 2, these gains show that non-trivial visibility decisions can transfer to a different decoder backbone without the need for re-training.

Zero-shot transfer across block sizes. We reuse $\mathrm{TIPS}_{\mathrm{full}}$ for 2×128, 4×64, and 8×32 decoding, keeping the total response length and decoder budget fixed, in order to examine TIPS' ability to generalize to different block sizes which, in turn, also shed light on its performance on different response lengths. Also seen in Figure 2 and Table 2, $\mathrm{TIPS}_{\mathrm{full}}$ can preserve accuracy at slightly smaller block sizes before eventual degradation at 8×32.

Handoff from a heuristic sampler. We evaluate whether TIPS can effectively continue decoding from intermediate states generated by another sampler. To do so, we run Fast-dLLM's confidence-thresholding sampler with default threshold of 0.9 for a fraction $q \in [0, 1]$ of the total forward calls then switch to TIPS. Across both TIPS variants and various budgets, 16 out of the 18 settings improve over the baseline confidence-thresholding sampler, demonstrating a consistent ability to recover from states generated from other decoding methods. For $T = 64$ specifically, as seen in Table 3, all settings q are able to improve over continued heuristic decoding. However, available comparisons indicate that discarding the inherited canvas and restarting TIPS with only the remaining $(1 - q)T$ calls can yield higher accuracy. Thus, handoff improves on continued heuristic decoding, but does not necessarily outperform restarting. See Appendix B.3.4 for details.

We further apply $\mathrm{TIPS}_{\mathrm{full}}$ after non-adjusted Fast-dLLM has completed its response: four additional decoder evaluations raise test micro-accuracy from 67.01% to 67.29%, correcting five initially incorrect answers without making any initially correct answer incorrect in this run (Appendix B.3.5).

4.4 Ablations

We ablate TIPS on a held-out validation set of 512 GSM8K and 512 MATH training examples, reporting micro-averaged accuracy. Its different dataset composition makes these accuracies not directly comparable to test results. Ablations cover supervised warmup, remasking, visibility-aware input features, the choice of and sensitivity to γ, policy architecture and runtime, and warmup and RL decoding budgets. Full experimental details and accuracy–compute curves appear in Appendix B.2.

5 Related Work

Training-free decoding order and adaptive parallelism. Masked diffusion allows many reveal orders, which can materially affect generation quality (Sahoo et al., 2024; Kim et al., 2025a). Samplers often prescribe how many positions to reveal at each pass, selecting their identities using decoder statistics. LLaDA uses confidence-ranked unmasking by default (Nie et al., 2025). More structured approaches spread reveals across dilated position groups or compare short candidate decoding paths before committing tokens (Luxembourg et al., 2026; Lee et al., 2026). Adaptive-parallelism methods include Fast-dLLM, which combines confidence-thresholded decoding with approximate block-wise KV caching (Wu et al., 2026), and EB-Sampler, which dynamically reveals multiple positions subject to an entropy-based error tolerance (Ben-Hamu et al., 2025). Other methods refine the selection criterion: UNCODE/PC-Sampler calibrates uncertainty scores using position- and token-informativeness priors (Huang et al., 2026a); KLASS incorporates the temporal KL stability

of token distributions (Kim et al., 2025b); and DAPD uses a self-attention-derived dependency graph to select weakly coupled positions for parallel decoding (Kim et al., 2026).

Learned unmasking schedules. A closely related line of work learns a controller around a pretrained denoiser. Jazbec et al. (2026) formulate masked-diffusion sampling as a Markov decision process and use reinforcement learning to train a one-layer transformer that maps decoder confidences to monotonic unmasking decisions. Their transitions preserve already visible tokens, whereas TIPS extends the action semantics to reversible next-step visibility. Where-to-Unmask trains a supervised ranker to imitate a ground-truth probability-margin oracle (Asano et al., 2026). Adaptive Order Policies reweight masked-diffusion loss terms by policy probabilities to favor positions at which the denoiser is likely to be correct, considering both frozen-denoiser and joint-training settings (Mohamud et al., 2026). Self-Aware Scheduling (SAS) derives a dense trajectory reward from a bound on sequential decoding mismatch and trains an order policy with group-relative policy optimization while keeping the denoiser fixed (Xu et al., 2026). UPO similarly learns a stochastic position-selection policy for a frozen masked diffusion model, using a KL-regularized, GRPO-style objective around an explicit heuristic reference policy (Hong et al., 2026a). Its actions select only currently masked positions, preserving monotonicity.

Re-masking and revision. Re-masking methods relax monotonicity by allowing earlier token commitments to be withdrawn. P2 and ReMDM allow visible positions to return to the mask state and be regenerated, with ReMDM deriving its sampler from a discrete diffusion process with a custom reverse transition (Peng et al., 2025; Wang et al., 2025). Running Confidence Remasking (RCR), introduced alongside MDPO, revisits positions with poor running confidence (He et al., 2025). WINO uses a training-free parallel draft-and-verify procedure to reveal candidates and remask suspicious visible tokens (Hong et al., 2026b), while CoRe identifies revision targets by testing prediction stability under masked-context perturbations (Zhai et al., 2026). These approaches prescribe revision rules rather than learning a separate visibility controller. RemeDi instead modifies the language model to jointly predict tokens and confidence scores, then trains its re-masking behavior through supervised and reinforcement-learning stages (Huang et al., 2026b). TIPS also uses a supervised warmup phase followed by reinforcement learning.

Reinforcement learning for diffusion language models. Policy optimization has also been applied directly to the dLLM. MDPO updates the masked token model along progressive denoising trajectories using intermediate rewards (He et al., 2025); mask-aware policy gradients account for both token and masking decisions (Raajesh et al., 2026); and SPG constructs upper- and lower-bound surrogates for the otherwise intractable sequence log-likelihood (Wang et al., 2026). TIPS instead applies policy gradients only to its auxiliary Bernoulli visibility policy, hence training requires neither sequence likelihood nor updates to decoder parameters.

6 Discussion

We introduced TIPS, a small policy that learns unmasking and remasking around a frozen dLLM, separating visibility control from token prediction while a fixed schedule regulates expected progress. On GSM8K and MATH-500, TIPS improves low- and moderate-budget accuracy and remains competitive at higher budgets without retraining; full-canvas and blockwise variants offer complementary strengths. Ablations, corruption recovery, heuristic-prefix handoffs, remasking and editing statistics, and transfer experiments highlight the value of warmup and remasking, suggest that revision may help by withdrawing harmful context rather than only replacing tokens, and reveal transferable but domain-dependent behavior.

Limitations. Our evaluation centers on 256-token mathematical reasoning, and the supplementary transfer experiments use a single seed; longer generations and broader task coverage remain important tests of generality. TIPS also uses a fixed decoding horizon rather than an adaptive stopping rule, preserving further revision but foregoing savings from early termination. Developing stopping criteria that account for possible future revisions, and evaluating the resulting accuracy–latency trade-off, are important directions for future work. Decoder-call budgets also do not directly capture wall-clock cost.

AI USE STATEMENT

In the preparation of this manuscript, GPT and Claude were utilized as writing and coding assistants. They were used to improve grammar, clarity, and readability, as well as to assist with writing code used to conduct and analyze experiments. All scientific ideas, experimental design, interpretation of results, and conclusions were conceived and formulated by the authors. All AI-assisted text and code, together with the resulting experimental outputs and analyses, were subsequently reviewed and verified by the authors to ensure correctness and preserve the intended scientific meaning.

ETHICS STATEMENT

TIPS is a general-purpose method for improving inference-time decoding in diffusion language models and is not designed for any particular high-risk application. Our experiments use existing pretrained language models and standard benchmark datasets, and do not involve the collection of new human-subject data. As with other methods that improve the efficiency or capability of language-model inference, TIPS may also make deployment of the underlying models easier or less costly; consequently, it inherits the potential benefits and risks of the models to which it is applied. We do not modify the underlying model's safety mechanisms or make claims regarding its suitability for safety-critical applications.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the TIPS architecture, training procedure, decoding algorithm, feature construction, scheduling mechanism, and experimental protocol. The paper and supplementary material specify the principal hyperparameters, inference budgets, model and decoding configurations, datasets, evaluation procedures, and random-seed settings used in our experiments. We also provide an anonymized codebase with the trained TIPS checkpoints and a self-contained inference script to facilitate reproduction. Ablations are conducted on a held-out validation set used for method development, while final results are evaluated on the designated test sets. Where applicable, we report results across multiple training seeds and explicitly identify experiments conducted with a single seed. These details are intended to enable faithful reproduction and comparison of our results.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block Diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

Hikaru Asano, Tadashi Kozuno, Kuniaki Saito, and Yukino Baba. Where-to-unmask: Ground-truth-guided unmasking order learning for masked diffusion language models, 2026. URL https://arxiv.org/abs/2602.09501.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S. Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, volume 38, pp. 62365–62391, 2025. doi: 10.52202/085713-1874. URL https://doi.org/10.52202/085713-1874.

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

Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. MDPO: Overcoming the training-inference divide of masked diffusion language models, 2025. URL https://arxiv.org/abs/2508.13148.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving discrete diffusion unmasking policies beyond explicit reference policies. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=on6cb46OhD.

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. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=XtLQHlNLxy.

Pengcheng Huang, Tianming Liu, Zhenghao Liu, Yukun Yan, Shuo Wang, Tong Xiao, Zulong Chen, and Maosong Sun. Empirical analysis of decoding biases in masked diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6853–6876, San Diego, California, United States, July 2026a. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.311. URL https://aclanthology.org/2026.acl-long.311/.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=BsZeTuB5fD.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=3aXoaet48C.

Bumjun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. DAPD: Dependency-aware parallel decoding via attention for diffusion LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=LZro8iqjiA.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30749–30768. PMLR, 2025a. URL https://proceedings.mlr.press/v267/kim25ah.html.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 102436–102470, 2025b. doi: 10.52202/085713-3087. URL https://doi.org/10.52202/085713-3087.

Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits reliable decoding in diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OdM5JRVEiK.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Omer Luxembourg, Haim Permuter, and Eliya Nachmani. Plan for speed: Dilated scheduling for masked diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=hcODkuS1cF.

Jama Hussein Mohamud, Mohsin Hasan, Mirco Ravanelli, and Yoshua Bengio. Adaptive order policies for masked diffusion, 2026. URL https://arxiv.org/abs/2606.00295. Accepted at the ReALM-GEN Workshop at ICLR 2026.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 56354–56392, 2025. doi: 10.52202/085713-1689. URL https://doi.org/10.52202/085713-1689.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sMyXP8Tanm.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling, 2025. URL https://arxiv.org/abs/2502.03540.

Haran Raajesh, Kulin Shah, Adam Klivans, and Philipp Krähenbühl. Mask-aware policy gradients for diffusion language models. In *The Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=mbjRRWwZDP.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 103131–103167, 2024.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. SPG: Sandwiched policy gradient for masked diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=18j5Q49GwN.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 163488–163545, 2025. doi: 10.52202/085713-4927. URL https://doi.org/10.52202/085713-4927.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Jiawei Xu, Minghui Liu, Aakriti Agrawal, Yifan Chen, and Furong Huang. Scheduling thoughts: Learning the order of thought in diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=j4UhHaxCpq.

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.

Kevin Zhai, Sabbir Mollah, Zhenyi Wang, and Mubarak Shah. CORE: Context-robust remasking for diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bmKHxLWkz9.

A EXPANDED METHODOLOGY

A.1 FITTING THE VISIBILITY SCHEDULE

We select γ by fitting the visibility trajectory of a separately trained, uncalibrated pilot policy. Another feasible approach is to optimize γ via hyperparameter search on the validation set. We first train this pilot from scratch with RLVR and terminal correctness rewards, without supervised warmup or logit calibration. We then freeze the pilot and collect N trajectories on the held-out validation prompts, again without calibration. Let T_{fit} be the rollout budget, including the deterministic final decoder call, and let $b_{t,i}^{(j)}$ be the pilot's logit at position i and transition t in trajectory j. The empirical mean expected next-state visible fraction is

$$\bar{v}_{t+1} = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{1}{L}\sum_{i=1}^{L}\sigma\left(b_{t,i}^{(j)}\right)\right], \qquad t = 0, \ldots, T_{\text{fit}} - 2. \tag{12}$$

Because each Bernoulli action specifies whether a position is visible in the next state, the inner average is that state's conditional expected visible fraction. Using probabilities instead of sampled mask counts removes the additional noise from the next Bernoulli sample, while retaining the states visited by the uncalibrated policy.

We fit the endpoint-anchored family $\kappa_\gamma(\tau) = \tau^{\exp(\gamma)}$ by solving the one-dimensional least-squares problem

$$\widehat{\gamma} \in \arg\min_{\gamma\in\mathbb{R}} \frac{1}{T_{\text{fit}} - 1}\sum_{t=0}^{T_{\text{fit}}-2}\left[\bar{v}_{t+1} - \left(\frac{t+1}{T_{\text{fit}}}\right)^{\exp(\gamma)}\right]^2. \tag{13}$$

This is a least-squares projection of the pilot's empirical schedule onto a continuous, monotone family with $\kappa_\gamma(0) = 0$ and $\kappa_\gamma(1) = 1$. The policy observes the current time t/T_{fit}, but the fit uses the next-state time $(t+1)/T_{\text{fit}}$. We exclude deterministic finalization because its full visibility is imposed by the decoding algorithm. The fit uses neither reference answers nor correctness labels for the validation prompts. We then fix $\widehat{\gamma}$ during subsequent calibrated policy training and evaluate the same normalized schedule at other inference budgets. The pilot supplies the schedule shape; it is not the final calibrated policy.

For blockwise decoding, we apply the same procedure to the active block, using its length and local decoder-call budget in place of L and T_{fit}. We average the expected visible fractions over validation trajectories and blocks at matching local times, exclude each block's deterministic finalization, and fit one shared γ for that decoding geometry.

A.2 FAST-DLLM BUDGET MATCHING

For a target budget T, we tune the confidence-decoding threshold using test-set rollouts until the average NFE lies in $[T, T + 0.5]$. The one-sided tolerance gives the baseline up to 0.5 additional decoder calls on average. We use this mean-adjusted setting with 8×32 decoding for the main results and 1×256 decoding for Dream transfer. Results without this adjustment are labeled Non-adjusted Fast-dLLM or Non-adjusted confidence thresholding, as appropriate, and use decoder call caps. In the main comparison, threshold adjustment covers $T = 8, 16, 32, 64, 128$; at $T = 256$, we reuse the non-adjusted results, whose runs terminate well below their caps, and impute the unavailable 4-call result from the 8-call result. The 4-call result is unavailable because during our sweeps, there was no confidence-threshold hyperparameter found such that the expected NFE was within the range of $[4, 4.5]$.

It is important to note that unless otherwise specified, Fast-dLLM refers to the mean-adjusted version, while the non-adjusted version with standard confidence threshold of 0.9 is referred to as the non-adjusted Fast-dLLM.

B Additional Experiments

B.1 Main results with all baselines

Table 4 expands the main comparison to all evaluated baseline configurations. Fast-dLLM uses the mean-adjusted 8×32 setting described in Section 4.1. We also retain Non-adjusted Fast-dLLM with the validation-selected 1×256 configuration and the 8×32 shared-cap configuration; i.e. their thresholds are not adjusted to match mean NFE.

Across the 256-token configurations, $\mathrm{TIPS}_{\mathrm{full}}$ has the highest mean accuracy at budgets 8–32, while $\mathrm{TIPS}_{\mathrm{block}}$ leads at 128 and 256. Fast-dLLM leads at the target budget of 64, reaching 65.59% compared with 64.49% for $\mathrm{TIPS}_{\mathrm{block}}$. The non-adjusted 8×32 configuration reaches 37.82% at its 64-call cap. The 4×32 UPO configuration generates only 128 response tokens and is shown separately from the 256-token comparisons.

B.2 Ablations

Protocol. We evaluate the design choices of $\mathrm{TIPS}_{\mathrm{full}}$ on the 1,024-example validation set comprising 512 GSM8K and 512 MATH problems, chosen then excluded from the training data. The same prompts were also used without correctness labels to fit the native schedule exponent. Unless stated otherwise, each comparison uses training seed 123, evaluation seed 42, a 256-token response length, and inference budgets $T \in \{16, 32, 64\}$. Accuracy is the number of correct answers divided by 1,024. Because this equally weighted dataset mixture differs from the 1,319:500 test mixture, its absolute accuracies should not be compared directly with the main test results. The feature-width and finalization-deferral studies are historical exceptions, with their protocols specified separately below.

B.2.1 Supervised warmup and reinforcement learning

We compare the complete supervised warmup–RLVR recipe with its supervised parent and a policy trained with RLVR alone. The RLVR-trained policies use the same 935-update continuation length, while the warmup-only policy receives no RLVR updates. As shown in Table 5 and Figure 3, both stages contribute to the final performance. Warmup followed by RLVR improves over warmup alone by 14.55, 17.87, and 14.45 percentage points at $T = 16, 32, 64$, respectively. Warmup also provides a useful initialization for RLVR: although the gain over RLVR alone is only 0.49 points at $T = 16$, it increases to 8.50 and 7.81 points at $T = 32$ and 64. Thus, the supervised policy is not simply sufficient on its own; supervised initialization and terminal-reward optimization complement one another.

B.2.2 The contribution of remasking

To separate dependence on an already learned behavior from the value of the action space itself, we evaluate two controls. The first disables remasking at inference for the frozen reference policy. The second retrains a monotone policy with supervised warmup and RLVR. In both cases, visible tokens retain their identities, and only masked positions can become newly visible. If V positions are already visible and $M = L - V$ remain masked, the reveal target is clipped to $[0, M]$ from $L\kappa_t - V$, so the intervention does not demand an infeasible decrease in visibility.

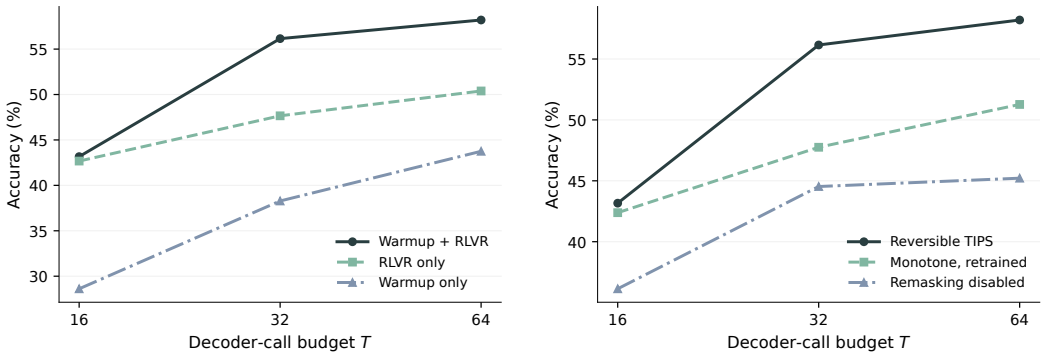
As seen in Figure 3, disabling remasking reduces accuracy by 7.03–12.99 percentage points across the three budgets. Retraining recovers part of this loss, but the monotone policy remains 0.78, 8.40, and 6.93 points below reversible TIPS at $T = 16, 32, 64$, respectively. These results support retaining the ability to reconsider visible context.

Table 4: **Complete accuracy–budget comparison on mathematical reasoning.** Micro-averaged test accuracy (%) on GSM8K and MATH-500 (1,819 problems). All baseline configurations are retained, with blocks denoting number of blocks × tokens per block. TIPS entries give the mean and, underneath, its standard error across three training-seed means; each seed mean averages three inference draws. Baselines are point estimates. Bold indicates the highest reported mean in each budget column among the 256-token configurations. Dashes indicate unreported cells. The non-adjusted Fast-dLLM shared-cap adaptation is retained for reference, and the updated mean-budget adaptation is added as a separate row. The final row uses only 128 response tokens and is displayed separately, without entering the matched comparison.

Method / configuration	Blocks	4	8	16	32	64	128	256
Single-canvas configurations: 256 response tokens								
Random / uniform	1×256	11.45	16.91	25.16	33.06	38.34	41.82	42.57
High confidence (fitted)	1×256	16.22	16.82	23.42	34.52	39.36	32.71	28.26
High confidence (linear)	1×256	5.06	12.26	30.84	37.82	33.32	27.32	29.19
Probability margin	1×256	9.84	17.59	31.61	39.53	35.51	27.54	30.40
Predictive entropy	1×256	4.45	7.81	20.51	32.49	25.73	19.79	28.59
Non-adjusted Fast-dLLM	1×256	17.70	22.37	31.67	33.32	33.48	33.48	33.48
EB-Sampler	1×256	12.04	16.88	19.85	21.72	25.40	27.21	31.39
UNCODE / PC-Sampler	1×256	12.26	14.35	21.50	29.74	53.66	63.66	67.73
KLASS	1×256	5.06	12.09	29.14	37.16	36.72	38.87	38.32
DAPD	1×256	8.36	14.79	21.37	28.95	50.47	50.82	50.82
Jazbec et al. (2026) (expert steering, retrained)	1×256	5.63	7.42	9.18	12.39	47.72	64.80	64.63
Jazbec et al. (2026) (vanilla, retrained)	1×256	5.81	9.05	16.40	38.34	48.30	47.32	47.90
UPO	1×256	5.94	7.15	8.03	10.23	12.37	17.59	49.70
SAS	1×256	9.84	20.95	33.53	41.56	43.65	43.65	43.76
ReMDM	1×256	0.55	4.93	21.73	32.40	41.14	40.52	33.53
RCR	1×256	8.34	19.72	33.33	38.67	35.79	33.88	36.93
WINO	1×256	**18.14**	21.06	34.80	46.34	49.26	49.37	49.37
CoRe	1×256	5.28	12.09	28.75	36.12	32.77	27.38	29.03
Block-wise configurations: 256 response tokens								
Non-adjusted Fast-dLLM	8×32	7.20	7.75	10.12	11.76	37.82	66.85	67.01
Fast-dLLM	8×32	—	17.48	14.73	37.93	**65.59**	67.01	67.01
KLASS (native block-wise)	4×64	14.07	21.99	28.70	40.68	59.98	66.03	66.90
UPO (block-trained; shared cap)	8×32	7.26	7.92	9.46	11.05	11.16	22.54	66.85
SAS (native block-wise)	8×32	—	17.48	23.75	33.26	53.22	61.46	66.14
WINO (rescored block-wise)	2×128	8.61	10.28	13.03	38.34	63.39	65.16	65.16
TIPS$_{\text{full}}$	1×256	16.46 ±1.79	**29.12** ±1.68	**48.05** ±0.73	**58.46** ±0.10	63.09 ±0.50	65.32 ±0.55	66.28 ±0.71
TIPS$_{\text{block}}$	2×128	12.53 ±0.45	20.95 ±0.99	42.12 ±2.04	57.97 ±1.23	64.49 ±0.80	**67.12** ±0.28	**68.18** ±0.36
Different response length: excluded from column-wise ranking								
UPO (128-token response)	4×32	—	—	—	—	—	60.25	—

Table 5: **Training stages and reversible visibility.** Validation-set micro-accuracy (%) over 1,024 problems, with one training seed and one evaluation seed. RLVR continuations use the same number of updates.

Configuration	$T = 16$	$T = 32$	$T = 64$
Warmup → RLVR	43.16	56.15	58.20
Supervised warmup only	28.61	38.28	43.75
RLVR only	42.68	47.66	50.39
Remasking disabled (frozen)	36.13	44.53	45.21
Monotone policy (retrained)	42.38	47.75	51.27



(a) Training stages

(b) Remasking controls

Figure 3: **Supervised initialization and reversible visibility.** Validation-set accuracy versus decoder-call budget on a logarithmic axis. Each curve is one trained policy evaluated with one inference seed.

B.2.3 Visibility-consistent confidence features

At an already visible position, the policy controls whether the current token remains in context, rather than whether the decoder's newly preferred token should be inserted. A global-top-1 confidence can consequently describe the wrong token. When analyzing our trajectories, the global argmax differed from the visible token in 73,930 of 3,407,617 recorded visible-position events, or 2.17%. This observation motivates anchoring the first feature to the current visible token and excluding that token when constructing the strongest alternative. While not necessarily an ablation, we believe that ensuring the policy inputs remain consistent with the current decoding state is important for the semantic correctness of our method.

B.2.4 Confidence-feature dimensionality study

We additionally report an earlier study of feature width $K \in \{1, 2, 10, 50\}$. This experiment uses a one-layer full-canvas policy trained with RLVR at $T_{\text{train}} = 32$. Inputs are raw full-vocabulary probabilities. The first coordinate measures support for the active token, using the greedy proposal at masked positions and the current token at visible positions; larger widths append the strongest alternative probabilities. The $K = 1$ and $K = 2$ variants therefore isolate whether to expose the strongest competitor.

Figure 4 reports all 14 recorded inference budgets. For each fixed trained policy, we average five seeds. Adding the strongest alternative improves the curve broadly: $K = 2$ exceeds $K = 1$ at every budget above one call, including gains of 3.56 points at $T = 16$, 2.14 points at $T = 32$, and 2.92 points at $T = 64$. Expanding the input to ten or fifty probabilities does not improve over $K = 2$. In this setting, information about the most competitive alternative is more useful than a much wider description of the probability distribution, supporting the choice of a two-channel feature.

Table 6: **Feature-width study.** Test micro-accuracy (%), reported as mean ± sample standard deviation across five evaluation seeds for each fixed policy.

Feature width K	$T = 16$	$T = 32$	$T = 64$	$T = 256$
1	40.12 ± 0.93	48.44 ± 1.03	51.38 ± 1.00	56.01 ± 0.96
2	43.68 ± 1.27	50.59 ± 0.48	54.30 ± 0.64	57.68 ± 0.48
10	38.90 ± 0.35	45.94 ± 0.60	49.97 ± 0.53	55.35 ± 1.08
50	40.13 ± 0.42	47.12 ± 1.12	49.97 ± 0.76	54.03 ± 1.10

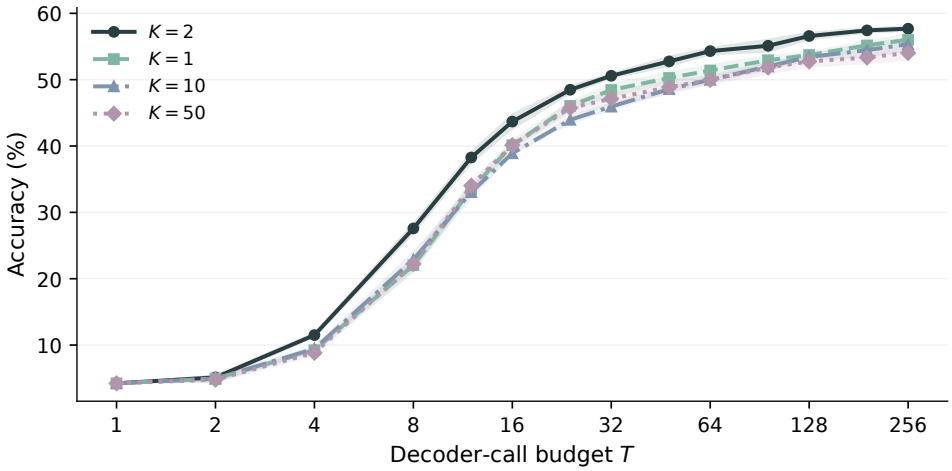


Figure 4: **Confidence-feature width.** Test micro-accuracy versus log decoder-call budget for $K \in \{1, 2, 10, 50\}$ raw-probability features. Markers include intermediate budgets as well as powers of two. Shading is one standard error across five evaluation seeds.

B.2.5 Occupancy control and finalization deferral

We revisit an earlier RLVR-only comparison of single-layer policies with one 256-token response canvas, $T = 32$ decoder calls, and three training seeds. Policies are trained with no occupancy constraint, a soft schedule loss, or fixed power-law calibration, while retaining reversible actions and deterministic finalization.

The soft schedule loss imposes a penalty that is multiplied against the binary terminal correctness reward. Specifically, if, by the finalization pass, the state has masked fraction $m \in [0, 1]$, then the assigned terminal correctness reward is multiplied by $(1 - m)$. For example, if the response is correct but only 50% of the canvas was unmasked by the finalization pass, the reward becomes 0.5 for that trajectory.

Figure 5 shows that the uncalibrated trajectory levels off before the final call. After the last learned transition, 57.11% of positions remain masked, versus 48.01% with the soft loss and 6.89% with fixed calibration (Table 7). The subsequent jump is imposed by deterministic completion, not learned visibility decisions: this is *finalization deferral*. We find the soft schedule loss, while non-constraining, insufficient for alleviating the finalization deferral phenomenon. Fixed calibration reduces this terminal workload and is accompanied by higher accuracy, from 45.90% to 50.83%.

Table 7: **Occupancy control reduces the finalization burden.** At $L = 256$ and $T = 32$, mask counts are means $\pm$ one standard deviation across three training seeds, measured at $\tau = 31/32$ with evaluation seed 42.

Occupancy control	Masks before finalization	Masked (%)	Accuracy (%)
None	146.20 ± 15.68	57.11	45.90
Soft schedule loss	122.91 ± 9.96	48.01	47.18
Fixed power-law calibration	17.65 ± 0.02	6.89	50.83

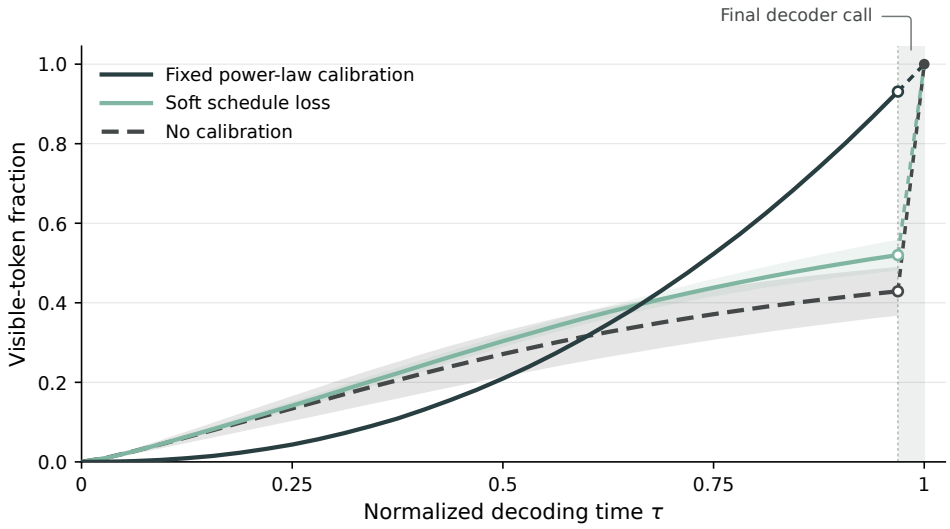


Figure 5: **Finalization deferral in RLVR-only policies.** Realized visible-token fraction at $T = 32$: mean $\pm$ one standard deviation over training seeds 123, 124, and 125, with evaluation seed 42 and example-weighted pooling of GSM8K test and MATH-500. The shaded final interval and dashed terminal segments denote deterministic finalization. The origin is the prescribed all-mask state.

B.2.6 Schedule shape and logit calibration

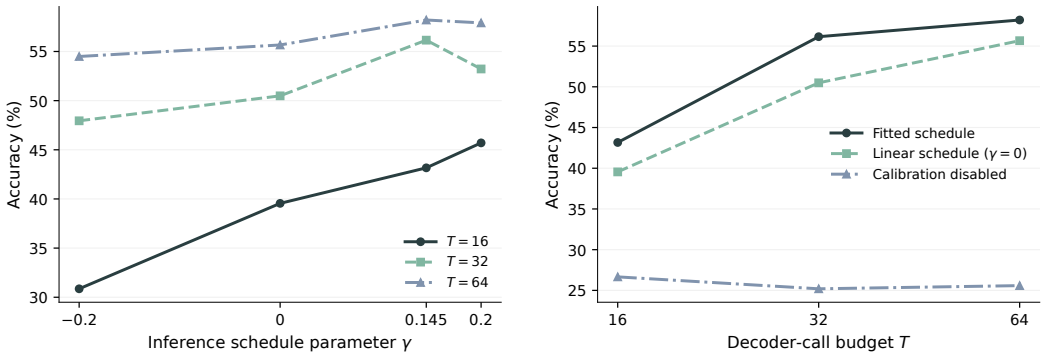
We freeze the reference policy and vary the inference schedule $\kappa(\tau) = \tau^{\exp(\gamma)}$. Table 8 and Figure 6 distinguishes changes in γ from removing calibration altogether. The fitted value is $\gamma = 0.14544$; the other evaluated settings are -0.2, 0, and 0.2. Note that $\gamma = 0$ is a calibrated linear schedule, not an uncalibrated policy.

The effect of schedule shape depends on the inference budget. Moving from the fitted value to $\gamma = 0.2$ raises accuracy at $T = 16$ from 43.16% to 45.70%, but reduces it at $T = 32$ from 56.15% to 53.22%; the $T = 64$ difference is only 0.29 points. More front-loaded visibility at $\gamma = -0.2$ is worse at all three budgets, particularly at $T = 16$. Therefore, nearby schedules can preserve much of the performance, but the schedule is not irrelevant and the best measured setting need not be the same at every budget.

Removing calibration. We next bypass the scalar occupancy-matching shift while keeping the checkpoint and deterministic finalization unchanged. Accuracy falls to 26.66%, 25.20%, and 25.59% at $T = 16, 32, 64$, respectively. In this intervention, additional decoder calls no longer yield the improvement observed with calibrated TIPS. The experiment establishes the importance of calibration for the learned policy's use of intermediate states.

Table 8: **Inference-time schedule interventions.** Validation-set accuracy (%) for frozen $\text{TIPS}_{\text{full}}$. All γ settings retain occupancy calibration; the final row bypasses the scalar logit shift entirely.

Frozen-policy setting	$T = 16$	$T = 32$	$T = 64$
Fitted: $\gamma = 0.14544$	43.16	56.15	58.20
$\gamma = -0.2$	30.86	47.95	54.49
$\gamma = 0$ (linear)	39.55	50.49	55.66
$\gamma = 0.2$	45.70	53.22	57.91
Calibration disabled	26.66	25.20	25.59



(a) Schedule-shape sensitivity

(b) Calibration intervention

Figure 6: **Schedule shape and occupancy calibration have distinct effects.** Left: validation-set accuracy at the four measured γ values, with a separate curve for each budget. Right: the fitted schedule, calibrated linear schedule, and calibration-disabled policy versus log budget. All settings use the same frozen checkpoint.

B.2.7 Policy architecture and runtime

We replace the two-layer DiT with either a one-layer DiT or a parameter-matched per-token multilayer perceptron, while retaining the training and evaluation protocol. The per-token network cannot exchange information between response positions within the controller. It nevertheless reaches 54.88% accuracy at $T = 64$, compared with 58.20% for the two-layer DiT. Thus, a substantial fraction of the policy's effectiveness can be retained with local processing of contextual decoder features. Cross-position processing still helps: the two-layer DiT exceeds the MLP at all three budgets and exceeds the one-layer DiT at $T = 32$ and 64.

Table 9 also reports the corresponding controller costs. At $T = 32$, cumulative policy time per generated example is 8.06 ms for the two-layer DiT, 6.88 ms for the one-layer DiT, and 6.22 ms for the MLP. These measurements sum the controller's work across the complete rollout.

Table 9: **Architecture and controller cost.** Validation-set accuracy (%) and total CUDA-measured policy time per generated example at $T = 32$. Timings accumulate all 31 policy decisions; they were collected on an H100 80GB with float32 policy weights and batch size at most 16. The decoder and feature-extraction cost are excluded from the policy-time column.

Policy	Parameters	$T = 16$	$T = 32$	$T = 64$	Policy time (ms)
Two-layer DiT	628,739	43.16	56.15	58.20	8.06
One-layer DiT	331,395	45.12	53.03	55.86	6.88
Per-token MLP	634,835	38.18	48.24	54.88	6.22

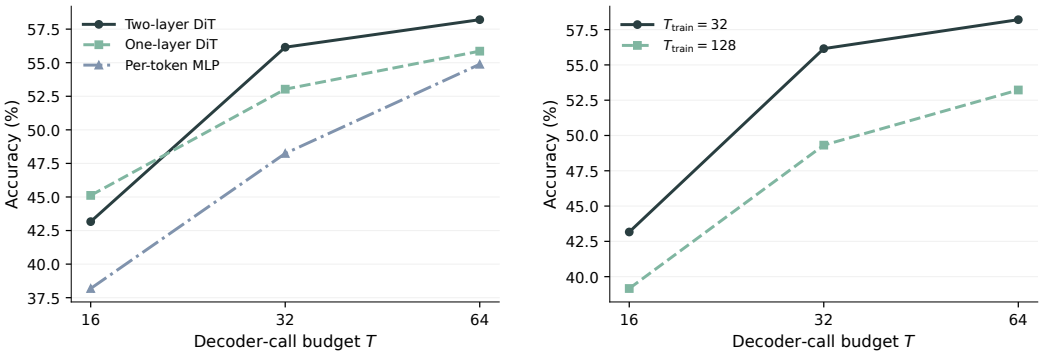
B.2.8 Training with a longer decoding budget

We compare $\text{TIPS}_{\text{full}}$ trained with $T_{\text{train}} = 32$ and $T_{\text{train}} = 128$ on the same validation problems. Unfortunately, computational constraints prevented a fully matched experiment, and the $T_{\text{train}} = 128$ configuration uses a different γ of approximately 0.8.

The $T_{\text{train}} = 32$ policy is more accurate at every measured inference budget, with gains of 4.00, 6.84, and 4.98 percentage points at $T = 16, 32, 64$, respectively. A possible explanation is that a longer training trajectory requires more visibility decisions to share the same terminal correctness signal, making credit assignment more difficult.

Table 10: **Training-budget comparison for** $\mathrm{TIPS}_{\mathrm{full}}$**.** Validation-set accuracy (%) at the indicated inference budgets.

Training configuration	$T = 16$	$T = 32$	$T = 64$
$T_{\mathrm{train}} = 32$	43.16	56.15	58.20
$T_{\mathrm{train}} = 128$	39.16	49.32	53.22



(a) Policy architecture

(b) Training decoding budget

Figure 7: **Controller architecture and training budget.** Validation-set accuracy versus log decoder-call budget.

B.2.9 Depth-matched comparison with the learned unmasking policy

To examine whether the comparison with Jazbec et al. (2026) is explained by controller depth, we evaluate one- and two-layer policies for both methods. We also vary the depth of $\mathrm{TIPS}_{\mathrm{block}}$ without changing its 2×128 decoding geometry. These depth-pair experiments use the same 1,024-example validation set and one training/evaluation seed pair. Within each method, the intended architectural change is the number of DiT layers; the training recipe is otherwise retained. Across methods, matching depth almost matches the parameter count: TIPS has 331,395 parameters at one layer and 628,739 at two, compared with 331,265 and 628,609 for the unmasking-only policy.

TIPS retains its advantage at either matched depth (Table 11). At $T = 32$, the one-layer full-canvas TIPS policy reaches 53.03%, compared with 43.85% for the one-layer comparison policy; at two layers, the respective accuracies are 56.15% and 40.33%. Moreover, doubling the comparison policy's depth does not reproduce the improvement from TIPS. Its accuracy falls from 32.32% to 20.70% at $T = 16$, and from 43.85% to 40.33% at $T = 32$, while the two depths become similar at larger decoder budgets. These single-seed comparisons show that the reported advantage is not removed by matching controller depth, without attributing it to any one of TIPS's other design choices.

For this depth comparison, we use the vanilla (non-expert-steered) variant of Jazbec et al. (2026) at both depths, motivated by its stronger low-budget performance in our reproduction. Table 1 instead reports the expert-steered variant, which achieves higher accuracy at large budgets. The validation split additionally places greater weight on the harder MATH dataset (50%, compared with 27.48% in the test mixture): it involves high-school competition-level mathematics requiring complex, multi-step symbolic reasoning, whereas GSM8K comprises elementary- and middle-school-level math word problems involving only basic arithmetic and simple multi-step algebraic manipulation. Thus, differences from Table 1 reflect changes in both the baseline variant and evaluation population.

The compute interpretation also differs between methods. Both TIPS depths use their entire allocated horizon, whereas the comparison policy may terminate early. At cap 64, its deeper controller consumes 50.30 decoder calls on average, compared with 35.42 for the one-layer controller, despite their nearly identical accuracies. We therefore report mean calls alongside the cap-based accuracy curves.

Table 11: **Matched policy-depth comparisons.** Validation-set accuracy (%) at the requested decoder-call caps. Each TIPS decoding geometry is unchanged when depth is varied.

Method	Layers	Parameters	4	8	16	32	64	128
$\text{TIPS}_{\text{full}}$	1	331,395	17.38	25.98	45.12	53.03	55.86	59.47
$\text{TIPS}_{\text{full}}$	2	628,739	18.26	28.42	43.16	56.15	58.20	59.67
$\text{TIPS}_{\text{block}}$	1	331,395	12.70	20.90	40.43	52.64	58.79	61.33
$\text{TIPS}_{\text{block}}$	2	628,739	15.04	23.54	41.60	53.03	59.18	61.33
Jazbec et al. (2026)	1	331,265	7.32	14.16	32.32	43.85	43.75	43.95
Jazbec et al. (2026)	2	628,609	7.23	11.62	20.70	40.33	43.65	44.82

Table 12: **Consumed decoder calls in the depth comparison.** Mean calls per validation sample. TIPS uses every allocated call, whereas the comparison policy may stop before reaching the cap. Equal caps therefore do not always imply equal consumed compute.

Method	Cap 16	Cap 32	Cap 64	Cap 128
TIPS (either geometry or depth)	16.00	32.00	64.00	128.00
Jazbec et al., one layer	16.00	29.52	35.42	35.92
Jazbec et al., two layers	16.00	31.95	50.30	51.38

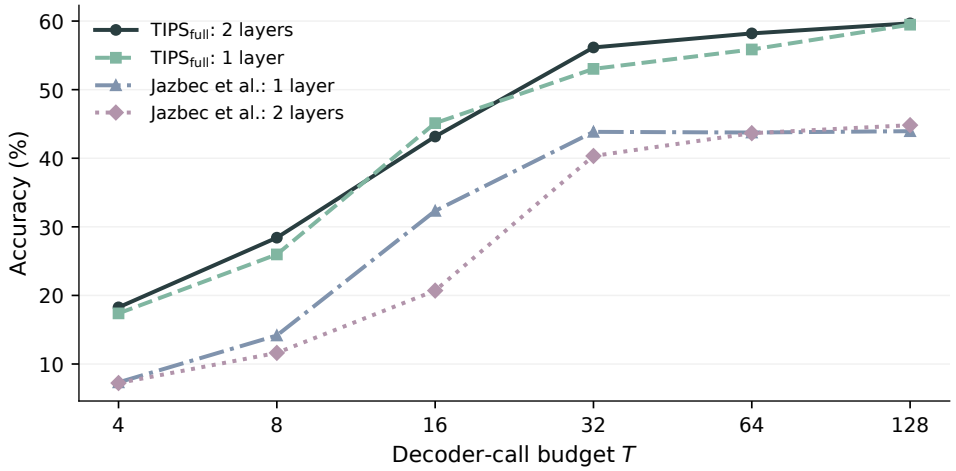


Figure 8: **Full-canvas comparison at matched policy depth.** Validation-set accuracy for one- and two-layer TIPS and the learned unmasking policy of Jazbec et al. (2026). The horizontal axis is the requested decoder-call cap; consumed calls for adaptive policies are reported in Table 12.

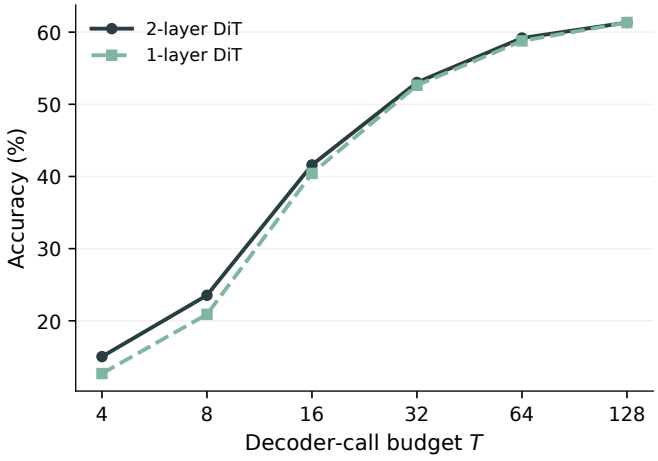


Figure 9: **Policy depth under two-block decoding.** Both curves use the same 2×128 geometry and exact global decoder-call budget. The one-layer variant approaches the two-layer reference at larger budgets and matches it at $T = 128$ in this single-seed comparison.

B.3 Supplementary analyses of visibility and revision

Protocol. These experiments use $\text{TIPS}_{\text{full}}$ and $\text{TIPS}_{\text{block}}$ checkpoints with training seed 123, one evaluation seed, and the full GSM8K and MATH-500 test populations. Except for the explicitly selected corruption-reference pools, each reported mathematical-reasoning accuracy is micro-averaged over all 1,819 problems. The decoder remains frozen and uses greedy token proposals. We preserve each checkpoint's native schedule and timing convention, including the block policy's next-state local time input. All budgets count decoder calls, including deterministic finalization.

B.3.1 Does TIPS select useful reveal and remask positions?

We examine a low-budget case, $\text{TIPS}_{\text{block}}$ at $T = 16$, by intervening on one decision along an otherwise on-distribution trajectory. At within-block phases 0.25, 0.5, and 0.75 in each of the two blocks, we preserve the numbers of revealed and remasked positions but randomize the reveal positions, remask positions, or both. The random selections remain within the corresponding eligible masked or visible sets. Each continuation is evaluated on all test problems, including cases where an intervention leaves the action unchanged.

As seen in Figure 10 and Table 13, randomizing both sets reduces accuracy at all six sites in this case study. For example, at phase 0.25 of the first block, accuracy falls from 44.42% to 40.52%; at phase 0.5 of the second block, it falls to 40.79%. The separate interventions are less uniform: randomizing reveal or remask positions at phase 0.75 of the first block slightly improves accuracy. Thus, the selected example provides evidence that the locations chosen by TIPS can matter beyond their counts, particularly early in the trajectory, while retaining cases where a particular native decision is not beneficial. It is a focused case study rather than a summary of all geometries and budgets.

Table 13: **Position-selection case study:** $\text{TIPS}_{\text{block}}$**,** $T = 16$**.** Test micro-accuracy (%) for a count-preserving intervention at one decision. All six sites for this selected setting are included. Block indices are one-based.

Block	Phase	Native	Random reveal	Random remask	Random both
1	0.25	44.42	41.29	43.65	40.52
1	0.50	44.42	43.32	42.33	42.33
1	0.75	44.42	44.91	44.58	43.87
2	0.25	44.42	41.89	43.05	41.34
2	0.50	44.42	40.79	43.76	40.79
2	0.75	44.42	42.99	44.53	43.21

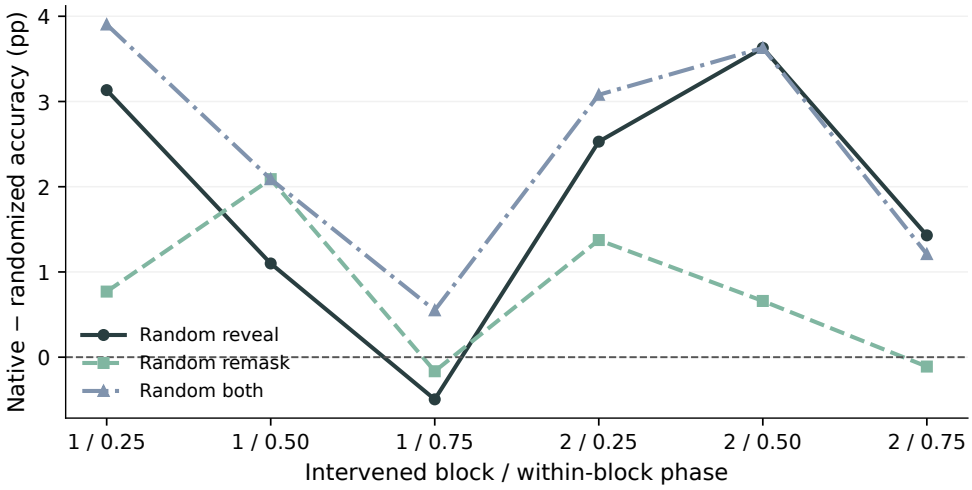


Figure 10: **Count-preserving position randomization for $\mathrm{TIPS}_{\mathrm{block}}$ at $T = 16$.** Positive values mean that native position selection yields higher final accuracy. All six intervention sites are shown, with one-based block indices. Reveal counts and remask counts are separately preserved at the intervened decision.

B.3.2 Recovery from controlled token corruption

We test whether a policy can recover from deliberately introduced token errors. For each policy, we first generate one $T = 32$ response per test problem and retain verifier-accepted references. At within-block phase 0.75, we construct matched clean and corrupted partial states, retaining only references with at least eight eligible visible tokens. We replace $k \in \{1, 2, 4, 8\}$ visible tokens with uniformly drawn vocabulary tokens, excluding the original token, the mask, and tokenizer-defined special tokens. Corruption sets are nested across k, and every k for a given policy and intervention block uses the same reference responses.

The intervention leaves eight calls for the full-canvas policy. For the two-block policy, corrupting the first block leaves four calls in that block and sixteen in the second, while corrupting the second block leaves four calls. Table 14 gives the resulting populations.

Token restoration and answer correctness. TIPS frequently restores the exact substituted token, even though neither the policy nor the decoder receives the reference during continuation. As seen in Table 15, for first-block corruption, $\mathrm{TIPS}_{\mathrm{block}}$ restores 76.53% of substituted tokens at $k = 1$ and 66.55% at $k = 8$; the corresponding second-block rates are 66.35% and 57.23%. The full-canvas rates range from 42.50% to 46.41%. Exact restoration and answer correctness nevertheless measure different outcomes: an alternative token can preserve a correct solution, and restoring one token need not repair every consequence of the corruption. All corrupted conditions have lower answer accuracy than their matched clean continuations.

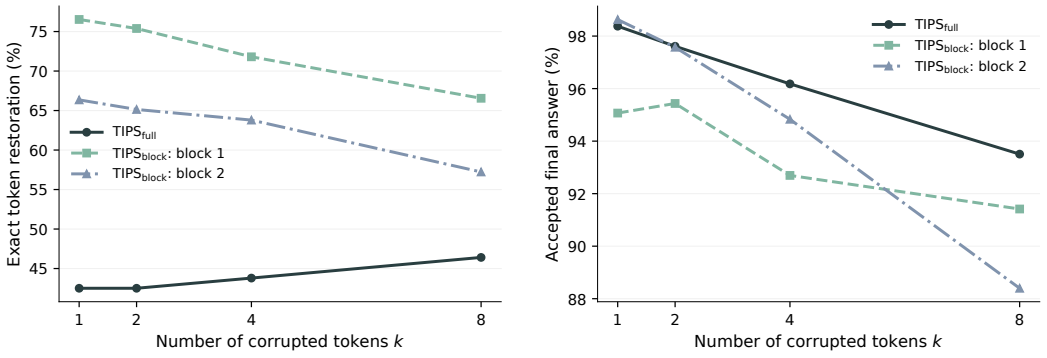
Restoration pathway and collateral drift. Most finally restored tokens last return through a learned reveal rather than deterministic finalization (Table 16). For example, with eight corrupted tokens in the first block, 5,011 of the 5,830 finally restored tokens return through learned reveals, and 819 return at finalization. However, corruption can also affect positions that were not substituted. Table 17 compares untouched-token disagreement with the reference against its paired clean-control rate. Excess disagreement is positive in all twelve corrupted conditions and generally grows with k. These measurements show both recovery and propagation of perturbations.

Table 14: **Reference selection for controlled corruption.** One $T = 32$ reference attempt is made per test problem for each policy. We retain verifier-accepted responses with at least eight eligible visible tokens at the intervention site. The same eligible pool is used for all k within a row.

Policy / active block	Accepted references	Eligible	Too few tokens
$\text{TIPS}_{\text{full}}$	1047/1819	1047	0
$\text{TIPS}_{\text{block}}$, block 1	1095/1819	1095	0
$\text{TIPS}_{\text{block}}$, block 2	1095/1819	948	147

Table 15: **Controlled corruption and recovery.** Answer correctness is measured over eligible problems; exact restoration is measured over the k substituted tokens per problem. $k = 0$ denotes the reconstructed clean continuation. Δ compares answer accuracy with that matched clean control.

Policy / active block	k	Correct / eligible	Answer (%)	Restored (%)	Δ (pp)
$\text{TIPS}_{\text{full}}$	0	1038/1047	99.14	—	+0.00
$\text{TIPS}_{\text{full}}$	1	1030/1047	98.38	42.50	-0.76
$\text{TIPS}_{\text{full}}$	2	1022/1047	97.61	42.50	-1.53
$\text{TIPS}_{\text{full}}$	4	1007/1047	96.18	43.79	-2.96
$\text{TIPS}_{\text{full}}$	8	979/1047	93.51	46.41	-5.64
$\text{TIPS}_{\text{block}}$, block 1	0	1076/1095	98.26	—	+0.00
$\text{TIPS}_{\text{block}}$, block 1	1	1041/1095	95.07	76.53	-3.20
$\text{TIPS}_{\text{block}}$, block 1	2	1045/1095	95.43	75.39	-2.83
$\text{TIPS}_{\text{block}}$, block 1	4	1015/1095	92.69	71.80	-5.57
$\text{TIPS}_{\text{block}}$, block 1	8	1001/1095	91.42	66.55	-6.85
$\text{TIPS}_{\text{block}}$, block 2	0	948/948	100.00	—	+0.00
$\text{TIPS}_{\text{block}}$, block 2	1	935/948	98.63	66.35	-1.37
$\text{TIPS}_{\text{block}}$, block 2	2	925/948	97.57	65.14	-2.43
$\text{TIPS}_{\text{block}}$, block 2	4	899/948	94.83	63.79	-5.17
$\text{TIPS}_{\text{block}}$, block 2	8	838/948	88.40	57.23	-11.60



(a) Exact substituted-token restoration

(b) Final-answer correctness

Figure 11: **Recovery from controlled corruption.** The horizontal axis is the number of substituted tokens. Each curve uses its own accepted-reference population and remaining decoding horizon, as detailed in the text. Panel (a) weights corrupted tokens; panel (b) weights eligible problems.

Table 16: **Where restoration occurs.** The learned-reveal and finalization columns partition the finally restored tokens by their last restoration event. The last column counts tokens restored temporarily but not present in their reference form at termination.

Policy / active block	k	Restored	Learned reveal	Finalization	Transient only
$\text{TIPS}_{\text{full}}$	1	445	434	11	0
$\text{TIPS}_{\text{full}}$	2	890	857	33	0
$\text{TIPS}_{\text{full}}$	4	1834	1767	67	0
$\text{TIPS}_{\text{full}}$	8	3887	3747	140	4
$\text{TIPS}_{\text{block}}$, block 1	1	838	736	102	0
$\text{TIPS}_{\text{block}}$, block 1	2	1651	1442	209	2
$\text{TIPS}_{\text{block}}$, block 1	4	3145	2724	421	3
$\text{TIPS}_{\text{block}}$, block 1	8	5830	5011	819	7
$\text{TIPS}_{\text{block}}$, block 2	1	629	536	93	0
$\text{TIPS}_{\text{block}}$, block 2	2	1235	1067	168	0
$\text{TIPS}_{\text{block}}$, block 2	4	2419	2094	325	0
$\text{TIPS}_{\text{block}}$, block 2	8	4340	3647	693	1

Table 17: **Drift at untouched positions.** We compare identical initially visible, nonspecial positions in each corrupted/clean pair, excluding substitutions. The first two rates and Δ_{token} weight token positions; Δ_{example} averages per-problem drift fractions. Differences are in percentage points. A token mismatch does not necessarily change the final answer.

Policy / active block	k	Corrupt (%)	Clean (%)	Δ_{token}	Δ_{example}
$\text{TIPS}_{\text{full}}$	1	0.15	0.06	+0.08	+0.08
$\text{TIPS}_{\text{full}}$	2	0.19	0.06	+0.13	+0.13
$\text{TIPS}_{\text{full}}$	4	0.30	0.06	+0.24	+0.24
$\text{TIPS}_{\text{full}}$	8	0.54	0.07	+0.47	+0.49
$\text{TIPS}_{\text{block}}$, block 1	1	0.30	0.14	+0.16	+0.16
$\text{TIPS}_{\text{block}}$, block 1	2	0.37	0.14	+0.23	+0.23
$\text{TIPS}_{\text{block}}$, block 1	4	0.73	0.15	+0.58	+0.58
$\text{TIPS}_{\text{block}}$, block 1	8	1.20	0.15	+1.05	+1.06
$\text{TIPS}_{\text{block}}$, block 2	1	0.13	0.02	+0.11	+0.13
$\text{TIPS}_{\text{block}}$, block 2	2	0.23	0.02	+0.21	+0.24
$\text{TIPS}_{\text{block}}$, block 2	4	0.41	0.02	+0.39	+0.53
$\text{TIPS}_{\text{block}}$, block 2	8	0.85	0.02	+0.82	+1.21

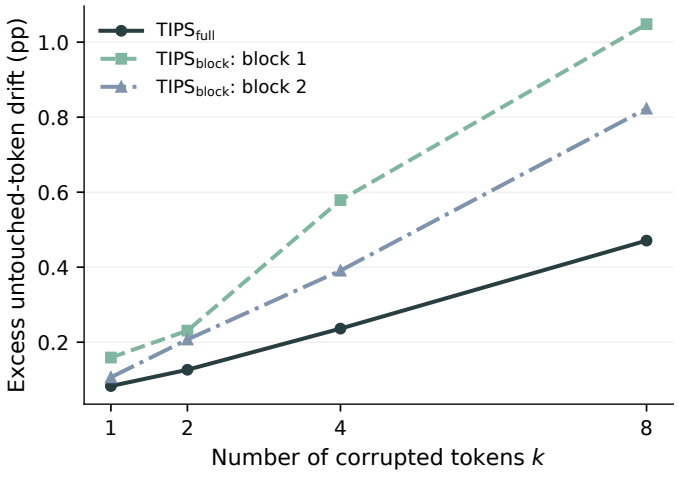


Figure 12: **Excess drift at initially untouched positions.** Token-weighted disagreement with the reference after a corrupted continuation, minus disagreement at the identical untouched positions in its matched clean continuation. Block numbers are one-based.

B.3.3 Remasking is more frequent than token editing

We distinguish a remasking event from a completed token edit. A remasking episode starts when a visible token becomes masked and ends when that position next becomes visible; it is an edit only if

the returning token differs from the original. Table 18 reports remasking under four denominators, preventing trajectory-level incidence from being confused with a per-position rate. Every observed trajectory contains at least one remask, but the fraction of visible position-opportunities that are remasked is approximately 18% for the full-canvas policy and 31–38% for the block policy.

Most remasking episodes do not change token identity. Across the measured budgets, 91.52–95.17% of full-canvas episodes and 92.91–98.16% of blockwise episodes return the original token. Every recorded episode resolves by the end of decoding, and the mean delay before re-reveal is 2.25–3.13 calls for full-canvas decoding and 1.20–1.41 calls for blockwise decoding (Table 19). Thus, remasking often temporarily withdraws context rather than replacing its content. Together with the remasking ablation, these observations motivate the hypothesis that hiding unreliable context can help other positions even when the hidden token ultimately returns unchanged. Another potential explanation is simply due to sampling noise, as each re-masking decision is sampled from a Bernoulli distribution informed by the policy network. The episode counts do not isolate that mechanism from the effects of actual token edits.

Table 18: **How often TIPS remasks.** All 1,819 test problems are included in each row. The four rates use, respectively, visible position-opportunities, all active position-opportunities, learned steps containing a remask, and trajectories containing a remask. A position-opportunity is one active position at one learned decision. Deterministic finalization is excluded.

Policy	T	Remasks	Visible (%)	All positions (%)	Steps (%)	Trajectories (%)
$\text{TIPS}_{\text{full}}$	16	503,803	18.04	7.21	93.29	100.00
$\text{TIPS}_{\text{full}}$	32	1,119,399	17.97	7.75	96.57	100.00
$\text{TIPS}_{\text{full}}$	64	2,457,136	18.71	8.38	97.97	100.00
$\text{TIPS}_{\text{block}}$	16	324,263	30.50	9.95	85.31	100.00
$\text{TIPS}_{\text{block}}$	32	929,911	34.23	13.31	92.91	100.00
$\text{TIPS}_{\text{block}}$	64	2,315,040	38.11	16.04	96.02	100.00

Table 19: **What happens after remasking.** An episode begins when a visible token is remasked and ends at its next reveal. Changed and same-token percentages use completed episodes; every recorded episode completes. Delay is the mean number of global forward-call intervals until re-reveal. Finalized counts episodes ending at deterministic finalization. Special is the fraction of remask events involving tokenizer-defined special tokens, without opportunity adjustment.

Policy	T	Changed	Changed (%)	Same (%)	Delay	Finalized	Special (%)
$\text{TIPS}_{\text{full}}$	16	42,714	8.48	91.52	2.245	29,553	45.36
$\text{TIPS}_{\text{full}}$	32	72,426	6.47	93.53	2.699	16,413	36.39
$\text{TIPS}_{\text{full}}$	64	118,743	4.83	95.17	3.126	8,332	31.96
$\text{TIPS}_{\text{block}}$	16	22,988	7.09	92.91	1.198	48,771	25.16
$\text{TIPS}_{\text{block}}$	32	31,264	3.36	96.64	1.294	31,638	18.69
$\text{TIPS}_{\text{block}}$	64	42,633	1.84	98.16	1.412	17,001	14.76

B.3.4 Continuing heuristic trajectories with TIPS

We extend the main-text handoff experiment to all budgets and switching fractions. A non-adjusted confidence-threshold sampler based on Fast-dLLM first runs for qT global decoder calls, with $q \in \{0, 1/4, 1/2, 3/4, 1\}$, after which TIPS uses the remaining budget. The heuristic reveals eligible masked proposals with confidence strictly above 0.9 and, when none qualifies, reveals the highest-confidence eligible proposal for each example. TIPS resumes at the corresponding point of its native schedule for normalized time input and logit calibration purposes.

For a matched budget comparison, all arms use exactly T real forward passes, including calls that leave the canvas unchanged and deterministic finalizations. In particular, the $q = 1$ endpoint is this fixed-horizon heuristic. We retain both endpoints and all three hybrid settings in Table 20.

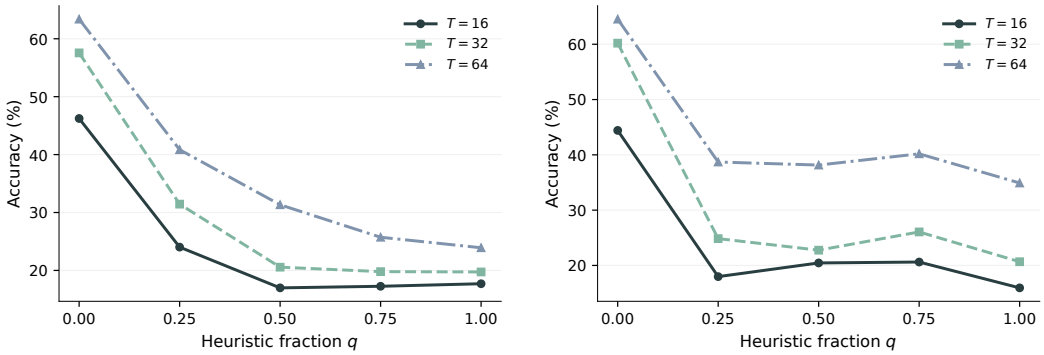
Sixteen of the eighteen hybrid conditions improve over the matched heuristic endpoint. At $T = 64$, full-canvas handoff at $q = 1/4$ raises accuracy from 23.91% to 40.85%, while blockwise handoff at $q = 3/4$ raises it from 34.91% to 40.19%. The paired counts show that these gains can include both repaired and newly incorrect final answers: the former full-canvas condition changes 491 heuristic failures into correct answers while changing 183 heuristic successes into failures. No hybrid exceeds

pure TIPS at the same budget. The result therefore supports TIPS's ability to improve continuations from another sampler, rather than recommending a heuristic prefix over starting with TIPS.

The results also support a comparison at the remaining budget. For $T = 64$ and $q = 1/2$, handoff leaves 32 calls for TIPS and yields 31.34% and 38.15% accuracy for $\text{TIPS}_{\text{full}}$ and $\text{TIPS}_{\text{block}}$, respectively. Their from-scratch 32-call runs instead achieve 57.56% and 60.20%, suggesting that restarting from an all-mask canvas can be preferable to retaining the heuristic state.

Table 20: **Complete heuristic-to-TIPS handoff results.** Test micro-accuracy (%) at all global switching fractions. $q = 0$ is pure TIPS; $q = 1$ is the matched fixed-horizon non-adjusted confidence-threshold heuristic. Every row uses exactly T real decoder calls, including no-op and finalization calls.

Policy	T	$q = 0$	$q = 1/4$	$q = 1/2$	$q = 3/4$	$q = 1$
$\text{TIPS}_{\text{full}}$	16	46.23	24.02	16.99	17.26	17.70
$\text{TIPS}_{\text{full}}$	32	57.56	31.45	20.56	19.79	19.74
$\text{TIPS}_{\text{full}}$	64	63.39	40.85	31.34	25.73	23.91
$\text{TIPS}_{\text{block}}$	16	44.42	17.98	20.45	20.62	15.94
$\text{TIPS}_{\text{block}}$	32	60.20	24.85	22.76	26.06	20.67
$\text{TIPS}_{\text{block}}$	64	64.54	38.70	38.15	40.19	34.91



(a) $\text{TIPS}_{\text{full}}$ (b) $\text{TIPS}_{\text{block}}$

Figure 13: **Heuristic prefixes followed by TIPS.** Test micro-accuracy at every evaluated switching fraction, including pure TIPS ($q = 0$) and the matched heuristic ($q = 1$). Each curve has a fixed total decoder-call budget.

Table 21: **Paired final-answer changes after handoff.** Counts compare each hybrid with the matched pure-heuristic endpoint on the same 1,819 problems. Correctness refers to terminal answers. Both beneficial and adverse transitions are retained.

Policy	T	q	Wrong → correct	Correct → wrong	Δ (pp)
$\text{TIPS}_{\text{full}}$	16	0.25	267	152	+6.32
$\text{TIPS}_{\text{full}}$	16	0.50	91	104	-0.71
$\text{TIPS}_{\text{full}}$	16	0.75	42	50	-0.44
$\text{TIPS}_{\text{full}}$	32	0.25	394	181	+11.71
$\text{TIPS}_{\text{full}}$	32	0.50	137	122	+0.82
$\text{TIPS}_{\text{full}}$	32	0.75	50	49	+0.05
$\text{TIPS}_{\text{full}}$	64	0.25	491	183	+16.93
$\text{TIPS}_{\text{full}}$	64	0.50	284	149	+7.42
$\text{TIPS}_{\text{full}}$	64	0.75	88	55	+1.81
$\text{TIPS}_{\text{block}}$	16	0.25	186	149	+2.03
$\text{TIPS}_{\text{block}}$	16	0.50	195	113	+4.51
$\text{TIPS}_{\text{block}}$	16	0.75	143	58	+4.67
$\text{TIPS}_{\text{block}}$	32	0.25	246	170	+4.18
$\text{TIPS}_{\text{block}}$	32	0.50	184	146	+2.09
$\text{TIPS}_{\text{block}}$	32	0.75	163	65	+5.39
$\text{TIPS}_{\text{block}}$	64	0.25	316	247	+3.79
$\text{TIPS}_{\text{block}}$	64	0.50	205	146	+3.24
$\text{TIPS}_{\text{block}}$	64	0.75	167	71	+5.28

B.3.5 Repairing completed non-adjusted Fast-dLLM responses

We next test whether TIPS can improve responses that another sampler has already completed, rather than continue a partially decoded trajectory. We first run non-adjusted Fast-dLLM on frozen LLaDA-8B-Instruct with 8×32 decoding, confidence threshold 0.9, no caching, and a global cap of 256 decoder calls. We then apply the frozen $\text{TIPS}_{\text{full}}$ policy to every completed 256-token canvas on the full GSM8K and MATH-500 test sets, using exactly four additional decoder evaluations. Repair retains the native schedule and follows the final four calls of a virtual 32-call horizon: three policy transitions receive normalized times $28/32$, $29/32$, and $30/32$, with next-state visibility targets $((t+1)/32)^{\exp(\gamma_{\text{full}})}$ for $t \in \{28, 29, 30\}$, followed by deterministic finalization. The decoder and policy remain frozen, and correctness feedback is used only for scoring. As shown in Table 22 and Figure 14, micro-accuracy increases from 67.01% to 67.29%, a gain of 0.27 percentage points. TIPS corrects five of the 600 initially incorrect answers—four on GSM8K and one on MATH-500—while all $1{,}219$ initially correct answers remain correct. This single-seed result shows a modest improvement from revisiting completed responses, extending revision beyond unfinished trajectories.

Table 22: **Four-call repair of completed non-adjusted Fast-dLLM responses.** Test accuracy before and after applying $\text{TIPS}_{\text{full}}$ to the same saved responses, with training and evaluation seed 123. W→C and C→W count wrong-to-correct and correct-to-wrong final-answer transitions, respectively. Every response is repaired, regardless of its original correctness. Non-adjusted Fast-dLLM has a 256-call global cap but terminates naturally on every problem. Differences are computed from unrounded accuracies.

		Accuracy (%)			Answer transitions	
Dataset	N	Non-adjusted Fast-dLLM	$+\text{TIPS}_{\text{full}}$	Δ (pp)	W→C	C→W
GSM8K	1,319	79.30	79.61	+0.30	4	0
MATH-500	500	34.60	34.80	+0.20	1	0
Micro-average	1,819	67.01	**67.29**	**+0.27**	5	0

B.4 Generalization beyond the training configuration

Protocol. We evaluate transfer with one training lineage and one evaluation seed, a 256-token response, and budgets $T \in \{16, 32, 64\}$. Mathematical-reasoning results use the complete GSM8K

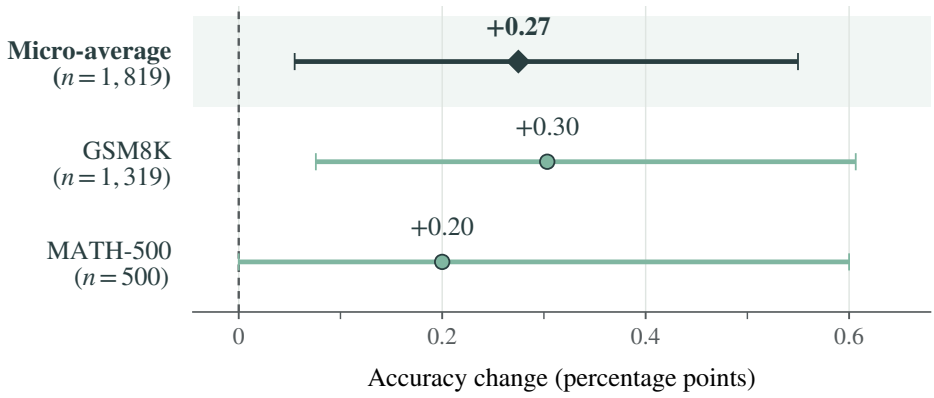


Figure 14: **Accuracy changes after four additional repair calls.** Points show the paired accuracy change from completed non-adjusted Fast-dLLM responses to their $\mathrm{TIPS}_{\mathrm{full}}$ revisions. Whiskers are 95% paired bootstrap intervals from $10{,}000$ resamples, sampling problems within each benchmark at its original size. The dashed line marks no accuracy change.

and MATH-500 test sets. Coding results use all 164 HumanEval and 500 MBPP test tasks and report pass@1 from one generation per task. Policy weights remain frozen for the zero-shot experiments; coding-trained policies are evaluated separately. Baseline settings are fixed independently of coding test outcomes. We retain a reduced baseline roster – selected based on achieving the best non-TIPS results in mathematical reasoning in Table 1 – due to computational constraints.

B.4.1 Zero-shot transfer to Dream-7B

We apply the LLaDA-trained $\mathrm{TIPS}_{\mathrm{full}}$ policy to Dream-v0-Instruct-7B without retraining. This comparison uses a single 1×256 canvas for both TIPS and Fast-dLLM and preserves the Dream logit-to-position alignment. As the policy observes confidence features rather than token identities or decoder hidden states, its input interface does not require a shared vocabulary or hidden-state dimension.

The transferred policy reaches 35.18%, 46.56%, and 52.39% test micro-accuracy at $T = 16, 32, 64$, compared with 22.87%, 27.43%, and 29.96% for the matched full-canvas Fast-dLLM baseline. Table 23 expands the main-text comparison with per-dataset scores and actual mean decoder calls. These results demonstrate transfer to another frozen backbone in this setting; they do not require adapting the policy on Dream outputs or selecting its parameters from Dream test scores.

Table 23: **Dream transfer by dataset and consumed compute.** Columns correspond to requested budgets 16, 32, and 64. Both methods decode a single 256-token canvas. The pooled row weights individual problems equally.

Dataset	Method	Accuracy (%)			Mean decoder calls		
		16	32	64	16	32	64
GSM8K	$\mathrm{TIPS}_{\mathrm{full}}$	41.77	55.72	63.84	16.00	32.00	64.00
GSM8K	Fast-dLLM	25.32	30.78	34.57	15.48	32.75	66.33
MATH-500	$\mathrm{TIPS}_{\mathrm{full}}$	17.80	22.40	22.20	16.00	32.00	64.00
MATH-500	Fast-dLLM	16.40	18.60	17.80	17.50	30.70	58.21
Micro-average	$\mathrm{TIPS}_{\mathrm{full}}$	35.18	46.56	52.39	16.00	32.00	64.00
Micro-average	Fast-dLLM	22.87	27.43	29.96	16.03	32.19	64.10

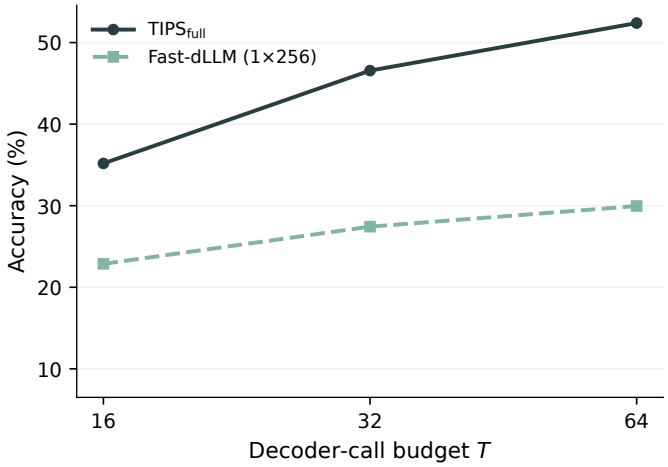


Figure 15: **Zero-shot backbone transfer on a matched full canvas.** Test micro-accuracy for the frozen LLaDA-trained $\mathrm{TIPS}_{\mathrm{full}}$ policy and Fast-dLLM on Dream-7B. Both use 1×256 decoding. The horizontal axis is the decoder-call budget; actual baseline consumption is reported in Table 23.

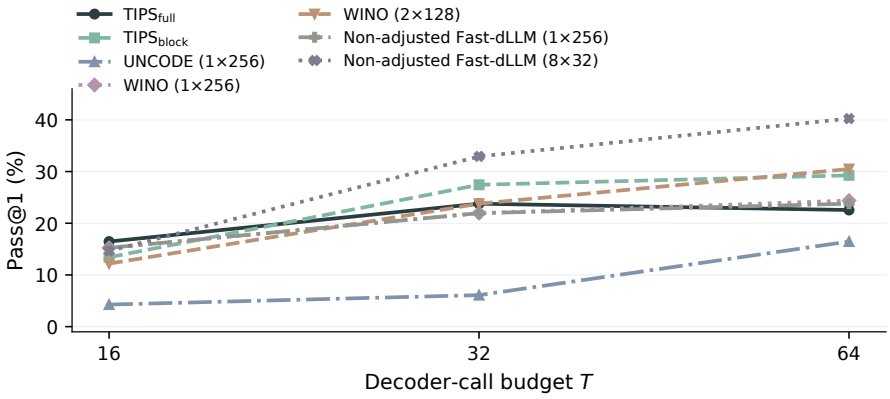
B.4.2 Zero-shot transfer from mathematics to coding

We next apply the math-trained full-canvas and blockwise policies to coding without adaptation. Unlike the backbone-transfer experiment, this changes the task domain while retaining LLaDA. Table 24 reports every retained baseline individually: UNCODE/PC-Sampler on 1×256, WINO on 1×256 and 2×128, and Non-adjusted Fast-dLLM on 1×256 and 8×32. These baselines retain their fixed mathematical-reasoning presets or assets rather than being tuned on coding test scores.

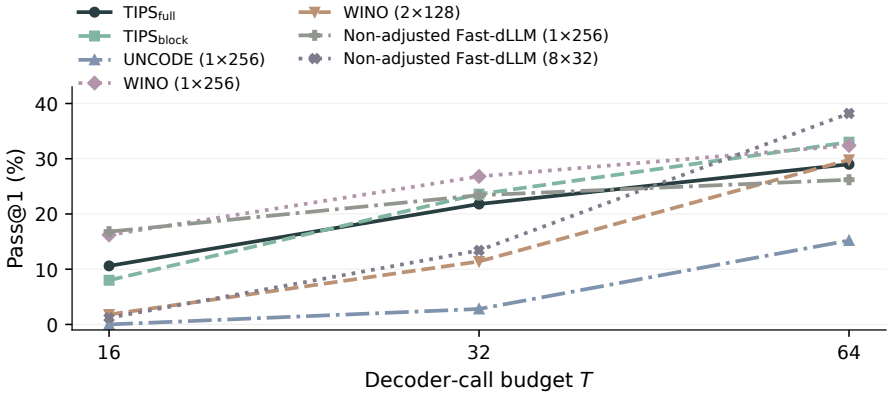
Transfer is mixed. At $T = 16$, $\mathrm{TIPS}_{\mathrm{full}}$ achieves the highest measured HumanEval score among these methods, 16.46%, but it does not maintain that advantage as the budget grows. At $T = 64$, Non-adjusted Fast-dLLM with 8×32 decoding reaches 40.24% on HumanEval, compared with 22.56% and 29.27% for the full and block TIPS policies. On MBPP, the two policies reach 29.00% and 33.00%, compared with 38.20% for that baseline. Thus, the mathematical-reasoning policy transfers nontrivially, but the gains observed in its training domain do not translate into consistent superiority on coding.

Table 24: **Zero-shot coding transfer.** TIPS uses frozen math-trained policies. Pass@1 (%), one generation per test task, against each of the five retained baselines individually. Baseline presets/assets are inherited from mathematical reasoning. The 8×32 baseline uses a shared global cap. Adaptive methods may consume fewer than T calls.

Method	Geometry	HumanEval (164)			MBPP (500)		
		$T = 16$	$T = 32$	$T = 64$	$T = 16$	$T = 32$	$T = 64$
UNCODE / PC-Sampler	1×256	4.27	6.10	16.46	0.00	2.80	15.20
WINO	1×256	15.24	21.95	24.39	16.20	26.80	32.40
WINO	2×128	12.20	23.78	30.49	1.80	11.40	29.80
Non-adjusted Fast-dLLM	1×256	15.24	21.95	23.78	16.80	23.40	26.20
Non-adjusted Fast-dLLM	8×32	14.63	32.93	40.24	1.20	13.40	38.20
$\mathrm{TIPS}_{\mathrm{full}}$	1×256	16.46	23.78	22.56	10.60	21.80	29.00
$\mathrm{TIPS}_{\mathrm{block}}$	2×128	13.41	27.44	29.27	8.00	23.60	33.00



(a) HumanEval



(b) MBPP

Figure 16: **Zero-shot coding transfer against individual baselines.** One-generation pass@1 versus log decoder-call cap. TIPS uses its frozen math-trained policies.

B.4.3 TRAINING TIPS FOR CODING

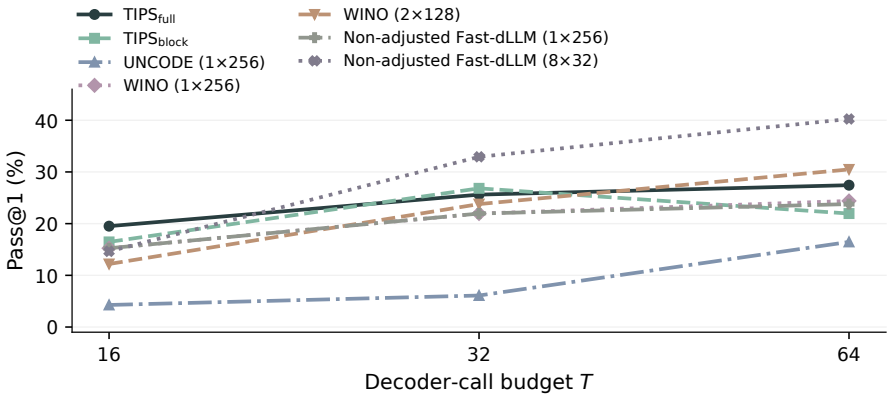
We separately train full-canvas and blockwise policies for coding while keeping the decoder frozen, using supervised warmup followed by an RLVR continuation to the fixed 935-update endpoint. Table 25 compares these policies with the same five baselines, so changes relative to the zero-shot table reflect the policy's domain adaptation.

Coding training improves the full-canvas policy in every measured dataset–budget condition. Its HumanEval pass@1 increases to 19.51%, 25.61%, and 27.44%, while its MBPP score increases to 24.00%, 31.80%, and 37.80%. The largest gains over the corresponding math-trained full-canvas policy occur on MBPP, reaching 13.40 percentage points at $T = 16$. The coding-trained full policy exceeds every retained baseline on HumanEval at $T = 16$ and on MBPP at $T = 16$ and 32. Nevertheless, Non-adjusted Fast-dLLM with 8×32 decoding remains stronger on HumanEval at $T = 32$ and 64, and slightly stronger on MBPP at $T = 64$.

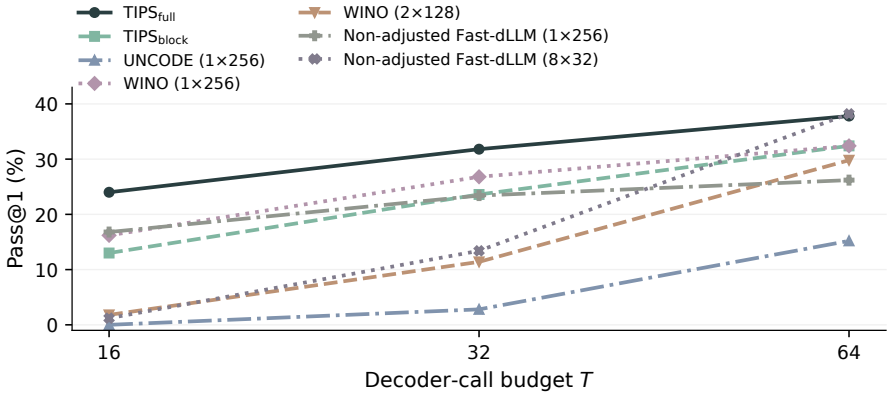
The block policy responds less consistently to coding training: it improves at $T = 16$ on both tasks but falls below its zero-shot counterpart at $T = 64$. These results support domain-specific policy training, especially for full-canvas decoding, while leaving room for a coding-specific recipe and hyperparameter study. We do not infer that the present coding configuration is optimal. Table 26 additionally reports actual mean calls, since adaptive baselines may finish below their requested caps.

Table 25: **Coding-trained TIPS.** TIPS policies are trained separately on coding; the decoder remains frozen. Pass@1 (%), one generation per test task, against each of the five retained baselines individually. Baseline presets/assets and TIPS settings are inherited from mathematical reasoning and untuned. The 8×32 baseline uses a shared global cap. Adaptive methods may consume fewer than T calls.

Method	Geometry	HumanEval (164)			MBPP (500)		
		$T = 16$	$T = 32$	$T = 64$	$T = 16$	$T = 32$	$T = 64$
UNCODE / PC-Sampler	1×256	4.27	6.10	16.46	0.00	2.80	15.20
WINO	1×256	15.24	21.95	24.39	16.20	26.80	32.40
WINO	2×128	12.20	23.78	30.49	1.80	11.40	29.80
Non-adjusted Fast-dLLM	1×256	15.24	21.95	23.78	16.80	23.40	26.20
Non-adjusted Fast-dLLM	8×32	14.63	32.93	40.24	1.20	13.40	38.20
$\mathrm{TIPS}_{\mathrm{full}}$	1×256	19.51	25.61	27.44	24.00	31.80	37.80
$\mathrm{TIPS}_{\mathrm{block}}$	2×128	16.46	26.83	21.95	13.00	23.60	32.40



(a) HumanEval



(b) MBPP

Figure 17: **Coding-trained TIPS against individual baselines.** The decoder is unchanged; only the TIPS policies are trained for coding. Baselines are identical to those in Figure 16. Curves show one evaluation sample.

Table 26: **Consumed compute in coding experiments.** Mean decoder calls at the indicated caps, for the same conditions as Tables 24 and 25. Both math-trained and coding-trained TIPS use the exact requested horizon.

Method	Geometry	HumanEval (164)			MBPP (500)		
		$T = 16$	$T = 32$	$T = 64$	$T = 16$	$T = 32$	$T = 64$
UNCODE / PC-Sampler	1×256	16.00	32.00	64.00	16.00	32.00	64.00
WINO	1×256	15.84	25.26	31.24	15.94	25.20	33.71
WINO	2×128	16.00	24.84	31.19	16.00	28.73	41.09
Non-adjusted Fast-dLLM	1×256	13.21	18.83	21.01	14.95	23.45	28.65
Non-adjusted Fast-dLLM	8×32	15.93	28.51	37.82	16.00	31.76	53.07
TIPS (both policies)	native	16.00	32.00	64.00	16.00	32.00	64.00

B.4.4 Zero-shot transfer across block sizes

We reuse the full-canvas-trained policy for target geometries 1×256, 2×128, 4×64, and 8×32, without retraining or fitting a geometry-specific schedule. The total response length remains 256 tokens, and the global decoder budget is divided equally across blocks, including one deterministic finalization per block. Consequently, increasing the number of blocks changes both the visible-context structure and the number of learned decisions available within each block.

Transfer is strongest when sufficient within-block compute remains available. At $T = 64$, the original full-canvas and transferred two-block configurations reach 62.67% and 62.89% micro-accuracy, respectively; the four- and eight-block configurations retain 59.81% and 55.25%. At $T = 16$, however, accuracy decreases from 47.00% on the full canvas to 22.43% with eight blocks. In the latter case, each block receives only two calls: one learned decision and one finalization. The results therefore show useful transfer to smaller blocks, particularly at larger budgets, while also exposing the cost of reducing the number of decisions per block.

Table 27: **Frozen-policy block-size transfer.** All rows use the full-canvas-trained checkpoint, without block-specific policy training. Values are test accuracy (%). The total 256-token response and total decoder-call budget remain fixed.

Target geometry	Dataset	$T = 16$	$T = 32$	$T = 64$
1×256	GSM8K	57.39	71.72	74.75
1×256	MATH-500	19.60	25.80	30.80
1×256	Micro-average	47.00	59.10	62.67
2×128	GSM8K	44.73	66.41	75.74
2×128	MATH-500	15.80	25.80	29.00
2×128	Micro-average	36.78	55.25	62.89
4×64	GSM8K	27.90	57.01	72.02
4×64	MATH-500	13.00	20.40	27.60
4×64	Micro-average	23.80	46.95	59.81
8×32	GSM8K	26.46	40.71	66.26
8×32	MATH-500	11.80	16.40	26.20
8×32	Micro-average	22.43	34.03	55.25

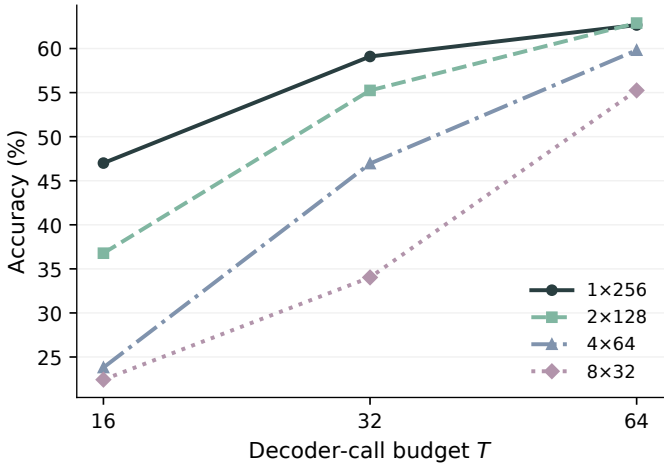


Figure 18: **Frozen-policy transfer across decoding geometries.** Every curve uses the same full-canvas-trained checkpoint and the same global 256-token response length. The decoder-call budget is shared across blocks.

B.4.5 Sensitivity to decoder proposal temperature

We vary token-proposal temperature over $\{0, 0.1, 0.5, 1.0\}$ for the frozen full-canvas policy, leaving policy-action sampling and the schedule unchanged. Finalization remains greedy in every condition. Small proposal noise is competitive: temperature 0.1 gives 48.32% at $T = 16$ and 63.33% at $T = 64$, slightly above the corresponding greedy scores of 47.00% and 62.67%, while greedy proposals are better at $T = 32$. Increasing temperature to 1.0 reduces accuracy at all three budgets, with the largest loss at $T = 16$ (37.77% versus 47.00%). These measurements support greedy decoding as a strong default.

Table 28: **Sensitivity to proposal temperature.** Frozen $\text{TIPS}_{\text{full}}$ on the pooled math test set. Only token-proposal temperature changes; finalization remains greedy. Values are single-seed micro-accuracy (%).

Proposal temperature	$T = 16$	$T = 32$	$T = 64$
0.0	47.00	59.10	62.67
0.1	48.32	58.11	63.33
0.5	46.89	56.02	62.62
1.0	37.77	51.35	60.75

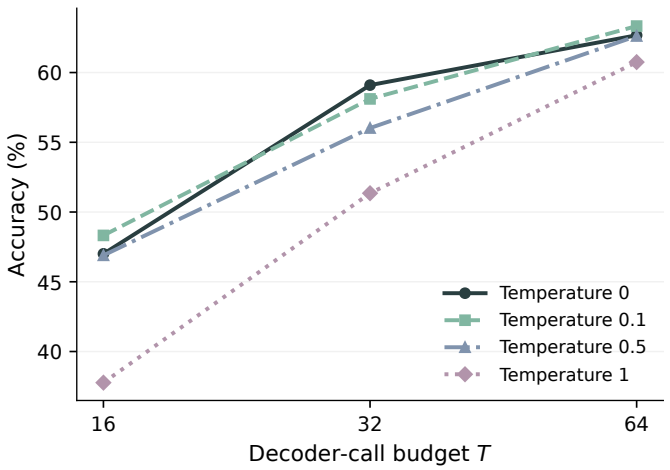


Figure 19: **Sensitivity to proposal temperature.** Test micro-accuracy for the same frozen $\mathrm{TIPS}_{\mathrm{full}}$ policy. Only token proposals change temperature; finalization remains greedy, and visibility actions retain the same sampling rule.

B.5 Training diagnostics

We report training traces separately from downstream test performance. In the mathematical-reasoning run, supervised warmup comprises an initial 3,230-update phase followed by a final 4,074-update phase after refreshing the state dataset. Figure 20 shows the recorded optimized loss in each phase.

Figure 21 additionally reports recomputed mathematical-reasoning validation observations on a fixed dataset of 36,363 states from 1,173 reserved training examples. These are post-hoc diagnostic evaluations and use a different population from the 1,024-example validation ablations. Figure 22 retains the coding run's recorded training and validation loss curves. All three figures concern supervised warmup of the small policy network.

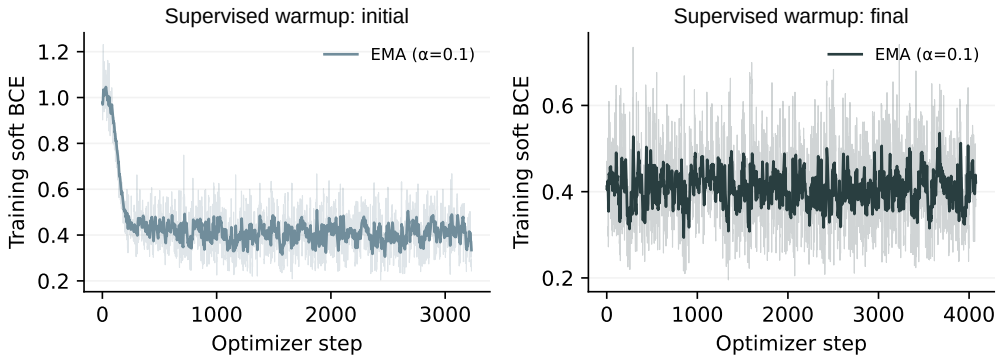


Figure 20: **Supervised warmup on mathematics.** Recorded training soft binary cross-entropy in the initial and final warmup phases. Faint traces are raw observations; the darker curves are exponential moving averages with smoothing coefficient 0.1. The phases have separate optimizer-step axes.

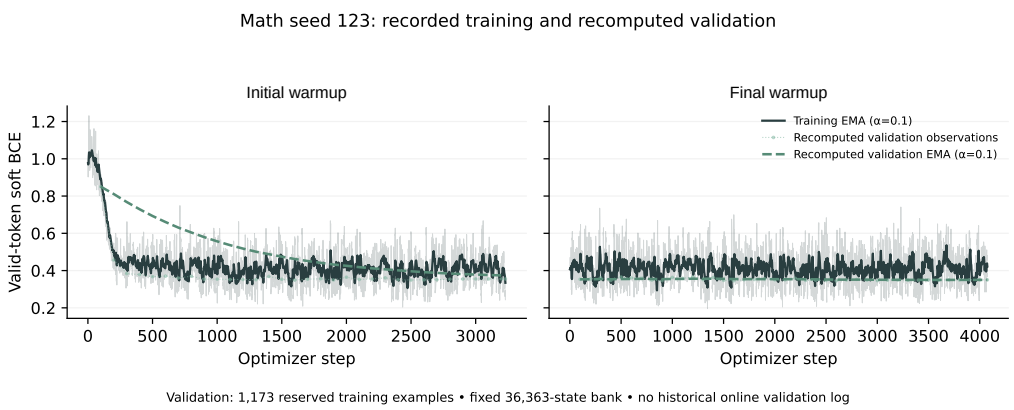


Figure 21: **Recomputed warmup validation curves.** Recorded mathematical-reasoning training loss and validation observations recomputed on a fixed state bank, with exponential moving averages for visualization.

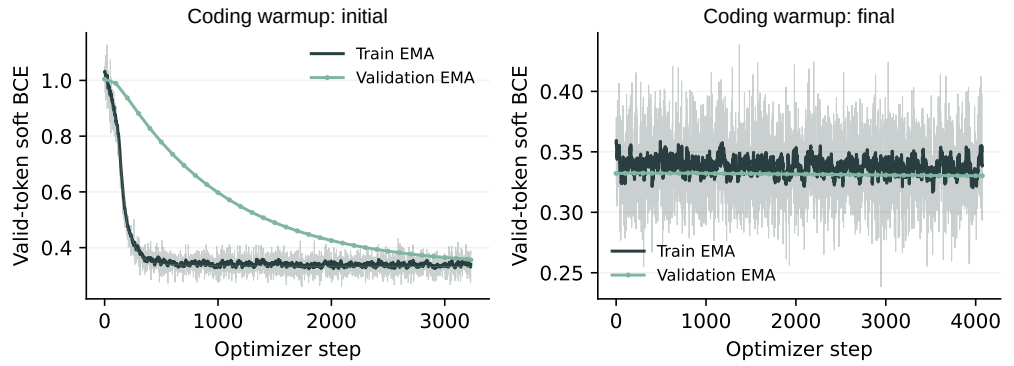
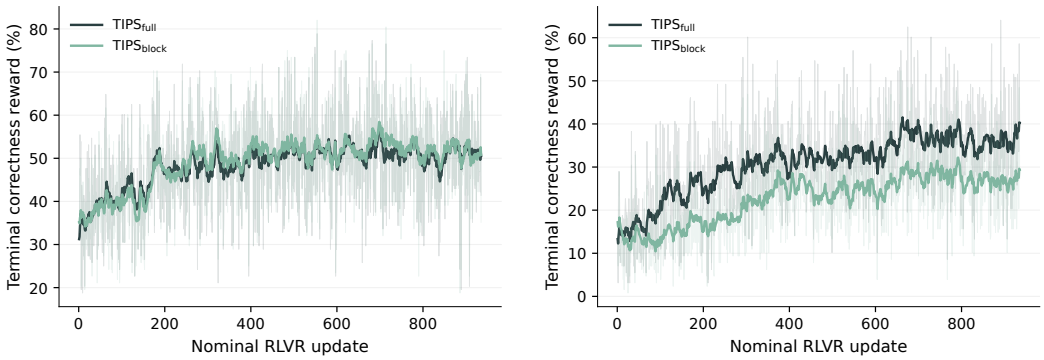


Figure 22: **Supervised warmup on coding.** Supplied training and validation soft binary cross-entropy traces for the initial and final warmup phases. The displayed averages summarize within-run loss observations.

RLVR trajectories. Figure 23 shows raw terminal correctness rewards and exponential moving averages for the mathematical-reasoning and coding runs. Each nominal update contains 128 terminal outcomes, and each continuation ends at nominal update 935. Table 29 summarizes the first and last 50 updates without smoothing. For mathematics, mean reward increases from approximately 38% to 50% for both geometries. For coding, it increases from 16.28% to 37.55% for the full policy and from 13.66% to 27.41% for the block policy. In the coding full-canvas run, two nominal updates are skipped by the original optimizer rule, leaving 933 optimizer steps; the plot retains all 935 nominal updates and their recorded outcomes.

Table 29: **RLVR training diagnostics.** Mean raw terminal-correctness reward (%) over the first and last 50 recorded updates of each run.

Domain	Policy	First 50 updates	Last 50 updates
Math	$\text{TIPS}_{\text{full}}$	38.19	50.73
Math	$\text{TIPS}_{\text{block}}$	37.55	50.41
Coding	$\text{TIPS}_{\text{full}}$	16.28	37.55
Coding	$\text{TIPS}_{\text{block}}$	13.66	27.41



(a) Mathematical reasoning (b) Coding

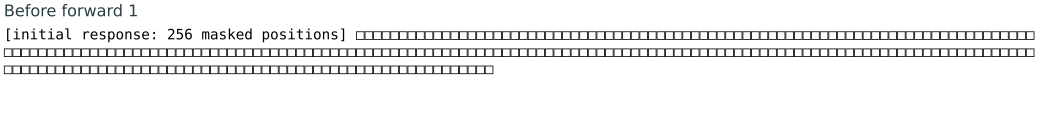
Figure 23: **RLVR training curves.** Terminal correctness reward per nominal update. Faint lines show the raw observations; darker lines show exponential moving averages with smoothing coefficient 0.1.

B.6 Illustrative decoding trajectories

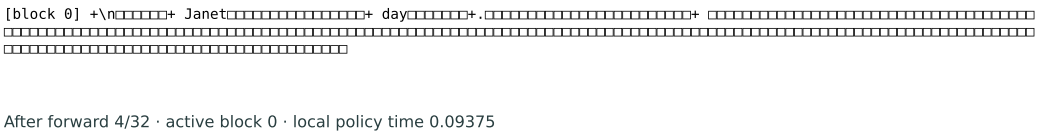
We show four trajectories with verifier-accepted final answers—one for each policy and mathematical-reasoning dataset—and one unsuccessful full-canvas trajectory. All use $T = 32$. These examples were selected for readable outputs and visible ordinary-token edits. In particular, an accepted terminal answer does not certify every intermediate reasoning step or establish that a displayed edit caused the success.

The canvases retain the recorded token-level displays. A square denotes a masked position; a prefixed $+$ marks a reveal, a prefixed $-$ a remask, and a prefixed $*$ a changed re-reveal. These prefixes are display annotations. Each snapshot is labeled by the number of decoder calls already consumed, while active-block indices in the recorded canvases are zero-based. Deterministic finalizations are identified explicitly, and earlier finalized blocks remain frozen. The prompt instructions are omitted from the figures, while the task is given in each caption.

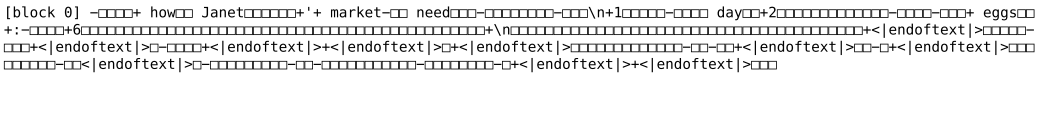
Before forward 1



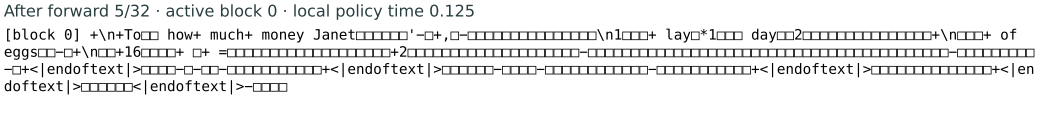
After forward 1/32 · active block 0 · local policy time 0.0



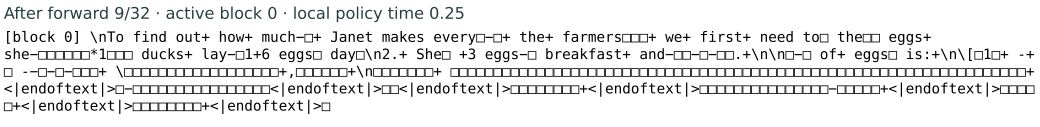
After forward 4/32 · active block 0 · local policy time 0.09375



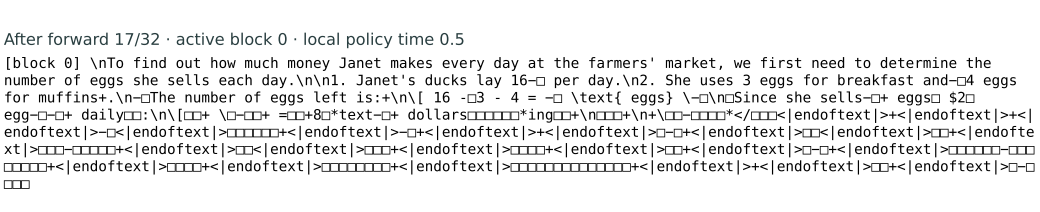
After forward 5/32 · active block 0 · local policy time 0.125



After forward 9/32 · active block 0 · local policy time 0.25



After forward 17/32 · active block 0 · local policy time 0.5



After forward 32/32 · active block 0 · local policy time — · DETERMINISTIC FINALIZATION

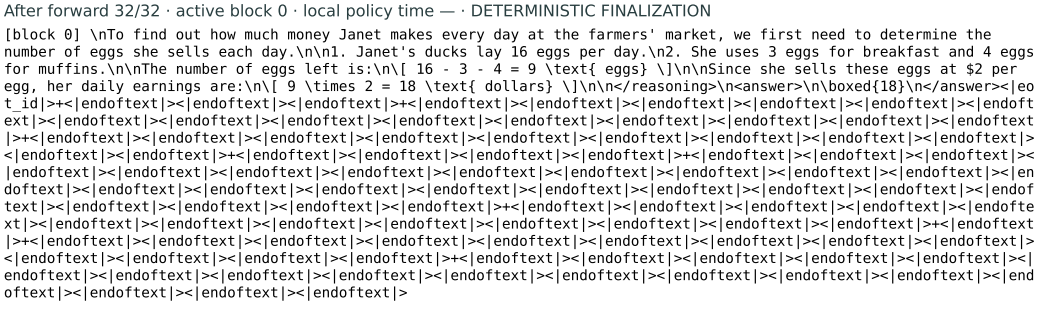
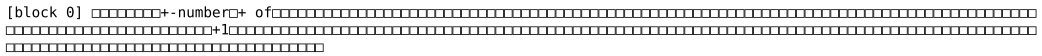


Figure 24: **Accepted full-canvas trajectory on GSM8K.** Janet's ducks lay 16 eggs daily; she uses three for breakfast and four for muffins, then sells the remainder for two dollars each. The final response correctly computes $(16 - 3 - 4) \times 2 = 18$ dollars. The snapshots show progressive visibility, remasking, and finalization.

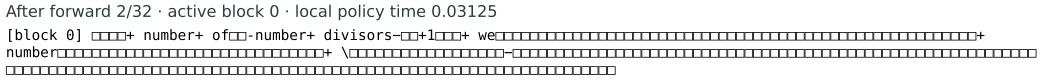
Before forward 1



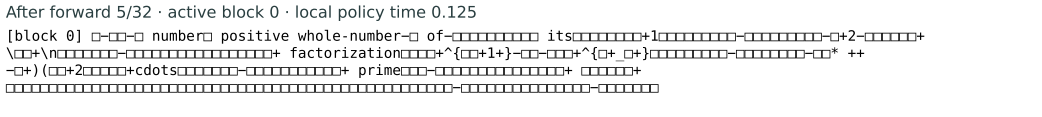
After forward 1/32 · active block 0 · local policy time 0.0



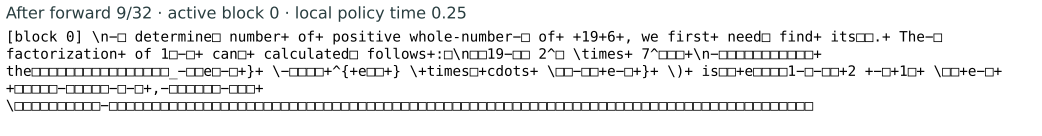
After forward 2/32 · active block 0 · local policy time 0.03125



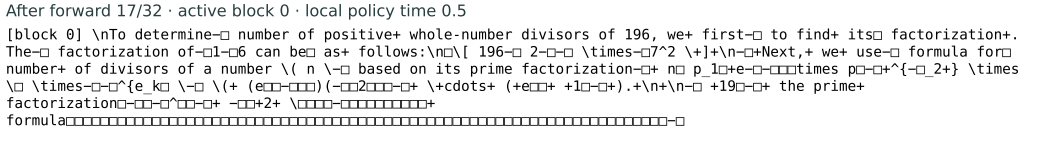
After forward 5/32 · active block 0 · local policy time 0.125



After forward 9/32 · active block 0 · local policy time 0.25



After forward 17/32 · active block 0 · local policy time 0.5



After forward 32/32 · active block 0 · local policy time — · DETERMINISTIC FINALIZATION

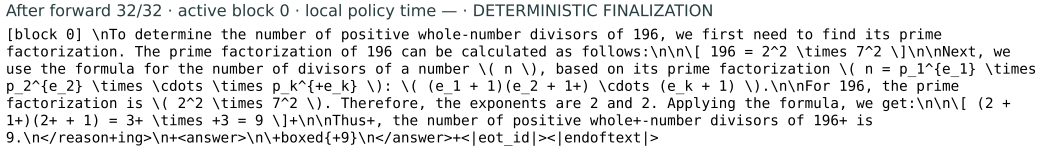
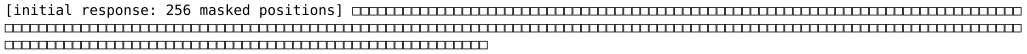
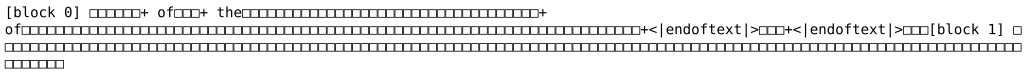


Figure 25: **Accepted full-canvas trajectory on MATH-500.** The task asks how many positive whole-number divisors 196 has. The response arrives at $196 = 2^2 \times 7^2$ and the accepted answer $(2 + 1)(2 + 1) = 9$. Token-level reveal and remask annotations are retained in the displayed intermediate and final canvases.

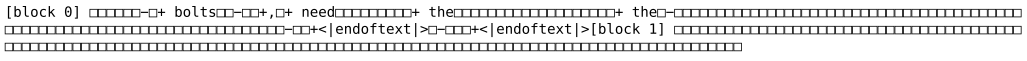
Before forward 1



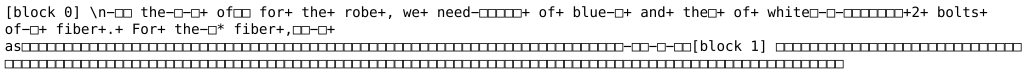
After forward 1/32 · active block 0 · local policy time 0.0625



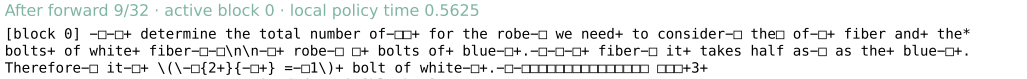
After forward 2/32 · active block 0 · local policy time 0.125



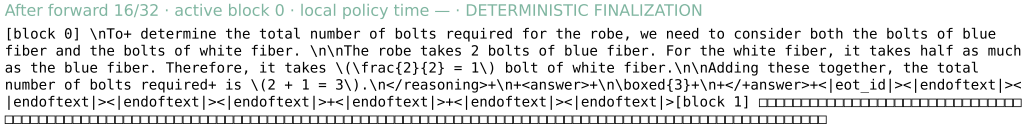
After forward 4/32 · active block 0 · local policy time 0.25



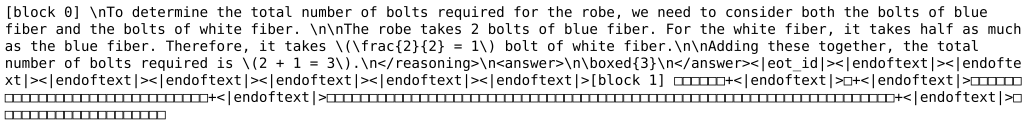
After forward 9/32 · active block 0 · local policy time 0.5625



After forward 16/32 · active block 0 · local policy time — · DETERMINISTIC FINALIZATION



After forward 17/32 · active block 1 · local policy time 0.0625



After forward 32/32 · active block 1 · local policy time — · DETERMINISTIC FINALIZATION

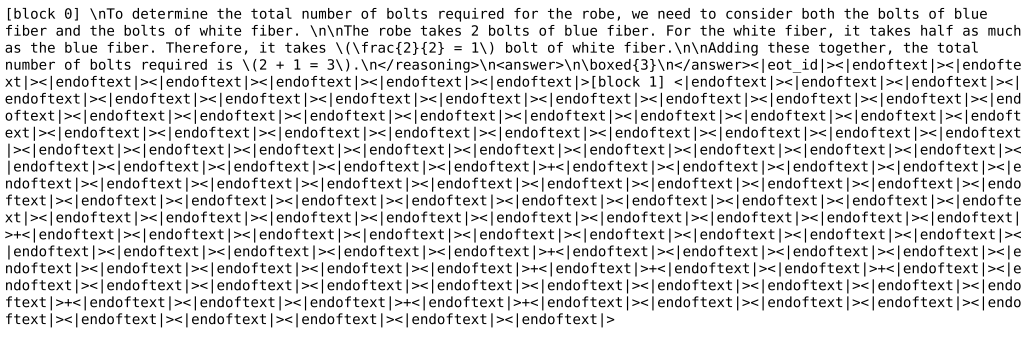
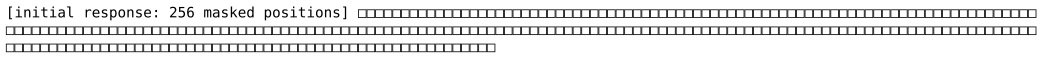


Figure 26: **Accepted blockwise trajectory on GSM8K.** A robe requires two bolts of blue fiber and half as much white fiber, for a total of three bolts. The first block contains the solution and is finalized after 16 calls; the second block is then decoded under the same global budget. This example also illustrates that TIPS continues to the allocated horizon even when the answer-bearing text appears earlier.

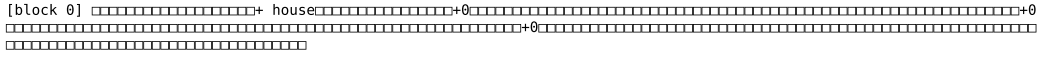
Before forward 1

[initial response: 256 masked positions] □□□□

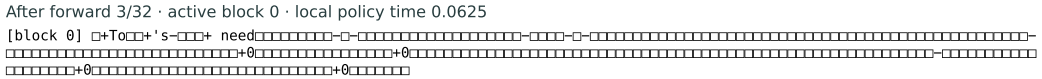
Before forward 1



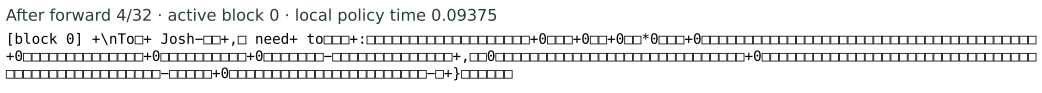
After forward 1/32 · active block 0 · local policy time 0.0



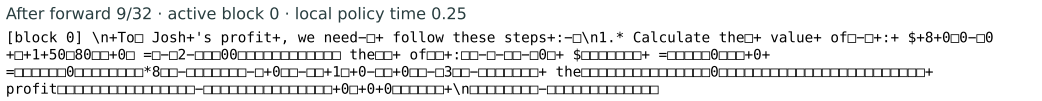
After forward 3/32 · active block 0 · local policy time 0.0625



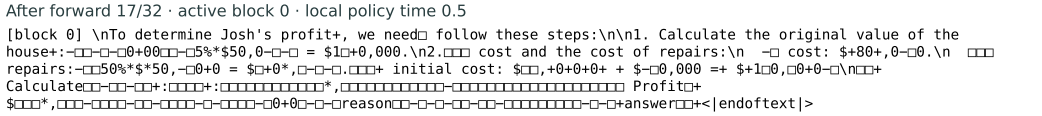
After forward 4/32 · active block 0 · local policy time 0.09375



After forward 9/32 · active block 0 · local policy time 0.25



After forward 17/32 · active block 0 · local policy time 0.5



After forward 32/32 · active block 0 · local policy time — · DETERMINISTIC FINALIZATION

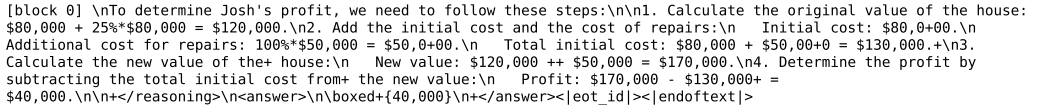


Figure 28: **Unsuccessful full-canvas revision on GSM8K.** Josh buys a house for $80,000, spends $50,000 on repairs, and increases its value by 150%. Despite repeated visibility changes, the response ends with an incorrect calculation and a generated answer of $40,000. The trajectory demonstrates that remasking is an opportunity to revise, not a guarantee of correct reasoning or a correct final answer.

C EXPERIMENTAL AND IMPLEMENTATION DETAILS

This section specifies the architecture, numerical implementation, and training procedure for the mathematical-reasoning checkpoints used in the main comparison. We distinguish the full-canvas policy, $\mathrm{TIPS}_{\mathrm{full}}$, from the two-block policy, $\mathrm{TIPS}_{\mathrm{block}}$, while identifying the supervised initialization shared by the two variants. Evaluation populations, compute accounting, and statistical summaries are specified in Appendix D.

C.1 FROZEN DECODER AND POLICY ARCHITECTURE

We use the frozen LLaDA-8B-Instruct decoder (Nie et al., 2025), identified as `GSAI-ML/LLaDA-8B-Instruct`. Only the visibility controller is optimized: decoder parameters are excluded from the optimizer, and the recorded decoder weights remain unchanged throughout training.

The controller is a two-layer RoPE-DiT (Peebles & Xie, 2023; Su et al., 2024) with hidden width 128, two attention heads per layer, and feed-forward width 512. A linear projection maps each two-dimensional confidence feature to the hidden width. A 128-dimensional sinusoidal time embedding is processed by a two-layer multilayer perceptron with SiLU activation, and a learned two-state embedding distinguishes masked from visible positions. The DiT layers use time- and state-conditioned adaptive layer normalization, bidirectional self-attention over response-position inputs, and GELU feed-forward activations. Rotary position embeddings are applied to attention queries and keys, with base $10{,}000$. A final layer normalization and scalar projection produce one visibility logit per position.

We use zero dropout and no separate conditioning on the absolute decoding budget. The adaptive normalization parameters are initialized to zero, the shared output bias is initialized to zero, and learned state-dependent logit offsets are initialized to -2 for masked positions and $+4$ for visible positions. Each final controller has 628,739 trainable parameters; fixed buffers are excluded from this count. The two response blocks of $\mathrm{TIPS}_{\mathrm{block}}$ share controller weights, so response-block count does not multiply the number of parameters.

Table 30: Configuration of the final mathematical-reasoning policies. The architecture is shared; decoding geometry, local timing, and fitted schedule differ.

Setting	$\mathrm{TIPS}_{\mathrm{full}}$	$\mathrm{TIPS}_{\mathrm{block}}$
Response geometry	1×256	2×128
DiT layers / hidden width	$2/128$	$2/128$
Attention heads / head width	$2/64$	$2/64$
Feed-forward width	512	512
Confidence channels	2	2
Time-embedding width	128	128
Dropout	0	0
Trainable policy parameters	628,739	628,739
Training decoder-call budget	32	32
Schedule power $\alpha = \exp(\gamma)$	1.1565491821	1.2060408032
Schedule parameter γ	0.1454407285	0.1873429312

C.2 CONFIDENCE FEATURES AND VISIBILITY UPDATES

We compute the features in Equation (6) from the decoder distribution before applying the visibility transition. At a masked position, the active token is the greedy proposal; at a visible position, it is the token already stored on the canvas. The strongest alternative is obtained from the two highest-probability vocabulary entries: it is the second entry when the active token is first, and the first entry otherwise. Thus, alternative probabilities are not renormalized over a reduced vocabulary.

For numerical stability, feature calculations are performed in float32 with $\epsilon = 10^{-6}$. The active probability is clipped to $[\epsilon, 1-\epsilon]$ for its log-odds, whereas both probabilities are clipped only from below by ϵ for the log-ratio. The implementation uses the ordering returned by the vocabulary top-two operation without an additional tie-breaking rule.

All position-wise visibility decisions are applied synchronously. A newly revealed position receives its proposal from the pre-transition decoder evaluation; a visible position selected to remain visible retains its existing token, even when the decoder now prefers another token. At deterministic finalization, one fresh decoder evaluation fills the remaining masks greedily and leaves already visible tokens unchanged. This finalization does not invoke the policy or contribute an action likelihood.

Block-local execution. Let B be the number of response blocks and $H = T/B$ the number of decoder calls allocated to each block. Each block receives $H - 1$ learned transitions and one deterministic finalization. The prompt and finalized prefix remain immutable, while future blocks remain masked until they become active. For local learned step $j \in \{0, \dots, H-2\}$, both variants target occupancy $((j+1)/H)^\alpha$ within the active block. The saved policies use different time inputs: $\mathrm{TIPS}_{\mathrm{full}}$ receives j/H, whereas $\mathrm{TIPS}_{\mathrm{block}}$ receives $(j+1)/H$, with the clock reset at each block. We preserve these checkpoint-specific conventions during evaluation.

C.3 Numerical calibration of the visibility logits

For the set $\mathcal{I}$ of positions eligible for scheduling, define

$$f(\delta) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sigma(b_i + \delta) - \bar{\kappa}, \qquad \bar{\kappa} = \mathrm{clip}(\kappa, 10^{-6}, 1 - 10^{-6}). \tag{14}$$

The scalar shift is initialized as $\delta^{(0)} = \mathrm{logit}(\bar{\kappa}) - |\mathcal{I}|^{-1} \sum_{i \in \mathcal{I}} b_i$. We first perform ten float32 Newton iterations,

$$\begin{aligned} \delta^{(r+1)} &= \delta^{(r)} - \mathrm{clip}\left(\frac{f(\delta^{(r)})}{\max\{f'(\delta^{(r)}), 10^{-8}\}}, -10, 10 \right), \\ f'(\delta) &= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sigma(b_i + \delta)\big[1 - \sigma(b_i + \delta)\big]. \end{aligned} \tag{15}$$

For rows whose resulting absolute residual is nonfinite or at least 10^{-6}, we select a safeguarded ten-iteration solution. Its initial bracket is $[\mathrm{logit}(\bar{\kappa}) - \max_{i \in \mathcal{I}} b_i, \mathrm{logit}(\bar{\kappa}) - \min_{i \in \mathcal{I}} b_i]$. Each safeguarded iteration updates the bracket and chooses the lower-residual candidate between a clipped Newton proposal and the bracket midpoint. Both solution paths remain in the automatic-differentiation graph, so gradients pass through the selected calibration computation during training. The detached teacher solve used for supervised targets is a separate procedure.

C.4 Supervised warmup

Reference responses. The training input contains 12,776 problems: 6,280 from GSM8K and 6,496 from MATH. We use each problem's original dataset answer or solution as its reference response. Each reference view ends at the first EOS token, or has EOS appended when none is present, and is then capped at 256 tokens. All training problems are retained, including the 2,113 whose reference view exceeds the canvas length before truncation. Reference inclusion is not conditioned on acceptance by the generated-answer verifier. Padded positions are marked invalid for the direct supervised loss.

Teacher and loss. The teacher uses the reference-token utility $u_{t,i}$ in Equation (10). The masked-position margin is computed by subtracting the strongest non-reference logit from the reference-token logit, which preserves the sign of the margin when the reference is the decoder's top prediction. In the implementation, valid-position utilities are transformed to $(u_{t,i} + 1)/2$ and multiplied by 10 before applying the occupancy-matching offset. Consequently, the sharpness in the raw-utility convention of the main text is $\beta = 5$.

Binary cross-entropy is averaged over reference-valid positions within each state and then equally over the states in the minibatch. Reference tokens and reference margins are used only to construct supervision; the student receives the decoder-derived confidence features, mask states, and time inputs.

Initial and refreshed state banks. Warmup has two successive stages, both using full-canvas decoding at $T_{\text{train}} = 32$. The first stage trains from a fresh initialization on an initial bank of 396,056 states. We then refresh the bank with 99,014 additional states and continue from the first-stage policy using the combined 495,070-state bank. The optimizer, learning-rate schedule, and local update counter restart for the second stage. Each stage visits its bank once, with seed-dependent packet order and within-packet state order. Thus, the two stages together comprise 7,304 optimizer updates.

Table 31: Training stages per seed. State presentations count supervised examples processed, not distinct training problems.

Stage	Updates	State presentations	Initialization
Initial supervised warmup	3,230	396,056	Fresh policy
Refreshed supervised warmup	4,074	495,070	Initial warmup endpoint
Full-canvas RLVR	935	—	Refreshed warmup endpoint
Two-block RLVR	935	—	Same warmup endpoint

C.5 RLVR AND OPTIMIZATION SETTINGS

For each training seed, the final supervised policy initializes separate full-canvas and two-block RLVR runs. Each run uses a global batch of 128 trajectories, formed from 16 prompt groups with eight trajectories per prompt, and a decoding horizon of 32 calls. Rollouts and likelihood replay use microbatches of 16 trajectories. Terminal reward is binary final-answer correctness, and the advantage for a trajectory is its reward minus the mean reward of its eight-trajectory group.

At each learned step, the action log-probability sums Bernoulli log-probabilities over the positions on which the policy acts. The new-to-old joint action-likelihood ratio is clipped to $[0.5, 1.5]$ in the GRPO surrogate, which is averaged over learned steps and trajectories. Finalization contributes neither an action nor a likelihood term. Hence, a training rollout contains 31 learned transitions for the full-canvas policy and 30 for the two-block policy. Both final mathematical-reasoning branches complete 935 nominal updates and 935 optimizer steps. Each branch therefore uses $935 \times 128 \times 32 = 3{,}829{,}760$ example-level decoder evaluations for training rollouts; this count excludes bootstrap training and supervised state collection.

Table 32: Optimization settings.

Setting	Supervised warmup	RLVR
Optimizer	AdamW	AdamW
Peak learning rate	3×10^{-5}	3×10^{-5}
Adam moments	$(0.9, 0.999)$	$(0.9, 0.99)$
Adam numerical constant	10^{-8}	10^{-8}
Weight decay	0	0.1; 0 for bias/norm
Learning-rate warmup	100 updates	100 updates
Subsequent schedule	Cosine decay	Cosine decay
Global gradient-norm clipping	0.2	0.2
Trajectories per prompt	—	8
Global trajectory batch	—	128
Likelihood-ratio clipping	—	$[0.5, 1.5]$

We evaluate the fixed 935-update endpoints for training seeds 123, 124, and 125. The checkpoint indices 1, 2, and 3 correspond to these seeds, respectively, for both geometries. Full-canvas and block policies with the same index share their supervised warmup ancestral weights, but their RLVR weights are trained separately.

C.6 BASELINE CONFIGURATIONS

The mathematical-reasoning comparison contains the configurations listed in Table 33. Multiple configurations of one method differ in response geometry or in the saved training or decoding variant; they are therefore retained as separate comparisons.

Table 33: Baseline configurations. A geometry denotes number of blocks $\times$ tokens per block. Separate entries within a row are distinct saved configurations. The 4×32 UPO configuration uses a 128-token response and is recorded only at $T = 128$.

Method	Configurations
Random / uniform	1×256
High confidence	1×256, fitted schedule; 1×256, linear schedule
Probability margin	1×256
Predictive entropy	1×256
Fast-dLLM	8×32, mean-adjusted as described in Section 4.1
Non-adjusted Fast-dLLM	1×256; 8×32
EB-Sampler	1×256
UNCODE / PC-Sampler	1×256
KLASS	1×256; 4×64
DAPD	1×256
Jazbec et al. (2026)	1×256, expert-steering; 1×256, vanilla
UPO	1×256; 8×32, block-trained; 4×32, 128-token response
SAS	1×256; 8×32
ReMDM	1×256
RCR	1×256
WINO	1×256; 2×128
CoRe	1×256

D Evaluation protocols

We first specify the mathematical-reasoning evaluation of the main TIPS policies, including its validation/test separation and aggregation over trained policies. We then describe the populations, interventions, and uncertainty measures used in the supplementary experiments, whose statistical units differ from those of the main result.

D.1 Datasets, validation split, and prompts

The mathematical-reasoning test population is the union of all 1,319 GSM8K test problems and all 500 MATH-500 problems (Cobbe et al., 2021; Hendrycks et al., 2021; Lightman et al., 2024). The 1,024-problem validation set contains 512 GSM8K and 512 MATH problems drawn from the training sources and excluded from the gradient-training input. The recorded training, validation, and test identifier sets are pairwise disjoint.

Table 34: Problem populations for the main mathematical-reasoning training and evaluation. The validation population is also used, without correctness labels, to fit the schedule.

Population	GSM8K	MATH / MATH-500	Total
Gradient-training input	6,280	6,496	12,776
Validation and schedule fitting	512	512	1,024
Final mathematical-reasoning test	1,319	500	1,819

The same validation prompts support both label-free schedule fitting and design ablations. Their equal task mixture also differs from the $1{,}319{:}500$ test mixture, so their absolute accuracies are not directly comparable to main-test accuracies.

Prompt construction. The main TIPS test prompts are zero-shot. We place the task instruction and question in one user message, separated by two newline characters, apply the tokenizer's chat template with `add_generation_prompt=True`, and append the literal prefix `<reasoning>`. The GSM8K instruction is:

> You are a math expert. You will be given a question to solve. Solve it step by step. Wrap the final answer in a `\boxed{}`.
> Respond in the following format:

```
<reasoning>
Your reasoning here
</reasoning>
<answer>
\boxed{...}
</answer>
```

The saved MATH-500 instruction has the same text, with a literal double-quotation mark and a newline after the closing `</answer>` tag. We retain this exact task-specific prompt rather than normalizing it after evaluation. The full tokenized prompt is preserved. The response canvas is a separate fixed-length sequence of 256 tokens.

D.2 Decoder-call budgets and actual consumption

The main budgets are $T \in \{4, 8, 16, 32, 64, 128, 256\}$. For TIPS, these are exact horizons: each example consumes all T calls, including deterministic finalization, even if an intermediate canvas is already fully visible. Full-canvas decoding uses $T - 1$ learned transitions and one finalization; two-block decoding uses $T-2$ learned transitions and two finalizations. For example, the two-block policy still makes two learned transitions at $T = 4$.

The horizontal coordinate in the main comparison denotes a decoder-call cap, not necessarily the number of calls consumed by an adaptive baseline. The displayed 8×32 Fast-dLLM configuration uses the mean-adjusted confidence threshold to match budget in expectation (refer to Appendix A.2). Other naturally terminating decoding can consume fewer calls than its cap. We therefore distinguish capped accuracy comparisons from measurements plotted against mean actual decoder calls. Controller evaluations and numerical calibration do not add decoder calls, although their runtime is not represented by this metric. Equal decoder-call counts alone do not imply equal wall-clock time or floating-point operation counts across response geometries.

D.3 Output processing and answer verification

For the main TIPS evaluation, we decode the complete 256-token response canvas with `skip_special_tokens=True`, without an additional stop-string truncation. GSM8K reference targets are converted to finite numerical values, using the final `####` target when a full dataset answer is supplied. MATH targets are normalized with the mathematical-answer parser, which extracts an answer from boxed, framed, or answer-tagged reference text when applicable. Generated responses are evaluated by the corresponding task-specific extraction and correctness functions. Each verification has a five-second timeout. Missing or nonfinite predictions, answer mismatches, timeouts, and verifier exceptions count as incorrect; they are not removed from the evaluation denominator.

D.4 Randomness, aggregation, and uncertainty

For each TIPS geometry and budget, the main evaluation uses three trained policies, with seeds 123, 124, and 125, and three inference streams per policy, with identifiers 42, 43, and 44. Token proposals are greedy, whereas visibility actions are independent Bernoulli draws conditional on the policy probabilities.

Let $c_{sri}(T) \in \{0, 1\}$ denote correctness on problem i for training seed s and inference stream r, and let $N = 1{,}819$. We first compute each trained policy's accuracy, averaging its three inference draws, and then average the three policy accuracies:

$$A_s(T) = \frac{1}{3N} \sum_{r=1}^{3} \sum_{i=1}^{N} c_{sri}(T), \qquad \bar{A}(T) = \frac{1}{3} \sum_{s=1}^{3} A_s(T). \tag{16}$$

This is a problem-level micro-average over GSM8K and MATH-500, not an equally weighted average of their dataset-level accuracies. For the main table and shaded bands, we report the standard error across the three trained-policy means:

$$\mathrm{SE}(T) = \frac{1}{\sqrt{3}} \left[\frac{1}{2} \sum_{s=1}^{3} \big(A_s(T) - \bar{A}(T) \big)^2 \right]^{1/2}. \tag{17}$$