DM-OPD: ALIGNING DIFFUSION LANGUAGE MODELS WITH AUTOREGRESSIVE TEACHERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Converting strong autoregressive (AR) models into diffusion language models promises parallel generation while retaining AR capabilities. Existing conversion methods fit intermediate predictions, but improving this fit need not improve the distribution of generated text. We analytically demonstrate this mismatch even on the student's own generation trajectories. We introduce Distribution Matching OPD (DM-OPD), which defines distillation on the text distribution induced by the student's actual decoder and derives an unbiased on-policy gradient estimator for each completed-block KL at student-generated prefixes. We make this objective trainable by estimating student text likelihoods, decomposing completed-text feedback into dependency-based action credit, and defining trainable probabilities for refinement decisions. Using Qwen3 initialization and teacher checkpoints with OPDLM's prompt corpus, we distill without answer-correctness rewards. Our 1.7B student outperforms the diffusion students SDAR and OPDLM on all eight benchmarks at 4K and 8K response limits, as does our 4B student except on HumanEval at 4K. With the same decoder and the guard disabled, our objective improves monitoring accuracy by 5.4 points over OPDLM's objective, whose loss keeps falling. Students trained for fewer refinement iterations outperform released diffusion models decoded with the same iteration budget on most benchmarks. These results establish completed-text distribution matching as an effective foundation for AR-to-diffusion distillation.

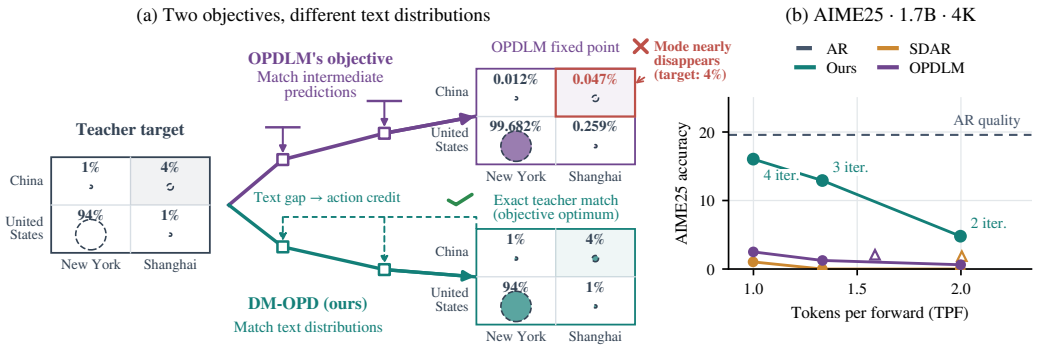

Figure 1: **Completed-text matching targets the teacher distribution and improves diffusion quality.** (a) Analytical example with a shared threshold decoder and two-step budget: OPDLM's objective suppresses Shanghai–China from 4% to $.047\%$; the completed-text objective preserves the full teacher distribution at its stable optimum (Appendix A.5). Outlines mark teacher probabilities, fills student probabilities, and square markers refinement steps. (b) AIME25 with 1.7B models, a 4K response limit, and thinking enabled. Lines connect static schedules with four, three, and two iterations, and labels give our students' iteration counts; triangles mark threshold decoders ($\tau = 0.90$).

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.

1 INTRODUCTION

Autoregressive (AR) language models acquire strong capabilities through large-scale pretraining and post-training, but their sequential generation limits decoding efficiency. Diffusion language models offer an opportunity to combine these capabilities with greater parallelism by predicting multiple tokens over successive refinement steps (Arriola et al., 2025). Converting a strong AR model into a diffusion model can reuse its existing knowledge while changing how it generates text. The central challenge is to preserve the AR teacher's quality while achieving a better quality–speed trade-off.

Existing conversion methods still leave a substantial quality gap to their AR references in our evaluations (Figure 1). Teacher-forced adaptation, such as SDAR (Cheng et al., 2025), reconstructs masked reference text, exposing the student to states different from those encountered during generation—the familiar exposure bias. Existing on-policy distillation (OPD), such as OPDLM (Su et al., 2026), addresses this state mismatch but still trains intermediate token predictions rather than directly matching the distribution of completed text. Training on the student's own trajectories therefore leaves an objective mismatch: improving these predictions can drive the completed-text distribution away from the teacher's (Figure 2).

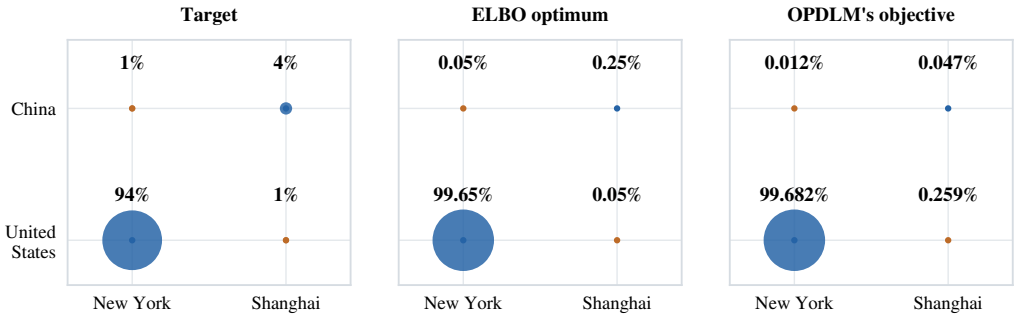

Figure 2: **Matching intermediate predictions can favor the wrong generated-text distribution.** Circle areas show completed city–country probabilities in an analytical example. The same threshold decoder and two-step budget can produce the teacher's target, yet Shanghai–China falls from 4% to $.25\%$ at the masked-ELBO minimum and $.047\%$ at the locally stable output of the OPDLM objective's update (Section 3).

We therefore start from the KL divergence between the student's completed-text distribution and the AR teacher's. Its gradient weights sampled generation decisions by the completed text's teacher-to-student log-likelihood ratio, exposing three challenges (Figure 3). First, many refinement paths produce the same text: its probability sums over these paths, so the likelihood of one sampled trajectory cannot provide it. Second, sharing one completed-text score across every action mixes useful feedback with fluctuations unrelated to that action. As more tokens contribute to the score, this noise can obscure individual learning signals. Fine-grained action credit must isolate useful feedback for each decision while preserving the expected update. Third, the decoder's decisions must be amenable to optimization. Threshold decoders provide token-sampling probabilities, but hard confidence and top-k selection can change commitment decisions abruptly. We need explicit action probabilities through which completed-text feedback can train refinement behavior.

We introduce Distribution Matching OPD (DM-OPD), which defines distillation directly on generated-text distributions. At student-generated prefixes, we match completed blocks to the AR teacher. The teacher's distribution is the optimum of each conditional KL; matching these conditionals recovers its response distribution. We derive their exact on-policy gradients to guide training. We make this framework practical through three components. An auxiliary AR model fitted to the student's outputs estimates its otherwise intractable text probabilities. A dependency-based credit decomposition translates completed-text feedback into fine-grained action updates while preserving the expected gradient. Our stochastic keep-or-replace decoder provides the action probabilities needed to train these decisions, using the same generation rule during training and inference (Section 4). The teacher defines the desired text distribution; the student learns the refinement process that produces it.

$$\min_{\theta} J(\theta), \qquad J(\theta) = \mathrm{KL}\big(p_\theta^{\mathrm{text}} \,\|\, q\big), \qquad x = F(\tau).$$

$$-\nabla_\theta J = \mathbb{E}_{\tau\sim P_\theta}\Big[\underbrace{\log \frac{q(x)}{p_\theta^{\mathrm{text}}(x)}}_{\text{1 Student text density?}} \sum_{t,i} \nabla_\theta \log \pi_\theta(a_{ti} \mid s_t)\Big].$$

$$\widehat{g}(\tau) = \sum_{t,i} \underbrace{A_{ti}(\tau)}_{\text{2 Fine-grained action credit?}} \underbrace{\nabla_\theta \log \pi_\theta(a_{ti} \mid s_t)}_{\text{3 Trainable action probabilities?}}$$

Figure 3: **From completed-text KL to action updates.** At a fixed context, student trajectory τ produces text x. q and p_θ^{text} are teacher and student text distributions; π_θ gives action a_{ti}'s probability given history s_t. Setting every A_{ti} to the completed-text log-ratio recovers the exact gradient above. Finer credit must preserve the expected update, $\mathbb{E}[\widehat{g}] = -\nabla_\theta J$. Red highlights the quantities to estimate or design.

Across eight benchmarks, our 1.7B and 4B students deliver broad gains over existing diffusion models under both thinking and non-thinking evaluation (Table 1; Appendix F) and advance the quality–efficiency frontier (Figure 1). They substantially narrow general OPDLM's gap to the AR teacher; our 4B student approaches it on AIME25 at matched response limits. Our main conversions share general OPDLM's Qwen3 initialization, teacher checkpoints, and prompt corpus, with thinking enabled during distillation and no answer-correctness rewards. The general-purpose 4B student outperforms the mathematics specialist OPDLM-MATH on all five mathematics and science benchmarks; both model sizes also benefit more than general OPDLM from longer generation on AIME24, AIME25, MATH500, and GPQA. These results establish completed-text distribution matching as an effective foundation for AR-to-diffusion post-training.

2 Background and Related Work

AR adaptation and OPD provide routes to parallel language generation, but retaining the teacher's quality remains the central difficulty.

Converting a pretrained AR model into a diffusion model aims to retain its capabilities. SDAR (Cheng et al., 2025) and Efficient-DLM (Fu et al., 2025) train AR models on masked reference text to obtain block diffusion models (Arriola et al., 2025), which generate blocks sequentially and the positions within each block in parallel; Efficient-DLM adjusts masking by position to resemble decoding. OPDLM (Su et al., 2026) instead supervises student-generated trajectories, querying an AR teacher on completed responses to train intermediate predictions. Our evaluations of SDAR and OPDLM still show a substantial gap to their AR references (Table 1).

Distillation from an existing diffusion teacher has the potential to improve student quality or reduce generation steps. TOPD matches teacher predictions on retained student decisions (Ren et al., 2026); dOPSD and COPSD use later refinements as teachers (Dat et al., 2026; Zhu et al., 2026). T3D trains on full-step teacher trajectories with a reverse-KL discriminative objective (Zhang et al., 2026). DiDi-Instruct estimates density ratios on forward-corrupted teacher and student responses (Zheng et al., 2025). Teacher trajectories and corrupted states differ from the states the student encounters during generation, and matching conditional predictions at these states is not matching the distribution of completed text: improving these predictions can drive the student's completed-text distribution away from the teacher's. We argue that this objective mismatch is a central obstacle to retaining teacher quality.

On-policy distillation is widely used for AR language models: GKD and MiniLLM train students on their own generations, with MiniLLM minimizing reverse KL (Agarwal et al., 2024; Gu et al., 2023). Distribution-level distillation has also been developed for continuous diffusion models: the DMD series matches output distributions (Yin et al., 2023; 2024) and DiffusionOPD matches denoising transitions (Li et al., 2026). For image generation, Di[M]O distills masked diffusion models into one-step generators by matching masked token-conditionals with an auxiliary denoiser (Zhu et al., 2025). Diffusion language models differ from both AR and continuous diffusion models. Their samples are discrete tokens, so an update cannot follow infinitesimal changes of a sample as in continuous diffusion; and many refinement paths produce the same text, so its probability is intractable, whereas

an AR model's text probability is the product of its token probabilities. Section 3 examines the objective mismatch; Section 4 estimates the completed-text probability with an auxiliary model and derives an estimator that connects completed-text feedback to the sampled refinement actions.

3 Matching Intermediate Predictions Can Miss an Attainable Distribution

Matching intermediate predictions can move the student's output distribution away from an attainable target, even on student-generated trajectories. We establish this for the masked-ELBO minimum and the fixed point of the existing OPDLM update under the same threshold decoder.

Teacher forcing has the familiar exposure-bias problem: reference-derived states differ from those encountered during generation (Appendix D illustrates dataset-dependent reveal orders). Our question is separate: does the loss target the decoder's output distribution, even when training uses student trajectories? We study the widely used setting of block-parallel generation with confidence-based commitment (Cheng et al., 2025; Su et al., 2026).

Consider two categorical fields: a city, New York or Shanghai, and a country, United States or China. Write their choices as $C, N \in \{0, 1\}$ and set

$$q(00) = .94, \qquad q(11) = .04, \qquad q(01) = q(10) = .01. \tag{1}$$

Starting with both fields masked, the student draws a candidate for each masked field in parallel and commits every candidate whose predicted probability is at least .9; if none qualifies, it commits the most confident candidate, or both when tied. Committed values remain fixed; at most two steps are allowed. The student has a separate prediction table for each masked field and visible context. Appendix A.1 specifies the decoder and fallback variants.

The standard population masked-ELBO loss (Sahoo et al., 2024) is minimized by exact marginals at fully masked fields and exact conditionals when the other field is visible. However, threshold selection depends on sampled candidates, so these predictions need not compose into the target output law.

Proposition 1 (A masked-ELBO minimum that misses an attainable target)**.** *Strictly positive prediction tables can generate Eq.* (1) *exactly under this decoder and budget. Yet the unique population masked-ELBO minimum produces* $(.9965, .0005, .0005, .0025)$ *in the order* $00, 01, 10, 11$*, with total-variation distance* $.0565$ *from the target.*

Shanghai–China falls from 4% to $.25\%$ at the loss minimum (Figure 2). The masked ELBO fits the target's marginals and conditionals, but confidence-based commitment combines them into a different completed-text distribution. Appendix A.1 also shows that matching these same conditionals on student-visited states does not repair the mismatch.

The second case applies OPDLM's objective to the same finite-state student. Its distillation update (Su et al., 2026, Section 4.2, Algorithm 1) samples a student trajectory, selects a nonterminal state uniformly, and supervises masked fields with the AR teacher's conditionals on prefixes of the completed response. Starting from tables that generate q exactly, gradient steps on its expected update converge to a stable output with only $.0471\%$ probability on Shanghai–China. OPDLM's loss, the KL between teacher and student token predictions on student-visited states, falls from $.3581$ to $.00474$, while the KL between the completed-text distributions rises from zero to $.05239$. Appendix A.2 derives the fixed point and its local stability.

Both objectives can therefore lead away from a target that the decoder can represent. Appendix A.3 also demonstrates on-policy objective drift with exactly two decoding steps. Lowering an intermediate prediction loss can move the generated-text distribution away from the target, even when training uses the student's own trajectories. We next define distillation on the completed-text distribution and derive the on-policy update that optimizes it.

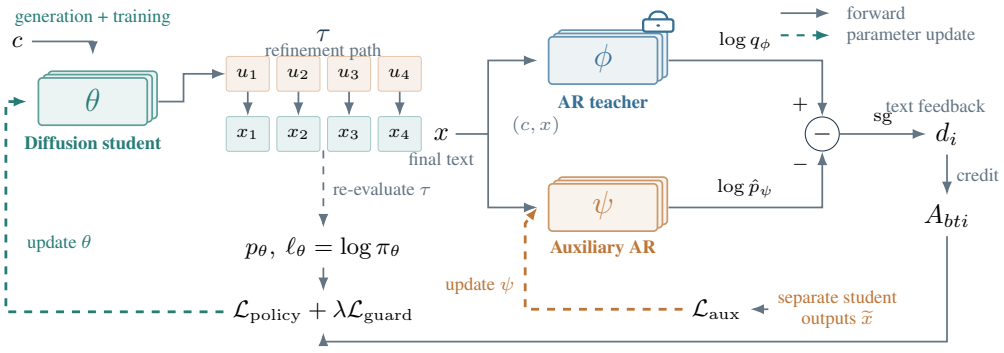

Figure 4: **Completed-text probabilities determine refinement credit.** The inference decoder generates text x. A frozen AR teacher and an auxiliary fitted to separate student outputs provide target and estimated student likelihoods. Their log difference credits sampled actions. Dashed colored arrows show parameter updates; gradients do not pass through sampled tokens.

4 Distribution Matching OPD

DM-OPD trains refinement decisions through the distribution of text they ultimately produce. We derive the on-policy gradient of this objective, then construct its density estimate, dependency-based action credit, and trainable decoder.

We extend conditional AR distillation from tokens to completed blocks, whose feedback trains the refinement actions that generated them. Matching these conditionals recovers the teacher's response distribution (Appendix A.4). For prompt c and student-generated prefix $h_b = (c, x_{<b})$, let τ_b be the next block's refinement trajectory and $x_b = F(\tau_b)$ its completed text. Holding h_b fixed, we match the decoder's output distribution to the frozen AR teacher q_ϕ:

$$
\begin{aligned}
J_b(\theta) &= \mathrm{KL}\big(p_\theta^{\text{text}}(\cdot \mid h_b) \,\|\, q_\phi(\cdot \mid h_b)\big)\,, \\
A_b^\star(\tau_b) &= \log \frac{q_\phi(x_b \mid h_b)}{p_\theta^{\text{text}}(x_b \mid h_b)}.
\end{aligned}
\tag{2}
$$

The teacher's attainable block distribution is a stable optimum of this objective, and it has a descent direction at the masked-ELBO minimum and at the fixed point of OPDLM's update in Section 3 (Appendix A.5). The log-ratio is positive for text the student underproduces relative to the teacher and negative for text it overproduces. Changing the decoder or iteration budget changes p_θ^{text}. We use the same rule and budget in training and inference; Appendix B.1 explains their role in the output law.

4.1 From the text objective to action updates

Differentiating the objective weights each generation path by the feedback on its completed text. If s_{bt} is the history before step t and $\mathcal{A}_b$ indexes its sampled token actions, draws within a step are independent given s_{bt}. The path probability and resulting update are

$$
P_\theta(\tau_b \mid h_b) = \prod_{(t,i)\in\mathcal{A}_b} \pi_\theta(a_{bti} \mid s_{bt}), \qquad \ell_{bti} = \log \pi_\theta(a_{bti} \mid s_{bt}), \tag{3}
$$

$$
g_b \equiv -\nabla_\theta J_b = \nabla_\theta \sum_{\tau_b} P_\theta(\tau_b \mid h_b)\ \mathrm{sg}[A_b^\star(\tau_b)]. \tag{4}
$$

Here sg holds feedback fixed. The identity uses exact text density and normalized differentiable action laws at the fixed prefix (Appendix A.4). The score-function identity yields the loss framework

$$
\mathcal{L}_{\text{path}}(\theta; \bar{\theta}) = -\mathbb{E}_{\tau_b \sim P_{\bar{\theta}}}\left[\mathrm{sg}[A_b(\tau_b)] \sum_{(t,i)\in\mathcal{A}_b} \ell_{bti}\right]. \tag{5}
$$

Here $\bar{\theta}$ denotes the sampling parameters. At $\theta = \bar{\theta}$ and $A_b = A_b^\star$, its gradient is $\nabla_\theta J_b$, an expectation under the current trajectory law, so the estimator uses on-policy samples. We now estimate the completed-text feedback, decompose it into action credit, and specify a decoder that exposes the required probabilities (Figure 4).

4.2 Estimating completed-text feedback

Teacher likelihood is tractable, but the student's completed-text probability sums over compatible paths: $p_\theta^{\text{text}}(x_b \mid h_b) = \sum_{\tau_b : F(\tau_b) = x_b} P_\theta(\tau_b \mid h_b)$. The log-ratio in Eq. (2) compares teacher and student text probabilities to identify overproduced responses. We therefore fit an auxiliary AR model $\hat{p}_\psi$ to completed student outputs (Figure 5a) and use its likelihood to estimate the missing density:

$$\mathcal{L}_{\text{aux}}(\psi) = -\mathbb{E}_{c, x \sim p_\theta^{\text{text}}} \log \hat{p}_\psi(x \mid c), \tag{6}$$

$$d_i = \text{sg}\left[\log \frac{q_\phi(x_i \mid c, x_{<i})}{\hat{p}_\psi(x_i \mid c, x_{<i})}\right], \qquad \widehat{A}_b = \sum_{i \in V_b} d_i. \tag{7}$$

Here V_b contains valid block positions. Maximum likelihood on student outputs minimizes $\text{KL}(p_\theta^{\text{text}} \| \hat{p}_\psi)$ at a fixed student: it targets the completed-text law, including all paths that produce it. Appendix A.7 relates density-estimation error to the accuracy and direction of the resulting update.

4.3 Decomposing feedback into action credit

Generation dependencies determine which completed-text feedback each action must retain. A feedback term independent of an action, conditional on its preceding state, contributes zero to its expected score update (Schulman et al., 2015). Intermediate actions can influence every final position through shared refinement states, so they retain the full block feedback. At the final step, draws are conditionally independent and d_j depends only on the completed prefix through j. Final action i therefore retains terms from its own position onward: earlier terms contribute zero, while later likelihood terms can depend on its token (Figure 5b). For fixed block width B and refinement budget T_b, this gives

$$A_{bti} = \frac{1}{B} \begin{cases} \displaystyle\sum_{j \in V_b} d_j, & t < T_b, \\ \displaystyle\sum_{j \in V_b :\, j \geq i} d_j, & t = T_b. \end{cases} \tag{8}$$

The common $1/B$ sets the feedback scale for all actions. With exact student density and fresh on-policy samples, these credits give an unbiased estimate of the conditional-block gradient $-\nabla_\theta J_b / B$ (Proposition 3). The implementation uses estimated density and the feedback processing specified in Appendix B.2. Both branches use only the completed text's token scores; Appendix A.6 provides their proofs.

4.4 Defining the sampled actions

Equation (3) requires per-token behavior probabilities and their gradients. Our keep-or-replace decoder factorizes each parallel transition into explicit token probabilities, piecewise differentiable in the model predictions. The first pass samples tokens for a fully masked block; later passes retain or replace tokens within a fixed iteration budget.

For previous token u, let $p(v)$ be the current vocabulary prediction and $\rho(u) > 0$ its vocabulary probability from the preceding iteration. Our keep-or-replace law is

$$\alpha = \min\left\{1, \frac{p(u)}{\rho(u)}\right\}, \qquad \pi_\theta(v \mid s) = \begin{cases} \alpha, & v = u, \\ (1 - \alpha)\dfrac{p(v)}{1 - p(u)}, & v \neq u. \end{cases} \tag{9}$$

The ratio retains a token when its current probability has not fallen below its preceding probability; otherwise, it can be replaced from p excluding u. Probabilistic retention and replacement provide gradients through both current and preceding predictions. Draws are independent given the shared history and jointly form the next block state. Appendix B.1 specifies initialization and sampling.

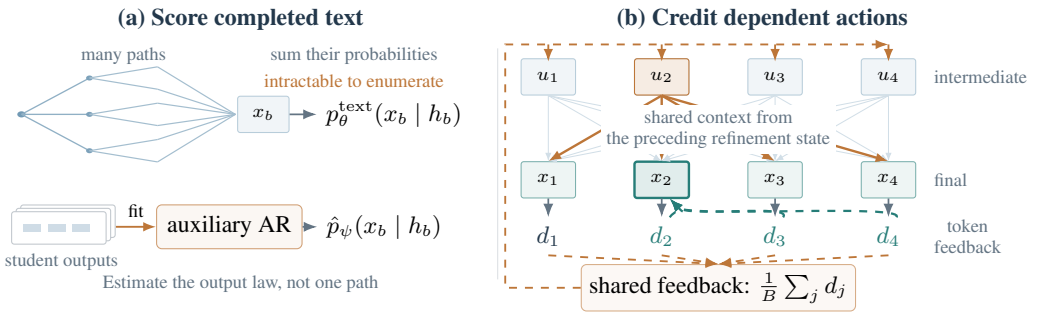

Figure 5: **Text density determines the score; dependencies determine its reach.** (a) The auxiliary estimates the combined probability of paths producing the same text. (b) Intermediate actions receive feedback from all final positions. Final action i receives feedback from positions i onward; the highlighted example routes $(d_2 + d_3 + d_4)/B$ to x_2. Both use the same block-width normalization. Dashed arrows show feedback, solid arrows generation dependencies.

4.5 Assembling the Training Loss

We combine the estimated density ratio, dependency-based credit, and decoder probabilities within the framework of Eq. (5). For fresh on-policy trajectories with matching generation and training probabilities, the core action loss is

$$\mathcal{L}_{\text{policy}} = -\frac{1}{Z} \sum_{(b,t,i)\in\mathcal{A}} \text{sg}[A_{bti}]\, \ell_{bti}, \qquad \mathcal{L}_{\text{DM-OPD}} = \mathcal{L}_{\text{policy}} + \lambda \mathcal{L}_{\text{guard}}, \tag{10}$$

where Z is the number of valid sampled actions. The guard encourages exploration and responsive refinement during training. It suppresses provisional tokens that disagree with the block's final sample and increasingly favors that sample in later steps (Figure 8). We use $\lambda = 0.005$; Appendix B.2 gives the formula and schedule, and Section 5.2 reports the ablation.

Appendix B gives the full recipe; Appendix B.3 derives the treatment of differences between training and inference probabilities.

5 Experiments

Setup. We distill with thinking enabled and no answer-correctness rewards, from general OPDLM's Qwen3 initialization, teachers, and prompt corpus, using Eq. (8) and the guard ($\lambda = 0.005$); Appendix B.5 gives the recipe. We write BbIi for b-token blocks generated with i parallel refinement iterations before commitment. We train B4I4 students at 1.7B and 4B, and B4I3/B4I2 students at 1.7B, each from the AR checkpoint with a 4,096-token training response limit. We train each run until its accuracy on a 2,977-response monitoring subset of the five mathematics and science benchmarks shows signs of saturation within our compute budget and evaluate the checkpoint given by its predeclared endpoint rule: an update count, a wall-clock cutoff, or a plateau in monitoring accuracy (Appendix B.5).

Qwen3-1.7B/4B are the AR references. Diffusion baselines are SDAR-1.7B/4B-Chat (Cheng et al., 2025), OPDLM-1.7B/4B, and OPDLM-MATH-4B-Thinking (Su et al., 2026), all with four-token blocks. Static baselines use a fixed number of iterations; threshold SDAR and OPDLM use $\tau = 0.90$. General OPDLM was trained without thinking; OPDLM-MATH-Thinking with thinking. **Protocol.** All models are evaluated with thinking enabled, temperature 1, and a common grader. No evaluation uses repetition-based early stopping: every response ends at EOS or the response limit. Efficiency is measured in tokens per forward (TPF), the number of retained response tokens per decoder iteration; Appendix H reports wall-clock time. Appendix E gives the details.

5.1 Generation Quality

Table 1 reports eight benchmarks at two response limits. The first five test mathematics and science; MMLU, IFEval, and HumanEval test knowledge, instruction following, and code. At 1.7B, DM-OPD

Table 1: **Absolute quality across eight benchmarks.** All models use thinking; Qwen3 is the AR reference. Teal marks ours; bold marks the best diffusion score per size and limit. Diffusion models use four-token blocks and four iterations; ours are trained with a 4,096-token response limit. Scores are percentages: GPQA is Diamond, MMLU uses 2,000 questions, IFEval reports strict prompt accuracy, and HumanEval base-test pass@1. No row uses repetition-based early stopping. —: not evaluated.

Model	#params	AIME24	AIME25	GSM8K	MATH500	GPQA	MMLU	IFEval	HumanEval
4K generation limit (4,096 tokens)									
Qwen3 (AR reference)	1.7B	16.67	19.58	89.27	76.75	29.80	69.30	70.66	76.07
Qwen3 (AR reference)	4B	24.58	20.00	94.48	79.45	43.56	79.60	83.46	89.02
SDAR-Chat	1.7B	1.67	1.04	76.04	53.00	25.63	49.05	37.94	48.17
OPDLM (think@eval)	1.7B	5.42	2.50	61.37	48.00	26.39	46.80	29.39	26.98
DM-OPD (Ours)	1.7B	**10.83**	**16.04**	**87.83**	**73.35**	**28.03**	**68.15**	**62.85**	**64.79**
SDAR-Chat	4B	3.33	4.79	88.48	62.10	28.03	66.95	52.22	**68.75**
OPDLM (think@eval)	4B	12.08	12.71	81.94	65.90	30.43	60.00	40.16	51.37
OPDLM-MATH (thinking)	4B	14.17	14.17	92.85	77.00	37.88	—	—	—
DM-OPD (Ours)	4B	**23.12**	**19.79**	**92.91**	**79.25**	**40.28**	**79.25**	**70.01**	66.62
8K generation limit (8,192 tokens)									
Qwen3 (AR reference)	1.7B	31.25	27.08	89.84	85.65	36.62	71.50	70.33	82.62
Qwen3 (AR reference)	4B	46.67	33.96	94.64	90.85	51.89	80.05	83.36	92.99
SDAR-Chat	1.7B	1.25	1.04	75.95	52.25	24.87	50.50	37.57	52.13
OPDLM (think@eval)	1.7B	4.58	2.92	62.23	49.75	25.13	47.50	29.07	24.24
DM-OPD (Ours)	1.7B	**23.12**	**21.46**	**88.21**	**83.15**	**34.09**	**69.35**	**64.79**	**69.66**
SDAR-Chat	4B	3.96	2.71	87.81	61.05	30.56	65.25	53.28	67.99
OPDLM (think@eval)	4B	13.54	10.00	81.88	67.90	29.04	60.35	40.02	49.70
OPDLM-MATH (thinking)	4B	30.63	25.00	93.82	85.90	42.17	—	—	—
DM-OPD (Ours)	4B	**42.92**	**33.12**	**94.11**	**90.75**	**49.75**	**79.25**	**72.64**	**69.82**

has the highest diffusion score on every benchmark at both limits. At 8K, AIME24/AIME25 reach 23.12/21.46 versus at most 4.58/2.92 for SDAR and OPDLM. At 4B, it leads on every benchmark at 8K and on all but HumanEval at 4K (66.62 versus 68.75 for SDAR). It also exceeds the mathematics specialist OPDLM-MATH on all five mathematics and science benchmarks at 8K, by 12.29/8.12 points on AIME24/AIME25, although that model trains on mathematics with 8,000-token rollouts; at 4K the two are nearly tied on GSM8K (92.91 versus 92.85). Longer generation helps our students more than the baselines: from 4K to 8K, our 1.7B student gains 12.29/5.42/9.80/6.06 points on AIME24/AIME25/MATH500/GPQA, whereas neither 1.7B diffusion baseline gains more than 1.75 points on these four benchmarks.

The students still trail their AR teachers. At 4B/8K the gap is below one point on AIME25 (33.12 versus 33.96), GSM8K, MATH500, and MMLU, 2–4 points on AIME24 and GPQA, and large on IFEval (72.64 versus 83.36) and HumanEval (69.82 versus 92.99). At 1.7B/8K the largest gaps are 8.13 points on AIME24 and 12.96 on HumanEval. Appendix F adds non-thinking training and evaluation.

5.2 Effect of the completed-text objective

With the decoder, thinking-enabled prompts, initialization, and seeds held fixed and the guard off in both runs, completed-text feedback exceeds OPDLM's objective by 5.44 points [95% CI: 3.84, 7.08] after 180 updates on the monitoring panel (Table 9). Decoder and data differences alone therefore do not explain this controlled gain. Removing the guard deepens an early accuracy drop but produces no detectable endpoint difference. Teacher-only feedback, which omits the estimated student density, performs worse at a matched update count. These comparisons support the completed-text objective; Appendix G gives the settings, curves, and paired intervals.

5.3 Quality–efficiency frontier

Table 2 pairs each score with its TPF. Fewer iterations raise TPF but degrade the released models sharply: with two iterations (TPF approximately 2), static SDAR and OPDLM reach 37.00/37.47

Table 2: **Quality and decoding efficiency at 1.7B, 4K limit.** Thinking is enabled; teal marks ours, each trained with the decoder it is evaluated with. TPF counts retained tokens per decoder iteration. BbIi denotes b-token blocks and i refinements; τ is the confidence threshold. Static B4I3/B4I2 baselines run the released weights with the same iteration counts as our students.

		AIME24		AIME25		GSM8K		MATH500		GPQA	
Model (1.7B)	Decoder	Score	TPF	Score	TPF	Score	TPF	Score	TPF	Score	TPF
Qwen3 (AR reference)	AR	16.67	1.00	19.58	1.00	89.27	1.00	76.75	1.00	29.80	1.00
SDAR-Chat	B4I4	1.67	1.00	1.04	1.00	76.04	0.99	53.00	1.00	25.63	0.98
SDAR-Chat	B4I3	0.62	1.33	0.00	1.33	60.92	1.33	33.05	1.33	20.96	1.31
SDAR-Chat	B4I2	0.00	2.00	0.00	2.00	37.00	1.99	12.70	2.00	19.70	1.94
SDAR-Chat	B4, τ=0.90	2.08	2.13	1.88	2.00	76.35	1.73	51.15	2.08	26.01	1.54
OPDLM (think@eval)	B4I4	5.42	1.00	2.50	1.00	61.37	1.00	48.00	1.00	26.39	1.00
OPDLM (think@eval)	B4I3	3.33	1.33	1.25	1.33	51.35	1.33	42.40	1.33	23.23	1.33
OPDLM (think@eval)	B4I2	1.25	2.00	0.62	2.00	37.47	1.99	31.40	2.00	23.23	2.00
OPDLM (think@eval)	B4, τ=0.90	3.96	1.66	2.08	1.59	60.58	1.46	49.30	1.70	27.65	1.68
DM-OPD (Ours)	B4I4	10.83	1.00	16.04	1.00	87.83	1.00	73.35	1.00	28.03	1.00
DM-OPD (Ours)	B4I3	10.62	1.33	12.92	1.33	81.50	1.33	63.35	1.33	26.39	1.33
DM-OPD (Ours)	B4I2	3.33	2.00	4.79	2.00	74.41	1.99	41.05	2.00	21.34	2.00

on GSM8K and 12.70/31.40 on MATH500. Our two-iteration student reaches 74.41 and 41.05 at comparable TPF and exceeds both on AIME24, AIME25, GSM8K, and MATH500; static OPDLM remains higher on GPQA (23.23 versus 21.34), and the AIME differences are small in absolute terms (at most 4.8 points). Fewer iterations also cost our students quality: from four to two iterations, GSM8K falls from 87.83 to 74.41 and MATH500 from 73.35 to 41.05. At TPF 1.33, our three-iteration student exceeds both static three-iteration baselines on all five benchmarks, for example 81.50 versus at most 60.92 on GSM8K and 63.35 versus at most 42.40 on MATH500. Threshold decoders choose the number of iterations adaptively and reach TPF of approximately 1.5–2.1. At these TPFs, at least one threshold decoder scores higher than our two-iteration student on AIME24, GSM8K, MATH500, and GPQA; ours is higher than both on AIME25 only. At a lower TPF of 1.33, our three-iteration student scores higher than both threshold decoders on every benchmark except GPQA against threshold OPDLM (26.39 versus 27.65). Combining adaptive commitment with our objective is left to future work. Appendix H reports wall-clock measurements.

6 Conclusion

Distribution Matching OPD derives diffusion post-training from a completed-text distributional objective. Its density estimate, dependency-based action credit, and decoder probabilities train multi-step refinement. Across eight benchmarks, our students improve diffusion quality and advance the quality–efficiency frontier. Appendix I discusses feedback accuracy and post-training cost.

AI USE STATEMENT

Generative AI tools assisted with comparing baseline training and evaluation settings; formulating mathematical claims and proofs; and designing small, controlled experiments to examine the completed-text objective and verify its gradient estimators. They also assisted with building, debugging, and optimizing the training and evaluation framework, including performance optimizations. Additional uses included literature searches, experimental analysis and interpretation, manuscript drafting and editing, and figure and table preparation. AI-assisted derivations, code, and numerical claims were checked against mathematical arguments, source code, experiment records, and independent reviews. The authors are responsible for the final text, claims, experimental results, and accompanying artifacts.

REPRODUCIBILITY STATEMENT

Appendix A.4 states the assumptions and derivations of the completed-text objective and estimator. Appendix B specifies the decoder, training loss, optimization settings, and checkpoint lineage. Appendices E and F document checkpoint revisions, evaluation settings, grading protocols, and the complete benchmark results, including the additional baseline checks.

ETHICS STATEMENT

This work studies post-training and inference of existing language models using public prompt datasets and benchmarks. It does not introduce a human-subject study or collect personal data. The resulting models retain the ethical considerations of their source models and data, including the possibility of biased or harmful generated content.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.13649.

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 *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2503.09573.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation, 2025. URL https://arxiv.org/abs/2510.06303.

Phuong Tuan Dat, Qi Li, and Xinchao Wang. dOPSD: On-policy self-distillation for diffusion language models, 2026. URL https://arxiv.org/abs/2607.04428.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1407–1416, 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-DLM: From autoregressive to diffusion language models, and beyond in speed, 2025. URL https://arxiv.org/abs/2512.14067.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-policy distillation of large language models, 2023. URL https://arxiv.org/abs/2306.08543.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. DiffusionOPD: A unified perspective of on-policy distillation in diffusion models, 2026. URL https://arxiv.org/abs/2605.15055.

Martin Marek and Max Ryabinin. Score centering stabilizes off-policy reinforcement learning, 2026. URL https://arxiv.org/abs/2609.20807.

Haolin Ren, Ziyang Huang, Chenhao Yuan, Jun Zhao, and Kang Liu. Trace-based on-policy distillation for masked diffusion language models, 2026. URL https://arxiv.org/abs/2607.16872.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.07524.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs, 2015. URL https://arxiv.org/abs/1506.05254.

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

Xingyu Su, Jacob Helwig, Shubham Parashar, Atharv Chagi, Lakshmi Jotsna, Degui Zhi, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Data-efficient autoregressive-to-diffusion language models via on-policy distillation, 2026. URL https://arxiv.org/abs/2606.06712.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation, 2023. URL https://arxiv.org/abs/2311.18828.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis, 2024. URL https://arxiv.org/abs/2405.14867.

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Chengzhi Mao, Hao Wang, Vladimir Pavlovic, and Dimitris N. Metaxas. Few-step diffusion language models via trajectory self-distillation, 2026. URL https://arxiv.org/abs/2602.12262.

Haoyang Zheng, Xinyang Liu, Cindy Xiangrui Kong, Nan Jiang, Zheyuan Hu, Weijian Luo, Wei Deng, and Guang Lin. Ultra-fast language generation via discrete diffusion divergence instruct, 2025. URL https://arxiv.org/abs/2509.25035.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

Huaisheng Zhu, MingYu Liu, Junze Liu, Zhen Ge, Tian Wang, Jiri Gesi, Dakuo Wang, Weiqi Zhang, Houyu Zhang, Yufan Guo, Xian Li, Bing Yin, and Sujay Sanghavi. On-policy self-distillation for efficient diffusion language models with early-stage calibration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26954–26965, 2026. doi: 10.18653/v1/2026.findings-acl.1344. URL https://aclanthology.org/2026.findings-acl.1344/.

Yuanzhi Zhu, Xi Wang, Stéphane Lathuilière, and Vicky Kalogeiton. Di[M]O: Distilling masked diffusion models into one-step generator, 2025. URL https://arxiv.org/abs/2503.15457.

A Analysis and Derivations

Matching intermediate predictions can stabilize the wrong output law; completed-text matching makes an attainable teacher law a stable optimum. The derivations below connect this objective to the student's sampled refinement actions.

A.1 An attainable target and an incorrect masked-ELBO optimum

The threshold decoder can represent the target exactly, yet the population masked-ELBO minimum produces a different completed-text distribution. The construction uses two categorical fields; each field may span several tokenizer tokens. All four target outcomes and all six Bernoulli prediction tables have strictly positive probabilities.

Let M denote a masked field. The six table entries are probabilities of predicting zero:

$$\begin{aligned} a &= p(C=0 \mid M, M), & b &= p(N=0 \mid M, M), \\ c_j &= p(C=0 \mid M, N=j), & d_j &= p(N=0 \mid C=j, M), \quad j \in \{0,1\}. \end{aligned} \tag{11}$$

Given a partially masked state, independently sample a candidate for every masked field, at temperature one without truncation. If h_i is the model probability of the sampled candidate at masked position i, commit the set

$$S = \left\{ i : h_i \geq \min\left(.9, \max_{j \in \mathcal{M}} h_j - \epsilon_{\text{tie}} \right) \right\}, \qquad \epsilon_{\text{tie}} = 10^{-5}, \tag{12}$$

where $\mathcal{M}$ is the set of masked positions. Unselected proposals are discarded and committed fields remain fixed. At least one field is committed per step, so the two-field process terminates in at most two steps. We also check a top-one fallback: commit every candidate with $h_i > .9$ when any passes; otherwise commit the highest-confidence candidate, resolving ties in favor of the country. These variants differ only in their fallback behavior for the comparisons below.

For the target in Eq. (1), both zero marginals are $m = .95$. The conditional probability of zero given that the other field is zero is $r_0 = 94/95$; given one it is $r_1 = 1/5$. Consider the following tables:

	a	b	c_0	c_1	d_0	d_1
Target-matching	$91/100$	$1/2$	$7/9$	$1/9$	$181/182$	$1/5$
ELBO optimum	$19/20$	$19/20$	$94/95$	$1/5$	$94/95$	$1/5$

(13)

At the target-matching tables, a city-zero proposal has confidence .91 and is committed alone; a city-one proposal has confidence .09 and loses to the country proposal's confidence .5, so only the country is committed. The second step samples the other field from its conditional table. With the other five parameters fixed, these selection branches hold throughout $b \in [.5, .6]$, and the completed-pair distribution is

$$\begin{aligned} p_{00} &= ad_0 + (1-a)bc_0 = .905 + .07b, \\ p_{01} &= a(1-d_0) + (1-a)(1-b)c_1 = .015 - .01b, \\ p_{10} &= (1-a)b(1-c_0) = .02b, \\ p_{11} &= (1-a)(1-b)(1-c_1) = .08(1-b). \end{aligned} \tag{14}$$

Setting $b = .5$ gives $p = q$ exactly, with two steps on every trajectory under either fallback rule. In particular, 00 and 01 each aggregate probability from two different commitment orders.

For time-independent tables, the standard absorbing-mask negative ELBO with linear mask schedule is (Sahoo et al., 2024; Su et al., 2026)

$$\mathcal{L}_{\text{mask}} = -\mathbb{E}_{x \sim q,\, t \sim U(0,1),\, s \sim \text{mask}_t(x)} \left[\frac{1}{t} \sum_{i \in \mathcal{M}(s)} \log p_i(x_i \mid s) \right]. \tag{15}$$

Write $\text{CE}(u, v) = -u \log v - (1-u)\log(1-v)$. A both-masked prediction has integrated weight $\int_0^1 t\, dt = 1/2$; a prediction with the other field visible has weight $\int_0^1 (1-t)\, dt = 1/2$, multiplied by that field's marginal. Hence

$$\begin{aligned} \mathcal{L}_{\text{mask}} = {} & \tfrac{1}{2}\,\text{CE}(m, a) + \tfrac{1}{2}\,\text{CE}(m, b) \\ & + \tfrac{m}{2}\big[\,\text{CE}(r_0, c_0) + \text{CE}(r_0, d_0)\big] \\ & + \tfrac{1-m}{2}\big[\,\text{CE}(r_1, c_1) + \text{CE}(r_1, d_1)\big]. \end{aligned} \tag{16}$$

All weights are positive. The unique minimizer is therefore the second row of Eq. (13), and the minimum equals $H(q)$ by the entropy chain rule. Each term is strictly convex in its logit, with an interior target; gradient descent at the tested step sizes converges to this minimum. The largest Hessian eigenvalue is at most $1/8$, so step sizes 1 and $.1$ satisfy the usual smooth convex descent condition.

At this optimum, the initial proposal probabilities are $(.9025, .0475, .0475, .0025)$. Proposal 00 is committed immediately. In each mixed proposal, only the zero-valued field is retained and the other field is resampled from its conditional. Proposal 11 has tied confidences $.05$, so both values are retained under Eq. (12). Consequently,

$$p^{\mathrm{ELBO}} = (.9965, .0005, .0005, .0025), \qquad \mathrm{TV}(p^{\mathrm{ELBO}}, q) = .0565. \tag{17}$$

Together with Eq. (14), this proves Proposition 1. The top-one fallback instead yields $(.9965, .001, .0005, .002)$ with country tie priority, or swaps the two mixed-pair probabilities with city tie priority; the total-variation gap is unchanged.

The masked ELBO remains a likelihood bound for its associated ancestral reverse process. Confidence-based commitment changes the sampling law; Proposition 1 concerns the completed-text distribution of that deployed decoder.

An oracle on-policy control makes the state/objective distinction explicit. Provide the exact $q(X_i \mid s)$ conditional at each student-visited masked state and minimize a positively weighted sum of token KLs on those states. The ELBO-optimal tables match every such conditional exactly, attaining zero local KL and thus a global minimum of this oracle loss, while their output remains Eq. (17). This oracle control is distinct from the causal AR teacher targets studied next. Both train intermediate predictions and are evaluated on the deployed threshold decoder.

A.2 A stable incorrect output under OPDLM's token-level update

OPDLM's expected update also has a locally stable output that suppresses the minority response mode under this decoder.

Let τ be a diffusion student trajectory, $x = (C, N)$ its completed pair, and k a uniformly selected nonterminal step. Following the target rule and uniform state weighting of OPDLM's Section 4.2 objective ($\mathcal{L}_{\mathrm{OPDLM}}$, unnumbered) and Algorithm 1 (Su et al., 2026), use

$$\ell(\theta; \tau, k) = \sum_{i \in \mathcal{M}(s_k)} \mathrm{KL}(q(X_i \mid x_{<i}) \,\|\, p_{\theta,i}(X_i \mid s_k)) \,. \tag{18}$$

The city teacher is $q(C)$; the country teacher is $q(N \mid C)$ evaluated at the completed city. Let ϑ collect the six table logits. The update field is $g(\vartheta) = \mathbb{E}_{\tau,k}[\nabla_\vartheta \ell]$, with the sampled trajectory and teacher targets held fixed inside the derivative. This expected supervised update holds trajectories and targets fixed, as in the published objective, and applies it to the finite prediction table.

In the region $b > a > .9$, with $b - a > \epsilon_{\mathrm{tie}}$, initial proposal 00 commits both fields, 01 commits the city, 10 commits the country, and 11 commits the city. Both fallback rules agree. Define

$$W = \frac{1 + ab}{2}, \qquad V = \frac{(1-a)[b(1-c_0) + (1-b)]}{2}. \tag{19}$$

Here W is the expected weight of the initial state under uniform nonterminal-step selection: one-step trajectories receive weight one and two-step trajectories weight $1/2$. Its portion associated with completed city one is V. Exact enumeration gives

$$\begin{aligned} g_a &= W(a - m), & g_b &= W(b - r_0) + (r_0 - r_1)V, \\ g_{c_0} &= \tfrac{(1-a)b}{2}(c_0 - m), & g_{c_1} &= 0, \\ g_{d_0} &= \tfrac{a(1-b)}{2}(d_0 - r_0), & g_{d_1} &= \tfrac{(1-a)(1-b)}{2}(d_1 - r_1). \end{aligned} \tag{20}$$

Subscripts identify the corresponding logit derivative. A fixed point has $a = c_0 = m$, $d_0 = r_0$, $d_1 = r_1$, and

$$380b^2 + 9b - 380 = 0, \qquad b_\star = \frac{\sqrt{577681} - 9}{760} \simeq .9882280100. \tag{21}$$

This root lies strictly inside the assumed selection region. The context with masked city and visible country one is unvisited there, so c_1 is a free coordinate that does not affect the output. The resulting family of fixed points shares the final distribution

$$\begin{aligned} p^\star_{00} &= .94 + .0575 b_\star, \quad p^\star_{01} = .01(1 - b_\star), \\ p^\star_{10} &= .01 - .0075 b_\star, \quad p^\star_{11} = .04(1 - b_\star), \end{aligned} \tag{22}$$

or approximately $(.99682311, .00011772, .00258829, .00047088)$. Thus $\mathrm{KL}(p^\star \| q) \simeq .05239412$ and the Shanghai–China mode is suppressed by a factor of $1/(1 - b_\star) \simeq 84.95$.

For stability, order the active logits as (a, c_0, b, d_0, d_1). The Jacobian of g at this point is triangular, with positive diagonal entries

$$\begin{aligned} \lambda_a &= \tfrac{1+mb_\star}{2} m(1-m), & \lambda_{c_0} &= \tfrac{(1-m)b_\star}{2} m(1-m), \\ \lambda_b &= (.01125 + .95 b_\star) b_\star (1 - b_\star), & & \\ \lambda_{d_0} &= \tfrac{m(1-b_\star)}{2} r_0(1-r_0), & \lambda_{d_1} &= \tfrac{(1-m)(1-b_\star)}{2} r_1(1-r_1). \end{aligned} \tag{23}$$

These are approximately .04605, .001174, .01105, 5.824×10^{-5}, and 4.709×10^{-5}. For the logit update $\vartheta \leftarrow \vartheta - \eta g(\vartheta)$, every active eigenvalue $1 - \eta\lambda$ lies strictly inside the unit circle for $\eta = 1$ and .1. The fixed output is therefore locally attracting under this update; the unused c_1 direction is neutral and leaves the output unchanged.

Starting from the target-matching tables, all eight population-update checks converged to the predicted limits: masked ELBO and on-policy token KL, each with both fallback rules and both learning rates. Stopping required a maximum expected logit-gradient magnitude below 10^{-11} and output error from the derived limit below 10^{-8}. The masked runs reached this gate at 5,000 and 50,000 updates; the on-policy runs at 264,000 and 2,637,000. At the on-policy limit, the local KL is .00474071, down from .35811818 initially, while the final-text KL has increased from zero to .05239412.

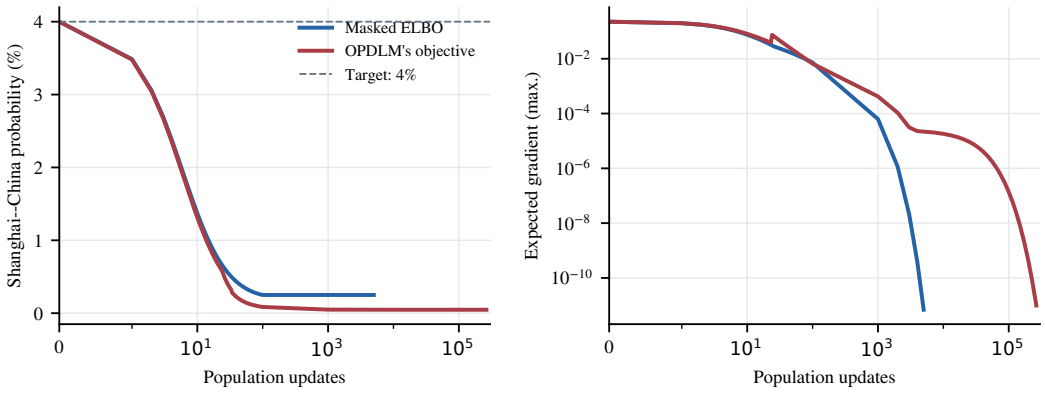

Figure 6: Population optimization from tables that initially generate the target exactly. The OPDLM curve applies its published token-level update to this finite-state construction. The Shanghai–China probability moves away from its target as both training procedures converge. The curves use the fallback in Eq. (12) and step size 1; step size .1 and the top-one fallback give the same qualitative result and the analytically predicted limits.

The two-step cap remains fixed, but adaptive commitment changes realized step counts: they average 1.095 at the masked-ELBO optimum and approximately 1.06118 at the on-policy limit, compared with two initially. Increasing the unmasking cap alone cannot repair a response whose fields are already committed. The next calculation isolates the objective mismatch while holding the realized step count fixed.

A.3 On-policy drift without changing selection or step count

The objective mismatch persists with exactly two decoding steps and unchanged commitment decisions. At the target-matching tables, hold five entries fixed and increase the both-masked country

probability from $b = .5$ to $.6$. The expected AR teacher target for this prediction is $.95$, because the completed distribution is initially q. The initial state receives weight $1/2$, so its expected logit gradient is $g_b = \frac{1}{2}(.5 - .95) = -.225$. A coordinate step of size $\eta = \log(1.5)/.225$ makes precisely this change. Equation (14) then gives

$$p^{\text{after}} = (.947, .009, .012, .032), \qquad \mathrm{KL}(p^{\text{after}} \| q) \simeq .00112502 > 0. \tag{24}$$

The on-policy token KL, re-evaluated on fresh student trajectories, decreases from $.35811818$ to $.26945286$. Every response still uses exactly two steps, and each initial proposal pair selects the same field as before. Thus this update reduces the actual expected local loss while moving an exactly matched output away from the target, without a threshold crossing or a change in realized decoding budget. An ordinary simultaneous update of all six logits at step size $.1$ gives the same separation: local KL decreases to $.35257170$ while final-text KL rises to 4.2236×10^{-6}, with the same selection branches and two steps throughout.

A.4 Terminal-distribution gradient

Proposition 2 (Completed-text optimum)**.** *For a fixed prefix h_b, $J_b(\theta) \geq 0$, with equality exactly when $p_\theta^{\text{text}}(\cdot \mid h_b) = q_\phi(\cdot \mid h_b)$. If the decoder can represent that distribution, it is a global optimum. On a finite output space with positive teacher and student probabilities, it is also the only stationary distribution in the interior simplex: there are no incorrect interior local minima of the completed-text KL. Matching every completed-block conditional at each reachable prefix also matches the joint response distribution.*

Proof. The first two statements follow from nonnegativity of KL and its equality condition for normalized distributions. For interior probabilities, $\partial J/\partial p(x) = \log(p(x)/q(x)) + 1$. Constrained stationarity requires these derivatives to be equal for all x, so $p(x)/q(x)$ is constant; normalization gives $p = q$. The Hessian $\mathrm{diag}(1/p(x))$ is positive on every nonzero simplex-tangent direction. For the last statement, factor each response probability into successive completed-block conditionals. Equality of the first conditional gives equal first-block laws; induction gives equal prefix laws and equal next-block conditionals, hence equal joint laws. The same argument applies to variable-length responses with absorbing EOS. □

This uniqueness also holds for exact parameter-gradient stationary points where the decoder's output Jacobian spans the simplex tangent space. That condition permits local redistribution of output probability; positive sampling support alone does not imply it. We next derive the on-policy update for this objective.

We give the identity for a fixed conditioning context c. For the per-block update, take $c = h_b$, $x = x_b$, and $\tau = \tau_b$; averaging over fixed sampled prefixes is immediate. Assume a fixed discrete trajectory space, a parameter-independent map F from trajectories to text, positive teacher probability on student-supported text, and differentiability with interchange of sums and derivatives justified. For a finite space it is sufficient to work on a differentiable region with fixed support; for unbounded text, the corresponding integrability conditions are required. Model-dependent proposal quantities are recomputed as functions of the fixed sampled discrete history and remain part of the differentiated trajectory law. Kinks in accept/reject or clipping require their usual piecewise interpretation.

Writing $p = p_\theta^{\text{text}}$ and $P = P_\theta$,

$$\nabla J = \sum_x \nabla p(x) \left(\log \frac{p(x)}{q_\phi(x)} + 1 \right) \tag{25}$$

$$= \sum_x \sum_{\tau : F(\tau) = x} \nabla P(\tau) \log \frac{p(x)}{q_\phi(x)} \tag{26}$$

$$= \mathbb{E}_\tau \left[\log \frac{p(F(\tau))}{q_\phi(F(\tau))} \nabla \log P(\tau) \right]. \tag{27}$$

The $+1$ term disappears because $\sum_x \nabla p(x) = 0$. With the feedback in Eq. (2), this gives the distribution-matching instance of Eq. (4); the density score is treated as a fixed coefficient when estimating this gradient.

Across blocks, we use each block's feedback at its sampled prefix, extending conditional AR distillation to blocks. A whole-response KL gradient would add the effects of earlier decisions on later prefixes and outcomes; those cross-block terms are outside this conditional objective.

A.5 Completed-text matching makes the target a stable optimum

Completed-text matching has its minimum at the distribution we want the decoder to produce. In the city–country construction, the teacher target is stable, while the masked-ELBO minimum and OPDLM fixed point still admit a correction toward it.

Let p be the four-outcome distribution generated by the city–country student and $J(p) = \mathrm{KL}(p\|q)$, with q from Eq. (1). The target is attainable under the same threshold decoder. It is the unique distribution with $J = 0$; every different output law has positive loss. More precisely, for a small perturbation δ with $\sum_x \delta_x = 0$,

$$J(q+\delta) = \frac{1}{2}\sum_x \frac{\delta_x^2}{q_x} + O(\|\delta\|^3). \tag{28}$$

All q_x are positive, so the curvature is positive in every direction that changes the completed-text distribution.

This stability is realized by the finite decoder's prediction tables. At the target-matching tables in Eq. (13), changing c_0, c_1, and d_0 transfers probability between $00/10$, $01/11$, and $00/01$, respectively. These three independent directions span the probability simplex locally. The loss therefore has positive curvature transverse to the set of parameters producing q, and sufficiently small exact-gradient updates locally attract the output law toward q. Parameter directions that preserve the output law leave the objective unchanged. With exact student density, matching q also gives $d_i = 0$ for every emitted token, so both the shared return and the suffix credits in Eq. (8) vanish at the target.

The limits of the masked ELBO and OPDLM's objective behave differently. Let $v = (-1, 0, 0, 1)$ transfer a small amount of probability from the overproduced New York–United States outcome to the suppressed Shanghai–China outcome. Then

$$\frac{d}{d\epsilon} J(p+\epsilon v)\Big|_{\epsilon=0} = \log \frac{p_{11}/q_{11}}{p_{00}/q_{00}}. \tag{29}$$

This derivative is -2.831 at the masked-ELBO minimum and -4.501 at the OPDLM fixed point. The prediction tables admit this direction locally: (a, c_0, d_0) span the output simplex at the former point, and (c_0, d_0, d_1) do so at the latter. Thus neither limit is stationary for the completed-text objective. Where these objectives have stopped, completed-text feedback still provides a descent direction that restores probability to the missing response mode.

Appendix A.4 derives the decoder-induced gradient of this objective. For our parallel refinement decoder, Appendix A.6 shows how shared intermediate feedback and final suffix feedback estimate that gradient at each fixed block prefix.

A.6 Dependency-based credit preserves the completed-block gradient

The block-mean and final-suffix credits in Eq. (8) give the same expected update as the normalized completed-block KL gradient. We first establish the shared-return identity, then use the refinement and likelihood dependencies to remove terms with zero expected contribution. Throughout, we condition on the incoming block prefix and a fixed refinement budget T_b.

Let $\mathcal{F}_{b,0}$ contain the incoming context and let $\mathcal{F}_{b,t}$ contain the block's history through step t. Include all stochastic initialization and refinement events in the recorded generation law. For bounded variable-length trajectories, pad the trace after termination with deterministic dummy actions whose scores are zero. Assume normalized conditional action laws, justified interchange of derivatives and sums, and integrability of the score-return products, as in Appendix A.4. Feedback is held fixed inside the score-function update.

Our dependency structure is block-wide across refinement steps: every token conditions on all tokens produced in the preceding step. The draws within one step factorize given that shared history, but

future refinements couple their effects on the completed block. Proposal probabilities are recomputed from the discrete token history at the current parameters, and their derivatives enter each action score. Because Eq. (9) is normalized for every value of ρ (Appendix B.1), each transition law remains normalized as a function of θ at a fixed discrete history. This structure permits a path from any intermediate action to every final position. For example, a provisional token at position 1 can change the next prediction at position 2, whose effect can survive even if position 1 is subsequently overwritten.

Write a_{bt} for a parallel action vector and $G_{bt} = \nabla_\theta \log K_\theta(a_{bt} \mid \mathcal{F}_{b,t-1})$. Differentiating the conditional normalization gives

$$\begin{aligned}\mathbb{E}[G_{bt} \mid \mathcal{F}_{b,t-1}] &= \sum_a K_\theta(a \mid \mathcal{F}_{b,t-1}) \nabla_\theta \log K_\theta(a \mid \mathcal{F}_{b,t-1}) \\ &= \nabla_\theta \sum_a K_\theta(a \mid \mathcal{F}_{b,t-1}) = 0.\end{aligned} \tag{30}$$

Thus G_{bt} is a martingale difference and $S_{bt} = \sum_{s \le t} G_{bs}$ is a score martingale. Any integrable baseline B_{bt} known before step t satisfies $\mathbb{E}[B_{bt} G_{bt}] = 0$. This is the causal-baseline property of score-function estimators (Schulman et al., 2015).

For completed-block feedback R_b, define $M_{bt} = \mathbb{E}[R_b \mid \mathcal{F}_{b,t}]$. The tower property implies $\mathbb{E}[M_{bt} \mid \mathcal{F}_{b,t-1}] = M_{b,t-1}$, so (M_{bt}) is the Doob martingale of the block outcome. Since G_{bt} is measurable after the step,

$$\begin{aligned}\mathbb{E}[R_b G_{bt}] &= \mathbb{E}[\mathbb{E}[R_b \mid \mathcal{F}_{b,t}] G_{bt}] \\ &= \mathbb{E}[M_{bt} G_{bt}] \\ &= \mathbb{E}[(M_{bt} - M_{b,t-1}) G_{bt}].\end{aligned} \tag{31}$$

The equality means that the sampled completed-block return gives the same expected score update as the change in conditional expected outcome at that step. It supplies an exact credit identity in terms of the observed return.

Our parallel transition law factorizes given the pre-step history,

$$K_\theta(a_{bt} \mid \mathcal{F}_{b,t-1}) = \prod_{i \in \mathcal{A}_{bt}} K_{\theta,i}(a_{bti} \mid \mathcal{F}_{b,t-1}), \qquad G_{bt} = \sum_{i \in \mathcal{A}_{bt}} g_{bti}. \tag{32}$$

Here $g_{bti} = \nabla_\theta \log K_{\theta,i}(a_{bti} \mid \mathcal{F}_{b,t-1})$. Each token-action score also has conditional mean zero. Therefore Eq. (31) holds separately for each g_{bti}, and sharing R_b across those token rows reproduces the block-event score update:

$$\mathbb{E}[R_b G_{bt}] = \sum_{i \in \mathcal{A}_{bt}} \mathbb{E}[R_b g_{bti}]. \tag{33}$$

To connect this identity to the dependency structure, take a fixed valid block span V_b and $R_b = \sum_{j \in V_b} d_j$. Then

$$\mathbb{E}[R_b g_{bti}] = \sum_{j \in V_b} \mathbb{E}[d_j g_{bti}]. \tag{34}$$

A term vanishes if d_j is conditionally independent of a_{bti} given $\mathcal{F}_{b,t-1}$: conditioning on that history factors the product and the score has zero mean. For intermediate actions, the block-wide dependencies provide no such justification for dropping any d_j. Each final position can carry feedback about the earlier action, so retaining the full return accounts for all these paths. This is the downstream-cost principle for stochastic computation graphs (Schulman et al., 2015), applied to iterative block generation.

Each action's score extracts its own expected contribution from the shared return. The conditional zero-mean property gives

$$\mathbb{E}[R_b g_{bti}] = \mathbb{E}[(\mathbb{E}[R_b \mid \mathcal{F}_{b,t-1}, a_{bti}] - M_{b,t-1})\, g_{bti}]\,. \tag{35}$$

Thus each action's score selects its own contribution in expectation from the sampled block return. The dependency structure determines which outcomes must be accounted for; the martingale identities justify using their observed sum.

The shared-return identity also holds per action:

$$\mathbb{E}[R_b g_{bti}] = \mathbb{E}[(M_{bt} - M_{b,t-1}) g_{bti}]. \tag{36}$$

The observed return thus credits an action's change in expected final feedback.

At the final step, every position is sampled independently conditional on $\mathcal{F}_{b,T_b-1}$. Let $\mathcal{H}_{bi} = \sigma(\mathcal{F}_{b,T_b-1}, x_{b,<i})$ include the earlier final tokens. Conditional independence gives $\mathbb{E}[g_{bT_bi} \mid \mathcal{H}_{bi}] = 0$. For $j < i$, d_j is measurable with respect to $\mathcal{H}_{bi}$ because both teacher and auxiliary likelihood factors condition only on the completed prefix through j. Therefore

$$\mathbb{E}[d_j g_{bT_bi}] = \mathbb{E}[d_j \mathbb{E}[g_{bT_bi} \mid \mathcal{H}_{bi}]] = 0, \qquad j < i. \tag{37}$$

The remaining terms give the exact suffix identity

$$\mathbb{E}[R_b g_{bT_bi}] = \mathbb{E}\left[\left(\sum_{j \in V_b:\, j \geq i} d_j\right) g_{bT_bi}\right]. \tag{38}$$

Later likelihood factors can depend on x_{bi} through their AR prefixes, so retaining only d_i would discard part of this update. The suffix is computed from the completed text's existing token scores.

For configured block width B, use the same $1/B$ for every action in Eq. (8). Token feedback d_j outside the completed response is zero; the trajectory sum retains intermediate actions that can influence retained outputs, including actions beyond the eventual EOS position. B does not change when a block contains fewer valid output tokens. Combining the intermediate and final identities yields

$$\mathbb{E}\left[\sum_{(t,i)\in\mathcal{A}_b} A_{bti} g_{bti}\right] = \frac{1}{B}\mathbb{E}\left[R_b \sum_{(t,i)\in\mathcal{A}_b} g_{bti}\right] = -\frac{1}{B}\nabla_\theta J_b, \tag{39}$$

where the last equality uses exact student density and the trajectory conditions in Appendix A.4. The first equality holds for any integrable causal token feedback, including an estimated AR density ratio: credit decomposition introduces no additional change in its expected score update. A common normalization sets the scale of J_b/B for intermediate and final actions alike.

Appendices B.2 and B.3 specify feedback clipping, centering and variance normalization used in training. The implemented sequence is token-feedback clipping, credit assignment, then centering and variance normalization.

Formal statement. The following proposition collects the guarantees above and their limits.

Proposition 3 (Objective and estimator)**.** *For a sampled prefix h_b let $J_b(\theta; h_b) = \mathrm{KL}(p_\theta^{\mathrm{text}}(\cdot \mid h_b) \,\|\, q_\phi(\cdot \mid h_b))$ and $J(\theta) = \mathbb{E}_{\mathrm{student}}[\sum_b J_b(\theta; h_b)]$.*

(i) $J \geq 0$, and $J = 0$ if and only if the student's completed-block conditionals equal the teacher's on all prefixes reachable under the student (almost surely), which recovers the teacher's response distribution. If these conditionals are jointly attainable, they minimize J.

(ii) At each fixed sampled prefix, under unprocessed feedback $d_j = \log q_j - \log p_j$, exact student density, fresh on-policy sampling, and the decoder and regularity assumptions of Appendix A.4, the credits of Eq. (8) satisfy $\mathbb{E}\big[\sum_{(t,i)} A_{bti} g_{bti}\big] = -\nabla_\theta J_b / B$. Prefix-sampling probabilities are held fixed, so these conditional estimators omit prefix-distribution derivatives and do not in general estimate the full gradient of J.

(iii) The practical update—feedback clipping, centering and variance normalization of advantages, estimated density, ratio and dual clipping, and the optional guard—is not in general an unbiased estimator of these gradients.

Part (i) follows from the chain rule of KL over blocks and Appendix A.4; part (ii) is Eq. (39); part (iii) is stated to delimit the guarantee.

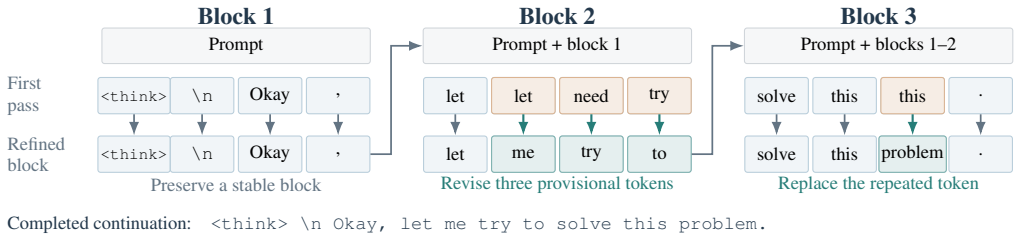

Figure 7: **Both retention and replacement are trainable actions.** Adapted from a recorded trace: three four-token blocks, two iterations each. The decoder records parallel decision probabilities. Highlighted tokens are revised; completed blocks provide context for the next.

Remark: a reward-augmented form of the same loss. Writing $J_b = \mathbb{E}_{x_b \sim p_\theta^{\text{text}}}[\log p_\theta^{\text{text}}(x_b \mid h_b) - \log q_\phi(x_b \mid h_b)]$ shows that minimizing the block KL maximizes the expected teacher log-likelihood plus the entropy of the decoder's completed-text distribution: an entropy-regularized policy objective over the decoder's output law, whose reward is the teacher's log-likelihood. Because the first equality in Eq. (39) holds for any integrable causal token feedback, adding a causal token reward βr_j to d_j gives, under exact density and on-policy sampling, the same estimator form for the objective $J_b - \beta\,\mathbb{E}[\sum_j r_j]$, a teacher-regularized reward objective. Effects on later blocks remain omitted, as in the distillation case. We note this as a natural extension of the estimator; we do not evaluate it here.

A.7 Auxiliary-density error

The auxiliary supplies student likelihoods for the completed-text ratio. Its estimation error matters through the resulting update direction.

At fixed θ, Eq. (6) equals $H(p_\theta^{\text{text}}) + \mathrm{KL}(p_\theta^{\text{text}} \| \hat{p}_\psi)$ after averaging over prompts. Population maximum likelihood therefore targets the actual student output law within the auxiliary model class. In the realizable case, its optimum is that law.

Let $e(x, c) = \log \hat{p}_\psi(x \mid c) - \log p_\theta^{\text{text}}(x \mid c)$ and $G(\tau, c) = \nabla \log P_\theta(\tau \mid c)$. For the full-return estimator, replacing the true density changes the expected update by $\Delta g = -\mathbb{E}[e(F(\tau), c)G(\tau, c)]$. Hence

$$\|\Delta g\| \le \sqrt{\mathbb{E}[e^2]}\sqrt{\mathbb{E}[\|G\|^2]}. \tag{40}$$

This Cauchy–Schwarz bound relates log-density error to update error. The identity in Eq. (39) also holds for the estimated AR ratio, so the normalized block update has error $\Delta g / B$. Exact density recovery is stronger than needed for a useful update. Let $g = -\nabla J$ and $\widehat{g} = g + \Delta g$. Within a differentiable region,

$$J(\theta + \eta\widehat{g}) = J(\theta) - \eta\big(\|g\|^2 + \langle g, \Delta g\rangle\big) + o(\eta). \tag{41}$$

Thus $\langle g, \widehat{g}\rangle > 0$ gives descent for sufficiently small positive η. For example, $\|\Delta g\| < \|g\|$ suffices when $g \neq 0$. The relevant first-order quantity is the error's projection onto the true descent direction.

B Practical Recipe and Reproducibility

The recipe alternates density fitting with one on-policy student update, using separate samples for the two objectives. Generation and training evaluate the same decoder actions and proposal dependencies.

B.1 Decoder action law and terminal emissions

The framework requires the probability of each sampled decoder event, including retention as well as replacement. We instantiate it with a stochastic refinement rule whose event probabilities can be evaluated during replay, while allowing later states to revise provisional tokens.

The output law is induced jointly by the model and its decoder. Replacing the keep-or-replace rule with threshold-based or irreversible commitment changes that law, even with the same weights and

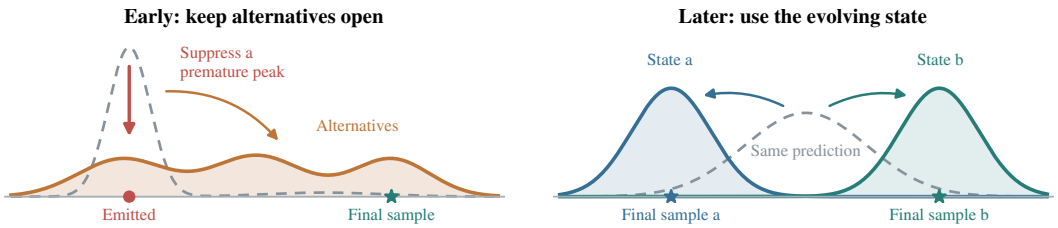

Figure 8: **Intended effect of the guard (schematic).** Early suppression of provisional tokens that disagree with the final sample is meant to keep alternatives available; later tracking of that sample encourages use of the evolving state. Dashed profiles show premature concentration or state-insensitive predictions. Probabilities are illustrative.

iteration count. Alignment learned under the original decoder need not carry over to the replacement. To target another decoder's output distribution, its trajectory law and action probabilities enter the objective and estimator, subject to the conditions in Appendix A.4.

At a new block, unfilled response positions contain mask tokens. The first iteration samples each position directly from its categorical model prediction. Predictions are next-aligned: the last prefix position predicts the first block token, and each preceding block position predicts the next token. Subsequent iterations retain or replace these proposals in parallel using Eq. (9), conditioned on the preceding block state. The model uses a causal completed prefix and bidirectional active-block attention. After T_b iterations, including the initial categorical pass, the final block is committed. A full block thus has BT_b emitted-token action records; prompt, padding, and stopped positions are excluded. Mask initialization is deterministic and contributes no sampled action.

Equation (9) defines the emitted-token law. At each position, let u_t be the token emitted at iteration t and p_t its current vocabulary distribution. After either retention or replacement, we store $\rho_t = p_t(u_t)$; the next iteration compares $p_{t+1}(u_t)$ with ρ_t. The initial categorical draw similarly stores $\rho_1 = p_1(u_1)$. Thus retaining a token refreshes its stored vocabulary probability. This quantity is distinct from the emitted-token action probability π_θ and is recomputed from the preceding iteration during training. The history s specifies the sampled tokens and attention context. Proposal probabilities are recomputed from that history at the current parameters; training includes both current- and proposal-side derivatives (Appendix C.2). The reported runs use a log-probability tolerance $\varepsilon_{\mathrm{acc}} = 10^{-3}$ at the acceptance boundary. Writing $\Delta = \log p(u) - \log \rho(u)$, their retention probability is

$$\alpha_{\mathrm{acc}} = \begin{cases} 1, & \Delta \geq -\varepsilon_{\mathrm{acc}}, \\ \exp(\Delta), & \Delta < -\varepsilon_{\mathrm{acc}}. \end{cases} \tag{42}$$

The emitted-token law uses α_{acc} in place of α in Eq. (9); $\varepsilon_{\mathrm{acc}} = 0$ recovers the displayed main-text rule. Sampling and training use the same branch test.

First-iteration categorical draws and later keep-or-replace emissions all enter the path probability. Intermediate iterations receive block feedback; the last iteration receives suffix feedback as specified in Eq. (8). Appendix A.6 derives both credit branches from their dependencies.

For $0 \leq \alpha < 1$, the replacement probabilities in Eq. (9) sum to $1 - \alpha$; adding retention gives a normalized action law. At $\alpha = 1$, the action retains u with certainty. When $p(u) > \rho(u)$, this locally certain retention has zero policy-score gradient. Probabilistic retentions and replacements carry the outcome-weighted feedback. The same normalization holds with α_{acc}.

B.2 Complete regularizer and policy update

The policy term trains actions through completed-text feedback. The guard adds a cross-entropy toward the block's final sample, weighted increasingly with the step, and a penalty on the probability of provisional tokens that disagree with it, weighted decreasingly. Together they form Eq. (10); the recipe below also specifies advantage processing and mismatch correction. Appendix B.5 records the credit variant and other settings used by the reported checkpoints.

In Eq. (8), intermediate refinements have $t < T_b$, where T_b is the last step for block b. Final emissions are the last refinement's actions, with suffix credit from their position onward. The implementation first bounds token feedback, $\delta_i = \mathrm{clip}(d_i, -C_{\mathrm{clip}}, C_{\mathrm{clip}})$, and applies Eq. (8) to δ to obtain A^{raw}. The fixed denominator B bounds every raw credit by C_{clip} and keeps the same scale for intermediate and final actions. Non-direct rows below are keep-or-replace refinement actions; first-iteration categorical draws are direct actions (Appendix B.1). Appendix B.3 specifies centering and whitening, $A = \mathcal{P}(A^{\mathrm{raw}})$.

For non-direct rows with $1 < t < T_b$, emitted token y_{bti}, detached final token x_{bi}, and raw prediction p_{bti}, define

$$a_{bt} = \frac{2t}{T_b}, \qquad c_{bti} = \eta_{\mathrm{g}} \frac{2(T_b - t)}{T_b} \mathbf{1}\{y_{bti} \neq x_{bi}\}. \tag{43}$$

$$\mathcal{L}_{\mathrm{guard}} = \frac{1}{Z_{\mathrm{g}}} \sum_{(b,t,i)\in\mathcal{G}} \left[\underbrace{-a_{bt} \log p_{bti}(x_{bi})}_{\text{track the terminal sample}} + \underbrace{c_{bti} p_{bti}(y_{bti})}_{\text{suppress mismatched concentration}} \right], \tag{44}$$

where $\mathcal{G}$ is the eligible-row set and $Z_{\mathrm{g}} = \sum_{\mathcal{G}} (a_{bt} + c_{bti})$. A batch with no eligible rows contributes zero guard loss. The guard's terminal-tracking term uses detached final-token labels, while the policy update assigns outcome-based credit. The guard acts on raw prediction probabilities and leaves the action law of Eq. (9) in place.

Let ℓ^{beh} be the recorded rollout log-probability and ℓ^{old} the fixed replay reference. Define token policy ratios, shared block-event importance weights, and the clipped surrogate:

$$\begin{aligned} r_{bti} &= \exp(\ell_{bti} - \ell^{\mathrm{old}}_{bti}), \\ \omega_{bt} &= \mathrm{sg}\left[\min\left\{ w_{\max}, \exp\left(\sum_i \ell^{\mathrm{old}}_{bti} - \sum_i \ell^{\mathrm{beh}}_{bti} \right) \right\} \right]. \end{aligned} \tag{45}$$

$$h(r, A) = \min\{rA, \mathrm{clip}(r, 1 - \epsilon_-, 1 + \epsilon_+)A\}. \tag{46}$$

For negative advantages, dual clipping additionally bounds the surrogate:

$$h_{\mathrm{dual}}(r, A) = \begin{cases} \max\{h(r, A), c_{\mathrm{dual}} A\}, & A < 0, \\ h(r, A), & A \geq 0, \end{cases} \qquad c_{\mathrm{dual}} > 1. \tag{47}$$

The implemented loss with this safeguard is

$$\mathcal{L}_{\text{DM-OPD}} = -\frac{1}{Z} \sum_{(b,t,i)\in\mathcal{A}} \omega_{bt}\, h_{\mathrm{dual}}(r_{bti}, A_{bti}) + \lambda \mathcal{L}_{\mathrm{guard}}. \tag{48}$$

Z is the globally agreed policy normalization mass. The joint block-event correction is shared across token rows because their credit can depend on other simultaneous actions. Scores, advantages, and the old reference remain fixed inside the replay loop. At a fresh update with identical rollout and replay probabilities, $r_{bti} = \omega_{bt} = 1$: neither policy-ratio clip is active, and the gradient reduces to the action-score form in Eq. (10), using the processed advantages A. This concerns ratio clipping; advantage preprocessing remains as specified in Appendix B.3.

The auxiliary is initialized from the teacher's AR checkpoint, so the teacher–auxiliary log-likelihood feedback starts near zero. Subsequent auxiliary updates fit the auxiliary to student outputs. Each fresh training stage uses this initialization; continuations restore the auxiliary parameters, optimizer, and scheduler state.

Each iteration generates a policy batch $\mathcal{B}_\theta$ of 128 responses and four separate auxiliary batches $\mathcal{B}^{(k)}_\psi$ of 128 responses each. The two prompt streams are sampled independently, with one response per sampled request. We fit the auxiliary first, then score the policy batch with the updated auxiliary and perform one student update ($U = 1$).

Algorithm 1: DM-OPD: training cycle

Models: diffusion student θ, frozen AR teacher q_ϕ, trainable AR density $\hat{p}_\psi$;
learning rates $\gamma_\theta, \gamma_\psi$.

1. $\theta_{\mathrm{beh}} \leftarrow \mathrm{Sync}(\theta)$
2. $\mathcal{B}_\theta, \{\mathcal{B}_\psi^{(k)}\}_{k=1}^4 \leftarrow \mathrm{Generate}(\theta_{\mathrm{beh}})$ *separate policy and density samples*
3. **for** $k = 1, \ldots, 4:$ $\psi \leftarrow \psi - \gamma_\psi \nabla_\psi \mathcal{L}_{\mathrm{aux}}(\mathcal{B}_\psi^{(k)})$
4. $d_i \leftarrow \mathrm{sg}[\log q_\phi(x_i \mid c, x_{<i}) - \log \hat{p}_\psi(x_i \mid c, x_{<i})]$ *score final text*
5. $A \leftarrow \mathcal{P}(\mathrm{Credit}(\mathrm{clip}(d, -C_{\mathrm{clip}}, C_{\mathrm{clip}})))$ *Eq.* (8)
6. $\ell^{\mathrm{old}} \leftarrow \mathrm{sg}[\mathrm{Replay}_\theta(\tau)]$ *freeze the reference*
7. $\ell \leftarrow \mathrm{Replay}_\theta(\tau), \quad (r, \omega) \leftarrow$ Eq. (45)
8. $\theta \leftarrow \theta - \gamma_\theta \nabla_\theta \mathcal{L}_{\text{DM-OPD}}(\mathcal{B}_\theta; \ell, A, \omega)$

Here $\mathcal{L}_{\mathrm{aux}}(\mathcal{B})$ is the batch mean negative log-likelihood in Eq. (6). Steps 4–8 use $(c, \tau, x, \ell^{\mathrm{beh}})$ from $\mathcal{B}_\theta$, with $x = F(\tau)$. Replay preserves the sampled attention and proposal contexts, including the first categorical pass. Appendix C describes the packed replay and adjacent-transition gradient implementation.

B.3 Managing training–inference mismatch

Centering removes the unintended policy update caused by a common feedback offset under rollout–replay mismatch. Event-level weights account for simultaneous-action mismatch, and variance normalization sets the update scale.

The reported recipe synchronizes rollout parameters each iteration and performs one student update per policy batch. Serving and replay can still assign different probabilities to the same action because of numerical execution. We preserve attention, proposal history, sampling and stopping rules, and compare recorded behavior likelihoods with a frozen replay reference (Appendix C).

For inference-time trajectory law P_{beh} and replay-reference law P_{old} at fixed prefix h_b, full importance correction would use

$$W_b(\tau_b) = \frac{P_{\mathrm{old}}(\tau_b \mid h_b)}{P_{\mathrm{beh}}(\tau_b \mid h_b)} = \exp\left[\sum_{(t,i)\in\mathcal{A}_b} \left(\ell_{bti}^{\mathrm{old}} - \ell_{bti}^{\mathrm{beh}}\right)\right]. \tag{49}$$

This corrects full-trajectory expectations when the behavior law covers the replay law's support. The product can concentrate weight on very few paths as discrepancies accumulate, making reliable estimates expensive (Espeholt et al., 2018; Marek & Ryabinin, 2026). We use the truncated joint-event weight ω_{bt} in Eq. (45), alongside the token policy ratio and clipping (Schulman et al., 2017). This correction accounts for simultaneous actions; preceding-state and later-refinement mismatch remains.

At the frozen reference, policy ratios equal one. For N valid action rows, write $z_j = \nabla_\theta \ell_j$, let w_j be the detached event weight, and set $\mu_A = N^{-1} \sum_j A_j^{\mathrm{raw}}$. The update decomposes as

$$\frac{1}{N}\sum_j w_j A_j^{\mathrm{raw}} z_j = \frac{1}{N}\sum_j w_j (A_j^{\mathrm{raw}} - \mu_A) z_j + \underbrace{\mu_A \frac{1}{N}\sum_j w_j z_j}_{\text{offset contribution}}. \tag{50}$$

Under mismatch, the last term can survive independently of completed-text quality. Its direction is explicit in the unweighted case: at a fixed refinement state s with frozen rollout law β,

$$\mathbb{E}_{a\sim\beta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s)] = -\nabla_\theta \mathrm{KL}(\beta(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s)). \tag{51}$$

A positive offset pulls the student toward the rollout law; a negative one reverses that direction. Centering removes this common-offset term from the weighted update in Eq. (50).

After feedback clipping and credit assignment, we apply

$$\mu_A = \frac{1}{N}\sum_j A_j^{\mathrm{raw}}, \qquad \sigma_A^2 = \frac{1}{N}\sum_j (A_j^{\mathrm{raw}} - \mu_A)^2, \qquad A_j = \mathrm{sg}\left[\frac{A_j^{\mathrm{raw}} - \mu_A}{\sigma_A + 10^{-6}}\right]. \tag{52}$$

These unweighted moments pool valid action rows across data-parallel ranks before microbatching. A common additive shift of the assigned raw action credits cancels exactly. A constant shift of token feedback before suffix summation can also produce action-dependent offsets; centering removes the common component, while preserving the differences between assigned credits.

The AR score-centering method of Marek & Ryabinin (2026) centers score vectors conditionally at each prefix; we center scalar advantages across the batch.

B.4 Training data, comparison controls, and token budget

Each point on the inference frontier has its own conversion budget, including student and auxiliary updates. We account for these streams at the evaluated checkpoints and specify their shared training data.

Our B4I4 runs use the corresponding Qwen3 checkpoint for student initialization and the frozen AR teacher, following general OPDLM (Su et al., 2026). The full processed OPDLM mixture contains 61,816 prompts: 21,594 code, 20,222 mathematics, 10,000 science, and 10,000 chat examples. Training and auxiliary batches retain all four domains. We use OPDLM's prompt builders with thinking enabled and no added system message; the 1.7B preparation checks establish text/token equality with that configuration on all training prompts and 2,977 monitoring requests. General OPDLM was distilled without thinking, so the two methods render the shared prompts with different templates. Task-reward mixing has weight zero; reward scores are used only for monitoring.

The training response cap follows a block-rounded cosine warmup from 100 to 4,096 tokens over 100 steps. Domain caps are 4,096 for mathematics, 2,048 for code and chat, and 1,024 for science. The general OPDLM script uses a 4,000-token endpoint. The initial checkpoint, teacher checkpoint, and prompt corpus are shared; the decoder, thinking mode, objective, and optimization procedure follow each method's recipe. Table 3 accounts for both student and auxiliary training.

Table 3 reports counters from the evaluated checkpoints. Saved step n contains the state after zero-based update $n - 1$.

Table 3: **Training budgets at the evaluated checkpoints, in millions of tokens.** Student and auxiliary columns count generated response tokens used in their separate update streams. These counters exclude evaluation and count response tokens once per use in an update. All students start from the raw AR model.

Student / step	Student	Aux.	Total
1.7B B4I4 / 197	47.711	170.538	218.249
4B B4I4 / 137	28.041	101.576	129.618
1.7B B4I3 / 209	51.666	168.724	220.390
1.7B B4I2 / 503	148.027	377.761	525.788
1.7B B4I4 non-thinking / 146	20.589	40.914	61.503

Token counts measure data consumption per update stream. For reference, the 1.7B B4I4 recipe had consumed 42.5M student and 151.6M auxiliary response tokens after 180 updates (Table 9 runs), and OPDLM's objective with our decoder (run B) used the same student-token budget with no auxiliary.

B.5 Reported training configuration

The evaluated students share an optimization recipe, with their iteration budgets, initialization and stopping settings specified below. The frozen teacher and trainable auxiliary use the same Qwen3 architecture and size as the student. Table 7 lists the public AR checkpoint revisions used for initialization. The student additionally uses a learned refinement-step embedding.

All reported DM-OPD checkpoints use the credit rule of Eq. (8), with one fixed-width normalization for the block mean and the final suffix sums. Token feedback is clipped at $C_{\text{clip}} = 20$, then we center and normalize the assigned advantages over valid action rows (Eq. (52)). The policy term is normalized by the global valid-action count. The ratio settings are $\epsilon_- = 0.2$, $\epsilon_+ = 0.28$, $c_{\text{dual}} = 3$, and $w_{\max} = 8$; token log-ratios are bounded to $[-20, 20]$ before exponentiation. The guard uses

Table 4: **Common optimization settings.** Batch sizes are global numbers of responses. Auxiliary warmup counts auxiliary optimizer updates, four of which occur per training iteration.

Setting	Student	Auxiliary
Optimizer	AdamW	AdamW
Learning rate	2×10^{-6}	2×10^{-5}
Adam betas	$(0.9, 0.9)$	$(0.9, 0.999)$
Adam epsilon / weight decay	10^{-8} / 0	10^{-8} / 0
Global batch per update	128	128
Updates per training iteration	1	4
Warmup optimizer steps	10	50
Schedule after warmup	Constant	Constant
Gradient-norm limit	4.0	1.0
Global update-norm limit	0.02	0.02
Training precision	BF16 mixed precision	BF16
Rollout precision	FP16	—
Master parameters / optimizer state	FP32	FP32

$\lambda = 0.005$ and $\eta_{\mathrm{g}} = 1$: terminal tracking applies to every eligible intermediate action, while emitted-token suppression applies only when it disagrees with the completed token. Its normalizer is the global sum of the scheduled tracking and suppression weights.

All students use four-token blocks with next-aligned predictions, a strictly causal completed prefix, and bidirectional attention within the active block. The decoder runs four, three, or two refinement iterations according to the student. Training samples use temperature 1, top-$p = 1$, no top-k truncation, and EOS stopping. Equation (42) specifies the 10^{-3} acceptance tolerance. Appendix B.1 specifies the masked initialization and first categorical pass.

Table 5: **Run-specific training settings.** All runs start from the corresponding Qwen3 checkpoint, use repetition stopping during training rollouts (four repeats of a pattern of at most eight tokens), and student and auxiliary prompt seeds $10086/10087$. Saved steps identify the evaluated checkpoints and equal completed updates.

Student / saved step	Thinking / training limit	Endpoint rule
1.7B B4I4 / 197	on / 4,096	first checkpoint ≥ 191
4B B4I4 / 137	on / 4,096	last checkpoint by a fixed time
1.7B B4I3 / 209	on / 4,096	first checkpoint after 188 updates
1.7B B4I2 / 503	on / 4,096	first checkpoint after plateau
1.7B B4I4 non-thinking / 146	off / 2,000	last checkpoint by a fixed time

Each run's endpoint rule was declared before its endpoint evaluation and used an update count, a wall-clock cutoff, or a plateau in monitoring accuracy (Table 5). A 2,977-response monitoring panel, evaluated every 20 updates, tracks saturation (Figure 10 shows it for the 1.7B B4I4 runs); some runs were still improving slowly at their endpoints. The B4I2 endpoint is the first checkpoint saved after its monitoring accuracy plateaued. The 4B run resumes full training state (model, optimizer, scheduler, token counters) twice, from checkpoints 32 and 94. For repetition stopping during training, the maximum pattern period is eight tokens; low-diversity tail stopping is disabled. Final evaluations disable repetition stopping (Appendix E).

Training uses BF16 mixed precision, with FP32 master parameters and optimizer states. Rollout uses FP16, and AR scoring uses BF16. Rollout attention uses FlashInfer; recorded AR and training attention resolutions use the native Triton implementation. Activation checkpointing is enabled. Packing limits are 16,384 tokens for student training and reference evaluation, 12,288 for auxiliary updates, and 16,384 for scoring.

The archived environment records Python 3.11.14, PyTorch 2.9.1, Triton 3.5.1, Transformers 5.3.0, SGLang 0.5.10.post1, FlashInfer 0.6.7.post3, and Ray 2.53.0. Training monitoring runs every 20

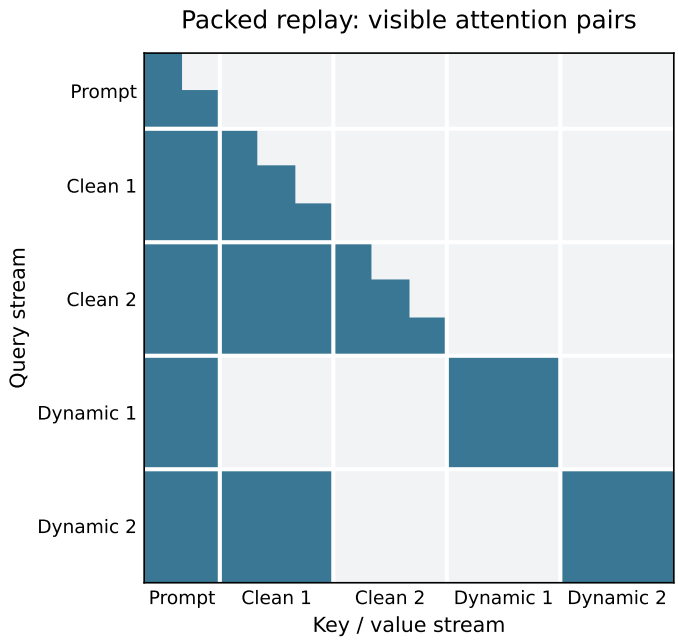

Figure 9: Logical replay attention for two response blocks under strict causal clean-prefix attention and bidirectional attention within the active block. A dynamic block sees the prompt, earlier clean blocks, and its own dynamic tokens. Its clean target and later blocks are inaccessible. Clean and dynamic copies share logical response positions while occupying different physical slots.

steps on 2,977 responses. The final paper evaluations use the separate 9,028-response protocol in Appendix E.

C Efficient Replay and Block-Diffusion Execution

Efficient training must preserve the probabilities of the actions that generated each response. Packing their contexts and rearranging adjacent derivatives lets us evaluate the trajectory loss while retaining its generation-time attention and proposal dependencies.

The engine records actions, proposal probabilities, attention context, and behavior log-probabilities. Training reconstructs those events with trainable parameters.

C.1 Packing clean context and dynamic blocks

A direct replay construction would make one example per active block, copying its completed prefix each time. Instead, at a given refinement iteration we pack a prompt, one clean response stream, and one dynamic response stream:

$$[\, c \,;\; x_1, \ldots, x_B \;;\; s_{1,t}, \ldots, s_{B,t} \,]. \tag{53}$$

Ignoring block-alignment padding, a prompt of length P and response of length R require approximately $P + 2R$ token slots per packed iteration. The clean stream supplies completed prefixes shared by later blocks; the dynamic stream supplies the sampled states whose action probabilities are being evaluated. Token storage is shared, while attention computation still depends on the number of allowed query–key pairs.

The mask reproduces the recorded rollout attention. For the strict-prefix configuration in Figure 9, clean queries use causal attention. A dynamic query in block b sees the prompt, clean blocks before b, and its own dynamic block bidirectionally. It cannot read its clean target, future clean tokens, or another dynamic block. Block descriptors and logical position IDs preserve these semantics under padding, packing, and sequence boundaries. The decoder's resolved logit alignment determines which output row predicts each action.

C.2 Rearranging adjacent-transition gradients

An accept/reject transition depends on current logits and a retained proposal quantity from the previous step. Simply detaching every proposal would omit one part of its derivative. We instead assign both uses of a model output to the forward that produced it.

Write $z_t = f_\theta(s_t)$, $p_t = h(z_t)$, and $q_t = g(z_t)$ for the current categorical distribution and the proposal information used by the next transition. Let $f_t(q_{t-1}, p_t)$ denote the scalar transition contribution with its fixed targets, credit, ratios, and masks. The part of the total derivative owned by z_t is obtained from

$$\nabla_{z_t} \left[f_t(\mathrm{sg}[q_{t-1}], p_t) + f_{t+1}(q_t, \mathrm{sg}[p_{t+1}]) \right]. \tag{54}$$

Summing these derivatives recovers both adjacent dependencies, with each transition counted once in the scalar loss.

Replay first obtains the detached adjacent-step quantities, then uses a trainable forward at each iteration to accumulate its current and next contributions before backward. For a two-iteration block, one extra detached forward supplies the adjacent information. For a general N-iteration block, the straightforward schedule has up to $N - 1$ such precomputations; activation checkpointing can introduce additional recomputation. This avoids holding two adjacent trainable forwards simultaneously merely to evaluate one transition. Direct categorical actions and inactive iterations follow their own recorded dependency structure.

C.3 Cache refresh and the next block

During a refinement forward, the model consumes the current block state and then samples updated tokens. Cached keys and values can consequently describe the pre-update tokens. Committing a block can also change its attention role from active bidirectional context to clean prefix. The engine refreshes the committed block under the prefix contract before later blocks consume it.

The fused-refresh path packs the pending refresh segment together with the next active block into one mixed extension. Each segment retains its logical positions, mask, cache-write locations, and output-selection map. This combines work that would otherwise require separate scheduling. Training-time evaluation preserves the recorded attention and proposal contexts.

C.4 Numerical replay and distributed normalization

The action law depends on temperature, support filtering, accept/reject thresholds, logit alignment, and arithmetic. These settings are recorded as a resolved contract. Mixed-precision serving and training paths are monitored for discrepancies between recorded and replayed log-probabilities. If a replay log-probability has additive error e, the corresponding likelihood ratio is multiplied by $\exp(e)$. For a joint block event with token log-probability errors e_i, the untruncated likelihood ratio is multiplied by $\exp(\sum_i e_i)$.

An opt-in deterministic forward path checks exact rollout/replay agreement by preserving reduction and precision-conversion order. The reported runs use the fast mixed-precision path, with mismatch handling described in Appendix B.3. Backward accumulates both terms in Eq. (54).

Before microbatch backward, the implementation computes global normalization masses for the policy objective and each active regularizer, so loss weights use the same denominator across ranks and ragged microbatches. The guard reuses target log-probabilities and emitted-token probabilities from the replay LM head, adding no model forward.

D Reveal-Order Diagnostic Protocol

The decoder visits structured, task-dependent reveal orders. In the second full block, positions $(1, 2)$ are revealed first in 39.2% of GSM8K and 52.1% of HumanEval traces; uniform ordering assigns this pair $1/(32 \cdot 31) \simeq .1008\%$. This diagnostic illustrates the state mismatch discussed in Section 3.

The diagnostic uses a Qwen3-0.6B-derived diffusion checkpoint. The decoder is the fixed-step low-confidence block decoder, with block-local bidirectional attention and the checkpoint's non-strict

clean-prefix setting. It uses 32-token blocks, 32 steps per block, temperature 1, and the tokenizer's mask-token ID. Each request generates 96 tokens with fixed-length trace collection and contributes its first two fully generated blocks. We collect two stochastic rollouts for every prompt: 2,638 GSM8K traces and 328 HumanEval traces. Generation and permutation sampling use seed 18.

For each selected block, every position must become non-mask exactly once in the reveal ordering, with one newly revealed token per step. We retain full blocks satisfying this ordering contract. For position i, let r_i denote its first non-mask iteration. The first reveal is $\arg\min_i r_i$ and the first pair is the ordered pair of positions with the two smallest ranks. Thus "first position" refers to temporal reveal order.

First-position and first-pair comparisons use Jensen–Shannon divergence of the empirical categorical distributions. The profile statistic is the mean squared difference between the datasets' mean position-wise reveal ranks. We permute dataset membership across prompt clusters 10,000 times, keeping both stochastic samples inside each prompt. The Monte Carlo p-value is $(1 + \text{exceedances})/10001$. The ordered-pair comparison in the second full block is the declared primary test of the later-state extension; the other two summaries are supporting diagnostics.

Table 6: Prompt-clustered permutation tests of dataset-dependent reveal orders in the first two complete blocks. Entries are Monte Carlo p-values.

Clustered permutation comparison	First full block	Second full block
First-position distribution	0.50005	0.00640
Ordered first-pair distribution	0.29307	0.00050
Mean reveal-rank profile	0.23098	0.00120

The permutation test compares the two datasets, whose prompt formats and contents differ. Uniform-order probabilities provide a descriptive reference. The second-block results show that the deployed reveal-state distributions differ across datasets and, as the uniform-order reference shows, from uniformly masked states.

E Evaluation Protocol and Complete Results

The quality comparison uses shared prompts, response limits, sampling settings, and grading across released baselines and our students. We report accuracy per benchmark and measure decoder-iteration efficiency separately.

Thinking is enabled for all rows in this appendix. Generation uses temperature 1, top-$p = 1$, no top-k filtering, a 4,096-token prompt cap, and the system message `You are a helpful assistant.` Training uses no added system message. Generation stops only at EOS or the response limit: no row uses repetition-based early stopping. The AR references use SGLang's direct engine interface, with the same resolved filtering settings. The main evaluations use seed 1; threshold OPDLM uses seed 1 plus its 128-request chunk index. Appendix F adds benchmark definitions, non-thinking evaluations, and the OPDLM-MATH reproduction check.

Each five-benchmark panel contains 9,028 responses: GSM8K has 1,319 questions with four responses each, MATH500 has 500 with four each, and GPQA Diamond has 198 with four each. AIME24 and AIME25 each contain 30 original questions with 16 responses per question. Scores are mean correctness (pass@1) within each benchmark. Repeated responses remain grouped by original question when estimating uncertainty.

All saved responses are graded with the same frozen protocol: `bounded_math_verify_grouped` for mathematics and OPDLM's pinned multiple-choice extractor and checker for GPQA. Parser versions are Math-Verify 0.9.0, latex2sympy2_extended 1.11.0, SymPy 1.14.0, and ANTLR runtime 4.9.3. Rendered questions, ground-truth multiplicities, and system prompts agree across configurations. The data records link checkpoint revisions, saved responses, and per-response scores; all grades are resolved.

AR references are Qwen3-1.7B and Qwen3-4B. SDAR uses the corresponding SDAR-Chat checkpoints under either a static commitment schedule or dynamic confidence selection at threshold 0.90.

OPDLM uses its released 1.7B and 4B checkpoints, labeled OPDLM (think@eval) to denote thinking enabled during evaluation. These general checkpoints were distilled without thinking; OPDLM also evaluates this inference mode as zero-shot extended thinking (Su et al., 2026). At 1.7B, we evaluate dynamic confidence selection at threshold 0.90, temperature 1, and four-token blocks. The static and threshold comparisons use the same released OPDLM weights. The additional 4B row uses OPDLM-MATH-4B-Thinking, separately distilled with thinking enabled on mathematics data. The evaluated public checkpoint revisions are listed in Table 7. These comparisons evaluate available checkpoints under the same response caps. Our B4I4 runs share general OPDLM's Qwen3 initialization, teacher checkpoints, and prompt corpus (Appendix B.4). We train with thinking enabled using our decoder and objective; SDAR and OPDLM retain their released checkpoints and training recipes. OPDLM-MATH-Thinking uses mathematics-specific training and a final maximum rollout length of 8,000 tokens (Su et al., 2026).

Table 7: **Public AR reference and diffusion baseline revisions.** SDAR evaluations use the released weights with the corresponding Qwen3-compatible configuration and tokenizer view.

Checkpoint	Revision
Qwen/Qwen3-1.7B	`70d244cc86ccca08cf5af4e1e306ecf908b1ad5e`
Qwen/Qwen3-4B	`1cfa9a7208912126459214e8b04321603b3df60c`
JetLM/SDAR-1.7B-Chat	`14fcc4b796f05b9a11aa565b2353199af757d079`
JetLM/SDAR-4B-Chat	`56bca3c8d927410589a86e7996ee281b6764fa37`
divelab/OPDLM-1.7B	`21eb6273f83f0dd73c5134b99cb6a6c4dd5d5a15`
divelab/OPDLM-4B	`5e5f00ec75543fa1aa293e22e05d7e9de2c03e0b`
divelab/OPDLM-MATH-4B-Thinking	`55aa2404e804dc67d4a40515c42d9917423ed7fe`

For our models, BbIi denotes b token positions per block and i parallel refinement iterations before committing each block. Thus B4I4 uses four-token blocks with four iterations; B4I3 and B4I2 use three and two iterations. The evaluated checkpoints are 1.7B B4I4 at step 197, 4B B4I4 at step 137, 1.7B B4I3 at step 209, and 1.7B B4I2 at step 503, all initialized from the AR model and trained with a 4,096-token response limit (Table 5). Both evaluation caps use the same checkpoint for each setting. The 4B student is distilled on the general OPDLM mixture of mathematics, code, science, and chat prompts. Its 8K evaluation tests generation beyond the 4,096-token training limit, with fixed weights and benchmark inputs. Each iteration count uses a separately trained checkpoint. A 2,977-response subset of the five benchmarks was evaluated every 20 updates during training for monitoring; endpoints follow the predeclared rules in Appendix B.5.

TPF is computed from the same full quality panels as the scores, with standard errors from question-clustered resampling; all reported values have standard errors below 0.01 except threshold decoders (Table 2). All quality generation uses BF16 inference. AR generation uses a causal one-token decoder. Our students use next-aligned predictions with a strictly causal completed prefix; SDAR and OPDLM use current-aligned predictions with their non-strict completed-prefix attention.

TPF is the ratio of total retained response tokens to total decoder iterations over a panel. Prompt tokens and padding are excluded; a genuine terminal token is included. Tokens beyond the response cap are excluded, while iterations already spent remain in the denominator. Prefill and re-prefill are excluded, and a cache refresh overlapped with the next block is not counted as an additional iteration. TPF measures sequential decoder-iteration efficiency. The work within each forward and cache refresh differs across decoders, so Appendix H reports wall-clock time separately.

We group responses by exact rendered question, so repeated AIME questions form one group. Standard errors and paired differences resample question clusters with 10,000 bootstrap draws.

F Additional Benchmarks and Training Recipes

The additional benchmarks test breadth beyond mathematics and science; the non-thinking training recipe and the OPDLM-MATH reproduction check examine other training and evaluation modes.

Table 1 includes three capability benchmarks beyond mathematics and science: MMLU (Hendrycks et al., 2021), IFEval (Zhou et al., 2023), and HumanEval (Chen et al., 2021). We use a shared

2,000-question MMLU subset with one response per question, all 541 IFEval prompts with four responses each, and all 164 HumanEval problems with four responses each. IFEval measures strict prompt-level instruction following; HumanEval reports pass@1 on the base tests. Both score the final answer when a response contains a separate thinking segment. The other five benchmarks retain the question sets and grading protocol of Appendix E. These three benchmarks were not part of the training-time monitoring panel.

Non-thinking training. We also vary the training recipe by disabling thinking for both student and teacher and using a 2,000-token training response limit. This 1.7B B4I4 student starts from Qwen3 with the same distillation framework, credit rule, and prompt corpus; its evaluated checkpoint is step 146. Table 8 evaluates it with thinking disabled at a 2,000-token response limit. It exceeds general OPDLM, which was also distilled without thinking, on all eight benchmarks, for example 61.75 versus 47.60 on MATH500 and 55.13 versus 39.70 on IFEval. It also exceeds SDAR on all benchmarks except HumanEval (36.89 versus 53.20). Evaluated with thinking at the 4,096-token limit, the same student exceeds both SDAR and general OPDLM on all eight benchmarks, for example 73.65 versus 48.00 for OPDLM on MATH500 and 62.65 versus 26.98 on HumanEval.

Table 8: **Non-thinking training at 1.7B.** The student is distilled with thinking disabled for 146 updates and evaluated with thinking disabled (2,000-token limit) and enabled (4,096-token limit); baseline rows under thinking match Table 1. Bold marks the best diffusion score. No row uses repetition-based early stopping.

Model	AIME24	AIME25	GSM8K	MATH500	GPQA	MMLU	IFEval	HumanEval
Non-thinking evaluation; 2,000 response tokens								
Qwen3 (AR reference)	7.08	7.08	80.86	70.15	27.90	59.05	68.99	66.01
SDAR-Chat	0.21	0.42	31.63	19.85	6.31	12.85	25.92	**53.20**
OPDLM	2.50	0.21	62.85	47.60	25.38	48.65	39.70	28.20
DM-OPD (Ours, non-thinking-trained)	**5.42**	**4.38**	**73.50**	**61.75**	**28.41**	**56.90**	**55.13**	36.89
Thinking evaluation; 4,096 response tokens								
Qwen3 (AR reference)	16.67	19.58	89.27	76.75	29.80	69.30	70.66	76.07
SDAR-Chat	1.67	1.04	76.04	53.00	25.63	49.05	37.94	48.17
OPDLM	5.42	2.50	61.37	48.00	26.39	46.80	29.39	26.98
DM-OPD (Ours, non-thinking-trained)	**10.83**	**15.83**	**87.62**	**73.65**	**27.90**	**67.05**	**51.25**	**62.65**

SDAR's low non-thinking scores frequently reflect immediate termination: at the 2,000-token limit, 64.75% of its 1.7B MMLU responses and 72.73% of its GPQA responses contain only an end token. The released model shows the same behavior outside our harness. The released SDAR-Chat template with thinking disabled reproduces all 13,848 of our non-thinking prompts token for token, and the configured stop tokens match the released generation settings. On a 384-response sample, SDAR's released inference engine stops after one token on 62.5% of responses, versus 57.0% in our harness; only a thinking-style prompt removes the behavior (0.0%). The early termination is therefore a property of the released model under its non-thinking template, and we keep these evaluations.

F.1 OPDLM-MATH Evaluation

The specialist comparison uses a shared evaluation protocol, supplemented by a check of OPDLM-MATH's released inference recipe.

We evaluate the released OPDLM-MATH-4B-Thinking checkpoint using its thinking-enabled user-only template and mathematics prompt suffix, greedy sampling (temperature 0, top-$k = 1$), an 8,000-token response limit from the released mathematics-evaluation script,[1] and static B4I4 decoding. Model weights and the output head use BF16, and repetition stopping is disabled. The released temperature-zero sampler randomly resolves tied maximum logits. Restricting the reproduction panel below to native-code responses, the eight draw slots that cover all 30 questions form a balanced native-only subset of 240 responses, selected by question coverage without consulting scores; it scores 30.00% (72/240), with 191 responses reaching the response limit. This subset is drawn from the same panel. The full panel, with 16 executions per question, combines 296 saved native-code responses with 184 from our decoder implementation and scores 31.04%, compared with the reported 43.3%

[1] OPDLM mathematics-evaluation script, commit `3bb8bf60`, lines 46–49.

(Su et al., 2026). The common grader and OPDLM's released grader agree on all 480 responses. In this evaluation, 385 responses reach the response limit. Table 1 uses the shared temperature-1 protocol for all models, under which OPDLM-MATH scores 30.63% on AIME24 at the 8,192-token limit.

G Ablations: Where the Gains Come From

This appendix gives the controlled comparisons summarized in Section 5.2. All runs start from Qwen3-1.7B and share the B4I4 decoder, prompt corpus, student and auxiliary prompt seeds (10086/10087), optimizer settings, response-length schedule, and the 2,977-response monitoring panel (4K limit, thinking). Each arm is a single training run. The replay embedding path used by C differs from that used by the other runs. Comparison steps were fixed before the compared scores were inspected: B, C, and D at 180 completed updates; C and F at 160; C and E at 100. Differences are paired over questions, with 95% intervals from 10,000 bootstrap draws.

Runs. C is DM-OPD as specified. D removes the guard. B replaces completed-text feedback with OPDLM's objective: a full-vocabulary forward KL from the teacher's next-token distribution at the completed causal prefix to the student's prediction, at every valid position and executed refinement step of the student's own trajectories. B uses OPDLM's objective with our decoder and training pipeline and the teacher alone as supervision. F keeps the guard and the auxiliary's training but uses teacher-only feedback, $d_i = \log q_\phi$.

Objective (D−B). With the guard off in both runs, distribution matching exceeds OPDLM's objective with our decoder by 5.44 points [3.84, 7.08] after 180 updates, including 11.20 points on MATH500 and 6.67 on AIME25 (Table 9). B's prediction loss falls from 7.82 to below 1 nat per token; its accuracy drops from 50.69 to 43.10 by update 80 and ends at 50.08, below its initialization, while D ends at 55.53 (Figure 10). In this run, lower prediction loss accompanies an early accuracy drop and an endpoint score below initialization.

Student density (C−F). Teacher-only feedback lowers accuracy by 5.64 points [4.09, 7.26] after 160 updates. How closely the auxiliary tracks the student's text distribution during training was not measured directly.

Guard (C−D). Without the guard, accuracy drops more deeply early in training (32.25 versus 49.75 after 40 updates) and recovers by about 100 updates; after 180 updates the two runs are within noise (-1.11 [-2.33, 0.14]). We therefore treat the guard as an optional stabilizer.

Table 9: **Ablations of the objective and its components (1.7B B4I4).** Monitoring panel: 2,977 responses, 4K limit, thinking. Rows share initialization, seeds, decoder, and response schedule; step counts completed updates and was fixed in advance. Bold marks 95% bootstrap intervals that exclude zero.

Run / contrast	Step	Overall	AIME24	AIME25	GSM8K	MATH500	GPQA
Monitoring accuracy (%)							
C: DM-OPD (distribution matching, guard)	180	54.42	8.96	12.92	85.90	65.80	26.77
D: C without guard	180	55.53	10.00	14.17	86.81	67.00	28.79
B: OPDLM objective, our decoder	180	50.08	6.88	7.50	82.26	55.80	29.29
F: C with teacher-only feedback	160	49.48	8.75	13.75	77.71	62.40	14.14
Paired difference (points) [95% CI for overall]							
D−B: objective	180	**+5.44 [+3.8, +7.1]**	+3.12	**+6.67**	**+4.55**	**+11.20**	-0.51
C−D: guard	180	-1.11 [-2.3, +0.1]	-1.04	-1.25	-0.91	-1.20	-2.02
C−F: student-density term	160	**+5.64 [+4.1, +7.3]**	+2.92	+1.25	**+8.57**	+2.00	**+12.63**

Endpoints. The comparisons above use matched update counts. Table 10 complements them with the full-protocol evaluation at the reported final checkpoints; the training curves appear in Figure 10.

Guard with OPDLM's objective (E). Run E adds the guard to B. It has no measurable effect there (E−B: -0.34 points [-1.70, 1.09] after 100 updates), and distribution matching with the guard (C) exceeds it by 8.30 points (Table 11). The objective contrast therefore holds with the guard on or off.

Full evaluation of the guard ablation. Table 12 evaluates the guard-free student (D, 191 updates) on the full eight-benchmark protocol of Table 1. Its scores are close to those of our main student (C,

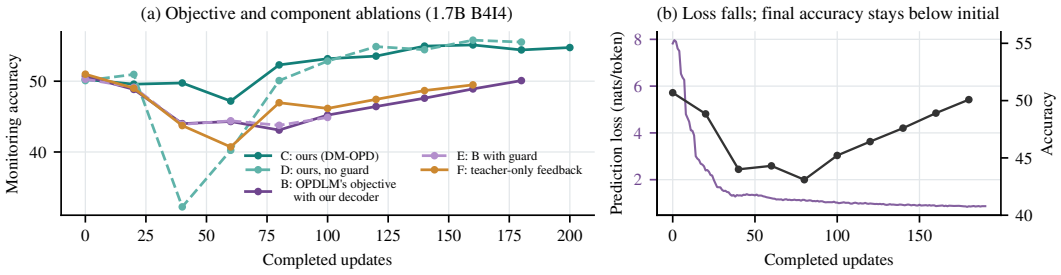

Figure 10: **Training curves of the ablation runs.** (a) Monitoring accuracy. (b) OPDLM's objective with our decoder (B) reduces its loss roughly ninefold, yet its accuracy first falls and ends below its initialization.

Table 10: **Ablation runs at their final checkpoints (1.7B B4I4, 4K limit, thinking, no repetition-based early stopping).** Full evaluation protocol of Table 1; accuracy in percent; updates are completed updates at the evaluated checkpoint.

Run	Updates	AIME24	AIME25	GSM8K	MATH500	GPQA	MMLU	IFEval	HumanEval
C: ours (guard)	197	10.83	16.04	87.83	73.35	28.03	68.15	62.85	64.79
D: ours, no guard	191	14.58	16.46	87.45	74.15	27.53	69.40	59.98	64.48
B: OPDLM objective, our decoder	191	7.92	9.58	84.44	65.60	29.67	63.45	42.79	34.91
E: B with guard	100	5.62	7.92	84.80	64.25	25.51	61.95	41.08	35.21
F: teacher-only feedback	161	10.21	13.12	78.73	70.10	21.97	64.65	35.26	60.21

Table 11: **Additional ablations (1.7B B4I4, monitoring panel).** Accuracy in percent and paired differences in points with a 95% bootstrap interval for the overall score; bold marks intervals excluding zero.

Run / contrast	Step	Overall	AIME24	AIME25	GSM8K	MATH500	GPQA
E: B with guard	100	44.88	3.12	4.17	76.80	48.40	23.23
E−B: guard with OPDLM's objective	100	-0.34 [-1.7, +1.1]	+0.21	-0.21	-0.76	+0.60	-1.52
C−E: objective with guard	100	**+8.30 [+6.2, +10.4]**	**+6.04**	**+11.04**	**+6.14**	**+16.40**	+1.01

197 updates): every difference is below 1.6 times its combined (unpaired) standard error, and neither run is higher on all benchmarks. The main-table row is the method as specified, with the guard; this choice was fixed before D's full evaluation.

Table 12: **Guard ablation on the full protocol (1.7B B4I4)**, with our objective (C, D) and with OPDLM's objective (E, B). Numbers after run names are completed updates; parentheses give question-clustered standard errors. No repetition-based early stopping.

Limit	Run	AIME24	AIME25	GSM8K	MATH500	GPQA	MMLU	IFEval	HumanEval
4K	C (guard), 197	10.83 (3.96)	16.04 (6.07)	87.83 (0.73)	73.35 (1.70)	28.03 (2.37)	68.15 (1.04)	62.85 (1.57)	64.79 (2.78)
4K	D (no guard), 191	14.58 (5.01)	16.46 (5.78)	87.45 (0.76)	74.15 (1.71)	27.53 (2.49)	69.40 (1.03)	59.98 (1.59)	64.48 (2.60)
4K	B (OPDLM objective, no guard), 191	7.92 (3.23)	9.58 (3.52)	84.44 (0.78)	65.60 (1.76)	29.67 (2.22)	63.45 (1.08)	42.79 (1.54)	34.91 (2.37)
4K	E (B with guard), 100	5.62 (2.69)	7.92 (3.77)	84.80 (0.75)	64.25 (1.83)	25.51 (2.00)	61.95 (1.09)	41.08 (1.50)	35.21 (2.52)
8K	C (guard), 197	23.12 (5.88)	21.46 (6.41)	88.21 (0.74)	83.15 (1.37)	34.09 (2.53)	69.35 (1.03)	64.79 (1.53)	69.66 (2.45)
8K	D (no guard), 191	25.21 (5.97)	24.17 (6.83)	88.06 (0.75)	83.10 (1.42)	32.83 (2.45)	70.40 (1.02)	61.41 (1.62)	66.46 (2.50)
8K	B (OPDLM objective, no guard), 191	16.67 (4.95)	18.54 (5.85)	84.67 (0.75)	75.00 (1.57)	33.33 (2.50)	65.85 (1.06)	44.32 (1.57)	39.94 (2.42)
8K	E (B with guard), 100	13.33 (4.49)	13.96 (5.01)	85.42 (0.74)	74.05 (1.61)	30.05 (2.49)	63.95 (1.07)	41.40 (1.55)	37.80 (2.47)

H Wall-Clock Measurements

TPF measures retained tokens per decoder iteration. The work per iteration differs across decoders and implementations, so we time generation on single RTX 5090 GPUs with a fixed workload: exactly 1,024 generated tokens per request (EOS and repetition stopping disabled), BF16 weights with an FP32 output head, temperature 1, prompts from MATH500. Latency uses one request; throughput submits 64 requests at once. Each row reports the median of three timed repetitions after one warmup, including prompt processing, refinement, and cache refresh. Measurements were taken on two nodes; relative values are computed against an AR reference timed on the same node, and cross-node comparisons are not meaningful (the AR throughput differs by a factor of 1.39 between nodes). The released baselines emit stop tokens before the cap but are decoded to the full 1,024 tokens, so all rows perform the same amount of generation.

Table 13: **Wall-clock cost of generating 1,024 tokens on one RTX 5090 (1.7B models).** Relative values use the same-node AR reference. For our 1.7B decoder, fewer refinement iterations reduce measured latency on this workload.

Node	Model / decoder	Latency (s)	Rel. latency	Throughput (tok/s)	Rel. throughput
1	Qwen3 (AR)	2.886	1.00	3795	1.00
1	Ours, B4I4	5.770	2.00	3900	1.03
1	Ours, B4I3	4.405	1.53	4873	1.28
1	Ours, B4I2	3.015	1.05	6046	1.59
2	Qwen3 (AR)	2.757	1.00	5280	1.00
2	SDAR-Chat, static B4I4	6.231	2.26	2731	0.52
2	SDAR-Chat, B4 τ=0.90	3.214	1.17	5552	1.05
2	OPDLM, static B4I4	5.575	2.02	2772	0.52
2	OPDLM, B4 τ=0.90	5.446	1.98	3944	0.75

At batch size one, every diffusion decoder in this implementation has higher latency than AR generation; the two-iteration decoder is closest (1.05 times). Under batched generation, our two- and three-iteration decoders exceed the AR throughput on the same node by factors of 1.59 and 1.28. These measurements are specific to our serving implementation and hardware; they illustrate that fewer refinement iterations translate into throughput.

I Limitations

Feedback accuracy depends on the auxiliary density estimate, and fitting that estimate adds post-training cost.

Auxiliary density errors can bias updates (Appendix A.7). Improving the density estimate and reducing the cost of auxiliary training and trajectory evaluation remain open directions.