ALPHA DIFFUSION LANGUAGE MODELS: FACTORIZATION ALONE IS NOT THE PROBLEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion language models can generate multiple tokens in parallel, but reducing the number of denoising steps can lead to inconsistent predictions. Standard cross-entropy training fits conditional token marginals, whereas parallel generation requires consistent joint predictions. We introduce *Alpha Diffusion Language Models (AlphaDLM)*, trained with a sequence-level alpha loss that recovers cross-entropy in the limit of vanishing alpha and has a joint-mode optimum at alpha one. Our analysis characterizes how the objective and factorization jointly determine the fitted distribution. We identify conditions under which intermediate alpha preserves multiple valid completions while excluding invalid token combinations. Trained on TinyGSM, AlphaDLM achieves 34.6% accuracy on GSM8K with only four model evaluations. We further scale the method to the larger SDAR model and evaluate it on code and mathematics benchmarks. These results show that changing the training objective can improve the accuracy–computation trade-off of factorized diffusion language models.

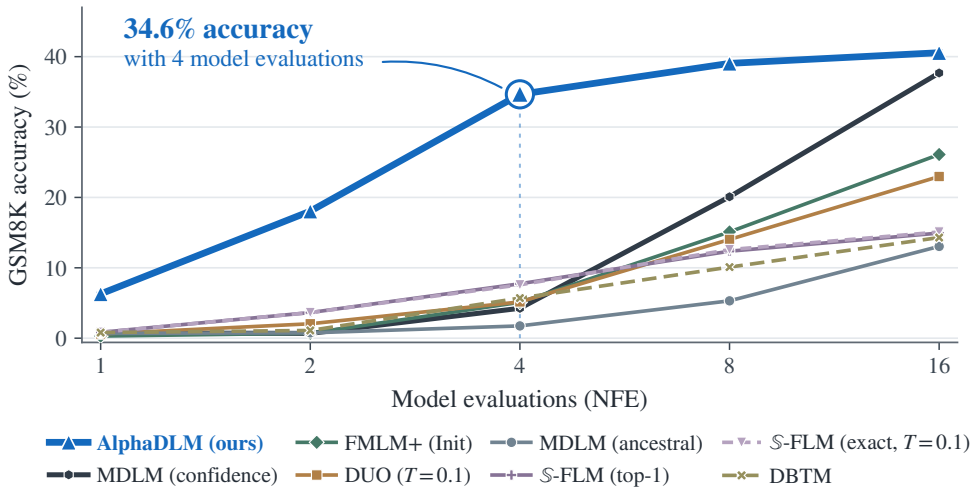


Figure 1: **GSM8K accuracy with 1–16 model evaluations.** A single AlphaDLM checkpoint reaches 34.6% in four evaluations. AlphaDLM and MDLM (confidence) use confidence-based unmasking with the token-reveal schedule of FMLM+. Other methods retain their respective samplers. Sources and protocols: Table 1 and Appendix B.

1 INTRODUCTION

Discrete diffusion language models generate text by reconstructing corrupted sequences (Sahoo et al., 2024; Nie et al., 2025). A single evaluation predicts distributions at many positions, allowing several tokens to be committed in parallel. Independently predicted tokens can each fit the visible context yet form an invalid sequence when generated together (Wu et al., 2026).

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.

Under cross-entropy training, the optimal factorized denoiser fits the conditional distribution of each token separately. When several completions are possible, independently combining these predictions can produce invalid sequences (Figure 2). One approach is to model token dependencies explicitly (Liu et al., 2025; Xu et al., 2025). However, a factorized distribution can already represent any single valid completion by assigning probability one to its tokens. Factorization therefore does not by itself prevent consistent parallel predictions. This suggests changing the training objective to favor token distributions that produce valid completions when combined.

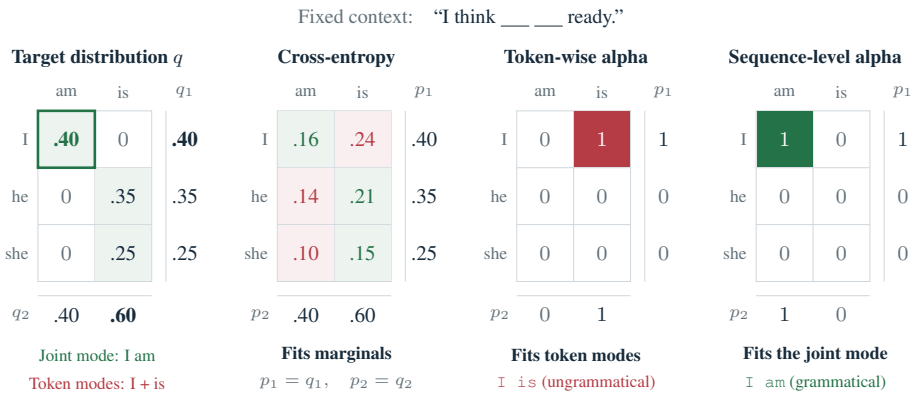


Figure 2: **Marginal fitting, token-mode fitting, and joint-mode fitting.** Cross-entropy assigns probability 0.48 to ungrammatical completions. At $\alpha = 1$, token-wise alpha loss predicts `I is`, while sequence-level alpha loss fits the joint mode `I am`. Each fitted distribution minimizes its population loss within the same factorized family. Green and red mark grammatical and ungrammatical outcomes with positive probability.

We introduce *Alpha Diffusion Language Models (AlphaDLM)*, trained with sequence-level alpha loss, a form of generalized cross-entropy (Zhang & Sabuncu, 2018). This loss includes cross-entropy as the limit $\alpha \to 0$, while at $\alpha = 1$, predicting a joint mode minimizes the expected loss. These endpoints suggest a way to move from fitting token marginals toward fitting complete sequences. We analyze what happens between them when the denoiser is factorized. The optimum can preserve several valid completions while excluding invalid combinations, rather than selecting only one answer. We establish sufficient conditions for this behavior and bound the probability of generating a valid completion when the loss is close to its minimum.

This change in the training objective is intended to improve generation when many tokens must be predicted in parallel within a small number of model evaluations. Recent approaches pursue this goal through trajectory supervision or self-distillation (Zhang et al., 2026; Chen et al., 2026; Kim et al., 2026). AlphaDLM instead trains directly on corrupted data, without teacher-generated trajectories, and retains the factorized denoiser. We evaluate whether this change improves accuracy at a fixed generation budget, using TinyGSM for comparisons between training objectives and SDAR-1.7B for experiments on code and mathematics. Appendix F discusses related work.

Contributions.

1. We introduce *Alpha Diffusion Language Models*, which train a factorized denoiser using sequence-level alpha loss. We characterize the joint distribution whose marginals minimize this objective and explain how it differs from applying alpha loss to individual tokens (Sections 3.1–3.3).
2. We establish conditions under which the optimal factorized denoiser trained with sequence-level alpha loss assigns probability only to valid completions while retaining several possible answers. We also bound the probability of generating a valid completion when the expected loss is close to its minimum (Section 3.2, Appendix A.1).
3. We empirically demonstrate that our method achieves 34.6% GSM8K accuracy with four model evaluations after training on TinyGSM, and that sequence-level training achieves

higher accuracy than token-wise alpha loss at matched evaluation budgets (Section 4.1). We also demonstrate improved accuracy–TPF trade-offs with SDAR-1.7B on code and mathematics benchmarks (Section 4.2).

2 BACKGROUND

To understand how the training objective affects parallel generation, we first review how discrete diffusion language models are trained and decoded (Section 2.1). We then turn to generalized cross-entropy, a family of losses that includes standard cross-entropy as a limiting case (Section 2.2).

2.1 DISCRETE DIFFUSION LANGUAGE MODELS

Discrete diffusion models learn to reverse a corruption process (Austin et al., 2021a; Lou et al., 2024). A denoiser predicts clean tokens from corrupted inputs, and a sampler uses these predictions to reconstruct the sequence.

Corruption process. Let $x \in \mathcal{V}^L$ be a clean sequence conditioned on a prompt or preceding blocks c, and let $z \sim q_t(\cdot \mid x)$ be its corruption at time $t \in [0, 1]$. Discrete corruption replaces tokens with vocabulary symbols or a mask. Masked diffusion independently retains each token or replaces it with $\mathtt{m} \notin \mathcal{V}$:

$$q_t(z_i \mid x_i) = \bar{\alpha}_t \mathbf{1}\{z_i = x_i\} + (1 - \bar{\alpha}_t)\mathbf{1}\{z_i = \mathtt{m}\}, \qquad q_t(z \mid x) = \prod_{i=1}^{L} q_t(z_i \mid x_i). \tag{1}$$

The retention probability decreases from $\bar{\alpha}_0 = 1$ to $\bar{\alpha}_1 = 0$. Despite independent corruption, the clean tokens can remain dependent conditional on z.

Factorized denoiser. Given the visible context, the denoiser predicts one categorical distribution per masked position $i \in M = \{i : z_i = \mathtt{m}\}$:

$$p_\theta(x_M \mid z, t, c) = \prod_{i \in M} p_{\theta,i}(x_i \mid z, t, c). \tag{2}$$

Factorization holds within each evaluation. Later predictions depend on previously revealed tokens, so the generated sequence can contain dependencies despite the factorized denoiser.

Training objective. The denoiser learns from corrupted inputs using weighted token cross-entropy (Sahoo et al., 2024; Shi et al., 2024):

$$\mathcal{L}_{\mathrm{CE}}(\theta) = \mathbb{E}_{x,c,t,z}\left[-w(t) \sum_{i \in M} \log p_{\theta,i}(x_i \mid z, t, c)\right], \tag{3}$$

The expectation is over data, training times, and corruptions $z \sim q_t(\cdot \mid x)$. With uniform times and $\bar{\alpha}_t = 1 - t$, the continuous-time masked diffusion bound uses $w(t) = 1/t$.

For fixed (z, t, c) and unrestricted token predictors, the population cross-entropy is minimized by the conditional token marginals, $p_i^*(v \mid z, t, c) = p_{\mathrm{data}}(x_i = v \mid z, t, c)$. Their product can differ from the joint posterior over the missing tokens. In particular, independently selecting their modes can produce invalid token combinations.

Parallel decoding. At generation time, the trained denoiser predicts tokens, and the sampler chooses positions to reveal. Fixed-budget decoding prescribes the number of evaluations. Adaptive confidence decoding (Wu et al., 2026) reveals predictions above a confidence threshold, with a fallback ensuring progress, so its evaluation count varies.

Blockwise diffusion. The same denoising procedure can operate within blocks generated autoregressively (Arriola et al., 2025; Cheng et al., 2025). The full sampling distribution is

$$p_\theta^{\mathrm{gen}}(x \mid c) = \prod_{b=1}^{B} p_\theta^{\mathrm{gen}}(x^{(b)} \mid x^{(<b)}, c). \tag{4}$$

Each block uses iterative calls to p_θ with the preceding context cached.

2.2 GENERALIZED CROSS-ENTROPY

Cross-entropy is the standard loss for categorical prediction. For a target class y, it takes the form $\ell_{\mathrm{CE}}(p, y) = -\log p(y)$. It is a limiting case of generalized cross-entropy, also known as alpha loss (Zhang & Sabuncu, 2018):

$$\ell_\alpha(p, y) = \frac{1 - p(y)^\alpha}{\alpha}, \qquad \lim_{\alpha \to 0} \ell_\alpha(p, y) = -\log p(y). \tag{5}$$

The parameter α controls the penalty for low target probabilities. Cross-entropy diverges as $p(y) \to 0$, whereas alpha loss is bounded by $1/\alpha$ for $\alpha > 0$. At $\alpha = 1$, it reduces to $1 - p(y)$.

3 ALPHA DIFFUSION LANGUAGE MODELS

We define the loss, analyze its optimum, and describe its use in training and sampling.

3.1 SEQUENCE-LEVEL DENOISING OBJECTIVE

We seek a training objective that improves joint predictions while retaining a factorized denoiser. Given a corrupted sequence z, the denoiser reconstructs the original tokens at the masked positions M. We denote these tokens, in their original order, by $x_M = (x_i)_{i \in M}$. The factorized denoiser assigns them the joint probability

$$p_\theta(x_M \mid z, t, c) = \prod_{i \in M} p_{\theta,i}(x_i \mid z, t, c).$$

Applying cross-entropy to this probability still gives a sum of token-level losses, because the logarithm turns the product into a sum. For positive α, generalized cross-entropy (Section 2.2) applied to this product no longer gives a sum of separate token losses. This yields the sequence-level alpha loss:

$$\ell_\alpha^{\mathrm{seq}}(\theta; x_M, z, t, c) = \frac{1 - \left[\prod_{i \in M} p_{\theta,i}(x_i \mid z, t, c)\right]^\alpha}{\alpha}. \tag{6}$$

For positive α, the loss evaluates the joint probability of the masked tokens rather than summing separate token losses. It recovers the sum of token cross-entropies as $\alpha \to 0$.

3.2 FROM MARGINAL FITTING TO JOINT MODE FITTING

Fix a corrupted sequence z, time t, and context c. Let q denote the conditional distribution of the original tokens at the masked positions:

$$q(y) = p_{\mathrm{data}}(x_M = y \mid z, t, c). \tag{7}$$

We compare the distribution minimizing the expected sequence-level alpha loss with and without factorization over the $m = |M|$ masked positions.

Without factorization. We first allow the predictor to represent any joint distribution. For $0 < \alpha < 1$, the expected loss is minimized by a lower-temperature version of q (Sypherd et al., 2022, Proposition 1):

$$p_\alpha^{*,\mathrm{joint}}(y) = q_T(y) = \frac{q(y)^{1/T}}{\sum_{y'} q(y')^{1/T}}, \qquad T = 1 - \alpha. \tag{8}$$

Here q_T is the original target q at a lower temperature $T = 1 - \alpha$, distinct from the sampling temperature. At $\alpha = 1$, any distribution supported on joint modes is optimal. For $\alpha > 1$, only point masses on joint modes are optimal. Appendix A gives the parameter correspondence and the elementary argument for $\alpha \geq 1$.

With factorization. Lowering the temperature does not remove token dependencies, so q_T generally cannot be represented by a factorized denoiser. What does the denoiser fit instead? Its optimal token probabilities are the marginals of a joint distribution that balances two goals: staying close to q_T and

reducing dependence between tokens. For $0 < \alpha < 1$, this distribution solves (Lapidoth & Pfister, 2019, Lemma 8, extended to m factors):

$$\pi^*_\alpha \in \arg\min_\pi \left\{ D_{\mathrm{KL}}(\pi \| q_T) + \frac{\alpha}{1-\alpha}\,\mathrm{TC}(\pi) \right\}, \quad p^{*,\mathrm{fact}}_\alpha = \prod_{i=1}^m \pi^*_{\alpha,i}. \tag{9}$$

The KL term measures the departure from q_T. Total correlation, $\mathrm{TC}(\pi) = D_{\mathrm{KL}}(\pi \| \prod_{i=1}^m \pi_i)$, measures dependence between tokens and is zero when they are independent. The marginals of π^*_α give the optimal token predictions. Their product is the distribution represented by the factorized denoiser. Appendix A gives the derivation, including zero probabilities.

Why sequence-level rather than token-wise? Applying alpha loss separately to each token probability gives

$$\ell^{\mathrm{tok}}_\alpha = \sum_{i \in M} \frac{1 - p_{\theta,i}(x_i \mid z, t, c)^\alpha}{\alpha}. \tag{10}$$

For $0 < \alpha < 1$, minimizing its expectation gives

$$p^{*,\mathrm{tok}}_{\alpha,i}(v) = \frac{q_i(v)^{1/T}}{\sum_u q_i(u)^{1/T}}, \qquad T = 1 - \alpha. \tag{11}$$

Here q_i is a marginal of q. Token-wise alpha loss sharpens these original marginals independently. Sequence-level alpha loss instead fits the marginals of π^*_α, which is determined by the joint distribution q. When q factorizes, the two objectives have the same optimum.

Preserving multiple valid completions. Reducing dependence does not require retaining only one answer. For example, let the first position predict either `he` or `she` and the second always predict `is`. Independent token predictions then produce only `he is` and `she is`, both valid completions. Every token combination within this group is valid. Combining their tokens with `I am` can instead produce `I is` or `he am` (Figure 3). Here, valid completions are those with positive probability under q. The following result gives conditions under which an optimal predictor assigns probability only within one such group while retaining every completion in that group.

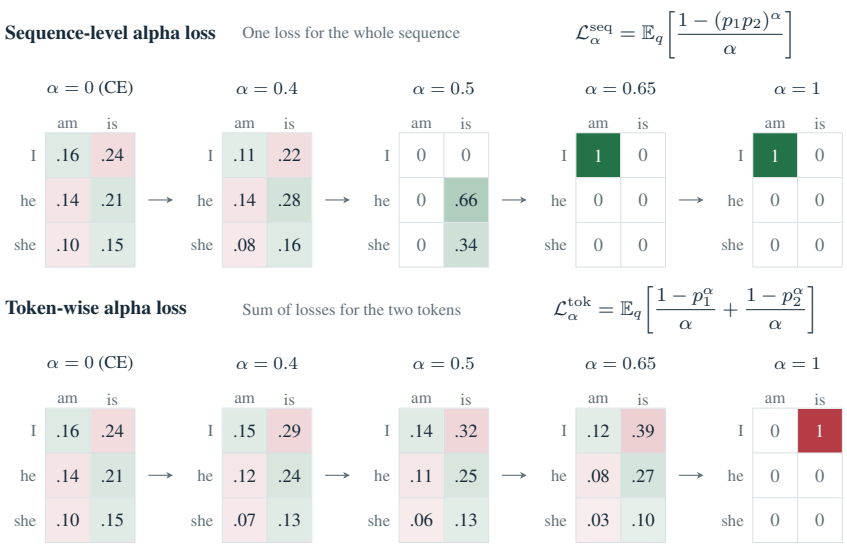


Figure 3: **Interpolating from marginal fitting to joint mode fitting.** Rows show globally optimal factorized distributions for Figure 2's target at matched α, with $\alpha = 0$ denoting cross-entropy and $p_i = p_i(x_i)$. Sequence-level fitting preserves two valid completions at $\alpha = 0.5$ before selecting the joint mode. Token-wise fitting approaches `I is`. Green marks grammatical combinations, red ungrammatical ones. Entries are rounded.

Proposition 1 (Multiple valid completions)**.** Suppose $m \geq 2$ and valid completions split into groups such that any within-group combination of tokens is valid, and different groups use disjoint token sets at every position. For $1/m < \alpha < 1$, every globally optimal factorized predictor selects one group and assigns positive probability to all completions in that group.

Selection of a single group was previously established for two tokens with one completion per group (Minka, 2005, Section 3.4 and Lapidoth & Pfister, 2019, Lemma 32). We allow multiple answers per group, with arbitrary dependencies within each group. Appendix A.1 gives the proof and an entropy-based explanation of group selection. With these groups and $1/m < \alpha \leq 1$, samples become valid with probability approaching one as the loss approaches its minimum (Proposition 2). This bound concerns sampling, not greedy accuracy away from the optimum.

In Figure 3, this behavior already occurs at the boundary $\alpha = 1/m = 0.5$. The selected distribution retains `he is` and `she is`. Since these completions differ only in the first token, this distribution factorizes and has zero total correlation, even though q_T still supports all three completions. Their probabilities are $49/74$ and $25/74$, giving lower expected loss than predicting only `I am`. Appendix A.1 verifies this boundary-case optimum separately. The fitted probability of `I am` first falls to zero and then rises to one when the optimum switches at $\alpha_\star \approx 0.616$. Token-wise fitting instead approaches `I is`. Excluding invalid combinations therefore does not require concentrating on a single completion.

3.3 Training and decoding

Training computes the loss from the denoiser's token probabilities without constructing the theoretical target π^*_α. The mask count $m = |M|$ varies across corruptions. For $m > 0$, we average log-probabilities over masked positions so that k acts on the mean token log-probability rather than a sum with a varying number of terms. We use $s_\theta = m^{-1} \sum_{i \in M} \log p_{\theta,i}(x_i \mid z, t, c)$ and a fixed coefficient k:

$$\ell_k(s_\theta) = \frac{1 - \exp(k s_\theta)}{k} = \frac{1 - p_\theta(x_M \mid z, t, c)^{\alpha_{\text{eff}}}}{m \alpha_{\text{eff}}}, \qquad \alpha_{\text{eff}} = \frac{k}{m}. \tag{12}$$

This is the loss in Equation 6 evaluated at $\alpha_{\text{eff}} = k/m$ and divided by m. At a fixed context and exponent, division by m leaves the optimum unchanged. The $k = 0$ limit recovers mean token cross-entropy. The gradient equals the mean-token CE gradient multiplied by $\exp(k s_\theta)$ (Appendix A). Large k therefore downweights examples with low mean token log-probability, motivating initialization from a pretrained denoiser. Inference retains the parallel or blockwise sampler (Section 2.1).

The connection to the analysis follows by substituting $\alpha = k/m$. Equation 9 applies for $0 < k < m$. Under the support condition of Proposition 1, its range $1/m < \alpha < 1$ becomes $1 < k < m$, where the optimum retains all completions in one group. For $k \geq m$, a joint-mode point mass is optimal (Appendix A).

4 Experiments

We compare training objectives on TinyGSM (Section 4.1) and test transfer to SDAR-1.7B on code and mathematics (Section 4.2). TinyGSM results use task accuracy versus the number of model evaluations (NFE). SDAR results use accuracy versus tokens per forward (TPF), defined as returned response tokens divided by denoising forwards plus one prompt-prefill forward per example.

4.1 TinyGSM

Setup. We train on TinyGSM (Liu et al., 2023) and evaluate Python solutions on all 1,319 GSM8K test problems (Cobbe et al., 2021). We use the TinyGSM evaluation setup of Agarwal et al. (2026). For our objective ablations, we train a shared $k = 1$ checkpoint for 200k updates, then continue training for 50k updates with each objective using the same architecture and optimizer settings. We sweep sequence-level $k \in \{1, 2, 4, 8, 16\}$ and token-wise $\alpha \in \{0.25, 0.375, 0.5, 0.75\}$, with larger k also tested for one-evaluation generation. The sequence-level exponent is $\alpha_{\text{eff}} = k/m$ as defined in Equation 12. We average ℓ_k over data and corruptions without additional time weighting. All ablations use one training seed per recipe and EMA weights. Appendix B specifies the protocols.

Accuracy with few model evaluations. Figure 1 compares one $k = 16$ checkpoint with diffusion and flow baselines. We use confidence-based unmasking with the token-reveal schedule of FMLM+ (Agarwal et al., 2026). For a budget of R model evaluations, each round reveals the highest-probability tokens at positions with confidence above 0.999, where confidence is the maximum predicted token probability. If necessary, it reveals additional positions in descending confidence order to distribute the remaining masked positions across the remaining rounds. It executes exactly R forwards, even if all masks are filled earlier.

Table 1: **GSM8K accuracy (%) ↑ at fixed NFE.** Recomputed MDLM, DUO, and $\mathbb{S}$-FLM results and published FMLM+, DBTM, and IDLM values. Protocols appear in Appendix B.

Method / NFE	4	8	16	32	64
MDLM (ancestral)	1.7	5.3	13.0	22.1	28.2
DUO ($T = 0.1$)	5.2	14.0	23.0	30.8	31.2
$\mathbb{S}$-FLM (exact, $T = 0.1$)	7.6	12.6	15.1	16.9	16.9
$\mathbb{S}$-FLM (top-1)	7.7	12.4	14.9	16.5	17.4
FMLM+	2.9	8.7	13.4	19.0	19.1
FMLM+ (Distill)	3.9	10.3	18.7	21.6	23.4
FMLM+ (Init)	5.1	15.1	26.1	31.8	33.6
DBTM	5.7	10.1	14.3	16.8	16.2
IDLM (MDLM)	–	–	–	12.8	14.9
IDLM (DUO)	–	–	–	15.4	19.0
MDLM (confidence)	4.2	20.1	37.7	–	–
AlphaDLM (ours, $k = 16$)	**34.6**	**39.0**	**40.6**	**40.9**	**40.9**

AlphaDLM reaches 34.6% at four NFE, compared with 4.2% for MDLM using the same sampler and at most 7.7% for the other baselines shown. It reaches 39.0% at eight NFE and 40.6% at sixteen. At 64 NFE, Table 1 gives 40.9% for AlphaDLM and 33.6% for FMLM+ Init.

Baselines use their original samplers unless labeled MDLM (confidence), which uses the same confidence-based unmasking sampler as AlphaDLM.

For a comparison under the same sampler, we also evaluate the released MDLM checkpoint with the confidence-based unmasking sampler used for AlphaDLM. It reaches 4.2%, 20.1%, and 37.7% at four, eight, and sixteen NFE. To isolate the training objective, we compare continuations from a shared $k = 1$ checkpoint at 200k updates.

Sequence-level versus token-wise training. The runs in Figure 4(a,b) share the same initialization, total update count, and sampler. At four NFE, sequence-level $k = 16$ reaches 34.65%, compared with 8.11% for the $k = 1$ continuation and 19.86% for the best token-wise setting. At sixteen NFE, $k = 8$ reaches 49.20%, compared with 44.35% for the best token-wise setting. The $k = 1$ and $k = 16$ continuations both obtain 40.56% at this budget. Thus, the preferred sequence-level parameter depends on the number of evaluations. Each plotted curve uses one checkpoint throughout. Appendix B.1 repeats these comparisons with adaptive confidence thresholds. With CE initialization, sequence-level training also outperforms the tested token-wise settings at four and sixteen NFE (Appendix D).

One-evaluation generation. With confidence threshold zero, the denoiser fills every masked position with its highest-probability token in one evaluation. Accuracy rises from 0.91% at $k = 1$ to 7.66% at $k = 32$, then decreases to 6.75% at $k = 64$ (Figure 4(c,d)). At $k = 16$, sequence-level training solves 83 of 1,319 problems (6.29%), compared with 28 (2.12%) for the strongest token-wise setting. A correct one-evaluation solution requires independently selected tokens to form a jointly correct answer. This measurement does not identify the true conditional mode. One-evaluation accuracy remains below the multi-step results.

Ancestral decoding. Figure 5 and Table 4 compare 32-step ancestral sampling using the same checkpoints, with stochastic reveal positions independent of confidence. Clean tokens are sampled at $T = 1$ or selected by exact argmax at $T = 0$. At $T = 0$, scalar logit sharpening changes neither token selection nor reveal probabilities. Increasing k from 1 to 16 raises accuracy from 8.19% to 32.52% at $T = 1$ and from 26.00% to 37.07% at $T = 0$. The gain at $T = 0$ therefore cannot be explained by scalar logit sharpening alone. The gains also persist without confidence-based position selection. Appendix E reports ancestral pass@K.

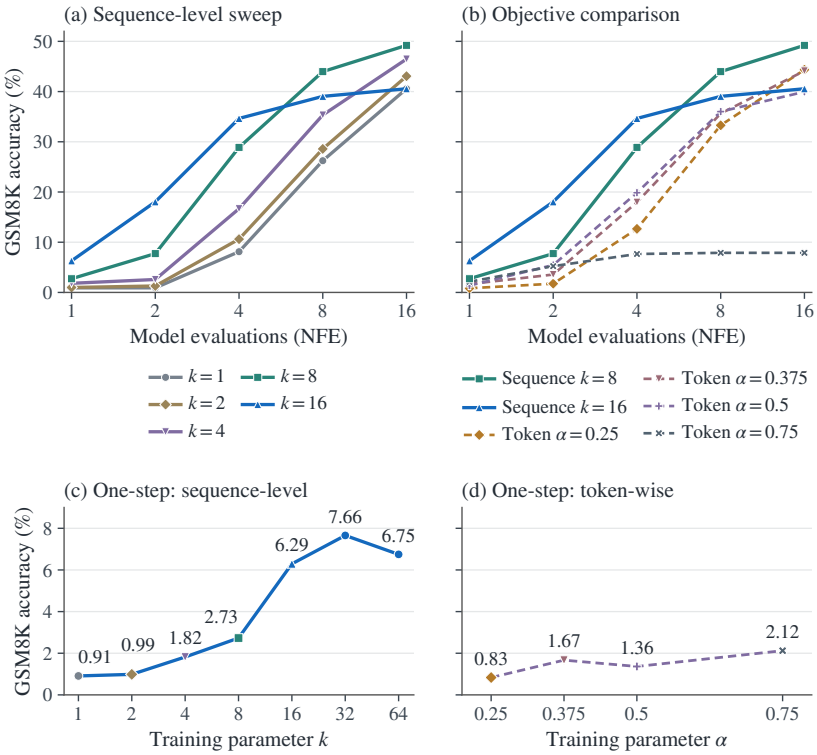


Figure 4: **How the objective changes parallel prediction.** All runs start from the same $k = 1$ checkpoint at 200k updates and continue to 250k. (a) AlphaDLM (ours) sweep and (b) token-wise comparison, using the same confidence-based sampler with exactly 1–16 forwards per example. (c,d) One-evaluation exact-argmax accuracy for sequence-level and token-wise sweeps. Values for (a,b): Table 3.

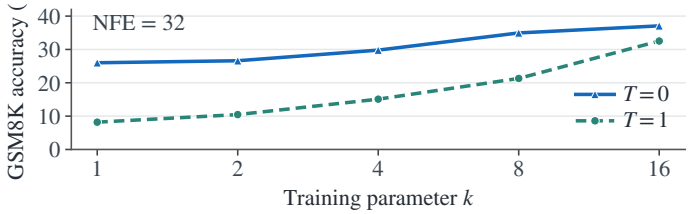


Figure 5: **Ancestral decoding at 32 NFE.** Values: Table 4.

4.2 Transfer to SDAR on mathematics and code

We fine-tune SDAR-1.7B-Chat (Cheng et al., 2025) on Nemotron-SFT-Math-v4 for mathematics and OpenCodeInstruct for code. We compare AlphaDLM with the original model, cross-entropy fine-tuning (CE), and entropy-regularized fine-tuning (CE+CAP) (Bie et al., 2025) on four benchmarks. We apply Equation 12 separately to each four-token response block. Sampling uses four-token blocks and a sweep of confidence thresholds. The main comparison uses $k = 0.8$ for mathematics and $k = 0.4$ for code. Appendix C specifies block-loss aggregation, training, and evaluation.

Mathematics. AlphaDLM improves both accuracy and TPF over CE at confidence threshold 0.99 (Figure 6(a,b)). It reaches 51.48% on MATH500 and 77.19% on GSM8K, compared with 50.60% and 76.35% for CE, while increasing TPF by 31.6% and 27.1%, respectively.

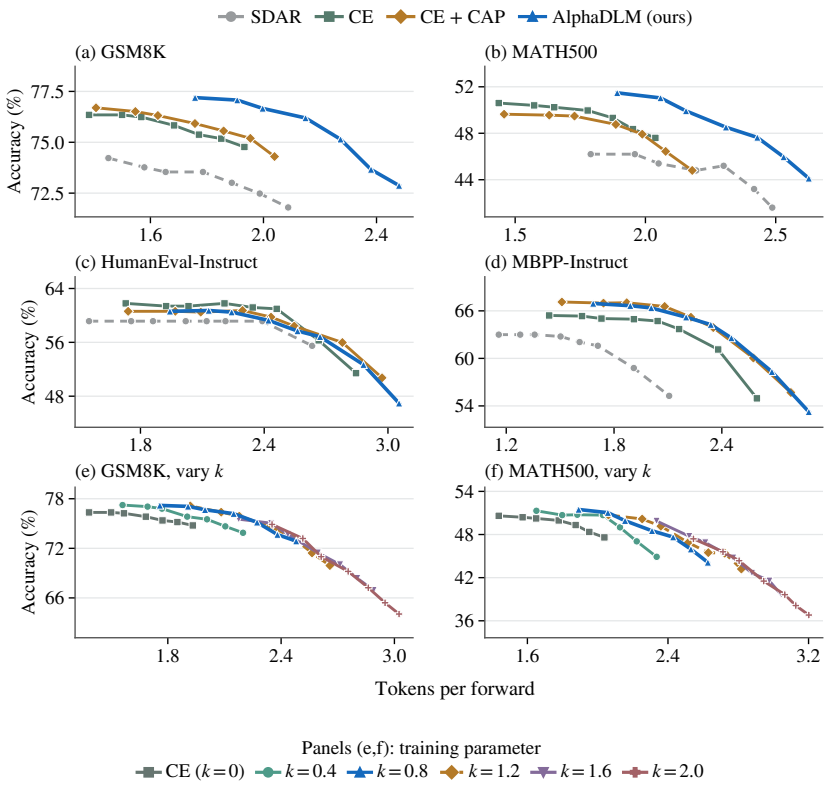


Figure 6: **Accuracy–TPF trade-offs with SDAR-1.7B.** (a–d) Objective comparisons on mathematics and code. AlphaDLM uses $k = 0.8$ for mathematics and $k = 0.4$ for code. (e,f) Mathematics parameter sweep. Points average accuracy and TPF across runs at each confidence threshold. Appendix C provides protocols and standard-deviation bands.

Code. At confidence threshold 0.99, AlphaDLM achieves higher TPF than CE+CAP with similar accuracy on both code benchmarks (Figure 6(c,d)). Both achieve 60.61% pass@1 on HumanEval-Instruct, while AlphaDLM increases TPF from 1.74 to 1.94. On MBPP-Instruct, TPF rises from 1.51 to 1.68 while pass@1 drops by 0.19 percentage points. Peak accuracy is highest for CE (HumanEval-Instruct) and CE+CAP (MBPP-Instruct).

Ablation on k values. The mathematics sweep shows how k controls the accuracy–computation trade-off (Figure 6(e,f)). Among the tested values, $k = 0.4$ and $k = 0.8$ give the highest peak accuracies. Increasing k to 1.6 or 2.0 raises TPF but reduces peak accuracy on both benchmarks.

5 Discussion and Conclusion

Parallel prediction depends on the interaction between factorization and the training objective. Cross-entropy fits the original token marginals, whereas sequence-level alpha loss changes the joint distribution whose marginals the denoiser fits. Under the conditions of Proposition 1, this change can exclude invalid combinations while preserving several valid completions. Changing the objective improves low-NFE accuracy while retaining the factorized architecture.

Our analysis describes the global optimum at a fixed corrupted context. Open questions include how these fixed-context results extend to neural denoisers trained across contexts, and how to choose the loss parameter for a given initialization and decoding budget.

AI USE STATEMENT

Generative AI assistants were used as supporting tools for literature search, implementation, formulation and checking of mathematical proofs, drafting and editing portions of the manuscript, and figure preparation. All results were verified by the authors, who take responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies training objectives for diffusion language models using existing datasets and benchmarks for mathematics and code generation. It does not involve studies with human participants. The resulting models may inherit biases from their training data and can generate incorrect or harmful text or code. Improved benchmark accuracy does not establish safety for deployment, and generated code should be treated as untrusted and evaluated in an isolated environment.

REPRODUCIBILITY STATEMENT

We provide anonymous code for training and evaluating our method on TinyGSM, together with configuration files and usage instructions. Appendix A states the assumptions and provides proofs of the theoretical results. Appendices B and C document the TinyGSM and SDAR experiments, including model configurations, training and sampling settings, data processing, evaluation protocols, and sources of baseline results. Appendix E describes the pass@K evaluation and estimator.

REFERENCES

Manan Agarwal, Sheel Shah, Chanhyuk Lee, Jaehoon Yoo, Jerry Huang, Seunghoon Hong, Aditi Raghunathan, Jinwoo Kim, and Nicholas M. Boffi. Posterior Refinement: Fast Language Generation via Any-Order Flow Maps, 2026. URL https://arxiv.org/abs/2606.24773.

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. OpenCodeInstruct: A Large-scale Instruction Tuning Dataset for Code LLMs, 2025. URL https://arxiv.org/abs/2504.04030.

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.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, volume 34, 2021a. URL https://arxiv.org/abs/2107.03006.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, 2021b. URL https://arxiv.org/abs/2108.07732.

Parikshit Bansal and Sujay Sanghavi. Enabling approximate joint sampling in diffusion lms, 2025. URL https://arxiv.org/abs/2509.22738.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling up diffusion language models to 100B, 2025. URL https://arxiv.org/abs/2512.15745.

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.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hVOcstAURb.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

Justin Deschenaux and Caglar Gulcehre. Language Modeling with Hyperspherical Flows, 2026. URL https://arxiv.org/abs/2605.11125.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1711.02281.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. In *NeurIPS Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2103.03874.

Fei Huang, Tianhua Tao, Hao Zhou, Lei Li, and Minlie Huang. On the Learning of Non-Autoregressive Transformers. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9356–9376. PMLR, 2022. URL https://proceedings.mlr.press/v162/huang22k.html.

Byoungkwon Kim and Minhyuk Sung. Tensor-train joint modeling for few-step discrete diffusion, 2026. URL https://arxiv.org/abs/2607.03788.

Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. CDLM: Consistency diffusion language models for faster sampling. In *Conference on Machine Learning and Systems*, 2026. URL https://arxiv.org/abs/2511.19269.

Amos Lapidoth and Christoph Pfister. Two Measures of Dependence. *Entropy*, 21(8):778, 2019. doi: 10.3390/e21080778. URL https://www.isiweb.ee.ethz.ch/papers/docu/alap-pfis-2019-2.pdf.

David Li, Nikita Gushchin, Dmitry Abulkhanov, Eric Moulines, Ivan Oseledets, Maxim Panov, and Alexander Korotin. IDLM: Inverse-distilled Diffusion Language Models. In *International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2602.19066.

Gaotang Li, Ruizhong Qiu, Xiusi Chen, Heng Ji, and Hanghang Tong. Beyond log likelihood: Probability-based objectives for supervised fine-tuning across the model capability continuum. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026b. URL https://gaotangli.github.io/project_page/Beyond-Log-Likelihood/static/pdfs/beyond-log-likelihood.pdf.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.01949.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. TinyGSM: Achieving $> 80\%$ on GSM8k with Small Language Models, 2023. URL https://arxiv.org/abs/2312.09241.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2310.16834.

Thomas Minka. Divergence Measures and Message Passing. Technical Report MSR-TR-2005-173, Microsoft Research, 2005. URL https://tminka.github.io/papers/message-passing/minka-divergence.pdf.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2502.09992.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.07524.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The Diffusion Duality. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52584–52619, 2025. URL https://arxiv.org/abs/2506.10892.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.04329.

Tyler Sypherd, Mario Diaz, John Kevin Cava, Gautam Dasarathy, Peter Kairouz, and Lalitha Sankar. A tunable loss function for robust classification: Calibration, landscape, and generalization. *arXiv preprint arXiv:1906.02314*, 2022. URL https://arxiv.org/abs/1906.02314v6. Version 6.

Sophia Tang and Shiyi Wang. Discrete Beckmann Transport Models for One-Step Language Modeling and Reasoning, 2026. URL https://arxiv.org/abs/2609.15903.

Tim van Erven and Peter Harremoës. Rényi Divergence and Kullback–Leibler Divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014. doi: 10.1109/TIT.2014.2320500. URL https://arxiv.org/abs/1206.2459.

Chuling Wen, Weijie Liang, and Jian Lu. Conditional Total Correlation and the Serial Depth of Adaptive Parallel Sampling, 2026. URL https://arxiv.org/abs/2608.25505.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.22618.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-based diffusion language models for text generation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.21357.

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. *arXiv preprint arXiv:2602.12262*, 2026. URL https://arxiv.org/abs/2602.12262.

Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.neurips.cc/paper_files/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html.

Yunxiao Zhao and Changxiao Cai. Adaptation to Intrinsic Dependence in Diffusion Language Models, 2026. URL https://arxiv.org/abs/2602.20126.

A POPULATION OPTIMUM OF SEQUENCE-LEVEL ALPHA LOSS

We derive the expected-loss optimum with and without factorization, then prove the guarantees for groups of valid completions and approximate loss minimization in Appendix A.1.

Fix a distribution q on the finite space of missing-token sequences and $0 < \alpha < 1$. The output distributions are free optimization variables. All logarithms below are natural. We use the variational identities of Lapidoth & Pfister (2019) to characterize the product approximation over m token positions. The derivation below makes this specialization explicit.

Equation (8) is the alpha-loss optimum in Sypherd et al. (2022, Proposition 1), under a change of parameter. Writing their parameter as β, their loss is $\beta(1-p(y)^{1-1/\beta})/(\beta-1)$. Setting $\beta = 1/(1-\alpha)$ recovers the sequence-level loss in Equation 6, and their optimum $p^*(y) \propto q(y)^\beta$ gives Equation (8). This correspondence applies for $0 < \alpha < 1$. We retain a short derivation for completeness.

Unrestricted joint predictor (Equation 8, restated). For $0 < \alpha < 1$, the unique distribution minimizing the expected sequence-level alpha loss over all joint distributions is

$$p_\alpha^{*,\text{joint}}(y) = q_T(y) = \frac{q(y)^{1/T}}{\sum_{y'} q(y')^{1/T}}, \qquad T = 1 - \alpha.$$

Derivation of Equation (8). For a candidate joint distribution p, define $F_\alpha(p) = \mathbb{E}_{y\sim q}[p(y)^\alpha]$. Minimizing expected sequence-level alpha loss is equivalent to maximizing this score. Let $Z_T = \sum_y q(y)^{1/T}$, $T = 1 - \alpha$, and $q_T(y) = q(y)^{1/T}/Z_T$. Hölder's inequality gives

$$F_\alpha(p) = Z_T^{1-\alpha} \sum_y q_T(y)^{1-\alpha} p(y)^\alpha \le Z_T^{1-\alpha}. \tag{13}$$

Equality holds precisely when $p = q_T$. Equivalently, the sum is $\exp(-\alpha D_{1-\alpha}(q_T\|p))$, which is at most one by Rényi-divergence nonnegativity (van Erven & Harremoës, 2014, Theorem 8). The $\alpha \ge 1$ cases follow from the bound given below under "Token-wise objective and scope." □

Factorized predictor (Equation 9, restated). For $0 < \alpha < 1$, every optimal factorized predictor consists of the marginals of a minimizing joint distribution below, and every such minimizing distribution gives an optimal predictor:

$$\pi_\alpha^* \in \arg\min_\pi \left\{ D_{\text{KL}}(\pi\|q_T) + \frac{\alpha}{1-\alpha}\,\text{TC}(\pi) \right\}, \quad p_\alpha^{*,\text{fact}} = \prod_{i=1}^m \pi_{\alpha,i}^*.$$

Derivation of Equation (9). The Gibbs variational identity yields

$$\log F_\alpha(p) = \max_{\pi \ll q} \{\alpha\, \mathbb{E}_\pi \log p(Y) - D_{\text{KL}}(\pi\|q)\}. \tag{14}$$

For fixed π, the product $p = \prod_i p_i$ maximizing $\mathbb{E}_\pi \log p(Y)$ has $p_i = \pi_i$ and achieves $-\sum_i H(\pi_i)$, where H is Shannon entropy. Interchanging the maximizations gives

$$\max_{p=\prod_i p_i} \log F_\alpha(p) = -\min_{\pi \ll q} \left\{ D_{\text{KL}}(\pi\|q) + \alpha \sum_i H(\pi_i) \right\}. \tag{15}$$

Using $\sum_i H(\pi_i) = H(\pi) + \text{TC}(\pi)$, the quantity being minimized can be rewritten as

$$\begin{aligned} D_{\text{KL}}(\pi\|q) + \alpha \sum_i H(\pi_i) = (1-\alpha) D_{\text{KL}}(\pi\|q_T) + \alpha\,\text{TC}(\pi) \\ - (1-\alpha)\log Z_T. \end{aligned} \tag{16}$$

The last term is constant in π. Dividing by $1 - \alpha$ proves Equation 9. Equality in the two maximizations also gives

$$\pi_\alpha^*(y) = \frac{q(y) p_\alpha^{*,\text{fact}}(y)^\alpha}{F_\alpha(p_\alpha^{*,\text{fact}})}, \qquad p_{\alpha,i}^{*,\text{fact}} = \pi_{\alpha,i}^*. \tag{17}$$

Thus the auxiliary target is shared across positions, although the fitted output distribution remains a product. Every minimizing auxiliary target gives an optimal product predictor, and every optimal product predictor arises this way. Zero probabilities are handled using extended KL divergence and the usual zero-mass conventions. In Equation 9, the minimization is over all joint distributions, with infinite KL excluding mass outside the support of q. □

Token-wise objective and scope. The token-wise risk separates across positions. Applying the unrestricted argument to each q_i proves Equation 11. When q factorizes, q_T also factorizes, attains zero in both terms of Equation 9, and gives the same optimum.

At $\alpha = 0$, the continuous extension of either loss is the sum of token cross-entropies, whose optimal factors are the marginals of q. For every $\alpha \geq 1$, $F_\alpha(p) \leq \sum_y q(y)p(y) \leq \max_y q(y)$, since $0 \leq p(y) \leq 1$. A point mass on a joint mode attains the bound and belongs to the product family. If the joint mode is unique, this point mass is the unique maximizing distribution. At $\alpha = 1$, any distribution supported on tied joint modes also attains this bound. For $\alpha > 1$, equality requires a point mass on one joint mode.

The logarithm preserves the maximizer of F_α for a fixed conditioning. It does not generally preserve an objective averaged over different conditionings with shared model parameters. To determine when the selected target yields only valid token combinations, Appendix A.1 imposes an explicit structure on the support of the data distribution.

Numerical implementation. The mask-normalized sequence-level loss has a shared gradient weight determined by the joint probability of the original tokens at all masked positions:

$$\nabla_\theta \left(\frac{\ell_\alpha^{\text{seq}}}{m} \right) = p_\theta(x_M \mid z, t, c)^\alpha \, \nabla_\theta \ell_{\text{CE}}, \qquad \ell_{\text{CE}} = -\frac{1}{m} \log p_\theta(x_M \mid z, t, c). \tag{18}$$

All missing-token predictions share this weight, whereas token-wise alpha loss assigns a separate weight to each position. Under $\alpha_{\text{eff}} = k/m$, the shared weight is $\exp(ks_\theta)$. Small target probabilities suppress this weight more strongly at large k, motivating a warm start. Averaging log-probabilities makes k comparable across mask counts, but does not remove this suppression.

The implemented loss evaluates $1 - \exp(ks_\theta)$ as $-\operatorname{expm1}(ks_\theta)$ to avoid cancellation near zero. Examples without active positions contribute zero.

A.1 Groups of valid completions at intermediate alpha

Consider the valid completions `I am`, `he is`, and `she is` in Figure 3. They split into two groups: $\{\texttt{I}\} \times \{\texttt{am}\}$ and $\{\texttt{he}, \texttt{she}\} \times \{\texttt{is}\}$. Any combination within a group is valid, but crossing groups produces an invalid completion. We now formalize this structure and show when an optimal product selects one group. The equal-variable special case appears under different parameter conventions in Minka (2005, Section 3.4) and Lapidoth & Pfister (2019, Lemma 32).

Support structure. Fix $m \geq 2$ positions and suppose

$$S = \operatorname{supp} q = \bigsqcup_{b=1}^{B} S_b, \qquad S_b = \prod_{i=1}^{m} A_{ib}, \qquad A_{ib} \cap A_{ib'} = \varnothing \quad (b \neq b'). \tag{19}$$

Thus every combination within a group has positive target probability, while tokens from different groups cannot form a valid completion. The disjointness holds at every position, so each token identifies the same group label. The distribution within a group can have arbitrary dependencies. Let $C_b(\alpha) > 0$ be the largest score attainable by a product supported on S_b:

$$C_b(\alpha) = \max_{r = \prod_i r_i \,:\, \operatorname{supp} r \subseteq S_b} \sum_{y \in S_b} q(y) r(y)^\alpha. \tag{20}$$

Proposition 1 (Multiple valid completions, restated). Suppose $m \geq 2$ and valid completions split into groups such that any within-group combination of tokens is valid, and different groups use disjoint token sets at every position. For $1/m < \alpha < 1$, every globally optimal factorized predictor selects one group and assigns positive probability to all completions in that group.

Proof of Proposition 1. **Step 1: Allocate probability across groups.** For any product p, let $t_{ib} = p_i(A_{ib})$ be the mass that token predictor i assigns to group b. Its geometric mean across positions is $v_b = (\prod_i t_{ib})^{1/m}$, so the product assigns mass v_b^m to this group. Hölder's inequality gives $\sum_b v_b \le 1$.

Step 2: Bound the score within each group. Conditioning each factor on group b bounds its score contribution by $C_b(\alpha)v_b^{m\alpha}$. Since all C_b are positive, completing the masses v_b to a probability vector cannot decrease this upper bound. Conversely, assign the same group masses $u \in \Delta_B$ to every factor and choose optimal conditional factors separately within each group. This attains the bound and gives

$$F_\alpha^* := \max_{p=\prod_i p_i} F_\alpha(p) = \max_{u\in\Delta_B} \sum_b C_b(\alpha) u_b^{m\alpha}. \tag{21}$$

Step 3: Select a group when $m\alpha > 1$. For $1/m < \alpha < 1$, the maximum is $\max_b C_b(\alpha)$. Every optimal product concentrates on one maximizing group, since $\sum_b v_b^{m\alpha} < 1$ whenever several v_b are positive or $\sum_b v_b < 1$. Within that group, fixing the other factors leaves a coordinate objective $\sum_v a_i(v)p_i(v)^\alpha$ with all $a_i(v) > 0$. Its optimum satisfies $p_i(v) \propto a_i(v)^{1/(1-\alpha)} > 0$. Thus all completions in the selected group retain positive probability. When the group is not a singleton, several outputs remain possible. □

For $0 < \alpha < 1/m$, Equation 21 instead gives $u_b \propto C_b^{1/(1-m\alpha)} > 0$ for every group. With $B \ge 2$, the resulting product produces invalid cross-group combinations. At $\alpha = 1/m$, a single-group optimum exists, and a unique best group guarantees that every optimal predictor selects it. Tied best groups permit optimal mixtures. For example, when $q(00) = q(11) = 1/2$ and $\alpha = 1/2$, independent uniform factors attain the optimum but assign half their probability to invalid pairs.

The transition also follows from Equation 15. If the auxiliary target has group probabilities u, its joint entropy contains $H(u)$ once, while its sum of marginal entropies contains $mH(u)$. After optimizing the conditional targets within groups, its objective becomes $(m\alpha - 1)H(u) - \sum_b u_b \log C_b(\alpha)$. Above $\alpha = 1/m$, uncertainty between groups is penalized, while variation within a group can remain.

Proposition 2 (Validity from approximate optimality)**.** Under the support condition in Equation 19, let $1/m < \alpha \le 1$ and $F_\alpha^* = \max_{p=\prod_i p_i} F_\alpha(p)$. Every product p satisfies the bound

$$V(p) := p(S) \ge \left(\frac{F_\alpha(p)}{F_\alpha^*}\right)^{\frac{m-1}{m\alpha-1}}. \tag{22}$$

The exponent is sharp over this class of targets.

Proof. Write $V(p) = \sum_b v_b^m$ and $\eta = (m\alpha - 1)/(m - 1)$. For $1/m < \alpha < 1$,

$$\frac{F_\alpha(p)}{F_\alpha^*} \le \sum_b v_b^{m\alpha} \le \left(\sum_b v_b\right)^{1-\eta} \left(\sum_b v_b^m\right)^{\eta} \le V(p)^\eta, \tag{23}$$

where the middle inequality is Hölder's inequality. Rearranging proves the bound. It also holds at $\alpha = 1$, since $F_1(p) \le (\max_y q(y))V(p) = F_1^* V(p)$. For sharpness, let q assign equal weights to $B \ge 2$ identical separated groups, and let every factor of p assign mass $1/B$ to each group with optimal conditional factors. Then $F_\alpha(p)/F_\alpha^* = B^{1-m\alpha}$ and $V(p) = B^{1-m}$, attaining equality. □

For the mask-normalized training loss, let $R_\alpha(p) = (1 - F_\alpha(p))/(m\alpha)$ and $R_\alpha^* = \min_{p=\prod_i p_i} R_\alpha(p) = (1 - F_\alpha^*)/(m\alpha)$. The score ratio equals $1 - m\alpha(R_\alpha(p) - R_\alpha^*)/F_\alpha^*$. Proposition 2 controls probability under product sampling, rather than deterministic argmax accuracy. At exact optimality all supported completions, including every argmax completion, are valid.

The illustration and the scope of the condition. For Figure 3, the group scores are $C_1 = 0.4$ and $C_2 = (0.35^\rho + 0.25^\rho)^{1/\rho}$, where $\rho = 1/(1 - \alpha)$. At $\alpha = 1/2$, $C_2 = \sqrt{0.185} > 0.4$, so the unique optimum retains `he is` and `she is`. Their probabilities are proportional to 0.35^2 and

0.25^2, giving $49/74$ and $25/74$. The two group scores cross at $\alpha_\star \approx 0.6162$. At that point either single-group predictor is optimal, while mixtures across them are suboptimal.

The support assumption is essential. For two positions, a coordinate optimum satisfies $p_1(v) \propto (\sum_w q(v,w)p_2(w)^\alpha)^{1/(1-\alpha)}$, and likewise for p_2. A token therefore has positive probability exactly when it has a positive-probability neighbor in the bipartite support graph of q. The support of an optimal product contains whole connected components of this graph. For $q = \left(\begin{smallmatrix} 0.4 & 0.3 \\ 0.3 & 0 \end{smallmatrix}\right)$, the graph is connected but incomplete, so every optimum gives positive probability to the absent pair at every $0 < \alpha < 1$. Equation 22 therefore relies on the Cartesian support structure, not merely on factorization or on increasing alpha. Like the preceding analysis, it concerns the population optimum over free distributions at a fixed context and gives no optimization guarantee for shared neural parameters.

B TinyGSM comparison protocols

We document the TinyGSM training and evaluation protocols and the sources of baseline results. Appendix B.1 then extends the fixed-budget comparisons to adaptive confidence sampling.

AlphaDLM. The teaser uses one EMA checkpoint at 250k updates, with $k = 16$ after $k = 1$ pre-training to 200k. Evaluation uses 512-token sequences, the SmolLM-135M tokenizer, exact argmax token selection, and untempered confidence scores. Each round reveals the union of positions above confidence 0.999 and the highest-confidence $\lceil m/r \rceil$ remaining positions, where m is the current mask count and r is the number of rounds left. Revealed tokens remain fixed. Confidence-based unmasking adapts the commitment rule of FMLM+ to an absorbing-mask denoiser with a fixed number of model evaluations.

Every example executes all R forwards, including rounds after all masks have been filled. Accuracy is the observed correct fraction over 1,319 test problems. These are single-checkpoint estimates, without repeated-training uncertainty.

Program verification. The shared GSM8K verifier extracts the first fenced code block when present, discards text before the first function definition, and attempts to remove trailing text that prevents parsing. It executes `simple_math_problem()` with a 5 s time limit and reads its numeric return value or, when unavailable, a numeric answer from its printed output. This value is compared with the number following `####` in the reference answer. Integers must match exactly, while floating-point comparisons use absolute tolerance 10^{-3}. Missing functions, parsing or execution failures, timeouts, and missing numeric answers count as incorrect.

Common-initialization objective comparisons. The $k = 1$ checkpoint at 200k updates initializes the sequence-level and token-wise continuations, which are evaluated at 250k. Table 2 specifies the shared training recipe. Sequence-level and token-wise objectives first normalize over the active positions of each example and then average over examples. Neither objective uses an additional $1/t$ weight. Examples with no active positions contribute zero.

We tokenize each TinyGSM example as a beginning-of-sequence token, the question, the literal two-character separator `\n`, the Python solution, and an end-of-sequence token. We exclude examples longer than 512 tokens and pad the remaining examples to this length without packing different examples together. The question and separator remain clean context. Response, end-of-sequence, and padding positions participate in corruption and training. A fixed 1% split is reserved for validation with split seed 42. The tokenizer is `HuggingFaceTB/SmolLM-135M`.

Training times are stratified using the implementation's antithetic sampler for a uniform distribution on $[\varepsilon, 1]$, with $\varepsilon = 10^{-3}$. The stabilized linear retention schedule is $\bar{\alpha}_t = \varepsilon + (1-\varepsilon)(1-t)$. It uses no adaptive schedule or importance sampling. The model does not receive a time-conditioning signal.

The one-evaluation token-wise control uses EMA weights, exact argmax tokens, and confidence threshold zero, revealing all missing tokens in a single forward. We verify one forward for every test example and reproduce all 1,319 archived $k = 16$ responses exactly. In the 32-evaluation ancestral comparison, $T = 0$ takes the argmax over clean-token predictions while retaining the ancestral

Table 2: **Shared TinyGSM training recipe.** Sequence-level and token-wise continuations use the same architecture, data processing, and optimizer settings.

Component	Setting
Denoiser	Bidirectional DiT with 12 blocks, 12 attention heads, hidden width 768, and conditioning width 128
Regularization	Dropout 0.1 and zero weight decay
Sequence length	512 tokens including prompt, response, and padding
Optimizer	AdamW, learning rate 3×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$
Learning-rate schedule	2,500 warmup updates followed by a constant rate
Optimization	Global batch size 512, gradient clipping 1.0, and training seed 1
Weight averaging	EMA decay 0.9999
Training duration	Common sequence-level $k = 1$ prefix through 200k updates, followed by 50k continuation updates

random choice of reveal positions. It is not an argmax over the complete reverse transition including the mask state.

Table 3: GSM8K accuracy (%) ↑ with a shared initialization and confidence-based unmasking decoder. All checkpoints use 250k total updates, including the common $k = 1$ prefix through 200k. Each value is an observed correct fraction over 1,319 examples.

Objective	1 NFE	2 NFE	4 NFE	8 NFE	16 NFE
Sequence $k = 1$	0.91	0.91	8.11	26.23	40.56
Sequence $k = 2$	0.99	1.29	10.61	28.58	43.06
Sequence $k = 4$	1.82	2.58	16.68	35.33	46.47
Sequence $k = 8$	2.73	7.73	28.89	43.97	49.20
Sequence $k = 16$	6.29	18.04	34.65	39.04	40.56
Token $\alpha = 0.25$	0.83	1.74	12.66	33.28	44.35
Token $\alpha = 0.375$	1.67	3.56	17.97	35.63	44.12
Token $\alpha = 0.5$	1.36	5.46	19.86	36.01	39.95
Token $\alpha = 0.75$	2.12	5.23	7.66	7.88	7.88

The fixed-budget objective comparison uses the teaser's configurations at 1, 2, 4, 8, and 16 forwards, changing only the checkpoint and output location. The confidence threshold is 0.999, tokens are exact argmax predictions, and all models use four GPUs with batch size 16 per GPU, EMA weights, and seed 1. An independent replay of $k = 16$ at four evaluations matches every archived per-example record. We also verify the prescribed forward count for all examples in every new evaluation. Table 3 reports all points.

MDLM with confidence-based unmasking. We evaluate the released MDLM checkpoint from the $\mathbb{S}$-FLM repository on the same 1,319 problems at $R \in \{1, 2, 4, 8, 16\}$. We use the same confidence-based unmasking implementation and evaluation configurations as AlphaDLM, including the tokenizer, confidence threshold, argmax token selection, and EMA weights. Each example's recorded forward count is verified against R. The respective correct counts are 10, 9, 56, 265, and 497. This controls the decoder, while the training objectives and initialization histories differ.

Reevaluation of released checkpoints. We evaluate the released TinyGSM MDLM, DUO, and $\mathbb{S}$-FLM checkpoints with the authors' sampling implementation (Deschenaux & Gulcehre, 2026). MDLM and DUO use ancestral sampling at $T = 0.1$. Both $\mathbb{S}$-FLM curves use the spherical backbone with the trained adaptive truncated schedule, with exact velocity at $T = 0.1$ or top-1 velocity at $T = 1$, followed by greedy final decoding. We use EMA weights, seed 1, four GPUs, and batch size 32 per GPU. The plot and table report integer correct counts divided by 1,319, rounded to one decimal in the table.

Published table values. FMLM+ uses greedy posterior refinement with threshold 0.999 from Table 8 of Agarwal et al. (2026). We show Init in the teaser and all three variants in Table 1. DBTM

uses greedy refinement with threshold 0.8 from Table 8 of Tang & Wang (2026). IDLM uses the reported sampling budgets and default ancestral decoder from Table 1 of Li et al. (2026a). These methods retain their published training recipes. FMLM+ Init and the distillation variants incur teacher training in addition to their own training, so this comparison does not match total training compute.

For context, the autoregressive references in Deschenaux & Gulcehre (2026) reach 53.9% with sampling and 63.3% with greedy decoding.

Table 4: **Ancestral accuracy (%) ↑ at 32 NFE.** Checkpoints share the $k = 1$ initialization and 250k total updates. At $T = 0$, clean-token predictions are argmax while reveal positions remain stochastic.

Training parameter	Ancestral ($T = 1$)	Ancestral ($T = 0$)
k=1	8.19	26.00
k=2	10.46	26.61
k=4	15.09	29.80
k=8	21.30	34.95
k=16	32.52	37.07

B.1 Adaptive confidence decoding

Sweeping the confidence threshold lets the model determine its evaluation count (Figure 7, Table 5). At threshold 0.9, $k = 8$ reaches 49.43% at 17.83 NFE, while $k = 1$ reaches 43.06% at 57.59 NFE. The $k = 16$ run attains 40.03% at 5.70 NFE. Token-wise $\alpha = 0.25$ reaches 47.31% at 32.00 NFE, so the sequence-level $k = 8$ point has 2.12 percentage points higher accuracy with 1.79 times fewer evaluations.

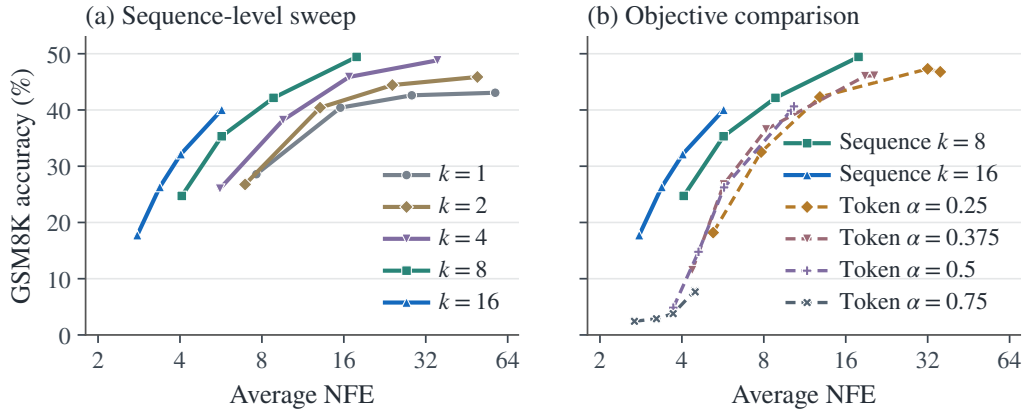


Figure 7: **Accuracy–NFE trade-offs under adaptive confidence decoding.** Left: AlphaDLM (ours) parameter sweep. Right: sequence-level and token-wise objectives. All checkpoints share the $k = 1$ initialization and 250k total updates. Points vary the confidence threshold. Higher accuracy and fewer evaluations are preferable.

Table 5: **Adaptive confidence decoding from a common initialization.** Each entry gives accuracy (%) ↑ / average NFE ↓ on all 1,319 examples. The checkpoints are evaluated at 250k updates after the common sequence-level $k = 1$ initialization at 200k.

Parameter	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$
$k = 1$	28.58 / 7.64	40.41 / 15.54	42.61 / 28.42	43.06 / 57.59
$k = 2$	26.76 / 6.94	40.41 / 13.09	44.43 / 24.12	45.87 / 49.54
$k = 4$	26.00 / 5.61	38.21 / 9.57	45.87 / 16.72	48.82 / 35.32
$k = 8$	24.72 / 4.06	35.33 / 5.69	42.15 / 8.83	49.43 / 17.83
$k = 16$	17.74 / 2.79	26.31 / 3.37	32.15 / 4.03	40.03 / 5.70

C SDAR EXPERIMENTAL DETAILS

Data and training. We fine-tune `JetLM/SDAR-1.7B-Chat` on the same data for all objectives within each domain. For mathematics, we use 272,872 examples from the CoT subset of Nemotron-SFT-Math-v4.[1] We retain assistant content, omit separate reasoning fields, remove lexical overlap candidates with GSM8K and MATH500, and keep complete examples with at most 256 prompt and 1,280 response tokens. For code, we use OpenCodeInstruct (Ahmad et al., 2025). For all setups, we use a batch size of 128 and a learning rate of 5×10^{-6}, and train for 4,000 steps. CE+CAP (Bie et al., 2025) uses a correctness-conditioned entropy regularizer of weight 0.5 to encourage confident correct predictions.

Block-level objective. AlphaDLM applies Equation 12 separately to each four-token response block. For the masked supervised positions M_b in block b, we use

$$s_b = \frac{1}{|M_b|} \sum_{j \in M_b} \log p_{\theta,j}(y_j \mid \text{context}), \qquad \ell_b(k) = \frac{1 - \exp(k s_b)}{k}. \tag{24}$$

The $k = 0$ limit is mean cross-entropy within the block. A block with no active targets contributes zero. We sum block losses within each example, divide by a fixed dataset estimate of the mean number of valid response blocks, and average over examples. Thus, the nonlinear transformation couples targets within a block rather than across the entire response.

Confidence-aware parallel training. CAP adds an entropy penalty on correctly predicted masked targets. Let C contain masked supervised positions whose logit argmax equals the target and let $r_j = \mathrm{softmax}(z_j/0.5)$, where z_j denotes the vocabulary logits. The objective is

$$\mathcal{L}_{\text{CE+CAP}} = \mathcal{L}_{\text{CE}} + \frac{0.5}{|C|} \sum_{j \in C} H(r_j). \tag{25}$$

The entropy term is zero when C is empty. The correctness gate is discrete, while gradients pass through the entropy of the selected distributions. The normalization uses the correct masked positions in each microbatch.

Evaluation. We evaluate accuracy on 1,319 GSM8K problems (Cobbe et al., 2021) and 500 MATH500 problems (Hendrycks et al., 2021), and pass@1 on 164 HumanEval-Instruct problems (Chen et al., 2021) and 427 MBPP-Instruct problems (Austin et al., 2021b). Decoding uses four-token blocks, with confidence thresholds $\tau \in \{0.75, 0.80, 0.85, 0.90, 0.95, 0.97, 0.99\}$ for mathematics and $\tau \in \{0.60, 0.70, 0.80, 0.85, 0.90, 0.95, 0.97, 0.99\}$ for code.

Candidate tokens are selected by argmax ($T = 0$). At each iteration, the decoder accepts remaining positions whose confidence is strictly above the threshold. If none qualifies, it accepts the highest-confidence position. Accepted tokens are not remasked. Each block therefore requires at most four denoising evaluations. Generation uses prefix KV caching and response limits of 1,280 tokens for mathematics and 384 for code, stopping after completing the block containing EOS or the turn-end token.

Tokens per forward (TPF). For a benchmark with N examples, let L_i be the number of returned response tokens and D_i the number of denoising forwards used for example i. We compute

$$\mathrm{TPF} = \frac{\sum_{i=1}^{N} L_i}{\sum_{i=1}^{N} (D_i + 1)}, \tag{26}$$

where the additional forward accounts for prompt prefill once per example. The numerator excludes prompt tokens, the first EOS or stop token, and all subsequent tokens, and respects the response-length limit. Denoising forwards are counted separately for each active example, including all steps in the block containing its stop token. Additional forwards used only to update the KV cache are excluded. We aggregate token and forward counts over the benchmark before taking their ratio, then average the resulting TPF values across runs at each confidence threshold. With four-token blocks, TPF is at most four. Prompt prefill and partially filled final blocks reduce the measured value even when each block requires only one denoising step.

[1] `https://huggingface.co/datasets/nvidia/Nemotron-SFT-Math-v4`.

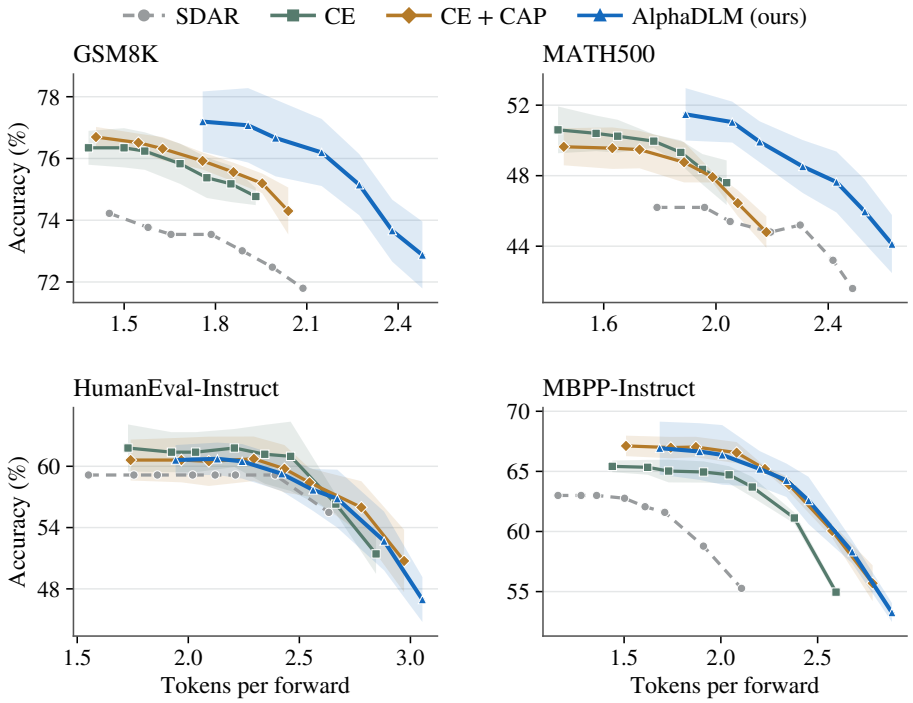


Figure 8: **SDAR accuracy–TPF trade-offs.** AlphaDLM uses $k = 0.8$ for mathematics and $k = 0.4$ for code. Points show means at each confidence threshold, with bands of ± 1 sample standard deviation in accuracy across runs.

Aggregation. At each threshold, we average accuracy and TPF separately across 5 runs. Figure 8 shows the main-text means with ± 1 sample standard deviation in accuracy.

Parameter sweep. Figure 9 compares $k \in \{0, 0.4, 0.8, 1.2, 1.6, 2.0\}$ on mathematics with a learning rate of 5×10^{-6} and a batch size of 128, using 4,000 training steps.

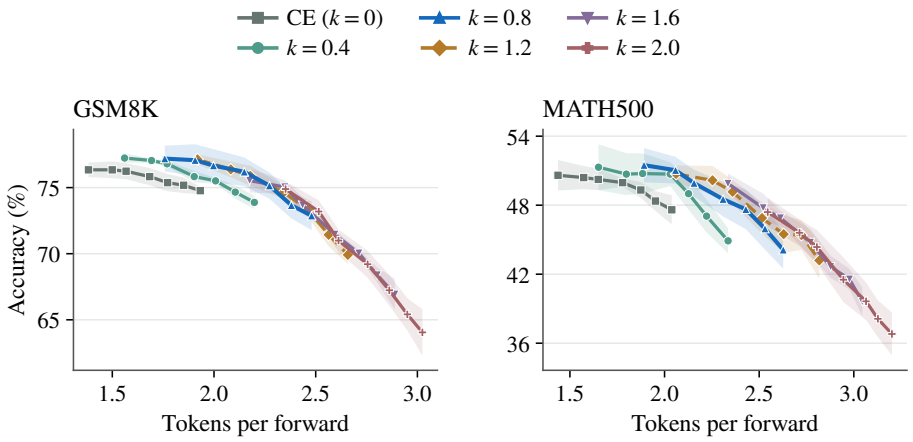


Figure 9: **SDAR parameter sweep for AlphaDLM (ours) and CE.** Bands show ± 1 sample standard deviation in accuracy.

D SENSITIVITY TO TRAINING INITIALIZATION

The main TinyGSM comparison starts from a sequence-level $k = 1$ checkpoint at 200k updates. Here we repeat the objective comparison from a CE-trained checkpoint at the same update count. Both sets of continuations train for another 50k updates. We compare sequence-level $k \in \{2, 4, 8, 16\}$ and token-wise $\alpha \in \{0.25, 0.5, 0.75\}$, using the same architecture, optimizer settings, and fixed-budget decoder. Each evaluation uses EMA weights, exact argmax token selection, confidence threshold 0.999, and exactly 1, 2, 4, 8, or 16 forwards on all 1,319 GSM8K problems. There is one training seed per recipe.

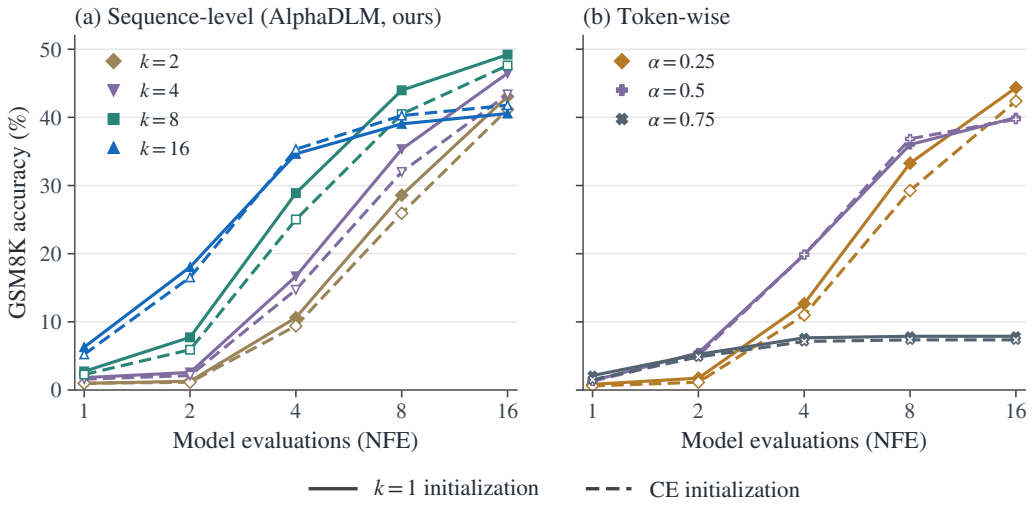


Figure 10: **Objective comparisons under two initializations.** Colors identify the continuation objective. Solid lines start from $k = 1$ and dashed lines from CE, both at 200k updates and evaluated at 250k. Only parameters available under both initializations are shown.

The qualitative advantage of sequence-level training persists under CE initialization (Figure 10). At four NFE, sequence-level $k = 16$ reaches 35.33%, compared with 19.86% for the strongest tested token-wise setting. At sixteen NFE, $k = 8$ reaches 47.61%, compared with 42.38% for the strongest tested token-wise setting. Among the tested sequence-level parameters, $k = 16$ is best at 1–4 NFE and $k = 8$ at 8–16 NFE under both initializations.

The $k = 1$ initialization generally yields higher accuracy for subsequent sequence-level $k = 2, 4, 8$ training. However, CE initialization improves $k = 16$ at 4–16 NFE, by 0.68–1.22 percentage points. Token-wise differences also depend on the loss parameter and evaluation budget (Table 6). These single-seed comparisons support robustness of the objective comparison, rather than a uniform advantage of either initialization.

This ablation covers fixed-budget decoding and the shared parameter grid only. The $k = 1$-start results for token-wise $\alpha = 0.375$, sequence-level $k = 32, 64$, ancestral decoding, and pass@K remain separate experiments.

Table 6: **GSM8K accuracy (%) ↑ under two initializations.** Each paired row uses the same continuation objective and decoding protocol. The final row reports the CE training run continued to 250k, whose 200k checkpoint initializes the CE-start experiments. It is a reference, not a matched-objective comparison with the $k = 1$ continuation.

Objective	Initialization	1 NFE	2	4	8	16
Sequence $k = 2$	$k = 1$	0.99	1.29	10.61	28.58	43.06
Sequence $k = 2$	CE	0.99	1.14	9.40	25.93	41.17
Sequence $k = 4$	$k = 1$	1.82	2.58	16.68	35.33	46.47
Sequence $k = 4$	CE	1.59	2.12	14.71	31.99	43.37
Sequence $k = 8$	$k = 1$	2.73	7.73	28.89	43.97	49.20
Sequence $k = 8$	CE	2.27	5.91	25.02	40.49	47.61
Sequence $k = 16$	$k = 1$	6.29	18.04	34.65	39.04	40.56
Sequence $k = 16$	CE	5.23	16.53	35.33	40.26	41.77
Token $\alpha = 0.25$	$k = 1$	0.83	1.74	12.66	33.28	44.35
Token $\alpha = 0.25$	CE	0.61	1.14	10.99	29.26	42.38
Token $\alpha = 0.5$	$k = 1$	1.36	5.46	19.86	36.01	39.95
Token $\alpha = 0.5$	CE	1.29	5.08	19.86	36.85	39.73
Token $\alpha = 0.75$	$k = 1$	2.12	5.23	7.66	7.88	7.88
Token $\alpha = 0.75$	CE	1.52	4.85	7.13	7.35	7.35
CE	CE	0.45	0.53	5.69	18.50	36.32

E MULTIPLE-ANSWER GENERATION

We study how the training objective affects the benefit of generating several answers to the same problem. We use the three sequence-level checkpoints with $k \in \{1, 8, 16\}$ at 250k training updates, all continued from the common $k = 1$ checkpoint at 200k updates. For each of the 1,319 GSM8K test problems, we generate $n = 32$ answers with independently seeded ancestral sampling. We consider $R \in \{4, 16\}$ model evaluations per answer and temperatures $T \in \{0, 1\}$, giving 12 configurations and 506,496 generated answers in total. At $T = 0$, the clean-token prediction is the argmax, while the ancestral reveal trajectory remains random. All runs use EMA weights, top-$p = 1$, the 512-token sequence limit, and the same program verifier as the single-answer evaluation.

Let c_i denote the number of correct answers among the 32 samples for problem i. We estimate the probability that at least one of K answers is correct using the unbiased pass@K estimator of Chen et al. (2021),

$$\widehat{\text{pass@}K} = \frac{1}{1319} \sum_{i=1}^{1319} \left(1 - \frac{\binom{32-c_i}{K}}{\binom{32}{K}} \right), \qquad K \in \{1, 2, 4, 8, 16, 32\}, \tag{27}$$

where the numerator is zero when $K > 32 - c_i$. Repeated answers are retained. This metric measures verifier-assisted coverage among K attempts, with a total generation budget of KR model evaluations per problem. The sampling repeats use one trained checkpoint for each value of k.

Figure 11 and Table 7 show that the preferred training parameter depends on the number of attempts. With $T = 0$ and $R = 16$, increasing k from 1 to 16 improves pass@1 from 16.83% to 32.24%. At 32 attempts, $k = 8$ reaches 74.22%, compared with 72.71% for $k = 16$ and 68.39% for $k = 1$. At $T = 1$, $k = 16$ has the highest observed pass@K across the tested values of K at both per-answer budgets. The computation view also separates improvements from simply generating more answers. For every tested checkpoint and temperature, using 16 evaluations per answer gives higher coverage than using four evaluations per answer at each shared total budget of 16, 32, 64, and 128 NFE.

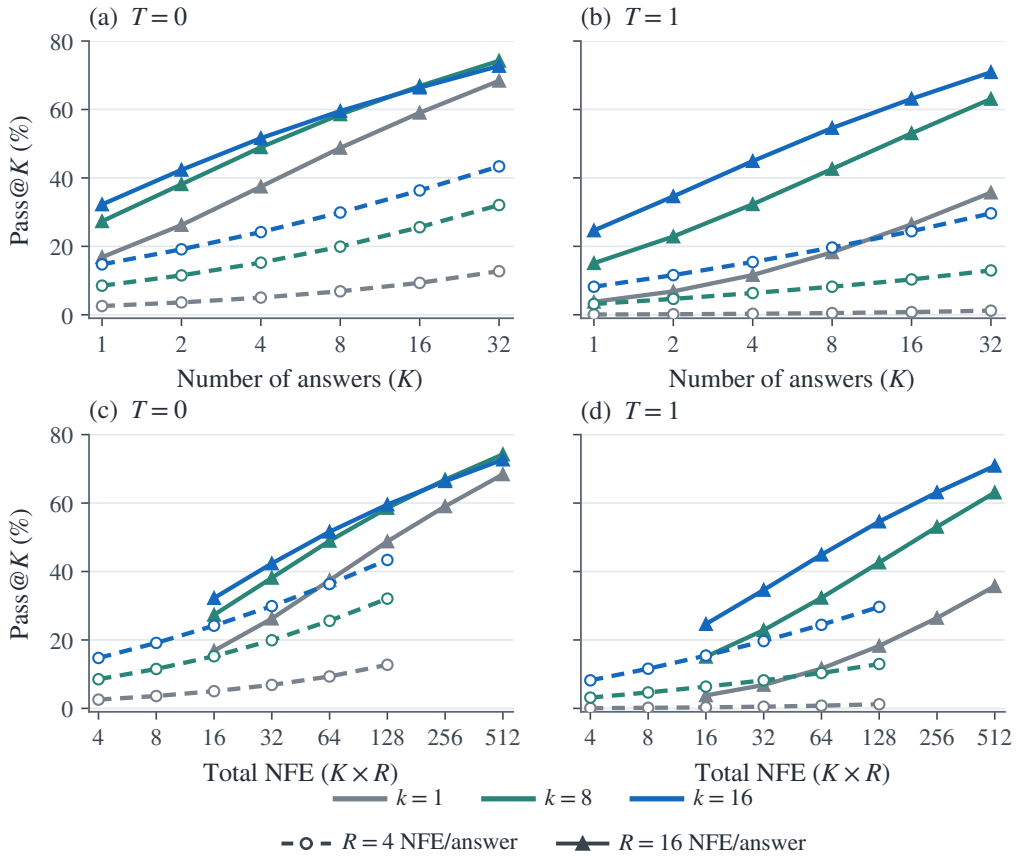


Figure 11: Multiple-answer evaluation on GSM8K with ancestral sampling. The top row shows pass@K against the number of generated answers, and the bottom row shows the same results against total NFE, KR. Colors identify the training parameter k, while the line style and markers identify the per-answer budget R. All curves use 32 sampling attempts per problem. The reveal trajectory remains random at $T = 0$.

Table 7: GSM8K pass@K (%) ↑ estimated from 32 answers per problem. T is the sampling temperature, R is NFE per answer, and k is the training parameter. Bold entries mark the highest value across training parameters for the same T, R, and K.

T	R	k	Pass@1↑	Pass@2↑	Pass@4↑	Pass@8↑	Pass@16↑	Pass@32↑
0	4	1	2.56	3.62	5.03	6.86	9.33	12.74
0	4	8	8.51	11.52	15.23	19.91	25.60	32.07
0	4	16	**14.73**	**19.14**	**24.17**	**29.89**	**36.34**	**43.37**
0	16	1	16.83	26.23	37.45	48.77	59.04	68.39
0	16	8	27.33	38.14	48.95	58.56	**66.85**	**74.22**
0	16	16	**32.24**	**42.35**	**51.62**	**59.55**	66.39	72.71
1	4	1	0.09	0.17	0.30	0.50	0.80	1.21
1	4	8	3.16	4.65	6.36	8.20	10.33	12.96
1	4	16	**8.20**	**11.59**	**15.42**	**19.67**	**24.42**	**29.64**
1	16	1	3.79	6.84	11.62	18.25	26.43	35.71
1	16	8	15.08	22.87	32.31	42.62	53.04	63.08
1	16	16	**24.64**	**34.61**	**44.91**	**54.59**	**63.12**	**70.89**

F Related Work

Diffusion language models and decoding. Masked diffusion models learn token predictions under a corruption process (Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025). Confidence-based decoding controls how many tokens are revealed at each evaluation (Wu et al., 2026), while SDAR combines parallel prediction within blocks with autoregression across blocks (Arriola et al., 2025; Cheng et al., 2025). AlphaDLM changes the training objective and can be used with either decoding structure.

Training for few-step generation. Several methods adapt diffusion language models for generation with fewer model evaluations. T3D (Zhang et al., 2026) combines trajectory self-distillation with a reverse-KL-inspired discriminative objective and analyzes how trajectory supervision reduces conditional token dependence. dParallel (Chen et al., 2026) combines trajectory supervision with entropy minimization on correctly predicted tokens to encourage earlier parallel commitments. CDLM (Kim et al., 2026) distills teacher trajectories into a block-causal student using teacher-distribution matching and consistency objectives. AlphaDLM shares the goal of improving generation under limited decoding budgets, but directly applies sequence-level alpha loss to training examples and their corruptions, without requiring teacher trajectories or distribution matching. Our analysis characterizes how this objective changes the joint target fitted by a factorized denoiser and establishes conditions under which its optimum preserves multiple valid completions while excluding invalid combinations.

Modeling token dependencies. Parallel prediction and its multimodality challenge predate diffusion language models, including non-autoregressive translation (Gu et al., 2018; Huang et al., 2022). For discrete diffusion, Discrete Copula Diffusion (Liu et al., 2025) supplements the denoiser with a generative model that supplies dependency information. EDLM (Xu et al., 2025) introduces a sequence-level energy correction. Kim & Sung (2026) represent the conditional clean distribution through low-rank tensor decompositions, while Bansal & Sanghavi (2025) train a lightweight sampler on top of a frozen diffusion model to approximate joint sampling. These approaches enrich the modeled dependencies or the sampling procedure. AlphaDLM retains the factorized denoiser and changes its training objective. Our analysis asks which joint target this restricted predictor fits and when independent predictions can exclude invalid combinations without retaining only one completion.

Evaluation and baselines. TinyGSM (Liu et al., 2023) is a shared setting for generating mathematical solutions as programs. $\mathbb{S}$-FLM (Deschenaux & Gulcehre, 2026) compares hyperspherical flows with MDLM and DUO (Sahoo et al., 2025). FMLM+ (Agarwal et al., 2026) uses posterior refinement, DBTM (Tang & Wang, 2026) learns a discrete transport map, and IDLM (Li et al., 2026a) distills a pretrained diffusion teacher. Our comparison retains their respective generation procedures. AlphaDLM uses confidence-based unmasking with the token-reveal schedule of FMLM+, while its objective ablations hold the sampler fixed. Appendix B also records TinyGSM results using different computation units.

Objectives and dependence. Generalized cross-entropy interpolates between cross-entropy and mean absolute error in classification (Zhang & Sabuncu, 2018). Li et al. (2026b) study probability-based objectives for autoregressive supervised fine-tuning, including the same alpha-loss family applied to individual token probabilities. They show that downweighting low-probability tokens can help when the pretrained model already has strong task-relevant capabilities, while cross-entropy performs better when these capabilities are weak. We apply alpha loss to the joint probability of masked tokens and analyze how factorization changes the distribution that minimizes this sequence-level objective. Huang et al. (2022) analyze dependence lost by non-autoregressive marginal fitting and interpret alternative objectives through proxy targets. Minka (2005) shows how the divergence shapes a factorized approximation, and Lapidoth & Pfister (2019) establish variational identities for product-distribution optimization. Our analysis connects this viewpoint to denoising objectives, characterizing the selected joint target, distinguishing sequence-level and token-wise fitting, and giving conditions and quantitative bounds for valid product predictions. Recent analyses relate unmasking schedules and parallel-sampling error to total correlation (Zhao & Cai, 2026; Wen et al., 2026). Our characterization concerns the joint target selected by training. Token-wise alpha loss and entropy-regularized SFT provide relevant empirical controls because both also change prediction confidence.